Email Notifications API

# Introduction

# The Email Notifications module enables the system to automatically send email updates to customers. This includes sending order confirmation emails after an order is created and shipping notifications when an order is marked as shipped. This feature improves customer communication and ensures transparency in the order management process.

# Features

* **Order Confirmation:** An email is sent to the customer when an order is successful, and the status of an order is updated to " COMPLETED".
* **Order Acknowledgment:** An email is sent to the customer immediately when an order is placed to confirm that the order has been received and is being processed.
* **Shipping Notification:** An email is sent to the customer when the status of an order is updated to "SHIPPED".
* **Stock shortage**: An email is sent to admin when the product stock is less than minimum.

**The minimum value is defined as default value for quantity of each product in inventory.**

# Email Settings

Email configuration in the project is controlled through the *settings.py* file. In development environments, a simple backend can be used, while in production, an SMTP server is required to send emails:

Ein Bild, das Text, Screenshot, Schrift, Software enthält.

Automatisch generierte Beschreibung

* In the development environment, emails are output to the console.
* In the production environment, the *EMAIL\_BACKEND* is set to *smtp.EmailBackend*, and the required SMTP server information is loaded from environment variables.

# API Endpoints

* **Base URL:** The API is accessible at **/orders/**.
* Endpoints and HTTP Methods:

|  |  |  |
| --- | --- | --- |
| Method | Endpoint | Description |
| GET | /orders/ | |  | | --- | | Retrieve all orders. |  |  | | --- | |  | |
| POST | /orders/ | |  | | --- | | Create a new order (triggers confirmation email). | |
| GET | /orders/{id}/ | |  | | --- | | Retrieve details of a specific order. |  |  | | --- | |  | |
| PUT | /orders/{id}/ | |  | | --- | | Update an existing order (triggers shipping notification if status changes to “SHIPPED”). |  |  | | --- | |  | |
| DELETE | /orders/{id}/ | Delete an order. |

## Examples

1. **Create an Order (POST)**

* **Endpoint:** /orders/
* **Request Type:** POST
* **Request Body (JSON):**

{

    "customer\_name": "John Doe",

    "customer\_email": "johndoe@example.com",

    "products": {

        "products": [

            {

                "name": "Product A",

                "quantity": 2

            }

        ]

    },

    "total\_price": "99.99",

}

* **Response:**

{

"id": 1,

"customer\_name": "Jane Doe",

"customer\_email": "janedoe@example.com",

"products": {

"item1": 2,

"item2": 1

},

"total\_price": 99.99,

"status": "PENDING",

"created\_at": "2025-01-25T12:34:56.789123Z",

"updated\_at": "2025-01-25T12:34:56.789145Z"

}

* **Action 1**: Triggers an order confirmation email to the provided customer email address.
* **Action 2**: When the payment is successfully completed, the order status will be set to “COMPLETED”. An order confirmation email will be sent again to the specified customer email address.

1. **Update Order Status (PUT)**

* **Endpoint:** /orders/{id}
* **Request Type:** PUT
* **Request Body (JSON):**

{

    "customer\_name": "John Doe",

    "customer\_email": "johndoe@example.com",

    "products": {

        "products": [

            {

                "name": "Product A",

                "quantity": 2

            }

        ]

    },

    "total\_price": "99.99",

    "status": "SHIPPED"

}

* **Response:**

{

"id": 1,

"customer\_name": "Jane Doe",

"customer\_email": "janedoe@example.com",

"products": {

        "products": [

            {

                "name": "Product A",

                "quantity": 2

            }

        ]

    },

"total\_price": 99.99,

"status": "SHIPPED",

"created\_at": "2025-01-25T12:23:39.930922Z",

"updated\_at": "2025-01-25T13:00:28.600840Z"

}

* **Action**: Sends a shipping notification email to the customer's email address.

## Testing the API

* **Manual Testing**

Use tools like **Postman** or **cURL** to manually test the endpoints.

* **Automated Testing**

Test case class *EmailNotificationTestCase (TestCase)* in *tests.py* to ensure API functionality:

Run the tests using:

**python manage.py test**