Belle Croissant Lyonnais
API Documentation

Version: 1.0

Last Updated: July 11, 2024

1. Introduction

The Belle Croissant Lyonnais API provides programmatic access to product, customer, and order data for the bakery's management system. This RESTful API allows for secure, efficient interaction with the bakery's database, supporting operations such as retrieving product information, managing customer data, and processing orders.

2. Getting Started

2.1 Authentication

The API uses JSON Web Tokens (JWT) for authentication. To obtain a token, make a POST request to the `/api/v1/auth/login` endpoint with your credentials.

3. API Endpoints

3.1 Authentication

Endpoints for user authentication and session management.

Method	**POST**
Endpoint	/api/v1/auth/login
Description	Authenticate and receive a JWT token.
Request Body (JSON): **{** **"username": "staff",** **"password": "BCLyon2024"** **}**	
Response (JSON): **{** **"access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",** **"session_id": "550e8400-e29b-41d4-a716-446655440000"** **}**	

3.2 Products

Endpoints for managing product information.

Method	GET
Endpoint	/api/v1/products
Description	Retrieve a list of all products.

Method	GET
Endpoint	/api/v1/products/{id}
Description	Retrieve a specific product by ID.

Method	POST
Endpoint	/api/v1/products
Description	Create a new product.

Request Body (JSON):

```
{
  "product_name": "Baguette",
  "category": "Bread",
  "price": 3.00,
  "cost": 1.50,
  "ingredients": "Flour, Water, Yeast, Salt",
  "seasonal": "FALSE",
  "active": "TRUE"
}
```

Response (JSON):

```
{
  "product_id": "2",
  "product_name": "Baguette",
  "category": "Bread",
  "price": "3.00",
  "cost": "1.50",
  "ingredients": "Flour, Water, Yeast, Salt",
  "seasonal": "FALSE",
  "active": "TRUE"
}
```

Method	PUT
Endpoint	/api/v1/products/{id}
Description	Update an existing product.

Request Body (JSON):

```
{
      "price": 3.50,
      "active": "FALSE"
  }
```

Response (JSON):

```
{
 "product_id": "2",
 "product_name": "Baguette",
 "category": "Bread",
 "price": "3.50",
 "cost": "1.50",
 "ingredients": "Flour, Water, Yeast, Salt",
 "seasonal": "FALSE",
 "active": "FALSE"
}
```

Method	DELETE
Endpoint	/api/v1/products/{id}
Description	Delete a product.

3.2 Customers

Endpoints for managing customer data.

Method	GET
Endpoint	/api/v1/customers
Description	Retrieve a list of all customers.

Method	GET
Endpoint	/api/v1/customers/<int:id>
Description	Retrieve a specific customer by ID.

Method	POST
Endpoint	/api/v1/customers
Description	Create a new customer.

Request Body (JSON):


```
{
      "first_name": "Jean",
      "last_name": "Martin",
      "email": "jean.martin@email.com",
      "age": "35",
      "gender": "M",
      "postal_code": "69006",
      "phone_number": "33612345678",
      "membership_status": "Basic"
  }
```


Response (JSON):


```
{
      "customer_id": "103",
      "first_name": "Jean",
      "last_name": "Martin",
      "age": "35",
      "gender": "M",
      "postal_code": "69006",
      "email": "jean.martin@email.com",
      "phone_number": "33612345678",
      "membership_status": "Basic",
      "join_date": "7/11/2024",
      "last_purchase_date": "",
      "total_spending": "0",
      "average_order_value": "0",
      "frequency": "0",
      "preferred_category": "",
      "churned": "FALSE"
  }
```


Method	PUT
Endpoint	/api/v1/customers/<int:id>
Description	Update an existing customer.

Request Body (JSON):


```
{
      "membership_status": "Gold",
      "phone_number": "33687654321"
  }
```

Response (JSON):


```
{
    "customer_id": "103",
    "first_name": "Jean",
    "last_name": "Martin",
    "age": "35",
    "gender": "M",
    "postal_code": "69006",
    "email": "jean.martin@email.com",
    "phone_number": "33687654321",
    "membership_status": "Gold",
    "join_date": "7/11/2024",
    "last_purchase_date": "",
    "total_spending": "0",
    "average_order_value": "0",
    "frequency": "0",
    "preferred_category": "",
    "churned": "FALSE"
  }
```


Method	DELETE
Endpoint	/api/v1/customers/<int:id>
Description	Delete a customer.

3.3 Orders

Endpoints for creating, retrieving, updating, and deleting orders.

Method	GET
Endpoint	/api/v1/orders
Description	Retrieve a list of all orders.

Method	GET
Endpoint	/api/v1/orders/<int:id>
Description	Retrieve a specific order by ID.

Method	POST
Endpoint	/api/v1/orders
Description	Create a new order.

Request Body (JSON):


```
{
    "customer_id": "103",
    "date": "2024-07-11",
    "time": "14:30:00",
    "product_id": "2",
    "quantity": 3,
    "price": 3.50,
    "payment_method": "Credit Card",
    "channel": "In-store",
    "store_id": "1",
    "promotion_id": "",
    "discount_amount": 0.0
  }
```


Response (JSON):


```
{
    "transaction_id": "3",
    "customer_id": "103",
    "date": "2024-07-11",
    "time": "14:30:00",
    "product_id": "2",
    "quantity": "3",
    "price": "3.50",
    "payment_method": "Credit Card",
    "channel": "In-store",
    "store_id": "1",
    "promotion_id": "",
    "discount_amount": "0.0"
  }
```


Method	PUT
Endpoint	/api/v1/orders/<int:id>
Description	Update an existing order.

Request Body (JSON):


```
{
    "quantity": 4,
    "discount_amount": 1.0
  }
```

Response (JSON):


```
{
      "transaction_id": "3",
      "customer_id": "103",
      "date": "2024-07-11",
      "time": "14:30:00",
      "product_id": "2",
      "quantity": "4",
      "price": "3.50",
      "payment_method": "Credit Card",
      "channel": "In-store",
      "store_id": "1",
      "promotion_id": "",
      "discount_amount": "1.0"
  }
```


Method	DELETE
Endpoint	/api/v1/orders/<int:id>
Description	Delete an order.

Method	GET
Endpoint	/api/v1/customers/<int:customer_id>/orders
Description	Retrieve all orders for a specific customer.

Method	GET
Endpoint	/api/v1/products/<int:product_id>/orders
Description	Retrieve all orders for a specific product.

3.4 Session Management

Method	POST
Endpoint	/api/v1/reset
Description	Reset session data.

Response (JSON):


```
{"message": "Session data reset successfully"}
```


4. Error Handling

The API uses conventional HTTP response codes to indicate the success or failure of an API request. In general:

2xx range indicate success

4xx range indicate an error that failed given the information provided (e.g., a required parameter was omitted, a resource was not found, or authentication failed)

5xx range indicate an error with our servers

All error responses will include a JSON object with an `error` and `message` field.

Method	
Endpoint	
Description	Example 404 Error:
Response (JSON):	

```
{
    "error": "Not Found",
    "message": "The requested resource was not found"
}
```

Method	
Endpoint	
Description	Example 401 Error:
Response (JSON):	

```
{
    "error": "Unauthorized",
    "message": "Authentication credentials were missing or incorrect"
}
```

Method	
Endpoint	
Description	Example 500 Error:
Response (JSON):	

```
{
    "error": "Internal Server Error",
    "message": "An unexpected error occurred on the server"
}
```

5. Rate Limiting

The API currently does not implement rate limiting. However, we recommend clients to implement reasonable request rates to ensure optimal performance for all users.

6. Versioning

The API is versioned using URL path versioning. The current version is v1. When we make backwards-incompatible changes to the API, we will release a new version number.

7. Security

- All API requests should be made over HTTPS in production environments
- Authentication tokens are transmitted securely
- Passwords are never returned in API responses
- API keys should be kept secure and not shared

8. Best Practices

- - Use HTTPS for all requests in production environments
- - Implement proper error handling in your applications
- - Cache API responses when appropriate
- - Use compression for large requests or responses
- - Implement exponential backoff when receiving server errors

9. Changelog

Version 1.0 (July 11, 2024)

- - Initial release of the API
- - Implemented CRUD operations for Products, Customers, and Orders
- - Added session management and authenticat