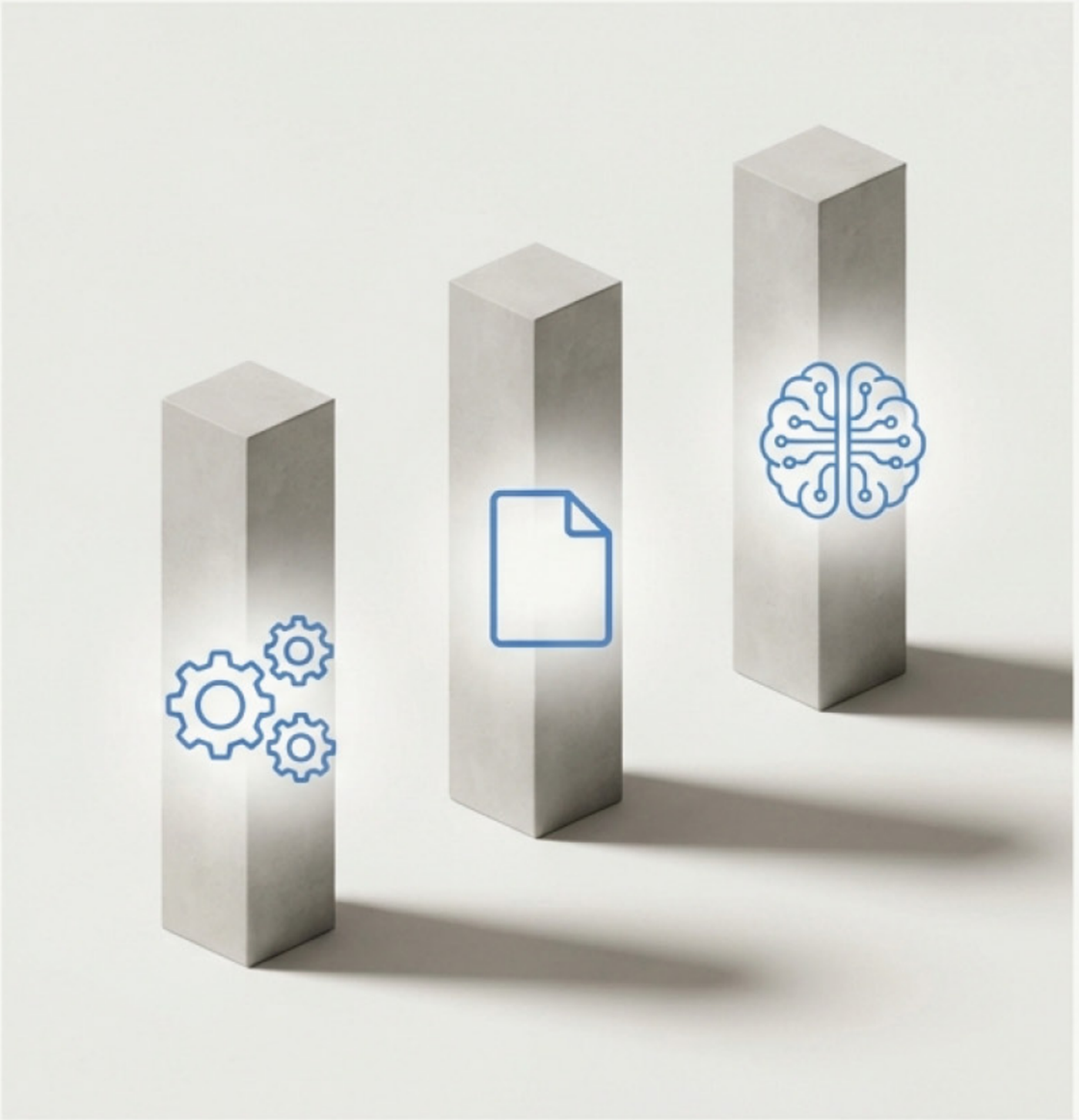
Construirea unui DataGrid Reactiv: Ghid Pas cu Pas

Pasul 1: Configurarea Inițială și Căutarea Reactivă

Vom explora arhitectura fundamentală a unui tabel de date interactiv, concentrându-ne pe cele trei componente esențiale: mediul tehnic, structura datelor și logica de interacțiune.



Obiectiv: O Interfață Funcțională pentru Vizualizarea Datelor

Scopul acestui pas este crearea unei interfețe complete pentru afișarea datelor, bazată pe sinergia dintre structura declarativă React și manipularea elementelor DOM. Vom acoperi:

1. **Mediul de Execuție**: Instalarea și configurarea dependențelor esențiale (Material UI).

2. **Structura Datelor**: Definirea unui model de date robust și unic identificabil.

3. **Logica de Interacțiune**: Implementarea unui mecanism de căutare manuală, controlat și reactiv.

Pilonul 1: Mediul Tehnic și Dependențele

Pentru a construi o componentă complexă precum DataGrid, ne bazăm pe biblioteci externe specializate. Acestea oferă funcționalități avansate *out-of-the-box*, accelerând dezvoltarea.

- **@mui/material**: Fundația. Furnizează componentele de bază pentru UI și sistemul de stilizare (bazat pe Emotion).
- **@mui/x-data-grid**: Piesa centrală. Oferă componenta `DataGrid` cu suport nativ pentru sortare, paginare și filtrare.

```
# Instalarea prin Node Package Manager (npm)
npm install @mui/material @emotion/react @emotion/styled
npm install @mui/x-data-grid
```

Pilonul 2: Blueprint-ul Datelor

Orice componentă de afișare este la fel de bună ca datele pe care le primește. O structură clară și corect identificată este non-negociabilă.

> **Cerința Fundamentală: Identificatorul Unic (`id`)**
>
> Fiecare obiect din colecție TREBUIE să posede un câmp `id` unic. Acesta este esențial pentru ca React să gestioneze eficient reconcilierea (*diffing*) și să prevină anti-pattern-ul folosirii indexului ca cheie.

```
// Modelul de resursă: O listă de obiecte JavaScript
const rows = [
  { id: 1, title: 'Arta Razboiului', author: 'Sun Tzu', pages: 260 },
  { id: 2, title: 'Meditatii', author: 'Marcus Aurelius', pages: 304 },
  { id: 3, title: 'Ghidul Autostopistului Galactic', author: 'Douglas Adams', pages: 224 },
  // ...alte înregistrări
];
```

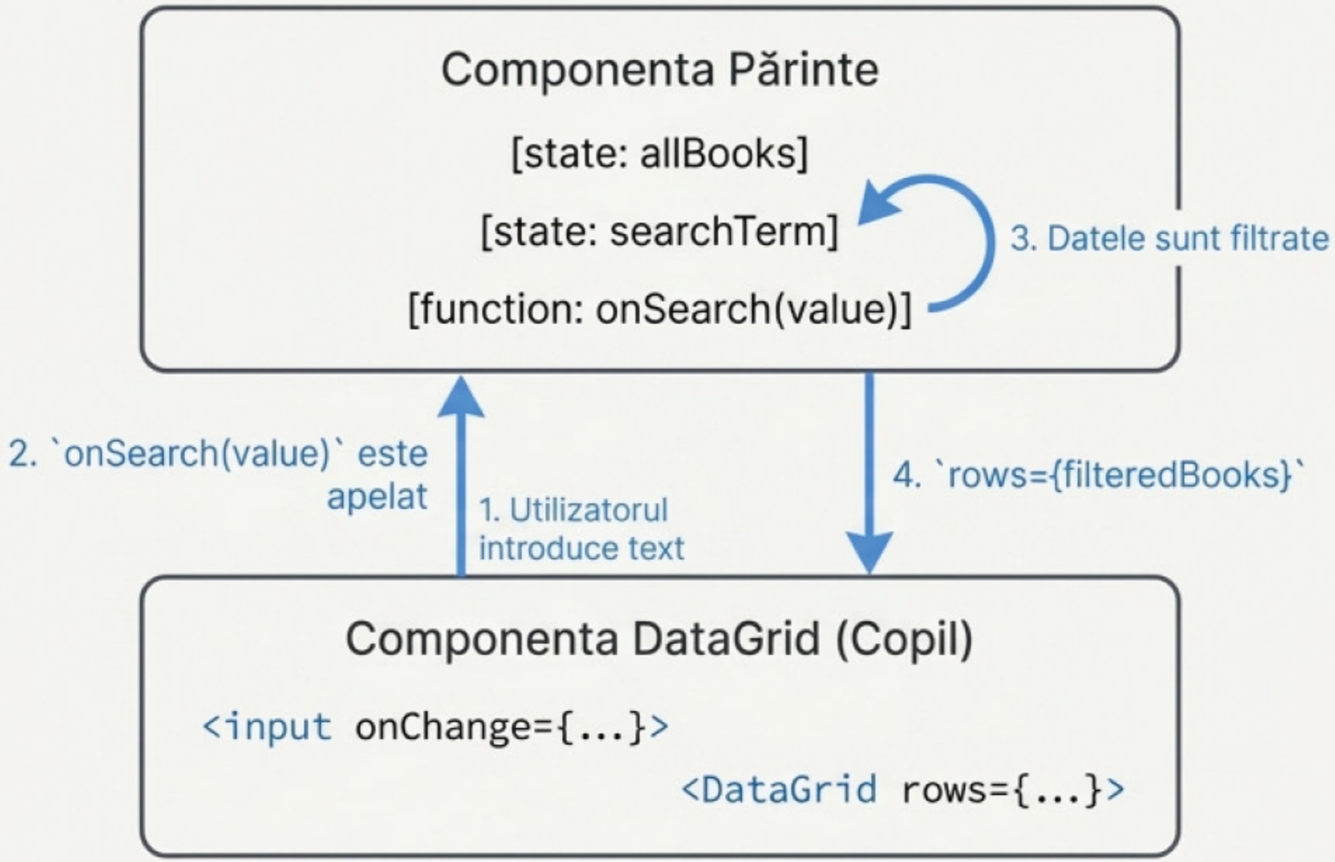
Pilonul 3: Motorul Interactiv

Implementarea Căutării prin Filtrare Manuală

Deși DataGrid are filtre proprii, vom implementa o căutare manuală pentru un control total. Logica nu va locui în componenta de afișare, ci în componenta sa părinte. Acest principiu se numește **'Lifting State Up'**.

- 1. **Interfața Utilizator**: Un element **<input>** pentru a capta textul.
- 2. **Mecanismul de Eveniment**: Funcția `onChange` care declanșează fluxul.
- 3. **Delegarea Logicii**: O funcție `callback` transmisă prin `props`.
- 4. **Filtrarea Datelor**: Logica efectivă care rulează în componenta părinte.

Arhitectura Fluxului de Date: 'Lifting State Up' în Acțiune



1. Utilizatorul scrie în câmpul de căutare.
2. Evenimentul `onChange` apelează funcția `onSearch` primită prin `props`, trimițând valoarea în sus.
3. Componenta părinte actualizează starea și filtrează lista de date originală.
4. Noua listă filtrată este trimisă înapoi componentei copil, cauzând o re-randare.

Implementarea Practică a Fluxului

Componenta de Afișare (*Stateless UI*)

```
function BookGrid({ rows, onSearch }) {
  return (
    <div>
      <input
        placeholder="Caută după titlu..."
        // Evenimentul care declanșează fluxul
        onChange={(e) => onSearch(e.target.value)}
      />
      <DataGrid rows={rows} ... />
    </div>
  );
}
```

Trimite valoarea în sus către părinte.

Componenta de Control (*Stateful*)

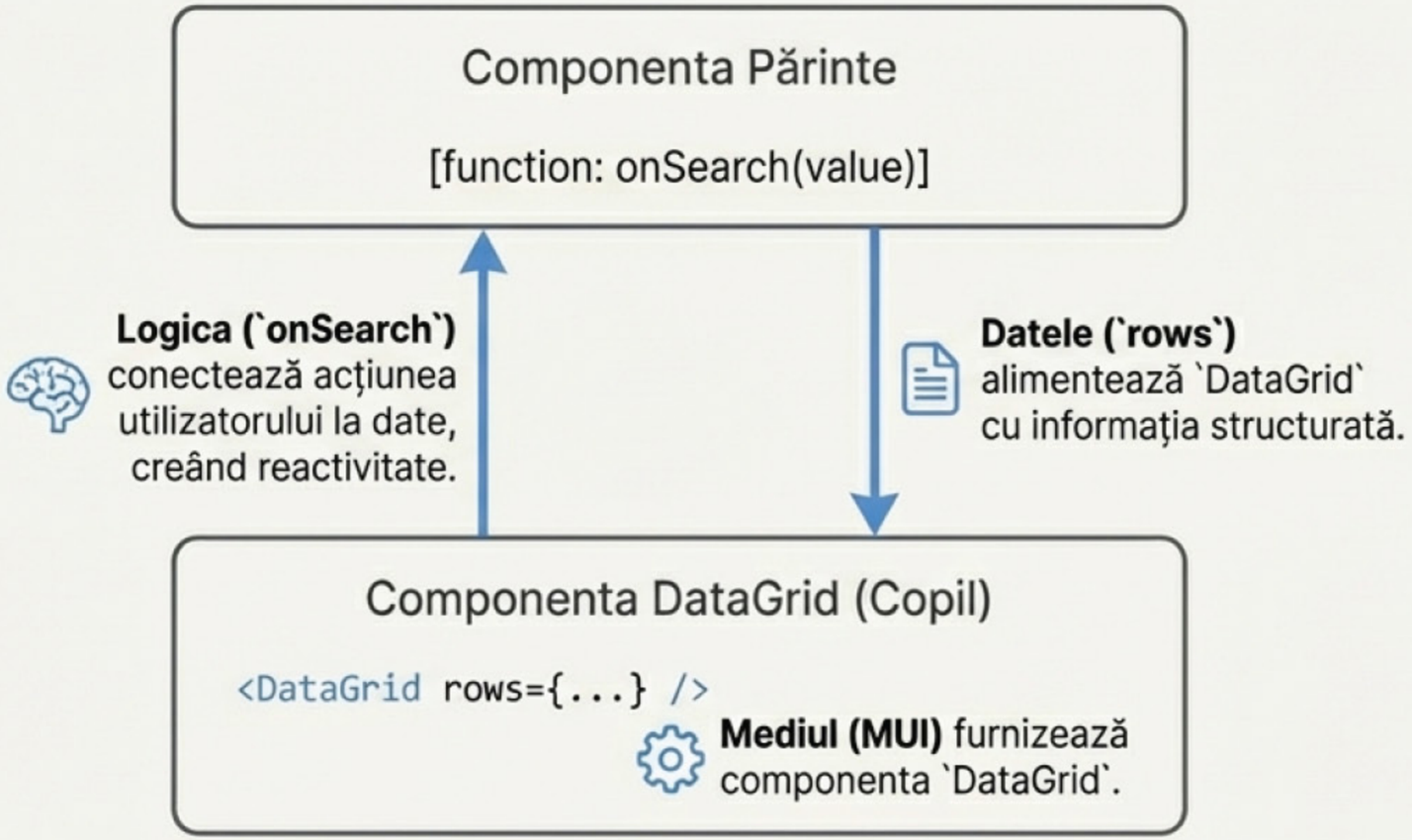
```
function App() {
  const [allBooks, setAllBooks] = useState([...]);
  const [filteredBooks, setFilteredBooks] = useState(allBooks);

  // Funcţia callback care execută filtrarea
  const handleSearch = (searchTerm) => {
    const filtered = allBooks.filter(book =>
     book.title.toLowerCase().includes(searchTerm.toLowerCase())
    );
    setFilteredBooks(filtered);
  };

  return <BookGrid rows={filteredBooks} onSearch={handleSearch} />;
}
```

Primește valoarea, filtrează datele și actualizează starea.

Sinteza: Cele Trei Piloni în Sinergie



Prin această arhitectură, separăm responsabilitățile: componenta `DataGrid` se ocupă exclusiv de afișare, în timp ce componenta părinte gestionează starea și logica de business.
Rezultatul este un cod mai curat, mai predictibil și mai ușor de întreținut.