

# Arquitectura de Dades Moderna

Angular 19, NgRx SignalStore i la nova Resource API: El futur del frontend.

---

---

# L'Era Post-Observable

Cap a una reactivitat nativa i simplificada.

---



# RxJS vs. Signals: El Canvi



## Paradigma RxJS

Basat en fluxos de dades continus, operadors complexos (switchMap, tap) i gestió de subscripcions manual o via async pipe.



## Paradigma Signals

Estat granular i reactiu. Sense subscripcions. Codi imperatiu asíncron (async/await) integrat en un flux reactiu pur.



# NgRx SignalStore

La SignalStore de NgRx redefineix la gestió d'estat a Angular:

- ✓ **Funcional i composable:** Afegeix funcionalitats mitjançant "features".
- ✓ **Lleugera:** Sense boilerplate de Redux (Actions, Reducers).
- ✓ **Tipat estricte:** Seguretat total en temps de compilació.

600 × 400



# Components Clau del Store



## withState

Defineix l'estat inicial del teu magatzem de dades de forma clara.



## withComputed

Crea senyals derivats que es recalculen automàticament quan l'estat canvia.



## withMethods

On resideix la lògica de negoci i les crides asíncrones al servidor.

---

# La Nova Resource API

Simplifiant les petitions HTTP a Angular 19.



# Gestió **Asíncrona** Nativa

La funció `resource()` gestiona el cicle de vida complet de la dada:

- ▶ Estat de càrrega (`isLoading`).
- ⚠ Gestió d'errors integrada.
- ↻ Recàrrega automàtica quan canvien els paràmetres.

600 × 400

# L'Impacte de l'AbortSignal

100%

Cancel·lació Automàtica

Angular 19 utilitza l'estàndard AbortSignal per "tallar el cable" de peticions obsoletes, evitant condicions de carrera i malbaratament de recursos.



# Cicle de Vida d'una Petició





# Implementant el Debounce

Per evitar saturar el servidor, debouncem el senyal de cerca:

```
// Store logic
query = signal('');
debouncedQuery = toSignal(
  toObservable(this.query).pipe(debounceTime(300))
);

users = resource({
  request: () => this.debouncedQuery(),
  loader: async ({request, abortSignal}) => { ... }
});
```

 **Retard controlat:** Esperem 300ms d'inactivitat.

 **Menys trànsit:** Només s'envia la petició final rellevant.

 **Híbrid perfecte:** RxJS per al temps, Signals per a l'estat.



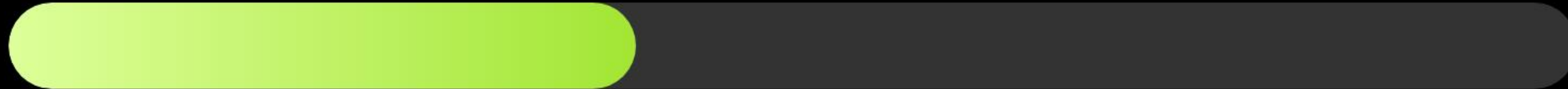
# Eficiència de Xarxa

Sense Debounce/Abort



10 Crides

Amb AbortSignal



4 Crides Completes

Amb Debounce + Abort



1 Crida Final

*Reducció dràstica de la càrrega del servidor i millora de la latència percebuda.*

---

# Preguntes?

Gràcies per la vostra atenció.

[modern-angular-19.dev](https://modern-angular-19.dev) | [@angular\\_signals](https://twitter.com/angular_signals)

---