CPT Meeting Room Scheduler

1 Introduzione 3

1.1 Informazioni sul progetto 3

1.2 Abstract 3

1.3 Scopo 3

Analisi 4

1.4 Analisi del dominio 4

1.5 Analisi e specifica dei requisiti 4

1.6 Use case 6

1.7 Pianificazione 6

1.8 Analisi dei mezzi 6

1.8.1 Software 6

1.8.2 Hardware 6

2 Progettazione 7

2.1 Design dell’architettura del sistema 7

2.2 Design dei dati e database 7

2.3 Design delle interfacce 7

2.4 Design procedurale 7

3 Implementazione 8

4 Test 8

4.1 Protocollo di test 8

4.2 Risultati test 9

4.3 Mancanze/limitazioni conosciute 9

5 Consuntivo 9

6 Conclusioni 9

6.1 Sviluppi futuri 9

6.2 Considerazioni personali 9

7 Bibliografia 9

7.1 Bibliografia per articoli di riviste: 9

7.2 Bibliografia per libri 9

7.3 Sitografia 9

8 Allegati 10

# Introduzione

## Informazioni sul progetto

Allievo coinvolto: Luca Di Bello

Classe: Informatica 4AC presso la Scuola di Arti e Mestieri a Trevano

Docenti responsabili: Fabrizio Valsangiacomo

Data inizio: 03 / 09 / 2019

Data fine: 20 / 12 / 2019

## Abstract

E’ 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 di questo progetto è quello di implementare un sistema utile per gestire le prenotazioni dell’aula riunioni della sezione informatica. Il tutto sarà controllato da remoto tramite delle interfacce web. Verrà posto un monitor collegato ad un Raspberry Pi il quale permetterà di leggere quando la sala riunioni è occupata.

Questo progetto verrà anche utilizzato per prepararci all’esame di maturità che dovremo affrontare alla fine dell’anno.

## Analisi

## Analisi del dominio

Fino ad ora i professori utilizzavano un foglio di carta appeso alla porta con scritto chi, quando e per quanto tempo verrà utilizzata l’aula, dunque si cerca un modo per rendere tutto più chiaro e semplice utilizzando un sistema informatico.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-001** | |
| **Nome** | Creazione banca dati |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | - |
| **Sotto requisiti** | |
| **001** | Si necessita una tabella che tiene in memoria tutti gli utenti ed i permessi che hanno |
| **002** | Si necessita una tabella che tiene in memoria quali riservazioni sono state fatte |
| **003** | Il database dev’essere normalizzato fino al livello BCNF per semplificare modifiche future |
| **004** | Utente aggiuntivo con permessi limitati da utilizzare per leggere i dati dal sito web |

|  |  |
| --- | --- |
| **ID: REQ-002** | |
| **Nome** | Login con LDAP |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Si necessitano i permessi per accedere al servizio di *active directory* di scuola |
| **Sotto requisiti** | |
| **001** | Pagina web con form che permette all’utente di inserire le proprie credenziali, quindi e-mail e password |
| **002** | Soltanto i professori possono accedere al servizio (…@edu.ti.ch) |
| **003** | Pagina responsive e funzionante su qualunque dispositivo (mobile e non) |

|  |  |
| --- | --- |
| **ID: REQ-003** | |
| **Nome** | Permessi degli utenti |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | - |
| **Sotto requisiti** | |
| **001** | Livello utente normale: può vedere ed inserire una riservazione. Inoltre, può eliminare delle riservazioni fittizie |
| **002** | Livello utente avanzato: ha gli stessi permessi di un utente normale ma può inserire e cancellare riservazioni di un altro utente |
| **003** | Livello utente amministratore: ha tutti i premessi disponibili |

|  |  |
| --- | --- |
| **ID: REQ-004** | |
| **Nome** | Pagina gestione utenti |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | - |
| **Sotto requisiti** | |
| **001** | Si necessita una pagina web dove un utente amministratore può creare, cancellare modificare agli utenti |
| **002** | L’amministratore deve poter creare degli utenti normali ed utenti avanzati (con permessi aggiuntivi) assegnando una password di default. |
| **003** | Al primo login di un utente gli deve essere chiesto di cambiare la password di default con una personale |
| **004** | L’amministratore deve poter cambiare i permessi di un utente in ogni momento. |
| **005** | Alla creazione di un nuovo utente la password generata verrà inviata per e-mail all’utente in questione |
| **006** | Ci deve sempre essere almeno un utente amministratore. |
| **006** | Non deve essere possibile la registrazione di un utente con un’e-mail esterna a quella scolastica (gmail, Sunrise, ..) |

|  |  |
| --- | --- |
| **ID: REQ-005** | |
| **Nome** | Gestione prenotazioni |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | - |
| **Sotto requisiti** | |
| **001** | Deve essere presente un form che permette di aggiungere una nuova prenotazione. I campi necessari sono: data, l’orario di inizio e di fine. Anche il nome, il cognome e la mail di chi fa la riservazione devono essere mostrati. Dev’essere presente un campo utile per aggiungere delle osservazioni. |
| **002** | Controllo sulla data ed ora: non devono essere passati e non dev’esserci già una prenotazione |
| **003** | L’orario della prenotazione dev’essere a blocchi di 15 minuti |
| **004** | Alla creazione e cancellazione di una riservazione verrà inviata un’e-mail di conferma al relativo utente |

|  |  |
| --- | --- |
| **ID: REQ-006** | |
| **Nome** | Schermo per visualizzazione prenotazioni |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Verrà posto (tramite un supporto in alluminio) davanti all’aula riservata ai colloqui. |
| **Sotto requisiti** | |
| **001** | Non si dovrà vedere nessuna pagina web |
| **002** | Devono essere mostrate le date, l’orario di inizio e fine della riservazione, e le informazioni relative al docente che ha eseguito la riservazione (nome e cognome) |
| **003** | Deve mostrare le osservazioni se aggiunte dal docente durante la riservazione |
| **004** | Il sistema verrà controllato tramite un Raspberry Pi Model B fornito dalla scuola |

|  |  |
| --- | --- |
| **ID: REQ-007** | |
| **Nome** | Generazione di report |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | - |
| **Sotto requisiti** | |
| **001** | I report devono essere in formato PDF |
| **002** | I file di report devono essere generati automaticamente dalla pagina web |
| **003** | 3 tipi di report: giornalieri, settimanali e mensili. |

## Use case

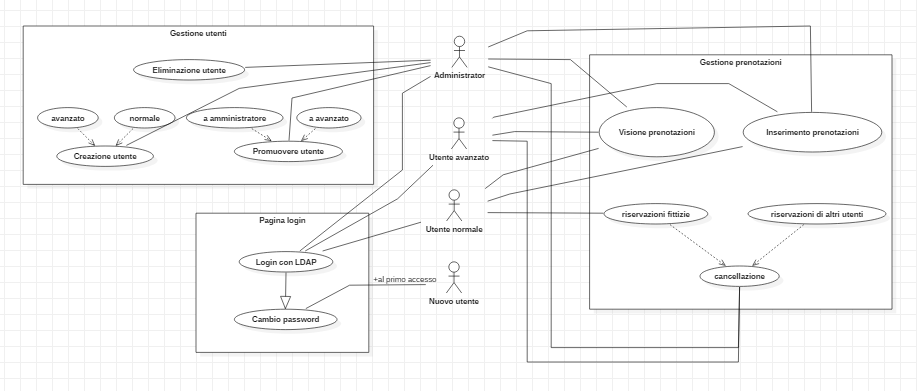


Figure 1 - Diagramma *Use Case*

Come si può vedere molto bene dall’immagine sovrastante ho deciso di dividere il diagramma in 3 macro-gruppi (gestione utenti, gestione prenotazioni e pagina login) per semplificare sia la lettura che la stesura del diagramma.

L’utente con i permessi di amministratore ha tutti i permessi sia nella gestione utenti sia nella gestione delle prenotazioni. L’utente avanzato invece ha tutti i permessi nella gestione delle prenotazioni però non può eliminare, creare o promuovere utenti.

L’utente normale invece può soltanto creare prenotazioni, visionare le prenotazioni fatte dagli altri professori ed eliminare le prenotazioni create da lui (non quelle di altri utenti).

Il “nuovo utente” invece come si può vedere, ha soltanto i permessi per cambiare la password. Questo perché al primo login di un utente il sistema richiede di cambiare la password di default con una password personale. Una volta cambiata la password l’utente viene promosso automaticamente ad “Utente normale”.

## Pianificazione

Prima di stabilire una pianificazione bisogna avere almeno una vaga idea del modello di sviluppo che si intende adottare. In questa sezione bisognerà inserire il modello concettuale di sviluppo che si seguirà durante il progetto. Gli elementi di riferimento per una buona pianificazione derivano da una scomposizione top-down della problematica del progetto.

La pianificazione può essere rappresentata mediante un diagramma di Gantt:

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

Se si usano altri metodi di pianificazione (es scrum), dovranno apparire in questo capitolo.

## Analisi dei mezzi

### Software

Per lo sviluppo di questo sistema ho utilizzato questi software:

* PHP 7.2.10: Linguaggio utilizzato per gestire la logica del sito da back-end
* Apache 2.4.27: Web server utilizzato per sviluppare il sito dalla mia macchina
* PhpMailer 6.0.7: Libreria PHP utilizzata per inviare mail agli utenti del sito.
* XDebug: Modulo aggiuntivo per PHP che fornisce di avere delle informazioni aggiuntive utili durante il debug
* PHPStorm
* Visual Studio Code

### Hardware

L’intero progetto è stato sviluppato sul mio laptop personale. Queste sono le specifiche:

* Nome: Asus ROG gl702vm
* Ram: 16GB
* CPU: Intel Core i7-7700HQ 2.80GHz
* Gpu: Nvidia GTX 1060 6GB
* OS: Windows 10 Home versione 1809

Per controllare il monitor invece ho utilizzato un Raspberry Pi Model B, queste sono le sue specifiche:

* Nome: Raspberry Pi Model B
* Ram: 1GB LPDDR2 (900 MHz)
* CPU: 4× ARM Cortex-A53, 1.2GHz
* OS: Raspbian

Come monitor ho utilizzato un HP Compaq LA1051g fornito dalla scuola.

# Progettazione

## Design dell’architettura del sistema

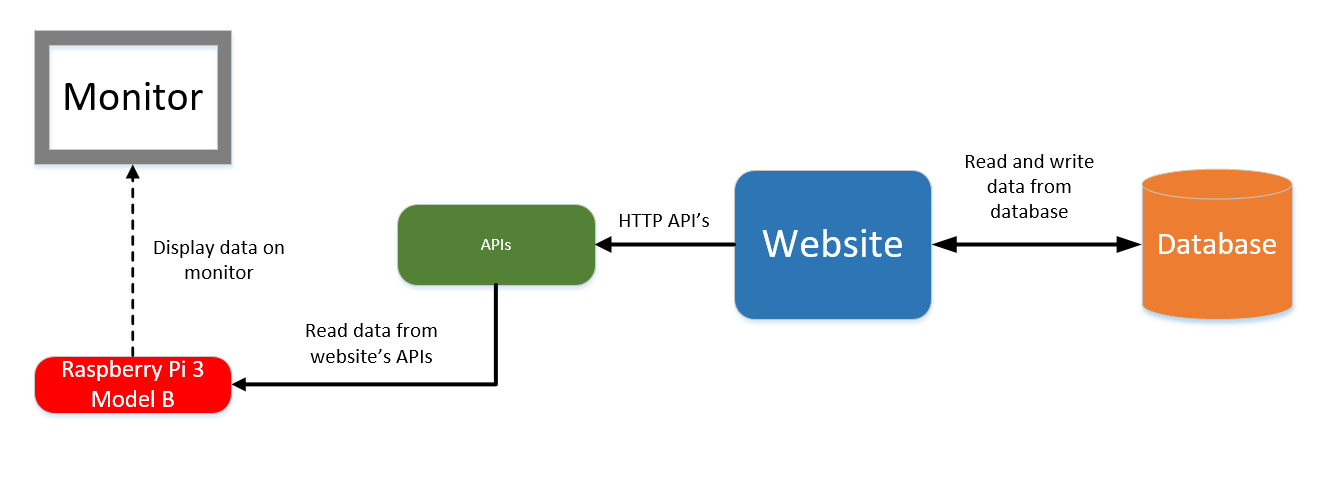


Figure 2 - Struttura del programma

Lo schema sovrastante è composto da 8 parti principali collegate tra loro in questo modo: Il sito web si interfaccerà con il database per la lettura e scrittura dei dati e fornirà delle API HTTP che verranno utilizzate dal Raspberry Pi per leggere i dati i quali verranno parsati (formattati) e mostrati all’utente finale tramite un Monitor.

## Design dei dati e database

Questo è il diagramma *entity relation* del database della mia applicazione:

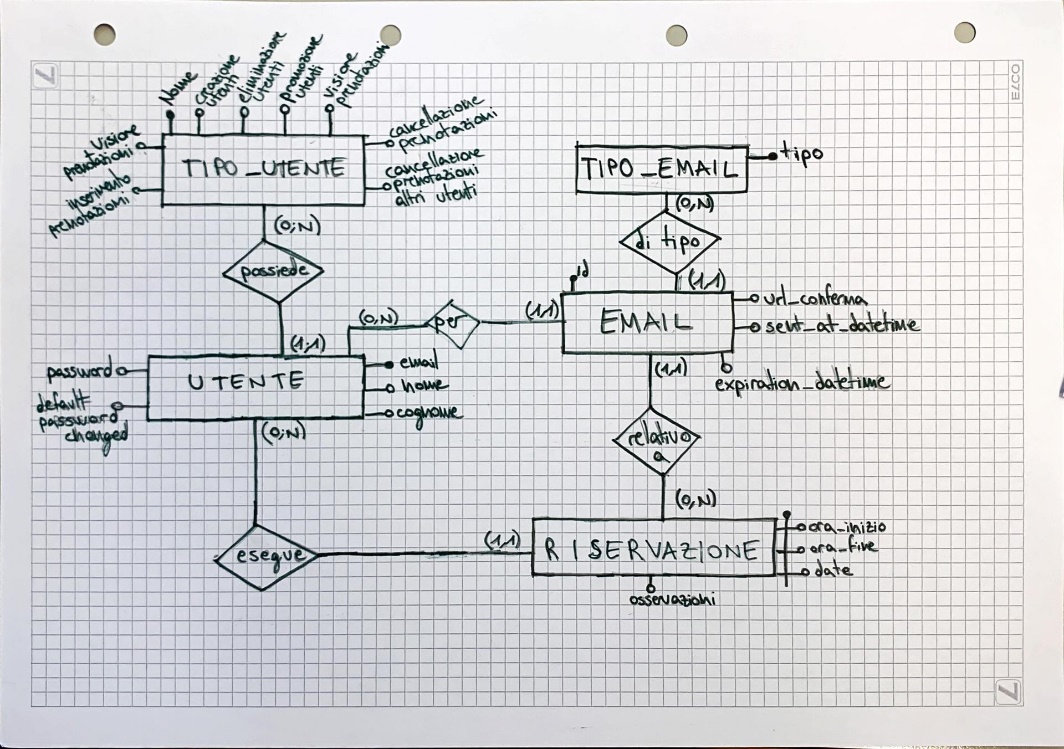


Figure 3 - Diagramma ER database

Lo schema è composto di 5 entità principali, di quali due non sono altro che generalizzazioni (*tipo\_utente* e *tipo\_email*).

L’entità *tipo\_utente* possiede molti attributi; ognuno di questi attributi è un permesso dell’utente. Ho deciso di farlo in questo modo per rendere l’applicazione “aggiornabile”: se in futuro si avrà la necessità di creare un nuovo tipo di utente in questa maniera non ci sarà la necessità di cambiare codice ma bisognerà soltanto aggiungere un record all’interno della tabella. Mi sono ispirato al funzionamento dei permessi di MySQL.

L’entità *utente* come si può intuire conterrà i dati dell’utente che utilizzerà l’applicazione tra questi anche il tipo di permessi che ha (indicato tramite l’associazione intitolata “*possiede”*). L’attributo *default\_password\_changed* servirà per tenere in memoria se l’utente ha cambiato la password al primo accesso o meno.

L’entità *riservazione* conterrà tutti i dati utili per l’identificazione della riservazione, quindi chi ha riservato e quando ha riservato.

L’entità *email* è collegata a ben 3 entità: utente, riservazione e *tipo\_email*. Questo perché un’e-mail è inviata ad un *utente* preciso nello specifico ed è riguardante una specifica *riservazione*. L’entità *tipo\_mail* è utilizzata per specificare se è un’e-mail di conferma di eliminazione o di creazione.

Gli attributi dell’entità *mail* servono per tenere in memoria l’url di conferma e il tempo di vita dell’url (24 ore solitamente).

### Schema logico

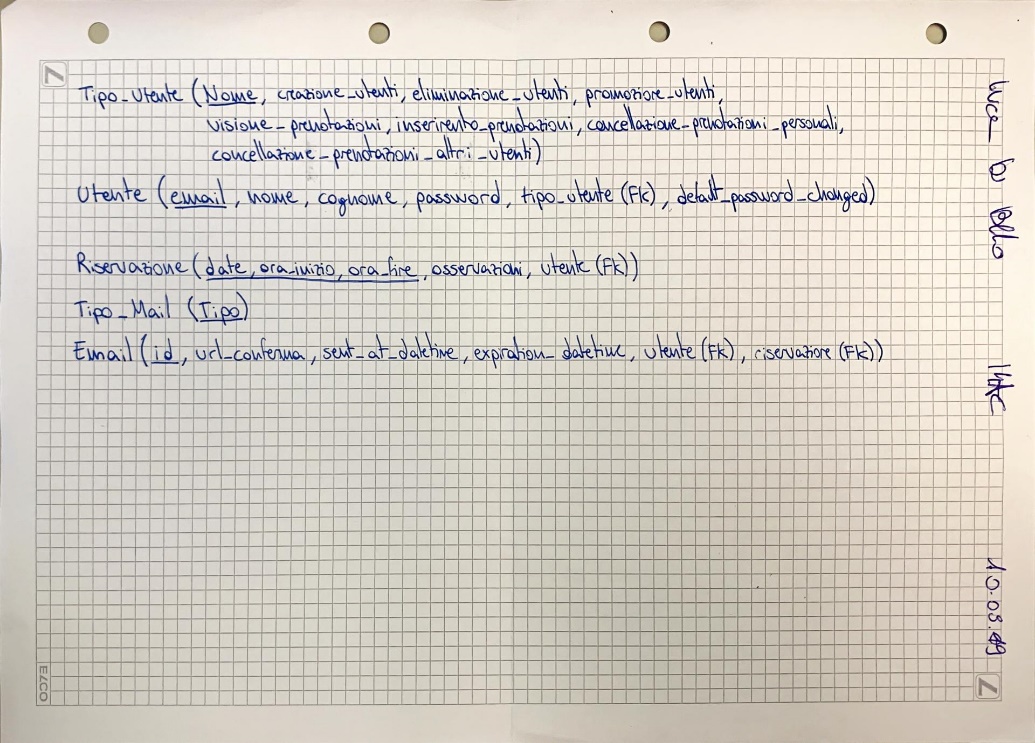
Questo è lo schema logico del database prodotto tramite il diagramma ER disegnato in precedenza:

Figure 4 - Schema logico database

### Tipi di dati

|  |  |
| --- | --- |
| Tipo\_Utente | |
| Nome attributo | Tipo attributo |
| Nome | *Primary Key, Varchar(50)* |
| Creazione\_utenti | *Bit, default 0* |
| Eliminazione\_utenti | *Bit, default 0* |
| Promozione\_utenti | *Bit, default 0* |
| Visione\_prenotazioni | *Bit, default 0* |
| Inserimento\_prenotazioni | *Bit, default 0* |
| Cancellazione\_prenotazioni\_personali | *Bit, default 0* |
| Cancellazione\_prenotazioni\_altri\_utenti | *Bit, default 0* |

|  |  |
| --- | --- |
| Utente | |
| Nome attributo | Tipo attributo |
| Email | *Primary Key, Varchar(255)* |
| Nome | *Varchar(100)* |
| Cognome | *Varchar(100)* |
| Password | *Varchar(128)* |
| Tipo\_utente | *Foreign Key, Varchar(50)* |
| Default\_password\_changed | *Bit, default 0* |

|  |  |
| --- | --- |
| Riservazione | |
| Nome attributo | Tipo attributo |
| Date | *Primary Key, Datetime* |
| Ora\_inizio | *Time* |
| Ora\_fine | *Time* |
| Password | *Varchar(128)* |
| Osservazioni | *Varchar(512)* |
| Utente | *Foreign Key, varchar(255)* |

|  |  |
| --- | --- |
| Tipo\_mail | |
| Nome attributo | Tipo attributo |
| Nome | *Primary Key, Varchar(50)* |

|  |  |
| --- | --- |
| Email | |
| Nome attributo | Tipo attributo |
| DA FINIRE | *Primary Key, Varchar(50)* |

## Design delle interfacce

ho iniziato a fare i mockup delle pagine web utilizzando un sito web chiamato

Ho realizzato 4 mockup utilizzato un sito web chiamato *mockflow.com:* pagina di login, pagina gestione utenti, pagina gestione prenotazioni e pagina visualizzazione prenotazioni (che verrà creata dal raspberry Pi).

Questo è il mockup che raffigura la pagina di gestione degli utenti:

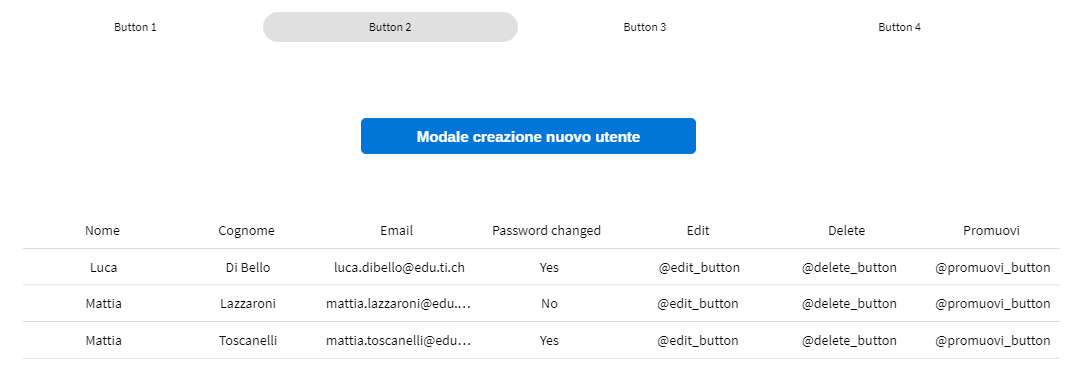


Figure 6 - Pagina gestione utenti

Questa è il mockup che raffigura la pagina di gestione delle prenotazioni:



Figure 5 - Pagina gestione prenotazioni

Questo è il mockup che raffigura la pagina di login, utilizzata per accedere al sito web:

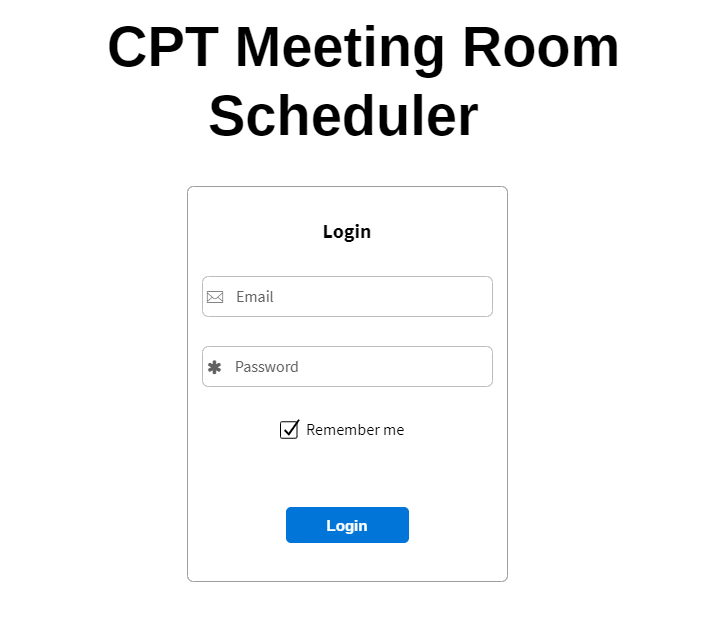


Figure 8 - Pagina di login

Ed infine questo è il mockup che raffigura la pagina di visualizzazione delle prenotazioni che verrà mostrata sul monitor:



Figure 7 - Visualizzazione prenotazioni su monitor

## 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

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

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 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
* …