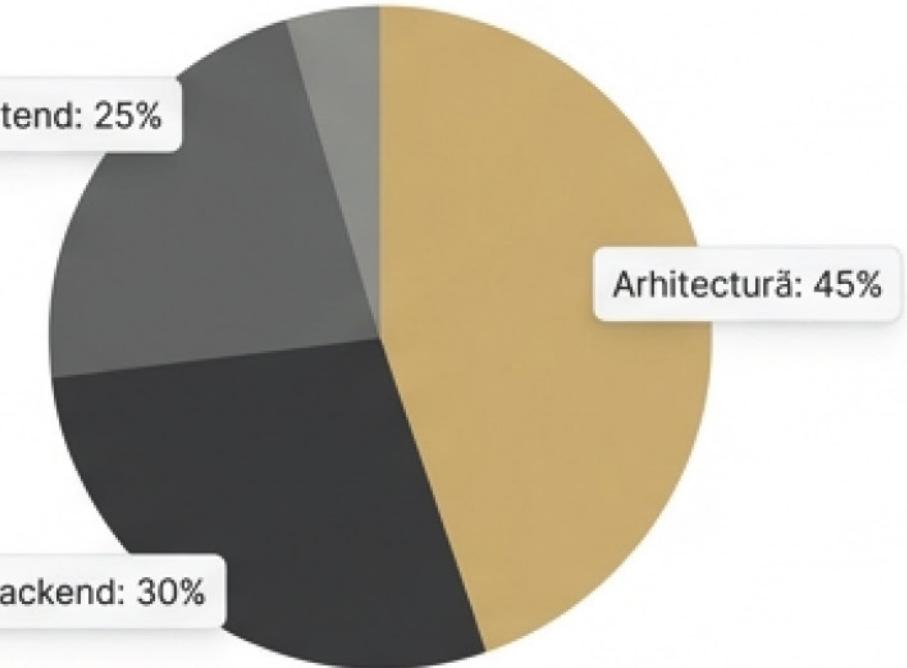


De la Clic la Grafic

Anatomia unui Flux de Date Reactiv și Unidirecțional

Titlu	Autor	An
Arhitectura Datelor Moderne	Elena Dumitrescu	2023
Arhitectura Datelor Moderne	Mara Simster	2023
Arhitectura Datelor Moderne	Elena Dumitrescu	2023
Arhitectura Datelor Moderne	Mara Simster	2023
Compute Datelor Moderne	Elena Dumitrescu	2023
Arhitectura Datelor Moderne	Mara Simster	2023
Arhitectura Ditelor Moderne	Elena Dumitrescu	2023

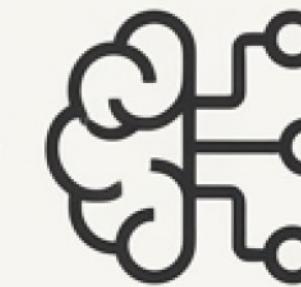
Distribuția Vizuală a Proiectului



⌚ Arhitectură: 45% 💼 Backend: 30% 📊 Frontend: 25%

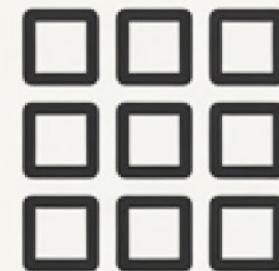
O analiză a modului în care o singură interacțiune a utilizatorului declanșează un lanț de evenimente controlate de starea aplicației, culminând cu o vizualizare dinamică a datelor.

Protagoniștii Scenariului Nostru



Componența Părinte (`<BooksApp />`)

Centrul de control. Implementează principiul 'Lifting State Up', orchestrând vizibilitatea și conținutul componentelor copil.



Componența `DataGrid`

Punctul de plecare al interacțiunii. Acționează ca un generator de evenimente, capturând selecția utilizatorului fără a reține starea acesteia.



Componența `Dialog` & `Pie Chart`

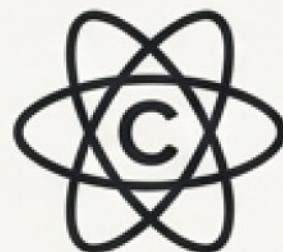
Rezultatul final. O componentă randată condiționat care primește datele prelucrate și le afișează într-un format vizual inteligibil.

Fundată Tehnică: Ecosistemul de Vizualizare

Dependențele Esențiale



Biblioteca fundamentală JavaScript, bazată pe canvas, pentru randarea graficelor.



Wrapper-ul React care permite utilizarea declarativă a graficelor ca și componente.



Furnizorul componentei <Dialog> utilizată pentru containerul modal.

Configurare Mandatorie (Chart.js v3+)

Chart.js adoptă un model modular. Pentru a randa un grafic de tip `Pie Chart`, înregistrarea manuală a elementelor constitutive este obligatorie și previne eșecul randării.

```
import { Chart, ArcElement, Tooltip, Legend }  
from 'chart.js';  
  
Chart.register(ArcElement, Tooltip, Legend);
```

Nerespectarea acestei înregistrări duce la un grafic gol sau la erori în consolă. Componentele `ArcElement`, `Tooltip` și `Legend` sunt esențiale pentru graficele circulare.

Călătoria Începe: Capturarea Evenimentului `onRowClick`

Book	Rand: ID	Columns	Rânduui
Chenweniers Leanny	1	Staran	1950
Throny Mordiron	2	Doltar	1993
Harriem Brook		Ammad	1987
The Opportunities	2	C'sran	1997
Germie Burdens	3	Jake	2009
John Bruils	5	Pravley	2007

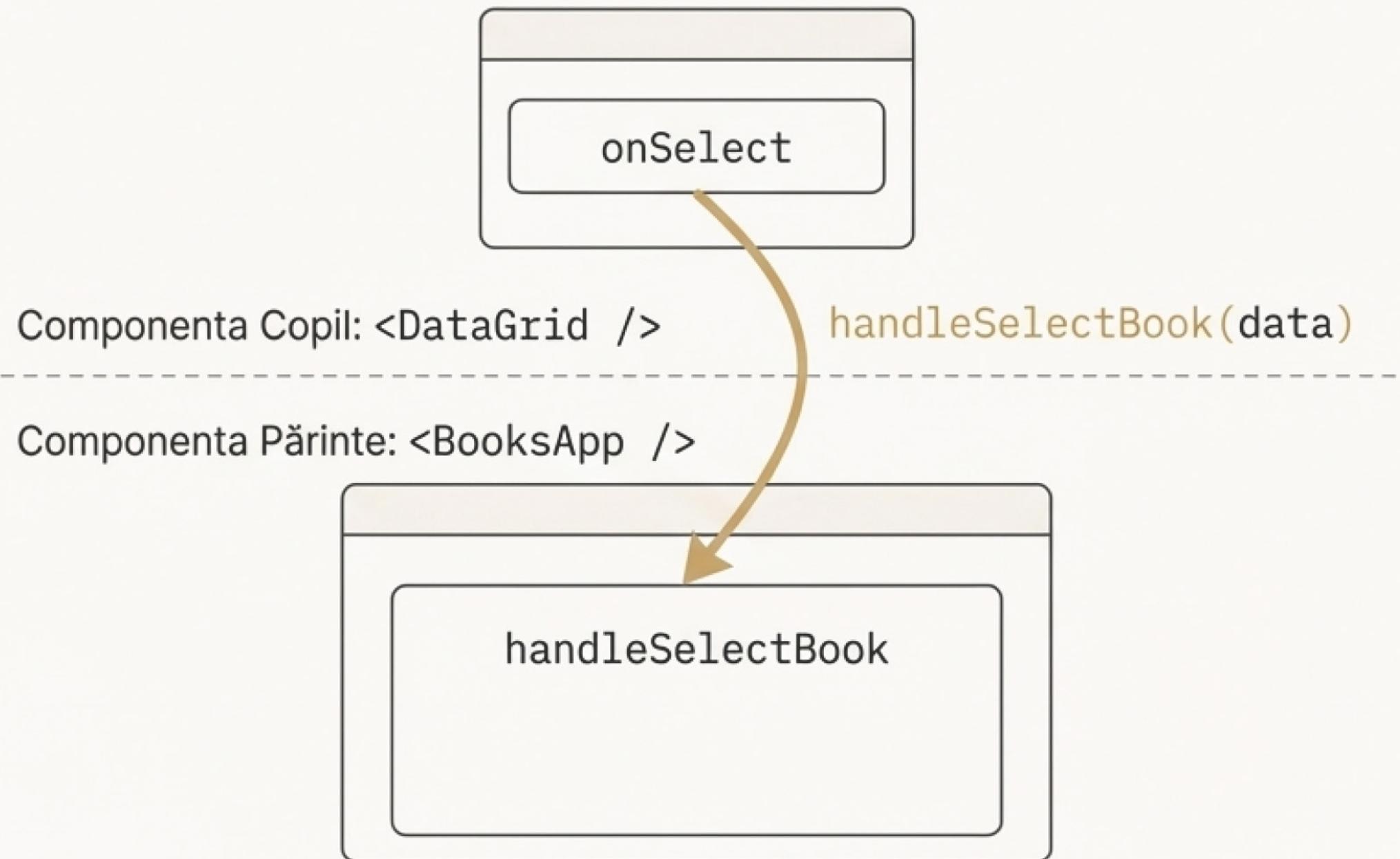
Interacțiunea este inițiată în componenta 'DataGrid' prin intermediul proprietății 'onRowClick'.

Acest 'prop' expune un handler care primește un obiect 'params', conținând datele complete ale rândului selectat (**params.row**).

```
<DataGrid  
  rows={books}  
  columns={columns}  
  onRowClick={(params) =>  
    onSelect(params.row)}  
/>
```

Acest obiect este 'ștafeta' – sarcina utilitară (payload) care va fi transmisă de-a lungul fluxului de date.

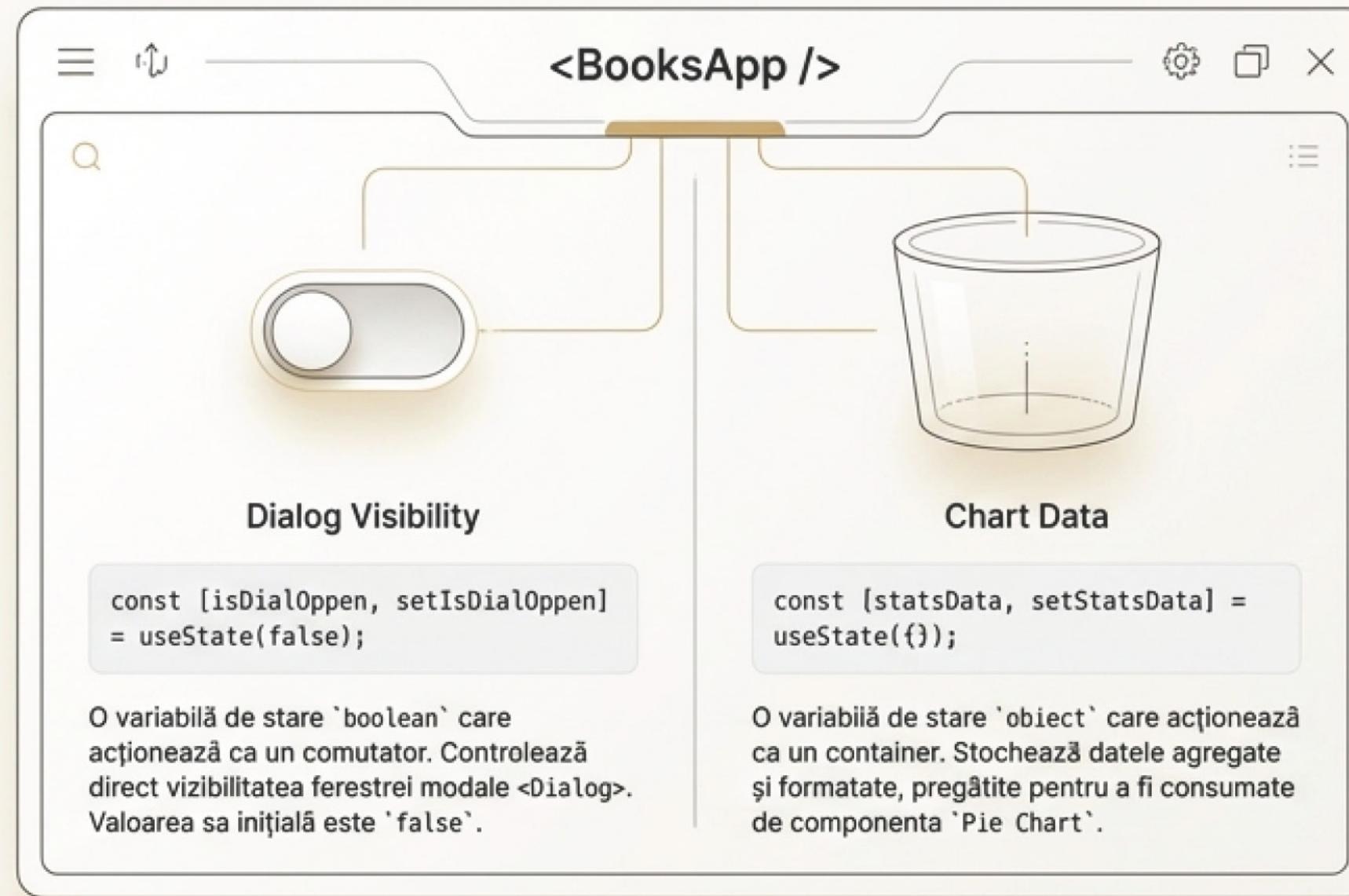
Predarea Ștafetei: Delegarea Controlului către Părinte



Flux de Date Invers (Callback)

- Componenta copil (`DataGrid`) nu gestionează efectul selecției. Ea doar notifică componenta părinte despre eveniment.
- Funcția `handleSelectBook` este un *callback* pasat ca *prop* către `DataGrid`, permitând transferul de date de la copil la părinte. Acest design menține componenta copil reutilizabilă și 'stateless'.

Centrul de Comandă: Principiul “Lifting State Up”

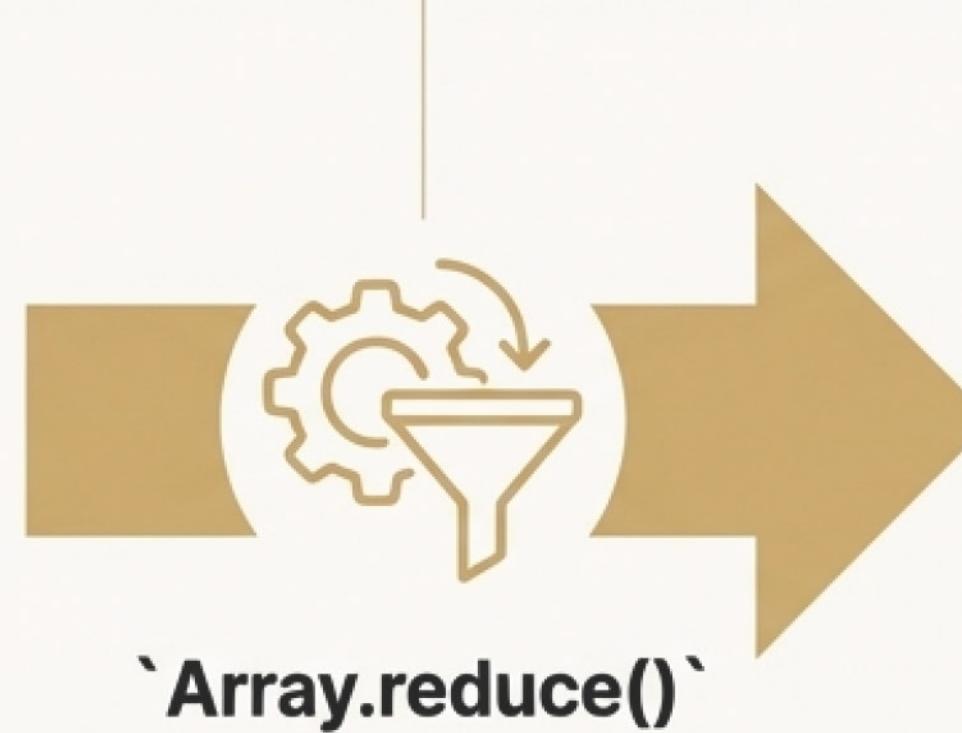


Logica de afișare și datele necesare graficului sunt centralizate în componenta părinte. Acesta este **"Lifting State Up"** în acțiune: starea este ridicată la cel mai apropiat strămoș comun al componentelor care depind de ea.

Prelucrarea Datelor: De la Structură Brută la Formatul Graficului

Date Brute

```
[[  
  { author: 'Autor1', ... },  
  { author: 'Autor2', ... },  
  { author: 'Autor1', ... }  
]]
```



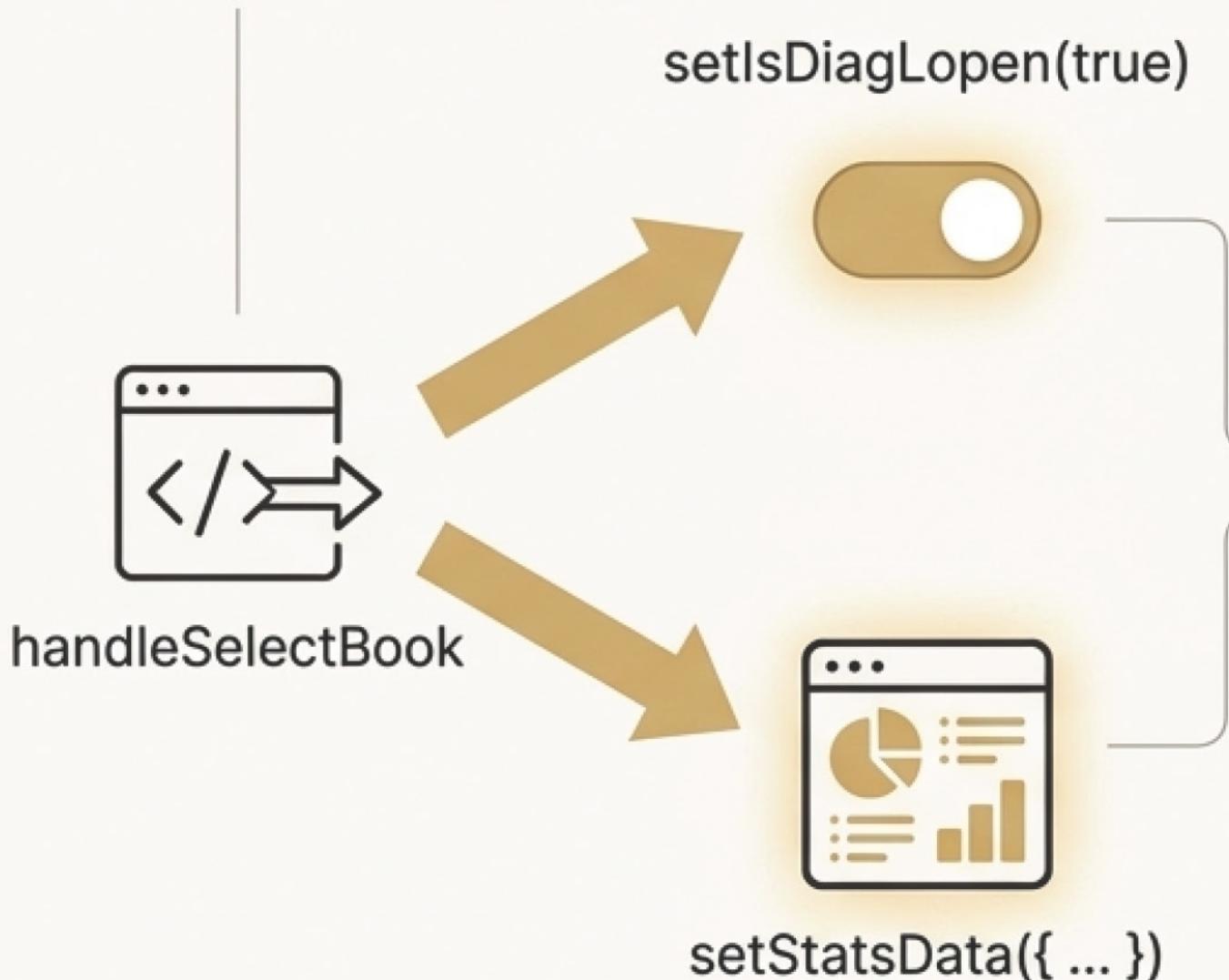
Structură `Chart.js`

```
{  
  labels: ['Autor1', 'Autor2'],  
  datasets: [{  
    data: [2, 1],  
    // ...alte opțiuni de stilizare  
  }]  
}
```

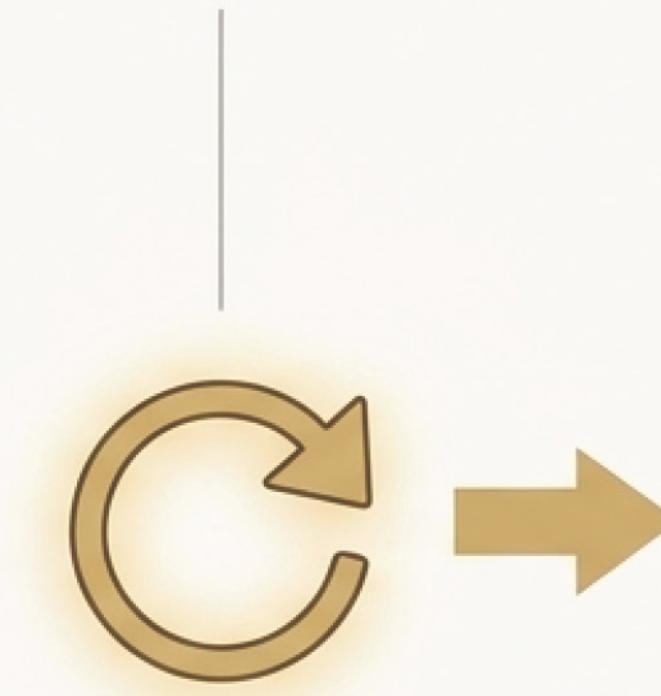
Funcția `handleSelectBook` execută un calcul dinamic (adesea folosind `Array.reduce()`) pentru a agrega datele. Rezultatul este un obiect perfect structurat, cu cheile `labels` și `datasets`, pregătit pentru a fi pasat direct componentei de grafic.

Marea Dezvăluire: Randarea Condiționată și Vizualizarea

The Cause



The Trigger



The Effect



```
{isDiaglopen && (
  <Dialog open={isDiaglopen}
  onClose={() => setIsDiaglopen(false)}>
    /* Alte elemente de dialog */
    <Pie data={statsData} />
  </Dialog>
)}
```

Actualizarea stării `isdiaglopen` la `true` satisfacă condiția de randare. Componenta `<Dialog>` este montată în DOM, primind datele finale din starea `statsData` prin props. Ciclul este complet.

Harta Completă a Fluxului de Date Unidirecțional

Actiune → Stare → UI

Acest ciclu este esența arhitecturii reactive și predictibile în React.



User Click (pe `<DataGrid />`)

Event: onRowClick

Callback către Părinte (`<BooksApp />`)

Mechanism: props.onSelect(rowData)

Actualizare Stare Părinte

Action: setIsDialogOpen(true), setStatsData(..)

Reîncărcare Declanșată

Principle: React's reconciliation

"Props Pasate Descendent" (către <Dialog /> & <Pie />)

Data: open={isDialogOpen}, data={statsData}

UI Final Randat

Result: Modal and Chart are visible.

Anexă: Fragmente de Cod Cheie

1. Capturarea Evenimentului (Componenta Copil)

```
// DataGridComponent.jsx
<DataGrid onClick={(params) =>
  props.onSelect(params.row)} />
```

2. Definiția Stării și a Handler-ului (Componenta Părinte)

```
// BooksApp.jsx
const [isDiagloppen, setIsDiagloppen] = useState(false);
const [statsData, setStatsData] = useState({});

const handleSelectBook = (bookData) => {
  // ...logica de calcul pentru 'processedData'
  setStatsData(processedData);
  setIsDiagloppen(true);
};
```

3. Pasarea Prop-urilor (Componenta Părinte)

```
// BooksApp.jsx
<DataGridComponent onSelect={handleSelectBook} />
```

4. Randarea Condiționată (Componenta Părinte)

```
// BooksApp.jsx
<Dialog open={isDiagloppen} onClose={() =>
  setIsDiagloppen(false)}>
  <Pie data={statsData} />
</Dialog>
```