Food delivery management system REST API Documentation

1. ENTITIES

The entities that the system will have are user, session, cart, category, subcategory, dishes, review, payment, order, notification, complaint, city, country, phone, address, photos, restaurant, plan, coupon, attribute, deposit_account, payment_methods and delivery.

2.1 FUNCTIONAL REQUIREMENTS.

USERS

Registration and login: The application must allow users to register and access with their credentials.

Personal Information Management: Users should be able to save basic information, location, phone numbers, and credit cards on the platform, as well as set preferences, such as preferred address and preferred payment method. They should also be able to change their password, email, phone, address, and status (unsubscribe).

CUSTOMER.

Save food preferences/restrictions: The application must allow Customers to set what type of food they prefer or what type of ingredients they do not want in their food so that the platform shows them the offer that meets their requirements.

See all the offers: Customers should be able to see all the gastronomic offers in-stock offered by the different restaurants.

View details of a dish: Customers should be able to get a detailed description of a dish, including its ingredients and an estimated delivery time to their location.

View reviews: Customers should be able to see reviews that other customers have left for a particular restaurant.

Rate Restaurant: Customers should be able to rate the experience they've had as a customer at a certain restaurant.

Filter Results: The application must allow customers to filter search results by location, categories, sub-categories, favorite restaurants, preferences and/or restrictions, attributes, and price.

Save favorites: Customers should be able to mark a dish or restaurant as a favorite.

Add to Cart: Customers must be able to add one or more items to the shopping cart which can lead to orders at one or more restaurants.

Discount coupons: Customers must be able to redeem a discount coupon during the purchase process.

Payment of the purchase: Customers must be able to pay for their orders through different forms of payment.

View Order Status: Customers should be able to see if their order has shipped and track it.

File a claim: Customers should be able to file a claim with the platform for poor service from a restaurant or delivery person, which could lead to a change of status of the order.

Confirm claim status: Customers should be able to notify whether or not their claim has been successfully completed.

RESTAURANT OWNER

Manage their restaurant information: The system should allow restaurant owners to create a restaurant and add important information such as contact information, location, history, and business identification. They must also have the ability to edit some of this information and deactivate the operation on the restaurant platform they created.

Manage their menu: Restaurant owners should be able to post dishes on the platform, as well as edit their description, price, and quantity available, or change their status to deactivate a post, as long as they don't have orders pending delivery.

Create coupons: The application must allow restaurant owners to create discount coupons and send them to consumers of their choice. To choose which type of consumer to send their coupons to, the restaurant owner must be able to categorize consumers to reach a target audience.

Receive orders: The system must allow restaurant owners to receive orders from their customers. They should receive a notification when a consumer has placed an order and see the detail.

Mark an order as ready for delivery: Restaurant owners should be able to report (change the status of an order) when an order has been prepared and is ready for delivery by the delivery person.

Cancel an order: Restaurant owners should have the option to cancel an order if the dish is out of stock.

Receive payments: The restaurant owner must be able to receive payments equal to their earnings.

Order history: Restaurant owners should be able to see a list with the history of orders placed, classifying them by date, customer, and status. They must also be able to see the detailed information of each order placed.

Sales Dashboard: The system must allow restaurant owners to see detailed information about their sales and profits.

File a claim: Restaurant owners should be able to file a complaint or claim against a delivery person.

Choose favorite couriers: Restaurant owners should be able to choose the couriers who have provided the best services to give them priority when notifying them when an order is due.

Block delivery drivers: Restaurant owners should be able to decide not to use the services of a specific delivery driver.

ADMINISTRATOR

Content management: The administrator must be able to create, edit and delete all the information related to users, restaurants, and dishes on the platform.

Category and attribute management: The administrator must be able to create, edit and delete categories, subcategories, and attributes.

Reports: The administrator must be able to see graphic pieces of all the information registered in the platform, such as consumer trends, sales history, restaurants with a bad reputation, most requested dishes, the total number of canceled orders, the

number of accounts created or deleted, and others kind of reports that facilitate decision making.

Earnings by Period: The admin should be able to view an earnings report categorized by periods and regions.

Sending notifications: The administrator must be able to group users and send notifications to them.

Claims status: The administrator must be able to see the status and details of all the claims made by the users and modify their status if necessary.

DELIVERY PERSON

See nearby orders: The delivery person must be able to see in real-time the available orders near his location.

Assign a shipment: The delivery person should be able to choose the shipment they want to make as long as they do not have restrictions from the restaurant.

View shipment history: The delivery person must be able to access a list of all the shipments they have made, including detailed information for each one.

Change delivery status: The delivery person should be able to change the delivery status to "delivered".

View optimal route: The driver should be able to see a suggested optimal route for the shipment that has been assigned to reach the customer's location more efficiently.

Notify changes: The delivery person should be able to notify any changes in the status of the shipment or if there are problems during delivery, such as missing products or delivery delays.

View contact information: The delivery person must be able to see the contact information of the customer to whom the delivery is being made to communicate if necessary.

Receive payments: The restaurant owner must be able to receive payments equal to their earnings.

CUSTOMER SERVICE PERSON

Access claim details: Customer service personnel should be able to access detailed information about the claims assigned to them.

Report claim status: Customer service personnel should be able to change the status of a claim registered in the system.

Receive payments: The restaurant owner must be able to receive payments equal to their earnings.

2.1 NON-FUNCTIONAL REQUIREMENTS

Security: The application must be secure and protect the information of the users.

3. REST API DESIGN

```
openapi: 3.0.0
info:
  version: 1.0.0
  title: delivery food
  description: >-
    Food sales and delivery application
     - Category: trade -
     It provides the following functionalities:
     1.- Filter gastronomic offer by characteristics and search by search box.
     2.- Registration and authentication with credentials
     3.- Tracking of shipments in real time
     4.- Automated payment system
     5.- Comment system
  contact:
    name: Karen Urbano
    email: quemojojojo@gmail.com

servers:
  - url: https://delivery_food.ue.r.appspot.com/api/v1
  - url: http://localhost:8081/api/v1

tags:
  - name: Dishes
  - name: Cart
  - name: Order
```

```
  - name: Payment_method
  - name: Deposit_account
  - name: Delivery
  - name: Reviews
  - name: Category
  - name: Subcategory
  - name: Restaurant
  - name: User
  - name: Auth
  - name: Payment

paths:
  /dishes:
    get:
      summary: Get all dishes
      description: Retrieve all dishes from the database
      tags:
        - Dishes
      parameters:
        - $ref: '#/components/parameters/Page'
        - $ref: '#/components/parameters/Limit'
      responses:
        '200':
          description: A list of dishes
          content:
            application/json:
              schema:
                type: object
                properties:
                  data:
                    type: array
                    items:
                      $ref: '#/components/schemas/dishes'
            application/xml:
              schema:
                type: object
                properties:
                  data:
                    type: array
                    items:
                      $ref: '#/components/schemas/dishes'
                xml:
                  name: 'dishes'
        400:
```

```
          $ref: '#/components/responses/BadRequest'
        404:
          $ref: '#/components/responses/NotFound'
        500:
          $ref: '#/components/responses/InternalServerError'
  /dishes/{slug}:
    get:
      summary: Get a single dish by SLUG
      description: Returns a single dish based on the slug provided
      operationId: getDishBySlug
      tags:
        - Dishes
      parameters:
        - $ref: '#/components/parameters/DishIdPathParam'
      responses:
        '200':
          description: OK
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/dish'
            application/xml:
              schema:
                $ref: '#/components/schemas/dish'
        400:
          $ref: '#/components/responses/BadRequest'
        404:
          $ref: '#/components/responses/NotFound'
        500:
          $ref: '#/components/responses/InternalServerError'
  /dishes/by_restaurant/{restaurantId}:
    get:
      summary: Retrieve a paginated list of dishes by restaurant ID
      description: Retrieve a paginated list of dishes that belong to a specific
restaurant, identified by the given ID.
      operationId: getDishesByRestaurant
      tags:
        - Dishes
      parameters:
        - $ref: '#/components/parameters/RestaurantIdPathParam'
        - $ref: '#/components/parameters/Page'
        - $ref: '#/components/parameters/Limit'
      responses:
        200:
```

```
        description: A paginated list of dishes belonging to the specified restaurant
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/dishes'
          application/xml:
            schema:
              $ref: '#/components/schemas/dishes'
      400:
        $ref: '#/components/responses/BadRequest'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
  /dishes/search:
    get:
      summary: Search dishes by name, category, subcategory, min price, max price,
city, and size.
      tags:
        - Dishes
      parameters:
        - name: name
          in: query
          description: Search dishes by name.
          schema:
            type: string
        - name: category
          in: query
          description: Search dishes by category.
          schema:
            type: string
        - name: subcategory
          in: query
          description: Search dishes by subcategory.
          schema:
            type: string
        - name: min_price
          in: query
          description: Search dishes with a minimum price.
          schema:
            type: number
            minimum: 0
        - name: max_price
          in: query
```

```
        description: Search dishes with a maximum price.
        schema:
          type: number
          minimum: 0
      - name: city
        in: query
        description: Search dishes by city.
        schema:
          type: string
      - name: size
        in: query
        description: Search dishes by size.
        schema:
          type: string
      - $ref: '#/components/parameters/Page'
      - $ref: '#/components/parameters/Limit'
    responses:
      '200':
        description: A list of dishes matching the search criteria.
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/dishes'
          application/xml:
            schema:
              $ref: '#/components/schemas/dishes'
      400:
        $ref: '#/components/responses/BadRequest'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
/dishes/search_box:
  get:
    summary: Search dishes by term of search
    description: Returns a list of dishes that match the specified search term.
    parameters:
      - in: query
        name: termOfSearch
        schema:
          type: string
        required: true
        description: The search term to match dishes against.
    responses:
```

```
      '200':
        description: A list of dishes that match the specified search term.
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/dishes'
          application/xml:
            schema:
              $ref: '#/components/schemas/dishes'
      400:
        $ref: '#/components/responses/BadRequest'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
    tags:
      - Dishes
/dishes/:
  post:
    summary: Create a new dish
    tags:
      - Dishes
    requestBody:
      description: Dish object to be created
      required: true
      content:
        multipart/form-data:
          schema:
            $ref: '#/components/schemas/dishForm'
        application/xml:
          schema:
            $ref: '#/components/schemas/dishForm'
    responses:
      '201':
        description: Dish created successfully
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/dish'
          application/xml:
            schema:
              $ref: '#/components/schemas/dish'
      400:
        $ref: '#/components/responses/BadRequest'
```

```
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
    security:
      - cookieAuth: []
/dishes/{slug}/:
  put:
    summary: Update a dish by ID
    description: Update an existing dish by its ID.
    tags:
      - Dishes
    parameters:
      - $ref: '#/components/parameters/DishIdPathParam'
    requestBody:
      required: true
      content:
        multipart/form-data:
          schema:
            $ref: '#/components/schemas/dishForm'
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/dish'
          application/xml:
            schema:
              $ref: '#/components/schemas/dish'
      '400':
        $ref: '#/components/responses/BadRequest'
      '401':
        $ref: '#/components/responses/Unauthorized'
      '403':
        $ref: '#/components/responses/Forbidden'
      '404':
        $ref: '#/components/responses/NotFound'
      '500':
        $ref: '#/components/responses/InternalServerError'
```

```
    security:
      - cookieAuth: []
/dishes/delete/{slug}/:
  delete:
    summary: Delete a dish
    tags:
      - Dishes
    parameters:
      - $ref: '#/components/parameters/DishIdPathParam'
    security:
      - cookieAuth: []
    responses:
      204:
        $ref: '#/components/responses/RemovedSuccessful'
      400:
        $ref: '#/components/responses/BadRequest'
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'

 #carro de un usuario

 #bearer

/cart/:
  get:
    summary: Get of the loged user
    security:
      - cookieAuth: []
    tags:
      - Cart
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/cart'
          application/xml:
```

```
            schema:
              $ref: '#/components/schemas/cart'
      400:
        $ref: '#/components/responses/BadRequest'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
/cart:
  post:
    summary: Create a new cart
    description: Creates a new cart.
    tags:
      - Cart
    requestBody:
      required: true
      content:
        multipart/form-data:
          schema:
            $ref: '#/components/schemas/cartForm'
        application/xml:
          schema:
            $ref: '#/components/schemas/cartForm'
    responses:
      '201':
        description: The created cart
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/cart'
          application/xml:
            schema:
              $ref: '#/components/schemas/cart'
      400:
        $ref: '#/components/responses/BadRequest'
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
    security:
```

```
      - cookieAuth: []
  /cart/{cartId}/:
    put:
      summary: Edit an existing cart
      description: Path on which the user who owns the cart makes changes such as
adding and removing items
      tags:
        - Cart
      security:
        - cookieAuth: []
      parameters:
        - $ref: '#/components/parameters/CartIdPathParam'
      requestBody:
        required: true
        content:
          multipart/form-data:
            schema:
              $ref: '#/components/schemas/cartForm'
      responses:
        '200':
          description: OK
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/cart'
            application/xml:
              schema:
                $ref: '#/components/schemas/cart'
        400:
          $ref: '#/components/responses/BadRequest'
        401:
          $ref: '#/components/responses/Unauthorized'
        403:
          $ref: '#/components/responses/Forbidden'
        404:
          $ref: '#/components/responses/NotFound'
        500:
          $ref: '#/components/responses/InternalServerError'

  /orders/{orderId}:
    get:
      summary: Get an order by id
      tags:
        - Order
```

```
    security:
      - cookieAuth: []
    parameters:
      - $ref: '#/components/parameters/OrderIdPathParam'
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/order'
          application/xml:
            schema:
              $ref: '#/components/schemas/order'
      400:
        $ref: '#/components/responses/BadRequest'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
/orders/customer/{userId}:
  get:
    summary: Get all orders by customer
    security:
      - cookieAuth: []
    description: Returns all the orders made by a specific customer.
    tags:
      - Order
    parameters:
      - $ref: '#/components/parameters/UserIdPathParam'
    responses:
      200:
        description: A list of all orders made by the customer
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/orders'
          application/xml:
            schema:
              $ref: '#/components/schemas/orders'
      400:
        $ref: '#/components/responses/BadRequest'
      404:
        $ref: '#/components/responses/NotFound'
```

```
      500:
        $ref: '#/components/responses/InternalServerError'
/orders/restaurant/{restaurantId}:
  get:
    summary: Get all orders by restaurant
    security:
      - cookieAuth: []
    description: Returns all the orders made to a specific restaurant.
    tags:
      - Order
    parameters:
      - $ref: '#/components/parameters/RestaurantIdPathParam'
    responses:
      '200':
        description: A list of all orders made to the restaurant
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/orders'
          application/xml:
            schema:
              $ref: '#/components/schemas/orders'
      400:
        $ref: '#/components/responses/BadRequest'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
/orders:
  post:
    summary: Create a new order for a customer
    description: Creates a new order for a customer.
    tags:
      - Order
    requestBody:
      required: true
      content:
        multipart/form-data:
          schema:
            $ref: '#/components/schemas/orderForm'
        application/xml:
          schema:
            $ref: '#/components/schemas/orderForm'
    responses:
```

```
      '201':
        description: The created order
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/order'
          application/xml:
            schema:
              $ref: '#/components/schemas/order'
      400:
        $ref: '#/components/responses/BadRequest'
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
    security:
      - cookieAuth: []

/reviews/{reviewId}:
  get:
    summary: Retrieve information about a review
    tags:
      - Reviews
    parameters:
      - $ref: '#/components/parameters/ReviewIdPathParam'
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/review'
          application/xml:
            schema:
              $ref: '#/components/schemas/review'
        headers:
          Cache-Control:
            description: Specifies caching behavior
            schema:
              type: string
```

```
            example: public, max-age=3600
      400:
        $ref: '#/components/responses/BadRequest'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
/reviews/restaurant/{restaurantId}:
  get:
    summary: Get all reviews for a restaurant
    tags:
      - Reviews
    parameters:
      - $ref: '#/components/parameters/RestaurantIdPathParam'
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/reviews'
          application/xml:
            schema:
              $ref: '#/components/schemas/reviews'
        headers:
          Cache-Control:
            description: Specifies caching behavior
            schema:
              type: string
            example: public, max-age=3600
      400:
        $ref: '#/components/responses/BadRequest'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
/reviews:
  post:
    summary: Create a new review
    operationId: createReview
    security:
      - cookieAuth: []
    tags:
      - Reviews
```

```
    requestBody:
      description: Review object that needs to be added
      required: true
      content:
        multipart/form-data:
          schema:
            $ref: "#/components/schemas/reviewForm"
        application/xml:
          schema:
            $ref: "#/components/schemas/reviewForm"
    responses:
      '200':
        description: Review created successfully
        content:
          application/json:
            schema:
              $ref: "#/components/schemas/review"
          application/xml:
            schema:
              $ref: "#/components/schemas/review"
      '400':
        description: Invalid input, object invalid
      '401':
        description: Unauthorized request
      '500':
        description: Internal server error
  /reviews/{reviewId}/:
    put:
      summary: Actualizar una review
      tags:
        - Reviews
      security:
        - cookieAuth: []
      parameters:
        - $ref: '#/components/parameters/ReviewIdPathParam'
      requestBody:
        required: true
        content:
          multipart/form-data:
            schema:
              $ref: '#/components/schemas/reviewForm'
      responses:
        '200':
          description: Review actualizada correctamente
```

```
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/review'
          application/xml:
            schema:
              $ref: '#/components/schemas/review'
      '400':
        $ref: '#/components/responses/BadRequest'
      '401':
        $ref: '#/components/responses/Unauthorized'
      '404':
        $ref: '#/components/responses/NotFound'
      '500':
        $ref: '#/components/responses/InternalServerError'

/categories:
  get:
    summary: Get all categories
    description: Returns all the categories.
    tags:
      - Category
    responses:
      '200':
        description: A list of all categories
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/categoriesNoPagination'
          application/xml:
            schema:
              $ref: '#/components/schemas/categoriesNoPagination'
        headers:
          Cache-Control:
            description: Specifies caching behavior
            schema:
              type: string
            example: public, max-age=3600
      400:
        $ref: '#/components/responses/BadRequest'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
```

```
/categories/{categoryId}:
  get:
    summary: Get a category by id
    tags:
      - Category
    parameters:
      - $ref: '#/components/parameters/CategoryIdPathParam'
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/category'
          application/xml:
            schema:
              $ref: '#/components/schemas/category'
      400:
        $ref: '#/components/responses/BadRequest'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
/categories/create:
  post:
    summary: Create a new category
    description: Creates a new category.
    tags:
      - Category
    requestBody:
      required: true
      content:
        multipart/form-data:
          schema:
            $ref: '#/components/schemas/categoryForm'
        application/xml:
          schema:
            $ref: '#/components/schemas/categoryForm'
    responses:
      '201':
        description: The created category
        content:
          application/json:
            schema:
```

```
            $ref: '#/components/schemas/category'
        application/xml:
          schema:
            $ref: '#/components/schemas/category'
    400:
      $ref: '#/components/responses/BadRequest'
    401:
      $ref: '#/components/responses/Unauthorized'
    403:
      $ref: '#/components/responses/Forbidden'
    404:
      $ref: '#/components/responses/NotFound'
    500:
      $ref: '#/components/responses/InternalServerError'
  security:
    - cookieAuth: []
/categories/edit/{categoryId}:
  put:
    summary: Edit an existing category
    description: Use this endpoint to edit an existing category.
    tags:
      - Category
    security:
      - cookieAuth: []
    parameters:
      - $ref: '#/components/parameters/CategoryIdPathParam'
    requestBody:
      required: true
      content:
        multipart/form-data:
          schema:
            $ref: '#/components/schemas/categoryForm'
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/category'
          application/xml:
            schema:
              $ref: '#/components/schemas/category'
      400:
        $ref: '#/components/responses/BadRequest'
```

```
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'

/subcategories:
  get:
    summary: Get all subcategories
    description: Returns all the subcategories.
    tags:
      - Subcategory
    parameters:
      - $ref: '#/components/parameters/Page'
      - $ref: '#/components/parameters/Limit'
    responses:
      '200':
        description: A list of all subcategories
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/subcategories'
          application/xml:
            schema:
              $ref: '#/components/schemas/subcategories'
        headers:
          Cache-Control:
            description: Specifies caching behavior
            schema:
              type: string
            example: public, max-age=3600
      400:
        $ref: '#/components/responses/BadRequest'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
/subcategories/{categoryId}:
  get:
    summary: Get all subcategories belonging to a certain category
    tags:
```

```
    - Subcategory
  parameters:
    - $ref: '#/components/parameters/CategoryIdPathParam'
  responses:
    '200':
      description: OK
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/subcategory'
        application/xml:
          schema:
            $ref: '#/components/schemas/subcategory'
    400:
      $ref: '#/components/responses/BadRequest'
    404:
      $ref: '#/components/responses/NotFound'
    500:
      $ref: '#/components/responses/InternalServerError'
/subcategories/{subcategoryId}:
  get:
    summary: Get a subcategory by id
    tags:
      - Subcategory
    parameters:
      - $ref: '#/components/parameters/SubcategoryIdPathParam'
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/subcategory'
          application/xml:
            schema:
              $ref: '#/components/schemas/subcategory'
      400:
        $ref: '#/components/responses/BadRequest'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
/subcategories/:
  post:
```

```
    summary: Create a new subcategory
    description: Creates a new subcategory.
    tags:
      - Subcategory
    requestBody:
      required: true
      content:
        multipart/form-data:
          schema:
            $ref: '#/components/schemas/subcategoryForm'
    responses:
      '201':
        description: The created subcategory
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/subcategory'
          application/xml:
            schema:
              $ref: '#/components/schemas/subcategory'
      400:
        $ref: '#/components/responses/BadRequest'
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
    security:
      - cookieAuth: []
/subcategories/{subcategoryId}/:
  put:
    summary: Edit an existing subcategory
    description: Use this endpoint to edit an existing subcategory.
    tags:
      - Subcategory
    security:
      - cookieAuth: []
    parameters:
      - $ref: '#/components/parameters/SubcategoryIdPathParam'
    requestBody:
      required: true
```

```
      content:
        multipart/form-data:
          schema:
            $ref: '#/components/schemas/subcategoryForm'
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/subcategory'
          application/xml:
            schema:
              $ref: '#/components/schemas/subcategory'
      400:
        $ref: '#/components/responses/BadRequest'
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'

/restaurants:
  get:
    summary: Retrieve all restaurants
    description: Retrieve a list of all restaurants
    tags:
      - Restaurant
    parameters:
      - $ref: '#/components/parameters/Page'
      - $ref: '#/components/parameters/Limit'
    responses:
      '200':
        description: A list of restaurants
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/restaurants'
          application/xml:
            schema:
              $ref: '#/components/schemas/restaurants'
```

```
      400:
        $ref: '#/components/responses/BadRequest'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
/restaurants/{restaurantId}:
  get:
    summary: Retrieve a restaurant by ID
    description: Retrieve a restaurant by its ID
    tags:
      - Restaurant
    parameters:
      - $ref: '#/components/parameters/RestaurantIdPathParam'
    responses:
      '200':
        description: The restaurant
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/restaurant'
          application/xml:
            schema:
              $ref: '#/components/schemas/restaurant'
      400:
        $ref: '#/components/responses/BadRequest'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
/restaurants/:
  post:
    summary: Create a new restaurant
    description: Restaurant object to be createduhu
    tags:
      - Restaurant
    requestBody:
      required: true
      content:
        multipart/form-data:
          schema:
            $ref: '#/components/schemas/restaurantForm'
        application/xml:
          schema:
```

```
            $ref: '#/components/schemas/restaurantForm'
      responses:
        '201':
          description: Restaurant created successfully
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/restaurant'
            application/xml:
              schema:
                $ref: '#/components/schemas/restaurant'
        400:
          $ref: '#/components/responses/BadRequest'
        401:
          $ref: '#/components/responses/Unauthorized'
        403:
          $ref: '#/components/responses/Forbidden'
        404:
          $ref: '#/components/responses/NotFound'
        500:
          $ref: '#/components/responses/InternalServerError'
      security:
        - cookieAuth: []
  /restaurants/{restaurantId}/:
    put:
      tags:
        - Restaurant
      summary: Updates an existing restaurant
      security:
        - cookieAuth: []
      operationId: updateRestaurant
      parameters:
        - $ref: '#/components/parameters/RestaurantIdPathParam'
      requestBody:
        required: true
        content:
          multipart/form-data:
            schema:
              $ref: '#/components/schemas/restaurantForm'
      responses:
        200:
          description: Restaurant updated successfully
          content:
            application/json:
```

```
            schema:
              $ref: '#/components/schemas/restaurant'
          application/xml:
            schema:
              $ref: '#/components/schemas/restaurant'
      400:
        $ref: '#/components/responses/BadRequest'
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
/restaurants/delete/{restaurantId}:
  delete:
    summary: Delete a restaurant. Just for admin role
    tags:
      - Restaurant
    parameters:
      - $ref: '#/components/parameters/RestaurantIdPathParam'
    responses:
      204:
        $ref: '#/components/responses/RemovedSuccessful'
      400:
        $ref: '#/components/responses/BadRequest'
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
    security:
      - cookieAuth: []

/users:
  get:
    summary: Get all users. Available only for the admin role
    description: Returns all users. .
    tags:
      - User
```

```
    security:
      - cookieAuth: []
    parameters:
      - $ref: '#/components/parameters/Page'
      - $ref: '#/components/parameters/Limit'
    responses:
      '200':
        description: A list of all users
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/users'
          application/xml:
            schema:
              $ref: '#/components/schemas/users'
      400:
        $ref: '#/components/responses/BadRequest'
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
  /users/{role}:
    get:
      summary: Get list of users by role. Available only for the admin role
      security:
        - cookieAuth: []
      tags:
        - User
      parameters:
        - $ref: '#/components/parameters/RolePathParam'
        - $ref: '#/components/parameters/Page'
        - $ref: '#/components/parameters/Limit'
      responses:
        '200':
          description: OK
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/user'
            application/xml:
```

```
            schema:
              $ref: '#/components/schemas/user'
      400:
        $ref: '#/components/responses/BadRequest'
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
/users/{userId}:
  get:
    summary: Get a user by id
    security:
      - cookieAuth: []
    tags:
      - User
    parameters:
      - $ref: '#/components/parameters/UserIdPathParam'
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/user'
          application/xml:
            schema:
              $ref: '#/components/schemas/user'
      400:
        $ref: '#/components/responses/BadRequest'
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
/users/:
  post:
    summary: Create a new user
```

```
    description: Creates a new user.
    tags:
      - User
    requestBody:
      required: true
      content:
        multipart/form-data:
          schema:
            $ref: '#/components/schemas/userForm'
        application/xml:
          schema:
            $ref: '#/components/schemas/userForm'
    responses:
      '201':
        description: The created user
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/user'
          application/xml:
            schema:
              $ref: '#/components/schemas/user'
      400:
        $ref: '#/components/responses/BadRequest'
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
    security:
      - cookieAuth: []
  /users/{userId}/:
    put:
      summary: Edit an existing user
      description: Use this endpoint to edit an existing user.
      tags:
        - User
      security:
        - cookieAuth: []
      parameters:
        - $ref: '#/components/parameters/UserIdPathParam'
```

```
    requestBody:
      required: true
      content:
        multipart/form-data:
          schema:
            $ref: '#/components/schemas/userForm'
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/user'
          application/xml:
            schema:
              $ref: '#/components/schemas/user'
      400:
        $ref: '#/components/responses/BadRequest'
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
  /users/delete/{userId}:
    delete:
      summary: Delete an existing user. Just for admin role
      description: Use this endpoint to delete an existing user.
      tags:
        - User
      security:
        - cookieAuth: []
      parameters:
        - $ref: '#/components/parameters/UserIdPathParam'
      responses:
        204:
          $ref: '#/components/responses/RemovedSuccessful'
        400:
          $ref: '#/components/responses/BadRequest'
        401:
          $ref: '#/components/responses/Unauthorized'
        403:
```

```
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'

/deliveries:
  get:
    summary: Get all deliveries. Available only for administratives roles
    description: Returns all the deliveries.
    security:
      - cookieAuth: []
    parameters:
      - $ref: '#/components/parameters/Page'
      - $ref: '#/components/parameters/Limit'
    tags:
      - Delivery
    responses:
      '200':
        description: A list of all deliveries
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/deliveries'
          application/xml:
            schema:
              $ref: '#/components/schemas/deliveries'
      400:
        $ref: '#/components/responses/BadRequest'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
/deliveries/{deliveryId}:
  get:
    summary: Get a delivery by id
    security:
      - cookieAuth: []
    tags:
      - Delivery
    parameters:
      - $ref: '#/components/parameters/DeliveryIdPathParam'
    responses:
      '200':
```

```
        description: OK
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/delivery'
          application/xml:
            schema:
              $ref: '#/components/schemas/delivery'
      400:
        $ref: '#/components/responses/BadRequest'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
/deliveries/:
  post:
    summary: Create a new delivery
    description: Creates a new delivery.
    tags:
      - Delivery
    requestBody:
      required: true
      content:
        multipart/form-data:
          schema:
            $ref: '#/components/schemas/deliveryForm'
        application/xml:
          schema:
            $ref: '#/components/schemas/deliveryForm'
    responses:
      '201':
        description: The created delivery
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/delivery'
          application/xml:
            schema:
              $ref: '#/components/schemas/delivery'
      400:
        $ref: '#/components/responses/BadRequest'
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
```

```
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
    security:
      - cookieAuth: []
/deliveries/{deliveryId}/:
  put:
    summary: Edit an existing delivery
    description: Use this endpoint to edit an existing delivery.
    tags:
      - Delivery
    security:
      - cookieAuth: []
    parameters:
      - $ref: '#/components/parameters/DeliveryIdPathParam'
    requestBody:
      required: true
      content:
        multipart/form-data:
          schema:
            $ref: '#/components/schemas/deliveryForm'
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/delivery'
          application/xml:
            schema:
              $ref: '#/components/schemas/delivery'
      400:
        $ref: '#/components/responses/BadRequest'
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
/deliveries/delete/{deliveryId}:
```

```
  delete:
    summary: Delete an existing delivery. Just for admin role
    description: Use this endpoint to delete an existing delivery.
    tags:
      - Delivery
    security:
      - cookieAuth: []
    parameters:
      - $ref: '#/components/parameters/DeliveryIdPathParam'
    responses:
      204:
        $ref: '#/components/responses/RemovedSuccessful'
      400:
        $ref: '#/components/responses/BadRequest'
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'

/deposit-accounts:
  get:
    summary: Get deposit accounts for the authenticated user or admin role users
    description: Returns all deposit accounts belonging to a user.
    security:
      - BearerAuth: []
    tags:
      - Deposit_account
    responses:
      '200':
        description: A list of deposit accounts belonging to the user
        content:
          application/json:
            schema:
              type: array
              items:
                $ref: '#/components/schemas/depositAccounts'
          application/xml:
            schema:
              type: array
              items:
```

```
              $ref: '#/components/schemas/depositAccounts'
            xml:
              name: 'deposit_account_list'
      400:
        $ref: '#/components/responses/BadRequest'
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
/deposit_accounts/{depositAccountId}:
  get:
    summary: Get a deposit account by id
    security:
      - cookieAuth: []
    tags:
      - Deposit_account
    parameters:
      - $ref: '#/components/parameters/DepositPathParam'
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/deposit_account'
          application/xml:
            schema:
              $ref: '#/components/schemas/deposit_account'
      400:
        $ref: '#/components/responses/BadRequest'
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
/deposit_accounts/:
  post:
```

```
    summary: Create a new deposit account
    description: Creates a new deposit account.
    tags:
      - Deposit_account
    requestBody:
      required: true
      content:
        multipart/form-data:
          schema:
            $ref: '#/components/schemas/depositAccountForm'
        application/xml:
          schema:
            $ref: '#/components/schemas/depositAccountForm'
    responses:
      '201':
        description: The created deposit account
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/deposit_account'
          application/xml:
            schema:
              $ref: '#/components/schemas/deposit_account'
      400:
        $ref: '#/components/responses/BadRequest'
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
    security:
      - cookieAuth: []
  /deposit_accounts/{depositAccountId}/:
    put:
      summary: Edit an existing deposit account
      description: Use this endpoint to edit an existing deposit account.
      tags:
        - Deposit_account
      security:
        - cookieAuth: []
      parameters:
```

```
    - $ref: '#/components/parameters/DepositPathParam'
  requestBody:
    required: true
    content:
      multipart/form-data:
        schema:
          $ref: '#/components/schemas/depositAccountForm'
  responses:
    '200':
      description: OK
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/deposit_account'
        application/xml:
          schema:
            $ref: '#/components/schemas/deposit_account'
    400:
      $ref: '#/components/responses/BadRequest'
    401:
      $ref: '#/components/responses/Unauthorized'
    403:
      $ref: '#/components/responses/Forbidden'
    404:
      $ref: '#/components/responses/NotFound'
    500:
      $ref: '#/components/responses/InternalServerError'
/deposit_accounts/delete/{depositAccountId}:
  delete:
    summary: Delete an existing deposit account
    description: Use this endpoint to delete an existing deposit account.
    tags:
      - Deposit_account
    security:
      - cookieAuth: []
    parameters:
      - $ref: '#/components/parameters/DepositPathParam'
    responses:
      204:
        $ref: '#/components/responses/RemovedSuccessful'
      400:
        $ref: '#/components/responses/BadRequest'
      401:
        $ref: '#/components/responses/Unauthorized'
```

```
        403:
          $ref: '#/components/responses/Forbidden'
        404:
          $ref: '#/components/responses/NotFound'
        500:
          $ref: '#/components/responses/InternalServerError'

  /payment_methods/{paymentMethodId}:
    get:
      summary: Get a payment method by ID
      security:
        - cookieAuth: []
      description: Returns a payment method with the given ID.
      tags:
        - Payment_method
      parameters:
        - $ref: '#/components/parameters/PaymentMethodIdPathParam'
      responses:
        '200':
          description: OK
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/payment_method'
            application/xml:
              schema:
                $ref: '#/components/schemas/payment_method'
        400:
          $ref: '#/components/responses/BadRequest'
        401:
          $ref: '#/components/responses/Unauthorized'
        403:
          $ref: '#/components/responses/Forbidden'
        404:
          $ref: '#/components/responses/NotFound'
        500:
          $ref: '#/components/responses/InternalServerError'
  /payment_methods:
    get:
      summary: Get all payment methods for the authenticated user or admin role
users
      description: Returns all payment methods associated with a user.
      tags:
        - Payment_method
```

```
    security:
      - BearerAuth: []
    responses:
      '200':
        description: A list of all payment methods for the authenticated user
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/payment_methodsNoPagination'
          application/xml:
            schema:
              $ref: '#/components/schemas/payment_methodsNoPagination'
      400:
        $ref: '#/components/responses/BadRequest'
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
  /payment_methods/:
    post:
      summary: Create a new payment method
      description: Use this endpoint to create a new payment method.
      tags:
        - Payment_method
      security:
        - cookieAuth: []
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/payment_methodForm'
          multipart/form-data:
            schema:
              $ref: '#/components/schemas/payment_methodForm'
      responses:
        '201':
          description: Created
          headers:
            Location:
```

```
            schema:
              type: string
            description: The URI of the created payment method
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/payment_method'
          application/xml:
            schema:
              $ref: '#/components/schemas/payment_method'
      400:
        $ref: '#/components/responses/BadRequest'
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
        $ref: '#/components/responses/Forbidden'
      500:
        $ref: '#/components/responses/InternalServerError'
/payment_methods/{paymentMethodId}/:
  put:
    summary: Edit an existing payment method
    description: Use this endpoint to edit an existing payment method.
    tags:
      - Payment_method
    security:
      - BearerAuth: []
    parameters:
      - $ref: '#/components/parameters/PaymentMethodIdPathParam'
    requestBody:
      required: true
      content:
        multipart/form-data:
          schema:
            $ref: '#/components/schemas/payment_methodForm'
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/payment_method'
          application/xml:
            schema:
              $ref: '#/components/schemas/payment_method'
```

```
      400:
        $ref: '#/components/responses/BadRequest'
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
/payment_methods/delete/{paymentMethodId}:
  delete:
    summary: Delete a payment method by id
    security:
      - BearerAuth: []
    tags:
      - Payment_method
    parameters:
      - $ref: '#/components/parameters/PaymentMethodIdPathParam'
    responses:
      204:
        $ref: '#/components/responses/RemovedSuccessful'
      400:
        $ref: '#/components/responses/BadRequest'
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'

/payment/pay:
  post:
    description: "create payment attempt"
    tags:
      - Payment
    security:
      - cookieAuth: []
    responses:
      '200':
        description: OK
        content:
```

```
          application/json:
            schema:
              type: object
              properties:
                message:
                  type: string
                  example: There is a cart to pay.
          application/xml:
            schema:
              type: object
              properties:
                message:
                  type: string
                  example: There is a cart to pay.
      400:
        $ref: '#/components/responses/BadRequest'
      401:
        $ref: '#/components/responses/Unauthorized'
      403:
        $ref: '#/components/responses/Forbidden'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'

/auth/login/:
    post:
      description: "User Registration Route"
      tags:
        - Auth
      requestBody:
        description: "User registration in the system"
        required: true
        content:
          multipart/form-data:
            schema:
              $ref: "#/components/schemas/loginForm"
      responses:
        '200':
          description: A list of all users
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/users'
```

```
          application/xml:
            schema:
              $ref: '#/components/schemas/users'
      400:
        $ref: '#/components/responses/BadRequest'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
/auth/signup/:
  post:
    summary: User Registration Route
    description: Creates a new user.
    tags:
      - Auth
    requestBody:
      required: true
      content:
        multipart/form-data:
          schema:
            $ref: '#/components/schemas/userForm'
    responses:
      '200':
        description: A list of all users
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/users'
          application/xml:
            schema:
              $ref: '#/components/schemas/users'
      400:
        $ref: '#/components/responses/BadRequest'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'
/auth/logout/:
  post:
    summary: User disconnection in the system, session end
    description: Creates a new user.
    security:
      - cookieAuth: []
    tags:
```

```
      - Auth
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              type: object
              properties:
                message:
                  type: string
                  example: Successful system shutdown.
          application/xml:
            schema:
              type: object
              properties:
                message:
                  type: string
                  example: Successful system shutdown.
      400:
        $ref: '#/components/responses/BadRequest'
      404:
        $ref: '#/components/responses/NotFound'
      500:
        $ref: '#/components/responses/InternalServerError'


components:
  schemas:
    dish:
      type: object
      properties:
        id:
          type: string
          description: The unique identifier for the dish.
          example: d290f1ee-6c54-4b01-90e6-d701748f0851
          xml:
            attribute: true
        name:
          type: string
          description: The name of the dish.
          example: Spaghetti Bolognese
        description:
          type: string
```

```
      description: A short description of the dish.
      example: Delicious spaghetti with rich tomato sauce and minced beef.
    ingredients:
      type: string
      description: A list of ingredients used in the dish.
      example: Spaghetti, tomato sauce, minced beef, onion, garlic, olive oil, salt,
pepper.
    price:
      type: number
      description: The price of the dish in the restaurant's currency.
      example: 12.99
    amount:
      type: number
      description: The quantity of the dish.
      example: 1
    discount:
      type: number
      description: The discount applied to the dish.
      example: 5.00
    quantity:
      type: integer
      description: The number of dishes available.
      example: 20
    size:
      type: string
      description: The size of the dish.
      example: Large
    sold_units:
      type: integer
      description: The number of dishes sold.
      example: 50
    status:
      type: string
      description: The current status of the dish.
      example: Available
    created_at:
      type: string
      format: date-time
      description: The date and time the dish was created.
      example: '2022-02-18T15:02:31Z'
    last_update:
      type: string
      format: date-time
      description: The date and time the dish was last updated.
```

```
        example: '2022-02-18T15:02:31Z'
      atributes:
        type: object
        description: Dish attributes
        xml:
          wrapped : true
          name: attributes
        example:
          - 'gluten-free'
          - 'vegetarian'
      category:
        type: string
        description: The category this dish belongs to.
        example: https://delivery_food.ue.r.appspot.com/api/v1/categories/j38huhu83h
      subcategory:
        type: string
        description: The subcategory this dish belongs to.
        example:
https://delivery_food.ue.r.appspot.com/api/v1/subcategories/3ie883hu8
      restaurant:
        type: string
        description: The restaurant that offers this dish.
        example:
https://delivery_food.ue.r.appspot.com/api/v1/restaurants/joi2u3h7378y7
    xml:
      name: 'dish'
  dishes:
    type: object
    properties:
      results:
        type: array
        items:
          $ref: '#/components/schemas/dish'
      total:
        type: integer
        description: The total number of dishes available.
        example: 100
      limit:
        type: integer
        description: The maximum number of dishes returned per page.
        example: 10
      offset:
        type: integer
```

```
        description: The number of dishes to skip before starting to collect the result
set.
        example: 0
      has_more:
        type: boolean
        description: Indicates whether there are more dishes available.
        example: true
    xml:
      name: 'dish_list'
      wrapped: false
  dishForm:
    type: object
    properties:
      name:
        type: string
      description:
        type: string
      ingredients:
        type: string
      price:
        type: number
      quantity:
        type: integer
      discount:
        type: number
      size:
        type: string
      attributes:
        type: array
        items:
          type: string
      subcategory_id:
        type: integer
      restaurant_id:
        type: integer
    required:
      - name
      - description
      - ingredients
      - price
      - quantity
      - size
      - subcategory_id
      - restaurant_id
```

```
    review:
      type: object
      properties:
        id:
          type: string
          example: "gygy6f7887yhh6ff443dg$"
        emisor:
          type: string
          description: The person who creates the review.
          example: https://delivery_food.ue.r.appspot.com/api/v1/users/shh38sh6d2
        receiver:
          type: string
          description: The restaurant that recives the review.
          example:
https://delivery_food.ue.r.appspot.com/api/v1/restaurants/3ju4h77whs
        emisor_role:
          type: integer
          example: 1
        receiver_role:
          type: integer
          example: 2
        rating:
          type: integer
          example: 4
        content:
          type: string
          example: "Great food and service!"
        status:
          type: integer
          example: 1
        created_at:
          type: string
          format: date-time
          example: "2022-02-21T12:34:56.000Z"
        last_update:
          type: string
          format: date-time
          example: "2022-02-21T12:34:56.000Z"
      xml:
        name: 'review'
    reviews:
      type: object
      properties:
```

```
        reviews:
          type: array
          items:
            $ref: '#/components/schemas/review'
        total:
          type: integer
          description: The total number of dishes available.
          example: 100
        limit:
          type: integer
          description: The maximum number of dishes returned per page.
          example: 10
        offset:
          type: integer
          description: The number of dishes to skip before starting to collect the result
set.
          example: 0
        has_more:
          type: boolean
          description: Indicates whether there are more dishes available.
          example: true
      xml:
        name: 'review-list'
    reviewForm:
      type: object
      properties:
        rating:
          type: integer
          example: 4
        content:
          type: string
          example: "Great food and service!"
      required:
        - rating
        - content

    category:
      type: object
      properties:
        id:
          type: string
          description: The unique identifier for the category.
          example: c290f1ee-6c54-4b01-90e6-d701748f0851
          xml:
```

```
          attribute: true
        name:
          type: string
          description: The name of the category.
          example: Mexican
        description:
          type: string
          description: A description of the category.
          example: Mexico typical food
        image:
          type: string
          description: The URL of the category image.
          example:
https://delivery_food.ue.r.appspot.com/api/v1/resources/images/appetizers.jpg
        created_at:
          type: string
          format: date-time
          description: The date and time the category was created.
          example: '2022-02-18T15:02:31Z'
        last_update:
          type: string
          format: date-time
          description: The date and time the category was last updated.
          example: '2022-02-18T15:02:31Z'
      xml:
        name: 'category'
    categoriesNoPagination:
      type: object
      properties:
        results:
          type: array
          items:
            $ref: '#/components/schemas/category'
      xml:
        name: 'category_list'
        wrapped: false
    categoryForm:
      type: object
      properties:
        name:
          type: string
        description:
          type: string
        image:
```

```
        type: string
      required:
        - name

    subcategory:
      type: object
      properties:
        id:
          type: string
          description: The unique identifier for the subcategory.
          example: s290f1ed701748f0851
          xml:
            attribute: true
        name:
          type: string
          description: The name of the subcategory.
          example: Spaghetti
        description:
          type: string
          description: A short description of the subcategory.
          example: A type of pasta that is long and thin.
        image:
          type: string
          description: The image URL of the subcategory.
          example:
https://delivery_food.ue.r.appspot.com/api/v1/resources/images/subcategories/spagh
etti.jpg
        category:
          type: string
          description: The category this subcategory belongs to.
          example: https://delivery_food.ue.r.appspot.com/api/v1/categories/hu2huh399
      xml:
        name: 'subcategory'
    subcategories:
      type: object
      properties:
        results:
          type: array
          items:
            $ref: '#/components/schemas/subcategory'
      xml:
        name: 'subcategory_list'
        wrapped: false
    subcategoryForm:
```

```
      type: object
      properties:
        name:
          type: string
        description:
          type: string
        image:
          type: string
        category_id:
          type: string
      required:
        - name
        - category_id

    restaurant:
      type: object
      properties:
        id:
          type: string
          description: The unique identifier for the restaurant.
          example: d290f1ee-6c54-4b01-90e6-d701748f0851
          xml:
            attribute: true
        created_by:
          type: string
          description: The user who created the restaurant.
          example:
https://delivery_food.ue.r.appspot.com/api/v1/users/d290f1ee-6c54-4b01-90e6-d701
748f0851
        name:
          type: string
          description: The name of the restaurant.
          example: The Great Italian
        history:
          type: string
          description: A brief history of the restaurant.
          example: Established in 1975, The Great Italian has been serving up delicious
Italian cuisine for over 40 years
        photo:
          type: string
          description: The URL of the main photo for the restaurant.
          example:
https://delivery_food.ue.r.appspot.com/api/v1/resources/images/restaurant/photo.jpg
        logo:
```

```
        type: string
        description: The URL of the logo for the restaurant.
        example:
https://delivery_food.ue.r.appspot.com/api/v1/resources/images/restaurant/logo.jpg
      email:
        type: string
        description: The email address of the restaurant.
        example: info@thegreatitalian.com
      delivery_blocked:
        type: boolean
        description: Indicates whether delivery is currently blocked for the restaurant.
        example: false
      status:
        type: integer
        description: The current status of the restaurant.
        example: 1
      created_at:
        type: string
        format: date-time
        description: The date and time the restaurant was created.
        example: '2022-02-18T15:02:31Z'
      last_update:
        type: string
        format: date-time
        description: The date and time the restaurant was last updated.
        example: '2022-02-18T15:02:31Z'
    required:
      - id
      - created_by
      - name
      - history
  restaurants:
    type: object
    properties:
      restaurants:
        type: array
        items:
          $ref: '#/components/schemas/restaurant'
      total:
        type: integer
        description: The total number of dishes available.
        example: 100
      limit:
        type: integer
```

```
        description: The maximum number of dishes returned per page.
        example: 10
      offset:
        type: integer
        description: The number of dishes to skip before starting to collect the result
set.
        example: 0
      has_more:
        type: boolean
        description: Indicates whether there are more dishes available.
        example: true
  restaurantForm:
    type: object
    properties:
      name:
        type: string
      history:
        type: string
      photo:
        type: string
        format: binary
      logo:
        type: string
        format: binary
      email:
        type: string
        format: email
      delivery_blocked:
        type: boolean
      status:
        type: integer
        enum: [0, 1, 2]
      created_by:
        type: string
      created_at:
        type: string
        format: date-time
      last_update:
        type: string
        format: date-time
    required:
      - name
      - history
      - email
```

```
    user:
      type: object
      properties:
        id:
          type: string
          description: The unique identifier for the user.
          example: 7c9e6679-7425-40de-944b-e07fc1f90ae7
          xml:
            attribute: true
        username:
          type: string
          description: The username of the user.
          example: johndoe
        plan:
          type: string
          description: The plan identifier of the user.
          example: https://delivery_food.ue.r.appspot.com/api/v1/plans/23hiwehh88
        password:
          type: string
          description: The password of the user.
          example:
$2a$10$ymTrvL8ZRwWV7iS1JjK.D.NvI8LKzaW7xdz4jbg71J7VzvI8wAp7S
        name:
          type: string
          description: The name of the user.
          example: John
        last_name:
          type: string
          description: The last name of the user.
          example: Doe
        email:
          type: string
          description: The email address of the user.
          example: johndoe@example.com
        role:
          type: integer
          description: The role of the user.
          example: 1
        status:
          type: integer
          description: The status of the user.
          example: 1
        created_at:
```

```
        type: string
        format: date-time
        description: The date and time the user was created.
        example: '2022-02-18T15:02:31Z'
      last_update:
        type: string
        format: date-time
        description: The date and time the user was last updated.
        example: '2022-02-18T15:02:31Z'
      DNI:
        type: string
        description: The DNI of the user.
        example: 12345678A
      food_preferences:
        type: string
        description: The food preferences of the user.
        example: vegan, fresh-ingredients
      food_restrictions:
        type: string
        description: The food restrictions of the user.
        example: spicy-food, meat
    xml:
      name: 'user'
  users:
    type: object
    properties:
      results:
        type: array
        items:
          $ref: '#/components/schemas/user'
      total:
        type: integer
        description: The total number of users available.
        example: 100
      limit:
        type: integer
        description: The maximum number of users returned per page.
        example: 10
      offset:
        type: integer
        description: The number of users to skip before starting to collect the result
set.
        example: 0
      has_more:
```

```
      type: boolean
      description: Indicates whether there are more users available.
      example: true
  xml:
    name: 'user_list'
    wrapped: false
userForm:
  type: object
  properties:
    username:
      type: string
    password:
      type: string
    name:
      type: string
    last_name:
      type: string
    email:
      type: string
    role:
      type: integer
    status:
      type: integer
    DNI:
      type: string
    food_preferences:
      type: string
    food_restrictions:
      type: string
  required:
    - username
    - password
    - name
    - last_name
    - email
loginForm:
  type: object
  properties:
    username:
      type: string
    password:
      type: string

delivery:
```

```
    type: object
    properties:
      id:
        type: string
        description: The unique identifier for the delivery.
        example: d290f1ee-6c54-4b01-90e6-d701748f0851
        xml:
          attribute: true
      delivery_person:
        type: string
        description: The delivery person.
        example:
https://delivery_food.ue.r.appspot.com/api/v1/users/d290f1ee-6c54-4b01-90e6-d701
748f0851
        xml:
          attribute: true
      started_at:
        type: string
        format: date-time
        description: The date and time the delivery was started.
        example: '2022-02-20T15:02:31Z'
      finished_at:
        type: string
        format: date-time
        description: The date and time the delivery was finished.
        example: '2022-02-20T15:32:31Z'
      price:
        type: number
        description: The price of the delivery in the restaurant's currency.
        example: 10.99
      coordinates:
        type: string
        description: The coordinates of the delivery location.
        example: '41.40338, 2.17403'
    xml:
      name: 'delivery'
  deliveries:
    type: object
    properties:
      results:
        type: array
        items:
          $ref: '#/components/schemas/delivery'
      total:
```

```
          type: integer
          description: The total number of deliveries available.
          example: 50
        limit:
          type: integer
          description: The maximum number of deliveries returned per page.
          example: 10
        offset:
          type: integer
          description: The number of deliveries to skip before starting to collect the
result set.
          example: 0
        has_more:
          type: boolean
          description: Indicates whether there are more deliveries available.
          example: true
      xml:
        name: 'delivery_list'
        wrapped: false
    deliveryForm:
      type: object
      properties:
        delivery_person_id:
          type: string
        started_at:
          type: string
          format: date-time
        finished_at:
          type: string
          format: date-time
        price:
          type: number
        coordinates:
          type: string
      required:
        - delivery_person_id
        - started_at
        - price
        - coordinates

    cart:
      type: object
      properties:
        customer:
```

```
        type: string
        description: The customer associated with the cart.
        example: https://delivery_food.ue.r.appspot.com/api/v1/users/8798why7
      status:
        type: integer
        description: The status of the cart.
        example: 2
      created_at:
        type: string
        format: date-time
        description: The date and time when the cart was created.
      last_update:
        type: string
        format: date-time
        description: The date and time when the cart was last updated.
  cartForm:
    type: object
    properties:
      customer_id:
        type: string
        description: The ID of the customer associated with the cart.
      status:
        type: integer
        description: The status of the cart.
    required:
      - customer_id
      - status

  order:
    type: object
    properties:
      id:
        type: string
        description: The unique identifier for the order.
        example: o290f1ee-6c54-4b01-90e6-d701748f0851
        xml:
          attribute: true
      cart:
        type: string
        description: The cart associated with the order.
        example:
https://delivery_food.ue.r.appspot.com/api/v1/cart/c290f1ee-6c54-4b01-90e6-d70174
8f0851
        xml:
```

```
          attribute: true
      payment_method:
        type: string
        description: The payment method used for the order.
        example: Credit Card
      delivery:
        type: string
        description: The delivery associated with the order.
        example:
https://delivery_food.ue.r.appspot.com/api/v1/deliveries/d290f1ee-6c54-4b01-90e6-d
701748f0851
        xml:
          attribute: true
      date:
        type: string
        format: date-time
        description: The date and time the order was placed.
        example: '2022-02-18T15:02:31Z'
      total:
        type: number
        description: The total cost of the order in the restaurant's currency.
        example: 25.99
      status:
        type: string
        description: The current status of the order.
        example: Processing
      dishes:
        type: array
        description: The list of dishes included in the order.
        items:
          type: object
          properties:
            link:
              type: string
              description: The link to the dish
              example:
https://delivery_food.ue.r.appspot.com/api/v1/dishes/p290f1ee-6c54-4b01-90e6-d70
1748f0851
            id:
              type: string
              description: The unique identifier for the product.
              example: p290f1ee-6c54-4b01-90e6-d701748f0851
              xml:
                attribute: true
```

```
            name:
              type: string
              description: The name of the product.
              example: Spaghetti Bolognese
            price:
              type: number
              description: The price of the product in the restaurant's currency.
              example: 12.99
            quantity:
              type: integer
              description: The quantity of the product in the order.
              example: 2
        xml:
          wrapped: false
          name: 'products'
    xml:
      name: 'order'
  orders:
    type: object
    properties:
      orders:
        type: array
        items:
          $ref: '#/components/schemas/order'
      total:
        type: integer
        description: The total number of orders.
        example: 100
      limit:
        type: integer
        description: The maximum number of orders returned per page.
        example: 10
      offset:
        type: integer
        description: The number of orders to skip before starting to collect the result
set.
        example: 0
      has_more:
        type: boolean
        description: Indicates whether there are more orders available.
        example: true
    required:
      - orders
  orderForm:
```

```
  type: object
  properties:
    cart_id:
      type: string
      description: The ID of the cart associated with the order.
    payment_method:
      type: string
      description: The payment method used for the order.
    delivery_id:
      type: string
      description: The ID of the delivery associated with the order.
    date:
      type: string
      format: date-time
      description: The date and time of the order.
    total:
      type: number
      format: decimal
      description: The total cost of the order.
    status:
      type: integer
      description: The status of the order.
    products:
      type: string
      description: The products in the order.
  required:
    - cart_id
    - payment_method
    - delivery_id
    - date
    - total
    - status
    - products

payment_method:
  type: object
  properties:
    id:
      type: string
      description: The unique identifier for the payment method.
      example: p290f1ee-6c54-4b01-90e6-d701748f0851
      xml:
        attribute: true
    name:
```

```
      type: string
      description: The name of the payment method.
      example: Credit card
    description:
      type: string
      description: A short description of the payment method.
      example: Payment with credit card.
    is_active:
      type: boolean
      description: Indicates whether the payment method is active or not.
      example: true
    created_at:
      type: string
      format: date-time
      description: The date and time the payment method was created.
      example: '2022-02-18T15:02:31Z'
    last_update:
      type: string
      format: date-time
      description: The date and time the payment method was last updated.
      example: '2022-02-18T15:02:31Z'
  xml:
    name: 'payment_method'
payment_methodsNoPagination:
  type: object
  properties:
    results:
      type: array
      items:
        $ref: '#/components/schemas/payment_method'
  xml:
    name: 'payment_method_list'
    wrapped: false
payment_methodForm:
  type: object
  properties:
    name:
      type: string
    description:
      type: string
    is_active:
      type: boolean
  required:
    - name
```

```
    - description

deposit_account:
  type: object
  properties:
    id:
      type: string
      description: The unique identifier for the deposit account.
      example: d290f1ee-6c54-4b01-90e6-d701748f0851
      xml:
        attribute: true
    user:
      type: string
      description: The user that owns the deposit account.
      example: https://delivery_food.ue.r.appspot.com/api/v1/users/88hhg6567
    number:
      type: integer
      description: The account number of the deposit account.
      example: 1234567890
    financial_entity:
      type: string
      description: The financial entity of the deposit account.
      example: Chase Bank
    name:
      type: string
      description: The name of the deposit account.
      example: My Checking Account
    created_at:
      type: string
      format: date-time
      description: The date and time the deposit account was created.
      example: '2022-02-18T15:02:31Z'
    last_update:
      type: string
      format: date-time
      description: The date and time the deposit account was last updated.
      example: '2022-02-18T15:02:31Z'
    type:
      type: string
      description: The type of the deposit account.
      example: Checking
  xml:
    name: 'deposit_account'
depositAccounts:
```

```
    type: object
    properties:
      results:
        type: array
        items:
          $ref: '#/components/schemas/deposit_account'
    xml:
      name: 'deposit_account_list'
      wrapped: false
  depositAccountForm:
    type: object
    properties:
      user_id:
        type: string
        description: The unique identifier for the user that owns the deposit account.
        example: d290f1ee-6c54-4b01-90e6-d701748f0851
      number:
        type: integer
        description: The account number of the deposit account.
        example: 1234567890
      financial_entity:
        type: string
        description: The financial entity of the deposit account.
        example: Chase Bank
      name:
        type: string
        description: The name of the deposit account.
        example: My Checking Account
      type:
        type: string
        description: The type of the deposit account.
        example: Checking
    required:
      - user_id
      - number
      - financial_entity
      - type
    xml:
      name: 'deposit_account_form'


  Unauthorized:
    type: object
    properties:
```

```
    message:
      type: string
      example: Unauthorized user
    code:
      type: integer
      example: 401
Forbidden:
  type: object
  properties:
    message:
      type: string
      example: Forbidden access
    code:
      type: integer
      example: 403
NotFound:
  type: object
  properties:
    message:
      type: string
      example: Resource not found
    code:
      type: integer
      example: 404
BadRequest:
  type: object
  properties:
    message:
      type: string
      example: Bad request
    code:
      type: integer
      example: 400
RemovedSuccessful:
  type: object
  properties:
    message:
      type: string
      example: Deleted successfully
    code:
      type: integer
      example: 204
InternalServerError:
  type: object
```

```
    properties:
      message:
        type: string
        example: Internal server error
      code:
        type: integer
        example: 500

securitySchemes:
  cookieAuth:
    type: apiKey
    in: cookie
    name: token

  BearerAuth:
    type: http
    scheme: bearer
    bearerFormat: JWT

parameters:
  Page:
    name: page
    in: query
    required: false
    description: The page number to retrieve (default is 1)
    schema:
      type: integer
      example: 2
  Limit:
    name: limit
    in: query
    required: false
    description: The number of items to retrieve per page (default is 10)
    schema:
      type: integer
      example: 20
  DishIdPathParam:
    name: slug
    in: path
    description: ID of the review to retrieve
    required: true
    schema:
      type: string
  ReviewIdPathParam:
```

```
    name: reviewId
    in: path
    description: ID of the review to retrieve
    required: true
    schema:
      type: string
  CategoryIdPathParam:
    name: categoryId
    in: path
    description: ID of the category to retrieve
    required: true
    schema:
      type: string
  SubcategoryIdPathParam:
    name: subcategoryId
    in: path
    description: ID of the subcategory to retrieve
    required: true
    schema:
      type: string
  RestaurantIdPathParam:
    name: restaurantId
    in: path
    description: ID of the restaurant to retrieve
    required: true
    schema:
      type: string
  UserIdPathParam:
    name: userId
    in: path
    description: ID of the user to retrieve
    required: true
    schema:
      type: string
  RolePathParam:
    name: role
    in: path
    description: role of the user to retrieve
    required: true
    schema:
      type: string
  DeliveryIdPathParam:
    name: deliveryId
    in: path
```

```
    description: ID of the delivery to retrieve
    required: true
    schema:
      type: string
  CartIdPathParam:
    name: cartId
    in: path
    description: ID of the cart to retrieve
    required: true
    schema:
      type: string
  OrderIdPathParam:
    name: orderId
    in: path
    description: ID of the order to retrieve
    required: true
    schema:
      type: string
  PaymentMethodIdPathParam:
    name: paymentMethodId
    in: path
    description: ID of the payment method
    required: true
    schema:
      type: string
  DepositPathParam:
    name: depositAccountId
    in: path
    description: ID of the deposit account to retrieve
    required: true
    schema:
      type: string

responses:
  RemovedSuccessful:
    description: deleted succesfully
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/RemovedSuccessful'
      application/xml:
        schema:
          $ref: '#/components/schemas/RemovedSuccessful'
  BadRequest:
```

```
  description: Bad Request
  content:
    application/json:
      schema:
        $ref: '#/components/schemas/BadRequest'
    application/xml:
      schema:
        $ref: '#/components/schemas/BadRequest'
Unauthorized:
  description: Unauthorized user
  content:
    application/json:
      schema:
        $ref: '#/components/schemas/Unauthorized'
    application/xml:
      schema:
        $ref: '#/components/schemas/Unauthorized'
Forbidden:
  description: Forbidden access
  content:
    application/json:
      schema:
        $ref: '#/components/schemas/Forbidden'
    application/xml:
      schema:
        $ref: '#/components/schemas/Forbidden'
NotFound:
  description: Resource not found
  content:
    application/json:
      schema:
        $ref: '#/components/schemas/NotFound'
    application/xml:
      schema:
        $ref: '#/components/schemas/NotFound'
InternalServerError:
  description: Internal Server Error
  content:
    application/json:
      schema:
        $ref: '#/components/schemas/InternalServerError'
    application/xml:
      schema:
        $ref: '#/components/schemas/InternalServerError'
```