UNIVERSITÀ DI TRENTO

Dipartimento di Ingegneria
e Scienza dell'Informazione

Sleep Code

Progetto per il Corso di Ingegneria del Software
Anno Accademico 2023-2024

Documento di Sviluppo

Descrizione:

Numero documento: D4 *Versione documento:* 1.0

Membri del gruppo:
Raffaele Castagna
Alberto Rovesti
Zeno Saletti

Numero gruppo: G17

Ultima revisione: 2 febbraio 2024

Indice

Consigli utili per la consultazione del testo: Se il lettore per file `.pdf` attualmente in uso lo consente, è possibile navigare con più semplicità e velocità all'interno di questo documento cliccando sugli elementi dell'indice.

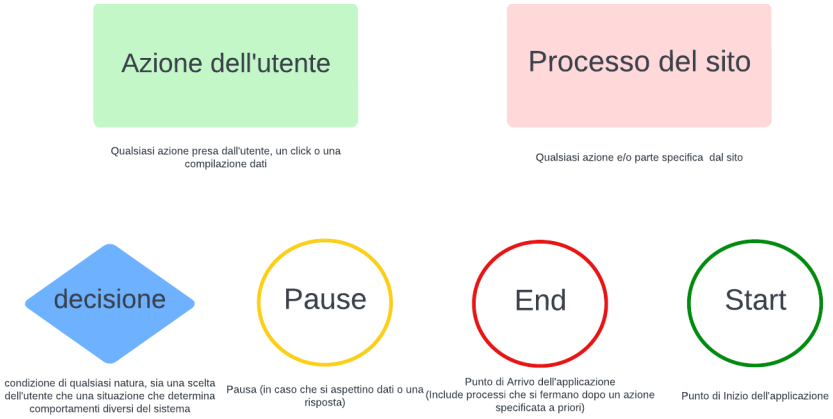
1 Scopo del documento

Il presente documento riporta tutte le informazioni richieste e necessarie per lo Sviluppo di una parte dell'applicazione Sleepcode. In particolare, presenta:

- User Flow legato al ruolo dell'utente (amministratore,autenticato e non)
- User Flow legato all'uso del sito
- Documentazione delle Api attraverso API Model e Modello delle risorse
- Api Fornite per interagire con l'applicazione
- Descrizione delle api fornite
- Risultati delle test suite applicata sulle api

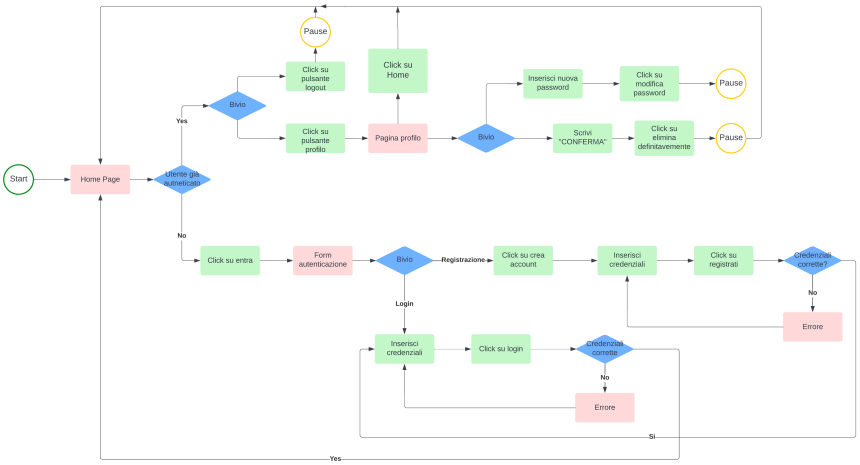
2 User Flows

In questa sezione del documento di sviluppo vengono riportati gli user Flows. Lo scopo degli User Flows è quello di poter specificare le azioni disponibili all'utente e le conseguenza di esse. Sono stati individuati 2 tipi di User flow, uno per tutto ciò che riguarda l'autenticazione e il profilo utente, e un'altro che riguarda le azioni disponibili ai diversi ruoli di utente. Teniamo a ricordare che tutte le immagini sono disponibili ad alta risoluzione nell'appropriata cartella del D4. Di seguito esponiamo la legenda per i simboli utilizzati



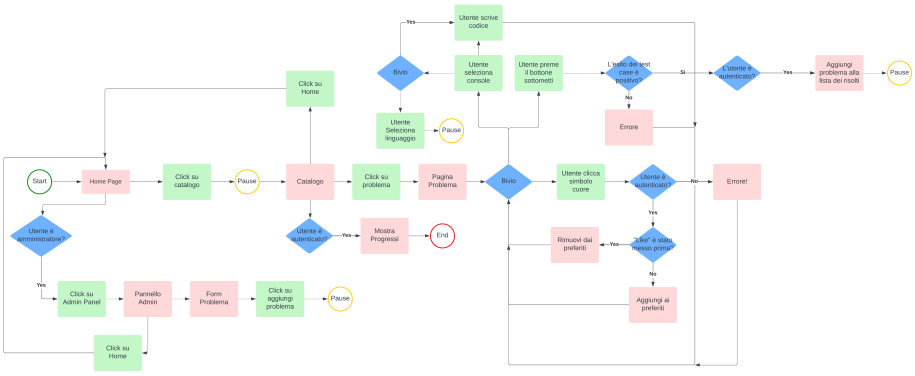
2.1 Azioni riguardanti l'autenticazione

Questo User Flow è specifico per tutte le azioni che riguardo l'autenticazion e ciò che fa parte di essa. Si ricorda che in ogni momento della navigazione l'utente è in grado di poter autenticarsi,tornare alla home e al catalogo tramite una apposita Navbar che è presente in ogni pagina del sito web. Parte di queste interazioni sono state rimosse per rendere l'User Flow pià leggibile.



2.2 Azioni riguardanti l'utilizzo del sito

Questo User Flow è specifico per tutte le azioni che ogni tipo di utente può intraprendere nell'utilizzo del sito, teniamo a precisare che la funzione di aggiunta di un problema tramite DB non è stata sviluppata, al momento il form esiste ma non dialoga col database, ci scusiamo per l'incovenienza. Ricordiamo che tramite Navbar l'utente è in grado di intraprendere tutte le azioni descritte nell'User Flow precedente.

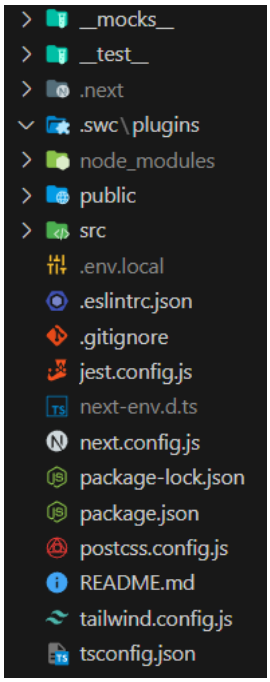


3 Documentazione e implementazione dell'applicazione

Nella precedente sezione abbiamo illustrato tutte le funzioni attualmente implementate nell'applicazione e un'idea di come l'utente può interagire con esse. L'applicazione **SleepCode** è stata sviluppata utilizzando Next.js vers. 14.0.3, un framework Javascript free-open source basato su React

3.1 Struttura del Progetto

Il software utilizzato per version control utilizzato è Git, come repository abbiamo utilizzato Github, il codice e la sua history è presente su una repository del membro del Team Raffaele Castagna, l'ultima versione stabile e quella utilizzata per hostare il sito è disponibile presso la repository CodeBase, all'interno di essa, troveremo le seguenti cartelle:



Ricordiamo che le cartelle **.next, .swc, .env** insieme a **package-lock.json, next-env.d.ts, eslintrc.json** sono state generate automaticamente

3.1.1 Directory: __mock__

Questa cartella contine delle funzioni "mock" che permettono a Jest di poter effetuare il testing senza fare chiamate dirette al database.

3.1.2 Directory: __test__

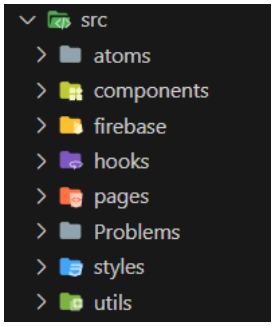
Questa cartella contiene tutti i test case e test suite dedicate al testing.

3.1.3 Directory: public

Questa cartella contiente tutte le immagini utilizzate all'interno del sito.

3.1.4 Directory: src

Questa cartella contiente tutte le parti del progetto sia front-end che back-end del progetto, procediamo con una vista più dettagliata



- **Atoms**: Abbiamo utilizzato una libreria di State management sviluppata da Google per tenere traccia dello stato dell'utente, la libreria utilizzata è: Recoil, gli atoms non sono altro che gli stati di certi componenti del sito.
- **Components**: Questa cartella contiene tutti i vari componenti del sito che vengono utilizzati dalle pagine, essi possono essere semplici Scheletri o componenti di maggior importanza.
- **Firebase**: Questa cartella contiene tutti i file relativi al setup di Firebase, il database utilizzato per il progetto.
- **Hooks**: Questa cartella contiene gli hooks sviluppati durante il progetto per compiere una funzione.
- **Pages**: Questa Cartella contiente le varie pagine le loro **routes** e le **API**.
- **Problems**: Questa cartella contiene il tipo di dato generico per i problemi.

- **Styles**: Questa cartella contiene diversi stili pre-impostati utilizzati assieme a TailwindCSS
- **Utils**: Questa cartella contiene i diversi testi e informazioni relative ai problemi attualmente disponibili, oltre che a form validators e funzioni comuni.

3.1.5 .env.local

Questa file contiene tutte le **variabili locali**, utilizzate per la connessione al database e necessarie per il corretto funzionamento dell'applicativo.

3.1.6 .gitignore

Questa cartella specifica quali file **git** non deve includere nelle varie pull/push requests. (.env.local è più importante in quanto contiente la **Chiave segreta**)

3.1.7 jest.config.js

Questo file contiene la configurazione di **Jest**, libreria utilizzata nel testing.

3.1.8 package.json

Questo file contiene le **dependency** del framework

3.1.9 README.md

Questo file contiene informazioni generali sul progetto.

3.1.10 postcss.config.js

Questo file è stato auto-generato da TailwindCSS

3.1.11 tailwind.config.js

Questo file contiene **pallet** di colori utilizzati da TailwindCSS

3.1.12 tsconfig.json

Questo file contiene regole utilizzate da ESLint un **patter checker** utilizzato durante lo sviluppo

3.2 Dipendenze del progetto

Il progetto utilizza diverse librerie, procediamo ad elencarne le più importanti e spiegare il loro funzionamento.

- **CodeMirror**: Utilizzata nel front-end avere uno stile simile a Vs-code durante la scrittura del codice.
- **Split**: Utilizzata nel front-end per rendere la pagina dei problemi dinamica (L'utente è in grado di modificare la grandezza dei componenti).
- **Toastify**: Libreria che offre componenti UI utilizzati per dialogare con l'utente.
- **Recoil**: Libreria di State Management sviluppata da Google.
- **Librerie fornite da Firebase**: Abbiamo utilizzato diverse librerie fornite da firebase come **auth,sdk,admin sdk** e molte altre.
- **Yup**: Utilizzato per la validazione RegEx di dati inseriti
- **ESLint**: Pattern checker utilizzato durante lo sviluppo per ottenere un codice di alta qualità
- **Jest**: Libreria utilizzata per il testing
- **node-mocks-http**: Libreria utilizzata per mandare richieste API finte durante il testing.

Oltre a queste abbia altre dipendenze minore tra vari **hooks** e pacchetti necessari per altri pacchetti inutili da elencare.

3.3 Dati del Progetti e Database