Sebelum masuk ke integrasi Midtrans dan payment, bagaimana dengan ticket?

Tentu, sebelum melanjutkan ke integrasi **Midtrans** dan **pembayaran**, fokus pada **Ticketing** adalah langkah yang tepat karena tiket merupakan bagian inti dari proses transaksi yang akan dilanjutkan ke pembayaran. Berikut adalah beberapa poin penting yang perlu dipertimbangkan saat membangun **CRUD Ticket**:

**Alur CRUD untuk Tabel ticket**

Tabel **ticket** menyimpan data terkait **tiket** yang dijual untuk spot wisata tertentu. Tabel ini biasanya berisi informasi seperti nama tiket, harga, stok, dan kode unik untuk setiap tiket yang dijual.

**1. Tabel ticket:**

* **ticket\_id**: ID unik untuk tiket.
* **ticket\_name**: Nama tiket, misalnya Tiket Masuk, VIP Ticket, dll.
* **ticket\_price**: Harga tiket.
* **ticket\_stock**: Stok tiket yang tersedia.
* **code**: Kode unik yang digunakan untuk tiket (misalnya, TICKET-001, VIP-003).
* **url\_qr**: URL untuk QR Code tiket yang digunakan saat pengecekan tiket.

**Alur API CRUD untuk ticket:**

**1. GET: Mengambil daftar tiket yang tersedia.**

* **Tujuan**: Mengambil semua tiket yang tersedia untuk spot wisata tertentu, termasuk stok dan harga tiket.
* **Endpoint**: /api/admin/ticket/route
* **Query**: Dapat mengakses tiket berdasarkan **spot\_id**.

**2. POST: Menambahkan tiket baru.**

* **Tujuan**: Admin menambahkan tiket baru untuk spot wisata.
* **Endpoint**: /api/admin/ticket/create

**3. PUT: Memperbarui tiket yang sudah ada.**

* **Tujuan**: Admin mengupdate tiket yang sudah ada (misalnya, mengubah harga atau stok).
* **Endpoint**: /api/admin/ticket/update/[id]

**4. DELETE: Menghapus tiket yang sudah ada.**

* **Tujuan**: Admin menghapus tiket yang tidak lagi dijual atau tidak diperlukan.
* **Endpoint**: /api/admin/ticket/delete/[id]

**Contoh Implementasi API CRUD untuk ticket:**

**1. GET untuk mengambil daftar tiket:**

|  |
| --- |
| // pages/api/admin/ticket/route.ts  import { NextApiRequest, NextApiResponse } from 'next'  import prisma from '@lib/prisma'  export default async function handler(req: NextApiRequest, res: NextApiResponse) {  if (req.method === 'GET') {  const { spot\_id } = req.query  try {  const tickets = await prisma.ticket.findMany({  where: {  spot\_id: spot\_id as string, // Mengambil tiket berdasarkan spot\_id  },  })  res.status(200).json(tickets)  } catch (error) {  res.status(500).json({ error: 'Error fetching tickets' })  }  } else {  res.status(405).json({ error: 'Method Not Allowed' })  }  } |

**2. POST untuk menambahkan tiket baru**:

|  |
| --- |
| // pages/api/admin/ticket/create.ts  import { NextApiRequest, NextApiResponse } from 'next'  import prisma from '@lib/prisma'  export default async function handler(req: NextApiRequest, res: NextApiResponse) {  if (req.method === 'POST') {  const { spot\_id, ticket\_name, ticket\_price, ticket\_stock, code } = req.body  try {  const newTicket = await prisma.ticket.create({  data: {  spot\_id,  ticket\_name,  ticket\_price,  ticket\_stock,  code,  },  })  res.status(200).json(newTicket)  } catch (error) {  res.status(500).json({ error: 'Failed to create ticket' })  }  } else {  res.status(405).json({ error: 'Method Not Allowed' })  }  } |

**3. PUT untuk memperbarui tiket yang sudah ada**:

|  |
| --- |
| // pages/api/admin/ticket/update/[id].ts  import { NextApiRequest, NextApiResponse } from 'next'  import prisma from '@lib/prisma'  export default async function handler(req: NextApiRequest, res: NextApiResponse) {  const { id } = req.query  if (req.method === 'PUT') {  const { ticket\_name, ticket\_price, ticket\_stock } = req.body  try {  const updatedTicket = await prisma.ticket.update({  where: { ticket\_id: id as string },  data: {  ticket\_name,  ticket\_price,  ticket\_stock,  },  })  res.status(200).json(updatedTicket)  } catch (error) {  res.status(500).json({ error: 'Failed to update ticket' })  }  } else {  res.status(405).json({ error: 'Method Not Allowed' })  }  } |

**4. DELETE untuk menghapus tiket**:

|  |
| --- |
| // pages/api/admin/ticket/delete/[id].ts  import { NextApiRequest, NextApiResponse } from 'next'  import prisma from '@lib/prisma'  export default async function handler(req: NextApiRequest, res: NextApiResponse) {  const { id } = req.query  if (req.method === 'DELETE') {  try {  await prisma.ticket.delete({  where: { ticket\_id: id as string },  })  res.status(200).json({ message: 'Ticket deleted successfully' })  } catch (error) {  res.status(500).json({ error: 'Failed to delete ticket' })  }  } else {  res.status(405).json({ error: 'Method Not Allowed' })  }  } |

**Bagaimana Mengelola Tiket dan Pembelian?**

* Setelah tiket ditambahkan, pengguna dapat memilih tiket untuk **spot wisata** tertentu, dan detail tiket akan disertakan dalam **transaksi**.
* **Tabel transaction\_detail** akan berisi detail tiket yang dibeli, termasuk jumlah dan harga.
* **Tabel ticket** akan menyimpan data stok tiket yang tersedia, yang akan berkurang saat pengguna melakukan pembelian.

**Hubungan antara ticket dan transaction:**

* **Satu transaksi** dapat berisi **beberapa tiket** yang dibeli. Oleh karena itu, kita menghubungkan **tabel ticket** dan **transaction\_detail** untuk melacak tiket yang dibeli dalam suatu transaksi.

**Perhitungan Stok Tiket:**

Saat tiket dibeli, **stok tiket** di **tabel ticket** harus diperbarui. Setiap kali transaksi selesai, stok tiket akan berkurang sesuai dengan jumlah yang dibeli.

**Contoh Pembaruan Stok Tiket:**

|  |
| --- |
| // Setelah transaksi berhasil  await prisma.ticket.update({  where: { ticket\_id: ticket\_id },  data: { ticket\_stock: { decrement: quantity } }, // Mengurangi stok tiket  }) |

**Kesimpulan:**

* **GET** untuk mengambil tiket yang tersedia.
* **POST** untuk menambah tiket baru.
* **PUT** untuk memperbarui informasi tiket (misalnya stok atau harga).
* **DELETE** untuk menghapus tiket.

Setelah tiket tersedia dan terhubung dengan transaksi, kamu bisa lanjutkan ke integrasi **Midtrans** untuk mengelola pembayaran.

Apakah kamu ingin melanjutkan dengan **integrasi Midtrans** atau ada hal lain yang ingin dibahas seputar **CRUD Ticket**?