SANTIKA *Seminar Nasional Informatika Bela Negara (SANTIKA)*
ISSN (Online) 2747-0563
Volume 5 Tahun 2025

Implementasi RESTful API Web Wedding Organizer Ruang Hati menggunakan Laravel dan VueJs

Albi Akhsanul Hakim[1], Dwijo Utomo Rahino Putro[2*], Leon Dawundaru Pramudyo[3], Kalfin Syah Kilau Mayya[4], Maulana Aditya Furqon Aryanda[5], Fawwaz Ali Akbar[6]

[1,2,3,4,5] Informatika, Universitas Pembangunan Nasional "Veteran" Jawa Timur
[2]22081010220@student.upnjatim.ac.id
[3]22081010221@student.upnjatim.ac.id
[4]22081010251@student.upnjatim.ac.id
[5]22081010310@student.upnjatim.ac.id
[6]fawwaz_ali.fik@upnjatim.ac.id
*Corresponding author email: [1]22081010194@student.upnjatim.ac.id

***Abstrak*— Perkembangan teknologi digital mendorong berbagai sektor industri untuk mengadopsi solusi berbasis sistem informasi guna meningkatkan efisiensi operasional. Salah satu sektor yang turut merasakan kebutuhan ini adalah industri jasa, termasuk layanan *Wedding Organizer*, yang memiliki kompleksitas tinggi dalam pengelolaan vendor, anggaran, dan komunikasi antar pihak. Digitalisasi sistem perencanaan pernikahan menjadi kunci untuk menyederhanakan proses yang sebelumnya manual dan kurang terintegrasi. Penelitian ini bertujuan untuk merancang dan mengimplementasikan sistem aplikasi berbasis *website* untuk layanan *Wedding Organizer* Ruang Hati dengan menggunakan arsitektur RESTful *API* guna meningkatkan efisiensi manajemen vendor dan anggaran secara terstruktur. Aplikasi dikembangkan dengan pendekatan arsitektur terpisah (*decoupled architecture*) antara *backend* dan *frontend*, di mana Laravel digunakan untuk membangun RESTful *API* sebagai layanan backend, dan VueJs sebagai *frontend* yang dinamis dan responsif. Metodologi pengembangan sistem mengikuti model *SDLC Waterfall*, dimulai dari tahap analisis kebutuhan, perancangan sistem, implementasi, pengujian, hingga pemeliharaan. Sistem ini dirancang untuk memudahkan pengguna dalam melakukan perencanaan pernikahan secara digital melalui fitur manajemen vendor, anggaran, *timeline*, serta komunikasi antar pihak. Uji coba dilakukan dengan metode *black-box testing* untuk memastikan setiap fitur dapat berjalan sesuai dengan fungsinya tanpa menguji struktur internal kode. Hasil pengujian menunjukkan bahwa sistem mampu menjalankan seluruh fungsionalitas secara optimal dan responsif, dengan antarmuka pengguna yang intuitif serta integrasi data yang stabil antara *backend* dan *frontend*. Implementasi arsitektur RESTful *API* terbukti meningkatkan skalabilitas serta memudahkan pengelolaan data secara modular. Aplikasi ini memberikan solusi digital yang efisien, fleksibel, dan *user-friendly* dalam mendukung layanan *Wedding Organizer* secara menyeluruh.**

Kata Kunci— **Wedding Organizer, RESTful API, Manajemen Vendor, Manajemen Anggaran, Arsitektur Terpisah.**

I. PENDAHULUAN

Perkembangan pesat industri pernikahan modern telah membuka peluang besar bagi jasa *Wedding Organizer* (*WO*) yang tidak hanya berfokus pada kesiapan acara, tetapi juga pada efisiensi manajemen vendor dan anggaran. Menurut Sinaga dan Sembiring, pasar *WO* kian menjanjikan karena perubahan gaya hidup millennial yang mengutamakan praktis dan instan dalam penyelenggaraan pernikahan [1]. Salah satu pendekatan yang kini banyak diterapkan dalam pengembangan sistem modern adalah penggunaan RESTful *API*, yakni sebuah standar arsitektur layanan berbasis *HTTP* yang memungkinkan pertukaran data secara terstruktur, fleksibel, dan efisien antar komponen atau sistem yang berbeda. Namun, masih banyak *WO* yang mengandalkan proses manual atau sistem monolitik, sehingga kurang responsif terhadap kebutuhan *real-time* dan kolaborasi lintas pihak antar vendor.

Untuk menjawab tantangan tersebut, implementasi arsitektur RESTful *API* dengan pemisahan komponen *frontend* dan *backend* menjadi solusi ideal. Salah satu pendekatan yang kini banyak diterapkan dalam pengembangan sistem modern adalah penggunaan RESTful *API*, yakni sebuah standar arsitektur layanan berbasis *HTTP* yang memungkinkan pertukaran data secara terstruktur, fleksibel, dan efisien antar komponen atau sistem yang berbeda. RESTful *API* memungkinkan komunikasi terstruktur antara VueJs sebagai antarmuka pengguna dan Laravel sebagai penyedia layanan server, sehingga setiap modul, termasuk manajemen vendor dan manajemen anggaran agar dapat diakses secara mandiri dan diintegrasikan dengan mudah ke sistem lain [2]. Pendekatan arsitektur terpisah ini juga mendukung skalabilitas, keamanan, dan pemeliharaan kode yang lebih terorganisir.

Dalam pengembangannya, sistem ini dibangun menggunakan metode *SDLC* (*Software Development Life Cycle*), yaitu pendekatan sistematis dan bertahap yang terdiri atas beberapa fase berurutan mulai dari analisis kebutuhan hingga pemeliharaan, sehingga memastikan setiap fitur dikembangkan secara terstruktur dan terdokumentasi dengan baik [3].

Dalam kerangka *SDLC Waterfall*, tahap *Requirement* melakukan identifikasi kebutuhan fungsional seperti pendaftaran vendor, alokasi anggaran, dan pencatatan transaksi. Tahap desain menghasilkan dokumentasi *UML* dan rancangan basis data yang memisahkan entitas Vendor, Anggaran, dan *Order-Wedding*. Fase implementasi kemudian mengembangkan RESTful endpoint di Laravel, sementara VueJs memanfaatkan *JSON payload* untuk *dynamic data binding* dan *user experience* yang responsive [4].

Verifikasi sistem menggunakan *black-box testing* menjamin bahwa tiap *endpoint API* berfungsi sesuai spesifikasi tanpa menelaah struktur internal. Metode ini menguji skenario

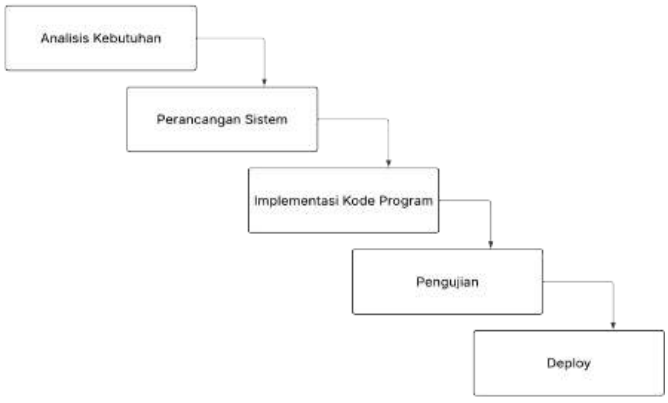
ISSN (Online) 2747-0563
Volume 5 Tahun 2025

autentikasi, *CRUD* (*Create*, *Read*, *Update*, dan *Delete*) vendor, yakni operasi dasar dalam mengolah data, serta kalkulasi dan pelaporan anggaran, memastikan keakuratan data dan kehandalan antarmuka terpisah [5]. Hasil uji coba menunjukkan integrasi mulus antara modul–modul terpisah, memudahkan tim manajemen klien dan keuangan dalam memonitor status vendor dan realisasi anggaran per paket layanan [6].

Dengan menerapkan arsitektur RESTful *API* dan *SDLC Waterfall*, aplikasi Ruang Hati tidak hanya memenuhi kebutuhan dasar *WO*, seperti pembuatan paket dan pengelolaan vendor, namun juga memberikan fleksibilitas yang tinggi untuk pengembangan fitur lanjut serta integrasi dengan pihak ketiga. Model pemisahan arsitektur ini sekaligus mencerminkan praktik terbaik dalam informatika modern, di mana layanan mikro dan antarmuka dinamis menjadi kunci keberhasilan aplikasi *web* skala perusahaan.

II. METODE PENELITIAN

Dalam penelitian ini, penulis mengadopsi model *Waterfall* sebagai kerangka kerja *SDLC* karena sifatnya yang linear, terstruktur, dan telah terbukti menjadi metode unggulan dalam perancangan sistem informasi pemesanan berbasis *web* seperti *Wedding Organizer* [6]. Tahapan model *Waterfall* yang diterapkan meliputi *requirement* untuk mengidentifikasi kebutuhan fungsional, termasuk manajemen vendor dan anggaran serta desain dengan pembuatan *UML* dan *ERD* untuk memetakan arsitektur terpisah, implementation berupa pengembangan RESTful *API* menggunakan Laravel dan integrasi VueJs pada *frontend*, verifikasi melalui pengujian *black-box* untuk memastikan setiap *endpoint API*. Mulai dari *CRUD* vendor hingga perhitungan anggaran berfungsi sesuai spesifikasi, serta *maintenance* untuk perbaikan dan penyempurnaan pasca-implementasi [5]. Rangkaian tahapan penelitian ini secara visual dapat dilihat pada Gbr. 1.



Gbr. 1 Pendekatan *Waterfall* Dalam Mengembangkan Aplikasi *Web.*

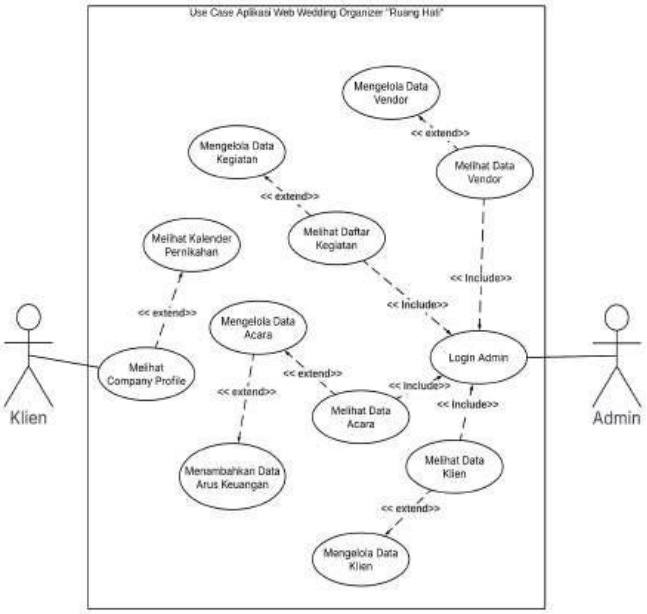
A. Analisis Kebutuhan

Tahap awal pengembangan dimulai dengan wawancara langsung dan observasi untuk menggali alur kerja, tantangan, dan kebutuhan mitra *Wedding Organizer* Ruang Hati, sebagaimana dilakukan oleh Pakaja et al. dalam pengembangan sistem serupa [4]. Narasumber meliputi pemilik usaha, staf operasional, dan koordinator acara. Dari proses ini teridentifikasi fitur-fitur utama yang harus disediakan: manajemen jadwal acara, profil perusahaan, daftar vendor, pengelolaan anggaran, manajemen staf, serta dashboard administrator. Kebutuhan manajemen vendor selaras dengan peran *Wedding Organizer* yang memfasilitasi koordinasi dengan pihak *catering*, dekorasi, fotografi, dan tata rias [1], sementara kebutuhan pengelolaan anggaran dan laporan keuangan terintegrasi telah diakui sebagai fitur krusial dalam kajian literature berbasis *web* [6].

B. Perancangan Sistem

Perancangan sistem bertujuan menyusun desain menyeluruh yang mencakup pemodelan fungsional dan struktural guna memberikan gambaran solid sebelum implementasi. Pemodelan fungsional dilakukan dengan *Unified Modeling Language* (*UML*), meliputi *use case* diagram untuk memetakan interaksi pengguna, *activity* diagram untuk alur kerja sistem, *sequence* diagram untuk urutan pesan antar objek, serta *class* diagram untuk struktur kelas dan relasinya [2]. Pemodelan data menggunakan *Entity Relationship Diagram* (*ERD*) untuk mendefinisikan entitas seperti klien, vendor, anggaran, dan pesanan beserta atribut dan relasi antar entitas tersebut [5]. Selain itu, diagram konteks dan *Data Flow Diagram* (*DFD*) level 0/1 disusun untuk memvisualisasikan arus data dan proses utama dalam aplikasi [4]. Terakhir, pembuatan flowchart aplikasi memudahkan pemahaman jalur eksekusi fitur secara *end-to-end*, menjamin konsistensi desain dan kesiapan tim pengembang dalam tahap implementasi.

1) Use Case Diagram

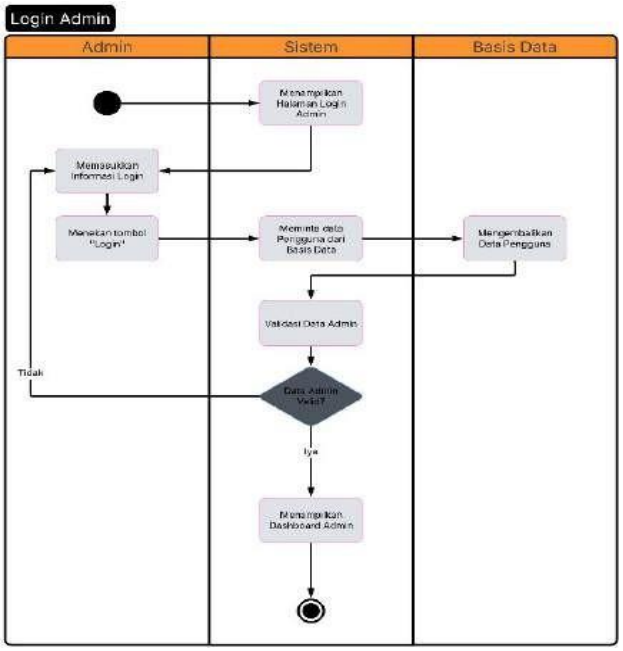


ISSN (Online) 2747-0563
Volume 5 Tahun 2025

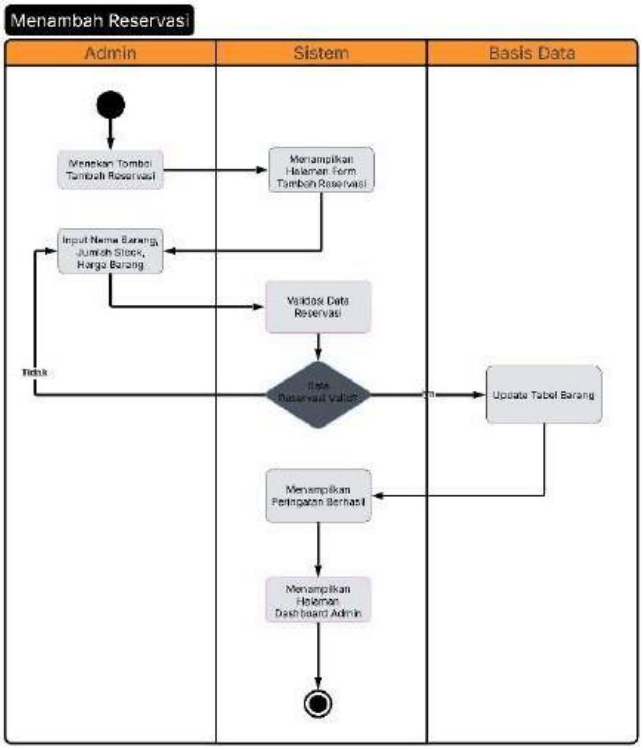
Gbr. 2 *Use Case* Diagram Interaksi Aktor Klien dan Admin dengan Sistem.

2) Activity Diagram

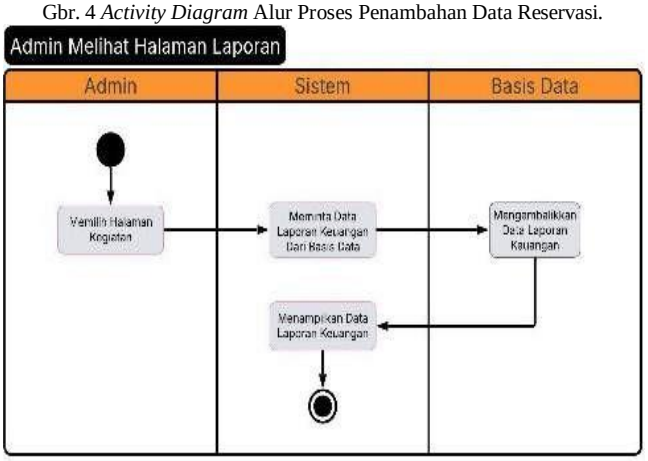
Activity Diagram adalah representasi visual yang menggambarkan alur kerja sistem secara terperinci, memodelkan urutan aktivitas yang dijalankan oleh sistem dalam suatu proses bisnis (bukan tindakan aktor), dan umum digunakan dalam perancangan aplikasi *web* berbasis arsitektur terpisah seperti pada pengembangan *Wedding Organizer* "Ruang Hati" [2], [5]. Gambar 3, 4, dan 5 secara berurutan menggambarkan proses saat admin melakukan *login*, serta menambahkan data reservasi baru, dan melihat data kegiatan.



Gbr. 3 *Activity Diagram* Alur Proses *Login* Admin.



Gbr. 4 *Activity Diagram* Alur Proses Penambahan Data Reservasi.



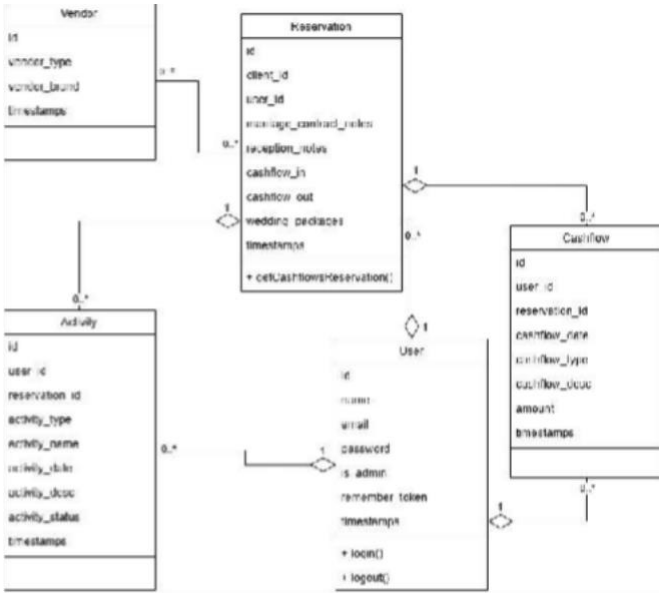
Gbr. 5 *Activity Diagram* Alur Proses Melihat Laporan Keuangan.

3) Class Diagram

Class diagram memodelkan struktur statis sistem dengan menampilkan kelas-kelas utama, atribut, metode, serta relasi seperti asosiasi, agregasi, dan pewarisan. Dalam konteks perancangan aplikasi *website Wedding Organizer* Ruang Hati, *class* diagram dibangun berdasarkan entitas domain seperti Vendor, Paket, Anggaran, dan *User* untuk mengilustrasikan tanggung jawab tiap kelas dan interaksi antar objek [2]. Pendekatan ini sejalan dengan praktik pemodelan dalam kerangka *Waterfall* yang menekankan verifikasi desain sebelum implementasi, sehingga memudahkan pengembangan RESTful API di Laravel dan integrasi komponen VueJs pada *frontend* [4]. Gambar 6 berikut memperlihatkan struktur kelas,

ISSN (Online) 2747-0563
Volume 5 Tahun 2025

atribut, metode, serta hubungan antar kelas yang digunakan dalam program penelitian ini.



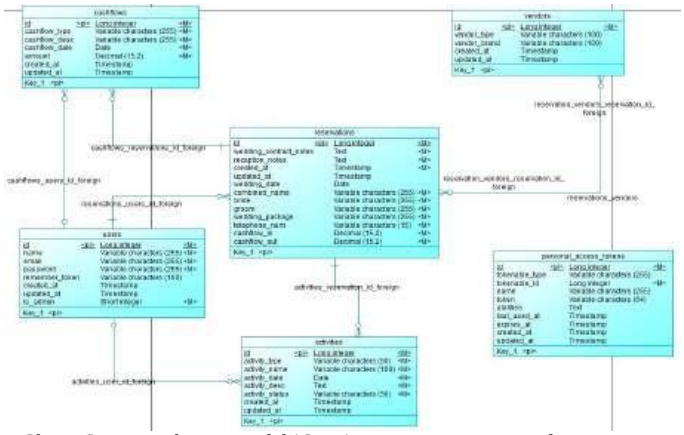
Gbr. 6 *Class Diagram* Struktur Logis dan Relasi Antar Objek Sistem.

4) Perancangan Basis Data

Basis data merupakan sekumpulan data yang disimpan secara terstruktur dalam suatu perangkat, sehingga memungkinkan pengelolaan dan pengambilan informasi secara efisien. Dalam proses perancangan basis data, terdapat dua model utama yang digunakan, yaitu:

- *Conceptual Data Model (CDM)*

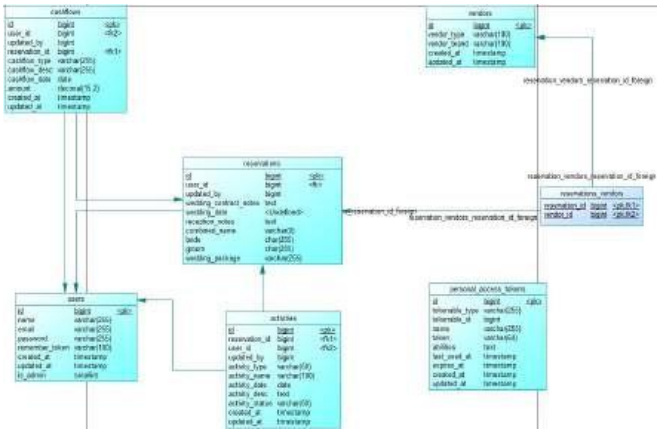
Conceptual Data Model (*CDM*) adalah Model data tingkat konseptual yang memetakan entitas, atribut, dan hubungan antar-entitas secara logis sesuai kebutuhan bisnis untuk membentuk kerangka skema basis data sebelum implementasi teknis dimana dengan tahap ini memastikan integritas logis dan kelengkapan relasi antar-komponen sistem.



Gbr. 7 *Conceptual Data Model* (*CDM*) Rancangan Konseptual Basis Data.

- *Physical Data Model (PDM)*

Physical Data Model (*PDM*) merupakan transformasi *CDM* ke skema fisik pada sistem manajemen basis data, di mana tabel, kolom, tipe data, serta indeks dirancang sesuai kebutuhan penyimpanan dan optimasi akses di MySQL, sehingga struktur basis data siap dipakai dalam implementasi nyata [7].



Gbr. 8 *Physical Data Model* (*PDM*) Rancangan Fisik dan Implementasi Basis Data.

C. Implementasi Kode Program

Aplikasi ini dikembangkan menggunakan Visual Studio Code sebagai *IDE* pilihan, menggantikan *Sublime Text* yang sebelumnya umum dipakai dalam penelitian serupa [4]. Pada sisi *frontend*, Vue.js dipilih untuk membangun antarmuka yang responsif dan interaktif, sejalan dengan implementasi Vue.js pada sistem pergudangan berbasis Laravel dan Vue.js yang telah terbukti mendukung dinamika tampilan [5]. Di *backend*, Laravel versi 10 digunakan sebagai *framework* utama karena arsitektur *MVC*-nya yang terstruktur, aman, dan mudah diperluas serupa dengan keberhasilan Laravel dalam sistem informasi rental *Wedding Organizer* [2]. Untuk penyimpanan data, MySQL dipakai sebagai *DBMS* handal, dengan desain skema relasional yang mendukung operasi *CRUD* pada RESTful *API* secara efisien. Aplikasi ini mencakup berbagai fitur utama, di antaranya:

1) *Form Input Data Reservasi:* Admin dapat menambahkan data reservasi baru yang mencakup informasi seperti nama klien, paket pernikahan yang dipilih, serta catatan tambahan terkait kebutuhan atau preferensi khusus dalam acara pernikahan.
2) *Form Menambahkan Arus Keuangan:* Selama proses persiapan pernikahan, admin dapat mencatat data arus keuangan, baik berupa pendapatan maupun pengeluaran. Informasi yang dimasukkan meliputi jenis transaksi, jumlah uang, deskripsi transaksi, serta identitas pengguna yang mencatat data tersebut, sehingga tercipta transparansi dan akuntabilitas.
3) *Kalender Kegiatan:* Pengguna biasa dapat mengakses kalender untuk melihat jadwal acara pernikahan yang telah dipesan, sehingga memudahkan dalam memilih tanggal

ISSN (Online) 2747-0563
Volume 5 Tahun 2025

yang masih tersedia. Sementara itu, admin dapat menggunakan kalender ini untuk melihat jadwal lengkap, termasuk kegiatan internal seperti rapat staf, penyewaan lokasi, dan pertemuan dengan vendor seperti katering atau dekorasi.

D. Pengujian

Setelah tahap implementasi selesai, pengujian aplikasi dilakukan menggunakan *Black-box Testing* untuk memverifikasi seluruh fungsionalitas RESTful *API* tanpa melihat struktur internal kode [2], [5]. Pengujian ini mencakup skenario operasi *CRUD* pada modul manajemen vendor dan anggaran, autentikasi serta alur data antar frontend VueJs dan backend Laravel, guna memastikan *output API* sesuai dengan kebutuhan dan ekspektasi pengguna [4].

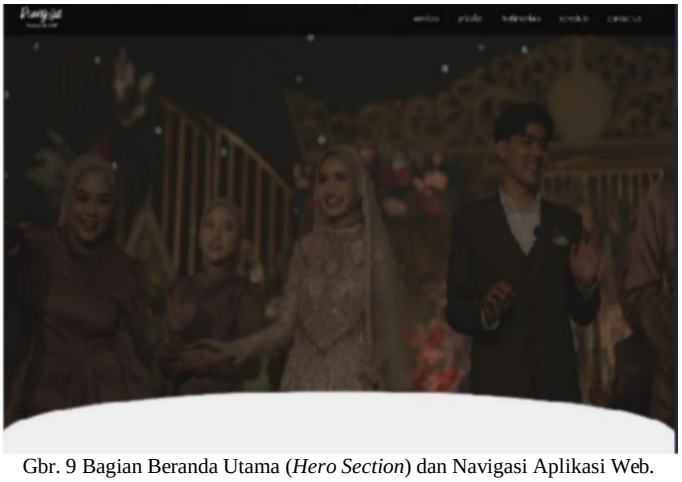
III. HASIL DAN PEMBAHASAN

A. Hasil Pembuatan Aplikasi Web

Hasil pembuatan yang telah dikerjakan adalah sebuah aplikasi web bernama "Ruang Hati", sebuah *platform* digital yang dirancang khusus untuk *Wedding Organizer*. Aplikasi web ini bertujuan untuk menyajikan informasi layanan, daftar harga, testimoni, dan jadwal ketersediaan secara komprehensif kepada calon klien. Sesuai dengan riset pengembangan yang telah dilakukan sebelumnya, aplikasi web "Ruang Hati" telah sukses mengimplementasikan pendekatan *Model-View-Controller* (*MVC*) untuk memastikan struktur *web* yang rapi, modular, dan mudah dikelola. Pendekatan ini memisahkan logika bisnis (*Model*), presentasi data (*View*), dan interaksi pengguna (*Controller*), sehingga meningkatkan skalabilitas dan pemeliharaan kode. Selain itu, aplikasi ini memanfaatkan Vue.js sebagai *framework front-end* untuk membangun antarmuka pengguna yang dinamis dan responsif, memberikan pengalaman navigasi yang mulus dan interaktif bagi pengguna. Berikut ini merupakan hasil pengerjaan dari aplikasi *web* "Ruang Hati"

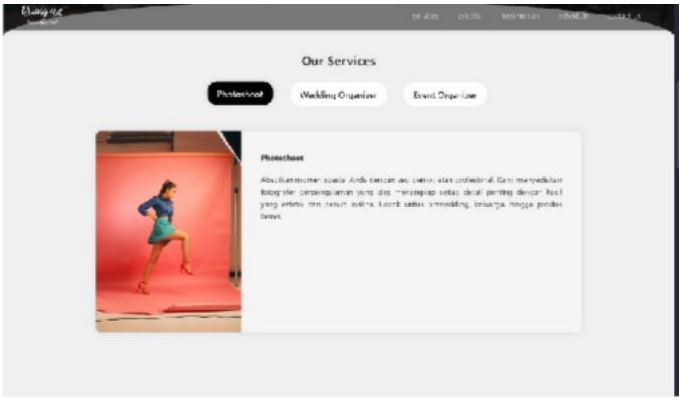
1) Halaman Utama Aplikasi Web Ruang Hati (Landing Page)

Aplikasi *web* "Ruang Hati" dirancang sebagai sebuah landing page tunggal yang informatif dan mudah diakses, menyajikan seluruh informasi penting dalam satu alur gulir vertikal. Desain ini memastikan pengguna dapat dengan cepat memahami cakupan layanan yang ditawarkan tanpa perlu berpindah halaman. Pada Gbr. 9, 10, 11, 12, 13, dan 14 terlihat tampilan komprehensif halaman utama yang mencakup seluruh bagian inti aplikasi, mulai dari bagian hero yang menarik perhatian, bagian layanan yang merinci penawaran, daftar harga yang transparan, testimoni dari klien yang puas, hingga bagian jadwal untuk memeriksa ketersediaan. Struktur ini dirancang untuk memandu pengguna melalui informasi secara logis dan intuitif.



Gbr. 9 Bagian Beranda Utama (*Hero Section*) dan Navigasi Aplikasi Web.

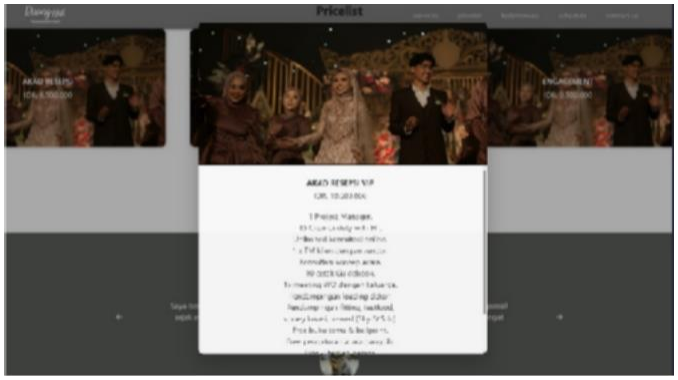
Bagian paling atas dari halaman utama berfungsi sebagai titik masuk visual pertama bagi pengguna, dirancang untuk memberikan kesan profesional dan menarik. Pada Gbr. 9, ditampilkan bagian beranda utama atau *hero section* yang menonjolkan identitas merek "Ruang Hati" melalui logo yang jelas dan navigasi utama yang terstruktur. Bilah navigasi ini mencakup tautan ke "*services*", "pricelist", "testimonials", "*schedule*", dan "*contact us*", memungkinkan pengguna untuk melompat langsung ke bagian yang diminati tanpa harus menggulir secara manual. Desain ini menekankan kemudahan akses dan efisiensi navigasi.



Gbr. 10 Bagian "Layanan Kami" dengan Detail Layanan "*Photoshoot*".

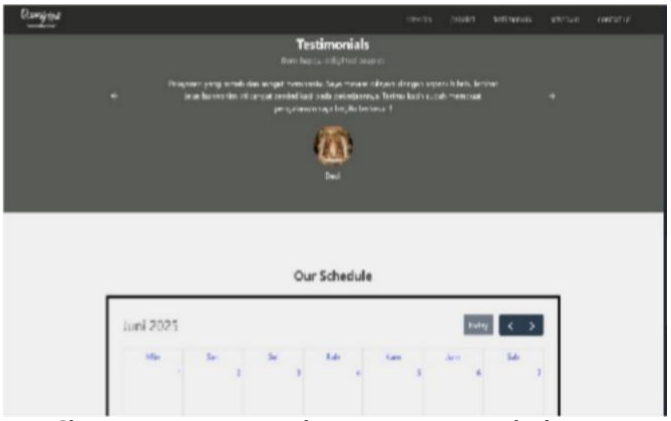
Bagian ini secara spesifik merinci jenis-jenis layanan yang disediakan oleh "Ruang Hati", memberikan informasi detail untuk setiap kategori. Pada Gbr. 10, terlihat bagian "Our *Services*" yang mengadopsi desain berbasis tab. Saat tab "*Photoshoot*" diaktifkan, deskripsi layanan yang relevan beserta contoh visual ditampilkan. Fitur tab ini memungkinkan penyajian informasi yang padat namun terorganisir, di mana pengguna dapat beralih antara "*Photoshoot*", "Wedding *Organizer*", dan "*Event Organizer*" untuk memahami cakupan masing-masing layanan secara terperinci. Pendekatan ini meminimalkan kebutuhan akan halaman terpisah untuk setiap layanan, menjaga landing page tetap ringkas.

ISSN (Online) 2747-0563
Volume 5 Tahun 2025



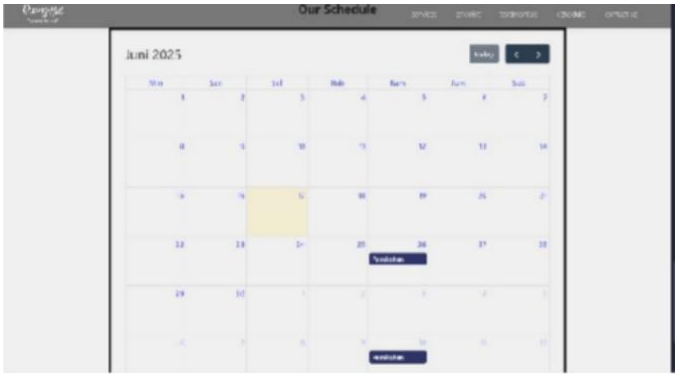
Gbr. 11 Tampilan Interaktif Bagian "Daftar Harga" dengan Jendela *Pop-up* Detail Paket.

Transparansi harga adalah kunci dalam layanan *Wedding Organizer*, dan bagian ini dirancang untuk menyajikan informasi paket secara jelas dan interaktif. Pada Gbr. 11, ditampilkan bagian "*Pricelist*" yang dilengkapi dengan fitur interaktif berupa jendela *pop-up* (modal). Ketika pengguna memilih salah satu paket harga, seperti "AKAD RESEPSI VIP", sebuah *pop-up* akan muncul menampilkan detail paket secara lengkap, termasuk harga (IDR 10.500.000) dan fasilitas yang didapatkan (misalnya, *Project Manager*, 15 *Crew*, konsultasi tanpa batas, dll.). Fitur ini memungkinkan pengguna untuk mendapatkan informasi mendalam tentang paket tanpa meninggalkan halaman utama, meningkatkan pengalaman pengguna dengan menyediakan detail yang relevan secara kontekstual.



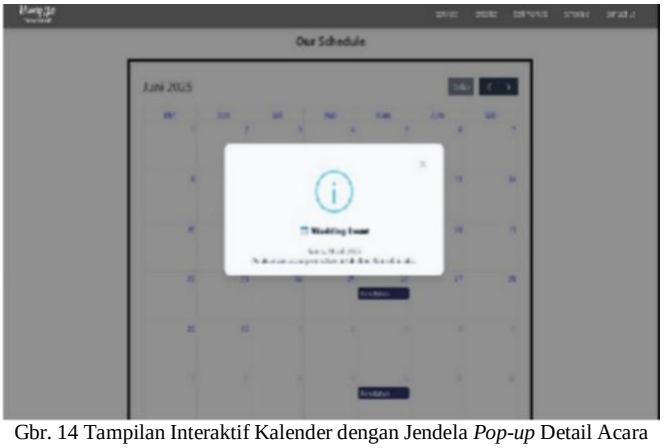
Gbr. 12 Bagian "Testimoni" dan Pratinjau Bagian "Jadwal Kami".

Kepercayaan klien dibangun melalui ulasan positif, dan bagian testimoni berfungsi untuk menampilkan umpan balik dari klien yang telah menggunakan jasa "Ruang Hati". Pada Gbr. 12, terlihat bagian "*Testimonials*" yang menampilkan ulasan dari klien, seperti "Dedi", yang memberikan kesan positif terhadap pelayanan. Bagian ini dirancang untuk membangun kredibilitas dan meyakinkan calon klien. Di bawahnya, pratinjau awal dari bagian "*Our Schedule*" juga mulai terlihat, mengindikasikan transisi ke informasi ketersediaan.



Gbr. 13 Bagian "Jadwal Kami" Menampilkan Kalender dengan Tanggal Acara yang Terisi.

Ketersediaan adalah informasi krusial bagi calon klien, dan bagian jadwal menyajikan kalender interaktif untuk memudahkan pengecekan. Pada Gbr. 13, ditampilkan bagian "*Our Schedule*" yang menyajikan kalender untuk bulan Juni 2025. Kalender ini secara visual menandai tanggal-tanggal yang sudah terisi dengan acara, seperti "Pernikahan" pada tanggal 25 Juni. Fitur ini memungkinkan pengguna untuk dengan cepat mengidentifikasi tanggal-tanggal yang tersedia atau yang sudah dipesan, membantu mereka dalam perencanaan awal acara.



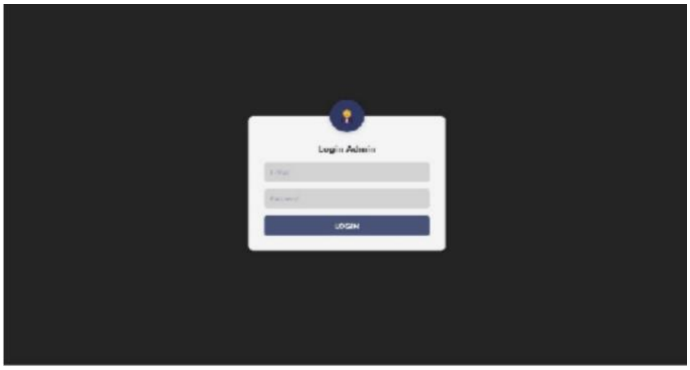
Gbr. 14 Tampilan Interaktif Kalender dengan Jendela *Pop-up* Detail Acara Pernikahan.

Untuk memberikan informasi lebih lanjut tentang acara yang sudah dipesan, kalender dilengkapi dengan fungsionalitas interaktif. Pada Gbr. 14, terlihat bagian "Jadwal Kami" dengan kalender yang sama, namun kali ini menampilkan *jendela pop-up* "*Wedding Event*" ketika tanggal yang sudah dipesan dipilih. *Pop-up* ini memberikan detail spesifik acara, seperti tanggal (Kamis, 10 Juli 2025) dan nama klien ("*client* Rano dan Julia"). Interaksi ini memberikan transparansi penuh kepada pengguna mengenai jadwal yang ada, sekaligus menunjukkan tingkat profesionalisme dalam pengelolaan acara.

2) Halaman Dasbor Administratif

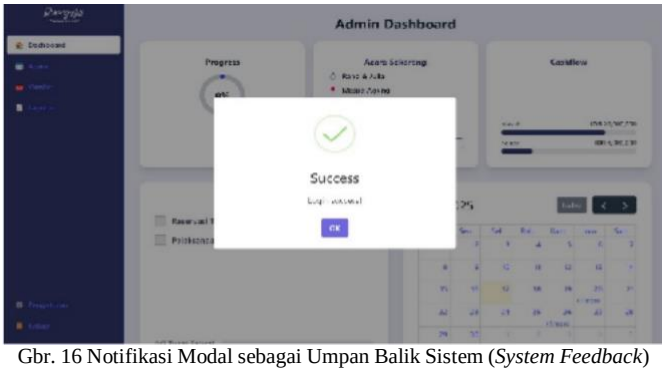
ISSN (Online) 2747-0563
Volume 5 Tahun 2025

Halaman Dasbor Administratif berfungsi sebagai pusat kendali operasional dan manajerial dari aplikasi web "Ruang Hati". Berbeda dengan *landing page* yang bersifat publik, area ini merupakan antarmuka multifaset yang aman, dirancang untuk menyajikan data secara terintegrasi dan memfasilitasi manajemen acara yang efisien. Pada Gbr. 15, 16, 17, dan 18, diperlihatkan alur kerja administratif secara lengkap, mulai dari gerbang otentikasi yang aman, dasbor utama yang menyajikan ringkasan informasi kritis seperti progres acara dan *cashflow*, hingga halaman laporan keuangan yang terperinci. Struktur ini dirancang secara strategis untuk memberikan administrator pandangan menyeluruh (*at-a-glance*) sekaligus alat yang kuat untuk pengambilan keputusan berbasis data.



Gbr. 15 Antarmuka Halaman *Login* untuk Otentikasi Administrator Sistem.

Pada titik akses awal, sistem menyajikan halaman *login* yang didedikasikan untuk administrator (Gbr. 15). Antarmuka ini dirancang dengan pendekatan minimalis yang disengaja, menampilkan formulir esensial yang hanya meminta *input* kredensial berupa "*E-mail*" dan "*Password*". Desain yang terfokus ini secara efektif menghilangkan distraksi dan mempertegas fungsi utamanya sebagai gerbang keamanan untuk melindungi data operasional dan finansial yang sensitif. Setelah proses otentikasi berhasil, sistem secara proaktif memberikan umpan balik visual yang jelas melalui sebuah notifikasi modal (Gbr. 16). Notifikasi bertuliskan "*Success*" ini muncul di atas dasbor yang sudah termuat di latar belakang, berfungsi sebagai konfirmasi eksplisit bahwa tindakan *login* telah berhasil. Mekanisme ini sangat penting untuk pengalaman pengguna karena memberikan kepastian dan mengorientasikan admin ke dalam lingkungan kerja sistem.



Gbr. 16 Notifikasi Modal sebagai Umpan Balik Sistem (*System Feedback*) atas Keberhasilan *Login*.

Setelah berhasil *login*, pengguna akan disambut oleh Dasbor Utama (Gbr. 17), yang berfungsi sebagai pusat komando dan kesadaran situasional (*situational awareness*) bagi administrator. Dasbor ini dirancang dengan cerdas menggunakan tata letak berbasis kartu (*card-based layout*) untuk menyajikan beragam informasi penting secara simultan dan terorganisir:

- *Informasi Kritis: Widget* seperti "Acara Sekarang" menampilkan detail acara yang paling relevan (nama klien, lokasi, tanggal), sementara ringkasan "*Cashflow*" memberikan gambaran finansial sekilas.
- *Manajemen Proyek:* Modul "Jadwal Kegiatan" berfungsi sebagai daftar periksa (*checklist*) untuk melacak tugas-tugas utama, yang kemajuannya divisualisasikan secara agregat melalui *widget* "*Progress*".
- *Penjadwalan Visual:* Integrasi kalender bulanan memberikan konteks temporal, menyoroti tanggal-tanggal penting yang terkait dengan jadwal kegiatan. Secara keseluruhan, dasbor ini memungkinkan admin untuk dengan cepat memahami status setiap proyek dari berbagai sudut pandang secara operasional, finansial, dan jadwal dalam satu layar terpadu.

Gbr. 17 Tampilan Utama Dasbor Admin yang Mengintegrasikan Ringkasan Informasi Kritis (Progress, Acara, *Cashflow*) dengan Modul Manajemen Tugas (Jadwal Kegiatan) dan Kalender Visual.

Untuk analisis finansial yang lebih mendalam, administrator dapat menavigasi ke halaman "Laporan" (Gbr. 18). Halaman ini menyediakan pandangan multi-faset terhadap kesehatan

ISSN (Online) 2747-0563
Volume 5 Tahun 2025

keuangan proyek, yang disajikan melalui tiga komponen utama yang saling berhubungan:

- *Visualisasi Tren:* "Grafik *Cashflow*" menggunakan diagram batang untuk memvisualisasikan aliran pendapatan dan pengeluaran dari waktu ke waktu, memungkinkan identifikasi pola dan tren finansial dengan cepat.
- *Ringkasan Agregat:* Modul "Ringkasan *Cashflow*" menyajikan angka-angka definitif untuk total pemasukan, total pengeluaran, dan saldo akhir, memberikan gambaran profitabilitas yang jelas dan terukur.
- *Audit Terperinci*: Tabel "Riwayat Transaksi" menawarkan data granular dari setiap transaksi yang terjadi, lengkap dengan tanggal, deskripsi, dan jumlah. Penggunaan kode warna secara intuitif membedakan antara pemasukan dan pengeluaran, memfasilitasi proses audit dan verifikasi data. Halaman ini secara efektif mengubah data transaksional mentah menjadi intelijen bisnis yang dapat ditindaklanjuti.

Gbr. 18 Halaman Laporan Keuangan yang Menyajikan Analisis Finansial Multi-Faset melalui Visualisasi Data (Grafik *Cashflow*), Ringkasan Agregat, dan Riwayat Transaksi yang Terperinci.

B. Pengujian Aplikasi Web

Pada tahapan pengujian, aplikasi "Ruang Hati" yang dibangun dengan arsitektur RESTful *API* diuji secara menyeluruh menggunakan *Black-Box Testing*. Metode ini memungkinkan evaluasi fungsionalitas endpoint tanpa melihat struktur kode, sehingga sesuai dengan praktik pengamanan *API* berbasis *REST* yang banyak diterapkan untuk mengidentifikasi celah autentikasi dan otorisasi [8].

Skenario pengujian mencakup verifikasi operasi *CRUD* pada modul Manajemen Vendor dan Manajemen Anggaran melalui Postman dimana sebagai *API client*, dengan berbagai kombinasi *HTTP method* (*GET*, *POST*, *PUT*, *DELETE*) dan *payload JSON*. Selain memastikan bahwa respon *API* (status code dan message) sesuai dengan kontrak yang telah ditetapkan, pengujian juga memeriksa konsistensi alur data antara *frontend* VueJs dan *backend* Laravel, memastikan setiap *request* diterjemahkan dan diproses sesuai ekspektasi tanpa kebocoran data antarpengguna [9]. Pada subbab selanjutnya akan dipaparkan hasil detail dan analisis setiap skenario pengujian.

1) Uji Fungsional Black Box

Dalam pengembangan aplikasi *Wedding Organizer* "Ruang Hati" yang mengintegrasikan Laravel sebagai *backend* dan VueJs sebagai *frontend*, pengujian fungsional *black box* memegang peranan krusial untuk memastikan setiap *endpoint API* berfungsi sesuai spesifikasi. Bagian ini menyajikan hasil uji fungsional *black box* yang dilakukan pada berbagai modul *API*, termasuk otentikasi, pengelolaan kegiatan, acara (reservasi), data vendor, dan arus keuangan. Pengujian ini dirancang untuk memvalidasi perilaku eksternal *API* tanpa memerlukan pengetahuan tentang struktur internal kode, berfokus pada apakah setiap permintaan *API* menghasilkan respon yang diharapkan, baik itu keberhasilan (misalnya, kode status *HTTP 200 OK*, *201 Created*) maupun penanganan kesalahan (misalnya, *401 Unauthorized*, *422 Unprocessable Content*, *404 Not Found*). Hasil yang disajikan dalam tabel-tabel berikut menjadi bukti konkret dari keandalan dan kepatuhan *API* terhadap persyaratan fungsional yang telah ditetapkan, sekaligus menjadi demonstrasi praktis pemahaman konsep Pemrograman *API*.

- *Pengujian API Autentikasi*

NO.	*URL*	*METHOD*	*TESTCASE*	HASIL YANG DIHARAPKAN	HASIL PENGUJIAN
1.	http://127.0.0.1:8000/api/login	*POST*	Mengirim data valid	*200 OK*	Berhasil
2.			Mengirim data tidak valid	*401 Unauthorized*	Berhasil
3.	http://127.0.0.1:8000/api/logout	*POST*	*Request* Dengan *API Key*	*200 OK*	Berhasil
4.			*Request* Tanpa *API Key*	*401 Unauthorized*	Berhasil

- *Pengujian API Kegiatan*

NO.	*URL*	*METHOD*	*TESTCASE*	HASIL YANG DIHARAPKAN	HASIL PENGUJIAN
1.	http://127.0.0.1:8000/api/activities	*GET*	*Request* tanpa parameter	*200 OK*	Berhasil
2.	http://127.0.0.1:8000/api/all-	*GET*	*Request* Dengan *API Key*	*200 OK*	Berhasil

ISSN (Online) 2747-0563
Volume 5 Tahun 2025

3.	activities		*Request* Tanpa *API Key*	*401 Unauthorized*	Berhasil
4.	http://127.0.0.1:8000/api/activities/{id}	*GET*	*Request* Dengan *API Key*	*200 OK*	Berhasil
5.			*Request* Tanpa *API Key*	*401 Unauthorized*	Berhasil
6.	http://127.0.0.1:8000/api/activities	*POST*	Mengirim data valid	*201 Created*	Berhasil
7.			Mengirim data tidak valid	*422 Unprocessable Content*	Berhasil
8.			*Request* Dengan *API Key*	*200 OK*	Berhasil
9.			*Request* Tanpa *API Key*	*401 Unauthorized*	Berhasil
10.	http://127.0.0.1:8000/api/activities/{id}	*PUT*	Mengirim data valid	*200 OK*	Berhasil
11.			Mengirim data tidak valid	*422 Unprocessable Content*	Berhasil
12.			*Request* Dengan *API Key*	*200 OK*	Berhasil
13.			*Request* Tanpa *API Key*	*401 Unauthorized*	Berhasil
14.	http://127.0.0.1:8000/api/activities/{id}	*DELETE*	*Request* Dengan *API Key*	*200 OK*	Berhasil
15.			*Request* Tanpa *API Key*	*404 Not Found*	Berhasil

- *Pengujian API Acara*

NO.	*URL*	*METHOD*	*TESTCASE*	HASIL YANG DIHARAPKAN	HASIL PENGUJIAN
1.	http://127.0.0.1:800	*GET*	*Request* Dengan *API*	*200 OK*	Berhasil
	0/api/reservations		*Key*		
2.			*Request* Tanpa *API Key*	*401 Unauthorized*	Berhasil
3.	http://127.0.0.1:8000/api/upcoming	*GET*	*Request* Dengan *API Key*	*200 OK*	Berhasil
4.			*Request* Tanpa *API Key*	*401 Unauthorized*	Berhasil
5.	http://127.0.0.1:8000/api/reservations/{id}/cashflows	*GET*	*Request* Dengan *API Key*	*200 OK*	Berhasil
6.			*Request* Tanpa *API Key*	*401 Unauthorized*	Berhasil
7.	http://127.0.0.1:8000/api/reservations	*POST*	Mengirim data valid	*201 Created*	Berhasil
8.			Mengirim data tidak valid	*422 Unprocessable Content*	Berhasil
9.			*Request* Dengan *API Key*	*200 OK*	Berhasil
10.			*Request* Tanpa *API Key*	*401 Unauthorized*	Berhasil
11.	http://127.0.0.1:8000/api/reservations/{id}	*PUT*	Mengirim data valid	*200 OK*	Berhasil
12.			Mengirim data tidak valid	*422 Unprocessable Content*	Berhasil
13.			*Request* Dengan *API Key*	*200 OK*	Berhasil
14.			*Request* Tanpa *API Key*	*401 Unauthorized*	Berhasil
15.	http://127.0.0.1:8000/api/reservations/{id}	*DELETE*	*Request* Dengan *API Key*	*404 Not Found*	Berhasil
16.			*Request* Tanpa *API Key*	*404 Not Found*	Berhasil

- *Pengujian API Vendor*

ISSN (Online) 2747-0563
Volume 5 Tahun 2025

NO.	*URL*	*METHOD*	*TESTCASE*	HASIL YANG DIHARAPKAN	HASIL PENGUJIAN
1.	http://127.0.0.1:8000/api/vendors	*GET*	*Request* Dengan *API Key*	*200 OK*	Berhasil
2.			*Request* Tanpa *API Key*	*401 Unauthorized*	Berhasil
3.	http://127.0.0.1:8000/api/vendors/{id}	*GET*	*Request* Dengan *API Key*	*200 OK*	Berhasil
4.			*Request* Tanpa *API Key*	*401 Unauthorized*	Berhasil
5.	http://127.0.0.1:8000/api/vendors	*POST*	Mengirim data valid	*201 Created*	Berhasil
6.			Mengirim data tidak valid	*422 Unprocessable Content*	Berhasil
7.			*Request* Dengan *API Key*	*200 OK*	Berhasil
8.			*Request* Tanpa *API Key*	*401 Unauthorized*	Berhasil
9.	http://127.0.0.1:8000/api/vendors/{id}	*PUT*	Mengirim data valid	*200 OK*	Berhasil
10.			Mengirim data tidak valid	*422 Unprocessable Content*	Berhasil
11.			*Request* Dengan *API Key*	*200 OK*	Berhasil
12.			*Request* Tanpa *API Key*	*401 Unauthorized*	Berhasil
13.	http://127.0.0.1:8000/api/vendors/{id}	*DELETE*	*Request* Dengan *API Key*	*404 Not Found*	Berhasil
14.			*Request* Tanpa *API Key*	*404 Not Found*	Berhasil

- *Pengujian API Arus Keuangan*

NO.	*URL*	*METHOD*	*TESTCASE*	HASIL YANG DIHARAPKAN	HASIL PENGUJIAN
1.	http://127.0.0.1:8000/api/cashflows	*GET*	*Request* Dengan *API Key*	*200 OK*	Berhasil
2.			*Request* Tanpa *API Key*	*401 Unauthorized*	Berhasil
3.	http://127.0.0.1:8000/api/recentCashflows	*GET*	*Request* Dengan *API Key*	*200 OK*	Berhasil
4.			*Request* Tanpa *API Key*	*401 Unauthorized*	Berhasil
7.	http://127.0.0.1:8000/api/cashflows	*POST*	Mengirim data valid	*201 Created*	Berhasil
8.			Mengirim data tidak valid	*422 Unprocessable Content*	Berhasil
9.			*Request* Dengan *API Key*	*200 OK*	Berhasil
10.			*Request* Tanpa *API Key*	*401 Unauthorized*	Berhasil
11.	http://127.0.0.1:8000/api/cashflows/{id}	*PUT*	Mengirim data valid	*200 OK*	Berhasil
12.			Mengirim data tidak valid	*422 Unprocessable Content*	Berhasil
13.			*Request* Dengan *API Key*	*200 OK*	Berhasil
14.			*Request Tanpa API Key*	*401 Unauthorized*	Berhasil
15.	http://127.0.0.1:8000/api/cashflows/{id}	*DELETE*	*Request* Dengan *API Key*	*404 Not Found*	Berhasil
16.			*Request* Tanpa *API Key*	*404 Not Found*	Berhasil

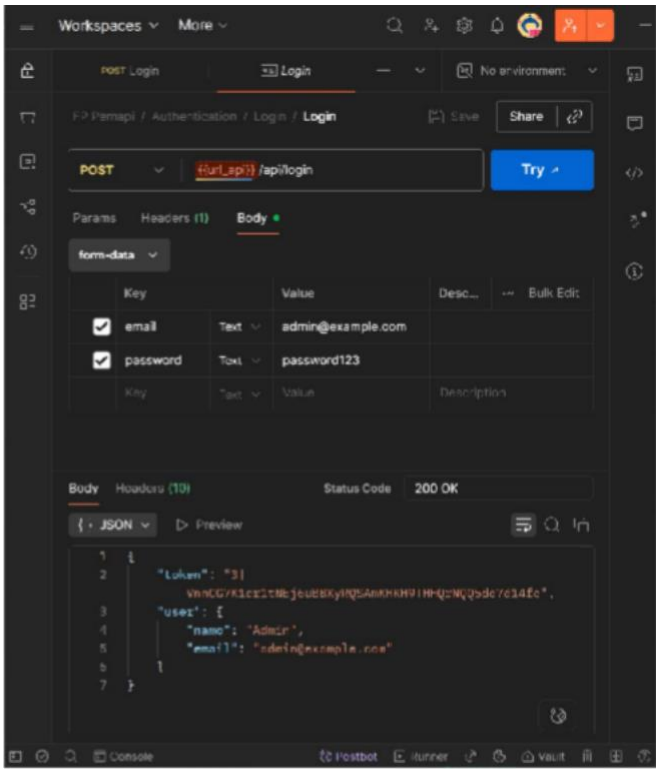
2) *Pengujian RESTful API dengan Postman*

ISSN (Online) 2747-0563
Volume 5 Tahun 2025

Bagian ini menyajikan rincian setiap pengujian *endpoint API* yang dilakukan menggunakan Postman, mengilustrasikan parameter permintaan, *header*, dan respon *JSON* yang sesuai. Setiap kasus uji memvalidasi fungsionalitas spesifik dari aplikasi *Wedding Organizer* "Ruang Hati", mulai dari otentikasi pengguna hingga manajemen sumber daya.

- *Pengujian Endpoint Login Pengguna*

Gambar ini menampilkan proses otentikasi pengguna yang berhasil dalam aplikasi *Wedding Organizer* "Ruang Hati". Sebuah permintaan *POST* dikirim ke *endpoint /api/login*. *Body* permintaan, yang dikonfigurasi sebagai *form-data*, menyertakan kredensial *email* (admin@example.com) dan *password* (password123). Setelah validasi berhasil oleh *backend* Laravel, *API* merespon dengan kode status *HTTP 200 OK*. *Body* respon, yang diformat dalam *JSON*, berisi token (sebuah Bearer token yang esensial untuk permintaan terotentikasi berikutnya) dan objek user yang merinci *id*, *name*, dan *email* pengguna yang terotentikasi. Token ini sangat penting untuk mengamankan akses ke sumber daya *API* yang dilindungi.

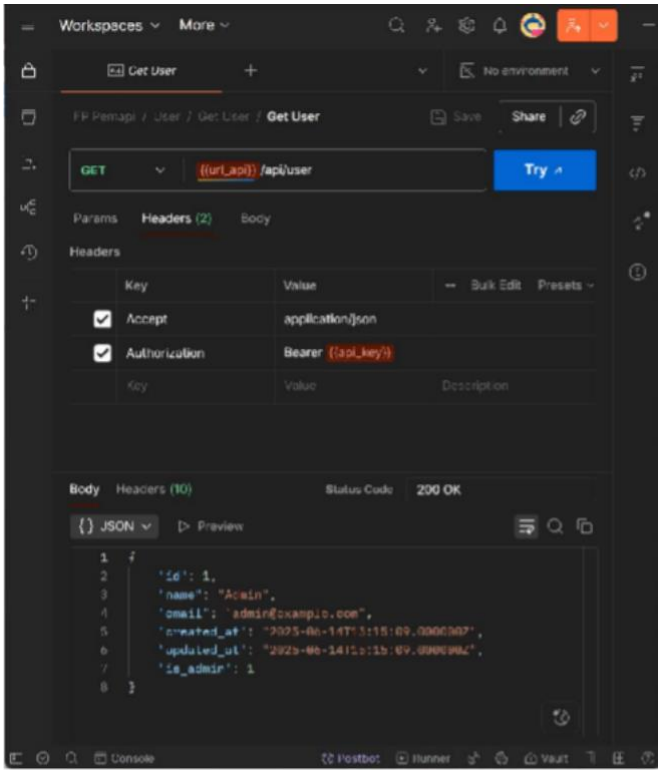


Gbr. 19 Pengujian *Endpoint Login* Pengguna.

Pengembalian token dan data pengguna segera setelah *login* yang berhasil menyoroti sifat stateless dari *API RESTful* dan mekanisme otentikasi yang dipilih. *API REST*, secara desain, bersifat *stateless*, yang berarti server tidak menyimpan informasi sesi klien. Token berfungsi sebagai bukti otentikasi klien untuk permintaan selanjutnya, menghilangkan kebutuhan untuk pengiriman kredensial berulang kali. Penggunaan *Bearer* token adalah metode umum dan efektif untuk mengimplementasikan keamanan *API*, memungkinkan klien untuk otentikasi sekali dan kemudian menggunakan token tersebut untuk akses yang diotorisasi, mengurangi paparan kredensial. Pilihan desain ini memastikan skalabilitas dan ketahanan, karena setiap instans server dapat memvalidasi token tanpa bergantung pada status sesi bersama, yang sangat penting untuk aplikasi *web* yang berkembang.

- *Pengujian Endpoint Mendapatkan Data Pengguna*

Gambar ini menunjukkan pengambilan data pengguna yang terotentikasi. Sebuah permintaan *GET* dibuat ke *endpoint /api/user*. Yang terpenting, permintaan ini menyertakan *header Authorization* dengan *Bearer* token, yang diperoleh dari proses *login* (seperti yang ditunjukkan pada Gbr. 19). *Header Accept: application/json* memastikan klien lebih memilih respon *JSON*. Respon yang berhasil, ditunjukkan dengan *200 OK*, menyediakan *id*, *name*, *email*, *created_at*, *updated_at*, dan status *is_admin* pengguna. Ini memvalidasi bahwa informasi profil pengguna yang dilindungi dapat diakses dengan aman setelah otentikasi.



Gbr. 20 Pengujian *Endpoint* Mendapatkan Data Pengguna.

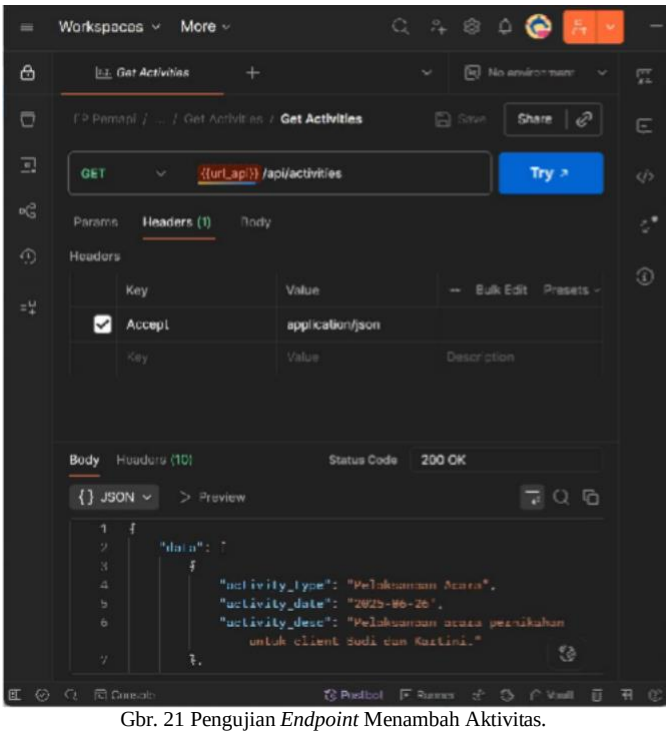
Persyaratan *header Authorization* dengan Bearer token untuk permintaan *GET* ini mengonfirmasi bahwa *API* mengimplementasikan kontrol akses yang kuat untuk data pengguna yang sensitif. Hal ini menunjukkan bahwa *endpoint /api/user* adalah sumber daya yang dilindungi, mencegah akses

ISSN (Online) 2747-0563
Volume 5 Tahun 2025

tidak sah ke profil pengguna. Selain itu, ini memperkuat hubungan sebab-akibat di mana *login* yang berhasil (Gbr. 19) merupakan prasyarat untuk mengakses sumber daya yang dilindungi, karena token adalah kuncinya. Model keamanan berlapis ini fundamental untuk setiap aplikasi yang menangani data pengguna, memastikan kerahasiaan dan integritas data, serta selaras dengan praktik terbaik dalam desain *API*.

- *Pengujian Endpoint Menambah Aktivitas*

Gambar ini mengilustrasikan pembuatan catatan aktivitas baru untuk *Wedding Organizer* "Ruang Hati". Sebuah permintaan *POST* dikirim ke *endpoint /api/activities*. *Body* permintaan dikirim sebagai *JSON* dan berisi parameter seperti *activity_type_koordinasi* (misalnya, "Staff Koordinasi"), *activity_status_pending* (misalnya, "*Pending*"), *activity_name* (misalnya, "Koordinasi Dengan Staff Humas"), *activity_date* (misalnya, "2025-07-04"), *activity_desc* (deskripsi koordinasi), dan *reservation_id* (misalnya, "2"), yang menghubungkan aktivitas ke reservasi tertentu. *API* merespon dengan *201 Created*, menunjukkan pembuatan sumber daya yang berhasil, dan mengembalikan data aktivitas yang baru dibuat termasuk *activity_id* uniknya dan semua atribut yang dikirimkan.

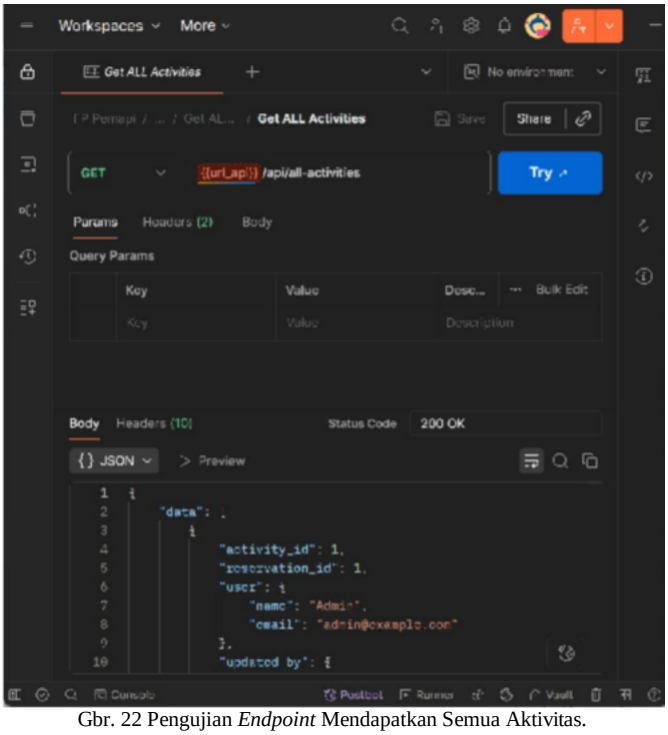


Gbr. 21 Pengujian *Endpoint* Menambah Aktivitas.

Kode status *201 Created*, ditambah dengan pengembalian data sumber daya yang baru dibuat, menunjukkan semantik *API RESTful* yang tepat untuk pembuatan sumber daya. *201 Created* adalah kode status *HTTP* standar untuk pembuatan sumber daya yang berhasil, secara jelas mengomunikasikan hasilnya kepada klien. Mengembalikan seluruh sumber daya yang baru dibuat (termasuk *activity_id*nya) sangat bermanfaat bagi klien (misalnya, *frontend* VueJs), karena klien segera memiliki akses ke representasi lengkap yang dibuat oleh server dari sumber daya tersebut tanpa memerlukan permintaan *GET* berikutnya. Kepatuhan terhadap prinsip-prinsip *REST* ini membuat *API* menjadi intuitif dan efisien bagi pengembang, mengurangi perjalanan pulang-pergi yang tidak perlu dan meningkatkan pengalaman pengguna aplikasi "Ruang Hati".

- *Pengujian Endpoint Mendapatkan Semua Aktivitas*

Gambar ini menunjukkan pengambilan daftar komprehensif semua aktivitas yang dikelola oleh sistem "Ruang Hati". Sebuah permintaan *GET* dikeluarkan ke *endpoint /api/all-activities*. Serupa dengan pengambilan data pengguna, *endpoint* ini memerlukan *header Authorization* dengan *Bearer* token, yang mengonfirmasi sifatnya yang dilindungi. *API* merespon dengan *200 OK* dan *array JSON* dari data, di mana setiap elemen mewakili sebuah aktivitas. Setiap objek aktivitas mencakup detail seperti *activity_id*, *reservation_id*, *user* (objek bersarang dengan *name* dan *email* pengguna yang terkait dengan aktivitas), *updated_by* (objek pengguna bersarang serupa), *activity_type*, *activity_name*, *activity_desc*, *activity_status*, *created_at*, dan *updated_at*. Pengujian ini memvalidasi kemampuan *API* untuk menyediakan tampilan konsolidasi dari semua aktivitas operasional.



Gbr. 22 Pengujian *Endpoint* Mendapatkan Semua Aktivitas.

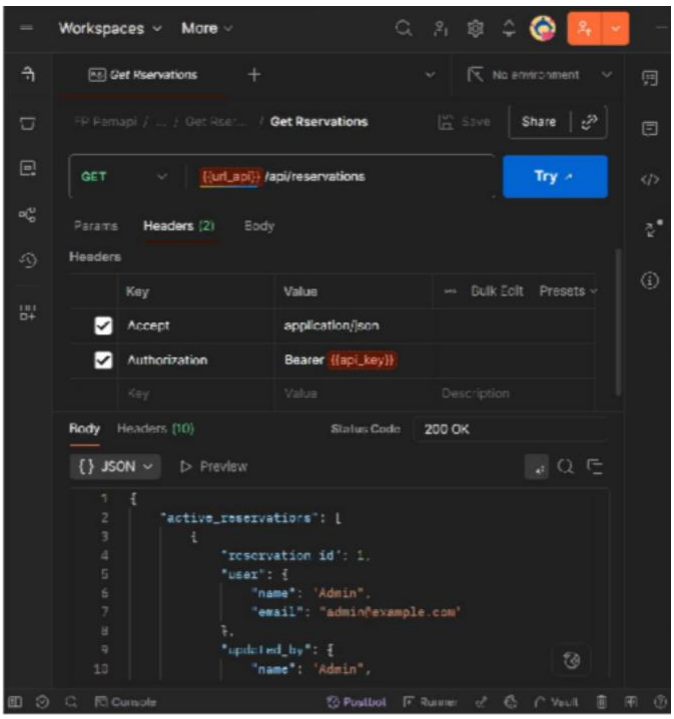
Penyertaan objek user dan *updated_by* yang bersarang dalam data aktivitas menunjukkan penggunaan yang efektif dari hubungan sumber daya dan denormalisasi data untuk pengambilan data yang efisien. Meskipun aktivitas dan pengguna kemungkinan merupakan entitas terpisah dalam basis data (misalnya, melalui kunci asing), respon *API* denormalisasi data ini dengan menyematkan detail pengguna

ISSN (Online) 2747-0563
Volume 5 Tahun 2025

secara langsung. Pendekatan ini mengurangi jumlah panggilan *API* yang dibutuhkan oleh *frontend*. Alih-alih mengambil aktivitas dan kemudian membuat panggilan terpisah untuk setiap pengguna terkait, semua informasi yang relevan diambil dalam satu kali, meningkatkan kinerja dan menyederhanakan pengembangan frontend. Pilihan desain ini menyeimbangkan normalisasi basis data dengan efisiensi konsumsi *API*, pola umum dalam desain *API REST* yang baik untuk mengoptimalkan rendering sisi klien dan mengurangi latensi, yang penting untuk dasbor *wedding organizer* yang responsif.

- *Pengujian Endpoint Mendapatkan Reservasi*

Gambar ini mengilustrasikan pengambilan detail reservasi aktif dalam aplikasi *Wedding Organizer* "Ruang Hati". Sebuah permintaan *GET* dikirim ke *endpoint /api/reservations*, yang memerlukan *header Authorization* dengan *Bearer* token. *API* merespon dengan *200 O*K dan objek *JSON* yang berisi *active_reservations*. Objek ini menyimpan array catatan reservasi, masing-masing dengan reservation_id, detail *user* terkait (nama, *email*), detail pengguna *updated_by*, dan *array vendors* yang bersarang. Setiap objek vendor mencakup *vendor_id*, *vendor_type*, dan vendor_brand. Pengujian ini mengonfirmasi kemampuan *API* untuk mengambil struktur data yang kompleks dan saling terhubung yang terkait dengan reservasi, menyediakan tampilan holistik dari rencana pernikahan yang sedang berjalan.

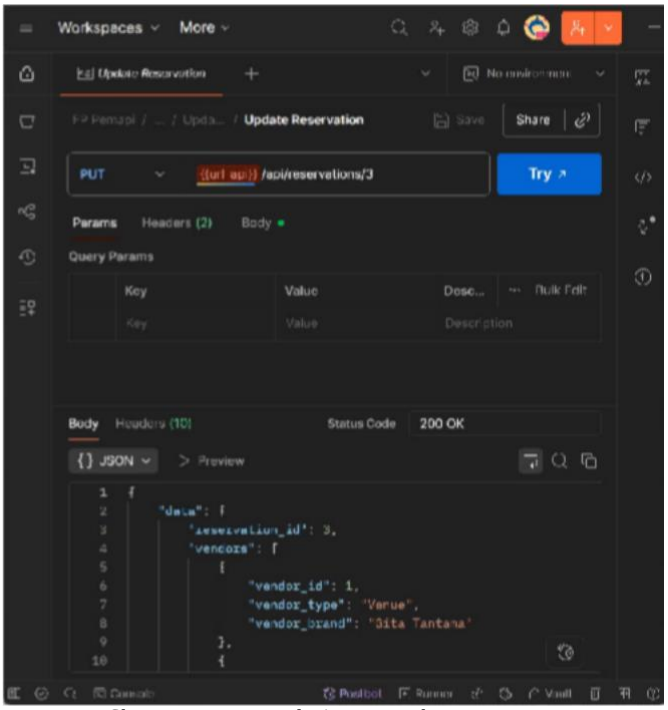


Gbr. 23 Pengujian *Endpoint* Mendapatkan Reservasi.

Penyarangan mendalam dari objek user dan vendors dalam respon reservations menyoroti kemampuan *API* untuk mengagregasi data terkait, mendukung persyaratan *UI* yang kompleks dengan panggilan *API* tunggal. Reservasi adalah entitas sentral, secara alami terhubung dengan klien, staf, dan beberapa vendor (misalnya, lokasi, katering, fotografi). *API* mencerminkan kompleksitas dunia nyata ini. Sebuah panggilan *GET /api/reservations* tunggal menyediakan semua informasi yang diperlukan untuk menampilkan tampilan reservasi yang terperinci di *frontend*, menghindari "masalah N+1" di mana N panggilan tambahan akan diperlukan untuk mengambil data terkait untuk setiap reservasi. Pendekatan ini mengoptimalkan transfer data dan mengurangi *overhead* jaringan, menghasilkan antarmuka pengguna yang lebih cepat dan responsif untuk mengelola reservasi pernikahan, yang sangat penting untuk pengalaman pengguna yang lancar dalam aplikasi profesional.

- Pengujian *Endpoint* Memperbarui Reservasi

Gambar ini menunjukkan proses pembaruan catatan reservasi yang sudah ada. Sebuah permintaan *PUT* dikirim ke *endpoint /api/reservations/2*, menargetkan reservasi spesifik berdasarkan *ID*-nya (dalam kasus ini, *ID 2*). *Body* permintaan, yang diformat sebagai *JSON*, berisi bidang yang diperbarui, seperti *wedding_contract_notes*, *invitation_notes*, *wedding_package_id*, *wedding_date*, *reservation_status* (misalnya, "Batal"), *vendor_ids* (*array ID* vendor untuk dikaitkan), dan combined_name. *API* merespon dengan *200 OK*, menunjukkan pembaruan yang berhasil, dan mengembalikan objek data dari reservasi yang diperbarui, termasuk *array vendors*nya, yang mencerminkan perubahan yang dilakukan. Ini memvalidasi kemampuan *API* untuk memodifikasi detail reservasi yang ada, yang penting untuk manajemen perencanaan pernikahan yang dinamis.



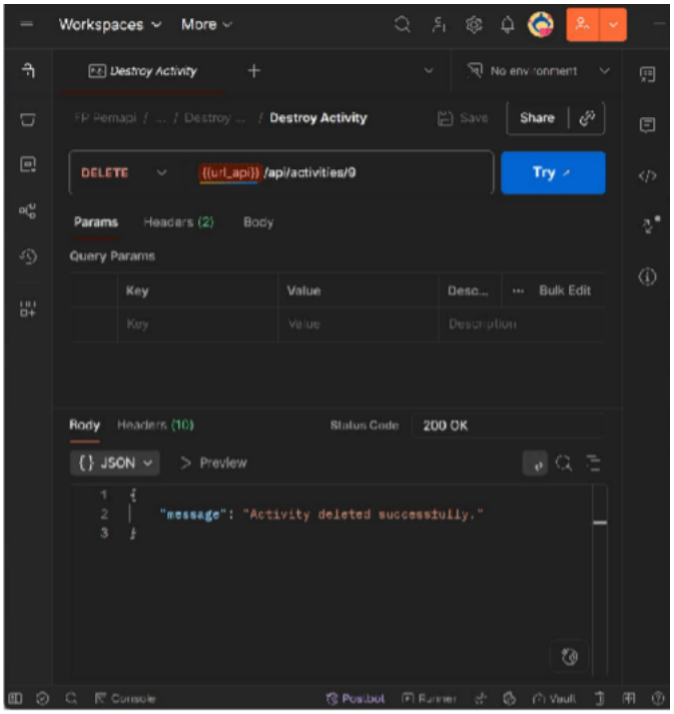
Gbr. 24 Pengujian *Endpoint* Memperbarui Reservasi.

ISSN (Online) 2747-0563
Volume 5 Tahun 2025

Penggunaan *PUT* untuk memperbarui sumber daya spesifik berdasarkan *ID*-nya (*/api/reservations/2*) secara ketat mematuhi prinsip-prinsip *RESTful* untuk pembaruan idempotent. *PUT* secara semantik digunakan untuk penggantian lengkap atau pembaruan sumber daya pada *URI* tertentu. Ini bersifat idempotent, yang berarti beberapa permintaan identik memiliki efek yang sama dengan satu permintaan. Dengan menargetkan *ID* spesifik, *API* memastikan bahwa hanya reservasi yang dimaksud yang dimodifikasi, mencegah pembaruan yang tidak disengaja pada catatan lain. Mekanisme pembaruan yang tepat dan idempotent ini fundamental untuk menjaga konsistensi dan keandalan data dalam aplikasi kritis seperti wedding organizer, di mana perubahan pada reservasi harus akurat dan dapat dilacak.

- *Pengujian Endpoint Menghapus Aktivitas*

Gambar ini mengilustrasikan penghapusan catatan aktivitas dari sistem *Wedding Organizer* "Ruang Hati". Sebuah permintaan *DELETE* dikirim ke *endpoint /api/activities/1*, menargetkan aktivitas dengan *ID 1*. Permintaan ini juga memerlukan *header Authorization* dengan **Bearer token*, memastikan bahwa hanya pengguna yang terotentikasi dan terotorisasi yang dapat melakukan operasi penghapusan. *API* merespon dengan *200 OK*dan pesan *JSON* sederhana:"*message*": "*Activity deleted successfully*"`. Pengujian ini mengonfirmasi kemampuan *API* untuk menghapus catatan aktivitas spesifik, menyediakan fungsionalitas *CRUD* (*Create*, *Read*, *Update*, *Delete*) lengkap untuk sumber daya aktivitas.



Gbr. 25 Pengujian *Endpoint* Menghapus Aktivitas.

Metode *DELETE*, ditambah dengan status *200 OK* dan pesan konfirmasi, menunjukkan pola yang jelas dan efektif untuk penghapusan sumber daya dalam *API RESTful*. *DELETE* adalah metode *HTTP* standar untuk menghapus sumber daya. Meskipun *204 No Content* sering digunakan untuk penghapusan yang berhasil tanpa *body* respon, *200 OK* dengan pesan konfirmasi juga dapat diterima dan memberikan umpan balik langsung kepada klien. Pesan eksplisit "*Activity deleted successfully*" memberikan umpan balik yang jelas kepada pengguna atau aplikasi klien bahwa operasi berhasil, yang sangat penting untuk tindakan destruktif. Melengkapi siklus *CRUD* dengan mekanisme penghapusan yang kuat memastikan bahwa aplikasi "Ruang Hati" memiliki kontrol penuh atas siklus hidup datanya, memungkinkan manajemen dan pembersihan data yang tepat, yang vital untuk kesehatan sistem dan kepatuhan.

IV. KESIMPULAN

Berdasarkan hasil perancangan, implementasi, dan pengujian aplikasi berbasis *web Wedding Organizer* "Ruang Hati" dengan arsitektur RESTful *API* menggunakan Laravel dan VueJs, dapat ditarik beberapa kesimpulan sebagai berikut. Pertama, penerapan arsitektur terpisah antara *backend* dan *frontend* terbukti meningkatkan modularitas dan skalabilitas sistem. Laravel mampu melayani endpoint *API* dengan konsistensi respon yang sesuai kontrak, sementara VueJs menyajikan antarmuka dinamis, responsif, dan intuitif bagi pengguna.

Kedua, model *SDLC Waterfall* memberikan kerangka kerja yang terstruktur dalam fase requirement, perancangan, implementasi, hingga pengujian. Dokumentasi *UML* dan *ERD* yang komprehensif memudahkan tim pengembang dalam menerjemahkan kebutuhan termasuk dalam memanajemen vendor dan anggaran hingga menjadi skema basis data dan alur kerja sistem yang jelas.

Ketiga, pengujian fungsional dengan *Black-Box Testing* dan verifikasi endpoint menggunakan Postman menegaskan bahwa seluruh fitur *CRUD* pada modul manajemen vendor, anggaran, reservasi, serta autentikasi berjalan sesuai spesifikasi tanpa kegagalan. Hasil ini meneguhkan keandalan aplikasi dalam menjaga integritas data dan keamanan akses antar komponen.

Keempat, integrasi kalender jadwal dan mekanisme notifikasi pop-up pada landing page memperkaya pengalaman pengguna dengan informasi ketersediaan dan detail event secara *real-time*. Hal ini menegaskan bahwa RESTful *API* tidak hanya sebagai lapisan layanan data, tetapi juga sebagai fondasi interaksi yang *seamless* antara server dan klien.

Sebagai rekomendasi, penelitian selanjutnya dapat mengeksplorasi penerapan arsitektur *micro-services* untuk memecah modul-modul berskala besar menjadi layanan independen, serta menambahkan pengujian performa (*load testing*) untuk menilai ketahanan sistem di bawah beban tinggi. Dengan demikian, "Ruang Hati" dapat terus berkembang

ISSN (Online) 2747-0563
Volume 5 Tahun 2025

menjadi solusi digital yang semakin fleksibel, aman, dan siap diintegrasikan dengan ekosistem layanan pernikahan modern.

UCAPAN TERIMA KASIH

Puji syukur kehadirat Tuhan Yang Maha Esa, tim penulis Kelompok 6 berhasil menyelesaikan karya tulis ilmiah "Implementasi RESTful *API Web Wedding Organizer* Ruang Hati menggunakan Laravel dan VueJs" sebagai syarat Evaluasi Akhir Semester mata kuliah Pemrograman API (B081) tahun ajaran 2024/2025. Kami mengucapkan terima kasih tulus kepada dosen pengampu dari Mata Kuliah Pilihan Pemrograman *API* (B081) pada program studi Informatika tahun ajaran 2024/2025 atas bimbingan dan arahannya, seluruh rekan Kelompok 6 atas kolaborasi dan komitmennya, serta semua pihak yang telah berkontribusi. Kami menyadari masih banyak kekurangan dalam karya tulis ini, sehingga kritik dan saran membangun sangat kami harapkan demi penyempurnaan. Besar harapan kami, penelitian dan aplikasi ini dapat memberikan manfaat serta kontribusi bagi pengembangan aplikasi berbasis *website* modern.

REFERENSI

[1] R. Sinaga and R. Ramadhana Sembiring, "Analisis Peluang Usaha Wedding Organizer Pada Kaum Millennial," *Journal of Millennial Community (JMIC)*, vol. 3, no. 2, pp. 107–113, 2021, doi: 10.24114/jmic.v3i2.32346.

[2] Nuranisyah, Yulianto, and Emil Riza Putra, "Web-Based Information System for Rental and Asset Data Collection on Wedding Organizer using Laravel Framework," *Tepian*, vol. 3, no. 3, pp. 132–138, 2022, doi: 10.51967/tepian.v3i3.741.

[3] T. Azra Rizkya Umri, Samsudin, and A. Muliani Harahap, "Sistem Informasi Pemesanan Nita Wedding Organizer Dengan Penerapan Customer Relationship Management Berbasis Web," *Journal of Science and Social Research*, vol. 7, no. 2, pp. 573–580, 2024, doi: https://doi.org/10.54314/jssr.v7i2.1875.

[4] F. Pakaja, A. Sudibyo, I. N. Susipta, and S. C. Hastuti, "Perancangan Sistem Informasi Feelight Moment Wedding Organizer dengan Framework Laravel," *Jurnal Teknologi Informatika dan Komputer*, vol. 10, no. 1, pp. 276–290, 2024, doi: 10.37012/jtik.v10i1.2121.

[5] H. Q. Zamani, P. P. Widagdo, and A. Irsyad, "Rancang Bangun Sistem Informasi Website Pergudangan Toko Mitra Mandiri Mebel Samarinda Berbasis Framework Laravel Dan Vue.Js Dengan Metode Waterfall," *Sains, Aplikasi, Komputasi dan Teknologi Informasi*, vol. 5, no. 1, pp. 18–31, 2023, doi: 10.30872/jsakti.v5i1.13517.

[6] K. F. Yufanka, E. Lutfina, A. Nugroho, and M. Z. Abdillah, "Systematic Literature Review: Perancangan Sistem Informasi Pemesanan Wedding Organizer Berbasis Web," *Science Technology and Management Journal*, vol. 3, no. 1, pp. 15–20, 2023, doi: http://dx.doi.org/10.26623/jtphp.v13i1.1845.kodeartikel.

[7] I. R. Mukhlis *et al.*, "Rancangan Basis Data Absensi Pegawai Menggunakan Mysql Dengan Conceptual Data Model (Cdm), Physical Data Model (Pdm), Dan Entity Relationship Diagram (Erd)," *Computing Insight : Journal of Computer Science*, vol. 6, no. 2, pp. 1–17, 2024, doi: 10.30651/comp_insight.v6i2.24337.

[8] A. W. Syahroni, N. P. Dewi, N. Ramadhani, Ubaidi, and B. Said, "Uji Keamanan Back end Aplikasi Berbasis Website Menggunakan Metode Black Box Testing," vol. 19, no. 2, pp. 215–226, 2024, [Online]. Available: https://doi.org/10.33998/processor.2024.19.2.1752

[9] D. Wintana, D. Pribadi, and M. Y. Nurhadi, "Analisis Perbandingan Efektifitas White-Box Testing dan Black-Box Testing," *Jurnal Larik: Ladang Artikel Ilmu Komputer*, vol. 2, no. 1, pp. 8–16, 2022, doi: 10.31294/larik.v2i1.1382.