Vater E. T. Prawira
C11210039

Github: https://github.com/vaterprawira/Project-PAT

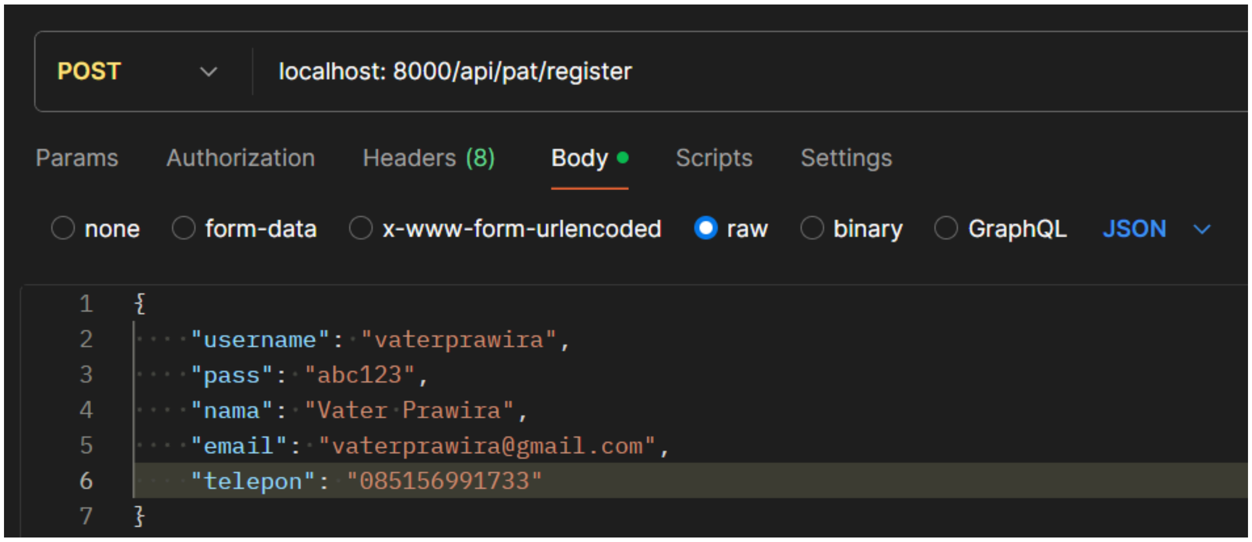
API

Services	Method	API	Parameters	Deskripsi
Register	POST	/api/pat/register	Username, password, nama, email, no hp	Mobile client dan desktop client login
Login	POST	/api/pat/login	Username, password	Mobile client dan desktop client membuat akun
Retrieve user by id	GET	/api/pat/register/:id	Username, nama, email, no hp	Menampilkan data user berdasarkan id di mobile client
Update user by id	PUT	/api/pat/register/:id	Username, password, nama, email, no hp	Memperbarui data user berdasarkan id di mobile client
Create event	POST	/api/pat/event	Event ID, nama konser, artis, deskripsi, kategori tiket, harga, jumlah tiket, lokasi, tanggal, metode pembayaran	Desktop client membuat event
Update event by id	PUT	/api/pat/event/desktop/:id	Event ID, nama konser, artis, deskripsi, kategori tiket, harga, jumlah tiket, lokasi, tanggal, metode pembayaran	Desktop client memperbarui data event berdasarkan id
Delete event by id	DELETE	/api/pat/event/desktop/:id	Event ID	Desktop client menghapus event berdasarkan id
Retrieve all event	GET	/api/pat/event	Event ID, nama konser, artis, deskripsi, kategori tiket, harga, jumlah tiket, lokasi, tanggal, metode	Menampilkan event pada mobile client dan desktop client

			pembayaran	
Order event	POST	/api/pat/order	Event ID, kategori tiket, jumlah tiket, total harga	Mobile client membuat order
Retrieve all order	GET	/api/pat/order	Event ID, kategori tiket, jumlah tiket, total harga	Menampilkan order pada desktop client
Payment event	POST	/api/pat/payment	Order ID, metode pembayaran, virtual account, status	Mobile client membuat payment
Retrieve all payment	GET	/api/pat/payment	Order ID, metode pembayaran, virtual account, status	Menampilkan payment pada desktop client
Retrieve payment by id	GET	/api/pat/payment/:id	Order ID, metode pembayaran, virtual account, status	Menampilkan payment berdasarkan id pada mobile client
Confirm payment	POST	/api/pat/confirm	Order ID, status, kode booking	Membuat confirm
Retrieve all confirm	GET	/api/pat/confirm	Order ID, status, kode booking	Menampilkan confirm pada desktop client
Retrieve confirm by id	GET	/api/pat/confirm/:orderId	Order ID, status, kode booking	Menampilkan confirm pada mobile client

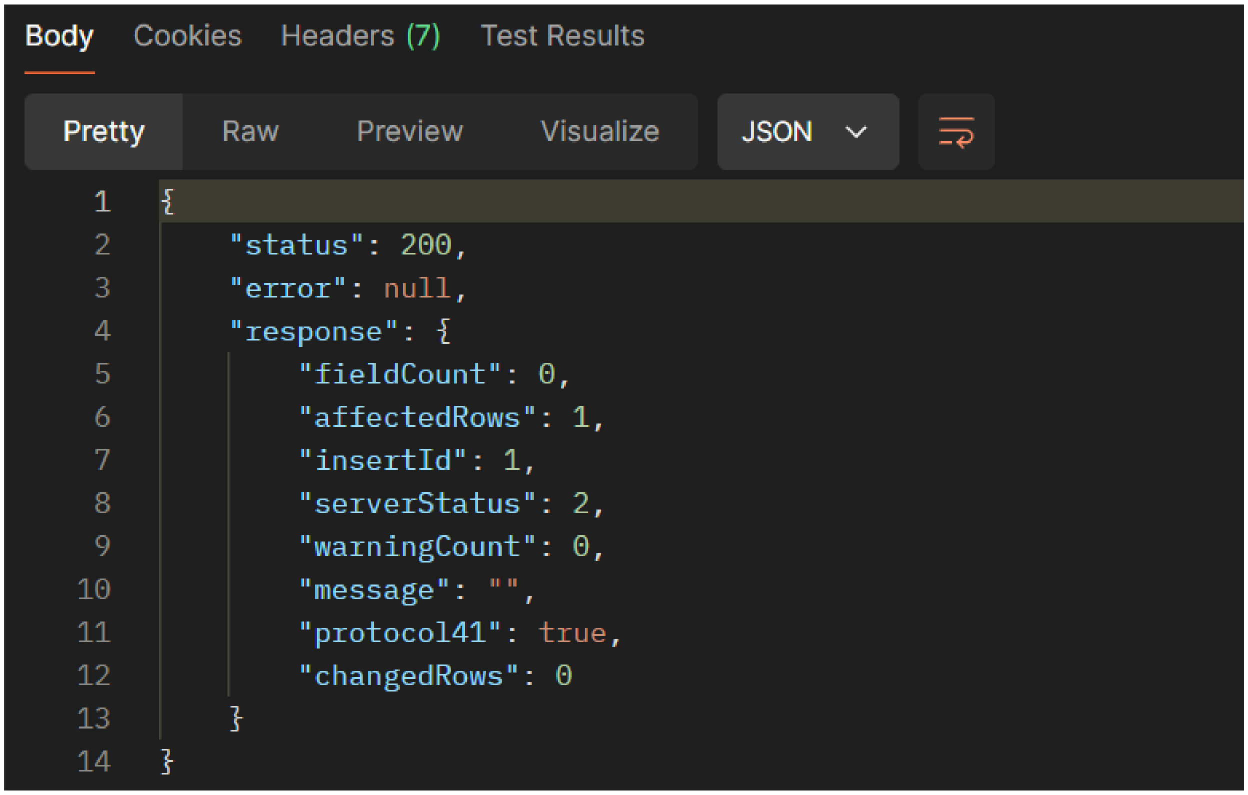
Register

```
app.post('/api/pat/register/', function (req, res) {
    let data = {
        username: req.body.username,
        pass: req.body.pass,
        nama: req.body.nama,
        email: req.body.email,
        telepon: req.body.telepon
    };

    let sql = "INSERT INTO register SET ?";
    conn.query(sql, data, function (err, result) {
        if (err) {
            console.error('Error inserting data: ', err);
            res.status(500).json({ error: 'Internal server error' });
            return;
        }
        res.json({
            "status": 200,
            "error": null,
            "response": result
        });
    });
});
```

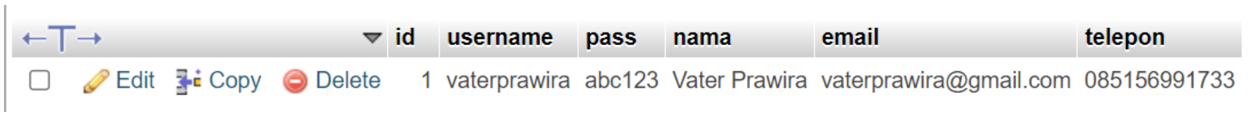


localhost: 8000/api/pat/register

```
{
    "username": "vaterprawira",
    "pass": "abc123",
    "nama": "Vater Prawira",
    "email": "vaterprawira@gmail.com",
    "telepon": "085156991733"
}
```



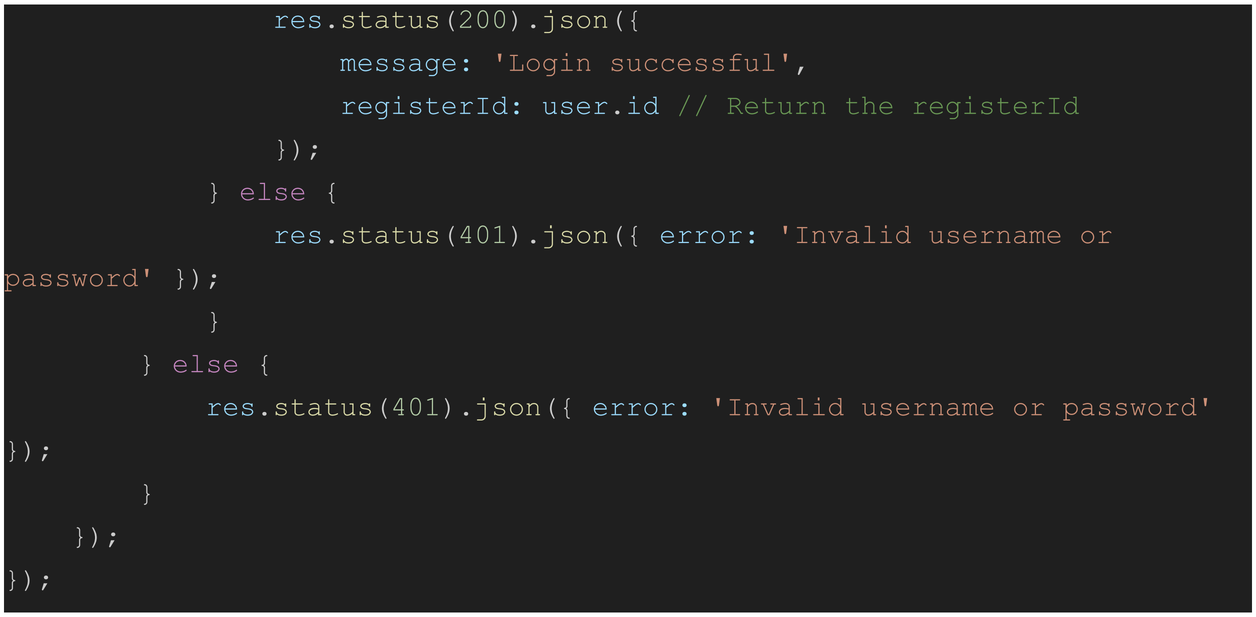
```
{
    "status": 200,
    "error": null,
    "response": {
        "fieldCount": 0,
        "affectedRows": 1,
        "insertId": 1,
        "serverStatus": 2,
        "warningCount": 0,
        "message": "",
        "protocol41": true,
        "changedRows": 0
    }
}
```



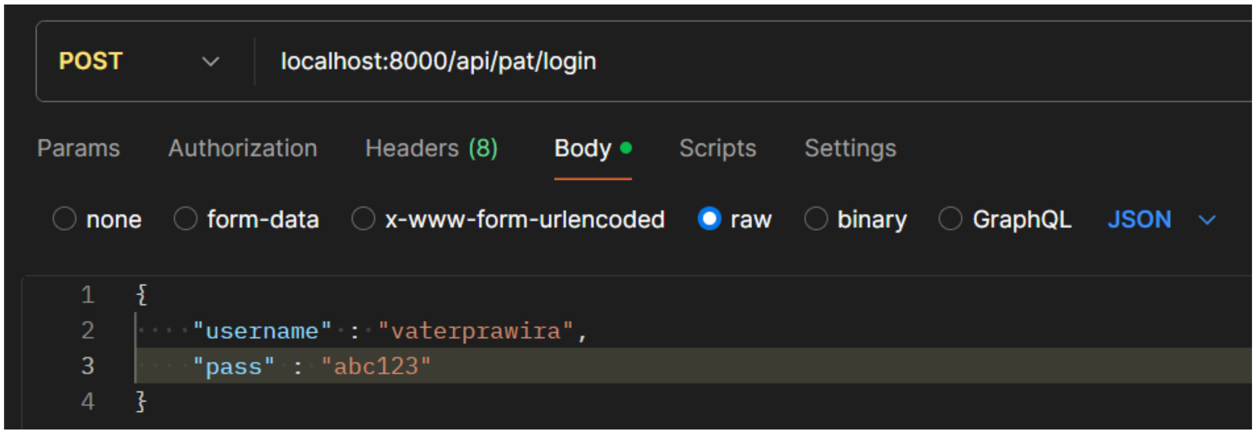
	id	username	pass	nama	email	telepon
Edit Copy Delete	1	vaterprawira	abc123	Vater Prawira	vaterprawira@gmail.com	085156991733

Login

```
app.post('/api/pat/login', function(req, res) {
    let username = req.body.username;
    let password = req.body.pass;

    let sql = "SELECT * FROM register WHERE username = ?";
    conn.query(sql, [username], function(err, result) {
        if (err) {
            console.error('Error querying database: ', err);
            res.status(500).json({ error: 'Internal server error' });
            return;
        }

        if (result.length > 0) {
            let user = result[0];
            if (user.pass === password) {
```



```
                res.status(200).json({
                    message: 'Login successful',
                    registerId: user.id // Return the registerId
                });
            } else {
                res.status(401).json({ error: 'Invalid username or
password' });
            }
        } else {
            res.status(401).json({ error: 'Invalid username or password'
});
        }
    });
});
```

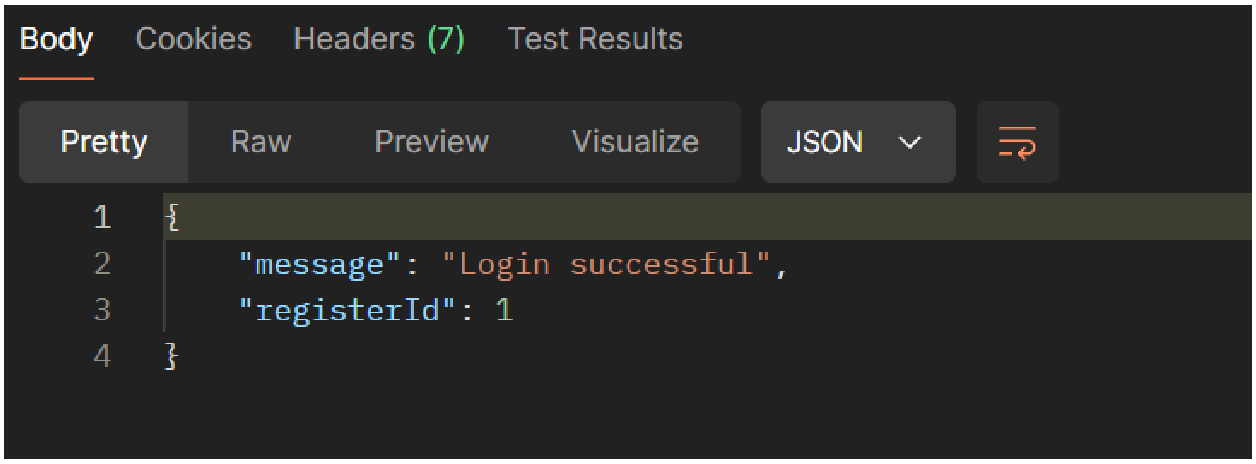


POST localhost:8000/api/pat/login

Params | Authorization | Headers (8) | Body ● | Scripts | Settings

none | form-data | x-www-form-urlencoded | [✓ raw] | binary | GraphQL | JSON

```
{
    "username" : "vaterprawira",
    "pass" : "abc123"
}
```

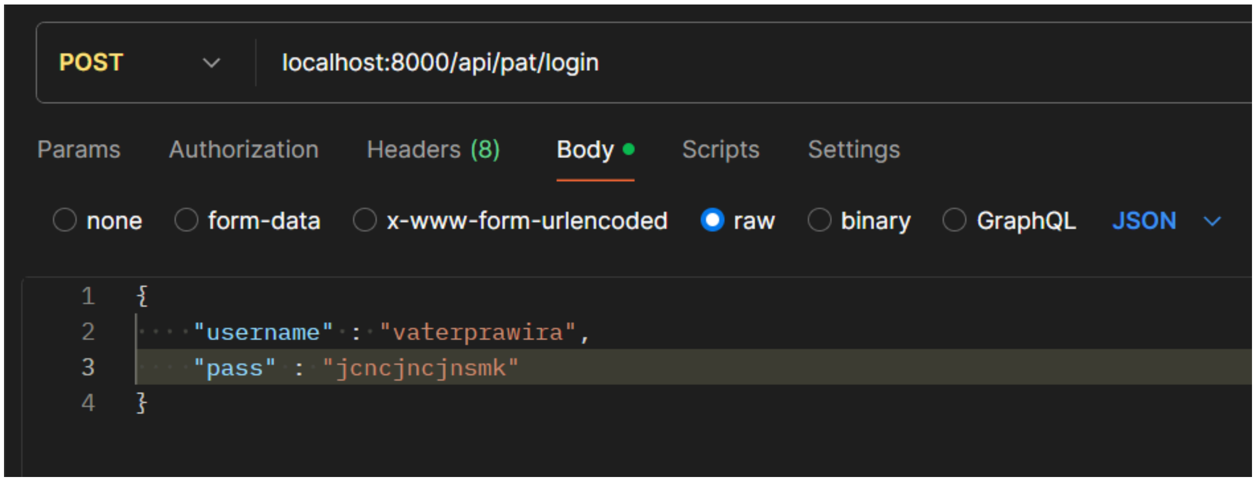


Body | Cookies | Headers (7) | Test Results

Pretty | Raw | Preview | Visualize | JSON

```
{
    "message": "Login successful",
    "registerId": 1
}
```

Jika password atau username salah

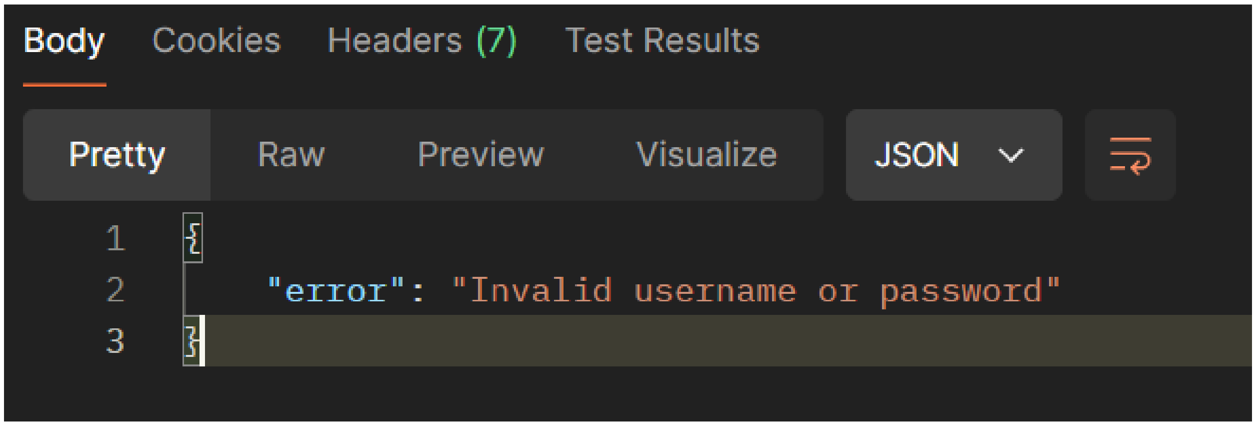


POST localhost:8000/api/pat/login

Params Authorization Headers (8) Body Scripts Settings

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{
    "username" : "vaterprawira",
    "pass" : "jcncjncjnsmk"
}
```



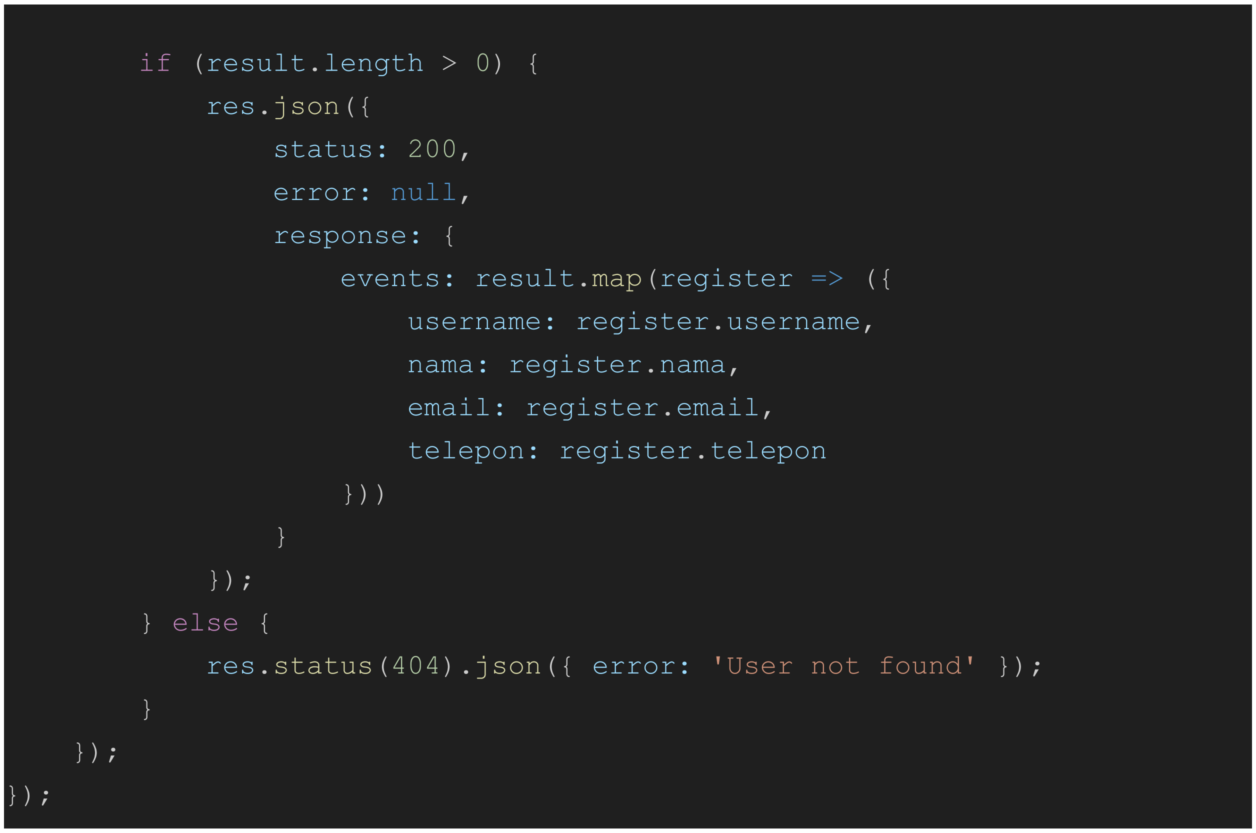
Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

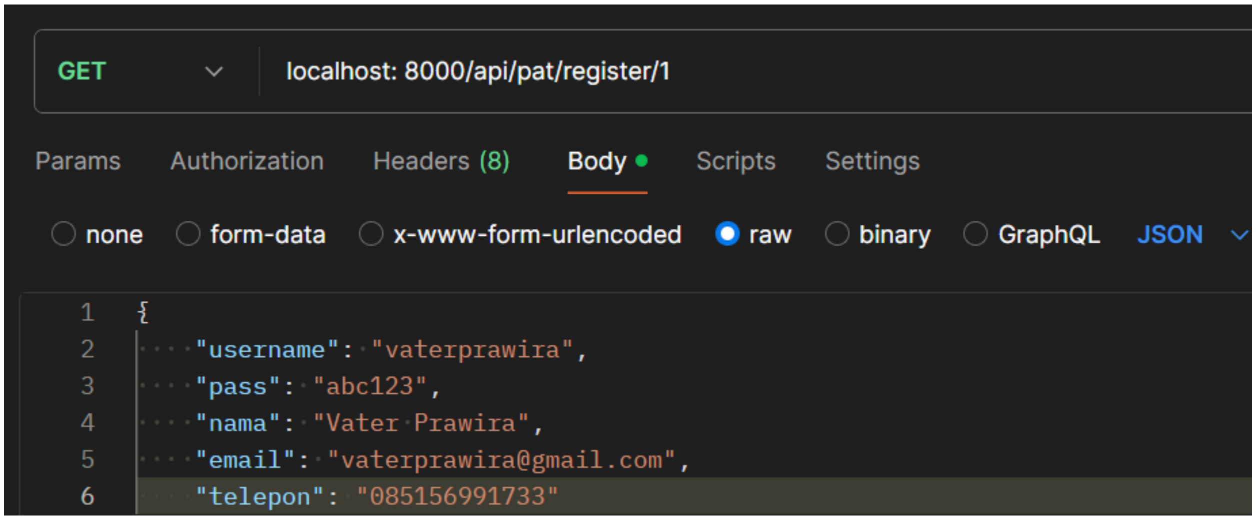
```
{
    "error": "Invalid username or password"
}
```

Retrieve user by id

```
app.get('/api/pat/register/:id', function(req, res) {
    const userId = parseInt(req.params.id);
    if (isNaN(userId)) {
        return res.status(400).json({ error: 'Invalid user ID' });
    }

    let sql = "SELECT * FROM register WHERE id = ?";
    conn.query(sql, [userId], function(err, result) {
        if (err) {
            console.error('Error querying database: ', err);
            res.status(500).json({ error: 'Internal server error' });
            return;
        }
```



```
        if (result.length > 0) {
            res.json({
                status: 200,
                error: null,
                response: {
                    events: result.map(register => ({
                        username: register.username,
                        nama: register.nama,
                        email: register.email,
                        telepon: register.telepon
                    }))
                }
            });
        } else {
            res.status(404).json({ error: 'User not found' });
        }
    });
});
```

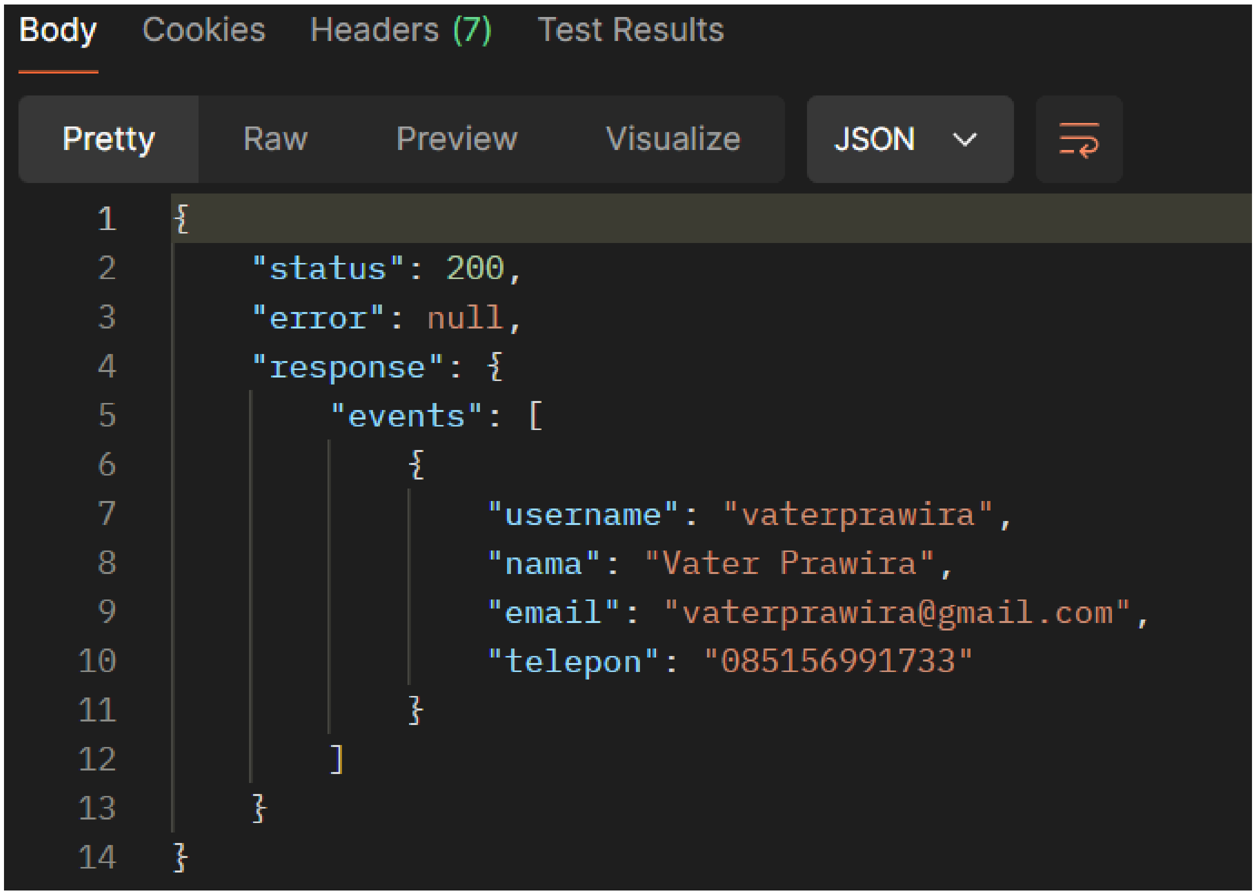


GET localhost: 8000/api/pat/register/1

Params Authorization Headers (8) Body Scripts Settings

( ) none ( ) form-data ( ) x-www-form-urlencoded (●) raw ( ) binary ( ) GraphQL JSON

```
{
    "username": "vaterprawira",
    "pass": "abc123",
    "nama": "Vater Prawira",
    "email": "vaterprawira@gmail.com",
    "telepon": "085156991733"
```



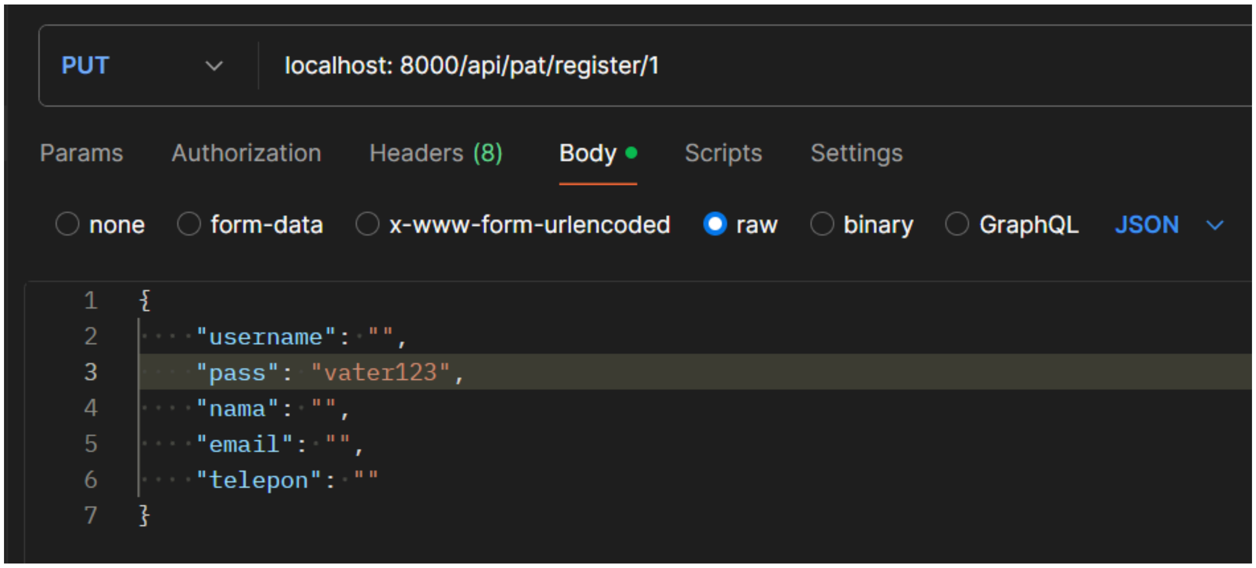
```json
{
    "status": 200,
    "error": null,
    "response": {
        "events": [
            {
                "username": "vaterprawira",
                "nama": "Vater Prawira",
                "email": "vaterprawira@gmail.com",
                "telepon": "085156991733"
            }
        ]
    }
}
```

Update user by id

```
app.put('/api/pat/register/:id', function(req, res) {
    let id = parseInt(req.params.id);
    if (isNaN(id)) {
        return res.status(400).json({ error: 'Invalid user ID' });
    }

    let newData = {
        username: req.body.username,
        pass: req.body.pass,
        nama: req.body.nama,
        email: req.body.email,
        telepon: req.body.telepon
    };

    let sql = "UPDATE register SET ";
    let values = [];
```

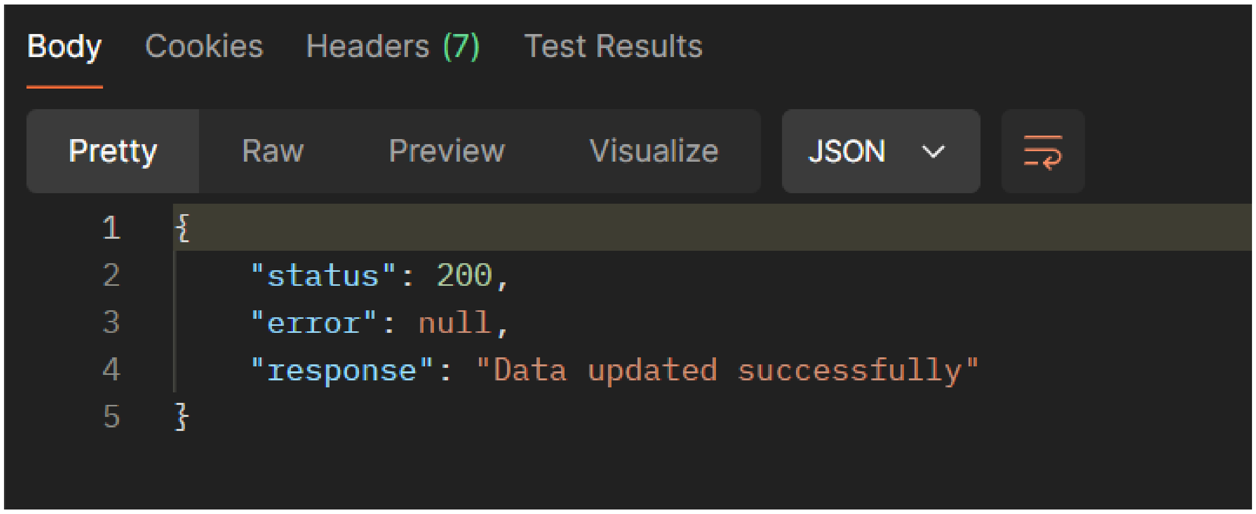
```
// Check if each field is present in the request body
if (newData.username) {
    sql += "username = ?, ";
    values.push(newData.username);
}
if (newData.pass) {
    sql += "pass = ?, ";
    values.push(newData.pass);
}
if (newData.nama) {
    sql += "nama = ?, ";
    values.push(newData.nama);
}
if (newData.email) {
    sql += "email = ?, ";
    values.push(newData.email);
}
if (newData.telepon) {
    sql += "telepon = ?, ";
    values.push(newData.telepon);
}

// Remove the trailing comma and space
sql = sql.slice(0, -2);

// Add the WHERE clause
sql += " WHERE id = ?";
values.push(id);

conn.query(sql, values, function(err, result) {
    if (err) {
        console.error('Error updating data: ', err);
        res.status(500).json({ error: 'Internal server error' });
        return;
    }
    if (result.affectedRows == 0) {
        // No rows were affected, meaning the user doesn't exist
        res.status(404).json({ error: 'User not found' });
        return;
        }
        res.json({
            status: 200,
            error: null,
            response: "Data updated successfully"
        });
    });
});
```

Dapat mengupdate data hanya di semua parameter ataupun 1 parameter saja

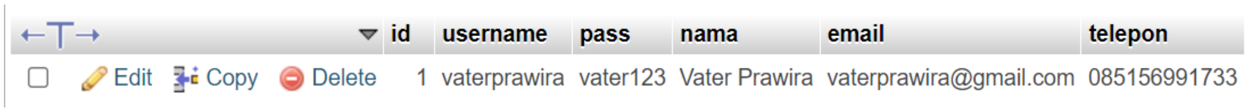


PUT localhost: 8000/api/pat/register/1

```
{
    "username": "",
    "pass": "vater123",
    "nama": "",
    "email": "",
    "telepon": ""
}
```



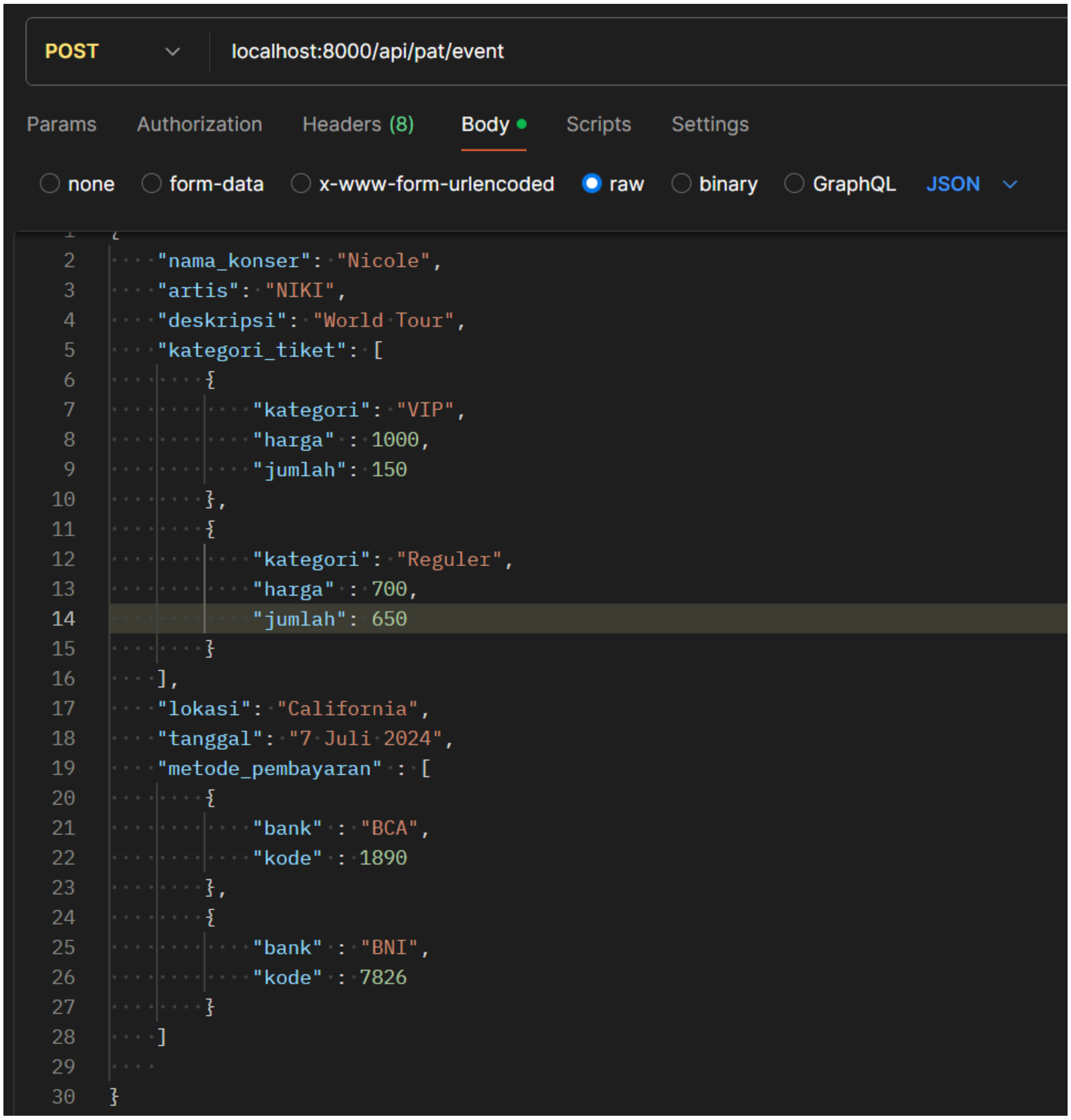
```
{
    "status": 200,
    "error": null,
    "response": "Data updated successfully"
}
```



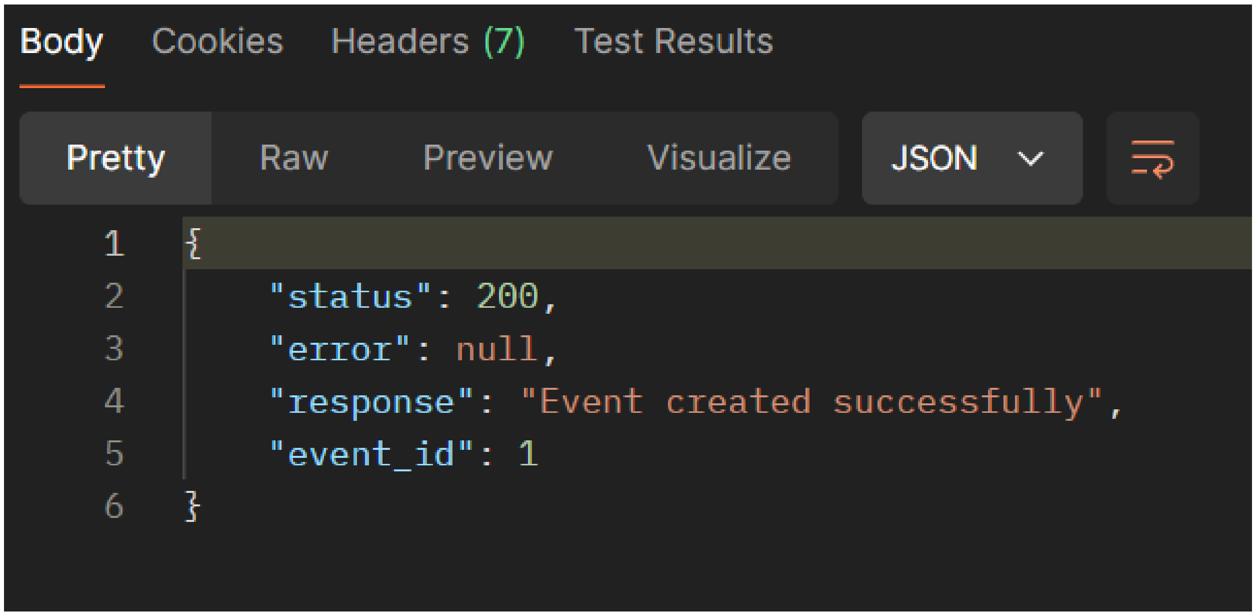
	id	username	pass	nama	email	telepon
Edit Copy Delete	1	vaterprawira	vater123	Vater Prawira	vaterprawira@gmail.com	085156991733

Create event

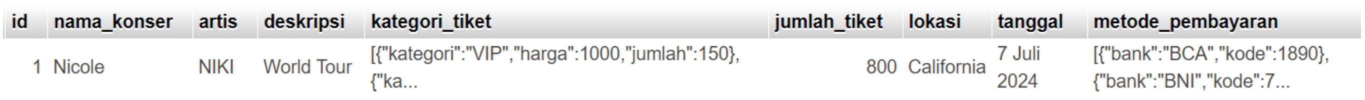
```
app.post('/api/pat/event', function(req, res) {
    try {
        // Parse body data
        let nama_konser = req.body.nama_konser;
        let artis = req.body.artis || ''; // Set default value if not
provided
        let deskripsi = req.body.deskripsi;
        let kategoriTiket = req.body.kategori_tiket;
        let lokasi = req.body.lokasi;
        let tanggal = req.body.tanggal;
        let metodePembayaran = req.body.metode_pembayaran;

        // Check if kategoriTiket and metodePembayaran are strings and
parse them if needed
        if (typeof kategoriTiket === 'string') {
            kategoriTiket = JSON.parse(kategoriTiket);
        }
        if (typeof metodePembayaran === 'string') {
            metodePembayaran = JSON.parse(metodePembayaran);
        }

        // Calculate total tickets
        let jumlahTiket = kategoriTiket.reduce((total, kategori) => total
+ (kategori.jumlah || 0), 0);

        // Create data object
        let data = {
            nama_konser,
            artis,
            deskripsi,
            kategori_tiket: JSON.stringify(kategoriTiket), // Convert
array to JSON string
            jumlah_tiket: jumlahTiket, // Use calculated total tickets
            lokasi,
            tanggal,
            metode_pembayaran: JSON.stringify(metodePembayaran) // Convert
array to JSON string
        };
        // Insert data into database
        let sql = "INSERT INTO event SET ?";
        conn.query(sql, data, function(err, result) {
            if (err) {
                console.error('Error inserting event: ', err);
                res.status(500).json({ error: 'Internal server error' });
                return;
            }
            res.json({
                "status": 200,
                "error": null,
                "response": "Event created successfully",
                "event_id": result.insertId
            });
        });
    } catch (error) {
        console.error('Error processing request: ', error);
        res.status(400).json({ error: 'Invalid request data' });
    }
});
```


POST localhost:8000/api/pat/event

```
    "nama_konser": "Nicole",
    "artis": "NIKI",
    "deskripsi": "World Tour",
    "kategori_tiket": [
        {
            "kategori": "VIP",
            "harga" : 1000,
            "jumlah": 150
        },
        {
            "kategori": "Reguler",
            "harga" : 700,
            "jumlah": 650
        }
    ],
    "lokasi": "California",
    "tanggal": "7 Juli 2024",
    "metode_pembayaran" : [
        {
            "bank" : "BCA",
            "kode" : 1890
        },
        {
            "bank" : "BNI",
            "kode" : 7826
        }
    ]
    
}
```



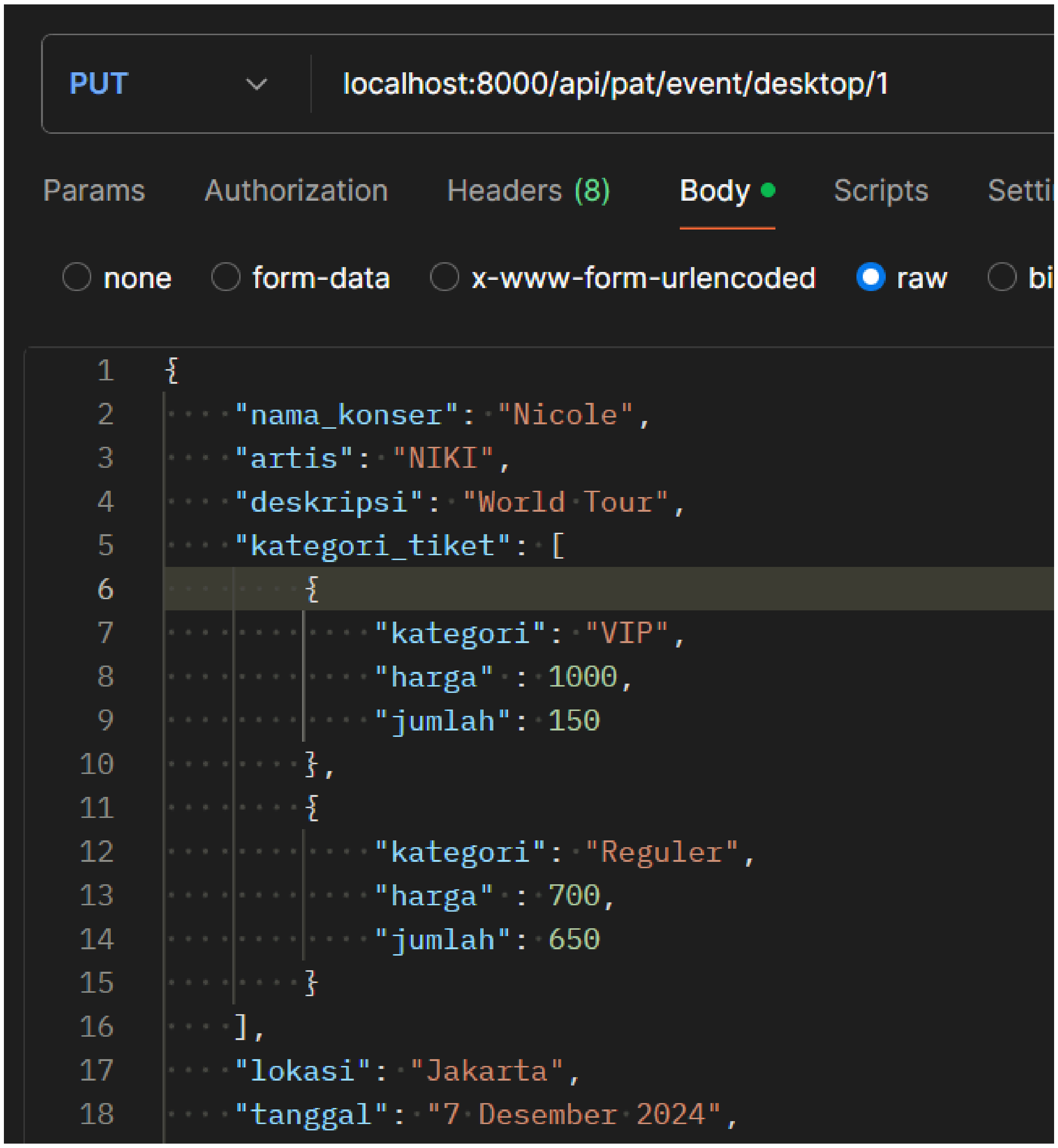
```
{
    "status": 200,
    "error": null,
    "response": "Event created successfully",
    "event_id": 1
}
```



id	nama_konser	artis	deskripsi	kategori_tiket	jumlah_tiket	lokasi	tanggal	metode_pembayaran
1	Nicole	NIKI	World Tour	[{"kategori":"VIP","harga":1000,"jumlah":150}, {"ka...	800	California	7 Juli 2024	[{"bank":"BCA","kode":1890}, {"bank":"BNI","kode":7...

Update event by id

```
app.put('/api/pat/event/desktop/:id', upload.single('gambar'),
function(req, res) {
    let id = req.params.id;
    let newData = {
        nama_konser: req.body.nama_konser,
        artis: req.body.artis,
        deskripsi: req.body.deskripsi,
        kategori_tiket: req.body.kategori_tiket ?
JSON.stringify(req.body.kategori_tiket) : null, // Convert to JSON string
if present
        jumlah_tiket: req.body.jumlah_tiket,
        lokasi: req.body.lokasi,
        tanggal: req.body.tanggal,
        metode_pembayaran: req.body.metode_pembayaran ?
JSON.stringify(req.body.metode_pembayaran) : null, // Convert to JSON
string if present

    };
    let sql = "UPDATE event SET ";
    let values = [];

    // Check if each field is present in the request body
    if (newData.nama_konser) {
        sql += "nama_konser = ?, ";
        values.push(newData.nama_konser);
    }
    if (newData.artis) {
        sql += "artis = ?, ";
        values.push(newData.artis);
    }
    if (newData.deskripsi) {
        sql += "deskripsi = ?, ";
        values.push(newData.deskripsi);
    }
    if (newData.kategori_tiket) {
        sql += "kategori_tiket = ?, ";
        values.push(newData.kategori_tiket);
    }
    if (newData.jumlah_tiket) {
        sql += "jumlah_tiket = ?, ";
        values.push(newData.jumlah_tiket);
    }
    if (newData.lokasi) {
        sql += "lokasi = ?, ";
        values.push(newData.lokasi);
    }
    if (newData.tanggal) {
        sql += "tanggal = ?, ";
        values.push(newData.tanggal);
    }
    if (newData.metode_pembayaran) {
        sql += "metode_pembayaran = ?, ";
        values.push(newData.metode_pembayaran);
    }

    // Remove the trailing comma and space
    sql = sql.slice(0, -2);
    // Add the WHERE clause
    sql += " WHERE id = ?";
    values.push(id);

    conn.query(sql, values, function(err, result) {
        if (err) {
            console.error('Error updating event: ', err);
            res.status(500).json({ error: 'Internal server error' });
            return;
        }
        if (result.affectedRows == 0) {
            // No rows were affected, meaning the event doesn't exist
            res.status(404).json({ error: 'Event not found' });
            return;
        }
        res.json({
            "status": 200,
            "error": null,
            "response": "Event updated successfully"
        });
    });
});
```

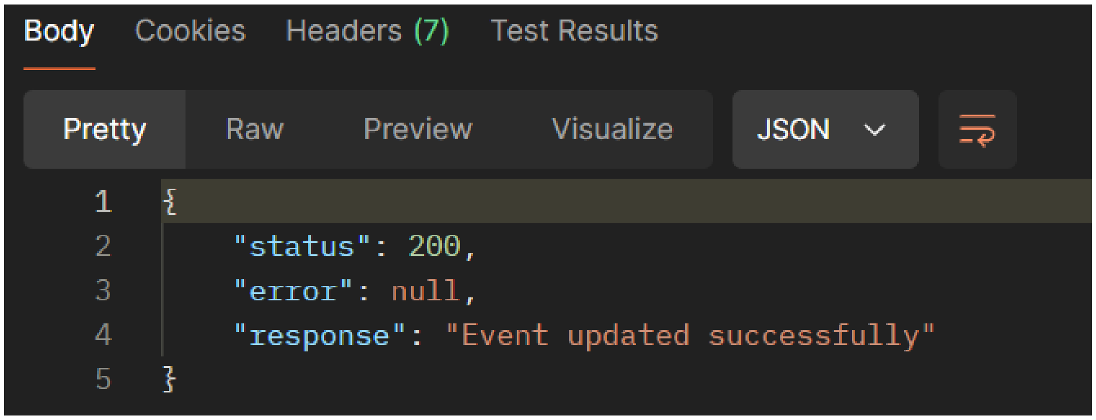


PUT localhost:8000/api/pat/event/desktop/1

Params   Authorization   Headers (8)   Body ●   Scripts   Setti

○ none   ○ form-data   ○ x-www-form-urlencoded   [✓ raw]   ○ bi

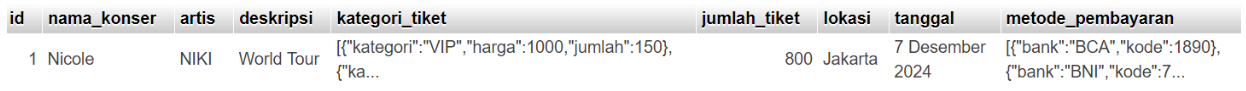
```
{
    "nama_konser": "Nicole",
    "artis": "NIKI",
    "deskripsi": "World Tour",
    "kategori_tiket": [
        {
            "kategori": "VIP",
            "harga" : 1000,
            "jumlah": 150
        },
        {
            "kategori": "Reguler",
            "harga" : 700,
            "jumlah": 650
        }
    ],
    "lokasi": "Jakarta",
    "tanggal": "7 Desember 2024",
```



Body   Cookies   Headers (7)   Test Results

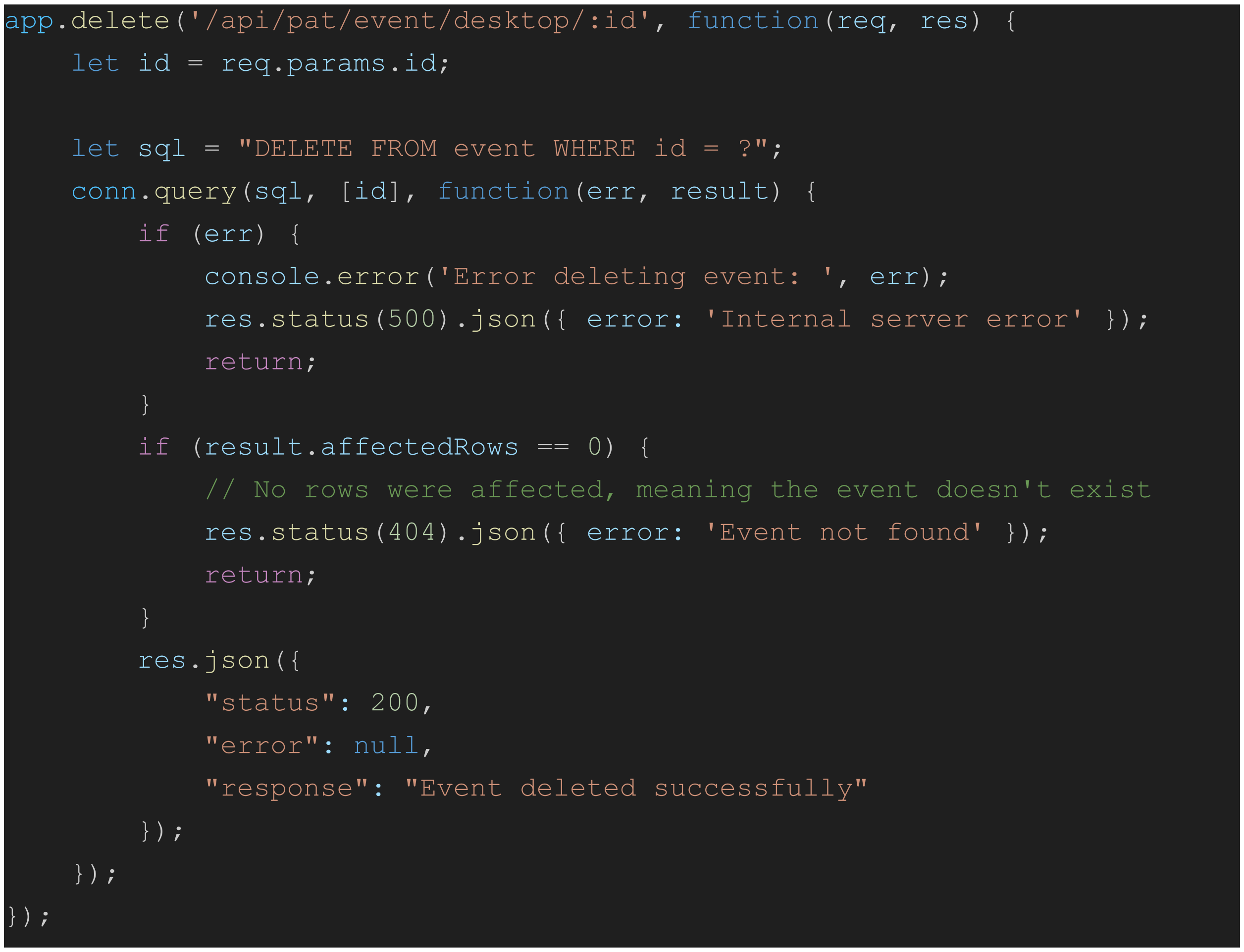
Pretty   Raw   Preview   Visualize   JSON

```
{
    "status": 200,
    "error": null,
    "response": "Event updated successfully"
}
```

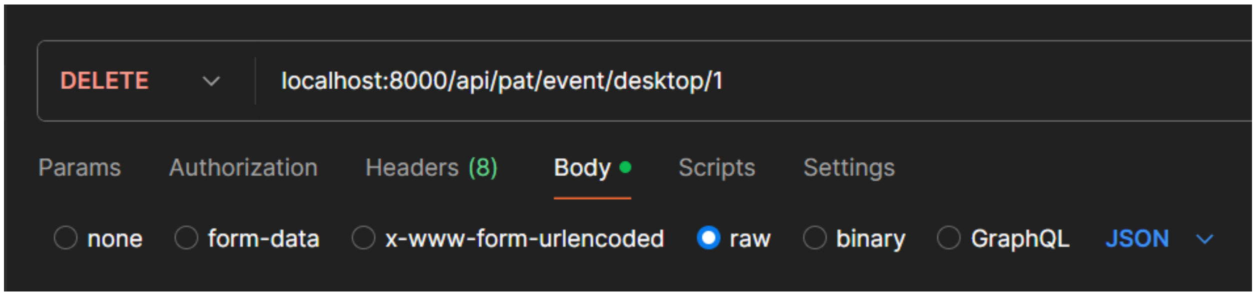


id	nama_konser	artis	deskripsi	kategori_tiket	jumlah_tiket	lokasi	tanggal	metode_pembayaran
1	Nicole	NIKI	World Tour	[{"kategori":"VIP","harga":1000,"jumlah":150}, {"ka...	800	Jakarta	7 Desember 2024	[{"bank":"BCA","kode":1890}, {"bank":"BNI","kode":7...

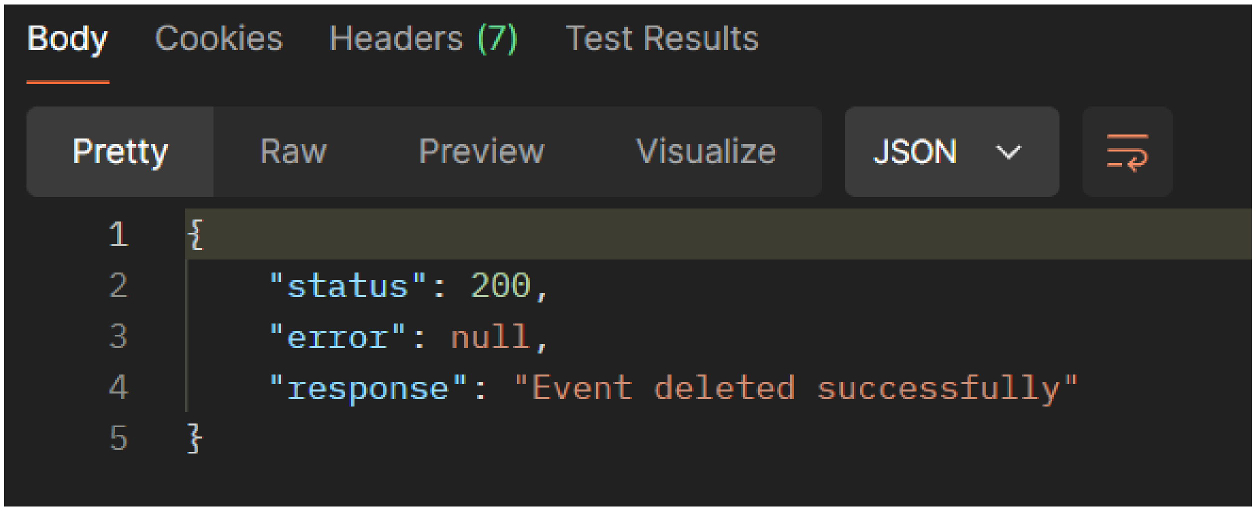
Delete event by id



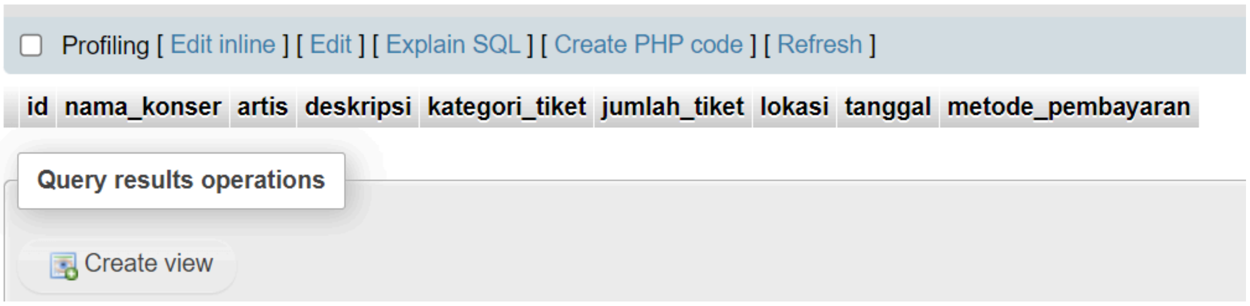
```
app.delete('/api/pat/event/desktop/:id', function(req, res) {
    let id = req.params.id;

    let sql = "DELETE FROM event WHERE id = ?";
    conn.query(sql, [id], function(err, result) {
        if (err) {
            console.error('Error deleting event: ', err);
            res.status(500).json({ error: 'Internal server error' });
            return;
        }
        if (result.affectedRows == 0) {
            // No rows were affected, meaning the event doesn't exist
            res.status(404).json({ error: 'Event not found' });
            return;
        }
        res.json({
            "status": 200,
            "error": null,
            "response": "Event deleted successfully"
        });
    });
});
```



DELETE localhost:8000/api/pat/event/desktop/1



```
{
    "status": 200,
    "error": null,
    "response": "Event deleted successfully"
}
```

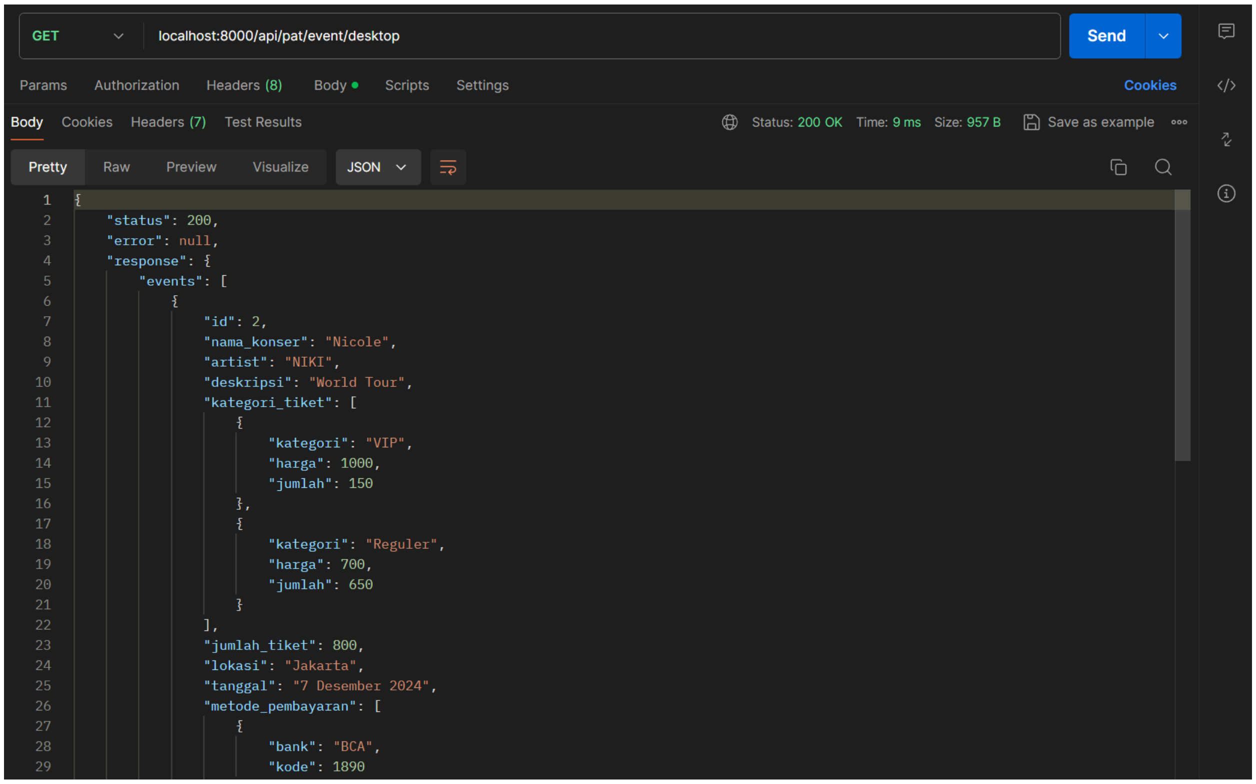

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Query results operations
Create view

id	nama_konser	artis	deskripsi	kategori_tiket	jumlah_tiket	lokasi	tanggal	metode_pembayaran

Retrieve all event

Menampilkan seluruh event di desktop client

```
app.get('/api/pat/event/desktop', function(req, res) {
    let sql = "SELECT * FROM event";
    conn.query(sql, function(err, result) {
        if (err) {
            console.error('Error fetching events: ', err);
            res.status(500).json({ error: 'Internal server error' });
            return;
        }

        res.json({
            "status": 200,
            "error": null,
            "response": {
                events: result.map(event => {
                    return {
                        id: event.id,
                        nama_konser: event.nama_konser,
                        artist: event.artis,
                        deskripsi: event.deskripsi,
                        kategori_tiket: JSON.parse(event.kategori_tiket),
// Convert JSON string back to object
                        jumlah_tiket: event.jumlah_tiket,
                        lokasi: event.lokasi,
                        tanggal: event.tanggal,
                        metode_pembayaran:
JSON.parse(event.metode_pembayaran), // Convert JSON string back to object
                    };
                })
            }
        });
    });
});
```



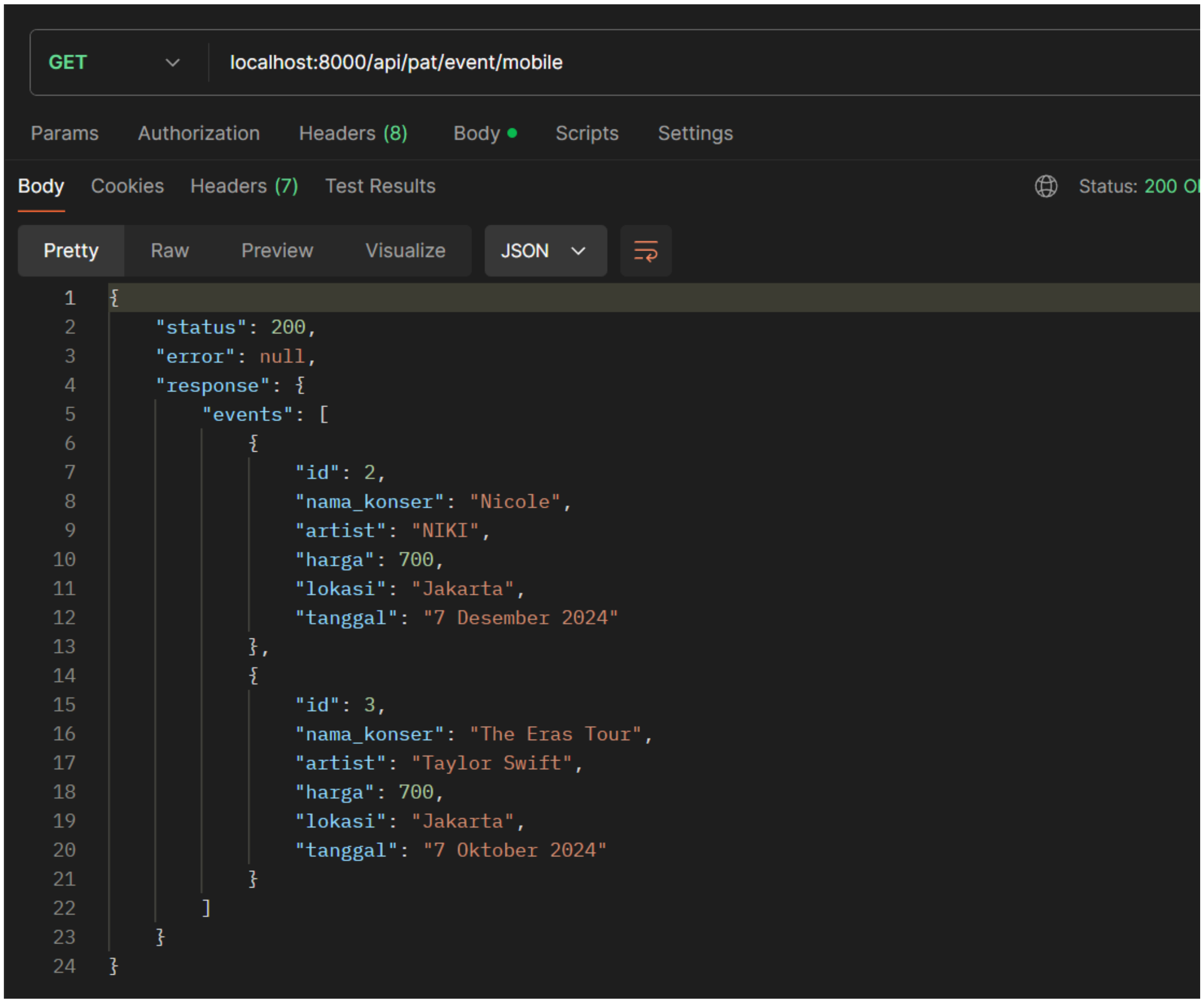
GET localhost:8000/api/pat/event/desktop

Status: 200 OK Time: 9 ms Size: 957 B

```
{
    "status": 200,
    "error": null,
    "response": {
        "events": [
            {
                "id": 2,
                "nama_konser": "Nicole",
                "artist": "NIKI",
                "deskripsi": "World Tour",
                "kategori_tiket": [
                    {
                        "kategori": "VIP",
                        "harga": 1000,
                        "jumlah": 150
                    },
                    {
                        "kategori": "Reguler",
                        "harga": 700,
                        "jumlah": 650
                    }
                ],
                "jumlah_tiket": 800,
                "lokasi": "Jakarta",
                "tanggal": "7 Desember 2024",
                "metode_pembayaran": [
                    {
                        "bank": "BCA",
                        "kode": 1890
```

Menampilkan seluruh event mobile client

```
app.get('/api/pat/event/mobile', function(req, res) {
    let sql = "SELECT * FROM event";
    conn.query(sql, function(err, result) {
        if (err) {
            console.error('Error fetching events: ', err);
            res.status(500).json({ error: 'Internal server error' });
            return;
        }

        res.json({
            "status": 200,
            "error": null,
            "response": {
                events: result.map(event => {
                    // Parse the kategori_tiket JSON string
                    const kategori_tiket =
JSON.parse(event.kategori_tiket);

                    // Extract the lowest price from kategori_tiket
                    const harga_terkecil =
Math.min(...kategori_tiket.map(ticket => ticket.harga));

                    return {
                        id: event.id,
                        nama_konser: event.nama_konser,
                        artist: event.artis,
                        harga: harga_terkecil,
                        lokasi: event.lokasi,
                        tanggal: event.tanggal

                    };
                })
            }
        });
    });
});
```

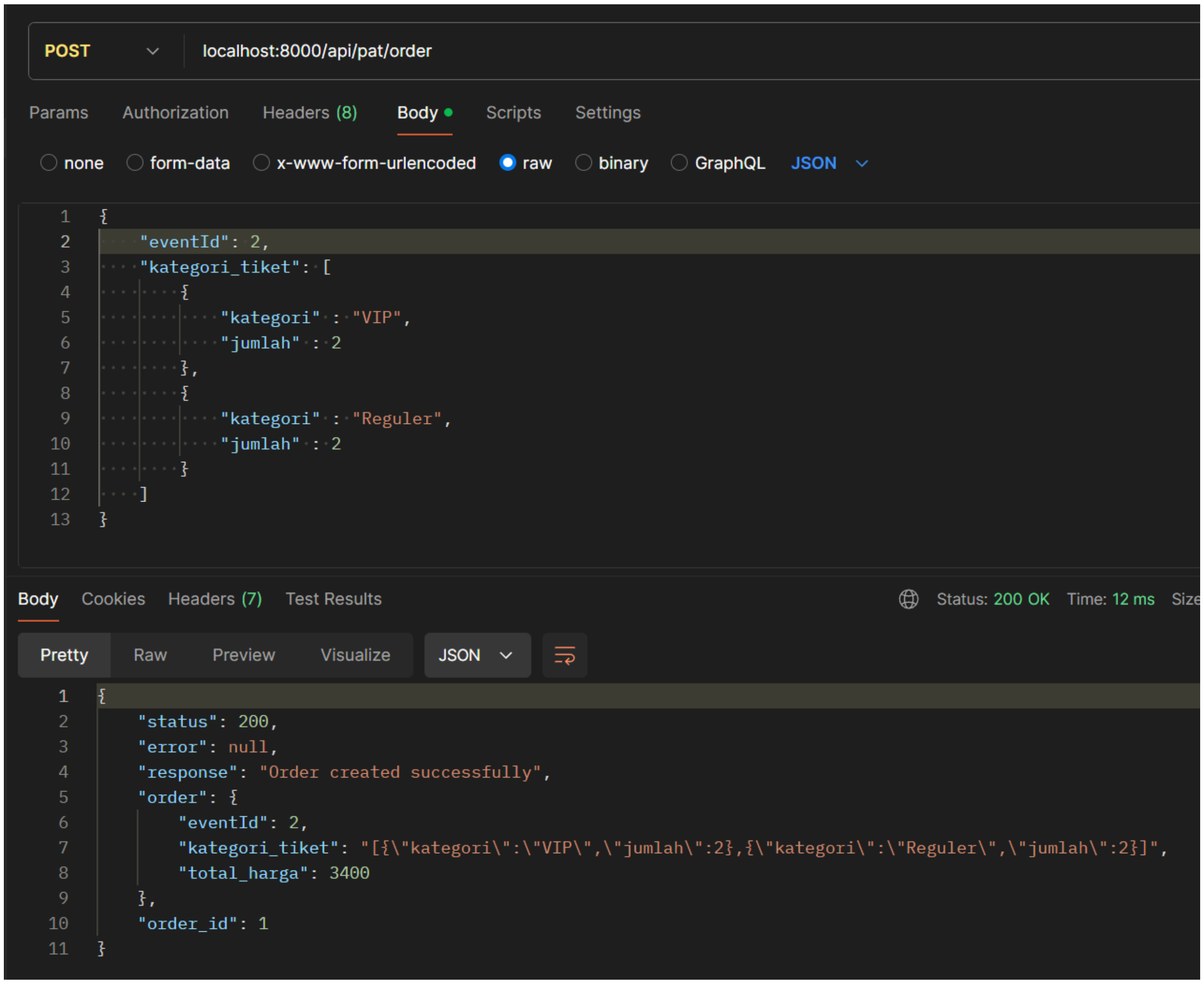


```
localhost:8000/api/pat/event/mobile
```

```json
{
    "status": 200,
    "error": null,
    "response": {
        "events": [
            {
                "id": 2,
                "nama_konser": "Nicole",
                "artist": "NIKI",
                "harga": 700,
                "lokasi": "Jakarta",
                "tanggal": "7 Desember 2024"
            },
            {
                "id": 3,
                "nama_konser": "The Eras Tour",
                "artist": "Taylor Swift",
                "harga": 700,
                "lokasi": "Jakarta",
                "tanggal": "7 Oktober 2024"
            }
        ]
    }
}
```

Order event

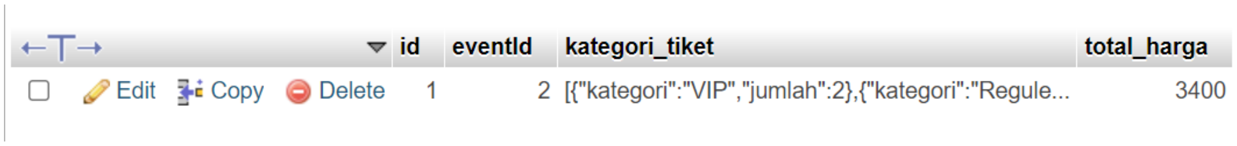
```javascript
app.post('/api/pat/order', function(req, res) {
    const { eventId, kategori_tiket } = req.body;

    if (!Array.isArray(kategori_tiket) || kategori_tiket.length === 0) {
        return res.status(400).json({ error: 'Invalid kategori_tiket
format' });
    }

    const sql = "SELECT * FROM event WHERE id = ?";

    conn.query(sql, [eventId], function(err, result) {
        if (err) {
            console.error('Error retrieving event: ', err);
            return res.status(500).json({ error: 'Internal server error'
});
        }

        if (result.length === 0) {
            return res.status(404).json({ error: 'Event not found' });
        }

        const event = result[0];

        let kategori_tiket_parsed;
        try {
            kategori_tiket_parsed = JSON.parse(event.kategori_tiket);
        } catch (error) {
            console.error('Error parsing JSON: ', error);
            return res.status(500).json({ error: 'Internal server error'
});
        }

        let total_harga = 0;
        let total_jumlah_tiket = 0;

        for (const tiket of kategori_tiket) {
            const { kategori, jumlah } = tiket;
            if (isNaN(jumlah) || jumlah < 1) {
                return res.status(400).json({ error: 'Invalid jumlah for a
ticket category' });
            }

            const ticketCategory = kategori_tiket_parsed.find(category =>
category.kategori === kategori);
            if (!ticketCategory) {
                return res.status(400).json({ error: `Invalid ticket
category: ${kategori}` });
            }

            const harga_tiket = ticketCategory.harga;
            if (isNaN(harga_tiket)) {
                console.error('Invalid ticket price:', harga_tiket);
                return res.status(500).json({ error: 'Internal server
error' });
            }

            total_harga += harga_tiket * jumlah;
            total_jumlah_tiket += jumlah;

            if (ticketCategory.jumlah < jumlah) {
                return res.status(400).json({ error: `Not enough tickets
available for category: ${kategori}` });
            }

            // Reduce the available tickets for the category
            ticketCategory.jumlah -= jumlah;
        }

        const updatedJumlahTiket = event.jumlah_tiket -
total_jumlah_tiket;

        const orderData = {
            eventId,
            total_harga,
            kategori_tiket: JSON.stringify(kategori_tiket) // Store as
JSON string
        };

        const updateEventSql = "UPDATE event SET kategori_tiket = ?,
jumlah_tiket = ? WHERE id = ?";
        const updatedKategoriTiket =
JSON.stringify(kategori_tiket_parsed);

        conn.query(updateEventSql, [updatedKategoriTiket,
updatedJumlahTiket, eventId], function(err, result) {
            if (err) {
                console.error('Error updating event: ', err);
                return res.status(500).json({ error: 'Internal server
error' });
            }
            const orderSql = "INSERT INTO `order` (`eventId`, `kategori_tiket`, `total_harga`) VALUES (?, ?, ?)";
            conn.query(orderSql, [orderData.eventId, orderData.kategori_tiket, orderData.total_harga], function(err, result) {
                if (err) {
                    console.error('Error inserting order: ', err);
                    return res.status(500).json({ error: 'Internal server error' });
                }

                res.json({
                    status: 200,
                    error: null,
                    response: "Order created successfully",
                    order: {
                        eventId: orderData.eventId,
                        kategori_tiket: orderData.kategori_tiket,
                        total_harga: orderData.total_harga
                    },
                    order_id: result.insertId
                });
            });
        });
    });
});
```


POST localhost:8000/api/pat/order

```
{
    "eventId": 2,
    "kategori_tiket": [
        {
            "kategori" : "VIP",
            "jumlah" : 2
        },
        {
            "kategori" : "Reguler",
            "jumlah" : 2
        }
    ]
}
```

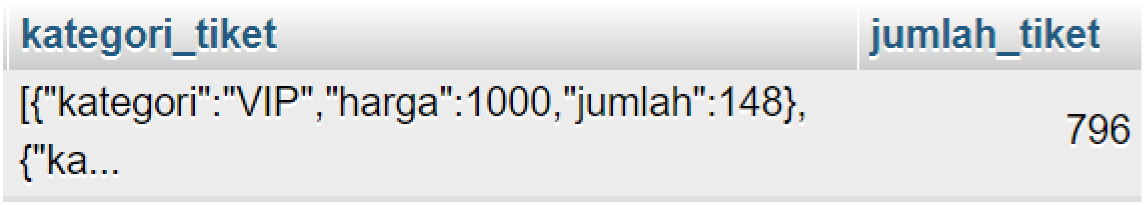
Status: 200 OK Time: 12 ms

```
{
    "status": 200,
    "error": null,
    "response": "Order created successfully",
    "order": {
        "eventId": 2,
        "kategori_tiket": "[{\"kategori\":\"VIP\",\"jumlah\":2},{\"kategori\":\"Reguler\",\"jumlah\":2}]",
        "total_harga": 3400
    },
    "order_id": 1
}
```



	id	eventId	kategori_tiket	total_harga
Edit Copy Delete	1	2	[{"kategori":"VIP","jumlah":2},{"kategori":"Regule...	3400

Ketika melakukan order, jumlah tiket yang ada pada database event akan berkurang sesuai dengan tiket yang di order.

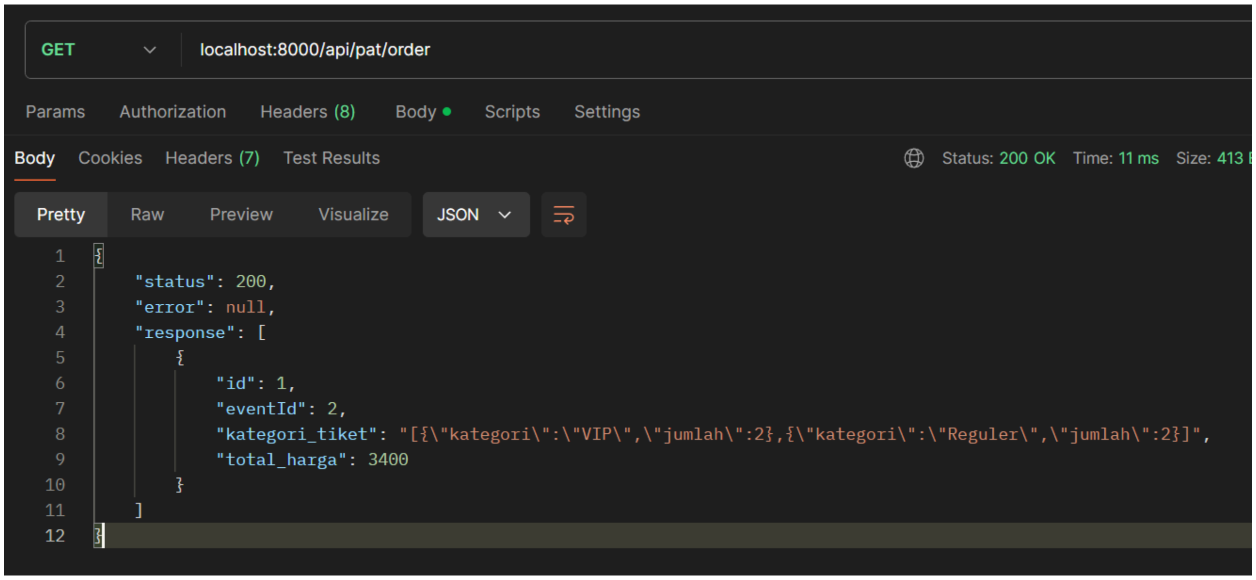


kategori_tiket	jumlah_tiket
[{"kategori":"VIP","harga":1000,"jumlah":148},{"ka...	796

Pengurangan tersebut terjadi untuk setiap kategori yang dipesan dan total ketersediaan tiket.

Retrieve all order

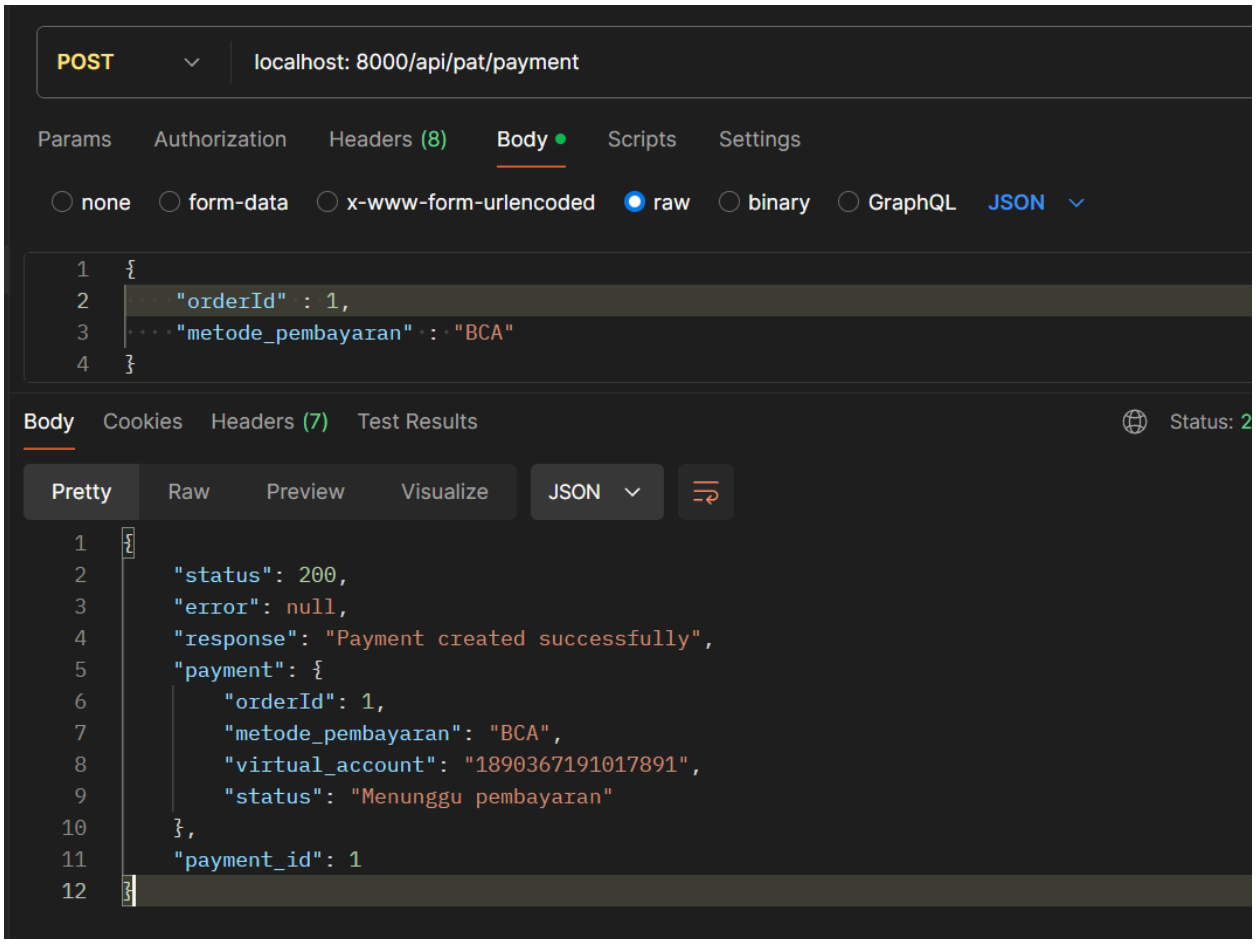
```
app.get('/api/pat/order', function(req, res) {
    const sql = "SELECT * FROM `order`";

    conn.query(sql, function(err, result) {
        if (err) {
            console.error('Error retrieving orders: ', err);
            res.status(500).json({ error: 'Internal server error' });
            return;
        }

        res.json({
            "status": 200,
            "error": null,
            "response": result
        });
    });
});
```



```
{
    "status": 200,
    "error": null,
    "response": [
        {
            "id": 1,
            "eventId": 2,
            "kategori_tiket": "[{\"kategori\":\"VIP\",\"jumlah\":2},{\"kategori\":\"Reguler\",\"jumlah\":2}]",
            "total_harga": 3400
        }
    ]
}
```

Payment event

```
app.post('/api/pat/payment', function(req, res) {
    const { orderId, metode_pembayaran } = req.body;

    if (!orderId) {
        return res.status(400).json({ error: 'Missing orderId' });
    }

    const orderSql = "SELECT eventId FROM `order` WHERE id = ?";
    conn.query(orderSql, [orderId], function(err, orderResult) {
        if (err) {
            console.error('Error retrieving order: ', err);
             return res.status(500).json({ error: 'Internal server error'
});
        }

        if (orderResult.length === 0) {
            return res.status(404).json({ error: 'Order not found' });
        }

        const eventId = orderResult[0].eventId;

         const eventSql = "SELECT metode_pembayaran FROM event WHERE id =
?";
        conn.query(eventSql, [eventId], function(err, eventResult) {
            if (err) {
                console.error('Error retrieving event: ', err);
                    return res.status(500).json({ error: 'Internal server
error' });
            }

            if (eventResult.length === 0) {
                return res.status(404).json({ error: 'Event not found' });
            }

            let metode_pembayaran_parsed;
            try {
                                            metode_pembayaran_parsed   =
JSON.parse(eventResult[0].metode_pembayaran);
            } catch (error) {
                   console.error('Error parsing metode_pembayaran JSON: ',
error);
                    return res.status(500).json({ error: 'Internal server
error' });
            }

             const paymentMethod = metode_pembayaran_parsed.find(method =>
method.bank === metode_pembayaran);
            if (!paymentMethod) {
                    return res.status(400).json({ error: 'Invalid payment
method' });
            }

            const kode = paymentMethod.kode;

            // Generate 12 random digits for the virtual account
            const randomDigits = () => Math.floor(Math.random() * 10);
             const virtual_account = `${kode}${Array.from({ length: 12 },
randomDigits).join('')}`;
            const status = "Menunggu pembayaran";

            const paymentData = {
                orderId,
                metode_pembayaran,
                virtual_account,
                status
            };

            const paymentSql = "INSERT INTO payment SET ?";
            conn.query(paymentSql, paymentData, function(err, result) {
                if (err) {
                    console.error('Error inserting payment: ', err);
                     return res.status(500).json({ error: 'Internal server
error' });
                }

                res.json({
                    status: 200,
                    error: null,
                    response: "Payment created successfully",
                    payment: {
                        orderId: paymentData.orderId,
                        metode_pembayaran: paymentData.metode_pembayaran,
                        virtual_account: paymentData.virtual_account,
                        status: paymentData.status
                    },
                    payment_id: result.insertId
                });
            });
        });
    });
});
```



POST localhost: 8000/api/pat/payment

Params | Authorization | Headers (8) | Body | Scripts | Settings

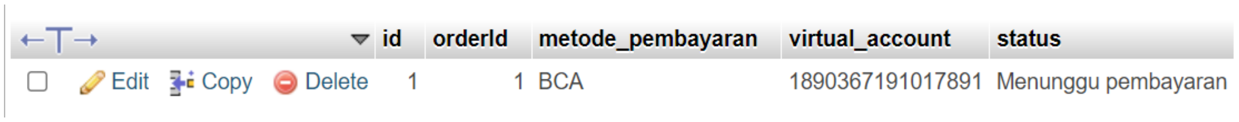
none | form-data | x-www-form-urlencoded | [✓ raw] | binary | GraphQL | JSON

```
{
    "orderId" : 1,
    "metode_pembayaran" : "BCA"
}
```

Body | Cookies | Headers (7) | Test Results — Status: 2

Pretty | Raw | Preview | Visualize | JSON

```
{
    "status": 200,
    "error": null,
    "response": "Payment created successfully",
    "payment": {
        "orderId": 1,
        "metode_pembayaran": "BCA",
        "virtual_account": "1890367191017891",
        "status": "Menunggu pembayaran"
    },
    "payment_id": 1
}
```

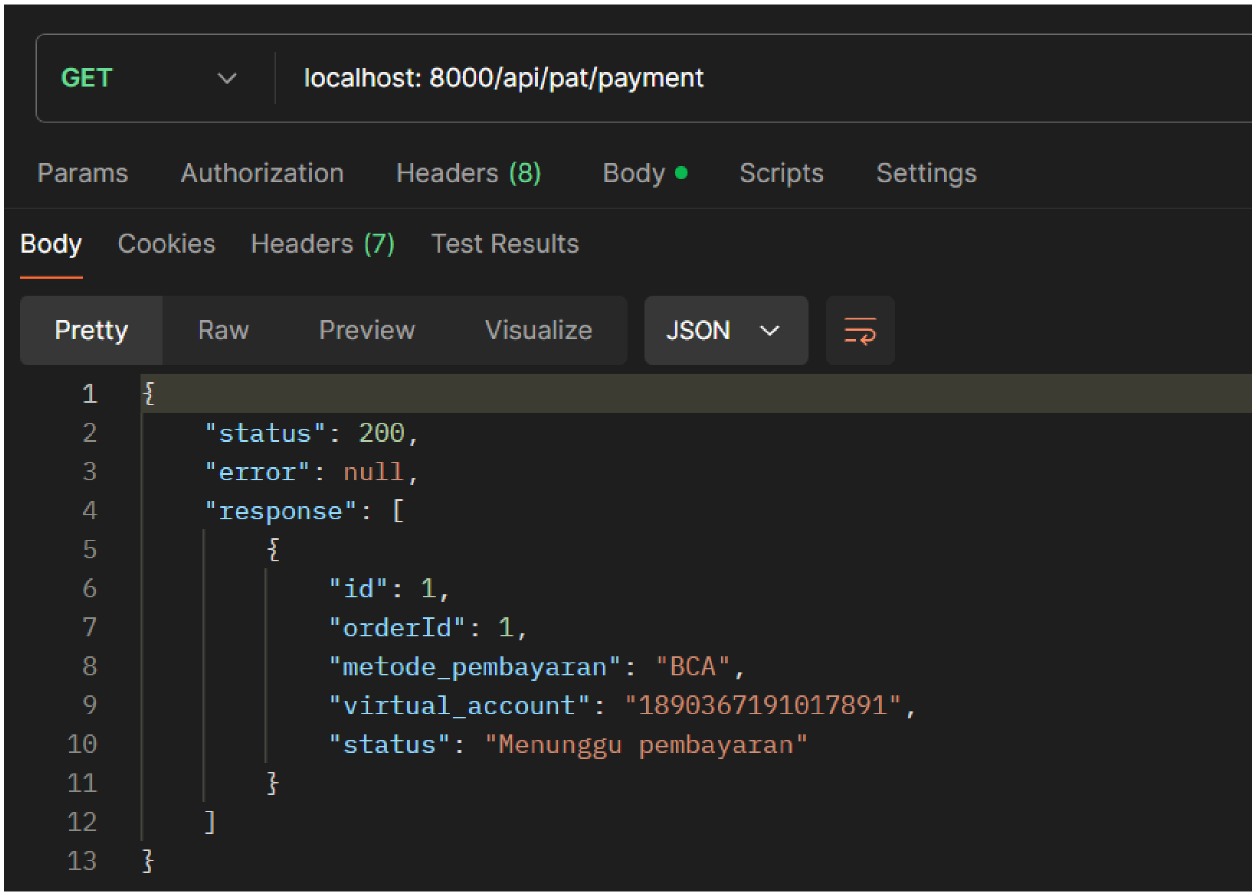


	id	orderId	metode_pembayaran	virtual_account	status
Edit Copy Delete	1	1	BCA	1890367191017891	Menunggu pembayaran

Virtual account terdiri dari 4 angka kode dari bank dan 12 angka yang di generate secara acak.

Retrieve all payment

```
app.get('/api/pat/payment', function(req, res) {
    const sql = "SELECT * FROM payment";

    conn.query(sql, function(err, result) {
        if (err) {
            console.error('Error retrieving payments: ', err);
            res.status(500).json({ error: 'Internal server error' });
            return;
        }

        res.json({
            "status": 200,
            "error": null,
            "response": result
        });
    });
});
```

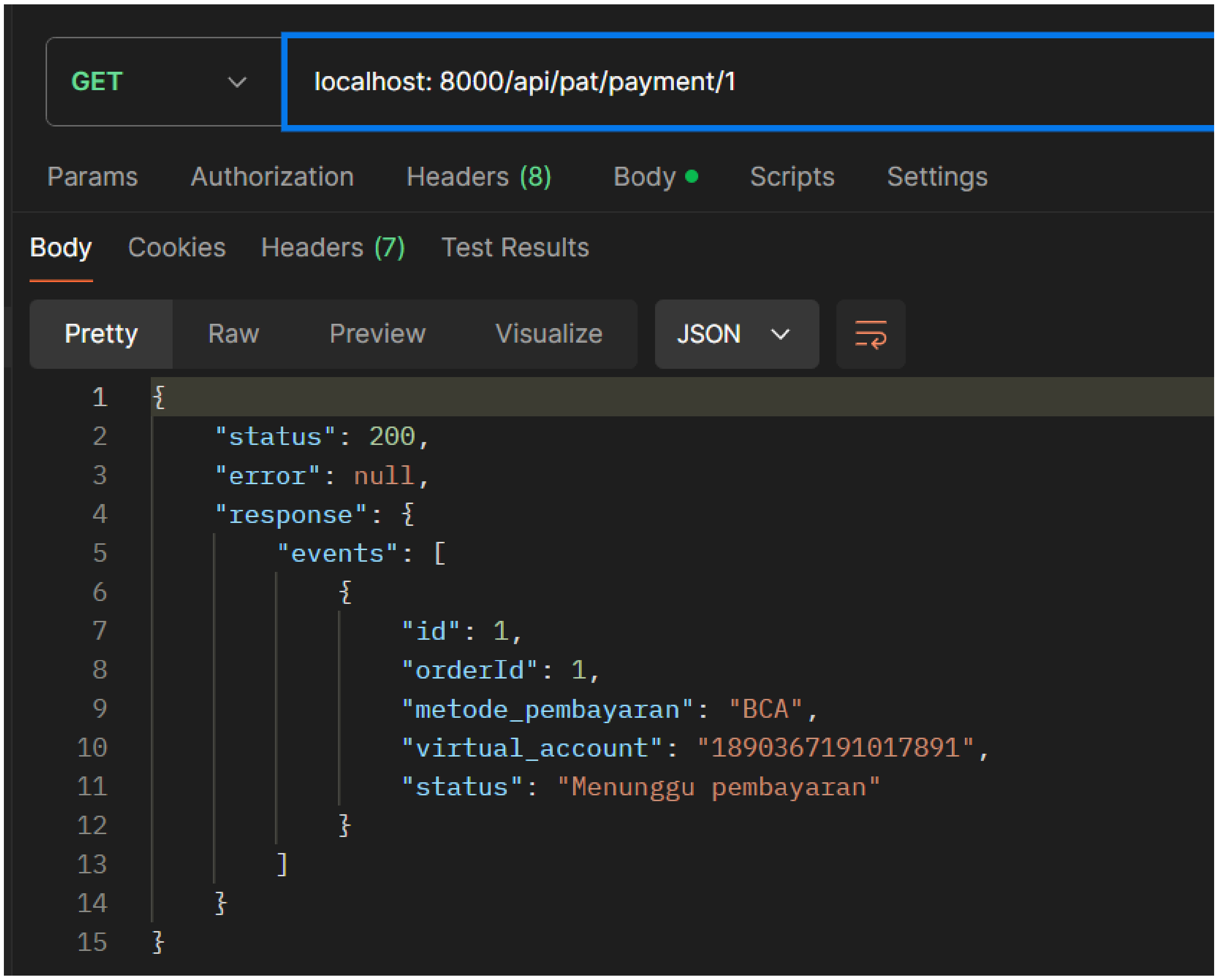


localhost: 8000/api/pat/payment

```
{
    "status": 200,
    "error": null,
    "response": [
        {
            "id": 1,
            "orderId": 1,
            "metode_pembayaran": "BCA",
            "virtual_account": "18903671910178891",
            "status": "Menunggu pembayaran"
        }
    ]
}
```

Retrieve payment by id

```
app.get('/api/pat/payment/:id', function(req, res) {
    const paymentId = req.params.id;

    const paymentSql = "SELECT * FROM payment WHERE id = ?";

    conn.query(paymentSql, [paymentId], function(err, result) {
        if (err) {
            console.error('Error retrieving payment: ', err);
            res.status(500).json({ error: 'Internal server error' });
            return;
        }

        if (result.length === 0) {
            res.status(404).json({ error: 'Payment not found' });
            return;
        }

        res.json({
            "status": 200,
            "error": null,
            "response": {
                events: result.map(payment => {
                    return {
                        id: payment.id,
                        orderId: payment.orderId,
                        metode_pembayaran: payment.metode_pembayaran,
                        virtual_account: payment.virtual_account,
                        status: payment.status
                    };
                })
            }
        });
    });
});
```

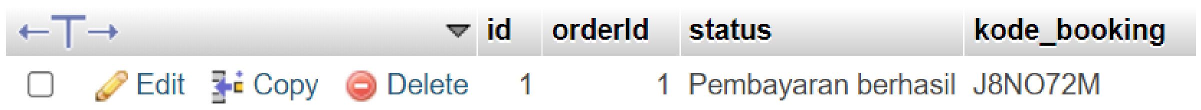


```
localhost: 8000/api/pat/payment/1
```

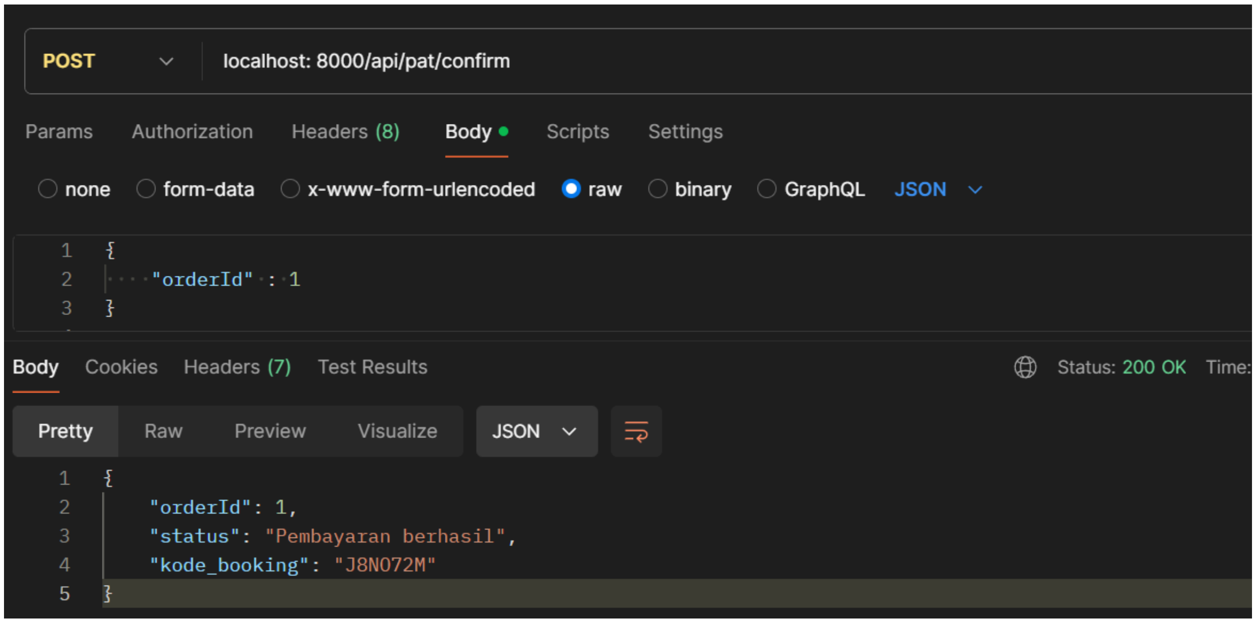
```
{
    "status": 200,
    "error": null,
    "response": {
        "events": [
            {
                "id": 1,
                "orderId": 1,
                "metode_pembayaran": "BCA",
                "virtual_account": "1890367191017891",
                "status": "Menunggu pembayaran"
            }
        ]
    }
}
```

Confirm payment

```
app.post('/api/pat/confirm', function(req, res) {
    console.log('Request body:', req.body); // Logging request body
    const { orderId } = req.body;

    // Pastikan orderId ada dalam request body
    if (!orderId) {
        console.log('Missing orderId'); // Logging error
        return res.status(400).json({ error: 'orderId is required' });
    }

    // Tentukan status dan kode_booking
    const status = "Pembayaran berhasil";
    const kode_booking = generateKodeBooking();
    // Buat objek data yang akan disimpan ke database
    const data = {
        orderId,
        status,
        kode_booking,
    };

    // Query untuk menyimpan data ke database
    const sql = "INSERT INTO confirm SET ?";
    conn.query(sql, data, function(err, result) {
        if (err) {
            console.error('Error inserting confirm: ', err);
              return res.status(500).json({ error: 'Internal server error'
});
        }
        res.json({
            orderId,
            status,
            kode_booking,
        });
    });
});
function generateKodeBooking() {
    const chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
    let kodeBooking = '';
    for (let i = 0; i < 7; i++) {
                 kodeBooking  +=  chars.charAt(Math.floor(Math.random()  *
chars.length));
    }
    return kodeBooking;
}
```



	id	orderId	status	kode_booking
Edit Copy Delete	1	1	Pembayaran berhasil	J8NO72M


POST localhost: 8000/api/pat/confirm

Params | Authorization | Headers (8) | Body | Scripts | Settings

none | form-data | x-www-form-urlencoded | [✓ raw] | binary | GraphQL | JSON

```
{
    "orderId" : 1
}
```

Body | Cookies | Headers (7) | Test Results — Status: 200 OK

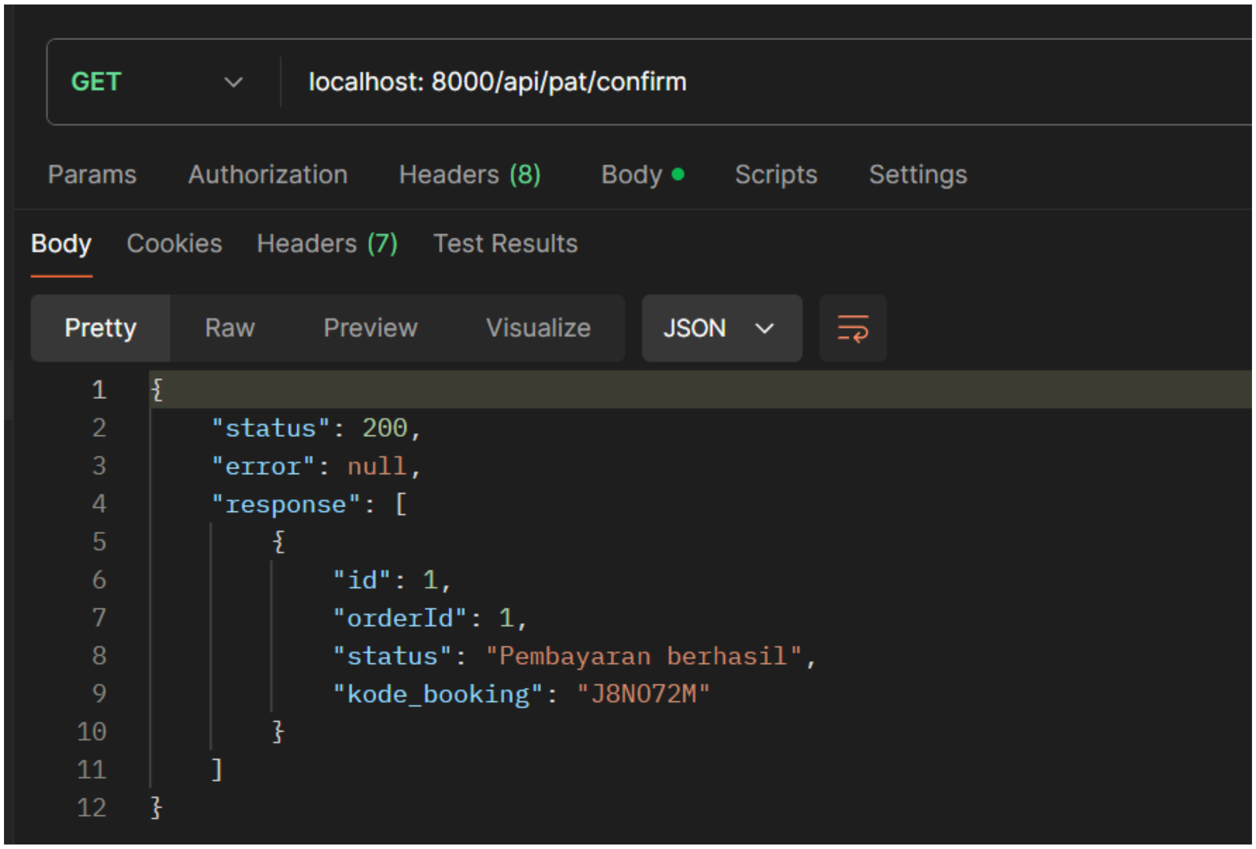
Pretty | Raw | Preview | Visualize | JSON

```
{
    "orderId": 1,
    "status": "Pembayaran berhasil",
    "kode_booking": "J8NO72M"
}
```

Kode booking didapat dari huruf A-Z dan angka 0-9 yang di generate secara acak untuk mendapatkan 6 karakter yang akan digunakan sebagai kode booking.

Retrieve all confirm

```
app.get('/api/pat/confirm', function(req, res) {
    const sql = "SELECT * FROM confirm";

    conn.query(sql, function(err, result) {
        if (err) {
            console.error('Error retrieving confirm: ', err);
            res.status(500).json({ error: 'Internal server error' });
            return;
        }

        res.json({
            "status": 200,
            "error": null,
            "response": result
        });
    });
});
```

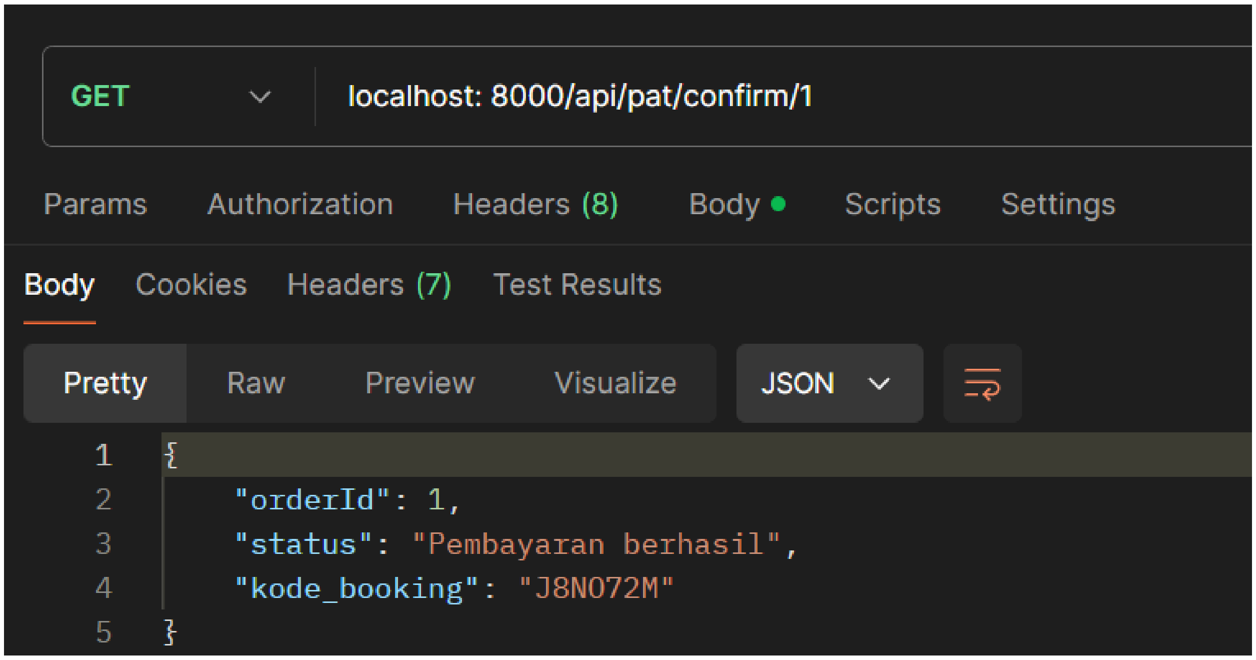


localhost: 8000/api/pat/confirm

```
{
    "status": 200,
    "error": null,
    "response": [
        {
            "id": 1,
            "orderId": 1,
            "status": "Pembayaran berhasil",
            "kode_booking": "J8N072M"
        }
    ]
}
```

Retrieve confirm by id

```
app.get('/api/pat/confirm/:orderId', function(req, res) {
    const orderId = parseInt(req.params.orderId);
    if (!orderId) {
        return res.status(400).json({ error: 'Missing order ID' });
    }

     const sql = "SELECT orderId, status, kode_booking FROM confirm WHERE
orderId = ?";
    conn.query(sql, [orderId], function(err, result) {
        if (err) {
            console.error('Error fetching order: ', err);
              return res.status(500).json({ error: 'Internal server error'
});
        }

        if (result.length === 0) {
            return res.status(404).json({ error: 'Order not found' });
        }

        res.json(result[0]);
    });
});
```



GET localhost: 8000/api/pat/confirm/1

```
{
    "orderId": 1,
    "status": "Pembayaran berhasil",
    "kode_booking": "J8NO72M"
}
```