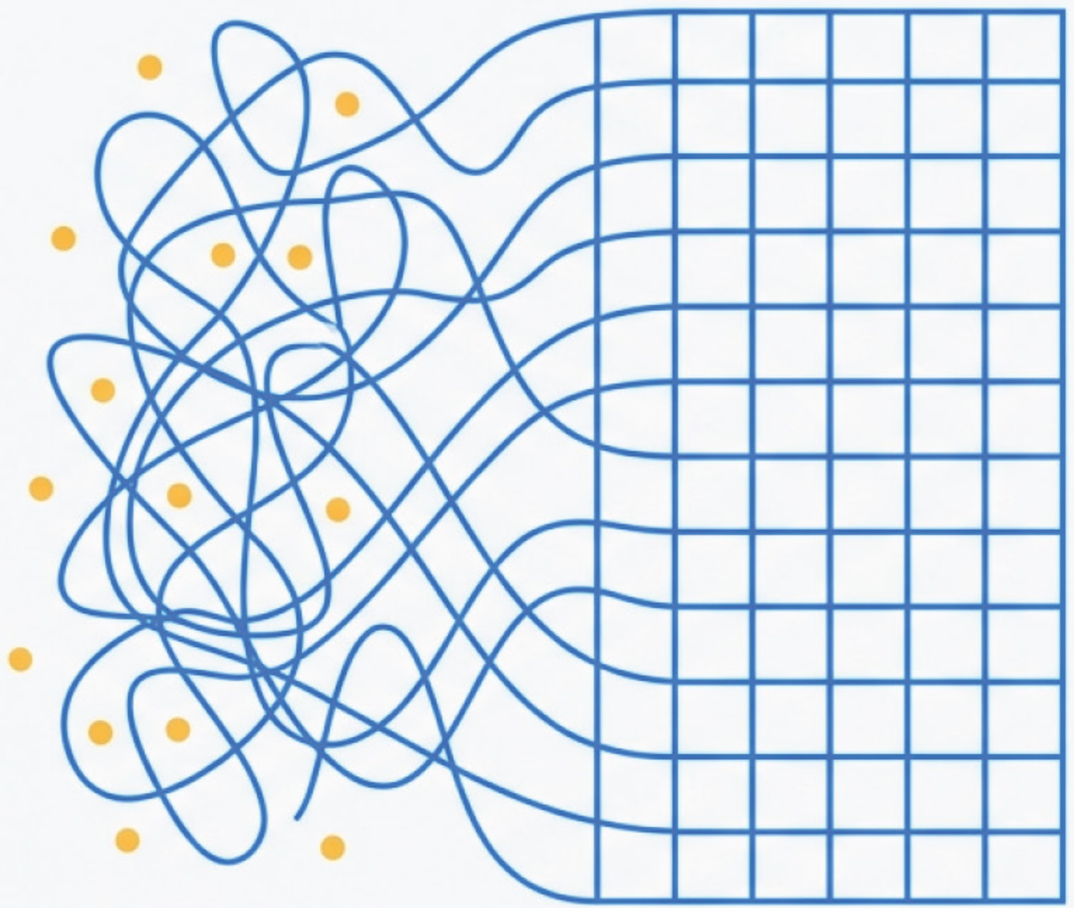
De la Haos la Claritate: Gestionarea Datelor în React prin Design Declarativ

Stăpânirea `DataGrid` pentru Paginare și Filtrare Eficientă



Provocarea: Supraîncărcarea UI-ului cu Date

Seturile mari de date pot bloca interfața și degrada experiența utilizatorului. Abordarea manuală în React implică o complexitate ridicată:

- Implementarea manuală a logicii de `slicing` a datelor.
- Gestionarea stării pentru pagina curentă și dimensiunea paginii.
- Construirea de la zero a controalelor de navigare.
- Scrierea unui cod imperativ, greu de întreținut și predispus la erori.

```
const [currentPage, setCurrentPage] = useState(1);
const itemsPerPage = 10;

const startIndex = (currentPage - 1) * itemsPerPage;
const endIndex = startIndex + itemsPerPage;

const paginatedData = data.slice(startIndex, endIndex);
```

Soluția: O Schimbare de Paradigmă spre Designul Declarativ

Imperativ (Cum)

Îi spui aplicației pas cu pas *cum* să facă ceva. «Calculează indexul, taie array-ul, randează bucata.»

Declarativ (Ce)

Declari *ce* vrei să obții, iar biblioteca se ocupă de implementare. «Vreau ca acest tabel să aibă paginare.»

Componentele moderne precum `DataGrid` (ex: MUI X) sunt construite pe acest principiu. Ele abstractizează complexitatea și oferă un API curat și declarativ.

Stăpânirea Volumelor Mari: Paginarea Activată printr-o Singură Proprietate

Pentru a activa mecanismul de paginare, pur și simplu adăugați proprietatea `**pagination**` componentei `**DataGrid**`. Componenta va randa automat controalele de navigare și va gestiona împărțirea datelor.

```
import { DataGrid } from '@mui/x-data-grid';

function MyTable() {
  return (
    <DataGrid
      rows={rows}
      columns={columns}
      pagination
    />
  );
}
```

Flexibilitate la Îndemână: Configurarea Paginării

Puteți personaliza cu ușurință opțiunile de paginare și starea inițială direct prin props:

- **pageSizeOptions**: Un array de numere ce definește opțiunile disponibile în dropdown-ul pentru dimensiunea paginii.
- **initialState**: Un obiect pentru a seta starea implicită, cum ar fi pagina și dimensiunea paginii la prima randare.

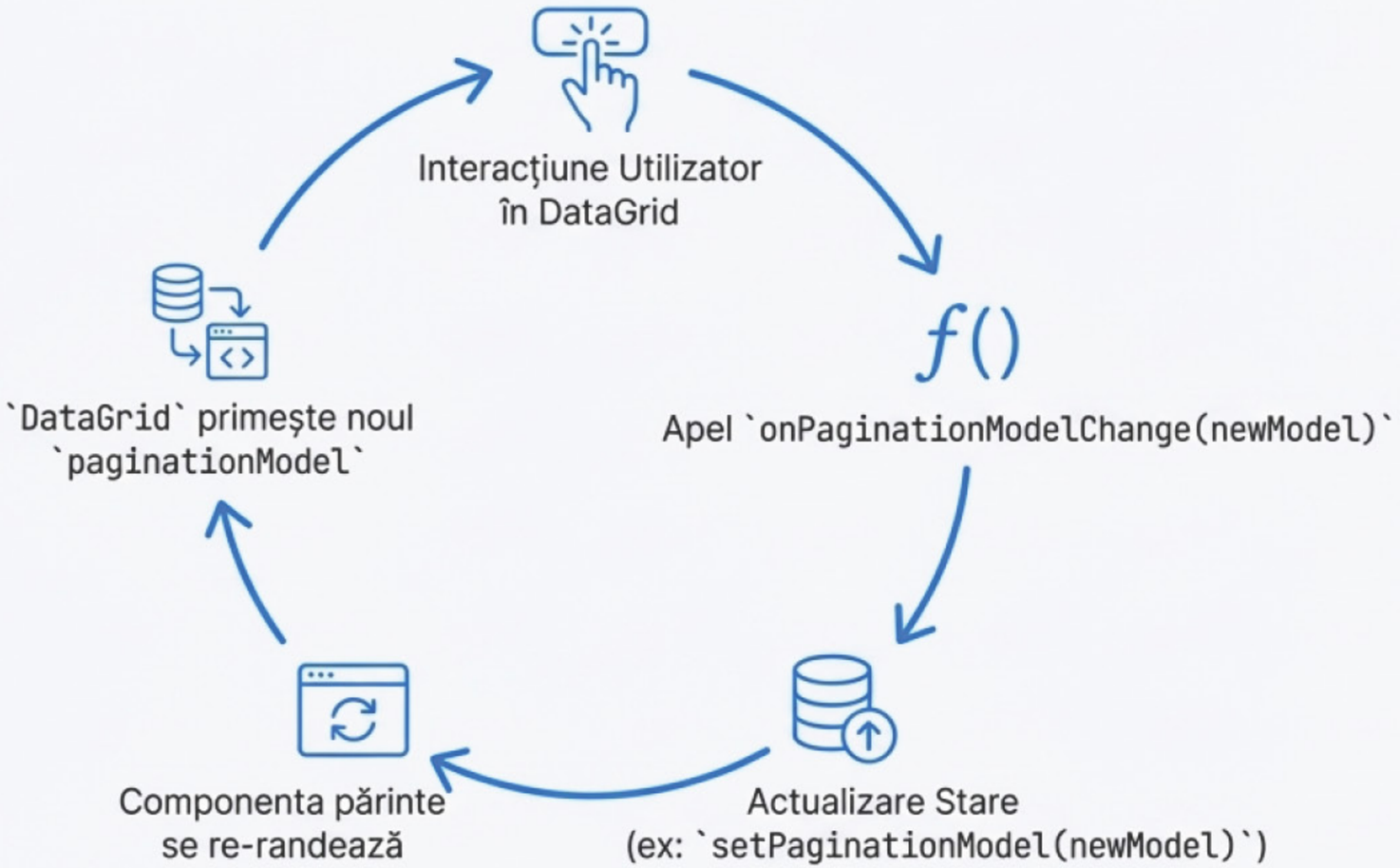
```
<DataGrid
  rows={rows}
  columns={columns}
  pagination
  pageSizeOptions={[5, 10, 25]}
  initialState={{
    pagination: {
      paginationModel: { pageSize: 5, page: 0 },
    },
  }}
/>
```

Sincronizarea Stării: Puterea Modelului Controlat

Pentru control total, putem gestiona starea paginării în componenta părinte. Acest lucru permite sincronizarea cu URL-ul, salvarea preferințelor utilizatorului sau declanșarea altor acțiuni.

paginationModel:
Prop care *primește* starea curentă (ex: `{ page: 1, pageSize: 10 }`).

onPaginationModelChange:
Handler care este *invocat* la fiecare schimbare făcută de utilizator.



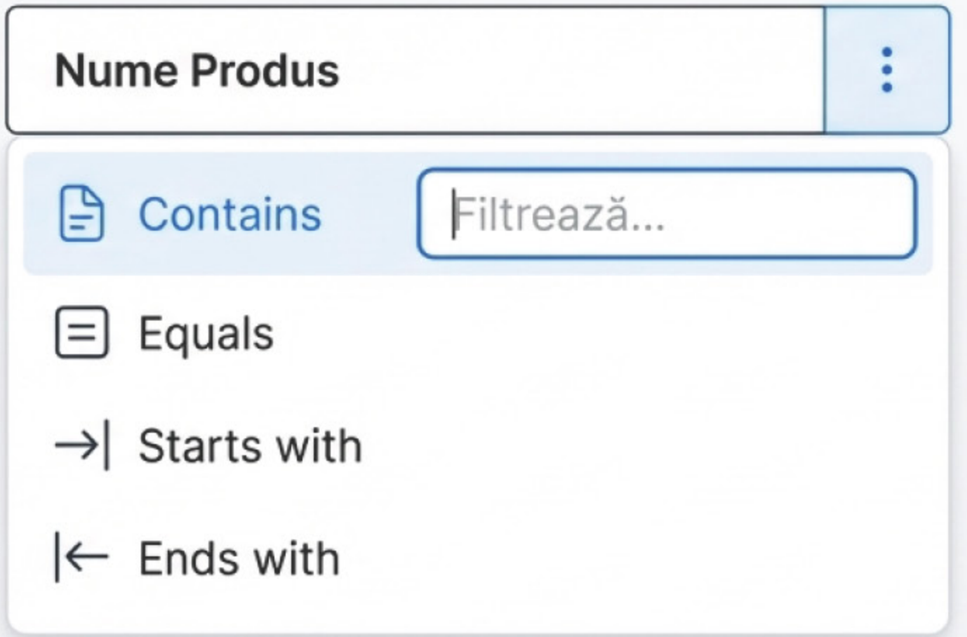
Rafinarea Datelor: Filtrarea Declarativă pe Coloană

Permiteți utilizatorilor să găsească exact informația de care au nevoie, direct în interfață. DataGrid oferă mecanisme de filtrare puternice, activate implicit pe coloanele configurate corespunzător.

Key Concept

Asemenea paginării, starea filtrelor poate fi controlată de componenta părinte pentru scenarii avansate, cum ar fi pre-popularea filtrelor sau salvarea lor.

- **`filterModel`**: Prop pentru a seta starea filtrelor.
- **`onFilterModelChange`**: Handler pentru a reacționa la schimbările de filtru.



Sinergie în Acțiune: Un Sistem de Date Coerent și Reactiv

Prin combinarea modelelor controlate pentru paginare și filtrare, starea completă a tabelului este gestionată de componenta React. Aceasta devine singura sursă de adevăr (`single source of truth`), permițând o logică de business complexă și o experiență predictibilă.

```
function ControlledDataGrid() {
  const [paginationModel, setPaginationModel] = React.useState({
    pageSize: 5,
    page: 0,
  });

  const [filterModel, setFilterModel] = React.useState({
    items: [],
  });

  return (
    <DataGrid
      // ... rows & columns
      pagination
      paginationMode="server"
      paginationModel={paginationModel}
      onPaginationModelChange={setPaginationModel}

      filterMode="server"
      filterModel={filterModel}
      onFilterModelChange={setFilterModel}
    />
  );
}
```

Adoptați Declarativul: Scrieți Mai Puțin Cod, Livrați Mai Multă Valoare

- **Cod Redus**: Eliminați zeci de linii de cod imperativ pentru managementul stării UI.
- **Mentenabilitate Crescută:** Logica este încapsulată și ușor de înțeles. API-ul declarativ este auto-documentat.
- **Experiență Superioară**: Oferiți utilizatorilor funcționalități avansate (paginare, filtrare, sortare) testate și performante, «out of the box».
- **Focus pe Logică:** Eliberați-vă de implementarea detaliilor de UI și concentrați-vă pe logica specifică aplicației dumneavoastră.