Todo-App – Kurz-Dokumentation (Logischer Ablauf)

1. Datenbasis

Die Aufgaben werden in einer JSON-Datei gespeichert. Wichtige Felder: Listen-ID, Name, Todos.

```
{ 'id': 32, 'name': '', 'todos': [ { 'id': 12, 'checked': false, 'priori
ty': 'HIGH', 'dueDate': '2026-01-20' } ] }
```

2. Datenservice (JsonService)

Der JsonService lädt die Datei, stellt sie im Speicher bereit und übernimmt Änderungen (hinzufügen, aktualisieren, löschen, abhaken).

Load() → Datei lesen und als JSON in den Speicher laden

AddTodo(title, priority, dueDate) → neues Todo mit automatisch erhöhter ID

SaveUnlocked() → Änderungen zuverlässig in die Datei schreiben

3. Serverstart & Annahme von Anfragen

Der Socket-Server liest die Konfiguration und lauscht auf dem lokalen Port.

```
IP=127.0.0.1 | Port=8080
```

Logischer Ablauf beim Server:

1) Start → Socket öffnen

2) Warten → HTTP-Anfrage entgegennehmen

3) Route erkennen → anhand des angefragten Pfads

4) Aktion ausführen → ggf. JsonService nutzen

5) Antwort zurücksenden → Text oder JSON

```
Beispiel Route: GET /lists → Antwort: '/lists- Route!'
Beispiel Route: GET /lists/create → Antwort: '/lists/create- Route!'
```

4. React-Frontend – Anfrage & Anzeige

Das Frontend ruft den Server über fetch() auf und zeigt die Antwort an.

```
fetch('http://127.0.0.1:8080/lists').then(r => r.text()).then(show)
```

Bei Klick auf 'Liste erstellen' wird die zweite Route aufgerufen und das Ergebnis angezeigt.

```
fetch('http://127.0.0.1:8080/lists/create').then(r => r.text()).then(sho
w)
```

5. Gesamtprozess in wenigen Schritten

• Benutzer öffnet das React-Frontend

• Frontend sendet eine Anfrage (GET /lists)

• Server erkennt die Route und liefert die Antwort

• Frontend zeigt den erhaltenen Text an

• Bei Bedarf wird eine Aktion ausgelöst (z. B. Liste erstellen)

Diese Kurz-Doku beschreibt den logischen Weg durch den Prozess – klar, knapp und auf Basis des vorhandenen Codes.