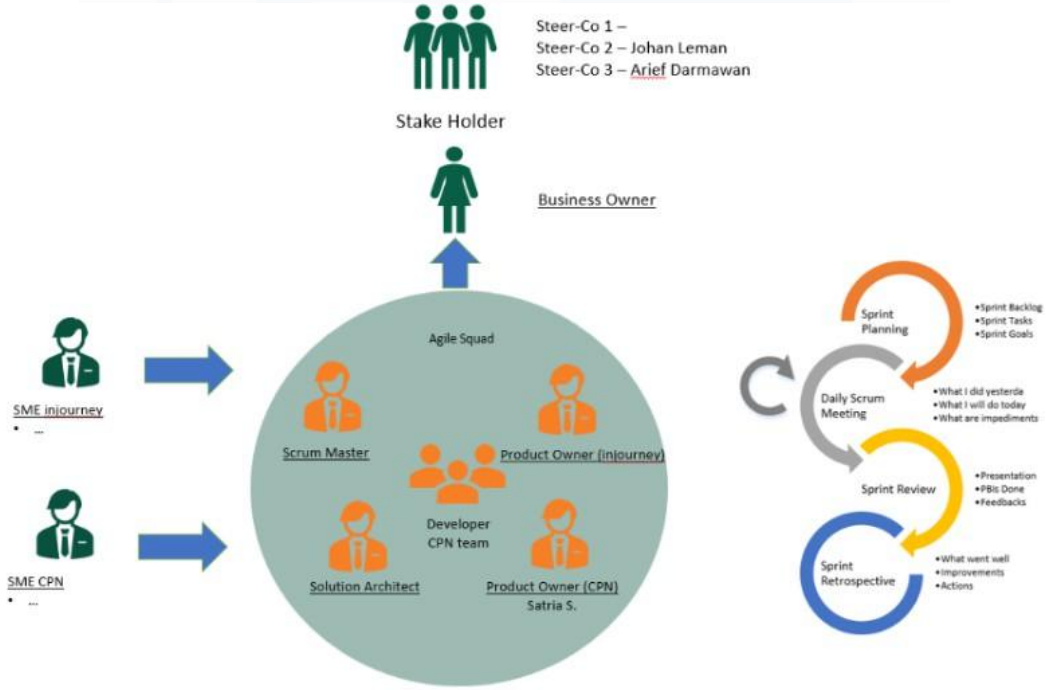
BAB III
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi



Gambar 3.1 Kedudukan & Koordinasi Perusahaan

Selama menjalani kegiatan magang di PT. Cipta Prima Nugraha (CPN), saya ditempatkan dalam satu tim kerja yang tergabung dalam Agile Squad sebagai bagian dari tim Developer Front-End. Dalam struktur organisasi proyek pengembangan website Bandara Internasional Soekarno-Hatta, posisi mahasiswa magang memiliki peran fungsional sebagai pelaksana teknis yang berkolaborasi erat dengan berbagai pihak internal perusahaan maupun perwakilan pengguna (user).

Tim Agile ini terdiri atas berbagai peran yang saling terkait dan berkoordinasi secara intensif melalui metode kerja agile dan scrum. Struktur koordinasi kerja dibagi menjadi beberapa lapisan yang memiliki peran dan tanggung jawab sebagai berikut:

Tabel 3. 1 Tabel Agile Organization

Nama	**Jabatan**	**Peran**
-	Steering Committee 1	Decision Maker.
-	Steering Committee 2	Decision Maker.
-	Steering Committee 3	Decision Maker.
-	Business Owner	Pemilik Flow dari Aplikasi, jika terjadi perubahan flow harus melalui persetujuan dari member yang memerankan ini.
-	Product Owner (CPN)	Pemilik Technical dari Aplikasi, di dalamnya terdapat penjagaan secara teknis, fitur, dan semua komponen teknis dalam produk.
-	Scrum Master	Pemilik Backlog dan Sprint Planning, mengelola semua backlog masing-masing sprint.
-	Solution Architect	Penanggung jawab arsitektur teknis aplikasi
-	SME Branch Communication	Mewakili user yang akan bertindak sebagai konsultan

Nama	Jabatan	Peran
		dengan perspektif dari sisi user yang menggunakan. User ini sebagai narasumber Ketika proses user requirement dilakukan.
-	SME Branch Quality Control	Mewakili user yang akan bertindak sebagai konsultan dengan perspektif dari sisi user yang menggunakan. Sebagai narasumber Ketika proses user requirement dilakukan.
-	SME Technical	Narasumber Ketika proses user requirement dilakukan.
-	Business Analyst	Sebagai wartawan dalam melakukan interview terhadap SME. Outputnya adalah blueprint dari Aplikasi yang akan dibangun.
-	Developer Team	Eksekutor dari sisi coding yang akan membuat aplikasi dari Blueprint Business Analyst. Terdiri dari Developer Front End, Back End, dan Database.

Nama	Jabatan	Peran
Mahasiswa Magang (Michael Leman)	Developer Front-End (Part of Developer Team)	Eksekutor implementasi antarmuka.

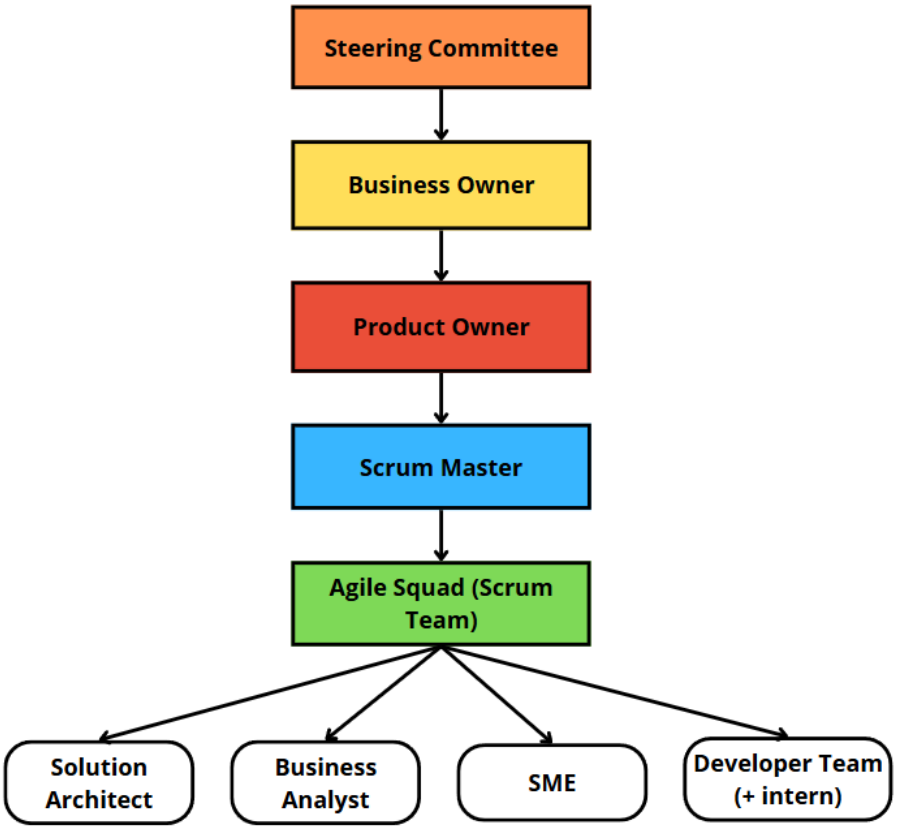
Tabel ini menampilkan struktur organisasi tim proyek dalam kerangka kerja Agile. Setiap peran memiliki fungsi spesifik dalam siklus Scrum untuk mendukung pengembangan CMS.

Mahasiswa magang mengikuti siklus pengembangan produk yang berjalan dalam kerangka kerja Scrum, yang terdiri dari:

- Sprint Planning: Menentukan backlog dan sasaran sprint.
- Daily Scrum: Update harian mengenai progres, hambatan, dan rencana kerja.
- Sprint Review: Menampilkan hasil pekerjaan kepada stakeholder dan menerima umpan balik.
- Sprint Retrospective: Evaluasi kerja sprint sebelumnya dan merumuskan perbaikan.

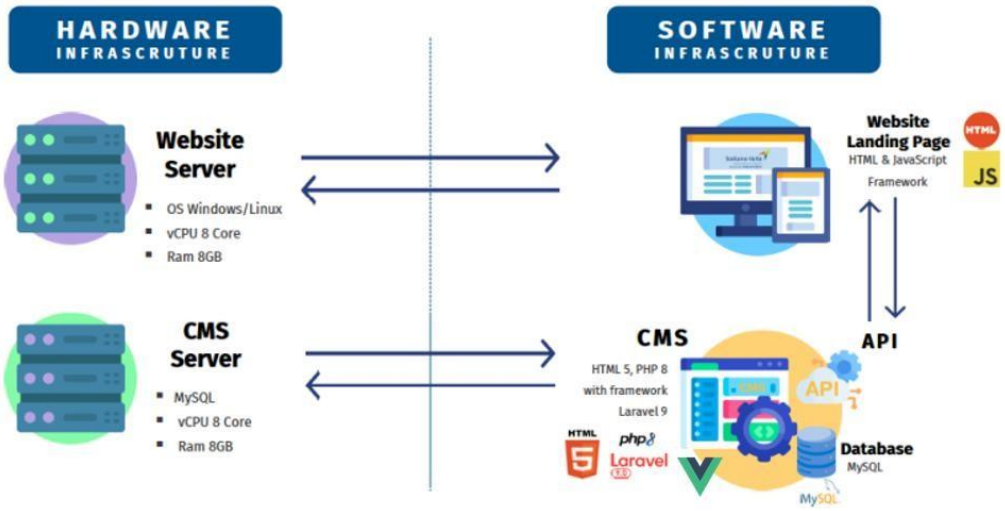
Setiap pengembangan dimulai dari hasil diskusi antara Business Analyst dan SME, lalu diterjemahkan menjadi Blueprint UI, yang kemudian diberikan kepada tim developer. Mahasiswa magang sebagai bagian dari tim developer, bertanggung jawab melakukan slicing desain, implementasi HTML/CSS/JS, dan memastikan responsivitas antarmuka sesuai dengan standar brand dan kebutuhan pengguna.

Koordinasi dilakukan secara hybrid antara meeting daring melalui platform komunikasi (Zoom, Google Meet, dsb.) dan pertemuan langsung jika dibutuhkan oleh project owner atau Scrum Master.



Gambar 3.2 Diagram Alur Kedudukan Mahasiswa Magang

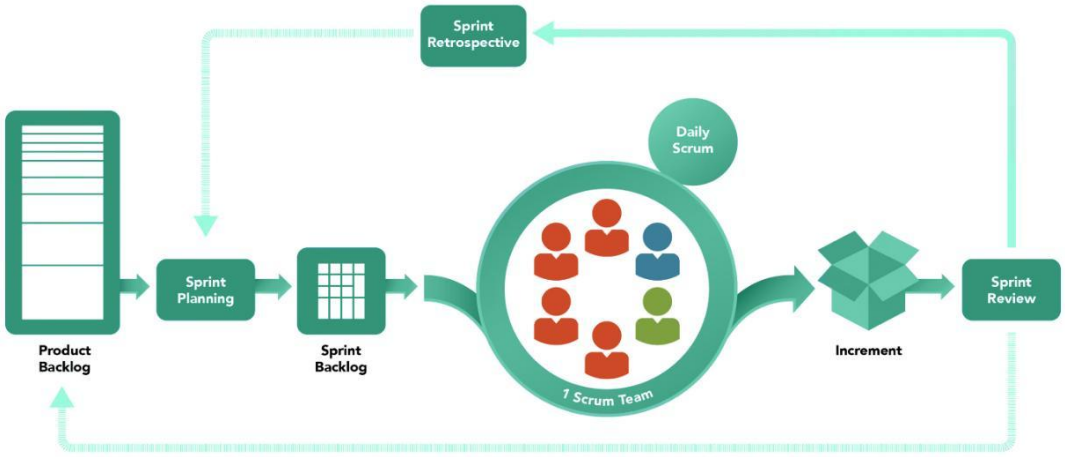
Dalam praktiknya, saya sebagai mahasiswa magang tidak hanya berfokus pada pengembangan antarmuka, tetapi juga turut serta memahami proses perencanaan, pemetaan kebutuhan user, dan evaluasi produk yang sedang dikembangkan. Ini memberikan wawasan menyeluruh tentang bagaimana sistem kerja kolaboratif antara berbagai peran dijalankan dalam lingkungan profesional berbasis Agile.



Gambar 3.3 Project Architecture

Detail Spesifikasi Aplikasi:

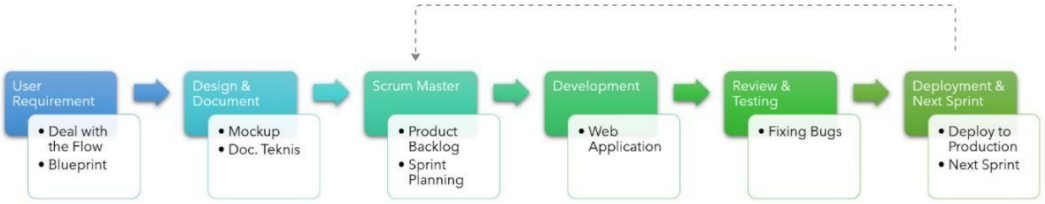
- Bahasa Pemrograman: PHP minimum 8.1
- Framework: Laravel 10.x
- Database: MySQL 8.1 / 8.2
- UI: Vue.js



Gambar 3.4 Agile Methodology

Agile Methodology adalah proses pembuatan aplikasi dengan menekankan fleksibilitas, kolaborasi antar member, publikasi secara bertahap. Di dalam Methodology memiliki iterasi yang pendek dalam proses developmentnya, untuk kemudian disebut Sprint. Di dalam Sprint memiliki beberapa step seperti:

- Product Backlog: adalah kumpulan task yang akan dikerjakan secara keseluruhan.
- Sprint Planning: Memilih beberapa task dari Product Backlog untuk kemudian dilakukan development selama dua atau tiga minggu.
- Scrum: Proses Development yang dimonitor dengan model scrum (daily / weekly) untuk identifikasi masalah agar tidak berkepanjangan.
- Increment: Setelah melakukan testing akan timbul increment untuk kemudian masuk ke dalam backlog atau bahkan langsung dikerjakan jika tidak mempengaruhi timeline.
- Sprint Review: Review dari ouput Sprint yang bersangkutan, outputnya bisa berupa backlog untuk dikerjakan di masa yang akan datang.
- Sprint Retrospective: Review terhadap semua member yang berpartisipasi dalam penyelesaian Sprint, jika ada yang kurang baik maka harus diperbaiki agar next sprint tidak terjadi lagi.
- Next Sprint: Setelah menjalani semua step di atas, maka kita kan masuk ke sprint berikutnya dan mengulangi step di atas dengan task yang berbeda.



Gambar 3.5 Development Step

1. User Requirement

User Requirement / Requirement Gathering adalah step terjadinya interview antara Business Analyst dengan SME dengan tujuan mendapatkan blueprint dan menyepakati flow dari blueprint yang sudah terbentuk.

2. Design & Document

Setelah blueprint terbentuk, maka akan di konversi menjadi mockup yang layak untuk diberikan kepada developer untuk proses development. Sebagai tambahan akan terbentuk dokumen teknis agar developer dapat mengerti secara keseluruhan.

3. Scrum Master

Setelah melakukan User Requirement dan Mockup juga sudah terbentuk, Scrum Master dapat membuat backlog product dengan lengkap, mulai deskripsinya sampai dengan tampilan mockupnya. Setelah itu perlu di adakan diskusi untuk membentuk sprint planning untuk periode 2 sampai 3 minggu pengerjaan. Sprint nya bisa berapapun sampai backlog product selesai dikerjakan semua.

4. Development

Proses development dilakukan per sprint. Developer perlu mengikuti semua backlog yang sudah dibentuk pada sprint yang bersangkutan. Jika menemui beberapa issue atau membutuhkan konfirmasi, dapat diutarakan pada daily scrum atau langsung menghubungi pihak yang berkaitan. Output dari step ini adalah web aplikasi sesuai list yang ada di dalam sprint yang bersangkutan.

5. Review & Testing

Business Analyst, Project Manager, Project Owner, dan Business Owner berkewajiban melakukan testing dan review atas apa yang sudah selesai dikerjakan oleh para developer. Output dari step ini adalah bugs atau issue yang muncul setelah testing atau increment backlog. Khusus untuk bugs harus

diselesaikan terlebih dahulu, tapi untuk increment bisa masuk ke dalam backlog dulu agar jadwal tidak mundur.

6. Deployment & Next Sprint

Jika review dan testing sudah dilakukan, maka developer akan melakukan deployment pada server yang bisa di access User SME, sehingga mereka bisa melakukan pengecekan apakah User Requirement sudah dipenuhi di dalam aplikasi. Jika sudah maka bisa masuk ke dalam Next Sprint dan mengulangi hal yang sama sampai backlog product selesai dikerjakan semua. Untuk Deployment pada server Go Live terbilang fleksibel jika menggunakan metode ini, deployment bisa dilakukan setiap akhir sprint atau menunggu semua sprint selesai baru deploy ke production server.

Tabel 3. 2 Matrik Komunikasi Pelaksanaan Projek

Tipe Komunikasi	Frekuensi	Media	Peserta	Keterangan
Daily	Harian	Online (WA, Teams, Zoom, etc)	Semua member yang membutuhkan	Jika muncul issue/butuh konfirmasi ketika proses development.
Weekly	1 Minggu Sekali	Online (WA, Teams, Zoom, etc)	Scrum Master, Project Owner, Business Owner, Business Analyst, Developer.	Output: Laporan Mingguan, MoM
Biweekly	2 Minggu Sekali	Visit/Offline	Scrum Master, Project Owner, Business Owner, Business Analyst, Developer.	Output: Laporan Mingguan, MoM

Monthly	1 Bulan Sekali	Online/Offline	Scrum Master, Project Owner, Business Owner, Business Analyst, Developer, Steering Committee	Output: Laporan Bulanan, MoM

Tabel diatas menunjukkan frekuensi, media, dan peserta dari komunikasi proyek, yang membantu menjaga sinkronisasi antar tim selama pengembangan CMS berlangsung.

3.2 Tugas dan Uraian Kerja Magang

Selama masa magang di PT. Cipta Prima Nugraha, saya ditugaskan untuk membangun dan mengembangkan empat modul utama pada CMS Website Informasi Bandara Internasional Soekarno-Hatta berbasis Vue.js. Empat modul tersebut adalah:

- Terminal
- TerminalType
- Floor
- Parking

Pekerjaan ini dilakukan dengan asumsi bahwa seluruh data yang ditampilkan sudah tersedia di database dan dapat diakses melalui API yang disediakan oleh backend developer. Oleh karena itu, tugas utama saya sebagai frontend developer adalah membangun user interface (UI) untuk setiap modul agar administrator dapat melakukan pengelolaan data dengan mudah.

Pengembangan dilakukan secara paralel antara CMS dan modul publik karena:

- Data sudah tersedia di backend.

- API dapat diakses bahkan tanpa UI selesai.
- Tidak ada dependensi antar menu.

Hal ini memungkinkan pengerjaan CMS berjalan efisien tanpa perlu menunggu sisi publik atau data dari backend selesai. Saya berfokus pada UI yang stabil, validasi form yang tepat, dan memastikan fungsi-fungsi CRUD berjalan sesuai rencana.

3.2.1 Perkenalan Organisasi, Briefing, Setting Environtment & Training Vue.js

Pada minggu awal magang, penulis mengikuti sesi perkenalan organisasi, pemahaman struktur proyek, dan pengaturan environment kerja. Setup mencakup instalasi Node.js, Vite, diberikan package.json, VS Code yang dibantu oleh atasan. Penulis juga mulai belajar kembali basic coding vue.js.

3.2.2 Modul Terminal

Modul Terminal merupakan salah satu bagian penting dalam CMS yang memungkinkan admin mengelola informasi mengenai terminal yang tersedia di Bandara Soekarno-Hatta. Proses pengerjaan dimulai dengan membangun tampilan tabel data, lalu mengembangkan form untuk penambahan, pengeditan, serta penghapusan data terminal. Fitur filter ditambahkan agar memudahkan pencarian data.

Fitur yang dikembangkan:

- Menampilkan data terminal dalam bentuk tabel.
- Filter berdasarkan kata kunci.
- Form untuk menambah dan mengedit terminal.
- Konfirmasi sebelum penghapusan data.
- Upload gambar terminal ke server MinIO.

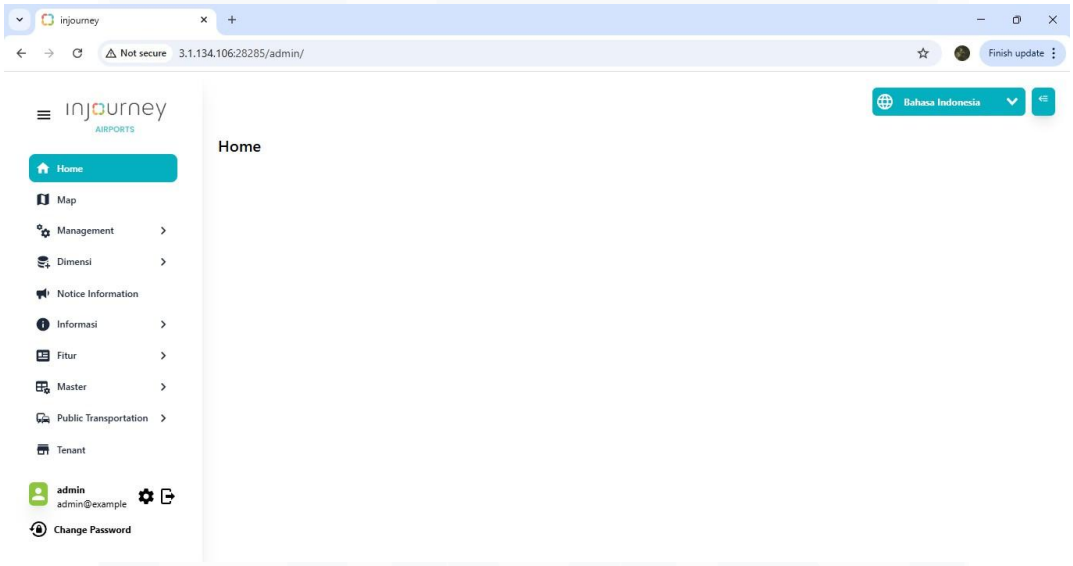
```
<template>
    <div class="w-full">

      <data-list
        v-if="data.gridCfg.setting && data.formCfg.setting"
        class="card"
        ref="listControl"
        :grid-config="data.gridCfg"
        :form-config="data.formCfg"
        grid-read="/cms/terminal/gets"
        form-read="/cms/terminal/get"
        grid-mode="grid"
        grid-delete="/cms/terminal/delete"
        form-keep-label
        form-insert="/cms/terminal/insert"
        form-update="/cms/terminal/update"
        stay-on-form-after-save
        :languages="languageStore().items"
        :selected-language="languageStore().selected"
        :init-app-mode="data.appMode"
        :init-form-mode="data.formMode"
        @formEditData="editRecord"
        @formNewData="newRecord"
        @postSave="postSave"
        :form-fields="['EventID','Name']"

      >
```

```
      >

        <template #form_footer_1="{item}">
            <Attachment ref="layoutCtl" :accepts="['image/*']"  show-image-zoom label="Layout" kind="terminal" :ref-id="item.id" :tags="
        </template>

      </data-list>
    </div>
</template>
<script setup>
    import { layoutStore } from "@/stores/layout";
    import { languageStore } from "@/stores/language";
    import Attachment from "@/components/common/Attachment.vue";
    import helper from "@/scripts/helper.js";
    import { DataList,createGridConfig, gridField,  util,
        formInput,
        createFormConfig,} from "suimjs";

    import { reactive, onMounted, inject,ref } from 'vue';


    layoutStore().name = "tenant";

    const listControl = ref(null);
    const layoutCtl = ref(null)
    const axios = inject('axios')

    const data =  reactive({
```

```
const data =  reactive({
    appMode: "grid",
    formMode: "edit",
    gridCfg:{},
    formCfg:{}
})

function newRecord(record){
    data.record = record
}
function editRecord(record){

}
function postSave(record){
    util.nextTickN(1, () => {
        layoutCtl.value.save()
    })
}
function postSaveImage(resp){
    helper.createImageThumbnail(axios, resp)
}


function createGridCfg(){

    const cfg = createGridConfig()
    cfg.setting.keywordFields = [helper.fieldLanguage("name")]
    cfg.fields = [
        new gridField({field:"name",label:"Name",language:true})
    ]
    data.gridCfg = cfg
}
function createFormCfg(){
    const cfg = createFormConfig("Terminal Form", true);
    cfg.setSectionConfig("row")
    cfg.setting.showTitle = false;


    cfg
        .addSection("General", false)
        .addRowAuto(1, new formInput({field:"name", language: true,languageDirection:"cols", required: true}) )
    data.formCfg = cfg.generateConfig();

}
onMounted(()=>{
    createGridCfg()
    createFormCfg()
})
</script>
```

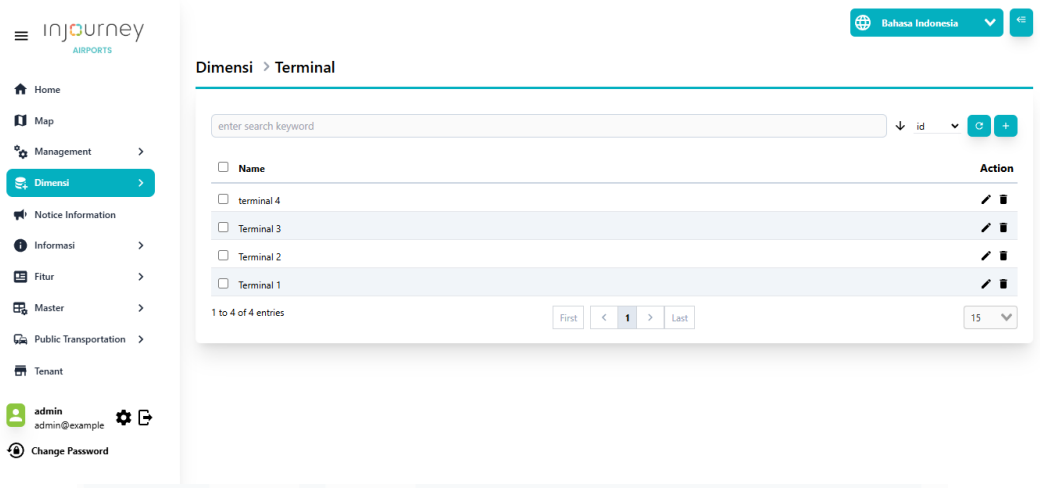
Gambar 3.6 Code Terminal

- Data terminal di-fetch dari API menggunakan axios dan disimpan dalam state lokal.
- v-model digunakan untuk binding data pada form input.
- Event handler untuk submitForm() dan deleteData() memastikan data terkirim dan terhapus dengan benar.
- Proses upload gambar memanfaatkan input file dan dikirimkan sebagai form-data.



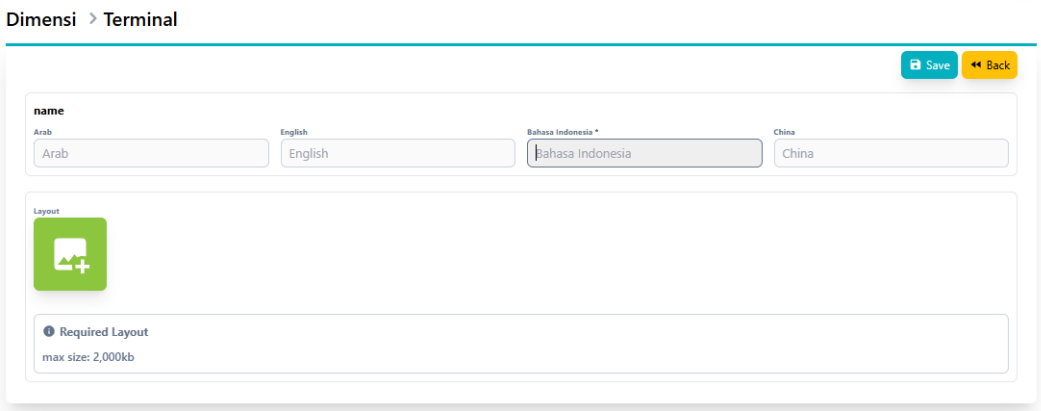
Gambar 3.7 Home CMS

Setelah administrator login ke dalam sistem CMS, mereka dapat mengakses menu “Dimension > Terminal” untuk mengelola informasi terminal yang tersedia di Bandara Soekarno-Hatta. Modul ini dirancang dengan antarmuka yang sederhana namun lengkap agar mudah digunakan oleh staf internal yang tidak memiliki latar belakang teknis.



Gambar 3.8 CMS Terminal

Saat modul dibuka, sistem secara otomatis menampilkan daftar terminal yang sudah tersedia dalam bentuk tabel. Di bagian atas tabel, terdapat kolom pencarian (search bar) yang memungkinkan admin menyaring terminal berdasarkan nama atau jenis. Fitur ini sangat membantu ketika jumlah terminal bertambah banyak atau ketika ingin mengakses data spesifik secara cepat.

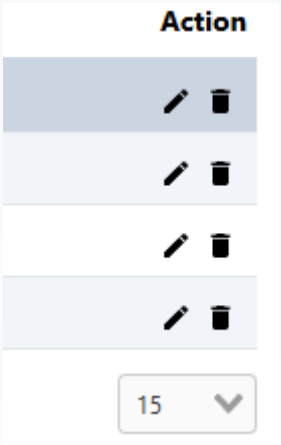


Gambar 3.9 CMS Terminal

Untuk menambah terminal baru, admin cukup menekan tombol “Tambah Terminal” atau “Add”. Akan muncul form input yang harus diisi dengan data berikut:

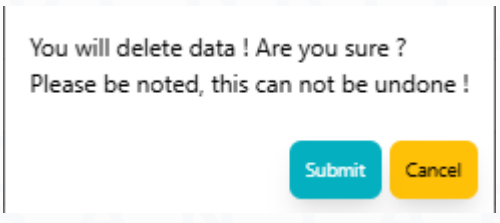
- Nama terminal
- Jenis terminal (menggunakan dropdown, Domestic/International)
- Deskripsi singkat
- Unggahan gambar (upload foto representatif terminal)

Gambar akan langsung ditampilkan sebagai preview, dan ketika form dikirim, data akan disimpan ke database serta gambar akan diunggah ke server MinIO. Hal ini memberikan efisiensi dan fleksibilitas dalam pengelolaan media visual.



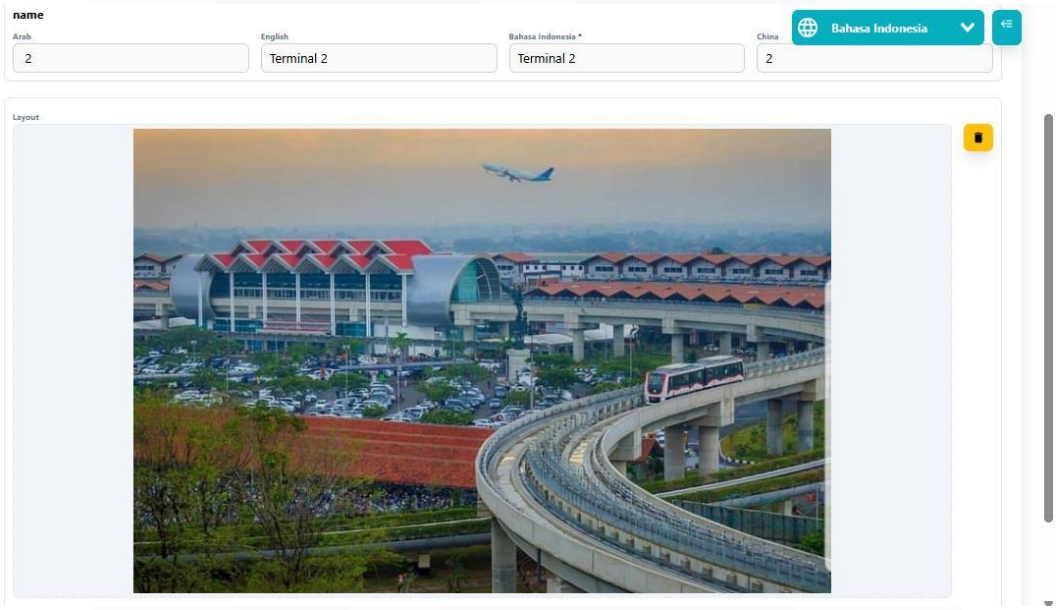
Gambar 3.10 Edit & Delete

Jika ingin memperbarui informasi suatu terminal, admin dapat menekan ikon “Edit” pada baris tabel yang diinginkan. Seluruh field akan dimuat ke dalam form, sehingga admin hanya perlu mengubah bagian tertentu lalu menyimpan perubahan. Ini sangat membantu dalam pembaruan informasi, misalnya saat terjadi renovasi atau perubahan fungsi terminal.



Gambar 3.11 Pop Up Delete

Fitur hapus dilengkapi dengan konfirmasi pop-up agar admin tidak menghapus data secara tidak sengaja. Setelah konfirmasi dilakukan, sistem akan menghapus data dari database dan mengarsipkan data jika diperlukan.

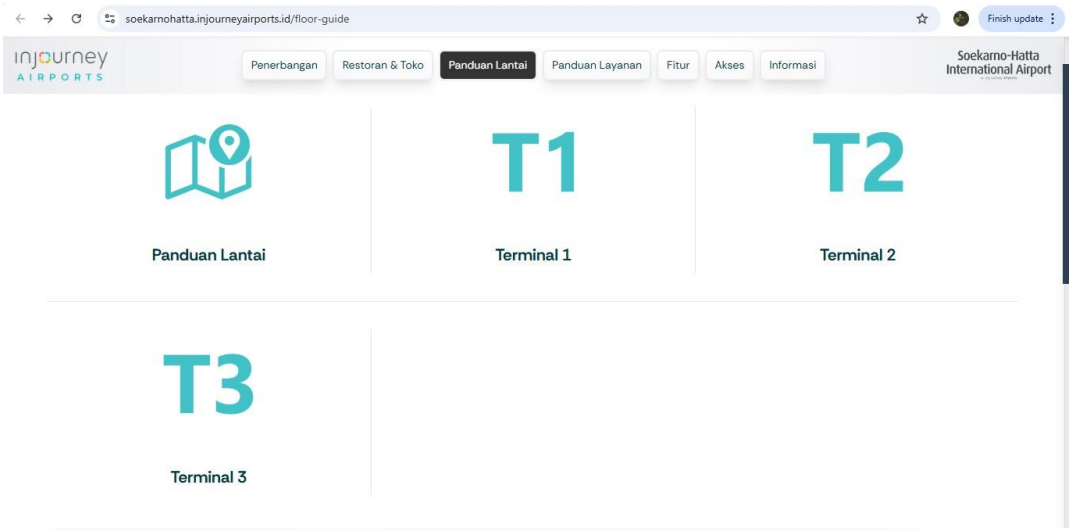


Gambar 3.12 CMS Terminal

Setiap terminal dapat disertai gambar yang diunggah oleh admin melalui form. Gambar tersebut akan dikirim ke layanan penyimpanan berbasis MinIO, yang digunakan karena efisiensi dan skalabilitasnya. Gambar ini kemudian otomatis terintegrasi dan ditampilkan juga di situs publik.

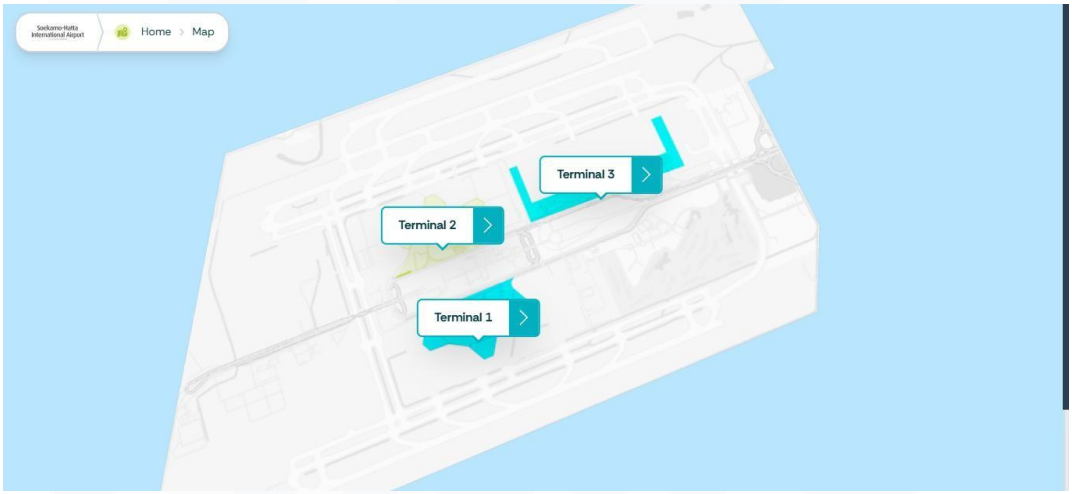
Semua data terminal yang dikelola dari CMS akan langsung berdampak ke tampilan publik di situs soekarnohatta.injourneyairports.id, khususnya pada halaman-halaman berikut:

- Panduan Lantai – Menampilkan informasi nama dan jenis terminal serta detail layout tiap lantai.

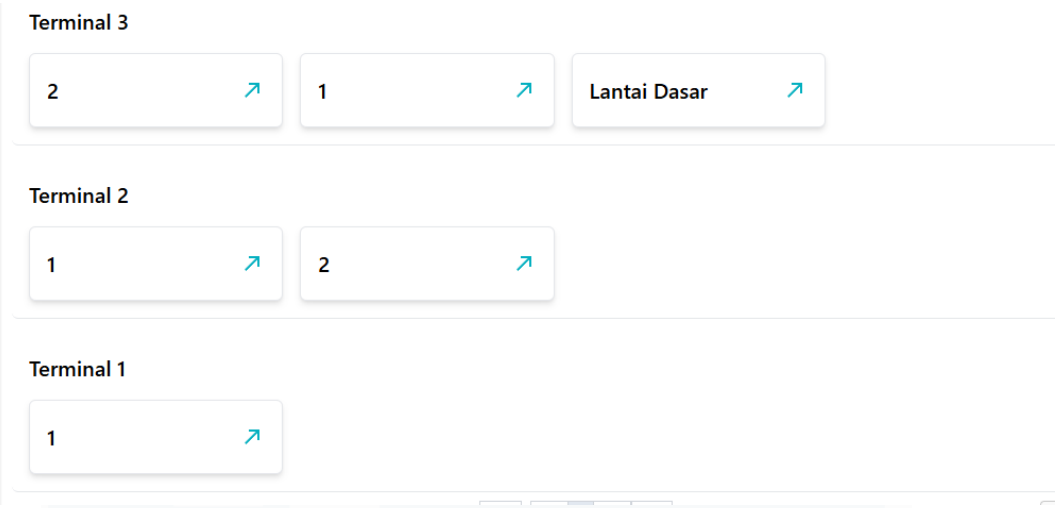


Gambar 3.13 Panduan Lantai

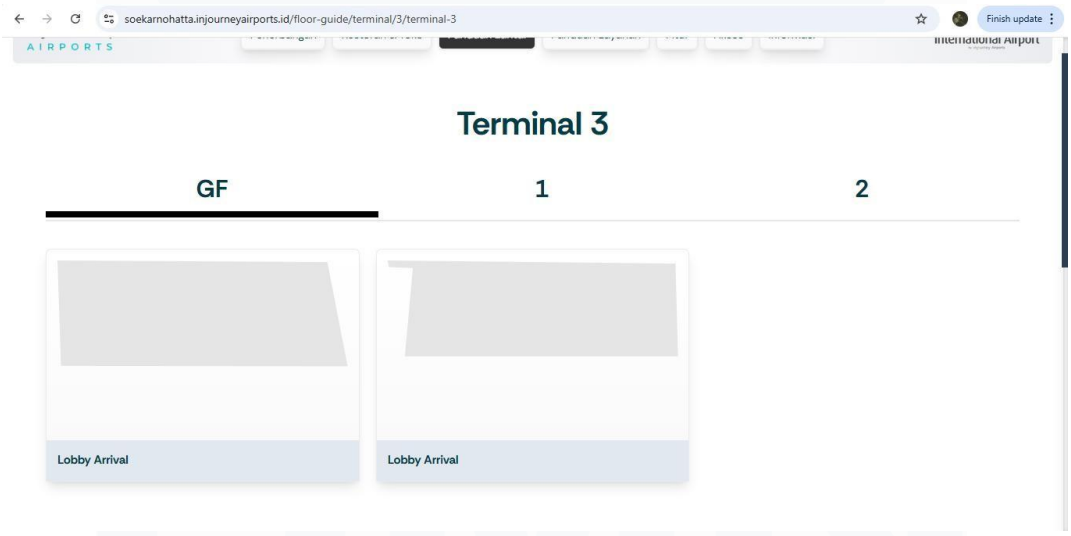
- Peta Interaktif – Posisi dan informasi terminal pada peta dikendalikan dari data CMS.



Gambar 3.14 Peta Interaktif



Gambar 3.15 CMS Terminal



Gambar 3.16 Web Terminal

- Informasi Maskapai & Penerbangan – Nama terminal ditampilkan untuk menjelaskan lokasi keberangkatan atau kedatangan penumpang.

Perubahan di CMS akan langsung tercermin di situs publik melalui API secara sinkron. Misalnya:

- Terminal baru ditambahkan → langsung muncul di daftar terminal publik.

- Gambar diganti → visual di halaman publik otomatis diperbarui.
- Nama terminal diubah → semua rujukan publik ikut menyesuaikan.

3.2.3 Modul TerminalType

Modul Terminal Type berfungsi sebagai pengelola kategori jenis terminal yang ada di Bandara Soekarno-Hatta, seperti Domestik dan Internasional. Modul ini tidak hanya berdiri sendiri, tetapi juga terintegrasi dengan modul lain seperti Terminal—karena setiap data terminal yang dikelola perlu ditautkan ke jenis terminal yang relevan.

Meskipun sederhana, modul ini menjadi bagian esensial dalam struktur data CMS karena bertindak sebagai referensi utama untuk klasifikasi terminal. Penambahan atau perubahan tipe terminal akan secara otomatis memengaruhi dropdown di modul Terminal, dan secara tidak langsung berdampak pada data yang ditampilkan di halaman publik situs bandara.

Fitur yang dikembangkan:

- Tabel data tipe terminal
- Filter nama tipe
- Form tambah dan edit
- Fitur hapus data dengan konfirmasi
- Upload gambar representasi tipe terminal

```
<template>
    <div class="w-full">
      <data-list
        v-if="data.gridCfg.setting && data.formCfg.setting"
        class="card"
        ref="listControl"
        :grid-config="data.gridCfg"
        :form-config="data.formCfg"
        grid-read="/cms/terminaltype/gets"
        form-read="/cms/terminaltype/get"
        grid-mode="grid"
        grid-delete="/cms/terminaltype/delete"
        form-keep-label
        form-insert="/cms/terminaltype/insert"
        form-update="/cms/terminaltype/update"
        stay-on-form-after-save
        :languages="languageStore().items"
        :selected-language="languageStore().selected"

        :init-app-mode="data.appMode"
        :init-form-mode="data.formMode"
        @formEditData="editRecord"
        @formNewData="newRecord"
        @postSave="postSave"

      >
        <template #form_footer_1="{item}">
            <Attachment ref="layoutCtl" :accepts="['image/*']"  show-image-zoom label="Layout" kind="terminaltype" :ref-id="item.id" :tags="[`/te
        </template>
```

```
      </data-list>
    </div>
</template>
<script setup>
    import { layoutStore } from "@/stores/layout";
    import { languageStore } from "@/stores/language";
    import Attachment from "@/components/common/Attachment.vue";
    import helper from "@/scripts/helper.js";
    import { DataList,createGridConfig, gridField, util,
        formInput,
        createFormConfig,} from "suimjs";

    import { reactive, onMounted, inject,ref } from 'vue';


    layoutStore().name = "tenant";

    const listControl = ref(null);
    const layoutCtl = ref(null)
    const axios = inject('axios')

    const data =  reactive({
        appMode: "grid",
        formMode: "edit",
        gridCfg:{},
        formCfg:{}
    })
```

```
function newRecord(record){
}
function editRecord(record){

}
function postSave(record){
    util.nextTickN(1, () => {
        layoutCtl.value.save()
    })
}


function createGridCfg(){

    const cfg = createGridConfig()
    cfg.setting.keywordFields = [helper.fieldLanguage("name")]
    cfg.fields = [
        new gridField({field:"name",language: true})
    ]
    data.gridCfg = cfg
}
function createFormCfg(){
    const cfg = createFormConfig("Terminal Type Form", true);
    cfg.setSectionConfig("row")
    cfg.setting.showTitle = false;
```

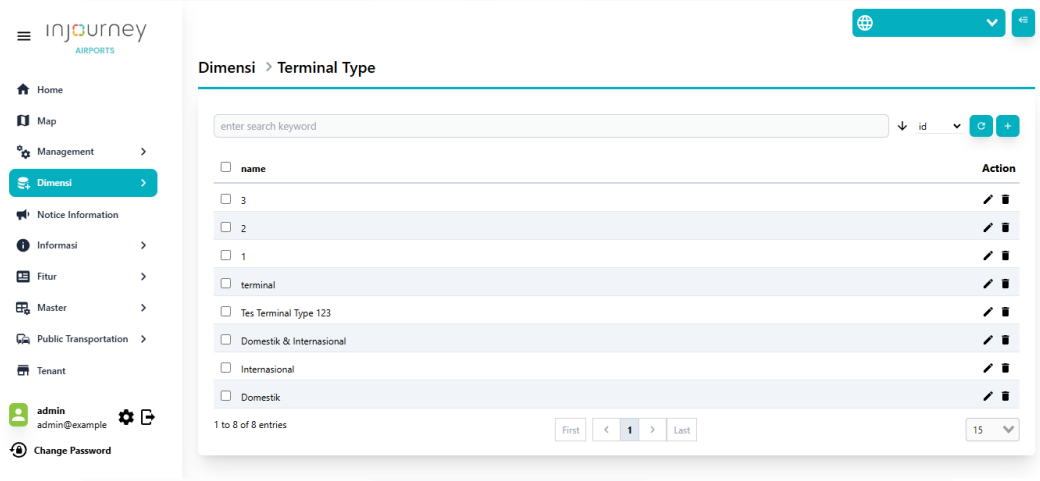
```
    const cfg = createFormConfig("Terminal Type Form", true);
    cfg.setSectionConfig("row")
    cfg.setting.showTitle = false;


    cfg
        .addSection("General", false)
        .addRowAuto(2, new formInput({field:"name", language: true,languageDirection:"cols", required: true}) )
    data.formCfg = cfg.generateConfig();

}
onMounted(()=>{
    createGridCfg()
    createFormCfg()
})
</script>
```

Gambar 3.17 Kodingan TerminalType

- Dropdown tipe terminal dibangun dengan binding nilai dari enum atau list yang telah ditentukan.
- Validasi dilakukan sebelum submit menggunakan pengecekan kondisi di submitForm().

- Gambar diupload menggunakan form-data dan langsung dipreview sebelum pengiriman.
- Struktur data disesuaikan dengan tipe yang tersedia dari backend.

Admin membuka CMS dan memilih submenu “Terminal Type” di bawah menu Dimension. Halaman akan langsung menampilkan daftar tipe terminal yang ada dalam bentuk tabel yang rapi.

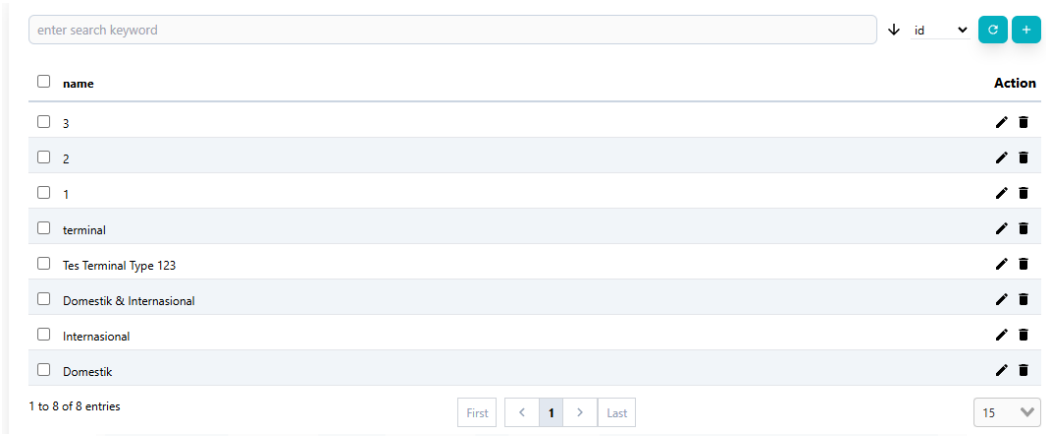


Gambar 3.18 CMS TerminalType

Setiap entri dalam tabel berisi:

- Nama jenis terminal (contoh: Domestik, Internasional)
- Gambar visual (misalnya ikon bendera negara, pesawat)
- Aksi (Edit dan Delete)

Admin dapat menggunakan fitur filter pencarian (search bar) untuk menyaring tipe berdasarkan kata kunci. Ini mempermudah navigasi saat jumlah tipe bertambah.



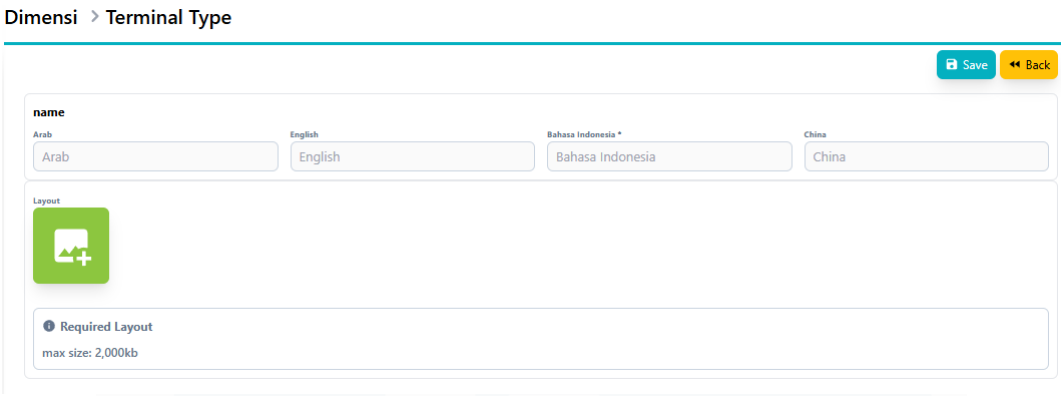
Gambar 3.19 CMS TerminalType

Admin dapat menekan tombol "Tambah Tipe" yang akan membuka form pengisian data. Di dalam form tersebut, terdapat beberapa input penting:

- Nama tipe terminal
- Upload gambar representasi tipe

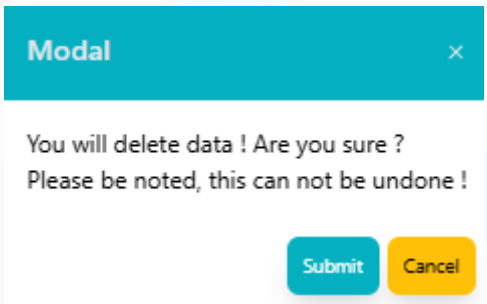
Saat gambar dipilih, akan langsung muncul preview di bawah form. Setelah seluruh input terisi dan validasi terpenuhi (tidak kosong, tidak duplikat), admin dapat menyimpan data. Proses ini akan mengirimkan data dan file ke backend melalui API, dan gambar tersimpan di penyimpanan MinIO.

Jika ada kebutuhan untuk mengganti nama tipe atau mengganti ikon gambar, admin dapat menekan tombol Edit pada baris tabel yang diinginkan. Data sebelumnya akan dimuat secara otomatis ke dalam form. Setelah perubahan dilakukan, data diperbarui di backend dan gambar dapat diganti jika diperlukan.



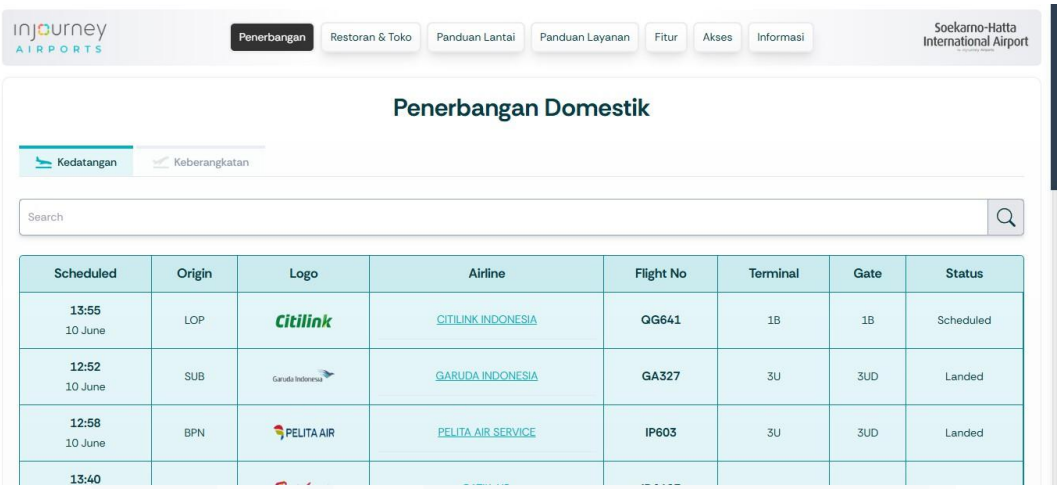
Gambar 3.20 CMS TerminalType

Fungsi penghapusan didesain aman, dengan pop-up konfirmasi. Admin harus menyetujui tindakan ini sebelum sistem menghapus tipe tersebut dari database. Jika tipe ini sedang digunakan oleh data Terminal, sistem akan memberikan notifikasi bahwa tindakan ini bisa berdampak pada modul lain.

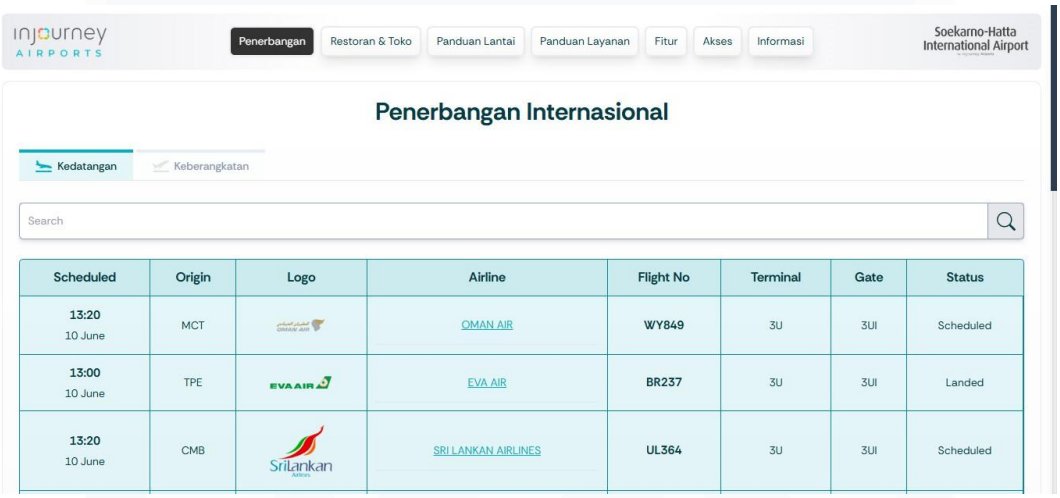


Gambar 3.21 Pop Up Delete

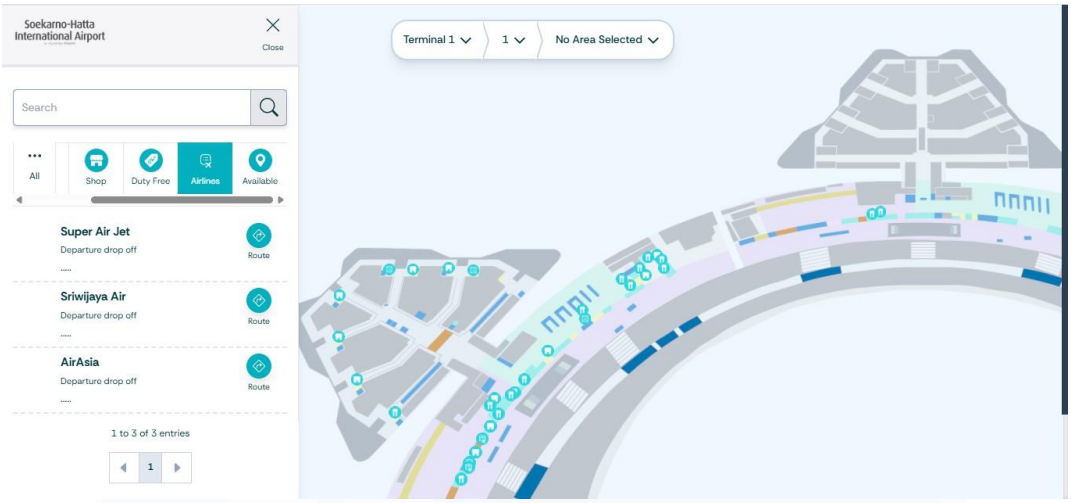
Secara langsung, informasi dari Terminal Type tidak ditampilkan secara eksplisit di situs publik soekarnohatta.injourneyairports.id, namun secara tidak langsung memengaruhi penyusunan konten dan klasifikasi data pada halaman-halaman penting.



Gambar 3.22 Penerbangan Domestik

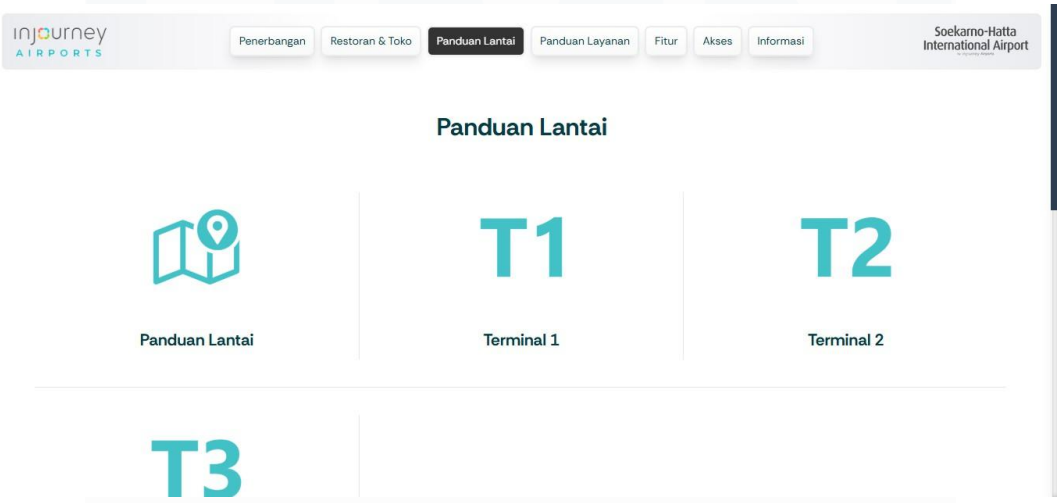


Gambar 3.23 Penerbangan Internasional



Gambar 3.24 Terminal Bandara

Terminal ditampilkan berdasarkan kategorinya (misalnya Terminal 1 untuk domestik, Terminal 3 untuk internasional). Penentuan ini dilakukan dengan cara mengaitkan terminal ke tipe terminal dari CMS. Saat admin menetapkan terminal sebagai "Domestik" atau "Internasional" di CMS, sistem backend menampilkan daftar penerbangan yang sesuai kategori. Terminal yang ditandai sebagai domestik akan terlihat di list penerbangan domestik, dan sebaliknya untuk internasional.



Gambar 3.25 Panduan Lantai

Penomoran dan labeling lantai pada halaman ini bergantung pada kategori terminal. Contoh: Terminal 1 dan 2 yang berfungsi sebagai domestik/internasional, akan memicu pemunculan opsi lantai berdasarkan tipenya masing-masing.

Pada daftar maskapai dan jadwal penerbangan, terminal yang ditampilkan telah diklasifikasi berdasarkan tipenya (tanpa ditampilkan eksplisit). Tipe ini diatur melalui modul Terminal yang merujuk ke data dari modul Terminal Type.

3.2.4 Modul Floor

Modul Floor merupakan komponen penting dalam CMS yang digunakan untuk mengelola informasi lantai dari setiap terminal di Bandara Internasional Soekarno-Hatta. Modul ini sangat bergantung pada relasi dengan data Terminal, karena satu terminal bisa memiliki lebih dari satu lantai. Oleh sebab itu, pengelolaan modul ini menuntut sistem yang dinamis dan konsisten dalam menjaga keterhubungan antar entitas.

Fitur yang dikembangkan:

- Tabel lantai berdasarkan terminal
- Filter pencarian lantai
- Form tambah/edit data lantai
- Upload layout gambar lantai

```
<template>
    <div class="w-full">
      <data-list
        v-if="data.gridCfg.setting && data.formCfg.setting"
        class="card"
        ref="listControl"
        :grid-config="data.gridCfg"
        :form-config="data.formCfg"
        grid-read="/cms/floor/gets"
        form-read="/cms/floor/get"
        grid-mode="grid"
        grid-delete="/cms/floor/delete"
        form-keep-label
        form-insert="/cms/floor/insert"
        form-update="/cms/floor/update"
        stay-on-form-after-save
        :languages="languageStore().items"
        :selected-language="languageStore().selected"

        :init-app-mode="data.appMode"
        :init-form-mode="data.formMode"
        @formEditData="editRecord"
        @formNewData="newRecord"
        @postSave="postSave"
        :form-fields="['EventID']"

      >
        <template #form_footer_1="{item}">
            <Attachment ref="layoutCtl" :accepts="['image/*']"  show-image-zoom label="Layout" kind="floor" :ref-id="item.id" :tags="[`/floor/${i
        </template>
```

```
      </data-list>
    </div>
</template>
<script setup>
    import { layoutStore } from "@/stores/layout";
    import Attachment from "@/components/common/Attachment.vue";
    import { languageStore } from "@/stores/language";
    import helper from "@/scripts/helper.js";

    import { DataList,createGridConfig, gridField, util,
        formInput,
        createFormConfig,} from "suimjs";

    import { reactive, onMounted, inject,ref } from 'vue';


    layoutStore().name = "tenant";

    const listControl = ref(null);
    const layoutCtl = ref(null)
    const axios = inject('axios')

    const data =  reactive({
        appMode: "grid",
        formMode: "edit",
        gridCfg:{},
        formCfg:{}
    })
```

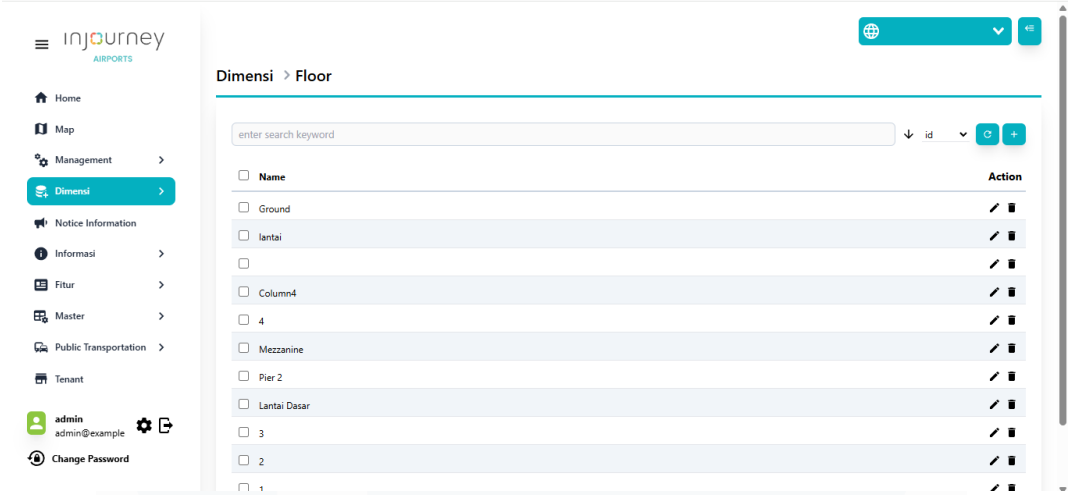
```
function newRecord(record){
}
function editRecord(record){

}
function postSave(record){

    util.nextTickN(1, () => {
        layoutCtl.value.save()
    })
}



function createGridCfg(){

    const cfg = createGridConfig()
    cfg.setting.keywordFields = [helper.fieldLanguage("name")]
    cfg.fields = [
        new gridField({field:"name",label:"Name", language: true})
    ]
    data.gridCfg = cfg
}
function createFormCfg(){
    const cfg = createFormConfig("Terminal Form", true);
    cfg.setSectionConfig("row")
    cfg.setting.showTitle = false;
```

```
        cfg.setSectionConfig("row")
        cfg.setting.showTitle = false;


        cfg
            .addSection("General", false)
            .addRowAuto(1, new formInput({field:"name", language: true,languageDirection:"cols", required: true}) )
        data.formCfg = cfg.generateConfig();

    }
    onMounted(()=>{
        createGridCfg()
        createFormCfg()
    })
</script>
```

Gambar 3.26 Kodingan Floor

- Validasi dilakukan pada form input seperti nama lantai dan nomor urut.
- Upload gambar ditujukan untuk memberikan blueprint layout setiap lantai.

- Data yang ditampilkan dapat difilter berdasarkan terminal yang dipilih.



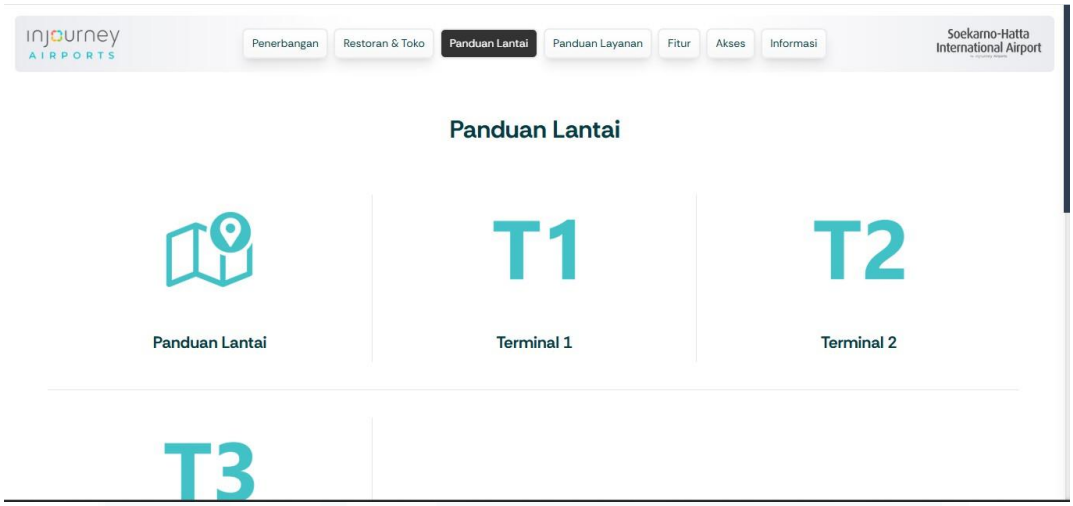
Gambar 3.27 CMS Floor

Admin membuka CMS dan masuk ke menu “Dimension > Floor”. Sistem langsung menampilkan daftar lantai dalam bentuk tabel yang dikaitkan dengan masing-masing terminal.

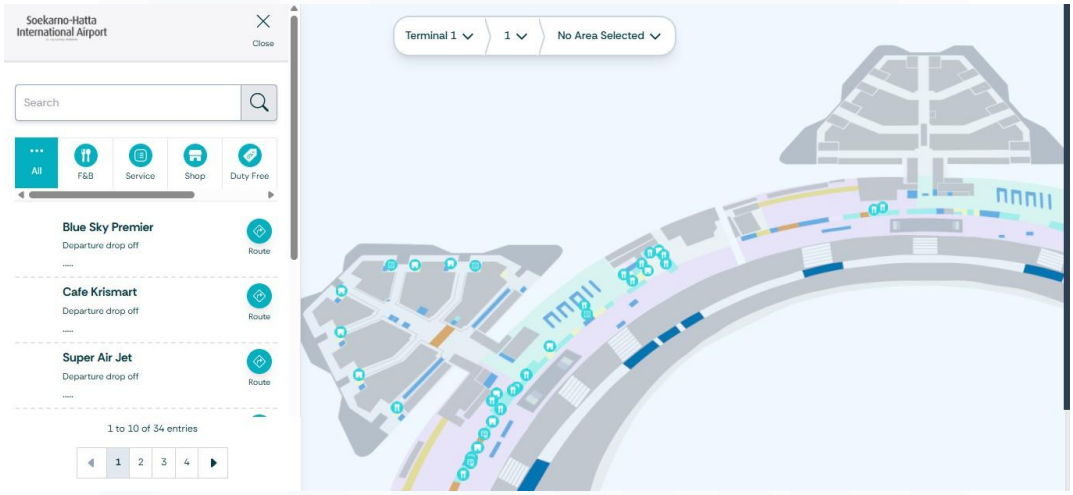
Admin dapat menyaring data berdasarkan nama lantai atau berdasarkan terminal tertentu. Fitur ini sangat berguna saat data lantai yang dikelola berjumlah banyak atau tersebar di berbagai terminal.

Jika ada perubahan, admin tinggal memilih tombol Edit pada baris lantai tertentu. Data yang sudah ada akan otomatis dimuat ke form, sehingga admin bisa memperbarui salah satu bagian tanpa menghapus data sebelumnya.

Fitur unggah gambar memungkinkan admin mengelola visualisasi layout lantai seperti denah fasilitas, rute masuk, jalur kursi roda, dll. Gambar yang diunggah bisa ditampilkan sebagai referensi visual di situs publik.



Gambar 3.28 Panduan Lantai



Gambar 3.29 Terminal Bandara

3.2.5 Modul Parking

Modul Parking merupakan salah satu modul paling kompleks dalam CMS karena tidak hanya menyimpan informasi lokasi parkir, tetapi juga tarif-tarif yang bervariasi berdasarkan jenis kendaraan. Untuk menangani kebutuhan ini, sistem dirancang dengan struktur parent-child table, di mana setiap data parkir (lokasi) memiliki anak data berupa daftar tarif per tipe kendaraan.

Dengan adanya modul ini, pihak pengelola bandara dapat secara fleksibel mengelola informasi lokasi dan harga parkir, termasuk menambahkan lokasi baru, memperbarui tarif secara berkala, dan menyajikan informasi yang selalu terkini ke publik.

Fitur yang dikembangkan:

- Menampilkan data lokasi parkir
- Form tambah/edit lokasi dan tarif kendaraan
- Filter berdasarkan nama lokasi parkir
- Tabel tarif per tipe kendaraan (anak dari parent lokasi)
- Upload gambar lokasi parkir

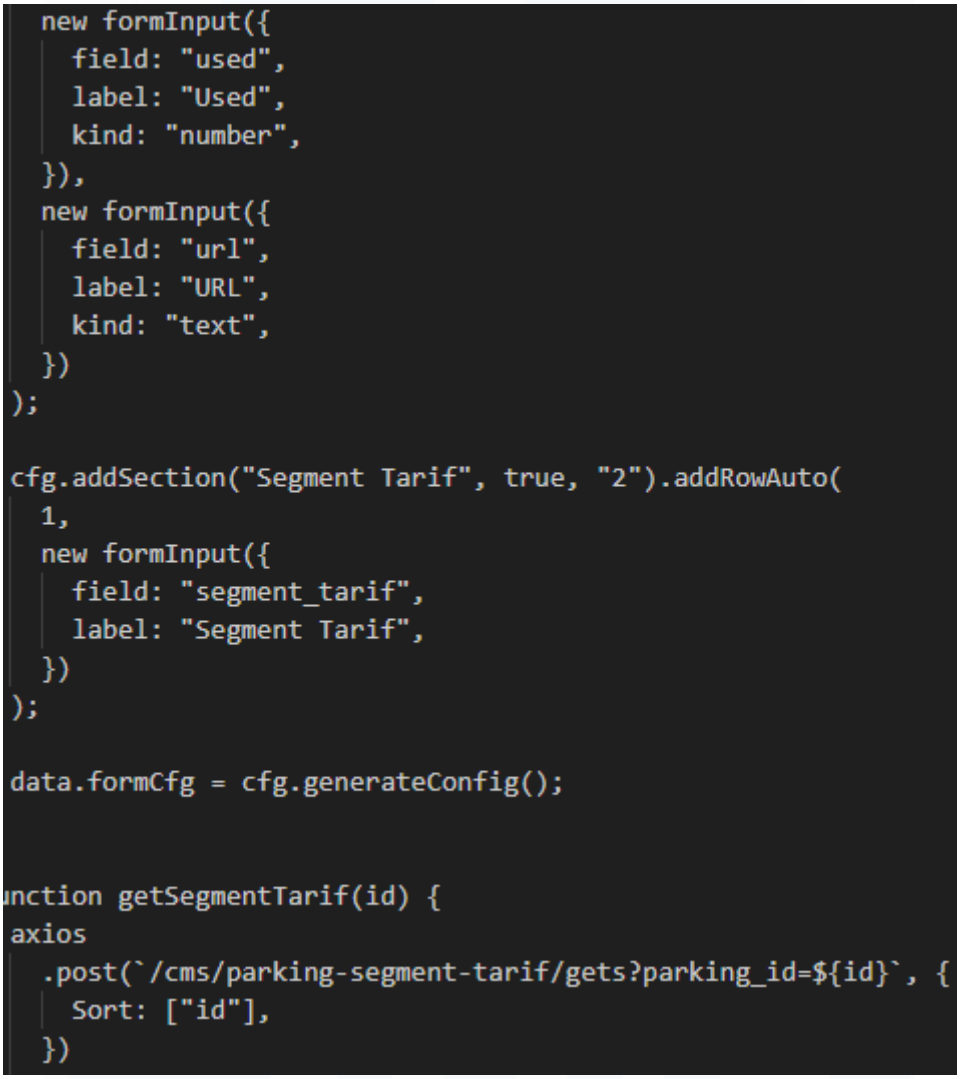
```
<template>
  <div class="w-full">
    <data-list
      v-if="data.gridCfg.setting && data.formCfg.setting"
      class="card"
      ref="listControl"
      :grid-config="data.gridCfg"
      :form-config="data.formCfg"
      grid-read="/cms/parking/gets"
      form-read="/cms/parking/get"
      grid-mode="grid"
      grid-delete="/cms/parking/delete"
      form-keep-label
      form-insert="/cms/parking/insert"
      form-update="/cms/parking/update"
      stay-on-form-after-save
      :languages="languageStore().items"
      :selected-language="languageStore().selected"
      :grid-fields="['available']"
      :form-fields="['segment_tarif', 'upload_foto']"
      :init-app-mode="data.appMode"
      :init-form-mode="data.formMode"
      @formEditData="editRecord"
      @formNewData="newRecord"
      @preSave="preSave"
      @postSave="postSave"
    >
      <template #grid_available="{ item }">
        <div v-if="item.total && item.used">
          {{ parseFloat(item.total) - parseFloat(item.used) }}
```

```
    >
      <template #grid_available="{ item }">
        <div v-if="item.total && item.used">
          {{ parseFloat(item.total) - parseFloat(item.used) }}
        </div>
      </template>

      <template #form_input_segment_tarif="{}">
        <segment-tarif
          v-model="data.segment_tarif"
          @delete-records="deleteSegmentTarif"
        ></segment-tarif>
      </template>
    </data-list>
  </div>
</template>

<script setup>
import { layoutStore } from "@/stores/layout";
import { languageStore } from "@/stores/language";
import helper from "@/scripts/helper.js";
import {
  DataList,
  createGridConfig,
  gridField,
  formInput,
  createFormConfig,
  util,
} from "suimjs";
```

```
} from "suimjs";
import { reactive, onMounted, inject, ref } from "vue";
import SegmentTarif from "./widget/SegmentTarif.vue";

layoutStore().name = "tenant";

const listControl = ref(null);
const axios = inject("axios");

const data = reactive({
  appMode: "grid",
  formMode: "edit",
  gridCfg: {},
  formCfg: {},
  record: {},
  segment_tarif: [],
});

function openForm(record) {
  util.nextTickN(1, () => {
    data.record = record;
  });
}

function newRecord(record) {
  openForm(record);
}
```

```
function editRecord(record) {
  getSegmentTarif(record.id);
  openForm(record);
}

function preSave(record) {
  record.available = record.total - record.used;
}

function postSave(record) {
  const segmentTarif = data.segment_tarif.map((item) => ({
    ...item,
    parking_id: record.id,
  }));
  saveSegmentTarif(segmentTarif);
}

function createGridCfg() {
  const cfg = createGridConfig();
  cfg.setting.keywordFields = [helper.fieldLanguage("description")];
  cfg.fields = [
    new gridField({
      field: "dimension_dimension_terminal_name_id",
      label: "Terminal",
    }),
    new gridField({
      field: "description",
      label: "Description",
      language: true,
    }),
    new gridField({
      field: "total",
      label: "Capacity",
    }),
    new gridField({
      field: "used",
      label: "Used",
    }),
    new gridField({
      field: "available",
      label: "Available",
    }),
  ];
  data.gridCfg = cfg;
}

function createFormCfg() {
  const cfg = createFormConfig("Parking Form", true);
  cfg.setSectionConfig("row");
  cfg.setGroupingSection(true);
  cfg.setting.showTitle = false;
```

```
cfg.addSection("General", false, "1").addRowAuto(
  1,
  new formInput({
    field: "description",
    label: "Description",
    multiRow: 4,
    required: true,
    language: true,
  })
);

cfg.addSection("General2", false, "1").addRowAuto(
  1,
  new formInput({
    field: "dimension_terminal_id",
    label: "Terminal",
    useList: true,
    lookupUrl: "/cms/terminal/gets",
    lookupKey: "id",
    lookupLabels: ["name_id"],
    lookupSearchs: ["name_id"],
  }),
  new formInput({
    field: "total",
    label: "Capacity",
    kind: "number",
  }),
  new formInput({
    field: "used",
```



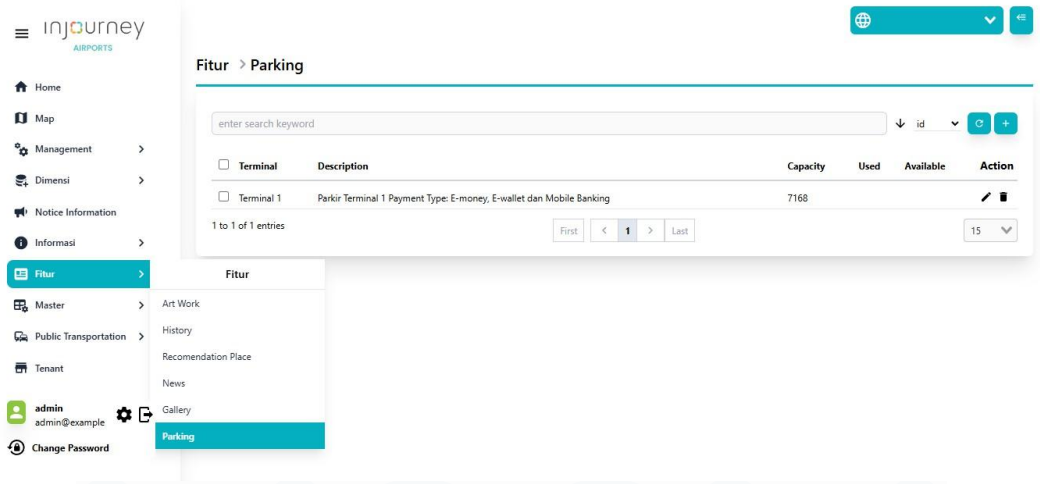
```
  new formInput({
    field: "used",
    label: "Used",
    kind: "number",
  }),
  new formInput({
    field: "url",
    label: "URL",
    kind: "text",
  })
);

cfg.addSection("Segment Tarif", true, "2").addRowAuto(
  1,
  new formInput({
    field: "segment_tarif",
    label: "Segment Tarif",
  })
);

data.formCfg = cfg.generateConfig();


unction getSegmentTarif(id) {
axios
  .post(`/cms/parking-segment-tarif/gets?parking_id=${id}`, {
    Sort: ["id"],
  })
```

```
function getSegmentTarif(id) {
  axios
    .post(`/cms/parking-segment-tarif/gets?parking_id=${id}`, {
      Sort: ["id"],
    })
    .then((r) => {
      data.segment_tarif = r.data.data;
    });
}

function saveSegmentTarif(records) {
  records.map((rec) => {
    if (rec.id) {
      axios.post("/cms/parking-segment-tarif/update", rec);
    } else {
      axios.post("/cms/parking-segment-tarif/insert", rec);
    }
  });
}

function deleteSegmentTarif(records) {
  records.map((rec) => {
    if (rec.id) {
      axios.post("/cms/parking-segment-tarif/delete", { id: rec.id })
    }
  });
}
```

```
function deleteSegmentTarif(records) {
  records.map((rec) => {
    if (rec.id) {
      axios.post("/cms/parking-segment-tarif/delete", { id: rec.id });
    }
  });
}

onMounted(() => {
  createGridCfg();
  createFormCfg();
});
</script>
```

Gambar 3.30 Kodingan Parking

- Data ditampilkan dalam bentuk nested table.
- Fungsi penambahan dan pengeditan menggunakan struktur array untuk sub-tabel tarif.
- Form validation dilakukan secara berlapis, baik untuk lokasi maupun detail tarif.
- Gambar diupload untuk dokumentasi visual lokasi parkir.
- Komunikasi ke API dilakukan untuk setiap perubahan data parent maupun child.



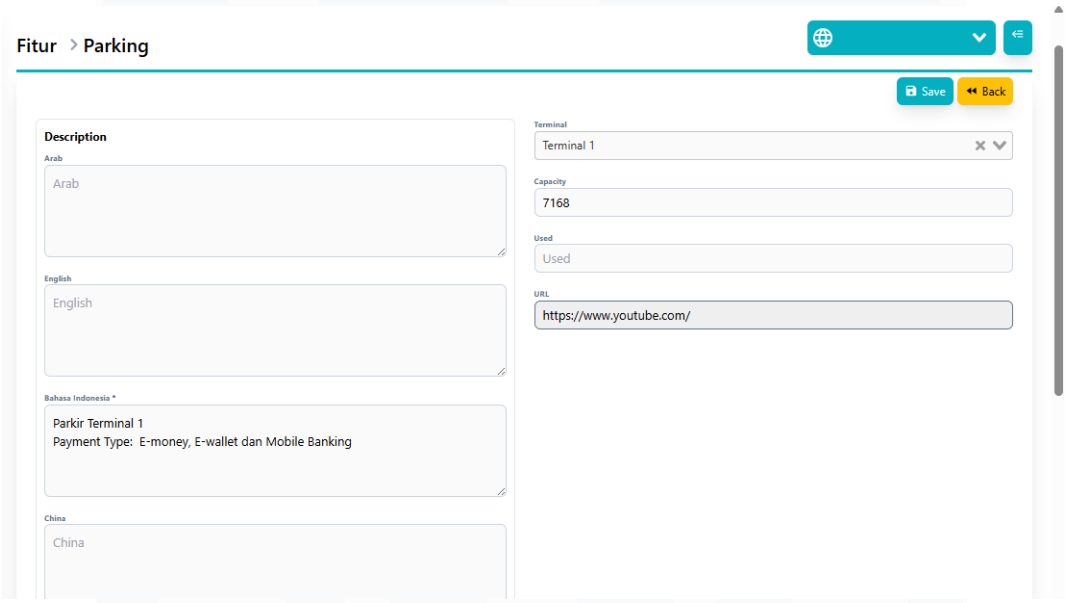
Gambar 3.31 CMS Parking

Admin membuka CMS dan mengakses menu “Fitur > Parking”. Di halaman ini, sistem menampilkan tabel utama berisi daftar lokasi parkir. Setiap baris dalam tabel utama dapat diperluas (expandable) untuk menampilkan nested table berisi rincian tarif parkir untuk berbagai tipe kendaraan. Fitur pencarian memudahkan admin untuk mencari lokasi parkir berdasarkan nama. Filter ini sangat membantu jika jumlah lokasi cukup banyak atau tersebar di beberapa terminal. Admin dapat menekan tombol “Tambah Parkir” yang akan menampilkan form input sebagai berikut:

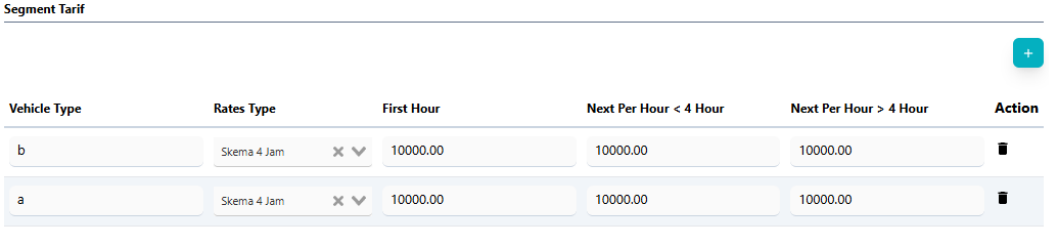
- Nama lokasi parkir

- Terminal atau area parkir
- Kapasitas
- Harga parkir

Admin dapat mengedit detail lokasi dan tarif kendaraan kapan saja. Data tarif yang lama dapat ditimpa atau ditambah sesuai kebutuhan. Untuk menghapus, sistem akan meminta konfirmasi dan memperingatkan apabila lokasi parkir tersebut sedang digunakan oleh modul lain atau ditautkan ke publik.



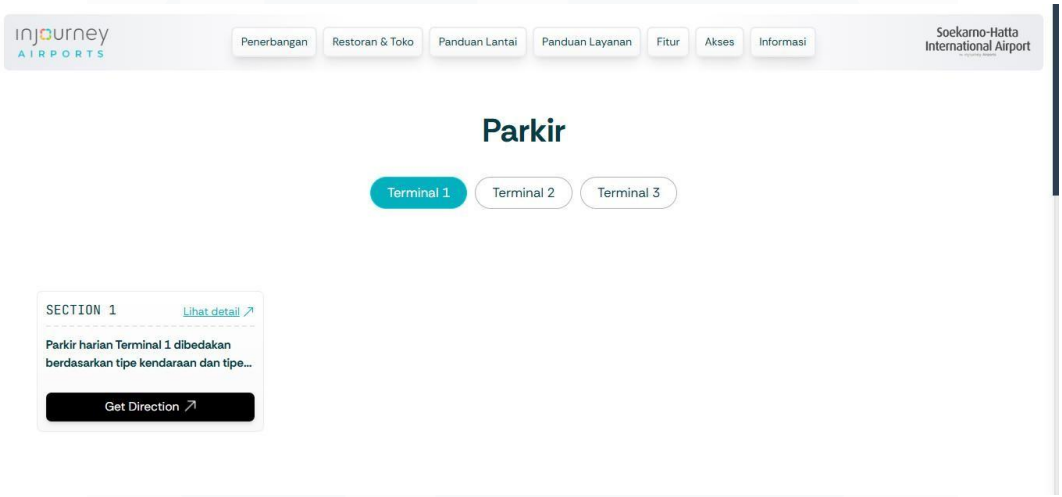
Gambar 3.32 CMS Parking



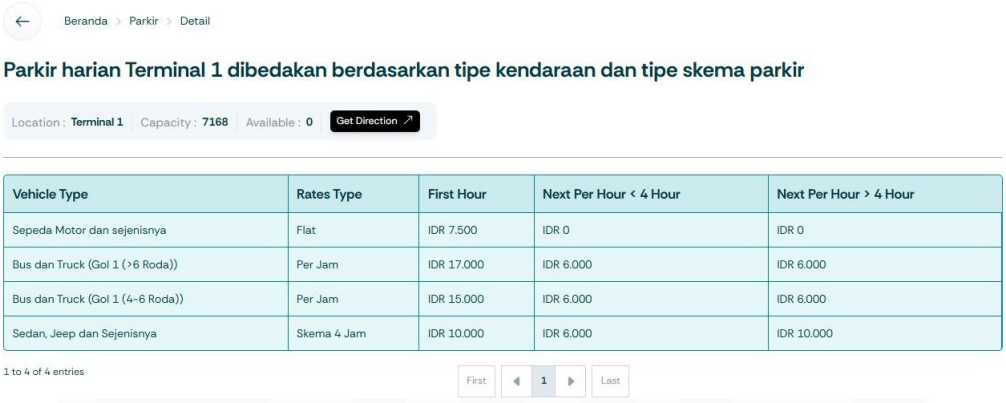
Gambar 3.33 Segment Tarif

Tipe kendaraan bisa diisi dan diikuti dengan:

- Rates Type (Skema 4 Jam, Per Jam, Flat)
- Tarif First Hour
- Tarif Next Per Hour < 4 Hour
- Tarif Next Per Hour > 4 Hour



Gambar 3.34 Website Parkir



Vehicle Type	Rates Type	First Hour	Next Per Hour < 4 Hour	Next Per Hour > 4 Hour
Sepeda Motor dan sejenisnya	Flat	IDR 7.500	IDR 0	IDR 0
Bus dan Truck (Gol 1 (>6 Roda))	Per Jam	IDR 17.000	IDR 6.000	IDR 6.000
Bus dan Truck (Gol 1 (4-6 Roda))	Per Jam	IDR 15.000	IDR 6.000	IDR 6.000
Sedan, Jeep dan Sejenisnya	Skema 4 Jam	IDR 10.000	IDR 6.000	IDR 10.000

Gambar 3.35 Harga Parkir Harian

- Data lokasi parkir: Nama lokasi dan jenis terminal langsung ditarik dari data CMS.
- Tarif kendaraan: Informasi tarif untuk motor, mobil, dan bus ditampilkan berdasarkan isian dari sub-tabel tarif di CMS.

- Nested display: Sama seperti di CMS, situs publik menampilkan detail tarif berdasarkan lokasi tertentu dengan struktur terorganisir.

3.2.6 QA Testing, Uji Tampilan, dan Dokumentasi

Tahap Quality Assurance (QA) testing dilakukan setelah pengembangan modul-modul utama CMS (Terminal, Terminal Type, Floor, Parking) selesai dikembangkan. QA dilakukan untuk memastikan fungsionalitas sistem berjalan sesuai rencana dan tidak terdapat bug yang mengganggu. Proses QA melibatkan beberapa langkah berikut:

1. Simulasi Demo CMS ke Atasan: Penulis melakukan presentasi internal kepada pembimbing proyek (PM) dari PT Cipta Prima Nugraha. Demo dilakukan dengan menampilkan modul-modul CMS yang telah selesai pada browser, memperlihatkan alur tambah, ubah, hapus data dan upload gambar pada modul Terminal dan Parking.

2. Audit Internal CMS Dimulai: Setelah demo, tim internal mengadakan audit internal untuk mengevaluasi logika CMS, aksesibilitas UI, performa, dan kesesuaian dengan kebutuhan client. Audit dilakukan berdasarkan checklist internal dari tim developer.

3. Fix Bug Hasil Audit Awal: Ditemukan beberapa masalah seperti kesalahan validasi input dan tampilan tabel yang tidak responsif. Perbaikan dilakukan oleh penulis dibantu oleh tim.

3.2.7 Review External

Setelah audit internal selesai, CMS diuji dan dievaluasi oleh tim eksternal (perwakilan dari klien, InJourney Airports dan PT Angkasa Pura Indonesia). Evaluasi ini menjadi bagian dari tahapan review antarmuka sistem dan struktur data CMS.

1. Perbaikan Komponen Hasil Evaluasi Audit: Terdapat masukan seperti penyederhanaan input form dan penggantian penggunaan istilah yang membingungkan seperti tarif parkir.

2. Tambah Validasi pada Form: Penulis menambahkan validasi pada modul Terminal dan Parking menggunakan vee-validate dan custom error message untuk memperkuat kontrol input.

3.2.8 Observasi & Buffer Revisi

Tahapan ini adalah buffer yang disediakan untuk melakukan perbaikan minor, observasi, dan testing tambahan berdasarkan masukan terakhir dari stakeholder.

1. Feedback Tambahan CMS: Diterima permintaan untuk menambahkan kolom catatan di modul Parking untuk mencatat kondisi atau lokasi fisik parkir.

2. QA Akhir Semua Modul CMS: Tahap final QA dilakukan untuk memastikan tidak ada error blocking, semua fitur berfungsi dengan benar, dan loading cepat.

3. Observasi Testing Fitur Tambahan: Tim melakukan observasi terhadap CMS yang telah dimodifikasi dengan feedback baru. Semua fitur dinyatakan lolos uji fungsionalitas.

3.2.9 Review CMS

1. Review Feedback Minor CMS: Feedback minor meliputi revisi label tombol, penyesuaian padding dan margin di beberapa komponen, serta optimalisasi UX.

2. Dokumentasi Akhir & Backup CMS: Penulis membantu tim dokumentasi teknis untuk membuat backup CMS dalam bentuk zip file.

3.2.10 Kick-Off Penyusunan Laporan Akhir & Dokumentasi Proyek

1. Mulai Bantu PM Susun Draft Laporan Akhir untuk Klien: Penulis mulai dilibatkan oleh Project Manager (PM) untuk membantu menyusun draft laporan akhir CMS bagian teknis.

2. Merangkum Hasil Sprint CMS Menjadi Dokumentasi Teknis: Setiap fitur CMS yang telah dikembangkan dirangkum menjadi halaman dokumentasi berisi: tujuan modul, diagram arsitektur, alur navigasi, dan kendala teknis.

3.2.11 Finalisasi Konten Laporan

1. Review Bersama PM: Struktur Laporan dan Lampiran: Penulis melakukan review bersama PM terkait isi laporan akhir.

2. Polishing Dokumentasi Teknis (Diagram Alur, Screenshot, dsb): Penulis menyusun ulang bagian visual dalam dokumentasi.

3.2.12 Penyelesaian Laporan & Presentasi Akhir

1. Finalisasi Isi Laporan Akhir (Word & PDF): Setelah semua bagian dipastikan lengkap, laporan dikompilasi dan diekspor dalam format Word & PDF. Penulis juga mencetak dokumen sesuai standar kampus.

Secara keseluruhan, kontribusi proyek ini tidak hanya terbatas pada pengembangan teknis CMS, tetapi juga memberikan nilai tambah strategis bagi:

- **Pihak Bandara**: sebagai pemilik sistem informasi publik yang kini lebih efisien dan terintegrasi.
- **PT. Cipta Prima Nugraha**: sebagai mitra teknologi yang memperoleh prototype CMS modern untuk portofolio layanan digital.
- **Mahasiswa dan Universitas**: sebagai contoh sukses kerja praktik dan kolaborasi antara akademisi dan industri.

- **Pengguna Publik**: karena website bandara menjadi lebih informatif, responsif, dan mudah diakses kapan saja.

3.3 Kendala yang Ditemukan

Selama proses pelaksanaan kerja magang, khususnya dalam pengembangan modul CMS berbasis Vue.js untuk website Bandara Soekarno-Hatta, penulis menghadapi berbagai kendala dan tantangan, baik dari segi teknis, komunikasi tim, maupun keterbatasan pribadi sebagai mahasiswa magang yang masih dalam tahap pembelajaran. Beberapa kendala utama yang ditemukan antara lain:

1. Keterbatasan Pengetahuan dan Pengalaman Vue.js

Sebagai mahasiswa magang, penulis masih dalam proses belajar dan belum memiliki pengalaman yang cukup dalam mengembangkan sistem CMS secara profesional, khususnya menggunakan framework Vue.js. Hal ini membuat proses pemahaman struktur kode, component-based architecture, serta komunikasi antar modul frontend-backend membutuhkan waktu adaptasi lebih lama.

2. Koordinasi dalam Tim Pengembangan

Koordinasi antara tim frontend, backend, serta Project Manager (PM) terkadang mengalami kendala, terutama saat terjadi perubahan spesifikasi. Penyesuaian antara kebutuhan client dengan realisasi teknis dari tim memerlukan komunikasi intens dan dokumentasi teknis yang rapi, yang kadang belum tersedia secara lengkap.

3. Perubahan Input dari Client

Terdapat beberapa perubahan mendadak dari pihak client terkait field input, baik dari segi label, tipe data, maupun struktur field. Perubahan ini menyebabkan ketidaksesuaian antara frontend dan backend, sehingga memicu bug saat penyimpanan data, validasi, maupun saat data ditampilkan di antarmuka pengguna.

4. Bug dalam Penyimpanan Data

Beberapa field dalam form tidak tersimpan atau tidak terupdate dengan benar di database. Bug ini umumnya disebabkan oleh perubahan spesifikasi input dari client, yang belum sepenuhnya disinkronkan antar bagian sistem. Akibatnya, data menjadi tidak konsisten atau hilang saat diakses kembali.

5. Permintaan Validasi Tambahan untuk Gambar Upload

Klien meminta penambahan validasi dalam proses upload gambar agar sesuai dengan standar tampilan situs versi sebelumnya. Validasi ini mencakup ukuran file maksimum, resolusi minimum, serta format gambar yang diizinkan. Penerapan validasi ini memerlukan pemahaman terhadap library dan teknik manipulasi file di sisi frontend.

3.4 Solusi atas Kendala yang Ditemukan

Untuk mengatasi kendala-kendala yang ditemukan selama pelaksanaan kerja magang, penulis bersama tim pengembangan mencoba menerapkan berbagai solusi, baik dari aspek teknis, komunikasi, maupun peningkatan kemampuan individu. Berikut adalah beberapa solusi yang dilakukan:

1. Belajar Mandiri dan Bimbingan dari Tim

Penulis aktif mempelajari dokumentasi Vue.js, membaca kode proyek yang sudah ada, serta berdiskusi dengan tim pengembang untuk memahami struktur dan alur kerja CMS. Dengan bantuan rekan kerja dan PM, penulis secara bertahap mampu menyelesaikan beberapa fitur seperti form input, integrasi tabel parent-child, dan fitur upload gambar.

2. Peningkatan Dokumentasi Teknis dan Standarisasi Komunikasi

Untuk mengurangi miskomunikasi dalam tim, dilakukan pencatatan perubahan spesifikasi dalam bentuk dokumentasi bersama (shared document). Komunikasi antara frontend dan backend dikonsolidasikan melalui standup meeting harian atau mingguan, serta penggunaan tools seperti Google Docs untuk mencatat kebutuhan dan update dari client secara berkala.

3. Sinkronisasi Cepat saat Ada Perubahan dari Client

Ketika terjadi perubahan input dari client, tim langsung melakukan analisis dampak terhadap sistem dan segera melakukan sinkronisasi antara frontend dan backend. Penulis terlibat dalam proses revisi struktur form dan label agar sesuai dengan spesifikasi terbaru, dan pengujian ulang dilakukan untuk mencegah bug di sisi pengguna.

4. Debugging dan Refactor Modul Data

Untuk menyelesaikan bug pada penyimpanan data, tim melakukan debugging secara menyeluruh, termasuk pemeriksaan kembali pada form binding, struktur model, serta endpoint API.

5. Implementasi Validasi Upload Gambar

Validasi ukuran, dimensi, dan format gambar ditambahkan. Penulis belajar langsung dari contoh-contoh validasi sebelumnya dan berlatih membuat error message yang ramah pengguna apabila file yang diupload tidak sesuai.

Sebelum pengembangan CMS berbasis Vue.js ini dilakukan, proses pengelolaan konten informasi pada website bandara masih dilakukan secara manual atau semi-manual, tanpa dashboard administratif yang memadai. Hal ini menyebabkan keterlambatan update informasi publik dan inkonsistensi tampilan antar halaman. CMS baru yang dikembangkan melalui proyek ini memberikan solusi yang signifikan dengan menyediakan antarmuka modular, real-time update, integrasi API antar modul, serta validasi otomatis untuk meningkatkan akurasi data dan pengalaman pengguna. Dengan sistem ini, seluruh pengelolaan konten kini dapat dilakukan melalui satu portal administratif yang intuitif dan efisien, berbeda dengan sistem sebelumnya yang fragmentaris.