

# BiblioTech

## ***ANALISI E PROGETTAZIONE***

### ***Analisi per frasi omogenee***

- 1) L'obiettivo di BiblioTech è informatizzare la gestione dei prestiti dei libri della biblioteca scolastica, così da sostituire il registro cartaceo.  
Il sistema deve consentire la gestione del catalogo dei libri, degli utenti e del ciclo di vita del prestito, differenziando le operazioni in base al ruolo dell'utente.

**Ipotesi aggiuntiva:** Il sistema è riservato a una sola scuola.

---

- 2) Nella biblioteca sono disponibili per ogni libro diverse copie. Il libro è caratterizzato dal titolo, autore, descrizione, editore e categoria.
- 

- 3) Il sistema deve tener conto della disponibilità di ogni libro per il prestito, aggiornando in automatico la disponibilità al momento della restituzione e prestito.

**Ipotesi aggiuntiva:** Le copie non possono andare in negativo.

---

- 4) Il sistema deve garantire che ogni utente che accede all'applicazione sia verificato tramite il metodo "2FA". Un utente può avere il ruolo di studente o bibliotecario.

**Ipotesi aggiuntiva:** A ogni utente viene rilasciata una chiave fondamentale per l'autenticazione con 2FA.

---

- 5) Gli studenti possono visualizzare catalogo, prendere in prestito libro (copie>0), visualizzare storico.  
Bibliotecari possono avere una visione globale, visualizzare libri in prestito con i dati dello studente, aggiungere copie di un libro, registrare restituzione.
- 

- 6) Il prestito è l'operazione con la quale un libro viene assegnato a un utente per un periodo di tempo. Può essere associato a un solo studente e ha una data di inizio e fine assegnata dal sistema.
- 

- 7) Il sistema deve impedire la gestione del prestito se le copie disponibili sono 0.
- 

- 8) Per accedere all'applicazione bisogna effettuare il login con username e password.
-

9) Il sistema deve tener conto dello storico dei prestiti di un utente.

---

### ***Analisi funzionale generale***

BiblioTech deve gestire i libri in prestito della scuola, monitorando le giacenze del magazzino e differenziando le operazioni in base al ruolo dell'utente (Studenti e Bibliotecari). Ci sono più titoli e la scuola possiede più copie fisiche (es. 10 copie di "1984" di Orwell). Il sistema deve conoscere:

- Il numero di copie possedute in totale (quindi anche quelle in prestito al momento)
- Il numero delle copie disponibili al prestito

L'accesso alla piattaforma è consentito solo tramite login:

- Studenti: Visualizzare catalogo, prendere in prestito libro (copie>0), visualizzare storico.
- Bibliotecari: Visione globale, visualizzare libri in prestito con i dati dello studente, registrare restituzione libro.

Quando si effettua il prestito, viene decrementato il contatore delle copie e si genera un record con data odierna.

Con la restituzione l'operazione deve chiudere il prestito aggiornando la data della restituzione e incrementare il contatore delle copie disponibili.

---

## **Ulteriori specifiche**

Accesso alla piattaforma tramite “2FA”.

Sfruttando “**2FAuth**”, una piattaforma **web-based e self-hosted** concepita per gestire i codici di autenticazione a due fattori in modo indipendente dai dispositivi fisici, avviene l’accesso all’applicazione. Il software si pone come un'alternativa aperta e privata ad app come Google Authenticator, consentendo agli utenti di **generare password temporanee** (OTP) direttamente dal browser. Grazie alla sua natura **multi-dispositivo**, lo strumento facilita l'accesso ai propri account anche in assenza di uno smartphone, garantendo al contempo un controllo totale sui **dati sensibili** tramite crittografia. Le sue funzionalità includono l'organizzazione dei profili in gruppi e la possibilità di gestire più account utente sulla stessa istanza.

L'**API REST di 2FAuth** consente di interfacciare l'applicazione con software esterni, permettendo di automatizzare gran parte delle sue funzioni. Le sue caratteristiche principali includono la **gestione degli account 2FA** (aggiunta, modifica e organizzazione in gruppi) e la **generazione programmatica di codici TOTP/HOTP** aggiornati. Questo strumento è fondamentale per integrare la sicurezza a due fattori in script personalizzati o flussi di lavoro automatizzati, garantendo l'accesso ai dati di sicurezza anche al di fuori dell'interfaccia web standard. Il sistema genera al momento della registrazione una chiave che l’utente dovrà inserire dentro <https://2fauth.app/> (è presente un collegamento diretto cliccando su un link al momento della registrazione) la prima volta che si registra. In seguito, basterà accedere al proprio account generare il codice e inserirlo in Bibliotech. La generazione del codice temporaneo è basata su due fattori: il codice TOTP e l’orario attuale, combinati tramite un algoritmo hash.

Se username e password sono già state verificate si passa alla verifica del codice inserito. Il sistema rigenera localmente il codice TOTP utilizzando la stessa chiave segreta e verifica la corrispondenza. Dopo aver confrontato i due codici se uguali l’accesso è consentito.

---

## **Implementazione di 2FAuth in Docker**

*Per la gestione dell'autenticazione a due fattori (2FA) è stato integrato il servizio 2FAuth all'interno dell'infrastruttura Docker dell'applicazione.*

*L'installazione e la configurazione sono state eseguite seguendo la documentazione ufficiale disponibile sul sito del progetto, adattando i parametri all'ambiente già esistente.*

*L'infrastruttura Docker è composta dai seguenti servizi:*

- *web: applicazione PHP con Apache*
- *db: database MySQL*
- *phpmyadmin: interfaccia web per la gestione del database*
- *2fauth: applicazione per la gestione dei codici di autenticazione a due fattori*

*Tutti i container comunicano tra loro tramite una rete Docker dedicata (app-network).*

---

## **Configurazione del servizio 2FAuth**

*Il servizio è stato aggiunto nel file docker-compose.yml utilizzando l'immagine ufficiale:*

- *image: 2fauth/2fauth*  
*Scarica l'immagine ufficiale di 2FAuth da Docker Hub.*
- *container\_name: 2fauth-app*  
*Nome assegnato al container per facilitarne la gestione.*
- *restart: always*  
*Il container viene riavviato automaticamente in caso di errore o riavvio del sistema, garantendo la disponibilità continua del servizio.*
- *ports: 8082:8000*  
*Espone l'applicazione sulla porta 8082 del computer host.*  
*Il servizio è accessibile dal browser all'indirizzo:*
- *http://localhost:8082*

---

## **Variabili d'ambiente**

### *Configurazione generale*

- **APP\_NAME=2FAuth-Biblioteca**  
*Nome dell'applicazione visualizzato nell'interfaccia.*
- **APP\_URL=<http://localhost:8082>**  
*Indica l'indirizzo pubblico dell'applicazione. È necessario per generare correttamente i collegamenti interni e caricare le risorse (CSS e JavaScript).*
- **APP\_TIMEZONE=Europe/Rome**  
*Imposta il fuso orario, importante per la corretta sincronizzazione dei codici temporanei (TOTP).*

---

## **Connessione al database**

2FAuth utilizza MySQL già presente nell'infrastruttura.

- **DB\_CONNECTION=mysql**  
*Tipo di database utilizzato.*
- **DB\_HOST=db**  
*Nome del servizio MySQL all'interno della rete Docker.*
- **DB\_PORT=3306**  
*Porta del database.*
- **DB\_DATABASE=myapp\_db**  
*Database utilizzato (può essere condiviso con l'applicazione principale o dedicato).*
- **DB\_USERNAME=myuser**
- **DB\_PASSWORD=mypassword**  
*Credenziali di accesso al database.*

---

## **Chiave di sicurezza**

- `APP_KEY=base64:...`

*Questa chiave è obbligatoria e viene utilizzata per la crittografia dei dati sensibili, in particolare i segreti utilizzati per generare i codici 2FA.*

*La chiave è stata generata utilizzando Laravel.*

---

### **Persistenza dei dati**

*Per evitare la perdita dei dati in caso di ricreazione del container, è stato configurato un volume Docker:*

- `2fauth_data:/srv/2FAuth/storage`

*Questo volume salva in modo permanente:*

- configurazioni dell'applicazione
- account configurati
- segreti TOTP

*Il volume è definito nella sezione volumes del file docker-compose.yml.*

---

### **Dipendenze e rete**

- `depends_on: db`  
*Garantisce che il database venga avviato prima del servizio 2FAuth.*
  - `networks: app-network`  
*Permette la comunicazione tra i container tramite il nome del servizio (ad esempio db).*
- 

### **Avvio del servizio**

*Dopo aver configurato il file docker-compose.yml, i container sono stati avviati con il comando:*

*`docker-compose up -d`*

*Una volta avviato, il servizio è accessibile da browser all'indirizzo:*

*`http://localhost:8082`*

*Al primo accesso è possibile completare la configurazione iniziale e iniziare ad aggiungere gli account per la gestione dell'autenticazione a due fattori.*

---

### **Note finali**

*L'integrazione di 2FAuth è stata realizzata seguendo le istruzioni ufficiali del progetto e adattandole all'ambiente Docker esistente.*

*Il servizio è stato collegato al database già presente e configurato con un volume dedicato per garantire la persistenza dei dati e la sicurezza delle informazioni.*

## **Schema E-R**

### **UTENTE**

Rappresenta chi accede al sistema (Studenti e Bibliotecari).

- **id\_utente** (PK)
- nome
- cognome
- email (UNIQUE)
- password\_hash
- ruolo (*Studente* | *Bibliotecario*)
- 2FA\_attivo (boolean)
- 2FA\_secret (chiave per OTP)
- data\_registrazione

### **Note**

- **2FA\_secret** contiene la chiave usata dall'app 2Fauth
- Il campo **ruolo** determina le operazioni disponibili



## LIBRO

Rappresenta il titolo posseduto dalla scuola (non la singola copia fisica).

- **id\_libro** (PK)
- titolo
- autore
- editore
- anno\_pubblicazione
- categoria
- descrizione
- copie\_totali
- copie\_disponibili

## Vincoli

- $\text{copie\_disponibili} \leq \text{copie\_totali}$
- Un prestito è possibile solo se:  $\text{copie\_disponibili} > 0$

## PRESTITO

Associazione tra UTENTE e LIBRO con attributi propri.

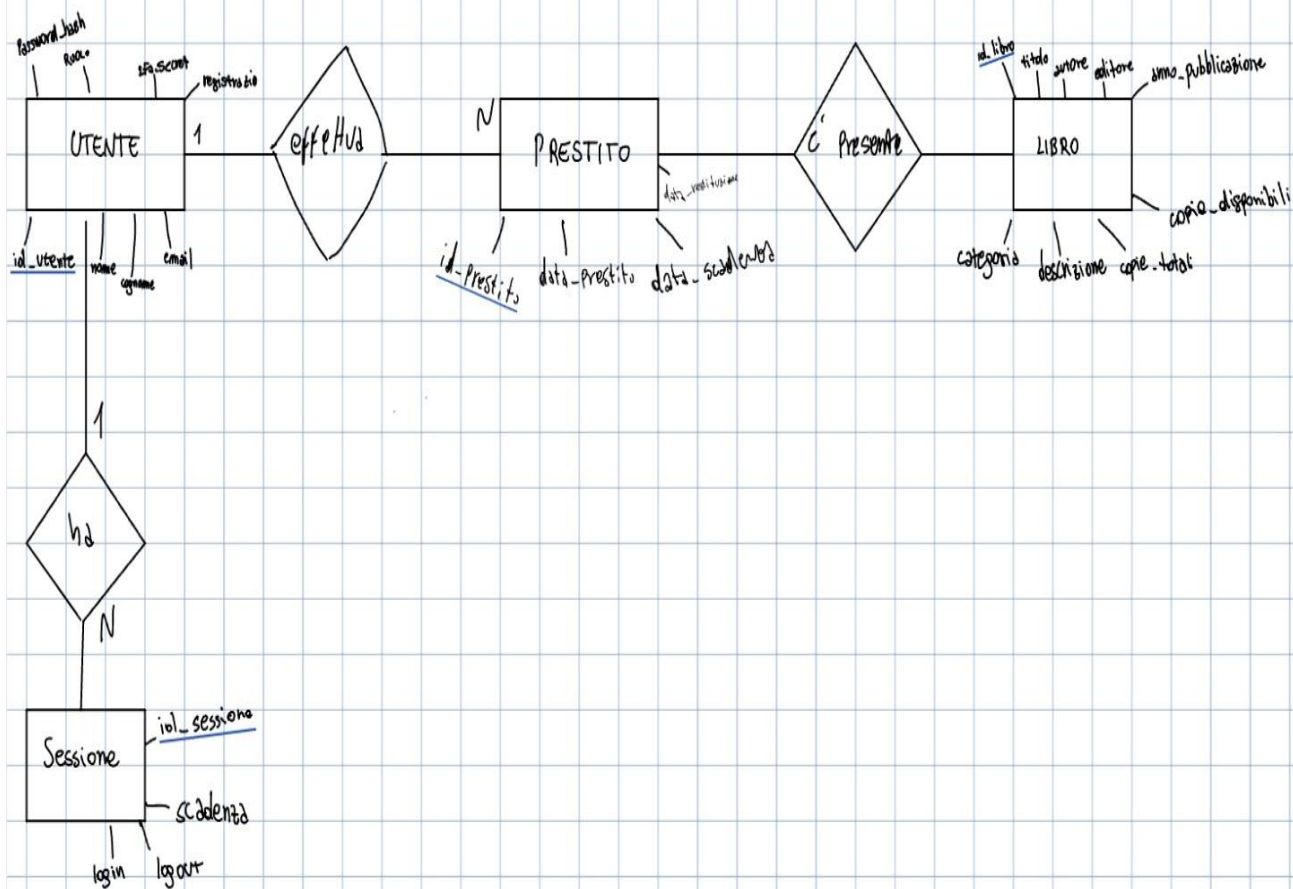
- **id\_prestito** (PK)
- id\_utente (FK → UTENTE.id\_utente)
- id\_libro (FK → LIBRO.id\_libro)
- data\_prestito
- data\_scadenza (opzionale)
- data\_restituzione (NULL se il libro non è stato restituito)
- stato (*Attivo / Restituito*)

## ***SESSIONE***

Dopo che l'utente ha effettuato l'accesso, viene stabilita una sessione.

- **Id\_sessione (PK)**
- Id\_utente(FK → UTENTE.id\_utente)
- Login
- Logout
- Scadenza

## ***Diagramma UML***



## ***Specifiche di sessione e sicurezza***

### **Dati salvati in \$\_SESSION dopo il login**

Dopo autenticazione (email + password + verifica 2FA), il sistema crea una sessione e memorizza le informazioni essenziali dell'utente.

### **Tabella dati di sessione**

<b>Chiave SESSION</b>	<b>Tipo</b>	<b>Descrizione</b>
session_id	int	Identificativo univoco della sessione
Id_utente	int	Id dell'utente
scadenza	Time stamp	Scadenza sessione
login	Time stamp	Orario di login
logout	Time stamp	Orario di logout

## **2. Autenticazione utente**

### **Password**

Le password non vengono salvate in chiaro ma come hash.

## **3. Autenticazione a due fattori (2FA)**

Il sistema garantisce sicurezza perché:

- Il codice TOTP cambia ogni 30 secondi
- La chiave segreta è unica per ogni utente
- Senza accesso all'app 2FAuth non è possibile completare il login

#### **4. Controllo accesso alle pagine protette**

Ogni pagina dell'applicazione include un controllo di sessione.

Questo impedisce l'accesso a:

- utenti non autenticati
- utenti che non hanno completato il 2FA

#### **6. Come il sistema impedisce accessi non autorizzati**

Il sistema utilizza tre livelli di controllo:

##### **1) Verifica sessione**

Se `logged_in` non è presente → redirect al login.

##### **2) Verifica 2FA**

Se il codice 2FA è errato o nullo → accesso negato.

##### **3) Verifica ruolo**

Se:

`ruolo` ≠ BIBLIOTECARIO

l'utente non può accedere alle pagine amministrative.

Anche se uno studente prova ad accedere direttamente tramite URL, verrà reindirizzato alla pagina di accesso negato.

#### ***Chiusura sessione***

Alla richiesta di logout la sessione viene eliminata tramite la funzione `setcookie('PHPSESSID', '', time() - 3600, '/')`. Così da prevenire accessi non autorizzati.

