Journal Computer and Technology
Vol. 2, No. 1, Juli 2024, hlm. 10-21
ISSN : 3048-1880 (Media Online)
DOI : https://doi.org/10.69916/comtechno.v2i1.138

IMPLEMENTASI *REST API* PADA SISTEM PENDATAAN KAPAL MENGGUNAKAN LARAVEL 9 DAN NUXTJS

Laurensius Sirwanti Saputra[1*] , Hendrik Fery Herdiyatmoko[2]

[1,2]Fakultas Sains dan Teknologi, Universitas Katolik Musi Charitas, Palembang, Indonesia
Email: [1]laurensiussirwantisaputra@gmail.com, [2]hendrik@ukmc.ac.id

SEJARAH ARTIKEL
Diterima: 17.07.2024
Direvisi: 26.07.2024
Diterbitkan: 30.07.2024

Hak Cipta © 2024 Penulis: Ini adalah artikel akses terbuka yang didistribusikan berdasarkan ketentuan Creative Commons Attribution 4.0 International License.

ABSTRAK

Perkembangan Teknologi, terutama Komputer, semakin pesat dan menjadi kebutuhan manusia sehari-hari. Di Distrik Navigasi Tipe A Kelas I Palembang, Teknologi Komputer berperan penting dalam meningkatkan kinerja yang efektif dan efisien. Distrik ini bertugas mengawasi telekomunikasi pelayaran, menyebarluaskan informasi cuaca pelayaran, serta mengawasi penyelenggaraan alur pelayaran dan telekomunikasi. Sistem pendataan keluar masuk kapal sebelumnya masih manual menggunakan *Microsoft Word* untuk pelaporan harian, bulanan, dan laporan cuaca. Untuk mengatasi masalah ini, dibuatlah sistem pendataan keluar masuk kapal berbasis *Website* menggunakan Teknologi *REST API Web Service*. *REST API* memfasilitasi pertukaran informasi berbasis *JSON* melalui jaringan *Internet*, memungkinkan *Integrasi* data antara pihak internal dan eksternal. Sistem ini dikembangkan dengan Framework Laravel 9 sebagai *Backend* dan NuxtJs sebagai *Frontend*. Hasil penelitian menunjukkan bahwa sistem pendataan keluar masuk kapal berbasis *Website* dengan Teknologi *REST API* berhasil diterapkan dan berjalan sesuai ketentuan serta format pendataan kapal di Distrik Navigasi Tipe A Kelas I Palembang.

Kata Kunci: *Website*, *Web Service*, Pendataan, *REST API*, *JSON*.

ABSTRACT

The development of technology, especially computers, is increasingly rapid and has become a daily human need. At the Palembang Type A Class I Navigation District, computer technology plays an important role in improving effective and efficient performance. This district is tasked with overseeing shipping telecommunications, disseminating shipping weather information, and overseeing the implementation of shipping lanes and telecommunications. The previous ship entry and exit data collection system was still manual using Microsoft Word for daily, monthly, and weather reports. To overcome this problem, a Website-based ship entry and exit data collection system was created using REST API Web Service technology. REST API facilitates JSON-based information exchange over the Internet network, allowing data integration between internal and external parties. This system was developed with Laravel 9 Framework as the Backend and NuxtJs as the Frontend. The results showed that the Website-based ship entry and exit data collection system with REST API technology was successfully implemented and ran according to the provisions and format of ship data collection at the Palembang Type A Class I Navigation District.

Keywords: *Website, Web Service, Data Collection, REST API, JSON.*

1. PENDAHULUAN

Web Service adalah sistem yang memungkinkan pertukaran informasi menggunakan *eXtensible Markup Language* (XML) atau *JavaScript Object Notation* (*JSON*) melalui jaringan *Internet*, untuk mendukung interaksi antara aplikasi dan perangkat [1]. Secara teknis, *Web Service* menyediakan mekanisme untuk interaksi antara sistem yang mendukung interoperabilitas, baik dalam bentuk agregasi (pengumpulan) maupun sindikasi (penyatuan) [2][3].

Perkembangan Teknologi, terutama Komputer, semakin pesat dan menjadi kebutuhan manusia sehari-hari. Di Distrik Navigasi Tipe A Kelas I Palembang, Teknologi Komputer berperan penting dalam meningkatkan kinerja yang efektif dan efisien [4]. Distrik Navigasi merupakan unit pelaksanan teknis di lingkungan Kementerian Perhubungan yang berada di bawah dan bertanggung jawab kepada Direktur Jenderal Perhubungan Laut [5]. Keberadaan Distrik Navigasi Tipe A Kelas I Palembang merupakan unit yang menyelenggarakan pelayanan Keselamatan Pelayaran dan Perlindungan Lingkungan Maritim pada Wilayah kerja yang mencakup perairan pada Provinsi Sumatera Selatan, Jambi dan Bangka Belitung [6]. Distrik Navigasi Tipe A Kelas I Palembang menghadapi masalah pendataan kapal di bagian VTS, di mana pelaporan harian dan bulanan dilakukan secara manual dengan *Microsoft Word* dan informasi kapal diperoleh melalui radio telepon langsung dari nahkoda kapal.

Berdasarkan hasil survei dengan pihak operator VTS (*Vessel Traffic Service*) yang mengungkapkan permasalahan dalam sistem pendataan kapal, sehingga peneliti mengusulkan dan membuat sistem pendataan kapal berbasis *Web Service* dengan penerapan *REST API*. *Web Service* merupakan sebuah perangkat lunak yang digunakan sebagai penghubung dari mesin ke mesin yang memungkinkan dapat saling berkomunikasi tanpa terpengaruh dengan perbedaan Platform dan tidak secara langsung terhubung ke *Database* yang dimiliki[7].

Penelitian ini menggunakan Teknologi *REST API* dengan Framework Laravel sebagai *Backend* untuk membuat sistem pendataan keluar masuk kapal di Distrik Navigasi Tipe A Kelas I Palembang, yang memungkinkan *Integrasi* data dengan *respons JSON* dan akses mudah oleh nahkoda kapal melalui *Website* dan *Web Mobile* untuk pihak *internal* dan *eksternal*. Dalam sistem REST, terdapat empat metode permintaan HTTP yang bisa digunakan: *GET*, *POST*, *PUT*, dan *DELETE*. Keempat metode ini menjalankan proses CRUD pada basis data, yaitu *CREATE*, *READ*, *UPDATE*, dan *DELETE*, dengan menggunakan format pertukaran data JSON [8][9][10].

Laravel adalah Framework PHP *open source* yang diciptakan oleh Taylor Otwell, menawarkan fitur-fitur kaya seperti *bundle*, migrasi, dan artisan CLI yang meningkatkan kecepatan pengembangan *Website*, menggunakan konsep HMVC, memiliki keamanan data dengan fitur blade, dan mengimplementasikan *routing* untuk menjembatani permintaan *user* dan *controller* [11].

API (*Application Programming Interface*) adalah bahasa dan format pesan yang digunakan aplikasi untuk berkomunikasi dengan sistem operasi atau program kontrol lainnya, sering disebut *Web Service* karena menyediakan layanan bagi klien, dan fungsinya meliputi *JSON*, XML, JavaScript, SOAP, dan REST [9]. REST adalah paradigma arsitektur layanan web yang menggunakan protokol HTTP, mengikuti prinsip CRUD, independen dari bahasa atau Platform, dan meskipun tidak menyediakan keamanan bawaan, memungkinkan implementasi fitur keamanan di atas HTTP, menjadikannya pilihan kuat untuk pengembangan layanan web [12].

JSON (*JavaScript Object Notation*) adalah format data yang ringan dan mudah dibaca, mendukung dua struktur dasar (*Arrays* dan *Objects*), dan digunakan secara luas untuk pertukaran data di berbagai aplikasi perangkat lunak, terutama dalam pengembangan *REST API*, karena sintaksnya identik dengan objek JavaScript dan didukung secara universal oleh banyak bahasa pemrograman [12][8].

JavaScript adalah bahasa pemrograman yang menambahkan *fungsionalitas* ke HTML dengan memungkinkan eksekusi perintah di sisi peramban, tergantung pada peramban yang memuat halaman web berisi skrip JavaScript dalam dokumen HTML [13].

Nuxt.Js merupakan *web application* Framework berbasis VueJs, NodeJs, Webpack dan *BabelJs* yang bersifat *open source* dan di rancang untuk pembuatan sebuah aplikasi atau *Website Universal* yang bisa di render dari sisi *server* atau menjadi sebuah *Website* statis [14].

Tahap akhir pembangunan *RESTful Web Service* adalah menguji *REST API* menggunakan Postman untuk memastikan *Response* dari *Request* yang dikirimkan, melibatkan pengiriman *Request* dalam bentuk *endpoint, HTTP method, headers,* dan *body*, serta menerima *Response* dalam format *JSON* dan *status code* [15].

2. METODE PENELITIAN

2.1. *Software Specification*

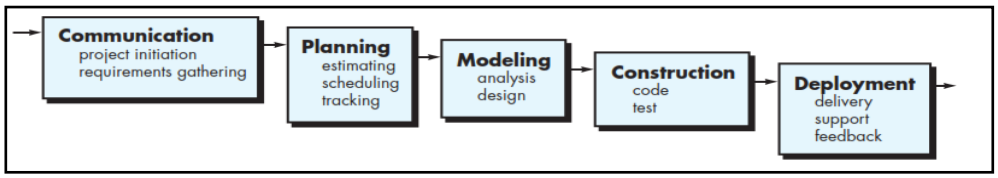
Software yang digunakan dalam menjalankan aplikasi ini adalah sebagai Berikut:

1. Laravel versi 9 PHP Framework
2. NuxtJs versi @4.0.0 Framework
3. XAMPP *Web Server*
4. Aplikasi Postman untuk pengujian *endpoint Response API* berupa *JSON* dari *REST Server*.

2.2. Tahap Desain

a) Metode Pengembangan sistem

Metode penelitian yang digunakan oleh peneliti adalah metode *Waterfall*. Menurut Presman *Model Waterfall* adalah *Model* klasik yang bersifat *sistematis* berurutan dalam membangun sebuah software [16]. Pada Gambar 1 dibawah ini adalah model penelitian yang digunakan.



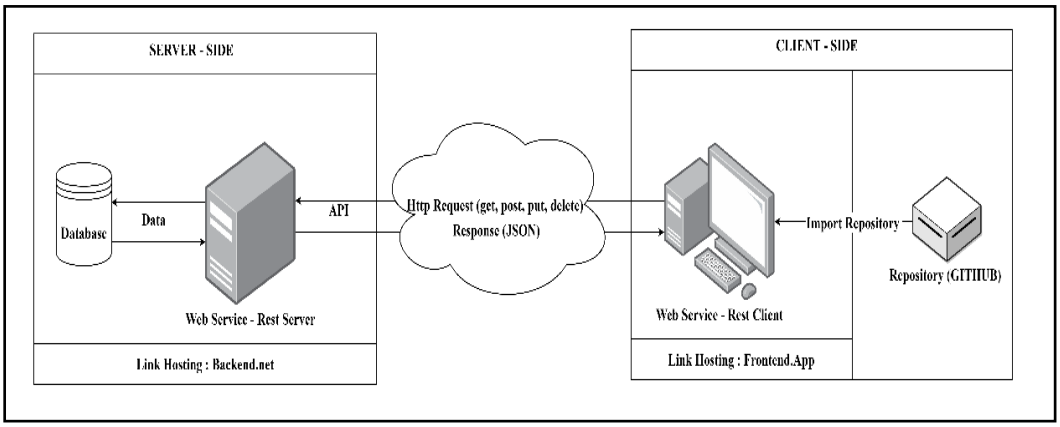
Gambar 1. Waterfall Model

Berikut rincian kegiatan yang dilakukan berdasarkan metode pengembangan sistem yang digunakan. Dapat diliat pada table 1 dibawah ini.

Tabel 1. Tahap Pengembangan Sistem

No.	Tahap	Kegiatan	Peralatan
1.	*Communication*	1. Pengamatan lapangan 2. Wawancaran dengan pihak Distrik Navigasi Tipe A Kelas I Palembang	1. Wawancara 2. Observasi 3. Buku-buku 4. Jurnal
2.	*Planning*	1. Merencanakan jadwal kegiatan 2. Menganalisa kebutuhan *hardware* dan *software*	1. Jadwal kegiatan
3.	*Modeling*	1. Merancang UML (*Unified Modeling Language*) 2. Merancang *Use Case Diagram* 3. Merancang *interface* 4. Merancang *database*	1. Figma 2. Draw.io
4.	*Construction*	1. Melakukan *coding* 2. Melakukan pengujian 3. Memperbaiki *error*	1. *POST*man 2. Visual Studio Code
5.	*Deployment*	1. Melakukan pengujian 2. Menyerahkan aplikasi kepada pihak Distrik Navigasi Tipe A Kelas I Palembang	1. Aplikasi Pendataan Kapal

b) Arsitektur Sistem



Gambar 2. Arsitektur Sistem Pendataan Kapal

Rest Client (*Admin*, Operator, Nahkoda) akan melakukan *request* dengan mengirimkan *HTTP request* berupa (*Get, Post, Put, Delete*) ke *REST Server* melalui *RESTful API* yang terdapat pada *controller* dan *model*. Sebelum *request* diterima, kemudian *controller* menerima *request* dari *client*, dan mengarahkan ke *service* yang sesuai untuk memproses *request* tersebut. Jika dibutuhkan, *Controller* akan memanggil *model* untuk kemudian membaca, mengisi, mengubah, dan menghapus data pada *database server*. *Database* handler pada *model* kemudian menerima data atau kondisi (berhasil/tidak) dari *database server* dan mengembalikan nilai data atau kondisi tersebut kepada *Controller*. *Controller* kemudian menyusun data *JSON* dan mengirim kembali *response* kepada *client*.

c) *Use Case Diagram*

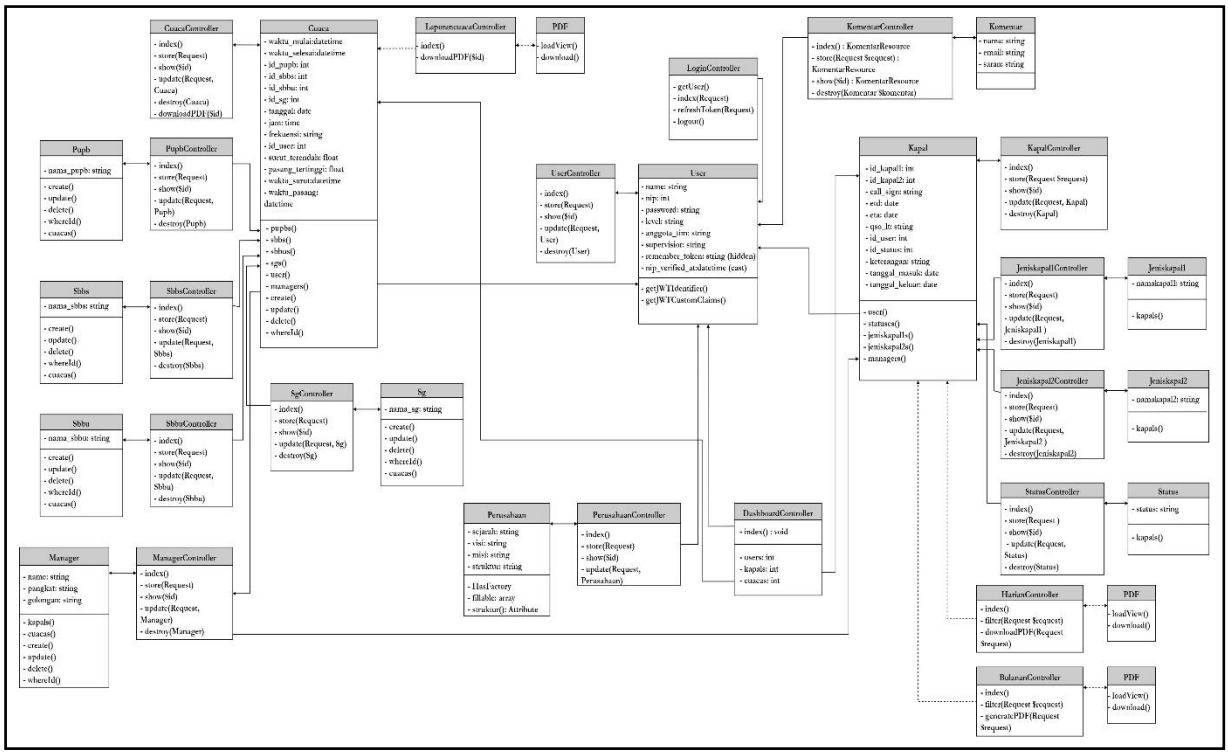
Use Case diagram merupakan pe*model*an untuk kelakuan sistem informasi yang akan dibuat. *Use Case diagram* digunakan untuk men*GET*ahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi - fungsi itu[17]. Berikut ini Tabel 2 *Use Case diagram*.

Tabel 2. Use Cas Diagram Input Data Kapal

Nama *Use Case*	*Input* Data Kapal
Aktor	Operator
Kondisi Awal	Operator sudah melakukan *login*
Aliran Kejadian	1. Menampilkan halaman *Input* data kapal. 2. Operator mengklik tombol tambah untuk menambah data kapal. 3. Operator mencari data kapal dengan mengisi pada search bar dan menekan tombol search. 4. Operator mengklik tombol edit untuk mengubah data kapal. Operator mengklik tombol *DELETE* untuk menghapus data kapal.
Kondisi Akhir	Operator masuk kedalam halaman *Input* data kapal dan dapat mengakses layanan untuk menampilkan data, mencari data, menambah data, mengedit data, dan menghapus data kapal.

d) *Class Diagram*

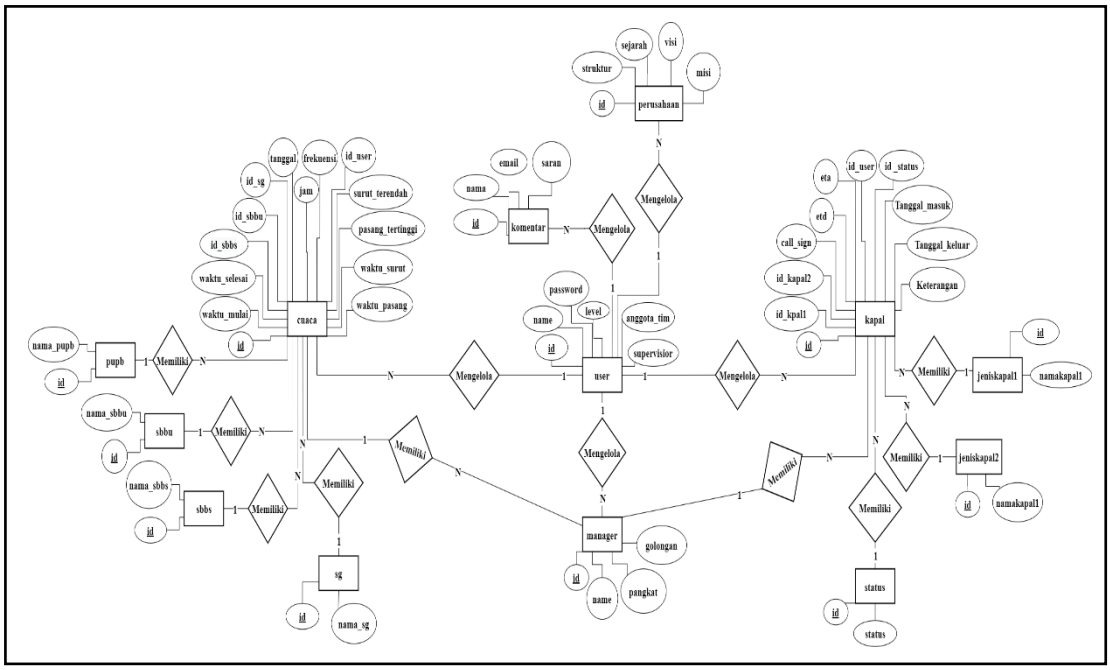
Class diagram menggambarkan struktur sistem dari segi pendefinisan *class-class* yang akan dibuat untuk membangun sistem [17].



Gambar 3. *Class Diagram* Sistem Pendataan Kapal

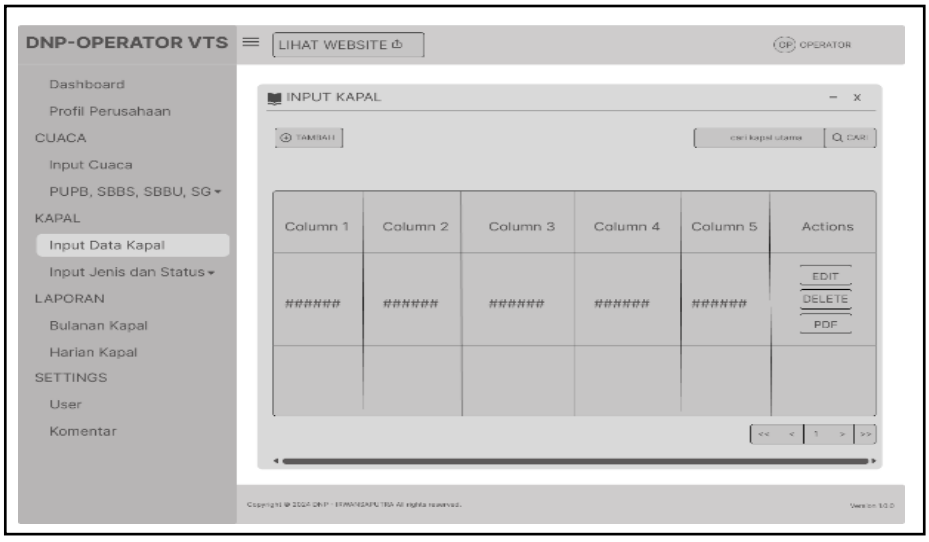
e) *Entity Relationship Diagram* (ERD)

Pemodelan awal basis data yang paling banyak digunakan adalah menggunakan *Entity Relationship Diagram* (ERD). ERD dikembangkan berdasarkan teori himpunan dalam bidang matematika. ERD digunakan untuk pemodelan basis data relasional [17]. Berikut ERD yang digunakan dapat dilihat pada gambar 4.



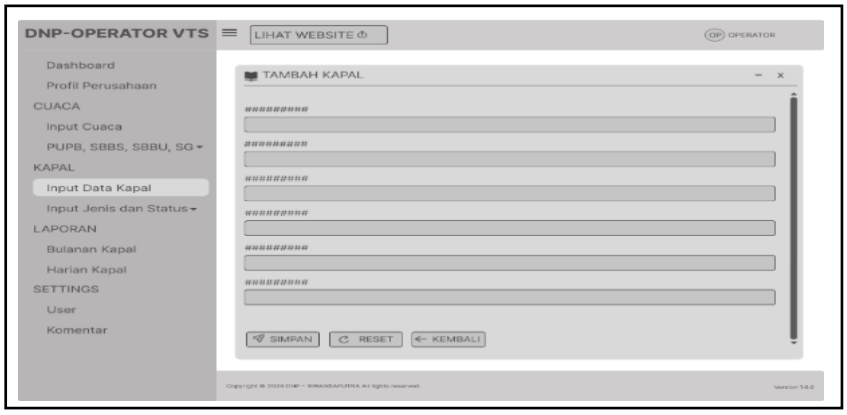
Gambar 4. ERD

f) Perancangan Interface Sistem



Gambar 5. Layout Menu View Data Kapal

Pada Gambar 5 diatas adalah *layout* menu *view* data kapal untuk *role* operator dimana akan di tampilkan data kapal serta beberapa fitur yang dapat di akses oleh operator seperti tambah data, *edit* data, dan *delete* data kapal.



Gambar 6. Layout Menu Tambah Data Kapal

Pada Gambar 6 diatas adalah *layout* menu untuk fitur tambah data kapal, dimana operator akan menginput data kapal lalu tekan tombol simpan selanjutnya data akan di validasi apakah data yang di *input* bernilai *true*/benar atau *false*/salah. Jika bernilai *true*/benar akan muncul notifikasi sukses dan data berhasil simpan di *database*, jika bernilai *false*/salah maka akan muncul notifikasi salah atau pesan *erorr*, untuk tombol *reset* berfungsi untuk menghapus kembali data yang sudah di isi dan untuk tombol kembali berfungsi untuk kembali kehalaman sebelumnya.

3. HASIL DAN PEMBAHASAN

Dalam penelitian ini digunakan empat *Endpoint API* [8] yang merepresentasikan CRUD data kapal. *Endpoint* tersebut adalah:

a) *GET*: http://127.0.0.1:8000/api/operator/kapals, akan mengembalikan semua data kapal dengan menerima *GET Request*.

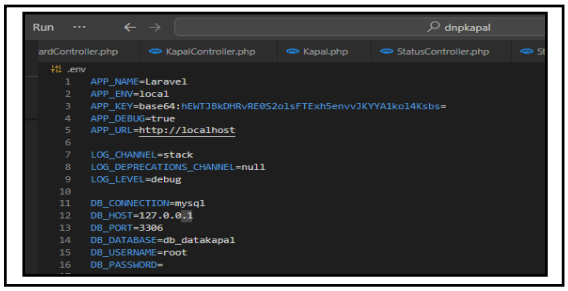
b) *POST*: http://127.0.0.1:8000/api/operator/kapals/{id}, akan membuat record kapal baru dan akan menerima *POST Request*.

c) *PUT*: http://127.0.0.1:8000/api/operator/kapals/{id}, akan mengubah data kapal yang ada dengan mengacu pada id kapal dan akan menerima *PUT Request*.

d) *DELETE:* http://127.0.0.1:8000/api/operator/kapals/{id}, akan menghapus data kapal dengan merujuk pada id kapal dan menerima *DELETE Request*

3.1. Konfigurasi *Database*

Tahap konfigurasi *database* dilakukan sebelum pembuatan *REST* Server, langkah ini bertujuan untuk mempersiapkan *database* yang akan digunakan dalam proyek. Gambar 9 menunjukkan *konfigurasi database* yang terdapat di file .env.



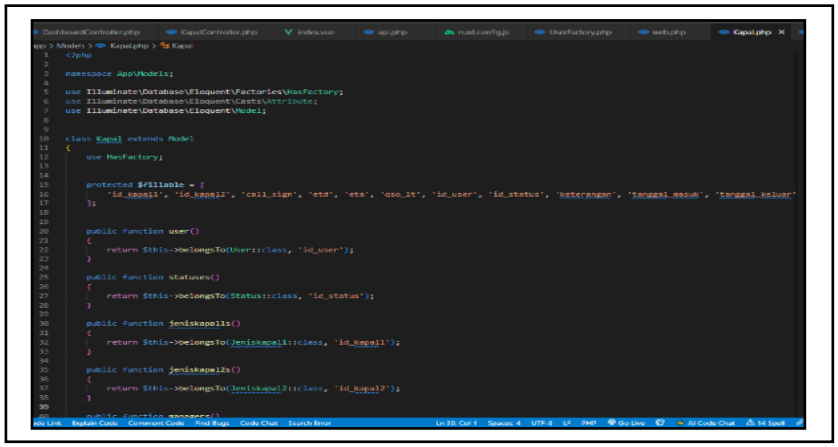
Gambar 7. Konfigurasi Database Laravel

3.2. Membuat *Model* dan *Migration* Kapal

Membuat *model* dan juga *migration* untuk kapal. Silahkan jalankan perintah berikut ini di dalam terminal/CMD : php artisan make:model Kapal -m.

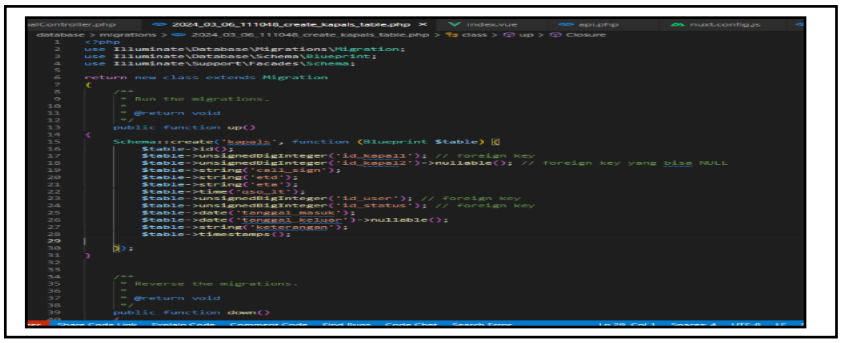
Jika perintah di atas berhasil, maka kita akan mendapatkan 2 file baru seperti pada Gambar 10 dan Gambar 11 dibawah ini:

1. app/Models/Kapal.php
2. *database*/migrations/2024_06_01_093404_create_kapals_table.php



```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Casts\Attribute;
use Illuminate\Database\Eloquent\Model;

class Kapal extends Model
{
    use HasFactory;

    protected $fillable = [
        'id_kapal1', 'id_kapal2', 'call_sign', 'etd', 'eta', 'oso_it', 'id_user', 'id_status', 'keterangan', 'tanggal_masuk', 'tanggal_keluar'
    ];

    public function user()
    {
        return $this->belongsTo(User::class, 'id_user');
    }

    public function statuses()
    {
        return $this->belongsTo(Status::class, 'id_status');
    }

    public function jeniskapal1s()
    {
        return $this->belongsTo(Jeniskapal1::class, 'id_kapal1');
    }

    public function jeniskapal2s()
    {
        return $this->belongsTo(Jeniskapal2::class, 'id_kapal2');
    }
```

Gambar 8. Model Kapal



```php
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('kapals', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('id_kapal1'); // foreign key
            $table->unsignedBigInteger('id_kapal2')->nullable(); // foreign key yang bisa NULL
            $table->string('call_sign');
            $table->string('etd');
            $table->string('eta');
            $table->time('oso_it');
            $table->unsignedBigInteger('id_user'); // foreign key
            $table->unsignedBigInteger('id_status'); // foreign key
            $table->date('tanggal_masuk');
            $table->date('tanggal_keluar')->nullable();
            $table->string('keterangan');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
```

Gambar 9. Migration Kapal

3.3. Membuat *Resource* dan *Controller* Kapal

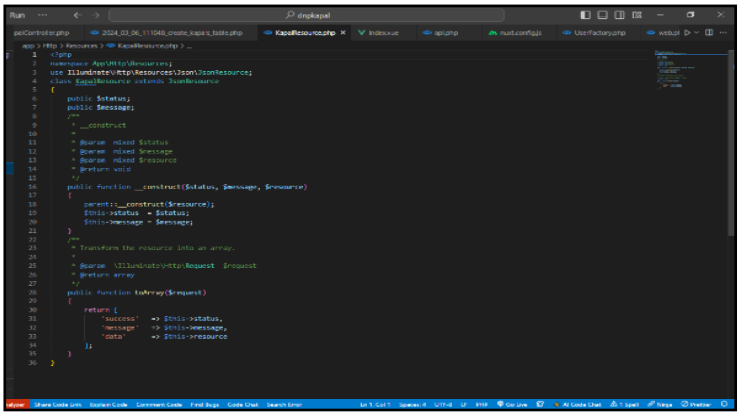
Membuat *resource* dan juga *controller* untuk Kapal. Silahkan jalankan perintah berikut ini di dalam terminal/CMD :

php artisan make:resource KapalResource

php artisan make:controller Api/Operator/KapalController

Jika perintah di atas berhasil, maka kita akan mendapatkan 2 file baru, yaitu :

- app/Http/Resources/KapalResource.php
- app/Http/Controllers/Api/Operator/KapalController.php



```php
<?php
namespace App\Http\Resources;
use Illuminate\Http\Resources\Json\JsonResource;
class KapalResource extends JsonResource
{
    public $status;
    public $message;
    /**
     * __construct
     *
     * @param  mixed $status
     * @param  mixed $message
     * @param  mixed $resource
     * @return void
     */
    public function __construct($status, $message, $resource)
    {
        parent::__construct($resource);
        $this->status  = $status;
        $this->message = $message;
    }
    /**
     * Transform the resource into an array.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return array
     */
    public function toArray($request)
    {
        return [
            'success'   => $this->status,
            'message'   => $this->message,
            'data'      => $this->resource
        ];
    }
}
```

Gambar 10. Resource Kapal

3.4. Routes Laravel

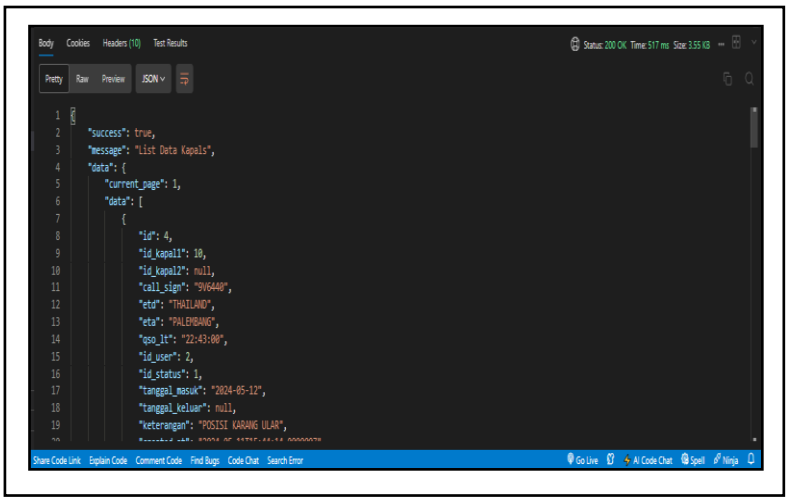
Routes berfungsi untuk mengatur lalu lintas file berdasarkan *request* dari pengguna. Pada Gambar 11 dijelaskan *konfigurasi* route di file api.php pada project ini.

```
// Route untuk Kapal
Route::apiResource('/kapals', App\Http\Controllers\Api\Operator\KapalController::class);
```

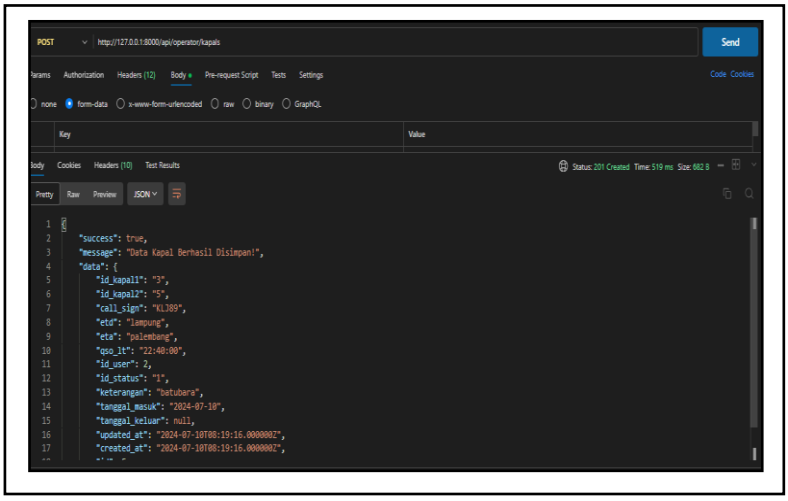
Gambar 11. Route API Kapal

3.5. Pengujian Postman

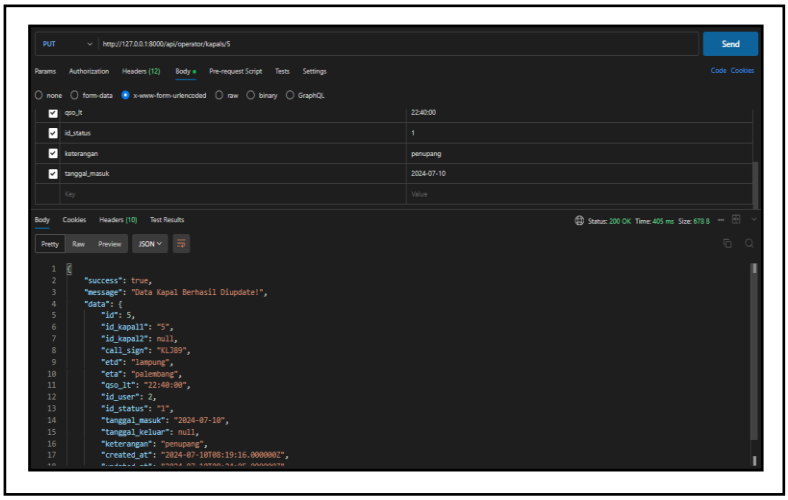
Berikut hasil pengujian *endpoint GET*, *POST*, *PUT* dan *DELETE* pada postman, dapat dilihat pada gambar dibawah ini.



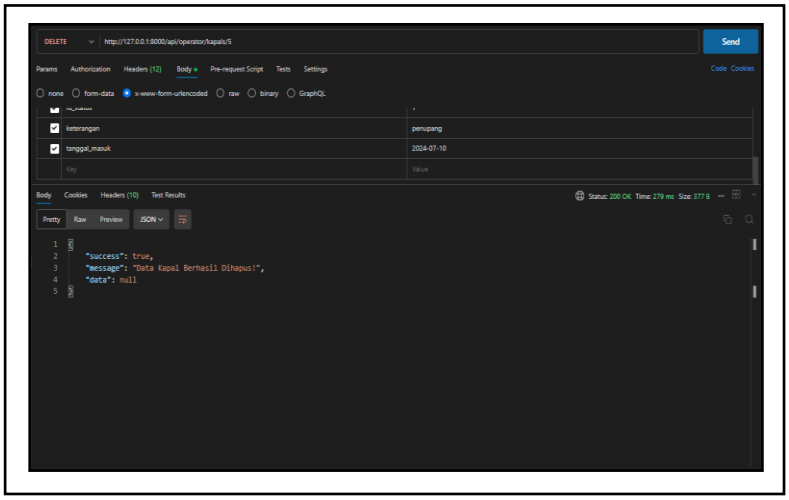
Gambar 12. Respon JSON GET Kapal



Gambar 13. Respon JSON POST Kapal



Gambar 14. Respon JSON PUT Kapal



Gambar 15. Respon JSON DELETE Kapal

3.6. Konfigurasi Axios pada NuxtJs

Berikut *konfigurasi* axios pada *project* NuxtJs yaitu pada file nuxt.config.js seperti pada Gambar 16.

```
axios: {
  baseURL: 'http://localhost:8000'
},
```

Gambar 16. Konfigurasi Axios

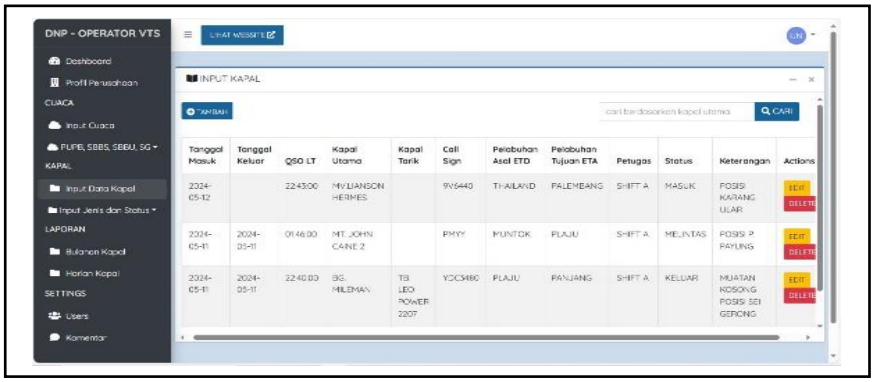
3.7. *Request POST* pada NuxtJs

Berikut cara *request API* pada *project* NuxtJs seperti pada Gambar 17.

```
await this.$axios
        .post("/api/operator/kapals", formData)
        .then(() => {
```

Gambar 17. Request API POST pada NuxtJs

3.8. Antarmuka Sistem



Gambar 18. Menu View Data Kapal

Pada Gambar 18 diatas adalah implementasi menu *view* data kapal untuk *role* operator dimana akan di tampilkan data kapal serta beberapa fitur yang dapat di akses oleh operator seperti tambah data, *edit* data, dan *delete* data kapal.

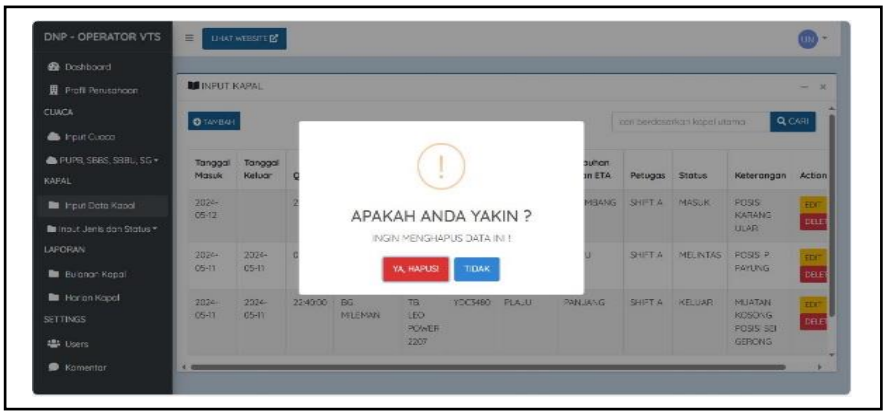
Gambar 19. Menu Tambah Data Kapal

Pada Gambar 19 diatas adalah implementasi menu untuk fitur tambah data kapal, dimana operator akan menginputkan data kapal antara lain tanggal masuk, tanggal keluar, qso lt, kapal utama, kapal tarik, call sign, pelabuhan asal ETD, pelabuhan tujuan ETA, status dan keterangan lalu tekan tombol simpan selanjutnya data akan di validasi apakah data yang di *input* bernilai *true*/benar atau *false*/salah. Jika bernilai *true*/benar akan muncul notifikasi sukses dan data berhasil simpan di *database*, jika bernilai *false*/salah maka akan muncul notifikasi salah atau pesan *erorr*, untuk tombol *reset* berfungsi untuk menghapus kembali data yang sudah di isi dan untuk tombol kembali berfungsi untuk kembali kehalaman sebelumnya.

Gambar 20. Menu Edit Data Kapal

Pada Gambar 20 diatas adalah implementasi menu *edit* data kapal untuk *role* operator akan mengedit data kapal antara lain tanggal masuk, tanggal keluar, qso lt, kapal utama, kapal tarik, call sign, pelabuhan asal ETD, pelabuhan tujuan ETA, status dan keterangan lalu tekan tombol simpan selanjutnya data akan di validasi apakah data yang di *Input* bernilai *true*/benar atau *false*/salah. Jika bernilai *true*/benar akan muncul notifikasi sukses dan data berhasil simpan di *database*, jika bernilai *false*/salah maka akan muncul notifikasi salah atau pesan *erorr*, untuk tombol *reset*

berfungsi untuk menghapus kembali data yang sudah di isi dan untuk tombol kembali berfungsi untuk kembali kehalaman sebelumnya.



Gambar 21. Menu Delete Data Kapal

Pada Gambar 21 diatas adalah dimana pada implementasi menu *view* data kapal ada tombol *delete* yang berfungsi untuk menghapus data *by id* ketika menekan tombol *delete* maka muncul notifikasi apakah data ini ingin di hapus jika tekan tombol ya,hapus maka akan hapus jika tekan tombol tidak maka data yang ingin *delete* akan di batalkan.

4. KESIMPULAN

Penelitian ini menyimpulkan bahwa sistem pendataan kapal yang dibangun dapat secara signifikan mempermudah operator dalam pendataan serta penyelesaian laporan harian dan bulanan kapal. Implementasi *REST API* pada sistem pendataan kapal dengan menggunakan teknologi *Client-Server*, di mana *REST Client* menggunakan framework Nuxt.js untuk mengonsumsi JSON dengan meminta layanan tertentu pada *REST Server* yang dibangun menggunakan framework Laravel, telah menunjukkan hasil yang positif. *REST Server* mampu menghasilkan respons berupa JSON dan menyediakan layanan yang diminta oleh Client dengan cepat dan akurat. Sistem pendataan kapal ini telah mendukung interoperabilitas dengan *REST API* berupa respons JSON. Pengujian *response* time pada *REST Server* telah dilakukan menggunakan platform postman, yang menunjukkan hasil *response* yang cepat dan sukses. Selain itu, beberapa pengujian sistem seperti *black-box testing* dan *white-box testing* juga telah dilakukan untuk menguji kesesuaian dan kehandalan sistem. Hasil pengujian ini menunjukkan bahwa sistem bekerja dengan baik dan memenuhi semua persyaratan yang ditetapkan. Secara keseluruhan, penelitian ini membuktikan bahwa sistem pendataan kapal yang dibangun tidak hanya efisien dalam hal operasional tetapi juga andal dan mudah diintegrasikan dengan sistem lain melalui *REST API*.

DAFTAR PUSTAKA

[1] F. Widyoutomo, H. Ajie, T. Elektro, T. Elektro, and U. Negeri, "Pengembangan Web Service," *J. Pinter*, vol. 5, no. 1, p. 8, 2021.

[2] R. C. Buwono, "Web Services Menggunakan Format JSON," *Respati*, vol. 14, no. 2, pp. 1–10, 2019, doi: 10.35842/jtir.v14i2.282.

[3] X. Chen, Z. Ji, Y. Fan, and Y. Zhan, "Restful API Architecture Based on Laravel Framework," vol. 910, pp. 1–6, 2019.

[4] N. Fitriani, H. Sulistiono, and D. Parwatiningtyas, "Sistem Informasi Dokumen Kapal pada PT Logindo Samudra Makmur Tbk," *J. Ris. dan Apl. Mhs. Inform.*, vol. 1, no. 03, pp. 320–326, 2020, doi: 10.30998/jrami.v1i03.333.

[5] KemenHub PM 19 Tahun 2022, "Peraturan Menteri Perhubungan Republik Indonesia Nomor Pm 19 Tahun 2022," *Menteri Kesehat. Republik Indones. Peratur. Menteri Kesehat. Republik Indones.*, vol. 69, no. 555, pp. 1–53, 2020.

[6] N. U. Mafra, "Pengaruh Kepuasan Kerja Dan Semangat Kerja Terhadap Produktivitas Kerja Pegawai Pada Distrik Navigasi Kelas I Palembang," *J. Ecoment Glob.*, vol. 2, no. 2, p. 9, 2019, doi: 10.35908/jeg.v2i2.248.

[7] B. I. Baharuddin, Hamka Wakkang, "Implementasi Web Service Dengan Metode Rest Api Untuk Integrasi Data Covid 19 Di Sulawesi Selatan," *J. Sintaks Log.*, vol. 2, no. 1, pp. 236–241, 2022, doi: 10.31850/jsilog.v2i1.1035.

[8] H. F. Herdiyatmoko, "Desain Sistem Backend Berbasis Rest Api Menggunakan Framework Laravel 7," *Skanika*, vol. 5, no. 2, pp. 136–144, 2022, doi: 10.36080/skanika.v5i2.2947.

[9] M. K. Naufal, F. Affrianto, and A. B. Cahyono, "Implementasi REST API Untuk Fitur Rencana Strategis Program Pada SIMPEDA," *Automata*, 2022.

[10] V. A. Rangga Gelar Guntara, "Implementasi dan Pengujian REST API Sistem Reservasi Ruang Rapat dengan Metode Black Box Testing," *J. Minfo Polgan*, vol. 12, no. 1, pp. 1229–1238, 2023, doi: 10.33395/jmp.v12i1.12691.

[11] A. C. M. Hanif and M. A. I. Pakereng, "Pengembangan Aplikasi Sistem Informasi Operasional Bus Berbasis Website Menggunakan Framework Laravel," *JATISI (Jurnal Tek. Inform. dan Sist. Informasi)*, vol. 8, no. 3, pp. 1027–1039, 2021, doi: 10.35957/jatisi.v8i3.973.

[12] S. bin Uzayr, *Learning WordPress REST API*, no. July. 2020.

[13] A. Sahi, "Aplikasi Test Potensi Akademik Seleksi Saringan Masuk Lp3i Berbasis Web Online Menggunakan Framework Codeigniter," vol. 7, no. 1, pp. 1–10, 2020.

[14] F. R. Maulayya, *Membangun Website CMS Dengan Laravel dan Nuxt Js*. 2022. [Online]. Available: https://eur-lex.europa.eu/legal-content/PT/TXT/PDF/?uri=CELEX:32016R0679&from=PT%0Ahttp://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:52012PC0011:pt:NOT

[15] O. D. Arianto and Y. A. Susetyo, "Penerapan Restful Web Service Dengan Framework Laravel Untuk Pembangunan Sistem Informasi Manajemen Sumber Daya Manusia," *JIPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform.*, vol. 7, no. 2, pp. 522–532, 2022, doi: 10.29100/jipi.v7i2.2870.

[16] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practioner's Approach*. 2020.

[17] M. A.S., Rosa dan Shalahuddin, *Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek*. Bandung, 2020.