Documentazione

[1 Introduzione 2](#_Toc53059369)

[1.1 Informazioni sul progetto 2](#_Toc53059370)

[1.2 Abstract 2](#_Toc53059371)

[1.3 Scopo 2](#_Toc53059372)

[2 Analisi 3](#_Toc53059373)

[2.1 Analisi del dominio 3](#_Toc53059374)

[2.2 Analisi e specifica dei requisiti 3](#_Toc53059375)

[2.3 Use case 4](#_Toc53059376)

[2.4 Pianificazione 4](#_Toc53059377)

[2.5 Analisi dei mezzi 5](#_Toc53059378)

[2.5.1 Software 5](#_Toc53059379)

[2.5.2 Hardware 5](#_Toc53059380)

[3 Progettazione 6](#_Toc53059381)

[3.1 Design dell’architettura del sistema 6](#_Toc53059382)

[3.2 Design dell’interfaccia 6](#_Toc53059383)

[3.3 Design dei flussi principali 7](#_Toc53059384)

[4 Implementazione 9](#_Toc53059385)

[5 Test 9](#_Toc53059386)

[5.1 Protocollo di test 9](#_Toc53059387)

[5.2 Risultati test 12](#_Toc53059388)

[5.3 Mancanze/limitazioni conosciute 12](#_Toc53059389)

[6 Consuntivo 12](#_Toc53059390)

[7 Conclusioni 12](#_Toc53059391)

[7.1 Sviluppi futuri 12](#_Toc53059392)

[7.2 Considerazioni personali 12](#_Toc53059393)

[8 Bibliografia 12](#_Toc53059394)

[8.1 Bibliografia per articoli di riviste: 12](#_Toc53059395)

[8.2 Bibliografia per libri 12](#_Toc53059396)

[8.3 Sitografia 12](#_Toc53059397)

[9 Allegati 13](#_Toc53059398)

# Introduzione

## Informazioni sul progetto

Allievi coinvolti nel progetto: Michea Colautti

Classe: Informatica 3AC presso la sede Scuola Arti e Mestieri Trevano

Docenti responsabili: Geo Petrini

Data inizio: 01.09.2020  
Data consegna: 23.12.2020

## Abstract

Per questo progetto è stato richiesto di sviluppare un software in grado di generare dei labirinti 2d con svariate forme o opzioni configurabili. Grazie a questo utile e divertente software sarà quindi possibile mettersi alla prova con svariati labirinti di difficoltà disparate, che saranno risolvibile direttamente sul sito. Inoltre sarà possibile ricreare questi labirinti tramite i parametri oppure un seme di generazione. Sarà possibile pure il salvataggio dei labirinti tramite un’immagine e la possibilità di visualizzare la soluzione

## Scopo

L’obiettivo del progetto è lo sviluppo di un’applicazione per la generazione di labirinti 2D.

In questa generazione dovrà essere possibile configurare dei parametri come la forma esterna e del percorso, l’inserimento di percorsi a loop, le posizioni di partenza e fine, le dimensioni del labirinto, l possibilità di esportare la soluzione tramite un’immagine e la generazione di soluzioni multiple. l’utente

L’utente non deve solo avere la possibilità di salvare i parametri oppure un seed di generazione, ma deve essere anche in grado di completare il labirinto online, lasciando dietro di sé la “scia” con il percorso seguito

# Analisi

## Analisi del dominio

Per questo progetto è stato chiesto di sviluppare un software, web nel mio caso, che permetta di generare labirinti 2D. Gli utenti a cui esso sarà indirizzato non dovranno avere particolari conoscenze informatiche, l’interfaccia grafica semplice permetterà infatti a tutti gli utenti il libero.

I labirinti saranno generati in maniera randomica e anche se esistono diversi siti web capaci di generare enigmi simili pochi di questi combinano la grande personalizzazione del labirinto con la possibilità di completarlo comodamente online.

Non sarà necessario creare strati di sicurezza, in quanto il progetto non richiede un database e sarà un'unica pagina internet e il codice java script è comunque in chiaro.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-001** | |
| **Nome** | Home Page |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Un pagina html semplice in cui disporre i controlli per la personalizzazione del labirinto |

|  |  |
| --- | --- |
| **ID: REQ-002** | |
| **Nome** | Campi di controllo per la personalizzazione |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Bottoni e campi di testo per la personalizzazione del labirinto |

|  |  |
| --- | --- |
| **ID: REQ-003** | |
| **Nome** | Logica per la generazione del labirinto (partendo da un seed) |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Un algoritmo che generi il labirinto partendo da un numero casuale/inserito dall’utente |

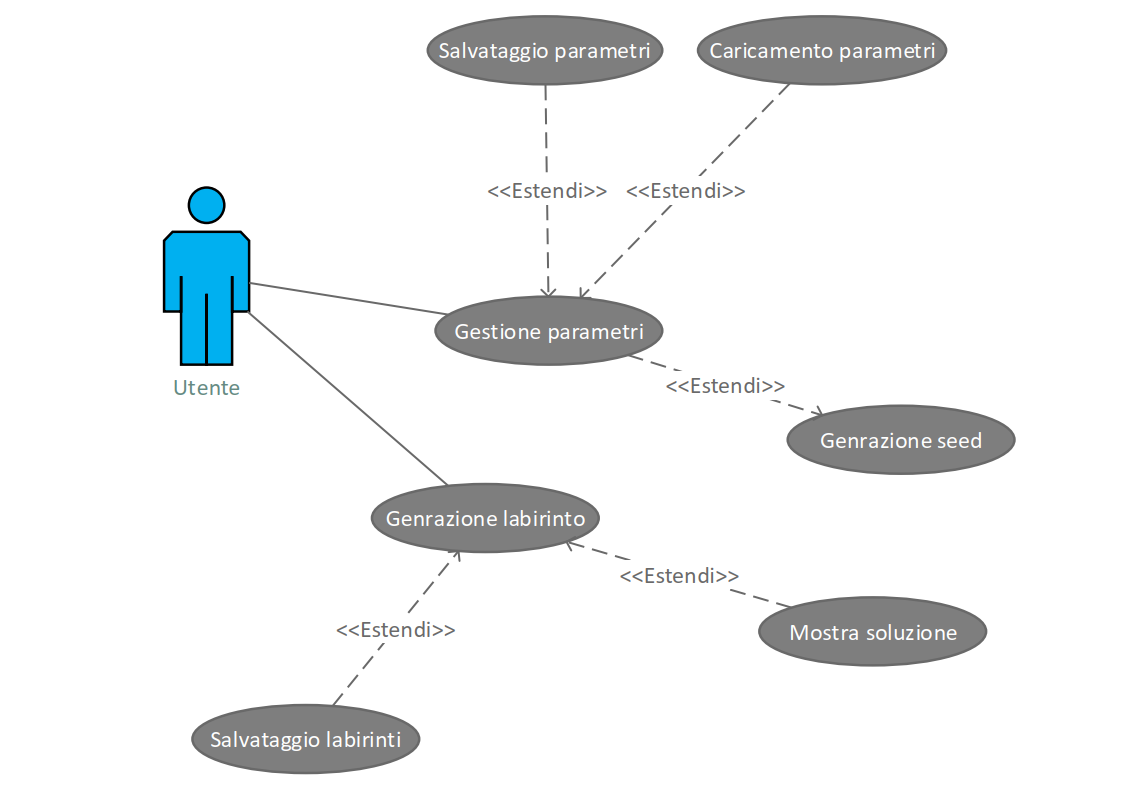
|  |  |
| --- | --- |
| **ID: REQ-004** | |
| **Nome** | Metodo che permetta il salvataggio del labirinto |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: REQ-005** | |
| **Nome** | Libreria (JQuery o simili) |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Per permettere all’utente di disegnare il labirinto |

|  |  |
| --- | --- |
| **ID: REQ-006** | |
| **Nome** | Sistema di salvataggio dei parametri di personalizzazione |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Si può utilizzare il metodo per salvare il labirinto per fare una foto ai parametri selezionati |

|  |  |
| --- | --- |
| **ID: REQ-007** | |
| **Nome** | Metodo per la conversione da caratteri ascii a muri |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Si può una canvas e un metodo dedicato trovato su internet |

## Use case



In questo digramma dell’use case possiamo osservare come l’utente accede al sito e imposta i parametri, dopodiché genera il suo labirinto. A questo punto può scegliere tra molte opzioni, tra cui mostrare la soluzione, salvare il labirinto oppure i suoi parametri. L’utente può anche caricare i parametri che creeranno il labirinto. Si noti che la generazione del seed avviene solamente se un seed non è stato immesso.

## Pianificazione

|  |
| --- |
| Figura 1: Gantt preventivo |

## Analisi dei mezzi

Per questo progetto ho deciso di utilizzare JavaScript. Questo perché è il linguaggio con cui mi trovo meglio e, soprattutto, mi permette di legare un’interfaccia grafica a del codice in maniera semplice e veloce. Inoltre avendo l’applicativo sul web posso renderlo compatibile con molte piattaforme e mi permette anche di utilizzare librerie esterne facilmente (JQuery per esempio).

Per lo sviluppo utilizzerò i computer della scuola e Visual Studio Code per la programmazione.

### Software

Windows 10

Visual Studio Code

Google Chrome, per testare il sito

JQuery per la traccia nel labirinto

<https://html2canvas.hertzen.com/>

mockflow.com. Per la progettazione dei flussi e della GUI

### Hardware

L’applicazione sarà eseguibile su un qualsiasi browser da PC o Mac, il browser con la migliore ottimizzazione sarà Google Chrome.

Durante lo sviluppo utilizzerò il pc messo a disposizione dalla scuola e Visual studio code per lo sviluppo.

# Progettazione

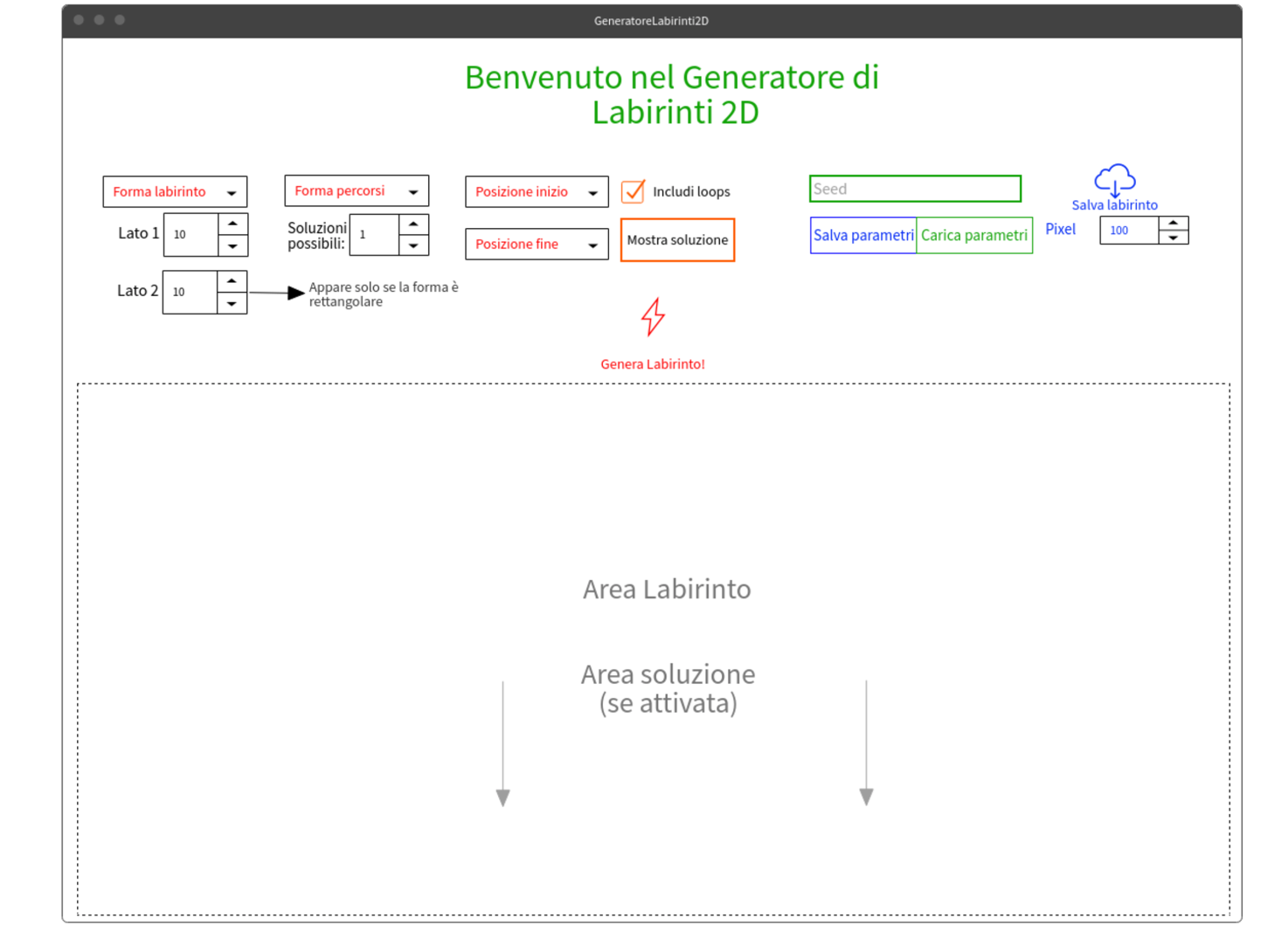
## Design dell’architettura del sistema

Quella che sto realizzando sarà un applicazione web stand-alone.

Pur cercando di abbellirla diversificando i colori la mia applicazione avrà una grafica estremamente semplice, e sarà tesata principalmente su Google Chrome, per quanto anche safari sarà in grado di supportarla.

Come detto anche prima l’app sarà sviluppata solo con Java script, con l’aiuto di librerie esterne come JQuery. Non prevedo di rendere l’applicazione responsive, sarà quindi difficile da utilizzare su un telefono ma se si poter l’opportunità in fase di implementazione farò in modo che il tutto sia responsive.

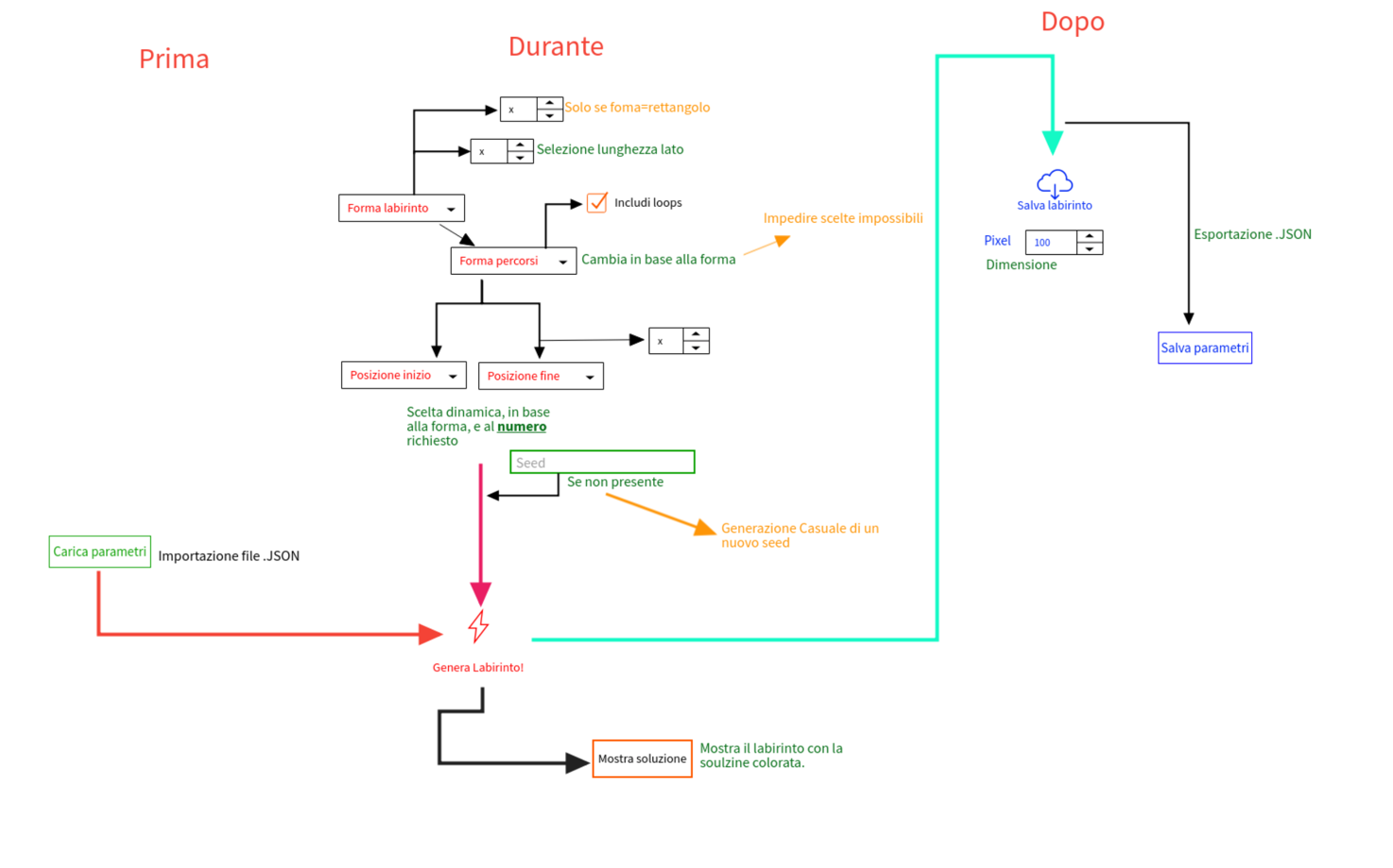
## Design dell’interfaccia



Qui sopra c’è una rappresentazione, creata grazie a MockFlow, di come apparirà la mia applicazione una volta aperta. Ho deciso di porre tutti i controlli nella parte superiore della pagina, quando un labirinto verrà generato l’area tratteggiata verrà riempita da esso.

Le frecce stanno a significare che, quando verrà richiesta la soluzione, essa apparirà sotto il labirinto già presente. Non l’ho rappresentato graficamente per non appesantire e rendere troppo grande inutilmente l’immagine. Il campo “Lato 2” è presente principalmente per rappresentare l’applicazione nella sua interezza ma, come spiegato meglio nel prossimo capitolo, per la maggior parte del tempo sarà nascosto.

## Design dei flussi principali



Qui sopra ho schematizzato quello che sarà il funzionamento generale della mia app.

Ho diviso questo schema in 3 sezioni, immaginando quello che potrebbe essere l’uso tipico.

Nella prima sezione, ho rappresentato la prima situazione possibile, ovvero quella in cui l’utente carica i parametri (scelte multiple e seed) tramite un file .json, dopodiché crea il labirinto.

Ora le scelte possibili sono 3, nessuna delle quali esclude l’altra.

La prima è quella in cui l’utente può decidere se salvare il labirinto, la seconda è se mostrare immediatamente la soluzione. La terza, non mostrata nel flusso, è quella in cui l’utente risolva il labirinto direttamente sulla pagina, trascinando il mouse.

Nella seconda sezione, è invece rappresentato quello che è il funzionamento dell’app nel caso in cui l’utente decida di creare un labirinto da 0. La struttura che ho scelto per rappresentare quelli che sono i vari controlli non è casuale, ho cercato di partire con le scelte più generiche, come la forma, fino a quelle più specifiche come le posizioni di inizio e fine. Cercherò di mantenere questa struttura anche nello sviluppo del codice.

Durante la scelta delle opzioni sono costretto a eseguire un controllo per evitare che l’utente faccia scelte irrealizzabili. Ad esempio, la forma del labirinto e la forma dei percorsi devono sempre coincidere, solo quando la forma è esagonale i percorsi possono essere esagonali e triangolari, e viceversa. Inoltre se la forma prescelta è quella rettangolare devo far apparire un campo numerico per scegliere la dimensione del secondo lato, altrimenti uscirebbero sempre labirinti quadrati. Questo aspetto è evidenziato anche nella GUI

Infine la terza sezione è comune ad entrambe le sezioni precedenti, l’unica cosa che permette di fare è infatti salvare i parametri e il labirinto, definendo la dimensione in pixel del png.

Si noti che, unendo con una freccia il pulsante “Salva parametri” con quello “carica parametri” si può vedere l’app come un ciclo. È anche questo che mi ha spinto al posizionare i due pulsanti vicini nella progettazione della GUI.

# Implementazione

In questo capitolo dovrà essere mostrato come è stato realizzato il lavoro. Questa parte può differenziarsi dalla progettazione in quanto il risultato ottenuto non per forza può essere come era stato progettato.

Sulla base di queste informazioni il lavoro svolto dovrà essere riproducibile.

In questa parte è richiesto l’inserimento di codice sorgente/print screen di maschere solamente per quei passaggi particolarmente significativi e/o critici.

Inoltre dovranno essere descritte eventuali varianti di soluzione o scelte di prodotti con motivazione delle scelte.

Non deve apparire nessuna forma di guida d’uso di librerie o di componenti utilizzati. Eventualmente questa va allegata.

Per eventuali dettagli si possono inserire riferimenti ai diari.

# Test

## Protocollo di test

In questo capitolo, basandomi sui requisiti da me definiti, ho cercato di elencare tutti i test che eseguirò, essi mi aiuteranno molto nel verificare il buon funzionamento di ogni parte del mio lavoro.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-002 | **Nome:** | Test sui campi |
| **Descrizione:** | Verificare che tutti I campi per il controllo utente non permettano scelte impossibili | | |
| **Prerequisiti:** | Campi per il controllo completati | | |
| **Procedura:** | 1. Selezionare l’opzione “rettangolare” come forma del labirinto 2. Verificare che l’unica opzione per la forma dei percorsi sia “rettangolari” 3. Verificare la presenza dei due campi numerici per i lati 4. Ripetere per tutte le forme a parte triangolare e esagonale 5. Se viene selezionato “triangolare” verificare la presenza dell’opzione “esagonali” per i percorsi e viceversa. | | |
| **Risultati attesi** | Scelte funzionanti | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-002  REQ-002 | **Nome:** | Test sull’integrità dei campi |
| **Descrizione:** | Verificare che non sia possibile lasciare campi non completati | | |
| **Prerequisiti:** | Campi per il controllo | | |
| **Procedura:** | 1. Completare tutti i campi (eccetto seed) e lasciarne uno vuoto 2. Provare a genare il labirinto 3. Ripetere operazione lasciando ogni volta un campo vuoto | | |
| **Risultati attesi** | Avviso che inviti l’utente a completare tutti i campi | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-003  REQ-003 | **Nome:** | Generazione seed |
| **Descrizione:** | Se il seed non è presente deve essere generato | | |
| **Prerequisiti:** | Campi per il controllo | | |
| **Procedura:** | 1. Completare tutti i campi (eccetto seed) 2. Provare a genare il labirinto | | |
| **Risultati attesi** | Campo seed deve essere riempito | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-004  REQ-003 | **Nome:** | Algoritmo di generazione |
| **Descrizione:** | L’algoritmo, basandosi sul seed e sui parametri, genera un labirinto consono alle scelte dell’utente | | |
| **Prerequisiti:** | Algoritmo di generazione completo, campi di controllo e seed | | |
| **Procedura:** | 1. Scegliere dei parametri 2. Provare a genare il labirinto 3. Ripetere con tutte le combinazioni possibili | | |
| **Risultati attesi** | Il labirinto deve attenersi alle istruzioni dell’utente | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-005  REQ-003 | **Nome:** | Soluzione |
| **Descrizione:** | Ad un click dell’utente, la soluzione viene mostrata | | |
| **Prerequisiti:** | Un labirinto generato | | |
| **Procedura:** | 1. Scegliere dei parametri 2. Provare a genare il labirinto 3. Generazione della soluzione 4. Ripetere con tutte le combinazioni possibili | | |
| **Risultati attesi** | Il programma deve mostrare una copia del labirinto con la soluzione in chiaro | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-006  REQ-006 | **Nome:** | Procedura di salvataggio/caricamento parametri |
| **Descrizione:** | Una volta generato un labirinto, al click dell’utente, il programma salva un file json con i parametri dell’utente | | |
| **Prerequisiti:** | Campi di controllo e seed completati | | |
| **Procedura:** | 1. Scegliere dei parametri 2. Provare a genare il labirinto 3. Provare a salvare i parametri 4. Eseguire un refresh della pagina 5. Caricare i parametri 6. Provare a genare il labirinto | | |
| **Risultati attesi** | 1. I parametri devono essere effettivamente salvati 2. I parametri devono essere effettivamente caricati 3. Il due labirinti devono essere identici | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-007  REQ-004 | **Nome:** | Procedura di salvataggio labirinto |
| **Descrizione:** | Una volta generato un labirinto, al click dell’utente, il programma salva un png con il labirinto | | |
| **Prerequisiti:** | Un labirinto generato, html2Canvas | | |
| **Procedura:** | 1. Provare a genare il labirinto 2. Scegliere la dimensione della foto 3. Provare a salvare il labirinto | | |
| **Risultati attesi** | La foto salvata è delle dimensioni specificate dall’utente. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-008  REQ-005 | **Nome:** | Disegno sul labirinto |
| **Descrizione:** | Una volta generato un labirinto, l’utente ha la possibilità di tracciare un percorso sopra la labirinto appena generato | | |
| **Prerequisiti:** | Un labirinto generato, mouseListener | | |
| **Procedura:** | 1. Provare a genare il labirinto 2. Provare a tracciare un percorso tenendo il mouse premuto | | |
| **Risultati attesi** | Il percorso viene tracciato. | | |

## Risultati test

Tabella riassuntiva in cui si inseriscono i test riusciti e non del prodotto finale. Se un test non riesce e viene corretto l’errore, questo dovrà risultare nel documento finale come riuscito (la procedura della correzione apparirà nel diario), altrimenti dovrà essere descritto l’errore con eventuali ipotesi di correzione.

## Mancanze/limitazioni conosciute

Descrizione con motivazione di eventuali elementi mancanti o non completamente implementati, al di fuori dei test case. Non devono essere riportati gli errori e i problemi riscontrati e poi risolti durante il progetto.

# Consuntivo

Consuntivo del tempo di lavoro effettivo e considerazioni riguardo le differenze rispetto alla pianificazione (cap 1.7) (ad esempio Gannt consuntivo).

# Conclusioni

Quali sono le implicazioni della mia soluzione? Che impatto avrà? Cambierà il mondo? È un successo importante? È solo un’aggiunta marginale o è semplicemente servita per scoprire che questo percorso è stato una perdita di tempo? I risultati ottenuti sono generali, facilmente generalizzabili o sono specifici di un caso particolare? Ecc.

## Sviluppi futuri

Migliorie o estensioni che possono essere sviluppate sul prodotto.

## Considerazioni personali

Cosa ho imparato in questo progetto? ecc

# Bibliografia

## Bibliografia per articoli di riviste:

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo dell’articolo (tra virgolette),
3. Titolo della rivista (in italico),
4. Anno e numero
5. Pagina iniziale dell’articolo,

## Bibliografia per libri

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo del libro (in italico),
3. ev. Numero di edizione,
4. Nome dell’editore,
5. Anno di pubblicazione,
6. ISBN.

## Sitografia

1. URL del sito (se troppo lungo solo dominio, evt completo nel diario),
2. Eventuale titolo della pagina (in italico),
3. Data di consultazione (GG-MM-AAAA).

**Esempio:**

* http://standards.ieee.org/guides/style/section7.html, *IEEE Standards Style Manual*, 07-06-2008.

# Allegati

Elenco degli allegati, esempio:

* Diari di lavoro
* Codici sorgente/documentazione macchine virtuali
* Istruzioni di installazione del prodotto (con credenziali di accesso) e/o di eventuali prodotti terzi
* Documentazione di prodotti di terzi
* Eventuali guide utente / Manuali di utilizzo
* Mandato e/o Qdc
* Prodotto
* …