

Rancang Bangun Website

Pizza Andaliman Balige

Tugas Proyek Akhir Semester
Mata Kuliah: Pengembangan Aplikasi Terdistribusi

Dipersiapkan oleh:

- | | |
|--------------|-------------------------|
| NIM 11322003 | Adinda Hutasoit |
| NIM 11322019 | Silvi Agustina Sitohang |
| NIM 11322035 | Niko Alvin Simanjuntak |
| NIM 11322063 | Hagai Natasha Sianturi |

Untuk:

Institut Teknologi Del
2024



DAFTAR ISI

1	Pendahuluan.....	1
1.1	Deskripsi Umum Sistem.....	1
1.2	Karakteristik Pengguna.....	2
1.3	Fungsi pada Sistem.....	3
2	Desain Rancangan Sistem.....	4
2.1	Use Case Diagram.....	4
2.2	Business Process Modeling Notation.....	4
3	Tampilan Sistem.....	19
4	Pengujian Database.....	26
5	Pengujian Sistem.....	30

1 Pendahuluan

Pada bab 1 berisi tentang deskripsi umum sistem, karakteristik pengguna, dan fungsi pada sistem.

1.1 Deskripsi Umum Sistem

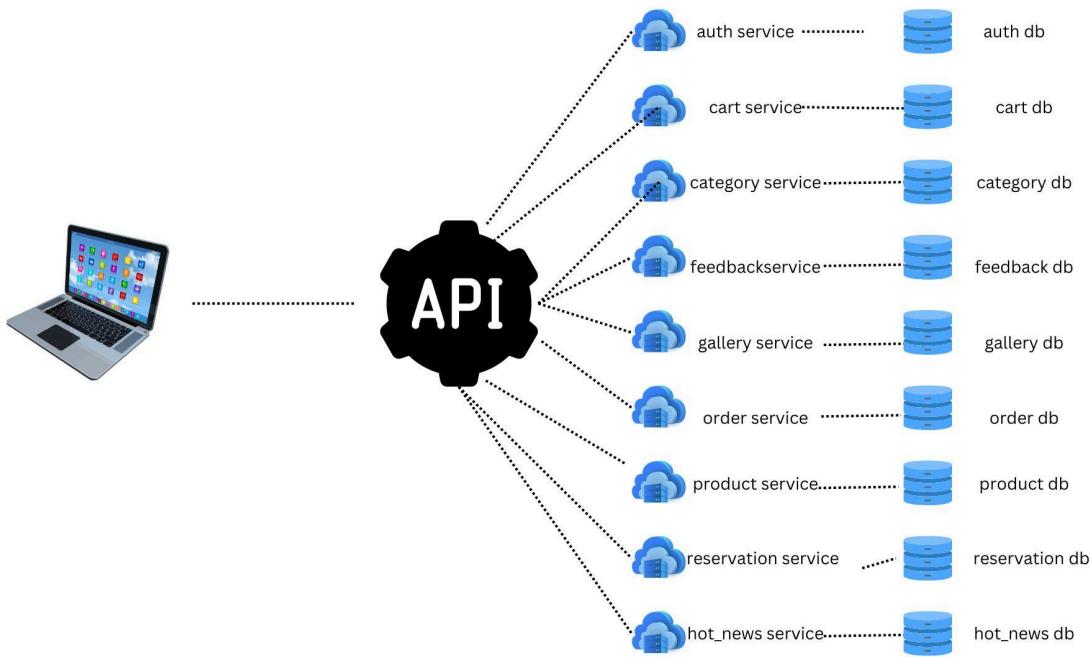
Sistem yang dibangun adalah *website* untuk Pizza Andaliman Balige. Proyek ini bertujuan untuk menyediakan sebuah *website* yang dapat mempermudah pemilik restoran, staf, dan pelanggan dalam berbagai kegiatan terkait restoran. Dengan adanya *website* Pizza Andaliman Balige, diharapkan proses seperti pemesanan, pengelolaan reservasi, dan memberikan ulasan dapat dilakukan dengan lebih efisien dan efektif. *Website* ini dirancang untuk diakses oleh staf restoran serta pelanggan yang telah memiliki akun terdaftar di dalam sistem. Namun, untuk pengunjung tanpa akun juga tetap dapat mengakses sebagian fitur yang tersedia, meskipun dalam kapasitas yang terbatas. Ini memungkinkan bagi pengunjung untuk tetap menjelajahi menu, informasi kontak, layanan, dan galeri foto yang terdapat pada Pizza Andaliman Balige. Dengan menggunakan *website* ini, diharapkan interaksi antara pelanggan dan restoran menjadi lebih lancar.

Sistem ini dibangun dengan menggunakan 2 bahasa pemrograman. Bagian *back-end* dan *front-end* dibangun dengan bahasa yang berbeda. Untuk bagian *back-end* digunakan bahasa *Go* dan untuk bagian *front-end* menggunakan bahasa *PHP*, khususnya *framework Laravel*.

Aplikasi ini menggunakan arsitektur microservice. Sesuai dengan namanya, arsitektur ini dirancang pada sebuah sistem untuk membagi *service* menjadi *service* yang lebih kecil. Dengan ini, maka setiap *service* akan memiliki *database*-nya masing-masing. *Service* yang ada pada *website* ini adalah *Auth Service*, *Product Service*, *Order Service*, *Cart Service*, *Category Service*, *Feedback Service*, *Gallery Service*, *Reservation Service*, *Hot News Service*. Setiap *service* akan berjalan secara *independent* sehingga satu *service* tidak akan memengaruhi *service* lainnya. Namun, meskipun berdiri sendiri, setiap *service* ini tetap saling terhubung antara satu dengan yang lainnya. Antarservice ini akan saling berkomunikasi dengan *HTTP Request* berstandar REST API.

Setiap *database* pada masing-masing *service* memiliki *port* yang sama. Namun, port yang digunakan pada setiap *service* tersebut akan berbeda. Hal ini ditujukan sebagai pemisah antar *service* sehingga saat satu *service* sedang tidak dapat dijalankan, maka *service* yang lainnya

masih tetap dapat berjalan.



Pada Gambar 1 terlampir arsitektur microservice yang digunakan pada pembangunan *website* Pizza Andaliman Balige. REST API digunakan sebagai penghubung antara *web ui* dengan *service* yang tersedia sehingga memungkinkan *service* dapat dikonsumsi oleh *web ui*. Hal inilah yang menyebabkan *service* dapat diakses oleh beberapa *platform* tanpa harus terbatas pada satu bahasa pemrograman saja. Sama halnya dengan pembangunan aplikasi web jual beli ini. Bahasa yang digunakan pada bagian *front-end* adalah bahasa PHP (*framework Laravel*) dan pada bagian *back-end* menggunakan bahasa Go. Meskipun memiliki bahasa yang berbeda, tetapi *website* akan tetap dapat berjalan dengan adanya REST API.

Pada bagian *back-end*, setiap *service* memiliki *database*-nya masing-masing. Setiap *service* dirancang memiliki *port* yang berbeda sehingga *port* tidak akan bertabrakan.

1.2 Karakteristik Pengguna

Pada *website* Pizza Andaliman Balige ini, terdapat 3 kategori pengguna yaitu admin, *customer*, dan *guest*. Karakteristik pengguna dilampirkan pada Tabel 1.

Tabel 1 Karakteristik Pengguna

Kategori Pengguna	Fungsi	Hak Akses ke Sistem
Admin	Mengelola informasi dalam <i>website</i>	1. Akses ke menu registrasi 2. Akses ke menu <i>login</i>

		<ol style="list-style-type: none"> 3. Akses untuk mengelola produk 4. Akses untuk mengelola kategori 5. Akses untuk mengelola informasi lainnya di dalam <i>website</i> 6. Akses ke menu <i>logout</i>
<i>Customer</i>	Melakukan pemesanan produk dan membuat reservasi dalam <i>website</i>	<ol style="list-style-type: none"> 1. Akses ke menu registrasi 2. Akses ke menu <i>login</i> 3. Akses untuk melakukan pemesanan 4. Akses untuk membuat reservasi 5. Akses ke menu <i>logout</i>
<i>Guest</i>	Melihat menu, galeri, dan berita terkini	<ol style="list-style-type: none"> 1. Akses untuk melihat menu produk 2. Akses untuk melihat galeri 3. Akses untuk melihat berita terkini

1.3 Fungsi pada Sistem

Adapun fungsi yang terdapat pada *website* Pizza Andaliman Balige ini, yaitu:

- 1. Fungsi autentikasi**

Fungsi autentikasi ini terdiri dari registrasi, *login*, dan *logout*. Registrasi dapat dilakukan oleh *guest* untuk dapat membuat akun. Setelah memiliki akun, mereka dapat melakukan *login* pada *website*.

- 2. Fungsi mengelola kategori**

Fungsi mengelola kategori terdiri dari membuat, mengedit, dan menghapus kategori. Fungsi ini digunakan oleh admin setelah melakukan *login* pada *website*.

- 3. Fungsi mengelola produk**

Fungsi mengelola produk terdiri dari membuat, mengedit, dan menghapus produk. Fungsi ini digunakan oleh admin setelah melakukan *login* pada *website*.

- 4. Fungsi menambahkan produk ke halaman keranjang**

Fungsi menambahkan produk ke halaman keranjang digunakan saat *customer* ingin melakukan pemesanan produk. *Customer* harus melakukan tahap ini sebelum lanjut ke dalam tahap pemesanan

- 5. Fungsi melakukan pemesanan**

Fungsi melakukan pemesanan ini dilakukan setelah *customer* menambahkan produk ke halaman keranjang. Setelah menambahkan produk ke halaman keranjang, *customer* dapat melanjutkan pemesanan produk sampai pada tahap pembayaran. Fungsi ini digunakan oleh *customer* setelah melakukan *login*.

- 6. Fungsi mengelola reservasi**

Fungsi mengelola reservasi terdiri dari membuat reservasi dan membatalkan reservasi. Fungsi ini digunakan oleh *customer* setelah melakukan *login*.

- 7. Fungsi membuat feedback**

Fungsi membuat feedback digunakan oleh *customer* saat ingin memberikan ulasan mengenai Pizza Andaliman Balige. Fungsi ini digunakan *customer* setelah melakukan *login*.

8. Fungsi mengelola galeri

Fungsi mengelola galeri terdiri dari menambah, mengedit, dan menghapus foto-foto pada bagian galeri. Fungsi ini digunakan oleh admin setelah melakukan *login*.

9. Fungsi mengelola berita terkini

Fungsi mengelola berita terkini terdiri dari menambah, mengedit, dan menghapus berita-berita yang akan ditampilkan pada *website* Pizza Andaliman Balige. Fungsi ini digunakan oleh admin setelah melakukan *login*.

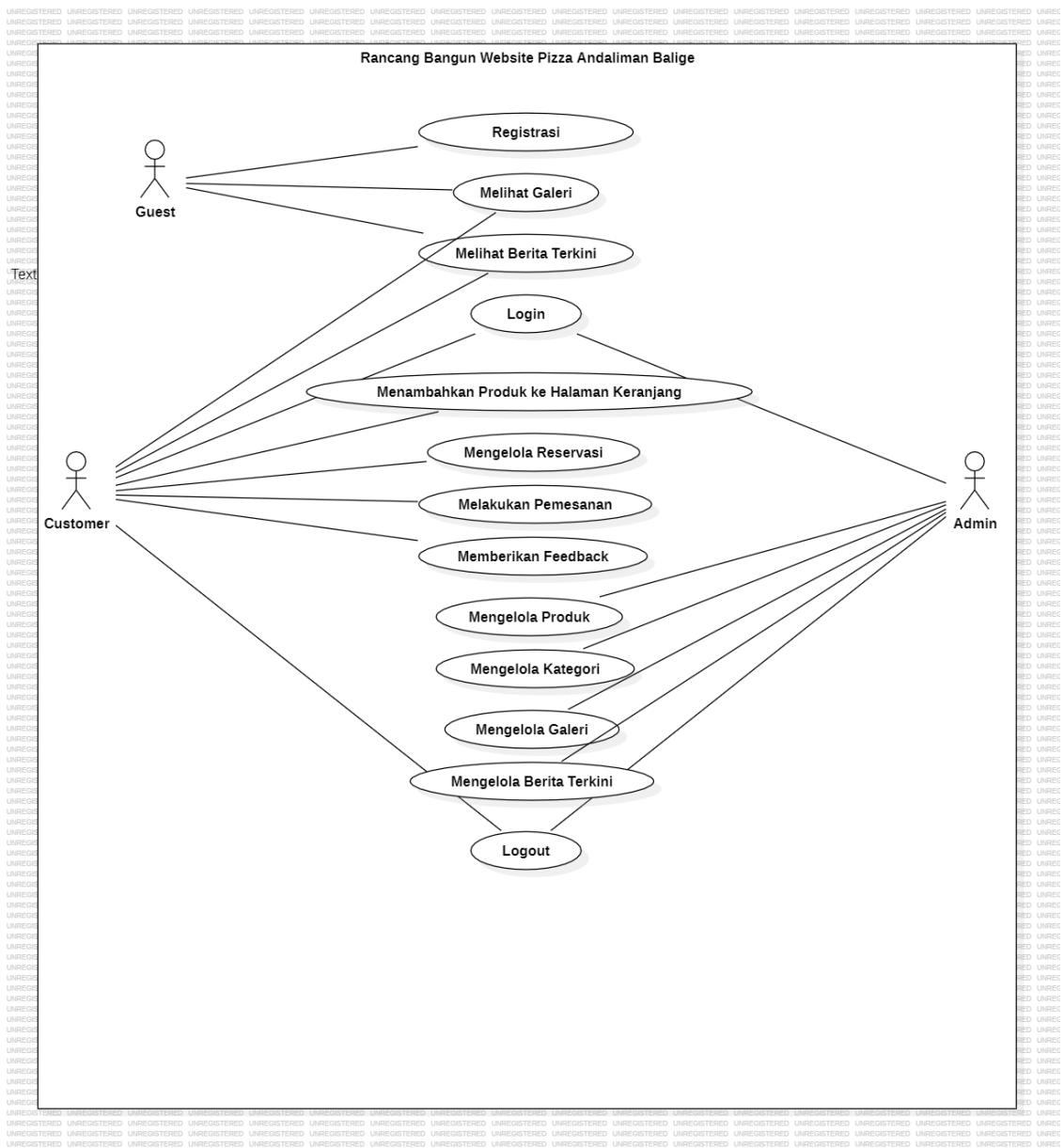
2 Desain Rancangan Aplikasi

Pada bab 2 berisi tentang desain rancangan *website*, yaitu *use case diagram* dan *business process modeling notation* (BPMN).

2.1 Use Case Diagram

Use case diagram pada Gambar 2 memuat setiap fungsi yang dapat dijalankan pada *website*. Website Pizza Andaliman Balige dapat digunakan oleh 3 *role*, yaitu *admin*, *customer*, dan *guest*. Admin dan *customer* harus melakukan registrasi terlebih dahulu agar dapat *login* ke dalam website, sedangkan *guest* tidak perlu melakukan registrasi. Yang menjadi pembeda terletak pada layanan yang dapat diakses keduanya setelah *login* ke dalam aplikasi. Admin dapat mengelola produk dan kategori, sedangkan *customer* hanya dapat melakukan pemesanan dan membuat reservasi.

Use case diagram *website* Pizza Andaliman Balige terlampir pada Gambar 2.



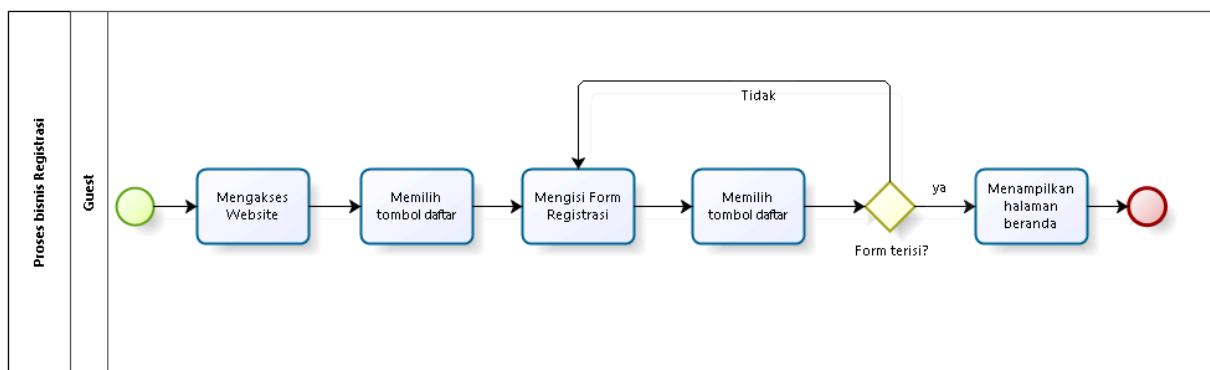
2.2 Business Process Modeling Notation

Setiap fungsi pada aplikasi dapat dijalankan dengan proses yang tertera pada setiap *business process modeling notation* berikut.

2.2.1

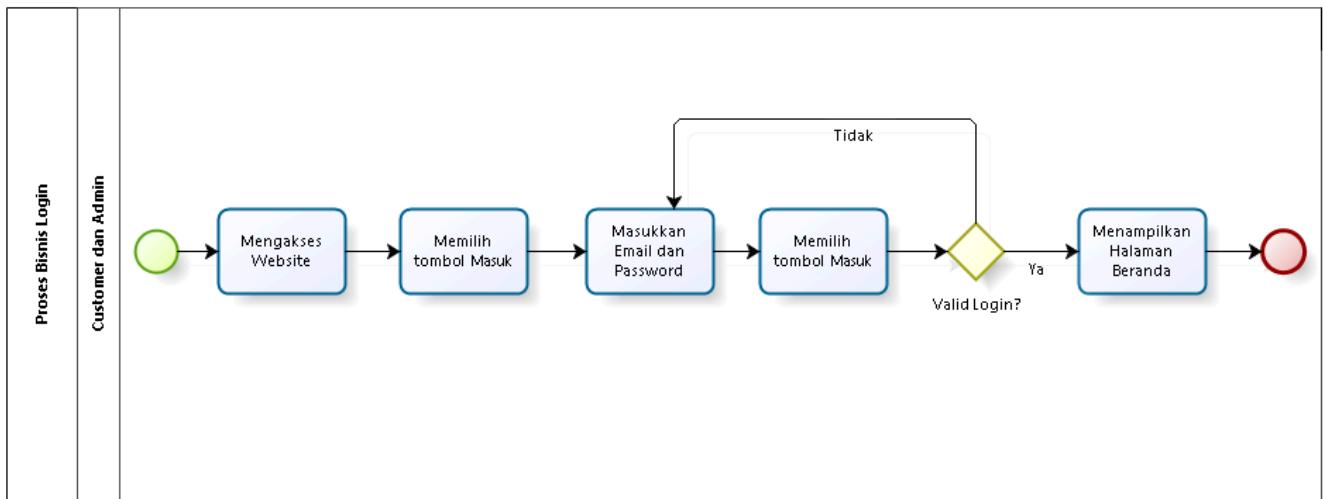
2.2.2 Proses Bisnis Registrasi

Proses bisnis yang diharapkan pada sistem yang dibangun yaitu *Customer* dapat melakukan pemesanan makanan dan minuman serta dapat melakukan reservasi dalam sistem. Untuk melakukan pemesanan dalam sistem ini, *Customer* harus mengakses *website* dan mendaftar akun terlebih dahulu dengan memilih tombol “Daftar” terlebih dahulu. Setelah itu, untuk mendaftarkan akun, *Customer* akan mengisi data pada *form* registrasi lalu memilih tombol Daftar. Jika data pada form lengkap, maka proses registrasi selesai. Namun, jika data tidak lengkap terisi, maka *Customer* akan mengisi ulang data pada *form* registrasi. Proses bisnis registrasi pada aplikasi terlampir pada Gambar 3.



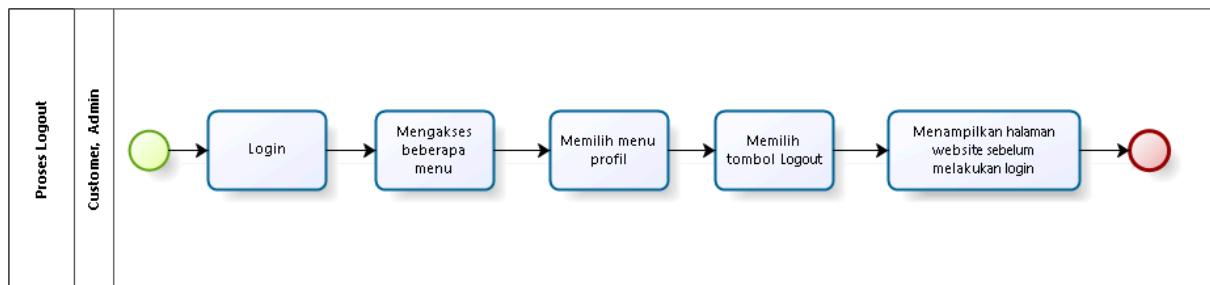
2.2.3 Proses Bisnis Login

Pada Proses Bisni *Login*, baik Admin maupun *Customer* harus memiliki akun terlebih dahulu. Saat *Customer* mengakses Website, *Customer* dapat memilih tombol Masuk. Setelah itu *Customer* memasukkan Email dan Password yang telah didaftarkan sebelumnya. kemudian memilih tombol “Masuk”. Jika pengisian Form Valid maka sistem akan menampilkan halaman beranda. Tetapi jika pengisian form tidak valid, maka *Customer* harus mengisi ulang form *Login*. Proses bisnis *login* pada aplikasi terlampir pada Gambar 4.



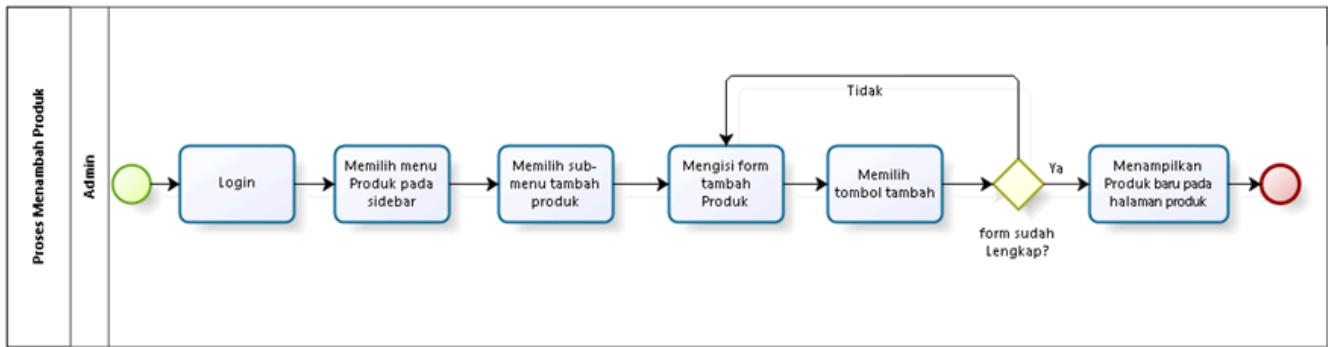
2.2.4 Proses Bisnis Logout

Pada proses bisnis *Logout*, baik Customer maupun Admin harus melakukan Login terlebih dahulu. Mengakses beberapa menu web, setelah itu memilih menu profil dan memilih tombol *logout*. Sistem akan menampilkan halaman website sebelum melakukan Login.



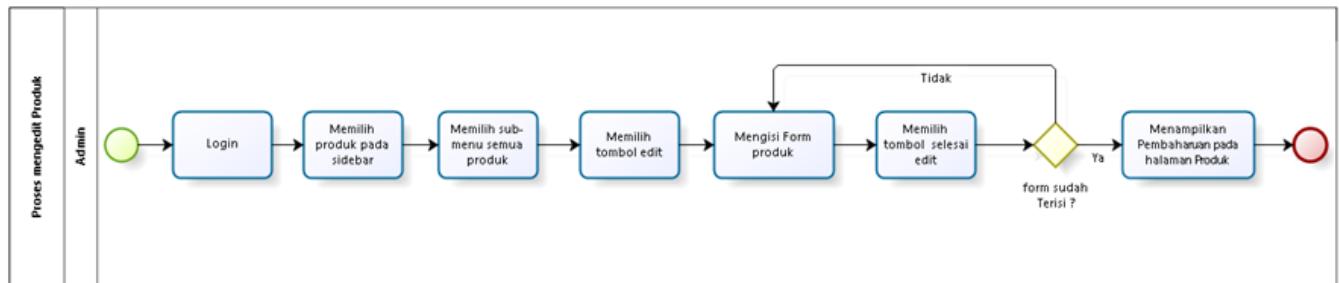
2.2.5 Proses Bisnis Menambah Produk

Pada proses bisnis menambah produk, *Admin* harus melakukan *login* terlebih dahulu. Admin memilih menu produk pada sidebar, kemudian memilih sub-menu tambah produk. Setelah itu Admin melakukan pengisian form tambah produk dan memilih tombol tambah. Jika form lengkap terisi maka produk baru akan tampil pada halaman produk. Jika form tambah produk tidak lengkap terisi maka Admin harus mengisi kembali form tambah produk.



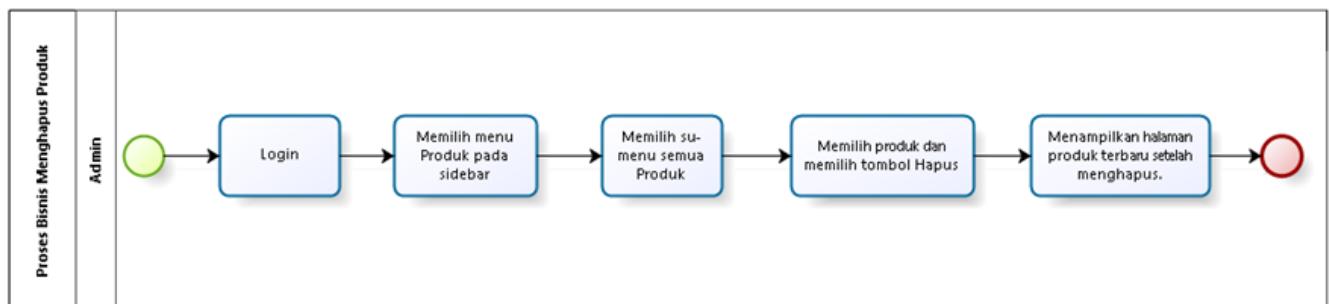
2.2.6 Proses Bisnis Mengedit Produk

Pada proses bisnis mengedit produk, Admin sebelumnya telah melakukan login terlebih dahulu. Kemudian Admin memilih menu Produk pada sidebar dan memilih submenu semua produk. Setelah itu Admin memilih tombol edit dan mengisi form produk. Terakhir Admin memilih tombol selesai. Jika form produk lengkap terisi maka perubahan yang dilakukan berhasil. Jika form produk tidak lengkap maka Admin harus mengisi ulang form produk hingga lengkap lalu menekan tombol selesai edit.



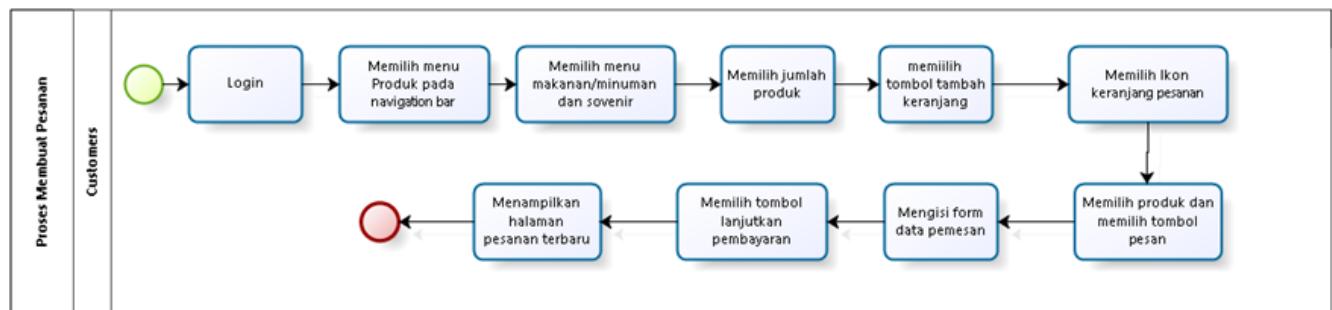
2.2.7 Proses Bisnis Menghapus Produk

Pada proses bisnis melakukan menghapus produk. Admin terlebih dahulu melakukan login. Setelah login Admin memilih menu produk pada sidebar dan memilih sub-menu semua produk. Kemudian Admin memilih produk dan memilih tombol hapus. Sistem akan menampilkan perubahan setelah Admin memilih menghapus produk



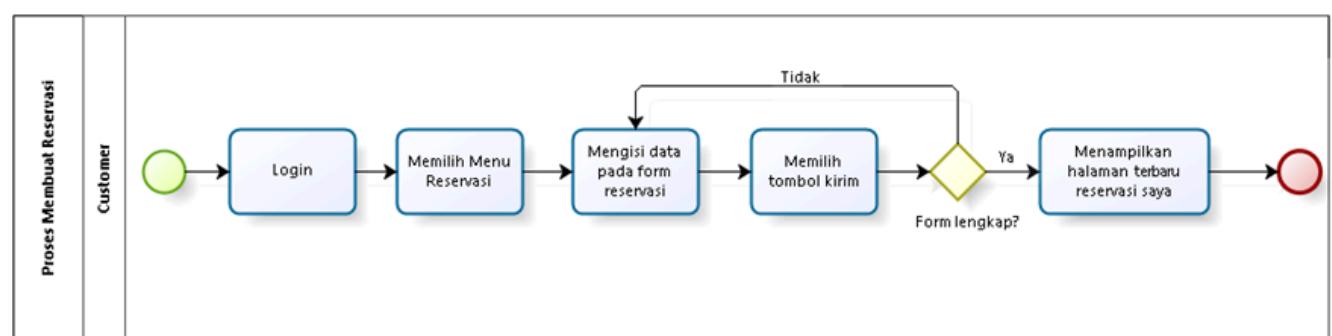
2.2.8 Proses Bisnis Membuat Pesanan

Pada proses bisnis membuat pesanan, Customer tentu harus melakukan login terlebih dahulu. Setelah login, Memilih menu Produk pada navigation bar, selanjutnya Customer memilih makanan/minuman dan memilih menu jumlah produk. Kemudian memilih tombol tambah keranjang. Langkah selanjutnya Customer memilih ikon keranjang pesanan. Kemudian Memilih produk lalu memilih tombol pesan. Customer harus mengisi form data pemesanan lalu memilih tombol lanjutkan pembayaran. Sistem nantinya akan menampilkan halaman pesanan yang telah dibuat



2.2.9 Proses Bisnis Membuat Reservasi

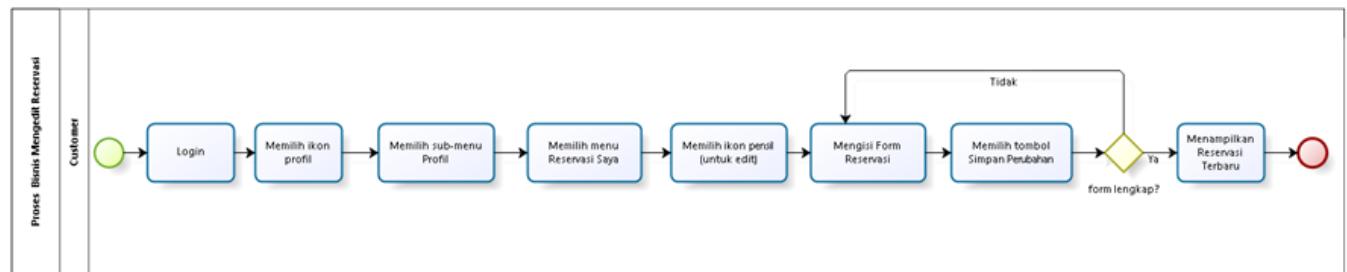
Pada proses bisnis membuat reservasi. *Customer* terlebih dahulu melakukan login, setelah itu *Customer* memilih menu reservasi. *Customer* mengisi form reservasi. Setelah form reservasi lengkap terisi, *Customer* memilih tombol kirim. Maka sistem menampilkan halaman terbaru pada reservasi saya. Jika *Customer* tidak mengisi form dengan lengkap maka sistem menampilkan form kembali untuk diisi dengan lengkap.



2.2.10 Proses Bisnis Mengedit Reservasi

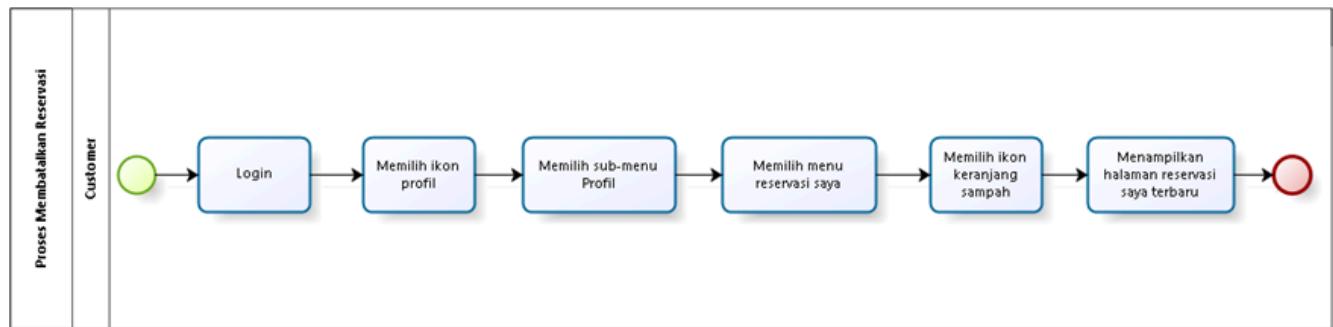
Pada proses bisnis mengedit reservasi, *Customer* harus melakukan login terlebih dahulu. Setelah Login, *Customer* memilih ikon profil dan memilih sub-menu profil. *Customer* memilih menu reservasi saya dan memilih ikon pensil (untuk mengedit reservasi). Kemudian *Customer* mengisi form reservasi dan memilih tombol simpan perubahan. Jika Form terisi dengan lengkap, maka sistem akan menampilkan halaman reservasi saya terbaru. Namun, jika *Customer* belum mengisi form dengan lengkap kemudian langsung

memilih tombol simpan perubahan, halaman akan tetap pada form reservasi, maka Customer harus mengisi dengan lengkap form reservasi lalu menekan tombol simpan perubahan.

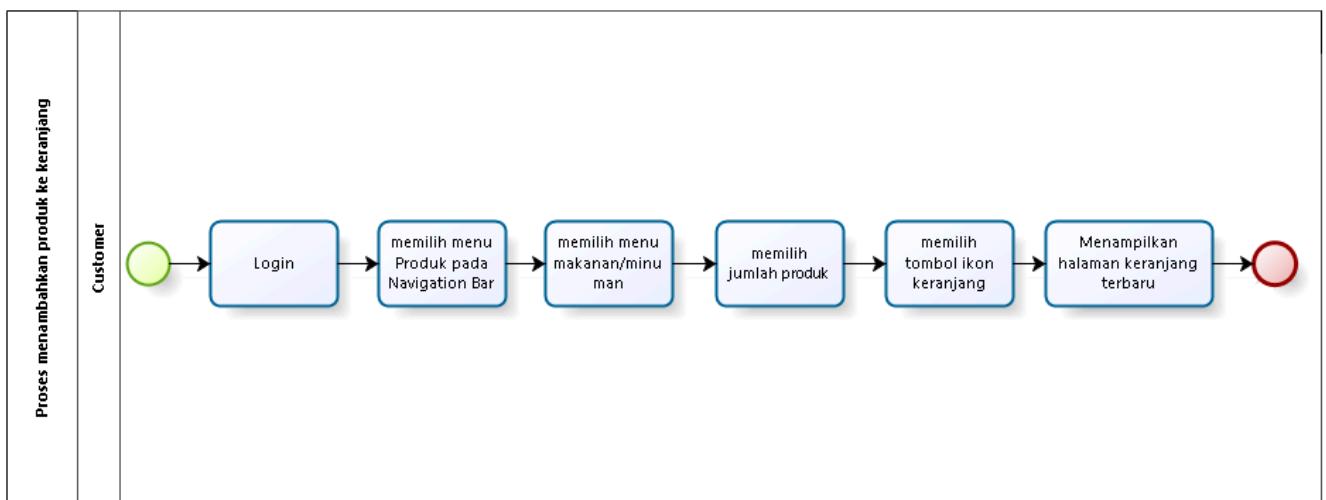


2.2.11 Proses Bisnis Membatalkan Reservasi

Pada Proses bisnis membatalkan reservasi, Customer tentu harus melakukan login terlebih dahulu. Kemudian Customer dapat memilih ikon profil dan memilih sub-menu profil. Setelahnya, Customer memilih menu reservasi saya dan memilih ikon keranjang sampah. Maka sistem akan menampilkan halaman terbaru untuk reservasi saya.

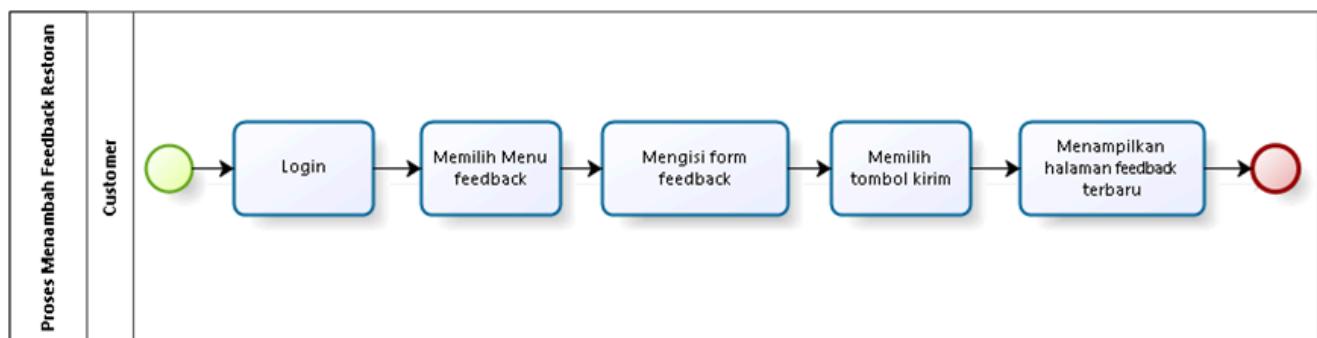


2.2.12 Proses Bisnis Menambah Produk Ke keranjang



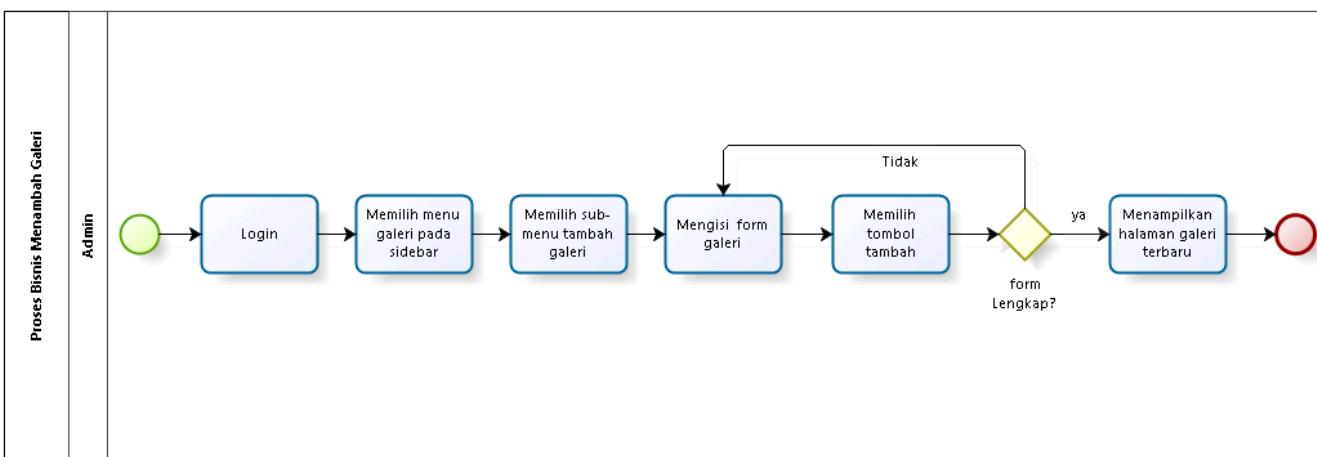
2.2.13 Proses Bisnis Menambah Feedback Restoran

Pada proses bisnis Memberikan *Feedback* restoran, *Customer* harus melakukan *login* terlebih dahulu. Setelah melakukan *login*, *Customer* memilih menu *feedback* dan mengisi form *feedback*. Setelah mengisi form *feedback*, Admin memilih tombol kirim. Sistem akan menampilkan halaman *Feedback* terbaru.



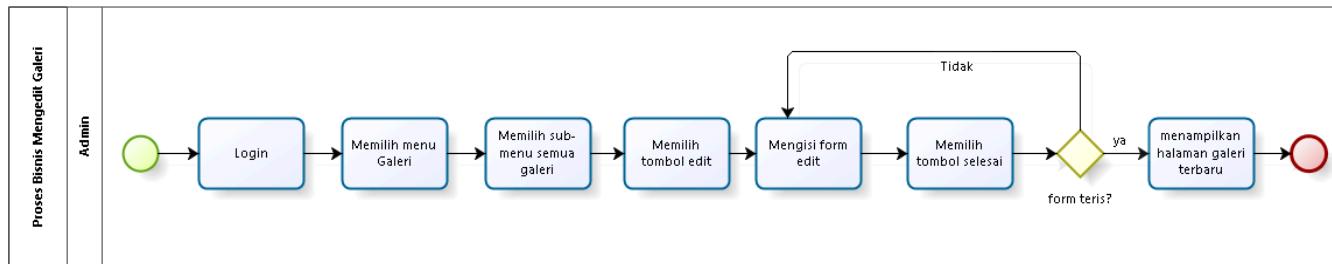
2.2.14 Proses Bisnis Menambah Galeri

Pada Proses Bisnis menambah galeri, *Admin* terlebih dahulu melakukan login. Kemudian *Admin* dapat memilih menu galeri pada sidebar dan memilih sub-menu tambah galeri. Selanjutnya Admin mengisi form galeri dan memilih tombol tambah. Jika Admin mengisi form dengan lengkap maka sistem akan menampilkan halaman galeri terbaru. Jika Admin tidak mengisi form dengan lengkap maka Admin harus mengisi form dengan lengkap terlebih dahulu baru memilih tombol tambah



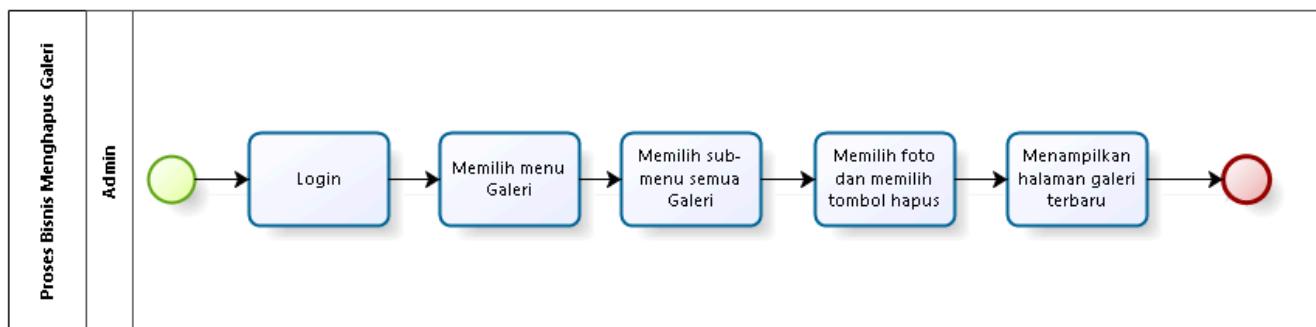
2.2.15 Proses Bisnis Mengedit Galeri

Pada proses bisnis mengedit galeri, Admin melakukan login terlebih dahulu. Setelah melakukan *login*, *Admin* dapat memilih menu galeri, kemudian *Admin* memilih sub-menu semua galeri. Setelahnya Admin memilih tombol edit dan mengisi from edit. Terakhir admin memilih tombol selesai. Jika admin mengisi form dengan lengkap maka akan tampil pembaharuan halaman galeri. Jika admin tidak mengisi form dengan lengkap maka admin harus melengkapi form galeri terlebih dahulu memilih tombol selesai. Proses bisnis Mengedit



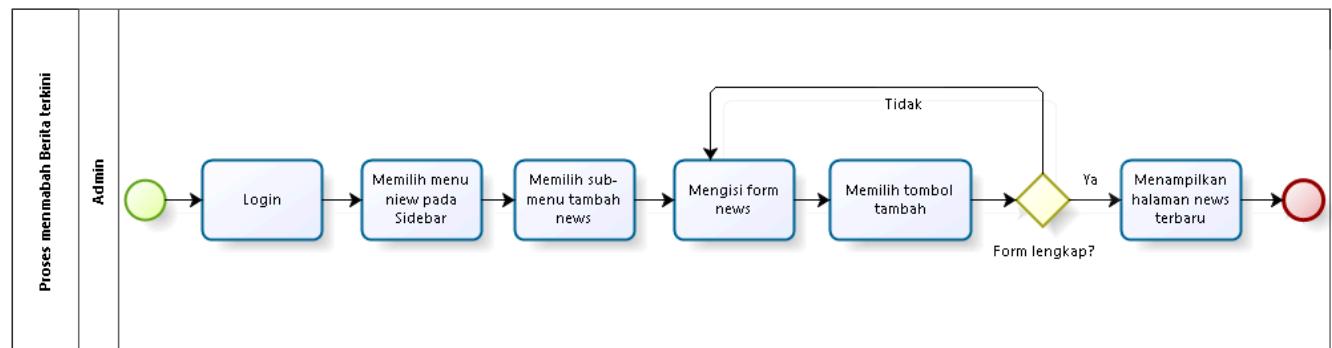
2.2.16 Proses Bisnis Menghapus Galeri

Pada Proses Bisnis galeri, admin melakukan login terlebih dahulu. Kemudian admin memilih menu galeri dan memilih sub-menu semua galeri. Setelah itu, Admin memilih galeri dan memilih tombol hapus.



2.2.17 Proses Bisnis Menambah Berita Terkini

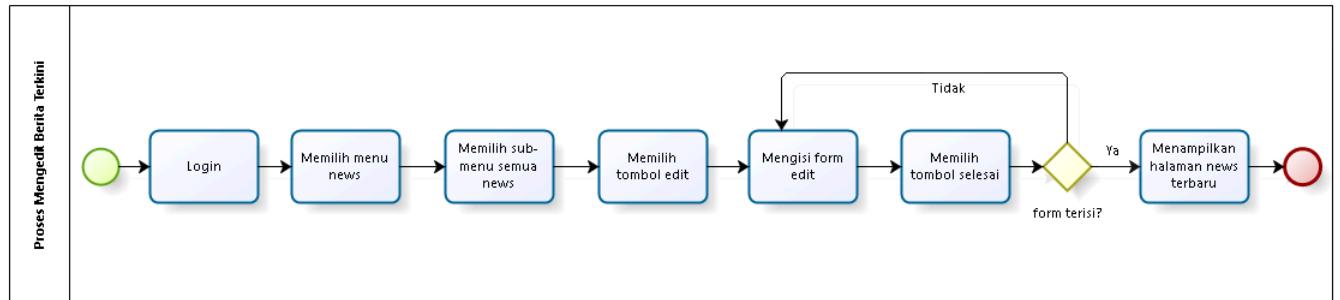
Pada Proses Bisnis menambah Berita Terkini, *Admin* terlebih dahulu melakukan login. Kemudian *Admin* dapat memilih menu news pada sidebar dan memilih sub-menu tambah news. Selanjutnya Admin mengisi form news dan memilih tombol tambah. Jika Admin mengisi form dengan lengkap maka sistem akan menampilkan halaman news terbaru. Jika Admin tidak mengisi form dengan lengkap maka Admin harus mengisi form dengan lengkap terlebih dahulu baru memilih tombol tambah.



2.2.18 Proses Bisnis Mengedit Berita terkini

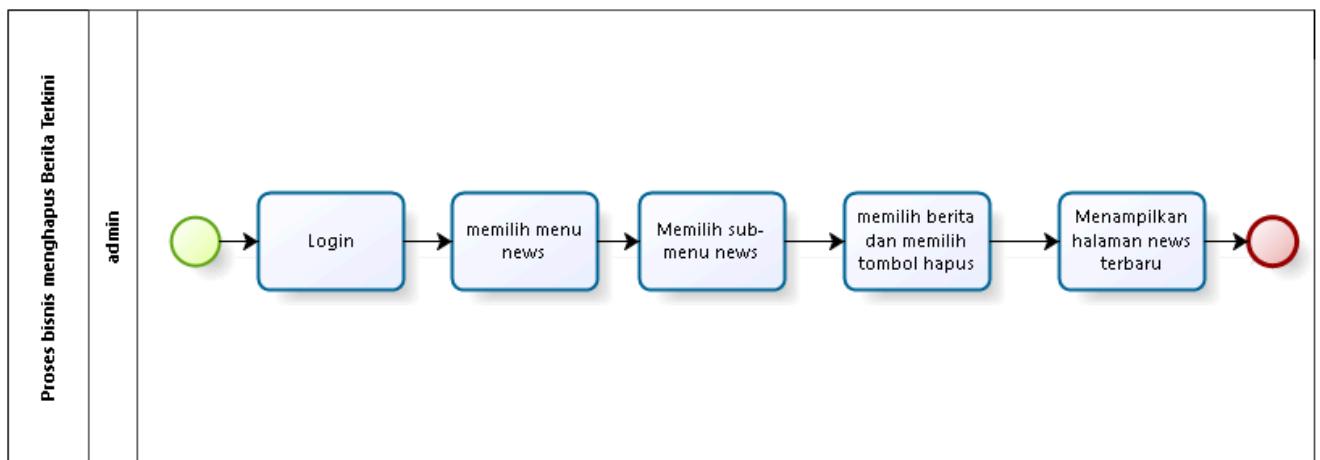
Pada proses bisnis mengedit berita terkini, Admin melakukan login terlebih dahulu. Setelah melakukan login, Admin dapat memilih menu news, kemudian Admin memilih sub-menu semua news. Setelahnya

Admin memilih tombol edit dan mengisi form edit. Terakhir admin memilih tombol selesai. Jika admin mengisi form dengan lengkap maka akan tampil pembaharuan halaman news. Jika admin tidak mengisi form dengan lengkap maka admin harus melengkapi form news terlebih dahulu memilih tombol selesai.



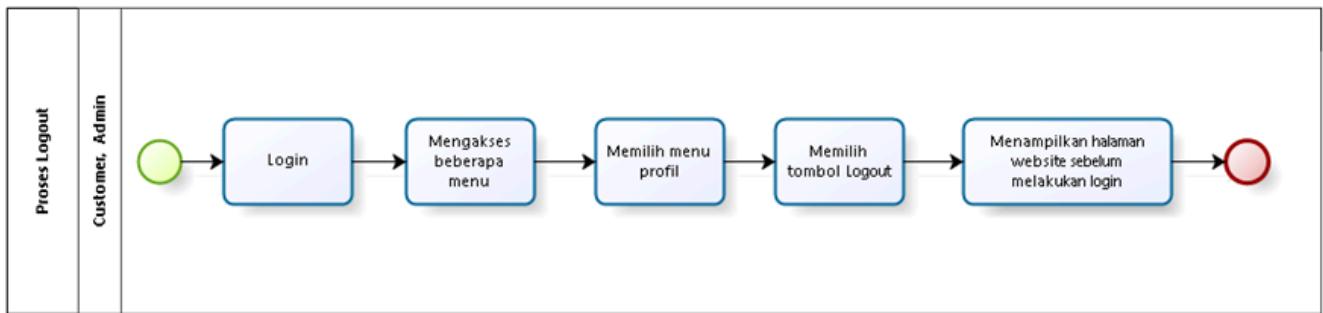
2.2.19 Proses Bisnis Menghapus Berita terkini

Pada Proses Bisnis berita terkini, admin melakukan login terlebih dahulu. Kemudian admin memilih menu news dan memilih sub-menu semua news. Setelah itu, Admin memilih news dan memilih tombol hapus



2.2.20 Proses Bisnis Logout

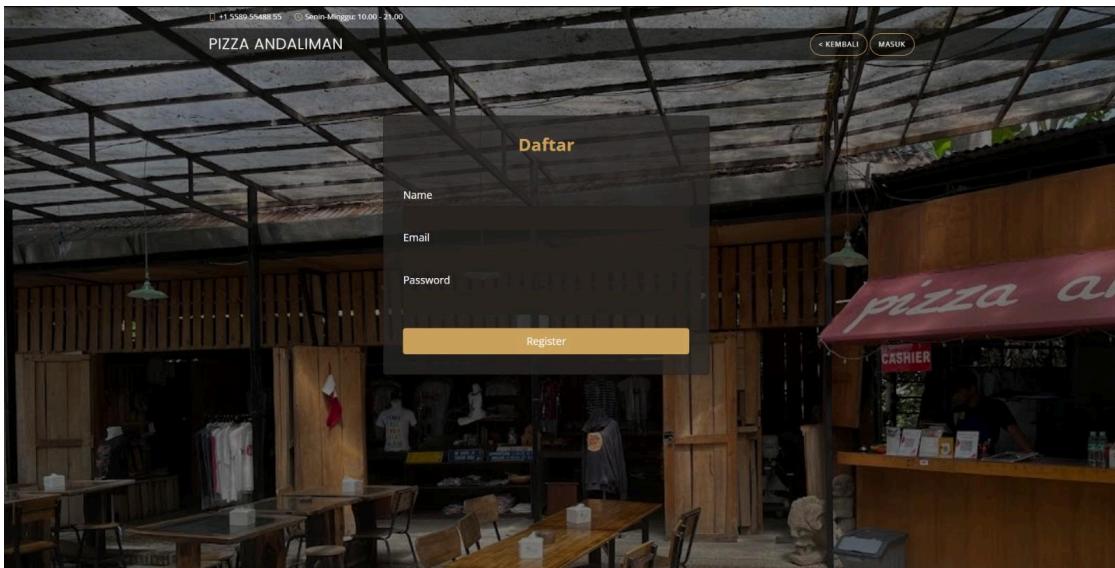
Pada proses bisnis *Logout*, baik Customer maupun Admin harus melakukan Login terlebih dahulu. Mengakses beberapa menu web, setelah itu memilih menu profil dan memilih tombol *logout*. Sistem akan menampilkan halaman website sebelum melakukan Login.



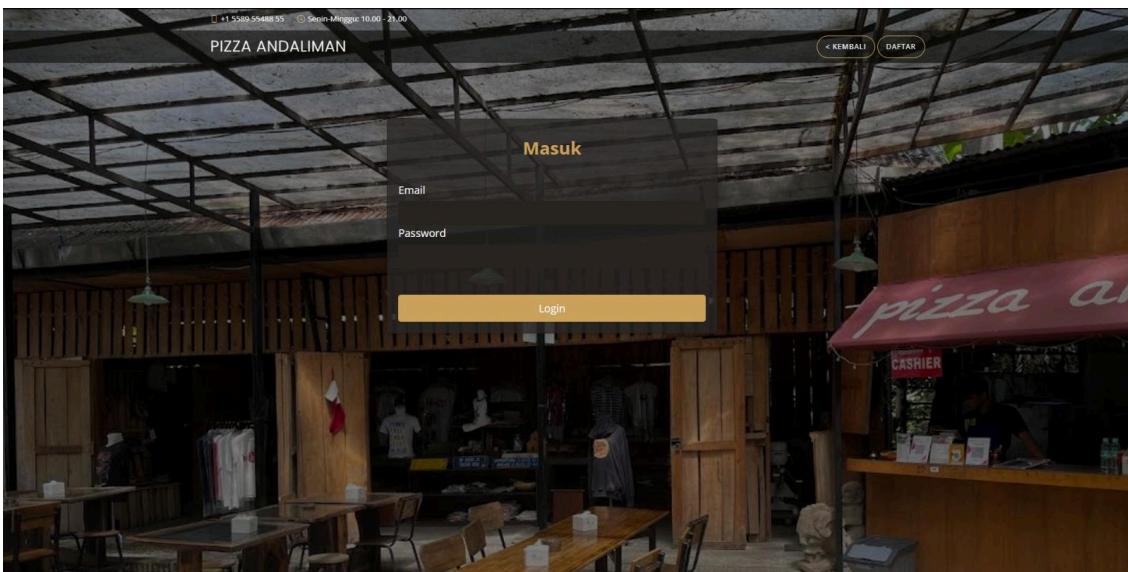
3 Tampilan Sistem

Pada bab 3 berisi tentang tampilan *website* yang telah dibangun.

1. Tampilan halaman registrasi



2. Tampilan halaman login



3. Tampilan halaman produk

SEMUA PRODUK
Produk

SEMUA KATEGORI ▾

Search here... 🔍

soft drink Ice Lemon Tea Rp 20.000	snack Kentang Goreng Rp 20.000	food Nasi Putih Rp 6.000
pizza Pizza Andaliman (Large) Rp 80.000	pizza Pizza Andaliman (Regular) Rp 65.000	soft drink Teh Manis Dingin Rp 10.000

4. Tampilan halaman reservasi

RESERVATION
Book a Table

Nama:

Email:

Telepon:

Tanggal: dd/mm/yyyy 📅

Waktu: -:- 🕒

Jumlah Orang:

Pesan Tambahan:

SUBMIT

5. Tampilan halaman kategori



6. Tampilan halaman feedback



7. Tampilan halaman form feedback

— FEEDBACK —

Berikan juga tanggapan anda

Pesan Kamu:

SUBMIT

8. Tampilan halaman galeri



9. Tampilan halaman menambahkan produk

Pages / Add Produk

informasi input

Nama Produk	Nama Produk
Harga Produk	Harga
Stok Produk	quantity
Deskripsi Produk	deskripsi
Pilih Categori	Select Category
Kirim Gambar	<input type="button" value="Choose File"/> No file chosen
<input type="button" value="Add product"/>	

10. Tampilan halaman mengedit produk

Pages / Edit Product

Input Information

Product Name	nasi putih
Price	6000
Quantity	1000
Description	nasi putih
Category	Pizza
Current Image	
Upload New Image	<input type="button" value="Choose File"/> No file chosen
<input type="button" value="Update Product"/>	

11. Tampilan halaman menambah kategori

Pages / Add Categori

Tambah Kategori Baru

Nama Category

informasi input

12. Tampilan halaman mengedit kategori

Pages / Edit Category

EDIT Kategori

Nama Category

informasi input

13. Tampilan halaman menambah galeri

Pages / Add Gallery

Tambah Gallery Baru

Judul Gallery

informasi input

Deskripsi

Kirim Gambar

No file chosen

14. Tampilan halaman menambah berita terkini

The screenshot shows a form titled 'Pages / Add News'. At the top right, there is a link labeled 'informasi input'. The form has two input fields: 'Judul' (Title) and 'Isi' (Content). The 'Isi' field contains placeholder text 'Berikan news terbaru..'. Below the fields is a red 'Send' button.

15. Tampilan halaman berita terkini

The screenshot shows a table titled 'Pages / All News' with a header 'All Categories'. The table has columns: 'ID', 'Judul', 'Isi', and 'Action'. There is one row of data: ID 1, Judul 'Diskon', Isi 'Terdapat diskon 50% setiap pembelian khusus hari ini.', and Action buttons 'Edit' and 'Delete'.

All Categories			
ID	Judul	Isi	Action
1	Diskon	Terdapat diskon 50% setiap pembelian khusus hari ini.	Edit Delete

4 Spesifikasi Database

Database yang dipakai dalam *website* Pizza Andaliman Balige sebagai berikut:

- a. users digunakan untuk menyimpan data admin dan data pelanggan

microservices_auth users	
#	id : bigint(20) unsigned
④	name : varchar(255)
④	email : varchar(255)
④	password : longtext
◆	role : enum('admin','customer')

- b. carts digunakan untuk menambahkan produk ke dalam halaman keranjang

microservices_cart carts	
#	id : bigint(20) unsigned
#	product_id : int(11)
④	product_name : varchar(255)
④	product_image : varchar(255)
#	quantity : int(11)
#	price : int(11)
#	total : int(11)
#	user_id : int(11)

- c. categories digunakan admin untuk mengelola kategori

microservices_category categories	
•	id : bigint(20) unsigned
•	created_at : datetime(3)
•	updated_at : datetime(3)
•	deleted_at : datetime(3)
•	name : varchar(255)

d. feedbacks digunakan untuk menambahkan feedback bagi *customer*

microservices_feedback feedbacks	
•	id : bigint(20) unsigned
•	created_at : datetime(3)
•	updated_at : datetime(3)
•	deleted_at : datetime(3)
#	user_id : bigint(20)
•	content : varchar(255)

e. galleries digunakan admin untuk mengelola galeri

microservices_gallery galleries	
•	id : bigint(20) unsigned
•	created_at : datetime(3)
•	updated_at : datetime(3)
•	deleted_at : datetime(3)
•	title : varchar(255)
•	description : varchar(255)
•	image : varchar(255)

f. hot_news digunakan admin untuk mengelola berita terkini

microservices_news hot_news	
id	: bigint(20) unsigned
created_at	: datetime(3)
updated_at	: datetime(3)
deleted_at	: datetime(3)
title	: varchar(255)
content	: text

g. order_items dan orders digunakan saat melakukan pemesanan



h. products digunakan oleh admin untuk mengelola produk

microservices_product products	
#	id : bigint(20) unsigned
□	created_at : datetime(3)
□	updated_at : datetime(3)
□	deleted_at : datetime(3)
□	name : varchar(255)
□	description : varchar(255)
#	price : bigint(20)
#	quantity : bigint(20)
□	image : varchar(255)
#	category_id : bigint(20)

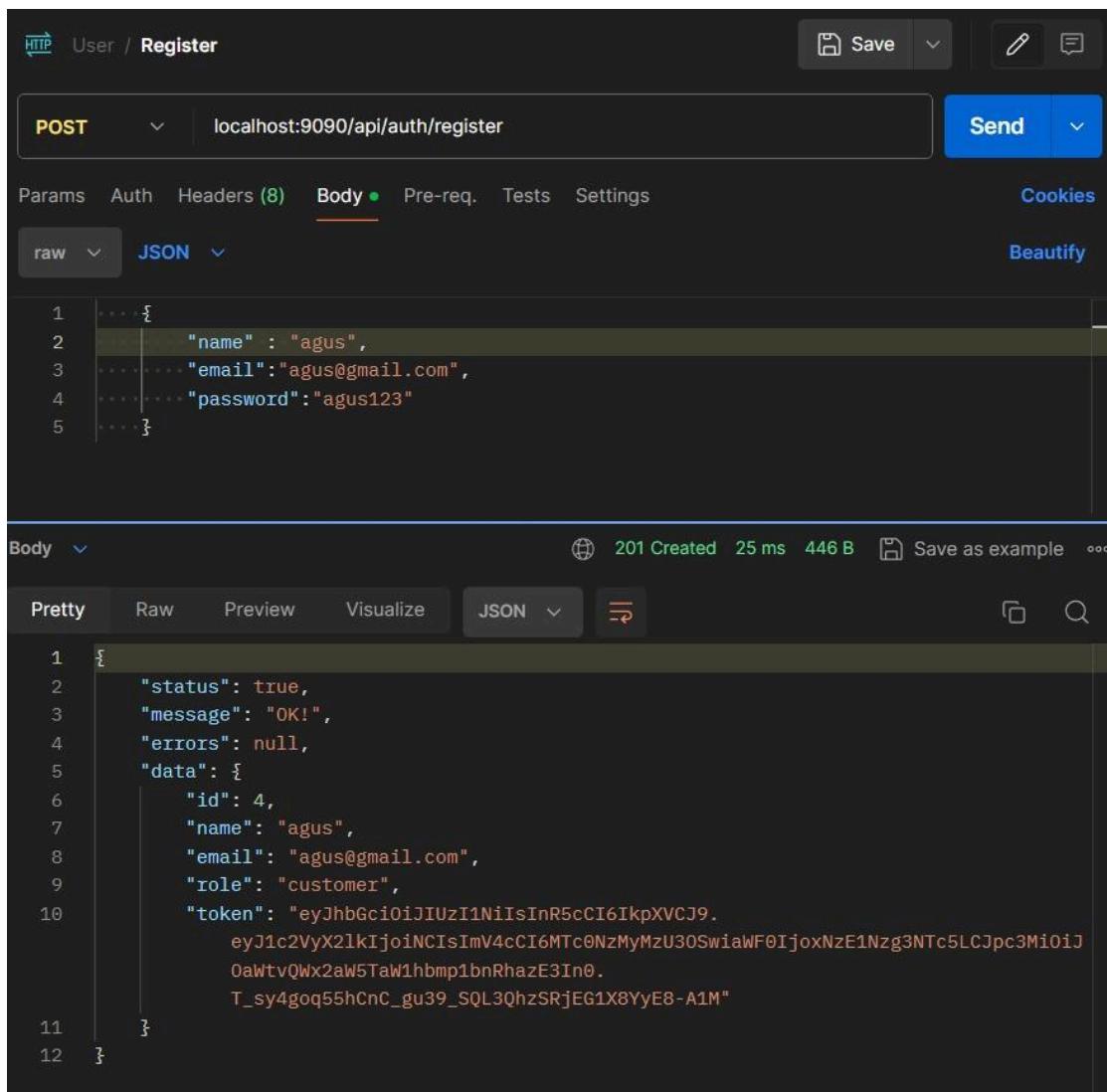
- i. reservations digunakan untuk melakukan reservasi

microservices_reservation reservations	
#	id : bigint(20) unsigned
#	user_id : int(11)
□	name : varchar(255)
□	email : varchar(255)
□	phone : varchar(20)
□	date : date
□	time : datetime(3)
#	people : int(11)
□	message : text
◆	status : enum('pending','approved','rejected','canceled')

5 Pengujian Sistem

5.1 Pengujian Autentikasi

POST: Register



The screenshot shows the Postman interface for a POST request to `localhost:9090/api/auth/register`. The request body is JSON, containing:

```
1  ...
2  "name": "agus",
3  "email": "agus@gmail.com",
4  "password": "agus123"
5 }
```

The response status is `201 Created` with a response time of `25 ms` and a size of `446 B`. The response body is:

```
1 {
2   "status": true,
3   "message": "OK!",
4   "errors": null,
5   "data": {
6     "id": 4,
7     "name": "agus",
8     "email": "agus@gmail.com",
9     "role": "customer",
10    "token": "eyJ1c2VyX2lkIjoiNCIsImV4cCI6MTc0NzMyMzU3OSwiWF0IjoxNzE1Nzg3NTc5LCJpc3MiOiJ
11    OaWtvQWx2aW5TaW1hbmp1bnRhazE3In0.
12    T_sy4goq55hCnC_gu39_SQL3QhzSRjEG1X8YyE8-A1M"
13  }
14 }
```

POST: Login

The screenshot shows the Postman interface for a POST request to `localhost:9090/api/auth/login`. The request body is JSON, containing:

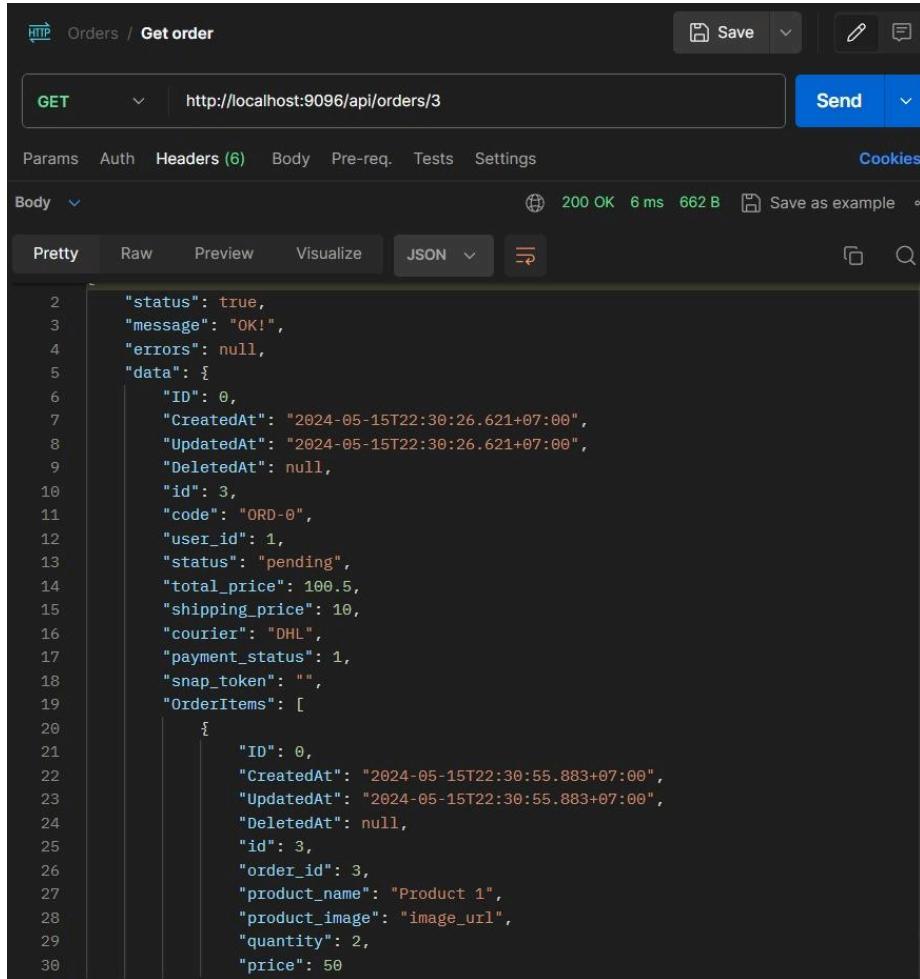
```
1  ....{
2  ....  "email": "agus@gmail.com",
3  ....  "password": "agus123"
4  .... }
```

The response status is 200 OK, with a response time of 12 ms and a size of 441 B. The response body is:

```
1 {
2   "status": true,
3   "message": "OK!",
4   "errors": null,
5   "data": {
6     "id": 4,
7     "name": "agus",
8     "email": "agus@gmail.com",
9     "role": "customer",
10    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJ1c2VyX2lkIjoiNCIsImV4cCI6MTc0NzMyMzYwNyviaWF0IjoxNzE1Nzg3NjA3LCJpc3MiOiJ
OaWtvQWx2aW5TaW1hbmp1bnRhazE3In0.
S9hM_P6S1bulbeAiVIoo7vDaskIrEMI0BVkuLJN-LYc"
```

5.2 Pengujian Orders

GET: Orders



The screenshot shows the Postman application interface. At the top, it says "HTTP Orders / Get order". Below that is a search bar with "GET" and the URL "http://localhost:9096/api/orders/3". To the right of the URL is a "Send" button. Above the main content area, there are tabs for "Params", "Auth", "Headers (6)", "Body", "Pre-req.", "Tests", and "Settings". The "Headers" tab is selected. Below the tabs, it says "Body" with a dropdown menu. To the right of the body section, it shows "200 OK 6 ms 662 B" and a "Save as example" button. The main content area displays a JSON response with line numbers from 1 to 38 on the left. The JSON data includes a "status": true message, a "data" object with various attributes like ID, code, user_id, status, total_price, shipping_price, courier, payment_status, snap_token, and OrderItems. Each OrderItem has attributes like ID, order_id, product_name, product_image, quantity, and price.

```
2   "status": true,
3   "message": "OK!",
4   "errors": null,
5   "data": {
6     "ID": 0,
7     "CreatedAt": "2024-05-15T22:30:26.621+07:00",
8     "UpdatedAt": "2024-05-15T22:30:26.621+07:00",
9     "DeletedAt": null,
10    "id": 3,
11    "code": "ORD-0",
12    "user_id": 1,
13    "status": "pending",
14    "total_price": 100.5,
15    "shipping_price": 10,
16    "courier": "DHL",
17    "payment_status": 1,
18    "snap_token": "",
19    "OrderItems": [
20      {
21        "ID": 0,
22        "CreatedAt": "2024-05-15T22:30:55.883+07:00",
23        "UpdatedAt": "2024-05-15T22:30:55.883+07:00",
24        "DeletedAt": null,
25        "id": 3,
26        "order_id": 3,
27        "product_name": "Product 1",
28        "product_image": "image_url",
29        "quantity": 2,
30        "price": 50
31      }
32    ]
33  }
34}
```

POST: Orders

The screenshot shows the Postman interface for a POST request to `http://localhost:9096/api/orders`. The request method is set to `POST`. The request body is a JSON object:

```
1 {
2   "user_id": 1,
3   "status": "pending",
4   "total_price": 100.50,
5   "shipping_price": 10.00,
6   "payment_status": 1,
7   "courier": "DHL"
8 }
```

The response status is `201 Created` with a response time of `12 ms` and a size of `457 B`. The response body is:

```
2 {
3   "status": true,
4   "message": "OK!",
5   "errors": null,
6   "data": {
7     "ID": 0,
8     "CreatedAt": "2024-05-15T22:30:26.621+07:00",
9     "UpdatedAt": "2024-05-15T22:30:26.621+07:00",
10    "DeletedAt": null,
11    "id": 3,
12    "code": "ORD-0",
13    "user_id": 1,
14    "status": "pending",
15    "total_price": 100.5,
16    "shipping_price": 10,
17    "courier": "DHL",
18    "payment_status": 1,
19    "snap_token": "",
20    "orderItems": null
21 }
```

DELETE: Orders

The screenshot shows the Postman interface for a DELETE request to `http://localhost:9096/api/orders/0`. The request method is set to `DELETE`. The request body is a JSON object:

```
1 {
2   "status": true,
3   "message": "Deleted",
4   "errors": null,
5   "data": {}
6 }
```

POST: Order-items

The screenshot shows the Postman application interface. At the top, it says "Orders / Post order-items". Below that, the method is set to "POST" and the URL is "http://localhost:9096/api/order-items". The "Body" tab is selected, showing the following JSON payload:

```
1 {
2   "order_id": 3,
3   "product_name": "Product 1",
4   "product_image": "image_url",
5   "quantity": 2,
6   "price": 50
7 }
```

Below the body, the response status is shown as "201 Created" with a timestamp of "5 ms" and a size of "394 B". The response body is displayed in "Pretty" format:

```
1 {
2   "status": true,
3   "message": "OK!",
4   "errors": null,
5   "data": {
6     "ID": 0,
7     "CreatedAt": "2024-05-15T22:30:55.883+07:00",
8     "UpdatedAt": "2024-05-15T22:30:55.883+07:00",
9     "DeletedAt": null,
10    "id": 3,
11    "order_id": 3,
12    "product_name": "Product 1",
13    "product_image": "image_url",
14    "quantity": 2,
15    "price": 50
16  }
17 }
```

5.3 Pengujian Users

GET: User

The screenshot shows the Postman interface with the following details:

- HTTP Method:** GET
- URL:** http://localhost:9090/api/user/profile
- Headers:** Authorization (value: eyJhbGciOiJIUzI1NilsInR5cCl6lkp...)
- Body:** 200 OK 6 ms 242 B
- Response Preview (Pretty):**

```
1 {
2     "status": true,
3     "message": "OK!",
4     "errors": null,
5     "data": {
6         "id": 4,
7         "name": "agus",
8         "email": "agus@gmail.com",
9         "role": "customer"
10    }
11 }
```

5.4 Pengujian Reservasi

GET

The screenshot shows the Postman application interface. At the top, there's a header bar with the title "Reservation / Get". Below it, the main interface has a "GET" method selected and the URL "http://localhost:9097/api/reservations?user_id=1". A "Send" button is visible on the right. The "Params" tab is active, showing a table for "Query Params" with one entry: "user_id" with value "1". Other tabs like "Auth", "Headers", "Body", "Pre-req.", "Tests", and "Settings" are also present. Below the params, the "Body" section shows a response with status 200 OK, 15 ms, and 399 B. The response content is displayed in "Pretty" JSON format:

```
1 {
2     "status": true,
3     "message": "OK!",
4     "errors": null,
5     "data": [
6         {
7             "ID": 2,
8             "UserID": 1,
9             "Name": "Niko",
10            "Email": "niko@example.com",
11            "Phone": "1234567890",
12            "Date": "2024-05-14T00:00:00+07:00",
13            "Time": "0001-01-01T00:00:00Z",
14            "People": 4,
15            "Message": "Please prepare a birthday cake.",
16            "Status": "pending"
17        }
18    ]
19 }
```

POST

The screenshot shows the Postman application interface. At the top, there's a header bar with the URL "Reservation / Post" and various icons for saving, editing, and messaging. Below the header is a search bar with the text "POST" and the URL "http://localhost:9097/api/reservations". To the right of the search bar is a large blue "Send" button.

The main area is divided into sections: "Params", "Auth", "Headers (8)", "Body", "Pre-req.", "Tests", and "Settings". The "Body" section is currently active and set to "JSON". It contains the following JSON payload:

```
4     "email": "niko@example.com",
5     "phone": "1234567890",
6     "date": "2024-05-14",
7     "time": "18:00",
8     "people": 4,
9     "message": "Please prepare a birthday cake."
10    }
11
```

Below the JSON input, the "Body" section has tabs for "Pretty", "Raw", "Preview", and "Visualize". The "Pretty" tab is selected, displaying the same JSON structure with line numbers and indentation. Above the "Pretty" tab, there are status indicators: a globe icon, "201 Created", "30 ms", "372 B", and a "Save as example" button.

At the bottom of the interface, there are additional buttons for "Pretty", "Raw", "Preview", "Visualize", "JSON", and a search icon.

PUT

HTTP Reservation / Edit

PUT http://localhost:9097/api/reservations/2

Params Auth Headers (8) Body **JSON** Pre-req. Tests Settings Cookies

raw JSON Beautify

```
1 {  
2     "id": 2,  
3     "name": "Niko Simanjuntak",  
4     "email": "niko@example.com",  
5     "phone": "081260757573",  
6     "date": "2024-05-17",  
7     "time": "13:00",  
8     "people": 10  
}
```

Body 200 OK 10 ms 375 B Save as example ...

Pretty Raw Preview Visualize JSON

```
1 {  
2     "status": true,  
3     "message": "OK!",  
4     "errors": null,  
5     "data": {  
6         "ID": 2,  
7         "UserID": 0,  
8         "Name": "Niko Simanjuntak",  
9         "Email": "niko@example.com",  
10        "Phone": "081260757573",  
11        "Date": "2024-05-17",  
12        "Time": "13:00",  
13        "People": 10,  
14        "Message": "Pesta Ulang Tahun Indoor",  
15        "Status": "pending"  
16    }  
17 }
```

DELETE

HTTP Reservation / Delete

DELETE http://localhost:9097/api/reservations/2

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

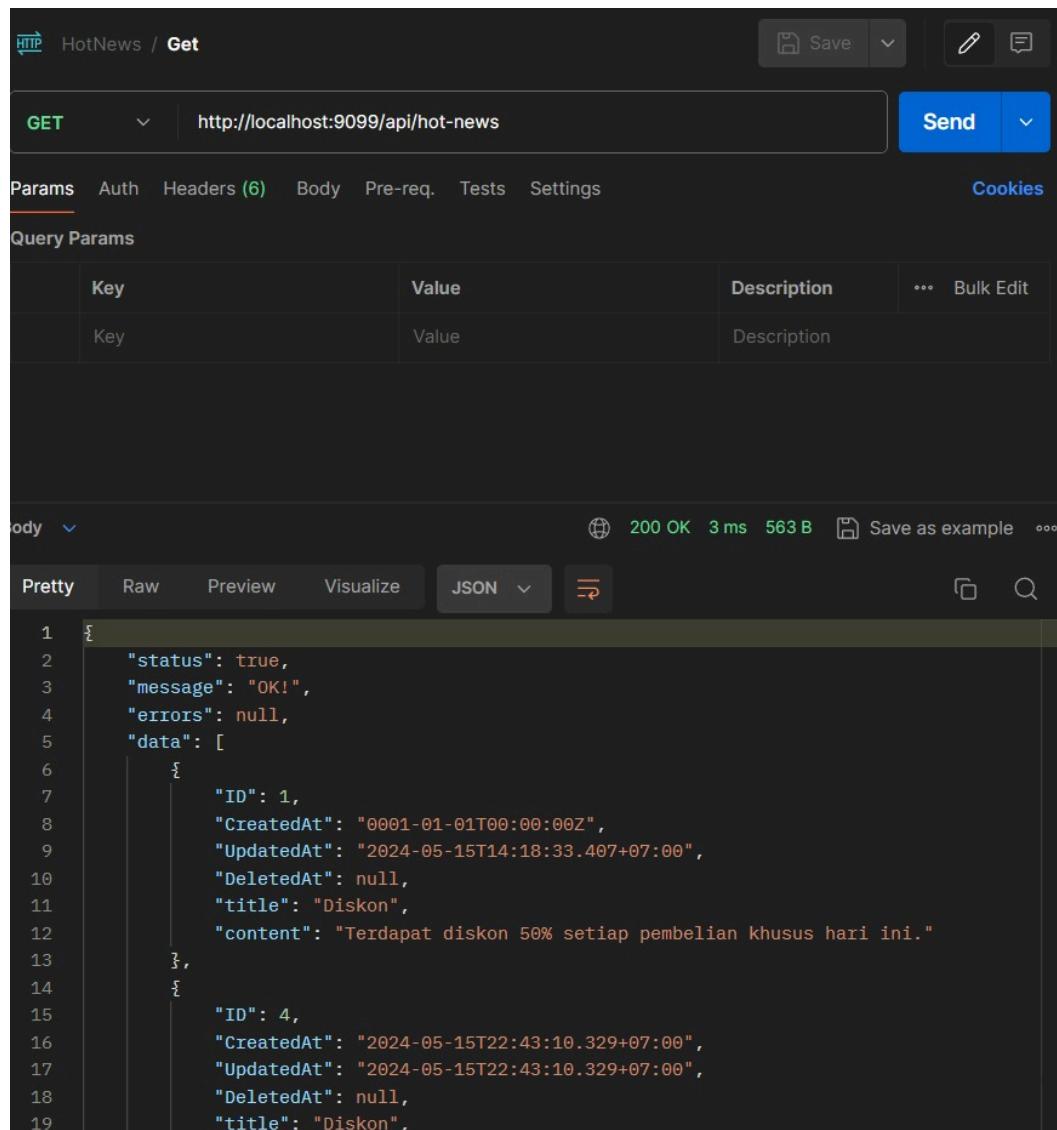
Body 200 OK 10 ms 182 B Save as example ...

Pretty Raw Preview Visualize JSON

```
1 {  
2     "status": true,  
3     "message": "Deleted",  
4     "errors": null,  
5     "data": {}  
6 }
```

5.5 Pengujian Berita Terkini

GET



The screenshot shows the Postman application interface. At the top, it displays the URL `http://localhost:9099/api/hot-news`. Below the URL, there are tabs for **Params**, **Auth**, **Headers (6)**, **Body**, **Pre-req.**, **Tests**, and **Settings**. The **Body** tab is currently selected. Under the **Body** tab, there is a table titled "Query Params" with columns for Key, Value, Description, and Bulk Edit. The table is empty. At the bottom of the body section, there is a JSON response preview. The response status is 200 OK, with a response time of 3 ms and a size of 563 B. The JSON content is as follows:

```
1 {  
2   "status": true,  
3   "message": "OK!",  
4   "errors": null,  
5   "data": [  
6     {  
7       "ID": 1,  
8       "CreatedAt": "0001-01-01T00:00:00Z",  
9       "UpdatedAt": "2024-05-15T14:18:33.407+07:00",  
10      "DeletedAt": null,  
11      "title": "Diskon",  
12      "content": "Terdapat diskon 50% setiap pembelian khusus hari ini."  
13    },  
14    {  
15      "ID": 4,  
16      "CreatedAt": "2024-05-15T22:43:10.329+07:00",  
17      "UpdatedAt": "2024-05-15T22:43:10.329+07:00",  
18      "DeletedAt": null,  
19      "title": "Diskon",  
20    }  
21  ]  
22}  
23
```

POST

The screenshot shows the Postman interface for a POST request to `http://localhost:9099/api/hot-news`. The request body is defined as follows:

```
1 {
2   "title": "Diskon",
3   "content": "Terdapat diskon 10% setiap pembelian khusus hari ini.",
4   "author": "Pizza Andaliman",
5   "published_at": "2024-05-15T12:00:00Z"
6 }
7
```

The response details are as follows:

- Status: 201 Created
- Time: 25 ms
- Size: 378 B
- Save as example

The response body is displayed in Pretty format:

```
1 {
2   "status": true,
3   "message": "OK!",
4   "errors": null,
5   "data": {
6     "ID": 4,
7     "CreatedAt": "2024-05-15T22:43:10.329+07:00",
8     "UpdatedAt": "2024-05-15T22:43:10.329+07:00",
9     "DeletedAt": null,
10    "title": "Diskon",
11    "content": "Terdapat diskon 10% setiap pembelian khusus hari ini."
12 }
```

PUT

The screenshot shows the Postman interface for a PUT request. The URL is `http://localhost:9099/api/hot-news/4`. The request body is a JSON object:

```
1 {
2     "ID": 4,
3     "title": "Diskon",
4     "content": "Terdapat diskon 40% setiap pembelian khusus hari ini.",
5     "author": "Pizza Andaliman",
6     "published_at": "2024-05-15T12:00:00Z"
7 }
```

The response status is 200 OK, with a response body:

```
1 {
2     "status": true,
3     "message": "OK!",
4     "errors": null,
5     "data": {
6         "ID": 4,
7         "CreatedAt": "2001-01-01T00:00:00Z",
8         "UpdatedAt": "2024-05-15T22:44:30.354+07:00",
9         "DeletedAt": null,
10        "title": "Diskon",
11        "content": "Terdapat diskon 40% setiap pembelian khusus hari ini."
12    }
13 }
```

DELETE

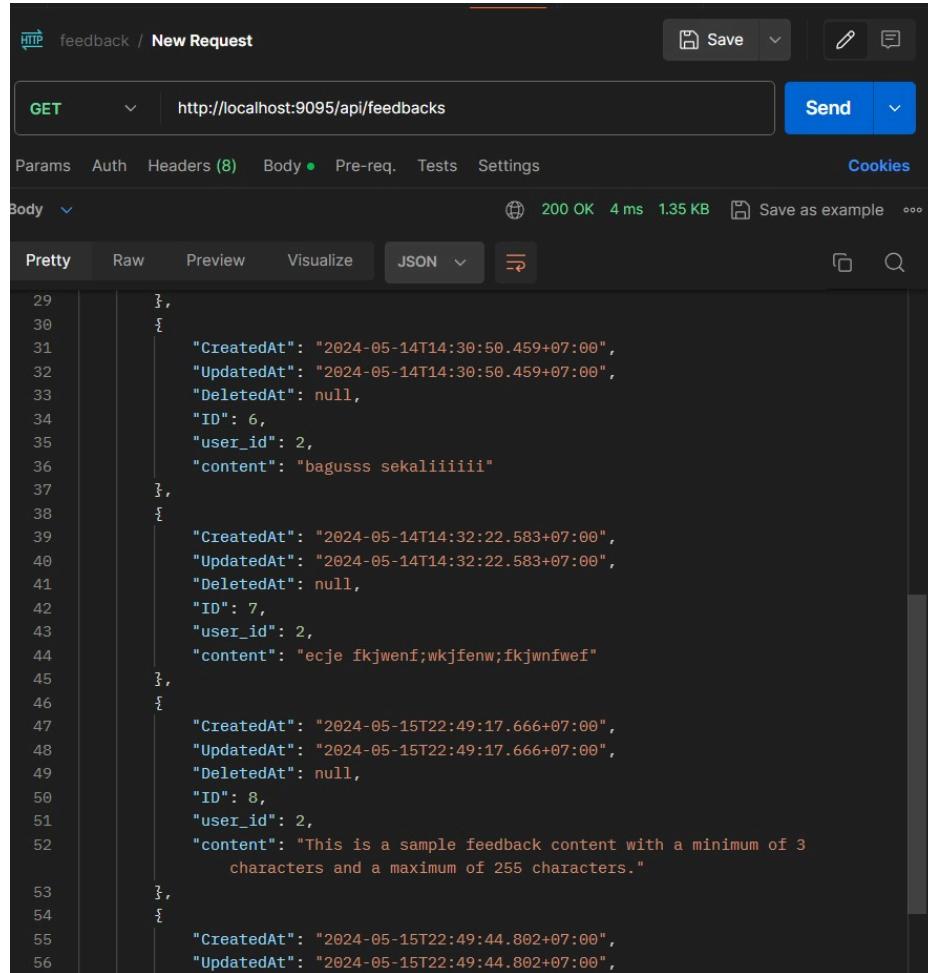
The screenshot shows the Postman interface for a DELETE request. The URL is `http://localhost:9099/api/hot-news/4`. The request body is empty.

The response status is 200 OK, with a response body:

```
1 {
2     "status": true,
3     "message": "Deleted",
4     "errors": null,
5     "data": {}
6 }
```

5.6 Pengujian Feedback

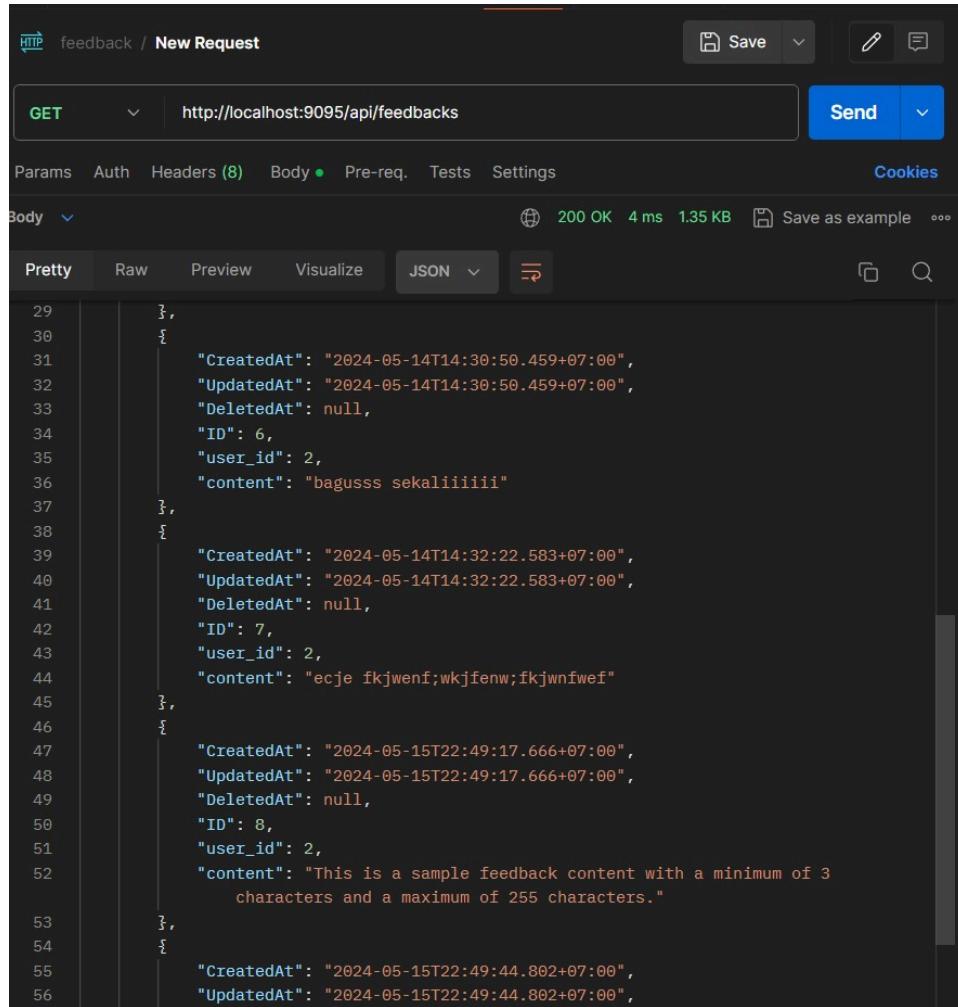
GET



The screenshot shows the Postman application interface. At the top, it says "feedback / New Request". Below that, there's a dropdown menu set to "GET" and the URL "http://localhost:9095/api/feedbacks". To the right of the URL are "Save", "Edit", and "Send" buttons. Underneath the URL, there are tabs for "Params", "Auth", "Headers (8)", "Body", "Pre-req.", "Tests", and "Settings". The "Body" tab is selected and has a sub-tab "JSON". The main content area displays a JSON response with line numbers from 29 to 56 on the left. The JSON data represents three feedback entries:

```
29 },
30 {
31     "CreatedAt": "2024-05-14T14:30:50.459+07:00",
32     "UpdatedAt": "2024-05-14T14:30:50.459+07:00",
33     "DeletedAt": null,
34     "ID": 6,
35     "user_id": 2,
36     "content": "bagusss sekaliiiiii"
37 },
38 {
39     "CreatedAt": "2024-05-14T14:32:22.583+07:00",
40     "UpdatedAt": "2024-05-14T14:32:22.583+07:00",
41     "DeletedAt": null,
42     "ID": 7,
43     "user_id": 2,
44     "content": "ecje fkjwenf;wkjfew;fkjwnfwef"
45 },
46 {
47     "CreatedAt": "2024-05-15T22:49:17.666+07:00",
48     "UpdatedAt": "2024-05-15T22:49:17.666+07:00",
49     "DeletedAt": null,
50     "ID": 8,
51     "user_id": 2,
52     "content": "This is a sample feedback content with a minimum of 3
53         characters and a maximum of 255 characters."
54 },
55 {
56     "CreatedAt": "2024-05-15T22:49:44.802+07:00",
57     "UpdatedAt": "2024-05-15T22:49:44.802+07:00",
```

POST

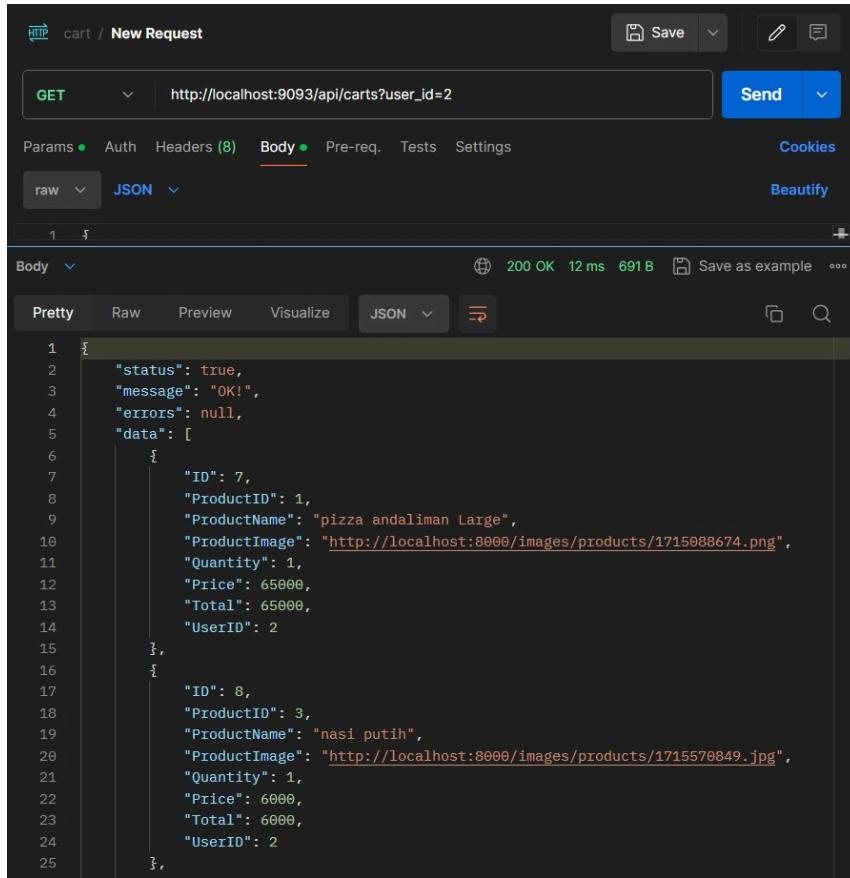


The screenshot shows the Postman application interface. At the top, there's a header bar with the URL "feedback / New Request". Below it, a search bar contains "http://localhost:9095/api/feedbacks". To the right of the search bar are "Save", "Edit", and "Send" buttons. Underneath the search bar, there are tabs for "Params", "Auth", "Headers (8)", "Body", "Pre-req.", "Tests", and "Settings". The "Body" tab is selected and has a sub-tab "Pretty" which is also selected. The main content area displays a JSON array of feedback objects, numbered from 29 to 56. Each object has properties like ID, user_id, and content.

```
29: {  
30:   "content": "bagusss sekaliiiii",  
31:   "createdAt": "2024-05-14T14:30:50.459+07:00",  
32:   "updatedAt": "2024-05-14T14:30:50.459+07:00",  
33:   "deletedAt": null,  
34:   "ID": 6,  
35:   "user_id": 2,  
36:   "content": "ecje fkjwenf;wkjfew;fkjwnfwef"  
37: },  
38: {  
39:   "content": "This is a sample feedback content with a minimum of 3  
40:   characters and a maximum of 255 characters.",  
41:   "createdAt": "2024-05-14T14:32:22.583+07:00",  
42:   "updatedAt": "2024-05-14T14:32:22.583+07:00",  
43:   "deletedAt": null,  
44:   "ID": 7,  
45:   "user_id": 2,  
46:   "content": "ecje fkjwenf;wkjfew;fkjwnfwef"  
47: },  
48: {  
49:   "content": "This is a sample feedback content with a minimum of 3  
50:   characters and a maximum of 255 characters.",  
51:   "createdAt": "2024-05-15T22:49:17.666+07:00",  
52:   "updatedAt": "2024-05-15T22:49:17.666+07:00",  
53:   "deletedAt": null,  
54:   "ID": 8,  
55:   "user_id": 2,  
56:   "content": "This is a sample feedback content with a minimum of 3  
   characters and a maximum of 255 characters."  
},  
{  
  "content": "This is a sample feedback content with a minimum of 3  
  characters and a maximum of 255 characters.",  
  "createdAt": "2024-05-15T22:49:44.802+07:00",  
  "updatedAt": "2024-05-15T22:49:44.802+07:00",  
},
```

5.6 Pengujian Carts

GET



HTTP cart / New Request

GET http://localhost:9093/api/carts?user_id=2 Send

Params Auth Headers (8) Body Cookies

raw JSON Beautify

Body 200 OK 12 ms 691 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "status": true,
3   "message": "OK!",
4   "errors": null,
5   "data": [
6     {
7       "ID": 7,
8       "ProductID": 1,
9       "ProductName": "pizza andaliman Large",
10      "ProductImage": "http://localhost:8000/images/products/1715088674.png",
11      "Quantity": 1,
12      "Price": 65000,
13      "Total": 65000,
14      "UserID": 2
15    },
16    {
17      "ID": 8,
18      "ProductID": 3,
19      "ProductName": "nasi putih",
20      "ProductImage": "http://localhost:8000/images/products/1715570849.jpg",
21      "Quantity": 1,
22      "Price": 6000,
23      "Total": 6000,
24      "UserID": 2
25    }
]
```

POST

The screenshot shows a POST request to `http://localhost:9093/api/carts/`. The request body is a JSON object:

```
1 {  
2   "product_id": 3,  
3   "product_image": "https://example.com/images/product1.jpg",  
4   "product_name": "nasi putih",  
5   "quantity": 2,  
6   "price": 50000,  
7   "total": 100000,  
8   "user_id": 2  
9 }
```

The response status is 201 Created with a response time of 80 ms and a body size of 341 B. The response body is:

```
1 {  
2   "status": true,  
3   "message": "OK!",  
4   "errors": null,  
5   "data": {  
6     "ID": 9,  
7     "ProductID": 3,  
8     "ProductName": "nasi putih",  
9     "ProductImage": "https://example.com/images/product1.jpg",  
10    "Quantity": 2,  
11    "Price": 50000,  
12    "Total": 100000,  
13    "UserID": 2  
14  }  
15 }
```

DELETE

The screenshot shows a DELETE request to `http://localhost:9093/api/carts/2`. The request body is a JSON object:

```
1 {
```

The response status is 200 OK with a response time of 4 ms and a body size of 182 B. The response body is:

```
1 {  
2   "status": true,  
3   "message": "Deleted",  
4   "errors": null,  
5   "data": []  
6 }
```

5.7 Pengujian Galeri

POST

The screenshot shows the Postman application interface. At the top, it says "HTTP galeri / New Request". Below that, the method is set to "POST" and the URL is "http://localhost:9094/api/galleries". The "Body" tab is selected, showing a JSON payload:

```
1 {
2   "title": "Resto",
3   "description": "Pesona Resto",
4   "image": "https://example.com/images/gallery1.jpg"
5 }
```

Below the body, the response status is shown as "201 Created" with other details: 29 ms, 398 B. The response body is displayed in "Pretty" format:

```
1 {
2   "status": true,
3   "message": "OK!",
4   "errors": null,
5   "data": {
6     "ID": 0,
7     "CreatedAt": "2024-05-15T22:58:03.373+07:00",
8     "UpdatedAt": "2024-05-15T22:58:03.373+07:00",
9     "DeletedAt": null,
10    "id": 13,
11    "title": "Resto",
12    "description": "Pesona Resto",
13    "image": "https://example.com/images/gallery1.jpg"
14  }
15 }
```

PUT

The screenshot shows the Postman interface with a PUT request to `http://localhost:9094/api/galleries/14`. The request body is a JSON object representing a gallery item:

```
1 {  
2   "id": 14,  
3   "title": "Sample Gallery Title",  
4   "description": "This is a sample description for the gallery item, with at least 3 characters and up to 255 characters.",  
5   "image": "https://example.com/images/gallery1.jpg"  
6 }  
7
```

The response status is 200 OK with 16 ms latency and 490 B size. The response body is:

```
1 {  
2   "status": true,  
3   "message": "OK!",  
4   "errors": null,  
5   "data": {  
6     "ID": 0,  
7     "CreatedAt": "0001-01-01T00:00:00Z",  
8     "UpdatedAt": "2024-05-15T23:00:27.998+07:00",  
9     "DeletedAt": null,  
10    "id": 14,  
11    "title": "Sample Gallery Title",  
12    "description": "This is a sample description for the gallery item, with at least 3 characters and up to 255 characters.",  
13    "image": "https://example.com/images/gallery1.jpg"  
14  }  
15 }
```

DELETE

The screenshot shows the Postman interface with a DELETE request to `http://localhost:9094/api/galleries/14`. The request body is identical to the PUT request:

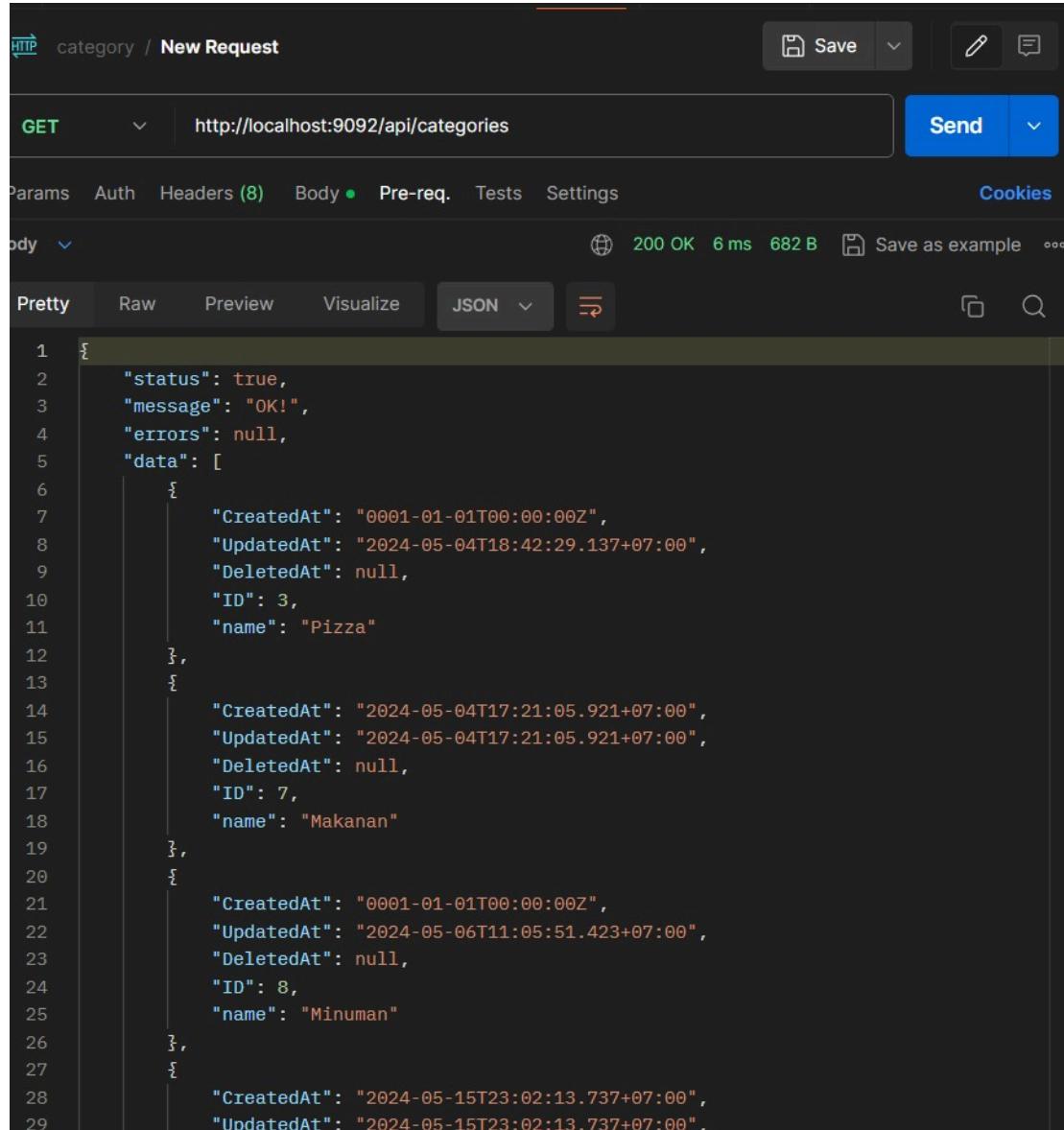
```
1 {  
2   "id": 14,  
3   "title": "Sample Gallery Title",  
4   "description": "This is a sample description for the gallery item, with at least 3 characters and up to 255 characters.",  
5   "image": "https://example.com/images/gallery1.jpg"  
6 }  
7
```

The response status is 200 OK with 11 ms latency and 182 B size. The response body is:

```
1 {  
2   "status": true,  
3   "message": "Deleted",  
4   "errors": null,  
5   "data": {}  
6 }
```

5.8 Pengujian Kategori

GET



The screenshot shows the Postman interface with a successful API call to `http://localhost:9092/api/categories`. The response body is a JSON object containing status, message, errors, and data. The data array contains four objects representing categories: Pizza, Makanan, Minuman, and an unnamed category with ID 8.

```
1 {
2     "status": true,
3     "message": "OK!",
4     "errors": null,
5     "data": [
6         {
7             "CreatedAt": "0001-01-01T00:00:00Z",
8             "UpdatedAt": "2024-05-04T18:42:29.137+07:00",
9             "DeletedAt": null,
10            "ID": 3,
11            "name": "Pizza"
12        },
13        {
14            "CreatedAt": "2024-05-04T17:21:05.921+07:00",
15            "UpdatedAt": "2024-05-04T17:21:05.921+07:00",
16            "DeletedAt": null,
17            "ID": 7,
18            "name": "Makanan"
19        },
20        {
21            "CreatedAt": "0001-01-01T00:00:00Z",
22            "UpdatedAt": "2024-05-06T11:05:51.423+07:00",
23            "DeletedAt": null,
24            "ID": 8,
25            "name": "Minuman"
26        },
27        {
28            "CreatedAt": "2024-05-15T23:02:13.737+07:00",
29            "UpdatedAt": "2024-05-15T23:02:13.737+07:00"
30        }
31    ]
32}
```

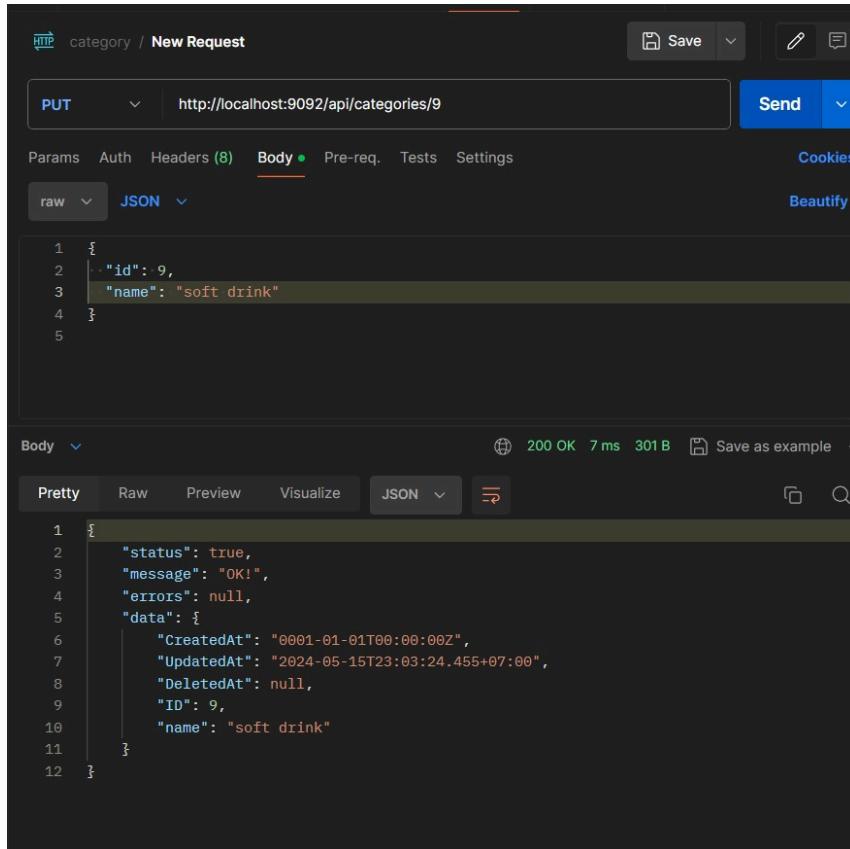
POST

The screenshot shows the Postman application interface. At the top, it says "category / New Request". Below that, the method is set to "POST" and the URL is "http://localhost:9092/api/categories". The "Body" tab is selected, showing a JSON payload:

```
1 {  
2   "name": "Minuman"  
3 }  
4
```

Below the body, the response status is shown as "201 Created" with a timestamp of "2024-05-15T23:02:13.737+07:00", a duration of "29 ms", and a size of "312 B". The response body is displayed in "Pretty" format:1 {
2 "status": true,
3 "message": "OK!",
4 "errors": null,
5 "data": {
6 "CreatedAt": "2024-05-15T23:02:13.737+07:00",
7 "UpdatedAt": "2024-05-15T23:02:13.737+07:00",
8 "DeletedAt": null,
9 "ID": 9,
10 "name": "Minuman"
11 }
12}

PUT



HTTP category / New Request

PUT http://localhost:9092/api/categories/9

Params Auth Headers (8) Body **JSON** Pre-req. Tests Settings

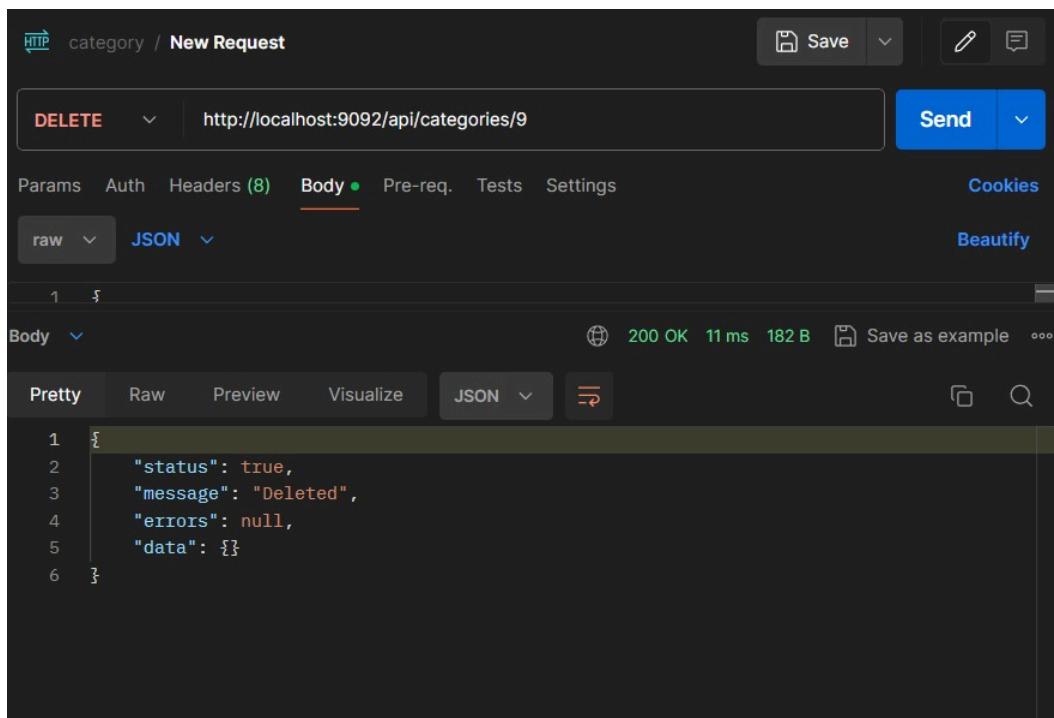
raw JSON Beautify

```
1 {
2   "id": 9,
3   "name": "soft drink"
4 }
```

Body **Pretty** Raw Preview Visualize JSON **200 OK** 7 ms 301 B Save as example

```
1 {
2   "status": true,
3   "message": "OK!",
4   "errors": null,
5   "data": {
6     "CreatedAt": "2001-01-01T00:00:00Z",
7     "UpdatedAt": "2024-05-15T23:03:24.455+07:00",
8     "DeletedAt": null,
9     "ID": 9,
10    "name": "soft drink"
11  }
12 }
```

DELETE



HTTP category / New Request

DELETE http://localhost:9092/api/categories/9

Params Auth Headers (8) Body **JSON** Pre-req. Tests Settings

raw JSON Beautify

Body **Pretty** Raw Preview Visualize JSON **200 OK** 11 ms 182 B Save as example

```
1 {
2   "status": true,
3   "message": "Deleted",
4   "errors": null,
5   "data": {}
6 }
```

5.8 Pengujian Produk

POST

The screenshot shows the Postman application interface. At the top, it says "HTTP product / New Request". Below that, the method is set to "POST" and the URL is "http://localhost:9091/api/products". There are buttons for "Save", "Edit", and "Send". The "Body" tab is selected, showing the JSON payload:

```
1 {  
2   "name": "Kopi",  
3   "description": "Kopi Hitam.",  
4   "price": 100000,  
5   "quantity": 10,  
6   "image": "https://example.com/images/product1.jpg",  
7   "category_id": 9  
8 }
```

Below the body, the response is shown with a status of 201 Created, 20 ms, and 432 B. The response body is also displayed in JSON format:

```
1 {  
2   "status": true,  
3   "message": "OK!",  
4   "errors": null,  
5   "data": {  
6     "CreatedAt": "2024-05-15T23:05:35.042+07:00",  
7     "UpdatedAt": "2024-05-15T23:05:35.042+07:00",  
8     "DeletedAt": null,  
9     "ID": 5,  
10    "name": "Kopi",  
11    "description": "Kopi Hitam.",  
12    "price": 100000,  
13    "quantity": 10,  
14    "image": "https://example.com/images/product1.jpg",  
15    "category_id": 9  
16  }  
17 }
```

PUT

The screenshot shows the Postman interface with a PUT request to `http://localhost:9091/api/products/5`. The request body is a JSON object representing a product:

```
1 {  
2   "id": 5,  
3   "name": "Kopi Andaliman",  
4   "description": "Kopi dengan rasa Andaliman.",  
5   "price": 100000,  
6   "quantity": 10,  
7   "image": "https://example.com/images/product1.jpg",  
8   "category_id": 9  
}
```

The response status is 200 OK with a response body:

```
1 {  
2   "status": true,  
3   "message": "OK!",  
4   "errors": null,  
5   "data": {  
6     "CreatedAt": "2001-01-01T00:00:00Z",  
7     "UpdatedAt": "2024-05-15T23:06:32.491+07:00",  
8     "DeletedAt": null,  
9     "ID": 5,  
10    "name": "Kopi Andaliman",  
11    "description": "Kopi dengan rasa Andaliman.",  
12    "price": 100000,  
13    "quantity": 10,  
14    "image": "https://example.com/images/product1.jpg",  
15    "category_id": 9  
16  }  
17 }
```

DELETE

The screenshot shows the Postman interface with a DELETE request to `http://localhost:9091/api/products/5`. The request body is a JSON object:

```
1 {  
2 }
```

The response status is 200 OK with a response body:

```
1 {  
2   "status": true,  
3   "message": "Deleted",  
4   "errors": null,  
5   "data": {}  
6 }
```