# Design Document

Project Work

[Design Document 1](#_Toc206677444)

[**1.** **Overview** 3](#_Toc206677445)

[1.1 Introduzione 3](#_Toc206677446)

[1.2 Tecnologie utilizzate 3](#_Toc206677447)

[1.3 Versioni Documento 3](#_Toc206677448)

[1.4 Obiettivo del progetto 3](#_Toc206677449)

[1.5 Fasi 4](#_Toc206677450)

[1.5.1 Fase 1 – Setup e Ambiente 4](#_Toc206677451)

[1.5.2 Fase 2 – DB MySQL 4](#_Toc206677452)

[1.5.3 Fase 3 – MuleSoft API: System APIs, Process API, Experience API 4](#_Toc206677453)

[1.5.4 Fase 4 – Frontend 5](#_Toc206677454)

[1.5.5 Fase 5 – Testing 5](#_Toc206677455)

[1.5.6 Fase 6 - Deploy 5](#_Toc206677456)

[**2. Fase 1 - Setup e Ambiente** 5](#_Toc206677457)

[2.1 Setup Google Developer Account & API Key su Google Cloud Console. 5](#_Toc206677458)

[2.2 Inizializziamo Git 8](#_Toc206677459)

[2.3 Setup Ambiente Locale 9](#_Toc206677460)

[**3 Fase 2 – DB MySQL** 12](#_Toc206677461)

[3.1 Creazione tabelle database 12](#_Toc206677462)

[*3.1.1 Compiliamo 01\_schema\_creation.sql* 12](#_Toc206677463)

[*3.1.2 Creiamo la tabella principale libri:* 12](#_Toc206677464)

[*3.1.3 creiamo la tabella autori:* 13](#_Toc206677465)

[*3.1.4 creiamo la tabella (many-to-many) libro-autori* 13](#_Toc206677466)

[*3.1.5 creiamo la tabella per identificare i libri (tramite ISBN, ISSN, ...)* 13](#_Toc206677467)

[*3.1.6 creiamo la tabella dei prezzi dei libri:* 14](#_Toc206677468)

[*3.1.7 creiamo la tabella categorie libri:* 14](#_Toc206677469)

[*3.1.8 creiamo la tabella (many-to-many) libri-categorie:* 14](#_Toc206677470)

[*3.1.9 creaiamo la tabella che raccoglierà le analytics (log delle ricerche):* 14](#_Toc206677471)

[*3.1.10 verifica di creazione delle tabelle create:* 15](#_Toc206677472)

[3.2 Test su PhpMyAdmin 15](#_Toc206677473)

[3.3 Settaggio ambiente per testing 15](#_Toc206677474)

[3.3.1 Settaggio iniziale per testare - Dettaglio dei Parametri che serviranno 15](#_Toc206677475)

[3.3.2 Mappatura dall'API Google Books 16](#_Toc206677476)

[3.3.3 Esempio Chiamata InsertBookComplete(): 18](#_Toc206677477)

[3.3.4 Creazione della procedura di inserimento 18](#_Toc206677478)

# **Overview**

## 1.1 Introduzione

Il presente documento contiene il dettaglio di progetto per una applicazione che funga da filo conduttore riguardo alle competenze che ho acquisito finora.

## 1.2 Tecnologie utilizzate

|  |  |  |
| --- | --- | --- |
| **Settore** | **Tecnologia** | **Note** |
| Front-end | React + Tailwind (su Vercel) |  |
| Middle-end | MuleSoft (RAML, Connectors, DataWeave, mUnit, ...) |  |
| Back-end | MySQL su MAMP + phpMyAdmin |  |

## 1.3 Versioni Documento

|  |  |  |
| --- | --- | --- |
| **Versione** | **Data** | **Note** |
| 1.0 |  | Prima Release |
|  |  |  |

## 

## 1.4 Obiettivo del progetto

Gestire un catalogo libri interno arricchito con Google Books API

├ **API-led Connectivity**

│ ├ **System API** – BooksSystemAPI

│ ∟ CRUD sul database MySQL locale

│ ├ **System API** – GoogleBooksSystemAPI

│ ∟ Wrapper su Google Books API (HTTP Request)

│ ├ **Process API** – BookFinderProcessAPI

│ ∟ Orchestrazione: cerca nel DB → se non trova chiama Google Books → salva nel DB → ritorna risultato

│ ∟ **Experience API** – BookFinderExperienceAPI

│ ∟ Espone endpoint REST a frontend React

∟ **Database** (MySQL su MAMP)

├ books

├ authors / categories (many-to-many)

├ book\_identifiers / book\_prices

∟ search\_logs (analytics)

Struttura Architetturale:

[Frontend React]

[Experience API - BookFinderExperienceAPI]

[Process API - BookFinderProcessAPI]

[BooksSystemAPI] [GoogleBooksSystemAPI] [SearchLogsSystemAPI]

[MySQL DB] [Google Books API]

## 1.5 Fasi

### 1.5.1 Fase 1 – Setup e Ambiente

Creazione repo, folders e variabili d’ambiente

### 1.5.2 Fase 2 – DB MySQL

Definizione schema libri, autori, categorie, log. Per non dipendere solo da Google Books api creiamo i dati interni strutturati da preservare.

### 1.5.3 Fase 3 – MuleSoft API: System APIs, Process API, Experience API

**System APIs**:   
- BookSystemAPI: API MuleSoft che espone il DB locale. Nessun frontend deve connettersi direttamente al DB, MuleSoft agisce da ‘porta sicura’.  
- GoogleBooksSystemAPI: API MuleSoft che incapsula Google Books. Standardizziamo la risposta (Google può cambiare formato, MuleSoft no). Chi usa la nostra API non deve adattarsi a Google.

**Process API:**  
- Orchestrazione: prima cerca nel DB, poi eventualmente in Google Books, infine salva il risultato. Il frontend non deve preoccuparsi di ‘dove cercare’.

**Experience API:**  
- Endpoint unico per il frontend: /books?q=javascript. L’app non deve preoccuparsi di DB, Google, logiche. Fa una richiesta ad una sola Api e riceve un JSON pulito

### 1.5.4 Fase 4 – Frontend

Interfaccia utente che consuma l’Experience API.

### 1.5.5 Fase 5 – Testing

Test unitari con mUnit e con Postman

### 1.5.6 Fase 6 - Deploy

Il trial su Cloudhub ha un limite di tempo, quindi valutiamo già da subito l’utilizzo del Docker.  
Aggiungiamo alla root del progetto il file docker-compose.yml con all’interno il seguente codice:

version: "3.8"

services:

frontend:

build: ./react-frontend

ports:

- "3000:3000"

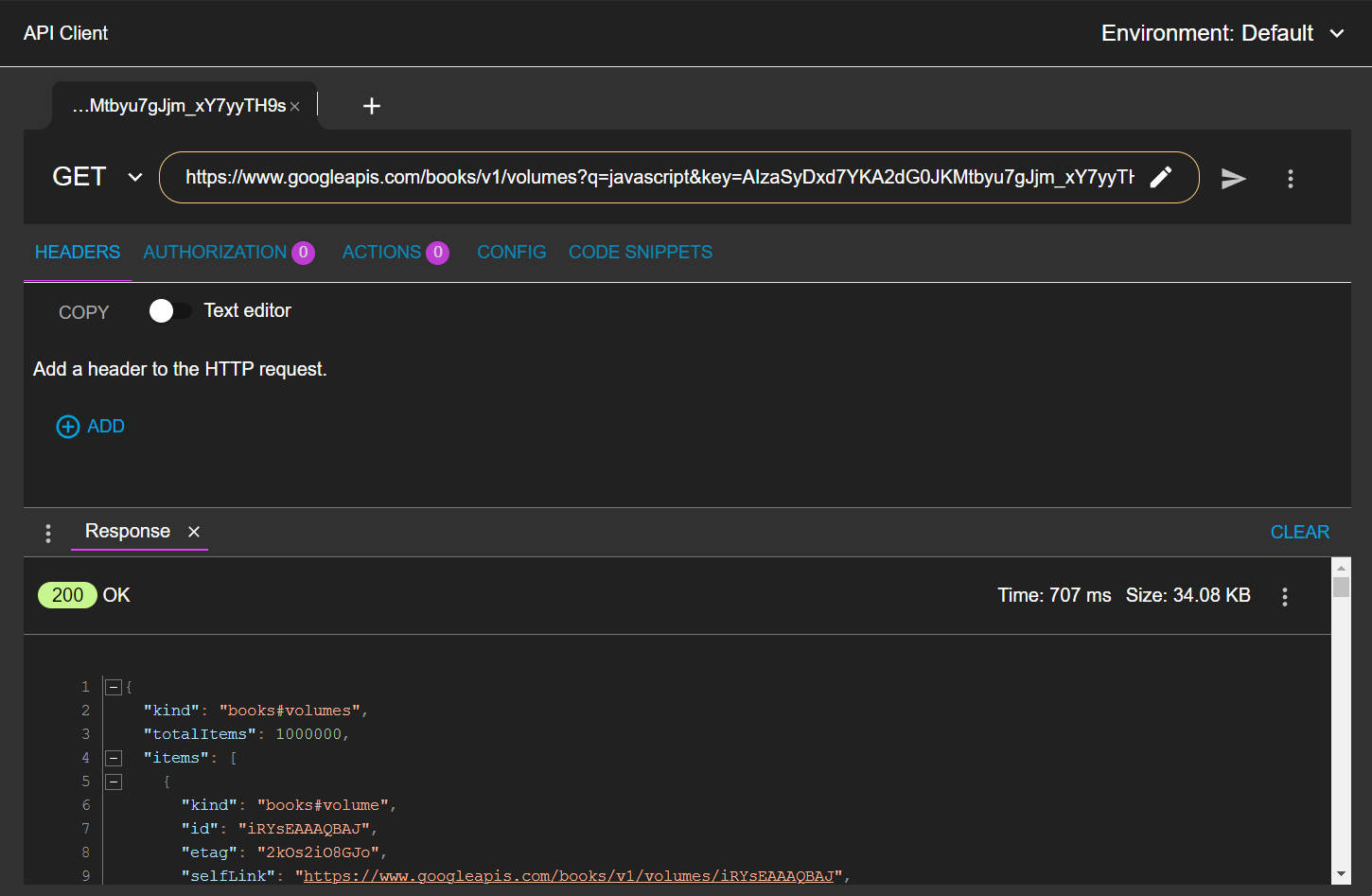
environment:

- NODE\_ENV=production

## **2. Fase 1 - Setup e Ambiente**

## 2.1 Setup Google Developer Account & API Key su Google Cloud Console.

|  |  |
| --- | --- |
| **Immagine di riferimento** | **Descrizione step by step** |
|  | Andiamo su Google Cloud Console (<https://console.cloud.google.com>) Click sul button in alto, a sinistra della barra di ricerca, per Aprire il selettore di progetti (oppure attraverso lo shortcut Ctrl O). |
|  | Si aprirà una finestra pop-up. Faccio click su “Nuovo progetto” a cui assegno il nome BookFinder-Portfolio |
|  | A sinistra, nel menu laterale a scomparsa, seleziono prima "API & Services" poi "Library" e nella barra di ricerca in alto centrale digito "Books", e cerco.  Verrà restituito un solo risultato, qualora in futuro ve ne fossero di più quello che interessa a noi e che dovremo abilitare (tramite click sul button “Abilita”) è quello che riporta la scritta:   “Books API - Google - The Google Books API allows clients to access the Google Books repository.” |
|  | All’abilitazione vedremo un redirect automatico su: <https://console.cloud.google.com/apis/api/books.googleapis.com/metrics?project=bookfinder-portfolio&inv=1&invt=Ab4VcQ>mentre la documentazione dell’API si trova a questo link: <https://developers.google.com/books/docs/v1/getting_started?csw=1&hl=it> |
|  | Clicco su “Credenziali” dal menu a sinistra,  poi sul button “+ Crea credenziali”  e dal menu a tendina seleziono “Chiave API” |
|  | Si aprirà una finestra riportante la nostra Api Key. Copiamolo e clicchiamo sul button “Chiudi” |
| Ora modifico le restrizioni, clicco sui tre | puntini a sinistra e su “Modifica chiave API”: |
|  |  |
|  | All’interno del radio button “Restrizioni dell’Applicazione” lascio: Nessuna (per sviluppo locale)  All’interno del radio button “Restrizione API” seleziono “Limita Chiave” e sul menu a tendina che si aprirà flaggo “Google Books API”,   poi clicco su OK e infine su Salva.  Fine del processo di configurazione. |

Come ultima cosa, eseguo un test su ARC, lanciando una GET su [https://www.googleapis.com/books/v1/volumes?q=javascript&key=YOUR\_API\_KEY](https://www.googleapis.com/books/v1/volumes?q=javascript&key=YOUR_API_KEY%22)   
(sostituendo ovviamente YOUR\_API\_KEY con la Chiave API assegnataci negli step precedenti)  
  
**✅ Validazione**: Risposta JSON con almeno 1 libro su JavaScript

## 2.2 Inizializziamo Git

|  |  |
| --- | --- |
| **Immagine di riferimento** | **Descrizione step by step** |
|  | Apriamo il cmd (io lo apro da VSC)  E col comando cd +path  Entramo nella nostra directory di lavoro |
|  | Passiamo dal cmd base alla bash  e inizializziamo GIT attraverso il comando  git init  echo "# BookFinder Portfolio Project" > README.md  git add README.md  git commit -m "Initial commit" |

## 

## 2.3 Setup Ambiente Locale

|  |  |
| --- | --- |
| **Immagine di riferimento** | **Descrizione step by step** |
|  | Avviamo Mamp  startando i server  Apache e MySQL |
|  | Creiamo un nuovo database  bookfinder\_db |
|  | Clicchiamo sul tab SQL e creiamo un USER con:  CREATE USER 'bookfinder'@'localhost' IDENTIFIED BY 'bf\_2025\_secure'; GRANT ALL PRIVILEGES ON bookfinder\_db.\* TO 'bookfinder'@'localhost'; FLUSH PRIVILEGES; |

|  |  |
| --- | --- |
|  | All’interno del folder BookFinder-Project creiamo la cartella react-frontend che ospiterà Next.Js  cd react-frontend/  npx create-next-app@latest . --typescript --tailwind --eslint --app  npm install axios lucide-react |
|  | Apriamo Anypoint Studio e creiamo un nuovo progetto sempre all’interno di BookFinder-Project e lo chiamiamo “mulesoft-api” |
|  | Ora che abbiamo creato i progetti per mulesoft e react/nextjs, apriamo la directory genitore su vsc e nella root creiamo i seguenti folder e file:  > folder database/  files al suo interno: 01\_schema\_creation.sql 02\_stored\_procedures.sql 03\_views\_and\_indexes.sql 04\_test\_data.sql README.md  > folder rpa-simulation/  folder al suo interno:  input-pdfs (servirà per i PDF di test)  output-csv (per i CSV generati)  scripts (per gli Script Python) README.md |
|  | > folder docs/  folder e files al suo interno  architecture-diagrams e screenshots  API\_DOCUMENTATION.md  DEPLOYMENT\_GUIDE.md  PROJECT\_OVERVIEW.md  TESTING\_GUIDE.md  > folder config/  File al suo interno:  .env.template (Template per env variables)  deployment.yaml (config deploy) README.md |
|  | Se non lo si è già fatto prima, creare la repository su git  (non spuntare il readMe perché lo abbiamo già creato nello step precedente) |
|  | Creare il file docker-compose.yml nella root del progetto.   version: "3.8"    services:  frontend:  build: ./react-frontend  ports:  - "3000:3000"  environment:  - NODE\_ENV=production |
|  | Aggiungere e pushare il file appena creato su git:  git add docker-compose.yml  git commit -m "add docker-compose file for backup deployment"  git push |
|  | Compilare le varie variabili d’ambiente:  BookFinder Project - Environment Variables Template  > Google Books API Configuration  GOOGLE\_BOOKS\_API\_KEY=XXXyour\_actual\_api\_key\_hereXXX GOOGLE\_BOOKS\_BASE\_URL=https://www.googleapis.com/books/v1  > Database Configuration (MAMP Local)  DB\_HOST=localhost DB\_PORT=3306 DB\_NAME=bookfinder\_db DB\_USER=bookfinder DB\_PASSWORD=bf\_2025\_secure  > MuleSoft Configuration  MULE\_APP\_NAME=bookfinder-api MULE\_ENV=local MULE\_HTTP\_PORT=8081  > React/Next.js Configuration  NEXT\_PUBLIC\_API\_BASE\_URL=http://localhost:8081/api NEXT\_PUBLIC\_APP\_NAME=BookFinder NEXT\_PUBLIC\_VERSION=1.0.0  > CloudHub Configuration (per deployment)  CLOUDHUB\_ORG\_ID=XXXyour\_org\_idXXX CLOUDHUB\_ENV=sandbox CLOUDHUB\_WORKER\_TYPE=micro CLOUDHUB\_WORKERS=1  > Docker Configuration  DOCKER\_IMAGE\_NAME=bookfinder-mulesoft-api DOCKER\_TAG=latest DOCKER\_PORT=8081  > Production Deployment (post-trial)  DOCKER\_REGISTRY=XXXyour\_docker\_hub\_usernameXXX BACKUP\_DEPLOYMENT\_STRATEGY=XXXgithub\_repo\_with\_docker\_composeXXX  > Development/Debug  DEBUG\_MODE=true LOG\_LEVEL=INFO RATE\_LIMIT\_REQUESTS\_PER\_MINUTE=60 |

## **3 Fase 2 – DB MySQL**

## 3.1 Creazione tabelle database

All’interno del file 01\_schema\_creation.sql storiamo gli schema per la creazione del nostro utente e delle tabelle libri, autori, categorie, prezzi, la tabella di raccolta dei log di ricerca, quella di relazione many-to-many libri-autori e quella per identificare i libri.

*3.1.1 Compiliamo 01\_schema\_creation.sql*  
  *iniziamo col selezionare il nostro db:*   
USE bookfinder\_db;   
 *creiamo il nostro utente (usando i dati settati in precedenza su PHPMyAdmin)*   
CREATE USER 'bookfinder'@'localhost' IDENTIFIED BY 'bf\_2025\_secure';   
 *assegnamo i privilegi all'utente:*   
GRANT ALL PRIVILEGES ON bookfinder\_db.\* TO 'bookfinder'@'localhost';   
 *quando creiamo un utente con GRANT, mysql potrebbe non applicare immediatamente i*  *cambiamenti, quindi aggiungiamo:*   
FLUSH PRIVILEGES;   
 *questa flush forza il reload, e serve a ricaricare le tabelle dei privilegi di mysql.*

*3.1.2 Creiamo la tabella principale libri:*   
CREATE TABLE books(   
 *usiamo l'id di google books in quanto google assegna id univoci alfanumerici, manteniamo*  *coerenza con la fonte esterna, preveniamo duplicati, e non ultimo semplifichiamo le*  *chiamate API in quanto il frontend passa direttamente l'id google. i 'comment' sono utili al*  *team dev, vengono mostrati quando si usa DESCRIBE o si usano tool di gestione DB*   
id VARCHAR(50) PRIMARY KEY COMMENT 'Google Books Volume ID',   
title VARCHAR(500) NOT NULL,   
subtitle VARCHAR (500) NULL,   
publisher VARCHAR (200) NULL,   
published\_year YEAR NULL,   
descr TEXT NULL,   
average\_rating DECIMAL(2,1) NULL CHECK (average\_rating BETWEEN 0 AND 5),   
rating\_countINT UNSIGNED DEFAULT 0,   
thumbnail\_url VARCHAR(500) NULL,   
lang VARCHAR(10) DEFAULT 'en',   
print\_type ENUM('BOOK', 'MAGAZINE') DEFAULT 'BOOK',   
main\_category VARCHAR(100) NULL,   
preview\_link VARCHAR(500) NULL,   
info\_link VARCHAR(500) NULL,   
created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,   
 *auto aggiorniamo la colonna ogni volta che il record viene modificato:*   
updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,  
 *inseriamo gli index per velocizzare le query (senza gli index mysql scannerizza*  *ogni riga invece di andare direttamente ai dati rilevanti)*  
INDEX idx\_title (title(100)),  
INDEX idx\_publisher (publisher),  
INDEX idx\_year (published\_year),  
INDEX idx\_rating (average\_rating),  
INDEX idx\_language (lang),  
FULLTEXT idx\_search (title, descr)  
) ENGINE=InnoDB COMMENT='Tabella books per memorizzare i libri';

*3.1.3 creiamo la tabella autori:*  
CREATE TABLE authors (   
id INT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,   
name VARCHAR(200) NOT NULL,   
name\_normalized VARCHAR(200) NOT NULL COMMENT 'Nome normalizzato per ricerche',   
created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,   
 *unique key impedisce duplicati nella colonna name, mysql blocca l'inserimento di autori già*  *esistenti (error 1062 duplicate entry for key 'uk\_name'), ed essendo un indice automatico*  *accelera le ricerche*  
UNIQUE KEY uk\_name(name),   
INDEX idx\_normalized(name\_normalized)  
) ENGINE=InnoDB COMMENT='Anagrafica autori';

*3.1.4 creiamo la tabella (many-to-many) libro-autori*   
CREATE TABLE book\_authors (   
book\_id VARCHAR(50) NOT NULL,   
author\_id INT UNSIGNED NOT NULL,   
author\_order TINYINT UNSIGNED DEFAULT 1 COMMENT 'Ordine autore nel libro',   
created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,   
PRIMARY KEY (book\_id, author\_id),   
FOREIGN KEY (book\_id) REFERENCES books(id) ON DELETE CASCADE,   
FOREIGN KEY (author\_id) REFERENCES authors(id) ON DELETE CASCADE,   
INDEX idx\_order (author\_order)   
) ENGINE=InnoDB COMMENT='Relazione molti-a-molti libri-autori';

*3.1.5 creiamo la tabella per identificare i libri (tramite ISBN, ISSN, ...)*   
CREATE TABLE book\_identifiers (   
id INT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,   
book\_id VARCHAR(50) NOT NULL,   
identifier\_type ENUM('ISBN\_10', 'ISBN\_13', 'ISSN', 'OTHER') NOT NULL,   
identifier\_value VARCHAR(20) NOT NULL,   
created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,   
UNIQUE KEY uk\_book\_type\_value (book\_id, identifier\_type, identifier\_value),   
FOREIGN KEY (book\_id) REFERENCES books(id) ON DELETE CASCADE,   
INDEX idx\_identifier (identifier\_value),   
INDEX idx\_type (identifier\_type)   
)ENGINE=InnoDB COMMENT='identificatori univoci dei libri';

*3.1.6 creiamo la tabella dei prezzi dei libri:*   
CREATE TABLE book\_prices(   
id INT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,   
book\_id VARCHAR(50) NOT NULL,   
price\_type ENUM('LIST', 'RETAIL') NOT NULL,   
amount DECIMAL(10,2) NOT NULL,   
currency\_code VARCHAR(3) NOT NULL DEFAULT 'EUR',   
country\_code VARCHAR(2) DEFAULT 'IT',   
saleability ENUM('FOR\_SALE', 'FREE', 'NOT\_FOR\_SALE', 'FOR\_PREORDER') DEFAULT 'FOR\_SALE', is\_ebook BOOLEAN DEFAULT FALSE,   
buy\_link VARCHAR(500) NULL,   
created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,   
updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,   
FOREIGN KEY (book\_id) REFERENCES books(id) ON DELETE CASCADE,   
INDEX idx\_price\_type (price\_type),   
INDEX idx\_currency (currency\_code),   
INDEX idx\_saleability (saleability)   
)ENGINE=InnoDB COMMENT='info prezzi e vendibilita';

*3.1.7 creiamo la tabella categorie libri:*   
CREATE TABLE categories(   
id INT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,   
name VARCHAR(100) NOT NULL,   
name\_normalized VARCHAR(100) NOT NULL,   
created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,   
UNIQUE KEY uk\_name(name),   
INDEX idx\_normalized(name\_normalized)   
)ENGINE=InnoDB COMMENT='categorie libri';

*3.1.8 creiamo la tabella (many-to-many) libri-categorie:*  
CREATE TABLE book\_categories (   
book\_id VARCHAR(50) NOT NULL,   
category\_id INT UNSIGNED NOT NULL,   
created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP, PRIMARY KEY (book\_id, category\_id),   
FOREIGN KEY (book\_id) REFERENCES books(id) ON DELETE CASCADE,   
FOREIGN KEY (category\_id) REFERENCES categories(id) ON DELETE CASCADE   
) ENGINE=InnoDB COMMENT='Relazione molti-a-molti libri-categorie';

*3.1.9 creaiamo la tabella che raccoglierà le analytics (log delle ricerche):*   
CREATE TABLE search\_logs(   
id INT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,   
search\_query VARCHAR(500) NOT NULL,   
result\_count INT UNSIGNED DEFAULT 0,   
execution\_time\_ms INT UNSIGNED NULL,   
search\_timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,   
source ENUM('GOOGLE\_API', 'LOCAL\_DB', 'BOTH') DEFAULT 'GOOGLE\_API',   
INDEX idx\_query (search\_query(100)),   
INDEX idx\_timestamp (search\_timestamp),   
INDEX idx\_source (source)   
)ENGINE=InnoDB COMMENT='log delle ricerche per analytics';

*3.1.10 verifica di creazione delle tabelle create:*   
SHOW TABLES;   
SELECT 'Schema creation completed successfully' as status;

## 3.2 Test su PhpMyAdmin

Torniamo su PhpMyAdmin e selezioniamo il nostro db bookfinder\_db, clicchiamo sul tab SQL e incolliamo il codice appena scritto nel punto 2.4. Clicchiamo su Esegui. Se va tutto bene, compariranno le varie tabelle sulla sinistra. Se fallisce con errore  
#1396 - Operation CREATE USER failed for 'bookfinder'@'localhost'  
significa che esiste già l’utente, quindi andiamo a eliminarlo eseguendo (sempre nel tab SQL)  
DROP USER IF EXISTS 'bookfinder'@'localhost';

## 3.3 Settaggio ambiente per testing

Su VSC compiliamo 04\_test.data.sql in modo da prepararci alcuni test da fare via via che procediamo. Facciamo una panoramica dei dati che ci servono per procedere agli inserimenti:

#### 3.3.1 Settaggio iniziale per testare - Dettaglio dei Parametri che serviranno

Logiche architetturali da seguire:

* **Atomicità**: o tutto inserito o niente (transazione)
* **Normalizzazione**: evitare duplicati in autori/categorie
* **Gestione Errori**: rollback automatico in caso di problemi
* **Performance**: una sola chiamata invece di 8+ INSERT
* **Consistenza**: logica centralizzata nel database

**Gestione Valori NULL:** La procedura gestisce automaticamente i campi NULL:

* **p\_subtitle**: Se Google non fornisce sottotitolo → NULL
* **p\_average\_rating**: Se non ci sono recensioni → NULL
* **p\_list\_price**: Se libro gratuito → NULL
* **p\_buy\_link**: Se non vendibile → NULL

Struttura Completa della Procedura:   
necessitiamo di **24 parametri totali** (15 semplici + 3 JSON + 6 pricing)

CREATE PROCEDURE InsertBookComplete(   
*--* ***PARAMETRI LIBRO PRINCIPALE*** (15 semplici)   
IN p\_book\_id VARCHAR(50),  
IN p\_title VARCHAR(500),  
IN p\_subtitle VARCHAR(500),  
IN p\_publisher VARCHAR(200),  
IN p\_published\_year YEAR,  
IN p\_description TEXT,  
IN p\_average\_rating DECIMAL(2,1),  
IN p\_ratings\_count INT,  
IN p\_thumbnail\_url VARCHAR(500),  
IN p\_page\_count INT,  
IN p\_language VARCHAR(10),  
IN p\_print\_type VARCHAR(20),  
IN p\_main\_category VARCHAR(100),   
IN p\_preview\_link VARCHAR(500),  
IN p\_info\_link VARCHAR(500),  
*--* ***PARAMETRI JSON COMPLESSI*** (3 Json)  
IN p\_authors JSON,  
IN p\_identifiers JSON,  
IN p\_categories JSON,   
*--* ***PARAMETRI PRICING*** (6 pricing)  
IN p\_list\_price DECIMAL(10,2),  
IN p\_retail\_price DECIMAL(10,2),  
IN p\_currency VARCHAR(3),  
IN p\_saleability VARCHAR(20),  
IN p\_is\_ebook BOOLEAN,  
IN p\_buy\_link VARCHAR(500)  
)

#### 3.3.2 Mappatura dall'API Google Books

**Parametri Semplici (15):**

|  |  |  |  |
| --- | --- | --- | --- |
| **Parametro** | **Tipo** | **Fonte API Google Books** | **Esempio Valore** |
| p\_book\_id | VARCHAR(50) | items[0].id | "gvQQCgAAQBAJ" |
| p\_title | VARCHAR(500) | items[0].volumeInfo.title | "JavaScript: The Good Parts" |
| p\_subtitle | VARCHAR(500) | items[0].volumeInfo.subtitle | "Unearthing the Excellence in JavaScript" |
| p\_publisher | VARCHAR(200) | items[0].volumeInfo.publisher | "O'Reilly Media" |
| p\_published\_year | YEAR | items[0].volumeInfo.publishedDate | 2008 (estratto da "2008-05-08") |
| p\_description | TEXT | items[0].volumeInfo.description | "Most programming languages contain..." |
| p\_average\_rating | DECIMAL(2,1) | items[0].volumeInfo.averageRating | 4.2 |
| p\_ratings\_count | INT | items[0].volumeInfo.ratingsCount | 892 |
| p\_thumbnail\_url | VARCHAR(500) | items[0].volumeInfo.imageLinks.thumbnail | "<http://books.google.com/books/content?id=>..." |
| p\_page\_count | INT | items[0].volumeInfo.pageCount | 170 |
| p\_language | VARCHAR(10) | items[0].volumeInfo.language | "en", "it", "fr" |
| p\_print\_type | VARCHAR(20) | items[0].volumeInfo.printType | "BOOK" o "MAGAZINE" |
| p\_main\_category | VARCHAR(100) | items[0].volumeInfo.mainCategory | "Computers", "Fiction" |
| p\_preview\_link | VARCHAR(500) | items[0].volumeInfo.previewLink | "<http://books.google.com/books?id=>..." |
| p\_info\_link | VARCHAR(500) | items[0].volumeInfo.infoLink | "<http://books.google.com/books?id=>..." |

**Parametri JSON Complessi** (JSON parsing interno per gestire array/object complessi):

> p\_authors JSON:

// Fonte: items[0].volumeInfo.authors[]  
// Input alla procedura:  
'["Douglas Crockford", "John Doe"]'  
// Elaborazione interna:  
// 1. Parse JSON array  
// 2. INSERT IGNORE INTO authors per ogni nome  
// 3. INSERT INTO book\_authors con author\_order

> p\_identifiers JSON:  
// Fonte: items[0].volumeInfo.industryIdentifiers[]  
// Input alla procedura:  
'[{"type":"ISBN\_13","identifier":"9780596517748"},{"type":"ISBN\_10","identifier":"0596517742"}]'  
// Elaborazione interna:  
// 1. Parse JSON objects  
// 2. INSERT INTO book\_identifiers  
// 3. Mapping: type -> identifier\_type, identifier -> identifier\_value

> p\_categories JSON:  
// Fonte: items[0].volumeInfo.categories[]  
// Input alla procedura:  
'["Computers", "Programming Languages", "JavaScript"]'  
// Elaborazione interna:  
// 1. Parse JSON array  
// 2. INSERT IGNORE INTO categories  
// 3. INSERT INTO book\_categories

**Parametri Pricing:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Parametro** | **Tipo** | **Fonte API Google Books** | **Esempio** |
| p\_list\_price | DECIMAL(10,2) | items[0].saleInfo.listPrice.amount | 29.99 |
| p\_retail\_price | DECIMAL(10,2) | items[0].saleInfo.retailPrice.amount | 24.99 |
| p\_currency | VARCHAR(3) | items[0].saleInfo.listPrice.currencyCode | "USD", "EUR" |
| p\_saleability | VARCHAR(20) | items[0].saleInfo.saleability | "FOR\_SALE", "FREE", "NOT\_FOR\_SALE" |
| p\_is\_ebook | BOOLEAN | items[0].saleInfo.isEbook | true / false |
| p\_buy\_link | VARCHAR(500) | items[0].saleInfo.buyLink | "<https://books.google.com/books?id=>..." |

#### 3.3.3 Esempio Chiamata InsertBookComplete():

CALL InsertBookComplete(  
 -- Dati semplici libro  
 'gvQQCgAAQBAJ', -- Google Books ID  
 'JavaScript: The Good Parts', -- Titolo  
 'Unearthing the Excellence in JavaScript', -- Sottotitolo  
 'O\'Reilly Media', -- Editore  
 2008, -- Anno pubblicazione  
 'Most programming languages contain good parts..', -- Descrizione  
 4.2, -- Rating medio  
 892, -- Numero recensioni  
 '<http://books.google.com/books/content?id=>...', -- URL thumbnail  
 170, -- Pagine  
 'en', -- Lingua  
 'BOOK', -- Tipo  
 'Computers', -- Categoria principale  
 '<http://books.google.com/books?id=...preview>', -- Link anteprima  
 '<http://books.google.com/books?id=...info>', -- Link info  
   
 -- JSON complessi  
 '["Douglas Crockford"]', -- Autori  
 '[{"type":"ISBN\_13","identifier":"9780596517748"}]', -- ISBN  
 '["Computers", "Programming Languages"]', -- Categorie  
   
 -- Pricing  
 29.99, -- Prezzo di listino  
 24.99, -- Prezzo retail  
 'USD', -- Valuta  
 'FOR\_SALE', -- Vendibilità  
 true, -- È ebook  
 '<https://books.google.com/books?id=...buy>' -- Link acquisto  
);

#### 3.3.4 Creazione della procedura di inserimento

Inseriamo una stored procedure (procedura memorizzata)  
CREATE PROCEDURE InsertBookComplete(   
-- Parametri in input IN p\_book\_id VARCHAR(50),   
-- ID Google Books IN p\_title VARCHAR(500),   
-- Titolo libro IN p\_authors JSON,   
-- ["Autore 1", "Autore 2"] IN p\_identifiers JSON,   
-- [{"type":"ISBN\_13","value":"123"}] IN p\_categories JSON,   
-- ["Fiction", "Drama"] -- ... altri parametri)