CLI e SDK

Panoramica CLI

La CLI di Riseact è uno strumento che ti permette di creare, sviluppare e distribuire applicazioni che si integrano direttamente con la piattaforma Riseact.

Puoi utilizzare la CLI per creare un nuovo progetto,

Funzionalità

- creare una nuova applicazione con un template predefinito
- distribuire l'applicazione su Riseact

Requisiti

- Avere Git installato
- Un token ngrok per creare un tunnel per il server di sviluppo
- Utilizzare l'ultima versione di Chrome o Firefox

Come iniziare

Per cominciare è necessario installare la CLI di Riseact. Per farlo, scarica la versione del tuo sistema operativo da questa pagina

Una volta scaricato il file, estrai il contenuto e spostati nella cartella tramite il terminale.

Per utilizzare la CLI è necessario aver configurato un account Partner, se non lo hai ancora fatto segui questa guida

Una volta configurato l'account, esegui il comando

```
$ riseact auth login
```


e inserisci le credenziali del tuo account Partner.

Assicurati di aver registrato un'applicazione su Riseact. Crea inizialmente un'applicazione privata su un organizzazione di sviluppo, in modo da poter testare l'applicazione senza pubblicarla.

Una volta fatto potrai inizializzare un nuovo progetto, utilizzando il comando

```
$ riseact app init
```


e segui le istruzioni a schermo. Questo comando creerà una nuova cartella con il nome dell'applicazione e all'interno troverai i file necessari per iniziare a sviluppare.

Per sviluppare l'applicazione, spostati all'interno della cartella e avvia il server di sviluppo con il comando

```
$ riseact app dev
```


Questo comando avvierà un server locale e un tunnel ngrok per permetterti di testare l'applicazione su Riseact.

A questo punto accedi alla tua organizzazione di sviluppo su Riseact e vai alla lista delle applicazioni private, da li potrai attivare l'applicazione e testarla.

Contenuto importato da https://dev.riseact.org/docs/cli/overview il 2026-04-23 durante la migrazione iniziale della KB Metadonors. Aggiornare se il sorgente cambia.

Struttura dell'app

Il template Node delle applicazioni Riseact è progettato per essere flessibile e adattabile a diversi tipi di progetti. Fornisce una struttura di base per l'applicazione, ma è possibile personalizzarla in base alle proprie esigenze.

Struttura di base

La struttura di base di un'applicazione Riseact è la seguente:

```
├── client
│   ├── index.html
│   ├── LICENSE
│   ├── package.json
│   ├── public
│   │   └── riseact.png
│   ├── src
│   │   ├── assets
│   │   │   └── react.svg
│   │   ├── components
│   │   │   ├── Navbar
│   │   │   │   ├── index.tsx
│   │   │   │   └── NavButton.tsx
│   │   │   └── Page
│   │   │       └── index.tsx
│   │   ├── config
│   │   │   ├── network.ts
│   │   │   └── routing.ts
│   │   ├── hooks
│   │   │   └── useOrganization.ts
│   │   ├── main.tsx
│   │   ├── Router.tsx
│   │   ├── routes
│   │   │   ├── Campaigns
│   │   │   │   ├── Create.tsx
```

```
│   │   │   │   ├── Detail.tsx
│   │   │   │   ├── Form.tsx
│   │   │   │   ├── index.tsx
│   │   │   │   └── List
│   │   │   │       ├── Filters.tsx
│   │   │   │       ├── index.tsx
│   │   │   │       └── VisibilityBadge.tsx
│   │   │   └── Home
│   │   │       └── index.tsx
│   │   ├── utils
│   │   │   └── enumTranslate.ts
│   │   └── vite-env.d.ts
│   ├── tsconfig.json
│   └── vite.config.ts
├── common
│   ├── gql-codegen.ts
│   ├── package.json
│   ├── src
│   │   ├── gql
│   │   │   ├── fragment-masking.ts
│   │   │   ├── gql.ts
│   │   │   ├── graphql.ts
│   │   │   └── index.ts
│   │   ├── queries.ts
│   │   └── types.ts
│   └── tsconfig.json
├── Dockerfile
├── LICENSE
├── package.json
├── package-lock.json
├── prisma
│   ├── migrations
│   │   ├── 20231031142218_init
│   │   │   └── migration.sql
│   │   └── migration_lock.toml
│   └── schema.prisma
├── README.md
├── server
```

```
│   ├── package.json
│   ├── src
│   │   ├── config
│   │   │   ├── database.ts
│   │   │   └── riseact.ts
│   │   ├── controllers
│   │   │   └── organization.ts
│   │   ├── env.d.ts
│   │   └── index.ts
│   └── tsconfig.json
├── tsconfig.build.json
└── tsconfig.json
```


Client

La cartella `client` contiene il codice dell'applicazione front-end. Questa cartella è strutturata in modo da separare i componenti, le pagine, i hooks e le utility in cartelle separate.

- `index.html`: file HTML principale dell'applicazione
- `package.json`: file di configurazione del progetto
- `public`: cartella contenente i file statici dell'applicazione
- `src`: cartella contenente il codice sorgente dell'applicazione
  - `assets`: cartella contenente i file multimediali
  - `components`: cartella contenente i componenti dell'applicazione
  - `config`: cartella contenente i file di configurazione
  - `hooks`: cartella contenente i custom hooks
  - `main.tsx`: file principale dell'applicazione
  - `Router.tsx`: file contenente la definizione delle rotte
  - `routes`: cartella contenente le pagine dell'applicazione
  - `utils`: cartella contenente le utility

Common

La cartella `common` contiene il codice condiviso tra il client e il server. Questa cartella è strutturata in modo da separare le query GraphQL, i tipi e i file di configurazione.

- `gql-codegen.ts`: file di configurazione per la generazione dei tipi GraphQL
- `package.json`: file di configurazione del progetto
- `src`: cartella contenente il codice sorgente condiviso

    - `gql`: cartella contenente i file GraphQL
    - `queries.ts`: file contenente le query GraphQL
    - `types.ts`: file contenente i tipi TypeScript

Server

La cartella `server` contiene il codice dell'applicazione back-end. Questa cartella è strutturata in modo da separare i controller, i file di configurazione e i file di ambiente.

- `package.json`: file di configurazione del progetto
- `src`: cartella contenente il codice sorgente dell'applicazione
    - `config`: cartella contenente i file di configurazione
    - `controllers`: cartella contenente i controller dell'applicazione
    - `env.d.ts`: file di definizione dei tipi per le variabili d'ambiente
    - `index.ts`: file principale dell'applicazione

Altri file

- `Dockerfile`: file di configurazione per la creazione dell'immagine Docker
- `LICENSE`: file di licenza del progetto
- `package.json`: file di configurazione del progetto
- `package-lock.json`: file di lock delle dipendenze
- `prisma`: cartella contenente i file di configurazione di Prisma
- `README.md`: file di descrizione del progetto
- `tsconfig.build.json`: file di configurazione per la compilazione del progetto
- `tsconfig.json`: file di configurazione TypeScript del progetto
