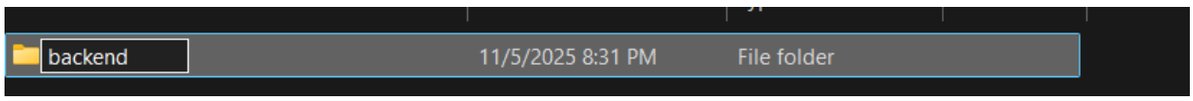
MYSQL + EXPRESS + VUE + NODE.JS

STEP 1 — Project Structure

- Create a parent folder anywhere on your PC.
- Inside, make two subfolders:



Note. Later we will create our frontend folder through vite-vue installation.

STEP 2 — Set up the Backend (Node + Express + MySQL)

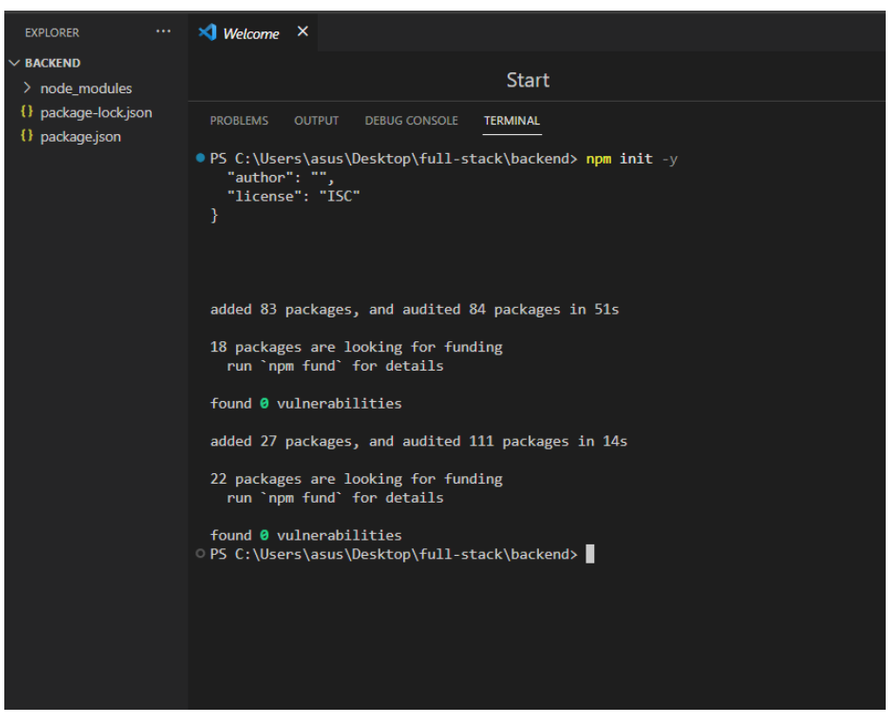
The backend exposes REST API routes:

- GET /api/users → get all users
- POST /api/users → create user
- PUT /api/users/:id → update user
- DELETE /api/users/:id → delete user

2.1 — Initialize Node project

```
PS C:\Users\asus\Desktop\full-stack\backend> npm init -y
>> npm install express cors dotenv mysql2
>> npm install -D nodemon
```

After:



- **express** -creates web server
- **cors** - allows requests from your Vue app

- **dotenv** - reads config from .env
- **mysql2** - connects to MySQL
- **nodemon** - restarts automatically when code changes (for development)

2.2 — Database connection file

Create db.js file.

```
const mysql = require('mysql2/promise');
require('dotenv').config();

const pool = mysql.createPool({
  host: process.env.DB_HOST,
  user: process.env.DB_USER,
  password: process.env.DB_PASSWORD,
  database: process.env.DB_NAME,
  port: process.env.DB_PORT,
  waitForConnections: true,
  connectionLimit: 10
});

module.exports = pool;
```

Explanation:

const mysql = require('mysql2/promise');

- This line imports the mysql2 library, which is a faster and more modern version of the original mysql library.
- The /promise version allows you to use async/await syntax instead of callbacks, making code cleaner and easier to manage.

require('dotenv').config();

- This loads your .env file so that environment variables (like DB username, password, etc.) are available via process.env.

const pool = mysql.createPool({
host: process.env.DB_HOST,
user: process.env.DB_USER,
password: process.env.DB_PASSWORD,
database: process.env.DB_NAME,
port: process.env.DB_PORT,
waitForConnections: true,
connectionLimit: 10
});

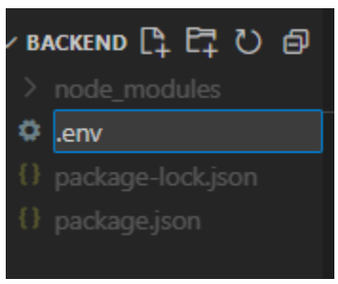
- mysql.createPool() creates a pool of database connections instead of just one connection.
- This means Node.js can reuse connections efficiently for multiple database queries.
- It's better for performance and scalability.

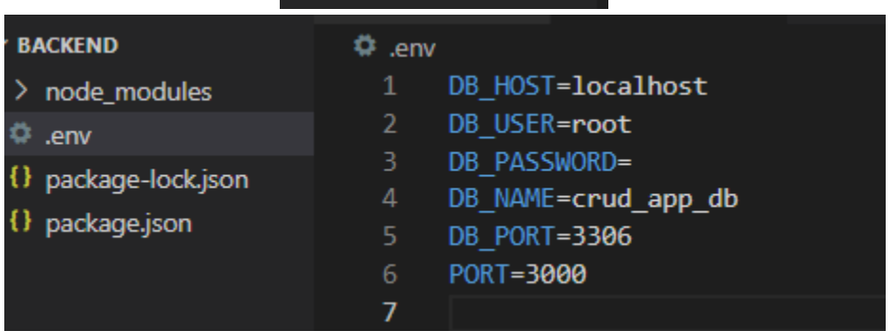
- The options:
    - host: Database server address (usually localhost or an IP).
    - user: MySQL username.
    - password: MySQL password.
    - database: Database name you want to connect to.
    - port: Port number (default for MySQL is 3306).
    - waitForConnections: When all connections are in use, new requests will wait until one is free.
    - connectionLimit: Maximum number of simultaneous connections (10 in this case).

module.exports = pool;

- This exports the pool so that other files in your project can use the same connection.

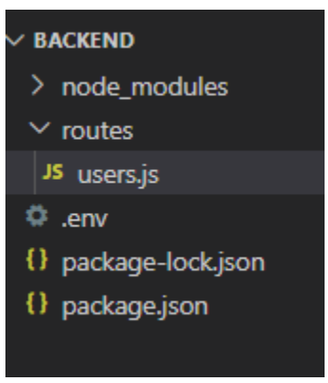
2.3 — Environment configuration





Why use .env: Keeps credentials separate so you can change them without editing code.

2.4 — Create REST API routes



Inside the backend folder, create a new folder and add a users.js file to contain all CRUD logic for organized code under api/users.

```
const express = require('express');
const router = express.Router();
const pool = require('../db');

// GET all users
router.get('/', async (req, res, next) => {
 try {
  const [rows] = await pool.execute('SELECT * FROM users ORDER BY id DESC');
  res.json(rows);
 } catch (err) {
  next(err);
 }
});

// POST create
router.post('/', async (req, res, next) => {
 try {
  const { name, email, age } = req.body;
  const [result] = await pool.execute(
   'INSERT INTO users (name, email, age) VALUES (?, ?, ?)',
   [name, email, age || null]
  );
  const [rows] = await pool.execute('SELECT * FROM users WHERE id=?',
[result.insertId]);
  res.status(201).json(rows[0]);
 } catch (err) {
  next(err);
 }
});

// PUT update
router.put('/:id', async (req, res, next) => {
 try {
  const { name, email, age } = req.body;
  await pool.execute(
   'UPDATE users SET name=?, email=?, age=? WHERE id=?',
   [name, email, age || null, req.params.id]
  );
  const [rows] = await pool.execute('SELECT * FROM users WHERE id=?',
[req.params.id]);
  res.json(rows[0]);
 } catch (err) {
  next(err);
 }
});
```

```
// DELETE
router.delete('/:id', async (req, res, next) => {
 try {
  await pool.execute('DELETE FROM users WHERE id=?', [req.params.id]);
  res.json({ message: 'User deleted' });
 } catch (err) {
  next(err);
 }
});

module.exports = router;
```

Explanation:

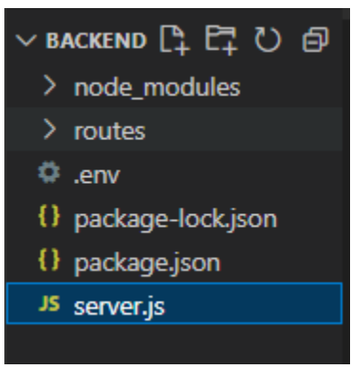
const express = require('express');
const router = express.Router();
const pool = require('../db');

- express.Router() - lets you define routes separately from the main server.
- pool - MySQL connection pool from db.js, used for executing queries.

Final API endpoints

- **GET /api/users**
- **POST /api/users**
- **PUT /api/users/:id**
- **DELETE /api/users/:id**

2.5 — Main server file
Create a server.js file under backend folder.




```
require('dotenv').config();
const express = require('express');
const cors = require('cors');
const usersRouter = require('./routes/users');

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

app.get('/', (req, res) => res.send({ message: 'API running...' }));
app.use('/api/users', usersRouter);

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => console.log(` Backend running at
http://localhost:${PORT}` ));
```

Explanation:

const express = require('express');
const cors = require('cors');
const usersRouter = require('./routes/users');

- express - the main web framework for building APIs.
- cors - allows your frontend (Vue) and backend (Node) to communicate, even though they run on different ports (5173 & 3000).
- usersRouter - imports all CRUD routes from routes/users.js.

const app = express();

- This creates an Express application instance — basically your API server.

app.use(cors());
app.use(express.json());

- cors() - enables Cross-Origin Resource Sharing. Without it, your browser would block API calls from Vue (different port).
- express.json() -tells Express to automatically parse JSON bodies in requests (like when Vue sends POST or PUT**).**

app.get('/', (req, res) => res.send({ message: 'API running...' }));
app.use('/api/users', usersRouter);

- Define routes

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => console.log(` ✅ Backend running at http://localhost:${PORT}`));

- Reads the port from .env, or uses 3000 by default.
- app.listen() starts the web server and listens for incoming requests.
- Logs a message to the console when the server is up and running.

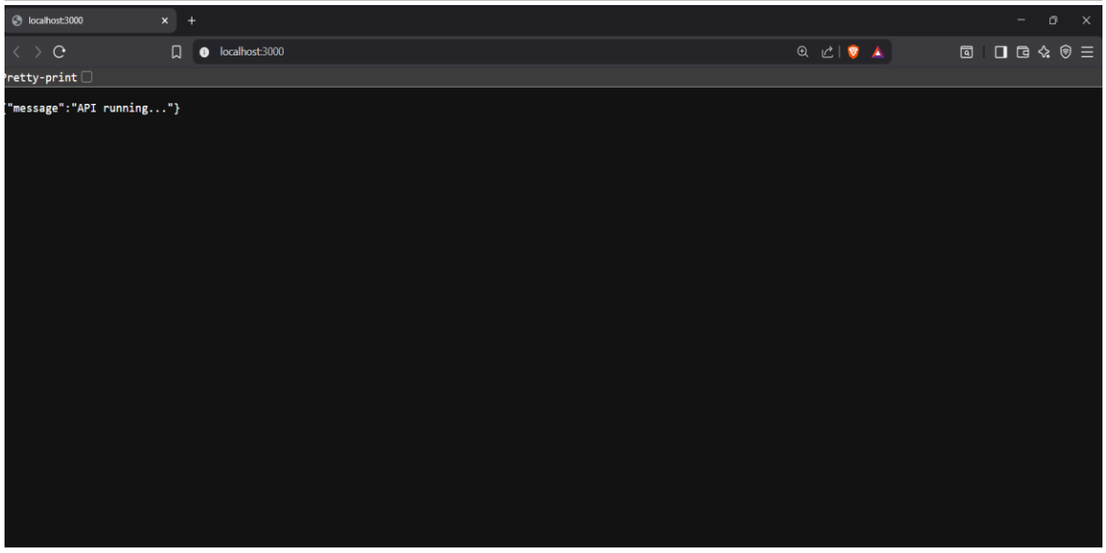
2.6 — package.json scripts

Before:

Updated:

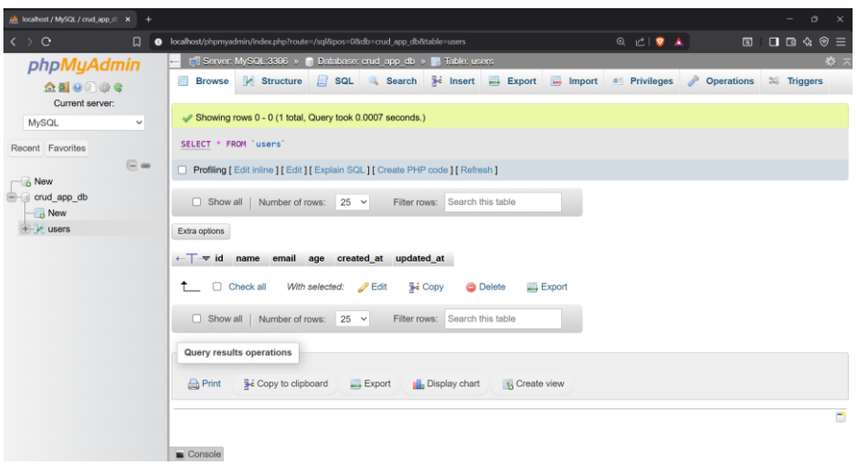
Testing:

- Type "**npm run dev**" to test your backend.



2.7 — Create MySQL database

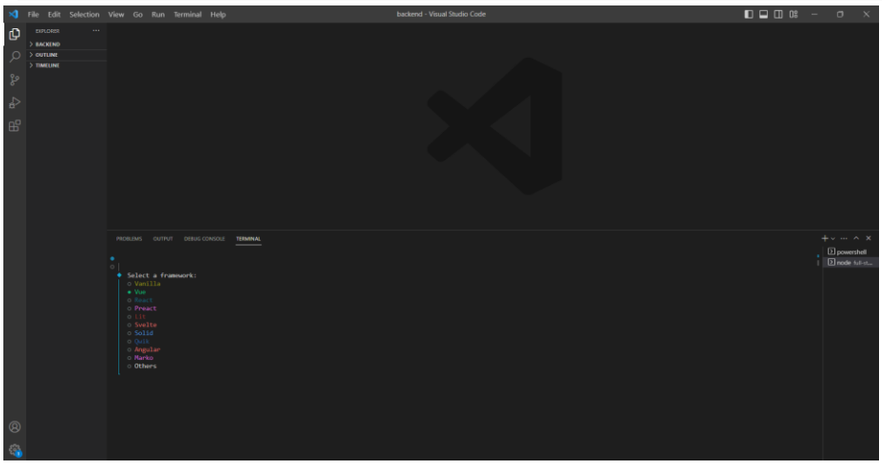
```
CREATE DATABASE crud_app_db;
USE crud_app_db;

CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100),
  email VARCHAR(150) UNIQUE,
  age INT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

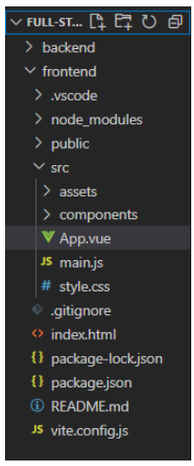


STEP 3 — Frontend (Vue 3 + Vite)

- Install Vue using vite.

```
PS C:\Users\asus\Desktop\full-stack\backend> cd ..
PS C:\Users\asus\Desktop\full-stack> npm create vite@latest frontend
```





- Then install dependencies.

```
PS C:\Users\asus\Desktop\full-stack\frontend> npm install
>> npm install axios
```

3.2 — Configure Vite for backend proxy

vite.config.js

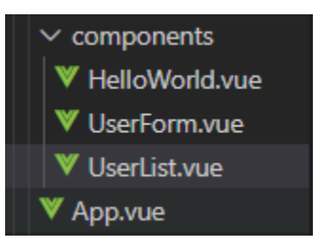
```
frontend > vite.config.js > ...
1   import { defineConfig } from 'vite'
2   import vue from '@vitejs/plugin-vue'
3
4   // https://vite.dev/config/
5   export default defineConfig({
6     plugins: [vue()],
7     server: {
8       proxy: { '/api': 'http://localhost:3000' }
9     }
10  })
11
```

Add that line in vite.config.js file so Vue dev server automatically forwards /api/... requests to Express.

3.3 — Add environment variable

```
frontend > .env
1   VITE_API_BASE=http://localhost:3000/api
```

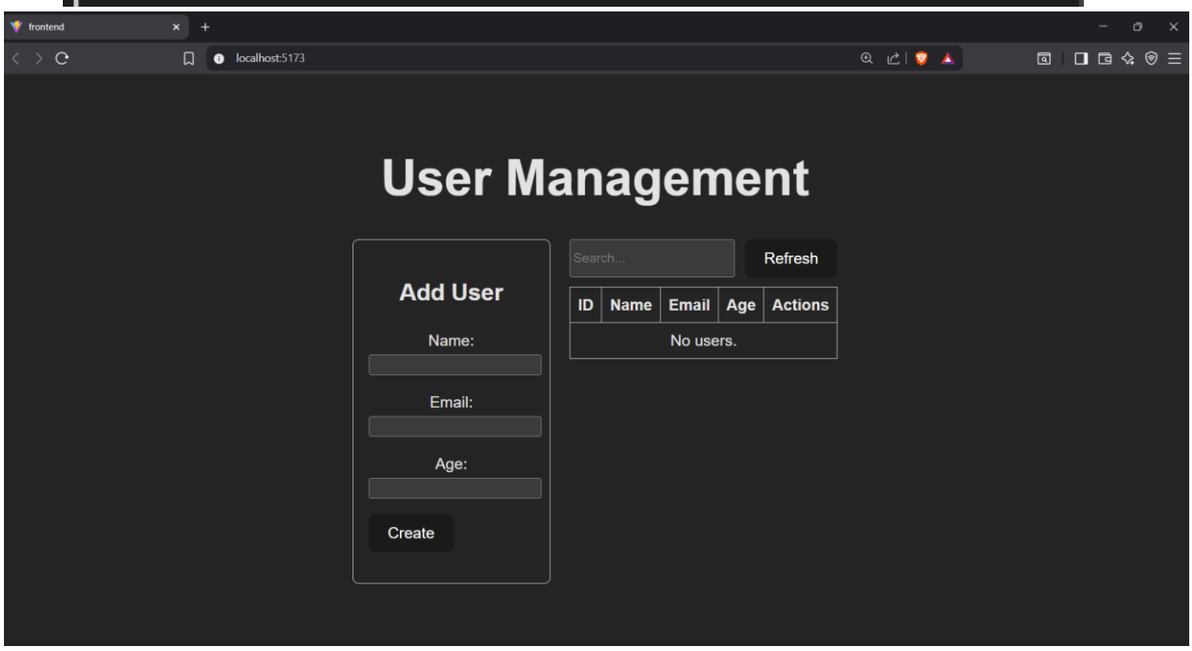
3.4 – Create your components.

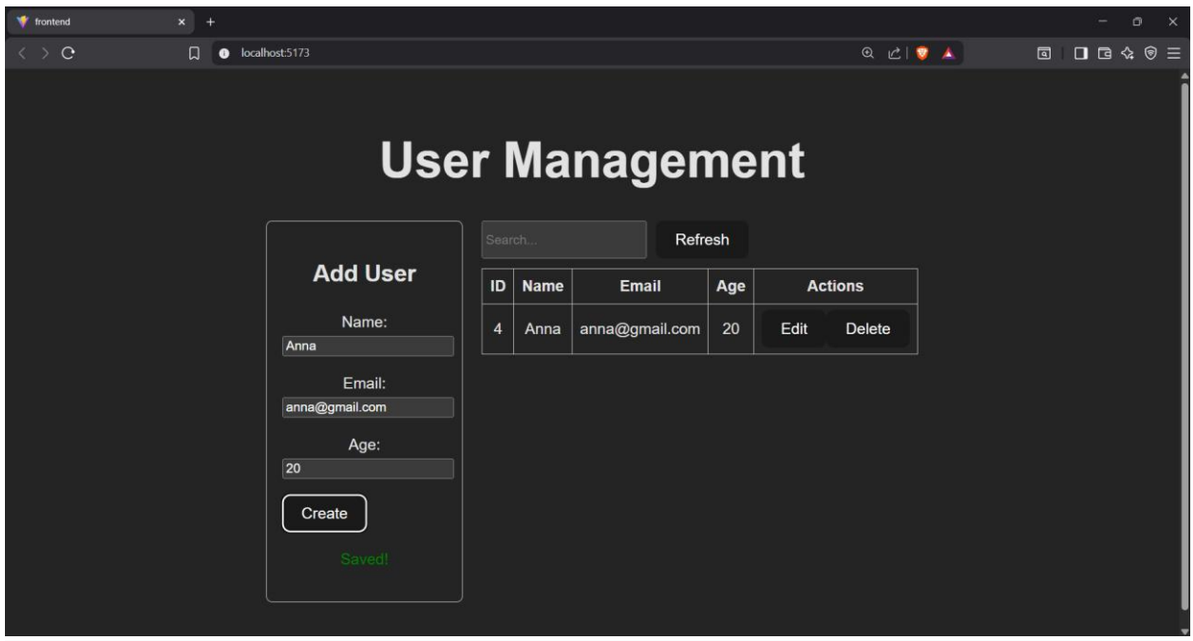


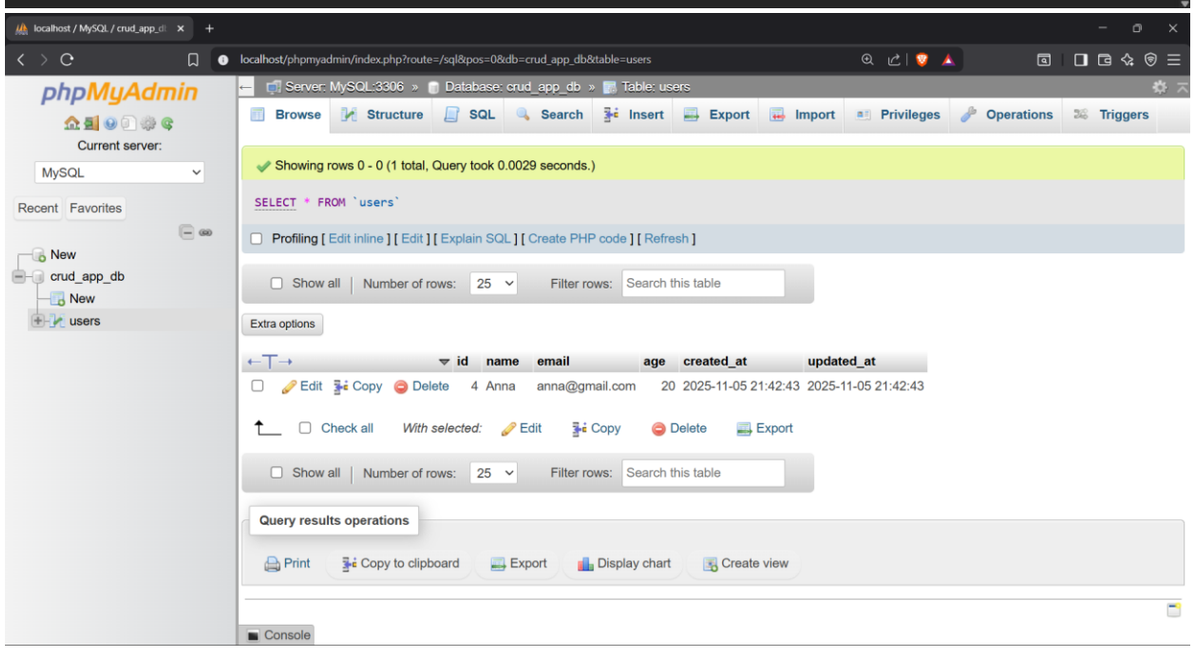
STEP 4 — Combine Frontend & Backend

- At the project root (full-stack/), create this file:

- Install and run:




frontend
localhost:5173
User Management
Add User
Name:
Anna
Email:
anna@gmail.com
Age:
20
Create
Saved!
Search...
Refresh
ID
Name
Email
Age
Actions
4
Anna
anna@gmail.com
20
Edit
Delete


localhost / MySQL / crud_app_db
localhost/phpmyadmin/index.php?route=/sql&pos=0&db=crud_app_db&table=users
phpMyAdmin
Current server:
MySQL
Recent
Favorites
New
crud_app_db
New
users
Server: MySQL:3306 » Database: crud_app_db » Table: users
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Triggers
Showing rows 0 - 0 (1 total, Query took 0.0029 seconds.)
SELECT * FROM `users`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows:
25
Filter rows:
Search this table
Extra options
id
name
email
age
created_at
updated_at
Edit
Copy
Delete
4
Anna
anna@gmail.com
20
2025-11-05 21:42:43
2025-11-05 21:42:43
Check all
With selected:
Edit
Copy
Delete
Export
Query results operations
Print
Copy to clipboard
Export
Display chart
Create view
Console