Copilot

Gerne! Hier ist eine vollständige Lösung für eine Kochrezepte-Anwendung mit Vite/React im Frontend und Express im Backend. Wir werden das MVC-Prinzip (Model-View-Controller) anwenden.

Schritt 1: Projektstruktur erstellen

Erstelle ein neues Verzeichnis für dein Projekt und navigiere in dieses Verzeichnis. Erstelle dann die folgende Ordnerstruktur:

```
kochrezepte-app/
├── backend/
│   ├── controllers/
│   ├── models/
│   ├── routes/
│   ├── config/
│   ├── server.js
├── frontend/
│   ├── public/
│   │   └── index.html
│   ├── src/
│       ├── components/
│       ├── App.jsx
│       ├── index.jsx
├── package.json
```

Schritt 2: Backend einrichten

Navigiere in das `backend` Verzeichnis und initialisiere ein neues Node.js-Projekt:

```
cd backend
npm init -y
npm install express mongoose cors
```

Erstelle die Datei `server.js`:

```
// backend/server.js
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const recipeRoutes = require('./routes/recipeRoutes');

const app = express();
const PORT = process.env.PORT || 5000;

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

mongoose.connect('mongodb://localhost:27017/kochrezepte', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});

app.use('/api/recipes', recipeRoutes);

app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

Erstelle das Modell für die Rezepte:

```
// backend/models/Recipe.js
const mongoose = require('mongoose');

const recipeSchema = new mongoose.Schema({
  title: String,
  ingredients: [String],
  instructions: String,
});

module.exports = mongoose.model('Recipe', recipeSchema);
```

Erstelle den Controller:

```
// backend/controllers/recipeController.js
const Recipe = require('../models/Recipe');

exports.getAllRecipes = async (req, res) => {
  try {
    const recipes = await Recipe.find();
    res.json(recipes);
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
};

exports.createRecipe = async (req, res) => {
  const recipe = new Recipe(req.body);
```

```
  try {
    const newRecipe = await recipe.save();
    res.status(201).json(newRecipe);
  } catch (error) {
    res.status(400).json({ message: error.message });
  }
};
```

Erstelle die Routen:

```
// backend/routes/recipeRoutes.js
const express = require('express');
const { getAllRecipes, createRecipe } = require('../controllers/recipeController');

const router = express.Router();

router.get('/', getAllRecipes);
router.post('/', createRecipe);

module.exports = router;
```

Schritt 3: Frontend einrichten

Navigiere in das `frontend` Verzeichnis und initialisiere ein neues Vite/React-Projekt:

```
cd frontend
npm init vite@latest
# Wähle React und JavaScript
npm install
```

Erstelle die Datei `src/App.jsx`:

```
// frontend/src/App.jsx
import React, { useState, useEffect } from 'react';
import axios from 'axios';

const App = () => {
  const [recipes, setRecipes] = useState([]);
  const [title, setTitle] = useState('');
  const [ingredients, setIngredients] = useState('');
  const [instructions, setInstructions] = useState('');

  useEffect(() => {
    const fetchRecipes = async () => {
      const response = await axios.get('http://localhost:5000/api/recipes');
      setRecipes(response.data);
    };
    fetchRecipes();
  }, []);

  const handleSubmit = async (e) => {
    e.preventDefault();
    const newRecipe = { title, ingredients: ingredients.split(','), instructions };
    await axios.post('http://localhost:5000/api/recipes', newRecipe);
    setRecipes([...recipes, newRecipe]);
  };

  return (
    <div>
      <h1>Kochrezepte</h1>
      <form onSubmit={handleSubmit}>
        <input type="text" value={title} onChange={(e) => setTitle(e.target.value)} placeholder="Titel" />
        <input type="text" value={ingredients} onChange={(e) => setIngredients(e.target.value)} placeholder="Zutaten (kommagetrennt)" />
        <textarea value={instructions} onChange={(e) => setInstructions(e.target.value)} placeholder="Anleitung"></textarea>
        <button type="submit">Rezept hinzufügen</button>
      </form>
      <ul>
        {recipes.map((recipe, index) => (
          <li key={index}>
            <h2>{recipe.title}</h2>
            <p>{recipe.ingredients.join(', ')}</p>
            <p>{recipe.instructions}</p>
          </li>
        ))}
      </ul>
    </div>
  );
};

export default App;
```

Erstelle die Datei `src/index.jsx`:

```
// frontend/src/index.jsx
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';

ReactDOM.render(<App />, document.getElementById('root'));
```

Erstelle die Datei `public/index.html`:

```html
<!-- frontend/public/index.html -->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Kochrezepte</title>
</head>
<body>
  <div id="root"></div>
  <script type="module" src="/src/index.jsx"></script>
</body>
</html>
```

Schritt 4: Vite konfigurieren

Erstelle die Datei `vite.config.js` im `frontend` Verzeichnis:

```
// frontend/vite.config.js
import { defineConfig } from 'vite';
import react from '@vitejs/plugin-react';

export default defineConfig({
  plugins: [react()],
  server: {
    proxy: {
      '/api': {
        target: 'http://localhost:5000',
        changeOrigin: true,
        rewrite: (path) => path.replace(/^\/api/, ''),
      },
    },
  },
});
```

Schritt 5: Anwendung starten

Starte das Backend:

```
cd backend
node server.js
```

Starte das Frontend:

```
cd frontend
npm run dev
```

Jetzt solltest du eine vollständige Kochrezepte-Anwendung haben, die nach dem MVC-Prinzip aufgebaut ist und sowohl ein Frontend mit Vite/React als auch ein Backend mit Express verwendet. Viel Erfolg! 🍳

Hast du noch Fragen oder benötigst du weitere Anpassungen?