Inventory Management System API Documentation

Technology Stack

Spring Boot 3

Spring Security

Spring Data JPA

PostgreSQL Database

Bean Validation

HTTP Basic Authentication

Overview

This REST API provides inventory management functionality with two distinct interfaces:

- **Admin Interface**: Read, add, and delete items from the database (requires authentication)
- **Customer Interface**: Read-only operations (public access)

Authentication

Admin endpoints require HTTP Basic Authentication:

- **Username**: `admin`
- **Password**: `admin123`

Credentials are configured via application properties: `app.security.username` and `app.security.password`

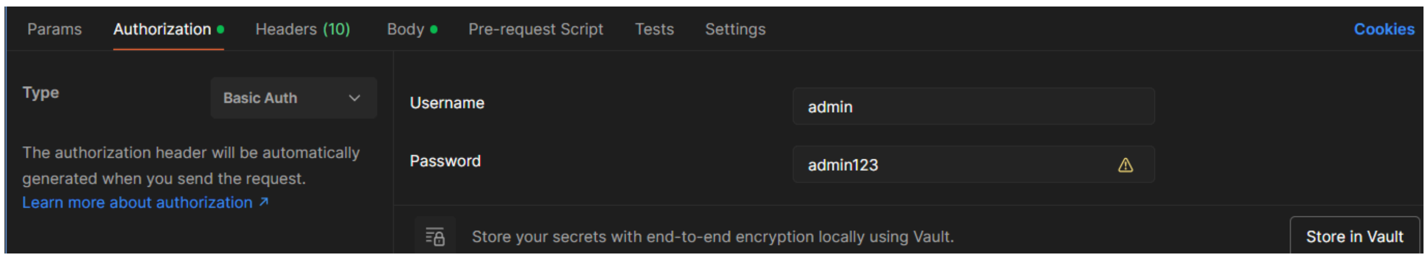
Using Postman

Authentication Setup:

1. Go to Authorization tab
2. Select "Basic Auth"
3. Username: `admin`
4. Password: `admin123`

Headers:

- `Content-Type: application/json` (for POST requests)



Type: Basic Auth

The authorization header will be automatically generated when you send the request.
Learn more about authorization ↗

Username: admin

Password: admin123

Store your secrets with end-to-end encryption locally using Vault.

Content Type

All requests and responses use `application/json` content type.

Base URL

http://localhost:8080

Admin API Endpoints

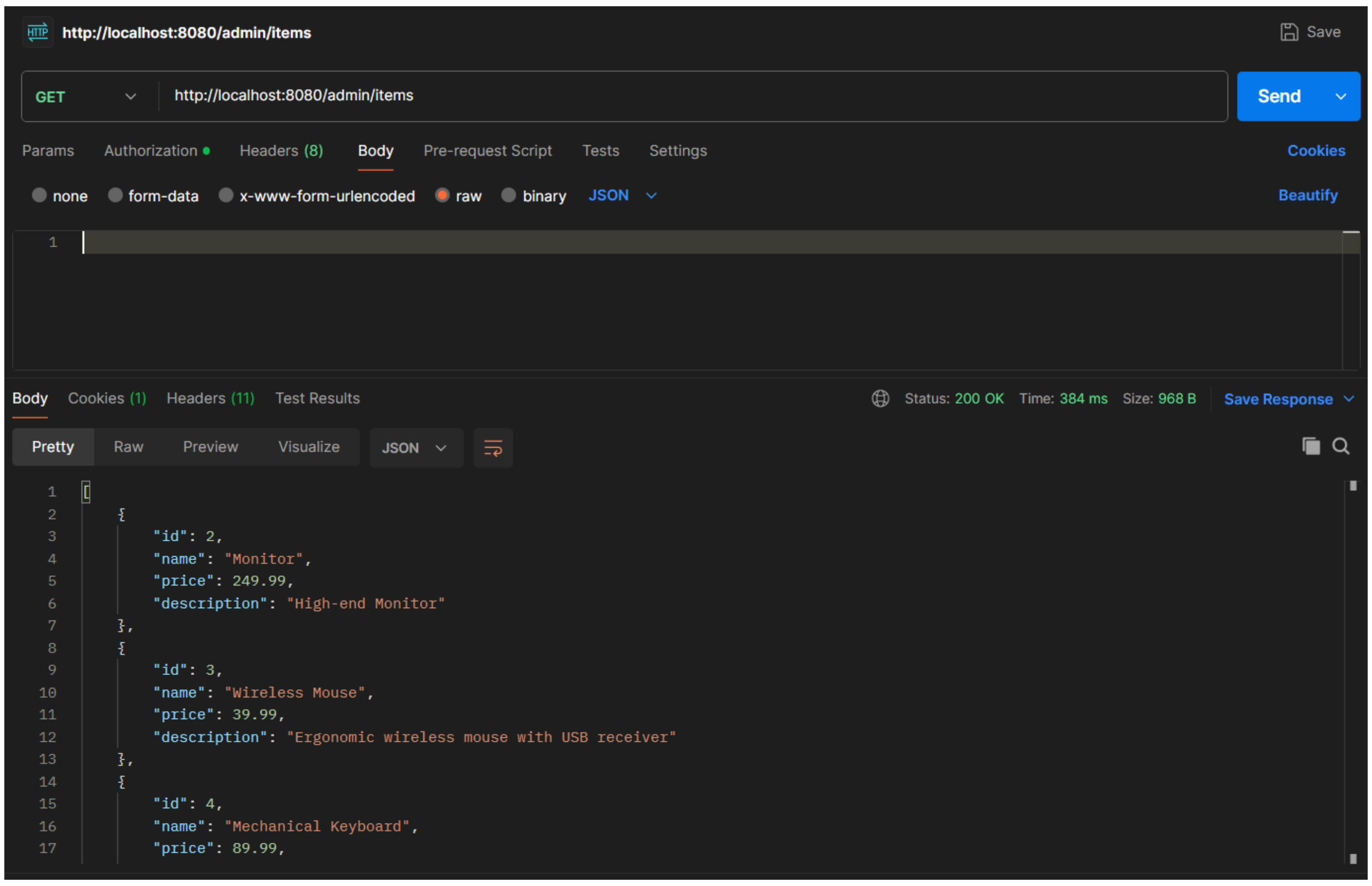
1. Get All Items

GET `/admin/items`

***Description**:* Retrieve all inventory items

***Authentication**:* Required (Admin)

***Response**:*



http://localhost:8080/admin/items

GET http://localhost:8080/admin/items

Status: 200 OK Time: 384 ms Size: 968 B

```
[
    {
        "id": 2,
        "name": "Monitor",
        "price": 249.99,
        "description": "High-end Monitor"
    },
    {
        "id": 3,
        "name": "Wireless Mouse",
        "price": 39.99,
        "description": "Ergonomic wireless mouse with USB receiver"
    },
    {
        "id": 4,
        "name": "Mechanical Keyboard",
        "price": 89.99,
```

HTTP Status Codes:

- `200 OK` - Success
- `401 Unauthorized` - Authentication required

2. Create New Item

POST `/admin/items`

***Description**:* Add a new item to the inventory

***Authentication**:* Required (Admin)

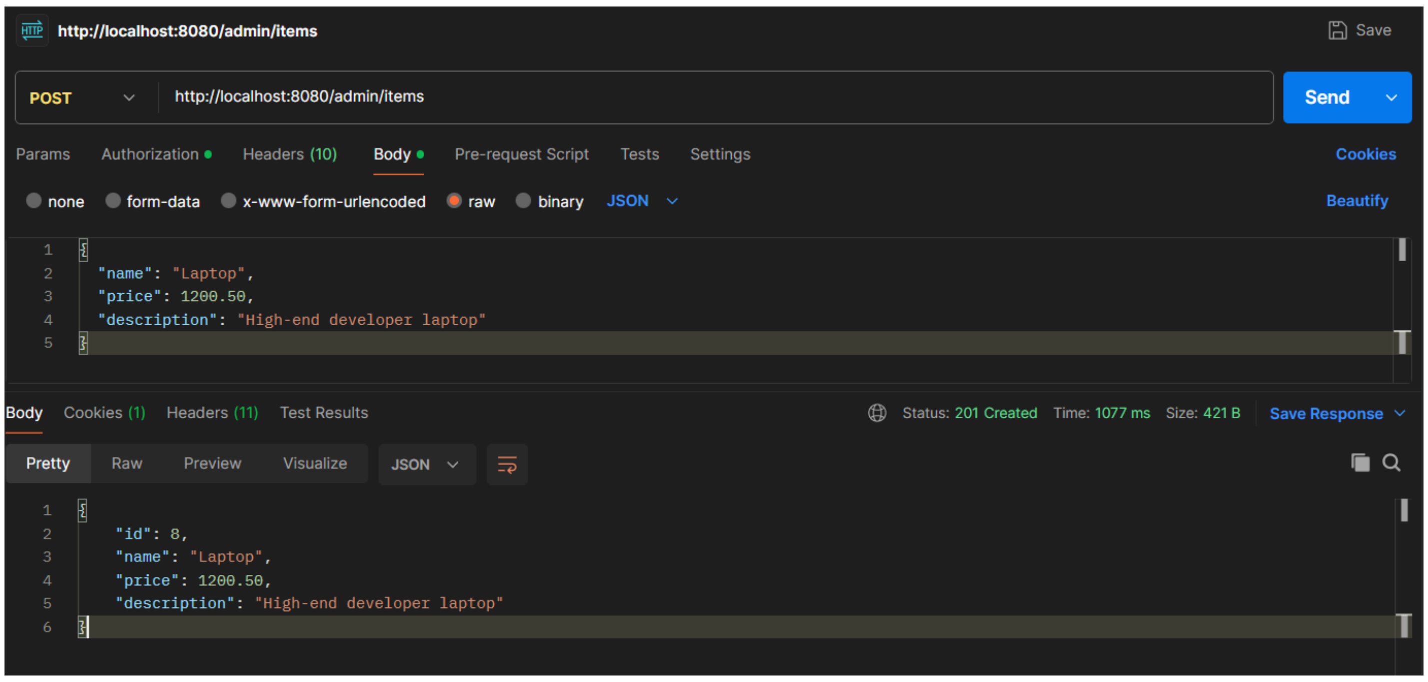
***Request Body**:*

```
{
  "name": "Laptop",
  "price": 1200.50,
  "description": "High-end developer laptop"
}
```

***Validation Rules**:*

- `name`: The field must not be `null`, empty, or only whitespace.
  the length must not exceed 255 characters.

- `price`: must not be `null`, must be at least `0.01` and must have up to **10 digits before the decimal** and **up to 2 digits after the decimal**.
- `description`: if provided (it's optional), must not exceed 1000 characters.



http://localhost:8080/admin/items

POST http://localhost:8080/admin/items

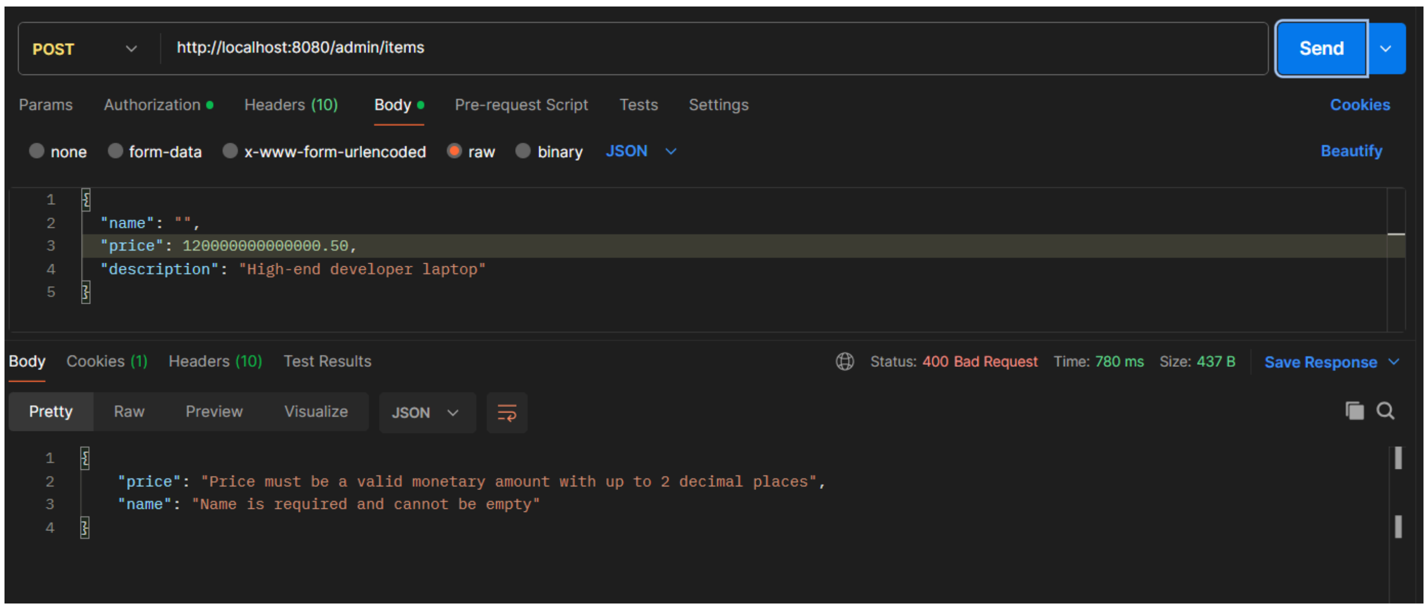
Body: raw, JSON

```
{
  "name": "Laptop",
  "price": 1200.50,
  "description": "High-end developer laptop"
}
```

Status: 201 Created Time: 1077 ms Size: 421 B

```
{
    "id": 8,
    "name": "Laptop",
    "price": 1200.50,
    "description": "High-end developer laptop"
}
```

Error Response Example:



POST http://localhost:8080/admin/items

Body: raw, JSON

```
{
  "name": "",
  "price": 1200000000000000.50,
  "description": "High-end developer laptop"
}
```

Status: 400 Bad Request Time: 780 ms Size: 437 B

```
{
    "price": "Price must be a valid monetary amount with up to 2 decimal places",
    "name": "Name is required and cannot be empty"
}
```

HTTP Status Codes:

- `201 OK` - Success
- `400 Bad Request` - Validation errors
- `401 Unauthorized` - Authentication required

3. Delete Item

DELETE `/admin/items/{id}`

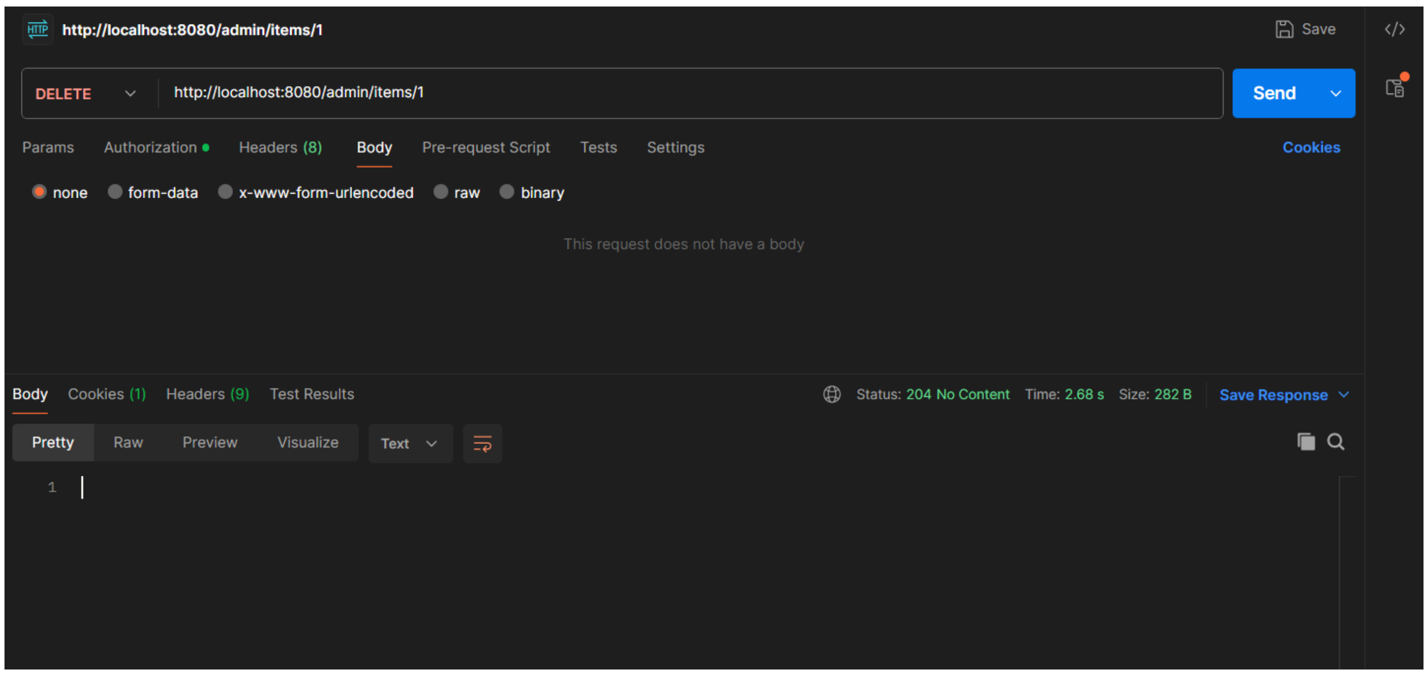
Description: Remove an item from the inventory

Authentication: Required (Admin)

Path Parameters:

- `id` (Long) - Item ID

Response: No content


http://localhost:8080/admin/items/1
Save
DELETE
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
● none ○ form-data ○ x-www-form-urlencoded ○ raw ○ binary

DELETE http://localhost:8080/admin/items/1

This request does not have a body

Status: 204 No Content Time: 2.68 s Size: 282 B

HTTP Status Codes:

- `204 No Content` - Item deleted successfully
- `401 Unauthorized` - Authentication required

Customer API Endpoints

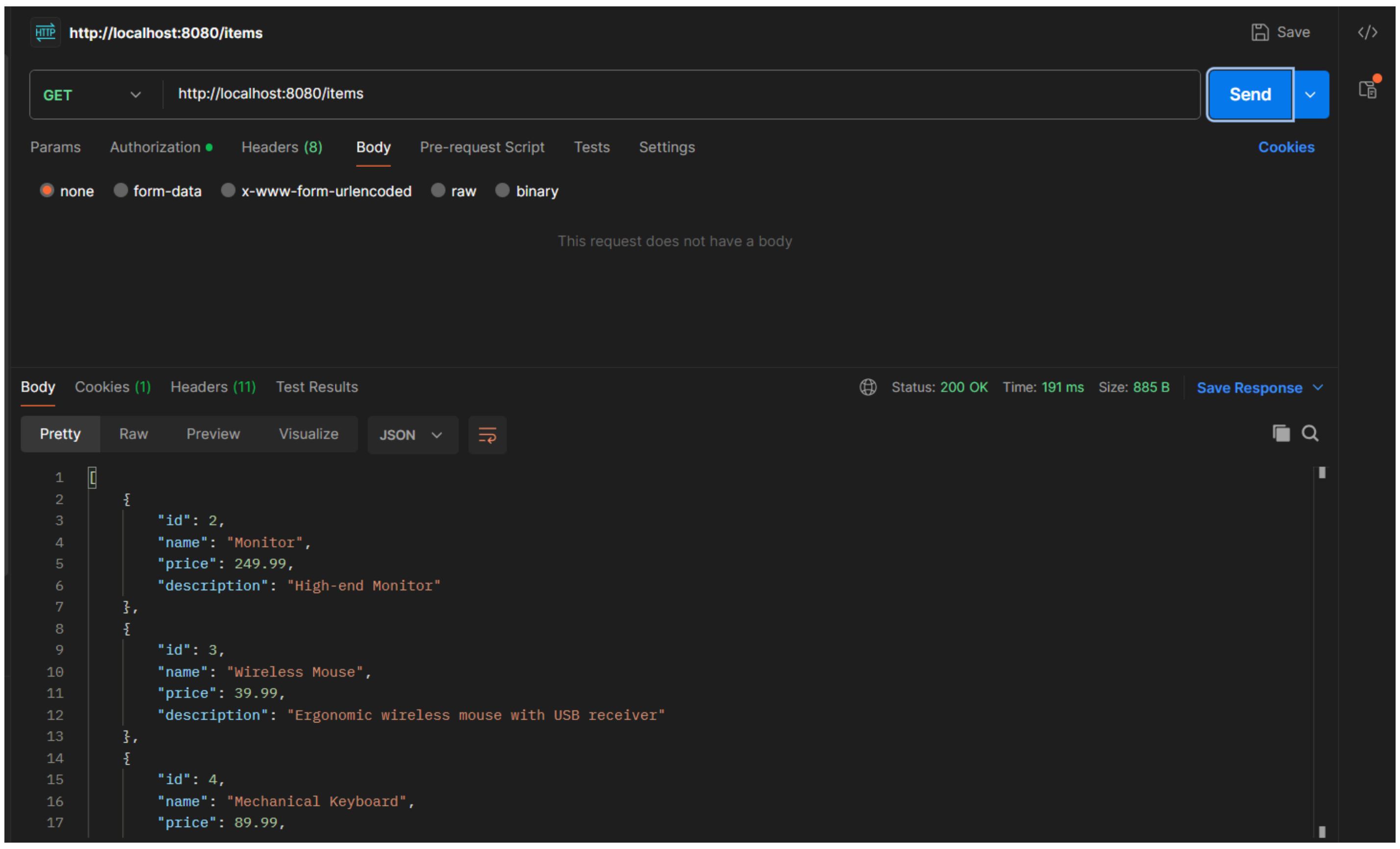
1. Get All Items (Customer)

GET `/items`

Description: Retrieve all available inventory items

Authentication: Not required

Response:


http://localhost:8080/items
Save
GET
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
● none ○ form-data ○ x-www-form-urlencoded ○ raw ○ binary

GET http://localhost:8080/items

This request does not have a body

Status: 200 OK Time: 191 ms Size: 885 B

```
[
    {
        "id": 2,
        "name": "Monitor",
        "price": 249.99,
        "description": "High-end Monitor"
    },
    {
        "id": 3,
        "name": "Wireless Mouse",
        "price": 39.99,
        "description": "Ergonomic wireless mouse with USB receiver"
    },
    {
        "id": 4,
        "name": "Mechanical Keyboard",
        "price": 89.99,
```

HTTP Status Codes:

- `200 OK` - Success

Database Schema

Items Table Structure

```
CREATE TABLE items (
    id BIGSERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    price NUMERIC(12,2) NOT NULL CHECK (price >= 0.01),
    description VARCHAR(1000)
);
```

Field Descriptions:

- `id`
    - Type: `BIGSERIAL`
    - Constraints: `PRIMARY KEY`
    - Description: Unique, auto-incrementing identifier for each item.
- `name`
    - Type: `VARCHAR(255)`
    - Constraints: `NOT NULL`
    - Description: Name of the item, cannot be null and limited to 255 characters.
- `price`
    - Type: `NUMERIC(12,2)`
    - Constraints: `NOT NULL` , `CHECK (price >= 0.01)`
    - Description: Price of the item, must be a positive number with up to 2 decimal places.
- `description`
    - Type: `VARCHAR(1000)`
    - Constraints: *(Optional)*
    - Description: Description of the item; can be null and up to 1000 characters.

Implementation Details

Security Features:

- HTTP Basic Authentication for admin endpoints
- Role-based access control (ADMIN role required for admin operations)
- SQL injection prevention through JPA/Hibernate

Error Handling:

- Global exception handler for consistent error responses
- Input validation with detailed error messages
- Proper HTTP status codes
- Structured error response format

Data Validation:

- Bean validation annotations
- Custom validation messages
- Server-side validation for all inputs
- Price validation to ensure positive values