

# Customer API Documentation

## Base URL

/api/customers

## Overview

This REST API provides endpoints for managing customer data including CRUD operations, search functionality, and filtering capabilities.

## Endpoints

### 1. Get All Customers (Paginated)

Retrieve a paginated list of all customers with optional sorting.

**Endpoint:** GET /api/customers

#### Query Parameters:

- page (optional, default: 0) - Page number (zero-indexed)
- size (optional, default: 10) - Number of items per page
- sortBy (optional) - Field name to sort by
- sortDir (optional, default: "asc") - Sort direction ("asc" or "desc")

**Response:** 200 OK

json

```
{
  "customers": [
    {
      "id": 1,
      "name": "John Doe",
      "email": "john@example.com",
      "status": "active"
    }
  ],
  "currentPage": 0,
  "totalItems": 50,
  "totalPages": 5
}
```

### Example Request:

```
GET /api/customers?page=0&size=10&sortBy=name&sortDir=asc
```

---

## 2. Get Customer by ID

Retrieve a specific customer by their ID.

**Endpoint:** `GET /api/customers/{id}`

### Path Parameters:

- `{id}` (required) - Customer ID

**Response:** `200 OK`

```
json
{
  "id": 1,
  "name": "John Doe",
  "email": "john@example.com",
  "status": "active"
}
```

**Error Response:** `404 Not Found` - Customer not found

---

### 3. Create Customer

Create a new customer.

**Endpoint:** `POST /api/customers`

**Request Body:**

```
json
{
  "name": "John Doe",
  "email": "john@example.com",
  "status": "active"
}
```

**Response:** `201 Created`

```
json
{
  "id": 1,
  "name": "John Doe",
  "email": "john@example.com",
  "status": "active"
}
```

**Error Response:** `400 Bad Request` - Validation error

---

### 4. Update Customer

Update an existing customer (full update).

**Endpoint:** `PUT /api/customers/{id}`

**Path Parameters:**

- `{id}` (required) - Customer ID

**Request Body:**

```
json
```

```
{
  "name": "John Smith",
  "email": "johnsmith@example.com",
  "status": "active"
}
```

**Response:** 200 OK

json

```
{
  "id": 1,
  "name": "John Smith",
  "email": "johnsmith@example.com",
  "status": "active"
}
```

### Error Responses:

- 400 Bad Request - Validation error
  - 404 Not Found - Customer not found
- 

## 5. Partial Update Customer

Partially update an existing customer (only provided fields).

**Endpoint:** PATCH /api/customers/{id}

### Path Parameters:

- {id} (required) - Customer ID

### Request Body:

json

```
{
  "status": "inactive"
}
```

**Response:** 200 OK

json

```
{
  "id": 1,
  "name": "John Doe",
  "email": "john@example.com",
  "status": "inactive"
}
```

**Error Response:** `404 Not Found` - Customer not found

---

## 6. Delete Customer

Delete a customer by ID.

**Endpoint:** `DELETE /api/customers/{id}`

**Path Parameters:**

- `{id}` (required) - Customer ID

**Response:** `200 OK`

json

```
{
  "message": "Customer deleted successfully"
}
```

**Error Response:** `404 Not Found` - Customer not found

---

## 7. Search Customers

Search customers by keyword across multiple fields.

**Endpoint:** `GET /api/customers/search`

**Query Parameters:**

- `keyword` (required) - Search keyword

**Response:** `200 OK`

json

```
[
  {
    "id": 1,
    "name": "John Doe",
    "email": "john@example.com",
    "status": "active"
  }
]
```

### Example Request:

```
GET /api/customers/search?keyword=john
```

---

## 8. Get Customers by Status

Retrieve all customers with a specific status.

**Endpoint:** `GET /api/customers/status/{status}`

### Path Parameters:

- `{status}` (required) - Customer status (e.g., "active", "inactive")

**Response:** `200 OK`

json

```
[
  {
    "id": 1,
    "name": "John Doe",
    "email": "john@example.com",
    "status": "active"
  }
]
```

### Example Request:

```
GET /api/customers/status/active
```

---

## 9. Advanced Search

Search customers with multiple optional filters.

**Endpoint:** `GET /api/customers/advanced-search`

### Query Parameters:

- `name` (optional) - Filter by name
- `email` (optional) - Filter by email
- `status` (optional) - Filter by status

**Response:** `200 OK`

```
json

[
  {
    "id": 1,
    "name": "John Doe",
    "email": "john@example.com",
    "status": "active"
  }
]
```

### Example Request:

```
GET /api/customers/advanced-search?name=John&status=active
```

---

## Common Response Codes

- `200 OK` - Request successful
- `201 Created` - Resource created successfully
- `400 Bad Request` - Invalid request or validation error
- `404 Not Found` - Resource not found
- `500 Internal Server Error` - Server error

## **CORS Configuration**

The API allows cross-origin requests from all origins ( $\textcircled{*}$ ). This enables frontend applications to consume the API from different domains.

## **Data Transfer Objects**

### **CustomerRequestDTO**

Used for creating and updating customers (full update).

### **CustomerUpdatedDTO**

Used for partial updates (PATCH requests).

### **CustomerResponseDTO**

Returned in all successful responses containing customer data.

## **Notes**

- All request and response bodies use JSON format
- The API uses standard HTTP methods (GET, POST, PUT, PATCH, DELETE)
- Request validation is performed using Jakarta Bean Validation annotations
- Pagination is zero-indexed (first page is 0)