



Gabriele Novello

5^C Informatica

Elaborato maturità

2021

Indice

1. [Introduzione](#)
2. [Principali linguaggi e tecnologie](#)
3. [Casi d'uso](#)
4. [Progettazione interfaccia utente](#)
5. [Architettura](#)
6. [Google Books API](#)
 - 6.1. [Parametri della chiamata](#)
 - 6.2. [Struttura della risposta](#)
7. [Progettazione database](#)
 - 7.1. [Progettazione concettuale](#)
 - 7.2. [Regole di derivazione](#)
 - 7.3. [Progettazione logica](#)
 - 7.4. [Normalizzazione](#)
 - 7.5. [Script di creazione in SQL](#)
 - 7.6. [Principali query utilizzate](#)
8. [Architettura della rete](#)
 - 8.1. [Schema logico](#)
 - 8.2. [Piano di indirizzamento](#)
9. [Configurazione servizi e NAT](#)
 - 9.1. [DHCP](#)
 - 9.2. [Web server](#)
 - 9.3. [DNS server](#)
 - 9.4. [NAT statico](#)
 - 9.5. [PAT](#)
10. [Sicurezza della rete](#)
 - 10.1. [DMZ](#)
11. [Protocolli di comunicazione](#)
 - 11.1. [WPA2-PSK](#)
 - 11.2. [HTTPS](#)
12. [Prevenzione interruzione servizi](#)
13. [Gestione progetto](#)
 - 13.1. [Project Charter](#)
 - 13.2. [Work Breakdown Structure](#)
 - 13.3. [Matrice delle responsabilità](#)
 - 13.4. [Stima dei costi](#)
 - 13.5. [Calcolo dei tempi](#)
14. [Conclusioni personali](#)
15. [Sitografia e bibliografia](#)
16. [Glossario](#)

1. Introduzione

Il progetto *libreria digitale* è nato dall'unione di varie idee. Si voleva sviluppare un sito web dinamico che comprendesse l'utilizzo di un [API REST](#) e la gestione di [file strutturati](#). La tematica principale, i libri, è stata scelta per rendere l'applicazione web utilizzabile in modo pratico nell'ambiente scolastico, ma anche al di fuori di esso.

Si è pensato, per esempio, che gli studenti ai quali vengono assegnati libri da leggere, possano usufruire del sito per verificare se sia possibile leggerli online gratuitamente (i libri classici/storici che vengono fatti leggere solitamente hanno molte probabilità di rientrare tra questi).

Si andrà quindi a realizzare un sito web con lo scopo di offrire una fonte di dati vasta tra cui cercare libri, con la possibilità di scaricare e/o visualizzarne i contenuti e permettendo il loro salvataggio all'interno di una libreria personale associata all'account di ogni utente.

Gli utenti non saranno obbligati a crearsi un account per consultare il sito. La registrazione (o il login) diverrà obbligatorio solo nel caso in cui l'utente voglia utilizzare la libreria personale, salvando i libri di proprio interesse e "legandoli" al proprio account per potervi interagire e accedere più facilmente.

Abbiamo utilizzato come fonte di dati quelli forniti dalla sezione libri di Google ([Google Books](#)) che mette a disposizione gratuitamente un'[API](#) con la quale è possibile ricevere i dati sui libri in file strutturati [JSON](#) ([per approfondimenti](#)).

2. Principali linguaggi e tecnologie

Per lo sviluppo del sito web saranno utilizzati diversi linguaggi. [HTML](#), [CSS](#) e [JavaScript](#) comporranno le pagine web visualizzate dal [client](#), [PHP](#) gestirà le funzioni lato [server](#) (per esempio le interazioni col [database](#)).

Per dare un'esperienza dinamica all'utente, molte funzioni del sito utilizzeranno [AJAX](#) che, gestendo la comunicazione client-server in modo asincrono, permetterà alle pagine di aggiornarsi dinamicamente senza un'effettiva ricarica manuale da parte dell'utente. Un esempio a cui è stato applicato AJAX è la ricerca dei libri.

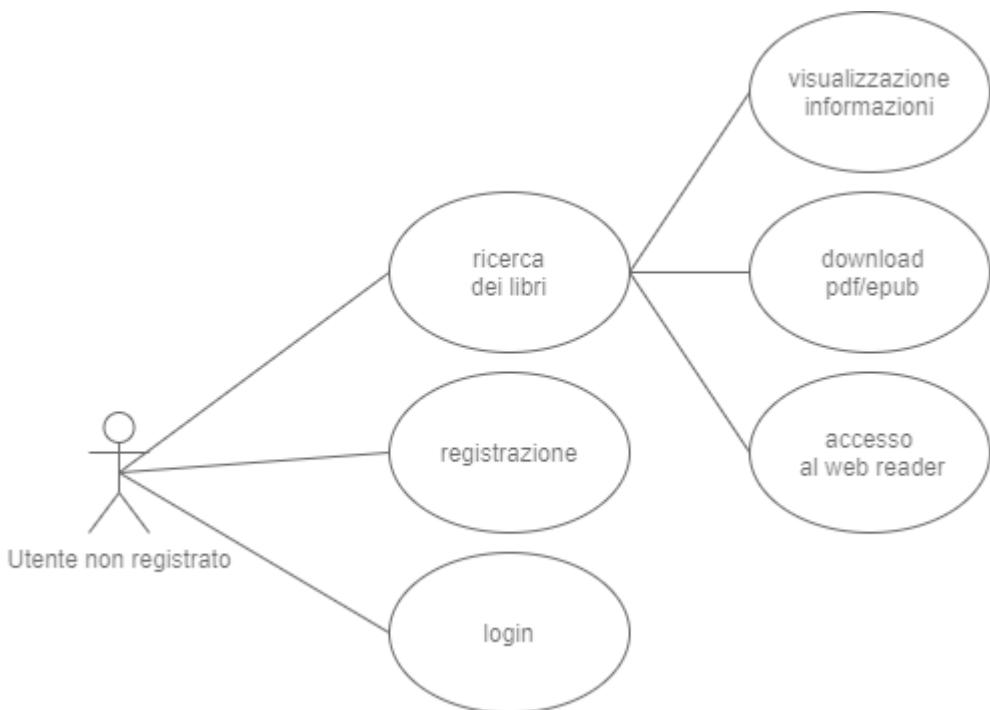
Per quanto riguarda il database, è stato scelto di tipo [relazionale](#) per facilitare la rappresentazione della relazione tra utenti e libri su cui si basava gran parte del sito. Esso sarà gestito dal [RDBMS MySQL](#).

L'intero sito verrà caricato sul servizio di hosting online gratuito [Altervista](#), il quale sarà accessibile a questo [indirizzo](#).

3. Casi d'uso

I casi d'uso della piattaforma coinvolgono due tipologie di utenti: gli utenti non registrati e gli utenti registrati. E' stato deciso di non obbligare gli utenti a crearsi un account quando non è effettivamente necessario (per esempio se un utente non è interessato all'utilizzo della libreria personale) in modo da risparmiare tempo al consumatore e spazio sul nostro database.

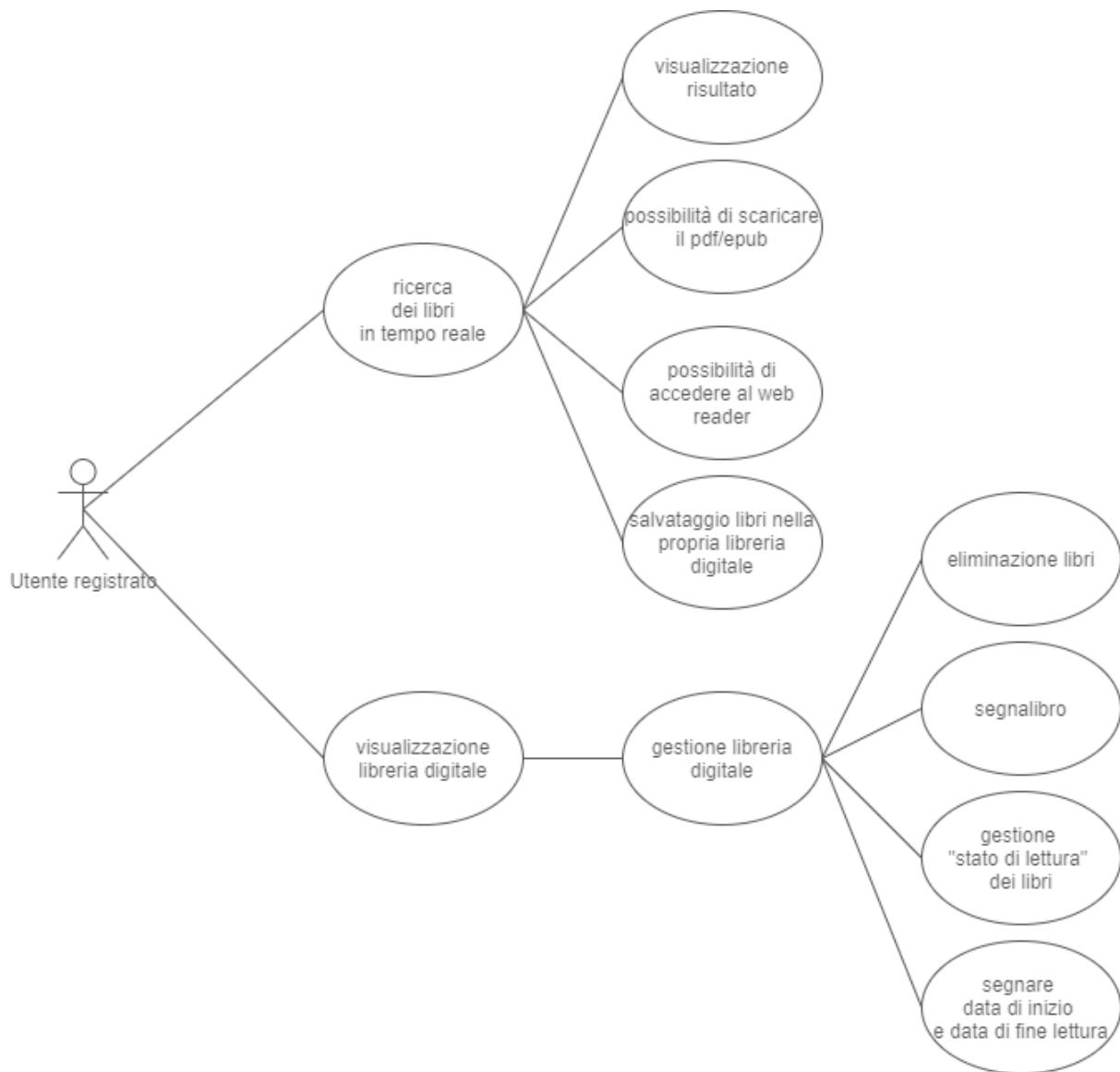
UTENTI NON REGISTRATI



Le funzioni che può svolgere un utente che non possiede un account sono:

- crearsi un account accedendo alla pagina di registrazione;
- accedere nella pagina di login se ne possiede già uno;
- cercare i libri nella barra di ricerca utilizzando varie tipologie di filtri, ordinamenti e opzioni di impaginazione disponibili;
 - una volta ricercato uno o più libri, l'utente può visualizzarne le informazioni;
 - se disponibile, è possibile scaricare il pdf e l'epub tramite l'apposito pulsante;
 - se disponibile, è possibile accedere al web reader di Google per leggere un estratto o il libro per intero.

UTENTI REGISTRATI



Le funzioni che invece può svolgere un utente registrato sono:

- cercare i libri nella barra di ricerca utilizzando tutti i tipi di filtri, ordinamenti e opzioni di impaginazione disponibili;
 - l'utente può visualizzarne le informazioni;
 - se disponibile, è possibile scaricare il pdf e l'epub tramite l'apposito pulsante;
 - se disponibile, è possibile accedere al web reader di Google per leggere un estratto o il libro intero;
 - può scegliere se aggiungere un determinato libro alla sua libreria personale;

- accedere alla propria libreria personale;
 - gestire la libreria a proprio piacimento con le funzioni messe a disposizione dalla piattaforma;
 - rimuovere i libri che non si vuole più avere salvati;
 - tenere il conto delle pagine lette;
 - gestire lo “stato di lettura” (in lettura, in pausa o completato);
 - modificare la data di inizio e di fine della lettura.

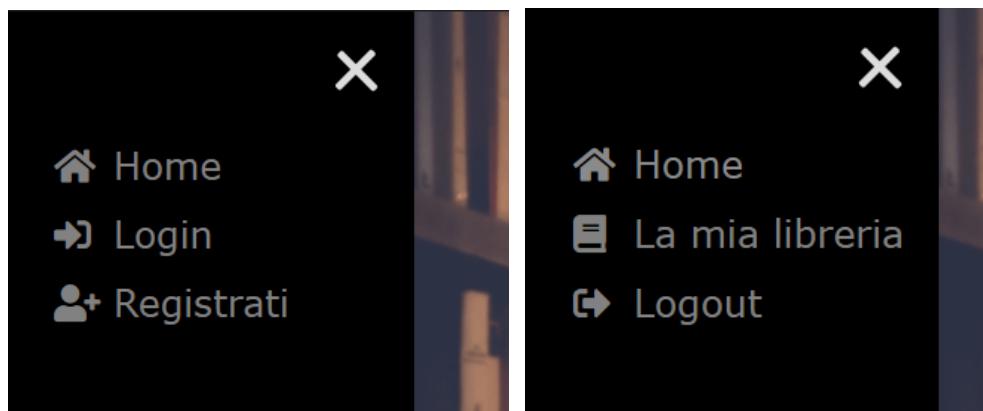
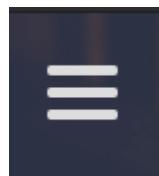
4. Progettazione interfaccia utente

Lo stile delle pagine del sito è stato progettato con l’obiettivo di rendere l’esperienza di navigazione semplice e veloce. Il tema scuro, invece, è stato scelto per alleggerire lo sforzo sugli occhi dell’utente.

Le pagine principali sono 4:

- la home page;
- la pagina di registrazione;
- la pagina di login;
- la libreria personale.

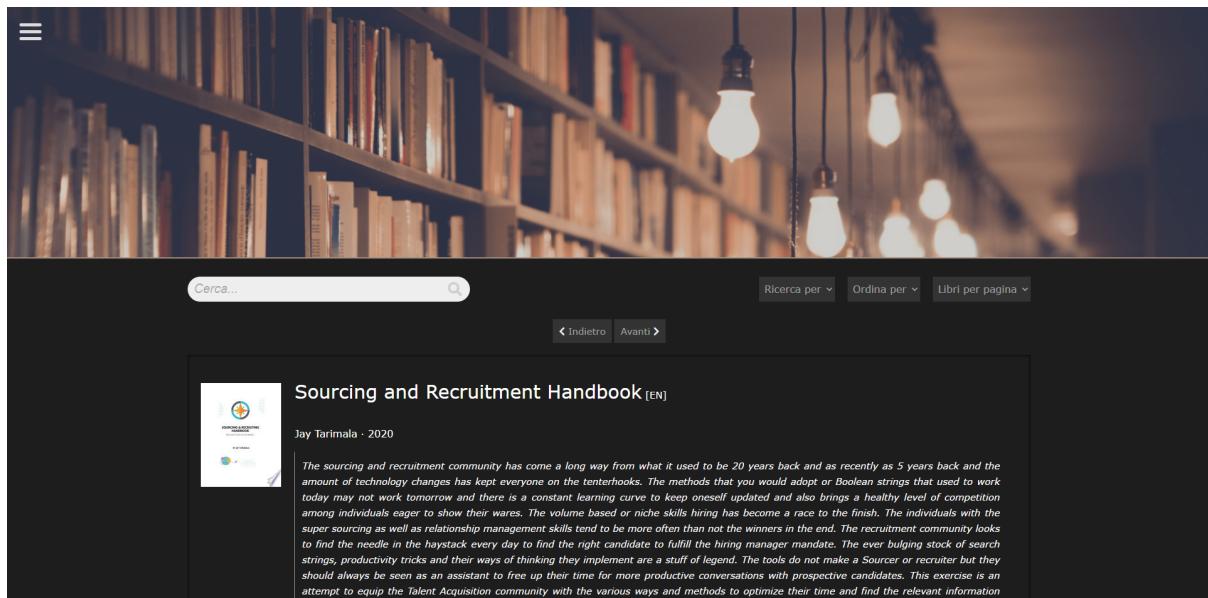
Sarà possibile navigare tra le pagine del sito utilizzando la navbar laterale (sidebar) accessibile cliccando l'*hamburger* in alto a sinistra in ogni pagina del sito.



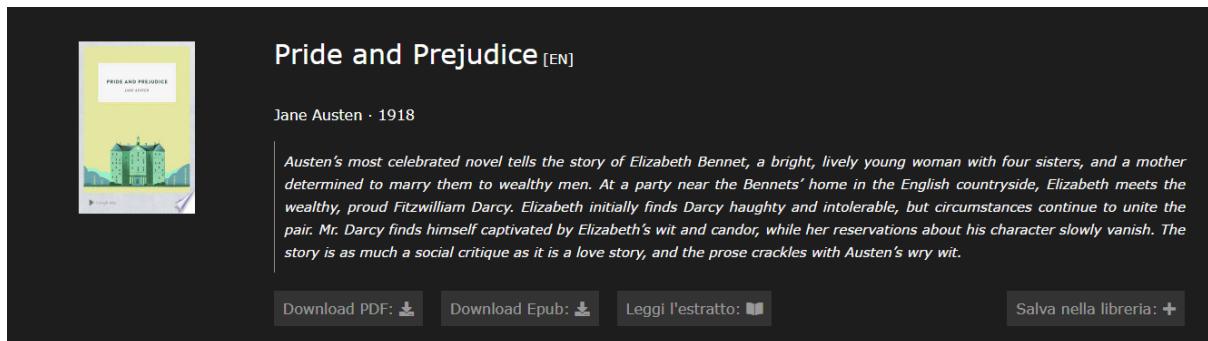
La homepage sarà la prima pagina che un utente vedrà appena collegato al sito. Al suo interno sarà presente la funzione principale del sito: la ricerca dei libri.

Il layout della pagina conterrà quindi: una barra di ricerca, una "zona filtri" (*ricerca per, ordina per, libri per pagina*) e una tabella in cui verranno visualizzati i libri cercati con relativo impaginamento.

Ogni ricerca viene effettuata utilizzando AJAX per inviare chiamate all'API che restituirà i dati in formato JSON. Essi verranno elaborati grazie a JavaScript e visualizzati graficamente.

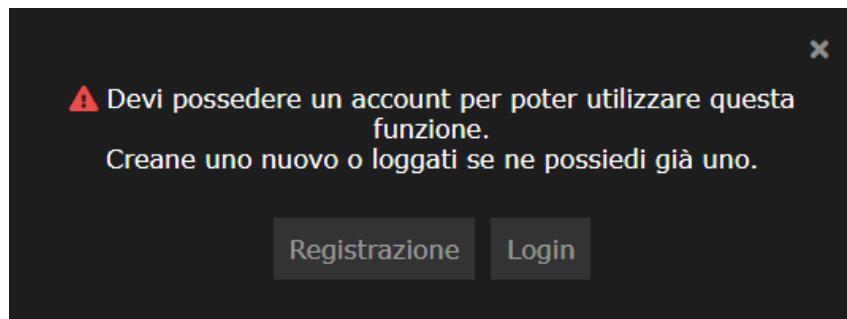


Di ogni libro visualizzato verranno visualizzati la copertina, il titolo, la lingua, gli autori, l'anno di pubblicazione, la descrizione e i pulsanti per effettuare le seguenti operazioni: download del pdf o dell'epub, lettura dell'estratto e salvataggio nella propria libreria.



Se si prova a salvare un libro senza possedere un account apparirà una finestra di pop up che avviserà l'utente (vedi sotto) e gli permetterà di raggiungere la pagina di registrazione o la pagina di login tramite un apposito pulsante.

Se l'utente non è interessato all'utilizzo della libreria personale, potrà chiudere il pop up e continuare a usufruire delle funzioni accessibili agli utenti non registrati elencate in precedenza ([vedi casi d'uso](#)).



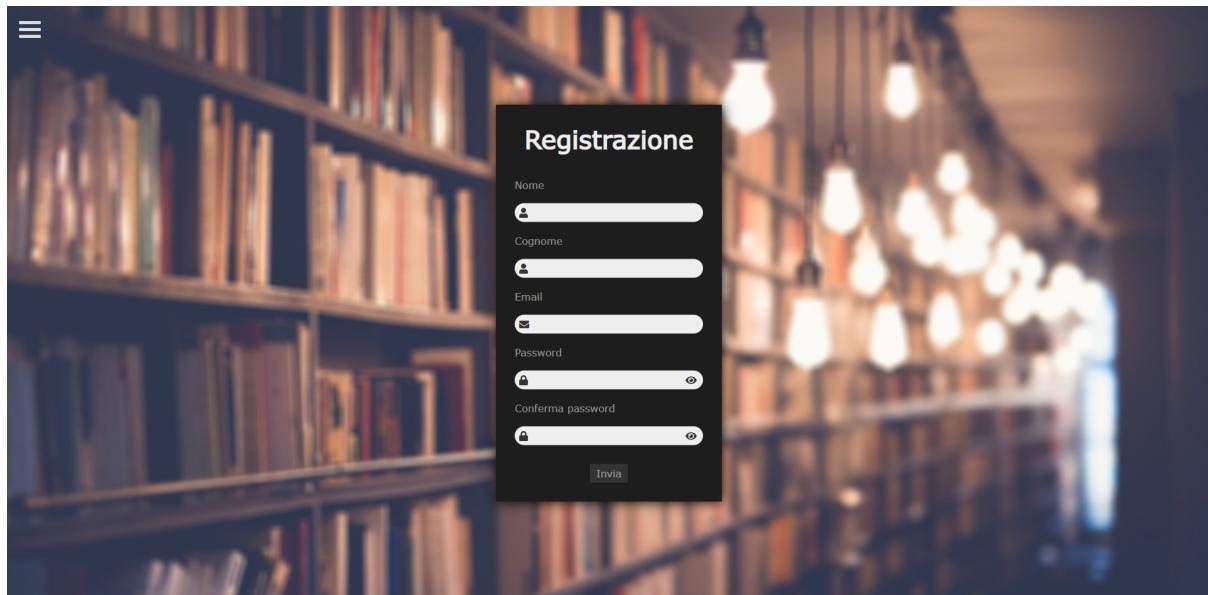
La pagina di registrazione ospiterà il form da compilare nel caso in cui un utente voglia crearsi un account. I campi da compilare sono: nome, cognome, email, password e una conferma di quest'ultima.

Sull'email è presente un controllo dinamico che permette di verificare in tempo reale, con AJAX, se un'email uguale a quella inserita è già esistente all'interno del database. Se c'è una corrispondenza il bordo diventerà rosso e non sarà possibile inviare il form, se non c'è corrispondenza il bordo diventerà verde e si potrà procedere. Sarà quindi possibile creare un solo account per ogni email inserita.

All'invio del modulo i dati verranno inseriti all'interno del database.

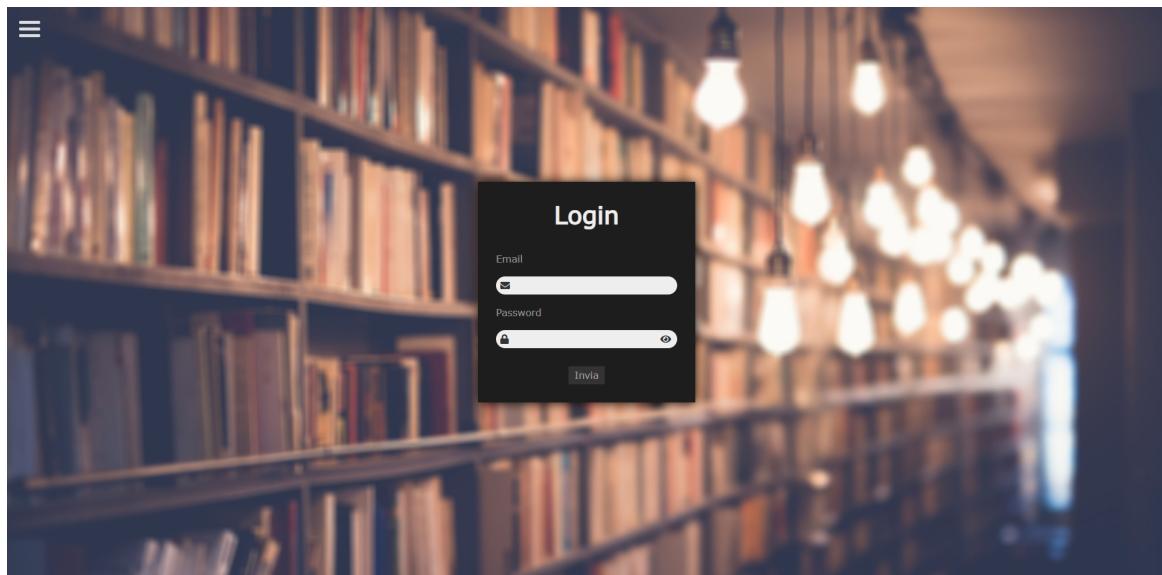
Per criptare e mantenere la segretezza della password nel database verrà applicata la [funzione di hashing](#) SHA256.

Se l'inserimento dei dati nel database avrà successo verrà automaticamente creata la sessione per l'utente, senza dover effettuare un ulteriore login.



La pagina di login richiede l'autenticazione dell'utente tramite email e password inseriti durante la registrazione. All'invio del form le due credenziali (in cui alla password verrà nuovamente applicata la funzione di hashing SHA256) verranno ricercate all'interno del database e confrontate. Se ci sarà una corrispondenza

l'utente sarà autenticato, verrà aperta una sessione e sarà reindirizzato alla homepage.



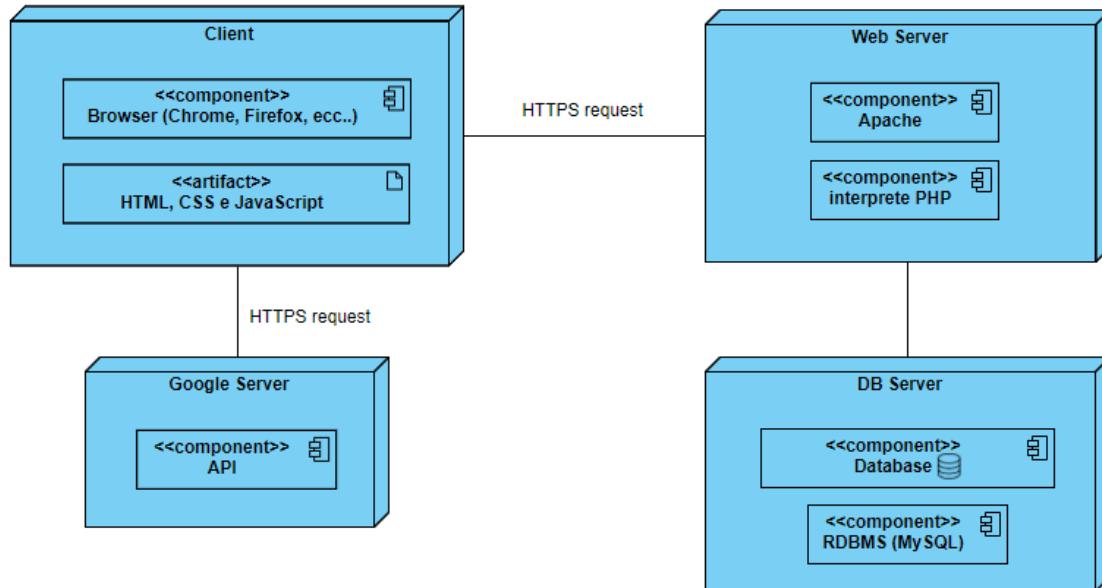
La libreria personale visualizza ogni libro salvato nel nostro account per permettere di raggiungerli con maggiore facilità e per usufruire delle funzioni disponibili: segnalibro (aggiornare le pagine lette), aggiornare la data di inizio e fine lettura e lo "stato di lettura". I libri saranno ordinati in modo che quelli *in lettura* risultino in cima per facilitare il loro raggiungimento.

Le informazioni mostrate sono titolo, lingua, il codice ISBN (per permettere a un utente di cercare quella specifica edizione del libro per acquistarlo) e le pagine totali del libro.

Infine è possibile eliminare ogni libro dalla libreria in qualsiasi momento premendo la "X" in alto a destra.

A screenshot of a personal library application. At the top, there's a navigation menu icon (three horizontal lines). Below it, a search bar shows the title "I Malavoglia [IT]". Underneath, there's a card for the book "I MALAVOGLIA" by Giacomo Leopardi, with the ISBN "IND:39000000816293", page count "0 su 465", and reading status "In Lettura". Below this, another card shows the book "Pride and Prejudice [EN]" by Jane Austen, with the ISBN "HARVARD:32044086796588", page count "0 su 401", and reading status "In Lettura". Both cards have a close ("X") button in the top right corner.

5. Architettura



L'architettura dell'applicazione web è composta da 3 livelli e comprende il client, il web server e il database server. In aggiunta, è rappresentato il server di Google al quale andremo a inoltrare le richieste API e che ci fornirà i dati riguardanti i libri.

Il client rappresenta tutti i dispositivi, mobili o fissi, utilizzati dagli utenti per accedere al sito web.

Il web server, nel nostro caso specifico rappresentato da Altervista (che utilizza Apache a sua volta), gestirà tutte le richieste **HTTPS** del client (porta 443) per ricevere le pagine web e farà da ponte tra quest'ultimo e il database. Si specifica che si fa riferimento al protocollo HTTPS (invece che il normale HTTP) perché Altervista offre gratuitamente la possibilità di usufruirne per una maggiore sicurezza del sito ([per approfondimenti](#)).

Il DB server contiene il database relazionale. Il client non potrà connettersi direttamente ad esso ma dovrà effettuare una richiesta al web server che avrà il compito di ponte tra i due.

Infine, il Google server nel quale è contenuta l'API dei libri che, in seguito a una chiamata, passerà i dati di questi ultimi al client che li utilizzerà nell'applicazione.

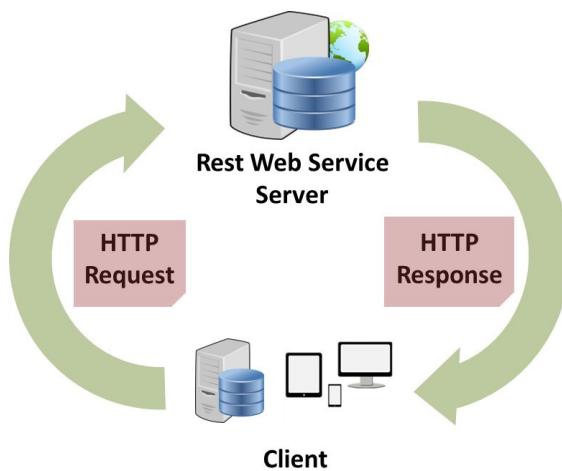
6. Google Books API

Prima di iniziare a parlare specificatamente dell'API utilizzata è bene introdurre il concetto stesso di API.

Un'API (Application Programming Interface) è un set di informazioni e protocolli con i quali vengono realizzati e integrati software applicativi. In breve, facilitano la comunicazione tra il proprio servizio e uno esterno senza essere a conoscenza di come essi vengano implementati. Questa proprietà delle API permette di risparmiare tempo e risorse ed è il principale motivo del loro successo.

Una API REST o RESTful (REpresentational State Transfer) è un elemento di intermediazione tra l'utente e le risorse. L'utente andrà infatti a richiedere all'API dei dati (chiamata) che verranno ritornati (risposta) in un formato strutturato e facilmente leggibile (JSON nel nostro caso).

Il principio su cui si basa l'insieme di vincoli architetturali REST è la comunicazione stateless tra client e server: le informazioni del client non verranno mai memorizzate sul server e ogni chiamata sarà distinta e non connessa.



E' stato scelto di utilizzare Google Books API perché contiene una vasta gamma di dati ed è in continuo aggiornamento da parte di Google, permettendo al nostro sito di essere autonomo, senza bisogno di un amministratore che inserisca, elimini e aggiorni i libri.

Durante lo sviluppo del nostro sito si è fatto riferimento alla documentazione messa a disposizione [qui](#).

6.1. Parametri della chiamata

Per effettuare la chiamata andremo ad utilizzare il [metodo GET](#) di HTTP e HTTPS. Esso permette di richiedere le risorse corrispondenti ad un determinato indirizzo URL inserito. E' possibile aggiungere determinati parametri a quest'ultimo, chiamati *query string*, per un risultato più specifico.

L'indirizzo al quale il nostro sito effettua le chiamate per ottenere i dati riguardanti i libri è il seguente:

```
https://www.googleapis.com/books/v1/volumes?q={parametro di ricerca}
```

q va a indicare il parametro obbligatorio di ricerca di default. A questo parametro possono aggiungersene ulteriori per filtrare, ordinare e impaginare i dati in risposta.

Di seguito sono indicati tutti i parametri che andremo ad utilizzare nella nostra applicazione:

Nome del parametro	Valore	Descrizione
<i>q</i>	string	Ricerca i libri che contengono questa stringa di testo. E' possibile utilizzare altri 4 termini di ricerca più specifica, come per esempio: <ul style="list-style-type: none">• <i>intitle</i>: - Ritorna i libri che contengono la stringa di testo nel titolo;• <i>inauthor</i>: - Ritorna i libri che contengono la stringa di testo negli autori;• <i>inpublisher</i>: - Ritorna i libri che contengono la stringa di testo nell'editore;• <i>isbn</i>: - Ritorna i libri che contengono la stringa di testo nell'ISBN.
<i>startIndex</i>	int	La posizione da cui partire a ricevere i libri. L'indice del primo libro è 0.
<i>maxResults</i>	int	Il massimo numero di libri che si possono ricevere per ogni richiesta. Il valore di default è 10, il massimo disponibile è 40.
<i>orderBy</i>	string	Cambia l'ordine in cui sono disposti i libri ritornati, può assumere due valori:

- relevance - Ordina i libri per rilevanza;
- newest - Ordina i libri dal più recente al meno recente (in base alla data di pubblicazione).

Se, per esempio, vogliamo cercare i libri dell'autore *George Orwell*, partendo dal *primo*, visualizzandone 20 per pagina e ordinandoli per *rilevanza* l'URL sarà composto così:

```
https://www.googleapis.com/books/v1/volumes?q=inauthor:George%20Orwell&startIndex=0&maxResults=20&orderBy=relevance
```

La ricerca dei libri, essendo in tempo reale, utilizzerà AJAX per effettuare chiamate all'API. Nello specifico, alla funzione di JavaScript verrà passata come parametro la query di ricerca (q) inserita nell'input di testo. All'interno della funzione verranno ricavati i filtri e infine effettuata la chiamata.

```
function liveSearch(str, startIndex) {

    //se l'indice supera il massimo numero di risultati non lo incremento
    if(startIndex > parseInt(document.getElementById("qLast").innerHTML)){
        return;
    }

    //ritornare in cima alla pagina quando si va avanti o indietro
    document.body.scrollTop = 0;

    //blocca l'invio dell'input di ricerca (cancellerebbe il testo inserito
    //facendo risultare la ricerca nulla)
    window.addEventListener('keydown',function(e){

        if(e.keyIdentifier=='U+000A' || e.keyIdentifier=='Enter' || e.keyCode==13){
            if(e.target.nodeName=='INPUT'&&e.target.type=='text'){
                e.preventDefault();
                return false;
            }
        }
    },true);

    //ricavo il filtro di ricerca dal select-option
    var filter = document.getElementById("filter").value;

    switch(filter) {
        case "t":
            filter = "intitle:";
```

```

        break;
    case "a":
        filter = "inauthor:";  

        break;
    case "e":
        filter = "inpublisher:";  

        break;
    case "i":
        filter = "isbn:";  

        break;
}
//ricavo il filtro di ordinamento dal select-option
var order = document.getElementById("order").value;

switch(order) {
    case "ri":
        order = "relevance";
        break;
    case "re":
        order = "newest";
        break;
}
//ricavo il filtro di impaginazione dal select-option
var maxResults = document.getElementById("maxResults").value;

//inizializzazione GET request all'API in AJAX
var xmlhttp = new XMLHttpRequest();

//in caso la richiesta abbia un risultato positivo:
xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {

        console.clear();
        //lettura file JSON in risposta
        var json = JSON.parse(this.responseText);

        //pulisco il div che contiene i libri precedenti
        var flexbox = document.getElementById('resultTable');
        flexbox.innerHTML = "";

        //imposto l'indice di partenza e il massimo numero di risultati
        document.getElementById('qFirst').innerHTML = startIndex;
        document.getElementById('qLast').innerHTML = json.totalItems;

        //stampa dell'URL di chiamata (DEBUG)
        console.log("q=" + filter + str + "&orderBy=" + order + "&startIndex="
```

```

+ startIndex + "&maxResults=" + maxResults);

    try {

        //convertendo ogni oggetto "item" del JSON in un oggetto JavaScript
        "Book"
        for(var i=0; i < json.items.length; i++){

            var b = new Book(json.items[i]);

            console.log(json.items[i]);

            //funzione che gestisce la grafica (vedi graphic.js)
            addTableRow(b, flexbox);
        }

    } catch (error) {

        //gestione degli errori
        console.log(error);
        var errorText = document.createElement("p");
        var resultTable = document.getElementById("resultTable");
        errorText.style.textAlign = "center";
        errorText.style.color = "white";
        errorText.style.textShadow = "0px 1px 1px black";

        errorText.innerHTML = "Nessun risultato";

        resultTable.appendChild(errorText);
    }
}

//chiamata API
xmlhttp.open("GET", "https://www.googleapis.com/books/v1/volumes?q=" + filter
+ str + "&orderBy=" + order + "&startIndex=" + startIndex + "&maxResults=" +
maxResults, true);
xmlhttp.send();
}

```

Questo è il processo che viene eseguito ogni qualvolta si digitano qualcosa nella barra di ricerca.

6.2. Struttura della risposta

Se la chiamata avrà successo ([stato 200 di HTTP e HTTPS](#)), i dati riguardanti i libri saranno ritornati in formato strutturato JSON per cui sarà semplice convertirli in oggetti di JavaScript ed elaborarne i dati.

La struttura del file sarà la seguente:

```
▼ {  
    "kind": "books#volumes",  
    "totalItems": 1059,  
    ▶ "items": [ ... ] // 10 items  
}
```

Possiamo distinguere 3 attributi:

- *kind* - è una stringa che indica la tipologia di dati richiesta;
- *totalItems* - è un numero intero che indica la quantità di oggetti (libri) corrispondenti all'URL della chiamata;
- *items* - è un array di oggetti *item*, corrispondenti ai libri.

Ogni oggetto *item* contiene al suo interno un certo numero di attributi che variano tra stringhe, interi, array e oggetti.

La struttura generica di un libro (*item*) è la seguente:

```
▼ "items": [  
  ▶ {  
      "kind": "books#volume",  
      "id": "oGyxm0Yc_zMC",  
      "etag": "sbzFYHnuqu8",  
      "selfLink": "https://www.googleapis.com/books/v1/volumes/oGyxm0Yc_zMC",  
      ▶ "volumeInfo": {  
          "title": "Vocabolario universale italiano compilato a cura della Società Tipografica Tramater e Ci. Vol. 1. [-7.]",  
          "subtitle": "A-Bu",  
          "publishedDate": "1829",  
          ▶ "industryIdentifiers": [ ... ], // 1 item  
          ▶ "readingModes": { ... }, // 2 items  
          "printType": "BOOK",  
          "maturityRating": "NOT_MATURE",  
          "allowAnonLogging": false,  
          "contentVersion": "0.1.1.0.full.1",  
          ▶ "panelizationSummary": { ... }, // 2 items  
          ▶ "imageLinks": { ... }, // 2 items  
          "language": "it",  
          "previewLink": "http://books.google.it/books?id=oGyxm0Yc_zMC&pg=PA47&dq=a&hl=&cd=1&source=gbs_api",  
          "infoLink": "https://play.google.com/store/books/details?id=oGyxm0Yc_zMC&source=gbs_api",  
          "canonicalVolumeLink": "https://play.google.com/store/books/details?id=oGyxm0Yc_zMC"  
      },  
      ▶ "saleInfo": { ... }, // 4 items  
      ▶ "accessInfo": { ... }, // 10 items  
      ▶ "searchInfo": { ... } // 1 item  
  },
```

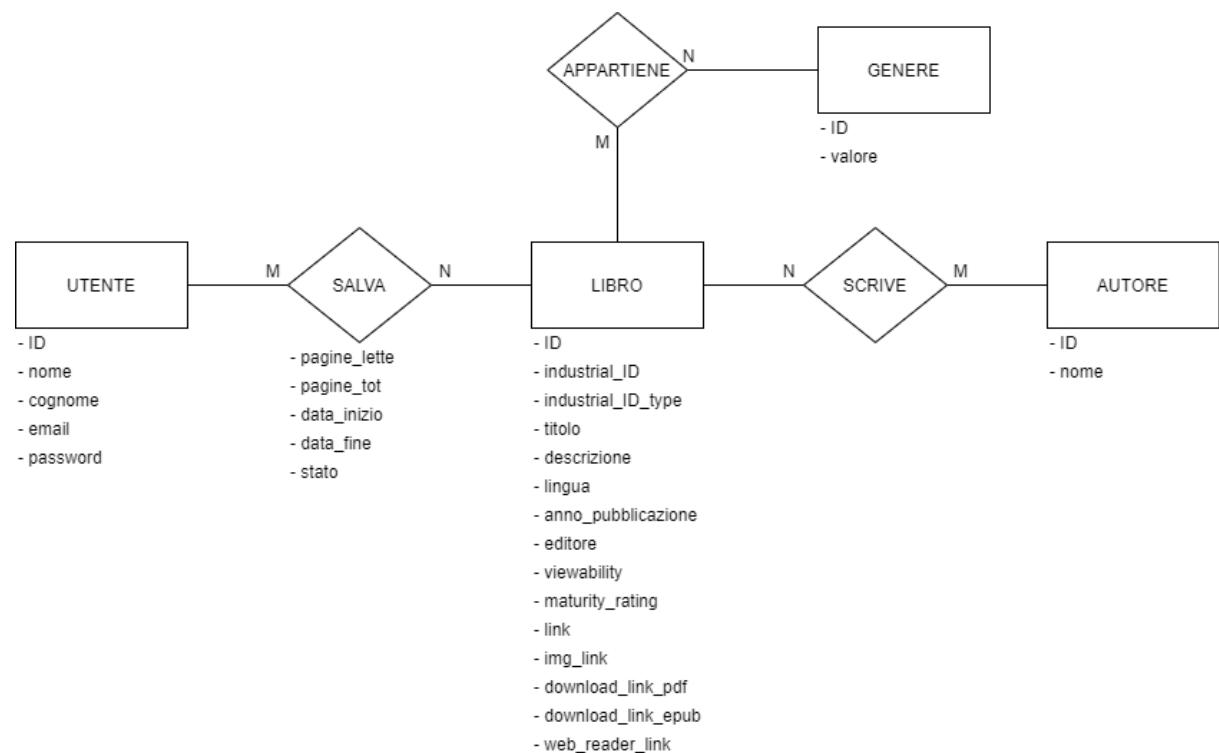
7. Progettazione del database

Come già ripetuto, è necessaria la creazione di un database relazionale per andare a gestire e memorizzare i dati presenti all'interno del nostro sito.

Lo scenario che si vuole rappresentare è il seguente:

“Ogni utente che accede all'interno del sito può salvare zero, uno o più libri all'interno della propria libreria personale. Ogni libro può essere salvato da più utenti diversi. Gli attributi del libro corrispondono a quelli estratti dal file JSON. Un libro può possedere più di un genere o più di un autore. Ogni libro salvato dagli utenti ha 4 attributi: pagine lette, pagine totali, data di inizio lettura, data di fine lettura e uno “stato di lettura” i cui valori possono essere: in lettura, in pausa o completato.”

7.1. Progettazione concettuale



Lo schema concettuale va a rappresentare lo scenario descritto.

In esso sono presenti 4 entità (*libro*, *genere*, *autore*, *utente*) e 3 associazioni (*appartiene*, *scrive*, *salva*). Il peso di ogni associazione è 2 (mettono in relazione 2 entità) e ognuna di loro ha cardinalità M a N.

Le relazioni sono ora descritte:

LIBRO - UTENTE

Un utente può salvare zero, uno o più libri.

Un libro può essere salvato da zero, uno o più utenti.

LIBRO - GENERE

Un libro può appartenere a zero, uno o più generi.

Ad un genere possono appartenere zero, uno o più libri.

LIBRO - AUTORE

Un autore può scrivere zero, uno o più libri.

Un libro può essere scritto da zero, uno o più autori.

Prima di passare alle regole di derivazione e bene introdurre due concetti base che riguardano i database relazionali che verranno nominati in seguito: le chiavi primarie e le chiavi esterne.

Una Primary Key (PK o chiave primaria) è composta da uno o più attributi e identifica in modo univoco ogni record di una tabella. Ogni attributo che può essere scelto come Primary Key è definito “chiave candidata”.

Una Foreign Key (FK o chiave esterna) è un vincolo che garantisce l'integrità dei dati e che rappresenta quella che nello schema concettuale è chiamata associazione.

Una chiave esterna è caratterizzata da una colonna figlia (di una tabella figlia) che si riferisce a una colonna madre (di una tabella madre). In pratica i valori assunti dalla Foreign Key della tabella figlia saranno limitati dai valori già esistenti nella Primary Key della tabella madre.

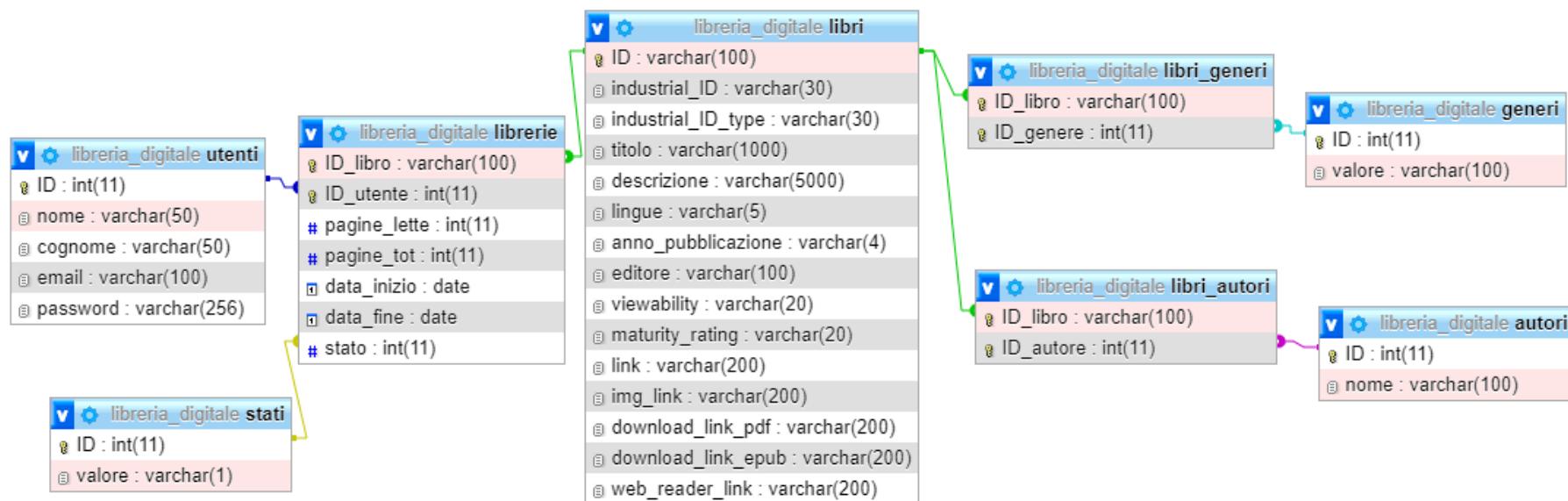
7.2. Regole di derivazione

Nel passaggio tra schema concettuale e schema logico avverranno 2 tipi di cambiamenti: ogni entità si trasformerà in una tabella e ogni associazione, facendo riferimento a 3 regole specifiche, si trasformerà a sua volta.

Ecco descritte le 3 regole (una per ogni tipologia di cardinalità):

- associazioni 1 a 1: nello schema logico verrà inserita una Foreign Key all'interno di una delle 2 tavole coinvolte;
- associazioni 1 a N: nello schema logico verrà inserita una Foreign Key all'interno della tabella a cui fa riferimento la N dell'associazione;
- associazioni M a N: nello schema logico verrà creata una nuova tabella associativa nella quale verranno inserite 2 Foreign Key, facenti riferimento rispettivamente a entrambe le tavole coinvolte.

7.3. Progettazione logica



Seguendo le regole di derivazione precedentemente descritte si andrà a comporre lo schema logico del database (sopra rappresentato). Si possono identificare 8 tabelle composte da:

- **LIBRI** - con PK *ID* e rispettivi attributi (i cui nomi fanno riferimento al JSON):
 - *industrial_ID*: ID industriale dei libri;
 - *industrial_ID_type*: tipo di ID industriale (ISBN_13, HARVARD, ecc...);
 - *titolo*: titolo del libro;
 - *descrizione*: descrizione del libro;
 - *lingue*: codice della lingua in cui è scritto (it, en, es, ecc...);
 - *anno_pubblicazione*: anno di pubblicazione dell'edizione specifica del libro;
 - *editore*: editore del libro;
 - *viewability*: grado di visibilità dell'estratto (NONE, PARTIAL o FULL);
 - *maturity_rating*: indice se il libro è destinato a un pubblico adulto;
 - *link*: link per informazioni sul libro;
 - *img_link*: link dell'immagine di copertina;
 - *download_link_pdf*: link pdf;
 - *download_link_epub*: link epub;
 - *web_reader_link*: link per entrare nell'applicativo di Google per leggere gli estratti;:
- **UTENTI** - con PK *ID* (con AUTO_INCREMENT) e rispettivi attributi (la lunghezza della password è data dalla funzione di hash utilizzata, SHA256, che produrrà un hash di 256 byte);
- **STATO** - con PK *ID* (con AUTO_INCREMENT) e rispettivi attributi;
- **LIBRERIE** - tabella associativa con PK composta da FK1 (*ID_libro*) e FK2 (*ID_utente*) e rispettivi attributi (l'attributo stato, che può assumere 3 valori, è un'ulteriore FK che fa riferimento alla tabella omonima);
- **GENERI** - PK *ID* (con AUTO_INCREMENT) e rispettivi attributi;
- **LIBRI_GENERI** - tabella associativa con PK composta da FK1 (*ID_libro*) e FK2 (*ID_genere*);
- **AUTORI** - PK *ID* (con AUTO_INCREMENT) e rispettivi attributi
- **LIBRI_AUTORI** - tabella associativa con PK composta da FK1 (*ID_libro*) e FK2 (*ID_autore*).

7.4. Normalizzazione

Per normalizzazione si intende un insieme di regole che vengono applicate a un database per evitare ridondanza e incoerenza tra i dati.

Le principali regole della normalizzazione sono chiamate forme normali e sono descritte qui di seguito.

PRIMA FORMA NORMALE (1FN)

Si dice che un database è in prima forma normale se:

- ciascun attributo è definito su un dominio con valori atomici (attributi indivisibili);
- ogni tabella deve avere una chiave primaria.

SECONDA FORMA NORMALE (2FN)

Si dice che un database è in seconda forma normale se:

- è in prima forma normale;
- ogni attributo non primo dipende funzionalmente dalla chiave primaria.

TERZA FORMA NORMALE (3FN)

Si dice che un database è in terza forma normale se:

- è in seconda forma normale;
- ogni attributo non primo non può dipendere da altri attributi non primi (quindi dipende esclusivamente dalla chiave).

Si vuole specificare che all'interno della tabella *autori* si indica con l'attributo *nome* sia nome che cognome di un autore. Si è dovuto considerare questo attributo atomicamente indivisibile perché, prendendo i dati da un file JSON che si riferisce all'autore con un solo attributo, non è stato possibile stabilire un criterio per dividere il nome dal cognome (si pensi a quante variazioni possono esistere al mondo). Per esempio:

```
    "authors": [
        ...
        "Antonio Tommaso Barbaro"
    ],
```

Per concludere, ogni attributo non primo nel nostro database dipende in modo diretto dalla chiave primaria e non da altri attributi non primi. Per cui possiamo considerare il nostro database normalizzato.

7.5. Script di creazione in SQL

```
--  
-- Creazione della tabella `autori`  
--  
  
CREATE TABLE `autori` (  
    `ID` int(11) NOT NULL AUTO_INCREMENT,  
    `nome` varchar(100) DEFAULT NULL  
)  
  
--  
-- Creazione della tabella `generi`  
--  
  
CREATE TABLE `generi` (  
    `ID` int(11) NOT NULL AUTO_INCREMENT,  
    `valore` varchar(100) DEFAULT NULL  
)  
  
--  
-- Creazione della tabella `librerie`  
--  
  
CREATE TABLE `librerie` (  
    `ID_libro` varchar(100) NOT NULL,  
    `ID_utente` int(11) NOT NULL,  
    `pagine_lette` int(11) DEFAULT NULL,  
    `pagine_tot` int(11) DEFAULT NULL,  
    `data_inizio` date DEFAULT NULL,  
    `data_fine` date DEFAULT NULL,  
    `stato` int(11) NOT NULL  
)  
  
ALTER TABLE `librerie`  
    ADD CONSTRAINT `librerie_ibfk_1` FOREIGN KEY (`ID_libro`) REFERENCES  
    `libri` (`ID`) ON DELETE CASCADE,  
    ADD CONSTRAINT `librerie_ibfk_2` FOREIGN KEY (`ID_utente`) REFERENCES  
    `utenti` (`ID`) ON DELETE CASCADE,  
    ADD CONSTRAINT `librerie_ibfk_3` FOREIGN KEY (`stato`) REFERENCES `stati`  
    (`ID`);  
--
```

```

-- Creazione della tabella `libri`
--

CREATE TABLE `libri` (
  `ID` varchar(100) NOT NULL,
  `industrial_ID` varchar(30) DEFAULT NULL,
  `industrial_ID_type` varchar(30) DEFAULT NULL,
  `titolo` varchar(1000) DEFAULT NULL,
  `descrizione` varchar(5000) DEFAULT NULL,
  `lingue` varchar(5) DEFAULT NULL,
  `anno_pubblicazione` varchar(4) DEFAULT NULL,
  `editore` varchar(100) DEFAULT NULL,
  `viewability` varchar(20) DEFAULT NULL,
  `maturity_rating` varchar(20) DEFAULT NULL,
  `link` varchar(200) DEFAULT NULL,
  `img_link` varchar(200) DEFAULT NULL,
  `download_link_pdf` varchar(200) DEFAULT NULL,
  `download_link_epub` varchar(200) DEFAULT NULL,
  `web_reader_link` varchar(200) DEFAULT NULL
)

-- 
-- Creazione della tabella `libri_autori`
-- 

CREATE TABLE `libri_autori` (
  `ID_libro` varchar(100) NOT NULL,
  `ID_autore` int(11) NOT NULL
)

ALTER TABLE `libri_autori`
  ADD CONSTRAINT `libri_autori_ibfk_1` FOREIGN KEY (`ID_libro`) REFERENCES
`libri` (`ID`) ON DELETE CASCADE,
  ADD CONSTRAINT `libri_autori_ibfk_2` FOREIGN KEY (`ID_autore`) REFERENCES
`autori` (`ID`) ON DELETE CASCADE;

-- 
-- Creazione della tabella `libri_generi`
-- 

CREATE TABLE `libri_generi` (
  `ID_libro` varchar(100) NOT NULL,
  `ID_genere` int(11) NOT NULL
)
ALTER TABLE `libri_generi`
  ADD CONSTRAINT `libri_generi_ibfk_1` FOREIGN KEY (`ID_libro`) REFERENCES
`libri` (`ID`) ON DELETE CASCADE,
  ADD CONSTRAINT `libri_generi_ibfk_2` FOREIGN KEY (`ID_genere`) REFERENCES

```

```

`generi` (`ID`) ON DELETE CASCADE;

-- 
-- Creazione della tabella `stati`
-- 

CREATE TABLE `stati` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `valore` varchar(1) NOT NULL
)
;

-- 
-- Creazione della tabella `utenti`
-- 

CREATE TABLE `utenti` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `nome` varchar(50) DEFAULT NULL,
  `cognome` varchar(50) DEFAULT NULL,
  `email` varchar(100) DEFAULT NULL,
  `password` varchar(256) DEFAULT NULL
)
-----
-- 
-- Indici per la tabella `autori`
-- 

ALTER TABLE `autori`
  ADD PRIMARY KEY (`ID`);

-- 
-- Indici per la tabella `generi`
-- 

ALTER TABLE `generi`
  ADD PRIMARY KEY (`ID`);

-- 
-- Indici per la tabella `librerie`
-- 

ALTER TABLE `librerie`
  ADD PRIMARY KEY (`ID_libro`, `ID_utente`),
  ADD KEY `librerie_ibfk_1` (`ID_libro`),
  ADD KEY `librerie_ibfk_2` (`ID_utente`),
  ADD KEY `stato` (`stato`);

-- 
-- Indici per la tabella `libri`
-- 

```

```

ALTER TABLE `libri`
  ADD PRIMARY KEY (`ID`);

-- 
-- Indici per la tabella `libri_autori`
-- 

ALTER TABLE `libri_autori`
  ADD PRIMARY KEY (`ID_libro`, `ID_autore`),
  ADD KEY `libri_autori_ibfk_2` (`ID_autore`);

-- 
-- Indici per la tabella `libri_generi`
-- 

ALTER TABLE `libri_generi`
  ADD PRIMARY KEY (`ID_libro`, `ID_genere`),
  ADD KEY `libri_generi_ibfk_2` (`ID_genere`);

-- 
-- Indici per la tabella `stati`
-- 

ALTER TABLE `stati`
  ADD PRIMARY KEY (`ID`);

-- 
-- Indici per la tabella `utenti`
-- 

ALTER TABLE `utenti`
  ADD PRIMARY KEY (`ID`);

```

7.6. Principali query utilizzate

Per leggere i dati all'interno del database verranno utilizzate le query di MySQL. Qui di seguito andremo a descrivere le principali SELECT che utilizzeremo all'interno del sito.

La prima query analizzata è eseguita nella libreria personale. PHP si occuperà di andare a leggere i dati di ogni libro salvato dall'utente nel database per poi stamparlo all'interno della pagina visualizzata.

La SELECT dovrà quindi includere una INNER JOIN (ovvero prendere i valori di altre tabelle in base a una determinata condizione) con la tabella *librerie* e avrà come vincolo l'ID del libro. In questo modo verranno selezionati tutte le informazioni di tutti i libri nella tabella *librerie*.

Ora dobbiamo specificare l'utente: andremo a identificare l'utente con l'id della sessione (che corrisponde all'ID utente) creata durante il login. Quindi andremo a

filtrare solo i libri salvati dall'utente il cui ID corrisponde all'id di sessione (in poche parole l'utente loggato).

Infine ordiniamo i libri in modo crescente in base allo stato, in modo che i libri *in lettura* appariranno sempre in cima alla libreria per velocizzare il loro raggiungimento.

```
SELECT * FROM libri
INNER JOIN librerie
ON librerie.ID_libro = libri.ID
WHERE librerie.ID_utente= '.$_SESSION["id"].'
ORDER BY librerie.stato ASC;
```

La seconda query analizzata è quella che serve ad autenticare il login dell'utente. PHP dovrà controllare che le credenziali inviate in [POST](#) dal form, corrispondano a una coppia di valori nella tabella *utente*.

Si andrà a fare una SELECT che selezionerà l'ID del record nella tabella *utenti* che avrà come email e come password (con applicato l'hash SHA256) quelle inserite nel form di login.

Se ci sarà una corrispondenza l'ID selezionato verrà usato come parametro da usare nella creazione della sessione. Se non ci sarà una corrispondenza le credenziali saranno errate e l'utente dovrà riprovare l'inserimento.

```
SELECT id
FROM utenti
WHERE email = "'.$_POST["email"].'"
AND password = "'.$hash('sha256', $_POST["password"]).'"
```

L'ultima query utilizzata serve a verificare se all'indirizzo email inserito durante la registrazione ne corrisponde già uno all'interno del database.

Questa SELECT viene eseguita in modo asincrono con AJAX e se troverà una corrispondenza non permetterà l'invio del form di registrazione (evitando quindi la creazione di più account con la stessa email).

```
SELECT ID
FROM utenti
WHERE email='".$email.'"
```

8. Architettura della rete

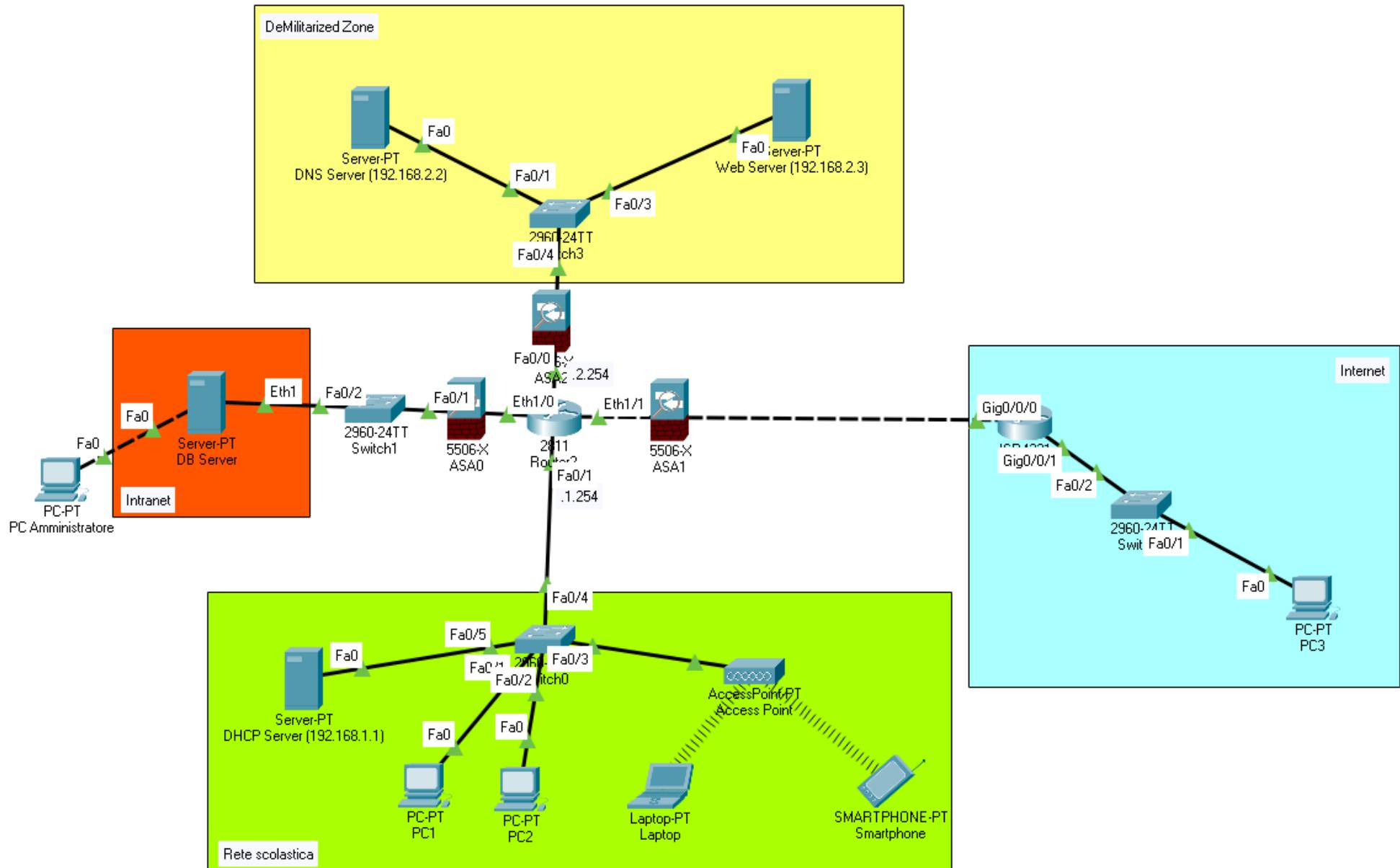
Nello scenario che andremo ad analizzare ipotizziamo che il Web Server sul quale è hostato il nostro sito si trovi all'interno della scuola. Andremo a prevedere tutte quelle misure che renderanno funzionale e sicuro l'accesso al sito sia da parte della rete interna scolastica che dall'esterno (Internet).

8.1. Schema logico

Lo schema logico della rete è composto da 3 parti principali: la rete scolastica, la DMZ e la Intranet. Inoltre è stata creata una rete esterna per simulare un accesso generico da un dispositivo in Internet.

Ecco descritte le 3 parti nel dettaglio:

- la rete scolastica accessibile agli alunni, ai docenti e al personale, divisa in rete via cavo (Ethernet) e WiFi (access-point);
- Intranet, la rete privata non accessibile dall'esterno, nella quale è contenuto il DB Server che sarà isolato da tutti gli altri dispositivi e potrà essere acceduto unicamente dal Web Server e dal PC dell'amministratore, direttamente connesso;
- DMZ (DeMilitarized Zone) in cui saranno posizionati e protetti il Web Server e il DNS Server.



8.2. Piano di indirizzamento

Nome rete	Nome dispositivo	Ind. IP e SM	Default Gateway
Rete scolastica	PC1	DHCP	DHCP
Rete scolastica	PC2	DHCP	DHCP
Rete scolastica	DHCP Server	192.168.1.1/24	192.168.1.254
Rete scolastica	Laptop	DHCP	DHCP
Rete scolastica	Smartphone	DHCP	DHCP
Intranet	DB Server	192.168.3.1/24	192.168.3.254
-	PC Amministratore	-	-
DMZ	DNS Server	192.168.2.2/24	192.168.2.254
DMZ	Web Server	192.168.2.3/24	192.168.2.254
-	Router2	int Fa0/0 - 192.168.2.254/24	
		int Fa0/1 - 192.168.1.254/24	
		int Eth1/0 - 192.168.3.254/24	
		int Eth1/1 - 200.100.50.1/30	
-	Router1	int Gig0/0/0 - 200.100.50.2/30	-
		int Gig0/0/1 - 192.168.10.254/24	
Internet	PC3	192.168.10.1/24	192.168.10.254

9. Configurazione servizi e NAT

I servizi implementati nella nostra rete sono 3: DHCP, DNS e Web.

Sui router, inoltre, verranno applicate delle tecniche di mascheramento degli indirizzi privati, NAT statico e NAT overload (PAT), per ridurre il consumo di indirizzi pubblici IPv4.

9.1. DHCP

Il DHCP (Dynamic Host Configuration Protocol) verrà utilizzato per assegnare e gestire in modo dinamico gli indirizzi IP privati della rete WiFi ed Ethernet accessibile da studenti, dai docenti e dal personale.

Questo protocollo è stato implementato per non dover assegnare manualmente a ogni dispositivo un indirizzo di rete. Il discorso vale soprattutto per la WLAN (Wireless LAN) che, permettendo l'accesso a una moltitudine di dispositivi diversi, renderebbe impraticabile la configurazione manuale.

Per configurare il DHCP sul rispettivo server, si creerà una nuova *pool di indirizzi* da assegnare agli host.

DHCP

Interface	FastEthernet0	Service	<input checked="" type="radio"/> On	<input type="radio"/> Off			
Pool Name	rete-scolastica						
Default Gateway	192.168.1.254						
DNS Server	192.168.2.2						
Start IP Address :	192	168	1	0			
Subnet Mask:	255	255	255	0			
Maximum Number of Users :	256						
TFTP Server:	0.0.0.0						
WLC Address:	0.0.0.0						
<input type="button" value="Add"/>		<input type="button" value="Save"/>	<input type="button" value="Remove"/>				
Pool Name	Default Gateway	DNS Server	Start IP Address	Subnet Mask	Max User	TFTP Server	WLC Address
rete-scolastica	192.168.1.1	192.168.2.1	192.168.1.1	255.255.21	256	0.0.0.0	0.0.0.0

La pool sarà nominata *rete-scolastica* e verrà configurata come mostrato in figura.

Per terminare la configurazione si abiliterà il DHCP all'interno dei singoli host. Successivamente questi ultimi e il server inizieranno lo scambio 4 messaggi.

Il DHCP server verrà identificato nel dominio di broadcast della rete (DHCP discover) e proporrà un indirizzo IP (DHCP offer), che dovrà essere disponibile all'interno della pool precedentemente configurata. L'host confermerà l'indirizzo proposto e chiederà al server l'autorizzazione per iniziare ad utilizzarlo (DHCP request). Quest'ultimo risponderà con un messaggio di conferma (DHCP ack) e l'host potrà iniziare ad utilizzare l'indirizzo IP per il tempo indicato. Al termine di questo tempo (TTL, Time To Live) gli indirizzi dovranno essere nuovamente confermati o eventualmente aggiornati dal server.

E' stato scelto di utilizzare un unico indirizzo di rete di classe C perché, sia per la rete Ethernet che per la rete WiFi, i permessi di accesso al resto della rete e ad Internet risulterebbero uguali (per cui una distinzione con l'utilizzo di VLAN, per esempio, sarebbe inutile).

Se in futuro fosse necessario un numero di indirizzi assegnabili maggiore, si potrà applicare il [supernetting](#) alla rete già esistente (riconfigurando solamente la subnet mask), senza effettuare ulteriori cambiamenti che potrebbero rendere inutilizzabile la rete per un periodo di tempo più o meno lungo.

9.2. Web Server

Per configurare il servizio Web su cui verranno ospitate le pagine del nostro sito, e che andrà ad interagire col DB Server, andremo ad abilitare i servizi HTTP (HyperText Transfer Protocol) e HTTPS (HTTP Secure). Questi protocolli risponderanno alle richieste di accesso del client e gli invieranno le pagine HTML.

HTTP

HTTP

On Off

HTTPS

On Off

File Manager

File Name	Edit	Delete
1 copyrights.html	(edit)	(delete)
2 cscoptlogo177x111.jpg		(delete)
3 helloworld.html	(edit)	(delete)
4 image.html	(edit)	(delete)
5 index.html	(edit)	(delete)

Il Web Server verrà posizionato nella DMZ per proteggerlo da possibili attacchi come per esempio DDoS: il Web Server viene inondato da un gran numero di richieste e, non essendo in grado di gestirle tutte, causa rallentamenti o addirittura interrompe il suo funzionamento.

9.3. DNS Server

Per rendere accessibile il nostro sito web inserendo all'interno della barra di ricerca un nome di dominio, avremo bisogno di configurare il DNS (Domain Name System) Server.

Esso avrà il compito di associare l'indirizzo IP pubblico (vedi NAT statico) del nostro Web Server (110.30.0.1) al nome di dominio (per esempio “digilib.altervista.org”).

DNS

DNS Service On Off

Resource Records

Name	Type
digilib.altervista.org	A Record

Address 110.30.0.1

No.	Name	Type	Detail
0	digilib.altervista.org	A Record	110.30.0.1

Se un dispositivo quindi inserirà nell'URL del browser “digilib.altervista.org” verrà mandata una richiesta al DNS Server che risponderà con l'indirizzo IP del Web Server, corrispondente al dominio inserito, a cui inoltrare la richiesta HTTP (o HTTPS). Ovviamente questa è una semplificazione dell'effettivo processo che coinvolgerebbe tutto il sistema gerarchico DNS (DNS root, TLD e authoritative).

Il DNS server sarà posizionato all'interno della DMZ per evitare possibili attacchi come per esempio il DNS spoofing: un possibile utente malintenzionato sostituisce l'indirizzo IP associato al dominio “digilib.altervista.org” con l'indirizzo di un altro server, reindirizzando gli utenti dove vuole e causando disguidi e possibili truffe.

9.4. NAT statico

Il DNS server e il Web Server appena configurati saranno server pubblici, ovvero accessibili anche al di fuori della nostra rete scolastica, perciò hanno bisogno di un indirizzo statico e pubblico che permetta di essere raggiungibili da qualsiasi dispositivo presente in Internet.

Per assegnare tali indirizzi andremo ad utilizzare il NAT (Network Address Translation) statico che tradurrà i nostri indirizzi privati in indirizzi pubblici al di fuori della rete.

Il NAT statico dovrà essere configurato all'interno del Router2 in cui andremo a distinguere la porta interna (inside) e la porta esterna (outside).

I comandi che utilizzeremo sono i seguenti:

```
Router(config)# ip nat inside source static 192.168.3.3 110.30.0.1
Router(config)# ip nat inside source static 192.168.3.2 110.30.0.2
Router(config)# int Fa0/0
Router(config-if)# ip nat inside
Router(config)# int Eth1/1
Router(config-if)# ip nat outside
```

Al termine della configurazione avremo la seguente corrispondenza:

Nome	Indirizzo privato	Indirizzo pubblico
DNS Server	192.168.2.2	110.30.0.2
Web Server	192.168.2.3	110.30.0.1

9.5. PAT

Per permettere alla rete scolastica di uscire dalla rete si cercherà di sprecare meno indirizzi IP pubblici possibili. Per il raggiungimento di tale scopo andremo ad usufruire del PAT (Port Address Translation) o NAT overload.

Grazie a questo servizio, che verrà nuovamente configurato sul Router2, andremo ad utilizzare un solo indirizzo pubblico (quello dell'interfaccia esterna del router, 200.100.50.1) e ogni dispositivo della nostra rete privata sarà identificato dal numero di porta associato ad esso a livello trasporto.

I comandi che verranno utilizzati per la configurazione del PAT sono i seguenti:

```
Router(config)# access-list 10 permit 192.168.1.0 0.0.0.255
```

```

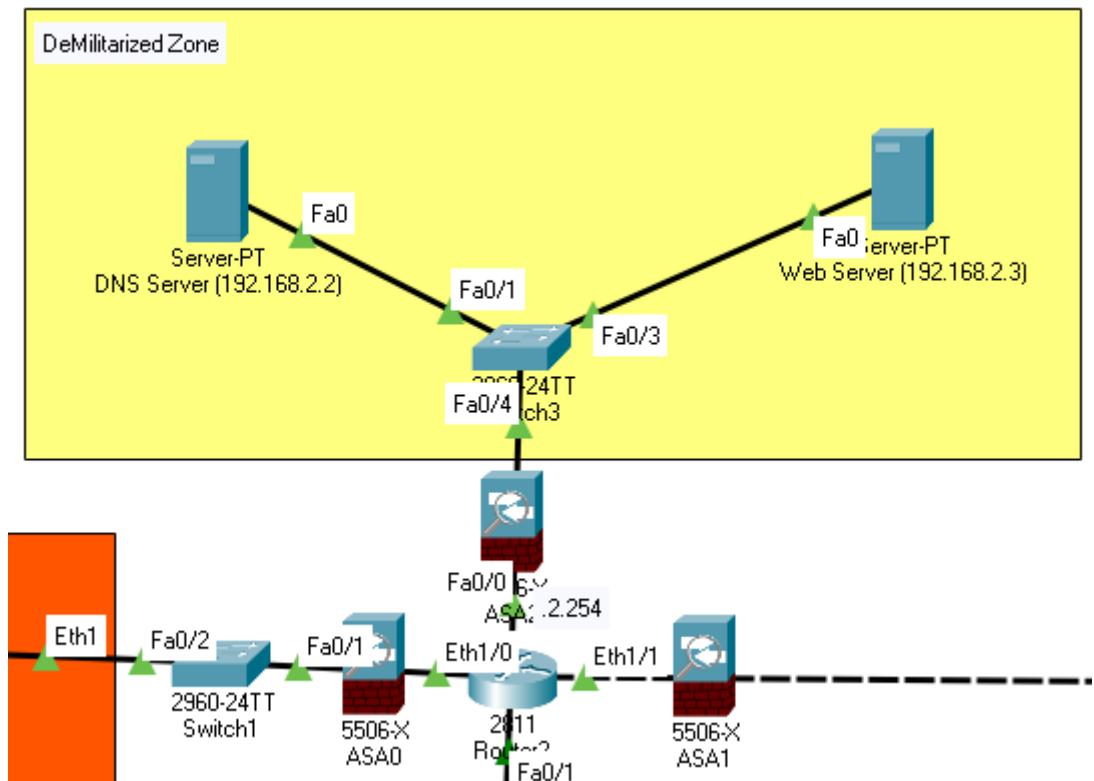
Router(config)# ip nat inside source list 10 int Eth1/1 overload
Router(config)# int Fa0/1
Router(config-if)# ip nat inside
Router(config)# int Eth1/1
Router(config-if)# ip nat outside

```

10. Sicurezza della rete

10.1. DMZ

La DMZ (DeMilitarized Zone) verrà stabilita per proteggere i server da possibili attacchi, sia interni che esterni alla rete.



Questa zona sarà delimitata da una zona “cuscinetto”, con agli estremi 2 ACL, che filtreranno i messaggi in entrata.

Le ACL (Access Control List) sono dei veri e propri gruppi di regole che permettono o negano l’accesso a un determinato tipo di messaggi o a determinati indirizzi.

Più specificatamente, utilizzeremo le ACL estese (in grado di filtrare specifici protocolli) e, in particolare, andremo ad agire sul livello di trasporto.

Inizieremo a configurare l'ACL esterna alla rete, ovvero quella che filtrerà i messaggi provenienti da Internet. Essa dovrà permettere il passaggio ai pacchetti DNS, HTTP e HTTPS per rendere accessibile da Internet il sito web e bloccare qualsiasi altro tipo di pacchetto malevolo o non necessario.

Il protocollo DNS si appoggia a livello trasporto su UDP, porta 53.

I protocolli HTTP e HTTPS si appoggiano su TCP, porte 80 e 443.

Le regole che inseriremo nell'ACL esterna sono:

```
Router(config)# access-list 102 permit tcp any host 110.30.0.1 eq 80
Router(config)# access-list 102 permit tcp any host 110.30.0.1 eq 443
Router(config)# access-list 102 permit udp any host 110.30.0.2 eq 53
```

Infine, andremo a indicare l'interfaccia e il verso sulla quale interverrà (in questo caso il controllo avverrà in entrata).

```
Router(config)# int Eth1/1
Router(config-if)# ip access-group 102 in
```

Successivamente passeremo alla configurazione dell'ACL interna, posta prima della DMZ. Essa dovrà svolgere la stessa funzione dell'ACL precedentemente configurata ma filtrando i pacchetti provenienti dalla rete interna. Inoltre dovrà permettere l'ulteriore passaggio dei pacchetti inviati dal DB Server al Web Server.

Le regole da inserire sono:

```
Router(config)# access-list 103 permit tcp any host 110.30.0.1 eq 80
Router(config)# access-list 103 permit tcp any host 110.30.0.1 eq 443
Router(config)# access-list 103 permit udp any host 110.30.0.2 eq 53
Router(config)# access-list 103 permit ip host 192.168.2.3 host 192.168.2.3
```

L'interfaccia sulla quale verrà applicata, in uscita, sarà la seguente:

```
Router(config)# int Fa0/0
Router(config-if)# ip access-group 103 out
```

Infine, un'ultima ACL verrà configurata per limitare gli accessi al database server. Ad esso potranno accedere solamente il Web Server, per il quale dovremmo implementare delle regole, e il PC dell'amministratore che, essendo direttamente connesso e utilizzando il protocollo SSH per la configurazione del database da remoto non necessita regole.

Configureremo quindi le ACL per permettere ai pacchetti che avranno come mittente il Web Server di passare:

```
Router(config)# access-list 101 permit ip host 192.168.2.3 host 192.168.3.1
```

Essa agirà in uscita dalla porta sulla quale sarà configurata:

```
Router(config)# int Eth1/0
Router(config-if)# ip access-group 101 out
```

11. Protocolli di comunicazione

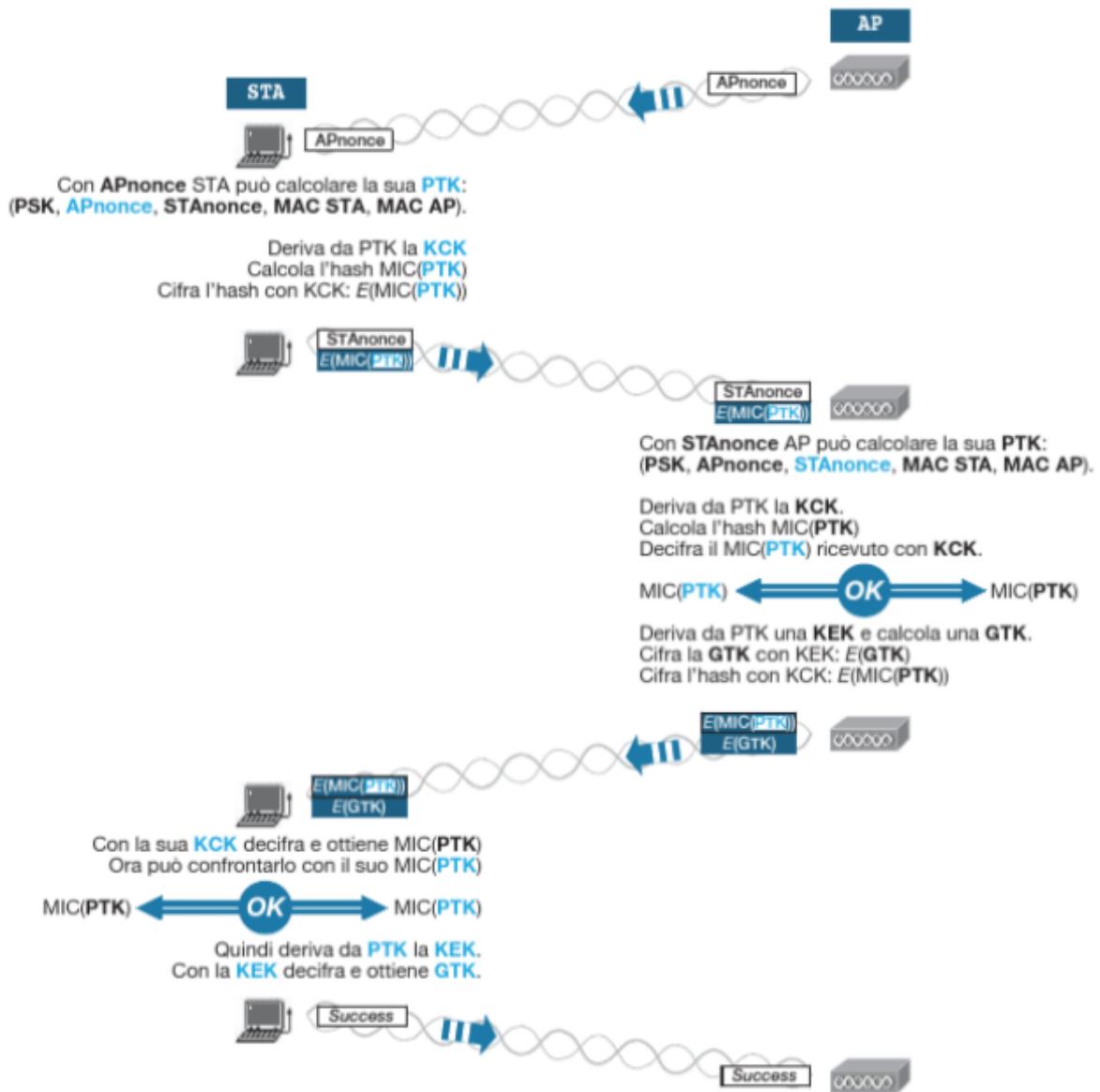
11.1. WPA2-PSK

L'access-point che utilizzeremo per configurare e per gestire la rete WiFi della scuola implementerà il protocollo WPA2 in modalità personal per gestire le comunicazioni.

La modalità personal utilizza una chiave condivisa di accesso (Pre Shared Key), la password del WiFi, che verrà utilizzata per autenticare il client e l'access-point una volta connessi.

Port 1			
Port Status	<input checked="" type="checkbox"/> On		
SSID	AvoSpot		
2.4 GHz Channel	6		
Coverage Range (meters)	140.00		
Authentication			
<input type="radio"/> Disabled	<input type="radio"/> WEP	WEP Key	
<input type="radio"/> WPA-PSK	<input checked="" type="radio"/> WPA2-PSK	PSK Pass Phrase	AvoSpotPsw
		User ID	
		Password	
Encryption Type		AES	

Configureremo l'SSID della rete come "AvoSpot" e la PSK come "AvoSpotPsw" (modificabile a proprio piacimento). Successivamente, potremo connettere i nostri dispositivi alla rete. Nel momento della connessione con l'access-point verrà effettuato uno scambio a 4 fasi (four-way-handshake) per autenticare sia il dispositivo (STA) che l'access-point (AP) e negoziare le chiavi di sessione con le quali verrà cifrata la comunicazione: PTK (chiave primaria di sessione) e GTK (chiave secondaria di sessione).



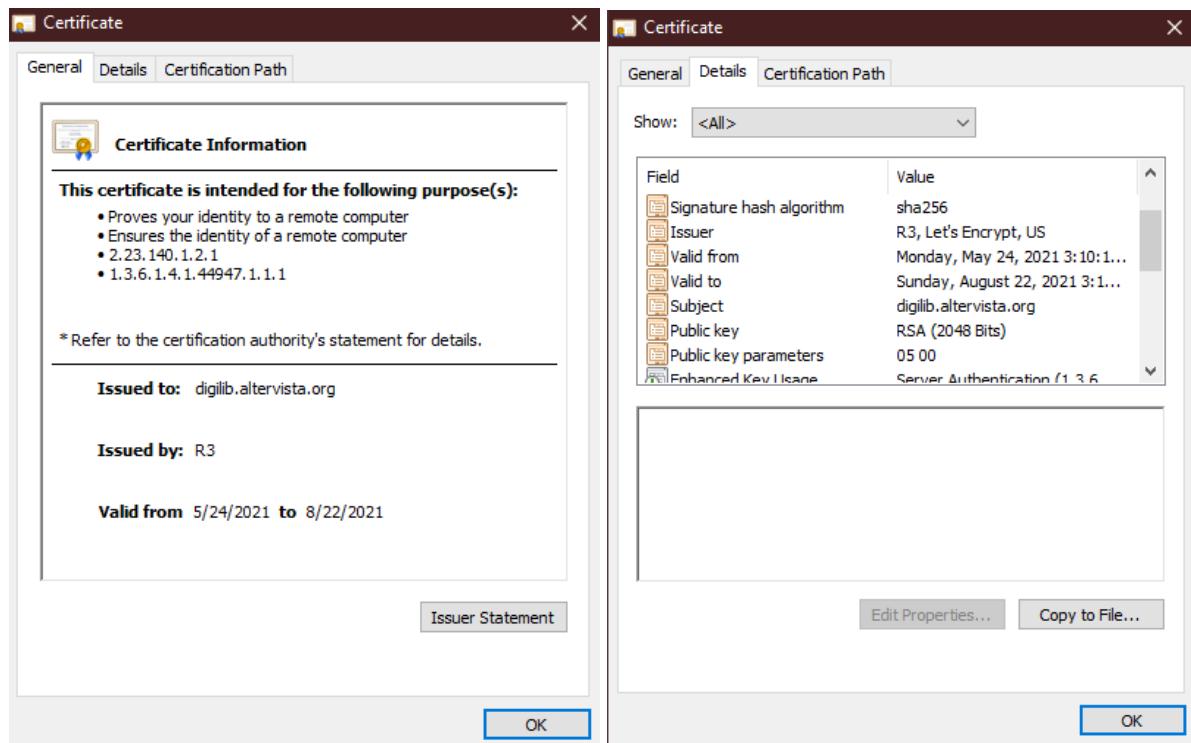
11.2. HTTPS

HTTPS (HyperText Transfer Protocol Secure) è il protocollo di trasferimento crittografato che va a regolare la comunicazione tra client, quindi il browser, e il Web Server, nel nostro caso Altervista.

Questo protocollo svolge 2 principali funzioni:

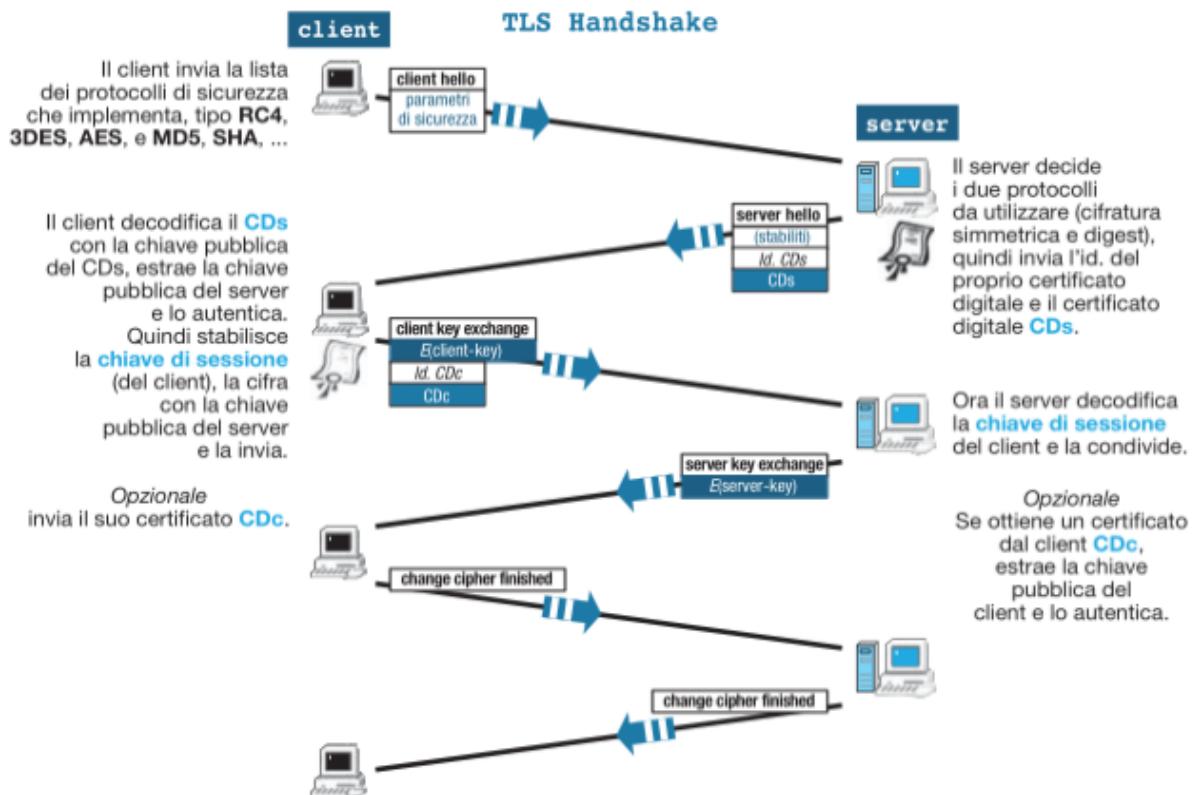
- mantiene la segretezza tra client e server impedendo a terzi di intercettare e leggere il contenuto di un eventuale messaggio;
- all'inizio della comunicazione il Web Server spedisce un certificato al client per confermare l'affidabilità del dominio.

E' possibile visualizzare il certificato assegnato al nostro sito, con le informazioni ad esso connesse, cliccando sul lucchetto accanto alla barra degli indirizzi.

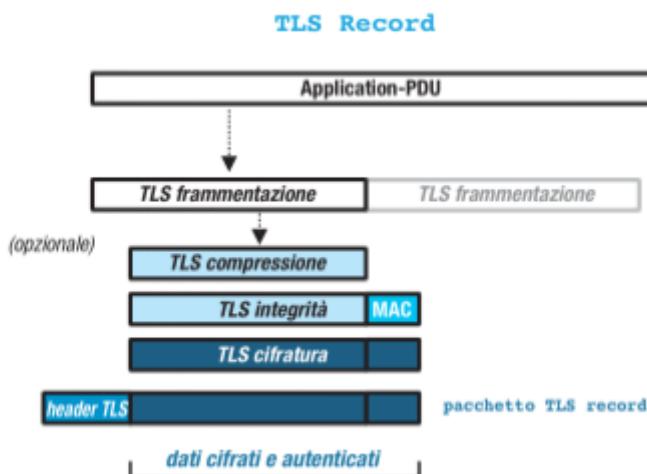


Da questo certificato è possibile ricavare la [Certification Authority](#) che lo ha rilasciato, la data di scadenza entro il quale dovrà essere rinnovato e tutte le chiavi e gli algoritmi di sicurezza utilizzati (per esempio SHA256 come hashing e RSA come algoritmo di cifratura simmetrica).

Il controllo del certificato appena inizializzato la connessione al server prende il nome di handshake ed è gestito dal protocollo SSL/TLS. Al termine di questo scambio di messaggi sia il client che il server possederanno la chiave di sessione simmetrica che utilizzeranno per cifrare e decifrare i messaggi.



Una volta terminata la fase di handshake inizierà la seconda fase, chiamata record, in cui il messaggio HTTP viene frammentato, compresso e infine cifrato e autenticato.



12. Prevenzione interruzioni servizi

Per garantire il continuo funzionamento dei servizi offerti dalla nostra rete, avremmo bisogno di prevenire qualsiasi tipo di evento ostile.

La sicurezza della rete, descritta in precedenza, è il principale metodo di protezione da attacchi informatici ma, una buona configurazione, dovrebbe prevedere anche eventuali attacchi fisici (allagamenti, malfunzionamento di un server, ecc...).

In questo caso, la prima norma da seguire è garantire il corretto posizionamento dei server: devono essere posti in luoghi coperti che non siano a rischio allagamento, arieggiati (in prevenzione di un possibile surriscaldamento) e asciutti.

La seconda norma, riguarda il posizionamento di server **mirror** che sostituiscano i principali in caso di guasti. Dovremmo quindi prevederne uno per il Web Server, il DNS Server e il DB Server. In questo modo, nel caso venga effettuata una richiesta al Web Server non funzionante, quest'ultima verrà reindirizzata al Web Server mirror.

L'ultima norma consiglia di utilizzare periodicamente la funzione di backup, per esempio all'interno del DB Server, per salvaguardare i dati e non perderli in caso di malfunzionamenti.

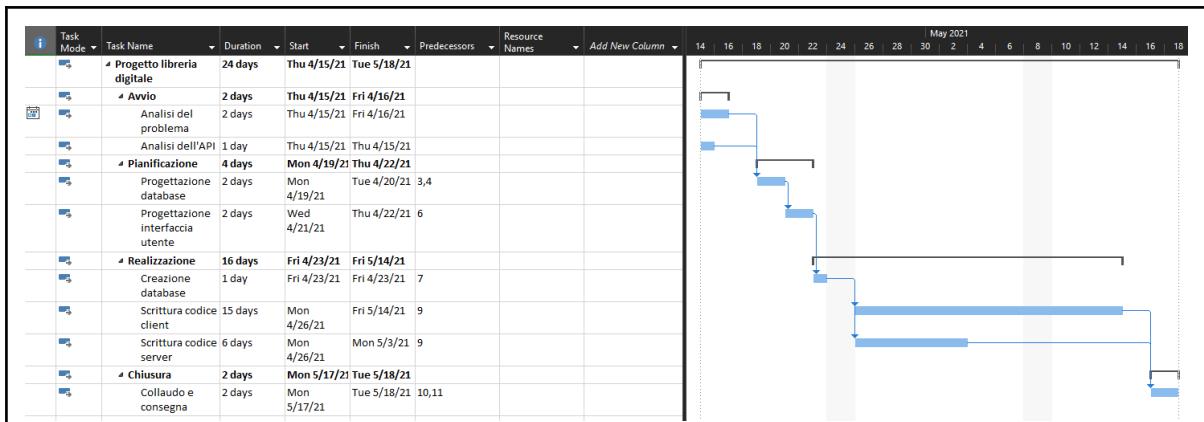
13. Gestione del progetto

13.1. Project Charter

Il Project Charter o piano del progetto è stato stilato all'inizio della progettazione per descriverne i caratteri generali.

Progetto: libreria digitale	Codice progetto: LD21
Data: 15 Aprile 2021	Revisione: 0
Cliente: nessuno	Sponsor: nessuno

1. OBIETTIVI Realizzazione di un sito web per offrire agli utenti una fonte di dati vasta tra cui cercare libri, visualizzandone le informazioni, e potendoli salvare all'interno di una libreria personale associata al proprio account.
2. PRINCIPALI DELIVERABLE Avvio: analisi del problema Avvio: analisi dell'API Progettazione: database Progettazione: interfaccia utente Realizzazione: database Realizzazione: scrittura codice client Realizzazione: scrittura codice server
3. MILESTONE Analisi Progettazione Realizzazione sito web Collaudo
4. VINCOLI E DIPENDENZE Il progetto farà riferimento a un'API esterna (Google Books), per cui i dati dei libri che si andranno a trattare sono vincolati da essa.
5. TEMPISTICA PRELIMINARE Il progetto inizia il 14 aprile 2021 e deve concludersi entro 24 giorni lavorativi. Qui rappresentato il diagramma di GANTT:



6. PRINCIPALI RISORSE E LIMITI DI COSTO

Costi riguardanti il personale e le postazioni di lavoro.

7. DOCUMENTI DI RIFERIMENTO E ALLEGATI

[Documentazione Google Books API](#)

8. STRUTTURA ORGANIZZATIVA

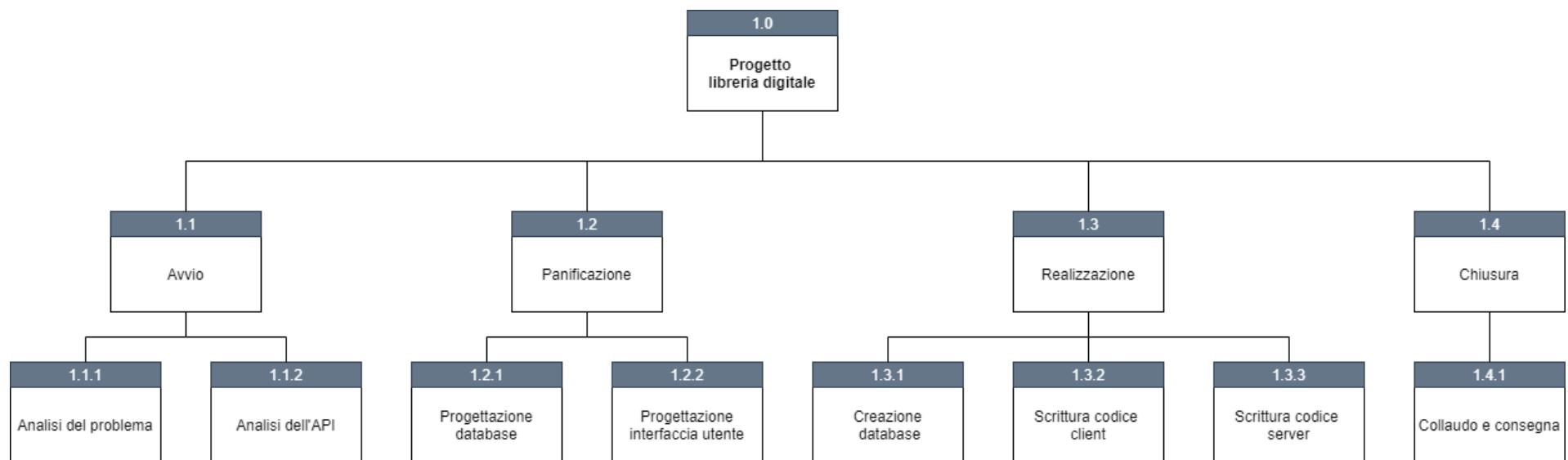
Il Project Manager del progetto sarà lo studente Gabriele Novello.

9. AUTORIZZAZIONE

Approvato da:		Data
---------------	--	------

13.2. Work Breakdown Structure

La WBS (Work Breakdown Structure) è stata utilizzata per rappresentare in modo gerarchico tutte le attività facenti parte al progetto. Vale la regola del 100% secondo la quale l'insieme delle attività figlie compone il 100% dell'attività madre, per cui tutte le attività inserite nella WBS sono tutte le attività da svolgere per completare a pieno il progetto.



Progetto: libreria digitale	Codice progetto: LD21
Data: 15 Aprile 2021	Revisione: 0
Cliente: nessuno	Sponsor: nessuno

WBS	Descrizione	Durata (gg)
1.1 AVVIO		
1.1.1	Analisi del problema	2
1.1.2	Analisi dell'API	1
1.2 PIANIFICAZIONE		
1.2.1	Progettazione database	2
1.2.2	Progettazione interfaccia utente	2
1.3 REALIZZAZIONE		
1.3.1	Creazione database	1
1.3.2	Scrittura codice client	15
1.3.3	Scrittura codice server	6
1.4 CHIUSURA		
1.4.1	Collaudo e consegna	2

13.3. Matrice delle responsabilità

La matrice delle responsabilità o RACI (Responsible Accountable Consulted Informed) viene utilizzata per rappresentare il ruolo di ogni membro del team di progetto (in questo caso una sola persona) in ogni singola attività.

I ruoli sono 4:

- R (Responsible) - rappresenta il responsabile di tipo operativo;
- A (Accountable) - rappresenta il responsabile dell'attività;
- C (Consulted) - è la persona che viene consultata prima di eventuali decisioni sull'attività;
- I (Informed) - è la persona che viene informata di eventuali cambiamenti dell'attività;

Progetto: libreria digitale	Codice progetto: LD21
Data: 15 Aprile 2021	Revisione: 0
Cliente: nessuno	Sponsor: nessuno

ATTIVITÀ	TEAM DI PROGETTO
	Gabriele Novello (PM)
Analisi del problema	A/R
Analisi dell'API	A/R
Progettazione database	A/R
Progettazione interfaccia utente	A/R
Creazione database	A/R
Scrittura codice client	A/R
Scrittura codice server	A/R
Collaudo e consegna	A/R

13.4. Stima dei costi

La stima dei costi è rappresentata dall'RBS (Resource Breakdown Structure) che racchiude e somma i costi delle risorse impiegate. In questo caso i programmi e i servizi utilizzati erano disponibili gratuitamente. Quindi, si è tenuto conto solo delle ore-uomo.

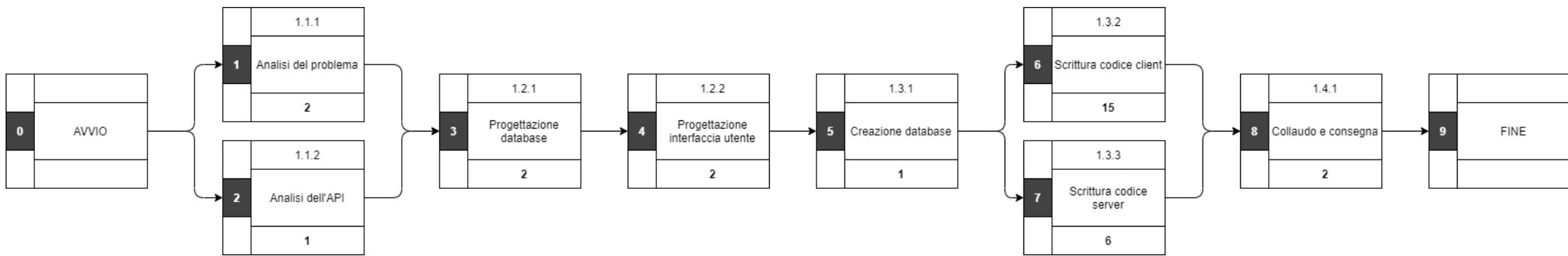
Progetto: libreria digitale	Codice progetto: LD21
Data: 15 Aprile 2021	Revisione: 0
Cliente: nessuno	Sponsor: nessuno

RBS e costi			
1.1.1 Analisi del problema		Tot.	0
Risorse umane	q = 16h		0
1.1.2 Analisi dell'API		Tot.	0
Risorse umane	q = 8h		0
1.2.1 Progettazione database		Tot.	0
Risorse umane	q = 16h		0
1.2.2 Progettazione interfaccia utente		Tot.	0
Risorse umane	q = 16h		0
1.3.1 Creazione database		Tot.	0
Risorse umane	q = 8h		0
1.3.2 Scrittura codice client		Tot.	0
Risorse umane	q = 120h		0
1.3.3 Scrittura codice server		Tot.	0
Risorse umane	q = 48h		0
1.4.1 Collaudo e consegna		Tot.	0
Risorse umane	q = 16h		0
Tot.			0

13.5. Calcolo dei tempi

N.	Attività	Durata	Predecessori
1	Analisi del problema	2	-
2	Analisi dell'API	1	-
3	Progettazione database	2	1,2
4	Progettazione interfaccia utente	2	3
5	Creazione database	1	4
6	Scrittura codice client	15	5
7	Scrittura codice server	6	5
8	Collaudo e consegna	2	6,7

Il diagramma delle dipendenze o PDM rappresenta graficamente i successori e predecessori di ogni attività nel tempo.



Grazie a questo diagramma è possibile individuare il cammino critico del progetto composto da tutte quelle attività il quale ritardo rappresenterebbe un ritardo dell'intero progetto. Le attività critiche avranno scorrimento (il valore che indica di quanti giorni può essere posticipata l'attività) uguale a 0.

E' possibile anche calcolare matematicamente lo scorrimento di un'attività sottraendo la data finale al più tardi (calcolate a partire dalla data finale del progetto) alla data finale al più presto (calcolate a partire dalla data iniziale del progetto).

N.	Attività	Scorrimento
1	Analisi del problema	0
2	Analisi dell'API	1
3	Progettazione database	0
4	Progettazione interfaccia utente	0
5	Creazione database	0
6	Scrittura codice client	0
7	Scrittura codice server	9
8	Collaudo e consegna	0

14. Conclusioni personali

Lo sviluppo di questo progetto è stato utile per mettere alla prova tutte le conoscenze apprese durante gli anni scolastici e produrre qualcosa di effettivamente utile partendo da zero. E' stato inoltre una prova per imparare a gestire un progetto e a rispettare i tempi di consegna prestabiliti simulando quello che sarà l'ambiente aziendale.

Le principali difficoltà si sono trovate nelle prime fasi, durante le quali si è capita l'importanza di analizzare bene lo scenario che ci si è proposti per avere una scaletta da seguire, e durante lo sviluppo del sito web, in cui si è dovuto normalizzare i dati ricevuti dall'API che spesso andavano a creare problemi.

In conclusione, posso affermare che svolgere questo progetto, nonostante le difficoltà, è stato utile alla mia crescita personale ed è stato, in un certo senso, divertente da realizzare.

15. Sitografia e bibliografia

Documentazione Google Books API: <https://cutt.ly/ebl4MK8>

Stack Overflow: <https://stackoverflow.com/>

W3School: <https://www.w3schools.com/>

Estratto libro: ["Internet Working - Sistemi e Reti - Seconda Edizione"](#)

16. Glossario

API REST, nota anche come API RESTful, è un'interfaccia di programmazione delle applicazioni (API) conforme ai vincoli dello stile di architettura REST.

file strutturati, file che rispettano una struttura ben definita per regolamentare lo scambio di dati tra sistemi informatici; i 3 formati più utilizzati sono CSV, JSON e XML.

API, acronimo di Application Programming Interface, sono set di definizioni e protocolli con i quali vengono realizzati e integrati software applicativi.

JSON, acronimo di JavaScript Object Notation, è un formato adatto all'interscambio di dati fra applicazioni client/server basato sul linguaggio JavaScript.

HTML, acronimo di HyperText Markup Language, è un linguaggio di markup nato per la formattazione e impaginazione di documenti ipertestuali web.

CSS, acronimo di Cascading Style Sheets, è un linguaggio usato per definire la formattazione e lo stile di documenti HTML.

JavaScript, è un linguaggio di programmazione orientato agli oggetti e agli eventi comunemente utilizzato nella programmazione web lato client.

Client, indica una determinata componente hardware o software che accede alle risorse o ai servizi erogati da un'altra componente, detta server.

PHP, Hypertext Preprocessor, è un linguaggio di scripting interpretato principalmente utilizzato per sviluppare applicazioni web lato server.

Server, computer di elevate prestazioni che in una rete fornisce un servizio agli altri elaboratori collegate, detti client.

AJAX, acronimo di Asynchronous JavaScript and XML, è una tecnica di sviluppo software per la realizzazione di applicazioni web interattive, basandosi su uno scambio di dati in background fra browser e server, consentendo così l'aggiornamento dinamico di una pagina web senza esplicito ricaricamento da parte dell'utente.

Database, archivio di dati strutturato in modo da razionalizzare la gestione e l'aggiornamento delle informazioni e da permettere lo svolgimento di ricerche complesse.

Modello relazionale, modello logico di rappresentazione e strutturazione dei dati come relazioni, viene implementato sul DBMS.

RDBMS, acronimo di Relational DataBase Management System, indica un sistema di gestione di database basato sul modello relazionale.

MySQL, è un sistema open source di gestione di database relazionali SQL sviluppato e supportato da Oracle.

Funzione di hashing, è una funzione non invertibile che mappa una stringa di lunghezza arbitraria in una stringa di lunghezza predefinita.

HTTPS, acronimo di HyperText Transfer Protocol Secure, è un protocollo per la comunicazione su Internet che protegge l'integrità e la riservatezza dei dati scambiati tra i computer e i siti.

GET, è il metodo dei protocolli HTTP e HTTPS con cui vengono richieste la maggior parte delle informazioni ad un web server, tali richieste vengono veicolate tramite query string, cioè la parte di un URL che contiene dei parametri da passare in input ad un'applicazione.

POST, è il metodo dei protocolli HTTP e HTTPS che consente di inviare dati ad un server senza mostrarli in query string all'interno dell'URL.

Stati di HTTP, gli stati definiti ritornati dal protocollo HTTP sono divisi in 4 categorie: messaggio 2XX (indica il successo di un'operazione), messaggio 3XX (indica il reindirizzamento di una pagina), messaggio 4XX (indica errori del client), messaggio 5XX (indica errori del server).

Primary Key, PK o chiave primaria, è composta da uno o più attributi e identifica in modo univoco ogni record di una tabella. Ogni attributo che può essere scelto come Primary Key è definito "chiave candidata".

Foreign Key, FK o chiave esterna è un vincolo che garantisce l'integrità dei dati e che rappresenta quella che nello schema concettuale è chiamata associazione. Una chiave esterna è caratterizzata da una colonna figlia (di una tabella figlia) che si riferisce a una colonna madre (di una tabella madre). In pratica i valori assunti dalla Foreign Key della tabella figlia saranno limitati dai valori già esistenti nella Primary Key della tabella madre.

supernetting, è un processo inverso di subnet, in cui diverse reti vengono unite in un'unica rete.

Certification Authority, è un soggetto di *trusted third part* abilitato ad emettere un certificato digitale.

Mirror, con mirror, in informatica, si intende una copia esatta di un insieme di dati.