BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Organisasi

Posisi yang ditetapkan selama pelaksanaan program kerja magang di PT. Vanz Inovatif Teknologi adalah sebagai *Developer* di departemen *Internal/Inhouse Application and Development*. Program kerja magang ini dilaksanakan dengan bimbingan Bapak Aditia Prasetio sebagai *supervisor* dan *Team Leader* dari tim *developer Internal/Inhouse Application Development*. Koordinasi dilakukan setiap harinya melalui aplikasi *Google Meet*, media komunikasi yang digunakan adalah WhatsApp, dan email untuk menginformasikan dan menjadwalkan *meeting*, baik *meeting* internal ataupun *meeting* dengan klien.

3.2 Tugas yang Dilakukan

Terdapat tugas-tugas yang diberikan oleh *supervisor* untuk merancang, membangun, dan melakukan *bug fixing* pada fitur-fitur yang ada pada aplikasi manajemen data berbasis *web*, baik dalam proyek yang masih dalam tahap *development* maupun proyek yang sudah *live*.

Tugas pertama dalam pelaksanaan program kerja magang di PT. Vanz Inovatif Teknologi adalah melanjutkan pembuatan aplikasi *e-commerce* yang merupakan proyek internal dari perusahaan. Proyek internal ini bertujuan untuk membuat sebuah *platform* untuk memfasilitasi penjualan atau pemberian barang bekas, baik barang yang masih layak pakai seperti perabotan, perangkat elektronik, dan furnitur, maupun makanan dan minuman yang masih layak dikonsumsi. Proyek ini memiliki aplikasi *mobile* yang dapat digunakan oleh penjual untuk mem-*posting* barang bekas yang ingin mereka jual atau berikan secara gratis dan pembeli untuk mencari barang yang mereka perlukan. Proyek ini juga memiliki *Content Management System (CMS)* yang dapat digunakan oleh pihak admin aplikasi yang dapat mengatur konten yang ditampilkan pada aplikasi, konten yang ditampilkan meliputi iklan, promosi, ataupun *posting* dari penjual.

Selama menjalankan tugas pada proyek internal tersebut, perusahaan juga sedang menjalankan kontrak dengan instansi pemerintah dalam pengembangan halaman *web e-commerce*. Halaman *web* tersebut dibuat dan dikembangkan oleh instansi pemerintah yang terkait dengan bantuan dari PT. Vanz Inovatif Teknologi.

Aplikasi tersebut digunakan oleh madrasah dan pesantren yang menerima bantuan dana dari pemerintah untuk membeli perabotan, perangkat, maupun peralatan yang mereka butuhkan. Pembelian tersebut dilakukan dengan mitra yang sudah mendaftarkan diri ke instansi pemerintah. Tugas dalam kontrak ini diberikan langsung oleh pegawai instansi pemerintah untuk membantu penambahan fitur pada lapisan *backend*.

Selanjutnya, diberikan tugas pada proyek perusahaan dengan klien untuk membangun aplikasi *online traning*. *Online training* ini diselenggarakan oleh World Scope Corporate College (WSCC) dengan nama BJB.it dengan materi seputar materi teknologi informasi. Pada proyek ini, diberikan tugas untuk menambahkan fitur baru beserta proses *create, read, update, dan delete (CRUD)* dasar dengan penambahan fungsi untuk melakukan impor data untuk memasukkan data dalam jumlah banyak. Dalam fungsi impor data, dilakukan validasi agar data yang diimpor tidak menyebabkan anomali atau *error* pada proses yang menggunakan data tersebut.

Tugas berikutnya adalah *request* dari klien perusahaan properti RE/MAX yang ingin membuat *website blog* dimana mereka dapat menampilkan berbagai artikel seputar properti dan artikel-artikel tersebut dapat dibaca oleh umum. Pada proyek ini, tugas yang diberikan adalah membangun *frontend web* blog yang diintegrasikan dengan *backend*.

Selama melaksanakan program kerja magang ini, didapat juga beberapa *bug* yang ditemukan oleh karyawan klien ataupun permintaan untuk penambahan fitur pada proyek TrackTrace. *Bug fixing* pada proyek ini diprioritaskan setiap kali mendapat laporan dari karyawan klien.

3.3 Uraian Pelaksanaan Magang

Selama durasi program kerja magang di PT. Vanz Inovatif Teknologi, tugas-tugas yang diberikan terbagi ke beberapa proyek, yaitu:

1. Proyek *E-Commerce* Internal Proyek ini adalah proyek internal perusahaan dimana perusahaan ingin membuat sebuah *platform* jual-beli barang bekas. Pengguna dapat menjual, membeli, ataupun memberikan barang bekas mereka secara gratis melalui *platform* tersebut.

2. Proyek BJB.it Proyek ini adalah pembuatan sebuah aplikasi *mobile* untuk mengadakan *training* bagi pelanggan klien. Perusahaan diminta untuk

membuat sistem untuk memanajemen data dari aplikasi tersebut, seperti proses *create, read, update, delete* (CRUD), menyimpan data *training*, memproses data pengguna, sistem penilaian, dan sistem absen harian.

3. Proyek Instansi Pemerintah Proyek ini adalah kerja sama antara perusahaan dengan instansi pemerintah untuk mengembangkan fitur pada *backend* dari *web online shop* yang ditujukan untuk pencairan dana bantuan bagi madrasah.

4. Proyek RE/MAX Blog Proyek ini adalah *request* dari klien perusahaan atas nama RE/MAX Indonesia untuk membuat sebuah halaman *web blog*. Pada *web blog* tersebut, pihak RE/MAX Indonesia dapat menampilkan artikel-artikel yang mereka tulis dan artikel-artikel tersebut dapat dibaca oleh publik.

5. Proyek TracknTrace Proyek ini adalah sebuah aplikasi berbasis *web* dimana klien dapat melakukan manajemen data produksi dan dsitribusi dari perusahaan mereka yang bergerak di bidang medis. Aplikasi pada proyek ini sudah operasional dan digunakan dalam proses bisnis klien, tetapi perusahaan masih menerima permintaan untuk menambahkan fitur apabila dibutuhkan oleh klien.

Pelaksanaan kerja magang diuraikan seperti pada Tabel 3.1.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu Ke -	Pekerjaan yang dilakukan
1	*Setup* untuk melanjutkan pengembangan aplikasi *e-commerce* proyek internal.
2, 3, & 4	Mengembangkan *frontend* aplikasi *e-commerce* dan melakukan integrasi dengan *backend*.
5, 6, & 7	*Setup* proyek BJB.it dan menambahkan fitur, serta menambahkan fitur pada *backend* pada aplikasi instansi pemerintah.
8	Menambahkan fitur untuk menampilkan data yang dikunci atau dinonaktifkan, menambahkan data produsen/*manufacturer* pada proyek TrackNTrace dan melanjutkan penambahan fitur pada proyek BJB.it
9 - 14	Pengembangan *frontend* dan integrasi *backend* untuk halaman *web blog* RE/MAX.
15 - 16	Pembuatan fitur ekspor laporan nilai pada proyek BJB.it

3.3.1 Proyek RE/MAX Blog

Proyek ini merupakan *request* dari klien PT. Vanz Inovatif Teknologi atas nama RE/MAX Indonesia yang bergerak di bidang penjualan dan penyewaan properti. Klien ingin membuat sebuah halaman *web blog* dimana mereka dapat menulis dan menampilkan berbagai artikel seputar properti yang dapat dilihat oleh pengguna umum. Halaman *web blog ini* dapat diakses melalui halaman *web* utama RE/MAX Indonesia.

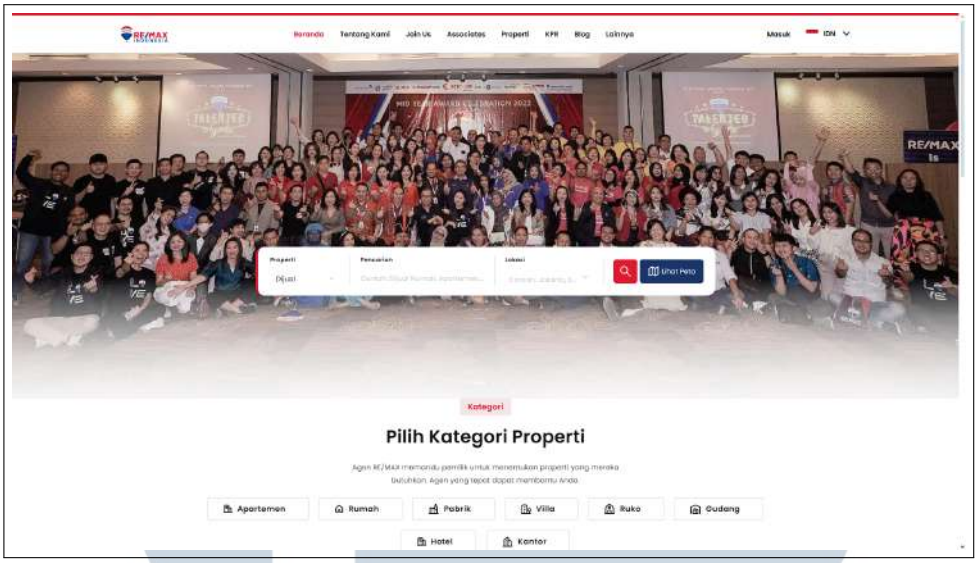
Desain tampilan halaman dari *web* RE/MAX Blog dibuat oleh tim UI/UX perusahaan menggunakan aplikasi Figma. Desain ini dibuat agar tim *developer* memiliki pedoman rancangan elemen visual seperti warna, *font*, dan *layout* yang seragam pada setiap halaman yang ada pada aplikasi *web* RE/MAX Blog. Selain itu, semua pemrosesan data seperti proses *create, read, update, and delete* (CRUD) pada *backend* beserta interaksinya dengan data pada *database* dan dokumentasi dari semua *endpoint* API *backend* yang dapat digunakan dibuat oleh rekan kerja dari tim *developer* perusahaan.

Halaman *web* RE/MAX Blog pada lapisan *frontend* dibangun dan dikembangkan menggunakan bahasa pemrograman TypeScript, *framework* Next.js, *library frontend* React, dan komponen-komponen UI dari *library* Ant Design. Perusahaan memberikan *base code* melalui repositori GitHub proyek RE/MAX Blog sebelum dilakukan pengembangan, sehingga pengerjaan proyek ini dapat langsung dimulai dengan pembangunan halaman utama.

Framework Next.js digunakan karena memiliki fitur *server-side rendering*. Fitur ini melakukan proses *render* halaman seperti pengambilan data dari *backend API* dan pemrosesan data tersebut dilakukan pada sisi *server*. *Browser* pembaca menerima dan menampilkan halaman dalam bentuk HTML lengkap tanpa harus melakukan proses *render* pada perangkat mereka masing-masing. Fitur ini mempercepat waktu muat halaman bagi pembaca dan meningkatkan pengalaman pembaca dalam menggunakan halaman *web* RE/MAX Blog.

Selain Next.js, *library* React dan Ant Design adalah dua *library frontend* yang digunakan dalam proyek RE/MAX Blog. React adalah *library* yang digunakan untuk membangun tampilan halaman *web* yang dinamis serta interaktif dan Ant Design adalah *library* UI berbasis React yang merupakan koleksi komponen siap pakai untuk membangun tampilan halaman yang konsisten dan responsif. Proyek RE/MAX Blog menggunakan *library* React and Ant Design untuk mempercepat dan mempermudah proses pengembangan melalui penggunaan komponen yang siap

pakai.

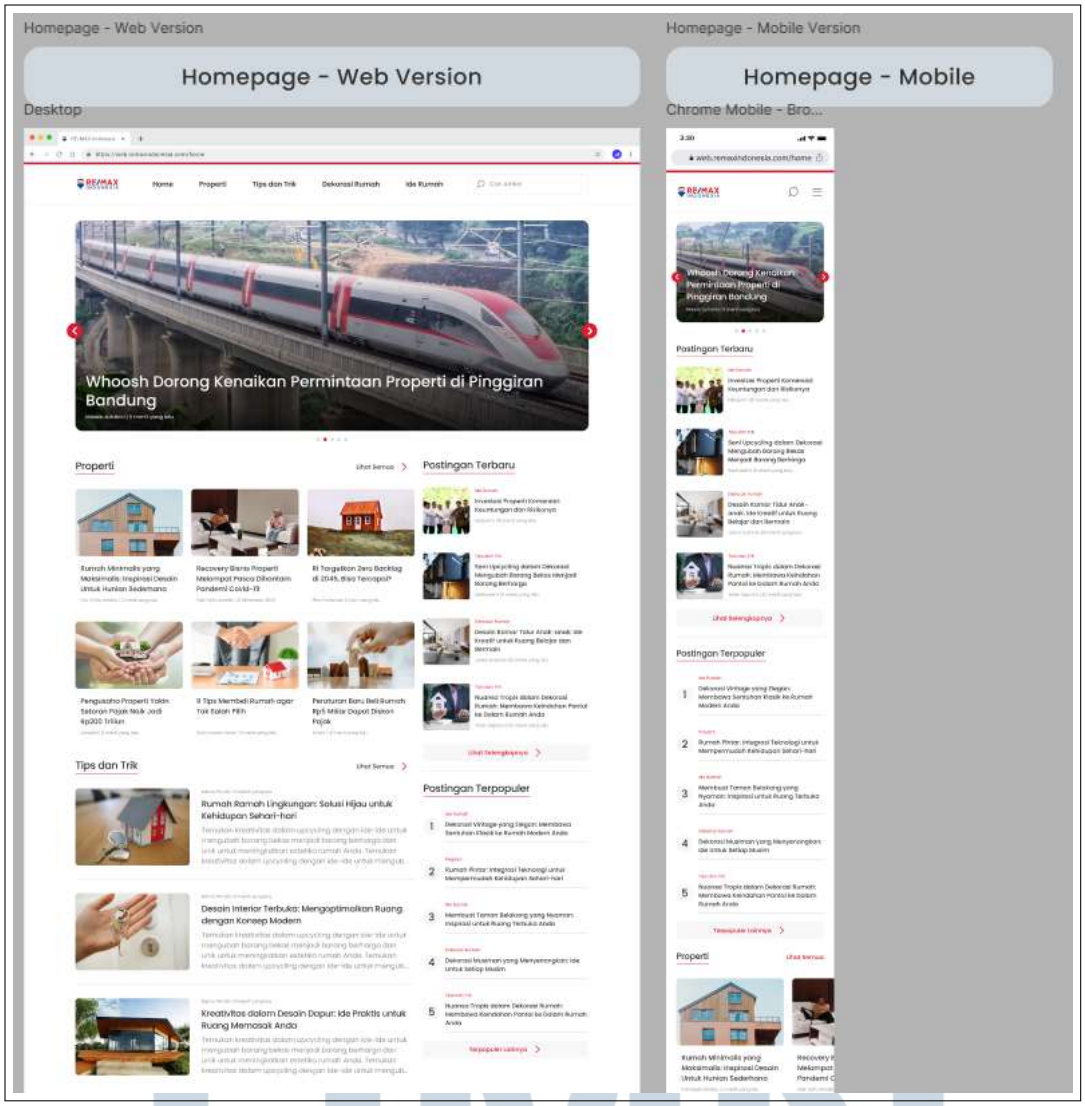


Gambar 3.1. Halaman utama *web* RE/MAX Indonesia

Pada proyek ini, tugas yang diberikan terfokus pada lapisan *frontend* halaman *web* RE/MAX Blog dengan bantuan dari rekan kerja pada lapisan *backend* dan desain UI/UX halaman *web*. Halaman *web* RE/MAX Blog merupakan cabangan dari halaman *web* utama RE/MAX yang berisi artikel-artikel dengan topik properti dan topik-topik lainnya yang berkaitan dengan properti. Halaman *web blog* ini dapat diakses melalui bar navigasi yang berada pada bagian atas halaman utama *web* RE/MAX yand sudah dibuat dan dikembangkan sebelumnya dan dapat dilihat pada Gambar 3.1.

A. Halaman utama

Halaman yang dibuat pertama kali dalam proyek RE/MAX Blog adalah halaman utama dari *web blog*. Halaman ini adalah halaman pertama yang ditampilkan kepada pembaca ketika mereka membuka RE/MAX Blog. Pada halaman ini, ditampilkan berbagai pratinjau artikel berdasarkan kategori, artikel terbaru, dan juga artikel terpopuler.



Gambar 3.2. Desain halaman utama *web* RE/MAX Blog

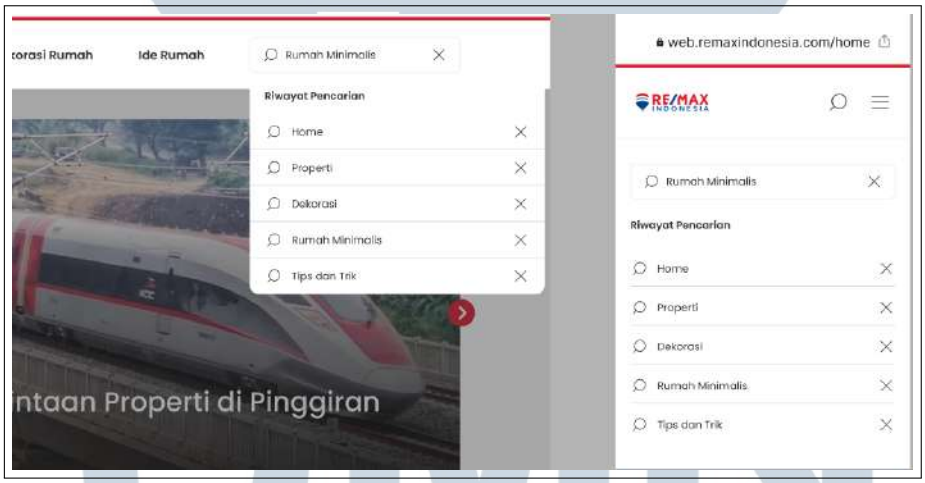
Desain halaman utama RE/MAX Blog pada *browser desktop* dan pada *browser mobile* dapat dilihat pada Gambar 3.2. Halaman ini dapat dibagi menjadi beberapa elemen utama, yaitu:

1. ***Bar navigasi***

   Bar navigasi ini menggunakan komponen yang sudah dibuat sebelumnya pada halaman utama *web* RE/MAX Indonesia dengan tombol-tombol yang diubah untuk mengarahkan pembaca ke halaman kategori yang mereka pilih.

Gambar 3.3. Desain bar navigasi *web* RE/MAX Blog

Desain dari bar navigasi dapat dilihat pada Gambar 3.3. Pada desain, contoh nama-nama kategori yang digunakan adalah ”Properti”, ”Tips dan Trik”, ”Dekorasi Rumah”, dan ”Ide Rumah”. Pada bar navigasi, ditampilkan semua nama kategori tersebut dalam sebuah tombol yang di sebelah kanan tombol ”Home”. Tombol ”Home” akan mengarahkan pembaca ke halaman utama RE/MAX Blog dan setiap tombol dengan nama kategori akan mengarahkan pembaca ke halaman kategori yang sesuai dengan tombol nama kategori yang mereka tekan, misalnya ketika pembaca menekan tombol ”Properti”, maka mereka akan diarahkan ke halaman kategori ”Properti”.



Gambar 3.4. Desain fitur pencarian artikel *web* RE/MAX Blog

Pada bagian kanan bar navigasi terdapat kolom *input* yang menggunakan komponen ”Input” dan ikon ”SearchOutlined” dari *library* Ant Design. Kolom *input* tersebut adalah fitur pencarian artikel dan desain dari tampilan fitur ini dapat dilihat pada Gambar 3.4. Ketika pembaca menekan kolom *input* pada *browser desktop*, ditampilkan riwayat kata kunci pencarian yang dicari sebelumnya. Sedangkan pada *browser mobile*, kolom *input* pada bar navigasi digantikan dengan ikon ”SearchOutlined” dan ketika pembaca menekan ikon tersebut, mereka akan diarahkan ke halaman pencarian. Pada halaman pencarian tersebut, terdapat kolom *input* untuk melakukan

pencarian dan ditampilkan riwayat pencarian. Riwayat pencarian pada *browser desktop* dan *browser mobile* dibuat menggunakan komponen ”List”, ikon ”SearchOutlined”, dan ikon ”CloseOutlined” dari *library* Ant Design.

Pembaca dapat melakukan pencarian dengan memasukkan kata kunci atau memilih kata kunci yang pernah dicari sebelumnya dari riwayat pencarian yang ditampilkan di bawah kolom *input*. Pencarian akan diproses setelah pembaca memasukkan kata kunci dan menekan ikon ”SearchOutlined” yang ada di bagian kiri kolom *input* atau ketika pembaca memilih kata kunci pada riwayat pencarian. Pembaca akan diarahkan ke halaman hasil pencarian setelah pencarian diproses oleh halaman *web*.

2. ***Carousel* artikel yang di-*highlight***

   Carousel adalah sebuah elemen interaktif yang berfungsi untuk menampilkan artikel-artikel yang di-*highlight*. Elemen ini terdiri beberapa *slide* dan setiap *slide* berisi gambar utama, judul, penulis, dan tanggal publikasi dari artikel. Artikel yang ditampilkan pada *slide* akan digeser secara otomatis oleh halaman *web* atau digeser oleh pembaca.

Gambar 3.5. Desain *carousel* artikel pada halaman utama *web* RE/MAX Blog

Desain dari *carousel* dapat dilihat pada Gambar 3.5. Tidak ada perbedaan tampilan pada *carousel* ketika ditampilkan pada *browser desktop* maupun *browser mobile*. *Carousel* ini dibuat menggunakan komponen ”Carousel” dari *library* Ant Design dengan tambahan dua ikon dari *library* Ant Design pada dua sisi *carousel*, ikon ”LeftOutlined” di sebelah kiri dan ikon ”RightOutlined” di sebelah kanan. Pada setiap *slide* dalam *carousel*, ditampilkan gambar utama, judul, nama penulis, dan tanggal publikasi

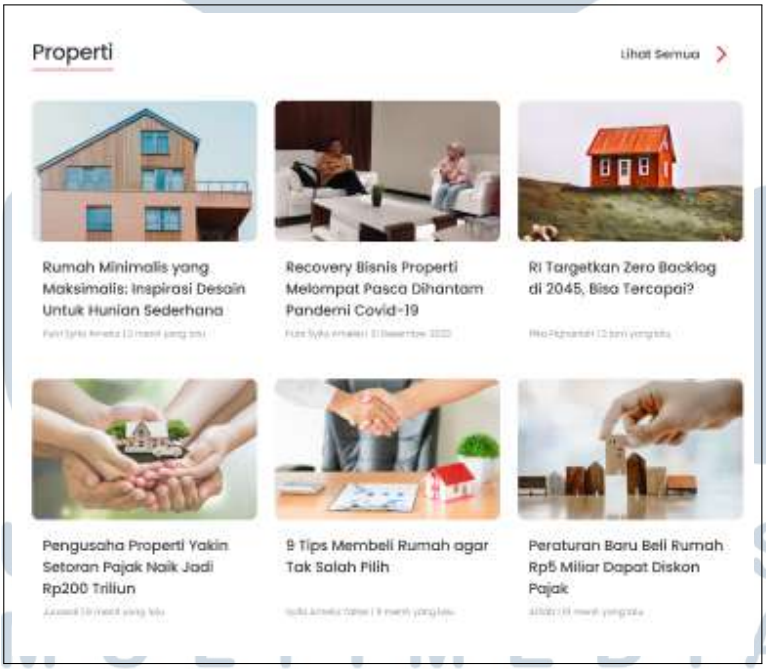
dari artikel. Jika pembaca ingin mengakses konten dari pratinjau artikel yang ditampilkan pada *slide carousel*, pembaca dapat menekan *slide* untuk membuka halaman dari artikel tersebut. *Slide* dari *carousel* akan bergeser secara otomatis jika tidak ada interaksi dari pembaca, tetapi pembaca juga dapat menggeser *slide* dengan menekan kedua ikon yang ada pada kedua sisi *carousel*.

3. **Pratinjau artikel per kategori**

   Pada halaman utama RE/MAX Blog, ditampilkan pratinjau/*preview* dari bermacam-macam artikel yang dapat dibaca oleh pembaca pada halaman *web* RE/MAX Blog. Berbagai pratinjau artikel tersebut dikelompokkan berdasarkan kategori utama dari artikel tersebut. Pengelompokkan berdasarkan kategori ini ditampilkan dalam dua susunan, yaitu:

   - Susunan *grid*

     Pada susunan *grid*, sekelompok pratinjau artikel ditampilkan dalam susunan tata letak yang berbasis baris dan kolom.



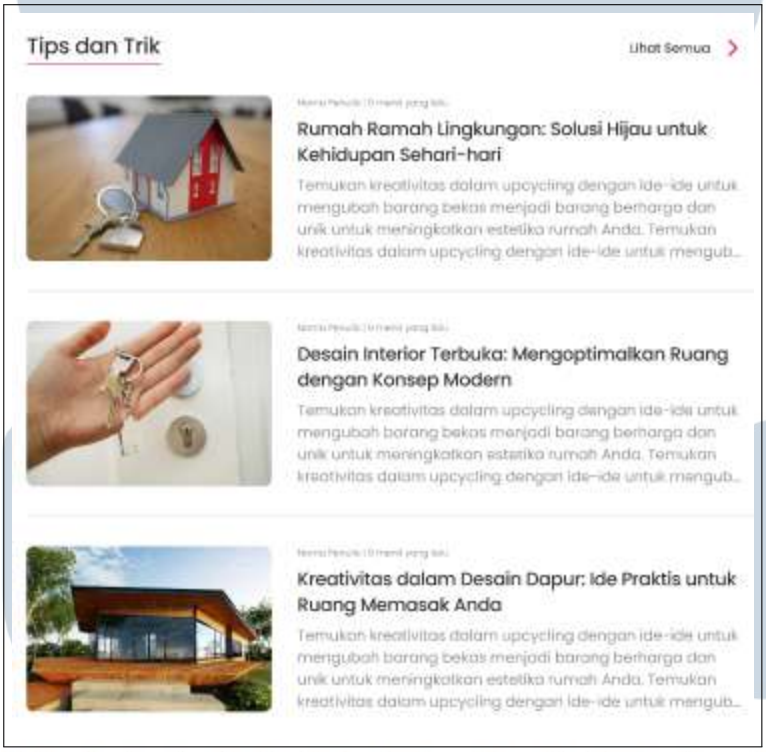
Gambar 3.6. Desain susunan *grid*

Desain dari susunan pratinjau artikel ini dapat dilihat pada Gambar 3.6. Tata letak dalam susunan ini dibuat menggunakan beberapa komponen dari *library* Ant Design, yaitu ”Grid” sebagai pembungkus komponen,

”Row” untuk membuat baris, dan ”Col” untuk membuat kolom di dalam sebuah komponen ”Row” . Setiap pratinjau artikel ditampilkan di dalam sebuah komponen ”Card” dari *library* Ant Design. Di dalam setiap ”Card”, ditampilkan gambar utama artikel, judul artikel, nama penulis, dan tanggal publikasi artikel. Pembaca dapat membuka halaman artikel dan membaca konten dari artikel tersebut dengan menekan elemen ”Card” dari artikel yang ingin dibaca.

- Susunan daftar/*list*

  Pada susunan daftar/*list*, sekelompok pratinjau artikel ditampilkan dalam susunan daftar/*list* dimana pratinjau artikel disusun di atas satu sama lain.



Gambar 3.7. Desain susunan daftar/*list*

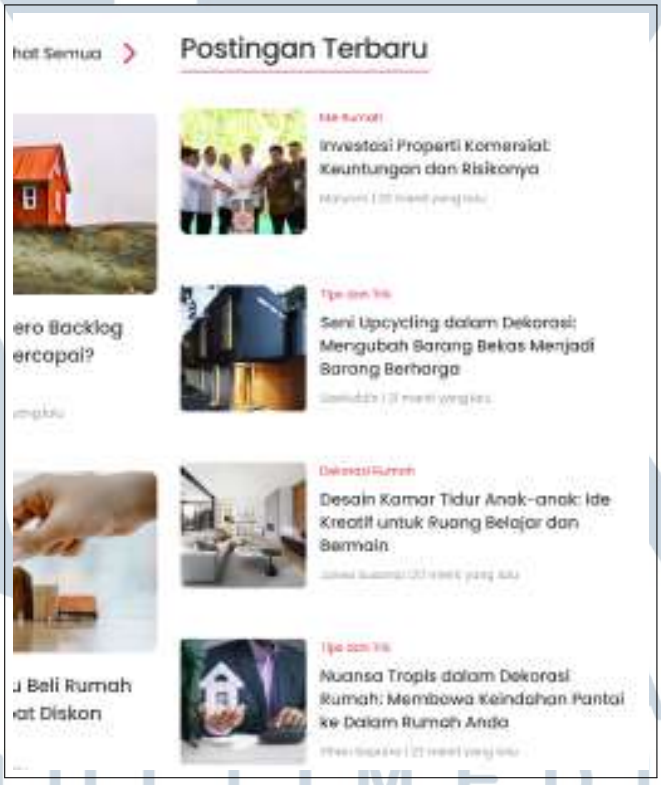
Desain dari susunan pratinjau artikel ini dapat dilihat pada Gambar 3.7. Tata letak dalam susunan ini dibuat menggunakan komponen ”List” dari *library* Ant Design. Setiap pratinjau artikel ditampilkan di dalam sebuah komponen ”Space” dengan properti *direction* vertikal dari *library* Ant Design. Di dalam setiap ”Space”, ditampilkan gambar utama artikel, nama penulis, tanggal publikasi artikel, judul artikel, dan sebagian dari

konten artikel. Pembaca dapat membuka halaman artikel dan membaca konten dari artikel tersebut dengan menekan gambar utama atau judul dari artikel yang ingin dibaca.

Pada setiap pengelompokan pratinjau artikel berdasarkan kategori, nama dari kategori dari artikel-artikel yang ditampilkan dapat dilihat pada sebelah kiri di atas susunan pratinjau artikel. Selain itu, terdapat tombol ”Lihat Semua” dengan ikon ”RightOutlined” dari *library* Ant Design pada sebelah kanan di atas susunan pratinjau artikel. Pembaca dapat membuka halaman kategori dengan menekan nama kategori atau menekan tombol ”Lihat Semua”.

4. **Daftar artikel terbaru**

   Elemen ini menampilkan artikel-artikel terbaru dari RE/MAX Blog berdasarkan tanggal publikasi artikel.



Gambar 3.8. Desain daftar artikel terbaru

Pratinjau artikel pada elemen ini disusun seperti susunan daftar/*list* yang digunakan pada pengelompokkan pratinjau artikel yang disusun secara daftar. Seperti susunan daftar/*list*, elemen ini juga menggunakan komponen ”List”

dari *library* Ant Design. Pratinjau-pratinjau artikel pada elemen ini diurutkan berdasarkan tanggal publikasi dari setiap artikel yang ditampilkan pada daftar. Pada daftar artikel terbaru, setiap pratinjau artikel menampilkan kategori utama artikel, judul artikel, nama penulis, dan tanggal publikasi dari artikel. Artikel yang paling baru diletakkan pada urutan pertama dan semakin bawah urutan artikel berada, semakin lawas/lama artikel tersebut.

5. **Daftar artikel terpopuler**

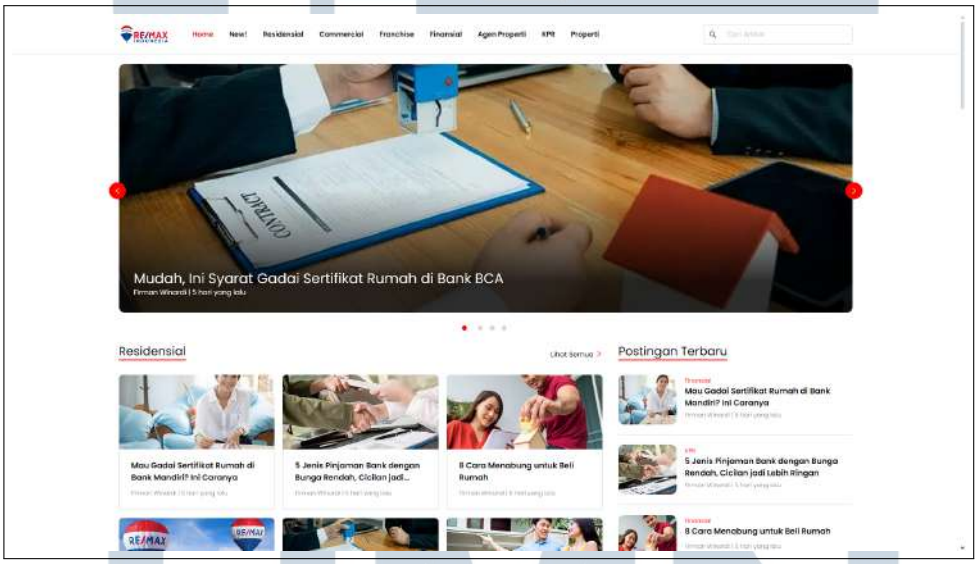
   Elemen ini menampilkan artikel-artikel terpopuler dari RE/MAX Blog berdasarkan jumlah pengunjung dari halaman artikel.

Gambar 3.9. Desain daftar artikel terpopuler

Pratinjau artikel pada elemen ini disusun seperti susunan daftar/*list* yang digunakan pada pengelompokkan pratinjau artikel yang disusun secara daftar. Seperti susunan daftar/*list*, elemen ini juga menggunakan komponen ”List” dari *library* Ant Design. Pratinjau-pratinjau artikel pada elemen ini diurutkan berdasarkan jumlah pengunjung dari setiap artikel yang ditampilkan pada daftar. Pada daftar artikel terpopuler, setiap pratinjau menampilkan urutan popularitas berdasarkan jumlah pengunjung, kategori utama artikel, dan judul artikel. Artikel yang paling populer/jumlah pengunjung terbanyak diletakkan

pada urutan pertama dan semakin bawah urutan artikel berada, semakin sedikit jumlah pengujungnya.

Beberapa dari elemen utama pada halaman utama ini digunakan kembali pada halaman RE/MAX Blog lainnya. Elemen bar navigasi, daftar artikel terbaru, daftar artikel terpopuler, serta tampilan susunan *grid* dan daftar/*list* digunakan berulang kali pada halaman-halaman lain yang ada pada proyek RE/MAX Blog. Pembuatan dan penggunaan elemen berulang kali diterapkan dengan tujuan untuk mempermudah dan mempercepat proses pembangunan halaman *web* lainnya pada proyek RE/MAX Blog.



Gambar 3.10. Halaman utama *web blog* RE/MAX Indonesia

Hasil dari pembangunan halaman utama RE/MAX Blog ditampilkan pada Gambar 3.10. Untuk menampilkan berbagai pratinjau artikel pada halaman *web* RE/MAX Blog, diperlukan pemanggilan ke API *backend* untuk mendapatkan data-data yang ingin ditampilkan pada *frontend web* RE/MAX Blog dari *database*. Pemanggilan API *backend* dilakukan di dalam fungsi ”getServerSideProps” yang didapat dari *framework* Next.js. Fungsi ini membuat semua proses di dalamnya dijalankan pada sisi server. Pada proyek ini, semua pemanggilan API *backend* dilakukan *server-side* dan *browser* pembaca tidak melakukan pemanggilan API sama sekali.

Pemanggilan API *backend* dilakukan pada urutan tertentu agar data yang didapat dalam bentuk yang terstruktur. Urutan pemanggilan API *backend* pada halaman utama RE/MAX Blog adalah:

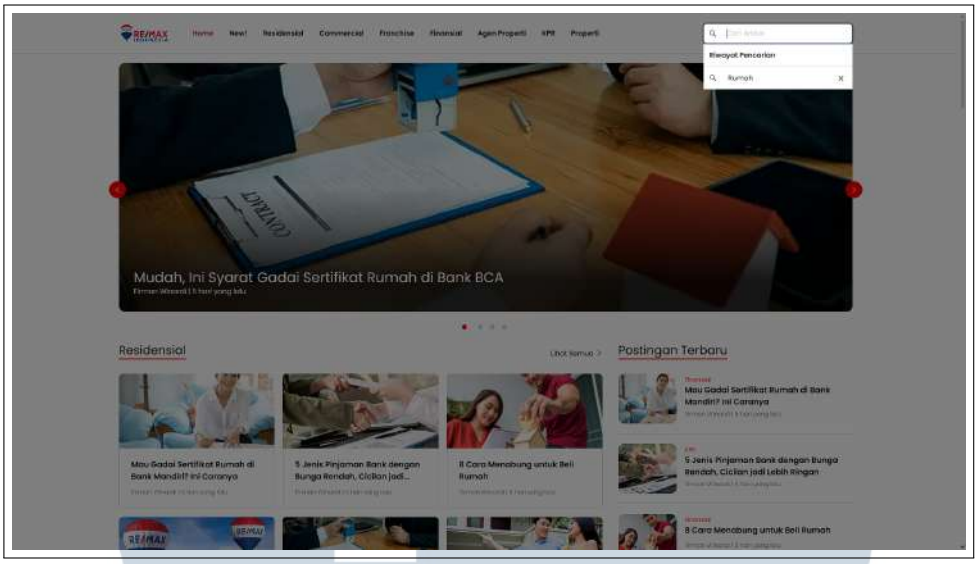
1. *Endpoint* ”/master-article-categories”
   Pemanggilan API ini dilakukan untuk mendapat semua data kategori utama. Contoh data yang didapat dari *endpoint* ini adalah nama kategori utama seperti ”Properti”, ”Tips dan Trik”, ”Dekorasi Rumah”, dan ”Ide rumah” beserta data pelengkap seperti *id* kategori, deskripsi, dan subkategori dari setiap kategori utama.

2. *Endpoint* ”/articles/public”
   Setelah mendapat data semua kategori utama, *id* kategori dari setiap data kategori utama yang didapat akan digunakan untuk mendapatkan data dari semua artikel yang ada pada kategori utama tersebut. *Id* kategori utama digunakan pada *params* pemanggilan *endpoint* ini agar *backend* mengetahui data artikel mana saja yang perlu dikirimkan ke *frontend*. Pemanggilan *endpoint* ini akan memberikan data artikel yang sudah dikelompokkan berdasarkan kategori utama dari setiap artikel. Data yang didapat dari *endpoint* ini ditampilkan pada elemen pratinjau artikel per kategori.

3. *Endpoint* ”articles/public?isHighlight=true” Pemanggilan API ini dilakukan untuk mendapat semua data artikel yang di-*highlight*. Data dari *endpoint* ini digunakan pada elemen *carousel*.

4. *Endpoint* ”articles/public?sort=createdAt:DESC” Pemanggilan API ini dilakukan untuk mendapat semua data artikel yang sudah diurutkan berdasarkan tanggal pembuatan/publikasi yang paling baru hingga tanggal publikasi yang paling lawas/lama. Data dari *endpoint* ini digunakan pada elemen daftar artikel terbaru.

5. *Endpoint* ”articles/public?sort=popularity:DESC” Pemanggilan API ini dilakukan untuk mendapat semua data artikel yang sudah diurutkan berdasarkan jumlah pengunjung dari setiap artikel, dari jumlah paling tinggi hingga paling rendah. Data dari *endpoint* ini digunakan pada elemen daftar artikel terpopuler.



Gambar 3.11. Fitur cari artikel pada *web blog* RE/MAX Indonesia

Pada halaman utama, pembaca dapat melakukan pencarian artikel menggunakan kolom *input* yang ada pada sebelah kanan bar navigasi. Tampilan dari fitur pencarian pada halaman *blog* dapat dilihat pada Gambar 3.11. Pembaca juga dapat melihat, menghapus, dan mencari ulang riwayat pencarian yang mereka lakukan sebelumnya. Proses pencarian pada halaman *web* RE/MAX Blog adalah sebagai berikut:

1. Halaman *web* menggunakan fitur *local storage* pada *browser* untuk menyimpan dan mengambil riwayat kata kunci pencarian dari pembaca. Ketika pembaca menekan kolom *input* pencarian, halaman *web* akan menampilkan riwayat pencarian. Kata kunci yang ditampilkan pada riwayat tersebut diambil dari *local storage* pada *browser* pembaca.

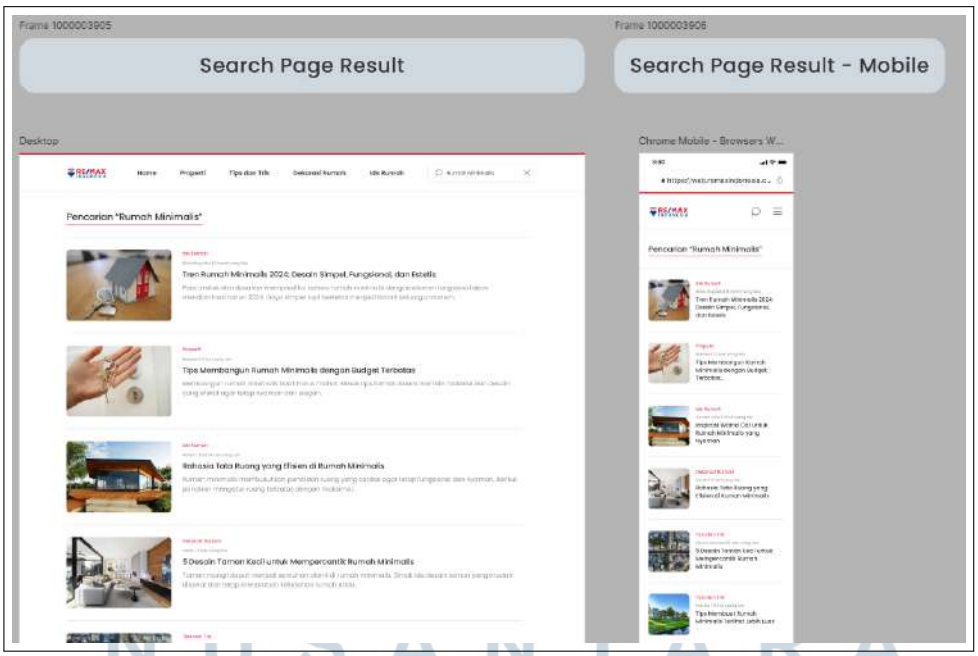
2. Setelah pembaca memasukkan kata kunci dan menekan ikon pencarian atau pembaca memilih kata kunci yang ada pada riwayat, halaman *web* akan menyimpan kata kunci tersebut pada *local storage*. Jika kata kunci berasal dari riwayat pencarian, maka kata kunci tersebut akan dihapus terlebih dahulu dari *local storage* sebelum ditambahkan kembali.

3. Pembaca akan diarahkan ke halaman pencarian dengan kata kunci yang mereka masukkan atau pilih ditambahkan pada bagian *query* tautan. Sebagai contoh, jika pembaca memasukkan kata kunci ”Rumah”, tautan pada *browser* akan diberikan tambahan ”?q=Rumah”.

4. Saat *browser* pembaca membuka halaman pencarian dengan *query* pada tautan, *server* akan memproses pencarian berdasarkan kata kunci yang ada pada *query* tautan. Proses pencarian ini dimulai dengan mengambil kata kunci pencarian dari *query* tautan dan menggunakan kata kunci tersebut untuk memanggil *endpoint* API.
5. Sebagai contoh, jika kata kunci yang diterima adalah ”Rumah”, maka *endpoint* yang dipanggil adalah ”/articles/public?search=Rumah”. *Endpoint* tersebut akan memberikan data artikel dengan judul yang memiliki kata ”Rumah”.
6. Halaman *web* menampilkan data yang didapat dari *endpoint* pada halaman pencarian yang sedang dibuka oleh *browser* pembaca.

B. Halaman hasil pencarian

Halaman ini ditampilkan ketika pembaca melakukan pencarian pada halaman *web* RE/MAX Blog. Ketika pembaca diarahkan ke halaman ini, halaman *web* akan menampilkan artikel-artikel hasil pencarian.



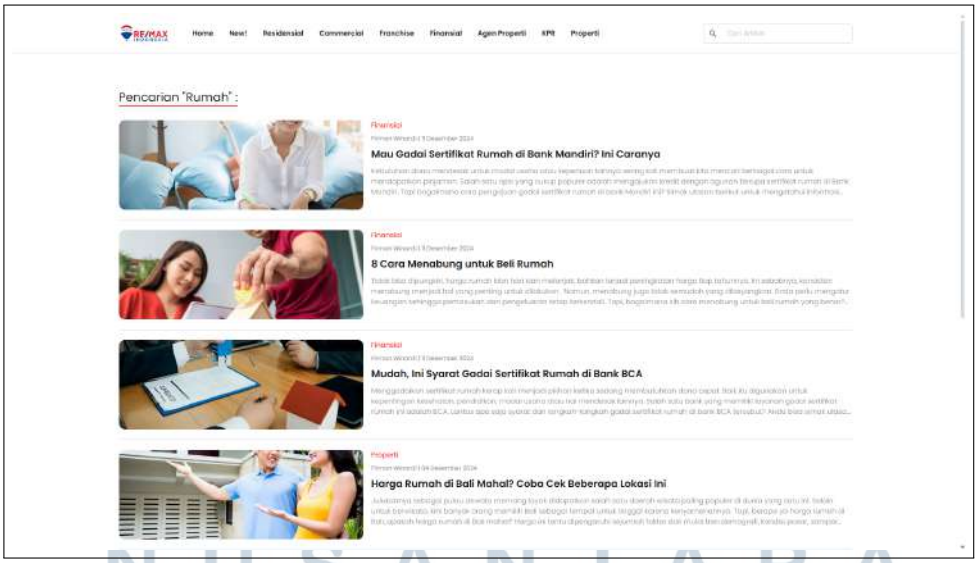
Gambar 3.12. Desain halaman hasil pencarian pada RE/MAX Blog

Desain dari halaman hasil pencarian RE/MAX Blog pada *browser desktop*

dan pada *browser mobile* dapat dilihat pada Gambar 3.12. Halaman ini dapat dibagi menjadi beberapa elemen utama, yaitu:

1. **Bar navigasi**
   Elemen ini merupakan komponen yang sudah dibuat sebelumnya untuk halaman utama RE/MAX Blog dengan memodifikasi komponen bar navigasi dari halaman *web* utama RE/MAX Indonesia. Komponen ini digunakan berulang kali pada semua halaman RE/MAX Blog.

2. **Daftar hasil pencarian**
   Elemen ini dibuat menggunakan susunan daftar/*list* yang sudah dibuat sebelumnya untuk elemen pratinjau artikel per kategori pada halaman utama RE/MAX Blog. Penggunaan komponen ”List” dari *library* Ant Design memungkinkan untuk menampilkan informasi mengenai artikel yang didapat dari kata kunci pencarian dengan lebih detail. Pada daftar hasil pencarian ini, ditampilkan kategori utama artikel, nama penulis, waktu publikasi, judul artikel, dan sebagian dari konten artikel. Di atas daftar artikel yang ditemukan dari hasil pencarian, terdapat tulisan ”Pencarian” dan kata kunci. Sebagai contoh jika kata kunci adalah ”Rumah”, maka di atas daftar artikel akan tertulis ”Pencarian ’Rumah’”.



Gambar 3.13. Halaman hasil pencarian RE/MAX Blog

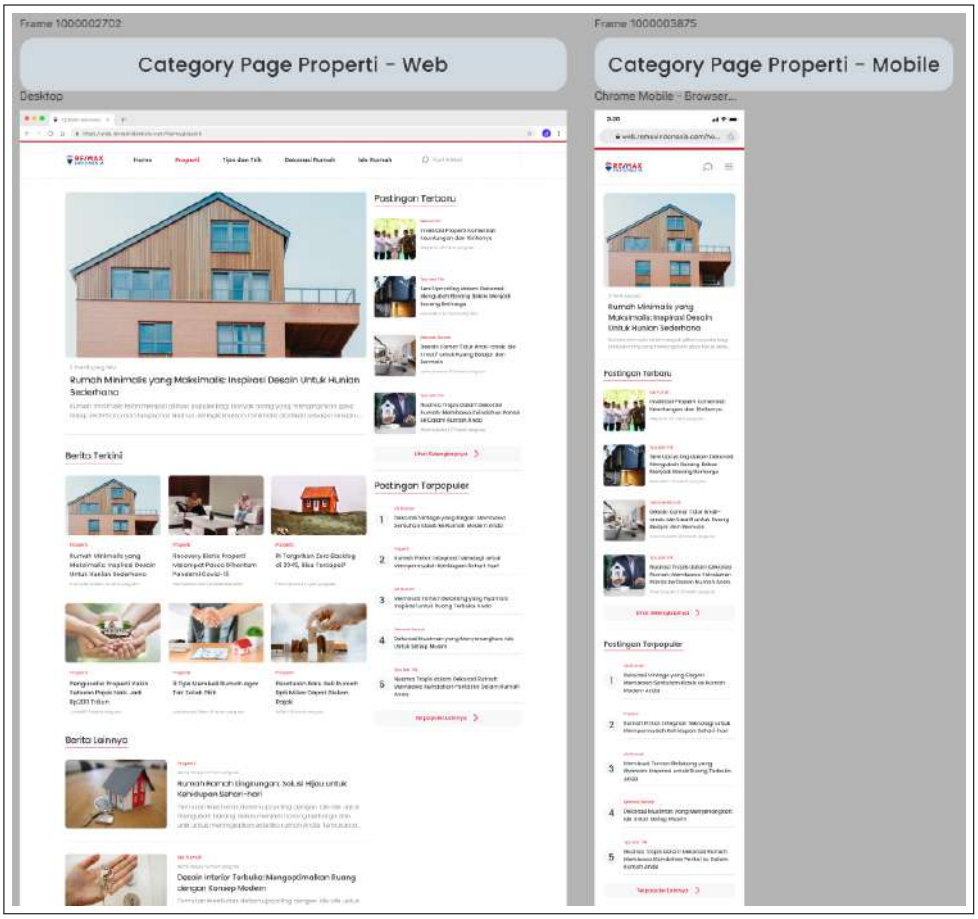
Hasil dari pembangunan halaman hasil pencarian RE/MAX Blog dapat dilihat pada Gambar 3.13. Untuk menampilkan artikel-artikel yang sesuai dengan

kata kunci pencarian, diperlukan pemanggilan ke API *backend* untuk mendapatkan data yang ingin ditampilkan. Sama seperti pada halaman utama, pemanggilan API backend dilakukan di dalam fungsi ”getServerSideProps” yang didapat dari framework Next.js. Semua halaman pada *web* RE/MAX Blog melakukan pemanggilan API di dalam fungsi tersebut agar semua pemanggilan API untuk mendapatkan data dijalankan pada sisi *server* dan tidak dilakukan oleh *browser* pembaca.

Pemanggilan API *backend* yang dilakukan pada halaman hasil pencarian adalah ke *endpoint* ”/articles/public?search=[katakunci]”. Bagian ”[katakunci]” pada ”?search=[katakunci]” diganti dengan kata kunci pencarian yang didapat dari bagian *query* dari tautan halaman. Sebagai contoh, ketika pembaca memasukkan ”Warna” sebagai kata kunci pencarian, tautan halaman hasil pencarian akan memiliki tambahan ”?q=Warna”. Halaman hasil pencarian akan mengambil kata kunci tersebut dan memanggil *endpoint* ”/articles/public?search=Warna” untuk mendapatkan data artikel yang memiliki kata ”Warna” pada judul. Setelah mendapatkan artikel-artikel yang sesuai dengan kata kunci, halaman hasil pencarian akan menampilkan artikel-artikel tersebut pada elemen daftar hasil pencarian.

C. Halaman kategori

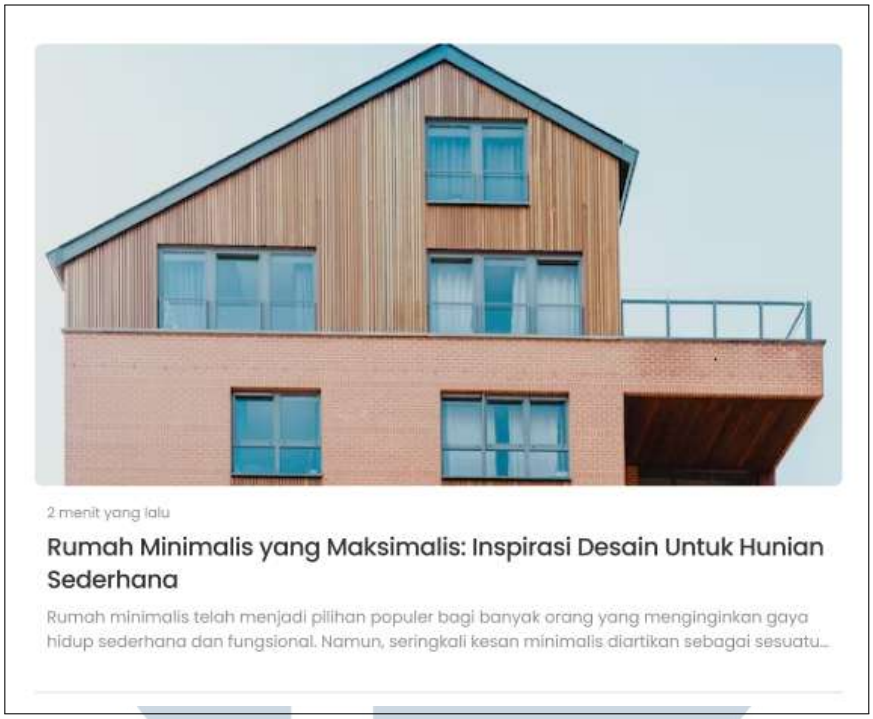
Halaman ini adalah halaman khusus untuk sebuah kategori utama. Pada halaman ini hanya ditampilkan artikel-artikel yang termasuk ke dalam kategori utama yang dibuka oleh pembaca. Halaman ini diakses melalui dua cara, yaitu menekan salah satu tombol nama kategori pada bar navigasi dan menekan nama kategori atau tombol ”Lihat Semua” pada tampilan susunan pratinjau artikel yang dikelompokkan berdasarkan kategori pada halaman utama.



Gambar 3.14. Desain halaman kategori RE/MAX Blog

Desain yang dibuat oleh tim UI/UX perusahaan untuk halaman kategori untuk *browser desktop* dan *browser mobile* dapat dilihat pada Gambar 3.14. Halaman ini dapat dibagi menjadi beberapa elemen utama, yaitu:

1. **Bar navigasi**
   Elemen ini merupakan komponen yang sudah dibuat sebelumnya untuk halaman utama RE/MAX Blog dengan memodifikasi komponen bar navigasi dari halaman *web* utama RE/MAX Indonesia. Komponen ini digunakan berulang kali pada semua halaman RE/MAX Blog.
2. **Pratinjau artikel terbaru**
   Pada halaman ini, artikel terbaru dari kategori utama yang sedang dibuka ditampilkan pada bagian atas halaman.

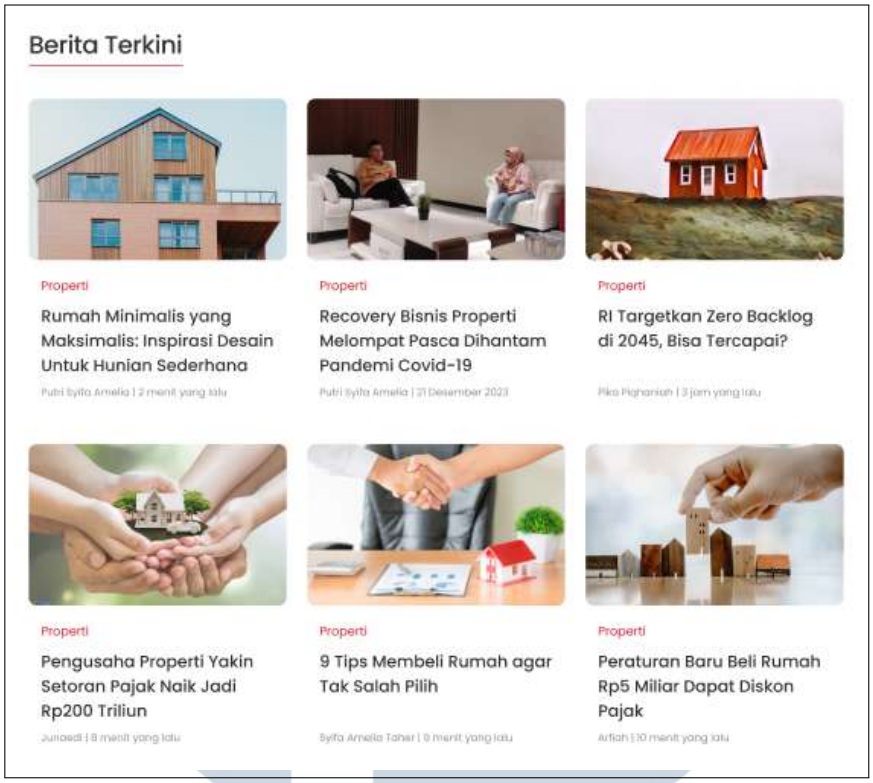


Gambar 3.15. Desain pratinjau artikel terbaru RE/MAX Blog

Desain dari elemen ini dapat dilihat pada Gambar 3.15. Artikel yang paling baru dari kategori utama ini ditampilkan menggunakan komponen ”Space” dari *library* Ant Design. Pada elemen ini, ditampilkan gambar utama artikel, waktu publikasi artikel, judul artikel, dan sebagian dari konten artikel. Pembaca dapat mengunjungi halaman dari artikel tersebut dengan menekan gambar utama artikel atau menekan judul artikel.

3. ***Grid* artikel terkini**

   Artikel-artikel lainnya ditampilkan di bawah pratinjau artikel paling baru pada halaman kategori. Pada elemen ini, ditampilkan artikel-artikel yang termasuk ke dalam kategori utama yang sedang dibuka yang diurutkan berdasarkan tanggal publikasi.

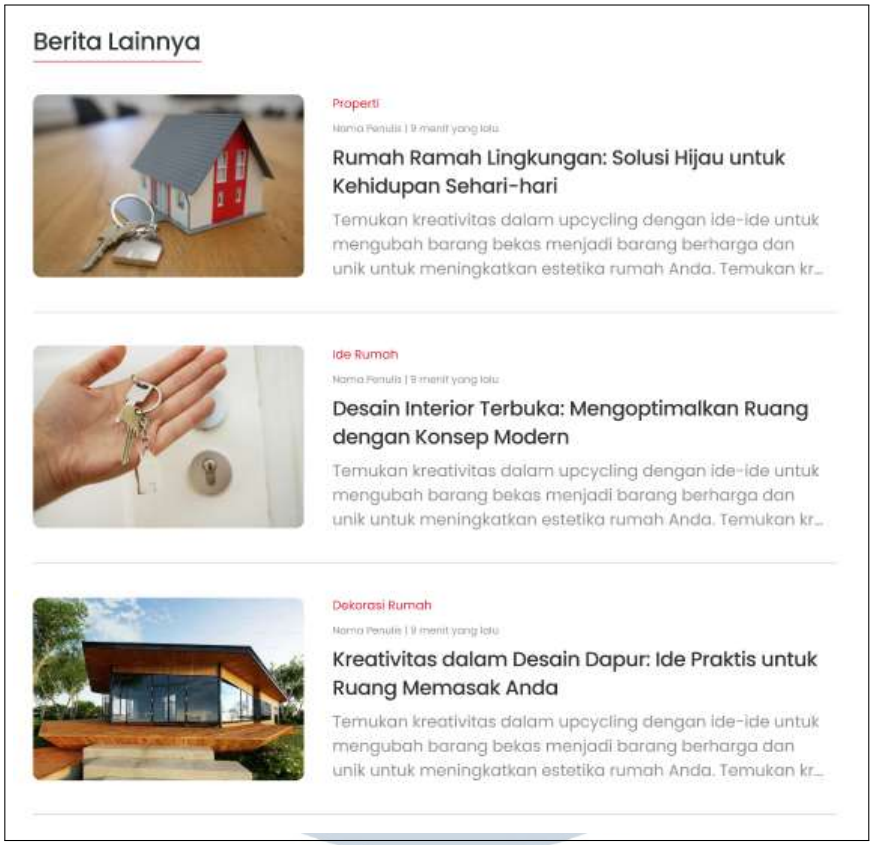


Gambar 3.16. Desain *grid* berita terkini RE/MAX Blog

Desain dari *grid* berita terkini dapat dilihat pada Gambar 3.16. Elemen ini dibuat menggunakan susunan *grid* yang sudah dibuat sebelumnya untuk pratinjau artikel per kategori pada halaman utama. Artikel-artikel yang ditampilkan pada elemen ini hanya artikel yang termasuk ke dalam kategori utama yang sedang dibuka dan diurutkan berdasarkan tanggal publikasi yang paling baru hingga tanggal publikasi yang paling lama/lawas. Di atas susunan *grid* berita terkini, diberikan tulisan ”Berita Terkini”.

4. **Daftar berita lainnya**

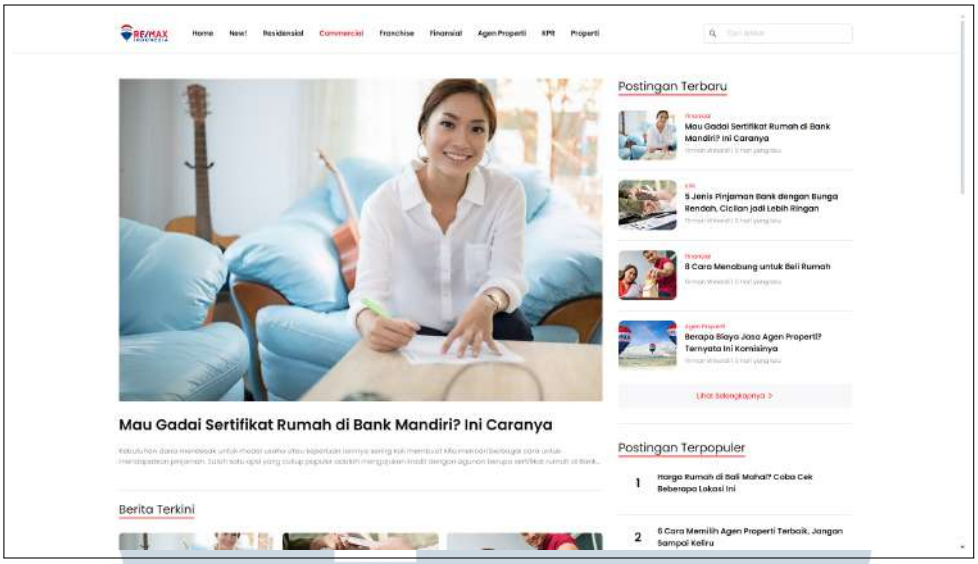
   Selain artikel terbaru dan beberapa artikel terkini, artikel-artikel lainnya ditampilkan dalam susunan daftar.



Gambar 3.17. Desain daftar berita lainnya RE/MAX Blog

Desain dari daftar berita lainnya dapat dilihat pada Gambar 3.17. Elemen ini dibuat menggunakan susunan daftar/*list* yang sudah dibuat sebelumnya untuk pratinjau artikel per kategori pada halaman utama. Artikel-artikel yang ditampilkan pada elemen ini hanya artikel yang termasuk ke dalam kategori utama yang sedang dibuka. Di atas susunan daftar berita lainnya, diberikan tulisan ”Berita Lainnya”.

5. **Daftar berita terbaru dan berita terpopuler**
   Pada bagian kanan halaman, ditampilkan elemen daftar berita terbaru dan daftar berita terpopuler yang sudah dibuat sebelumnya menggunakan susunan daftar/*list*. Kedua elemen ini menampilkan artikel terbaru dan terpopuler dari semua artikel yang ada pada RE/MAX Blog, tidak hanya dari kategori utama yang sedang dibuka.



Gambar 3.18. Halaman kategori RE/MAX Blog

Hasil pembangunan halaman kategori pada RE/MAX Blog dapat dilihat pada Gambar 3.18. Fungsi utama dari halaman ini adalah menampilkan artikel-artikel yang termasuk ke dalam suatu kategori utama. Oleh karena itu, pemanggilan API *backend* dilakukan untuk mendapatkan data artikel-artikel yang sesuai. Contohnya pada halaman kategori ”Ide Rumah”, data artikel-artikel yang didapatkan dari *endpoint* API hanyalah artikel-artikel yang termasuk ke dalam kategori utama ”Ide Rumah”.

Proses pengambilan data melalui API *backend* yang dilakukan pada halaman ini adalah sebagai berikut:

1. Ketika pembaca membuka halaman kategori, nama dari kategori utama yang mereka pilih melalui tombol pada bar navigasi maupun tombol ”Lihat Semua” pada elemen pengelompokkan artikel per kategori akan menjadi bagian dari tautan halaman kategori. Sebagai contoh ketika pembaca membuka halaman kategori ”Dekorasi Rumah”, maka tautan dari halaman akan memiliki ”/dekorasi-rumah”. Tambahan nama kategori pada tautan ini disebut sebagai *slug*.
2. Halaman kategori akan mengambil *slug* dari tautan halaman dan menggunakan *slug* tersebut untuk melakukan pemanggilan API.
3. *Endpoint* ”master-article-categories” dipanggil untuk mendapatkan semua data kategori utama.

4. Data yang didapat akan dilakukan pencarian berdasarkan *slug* yang diambil dari tautan halaman. Ketika data kategori utama dengan *slug* yang sesuai berhasil ditemukan, *id* kategori dari data tersebut akan diambil untuk melakukan pemanggilan API selanjutnya.

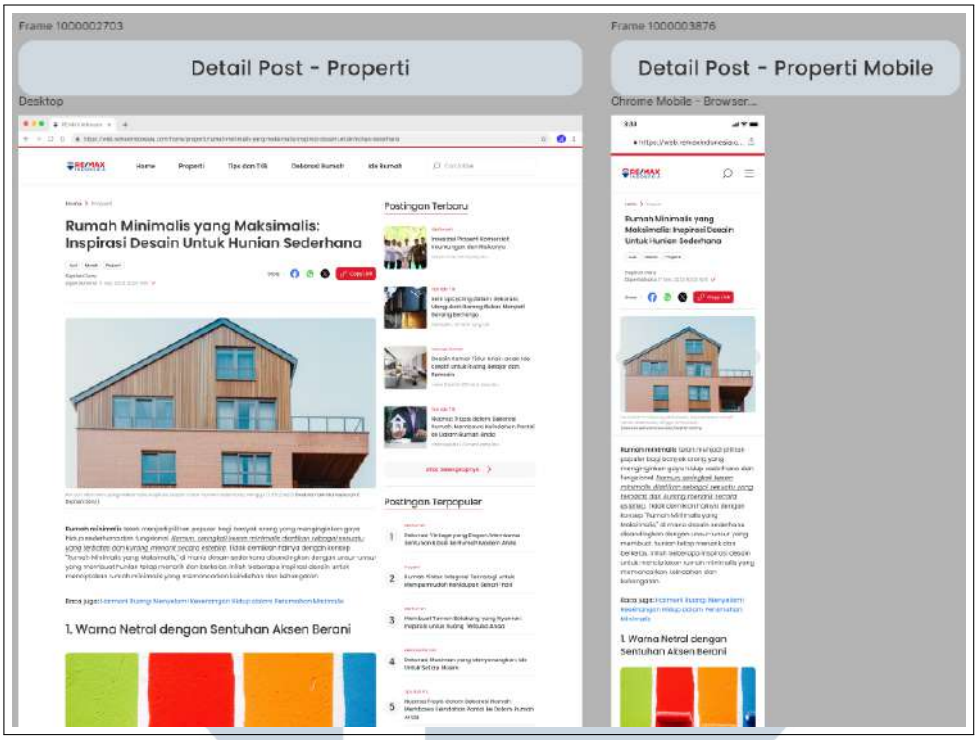
5. *Endpoint* ”articles/public?parentCategoryIds=[*id* kategori]” dipanggil untuk mendapatkan semua artikel yang termasuk ke dalam kategori utama yang sedang dibuka. Bagian ”[*id* kategori]” pada *endpoint* diganti dengan *id* kategori yang ditemukan pada tahap sebelumnya.

6. Data artikel yang didapat dari *endpoint* tersebut sudah diurutkan berdasarkan tanggal publikasi. Sehingga pada *frontend* halaman, beberapa artikel terbaru dipisahkan untuk ditampilkan pada elemen berita terkini dan artikel-artikel lainnya ditampilkan pada elemen berita lainnya.

7. *Endpoint* ”articles/public?sort=createdAt:DESC” dan endpoint ”articles/public?sort=popularity:DESC” dipanggil untuk mendapatkan data artikel yang ditampilkan pada elemen daftar artikel terbaru dan elemen daftar artikel terpopuler.

D. Halaman artikel

Halaman ini adalah halaman dimana pembaca dapat melihat dan membaca konten dari artikel yang dikunjungi. Halaman ini ditampilkan ketika pembaca memilih artikel dari berbagai pratinjau artikel yang ada pada halaman-halaman lainnya, seperti halaman utama, halaman kategori, maupun halaman hasil pencarian.



Gambar 3.19. Desain halaman artikel RE/MAX Blog

Desain untuk halaman artikel pada *browser desktop* dan *browser mobile* dapat dilihat pada Gambar 3.19. Halaman ini dapat dibagi menjadi beberapa elemen utama, yaitu:

1. **Bar navigasi**

   Seperti pada halaman-halaman sebelumnya, bar navigasi menggunakan komponen hasil modifikasi dari halaman utama *web* RE/MAX Indonesia yang dibuat pada halaman utama RE/MAX Blog.

2. ***Header* artikel**

   Elemen *header* artikel ini menampilkan informasi seputar artikel yang sedang dibuka. Informasi tersebut meliputi *breadcrumbs*, judul artikel, nama penulis, tanggal publikasi, dan tombol-tombol untuk membagikan tautan artikel.

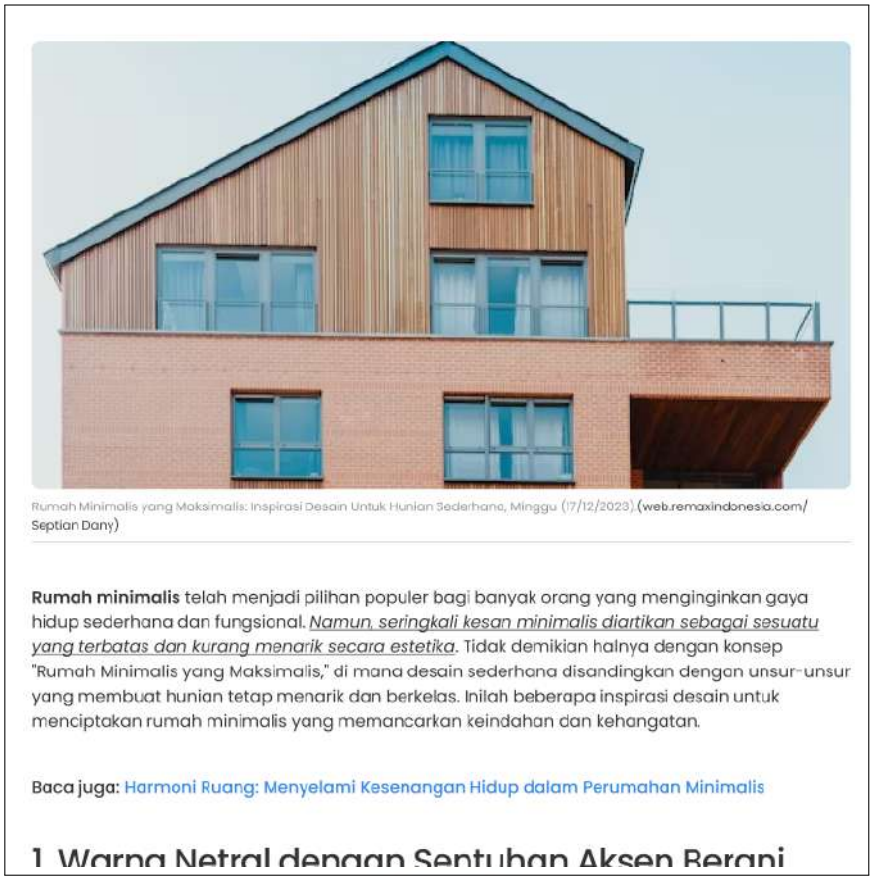
Gambar 3.20. Desain *header* artikel RE/MAX Blog

Desain dari elemen *header* artikel dapat dilihat pada Gambar 3.20. Elemen ini dibuat menggunakan komponen ”Space” dari *library* Ant Design. Di dalam komponen ”Space”, terdapat *breadcrumbs* pada bagian paling atas. *Breadcrumbs* adalah pembantu navigasi sekunder yang memberi pembaca informasi mengenai lokasi mereka dalam *web* RE/MAX Blog. Struktur dari *breadcrumbs* adalah tulisan ”Home”, ikon ”RightOutlined” dari *library* Ant Design, dan tulisan nama kategori utama. Contoh pada desain adalah tulisan ”Home”, ikon ”RightOutlined” dari *library* Ant Design, dan tulisan ”Properti”. Tulisan ”Properti” pada desain *breadcrumbs* akan mengarahkan pembaca ke halaman kategori ”Properti” ketika ditekan oleh pembaca.

Di bawah *breadcrumbs*, ditampilkan judul artikel, *tag* artikel, nama penulis, dan tanggal publikasi yang disusun secara vertikal dan diletakkan dengan posisi rata kiri. Pada bagian kanan *header*, diberikan beberapa tombol yang memfasilitasi pembaca untuk membagikan tautan artikel ke berbagai media sosial. Ikon media sosial yang digunakan pada tombol-tombol tersebut didapat dari *library* next-share.

3. **Konten artikel**

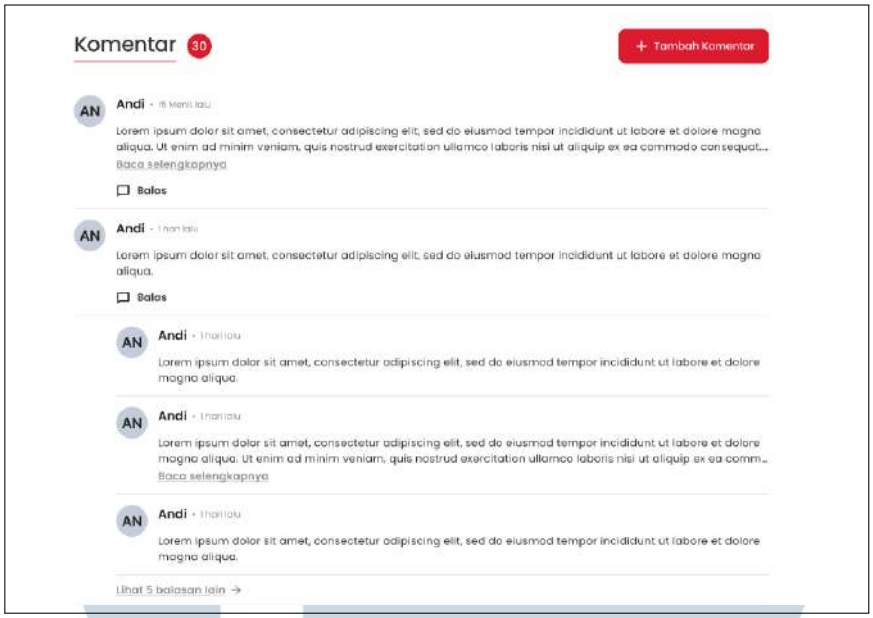
   Di bawah *header* artikel, ditampilkan konten dari artikel. Konten yang dimaksud termasuk teks, gambar, video, maupun tautan ke halaman *web* lain.



Gambar 3.21. Desain konten artikel RE/MAX Blog

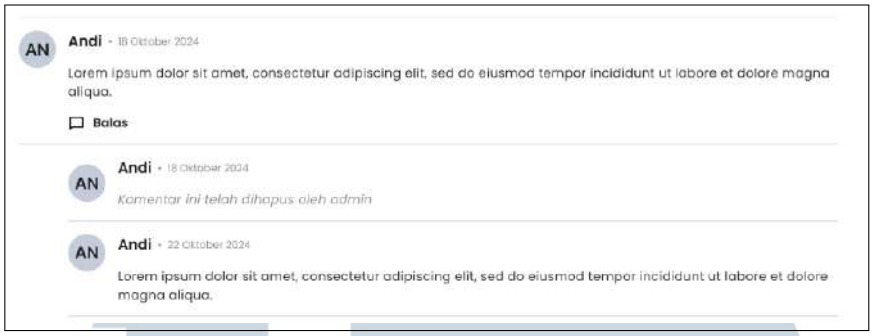
Desain dari elemen konten artikel dapat dilihat pada Gambar 3.21. Elemen ini terdiri dari gambar utama artikel dan pembungkus konten. Di bawah gambar utama artikel, ditampilkan *caption* dari gambar utama tersebut. Konten dari artikel ditampilkan di dalam pembungkus yang dibuat di bawah *caption* gambar utama. Pembungkus ini dibuat menggunakan subkomponen ”Paragraph” dari komponen ”Typography” *library* Ant Design. Data dari komponen artikel didapat dalam bentuk HTML dan di-*set* langsung ke dalam komponen ”Paragraph” melalui properti ”dangerouslySetInnerHTML”.

4. **Kolom komentar**
   Pada akhir konten artikel, disediakan kolom komentar dimana pembaca dapat menambahkan komentar terhadap artikel yang mereka sudah baca.



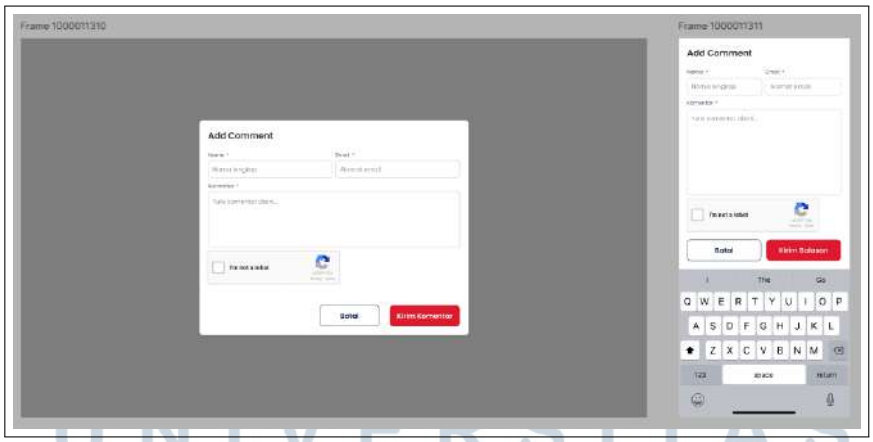
Gambar 3.22. Desain kolom komentar artikel RE/MAX Blog

Desain dari kolom komentar artikel dapat dilihat pada Gambar 3.22. Kolom komentar ini menampilkan semua komentar yang diberikan oleh pembaca-pembaca sebelumnya beserta dengan balasan yang diberikan kepada komentar-komentar tersebut. Kolom komentar dibuat menggunakan komponen ”List” dari *library* Ant Design. Di atas sebelah kiri tampilan daftar komentar, dituliskan ”Komentar” beserta dengan total jumlah komentar. Pada bagian kanan terdapat tombol untuk menambahkan komentar baru terhadap artikel. Tombol tersebut dibuat menggunakan komponen ”Button” dari *library* Ant Design dengan ikon ”PlusOutlined” dan tulisan ”Tambah Komentar” di dalamnya.



Gambar 3.23. Desain komentar artikel RE/MAX Blog

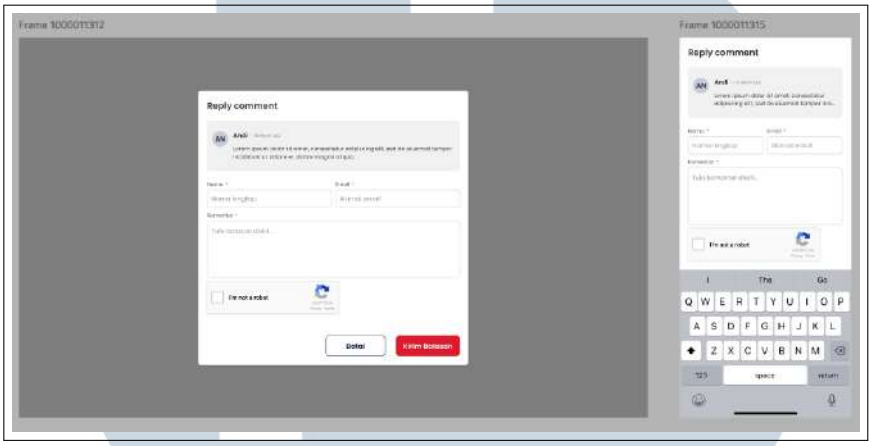
Desain dari tampilan setiap komentar dapat dilihat pada Gambar 3.23. Sebuah *item* komentar menampilkan inisial dari nama pembuat komentar dalam komponen ”Avatar” dari *library* Ant Design, nama pembuat komentar, tanggal komentar dibuat, dan isi komentar, serta tombol untuk membuat balasan menggunakan ikon ”ReplyIcon” yang dibuat oleh tim UI/UX dan tulisan ”Balas”. Pada setiap *item* komentar, terdapat juga daftar balasan terhadap komentar. Bentuk daftar balasan ini menggunakan desain yang sama dengan *item* komentar, tetapi dengan tambahan *margin* di sebelah kiri untuk membuat indentasi.



Gambar 3.24. Desain *modal* komentar artikel RE/MAX Blog

Ketika pembaca menekan tombol tambah komentar, halaman *web* akan menampilkan jendela dialog/*modal* untuk menambahkan komentar. Desain dari *modal* tersebut dapat dilihat pada Gambar 3.24. *Modal* penambahan komentar ini dibuat menggunakan komponen dari *library* Ant Design, yaitu ”Modal” dan ”Input” beserta komponen *custom* sebagai pembungkus fitur

verifikasi *captcha*. Pada bagian bawah *modal* terdapat dua tombol, yaitu tombol ”Batal” dan tombol ”Kirim Komentar”. Tombol ”Batal” digunakan untuk membatalkan pembuatan komentar dan tombol ”Kirim Komentar” digunakan untuk mengirimkan komentar ke artikel.



Gambar 3.25. Desain *modal* balasan komentar RE/MAX Blog

Pembaca dapat memberikan balasan terhadap komentar yang ada pada artikel dengan menekan tombol ”Balas” pada sebuah *item* komentar. Halaman akan menampilkan *modal* untuk membalas komentar ketika tombol ”Balas” ditekan oleh pembaca. Desain dari *modal* balas komentar dapat dilihat pada Gambar 3.25. *Modal* membalas komentar ini juga dibuat menggunakan komponen dari *library* Ant Design, yaitu ”Modal” dan ”Input” beserta komponen *custom* sebagai pembungkus fitur verifikasi *captcha*. Perbedaannya dengan *modal* komentar adalah pada *modal* balasan komentar, ditampilkan *item* komentar yang akan dibalas pada bagian atas *modal*. Pada bagian bawah *modal* terdapat dua tombol, yaitu tombol ”Batal” dan tombol ”Kirim Balasan”. Tombol ”Batal” digunakan untuk membatalkan pembuatan balasan dan tombol ”Kirim Balasan” digunakan untuk mengirimkan balasan ke komentar.

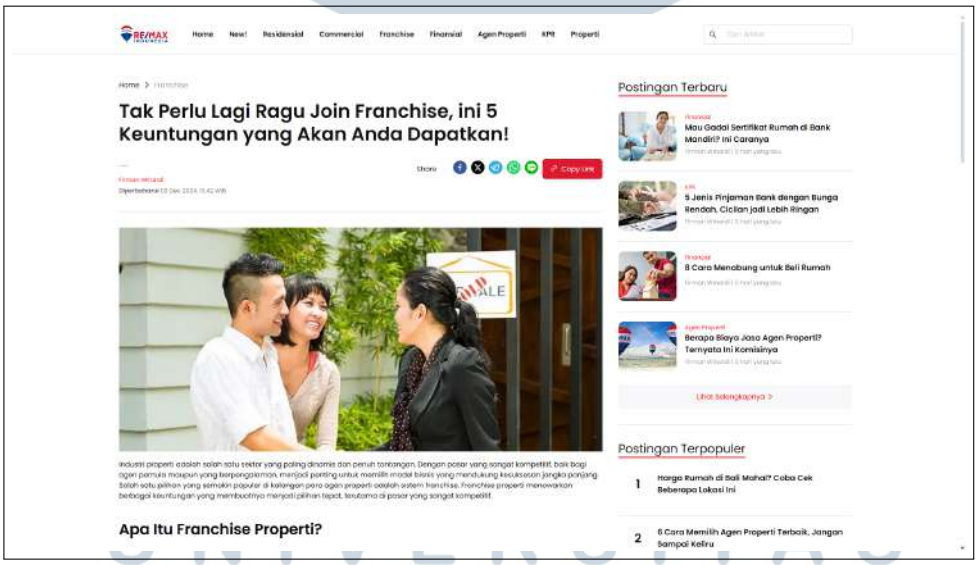
Pada *modal*, pembaca harus mengisi nama, email, dan isi dari komentar/balasan pada kolom-kolom *input* yang tersedia. Semua kolom *input* tersebut harus diisi sebelum komentar/balasan dapat dikirimkan dengan menekan tombol ”Kirim Komentar” atau ”Kirim Balasan”. Selain itu, tombol ”Kirim Komentar” dan tombol ”Kirim Balasan” hanya dapat ditekan apabila verifikasi *captcha* berhasil dilakukan oleh pembuat komentar atau balasan.

Verifikasi *captcha* pada *modal* komentar dan balasan ini menggunakan *library* react-simple-captcha.

Kolom komentar ini tidak menampilkan semua komentar secara langsung. Pada desain, jumlah awal komentar yang ditampilkan adalah 5 komentar. dan jumlah balasan awal yang ditampilkan pada setiap komentar adalah 3 balasan. Pembaca dapat menekan tombol ”Lihat Komentar Lain” untuk menambahkan jumlah komentar yang ditampilkan. Pembaca juga dapat menekan tombol ”Lihat Balasan Lain” untuk menambahkan jumlah balasan yang ditampilkan pada sebuah *item* komentar.

5. ***Grid* berita lainnya**

   Elemen ini dibuat menggunakan susunan *grid* yang sudah dibuat sebelumnya untuk pratinjau artikel per kategori pada halaman utama. Artikel-artikel yang ditampilkan pada elemen ini adalah artikel-artikel yang memiliki kategori utama yang sama dengan artikel yang sedang dibuka. Di atas susunan daftar berita lainnya, diberikan tulisan ”Berita Lainnya”.



Gambar 3.26. Halaman artikel RE/MAX Blog

Hasil pembangunan halaman artikel dapat dilihat pada Gambar 3.26. Halaman artikel berisi informasi mengenai artikel dan konten dari artikel itu sendiri. Informasi tersebut meliputi nama penulis, tanggal publikasi, dan opsi untuk membagikan tautan artikel tersebut ke berbagai media sosial. Konten dari artikel yang ditampilkan dapat berupa teks, gambar, tautan, maupun video. Setelah

membaca konten dari artikel yang dibuka, pembaca juga dapat melihat dan menulis komentar maupun balasan terhadap komentar lain. Pembaca juga dapat melihat artikel lain yang terkait dengan artikel yang mereka baca di bawah kolom komentar.

Ketika pembaca membuka halaman artikel, *frontend* akan melakukan pemanggilan ke API *backend*. *Endpoint* yang dipanggil adalah ”article-visitors” dengan *body* data yang berisi *id* artikel dan tautan halaman artikel. Pemanggilan *endpoint* ini bertujuan untuk menambah jumlah pengunjung dari artikel yang dibuka pada *database* melalui *backend*.

Untuk menampilkan konten artikel, halaman *web* melakukan beberapa pemanggilan API *backend*. Proses pemanggilan API pada halaman artikel adalah sebagai berikut:

1. Ketika pembaca membuka halaman artikel, judul dari artikel yang dibuka akan ditambahkan ke tautan halaman. Contoh jika judul artikel yang dibuka adalah ”Properti Murah dan Nyaman”, maka pada tautan halaman diberi tambahan ”/properti-murah-dan-nyaman”. Tambahan ini disebut sebagai *slug* dan digunakan untuk memanggil *endpoint* untuk mendapatkan data artikel.

2. *Endpoint* ”articles/check-slug?slug=[slug]” dipanggil untuk mendapatkan *id* artikel. Bagian ”[slug]” pada *endpoint* digantikan dengan *slug* yang didapatkan dari tautan halaman.

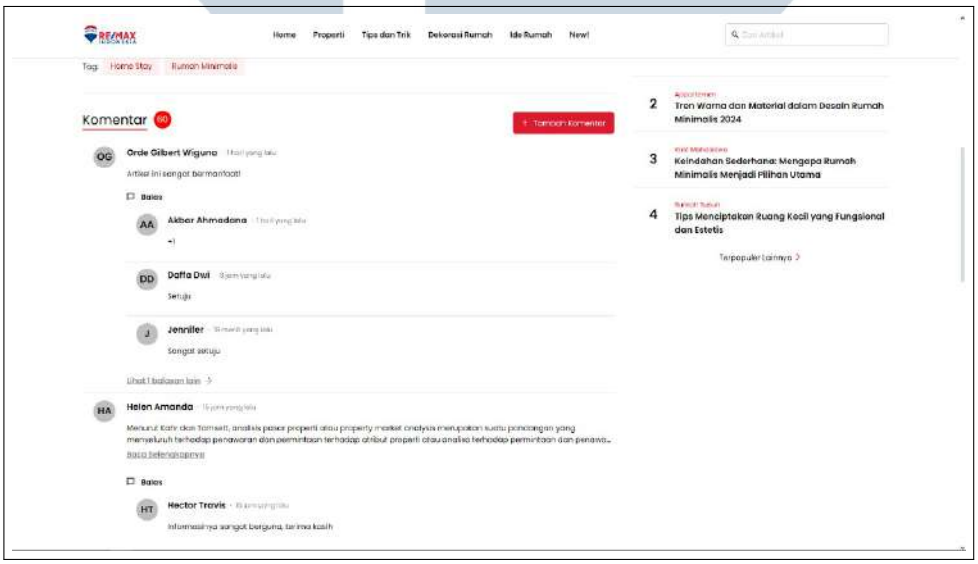
3. *Endpoint* ”articles/[*id*]/public” dipanggil untuk mendapatkan data artikel. Bagian ”[*id*]” pada *endpoint* digantikan dengan *id* artikel yang didapat pada tahap sebelumnya.

4. *Enpdoint* ”articles/public?excludeArticleIds=[*id*]&parentCategoryIds=[*id* kategori]” dipanggil untuk mendapatkan artikel lain yang memiliki kategori utama yang sama dengan artikel yang sedang dibuka. Bagian ”[*id*]” pada *endpoint* diganti dengan *id* artikel yang sedang dibuka dan bagian ”[*id* kategori]” diganti dengan *id* kategori utama yang didapat dari data artikel yang sedang dibuka. Pemanggilan *endpoint* ini bertujuan untuk mendapatkan data artikel lainnya selain artikel yang sedang dibuka yang memiliki kategori utama yang sama.

5. *Endpoint* ”articles/public?sort=createdAt:DESC” dan endpoint ”articles/public?sort=popularity:DESC” dipanggil untuk mendapatkan data artikel yang ditampilkan pada elemen daftar artikel terbaru dan elemen daftar artikel terpopuler.

Gambar 3.27. Tombol pembagian tautan artikel *web* RE/MAX Blog

Pembaca juga dapat membagikan tautan artikel ke berbagai sosial media atau menyalin tautan ke *clipboard* perangkat mereka. Tombol-tombol untuk melakukan pembagian dan penyalinan tautan dapat dilihat pada Gambar 3.27. Untuk melakukan pembagian tautan ke media sosial, pembaca harus masuk ke akun sosial media pada *browser* mereka. Contohnya, untuk membagikan tautan artikel ke sosial media Instagram, pembaca harus masuk ke akun Instagram mereka pada halaman *web* Instagram melalui *browser*.

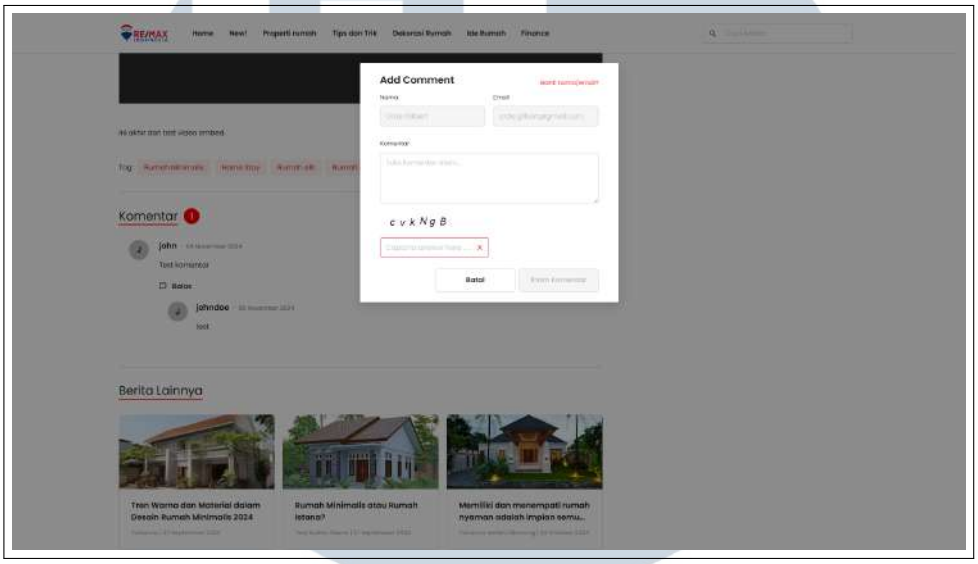


Gambar 3.28. Kolom komentar artikel *web* RE/MAX Blog

Hasil pembangunan kolom komentar dapat dilihat pada Gambar 3.28. Pembaca dapat mengirimkan komentar atau membalas komentar dengan mengisi nama, email, beserta dengan isi komentar ke artikel atau balasan terhadap komentar yang ada pada artikel. Komentar pada artikel dapat dilihat oleh publik dan juga dapat dibalas oleh pembaca lain.

Data komentar dan balasan didapat dari data artikel yang diterima dari pemanggilan *endpoint* ”articles/[*id*]/public”. Di dalam data artikel yang diterima,

terdapat satu kumpulan data yang berisi semua komentar dan semua balasan terhadap semua komentar yang ada pada artikel tersebut. Komentar pada artikel akan ditampilkan dalam bentuk daftar di bawah konten artikel dan balasan juga ditampilkan dalam bentuk daftar di bawah komentar dimana balasan tersebut diberikan.



Gambar 3.29. *Modal* komentar artikel pada RE/MAX Blog

Ketika pembaca menekan tombol ”Tambah Komentar” pada kolom komentar, halaman *web* akan menampilkan *modal* penambahan komentar. Tampilan dari*modal* tersebut dapat dilihat pada Gambar 3.29. Proses penambahan komentar pada sebuah artikel adalah sebagai berikut:

1. Pembaca menekan tombol ”Tambah Komentar” pada kolom komentar halaman artikel.
2. Halaman *web* akan menampilkan *modal* penambahan komentar seperti pada Gambar 3.29.
3. Jika pembaca sudah pernah mengirimkan komentar atau balasan, kolom *input* nama dan email akan otomatis terisi berdasarkan *input* nama dan email pada penambahan komentar/balasan sebelumnya yang tersimpan pada *local storage browser* pembaca. Pembaca dapat merubah nama dan email pada *modal* dengan menekan teks ”Ganti nama/email?” dan modal akan mengosongkan kolom *input* untuk nama dan email.

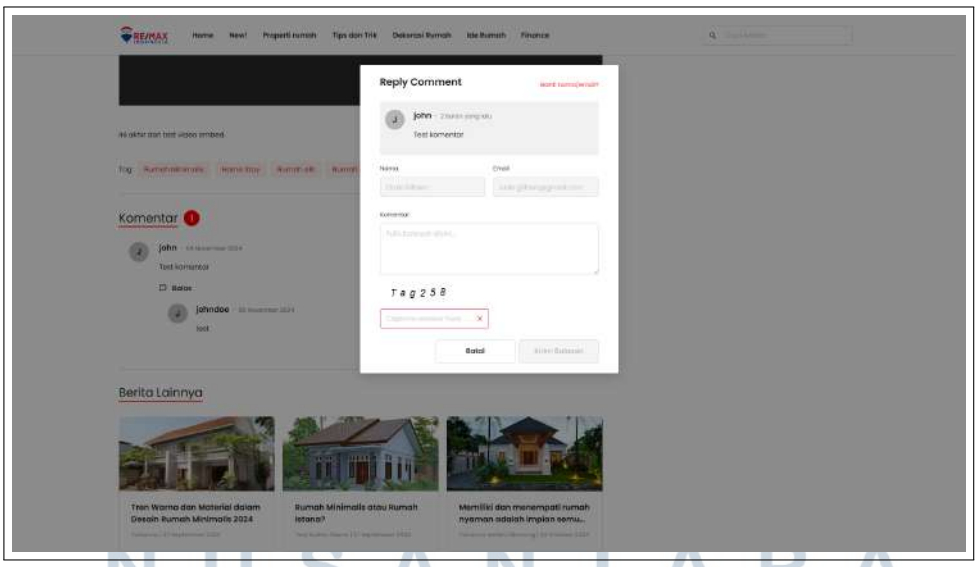
4. Jika pembaca belum pernah mengirimkan komentar atau balasan apapun, kolom *input* nama dan email akan kosong dan pembaca harus mengisi kedua kolom *input* tersebut.

5. Setelah mengisi nama dan email, pembaca harus memberikan isi komentar/balasan pada kolom yang tersedia. Kolom *input* nama, email, dan isi komentar/balasan tidak dapat kosong ketika komentar akan dikirim dan *modal* akan memberikan peringatan bahwa ada kolom *input* yang kosong

6. Sebelum mengirim komentar/balasan, pembaca juga harus melakukan verifikasi *captcha*. Tombol ”Kirim Komentar” atau ”Kirim Balasan” hanya dapat ditekan jika verifikasi *captcha* berhasil dilakukan oleh pembaca.

7. Jika pembaca ingin mengirimkan komentar, ketika tombol ”Kirim Komentar” ditekan, maka data nama, email, dan isi balasan dikirimkan di dalam *body* data bersama dengan *id* artikel ke *endpoint* ”article-comments” untuk disimpan di *database* melalui *backend*.

8. Jika proses ini berhasil, nama dan email pembaca akan disimpan ke dalam *local storage browser* dan halaman *web* akan memuat ulang sehingga komentar/balasan dapat terlihat pada kolom komentar artikel.



Gambar 3.30. *Modal* balas komentar pada RE/MAX Blog

Ketika pembaca menekan tombol ”Balas” pada suatu *item* komentar, halaman *web* akan menampilkan *modal* penambahan balasan. Tampilan dari*modal*

tersebut dapat dilihat pada Gambar 3.30. Proses penambahan balasan pada sebuah komentar yang ada pada artikel adalah sebagai berikut:

1. Pembaca menekan tombol ”Balas” pada sebuah *item* komentar yang ada pada kolom komentar halaman artikel.

2. Halaman *web* akan menampilkan *modal* penambahan balasan seperti pada Gambar 3.30. Komentar yang akan dibalas juga ditampilkan pada bagian atas *modal*.

3. Jika pembaca sudah pernah mengirimkan komentar atau balasan, kolom *input* nama dan email akan otomatis terisi berdasarkan *input* nama dan email pada penambahan komentar/balasan sebelumnya yang tersimpan pada *local storage browser* pembaca. Pembaca dapat merubah nama dan email pada *modal* dengan menekan teks ”Ganti nama/email?” dan modal akan mengosongkan kolom *input* untuk nama dan email.

4. Jika pembaca belum pernah mengirimkan komentar atau balasan apapun, kolom *input* nama dan email akan kosong dan pembaca harus mengisi kedua kolom *input* tersebut.

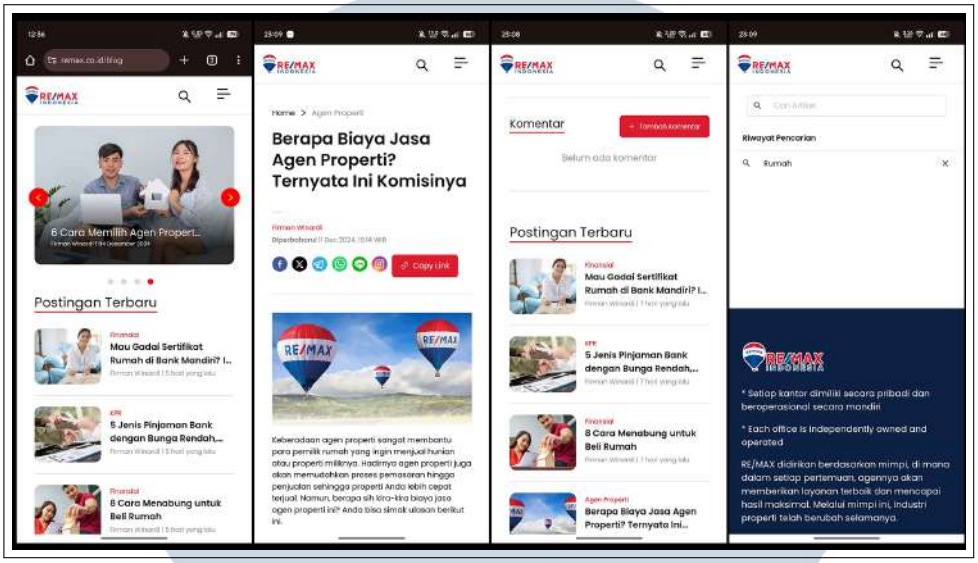
5. Setelah mengisi nama dan email, pembaca harus memberikan isi balasan pada kolom yang tersedia. Kolom *input* nama, email, dan isi balasan tidak dapat kosong ketika balasan akan dikirim dan *modal* akan memberikan peringatan bahwa ada kolom *input* yang kosong

6. Sebelum mengirim balasan, pembaca juga harus melakukan verifikasi *captcha*. Tombol ”Kirim Balasan” hanya dapat ditekan jika verifikasi *captcha* berhasil dilakukan oleh pembaca.

7. Ketika tombol ”Kirim Balasan” ditekan, maka data nama, email, dan isi balasan dikirimkan di dalam *body* data bersama dengan *id* komentar yang akan dibalas ke *endpoint* ”article-reply-comments” untuk disimpan di *database* melalui *backend*.

8. Jika proses ini berhasil, nama dan email pembaca akan disimpan ke dalam *local storage browser* dan halaman *web* akan memuat ulang sehingga balasan dapat terlihat pada *item* komentar yang ada pada kolom komentar artikel.

E. Tampilan *browser mobile*



Gambar 3.31. Tampilan *mobile web blog* RE/MAX Indonesia

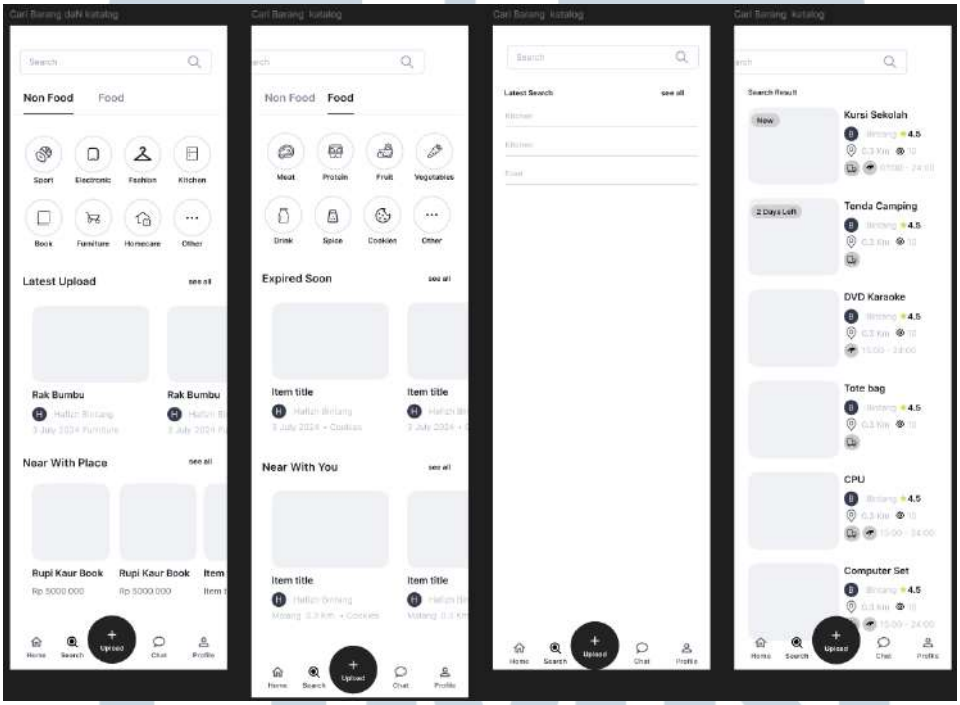
Untuk mempermudah akses pembaca ke halaman *web blog*, *frontend* dari halaman blog dikembangkan untuk responsif dengan ukuran layar dari perangkat yang digunakan. Fitur responsif ini dicapai dengan menggunakan properti ”useBreakpoints” dari komponen ”Grid” *library* Ant Design. Dengan menggunakan ”useBreakpoints”, halaman *web* dapat merubah isi halaman yang ditampilkan seperti peletakkan elemen, jumlah kolom dan baris pada susunan *grid*, ataupun jumlah baris potongan konten yang ditampilkan pada pratinjau artikel berdasarkan lebar jendela *browser* yang digunakan oleh pembaca. Oleh karena itu, halaman *web blog* dapat diakses pada bermacam-macam perangkat yang dimiliki pembaca. Keseluruhan dari halaman RE/MAX Blog juga dapat diakses pada perangkat *mobile* seperti yang terlihat pada Gambar 3.31.

3.3.2 Proyek Aplikasi *E-Commerce* Internal

Proyek ini adalah proyek internal PT. Vanz Inovatif Teknologi untuk mengembangkan sebuah aplikasi *mobile e-commerce*. Aplikasi ini dibuat dengan tujuan untuk memfasilitasi proses penjualan maupun pemberian barang bekas. Barang bekas yang dapat ditawarkan oleh penjual dapat berupa barang yang masih layak digunakan seperti perabotan, perangkat elektronik, dan furnitur, serta

makanan dan minuman yang masih layak untuk dikonsumsi. Proyek ini terdiri dari aplikasi *mobile* dan aplikasi *web Content Management System*/CMS.

Sebelum mengerjakan tugas yang diberikan, perlu dilakukan *setup* proyek ini di perangkat lokal. *Setup* dimulai dengan melakukan *clone* repositori *Github* proyek ke instansi *Windows Subsystem for Linux* (WSL) yang sedang berjalan pada perangkat. Setelah itu perlu dilakukan instalasi semua *dependency* yang diperlukan dalam menjalankan aplikasi. Jika instalasi sudah berhasil, aplikasi *mobile* dan aplikasi *web* CMS dapat dijalankan secara lokal.



Gambar 3.32. Prototype aplikasi *mobile e-commerce*

Prototype dari proyek ini dapat dilihat pada Gambar 3.32. *Prototype* yang ditampilkan diberikan oleh tim UI/UX perusahaan kepada tim *developer*. Proyek ini terdiri dari dua aplikasi, pertama aplikasi *mobile* yang tersedia untuk publik dan kedua adalah aplikasi *web Content Management System* (CMS) yang hanya tersedia untuk *admin/moderator* dari aplikasi *e-commerce*. Terdapat dua jenis pengguna pada aplikasi *mobile* proyek ini, yaitu penjual dan pembeli. Penjual dapat mem-*post* barang bekas yang ingin mereka jual atau berikan secara gratis kepada pembeli yang berminat. Pembeli dapat mencari barang yang mereka butuhkan atau inginkan dan mendapatkan kontak penjual untuk melakukan komunikasi untuk melakukan diskusi secara langsung seputar transaksi yang akan dilakukan.

Aplikasi *web* CMS adalah aplikasi berbasis *web* yang hanya dapat diakses oleh pihak admin *e-commerce*. Aplikasi ini digunakan untuk mengatur konten yang ditampilkan dan melakukan moderasi. Konten yang dimaksud adalah iklan, promosi, dan produk yang di-*post* oleh penjual. Selain itu, admin juga dapat melakukan moderasi terhadap akun yang dilaporkan jika terjadi masalah antara penjual dan pembeli.

Pada proyek ini, ada dua tugas yang diberikan:

1. **Halaman daftar produk**

   Pembuatan halaman daftar produk yang memiliki dua tab, yaitu *Non-Food* dan *Food*. Pada tab *Non-Food*, ditampilkan *post* yang termasuk dalam kategori bukan makanan. Contoh kategori yang termasuk sebagai bukan makanan adalah peralatan olahraga, barang elektronik, pakaian, peralatan dapur, buku, dan perabotan rumah. Pada tab *Food*, ditampilkan *post* yang termasuk ke dalam kategori makanan, seperti daging, buah-buahan, sayur, dan minuman.

   Komponen-komponen yang digunakan dalam halaman ini diambil dari *library* Ant Design. Komponen yang dipakai dalam pembangunan halaman daftar produk adalah ”Tabs”, ”Grid”, ”Row”, ”Col”, ”List”, dan ”Button”. Selain komponen dari *library* Ant Design, halaman ini juga menggunakan komponen *custom* yang sudah dibuat sebelumnya, yaitu ”BottomNavigation” dan ”ProductCard”.

   Pada bagian atas halaman, terdapat komponen ”Tabs” yang diberi nama *Non-Food* dan *Food*. Pada setiap *tab* terdapat ”Grid” yang menampilkan berbagai ikon dan nama kategori barang yang termasuk ke dalam nama *tab*. Contohnya pada *tab Non-Food*, kategori-kategori yang ditampilkan di dalam ”Grid” adalah kategori yang termasuk sebagai bukan makanan adalah peralatan olahraga, barang elektronik, pakaian, peralatan dapur, buku, dan perabotan rumah. Ikon yang digunakan pada ”Grid” kategori didapat dari *library* Ant Design dan dibuat oleh tim UI/UX perusahaan.

   Setelah ”Grid” kategori, ditampilkan produk-produk yang tergolong ke dalam *tab*. Produk-produk ini ditampilkan dalam sebuah komponen *endless scroll* dimana produk ditampilkan dalam satu baris dan pengguna dapat menggeser baris produk untuk melihat produk lainnya. Produk juga dikelompokkan berdasarkan kriteria tertentu dengan komponen *endless scroll* untuk setiap pengelompokkan. Sebagai contoh dari desain tim UI/UX,

terdapat pengelompokkan produk *Non-Food* berdasarkan ”Latest Upload” (unggahan terakhir) dan ”Near With Place” (dekat dengan lokasi).

Data kategori dan data produk yang ditampilkan pada halaman ini didapat melalui pemanggilan API *backend*. Untuk data kategori, *endpoint* ”product/categories?isPublished=true” dipanggil untuk mendapatkan semua kategori yang sudah dipisahkan berdasarkan *super category* ”Non-Food” dan ”Food”. Untuk data produk, *endpoint* ”public/products” dipanggil untuk mendapatkan data dari produk yang diunggah ke aplikasi.

2. **Fitur pencarian**

   Fitur pencarian barang berdasarkan nama barang, baik kategori makanan (*Food*) ataupun kategori bukan makanan (*Non-Food*). Hasil yang ditemukan dari kata kunci pencarian ditampilkan dalam bentuk daftar barang. Pada fitur ini juga ditampilkan riwayat pencarian yang dilakukan sebelumnya jika pengguna ingin melakukan pencarian ulang.

   Fitur ini terdiri atas sebuah kolom *input* yang digunakan pada beberapa halaman di dalam proyek ini. Kolom *input* tersebut dibuat menggunakan komponen ”List” dan ikon ”SearchOutlined” dari *library* Ant Design. Selain kolom *input* pencarian, fitur ini juga dapat menampilkan riwayat pencarian pengguna yang disimpan pada *local storage browser*. Setiap kali pengguna melakukan pencarian, kata kunci yang mereka masukkan disimpan ke dalam *local storage browser* mereka.

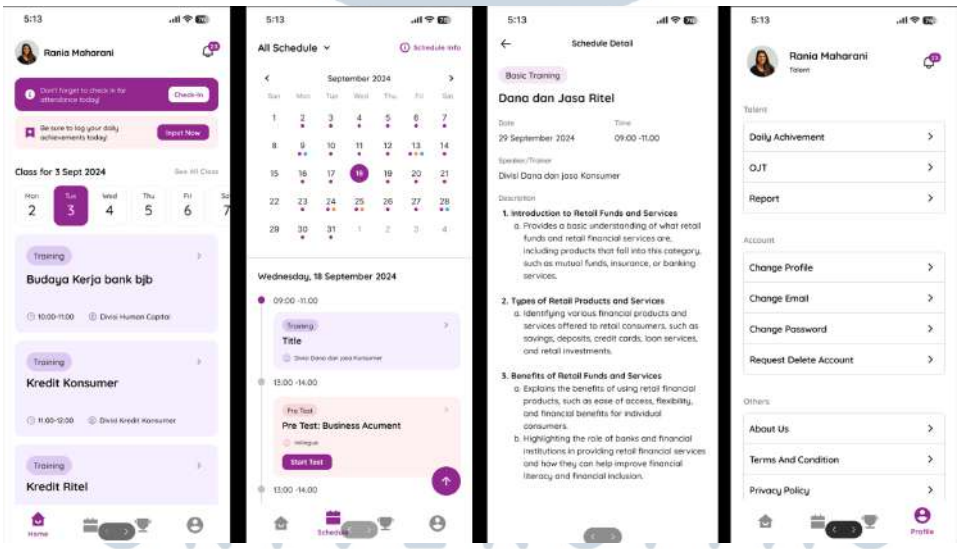
   Setelah melakukan pencarian, pengguna akan diarahkan ke halaman hasil pencarian. Pada halaman ini, produk-produk yang namanya sesuai dengan kata kunci pencarian ditampilkan dalam komponen ”List” dari *library* Ant Design. Proses pencarian ini dilakukan dengan melakukan penyaringan/*filter* dari semua data produk yang didapat dari *endpoint* ”public/products” dan menampilkan produk-produk yang nama produknya memiliki kata-kata pada kata kunci pencarian.

Tugas-tugas yang dikerjakan pada proyek ini tidak selesai dan pengembangan proyek dihentikan secara keseluruhan. Perusahaan tidak memberikan alasan apapun mengenai penghentian ini dan *supervisor* dari perusahaan mengarahkan untuk berpindah proyek dan memberikan tugas pada proyek lain. Oleh karena itu, tidak dapat ditampilkan contoh tampilan halaman apapun.

3.3.3 Proyek BJB.it

Proyek ini adalah kontrak antara World Scope Corporate College (WSCC) dengan PT. Vanz Inovatif Teknologi dalam membangun aplikasi *mobile online training*. Aplikasi ini diberi nama BJB.it dan memfasilitasi *online training* pada bidang teknologi informasi. Proyek ini terdiri dari aplikasi *mobile*, aplikasi *web Content Management System* (CMS) dan *backend*.

Sebelum mengerjakan tugas yang diberikan, perlu dilakukan *setup* proyek ini di perangkat lokal. *Setup* dimulai dengan melakukan *clone* repositori *Github* proyek ke instansi *Windows Subsystem for Linux* (WSL) yang sedang berjalan pada perangkat. Dalam mengerjakan tugas pada proyek ini, ada dua repositori yang perlu di-*clone*, yaitu repositori *backend* dan *frontend* CMS. Setelah itu perlu dilakukan instalasi semua *dependency* yang diperlukan dalam menjalankan *backend* dan aplikasi *web* CMS. Jika instalasi sudah berhasil, *backend* dan aplikasi *web* CMS dapat dijalankan secara lokal. Jika *backend* berhasil dijalankan, perlu dilakukan *migration* untuk membuat skema tabel yang digunakan dalam menjalankan proyek ini. *Database* pada proyek ini menggunakan sistem PostgreSQL.



Gambar 3.33. Contoh tampilan aplikasi *mobile* BJB.it

Aplikasi *mobile* digunakan oleh pelajar yang sudah mendaftar pada program *online training*. Tampilan dari aplikasi *mobile* yang sudah dibuat oleh tim *developer* pada bagian pengembangan aplikasi dapat dilihat pada Gambar 3.33. Pada aplikasi tersebut, pelajar dapat melihat tugas apa saja yang harus mereka lakukan dan proses pembelajaran *online* apa saja yang harus mereka ikuti. Selain itu, proyek ini juga

memiliki aplikasi *web* CMS dimana pengajar dapat memberikan penilaian pada proses pembelajaran kepada setiap pelajar.

Pada proyek ini, tugas-tugas yang dikerjakan adalah:

- Pembuatan modul *backend* yang berupa proses *create, read, update, and delete* (CRUD) untuk fitur *course/training* dan melakukan integrasi modul tersebut ke aplikasi *web* CMS.

  Tugas ini dimulai dengan membuat tabel baru untuk menyimpan data *course/training* yang akan dibuat. Desain tabel *courses* dapat dilihat pada Tabel 3.2.

Tabel 3.2. Tabel *courses*

Nama Kolom	**Tipe Data**	**Deskripsi**
courseId (PK)	VARCHAR(255)	*Id course* sebagai tanda pengenal unik
courseType	VARCHAR(255)	Tipe *course*
title	VARCHAR(255)	Judul *course*
trainer	TEXT	Nama pengajar *course*
objective	TEXT	Deskripsi tujuan pembelajaran *course*
createdByUserId	VARCHAR(255)	*Id user* yang membuat *course*
createdAt	DATE	Tanggal pembuatan *course*
updatedAt	DATE	Tanggal terakhir *course* diperbaharui

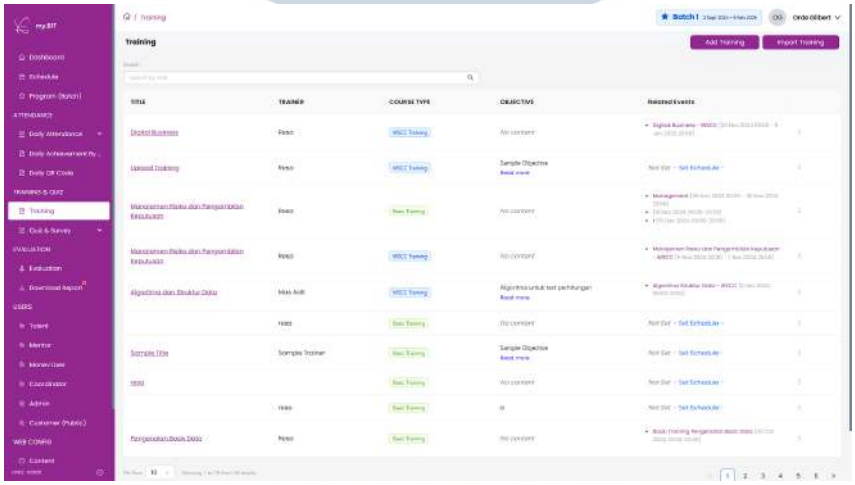
Setelah tabel berhasil dibuat pada *database*, perlu dibuat fungsi-fungsi pada *backend* untuk menjalankan proses CRUD dan *endpoint* untuk mengakses fungsi-fungsi tersebut. *Endpoint-endpoint* yang dibuat pada fitur *course/training* ini adalah sebagai berikut:

- GET ”/course” untuk mendapatkan data semua *course* yang dibuat dari *database*.
- GET ”/course/[courseId]” untuk mendapatkan data dari satu *course* sesuai dengan *courseId* yang diterima *endpoint* dari *database*.

– POST ”/course” dengan *body* data yang berisi data *course* yang disimpan pada tabel untuk membuat dan menyimpan data *course* baru ke *database*.
– PATCH ”/course/[courseId]” dengan *body* data yang berisi data *course* baru. Pemanggilan *endpoint* ini akan mengganti data *course* yang diidentifikasikan menggunakan *courseId* dengan data *course* yang baru dan menyimpan data yang baru pada *database*.
– DELETE ”/course/[courseId]” untuk menghapus data *course* yang ditemukan menggunakan *courseId*.

Kemudian, *endpoint-endpoint* yang sudah dibuat diintegrasikan ke aplikasi *web* CMS BJB.it. Integrasi dimulai dengan membuat halaman untuk fitur *course*. Halaman-halaman yang dibuat untuk fitur *course* adalah:

– **Halaman *index course***
Halaman ini memuat semua data *course* yang tersimpan di dalam *database*.



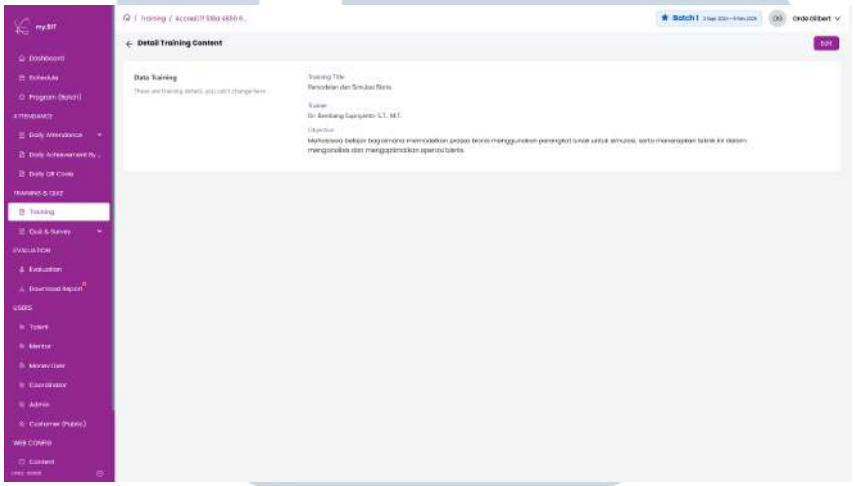
Gambar 3.34. Halaman *index course/training*

Desain dari semua halaman fitur pada aplikasi *web* CMS sudah distandarisasi, sehingga pembuatan halaman baru dapat melakukan referensi langsung dari halaman lain yang sudah dibuat sebelumnya. Untuk halaman *index course/training*, kolom-kolom pada tabel disesuaikan dengan data *course* dan *endpoint* sumber data diganti menjadi ”/course”. Tampilan dari halaman ini dapat dilihat pada

Gambar 3.34. Pengguna dapat menekan judul *course* pada kolom ”TITLE” untuk membuka halaman *detail* dari *course*. Pada kanan atas halaman juga terdapat tombol ”Add Course” dimana pengguna akan diarahkan ke halaman *add/edit course* untuk membuat *course baru*.

– **Halaman *detail course***

Halaman ini memuat detail *course* yang tersimpan di dalam *database*.

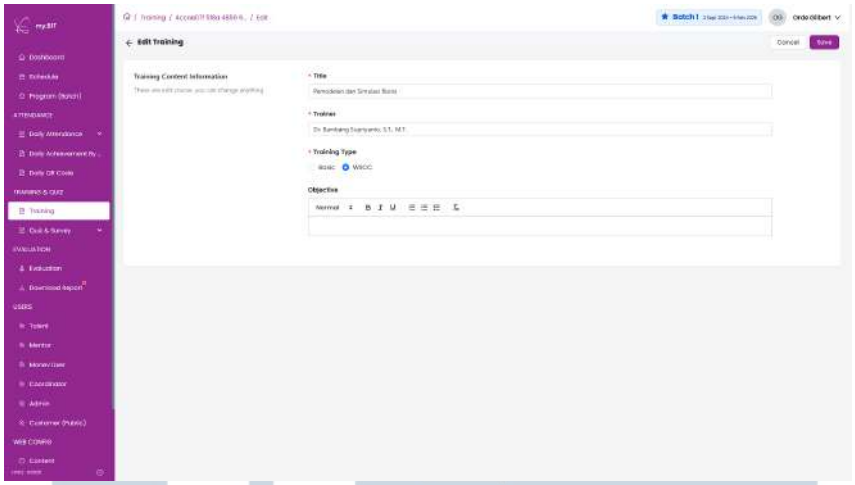


Gambar 3.35. Halaman *detail course/training*

Desain halaman *detail* juga distandarisasi dan hanya perlu mengubah data-data yang ditampilkan sesuai dengan data *course*. Data detail *course* pada halaman ini didapat dari *endpoint* ”/course/[courseId]” dimana ”[courseId]” didapat dari tautan halaman. Data yang ditampilkan pada halaman *detail* adalah judul *course*, nama *trainer*, dan *objective course*. Pada bagian kanan atas halaman terdapat tombol ”Edit” dimana pengguna dapat mengubah data dari *course* yang sedang mereka buka pada halaman *add/edit course*.

– **Halaman *add/edit course***

Pengguna dapat menambahkan data *course* baru atau mengubah data *course* yang sudah tersimpan pada *database*



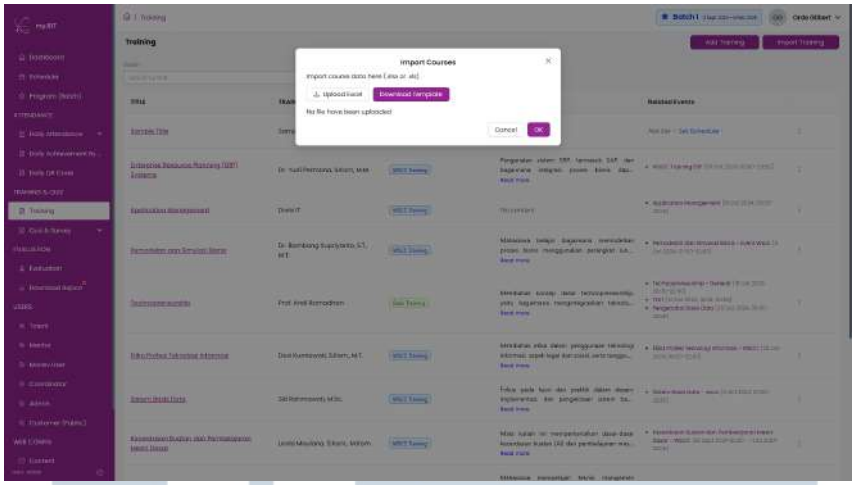
Gambar 3.36. Halaman *add/edit course/training*

Desain halaman *add/edit* juga distandarisasi dan hanya perlu perubahan pada nama kolom *input* dan jenis kolom *input*. Jika pengguna mengakses halaman ini dari halaman *index*, maka semua kolom *input* akan kosong. Tetapi jika pengguna mengakses halaman ini melalui halaman *detail*, kolom-kolom *input* akan terisi dengan data *course* yang ingin mereka ubah. Pengguna dapat menyimpan data baru/perubahan data dengan menekan tombol ”Save” pada bagian kanan atas halaman. *Endpoint* yang akan dipanggil adalah ”/course” dengan metode POST untuk menyimpan data baru atau *endpoint* ”/course/[courseId]” dengan metode PATCH untuk melakukan perubahan data.

- Penambahan fitur pada CMS untuk melakukan impor *file* untuk memasukkan data *course*/*training* dalam jumlah besar melalui aplikasi *web* CMS.

  Untuk melakukan impor data, diperlukan *endpoint* tambahan pada modul *course*. *Endpoint* ”/course/bulk” dengan *body* data berupa kumpulan data *course*. *Backend* akan membaca kumpulan data *course* yang diterima dan menyimpan semua data yang diterima ke *database*.

  Setelah itu, dibuat *modal* untuk melakukan impor file *excel sheet* dari perangkat pengguna ke halaman *web* CMS.

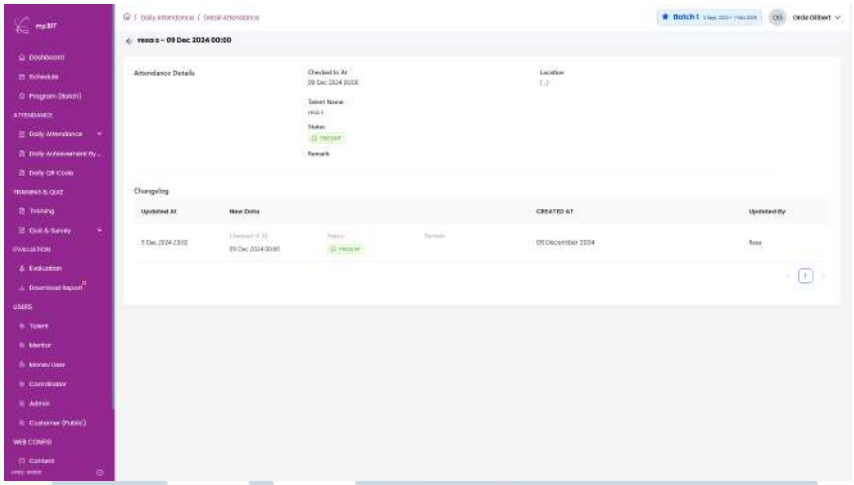


Gambar 3.37. *Modal* impor data *course/training*

Tampilan dari *modal* impor data dapat dilihat pada Gambar 3.37. *Modal* import dibuka dengan menekan tombol ”Import Courses” pada halaman *index course* (Gambar 3.34. Ketika pengguna menunggah sebuah *file excel* melalui *modal* tersebut, halaman *web* akan memproses data dari *file excel* yang diunggah menjadi objek-objek data *course*. *Modal* akan menampilkan *preview* dari data yang diterima dalam bentuk tabel seperti tabel pada halaman *index* yang dapat dilihat pada Gambar 3.34. Pengguna dapat mengirimkan data yang telah diproses tersebut dengan menekan tombol ”OK” dan *web* CMS akan mengirimkan data tersebut sebagai *body* ke *endpoint* ”/course/bulk” untuk disimpan ke *database*.

- Menambahkan *changelog* pada fitur absen/*attendance* untuk menampilkan perubahan data pada absensi pelajar. Fitur tersebut menyimpan riwayat data absen dari seorang pelajar, baik ketika mereka melakukan *daily attendance* ataupun data absen mereka diubah oleh pihak admin BJB.it melalui aplikasi *web* CMS.

  Pada halaman *detail* data *attendance* yang sudah dibuat sebelumnya, ditambahkan tabel *changelog* sebagai riwayat perubahan dari data *attendance* tersebut.



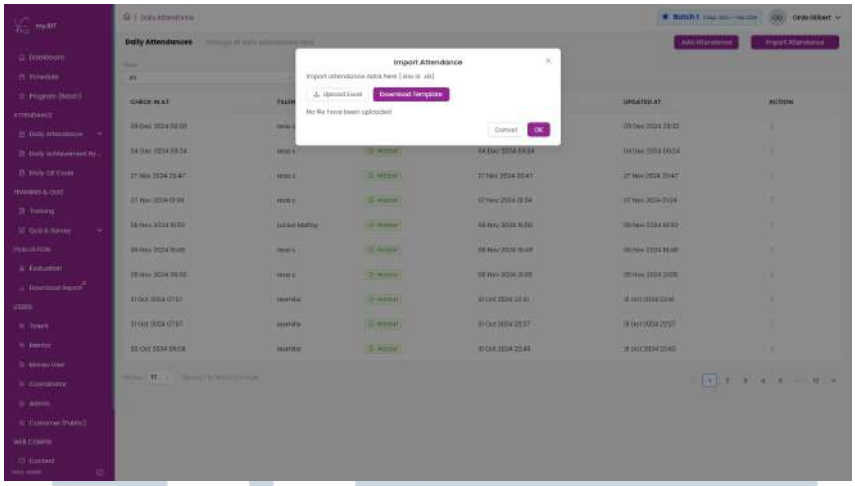
Gambar 3.38. Tampilan tabel *changelog* pada halaman *detail attendance*

Tampilan dari tabel dapat dilihat pada Gambar 3.38. Tabel memuat riwayat perubahan data dengan menampilkan tanggal perubahan, tanggal dan jam *attendance*, status *attendance*, *remark*/catatan, dan nama *admin* yang melakukan perubahan.

Data *changelog* didapat dari *endpoint* ”attendance-change-logs/[attendanceId]”. Bagian ”[attendanceId]” dari *endpoint* diganti dengan *id* attendance yang didapat dari tautan halaman *detail*.

- Menambahkan fitur impor *file* untuk absensi/*attendance* pada halaman *attendance* di aplikasi *web* CMS.

  Fitur impor *attendance* menggunakan proses yang sama dengan fitur impor pada data *course*. Pengguna dapat menunggah *file excel* ke *modal* impor data dan melakukan penyimpanan.



Gambar 3.39. *Modal* impor data *attendance*

Modal impor *attendance* yang ditampilkan pada Gambar 3.39 dapat dibuka oleh pengguna dengan menekan tombol ”Import Attendance” pada bagian kanan atas halaman *index attendance*. Ketika pengguna menunggah sebuah *file excel* melalui *modal* tersebut, halaman *web* akan memproses data dari *file excel* yang diunggah menjadi objek-objek data *attendance*. *Modal* akan menampilkan *preview* dari data yang diterima dalam bentuk tabel seperti tabel pada halaman *index* yang dapat dilihat pada Gambar 3.34. Pengguna dapat mengirimkan data yang telah diproses tersebut dengan menekan tombol ”OK” dan *web* CMS akan mengirimkan data tersebut sebagai *body* ke *endpoint* ”/attendance/bulk” untuk disimpan ke *database*.

- Pembuatan modul *backend* yang berupa proses *create, read, update, and delete* (CRUD) untuk fitur *program*/*batch* dan melakukan integrasi ke aplikasi *web* CMS. Selain proses CRUD, modul ini juga memiliki fitur untuk menambahkan pelajar ke suatu *program*/*batch* yang sudah dibuat.

  Tugas ini dimulai dengan membuat tabel baru untuk menyimpan data *program* yang akan dibuat. Desain tabel *program* dapat dilihat pada Tabel 3.3.

Tabel 3.3. Tabel *programs*

Nama Kolom	**Tipe Data**	**Deskripsi**
programId (PK)	VARCHAR(255)	*Id program* sebagai tanda pengenal unik
title	VARCHAR(255)	Judul *program*
startAt	DATE	Tanggal mulai *program*
endAt	DATE	Tanggal selesai *program*
maxAttendanceInDays	INTEGER	Jumlah hari *program*
createdByUserId	VARCHAR(255)	*Id user* yang membuat *program*
createdAt	DATE	Tanggal pembuatan *program*
updatedAt	DATE	Tanggal terakhir *program* diperbaharui

Selain itu, dibuat juga tabel baru untuk menyimpan data relasi antara *program* dengan *talent*/pelajar yang akan dibuat. Desain tabel *program_talents* dapat dilihat pada Tabel 3.4.

Tabel 3.4. Tabel *program_talents*

Nama Kolom	**Tipe Data**	**Deskripsi**
id (PK)	VARCHAR(255)	*Id relasi* sebagai tanda pengenal unik
programId	VARCHAR(255)	*Id program* sebagai tanda pengenal *program*
talentId	VARCHAR(255)	*Id talent* sebagai tanda pengenal *talent*/pelajar
createdByUserId	VARCHAR(255)	*Id user* yang membuat *program*
createdAt	DATE	Tanggal pembuatan *program*
updatedAt	DATE	Tanggal terakhir *program* diperbaharui

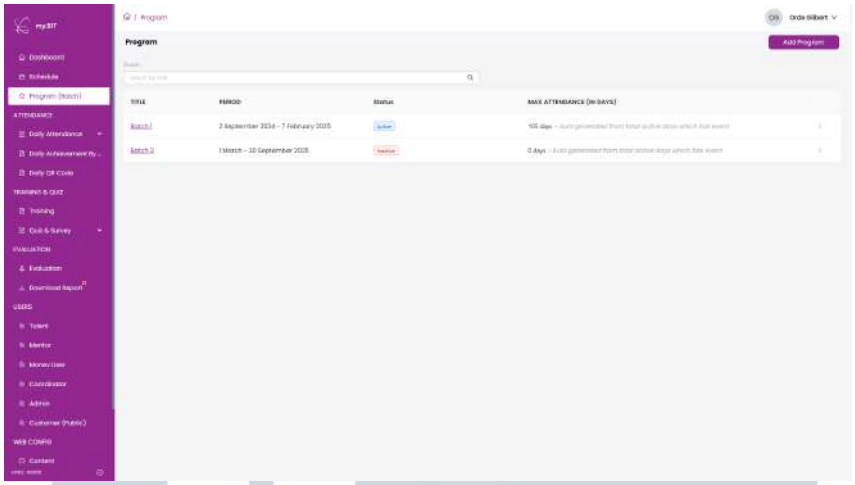
Setelah kedua tabel berhasil dibuat pada *database*, perlu dibuat fungsi-fungsi pada *backend* untuk menjalankan proses CRUD dan *endpoint* untuk

mengakses fungsi-fungsi tersebut. *Endpoint-endpoint* yang dibuat pada fitur *program* ini adalah sebagai berikut:

- GET ”/program” untuk mendapatkan data semua *program* yang dibuat dari *database*.
- GET ”/program/[programId]” untuk mendapatkan data dari satu *program* sesuai dengan *programId* yang diterima *endpoint* dari *database*.
- POST ”/program” dengan *body* data yang berisi data *program* yang disimpan pada tabel untuk membuat dan menyimpan data *program* baru ke *database*.
- PATCH ”/program/[programId]” dengan *body* data yang berisi data *program* baru. Pemanggilan *endpoint* ini akan mengganti data *program* yang diidentifikasikan menggunakan *programId* dengan data *program* yang baru dan menyimpan data yang baru pada *database*.
- DELETE ”/program/[programId]” untuk menghapus data *program* yang ditemukan menggunakan *programId*.
- GET ”/program-talents?programId=[programId]” untuk mendapatkan semua data relasi dari sebuah *program* yang diidentifikasikan menggunakan *programId*.
- POST ”/program-talents” dengan *body* data berisi *id program* dan *id talent*. *Endpoint* ini akan menyimpan relasi baru ke dalam tabel *program-talents* pada *database*.
- DELETE ”/program-talents/[id]” untuk menghapus relasi dari tabel *program_talents*.

Kemudian, *endpoint-endpoint* yang sudah dibuat diintegrasikan ke aplikasi *web* CMS BJB.it. Integrasi dimulai dengan membuat halaman untuk fitur *program*. Halaman-halaman yang dibuat untuk fitur *program* adalah:

- **Halaman *index program***
  Halaman ini memuat semua data *program* yang tersimpan di dalam *database*.

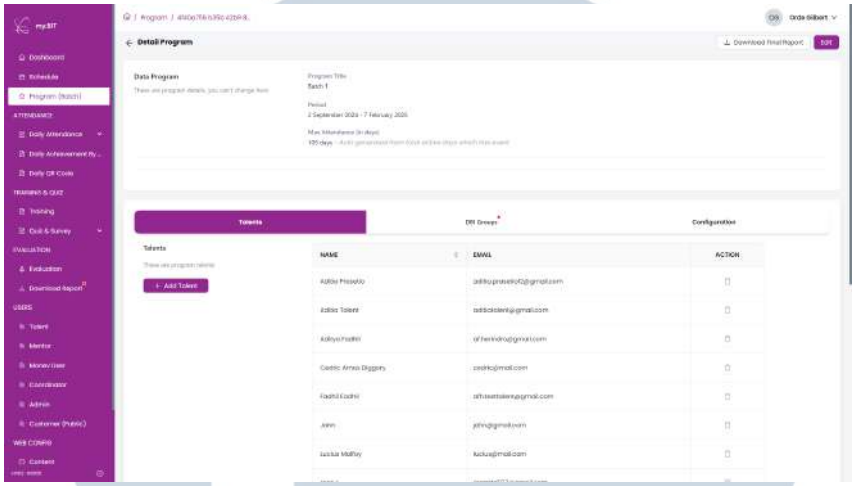


Gambar 3.40. Halaman *index program*

Desain dari semua halaman fitur pada aplikasi *web* CMS sudah distandarisasi, sehingga pembuatan halaman baru dapat melakukan referensi langsung dari halaman lain yang sudah dibuat sebelumnya. Untuk halaman *index program/training*, kolom-kolom pada tabel disesuaikan dengan data *program* dan *endpoint* sumber data diganti menjadi ”/program”. Tampilan dari halaman ini dapat dilihat pada Gambar 3.34. Pengguna dapat menekan judul *program* pada kolom ”TITLE” untuk membuka halaman *detail* dari *program*. Pada kanan atas halaman juga terdapat tombol ”Add Program” dimana pengguna akan diarahkan ke halaman *add/edit program* untuk membuat *program baru*.

– **Halaman *detail program***

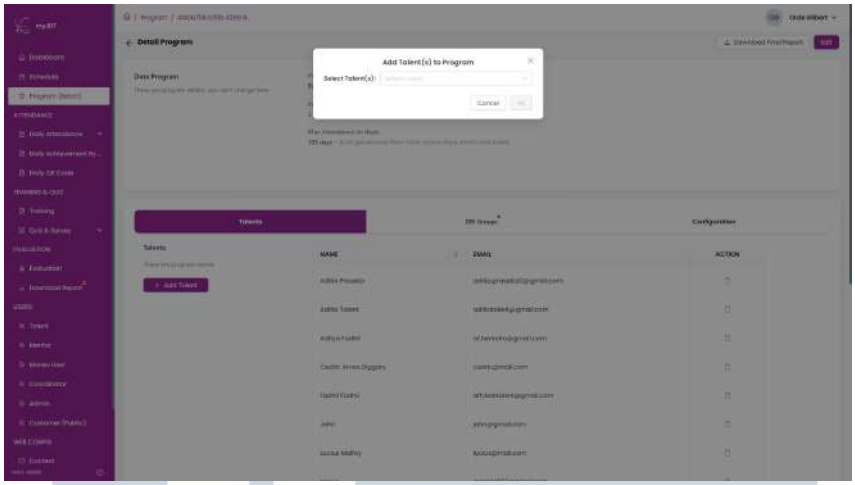
Halaman ini memuat detail *program* yang tersimpan di dalam *database*.



Gambar 3.41. Halaman *detail program*

Desain halaman *detail* juga distandarisasi dan hanya perlu mengubah data-data yang ditampilkan sesuai dengan data *program*. Data detail *program* pada halaman ini didapat dari *endpoint* ”/program/[programId]” dimana ”[programId]” didapat dari tautan halaman. Data yang ditampilkan pada halaman *detail* adalah judul *program*, periode *program*, dan jumlah total *attendance program*. Pada bagian kanan atas halaman terdapat tombol ”Edit” dimana pengguna dapat mengubah data dari *program* yang sedang mereka buka pada halaman *add/edit program*.

– *Modal penambahan talent* Pengguna dapat menambahkan *talent* ke dalam sebuah *program* melalui fitur yang ada pada halaman *detail program*.

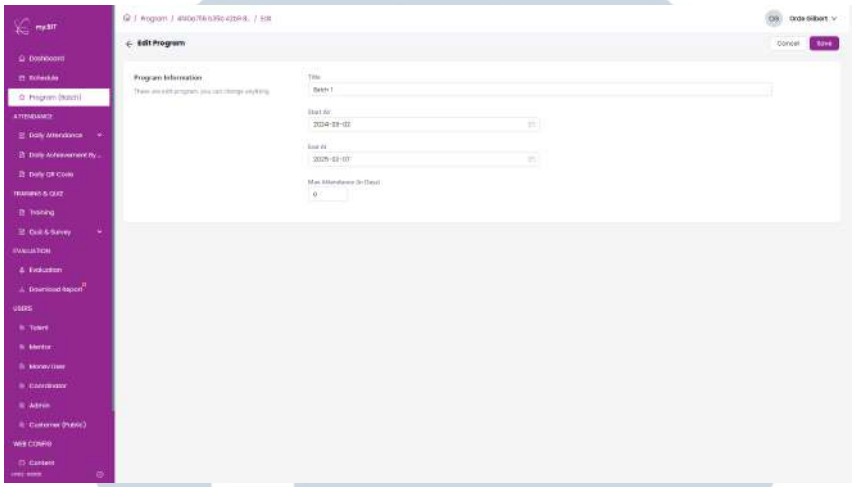


Gambar 3.42. *Modal* penambahan *talent*

Pengguna dapat menekan tombol ”Add Talent” untuk menambahkan *talent* ke dalam sebuah *program*. Ketika tombol tersebut ditekan, *modal* penambahan *talent* akan ditampilkan. Tampilan dari *modal* dapat dilihat pada Gambar 3.42. Dalam *modal*, pengguna dapat memilih *talent* yang ingin ditambahkan ke *program* dari *drop-down* yang ada pada *modal*. Setelah memilih *talent*, pengguna dapat menekan tombol ”OK” dan *web* CMS akan memanggil *endpoint* ”program-talents” dengan *body* data berisi *id program* dan *id talent* agar *backend* dapat membuat relasi baru dan menyimpan ke *database*.

– **Halaman *add/edit program***

Pengguna dapat menambahkan data *program* baru atau mengubah data *program* yang sudah tersimpan pada *database*



Gambar 3.43. Halaman *add/edit program*

Desain halaman *add/edit* juga distandarisasi dan hanya perlu perubahan pada nama kolom *input* dan jenis kolom *input*. Jika pengguna mengakses halaman ini dari halaman *index*, maka semua kolom *input* akan kosong. Tetapi jika pengguna mengakses halaman ini melalui halaman *detail*, kolom-kolom *input* akan terisi dengan data *program* yang ingin mereka ubah. Pengguna dapat menyimpan data baru/perubahan data dengan menekan tombol ”Save” pada bagian kanan atas halaman. *Endpoint* yang akan dipanggil adalah ”/program” dengan metode POST untuk menyimpan data baru atau *endpoint* ”/program/[programId]” dengan metode PATCH untuk melakukan perubahan data.

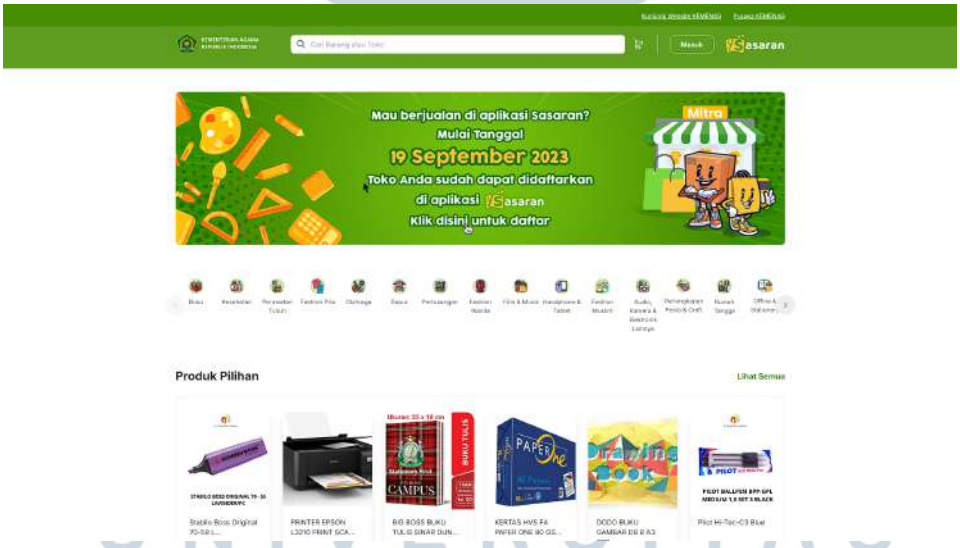
- Pembuatan modul *backend* untuk mengumpulkan nilai semua pelajar pada suatu *program* dan melakukan ekspor rapor hasil pembelajaran dalam bentuk *excel sheet*.

  Penambahan modul ini tidak memerlukan penambahan tabel baru pada *database*. Modul ini berfungsi untuk mengambil data nilai semua *talent*/pelajar dari sebuah *program* dan menulis data-data tersebut ke dalam sebuah *file excel sheet*. Modul ini hanya memiliki satu *endpoint*, yaitu ”/finalreport/download/[programId]”. *Endpoint* tersebut menerima

id program dan mendapatkan data dari modul *report*. Modul *report* adalah modul yang mengumpulkan semua data nilai yang dikelompokkan berdasarkan *talent* yang ada di dalam sebuah *program*. Fungsi yang dijalankan oleh *endpoint* ”/finalreport/download/[programId]” menerima data *report* tersebut dan menulis data yang didapatkan ke sebuah *file excel sheet* yang dapat diunduh oleh pengguna.

3.3.4 Proyek Instansi Pemerintah

Proyek ini adalah proyek yang diselenggarakan oleh sebuah instansi pemerintah dalam bentuk halaman *web e-commerce* dengan nama Sasaran. Halaman *web* ini dibuat untuk membantu madrasah dan pesantren dalam membeli perabotan, perangkat, maupun peralatan yang mereka butuhkan menggunakan dana bantuan yang mereka terima dari intansi pemerintah yang terkait. Penjual pada halaman *web* ini merupakan mitra dagang yang sudah mendaftarkan diri ke instansi pemerintah tersebut. Tampilan dari halaman utama *web* Sasaran diperlihatkan pada Gambar 3.44



Gambar 3.44. Halaman utama *web e-commerce* Sasaran

Halaman utama *web e-commerce* yang ditampilkan pada Gambar 3.44 adalah halaman *web* yang sudah dibuat dan dikembangkan sebelumnya oleh tim *developer* dari instansi pemerintah, baik pada lapisan *frontend* maupun *backend*. PT. Vanz Inovatif Teknologi dikontrak untuk membantu pengembangan *backend* halaman *web* tersebut. Tugas pada proyek ini diberikan langsung oleh tim

developer dari instansi pemerintahan. Tugas yang diberikan sebagian besar adalah penambahan fungsi pada *backend website e-commerce* tersebut. *Backend* dari proyek ini dibagi menjadi beberapa *microservice* dimana untuk mengerjakan tugas, diberikan akses ke *microservice order*, *microservice user* dan *microservice catalog*. Gambaran umum dari tugas-tugas yang diberikan pada proyek ini adalah:

- Membuat *endpoint* untuk melakukan pembayaran menggunakan *e-wallet* khusus Sasaran yang berisi dana dari pemerintah untuk madrasah sebagai sumber dana mereka untuk melakukan pembelanjaan pada *web* Sasaran.

  Endpoint yang dibuat adalah ”/submitWalletPayment” pada *microservice catalog* dimana data pembayaran dikirimkan ke *microservice order* melalui metode komunikasi antar *microservice* berupa *channel* RabbitMQ. RabbitMQ adalah perangkat lunak pengiriman data berupa teks dan digunakan pada proyek ini untuk mengirimkan data dari satu *microservice* ke *microservice* lain. Data pembayaran yang diterima oleh *microservice order* akan diteruskan ke API Bank Mandiri untuk memproses pembayaran.

- Membuat fungsi untuk *generate invoice* sebagai bukti pembelian dan pembayaran yang sudah dilakukan oleh pembeli

  Fungsi ini merupakan fungsi tambahan ke *endpoint* ”/submitWalletPayment” dimana berdasarkan data pembayaran yang dibuat pada *microservice catalog*, dibuat objek data *invoice*. *Invoice* hanya akan dibuat jika data pembayaran berhasil diterima oleh *microservice order* dan data pembayaran dikirimkan ke API Bank Mandiri untuk diproses.

- Membuat fungsi untuk pembuatan *e-wallet* Sasaran secara otomatis setelah pengguna membuat akun

  Fungsi pembuatan *e-wallet* ini dibuat untuk melengkapi proses pembuatan akun *store* atau *organization* pada *web* Sasaran. Fungsi ini akan dijalankan setelah fungsi *approveStoreRegistration* untuk akun *store* atau fungsi *addOrganization* untuk akun *organization* berhasil dijalankan pada *microservice user*. *Microservice user* akan mengirimkan data akun yang baru dibuat ke *microservice order* untuk diproses pembuatan *e-wallet*. Fungsi yang dibuat pada tugas ini memproses pengiriman data dari *microservice user* ke *microservice order*.

- Membuat *endpoint* baru yang digunakan untuk melakukan

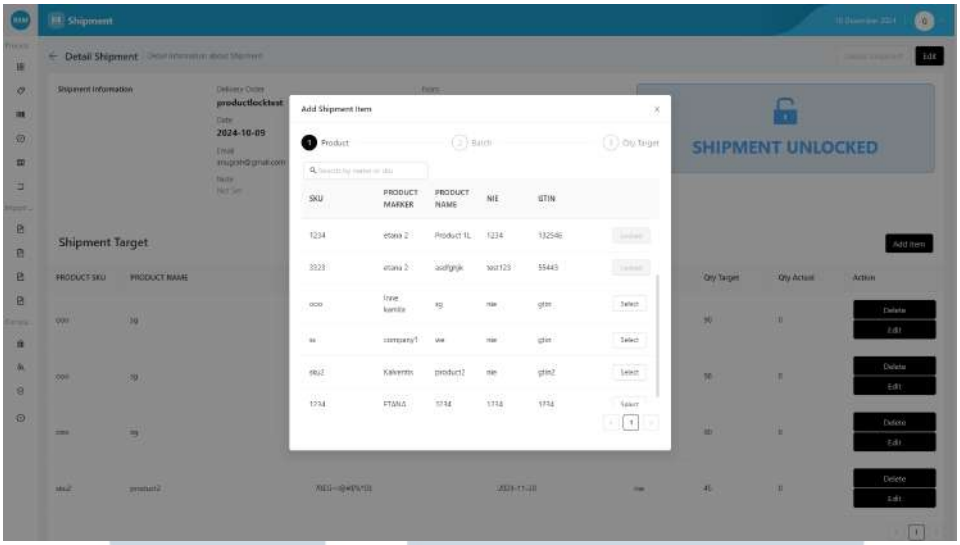
resolusi/menyelesaikan komplain/keluhan yang diajukan oleh pembeli mengenai barang yang mereka terima kepada penjual

Endpoint baru yang dibuat pada tugas ini adalah ”/complaint/buyer/setStatusCompleted” dengan *body* data berisi *id* komplain dan status ”COMPLETED”. Ketika *endpoint* ini dipanggil, *backend* akan mengecek apakah komplain yang didefinisikan melalui *id* komplain pada *body* data statusnya masih ”IN_PROGRESS” atau tidak. Jika masih, data komplain akan diubah statusnya menjadi ”COMPLETED” dan data transaksi yang terkait dengan komplain tersebut juga akan mengalami perubahan status menjadi ”COMPLETED”. Jika status komplain bukan ”IN_PROGRESS”, maka *endpoint* akan mengembalikan pesan *error*.

3.3.5 Proyek TracknTrace

Proyek TracknTrace adalah proyek kerja sama PT. Vanz Inovatif Teknologi dengan klien untuk memantau produksi dan distribusi produk. TracknTrace adalah aplikasi *Content Management System* (CMS) berbasis *web* yang menyimpan data produk, *batch*, *workorder*, *shipment*, dan *return* sebagai sistem pendataan dari berbagai proses dalam produksi dan distribusi produk obat-obatan.

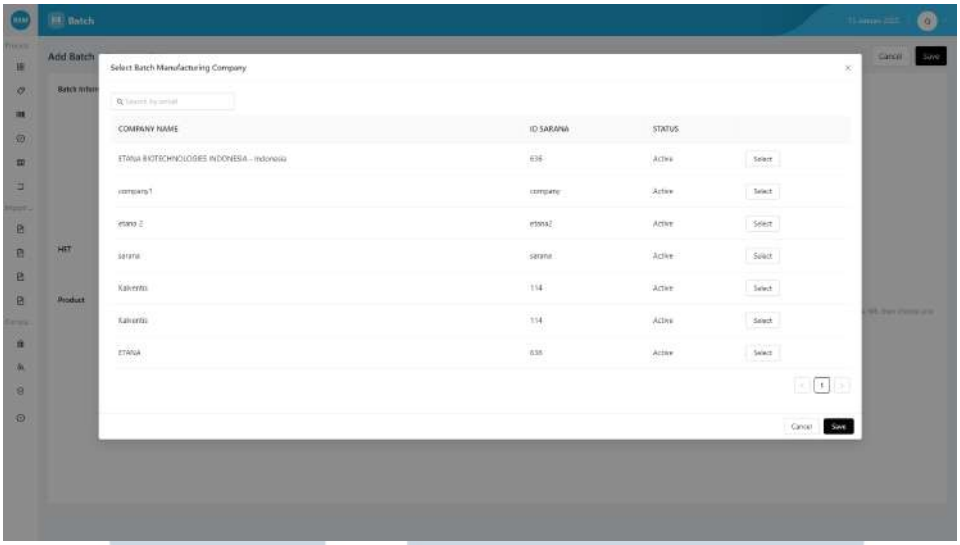
Hasil dari proyek ini adalah sebuah aplikasi berbasis *web* yang sudah digunakan oleh klien dalam aktivitas operasional perusahaannya. Pada proyek ini, tugas yang dilakukan adalah menambahkan konfigurasi untuk menampilkan data yang dikunci atau inaktif. Konfigurasi tersebut mengatur aplikasi untuk menampilkan data inaktif atau tidak. Meskipun ditampilkan melalui konfigurasi tersebut, data yang inaktif tetap tidak dapat dipilih dalam proses operasional aplikasi.



Gambar 3.45. Tampilan penambahan *batch* pada fitur *shipment*

Gambar 3.45 menunjukkan contoh tampilan aplikasi ketika konfigurasi menampilkan data yang inaktif/dikunci. Contoh tersebut berada pada fitur *shipment* dimana setiap *shipment* dapat ditambahkan sebuah *batch* dan produk, beserta dengan jumlah produk yang akan dikirimkan. Data *batch* yang dikunci atau inaktif tidak dapat ditambahkan ke sebuah *shipment*, tetapi masih ditampilkan pada tahap memilih *batch*. Jika pada pengaturan konfigurasi data-data tersebut diatur untuk tidak ditampilkan, maka aplikasi tidak akan menampilkan data-data yang inaktif/dikunci pada semua tahap pemilihan data.

Konfigurasi yang dibuat ini mempengaruhi proses pengambilan data setiap kali *modal* penambahan data ditampilkan. Ketika konfigurasi dibuat *true*, maka pemanggilan *endpoint* data akan mendapat tambahan *params* dengan tujuan untuk hanya mengambil data yang statusnya ”ACTIVE”.



Gambar 3.46. Tampilan penambahan data *manufactured by* pada halaman pembuatan *batch*

Selain penambahan konfigurasi, pada proyek TracknTrace juga diberikan tugas untuk menambahkan kolom data *manufactured by* pada fitur pembuatan *batch* baru. Tampilan dari aplikasi *web* ketika melakukan penambahan data *manufactured by* pada halaman pembuatan *batch* dapat dilihat pada Gambar 3.46. Data *manufactured by* pada *batch* menyimpan informasi berupa *id* dari perusahaan/pabrik yang memproduksi *batch*.

Pada lapisan *backend*, fungsi pembuatan *batch* disesuaikan untuk memasukkan data *manufacturedBy* yang menyimpan *id* perusahaan/pabrik yang memproduksi *batch*. Tabel *batch* pada *database* juga ditambahkan kolom *iManufacturedBy* untuk menyimpan *id* perusahaan/pabrik yang diterima dari data pembuatan *batch* baru yang didapat dari *backend*.

3.4 Kendala dan Solusi yang Ditemukan

Dalam pelaksanaan rancang bangun *frontend* halaman *web* blog RE/MAX Indonesia, ditemukan beberapa kendala dan diterapkan beberapa solusi terhadap kendala tersebut. Berikut adalah kendala yang dihadapi dan solusi yang diterapkan pada proses rancang bangun halaman *web blog*:

3.4.1 Kendala

Kendala yang ditemukan selama pelaksanaan program kerja magang di antara lain adalah sebagai berikut:

1. Kendala pada perangkat dan sistem operasi yang digunakan berbeda dengan standar operasional dari perusahaan. Selama pelaksanaan program kerja magang, perangkat yang digunakan menggunakan sistem operasi *Windows*, sedangkan produk-produk yang dikembangkan dan dibuat pada sistem operasi *Linux*. Oleh karena itu, perangkat harus melakukan instalasi *Windows Subsystem for Linux (WSL)*. Kendala muncul karena WSL tersebut membutuhkan memori perangkat yang besar, sehingga seringkali terjadi *crash*. Akibatnya, proses rancang bangun proyek beberapa kali terhambat untuk menyalakan WSL kembali karena folder proyek tidak dapat diakses tanpa adanya instansi WSL yang sedang berjalan.
2. Komunikasi dengan rekan kerja maupun *team leader* dari divisi *Internal/Inhouse Application Development* maupun dari divisi lain yang tidak dilakukan secara langsung. Karena perusahaan beroperasi secara daring/WFH, selama pelaksanaan proses program kerja magang komunikasi antar rekan kerja dilakukan melalui WhatsApp dan seringkali harus menunggu respon yang cenderung cukup lama. Akibatnya, komunikasi cenderung kurang bisa berjalan lancar dan seringkali masalah yang ingin ditanyakan sulit dijelaskan tanpa adanya diskusi/pertemuan langsung.

3.4.2 Solusi

Solusi yang diterapkan untuk menyelesaikan kendala yang dihadapi di atas adalah:

1. Solusi yang diterapkan untuk kendala pada perangkat adalah menambah memori fisik perangkat dan memastikan bahwa tidak ada proses lain yang sedang berjalan pada perangkat sebelum menyalakan instansi WSL.
2. Solusi untuk kendala komunikasi adalah menjadwalkan *meeting* bersama dengan rekan kerja atau *team leader* pada divisi *Internal/Inhouse Application Development* untuk melakukan diskusi langsung. Melalui *meeting* tersebut, rekan kerja satu dengan yang lain dapat membagikan tampilan layar mereka agar bisa menunjukkan langsung masalah yang mereka hadapi dan melakukan perbincangan/diskusi secara langsung.