Perfect 👌 Let's dive into **8. RESTful APIs with Express.js**.
Since you already know Laravel, I'll explain with **Laravel → Express mappings** so it clicks quickly.

---

8. RESTful APIs with Express.js

📌 What is REST?

- **REST (Representational State Transfer)** = API design style that uses **HTTP methods** (GET, POST, PUT, DELETE) to manage resources.
- **Resource-based URLs**:
  - `/users` → List/Create users
  - `/users/:id` → Get/Update/Delete a single user

☑ Same as Laravel `Route::resource('users', UserController::class);`

📌 Express.js REST API Flow

1. **Install Express**

```
npm install express
```

2. **Basic Server**

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

app.use(express.json()); // For parsing JSON requests

app.listen(3000, () => {
  console.log("Server running on http://localhost:3000");
});
```

📌 CRUD Routes Example (Users API)

```
// Get all users
app.get('/users', (req, res) => {
  res.json([{ id: 1, name: "Alice" }]);
});

// Get single user
```

```
app.get('/users/:id', (req, res) => {
  res.json({ id: req.params.id, name: "Alice" });
});

// Create new user
app.post('/users', (req, res) => {
  const user = req.body;
  res.status(201).json({ message: "User created", user });
});

// Update user
app.put('/users/:id', (req, res) => {
  res.json({ message: "User updated", id: req.params.id, data: req.body });
});

// Delete user
app.delete('/users/:id', (req, res) => {
  res.json({ message: "User deleted", id: req.params.id });
});
```

👉 In Laravel this is equal to:

```
Route::resource('users', UserController::class);
```

📌 Router & Controller Structure (Laravel-like)

Instead of writing routes in `server.js`, you can structure like Laravel.

routes/userRoutes.js

```
const express = require('express');
const router = express.Router();
const userController = require('../controllers/userController');

router.get('/', userController.getUsers);
router.get('/:id', userController.getUser);
router.post('/', userController.createUser);
router.put('/:id', userController.updateUser);
router.delete('/:id', userController.deleteUser);

module.exports = router;
```

controllers/userController.js

```
exports.getUsers = (req, res) => {
  res.json([{ id: 1, name: "Alice" }]);
};

exports.getUser = (req, res) => {
  res.json({ id: req.params.id, name: "Alice" });
};

exports.createUser = (req, res) => {
  res.status(201).json({ message: "User created", data: req.body });
};

exports.updateUser = (req, res) => {
  res.json({ message: "User updated", id: req.params.id, data: req.body });
};

exports.deleteUser = (req, res) => {
  res.json({ message: "User deleted", id: req.params.id });
};
```

server.js

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

app.use(express.json());

const userRoutes = require('./routes/userRoutes');
app.use('/users', userRoutes);

app.listen(3000, () => console.log("Server running on port 3000"));
```

☑ Now it looks like Laravel's `routes/web.php + controllers/UserController.php`.

📌 Middleware in Express (Like Laravel Middleware)

```
// Custom middleware
function logger(req, res, next) {
  console.log(`${req.method} ${req.url}`);
  next();
}

app.use(logger);
```

Laravel equivalent:

```php
public function handle($request, Closure $next) {
    Log::info($request->method().' '.$request->path());
    return $next($request);
}
```

📌 Error Handling (Global Middleware)

```javascript
app.use((err, req, res, next) => {
  console.error(err.stack);
  res.status(500).json({ error: "Something broke!" });
});
```

📌 Testing API with Postman

- **GET /users** → list all users
- **POST /users** → create user (send JSON body)
- **PUT /users/:id** → update
- **DELETE /users/:id** → remove

📌 Interview-Focused Q&A

Q1: What is REST in Node.js?
→ REST is an API style where resources are managed using HTTP methods.

Q2: How is Express.js different from Laravel?

- Laravel → batteries included (ORM, routing, validation, middleware).
- Express → minimal, you add packages as needed.

Q3: How do you handle middleware in Express?
→ With `app.use()` or route-level middleware.

Q4: How do you return JSON in Express?
→ Using `res.json(data)` (similar to Laravel's `return response()->json($data)`).

☑ **Key Takeaway**:
Express makes REST API building simple, but unlike Laravel, **you have to bring your own ORM, validation, authentication**.

👉 Next we can go into **9. Authentication & Security in Node.js** (JWT, bcrypt, middleware auth like Laravel guards).

Do you want me to cover **JWT-based auth** (most common in Node.js) or also include **session-based auth** like Laravel's default?