# 📚 Amazon Books Manager - Documentazione Completa

## 📋 Indice

1. [Panoramica del Progetto](#panoramica-del-progetto)

2. [Architettura del Sistema](#architettura-del-sistema)

3. [Installazione e Setup](#installazione-e-setup)

4. [Struttura del Database](#struttura-del-database)

5. [API Endpoints](#api-endpoints)

6. [Interfaccia Web](#interfaccia-web)

7. [Formato Dati](#formato-dati)

8. [Deploy e Produzione](#deploy-e-produzione)

9. [Troubleshooting](#troubleshooting)

10. [Sviluppi Futuri](#sviluppi-futuri)

──────────────────────────────────────────────────

## 🎯 Panoramica del Progetto

**Amazon Books Manager** è una applicazione web per la gestione e analisi di libri Amazon, progettata per aiutare editori e analisti a monitorare performance, royalties e metriche di vendita.

### Caratteristiche Principali

• 💾 \*\*Database SQLite\*\* per storage persistente

• 🔍 \*\*Sistema di filtri avanzati\*\* (BSR, NET gain, parole chiave)

• 📊 \*\*Dashboard con statistiche\*\* in tempo reale

• 🖼️ \*\*Visualizzazione immagini\*\* delle copertine

• 📱 \*\*Interfaccia responsive\*\* desktop/mobile

• 🗑️ \*\*Gestione CRUD completa\*\* dei libri

### Stack Tecnologico

• \*\*Backend\*\*: Node.js + Express.js

• \*\*Database\*\*: SQLite3

• \*\*Frontend\*\*: HTML5 + CSS3 + Vanilla JavaScript

• \*\*Styling\*\*: CSS Grid + Flexbox (design moderno)

──────────────────────────────────────────────────

## 🏗️ Architettura del Sistema

amazon-books-manager/
├── server.js # Server Express principale
├── public/ # File statici web
│ └── index.html # Interfaccia utente
├── amazon\_books.db # Database SQLite (auto-generato)
├── package.json # Dipendenze Node.js
└── README.md # Questa documentazione

### Flusso dei Dati

1. \*\*Input\*\*: Dati JSON tramite POST API o interfaccia web

2. \*\*Processing\*\*: Validazione e trasformazione nel server Express

3. \*\*Storage\*\*: Persistenza in database SQLite

4. \*\*Output\*\*: Visualizzazione nell'interfaccia web con filtri

──────────────────────────────────────────────────

## ⚙️ Installazione e Setup

### Prerequisiti

• \*\*Node.js\*\* >= 14.0.0

• \*\*npm\*\* >= 6.0.0

### Setup Passo-Passo

1. \*\*Clona/Scarica il progetto\*\*

mkdir amazon-books-manager
cd amazon-books-manager

1. \*\*Inizializza il progetto Node.js\*\*

npm init -y

1. \*\*Installa le dipendenze\*\*

npm install express sqlite3 cors

1. \*\*Crea la struttura cartelle\*\*

mkdir public

1. \*\*Salva i file\*\*
- Copia `server.js` nella root
- Copia `index.html` in `public/index.html`

2. \*\*Avvia il server\*\*

node server.js

1. \*\*Accedi all'applicazione\*\*
- Apri il browser su: `http://localhost:3100`

### Script package.json Consigliati

{
"scripts": {
"start": "node server.js",
"dev": "nodemon server.js",
"test": "echo \"No tests specified\""
}
}

──────────────────────────────────────────────────

## 🗄️ Struttura del Database

### Tabella `books`

### Indici Automatici

• \*\*PRIMARY KEY\*\* su `asin`

• \*\*Timestamp automatici\*\* per `created\_at` e `updated\_at`

──────────────────────────────────────────────────

## 🛠️ API Endpoints

### Base URL: `http://localhost:3100`

### 1. \*\*POST /api/books\*\* - Salva Libri

Salva uno o più libri nel database.

**Request Body:**

{
"asin": "B08XYZ123",
"title": "Il Grande Gatsby",
"author": "F. Scott Fitzgerald",
"price": "$12.99",
"bsr": 1500,
"ratings": 1250,
"monthlyNETGain": 75.00,
"description": "Un classico della letteratura americana..."
}

**Response:**

{
"success": true,
"message": "1 libri salvati",
"errors": []
}

### 2. \*\*GET /api/books\*\* - Lista Libri con Filtri

Recupera libri con filtri e paginazione.

**Query Parameters:**

?titleKeyword=gatsby
&minNetGain=50
&maxNetGain=200
&minBsr=1000
&maxBsr=10000
&publisher=penguin
&sortBy=monthly\_net\_gain
&sortOrder=DESC
&limit=20
&offset=0

**Response:**

{
"books": [...],
"count": 20,
"total": 156,
"hasMore": true
}

### 3. \*\*GET /api/books/:asin\*\* - Dettaglio Libro

Recupera un libro specifico.

**Response:**

{
"asin": "B08XYZ123",
"title": "Il Grande Gatsby",
"author": "F. Scott Fitzgerald",
"monthly\_net\_gain": 75.00,
"description": "Un classico...",
...
}

### 4. \*\*DELETE /api/books/:asin\*\* - Elimina Libro

Rimuove un libro dal database.

**Response:**

{
"message": "Libro eliminato con successo"
}

### 5. \*\*GET /api/stats\*\* - Statistiche

Statistiche generali del database.

**Response:**

{
"total": { "count": 156 },
"byPublisher": [
{ "publisher": "Penguin", "count": 23 },
...
],
"avgRoyalties": { "avg\_royalties": 2.45 },
"topEarners": [...]
}

──────────────────────────────────────────────────

## 🌐 Interfaccia Web

### Pagina Principale (`/`)

Dashboard completa con:

#### Header con Statistiche

• \*\*Libri Totali\*\*: Conteggio database

• \*\*Royalties Medie\*\*: Media royalties per libro

• \*\*Libri Visualizzati\*\*: Risultati filtri correnti

#### Pannello Filtri

• 🔍 \*\*Parole chiave nel titolo\*\*

• 💰 \*\*NET Gain min/max\*\*

• 📊 \*\*BSR min/max\*\*

• 🏢 \*\*Editore\*\*

• 📋 \*\*Ordinamento\*\* (campo + direzione)

#### Griglia Libri

Ogni card mostra:

• 🖼️ \*\*Immagine copertina\*\* (con fallback)

• 📖 \*\*Titolo e autore\*\*

• 💵 \*\*NET Gain mensile\*\* (colorato)

• 📊 \*\*BSR e recensioni\*\*

• ⭐ \*\*Rating medio\*\*

• 🔗 \*\*Link Amazon\*\*

• 🗑️ \*\*Elimina\*\* e 📖 \*\*Dettagli\*\*

#### Modal Dettagli

Informazioni complete:

• \*\*Dati base\*\*: ASIN, editore, tipo, prezzo

• \*\*Metriche\*\*: BSR, recensioni, rating, vendite

• \*\*Finanziari\*\*: Costi, royalties, guadagni

• \*\*Fisici\*\*: Pagine, dimensioni, colori

• \*\*Descrizione completa\*\*

### Responsive Design

• \*\*Desktop\*\*: Griglia multi-colonna

• \*\*Tablet\*\*: Griglia adattiva

• \*\*Mobile\*\*: Singola colonna

──────────────────────────────────────────────────

Basandomi sui documenti forniti, posso descrivere la funzionalità del **convertitore Markdown** che è stata aggiunta al progetto Amazon Books Manager:

## 🔄 Convertitore Markdown → DOCX

### Panoramica della Funzionalità

Il convertitore permette di trasformare file Markdown (.md, .markdown) in documenti Word (.docx) mantenendo la formattazione originale. È integrato come modulo aggiuntivo nell'applicazione principale.

### Architettura Tecnica

#### Backend (server.js)

**Dipendenze aggiunte:**

• `multer`: Gestione upload file (max 10MB)

• `marked`: Parser Markdown

• `docx`: Generazione documenti Word

**Endpoint API:**

• `POST /api/convert-markdown`: Riceve file Markdown e restituisce DOCX

• `GET /markdown-converter`: Serve la pagina web del convertitore

**Funzioni principali:**

• `markdownToDocx()`: Converte il contenuto Markdown in documento Word

• `parseInlineFormatting()`: Gestisce formattazione inline (grassetto, corsivo, codice)

#### Frontend (markdown-converter.html)

**Interfaccia utente moderna:**

• \*\*Upload zone\*\* con drag & drop

• \*\*Validazione file\*\* (tipo e dimensione)

• \*\*Preview\*\* del file selezionato

• \*\*Indicatore di progresso\*\* durante conversione

• \*\*Download automatico\*\* del file convertito

### Funzionalità Supportate

Il convertitore gestisce questi elementi Markdown:

### Flusso di Conversione

1. \*\*Upload\*\*: Utente carica file .md tramite interfaccia web

2. \*\*Validazione\*\*: Controllo tipo file e dimensione (max 10MB)

3. \*\*Parsing\*\*: `marked.js` analizza la sintassi Markdown

4. \*\*Trasformazione\*\*: Conversione in oggetti `docx` (Paragraph, TextRun, etc.)

5. \*\*Generazione\*\*: Creazione buffer documento Word

6. \*\*Download\*\*: File DOCX scaricato automaticamente nel browser

### Navigazione Integrata

L'applicazione ora include una **navbar** che permette di navigare tra:

• 📚 \*\*Gestione Libri\*\* (funzionalità principale)

• 📝 \*\*Converti Markdown\*\* (nuova funzionalità)

### Caratteristiche UX

**Design responsive:**

• Layout adattivo desktop/mobile

• Drag & drop intuitivo

• Feedback visivo (loading, errori, successo)

• Animazioni moderne

**Gestione errori:**

• Validazione formati file

• Controllo dimensioni

• Messaggi di errore chiari

• Cleanup automatico file temporanei

### Sicurezza

• \*\*Validazione rigorosa\*\* dei tipi di file

• \*\*Limite dimensioni\*\* upload (10MB)

• \*\*Sanitizzazione\*\* del contenuto Markdown

• \*\*Cleanup automatico\*\* dei file temporanei sul server

Questa funzionalità estende significativamente l'utilità dell'applicazione, permettendo agli utenti di convertire facilmente documentazione tecnica, note o altri contenuti Markdown in formato Word professionale.

──────────────────────────────────────────────────

## 📊 Formato Dati

### Mapping Campi Input → Database

### Esempi di Payload Completi

**Libro Fisico:**

{
"asin": "1234567890",
"title": "Manuale di Programmazione",
"author": "Mario Rossi",
"publisher": "TechBooks",
"type": "Paperback",
"price": "€29.99",
"bsr": 5000,
"ratings": 847,
"ratingAvg": "4.3",
"pagesnum": 450,
"dimensions": "23.5 x 19.1 x 2.8 cm",
"printCost": 8.50,
"royalties": 5.20,
"monthlyNETGain": 156.00,
"description": "Guida completa alla programmazione...",
"isColor": "False"
}

**eBook Kindle:**

{
"asin": "B08ABCD123",
"title": "Digital Marketing 2024",
"author": "Laura Bianchi",
"type": "Kindle Edition",
"price": "€9.99",
"bsr": 1200,
"ratings": 234,
"ratingAvg": "4.7",
"royalties": 3.50,
"monthlyNETGain": 245.00,
"description": "Strategie moderne di marketing digitale..."
}

──────────────────────────────────────────────────

## 🚀 Deploy e Produzione

### Opzioni di Deploy

#### 1. \*\*Server VPS/Dedicato\*\*

# Installa PM2 per gestione processi
npm install -g pm2
# Avvia in produzione
pm2 start server.js --name "books-manager"
pm2 startup
pm2 save

#### 2. \*\*Docker Container\*\*

FROM node:16-alpine
WORKDIR /app
COPY package\*.json ./
RUN npm install --production
COPY . .
EXPOSE 3100
CMD ["node", "server.js"]

#### 3. \*\*Heroku\*\*

// package.json
{
"scripts": {
"start": "node server.js"
},
"engines": {
"node": "16.x"
}
}

### Configurazioni Produzione

#### Variabili Ambiente

export NODE\_ENV=production
export PORT=3100
export DB\_PATH=/data/amazon\_books.db

#### Reverse Proxy (Nginx)

server {
listen 80;
server\_name books.tuodominio.com;
location / {
proxy\_pass http://localhost:3100;
proxy\_set\_header Host $host;
proxy\_set\_header X-Real-IP $remote\_addr;
}
}

### Backup Database

# Backup automatico
cp amazon\_books.db backups/books\_$(date +%Y%m%d\_%H%M%S).db
# Cronjob giornaliero
0 2 \* \* \* /path/to/backup-script.sh

──────────────────────────────────────────────────

## 🔧 Troubleshooting

### Problemi Comuni

#### 1. \*\*Server non si avvia\*\*

# Verifica porta occupata
lsof -i :3100
# Cambia porta nel server.js se necessario

#### 2. \*\*Database non accessibile\*\*

# Verifica permessi
ls -la amazon\_books.db
chmod 664 amazon\_books.db

#### 3. \*\*Immagini non caricano\*\*

• Controlla URL immagini nel database

• Verifica CORS se immagini esterne

• Usa fallback automatico integrato

#### 4. \*\*Filtri non funzionano\*\*

• Controlla console browser per errori JavaScript

• Verifica formato parametri query

• Controlla validazione lato server

### Debug Mode

// Aggiungi al server.js per logging dettagliato
app.use((req, res, next) => {
console.log(`${new Date().toISOString()} - ${req.method} ${req.path}`);
console.log('Query:', req.query);
console.log('Body:', req.body);
next();
});

### Log Monitoring

# Con PM2
pm2 logs books-manager
# Tradizionale
node server.js >> app.log 2>&1

──────────────────────────────────────────────────

## 🔮 Sviluppi Futuri

### Features Pianificate

#### 1. \*\*Analisi Avanzate\*\*

• 📈 \*\*Grafici trend\*\* vendite/BSR

• 📊 \*\*Report PDF\*\* esportabili

• 🎯 \*\*Previsioni ML\*\* performance

#### 2. \*\*Integrazione APIs\*\*

• 🔄 \*\*Sync automatico\*\* dati Amazon

• 📧 \*\*Notifiche email\*\* BSR changes

• 📱 \*\*App mobile\*\* companion

#### 3. \*\*Multi-utente\*\*

• 👥 \*\*Sistema login\*\* utenti

• 🔐 \*\*Permessi\*\* per editore

• 👨‍💼 \*\*Dashboard admin\*\*

#### 4. \*\*Business Intelligence\*\*

• 💹 \*\*KPI Dashboard\*\* avanzata

• 🏆 \*\*Competitor analysis\*\*

• 📋 \*\*Custom reports\*\*

### Architettura Scalabile

┌─────────────────┐ ┌─────────────────┐
│ Load Balancer │ │ Redis │
│ (Nginx) │ │ (Cache) │
└─────────────────┘ └─────────────────┘
│ │
┌────┴────┐ │
│ │ │
┌───▼───┐ ┌───▼───┐ │
│Node.js│ │Node.js│ │
│App #1 │ │App #2 │ │
└───┬───┘ └───┬───┘ │
│ │ │
└────┬────┘ │
│ │
┌────────▼─────────────────────▼─┐
│ PostgreSQL │
│ (Production DB) │
└────────────────────────────────┘

### Contributing Guidelines

1. \*\*Fork\*\* del repository

2. \*\*Feature branch\*\*: `git checkout -b feature/nuova-funzionalita`

3. \*\*Test\*\* delle modifiche

4. \*\*Pull request\*\* con descrizione dettagliata

5. \*\*Code review\*\* prima del merge

──────────────────────────────────────────────────

## 📞 Supporto

### Risorse Utili

• 📖 \*\*Express.js Docs\*\*: https://expressjs.com/

• 💾 \*\*SQLite Docs\*\*: https://sqlite.org/docs.html

• 🌐 \*\*MDN Web Docs\*\*: https://developer.mozilla.org/

### Contatti

• 💬 \*\*Issues\*\*: Apri issue su GitHub

• 📧 \*\*Email\*\*: supporto@tuodominio.com

• 📞 \*\*Telefono\*\*: +39 XXX XXX XXXX

──────────────────────────────────────────────────

## 📄 Licenza

Questo progetto è rilasciato sotto licenza **MIT**. Vedi file `LICENSE` per dettagli completi.

──────────────────────────────────────────────────

*Documentazione aggiornata al: Giugno 2025*
*Versione progetto: 1.0.0*