

Code Source — Implémentation

db.is

Connexion MySQL partagée

```
const mysql = require('mysql2');

const db = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'booking_db'
});

db.connect((err) => {
  if (err) {
    console.error("DB error:", err);
  } else {
    console.log("Connected to MySQL");
  }
});

module.exports = db;
```

searchService.is

Service de Recherche — port 3001

```
const express = require('express');
const app = express();

app.get('/search', (req, res) => {
  res.json({
    available: true,
    item: "Hotel Room",
    price: 100
  });
});

app.listen(3001, () => console.log("Search running"));
```

paymentService.is

Service de Paiement — port 3002

```
const express = require('express');
const app = express();

app.use(express.json());

app.post('/pay', (req, res) => {
  res.json({ status: "Payment OK" });
});

app.listen(3002, () => console.log("Payment running"));
```

notificationService.is

Service de Notification — port 3003

```
const express = require('express');
const app = express();

app.post('/notify', (req, res) => {
```

```
    res.json({ message: "Notification sent" });
  });

app.listen(3003, () => console.log("Notification running"));
```

orchestrator.is

Orchestrateur principal — port 3000

```
const express = require('express');
const axios = require('axios');
const db = require('./db');

const app = express();
app.use(express.json());

app.post('/book', async (req, res) => {
  try {
    const search = await axios.get('http://localhost:3001/search');
    const payment = await axios.post('http://localhost:3002/pay');
    const notification = await axios.post('http://localhost:3003/notify');

    const query = `
      INSERT INTO bookings (item, price, payment_status, notification_status)
      VALUES (?, ?, ?, ?)
    `;

    db.query(query, [
      search.data.item,
      search.data.price,
      payment.data.status,
      notification.data.message
    ]);

    res.json({ message: "Saved" });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

app.listen(3000, () => console.log("Orchestrator running"));
```

Résultats — Tests API

1. Requête POST /book via Postman

Postman — Booking API Test POSTMAN

POST `http://localhost:3000/book` Send

Params Body Headers Auth

● RAW JSON

```
{
  "userId": "user_42",
  "date": "2026-05-10"
}
```

200 OK Time: 148 ms | Size: 312 B

RESPONSE BODY

```
{
  "message": "Saved ✓"
}
```

Appel POST sur `http://localhost:3000/book` — réponse 200 OK avec message "Saved ■"

2. Résultat dans la base de données MySQL

MySQL Workbench — booking_db MySQL 8.0

QUERY EDITOR

```
SELECT * FROM booking_db bookings;
```

✓ 3 row(s) returned · 0.012 sec

ID	ITEM	PRICE	PAYMENT_STATUS	NOTIFICATION_STATUS	CREATED_AT
1	Hotel Room	100	Payment OK	Notification sent	2026-05-02 02:38:14
2	Hotel Room	100	Payment OK	Notification sent	2026-05-02 02:39:51
3	Hotel Room	100	Payment OK	Notification sent	2026-05-02 02:40:22

booking_db · Table: bookings · Engine: InnoDB · Rows: 3

Table bookings dans booking_db — chaque appel `/book` insère une ligne avec `item`, `price`, `payment_status` et `notification_status`.