

Generatore di frequenze con vari accessori gestito via web

Titolo del progetto: Generatore di frequenze con vari accessori gestito via web
Alunno/a: Giulio Bosco
Classe: SAMT I4AA
Anno scolastico: 2019/2020
Docente responsabile: Fabrizio Valsangiacomo

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

1 Introduzione

1.1 Informazioni sul progetto

Docente Responsabile: Fabrizio Valsangiacomo

Apprendista: Giulio Bosco

Data inizio progetto: 05.09.2019

Data consegna progetto: 20.12.2019

1.2 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.

1.3 Scopo

Creare un interfaccia di gestione per un generatore di frequenze ultrasoniche, con la possibilità di essere gestito da più utenti, quindi si necessita anche una piattaforma per gestire gli utenti. Il generatore deve poter essere acceso e spento tramite un telecomando, tramite un suono (un suono qualunque più alto di una determinata soglia, in decibel), tramite interfaccia WEB.

2 Analisi

2.1 Analisi del dominio

Questo capitolo dovrebbe descrivere il contesto in cui il prodotto verrà utilizzato, da questa analisi dovrebbero scaturire le risposte a quesiti quali ad esempio:

Background/Situazione iniziale

Quale è e come è organizzato il contesto in cui il prodotto dovrà funzionare?

Come viene risolto attualmente il problema? Esiste già un prodotto simile?

Chi sono gli utenti? Che bisogni hanno? Come e dove lavorano?

Che competenze/conoscenze/cultura posseggono gli utenti in relazione con il problema?

Esistono convenzioni/standard applicati nel dominio?

Che conoscenze teoriche bisogna avere/acquisire per poter operare efficacemente nel dominio?

...

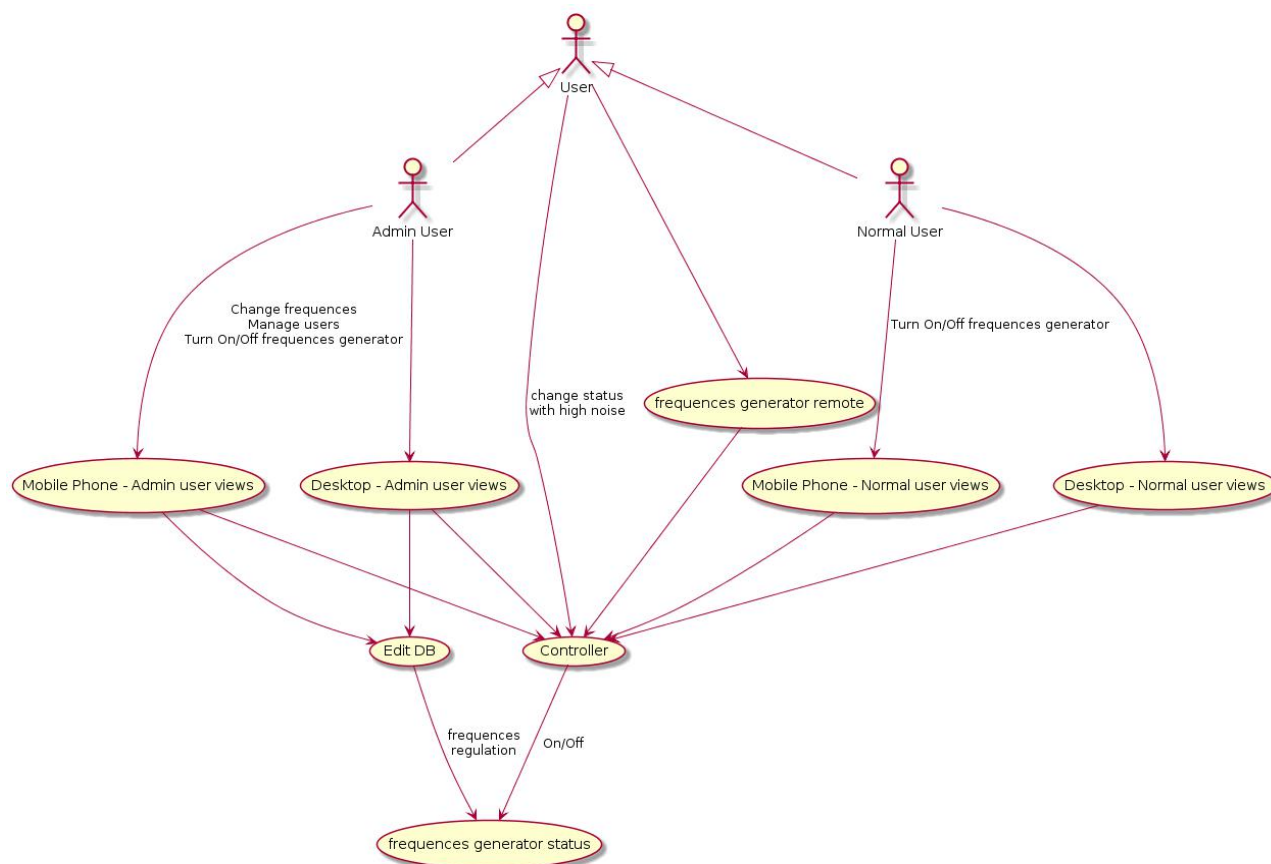
2.2 Analisi e specifica dei requisiti

ID: REQ-01	
Nome	Gestione generatore tramite WEB
Priorità	1
Versione	1.0
Note	-
	Sotto requisiti
Sub REQ 1	0-25'000 Hz
Sub REQ 2	Gestione Wireless del generatore
Sub REQ 3	Regolazione della frequenza
Sub REQ 4	Accensione/Spegnimento del generatore
Sub REQ 5	Mantenimento memoria
Sub REQ 6	Scatola finale cablata e protetta (Priorità 3)
Sub REQ 7	Minima attenuazione segnale (Priorità 3)

ID: REQ-02	
Nome	Gestione generatore tramite Telecomando
Priorità	2
Versione	2.0
Note	-
	Sotto requisiti
Sub REQ 1	Tasto Start
Sub REQ 2	Tasto Stop

ID: REQ-03	
Nome	Gestione generatore tramite Rumore
Priorità	2
Versione	2.0
Note	-
	Sotto requisiti
Sub REQ 1	Gestione tramite decibel
Sub REQ 2	Timer per spegnimento

ID: REQ-04	
Nome	Pagina WEB - Amministrazione utenti
Priorità	1
Versione	1.0
Note	-
	Sotto requisiti
Sub REQ 1	Funzionamento con maggiori browser
Sub REQ 2	Funzione di amministrazione
Sub REQ 3	Pagina gestione utenti
Sub REQ 4	Pagina gestione permessi
Sub REQ 5	Funzione di utente base
Sub REQ 6	Password provvisoria (Priorità 3)



2.3 Use case

Questo progetto ha come scopo di sviluppare la gestione di un generatore di frequenze tra 0 e 25'000 Hz, il quale deve poter essere gestito in maniere diverse. La frequenza deve poter essere regolata tramite una pagina web, solamente dall'utente amministratore. Mentre per quanto riguarda l'accensione e lo spegnimento del generatore, deve poter essere fatto tramite la pagina web (da un utente di base), tramite un telecomando e tramite un suono regolato in decibel (che di deve poi spegnere allo scadere di un timer).

Nel diagramma si può notare, un utente che tramite un forte rumore oppure con un telecomando può accendere o spegnere il generatore.

Oppure l'utente tramite l'applicativo (con un utente di base) può accendere o spegnere il generatore, mentre se esegue il login con un utente amministratore deve essere in grado di accendere o spegnere il generatore, cambiare la regolazione del generatore e gestire gli utenti.

2.4 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:

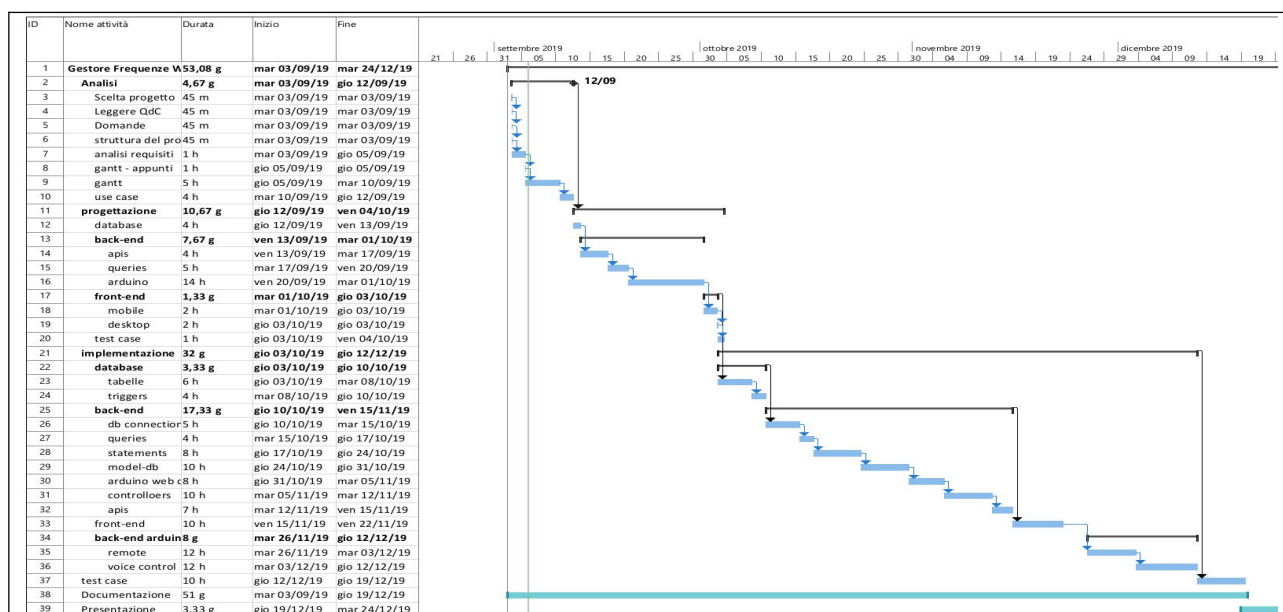


Figura 1: Diagramma di Gantt del progetto

Commento

2.5 Analisi dei mezzi

2.5.1 Software

- Arduino IDE (v1.8.9)
- Atom IDE (v1.38.2)
- plantuml (Version 1.2019.9)
- Libre Office (Version: 6.2.5.2)
- Java (v1.8)
 - Libreria org.gretty (v2.2.0)
 - Libreria Junit (v4.12)
 - Libreria JDBC (MySQL Connector Java - v8.0.11)
 - Libreria JSON (org.json:json:20171018)

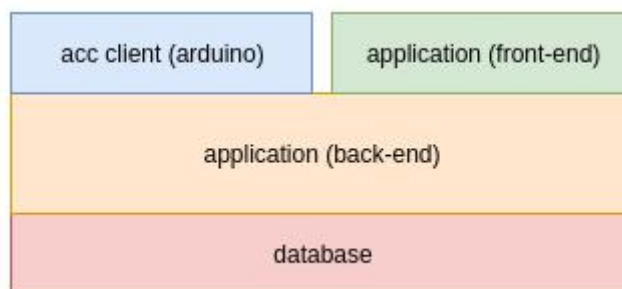
2.5.2 Hardware

- Mac Book Pro 2018 Intel Core i7 CPU 3.1GHz RAM 16GB
- Arduino UNO (Rev 3)
- Arduino YUN (Rev 2)
- Raspberry PI 3 Model B
- Circuito amplificatore

3 Progettazione

3.1 Design dell'architettura del sistema

Il progetto è sviluppato su quattro diversi elementi, che sono messi in evidenza nello schema sottostante:



Alla base di tutto vi è il database nel quale vengono salvati tutti i dati dell'applicativo, il quale è diviso in 2 due parti, il back-end ed il front-end.

Tutto il sistema, verrà strutturato in maniera modulare, in modo che tutti gli elementi del progetto siano indipendenti, quindi siano più facili da testare, singolarmente, per cercare di avere meno problemi possibili durante il corso del progetto, che siano anche più facili da mantenere e modificare in futuro e che possano essere riutilizzati in progetti futuri. Alcuni moduli del progetto verranno ripresi da vecchi progetti che ho svolto in questi anni alla SAMT e probabilmente riscritti siccome ora ho più conoscenze ed esperienza, viene utilizzato lo stesso concetto, ma riscrivendo il codice solamente in parte oppure totalmente.

I vari moduli, possono avere dei sotto moduli, per semplificare ancora lo sviluppo ed il mantenimento dell'applicativo.

Per ogni modulo è stata fatta una progettazione, in alcuni più approfondita mentre in altri meno. Questo perché a dipendenza dei moduli vi sono più o meno complessi, mentre per quelli che si pensa di prendere da dei vecchi progetti non è proprio stata fatta, verranno fatti degli adattamenti direttamente durante lo sviluppo.

Per esempio la parte relativa al *acc client (Arduino)* verrà preso dal progetto *domotics* (<https://github.com/giuliobosco/domotics>), come il modulo *application (front-end)*.

Per quanto riguarda la web app, sarà sviluppata con le Rest API, per la comunicazione fra il front-end ed il back-end (ce ne saranno anche altre per la parte dell'Arduino). Ho preso questa decisione, perché utilizzarle è una best practices, del trend attuale dello sviluppo web.

Le RestAPI servono per uniformare il formato dei dati, per facilitare l'uniformità e l'integrazione delle varie applicazioni web disponibili. Un gran punto di forza di è quello di poter eseguire delle richieste al server o aggiornare i dati senza ricaricare la pagina.

Le RestAPI, non sono altro che i dati del back-end scritti in formato JSON. Come per esempio:

```

{
  "users": [
    {
      "username": "giulio.bosco",
      "firstname": "Giulio",
      "lastname": "Bosco"
    },
  ],
}
```



```
{
  {
    "username": "fabrizio.valsangiaco",
    "firstname": "Fabrizio",
    "lastname": "Valsangiaco"
  }
}
```

3.2 Design dei dati e database

Per lo sviluppo del database, ho prima di tutto creato il minimo indispensabile per il progetto, quindi tutte le tabelle di cui necessito, tutti gli attributi, senza il quale il progetto non funziona.

Lista delle tabelle:

- generators
- users
- groups
- permissions

Dopo di che ho pensato potesse essere una buona idea tenere traccia delle operazioni eseguite sul database. Questo perché è dal lato della piattaforma WEB, il progetto è piccolo. Quindi potrei investire del tempo nello sviluppare questa parte del progetto, che potrebbe comunque essere riutilizzata in qualunque progetto.

Per eseguire i log delle azioni effettuate sulle banche dati, vi sono diversi modi:

1. Eseguire il log delle azioni di tutte le tabelle in una tabella di log, nella quale si inserisce la query eseguita, l'autore, la data e l'ora.
2. Eseguire il log in una tabella dedicata per ogni tabella, nella quale si inseriscono tutte le azioni che vengono eseguite sulle tabelle.
3. Eseguire il log nell'applicativo, quindi inserire una parte del software che esegua il log del database.
4. Si potrebbe utilizzare un software di monitoraggio della banca dati.

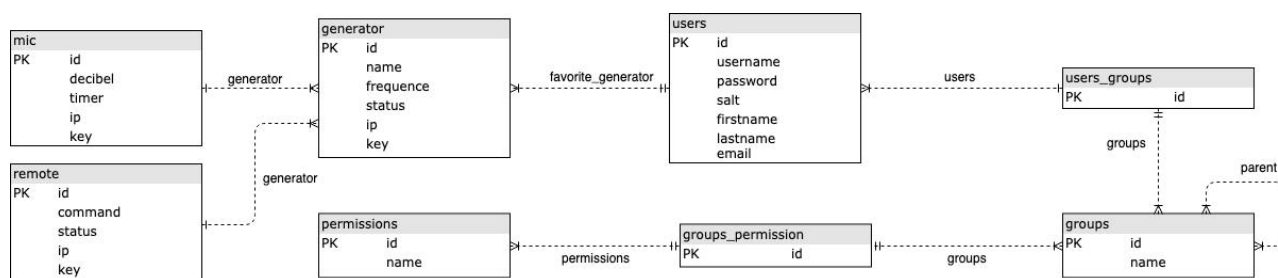
Analizzando queste opzioni ho trovato questi pro e questi contro, di ogni metodologia.

Metodo	Punti a favore	Punti a sfavore
1	Semplice implementazione e integrazione con la banca dati Facile ricostruzione del database	Difficile creare dei report sui log
2	Facilita generazione di report sulle azioni eseguite sul database mantiene separati i dati	Implementazione del database più complessa
3	Facile implementazione del database	Complica la struttura del software Nel caso in cui vi fossero in futuro più elementi che interagiscono con lo stesso database, i log non sarebbero

		più autentici
4	Semplificherebbe il codice	<p>Dovrei prendere del tempo per imparare ad utilizzare un altro software</p> <p>Richiederebbe più risorse</p>

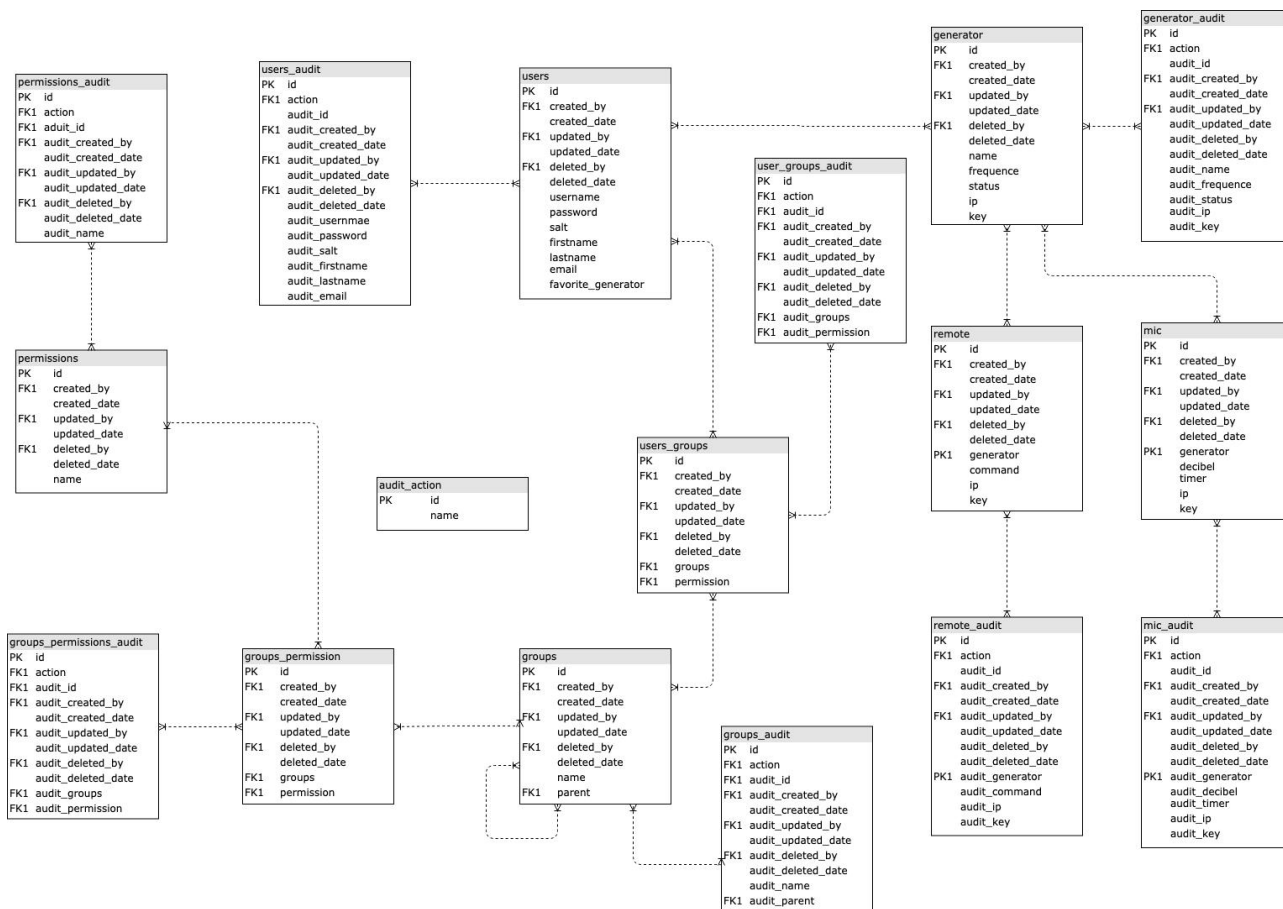
Analizzando questi punti, ho deciso di utilizzare il secondo metodo.

Del database, vi sono 2 schemi. Uno che contiene lo schema di base, quindi solamente le informazioni rilevanti per il progetto. Ed uno completo che comprenderà anche gli attributi e le tabelle legati al log.



In questo schema vi sono le tabelle con gli attributi minimi, per far funzionare l'applicativo. Vi è una tabella `generator` che dà la possibilità di inserire diversi generatori, i quali hanno un nome, una frequenza alla quale devono lavorare, uno stato, un IP ed una key (pensata per creare una comunicazione sicura fra il server ed i controller degli altoparlanti). Questo permette al progetto di essere espandibile.

Dopo di che vi sono le tabelle users, groups, e permissions, che servono per gestire i permessi. fra le varie tabelle vi sono anche le tabelle ponte, per permettere le relazioni molti a molti.



In questo schema si possono notare molte più tabelle e più attributi, per ogni tabella che vi era nello schema antecedente, 6 attributi, che servono per salvare la data e l'autore delle azioni principali che si possono fare sul database (create, update, delete). In oltre vi è un'altra tabella, nella quale si inseriscono gli audit, cioè gli stessi parametri, con in più l'azione eseguita ed un id per ogni audit. Le relazioni fra le tabelle di audit e la tabella `audit_aciton` non sono rappresentate, come le relazioni fra tutti i campi discussi prima e la tabella `users`, questo per permettere una miglior leggibilità dello schema.

3.3 Design delle interfacce

Descrizione delle interfacce interne ed esterne del sistema e dell'interfaccia utente. La progettazione delle interfacce è basata sulle informazioni ricavate durante la fase di analisi e realizzata tramite mockups.

3.4 Design procedurale

3.4.1 Applicativo (back-end)

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.

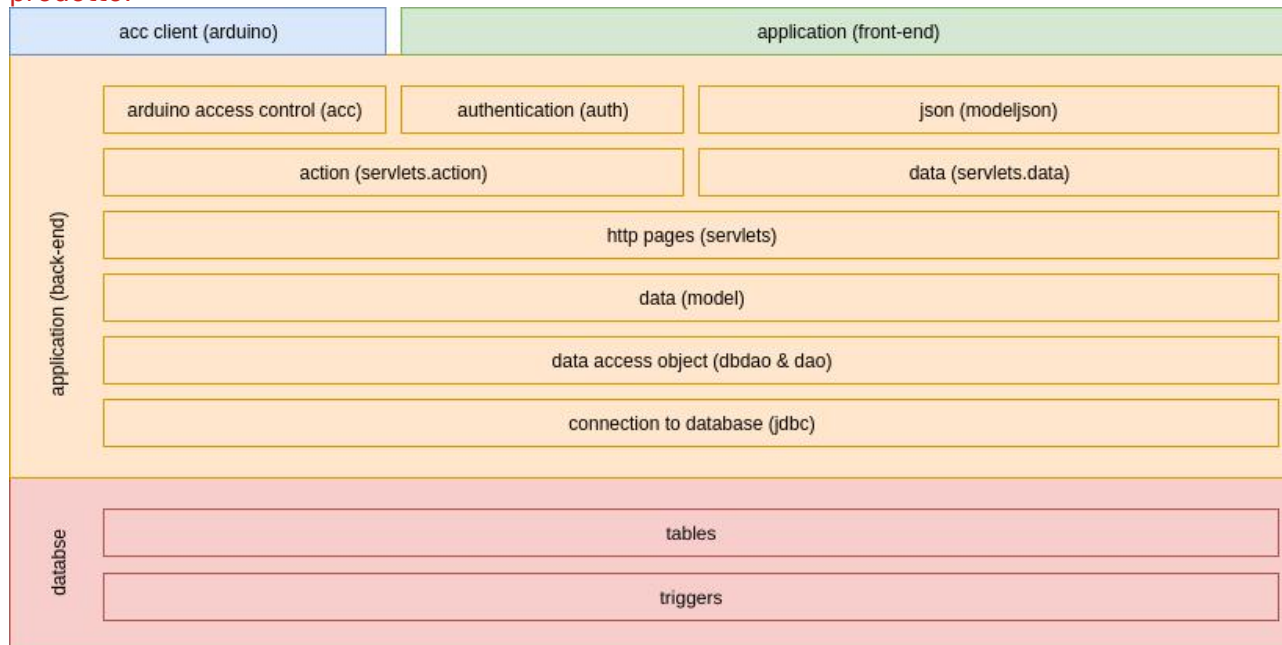


Diagramma del prodotto con i sotto moduli in evidenza... ATTENZIONE

Per quanto riguarda la progettazione del back-end, che sarà abbastanza complesso, siccome vi deve essere un sistema di gestione degli utenti e dei permessi, basato su un database. Vi deve essere una rappresentazione dei dati (*model*), un interfaccia che permetta di comunicare con la banca dati, ed un interfaccia che permetta di comunicare con il lato dell'applicativo *front-end* e vi deve essere la possibilità di eseguire delle azioni dal *front-end*.

Siccome il lato back-end dell'applicativo è molto complesso, ho deciso di suddividerlo in dei sotto moduli:

- connection to database (jdbc): layer nel quale avviene la connessione al database
- data access object (dbdao & dao): interfaccia dei dati con il database
- data (model): rappresentazione dei dati
- http pages (servlets): **sono i punti di accesso ai dati e alle azioni**
- data (servlets.data): una sotto cartella nella quale vengono esposti i dati
- actions (servlets.action): una sotto cartella di *servlets* nella quale vengono espote le azioni possibili sull'applicativo
- json (jsonmodel): in questo modulo, vengono trasformati gli elementi modello in elementi in formato JSON, usati come interfaccia per il lato front-end dell'applicativo.
- authentication (auth): modulo di autenticazione e gestione dei permessi
- arduino access controll (arduino): modulo di gestione del controller arduino

3.4.1.1 connection to database (jdbc):

La connessione al database avviene tramite il driver di default di java.

CONTINUARE

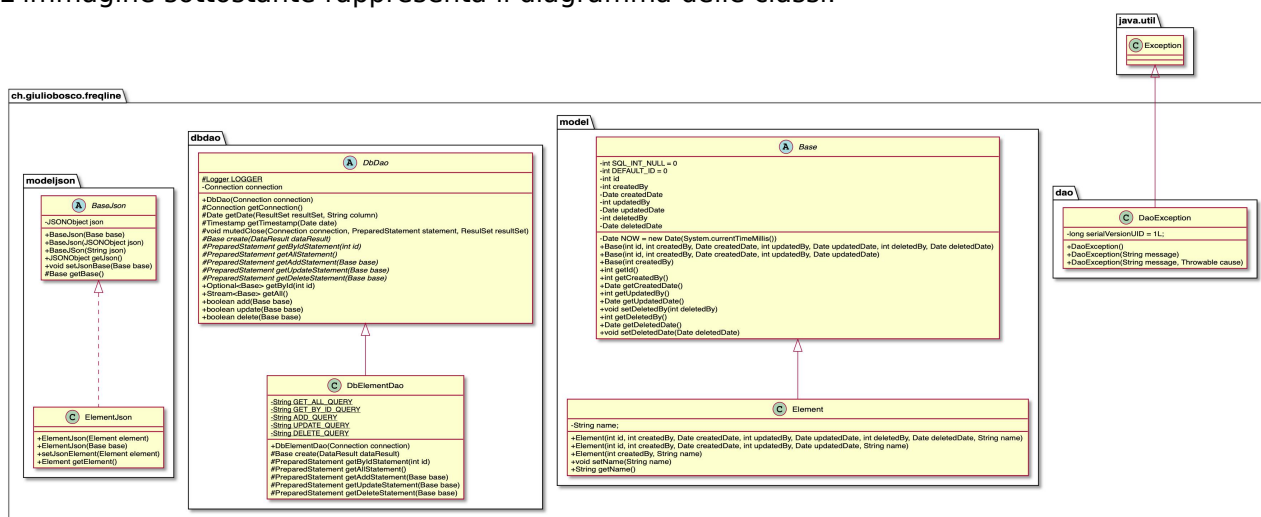
L'interfaccia dei dati fra le rappresentazioni dei modelli e il database (*dbdao*), verrà implementato sul modello DAO (Data Access Object), che dovrà essere sviluppato per ogni tabella. Il modello DAO, comprende 5 possibili interazioni con il database:

- getById*: seleziona un elemento dalla tabella, tramite il suo id
- getAll*: seleziona tutti gli elementi dalla tabella
- add*: aggiunge un elemento alla tabella
- update*: aggiorna un elemento nella tabella, sostituisce tutti i parametri
- delete*: Elimina un elemento dalla tabella, tramite l'id

Ognuno di questi metodi, comprende l'inizializzazione della *query*, con il *prepared statement*, l'inserimento dei dati nello *statement*, la sua esecuzione e l'analisi del suo risultato nel caso sia una *query* di selezione. Per questo motivo, ho deciso di progettare il software in maniera che sia il più astratto possibile, cioè che sia meno ripetitivo possibile. Quindi anche per quanto riguarda i modelli di dati (*model*) necessito una struttura che sia in parte astratta.

I modelli verranno trasformati in delle stringhe in formati JSON, anche per questo processo ho deciso di utilizzare una struttura astratta per riciclare il più possibile il codice.

L'immagine sottostante rappresenta il diagramma delle classi.



Per lo sviluppo del software, e le connessioni al database, ho deciso di basarmi sul modello DAO (Data Access Object).

Prima cosa che viene fatta durante lo sviluppo, è la scrittura delle classi che rappresentano le varie tabelle del database. Dopo di che si implementano le classi che si collegano al database.

Poi delle classi che trasformano i modelli delle tabelle in JSON, nascondendo le informazioni che non devono essere accessibili tramite le API.

3.4.2 Applicativo (front-end)

Il lato front-end, dell'applicativo, verrà sviluppato basandosi su

4 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.

5 Test

5.1 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:	TC-001	Nome:	Import a card with KIC, KID and KIK keys, but not shown with the GUI
Riferimento:	REQ-012		
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 h.n) and Cards_1.2.001.txt (appendix h.n). PIN (OTA_VIEW_PIN_PUK_KEY) and ADM (OTA_VIEW_ADM_KEY) user right not set.		
Procedura:	<ol style="list-style-type: none"> Go to "Cards manager" menu, in main page click "Import Profiles" link, Select the "1.2.001.xml" file, Import the Profile 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 Research the "41795924770" Card, Click the imsi card link Check the card details Execute the SQL: SELECT imsi, dir, keyset, cntr, rawtohex(kickey), rawtohex(kidkey), rawtohex(kickey), 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)		

5.2 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.

5.3 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.

6 Consuntivo

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

7 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

7.1 Sviluppi futuri

Migliorie o estensioni che possono essere sviluppate sul prodotto.

7.2 Considerazioni personali

Cosa ho imparato in questo progetto? ecc

8 Bibliografia

8.1 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,

8.2 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.

8.3 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.

9 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
...