Dipartimento di Ingegneria e Sceienza dell'Informazione

Sleep Code

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

Analisi dei Requisiti

Descrizione: documento di analisi dei requisiti funzionali, non funzionali, front-end e back-end.

Numero documento: D1

Membri del gruppo:
Raffaele Castagna
Alberto Rovesti
Zeno Saletti

Numero gruppo: G17

Ultima revisione: 29 settembre 2023

Indice

1 Introduzione e scopi

1.1 Scopo del documento

Le informazioni contenute in questo documento concorrono ad esporre l'analisi dei requisiti relativa al progetto *SleepCode*. In particolare, dopo aver specificato gli obiettivi e gli attori coinvolti—utenti finali e utilizzatori del frutto di questo progetto—verranno definiti i requisiti funzionali e non funzionali; verrà presentata una proposta di design di back-end; infine saranno riportati i servizi di back-end.

1.2 Obiettivo del progetto

Il progetto proposto si prefigge, come scopo fondante, di fornire alla comunità di giovani informatici un servizio online di *esercitazione* e di *raccolta* di problemi mirati alla programmazione e alla progettazione di piccoli algoritmi risolutivi, mediante la scrittura di codice.

1.3 Attori coinvolti ed esigenze

Per comprendere meglio i requisiti che verranno descritti in seguito (in particolar modo quelli funzionali), è innanzitutto essenziale specificare il pubblico, insieme alle loro potenziali esigenze, al quale il servizio intende rivolgersi. Tale servizio vuole rendersi utile soprattutto a coloro che sono coinvolti in percorsi di studio attinenti all'ambito informatico, ma specialmente anche a chiunque desideri cimentarsi nella risoluzione di piccoli problemi di programmazione; pertanto ci si aspetta che chiunque desideri usufruire del servizio possieda almeno le conoscenze basilari della programmazione. Esempi di queste nozioni pregresse, che tuttavia non devono necessariamente essere ampie e approfondite per utilizzare il servizio[1], sono: cosa si intende per algoritmo e linguaggio di programmazione, familiarità nell'uso di qualche linguaggio di programmazione, tipi e strutture di dati più comuni.

Di fatto, il progetto che verrà sviluppato ha come scopo principale di creare una piattaforma accessibile a singoli utilizzatori che desiderano esercitarsi, valutare e approfondire le personali conoscenze e abilità di *problem solving* legate alla programmazione. D'ora in avanti, in questo e nei successivi documenti, questo pubblico di individui appena descritti verrà indicato con il termine *utenti*.

[1] Gli utenti più esperti possono indubbiamente trarre vantaggio dal loro bagaglio culturale per approcciarsi con maggior facilità al servizio.

2 Requisiti funzionali

Vengono di seguito elencati i principali requisiti funzionali (RF) del progetto. Essi sono organizzati secondo uno schema che segue gli obiettivi elencati nei paragrafi precedenti. Più in particolare, ogni sottosezione di questa parte del documento risponde a diversi scopi precedentemente accennati, suddividendo eventuali macro-funzioni in requisiti minori nel caso di obiettivi di più ampia portata.

2.1 Accesso

RF 1. Altro requisito funzionale

2.2 Consultazione dei problemi

RF 2. Consultazione del catalogo dei problemi: Il servizio deve mettere a disposizione un insieme di problemi sui quali l'utente possa esercitarsi. L'utente deve poter consultare un catalogo, atto a raccogliere i quesiti, e navigare al suo interno. Deve quindi essere possibile:

1. Visualizzare tale catalogo in una vista dedicata.
2. Cercare uno o più problemi specifici mediante ricerca filtrata per campi, specificati dai metadati elencati al RNF 2, oppure non filtrata.
3. Selezionare dal catalogo un problema specifico, cosicché la descrizione dell'intero problema possa essere visionata per mezzo delle funzionalità di cui al RF 3.

RF 3. Consultazione di un problema: Deve essere fornito un visualizzatore del documento contenente le informazioni del singolo problema precedentemente selezionato dal catalogo. La visualizzazione deve rendere disponibile alla vista dell'utente i dati relativi al problema e specificati nel RNF 1.

2.3 Esercitazione

RF 4. Avviare una sessione di esercitazione: L'utente deve poter selezionare, attraverso l'apposito catalogo, il problema desiderato avviando una sessione di esercitazione con lo scopo di risolverlo. A tal fine, l'utente deve poter:

1. Attivare, dopo averlo visualizzato, il problema scelto. L'attivazione permette di accedere alle funzionalità descritte nei prossimi punti, oltre a incrementare di una unità il numero di tentativi effettuati dall'utente su quel problema (a meno che il problema non sia già stato risolto in precedenza). L'avvio della sessione di esercitazione deve avvenire previa conferma da parte dell'utente.
2. Essere al corrente di quale linguaggio di programmazione sia attualmente attivo per la scrittura di codice, tramite un menu dedicato dal quale deve altresì essere possibile selezionare uno dei linguaggi disponibili[2].
3. Scrivere, sotto forma di codice nel linguaggio di programmazione scelto, l'algoritmo risolutivo del problema attualmente attivo. La scrittura deve poter essere effettuata in una vista o finestra apposita.

[2]Per approfondimenti sulla disponibilità dei linguaggi, si veda RNF 3.1.

RF 5. Correttezza sintattica del codice: L'utente deve poter verificare che il codice scritto sia sintatticamente corretto e pronto per l'esecuzione. Quindi devono essere messe a disposizione le seguenti funzionalità:

1. Compilazione del codice.

2. Visualizzazione di avvisi relativi a eventuali errori di sintassi *oppure* di compilazione andata a buon fine. In caso di errori di scrittura, l'utente deve poter correggere tali errori riscrivendo nell'area destinata al codice.

RF 6. Verifica della correttezza[3]: L'utente deve poter verificare la correttezza del codice scritto eseguendolo e testandolo:

1. Il codice deve essere eseguito sottoponendolo ad un certo numero di test cases (al minimo 3), cioè fornendo opportune istanze di input (Per esempio, se un problema richiede di sommare due numeri interi, il codice risolutivo proposto dall'utente verrà eseguito fornendo ad esso coppie di interi e registrando i risultati in output.)

2. L'utente deve poter verificare quali e quanti test cases sono andati a buon fine. Per ogni test case, solo le istanze in input potranno essere visualizzate. Inoltre, l'utente deve poter riscrivere e perfezionare l'algoritmo e sottoporre ripetutamente il codice ai test cases.

3. L'utente deve poter terminare la sessione in un tempo finito. Pertanto: nel caso in cui tutti i test cases sono andati a buon fine la sessione di esercitazione deve terminare e la carriera dell'utente deve essere aggiornata di conseguenza, quindi contrassegnando come *risolto* il problema corrente insieme al numero di tentativi impiegati fino all'istante della risoluzione; in caso contrario, l'utente deve avere la possibilità di rinunciare a fornire soluzioni al problema, terminando la sessione di esercitazione.

2.4 Gestione profilo

Si intende per *carriera* l'insieme dei dati estratti dai tentativi effettuati dal singolo utente durante le esercitazioni sui problemi. Tramite la definizione di questa carriera, realizzata per mezzo delle funzionalità che seguono, si raggiunge l'obiettivo dell'autovalutazione delle personali capacità dell'utente.

RF 7.

[3]La *correttezza* di cui si parla in questo caso riguarda solo l'efficacia risolutiva dell'algoritmo.

3 Requisiti non funzionali

Caratteristiche delle risorse

RNF 1. Struttura di un problema: Tutti i problemi forniti possiedono:

- Un titolo.
- Un testo, scritto prevalentemente in linguaggio naturale, che descrive uno scenario che richiede di essere risolto per mezzo di un algoritmo. Eventuali immagini e proposizioni matematiche formali possono accompagnare i testi.
- Almeno un esempio di input insieme al relativo output che mostra il risultato atteso.

RNF 2. Metadati dei problemi: Ogni problema è essenzialmente un'entità caratterizzata dai seguenti dati descrittivi:

- Nome identificativo: si tratta di una stringa alfanumerica, che riprende una o più parole chiave del titolo del problema di cui al RNF 1.
- Difficoltà: ogni problema possiede un'etichetta (*tag*) associata che ne valuta la difficolta in modo indicativo: bassa intermedia, alta.

3.1 Caratteristiche di sistema

RNF 3. Scalabilità: L'infrastruttura del servizio deve essere scalabile e aperta alle esigenze derivanti dall'aumento di nuovi utenti. Questo requisito è motivato dalla disponibilità online del servizio che verrà sviluppato. In particolare:

1. Data l'eterogeneità di linguaggi di programmazione esistenti al momento della stesura di questo documento, è importante che il servizio sia in grado di accogliere con l'avanzare del tempo codici scritti in linguaggi differenti.

RNF 4. Usabilità: Le funzionalità

RNF 5. Lingua di sistema: Il servizio sarà erogato in lingua italiana. Altrettanto sarà fatto per i testi dei problemi.

RNF 6. Prestazioni:

3.2 Privacy e sicurezza

RNF 7. Privacy: Il servizio deve essere progettato e realizzato in ottemperanza delle vigenti disposizioni di legge in materia di tutela della privacy e trattamento dei dati:

1. L'applicazione fornita dal servizio deve essere conforme al regolamento UE n.2016/679 per la protezione dei dati.

RNF 8. Sicurezza:

4 Design front-end

5 Design back-end