

**RANCANG BANGUN WEB SERVICE DAN WEBSITE
SEBAGAI STORAGE ENGINE DAN MONITORING DATA SENSING
UNTUK BUDIDAYA IKAN AIR TAWAR**

Skripsi

**Disusun untuk memenuhi salah satu syarat
memperoleh gelar Sarjana Komputer**



**Oleh:
Fadhilah Perwira Hadi
3145163442**

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

2021

LEMBAR PERSETUJUAN

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

Nama : Fadhilah Perwira Hadi
No. Registrasi : 3145163442
Program Studi : Ilmu Komputer
Judul : Rancang Bangun Web Service dan Website sebagai
Storage Engine dan Monitoring Data Sensing
untuk Budidaya Ikan Air Tawar.

Menyatakan bahwa proposal ini telah siap diajukan untuk sidang skripsi.

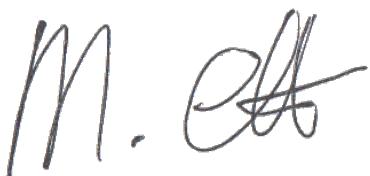
Menyetujui,

Dosen Pembimbing I



Drs. Mulyono, M.Kom.
NIP. 19660517 199403 1 003

Dosen Pembimbing II



Muhammad Eka Suryana, M.Kom.
NIP. 19851223 201212 1 002

Mengetahui,

Koordinator Program Studi Ilmu Komputer



Ir. Fariani Hermin Indiyah, M.T.
NIP. 19600211 198703 2 001

LEMBAR PERNYATAAN

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul **Rancang Bangun Web Service dan Website sebagai Storage Engine dan Monitoring Data Sensing untuk Budidaya Ikan Air Tawar** yang disusun sebagai syarat untuk memperoleh gelar Sarjana komputer dari Program Studi Ilmu Komputer Universitas Negeri Jakarta adalah karya ilmiah saya dengan arahan dari dosen pembimbing.

Sumber informasi yang diperoleh dari penulis lain yang tulisannya telah dipublikasikan yang disebutkan dalam teks skripsi ini, telah dicantumkan dalam Daftar Pustaka sesuai dengan norma, kaidah dan etika penulisan ilmiah.

Jika dikemudian hari ditemukan sebagian besar skripsi ini bukan hasil karya saya sendiri dalam bagian-bagian tertentu, saya bersedia menerima sanksi pencabutan gelar akademik yang saya sanding dan sanksi-sanksi lainnya sesuai dengan peraturan perundang-undangan yang berlaku.

Jakarta, 27 Mei 2021

Fadhilah Perwira Hadi

ABSTRAK

FADHILAH PERWIRA HADI. Rancang Bangun Web service dan Website sebagai *Storage Engine* dan *Monitoring Data Sensing* untuk Budidaya Ikan Air Tawar Menggunakan Metode Scrum. 2021. Di bawah bimbingan Drs. Mulyono, M.Kom dan Muhammad Eka Suryana, M.Kom.

Budidaya Ikan Air Tawar merupakan kegiatan untuk membudidayakan ikan air tawar dari bibit sampai siap dijual. Proses budidaya ikan air tawar memerlukan waktu 3-4 bulan untuk dapat dipanen. Di dalam kegiatan budidaya ikan air tawar diperlukan untuk mengecek indikator air untuk menentukan kelayakan air kolam untuk kehidupan ikan. Skripsi ini bertujuan untuk membangun suatu sistem untuk menyimpan data-data indikator air kolam yang dihasilkan oleh sensor dan dikirim ke sistem, serta untuk memonitoringnya. Web service sangat dibutuhkan di dalam pengembangan sistem ini dikarenakan banyaknya data *sensing* yang dikirimkan dari masing masing kolam, sehingga membutuhkan web service sebagai penghubungnya. Data yang digunakan untuk pengujian sistem ini menggunakan data *dummy* yang format datanya menyesuaikan format data Arduino, dikarenakan *device sensing* yang nanti akan dikembangkan menggunakan Arduino. Sistem informasi ini dikembangkan menggunakan metode Scrum, *microframework* Lumen sebagai *back-end* web service-nya serta menggunakan Bootstrap, PHP, Javascript, Ajax dan JQuerry sebagai *back-end* dan *front-end* untuk websitenya. Pada akhir pengembangan dilakukan UAT (User Acceptance Test) menggunakan metode *black box testing* yaitu pengujian yang hanya menguji fungsional sistemnya saja. Berdasarkan hasil UAT (User Acceptance Test) yang dilakukan, user menyatakan sistem yang sudah dirancang telah lulus uji dan siap dipakai secara fungsional.

Kata Kunci: Budidaya Ikan Air Tawar, Web service, Scrum, *microframework* Lumen, Arduino, data *sensing*, data *dummy*, UAT, *black box testing*.

ABSTRACT

FADHILAH PERWIRA HADI. *Design and Build Web services and Websites as a Storage Engine and Monitoring Data Sensing for Freshwater Fish Cultivation Using the Scrum Method.* 2021. Under the guidance of Drs. Mulyono, M. Kom and Muhammad Eka Suryana, M. Kom.

Freshwater Fish Cultivation is an activity for cultivating fish fresh water from seed to ready for sale. The process of freshwater fish cultivation takes 3-4 months to be harvested. In freshwater fish farming activities, it is necessary to check water indicators to determine the suitability of pond water for fish life. This thesis aims to build a system for store pool water indicator data generated by sensors and sent to the system, and to monitor it. Web service is very much needed in The development of this system is due to the large number of sensing data sent from each pool, thus requiring a web service as a liaison. The data used for testing this system uses dummy data whose data format adapts to the Arduino data format, because the sensing device will later be developed using Arduino. This information system developed using the Scrum method, the Lumen microframework as the backend of the web service and using Bootstrap, PHP, Javascript, Ajax and JQuery as the back-end and front-end for the website. At the end of development UAT (User Acceptance Test) is carried out using the black box testing method, namely: testing that only tests the functionality of the system. Based on UAT (User Acceptance Test), the system that has been designed is declared to have passed the test and is ready functional use.

Keywords: *Freshwater Fish Cultivation, Web service, Scrum, Lumen microframework, Arduino, data sensing, dummy data, UAT, black box testing.*

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Allah SWT atas segala limpahan Rahmat dan Hidayah-Nya maka karya ilmiah ini berhasil diselesaikan. Jenis penelitian yang dipilih adalah penelitian Rekayasa dan Aplikasi Produk dengan judul Rancang Bangun Web Service dan Website sebagai *Storage Engine* dan Monitoring Data Sensing untuk Budidaya Ikan Air Tawar. Keberhasilan dalam menyusun skripsi ini tidak lepas dari bantuan berbagai pihak yang mana dengan tulus dan ikhlas memberikan saran dan masukan yang bermanfaat dalam proses penyusunan skripsi ini, oleh karena itu dalam kesempatan ini dengan kerendahan hati, penulis mengucapkan terima kasih kepada:

1. Ibu Ir. Fariani Hermin Indiyah, M.T., selaku Koordinator Program Studi Ilmu Komputer FMIPA UNJ,
2. Bapak Drs. Mulyono, M.Kom, selaku dosen pembimbing I yang telah memberikan bantuan, masukan, dan saran baik secara konten maupun penulisan,
3. Bapak Muhammad Eka Suryana, M.Kom, selaku dosen pembimbing II sekaligus pembimbing akademik saya yang telah memberikan bantuan, masukan, dan saran baik secara konten maupun penulisan,
4. Seluruh Dosen Program Studi Ilmu Komputer FMIPA UNJ atas ilmu dan bimbingannya selama penulis berkuliah di Ilmu Komputer FMIPA UNJ,
5. Kedua orang tua penulis yang selama ini telah senantiasa sabar membimbing, memberikan semangat, mengingatkan, dan mendo'akan penulis,
6. Teman-teman Ilmu Komputer angkatan 2016 yang senantiasa menemani, memberikan semangat, dan motivasi dari semenjak awal dunia perkuliahan,

Kelak penulis siap menjadi saksi atas kebaikan-kebaikan yang telah diberikan. Akhir kata, teriring permintaan maaf apabila terdapat kesalahan maupun kekeliruan

dalam penulisan skripsi ini. Besar harapan agar skripsi ini dapat bermanfaat. Terima kasih.

Jakarta, 27 Mei 2021

Fadhilah Perwira Hadi

DAFTAR ISI

KATA PENGANTAR	vi
DAFTAR ISI	x
DAFTAR GAMBAR	xiii
DAFTAR TABEL	xiv
I PENDAHULUAN	1
1.1 Latar Belakang Masalah	1
1.2 Rumusan Masalah	5
1.3 Pembatasan Masalah	5
1.4 Tujuan Penelitian	6
1.5 Manfaat Penelitian	6
II KAJIAN PUSTAKA	7
2.1 Budidaya ikan air tawar	7
2.2 Testing	7
2.2.1 <i>User Acceptance Test (UAT)</i>	8
2.2.2 Metode <i>Black Box Testing</i>	9
2.3 Data sensor	9
2.4 Arduino	11
2.5 <i>Storage Engine</i>	11
2.5.1 InnoDB	12
2.6 Variabel <i>dummy</i>	13
2.7 Teori pH	13
2.8 <i>Dissolved Oxygen</i>	14
2.9 Suhu	14
2.10 Generate dan Post Data	15

2.10.1	Menentukan <i>Range</i> Dari Variabel <i>Dummy</i> Untuk Data Indo-kator Air	16
2.10.2	Membuat Collection di POSTMAN	17
2.10.3	Membuat <i>Request</i> Pada Collection	17
2.10.4	Melakukan Generate dan Post Data	18
2.11	REST API	19
2.11.1	HTTP GET	20
2.11.2	Membaca sebuah HTTP response	21
2.11.3	HTTP POST	23
2.11.4	Protokol semantik HTTP	25
2.12	Lumen Framework	30
2.12.1	<i>Routing</i>	30
2.12.2	<i>Controllers</i>	37
2.12.3	<i>Views</i>	40
2.13	Scrum	41
2.13.1	Nilai-nilai Scrum	42
2.13.2	Tim Scrum	43
2.13.3	Aktivitas-aktivitas Scrum (<i>Scrum Events</i>)	47
2.13.4	Komponen Scrum (<i>Scrum Artifacts</i>)	50
III	Metode Penelitian	53
3.1	Pengumpulan Data	55
3.2	Analisis Kebutuhan	55
3.2.1	Kebutuhan Perangkat Keras	55
3.2.2	Kebutuhan Perangkat Lunak	56
3.3	Perancangan sistem	56
3.3.1	Produk Backlog	56
3.3.2	<i>Sprint Backlog</i>	58
3.3.3	<i>Sprint</i>	59

3.3.4	<i>Daily Scrum</i>	59
3.3.5	<i>Deploy</i>	59
3.4	UAT (<i>User Acceptance Test</i>)	59
IV Hasil Dan Pembahasan		61
4.1	Perancangan Sistem	61
4.1.1	<i>Sprint reports iteration</i>	61
4.1.2	<i>Sprint-6</i>	77
4.1.3	Hasil akhir <i>sprint</i> keseluruhan	86
4.2	<i>Testing</i>	105
4.2.1	Hasil UAT (<i>User Acceptance Test</i>)	105
4.2.2	Kesimpulan Pengujian	107
4.3	Pembahasan	108
V KESIMPULAN DAN SARAN		109
5.1	Kesimpulan	109
5.2	Saran	109
DAFTAR PUSTAKA		112
LAMPIRAN		113
A Transkrip Wawancara		113
B Kode Controller Fungsi Registrasi Kolam		115
C Kode Model Fungsi Registrasi Kolam		116
D Beta Testing		117
E User Acceptance Test		118

DAFTAR GAMBAR

Gambar 2.1	Collection Postman	17
Gambar 2.2	Membuat Request	18
Gambar 2.3	Konfigurasi Request	19
Gambar 2.4	HTTP response data	21
Gambar 3.1	Desasin Penelitian	54
Gambar 4.1	Desain Database <i>Sprint-1</i>	62
Gambar 4.2	Desain MVC <i>sprint-1</i>	62
Gambar 4.3	Registrasi Kolam web service (<i>input</i>)	63
Gambar 4.4	Registrasi Kolam web service (<i>output</i>)	63
Gambar 4.5	Desain Database <i>Sprint-2</i>	65
Gambar 4.6	Desain MVC <i>sprint-2</i>	66
Gambar 4.7	Login Admin web service (<i>input</i>)	67
Gambar 4.8	Login Admin web service (<i>output</i>)	67
Gambar 4.9	Login User web service (<i>input</i>)	68
Gambar 4.10	Login User web service (<i>output</i>)	68
Gambar 4.11	Tampilan Login	69
Gambar 4.12	Tampilan registrasi kolam	69
Gambar 4.13	Tampilan registrasi user	70
Gambar 4.14	Desain MVC <i>sprint-3</i>	72
Gambar 4.15	Tampilan list kolam	72
Gambar 4.16	Update kolam web service (<i>input</i>)	73
Gambar 4.17	Update kolam web service (<i>output</i>)	73
Gambar 4.18	Tampilan edit kolam	74
Gambar 4.19	Tampilan <i>delete</i> kolam	74
Gambar 4.20	Desain Database <i>Sprint-6</i>	78
Gambar 4.21	POST data sensor web service (<i>input</i>)	78

Gambar 4.22 POST data sensor web service (<i>output</i>)	79
Gambar 4.23 Desain Database <i>Sprint-7</i>	81
Gambar 4.24 Desain MVC <i>sprint-7</i>	81
Gambar 4.25 List histori pembacaan sensor	82
Gambar 4.26 <i>Min, max</i> dan rata-rata sensor ph	82
Gambar 4.27 <i>Min, max</i> dan rata-rata sensor DO	83
Gambar 4.28 <i>Min, max</i> dan rata-rata sensor suhu	83
Gambar 4.29 Tampilan form tambah ikan	84
Gambar 4.30 Tampilan list info ikan	84
Gambar 4.31 Chart ph	85
Gambar 4.32 Chart DO	85
Gambar 4.33 Chart suhu	86
Gambar 4.34 Desain Database Sistem	88
Gambar 4.35 Desain MVC Sistem	88
Gambar 4.36 Registrasi Kolam web service (<i>input</i>)	92
Gambar 4.37 Registrasi Kolam web service (<i>output</i>)	93
Gambar 4.38 Login Admin web service (<i>input</i>)	93
Gambar 4.39 Login Admin web service (<i>output</i>)	94
Gambar 4.40 Login User web service (<i>input</i>)	94
Gambar 4.41 Login User web service (<i>output</i>)	95
Gambar 4.42 Tampilan Login	95
Gambar 4.43 Tampilan registrasi kolam	96
Gambar 4.44 Tampilan registrasi user	96
Gambar 4.45 Tampilan list kolam	97
Gambar 4.46 Update kolam web service (<i>input</i>)	97
Gambar 4.47 Update kolam web service (<i>output</i>)	98
Gambar 4.48 Tampilan edit kolam	98
Gambar 4.49 Tampilan <i>delete</i> kolam	99
Gambar 4.50 POST data sensor web service (<i>input</i>)	99

Gambar 4.51 POST data sensor web service (<i>output</i>)	100
Gambar 4.52 List histori pembacaan sensor	101
Gambar 4.53 <i>Min</i> , <i>max</i> dan rata-rata sensor ph	101
Gambar 4.54 <i>Min</i> , <i>max</i> dan rata-rata sensor DO	102
Gambar 4.55 <i>Min</i> , <i>max</i> dan rata-rata sensor suhu	102
Gambar 4.56 Tampilan form tambah ikan	103
Gambar 4.57 Tampilan list info ikan	103
Gambar 4.58 Chart ph	104
Gambar 4.59 Chart DO	104
Gambar 4.60 Chart suhu	105

DAFTAR TABEL

Tabel 3.1	Produk Backlog	57
Tabel 3.2	Skenario Pengujian Admin	60
Tabel 3.3	Skenario Pengujian User	60
Tabel 4.1	<i>Sprint-1</i>	61
Tabel 4.2	Sprint-2	64
Tabel 4.3	<i>Sprint-3</i>	71
Tabel 4.4	<i>Sprint-4</i>	75
Tabel 4.5	<i>Sprint-6</i>	77
Tabel 4.6	<i>Sprint-7</i>	80
Tabel 4.7	Product Backlog Tested	87
Tabel 4.8	Routing Table	89
Tabel 4.9	Pengujian Login admin dan user	106
Tabel 4.10	Pengujian Fungsi dan Halaman Admin	106
Tabel 4.11	Pengujian Fungsi dan Halaman User	107

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Budidaya ikan air merupakan salah satu bentuk budidaya perairan yang khusus membudidayakan ikan, baik itu di kolam atau tempat lainnya guna menghasilkan bahan pangan, ikan hias, dan rekreasi (pemancingan) [infishta (2019)]. Salah satu bidang budidaya ikan yaitu budidaya ikan air tawar. Pemeliharaan ikan air tawar pada umumnya jarang dilakukan hanya untuk memelihara satu jenis ikan saja, tetapi pada umumnya merupakan pemeliharaan campuran hal ini disebabkan di dalam kolom sebenarnya sudah terdapat dengan sendirinya berbagai macam makanan untuk berbagai jenis ikan, walaupun demikian kita harus memperhatikan jenis ikan apa yang cocok sebagai peliharaan pokok dan peliharaan tambahan.

Dalam pembudidayaan ikan air tawar perlu untuk melakukan pengecekan terhadap indikator air untuk kelayakan habitat ikan air tawar yang akan dibudidayakan. Untuk indikator pengecekan yang dibutuhkan yaitu pengukuran kadar oksigen dalam air, pengukuran kadar ph dalam air, DO (*Dissolved Oxygen*), Suhu, dan Ammonia. Indikator air tersebut sangat berpengaruh terhadap kelayakan habitat ikan air tawar [Meilinda Pramleonita (2018)]. Perlunya pengecekan keempat indikator tersebut secara intensif maka diperlukan sistem untuk melakukan itu.

Perancangan sistem ini dilakukan berdasarkan kebutuhan pengecekan indikator air untuk budidaya ikan air tawar yaitu pengecekan kadar ph, DO (*Dissolved Oxygen*), dan suhu. Untuk pengecekan ammonia tidak terdapat didalam sistem dikarenakan pengecekan ammonia masih menggunakan laksus dan belum ada sensor untuk pengukuran tersebut. Perancangan sistem yang akan dilakukan hanya membuat web service dan website dengan konsep menyerupai IoT yaitu dengan simulasi mengirimkan data sensor indikator air dari luar dengan aplikasi POSTMAN menuju sistem ini. Untuk data indikator ph, DO, dan suhu akan digenerate dan dikirimkan

menggunakan aplikasi POSTMAN kedalam sistem.

Internet of Things (IoT) adalah IoT konsep yang menghubungkan semua perangkat ke internet dan memungkinkan perangkat IoT berkomunikasi satu sama lain melalui internet. IoT adalah jaringan raksasa dari perangkat yang terhubung yang mengumpulkan dan membagikan data tentang bagaimana suatu perangkat tersebut digunakan dan lingkungan di mana perangkat tersebut dioperasikan [Sidiq (2018)]. IoT dalam kehidupan modern diterapkan untuk berbagai sektor yaitu monitoring lingkungan, pengelolaan infrastruktur, sensor peralatan, bidang kesehatan, automasi gedung dan perumahan. Dalam sektor monitoring lingkungan, IoT dapat diterapkan dalam pembudidayaan ikan air tawar, karena dalam pembudidayaan ikan air tawar memerlukan sarana untuk memonitoring habitat ikan.

IoT memiliki cara kerja yang mengacu pada tiga elemen pada arsitektur IoT yaitu *embedded device*, *networking*, dan *back-end system (application)*. *embedded device* merupakan sebuah perangkat yang sudah ditanamkan sebuah sistem komputer sebagai pengendali perangkat yaitu *embedded system*. Pada *embedded device* program telah dirancang khusus dan ditentukan sebelumnya, biasanya digunakan pada sistem mekanik maupun listrik dengan skala besar termasuk implementasi pada perangkat IoT. *Networking* pada IoT merupakan sebuah infrastruktur jaringan yang berfungsi sebagai penghubung antara *embedded device* dengan *back-end system(application)*. *Back-end system* IoT merupakan sebuah aplikasi yang dapat menerima dan menyimpan data yang dikirimkan oleh *embedded device*. Salah satu *back-end system* dapat digunakan adalah web service. Web service dapat mempermudah proses pengiriman, penyimpanan, dan pengambilan data pada data *storage*.

Web services adalah mekanisme komunikasi dua aplikasi atau mesin terlepas dari arsitektur dan teknologi yang digarisbawahi [Rumagit (2019)]. Metode web service ada dua macam yaitu SOAP dan REST. SOAP (*Simple Object Access Protocol*) adalah standar untuk bertukar pesan-pesan berbasis XML melalui jaringan komputer atau sebuah jalan untuk program yang berjalan pada suatu sistem operasi (OS) untuk berkomunikasi dengan program pada OS yang sama maupun ber-

beda dengan menggunakan HTTP dan XML sebagai mekanisme untuk pertukaran data, maka SOAP dapat berkomunikasi dengan berbagai aplikasi meskipun terdapat perbedaan sistem operasi, teknologi, dan bahasa pemrogramannya [Feridi (2016)]. REST(*REpresentational State Transfer*) merupakan standar arsitektur komunikasi berbasis web yang sering diterapkan dalam pengembangan layanan berbasis web. Umumnya menggunakan HTTP(*Hypertext Transfer Protocol*) sebagai *protocol* untuk komunikasi data. REST pertama kali diperkenalkan oleh Roy Fielding pada tahun 2000 [Feridi (2019)].

Perancangan web service untuk IoT *device* ini bertujuan untuk menerima dan menyimpan data *output* dari IoT *device*. Dalam penelitian ini web service dibutuhkan karena dalam memonitoring kadar dalam air membutuhkan sarana yang menyediakan dan menyimpan data secara akurat dari *embedded device*. Perancangan web service ini hanya berfokus pada penerimaan dan penyimpanan data-data *sensing* yang dihasilkan IoT *device*. Manfaat yang nantinya akan dihasilkan dari web service ini yaitu para petugas atau karyawan yang bertugas biasanya melakukan pemantauan di lokasi, dapat menggunakan web service ini untuk pemantauan jarak jauh secara efektif.

Perancangan web service untuk IoT *device* ini akan menggunakan *microframework* Lumen. Microframework Lumen dapat menghandle hingga 1900 *request/second* dibandingkan dengan microframework sejenis yaitu Slim yang hanya dapat menghandle 1800 *request/second* [Fauzi (2017)]. Microframework Lumen memiliki *library* untuk UI yaitu *views* yang merupakan *library* untuk menampilkan halaman, sehingga mempermudah dalam perancangan web service ini.

Penelitian terkait dalam bidang Monitoring data *sensing* pada budidaya ikan air tawar sudah dilakukan oleh Fardian dalam penelitian yang berjudul "Rancang Bangun Prototipe Pemantauan Kadar pH dan Kontrol Suhu Serta Pemberian Pakan Otomatis pada Budidaya Ikan Lele Sangkuriang Berbasis IoT". Akan tetapi dalam penelitian tersebut data *sensing* dikirimkan ke dalam sebuah IoT *cloud* yaitu *ubidots cloud* lalu dikirimkan ke email atau sms, tidak terdapat back-end systemnya [Al Qalit (2017)]. Kemudian oleh Rian Bayu Pembudi dalam penelitian yang berjudul

"Implementasi Node Sensor untuk Sistem Pengamatan pH Air Pada Budidaya Ikan Air Tawar". Dalam penelitian tersebut data *sensing* langsung dikirimkan ke PC atau Laptop melalui perantara *Access Point* dengan format data JSON. format data JSON akan digunakan dalam penelitian ini sebagai format data Dalam web service [Rian Bayu Pembudi (2018)]. Kemudian oleh Kabul Rizalul Haqim dalam penelitian yang berjudul "Perancangan Web Monitoring Dan Kontroling Aquaponic Untuk Budidaya Ikan Lele Berbasis Internet of Things". Dalam penelitian tersebut data *sensing* dikirimkan ke Firebase lalu ditampilkan dalam website. Akan tetapi Firebase merupakan platform yang berbayar dan Firebase merupakan *host* milik google, sehingga tidak mungkin untuk merecovery data yang hilang [Kabul Rizalul Haqim (2018)].

Penelitian dalam bidang perancangan web service juga sudah dilakukan oleh Ali Firdaus yang berjudul "Rancang Bangun Sistem Informasi Perpustakaan Menggunakan web service Pada Jurusan Teknik Komputer POLSRI". Dalam penelitian tersebut perancangan Web service menggunakan metode REST API. Pada Sistem REST API menggunakan method PUT, GET, POST dan DELETE serta menggunakan operasi database atau *DML(Data Manipulating Language)* yaitu CRUD(CREATE, READ, UPDATE, DELETE). Untuk format data yang digunakan dalam operasi web service yaitu JSON. Semua itu akan dipakai dalam penelitian ini sebagai bagian dari perancangan web service [Adi Firdaus (2019)]. Kemudian oleh Muhamman Femy Mulya yang berjudul "Analisis dan Perancangan Sistem *Mediation* dengan Protokol SOAP pada web service untuk Mengintegrasikan Antar Sistem Informasi yang Berbeda *Platform*". Dalam Penelitian tersebut perancangan web service menggunakan metode SOAP. Pada Metode SOAP tersebut data dipost ke server oleh HTTP SOAP *Request* dan diterima oleh HTTP SOAP *Response*. Karena data yang diterima beda platform maka diperlukan *mapping* data terlebih dahulu dan format datanya yaitu XML. Akan tetapi dalam penelitian ini kita akan menggunakan REST API, sehingga yang dapat diambil hanya HTTP *Request* dan HTTP *Response* saja [Muhamman Femy Mulya (2017)]. Salah satu upaya untuk mengembangkan sistem monitoring untuk budidaya ikan air tawar yaitu dengan me-

rancang suatu sistem back-end yang aman dan hemat biaya yaitu sebuah web service.

Dalam penelitian ini kita tidak akan menggunakan layanan atau aplikasi pihak ketiga seperti ubidots cloud dan firebase, kemudian tidak mengirim data *sensing* langsung ke *device* tujuan tanpa *back-end system*. Penelitian ini akan membuat *back-end system* yaitu sebuah web service.

Sistem ini dirancang dengan tujuan untuk memudahkan pengecekan serta merekap indikator air dalam kegiatan budidaya ikan air tawar. Web service sangat dibutuhkan di dalam pengembangan sistem ini dikarenakan banyaknya data *sensing* yang akan dikirimkan dari berbagai kolam, sehingga web service berfungsi sebagai penghubung dari data *sensing* yang dikirimkan dari berbagai kolam oleh *embedded device* di masing-masing kolam.

1.2 Rumusan Masalah

Berdasarkan fokus penelitian di atas didapat rumusan masalah penelitian ini yaitu:

1. Bagaimana konsep perancangan web service sebagai penerima dan penyimpanan data-data *sensing* (*storage engine*) dari IoT *device*?

1.3 Pembatasan Masalah

Adapun batasan-batasan masalah yang digunakan agar lebih terarah dan sesuai dengan yang diharapkan serta terorganisasi dengan baik adalah:

1. Web service yang dirancang hanya untuk menerima dan menyimpan data-data *sensing* (*storage engine*) dari IoT *device*.
2. Perancangan web service menggunakan *Microframework* Lumen.
3. Web service berjalan secara lokal di lokasi tempat budidaya.
4. Data sistem menggunakan data *dummy*.

5. Pengujian sistem hanya pengujian fungsional saja dengan metode *black box testing*.

1.4 Tujuan Penelitian

Penelitian yang dilakukan bertujuan untuk merancang sebuah web service sebagai penerima dan penyimpanan data-data *sensing (storage engine)* dari IoT *device* agar dapat mempermudah pengecekan data kandungan unsur-unsur dalam air.

1.5 Manfaat Penelitian

Hasil penelitian ini diharapkan dapat memberi manfaat, yaitu:

1. Bagi pembudidaya ikan air tawar

Hasil perancangan web service ini dapat digunakan untuk mengecek kandungan unsur-unsur dalam air yang tadinya harus mengecek langsung ke tempat budidaya menjadi pengecekan dari jarak jauh dan dapat dilakukan secara intensif. Web service yang dirancang diharapkan dapat meminimalisir ketidakefektifan dan kemungkinan terjadinya *human error* dalam pembudidayaan ikan air tawar.

2. Bagi Penulis

Hasil perancangan web service ini dapat menambah wawasan dalam pengembangan web service sebagai penerima dan penyimpanan data dan dapat digunakan sebagai media pembelajaran untuk kedepannya.

BAB II

KAJIAN PUSTAKA

2.1 Budidaya ikan air tawar

Budidaya ikan air merupakan salah satu bentuk budidaya perairan yang khusus membudidayakan ikan, baik itu di kolam atau tempat lainnya guna menghasilkan bahan pangan, ikan hias, dan rekreasi (pemancingan) [infishta (2019)]. Salah satu bidang budidaya ikan yaitu budidaya ikan air tawar. Pemeliharaan ikan air tawar pada umumnya jarang dilakukan hanya untuk memelihara satu jenis ikan saja, tetapi pada umumnya merupakan pemeliharaan campuran hal ini disebabkan di dalam kolom sebenarnya sudah terdapat dengan sendirinya berbagai macam makanan untuk berbagai jenis ikan, walaupun demikian kita harus memperhatikan jenis ikan apa yang cocok sebagai peliharaan pokok dan peliharaan tambahan.

Dalam pembudidayaan ikan air tawar perlu untuk melakukan pengecekan terhadap indikator air untuk kelayakan habitat ikan air tawar yang akan dibudidayakan. Untuk indikator pengecekan yang dibutuhkan yaitu pengukuran kadar oksigen dalam air, pengukuran kadar ph dalam air, DO (*Dissolved Oxygen*), Suhu, dan Ammonia. Indikator air tersebut sangat berpengaruh terhadap kelayakan habitat ikan air tawar [Meilinda Pramleonita (2018)].

2.2 Testing

Menurut Singh, *testing* adalah proses untuk memeriksa atau mengevaluasi sistem atau komponen sistem secara manual atau terotomatisasi yang bertujuan untuk melakukan verifikasi bahwa sistem tersebut memenuhi persyaratan tertentu atau untuk mengidentifikasi perbedaan antara *expected result* dan *actual result* [Chehal (2012)].

Sedangkan menurut Lewis, *software testing* adalah aktivitas menjalankan se-

rangkaian eksekusi yang dinamis pada program *software* setelah *source code software* tersebut telah dikembangkan. *Software testing* dilakukan untuk menemukan dan memperbaiki sebanyak mungkin potensi kesalahan sebelum *software* tersebut digunakan oleh pelanggan atau *end user* [Lewis (2009)].

Dari definisi di atas, testing merupakan aktivitas atau proses memeriksa dan mengevaluasi sistem dengan tujuan untuk menemukan kesalahan pada sistem tersebut.

2.2.1 *User Acceptance Test (UAT)*

Menurut Perry, William E, *User Acceptance Testing* (UAT) merupakan pengujian yang dilakukan oleh end-user dimana user tersebut adalah staff/karyawan perusahaan yang langsung berinteraksi dengan sistem dan dilakukan verifikasi apakah fungsi yang ada telah berjalan sesuai dengan kebutuhan/fungsinya. Setelah dilakukan sistem testing, *acceptance testing* menyatakan bahwa sistem perangkat lunak memenuhi persyaratan [Perry (2006)].

Setelah dilakukan *system testing*, *acceptance testing* menyatakan bahwa sistem *software* memenuhi persyaratan. *Acceptance testing* merupakan pengujian yang dilakukan oleh pengguna yang menggunakan teknik pengujian *black box* untuk menguji sistem terhadap spesifikasinya. Pengguna akhir bertanggung jawab untuk memastikan semua fungsionalitas yang relevan telah diuji.

Pengujian penerimaan pengguna (UAT) adalah fase terakhir dari proses pengujian perangkat lunak. Selama UAT, perangkat lunak perangkat lunak diuji untuk memastikan tugas-tugas apakah sudah sesuai dengan spesifikasinya. UAT adalah salah satu prosedur proyek perangkat lunak final dan paling penting yang harus terjadi sebelum perangkat lunak tersebut dikembangkan dan diluncurkan ke pasar. UAT juga dikenal sebagai pengujian beta, pengujian aplikasi atau pengujian pengguna akhir [Cimperman (2006)].

2.2.2 Metode *Black Box Testing*

Pengujian *blackbox* adalah metode pengujian yang berfokus pada apakah unit program memenuhi kebutuhan (*requirement*) yang disebutkan dengan spesifikasi. Pada *blackbox testing*, cara pengujian hanya dilakukan dengan menjalankan atau meng-eksekusi unit atau modul, kemudian diamati apakah hasil dari unit itu sesuai dengan proses bisnis yang diinginkan [Al Fatta (2007)].

Pengujian *blackbox* (*blackbox testing*) adalah salah satu metode pengujian perangkat lunak yang berfokus pada sisi fungsionalitas, khususnya pada *input* dan *output* aplikasi (apakah sudah sesuai dengan apa yang diharapkan atau belum). Tahap pengujian merupakan salah satu tahap yang harus ada dalam sebuah siklus pengembangan perangkat lunak [Iskandaria (2012)].

Adapun ciri-ciri dari metode *black box testing* itu sendiri, diantaranya sebagai berikut:

1. *Black box testing* berfokus pada kebutuhan fungsionalitas pada *software*, berdasarkan pada spesifikasi kebutuhan dari *software*.
2. *Black box testing* bukan teknik alternatif dari pada *white box testing*. Lebih dari pada itu, ia merupakan pendekatan pelengkap dalam mencakup *error* dengan kelas yang berbeda dari metode *white box testing*.
3. *Black box testing* melakukan pengujian tanpa pengetahuan detail struktur internal dari sistem atau komponen yang dites. Juga disebut sebagai *behavioral testing*, *specification-based testing*, *input/output testing* atau *functional testing* [Shihab (2011)].

2.3 Data sensor

Data sensor atau dapat disebut juga data *sensing* merupakan *output* dari perangkat yang mendekksi dan merespon beberapa jenis input dari lingkungan fisik.

Output dapat digunakan untuk memberikan informasi atau masukan ke sistem lain atau untuk menjalankan suatu proses.

Sensor dapat digunakan untuk mendeteksi hampir semua elemen fisik. Berikut adalah beberapa contoh sensor, hanya untuk memberikan gambaran tentang jumlah dan keragaman aplikasinya:

- Akselerometer untuk mendeteksi perubahan akselerasi gravitasi di perangkat yang terpasang, seperti smartphone atau pengontrol game, untuk menentukan akselerasi, kemiringan, dan getaran.
- Sebuah fotosensor untuk mendeteksi keberadaan cahaya tampak, transmisi inframerah (IR) dan atau energi ultraviolet (UV).
- *Charge-coupled device* (CCD) untuk menyimpan dan menampilkan data untuk gambar sedemikian rupa sehingga setiap piksel diubah menjadi muatan listrik, yang intensitasnya terkait dengan warna dalam spektrum warna.
- Sensor jaringan pintar dapat menyediakan data waktu nyata tentang kondisi jaringan, mendeteksi pemadaman, kesalahan, dan memuat serta memicu alarm.

Jaringan sensor nirkabel menggabungkan transduser khusus dengan infrastruktur komunikasi untuk memantau dan merekam kondisi di lokasi yang beragam. Parameter yang biasanya dipantau meliputi suhu, kelembaban, tekanan, arah dan kecepatan angin, intensitas penerangan, intensitas getaran, intensitas suara, tegangan saluran listrik, konsentrasi bahan kimia, tingkat polutan, dan fungsi vital tubuh.

Data sensor merupakan komponen penting di lingkungan Internet of Things (IoT). Dalam skenario IoT, hampir semua entitas dilengkapi dengan pengenal unik (UID) dan kapasitas untuk mentransfer data melalui jaringan. Sebagian besar data yang dikirimkan adalah data sensor [ivy Wigmore (2015)].

2.4 Arduino

Arduino adalah kit elektronik atau papan rangkaian elektronik *open sorce* yang di dalamnya terdapat komponen utama, yaitu sebuah chip mikrokontroler dengan jenis AVR dari perusahaan Atmel. Mikrokontroler itu sendiri adalah chip atau IC (*integrated circuit*) yang bisa diprogram menggunakan komputer. Tujuan menanamkan program pada mikrokontroler adalah agar rangkaian elektronik dapat membaca *input*, memproses *input* tersebut dan kemudian menghasilkan output sesuai yang diinginkan. Jadi mikrokontroler bertugas sebagai "otak" yang mengendalikan *input*, proses dan *output* sebuah rangkaian elektronik. Salah satu jenis arduino yang sering digunakan yaitu jenis Arduino Uno.

Arduino Uno adalah *board* berbasis mikrokontroler pada ATMega 328. *Board* ini memiliki 14 digital *input / ouput* pin (dimana 6 pin dapat digunakan sebagai ouput PWM), 6 *input* analog, 16 MHz osilator kristal, koneksi USB, jack listrik dan tombol reset. Pin – pin ini berisi semua yang diperlukan untuk mendukung mikrokontroler, hanya terhubung ke komputer dengan kabel USB atau sumber tekanan bisa didapat dari adaptor AC – DC atau baterai untuk menggunakannya. Arduino Uno R3 berbeda dengan semua board sebelumnya karena Arduino Uno R3 ini tidak menggunakan *chip driver* FTDI USB-to-serial. Melainkan menggunakan fitur dari ATMega 16U2 yang diprogram sebagai konverter USB-to-serial [].

2.5 Storage Engine

Mesin penyimpanan (*storage engine*) adalah sistem manajemen modul perangkat lunak yang digunakan untuk membuat, membaca, dan mengupdate data dari database, alias tempat penyimpanan data. Ada dua tipe storage engine di MySQL, yaitu *transactional* dan *non-transactional*. Selain itu, ada pula berbagai jenis *storage engine* pada MySQL, yaitu:

- InnoDB

- MyISAM
- Memory
- CSV
- Merge
- Archive
- dan masih banyak yang lain

Dikarenakan penelitian ini menggunakan *storage engine* InnoDB, maka di dalam *selection* ini akan menjelaskan apa itu InnoDB serta kelebihan dan kekurangannya.

2.5.1 InnoDB

InnoDB menjadi salah satu produk Oracle Corp. setelah diakuisisi Oracle pada 2005. InnoDB Storage Engine lebih luas digunakan untuk proses transaksi (*transactional*) karena mendukung *row-level locking*, *crash recovery*, dan *multi-version concurrency control*.

InnoDB menjadi salah satu *Engine* yang menyediakan *foreign key*. Dimana lebih cocok digunakan ketika integritas data lebih diprioritaskan.

1. Kelebihan InnoDB

- Mendukung integritas data karena adanya *foreign key* dan *constraints*.
- Memiliki fitur pemulihan data tersendiri.
- Proses *insert* dan *update* data lebih cepat.
- Mendukung *row-level-locking*.
- Mendukung adanya transaksi antartabel.

2. Kekurangan InnoDB

- *Full Text Search* tidak tersedia pada *Storage Engine* InnoDB.
- Resource yang dibutuhkan lebih besar.

2.6 Variabel *dummy*

Variabel *dummy* adalah variabel yang nilainya sebenarnya adalah buatan, karena nilai variabel tersebut sebenarnya bukanlah skala. Atau secara mudahnya, variabel *dummy* adalah variabel independen yang wujudnya berskala non-metrik atau kategori. Jika variabel independen berukuran kategori atau dikotomi, maka dalam model regresi kita harus nyatakan sebagai variabel *dummy* dengan memberi kode 0 atau 1. Setiap variabel *dummy* menyatakan satu kategori variabel, dan setiap variabel dengan k kategori dapat dinyatakan dalam $k-1$ variabel *dummy*.

Analisis dengan variabel *dummy* dilakukan pada saat kita tertarik pada pengaruh variabel independen kategori, atau kita ingin memasukkan variabel kategori tersebut untuk meningkatkan kualitas penelitian kita. Analisis regresi dengan menggunakan variabel *dummy* memiliki kompleksitas, hasil analisis ini memiliki kemiripan dengan analisis kovarian (anakova), namun sistematika komputasinya sedikit berbeda. Ingat, prosedur yang disajikan dalam tulisan ini adalah prosedur analisis regresi dengan variabel independen berupa kategori, namun jika yang berwujud kategori adalah variabel dependen, maka teknik analisis yang dilakukan adalah dengan analisis regresi logistik [Akhtar (2018)].

2.7 Teori pH

pH atau derajat keasaman digunakan untuk menyatakan tingkat keasaman atau basa yang dimiliki oleh suatu zat, larutan atau benda. pH normal memiliki nilai 7 sementara bila nilai pH lebih dari 7 menunjukkan zat tersebut memiliki sifat basa sedangkan nilai pH kurang dari 7 menunjukkan keasaman. pH 0 menunjukkan derajat keasaman yang tinggi, dan pH 14 menunjukkan derajat kebasaan tertinggi. Umumnya indikator sederhana yang digunakan adalah kertas laksus yang berubah menjadi merah bila keasamannya tinggi dan biru bila keasamannya rendah.

Selain menggunakan kertas laksus, indikator asam basa dapat diukur dengan pH meter yang bekerja berdasarkan prinsip elektrolit/konduktivitas suatu larutan. Sis-

tem pengukuran pH mempunyai tiga bagian yaitu elektroda pengukuran pH, elektroda referensi dan alat pengukur impedansi tinggi. Istilah pH berasal dari "p", lambang matematika dari negatif logaritma, dan "H" lambang kimia untuk unsur Hidrogen. Definisi yang formal tentang pH adalah negatif logaritma dari aktivitas ion Hidrogen. pH adalah singkatan dari *power of Hydrogen* [Ardiansyah (2015)].

2.8 *Dissolved Oxygen*

Oksigen terlarut (*Dissolved Oxygen*) merupakan parameter mutu air yang penting karena nilai oksigen terlarut dapat menunjukkan tingkat pencemaran atau tingkat pengolahan air limbah. Oksigen terlarut ini akan menentukan kesesuaian suatu jenis air sebagai sumber kehidupan biota flora dan fauna di suatu daerah [Pramudya (2001)].

Distribusi oksigen terlarut (*Dissolved Oxygen*) di perairan sungai umumnya lebih merata dibandingkan dengan perairan tergenang. Hal ini disebabkan oleh adanya gerakan air yang kontinyu, sehingga memungkinkan terlarutnya oksigen dari udara ke air. Oksigen terlarut dalam air pada umumnya berasal dari difusi oksigen udara melalui permukaan air, aliran air, air hujan dan hasil fotosintesis tumbuhan air pada siang hari [Agung (2017)].

Oksigen diperlukan untuk menguraikan bahan organik. Oleh karena itu, penurunan kadar oksigen terlarut di dalam air merupakan indikasi kuat adanya pencemaran. Semakin tinggi tingkat pencemar air, semakin berkurang kadar oksigen terlarut dalam air. Oksigen terlarut sangat diperlukan untuk mempertahankan hidup bagi makhluk yang tinggal di air, baik tanaman maupun hewan [Pramudya (2001)].

2.9 Suhu

Suhu adalah besaran yang menyatakan derajat panas dingin suatu benda dan alat yang digunakan untuk mengukur suhu adalah thermometer. Dalam kehidupan sehari-hari masyarakat untuk mengukur suhu cenderung menggunakan indera peraba.

Tetapi dengan adanya perkembangan teknologi maka diciptakanlah termometer untuk mengukur suhu dengan valid.

Suhu memperlihatkan suatu drajat panas pada benda. Atau mudahnya, semakin tinggi suhu benda, maka semakin panas benda tersebut. Secara mikroskopis, suhu menunjukkan energi yang dipunya oleh suatu benda. Pada setiap atom dalam benda masing-masing bergerak, baik itu dalam bentuk perpindahan ataupun gerak di lokasi getaran. Makin tinggi energi atom-atom penyusun benda, maka semakin tinggi suhu benda tersebut. Suhu juga dapat disebut sebagai temperatur yang diukur dengan alat bernama termometer. Ada empat jenis termometer yang paling dikenal, yaitu Celcius, Fahrenheit, Reaumur serta Kelvin. Perbandingan antara satu macam termometer dengan yang lainnya mengikuti:

$$C : R : (F - 32) = 5 : 4 : 9$$

$$K = C + 273 \cdot (\text{derajat})$$

Sebab dari Kelvin ke derajat Celcius, Kelvin memulai dari 273 derajat, tidak dari -273 derajat. Serta derajat Celcius dimulai dari 0 derajat. Suhu Kelvin sama perbandingannya terhadap derajat Celcius yaitu 5:5, maka untuk mengubah suhu itu ke suhu yang lainnya, lebih baik memakai atau mengubahnya ke derajat Cecius terlebih dahulu, sebab bila kita memakai Kelvin akan lebih rumit untuk mengubahnya ke suhu yang lainnya [Kurniawan (2021)].

2.10 Generate dan Post Data

Data indikator air yang akan digunakan di dalam pengembangan sistem ini merupakan variabel *dummy* yang dihasilkan dengan cara mengeneratenya. Data akan digenerate dan dipost menggunakan aplikasi POSTMAN. Berikut merupakan langkah-langkah untuk mengenerate dan meng-post data indikatornya.

2.10.1 Menentukan *Range* Dari Variabel *Dummy* Untuk Data Indikator Air

Variabel *Dummy* yang diperlukan yaitu indikator ph, DO *Dissolved Oxygen*, dan suhu.

1. Data Indikator pH

Berdasarkan penelitian oleh Meilinda Pramleonita yang berjudul "PARAMETER FISIKA DAN KIMIA AIR KOLAM IKAN NILA HITAM" standar indikator pH untuk budidaya ikan nila yang memenuhi persyaratan SNI 7550 2009, yaitu sebesar 6,5-8,5 [Meilinda Pramleonita (2018)].

Dari data di atas dapat diambil kesimpulan bahwa variabel *dummy* yang akan digunakan nanti yaitu berada di *range*

$$6,5 < x < 8,5$$

2. Data indikator DO (*Dissolved Oxygen*)

Berdasarkan penelitian oleh Meilinda Pramleonita yang berjudul "PARAMETER FISIKA DAN KIMIA AIR KOLAM IKAN NILA HITAM" standar indikator DO untuk budidaya ikan nila yang memenuhi persyaratan SNI 7550 2009, yaitu minimal 3 mg/L serta untuk data maksimumnya diambil dari data maksimum pengukuran pada penelitian tersebut yaitu 14,5 mg/L [Meilinda Pramleonita (2018)].

Dari data di atas dapat diambil kesimpulan bahwa variabel *dummy* yang akan digunakan nanti yaitu berada di *range*

$$3 < x < 14,5$$

3. Data indikator Suhu

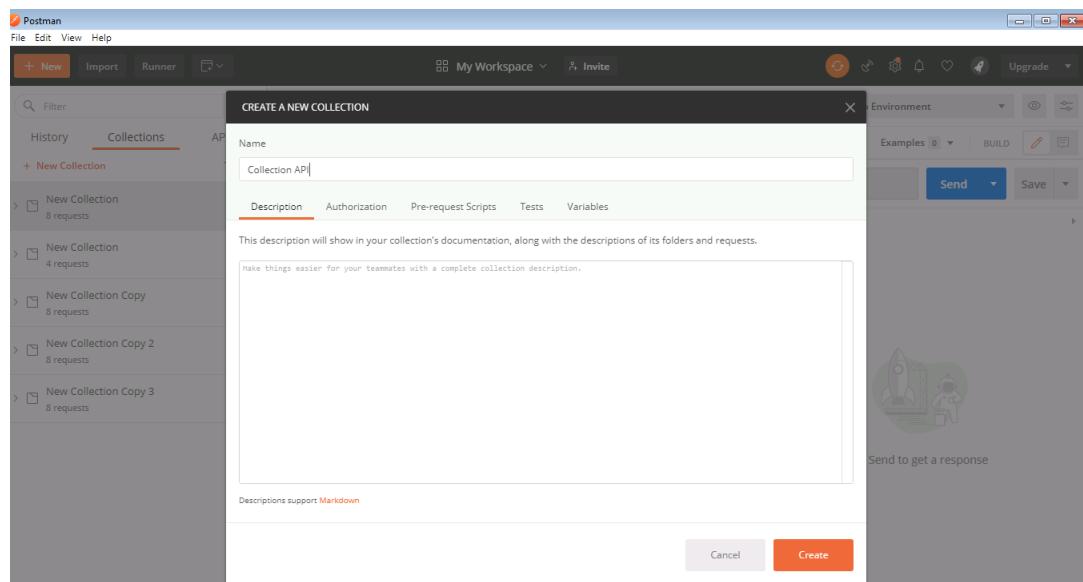
Berdasarkan penelitian oleh Meilinda Pramleonita yang berjudul "PARAMETER FISIKA DAN KIMIA AIR KOLAM IKAN NILA HITAM" standar indikator Suhu untuk budidaya ikan nila yang memenuhi persyaratan SNI 7550 2009, yaitu sebesar 25-32 derajat Celcius [Meilinda Pramleonita (2018)].

Dari data di atas dapat diambil kesimpulan bahwa variabel *dummy* yang akan digunakan nanti yaitu berada di *range*

$$25 < x < 32$$

2.10.2 Membuat Collection di POSTMAN

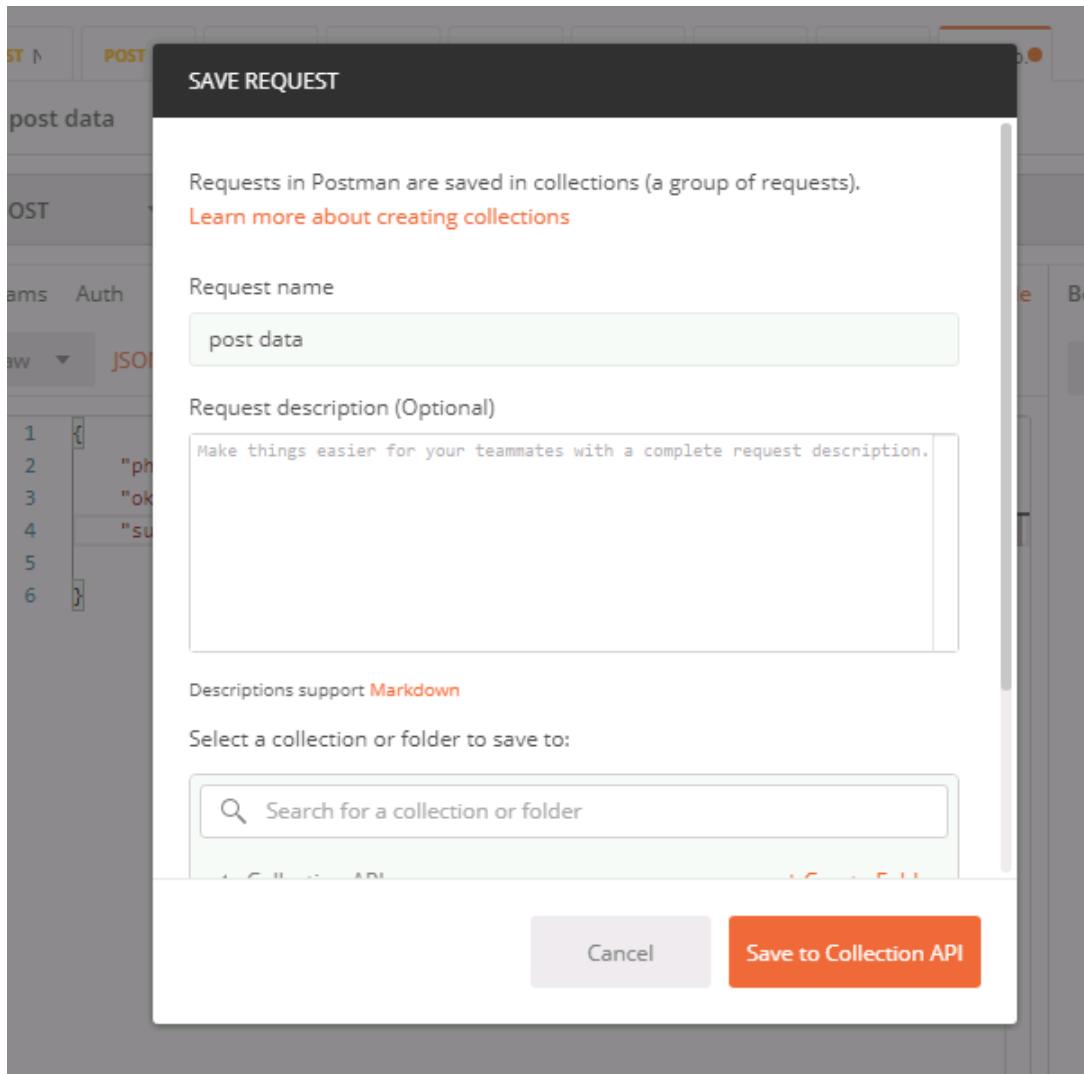
Setelah aplikasi postman terbuka maka akan muncul menu new collection. Klik menu tersebut lalu akan muncul tampilan seperti **Gambar 2.1**, setelah itu masukkan nama collection yang akan dibuat.



Gambar 2.1: Collection Postman

2.10.3 Membuat Request Pada Collection

Setelah Collection sudah dibuat maka selanjutnya adalah membuat request baru dengan mengklik menu colecion yang sudah dibuat lalu pilih new request. Setelah itu akan muncul tampilan seperti **Gambar 2.2**, lalu masukkan nama request yang akan dibuat



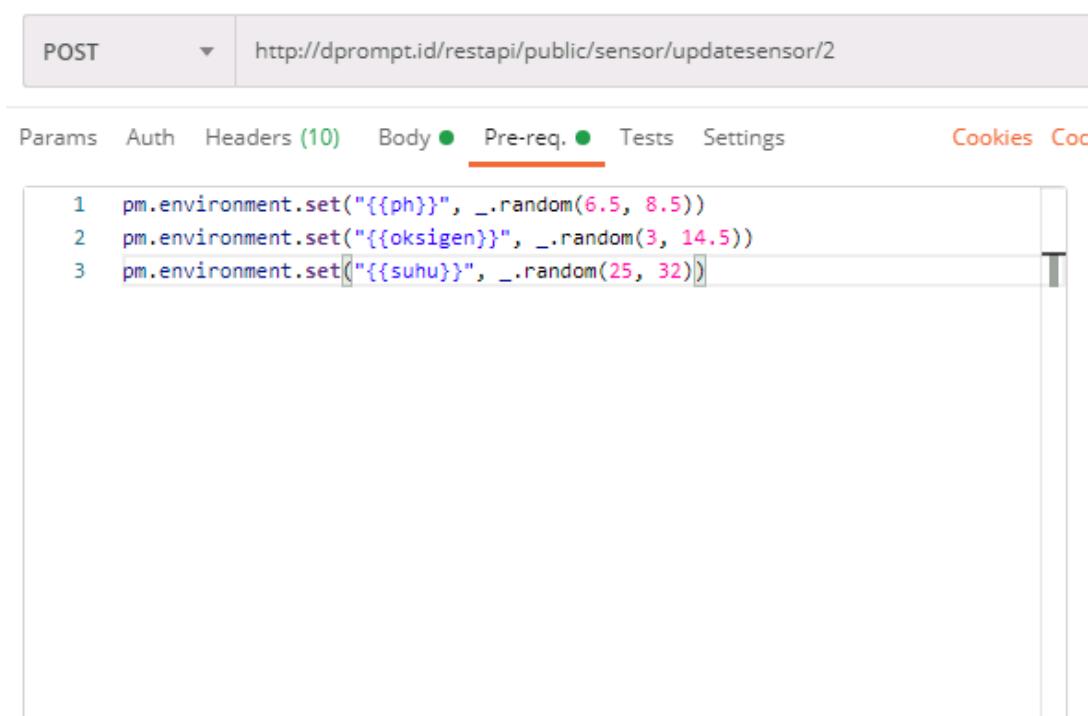
Gambar 2.2: Membuat Request

2.10.4 Melakukan Generate dan Post Data

Setelah Request selesai dibuat, ubah request ke POST lalu masukkan url. Setelah itu masukkan code berikut ke menu pre-req

```
pm.environment.set("{ph}", _.random(6.5, 8.5))
pm.environment.set("{oksigen}", _.random(3, 14.5))
pm.environment.set("{suhu}", _.random(25, 32))
```

untuk keseluruhan penjelasan tersebut terdapat pada **Gambar 2.3**



Gambar 2.3: Konfigurasi Request

Setelah konfigurasi request dibuat, langkah selanjutnya yaitu Post data dengan mengklik tombol send lalu data akan ter-post dan tergenerate secara otomatis.

2.11 REST API

REST merupakan singkatan dari REpresentational State Transfer. REST adalah arsitektur berbasis web standar dan menggunakan protokol HTTP. REST berputar di sekitar resource di mana setiap komponen adalah resourcenyanya dan resourcenyanya diakses oleh sebuah penghubung menggunakan metode HTTP standar. Dalam arsitektur REST, server REST hanya menyediakan akses ke resource dan client REST mengakses dan memodifikasi resource. Di sini setiap resource diidentifikasi oleh URI atau global ID. REST menggunakan berbagai representasi untuk mewakili resource seperti teks, JSON, XML.

Web service yang dikembangkan berdasarkan Arsitektur REST dikenal sebagai RESTful web service. Web service ini menggunakan metode HTTP untuk

mengimplementasikan konsep Arsitektur REST. RESTful Web service biasanya selalu mendefinisikan URI sebagai Uniform Resource Identifier sebuah layanan yang menyediakan representasi resource seperti JSON dan kumpulan *method* HTTP.

2.11.1 HTTP GET

Jika memiliki URL yang dimulai dengan http:// atau https://, dan tidak tahu apa yang ada di belakangnya, hal pertama yang harus dilakukan adalah mengeluarkan HTTP GET request. Dalam istilah REST URL ke resourceny, diperlukan menemukan pilihan, dalam arti mendapatkan representasi resourceny. Itulah gunanya HTTP GET.

Kode dapat ditulis dalam bahasa pemrograman untuk membuat GET request. tetapi saat melakukan peninjauan awal terhadap API sering kali lebih mudah menggunakan *command-line tool* seperti Wget. Di dalam pembahasan ini menggunakan opsi -S, yang menghasilkan HTTP response lengkap dari server, dan opsi -O -, yang menghasilkan dokumen daripada menyimpannya ke dalam file:

```
$ wget -S -O - http://www.youtypeitwepostit.com/api/
```

Command ini mengirimkan http request ke server sebagai berikut:

GET /api/ HTTP/1.1

Host: www.youtypeitwepostit.com

Dalam standar HTTP dikatakan bahwa GET request merupakan permintaan untuk representasi. Ini bukan dimaksudkan untuk mengubah status resource apa pun di server. Artinya jika memiliki URL ke resource dan tidak tahu apa-apa lagi, maka dapat membuat GET request dan mendapatkan representasinya. GET request tidak akan menghasilkan masalah seperti menghapus semua data karena GET adalah *method* yang aman.

Di dalam penerapannya, tidak ada jaminan bahwa HTTP GET aman. Beberapa fitur yang lama akan memaksa untuk membuat HTTP GET request jika ingin menghapus beberapa data tetapi kesalahan ini sangat jarang dalam fitur yang lebih

baru. Kebanyakan perancang API sekarang memahami bahwa client sering Menda-patkan(GET) URL hanya untuk melihat apa yang ada di belakangnya.

2.11.2 Membaca sebuah HTTP response

Sebagai respon dari GET request, server akan mengirimkan sejumlah besar data yang terlihat seperti berikut ini:

```
{
  "collection": [
    {
      "version": "1.0",
      "href": "http://www.youtypeitwepostit.com/api/",
      "items": [
        {
          "href": "http://www.youtypeitwepostit.com/api/messages/21818525390699506",
          "data": [
            {
              "name": "text",
              "value": "Test."
            },
            {
              "name": "date_posted",
              "value": "2013-04-22T05:33:58.930Z"
            }
          ],
          "links": []
        },
        {
          "href": "http://www.youtypeitwepostit.com/api/messages/3689331521745771",
          "data": [
            {
              "name": "text",
              "value": "Hello."
            },
            {
              "name": "date_posted",
              "value": "2013-04-20T12:55:59.685Z"
            }
          ],
          "links": []
        },
        {
          "href": "http://www.youtypeitwepostit.com/api/messages/7534227794967592",
          "data": [
            {
              "name": "text",
              "value": "Pizza?"
            },
            {
              "name": "date_posted",
              "value": "2013-04-18T03:22:27.485Z"
            }
          ],
          "links": []
        }
      ],
      "template": {
        "data": [
          {"prompt": "Text of message", "name": "text", "value": ""}
        ]
      }
    }
  ]
}
```

Gambar 2.4: HTTP response data

Setiap HTTP response dibagi menjadi 3 bagian yaitu:

1. *Status code*

Status code merupakan tiga digit angka yang menjelaskan bagaimana request itu berjalan. Response code adalah hal yang pertama dilihat oleh API client dan mengatur ritme untuk response yang tersisa. Di gambar di atas *Status code* adalah 200(OK). ini adalah *Status code* yang diharapkan client karena itu berarti

semua berjalan dengan baik. berikut merupakan *Status code* beserta pesannya:

- 1xx(*informational*)

Response code ini hanya digunakan dalam komunikasi antara HTTP client dan server

- 2xx(*successful*)

Apa pun status transisi yang diminta client telah terjadi dan berhasil.

- 3xx(*Redirection*)

Status transisi yang diminta client tidak terjadi. Tetapi jika client mau untuk membuat HTTP request yang sedikit berbeda, permintaan itu harus melakukan apa yang diminta client.

- 4xx(*Client Error*)

Status transisi yang diminta client tidak terjadi, karena ada masalah dengan HTTP request. Permintaan itu formatnya salah, tidak koheren, kontradiktif, atau tidak dapat diterima server

- 5xx(*Server Error*)

Status transisi yang diminta client tidak terjadi, karena masalah pada sisi server. Mungkin tidak ada yang dapat dilakukan client selain menunggu masalahnya diperbaiki

2. *Entity-body*

Entity-body merupakan dokumen yang ditulis dalam beberapa format data, di mana client diharapkan memahaminya. Jika Get request dianggap sebagai permintaan untuk representasi, maka dapat memikirkan *Entity-body* sebagai representasi(secara teknis, seluruh HTTP response merupakan "representasi", tetapi informasi penting biasanya berada di *Entity-body*).

3. *Response headers*

Response headers merupakan seri dari rangkaian *key-value* yang mendeskripsikan *Entity-body* dan HTTP response secara umum. Response headers dikirim antara *Status code* dan *Entity-body*.

Bagian penting dari HTTP header adalah *Content-Type*(jenis konten), yang memberi tahu HTTP client bagaimana memahami *Entity-body*. Sangat penting bahwa valuenya memiliki nama yang spesial atau spesifik. Value dari *Content-Type header* adalah jenis media dari *entity-body*'s.

Bagian dari web yang dapat dilihat manusia dengan web browser, jenis media yang paling umum yaitu teks atau html dan jenis gambarnya yaitu image atau jpeg. Di sini, jenis media adalah salah satu yang mungkin belum pernah dilihat sebelumnya: application/ vnd.collection+json.

2.11.3 HTTP POST

Untuk menambahkan item baru ke collection, maka harus mengirim permintaan POST ke URL milik collection. Ini bukan hanya bagaimana collection+JSON bekerja. itu adalah fakta dasar tentang HTTP.

POST dirancang supaya metode yang seragam dapat mencakup fungsi-fungsi berikut:

- Sebagai keterangan *resource* yang tersedia;
- Memost pesan ke papan buletin, *newsgroup*, *mailing list*. atau artikel;
- Memberikan kumpulan data seperti hasil dari pengiriman formulir ke proses penaganan data;
- Memperluas database melalui operasi penambahan;

POST *request* yang dikirim sangat mirip dengan HTTP *response*. Terdapat *content-type header* dan entity-body. Meskipun GET *request* yang ditunjukkan sebelumnya tidak memberikan header apa pun, HTTP *request* apa pun dapat memiliki header, dan ada sejumlah header yang sangat penting dalam GET *request*.

Berikut tanggapan yang didapatkan untuk POST *request*-nya:

201 Created

Location: http://www.youtypeitwepostit.com/api/47210977342911065

Saat mendapatkan *response code* 201, lokasi header memberi tahu tempat untuk mencari sesuatu yang telah dibuat. RFC 2616 menetapkan arti dari *response code* 201 dan lokasinya, tetapi Collection+JSON juga menyebutkannya secara spesifik tetapi hanya untuk memperjelas saja.

Jika mengirim GET *request* seperti berikut:

GET /api/47210977342911065 HTTP/1.1

Host: www.youtypeitwepostit.com

maka akan melihat seperti berikut ini:

HTTP/1.1 200 OK

Content-Type: application/vnd.collection+json

```
{
  "collection": {
    "version" : "1.0",
    "href" : "http://www.youtypeitwepostit.com/api/
47210977342911065",
    "items" : [
      { "href" : "http://www.youtypeitwepostit.com/api/
messages/47210977342911065",
      "data": [
        { "name": "date_posted", "value": "2014-04-20T20:15:32.858Z" },
        { "name": "text", "value": "Squid!" }
      ],
      "links": []
    }
  ]
}
```

```
}
```

```
}
```

Ini merupakan fitur yang nyaman dari Collection+JSON. Hampir semuanya yang ada di dokumen bersifat opsional. Artinya tidak perlu menulis pengurai yang berbeda untuk tipe dokumen yang berbeda. Collection+JSON menggunakan format JSON yang sama untuk mewakili daftar item, item individu, template yang sudah diisi, dan hasil pencarian.

2.11.4 Protokol semantik HTTP

Meskipun *resource* bisa berupa apa saja, *client* tidak dapat melakukan apa pun yang diinginkannya untuk *resource*. Dalam sistem RESTful, *client* dan server hanya berinteraksi dengan saling mengirim pesan yang mengikuti protokol yang telah ditentukan sebelumnya.

HTTP standar mendefinisikan delapan jenis pesan. Berikut ini empat jenis pesan yang paling banyak digunakan:

1. GET

Method GET bekerja untuk mendapatkan representasi dari suatu *resource*. *client* mengirimkan GET *request* untuk meminta representasi *resource* yang diidentifikasi oleh URL. Di sini, *client* meminta representasi dari *microblog post*nya, dan server mengirimkan ke format application/vnd.collection+json seperti ini:

GET /api/45ty HTTP/1.1

Host: www.youtypeitwepostit.com

Content-Type: application/vnd.collection+json

```
{
  "collection" :
```

```
{  
    "version" : "1.0",  
    "href" : "http://localhost:1337/api/",  
    "items" :  
    [ {  
        "href": "http://localhost:1337/api/  
2csl73jr6j5",  
        "data": [  
            {  
                "name": "text",  
                "value": "Bird"  
            },  
            {  
                "name": "date_posted",  
                "value": "2013-01-24T18:40:42.190Z"  
            }  
        ]  
    } ],  
    "template" : {  
        "data" : [  
            {"prompt" : "Text of message",  
             "name" : "text", "value" : ""}  
        ]  
    }  
}
```

Disebutkan sebelumnya bahwa GET didefinisikan sebagai *method* HTTP yang aman. mengirim GET *request* ke server harus memiliki efek yang sama pada

status *resource* sebagai yang tidak mengirim GET *request*, artinya tidak berpengaruh sama sekali. Efek samping insidental seperti logging dan pembatasan kecepatan tidak masalah, tetapi *client* tidak boleh membuat GET *request* yang kemungkinan bisa mengubah status *resource*. *response code* yang paling umum untuk GET *request* adalah 200 (OK).

2. DELETE

Method DELETE bekerja untuk menghapus sebuah resource. *client* mengirim permintaan *DELETE* ketika ingin *resource* tersebut hilang. *client* ingin server menghapus *resource* dan tidak pernah merujuknya lagi. Tentu saja, server tidak berkewajiban untuk menghapus sesuatu yang tidak diinginkannya.

Dalam potongan HTTP ini, *client* meminta untuk menghapus sebuah *microblog post*:

DELETE /api/45ty HTTP/1.1

Host: www.youtypeitwepostit.com

Server mengembalikan *status code* 204 (Tidak ada konten), yang menunjukkan bahwa itu telah menghapus postingan dan tidak ada lagi konten postingan tentang itu:

HTTP/1.1 204 No Content

Jika *DELETE request* berhasil, *status code* yang mungkin adalah 204 (No Content, yaitu, "itu dihapus dan tidak ada lagi konten tentang itu"), 200 (OK, yaitu, "itu dihapus, dan inilah pesan tentang itu"); dan 202 (Diterima, yaitu, "pesan diterima, dan akan menghapusnya nanti").

Jika *client* mencoba untuk mendapatkan *resource* yang telah dihapus, server akan mengembalikan *error response code*, biasanya 404(tidak ditemukan) atau 410(hilang):

GET /api/45ty HTTP/1.1

Host: www.youtypeitwepostit.com

HTTP/1.1 404 Not Found

3. POST

Method POST memiliki dua pekerjaan, yang akan dibahas secara terpisah. Yang pertama adalah POST-to-append, di mana mengirimkan POST *request*

ke resoruce untuk membuat resource baru di bawahnya. Saat *client* mengirim POST-to-append *request*, ia mengirimkan representasi *resource* yang ingin dibuatnya di *entity-body* yang telah diminta.

Response code yang paling umum untuk POST-to-append *request* adalah 201 (dibuat). itu memungkinkan *client* tahu bahwa *resource* baru telah dibuat. *Location header* memungkinkan *client* mengetahui URL ke *resource* baru ini. *response code* umum lainnya adalah 202 (telah diterima), yang berarti bahwa server bermaksud untuk membuat *resource* baru berdasarkan representasinya, tetapi belum benar-benar membuatnya.

Method POST tidak aman Jika mengirim permintaan post ini lima kali, itu mungkin akan membuat sebuah post baru sebanyak lima kali, masing-masing dengan *text* yang sama tetapi sedikit berbeda di bagian *date_created*.

Itu semua adalah POST-to-append. Tetapi mungkin pernah menggunakan POST untuk segala macam hal lainnya selain membuat *resource* baru. itu adalah pekerjaan POST lainnya. itu disebut *overloaded* POST.

Method HTTP POST memiliki sebuah rahasia lainnya, yang pasti akan ditemukan jika pernah bekerja di pengembangan website. POST tidak hanya digunakan untuk membuat *resource* baru. Jika ditelusuri lebih jauh, HTTP POST digunakan untuk menyampaikan segala jenis perubahan (*update*) yaitu PUT, DELETE, PATCH, LINK, dan UNLINK semuanya digabung menjadi satu yang disebut dengan *overloaded POST*.

4. PUT

Method PUT bekerja untuk mengubah status *resource*. *client* mengambil representasi yang didapat dari GET *request*, memodifikasikannya, dan mengirimkannya kembali sebagai isi dari PUT *request*.

Server bebas mengolah PUT *request* karena *entity-body* mencoba mengubah sedikit status *resource* yang dianggap server hanya membaca, atau benar-benar

karena alasan yang lainnya. Jika server memutuskan untuk menerima PUT *request*, server mengubah status *resource* supaya sesuai dengan apa yang dikatakan *client* dalam representasinya, dan biasanya mengirim status 200 (OK) atau 204 (Tanpa Konten).

PUT memiliki hasil yang sama seperti DELETE. Jika mengirim permintaan PUT yang sama 10 kali, Hasilnya akan sama seperti jika mengirimnya se kali. *client* juga dapat menggunakan PUT untuk membuat *resource* baru, jika mengetahui URL di mana *resource* baru itu dapat ditempatkan [Wellek (2013)].

2.12 Lumen Framework

Lumen adalah *micro framework* yang pada dasarnya merupakan laravel yang lebih ringan, dengan *routing library* yang mendapatkan throughput lebih tinggi untuk *request* per menit. *Benchmark* Lumen juga dibandingkan dengan *micro framework* PHP yang lainnya menunjukkan bahwa Lumen merupakan salah satu yang tercepat untuk fitur yang disediakannya. Ini membuat Lumen bagus untuk API, layanan sederhana, dan website kecil.

2.12.1 *Routing*

1. *Basic Routing*

Untuk menentukan sebagian besar *routes* pada aplikasi, maka dapat membuka file di direktori app/Http/routes.php yang dimuat oleh file bootstrap/app.php/. Ini merupakan *routes* lumen paling dasar untuk menerima URI dan *Closure*:

```
$app->get('/', function () {  
  
    return 'Hello World';  
  
});  
  
  
  
  
  
  
$app->post('foo/bar', function () {  
  
    return 'Hello World';  
  
});  
  
  
  
  
  
$app->put('foo/bar', function () {  
  
    //  
  
});  
  
  
  
$app->delete('foo/bar', function () {  
  
    //  
  
});
```

untuk membuat URL ke *application's routes* menggunakan *url helper*:

```
$url = url('foo');
```

2. Route Parameters

Terkadang diperlukan untuk menangkap segmen URI dalam route yang dimiliki. Misalnya, mungkin perlu mengambil ID pengguna dari URL. Itu dapat dilakukan dengan menentukan parameter dalam *route*:

```
$app->get('user/{id}', function ($id) {
    return 'User ' . $id;
});
```

Parameter dapat ditentukan *route* sebanyak yang diperlukan oleh *route* yang dimiliki:

```
$app->get('posts/{post}/comments/{comment}', function ($postId,
    $commentId) {
    // ...
});
```

Route parameters selalu di dalam tanda kurung kurawal. Parameter akan diteruskan ke *route's Closure* saat *route* dieksekusi.

Format *route parameters* dapat dibatasi dengan menentukan *regular expression* dalam mendefinisikan *route*:

```
$app->get('user/{name:[A-Za-z]+}', function ($name) {
    // ...
});
```

3. Named Routes

Named Routes memungkinkan untuk membuat URL dengan mudah untuk *route* tertentu. Nama untuk rute dapat ditentukan menggunakan "as array key" saat mendefinisikan *route*:

```
$app->get('user/profile', ['as' => 'profile', function () {
    // ...
}]);
```

Nama *route* juga dapat ditentukan untuk *controller actions*:

```
$app->get('user/profile', [
    'as' => 'profile', 'uses' => 'UserController@showProfile'
]);
```

Setelah menetapkan nama untuk *route* tertentu, maka dapat menggunakan nama *route* tersebut saat membuat URL melalui fungsi *route*:

```
$url = route('profile');

// $url = route('profile');

$redirect = redirect()->route('profile');
```

Jika *route* mendefinisikan parameter, maka dapat meneruskan parameter sebagai argumen kedua ke metode *route*. Parameter yang diberikan secara otomatis akan dimasukkan ke dalam URL:

```
$app->get('user/{id}/profile', ['as' => 'profile',

function ($id) {
    // ...
}]);
```

\$url = route('profile', ['id' => 1]);

4. Route Groups

Route groups memungkinkan untuk berbagi atribut *route*, seperti *middleware* atau *namespace*, di sejumlah besar *route* tanpa perlu menentukan atribut tersebut di setiap *route*. Atribut bersama ditentukan dalam format *array* sebagai parameter pertama ke *\$app->group method*.

untuk menetapkan *middleware* ke semua *route* dalam grup, maka dapat menggunakan "middleware key" dalam "group attribute array".

Middleware akan dieksekusi dalam urutan yang telah didefinisikan dengan *array* ini:

```
$app->group(['middleware' => 'auth'], function ($app) {  
  
    $app->get('/', function () {  
  
        // Uses Auth Middleware  
  
    });  
  
    $app->get('user/profile', function () {  
  
        // Uses Auth Middleware  
  
    });  
});
```

Penggunaan umum lainnya untuk *route group* adalah menetapkan *namespace* PHP yang sama ke grup *controllers*. Parameter *namespace* dapat digunakan di *array* atribut grup untuk menentukan *namespace* untuk semua *controllers* dalam grup:

```
$app->group(['namespace' =>  
    'App\Http\Controllers\Admin'], function ($app) {  
  
    // Controllers Within The  
  
    "App\Http\Controllers\Admin" Namespace  
  
});
```

Atribut *prefix group array* dapat digunakan untuk mengawali setiap *route* dalam grup dengan URI tertentu. Misalnya, ingin memberi awalan semua URI *route* dalam grup dengan *admin*:

```
$app->group(['prefix' => 'admin'], function ($app) {  
  
    $app->get('users', function () {  
  
        // Matches The "/admin/users" URL  
  
    });  
  
});
```

Prefix parameter juga dapat digunakan untuk menentukan parameter umum untuk *route* yang dikelompokkan:

```
$app->group(['prefix' =>

'accounts/{account_id}'], function ($app) {

    $app->get('detail', function ($account_id) {
        // Matches the accounts/{account_id}

        /detail URL

    });
});
```

2.12.2 *Controllers*

Selain menentukan semua *request* logika penanganan dalam satu file routes.php, mungkin ingin mengatur ini dengan menggunakan kelas *Controller*. *Controllers* dapat mengelompokkan *request* logika penanganan HTTP terkait ke dalam kelas. *Controllers* biasanya disimpan dalam direktori app/HTTP/Controllers.

1. *Basic Controllers*

Berikut adalah contoh kelas *controller* dasar. semua *controllers* lumen harus memperluas kelas *controller* dasar yang disertakan dengan penginstalan lumen *default*:

```
<?php

namespace App\Http\Controllers;

use App\User;

class UserController extends Controller {

    public function showProfile($id){

        return view('user.profile',
            ['user' => User::findOrFail($id)]);

    }

}
```

Kita dapat mengarahkan dari *route* ke *controller* seperti ini:

```
$app->get('user/{id}', 'UserController@showProfile');
```

Sekarang, ketika *request* cocok dengan URI *route* yang ditentukan, *method* pada *showProfile* ke *UserController* akan dieksekusi. Tentu saja parameter *route* juga akan diteruskan ke *method*.

2. Controller Middleware

Middleware dapat ditugaskan ke rute *controller* seperti ini:

```
$app->get('profile', [  
  
    'middleware' => 'auth',  
  
    'uses' => 'UserController@showProfile'  
  
]);
```

Namun, akan lebih mudah untuk menetapkan *middleware* dalam konstruktor *controller*. dengan menggunakan *method middleware* dari konstruktor *controller*, maka dapat dengan mudah menetapkan *middleware* ke *controller*. *Middleware* bahkan dapat dibatasi hanya untuk *method* tertentu pada kelas *controller*:

```
class UserController extends Controller {  
  
    public function __construct() {  
  
        $this->middleware('auth');  
  
        $this->middleware('log', ['only' => ['fooAction', 'barAction']]));  
  
        $this->middleware('subscribed', ['except' =>  
  
            ['fooAction', 'barAction']]));  
  
    }  
}
```

2.12.3 Views

views berisi HTML yang disajikan oleh aplikasi dan memisahkan *controller* atau logika dari aplikasi dari logika yang dipresentasikan. *Views* disimpan dalam direktori resources/views.

view sederhana mungkin terlihat seperti ini:

```
<html>

    <body>

        <h1>Hello, <?php echo $name; ?></h1>

    </body>

</html>
```

Karena *view* ini disimpan di dalam resource/view/greet.php, kita dapat mereturnnya menggunakan fungsi *global view helper* seperti:

```
$app->get('/', function () {
    return view('greeting', ['name' => 'James']);
});
```

Seperti yang dapat dilihat, argumen pertama yang diteruskan ke *view helper* sesuai dengan nama file *view* di direktori resources/view. Argumen kedua yang diteruskan ke *helper* adalah *array* dari data yang harus tersedia untuk *view*. Dalam hal ini, kami meneruskan variabel nama, yang ditampilkan dalam *view* hanya dengan menjalankan echo pada variabel.

Tentu saja, *views* juga dapat ditempatkan di dalam sub-direktori dari direktori resource/views. Notasi "Dot" dapat digunakan untuk mereferensikan *nested views*. Misalnya, jika tampilan disimpan di resource/views/admin/profile.php, maka dapat mereferensikannya seperti ini:

```
return view('admin.profile', $data);
```

Jika perlu menentukan apakah *view* tersedia, maka dapat menggunakan *method* yang tersedia setelah memanggil *view helper* tanpa argumen. *Method* ini akan mereturn *value true* jika *view* tersedia:

```
if (view()->exists('emails.customer')) {  
    //  
}
```

Ketika *view helper* dipanggil tanpa argumen, sebuah *instance* dari Illuminate/Contracts/View/Factory direturn, memberi akses ke salah satu *factory's method* [Documentation (2020)].

2.13 Scrum

Scrum didasarkan pada pengalaman dan pemikiran yang efektif. Empirisme menegaskan bahwa pengetahuan berasal dari pengalaman dan pengambilan keputusan berdasarkan apa yang diamati. Berfikir efektif dan berfokus pada hal-hal penting.

Scrum menggunakan pendekatan yang berulang dan bertahap untuk mengoptimalkan prediktabilitas dan mengendalikan risiko. Scrum melibatkan sekelompok orang yang secara kolektif memiliki semua keterampilan dan keahlian untuk melakukan pekerjaan dan berbagi atau memiliki keterampilan yang diperlukan.

Scrum menggabungkan empat aktivitas formal untuk inspeksi dan adaptasi dalam aktivitas yang memiliki konten, yaitu *Sprint*. Aktivitas ini dapat berhasil jika menerapkan pilar empiris Scrum yaitu transparansi, inspeksi, dan adaptasi.

1. Transparansi

Proses dan pekerjaan yang muncul harus terlihat oleh mereka yang melakukan pekerjaan serta mereka yang menerima pekerjaan. Dengan Scrum, pengambilan keputusan sangat penting didasarkan pada kondisi ketiga komponen formal. Komponen Scrum yang memiliki transparansi rendah dapat mengurangi nilai dan meningkatkan risiko.

2. Inspeksi

Komponen Scrum dan *progress* untuk mencapai tujuan yang disepakati harus sering diperiksa untuk mendeteksi masalah atau masalah yang mungkin tidak diinginkan. Untuk membantu pemeriksaan, Scrum memberikan suatu instruksi dalam lima aktivitasnya.

3. Adaptasi

Jika produk yang dihasilkan tidak dapat diterima, maka proses yang diterapkan atau keperluan yang diproduksi harus disesuaikan. Penyesuaian harus dilakukan secepat mungkin untuk meminimalisir kesalahan lebih lanjut.

Adaptasi akan menjadi lebih sulit ketika orang-orang yang terlibat bekerja sendiri. Tim Scrum diharapkan dapat beradaptasi saat mempelajari sesuatu yang baru melalui inspeksi.

2.13.1 Nilai-nilai Scrum

Keberhasilan penggunaan metode Scrum bergantung pada orang-orang yang menjalankan lima nilai Scrum, yaitu: Komitmen, Fokus, Keterbukaan, Rasa Hormat, dan Keberanian.

Tim Scrum berkomitmen untuk mencapai tujuannya dan mendukung satu sama lain. Fokus utama mereka adalah pada pekerjaan dalam *Sprint* untuk membuat *process* sebaik mungkin. Tim Scrum dan *stakeholdernya* terbuka mengenai pekerjaan dan tantangannya. Anggota Tim Scrum saling menghormati satu sama lain untuk menjadi orang yang mampu, mandiri, dan dihormati oleh orang-orang yang bekerja dengan mereka. Anggota Tim Scrum memiliki keberanian untuk melakukan hal yang benar, untuk mengatasi masalah yang sulit.

Nilai-nilai ini memberikan arahan kepada Tim Scrum terkait dengan pekerjaan, tindakan, dan perilaku mereka. Keputusan yang dibuat, langkah-langkah yang diambil, dan bagaimana Scrum digunakan harus memperkuat nilai-nilai ini, bukan malah mengurangi atau merusaknya. Anggota Tim Scrum dapat belajar dan mengeksplorasi nilai-nilai saat mereka bekerja dengan menjalankan komponen Scrum. Ketika nilai-nilai ini diwujudkan oleh Tim Scrum dan orang-orang yang bekerja dengan mereka, maka pilar-pilar empiris Scrum yaitu transparansi, inspeksi, dan adaptasi dapat membangun kepercayaan di dalam Tim Scrum.

2.13.2 Tim Scrum

Unit fundamental dari Scrum adalah sebuah tim kecil yang terdiri dari beberapa orang, yaitu Tim Scrum. Tim Scrum terdiri dari satu Scrum Master, satu Pemilik Produk, dan Pengembang. Di dalam Tim Scrum, tidak ada pangkat dan jabatan. Ini adalah unit kohesif profesional yang berfokus pada satu tujuan pada satu waktu, yaitu Tujuan Produk.

Tim Scrum memiliki lintas fungsi, artinya anggotanya memiliki semua keterampilan yang diperlukan untuk melakukan pekerjaan di setiap *Sprint*. Mereka juga mengatur diri sendiri, artinya mereka secara internal memutuskan siapa melakukan apa, kapan, dan bagaimana.

Tim Scrum cukup kecil untuk selalu cekatan dan cukup besar untuk menyelesaikan pekerjaan penting dalam *Sprint*, biasanya 10 orang atau kurang. Secara umum, tim yang lebih kecil berkomunikasi dengan lebih baik dan lebih produktif. Jika Tim

Scrum menjadi terlalu besar, mereka harus mempertimbangkan untuk mengatur ulang menjadi beberapa Tim Scrum yang kohesif, masing-masing berfokus pada produk yang sama. Oleh karena itu, mereka harus berbagi *Product Goal*, *Product Backlog*, dan *Product Owner* yang sama.

Tim Scrum bertanggung jawab atas semua aktivitas terkait produk mulai dari kolaborasi *stakeholder*, verifikasi, pemeliharaan, pengoperasian, eksperimen, penelitian dan pengembangan, dan hal lain yang mungkin diperlukan. Itu semua telah terorganisir untuk mengatur pekerjaan mereka sendiri. Bekerja dalam *Sprint* dengan tempo yang sangat cepat dapat meningkatkan fokus dan konsistensi Tim Scrum.

Seluruh Tim Scrum bertanggung jawab untuk menciptakan Peningkatan (*Increment*) yang berguna di setiap *Sprint*. Scrum menetapkan tiga akuntabilitas khusus dalam Tim Scrum, yaitu: Pengembang, Pemilik Produk, dan *Scrum Master*.

1. Pengembang (*Developers*)

Developer adalah orang-orang di Tim Scrum yang berkomitmen mengerjakan aspek apa pun untuk menghasilkan peningkatan (*Increment*) yang berguna di setiap *Sprint*.

Keterampilan khusus yang dibutuhkan oleh Pengembang harus luas dan bervariasi sesuai dengan yang dibutuhkan pekerjaan. Namun, Pengembang selalu bertanggung jawab untuk:

- Membuat rencana untuk *Sprint*, yaitu *Sprint Backlog*;
- Menanamkan kualitas hasil pekerjaan dengan mengikuti "*Definition of Done*" dari Scrum;
- Menyesuaikan rencana mereka setiap hari untuk keberhasilan dari setiap *Sprint*; dan,
- Saling meminta pertanggungjawaban sebagai profesional.

2. Pemilik Produk

Pemilik Produk bertanggung jawab untuk memaksimalkan nilai produk yang dihasilkan dari hasil kerja Tim Scrum. Bagaimana hal ini dilakukan dapat sangat bervariasi antar organisasi, Tim Scrum, dan individu.

Pemilik Produk juga bertanggung jawab atas pengelolaan *Product Backlog* yang efektif, yang meliputi:

- Mengembangkan dan secara eksplisit mengkomunikasikan Tujuan Produk;
- Membuat dan mengkomunikasikan *Product Backlog items* dengan jelas;
- Menginstruksikan terkait *Product Backlog items*; dan,
- Memastikan *Product Backlog* yang transparan, jelas dan dipahami.

Pemilik Produk dapat melakukan pekerjaan di atas atau dapat memberikan tanggung jawab kepada orang lain. Terlepas dari itu, Pemilik Produk harus tetap bertanggung jawab.

Agar Pemilik Produk berhasil, seluruh anggota harus menghormati keputusannya. Keputusan ini dapat terlihat dalam konten dan instruksi di dalam *Product Backlog*, dan melalui sebuah Peningkatan (*Increment*) yang dapat diperiksa di *Sprint Review*.

Pemilik Produk adalah satu orang. Pemilik Produk dapat mewakili kebutuhan dari banyak *stakeholder* dalam *Product Backlog*. Mereka yang ingin mengubah *Product Backlog* dapat melakukannya dengan mencoba untuk meyakinkan Pemilik Produk.

3. *Scrum Master*

Scrum Master bertanggung jawab untuk memimpin jalannya Scrum seperti yang dijelaskan dalam Panduan Scrum. Mereka melakukan ini dengan mem-

bantu semua orang dalam memahami teori dan praktik Scrum, baik di dalam Tim Scrum maupun organisasi.

Scrum Master bertanggung jawab atas efektivitas Tim Scrum. Mereka melakukan ini dengan tujuan untuk meningkatkan kinerjanya. Dalam *Scrum*, *Scrum Master* adalah pemimpin yang melayani Tim Scrum dan organisasi yang lebih besar.

Scrum Master melayani Tim Scrum dengan beberapa cara, Seperti:

- Melatih anggota tim dalam pengelolaan diri dan pengelolaan dalam berbagai bidang;
- Membantu Tim Scrum fokus dalam menghasilkan Peningkatan (*Increment*) dengan nilai tinggi yang memenuhi "*Definition of Done*" dari Scrum;
- Menghilangkan hambatan-hambatan untuk kemajuan Tim Scrum; dan,
- Memastikan bahwa semua aktivitas Scrum berlangsung secara positif, produktif, dan telah didokumentasikan.

Scrum Master melayani Pemilik Produk dengan beberapa cara, Seperti:

- Membantu menemukan sebuah teknik untuk mencapai Tujuan Produk yang efektif dan memanajemen *Product Backlog*;
- Membantu Tim Scrum memahami kebutuhan akan *Product Backlog items* yang jelas dan ringkas;
- Membantu membangun perencanaan produk untuk keadaan yang kompleks; dan,
- Memfasilitasi *stakeholder* sesuai permintaan atau kebutuhan.

Scrum Master melayani organisasi dengan beberapa cara, Seperti:

- Memimpin, melatih, dan membimbing organisasi dalam penerapan Scrum;

- Merencanakan dan memberi masukan tentang implementasi Scrum dalam organisasi;
- Membantu karyawan dan *stakeholder* memahami dan melakukan pendekatan untuk pekerjaan yang kompleks; dan,
- Menghilangkan penghalang antara *stakeholder* dan Tim Scrum.

2.13.3 Aktivitas-aktivitas Scrum (*Scrum Events*)

Setiap aktivitas di Scrum adalah sebuah kesempatan untuk memeriksa dan mengadaptasi komponen Scrum. Aktivitas ini secara khusus dirancang untuk menghasilkan transparansi yang diperlukan. Kegagalan untuk mengoperasikan aktivitas yang ditentukan mengakibatkan hilangnya peluang untuk memeriksa dan beradaptasi. Aktivitas digunakan di Scrum untuk menciptakan kedisiplinan dan meminimalkan kebutuhan akan rapat yang tidak ditentukan di Scrum.

Secara optimal, semua aktivitas diadakan pada waktu dan tempat yang sama untuk mengurangi tingkat kerumitan.

1. Sprint

Sprint merupakan bagian penting dari Scrum, di mana ide diubah menjadi nilai. *Sprint* adalah aktivitas berdurasi tetap selama satu bulan atau kurang untuk menciptakan konsistensi. *Sprint* yang baru dimulai segera setelah *Sprint* sebelumnya berakhir.

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

- Tidak ada perubahan yang dilakukan yang akan membahayakan tujuan *Sprint*;
- Kualitas produk tidak berkurang
- Product Backlog diperbaiki sesuai kebutuhan; dan,

- Cakupan dapat diklarifikasi dan dinegosiasikan ulang dengan Pemilik Produk seiring dengan semakin banyaknya hal yang dipelajari.

Sprint memungkinkan prediktabilitas dengan memastikan pemeriksaan dan adaptasi *progress* untuk mencapai Tujuan Produk setidaknya setiap bulan kalender. Ketika waktu *Sprint* terlalu panjang, tujuan *Sprint* mungkin menjadi tidak valid, kompleksitas bisa meningkat, dan risiko bisa meningkat. *Sprint* yang Lebih Pendek dapat menghasilkan lebih banyak siklus pembelajaran dan membatasi risiko biaya dan upaya ke dalam *time frame* yang lebih kecil. Setiap *Sprint* dapat dianggap sebagai proyek pendek.

Berbagai praktik dapat dilakukan untuk memperkirakan *progress*, seperti *burn-down*, *burn-up*, atau arus kumulatif. Meskipun terbukti bermanfaat, hal ini tidak dapat menggantikan pentingnya pengalaman. Dalam keadaan yang kompleks, apa yang akan terjadi tidak dapat diketahui. Hanya apa yang telah terjadi yang dapat digunakan untuk pengambilan keputusan penting ke depan.

Sprint dapat dibatalkan jika tujuan *Sprint* tidak terpakai. Hanya Pemilik Produk yang memiliki kewenangan untuk membatalkan *Sprint*.

2. Perencanaan *Sprint*

Perencanaan *Sprint* dimulai dengan mengatur pekerjaan yang akan dilakukan untuk *Sprint*. Rencana yang dihasilkan ini dibuat oleh kerja kolaboratif dari seluruh Tim Scrum.

Pemilik Produk memastikan bahwa Tim Scrum dipersiapkan untuk membahas *Product Backlog items* yang paling penting dan bagaimana mereka memetakannya ke Tujuan Produk. Tim Scrum juga dapat mengundang orang lain untuk menghadiri perencanaan *Sprint* untuk memberikan masukan.

3. Scrum harian (*Daily Scrum*)

Tujuan dari Scrum harian adalah untuk memeriksa *progress* untuk mencapai tujuan *Sprint* dan menyesuaikan *Sprint Backlog* seperlunya dengan

menyesuaikan rencana kerja yang akan datang.

Scrum harian adalah kegiatan 15 menit untuk Pengembang Tim Scrum dalam mengurangi kerumitan. Scrum harian diadakan di waktu dan tempat yang sama setiap hari kerja *Sprint*. Jika Pemilik Produk atau Scrum Master secara aktif mengerjakan item di *Sprint Backlog*, mereka berpartisipasi sebagai Pengembang.

Pengembang dapat memilih struktur dan teknik apa pun yang mereka inginkan, selama Scrum harian mereka berfokus pada *progress* untuk mencapai tujuan *Sprint* dan menghasilkan rencana pekerjaan untuk hari kerja berikutnya.

Scrum harian dapat meningkatkan komunikasi, mengidentifikasi hambatan, mendorong pengambilan keputusan yang cepat, dan menghilangkan kebutuhan untuk rapat yang tidak perlu.

Scrum harian bukan satu-satunya kesempatan bagi Pengembang untuk menyesuaikan rencana mereka. Mereka sering bertemu sepanjang hari untuk diskusi yang lebih mendetail tentang mengadaptasi atau merencanakan ulang sisa pekerjaan pada *Sprint*.

4. pembahasan *Sprint* (*Sprint Review*)

Tujuan dari pembahasan *Sprint* adalah untuk memeriksa hasil dari penggerjaan *Sprint* dan menentukan adaptasi kedepannya. Tim Scrum mempresentasikan hasil kerja mereka kepada *stakeholder* utama dan mendiskusikan *progress* untuk mencapai Tujuan Produk.

Selama kegiatan berlangsung, Tim Scrum dan *stakeholder* meninjau apa yang telah dicapai di *Sprint* dan apa yang telah berubah di sekitar mereka. Berdasarkan informasi ini, Tim Scrum berkolaborasi tentang apa yang harus dilakukan selanjutnya. *Product Backlog* juga dapat disesuaikan untuk memenuhi keputusan baru. Pembahasan *Sprint* adalah sesi di mana Tim Scrum harus menghindari pembatasan hanya dalam presentasi, dalam arti harus aktif berdis-

kusi.

Pembahasan *Sprint* adalah kegiatan kedua dari terakhir *Sprint* dan memiliki batas waktu maksimum empat jam untuk *Sprint* satu bulan. Untuk *Sprint* yang lebih cepat, kegiatannya biasanya lebih singkat.

5. *Sprint* Retrospektif

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

Tim Scrum memeriksa bagaimana *Sprint* terakhir berjalan yang berkaitan dengan individu, interaksi, proses, alat, dan "*Definition of Done*" mereka. Element yang diinspeksi sering kali berbeda dengan domain pekerjaan. Asumsi Tim Scrum yang kurang tepat diidentifikasi dan asal-usulnya dieksplorasi. Tim Scrum membahas apa yang berjalan dengan baik selama *Sprint*, masalah apa yang dihadapi, dan bagaimana masalah tersebut (atau tidak) diselesaikan.

Tim Scrum mengidentifikasi perubahan yang paling berguna untuk meningkatkan efektivitasnya. Perbaikan yang paling berdampak akan ditangani secepat mungkin. itu semua bahkan dapat ditambahkan ke dalam *Sprint Backlog* untuk *Sprint* berikutnya.

Sprint Retrospektif merupakan bagian akhir dari *Sprint*. Batas waktunya maksimal tiga jam untuk *Sprint* satu bulan. Untuk *Sprint* yang lebih cepat, kegiatannya biasanya lebih singkat.

2.13.4 Komponen Scrum (*Scrum Artifacts*)

Komponen Scrum mewakili pekerjaan atau nilai. Mereka dirancang untuk memaksimalkan transparansi informasi utama. Jadi, setiap orang yang memeriksanya memiliki dasar yang sama untuk adaptasi.

Setiap Komponen berisi komitmen untuk memastikannya memberikan informasi yang dapat meningkatkan transparansi dan fokus yang dapat diukur *progress*-

nya:

- Untuk *Product Backlog*, yaitu Tujuan Produk.
- Untuk *Sprint Backlog*, yaitu Tujuan *Sprint*.
- untuk Peningkatan (*Increment*), yaitu "*Definition of Done*".

Komitmen ini ada untuk memperkuat pengalaman dan nilai-nilai Scrum bagi Tim Scrum dan *stakeholder* mereka.

1. *Product Backlog*

Product Backlog adalah sebuah *list* yang dibutuhkan untuk meningkatkan produk. Ini adalah satu-satunya pekerjaan yang dilakukan oleh Tim Scrum untuk hal tersebut.

Product Backlog items yang dapat dilakukan oleh Tim Scrum dalam satu *Sprint* dianggap siap untuk dipilih dalam kegiatan perencanaan *Sprint*. Mereka biasanya memperoleh tingkat transparansi ini setelah melakukan aktivitas perbaikan. Perbaikan *Product Backlog* adalah tindakan untuk memecah dan mendefinisikan lebih lanjut *Product Backlog items* menjadi item yang lebih kecil dan lebih tepat. Ini adalah aktivitas berkelanjutan untuk menambahkan detail, seperti deskripsi, urutan, dan ukuran.

Pengembang yang akan melakukan pekerjaan tersebut bertanggung jawab atas pengukuran tersebut. Pemilik Produk dapat memengaruhi Pengembang dengan membantu mereka untuk memahami dan memilih.

2. *Sprint Backlog*

Sprint Backlog terdiri dari tujuan *Sprint* (mengapa), *Product Backlog items* yang dipilih untuk *Sprint* (apa), serta rencana yang dapat ditindaklanjuti untuk menyampaikan Peningkatan (*Increment*) (bagaimana).

Sprint Backlog adalah rencana oleh dan untuk Pengembang. Ini adalah gambaran real-time yang sangat terlihat dari pekerjaan yang direncanakan Pengembang selama *Sprint* untuk mencapai tujuan *Sprint*. Akibatnya, *Sprint Backlog* diperbarui sepanjang berjalannya *Sprint* seiring dengan semakin banyaknya hal yang dipelajari. Ini harus cukup detail sehingga mereka dapat memeriksa seluruh *progress* mereka di Scrum Harian.

3. Peningkatan (*Increment*)

Peningkatan (*Increment*) adalah batu loncatan untuk mencapai Tujuan Produk. Setiap peningkatan merupakan tambahan untuk semua Penambahan sebelumnya dan diverifikasi secara menyeluruh, memastikan bahwa semua Penambahan berfungsi secara maksimal.

Beberapa Penambahan dapat dibuat di dalam *Sprint*. Jumlah seluruh penambahan disajikan di pembahasan *Sprint*. Namun, Peningkatan (*Increment*) dapat dikirimkan ke *stakeholder* sebelum *Sprint* selesai. Pembahasan *Sprint* tidak boleh dianggap sebagai acuan untuk penilaian [Guides (2020)].

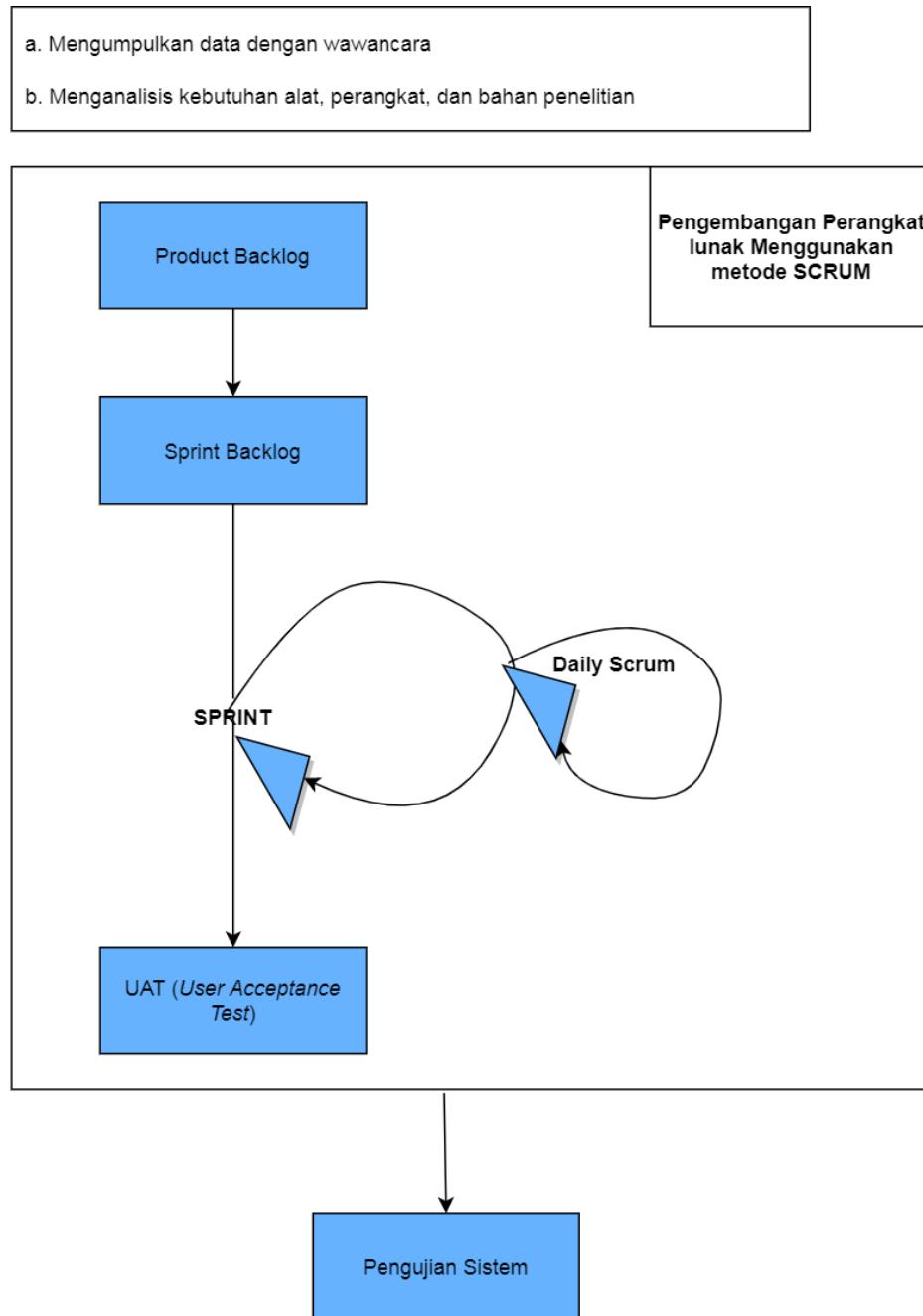
BAB III

Metode Penelitian

Untuk memudahkan penulis dalam melakukan penelitian, dibutuhkan desain penelitian. Adapun tahapan-tahapan dalam desain penelitian yang dilakukan penulis dalam proses penelitian skripsi yang berjudul "Rancang Bangun Web Service dan Website sebagai *Storage Engine* dan Monitoring Data Sensing untuk Budidaya Ikan Air Tawar" dapat dilihat pada **Gambar 3.1** yang menunjukkan rencana atau struktur penelitian yang digunakan untuk memecahkan permasalahan yang diangkat dalam penelitian ini.

Tahapan-tahapan desain penelitian yang akan dilakukan pertama-tama adalah mengumpulkan dan mempelajari data-data indikator air di dalam budidaya ikan air tawar yang nantinya akan dipakai untuk pengujian fungsi penyimpanan database sistem. Selain itu dikumpulkan juga data tentang menode pengembangan sistem

Setelah data-data sudah mencukupi barulah mempersiapkan alat dan bahan yang akan digunakan untuk penelitian, alat dan bahan akan dijabarkan pada subbab "alat dan bahan". Seletah alat dan bahan dipersiapkan, barulah masuk ke tahapan pengembangan perangkat lunak.



Gambar 3.1: Desain Penelitian

Pengembangan sistem menggunakan metode Scrum. Pengembangan akan dilakukan sesuai dengan tahapan standar Scrum yang didapat dari pengumpulan data.

Setelah sistem selesai dikembangkan, dilakukan pengujian. Pengujian dilakukan menggunakan metode *black box testing* dan menggunakan *hosting* sebagai ling-

kungan pengujian.

Penjelasan detail terkait desain di **Gambar 3.1** akan dijelaskan secara rinci dan urut pada sub-bab berikut

3.1 Pengumpulan Data

Data diambil dari wawancara dengan pemilik farm sekaligus klien dari penelitian ini sendiri. Wawancaranya bertopik pengembangan sistem untuk mendukung budidaya ikan di lokasi dengan pertanyaan-pertanyaan seputar pengembangan sistem. Untuk Transkrip wawancaranya tersedia pada **Lampiran A**.

3.2 Analisis Kebutuhan

Kebutuhan dibagi dalam dua bagian yaitu kebutuhan perangkat lunak, dan perangkat keras.

3.2.1 Kebutuhan Perangkat Keras

Perangkat keras yang dibutuhkan untuk membuat system ini adalah sebagai berikut:

1. PC Intel i5-4460 quad core
2. Memory 4gb
3. LCD Monitor
4. Hotspot Portable

Kebutuhan diatas didasarkan pada kebutuhan spesifikasi sistem yang akan dirancang. Kebutuhan perangkat keras diatas sudah memenuhi syarat untuk menjalankan XAMPP server untuk server localnya dan phpmyadmin untuk server database localnya. Sedangkan kebutuhan untuk coding yaitu Notepad++ yang sangat ringan.

3.2.2 Kebutuhan Perangkat Lunak

Perangkat lunak yang digunakan untuk membuat sistem ini adalah

1. Windows 7 64-bit

Windows 7 64-bit sebagai sistem operasi dikarenakan support untuk kebutuhan perancangan sistem yang akan dilakukan.

2. XAMPP

XAMPP sebagai server lokal untuk sistem yang akan dirancang.

3. InnoDB

InnoDB sebagai *storage engine* untuk sistem yang akan dirancang.

4. Notepad ++

Notepad++ sebagai *coding tool* untuk sistem yang akan dirancang.

3.3 Perancangan sistem

Perancangan sistem di dalam penelitian ini menggunakan metode Scrum. Sesuai dengan desain penelitian di **Gambar 3.1** terdapat komponen - komponen dalam metode scrum yang akan dijelaskan secara detail di dalam *section* ini yaitu Produk Backlog, *Sprint Backlog*, *Sprint*, *Daily Scrum*, dan *Deploy*.

3.3.1 Produk Backlog

Setelah Melakukan Pengumpulan Kebutuhan sistem, maka dari kebutuhan sistem tersebut dibuat Product Backlog. Berikut merupakan produk backlog dalam penelitian ini.

Tabel 3.1: Produk Backlog

NO	Story	Role	Sprint	Status	Priority
1	Kolam yang sudah ada di tempat penelitian dapat teregistrasi didalam sistem	Admin	1,2	Completed	Penting
2	Autentikasi untuk Admin	Admin			Penting
3	Dapat meregistrasi User	Admin			Penting
4	Autentikasi untuk User/Pengelola	User/Pengelola			Penting
5	Kolam yang sudah teregistrasi didalam sistem dapat dilihat semua dalam bentuk list.	User/Pengelola			Penting
6	Kolam yang sudah ada dalam list dapat diubah nama dan lokasi kolamnya. Kolam juga dapat dihapus	Admin			Penting
7	Membuat struktur data sensor yang akan di baca oleh web service	Sistem			Penting
8	Web-service dapat membaca data sensor yang telah dikirim.	Sistem			Penting
9	Dapat melihat history pembacaan sensor per 5 menit, per hari, per bulan ketika melihat detail.	User/Pengelola			Penting
10	Dapat melihat history pembacaan sensor, dari masing-masing jenis sensor per beberapa hari kebelakang. Dalam bentuk min, max, dan rata-rata dari seluruh reading sensor pada hari dan bulan tersebut per-satu jenis sensor	User/Pengelola			Penting
11	Dapat menambahkan informasi jenis ikan yang sedang dibudidayakan di kolam tertentu dengan menambahkan detail terkait: Jenis ikan, Jumlah ikan, Tanggal ikan masuk, Berat gram saat ikan masuk. Dapat memfasilitasi lebih dari satu jenis ikan jika dibutuhkan.	User/Pengelola			Penting
12	Data indikator kolam dapat dilihat dalam bentuk Chart dan diupdate secara realtime dengan jangka waktu 5 menit per-update	User/Pengelola			Penting

Dari **Tabel 3.1** di atas, Produk Backlog terdiri dari lima komponen yaitu, *Story*, *Role*, *Sprint*, *Status*, dan *Priority*. *Story* merupakan sebuah pekerjaan besar yang nantinya akan dicacah menjadi bagian *task* kecil di dalam *Sprint*. Di dalam penelitian ini terdapat dua belas *Story*. Untuk *role* yaitu oleh siapa fungsi yang akan dibuat akan dipakai. *Role* dalam penelitian ini dibagi menjadi tiga yaitu Admin, User atau Pengelola, dan Sistem. Untuk bagian *sprint* ini untuk menandakan di *sprint* berapa *story* tersebut akan dilaksanakan. Untuk *status* yaitu untuk menandakan apakah *story* sudah diselesaikan atau belum. Untuk *priority* yaitu untuk menandakan yang mana *story* yang harus dikerjakan terlebih dahulu.

3.3.2 *Sprint Backlog*

Sprint Backlog merupakan perencanaan sprint dari awal hingga akhir yang keputusannya diambil berdasarkan Produk Backlog. Perancangan sistem di dalam penelitian ini terdapat tujuh iterasi *sprint* yang telah direncanakan yaitu dari *sprint-1* hingga *sprint-7*. Berikut perancanaan *sprint*-nya

1. *Sprint-1*

Sprint-1 dilaksanakan selama satu minggu yaitu pada tanggal 7 sampai 13 juni 2021

2. *Sprint-2*

Sprint-2 dilaksanakan selama satu minggu yaitu pada tanggal 14 sampai 20 juni 2021

3. *Sprint-3*

Sprint-3 dilaksanakan selama satu minggu yaitu pada tanggal 21 sampai 27 juni 2021

4. *Sprint-4*

Sprint-4 dilaksanakan selama satu minggu yaitu pada tanggal 28 juni sampai 4 juli 2021

5. *Sprint-5*

Sprint-5 dilaksanakan selama satu minggu yaitu pada tanggal 5 sampai 11 juli 2021

6. *Sprint-6*

Sprint-6 dilaksanakan selama satu minggu yaitu pada tanggal 12 sampai 18 juli 2021

7. *Sprint-7*

Sprint-7 dilaksanakan selama satu minggu yaitu pada tanggal 19 sampai 25 juli 2021

3.3.3 *Sprint*

Setelah dilakukan perencanaan pada *Sprint Backlog*, maka dalam penggerjaan *sprint* ini harus mengikuti jadwal penggerjaan yang telah ditentukan oleh *Sprint Backlog*. Di dalam *sprint* ini terdapat berbagai *task* kecil yang sudah dipecah dari *story* yang ada di Produk Backlog.

3.3.4 *Daily Scrum*

Pada akhir pekan diakhir penggerjaan setiap *sprint* yaitu di hari minggu, Akan dilaksanakan evaluasi mengenai perkembangan dan hambatan selama periode satu minggu penggerjaan setiap *sprint*.

3.3.5 *Deploy*

Setelah seluruh pekerjaan *sprint* yang telah direncanakan pada *sprint backlog* selesai, Sistem akan di *Deploy* di sebuah *hosting* yang nantinya berfungsi untuk UAT (*User Acceptance Test*).

3.4 UAT (*User Acceptance Test*)

UAT (*User Acceptance Test*) dilakukan dengan menggunakan metode *black box*. UAT (*User Acceptance Test*) dilakukan secara daring melalui aplikasi *Skype* dan sistem diuji melalui *hosting* sementara dan aplikasi *Postman*. Pengujian UAT (*User Acceptance Test*) dilakukan selama dua kali yaitu tanggal 28 Juli 2021 dan tanggal 30 Juli 2021. Pengujian UAT (*User Acceptance Test*) dilakukan dengan *requirement* dan skenario pengujian sebagai berikut:

Tabel 3.2: Skenario Pengujian Admin

Uji Fitur	Detail Pengujian	Jenis Pengujian
Login	Mengisi form login dengan username dan password admin lalu submit	<i>Black box</i>
Registrasi User	Mengisi form Registrasi user lalu submit	<i>Black box</i>
Registrasi Kolam	Mengisi form Registrasi kolam lalu submit	<i>Black box</i>
List kolam	Menampilkan data kolam yang telah di registrasi	<i>Black box</i>
Edit Kolam	Mengklik tombol edit kolam pada list kolam tertentu, isi form edit kolam lalu submit	<i>Black box</i>
Button delete kolam	Mengklik tombol delete kolam lalu klik OK	<i>Black box</i>
Form Tambah Ikan	Mengisi form tambah ikan di kolam yang dipilih lalu submit	<i>Black box</i>

Tabel 3.3: Skenario Pengujian User

Uji Fitur	Detail Pengujian	Jenis Pengujian
Login	Mengisi form login dengan username dan password user lalu submit	<i>Black box</i>
List kolam	Menampilkan data kolam yang telah di registrasi	<i>Black box</i>
Histori pembacaan sensor per-kolam	Menampilkan data sensor per kolam dalam bentuk tabel dan dapat dipilih berdasarkan bulan atau tanggal	<i>Black box</i>
Histori penghitungan min, max, average sensor per-kolam	Menampilkan data penghitungan sensor per hari dalam bentuk tabel dan dapat dipilih berdasarkan bulan atau tanggal	<i>Black box</i>
List ikan per kolam	Menampilkan data ikan yang telah ditambahkan	<i>Black box</i>
Chart realtime sensor per-kolam	Menampilkan data sensor per kolam dalam bentuk chart realtime.	<i>Black box</i>

BAB IV

Hasil Dan Pembahasan

4.1 Perancangan Sistem

Perancangan Web service dan Website sebagai *Storage Engine* dan *Monitoring Data Sensing* untuk Budidaya Ikan Air Tawar menggunakan metode Scrum. Dalam metode Scrum ini pengembangan sistem dilakukan dengan iterasi dalam setiap *Sprint* yang urutan pelaksanaan *Sprint*-nya terdapat pada *Product Backlog*. Berikut langkah-langkah yang dilakukan pada proses pengembangan sistem:

4.1.1 *Sprint reports iteration*

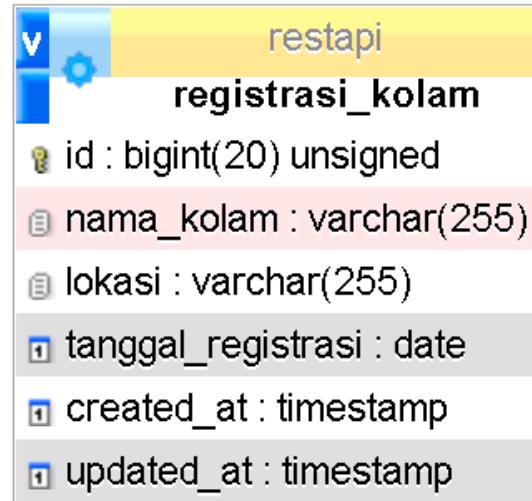
Di dalam *sprint reports* ini akan menjelaskan iterasi progress pelaksanaan *sprint* dari *sprint-1* hingga *sprint-7*.

1. *Sprint-1*

Tabel 4.1: *Sprint-1*

NO	Story	Task	Status
1	Kolam yang sudah ada di tempat penelitian dapat teregristrasi didalam sistem	Membuat desain MVC	completed
2	Kolam yang sudah ada di tempat penelitian dapat teregristrasi didalam sistem	Membuat desain database untuk fungsi registrasi kolam	completed
3	Kolam yang sudah ada di tempat penelitian dapat teregristrasi didalam sistem	Membuat fungsi untuk registrasi kolam	completed

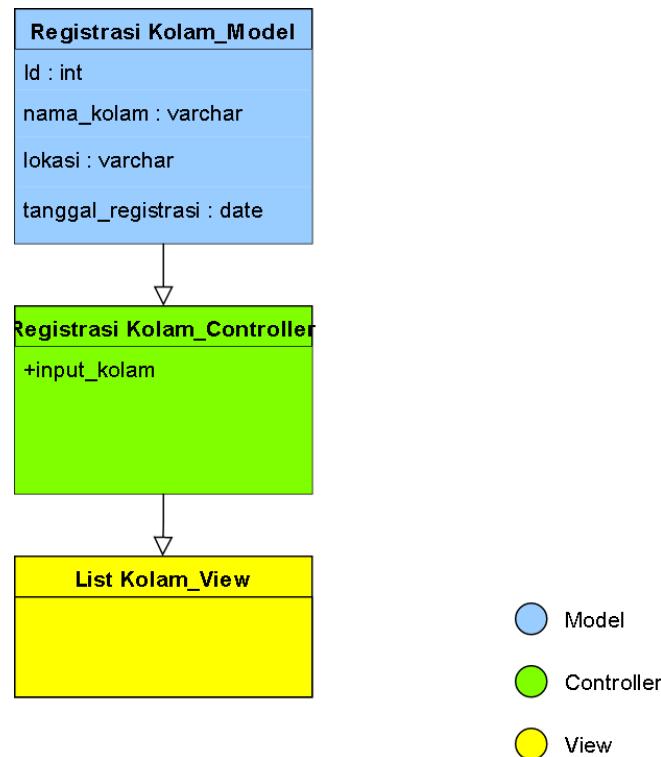
- Desain database



restapi	
registrasi_kolam	
id	: bigint(20) unsigned
nama_kolam	: varchar(255)
lokasi	: varchar(255)
tanggal_registrasi	: date
created_at	: timestamp
updated_at	: timestamp

Gambar 4.1: Desain Database *Sprint-1*

- MVC (*Model View Controller*)



Gambar 4.2: Desain MVC *sprint-1*

- Registrasi Kolam web service (*input*)

The screenshot shows the Postman interface with a 'New Request' button. The method is set to 'POST' and the URL is 'http://dprompt.id/restapi/public/kolam/create'. The 'Body' tab is selected, showing the following JSON input:

```

1  {
2    "id": 3,
3    "nama_kolam": "Kolam 3",
4    "lokasi": "Teras Bawah",
5    "tanggal_registrasi": "2021-08-08"
6  }
7

```

Below the body, there are tabs for Params, Authorization, Headers (10), Pre-request Script, Tests, and Settings. The 'Body' tab is highlighted with a green dot.

Gambar 4.3: Registrasi Kolam web service (*input*)

- Registrasi Kolam web service (*output*)

The screenshot shows the Postman interface with the JSON response tab selected. The response is a JSON array containing three objects, each representing a kolam entry. The objects are numbered 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, and 32. The data includes fields like id, nama_kolam, lokasi, tanggal_registrasi, created_at, and updated_at.

```

12  {
13    "nama_kolam": "kolam2 edit",
14    "lokasi": "teras",
15    "tanggal_registrasi": "2020-12-12",
16    "created_at": "2021-07-18T01:29:49.000000Z",
17    "updated_at": "2021-07-26T21:20:28.000000Z"
18  },
19  {
20    "id": 3,
21    "nama_kolam": "Kolam 3",
22    "lokasi": "Teras Bawah",
23    "tanggal_registrasi": "2021-08-08",
24    "created_at": "2021-08-17T09:33:04.000000Z",
25    "updated_at": "2021-08-17T09:33:04.000000Z"
26  },
27  {
28    "id": 4,
29    "nama_kolam": "kolam 4 ",
30    "lokasi": "teras ",
31    "tanggal_registrasi": "2021-07-12",
32    "created_at": "2021-07-25T20:30:23.000000Z",
    "updated_at": "2021-07-28T22:58:29.000000Z"
}

```

Gambar 4.4: Registrasi Kolam web service (*output*)

- *Source code*

Untuk source code *sprint-1* terdapat di link github, untuk link githubnya yaitu : <https://github.com/Fadhilah24/Web-Service/tree/sprint-1>.

2. *Sprint-2*

Tabel 4.2: Sprint-2

NO	Story	Task	Status
1	Kolam yang sudah ada di tempat penelitian dapat teregristrasi didalam sistem	Membuat antar muka fungsi registrasi kolam	completed
2	Autentikasi untuk Admin dan User/Pengelola	Membuat desain MVC	completed
3	Autentikasi untuk Admin dan User/Pengelola	Membuat desain database	completed
4	Autentikasi untuk Admin dan User/Pengelola	Membuat fungsi login untuk admin	completed
5	Autentikasi untuk Admin dan User/Pengelola	Membuat antarmuka login	completed
6	Admin Dapat meregistrasi User	Membuat desain MVC	completed
7	Admin Dapat meregistrasi User	Membuat desain database	completed
8	Admin Dapat meregistrasi User	Membuat fungsi untuk registrasi user	completed
9	Admin Dapat meregistrasi User	Membuat antarmuka registrasi User	completed

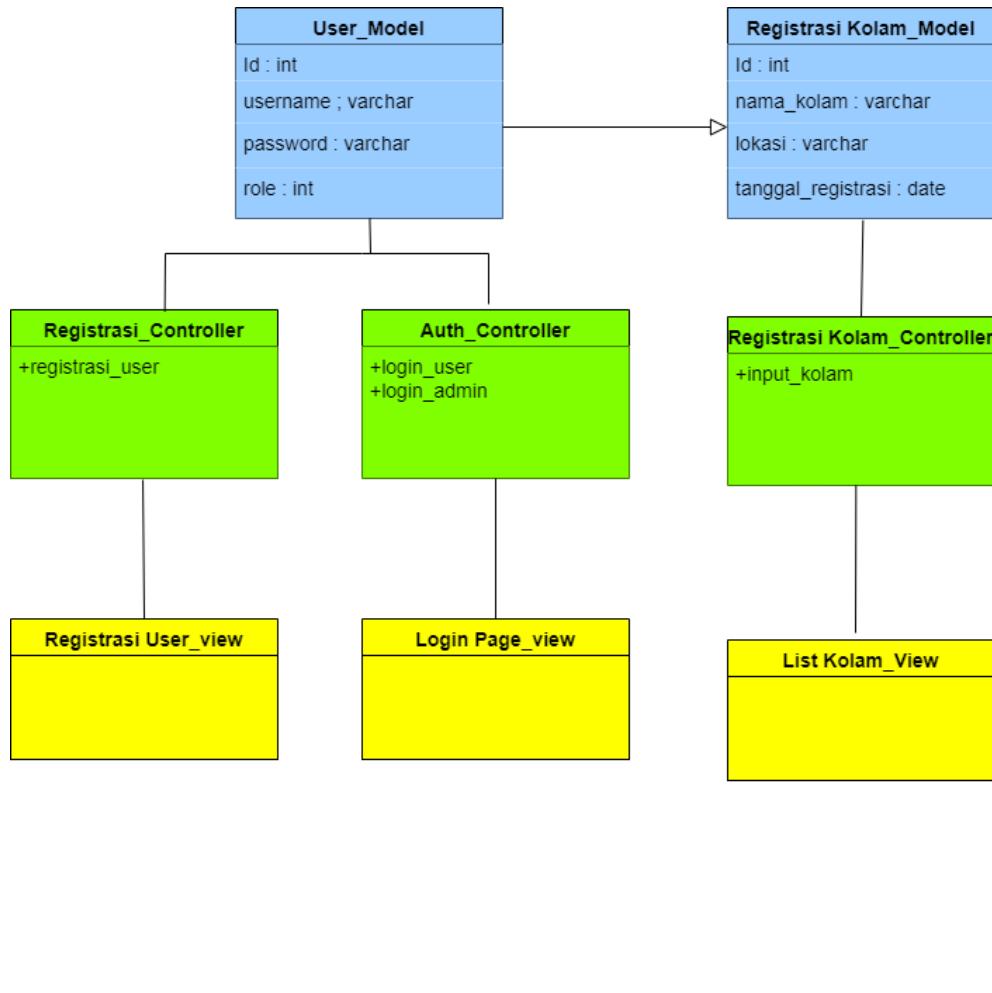
- Desain database

u8810863_restapijfarm users	
!	id : bigint(20) unsigned
!	username : varchar(255)
!	password : varchar(255)
#	role : int(11)
!	token : varchar(255)
!	created_at : timestamp
!	updated_at : timestamp

u8810863_restapijfarm registrasi_kolam	
!	id : bigint(20) unsigned
!	nama_kolam : varchar(255)
!	lokasi : varchar(255)
!	tanggal_registrasi : date
!	token : varchar(255)
!	created_at : timestamp
!	updated_at : timestamp

Gambar 4.5: Desain Database *Sprint-2*

- MVC (*Model View Controller*)



Gambar 4.6: Desain MVC *sprint-2*

- Login Admin web service (*input*)

The screenshot shows a POST request to `http://dprompt.id/restapi/public/login`. The 'Body' tab is selected, showing the following JSON input:

```
1 {
2   "username" : "3145163442",
3   "password" : "vcg282ei58"
4 }
```

Gambar 4.7: Login Admin web service (*input*)

- Login Admin web service (*output*)

The screenshot shows the JSON response from the login request. The 'Body' tab is selected, displaying the following JSON output:

```
1 {
2   "message": "login_success",
3   "code": 200,
4   "result": {
5     "token": "hfnP4N60DAjuXzR350QHYpFqk4rtt2IjLhN",
6     "role1": 2
7   }
}
```

Gambar 4.8: Login Admin web service (*output*)

- Login User web service (*input*)

The screenshot shows a POST request to `http://dprompt.id/restapi/public/login`. The 'Body' tab is selected, showing the following JSON input:

```

1 {
2   "username": "fadhilahph",
3   "password": "vcg282ei58"
4 }

```

Gambar 4.9: Login User web service (*input*)

- Login User web service (*output*)

The screenshot shows the JSON output of the login request. The 'Body' tab is selected, displaying the following response:

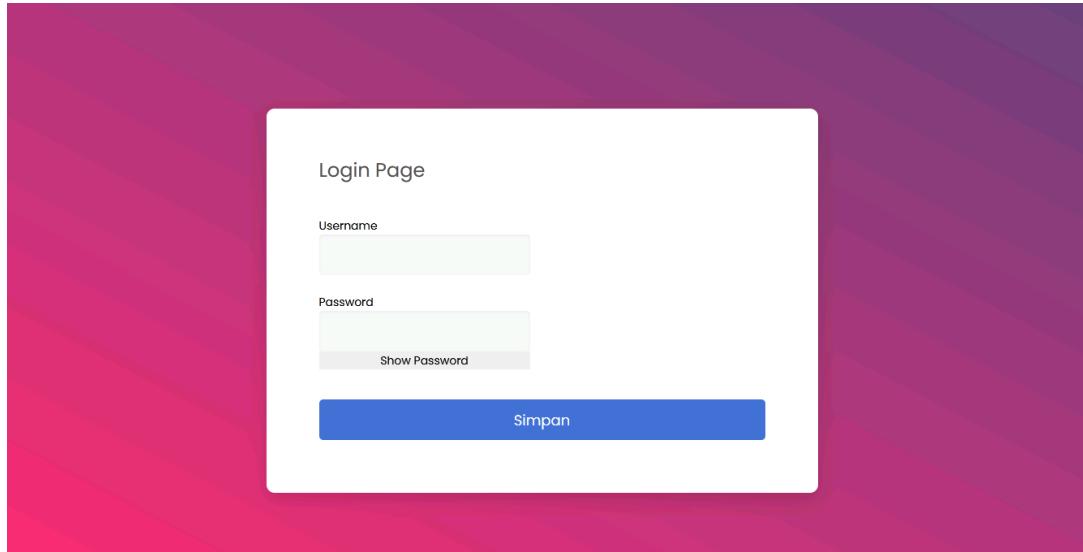
```

1 {
2   "message": "login_success",
3   "code": 200,
4   "result": {
5     "token": "rGJNZRtv1xX1BvDJKZvT0ZJUJQdYYqRigb0",
6     "role": 1
7   }

```

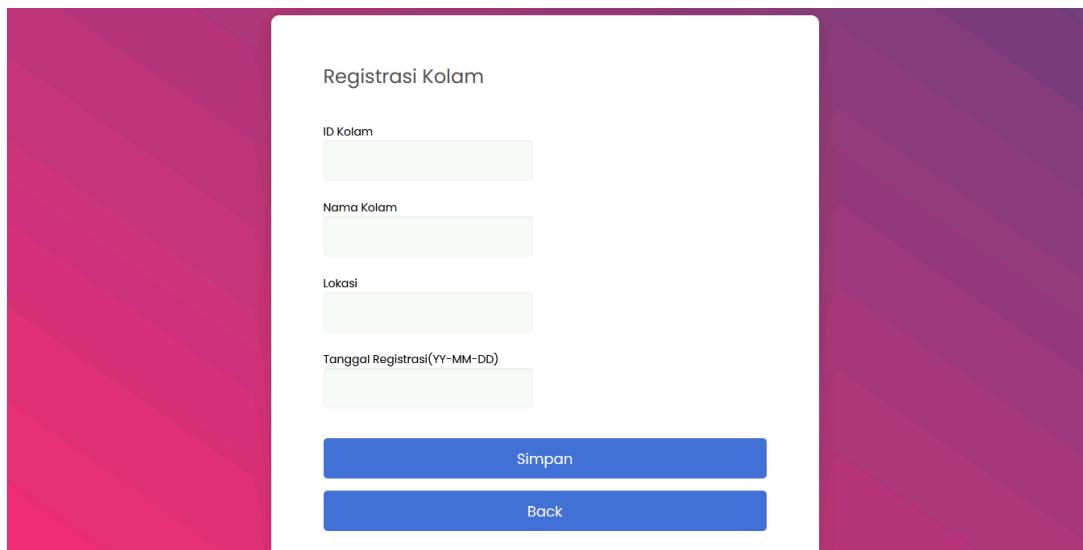
Gambar 4.10: Login User web service (*output*)

- Tampilan Login



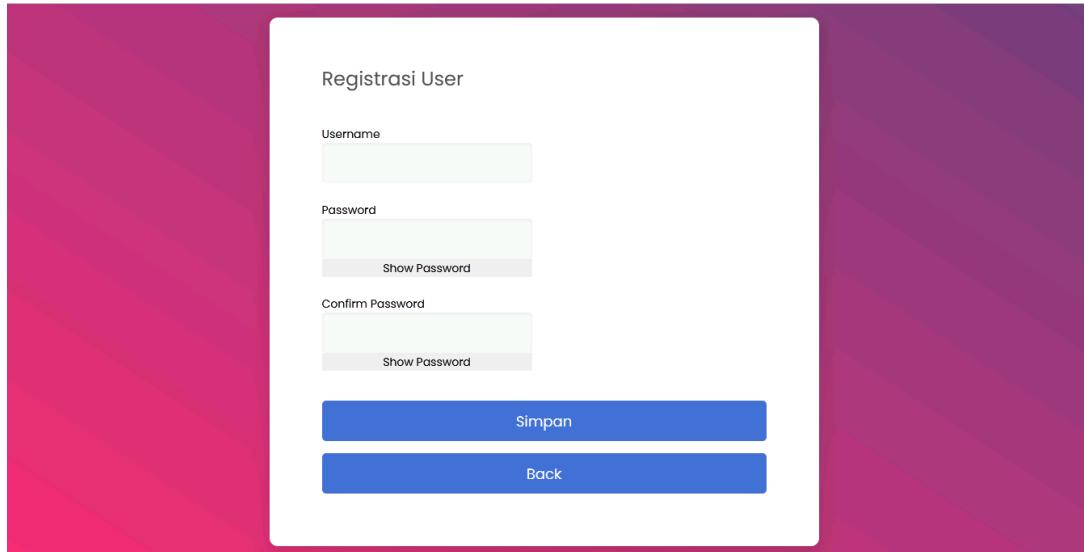
Gambar 4.11: Tampilan Login

- Registrasi Kolam



Gambar 4.12: Tampilan registrasi kolam

- Registrasi user



Gambar 4.13: Tampilan registrasi user

- *Source code*

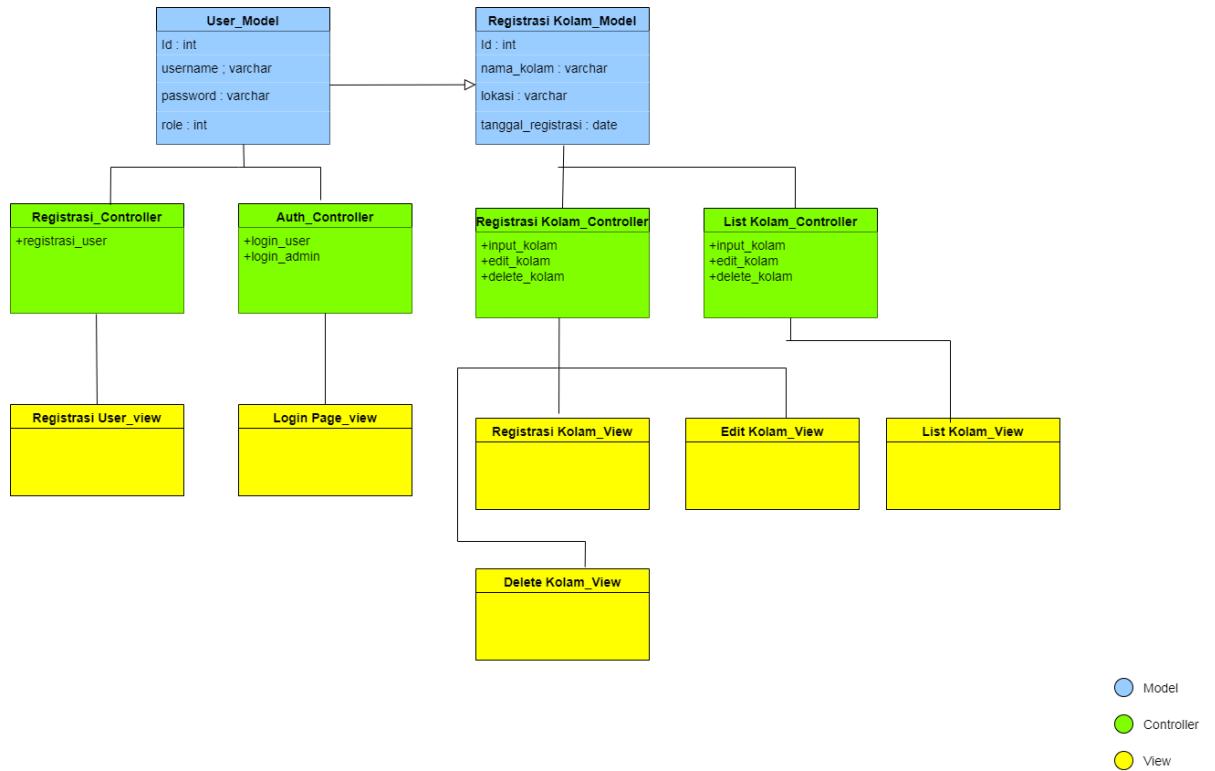
Untuk source code *sprint-2* terdapat di link github, untuk link githubnya yaitu : <https://github.com/Fadhilah24/Web-Service/tree/sprint-2>.

3. Sprint-3

Tabel 4.3: Sprint-3

NO	Story	Task	Status
1	Kolam yang sudah teregistrasi didalam sistem dapat dilihat semua dalam bentuk list.	Membuat desain MVC	completed
2	Kolam yang sudah teregistrasi didalam sistem dapat dilihat semua dalam bentuk list.	Membuat antarmuka untuk menampilkan list kolam	completed
3	Kolam yang sudah ada dalam list dapat diubah nama dan lokasi kolamnya. Kolam juga dapat dihapus	Membuat desain MVC	completed
4	Kolam yang sudah ada dalam list dapat diubah nama dan lokasi kolamnya. Kolam juga dapat dihapus	Membuat fungsi untuk mengedit dan menghapus kolam	completed
5	Kolam yang sudah ada dalam list dapat diubah nama dan lokasi kolamnya. Kolam juga dapat dihapus	Membuat antarmuka untuk mengedit dan menghapus kolam	completed

- **MVC (Model View Controller)**



Gambar 4.14: Desain MVC *sprint-3*

- **List Kolam**

J-FARM
List Kolam
Registrasi User
Registrasi Kolam
Logout

List Kolam

ID Kolam	Nama Kolam	Lokasi	Tanggal Registrasi	Edit	Delete	Details
1	kolam1	teras	2020-12-12	Edit	Delete	Details
2	kolam2 edit	teras	2020-12-12	Edit	Delete	Details
4	kolam 4	teras	2021-07-12	Edit	Delete	Details
5	kolam5	teras1	2021-07-12	Edit	Delete	Details
6	kolam6	teras	2021-07-23	Edit	Delete	Details
9	Kolam 9	Teras bawah	2021-07-07	Edit	Delete	Details

Gambar 4.15: Tampilan list kolam

- Update kolam web service (*input*)

The screenshot shows a POSTMAN interface. The method is set to PUT, and the URL is `http://dprompt.id/restapi/public/kolam/updatekolam/3`. The 'Body' tab is selected, and the content type is set to JSON. The JSON payload is:

```
1 {  
2   "nama_kolam": "Kolam 3 edit",  
3   "lokasi": "Teras Bawah"  
4 }  
5
```

Gambar 4.16: Update kolam web service (*input*)

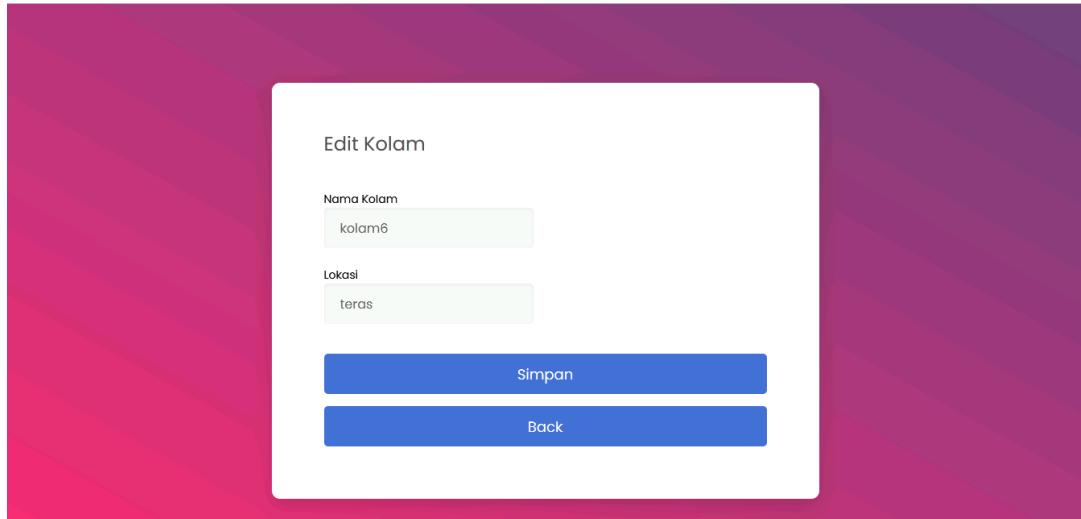
- Update kolam web service (*output*)

The screenshot shows a POSTMAN interface with the 'Body' tab selected. The status bar indicates a successful 200 OK response with 263 ms latency and 618 B size. The JSON response is:

```
1 {  
2   "id": 3,  
3   "nama_kolam": "Kolam 3 edit",  
4   "lokasi": "Teras Bawah",  
5   "tanggal_registrasi": "2021-08-08",  
6   "created_at": "2021-08-17T09:33:04.000000Z",  
7   "updated_at": "2021-08-17T09:54:38.000000Z"  
8 }
```

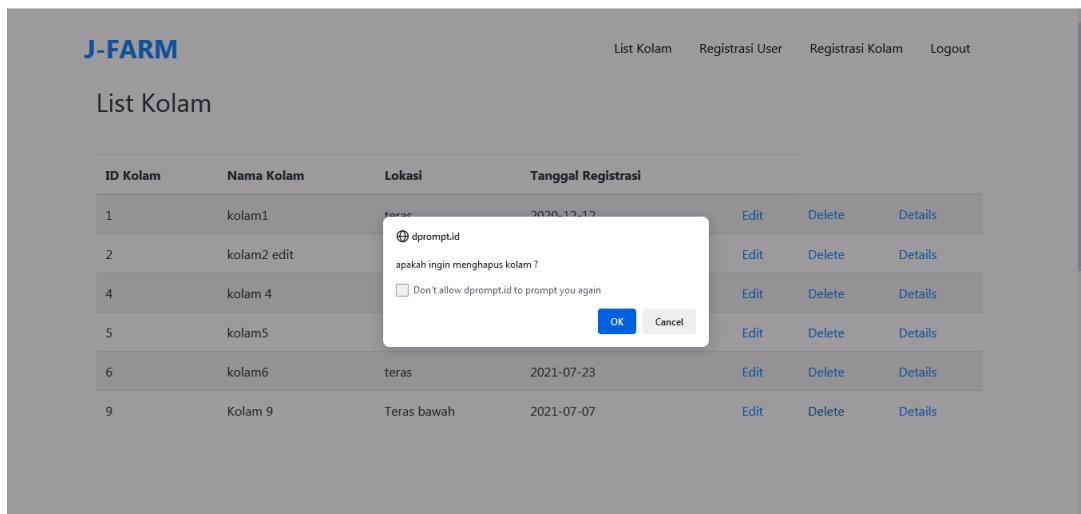
Gambar 4.17: Update kolam web service (*output*)

- *Edit Kolam*



Gambar 4.18: Tampilan edit kolam

- *Delete Kolam*



Gambar 4.19: Tampilan *delete kolam*

- *Source code*

Untuk source code *sprint-3* terdapat di link github, untuk link githubnya yaitu : <https://github.com/Fadhilah24/Web-Service/tree/sprint-3>.

4. Sprint-4

Tabel 4.4: Sprint-4

NO	Story	Task	Status
1	Membuat struktur data sensor yang akan dibaca oleh web service	Membuat desain terkait struktur data yang akan dibaca oleh web service	completed
2	Web-service dapat membaca data sensor yang telah dikirim.	data sensor yang dikirim kedalam sistem dapat dibaca oleh web service dan disimpan didalam database	todo

- Format data Json Sensor

```
{
    ph : value,
    oksigen : value,
    suhu : value
}
```

- Type data

```
{
    data : value
}
```

- id : int
- username : varchar
- password : varchar
- nama_kolam : varchar
- lokasi : varchar
- tanggal_registrasi : date
- idkolam : int
- jenisikan : varchar
- jumlah ikan : varchar
- tanggalikanmasuk : date
- ph : decimal
- oksigen : decimal
- suhu : decimal

5. Sprint-5

Untuk *sprint-5* tidak ada progress dikarenakan alasan tertentu.

4.1.2 Sprint-6

Tabel 4.5: Sprint-6

NO	Story	Task	Status
1	Web-service dapat membaca data sensor yang telah dikirim.	Membuat desain database untuk menyimpan data sensor yang dikirim	completed
2	Web-service dapat membaca data sensor yang telah dikirim.	data sensor yang dikirim kedalam sistem dapat dibaca oleh web service dan disimpan didalam database	completed

- Desain database

u8810863_restapijfarm users	
✓	id : bigint(20) unsigned
✗	username : varchar(255)
✗	password : varchar(255)
#	role : int(11)
✗	token : varchar(255)
✗	created_at : timestamp
✗	updated_at : timestamp

u8810863_restapijfarm datasensor	
✓	id : bigint(20) unsigned
✗	idkolam : bigint(20)
#	ph : decimal(20,2)
#	oksigen : decimal(20,2)
#	suhu : decimal(20,1)
✗	created_at : timestamp
✗	updated_at : timestamp

u8810863_restapijfarm registrasi_kolam	
✓	id : bigint(20) unsigned
✗	nama_kolam : varchar(255)
✗	lokasi : varchar(255)
✗	tanggal_registrasi : date
✗	token : varchar(255)
✗	created_at : timestamp
✗	updated_at : timestamp

Gambar 4.20: Desain Database *Sprint-6*

- POST data sensor web service (*input*).

The screenshot shows a POST request in Postman. The URL is `http://dprompt.id/restapi/public/sensor/updatesensor/2`. The Headers tab shows 9 items. The Body tab is selected and contains the following JSON input:

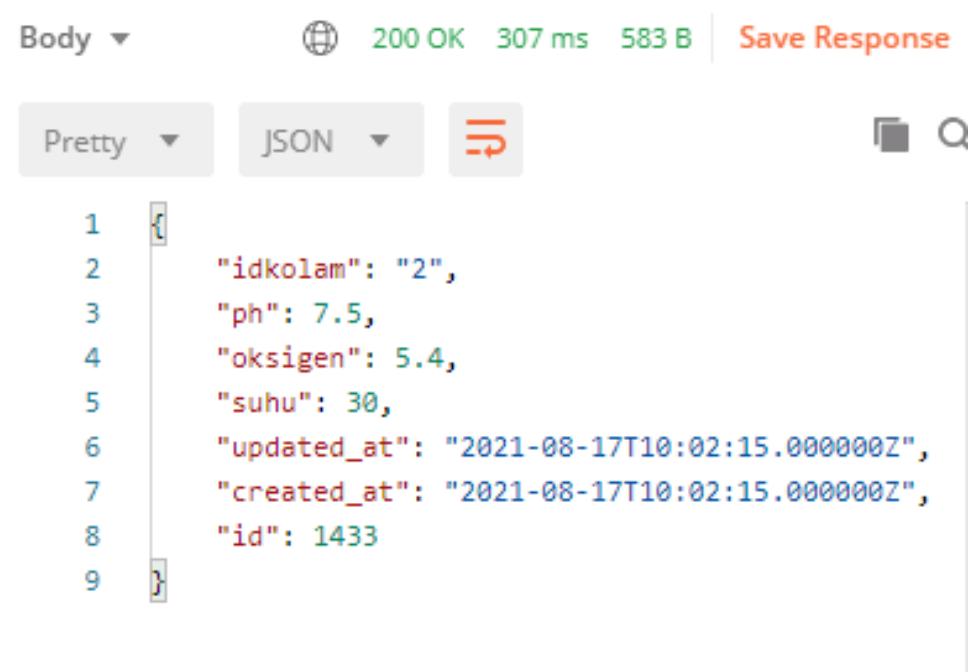
```

1  {
2    "ph": 7.5,
3    "oksigen": 5.4,
4    "suhu": 30
5
6 }

```

Gambar 4.21: POST data sensor web service (*input*)

- POST data sensor web service (*output*).



The screenshot shows a POST request response in Postman. At the top, it displays "Body" with a dropdown, a globe icon, "200 OK", "307 ms", "583 B", and a "Save Response" button. Below this, there are three tabs: "Pretty" (selected), "JSON" (with a dropdown), and a red "Raw" tab. The JSON response is displayed in a code editor-like format with line numbers:

```
1  {
2      "idkolam": "2",
3      "ph": 7.5,
4      "oksigen": 5.4,
5      "suhu": 30,
6      "updated_at": "2021-08-17T10:02:15.000000Z",
7      "created_at": "2021-08-17T10:02:15.000000Z",
8      "id": 1433
9 }
```

Gambar 4.22: POST data sensor web service (*output*)

Dari gambar di atas ditampilkan bahwa POST data sensor ke sistem berhasil dengan *status code* 200.

- *Source code*

Untuk source code *sprint-6* terdapat di link github, untuk link githubnya yaitu : <https://github.com/Fadhilah24/Web-Service/tree/sprint-6>.

6. Sprint-7

Tabel 4.6: Sprint-7

No	Story	Task	Status
1	Dapat membaca histori pembacaan sensor per 5 menit, per hari, per bulan ketika melihat detail	Membuat desain MVC	completed
2	Dapat membaca histori pembacaan sensor per 5 menit, per hari, per bulan ketika melihat detail	Membuat fungsi untuk mengambil data sensor dari database dan ditampilkan ke halaman detail	completed
3	Dapat membaca histori pembacaan sensor per 5 menit, per hari, per bulan ketika melihat detail	Membuat antarmuka detail	completed
4	Dapat melihat history pembacaan sensor, dari masing-masing jenis sensor per-beberapa hari kebelakang Dalam bentuk min, max, dan rata-rata dari seluruh reading sensor pada hari dan bulan tersebut per-satu jenis sensor	Membuat desain MVC	completed
5	Dapat melihat history pembacaan sensor, dari masing-masing jenis sensor per-beberapa hari kebelakang Dalam bentuk min, max, dan rata-rata dari seluruh reading sensor pada hari dan bulan tersebut per-satu jenis sensor	Membuat fungsi untuk menghitung rata-rata, max dan min sensor per beberapa hari kebelakang.	completed
6	Dapat melihat history pembacaan sensor, dari masing-masing jenis sensor per-beberapa hari kebelakang Dalam bentuk min, max, dan rata-rata dari seluruh reading sensor pada hari dan bulan tersebut per-satu jenis sensor	Membuat antarmuka untuk menampilkan rata-rata, max dan min sensor per beberapa hari kebelakang.	completed
7	Dapat menambahkan informasi jenis ikan yang sedang dibudidayakan di kolam tertentu dengan menambahkan detail terkait: Jenis ikan, Jumlah ikan, Tanggal ikan masuk, Berat gram saat ikan masuk. Dapat memfasilitasi lebih dari satu jenis ikan jika dibutuhkan.	Membuat fungsi untuk menambahkan informasi jenis ikan yang sedang dibudidayakan di kolam tertentu.	completed
8	Dapat menambahkan informasi jenis ikan yang sedang dibudidayakan di kolam tertentu dengan menambahkan detail terkait: Jenis ikan, Jumlah ikan, Tanggal ikan masuk, Berat gram saat ikan masuk. Dapat memfasilitasi lebih dari satu jenis ikan jika dibutuhkan.	Membuat antarmuka berupa form untuk menginput informasi jenis ikan.	completed
9	Dapat menambahkan informasi jenis ikan yang sedang dibudidayakan di kolam tertentu dengan menambahkan detail terkait: Jenis ikan, Jumlah ikan, Tanggal ikan masuk, Berat gram saat ikan masuk. Dapat memfasilitasi lebih dari satu jenis ikan jika dibutuhkan.	Membuat antarmuka berupa list table untuk menampilkan informasi jenis ikan pada kolam tertentu.	completed
10	Data indikator kolam dapat dilihat dalam bentuk Chart dan diupdate secara realtime dengan jangka waktu 5 menit per-update	Membuat desain MVC	completed
11	Data indikator kolam dapat dilihat dalam bentuk Chart dan diupdate secara realtime dengan jangka waktu 5 menit per-update	Membuat Chart untuk menampilkan sensor ph, DO, dan suhu.	completed

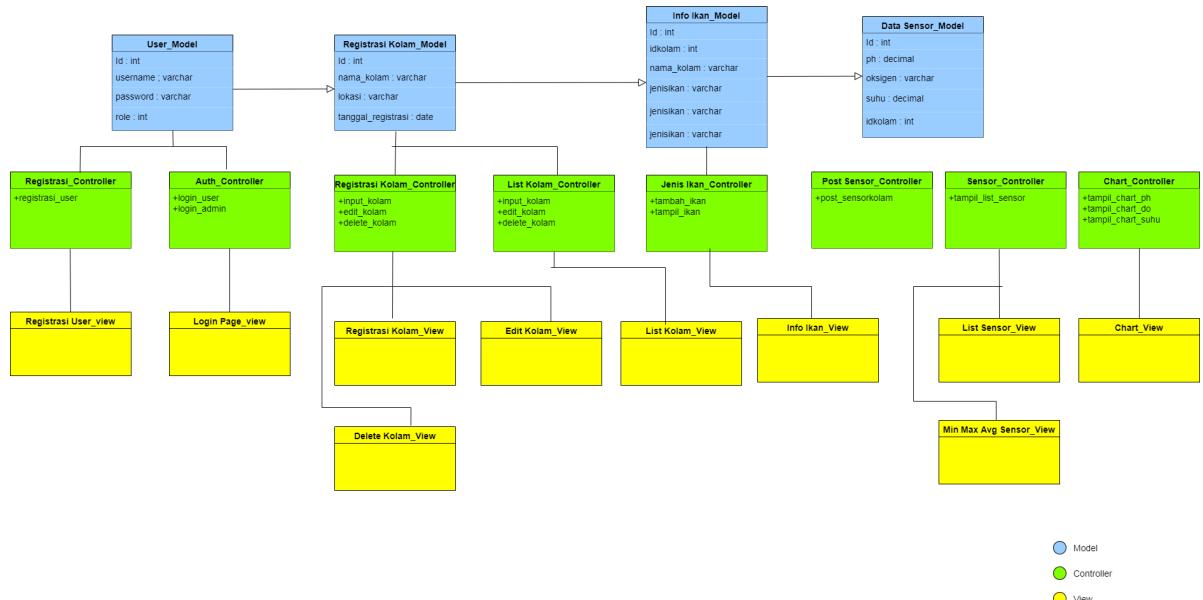
- Desain database

u8810863_restapijfarm users	u8810863_restapijfarm datasensor
id : bigint(20) unsigned	
username : varchar(255)	
password : varchar(255)	
role : int(11)	
token : varchar(255)	
created_at : timestamp	
updated_at : timestamp	

u8810863_restapijfarm registrasi_kolam	u8810863_restapijfarm infoikan
id : bigint(20) unsigned	
nama_kolam : varchar(255)	
lokasi : varchar(255)	
tanggal_registrasi : date	
token : varchar(255)	
created_at : timestamp	
updated_at : timestamp	

Gambar 4.23: Desain Database *Sprint-7*

- MVC (*Model View Controller*)



Gambar 4.24: Desain MVC *sprint-7*

- Histori Pembacaan sensor di halaman detail

ph	Do	Suhu	Waktu
7.10	5.20	30.0	2021-07-25 15:57:45
7.10	5.20	30.0	2021-07-25 15:59:35
7.10	5.20	30.0	2021-07-25 16:09:15
7.50	5.40	30.0	2021-07-25 16:09:20
7.40	5.40	28.0	2021-07-25 16:09:25
7.30	5.30	29.0	2021-07-25 16:09:31
7.30	5.30	28.0	2021-07-25 16:09:36
7.20	5.20	30.0	2021-07-25 16:09:41

Gambar 4.25: List histori pembacaan sensor

- *Min, max dan rata-rata sensor ph, DO dan suhu per hari dalam sebulan*

avg ph	min ph	max ph	Tanggal
7.26086956521739	7.5	7.1	7/25/2021
7.5	7.5	7.5	7/27/2021
6.989523809523814	8	6.1	7/28/2021
7.00800000000001	8	6.1	7/30/2021
7.5	7.5	7.5	8/1/2021

Gambar 4.26: *Min, max dan rata-rata sensor ph*

J-FARM																											
List Kolam	Registrasi User	Registrasi Kolam	Kelola Ikan ▾																								
Sensor ph																											
<input type="text" value="2021-07"/> <input type="button" value="OK"/> <input type="text" value="Pilih Tanggal"/> <input type="button" value="OK"/> <table border="1"> <thead> <tr> <th>avg DO</th> <th>min DO</th> <th>max DO</th> <th>Tanggal</th> </tr> </thead> <tbody> <tr> <td>5.286956521739131</td> <td>5.4</td> <td>5.2</td> <td>7/25/2021</td> </tr> <tr> <td>5.4</td> <td>5.4</td> <td>5.4</td> <td>7/27/2021</td> </tr> <tr> <td>5.29285714285714</td> <td>5.4</td> <td>5.2</td> <td>7/28/2021</td> </tr> <tr> <td>5.284</td> <td>5.4</td> <td>5.2</td> <td>7/30/2021</td> </tr> <tr> <td>5.4</td> <td>5.4</td> <td>5.4</td> <td>8/1/2021</td> </tr> </tbody> </table>				avg DO	min DO	max DO	Tanggal	5.286956521739131	5.4	5.2	7/25/2021	5.4	5.4	5.4	7/27/2021	5.29285714285714	5.4	5.2	7/28/2021	5.284	5.4	5.2	7/30/2021	5.4	5.4	5.4	8/1/2021
avg DO	min DO	max DO	Tanggal																								
5.286956521739131	5.4	5.2	7/25/2021																								
5.4	5.4	5.4	7/27/2021																								
5.29285714285714	5.4	5.2	7/28/2021																								
5.284	5.4	5.2	7/30/2021																								
5.4	5.4	5.4	8/1/2021																								

Gambar 4.27: Min, max dan rata-rata sensor DO

J-FARM																											
List Kolam	Registrasi User	Registrasi Kolam	Kelola Ikan ▾																								
Sensor ph																											
<input type="text" value="2021-07"/> <input type="button" value="OK"/> <input type="text" value="Pilih Tanggal"/> <input type="button" value="OK"/> <table border="1"> <thead> <tr> <th>avg suhu</th> <th>min suhu</th> <th>max suhu</th> <th>Tanggal</th> </tr> </thead> <tbody> <tr> <td>29.043478260869566</td> <td>30</td> <td>28</td> <td>7/25/2021</td> </tr> <tr> <td>30</td> <td>30</td> <td>30</td> <td>7/27/2021</td> </tr> <tr> <td>28.952380952380953</td> <td>30</td> <td>28</td> <td>7/28/2021</td> </tr> <tr> <td>29.04</td> <td>30</td> <td>28</td> <td>7/30/2021</td> </tr> <tr> <td>30</td> <td>30</td> <td>30</td> <td>8/1/2021</td> </tr> </tbody> </table>				avg suhu	min suhu	max suhu	Tanggal	29.043478260869566	30	28	7/25/2021	30	30	30	7/27/2021	28.952380952380953	30	28	7/28/2021	29.04	30	28	7/30/2021	30	30	30	8/1/2021
avg suhu	min suhu	max suhu	Tanggal																								
29.043478260869566	30	28	7/25/2021																								
30	30	30	7/27/2021																								
28.952380952380953	30	28	7/28/2021																								
29.04	30	28	7/30/2021																								
30	30	30	8/1/2021																								

Gambar 4.28: Min, max dan rata-rata sensor suhu

- Tampilan form tambah ikan

Tambah Jenis Ikan

Jenis Ikan

Jumlah Ikan

Tanggal ikan masuk(YY-MM-DD)

Simpan

Back

Gambar 4.29: Tampilan form tambah ikan

- Tampilan list info ikan

J-FARM

List Kolam Registrasi User Registrasi Kolam Logout

Info Ikan kolam2 edit

Jenis Ikan	Jumlah Ikan	Tanggal Masuk Ikan
nila	1000	2021-07-23
nila	1000	2021-07-23

Gambar 4.30: Tampilan list info ikan

- Tampilan chart sensor ph, DO dan suhu

J-FARM

List Kolam

Registrasi User

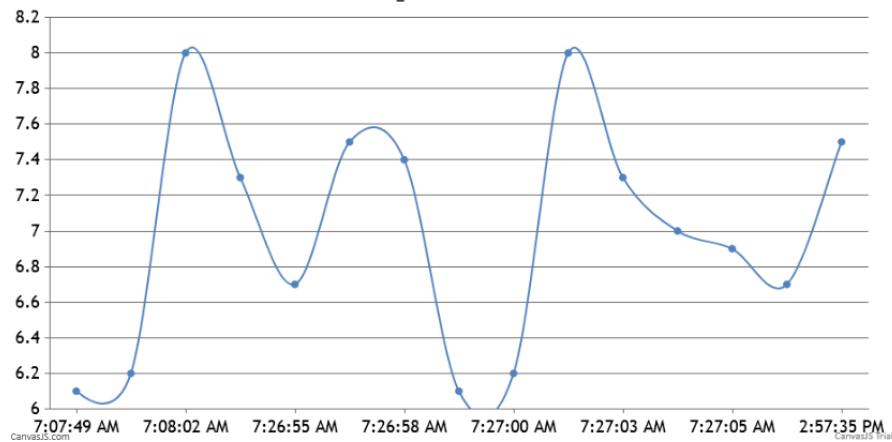
Registrasi Kolam

Kelola Ikan ▾

Chart ▾

Logout

Sensor ph kolam2 edit



Gambar 4.31: Chart ph

J-FARM

List Kolam

Registrasi User

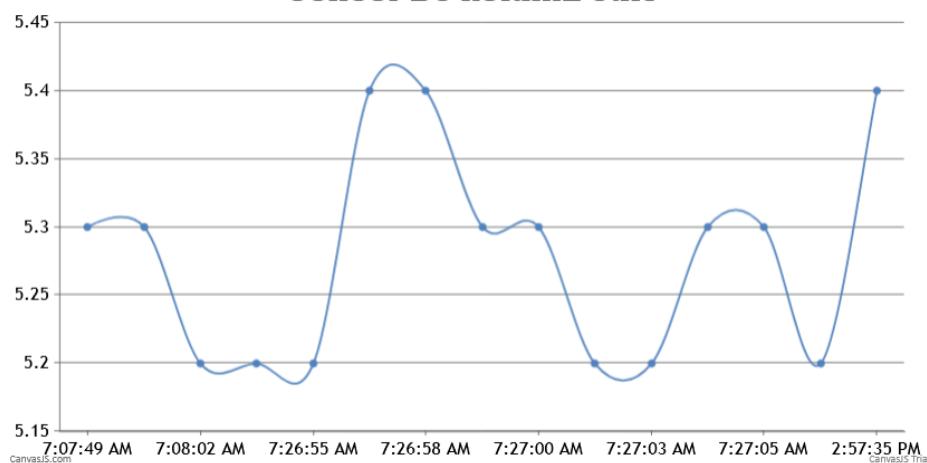
Registrasi Kolam

Kelola Ikan ▾

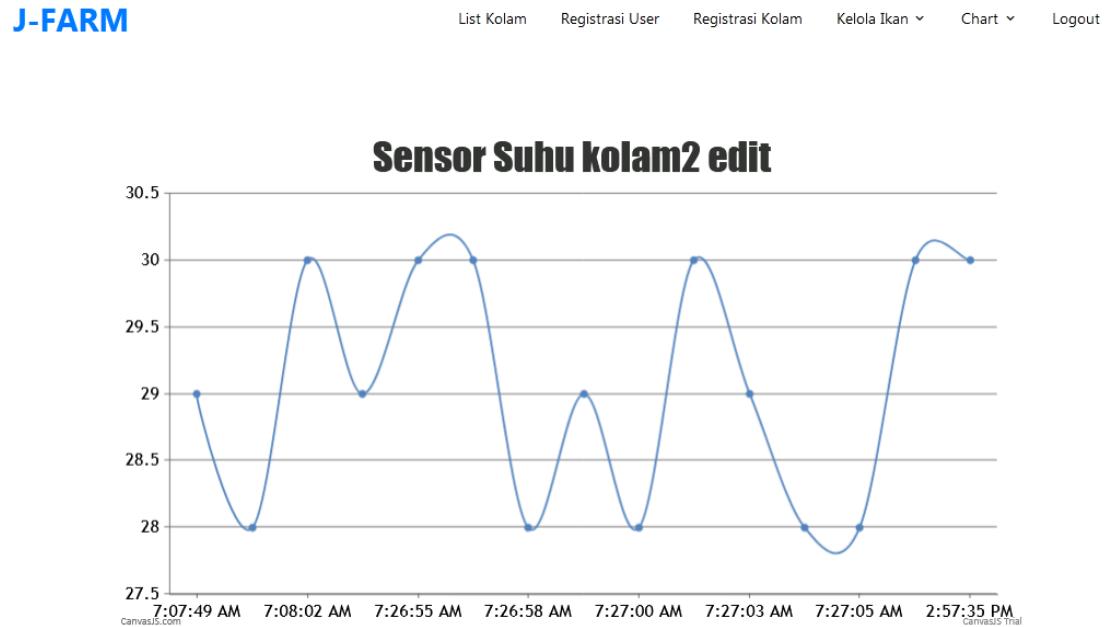
Chart ▾

Logout

Sensor DO kolam2 edit



Gambar 4.32: Chart DO



Gambar 4.33: Chart suhu

- *Source code*

Untuk source code *sprint-7* terdapat di link github, untuk link githubnya yaitu : <https://github.com/Fadhilah24/Web-Service/tree/sprint-7>.

4.1.3 Hasil akhir *sprint keseluruhan*

Di dalam section hasil akhir *sprint keseluruhan* akan menjelaskan keseluruhan hasil dari iterasi *sprint* yang telah dikerjakan yaitu dari *sprint-1* hingga *sprint-7*.

1. Product Backlog Tested

Tabel 4.7: Product Backlog Tested

NO	Story	Role	Sprint	Status
1	Kolam yang sudah ada di tempat penelitian dapat teregistrasi didalam sistem	Admin	1,2	Tested
2	Autentikasi untuk Admin	Admin	2	Tested
3	Dapat meregistrasi User	Admin	2	Tested
4	Autentikasi untuk User/Pengelola	User/Pengelola	2	Tested
5	Kolam yang sudah teregistrasi didalam sistem dapat dilihat semua dalam bentuk list.	User/Pengelola	3	Tested
6	Kolam yang sudah ada dalam list dapat diubah nama dan lokasi kolamnya. Kolam juga dapat dihapus	Admin	3	Tested
7	Membuat struktur data sensor yang akan di baca oleh web service	Sistem	4	Tested
8	Web-service dapat membaca data sensor yang telah dikirim.	Sistem	4,6	Tested
9	Dapat melihat history pembacaan sensor per 5 menit, per hari, per bulan ketika melihat detail.	User/Pengelola	7	Tested
10	Dapat melihat history pembacaan sensor, dari masing-masing jenis sensor per-beberapa hari kebelakang. Dalam bentuk min, max, dan rata-rata dari seluruh reading sensor pada hari dan bulan tersebut per-satu jenis sensor	User/Pengelola	7	Tested
11	Dapat menambahkan informasi jenis ikan yang sedang dibudidayakan di kolam tertentu dengan menambahkan detail terkait: Jenis ikan, Jumlah ikan, Tanggal ikan masuk, Berat gram saat ikan masuk. Dapat memfasilitasi lebih dari satu jenis ikan jika dibutuhkan.	User/Pengelola	7	Tested
12	Data indikator kolam dapat dilihat dalam bentuk Chart dan diupdate secara realtime dengan jangka waktu 5 menit per-update	User/Pengelola	7	Tested

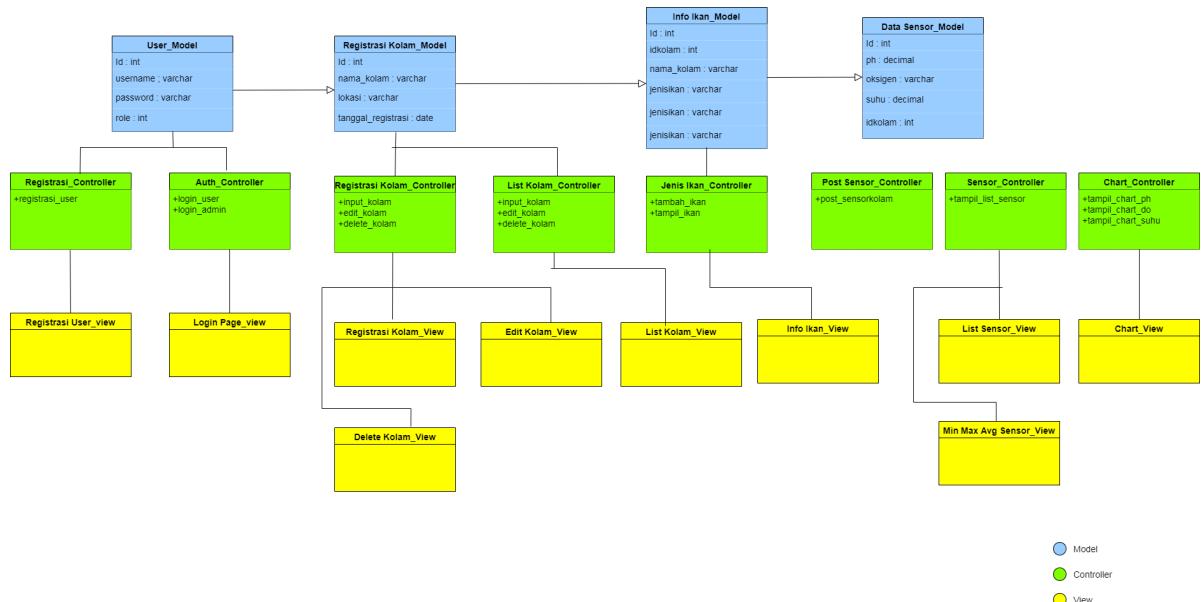
2. Desain database sistem

u8810863_restapijfarm users	u8810863_restapijfarm datasensor
id : bigint(20) unsigned	
username : varchar(255)	
password : varchar(255)	
role : int(11)	
token : varchar(255)	
created_at : timestamp	
updated_at : timestamp	

u8810863_restapijfarm registrasi_kolam	u8810863_restapijfarm infoikan
id : bigint(20) unsigned	
nama_kolam : varchar(255)	
lokasi : varchar(255)	
tanggal_registrasi : date	
token : varchar(255)	
created_at : timestamp	
updated_at : timestamp	

Gambar 4.34: Desain Database Sistem

3. MVC (*Model View Controller*) sistem



Gambar 4.35: Desain MVC Sistem

4. Routing Table

Tabel 4.8: Routing Table

Route Name	Route	Method	Description	Body	Parameter
LoginPage	/restapi/public/loginpage	GET	Login Page	-	-
LoginPost	/restapi/public/login	POST	Fungsi login dan response json login	username, password	-
RegisterUserPage	/restapi/public/registerus	GET	Register user page	-	-
RegisterUser	/restapi/public/register	POST	Fungsi Register User untuk meregistrasi user admin yang dilakukan oleh admin dan response json register user	username, password	-
RegistrasiKolamPage	/restapi/public/kolam/input	GET	Registrasi kolam page	-	-
RegistrasiKolam	/restapi/public/kolam/input	POST	Fungsi Registrasi kolam untuk menambahkan kolam dan response json registrasi kolam	id, nama_kolam, lokasi, tanggal_registrasi	-
ListKolam	/restapi/public/kolam/daftar	GET	List kolam page	-	-
ListKolamJson	/restapi/public/kolam/daftarjson	GET	Json list kolam	-	-
EditKolamPage	/restapi/public/kolam/editkolam/{id}	GET	Edit kolam page	-	id

Route Name	Route	Method	Description	Body	Parameter
EditKolamFunction	/restapi/public/kolam/updatekolam/{id}	PUT	Fungsi Edit Kolam	nama_kolam, lokasi	id
DeleteKolam	/restapi/public/kolam/hapus/{id}	DELETE	Fungsi Hapus Kolam	-	id
ListSensorPage	/restapi/public/sensor/sensorkolam/{id}	GET	List Sensor Page	-	id
ListSensorJson	/restapi/public/sensor/sensorjson	GET	Data sensor JSON	-	-
MinMaxAvgSensorsph	/restapi/public/sensor/sensorkolam/sensorsph/{id}	GET	Menampilkan halaman dengan min, max, avg sensor ph	-	id
MinMaxAvgSensorsordo	/restapi/public/sensor/sensorkolam/sensorsordo/{id}	GET	Menampilkan halaman dengan min, max, avg sensor DO	-	id
MinMaxAvgSensorsuhu	/restapi/public/sensor/sensorkolam/sensorsuhu/{id}	GET	Menampilkan halaman dengan min, max, avg sensor suhu	-	id
TambahJenisikanPage	/restapi/public/sensor/sensorkolam/kolam/tambahikanform/{id}	GET	Tambah jenis ikan page	-	id
TambahJenisikan	/restapi/public/kolam/tambahikan/{id}	POST	Fungsi tambah jenis ikan	idkolam, jenisikan, jumlahikan, tanggalikanmasuk	id

Route Name	Route	Method	Description	Body	Parameter
InfoJenisIkanList	/restapi/public/sensor/sensorkolam/kolam/infoikan/{id}	GET	Info jenis ikan list page	-	id
InfoJenisIkanJson	/restapi/public/sensor/sensorkolam/kolam/infoikanjson	GET	data JSON info jenis ikan	-	-
Chartph	/restapi/public/sensor/sensorkolam1/chart/{id}	GET	Chart sensor ph	-	id
Chartdo	/restapi/public/sensor/sensorkolam1/chartdo/{id}	GET	Chart sensor Do	-	id
Chartsuhu	/restapi/public/sensor/sensorkolam1/chartsuhu/{id}	GET	Chart sensor suhu	-	id
Postsensor	/restapi/public/sensor/updatesensor/{id}	POST	Post data sensor ke dalam web service	ph, oksigen, suhu	id

5. Hasil keseluruhan sistem

- Format data Json Sensor

```
{
    ph : value,
    oksigen : value,
    suhu : value
}
```

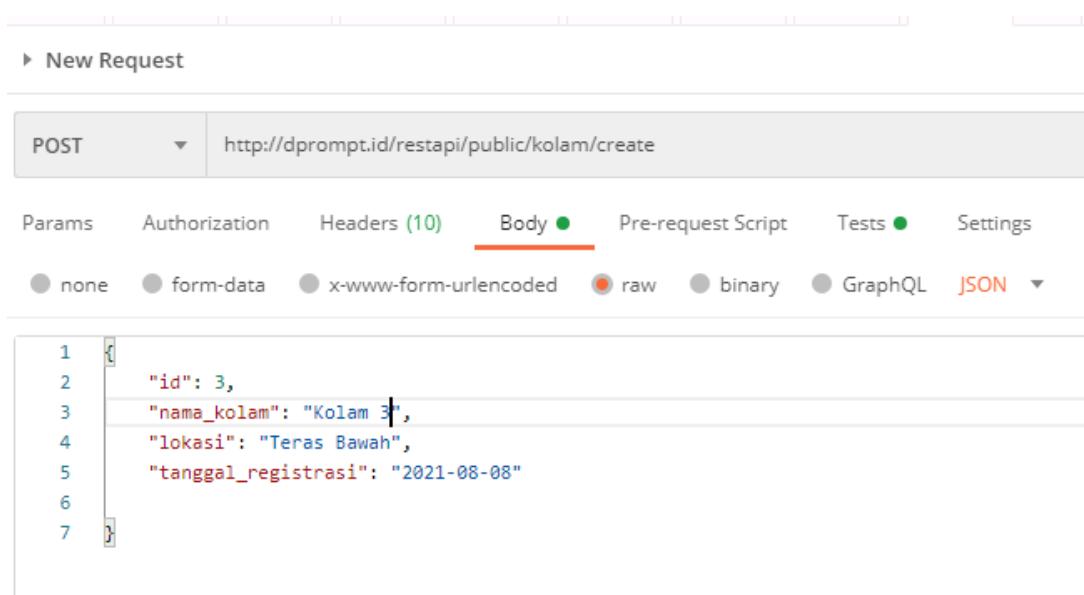
- Type data

```
{
    data : value
}
```

- id : int

- username : varchar
- password : varchar
- nama_kolam : varchar
- lokasi : varchar
- tanggal_registrasi : date
- idkolam : int
- jenisikan : varchar
- jumlah_ikan : varchar
- tanggalikanmasuk : date
- ph : decimal
- oksigen : decimal
- suhu : decimal

- Registrasi Kolam web service (*input*)



The screenshot shows the Postman interface with a 'New Request' dialog open. The method is set to 'POST' and the URL is 'http://dprompt.id/restapi/public/kolam/create'. The 'Body' tab is selected, showing a JSON payload:

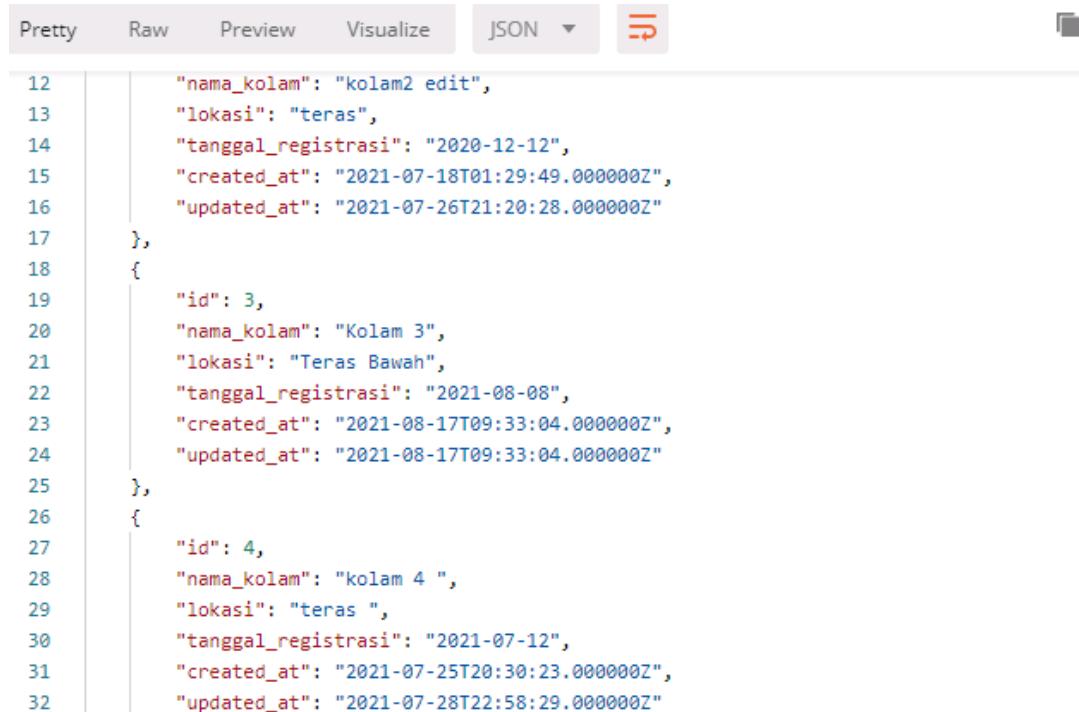
```

1 {
2   "id": 3,
3   "nama_kolam": "Kolam 3",
4   "lokasi": "Teras Bawah",
5   "tanggal_registrasi": "2021-08-08"
6 }
7
  
```

The JSON structure includes fields for id, nama_kolam, lokasi, and tanggal_registrasi.

Gambar 4.36: Registrasi Kolam web service (*input*)

- Registrasi Kolam web service (*output*)



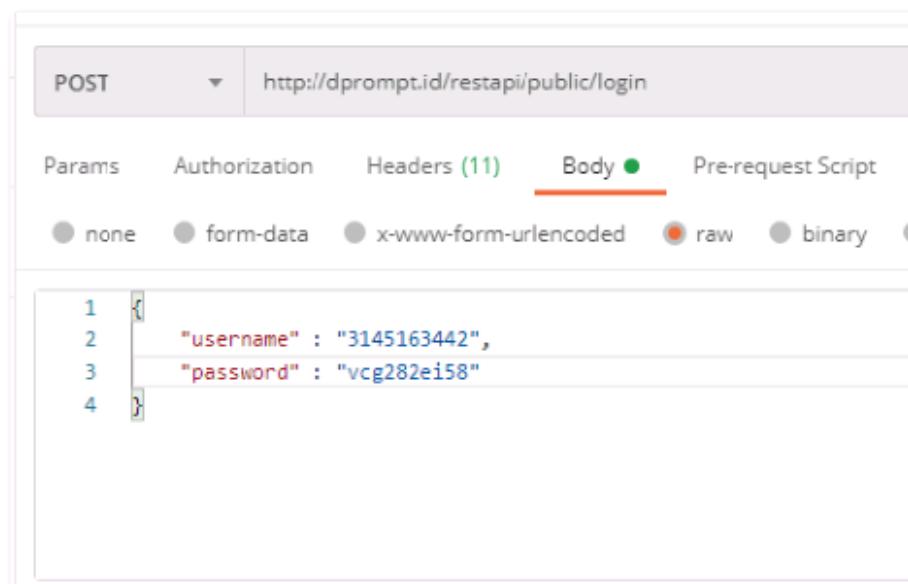
```

Pretty Raw Preview Visualize JSON ↗
12     "nama_kolam": "kolam2 edit",
13     "lokasi": "teras",
14     "tanggal_registrasi": "2020-12-12",
15     "created_at": "2021-07-18T01:29:49.000000Z",
16     "updated_at": "2021-07-26T21:20:28.000000Z"
17 },
18 {
19     "id": 3,
20     "nama_kolam": "Kolam 3",
21     "lokasi": "Teras Bawah",
22     "tanggal_registrasi": "2021-08-08",
23     "created_at": "2021-08-17T09:33:04.000000Z",
24     "updated_at": "2021-08-17T09:33:04.000000Z"
25 },
26 {
27     "id": 4,
28     "nama_kolam": "kolam 4 ",
29     "lokasi": "teras ",
30     "tanggal_registrasi": "2021-07-12",
31     "created_at": "2021-07-25T20:30:23.000000Z",
32     "updated_at": "2021-07-28T22:58:29.000000Z"

```

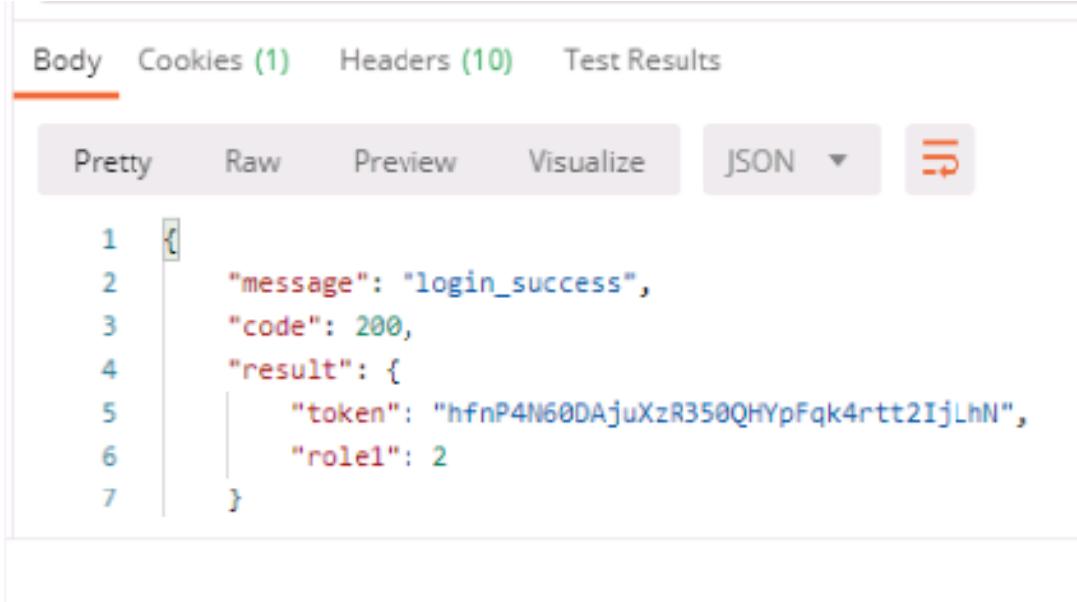
Gambar 4.37: Registrasi Kolam web service (*output*)

- Login Admin web service (*input*)



Gambar 4.38: Login Admin web service (*input*)

- Login Admin web service (*output*)



The screenshot shows the Postman interface with the 'Body' tab selected. The response body is displayed in JSON format:

```

1  {
2      "message": "login_success",
3      "code": 200,
4      "result": {
5          "token": "hfnP4N60DAjuXzR350QHYpFqk4rtt2IjLhN",
6          "role1": 2
7      }

```

Gambar 4.39: Login Admin web service (*output*)

- Login User web service (*input*)



The screenshot shows the Postman interface with the 'POST' method selected and the URL `http://dprompt.id/restapi/public/login`. The 'Body' tab is selected, showing the request body in raw JSON format:

```

1  {
2      "username" : "fadhilahph",
3      "password" : "vcg282ei58"
4  }

```

Gambar 4.40: Login User web service (*input*)

- Login User web service (*output*)

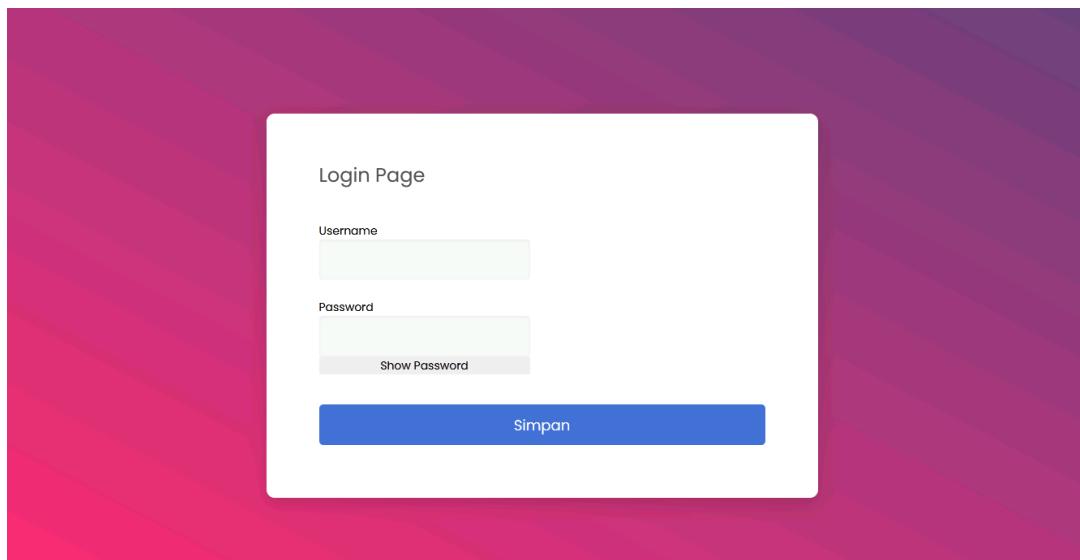
Body Cookies (1) Headers (10) Test Results

Pretty Raw Preview Visualize JSON ▾

```
1 {
2     "message": "login_success",
3     "code": 200,
4     "result": {
5         "token": "rGJNZRtv1xX1BvDJKZvTOZJUJQdYYqRigb0",
6         "role1": 1
7     }
}
```

Gambar 4.41: Login User web service (*output*)

- Tampilan Login



Gambar 4.42: Tampilan Login

- Registrasi Kolam

The screenshot shows a mobile application interface titled "Registrasi Kolam". The form contains four input fields: "ID Kolam" (Kolam ID), "Nama Kolam" (Lake Name), "Lokasi" (Location), and "Tanggal Registrasi(YY-MM-DD)" (Registration Date). Below the inputs are two blue buttons: "Simpan" (Save) and "Back".

Gambar 4.43: Tampilan registrasi kolam

- Registrasi user

The screenshot shows a mobile application interface titled "Registrasi User". The form contains three input fields: "Username", "Password", and "Confirm Password". Each password field includes a "Show Password" button. Below the inputs are two blue buttons: "Simpan" (Save) and "Back".

Gambar 4.44: Tampilan registrasi user

- List Kolam

ID Kolam	Nama Kolam	Lokasi	Tanggal Registrasi	Edit	Delete	Details
1	kolam1	teras	2020-12-12	Edit	Delete	Details
2	kolam2 edit	teras	2020-12-12	Edit	Delete	Details
4	kolam 4	teras	2021-07-12	Edit	Delete	Details
5	kolam5	teras1.	2021-07-12	Edit	Delete	Details
6	kolam6	teras	2021-07-23	Edit	Delete	Details
9	Kolam 9	Teras bawah	2021-07-07	Edit	Delete	Details

Gambar 4.45: Tampilan list kolam

- Update kolam web service (*input*)

PUT <http://dprompt.id/restapi/public/kolam/updatekolam/3>

Params Auth Headers (10) **Body** Pre-req. Tests Settings

raw JSON

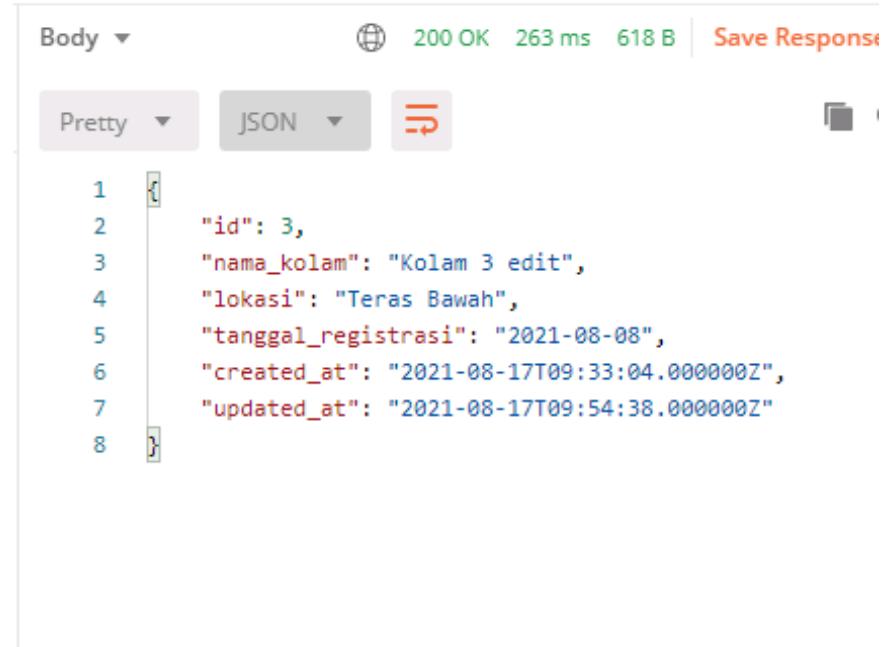
```

1 {
2   "nama_kolam": "Kolam 3 edit",
3   "lokasi": "Teras Bawah"
4 }
5

```

Gambar 4.46: Update kolam web service (*input*)

- Update kolam web service (*output*)

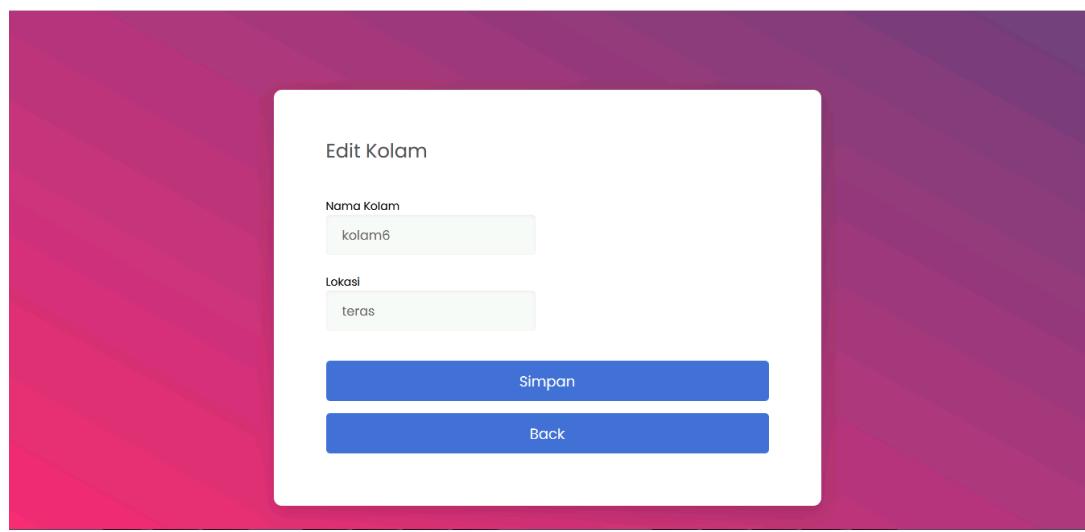


A screenshot of a POST request response in a REST client. The status bar shows "200 OK" and "263 ms". The response body is displayed in JSON format:

```
1  {
2      "id": 3,
3      "nama_kolam": "Kolam 3 edit",
4      "lokasi": "Teras Bawah",
5      "tanggal_registrasi": "2021-08-08",
6      "created_at": "2021-08-17T09:33:04.000000Z",
7      "updated_at": "2021-08-17T09:54:38.000000Z"
8 }
```

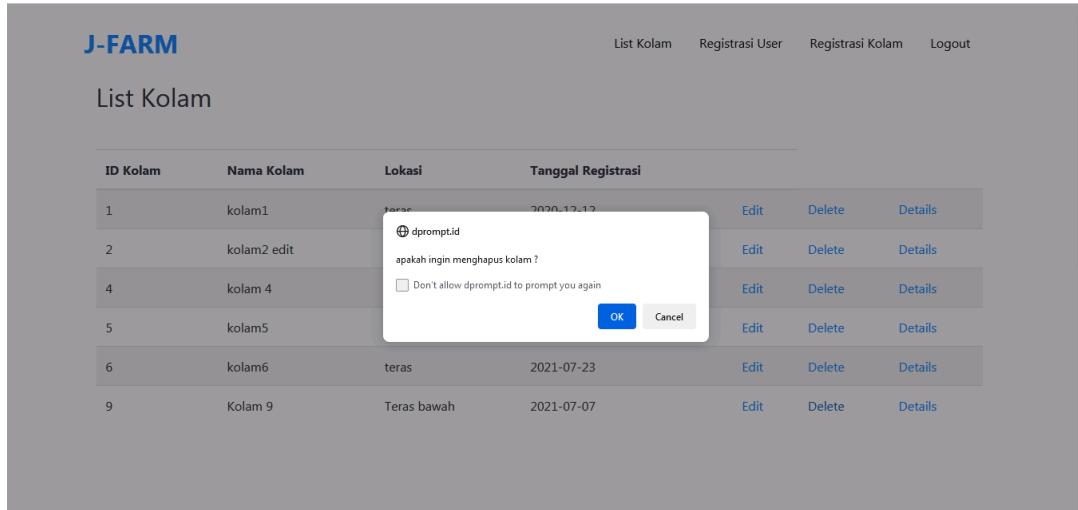
Gambar 4.47: Update kolam web service (*output*)

- *Edit Kolam*



Gambar 4.48: Tampilan edit kolam

- Delete Kolam



Gambar 4.49: Tampilan *delete* kolam

- POST data sensor web service (*input*).

The screenshot shows a POST request in Postman. The URL is `http://dprompt.id/restapi/public/sensor/updatesensor/2`. The "Body" tab is selected, and the "JSON" option is chosen. The JSON payload is a stringified object:

```

1 {
2   "ph": 7.5,
3   "oksigen": 5.4,
4   "suhu": 30
5
6 }
    
```

Gambar 4.50: POST data sensor web service (*input*)

- POST data sensor web service (*output*).

The screenshot shows a REST client interface with the following details:

- Body** dropdown is set to "Pretty".
- Response status: 200 OK.
- Time taken: 307 ms.
- Size: 583 B.
- Save Response** button is visible.
- Content type: JSON (indicated by the "JSON" dropdown).
- Copy icon and search icon are present.
- JSON data is displayed in a numbered, collapsible tree view:

```
1 {  
2   "idkolam": "2",  
3   "ph": 7.5,  
4   "oksigen": 5.4,  
5   "suhu": 30,  
6   "updated_at": "2021-08-17T10:02:15.000000Z",  
7   "created_at": "2021-08-17T10:02:15.000000Z",  
8   "id": 1433  
9 }
```

Gambar 4.51: POST data sensor web service (*output*)

Dari gambar di atas ditampilkan bahwa POST data sensor ke sistem berhasil dengan *status code* 200.

- Histori Pembacaan sensor di halaman detail

ph	Do	Suhu	Waktu
7.10	5.20	30.0	2021-07-25 15:57:45
7.10	5.20	30.0	2021-07-25 15:59:35
7.10	5.20	30.0	2021-07-25 16:09:15
7.50	5.40	30.0	2021-07-25 16:09:20
7.40	5.40	28.0	2021-07-25 16:09:25
7.30	5.30	29.0	2021-07-25 16:09:31
7.30	5.30	28.0	2021-07-25 16:09:36
7.20	5.20	30.0	2021-07-25 16:09:41

Gambar 4.52: List histori pembacaan sensor

- *Min, max dan rata-rata sensor ph, DO dan suhu per hari dalam sebulan*

avg ph	min ph	max ph	Tanggal
7.26086956521739	7.5	7.1	7/25/2021
7.5	7.5	7.5	7/27/2021
6.989523809523814	8	6.1	7/28/2021
7.00800000000001	8	6.1	7/30/2021
7.5	7.5	7.5	8/1/2021

Gambar 4.53: *Min, max dan rata-rata sensor ph*

J-FARM		List Kolam	Registrasi User	Registrasi Kolam	Kelola Ikan ▾	Chart ▾	Logout																							
Sensor ph																														
<input type="text" value="2021-07"/> <input type="button" value="OK"/> <input type="text" value="Pilih Tanggal"/> <input type="button" value="OK"/> <table border="1"> <thead> <tr> <th>avg DO</th> <th>min DO</th> <th>max DO</th> <th>Tanggal</th> </tr> </thead> <tbody> <tr> <td>5.286956521739131</td> <td>5.4</td> <td>5.2</td> <td>7/25/2021</td> </tr> <tr> <td>5.4</td> <td>5.4</td> <td>5.4</td> <td>7/27/2021</td> </tr> <tr> <td>5.29285714285714</td> <td>5.4</td> <td>5.2</td> <td>7/28/2021</td> </tr> <tr> <td>5.284</td> <td>5.4</td> <td>5.2</td> <td>7/30/2021</td> </tr> <tr> <td>5.4</td> <td>5.4</td> <td>5.4</td> <td>8/1/2021</td> </tr> </tbody> </table>							avg DO	min DO	max DO	Tanggal	5.286956521739131	5.4	5.2	7/25/2021	5.4	5.4	5.4	7/27/2021	5.29285714285714	5.4	5.2	7/28/2021	5.284	5.4	5.2	7/30/2021	5.4	5.4	5.4	8/1/2021
avg DO	min DO	max DO	Tanggal																											
5.286956521739131	5.4	5.2	7/25/2021																											
5.4	5.4	5.4	7/27/2021																											
5.29285714285714	5.4	5.2	7/28/2021																											
5.284	5.4	5.2	7/30/2021																											
5.4	5.4	5.4	8/1/2021																											
avg DO	min DO	max DO	Tanggal																											
5.286956521739131	5.4	5.2	7/25/2021																											
5.4	5.4	5.4	7/27/2021																											
5.29285714285714	5.4	5.2	7/28/2021																											
5.284	5.4	5.2	7/30/2021																											
5.4	5.4	5.4	8/1/2021																											

Gambar 4.54: Min, max dan rata-rata sensor DO

J-FARM		List Kolam	Registrasi User	Registrasi Kolam	Kelola Ikan ▾	Chart ▾	Logout																							
Sensor ph																														
<input type="text" value="2021-07"/> <input type="button" value="OK"/> <input type="text" value="Pilih Tanggal"/> <input type="button" value="OK"/> <table border="1"> <thead> <tr> <th>avg suhu</th> <th>min suhu</th> <th>max suhu</th> <th>Tanggal</th> </tr> </thead> <tbody> <tr> <td>29.043478260869566</td> <td>30</td> <td>28</td> <td>7/25/2021</td> </tr> <tr> <td>30</td> <td>30</td> <td>30</td> <td>7/27/2021</td> </tr> <tr> <td>28.952380952380953</td> <td>30</td> <td>28</td> <td>7/28/2021</td> </tr> <tr> <td>29.04</td> <td>30</td> <td>28</td> <td>7/30/2021</td> </tr> <tr> <td>30</td> <td>30</td> <td>30</td> <td>8/1/2021</td> </tr> </tbody> </table>							avg suhu	min suhu	max suhu	Tanggal	29.043478260869566	30	28	7/25/2021	30	30	30	7/27/2021	28.952380952380953	30	28	7/28/2021	29.04	30	28	7/30/2021	30	30	30	8/1/2021
avg suhu	min suhu	max suhu	Tanggal																											
29.043478260869566	30	28	7/25/2021																											
30	30	30	7/27/2021																											
28.952380952380953	30	28	7/28/2021																											
29.04	30	28	7/30/2021																											
30	30	30	8/1/2021																											
avg suhu	min suhu	max suhu	Tanggal																											
29.043478260869566	30	28	7/25/2021																											
30	30	30	7/27/2021																											
28.952380952380953	30	28	7/28/2021																											
29.04	30	28	7/30/2021																											
30	30	30	8/1/2021																											

Gambar 4.55: Min, max dan rata-rata sensor suhu

- Tampilan form tambah ikan

Tambah Jenis Ikan

Jenis Ikan

Jumlah Ikan

Tanggal ikan masuk(YY-MM-DD)

Simpan

Back

Gambar 4.56: Tampilan form tambah ikan

- Tampilan list info ikan

J-FARM

List Kolam Registrasi User Registrasi Kolam Logout

Info Ikan kolam2 edit

Jenis Ikan	Jumlah Ikan	Tanggal Masuk Ikan
nila	1000	2021-07-23
nila	1000	2021-07-23

Gambar 4.57: Tampilan list info ikan

- Tampilan chart sensor ph, DO dan suhu

J-FARM

List Kolam

Registrasi User

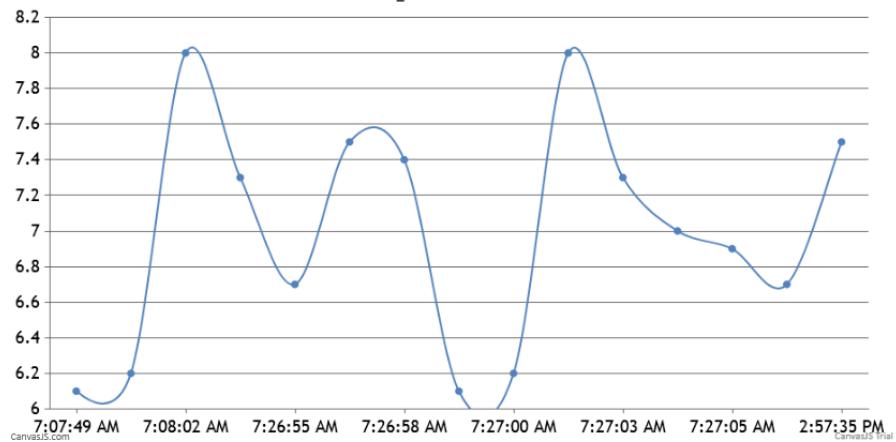
Registrasi Kolam

Kelola Ikan ▾

Chart ▾

Logout

Sensor ph kolam2 edit



Gambar 4.58: Chart ph

J-FARM

List Kolam

Registrasi User

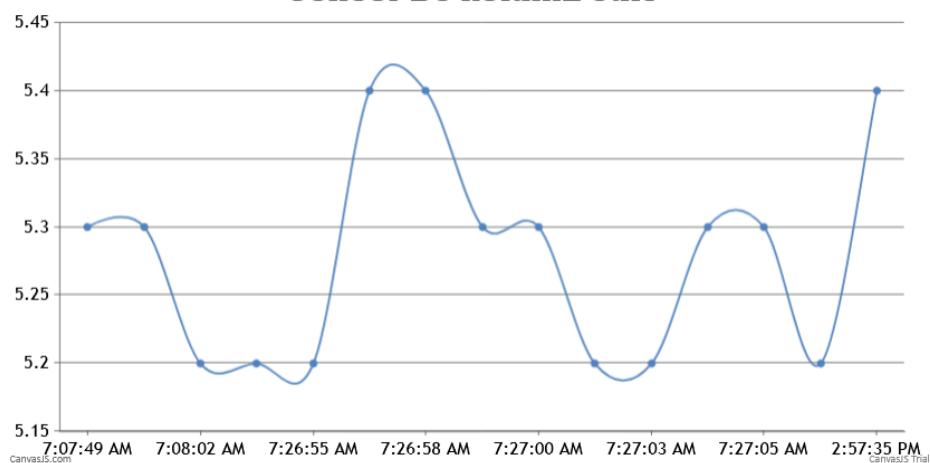
Registrasi Kolam

Kelola Ikan ▾

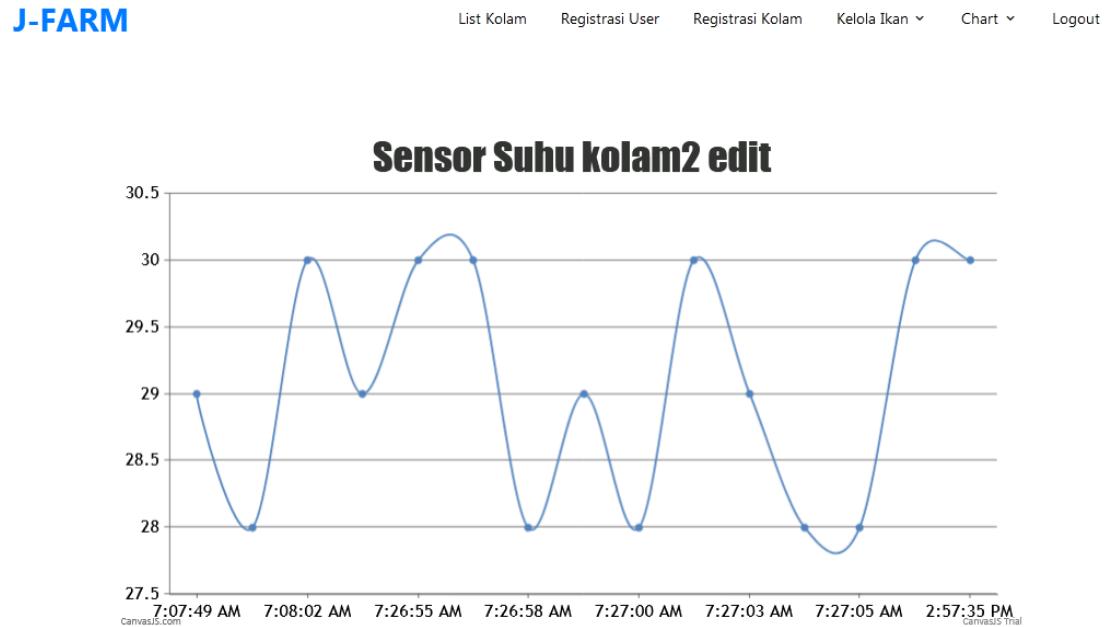
Chart ▾

Logout

Sensor DO kolam2 edit



Gambar 4.59: Chart DO



Gambar 4.60: Chart suhu

- *Source code* sistem keseluruhan

Untuk source code sistem keseluruhan terdapat di link github, untuk link githubnya yaitu : <https://github.com/Fadhilah24/Web-Service/tree/master>.

4.2 *Testing*

4.2.1 Hasil UAT (*User Acceptance Test*)

Berikut ini adalah hasil dari UAT (*User Acceptance Test*) dari sistem:

Tabel 4.9: Pengujian Login admin dan user

Kasus dan Hasil Uji Benar (Data Benar)			
Input	Output	Pengamatan	Kesimpulan User
Mengisi data login admin	Jika data login valid, maka akan masuk ke dalam halaman admin	Data Login Valid	Diterima
Mengisi data login user	Jika data login valid, maka akan masuk ke dalam halaman User	Data Login Valid	Diterima
Kasus dan Hasil Uji Salah(Data Salah)			
Input	Output	Pengamatan	Kesimpulan User
Data login admin atau user salah atau belum diisi	Dapat menampilkan pesan kesalahan	Menampilkan pesan kesalahan	Diterima

Tabel 4.10: Pengujian Fungsi dan Halaman Admin

Kasus dan Hasil Uji Benar (Data Benar)			
Skenario pengujian	Hasil yang diharapkan	Pengamatan	Kesimpulan User
Mengisi form registrasi user dengan lengkap	Jika data registrasi user sudah lengkap dan tepat, maka sistem akan menyimpan data tersebut dan redirect ke menu List Kolam	Data yang dimasukkan sudah lengkap dan tepat, sistem menyimpan data tersebut lalu redirect ke menu List Kolam	Diterima
Mengisi form registrasi kolam dengan lengkap	Jika data registrasi kolam sudah lengkap dan tepat, maka sistem akan menyimpan data tersebut dan redirect ke menu List Kolam	Data yang dimasukkan sudah lengkap dan tepat, sistem menyimpan data tersebut lalu redirect ke menu List Kolam	Diterima
Melihat data List Kolam di halaman List Kolam	Data yang ditampilkan merupakan data kolam yang telah diregistrasi	Menampilkan data kolam yang telah di registrasi	Diterima
Mengklik tombol edit di salah satu kolam lalu mengisi form edit kolam	Jika tombol edit diklik, maka akan redirect ke form edit kolam. lalu jika form edit kolam sudah lengkap dan tepat, maka akan redirect ke menu List Kolam	Halaman redirect ke form edit kolam dan data yang dimasukkan di dalam form edit kolam sudah lengkap dan tepat. Sistem menyimpan data tersebut lalu redirect ke menu List Kolam	Diterima
Mengklik tombol delete di salah satu kolam lalu lalu mengklik tombol OK	Jika tombol delete di salah satu kolam diklik lalu mengklik tombol OK, maka kolam tersebut terhapus dari List Kolam	Tombol delete di salah satu kolam diklik lalu mengklik tombol OK dan kolam tersebut terhapus dari List Kolam	Diterima
Mengisi form tambah ikan dengan lengkap	Jika data registrasi user sudah lengkap dan tepat, maka sistem akan menyimpan data tersebut dan redirect ke menu List Ikan pada kolom yang ditambahkan	Data yang dimasukkan sudah lengkap dan tepat, sistem menyimpan data tersebut lalu redirect ke menu List Ikan pada kolom yang ditambahkan	Diterima
Kasus dan Hasil Uji Salah (Data Salah)			
Skenario pengujian	Hasil yang diharapkan	Pengamatan	Kesimpulan User
Data form tidak diisi dengan lengkap.	Dapat menampilkan pesan kesalahan	Menampilkan pesan kesalahan	Diterima
Penginputan data id kolam pada form registrasi kolam yang sudah ada didalam sistem.	Dapat menampilkan pesan kesalahan	Menampilkan pesan kesalahan	Diterima
Penginputan data username pada form registrasi user yang sudah ada didalam sistem	Dapat menampilkan pesan kesalahan	Menampilkan pesan kesalahan	Diterima
Penginputan data confirm password pada form registrasi user yang tidak sesuai dengan data password	Dapat menampilkan pesan kesalahan	Menampilkan pesan kesalahan	Diterima

Tabel 4.11: Pengujian Fungsi dan Halaman User

Kasus dan Hasil Uji Benar (Data Benar)			
Skenario pengujian	Hasil yang diharapkan	Pengamatan	Kesimpulan User
Melihat data List Kolam di halaman List Kolam	Data yang ditampilkan merupakan data kolam yang telah diregistrasi	Menampilkan data kolam yang telah di registrasi	Diterima
Melihat histori pembacaan sensor per kolam dengan mengklik tombol detail di salah satu kolam dalam list lalu memilih bulan atau tanggal untuk menampilkan data sensor pada bulan atau tanggal yang dipilih	Jika tombol detail di salah satu kolam dalam list diklik, maka akan redirect ke halaman histori pembacaan sensor. Jika memilih bulan atau tanggal di dalam halaman histori pembacaan sensor, maka data sensor yang ditampilkan sesuai dengan bulan atau tanggal yang dipilih	Halaman redirect histori pembacaan sensor lalu data yang ditampilkan sesuai dengan bulan atau tanggal yang dipilih	Diterima
Melihat histori min, max, avarage per sensor per-kolam jika mengklik tombol ph, Do, atau Suhu di halaman histori pembacaan sensor lalu memilih bulan atau tanggal untuk menampilkan data perhitungan min, max, avarage sensor untuk menampilkan data perhitungan pada bulan atau tanggal yang dipilih	Jika tombol ph, Do, atau suhu di halaman histori pembacaan sensorlist diklik, maka akan redirect ke halaman histori penghitungan min, max, avarage sensor. Jika memilih bulan atau tanggal di halaman histori penghitungan min, max, avarage sensor, maka data penghitungan yang ditampilkan sesuai dengan bulan atau tanggal yang dipilih	Halaman redirect histori penghitungan min, max, avarage sensor lalu data penghitungan yang ditampilkan sesuai dengan bulan atau tanggal yang dipilih	Diterima
Melihat Info ikan pada kolam yang dipilih dengan mengklik detail lalu ke menu kelola ikan lalu klik menu info ikan.	Jika tombol detail diklik lalu ke menu kelola ikan lalu menu info ikan diklik, maka akan redirect ke halaman info ikan dan data yang ditampilkan merupakan data ikan yang telah ditambahkan pada kolam tersebut	Halaman redirect ke halaman info ikan dan menampilkan data ikan yang telah ditambahkan pada kolam tersebut	Diterima
Melihat chart realtime sensor dengan mengklik detail lalu ke menu chart lalu klik menu chart yang ingin dilihat	Jika tombol detail diklik lalu ke menu chart lalu menu chart yang ingin dilihat diklik, maka akan redirect ke halaman chart realtime yang dipilih. Chart bergerak dan update secara realtime sesuai waktu yang ditentukan	Halaman redirect ke halaman chart yang dipilih, data yang ditampilkan sesuai dengan chart yang dipilih dan data chart bergerak dan update secara realtime sesuai waktu yang ditentukan	Diterima

4.2.2 Kesimpulan Pengujian

UAT (*User Acceptance Test*) dilakukan dengan menggunakan metode *black box*. Metode *black box* merupakan pengujian sistem yang hanya mengecek fungsionalitas dari sistem saja dan bertujuan untuk menemukan kesalahan kesalahan atau kekurangan pada sistem yang diuji. Di dalam UAT (*User Acceptance Test*) ini semua fungsi yang diuji di dalam skenario UAT (*User Acceptance Test*) yang telah ditentukan seluruhnya telah berjalan dengan baik. Dengan hasil UAT (*User Acceptance Test*) yang telah dilakukan maka dapat disimpulkan bahwa sistem yang dirancang telah lulus uji. Untuk surat serah terima sistem dilampirkan pada **Lampiran E**.

4.3 Pembahasan

Sistem Web service dan Website sebagai *Storage Engine* dan *Monitoring Data Sensing* untuk Budidaya Ikan Air Tawar berfungsi sebagai penyimpanan data-data *sensing* dari sensor yang dikirimkan ke sistem serta memonitoringnya dalam bentuk table dan grafik *real-time*.

Perancangan sistem ini menggunakan *microframework* Lumen sebagai *back-end* web service-nya serta menggunakan Bootstrap, PHP, Javascript, Ajax dan JQuerry sebagai *back-end* dan *front-end* untuk websitenya. Perancangan sistem ini menggunakan metode penelitian Scrum di mana dalam metode ini perancangan dilakukan dengan iterasi per-*Sprint* yang urutan pekerjaannya terdapat pada *Product Backlog*.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil implementasi dan pengujian sistem ini, maka didapat kesimpulan sebagai berikut:

1. Berdasarkan hasil *User Acceptance Test* dengan pengujian menggunakan metode *black box testing* atau pengujian fungsional sistem didapatkan bahwa fungsi-fungsi yang terdapat pada sistem berjalan seluruhnya dengan baik dan lulus uji secara fungsional.
2. Sistem yang dirancang belum melakukan pengujian nilai kepuasan user atau metode pengujian skala likert.
3. Data yang digunakan dalam pengujian sistem ini menggunakan variabel *dummy* yang degenerate dan dipost melalui aplikasi POSTMAN.

5.2 Saran

Adapun beberapa saran untuk penelitian selanjutnya yaitu:

1. Untuk pengembang yang mengembangkan sistem ini selanjutnya, sebaiknya sistem ini dijalankan secara online supaya konsep IoT di dalam sistem ini berjalan seluruhnya.
2. Sebaiknya untuk data yang digunakan sistem ini selanjutnya menggunakan data asli dari sensor sehingga dapat dilihat apakah sistem dapat berjalan dengan data asli atau tidak.
3. Sebaiknya dalam pengujian *User Acceptance Test* selanjutnya dilakukan pengujian kepuasan user yaitu menggunakan metode pengujian likert.

DAFTAR PUSTAKA

- Adi Firdaus A. S., Slamet Widodo, A. S. (2019). Rancang bangun sistem informasi perpustakaan menggunakan web service pada jurusan teknik komputer polsri. *Jurnal Informatika*, 5.
- Agung A. N., A. N. (2017). Menumbuhkan service loyalty melalui kualitas pelayanan dan pengelolaan respon emosi konsumen pada perusahaan jasa. *Jurnal Manajemen*, 5.
- Akhtar H., H. (2018). Analisis regresi dengan variabel dummy di spss. <https://www.semestapsikometrika.com/2018/04/analisis-regresi-dengan-variabel-dummy.html>.
- Al Fatta H., H. (2007). *Analisis dan Perancangan Sistem Informasi untuk Keunggulan Bersaing Perusahaan dan Organisasi Modern*. Andi Offset, Yogyakarta.
- Al Qalit A. R., Fardian, A. R. (2017). Rancang bangun prototipe pemantauan kadar ph dan kontrol suhu serta pemberian pakan otomatis pada budidaya ikan lele sangkuriang berbasis iot. *Jurnal Online Teknik Elektro*, 2:8–15.
- Ardiansyah N., N. (2015). *Rancang Bangun pH Meter*.
- Chehal K., R. dan Singh, K. (2012). Efeciency and security of data with symmetric encryption algoritms. *International Jurnal of Advanced Research in Computer Science an SoftwareEngenering*, 2:472–475.
- Cimperman R., R. (2006). *UAT Defined: A Guide to Practical User Acceptance Testing*. Pearson Education.
- Documentation L., L. (2020). Lumen documentation. <https://lumen.laravel.com/docs/master>.

- Fauzi R., R. (2017). Pengenalan lumen framework, micro framework berbasis php. <https://www.codepolitan.com/pengenalan-lumen-framework-micro-framework-berbasis-php-59f19fe6ea010>.
- Feridi (2016). Mengenal soap web services. <https://www.codepolitan.com/mengenal-soap-web-services>.
- Feridi (2019). Mengenal restful web services. <https://www.codepolitan.com/mengenal-restful-web-services>.
- Guides S., S. (2020). Scrum guides. <https://scrumguides.org/scrum-guide.html>.
- infishta (2019). Jenis-jenis budidaya ikan air tawar. <https://infishta.com/blogs/jenis-jenis-budidaya-ikan-air-tawar>.
- Iskandaria (2012). *blackbox (blackbox testing), metode pengujian perangkat lunak yang berfokus pada sisi fungsionalitas*.
- ivy Wigmore (2015). Sensor data. <https://internetofthingsagenda.techtarget.com/definition/sensor-data>.
- Kabul Rizalul Haqim U. S. S. M., Ir. Agus Ganda Permana M.T., U. S. S. M. (2018). Perancangan web monitoring dan kontroling Aquaponicuntuk budidaya ikan lele berbasis internet of things. *e-Proceeding of Applied Science*, 2:2786.
- Kurniawan A., A. (2021). Pengertian suhu beserta alat ukurnya. <https://www.gurupendidikan.co.id/pengertian-suhu/>.
- Lewis W. E., W. E. (2009). *Software Testing and Continuous Quality Improvement*. CRC Press, Boca Raton.
- Meilinda Pramleonita R. A., Nita Yuliani, R. A. (2018). Parameter fisika dan kimia air kolam ikan nila hitam. *Jurnal Sains Natural Universitas Nusa Bangsa*, 8:24–34.

- Muhamman Femy Mulya N. R., N. R. (2017). Analisis dan perancangan sistem Mediation dengan protokol soap pada web service untuk mengintegrasikan antar sistem informasi yang berbeda Platform. *ULTIMA Infosys*, 8.
- Perry W. E., W. E. (2006). *Effective Methods for Software Testing 3rd*. Indianapolis, Indiana.
- Pramudya S., S. (2001). *Melindungi Lingkungan dengan Menerapkan ISO 14001*. Grasindo, Jakarta.
- Rian Bayu Pambudi R. A. S., Widhi Yahya, R. A. S. (2018). Implementasi node sensor untuk sistem pengamatan ph air pada budidaya ikan air tawar. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 2:2861–2868.
- Rumagit R. Y., R. Y. (2019). Pengenalan web services. <https://socs.binus.ac.id/2019/12/26/pengenalan-web-services/>.
- Shihab (2011). Metode white box dan black box testing. <http://rijjashihabuddin.metode-white-box-dan-black-box-testing.html>.
- Sidiq M., M. (2018). Pengertian Internet of Things (iot). <https://otomasi.sv.ugm.ac.id/2018/06/02/pengertian-internet-of-things-iot/>.
- Wellek S., S. (2013). *RESTful Web APIs*. O'Reilly Media, United States of America.

LAMPIRAN A

Transkrip Wawancara

Hari : Selasa

Tanggal: 8 Desember 2020

Lokasi : Perpustakaan Nasional, Jakarta Pusat

PW = Pewawancara

KL = Klien(Pemilik Farm)

PW : Kebutuhan apa saja yang dibutuhkan oleh sistem untuk penelitian ini?

KL : Untuk kebutuhan sistem, kita akan melihat nanti apa saja model bisnis yang akan dijalankan, karena sistem ini nantinya untuk mendukung budidaya ikan. sistem yang akan kita bangun nanti yaitu dari sisi IoT dan dari sisi Web Service-nya. dari sisi Web Service-nya fungsinya untuk monitoring data dari IoT-nya.

PW : Apakah sistem ini akan berjalan secara online atau lokal?

KL : Sistem ini akan berjalan secara lokal, jadi nanti pengelola kolam dapat melihat data-data kolam yang tersedia di komputer yang sudah terhubung di lokasi. Dimana nanti komputer tersebut terhubung dengan embedded device pada kolam melalui Web Service. Selanjutnya, untuk database nantinya akan disimpan didalam komputer yang berada di lokasi dan tidak perlu hosting

PW : Fungsi apasajakah nanti yang akan tersedia di dalam sistem

KL : Pertama, kalau kita budidaya kita harus tau ada berapa kolam, jadi nanti sistemnya harus punya fungsi untuk menghitung ada berapa kolam yang tersedia. Kedua, di setiap kolam ada sensing dimana nantinya sistem dapat menampilkan data sensing setiap kolam. Sistem juga harus dapat mendekripsi sensing yang terdapat di setiap kolam itu apa saja. Ketiga, untuk monitoring sistem harus menampilkan grafik, dan grafiknya itu realtime. Jadi data-data dari sensing setiap kolam akan di realisasikan kedalam grafik realtime. Tetapi data-datanya itu per kolam, bukan

seluruhnya.

PW : Untuk development aplikasinya dizinkan untuk memakai framework apa?

KL : Untuk framework memakai lumen karena simple dan ringan untuk Web Service.

LAMPIRAN B

Kode Controller Fungsi Registrasi Kolam

```
public function create(Request $request)
{
    $registrasikolam = new RegistrasiKolam;
    $registrasikolam->id = $request->id;
    $registrasikolam->nama_kolam = $request->
    nama_kolam;
    $registrasikolam->lokasi = $request->lokasi;
    $registrasikolam->tanggal_registrasi =
    $request->tanggal_registrasi;
    $registrasikolam->save();
    return response()->json($registrasikolam);
}
```

LAMPIRAN C

Kode Model Fungsi Registrasi Kolam

```
class RegistrasiKolam extends Model
{
    protected $table = 'registrasi_kolam';

    protected $fillable = [
        'nama_kolam', 'lokasi'
    ];

}
```

LAMPIRAN D

Beta Testing

NO	Story	Role	Yes	No
1	Kolam yang sudah ada di tempat penanaman dapat ter registrasi di dalam sistem	Admin	✓	
2	Autentikasi untuk Admin	Admin	✓	
3	Dapat mengejeksi User	Admin	✓	
4	Autentikasi untuk User/Pengelola	User/Pengelola	✓	
5	Kolam yang sudah terdaftar di dalam sistem dapat dilihat semua dalam bentuk list	User/Pengelola	✓	
6	Kolam yang sudah ada dalam list dapat diubah nama dan lokasi kolamnya. Kolam juga dapat dihapus	Admin	✓	
7	Membuat struktur data sensor yang akan dibaca oleh web service	Sistem	✓	
8	Web-service dapat membaca data sensor yang telah dikirim	Sistem	✓	
9	Dapat melihat history pembacaan sensor per 5 menit ketika melihat detail	User/Pengelola	✓	
10	Dapat melihat history pembacaan sensor dari masing-masing jenis sensor per beberapa hari kebelakang. Dalam bentuk min, max, dan rata-rata dari seluruh reading sensor pada hari tersebut per satuan jenis sensor	User/Pengelola	✓	
11	Dapat memperoleh informasi jenis ikan yang sedang dibudidayakan di kolam tertentu dengan menambahkan detail terkait jenis ikan, jumlah ikan, Tanggal ikan masuk, Berat gram saat ikan masuk. Dapat memfasilitasi lebih dari satu jenis ikan jika dibutuhkan	User/Pengelola	✓	
12	Data indikator kolam dapat dilihat dalam bentuk Chart dan diupdate secara real-time dengan jangka waktu 5 menit per update	User/Pengelola	✓	

Jasinga, 26 Juli 2021,



Guruh Dwi Purnama

LAMPIRAN E

User Acceptance Test

HASIL USER ACCEPTANCE TEST

Saya yang bertanda tangan di bawah ini, menyatakan pada hari Jumat, 30 Juli 2020 telah dilakukan pengujian *User Acceptance Test* dengan *requirement* pengujian sebagai berikut:

1. Fitur back-end sistem secara keseluruhan.
2. Tampilan back-end admin untuk mengecek histori pengecekan sensor yang dikirim dari sensor ke sistem.
3. Fitur monitoring sensor secara Real Time.

Dengan hasil Lulus Uji.



JFarm Teknologi
(.....)