

Applikationsflöde (beskrivning steg för steg)

// 📦 1. App startar

När `App.jsx` körs i React:

- `gridSize` sätts till 1
 - `gridData` är tom (`[]`)
 - `useEffect()` anropas direkt → som i sin tur kör `loadGridFromApi()`
→ Hämtar eventuell tidigare sparad grid från API (backend/JSON)
-

🧠 2. Funktioner som definieras i `App.jsx`

🎨 `getNextColor(prev)`

- Returnerar en ny slumpmässig färg som **inte är samma som föregående**
- Används för att skapa variation i rutorna

💾 `saveGridToApi(grid)`

- Skickar hela `gridData` till backend som **POST**-request till API:t

🔄 `loadGridFromApi()`

- Hämtar `gridData` från API (via **GET**)
- Uppdaterar både `gridData` och `gridSize` baserat på innehållet i `grid.json`

+ `addSquare()`

- Räknar ut nästa lediga `[row, col]` via `generateQuarterSpiral()`

- Genererar färg
- Uppdaterar en **kopierad** version av `gridData` (`spread ...`)
- Skickar uppdaterad data till API via `saveGridToApi()`
- Om lyckat → uppdaterar React state





`clearGrid()`

- Återställer `gridData` till tom lista
- Återställer `gridSize` till 1
- Sparar det direkt till backend (API)

3. UI (return JSX)

- `<button onClick={addSquare}>` → Lägger till ny ruta
- `<button onClick={clearGrid}>` → Rensar rutnätet
- `<div className="grid">` → Renderar ett `gridSize x gridSize` rutnät
 - Varje cell kollar om det finns `{ row, col, color }` i `gridData`
 - Om ja: sätter bakgrundsfärg
 - Om nej: visar tom cell med streckad kant

4. Backendstruktur (.NET Web API)

-  **Controller:**
`GridController.cs` hanterar inkommande `GET` och `POST` till `/api/grid`
 -  **Service-layer:**
`GridService.cs` ansvarar för affärslogik och kallar på repository
 -  **Repository-layer:**
`GridRepository.cs` hanterar själva filinläsning och skrivning av `JsonData/grid.json`
(Skapar mappen + fil automatiskt vid behov)
 -  **CORS** är aktiverat i `Program.cs`:
`WithOrigins("http://localhost:5173")` så att React får göra anrop
-

Sammanfattning

- Frontend använder React `useState` + `useEffect` + `fetch`
- Backend är uppdelad enligt **Repository + Service + Controller**-mönster (SOLID-inspirerat)
- Grid byggs lager för lager enligt `generateQuarterSpiral()`
- Alla klick hanteras, sparas och laddas korrekt via API
- JSON-filen är den **permanenta persistenta källan** i `JsonData/grid.json`