Documentazione Enea Corti

1 Introduzione 3

1.1 Informazioni sul progetto 3

1.2 Abstract 3

1.3 Scopo 3

2 Analisi 4

2.1 Analisi del dominio 4

2.2 Analisi e specifica dei requisiti 4

2.2.1 Spiegazione elementi tabella dei requisiti: 5

2.3 Use case 5

2.4 Pianificazione 5

2.5 Analisi dei mezzi 5

2.5.1 Software 6

2.5.2 Hardware 6

3 Progettazione 6

3.1 Design dell’architettura del sistema 6

3.2 Design dei dati e database 6

3.3 Design delle interfacce 6

3.4 Design procedurale 6

4 Implementazione 7

5 Test 7

5.1 Protocollo di test 7

5.2 Risultati test 8

5.3 Mancanze/limitazioni conosciute 8

6 Consuntivo 8

7 Conclusioni 8

7.1 Sviluppi futuri 8

7.2 Considerazioni personali 8

8 Glossario 8

9 Bibliografia 9

9.1 Bibliografia per articoli di riviste: 9

9.2 Bibliografia per libri 9

9.3 Sitografia 9

10 Allegati 9

# Introduzione

## Informazioni sul progetto

Progetto sviluppato dall’allievo Enea Corti sotto la supervisione di Geo Petrini nel periodo di tempo che va dal 01.09.2023 al 01.12.2023 ed andrà presentato in data 15.12.2023.

## Abstract

È una breve e accurata rappresentazione dei contenuti di un documento, senza notazioni critiche o valutazioni. Lo scopo di un abstract efficace dovrebbe essere quello di far conoscere all’utente il contenuto di base di un documento e metterlo nella condizione di decidere se risponde ai suoi interessi e se è opportuno il ricorso al documento originale.

Può contenere alcuni o tutti gli elementi seguenti:

* **Background/Situazione iniziale**
* **Descrizione del problema e motivazione**: Che problema ho cercato di risolvere? Questa sezione dovrebbe includere l'importanza del vostro lavoro, la difficoltà dell'area e l'effetto che potrebbe avere se portato a termine con successo.
* **Approccio/Metodi**: Come ho ottenuto dei progressi? Come ho risolto il problema (tecniche…)? Quale è stata l’entità del mio lavoro? Che fattori importanti controllo, ignoro o misuro?
* **Risultati**: Quale è la risposta? Quali sono i risultati? Quanto è più veloce, più sicuro, più economico o in qualche altro aspetto migliore di altri prodotti/soluzioni?

Esempio di abstract:

*As the size and complexity of today’s most modern computer chips increase, new techniques must be developed to effectively design and create Very Large-Scale Integration chips quickly. For this project, a new type of hardware compiler is created. This hardware compiler will read a C++ program, and physically design a suitable microprocessor intended for running that specific program. With this new and powerful compiler, it is possible to design anything from a small adder, to a microprocessor with millions of transistors. Designing new computer chips, such as the Pentium 4, can require dozens of engineers and months of time. With the help of this compiler, a single person could design such a large-scale microprocessor in just weeks.*

## Scopo

Lo scopo del progetto (scopi didattici/scopi operativi). Dovrebbe descrivere il mandato, ma non vanno ricopiate le informazioni del quaderno dei compiti (che va invece allegato).

# Analisi

## Analisi del dominio

Il gioco potrà essere utilizzato da tutte le fasce di età, comprende la possibilità di scegliere il livello, se bambino oppure adulto e dà la possibilità di scegliere tra 3 grandezze di tabella.

Non occorre una particolare conoscenza di base per poter utilizzare il prodotto se non sapere come funziona il gioco, quindi quali incroci posso fare, in che direzione, eccetera.

Esistono molti atre pagine che eseguono la stessa operazione, questo progetto non vuole essere un affronto verso questi ultimi.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-01** | |
| **Nome** | Interfaccia |
| **Priorità** |  |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | Solo creazione tabella e non gioco |
| **002** | Presenza del bordo intorno al rettangolo |
| **003** | Multicolonna x le parole |
| **004** | Nessuna immagine all’interno del rettangolo |

|  |  |
| --- | --- |
| **ID: REQ-02** | |
| **Nome** | Input dell’utente |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | Possibilità di immettere un dizionario a scelta al posto del def. |
| **002** | Possibilità di scegliere la grandezza della tabella (piccola, media, grande) |
| **003** | Possibilità di scegliere livello (bambini o adulti) |
| **004** | Deve essere possibile scegliere il font (default Monospace) |

|  |  |
| --- | --- |
| **ID: REQ-03** | |
| **Nome** | Livello |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | Per i bambini generare solo griglia con parole da trovare |
| **002** | Per gli adulti generare anche par  ola finale da trovare |

|  |  |
| --- | --- |
| **ID: REQ-04** | |
| **Nome** | Export |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | La stampa è pensata per un foglio A4 |
| **002** | Formato PNG o HTML |

|  |  |
| --- | --- |
| **ID: REQ-05** | |
| **Nome** | Multipiattaforma |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | Deve essere utilizzabile in multipiattaforma |

|  |  |
| --- | --- |
| **ID: REQ-06** | |
| **Nome** | Formattazione |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | Tutte le lettere devono essere in maiuscolo |
| **002** | Nessun accento o apostrofo |
| **003** | Le parole devono essere in ordine alfabetico |

|  |  |
| --- | --- |
| **ID: REQ-07** | |
| **Nome** | Scrittura |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | Le parole saranno scritte in tutte le direzioni |

|  |  |
| --- | --- |
| **ID: REQ-08** | |
| **Nome** | Sovrapposizione |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |
| **Sotto requisiti** | |
| **001** | Le parole possono essere sovrapposte solo se in direzioni diverse |

## Use case

## Pianificazione

Gantt della pianificazione:

|  |
| --- |
| Figura 1:diagramma di Gantt. |

## Analisi dei mezzi

Elencare e descrivere i mezzi disponibili per la realizzazione del progetto. Ricordarsi di sempre descrivere nel dettaglio le versioni e il modello di riferimento.

### Software

Visual studio Code 1.78.2

HTML 5

Javascript 1.5

### Hardware

1 pc con i seguenti componenti:

* CPU 🡪 Intel i7-9700
* Memoria 🡪 32 GB RAM
* Scheda grafica 🡪 NVIDIA GeForce RTX 2060
* SSD 🡪 512 GB

# Progettazione

Questo capitolo descrive esaustivamente come deve essere realizzato il prodotto fin nei suoi dettagli. Una buona progettazione permette all’esecutore di evitare fraintendimenti e imprecisioni nell’implementazione del prodotto.

## Design dell’architettura del sistema

Descrive:

* La struttura del programma/sistema lo schema di rete...
* Gli oggetti/moduli/componenti che lo compongono.
* I flussi di informazione in ingresso ed in uscita e le relative elaborazioni. Può utilizzare *diagrammi di flusso dei dati* (DFD).
* Eventuale sitemap

## Design dei dati e database

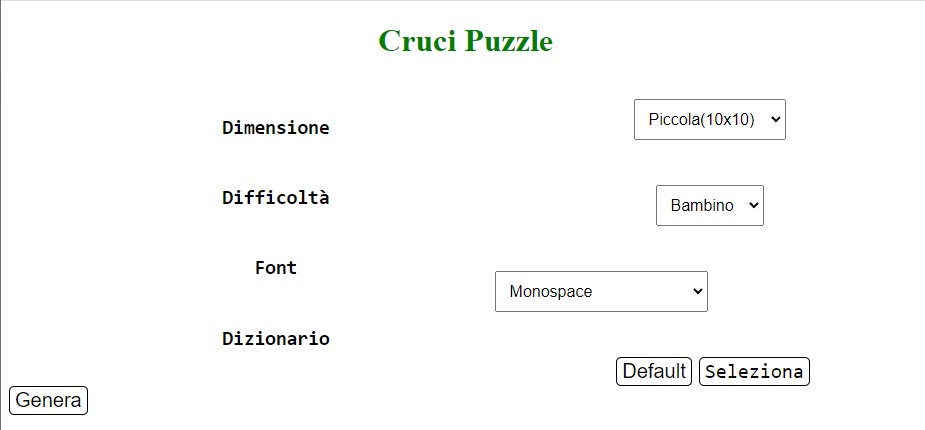
Descrizione delle strutture di dati utilizzate dal programma in base agli attributi e le relazioni degli oggetti in uso.

Schema E-R, schema logico e descrizione.

Se il diagramma E-R viene modificato, sulla doc dovrà apparire l’ultima versione, mentre le vecchie saranno sui diari.

## Design delle interfacce

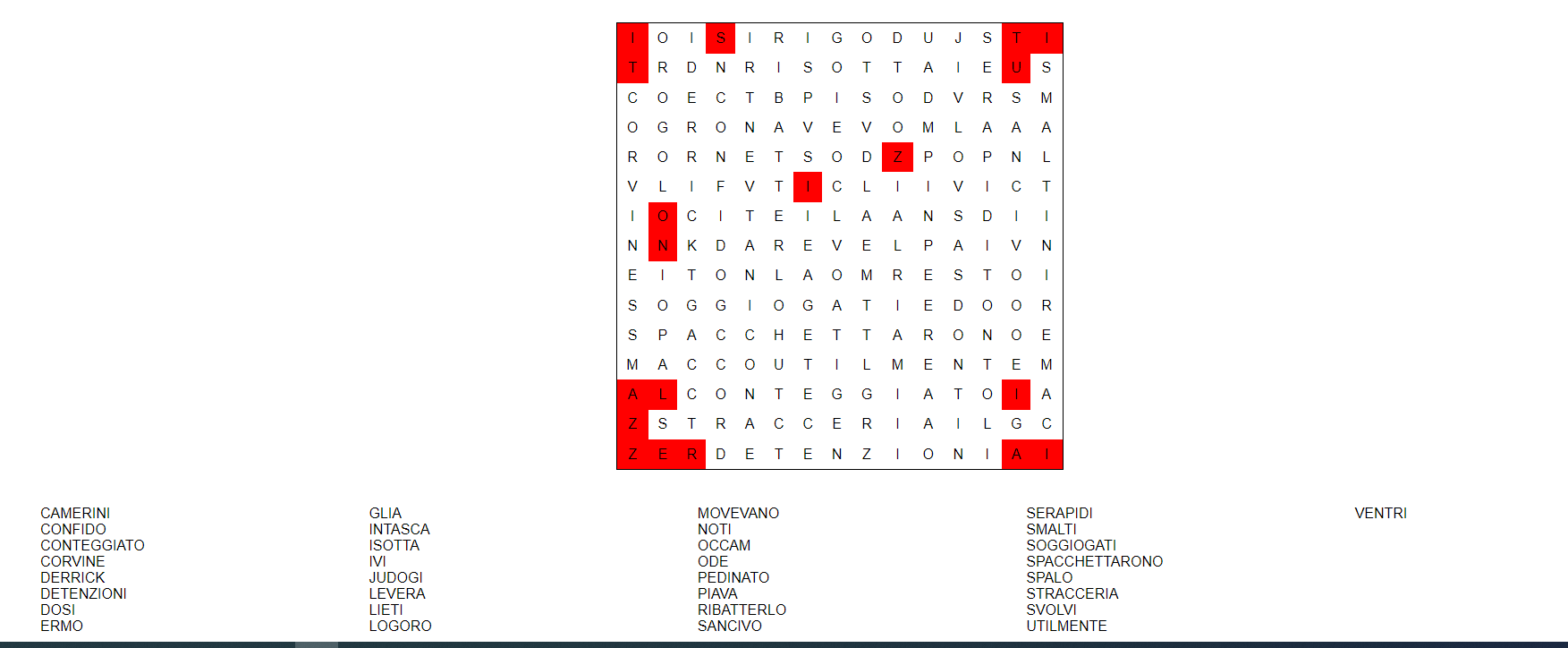
### Design iniziale:



La primissima cosa che mostro all’utente quando apre la pagina è questa, dove può scegliere la dimensione del campo di gioco, la difficoltà, il font da usare sia per le lettere della tabella che per le parole sottostanti.

Inoltre può scegliere se usare il dizionario di default oppure selezionarne uno personale con le parole scelte da lui.

### Design tabella:

****

Il design della tabella e delle parole sotto è semplice;

La tabella è posizionata al centro e al di sotto le parole che si devono trovare sono suddivise in colonne.

Il numero di colonne dipende dalla dimensione della tabella(ovvero del numero di parole da trovare).

## Design procedurale

Descrive i concetti dettagliati dell’architettura/sviluppo utilizzando ad esempio:

* Diagrammi di flusso e Nassi.
* Tabelle.
* Classi e metodi.
* Tabelle di routing
* Diritti di accesso a condivisioni …

Questi documenti permetteranno di rappresentare i dettagli procedurali per la realizzazione del prodotto.

# Implementazione

## Variabili globali:

* listaParole 🡪 verrà riempita con tutte le parole presenti nel file passato come dizionario
* listaParoleNascosta 🡪 verrà riempita con tutte le parole presenti nel file locale delle parole nascoste
* parolaRan 🡪gli verrà assegnato il valore temporaneo della parola che viene pescata
* x 🡪 coordinata per le righe dell’array
* y 🡪 coordinata per le colonne dell’array
* dimensione 🡪 dimensione scelta per la tabella
* array 🡪 array che verrà riempito con le lettere che formeranno le parole da trovare
* arrayParole 🡪 verrà riempita con tutte le parole che sono state inserite nell’array
* pos 🡪 verrà usata come condizione per la durata del ciclo, stabilisce la fine della pesca delle parole
* pNascosta 🡪 verrà assegnata la parola nascosta pescata
* nParole 🡪 numero di parole che devono essere inserite nella tabella, il numero è il 15% di dimensione x dimensione
* direzione 🡪 usata per determinare che direzione avrà la parola con un random
* vuoti 🡪 verrà assegnato il numero di spazi vuoti che sono rimasti una volta inserite le parole
* indiceNascosta 🡪 usata per inserire la parola nascosta nell’array
* uguale 🡪 usata per determinare se sono state prese parole uguali
* arrayControllo 🡪 Usato per contare vuoti presenti una volta finito l’inserimento di parole
* dif 🡪 verrà assegnata la difficoltà scelta dall’utente
* coloriCelle 🡪 usata per stampare la soluzione con ogni parola colorata in modo diverso.

## Funzione GeneraArray:

La funzione generaArray è relativamente semplice, è il main di questo progetto.

L’array dir mi servirà poco dopo per scegliere una direzione casuale da assegnare alla parola.

Nel primo ciclo for inizializzo i due array che diventano bidimensionali, mi serviranno da matrice per il gioco.

Nel secondo ciclo invece, che finisce quando ho inserito nella matrice il numero di parole che mi servono.

Successivamente giro la metà delle parole, così che quando le inserisco ho anche quel tipo di direzione.

Pesco una direzione casuale e richiamo la funzione cordinate che mi serve per prendere la x e la y, se le cordinate sono entrambe positive o sono 0 le utilizzo.

## Funzione cordinate:

Questa funzione si occupa della scelta casuale di una di una direzione che avrà la parola, in base alla coordinata che è uscita(v = verticale, ob = obliquo, or = orizzontale, obSx = obliquo a sinistra) genero le x e le y che non vadano fuori dalla dimensione della tabella una volta incrementate.

**Direzione = V:**

In questo caso la y dove posiziono la prima lettera non deve essere più grande della (dimensione della griglia – lunghezza della parola), così da no fuoriuscire dalla tabella

Mentre la x ovviamente non deve essere un numero più grande della larghezza dell’array.

**Direzione = OR:**

In questo caso faccio il contrario che facevo per il verticale, la x non deve essere più grande della (dimensione della griglia – lunghezza della parola).  
Mentre la y non deve essere un numero maggiore della altezza dell’array.

**Direzione = OB oppure direzione = OBSX:**

In questo caso sia la x che la y non dovranno essere maggiori della(dimensione – lunghezza della parola), così mi garantisco di non fuoriuscire dalla tabella.

## Lettura file:

### Funzione fetchWords:

La funzione FetchWords mi serve per verificare la validità del file che ho passato(url), nel caso sia un file valido, lo trasformo in un array, altrimenti ritorno un errore.

### Funzione fileDefault:



La funzione fileDefault è molto semplice, assegna alla costante words tutte le parole che sono contenute nel file locale dizionario.txt, dopodiché alla variabile listaParole assegna il valore di words

Infine richiama la funzione generaArray.

### Funzione readFile:



La funzione readFile, come dice il nome, si occupa della lettura del file contenete le parole che saranno poi quelle da trovare nel gioco.

Per prima cosa il valore dell’input dizionario viene assegnato alla costante fileInput, successivamente assegno alla costante chiamata file il primo valore di fileInput, nel caso fosse ci fosse più di un input.

Controllo che file non sia nullo, in caso che abbia un valore, ovvero che sia stato inserito un file, alla costante words, grazie al richiamo della funzione fetchWords, assegno tutte le parole contenute nel file.

Nel caso in cui il file passato abbia effettivamente al suo interno delle parole, assegno alla variabile listaParole il contenuto di words, successivamente chiamo la funzione generaArray vista prima.

In caso che no venga passato nessun file, oppure il file passato sia vuoto, richiamo la funzione fileDefault.

## Riempimento Array:

### Funzione pescaParola:



Nella funzione pescaParola vado a leggere il file (il dizionario usato) e prendo una parola, per farlo utilizzo una variabile chiamata continua a cui assegno il valore true.

Successivamente creo un loop, alla variabile parolaRan prima assegno un numero casuale da 0 fino ad un massimo uguale alla grandezza della lista di parole, ovvero il numero di quante parole possibili ci sono.

Dopo uso sempre parolaRan come variabile, ma in questo caso le assegno una parola alla posizione del numero random generato poco prima.

Il ciclo for invece mi serve per passare tutto l’array di parole già inserite fino a questo punto, per controllare che non ce ne siano di uguali, in questo caso assegno alla variabile uguale il valore di false.

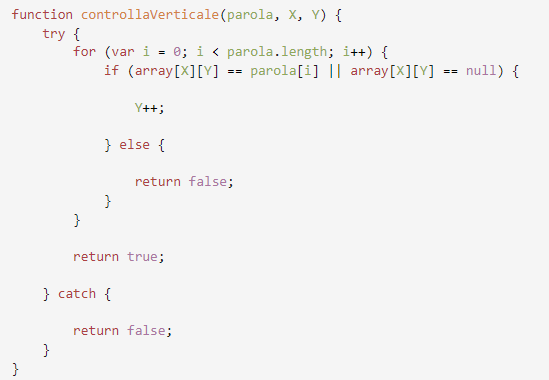
Per evitarmi di dover eseguire dopo controlli che mi rallentano il programma, prima id fare il ritorno controllo che la parola pescata abbia almeno tre caratteri, che la parola pescata abbia una lunghezza minore della dimensione e che non sia già stata pescata in precedenza.

Se tutti i requisiti sono come necessario, assegno a continua il valore di false, così che il ciclo finisce e la parola è stata presa.

### Funzioni di controllo inserimento:

Le mie funzioni di controllo per l’inserimento sono quattro, una per ogni possibile direzione che possono avere le parole quando le inserisco nell’array.

Un esempio di funzione di controllo che ho utilizzato:



In questo esempio mostro la funzione che mi serve per controllare la possibilità di inserimento per una parola in verticale.

Vengono passati tre valori; parola, X e Y.

Il programma prova a fare un ciclo di tutto l’array contenente le parole da trovare, e fa due controlli:  
Se alle coordinate dell’array si trova già la lettera della parola che si vorrebbe inserire, oppure a quelle coordinate la cella è vuota, procede con l’incremento delle Y in questo caso, siccome si tratta della funzione per il controllo verticale. Se alla fine del ciclo le condizioni sono state sempre rispettate ritorna true, mentre nel momento in cui una entrambe le due condizioni non sono rispettate si interrompe e ritorna false.

In questo caso viene incrementato solo Y per il controllo, ma come detto prima le funzioni di controllo sono quattro, nello specifico:

**ControllaOrizzontale**, con la stessa logica della funzione controllaVerticale spiegata in precedenza, l’unica differenza è che al posto di incrementare la Y per la verifica della possibilità di inserimento si incrementa X.

**ControllaObliquo,** anche questa funzione con la stessa logica delle precedenti, ma in questo caso entrambe le variabili X ed Y vengono incrementate per la verifica.

**ControllaObliquoSx,** stesso ragionamento e logica delle funzioni descritte sopra, anche in questo caso vengono toccate sia X che Y, ma con la differenza rispetto a prima che Y viene incrementata ad ogni ciclo, mentre X ad ogni ciclo viene decrementata.

### Funzioni di inserimento:

Anche in questo caso le funzioni sono quattro, come prima una per ogni direzione che posso avere, anche in

questo caso come esempio ho usato la funzione verticale:

Le funzioni di inserimento le ho sviluppate con la stessa logica di quelle di controllo:

Per prima cosa, quando viene chiamata la funzione, richiamo la funzione di controllo mostrata prima, gli passo parolRan, che sarebbe la parola che ho pescato come mostrato nella funzione pescaParola, mentre x ed y, che sono le coordinate generate prima con la funzione coordinate.

Richiamo la funzione randomRGBColor, che spiegherò dopo quando parlo della stampa della soluzione, per assegnare il colore.

Ciclo tutta la parola appena pescata, alle coordinate x ed y assegno la lettera corrispondente della parola da inserire.

In questo caso incremento la y, mentre negli altri casi:

**Orizzontale 🡪** Incremento la x

**Obliquo 🡪** Incremento sia x che y

**ObliquoSx 🡪** Incremento y e decremento x

# Test

## Protocollo di test

Definire in modo accurato tutti i test che devono essere realizzati per garantire l’adempimento delle richieste formulate nei requisiti. I test fungono da garanzia di qualità del prodotto. Ogni test deve essere ripetibile alle stesse condizioni.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-012 | **Nome:** | Import a card with KIC, KID and KIK keys, but not shown with the GUI |
| **Descrizione:** | Import a card with KIC, KID and KIK keys with no obfuscation, but not shown with the GUI | | |
| **Prerequisiti:** | Store on local PC: Profile\_1.2.001.xml (appendix n\_n) and Cards\_1.2.001.txt (appendix n\_n).  PIN (OTA\_VIEW\_PIN\_PUK\_KEY) and ADM (OTA\_VIEW\_ADM\_KEY) user right not set. | | |
| **Procedura:** | 1. Go to “Cards manager” menu,  in main page click “Import Profiles” link, Select the “1.2.001.xml” file, Import the Profile 2. Go to “Cards manager” menu,  in main page click “Import Cards” link, Select the “1.2.001.txt” file, Delete the cards,  Select the “1.2.001.txt” file, Import the cards 3. Research the “41795924770” Card, Click the imsi card link Check the card details 4. Execute the SQL: SELECT imsi, dir, keyset, cntr, rawtohex(kickey), rawtohex(kidkey), rawtohex(kikkey), rawtohex(chv), rawtohex(dap)FROM otacardkey a where imsi='340041795924770' ORDER BY keyset; | | |
| **Risultati attesi:** | Keys visible in the DB (OtaCardKey) but not visible in the GUI (Card details) | | |

## 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 Gantt 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.

# Glossario

Inserite una semplice tabella con due colonne che spieghi i termini specifici del progetto (lista dei termini in ordine alfabetico A-Z)

Esempio:

|  |  |
| --- | --- |
| **Termine** | **Descrizione** |
| AJAX | **Asynchronous JavaScript And XML**: una tecnica che permette di eseguire richieste ed ottenere dati da una pagina web in modo asincrono. |
| CSS | **Cascading Style Sheets**: linguaggio che permette di definire il layout e la grafica di una pagina web. |

# 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
* …