Web API Design with SpringBoot Week 2 Coding Assignment

Points possible: 70

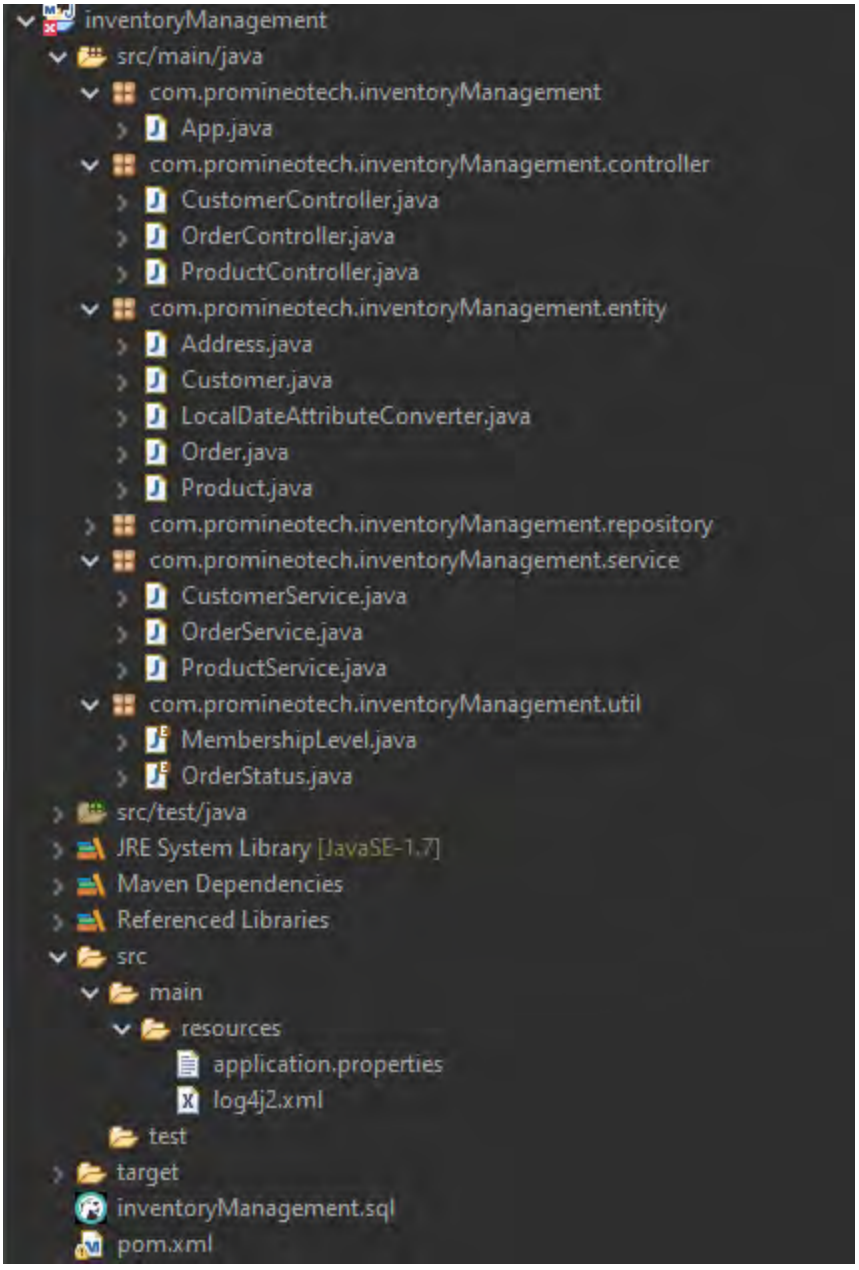
Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete.	25

Instructions: In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document, with your Java project code, to the repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

Coding Steps:

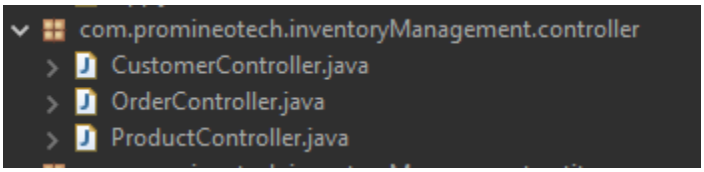
Follow the tutorial in the homework section to build an API. Paste screenshots of your code and your postman requests and responses to show the API works. Push your project to GitHub and paste the link below.

Screenshots of Code:



App.java

```java
package com.promineotech.inventoryManagement;

import org.springframework.boot.SpringApplication;

@ComponentScan("com.promineotech.inventoryManagement")
@SpringBootApplication
public class App
{
    public static void main( String[] args )
    {
        SpringApplication.run(App.class, args);
    }
}
```



CustomerController.java

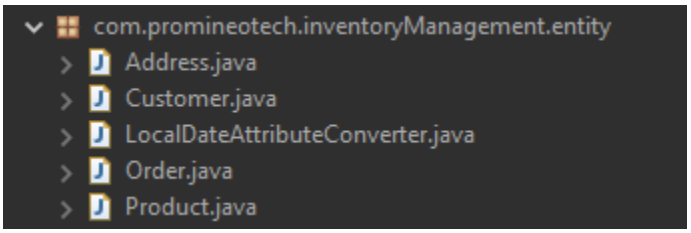
```java
package com.promineotech.inventoryManagement.controller;


import org.springframework.beans.factory.annotation.Autowired;

@RestController
@RequestMapping("/customer")
public class CustomerController {

    @Autowired
    private CustomerService service;

    @RequestMapping(value="/{id}", method=RequestMethod.GET)
    public ResponseEntity<Object> getCustomer(@PathVariable Long id) {
        try {
            return new ResponseEntity<Object>(service.getCustomerById(id), HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
        }
    }

    @RequestMapping(method=RequestMethod.GET)
    public ResponseEntity<Object> getCustomer() {
        return new ResponseEntity<Object>(service.getCustomer(), HttpStatus.OK);
    }

    @RequestMapping(method=RequestMethod.POST)
    public ResponseEntity<Object> createCustomer(@RequestBody Customer customer) {
        return new ResponseEntity<Object>(service.createCustomer(customer), HttpStatus.CREATED);
    }

    @RequestMapping(value="/{id}", method=RequestMethod.PUT)
    public ResponseEntity<Object> updateCustomer(@RequestBody Customer customer, @PathVariable Long id) {
        try {
            return new ResponseEntity<Object>(service.updateCustomer(customer, id), HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
        }
    }

    @RequestMapping(value="/{id}", method=RequestMethod.DELETE)
    public ResponseEntity<Object> deleteCustomer(@PathVariable Long id) {
        try {
            service.deleteCustomer(id);
            return new ResponseEntity<Object>("Successfully deleted customer with id: " + id, HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
        }
    }

}
```

OrderController.java

```java
package com.promineotech.inventoryManagement.controller;

import java.util.Set;

@RestController
@RequestMapping("customer/{id}/orders")
public class OrderController {

    @Autowired
    private OrderService service;

    @RequestMapping(method=RequestMethod.POST)
    public ResponseEntity<Object> createCustomer(@RequestBody Set<Long> productIds, @PathVariable Long id) {
        try {
            return new ResponseEntity<Object>(service.submitNewOrder(productIds, id), HttpStatus.CREATED);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e, HttpStatus.BAD_REQUEST);
        }
    }

    @RequestMapping(value="/{orderId}", method=RequestMethod.PUT)
    public ResponseEntity<Object> updateOrder(@RequestBody Order order, @PathVariable Long orderId) {
        try {
            if (order.getStatus().equals(OrderStatus.CANCELED)) {
                return new ResponseEntity<Object>(service.cancelOrder(orderId), HttpStatus.OK);
            } else if (order.getStatus().equals(OrderStatus.DELIVERED)) {
                return new ResponseEntity<Object>(service.completeOrder(orderId), HttpStatus.OK);
            }
            return new ResponseEntity<Object>("Invalid update request.", HttpStatus.BAD_REQUEST);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
        }
    }
}

```

ProductController.java

```java
package com.promineotech.inventoryManagement.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import com.promineotech.inventoryManagement.entity.Product;
import com.promineotech.inventoryManagement.service.ProductService;

@RestController
@RequestMapping("/products")
public class ProductController {

    @Autowired
    private ProductService service;

    @RequestMapping(method=RequestMethod.GET)
    public ResponseEntity<Object> getProducts() {
        return new ResponseEntity<Object>(service.getProducts(), HttpStatus.OK);
    }

    @RequestMapping(method=RequestMethod.POST)
    public ResponseEntity<Object> createProduct(@RequestBody Product product) {
        return new ResponseEntity<Object>(service.createProduct(product), HttpStatus.CREATED);
    }

    @RequestMapping(value="/{id}", method=RequestMethod.PUT)
    public ResponseEntity<Object> updateProduct(@RequestBody Product product, @PathVariable Long id) {
        try {
            return new ResponseEntity<Object>(service.updateProduct(product, id), HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>("Unable to update product.", HttpStatus.BAD_REQUEST);
        }
    }

    @RequestMapping(value="/{id}", method=RequestMethod.DELETE)
    public ResponseEntity<Object> deleteProduct(@PathVariable Long id) {
        try {
            service.removeProduct(id);
            return new ResponseEntity<Object>("Successfully deleted product with id: ", HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>("Unable to delete product.", HttpStatus.BAD_REQUEST);
        }
    }
}

```



Address.java

```java
package com.promineotech.inventoryManagement.entity;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToOne;

import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity
public class Address {

    private Long id;
    private String street;
    private String city;
    private String state;
    private String zip;

    @JsonIgnore
    private Customer customer;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getStreet() {
        return street;
    }

    public void setStreet(String street) {
        this.street = street;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public String getState() {
        return state;
    }

    public void setState(String state) {
        this.state = state;
    }

    public String getZip() {
        return zip;
    }

    public void setZip(String zip) {
        this.zip = zip;
    }

    @OneToOne(mappedBy = "address")
    public Customer getCustomer() {
        return customer;
    }

    public void setCustomer(Customer customer) {
        this.customer = customer;
    }

}
```

Customer.java

```java
package com.promineotech.inventoryManagement.entity;

import java.util.Set;

@Entity
public class Customer {

    private Long id;
    private String firstName;
    private String lastName;
    private Address address;
    private MembershipLevel level;
    private Set<Order> orders;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "address_id")
    public Address getAddress() {
        return address;
    }
    public void setAddress(Address address) {
        this.address = address;
    }
    public MembershipLevel getLevel() {
        return level;
    }
    public void setLevel(MembershipLevel level) {
        this.level = level;
    }
    @OneToMany(mappedBy = "customer")
    public Set<Order> getOrders() {
        return orders;
    }
    public void setOrders(Set<Order> orders) {
        this.orders = orders;
    }



}

```

LocalDateAttributeConverter.java

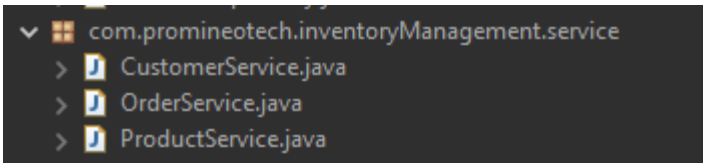
```java
package com.promineotech.inventoryManagement.entity;

import java.sql.Date;

@Converter(autoApply = true)
public class LocalDateAttributeConverter implements AttributeConverter<LocalDate, Date> {

    //@Override
    public Date convertToDatabaseColumn(LocalDate locDate) {
        return (locDate == null ? null : Date.valueOf(locDate));
    }

    //@Override
    public LocalDate convertToEntityAttribute(Date sqlDate) {
        return (sqlDate == null ? null : sqlDate.toLocalDate());
    }

}
```

Order.java

```java
package com.promineotech.inventoryManagement.entity;

import java.time.LocalDate;
import java.util.Set;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToMany;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

import com.fasterxml.jackson.annotation.JsonIgnore;
import com.promineotech.inventoryManagement.util.OrderStatus;

@Entity
@Table(name = "orders")
public class Order {

    private Long id;
    private LocalDate ordered;
    private LocalDate estimatedDelivery;
    private LocalDate delivered;
    private double invoiceAmount;
    private OrderStatus status;
    private Set<Product> products;

    @JsonIgnore
    private Customer customer;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public LocalDate getOrdered() {
        return ordered;
    }
    public void setOrdered(LocalDate ordered) {
        this.ordered = ordered;
    }
    public LocalDate getEstimatedDelivery() {
        return estimatedDelivery;
    }
    public void setEstimatedDelivery(LocalDate estimatedDelivery) {
        this.estimatedDelivery = estimatedDelivery;
    }
    public LocalDate getDelivered() {
        return delivered;
    }
    public void setDelivered(LocalDate delivered) {
        this.delivered = delivered;
    }
    public double getInvoiceAmount() {
        return invoiceAmount;
    }
    public void setInvoiceAmount(double invoiceAmount) {
        this.invoiceAmount = invoiceAmount;
    }
    // It's plain an order can contain many produts, and a product can be in many orders.
    //This is the "owned" side
    @ManyToMany(mappedBy = "orders")
    public Set<Product> getProducts() {
        return products;
    }
    public void setProducts(Set<Product> products) {
        this.products = products;
    }

    @ManyToOne
    @JoinColumn(name = "customerId")
    public Customer getCustomer() {
        return customer;
    }

    public void setCustomer(Customer customer) {
        this.customer = customer;
    }

    public OrderStatus getStatus() {
        return status;
    }
    public void setStatus(OrderStatus status) {
        this.status = status;
    }

}
```

Product.java

```java
package com.promineotech.inventoryManagement.entity;

import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;

import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity
public class Product {

    private Long id;
    private String name;
    private String description;
    private double price;

    @JsonIgnore
    private Set<Order> orders;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    @ManyToMany(cascade = CascadeType.ALL)
    @JoinTable(name = "product_order",
            joinColumns = @JoinColumn(name = "productId", referencedColumnName = "id"),
            inverseJoinColumns = @JoinColumn(name = "orderId", referencedColumnName = "id"))
    public Set<Order> getOrders() {
        return orders;
    }

    public void setOrders(Set<Order> orders) {
        this.orders = orders;
    }


}

```



AddressRepository.java

```java
package com.promineotech.inventoryManagement.repository;

import org.springframework.data.repository.CrudRepository;

public interface AddressRepository extends CrudRepository<Address, Long> {

}

```

CustomerRepository.java

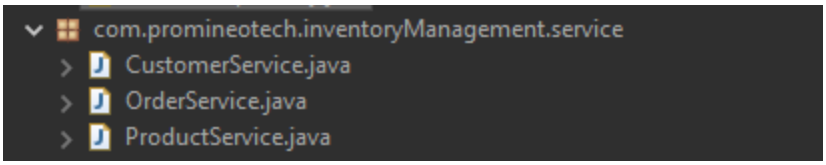
```java
package com.promineotech.inventoryManagement.repository;

import org.springframework.data.repository.CrudRepository;

public interface CustomerRepository extends CrudRepository<Customer, Long> {

}

```

OrderRepository.java

```java
package com.promineotech.inventoryManagement.repository;

import org.springframework.data.repository.CrudRepository;

public interface OrderRepository extends CrudRepository<Order, Long> {

}

```

ProductRepository.java

```java
package com.promineotech.inventoryManagement.repository;

import org.springframework.data.repository.CrudRepository;

public interface ProductRepository extends CrudRepository<Product, Long> {

}

```



CustomerService.java

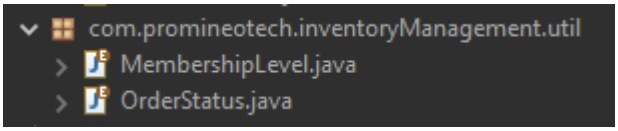
```java
package com.promineotech.inventoryManagement.service;


import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.promineotech.inventoryManagement.entity.Customer;
import com.promineotech.inventoryManagement.repository.CustomerRepository;

@Service
public class CustomerService {

    private static final Logger logger = LogManager.getLogger(CustomerService.class);

    @Autowired
    private CustomerRepository repo;

    public Customer getCustomerById(Long id) throws Exception {
        try {
            return repo.findOne(id);
        } catch (Exception e) {
            logger.error("Exception occurred while trying to retrieve customer: " + id, e);
            throw e;
        }
    }

    public Iterable<Customer> getCustomer() {
        return repo.findAll();
    }

    public Customer createCustomer(Customer customer) {
        return repo.save(customer);
    }

    public Customer updateCustomer(Customer customer, Long id) throws Exception {
        try {
            Customer oldCustomer = repo.findOne(id);
            oldCustomer.setAddress(customer.getAddress());
            oldCustomer.setFirstName(customer.getFirstName());
            oldCustomer.setLastName(customer.getLastName());
            oldCustomer.setLevel(customer.getLevel());
            return repo.save(oldCustomer);
        } catch (Exception e) {
            logger.error("Exception occurred while trying to update customer: " + id, e);
            throw new Exception("unable to update customer.");
        }
    }

    public void deleteCustomer(Long id) throws Exception {
        try {
            repo.delete(id);
        } catch (Exception e) {
            logger.error("Exception occurred while trying to delete customer: " + id, e);
            throw new Exception("unable to delete customer.");
        }
    }

}
```

OrderService.java

```java
package com.promineotech.inventoryManagement.service;

import java.time.LocalDate;
import java.util.HashSet;
import java.util.Set;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.promineotech.inventoryManagement.entity.Customer;
import com.promineotech.inventoryManagement.entity.Order;
import com.promineotech.inventoryManagement.entity.Product;
import com.promineotech.inventoryManagement.repository.CustomerRepository;
import com.promineotech.inventoryManagement.repository.OrderRepository;
import com.promineotech.inventoryManagement.repository.ProductRepository;
import com.promineotech.inventoryManagement.util.MembershipLevel;
import com.promineotech.inventoryManagement.util.OrderStatus;

@Service
public class OrderService {

    private static final Logger logger = LogManager.getLogger(OrderService.class);
    private final int DELIVERY_DAYS = 7;

    @Autowired
    private OrderRepository repo;

    @Autowired
    private CustomerRepository customerRepo;

    @Autowired
    private ProductRepository productRepo;

    public Order submitNewOrder(Set<Long> productIds, Long customerId) throws Exception {
        try {
            Customer customer = customerRepo.findOne(customerId);
            Order order = initializeNewOrder(productIds, customer);
            return repo.save(order);
        } catch (Exception e) {
            logger.error("Exception occurred while trying to create new order for customer: " + customerId, e);
            throw e;
        }
    }

    public Order cancelOrder(Long orderId) throws Exception {
        try {
            Order order = repo.findOne(orderId);
            order.setStatus(OrderStatus.CANCELED);
            return repo.save(order);
        } catch (Exception e) {
            logger.error("Exception occurred while trying to cancel order: " + orderId, e);
            throw new Exception("unable to update order.");
        }
    }

    public Order completeOrder(Long orderId) throws Exception {
        try {
            Order order =repo.findOne(orderId);
```

ProductService.java

```java
package com.promineotech.inventoryManagement.service;


import org.apache.logging.log4j.LogManager;

@Service
public class ProductService {

    private static final Logger logger = (Logger) LogManager.getLogger(ProductService.class);

    @Autowired
    private ProductRepository repo;

    public Iterable<Product> getProducts() {
        return repo.findAll();
    }

    public Product createProduct(Product product) {
        return repo.save(product);
    }

    public Product updateProduct(Product product, Long id) throws Exception {
        try {
            Product oldProduct = repo.findOne(id);
            oldProduct.setDescription(product.getDescription());
            oldProduct.setName(product.getName());
            oldProduct.setPrice(product.getPrice());
            return repo.save(oldProduct);
        } catch (Exception e) {
            logger.error("Exception occurred while trying to update product: " + id, e);
            throw new Exception("unable to update product.");
        }
    }

    public void removeProduct(Long id) throws Exception {
        try {
            repo.delete(id);
        } catch (Exception e) {
            logger.error("Exceptioin occured while trying to delete product: " + id, e);
            throw new Exception("unable to delete product.");
        }
    }

}

```



```java
package com.promineotech.inventoryManagement.util;

public enum MembershipLevel {

    SILVER(.05),
    GOLD(.10),
    DIAMOND(.15),
    PLATINUM(.20);

    private double discount;

    MembershipLevel(double discount) {
        this.discount = discount;
    }

    public double getDiscount() {
        return discount;
    }

}

```

OrderStatus.java

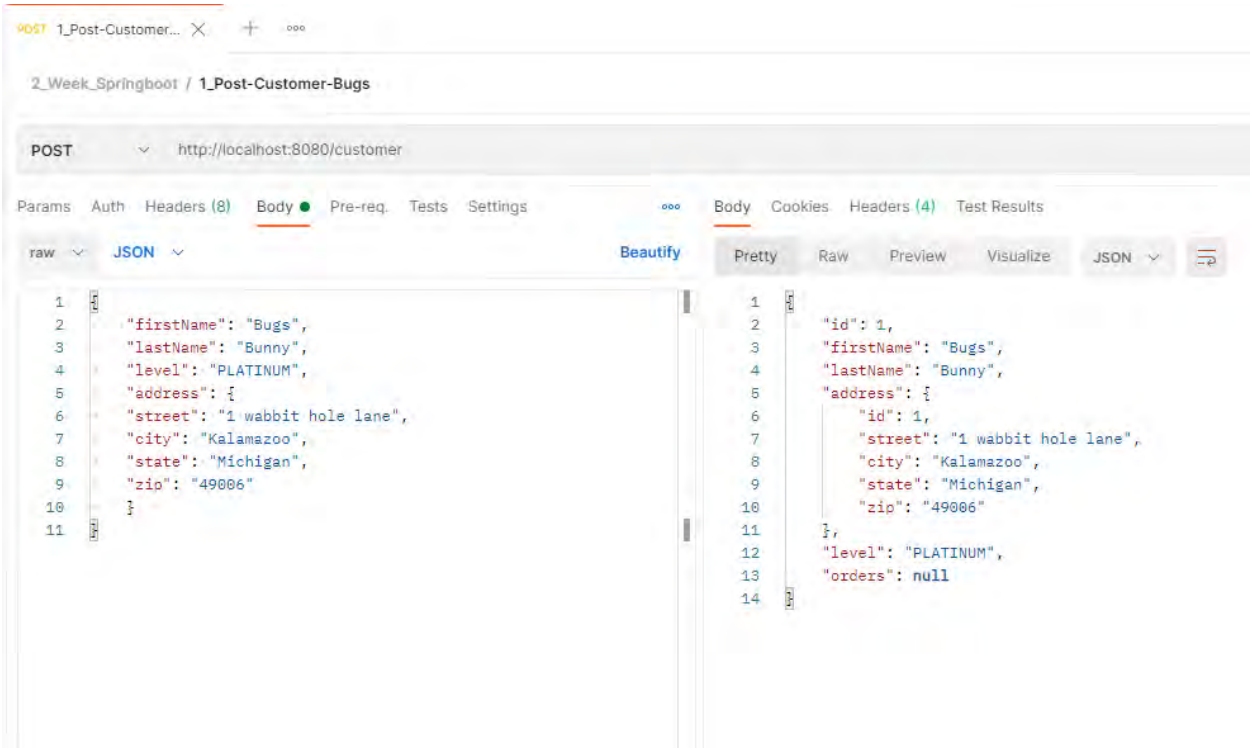
```java
package com.promineotech.inventoryManagement.util;

public enum OrderStatus {

    ORDERED,
    DELIVERED,
    CANCELED;
}

```

Screenshots of Running Application:

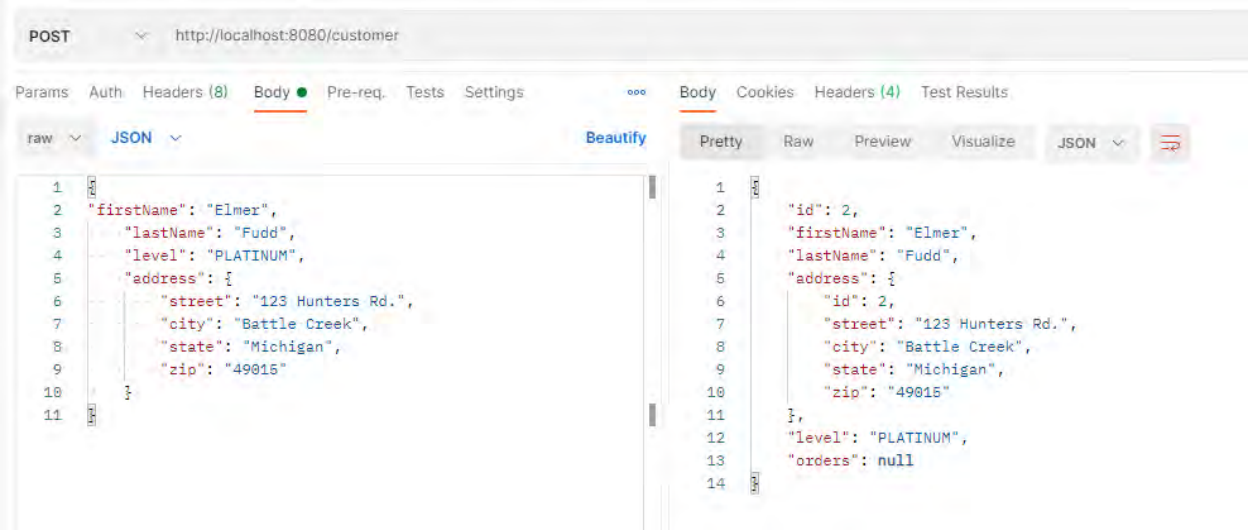
Text document titled “JeffBeck_Week_2_CodingAssignment_Postman_Tests” saved to Git details data used and steps for testing code in postman. Each step in text document corresponds with steps listed below:

Text document titled “CommandPrompt-mvn spring-boot run” is the CLI documentations showing the spring-boot API was able to make establish the necessary endpoints in order to carry out and test the API in Postman…

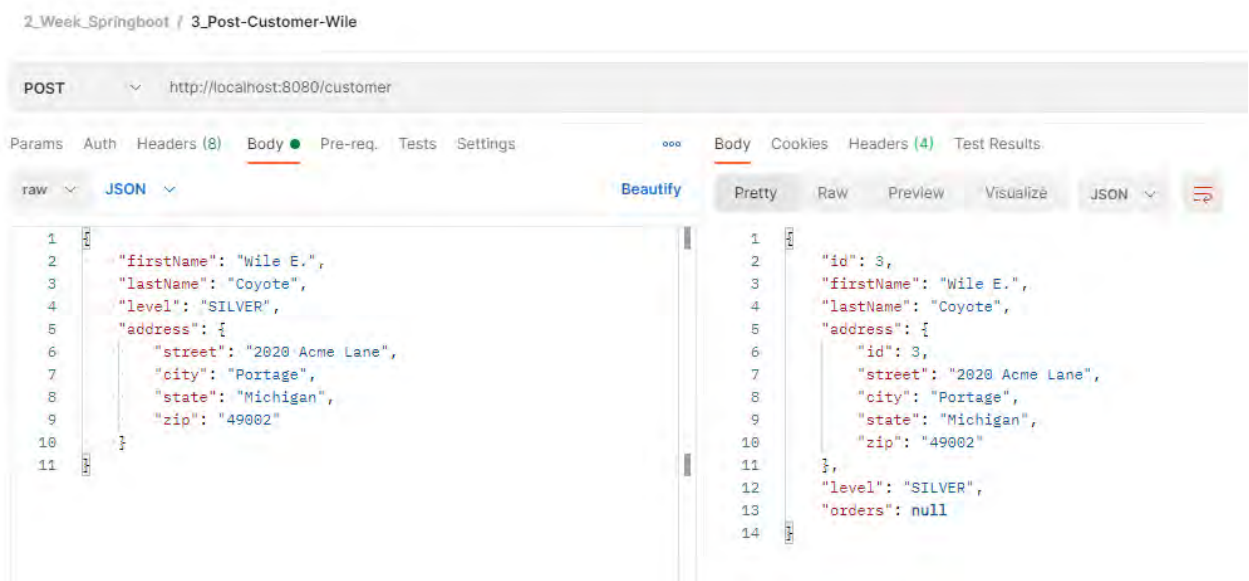
(1) POST (body/raw/JSON) http://localhost:8080/customer



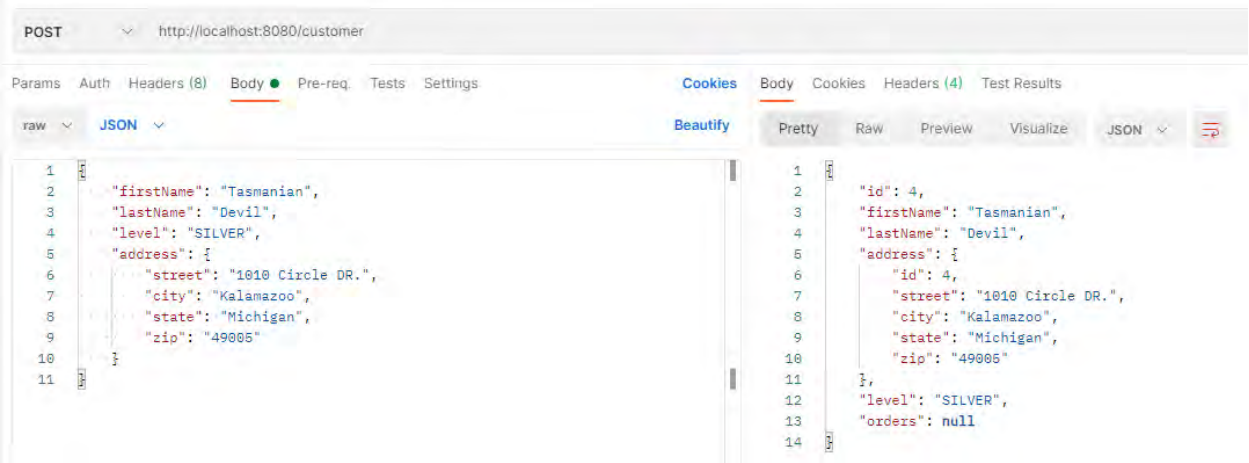
(2) //2_Post-Customer-Elmer



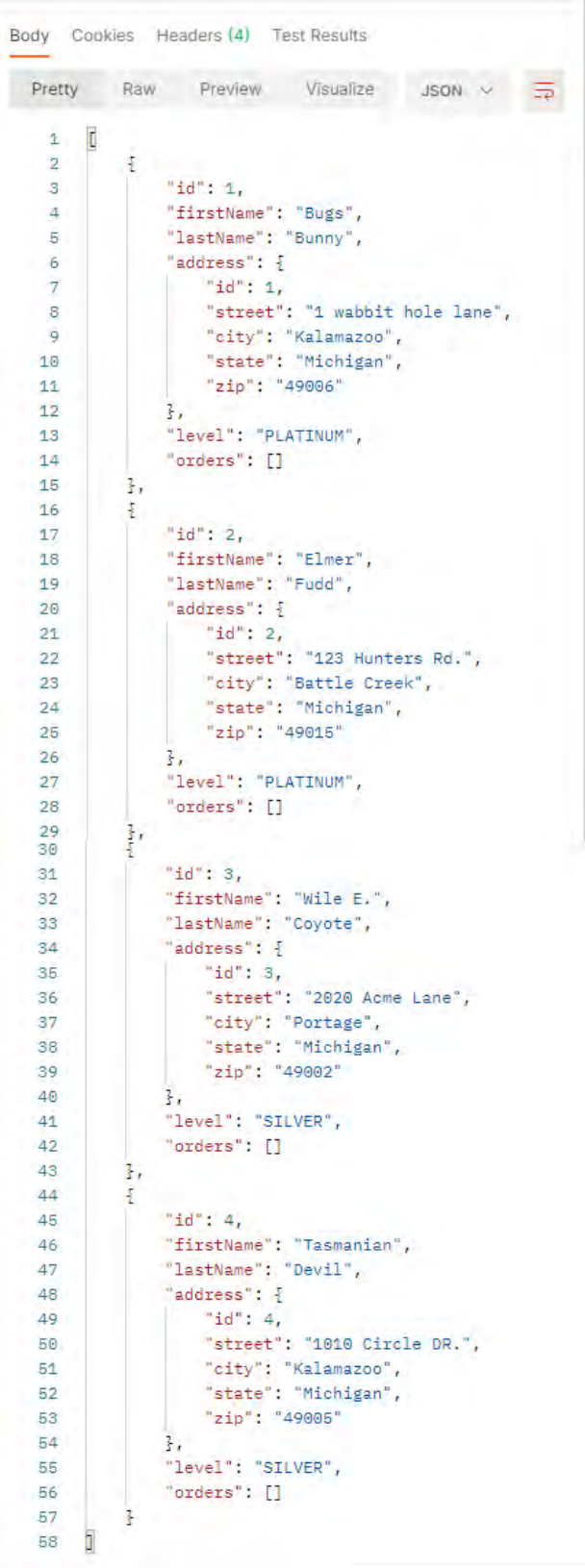
(3) //3_Post-Customer-Wile



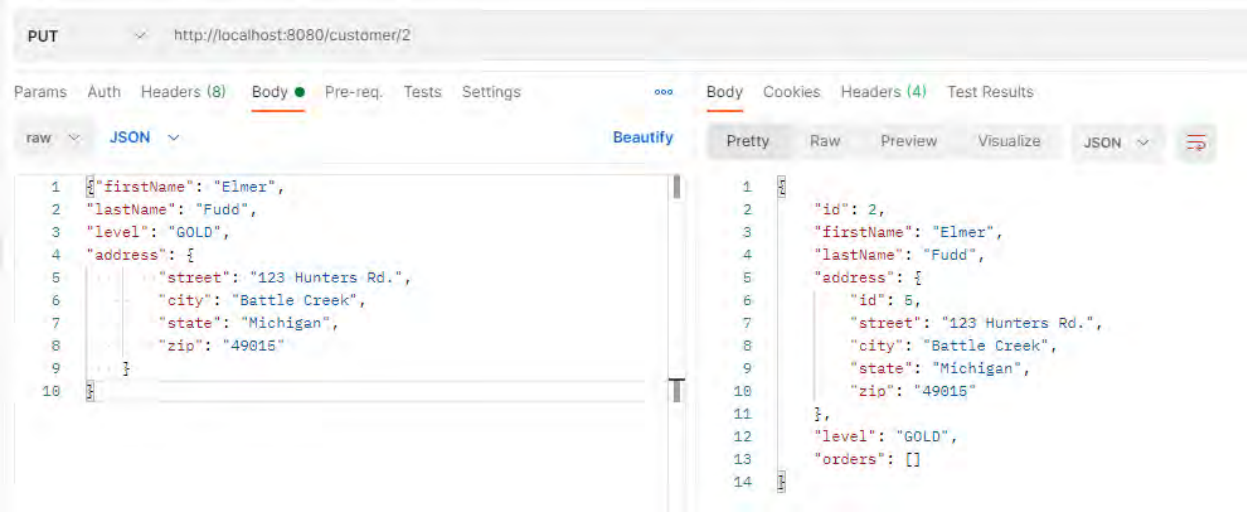
(4) //4_Post-Customer-Tasmanian



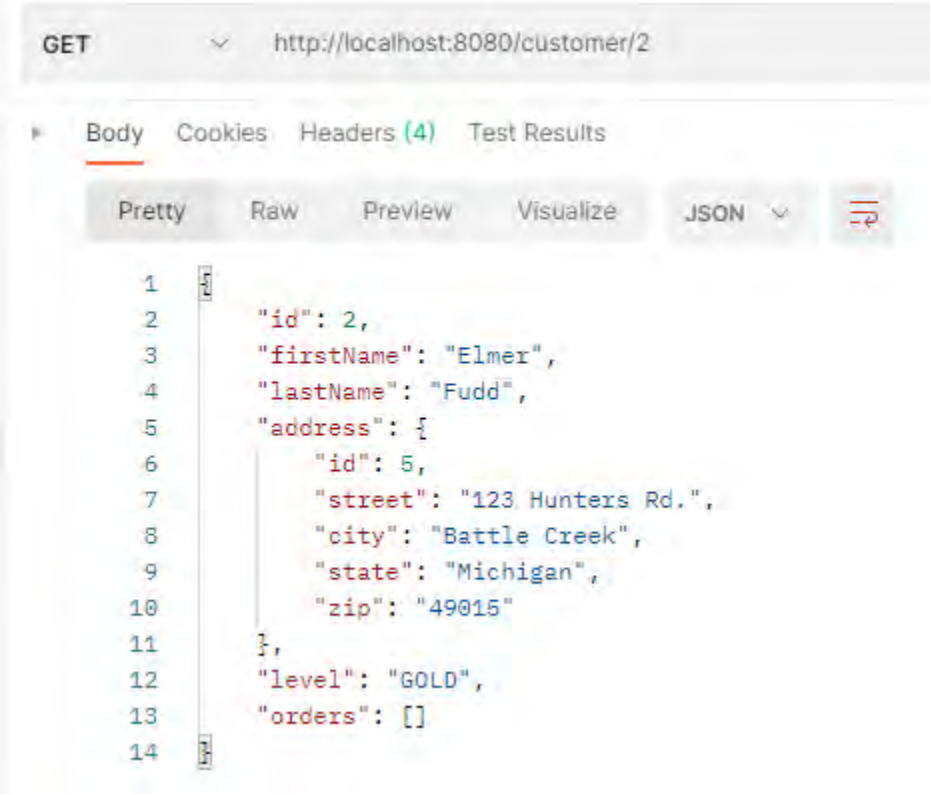
(5) GET (body/raw/JSON) http://localhost:8080/customer



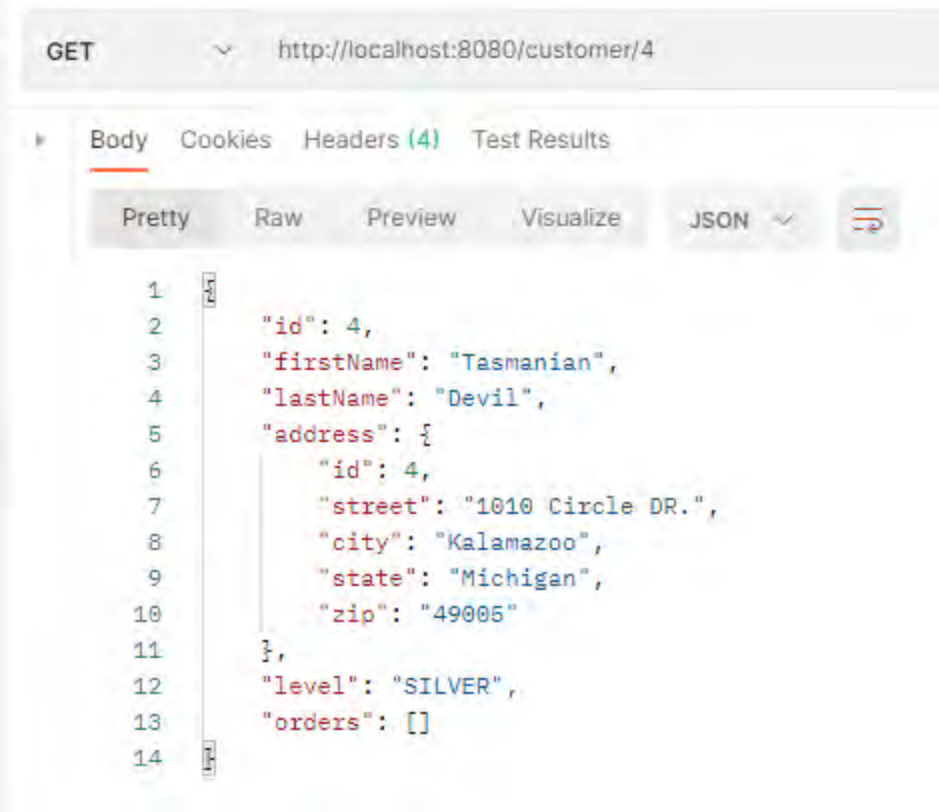
(6) PUT (body/raw/JSON) http://localhost:8080/customer/2



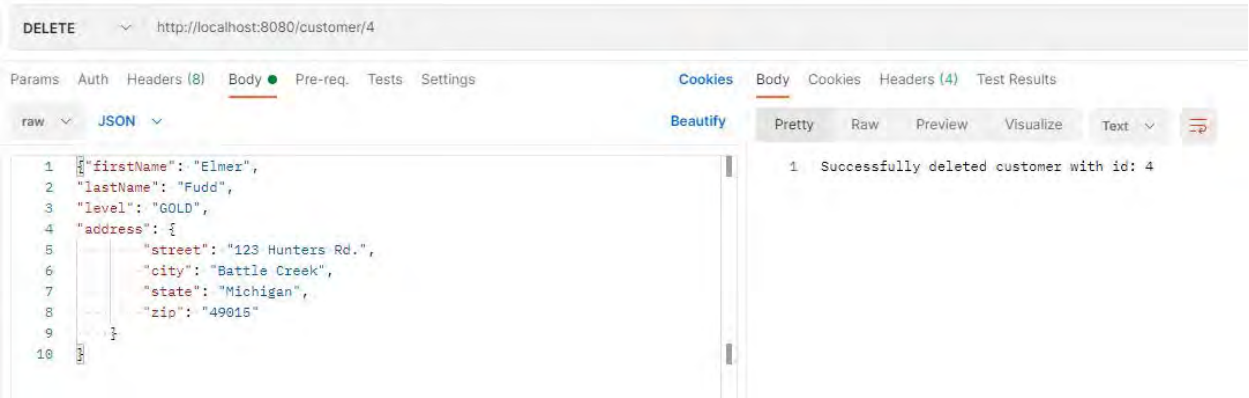
(7) GET (body/raw/JSON) http://localhost:8080/customer/2



(8) GET (body/raw/JSON) http://localhost:8080/customer/4



(9) DELETE (body/raw/JSON) http://localhost:8080/customer/4

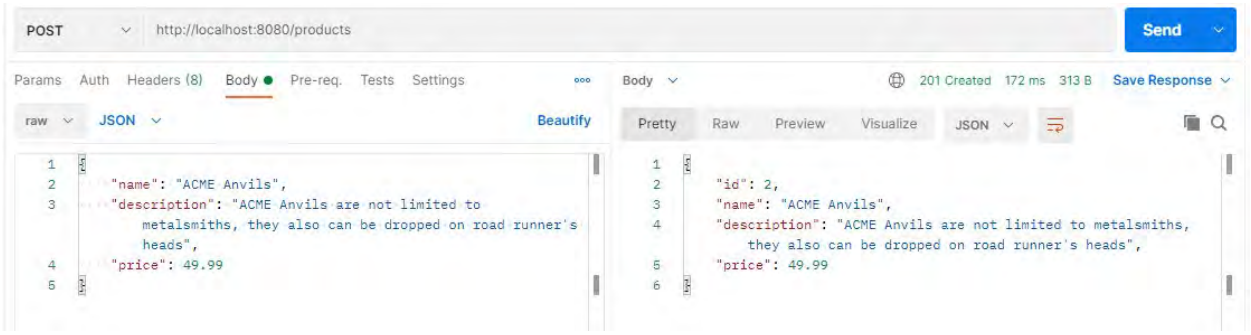


(10) GET (body/raw/JSON) http://localhost:8080/customer/


```
[
    {
        "id": 1,
        "firstName": "Bugs",
        "lastName": "Bunny",
        "address": {
            "id": 1,
            "street": "1 wabbit hole lane",
            "city": "Kalamazoo",
            "state": "Michigan",
            "zip": "49006"
        },
        "level": "PLATINUM",
        "orders": []
    },
    {
        "id": 2,
        "firstName": "Elmer",
        "lastName": "Fudd",
        "address": {
            "id": 5,
            "street": "123 Hunters Rd.",
            "city": "Battle Creek",
            "state": "Michigan",
            "zip": "49015"
        },
        "level": "GOLD",
        "orders": []
    },
    {
        "id": 3,
        "firstName": "Wile E.",
        "lastName": "Coyote",
        "address": {
            "id": 3,
            "street": "2020 Acme Lane",
            "city": "Portage",
            "state": "Michigan",
            "zip": "49002"
        },
        "level": "SILVER",
        "orders": []
    }
]
```

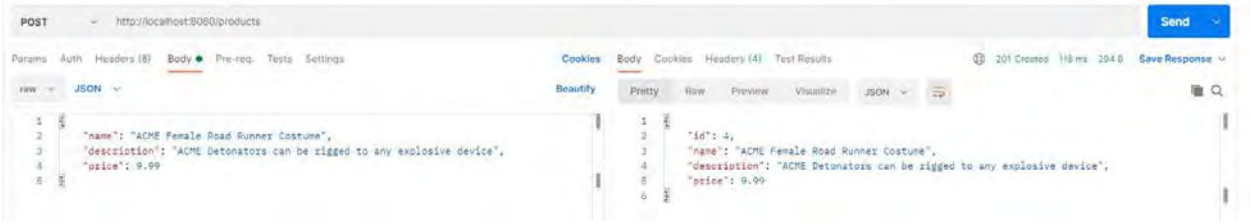
(11) POST (body/raw/JSON) http://localhost:8080/products

(12) // POST-Products-Anvils

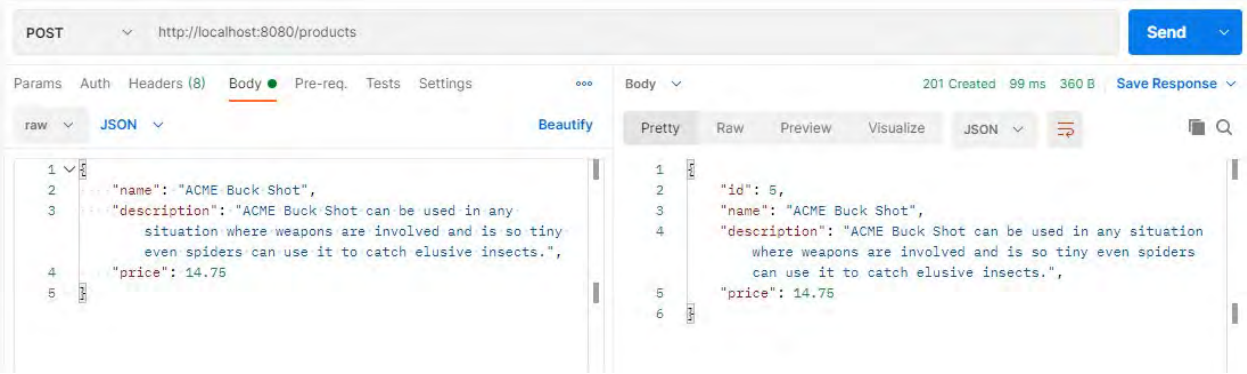


(13) // POST-Products-Detonator

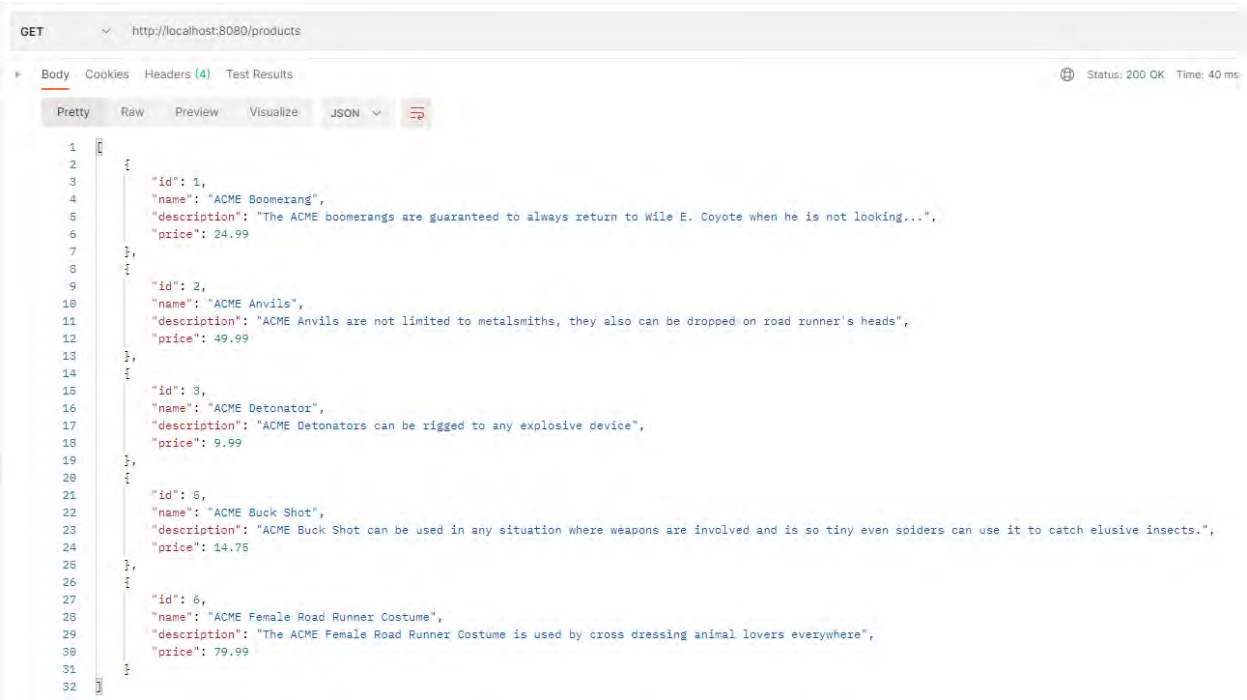
(14) // POST-Products-FemaleRoadRunnerCostume



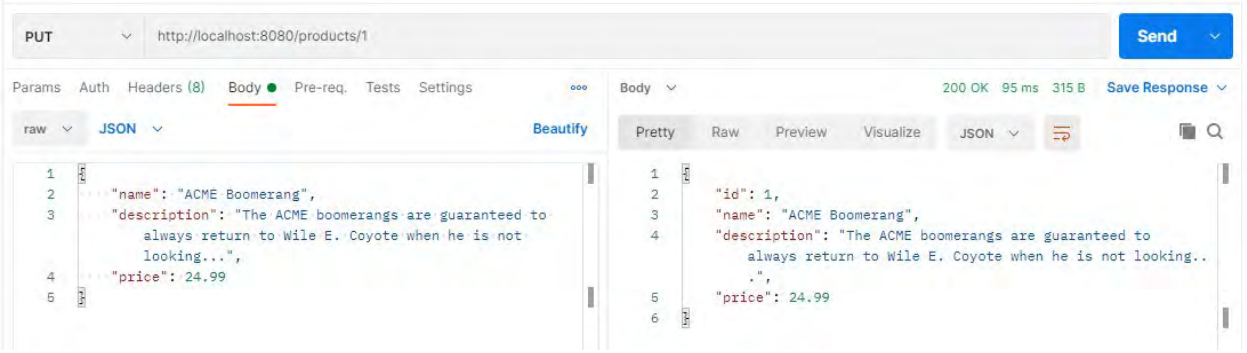
(15) // POST-Products-BuckShot



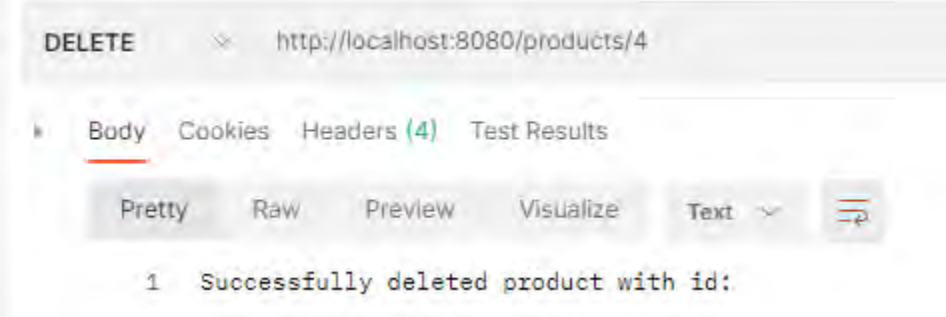
(16) // GET (body/raw/JSON) http://localhost:8080/products



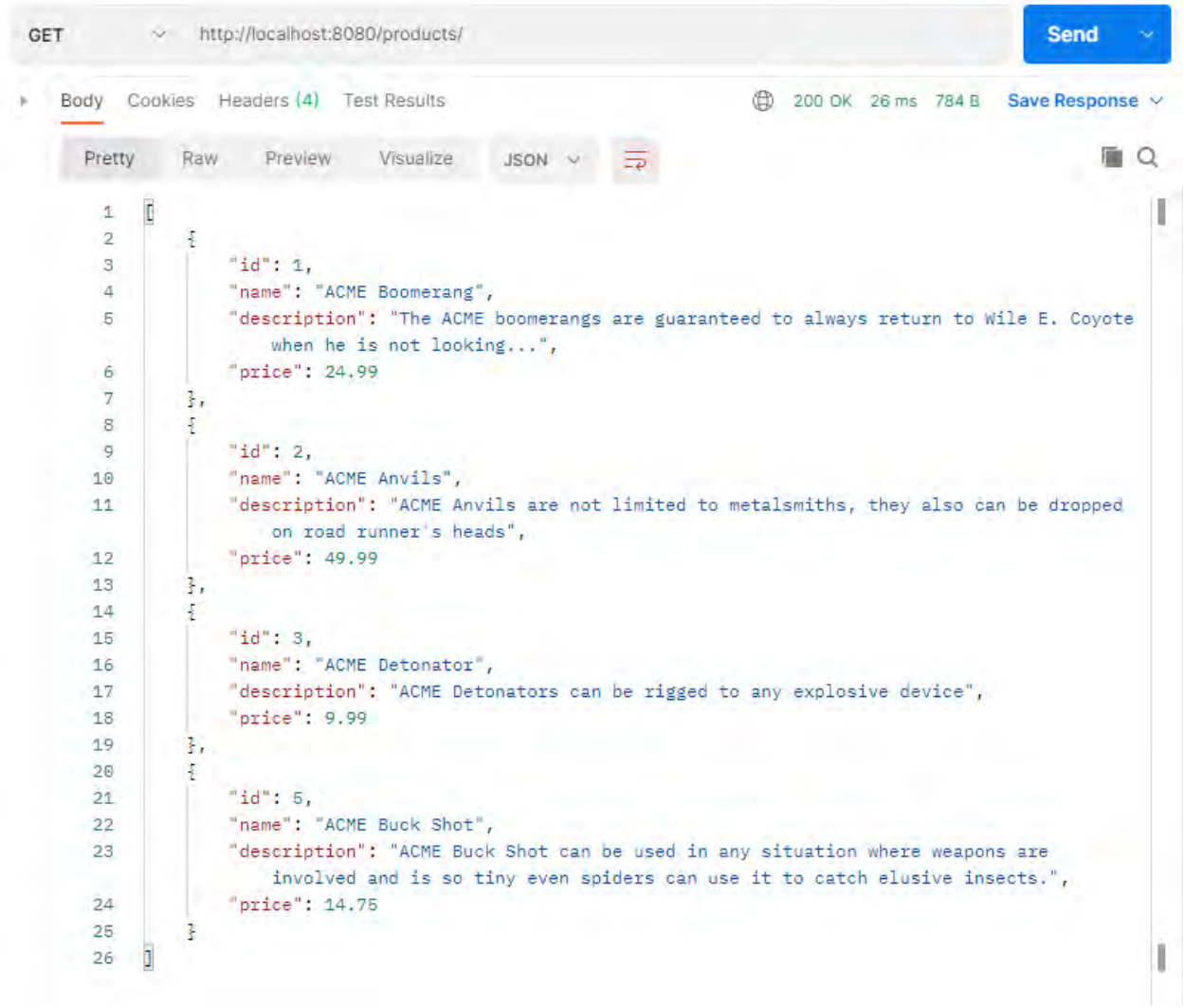
(17) //PUT /products/1



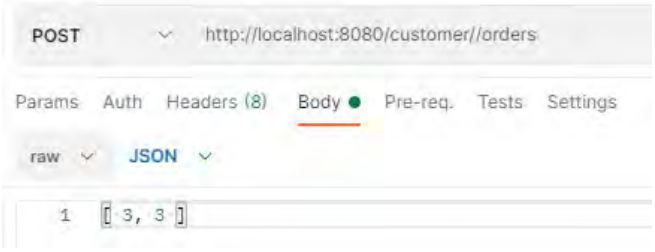
(18) // DELETE /products/4

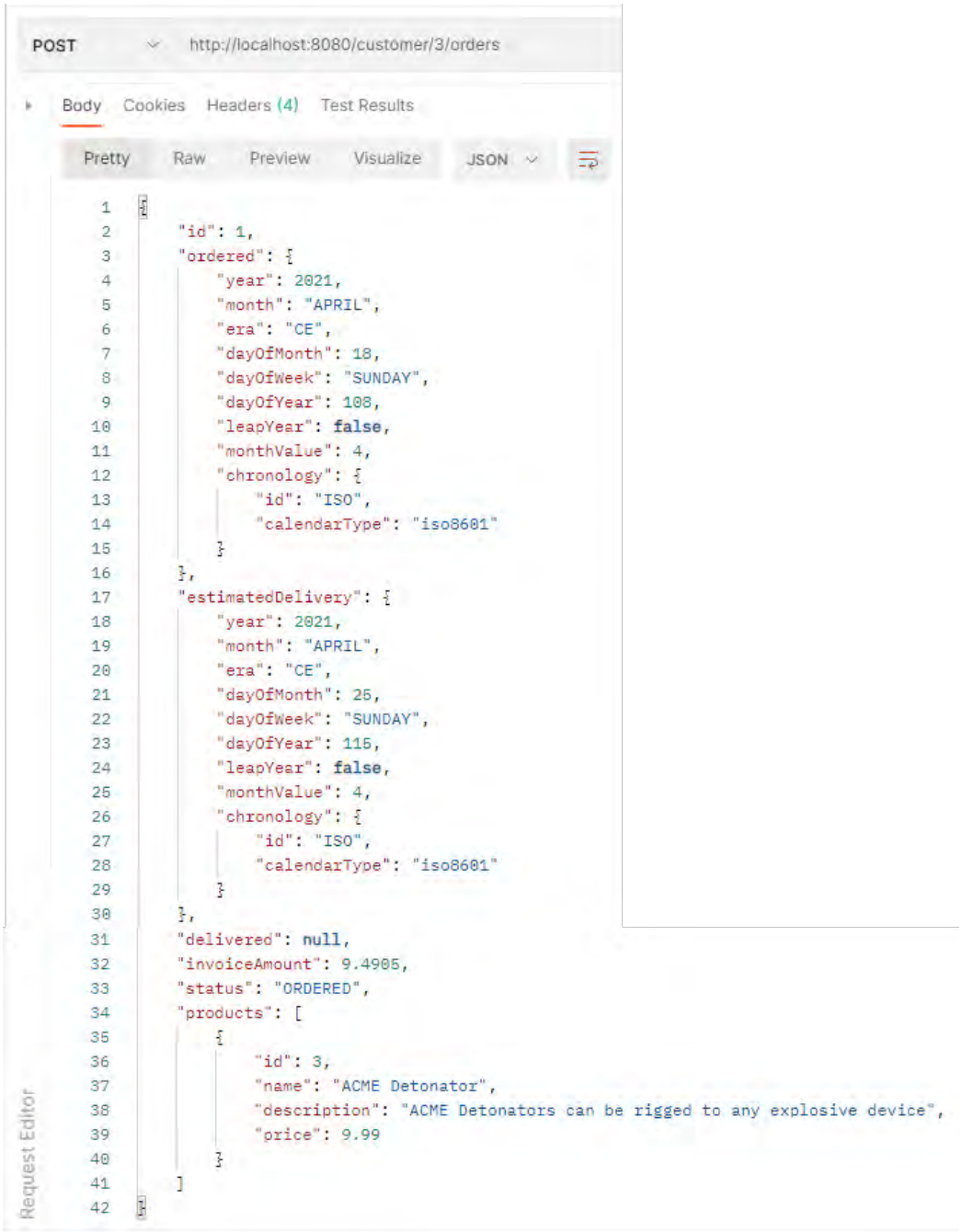


(18) // GET (body/raw/JSON) http://localhost:8080/products

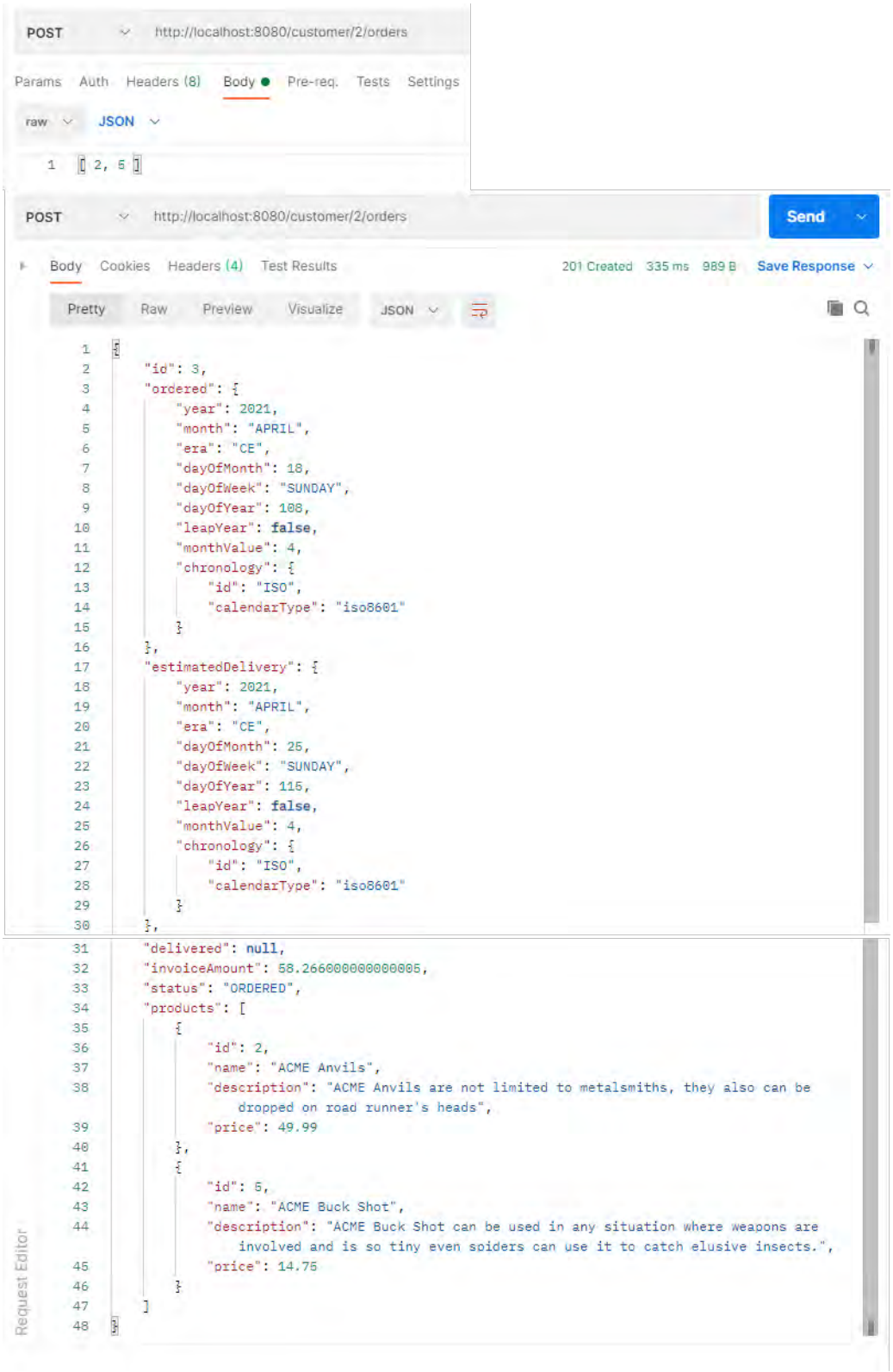


(19) // POST /customers/3/orders -- Wile E. Coyote(3) - Detonator (3) –

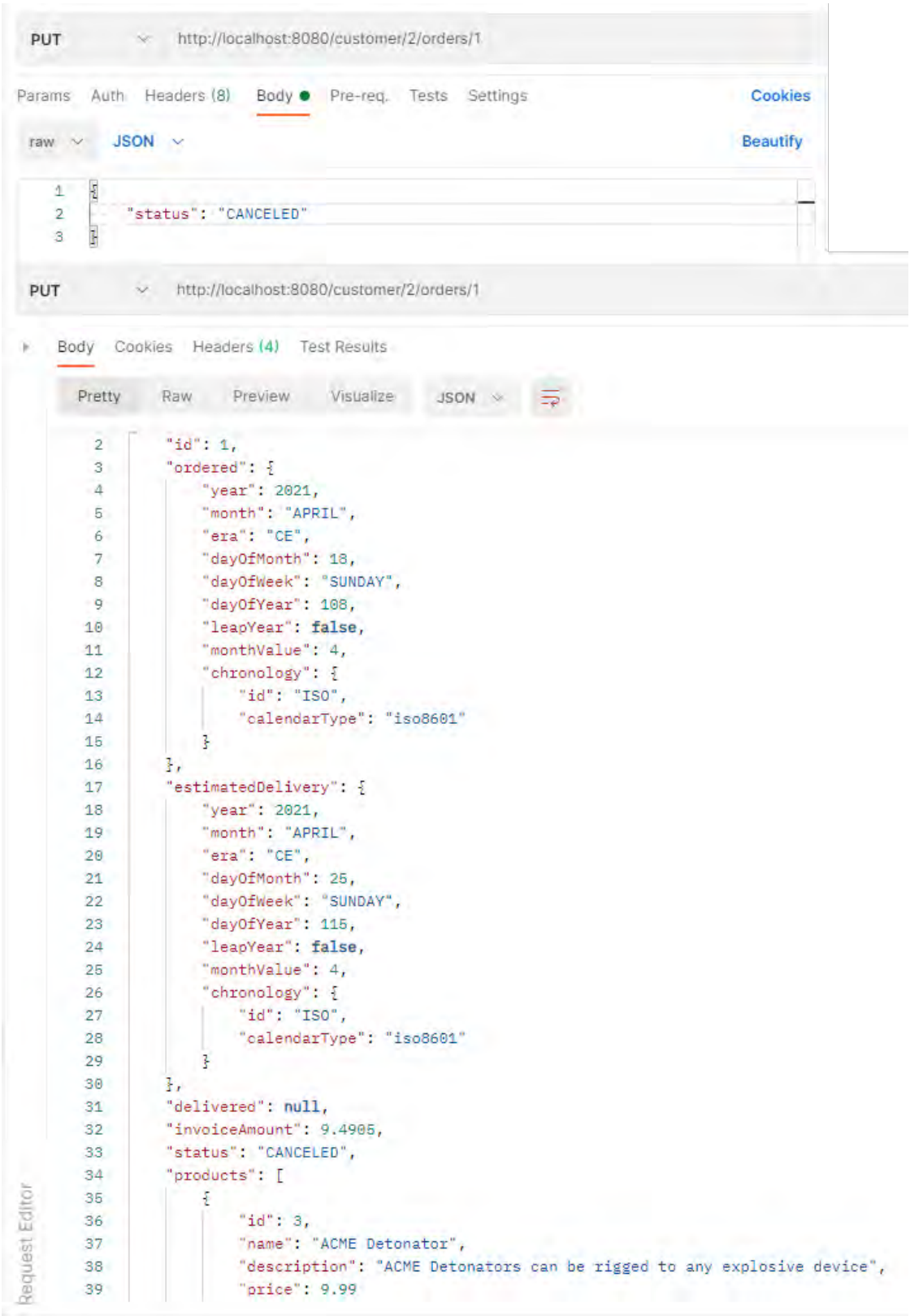




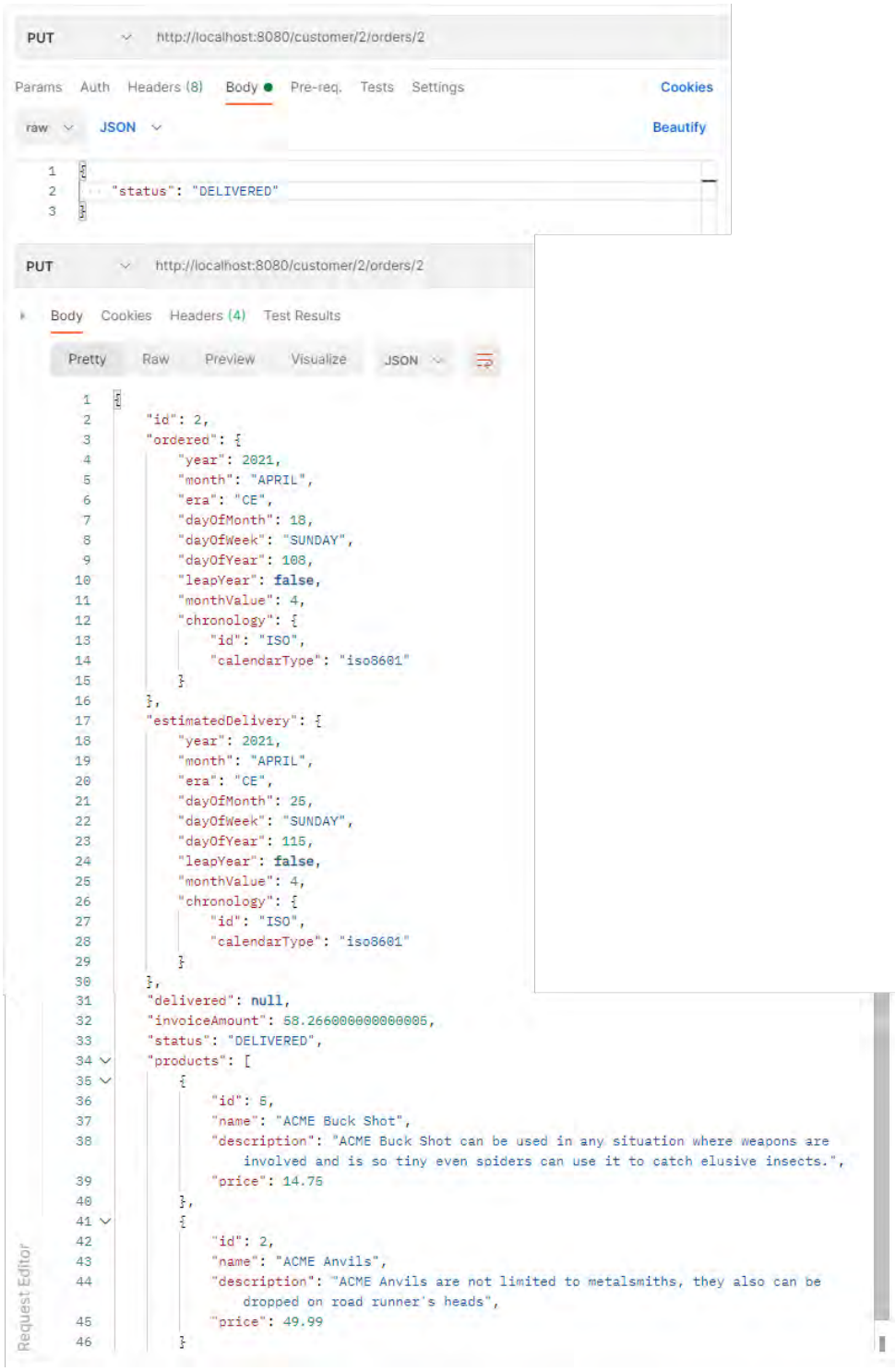
(20) POST /customer/2/orders – Elmer Fudd (2) – Anvils (5)



(21) // PUT /customer/2/orders/1_CANCELED



(22) // PUT /customers/2/orders/5_DELIVERED



URL to GitHub Repository: https://github.com/becje/Springboot_Coding_Assignments