ANALISI TECNICA COMPLETA

SISTEMA BIBLIOTECH

Documentazione di Architettura, Sicurezza e Sviluppo

Versione: 1.0.0 (Release Definitiva)

Stack: PHP 8.2 / MySQL 8.0 / Docker

Repository: Gigichx/Bibliotech-Lattanzio

Data: Febbraio 2024

Indice dei Contenuti

1. Introduzione e Contesto del Progetto

L'istituto scolastico necessita di una migrazione dal vecchio registro cartaceo dei prestiti bibliotecari a una soluzione digitale centralizzata denominata "BiblioTech". Il sistema è progettato per garantire l'integrità dei dati, eliminare errori di trascrizione e fornire un accesso differenziato tra studenti e personale bibliotecario.

L'obiettivo primario è automatizzare il ciclo di vita del prestito (uscita e rientro), monitorando in tempo reale le giacenze di magazzino. Un punto cardine del progetto è l'eliminazione delle password tradizionali a favore di un sistema di autenticazione "Passwordless" (Magic Link) per massimizzare la sicurezza e semplificare l'esperienza utente.

2. Stack Tecnologico Completo

Componente	Tecnologia / Versione
Backend	PHP 8.2 (Procedurale con PDO)
Database	MySQL 8.0
Web Server	Apache (Image: php:8.2-apache)
Frontend	HTML5, Bootstrap 5.3.0, CSS Custom
Email Service	PHPMailer 6.9 + Mailtrap SMTP
Environment	Docker + Docker Compose
Porta Applicazione	8085 (http://localhost:8085)

Glossario Tecnologico e Motivazioni

> ## PDO (PHP Data Objects)
>
> **Cos'è:** PDO è un'interfaccia PHP per accedere ai database in modo uniforme. Fornisce un livello di astrazione che permette di utilizzare lo stesso codice per diversi DBMS (MySQL, PostgreSQL, SQLite, etc.).
>
> **Perché lo usiamo:**
>
> - **Prepared Statements nativi:** Protezione automatica contro SQL Injection
> - **Portabilità:** Possiamo cambiare database con modifiche minime
> - **Performance:** Le query preparate possono essere riutilizzate con parametri diversi (cache del piano di esecuzione)
> - **Gestione errori:** Supporto nativo per eccezioni (PDO::ERRMODE_EXCEPTION)

- **Transazioni:** Supporto completo per BEGIN, COMMIT, ROLLBACK

Alternativa scartata: `mysqli` (limitato a MySQL, API meno elegante)

ENUM (Enumerated Type)

Cos'è: ENUM è un tipo di dato MySQL che limita i valori di una colonna a un insieme predefinito di stringhe. Nel nostro caso: `ruolo ENUM('studente', 'bibliotecario')`

Perché lo usiamo:

- **Validazione a livello database:** Impossibile inserire valori non previsti
- **Ottimizzazione storage:** MySQL memorizza ENUM come integer (1 byte) internamente
- **Documentazione autoesplicativa:** Lo schema DB mostra i valori possibili
- **Integrità referenziale:** Non serve una tabella separata per 2 soli valori

Alternativa scartata: Tabella `ruoli` separata (overhead eccessivo per 2 valori statici)

CSPRNG (Cryptographically Secure Pseudo-Random Number Generator)

Cos'è: Un generatore di numeri casuali crittograficamente sicuro. In PHP: `random_bytes()`

Perché lo usiamo:

- **Sicurezza:** Genera token imprevedibili anche per attaccanti con conoscenza dell'algoritmo
- **Entropia:** Usa fonti di casualità del sistema operativo (/dev/urandom su Linux)
- **Standard:** Conforme a requisiti crittografici moderni

Alternativa scartata: `rand()` o `mt_rand()` (predicibili, NON sicuri per crittografia)

FOR UPDATE (Lock Pessimistico)

Cos'è: Clausola SQL che blocca le righe selezionate fino alla fine della transazione, impedendo ad altre sessioni di leggerle o modificarle.

Perché lo usiamo:

- **Race Condition Protection:** Se due utenti tentano di prendere l'ultima copia disponibile, solo uno ci riuscirà
- **Atomicità:** Lettura e scrittura diventano un'operazione indivisibile

- **Consistenza:** Il contatore `copie_disponibili` rimane sempre accurato

Esempio scenario critico:

```
Tempo | Utente A                | Utente B                | DB copie_disp
------|-------------------------|-------------------------|--------------
T1    | SELECT (copie=1)        |                         | 1
T2    |                         | SELECT (copie=1)        | 1
T3    | UPDATE (copie=0)        |                         | 0
T4    |                         | UPDATE (copie=-1) ❌    | -1 (ERRORE!)

CON FOR UPDATE:
T1    | SELECT FOR UPDATE       |                         | 1 (LOCKED)
T2    |                         | SELECT FOR UPDATE       | ⏳ ATTENDE
T3    | UPDATE (copie=0)        |                         | 0 (LOCKED)
T4    | COMMIT                  |                         | 0 (UNLOCKED)
T5    |                         | Legge copie=0 ✓          | 0
T6    |                         | ROLLBACK (non disp.)    | 0
```

Alternativa scartata: Lock ottimistico con versioning (più complesso, richiede colonna `version`)

Bootstrap 5.3.0

Cos'è: Framework CSS/JS per interfacce responsive. Fornisce componenti pre-costruiti (navbar, card, form, etc.)

Perché lo usiamo:

- **Velocità sviluppo:** Componenti pronti all'uso
- **Responsive design:** Mobile-first, grid system flessibile
- **Cross-browser:** Testato su tutti i browser moderni
- **Accessibilità:** Supporto ARIA attributes integrato
- **Personalizzabile:** Variabili CSS custom per theming

Alternativa scartata: Tailwind CSS (sintassi verbose, curva apprendimento)

PHPMailer 6.9

Cos'è: Libreria PHP per l'invio di email con supporto SMTP, HTML, allegati, autenticazione.

Perché lo usiamo:

- **SMTP sicuro:** Supporto TLS/SSL, autenticazione moderna
- **Email HTML:** Template elaborati con inline CSS
- **Gestione errori:** Exceptions dettagliate per debugging
- **Standard RFC:** Conforme a RFC 2822, RFC 5321

Alternativa scartata: `mail()` nativa PHP (limitata, spesso bloccata da firewall)

Docker + Docker Compose

Cos'è: Docker containerizza applicazioni, Docker Compose orchestra più container (web + db).

Perché lo usiamo:

- **Riproducibilità:** "Funziona sulla mia macchina" → funziona ovunque
- **Isolamento:** Nessuna interferenza con software locali
- **Setup rapido:** Un solo comando (`docker-compose up`)
- **Versioning:** PHP 8.2, MySQL 8.0 garantiti, non dipende dal sistema host
- **Sviluppo = Produzione:** Stesso ambiente in dev e prod

Alternativa scartata: XAMPP/WAMP (versioni fisse, configurazione manuale, non portabile)

Mailtrap

Cos'è: Servizio di testing email SMTP. Cattura email senza inviarle realmente ai destinatari.

Perché lo usiamo:

- **Testing sicuro:** Nessun rischio di inviare email a utenti reali per errore
- **Debug:** Anteprima HTML, analisi header, controllo spam score
- **Gratuito:** Piano free sufficiente per sviluppo
- **SMTP reale:** Comportamento identico a server di produzione

Transizione produzione: Sostituire con SendGrid, AWS SES, Mailgun

UTF-8 / utf8mb4

Cos'è: Charset per encoding Unicode. `utf8mb4` in MySQL supporta tutti i caratteri Unicode (emoji incluse).

Perché lo usiamo:

- **Internazionalizzazione:** Supporto caratteri accentati, cirillico, kanji, emoji
- **Collation:** `utf8mb4_unicode_ci` ordina correttamente lettere accentate
- **Standard moderno:** Raccomandato da W3C e WHATWG

Alternativa scartata: `utf8` MySQL (limitato a 3 byte, no emoji)

Prepared Statements

Cos'è: Query SQL con placeholder (`?`) compilate separatamente dai dati. I parametri vengono inviati separatamente e non possono alterare la struttura della query.

Perché lo usiamo:

- **SQL Injection Prevention:** Impossibile iniettare codice SQL
- **Performance:** Query compilate una volta, eseguite più volte
- **Leggibilità:** Separazione logica tra query e dati

Esempio:

```
// ❌ VULNERABILE
$sql = "SELECT * FROM utenti WHERE email = '$email'";
// Attacco: $email = "' OR '1'='1"

// ✅ SICURO
$stmt = $pdo->prepare("SELECT * FROM utenti WHERE email = ?");
$stmt->execute([$email]);
// Qualsiasi valore in $email viene trattato come stringa
```

Session Regenerate ID

Cos'è: Genera un nuovo ID di sessione, invalidando quello precedente. Chiamato dopo operazioni critiche come il login.

Perché lo usiamo:

- **Session Fixation Protection:** Impedisce attacchi dove l'attaccante forza un ID noto
- **Best Practice:** Raccomandato da OWASP

Scenario attacco prevenuto:

```
1. Attaccante ottiene PHPSESSID=abc123 (session fissata)
2. Attaccante invia link alla vittima: login.php?PHPSESSID=abc123
3. Vittima fa login con successo
4. session_regenerate_id() → nuovo ID = xyz789
5. Attaccante con abc123 NON ha accesso ✓
```

3. Architettura del Sistema

Il sistema segue un'architettura **Monolitica Modulare** basata su PHP puro. Non vengono utilizzati framework MVC pesanti per mantenere il codice leggero e didatticamente ispezionabile, ma si applicano principi di separazione delle responsabilità tramite l'inclusione di file e configurazioni centralizzate.

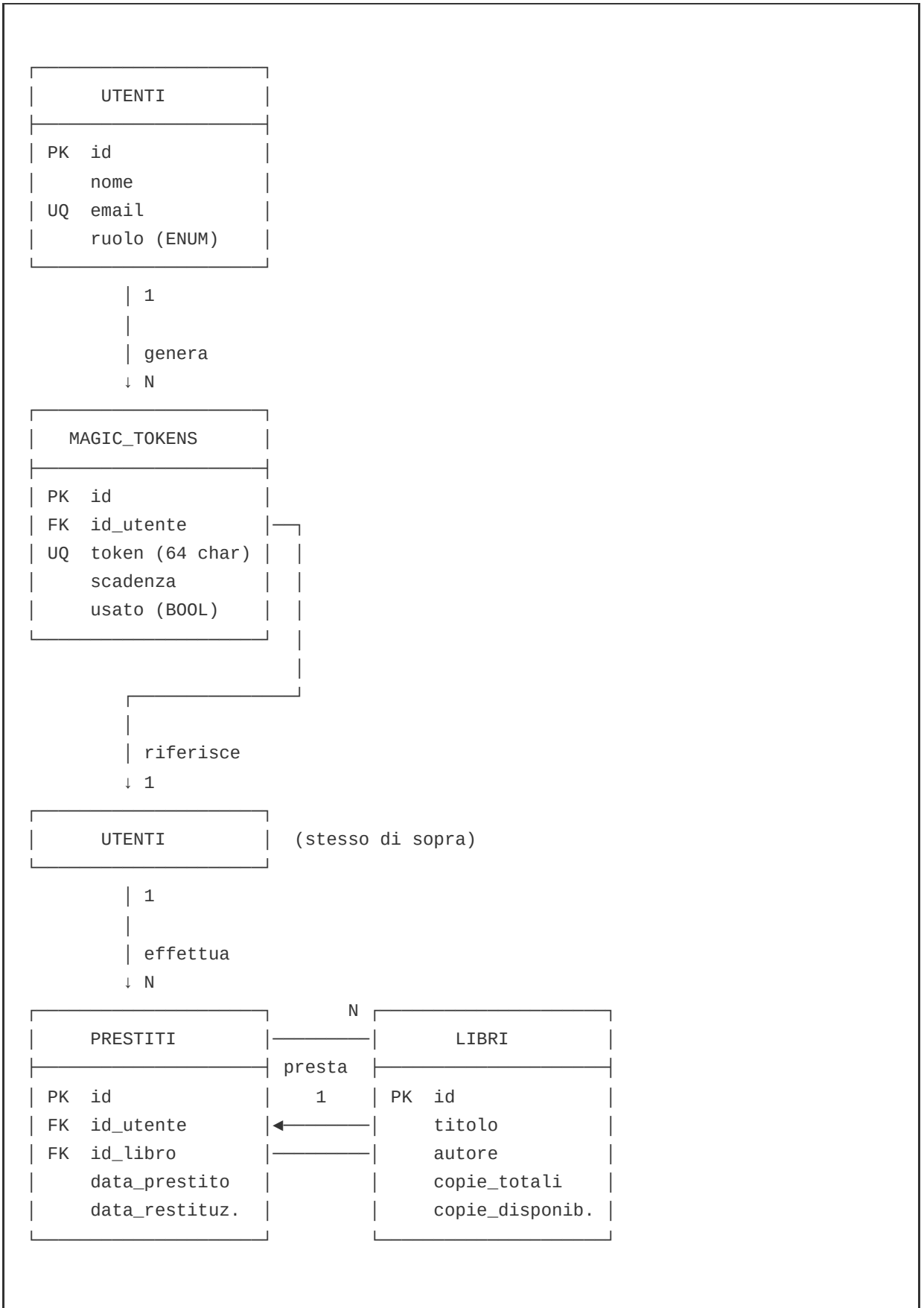
Struttura dei File (Reale)

```
Bibliotech-Lattanzio/
├── .env                        # Credenziali Mailtrap e DB
├── .gitattributes
├── docker-compose.yaml         # Orchestrazione container
├── Dockerfile                  # Build immagine PHP custom
├── README.md
├── sql/
│   └── database.sql            # Schema DB e Seed dati
└── src/
    ├── .DS_Store
    ├── composer.json           # Dipendenze (PHPMailer 6.9)
    ├── index.php               # Router/Redirect intelligente
    ├── login.php               # Form email + invio magic link
    ├── verify.php              # Validazione token + creazione sessione
    ├── logout.php              # Distruzione sessione
    ├── libri.php               # Catalogo con ricerca e filtri
    ├── libro.php               # Dettaglio + azione prestito
    ├── prestiti.php            # Dashboard studente
    ├── gestione_restituzioni.php  # Dashboard bibliotecario
    ├── config/
    │   └── db.php              # PDO + helper functions
    ├── includes/
    │   ├── auth.php            # Auth logic & helper functions
    │   └── navbar.php          # Componente UI condiviso
    └── assets/
        ├── CSS/
        │   └── style.css       # Stili custom
        └── IMG/
            └── logo.png        # Logo BiblioTech
```

4. Schema Database Dettagliato

Il database è normalizzato per garantire l'integrità referenziale. Una caratteristica distintiva è l'assenza del campo `password` nella tabella utenti, sostituito dalla tabella `magic_tokens` per la gestione degli accessi temporanei.

Diagramma ER (Entità-Relazione)



```
Legenda:
  PK = Primary Key (Chiave Primaria)
  FK = Foreign Key (Chiave Esterna)
  UQ = Unique (Vincolo di Unicità)
  1:N = Cardinalità uno-a-molti
  N:1 = Cardinalità molti-a-uno
```

Tabella delle Relazioni

Relazione	Cardinalità	Descrizione
UTENTI → MAGIC_TOKENS	1:N	Un utente può generare più token nel tempo (nuovi login)
UTENTI → PRESTITI	1:N	Un utente può effettuare più prestiti (anche simultanei)
LIBRI → PRESTITI	1:N	Un libro può essere prestato più volte (in momenti diversi o copie diverse)

Vincoli di Integrità

- **CHECK:** `copie_disponibili >= 0` (previene valori negativi)
- **CHECK:** `copie_disponibili <= copie_totali` (coerenza logica)
- **CASCADE DELETE:** Se utente eliminato → magic_tokens e prestiti eliminati automaticamente
- **UNIQUE:** Un token non può essere duplicato (sicurezza crittografica)
- **UNIQUE:** Una email non può essere duplicata (unicità utente)

Specifiche Tabelle (Dati Reali)

Tabella: `utenti`

Anagrafica utenti. Il campo `ruolo` è un ENUM ('studente', 'bibliotecario').

- **Seed:** Mario Rossi, Laura Bianchi, Giuseppe Verdi (Studenti)
- **Seed:** Anna Biblioteca (Bibliotecario)
- **Email domain:** @example.com

Tabella: `magic_tokens`

Gestisce i token di accesso usa-e-getta.

- **token:** Hash casuale (64 char).
- **scadenza:** DATETIME.

- **usato:** BOOLEAN (DEFAULT 0).
- **Indici:** idx_token, idx_scadenza, idx_usato.

Tabella: `libri`

Gestisce il catalogo. `copie_disponibili` è un contatore denormalizzato aggiornato transazionalmente.

- **Check Constraints:** `copie_disponibili >= 0`, `copie_disponibili <= copie_totali`.
- **Indici:** idx_titolo, idx_autore.

Tabella: `prestiti`

Storico operazioni. Un prestito è "attivo" se `data_restituzione` è NULL.

- **Indici:** idx_utente, idx_libro, idx_data_restituzione.
- **Foreign Keys:** ON DELETE CASCADE su utente e libro.

5. Sistema Magic Link Passwordless

Flusso Tecnico di Autenticazione

1. **Input:** L'utente inserisce la propria email in `login.php`.
2. **Verifica:** `db_fetch_one` controlla se l'email esiste in `utenti`.
3. **Generazione:**
   ```
   $token = bin2hex(random_bytes(32)); // 64 caratteri
   ```
4. **Salvataggio:** Token salvato in `magic_tokens` con scadenza +15 minuti (900s).
5. **Invio:** PHPMailer invia un'email HTML formattata tramite Mailtrap SMTP.
6. **Link:** `http://localhost:8085/verify.php?token=xyz...`
7. **Verifica Token (verify.php):**
   - Token esiste?
   - Token non usato? (`usato == 0`)
   - Token non scaduto? (`strtotime($scadenza) > time()`)
8. **Login:** Se valido, `usato=1`, rigenerazione sessione, redirect.

Gestione Errori

- **Token mancante:** "Token mancante. Richiedi un nuovo link."
- **Token non valido:** "Token non valido."
- **Token scaduto:** "Questo link è scaduto."
- **Token usato:** "Questo link è già stato utilizzato."

6. Gestione Sessioni, Cookie e Redirect

Variabili di Sessione Reali

- `$_SESSION['user_id']`: ID utente.
- `$_SESSION['ruolo']`: 'studente' o 'bibliotecario'.
- `$_SESSION['nome']`: Nome completo visualizzato.
- `$_SESSION['login_time']`: Timestamp del login.
- `$_SESSION['redirect_after_login']`: URL richiesto prima del login.

Logica di Redirect Intelligente

Il sistema ricorda dove voleva andare l'utente prima di essere reindirizzato al login:

```
// auth.php
function requireAuth() {
    if (!isAuthenticated()) {
        $_SESSION['redirect_after_login'] = $_SERVER['REQUEST_URI'];
        header('Location: /login.php');
        exit;
    }
}
```

7. Funzioni Helper e Utility

Helper Autenticazione (`includes/auth.php`)

Funzione	Descrizione
`isAuthenticated()`	Verifica presenza variabili sessione
`requireAuth()`	Redirect a login se non autenticato
`requireRole($roles)`	Verifica ruolo (array o stringa), logga tentativi non autorizzati
`getCurrentUserId()`	Restituisce ID utente o null
`isBibliotecario()`	True se ruolo === 'bibliotecario'
`isStudente()`	True se ruolo === 'studente'
`createUserSession($user)`	Rigenera ID, setta variabili sessione
`getRedirectAfterLogin()`	Gestisce il redirect post-login

Helper Database (`config/db.php`)

Funzione	Descrizione
`db_query($sql, $params)`	Esegue query preparata, restituisce stmt
`db_fetch_one($sql, $params)`	Restituisce singola riga (assoc)
`db_fetch_all($sql, $params)`	Restituisce tutte le righe (array)

8. Flussi Operativi e Transazioni

Prestito Libro (con Lock Pessimistico)

```
// libro.php
$pdo->beginTransaction();

// 1. SELECT FOR UPDATE (Blocca la riga)
$libro = db_fetch_one(
    'SELECT * FROM libri WHERE id = ? AND copie_disponibili > 0 FOR UPDATE',
    [$libro_id]
);
```

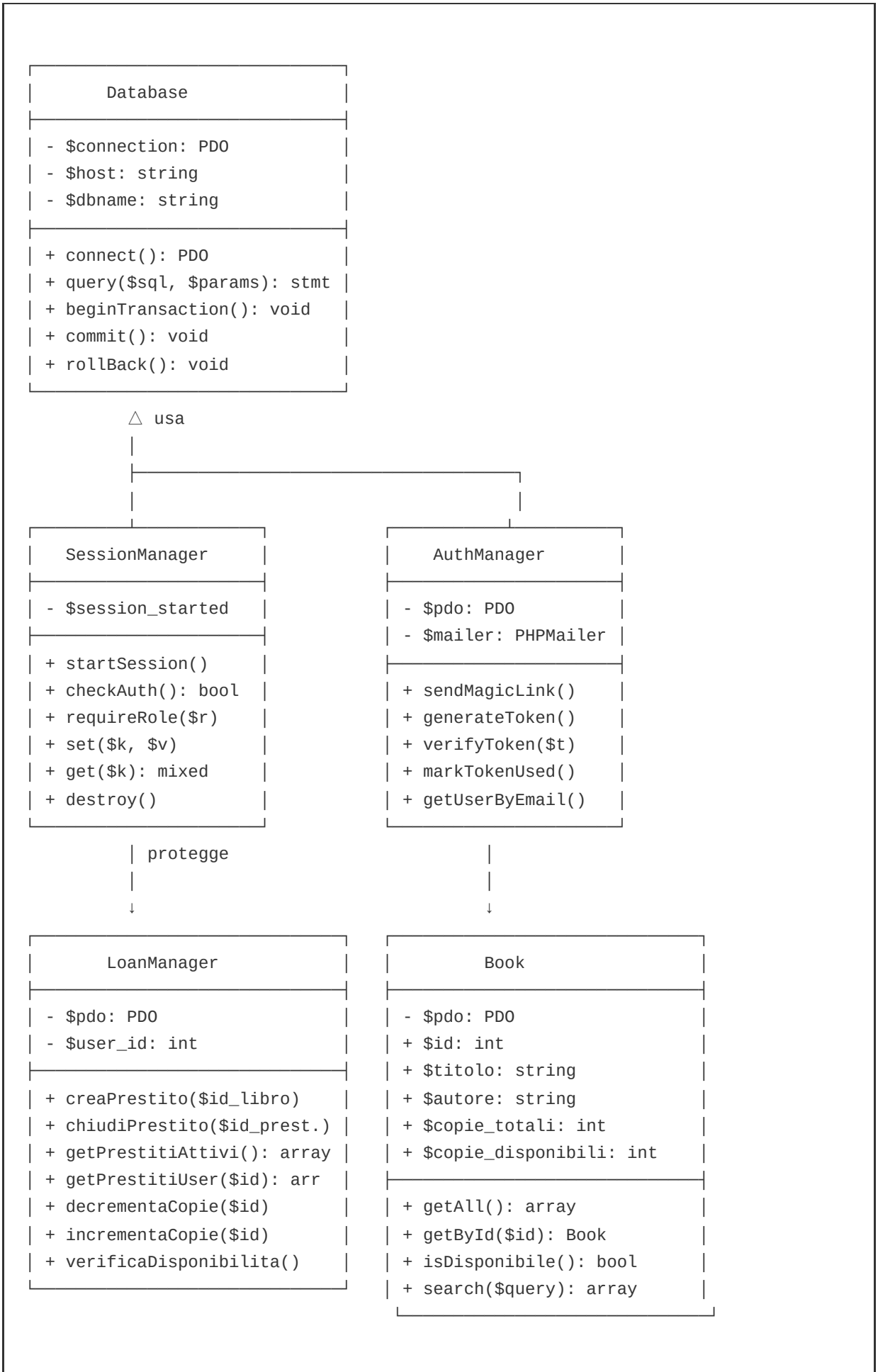
```
// 2. Controllo duplicati
$existing = db_fetch_one(
       'SELECT id FROM prestiti WHERE id_utente = ? AND id_libro = ? AND
data_restituzione IS NULL',
    [$user_id, $libro_id]
);

// 3. INSERT prestito
db_query('INSERT INTO prestiti ...');

// 4. UPDATE giacenza
db_query('UPDATE libri SET copie_disponibili = copie_disponibili - 1 ...');

$pdo->commit();
```

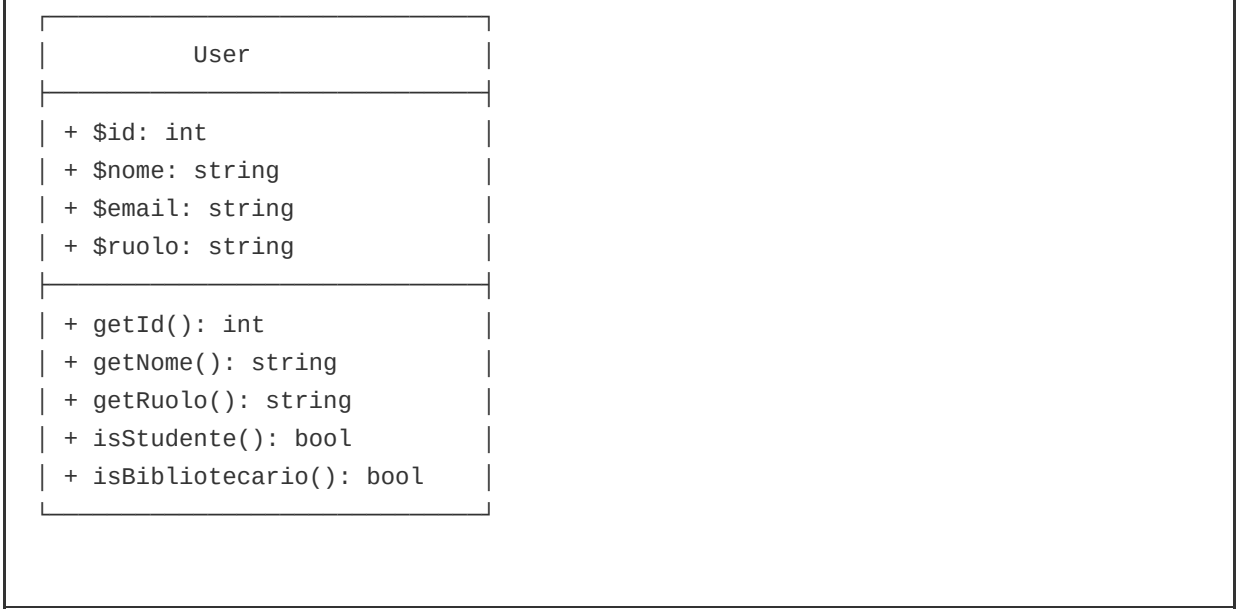
Restituzione (Bibliotecario)

```
// gestione_restituzioni.php
$pdo->beginTransaction();

// 1. SELECT FOR UPDATE
$prestito = db_fetch_one('SELECT ... FOR UPDATE');

// 2. UPDATE data_restituzione
db_query('UPDATE prestiti SET data_restituzione = CURDATE() ...');

// 3. UPDATE giacenza
db_query('UPDATE libri SET copie_disponibili = copie_disponibili + 1 ...');

$pdo->commit();
```

9. Diagramma UML Classi Logiche

Rappresentazione logica delle classi (o moduli funzionali in PHP procedurale) che gestiscono il sistema.





Descrizione Classi/Moduli

Database

Gestisce la connessione PDO centralizzata e fornisce metodi per query parametrizzate e gestione transazioni. Implementato in `config/db.php`.

SessionManager

Gestisce il ciclo di vita della sessione PHP: avvio, verifica autenticazione, controllo ruoli, distruzione. Implementato in `includes/auth.php`.

AuthManager

Gestisce l'intero flusso di autenticazione passwordless: generazione token CSPRNG, invio email via PHPMailer, verifica validità token (esistenza, scadenza, monouso), creazione sessione utente. Implementato in `login.php` e `verify.php`.

LoanManager

Core business logic per la gestione dei prestiti. Gestisce transazioni atomiche per garantire coerenza tra tabella prestiti e contatore copie disponibili. Include protezione race condition con FOR UPDATE. Implementato in `libro.php` e `gestione_restituzioni.php`.

Book

Rappresenta un libro del catalogo con metodi per recupero dati, verifica disponibilità, ricerca. Implementato in `libri.php` e `libro.php`.

User

Rappresenta un utente autenticato con metodi per verificare il ruolo e recuperare informazioni. Dati memorizzati in `$_SESSION` dopo login.

Pattern e Principi Applicati

- **Singleton Pattern:** Connessione database unica e condivisa
- **Guard Pattern:** Controllo accessi centralizzato in auth.php
- **Transaction Script:** Logica di business in funzioni procedurali
- **Separation of Concerns:** Config separato, auth separato, business logic separata
- **DRY (Don't Repeat Yourself):** Codice riutilizzabile attraverso include

10. Dettaglio Implementazione File per File

`src/config/db.php`

Gestisce la connessione PDO e definisce le funzioni helper `db_query`, `db_fetch_one`, `db_fetch_all`. Usa variabili d'ambiente con fallback.

`src/includes/auth.php`

Contiene tutta la logica di sessione e autorizzazione. Implementa `requireRole` che accetta sia stringa che array di ruoli.

`src/login.php`

Gestisce l'input email. Se l'utente esiste, genera token, lo salva in DB e invia email tramite PHPMailer (configurato con Mailtrap).

`src/verify.php`

Valida il token (esistenza, scadenza, uso). Se valido, crea la sessione e reindirizza all'URL salvato o alla dashboard corretta.

`src/libri.php`

Mostra il catalogo. Supporta filtri GET: `search` (titolo/autore) e `filter` (disponibili/non disponibili).

`src/libro.php`

Dettaglio libro. Gestisce il POST per il prestito. Controlla se l'utente ha già una copia attiva dello stesso libro.

`src/gestione_restituzioni.php`

Dashboard bibliotecario. Mostra statistiche (card colorate) e tabella prestiti attivi con azioni di restituzione.

11. Docker e Containerizzazione

Il progetto utilizza Docker Compose con due servizi principali.

docker-compose.yaml (REALE)

```
services:
  web:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: bibliotech_web
    ports:
      - "8085:80"  # Porta 8085 mappata
    environment:
      - DB_HOST=db
      - MAILTRAP_HOST=sandbox.smtp.mailtrap.io
      - APP_URL=http://localhost:8085
    volumes:
      - ./src:/var/www/html
      - vendor_data:/var/www/html/vendor

  db:
    image: mysql:8.0
    container_name: bibliotech_db
    ports:
      - "3306:3306"
    volumes:
      - db_data:/var/lib/mysql
      - ./sql:/docker-entrypoint-initdb.d:ro
```

12. Frontend e UI/UX

L'interfaccia utilizza **Bootstrap 5.3.0** e CSS personalizzato.

Caratteristiche UI

- **Badge Colorati:**
  - Verde: Prestito OK / Libro Disponibile
  - Giallo: Scadenza vicina (21-30 gg)
  - Rosso: In ritardo (>30 gg) / Non disponibile
- **Animazioni:** Classe `.fade-up` con delay progressivi (`fade-up-delay-1`, etc.).
- **Font:** 'Playfair Display' per titoli, 'DM Sans' per testo.
- **Logo:** `/assets/IMG/logo.png` integrato in navbar e login.

13. Email HTML Template

> Le email inviate non sono testo semplice, ma HTML stilizzato:
>
> ```
> Subject: BiblioTech — Il tuo link di accesso
> ```

[Header Blu Scuro: 📚 BiblioTech]

Ciao Mario Rossi,
Hai richiesto di accedere a BiblioTech. Clicca sul pulsante qui sotto:

[Pulsante Blu: Accedi a BiblioTech →]

Oppure copia questo link:
http://localhost:8085/verify.php?token=...

⏱ Valido 15 minuti · Monouso

14. Sicurezza e Best Practices

Feature	Implementazione Reale
Session Fixation	`session_regenerate_id(true)` in `createUserSession`
Race Conditions	`SELECT ... FOR UPDATE` nelle transazioni
Sanitizzazione	`filter_input`, `FILTER_VALIDATE_EMAIL`, `FILTER_VALIDATE_INT`
SQL Injection	Prepared Statements (PDO) ovunque
Token Security	`bin2hex(random_bytes(32))` (CSPRNG)
Logging	`error_log()` per tentativi accesso non autorizzati
Access Control	`requireRole()` blocca accesso studenti a pagine admin

15. Testing e Validazione

Checklist Funzionale

- Login con `mario.rossi@example.com` -> Ricezione email su Mailtrap.
- Click link -> Redirect a `/libri.php`.
- Login con `anna.biblioteca@example.com` -> Redirect a `/gestione_restituzioni.php`.
- Prestito libro -> Decremento copie, comparsa in "I Miei Prestiti".
- Tentativo doppio prestito stesso libro -> Errore bloccante.
- Restituzione -> Incremento copie, aggiornamento stato.

16. Deployment e Configurazione Produzione

Per il deployment in produzione:

1. Cambiare le credenziali nel file `.env`.
2. Impostare `APP_URL` al dominio reale.
3. Sostituire le credenziali Mailtrap con un server SMTP reale (SendGrid, AWS SES).
4. Rimuovere la mappatura porta 3306 del database per sicurezza.
5. Configurare HTTPS (SSL/TLS).

17. Conclusioni

Il sistema BiblioTech, nella sua implementazione attuale (v1.0.0), offre una soluzione robusta e sicura per la gestione bibliotecaria. L'uso di tecnologie standard (PHP/MySQL/Docker) combinato con pattern di sicurezza avanzati (Magic Link, Lock Pessimistico) garantisce affidabilità e protezione dei dati. L'interfaccia utente curata e responsive assicura un'esperienza utente fluida sia per gli studenti che per il personale.

— Fine Documentazione Tecnica —