

## REQUISITI FUNZIONALI

### Backend – PocketBase

- Il backend deve essere avviato tramite PocketBase in modalità serve.
- Deve esistere una collezione chiamata spese.
- La collezione deve contenere i campi:
  - titolo (string)
  - importo (number/float)
  - data (date)
- I permessi API devono consentire:
  - List
  - View
  - Create
  - Update
  - Delete
- Deve essere possibile leggere, creare, modificare ed eliminare record tramite API REST.
- La risposta deve contenere i record nella chiave items.

### Frontend – React

#### Visualizzazione Spese

- L'applicazione deve recuperare automaticamente tutte le spese da PocketBase al montaggio.
- Le spese devono essere mostrate in una lista ordinata con titolo, data formattata e importo.
- Deve essere mostrato il numero totale di spese.

#### Aggiunta Nuova Spesa

- L'utente deve poter aggiungere una nuova spesa tramite un form con titolo, importo e data.
- Alla sottomissione deve essere inviato un POST a PocketBase.
- L'interfaccia deve aggiornarsi dinamicamente dopo la creazione.

#### Modifica Spesa

- Cliccando su una spesa deve apparire un modale con i campi precompilati.
- L'utente può modificare e salvare i dati.
- L'app deve inviare una richiesta PATCH/UPDATE a PocketBase.
- L'interfaccia deve aggiornarsi senza ricaricare la pagina.

#### Eliminazione Spesa

- Il modale deve includere un pulsante Elimina.
- Deve essere inviata una richiesta DELETE a PocketBase.
- L'interfaccia deve aggiornarsi rimuovendo l'elemento eliminato.

#### Dashboard Grafica

- Deve essere presente una sezione Distribuzione Spese.
- Il grafico deve essere realizzato con Recharts utilizzando un PieChart.
- Le percentuali devono essere calcolate automaticamente.
- La dashboard deve aggiornarsi dopo ogni operazione CRUD.

#### Calcolo Totale

- L'app deve calcolare e mostrare il totale delle spese.
- Il totale deve aggiornarsi in tempo reale.

### REQUISITI NON FUNZIONALI

#### Usabilità

- L'interfaccia deve essere moderna, leggibile e responsiva.
- Gli elementi del layout devono essere ben distanziati.

#### Performance

- Nessun reload completo deve essere necessario per aggiornare la UI.
- Le operazioni CRUD devono essere ottimizzate tramite stato locale.

#### Affidabilità

- Il sistema deve gestire errori di rete e input non validi.

### Manutenibilità

- Il codice deve essere suddiviso in componenti:

- App
- FormSpesa
- ListaSpese
- ItemSpesa
- GraficoSpese

### Portabilità

- Il progetto deve funzionare in qualsiasi ambiente con Node.js e PocketBase.

### Coerenza Tecnologica

- Il frontend deve utilizzare:

- React
- useState, useEffect
- Fetch API
- Recharts