wavebinder™

Rivoluzione nella gestione dei dati front-end

Indice

PREMESSA

Wavebinder™ è una libreria progettata per aiutare i frontendisti a facilitare la gestione delle dipendenze fra i dati nelle applicazioni front-end.

La crescente domanda di applicazioni web richiede front-end sempre più complessi, che risultano difficili da sviluppare e mantenere.

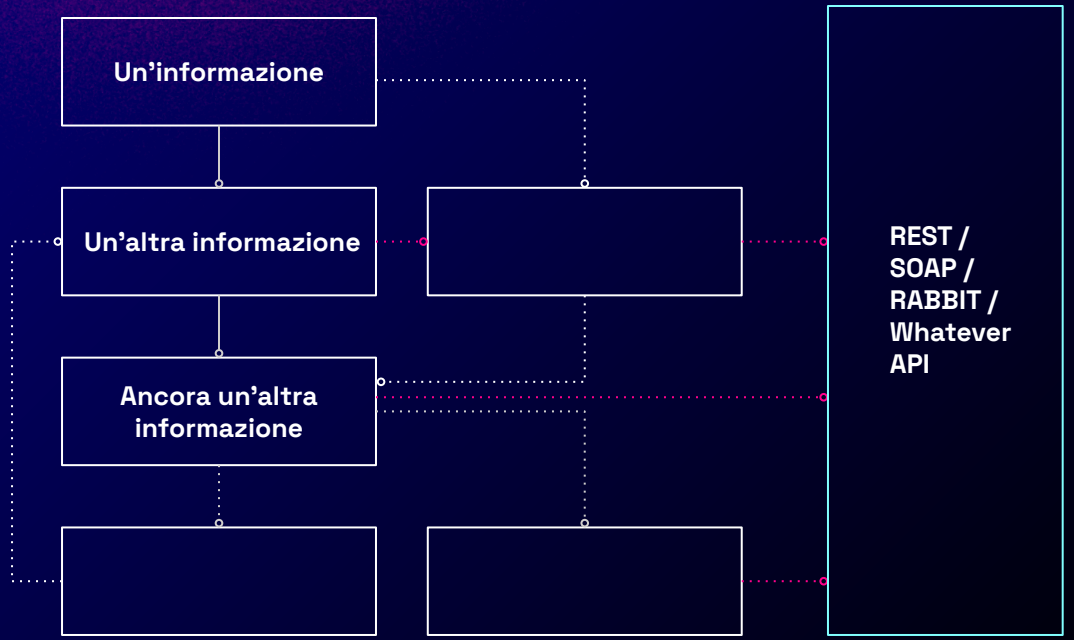
1.

Il problema dei flussi dati

IL PROBLEMA DEI FLUSSI DATI

Perché la gestione dei dati è così complessa?

Nelle applicazioni moderne, ogni campo e componente dipende da altri dati.
I dati non si aggiornano in modo lineare.
Le complessità aumentano quando si utilizzano API lente e asincrone.
Le interfacce sono sempre più complesse e dinamiche.



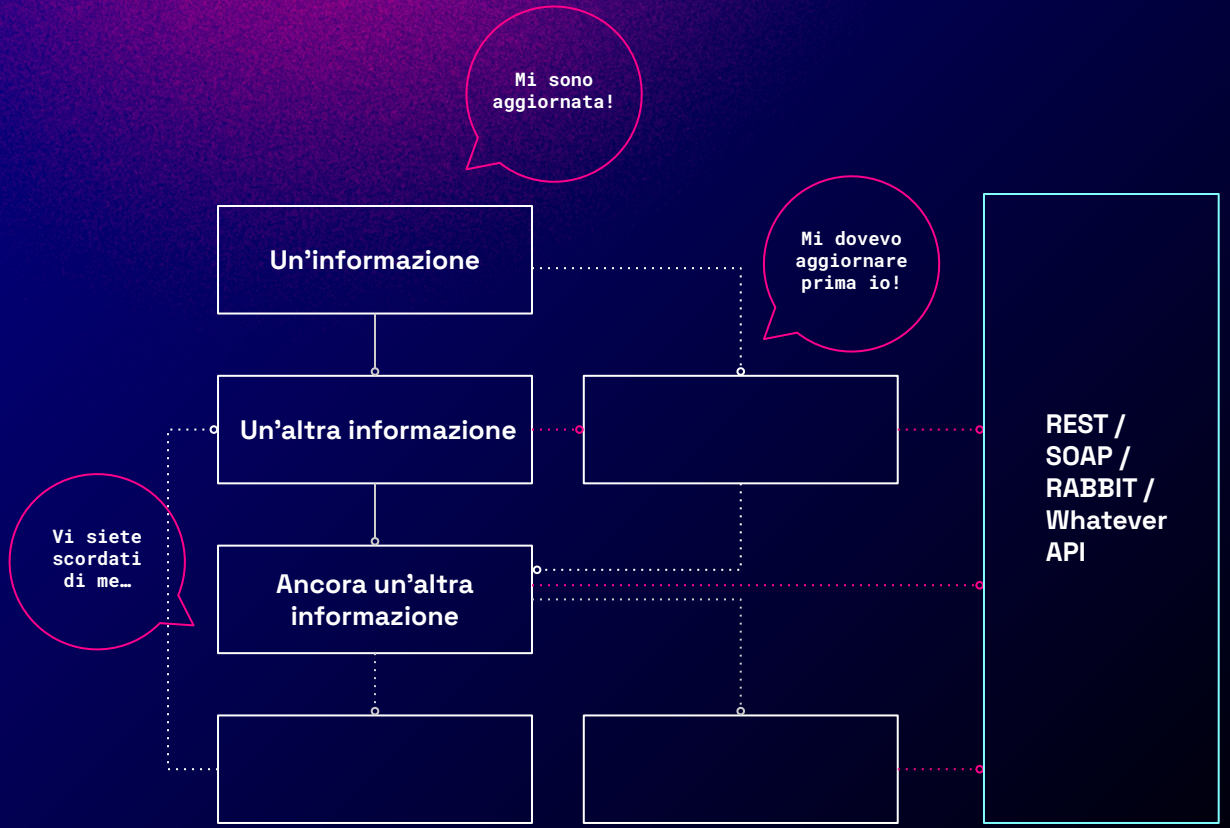
Quali sono le conseguenze?

1. Tempi lunghi e ritardi nello sviluppo delle interfacce

2. Errori e comportamenti indesiderati

3. Esperienza utente non ottimale

4. Difficoltà nel mantenimento e dell'aggiornamento delle logiche



2.

Le sfide dei frontendisti

Cosa cercano gli sviluppatori front-end?

Gli sviluppatori front-end cercano strumenti che semplifichino la loro vita e ottimizzino il flusso di dati tra componenti.

La gestione manuale delle dipendenze dati diventa insostenibile in applicazioni complesse.

Qual è la risposta ai loro bisogni?

WAVEBINDER™ è stato sviluppato per rispondere a queste esigenze, offrendo automazione, flessibilità e controllo.

Wavebinder™

WAVEBINDER™ offre una compatibilità e una facilità d'uso che lo rendono ideale per l'integrazione nei tuoi progetti front-end.

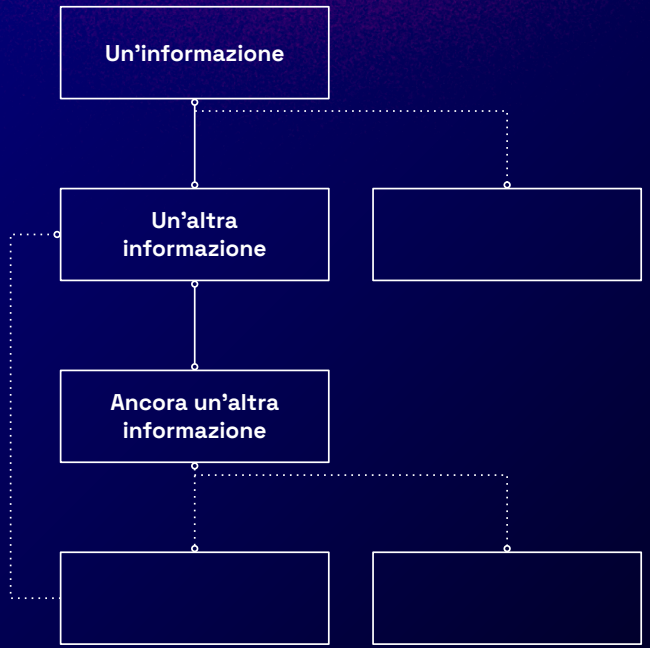
Ecco alcune delle sue caratteristiche principali:

- Disponibile come pacchetto NPM
- Basato su RXJS
- Funziona perfettamente con i principali framework:
  - Vue.js
  - Angular
  - React

3.

Come funziona Wavebinder™

Modellazione dei dati come nodi di un grafo

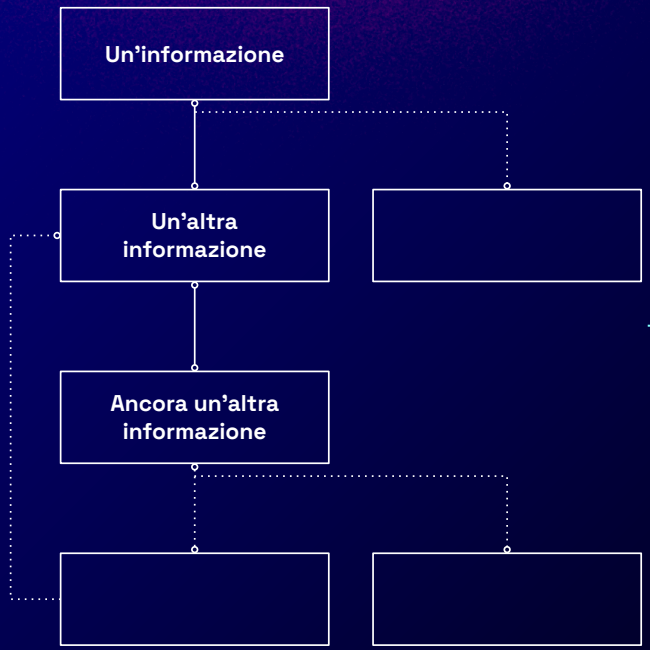


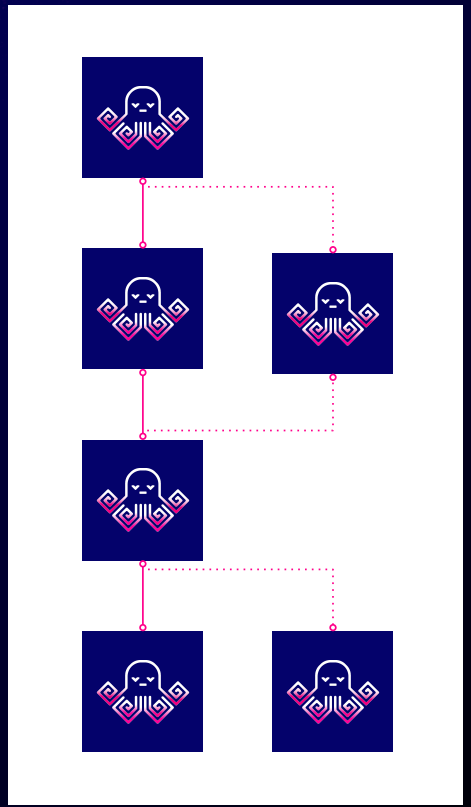
Definizione dei nodi in un oggetto JSON

```
const config: =
[
  {
    "name": "region",
    "type": "MULTI",
    "path": "/region",
    "la": {
      "type": "GET",
      "addr": "/region/list",
      "serviceName": "RETRIEVE_DATA"
    },
    "dep": []
  },
  {
    "name": "province",
    "type": "MULTI",
    "path": "/province",
    "la": {
      "type": "GET",
      "addr": "/{region}/province/list",
      "serviceName": "RETRIEVE_DATA"
    }
  },
  ...
]
```

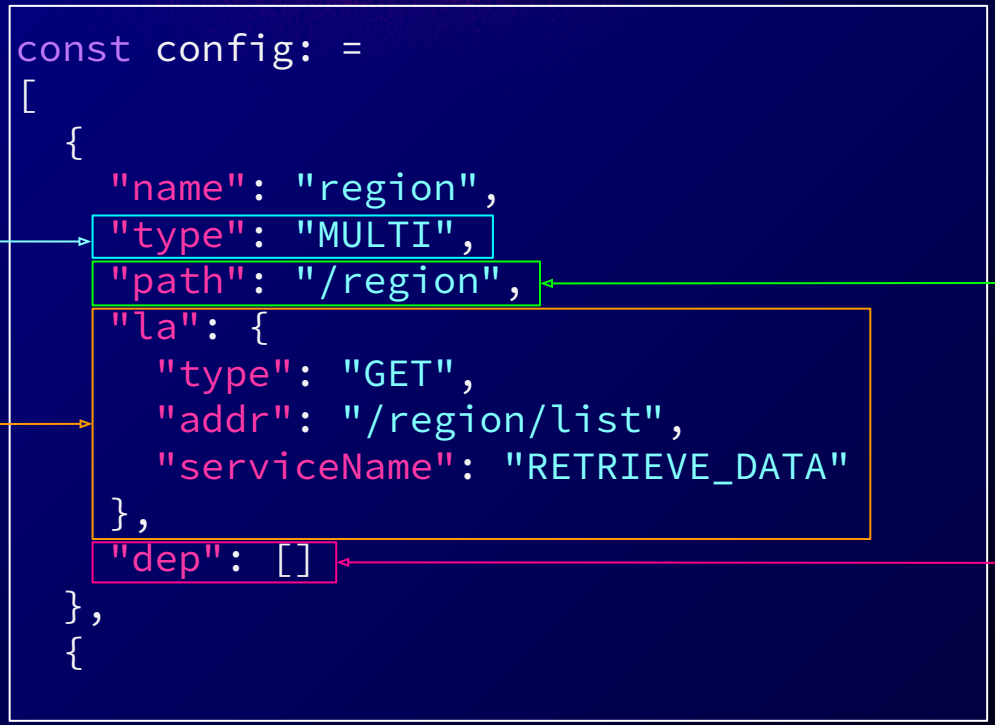
Dipendenze tra i nodi nella struttura JSON

```
{
   "name": "city",
   "type": "MULTI",
   "path": "/city",
   "la": {
     "type": "GET",
     "addr": "/{region}/{province}/city/list",
      "serviceName": "RETRIEVE_DATA"
    },
   "dep": [
     {
       "nodeName": "region",
       "isOptional": false,
       "onUpdate": true,
       "type": "PATH_VARIABLE"
     },
     {
       "nodeName": "province",
       "isOptional": false,
       "onUpdate": true,
       "type": "PATH_VARIABLE"
     }
   ]
}
```





Specificare le diverse proprietà per ogni nodo

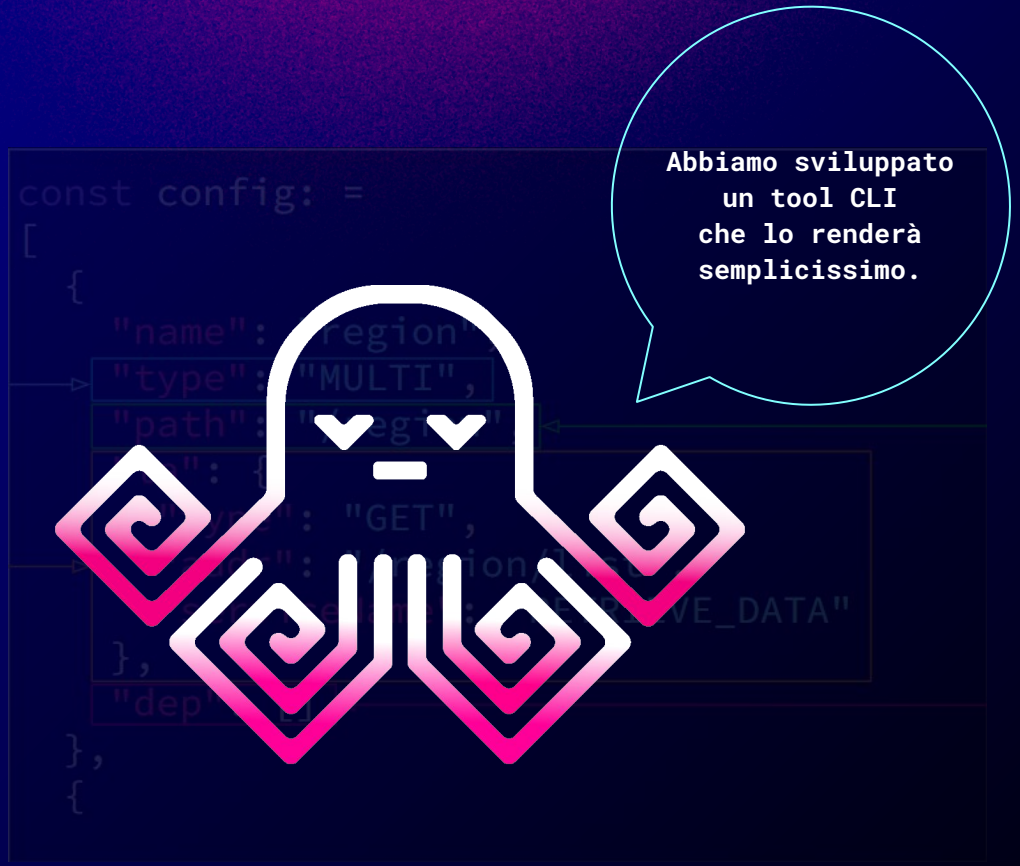


COME FUNZIONA WAVEBINDER™
IL POTERE DELLA MODELLAZIONE A NODI

Specificare le diverse proprietà per ogni nodo

Il tipo di dato che contiene (es. un singolo valore, una lista, un oggetto, ...)

L'azione necessaria per caricare il suo valore (es. una chiamata API, una selezione da parte dell'utente, ...)



Il percorso in cui memorizzarlo su un modello

Le sue dipendenze da altri nodi

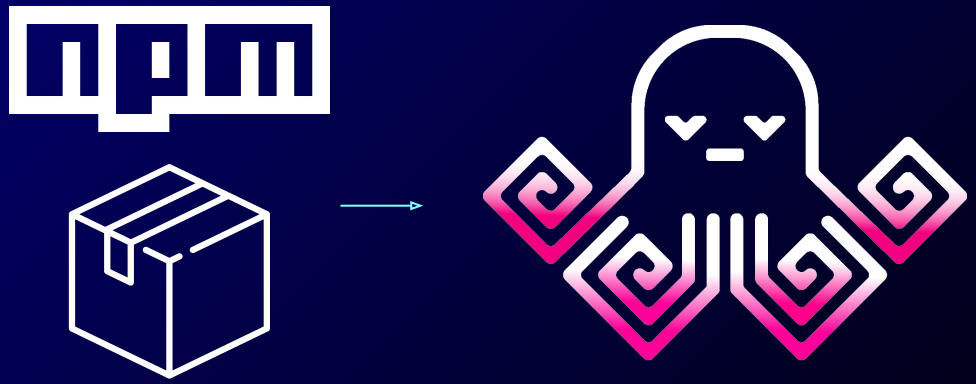
Installazione e configurazione facili

WAVEBINDER™ si integra facilmente nei progetti esistenti.

Basta eseguire «npm install wave-binder» per iniziare.

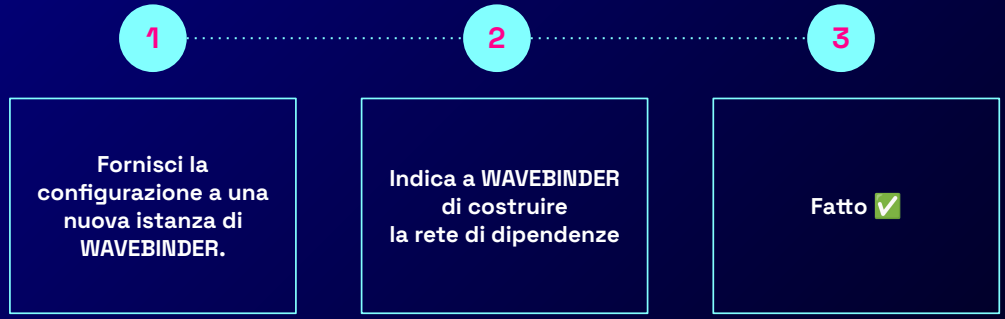
Con pochi passaggi, puoi configurare i nodi e le loro dipendenze usando JSON.

Anche per gli sviluppatori alle prime armi conl'uso di nodi, il tool CLI semplifica la creazione e la gestione delle dipendenze dati.



Funzionalità

Una volta installato **WAVEBINDER™** e modellato il tuo problema nella struttura di configurazione, sei quasi pronto per partire.




```
const wb: WaveBinder = new WaveBinder(config);
```

```
wb.tangleNodes();
```

Funzionalità

WAVEBINDER™ espone:

- I metodi necessari per recuperare i nodi

```
const listNode = wb.getNodeByName('myListNode');
listNode.next(3);
```

```
const regionNode = wb.getNodeByName('region');
regionNode.setSelection(1);
```

```
const regionListNode = wb.getNodeByNameAndType('regionList', 'LIST');
regionListNode.children[1].setSelection(1);
```

```
public constructor(protoNodes: ProtoNode[],
                   extApis: Map<string, HttpServiceSetting>,
                   customFunctions: FunctionInstance[]) {
}

public tangleNodes(): void {
}
```

Funzionalità

WAVEBINDER™ espone:

- I metodi necessari per recuperare i nodi
- Vari modi per interagire con essi.

```
const regionListNode = wb.getNodeByNameAndType('regionList', 'LIST');
regionListNode.children[1].setSelection(1);
```

```
public constructor(protoNodes: ProtoNode[],
                   extApis: Map<string, HttpServiceSetting>,
                   customFunctions: FunctionInstance[]) {
}

public tangleNodes(): void {
}

public getNodes(): WaveBinderNode[] {
}

public getDataPool(): void {
}

public getNodesInfo(logDepth?: number): WaveBinderNodeView[] {
}

public getNodeByName(name: string): WaveBinderNode {
}

public getNodeByNameAndType(name: string,
                            type: string): WaveBinderNode {
```

Funzionalità

WAVEBINDER™ espone:

- I metodi necessari per recuperare i nodi
- Vari modi per interagire con essi.

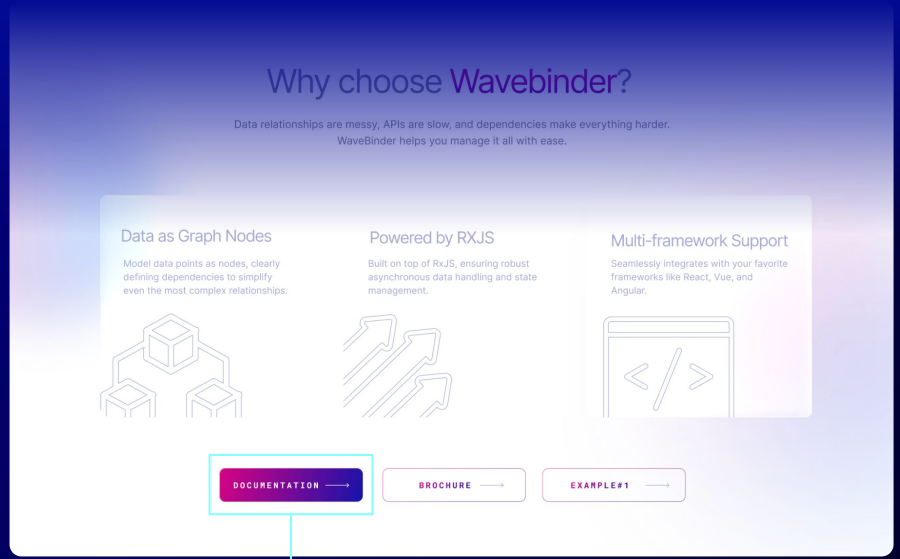
```
}

public getDataPool(): void {
}

public getNodesInfo(logDepth?: number): WaveBinderNodeView[] {
}

public getNodeByName(name: string): WaveBinderNode {
}

public getNodeByNameAndType(name: string,
                            type: string): WaveBinderNode {
}

public getNodeByNameAmongNodes(name: string,
                               nodes: WaveBinderNode[])
                              : WaveBinderNode {
}

public getNodeByNameAndTypeAmongNodes(name: string,
                                      type: string,
                                      nodes: WaveBinderNode[])
                                     : WaveBinderNode {
}
```

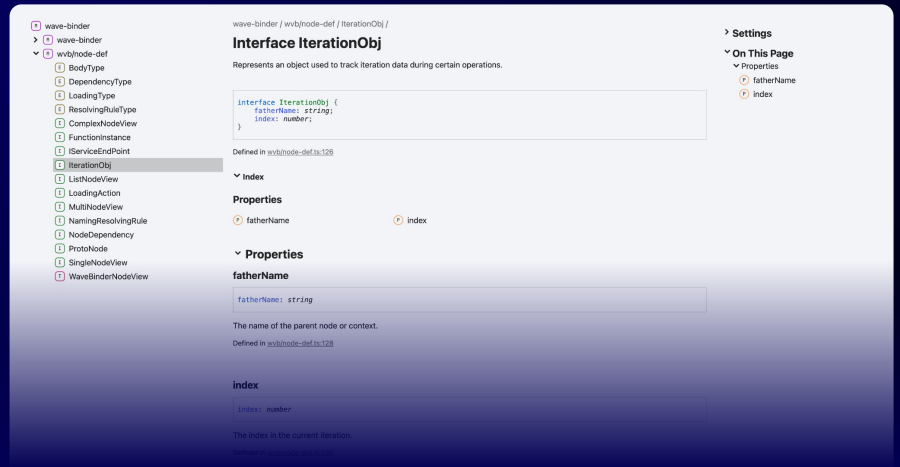
COME FUNZIONA WAVEBINDER™

Documentazione

Ogni aspetto della configurazione e gestione di **WAVEBINDER™** è documentato su **www.wavebinder.it**.

Questo assicura agli sviluppatori una guida costante per configurazioni avanzate e casi particolari, rendendo l'adozione ancora più semplice.





4.

Perché Wavebinder™ funziona

Un libreria potente, con anni di esperienza, che ha la visione di astrarre la complessità una volta per tutte.

Permette a chiunque di concentrarsi solo su ciò che il cliente desidera davvero: business logic.

Ordine e precisione: dipendenze e azioni che organizzano il tuo codice

Per ogni nodo, ti viene chiesto di specificare due cose molto importanti:

- Dipendenze
- Azione di caricamento

```
"dep": [
    {
      "nodeName": "region",
      "isOptional": false,
      "onUpdate": true,
      "type": "PATH_VARIABLE"
    },
    {
      "nodeName": "province",
      "isOptional": false,
      "onUpdate": true,
      "type": "PATH_VARIABLE"
    }
 ]
```

Le dipendenze sono semplicemente riferimenti ad altri nodi

```
"la": {
  "type": "GET",
  "adr": "/region/list",
  "serviceName":"RETRIEVE_DATA"
}
```

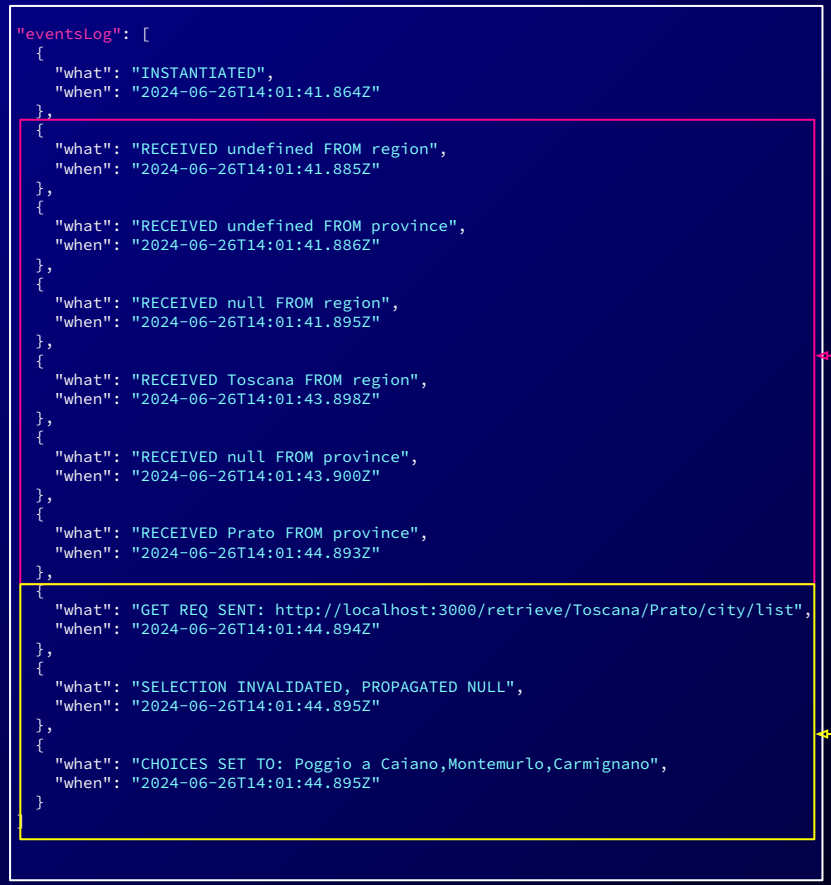
Se in qualsiasi momento quei nodi di riferimento sono completamente e correttamente caricati, l'azione di caricamento viene attivata

Questo, naturalmente, può attivare le azioni di caricamento di altri nodi

PERCHÉ WAVEBINDER™ FUNZIONA

Controllo costante su ogni azione della tua interfaccia

Ogni nodo mantiene un registro di ciò che gli è successo.



wavebinder™

Grazie