

**PENGEMBANGAN *WEBSITE* PARIWISATA KARATON
NGAYOGYAKARTA HADININGRAT DENGAN ARSITEKTUR
JAMSTACK MENGGUNAKAN *ASTRO***



Disusun Oleh:

N a m a : Zaky Abiyyu Andri
NIM : 21523086

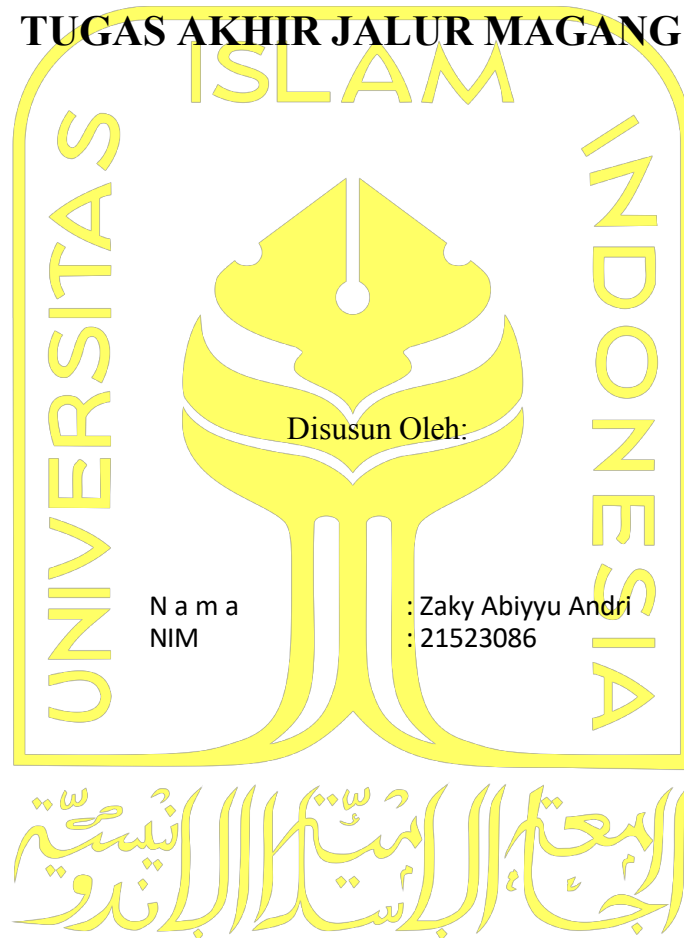
**PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA FAKULTAS
TEKNOLOGI INDUSTRI UNIVERSITAS ISLAM INDONESIA**

2025

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENGEMBANGAN *WEBSITE* PARIWISATA KARATON
NGAYOGYAKARTA HADININGRAT DENGAN ARSITEKTUR
JAMSTACK MENGGUNAKAN *ASTRO*

TUGAS AKHIR JALUR MAGANG



Yogyakarta, 1 Maret 2025

Pembimbing,

(Ari Sujarwo, S.Kom., MIT(Hons))

HALAMAN PENGESAHAN DOSEN PENGUJI
PENGEMBANGAN *WEBSITE* PARIWISATA KARATON
NGAYOGYAKARTA HADININGRAT DENGAN ARSITEKTUR
JAMSTACK* MENGGUNAKAN *ASTRO

TUGAS AKHIR JALUR MAGANG

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia Yogyakarta, 11 November 2025

Tim Penguji

Ari Sujarwo, S.KOM., MIT.



Anggota 1

Dr. Novi Setiani, S.T., M.T.



Anggota 2

Kurniawan Dwi Irianto S.T., M.SC.



Mengetahui,

Ketua Program Studi Informatika –
 Program Sarjana Fakultas Teknologi
 Industri

Universitas Islam Indonesia



(Thomas Hatta Fudholi, S. T., M.Eng., Ph.D.)



HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Zaky Abiyyu Andri
NIM : 21523086

Tugas akhir dengan judul:

PENGEMBANGAN *WEBSITE* PARIWISATA KARATON NGAYOGYAKARTA HADININGRAT DENGAN ARSITEKTUR *JAMSTACK* MENGGUNAKAN *ASTRO*

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 1 Maret 2025



Zaky Abiyyu Andri

HALAMAN PERSEMBAHAN

Alhamdulillah, segala puji syukur bagi Allah SWT Yang Maha Pengasih dan Maha Penyayang, yang telah memberikan petunjuk, rahmat serta hidayat-Nya sehingga penulis dapat menyelesaikan laporan Tugas Akhir ini. Sholawat serta salam senantiasa dilimpahkan kepada Nabi Muhammad SAW, beserta keluarganya, sahabat, dan pengikutnya hingga hari akhir. Laporan Tugas Akhir ini saya persembahkan kepada:

1. Kedua orangtua yang saya cintai, Bapak Andrios dan Ibu Miswanti yang senantiasa memberikan kasih sayang, doa, dan dukungan tanpa henti. Berkat cinta dan dorongan mereka, saya dapat menyelesaikan studi saya di Universitas Islam Indonesia. Tugas akhir ini saya persembahkan sebagai bentuk penghargaan dan rasa terima kasih yang tulus kepada mereka.
2. Bapak Ari Sujarwo, S.Kom., MIT(Hons). Selaku dosen pembimbing yang telah bimbingan, saran, dan masukan berharga selama proses penyusunan laporan tugas akhir ini.
3. Teman-teman yang selalu menyemangati, memberi dukungan, serta mengingatkan saya untuk tetap fokus dalam menyelesaikan penyusunan laporan tugas akhir ini.

Terima kasih saya ucapkan semoga Allah Swt selalu memberikan karuniaNya berupa kesehatan, rezeki, dan perlindungan. Amin ya Rabbal Alamin.

HALAMAN MOTO

“Hai orang-orang beriman apabila dikatakan kepadamu: “Berlapang-lapanglah dalam majelis”, maka lapangkanlah niscaya Allah akan memberi kelapangan untukmu. Dan apabila dikatakan: “Berdirilah kamu”, maka berdirilah, niscaya Allah akan meninggikan orang-orang yang beriman di antaramu dan orang-orang yang diberi ilmu pengetahuan beberapa derajat. Dan Allah Maha Mengetahui apa yang kamu kerjakan.”

(QS. Surat Al-Mujadalah ayat: 11)

“Ilmu adalah harta yang tidak akan berkurang, jadi carilah sebanyak-banyaknya ilmu.”

Ali bin Abi Thalib

“Barangsiapa yang pergi untuk menuntut ilmu, maka dia telah termasuk golongan sabilillah (orang yang menegakkan agama Allah) hingga ia pulang kembali.”

(HR. Tirmidzi)

“Sesungguhnya keutamaan seorang yang berilmu dibanding ahli ibadah, seperti keutamaan bulan di malam purnama dibanding seluruh bintang-bintang.”

(HR. Abu Dawud (3641), Ibnu Majah (223), dari hadits Abu Darda' Radhiallahu Anhu)

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Alhamdulillah penulis haturkan kepada Allah SWT, yang telah melimpahkan rahmat dan taufik serta hidayah-Nya, sehingga penulis dapat menyelesaikan laporan tugas akhir yang berjudul **“Pengembangan Website Pariwisata Karaton Ngayogyakarta Hadiningrat dengan Arsitektur Jamstack Menggunakan Astro”**. Sholawat dan salam semoga senantiasa tercurah kepada Nabi Muhammad Shallallahu Alaihi Wasallam, yang telah membawa umat Islam dari zaman kegelapan menuju zaman yang terang benderang dengan Islam dan ilmu pengetahuan.

Laporan tugas akhir ini disusun untuk memenuhi persyaratan kelulusan pada jalur magang di Fakultas Teknologi Industri Jurusan Informatika Universitas Islam Indonesia. Dalam penyusunan laporan akhir ini, tidak lepas dari arahan dan bimbingan dari berbagai pihak. Penulis mengucapkan rasa hormat dan terima kasih kepada semua pihak yang telah membantu. Pihak-pihak yang terkait antaranya sebagai berikut:

1. Kedua orang tua tercinta (Bapak Andrios, Ibu Miswanti) yang senantiasa memberikan dukungan, doa, serta bantuan positif, baik secara moral maupun material, selama menjalani kegiatan perkuliahan dan magang.
2. Seluruh keluarga yang senantiasa memberikan doa dan dukungan kepada penulis selama proses penyelesaian laporan akhir ini.
3. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc. selaku Ketua Jurusan Informatika Fakultas Teknologi Industri Universitas Islam Indonesia
4. Bapak DThomas Hatta Fudholi, S.T., M.Eng., Ph.D. selaku Ketua Program Studi Informatika Program Sarjana Fakultas Teknologi Industri Universitas Islam Indonesia.
5. Bapak Ari Sujarwo, S.Kom., MIT(Hons). Selaku Dosen Pembimbing yang telah bersedia membimbing dan mengarahkan penyusunan laporan tugas akhir ini.
6. Bapak dan Ibu dosen Informatika Fakultas Teknologi Industri Universitas Islam Indonesia yang telah memberi banyak ilmu, pelajaran, arahan, serta bimbingan yang sangat bermanfaat bagi penulis.
7. PT Lumintu Logic, beserta seluruh pimpinan dan staf yang telah memberi penulis kesempatan magang dan membantu penulis dalam menyelesaikan laporan akhir ini.

8. Bapak Happy Hartana sebagai mentor yang telah membimbing penulis dengan baik selama proses magang berlangsung.
9. Teman-teman Informatika UII Angkatan 2021 yang telah berjuang bersama selama kuliah.
10. Teman-teman seperjuangan yang selalu membantu dan memberi dukungan kepada penulis Tegar Bagus Hartadi, Mutiara Irdina, dan Shabrina Hazrati Bahirah.

Dalam penyusunan laporan ini, penulis menyadari bahwa laporan ini masih jauh dari kata sempurna. Penulis menerima kritik dan saran guna menyempurnakan laporan tugas akhir ini.

Wassalamu'alaikum Wr. Wb.

Yogyakarta, 1 Maret 2025



Zaky Abiyyu Andri

SARI

Website merupakan halaman informasi yang dapat diakses secara global melalui jaringan internet dan terdiri dari berbagai elemen seperti teks, gambar, suara, maupun animasi yang tersusun dalam satu kesatuan konten, baik bersifat statis maupun dinamis. Karaton Ngayogyakarta Hadiningrat sebagai pusat kebudayaan yang memiliki banyak koleksi bersejarah, mulai melakukan upaya digitalisasi untuk memperluas jangkauan informasi, salah satunya melalui pengembangan *website* pariwisata oleh unit Radyakartiyasa. Website ini bertujuan mempromosikan pariwisata Karaton, menarik minat wisatawan, serta mengenalkan berbagai aktivitas dan program yang ditawarkan kepada masyarakat luas, sehingga meningkatkan visibilitas budaya dan nilai historis yang dimiliki Karaton. Dalam proses pengembangannya, Radyakartiyasa berperan sebagai *product owner* yang kemudian bekerja sama dengan unit Kawedanan Tandha Yekti dan PT Lumintu Logic sebagai perusahaan pengembang sistem. Penulis, sebagai peserta magang di PT Lumintu Logic, terlibat secara langsung dalam pembangunan *website* tersebut menggunakan *framework Astro* dengan arsitektur *Jamstack*, yang dikenal karena kemampuannya menghasilkan performa tinggi. Penelitian ini berfokus pada penerapan Static Site Generation (SSG) untuk meningkatkan performa, fleksibilitas, *maintainability*, dan skalabilitas *website*, sekaligus memastikan bahwa informasi yang disajikan dapat diakses dengan cepat dan stabil oleh pengguna dari berbagai perangkat. Hasil penelitian menunjukkan bahwa penggunaan *Jamstack* dan *Astro* memberikan kontribusi signifikan terhadap kecepatan dan efisiensi *website*, serta dapat menjadi referensi untuk pengembangan *website* modern berbasis *Jamstack* di masa mendatang,, terutama bagi institusi budaya yang ingin memperkuat kehadiran digital melalui platform yang ringan, aman, dan mudah dikembangkan.

Kata kunci: [*Jamstack*, *Astro*, Static Site Generation.]

GLOSARIUM

<i>API (Application Programming Interface)</i>	Antarmuka yang memungkinkan komunikasi antar sistem perangkat lunak untuk saling bertukar data atau fungsi.
<i>Astro</i>	Framework <i>JavaScript</i> modern yang dapat membangun situs web statis dengan pendekatan <i>server-first</i> serta efisien dalam mengelola beban <i>JavaScript</i> di sisi klien.
<i>Build Time</i>	Waktu di mana halaman-halaman <i>website</i> dihasilkan secara statis sebelum proses <i>deployment</i> dilakukan ke <i>server</i> atau jaringan distribusi konten (CDN).
<i>Client-Side Rendering</i>	Metode rendering halaman web yang dilakukan sepenuhnya di sisi browser setelah <i>JavaScript</i> dimuat.
<i>Component-based Architecture</i>	Pendekatan dalam pengembangan antarmuka pengguna yang membagi tampilan menjadi komponen-komponen kecil dan dapat digunakan ulang.
<i>Content Delivery Network(CDN)</i>	Jaringan <i>server</i> yang tersebar di berbagai lokasi geografis untuk mendistribusikan konten secara lebih cepat dan efisien kepada user.
<i>Jamstack</i>	Antarmuka yang memungkinkan komunikasi antar sistem perangkat lunak untuk saling bertukar data atau fungsi.
<i>Routing</i>	Framework <i>JavaScript</i> modern yang dapat membangun situs web statis dengan

Markdown

pendekatan *server-first* serta efisien dalam mengelola beban *JavaScript* di sisi klien.

Waktu di mana halaman-halaman *website*

dihasilkan secara statis sebelum proses *deployment* dilakukan ke *server* atau jaringan distribusi konten (CDN).

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PENGESAHAN DOSEN PEMBIMBING	ii
HALAMAN PENGESAHAN DOSEN PENGUJI	iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR.....	iv
HALAMAN PERSEMBAHAN	v
HALAMAN MOTO	vi
KATA PENGANTAR.....	vii
SARI.....	ix
GLOSARIUM	x
DAFTAR ISI	xii
DAFTAR TABEL	xiv
DAFTAR GAMBAR.....	xv
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Ruang Lingkup.....	2
1.3 Tujuan	3
1.4 Manfaat	4
1.5 Sistematika Penulisan.....	4
BAB II KAJIAN PUSTAKA.....	5
2.1 Konsep Pariwisata.....	5
2.2 Konsep <i>Website</i> Pariwisata Karaton Ngayogyakarta Hadiningrat	5
2.3 <i>Website</i>	6
2.4 <i>Website</i> Static dan <i>Website</i> Dynamic	8
2.5 Jamstack.....	9
2.6 <i>Static Site Generator</i>	10
2.7 <i>Astro</i>	11
2.8 Content Delivery Network	12
2.9 Content Management System (CMS).....	13
2.10 Headless CMS	14
2.11 Time to First Byte (TTFB)	14
2.12 SpeedVitals.....	16
2.13 Tinjauan Pustaka	16
BAB III PELAKSANAAN MAGANG	19
3.1 Manajemen Proyek	19
3.1.1 Plan Human Centered Process	19
3.1.2 Schedule Weekly Meeting.....	20
3.1.3 UI/UX Design Radyakartiyasa	20
3.2 Proses dan Hasil Pelaksanaan Proyek	23
3.2.1 Fetching Data from CMS	23
3.2.2 Download Assets From CMS.....	30
3.2.3 the Assets In Public Folder.....	34
3.2.4 Creating Component for Front-end Project.....	36
3.2.5 Styles For Styling The Content <i>Website</i>	63
3.2.6 Script To Make It Interactive <i>Website</i>	64
3.2.7 The Contents to Show to Users	70

3.2.8	Configuration Of The Project	87
3.2.9	Activating the Build Process for Project Using <i>Astro</i>	91
3.2.10	CI/CD Flow	91
3.2.11	Melakukan pengujian TTFB <i>jamstack</i> dengan non <i>jamstack</i>	94
BAB IV REFLEKSI PELAKSANAAN MAGANG		100
4.1	Relevansi Akademik	100
4.2	Pembelajaran Magang.....	101
4.2.1	Manfaat.....	101
4.2.2	Kendala, Hambatan, dan Tantangan	101
BAB V PENUTUP		103
5.1	Kesimpulan	103
5.2	Saran.....	104
DAFTAR PUSTAKA.....		105
LAMPIRAN		109

DAFTAR TABEL

Tabel 3.1 Pengujian TTFB <i>JAMstack</i>	95
Tabel 3.2 Pengujian TTFB <i>non-JAMstack</i>	95
Tabel 3.3 Pengujian regional TTFB <i>JAMstack</i>	96
Tabel 3.4 Pengujian regional TTFB <i>non-JAMstack</i>	96

DAFTAR GAMBAR

Gambar 2.1 Proses <i>Client</i> dan <i>Server</i>	7
Gambar 2.2 Desain <i>Website Dynamic</i>	9
Gambar 2.3 Komponen <i>Astro</i> menampilkan 3 perbedaan <i>counter</i>	12
Gambar 2.4 <i>Content Delivery Network</i>	13
Gambar 2.5 Time to First Byte (TTFB)	15
Gambar 3.1 Navbar.....	20
Gambar 3.2 Footer.....	21
Gambar 3.3 Desain konten	21
Gambar 3.4 Konten tentang Kagungan	22
Gambar 3.5 Konten destinasi	22
Gambar 3.6 Konten event.....	23
Gambar 3.7 Struktur folder	24
Gambar 3.8 <i>Host</i> dan <i>headers</i>	24
Gambar 3.9 <i>Session</i> token	24
Gambar 3.10 Cms images	24
Gambar 3.11 <i>getLang</i>	25
Gambar 3.12 <i>FetchLang</i>	25
Gambar 3.13 <i>FetchIdSlug</i>	26
Gambar 3.14 <i>Fetch_data</i>	27
Gambar 3.15 <i>Slugify</i>	28
Gambar 3.16 <i>Date_format</i>	28
Gambar 3.17 <i>Strip_tags</i>	29
Gambar 3.18 <i>File_handle</i>	29
Gambar 3.19 <i>Get_interface</i>	30
Gambar 3.20 <i>Variable ass</i>	31
Gambar 3.21 Kode <i>promise</i>	31
Gambar 3.22 <i>Variable files</i>	31
Gambar 3.23 Mengecek format file	32
Gambar 3.24 <i>LoadingAnimation</i>	33
Gambar 3.25 Mengimpor fungsi dan data gambar.....	37

Gambar 3.26 <i>Data_navigations</i>	37
Gambar 3.27 <i>Main navigation</i>	38
Gambar 3.28 <i>Header-action</i>	38
Gambar 3.29 <i>Dropdown</i>	38
Gambar 3.30 <i>Closing button</i>	38
Gambar 3.31 <i>Nav-link</i>	38
Gambar 3.32 <i>Header-action</i>	39
Gambar 3.33 <i>Dropdown-menu</i>	39
Gambar 3.34 <i>Nav-item</i>	39
Gambar 3.35 <i>Storing to sessionStorage</i>	40
Gambar 3.36 <i>Mengelola bahasa</i>	40
Gambar 3.37 <i>wScroll</i>	40
Gambar 3.38 <i>Import package</i>	41
Gambar 3.39 <i>Variables currentYear</i>	41
Gambar 3.40 <i>Class footer</i>	42
Gambar 3.41 <i>Footer-column</i>	42
Gambar 3.42 <i>Footer-brand</i>	42
Gambar 3.43 <i>Deskripsi terjemahan sesuai bahasa</i>	43
Gambar 3.44 <i>Sitemap navigasi</i>	43
Gambar 3.45 <i>Daftar navigasi</i>	43
Gambar 3.46 <i>Tambahan konten navigasi</i>	44
Gambar 3.47 <i>Copyright</i>	44
Gambar 3.48 <i>Social-media</i>	44
Gambar 3.49 <i>Whatsapp</i>	45
Gambar 3.50 <i>Email</i>	45
Gambar 3.51 <i>HTML meta tags</i>	46
Gambar 3.52 <i>Meta apple-mobile-web-app-capable</i>	47
Gambar 3.53 <i>html semantics tags</i>	47
Gambar 3. 54 <i>Script jquery</i>	48
Gambar 3.55 <i>Import about.astro</i>	49
Gambar 3.56 <i>Section about.astro</i>	49
Gambar 3.57 <i>Body-area about.astro</i>	50

Gambar 3.58 <i>Swiper event.astro</i>	52
Gambar 3.59 <i>Tag faq.astro</i>	53
Gambar 3.60 <i>Swiper-container gallery.astro</i>	54
Gambar 3.61 <i>Swiper gallery.astro</i>	55
Gambar 3.62 <i>Swiper cover slide hero.astro</i>	56
Gambar 3.63 <i>New swiper hero.astro</i>	57
Gambar 3.64 <i>Intro-card hero.astro</i>	57
Gambar 3.65 <i>Swiper wrapper testimonials.astro</i>	58
Gambar 3.66 <i>Pagination-area testimonials.astro</i>	59
Gambar 3.67 <i>Swiper testimonials.astro</i>	60
Gambar 3.68 <i>Scroll backtotop.astro</i>	61
Gambar 3.69 <i>Scrollto backtotop.astro</i>	61
Gambar 3.70 <i>Props sectiontitle.astro</i>	62
Gambar 3.71 <i>Section-title sectiontitle.astro</i>	62
Gambar 3.72 <i>Badge sectiontitle.astro</i>	62
Gambar 3.73 <i>Section-heading sectiontitle.astro</i>	62
Gambar 3.74 <i>Deskripsi & tautan sectiontitle.astro</i>	63
Gambar 3.75 <i>Tampilan dropdown main.js</i>	65
Gambar 3.76 <i>Menampilkan & menyembunyikan main.js</i>	65
Gambar 3.77 <i>Magnific popup main.js</i>	66
Gambar 3.78 <i>Berbagi main.js</i>	67
Gambar 3.79 <i>Pagination_handle pagination.min.js</i>	68
Gambar 3.80 <i>Plugin plugins.js</i>	69
Gambar 3.81 <i>Menerima dua parameter middleware.js</i>	70
Gambar 3.82 <i>Reference path env.d.ts</i>	70
Gambar 3.83 <i>Not error 404.astro</i>	71
Gambar 3.84 <i>Komen meta index.astro</i>	71
Gambar 3.85 <i>Import about.astro</i>	72
Gambar 3.86 <i>GetStaticPaths about.astro</i>	72
Gambar 3.87 <i>Mengambil data about.astro</i>	72
Gambar 3.88 <i>Struktur about.astro</i>	72
Gambar 3.89 <i>Single-content about.astro</i>	73

Gambar 3.90 <i>Import homepage index.astro</i>	73
Gambar 3.91 <i>GetStaticPaths homepage index.astro</i>	74
Gambar 3.92 <i>Fetching data homepage index.astro</i>	74
Gambar 3.93 <i>Layout homepage index.astro</i>	74
Gambar 3. 94 <i>About & Destination homepage index.astro</i>	75
Gambar 3.95 <i>Promo section homepage index.astro</i>	75
Gambar 3. 96 <i>Couple section homepage index.astro</i>	76
Gambar 3.97 <i>Contact section homepage index.astro</i>	76
Gambar 3.98 <i>Import destination index.astro</i>	77
Gambar 3.99 <i>GetStaticPaths() destination index.astro</i>	77
Gambar 3.100 <i>Hero-2-section destination index.astro</i>	78
Gambar 3.101 <i>Section destination index.astro</i>	78
Gambar 3.102 <i>Pagination destination index.astro</i>	78
Gambar 3.103 <i>Import destination [id]-[slug].astro</i>	79
Gambar 3.104 <i>GetStaticPaths destination [id]-[slug].astro</i>	79
Gambar 3.105 <i>Images destination [id]-[slug].astro</i>	80
Gambar 3.106 <i>Deskripsi & daftar destination [id]-[slug].astro</i>	80
Gambar 3.107 <i>Jam operasional destination [id]-[slug].astro</i>	81
Gambar 3.108 <i>Maps & Tombol tiket destination [id]-[slug].astro</i>	81
Gambar 3.109 <i>Import event index.astro</i>	82
Gambar 3.110 <i>GetStaticPaths event index.astro</i>	82
Gambar 3.111 <i>Layout event index.astro</i>	82
Gambar 3.112 <i>Daftar konten event index.astro</i>	83
Gambar 3.113 <i>Penjelasan event index.astro</i>	83
Gambar 3.114 <i>Pagination event index.astro</i>	83
Gambar 3.115 <i>Import event [id]-[slug].astro</i>	84
Gambar 3.116 <i>GetStaticPaths event [id]-[slug].astro</i>	84
Gambar 3.117 <i>Parameter event [id]-[slug].astro</i>	85
Gambar 3.118 <i>Header event [id]-[slug].astro</i>	85
Gambar 3.119 <i>Informasi event [id]-[slug].astro</i>	86
Gambar 3.120 <i>Tombol tiket URL event [id]-[slug].astro</i>	86
Gambar 3.121 <i>Navigasi Dos & Don'ts event [id]-[slug].astro</i>	86

Gambar 3. 122 Informasi <i>Don'ts & Dos event [id]-[slug].astro</i>	87
Gambar 3.123 Logo <i>event [id]-[slug].astro</i>	87
Gambar 3.124 <i>Import astro.config.mjs</i>	88
Gambar 3.125 <i>DefineConfig astro.config.mjs</i>	88
Gambar 3.126 Il8n <i>astro.config.mjs</i>	89
Gambar 3.127 <i>Server astro.config.mjs</i>	89
Gambar 3. 128 <i>Projects name package.json</i>	90
Gambar 3.129 <i>Scripts package.json</i>	90
Gambar 3.130 <i>Dependencies package.json</i>	90
Gambar 3.131 Parameter TTFB SpeedVitals(Patel M, 2025)	94
Gambar 3. 132 Linear Regression <i>JAMstack</i> dan <i>Non-JAMstack</i>	98

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi yang semakin pesat mengakibatkan terjadinya transisi ke digital. Digitalisasi merupakan proses transformasi media informasi analog ke media digital yang diproses melalui komputer (Bakhtiar et al., 2019). Teknologi digital memberikan berbagai dampak positif bagi manusia di antaranya yaitu mempermudah akses terhadap informasi secara lebih cepat melalui internet, meningkatkan inovasi teknologi digital pada beberapa aspek kehidupan, serta meningkatkan kualitas sumber daya manusia dengan mengembangkan dan memanfaatkan teknologi informasi. Salah satu pemanfaatan digitalisasi adalah dikembangkannya *Website* pariwisata yang bertujuan untuk sebagai sarana komunikasi dengan masyarakat terkait program pariwisata yang dimiliki, penyaluran informasi secara cepat dan akurat, serta memperluas jangkauan pemasaran marketing wisata (Susanto et al., 2022).

Sejak tahun 2024 Karaton Ngayogyakarta Hadiningrat mulai mengembangkan *website* sebagai upaya marketing pariwisata. Diprakarsai oleh GKR Bendera dengan unit radyakartiyasa sebagai product owner untuk disampaikan dan diambil alih secara efektif kepada tim Tandha Yekti. Tim tersebut dipimpin langsung oleh Happy Hartana dengan pengawasan paling puncak berada pada GKR Hayu. Perlunya *website* pariwisata untuk memisahkan informasi yang bersifat marketing pariwisata dari informasi yang bersifat edukasi yang telah difasilitasi oleh situs resmi karaton, yang diharapkan tidak membingungkan masyarakat.

Berdasarkan latar belakang tersebut, dikembangkanlah sebuah *website* yang diurus oleh Radyakartiyasa sebagai bagian dari kawedanan nityabudoyo, yang merupakan unit pengelola museum, tempat wisata, dan objek wisata milik Karaton Ngayogyakarta Hadiningrat. Pada saat magang, nama resmi *website* ini masih belum ditetapkan, dan rencananya akan diumumkan secara resmi bertepatan dengan peluncuran *website* pada tanggal 23 agustus 2025. *Website* ini merupakan sistem untuk menampilkan destinasi dan acara yang diselenggarakan oleh Karaton Yogyakarta. *Website* dapat diakses secara langsung melalui peramban yang tersedia. Proses pengembangan *website* dilakukan secara bertahap, sementara ini tersedia informasi mengenai destinasi di Karaton Ngayogyakarta, dan berbagai acara dan pameran. Tujuan dikembangkannya sistem ini adalah untuk mempermudah penyajian informasi mengenai destinasi dan acara yang ditawarkan oleh Karaton Yogyakarta. Selain itu,

sistem ini juga bertujuan untuk meningkatkan jumlah wisatawan yang berkunjung ke destinasi di dalam kraton, menarik lebih banyak peserta untuk mengikuti berbagai aktivitas yang ditawarkan, serta memperluas variasi kegiatan yang dapat diselenggarakan oleh pihak pengelola. Saat ini, *Website* Radyakartiyasa sedang dalam proses pengembangan guna meningkatkan performa dan layanan, sehingga dapat lebih optimal dalam memenuhi kebutuhan pengguna saat mengaksesnya.

Website Radyakartiyasa sendiri dikembangkan secara efektif dan efisien dengan pendekatan *Arsitektur Jamstack* yang menggunakan *framework Astro*. Tujuan *Jamstack* adalah untuk meningkatkan performa, dan skalabilitas situs (Nurhidayat T, 2022). *Astro* adalah sebuah *framework* pengembangan situs web yang berfungsi sebagai Static Site Generator(SSG),dirancang untuk membangun situs web dengan pendekatan *Jamstack*.Tugas akhir ini berisi bagaimana *Jamstack* itu bisa secara efektif diwujudkan dengan SSG yang disediakan oleh *astro*.

Pengembangan *website* Radyakartiyasa menggunakan pendekatan *Jamstack*. *Jamstack* merupakan pendekatan yang berfokus kepada pemisahan *frontend* dan *backend* dengan memanfaatkan penggunaan API serta menghasilkan halaman yang statis. Pendekatan *Jamstack* dapat berpengaruh secara positif karena fleksibilitas dan efisiensi dalam pengembangan *website*. Selain itu, *Astro* memiliki kemampuan yang memungkinkan penggunaan berbagai framework seperti *React*, dan *Vue* di dalam satu proyek. Dengan kemampuan ini, developer dari latar belakang teknologi yang berbeda tetap dapat berkontribusi tanpa harus sepenuhnya mempelajari *stack* baru, sehingga membuat proses kolaborasi lebih fleksibel dan efisien.

1.2 Ruang Lingkup

Pelaksanaan magang di PT Lumintu Logic berlangsung dari tanggal 19 September 2024 sampai dengan 19 Desember 2024. Kantor PT Lumintu Logic terletak di Jl. Tegal Arum Gamping Kidul, Tamantirto, Kec. Gamping, Kabupaten Sleman, Daerah Istimewa Yogyakarta. PT Lumintu Logic merupakan perusahaan yang menawarkan SaaS (*Software as a Service*). Produk IT yang dikembangkan berupa situs web yang dapat digunakan di berbagai sektor, seperti pendidikan, rumah sakit, perusahaan, dan pemerintahan. Pada akhir tahun 2023 PT Lumintu Logic Indonesia terdaftar sebagai Lembaga Pelatihan Vokasi di bidang Teknologi Informasi dengan program TRIPLE C yang memberikan 6 bulan magang dan sertifikasi. Terdapat 2 posisi

untuk peserta magang PT Lumintu Logic, yaitu *front end developer*, dan *back end developer*. Adapun peran dan posisi penulis selama kegiatan magang adalah sebagai *front end developer*. Proyek yang akan penulis ajukan dalam laporan tugas akhir ini adalah pengembangan *website* pariwisata Karaton Ngayogyakarta Hadiningrat dengan arsitektur *Jamstack* menggunakan *Astro*. Adapun seluruh aktivitas yang dilakukan selama magang adalah sebagai berikut:

Terlibat dalam proyek Radyakartiyasa yaitu sebuah proyek pengembangan *website* pariwisata untuk Karaton Ngayogyakarta Hadiningrat. Ruang lingkup pekerjaan adalah terlibat dalam pengembangan situs web di sisi *front end* dengan pendekatan *Jamstack*. Proses dimulai dengan mengubah desain UI/UX yang sudah disediakan menjadi struktur *HTML*, *styling CSS*, dan interaktivitas *JavaScript* responsif. Kemudian, situs tersebut dimigrasikan ke *Astro* untuk diubah menjadi situs statis SSG, yang kemudian akan di-deploy ke platform *Content Delivery Network* seperti *Cloudflare*. Konten untuk situs web akan diambil dari backend menggunakan API, yang dapat berasal dari berbagai sumber, asalkan dapat diakses melalui API. Proyek ini melibatkan 4 orang dari seluruh peserta magang dengan pembagian tim *front end* berisi dua orang dan tim *back end* berisi 2 orang. Seluruh hasil pengembangan *website* tersebut kemudian dievaluasi dan dinilai oleh *supervisor* PT Lumintu Logic. Hasil pengembangan *website* Radyakartiyasa akan dijadikan *insight* tambahan pada tahap pengembangan oleh tim *developer* PT Lumintu Logic. Adapun seluruh aktivitas yang dilakukan selama magang meliputi melakukan *scaffold* proyek *Astro* sebagai fondasi awal, membuat repositori Git di *Github* untuk kebutuhan kolaborasi, serta membangun komponen umum seperti *navbar* dan *footer* berdasarkan desain UI/UX dari *Figma*. Penulis juga mengembangkan komponen halaman *homepage* yang terdiri dari berbagai *section* (*Hero*, *About us*, *Destination & Event Slider*, *Promo*, *Home Gallery*, *Testimonial*, *FAQ*, dan *Contact Us*) menggunakan *HTML* dan *CSS*. Setelah seluruh tampilan statis selesai, struktur halaman dipindahkan ke dalam format *Astro* yang lebih terorganisir, mencakup halaman destinasi, *event*, serta halaman indeks masing-masing. Penulis membuat *frontmatter mockup* untuk kebutuhan *permalink* pada setiap halaman.

1.3 Tujuan

Tujuan laporan tugas akhir yang diambil dari ruang lingkup magang sebagai fokus yaitu:

Tujuan dilakukannya proses pengembangan *Website* Pariwisata Karaton Ngayogyakarta Hadiningrat adalah untuk menghasilkan situs web statis (SSG) dengan pendekatan *Jamstack* menggunakan *Astro* untuk meningkatkan performa, fleksibilitas, *maintainability* dan skalabilitas situs web.

1.4 Manfaat

Manfaat dilakukannya proses pengembangan *Website* Pariwisata Karaton Ngayogyakarta Hadiningrat yaitu Radyakartiyasa dengan pendekatan *Jamstack* menggunakan *Astro*:

1. Meningkatkan performa *website* melalui konten statis yang dihasilkan sebelumnya dengan pendekatan *Jamstack*, sehingga memberikan *user experience* yang lebih baik dan efisien.
2. Fleksibilitas *Jamstack* adalah kemampuan menghasilkan halaman statis dengan menggunakan komponen dari *front end framework Astro* yang tersedia sesuai kebutuhan proyek.
3. *Maintainability* dengan *Astro* yang bisa melakukan *refactoring* dengan membagi struktur laman menjadi komponen-komponen yang dapat digunakan berulang (*reusable components*), sehingga memudahkan pemeliharaan dengan mengurangi duplikasi kode dan meningkatkan efisiensi.
4. Kemudahan skalabilitas untuk *website* Radyakartiyasa dengan melayani jumlah pengunjung yang lebih banyak tanpa penurunan kinerja sehingga menjaga *user experience* tetap optimal.

1.5 Sistematika Penulisan

Sistematika penulisan disusun untuk memudahkan dalam memahami laporan tugas akhir ini. Sistematika penulisan laporan tugas akhir adalah sebagai berikut:

a. Bab1: Pendahuluan

Bab ini berisi latar belakang, ruang lingkup magang, tujuan dan manfaat penelitian, serta sistematika penulisan laporan Tugas Akhir.

b. Bab 2: Landasan Teori

Bab ini berisi pembahasan beberapa teori yang mendukung proses penyusunan laporan Tugas Akhir.

c. Bab 3: Pelaksanaan Magang

Bab ini berisi tahapan proses serta hasil pengembangan *Website* Parawisata Karaton Ngayogyakarta dengan pendekatan *Jamstack* menggunakan *Astro*.

d. Bab 4: Kesimpulan dan Saran

Bab ini berisi kesimpulan dari seluruh hasil pengerjaan Tugas Akhir, serta saran untuk pengembangan selanjutnya agar dapat terlaksana dengan lebih baik.

BAB II KAJIAN PUSTAKA

2.1 Konsep Pariwisata

Secara etimologi, pariwisata berasal dari kata ‘wisata’ yang berarti perjalanan (*traveling*), kata wisatawan yang merujuk pada orang yang melakukan perjalanan (*traveler*), dan ‘kepariwisataan’ yang berarti segala aktivitas atau hal yang berhubungan dengan pariwisata (G. Y. K. Pradana, 2019). Kegiatan pariwisata memiliki dampak sosial, ekonomi, dan budaya yang muncul sebagai efek dari perjalanan wisata (G. Y. K. Pradana, 2019). Pariwisata mencakup berbagai kegiatan wisata yang didukung oleh fasilitas dan layanan yang disediakan oleh masyarakat, pengusaha, Pemerintah dan Pemerintah Daerah (Akib, 2020). Pariwisata yang dilakukan wisatawan dengan perpindahan sementara ke beberapa tempat tujuan selain tempat kerja dan tempat tinggal (Mason dalam G. Y. K. Pradana, 2019).

Perjalanan wisata dilakukan dengan durasi minimal 2 jam dan maksimal 6 bulan, yang berorientasikan untuk kesenangan, kesehatan, rekreasi, pendidikan, relaksasi, dll di tempat tujuan wisata (G. Y. K. Pradana, 2019). Pariwisata merupakan suatu bentuk perjalanan yang dilakukan seseorang dalam jangka waktu tertentu, berpindah dari satu tempat ke tempat lain dengan meninggalkan lokasi asalnya (Kane, M. J., & Tucker, H dalam Akib, 2020). Perjalanan ini dilakukan berdasarkan perencanaan, bukan untuk bekerja atau mencari penghasilan di tempat yang dituju, melainkan semata-mata untuk menikmati kegiatan hiburan, rekreasi, atau memenuhi berbagai keinginan pribadi (Kane, M. J., & Tucker, H dalam Akib, 2020).

2.2 Konsep *Website* Pariwisata Karaton Ngayogyakarta Hadiningrat

Happy hartana menjelaskan bahwa *Website* ini merupakan hasil dari inisiatif GKR Bendera yang merupakan Penghageng I Kawedanan Hageng Nityabudoyo (komunikasi pribadi, 17 Mei 2025). Beliau berharap agar hadir sebuah platform yang secara khusus menyoroti destinasi dan *event* wisata, mengingat konten semacam ini bersifat lebih dinamis dibandingkan artikel pengetahuan umum atau sejarah yang saat ini telah difasilitasi melalui situs utama kratonjogja.id (H. Hartana, komunikasi pribadi, 17 Mei 2025). Saat ini, cakupan pembahasan destinasi dan *event* di dalam *website* masih terbatas pada yang berada bawah naungan Kawedanan Radyakartiyasa (H. Hartana, komunikasi pribadi, 17 Mei 2025). Namun, ruanglingkup tersebut tidak hanya mencakup museum atau lokasi yang berada di dalam

kawasan Karaton atau Kedhaton saja, melainkan juga mencakup lokasi di luar area tersebut, seperti Tamansari dan Museum Kereta Wahanarata (H. Hartana, komunikasi pribadi, 17 Mei 2025). *Website* ini akan menjadi *subdomain* dari *kratonjogja.id* dan berada di bawah pengelolaan Kawedanan Radyakartiyasa, sebuah unit yang fokus pada pengembangan pariwisata di Karaton Ngayogyakarta Hadiningrat (H. Hartana, komunikasi pribadi, 17 Mei 2025). Kawedanan Radyakartiyasa sendiri berada dalam naungan Kawedanan Hageng Nityabudoyo (H. Hartana, komunikasi pribadi, 17 Mei 2025). Kawedanan Hageng Nityabudoyo yang bertugas mengurus kebudayaan Karaton Ngayogyakarta seperti museum dan pariwisata (Juno, 2021). Rencananya, *website* ini akan diluncurkan dengan alamat *tourism.kratonjogja.id* (H. Hartana, komunikasi pribadi, 17 Mei 2025).

2.3 *Website*

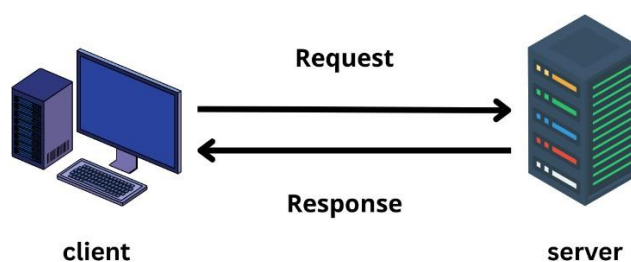
Website merupakan halaman informasi yang disediakan melalui jaringan internet sehingga bisa di akses oleh seluruh dunia (Akgül, 2021). *Website* terdiri dari komponen-komponen berupa teks, gambar, video dan sebagainya (Akgül, 2021). Untuk mengakses *website* perlu menggunakan perangkat lunak yang disebut browser (Nurlailah & Wardani, 2023). Membuat sebuah halaman web perlu menggunakan bahasa kode berbasis teks yaitu HTML (*Hyper Text Markup Language*) (Andi dalam Pranata et al., 2015). Protokol yang digunakan untuk mengakses *website* adalah *Hypertext Transfer Protocol/Secure* yang mencegah kemungkinan kebocoran informasi penting selama proses komunikasi berlangsung antara *user* dengan *web server* atau sebaliknya (Zabar & Novianto, 2015).

Website menggunakan komponen *Web Server* yang memberikan layanan data untuk menerima permintaan HTTP/S yang dikirim oleh klien melalui *web browser* dan mengirimkan kembali hasilnya dalam bentuk halaman web berbentuk dokumen HTML (Ihsan et al., 2023). *Web server* berguna sebagai tempat aplikasi web dan sebagai penerima *request* dari *client* (Warman I. M., Zaini A. dalam Ihsan et al., 2023). Dua contoh dari *web server* yang sering digunakan saat ini adalah Apache dan Internet Information Services (Ihsan et al., 2023). *Web Browser* sebagai perangkat lunak yang digunakan untuk mengoperasikan internet untuk mengakses *website* (Rahmadoni dalam Abdiati et al., 2021). *Web browser* berfungsi menerima dan menyajikan sumber informasi dari internet (Rusdi et al., 2022). Sumber informasi dapat berupa halaman web, gambar, video, atau jenis konten lainnya yang berada di internet (Rusdi

et al., 2022). Contoh web *browser* yang banyak digunakan adalah *Chrome* dan *Firefox*, *Opera mini*, dan UC Browser (Abdiati et al., 2021).

Website yang memiliki data akan tersimpan di *Database Server* sebagai tempat penyimpanan data (Suryanto, 2015). Database ini umumnya dipanggil *Database Management System* (DBMS), digunakan untuk mengelola suatu basis data dan menjadi perantara user dengan *database* (Hidayah et al., 2019). Contoh *database* seperti MySQL, Oracle dan lain-lain (Hidayah et al., 2019). *Website* menggunakan protokol untuk berkomunikasi antara browser web dan *server* (Nazarina et al., 2024). Protokol yang digunakan untuk membuat komunikasi antara *client* dan *server* adalah HTTP (Zabar & Novianto, 2015). Sebuah *client* HTTP, seperti web *browser*, umumnya memulai permintaan dengan membangun koneksi TCP/IP, biasanya melalui *port* 80 (Zabar & Novianto, 2015). HTTP yang melalui *Socket Layer* yang terenkripsi SSL (*Secure*) atau *Transport Layer Security* (TLS) disebut HTTPS (Zabar & Novianto, 2015). *port default* TCP https adalah 443 (Zabar & Novianto, 2015).

Dalam proses pengolahan *website* secara umum terdapat konsep *Client-Server* (Hidayatullah & Arief, 2016).



Gambar 2.1 Proses Client dan Server

Seperti pada Gambar 2.1 untuk *client* merupakan sistem atau operasi yang membuat permintaan(*request*) untuk data atau layanan ke *server* (Maryani et al., 2018). Sementara itu, *server* adalah sistem atau proses yang menyediakan(*response*) data atau layanan yang diminta oleh *client* (Maryani et al., 2018). Seiring dengan perkembangan teknologi *client-server*, beberapa arsitektur telah dikembangkan, di antaranya adalah 1-Tier (*standalone*), 2-Tier dan N-Tier (Woro, 2010):

a. One-Tier

Konsep *One-Tier* adalah sebuah komputer yang mengakses sebuah database dari komputer sendiri. Dengan kata lain, aplikasi antarmuka *user* dan aplikasi DMBS terdapat pada komputer yang sama.

b. Two-Tier

Pada arsitektur ini terbagi menjadi dua bagian yaitu *Client-tier* dan *server-tier* dengan tingkat pertama adalah *client* dan tingkat kedua adalah *server*. *Client* untuk menyediakan antarmuka untuk *user*, permintaan (*request data*) ke DBMS *Server*, dll. Sementara *server* mengelola *database* yang dapat dipanggil *client* untuk mengolah data.

c. N-tier

Arsitektur ini *server* dibagi menjadi 2 yaitu *server* yang dipakai sebagai business object (*middle tier*) dan satu *server* yang hanya menyimpan *database* (*server tier*) yang bisa disebut sebagai 3-tier. Lapisan pertama *client* yang menampilkan antarmuka kepada *user*. Lapisan kedua HTTP *server* yang menerjemahkan script *server side*. Kemudian, lapisan ketiga adalah *server* yang menyediakan database yang diminta oleh web atau HTTP *server*.

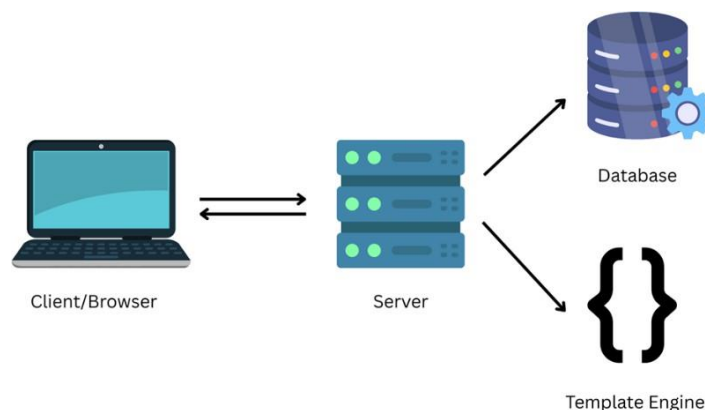
2.4 Website Static dan Website Dynamic

Website statis (Static Website) adalah jenis *website* yang pengunjung tidak memiliki peran aktif selain hanya melihat konten yang disajikan (Astuti & Agustina, 2022). Proses pembaruan kontennya dilakukan secara manual, yaitu dengan membuka halaman web dan mengetikkan perubahan secara langsung di dalamnya (Astuti & Agustina, 2022). *Website* statis adalah jenis situs web yang kontennya hanya dapat diubah secara manual, biasanya oleh individu, dan dikelola menggunakan berbagai *software* editor (Saraswati, 2011). *Website* statis adalah jenis situs yang terdiri dari file HTML tetap di *server*, dimana setiap halaman menampilkan konten yang sama untuk semua pengunjung (Mildenberger, 2024). Dibuat dengan HTML, CSS, dan *JavaScript*, situs ini cepat diakses karena tidak memerlukan pemrosesan *server* saat dimuat (Mildenberger, 2024).

Kecepatannya mendukung SEO dan memberikan pengalaman pengguna yang baik (Mildenberger, 2024). Namun, keterbatasannya terletak pada fungsionalitas seperti tidak mendukung login, tampilan data secara real time, atau forum interaktif, karena tidak adanya proses pemrograman di sisi *server* (Mildenberger, 2024).

Website Dinamis (Dynamic Website) adalah situs web yang memungkinkan pengguna untuk mengubah atau memperbarui konten pada halaman tertentu langsung melalui browser (M. Kurniawan et al., 2022). Tidak seperti situs statis yang hanya menampilkan konten tetap, *website* dinamis menghasilkan konten secara real-time berdasarkan interaksi pengguna, input data, atau kondisi tertentu (Mildenberger, 2024). Hal ini dimungkinkan berkat penggunaan

bahasa pemrograman *server-side* seperti PHP, Python, atau Framework *Javascript* dengan dukungan *database* (Mildenberger, 2024) dengan desain seperti pada Gambar 2.2.



Gambar 2.2 Desain *Website Dynamic*

Keunggulan utamanya adalah fleksibilitas tinggi dan pengalaman pengguna yang responsif (Mildenberger, 2024). Konten bisa disesuaikan dengan perilaku pengguna dan diperbarui secara berkala (Mildenberger, 2024). Namun, *website* dinamis juga lebih kompleks untuk dikembangkan dan memerlukan lebih banyak sumber daya, baik dari sisi pemrograman maupun biaya (Mildenberger, 2024). Selain itu, waktu muat halaman cenderung lebih lambat dibandingkan situs statis karena konten dihasilkan secara langsung saat diminta sehingga berdampak dengan pengalaman *user* dan SEO (Mildenberger, 2024).

2.5 Jamstack

Jamstack atau kepanjangannya *JavaScript*, *API*, *MARKUP* memisahkan *front end* dan *back end* (Hanafi, 2023). membangun *front end* aplikasi dengan *HTML*, *CSS*, dan *JavaScript*, dan *back end* aplikasi diterapkan pada *API* (Hanafi, 2023). *JavaScript* akan menangani logika dan interaksi dalam aplikasi web, *API* digunakan jika menggunakan layanan *back end* pihak ketiga, Markup adalah *template* atau struktur halaman web yang dihasilkan secara statis oleh *SSG* (E. R. Pradana, 2025). *Jamstack* adalah arsitektur di mana situs web dikirim secara statis, seperti menyajikan *HTML* dari hosting statis atau CDN, tetapi menyediakan konten dinamis dan pengalaman interaktif melalui *Javascript* (Fayock C, 2022). Konsep *Jamstack* memiliki beberapa kunci elemen yang perlu diperhatikan sebagai berikut (Camden & Rinaldi, 2022):

- a. Situs *Jamstack* terutama dibangun dengan menggunakan file statis. Artinya, file situs

ini tidak dibuat secara dinamis oleh *server* saat pengguna meminta halaman, melainkan file situs dibuat saat proses pembangunan. Jadi, setiap kali pengguna membuka halaman tertentu, mereka akan mendapatkan file statis yang sama. Namun, ini tidak berarti konten situs selalu statis. Situs *Jamstack* modern menawarkan berbagai cara untuk menampilkan konten halaman, termasuk menggunakan rendering statis atau *server-side*.

- b. Situs *Jamstack* dibangun menggunakan *static site generation*. Aset statis di situs *Jamstack* dihasilkan menggunakan *static site generator* (SSG). Secara sederhana, SSG adalah alat yang menggabungkan *template* dengan konten. Konten bisa disimpan dalam file seperti *Markdown*, *YAML*, *JSON*, atau diambil dari API. Konten dan *template* ini digabungkan untuk menghasilkan *HTML*, *CSS*, dan *JavaScript* situs secara dinamis. Proses ini mirip dengan yang dilakukan *server* web dinamis seperti PHP, tetapi perbedaannya, sebagian besar proses ini terjadi saat *build time* sebelum situs di *deploy*.
- c. Situs *Jamstack* memanfaatkan API. Perbedaan utama antara situs *Jamstack* dan situs statis biasa adalah meskipun situs *Jamstack* terdiri dari aset statis, ia bisa sangat dinamis. Kunci pertama untuk menciptakan fungsionalitas dinamis ini adalah penggunaan API. API ini dapat dipanggil oleh browser saat runtime atau bahkan dipanggil oleh *static site generator* saat proses pembangunan.
- d. Situs *Jamstack* menggunakan *JavaScript* untuk fungsionalitas dinamis. Kunci kedua untuk membuat situs *Jamstack* menjadi dinamis adalah kemampuannya untuk memanggil API dan layanan lainnya secara asinkron pada klien melalui *JavaScript*. *JavaScript* memungkinkan aset statis berubah secara dinamis melalui manipulasi model objek dokumen (*DOM*).

2.6 *Static Site Generator*

Secara sederhana *Static Site Generator* (SSG) adalah alat yang menghasilkan situs web HTML statis berdasarkan data mentah dan *template* yang ditentukan (Hanafi, 2022). *Static Site Generators* adalah alat yang digunakan untuk membuat situs web yang hanya terdiri dari HTML, CSS, dan *JavaScript* (Newson, 2017). Situs yang dihasilkan oleh SSG akan ditampilkan kepada pengguna sesuai dengan apa adanya yang ada di *server* web (Newson, 2017). Sebelum itu situs statis adalah halaman web yang menyajikan informasi tetap yang tidak berubah berdasarkan interaksi pengunjung (Mirza, 2024). Pada dasarnya SSG mengotomatiskan tugas pengkodean halaman HTML individual sehingga tidak menulis kode

secara manual untuk setiap halaman dan menyiapkan halaman tersebut untuk disajikan kepada pengunjung *website* (R. Kurniawan, 2021). Perbedaanya dengan *Content Management System* yang menyimpan konten di database dan saat terjadi *request* dari *user* akan menjalankan kueri untuk konten yang diinginkan sementara SSG hampir mirip tetapi dilakukan *build time* sebelum adanya permintaan pengguna (R. Kurniawan, 2021). ini akan membuat waktu loading lebih singkat dikarenakan halaman telah tersedia sebelumnya tanpa harus melakukan kueri ke dalam database (R. Kurniawan, 2021). *Static Site Generator* biasanya terbuat dari (Newson, 2017):

- a. Sebuah *templating language* untuk *website layout* dan tema situs web.
- b. Bahasa *markup* untuk pembuatan konten web (misalnya *Markdown*).
- c. Sebuah *server* pengembangan lokal untuk *preview* dan menguji situs sebelum dibangun.
- d. Proses kompilasi yang menghasilkan folder situs akhir dalam format HTML, CSS, dan *Javascript* (Rinaldi dalam Newson, 2017).

.Beberapa *Static Site Generator (SSG) tools* yang dapat digunakan untuk membangun *website* statis dengan efisien dan optimal antara lain adalah Next.js, Hugo, dan *Astro* (Hanafi, 2022).

2.7 *Astro*

Astro adalah sebuah *meta-framework* awalnya, *Astro* dikembangkan dengan fokus pada *static site generation (SSG)* (Paakkanen, 2023). Selain itu, *Astro* bersifat *framework-agnostic*, yang berarti memungkinkan penggunaan berbagai *framework* front-end *JavaScript* populer, bahkan bisa menggabungkan beberapa *framework* dalam satu proyek, seperti yang ditunjukkan pada Gambar 2.3 (Paakkanen, 2023). *Astro* sering dipuji sebagai solusi yang baik, terutama untuk konten yang bersifat statis, karena memiliki dukungan yang sangat baik untuk menampilkan markdown dan format statis lainnya (Paakkanen, 2023).

Astro menggunakan file khusus dengan ekstensi *.astro*, dan setiap komponen dibagi menjadi dua bagian (Paakkanen, 2023). Bagian pertama disebut *frontmatter*, yang ditulis di antara dua garis pembatas kode (*code fence*) yang ditandai dengan ---. Di bagian ini, bisa mendeklarasikan variabel lokal atau mengimpor komponen lain (Paakkanen, 2023). Bagian kedua adalah *templating*, yaitu bagian tampilan yang merupakan perluasan dari *HTML* (superset of *HTML*) dan cara penggunaannya mirip dengan *JSX* (Paakkanen, 2023). Struktur ini ditunjukkan dalam Gambar 2.3 (Paakkanen, 2023).

```

1  ---
2  import ReactCounter from '../components/ReactCounter.jsx'
3  import SvelteCounter from '../components/Counter.svelte'
4  import QwikCounter from '../components/QwikCounter.tsx'
5  const title = 'Multiple counter islands'
6  ---
7
8  <div>
9    <h2>{title}</h2>
10   <ReactCounter client:load />
11   <SvelteCounter client:idle />
12   <QwikCounter />
13 </div>

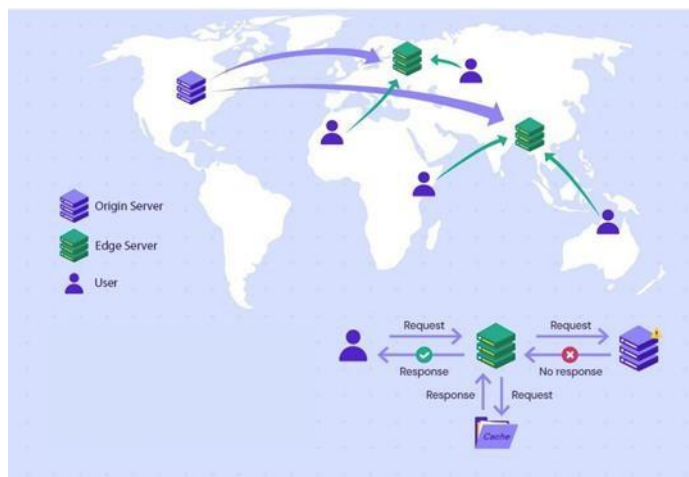
```

Gambar 2.3 Komponen *Astro* menampilkan 3 perbedaan *counter*

SSG *Tools* yang akan digunakan adalah *Framework Astro* dengan kerangka kerja web modern yang berfungsi sebagai *Static Site Generator (SSG)*, memungkinkan pembuatan situs web statis yang cepat dan efisien (Bratslavsky, 2025). Dengan pendekatan "*Server-First*", *Astro* merender komponen di sisi *server*, menghasilkan *HTML* yang ringan dan mengurangi beban *JavaScript* di sisi klien, sehingga meningkatkan performa situs (Bratslavsky, 2025). Selain itu, *Astro* memungkinkan pengelolaan konten dari berbagai sumber, seperti sistem file lokal, API eksternal, atau CMS, memberikan fleksibilitas dalam pengembangan (Bratslavsky, 2025).

2.8 Content Delivery Network

Content Delivery Network (CDN) adalah kumpulan kolaboratif dari elemen-elemen jaringan yang tersebar di seluruh internet, di mana konten di replikasi ke beberapa *server* web untuk melakukan pengiriman konten yang efektif dan transparan kepada pengguna akhir (Buyya et al., 2008). Kumpulan *server* global yang terletak di banyak pusat data dan tersebar di berbagai negara (Farid, 2024). Jaringan ini digunakan untuk mengirim konten dari *server* ke situs web (Farid, 2024) seperti pada Gambar 2.4.



Gambar 2.4 *Content Delivery Network*

Katakanlah pengunjung berada di India, sedangkan *server* situs web berada di AS (Farid, 2024). Karena jarak antara lokasi pengunjung dan *server*, pengiriman konten memakan waktu yang lama (Farid, 2024). Namun, dengan *server* CDN yang berlokasi dekat dengan India, pengiriman konten bisa lebih cepat (Farid, 2024). Fungsionalitas yang biasanya terdapat di CDN (Buyya et al., 2008):

- e. *Request redirection and content delivery services*, untuk mengarahkan permintaan ke *server cache CDN* yang sesuai dengan menggunakan mekanisme untuk menghindari kemacetan, sehingga dapat mengatasi lonjakan lalu lintas *server*.
 - f. *Content outsourcing and distribution services*, untuk mereplikasi dan/atau menyimpan *cache* konten dari *server* asal ke *server* web terdistribusi.
 - g. *Content negotiation services*, untuk memenuhi kebutuhan spesifik dari setiap pengguna individu.
- a.

2.9 Content Management System (CMS)

CMS merupakan suatu perangkat lunak yang berfungsi untuk mengelola berbagai jenis konten (Siambaton & Fakhriza, 2016). Konten yang dimaksud mencakup seluruh bentuk informasi digital, seperti file gambar, audio, video, teks, maupun jenis file digital lainnya (Siambaton & Fakhriza, 2016). CMS umumnya merupakan aplikasi berbasis web yang dibangun menggunakan bahasa pemrograman yang mendukung pengembangan web (Siambaton & Fakhriza, 2016).

Content Management System (CMS) adalah aplikasi berbasis web yang dirancang untuk Memudahkan pengguna maupun pengelola dalam mengelola konten (Iqbal, 2009). Dengan

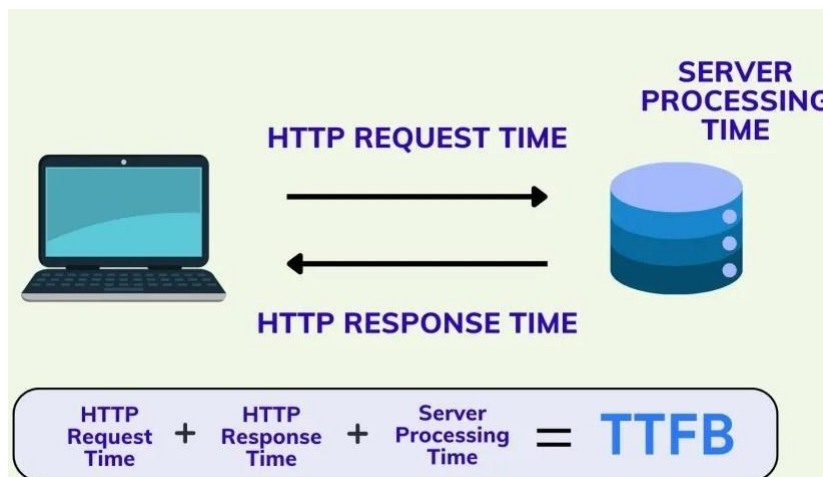
memisahkan antara isi dan desain, CMS menjaga konsistensi tampilan yang mempermudah penggunaan kembali berbagai informasi yang ada dalam *server* (Iqbal, 2009). CMS juga menawarkan berbagai fitur yang beragam, seperti pengelolaan tata letak situs untuk mengubah tampilan *website*, fitur pencarian, *editing* berita, *editing* foto, editing produk, dan fitur lainnya sesuai kebutuhan (Iqbal, 2009).

2.10 Headless CMS

Headless CMS mendaur ulang sebuah CMS dengan menyimpan dan mempublikasikan kontennya melalui konektivitas API dan teknologi *backend* (Sobri et al. dalam Tandra et al., 2025). *Headless* CMS memungkinkan penyimpanan dan penyebaran kontennya dikelola oleh aplikasi lain (Sobri et al. dalam Tandra et al., 2025). Tujuan utama dari *headless CMS* adalah memisahkan fungsi sistem dalam pembuatan, penyimpanan, dan pengelolaan konten dari fungsi sistem yang bertanggung jawab atas penyajian dan distribusi konten (Sobri et al. dalam Tandra et al., 2025). Dengan pemisahan peran ini, konten dan desain menjadi lebih fleksibel dan mudah disesuaikan (Sobri et al. dalam Tandra et al., 2025).

2.11 Time to First Byte (TTFB)

Time to first byte adalah waktu yang dibutuhkan sesaat setelah *user* melakukan *request url* (Hanggara, 2018). Prosesnya meliputi dibukanya *socket* koneksi internet, *DNS Resolving*, sampai web *server* memberikan *output* respons beberapa *bytes* data pertama (*first bytes*) (Hanggara, 2018). Secara sederhana, TTFB dapat didefinisikan sebagai waktu yang dihabiskan untuk menunggu respons pertama, yang mengukur latensi perjalanan bolak-balik ke *server* (Lau, 2023). Ini adalah waktu yang diperlukan untuk mengirimkan permintaan dan waktu berikutnya yang dibutuhkan *server* untuk mengirim respons kembali (Lau, 2023).



Gambar 2.5 Time to First Byte (TTFB)

Time to First Byte (TTFB) pada Gambar 2.5 mencakup akumulasi dari HTTP Request Time + HTTP Response Time + *Server Processing Time* (Lau, 2023). TTFB terdiri tiga komponen terukur (Lau, 2023):

- a. Waktu yang dibutuhkan untuk mengirimkan HTTP *request*.
- b. Waktu yang dibutuhkan untuk memproses *request* di *server*.
- c. Waktu yang dibutuhkan *server* untuk mengirimkan respons yaitu *byte* pertama ke *browser*.

Tools untuk mengukur TTFB *website* terbagi menjadi dua yaitu (Lau, 2023):

1. *Lab Tools* adalah alat mensimulasikan sesi penjelajahan pengunjung situs, memungkinkan untuk menguji fitur dan memecahkan masalah sebelum menerapkan perubahan ke produksi pada lingkungan terkontrol (Lau, 2023). Contoh *tools* seperti *Google Lighthouse* dan *Chrome Devtools*.
2. *Field Tools* adalah alat yang berjalan menggunakan informasi yang dikumpulkan dari pengunjung situs sebenarnya (Lau, 2023). Data ini mencerminkan beragam perangkat, kondisi jaringan, dan lokasi geografis pengguna (Lau, 2023). Contoh *tools* seperti *speed vitals* dan *Google Search Console*.

Time to First Byte (TTFB) sangat penting bagi *website* karena berbagai alasan (Lau, 2023). TTFB yang lebih tinggi menunjukkan proses pengambilan data yang lebih lambat dari *server*, yang secara langsung mempengaruhi kemampuan pengguna untuk melihat konten situs dalam waktu yang wajar (Lau, 2023). Dengan mengurangi TTFB akan mempengaruhi responsivitas situs terhadap pengguna seperti pengurangan waktu tunggu, peningkatan *user engagement*, dan mengurangi *bounce rates* (Lau, 2023).

2.12 SpeedVitals

SpeedVitals merupakan platform pengujian kinerja *website* yang berfokus pada pengukuran *Core Web Vitals* dan berbasis pada teknologi *Google Lighthouse*. Platform ini dilengkapi dengan berbagai fitur, seperti pengujian spesifik berdasarkan perangkat(*device-specific*), lokasi(*location-specific*), serta visualisasi pergeseran tata letak (*layout shifts*) (Kumawat K, 2021a).

SpeedVitals menyediakan TTFB *Testing tool* yang mampu mengukur *Time to First Byte* dari 25 lokasi di 18 negara hanya dalam satu kali pengujian. Fitur *Region-wise TTFb & Average Scores* mengelompokkan seluruh lokasi berdasarkan wilayah untuk memudahkan analisis dan menampilkan rata-rata TTFB setiap wilayah. Fitur *Interactive Map* memudahkan visualisasi lokasi di seluruh dunia, di mana pengguna dapat mengarahkan kursor ke titik tertentu untuk melihat nilai TTFB. Selain itu, fitur *Headers Section* memungkinkan pengguna memeriksa HTTP *headers* dari halaman web di 25 lokasi berbeda untuk memastikan kinerja yang konsisten di seluruh dunia (Kumawat K, 2021b). Setiap minggu, *SpeedVitals* juga menyediakan 30 kredit pengujian TTFB untuk mengetes URL yang akan berkurang sebanyak 3 kredit sekali pengujian (James H, 2024).

2.13 Tinjauan Pustaka

Penelitian yang dilakukan oleh Prayudi Utomo dan Falahah pada tahun 2020 dengan judul “*Building Serverless Website on Github Pages*” meneliti tentang cara membangun *website* tanpa *server* (*serverless*) menggunakan pendekatan *Jamstack* pada platform *Github Pages*. Pada penelitian tersebut mengimplementasikan Hugo sebagai *Static Site Generator* (SSG) dan *Github* sebagai *Content Delivery Network* (CDN). *Website* yang dibangun memperlihatkan pemanggilan API untuk menunjukkan kemudahan mengintegrasikan API dalam pembuatan *website* dengan menggunakan metode *Jamstack*. Studi kasus yang digunakan adalah membangun sebuah *website* berisi informasi tentang Kodein, sebuah startup yang menyediakan layanan IT. Terdapat tiga tahapan yang diterapkan dalam membangun *website* menggunakan *Jamstack* yaitu pembuatan konten, menerapkan *Jamstack* yaitu menggunakan *Static Site Generator* yang terdiri dari struktur *archetypes*, *data*, *layouts*, *static*, *themes* dan *config.toml*. Dalam pengembangan *website* ini memanggil API dari *disqus.com* yang digunakan untuk menyimpan dan menampilkan komentar dari pengunjung. Proses *deployment* dilakukan

menggunakan layanan *Github Pages* karena bersifat gratis, memiliki komunitas besar, serta terdapat *versioning* untuk melacak perubahan. Setelah *website* berhasil di *deploy* dilakukan *testing* dan pengukuran performa *website* tersebut. *Testing* dilakukan dengan memasukkan komentar langsung dan ditampilkan dalam *website*, kemudian pengujian performa menggunakan GTMetrix dengan parameter *Google PageSpeed* dan *YSlow*, dari hasil pengujian dihasilkan *website* dalam kondisi performa baik dengan skor *Page Speed* 97% dan *YSlow* 91%. Hasil ini membuktikan bahwa *website* tetap berjalan optimal meskipun terkoneksi dengan banyak API (Utomo & Falahah, 2020).

Pada tahun 2021, Florence Elfriede Sinthauli Silalahi bersama Mugi Prayitno, Windy Gambetta, Fahmi Amhar, M. Nur Qomari Adi Wijaya, dan Tia Rizka Nuzula Rachma melakukan penelitian berjudul “Identifikasi Pertumbuhan Data dan Penggunaan Application Performance Index (APDEX) Score dalam Penilaian Kinerja Aplikasi Penyaji Informasi Geospasial” Membahas tentang diperlukannya peningkatan sistem setelah dilakukan evaluasi kinerja menggunakan *Application Performance Index (APDEX) Score* dengan menggunakan Jmeter, *Google Analytic*, dan GT Metrix. Penelitian ini mengidentifikasi pertumbuhan data dan layanan aplikasi Informasi Geospasial (IG) terkait akses data, dukungan media penyimpanan, dan manajemen data. Selanjutnya, dilakukan *Load* dan *Stability Testing* dengan mensimulasikan berbagai jumlah *user* pada beberapa aplikasi (Ina-Geoportal, Palapa, IPASOET, SAKTI, WebGIS Batimetri, DEMNAS), yang bertujuan untuk mengetahui titik permasalahan sistem. Dari hasil pengujian diketahui angka pengguna aplikasi pemetaan cukup tinggi dimana hasil APDEX *score* menunjukkan *range* 0.156 hingga 0.945. Namun, berdasarkan pengukuran menggunakan GTMetrix aplikasi belum menjalankan performa terbaik melihat dari parameter *Page Speed* dan *YSlow*, dimana *Page Speed* dikategorikan sebagai *grade* E dan F kemudian *YSlow* dikategorikan sebagai *grade* D, E, dan F. Penulis merekomendasikan adanya perbaikan dari sisi scripts, utilisasi *server*, serta *bandwidth* yang tersedia. Selanjutnya, diperlukan adanya pemanfaatan CDN atau *proxy* untuk *cache script* js, css, dan peta dasar (*images*), serta optimalisasi sesuai rekomendasi GTMetrix (Silalahi et al., 2021).

Penelitian serupa dilakukan oleh Widhi Lestari dan Aris Susanto berjudul “Analisis Performa *Website* ISI Surakarta dan Universitas Diponegoro Menggunakan Automated Software Testing GTmetrix” membahas tentang pentingnya performa *website* pada perguruan tinggi untuk kebutuhan informasi dan komunikasi. Pengukuran performa menggunakan GTMatrix dengan

melihat parameter *Page Speed* dan YSlow. Penelitian ini membandingkan performa *website* ISI Surakarta dan Universitas Diponegoro serta memberikan saran perbaikan kepada pengelola *website* terkait. Dari hasil pengukuran ISI Surakarta diperoleh rata-rata *PageSpeed* grade 54% atau di *grade* E, dan rata-rata YSlow *grade* 52% atau di *grade* E, sedangkan *website* Universitas Diponegoro tidak menampilkan hasil analisis karena *GTMetrix* bisa dikatakan bahwa *tools* tersebut tidak bisa mengambil data komponen karena *website* tidak memberikan akses untuk dilakukan *testing* (Lestari & Susanto, 2017).

BAB III

PELAKSANAAN MAGANG

3.1 Manajemen Proyek

Tahapan manajemen proyek yang digunakan selama proses pengerjaan Front-end *Website* Radyakartiyasa menggunakan metode *Jamstack*. Berikut merupakan tahapan yang diterapkan dalam pengerjaan Front-end *website* Radyakartiyasa:

3.1.1 Plan Human Centered Process

Tahapan pertama yang dilakukan adalah diskusi dengan semua pihak yang terlibat dalam pengerjaan proyek *website* Radyakartiyasa untuk mendapatkan kesepakatan bahwa proses pengembangan Front-end dilakukan dengan berfokus pada penggunaan Framework. *Website* ini dimulai dengan pembuatan *mockup website* yang telah disetujui. Proyek *Website* Radyakartiyasa akan melibatkan Framework *Astro*. Terdapat beberapa peran yang ditugaskan kepada sejumlah pihak dalam proyek *website* Radyakartiyasa di antaranya sebagai berikut:

- a. *Supervisor*, bertanggung jawab dalam memimpin pelaksanaan proyek dengan memberikan arahan kepada tim pengembang, menentukan ruang lingkup sistem yang akan dibuat, serta melakukan penilaian dan evaluasi atas hasil pengembangan sistem tersebut.
- b. *Scrum Master*, bertanggung jawab untuk memastikan pelaksanaan metode *scrum* pada pelaksanaan proyek dengan memfasilitasi kegiatan *scrum* harian dan *sprint review*. Selain itu, Scrum Master juga memantau penggunaan *project management software* yang digunakan oleh tim pengembang guna mengawasi proses pelaksanaan proyek.
- c. *Front end developer* Bertugas untuk mengimplementasikan desain dari Figma ke dalam *Astro* serta menyusun struktur laman *website* untuk pengembangan SSG.
- d. *Backend developer* bertanggung jawab dalam membangun sisi *backend* dengan memanfaatkan Headless CMS yang menyediakan akses melalui API. API ini kemudian digunakan oleh bagian *frontend* yang dikembangkan menggunakan *framework Astro* dalam proses pembuatan situs web berbasis *Static Site Generation* (SSG).

3.1.2 Schedule Weekly Meeting

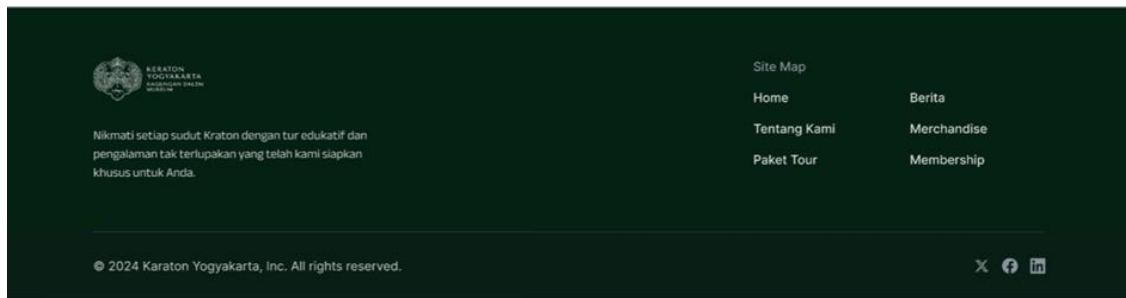
Tahapan kedua dalam proses pengembangan proyek difokuskan pada aktivitas pemantauan dan evaluasi kemajuan kerja tim. Pada tahap ini, ditetapkan sejumlah *milestone* sebagai indikator pencapaian utama yang harus dicapai dalam kurun waktu tertentu sesuai dengan rencana kerja yang telah disusun. Penentuan *milestone* ini bertujuan untuk memberikan tolak ukur yang jelas terhadap progres pengembangan proyek serta membantu dalam mengidentifikasi potensi rintangan atau keterlambatan secara dini. Ruang lingkup pekerjaan peserta magang di Lumintu Logic Indonesia difokuskan pada penyelesaian deliverables yang tercantum dalam Milestone I, dengan memastikan hasilnya selaras dengan desain yang telah ditetapkan di Figma. Untuk memastikan setiap *milestone* tercapai sesuai jadwal, dilakukan kegiatan *weekly meeting* secara rutin setiap hari Selasa. Pertemuan ini berfungsi sebagai forum koordinasi antaranggota tim untuk pembagian tugas secara individual kepada setiap anggota tim sesuai dengan keahlian masing-masing, menyampaikan progres terkini, mendiskusikan permasalahan yang dihadapi, serta merumuskan langkah tindak lanjut.

3.1.3 UI/UX Design Radyakartiyasa

Desainer UI/UX diberikan waktu khusus untuk mengembangkan desain situs *website* Radyakartiyasa menggunakan platform desain digital. Fokus utama dalam proses pengembangan ini adalah pembuatan dan penyempurnaan berbagai bagian penting dari situs web, termasuk komponen-komponen umum(*common components*) seperti navbar(Gambar 3.1) untuk navigasi web dan *footer*(Gambar 3.2)pada bagian bawah halaman yang berisi konten serta sosial media Radyakartiyasa.

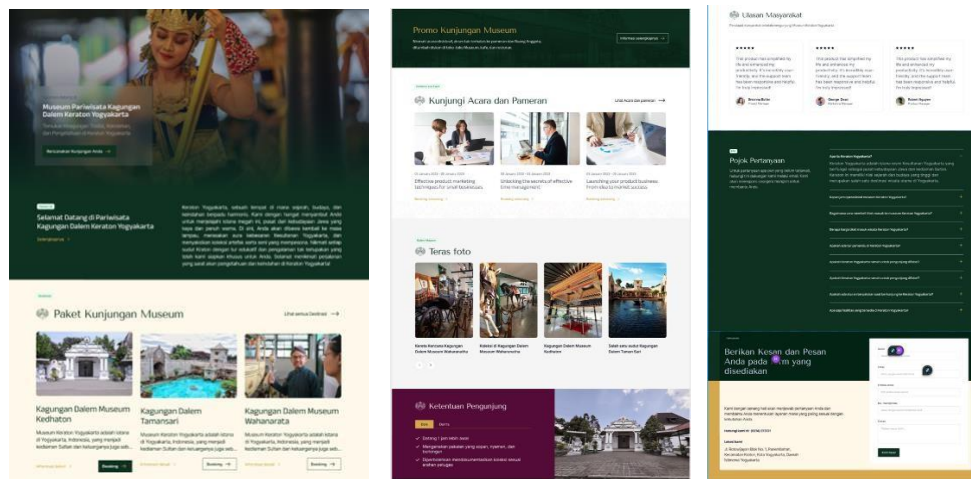


Gambar 3.1 Navbar



Gambar 3.2 Footer

Selain itu, untuk halaman beranda/*homepage* (Gambar 3.3), desainer bertanggung jawab dalam merancang dan mengembangkan sejumlah komponen yang meliputi Hero yang terdiri dari banner dan kartu informasi, bagian About Us (Gambar 3.4) yang memberikan gambaran mengenai profil atau tujuan situs, *destination slider* yang menampilkan pilihan destinasi secara interaktif, *event slider* untuk mempromosikan acara atau kegiatan terkini, bagian promo yang menampilkan penawaran khusus, galeri visual pada Home Gallery, Testimonial sebagai ruang untuk menampilkan ulasan pengguna atau pelanggan, FAQ (Frequently Asked Questions) untuk menjawab pertanyaan umum pengunjung, serta Contact Us sebagai sarana komunikasi antara pengunjung dan pengelola situs.



Gambar 3.3 Desain konten

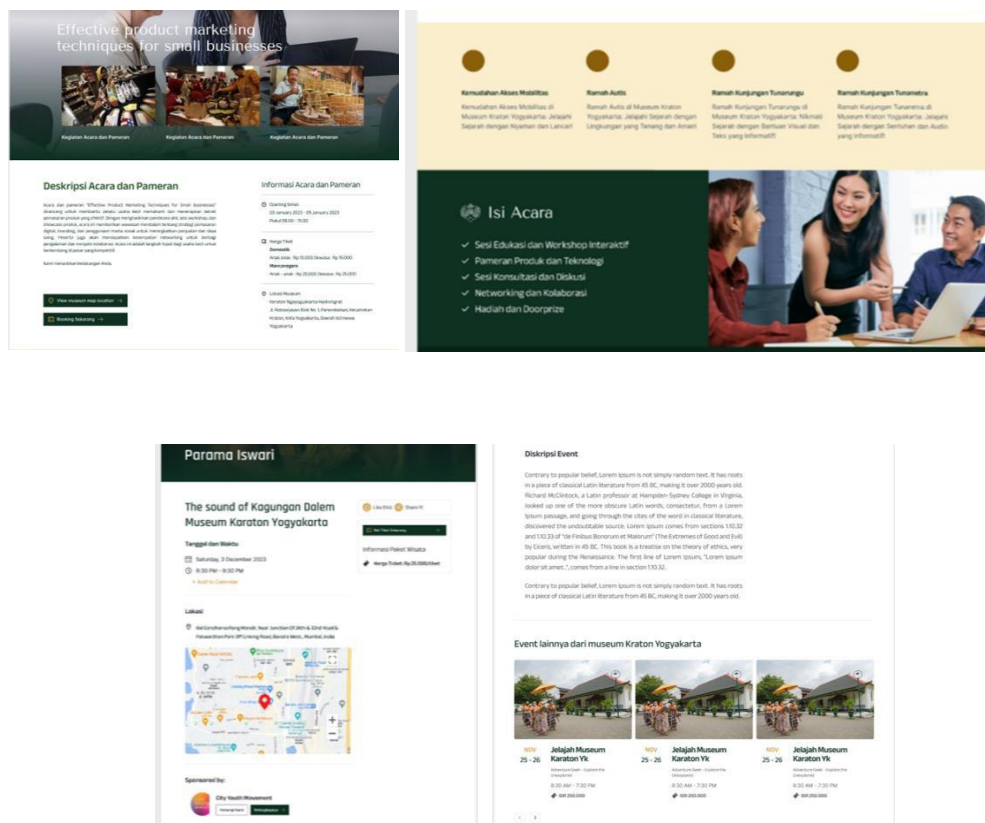


Gambar 3.4 Konten tentang Kagungan

Selanjutnya, pada halaman Destinasi (Gambar 3.5) dan *Event* (Gambar 3.6), layout yang dikembangkan mencakup komponen-komponen seperti *destination slider*, *event slider*, dan *Hero section* yang menjadi bagian utama tampilan visual halaman tersebut. Komponen-komponen spesifik untuk halaman Destinasi dan *Event* dirancang secara detail meliputi Hero sebagai elemen pembuka, Deskripsi yang menjelaskan informasi utama, informasi *sidebar* yang menyediakan data tambahan seperti jam buka dan lokasinya, bagian *Accessibilities* yang menginformasikan fasilitas dan kemudahan akses, serta bagian *Do's and Don'ts* yang memberikan panduan atau aturan terkait destinasi atau *event* yang bersangkutan.



Gambar 3.5 Konten destinasi



Gambar 3.6 Konten event

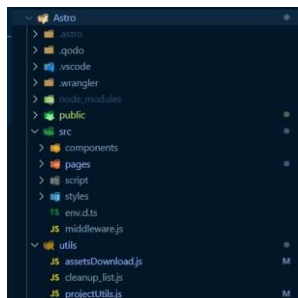
Perlu dicatat bahwa desain ini bersifat dinamis dan akan mengalami perubahan serta penyesuaian selama proses pengembangan berlangsung, agar dapat memenuhi kebutuhan teknis dan fungsional yang diminta yang muncul seiring dengan tahap implementasi kode dan integrasi sistem. Dengan demikian, desain yang telah dibuat merupakan fondasi awal yang fleksibel untuk diadaptasi sesuai dengan perkembangan proyek secara keseluruhan.

3.2 Proses dan Hasil Pelaksanaan Proyek

Berikut merupakan hasil dari implementasi *Astro* pada *website* Radyakartiyasa yang dijelaskan per tahap sesuai dengan Alur pengerjaan.

3.2.1 Fetching Data from CMS

Membuat file `projectUtils.js` di dalam folder `utils` yang berfungsi untuk mengambil data dari CMS melalui API untuk *website* dengan keberadaan filenya pada Gambar 3.7.



Gambar 3.7 Struktur folder

a. Gambar 3.8 menunjukkan pemanggilan API ke CMS dilakukan dengan menggunakan host yang berisi URL *endpoint*. Untuk mengakses data di CMS tersebut, diperlukan headers yang terdiri dari beberapa parameter penting. Header Authorization berisi token akses dalam format Bearer yang digunakan untuk otentikasi. Selain itu terdapat *client* id dan *client* secret, yaitu kredensial yang diambil dari layanan Cloudflare untuk membatasi dan mengamankan akses ke CMS melalui proteksi tambahan. Kombinasi ini memastikan hanya klien yang sah yang dapat mengambil data dari CMS.

```
// directus' backend parameters
export const host = "https://directus_lap_skripsi.zakyabiyu.xyz";
export const headers = {
  Authorization: "Bearer statictokenFromAstroSSG",
  "CF-Access-Client-Id": "cloudflareServiceTokenClientID.access",
  "CF-Access-Client-Secret": "CloudflareServiceTokenClientSecret"
};
```

Gambar 3.8 Host dan headers

b. *Session token* pada Gambar 3.9 digunakan untuk mengidentifikasi dan menjaga sesi pengguna saat berkomunikasi dengan CMS agar tetap aman dan terkontrol.

```
export const session_token_name = "radya_triplec_session_token";
```

Gambar 3.9 Session token

c. Gambar 3.10 menunjukkan variabel *cms_images* adalah folder path yang digunakan untuk menyimpan gambar yang diambil dari CMS. Kemudian *img_arr* adalah daftar tipe file gambar yang bisa dioptimasi atau dikonversi ke format .webp, yaitu tipe gambar JPEG, PNG, dan WebP.

```
// astro paths
export const cms_images = "assets/cms_images";

//list of optimizable image formats, gambar di backend dengan format tertentu akan bisa dioptimasi
export const img_arr = [
  "image/jpeg",
  "image/png",
  "image/webp"
];
```

Gambar 3.10 Cms images

d. Gambar 3.11 menunjukkan variabel `getLang` digunakan untuk menerima parameter `lang` untuk multilingual konten seperti 'en' atau 'id'. String *query* yang akan digunakan untuk API CMS supaya mengambil data dengan filter bahasa tertentu. Artinya, API akan menyesuaikan item yang memiliki terjemahan dengan kode bahasa sama dengan *lang*.

```
// api to get lang
export const getLang = (lang) => {
  return `deep[translations][_filter][languages_code][_eq]=${lang}`;
}
```

Gambar 3.11 `getLang`

e. Gambar 3.12 menunjukkan variabel `fetchLang` menggunakan *async function* untuk menjalankan operasi asinkron yang mudah dibaca seperti mengambil daftar bahasa dari *collections languages* menggunakan metode *fetch* dan menyertakan *headers* untuk autentikasi. Respon yang diterima, data hasil `res.data` diproses dengan `map()`. Setiap item *languages* diubah menjadi objek yang terdiri dari:

- `Params.lang` yaitu kode bahasa 'en' atau 'id' untuk routing atau parameter URL.
- `Props` berisi objek dengan *name* (nama bahasa, misal "English") dan *direction* (arah teks, misal "ltr" untuk kiri ke kanan atau "rtl" untuk kanan ke kiri).

Fungsi mengembalikan array objek yang siap digunakan untuk keperluan seperti *dynamic route* atau pengaturan tampilan berdasarkan bahasa.

```
// memanggil collection languages, async function mengatur kapan mau menunggu dan kapan mau jalan terus
export const fetchLang = async function () {
  const data_lang = await fetch(`${host}/items/languages`, {
    headers: headers
  }).then(res => res.json()).then(res => res.data);

  return data_lang.map(function (item) {
    return {
      params: {
        // Mengambil field code dari collection languages untuk routing atau parameter URL
        lang: item.code
      },
      props: {
        // Mengambil field name dan direction dari collection languages untuk props
        name: item.name,
        direction: item.direction,
      }
    };
  });
}
```

Gambar 3.12 `FetchLang`

f. Gambar 3.13 menunjukkan variabel `fetchIdSlug` memiliki parameter fungsi yang terdiri dari:

- *endpoint* untuk URL *endpoint* API CMS untuk mengambil data item (contoh : `/items/events?limit=-1`).

- `idPath`: nama properti di objek hasil API yang mewakili ID item (misal “id” atau “`event_id`”).
- `titlePath`: nama properti di objek hasil API yang berisi judul/nama item yang akan diubah menjadi *slug* (misal “title” atau “name”). `pathArr` digunakan untuk array kosong untuk menampung hasil akhir berupa *object routing*. Kode `await Promise.all(data_lang.map(async function (item) ...` melakukan iterasi `map()` terhadap semua bahasa, dan `Promise.all` memastikan semua permintaan asinkronus dijalankan paralel dan menunggu semuanya selesai. Kode `events` mengambil data dari *endpoint* yang disediakan, ditambah filter berdasarkan bahasa lang dari fungsi `getLang(item.code)`. Kode `await Promise.all(events.map(function(item2) ...` untuk setiap item dalam data yang didapat (*events*) di *push* ke array `pathArr` dengan struktur `{lang, id, slug}`, fungsi *slug* akan mengubah judul menjadi URL-friendly format, misalnya: “Berita Baru” menjadi *berita-baru*. Return `pathArr` akan mengembalikan array yang sudah berisi *object* dengan parameter routing multibahasa terstruktur.

```
export const fetchIdSlug = async function (endpoint, idPath, titlePath) {
  let pathArr = [];

  const data_lang = await fetch(`${host}/items/languages`, {
    headers: headers
  }).then(res => res.json()).then(res => res.data);

  await Promise.all(data_lang.map(async function (item) {
    const events = await fetch(`${host}${endpoint}${getLang(item.code)}`, {
      headers: headers
    }).then(res => res.json()).then(res => res.data);

    await Promise.all(events.map(function (item2) {
      pathArr.push({
        params: {
          lang: item.code,
          id: item2.idPath,
          slug: slugify(item2.titlePath)
        }
      });
    }));
  }));

  return pathArr;
}
```

Gambar 3.13 FetchIdSlug

g. Gambar 3.14 menunjukkan variabel `fetch_data` menerima tiga parameter yang terdiri dari:

- *Endpoint* untuk *collection* dari CMS yang akan diakses.
- *Lang* bersifat opsional yang berisi nilai kode bahasa seperti ‘id’ atau ‘en’ yang jika diisi akan menambahkan parameter filter bahasa pada query URL.
- *Cust_fil* bersifat opsional filter tambahan dalam format *string query* yang dapat dikustomisasi untuk kebutuhan spesifik, seperti pencarian berdasarkan kategori,

status, atau tanggal.

Kode `fil_lang` digunakan untuk jika parameter `lang` diisi, maka akan dibentuk string query filter multibahasa sesuai struktur *nested* dari CMS menggunakan *query* `deep[translations]`, jika tidak diisi maka `fil_lang` bernilai string kosong. Kode `fil_cust` yang nilainya akan langsung digunakan sebagai bagian dari query URL, jika tidak nilainya dikosongkan. Kode `url` digunakan untuk pembentukan URL lengkap untuk permintaan data dibentuk dengan menggabungkan *host*, nama koleksi *endpoint*, filter bahasa, dan filter kustom. Kode `res` akan melakukan *fetch* data sesuai dengan kode `url`, diikuti penguraian data dalam format json dan ekstraksi field data yang berisi isi koleksi. Di *return res* yang menghasilkan array atau objek.

```
// helper fetch data
export const fetch_data = async function (endpoint, lang = false, cust_fil = false) {
  // dibentuk string query filter multibahasa sesuai struktur nested dari Directus
  const fil_lang = (lang ? `deep[translations][_filter][languages_code][_eq]=${lang}` : '');
  // nilainya akan langsung digunakan sebagai bagian dari query URL
  const fil_cust = (cust_fil ? cust_fil : '');
  // URL lengkap untuk permintaan data dibentuk dengan menggabungkan host, nama koleksi endpoint, serta filter bahasa dan filter kustom
  const url = `${host}/${items}/${endpoint}${fil_lang}${fil_cust}`;

  // Fetch data dari URL yang telah dibentuk
  const res = await fetch(url, {
    headers: headers
  }).then(res => res.json()).then(res => res.data);

  return res;
}
```

Gambar 3.14 Fetch_data

h. Gambar 3.15 menunjukkan Variabel *slugify* menerima satu parameter yaitu *str* sebuah string yang akan diproses menjadi *slug*, jika parameter yang diberikan bukan string, fungsi akan langsung melempar error dengan pesan “Input harus berupa string” untuk menjaga validitas input. Fungsi melakukan serangkaian transformasi berturut-turut pada string input *str* dengan `toLowerCase()` mengubah semua huruf menjadi huruf kecil agar konsisten, `trim()` untuk menghapus spasi kosong di awal dan akhir string agar *slug* tidak dimulai atau diakhiri dengan spasi. Kode `.replace(/[^\w\s-]/g, '')` untuk menghapus semua karakter yang bukan huruf, angka, spasi, atau tanda hubung. Kode `.replace(/[s_-]+/g, '-')` mengganti satu atau lebih spasi, garis bawah, atau tanda hubung berturut-turut menjadi satu tanda hubung. Kode `.replace(/^-\+|-\+$/g, '')` Menghapus tanda hubung yang berada di awal atau di akhir string agar *slug* tidak diawali atau diakhiri tanda hubung. Hasilnya `https://example.com/artikel/wisata-alam-krinjing`.

```
// slugify
export const slugify = (str) => {
  // Jika parameter yang diberikan bukan string, fungsi akan langsung melempar error dengan pesan "Input harus berupa string" untuk menjaga validitas input
  if (typeof str !== 'string') {
    throw new Error('Input harus berupa string');
  }

  return str
    .toLowerCase() // Mengubah semua huruf menjadi huruf kecil agar konsisten dan menghindari masalah case sensitivity dalam URL
    .trim() // Menghapus spasi kosong di awal dan akhir string agar slug tidak dimulai atau diakhiri dengan spasi
    .replace(/[^\w\s-]/g, '') // Menghapus semua karakter yang bukan huruf, angka, spasi, atau tanda hubung
    .replace(/[s_-]+/g, '-') // Mengganti satu atau lebih spasi, garis bawah, atau tanda hubung berturut-turut menjadi satu tanda hubung
    .replace(/^-\+|-\+$/g, '') // Menghapus tanda hubung yang berada di awal atau akhir string agar slug tidak diawali atau diakhiri tanda hubung
    .replace(/-+/g, '-');
}
```


saja. Hasilnya `strip_tags("<p>Halo Dunia</p>") => "Halo Dunia"`

```
// strip tags, string yang berisi teks dengan kemungkinan terdapat tag HTML atau XML.
export const strip_tags = (str) => {
  // metode replace() pada string dengan menggunakan regular expression (regex) untuk menemukan dan menghapus seluruh tag markup.
  return str.replace(/<\/?[>]+(>|<|<\/)/g, '');
}
```

Gambar 3.17 Strip_tags

k. Gambar 3.18 menunjukkan fungsi `file_handle` untuk menerima parameter `asset` yang berisi informasi tentang sebuah file seperti gambar dari CMS. Jika `asset` tidak ada atau `null`, fungsi ini akan mengembalikan `path` gambar `default` sebagai `fallback`.

Jika `asset` ada maka dicek apakah tipe file (`asset.type`) termasuk dalam daftar `img_arr`. Jika ya, maka URL gambar yang dikembalikan adalah `path` ke folder `cms_images` dengan nama file berdasarkan `asset.id` dan ekstensi `webp`. Ini berarti gambar akan menggunakan versi `.webp` yang sudah di optimasi. Jika tipe file tidak ada di `img_arr` seperti dokumen atau video, fungsi akan mengembalikan URL langsung ke file asli dengan nama berdasarkan `asset.filename_disk` tanpa diubah formatnya.

```
export const file_handle = (asset) => {
  if (!asset) {
    return "/assets/img/no-image.webp";
  } else {
    let src;
    if (img_arr.includes(asset.type)) {
      src = `/${cms_images}/${asset.id}.webp`;
    } else {
      src = `/${cms_images}/${asset.filename_disk}`;
    }

    return src;
  }
}
```

Gambar 3.18 File_handle

l. Variabel `day_format` bertujuan untuk mengonversi indeks hari (berupa angka 0-6) menjadi nama hari sesuai bahasa dan format yang ditentukan. Parameter masukan yang terdiri dari:

- Str adalah angka yang mewakili indeks hari dalam *JavaScript*, seperti 0 adalah minggu/*sunday* hingga 6 adalah sabtu/*saturday*.
- Lang sebagai kode lokal bahasa seperti “id-ID” untuk bahasa Indonesia.
- Format yang berisi opsi *long* untuk menampilkan bentuk tampilan nama hari, dengan dua kemungkinan “long” dengan nama hari lengkap, seperti senin atau *monday* lalu “short” untuk nama singkat tiga huruf seperti sen atau mon.

Objek bernama `day_trans` yang menyimpan daftar nama hari untuk dua kode lokal bahasa dengan format *long* dan *short* masing-masing. Nilai nama hari yang sesuai diambil dengan cara mengakses objek `day_trans` berdasarkan tiga indeks `return day_trans`.

m. Gambar 3.19 menunjukkan variabel `get_interface` menerima dua parameter masukkan sebagai berikut:

- Interfaces adalah tabel berisi array yang terdiri dari objek-objek, di mana setiap objek memiliki properti `key`.
- Key adalah properti `key` yang dimiliki *collection interfaces*. Parameter ini dapat berupa kata kunci lengkap atau sebagian dari value dari nama interface yang dicari.
- Menggunakan metode `filter` dengan kondisi `item.key.includes(key)`, artinya properti `key` dari objek `item` harus mengandung substring `key` yang diberikan sebagai parameter fungsi. Metode `includes(key)` mengembalikan nilai boolean `true` jika substring ditemukan, dan `false` jika tidak.

```
export const get_interface = (interfaces, key) => {
  return interfaces.filter((item) => item.key.includes(key));
}
```

Gambar 3.19 Get_interface

3.2.2 Download Assets From CMS

Import got digunakan untuk *merequest* HTTP protokol termasuk mendownload konten dari CMS, dengan *got* bisa mengakses resource dari web atau API. *Import fs* yang menyediakan fungsi untuk berinteraksi dengan sistem berkas komputer seperti membaca file, menulis file, membuat direktori, dll. *Import path* yang digunakan untuk memanipulasi dan menangani jalur file atau direktori dengan cara yang aman dan kompatibel antar sistem operasi. *Import* yang terdiri dari `host`, `headers`, `cms_images`, `img_arr` dari file bernama *projectUtils.js* yang berada di folder sama. Variabel `directory` yang berisi nilainya adalah string *path* folder yang dibentuk secara dinamis menggunakan *template string*, *path* ini mengacu ke folder lokal `public` lalu dilanjutkan dengan nilai dari variabel `cms_images`.

Variabel *usedAssets* menggunakan fungsi asinkron jadi bisa menggunakan `await` untuk menunggu proses seperti `fetch/http` selesai atau yang lainnya.

Melakukan request HTTP GET ke URL untuk mengambil daftar `fields/kolom` dari CMS. Request disertai *header* autentikasi. Respons diproses jadi JSON dan ambil properti `.data` yang

berisi array *field* metadata. Simpan hasilnya di variabel *ass* dalam bentuk array (Gambar 3.20).

```
const ass = await fetch(`${host}/fields`, {
  headers: headers
}).then((res) => res.json()).then((res) => res.data);
```

Gambar 3.20 Variable *ass*

Siapkan array kosong untuk menyimpan data file (Gambar 3.21). Iterasi *ass* dengan *map*, lakukan proses asinkron di setiap item. *Promise.all* agar semua proses berjalan paralel dan menunggu semua selesai. Filter *field* berdasarkan tipe *interface* yang berkaitan dengan file.

```
let assets = [];
await Promise.all(ass.map(async (item) => {
  if (item.meta?.interface === "file" || item.meta?.interface === "file-image" || item.meta?.interface === "files") {
```

Gambar 3.21 Kode *promise*

Jika *interface* memiliki nilai *file* atau *file-image*, ambil semua kolom dari *field* tersebut. Kalau *interfacenya* *files*, ambil kolom dari *nested object cms_files_id* di dalam array. Gambar 3.22 menunjukkan melakukan *Request* ke *endpoint* */items/\${item.collection}* dengan query parameter *fields* sesuai *field selector* yang dibuat. Ambil data JSON, kemudian data yang berisi array objek data dari *collection* tersebut.

```
const files = await fetch(`${host}/items/${item.collection}?fields=${field_selectors}`, {
  headers: headers
}).then((res) => res.json()).then((res) => res.data);
```

Gambar 3.22 Variable *files*

Gambar 3.23 menunjukkan *files* adalah array dengan loop tiap file dengan *map()*. Jika *interface* adalah *file* atau *file-image* maka cek *file[item.field]* apakah bukan *null* dan jika ada tambahkan ke *assets*. Jika *interface multiple files*, loop semua file dalam array *file[item.field]* dan tambahkan properti *cms_files_id* ke *assets*. Jika *files* bukan array (kosong atau objek tunggal), lakukan pengecekan yang mirip dengan di atas. Fungsi *usedAssets()* tidak digunakan untuk mengunduh file secara langsung ke folder *cms_images*, melainkan untuk mengumpulkan daftar asset dari backend, yang kemudian diproses oleh bagian kode berikutnya untuk diunduh dan disimpan ke folder tersebut.

```

if (files.length > 0) {
  files.map((file) => {
    if (item.meta.interface === "file" || item.meta.interface === "file-image") {
      if (file[item.field] !== null) { assets.push(file[item.field]); }
    } else {
      file[item.field].map((many_files) => {
        assets.push(many_files['directus_files_id']);
      });
    }
  });
} else {
  if (item.meta.interface === "file" || item.meta.interface === "file-image") {
    if (files[item.field] !== null) { assets.push(files[item.field]); }
  } else {
    files[item.field].map((many_files) => {
      assets.push(many_files['directus_files_id']);
    });
  }
}
});
return assets;

```

Gambar 3.23 Mengecek format file

Gambar 3.24 menunjukkan loading animation memiliki parameter tersedia:

- Params *text* untuk teks string yang ingin ditampilkan di samping animasi.
- Params *chars* yaitu array karakter khusus yang akan dipakai untuk animasi loading.

Berisi 10 karakter unicode untuk membentuk animasi bergerak ala spinner.

- Param *delay* yaitu interval waktu dalam milidetik antara pergantian karakter animasi.

a.

Variabel *x* diinisialisasi dengan 0. Kode `process.stdout.write` adalah cara menulis teks langsung ke terminal/console tanpa menambahkan baris baru. Untuk “\r” adalah *carriage return*, artinya kursor dibawa ke awal baris tanpa membuat baris baru, sehingga teks akan menimpa baris sebelumnya di terminal. Penggunaan `chars[x++]` mengambil karakter animasi berdasarkan indeks *x*, lalu *x* ditambah 1 untuk iterasi berikutnya. Nilai `x % chars.length` membuat indeks *x* kembali ke 0 setelah mencapai akhir *array chars*, sehingga animasi berputar terus menerus. Kemudian *text* adalah string statis yang selalu muncul setelah karakter animasi. Fungsi *setInterval* mengembalikan sebuah ID interval yang bisa dipakai untuk menghentikan animasi nanti dengan *clearInterval(id)*, jadi *loadingAnimation* mengembalikan ID interval ini untuk dikontrol di luar. Memanggil fungsi *loadingAnimation* dengan teks “*downloading assets to public folder*”. Dijalankan di terminal sebagai bukti ada proses pengunduhan asset dari *backend*.

(*finalsize*) dengan `fs.statSync`. Periksa apakah tipe file `item.type` sudah ada di `type_sets`, jika sudah maka tambah jumlah file dari total ukuran asli (*filesize* dari backend), dan total ukuran hasil optimasi (*finalsize*). Jika belum ada, inisialisasi objek tipe file baru dengan nilai-nilai awal yang akan membuat entri baru di objek `type_sets` untuk tipe file yang belum pernah dicatat sebelumnya. Hitung rasio optimasi (ukuran akhir dibagi ukuran asli), dibulatkan tiga digit desimal.

Jika jumlah file selesai (*done*) sama dengan total file di *content*, berarti unduhan selesai semua. Hentikan animasi *loading* dengan `clearInterval(loader)`. Cetak ke *console* objek statistik `type_sets` hasil optimasi. Jika terjadi *error* saat proses unduh *streaming*, lempar *error* supaya program bisa tahu ada kegagalan.

3.2.3 the Assets In Public Folder

Folder *public/* digunakan untuk file dan aset dalam proyek yang tidak memerlukan pemrosesan selama proses *build* di *Astro*. File-file dalam folder ini akan disalin ke dalam folder *build* tanpa diubah, dan situs radyakartiyasa akan dibangun berdasarkan file tersebut. Perilaku ini membuat folder *public/* ideal untuk aset umum yang tidak memerlukan pemrosesan yang di dalam proyek ini terdapat file *js* untuk interaktivitas dan folder *cms_images*. Tetapi harus disadari bahwa file-file tersebut tidak akan dioptimalkan dalam *build* akhir situs.

Folder *public* berisi folder *assets* yang menyimpan beberapa folder yang akan digunakan untuk proyek situs radyakartiyasa:

- a. Folder *cms_images* digunakan untuk menyimpan konten CMS yang diunduh melalui file *assetsDownload.js*. Konten tersebut mencakup gambar yang dioptimasi dengan ekstensi *.webp*, file video dengan ekstensi *.mp4*, dan ikon situs dengan ekstensi *.svg*. Folder *js* untuk interaktivitas yang berisi dua file:
 1. File *main.js* menangani beberapa interaksi di halaman web menggunakan *Jquery* dan beberapa *plugin*, *Select2* dan *Magnific Popup*, serta menyediakan fungsionalitas untuk navigasi dan notifikasi.
 - a) Inisialisasi *Select2* pada elemen dengan kelas *.select-custome* untuk meningkatkan tampilan dropdown.
 - b) Menampilkan notifikasi dengan ikon yang sesuai menggunakan fungsi *alertNotification*.

- c) Mengelola navigasi *header*, yaitu membuka dan menutup menu saat tombol diklik pada responsif desain.
- d) Menampilkan gambar dalam *lightbox* menggunakan *plugin Magnific Popup* untuk galeri gambar interaktif.

2. File *plugins.js* akan mengimplementasi *smooth scroll* di halaman web yang memungkinkan user menggulirkan halaman dengan cara yang lebih halus dan terkontrol.

- a) Pengaturan *scroll*: mengatur kecepatan, akselerasi, dan waktu animasi menggunakan parameter seperti *frameRate*, *animationTime*, dan *accelerationMax*.
- b) *Event listener*: mengatur *scroll* dengan mouse, tombol arah keyboard, dan *resize* layar.
- c) Performa optimal: menggunakan *requestAnimationFrame* untuk mengelola *scroll* secara halus.
- d) Kompatibilitas browser: mendeteksi berbagai platform dan browser (*safari*, *chrome*, dan *windows*) dan menyesuaikan fungsionalitas untuk kompatibilitas maksimal.

b. Folder vendors berisi berbagai css dan js *plugin* untuk menambah interaktivitas dan *styling* situs radyakartiyasa :

1. Folder *bootstrap* terkait *framework* bootstrap yang berisi folder css dan js yang digunakan untuk mendesain dan mengelola tampilan dan interaktivitas web.
2. Folder *feather* berisi ikon-ikon yang digunakan pada situs radyakartiyasa. Di dalamnya terdapat dua subfolder, yaitu css yang menyimpan file CSS untuk mengatur tampilan ikon, dan fonts yang berisi file font dalam berbagai format seperti .eot, .woff, .ttf, dan svg untuk mendukung tampilan ikon di berbagai perangkat dan browser. Dengan menggunakan kode *@font-face* dalam CSS, folder ini memastikan ikon dapat ditampilkan dengan benar di situs web.
3. Folder *magnific-popup* yang digunakan untuk menampilkan konten media (gambar, video, atau *iframe*) dalam *lightbox (popup)* yang interaktif. Di dalam terdapat dua file yaitu *magnific-popup.css* untuk mengatur tampilan dan desain elemen *popup*,

termasuk animasi, dan layout konten. Sementara itu, *magnific-popup.min.js* yang menangani interaktivitas *popup*, seperti pengaturan navigasi antar gambar, penutupan *popup*, preloading konten, dan pengaturan khusus untuk berbagai jenis media seperti gambar, video, dan iframe.

4. Folder *swiper* yang digunakan untuk membuat *slider* interaktif pada section tertentu seperti section destinasi dan *event* di *homepage*. Di dalam folder ini terdapat dua file css dan satu file js yang mendukung pembuatan *slider* dengan transisi yang mulus dan interaktif.
5. File *smooth-scroll.min.js* mengimplementasikan *smooth scrolling* di halaman web, yang memungkinkan user menggulir halaman dengan cara yang lebih halus dan terkontrol, baik menggunakan *mouse wheel* maupun keyboard.

3.2.4 Creating Component for Front-end Project

Component adalah desain UI/UX yang diubah dari platform desain menjadi kode HTML dan CSS yang diterapkan pada *Astro*. Dalam proyek ini, terdapat folder *component* yang berisi folder *element*, *layouts*, dan *section*, yang masing-masing menyimpan file komponen UI/UX untuk *website*. Sebagai langkah awal agar tampilan lebih baik, digunakan file *global.css* yang berada dalam folder *styles*.

a. Making Layouts Components

Di dalam folder *layouts* terdapat beberapa file *Header.astro*, *Footer.astro*, dan *Base.astro*. *Header.astro* yang berfungsi untuk menyusun bagian navbar yang mencakup menu navigasi yang memungkinkan pengguna untuk berpindah antar halaman dengan mudah, dan memilih bahasa yang diinginkan. *Footer.astro* berisi elemen-elemen seperti pesan singkat yang biasanya menunjukkan hak cipta, menu navigasi tambahan dan link menuju media sosial Radyakartiyasa. *Base.astro* mendefinisikan struktur dasar halaman dengan *header*, konten utama, dan *footer*, serta menyertakan berbagai *css* dan *javascript* untuk memperkaya interaktivitas dan desain halaman.

1. *Header.astro* yang digunakan untuk navigasi *website* yang terdiri dari:

```
import { Image } from 'astro:assets';
import { fetch_data , file_handle } from "@utils/projectUtils.js";
import logoImage from '@assets/img/logo-white.webp';
```

Gambar 3.25 Mengimpor fungsi dan data gambar

Gambar 3.25 menunjukkan kode ini mengimpor beberapa fungsi dan data yang digunakan untuk menampilkan elemen-elemen di halaman. *Image* digunakan untuk mengimpor gambar melalui *Astro*, dan dalam hal ini, gambar logo dengan nama *logo-white.webp* diimpor untuk ditampilkan di *header* situs. Fungsi *fetch_data* digunakan untuk mengambil data dari API, yang berfungsi untuk mengambil informasi terkait navigasi dan bahasa yang digunakan di halaman. Sementara itu, *file_handle* digunakan untuk mengelola file, seperti mendapatkan URL gambar atau file lainnya yang disimpan di CMS. Terakhir, *logoImage* adalah variabel yang mengimpor gambar logo dalam format *webp*, yang kemudian digunakan di *header* untuk menampilkan logo tersebut.

```
const data_navigations = await fetch_data(`navigations?fields=*,translations.*,sub_navigation.*,
sub_navigation.translations.*&filter[parent][_null]=true&deep[sub_navigation][translations][_filter]
[languages_code][_eq]=${lang}`, lang);
const data_languages = await fetch_data('languages?fields=*,lang_flag.*');
```

Gambar 3.26 Data_navigations

Kode *data_navigations* mengambil konten dengan *endpoint* yang telah ditentukan, disesuaikan dengan bahasa yang dipilih pengguna. Kode *data_languages* mengambil data bahasa yang tersedia beserta ikon bendera untuk dropdown pemilihan bahasa. Semua data ini digunakan untuk menampilkan navigasi dan pilihan bahasa yang disesuaikan dengan preferensi user (Gambar 3.26).

Header utama dari *website* yang berfungsi sebagai *container* untuk navigasi dan elemen penting lainnya. Diberi *class header* dan *id header* untuk styling dan manipulasi *JavaScript*. Sebuah pembungkus *container* yang mengelompokkan konten *header*, memberikan batasan lebar agar tampilan lebih terstruktur dan responsif.

Membungkus seluruh elemen dalam *header* untuk mengatur *layout* atau struktur internal dengan styling tertentu. Area logo *website*, berisi logo yang akan ditampilkan ke halaman utama, menggunakan tag *a* dengan tautan *import*. Bagian utama *header* yang berisi navigasi dan elemen lainnya.

```

<ul class="main-navigation navbar-nav">
  {data.navigations.map((item, index) => {
    const nav_active = (activePage === '/' + item.URL) ? 'active': '';

    return item.showed_on.includes("navbar") ? (
      item.sub_navigation.length === 0 ? (
        <li class="nav-item">
          <a class="nav-link {nav_active}" href="/{lang}/{item.URL}"><span>{item.translations[0].text}</span></a>
        </li>
      ) : (
        <li class="nav-item dropdown" id="dropdown-{index}">
          <a href="javascript:void(0)" class="nav-link dropdown-toggle" data-bs-target="#dropdown-menu-1" data-bs-toggle="dropdown" aria-expanded="false">{item.translations[0].text}</a>
          <ul class="dropdown-menu" aria-labelledby="dropdown-{index}">
            {item.sub_navigation.map((item_sub, index_sub) => {
              <li class="dropdown-item section-description">
                <a class="nav-link {nav_active}" href="/{lang}/{item_sub.URL}">{item_sub.translations[0].text}</a>
              </li>
            })}
          </ul>
        </li>
      )
    ) : null;
  })}
</ul>

```

Gambar 3.27 Main navigation

Navigasi utama *website* yang menampilkan menu dengan daftar link yang dapat di klik untuk berpindah halaman. Terdapat kelas *main-navigation navbar-nav* yang berisi kelas *nav-item* untuk setiap menu dan sub-menu. Kelas *dropdown-menu* untuk menampilkan sub-menu yang dapat dipilih (Gambar 3.27).

```

<div class="header-action">

```

Gambar 3.28 Header-action

Bagian ini berisi elemen-elemen tambahan, seperti pengaturan bahasa dan tombol untuk membuka menu responsif (Gambar 3.28).

```

<div class="dropdown dropdown-menu-left dropdown-lang">

```

Gambar 3.29 Dropdown

Dropdown untuk memilih bahasa. Menggunakan gambar bendera *img* untuk menunjukkan pilihan bahasa yang tersedia (Gambar 3.29).

```

<a href="javascript:void(0)" class="header-close-navigation hv-trigger">
  <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" fill="none">
    <path d="M13.4118 12.005L17.7071 7.71321C18.0976 7.32305 18.0976 6.69279 17.7071 6.30263C17.3166 5.91246 16.6859 5.91246 16.2954 6.30263L12 10.5944L7.70463 6.29262C7.31414 5.90246 6.68335 5.90246 6.29287 6.29262C5.90238 6.68278 5.90238 7.31305 6.29287 7.70321L10.5882 12.005L6.29287 16.2968C5.90238 16.687 5.90238 17.3172 6.29287 17.7071C6.68335 18.0975 7.31414 18.0975 7.70463 17.7071L12 13.4156L16.2954 17.7071C16.6859 18.0975 17.3166 18.0975 17.7071 17.7071C18.0976 17.3172 18.0976 16.687 17.7071 16.2968L13.4118 12.0052" fill="#ffffff"/>
  </svg>
</a>

```

Gambar 3.30 Closing button

Tombol untuk menutup navigasi atau menu yang muncul untuk mode responsif, menggunakan elemen `<svg>` yang berisi ikon untuk menutup menu (Gambar 3.30).

```

<a href="javascript:void(0);" class="nav-link header-open-navigation">

```

Gambar 3.31 Nav-link

Tombol untuk membuka navigasi atau menu, menggunakan `<svg>` untuk ikon tiga garis (hamburger) yang biasa digunakan pada menu *mobile* (Gambar 3.31).

```
<div class="header-action">
  <div class="nav-item">
    <div class="dropdown dropdown-menu-left dropdown-lang">
      <a href="javascript:void(0)" class="nav-link dropdown-toggle rmarrow"
        id="lang-dropdown" data-bs-toggle="dropdown">
        <img src={file_handle(data_languages.filter((item)=>{return item.
          code==lang})[0].lang_flag)} class="img-fluid lang-img"
          loading="lazy" decoding="async" />
      </a>
    </div>
  </div>
</div>
```

Gambar 3.32 Header-action

Menampilkan gambar bendera bahasa yang aktif, yang diambil menggunakan `file_handle` berdasarkan bahasa yang dipilih (`lang`). Ketika tombol ini di klik, *dropdown* akan muncul dan menampilkan daftar bahasa yang tersedia (Gambar 3.32).

```
<div class="dropdown-menu dropdown-menu-end"
  aria-labelledby="lang-dropdown">
  {data_languages.map((val_lang) => {
    const img_lang = file_handle(val_lang.lang_flag);
    return (
      <a class="dropdown-item trigg-lang" href="javascript:void(0)"
        data-lang={val_lang.code}>
        {img_lang && (
          <img src={img_lang} alt={val_lang.name} class="icon
            img-fluid lang-img" loading="lazy"
            decoding="async" />
        )}
        <span class="text section-description-sm">{val_lang.
          name}</span>
      </a>
    );
  })};
</div>
```

Gambar 3.33 Dropdown-menu

Daftar bahasa yang tersedia dari mengiterasi `data_languages` dan menampilkan setiap bahasa yang tersedia dengan gambar bendera dan nama bahasa. Saat user memilih bahasa, *event click* pada `trigg_lang` akan memperbarui bahasa halaman dengan mengganti URL berdasarkan kode bahasa yang dipilih (Gambar 3.33).

```
<div class="nav-item">
  <a href="javascript:void(0);" class="nav-link header-open-navigation">
    <svg viewBox="0 0 24 24" fill="none" xmlns="http://www.w3.org/2000/svg">
      <path fill-rule="evenodd" clip-rule="evenodd" d="M3 5H21C21.55 5 22
        4.55 22 4C22 3.45 21.55 3 21 3H3C2.45 3 2 3.45 2 4.55 2 5 3
        5.2M21 11H3C2.45 11 2 11.45 2 12C2 12.55 2.45 13 3 13H21C21.55 13 22
        12.55 22 12C22 11.45 21.55 11 21 11ZM3 19H21C21.55 19 22 19.45 22
        20C22 20.55 21.55 21 21 21H3C2.45 21 2 20.55 2 20C2 19.45 2.45 19 3
        19Z" fill="#ffffff"/>
    </svg>
  </a>
</div>
```

Gambar 3.34 Nav-item

Tombol ini menggunakan ikon *hamburger* yang sering digunakan untuk membuka

navigasi di tampilan *mobile*. Ketika diklik, tombol ini akan membuka atau menutup menu navigasi samping (Gambar 3.34).

```
<script>
  window.addEventListener('load', () => {
    sessionStorage.setItem('lastPage', window.location.href);
  });
```

Gambar 3.35 Storing to *sessionStorage*

Gambar 3.35 Storing to *sessionStorage*

Kode ini menyimpan URL halaman terakhir saat halaman dimuat. URL halaman disimpan di *sessionStorage* dengan kunci *lastPage*. Ini berguna untuk mengganti bahasa tanpa kehilangan halaman yang sedang dilihat (Gambar 3.35).

```
$(document).on('click', '.trigg-lang', function() {
  var lastPage = sessionStorage.getItem('lastPage');
  var lang = $(this).data('lang');
  var updatedUrl = lastPage.replace(/(id-ID|en-US)/, lang);
  window.location.href = updatedUrl;
});
```

Gambar 3.36 Mengelola bahasa

Mengelola perubahan bahasa saat pengguna mengklik pilihan bahasa dalam *dropdown*. Saat user mengklik pilihan bahasa (elemen dengan kelas *.trigg-lang*), kode ini mengambil URL halaman terakhir dari *sessionStorage* dan mengganti kode bahasa (id-ID atau en-US) dengan bahasa yang dipilih (lang). Setelah itu, halaman akan diarahkan ke URL yang diperbarui, yang mencerminkan bahasa baru (Gambar 3.36).

```
document.addEventListener('DOMContentLoaded', () => {
  let lastScrollTop = 0;

  window.addEventListener('scroll', () => {
    const wScroll = window.scrollY;
    const contentHeight = document.body.scrollHeight;
    const header = document.getElementById('header');
    const headerHeight = header?.offsetHeight || 0;

    if (contentHeight > 0) {
      if (wScroll > headerHeight + 500) {
        if (wScroll < lastScrollTop) {
          header.classList.remove('slide-up');
          header.classList.add('slide-down');
        } else {
          header.classList.remove('slide-down');
          header.classList.add('slide-up');
        }
      }

      if (wScroll > headerHeight - 6) {
        header.classList.add('shrink');
        document.querySelector('.header .header-nav')?.classList.remove('shown');
      } else {
        header.classList.remove('shrink');
        header.classList.remove('slide-down');
      }

      lastScrollTop = wScroll;
    }
  });
});
```

Gambar 3.37 wScroll

Kode `wScroll` mendapatkan posisi `scroll` vertikal halaman. Kode `header.classList.add('shrink')` jika pengguna menggulir lebih dari ketinggian `header`, `header` akan menyusut (dengan menambahkan kelas `shrink`). `Slide-up` dan `slide-down` untuk user menggulir ke atas, `header` akan muncul dengan menambahkan kelas `slide-down`. Sebaliknya, jika menggulir ke bawah, `header` akan disembunyikan dengan menambahkan kelas `slide-up` (Gambar 3.37).

Secara keseluruhan, kode ini mengelola navigasi, pilihan bahasa, dan efek `scroll` dinamis pada `header`, memberikan pengalaman pengguna yang interaktif, responsif, dan disesuaikan dengan preferensi bahasa yang dipilih.

2. `Footer.astro` yang berisi menu navigasi, deskripsi singkat dan sosial media `radyakartiyasa` yang terdiri dari:

```
import { fetch_data } from "@utils/projectUtils.js";
import { Image } from 'astro:assets';
import logoImage from '@assets/img/logo-white.webp';
```

Gambar 3.38 *Import package*

Fungsi `fetch_data` digunakan untuk mengambil data dari API, yang memungkinkan aplikasi untuk mengakses dan menampilkan informasi dari sumber eksternal. Di sisi lain, `Image` digunakan untuk mengimpor gambar, seperti gambar logo yang ditampilkan di `header` halaman. Gambar logo tersebut disimpan dalam variabel `logoImage`, yang diimpor dalam format `webp` untuk memastikan kualitas gambar yang baik dengan ukuran file yang lebih kecil, yang digunakan untuk menampilkan logo situs pada tampilan halaman (Gambar 3.38).

```
const {lang} = Astro.props;
const currentYear = new Date().getFullYear();

const data_site = await fetch_data('site_setting?fields=*,translations.*', lang)
const data_navigations = await fetch_data('navigations?fields=*,translations.*', sub_navigation.translations.*&filter[parent][_null]=true&deep[sub_navigation][translations][_filter][languages_code][_eq]=${lang}`, lang);
```

Gambar 3.39 *Variables currentYear*

`Lang` diambil dari properti `Astro.props`, yang berisi bahasa yang dipilih pengguna, seperti `en-US` atau `id-ID`. Data yang diambil dari API kemudian akan disesuaikan dengan bahasa ini. Variabel `currentYear` digunakan untuk mendapatkan tahun saat ini menggunakan

`newDate().getFullYear()`, yang memungkinkan penampilan tahun dinamis, seperti pada hak cipta. Variabel `data_site` mengambil data pengaturan situs dari API, khususnya data `site_setting`, yang mencakup informasi seperti deskripsi situs dan informasi sosial yang disesuaikan dengan bahasa yang dipilih. Sedangkan `data_navigations` mengambil data navigasi utama dan sub-navigasi dari API, dengan menggunakan filter untuk menampilkan menu utama yang tidak memiliki *parent* (`filter[parent][_null]=true`) bersama dengan sub-menu. Terjemahkan untuk sub-navigasi juga disesuaikan dengan bahasa yang dipilih oleh pengguna (Gambar 3.39).

```
<footer class="footer section section-lg">
```

Gambar 3.40 *Class footer*

Elemen *footer* yang berfungsi sebagai bagian bawah halaman, diberi kelas *footer*, *section*, *section-lg* untuk *styling* (Gambar 3.40).

Kelas *container* untuk membungkus konten *footer*, memberikan batasan lebar dan menjaga tampilan responsif.

Menggunakan grid untuk row untuk tata letak kolom dalam *footer*, diatur agar konten distribusinya merata antara kolom, *gap* untuk jarak antar kolom pada berbagai ukuran layar. Menggunakan grid maka secara responsif menyesuaikan lebar kolom pada berbagai ukuran layar yang telah ditentukan.

```
<div class="footer-column">
```

Gambar 3.41 *Footer-column*

```
<div class="footer-brand">
  <a href="/">
    <Image src={logoImage} alt={data_site.site_name} class="img-fluid
      brand-img" />
  </a>
</div>
```

Gambar 3.42 *Footer-brand*

Pada Gambar 3.41 terdapat kelas yang membungkus elemen- elemen kolom pertama yang berisi logo dan deskripsi singkat situs.

Gambar 3.42 terdapat kelas menampilkan logo situs. Menggunakan tag *a* yang mengarah ke halaman utama, dengan gambar logo yang dimuat menggunakan tag *image*.

```
{data_site.translations.map((item_lang) => (
  item_lang.meta_description && (
    <p class="description section-description-md">{item_lang.
      meta_description}</p>
  )
)})}
```

Gambar 3.43 Deskripsi terjemahan sesuai bahasa

Gambar 3.43 menampilkan deskripsi singkat yang diterjemahkan sesuai bahasa yang dipilih.

```
<div class="col-xl-4 col-lg-12">
  <h3 class="footer-title section-description-md">Site Map</h3>
  <div class="row gap-sm-0 gap-2">
    <div class="col-xl-6 col-lg-4 col-sm-6 col-12">
      <div class="footer-column">
```

Gambar 3.44 Sitemap navigasi

Kolom kedua yang menampilkan *sitemap* berisi menu navigasi untuk halaman-halaman utama. Judul bagian untuk *sitemap*. Row yang mengelompokkan kolom *sitemap* untuk menu navigasi dengan gap. Kolom untuk menampilkan letak menu navigasi (Gambar 3.44).

```
<div class="list-info">
  { data_navigations.slice(0,4).map((item_nav) => {
    let url_menu = item_nav.URL != null ? `/${lang}/${item_nav.URL}`
      : `/${lang}`;
    return item_nav.showed_on.includes("site-map") ? (
      <div class="list-item">
        <a href= {url_menu} class="section-description-md">
          {item_nav.translations[0]?.text || "Teks Kosong"}
        </a>
      </div>
    ) : null;
  })
}</div>
```

Gambar 3.45 Daftar navigasi

Menampilkan daftar navigasi dengan setiap item ditampilkan menggunakan kelas *list-item*. Tautan navigasi mengarah ke halaman yang sesuai berdasarkan data *item_nav.URL* dan *lang* (Gambar 3.45).

```

<div class="col-xl-6 col-lg-4 col-sm-6 col-12">
  <div class="footer-column">
    <div class="list-info">
      { data_navigations.slice(4).map((item_nav_2) => {
        let url_menu = item_nav_2.URL != null ? `/${lang}/${item_nav_2.URL}` : '';
        return item_nav_2.showed_on.includes("site-map") ? (
          <div class="list-item">
            <a href={url_menu} class="section-description-md">
              {item_nav_2.translations[0]?.text || "Teks Kosong"}
            </a>
          </div>
        ) : null;
      })}
    </div>
  </div>
</div>

```

Gambar 3.46 Tambahan konten navigasi

Kolom kedua untuk menampilkan jika lebih banyak item navigasi dari *sitemap* (Gambar 3.46).

```

<div class="copyright">
  <p class="section-description-sm">&copy; {currentYear} Karaton Yogyakarta. All rights reserved.</span>

```

Gambar 3.47 Copyright

Gambar 3.47 menunjukkan kode yang menampilkan informasi dengan tahun dinamis menggunakan *currentYear*. Tahun ini diambil dengan *new Date().getFullYear()* untuk menampilkan tahun yang tepat.

```

<div class="social-media">
  {data_site.socials.map((val_social) => (
    val_social.Social_Network === "facebook" && (
      <div class="social-media-item">
        <a href={val_social.Account} class="icon" target="_blank" rel="noopener noreferrer"><i class="feather icon-facebook"></i></a>
      </div>
    ) || val_social.Social_Network === "instagram" && (
      <div class="social-media-item">
        <a href={val_social.Account} class="icon" target="_blank" rel="noopener noreferrer"><i class="feather icon-instagram"></i></a>
      </div>
    ) || val_social.Social_Network === "x" && (
      <div class="social-media-item">
        <a href={val_social.Account} class="icon" target="_blank" rel="noopener noreferrer"><i class="feather icon-twitter"></i></a>
      </div>
    ) || val_social.Social_Network === "youtube" && (
      <div class="social-media-item">
        <a href={val_social.Account} class="icon" target="_blank" rel="noopener noreferrer"><i class="feather icon-youtube"></i></a>
      </div>
    ) || val_social.Social_Network === "linkedin" && (
      <div class="social-media-item">
        <a href={val_social.Account} class="icon" target="_blank" rel="noopener noreferrer"><i class="feather icon-linkedin"></i></a>
      </div>
    )
  )}

```

Gambar 3.48 Social-media

Gambar 3.48 menunjukkan bagian yang menampilkan ikon media sosial. Ikon diwakili oleh elemen kelas *social-media-item* dengan tautan yang mengarah ke akun sosial media yang relevan. Menggunakan ikon dari pustaka *feather*.

Kode ini mengiterasi data sosial yang ada pada *data_site.socials* dan memeriksa jenis jaringan sosial. Untuk masing-masing jenis jaringan sosial terdapat tautan *val_social.Account* mengarah ke akun media sosial terkait. Target bernilai *_blank* memastikan tautan terbuka di tab baru. *Noopener noreferrer* untuk meningkatkan keamanan saat membuka tautan di tab baru.

```

| val_social.Social_Network == "whatsapp" && (
|   <div class="social-media-item">
|     <a href="{ 'https://api.whatsapp.com/send?phone=${val_social.Account}'}"
|       class="icon" target="_blank" rel="noopener noreferrer">
|       <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24"
|         viewBox="0 0 36 36" fill="none">
|         <path d="M28.5742 7.36509C27.199 5.97598 25.561 4.87459 23.7557
| 4.12517C21.9584 3.37574 20.0139 2.99328 18.0592 3.00009C9.86922
| 3.00009 3.19422 9.67509 3.19422 17.8651C3.19422 20.4901 3.88422
| 23.0401 5.17422 25.2901L3.07422 33.0001L10.9492 30.9301C13.1242
| 32.1151 15.5692 32.7451 18.0592 32.7451C26.2492 32.7451 32.9242
| 26.0701 32.9242 17.8801C32.9242 13.9051 31.3792 10.1701 28.5742
| 7.36509ZM18.0592 30.2251C15.8392 30.2251 13.6642 29.6251 11.
| 7592 28.5001L11.3092 28.2301L6.62922 29.4601L7.87422 24.9001L7.
| 57422 24.4351C6.34084 22.4655 5.68593 20.189 5.68422 17.8651C5.
| 68422 11.0551 11.2342 5.50509 18.0442 5.50509C21.3442 5.50509
| 24.4492 6.79509 26.7742 9.13509C27.9254 10.281 28.8378 11.6441
| 29.4583 13.1452C30.0789 14.6464 30.3953 16.2558 30.3892 17.
| 8801C30.4192 24.6901 24.8692 30.2251 18.0592 30.2251ZM24.8392
| 20.9851C24.4642 20.8051 22.6342 19.9051 22.3042 19.7701C21.9592
| 19.6501 21.7192 19.5901 21.4642 19.9501C21.2092 20.3251 20.5042
| 21.1651 20.2942 21.4051C20.0842 21.6601 19.8592 21.6901 19.4842
| 21.4951C19.1092 21.3151 17.9092 20.9101 16.4992 19.6501C15.3892
| 18.6601 14.6542 17.4451 14.4292 17.0701C14.2192 16.6951 14.3992
| 16.5001 14.5942 16.3051C14.7592 16.1401 14.9692 15.8701 15.1492
| 15.6601C15.3292 15.4501 15.4042 15.2851 15.5242 15.0451C15.6442
| 14.7901 15.5842 14.5801 15.4942 14.4001C15.4042 14.2201 14.6542
| 12.3901 14.3542 11.6401C14.0542 10.9201 13.7392 11.0101 13.5142
| 10.9951H12.7942C12.5392 10.9951 12.1492 11.0851 11.8042 11.
| 4601C11.4742 11.8351 10.5142 12.7351 10.5142 14.5651C10.5142 16.
| 3951 11.8492 18.1651 12.0292 18.4051C12.2092 18.6601 14.6542 22.
| 4101 18.3742 24.0151C19.2592 24.4051 19.9492 24.6301 20.4892 24.
| 7951C21.3742 25.0801 22.1842 25.0351 22.8292 24.9451C23.5492 24.
| 8401 25.0342 24.0451 25.3342 23.1751C25.6492 22.3051 25.6492 21.
| 5701 25.5442 21.4051C25.4392 21.2401 25.2142 21.1651 24.8392 20.
| 9851Z" fill="white"/>
|     </a>
|   </div>
| )

```

Gambar 3.49 *Whatsapp*

Jaringan sosial WhatsApp akan menggunakan tautan dengan URL API WhatsApp untuk mengirim pesan langsung melalui nomor telepon yang disediakan (Gambar 3.49).

```

| val_social.Social_Network == "email" && (
|   <div class="social-media-item">
|     <a href="{ 'mailto:${val_social.Account}'}" class="icon" target="_blank"
|       rel="noopener noreferrer"><i class="feather icon-mail"></i></a>
|   </div>
| )
| )

```

Gambar 3.50 *Email*

Jaringan sosial adalah email, maka tautan *mailto:* digunakan untuk membuka aplikasi email dengan alamat yang ditentukan (Gambar 3.50).

File ini menampilkan hak cipta dan ikon media sosial yang dapat diklik di bagian *footer*. Ikon media sosial ditampilkan sesuai dengan data yang tersedia pada *data_site.socials*, dan pengguna dapat diarahkan ke akun sosial yang sesuai, seperti facebook dan email. Selain itu, gambar ikon sosial diambil dan ditampilkan menggunakan kelas *feather icon* untuk masing-masing platform sosial.

3. Base.astro yang digunakan untuk membentuk dasar halaman *website* Radyakartiyasa dengan elemen-elemen yang dapat diakses dan dinamis:

```
<html translate="no">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <meta name="robots" content="index, follow" />
  <meta name="googlebot" content="index, follow, max-video-preview:-1, max-image-preview:large, max-snippet:-1" />
  <meta name="theme-color" content="#0E3121" />
  <meta name="description" content="Kraton Ngayogyakarta Hadiningrat merupakan istana resmi Kesultanan Ngayogyakarta Hadiningrat yang kini berlokasi di Kota Yogyakarta, Daerah Istimewa Yogyakarta, Indonesia." />
  <meta name="keywords" content="Kraton Ngayogyakarta Hadiningrat" />
  <meta name="author" content="kraton crew" />
  <meta name="generator" content={Astro.generator} />
</head>
<body>
  <div>
    <h1>{pageTitle}</h1>
  </div>
</body>
</html>
```

Gambar 3.51 *HTML meta tags*

Penggunaan `translate="no"` adalah atribut yang digunakan untuk memberi tahu mesin penerjemah agar tidak menerjemahkan halaman web tersebut meskipun user dari negara atau wilayah yang berbeda dapat memilih bahasa inggris selain indonesia. Kode `utf-8` untuk menentukan karakter set untuk halaman web. *Meta viewport* untuk menyesuaikan tampilan halaman agar responsif pada berbagai ukuran layar seperti perangkat *mobile*. *Meta robots* untuk memberitahu mesin pencari untuk mengindeks halaman dan mengikuti(*follow*) tautan di dalamnya. *Meta Googlebot* memiliki *max-video-preview* untuk *Googlebot* tidak akan menampilkan pratinjau video apapun pada hasil pencarian, *max-image- preview* memberikan izin kepada *Google* untuk menampilkan pratinjau gambar berukuran besar di hasil pencarian, *max-snippet* menentukan bahwa potongan teks yang ditampilkan di hasil pencarian akan diukur oleh *Google* untuk membantu user mencari konten dan mengarahkan ke situs. *Meta theme color* digunakan untuk mengatur warna tema di perangkat *mobile* atau aplikasi web yang mendukung pengaturan tema warna. *Meta description* untuk menyediakan deskripsi singkat tentang halaman yang akan ditampilkan di hasil pencarian. *Meta keywords* untuk menyediakan kata kunci yang relevan untuk *search engines*. *Meta author* untuk menyebutkan pembuat atau pemilik situs. *Meta generator* untuk menyebutkan alat(generator) yang digunakan untuk membuat halaman ini. *Title* memiliki nilai `{pageTitle}` untuk mendinamiskan tampilan judul halaman sesuai dengan laman yang dikunjungi di tab browser (Gambar 3.51).

```

<title>{pageTitle}</title>

<meta name="apple-mobile-web-app-capable" content="yes" />
<meta name="apple-mobile-web-app-status-bar-style" content="#0E3121" />
<meta name="apple-mobile-web-app-title" content="" />

<link rel="apple-touch-icon" href="/assets/img/favicon.webp" />
<link rel="shortcut icon" href="/assets/img/favicon.webp" />
<link rel="icon" type="image/x-icon" href="/assets/img/favicon.webp" />

<link rel="stylesheet" href="/assets/vendors/feather/css/iconfont.min.css" />

<link rel="preconnect" href="https://fonts.googleapis.com" />
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
<link href="https://fonts.googleapis.com/css2?family=Tenor+Sans&display=swap" rel="stylesheet" />
<link href="https://fonts.googleapis.com/css2?family=Anek+Bangla:wght@100;200;300;400;500;600;700;800&display=swap" rel="stylesheet" />
<link href="https://fonts.googleapis.com/css2?family=Inter:ital,opsz,wght@14..32,100..900;1,14..32,100..900&display=swap" rel="stylesheet" />
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm30Dgp3LW9Nac84YzicQTWf3ygD65VohhpuuCOmLASjC" crossorigin="anonymous" />
<link href="https://cdn.jsdelivr.net/npm/select2@4.1.0-rc.0/dist/css/select2.min.css" rel="stylesheet" />
<link href="https://cdn.jsdelivr.net/npm/swiper@9/swiper-bundle.min.css" rel="stylesheet" />
<link href="https://cdn.jsdelivr.net/npm/magnific-popup@1.2.0/dist/magnific-popup.min.css" rel="stylesheet" />
<link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.2/css/all.min.css" rel="stylesheet" />

<!-- <ClientRouter /> -->
</head>

```

Gambar 3.52 *Meta apple-mobile-web-app-capable*

Meta apple-mobile-web-app-capable menyatakan bahwa aplikasi web mendukung tampilan layar penuh di perangkat *apple*. *Meta apple-mobile-web-app-status-bar-style* akan menunjukkan status bar sesuai dengan warna yang ditentukan. *Meta apple-mobile-web-app-title* tidak menentukan judul aplikasi web di perangkat *apple*. *Meta apple-touch-icon* untuk menetapkan ikon aplikasi untuk perangkat *apple* jika pengguna menyimpannya sebagai aplikasi web. *Link shortcut icon* dan *link icon* menyediakan gambar favicon yang berisi logo radya. *Link stylesheet* menyertakan file CSS untuk tampilan situs, termasuk penggunaan pustaka seperti *Feather Icons*, *Bootstrap*, dan *Swiper*, serta *font* dari *Google fonts* (Gambar 3.52).

```

<body class="">

  <Header lang={lang} activePage={activePage}></Header>

  <main class="wrap-site" id="wrap-site">
    <slot/>
  </main>

  <Footer lang={lang}></Footer>

  <Backtotop></Backtotop>

```

Gambar 3.53 *html semantics tags*

Tag *Header* untuk memasukkan komponen *header.astro* ke *base.astro* dengan parameter *lang*(bahasa) dan *activePage* yang menunjukkan halaman aktif dengan menggunakan *astro import*. Tag *main* dengan kelas dan id yaitu *wrap-site* bagian utama halaman tempat konten dinamis menggunakan *astro* diletakkan, yang berfungsi untuk menampilkan isi halaman menggunakan `<slot/>`. Jadi slot akan diisi dengan konten dari halaman spesifik yang sedang dimuat. Tag *Footer* dengan parameter *lang* untuk bahasa yang aktif serta memasukkan komponen *Footer.astro* dengan menggunakan *astro import*. Tag *Backtotop* adalah komponen tombol “Kembali ke atas” yang memungkinkan user untuk kembali ke bagian atas halaman dengan mudah (Gambar 3.53)

Script jQuery digunakan untuk manipulasi DOM dan interaksi UI. *Select2* menambahkan fungsionalita dropdown yang lebih canggih untuk elemen seleksi. *Jquery-validation* untuk memvalidasi formulir pengguna. *Popper.js* digunakan oleh bootstrap untuk posisi elemen pop-up (seperti dropdown). Bootstrap sebagai framework untuk tata letak responsif dan elemen UI interaktif. *Swiper* untuk membuat *slider* gambar atau konten. *Magnific popup* untuk membuat *popup* atau *lightbox* untuk gambar dan konten lainnya. *Plugin.js* dan *main.js* sebagai custom scripts yang menyediakan fungsionalitas khusus untuk situs, seperti interaktivitas dan *plugin* tambahan (Gambar 3. 54).

File ini ini mengelola elemen-elemen *header*, *footer*, dan halaman utama pada situs web

```
<script is:inline src="https://code.jquery.com/jquery-3.7.0.min.js"
integrity="sha256-2PmVv0kU780enSVLm6bvf8SSHxUJ+3A7x6P5Ebd07/g=" crossorigin="anonymous"></script>
<script is:inline src="https://cdn.jsdelivr.net/npm/select2@4.1.0-rc.0/dist/js/select2.min.js"></script>
<script is:inline src="https://cdn.jsdelivr.net/npm/jquery-validation@1.21.0/dist/jquery.validate.min.js"></script>
<script is:inline src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js" integrity="sha384-IQsoLXl5PILFhosVNubq5LC7Qb9DXgDA91+tq8ZJ3iwWAwPtGFTxbJ8NT4GN1R6p" crossorigin="anonymous"></script>
<script is:inline src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js" integrity="sha384-cVKIPhGWiC2Al4u+LWgxfKTRIcfu0JTxR+EQDzBgldoEyl4H0zUf0qQbR30EcQF" crossorigin="anonymous"></script>
<script is:inline src="https://cdn.jsdelivr.net/npm/swiper@9/swiper-bundle.min.js"></script>
<script is:inline src="https://cdn.jsdelivr.net/npm/magnific-popup@1.2.0/dist/jquery.magnific-popup.min.js"></script>
<script is:inline src="/assets/js/plugins.js"></script>
<script is:inline src="/assets/js/main.js"></script>
</body>
</html>
```

Gambar 3. 54 *Script jquery*

dengan menggunakan *Astro*. Data untuk navigasi, pengaturan situs, dan bahasa diambil dari API, memastikan konten halaman disesuaikan dengan bahasa yang dipilih pengguna. Skrip eksternal dan styling diimpor untuk mendukung fitur interaktif seperti *slider*, dropdown, dan validasi formulir. Selain itu, favicon dan ikon aplikasi diatur untuk memberikan pengalaman pengguna yang lebih baik pada browser dan perangkat *mobile*.

b. Making Section component

File-file dalam folder *section* berisi konten khusus dari setiap bagian situs *homepage*. File-file ini akan menggunakan *Base.astro* sebagai layout utama dengan cara memasukkan kontennya ke dalam `<slot/>` yang tersedia di *Base.astro*. Folder *section* menyediakan isi konten dinamis yang *reusable* dan akan di render di dalam struktur *Base.astro* tersebut.

1. *About.astro* sebagai *section homepage* untuk menampilkan penjelasan singkat tentang situs radyakartiyasa dan tautan yang mengarahkan ke laman about us radyakartiyasa terdiri dari:

```

---
import { fetch_data, strip_tags, get_interface } from "@utils/projectUtils.js";
const {lang, interfaces} = Astro.props;

const data_about_us = await fetch_data('about_us?fields=*,translations.sub_title,translations.title,translations.short_description', lang);
const inf_read_more = get_interface(interfaces,"read-more")[0];
---

```

Gambar 3.55 *Import about.astro*

Fetch_data digunakan untuk mengambil data dari CMS dengan filter params yang telah ditentukan. *Strip_tags* akan menghapus tag HTML dari string, sehingga hanya teks murni yang ditampilkan. *Get_interface* untuk mencari dan mengambil elemen dari daftar *interfaces* berdasarkan kunci yang didapatkan. Nilai *lang*(bahasa) dan *interfaces* dari properti yang diteruskan ke komponen *astro*. Nilai *data_about_us* mengambil data dari *endpoint* *about_us* dengan filter untuk mengambil *sub_title*, *title*, dan *short_description* yang sudah diterjemahkan sesuai bahasa yang dipilih. Nilai *inf_read_more* mengambil elemen pertama dari *interfaces* yang memiliki key “read-more”. Fungsi keseluruhan kode digunakan untuk mengambil data dinamis tentang “about us” dan informasi terkait tombol tautan berdasarkan bahasa yang dipilih (Gambar 3.55).

```

<section class="about-section section section-lg">
  <div class="container">
    <div class="inner">
      <div class="head-area section-title mb-0">
        <p class="subtitle badge badge-sm badge-cs section-description-sm">{data_about_us.translations[0].sub_title}</p>
        <h2 class="title section-heading font-2">{data_about_us.translations[0].title}</h2>
        <a href={`/${lang}/about`} title={inf_read_more.translations[0].hover_text} class="link link-arrow second">{inf_read_more.translations[0].text}</a>
      </div>
    </div>
  </div>
</section>

```

Gambar 3.56 *Section about.astro*

Elemen `<section>` ini membungkus seluruh bagian “About us”.Kelas *about-section*, *section*, dan *section-lg* digunakan untuk *styling* dan pengaturan jarak sesuai dengan desain proyek. Kelas *container* digunakan untuk membatasi lebar konten agar tidak *overflow* layar

(menggunakan grid sistem atau pembatas *padding*). Digunakan oleh Bootstrap untuk *styling*. Kelas *inner* pembungkus untuk isi bagian dalam *section*. Kelas *head-area section-title* merupakan *styling* bagian judul selamat datang ke situs radyakartiyasa dengan *mb-0* memastikan tidak ada jarak bawah tambahan. Di dalamnya memiliki kelas *subtitle badge badge-sm badge-cs section-description-sm* menampilkan subjudul sebagai *badge* kecil. Kelas *title section-heading font-2* menampilkan judul utama dari *section*. Kelas *link link-arrow second* digunakan untuk tampilan link dengan ikon panah. Tautan */about* yang mengarah ke halaman “*about us*” sesuai dengan bahasa yang dipilih. *Title* dengan *hover_text* digunakan untuk memunculkan teks saat pengguna mengarahkan kursor ke tautan. Teks tautan muncul dengan *inf_read_more.translations[0].text* yang berisi “Selengkapnya” atau “*Read more*” (Gambar 3.56).

```
<div class="body-area">
  <p class="description section-description-md text-justify">{strip_tags(data_about_us.
    translations[0].short_description)}</p>
  <a href='{/s(lang)/about}' title={inf_read_more.translations[0].hover_text} class="link
    link-arrow second">{inf_read_more.translations[0].text}</a>
</div>
</div>
</section>
```

Gambar 3.57 *Body-area about.astro*

Kelas *body-area* digunakan untuk membungkus deskripsi lebih rinci dari bagian *section AboutUs*. Di dalamnya terdapat kelas *description* memberi gaya pada deskripsi teks, *section-description-md* mengatur ukuran teks menjadi medium, *text-justify* memastikan bahwa teks paragraf ini diratakan ke kiri dan kanan sehingga tampil rapi. Tautan *href/about* sama fungsinya seperti tautan di bagian atas (Gambar 3.57).

Secara keseluruhan, file ini menampilkan halaman “About Us” yang kontennya disesuaikan dengan bahasa yang dipilih, memungkinkan user untuk melihat informasi singkat dan mengakses halaman lebih lengkap melalui tautan “Selengkapnya”.

2. *Destination.astro* menampilkan *section* destinasi yang tersedia dengan penjelasan singkat tentang destinasi tersebut dengan pilihan “lihat detail” untuk melihat penjelasan yang lebih lengkap dan “Booking sekarang” untuk langsung melihat pilihan tiket yang tersedia. *Destination.astro* terdiri dari:

Strip_tags digunakan untuk menghapus tag HTML dari string dan hanya menampilkan teks murni. *Slugify* mengubah teks menjadi format URL-friendly dengan mengganti spasi dan karakter khusus menjadi tanda hubung(-). *File_handle* untuk menangani file gambar yang

teroptimasi untuk menentukan jalur file. `Get_interface` untuk mengambil elemen *interface* dengan *key* tertentu.

Lang untuk menyimpan informasi bahasa yang dipilih antara ‘en’ dan ‘id’. *Interfaces* berisi konfigurasi antarmuka pengguna, termasuk teks dan elemen UI lainnya. *Data_destination* berisi data terkait dengan destinasi yang dipilih dalam slider.

Kelas *swiper* sebagai tempat untuk menampilkan slider/*swiper* destinasi dan id *destination-swiper* digunakan untuk menghubungkan dengan kode javascript nanti. Kelas *swiper-wrapper* sebagai pembungkus untuk semua slide dalam *swiper*. Iterasi *data_destination* menggunakan `map()`. `Item.id` digunakan untuk membuat URL dinamis. `Slugify(item.translations[0].title)` mengubah judul destinasi menjadi format URL yang ramah. `File_handle(item.thumbnail)` mengambil gambar dari path file gambar destinasi. `Strip_tags(item.translations[0].short_description)` untuk menghapus tag HTML dari deskripsi singkat destinasi. `Get_interface(interfaces, "see- detail")` mengambil teks tombol “See Detail” untuk destinasi dari konfigurasi antarmuka. `Item.ticketing_url` akan menampilkan tombol jika URL untuk pemesanan tiket itu ada.

`Addeventlistener(‘DOMContentLoaded’, )` digunakan untuk menunggu sampai halaman selesai dimuat, lalu inisialisasi *swiper*. Kode `new swiper` membuat instance baru dari *swiper* untuk elemen dengan ID *destination-swiper*. Kode *lazy* untuk mengaktifkan *lazy loading* gambar. Loop memiliki nilai `false` untuk menonaktifkan loop slide sehingga tidak ada slide yang terulang kembali. Kode `spacebetween` menetapkan jarak antara slide dengan nilai 16. Kode `autoplay` memiliki nilai `false` untuk menonaktifkan pemutaran otomatis. Kode `breakpoints` menetapkan konfigurasi responsif untuk berbagai ukuran layar.

Kode ini membuat *slider* destinasi wisata yang menampilkan informasi seperti gambar, judul, deskripsi singkat, dan tautan untuk melihat detail atau memesan tiket. *Swiper* digunakan untuk membuat tampilan interaktif dan responsif dengan pengaturan yang menyesuaikan jumlah slide berdasarkan ukuran layar perangkat. *loading* diterapkan pada gambar untuk mengoptimalkan pemuatan halaman, dan seluruh tampilan disesuaikan dengan bahasa yang dipilih oleh pengguna.

3. *Event.astro* untuk menampilkan section *event* di homepage yang menampilkan acara yang tersedia seperti gambar, tanggal mulainya, judul, deskripsi singkatnya, dan tautan untuk melihat penjelasan yang lebih lengkap. *Event.astro* terdiri dari:

Mengimpor berbagai fungsi yang digunakan untuk memproses data, seperti menghapus tag HTML(`strip_tags`), membuat URL ramah mesin pencari(`slugify`), menformat tanggal(`date_format`), menangani file gambar(`file_handle`), dan mengambil elemen antarmuka pengguna(`get_interface`).

Kelas *swiper* digunakan untuk membuat *slider* acara. `Data_event.map(...)` untuk looping data acara dan membuat slide untuk setiap acara. `Item.thumbnail` untuk menampilkan gambar acara. `Item.open_date` untuk menampilkan tanggal acara dengan menggunakan `date_format`. `Item.translations[0].title` untuk menampilkan judul acara. `Item.translations[0].short_description` untuk deskripsi singkat acara. Tautan `/event/${item.id}` untuk ke laman acara tersebut yang diformat dengan `slugify`. Tulisan “Lihat Detail” menggunakan `get_interface` untuk menampilkan teks tombol yang dapat disesuaikan dengan bahasa yang dipilih.

```
<script>
// document.addEventListener('astro:page-load', () => {
document.addEventListener('DOMContentLoaded', () => {

  new Swiper('#event-swiper', {
    lazy: true,
    loop: false,
    spaceBetween: 16,
    centeredSlides: false,
    initialSlide: 0,
    preloadImages: true,
    autoplay: false,
    breakpoints: {
      0: {
        slidesPerView: 1.2,
        spaceBetween: 16,
      },
      580: {
        slidesPerView: 1.5,
        spaceBetween: 20,
      },
      768: {
        slidesPerView: 2.1,
      },
      991: {
        slidesPerView: 2.6,
        spaceBetween: 20,
      },
      1200: {
        slidesPerView: 3,
        spaceBetween: 30,
      },
    },
  });
});
</script>
```

Gambar 3.58 *Swiper event.astro*

Menunggu hingga halaman dimuat(`DOMContentLoaded`), kemudian *Swiper* diinisialisasi pada elemen dengan id `event-swiper`. memiliki nilai `true` mengaktifkan lazy loading untuk gambar. Loop bernilai `false` untuk menonaktifkan loop *slider*. `SpaceBetween` menetapkan jarak antar slide dengan nilai 16. `Breakpoints` menentukan jumlah slide yang ditampilkan berdasarkan ukuran layar (Gambar 3.58).

4. *Faq.astro* untuk menampilkan pertanyaan yang sering ditanyakan dengan jawaban yang sudah tersedia. *Faq.astro* terdiri dari:

```
import { fetch_data, strip_tags , file_handle } from "@utils/projectUtils.js";
const { data_faq } = Astro.props;
// const data_faq = await fetch_data('qnas_translations?fields=languages_code,q
filter={"languages_code":"'${lang}'"}');
```

Gambar 3.59 Tag *faq.astro*

Bagian ini pertama-tama mengimpor *strip_tags* untuk menghapus tag HTML. Data FAQ(*data_faq*) diteruskan ke komponen ini sebagai properti (*Astro.props*), yang kemudian diiterasikan menggunakan *map* untuk merender setiap item FAQ dalam bentuk *accordion* dengan menggunakan *bootstrap* (Gambar 3.59).

Setiap item FAQ terdiri dari dua bagian utama: tombol yang berisi pertanyaan dengan kelas *accordion-head section-description* dan *div* yang berisi jawaban dengan kelas *accordion-body*. Tombol tersebut menggunakan kelas *bootstrap* untuk mengatur perilaku *collapse*, di mana tombol pertama akan terbuka secara *default*, sementara tombol lainnya akan dimulai dalam keadaan tertutup. Atribut *aria-expanded* dan *data-bs-toggle* digunakan untuk memastikan bahwa tombol dapat mengontrol elemen yang dapat dilipat, yaitu jawaban dari setiap pertanyaan. Fungsi *strip_tags* digunakan untuk menghapus tag HTML dari teks pertanyaan dan jawaban, memastikan hanya teks bersih yang ditampilkan. Bagian jawaban pada setiap item FAQ akan terbuka atau tertutup sesuai dengan interaksi pengguna, dan hanya jawaban dari FAQ pertama yang akan terbuka secara *default*.

Dengan cara ini, komponen FAQ ini memungkinkan pengguna untuk menavigasi pertanyaan yang sering diajukan dengan cara yang interaktif dan dinamis, membuatnya lebih mudah untuk mengakses informasi tanpa membebani tampilan halaman dengan teks panjang.

5. *Gallery.astro* adalah *section* yang menampilkan foto-foto untuk memberikan gambaran visual yang menarik kepada pengunjung situs tentang tempat-tempat wisata atau destinasi yang tersedia. *Gallery.astro* terdiri dari:

File_handle berfungsi untuk mendapatkan URL gambar teroptimasi yang benar dari file yang ada di CMS. Data gambar(*data_gallery*) diteruskan ke komponen melalui properti *Astro.props*, yang berisi informasi mengenai gambar yang akan ditampilkan dalam galeri.

```

<div class="swiper-container">
  <div class="swiper" id="gallery-swiper">
    <div class="swiper-wrapper">
      {data_gallery.map(item, index) => {
        <div class="swiper-slide">
          <article class="post hv-zoom">
            <div class="post-head post-head-23 rounded">
              <a href={file_handle(item.directus_files_id)} class="this-mp-img">
                <img src={file_handle(item.directus_files_id)} alt="image" class="img-fluid post-img rounded" loading="lazy"
                  decoding="async" />
              </a>
            </div>
            <div class="post-body">
              <a href={file_handle(item.directus_files_id)} class="this-mp-img">
                <h2 class="post-title section-description">{item.translations[0]?.caption}</h2>
              </a>
            </div>
          </article>
        </div>
      )}
    </div>
  </div>

```

Gambar 3.60 *Swiper-container gallery.astro*

Data ini dipetakan menggunakan `data_gallery.map(...)` untuk menghasilkan elemen *swiper-slide* bagi setiap gambar dalam galeri. Di dalam setiap elemen *swiper-slide*, terdapat sebuah artikel yang membungkus gambar dan *caption*. Gambar tersebut diambil menggunakan fungsi `file_handle(item.directus_files_id)`, yang mengonversi ID file dari CMS menjadi URL gambar yang dapat ditampilkan. Gambar ini diletakkan di dalam elemen *img* dengan atribut *loading* bernilai *lazy* dan *decoding* bernilai *async* untuk mengoptimalkan pemuatan gambar dengan cara *loading*. *Caption* gambar ditampilkan di bawah gambar menggunakan `item.translations[0]?.caption` (Gambar 3.60).

Navigasi antar slide, kode ini menyediakan dua tombol yang masing-masing terhubung dengan kelas *swiper-prev-gallery* dan *swiper-next-gallery*. Tombol-tombol ini memungkinkan pengguna untuk beralih ke slide sebelumnya atau berikutnya.

```

<script>
// document.addEventListener('astro:page-load', () => {
document.addEventListener('DOMContentLoaded', () => {
  new Swiper('#gallery-swiper', {
    lazy: true,
    loop: false,
    spaceBetween: 16,
    centeredSlides: false,
    initialSlide: 0,
    preloadImages: true,
    autoplay: false,
    navigation: {
      nextEl: '.swiper-next-gallery',
      prevEl: '.swiper-prev-gallery',
    },
    breakpoints: {
      0: {
        slidesPerView: 1.4,
        spaceBetween: 16,
      },
      580: {
        slidesPerView: 2.2,
        spaceBetween: 20,
      },
      768: {
        slidesPerView: 2.7,
      },
      991: {
        slidesPerView: 4,
        spaceBetween: 20,
      },
      1200: {
        slidesPerView: 4,
        spaceBetween: 30,
      },
    },
  });
});
</script>

```

Gambar 3.61 *Swiper gallery.astro*

Swiper terinisialisasi, beberapa opsi seperti bernilai *true* diaktifkan untuk memastikan gambar dimuat hanya ketika diperlukan, sementara *loop* bernilai *false* memastikan galeri tidak berputar terus menerus setelah mencapai slide terakhir. Kode *spaceBetween* bernilai 16 mengatur jarak antar slide, dan *centeredSlides* bernilai *false* memastikan slide tidak selalu berada di tengah. Pengaturan *breakpoints* digunakan untuk menyesuaikan jumlah slide yang terlihat dan jarak antar slide berdasarkan ukuran layar, misalnya pada layar di bawah 580px, galeri akan menampilkan 1.4 slide, sementara pada layar lebih besar akan menampilkan 4 slide (Gambar 3.61).

Dengan pengaturan ini, galeri gambar menjadi responsif dan interaktif, memungkinkan pengguna untuk menavigasi antar gambar dengan tombol navigasi di berbagai perangkat dengan tampilan yang disesuaikan.

6. *Hero.astro* merupakan *section hero* yang menampilkan latar belakang berupa video. Bagian ini dirancang untuk memberikan kesan visual yang kuat dan menarik perhatian pengunjung situs. *Hero.astro* terdiri dari:

Kode `fetch_data` berfungsi mengambil data untuk bagian *hero banner*. Data yang diambil mencakup informasi mengenai kartu (*cards*) dan latar belakang (*backdrop*), baik itu video maupun gambar. Data untuk kartu meliputi URL dan terjemahan terkait, sementara data latar belakang mencakup jenis media seperti video atau gambar, dan sumbernya. Properti *lang* digunakan untuk memfilter data sesuai dengan bahasa yang diteruskan ke komponen. Hasil dari permintaan API disimpan dalam variabel `data_hero`, yang kemudian dapat digunakan untuk merender bagian hero banner di halaman.



```

<section class="intro-section section">
  <div class="swiper cover-slide" id="intro-swiper">
    <div class="swiper-wrapper">
      {data_hero.backdrop.map((item, index) => {
        <div class="swiper-slide">
          {item.type === 'video' ?
            <video muted playsinline autoplay loop class="this-cover"><source src={item.
              video_source === 'Internal'?file_handle(item.video):item.url} type={item.
              video_source === 'Internal'?item.video.type:'video/mp4'}></video>
            :
            <img src={file_handle(item.image)} alt="cover" class="img-fluid this-cover"
              loading="lazy" decoding="async" />
          }
        </div>
      )}
    </div>
  </div>

```

Gambar 3.62 Swiper cover slide hero.astro

Pada kode ini, latar belakang menggunakan *swiper* dengan id *intro-swiper* di bagian ini, `data_hero.backdrop.map(...)` digunakan untuk menampilkan gambar atau video berdasarkan data. Jika tipe `item.type` adalah 'video', maka sebuah elemen `<video>` akan digunakan, dengan sumber video yang diambil dari `item.video` atau `item.url`. Video akan diputar secara otomatis (*autoplay*), tanpa suara (*muted*), dan berulang kali (*loop*). Jika item adalah gambar, maka elemen `<img>` digunakan, dan gambar diambil menggunakan fungsi `file_handle(item.image)` (Gambar 3.62).

Bagian kartu informasi ditampilkan menggunakan *swiper* kedua dengan id *intro-card-swiper*. Kartu-kartu ini berisi informasi yang diambil dari `data_hero.cards`. Setiap kartu menampilkan *subtitle*, *title*, jargon, dan tombol dengan teks yang diambil dari `translations[0]` sesuai dengan bahasa yang dipilih. URL untuk setiap kartu diambil dari `item.url`, dan teks tombol diambil dari `item.translations[0].link_text`.

```
document.addEventListener('DOMContentLoaded', () => {
  new Swiper('#intro-swiper', {
    lazy: true,
    loop: false,
    spaceBetween: 0,
    slidesPerView: 1,
    centeredSlides: false,
    initialSlide: 0,
    preloadImages: true,
    autoplay: {
      delay: 10000,
    },
    effect: 'fade',
    fadeEffect: {
      crossFade: true
    },
  });
});
```

Gambar 3.63 New swiper hero.astro

Swiper dengan id *intro-swiper* digunakan untuk latar belakang gambar atau video. Pengaturan *swiper* untuk latar belakang ini menggunakan *autoplay* dengan *delay* 10 detik untuk berpindah antar gambar atau video. Efek transisi yang digunakan adalah *fade*, di mana gambar atau video akan berganti dengan efek saling memudar (*crossFade*) (Gambar 3.63).

```
new Swiper('#intro-card-swiper',
  lazy: true,
  loop: false,
  spaceBetween: 16,
  centeredSlides: false,
  initialSlide: 0,
  preloadImages: true,
  autoplay: false,
  breakpoints: {
    0: {
      slidesPerView: 1.1,
      spaceBetween: 16,
    },
    768: {
      slidesPerView: 1.5,
    },
    992: {
      slidesPerView: 2.1,
    },
    1200: {
      slidesPerView: 2.3,
    },
    1400: {
      slidesPerView: 2.5,
    },
  },
});
```

Gambar 3.64 Intro-card hero.astro

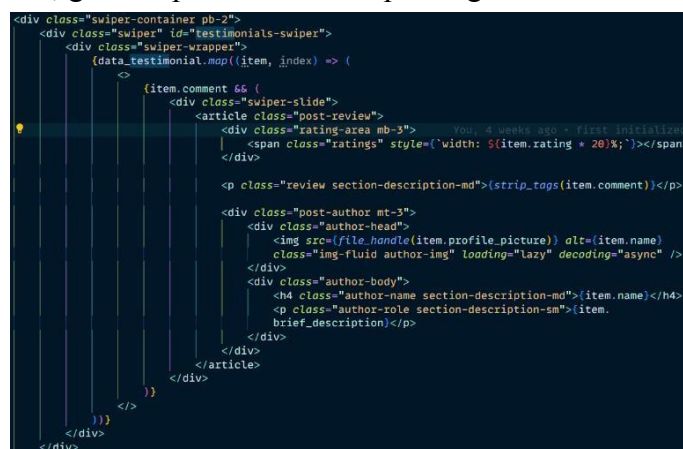
Pada *swiper* id *intro-card-swiper*, *spacebetween* diatur ke 16px, yang berarti ada jarak antar slide. *Breakpoints* memungkinkan jumlah slide yang ditampilkan berubah berdasarkan ukuran layar perangkat. Misalnya, pada layar dengan lebar 0-768px, hanya 1.1 slide yang ditampilkan, sedangkan pada layar lebih besar, akan ada lebih banyak slide yang ditampilkan (Gambar 3.64).

Secara keseluruhan, kode ini menggabungkan elemen gambar atau video yang dinamis dengan kartu informasi interaktif, keduanya dikendalikan oleh *swiper* untuk pengalaman pengguna yang menarik dan responsif.

7. *Testimonials.astro* adalah sebuah *section* yang dirancang untuk menampilkan komentar-komentar dari pengunjung yang telah mengunjungi destinasi dan acara yang ada di radyakartiyasa. *Section* ini tidak hanya menampilkan komentar secara tekstual, tetapi juga mencakup sistem *rating* bintang yang menggambarkan tingkat kepuasan pengunjung terhadap pengalaman mereka. *Testimonials.astro* terdiri dari:

Kode `strip_tags` digunakan untuk menghapus tag HTML dari teks, memastikan hanya teks biasa yang ditampilkan, khususnya untuk komentar yang mungkin mengandung HTML. Kode `file_handle` digunakan untuk menghasilkan URL yang dapat diakses dari file gambar yang ada di CMS, seperti untuk gambar profil.

Kode `data_testimonial` adalah data yang diteruskan ke komponen ini melalui properti *Astro.props*, yang berisi array dari testimonial. Setiap testimonial mencakup informasi seperti komentar, *rating*, nama, gambar profil dan deskripsi singkat.



```

<div class="swiper-container pb-2">
  <div class="swiper" id="testimonials-swiper">
    <div class="swiper-wrapper">
      {data_testimonial.map((item, index) => (
        <div class="swiper-slide">
          <article class="post-review">
            <div class="rating-area mb-3">
              <span class="ratings" style="width: {item.rating * 20}%;></span>
            </div>
            <p class="review section-description-md">{strip_tags(item.comment)}</p>
            <div class="post-author mt-3">
              <div class="author-head">
                <img src={file_handle(item.profile_picture)} alt={item.name}
                  class="img-fluid author-img" loading="lazy" decoding="async" />
              </div>
              <div class="author-body">
                <h4 class="author-name section-description-md">{item.name}</h4>
                <p class="author-role section-description-sm">{item.brief_description}</p>
              </div>
            </div>
          </article>
        </div>
      )})
    </div>
  </div>

```

Gambar 3.65 *Swiper wrapper testimonials.astro*

Kode ini digunakan untuk membuat *slider* testimonial interaktif menggunakan *swiper*. Data testimonial diambil dan dirender dalam elemen *swiper-slide*. Setiap testimonial terdiri dari beberapa bagian: pertama, *rating* ditampilkan dengan menggunakan elemen `<span class="ratings">`, di mana lebar dari elemen ini dihitung berdasarkan nilai *rating* yang diberikan, yakni $\text{item.rating} \times 20$ (karena *rating*nya dalam skala 1-5, dan *swiper* mengharapkan persentase antara 0-100%). Komentar kemudian ditampilkan menggunakan elemen `<p>` dengan fungsi `strip_tags(item.comment)` yang memastikan hanya teks yang bersih tanpa tag HTML.

yang ditampilkan. Selain itu, informasi profil pengguna (foto, nama, dan deskripsi) juga ditampilkan. Foto profil diambil dengan menggunakan `file_handle(item.profile_picture)` untuk menghasilkan URL gambar yang benar. Nama pengguna ditampilkan dengan `item.name`, dan deskripsi singkat menggunakan `item.brief_description`. Dalam bagian `{data_testimonial.map(...)}` digunakan untuk melakukan iterasi pada array `data_testimonial`, yang berisi semua testimonial, untuk setiap item, jika ada komentar (`item.comment`), maka testimonial tersebut akan dirender. Foto profil ditampilkan dengan elemen `<img>` dan menggunakan `file_handle(item.profile_picture)` untuk menghasilkan URL gambar yang benar (Gambar 3.65).

```
<div class="pagination-area">
  <div class="swiper-pagination swiper-pagination-bullet-2 offset-bottom
    swiper-pagination-testimonials"></div>
</div>
```

Gambar 3.66 *Pagination-area testimonials.astro*

Kode *pagination-area* menunjukkan kontrol *pagination* berbentuk *bullets* di bawah slider *swiper*. Kode *swiper-pagination* adalah kelas yang digunakan *swiper* untuk membuat kontrol *pagination*. Kode *swiper-pagination-bullet-2* mengatur tampilan *bullets*, seperti ukuran atau gaya visual. Kode *offset-bottom* memberikan jarak antara *pagination* dan elemen lain di halaman. Kode *swiper-pagination-testimonials* adalah kelas kustom yang memungkinkan

pengaturan khusus untuk *pagination* di *slider* testimonial (Gambar 3.66).

```
<script>
// document.addEventListener('astro:page-load', () =>
document.addEventListener('DOMContentLoaded', () => {
  new Swiper('#testimonials-swiper', {
    lazy: true,
    loop: false,
    spaceBetween: 16,
    centeredSlides: false,
    initialSlide: 0,
    preloadImages: true,
    autoplay: false,
    pagination: {
      el: '.swiper-pagination-testimonials',
      clickable: true,
    },
    breakpoints: {
      0: {
        slidesPerView: 1.2,
        spaceBetween: 16,
      },
      580: {
        slidesPerView: 1.8,
        spaceBetween: 20,
      },
      768: {
        slidesPerView: 2.3,
      },
      991: {
        slidesPerView: 2.6,
        spaceBetween: 20,
      },
      1200: {
        slidesPerView: 3,
        spaceBetween: 30,
      },
    },
  });
});
</script>
```

Gambar 3.67 *Swiper testimonials.astro*

Kode *lazy* bernilai *true* mengaktifkan *loading* untuk gambar, memastikan gambar hanya dimuat saat dibutuhkan (saat testimonial muncul di layar). Kode *loop* bernilai *false* agar mengatur agar *slider* tidak berulang setelah mencapai testimonial terakhir. Fungsi *pagination* menambahkan elemen navigasi (*pagination*) di bawah *slider* yang memungkinkan pengguna untuk berpindah antar testimonial menggunakan *bullet points*. Kode *breakpoints* menentukan responsivitas *slider*. Sebagai contoh, pada layar dengan lebar 0-580px, *swiper* akan menampilkan 1.2 slide, sedangkan pada layar lebih besar, jumlah *slide* yang ditampilkan akan bertambah (Gambar 3.67).

Dengan pengaturan ini, *slider* testimonial dapat menampilkan komentar dari berbagai individu dalam format yang responsif dan dinamis pada berbagai perangkat.

c. Making element components

Folder *element* berisi file *Back to Top.astro* yang muncul ketika laman di *scroll* kebawah melewati batas yang telah ditentukan maka akan ada ikon “back- to-top” yang memungkinkan user untuk kembali ke bagian atas halaman dengan klik. File *Section Title.astro* untuk menampilkan judul bagian di halaman web dengan elemen-elemen opsional seperti *subtitle*,

deskripsi, dan tautan. Berikut adalah penjelasan lebih detail dengan menyertakan kode dari file-file tersebut:

1. *Backtotop.astro* digunakan ketika user menggulir halaman lebih dari 500px ke bawah dan memungkinkan pengguna untuk kembali ke bagian atas halaman dengan klik. *Backtotop.astro* terdiri dari:

Tombol “Back to Top” diwakili oleh elemen `<button>` dengan id *back-to-top* dan kelas *back-to-top* yang mengatur penampilan tombol tersebut.

Kode *event listener* untuk menjalankan fungsi setelah seluruh halaman selesai dimuat (*DOMContentLoaded*). Variabel *t* mengambil nilai id dari elemen *button*, yang nantinya dapat digunakan untuk manipulasi lebih lanjut pada elemen tersebut.

```

window.addEventListener('scroll', () => {
  if (window.scrollY > 500) {
    t.classList.add('show');
  } else {
    t.classList.remove('show');
  }
});

```

Gambar 3.68 Scroll *backtotop.astro*

User berada di posisi *scroll* halaman lebih dari 500px (dengan *window.scrollY > 500*), maka kelas *show* dihapus, dan tombol menjadi tersembunyi (Gambar 3.68).

```

t.addEventListener('click', () => {
  window.scrollTo({ top: 0, behavior: 'smooth', });
});

```

Gambar 3.69 Scrollto *backtotop.astro*

Selanjutnya, ketika tombol diklik, halaman akan digulirkan kembali ke atas dengan menggunakan *window.scrollTo* dengan nilai *top* yaitu 0 memastikan halaman digulirkan ke bagian atas, sementara nilai *behaviour* yaitu *smooth* memberikan efek transisi halus saat menggulir. Secara keseluruhan, kode ini menambahkan fitur navigasi yang nyaman bagi pengguna dengan menyediakan tombol “Back to Top” yang hanya muncul setelah menggulir halaman ke bawah, memungkinkan pengguna untuk kembali ke atas halaman dengan klik secara halus (Gambar 3.69).

2. *SectionTitle.astro* adalah komponen yang digunakan untuk menampilkan judul bagian di

halaman web, lengkap dengan elemen- elemen opsional seperti *subtitle*, deskripsi, dan tautan. *SectionTitle.astro*:

```
const { subtitle, title, description, linkText, linkUrl, class: additionalClass } = Astro.props;
```

Gambar 3. 70 Props *sectiontitle.astro*

Komponen *SectionTitle* menerima beberapa properti, yaitu *subtitle* digunakan untuk menampilkan teks tambahan atau label di bawah judul, sementara *title* adalah judul utama dari bagian tersebut. Jika diberikan, properti *description* akan menampilkan penjelasan lebih lanjut sedangkan *linkText* dan *linkUrl* digunakan untuk menampilkan tautan ke halaman lain. Properti *class* memungkinkan penambahan kelas tambahan untuk menyesuaikan gaya komponen (Gambar 3. 70).

```
<div class={`section-title ${additionalClass} || ''`} >
```

Gambar 3.71 *Section-title sectiontitle.astro*

Komponen ini dimulai dengan mendeklarasikan elemen div dengan kelas *section-title* yang dapat ditambahkan kelas tambahan sesuai dengan properti *class* yang diberikan (Gambar 3.71).

```
{subtitle && (
  <p class="subtitle badge badge-sm badge-cs section-description-sm">{subtitle}</p>
)}
```

Gambar 3.72 *Badge sectiontitle.astro*

Jika properti *subtitle* diberikan, maka akan menampilkan elemen paragraf dengan kelas *section-description-sm* (Gambar 3.72).

```
{title && (
  <div class="title-area">
    <svg xmlns="http://www.w3.org/2000/svg" width="50" height="46" viewBox="0 0 50 46" fill="none">
      <path fill-rule="evenodd" clip-rule="evenodd" d="M25.1393 0.0981924C25.1902 0.174959 25.169 0.
      278332 25.0922 0.329082C24.3617 0.81101 24.0823 1.24418 24.1091 1.62811C24.3592 4.8628 24.
      5303 6.02233 24.674 6.9954C24.7343 7.40479 24.7899 7.78118 24.8444 8.26523C24.8547 8.35668 24.
      7888 8.43914 24.6972 8.44943C24.6055 8.4597 24.5228 8.39389 24.5126 8.30244C24.4594 7.83036
      24.4046 7.45865 24.3447 7.05214C24.2008 6.07628 24.0272 4.89991 23.7761 1.65317V1.65207C23.
      7353 1.0734 24.1609 0.544012 24.9079 0.0510839C24.9849 0.000334179 25.0884 0.0214254 25.1393
      0.0981924" fill="#0E3121"/>
    </svg>
    <h2 class="title section-heading">{title}</h2>
  </div>
)}
```

Gambar 3.73 *Section-heading sectiontitle.astro*

Jika properti *title* diberikan, maka menampilkan judul utama dalam sebuah elemen div dengan tambahan elemen SVG untuk dekorasi (Gambar 3.73).

```

    {description && (
      |
      <p class="description section-description-md">{description}</p>
      |
    )}
  </div>

  <div class="st-right">
    {linkText && (
      |
      <a href={linkUrl} class="link link-arrow accent section-description-md">{linkText}</a>
      |
    )}
  </div>
</div>

```

Gambar 3.74 Deskripsi & tautan *sectiontitle.astro*

Kode ini menampilkan deskripsi dan tautan jika properti yang relevan diberikan. Bagian pertama memeriksa *description* dan menampilkan teks dalam elemen paragraf dengan kelas khusus. Bagian kedua memeriksa *linkText* dan *linkUrl*, kemudian menampilkan tautan dengan kelas CSS untuk penataan dan efek visual, seperti ikon panah dan warna aksen. Tautan akan mengarah ke URL yang ditentukan oleh *linkUrl* (Gambar 3.74).

Dengan menggunakan komponen *SectionTitle* memberikan fleksibilitas untuk menampilkan judul bagian dengan elemen-elemen opsional seperti *subtitle*, deskripsi, dan tautan, yang dapat disesuaikan dengan kebutuhan desain halaman web.

3.2.5 Styles For Styling The Content Website

Pada folder *styles* terdapat file *global.css* yang diimpor ke *Base.astro* dan berfungsi sebagai stylesheet utama yang mengatur tampilan visual keseluruhan situs web. File ini mengatur berbagai aspek desain, mulai dari tipografi, layout responsif, *padding*, margin, warna, hingga animasi transisi dan komponen UI khusus. Berikut ini adalah penjelasan ringkas beberapa kelompok aturan penting yang terdapat dalam file tersebut.

1. Reset dan Normalisasi

Bagian awal file melakukan reset pada elemen-elemen dasar HTML seperti ***, *html*, dan *body* agar perilaku tampilan konsisten di berbagai browser seperti menggunakan *padding* dan *margin* bernilai 0, serta *box-sizing* bernilai *border-box* memastikan semua elemen menggunakan model box yang seragam.

2. Pengaturan Tipografi

File ini mengatur font-family standar seperti *Tenor Sans*, *Anek Bangla*, dan *Inter*, yang sudah diimpor melalui *Google fonts* di file *Base.atro*. Ada juga pengaturan ukuran font dan tinggi baris untuk berbagai heading (*h1* hingga *h6*) dan paragraf.

3. Utilitas Layout dan Responsivitas

CSS ini memanfaatkan utilitas berbasis grid dan flex yang bertujuan untuk memastikan tata letak halaman tetap responsif di berbagai ukuran layar.

4. Komponen Navigasi dan Footer

Kelas seperti *.header*, *.footer*, *.main-navigation*, dan *.dropdown-menu* digunakan untuk mendesain tampilan *header* dan *footer* situs. Kode ini mendukung elemen seperti menu *horizontal*, *dropdown*, bahasa dan ikon sosial media.

5. Desain Accordion, Swiper dan Tombol

Elemen interaktif seperti FAQ *accordion* dan slider galeri (*Swiper*) memiliki gaya khusus agar tampilan lebih menarik dan responsif. Misalnya *.accordion-head* diberi gaya *border* dan *padding*, serta efek *collapsed* ditangani secara visual.

6. Tombol dan Link

Berbagai kelas seperti *.btn*, *.link*, *.section-description-md*, dan *.btn-hover-icon-right* mengatur tampilan dan transisi tombol serta teks link. File ini juga memanfaatkan efek *hover* dan animasi ikon agar lebih interaktif saat diklik atau disentuh user.

7. Animasi dan Efek

Transisi Kode CSS mencakup efek transisi untuk komponen seperti *.slide-up*, *.slide-down*, atau *.shrink* pada *header*, agar terasa dinamis saat user menggulir halaman.

Secara keseluruhan file *global.css* merupakan pondasi visual dari situs web. Dengan mencakup pengaturan layout, tipografi, warna, dan interaksi UI seperti tombol, navigasi, galeri, dan responsivitas, file ini memastikan tampilan situs tetap konsisten, estetik, dan ramah terhadap user di berbagai perangkat dan ukuran layar.

3.2.6 Script To Make It Interactive Website

Di dalam folder *script* terdapat beberapa file untuk meningkatkan interaktivitas dan pengalaman pengguna di situs web. File *main.js* mengelola elemen-elemen interaktif seperti dropdown kustom dengan *Select2*, navigasi *mobile*, dan integrasi *Magnific Popup* untuk tampilan gambar dengan efek zoom, serta fitur berbagi ke media sosial. File *pagination.min.js* menangani paginasi konten dengan tombol “Load More” untuk memuat lebih banyak konten

secara dinamis. Sedangkan File *plugins.js* mengelola *scroll* halus, memberikan efek *scroll* yang lebih fluid dan mendukung navigasi menggunakan mouse atau keyboard. Ketiga file ini bekerja bersama untuk menciptakan situs yang responsif, dinamis, dan memberikan pengalaman user yang lebih baik.

1. File *main.js* mengelola interaksi UI seperti dropdown, navigasi seluler, dan berbagi media sosial. Kode *main.js* terdiri dari:

```
if (document.querySelectorAll('.select-custome').length > 0) {
  document.querySelectorAll('.select-custome').forEach(function(element) {
    $(element).select2({
      minimumResultsForSearch: -1
    });
  });
}
```

Gambar 3.75 Tampilan *dropdown main.js*

Kode ini terutama *Select2* digunakan untuk meningkatkan tampilan dropdown. Kode ini mencari elemen dengan kelas *.select-custome* dan menginisialisasi *Select2* untuk elemen-elemen tersebut, dengan menonaktifkan fitur pencarian di dropdown (*minimumResultsForSearch: -1*) (Gambar 3.75).

```
(function ($) {
  $('.header-open-navigation').on('click', function () {
    var b = $('body');
    var t = $('.header .header-nav')

    if (t.hasClass('shown')) {
      t.removeClass('shown');
      b.addClass('overflow-hidden')
    } else {
      t.addClass('shown');
      b.removeClass('overflow-hidden')
    }
  });

  $('.header-close-navigation').on('click', function () {
    var t = $('.header .header-nav')

    if (t.hasClass('shown')) {
      t.removeClass('shown');
    } else {
      t.addClass('shown');
    }
  });
});
```

Gambar 3.76 Menampilkan & menyembunyikan *main.js*

Kode ini menampilkan atau menyembunyikan menu navigasi samping saat tombol *header-open-navigation* atau *header-close-navigation* diklik. Menu akan dibuka dengan menambahkan kelas *shown* pada elemen navigasi dan mengunci *scroll* halaman (*overflow-hidden* pada *body*) (Gambar 3.76).

```

// magnific popup
if ($.this-mp-img).length {
  $.this-mp-img.magnificPopup({
    type: 'image',
    preloader: true,
    closeOnContentClick: true,
    closeBtnInside: true,
    fixedContentPos: true,
    mainClass: 'mfp-no-margins mfp-with-zoom',
    gallery: {
      enabled: true,
      arrowMarkup: '<button title="%title%" type="button"
class="mfp-arrow mfp-arrow-%dir%"></button>',
      tPrev: 'Previous',
      tNext: 'Next',
      // tCounter: '<span class="mfp-counter">%curr% of
      %total%</span>'
    },
    image: {
      verticalFit: true,
      titleSrc: function(item) {
        return item.el.attr('data-title');
      }
    },
    zoom: {
      enabled: true,
      duration: 600
    }
  });
}

```

Gambar 3.77 *Magnific popup main.js*

Kode ini mengaktifkan *plugin Magnific Popup* elemen dengan kelas `.this-mp-img`, yang memungkinkan gambar dibuka dalam modal dengan efek zoom dan navigasi galeri (previous/next) (Gambar 3.77).

```
// ----- share social media

$(document).on('click', '.post-share-facebook', function() {
    var t = $(this).attr('data_title');
    var u = $(this).attr('data_url');

    window.open('http://www.facebook.com/sharer.php?u='+encodeURIComponent(u)+'&t='+
    +encodeURIComponent(t), 'sharer', 'toolbar=0,status=0,width=626,height=436');return
    false;
});

$(document).on('click', '.post-share-twitter', function() {
    var t = $(this).attr('data_title');
    var u = $(this).attr('data_url');

    window.open('https://twitter.com/intent/tweet?url='+encodeURIComponent(u)+'&text='
    +encodeURIComponent(t), 'sharer', 'toolbar=0,status=0,width=626,height=436');return
    false;
});

$(document).on('click', '.post-share-linkedin', function() {
    var u = $(this).attr('data_url');

    window.open('https://www.linkedin.com/shareArticle?mini=true&url='
    +encodeURIComponent(u), 'sharer', 'toolbar=0,status=0,width=626,height=436');return
    false;
});

$(document).on('click', '.post-share-whatsapp', function() {
    var t = $(this).attr('data_title');
    var u = $(this).attr('data_url');

    if ($(window).width() > 960) {
        window.open('https://web.whatsapp.com/send?text='+encodeURIComponent(u)+' - '
        +encodeURIComponent(t)+'?utm_source=whatsapp&utm_medium=sharer&
        utm_campaign=social');return false;
    } else {
        window.open('whatsapp://send?text='+encodeURIComponent(u)+' - '
        +encodeURIComponent(t)+'?utm_source=whatsapp&utm_medium=sharer&
        utm_campaign=social');return false;
    }
});

$(document).on('click', '.post-share-telegram', function() {
    var t = $(this).attr('data_title');
    var u = $(this).attr('data_url');

    window.open('https://t.me/share/url?url='+encodeURIComponent(u)+'&text='
    +encodeURIComponent(t), 'sharer', 'toolbar=0,status=0,width=626,height=436');return
    false;
});

$(document).on('click', '.post-share-googleplus', function() {
    var u = $(this).attr('data_url');

    window.open('https://plus.google.com/share?url='+encodeURIComponent(u), 'sharer',
    'toolbar=0,status=0,width=626,height=436');return false;
});

// ----- end share social media

})(jQuery);
```

Gambar 3.78 Berbagi *main.js*

Kode ini menangani fitur berbagi ke berbagai platform media sosial. Setiap elemen dengan kelas *.post-share-** berfungsi sebagai tombol berbagi, yang membuka jendela baru untuk membagikan konten yang ditentukan. Misalnya, untuk *facebook*, tombol berbagi membuka jendela untuk membagikan URL dan judul menggunakan <http://www.facebook.com/sharer.php>. Begitu juga dengan *twitter*, di mana tautan artikel dan teks dibagikan melalui URL <https://twitter.com/intent/tweet>. Untuk *whatsapp*, kode ini membedakan antara tampilan *desktop* dan *mobile*, jika layar lebih besar dari 960px, berbagi dilakukan melalui *web.whatsapp.com*, sementara untuk perangkat *mobile*, berbagi dilakukan melalui aplikasi *whatsapp* langsung. Kode ini juga mendukung berbagi ke *linkedin*, *telegram*, dan *Google plus* (Gambar 3.78).

Secara keseluruhan file *main.js* mengelola berbagai fitur interaktif di halaman web, termasuk dropdown kustom dengan *Select2*, *magnific popup* untuk galeri gambar, serta tombol

berbagi media sosial untuk platform sosial media. Skrip ini juga menangani navigasi *mobile*, memungkinkan pengguna membuka dan menutup menu samping dengan animasi, dan meningkatkan pengalaman pengguna di perangkat *mobile* maupun *desktop*.

2. File *pagination.min.js* berfungsi untuk mengatur sistem paginasi dinamis pada halaman web dengan model “Lihat lebih banyak”. Kode *pagination.min.js* terdiri dari:

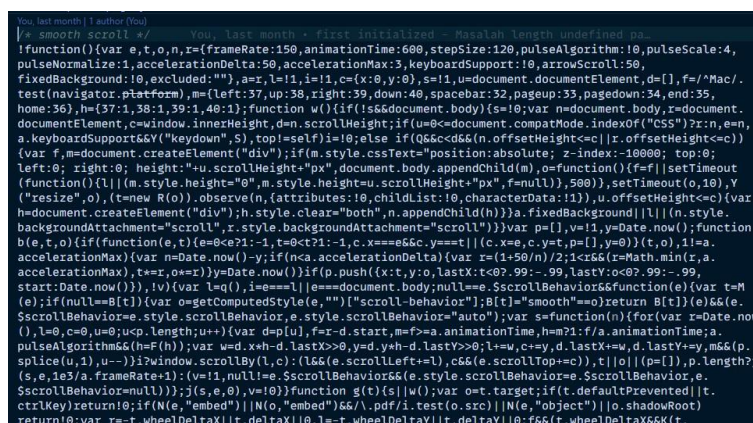
```
export function pagination_handle(a=!1,i=!1){let n=a||9,o=i||"#pagination-area",e=1,t=$(o+" > *").length,l=Math.ceil(t/n);function d(a){$(o).after('<div class="load-more-area" id="pagination-load-more"><a href="javascript:void(0)" class="btn">Lihat lebih banyak</a></div>');let i=(a-1)*n,e=a*n;$(o+" > *").each(function(a){$(this).addClass("pagination-item-"+a),a=i&&a<e&&$(o+" .pagination-item-"+a).show()),a>l&&$(".load-more-area").remove();$(o+" > *").hide(),d(e),$( "#pagination-load-more > a").on("click",function(){e<l&&d(++e)})})}
```

Gambar 3.79 *Pagination_handle pagination.min.js*

Fungsi utamanya , yaitu *pagination_handle(a,i)*, menerima dua parameter opsional: *a* untuk menentukan jumlah item yang ditampilkan per halaman (*default* 9), dan *i* sebagai selector dari elemen kontainer yang ingin dipaginasi (*default* *#pagination-area*). Di dalam fungsi, nilai-nilai seperti jumlah total item (*t*), jumlah halaman (*l*), dan halaman aktif (*e*) dihitung terlebih dahulu. Selanjutnya, fungsi *d(a)* digunakan untuk menampilkan item yang sesuai dengan halaman aktif. Fungsi ini menambahkan tombol “Lihat lebih banyak” setelah elemen kontainer, memberi setiap item kelas khusus berdasarkan urutan, lalu hanya menampilkan item yang berada di antara indeks yang sesuai. Bila halaman terakhir sudah tercapai, tombol tersebut akan dihapus dari DOM. Saat fungsi pertama kali dijalankan, semua item dalam kontainer disembunyikan dengan *\$(o + “ > *”).hide()*, lalu hanya halaman pertama yang ditampilkan. Setelah itu, *event listener* dipasang pada tombol *#pagination-load-more > a* agar saat diklik, halaman berikutnya ditampilkan dengan *d(++e)*. Dengan pendekatan ini, pengguna tidak perlu memuat ulang halaman untuk melihat lebih banyak konten. Penggunaan metode ini sangat berguna dalam konteks tampilan konten panjang, seperti daftar artikel, galeri gambar, atau produk, sehingga meningkatkan kenyamanan pengguna dan efisiensi tampilan. Secara keseluruhan, file *pagination.min.js* menyediakan mekanisme paginasi dinamis berbasis tombol “Lihat lebih banyak” yang memungkinkan konten ditampilkan secara bertahap tanpa memuat ulang halaman. Dengan, membatasi jumlah item yang ditampilkan per halaman dan menambahkan

tombol interaktif untuk memuat konten berikutnya, fungsi ini membantu menjaga tampilan tetap ringkas dan responsif. Pendekatan ini tidak hanya meningkatkan efisiensi pemuatan halaman, tetapi juga memberikan pengalaman pengguna yang lebih nyaman dan terkontrol saat menjelajahi daftar konten yang panjang (Gambar 3.79).

3. File *plugins.js* adalah sebuah *plugin javascript* yang berfungsi untuk menerapkan *smooth scrolling* atau efek gulir halus pada halaman web, menggantikan perilaku *scroll* bawaan browser yang biasanya bersifat langsung dan *abrupt*. Kode *plugins.js* terdiri dari:



```

You: last month | 1 author (You)
// smooth scroll s/
function (var e, t, o, n, r={frameRate:150, animationTime:600, stepSize:120, pulseAlgorithm:10, pulseScale:4,
pulseNormalize:1, accelerationDelta:50, accelerationMax:3, keyboardSupport:10, arrowScroll:50,
fixedBackground:10, excluded:""}, a=r, l=1, i=1, c={x:0, y:0}, s=1, u=document.documentElement, d=[], f="/Mac/
test(navigator.platform), m={left:37, up:38, right:39, down:40, spacebar:32, pageup:33, pagedown:34, end:35,
home:36}, h={37:1, 38:1, 39:1, 40:1}, function w() {if (!s&&document.body) {s=!0; var n=document.body, r=document.
documentElement, c=window.innerHeight, d=n.scrollHeight; if (u&&c<document.compatMode.indexOf("CSS")) {r:n, e=n,
a.keyboardSupport&&v("keydown", S), top!=self) i=!0; else if (Q&&c<d&&(n.offsetHeight<c || r.offsetHeight<c))
(var f, m=document.createElement("div")); if (m.style.cssText="position:absolute; z-index:-10000; top:0;
left:0; right:0; height:"+u.scrollHeight+"px", document.body.appendChild(m), o=function() {if (f||setInterval
(function() {l||(m.style.height="0", m.style.height=u.scrollHeight+"px", f=null), 500}), setTimeout(o, 10), Y
("resize", o), (t=new R(o)).observe(n, {attributes:!0, childList:!0, characterData:!1}), u.offsetHeight<c) (var
h=document.createElement("div")); h.style.clear="both", n.appendChild(h)) a.fixedBackground||l||(n.style.
backgroundAttachment="scroll", r.style.backgroundAttachment="scroll")); var p=[], v=1, y=Date.now(); function
b(e, t, o) {if (function(e, t) {e=e&e?1:-1, t=t&t?1:-1, c.x==m&&c.y==t} || (c.x=e, c.y=t, p=[], y=0)) (t, o), l!=a.
accelerationMax) (var n=Date.now(), y;if (n-a.accelerationDelta) (var r=(1+50/n)/2, a=c&&(r=Math.min(r, a.
accelerationMax), t+=r, o+=r)) y=Date.now(); if (p.push({x:t, y:o, lastX:t<0?-.99:-.99, lastY:o<0?-.99:-.99,
start:Date.now()}), l) (var l=n(), i=e==l || e==document.body; null==e.$scrollBehavior&&function(e) {var t=M
(e); if (null==B[t]) (var o=getComputedStyle(e, "")) ["scroll-behavior"]; B[t]="smooth"==o?return B[t]:(e&&(e.
$scrollBehavior=e.style.scrollBehavior, e.style.scrollBehavior="auto"); var s=function(n) {for (var r=Date.now
(), l=0, c=0, u=0; u<p.length; u++) {var d=p[u], f=r-d.start, m=f>=a.animationTime, h=m?1:f/a.animationTime; a.
pulseAlgorithm&&(h=f(h)); var w=d.x-h-d.lastX>0?0:y=d.y-h-d.lastY>0?0:1+w, c+=y, d.lastX=w, d.lastY=y, m&&(p.
splice(u, 1, u--)); if (window.scrollBy(L, c): (l&&(e.scrollLeft+=l, c&&(e.scrollTop+=c)), t||o) (p=[], p.length?j
(s, e, 1e3/a.frameRate+1): (v=!1, null!=e.$scrollBehavior&&(e.style.scrollBehavior=e.$scrollBehavior, e.
$scrollBehavior=null)); j(s, e, 0, v=!0)} function g(t) {s||w(); var o=t.target; if (t.defaultPrevented || t.
ctrlKey) return !0; if (N(e, "embed")) || N(o, "pdf") || test(o, src) || N(e, "object") || o.shadowRoot)
return !0; var z=-t.wheelDeltaX||t.deltaX||0, l=-t.wheelDeltaY||t.deltaY||0; f&&(t.wheelDeltaX&&k(t.

```

Gambar 3.80 *Plugin plugins.js*

Di awal, *plugin* ini menetapkan konfigurasi dasar dalam variabel *r*, yang mencakup pengaturan seperti *frameRate* (kecepatan frame), *animationTime* (durasi animasi), *stepSize* (jarak setiap langkah *scroll*), hingga dukungan untuk *scroll* melalui keyboard. Fungsi utama untuk menangani efek *scroll* adalah *b(e, t, o)*, yang menghitung dan menjalankan animasi *scroll* berdasarkan arah dan jarak guliran. Fungsi ini menggunakan *requestAnimationFrame* untuk menghasilkan animasi yang halus dan terkontrol. *Plugin* ini juga menangani berbagai jenis input *scroll* dari pengguna, seperti melalui *mouse wheel* (dengan fungsi *g(event)*) dan keyboard (dengan fungsi *S(event)*). Setiap *event* tersebut diikat melalui pemanggilan *Y("keydown", S)* untuk keyboard dan *Y(oe, g, te)* untuk *scroll mouse*. Selain itu, sistem ini memiliki logika untuk menentukan elemen mana yang menjadi target *scroll*, memastikan efek *smooth* hanya diterapkan pada elemen yang relevan. *Plugin* ini juga dilengkapi deteksi browser dan sistem operasi, seperti *chrome*, *safari*, serta perangkat *mobile*, untuk memastikan hanya diaktifkan pada lingkungan yang sesuai. Pada bagian akhir, diekspor agar dapat digunakan di berbagai

jenis proyek *javascript*, baik berbasis *CommonJS*, maupun sebagai variabel global *window.SmoothScroll*. Dengan semua fungsi tersebut, *plugins.js* memberikan pengalaman *scroll* yang lebih nyaman, responsif, dan estetik, terutama pada halaman dengan konten panjang atau banyak navigasi (Gambar 3.80).

3.2.7 The Contents to Show to Users

a. File *middleware.js* yang terdapat fungsi *onRequest* yang diekspor dan digunakan untuk menangani permintaan (request) sebelum halaman atau rute diproses seperti ini:



```

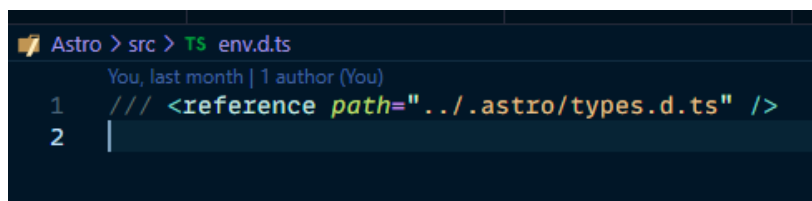
Astro > src > JS middleware.js > onRequest
You, last month | 1 author (You)
1 export const onRequest = async (context, next) => {
2   return next();
3 }
You, last month • first initialized - Masalah length

```

Gambar 3.81 Menerima dua parameter *middleware.js*

Fungsi ini menerima dua parameter, yaitu *context* yang berisi informasi tentang permintaan (seperti URL dan *header*), serta *next function* yang berfungsi untuk meneruskan proses ke *middleware* atau handler berikutnya. Dalam implementasinya saat ini, *middleware* hanya menjalankan dan mengembalikan *next function* tanpa melakukan modifikasi apa pun terhadap permintaan. Artinya, file ini berfungsi sebagai kerangka awal yang masih kosong dan akan dikembangkan lebih lanjut (Gambar 3.81).

b. File *env.d.ts* adalah file deklarasi *typescript*(.d.ts) yang berfungsi untuk memberikan informasi tipe kepada *typescript* dan kode editor tentang lingkungan *astro*. Isi kode dari file tersebut adalah:



```

Astro > src > TS env.d.ts
You, last month | 1 author (You)
1 /// <reference path="../../astro/types.d.ts" />
2

```

Gambar 3.82 Reference path *env.d.ts*

Variabel *reference path* bertujuan untuk mengimpor deklarasi tipe dari file internal *astro* ke dalam proyek *typescript*, sehingga fitur seperti *Astro.props*, *Astro.url*, dan elemen spesifik lainnya dikenali oleh *compiler* dan editor (Gambar 3.82).

c. Folder *pages* berfungsi sebagai tempat untuk mendefinisikan rute

halaman berdasarkan file *.astro*. Setiap file di dalam folder ini akan menjadi halaman yang diakses melalui URL yang sesuai:

1. File *404.astro* yang akan menampilkan halaman error ketika pengguna mengakses rute yang ditemukan. Kode di dalam *404.astro*:



```

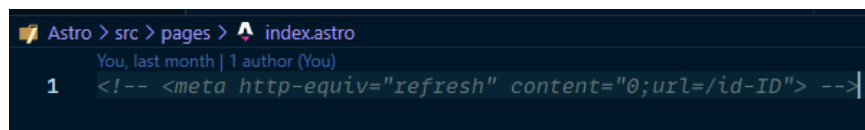
Astro > src > pages > 404.astro > h1
You, last month | 1 author (You)
1 <!-- <meta http-equiv="refresh" content="0;url=/id-ID" -->
2 <h1>404 Not Found</h1>

```

Gambar 3.83 *Not error 404.astro*

Gambar 3.83 menunjukkan kode yang pesan 404 ketika halaman yang diminta tidak ada.

2. File *index.astro* pada *root* folder *pages* adalah kode yang memberi instruksi Ke browser untuk mengalihkan user ke halaman lain dengan *url /id-ID* segera setelah halaman dimuat.



```

Astro > src > pages > index.astro
You, last month | 1 author (You)
1 <!-- <meta http-equiv="refresh" content="0;url=/id-ID" -->

```

Gambar 3.84 Komen *meta index.astro*

Kode *content="0;url=/id-ID"* menunjukkan waktu penundaan pengalihan yang diatur ke 0 detik, artinya pengalihan akan dilakukan langsung setelah halaman dimuat. URL tujuan pengalihan adalah */id-ID*. Sekarang dikomen karena ada cara lain untuk pengaturan routing yang lebih fleksibel (Gambar 3.84).

3. Folder *[lang]* terdapat berbagai file yang masing masing berfungsi untuk menangani rute dinamis berdasarkan bahasa yang dipilih:

- a) File *about.astro* berfungsi untuk menampilkan informasi tentang Deskripsi atau profil radyakartiyasa, dengan data yang diambil secara dinamis dari API dan ditampilkan sesuai dengan bahasa yang dipilih oleh pengguna. Berikut adalah penjelasan kode *about.astro*:

```

---
import Base from "@components/layouts/Base.astro"  Cannot find module '@components/layo
import { fetchLang, fetch_data, strip_tags, file_handle } from "@utils/projectUtils.js";

```

Gambar 3.85 *Import about.astro*

Dalam kode ini, Base digunakan sebagai komponen layout utama untuk membungkus halaman dengan elemen umum seperti *header* dan *footer*. Fungsi *fetchLang* mengambil data bahasa yang dipilih pengguna, sedangkan *fetch_data* digunakan untuk mengambil konten halaman About US dari API. *Strip_tags* berfungsi menghapus tag HTML dari deskripsi agar hanya teks bersih yang ditampilkan, dan *file_handle* mengubah ID file atau gambar dari API menjadi URL yang bisa ditampilkan di halaman (Gambar 3.85).

```

export async function getStaticPaths() {
  |   return fetchLang();
}

const {lang} = Astro.params;

```

Gambar 3.86 *GetStaticPaths about.astro*

Kode *getStaticPaths* yang digunakan untuk rute dinamis, yang mengembalikan hasil dari *fetchLang()*, yang mengambil data bahasa yang tersedia, dan memastikan halaman dipra-render untuk setiap bahasa yang dipilih. *Astro.params* berdasarkan bahasa yang dipilih oleh pengguna dengan parameter *lang* (Gambar 3.86).

```

const data_about_us = await fetch_data('about_us?fields=cover_image.*,translations.*', lang);
export const prerender = true;
---

```

Gambar 3.87 Mengambil data *about.astro*

Data halaman About Us diambil menggunakan fungsi *fetch_data* untuk mengambil informasi seperti *cover image(cover_image)* dan terjemahan dari teks seperti judul dan deskripsi. Prerender bernilai *true* memastikan halaman ini di pra-render saat proses build (Gambar 3.87).

```

<Base pageTitle={data_about_us.translations[0].sub_title} lang={lang} activePage="/about">
  <section class="intro-section section">
    <img src={file_handle(data_about_us.cover_image)} alt={data_about_us.translations[0].title}
      class="img-fluid this-cover" loading="lazy" decoding="async" />
    <div class="container">
      <div class="inner">
        <div class="section-title white">
          <h2 class="title section-heading-lg">{data_about_us.translations[0].title}</h2>
        </div>
      </div>
    </div>
  </section>

```

Gambar 3.88 Struktur *about.astro*

Bagian body halaman menggunakan layout *BASE*, yang menerima properti *pageTitle*, *lang*, dan *activePage*. Di dalamnya, terdapat dua *section*. *Section* pertama menampilkan gambar sampul (*cover_image*) serta judul dari halaman yang diambil dari *data_about_us.translations[0].title*. Gambar diolah menggunakan *file_handle()* dan teks dijaga keamanannya dengan penggunaan atribut *alt* dan *loading="lazy"* (Gambar 3.88).

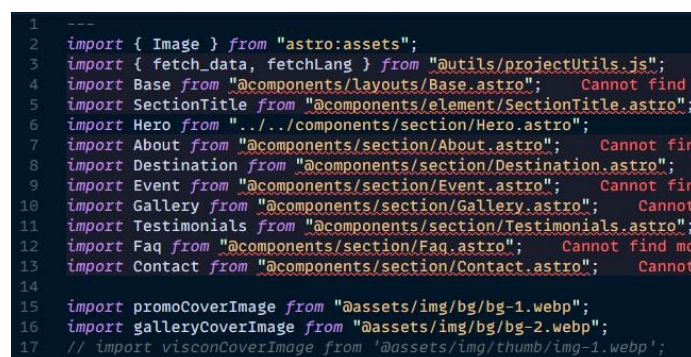


```
<section class="single-content section section-lg">
  <div class="container">
    <div class="reset-style">{strip_tags(data_about_us.translations[0].description)}</div>
  </div>
</section>
</Base>
```

Gambar 3.89 *Single-content about.astro*

Kemudian, *section* kedua menampilkan deskripsi konten yang juga berasal dari *data_about_us.translations[0].description*. Untuk mencegah *HTML injection* atau gangguan tampilan, deskripsi ini diproses menggunakan *strip_tags()* agar hanya teks yang bersih yang ditampilkan. Secara keseluruhan, file ini menyusun halaman *About US* dengan pendekatan modular, aman, dan multibahasa, memanfaatkan data dari API untuk mengisi kontennya secara dinamis tergantung pada parameter bahasa di URL (Gambar 3.89).

- b) File *index.astro* ini merupakan halaman beranda utama (*homepage*) *rad yakartiyasa*. File ini menyusun struktur berbagai bagian utama situs seperti hero banner, bagian tentang situs, destinasi, *event*, promo, galeri, testimoni, FAQ, kontak secara dinamis berdasarkan bahasa yang dipilih oleh user. Berikut penjelasan terperinci dalam bentuk narasi dengan kutipan kode:



```
1 ---
2 import { Image } from "astro:assets";
3 import { fetch_data, fetchLang } from "@utils/projectUtils.js";
4 import Base from "@components/layouts/Base.astro";
5 import SectionTitle from "@components/element/SectionTitle.astro";
6 import Hero from "../components/section/Hero.astro";
7 import About from "@components/section/About.astro";
8 import Destination from "@components/section/Destination.astro";
9 import Event from "@components/section/Event.astro";
10 import Gallery from "@components/section/Gallery.astro";
11 import Testimonials from "@components/section/Testimonials.astro";
12 import Faq from "@components/section/Faq.astro";
13 import Contact from "@components/section/Contact.astro";
14
15 import promoCoverImage from "@assets/img/bg/bg-1.webp";
16 import galleryCoverImage from "@assets/img/bg/bg-2.webp";
17 // import visconCoverImage from "@assets/img/thumb/img-1.webp";
```

Gambar 3.90 *Import homepage index.astro*

Terdapat sejumlah *import* yang mencakup komponen dan utilitas yang akan digunakan seperti layout dasar *Base.astro*, fungsi- fungsi utilitas dari *projectUtils.js*, serta berbagai komponen konten seperti *Hero*, *About*, *Destination*, *Event*, dan lainnya (Gambar 3.90).

```
export async function getStaticPaths() {
  return fetchLang();
}

const { lang } = Astro.params;
```

Gambar 3.91 *GetStaticPaths homepage index.astro*

Fungsi *getStaticPaths()* digunakan untuk menghasilkan semua jalur(path) berdasarkan bahasa yang tersedia dari *backend*. Ini memungkinkan halaman *homepage* dirender secara statis untuk tiap bahasa. Setelah itu, file mengekstrak parameter bahasa dari URL (Gambar 3.91).

```
const interfaces = await fetch_data(
  "interfaces?fields=key,translations.*",
  lang,
);
const home_title = await fetch_data(
  "navigations?filter[URL][_empty]=true&fields=translations.text",
  lang,
);
const destination_max_items = await fetch_data(
  "destination_index?fields=max_items",
  lang,
);
const destination_section = await fetch_data(
  "destination_index?fields=translations.*,destinations.id,destinations.ticketing_url,destinations.thumbnail.*,destinations.translations.title,destinations.translations.short_description&deep[destinations][_limit]=5&destination_max_items.max_items&deep[destinations][translations][_filter][languages_code][_eq]=5(lang)",
  lang,
);
const event_max_items = await fetch_data("event_index?fields=max_items");
const event_section = await fetch_data(
  "event_index?fields=translations.*,events.id,events.open_date,events.close_date,events.ticketing_url,events.thumbnail.*,events.translations.title,events.translations.short_description&deep[events][_limit]=5(event_max_items.max_items)&deep[events][translations][_filter][languages_code][_eq]=5(lang)",
  lang,
);
const promo_section = await fetch_data("promo?fields=*,translations.*", lang);
const gallery_section = await fetch_data(
  "home_gallery?fields=items.directus_files_id.*,items.translations.*,translations.*&deep[items][translations][_filter][languages_code][_eq]=5(lang)",
  lang,
);
const testimonial_section = await fetch_data(
  "testimonial?fields=translations.*,testimonies.*,testimonies.profile_picture.*",
  lang,
);
const faq_section = await fetch_data(
  "faq?fields=translations.*,question_and_answer.translations.*&deep[question_and_answer][translations][_filter][languages_code][_eq]=5(lang)",
  lang,
);
```

Gambar 3.92 *Fetching data homepage index.astro*

Berbagai data kemudian diambil dari *backend* API menggunakan fungsi *fetch_data()* berdasarkan lang, seperti informasi navigasi, daftar destinasi, *event*, promo yang aktif, galeri, testimoni, dan FAQ (Gambar 3.92).

```
<Base pageTitle={home_title[0].translations[0].text} lang={lang} activePage="/">
  <Hero lang={lang} />
```

Gambar 3.93 *Layout homepage index.astro*

Halaman ini dibungkus dalam komponen layout *Base* yang menerima properti *pageTitle*, *lang*, dan *activePage*. Hal pertama yang ditampilkan adalah komponen *Hero* (Gambar 3.93).

```
<About lang={lang} interfaces={interfaces} />

<section class="destination-section section section-lg" id="destinations">
  <div class="container">
    <SectionTitle
      class="style-1"
      subtitle={destination_section.translations[0].tag
        ? destination_section.translations[0].tag
        : ""}
      title={destination_section.translations[0].title
        ? destination_section.translations[0].title
        : ""}
      description=""
      linkText={destination_section.translations[0].see_all_text
        ? destination_section.translations[0].see_all_text
        : ""}
      linkUrl={`/${lang}/destination`}
    />
    <Destination
      lang={lang}
      interfaces={interfaces}
      data_destination={destination_section.destinations}
    />
  </div>
</section>
```

Gambar 3. 94 *About & Destination homepage index.astro*

Kemudian diikuti oleh komponen *About* dan *Destination*, yang masing-masing menampilkan informasi tentang situs dan daftar destinasi wisata (Gambar 3. 94).

```
{
  promo_section.status == "active" ? {
    <section class="promo-section section section-lg">
      <Image
        src={promoCoverImage}
        alt="cover"
        class="img-fluid cover-img"
      />
      <div class="container">
        <div class="inner">
          <div class="section-title white style-1">
            <div>
              <h2 class="title section-heading">
                {promo_section.translations[0].heading}
              </h2>
              <p class="description section-description-md">
                {promo_section.translations[0].description}
              </p>
            </div>
            <div>
              <a
                href={promo_section.button_url}
                class="btn btn-padd btn-outline-white"
              >
                {promo_section.translations[0].button_label}
              </a>
            </div>
          </div>
        </div>
      </div>
    </section>
  } : {
    You, 5 days ago • folder tour dan merchandise.astro masih ada
  }
  <Fragment />
}
```

Gambar 3.95 *Promo section homepage index.astro*

Jika promo sedang aktif dengan ditentukan oleh *promo_section.status == "active"*, maka akan ditampilkan sebuah *section* khusus promo, lengkap dengan gambar latar dan tombol aksi (Gambar 3.95).

```
<section class="event-section section section-md">
  <div class="container">
    <SectionTitle
      class="style-1"
      subtitle={event_section.translations[0].tag
        ? event_section.translations[0].tag
        : ""}
      title={event_section.translations[0].title
        ? event_section.translations[0].title
        : ""}
      description=""
      linkText={event_section.translations[0].see_all_text
        ? event_section.translations[0].see_all_text
        : ""}
      linkUrl={'/$lang/event'}
    />
    <Event
      lang={lang}
      interfaces={interfaces}
      data_event={event_section.events}
    />
  </div>
</section>

<section class="gallery-section section section-md">...
</section>

<section class="testimonials-section section section-lg">...
</section>

<section class="faq-section section section-lg">...
</section>
```

Gambar 3. 96 Couple section homepage index.astro

Setelah itu ditampilkan *event* mendatang melalui komponen *Event*, lalu galeri visual melalui *Gallery*, testimoni user lewat *Testimonials*, dan bagian tanya-jawab melalui *Faq* (Gambar 3.96).

```
<Contact lang={lang} />
</Base>
```

Gambar 3.97 Contact section homepage index.astro

Terakhir, bagian kontak pengunjung yang dapat digunakan user untuk menyampaikan pesan. Secara keseluruhan file *index.astro* membentuk struktur halaman *homepage* untuk situs radyakartiyasa. Dengan memanfaatkan *fetch_data()* dan parameter bahasa (*lang*), halaman ini menampilkan konten dinamis yang mencakup berbagai bagian situs seperti destinasi, *event*, promo, dan galeri, dalam bahasa yang sesuai. Komponen-komponen yang digunakan tertata rapi dengan desain modular, dan setiap bagian ditampilkan dengan data relevan tersedia dari

CMS (Gambar 3.97).

c) Folder *destination* merupakan bagian sistem routing dinamis yang berfungsi untuk menampilkan halaman daftar destinasi dan detail tiap destinasi wisata sesuai bahasa yang dipilih. Terdapat dua file utama:

1) File *index.astro* digunakan untuk menampilkan daftar semua destinasi wisata dalam bahasa yang ditentukan. Berikut potongan kodenya:

```

---
import { Image } from 'astro:assets';
import { fetch_data, fetchLang, slugify, strip_tags, file_handle,
get_interface } from '@utils/projectUtils.js';
import Base from '@components/layouts/Base.astro';
import SectionTitle from '@components/element/SectionTitle.astro';
import heroCover from '@assets/img/bg/bg-1.webp';

```

Gambar 3.98 Import destination *index.astro*

Modul *fetch_data* dan *fetchlang* digunakan untuk mengambil data dari CMS, sedangkan *slugify*, *strip_tags*, dan *file_handle* membantu memproses konten sebelum ditampilkan. Komponen *Base* dan *SectionTitle* digunakan untuk membangun tata letak halaman yang konsisten (Gambar 3.98).

```

export async function getStaticPaths() {
  return fetchLang();
}

const { lang } = Astro.params;

const interfaces = await fetch_data('interfaces?fields=key,translations.*', lang);
const data_destination = await fetch_data('destination_index?fields=destinations.*, destinations.thumbnail.*,
destinations.translations.*, translations.*&deep[destinations][translations][_filter][languages_code][_eq]=%'
+ lang, lang);

```

Gambar 3.99 *GetStaticPaths()* destination *index.astro*

Fungsi *getStaticPaths()* digunakan agar halaman ini bisa di prerender untuk setiap bahasa yang tersedia, memungkinkan URL seperti */id-ID/destination* atau */en-US/destination*. Kemudian, parameter *lang* diambil dari URL, dan digunakan untuk mengambil data daftar destinasi serta antarmuka teks *interfaces* dari API (Gambar 3.99).

```

<Base pageTitle={data_destination.translations[0].title} lang={lang} activePage="/destination">
  <section class="hero-2-section">
    <Image src={heroCover} alt="cover" class="img-fluid cover-img" loading="lazy" />
    <div class="container">
      <div class="inner">
        <SectionTitle class="style-1 white mb-0" subtitle="" title={data_destination.translations
[0].title ? data_destination.translations[0].title : ''} description="" linkText=""
linkUrl=""></SectionTitle>
      </div>
    </div>
  </section>

```

Gambar 3.100 *Hero-2-section destination index.astro*

Struktur halaman utama *destination* dibungkus oleh layout *Base*, yang menerima judul halaman, parameter bahasa, dan tanda halaman aktif untuk navigasi. Bagian pertama adalah hero section, yang menampilkan gambar sampul (heroCover) dan judul utama halaman destinasi menggunakan *SectionTitle* (Gambar 3.100).

```

<section class="section section-lg">
  <div class="container">
    <div class="list-post-area" id="pagination-area">
      {data.destination.destinations.map((item, index) => {
        const url_detail = `//${lang}/destination/${item.id}-${simplify(item.translations[0].title)}`;
        return (
          <div class="list-item">
            <article class="post hv-zoom">
              <div class="post-head">
                <a href={url_detail}>
                  <img src={file_handle(item.thumbnail)} alt={item.translations[0].title} class="img-fluid post-img" loading="lazy" decoding="async" />
                </a>
              </div>
              <div class="post-body">
                <a href={url_detail}>
                  <h2 class="post-title section-description">{item.translations[0].title}</h2>
                  </a>
                  {item.translations[0].short_description ? (
                    <p class="post-description section-description-md mb-0">
                      {strip_tags(item.translations[0].short_description)}</p>
                  ) : ''}
                <div class="post-act">
                  <a href={url_detail} class="link link-arrow" title={get_interface(interfaces, "see-detail")[0].translations[0].hover_text}>
                    {get_interface(interfaces, "see-detail")[0].translations[0].text}</a>
                    {item.ticketing_url && item.ticket_status == 'active' ? (
                      <a href={item.ticketing_url} class="btn btn-outline-accent btn-hover-icon-right">
                        {get_interface(interfaces, "booking-link")[0].translations[0].text} <i class="icon feather icon-arrow-right"></i></a>
                    ) : ''}
                  </div>
                </div>
              </article>
            </div>
          </div>
        )
      })}
    </div>
  </div>
</section>
</Base>

```

Gambar 3.101 *Section destination index.astro*

Setelah itu, daftar destinasi ditampilkan dalam struktur grid. Setiap destinasi dibuat menggunakan `map()` yang menghasilkan elemen *article* dengan gambar, judul, deskripsi pendek, serta dua tombol yaitu “Lihat Detail” dan “Booking sekarang” (jika tersedia) (Gambar 3.101).

```

71
72 <script type="module">
73   import { pagination_handle } from "/src/script/pagination.min.js";
74
75   document.addEventListener('DOMContentLoaded', () => {
76     pagination_handle();
77   });
78 </script>

```

Gambar 3.102 *Pagination destination index.astro*

Terakhir, *script* tambahan disisipkan untuk menangani *pagination* (Gambar 3.102).

Secara keseluruhan, file ini membentuk halaman daftar destinasi wisata yang bersifat


```

<base page={title} lang={lang} sectionPage="/destination">
<section class="intro-section section">
  <div class="pt-cover" data-bg={theme_color}></div>
  <img src={file_handle(thumbnail)} alt="cover" class="img-fluid this-cover" loading="lazy" decoding="async" />

  <div class="container">
    <div class="inner">
      <div class="section-title white">
        <h2 class="title section-heading-lg">{title}</h2>
      </div>

      <div class="swiper-container">
        <div class="swiper" id="intro-slider">
          <div class="swiper-wrapper">
            <!-- {data_gallery.map(item_data) => {
              (item_data.translations[0].item_data.translations.map(item_data_t) => {
                <li class="title-item">{item_data_t.slug}</li>
              })
            }) -->

            (data_gallery.map(item_gal) => {
              item_gal.gallery.map(item_gal_file) => {
                <div class="swiper-slide">
                  <article class="post hv-zoom">
                    <div class="post-head">
                      <a href={file_handle(item_gal_file.directus_files_id)} class="this-mg-img"
                        data-title="">
                        <img src={file_handle(item_gal_file.directus_files_id)} alt="image gallery"
                          class="img-fluid post-img" loading="lazy" decoding="async" />
                      </div>
                    </div>
                  </article>
                </div>
              })
            })
          </div>
          <!-- <div class="swiper-button-2 swiper-button-next swiper-intro-slider-next"></div> -->
          <!-- <div class="swiper-button-2 swiper-button-prev swiper-intro-slider-prev"></div> -->
        </div>
      </div>
    </div>
  </section>

```

Gambar 3.105 Images destination [id]-[slug].astro

Bagian ini menampilkan gambar latar belakang dengan warna dinamis yang diambil dari *theme_color*. Gambar destinasi ditampilkan sebagai *cover* dengan atribut *lazy loading* untuk mengoptimalkan performa halaman. Judul dari destinasi juga ditampilkan di dalam elemen tag *h2*. Kelas *swiper-container* menggunakan *swiper slider* untuk menampilkan gambar-gambar destinasi secara interaktif. Setiap gambar dalam *data_gallery* di *loop* dan ditampilkan dalam bentuk slide yang bisa digulir oleh pengguna. Gambar diambil dengan menggunakan *file_handle()* (Gambar 3.105).

```

<section class="single-content destination-single-section section section-lg">
  <div class="container">
    <div class="pt-square" data-bg={theme_color}></div>

    <div class="eli-area">
      <div class="eli-content">
        <div class="reset-style">{strip_tags(description)}</div>

        <div class="eli-list-area mt-5">
          {attractions.map(item_at, index) => {
            <div class="eli-list-item">
              <div class="thumb">
                <img src={file_handle(item_at.image)} alt={item_at.translations[0].
                  title} class="img-fluid" loading="lazy" decoding="async" />
              </div>
              <div class="content">
                <h3 class="title section-description-md">{item_at.translations[0].
                  title}</h3>
                <p class="description section-description-md">{strip_tags(item_at.
                  translations[0].description)}</p>
              </div>
            </div>
          })
        </div>
      </div>
    </div>
  </section>

```

Gambar 3.106 Deskripsi & daftar destinasi [id]-[slug].astro

Bagian ini menampilkan deskripsi dan daftar atraksi wisata terkait destinasi. Setiap atraksi menampilkan gambar dan deskripsi singkat. Deskripsi juga ditampilkan setelah diproses dengan *strip_tags* (Gambar 3.106).

d) Folder *event* adalah folder yang berisi file-file penting yang berfungsi untuk menampilkan daftar *event* dan detail tiap *event* dalam bahasa yang ditentukan. Terdapat dua file utama:

- 1) File *index.astro* ini digunakan untuk menampilkan halaman daftar *event* di situs yang dibangun untuk mendukung tampilan bahasa yang ditentukan. Berikut potongan kodenya:

```
import { Image } from 'astro/assets';
import { fetch_data, fetchLang, slugify, strip_tags, date_format, file_handle, get_interface } from '@utils/projectutils.js';
import Base from '@components/layouts/Base.astro';
import SectionTitle from '@components/element/SectionTitle.astro';
import heroCover from '@assets/img/bg/bg-1.webp';
```

Gambar 3.109 *Import event index.astro*

Halaman ini pertama-tama mengimpor modul-modul dan utilitas yang diperlukan, seperti *Image* untuk penanganan gambar, *fetch_data* dan *fetchLang* untuk mengambil data dari API, serta beberapa fungsi lainnya untuk menformat data dan URL (Gambar 3.109).

```
export async function getStaticPaths() {
  return fetchLang();
}

const { lang } = Astro.params;

const interfaces = await fetch_data('interfaces?fields=key,translations.', lang);
const data_event = await fetch_data('event.index?fields=events.*,events.thumbnail.*,events.event_logo.*,events.translations.*,translations.*&deep[events][translations][_filter][languages_code][_eq]=' + lang, lang);
```

Gambar 3.110 *GetStaticPaths event index.astro*

Fungsi *getStaticPaths()* digunakan untuk mendapatkan semua path bahasa yang tersedia, yang memungkinkan halaman ini diprerender untuk setiap bahasa (seperti id-ID/*event* atau en-US/*event*). Kemudian, data untuk *event* dan antarmuka pengguna dalam bahasa yang dipilih diambil dari CMS menggunakan *fetch_data()*. Ini termasuk data *event* seperti gambar, judul, deskripsi singkat, dan detail lainnya, yang akan ditampilkan di halaman (Gambar 3.110).

```
<Base pageTitle={data_event.translations[0].title} lang={lang} activePage="/event">
  <section class="hero-2-section">
    <Image src={heroCover} alt="cover" class="img-fluid cover-img" loading="lazy" />
    <div class="container">
      <div class="inner">
        <SectionTitle class="style-1 white mb-8" subTitle="" title={data_event.translations[0].title ? data_event.translations[0].title : ''} description="" linkText="" linkUrl="" />
      </div>
    </div>
  </section>
```

Gambar 3.111 *Layout event index.astro*

Bagian selanjutnya adalah struktur halaman yang dibungkus dalam layout Base. Layout ini menerima beberapa properti seperti `pageTitle`, `lang`, dan `activePage`. Hero-2-section menampilkan gambar dan judul halaman (Gambar 3.111).

```
<section class="section section-lg">
  <div class="container">
    <div class="list-post-area" id="pagination-area">
      {data_event.events.map((item, index) => {
        const url_detail = `/s{lang}/event/${item.id}-${slugify(item.translations[0].title)}`;

```

Gambar 3.112 Daftar konten *event index.astro*

Setelah itu, daftar *event* ditampilkan menggunakan `.map()` untuk memproses setiap item *event*. Setiap *event* terdiri dari gambar thumbnail, judul, tanggal mulai dan berakhir, serta deskripsi pendek. Ketika user mengklik judul *event*, mereka diarahkan ke halaman detail *event* yang sesuai (Gambar 3.112).

```
<div class="post-act">
  <a href={url_detail} class="link link-arrow" title={get_interface(interfaces,"see-detail")[0].
    translations[0].hover_text}>{get_interface(interfaces,"see-detail")[0].translations[0].text}</a>
</div>
```

Gambar 3.113 Penjelasan *event index.astro*

Di dalam loop `.map()`, gambar dan judul *event* ditampilkan, dan deskripsi pendek akan ditampilkan jika ada. Sebuah tombol “Lihat Detail” juga diberikan, yang teks dan tautannya diterjemahkan menggunakan `get_interface()` (Gambar 3.113).

```
<script type="module">
  import { pagination_handle } from "/src/script/pagination.min.js";

  document.addEventListener('DOMContentLoaded', () => {
    pagination_handle();
  });
</script>
```

Gambar 3.114 *Pagination event index.astro*

Terakhir bagian *pagination* diaktifkan dengan memanfaatkan script `pagination_handle` yang diimpor, yang memungkinkan pengguna untuk menavigasi antara halaman daftar *event* jika jumlah *event* terlalu banyak untuk ditampilkan dalam satu halaman (Gambar 3.114).

File *index.astro* ini membentuk halaman daftar *event* yang dinamis, di mana informasi tentang *event* seperti gambar, judul, deskripsi, dan tombol pemesanan tiket ditampilkan dengan data yang diambil dari CMS menggunakan API. Halaman ini mendukung tampilan bahasa yang ditentukan dengan menggunakan parameter `lang` dan menyediakan *pagination* untuk navigasi yang lebih baik saat jumlah *event* banyak.

- 2) File `[id]-[slug].astro` ini digunakan untuk menampilkan halaman detail *event* berdasarkan *ID* dan *slug event* yang diterima dari URL. Halaman ini menampilkan berbagai informasi terkait *event*, termasuk gambar, deskripsi, jam operasional, lokasi, dan tombol pemesanan tiket. Berikut adalah penjelasan terperinci potongan kode:

```
import { host, headers, slugify, getLang, strip_tags, date_format,
day_format, file_handle } from "@utils/projectUtils.js";
import Base from "@components/layouts/Base.astro";
import SectionTitle from "@components/element/SectionTitle.astro";
```

Gambar 3.115 Import event `[id]-[slug].astro`

Pada bagian awal di Gambar 3.115, file ini mengimpor beberapa modul penting, Seperti *Base* untuk layout halaman *SectionTitle* untuk menampilkan judul *section*, dan beberapa fungsi dari `@utils/projectUtils.js` untuk mengambil dan memanipulasi data. Di sini, juga terdapat pengaturan untuk mengakses data destinasi dan *event* dari API (Gambar 3.115).

```
const event = await fetch(`${host}/api/event/${id}/${slug}`).then((res) => res.json());
const event_data = event.data;

const event_data = {
  id: event_data.id,
  slug: event_data.slug,
  title: event_data.title,
  description: event_data.description,
  image: event_data.image,
  location: event_data.location,
  time: event_data.time,
  status: event_data.status,
  created_at: event_data.created_at,
  updated_at: event_data.updated_at,
};

const event_data = {
  id: event_data.id,
  slug: event_data.slug,
  title: event_data.title,
  description: event_data.description,
  image: event_data.image,
  location: event_data.location,
  time: event_data.time,
  status: event_data.status,
  created_at: event_data.created_at,
  updated_at: event_data.updated_at,
};

const event_data = {
  id: event_data.id,
  slug: event_data.slug,
  title: event_data.title,
  description: event_data.description,
  image: event_data.image,
  location: event_data.location,
  time: event_data.time,
  status: event_data.status,
  created_at: event_data.created_at,
  updated_at: event_data.updated_at,
};
```

Gambar 3.116 *GetStaticPaths* event `[id]-[slug].astro`

Selanjutnya, pada Gambar 3.116 fungsi `getStaticPaths()` digunakan untuk menghasilkan jalur dinamis untuk setiap *event* berdasarkan *ID* dan *slug* yang dihasilkan dari parameter `lang`. Hal ini memungkinkan halaman untuk diprerender berdasarkan bahasa yang dipilih.

```
const {lang, id, slug} = Astro.params;
const {
  event_logo,
  title,
  location,
  ticketing_url,
  cover_image,
  description,
  date,
  maps_url,
  operational_hours,
  supported_by,
  dos,
  dnts,
  reg_section,
  other_section,
  period_section,
  organizer_section,
  info_section,
  ops_section,
  loc_section,
  loc_button,
  res_button
} = Astro.props;
```

Gambar 3.117 Parameter *event [id]-[slug].astro*

Setelah data destinasi diambil, pada Gambar 3.117 parameter *lang*, *id*, dan *slug* digunakan untuk mengambil data *event* yang sesuai. Informasi seperti *title*, *description*, dan *cover_image* dimasukkan ke dalam properti yang nantinya akan digunakan untuk ditampilkan di halaman.

```
<section class="intro-section section">
  {cover_image.type === 'video/mp4' ?
    <video muted playsinline autoplay loop class="this-cover">
      <source src={file_handle(cover_image)} type="video/mp4"/></video>
    :
    <img src={file_handle(cover_image)} alt="cover" class="img-fluid this-cover" loading="lazy" decoding="async" />
  }

  <div class="container">
    <div class="inner">
      <div class="section-title white">
        <h2 class="title section-heading-lg">{title}</h2>
      </div>
    </div>
  </div>
</section>
```

Gambar 3.118 *Header event [id]-[slug].astro*

Di Gambar 3.118, *section* ini menampilkan gambar atau video sebagai latar belakang, diikuti dengan judul *event* yang diambil dari *title*. Jika *cover_image* bertipe video mp4, maka video akan diputar, jika tidak, gambar biasa akan ditampilkan.

```

<section class="destination-single-section section section-lg">
  <div class="container">
    <div class="el1-area el1-event">
      <div class="el1-content">
        {date && operational_hours &&
          <div class="mb-5">
            <h4 class="el1-title section-description">Tanggal dan Waktu</h4>
            <div class="el-info">...
          </div>
        </div>
      </div>
    </div>
  </div>
</section>

```

Gambar 3.119 Informasi event *[id]-[slug].astro*

Bagian berikutnya pada Gambar 3.119 adalah informasi *event* seperti tanggal dan jam operasional. Tanggal *event* ditampilkan menggunakan *date_format()*, dan jam operasional ditampilkan dalam format yang telah diproses, menggunakan *operational_hours*.

```

{ ticketing_url := null &&
  <a href={ticketing_url} class="btn btn-cta btn-padd w-100 text-start" target="blank">
    <svg class="icon me-1" xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none">
      <path d="M10 8V7M10 12.5V11.5M10 17V16M5.2 4H18.8C19.9201 4 20.4802 4 20.988 4.21799C21.2843 4.40973 21.5903
        4.71569 21.782 5.09202C22 5.51984 22 6.0799 22 7.2V8.5C20.067 8.5 18.5 10.067 18.5 12.0185 13.933 20.067 15.
        5 22 15.5V16.8C22 17.9201 22 18.4802 21.782 18.988C21.5903 19.2843 21.2843 19.5903 20.988 19.782C20.4802 20.
        10 9201 20 19.8 20H5.2C4.0799 20 3.51984 20 3.09202 19.782C2 11569 19.5903 2.40973 19.2843 2.21799 18.988C2
        18.4802 2 17.9201 2 16.8V15.5C3.933 15.5 5.5 13.933 5.5 12C5.5 10.067 3.933 8.5 2 8.5V7.2C2 6.0799 2 5.51984
        2.21799 5.09202C2.40973 4.71569 2.71569 4.40973 3.09202 4.21799C3.51984 4 4.0799 4 5.2 4Z" stroke="#D6A84D"
        stroke-width="2" stroke-linecap="round" stroke-linejoin="round"/>
    </svg>
    <span>{res_button}</span>
  </a>
  {/ <i class="feather icon-arrow-right ms-1"></i> </ > }
}

```

Gambar 3.120 Tombol tiket URL event *[id]-[slug].astro*

Di Gambar 3.120 tombol pemesanan tiket akan ditampilkan jika *ticketing_url* tersedia, memberikan pengunjung akses untuk memesan tiket secara langsung.

```

<div class="content">
  <SectionTitle class="style-1 white" subtitle="" title={reg_section} description="" linkText=""
    linkUrl=""></SectionTitle>
  <ul class="nav nav-tabs" id="visconditions-tab" role="tablist">
    <li class="nav-item" role="presentation">
      <a class="nav-link section-description-md active" id="nav-viscon-dos-tab"
        data-bs-toggle="tab" data-bs-target="#viscon-dos-tab" role="tab"
        aria-controls="viscon-dos-tab" aria-selected="true">
        Dos
      </a>
    </li>
    <li class="nav-item" role="presentation">
      <a class="nav-link section-description-md" id="nav-viscondont-tab" data-bs-toggle="tab"
        data-bs-target="#viscondont-tab" role="tab" aria-controls="viscondont-tab"
        aria-selected="false">
        Don'ts
      </a>
    </li>
  </ul>
</div>

```

Gambar 3.121 Navigasi *Dos & Don'ts event [id]-[slug].astro*

SectionTitle digunakan untuk menampilkan judul besar yang dapat disesuaikan, di sini mengambil nilai dari properti *reg_section*. Selanjutnya, terdapat navigasi tab yang memungkinkan pengguna untuk beralih antara dua kategori informasi yaitu *Dos* dan *Don'ts* (Gambar 3.121).

```

<div class="tab-content mt-4" id="visconditions-tabContent">
  <div class="tab-pane fade show active" id="viscon-dos-tab" role="tabpanel"
    aria-labelledby="nav-viscon-dos-tab">
    <ul class="list-check">
      {dos && dos.map((item_dos) => (
        item_dos.translations[0] && (
        <li class="list-item">{item_dos.translations[0].entry}</li>
        ))}
      </ul>
    </div>

    <div class="tab-pane fade" id="viscondont-tab" role="tabpanel"
      aria-labelledby="nav-viscondont-tab">
      <ul class="list-check">
        {dons && dons.map((item_dons) => (
          item_dons.translations[0] && (
          <li class="list-item">{item_dons.translations[0].entry}</li>
          ))}
        </ul>
      </div>

```

Gambar 3. 122 Informasi *Don'ts & Dos event [id]-[slug].astro*

Setelah tab navigasi, ada konten tab yang diisi dengan informasi mengenai *Dos* dan *Don'ts*. Setiap tab menampilkan daftar aturan atau tindakan yang harus dilakukan (*Dos*) dan yang tidak boleh dilakukan (*Don'ts*). Data untuk setiap aturan diambil dan ditampilkan menggunakan `.map()` untuk iterasi elemen, dan masing-masing item ditampilkan dalam elemen tag `li` sesuai dengan bahasa yang ditentukan (Gambar 3. 122).

```

<div class="thumb">
  <img src={file_handle(event_logo)} alt="cover" class="img-fluid thumb-img" loading="lazy"
    decoding="async" />
</div>

```

Gambar 3.123 Logo event *[id]-[slug].astro*

Di bagian akhir section, terdapat logo *event* yang diambil dari `event_logo`. Gambar logo ditampilkan menggunakan `file_handle()` untuk memastikan URL gambar yang valid dan ditampilkan secara asinkron dengan atribut `loading-“lazy”` (Gambar 3.123).

File ini membuat halaman detail *event* yang menampilkan berbagai informasi tentang *event*, termasuk gambar, deskripsi, waktu, lokasi, serta tombol untuk pemesanan tiket. Data *event* yang ditampilkan diambil secara dinamis dari API berdasarkan *ID* dan *slug event*. Fitur pemilihan bahasa yang menggunakan parameter `lang` pada URL.

3.2.8 Configuration Of The Project

a. File `astro.config.mjs` berisi konfigurasi untuk proyek situs *rad yakartiyasa*. File ini menyertakan pengaturan untuk mendukung multibahasa, dengan bahasa indonesia sebagai bahasa *default* dan pengalihan otomatis ke halaman yang sesuai. Selain itu, file ini mengonfigurasi *server* pengembangan lokal untuk mempermudah proses pengembangan, serta menyediakan alias direktori untuk kemudahan impor file. Menyiapkan proyek untuk di deploy dengan adapter *cloudflare* untuk memastikan situs berjalan dalam *environment cloudflare*.

```
import { defineConfig } from 'astro/config';

import node from '@astrojs/node'; 1.3k (gzipped: 671)

import cloudflare from '@astrojs/cloudflare';
```

Gambar 3.124 *Import astro.config.mjs*

Pada bagian pertama, kode mengimpor beberapa modul, yaitu *defineConfig* dari *Astro*, serta adapter untuk *Node.js* dan *Cloudflare*. *Cloudflare* digunakan untuk menghosting situs ini di platform *Cloudflare Workers*, sementara adapter *Node.js* tidak digunakan untuk saat ini (Gambar 3.124).

```
export default defineConfig({
  vite: {
    server: {
      allowedHosts: ['astro-radya.zakyabiyyu.com']
    },
    resolve: {
      alias: {
        '@src': '/src',
        '@components': '/src/components',
        '@assets': '/public/assets',
        '@utils': '/utils'
      }
    }
  },
});
```

Gambar 3.125 *DefineConfig astro.config.mjs*

Pada bagian berikutnya, konfigurasi vite disesuaikan dengan menambahkan pengaturan untuk *server* dan resolusi alias direktori. *AllowedHosts* membolehkan *astro-radya.zakyabiyyu.com* saja yang dapat mengakses *server*. Selain itu, *resolve.alias* memungkinkan penggunaan alias untuk direktori seperti *@src* dan *@components* yang memudahkan impor file di dalam proyek (Gambar 3.125).

```

i18n: {
  defaultLocale: "id-ID",
  locales: ["id-ID", "en-US"]
},
redirects: {
  '/': {
    destination: "/id-ID",
    status: 301,
    permanent: true
  }
},

```

Gambar 3.126 I18n *astro.config.mjs*

Selanjutnya, pengaturan internasionalisasi (i18n) diterapkan untuk mendukung dua bahasa, yaitu bahasa Indonesia (id-ID) sebagai bahasa *default* dan bahasa Inggris (en-US). User akan diarahkan ke versi bahasa indonesia dari situs secara otomatis melalui pengaturan *redirect* yang disediakan (Gambar 3.126).

```

server: {
  host: true,
  port: 2709,
},
output: 'server',
adapter: cloudflare(),

```

Gambar 3.127 Server *astro.config.mjs*

Pengaturan *server* menentukan bahwa *server* akan di-host secara lokal, dengan *port* yang ditetapkan pada 2709. Ini memungkinkan pengembangan di lingkungan lokal sebelum situs di-deploy ke *server* produksi. Pada bagian output, proyek ini ditetapkan untuk menghasilkan aplikasi dengan output berupa *server*, yang sesuai dengan pengaturan *cloudflare adapter* yang digunakan (Gambar 3.127).

b. File *package.json* yang digunakan untuk mengonfigurasi proyek dan mengelola skrip serta dependensi yang digunakan dalam proyek.

```
"name": "radyakartiyasa",
  "type": "module",
  "version": "0.0.1",
```

Gambar 3.128 Projects name package.json

Pada bagian informasi proyek, file ini mendefinisikan nama proyek sebagai *radyakartiyasa*, dengan tipe *module* yang menunjukkan bahwa proyek ini menggunakan ES modules untuk struktur kode *javascript* nya, dan versi proyek saat ini adalah 0.0.1 (Gambar 3.128).

```
"scripts": {
  "downloadAssets": "node utils/assetsDownload.js",
  "dev": "npm run downloadAssets && astro dev",
  "start": "astro dev",
  "buildonly": "astro build",
  "build": "npm run downloadAssets && astro build",
  "preview": "npm run build && astro preview",
  "astro": "astro"
},
```

Gambar 3.129 Scripts package.json

Di bagian *scripts*, terdapat beberapa perintah yang bisa dijalankan melalui npm, seperti *downloadAssets* untuk mengunduh aset, *dev* untuk memulai *server* pengembangan, dan *build* untuk membangun proyek menjadi versi statis (Gambar 3.129).

```
"dependencies": {
  "@astrojs/cloudflare": "^12.1.0",
  "@astrojs/node": "^9.0.0",
  "@astrojs/sitemap": "^3.2.1",
  "astro": "^5.0.4",
  "got": "^14.4.7",
  "jsdom": "^25.0.1",
  "node-fetch": "^3.3.2",
  "parse-url": "^9.2.0"
},
"devDependencies": {
  "@types/jsdom": "^21.1.7"
}
```

Gambar 3.130 Dependencies package.json

Bagian *dependencies* mencantumkan pustaka yang diperlukan untuk menjalankan proyek, termasuk *cloudflare* adapter untuk menghosting situs di *cloudflare workers*, *node.js*

adapter untuk menjalankan aplikasi di node.js, serta pustaka seperti *got* untuk permintaan HTTP, *jsdom* untuk manipulasi DOM di *server-side*, dan *node-fetch* untuk permintaan HTTP ke node.js. Sementara itu bagian *devDependencies* mencantumkan pustaka *@types/jsdom*, yang menyediakan definisi *typescript* untuk pustaka *jsdom* (Gambar 3.130).

3.2.9 Activating the Build Process for Project Using Astro

File *projectUtils.js* berisi konfigurasi dan fungsi yang digunakan untuk berinteraksi dengan CMS dan memproses data. Variabel *host* menyimpan URL untuk CMS, sementara objek *headers* berisi detail otentikasi yang diperlukan untuk mengakses API CMS, termasuk *authorization bearer token* untuk autentikasi CMS dan *service token* untuk akses yang aman ke aset yang disimpan di *cloudflare*.

Proses untuk membangun proyek dimulai dengan membuka terminal dan menjalankan perintah *npm run build*. Perintah ini menjalankan *npm run downloadAssets*, yang memanggil skrip *assetsDownload.js* untuk mengunduh aset yang diperlukan dari CMS ke folder lokal (*cms_images*). Setelah itu, perintah *astro build* dijalankan untuk mengompilasi proyek *astro* menjadi situs statis. Setelah proses *build* selesai, aplikasi akan berjalan secara lokal di *port 2709* sesuai konfigurasi di *astro.config.mjs*, dan hasil *build* dapat diakses dengan membuka URL *localhost:2709* di browser.

3.2.10 CI/CD Flow

Dalam pengembangan proyek *website* Radyakartiyasa yang dibangun menggunakan *framework Astro (jamstack)*, penulis memulai dengan membuat sebuah branch lokal bernama *preview*. Branch ini berfungsi sebagai tempat untuk menyimpan fitur atau perubahan baru yang masih dalam tahap pengujian dan belum final. Tujuan dari branch *preview* ini adalah untuk memastikan setiap pembaruan dapat direview terlebih dahulu sebelum masuk ke tahap produksi (*production*).

Setelah membuat branch *preview*, penulis menjalankan *npx astro add cloudflare*. Perintah ini digunakan untuk menambahkan dukungan untuk Cloudflare sebagai platform deployment, khususnya dengan memanfaatkan Cloudflare yang mendukung integrasi langsung dengan proyek *Astro*. Dengan langkah ini, proyek disiapkan agar dapat berjalan dalam lingkungan *serverless* yang disediakan oleh Cloudflare.

Di sisi remote repository, penulis membuat sebuah repository privat di github dengan nama *radyakartiyasa-astro-jamstack*. Repository ini merupakan repository utama untuk menyimpan kode sumber proyek. Di dalamnya, dibuat branch utama bernama *main*, yang nantinya digunakan secara eksklusif untuk menyimpan versi final atau versi produksi dari *website*.

Setelah pengembangan selesai pada branch *preview*, penulis menjalankan perintah `npm run build`. Perintah ini memicu proses build dari framework *Astro*, di mana *Astro* akan merender halaman-halaman menjadi file HTML statis dan mendownload aset-aset yang diperlukan. Hasil build akan disimpan ke dalam folder output default yaitu *dist*.

Kemudian, branch *preview* yang berisi hasil build dan perubahan tersebut di-push ke *github*, sehingga branch tersebut tersedia secara online dalam repository *radyakartiyasa-astro-jamstack*. Untuk menyatukan perubahan ke branch *main*, penulis melakukan pull request dari branch *preview* ke *main*. Pull request ini hanya akan digabungkan setelah mendapatkan persetujuan dari supervisor, untuk memastikan tidak ada kesalahan yang masuk ke produksi.

Setelah branch *main* diperbarui, penulis menghubungkan repository github tersebut ke Cloudflare. Pada proses konfigurasi, dipilih branch *main* sebagai branch yang digunakan untuk deployment ke lingkungan produksi. Dalam pengaturan deployment Cloudflare, framework yang digunakan ditentukan sebagai *Astro*, perintah build yang digunakan adalah `npm run build`, dan output dari build diarahkan ke folder *dist*, yang merupakan folder default dari *Astro*.

Setelah konfigurasi selesai, Cloudflare akan secara otomatis melakukan build dan deployment setiap kali terdapat perubahan pada branch *main*. Hasilnya adalah sebuah *domain* bawaan yang diberikan oleh Cloudflare, yaitu *radyakartiyasa-astro-jamstack.pages.dev*.

Namun, demi keamanan, *domain* ini tidak dibiarkan terbuka untuk umum. Penulis mengaktifkan fitur keamanan dari Cloudflare, yang membatasi akses ke *domain* tersebut. Dengan pengaturan ini, hanya orang-orang dengan email tertentu yang telah ditentukan sebelumnya yang dapat mengakses halaman *website* melalui *domain* ini. Ini sangat berguna untuk keperluan review internal sebelum *website* benar-benar dipublikasikan ke masyarakat.

Untuk kebutuhan publikasi ke khalayak umum, penulis menambahkan sebuah custom *domain*, yaitu *radyakartiyasa-astro-jamstack.zakyabiyyu.com*. Custom *domain* ini dikonfigurasi melalui dashboard Cloudflare dengan cara menghubungkan *subdomain* dari

domain pribadi *zakyabiyyu.com* ke proyek Cloudflare pages. *Domain* ini tidak dibatasi oleh Cloudflare access, sehingga semua orang dapat mengaksesnya tanpa perlu otentikasi. Dengan demikian, *domain* ini digunakan sebagai alamat resmi yang dapat diakses secara publik oleh pengunjung *website*.

a. Proses Deployment dapat dipicu oleh dua hal, yaitu perubahan konten dan perubahan kode. Perubahan konten akan memicu deploy melalui URL Deploy Hook yang tersimpan di CMS, sedangkan perubahan kode akan memicu deploy secara otomatis saat perubahan tersebut dipush ke repositori github:

1. Di *Cloudflare*, dihasilkan URL *deploy hook* yang bersifat rahasia dan terhubung dengan repositori Github *radyakartiyasa-astro-jamstack*. URL ini digunakan untuk disimpan di CMS, sehingga setiap kali ada perubahan konten di CMS, sistem dapat secara otomatis memicu *redploy website* melalui *deploy hook* tersebut.
2. Untuk menerapkan perubahan pada kode, perubahan tersebut harus *dipush* ke repositori *github* yang telah terhubung dengan *cloudflare*. Setelah itu, *cloudflare* akan secara otomatis menjalankan proses build ulang untuk memperbarui dan menampilkan versi terbaru dari *website* berdasarkan perubahan kode tersebut.

Cara lainnya yaitu tidak menggunakan *jamstack*(non-*JAMstack*) dengan menjalankan proyek secara lokal namun tetap bisa diakses dari internet adalah dengan menghubungkan *server* lokal ke *domain* publik menggunakan layanan *Cloudflare*. Dalam hal ini, dibuat sebuah *hostname* publik bernama *astro-radya.zakyabiyyu.com*, yang diarahkan ke *server* lokal pada alamat *localhost* yaitu 2709.

Port 2709 ini merupakan *port default* yang telah ditentukan di konfigurasi proyek, tepatnya di file *astro.config.mjs*. Untuk menjalankan *server* pengembangan *Astro*, cukup gunakan perintah *npm run preview* di terminal. Perintah ini akan menjalankan dua proses utama:

1. Menjalankan *npm run download assets* untuk mengunduh aset-aset yang dibutuhkan proyek.
2. Menjalankan *astro dev*, yang akan memulai *server* pengembangan *Astro* di *port 2709*.

Setelah *server* lokal aktif, *Cloudflare* akan meneruskan akses dari *domain* publik *astro-radya.zakyabiyyu.com* ke *localhost* yaitu 2709. Dengan demikian, proyek dapat diakses secara publik melalui *domain* tersebut meskipun sebenarnya dijalankan secara lokal di perangkat developer.

3.2.11 Melakukan pengujian TTFB *jamstack* dengan non *jamstack*

Dilakukan pengujian terhadap arsitektur *jamstack* dengan arsitektur non-*jamstack* menggunakan parameter *Time to First Byte* (TTFB) melalui platform *SpeedVitals*, mensimulasikan seberapa cepat *website* dapat diakses oleh pengguna baru yang belum pernah membuka situs tersebut sebelumnya. Kedua arsitektur memiliki *source code* yang sama, dengan perbedaan yang terletak pada metode penyajiannya. *JAMstack* akan menyajikan konten secara statis melalui CDN, sedangkan pada non-*JAMstack* menyajikan konten secara statis dari sebuah *server* yang terhubung melalui internet dan dapat diakses oleh publik. Pengujian akan dilakukan menggunakan URL *website JAMstack* (*radhakartiyasa-astro-jamstack.zakyabiyyu.com*.) dan non-*JAMstack* (*astro-radya.zakyabiyyu.com*) yang dimasukkan ke dalam alat uji TTFB *SpeedVitals*. *Website* berbasis *JAMstack* diuji sebanyak dua kali, sedangkan *website non-JAMstack* diuji dua kali pada masing-masing dari tiga lokasi yang berbeda di mana *server* pengujian dijalankan. Pengukuran dilakukan menggunakan beberapa metrik utama, yaitu DNS(waktu yang dibutuhkan hingga *query* penerjemahan dari nama *domain* jadi alamat IP selesai), *Connect*(waktu yang dibutuhkan sejak DNS selesai hingga *server* merespon mengenai versi http yang dijalankan), TLS(waktu yang dibutuhkan sejak *Connect* selesai hingga proses *handshake* SSL/TLS selesai), dan TTFB itu sendiri(Waktu yang dibutuhkan hingga mendapatkan respon *byte* konten pertama dari *server*).



Gambar 3.131 Parameter TTFB *SpeedVitals*(Patel M, 2025)

Hasil TTFB dievaluasi berdasarkan pengukuran yang dilakukan melalui pengujian menggunakan *SpeedVitals*, seperti yang ditunjukkan pada Gambar 3.131. Kategori inilah yang digunakan untuk menentukan *performance grade* dari nilai TTFB tersebut.

Tabel 3.1 Pengujian TTFB *JAMstack*

<i>Server Origin</i>	TES	<i>Performance Grade</i>
CDN <i>Cloudflare</i>	Tes 1	Excellent
CDN <i>Cloudflare</i>	Tes 2	Excellent

Seperti yang ditunjukkan pada Tabel 3.1, kedua pengujian pada arsitektur *JAMstack*—baik pada Tes 1 maupun Tes 2 mendapatkan *performance grade* “*Excellent*”. Ini menunjukkan bahwa waktu respon awal dari *server* ke browser sangat cepat. Hal ini juga mengindikasikan bahwa distribusi konten statis melalui CDN *Cloudflare* mampu meminimalkan hambatan jaringan dan latensi, sehingga menghasilkan pengalaman pengguna yang lebih baik. Keunggulan ini sangat penting, terutama untuk *website* pariwisata yang bertujuan menyajikan informasi secara cepat, stabil, dan dapat diakses dari berbagai wilayah tanpa mengalami keterlambatan dalam pemuatan halaman.

Tabel 3.2 Pengujian TTFB *non-JAMstack*

<i>Server Origin</i>	TES	<i>Performance Grade</i>
Yogyakarta, FirstMedia	Tes 1	Below Average
	Tes 2	Below Average
Yogyakarta, Biznet	Tes 1	Below Average
	Tes 2	Below Average
Yogyakarta, UIINET	Tes 1	Below Average
	Tes 2	Below Average

Berdasarkan Tabel 3.2 seluruh pengujian pada arsitektur *non-JAMstack* yang dilakukan dari tiga *server* origin berbeda di Yogyakarta yakni menggunakan ISP FirstMedia, Biznet, dan UIINET—menunjukkan *performance grade* “*Below Average*” pada Tes 1 dan Tes 2. Hasil ini menunjukkan bahwa waktu respon awal dari *server* ke browser membutuhkan durasi yang lebih lama dibandingkan dengan arsitektur *JAMstack*. Penyebab utama dari performa yang kurang optimal ini adalah metode penyajian konten statis yang bersumber dari satu *server*, sehingga terjadi penumpukan permintaan pada *server* penyaji, yang membuat waktu pemrosesan dan pengiriman byte pertama ke browser menjadi lebih tinggi. Selain itu, keterbatasan lokasi *server* yang hanya terpusat di satu wilayah tanpa dukungan CDN global membuat distribusi konten menjadi lebih lambat ketika diakses dari lokasi yang jauh baik secara geografis, maupun secara jaringan. Hal ini tentu menjadi tantangan dalam memberikan pengalaman pengguna yang baik,

terutama pada *website* pariwisata yang membutuhkan akses informasi kepada khalayak yang banyak dan tersebar.

Tabel 3.3 Pengujian regional TTFB *JAMstack*

<i>Server origin</i>					
CDN <i>Cloudflare</i>	Tes 1	<i>DNS</i>	<i>Connect</i>	<i>TLS</i>	<i>TTFB</i>
	Eropa	25 ms	4 ms	1 ms	70 ms
	Amerika	43 ms	13 ms	0 ms	113 ms
	Asia Pasifik	121 ms	21 ms	0 ms	228 ms
	Tes 2	<i>DNS</i>	<i>Connect</i>	<i>TLS</i>	<i>TTFB</i>
	Eropa	2 ms	4 ms	0 ms	54 ms
	Amerika	2 ms	15 ms	0 ms	71 ms
	Asia Pasifik	9 ms	16 ms	1 ms	92 ms

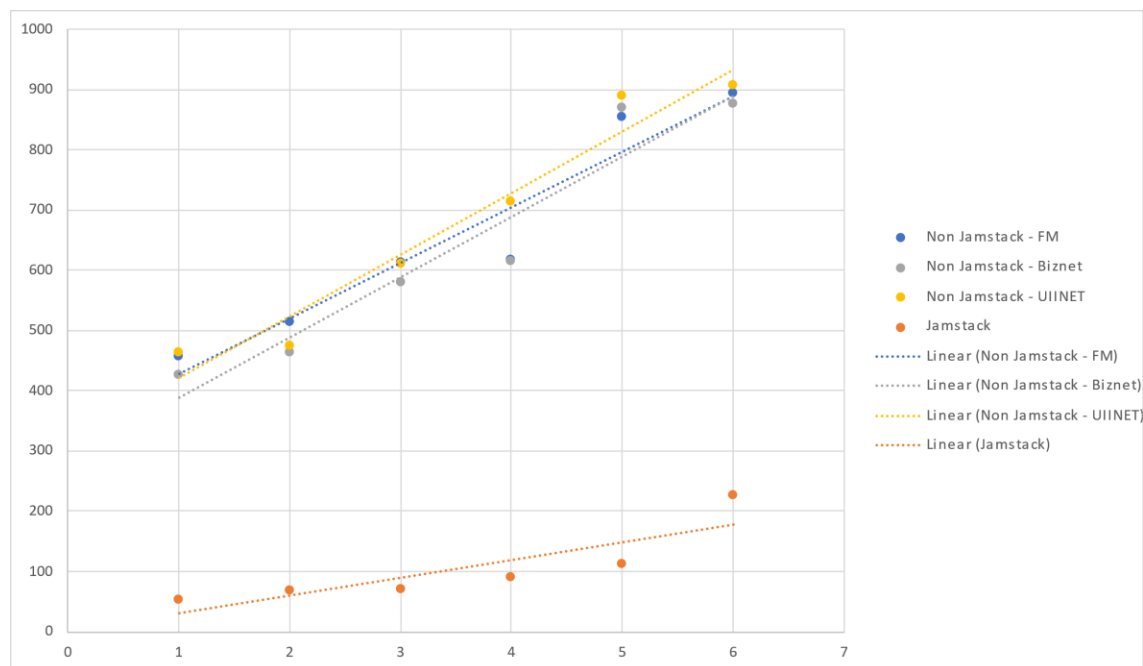
Tabel 3.4 Pengujian regional TTFB *non-JAMstack*

<i>Server origin</i>					
	Tes 1	<i>DNS</i>	<i>Connect</i>	<i>TLS</i>	<i>TTFB</i>
Yogyakarta, FirstMedia	Eropa	23 ms	4 ms	19 ms	613 ms
	Amerika	45 ms	14 ms	28 ms	895 ms
	Asia Pasifik	74 ms	22 ms	39 ms	514 ms
	Tes 2	<i>DNS</i>	<i>Connect</i>	<i>TLS</i>	<i>TTFB</i>
	Eropa	2 ms	4ms	19 ms	619 ms
	Amerika	5 ms	14 ms	29 ms	855 ms
	Asia Pasifik	17 ms	22 ms	38 ms	457 ms
Yogyakarta, Biznet	Tes 1	<i>DNS</i>	<i>Connect</i>	<i>TLS</i>	<i>TTFB</i>
	Eropa	29 ms	4 ms	20 ms	615 ms
	Amerika	32 ms	16 ms	33 ms	877 ms
	Asia Pasifik	70 ms	14 ms	30 ms	465 ms
	Tes 2	<i>DNS</i>	<i>Connect</i>	<i>TLS</i>	<i>TTFB</i>
	Eropa	2 ms	4 ms	19 ms	580 ms
	Amerika	2 ms	17 ms	34 ms	871 ms
	Asia Pasifik	26 ms	16 ms	31 ms	427 ms
Yogyakarta, UIINET	Tes 1	<i>DNS</i>	<i>Connect</i>	<i>TLS</i>	<i>TTFB</i>
	Eropa	25 ms	5 ms	21 ms	715 ms
	Amerika	30 ms	19 ms	35 ms	909 ms
	Asia Pasifik	75 ms	15 ms	31 ms	476 ms
	Tes 2	<i>DNS</i>	<i>Connect</i>	<i>TLS</i>	<i>TTFB</i>
	Eropa	2 ms	4 ms	19 ms	612 ms
	Amerika	3 ms	18 ms	36 ms	891 ms
	Asia Pasifik	43 ms	16 ms	32 ms	464 ms

Jika dilihat secara lebih rinci hasil pengujian rata-rata Time to First Byte (TTFB) yang dilakukan dari tiga wilayah utama Eropa dengan 10 titik lokasi, Amerika dengan 10 titik lokasi, dan Asia Pasifik dengan 12 titik lokasi yang tersedia di *SpeedVitals* terdapat perbedaan performa yang sangat signifikan antara pendekatan arsitektur *JAMstack* dan non-*JAMstack*. Pengujian ini menggunakan metrik DNS, *Connect*, TLS, dan TTFB untuk memberikan gambaran menyeluruh terhadap kecepatan respon awal *server* terhadap permintaan pengguna. *Website* dengan arsitektur *JAMstack* yang disajikan melalui CDN *Cloudflare* menunjukkan performa yang sangat baik yang ditampilkan pada Tabel 3.3. Pada Tes 1, wilayah Eropa mencatatkan TTFB sebesar 70 ms, Amerika 113 ms, dan Asia pasifik 228 ms. Sementara pada Tes 2, performa meningkat dengan TTFB Eropa sebesar 54 ms, Amerika 71 ms, dan Asia pasifik 92 ms. Performa yang baik ini didukung oleh jaringan global *Cloudflare* yang memiliki lebih dari 330 lokasi, memungkinkan distribusi konten statis yang cepat dan efisien (Herdes B, 2024). Meskipun pada Tes 1 wilayah Asia pasifik menunjukkan angka TTFB yang relatif tinggi, namun terjadi penurunan signifikan pada Tes 2, yang menunjukkan stabilitas dan keandalan CDN dalam skenario pengujian ulang.

Sebaliknya, *website* non-*JAMstack* yang dijalankan melalui *server* lokal di Yogyakarta menunjukkan performa TTFB yang lebih lambat dan bervariasi antar wilayah yang ditampilkan pada Tabel 3.4. Misalnya, pada koneksi menggunakan ISP FirstMedia, TTFB di wilayah Eropa dan Amerika selalu berada di atas 600 ms hingga hampir menyentuh 900 ms, sementara Asia pasifik menunjukkan performa yang lebih baik dengan nilai di kisaran 450–514 ms. Pola serupa juga terlihat pada koneksi Biznet dan UIINET, di mana Asia pasifik selalu menunjukkan nilai TTFB yang lebih rendah dibandingkan wilayah lain. Ini mengindikasikan bahwa kedekatan geografis dengan *server* lokal sangat memengaruhi performa akses awal.

Namun demikian, meskipun metrik DNS, *Connect*, dan TLS pada non-*JAMstack* tampak cukup kompetitif (misalnya DNS 2–5 ms, *Connect* 4–22 ms, dan TLS 19–38 ms), nilai TTFB tetap tinggi. Hal ini menandakan bahwa *bottleneck* terjadi setelah proses handshake TLS, yaitu saat *server* dinamis memproses dan mengirimkan konten ke browser. Tidak adanya distribusi konten melalui CDN membuat pengiriman data lebih lambat, terutama saat diakses dari wilayah yang jauh dari lokasi *server*.



Gambar 3. 132 Linear Regression *JAMstack* dan *Non-JAMstack*

Gambar 3. 132 menunjukkan data TTFB dari masing-masing regional—baik dari arsitektur *JAMstack* maupun non-*JAMstack*—dikelompokkan berdasarkan *server* origin dan diuji sebanyak masing-masing dua kali untuk tiap *server* origin pada tiga regional yang berbeda. Seluruh nilai TTFB tersebut kemudian diurutkan dari yang paling rendah hingga yang paling tinggi. Hasil pengurutan ini divisualisasikan dalam bentuk grafik *scatter plot*, dengan tambahangaris regresi linear pada masing-masing kategori untuk mempermudah analisis konsistensi performa. Garis regresi linear berfungsi untuk menunjukkan fluktuasi TTFB selama pengujian. Semakin landai garis tersebut, maka semakin stabil dan konsisten nilai TTFB yang dihasilkan oleh arsitektur tersebut. Dari grafik dapat diamati bahwa *JAMstack* menunjukkan garis regresi yang lebih landai dibandingkan dengan non-*JAMstack*, yang memiliki kemiringan lebih tajam. Hal ini menandakan bahwa TTFB pada *JAMstack* tidak hanya lebih cepat, tetapi juga lebih konsisten di berbagai regional, sedangkan TTFB pada non-*JAMstack* sangat berfluktuasi dan menunjukkan perbedaan yang mencolok antara TTFB terkecil dan terbesar. Temuan ini mendukung klaim bahwa arsitektur *JAMstack* menawarkan kecepatan dan konsistensi akses yang lebih baik secara global.

Nilai TTFB yang rendah memungkinkan browser mulai merender halaman lebih cepat, sehingga pengalaman pengguna menjadi lebih responsif dan nyaman. Selain itu, TTFB yang cepat berdampak langsung pada optimasi mesin pencari (SEO), karena *Google* dan mesin

pencari lainnya memprioritaskan situs dengan performa tinggi untuk mendukung proses *crawling*. Dengan demikian, optimalisasi TTFB tidak hanya meningkatkan kecepatan akses dari sisi pengguna, tetapi juga berkontribusi pada peningkatan visibilitas *website* dalam hasil pencarian serta kemudahan akses melalui berbagai browser.

Secara keseluruhan, hasil pengujian ini menunjukkan bahwa pendekatan *JAMstack* memiliki keunggulan signifikan dalam hal kecepatan respon awal *server* di berbagai lokasi global. Dengan TTFB yang konsisten berada dalam kategori "*Excellent*" hingga "*Good*", *JAMstack* terbukti lebih unggul dibandingkan non-*JAMstack* yang cenderung masuk kategori "*Below Average*" hingga "*Poor*", terutama saat diakses dari luar wilayah *server* lokal. Temuan ini memperkuat posisi *JAMstack* sebagai pilihan arsitektur web modern yang optimal dalam konteks performa awal akses pengguna.

BAB IV

REFLEKSI PELAKSANAAN MAGANG

4.1 Relevansi Akademik

Dalam pelaksanaan magang di PT Lumintu Logic, penulis terlibat langsung dalam pengembangan *Website* Radyakartiyasa menggunakan pendekatan Jamstack dengan framework *Astro*. *Website* Radyakartiyasa menyediakan konten wisata yang menampilkan destinasi dan acara yang diselenggarakan oleh Karaton Yogyakarta untuk pengunjung dapat akses lewat browser. Selama magang, penulis mengaplikasikan teori-teori yang telah dipelajari di bootcamp yang disediakan oleh Lumintu logic mengenai arsitektur *website*, pengembangan *website* statis (SSG), serta teknologi *Content Delivery Network* (CDN) dan *API*.

Sebagai contoh, teori yang diajarkan mengenai arsitektur Jamstack lebih menekankan pada pemisahan yang tegas antara frontend dan backend dengan penggunaan *API* untuk mengambil data. Dalam prakteknya, penulis terlibat dalam integrasi *API* CMS untuk mengambil konten dan menampilkan data di *website*. Namun, meskipun teori mengedepankan penggunaan *API* secara konsisten, dalam implementasi nyata terdapat tantangan dalam mengelola dan memastikan data yang ditarik dari *API* dapat diproses dengan cepat. Misalnya, penulis harus menambahkan lapisan pengolahan data (seperti pembersihan tag HTML menggunakan `strip_tags` dan optimasi gambar dengan `file_handle`) sebelum menampilkan konten di halaman. Proses ini memerlukan perhatian ekstra pada pengolahan data dinamis yang tidak dibahas secara mendalam dalam teori.

Selain itu, teori mengenai penggunaan *Static Site Generation* (SSG) untuk meningkatkan performa *website* dapat diobservasi dalam hasil pengembangan *website* radyakartiyasa yang mengutamakan kecepatan loading halaman. Implementasi SSG dengan framework *astro* terbukti meningkatkan performa *website* secara signifikan, terutama dalam hal kecepatan akses melalui CDN. Salah satu contoh spesifik adalah ketika halaman “destinasi” dioptimalkan dengan menggunakan gambar berformat `.webp` yang lebih ringan, yang mempercepat waktu muat halaman di seluruh dunia melalui CDN. Hal ini sejalan dengan teori bahwa SSG dapat mengurangi beban *server* dan meningkatkan kecepatan pengiriman konten ke user.

Namun, di lapangan, terdapat tantangan tambahan terkait seperti integrasi data dari berbagai sumber yang tidak selalu seragam dari CMS. Meskipun dalam teori penggunaan *API* untuk mengakses data dari CMS dan layanan eksternal sangat efisien, dalam praktiknya penulis

menghadapi masalah terkait ketidaksesuaian format dan sktruktur data yang diterima. Sebagai contoh, penulis sering kali mendapati bahwa beberapa data dari CMS memiliki format yang tidak konsisten, seperti tanggal yang tidak terkstruktur dengan baik atau data yang tidak terhubung dengan benar antar koleksi. Hal ini menyebabkan penulis perlu menambahkan lapisan pemrosesan tambahan, seperti validasi dan pembersihan data, untuk memastikan bahwa data yang ditampilkan di *website* sesuai dengan ekspektasi. Salah satu langkah yang diambil adalah menggunakan metode `strip_tags` untuk menghapus elemen html yang tidak diinginkan dari deskripsi dan `date_format` untuk memastikan tanggal ditampilkan dalam format yang seragam dan mudah dipahami.

Terakhir pengelolaan file media yang cukup besar, seperti gambar dan video. Meskipun dalam teori, penggunaan SSG dinilai cukup efisien, penulis harus memitigasi tantangan dalam pengelolaan file gambar dari CMS. Sebagai contoh, penulis harus menggunakan metode `file_handle` untuk memastikan bahwa gambar yang diambil dari CMS dapat dioptimalkan dengan mengubah gambar ke format *.webp* untuk efisiensi.

4.2 Pembelajaran Magang

4.2.1 Manfaat

Manfaat yang dapat dirasakan oleh penulis dalam menerapkan pendekatan *jamstack* dengan *framework astro* adalah mempermudah pengembangan *website* Radyakartiyasa. Dengan menggunakan pendekatan *jamstack*, *website* dapat dibangun dengan lebih efisien karena *astro* memungkinkan penggunaan *static site generation* yang menghasilkan halaman-halaman statis yang dapat dimuat lebih cepat di browser user.

Dengan menggunakan *astro* yang berbasis pada *jamstack* (*JavaScript*, *API*, Markup), pengembangan *website* juga menjadi lebih terstruktur dan modular. Pendekatan ini memungkinkan developer untuk mengelola berbagai bagian dari situs (seperti elemen statis dan dinamis) secara terpisah dan lebih mudah, sehingga pengembangan dan pemeliharaan situs menjadi lebih mudah dan efisien.

4.2.2 Kendala, Hambatan, dan Tantangan

Kendala dan hambatan yang dirasakan penulis selama mengerjakan proyek situs Radyakartiyasa dengan pendekatan *jamstack* menggunakan *framework astro* adalah ketika data yang diterima dari *API* harus disesuaikan untuk optimasi *website*. Salah satu tantangannya adalah memastikan bahwa data yang diambil dari *API* dapat diproses dengan efisien dan

ditampilkan dengan cepat di halaman web. Beberapa data dari *API* CMS terkadang memiliki format yang kurang sesuai, atau memerlukan pengolahan tambahan sebelum bisa diintegrasikan dengan baik ke dalam struktur halaman statis.

Selain itu penggunaan modul-modul *astro* yang belum pernah diajarkan sebelumnya menjadi tantangan tersendiri karena memerlukan waktu tambahan untuk memahami cara kerja modul-modul tersebut agar dapat diimplementasikan dengan efektif dalam proyek.

Kendala lainnya adalah dalam integrasi beberapa *API* eksternal yang tidak selalu memiliki dokumentasi yang jelas atau membutuhkan penyesuaian dalam pengambilan data untuk dapat sesuai dengan struktur dan kebutuhan situs. Hal ini memerlukan waktu ekstra untuk memahami dan menyesuaikan integrasi, serta memastikan bahwa data yang diterima dapat digunakan tanpa menurunkan performa situs.

Terakhir penulis juga harus dapat beradaptasi dengan cepat terhadap ritme kerja yang dinamis dan mempelajari hal-hal baru dalam waktu singkat. Penulis juga perlu menjaga fokus dalam bekerja agar dapat menyelesaikan pekerjaan sesuai dengan waktu yang ditetapkan.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil perancangan dan pengembangan website Radyakartiyasa dengan pendekatan jamstack menggunakan framework *astro*, dapat disimpulkan bahwa:

1. Pengembangan *website* Radyakartiyasa dengan pendekatan *jamstack* menggunakan *framework astro* menghasilkan *website* yang cepat, responsive, dan efisien.
2. Pengembangan *website* Radyakartiyasa dilakukan melalui beberapa tahap yang telah disesuaikan dengan kebutuhan. Proses dimulai dengan diskusi bersama semua pihak terkait mengenai alasan pembuatan *website*, dilanjutkan dengan pembuatan *mockup* desain situs, penentuan tim yang akan mengerjakan proyek ini, dan akhirnya penerapan *framework Astro* untuk mengembangkan *website* tersebut sesuai dengan persetujuan dari semua pihak yang terlibat.
3. *Website* sudah siap dengan desain yang telah diterjemahkan menjadi kode *frontend*, kemudian dimasukkan ke dalam struktur *framework Astro*. Selanjutnya, integrasi dengan *API* CMS dilakukan, di mana variabel *API* diterapkan pada konten untuk mendinamiskan tampilan sesuai dengan bahasa dan konten yang tersedia. Selain itu, konten yang ada di *Astro* sudah dioptimalkan untuk memastikan performa yang maksimal.
4. Website radyakartiyasa telah berhasil dijalankan menggunakan perintah *npm run dev*, *npm run preview* dan *npm run build*, yang menandakan bahwa situs sudah siap untuk ditampilkan ke user.
5. Berdasarkan perkembangan *website* Radyakartiyasa sudah siap untuk dimasukkan ke tahap staging deployment. Pada tahap ini, *website* akan ditempatkan di *server staging* untuk dilakukan pengujian lebih lanjut lalu dipindahkan ke tahap selanjutnya yaitu tahap *server preview* dan produksi.
6. Berdasarkan pengujian perbandingan antara *JAMstack* dan non *JAMstack* dengan parameter TTFB menggunakan *source code* yang sama, diperoleh hasil bahwa *JAMstack* menunjukkan performa yang lebih stabil dan unggul. Sebaliknya, non-*JAMstack* menghasilkan nilai yang sangat fluktuatif dan kurang konsisten pada setiap pengujian.

5.2 Saran

Terdapat beberapa saran yang dapat dilakukan pada penelitian berikutnya agar dapat memaksimalkan hasil pengembangan *website* Radyakartiyasa dengan pendekatan *jamstack* menggunakan *framework astro*:

1. Salah satu aspek *website* radyakartiyasa yang belum sepenuhnya tercapai adalah aspek interaktivitas *website*, terutama dalam form penyampaian pesan dari masyarakat. Saat ini, fitur formulir yang tersedia belum terhubung secara langsung ke sistem CMS. Untuk itu, pada pengembangan berikutnya disarankan agar sistem pengumpulan data masyarakat dan formulir interaktif dapat diintegrasikan secara penuh dengan CMS yang digunakan, agar pengelolaan masukan dari user dapat dilakukan secara otomatis dan efisien.
2. Tampilan *website* radyakartiyasa yang telah dikembangkan masih memiliki ruang penyempurnaan dari sisi tampilan visual dan navigasi, terutama untuk memudahkan masyarakat yang belum terbiasa dengan penggunaan teknologi digital. Oleh karena itu, disarankan agar pada pengembangan selanjutnya dilakukan evaluasi antarmuka dengan melibatkan user langsung, guna meningkatkan aksesibilitas, keterbacaan, dan kemudahan interaksi di setiap halaman *website*.
3. *Website* saat ini belum dilengkapi dengan fitur pencarian yang memudahkan user menemukan informasi destinasi, acara, atau konten lainnya secara cepat. Oleh karena itu, disarankan agar pada versi selanjutnya ditambahkan fitur *search* yang terintegrasi dengan data yang ada, agar user dapat mengakses informasi secara lebih efisien tanpa harus menelusuri halaman satu persatu.

DAFTAR PUSTAKA

- Abdiati, A. D., Setiawan, S., & Supendar, H. (2021). Pemilihan web browser pada mobile menggunakan metode analytical hierachy process. *Jurnal Infortech*, 3(1), 26–32.
- Akgül, Y. (2021). *App and Website Accessibility Developments and Compliance Strategies*. IGI Global.
- Akib, E. (2020). Pariwisata Dalam Tinjauan Pendidikan: Studi Menuju Era Revolusi Industri. *Pusaka: Journal of Tourism, Hospitality, Travel and Business Event*, 1–8.
- Astuti, F. K., & Agustina, D. S. (2022). Membangun Website MTS Negeri 01 OKU Timur Menggunakan Php dan Mysql. *JIK: Jurnal Informatika Dan Komputer*, 13(1), 7–14.
- Bakhtiar, Y., Dosen, S., Tinggi, S., Komunikasi, I., & Tarakanita -Jakarta, S. (2019). DIGITALISASI ARSIP UNTUK EFISIENSI PENYIMPANAN DAN AKSESIBILITAS. *Jurnal Administrasi Dan Kesekretarian*, (Vol. 4).
- Bratslavsky, P. (2025, March 27). *What is Astro and Why Use it*. Strapi. <https://strapi.io/blog/what-is-astro>
- Buyya, R., Pathan, M., & Vakali, A. (2008). *Content delivery networks* (Vol. 9). Springer Science & Business Media.
- Camden, Raymond., & Rinaldi, Brian. (2022). *The Jamstack Book: Beyond Static Sites with JavaScript, APIs, and Markup*. Simon and Schuster.
- Farid, A. (2024, February 12). *CDN Adalah : Pengertian, Cara Kerja, Manfaat dan Kelebihan*. Exabytes. <https://www.exabytes.co.id/blog/cdn-adalah/>
- Fayock C. (2022). *The Jamstack in 2022: Why (and How) to Get Started*. SNIPCART. <https://snipcart.com/blog/jamstack>
- Hanafi, F. (2022, September 6). *Mengenal apa itu SSG (Static Site Generator) - CODEPOLITAN*. CODEPOLITAN. <https://www.codepolitan.com/blog/mengenal-apa-itu-ssg-static-site-generator/>
- Hanafi, F. (2023). *Mengenal JAMStack (Javascript,API,Markup) - CODEPOLITAN*. CODEPOLITAN. <https://www.codepolitan.com/blog/mengenal-jamstack-javascriptapimarkup/>
- Hanggara, Y. (2018). *Web Aplikasi Lambat? Kenali apa itu Time to First Byte (TTFB) | by Yoga Hanggara | Medium*. Medium. <https://medium.com/@yohangdev/web-aplikasi-kurang-responsif-kenali-apa-itu-time-to-first-byte-ttfb-156154a680b2>

- Herdes B, T. M. (2024). *The backbone behind Cloudflare's Connectivity Cloud*.
CLOUDFLARE. <https://blog.cloudflare.com/backbone2024/>
- Hidayah, A., Yani, A., & others. (2019). Membangun Website SMA PGRI Gunung Raya Ranau Menggunakan PHP dan MySQL. *JTIM: Jurnal Teknik Informatika Mahakarya*, 2(2), 41–52.
- Hidayatullah, A. R., & Arief, M. R. (2016). Analisis dan Perancangan Sistem Informasi Manajemen Zakat Berbasis Client Server pada Badan Amil Zakat Masjid Agung Baitul Qadim Loloan Timur. *Semnasteknomedia Online*, 4(1), 1–4.
- Ihsan, I., Lesmidayarti, D., Hidayati, Q., & Nugroho, T. R. (2023). Perancangan Infrastruktur dan Implementasi Web Server Untuk Website Sekolah Sebagai Media Informasi dan Komunikasi di SMP PJHI Balikpapan. *JTT (Jurnal Teknologi Terpadu)*, 11(1), 66–72.
- Iqbal. (2009). Rekayasa Content Management System (Cms) Joomla Berbasis Open Source Untuk Pengembangan Sistem Informasi Berbasis Online. *Jurnal Informatika Ahmad Dahlan*, 3(1), 103430.
- James H. (2024). *What GTmetrix Did That Made Me Switch to SpeedVitals*. LINUXBLOG.IO. <https://linuxblog.io/gtmetrix-made-me-switch-to-speedvitals/>
- Juno. (2021). *Mengenal Empat Kawedanan Hageng Keraton Yogyakarta dan Tugasnya - Baca Jogja*. Bacajogja.Id. <https://bacajogja.id/2021/06/28/mengenal-empat-kawedanan-hageng-keraton-yogyakarta-dan-tugasnya/>
- Kumawat K. (2021a). *Introducing SpeedVitals Alpha Version! | SpeedVitals Blog*. SpeedVitals. <https://speedvitals.com/blog/introducing-speedvitals/>
- Kumawat K. (2021b). *Introducing SpeedVitals TTFB Test | SpeedVitals Blog*. SpeedVitals. <https://speedvitals.com/blog/introducing-speedvitals-ttfb-test/>
- Kurniawan, M., Saputro, H., & others. (2022). Membangun Digital Library Sma Xeverius Baturaja. *JTIM: Jurnal Teknik Informatika Mahakarya*, 5(1), 9–16.
- Kurniawan, R. (2021). *Apa Itu Static Site Generator? | Ruang Developer*. Blog. <https://blog.ruangdeveloper.com/apa-itu-static-site-generator/>
- Lau, L. (2023, November). *Time to First Byte (TTFB): What Is It and How To Improve*. <https://www.valido.ai/en/ttfb-how-to-improve/>
- Lestari, W., & Susanto, A. (2017). Analisis performa website ISI Surakarta dan Universitas Diponegoro menggunakan automated software testing GTMetrix. *Jurnal Sistem Informasi Dan Sistem Komputer*, 2(3), 1–8.

- Maryani, I., Ishaq, A., & Mulyadi, D. S. (2018). Sistem Informasi Pemesanan Minuman Berbasis Client Server Pada Kampung Dahar Purwokerto. *Evolusi: Jurnal Sains Dan Manajemen*, 6(2).
- Mildenberger, T. (2024). *Static Websites vs. Dynamic Websites: A Comparison* | Contabo. CONTABO. <https://contabo.com/blog/static-websites-vs-dynamic-websites-a-comparison/>
- Mirza, F. (2024, November 9). *Apa yang Dimaksud Static Site Generator, dan Mengapa Penting?* | UtaHost Blog. Utahost. <https://ultahost.com/blog/id/apa-yang-dimaksud-static-site-generator-apakah-perlu-menggunakannya-dalam-pembuatan-website/>
- Nazarina, P. P., Rosdilillah, S. M., & Fatah, A. S. (2024). KEAMANAN DATA PADA SITUS WEB MENGGUNAKAN PROTOKOL HTTP/HTTPS DENGAN SSL. *Jurnal Ilmiah Multidisiplin Terpadu*, 8(6).
- Newson, K. (2017). Tools and workflows for Collaborating on static website projects. *Code{4}lib*.
- Nurhidayat T. (2022, August 13). *Apa itu Jamstack? Bagaimana Cara Kerja dan Kelebihannya?* Dewabiz. <https://dewabiz.com/apa-itu-jamstack/>
- Nurlailah, E., & Wardani, K. R. N. (2023). Perancangan Website Sebagai Media Informasi Dan Promosi Oleh-Oleh Khas Kota Pagaralam. *JUPI (Jurnal Ilmiah Penelitian Dan Pembelajaran Informatika)*, 8(4), 1175–1185.
- Paakkanen, J. (2023). Upcoming JavaScript web frameworks and their techniques. *Bachelor's Thesis, Aalto University, School Of Science, Computer Science, Dec.*
- Patel M. (2025). *Quic.cloud CDN Review: Is It the Best CDN for WordPress?* Mpateldigital.Com. <https://mpateldigital.com/quic-cloud-cdn-review/>
- Pradana, E. R. (2025, February 17). *Mengenal JAMstack: Masa Depan Pengembangan Web?* - Feducation. FEDU.CO.ID. <https://fedu.co.id/2025/02/17/mengenal-jamstack-masa-depan-pengembangan-web/>
- Pradana, G. Y. K. (2019). Sosiologi pariwisata. In *STPBI PRESS*. Denpasar: STPBI Press.
- Pranata, D., Hamdani, H., & Khairina, D. M. (2015). Rancang Bangun Website Jurnal Ilmiah Bidang Komputer (Studi Kasus: Program Studi Ilmu Komputer Universitas Mulawarman). *Informatika Mulawarman: Jurnal Ilmiah Ilmu Komputer*, 10(2), 25–29.
- Rusdi, M., Kadarsih, & Murti, W. (2022). Membuat Aplikasi Penjualan pada Cv. Sumber Bakti Mandiri Berbasis Website Menggunakan Php dan Mysql. *JIK: Jurnal Informatika Dan Komputer*, 13(2), 51–56.

- Saraswati, N. W. S. (2011). Rancang Bangun Sistem Penjualan Online Berbasis Web Dinamis. *Jurnal Teknologi Dan Sistem Informasi*, 2(2), 135–144.
- Siambaton, M. Z., & Fakhriza, M. (2016). Aplikasi content management system (cms) pada Joomla untuk membuat web service. *InfoTekJar (Jurnal Nasional Informatika Dan Teknologi Jaringan)*, 1(1), 11–13.
- Silalahi, F. E. S., Prayitno, M., Gambetta, W., Amhar, F., Wijaya, M., & Rachma, T. R. N. (2021). Identifikasi Pertumbuhan Data dan Penggunaan Application Performance Index (APDEX) Score dalam Penilaian Kinerja Aplikasi Penyaji Informasi Geospasial. *Jurnal Pekommas*, 6(1), 95–104.
- Suryanto, S. (2015). Implementasi Clustering Database Server Menggunakan Pgcluster untuk Optimalisasi Kinerja Sistem Basis Data. *Jurnal Khatulistiwa Informatika*, 1(1), 134–143.
- Susanto, B., Filiana, A., & Oentoro, K. (2022). Pengembangan Situs Badan Promosi Pariwisata Kota Yogyakarta. *Kumawula: Jurnal Pengabdian Kepada Masyarakat*, (Vol 5)(2), 268–277.
- Tandra, R., Siallagan, D. Y., & Prestoroika, E. (2025). INTEROPERABILITAS SISTEM INFORMASI PROFIL JURUSAN BERBASIS HEADLESS CONTENT MANAGEMENT SYSTEM WEBSITE JURUSAN. *Jurnal Informasi Interaktif Vol*, 10(1).
- Utomo, P., & Falahah. (2020). Building Serverless Website on GitHub Pages. *IOP Conference Series: Materials Science and Engineering*, 879(1), 12077. <https://doi.org/10.1088/1757-899X/879/1/012077>
- Woro, W. (2010). *Panduan Aplikatif dan Solusi Membuat Aplikasi Client Server dengan Visual Basic* 2008. Penerbit Andi.
- Zabar, A. A., & Novianto, F. (2015). Keamanan Http Dan Https Berbasis Web Menggunakan Sistem Operasi Kali Linux. *Komputa: Jurnal Ilmiah Komputer Dan Informatika*, 4(2), 69–74.

LAMPIRAN

A. Proyek *Astro*

