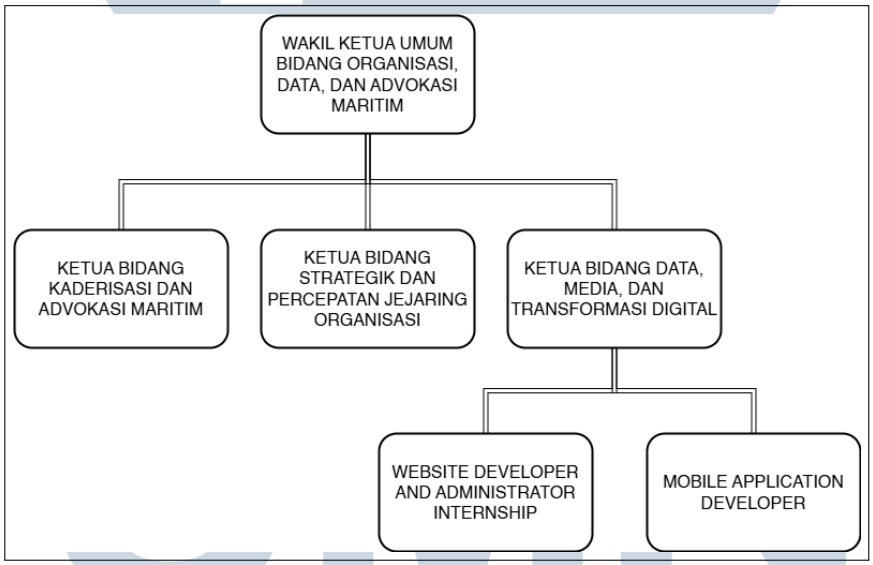
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama pelaksanaan kegiatan kerja magang di Perkumpulan Maritim Muda Nusantara, posisi yang ditempati adalah sebagai *Website Developer and Administrator*. Dalam menjalankan tanggung jawab pada posisi tersebut, kegiatan magang dilakukan di bawah bimbingan dari Bapak Nasrullah Abdulrahman Rafli Junaid selaku mentor dan Ketua Bidang Data, Media, dan Transformasi Digital. Posisi ini memiliki fokus utama pada pengelolaan serta pengembangan infrastruktur digital organisasi guna mendukung berbagai aktivitas dan kebutuhan internal maupun eksternal.



Gambar 3.1. Struktur Organisasi Tim *Website Developer and Administrator*

3.2 Tugas yang Dilakukan

Selama masa pelaksanaan kerja magang di Perkumpulan Maritim Muda Nusantara, tanggung jawab yang diberikan berkaitan dengan pengelolaan dan pengembangan sistem informasi berbasis web. Adapun rincian tugas yang dilaksanakan adalah sebagai berikut:

1. Bertanggung jawab dalam memastikan seluruh situs yang berada di bawah naungan Perkumpulan Maritim Muda Nusantara dapat berjalan dengan baik

dan optimal tanpa mengalami kendala teknis. Fokus utama pemantauan dan pemeliharaan dilakukan terhadap dua situs, yaitu maritimpreneur.com dan geoparksyouth.net sesuai dengan arahan yang telah diberikan sebelumnya.

2. Mengembangkan sebuah halaman baru bernama Events pada situs maritimpreneur.com. Halaman ini dirancang untuk menyediakan informasi terkait kegiatan-kegiatan kewirausahaan, khususnya yang berkaitan dengan sektor kemaritiman.

3. Pengembangan antarmuka pada situs maritimpreneur.com berdasarkan arahan serta masukan yang diberikan oleh mentor. Penyesuaian ini dilakukan guna meningkatkan kenyamanan dan kemudahan pengguna dalam mengakses informasi.

4. Pengembangan *Application Programming Interface* (API) pada situs hub. maritimmuda.id. API ini berfungsi sebagai jembatan integrasi untuk fitur *chat* pada aplikasi Maritimmuda Connect yang dikembangkan oleh tim *Mobile Application Developer*.

Tabel 3.1. *Timeline* Kerja Magang

Minggu ke-	**Kegiatan**
1	Pengenalan organisasi, lingkungan kerja, rekan kerja, pembimbing, pembagian kelompok kerja serta pembagian tugas awal.
2	Proses *cloning* repositori situs maritimpreneur.com dan geoparksyouth.net dari GitHub, serta mempelajari *source code* dari kedua situs tersebut, serta memastikan bahwa kedua situs berfungsi secara optimal.
3–5	Perancangan desain halaman Events pada situs maritimpreneur.com.
6–9	Implementasi desain halaman Events ke dalam *source code* maritimpreneur.com.
10–11	Membuat desain keseluruhan untuk memperbarui dan memperbaiki tampilan situs maritimpreneur.com.
12–13	Implementasi pembaruan desain situs maritimpreneur.com.

Minggu ke-	Kegiatan
14	Proses *cloning* repositori situs hub.maritimmuda.id dari GitHub serta memulai tahap analisa *source code* dari situs tersebut. Juga memulai perancangan awal API *chat*.
15–16	Perancangan model, *flow*, dan *database* API *chat* yang akan diintegrasikan dengan aplikasi Maritimmuda Connect.
17–19	Implementasi hasil rancangan API *chat*.
20–21	Uji coba API *chat* yang telah diimplementasikan pada *source code* situs hub.maritimmuda.id.

3.3 Uraian Kerja Magang

3.3.1 *Cloning* Repositori Situs Maritimpreneur.com dan Geoparksyouth.net

Pada minggu kedua pelaksanaan kegiatan magang, dilakukan proses *cloning* repositori dari proyek situs maritimpreneur.com dan geoparksyouth.net yang tersedia pada platform GitHub. Tujuan dari proses ini adalah untuk memperoleh salinan lokal dari *source code* sebagai dasar dalam pengembangan dan perancangan lebih lanjut terhadap kedua situs tersebut.

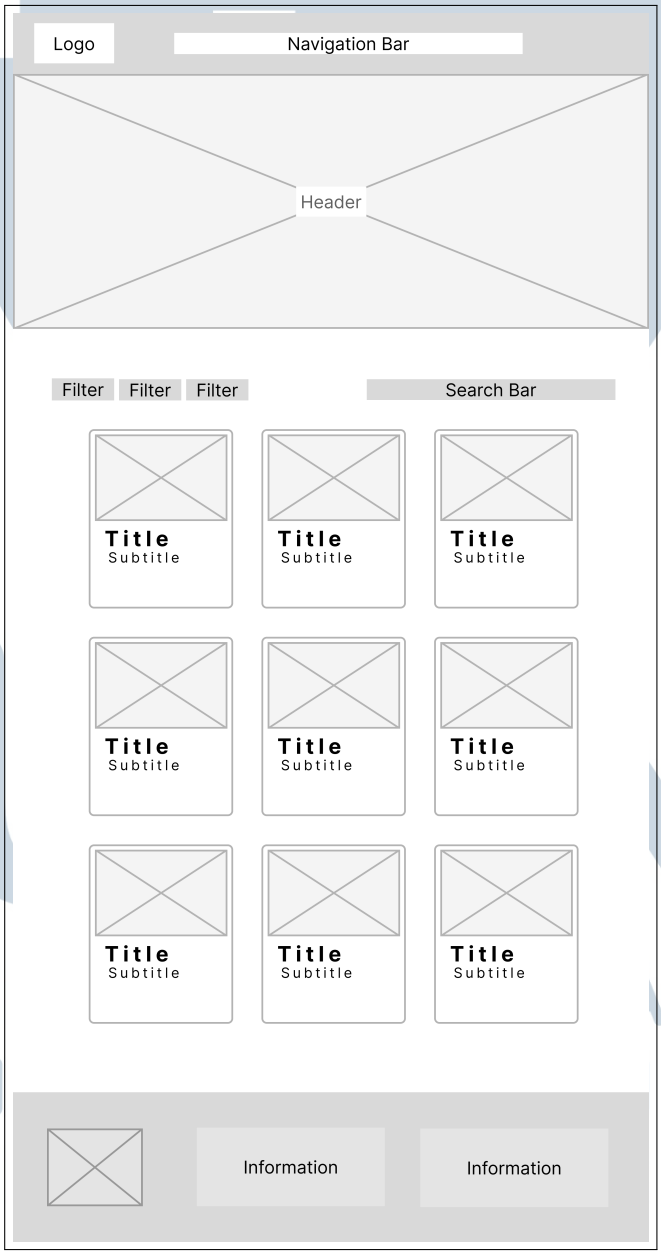
Adapun *repository* dari masing-masing situs dapat diakses melalui tautan berikut:

- Maritimpreneur.com: https://github.com/maritimmuda-id/maritimepreneur.com
- Geoparksyouth.net: https://github.com/maritimmuda-id/geoparksyouth.net

Setelah proses *cloning* selesai dilakukan, *source code* dari kedua situs tersebut dianalisis secara menyeluruh guna memahami struktur proyek, alur kerja aplikasi, serta teknologi yang digunakan dalam proses pengembangannya. Analisis ini mencakup identifikasi *dependency* proyek, konfigurasi lingkungan (*environment*), serta cara kerja setiap komponen utama dalam sistem. Tujuan dari kegiatan ini adalah untuk memastikan bahwa proses pengembangan dan pemeliharaan aplikasi dapat berjalan secara efisien, terstruktur, dan sesuai dengan standar teknis yang telah ditetapkan oleh tim pengembang.

3.3.2 Perancangan Desain Halaman Events pada Situs Maritimpreneur.com

Gambar 3.2 menunjukkan rancangan *low-fidelity prototype* untuk halaman Events yang direncanakan sebagai fitur tambahan pada situs maritimpreneur.com. Desain ini bertujuan untuk memberikan gambaran awal mengenai arsitektur informasi (*information architecture*) dan hierarki visual (*visual hierarchy*) halaman sebelum sumber daya dialokasikan untuk pengembangan yang lebih detail [5].

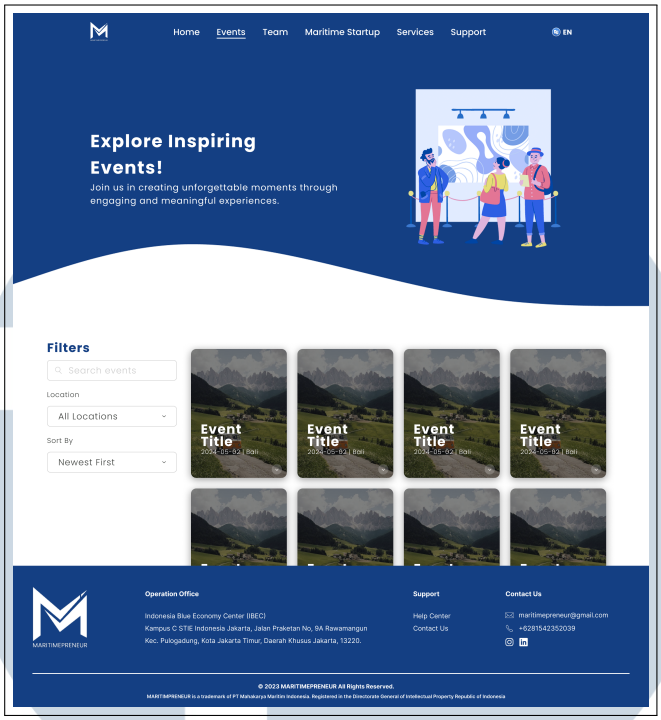


Gambar 3.2. *Low-fidelity Prototype* Halaman Events

Bagian atas halaman mencakup elemen-elemen seperti logo, *navigation bar*, dan *header*. Penempatan konsisten elemen-elemen ini sangat penting untuk membangun identitas merek dan kemudahan navigasi, sejalan dengan Heuristik *Usability* Nielsen tentang ”Konsistensi dan Standar” (*Consistency and Standards*) [6]. Selanjutnya, terdapat komponen fungsional seperti *filter* dan *search bar* yang membentuk sistem navigasi facet (*faceted navigation*). Sistem ini membantu pengguna mempersempit hasil pencarian, yang secara efektif menjembatani ”Gulf of Execution and Evaluation”—celah antara intensi pengguna dan aksi yang diperlukan—sebuah konsep dari Don Norman [7].

Konten utama pada halaman Events ditampilkan dalam bentuk kumpulan *card* yang tersusun secara *grid*. Desain berbasis *card* ini sejalan dengan Prinsip Gestalt tentang Kedekatan (*Proximity*), di mana elemen yang berdekatan dipersepsikan sebagai satu kelompok [8]. Susunan *grid* juga mendukung pola pemindaian alami pengguna seperti pola-F (*F-Pattern*) [9]. Setiap *card* menerapkan prinsip *progressive disclosure* dengan menyajikan informasi penting terlebih dahulu untuk mengurangi beban kognitif (*cognitive load*) [10]. Penataan ini dirancang agar tetap responsif (*responsive*) di berbagai perangkat, menggunakan teknik seperti yang dipopulerkan oleh Ethan Marcotte [11].

Pada bagian bawah halaman, ditambahkan elemen *footer* yang berfungsi sebagai sistem navigasi sekunder. Keberadaan *footer* ini mendukung konsep *wayfinding* dalam ruang digital dan sejalan dengan teori pencarian informasi (*information foraging theory*) [12]. Secara keseluruhan, desain yang masih bersifat *low-fidelity* ini fokus pada penempatan elemen (struktur dan alur navigasi) untuk mempermudah evaluasi awal, sebelum masuk ke tahap *high-fidelity prototype* yang akan menyempurnakan aspek visual seperti warna dan tipografi sesuai dengan Heuristik Nielsen tentang Desain Estetik dan Minimalis [6].



Gambar 3.3. *High-fidelity Prototype* Halaman Events

Tahapan berikutnya dalam proses perancangan antarmuka adalah mengembangkan *low-fidelity prototype* menjadi *high-fidelity prototype* untuk halaman Events. Sebagaimana ditunjukkan pada Gambar 3.3, rancangan antarmuka pada tahap ini telah disempurnakan. Transisi ini bukan hanya sekadar penambahan estetika, melainkan sebuah langkah krusial untuk melakukan uji usabilitas yang lebih mendalam dan memvalidasi alur interaksi. Tujuannya adalah untuk memberikan representasi yang akurat mengenai produk akhir, yang sejalan dengan prinsip *Aesthetic-Usability Effect*, di mana pengguna seringkali mempersepsikan desain yang lebih estetis sebagai desain yang lebih mudah digunakan [13].

Beberapa penyesuaian utama yang diterapkan pada *high-fidelity prototype* antara lain:

- Penggunaan Warna dan Tipografi: Desain telah dilengkapi dengan skema warna yang konsisten, termasuk penggunaan warna utama (biru laut). Pemilihan warna ini merepresentasikan tema maritim serta membangun identitas merek yang profesional namun tetap ramah.
- Visualisasi Konten: Elemen visual seperti ilustrasi dan gambar pendukung pada *card event* ditambahkan untuk memberikan konteks. Hal ini didasarkan

pada *Picture Superiority Effect*, sebuah fenomena kognitif di mana gambar dan citra lebih mudah diingat daripada teks, sehingga dapat memperkuat daya tarik dan pemahaman informasi acara [14].

- Desain Kartu Event: Setiap *card* kini menampilkan informasi yang lebih komprehensif. Penataan elemen seperti judul, lokasi, dan tanggal diatur berdasarkan hierarki informasi (information hierarchy) yang jelas, menerapkan Prinsip Gestalt tentang Wilayah Bersama (*Common Region*) untuk memastikan semua informasi dalam kartu dipersepsikan sebagai satu kesatuan yang utuh [8].
- Navigasi dan Interaktivitas: Elemen navigasi seperti *search bar* dan *category filter* telah disusun ulang agar lebih intuitif. Penyesuaian ini bertujuan untuk mengurangi beban kognitif (*cognitive load*) pengguna saat mencari informasi, sehingga meningkatkan efisiensi dan kepuasan dalam pengalaman pencarian [15].

Proses transisi dari *low-fidelity* ke *high-fidelity* ini dilakukan menggunakan aplikasi Figma. Alat desain kolaboratif ini dipilih untuk memastikan bahwa spesifikasi desain seperti warna, tipografi, dan jarak dapat diimplementasikan dengan tingkat ketepatan yang tinggi pada sisi *frontend*.

3.3.3 Perancangan Desain Antarmuka Maritimpreneur.com

Pada tahap ini, tampilan keseluruhan situs maritimpreneur.com dikembangkan ulang dengan mengacu pada desain halaman Events sebagai acuan utama. Proses pengembangan ini mencakup penyelarasan *layout* (tata letak) di seluruh halaman, standarisasi penggunaan tipografi, pemilihan aset visual dengan kualitas yang lebih tinggi, serta penyesuaian ukuran aset agar sesuai dengan kebutuhan desain dan memastikan konsistensi tampilan antarmuka di seluruh bagian situs.

A Desain Halaman *Home*

Gambar 3.4. Desain Halaman Home

Gambar 3.4 merupakan tampilan dari halaman Home pada situs maritimpreneur.com. Pada bagian (a) ditampilkan *navigation bar* yang merupakan komponen krusial dari arsitektur informasi (*information architecture*) situs. Kejelasan navigasi ini secara langsung memengaruhi *findability*, atau kemudahan pengguna dalam menemukan informasi yang dibutuhkan [16]. Selain itu, terdapat bagian *header*, yang dalam praktik desain modern sering disebut sebagai *Hero Section*. Area ini dirancang untuk secara cepat mengkomunikasikan proposisi nilai (*value proposition*) situs melalui kombinasi judul utama, slogan, dan ilustrasi yang memperkuat identitas dan menarik perhatian pengunjung [17].

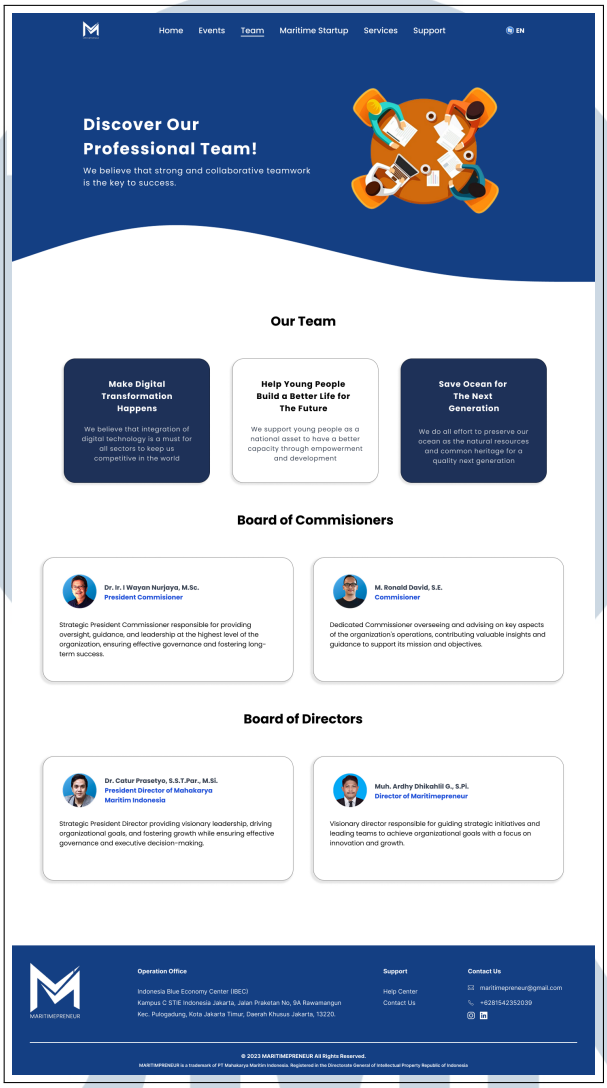
Pada bagian (b), ditampilkan beberapa fitur utama yang ditawarkan oleh Maritimpreneur, di antaranya:

- Maritimpreneur Business Academy
- Maritimpreneur Venture Builder
- Maritimpreneur Event
- Maritimpreneur Tech

Selanjutnya, pada bagian (c) terdapat informasi mengenai *Our Clients*, yang menampilkan logo-logo dari institusi atau mitra yang telah bekerja sama. Bagian ini berfungsi sebagai bentuk bukti sosial (*social proof*), sebuah prinsip persuasi di mana orang mengasumsikan tindakan orang lain sebagai cerminan perilaku yang benar, sehingga hal ini dapat meningkatkan kepercayaan pengunjung [18]. Terdapat pula bagian *Frequently Asked Questions* (FAQ) yang bertujuan membantu pengguna mengatasi masalah umum secara mandiri (*self-serve*), sehingga dapat mengurangi friksi dan meningkatkan pengalaman pengguna secara keseluruhan [19].

Pada bagian (d), ditampilkan formulir *Contact Us* yang memungkinkan pengguna mengirimkan pesan secara langsung. Kemudahan formulir ini, yang hanya terdiri dari kolom esensial dan tombol *Call to Action* (CTA) yang jelas, merupakan penerapan praktik terbaik desain formulir web untuk memaksimalkan usabilitas dan tingkat konversi [20]. Bagian paling bawah halaman, yaitu *footer*, mencantumkan logo, hak cipta, dan tautan penting lainnya. Desain *footer* yang sederhana dan tidak memuat terlalu banyak tautan sejalan dengan Hukum Miller (*Miller's Law*), yang menyatakan bahwa rata-rata manusia hanya dapat menyimpan sekitar tujuh item dalam memori kerja mereka, sehingga antarmuka yang simpel lebih efektif [21].

B Desain Halaman Team

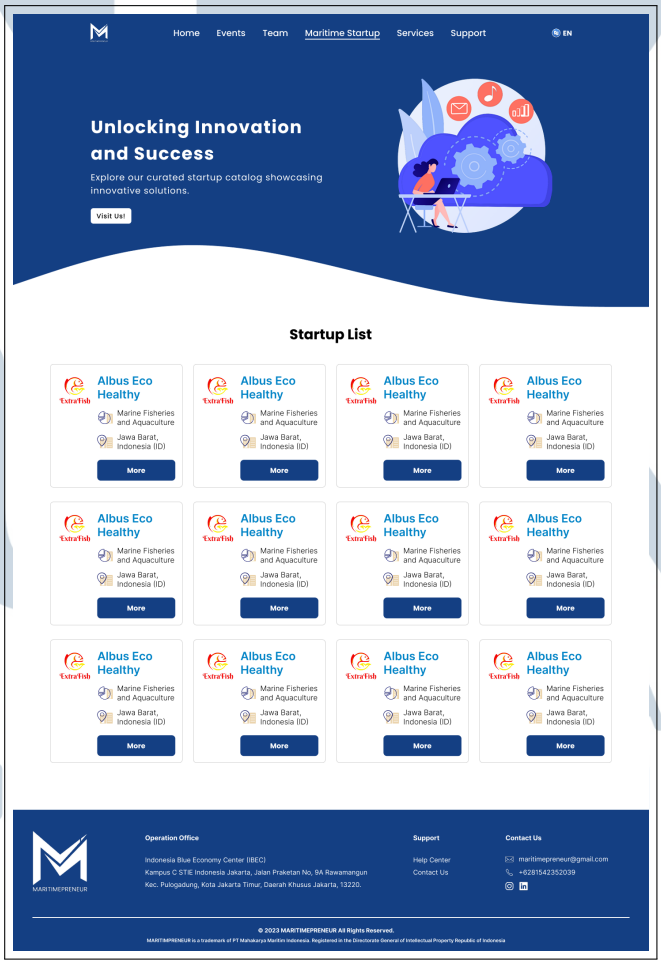


Gambar 3.5. Desain Halaman Team

Halaman Team berfungsi untuk menampilkan informasi mengenai struktur kepengurusan Maritimpreneur. Sama seperti halaman Home sebelumnya, halaman Team juga diawali dengan bagian pembuka yang berisi ilustrasi serta teks yang menggambarkan semangat kolaboratif tim. Bagian ini dirancang untuk mengkomunikasikan filosofi dan budaya organisasi, sebuah aspek penting selain proposisi nilai produk [17]. Setelah itu, ditampilkan tiga nilai inti yang menjadi dasar kerja tim dalam bentuk tiga buah kartu. Penggunaan kartu ini menerapkan Prinsip Gestalt tentang Kedekatan (Proximity), di mana setiap ikon dan teksnya dipersepsikan sebagai satu unit yang utuh [8].

Pada bagian berikutnya, halaman menampilkan daftar *Board of Commissioners* dan *Board of Directors*. Masing-masing bagian ini menggunakan Prinsip Gestalt tentang Wilayah Bersama (*Law of Common Region*) dengan menempatkan setiap dewan dalam area terpisah untuk memudahkan pengguna membedakan kedua kelompok tersebut [22]. Setiap anggota dewan disajikan dalam kartu profil yang memuat foto, nama, dan jabatan; sebuah pendekatan untuk menghumanisasi desain (*humanizing design*) dan membangun kepercayaan [23]. Seluruh elemen ini menerapkan konsistensi visual pada warna, tipografi, dan ikon yang sejalan dengan Heuristik Nielsen tentang Konsistensi dan Standar untuk memastikan tampilan yang profesional dan mudah dipahami [6].

C Desain Halaman Maritime Startup

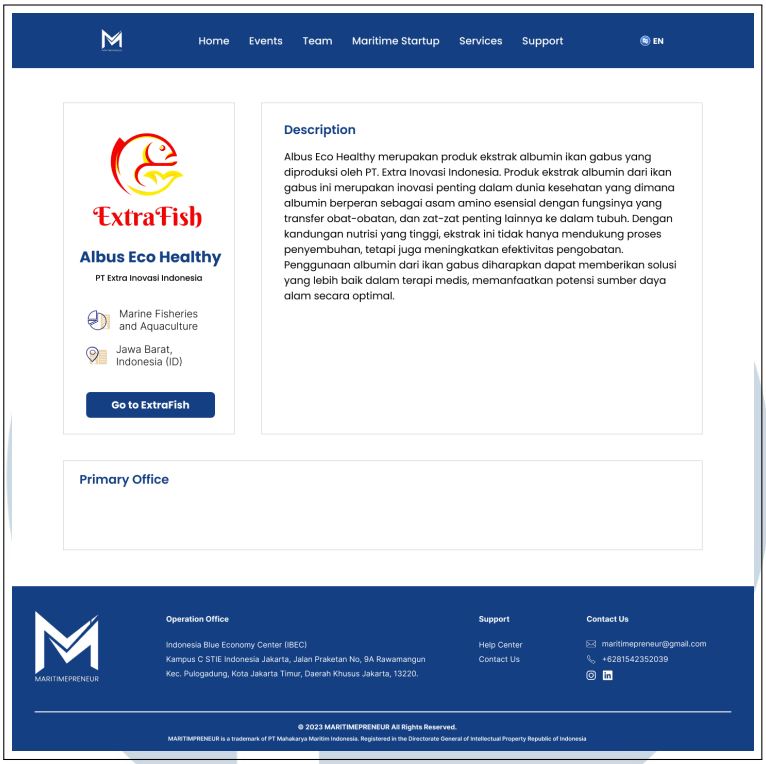


Gambar 3.6. Desain Halaman Maritime Startup

Halaman Maritime Startup (Gambar 3.6) dirancang sebagai katalog digital yang menampilkan daftar startup di sektor kelautan dan perikanan. Setiap entri ditampilkan dalam bentuk elemen visual yang seragam, sebuah penerapan dari Prinsip Gestalt tentang Kesamaan (*Law of Similarity*), yang membantu pengguna untuk langsung mengenali bahwa setiap item memiliki fungsi yang setara [8]. Informasi ringkas yang dimuat pada setiap entri adalah sebagai berikut:

- Nama startup
- Bidang industri atau sektor usaha
- Lokasi operasional
- Logo perusahaan
- Kategori perusahaan
- Status sertifikasi
- Tombol aksi untuk melihat detail

Penyusunan konten ini berfungsi sebagai jejak informasi (*information scent*), yang memberikan petunjuk bagi pengguna untuk menilai apakah mereka ingin menggali lebih dalam, sesuai dengan teori *Information Foraging* [12]. Struktur berulang yang dapat diprediksi ini juga mengurangi beban kognitif (*cognitive load*) pengguna [15].



Gambar 3.7. Desain Detail Startup pada Halaman Maritime Startup

Ketika pengguna memilih salah satu entri, sistem akan mengarahkan ke halaman detail (Gambar 3.7). Halaman ini menyajikan informasi yang lebih lengkap dan terstruktur, menciptakan sebuah pemetaan (*mapping*) yang jelas antara tindakan pengguna (mengklik tombol) dengan hasil yang didapatkan, sebuah prinsip fundamental dalam desain interaksi [7]. Informasi yang ditampilkan antara lain:

- Deskripsi singkat mengenai *startup*
- Profil dan nama perusahaan
- Lokasi operasional dan alamat lengkap
- Tautan eksternal (jika tersedia) untuk mengakses situs web atau sumber informasi lain

Pemisahan antara halaman daftar dan halaman detail ini merupakan penerapan dari pola desain Master-Detail (Master-Detail *Pattern*), yang efektif untuk mengelola kumpulan data yang kompleks secara sistematis [24]. Pola ini memberikan pengalaman pengguna yang lebih terfokus karena sesuai dengan model

mental pengguna: mode pemindaian pada halaman daftar dan mode fokus pada halaman detail, sehingga pengguna tidak perlu berpikir keras untuk menavigasi konten [16].

D Desain Halaman Services

Gambar 3.8. Desain Halaman Services

Halaman Services dirancang untuk memperkenalkan berbagai layanan profesional yang ditawarkan oleh platform maritimepreneur.com. Struktur halaman dibagi menjadi empat bagian utama layanan, sebuah strategi pengelompokan konten atau *chunking* yang sejalan dengan Hukum Miller (*Miller's Law*). Dengan menyajikan informasi dalam jumlah yang terbatas (empat bagian), pengguna dapat memproses dan memahami gambaran besar layanan secara lebih efisien tanpa

merasa terbebani [21]. Setiap bagian diwakili oleh judul, ilustrasi visual, dan daftar poin yang merangkum cakupan layanan.

Secara visual, halaman ini mengimplementasikan komposisi dua kolom di setiap bagian layanan. Penempatan teks deskriptif yang bersebelahan dengan ilustrasi tematik merupakan penerapan dari Prinsip Gestalt tentang Kedekatan (*Proximity*), yang secara visual mengisyaratkan bahwa kedua elemen tersebut saling berhubungan erat [8]. Komposisi seimbang antara informasi verbal dan visual ini juga didasarkan pada Teori Pengkodean Ganda (*Dual-Coding Theory*), yang menyatakan bahwa otak memproses kata dan gambar melalui saluran terpisah. Kombinasi keduanya akan memperkuat pemahaman dan ingatan pengguna terhadap konteks layanan yang ditawarkan [14].

E Desain Halaman Support

Gambar 3.9. Desain halaman dukungan Support

Halaman Support, seperti yang ditunjukkan pada Gambar 3.9, dirancang untuk menyediakan akses langsung terhadap dokumen-dokumen penting. Tujuan utama halaman ini adalah untuk memaksimalkan efisiensi pengguna dalam menyelesaikan tugas spesifik (menemukan dan mengunduh dokumen), sebuah

implementasi dari prinsip usabilitas inti untuk tidak membuat pengguna berpikir keras [16].

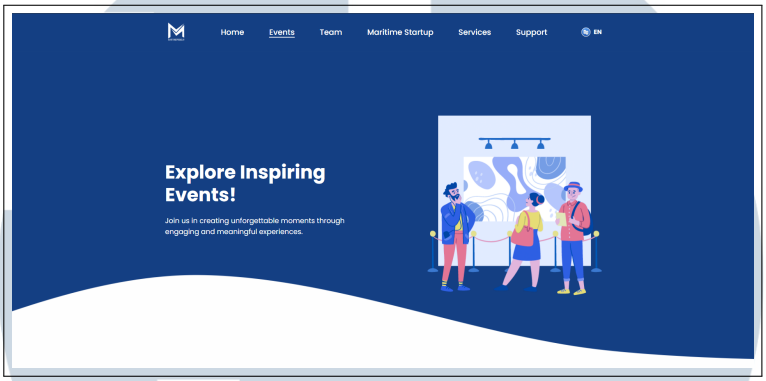
Pada bagian tengah halaman, ditampilkan judul *Essential Downloads* yang dilanjutkan dengan tiga kartu unduhan. Pembatasan jumlah pilihan menjadi tiga item ini sejalan dengan Hukum Miller (*Miller's Law*), yang membantu mengurangi beban kognitif pengguna [21]. Dokumen-dokumen yang tersedia antara lain:

- Company Profile
- Indonesia – Law of the Sea
- Indonesian Ocean Policy

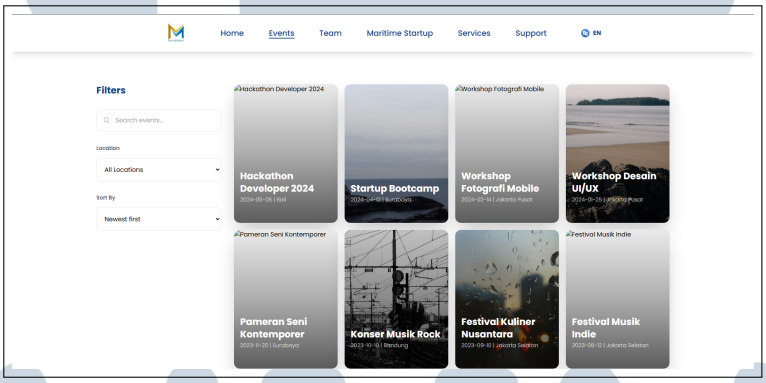
Setiap kartu memiliki desain seragam, menerapkan Prinsip Gestalt tentang Kesamaan (*Law of Similarity*) untuk menandakan bahwa semua item memiliki fungsi yang serupa [8]. Penggunaan tombol *Download* berwarna kontras menciptakan hierarki visual (*visual hierarchy*) yang kuat, menarik perhatian pengguna langsung ke tindakan utama yang dapat dilakukan [25]. Desain tombol yang jelas ini juga memberikan *affordance* yang tinggi, di mana properti visualnya secara intuitif mengisyaratkan cara penggunaannya [7]. Tata letak kartu yang menggunakan grid tiga kolom yang responsif [11] memastikan keseimbangan visual dan fungsionalitas di berbagai ukuran perangkat.

3.3.4 Implementasi Desain Maritimpreneur.com

A Halaman Events



(a)



(b)

Gambar 3.10. Halaman Events

Pada tahap ini, desain halaman Events telah diimplementasikan ke dalam *source code* dari situs maritimpreneur.com. Halaman ini dibangun menggunakan *React Functional Component*.

```
import HeaderEvent from "../component/Eventpage/HeaderEvent.jsx";
import NavBar from "../component/Navbar";
import Footer from "../component/Footer.jsx";
import EventList from "../component/Eventpage/EventList.jsx";

const EventPage = () => {
    return (
        <div>
            <NavBar/>
            <HeaderEvent/>
            <EventList/>
            <Footer/>
```

```
        </div>
    );
}

export default EventPage;
```

Kode 3.1: Komponen Halaman Events

Kode 3.1 merupakan *source code* dari halaman Events. Komponen `EventPage` ini menerapkan pola desain arsitektur berbasis komponen (*Component-Based Architecture*), sebuah paradigma fundamental dalam rekayasa perangkat lunak modern yang menekankan pada modularitas dan penggunaan kembali kode [26].

Dalam struktur ini, `EventPage` berperan sebagai *container component* (komponen penampung). Tugasnya bukan untuk menampilkan elemen UI secara langsung, melainkan untuk menyusun dan mengelola beberapa komponen lain yang lebih spesifik. Pola pemisahan antara komponen penampung (yang mengatur data dan *state*) dan komponen *presentasional* (yang hanya menampilkan UI) ini dipopulerkan untuk menjaga agar struktur aplikasi tetap terorganisir dan mudah dikelola [27]. Komponen-komponen seperti `<NavBar/>`, `<HeaderEvent/>`, `<EventList/>`, dan `<Footer/>` diimpor dan disusun di dalamnya untuk membentuk keseluruhan tata letak dan fungsionalitas halaman Events secara kohesif.

```
import React, { useEffect, useContext } from "react";
import BgEvent from "../../assets/bg-event.png";
import { LanguageContext } from "../../utils/LanguageContext";
import AOS from "aos";

const HeaderEvent = () => {
  const { t } = useContext(LanguageContext);
  useEffect(() => {
    AOS.init({
      once: true,
      duration: 1500,
    });
  }, []);

  return (
    <div className="bg-cover bg-center min-h-screen flex justify-center
    items-center team-hero bg-service">
      <div className="container xl:space-x-28 2xl:px-52 -mt-10 mb-20 mx-auto px-6
    sm:px-20 py-8 items-center flex flex-col md:flex-row">
        <div className="md:w-1/2 md:pr-8" data-aos="fade-right">
          <p className="text-white font-bold text-3xl lg:text-5xl mb-6 text-team
    event-mt">
```

```
            {t("eventtitle")}
          </p>
          <p className="text-white text-l sm:text-lg mb-8">
            {t("eventsubtitle")}
          </p>
        </div>
        <div className="md:w-1/3" data-aos="fade-left">
          <img
            className="rounded-lg lg:max-w-md max-w-xs"
            src={BgEvent}
            alt="gambar"
          />
        </div>
      </div>
    </div>
  );
};

export default HeaderEvent;
```

Kode 3.2: Komponen *Header*

Kode 3.2 merupakan komponen yang digunakan di setiap halaman situs maritimpreneur.com. Fungsi dari komponen ini adalah untuk menampilkan bagian *header* pada seluruh halaman situs, sehingga menciptakan konsistensi tampilan dan navigasi antarhalaman.

```
                  import Aos from "aos";
import { useEffect, useState } from "react";
import getEventInitialData from "../../utils/EventData";
import EventCard from "./EventCard";
import { IoIosSearch, IoMdClose } from "react-icons/io";
import { AiOutlineExclamationCircle } from "react-icons/ai";

const EventList = () => {
  const events = getEventInitialData();
  const [isSearching, setIsSearching] = useState(false);
  const [searchTerm, setSearchTerm] = useState("");
  const [selectedLocation, setSelectedLocation] = useState("");
  const [sortBy, setSortBy] = useState("newest");

  const locations = [...new Set(events.map((event) => event.location))];

  const filteredEvents = events
    .filter((event) => {
      const matchesSearch = event.title
        .toLowerCase()
        .includes(searchTerm.toLowerCase());
      const matchesLocation =
        selectedLocation === "" \/ event.location === selectedLocation;
      return matchesSearch /\ matchesLocation;
    })
```

```
    .sort((a, b) => {
      switch (sortBy) {
        case "newest":
          return new Date(b.date) - new Date(a.date);
        case "oldest":
          return new Date(a.date) - new Date(b.date);
        case "name-asc":
          return a.title.localeCompare(b.title);
        case "name-desc":
          return b.title.localeCompare(a.title);
        default:
          return 0;
      }
    });

  useEffect(() => {
    Aos.init({
      once: true,
      duration: 2000,
    });
  }, []);

  return (
    <div className="container mx-auto px-4 py-8 lg:px-8">
      <div className="flex flex-col lg:flex-row gap-8">
        <div className="w-full lg:w-80 space-y-8" data-aos="fade-right">
          <h1 className="text-2xl font-bold text-primary-color">
            Filters
          </h1>

          <div className="relative">
            <input
              type="text"
              placeholder="Search events..."
              className="w-full pl-12 pr-8 py-4 rounded-lg border border-gray-200
    focus:border-blue-500 focus:ring-2 focus:ring-blue-200 transition-colors"
              value={searchTerm}
              onChange={(e) => {
                setSearchTerm(e.target.value);
                setIsSearching(true);
              }}
            />
            <IoIosSearch className="absolute left-4 top-1/2 -translate-y-1/2
    text-gray-400 text-xl" />
            {isSearching ∧ searchTerm ≠= "" ∧ (
              <button
                onClick={() => {
                  setIsSearching(false);
                  setSearchTerm("");
                }}
                className="absolute right-4 top-1/2 -translate-y-1/2"
              >
                <IoMdClose className="scale-125 fill-black/60" />
              </button>
```

```
            ))}
          </div>

          <div className="space-y-4">
            <label className="block text-sm font-medium text-gray-700">
              Location
            </label>
            <select
              className="w-full px-4 py-4 rounded-lg border border-gray-200
    focus:border-blue-500 focus:ring-2 focus:ring-blue-200 transition-colors"
              value={selectedLocation}
              onChange={(e) =>
                setSelectedLocation(e.target.value)
              }
            >
              <option value="">All Locations</option>
              {locations.map((location, index) => (
                <option key={index} value={location}>
                  {location}
                </option>
              ))}
            </select>
          </div>

          <div className="space-y-4">
            <label className="block text-sm font-medium text-gray-700">
              Sort By
            </label>
            <select
              className="w-full px-4 py-4 rounded-lg border border-gray-200
    focus:border-blue-500 focus:ring-2 focus:ring-blue-200 transition-colors"
              value={sortBy}
              onChange={(e) => setSortBy(e.target.value)}
            >
              <option value="newest">Newest first</option>
              <option value="oldest">Oldest first</option>
              <option value="name-asc">Name A to Z</option>
              <option value="name-desc">Name Z to A</option>
            </select>
          </div>
        </div>

        <div className="flex-1" data-aos="fade-left">
          {filteredEvents.length > 0 ? (
            <div className="grid grid-cols-1 sm:grid-cols-2 xl:grid-cols-3 2
    xl:grid-cols-4 gap-4">
              {filteredEvents.map((event) => (
                <EventCard key={event.id} {...event} />
              ))}
            </div>
          ) : (
            <div className="h-96 flex flex-col items-center justify-center
    text-center">
              <div className="text-gray-400 text-5xl mb-4">
```

```
                <AiOutlineExclamationCircle />
              </div>
              <h3 className="text-xl font-semibold text-gray-800">
                No events found
              </h3>
            </div>
          )}
        </div>
      </div>
    </div>
  );
};

export default EventList;
```

Kode 3.3: Komponen *Event List*

Komponen `EventList` merupakan komponen inti pada halaman Events yang berfungsi untuk menampilkan daftar acara atau kegiatan. Komponen ini mengambil data acara, baik dari API maupun data statis, dan merendernya dalam format yang terstruktur. Di dalam komponen ini juga terdapat komponen `EventCard` yang bertugas untuk menampilkan detail singkat dari masing-masing acara.

```
import React from "react";
import { Link } from "react-router-dom";

const EventCard = (props) => {
  const { id, title, description, date, image, location } = props;

  return (
    <div className="group bg-white w-full max-w-[320px] min-w-[240px] rounded-2xl shadow-2xl overflow-hidden border border-gray-200 hover:shadow-lg transition-all duration-300 relative aspect-[3/4]">
      <div className="relative w-full h-full">
        <img
          className="w-full h-full object-cover object-center"
          src={image}
          alt={title}
          loading="lazy"
        />

        <div
          className={`absolute inset-0 bg-gradient-to-t from-black/80 via-black/40 to-black/10 transition-opacity duration-500 opacity-80`}
        />
      </div>

      <div
        className={`absolute inset-x-0 bottom-8 p-4 transition-all duration-500 ease-in-out`}
```

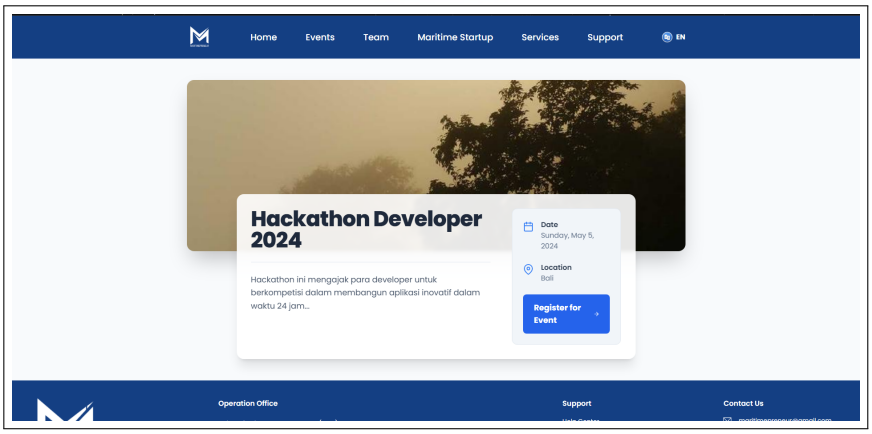
```
      >
        <Link to={`/events/detail/${id}`}>
          <h1
            className={`text-white text-2xl font-bold line-clamp-2 hover:underline
    cursor-pointer`}
          >
            {title}
          </h1>
        </Link>

        <span className="text-white/75 text-sm font-light">
          {date} | {location}
        </span>
      </div>
    </div>
  );
};

export default EventCard;
```

Kode 3.4: Komponen *Event Card*

Komponen `EventList` akan merender `EventCard`, yang kemudian berfungsi sebagai penampung informasi individual dari setiap acara atau kegiatan yang tersedia.



Gambar 3.11. Komponen Detail *Event*

Selain komponen `EventList` dan `EventCard` yang merupakan dua komponen penting pada halaman Events, terdapat pula komponen `EventDetail`, seperti yang ditunjukkan pada Gambar 3.11. Komponen ini memiliki peran krusial dalam menyajikan informasi dan deskripsi lengkap mengenai acara yang akan diselenggarakan, dan dapat diakses dengan mengklik judul kegiatan yang ditampilkan.

```jsx
import React, { useState, useEffect, useContext } from "react";
import { useParams } from "react-router-dom";
import getEventInitialData from "../../utils/EventData";

import {
  FiCalendar,
  FiMapPin,
  FiAlertCircle,
  FiArrowRight,
} from "react-icons/fi";
import { LanguageContext } from "../../utils/LanguageContext";

const EventDetail = () => {
  const { id } = useParams();
  const [event, setEvent] = useState(null);
  const [loading, setLoading] = useState(true);
  const { t } = useContext(LanguageContext);

  useEffect(() => {
    setLoading(true);

    const timer = setTimeout(() => {
      const allEvents = getEventInitialData();
      const foundEvent = allEvents.find((e) => e.id === Number(id));
      setEvent(foundEvent);
      setLoading(false);
    }, 500);

    return () => clearTimeout(timer);
  }, [id]);

  if (loading) {
    return (
      <div className="flex justify-center items-center h-screen text-lg
    text-gray-500">
        Loading Event...
      </div>
    );
  }

  if (!event) {
    return (
      <div className="flex flex-col justify-center items-center h-screen
    text-gray-500">
        <FiAlertCircle className="text-6xl text-red-400 mb-4" />
        <h2 className="text-2xl font-bold text-red-500">
          Event Not Found
        </h2>
        <p className="mt-2">
          Sorry, we couldn't find an event with this ID.
        </p>
      </div>
    );
  }
```

```
53
54    return (
55      <div className="bg-slate-50 font-sans">
56        <div className="container mx-auto max-w-6xl py-12 px-4">
57          <div
58            className="w-full h-72 lg:h-96 bg-cover bg-center rounded-2xl shadow-2xl
      "
59            style={{ backgroundImage: `url(${event.image})` }}
60          />
61          <div className="relative -mt-32">
62            <div className="bg-white/90 backdrop-blur-sm p-8 mx-auto max-w-4xl
      rounded-2xl shadow-xl">
63              <div className="grid grid-cols-1 lg:grid-cols-3 gap-x-12">
64                <div className="lg:col-span-2 space-y-6">
65                  <h1 className="text-4xl lg:text-5xl font-extrabold text-slate-800
      tracking-tight">
66                    {event.title}
67                  </h1>
68                  <hr className="border-slate-200" />
69                  <p className="text-slate-600 text-lg leading-relaxed">
70                    {event.description}
71                  </p>
72                </div>
73                <div className="mt-8 lg:mt-0">
74                  <div className="bg-slate-100 p-6 rounded-xl border
      border-slate-200 space-y-6">
75                    <div className="flex items-start space-x-4">
76                      <FiCalendar className="text-blue-500 text-2xl flex-shrink-0
      mt-1" />
77                      <div>
78                        <h4 className="font-semibold text-slate-700">
79                          Date
80                        </h4>
81                        <p className="text-slate-500">
82                          {new Date(
83                            event.date
84                          ).toLocaleDateString("en-US", {
85                            weekday: "long",
86                            year: "numeric",
87                            month: "long",
88                            day: "numeric",
89                          })}
90                        </p>
91                      </div>
92                    </div>
93                    <div className="flex items-start space-x-4">
94                      <FiMapPin className="text-blue-500 text-2xl flex-shrink-0 mt-1
      " />
95                      <div>
96                        <h4 className="font-semibold text-slate-700">
97                          Location
98                        </h4>
99                        <p className="text-slate-500">
100                         {event.location}
```

```
101                       </p>
102                     </div>
103                   </div>
104                   <a
105                     href="#"
106                     className="group w-full flex items-center justify-center px-6
    py-4 bg-blue-600 text-white font-bold text-lg rounded-lg hover:bg-blue-700
    hover:shadow-lg focus:outline-none focus:ring-4 focus:ring-blue-300
    transition-all duration-300 transform hover:-translate-y-1"
107                   >
108                     Register for Event
109                     <FiArrowRight className="ml-2 transition-transform
    duration-300 group-hover:translate-x-1" />
110                   </a>
111                 </div>
112               </div>
113             </div>
114           </div>
115         </div>
116       </div>
117     </div>
118   );
119 };
120
121 export default EventDetail;
```

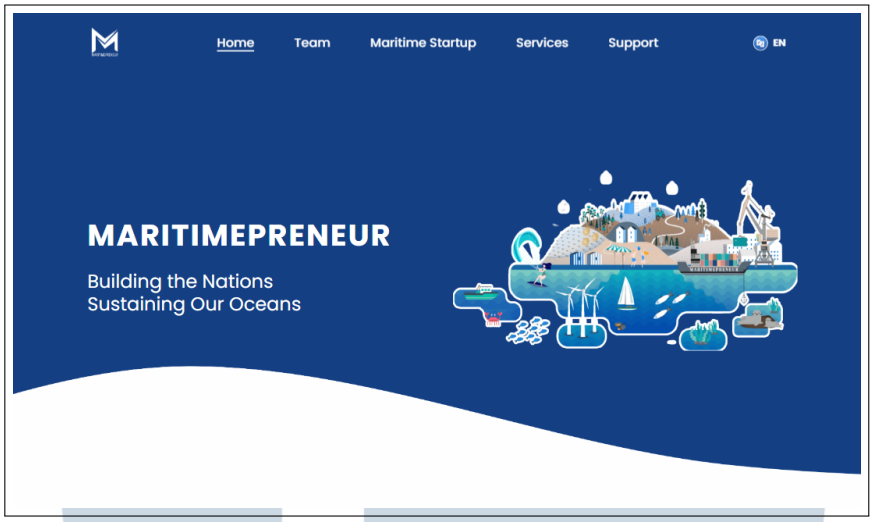
Kode 3.5: Komponen *Event Detail*

Kode 3.5 merupakan *source code* dari komponen `EventDetail`.

B Halaman Home

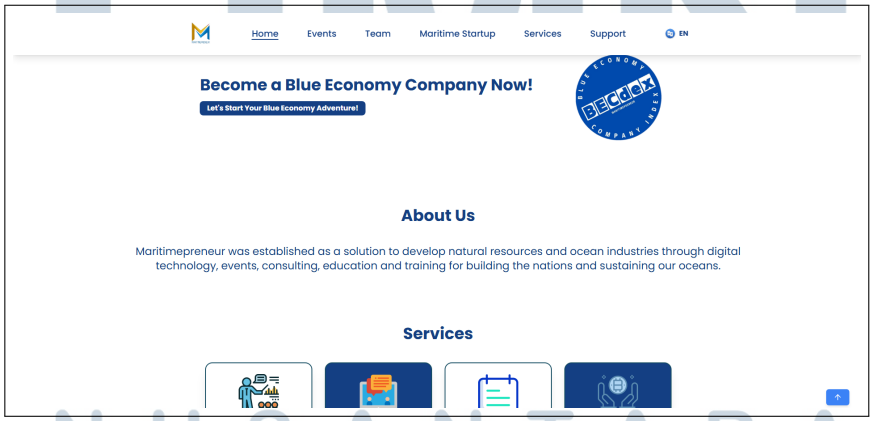
Pada halaman Home, dilakukan beberapa pembaruan dan penyesuaian, antara lain:

- Penyesuaian dan penyelarasan tata letak pada bagian *header*.
- Penyesuaian warna dan ukuran teks pada konten halaman Home.
- Pembaruan elemen-elemen desain pada halaman.



Gambar 3.12. Komponen *Header* Halaman Home

Pada bagian *header* yang ditampilkan pada Gambar 3.12, telah dilakukan serangkaian penyesuaian guna meningkatkan tampilan visual serta keterbacaan informasi yang disajikan. Penyesuaian tersebut meliputi pengaturan ulang ukuran dan posisi teks agar lebih proporsional serta selaras dengan keseluruhan desain halaman. Selain itu, ilustrasi yang terletak di sebelah kanan teks juga disesuaikan, baik dari segi ukuran maupun posisinya, guna memastikan keselarasan dengan elemen teks dan menjaga keseimbangan komposisi visual pada bagian *header*. Penyesuaian ini diterapkan pada seluruh komponen *header* di setiap halaman situs maritimpreneur.com.

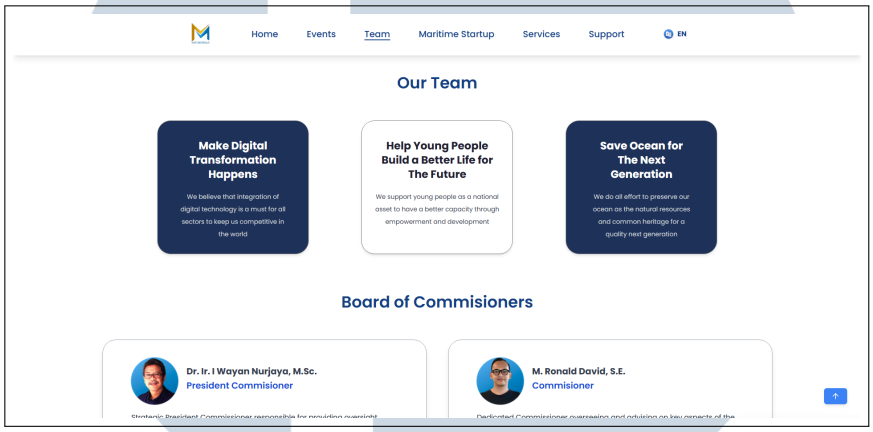


Gambar 3.13. Konten Halaman Home

Ilustrasi pada Gambar 3.13 menunjukkan modifikasi menyeluruh terhadap aspek tipografis, termasuk pemilihan dan penyesuaian jenis huruf, variasi warna,

serta penebalan huruf yang digunakan. Selain itu, dilakukan pula berbagai penyesuaian minor lainnya, seperti pengaturan warna pada tepi atau bingkai dari elemen *card* untuk memperkuat kesan visual yang konsisten dan profesional.

C Halaman Team



Gambar 3.14. Konten Halaman Team

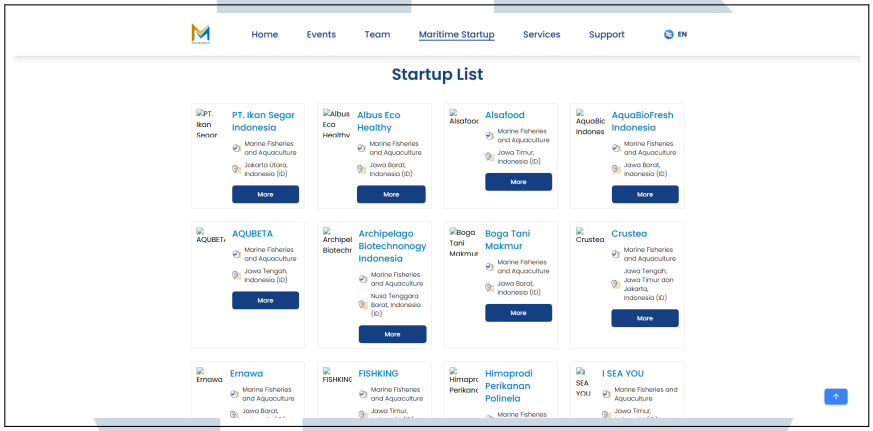
Seperti halnya pada halaman-halaman sebelumnya, halaman Team juga mengalami sejumlah perubahan dan penyesuaian guna meningkatkan kualitas tampilan serta konsistensi desain antarkomponen. Perubahan yang dilakukan antara lain mencakup penyesuaian pada komponen *header*, agar selaras dengan gaya visual yang diterapkan di seluruh situs.

Selain itu, dilakukan optimalisasi pada aspek tipografi, meliputi pemilihan *font* (jenis huruf), ukuran huruf, serta pengaturan spasi antarbaris dan antarparagraf. Langkah ini bertujuan untuk meningkatkan keterbacaan konten serta menciptakan tampilan yang lebih bersih dan terstruktur. Konsistensi tipografi memainkan peran penting dalam menjaga identitas visual situs secara keseluruhan, sehingga pengguna dapat dengan mudah memahami struktur dan isi dari setiap halaman yang ditampilkan.

D Halaman Maritime Startup

Perubahan yang dilakukan pada halaman Maritime Startup tidak jauh berbeda dengan halaman-halaman sebelumnya. Selain penyesuaian pada bagian *header*, dilakukan pula penyesuaian desain pada tampilan *card* yang

menampilkan informasi mengenai masing-masing *startup*. Penyesuaian ini mencakup penempatan logo *startup* yang kini disusun bersebelahan dengan judul dan informasi tambahan lainnya, guna menciptakan tata letak yang lebih ringkas dan informatif.



Gambar 3.15. Konten Halaman Maritime Startup

Seluruh data ditampilkan secara statis, tanpa melibatkan penggunaan API maupun integrasi dengan basis data (*database*). Pendekatan ini dipilih untuk menyederhanakan proses pengelolaan konten.

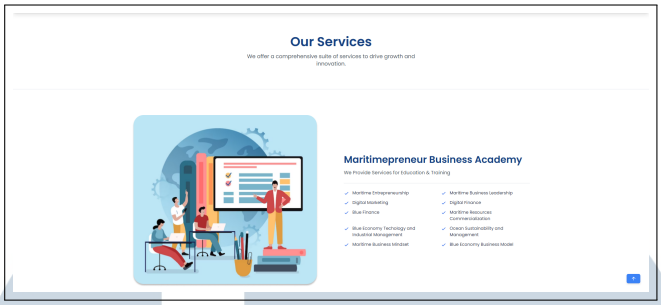
```
const getInitialData = () => [
  {
    id: 1,
    name: "PT. Ikan Segar Indonesia",
    description: "desc_pt_ikan_segar_indonesia",
    company: "2023-08-24",
    sector: "Marine Fisheries and Aquaculture",
    location: "Jakarta Utara, Indonesia (ID)",
    category: "Standard Blue Economy Company",
    address:
      "Gedung Pasar Ikan Modern Muara Baru (PIM), Jakarta. (Operasional). Kirana Tower II, Level 10-A, JL. Boulevard Timur No. 88 Kelapa Gading, Jakarta Utara.
    ",
    logo: "https://i.postimg.cc/YSFP9gnc/isi.png",
    certificateStatus: "certificateStatus",
    link: "",
  },
  {
    id: 2,
    name: "Albus Eco Healthy",
    description: "desc_albus_eco_healthy",
    company: "PT Extra Inovasi Indonesia",
    sector: "Marine Fisheries and Aquaculture",
    location: "Jawa Barat, Indonesia (ID)",
    category: "Standard Blue Economy Company",
```

```
    address: "",
    logo: "https://i.postimg.cc/4yxHdjwn/1.png",
    certificateStatus: "certificateStatus",
    link: "",
  },
  {
    id: 3,
    name: "Alsafood",
    description: "desc_alsafood",
    company: "CV Alsafood",
    sector: "Marine Fisheries and Aquaculture",
    location: "Jawa Timur, Indonesia (ID)",
    category: "Standard Blue Economy Company",
    address: "",
    logo: "https://i.postimg.cc/8P3fpKvP/2.png",
    certificateStatus: "certificateStatus",
    link: "https://www.instagram.com/abonikan_alsafood",
  },
    ...
```

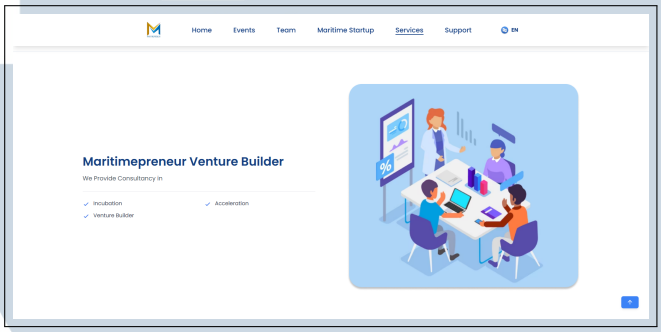
Kode 3.6: Data *Startup* Statis

E Halaman Services

Desain dari halaman Services telah mengalami perubahan yang cukup signifikan dibandingkan dengan versi awal. Penyesuaian ini bertujuan untuk meningkatkan kualitas tampilan visual serta memberikan pengalaman pengguna yang lebih nyaman dan informatif saat mengakses berbagai layanan yang ditawarkan. Perubahan mencakup penyusunan ulang tata letak konten agar lebih terstruktur, penggunaan ilustrasi visual yang relevan untuk setiap layanan, serta penyesuaian tipografi dan warna yang konsisten dengan keseluruhan identitas visual situs. Dengan demikian, halaman Services kini mampu menyampaikan informasi secara lebih efektif sekaligus memperkuat kesan profesional dari platform maritimpreneur.com.



(a)



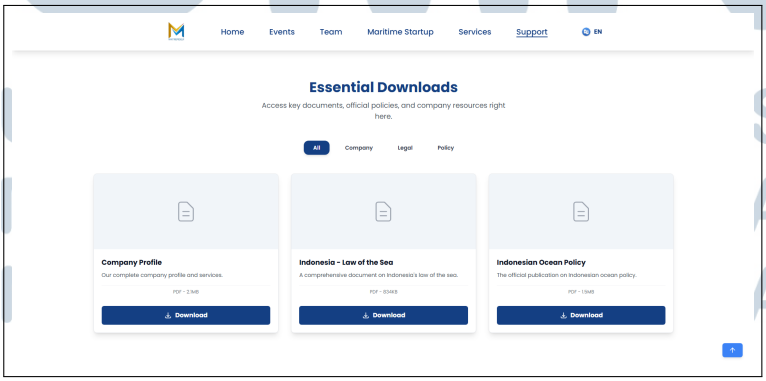
(b)

Gambar 3.16. Konten Halaman Services

Seperti yang ditampilkan pada Gambar 3.16, implementasi terbaru dari halaman Services menunjukkan adanya pembaruan dari sisi tata letak, tipografi, jarak antar elemen, serta efek visual pada komponen kartu layanan.

F Halaman Support

Halaman Support juga telah mengalami perubahan yang signifikan dari rancangan desain awalnya.



Gambar 3.17. Konten Halaman Support

Seperti yang terlihat pada Gambar 3.17, halaman Support telah mengalami pembaruan yang mencakup penyesuaian pada tata letak, tipografi, dan elemen desain lainnya. Penyesuaian ini bertujuan untuk meningkatkan keterbacaan, estetika, dan konsistensi visual di seluruh halaman.

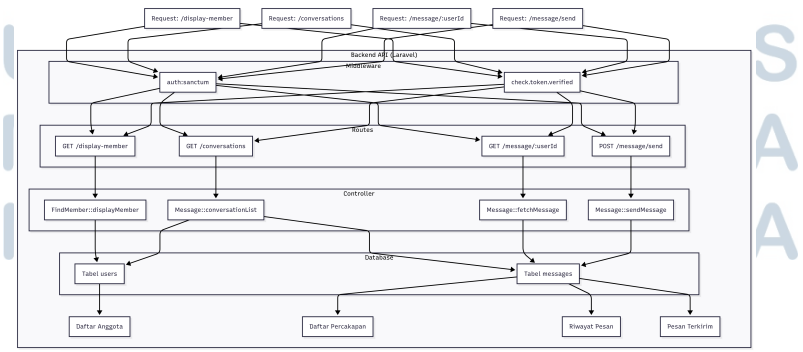
3.3.5 *Cloning* Repositori Hub.Maritimmuda.id

Pada minggu ke-14, dilakukan proses *cloning* repositori dari proyek situs hub.maritimmuda yang tersedia di GitHub. Tujuan dari kegiatan ini adalah untuk memperoleh salinan lokal dari kode sumber proyek yang nantinya akan digunakan sebagai dasar dalam pengembangan, analisis struktur sistem, serta perancangan *Application Programming Interface* (API) tambahan yang dibutuhkan oleh platform.

Repositori dari situs hub.maritimmuda.id dapat diakses melalui URL berikut: https://github.com/maritimmuda-id/hub.maritimmuda.id.git

A *Flow Chart* API *Chat* Hub.maritimmuda.id

Pada Gambar 3.18 diilustrasikan alur kerja dari sisi server (*backend*) yang bertanggung jawab menangani seluruh permintaan yang dikirimkan oleh aplikasi klien. Sistem *backend* dibangun menggunakan *framework* Laravel yang menerapkan pola arsitektur *Model-View-Controller* (MVC) untuk memisahkan logika, data, dan tampilan [28]. Komponen pertama adalah *middleware*, yang mengimplementasikan pola desain *Chain of Responsibility* untuk memastikan semua permintaan privat melewati lapisan keamanan, seperti autentikasi berbasis token melalui auth:sanctum [29], [30]. Komponen kedua adalah *controller*, yang memuat logika untuk setiap permintaan (*request*).



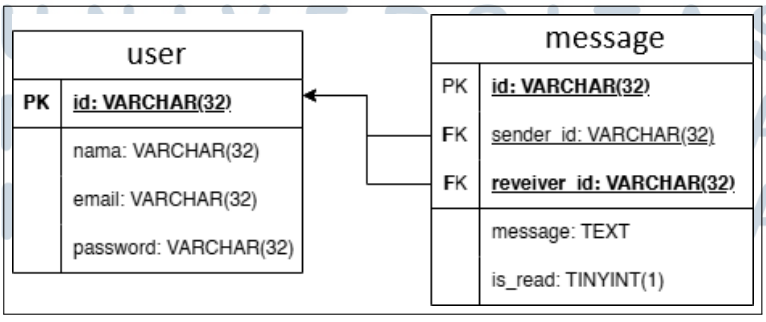
Gambar 3.18. Diagram Alur Proses *Backend*.

Interaksi antara klien dan server ini dirancang mengikuti gaya arsitektur REST (Representational State Transfer), yang memanfaatkan metode-metode standar dari protokol HTTP [31]. Saat pengguna mencari anggota lain, aplikasi akan mengirimkan permintaan `GET` ke *endpoint* `/api/display-member`. Metode `GET` bersifat idempoten dan aman, artinya tidak mengubah data di server. Permintaan ini diterima oleh `FindMemberController`, yang melakukan *query* ke tabel `users`. Server kemudian mengembalikan respons berupa data dalam format JSON (*JavaScript Object Notation*), sebuah standar format pertukaran data yang ringan dan mudah dibaca [32].

Selanjutnya, ketika pengguna ingin melihat riwayat percakapan, aplikasi akan mengirimkan permintaan `GET` ke *endpoint* `/api/conversations`. Permintaan ini diproses oleh `MessageController` yang mengambil data dari beberapa tabel, kemudian mengelompokkannya berdasarkan percakapan sebelum mengembalikannya dalam format JSON. Pembacaan isi pesan dari salah satu percakapan dilakukan dengan mengirimkan permintaan `GET` ke *endpoint* `/api/message/{userId}`.`MessageController` pada proses ini memperbarui status pesan menjadi telah dibaca. Setelah pembaruan, riwayat pesan diambil dan dikirimkan sebagai respons JSON.

Proses pengiriman pesan dilakukan ketika aplikasi mengirimkan permintaan `POST` ke *endpoint* `/api/message/send`, sesuai dengan peran metode `POST` untuk membuat sumber daya baru. `MessageController` akan melakukan validasi terhadap data yang diterima, kemudian menyimpan pesan baru ke dalam tabel `messages`. Server selanjutnya memberikan respons berupa data pesan yang baru dibuat dalam format JSON sebagai konfirmasi.

B Skema *Database*



Gambar 3.19. Rancangan Skema *Database* API *Chat*

Pada Gambar 3.19 ditampilkan skema basis data relasional (*relational database schema*) yang digunakan dalam implementasi API *chat*. Desain ini menggunakan dua tabel utama untuk mendukung fungsionalitas inti, yaitu tabel `user` dan tabel `message`.

Tabel `message` memiliki dua atribut yang berperan sebagai *foreign key*, yaitu `sender_id` dan `receiver_id`. Kedua atribut ini membentuk dua relasi terpisah berjenis satu-ke-banyak (*one-to-many*) dari tabel `user` ke tabel `message`. Struktur ini secara fundamental penting karena menegakkan integritas referensial (*referential integrity*), yang memastikan bahwa setiap nilai pada kolom `sender_id` dan `receiver_id` harus merujuk pada `id` pengguna yang valid di tabel `user` [33]. Relasi ini tidak hanya memungkinkan identifikasi pengirim dan penerima, tetapi juga memfasilitasi pengambilan data riwayat percakapan secara efisien melalui operasi `JOIN` dalam *query* SQL.

3.3.6 Implementasi Hasil Rancangan API

```php
    <?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateMessagesTable extends Migration
{
    public function up()
    {
        Schema::create('messages', function (Blueprint $table) {
            $table->id();
            $table->foreignId('sender_id')->constrained('users')->onDelete('cascade')
    ;
            $table->foreignId('receiver_id')->constrained('users')->onDelete('cascade
    ');
            $table->text('message');
            $table->boolean('is_read')->default(false);
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('messages');
    }
}
```

Kode 3.7: Tabel *Migration* Messages

Kode 3.7 merupakan kode migrasi yang digunakan untuk membuat tabel *messages* pada basis data dalam rangka mendukung integrasi API *chat*. Tabel ini memiliki dua *foreign key*, yaitu `sender_id` dan `receiver_id`, yang masing-masing berelasi dengan tabel `users` guna mengidentifikasi pengirim dan penerima pesan dalam sistem.

Selain itu, atribut `message` berfungsi untuk menyimpan isi pesan yang dikirimkan oleh pengguna, sedangkan atribut `is_read` digunakan sebagai indikator status pesan—apakah pesan tersebut telah dibaca oleh penerima atau belum. Nilai awal dari `is_read` diatur sebagai `false` (belum dibaca). Dengan struktur ini, sistem dapat mencatat, mengelola, dan melacak setiap interaksi komunikasi antar pengguna secara efisien.

```php
<?php

namespace App\Models;

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

class Message extends Model
{
    protected $fillable = [
        'sender_id',
        'receiver_id',
        'message',
        'is_read',
    ];

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

    public function receiver() {
        return $this->belongsTo(User::class, 'receiver_id');
    }
}
```

Kode 3.8: Model dari *Message*

Kode 3.8 merupakan model `Message` pada Laravel yang merepresentasikan tabel `messages` dalam basis data. Model ini digunakan dalam proses integrasi API *chat* pada sistem. Pada atribut `$fillable`, didefinisikan kolom-kolom yang dapat diisi secara massal, yaitu `sender_id`, `receiver_id`, `message`, dan `is_read`. Pendekatan ini memudahkan proses penyimpanan data pesan ke dalam basis data secara otomatis dan efisien.

Model ini juga memiliki dua relasi `belongsTo` ke model `User`. Fungsi `sender()` berfungsi untuk mengakses data pengguna yang mengirim pesan, sedangkan `receiver()` digunakan untuk mengakses data pengguna yang menerima pesan. Relasi ini sangat penting dalam menampilkan informasi lengkap pengguna terkait pesan yang dikirimkan dalam sistem percakapan.

```
    ...
Route::middleware(['auth:sanctum', 'check.token.verified'])->group(function () {
    Route::get('/dashboard', [ViewDashboardController::class, 'apiDashboard']);

    // Mobile message api
    Route::post('/message/send', [MessageController::class, 'sendMessage']);
    Route::get('/message/{userId}', [MessageController::class, 'fetchMessage']);
    Route::get('/conversations', [MessageController::class, 'conversationList']);

    ...

    Route::get('display-member', [FindMemberController::class, 'displayMember']);
```

Kode 3.9: *Route* API *chat*

Kode 3.9 merupakan daftar *route* atau *endpoint* URL yang dapat diakses oleh aplikasi mobile untuk mengonsumsi layanan API terkait fitur *chat*. Seluruh *route* dibungkus dalam `middleware` `auth:sanctum` dan `check.token.verified` yang bertugas memastikan bahwa hanya pengguna yang telah terautentikasi dan memiliki token yang valid yang dapat mengaksesnya.

Adapun rincian *route* yang tersedia adalah sebagai berikut:

1. POST /message/send: Mengirim pesan baru ke pengguna lain melalui fungsi `sendMessage` pada `MessageController`.
2. GET /message/{userId}: Mengambil seluruh riwayat pesan antara pengguna saat ini dan pengguna lain dengan ID tertentu, menggunakan fungsi `fetchMessage`.
3. GET `/conversations`: Mengambil daftar percakapan beserta pesan terakhir dari setiap percakapan melalui fungsi `conversationList`.
4. GET /display-member: Menampilkan daftar anggota yang terdapat pada *database* Maritim Muda Nusantara melalui fungsi `displayMember` pada `FindMemberController`.

Semua *endpoint* ini dirancang untuk mendukung komunikasi dua arah antar pengguna di dalam aplikasi dan dilindungi oleh mekanisme autentikasi demi menjaga keamanan data percakapan.

```
    public function sendMessage(Request $request)
    {
        $request->validate([
            'receiver_id' => 'required|exists:users,id',
            'message' => 'required|string',
        ]);

        $message = Message::create([
            'sender_id' => auth()->id(),
            'receiver_id' => $request->receiver_id,
            'message' => $request->message,
        ]);

        return response()->json([
            'message' => 'Message sent',
            'data' => $message,
        ]);
    }
```

Kode 3.10: Fungsi `sendMessage()`

Kode 3.10 merupakan implementasi dari fungsi `sendMessage` yang bertugas untuk menangani proses pengiriman pesan antar pengguna dalam API. Fungsi ini termasuk dalam `MessageController` dan dapat diakses melalui endpoint `POST /api/message/send`.

Alur kerja fungsi:

1. Validasi Request: Sistem memverifikasi bahwa `receiver_id` wajib diisi dan harus ada di tabel `users`, serta `message` harus berupa teks.
2. Pembuatan Pesan: Setelah validasi berhasil, sistem akan membuat data pesan baru dengan menyimpan `sender_id` sebagai ID pengguna yang sedang login, `receiver_id` dari permintaan, dan isi pesan.
3. Respons JSON: Sistem memberikan respons dalam format JSON yang berisi konfirmasi bahwa pesan berhasil dikirim serta data pesan yang baru dibuat.

```
    public function fetchMessage($userId)
    {
        Message::where('sender_id', $userId)
            ->where('receiver_id', auth()->id())
```

```
            ->where('is_read', false)
            ->update(['is_read' => true]);

        $messages = Message::where(function ($query) use ($userId) {
            $query->where('sender_id', auth()->id())
                ->where('receiver_id', $userId);
        })->orWhere(function ($query) use ($userId) {
            $query->where('sender_id', $userId)
                ->where('receiver_id', auth()->id());
        })->orderBy('created_at')->get();

        return response()->json($messages);
    }
```

Kode 3.11: Fungsi `fetchMessage()`

Kode 3.11 merupakan implementasi fungsi `fetchMessage` yang digunakan untuk mengambil seluruh riwayat percakapan antara pengguna yang sedang login dengan pengguna lain berdasarkan `userId` yang dikirim dari permintaan API. Fungsi ini juga otomatis memperbarui status pesan menjadi telah dibaca.

Berikut penjelasan alur dari fungsi tersebut:

1. Menandai Pesan sebagai Terbaca: Sistem akan mencari pesan yang dikirim oleh pengguna dengan `userId` tujuan ke pengguna yang sedang login dan memiliki status `is_read` bernilai `false`. Jika ditemukan, status pesan tersebut diubah menjadi `true`.
2. Mengambil Riwayat Pesan: Sistem kemudian mengambil seluruh pesan yang saling dipertukarkan antara pengguna yang login dan pengguna dengan `userId` tersebut, baik sebagai pengirim maupun penerima.
3. Pengurutan Berdasarkan Waktu: Data pesan diurutkan berdasarkan waktu pembuatan (`created_at`) untuk memastikan tampilan percakapan yang kronologis.
4. Mengembalikan Respons: Data dikembalikan dalam bentuk JSON yang dapat ditampilkan oleh aplikasi mobile.

```
    public function conversationList()
    {
        $userId = auth()->id();

        $messages = Message::where(function ($query) use ($userId) {
            $query->where('sender_id', $userId)
```

```
                ->orWhere('receiver_id', $userId);
        })
            ->orderBy('created_at', 'desc')
            ->get()
            ->groupBy(function ($item) use ($userId) {
                return $item->sender_id == $userId ? $item->receiver_id : $item->
    sender_id;
            });

        $result = [];

        foreach ($messages as $partnerId => $msgs) {
            $result[] = [
                'user' => User::find($partnerId),
                'last_message' => $msgs->first(),
            ];
        }

        return response()->json($result);
    }
```

Kode 3.12: Fungsi `conversationList()`

Kode 3.12 merupakan fungsi `conversationList` yang digunakan untuk menampilkan daftar percakapan pengguna. Sistem akan mencari semua pesan yang melibatkan pengguna yang sedang login, baik sebagai pengirim maupun penerima. Setelah itu, pesan-pesan tersebut dikelompokkan berdasarkan pasangan percakapan (pengguna lain). Untuk setiap pasangan, sistem akan mengambil informasi pengguna dan pesan terakhir dari percakapan tersebut. Hasil akhirnya berupa daftar percakapan yang mencakup pengguna yang terlibat dan pesan terakhir, lalu dikembalikan dalam format JSON kepada aplikasi mobile.

```
        Route::get('display-member', [FindMemberController::class, 'displayMember'])
    ;
```

Kode 3.13: *Endpoint Display Member*

Pada Kode 3.13, *endpoint* `GET /api/display-member` digunakan untuk menampilkan daftar anggota berdasarkan data yang ada pada *database* Maritim Muda Nusantara.

```
    public function displayMember(Request $request): JsonResponse
    {
        $query = User::query();

        $hasSearch = $request->has('search') ^ !empty($request->search);

        if ($hasSearch) {
```

```
            $search = $request->search;

            $query->where(function ($q) use ($search) {
                $q->where('name', 'LIKE', "%{$search}%")
                    ->orWhere('serial_number', 'LIKE', "%{$search}%")
                    ->orWhere('email', 'LIKE', "%{$search}%");
            });
        }

        $query->limit(50);

        $members = $query->select([
            'id',
            'uuid',
            'uid',
            'serial_number',
            'name',
            'locale',
            'province_id',
            'email',
        ])->get();

        return response()->json([
            'success' => true,
            'members' => $members,
            'count' => $members->count()
        ]);
    }
```

Kode 3.14: Fungsi `displayMember()`

Kode 3.14 menampilkan logika pencarian anggota berdasarkan parameter `search` yang dikirim melalui permintaan HTTP. Parameter ini dapat berupa nama (`name`), nomor seri (`serial_number`), atau alamat surel (`email`). Jika parameter pencarian disertakan, maka sistem akan melakukan pencocokan data menggunakan operator `LIKE` terhadap ketiga *field* tersebut.

Terdapat dua skenario utama dalam proses pencarian anggota berdasarkan parameter `search`, yaitu:

1. Tanpa Parameter Pencarian

   Jika parameter `search` tidak disediakan atau kosong, maka sistem akan mengambil hingga 50 data anggota secara default tanpa filter pencarian.

2. Dengan Parameter Pencarian

   Jika parameter `search` diberikan, maka sistem hanya akan menampilkan anggota yang memiliki nama, nomor seri, atau email yang mengandung nilai dari parameter tersebut.

Hasil dari *endpoint* ini dikembalikan dalam bentuk respons JSON yang berisi:

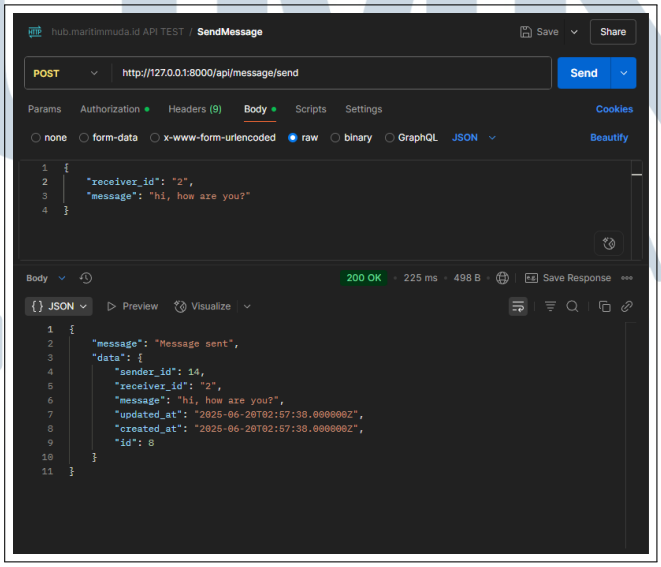
- `success` – status permintaan
- `members` – daftar anggota yang ditemukan
- `count` – jumlah data anggota yang ditemukan

3.3.7 Uji Coba *Application Programming Interface* (API) *Chat*

Tahap uji coba ini merupakan bagian dari fase pengujian fungsional (*functional testing*), yang dilakukan dengan menggunakan perangkat lunak Postman. Postman berperan sebagai klien HTTP untuk melakukan simulasi permintaan dan penerimaan respons dari setiap *endpoint* API. Tujuan utama dari pengujian ini adalah untuk melakukan validasi, yaitu memastikan bahwa setiap *endpoint* yang dikembangkan telah berfungsi sesuai dengan spesifikasi dan kebutuhan bisnis dari fitur *chat* [34]. Proses ini secara efektif menguji setiap unit fungsional dari API sebelum diintegrasikan dengan aplikasi klien.

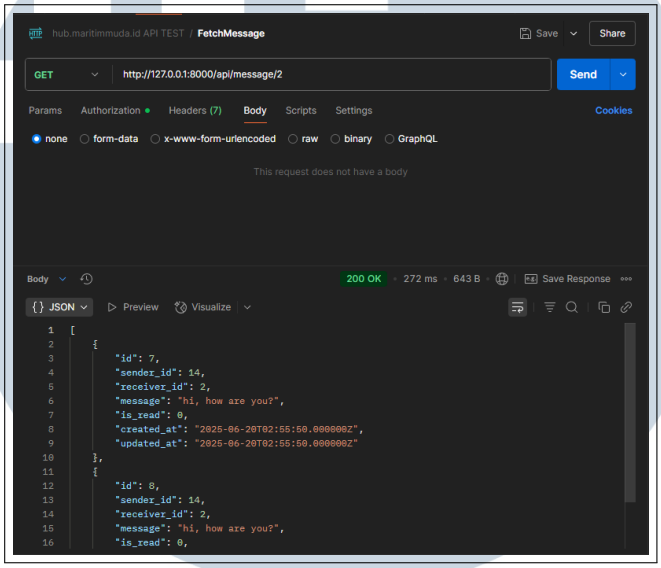
Pengujian dilakukan pada beberapa *endpoint* utama berikut:

1. *Endpoint* POST /api/message/send *Endpoint* ini digunakan untuk mengirimkan pesan dari satu pengguna kepada pengguna lainnya. Uji coba berhasil dilakukan dengan mengirimkan *body* yang berisi parameter `receiver_id` dan `message`, seperti yang ditampilkan pada Gambar 3.20.



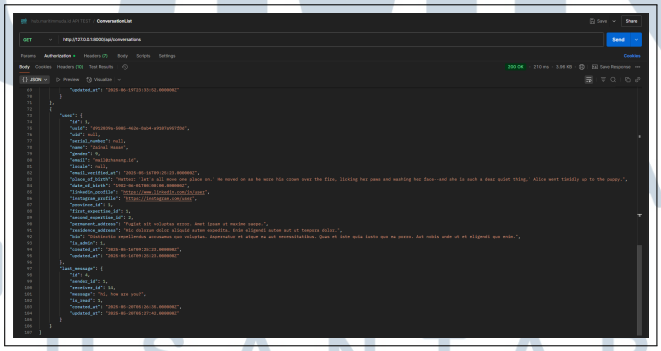
Gambar 3.20. Postman: http://127.0.0.1:8000/api/message/send

2. *Endpoint* GET /api/message/{userId} *Endpoint* ini digunakan untuk mengambil seluruh riwayat percakapan antara pengguna yang sedang masuk dengan pengguna lain berdasarkan `userId`. Hasil pengujian pada Gambar 3.21 menampilkan daftar pesan secara berurutan berdasarkan waktu pengiriman.



Gambar 3.21. Postman: http://127.0.0.1:8000/api/message/2

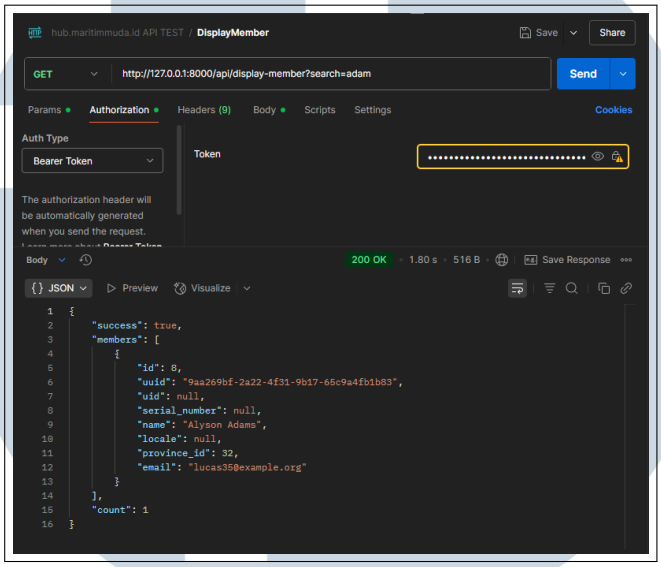
3. *Endpoint* GET /api/conversations *Endpoint* ini menampilkan daftar percakapan pengguna beserta pesan terakhir dari masing-masing percakapan. Uji coba yang terlihat pada Gambar 3.22 menunjukkan bahwa sistem berhasil mengelompokkan pesan berdasarkan pengguna lain yang terlibat.



Gambar 3.22. Postman: http://127.0.0.1:8000/api/conversations

4. *Endpoint* GET /api/display-member *Endpoint* ini digunakan untuk menampilkan hasil pencarian anggota. Parameter URL

`?search={username/email/serial_number}` dapat ditambahkan untuk mencari data anggota tertentu dari basis data Maritim Muda. Apabila parameter `?search={}` tidak digunakan, maka *endpoint* ini akan menampilkan 50 anggota secara acak.



Gambar 3.23. Postman: http://127.0.0.1:8000/api/display-member

Seluruh pengujian menunjukkan bahwa API chat dapat berfungsi dengan baik dan merespon sesuai dengan logika yang telah diimplementasikan.

3.4 Kendala dan Solusi yang Ditemukan

Selama pelaksanaan kegiatan magang, terdapat beberapa kendala yang dihadapi. Berikut ini adalah beberapa permasalahan yang ditemukan beserta solusi yang dilakukan untuk mengatasinya:

1. Kurangnya Kejelasan dalam Pembagian Tugas pada Awal Magang. Pada tahap awal pelaksanaan magang, pembagian tugas belum tersampaikan secara rinci dan menyeluruh, sehingga menimbulkan kebingungan dalam pelaksanaan pekerjaan.

   Solusi: Melakukan komunikasi lebih intensif dengan mentor melalui pesan pribadi maupun diskusi tim untuk mendapatkan arahan yang lebih jelas terkait tugas yang sedang dan akan dikerjakan.

2. Alur kerja sama antaranggota tim masih belum optimal. Hal ini dipengaruhi oleh pembagian tugas yang belum terstruktur dengan baik, sehingga sulit untuk menjalankan kolaborasi secara efektif.

   Solusi: Mengadakan diskusi tim secara berkala untuk memperjelas tanggung jawab masing-masing anggota serta mendorong inisiatif komunikasi antaranggota tim dalam menyelesaikan tugas bersama.