**ReadWithPauline API Documentation***REST API Specification*Version 4.1.4 — October 2025  
Author: Pauline Chassigneu

Summary

The purpose of this document is to provide comprehensive documentation for the **ReadWithPauline API**, an online bookstore.

ReadWithPauline is a **modern online bookstore platform** where readers can browse, search, and purchase books curated by Pauline. The API enables customers to search for books, retrieve detailed information, and place orders. It exposes **REST endpoints** for book browsing, customer management, orders, and notifications. This API serves as the **backbone** for all user interactions on the ReadWithPauline platform.

All requests and responses use **HTTPS** with **JSON** format. Errors follow a **consistent JSON structure** for easier integration.

**Main components:**

- Frontend: Web application consuming the API.

- Backend: REST API

- Database: MySQL relational schema.

- External services: Payment gateway (mocked in v1), email/notification service.

**Data flow:**

1. Client → API Gateway → Backend

2. Backend → Database

3. Background jobs → Notifications and email sending

Document version control

| **Version** | **Date** | **Autor** | **Changes** |
| --- | --- | --- | --- |
| 1.0 | 23 october 2025 | Pauline CHASSIGNEU | Initial draft |
|  |  |  |  |

Table of Contents

[**1/ Swagger API Overview 2**](#_lgep2uq9v8n3)

[**2/ Authentication and Authorization 5**](#_bw33icc1z7s8)

[2.1 Authentication Method 5](#_vq8ij6ol0fln)

[2.2 Permissions Matrix 6](#_mbt8sqz6z5k)

[2.3 Security and audit 7](#_3mpwdvwrog47)

[**3/ Field Mapping 8**](#_cw9zpw74aw1r)

[**4/ Conversions 15**](#_c026xbvhjo0r)

[**5/ Error handling 18**](#_fb4w2dq3ck8v)

[5.1 Standard Error Structure 18](#_mk8wuygab6yk)

[5.2 HTTP status code used 18](#_510xe4df7n7t)

[5.3 Error codes by Category 19](#_ksfoga9lffq)

[**6/ Acceptance criteria 24**](#_4qwdbxrzt31z)

[6.1 General acceptance criteria 24](#_w7fdfb1o4f8z)

[6.2 Performance and reliability criteria 25](#_l1kufkhn5gku)

[**7/ Test case scenarios 26**](#_iq9gwbptqvc4)

[7.1 General 26](#_2fqj5jy2s9jn)

[7.2 Books 26](#_kqtw1g5lbo4j)

[7.3 Customers 28](#_nwv1oh4t7irb)

[7.4 Orders 29](#_ymwveeo6nwhg)

[7.5 Notifications 31](#_80gqni7o1eh7)

[**8/ SQL queries 32**](#_iuzd73k3a3i8)

[**9/ Use Case Diagrams and Flow Diagrams 35**](#_qoe0ub2soeit)

[9.1 Use Case Diagrams 35](#_xgsx7vh9trkb)

[9.2 Flow Diagrams 36](#_9lkcmohgqn6f)

[**10/ User stories 39**](#_pxt884jmqynn)

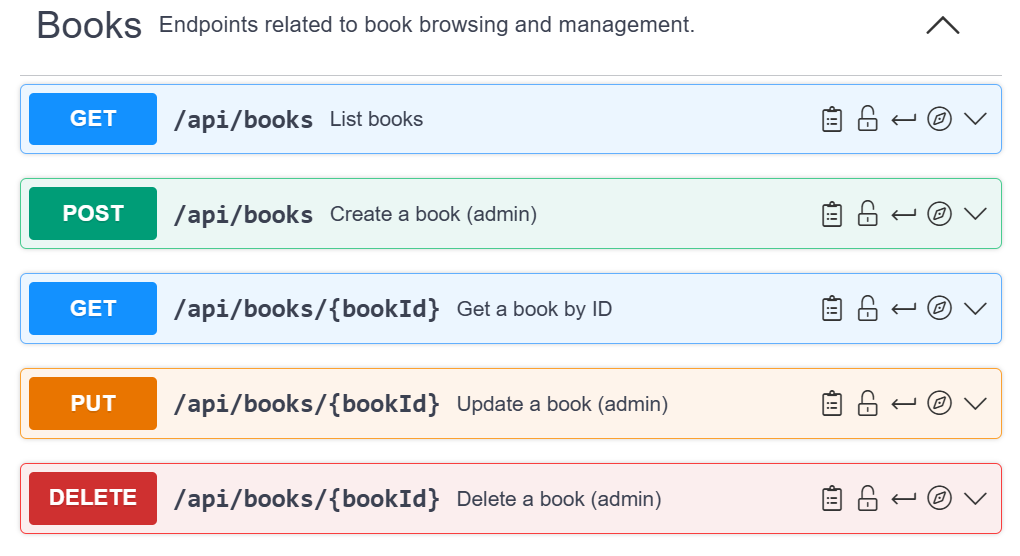
# **1/ Swagger API Overview**

RESTful API for browsing books, placing orders, managing customers, and availability notifications. Secured with API key authentication.

Here is the list of endpoints to consider for this first version of the API :

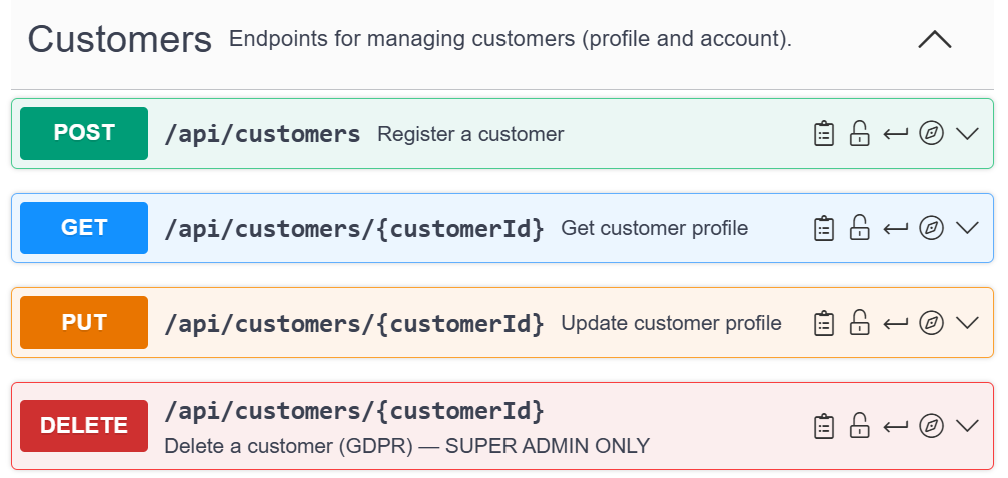
* **Books**

Endpoints related to book browsing and management



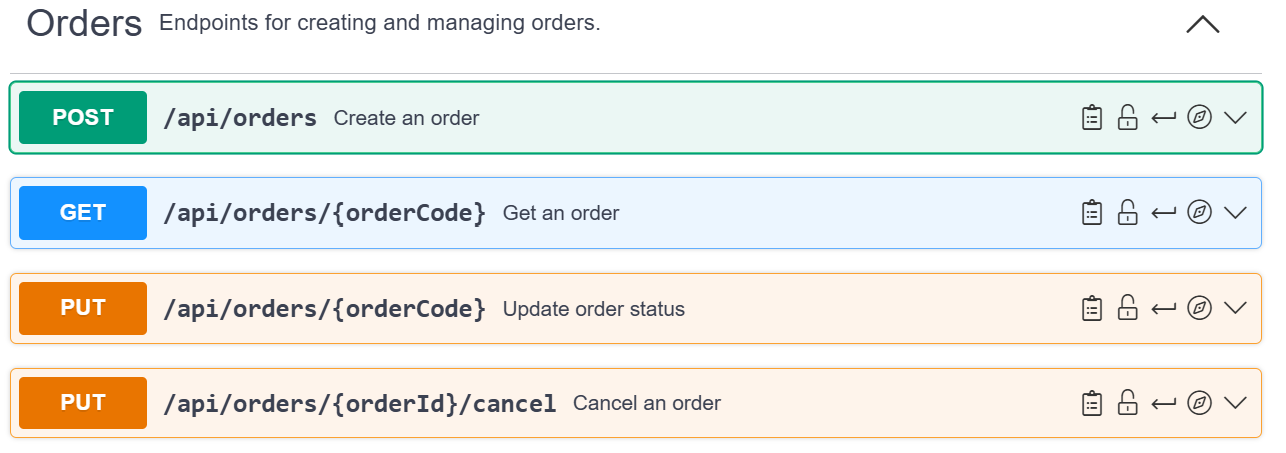
* **Customers**

Endpoints for managing customers (customer profile and account)

****

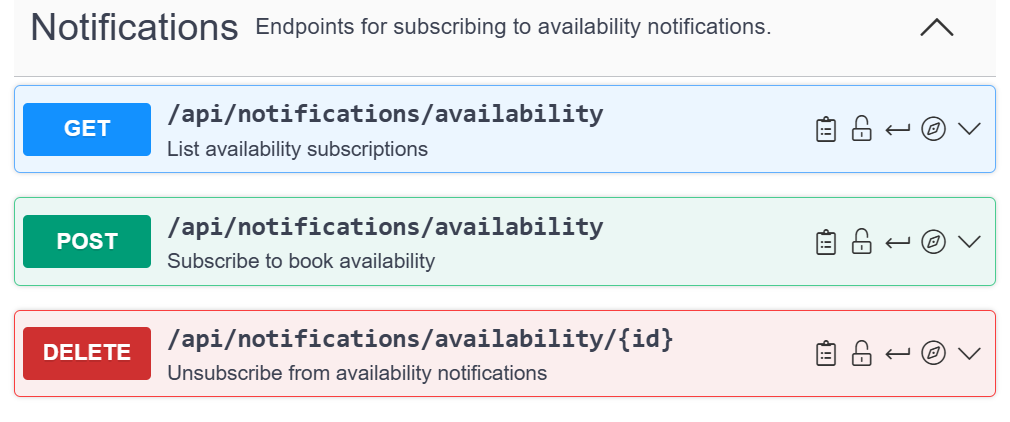
* **Orders**

Endpoints for creating and managing orders (link between books and customers)



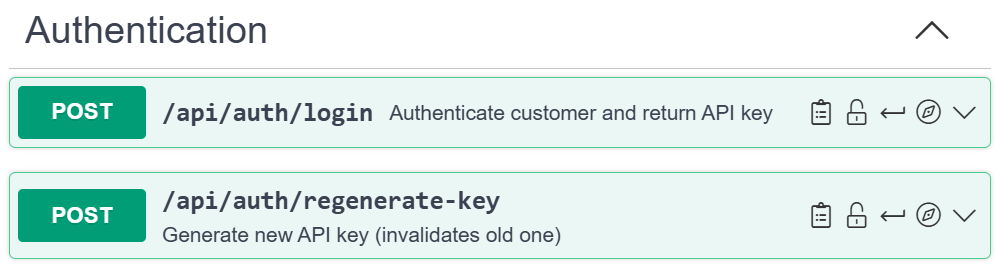
* **Notifications**

Endpoints for sending notifications to the customer. For this first version, it concerts book availability notifications.

****

* **Authentication**

Endpoints to authenticate customers and return API key

****

You can find more details about this endpoints directly on SwaggerHub : <https://app.swaggerhub.com/apis/paulinechassigneubnp/readwithpauline-bookstore-api/4.1.4>

**API Versionning :**

* Current Version : v4.1.4
* Depreciation policy : Deprecated endpoints remain functional for 6 months

# 2/ Authentication and Authorization

## 2.1 Authentication Method

The API uses an **API key authentication for this MVP** version. Future versions will migrate to JSON Web Token authentication for enhanced security and user session management.

**API Key Format**

* **Customer keys**: cus\_ prefix followed by 32 random characters (e.g., cus\_a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6)
* **Admin keys**: adm\_ prefix + 32 characters
* **Super Admin keys**: sup\_ prefix + 32 characters

**API Key Lifecycle (customer key)**

1. **Registration** (POST/api/customers) : customers provide email, password and required fields. System creates a customer account with UUID. System automatically generates and returns an API key
2. **Login** (POST/api/auth/login) : customers provides email + password, system validates credentials and returns existing API key
3. **Key data Management** : Keys expire after **90 days of inactivity**. Customers can regenerate keys via POST/api/auth/regenerate-key

**For Admins**

Admin and Super Admin keys are created **manually** by system administrators. **No expiration** (revoked manually only)

All API requests (except registration and login) must include : **Authorization: ApiKey YOUR\_API\_KEY**

## 2.2 Permissions Matrix

Three roles are supported: Customer, Admin, Super Admin. Check the user’s **permissions** according to the table below.

* **Permissions for Customers Key (cus\_)**

| **Endpoint / Action** | **Access** |
| --- | --- |
| GET/api/books/\* | **OK** |
| POST/api/customers (registration only) | **OK** |
| GET/api/customers/{own\_id} | **OK** |
| PUT/api/customers/{own\_id} | **OK** |
| POST/api/orders | **OK** |
| GET/api/orders/{own\_orders} | **OK** |
| PUT/api/orders/{own\_orders}/cancel | **OK** |
| POST/api/notifications/availability | **OK** |
| GET /api/notifications/availability/{own\_notifications} | **OK** |
| DELETE/api/notification/availability/{own\_notifications} | **OK** |
| GET/api/notifications/availability *(view all notifications)* | **KO** |
| DELETE/api/customers/\* | **KO** |
| POST/api/books | **KO** |
| Access other customers’data | **KO** |

* **Permissions for Admin Key (adm\_)**

| **Endpoint / Action** | **Access** |
| --- | --- |
| All Customer Key permissions | **OK** |
| POST/api/books | **OK** |
| PUT/api/books/{book\_id} | **OK** |
| DELETE/api/books/{book\_id} | **OK** |
| GET/api/customers/\* (view all) | **OK** |
| GET/api/orders/\* (view all) | **OK** |
| PUT/api/orders/{order\_id} | **OK** |
| GET /api/notifications/availability (view all) | **OK** |
| DELETE/api/notification/availability/{notification\_id} (all) | **OK** |
| DELETE/api/customers/\* | **KO** |

* **Permissions for Super Admin Key (sup\_)**

| **Endpoint / Action** | **Access** |
| --- | --- |
| All Admin Key permissions | **OK** |
| DELETE/api/customers/[customer\_id} | **OK** |
| System operations (config, maintenance,...) | **OK** |

## 

## 2.3 Security and audit

**HTTPS Enforcement** : All HTTP request automatically rejected with HTTP error 403 Forbidden

**Key storage** : API keys stored hashed in DB (using bcrypt)

**Payment Security** : We don’t store credit card information. All payment are processed through certified payment gateway

**Audit Logging** : For GDPR compliance all actions using customer data are logged with this required information :

* Timestamp
* API key used (hashed)
* Endpoint accessed
* Resource ID
* Response status

Logs retained for **90 days**, then **anonymized.**

# 3/ Field Mapping

This section describes how each major API resource (Book, Customer, Order, Notification) maps to the database tables and columns with an example for each to better visualize the expected result.

* **Books**

| **API Field** | **DB Table** | **DB Column** | **Rules/Notes** | ***Example*** |
| --- | --- | --- | --- | --- |
| id | books | books\_id | **Primary Key**  **INT**  **AUTO\_INCREMENT Unique book identifier, auto-incremented by the database**  Required field | *“1”* |
| title | books | title | **text**  Required field | *Le Comte de Monte-Cristo* |
| author | books | author | **text**  Required field | *Alexandre Dumas* |
| price | books | price | **DECIMAL (10,2)**  > 0, max 2 decimals  United price in store currency (**EUR**)  Required field | 20,91 EUR |
| description | books | description | **text** | A timeless adventure novel by Alexandre Dumas, first published in 1844 |
| category | books | category | **VARCHAR (100)**  Required field | *Classic Literature* |
| stock | books | stock\_quantity | **INT (integer)**  Units available in stock  Required field | *9* |
| ISBN | books | isbn | **VARCHAR (13)**  International Standard Book Number  Exactly 13 characters  Required field | *9782072895640* |
| EAN | books | ean | **VARCHAR (13)**  European Article Number  Exactly 13 characters  Required field | *9782072895640* |
| publicationDate | books | publication\_date | **DATE** | *2020-09-15* |
| publisher | books | publisher\_name | **VARCHAR(255)** | *Éditions Gallimard* |
| summary | books | summary | **text** | *An epic tale of revenge and redemption following Edmond Dantès' transformation into the Count of Monte Cristo.* |
| rating | books\_reviews | average\_rating | **FLOAT**  Min : 0  Max : 5  Computed average from customer review | *4,8*  *Average customer rating out of 5* |
| shareLinks | - | generated dynamically | **Not stored, built via URL templates**  URLs to share the book via web, email, and social platforms (X, Facebook, Instagram, TikTok) | *'*[*https://readwithpauline.com/books/le-comte-de-monte-cristo*](https://readwithpauline.com/books/le-comte-de-monte-cristo)*'*  *@ : 'mailto:?subject=Check%20this%20book!&body=*[*https://readwithpauline.com/books/le-comte-de-monte-cristo*](https://readwithpauline.com/books/le-comte-de-monte-cristo)*'*  *or*  *Facebook : '*[*https://www.facebook.com/sharer/sharer.php?u=https://readwithpauline.com/books/le-comte-de-monte-cristo*](https://www.facebook.com/sharer/sharer.php?u=https://readwithpauline.com/books/le-comte-de-monte-cristo)*'* |

* **Customers**

| **API Field** | **Database Table** | **DB Column** | **Notes** | ***Example*** |
| --- | --- | --- | --- | --- |
| customerId | customers | customer\_id | **Primary Key** (unique customer ID)  Unique customer identifier (**UUID**)  Automatically generated by the database  Required field | *550e8400-e29b-41d4-a716-446655440000* |
| firstName | customers | first\_name | **VARCHAR(100)**  Required field | *PAULINE* |
| lastName | customers | last\_name | **VARCHAR(100)**  Required field | *CHASSIGNEU* |
| email | customers | email | Must be **valid format + unique**  Required field | *pauline.chassigneu@gmail.com* |
| phoneNumber | customers | phone\_numer | **VARCHAR(20)**  Customer's phone number in international E.164 format. Must start with '+' followed by country code and digits only | *+33616382531* |
| password | customers | password\_hash | **Encrypted using bcrypt**  Password must be at least 12 characters long and include:  - at least one uppercase letter (A–Z)  - one lowercase letter (a–z)  - one number (0–9)  - one special character (!@#$%^&\*,...) | *P@ulineBTW20* |
| createdAt | customers | created\_at | **date-time**  Account creation timestamp.  **ISO 8601 timestamp in UTC**  YYYY-MM-DDTHH:mm:ssZ  **YYYY** = 4-digit year  **MM** = Month (01–12)  **DD** = Day of the month (01-31)  **T** = Separator between date and time  **HH** = Hour (00–23) in 24-hour format  **mm** = Minutes (00–59)  **ss** = Seconds (00–59)  **Z** =Indicates UTC (Coordinated Universal Time)  Required field | *2025-10-18T15:45:30Z* |
| adress.streetNumber | customers | customers.street\_number | **VARCHAR (10)**  Street number. May include one trailing letter. Must contain only digits followed optionally by one letter (no spaces)  **Related via customer\_id**  Required field | *35B* |
| adress.streetName | customers | customers.street\_name | **VARCHAR (100)**  Street name street, supports accented letters, hyphens, and spaces.  Length between 2 and 100 characters.  **Related via customer\_id**  Required field | *Avenue Alphonse XIII* |
| adress.postalCode | customers | customers.postal\_code | **VARCHAR (10)**  Postal or ZIP code. Accepts letters, digits, and hyphens. Compatible with international formats  **Related via customer\_id**  Required field | *1180* |
| adress.state | customers | customers.state | **VARCHAR (100**)  State, province, or administrative region (optional)  **Related via customer\_id** | *Brussels-Capital* |
| adress.city | customers | customers.city | **VARCHAR (100)**  City or municipality name.  Supports accented or hyphenated characters  **Related via customer\_id**  Required field | *UCCLE* |
| adress.country | customer\_adresscustomers | customers.country | **CHAR(2)**  **ISO 3166-1 alpha-2** code Only the following country codes are authorized : FR → France or  BE → Belgium  **Related via customer\_id**  Required field | BE (for Belgium) or FR (for French) |

* **Orders**

| **API Field** | **Database Table** | **DB Column** | **Notes** | ***Example*** |
| --- | --- | --- | --- | --- |
| orderId | orders | order\_id | **Primary Key** (unique order ID)  Required field  **INT**  Internal database identifier (auto-incremented by the database)  **Never exposed in API responses**  Required field | *12345* |
| orderCode | orders | order\_code | **VARCHAR (20) Automatically generated by the DB**  Format : ORD-YYMMDD-XXXX  Where :  -ORD = fixed prefix  -YYMMDD = current date  -XXXX : 4 digit sequential number for that day  **This is what customers see**  Required field | *ORD-251018-0001*  *for a first order on Oct 18, 2025* |
| orderDate | orders | order\_date | **date-time**  Order date in timestamp.  **ISO 8601 timestamp in UTC**  Required field | *2025-10-18T12:09:00Z* |
| status | orders | status | **ENUM** (‘processing’, ‘shipped’, ‘delivered’, ‘canceled”)  Required field | *Processing* |
| items | order\_items | (book\_id, quantity, unit\_price) | **List of ordered books.**  **1–n relation with orders**  Required field | *bookId: 1 title: Le Comte de Monte-Cristo*  *quantity: 2*  *price: 20,91 EUR* |
| totalAmount | orders | total\_amount | **DECIMAL (10,2)**  Total amount of orders in store currency (EUR)  Required field | *41,82 EUR*  *(20,91\*2)* |
| shippingAddress | order\_shipping | same fields as customer\_adresses | Shipping address used for this specific order (copied at the time of purchase).  **Duplicated from the address data from customer\_addresses into order\_shipping.**  Required field | *35b, avenue Alphonse XIII*  *1180 UCCLE*  *BE* |
| paymentMethod | orders | payment\_method | **ENUM**(‘card’, ‘paypal’, ‘apple\_pay’, ‘bancontact’)  Required field | *CARD* |
| deliveryMethod | orders | delivery\_method | **ENUM**(‘standard’,’same-day’,’express’)  Required field | *standard* |
| customerId | orders | customer\_id | **Foreign Key (FK) to customers**  Customer who placed the order  Required field | *550e8400-e29b-41d4-a716-446655440000* |

* **Notifications**

| **API Field** | **Database Table** | **DB Column** | **Notes** | ***Example*** |
| --- | --- | --- | --- | --- |
| id | notifications | notifications\_id | **Primary Key** (unique notification ID)  **INT**  Internal numeric order ID (auto-incremented by the database)  Required field | *1* |
| bookId | notifications | book\_id | **Foreign Key** (FK) **to books**  Book being tracked  Required field | *1* |
| customerId | notifications | customer\_id | **Foreign Key** (FK) **to customers**  Customer subscribed to the notification  Required field | *550e8400-e29b-41d4-a716-446655440000* |
| status | notifications | status | **ENUM** (‘pending’,’sent’,’failed’)  Required field | *pending* |
| createdAt | notifications | created\_at | **date-time**  Date and time when the notification was created in **ISO 8601 format** (UTC).  Required field | *2025-10-18T12:09:00Z* |

# 4/ Conversions

This section describes all data transformations that occur between the API layer and the database during processing. These conversions ensure data consistency, compliance with standards, and proper formatting for user-facing APIs.

* **Customer Data**

| **API Field** | **Conversion** | **Reason** |
| --- | --- | --- |
| customerId | Generated as a **UUID** in the database when a new customer is created | Ensures global uniqueness ID. |
| firstName, lastName | **Automatically capitalise** before storage  Pauline => PAULINE  Chassigneu => CHASSIGNEU | Consistency in display and search operations |
| phoneNumber | Converted to **E.164 format** (+33616382531 for a french number) | Standardized international phone numbers |
| password | **Hashed using bcrypt** before being stored in password\_hash | Security  Passwords must never stored in the DB |
| createdAt | Converted to an **ISO 8601 timestamp**  (2025-10-18T15:45:30Z) | Ensures consistent UTC storage for all time zones |

* **Address Data**

| **API Field** | **Conversion** | **Reason** |
| --- | --- | --- |
| address.country | Converted to **uppercase ISO 3166-1 alpha-2 code** ("be" → "BE").  **Only the following country codes are authorized :**   * FR → France * BE → Belgium | Ensures uniform international representation  The API must validate that customer addresses are only accepted for **France (FR)** and **Belgium (BE)** |
| adress.postalCode and address.country | Each country has a specific pattern:  -France (FR) : 5 digits → ^[0-9]{5}$  -Belgium (BE) : 4 digits → ^[0-9]{4}$  **The postal code must match with the country’s pattern** | Ensures coherences between postal code and city according to the country’s postal system |

* **Book data**

| **API Field** | **Conversion** | **Reason** |
| --- | --- | --- |
| bookId | **Auto-incremented integer** generated by the database  Returned as integer from books\_id | Simplifies indexing and internal reference |
| price | Rounded to **two decimals using store currency** (EUR) | Consistent monetary formatting. |
| rating | **Computed dynamically as the average** of all user reviews  Calculated as AVG(books\_reviews.rating) | Ensures accuracy |
| sharedLinks | **Dynamically generated URLs, not stored in DB** | Created at runtime for social sharing |
| stock | **Retrieved from books.stock\_quantity** | Direct maping |

* **Order data**

| **API Field** | **Conversion** | **Reason** |
| --- | --- | --- |
| orderId | Auto-incremented integer generated by the database  **Never exposed in API responses** | Provides unique sequential order tracking |
| orderCode | **Automatically generated by the DB**  Format : ORD-YYMMDD-XXXX  Where :  -ORD = fixed prefix  -YYMMDD = current date  -XXXX : 4 digit sequential number for that day  **This is what customers see** | Provides a human-readable order reference (improves user experience) |
| orderDate | Converted to an ISO 8601 timestamp  (2025-10-18T15:45:30Z) | Ensures consistent UTC storage for all time zones |
| totalAmount | Calculated as the sum of (quantity \* price) from order\_items | Prevents mismatch between stored and displayed totals (the Total Amount is always calculated and saved by the backend. |
| shippingAddress | Same fields as customer\_addresses  When an order is placed, **you must copy the address data from customer\_addresses into order\_shipping.** | Duplicated for order history consistency, ensuring the address reflects the state at purchase time, even if the customer updates their profile later. |

* **Notification data**

| **API Field** | **Conversion** | **Reason** |
| --- | --- | --- |
| createdAt | Converted to an **ISO 8601 timestamp**  (2025-10-18T15:45:30Z) | Ensures consistent UTC storage for all time zones |
| bookId / customerId | **Foreign key references validated** before insertion | Guarantees data integrity |
| status | Default to “pending” on creation | Initial state for new notifications |

* **Sending Notifications (Background Job)**

The job runs **every 5 minutes**, retrieves notifications with **status = 'pending'** and sends the emails. Then updates **status = ‘sent”.**

# 5/ Error handling

The ReadWithPauline API uses **standard HTTP status codes** and **returns structured JSON error** responses to facilitate integration and debugging.

## 5.1 Standard Error Structure

All errors follow this consistent JSON format:

****

Below the definition of the different fields**:**

* **timestamp**: Error timestamp in ISO 8601 UTC format
* **status**: the HTTP status code returned (e.g., 400, 404,...)
* **error**: the standard HTTP error phrase associated with the code (e.g., bad request or unauthorized)
* **code**: Unique error identifier for programmatic handling (e.g.,’VALIDATION\_ERROR’)
* **message**: Clear error message in comprehensive language
* **path**: API endpoint where the error occurred
* **details**: Additional context or resolution suggestions (optional) like the name of the field that failed validation or a short description of the validation problem for that field.

## 5.2 HTTP status code used

| **Status** | **Error** | **Meaning** | **When it Occurs** |
| --- | --- | --- | --- |
| 200 | OK | Request was successful | Standard response for successful GET, POST, PUT and DELETE |
| 201 | Created | Ressource successfully created | For example; when a new book, customer or order is added |
| 400 | Bad Request | Client sent invalid data or missing fields | Missing parameters, incorrect format, validation failed (invalid email, missing address,...) |
| 401 | Unauthorized | Missing or invalid API key | Authentication error in protected endpoints |
| 402 | Payment Required | Payment gateway declined the transaction | Transaction declined by payment processor |
| 403 | Forbidden | Valid API key but insufficient permissions | Trying to access another user’s data |
| 404 | Not Found | Requested resource doesn’t exist | Book ID, order ID, customer ID or notification ID not found |
| 409 | Conflict | Ressource already exists or violates uniqueness constraints | Email already in use, insufficient stock |
| 422 | Unprocessable Entity | Data is syntactically correct but semantically invalid | invalid ISBN format or negative quantity |
| 429 | Too many requests | Too many requests within allowed time window | API client making too many calls too quickly (for example, 200 requests per minute) or Brute force attempts (same endpoints called repeatedly) |
| 500 | Internal Server Error | Unexpected server-side failure | System or database error |
| 503 | Service Unavailable | External dependency or database temporarily unavailable | Temporary maintenance or overload or when the payment gateway is down |

## 5.3 Error codes by Category

* **Authentification Errors**

| **Code** | **HTTP Status** | **Description** | ***Example*** |
| --- | --- | --- | --- |
| AUTH\_MISSING\_API\_KEY | Code : 401  Error : Unauthorized | API key missing from Authorization header | *Missing Authorization header* |
| AUTH\_INVALID\_API\_KEY | Code : 401  Error : Unauthorized | Invalid or expired API key | *The client used an incorrect or revoked API key.* |
| AUTH\_INSUFFICIENT\_PERMISSIONS | Code : 403  Error : Forbidden | Valid API key but insufficient rights | *A basic user API key attempts to access admin-only endpoints.* |

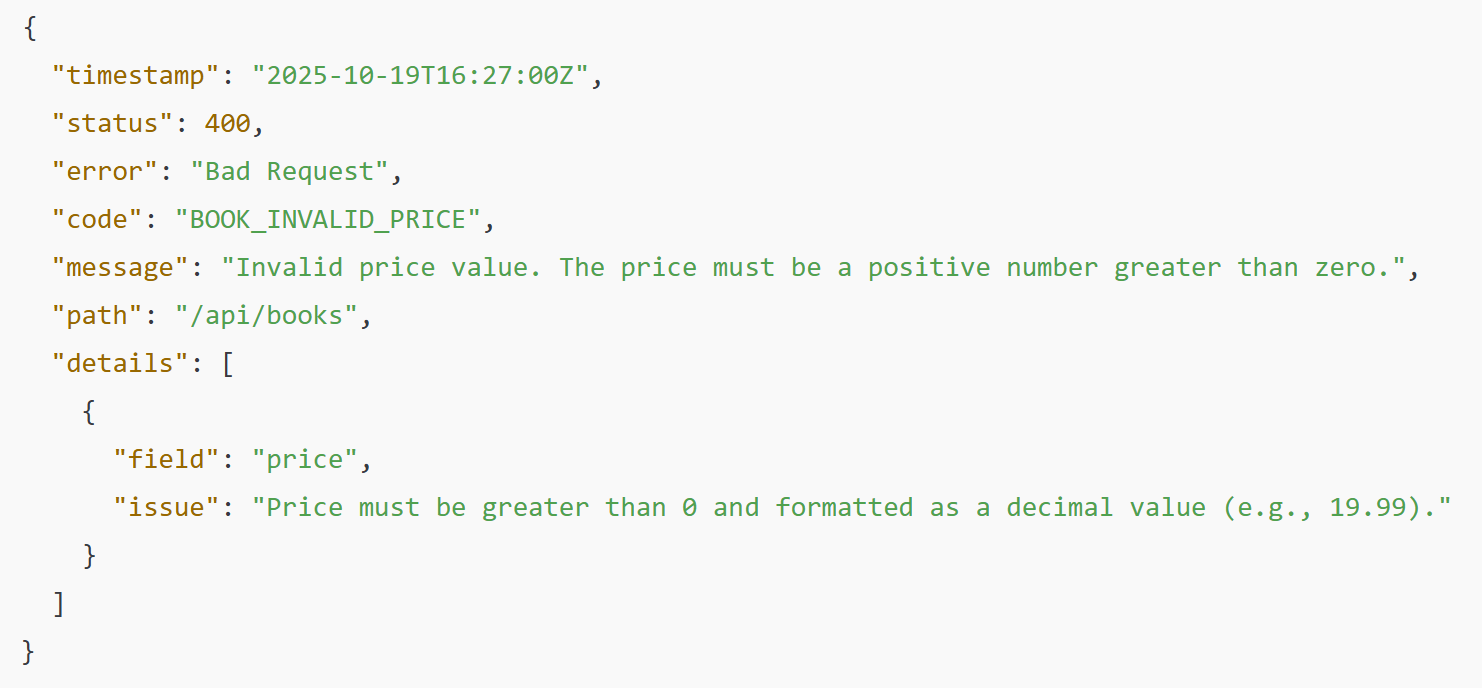
*Example of JSON response :*

****

* **Validation Errors (Books)**

| **Code** | **HTTP Status** | **Description** | ***Example*** |
| --- | --- | --- | --- |
| BOOK\_NOT\_FOUND | Code : 404  Error : Not found | Book ID doesn’t exist | *GET/books/99999* |
| BOOK\_INVALID\_ISBN | Code : 400  Error : Bad request | Invalid ISBN format | *ISBN must contain 13 characters* |
| BOOK\_INVALID\_EAN | Code : 400  Error : Bad request | Invalid EAN format | *EAN must contain 13 characters* |
| BOOK\_INVALID\_PRICE | Code : 400  Error : Bad request | Negative or incorrect price format | *Price must be > 0* |
| BOOK\_OUT\_OF\_STOCK | Code : 409  Error : Conflict | Insufficient stock for requested quantity | *Available stock : 2*  *Requested : 5* |

*Example of JSON response :*

**

* **Validation Errors (customers)**

| **Code** | **HTTP Status** | **Description** | ***Example*** |
| --- | --- | --- | --- |
| CUSTOMER\_NOT\_FOUND | Code : 404  Error : Not found | Customer ID doesn’t exist  Invalid customers UUID | *GET/customers/99999* |
| CUSTOMER\_EMAIL\_EXISTS | Code : 409  Error : Conflicts | Email already registered | *Email is already used by another account* |
| CUSTOMER\_INVALID\_EMAIL | Code : 400  Error : Bad request | Invalid Email format | *Email must follow standard format* |
| CUSTOMER\_INVALID\_PHONE | Code : 400  Error : Bad request | Invalid Phone number | *Phone number must be in E.164 format (+33…)* |
| CUSTOMER\_WEAK\_PASSWORD | Code : 400  Error : Bad request | Password too weak | *Minimum 12 characters with uppercase, lowercase, digit and special character* |
| CUSTOMER\_INVALID\_POSTAL\_CODE | Code : 400  Error : Bad request | Invalid postal code  Incorrect format for specified country | *Postal code must match the country format* |
| CUSTOMER\_INVALID\_COUNTRY | Code : 400  Error : Bad request | Invalid country code  Only addresses in France, Belgium are supported | *Must be ISO 3166-1 alpha-2 code.*  *Only values “BE” and “FR” are supported* |

*Example of JSON response :*

**

* **Validation Errors (orders)**

| **Code** | **HTTP Status** | **Description** | ***Example*** |
| --- | --- | --- | --- |
| ORDER\_NOT\_FOUND | Code : 404  Error : Not found | Customer ID doesn’t exist  Invalid customers UUID | *GET/orders/99999* |
| ORDER\_INVALID\_STATUS | Code : 400  Error : Bad request | Invalid order status | *Status must be : processing, shipped, delivered or cancelled* |
| ORDER\_EMPTY\_ITEMS | Code : 400  Error : Bad request | Empty order | *Order must contain at least one item* |
| ORDER\_INVALID\_QUANTITY | Code : 400  Error : Bad request | Invalid quantity | *Quantity must be >=1* |
| ORDER\_PAYMENT\_FAILED | Code : 402  Error : Payment gateway declined | Payment failure | *Transaction declined by payment processor* |
| ORDER\_CANNOT\_CANCEL | Code : 409  Error : Conflicts | Order cannot be cancelled | *Delivered orders cannot be cancelled* |

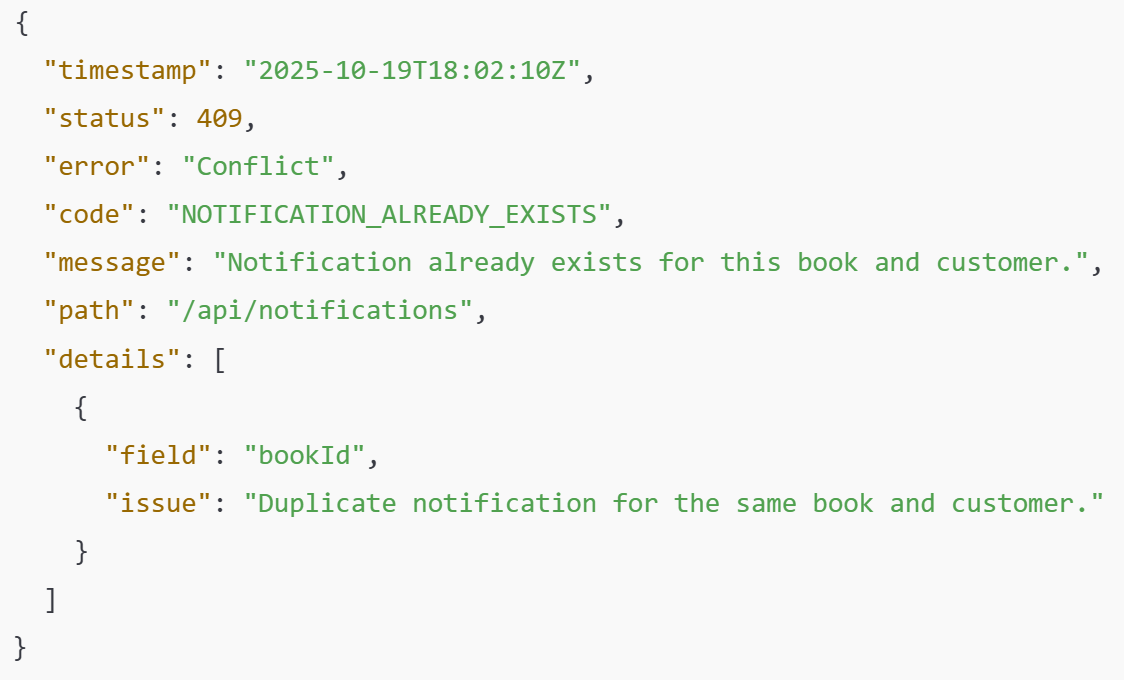
*Example of JSON response :*

**

* **Validation Errors (notifications)**

| **Code** | **HTTP Status** | **Description** | ***Example*** |
| --- | --- | --- | --- |
| NOTIFICATION\_NOT\_FOUND | Code : 404  Error : Not found | Notification doesn’t exist | *GET/notifications/availability/99999*  *Invalid notification ID* |
| NOTIFICATION\_ALREADY\_EXISTS | Code : 409  Error : Conflicts | Notification already created | *Customer is already subscribed to notifications for this book* |
| NOTIFICATION\_BOOK\_AVAILABLE | Code : 422  Error : Unprocessable Entity | Book already available | *Cannot create notification for in-stock book* |

*Example of JSON response :*

****

General information : When multiple fields are invalid, the API returns a detailed list of errors.

# 6/ Acceptance criteria

This section defines the acceptance criteria for the ReadWithPauline API, including what constitutes a successful request, expected response formats, and how the API handles edge cases and errors.

## 6**.1 General acceptance criteria**

**All API requests must :**

1. **Include valid authentication**
   1. Every request must contain a valid API key in the Authorization header
   2. Format: Authorization: ApiKey YOUR\_API\_KEY
   3. Invalid or missing API keys result in 401 Unauthorized
2. **Use HTTPS protocol**
   1. Every request method (GET, POST, PUT, DELETE) matches the intended operation
   2. All requests must use HTTPS for secure communication
   3. HTTP requests are automatically rejected with 403 Forbidden
3. **Send valid JSON response**
   1. Content-Type header must be application/json
   2. Request body must be valid JSON syntax using structured error responses. (e.g. 4.1 Standard Error Structure)
   3. Malformed JSON returns 400 Bad Request
4. **Include required fields**
   1. All required fields must be present in the request
   2. Missing required fields return 400 Bad Request with field-specific error messages
5. **Respect data type constraints**
   1. Fields must match their expected data types (string, integer, etc.)
   2. Type mismatches return 400 Bad Request
6. **Validation passes for constraints such as:**
   1. ISBN/EAN length = 13 characters
   2. Valid country codes (FR, BE)
   3. Postal code matches the country format
   4. Password complexity rules (≥12 chars, uppercase, lowercase, digit, special character)

The backend processes the request without integrity or constraint violations :

| **Endpoint type** | **Expected HTTP Code** | **Description** |
| --- | --- | --- |
| GET (retrieve data) | 200 OK | Data successfully retrieved |
| POST (create data) | 201 Created | Ressource successfully created |
| PUT (update data) | 200 OK | Ressource successfully updated |
| DELETE (remove data) | 200 OK | Ressource successfully deleted |

## 6.2 Performance and reliability criteria

* **Average response time:**

| **Endpoint type** | **Max Response time** | **Acceptance** |
| --- | --- | --- |
| GET (single resource) | < 200ms | OK |
| POST (list) | <500ms | OK |
| POST/PUT | <1000ms | OK |
| DELETE (remove data) | <500ms | OK |

* Maximum concurrent requests per client: **200 / minute**
* Data integrity: **IDs and references** (bookId, customerId, orderId, notificationId) **must always exist before insertion**
* **All timestamps stored in UTC** to ensure global consistency

# 7/ Test case scenarios

These test scenarios ensure that the ReadWithPauline API behaves correctly in normal, invalid, and boundary situations.

## 7.1 General

| **Test ID** | **Scenarios** | **Endpoint** | **Expected results** |
| --- | --- | --- | --- |
| G-01 | Missing API key | any endpoint | Returns 401 Unauthorized  AUTH\_MISSING\_API\_KEY |
| G-02 | Invalid API Key | any endpoint | Returns 401 Unauthorized  AUTH\_INVALID\_API\_KEY |
| G-03 | >200 requests per minute | any endpoint | Returns 429 Too Many Requests |

## 7.2 Books

**Pre conditions : All requests use HTTPS + JSON, authenticated via an API key in the header.**

| **Test ID** | **Scenarios** | **Endpoint** | **Expected results** |
| --- | --- | --- | --- |
| B-01 | Get the list of all books in the DB  *Pre-conditions : At least two books are on the DB (including a book with title contain “Monte-Christo”)* | GET/api/books (List all books) | -Returns HTTP (200 OK)  -Each book includes required fields : : id, title, author, price, category, stock, ISBN, EAN  -Response contains an array of book objects  -Books are sorted by default (e.g., by id ascending) |
| B-02 | Search for a book by title  *Pre-conditions : At least two books are on the DB (including a book with title contain “Monte-Christo”)* | GET/api/books?title=Monte-Cristo | -Returns HTTP (200 OK) with list of matching book with required fields  -Books are sorted by default (e.g., by id ascending) |
| B-03 | Retrieve a book by ID | GET/api/books/1 | -Returns HTTP (200 OK) with complete book details  -shareLinks are dynamically generated (web, email, Facebook, Instagram, X, TikTok) |
| B-04 | Book not found | GET/api/books/9999 | Returns 404 Not Found when book ID doesn't exist |
| B-05 | No books in database | GET/api/books (List all books) | Returns 200 OK with empty array |
| B-06 | Invalid ID format | GET/api/books/-7 | Returns 400 Bad Request |
| B-07 | Book out of stock  Retrieve a book with stock = 0 | GET/api/books/{bookId} | Returns 200 OK but stock: 0 |
| B-08 | Create a new book on the DB with this fields :  -title (string, required)  -author (string, required)  -price (decimal, required, > 0)  -category (string, required)  -stock (integer, required, >= 0)  -ISBN (string, 13 chars, required)  -EAN (string, 13 chars, required) | POST/api/books (Create new book) | -Returns HTTP 201 Created on successful creation  -Response includes new book URL  -bookId is auto-generated by the database  - Price is rounded to 2 decimal places  -Stock must be >= 0 |
| B-09 | Duplicate this fields to create a new book in the DB | POST/api/books (Create new book) | -Duplicate ISBN → 409 Conflict - "Book with this ISBN already exists"  -Duplicate EAN → 409 Conflict - "Book with this EAN already exists" |
| B-10 | Create a new book with a negative price | POST/api/books (Create new book) | Returns 400 BOOK\_INVALID\_PRICE - Price must be > 0 |
| B-11 | Create a new book with a negative stock | POST/api/books (Create new book) | Returns 400 Bad Request - "Stock cannot be negative" |
| B-12 | Create a new book without a category | POST/api/books (Create new book) | Returns 400 bad request  Field category is required to create a new book |

## 7.3 Customers

**Pre conditions : All requests use HTTPS + JSON, authenticated via an API key in the header.**

| **Test ID** | **Scenarios** | **Endpoint** | **Expected results** |
| --- | --- | --- | --- |
| C-01 | Create a new customer on the DB with this fields:  -firstName (string, required)  -lastName (string, required)  -email (string, required, unique, valid format)  -phoneNumber (string, required, E.164 format)  -password (string, required, min 12 chars with complexity)  -address.streetNumber (string, required)  -address.streetName (string, required, 2-100 chars)  -address.postalCode (string, required, country-specific format)  -address.city (string, required)  -address.country (string, required, only FR or BE) | POST/api/customers (create a new customer) | -Returns HTTP 201 Created on successful registration  -customerId (UUID) is auto-generated  -firstName and lastName are automatically capitalized  -phoneNumber is validated and converted to E.164 format  -password is hashed using bcrypt before storage  -createdAt timestamp is set to current UTC time  -Address country is limited to FR or BE |
| C-02 | Duplicate this fields to create a new customer in the DB | POST/api/customers (create a new customer) | -Returns 409 CUSTOMER\_EMAIL\_EXISTS  Email already registered |
| C-03 | Create a new customer with this email : user1.com | POST/api/customers (create a new customer) | Returns 400 CUSTOMER\_INVALID\_EMAIL  Invalid Email format |
| C-04 | Create a new customer with no digit on the password | POST/api/customers (create a new customer) | Returns 400 CUSTOMER\_WEAK\_PASSWORD  Password too weak  Minimum 12 characters with uppercase, lowercase, digit and special character |
| C-05 | Create a new customer with this phone number : 0616382531 | POST/api/customers (create a new customer) | Returns 400 CUSTOMER\_INVALID\_PHONE  Phone number must be in E.164 format (+33…) |
| C-06 | Create a new customer with address.country = NL | POST/api/customers (create a new customer) | Returns 400 CUSTOMER\_INVALID\_COUNTRY  Only FR and BE are supported |
| C-07 | Create a new customer with address.country = FR  address.postalCode = 1180 | POST/api/customers (create a new customer) | Returns 400 CUSTOMER\_INVALID\_POSTAL\_CODE  Invalid postal code  Incorrect format for specified country |
| C-08 | Get customer profile | GET/api/customers/{customerId} | -Returns HTTP 200 OK when customer exists  -Password hash is never returned in response |
| C-09 | Customer not found | GET/api/customers/{customerId} | Returns 404 Not Found when customer doesn't exist |
| C-10 | Invalid UUID format | GET/api/customers/{customerId} | Returns 400 Bad Request |
| C-11 | Delete a customer account (GDPR reason)   |  | | --- | | DELETE/api/customer/{customerId}  SUPER ADMIN ONLY | -Returns 200 OK, confirmation message if the customer doesn’t have an existing order  -Returns 409 Conflict if the customer have an existing order |

## 7.4 Orders

**Pre conditions : All requests use HTTPS + JSON, authenticated via an API key in the header.**

| **Test ID** | **Scenarios** | **Endpoint** | **Expected results** |
| --- | --- | --- | --- |
| O-01 | Create a new order on the DB with this fields:  -customerId (UUID, required, must exist)  -items (array, required, min 1 item)  -status = processing  -bookId (string, required, must exist)  -quantity (integer, required, >= 1)  -paymentMethod (enum, required: card, paypal, apple\_pay, bancontact)  -deliveryMethod (enum, required: standard, same-day, express) | POST/api/orders  (Create new order) | -Returns HTTP 201 Created on successful order  -orderId is auto-generated (INT32) in the DB (not exposed on API)  -Returns orderCode in format ORD-YYMMDD-XXXX  -orderDate is set to current UTC timestamp  -totalAmount is calculated automatically from order items (sum of quantity \* unit\_price)  -Stock is decremented for each ordered book  -Shipping address is copied from customer address (snapshot at order time)  - Stock availability is checked before order confirmation |
| O-02 | Create a new order without items (Empty order) | POST/api/orders  (Create new order) | Returns 400 ORDER\_EMPTY\_ITEMS  Order must contain at least one item |
| 0-03 | Create a new order with a customerId that doesn’t exist | POST/api/orders  (Create new order) | Returns 404 CUSTOMER\_NOT\_FOUND  Customers not found |
| O-04 | Create a new order with a bookId that doesn’t exist | POST/api/orders  (Create new order) | Returns 404 BOOK\_NOT\_FOUND  Book doesn’t exist |
| O-05 | Place an order with multiple books (mixed stock availability) | POST/api/orders  (Create new order) | Partial validation; Reject with returns 400 BOOK\_OUT\_OF\_STOCK if any item unavailable |
| O-06 | Plan an order with invalid quantity (0 or negative) | POST/api/orders  (Create new order) | Returns 400 ORDER\_INVALID\_QUANTITY |
| O-07 | Payment failure | POST/api/orders  (Create new order) | Returns 422 ORDER\_PAYMENT\_FAILED |
| O-08 | Plan an order with status = new | POST/api/orders  (Create new order) | Returns 400 ORDER\_INVALID\_STATUS  Status must be : processing, shipped, delivered or cancelled |
| O-09 | Get the details of an order | GET/api/orders/{orderCode} | -Returns HTTP 200 OK when order exists  -Response includes full order details with items |
| O-10 | Order not found | GET/api/orders/{orderCode} | Returns 404 ORDER\_NOT\_FOUND |
| O-11 | A user tries to access another customer’s order details | GET/api/orders/{orderId} | Returns 403 Forbidden  Users can only access their own orders |
| O-12 | Update an order status (from processing to shipped) | PUT/api/orders/{orderId} | -Returns HTTP 200 ok when updated status returned  -Order status changes to shipped |
| O-13 | Cancel an order with the status = processing | PUT /orders/{orderId}/cancel | -Returns HTTP 200 OK when cancellation succeeds  -Order status changes to cancelled  -Stock is restored for all items |
| O-14 | Cancel an order with the status = shipped | PUT /orders/{orderId}/cancel | Returns 409 ORDER\_CANNOT\_CANCEL - Shipped orders cannot be cancelled  Only orders with status processing can be cancelled |

## 7.5 Notifications

**Pre conditions : All requests use HTTPS + JSON, authenticated via an API key in the header.**

| **Test ID** | **Scenarios** | **Endpoint** | **Expected results** |
| --- | --- | --- | --- |
| N-01 | Create a new notification on the DB with this fields:  -bookId (string, required, must exist)  -customerId (UUID, required, must exist) | POST/api//notifications/availability  (Subscribe to book availability) | -Returns HTTP 201 Created on successful subscription  -Customer receives notification when book is back in stock  -createdAt is set to current UTC timestamp  -notificationId is auto-generated (INT32) |
| N-02 | Duplicate this fields to create a new notification in the DB | POST/api//notifications/availability  (Subscribe to book availability) | Returns 409 NOTIFICATION\_ALREADY\_EXISTS  Notification already created |
| N-03 | Create a new notification for a book already in stock | POST/api//notifications/availability  (Subscribe to book availability) | Returns 422 NOTIFICATION\_BOOK\_AVAILABLE  Book is currently available |
| N-04 | Create a new notification for a customer who doesn’t exist on the DB | POST/api//notifications/availability  (Subscribe to book availability) | Returns 404 CUSTOMER\_NOT\_FOUND  Customer doesn't exist |
| N-05 | Create a new notification for a book that doesn’t exist on the DB | POST/api//notifications/availability  (Subscribe to book availability) | Returns 404 BOOK\_NOT\_FOUN  Book doesn’t exist |

# 8/ SQL queries

## 

Database summary :

| **Table** | **Purpose** |
| --- | --- |
| books | Store all books in the catalog |
| customers | Store customers accounts and addresses |
| order | Store customers orders |
| order\_items | Store items in each order |
| notifications | Track availability alerts |

Relationships summary :

* **customers ↔ orders:** One customer can have many orders **(1:N)**
* **orders ↔ order\_items:** One order contains many items **(1:N)**
* **books ↔ order\_items:** One book can appear in many order items **(1:N)**
* **customers ↔ notifications:** One customer can subscribe to many book alerts **(1:N)**
* **books ↔ notifications:** One book can have many notification subscriptions **(1:N)**

Below SQL queries to create the database tables.

1. **Create Table BOOKS**

CREATE TABLE books (

books\_id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(255) NOT NULL,

author VARCHAR(255) NOT NULL,

price DECIMAL(6,2) NOT NULL,

description TEXT,

category VARCHAR(100) NOT NULL,

stock\_quantity INT NOT NULL DEFAULT 0,

isbn VARCHAR(13) NOT NULL UNIQUE,

ean VARCHAR(13) NOT NULL UNIQUE,

publication\_date DATE,

publisher\_name VARCHAR(255),

summary TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

1. **Create Table CUSTOMERS**

CREATE TABLE customers (

customer\_id VARCHAR(36) PRIMARY KEY,

first\_name VARCHAR(100) NOT NULL,

last\_name VARCHAR(100) NOT NULL,

email VARCHAR(255) NOT NULL UNIQUE,

phone\_number VARCHAR(20) NOT NULL,

password\_hash VARCHAR(255) NOT NULL,

*- Address fields (integrated)*

street\_number VARCHAR(10) NOT NULL,

street\_name VARCHAR(100) NOT NULL,

postal\_code VARCHAR(10) NOT NULL,

city VARCHAR(100) NOT NULL,

state VARCHAR(100),

country CHAR(2) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

1. **Create Table ORDERS**

CREATE TABLE orders (

order\_id INT AUTO\_INCREMENT PRIMARY KEY, *-- Internal use only*

order\_code VARCHAR(20) NOT NULL UNIQUE, *-- Exposed in API*

customer\_id VARCHAR(36) NOT NULL,

order\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

status ENUM('processing', 'shipped', 'delivered', 'cancelled') DEFAULT 'processing',

total\_amount DECIMAL(10,2) NOT NULL,

payment\_method ENUM('card', 'paypal', 'apple\_pay', 'bancontact') NOT NULL,

delivery\_method ENUM('standard', 'same-day', 'express') NOT NULL,

*-- Shipping address (snapshot at order time)*

shipping\_street\_number VARCHAR(10) NOT NULL,

shipping\_street\_name VARCHAR(100) NOT NULL,

shipping\_postal\_code VARCHAR(10) NOT NULL,

shipping\_city VARCHAR(100) NOT NULL,

shipping\_state VARCHAR(100),

shipping\_country CHAR(2) NOT NULL,

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id),

);

1. **Create Table ORDERS\_ITEMS**

CREATE TABLE order\_items (

order\_item\_id INT AUTO\_INCREMENT PRIMARY KEY,

order\_id INT NOT NULL,

book\_id INT NOT NULL,

quantity INT NOT NULL,

unit\_price DECIMAL(6,2) NOT NULL,

FOREIGN KEY (order\_id) REFERENCES orders(order\_id) ON DELETE CASCADE,

FOREIGN KEY (book\_id) REFERENCES books(books\_id)

);

1. **Create Table NOTIFICATIONS (Availability Alert)**

CREATE TABLE notifications (

notifications\_id INT AUTO\_INCREMENT PRIMARY KEY,

book\_id INT NOT NULL,

customer\_id VARCHAR(36) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

notified\_at TIMESTAMP NULL,

FOREIGN KEY (book\_id) REFERENCES books(books\_id) ON DELETE CASCADE,

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id) ON DELETE CASCADE,

*-- Prevent duplicates*

UNIQUE KEY (book\_id, customer\_id)

);

1. **Create Table BOOKS\_REVIEWS**

CREATE TABLE books\_reviews (

review\_id INT AUTO\_INCREMENT PRIMARY KEY,

book\_id INT NOT NULL,

customer\_id VARCHAR(36) NOT NULL,

rating DECIMAL(2,1) NOT NULL,

review\_text TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (book\_id) REFERENCES books(books\_id) ON DELETE CASCADE,

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id) ON DELETE CASCADE,

);

1. **INDEXES for performance**

*-- Indexes on books*

CREATE INDEX idx\_books\_category ON books(category);

CREATE INDEX idx\_books\_author ON books(author);

CREATE INDEX idx\_books\_stock ON books(stock\_quantity);

*-- Indexes on customers*

CREATE INDEX idx\_customers\_email ON customers(email);

*-- Indexes on orders*

CREATE INDEX idx\_orders\_customer ON orders(customer\_id);

CREATE INDEX idx\_orders\_status ON orders(status);

CREATE INDEX idx\_orders\_date ON orders(order\_date);

*-- Indexes on order\_items*

CREATE INDEX idx\_order\_items\_order ON order\_items(order\_id);

CREATE INDEX idx\_order\_items\_book ON order\_items(book\_id);

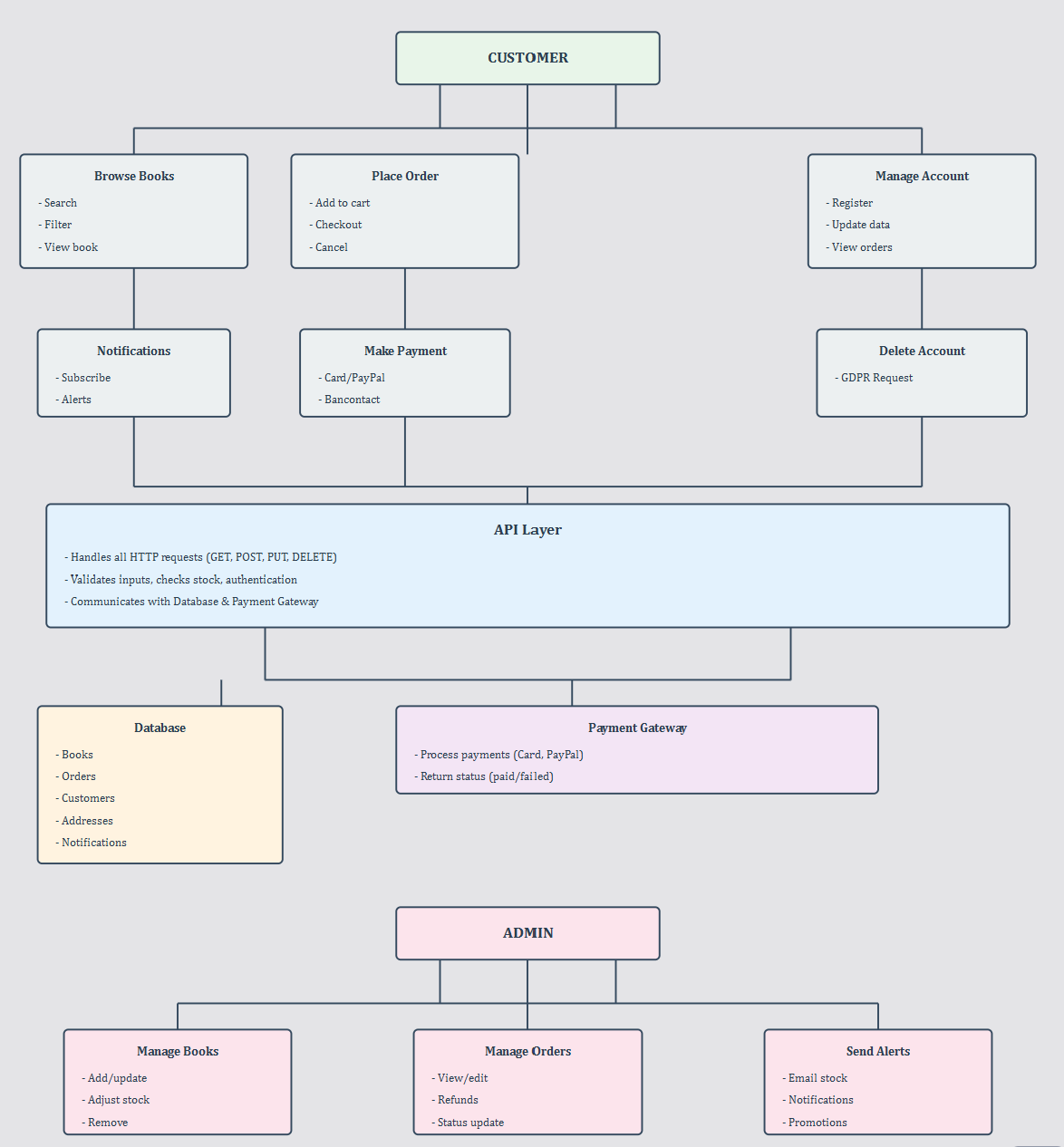
*-- Indexes on notifications*

CREATE INDEX idx\_notifications\_book ON notifications(book\_id);

CREATE INDEX idx\_notifications\_customer ON notifications(customer\_id);

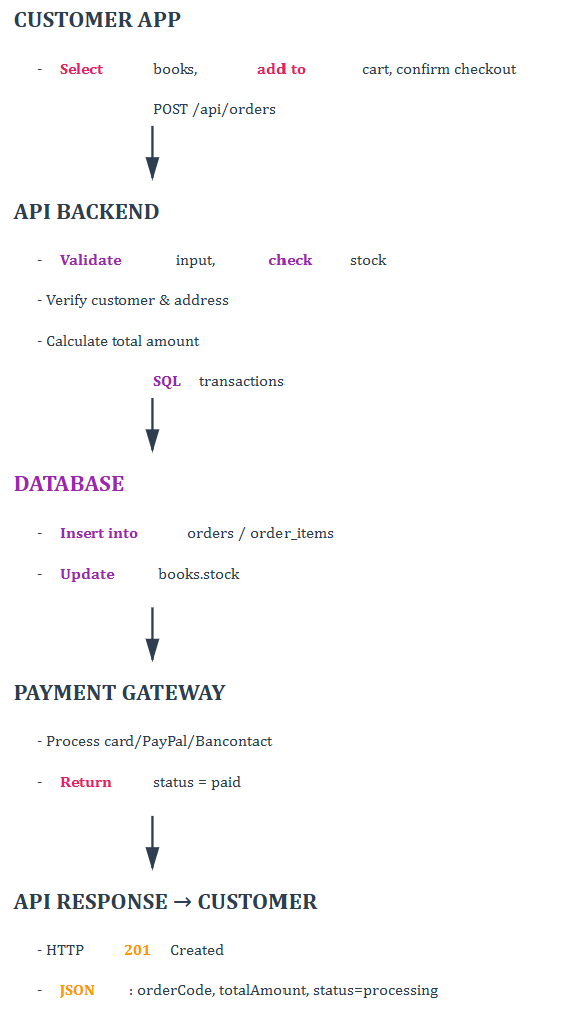
# 9/ Use Case Diagrams and Flow Diagrams

## 9.1 Use Case Diagrams



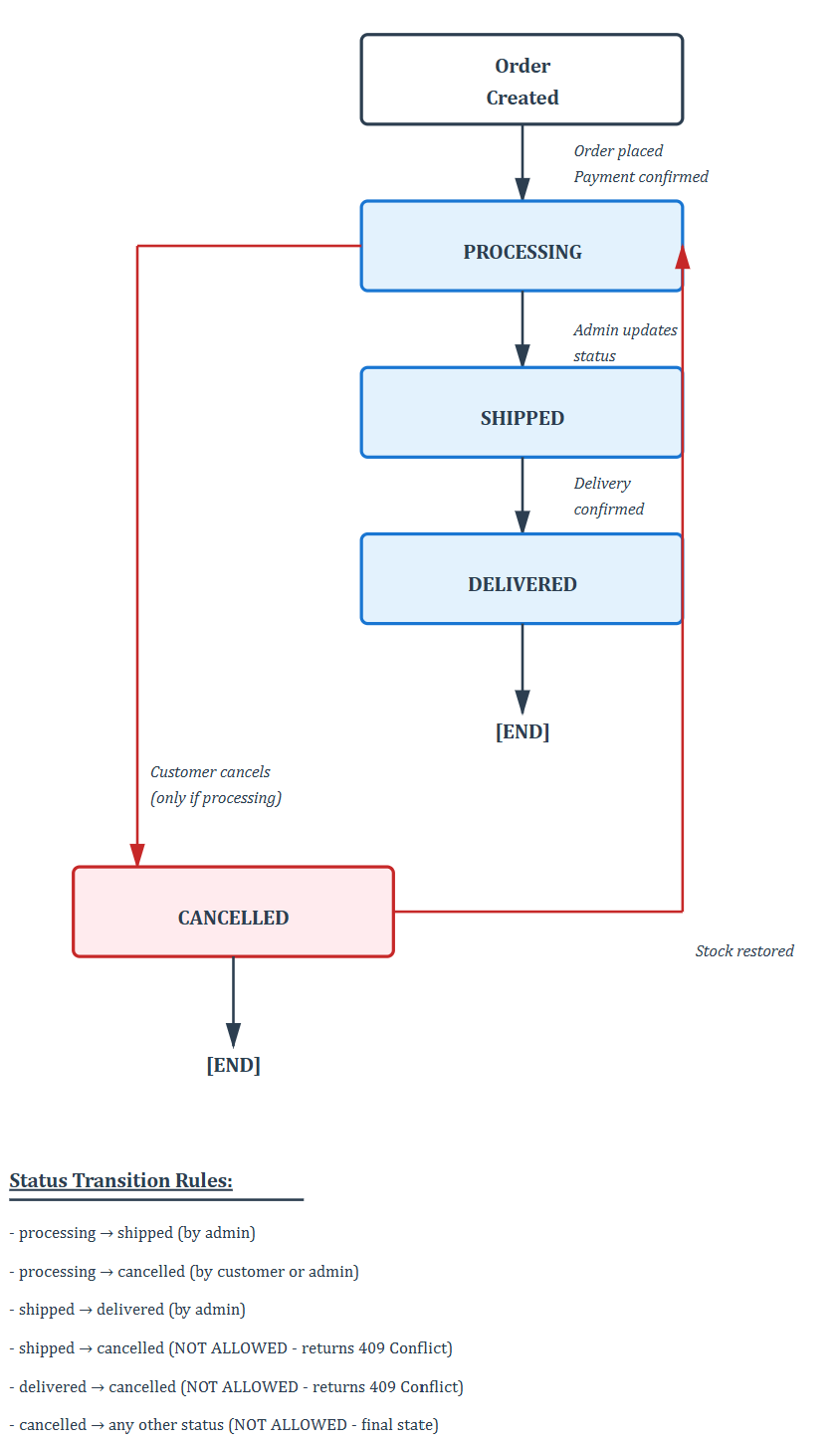
## 9.2 Flow Diagrams

Below flow diagrams “Place an order”:



The following diagram illustrates the complete lifecycle of an order within the ReadWithPauline platform.

It defines each possible order state and the valid transitions between them, ensuring consistent business logic and data integrity.



# 10/ User stories

Below 8 user stories to add to the backlog for next sprints.

**User story 1**

**Feature :** Customer Account Management

**Id :** US-01

**User story (title) :** Customer Registration

**Description :** As a customer, I want to create an account on ReadWithPauline bookstore library, so that I can browse books and place an order

Allow customers to register by providing personal information, address and creating a password. The system validates all required fields, ensures email uniqueness and enforces password strength. The system must accept only addresses in FR and BE.

**Acceptance criteria :**

* Valid registration creates account with UUID, hashed password, capitalized names (Returns 201 Created)
* Missing required fields (Returns 400 VALIDATION\_FAILED)
* Email already used (Returns 409 CUSTOMER\_EMAIL\_EXISTS) or invalid email format (Returns 400 CUSTOMER\_INVALID\_EMAIL)
* Unsupported country (Returns 400 CUSTOMER\_INVALID\_COUNTRY)
* Wrong password : < 12 characters or missing complexity (Returns 400 CUSTOMER\_WEAK\_PASSWORD)
* Postal code doesn’t match country format (Returns 400 CUSTOMER\_INVALID\_POSTAL\_CODE)

**Technical requirements :**

Endpoint : POST /api/customers

Password: bcrypt hashing

UUID: v4 format

Phone: E.164 validation

Names: Auto-capitalize to UPPERCASE

**DoD :**

* Code fully implemented according to user story acceptance criteria and API documentation
* All data conversions and data validations enforced
* Code peer-reviewed and approved by at least one other developer
* Unit and integration tests passing (including error scenarios)
* API documentation updated
* Automated regression tests updated

**Story Point :** To be defined with the development team

**User story 2**

**Feature :** Book searching

**Id :** US-02

**User story (title) :** Browse books on the catalog

**Description :** As a customer, I want to browse a book or discover all available books on the catalog by using filters, so I can find books that interest me to purchase

Provide an endpoint (Implement GET/api/books) with query parameters and filtering (category, author, price range). Results are paginated.

**Acceptance criteria :**

* Request without filters returns all books paginated (Returns 200 OK)
* Filter by category returns only matching books (Returns 200 OK)
* Empty catalog returns empty array (Returns 200 OK)
* Missing API key (Returns 401 AUTH\_MISSING\_API\_KEY)

**Technical requirements :**

Endpoint : GET/api/books

Default: page=1, limit=20

Max limit: 100 books/page

**DoD :**

* Code fully implemented according to user story acceptance criteria and API documentation
* All data conversions and data validations enforced
* Code peer-reviewed and approved by at least one other developer
* Unit and integration tests passing (including error scenarios)
* API documentation updated
* Automated regression tests updated

**Story Point :** To be defined with the development team

**User story 3**

**Feature :** Book searching

**Id :** US-03

**User story (title) :** View book details

**Description :** As a customer, I want to see full details of a selected book, so I can make a better buying decision

Retrieve complete details for a book by ID, including dynamically generated share links for social media platforms.

**Acceptance criteria :**

* Valid book ID returns complete details (ISBN, EAN, summary, publisher, stock, price,...) and share links (returns 200 OK)
* Invalid book ID (returns 404 BOOK\_NOT\_FOUND)

**Technical requirements :**

Endpoint: GET/api/books/{id}

Share link template not store in the DB

**DoD :**

* Code fully implemented according to user story acceptance criteria and API documentation
* All data conversions and data validations enforced
* Code peer-reviewed and approved by at least one other developer
* Unit and integration tests passing (including error scenarios)
* API documentation updated
* Automated regression tests updated

**Story Point :** To be defined with the development team

**User story 4**

**Feature :** Order Management

**Id :** US-04

**User story (title) :** Place an order

**Description :** As a customer, I want to plan an order, so I can purchase books online

Allow customers to create orders by selecting books and quantities. System validates stock, calculates total, processes payment, decrements stock, and generates order code.

**Acceptance criteria :**

* Valid order creates an order code with status "processing", decrements stock, copies shipping address (Returns 201 Created)
* Book out of stock (Returns 409 BOOK\_OUT\_OF\_STOCK)
* Empty cart (Returns 400 ORDER\_EMPTY\_ITEMS)
* Payment failure (Returns 422 ORDER\_PAYMENT\_FAILED) -Invalid quantity (Returns 400 ORDER\_INVALID\_QUANTITY)
* Book not found (Returns 404 BOOK\_NOT\_FOUND)
* Customer not found (Returns 404 CUSTOMER\_NOT\_FOUND)

**Technical requirements :**

Endpoint: POST/api/orders

Order code format: ORD-XXXXXX (zero-padded 6 digits)

Payment gateway integration (Simulated payment processed)

**DoD :**

* Code fully implemented according to user story acceptance criteria and API documentation
* All data conversions and data validations enforced
* Code peer-reviewed and approved by at least one other developer
* Unit and integration tests passing (including error scenarios)
* API documentation updated
* Automated regression tests updated

**Story Point :** To be defined with the development team

**User story 5**

**Feature :** Order Management

**Id :** US-05

**User story (title) :** Cancel an order

**Description :** As a customer, I want to cancel an order, so that I can modify the order or stop it before delivery.

Allow customers to cancel orders in "processing" status. Canceling restores stock quantities and changes status to "cancelled". Shipped/delivered orders cannot be cancelled.

**Acceptance criteria :**

* Cancel "processing" order → Status changes to "cancelled", stock restored (Returns 200 OK)
* Cancel "shipped" or "delivered" order (Returns 409 ORDER\_CANNOT\_CANCEL)
* Order not found (Returns 404 ORDER\_NOT\_FOUND)

**Technical requirements :**

Endpoint: PUT/api/orders/{orderId}/cancel

Database: UPDATE orders + restore stock quantities

**DoD :**

* Code fully implemented according to user story acceptance criteria and API documentation
* All data conversions and data validations enforced
* Code peer-reviewed and approved by at least one other developer
* Unit and integration tests passing (including error scenarios)
* API documentation updated
* Automated regression tests updated

**Story Point :** To be defined with the development team

**User story 6**

**Feature :** Notifications

**Id :** US-06

**User story (title) :** Subscribe to Book Availability Notification

**Description :** As a customer, I want to subscribe for a book’s availability, so that I can receive notifications when its restocked.

Allow customers to subscribe to availability notifications for out-of-stock books. Customers receive one notification per subscription when book is back in stock.

**Acceptance criteria :**

* Subscribe to out-of-stock book creates notification (Returns 201 Created)
* Subscribe to in-stock book (Returns 422 NOTIFICATION\_BOOK\_AVAILABLE)
* Duplicate subscription (Returns 409 NOTIFICATION\_ALREADY\_EXISTS)
* Book not found (Returns 404 BOOK\_NOT\_FOUND)
* Customer not found (Returns 404 CUSTOMER\_NOT\_FOUND)

**Technical requirements :**

Endpoint : POST /api/notifications/availability

Check stock quantity before

**DoD :**

* Code fully implemented according to user story acceptance criteria and API documentation
* All data conversions and data validations enforced
* Code peer-reviewed and approved by at least one other developer
* Unit and integration tests passing (including error scenarios)
* API documentation updated
* Automated regression tests updated

**Story Point :** To be defined with the development team

**User story 7**

**Feature :** Book management (admin)

**Id :** US-07

**User story (title) :** Add new book to the catalog

**Description :** As an admin, I want to add new books to the catalog, so customers can find and purchase them

Provide admin only endpoint to add new books. System validates all fields, ensures ISBN/EAN uniqueness, and auto-generates book ID.

**Acceptance criteria :**

* Valid book data creates new book with auto-generated ID (Returns 201 Created)
* Duplicate ISBN or EAN (Returns 409 Conflict)
* Invalid price (Returns 400 BOOK\_INVALID\_PRICE)
* Non-admin access (Returns 403 AUTH\_INSUFFICIENT\_PERMISSIONS)
* Missing required fields →(Returns 400 VALIDATION\_FAILED)

**Technical requirements :**

Endpoint : POST/api/books

Authorization: Admin API key required

Auto-increment books\_id

Validate ISBN/EAN length = 13

**DoD :**

* Code fully implemented according to user story acceptance criteria and API documentation
* All data conversions and data validations enforced
* Code peer-reviewed and approved by at least one other developer
* Unit and integration tests passing (including error scenarios)
* API documentation updated
* Automated regression tests updated

**Story Point :** To be defined with the development team

**User story 8**

**Feature :** Customer Account Management

**Id :** US-08

**User story (title) :** Update customer profile

**Description :** As a customer, I want to update my customer profile or address or password, so that my account reflects current details

Allow customers to update their profile information. Email cannot be changed. Password update requires old password verification.

**Acceptance criteria :**

* Valid updates save to database (Returns 200 OK with updated profile)
* Update email (Returns 400 Bad Request)
* Update password with wrong old password (Returns 401 Unauthorized)
* Invalid postal code format (Returns 400 CUSTOMER\_INVALID\_POSTAL\_CODE)
* Unsupported country (Returns 400 CUSTOMER\_INVALID\_COUNTRY)

**Technical requirements :**

Endpoint : PUT/api/customers/{customerId}

Password change requires: oldPassword + newPassword

Apply same validations as registration

**DoD :**

* Code fully implemented according to user story acceptance criteria and API documentation
* All data conversions and data validations enforced
* Code peer-reviewed and approved by at least one other developer
* Unit and integration tests passing (including error scenarios)
* API documentation updated
* Automated regression tests updated

**Story Point :** To be defined with the development team