Tugas Implementasi Awal
II3160 Teknologi Sistem Terintegrasi

Dosen Pembimbing:
Daniel Wiyogo Dwiputro, S.T., M.T

Dibuat oleh:
Rasyid Rizky Susilo Nurdwiputro (18223114)

Program Studi Sistem dan Teknologi Informasi
STEI - ITB

Jl. Ganesa No.10, Lb. Siliwangi, Kecamatan Coblong,
Kota Bandung, Jawa Barat 40132

Daftar Isi

BAB I Pendahuluan

1.1 Deskripsi Dokumen

Dokumen ini berisi proses penerjemahan model domain, khususnya Reservation Management Aggregate, ke dalam bentuk implementasi perangkat lunak menggunakan kerangka kerja FastAPI. Dokumen ini menjembatani hasil perancangan domain pada tahap sebelumnya (analisis subdomain, bounded context, aggregate, dan class diagram) dengan tahap implementasi teknis yang diwujudkan dalam bentuk API operasional.

Secara khusus, dokumen ini menjelaskan bagaimana komponen-komponen domain seperti Entity, Value Object, Aggregate Root, dan Domain Event direpresentasikan dalam kode Python melalui pendekatan tactical design dari Domain-Driven Design (DDD). Selanjutnya, dokumen ini juga memaparkan implementasi layanan (API) yang dibangun berdasarkan model tersebut, meliputi API reservasi sebagai layanan utama, sementara aspek pembayaran dan notifikasi direpresentasikan secara konseptual melalui struktur data dan simulasi integrasi.

Dokumen ini juga menyertakan instruksi teknis mengenai cara menjalankan aplikasi FastAPI, dependensi yang diperlukan, serta tautan menuju repositori Git yang berisi kode lengkap implementasi. Dengan demikian, dokumen ini berfungsi sebagai panduan komprehensif yang menghubungkan analisis domain konseptual dengan implementasi perangkat lunak yang dapat dijalankan.

1.2 Tujuan Penulisan Dokumen

Dokumen ini disusun untuk menjelaskan secara sistematis proses implementasi model domain ke dalam bentuk layanan perangkat lunak berbasis API menggunakan pendekatan Domain-Driven Design (DDD) dan framework FastAPI. Adapun tujuan utama dari dokumen ini meliputi:

1. **Menerjemahkan model domain ke dalam kode program**
   Menyajikan bagaimana konsep-konsep domain seperti Entity, Value Object, Aggregate, dan Domain Event direalisasikan dalam kode Python. Implementasi ini memastikan bahwa struktur dan perilaku domain tetap konsisten dengan aturan bisnis yang telah dirumuskan pada tahap analisis.

2. **Membangun API fungsional berdasarkan core domain**
   Mengembangkan serangkaian endpoint utama pada konteks Reservation Management yang mencakup pembuatan reservasi, pemrosesan status, pembayaran deposit, serta penyiapan data

untuk integrasi notifikasi dan pembayaran.. API yang dihasilkan menjadi representasi teknis dari model domain yang telah dirancang.

3. **Menyediakan dokumentasi struktur dan arsitektur aplikasi**
   Menjelaskan desain arsitektur backend, struktur folder, dependensi internal, serta pemisahan tanggung jawab antar layer. Dokumentasi ini memudahkan pengembangan, pemeliharaan, dan perluasan sistem ke depannya.

4. **Menyediakan bukti implementasi melalui repositori Git**
   Mengumpulkan dan menyajikan tautan repositori Git sebagai bukti nyata implementasi kode, termasuk commit yang menunjukkan pembangunan API awal dan struktur domain.

Dengan adanya tujuan-tujuan ini, dokumen ini diharapkan mampu menjadi referensi lengkap yang menghubungkan aspek konseptual domain model dengan implementasi teknis yang dapat dijalankan dan dievaluasi.

1.3 Ruang Lingkup

Ruang lingkup dokumen ini difokuskan pada proses implementasi teknis dari model domain Reservation Management Context ke dalam bentuk layanan API berbasis FastAPI. Lingkup pembahasan tidak mencakup seluruh fungsi operasional restoran, tetapi dibatasi pada konteks inti (core context) yang diperlukan untuk memenuhi kebutuhan sistem. Adapun ruang lingkup yang dicakup dalam dokumen ini meliputi:

1. **Penerjemahan Model Domain (Aggregate) ke Implementasi Kode**
   Dokumen ini hanya memodelkan dan mengimplementasikan aggregate utama, yaitu Reservation Aggregate, beserta entitas, value object, dan domain event yang relevan dalam konteks tersebut. Konteks lain seperti Table Management, Customer Management, atau Menu & Inventory tidak diimplementasikan secara penuh, kecuali dalam bentuk data atau referensi sederhana.

2. **Pengembangan API inti menggunakan FastAPI**
   Fokus implementasi terdapat pada pembangunan API dasar yang merepresentasikan operasi-operasi utama dalam domain reservasi, seperti:
   - pembuatan reservasi,
   - validasi kebijakan,
   - simulasi pemrosesan status pembayaran (melalui stub),

- serta referensi data pelanggan dan persiapan trigger notifikasi tanpa implementasi layanan penuh.

API disusun menggunakan struktur layered architecture yang meliputi domain, service, repository, dan API layer.

3. **Implementasi Persistensi Data Menggunakan PostgreSQL**
   Penyimpanan data diimplementasikan menggunakan basis data relasional PostgreSQL. Interaksi antara aplikasi dan basis data dikelola menggunakan SQLAlchemy ORM yang memetakan objek domain ke dalam tabel relasional. Hal ini memastikan durabilitas data dan integritas referensial antar entitas.

4. **Penyediaan Repositori Git sebagai Bukti Implementasi**
   Hasil implementasi disertakan dalam bentuk tautan repositori Git yang memuat struktur kode lengkap dan commit historis yang menunjukkan proses pembuatan API awal sesuai instruksi tugas.

Dengan batasan ruang lingkup tersebut, dokumen ini menyediakan fokus pada penerapan DDD Tactical Design secara nyata dalam pembangunan API inti.

1.4 Definisi dan Istilah

Bagian ini menyajikan istilah-istilah penting yang digunakan dalam dokumen ini. Tujuannya adalah untuk memastikan kesamaan pemahaman terkait konsep-konsep teknis dan domain yang menjadi dasar implementasi sistem.

Istilah	Definisi
Aggregate	Struktur domain yang terdiri dari satu atau lebih Entity dan Value Object yang diperlakukan sebagai satu kesatuan logis. Aggregate memiliki satu Aggregate Root yang bertanggung jawab menjaga konsistensi data dan mengatur aturan bisnis internal.
Entity	Objek dalam domain yang memiliki identitas unik (ID) dan dapat mengalami perubahan keadaan sepanjang siklus hidupnya. Dalam konteks ini, Reservation, TableAssignment, dan Waitlist adalah contoh entity yang relevan.
Value Object	Objek tanpa identitas khusus yang merepresentasikan nilai atau karakteristik tertentu. Value Object bersifat immutabel dan digunakan untuk memperkuat makna atribut, seperti ReservationTime, ReservationPolicy, dan ContactInfo.

Aggregate Root	Entitas utama dalam sebuah aggregate yang menjadi titik akses bagi entitas lainnya. Dalam konteks ini, Reservation bertindak sebagai aggregate root yang mengoordinasikan struktur internal dan aturan bisnis reservasi.
Domain Event	Objek yang merepresentasikan kejadian penting dalam domain dan dipublikasikan setelah aggregate melakukan perubahan signifikan. Contohnya adalah event seperti ReservationCreated, ReservationConfirmed, dan PaymentCompleted, yang dapat digunakan untuk berkomunikasi dengan bounded context lain seperti Payment atau Notification.
Repository	Abstraksi yang menyediakan mekanisme penyimpanan dan pengambilan aggregate dari sumber data. Pada implementasi ini, repository menggunakan struktur penyimpanan database untuk mempermudah eksekusi dan pengujian.
API (Application Programming Interface)	Antarmuka yang memungkinkan interaksi pengguna atau sistem lain dengan layanan yang dibangun. API FastAPI digunakan untuk menerima request dari klien, memanggil service layer, dan mengembalikan response sesuai kebutuhan.
FastAPI	Framework modern berbasis Python yang digunakan untuk membangun API dengan performa tinggi. FastAPI mendukung validasi otomatis menggunakan Pydantic dan menghasilkan dokumentasi otomatis melalui Swagger UI.
Swagger UI	Dokumentasi antarmuka interaktif untuk API yang dihasilkan otomatis oleh FastAPI, dapat diakses melalui endpoint /docs.
Bounded Context	Batasan yang mendefinisikan ruang lingkup model yang konsisten dalam DDD. Setiap bounded context memiliki model sendiri, istilah sendiri, dan aturan bisnis sendiri. Dalam proyek ini, fokus implementasi adalah pada Reservation Management Context sebagai core context.

BAB II Pemetaan Model Domain ke Kode

2.1 Ringkasan Model Domain

Model domain dalam proyek ini berfokus pada Reservation Management Context sebagai core domain dalam keseluruhan sistem. Konteks ini dipilih karena memiliki peran strategis dalam mengatur seluruh proses reservasi pelanggan, yang menjadi pusat operasional restoran dan sumber utama nilai bisnis. Model domain yang dirancang sebelumnya menetapkan struktur logis berupa Entity, Value Object, Aggregate, serta Domain Event yang mencerminkan aturan bisnis inti dalam proses reservasi.

Pada tingkat konseptual, proses reservasi melibatkan beberapa komponen penting:

1. **Reservation (Aggregate Root)**
   Merupakan entitas utama yang mewakili satu pemesanan meja oleh pelanggan. Reservation bertanggung jawab menjaga konsistensi seluruh aturan bisnis reservasi, mulai dari validasi waktu, alokasi meja, kebijakan pembatalan, hingga pelacakan status dan pembayaran deposit.

2. **TableAssignment (Entity)**
   Entitas yang mewakili alokasi satu atau lebih meja untuk sebuah reservasi. Entitas ini berada di dalam Reservation Aggregate karena alokasi meja merupakan bagian dari aturan konsistensi reservasi.

3. **ReservationHistory (Entity)**
   Entitas yang menampung riwayat pemesanan meja oleh pelanggan. ReservationHistory bertanggung jawab mencatat waktu pertama kali reservasi dibuat, aksi yang dilakukan, dan catatan khusus.

4. **ReservationStatus, ContactInfo, ReservationTime, CancellationReason, dan ReservationPolicy (Value Object)**
   Kumpulan objek nilai yang digunakan untuk memperkuat makna bisnis atribut-atribut reservasi, seperti waktu, kebijakan durasi, aturan pembatalan, dan detail pembayaran deposit. Value object ini bersifat immutabel untuk menjaga konsistensi data.

Model domain ini dirancang menggunakan prinsip tactical design dari DDD, yang mengutamakan keselarasan antara konsep bisnis dan implementasi teknis. Struktur ini kemudian menjadi dasar untuk menyusun kode Python yang merepresentasikan perilaku domain secara eksplisit melalui aggregate logic, method behavior, dan domain invariants.

Pada bagian selanjutnya, setiap komponen dalam model domain akan dipetakan secara langsung ke implementasi kode di dalam struktur program FastAPI, sehingga hubungan antara model konseptual dan model operasional dapat terlihat dengan jelas.

2.2 Pemetaan Komponen Domain ke Kode

Pada tahap implementasi, seluruh komponen domain diterjemahkan ke dalam kode Python melalui struktur folder yang telah ditetapkan pada konteks Reservation Management. Karena implementasi masih berada pada tingkat dasar (tanpa database dan logika kompleks penuh), pemetaan komponen dilakukan dalam satu tabel terpadu untuk menunjukkan bagaimana model konseptual direpresentasikan dalam kode operasional. Tabel berikut merangkum seluruh komponen domain dan lokasi implementasinya.

Tabel Pemetaan Komponen Domain ke Kode

Komponen Domain	Implementasi Kode	Lokasi	Keterangan
Reservation (Aggregate Root)	class Reservation & class ReservationORM	src/domain/ & src/infrastructure/	Kelas utama yang mengontrol konsistensi data. Memiliki daftar Entities dan Value Objects di dalamnya. Memiliki representasi ganda: Domain Model di domain layer dan ORM Model di infrastructure layer.
TableAssignment (Entity)	class TableAssignment	src/domain/models.py	Entitas anak yang hanya bisa diakses melalui Reservation. Memiliki ID unik (assignment_id).
ReservationHistory (Entity)	class ReservationHistory	src/domain/models.py	Mencatat log perubahan status (Audit Trail) di dalam agregat reservasi.
ReservationStatus	class ReservationStatus(str, Enum)	src/domain/value_objects.py	Menggunakan Enum untuk membatasi status valid (PENDING, CONFIRMED, dll).
ContactInfo	class ContactInfo(BaseModel)	src/domain/value_objects.py	Mengelompokkan Nama, HP, dan Email. Bersifat Immutable (menggunakan Pydantic).
ReservationTime	class ReservationTime(BaseModel)	src/domain/value_objects.py	Mengenkapsulasi logika waktu mulai dan durasi.
ReservationPolicy	class	src/domain/value_objects.py	Menyimpan aturan seperti

	ReservationPolicy(BaseModel)		auto_cancel_minutes.
CancellationReason	class CancellationReason(BaseModel)	src/domain/value_objects.py	Objek nilai untuk menampung alasan saat pembatalan terjadi.
Domain Event	class ReservationCreated, class ReservationConfirmed, class ReservationCancelled	src/domain/events.py	Objek immutabel yang merepresentasikan kejadian bisnis penting (perubahan status).
confirmReservation()	def confirm_reservation(self)	class Reservation	Memvalidasi status saat ini sebelum mengubahnya menjadi CONFIRMED.
assignTable()	def assign_table(self, ...)	class Reservation	Membuat objek TableAssignment dan menghubungkannya ke reservasi.
cancelReservation()	def cancel_reservation(self, reason)	class Reservation	Menerima CancellationReason dan mengubah status menjadi CANCELLED.
recordHistory()	def _record_history(self, ...)	class Reservation	Method internal yang otomatis dipanggil saat status berubah untuk mencatat log.

2.3 Implementasi Endpoint

Implementasi API pada aplikasi ini dibangun berdasarkan model domain Reservation Management. Endpoint-endpoint tersebut diorganisasi ke dalam modul yang merepresentasikan area fungsional tertentu. Setiap endpoint disusun menggunakan FastAPI dan mengandalkan service layer untuk menjalankan logika aplikasi sesuai kaidah Domain-Driven Design (DDD). Bagian ini menyajikan daftar lengkap endpoint dengan metode, url, dan fungsinya.

Tabel Implementasi Endpoint

Metode HTTP	Endpoint URL	Fungsi / Deskripsi
POST	/api/reservations	Membuat Reservasi Baru. Menerima data pelanggan dan waktu, mengembalikan ID reservasi baru dengan status PENDING.
GET	/api/reservations/{reservation_id}	Melihat Detail Reservasi. Mengambil informasi lengkap reservasi berdasarkan ID (termasuk status dan meja jika ada).

POST	/api/reservations/{reservation_id}/confirm	Konfirmasi Reservasi. Mengubah status reservasi dari PENDING menjadi CONFIRMED.
POST	/api/reservations/{reservation_id}/assign-table	Alokasi Meja. Menetapkan nomor/area meja untuk reservasi tertentu.
POST	/api/reservations/{reservation_id}/cancel	Batalkan Reservasi. Membatalkan reservasi dengan menyertakan alasan pembatalan.
GET	/api/reservations	List & Filter Reservasi. Mengambil daftar reservasi dengan fitur paginasi dan penyaringan (filtering) berdasarkan status reservasi.
POST	/api/reservations/{id}/check-in	Check-In Tamu. Menandai pelanggan telah tiba di restoran. Mengubah status menjadi CHECKED_IN.
POST	/api/reservations/{id}/complete	Selesaikan Reservasi. Menandai pelanggan selesai makan dan meja kembali kosong. Mengubah status menjadi COMPLETED.
GET	/api/stats	Dashboard Statistik. Endpoint analitik yang mengembalikan ringkasan data operasional seperti total reservasi, jumlah per status, dan total estimasi pendapatan (revenue).

Catatan:

- **Format Data**: Semua endpoint menerima dan mengirim data dalam format JSON.
- **Parameter**: {reservation_id} pada URL adalah UUID unik yang didapatkan saat membuat reservasi pertama kali.

BAB III Desain dan Implementasi API (FastAPI)

3.1 Arsitektur Aplikasi

Arsitektur aplikasi yang digunakan pada implementasi ini dirancang berdasarkan prinsip Domain-Driven Design (DDD) dengan pendekatan Layered Architecture yang disederhanakan. Struktur ini bertujuan untuk memisahkan logika bisnis murni (Domain) dari detail teknis framework (FastAPI) dan validasi data eksternal. Hal ini memastikan bahwa aturan bisnis tetap menjadi pusat dari pengembangan perangkat lunak.

Aplikasi ini terbagi menjadi empat lapisan logis utama: Application Entry Point, Interface Layer, Domain Layer, dan Infrastructure Layer (Simulated). Berikut adalah rincian tanggung jawab setiap lapisan:

1. Application Entry Point (main.py)
   File ini berfungsi sebagai gerbang utama aplikasi. Tanggung jawab utamanya meliputi:
   - Inisialisasi Framework: Mengaktifkan instance aplikasi FastAPI.
   - Konfigurasi Routing: Mendaftarkan router dari lapisan API agar endpoint dapat diakses.
   - Server Bootstrapping: Menjalankan server aplikasi menggunakan Uvicorn.

2. Interface Layer (src/api/ & src/schemas/)
   Lapisan ini bertanggung jawab menangani interaksi dengan dunia luar (klien HTTP). Lapisan ini bertindak sebagai penerjemah antara permintaan eksternal dan logika internal aplikasi.
   - API Routes (src/api/routes.py):
     - Mendefinisikan endpoint HTTP (GET, POST) sesuai spesifikasi REST API.
     - Menerima input dari klien dan meneruskannya ke model domain.
     - Menangani respon HTTP dan kode status (misalnya: 200 OK, 404 Not Found, 422 Unprocessable Entity).
     - Pada implementasi dasar ini, routes juga bertindak sebagai pengendali alur aplikasi sederhana.
   - Schemas / DTO (src/schemas/reservation.py):
     - Berisi model Pydantic untuk Data Transfer Objects (DTO).
     - Melakukan validasi format data masuk (input sanitization) sebelum data menyentuh lapisan domain (misalnya: memastikan format UUID valid).
     - Mendefinisikan struktur data keluar (output serialization) untuk dikirimkan kembali ke klien.

3. Domain Layer (src/domain/)

   Layer ini adalah inti dari perangkat lunak yang merepresentasikan Core Domain yaitu Reservation Management. Lapisan ini murni berisi logika Python tanpa ketergantungan pada framework eksternal atau basis data .

   - Aggregate & Entities (src/domain/models.py):
     - Berisi implementasi Reservation sebagai Aggregate Root yang menjaga konsistensi data transaksi.
     - Menyimpan logika perilaku bisnis (behavior), seperti metode confirm_reservation(), assign_table(), dan cancel_reservation().
     - Mengelola entitas pendukung seperti TableAssignment dan ReservationHistory.
   - Value Objects (src/domain/value_objects.py):
     - Mendefinisikan objek nilai yang bersifat immutable (tidak berubah) untuk merepresentasikan konsep deskriptif seperti ReservationTime, ContactInfo, dan ReservationStatus.
     - Mengandung aturan validasi intrinsik (misalnya: durasi reservasi default).
   - Domain Events (src/domain/events.py):
     - Mendefinisikan objek kejadian bisnis (business events) yang bersifat immutable, seperti ReservationCreated dan ReservationConfirmed.
     - Merepresentasikan perubahan status penting (state change) yang terjadi di dalam Aggregate Root untuk keperluan audit trail atau pemrosesan lanjutan.

4. Infrastructure Layer (src/infrastructure/)

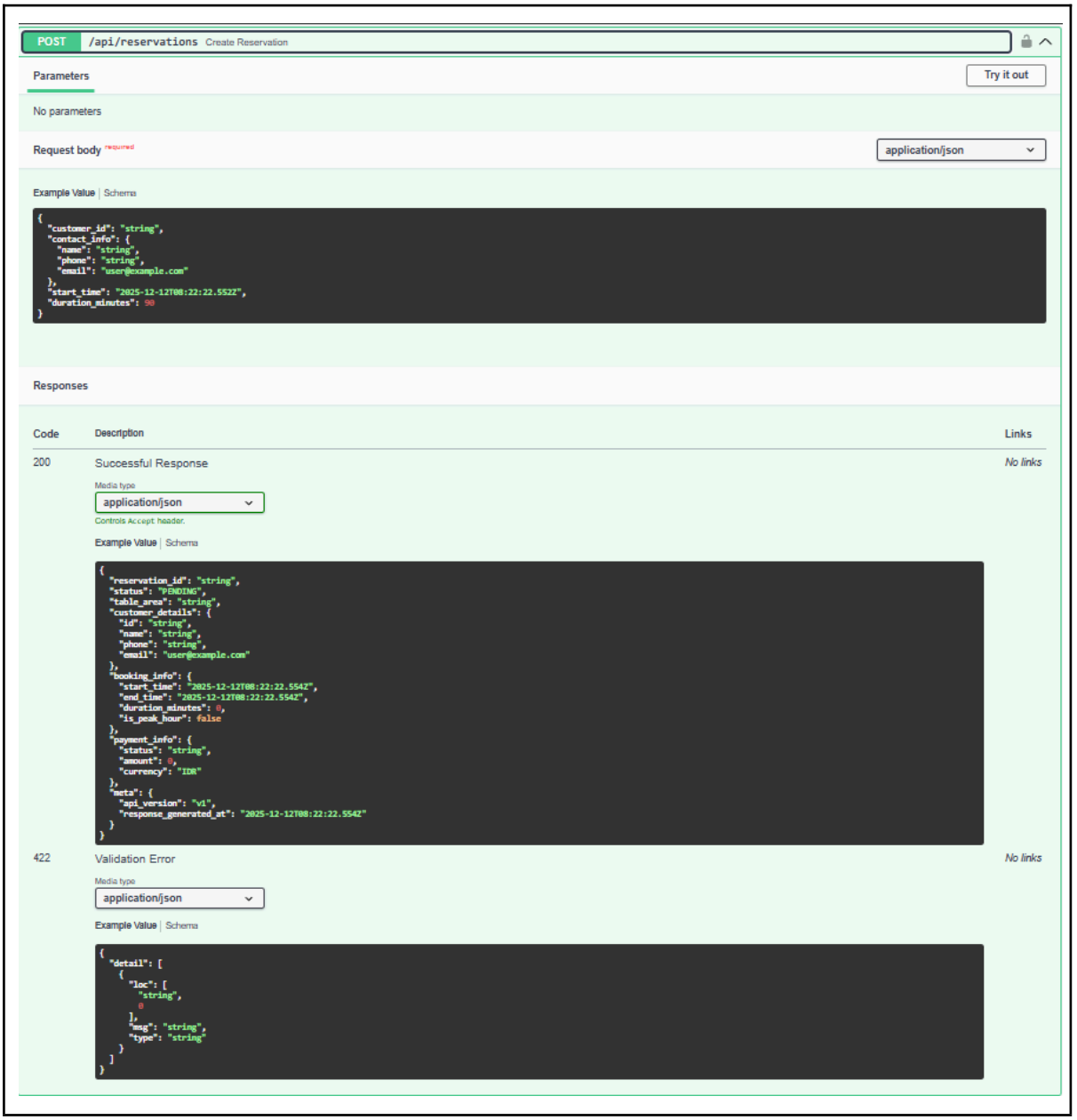
   Dalam implementasi prototipe API dasar ini, lapisan ini menyediakan mekanisme persistensi data yang nyata menggunakan teknologi basis data relasional.

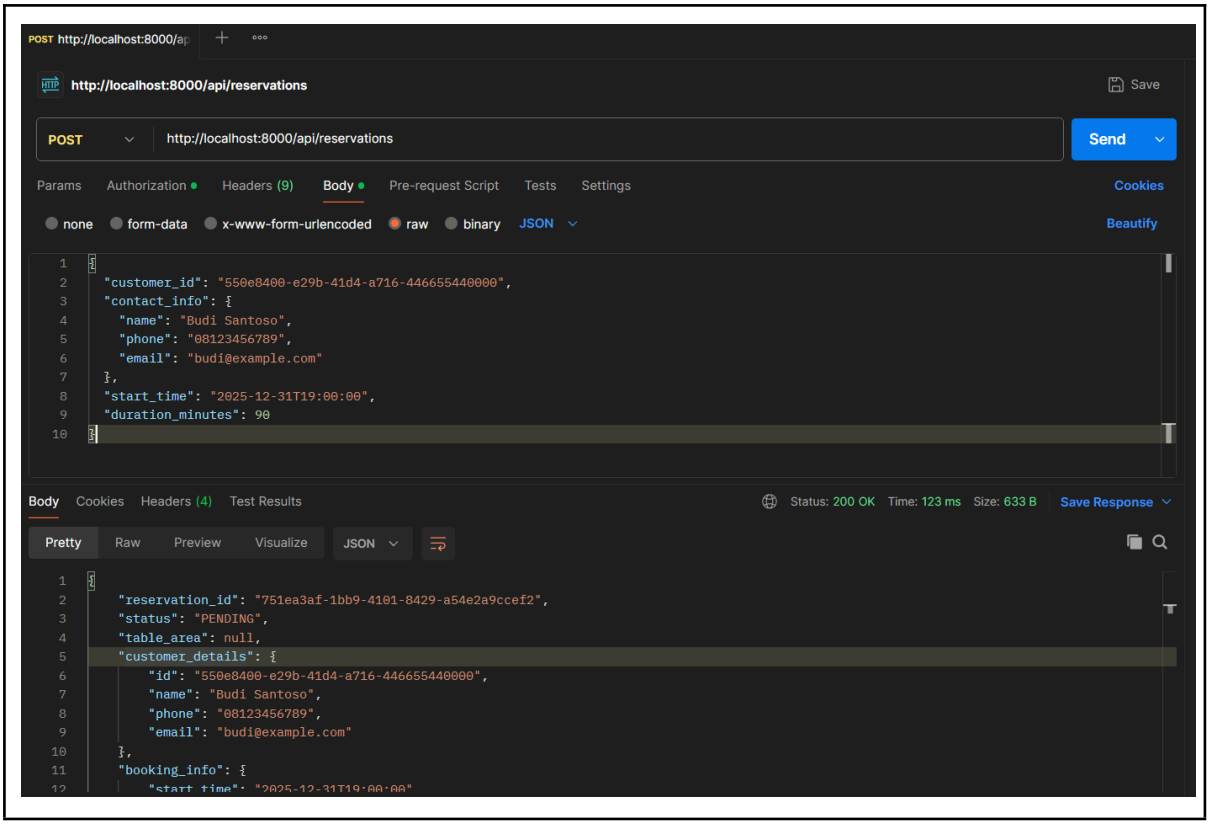
   - Database Configuration (database.py):
     - Mengelola koneksi ke server PostgreSQL menggunakan SQLAlchemy Engine dan SessionLocal.
   - ORM Models (orm_models.py):
     - Mendefinisikan struktur tabel database (ReservationORM) yang memetakan atribut Domain Entity dan Value Object menjadi kolom-kolom tabel fisik.

   Struktur ini memastikan bahwa Domain Layer tetap bersih dan tidak terpolusi oleh detail teknis seperti HTTP request atau query database, sehingga memudahkan pengujian unit dan pemeliharaan logika bisnis di masa depan.

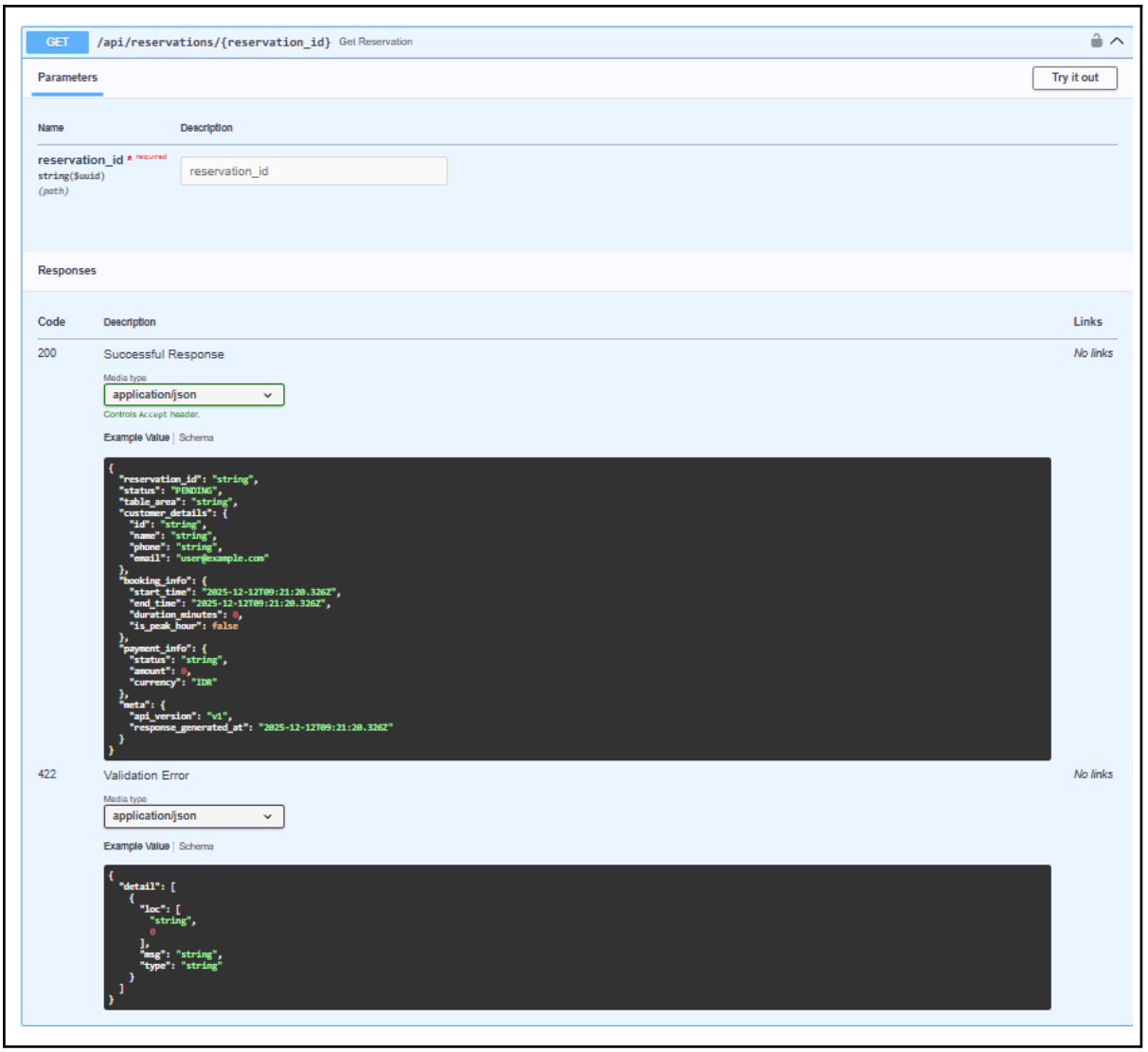
BAB IV Pengujian API

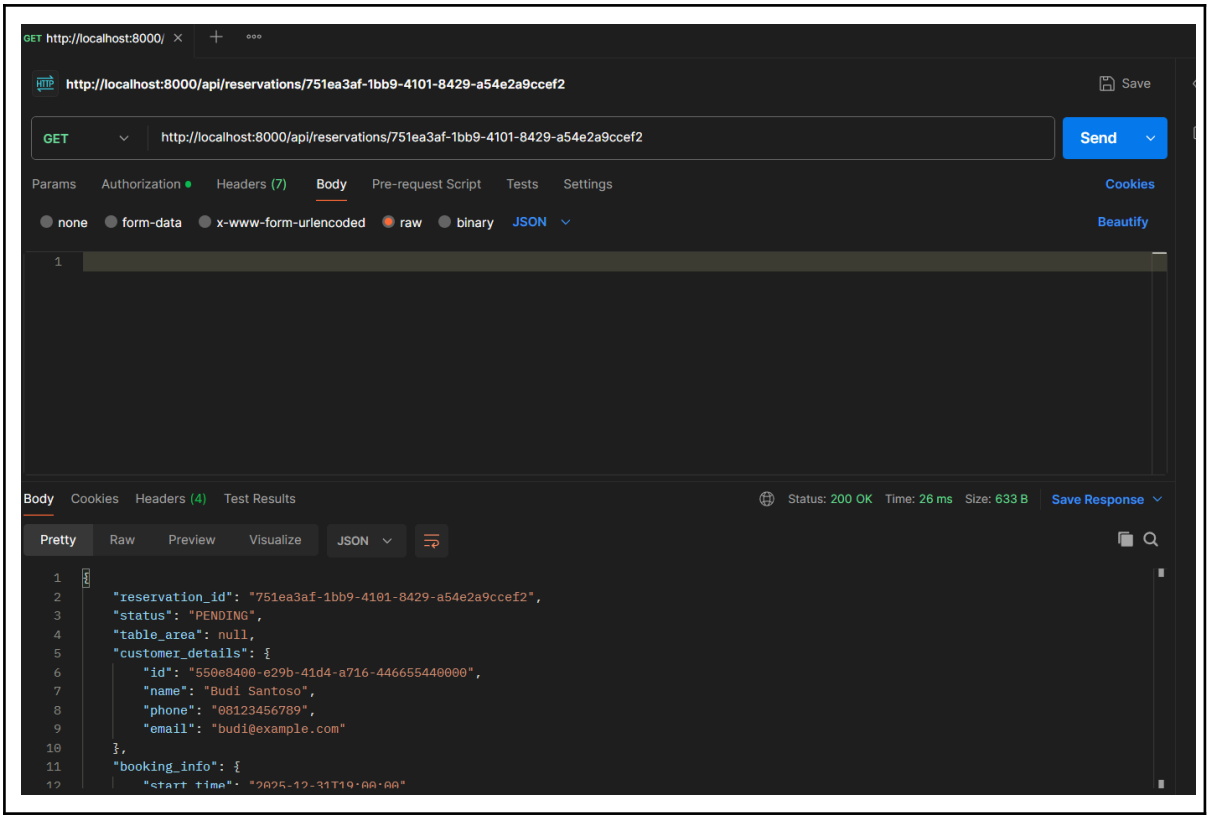
4.1 POST /api/reservations



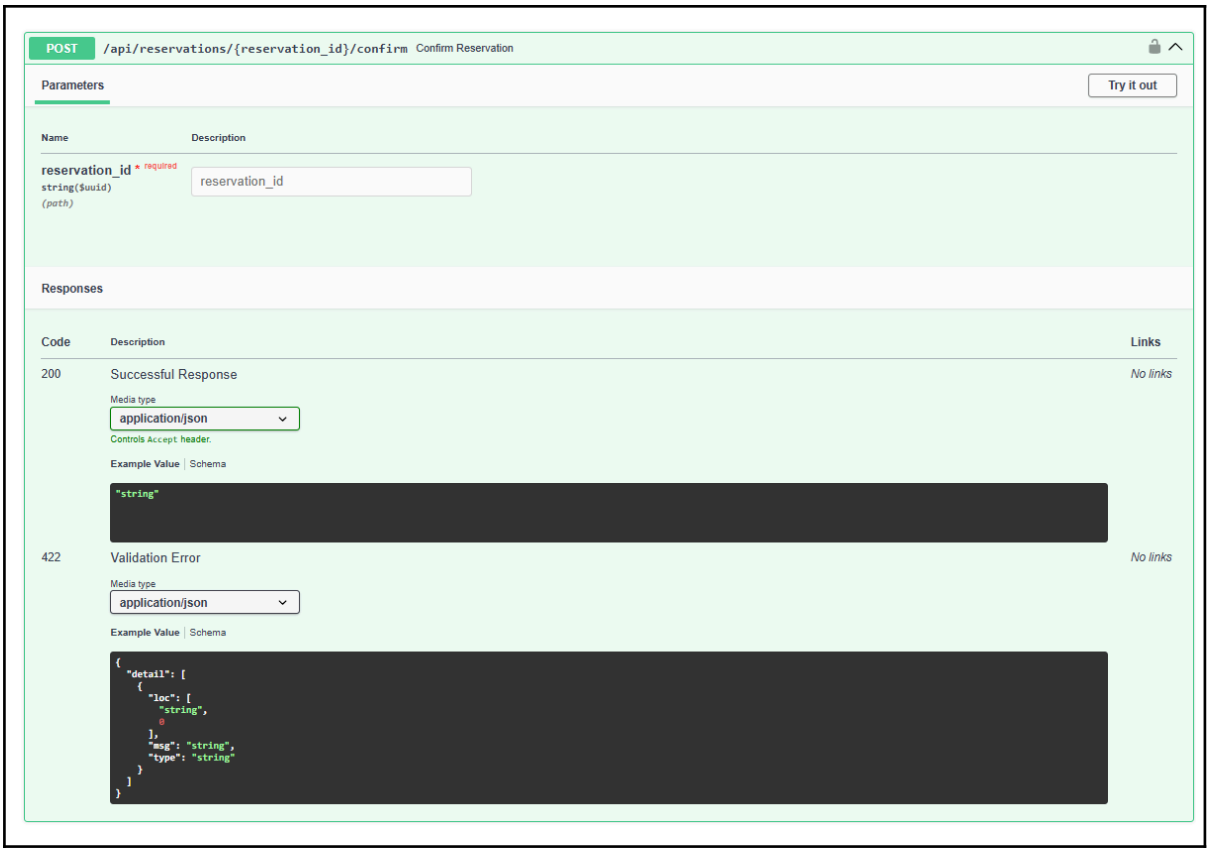

POST http://localhost:8000/ap
http://localhost:8000/api/reservations
Save
POST
http://localhost:8000/api/reservations
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
JSON
Beautify
{
"customer_id": "550e8400-e29b-41d4-a716-446655440000",
"contact_info": {
"name": "Budi Santoso",
"phone": "08123456789",
"email": "budi@example.com"
},
"start_time": "2025-12-31T19:00:00",
"duration_minutes": 90
}
Body
Cookies
Headers (4)
Test Results
Status: 200 OK
Time: 123 ms
Size: 633 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"reservation_id": "751ea3af-1bb9-4101-8429-a54e2a9ccef2",
"status": "PENDING",
"table_area": null,
"customer_details": {
"id": "550e8400-e29b-41d4-a716-446655440000",
"name": "Budi Santoso",
"phone": "08123456789",
"email": "budi@example.com"
},
"booking_info": {

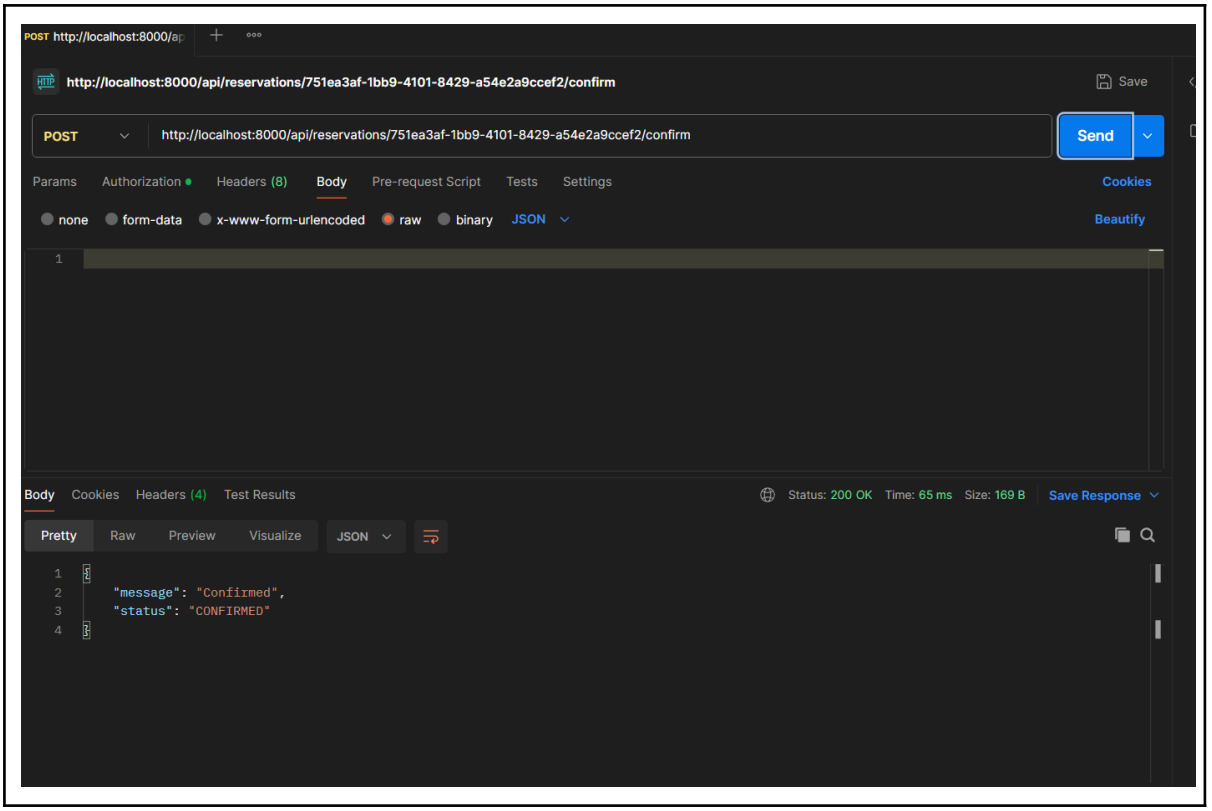
4.2 GET /api/reservations/{reservation_id}



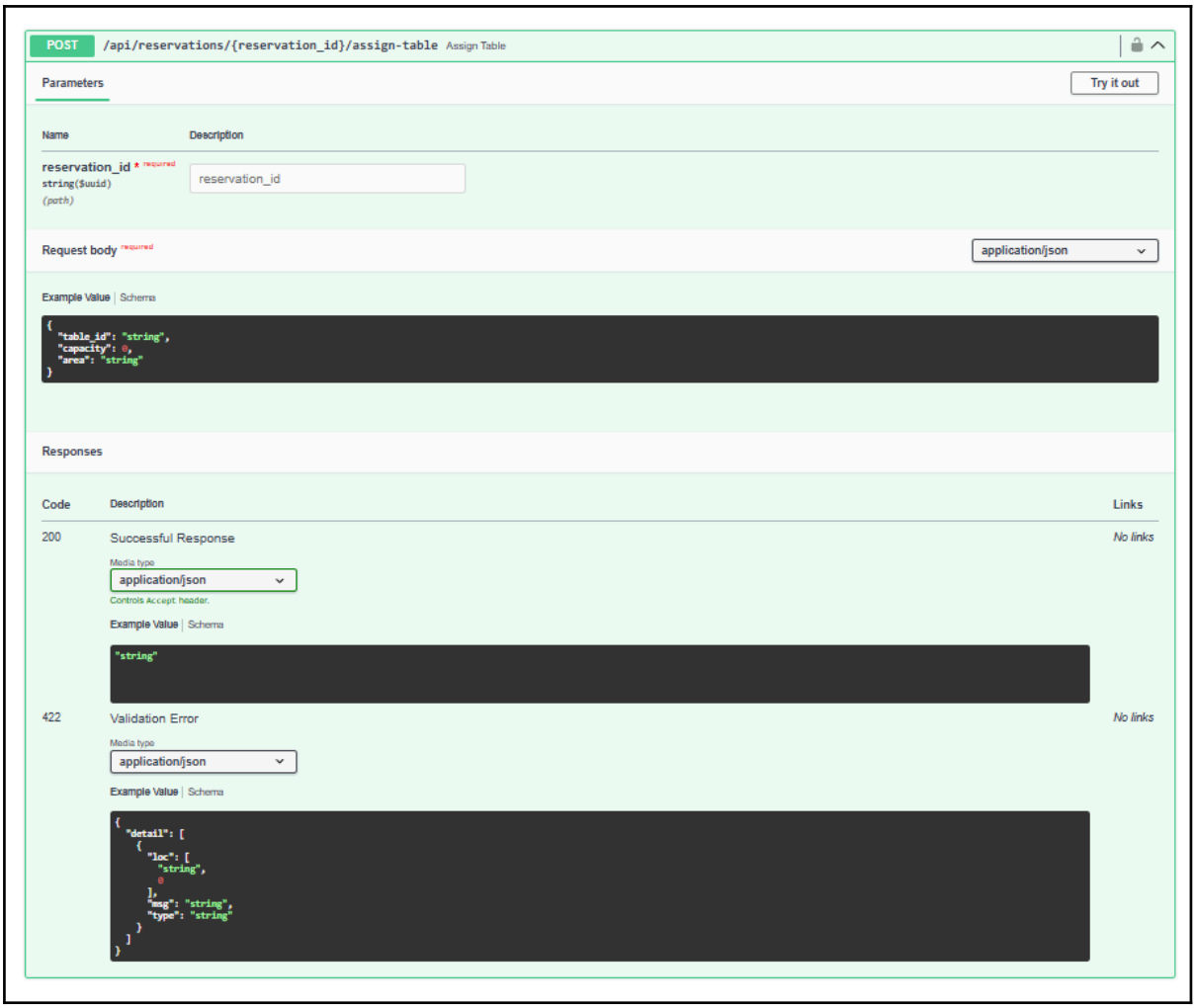


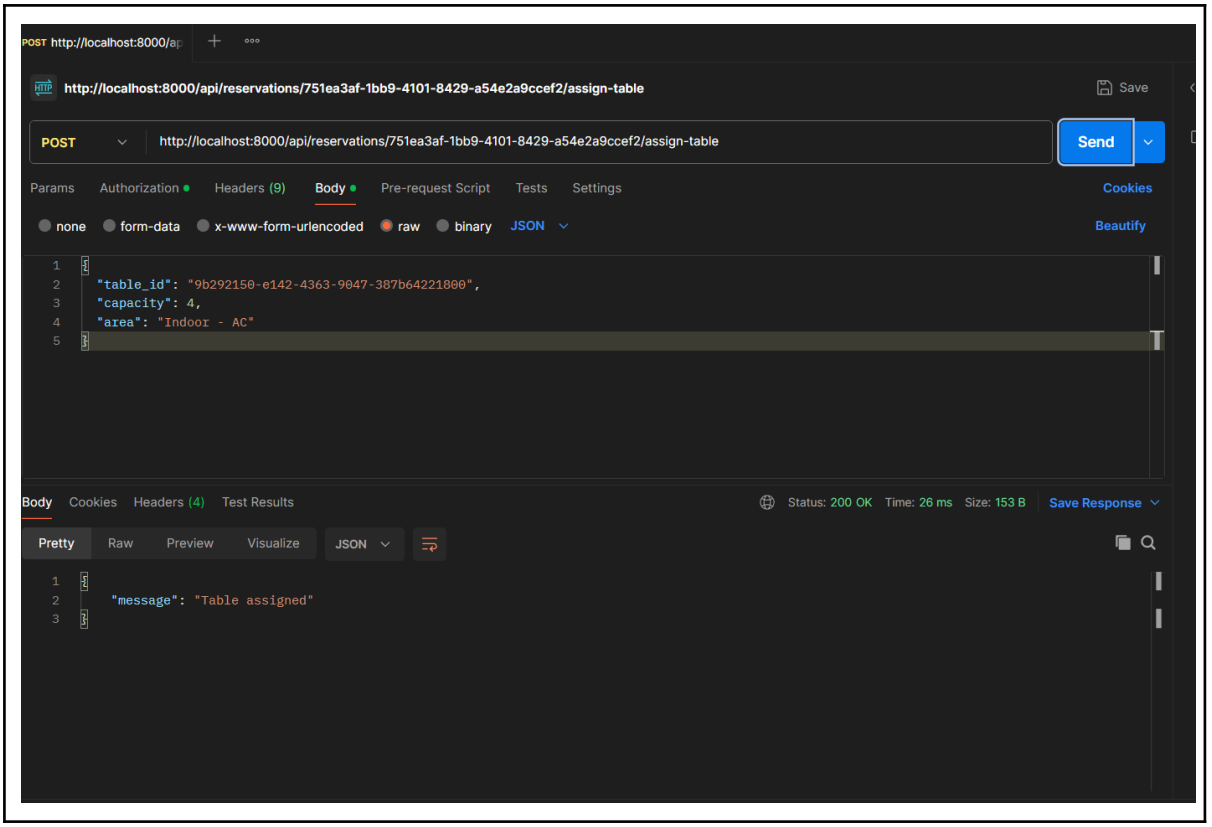
4.3 POST /api/reservations/{reservation_id}/confirm



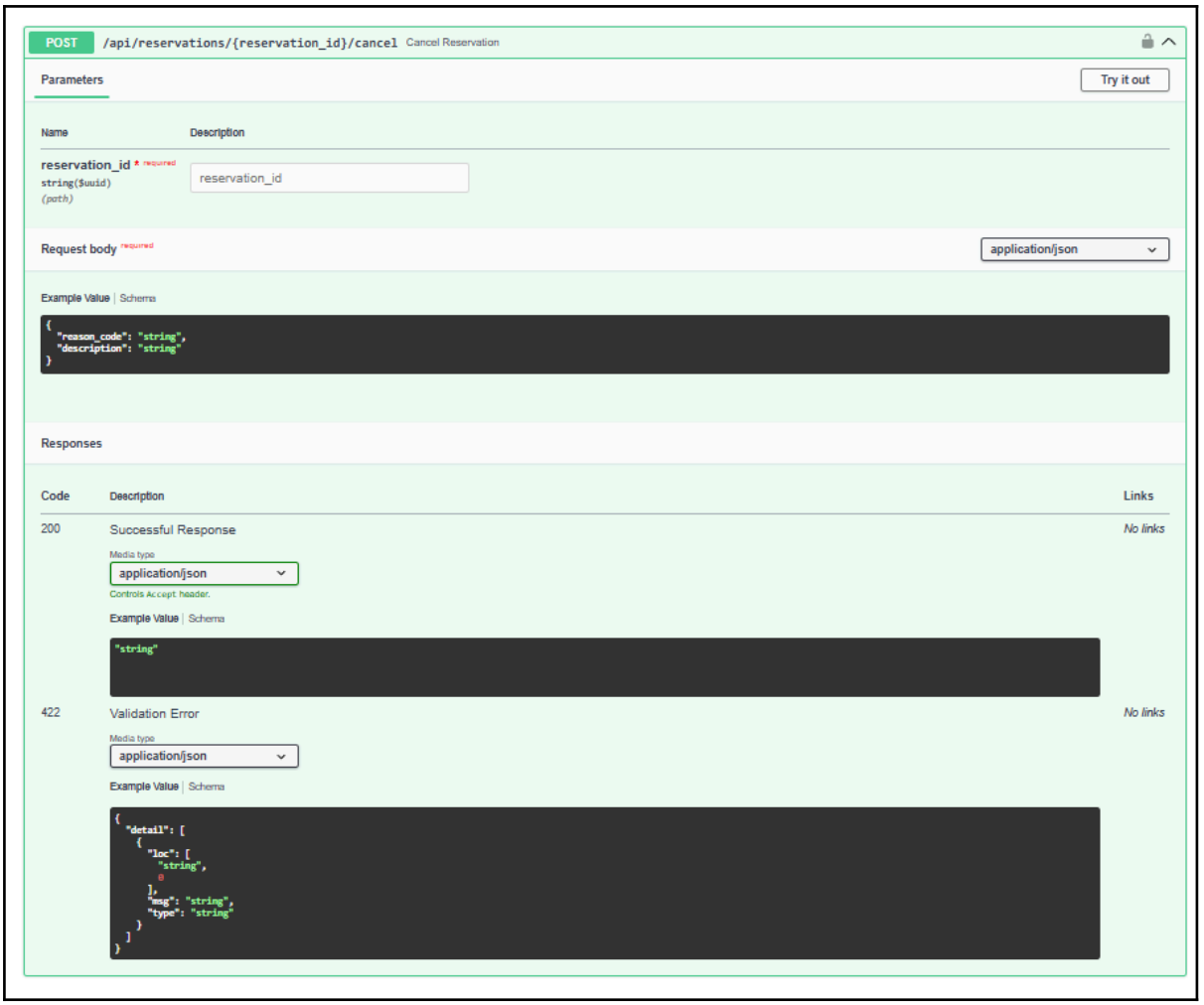

POST http://localhost:8000/ap
http://localhost:8000/api/reservations/751ea3af-1bb9-4101-8429-a54e2a9ccef2/confirm
Save
POST
http://localhost:8000/api/reservations/751ea3af-1bb9-4101-8429-a54e2a9ccef2/confirm
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
JSON
Beautify
1
Body
Cookies
Headers (4)
Test Results
Status: 200 OK
Time: 65 ms
Size: 169 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
1
2 "message": "Confirmed",
3 "status": "CONFIRMED"
4

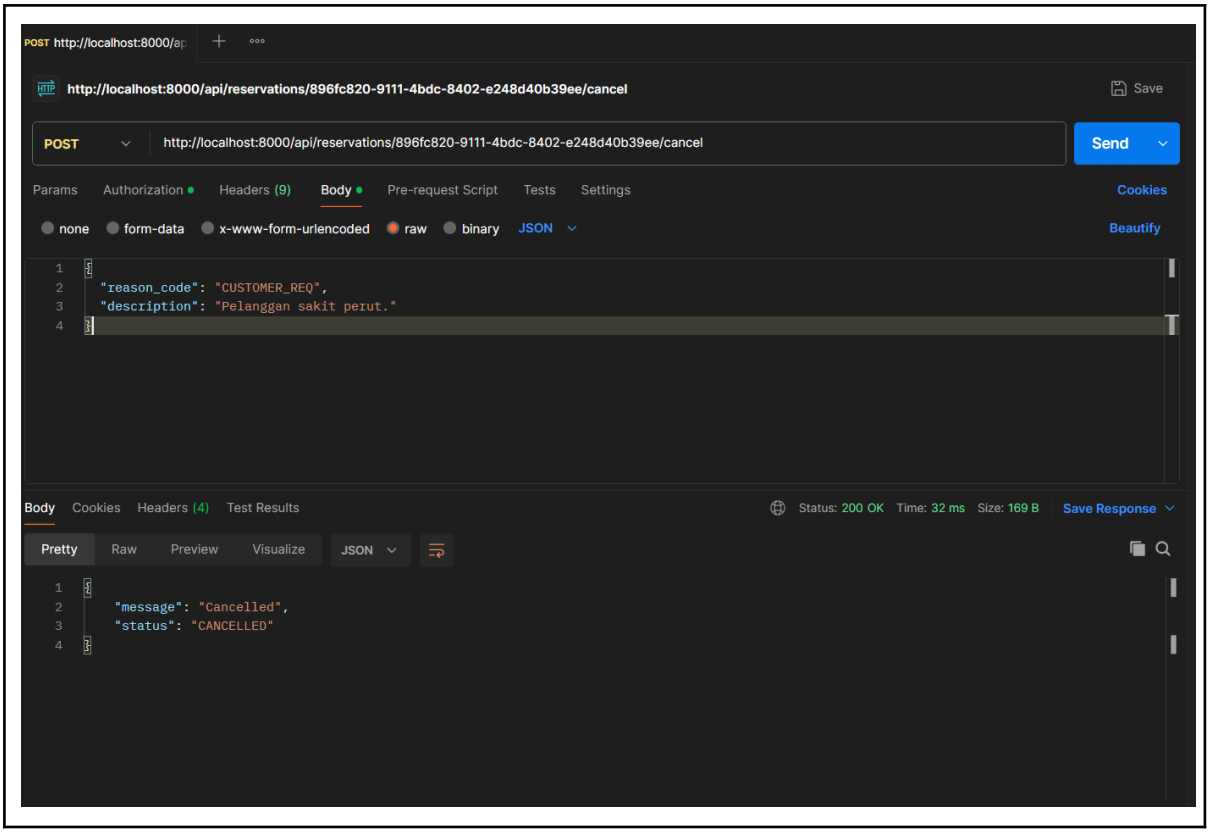
4.4 POST /api/reservations/{reservation_id}/assign-table



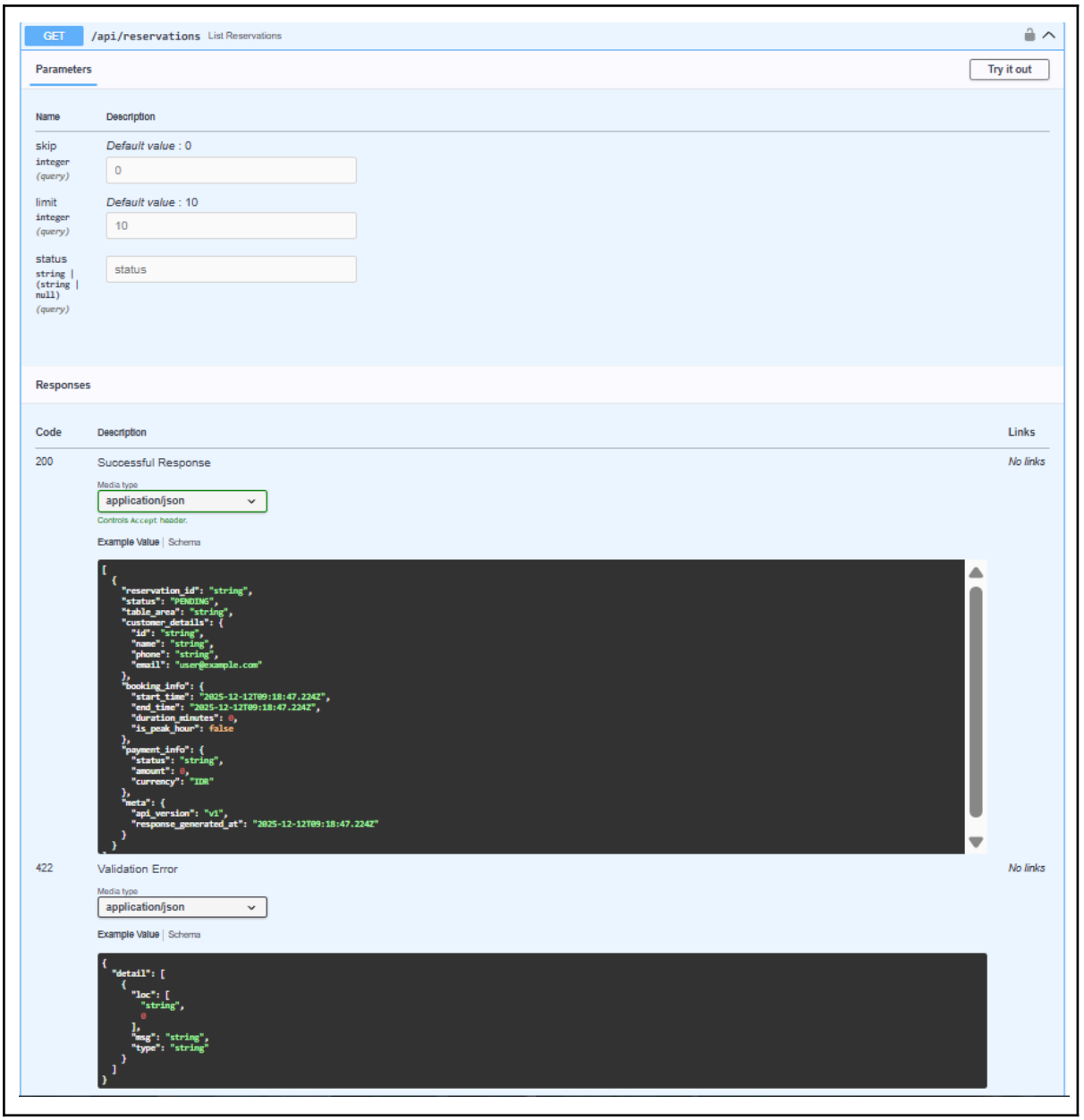

POST http://localhost:8000/ap
http://localhost:8000/api/reservations/751ea3af-1bb9-4101-8429-a54e2a9ccef2/assign-table
Save
POST
http://localhost:8000/api/reservations/751ea3af-1bb9-4101-8429-a54e2a9ccef2/assign-table
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
JSON
Beautify
1
2 "table_id": "9b292150-e142-4363-9047-387b64221800",
3 "capacity": 4,
4 "area": "Indoor - AC"
5
Body
Cookies
Headers (4)
Test Results
Status: 200 OK
Time: 26 ms
Size: 153 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
1
2 "message": "Table assigned"
3

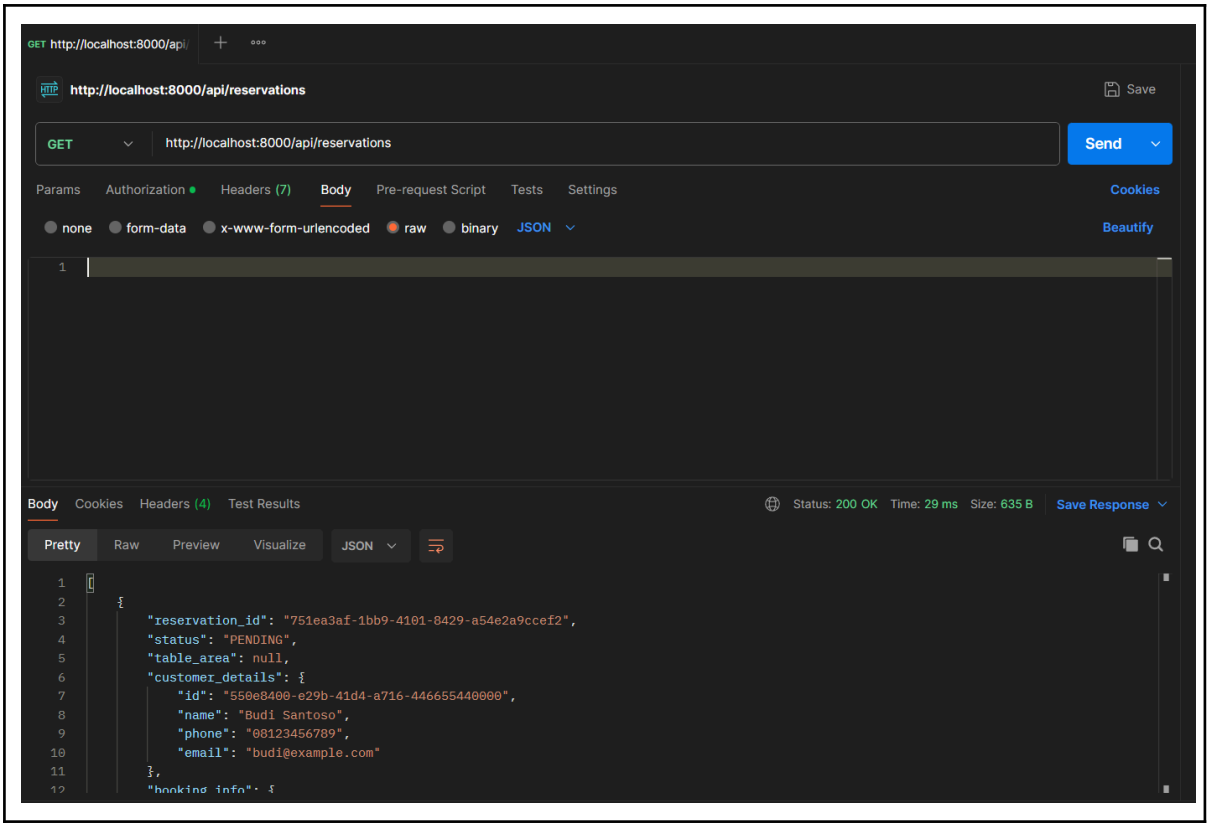
4.5 POST /api/reservations/{reservation_id}/cancel



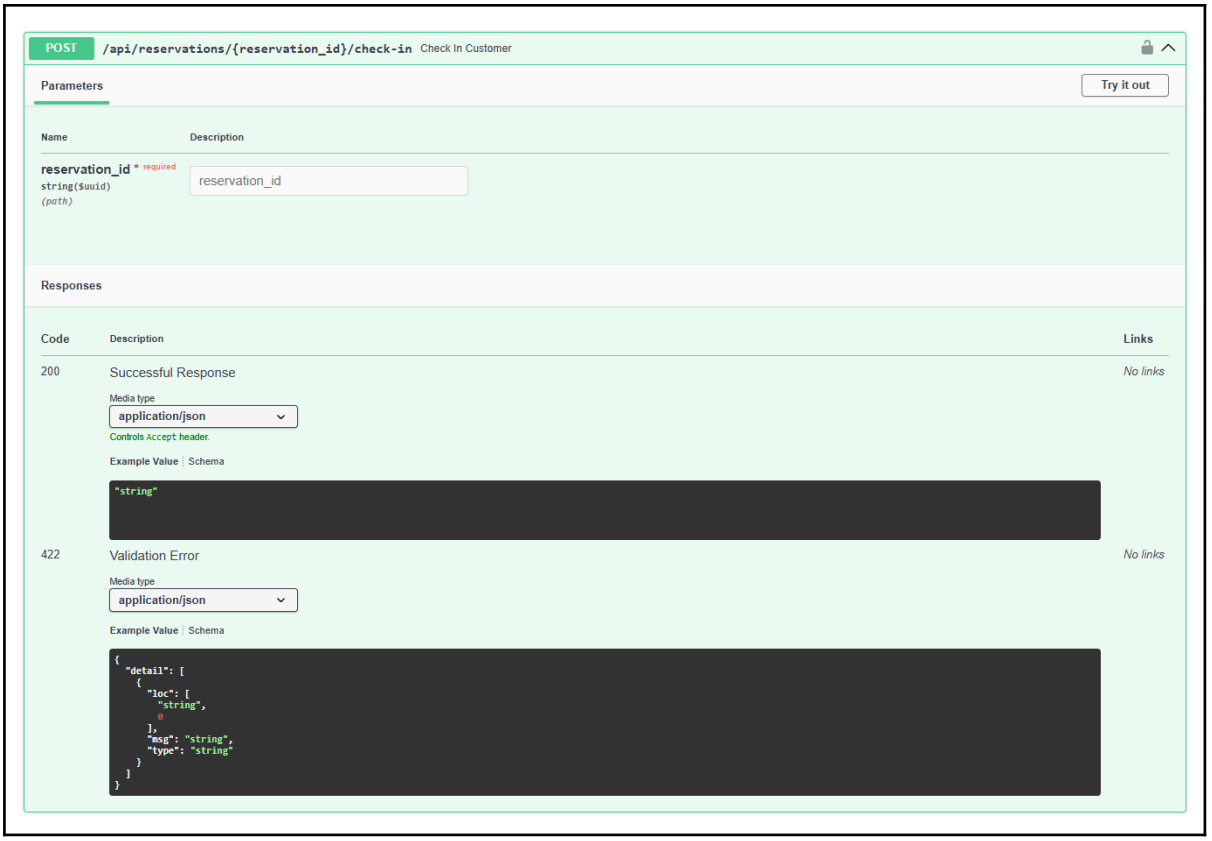

POST http://localhost:8000/ap
http://localhost:8000/api/reservations/896fc820-9111-4bdc-8402-e248d40b39ee/cancel
Save
POST
http://localhost:8000/api/reservations/896fc820-9111-4bdc-8402-e248d40b39ee/cancel
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
JSON
Beautify
"reason_code": "CUSTOMER_REQ",
"description": "Pelanggan sakit perut."
Body
Cookies
Headers (4)
Test Results
Status: 200 OK
Time: 32 ms
Size: 169 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
"message": "Cancelled",
"status": "CANCELLED"

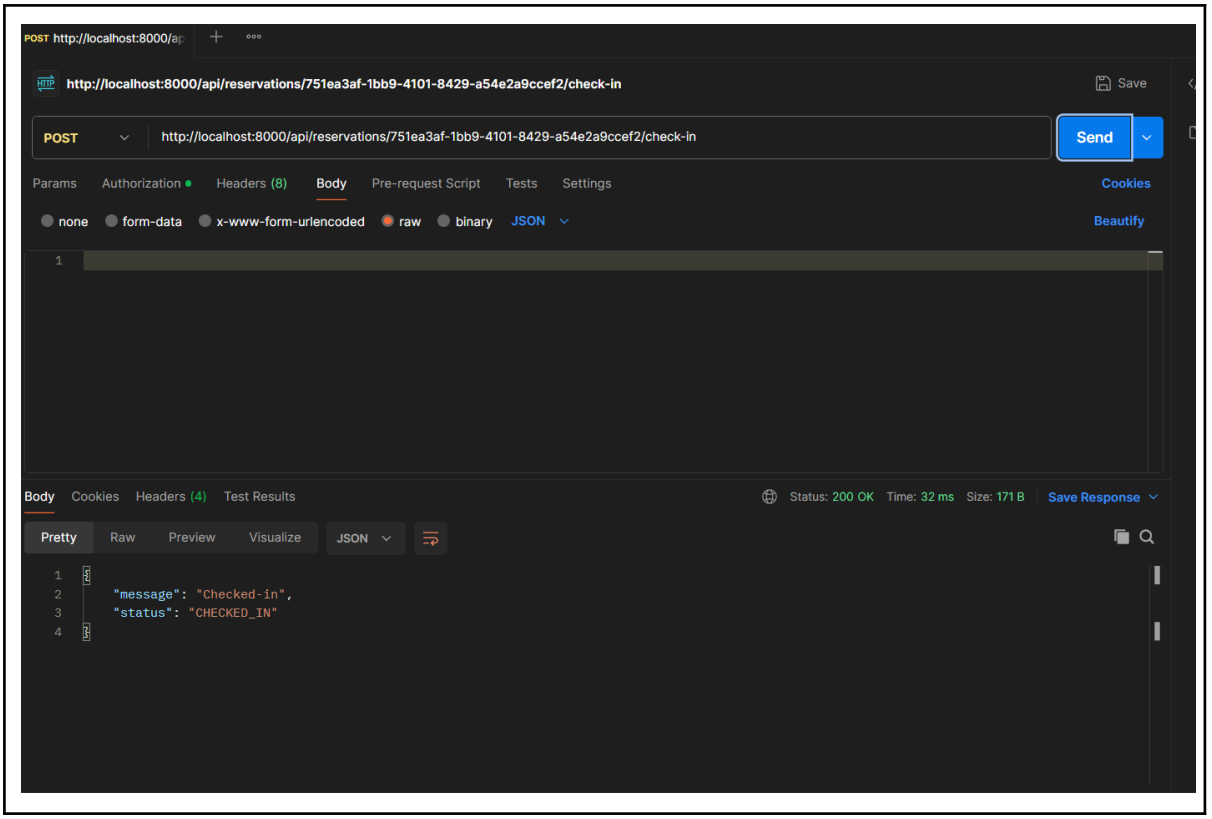
4.6 GET /api/reservations



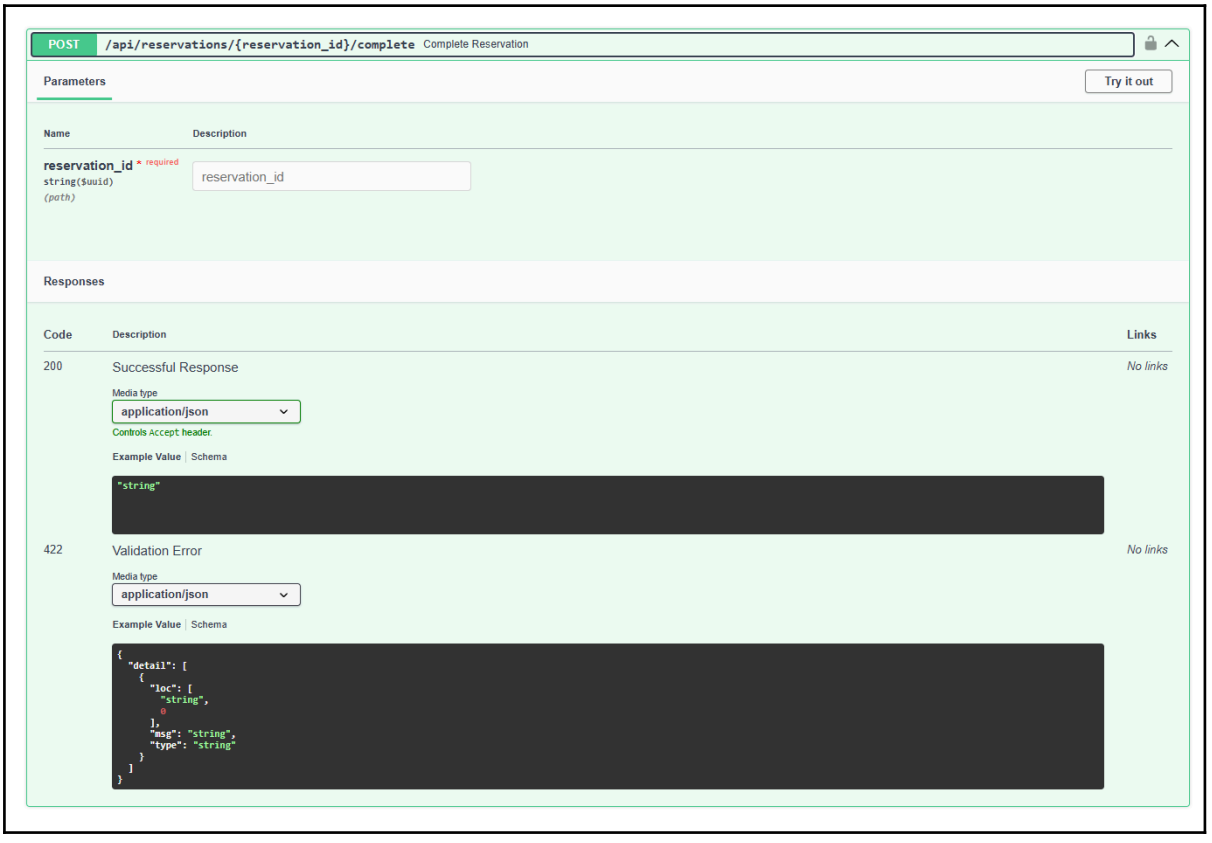


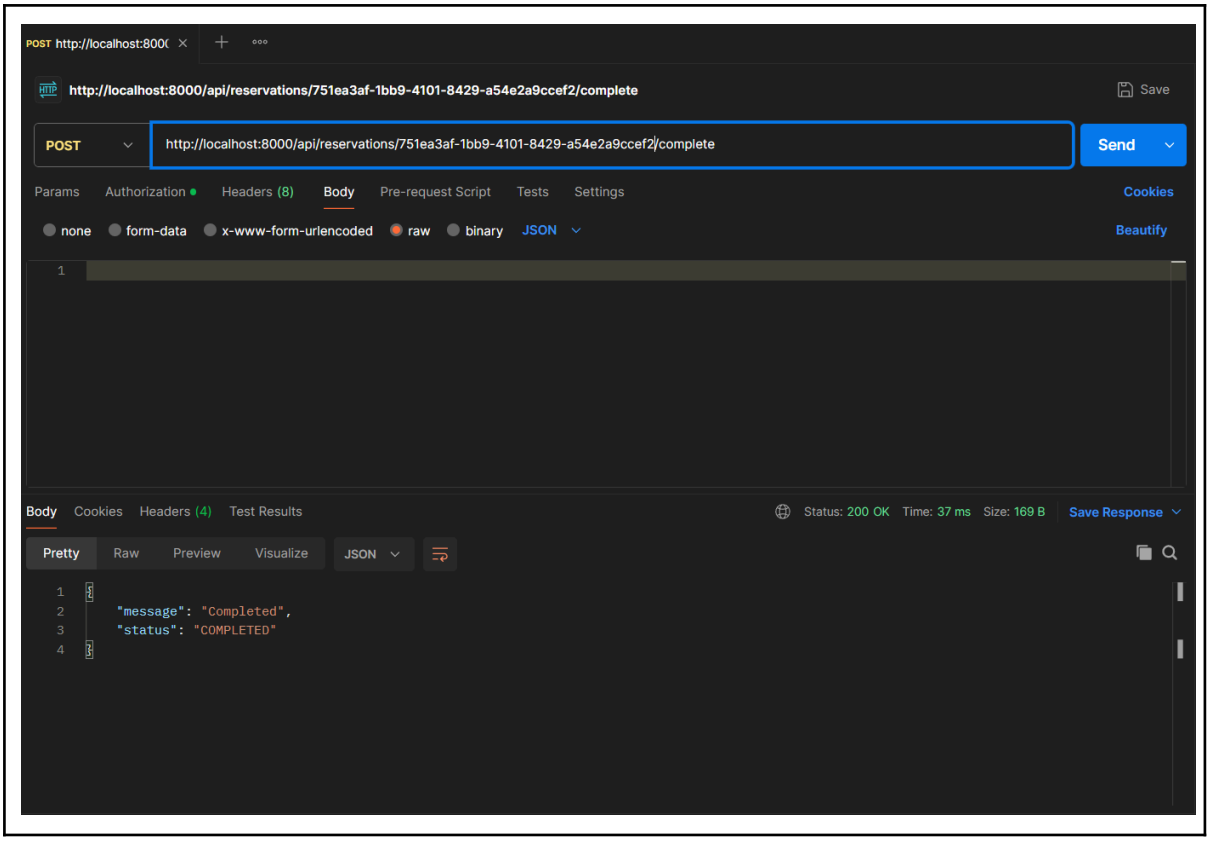
4.7 POST /api/reservations/{id}/check-in



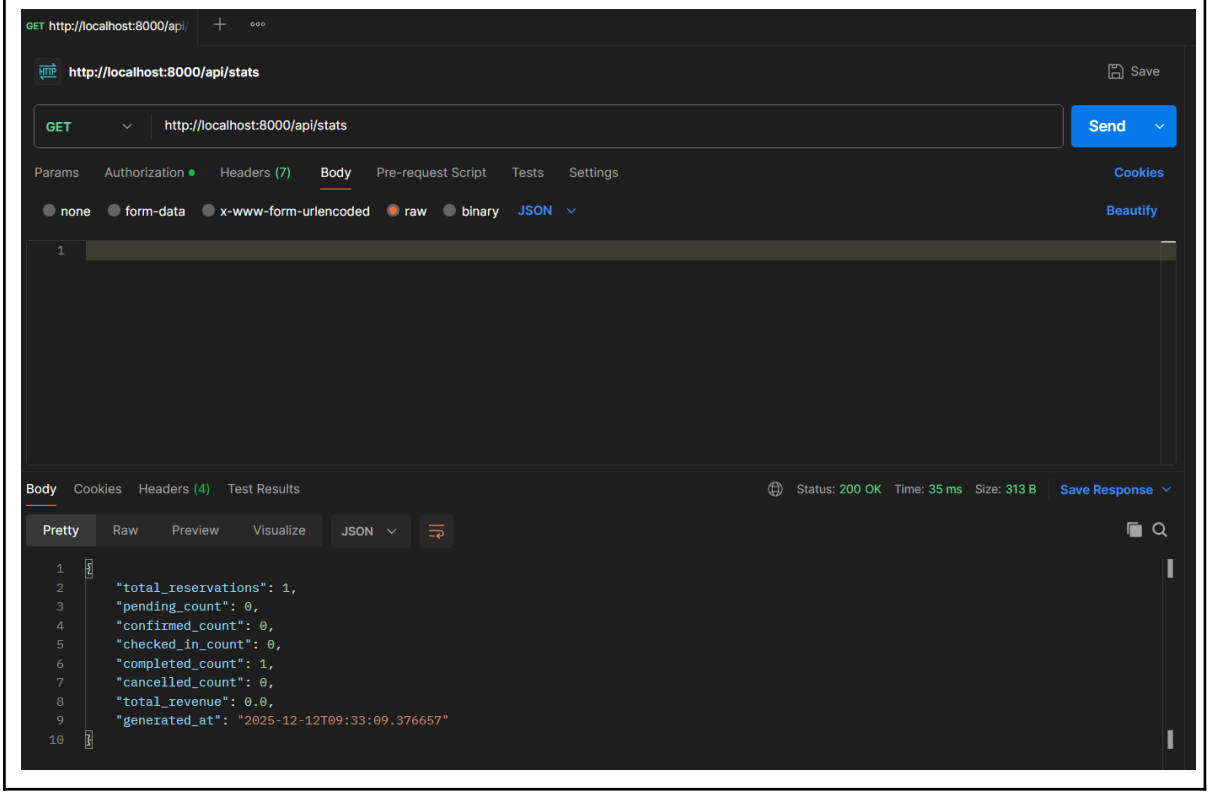


4.8 POST /api/reservations/{id}/complete





4.9 GET /api/stats



BAB V Kesimpulan

Dokumen ini telah menjelaskan proses perancangan dan implementasi sistem berbasis Domain-Driven Design (DDD) untuk membangun layanan inti Reservation Management pada Sistem Reservasi & Manajemen Restoran Terintegrasi. Pendekatan ini memastikan bahwa logika bisnis kompleks pada domain reservasi dapat dimodelkan dengan baik melalui struktur domain yang jelas, konsisten, dan mudah dikembangkan.

Pada tahap awal, aggregate model diterjemahkan menjadi komponen teknis menggunakan pendekatan layered architecture, yang terdiri dari domain layer, application service layer, infrastructure layer, dan API layer. Proses ini memastikan bahwa setiap lapisan memiliki tanggung jawab yang terpisah secara tegas, sehingga model domain tetap bersih dan tidak bercampur dengan detail teknis seperti mekanisme storage.

Implementasi API menggunakan FastAPI menunjukkan bagaimana konsep-konsep Domain-Driven Design, seperti Entity, Value Object, dan Domain Event, dapat terhubung langsung ke endpoint layanan yang siap diakses oleh aplikasi front-end maupun service lain.

Secara keseluruhan, penerapan DDD terbukti memberikan keuntungan dalam hal kejelasan model, kemudahan pemeliharaan, dan fleksibilitas untuk mengembangkan fitur baru di masa mendatang. Meskipun implementasi masih berada pada tahap awal dan bersifat sederhana, fondasi yang telah dibangun memberikan dasar yang kuat untuk diperluas pada fase pengembangan selanjutnya, termasuk integrasi persistence layer yang lebih stabil, penanganan domain event secara asynchronous, dan penerapan arsitektur cloud-native.

Dengan tercapainya implementasi awal aggregate dan API dasar, proyek ini telah memenuhi tujuan utama penugasan, yaitu menerjemahkan model domain ke dalam bentuk kode operasional yang dapat dijalankan dan diuji melalui antarmuka REST API. Tahapan selanjutnya akan difokuskan pada peningkatan kapabilitas teknis dan perluasan fitur untuk mendukung keseluruhan ekosistem restoran yang lebih kompleks.

Daftar Pustaka

Evans, E. (2003). Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley.

Vernon, V. (2013). Implementing Domain-Driven Design. Addison-Wesley Professional.

Richardson, C. (2018). Microservices Patterns: With Examples in Java. Manning Publications.

FastAPI Documentation. (2024). FastAPI: Modern, Fast (High-Performance) Web Framework for Building APIs with Python. Retrieved from https://fastapi.tiangolo.com

Python Software Foundation. (2024). Python 3.11 Documentation. Retrieved from https://docs.python.org/3/

Stripe API Documentation. (2024). Stripe Payments API Reference. Retrieved from https://stripe.com/docs/api

Twilio API Documentation. (2024). Twilio Messaging API Reference. Retrieved from https://www.twilio.com/docs/messaging/api

Clearbit API Documentation. (2024). Clearbit Enrichment API. Retrieved from https://dashboard.clearbit.com/docs

Fowler, M. (2004). Patterns of Enterprise Application Architecture. Addison-Wesley.

Microsoft Azure Architecture Center. (2024). Domain-Driven Design Fundamentals. Retrieved from https://learn.microsoft.com/en-us/azure/architecture/microservices/model/domain-analysis

OpenAPI Initiative. (2024). OpenAPI Specification v3.1. Retrieved from https://spec.openapis.org/oas/latest.html

Lampiran

Link Repository: https://github.com/rasyidrizky/Restaurant-Reservation