**Drink Dispenser Application**

**Overview**

The Drink Dispenser Application is a simple Java application built using Spring Boot that simulates the operation of a drink dispenser within the DevOn office. The dispenser accepts various coins and allows employees to select and purchase drinks. The application is designed to showcase object-oriented design principles and clean code practices.

**Functionality**

The drink dispenser application has the following key features:

**Coin Acceptance:** Accepts 5 cents, 10 cents, 20 cents, 50 cents, 1$, and 2$ coins.

**Product Options:** Employees can choose from the following drinks:

Coca (1$)

Redbull (1$25)

Water (50 cents)

Orange Juice (1$95)

**Order Process:**

Employees insert coins.

Employees type the drink code.

The machine dispenses the selected drink and returns change if necessary.

**Cancellation:**

Employees can cancel an order, and the machine returns the inserted coins.

**Display Information:**

The machine can display its stock, status, and state.

**Extensibility:**

Adding new drinks to the machine is simple and can be done through an API endpoint.

**Code Structure**

**DrinkDispenserService**

This service class manages the business logic of the drink dispenser. It handles coin insertion, drink selection, order cancellation, and displays information about stock and status.

**DrinkDispenserController**

The controller class exposes RESTful endpoints for interacting with the drink dispenser. It maps HTTP requests to corresponding methods in the DrinkDispenserService.

**Models: Coin and Drink**

These classes represent the coin and drink entities in the system.

**Exception Handling**

The application includes a custom exception class, DrinkDispenserException, and an exception handler to provide meaningful error messages to users.

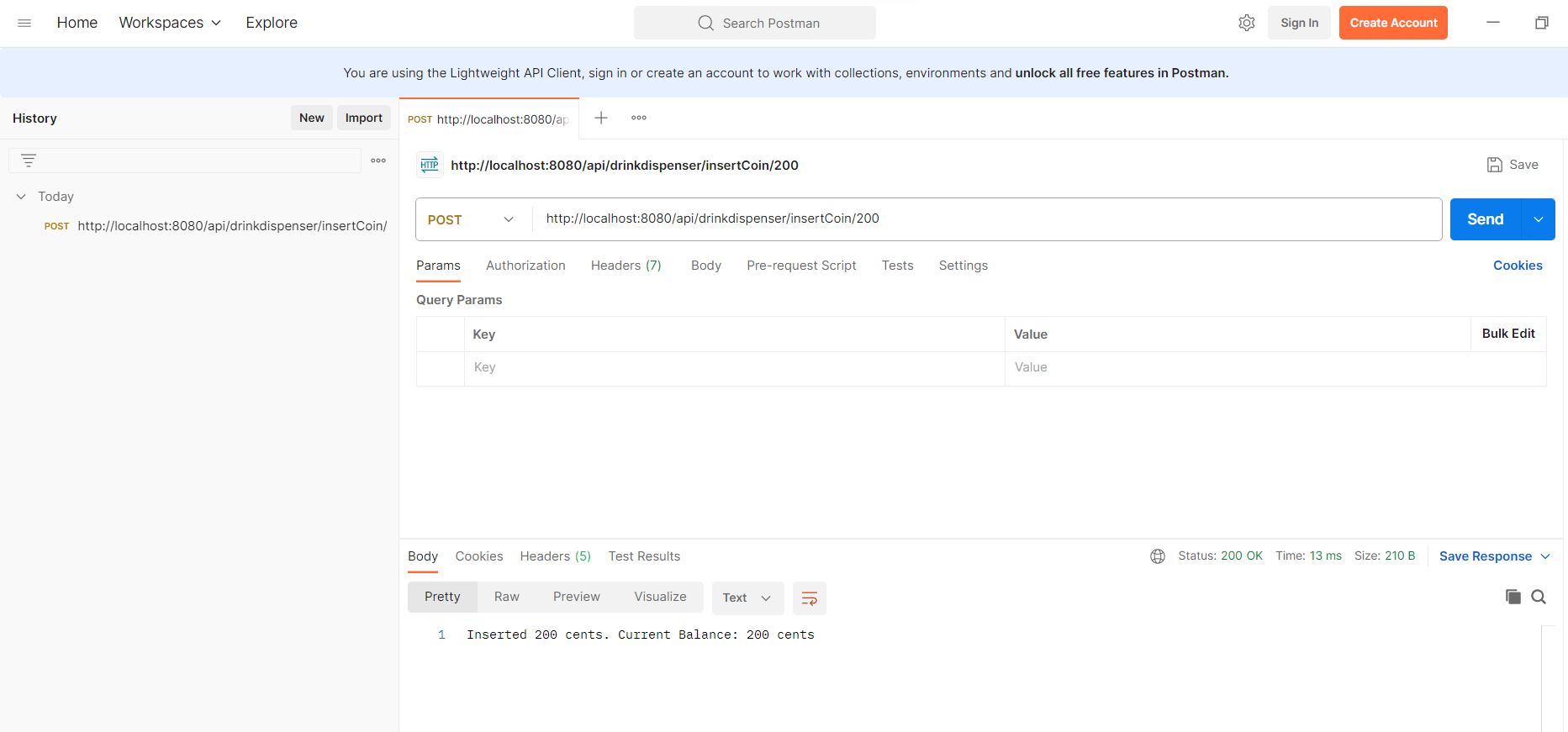
**Endpoints**

**Insert Coin:**

**Endpoint:** /api/drinkdispenser/insertCoin/{coinValue}

**Example:** http://localhost:8080/api/drinkdispenser/insertCoin/200

**Method:** POST

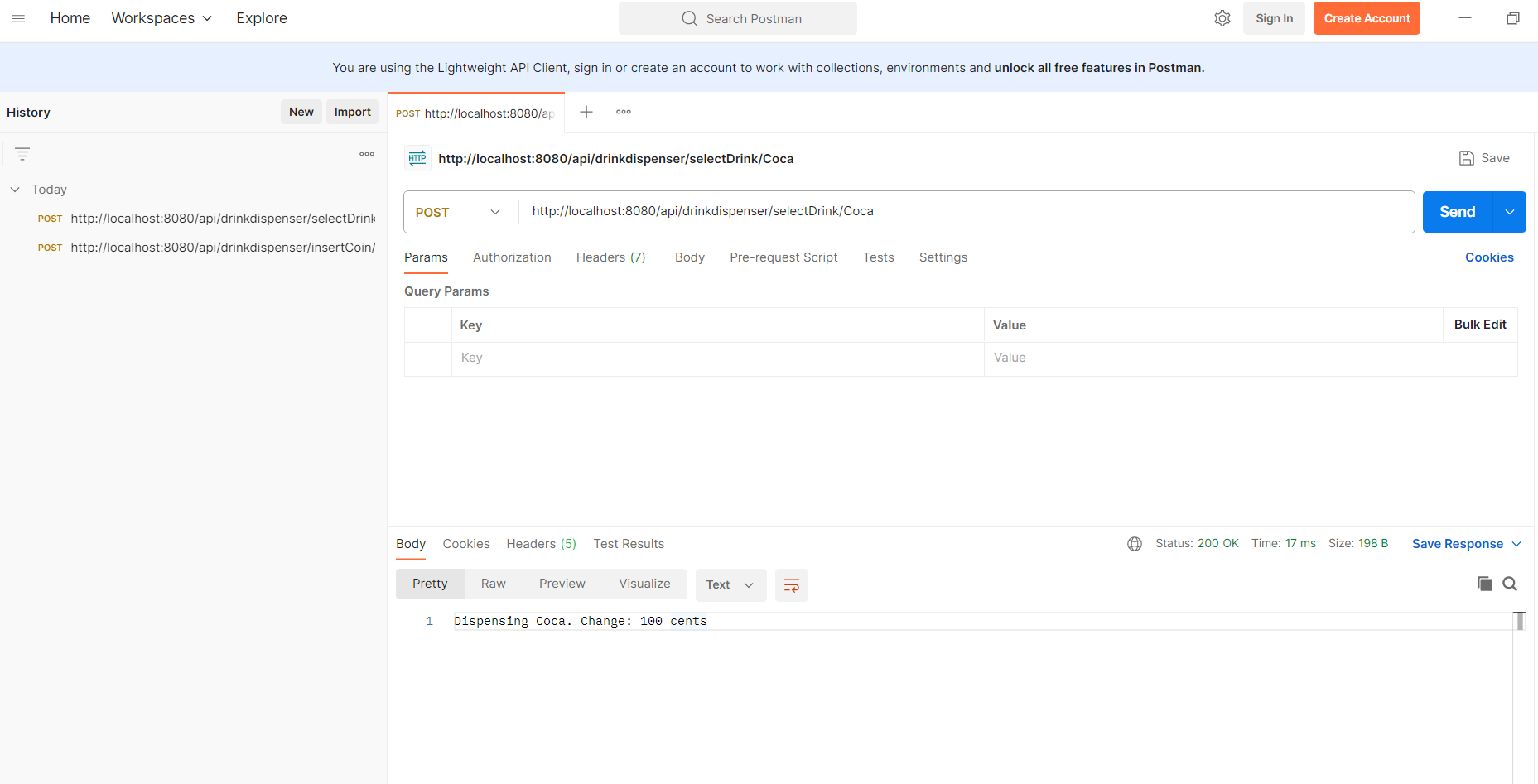


**Select Drink:**

**Endpoint:** /api/drinkdispenser/selectDrink/{drinkCode}

**Example:** http://localhost:8080/api/drinkdispenser/selectDrink/Coca

**Method:** POST

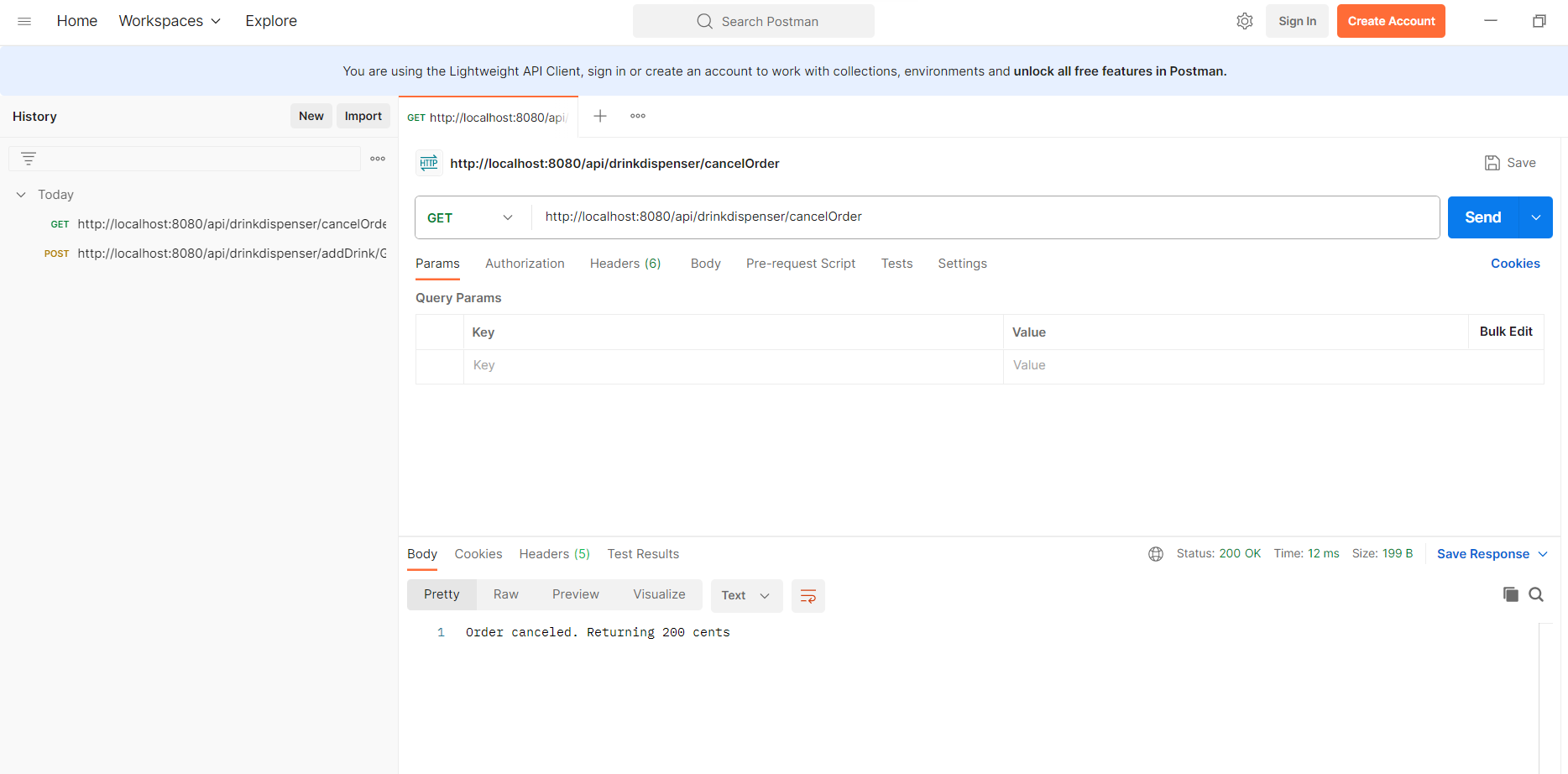


**Cancel Order:**

**Endpoint:** /api/drinkdispenser/cancelOrder

**Example: http://localhost:8080/api/drinkdispenser/cancelOrder**

**Method:** GET

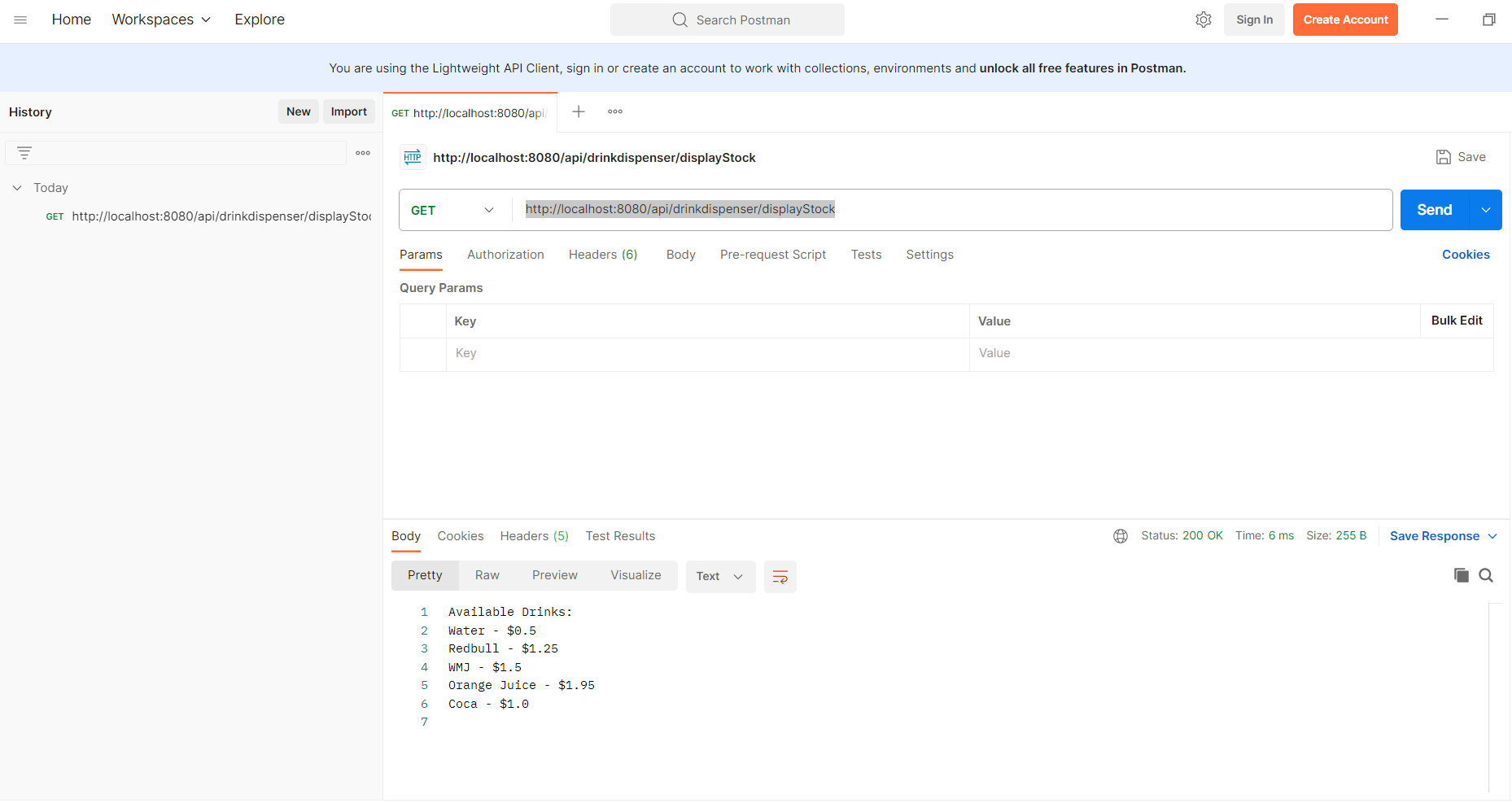


**Display Stock:**

**Endpoint:** /api/drinkdispenser/displayStock

**Example:** http://localhost:8080/api/drinkdispenser/displayStock

**Method:** GET

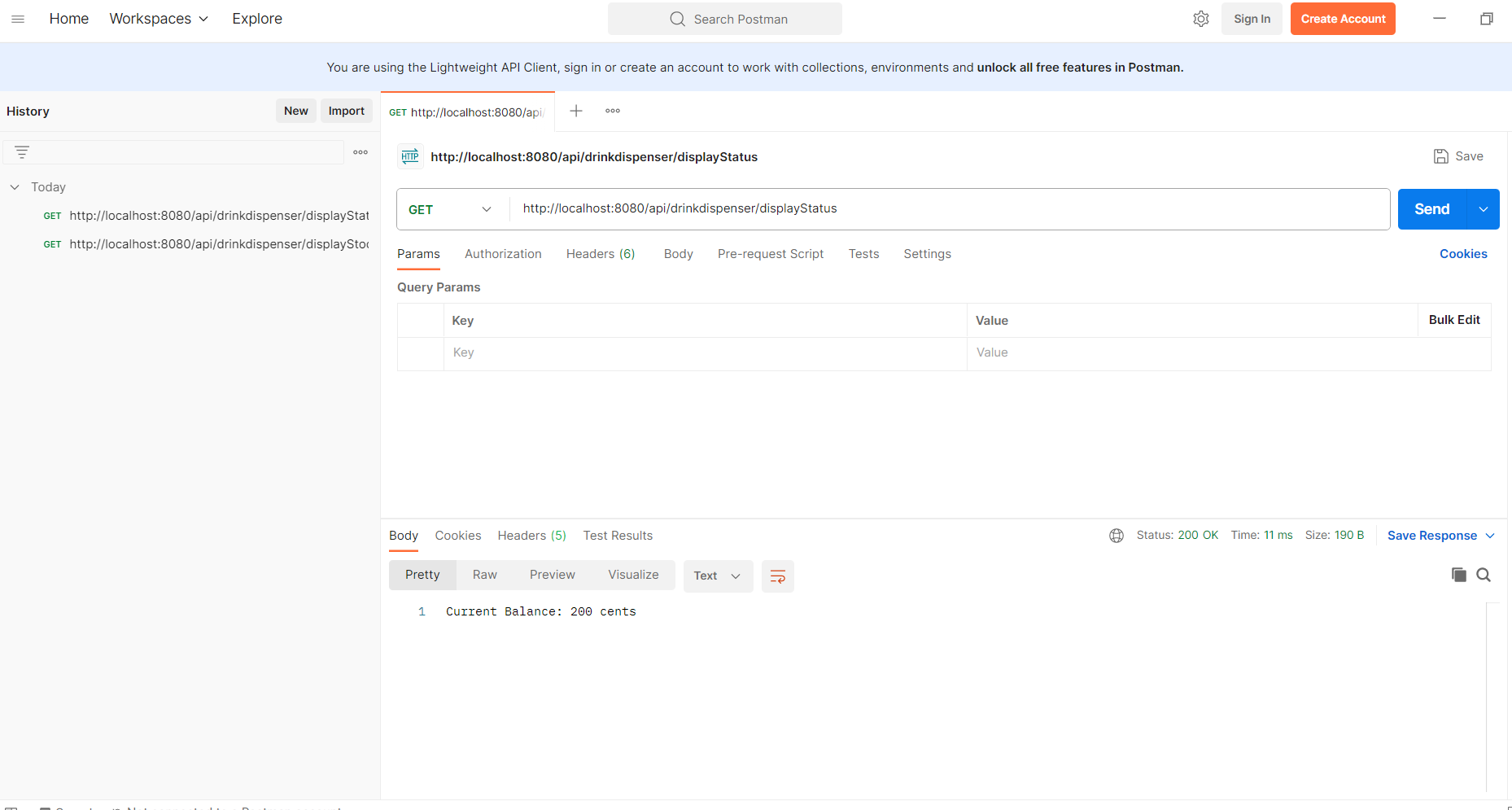


**Display Status:**

**Endpoint:** /api/drinkdispenser/displayStatus

**Example:** http://localhost:8080/api/drinkdispenser/displayStatus

**Method:** GET

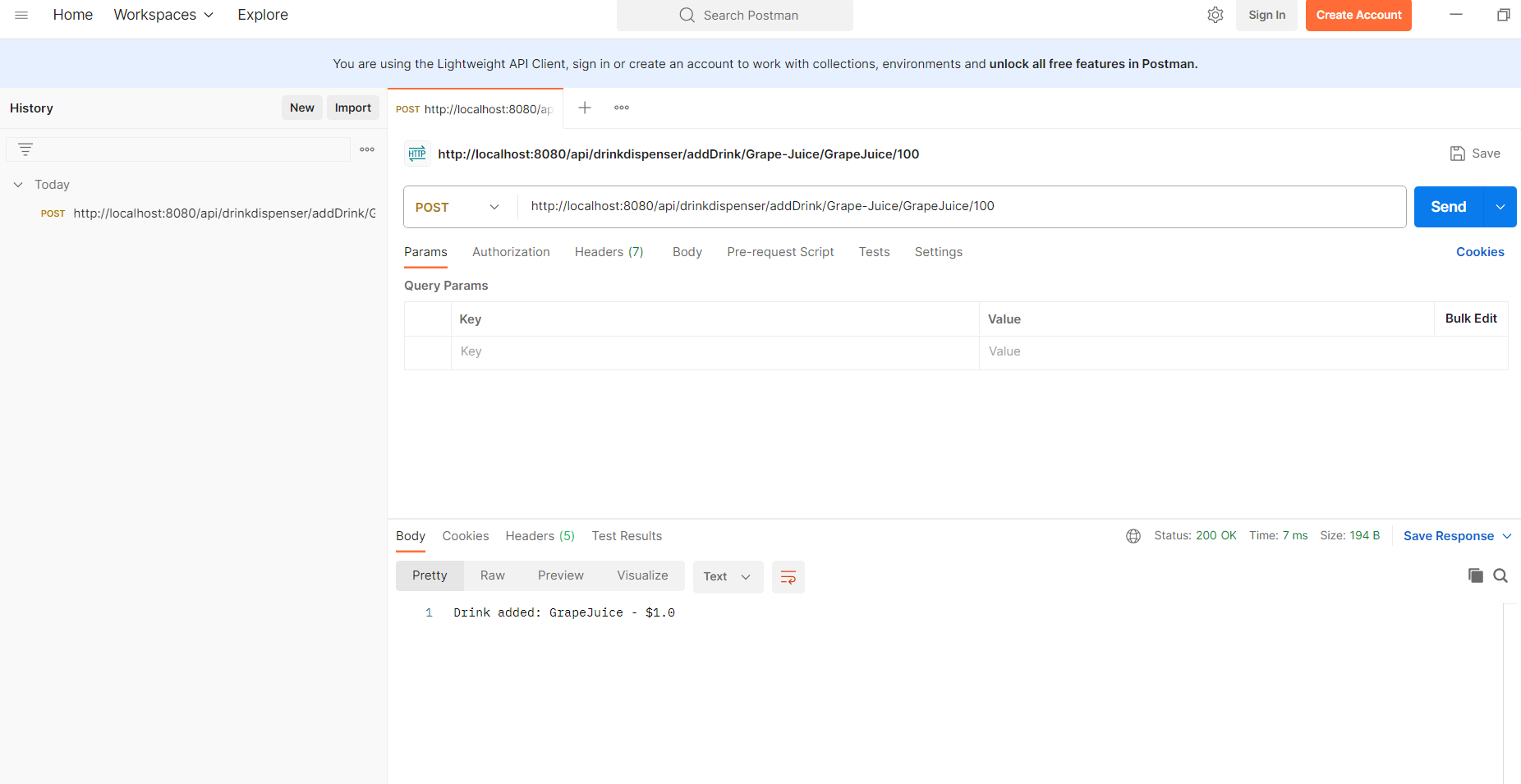


**Add Drink:**

**Endpoint:** /api/drinkdispenser/addDrink/{drinkCode}/{drinkName}/{price}

**Example:** <http://localhost:8080/api/drinkdispenser/addDrink/Grape>-Juice/GrapeJuice/100

**Method:** POST



**Testing**

JUnit tests have been provided to ensure the correctness of the application's functionality. These tests cover various scenarios, including coin insertion, drink selection, cancellation, and error handling.

**How to Run**

Ensure you have Java 11 and Maven installed on your machine.

Clone the repository.

Navigate to the project root directory.

Run the application using the command: mvn spring-boot:run.

**Future Improvements**

**Refill Drinks Functionality:** Currently, there is no implementation for refilling drinks. This could be added in the future to simulate restocking the dispenser.

**Security:** Consider implementing security measures, especially if this application is deployed in a production environment.

**Logging:** Enhance logging to provide detailed information about transactions and errors.