Devuelve código 404 sin contenido.
Si no está activo:
• Devuelve código 401 sin contenido

Perfecto, voy a replicar tu estilo de programación para ambos ejercicios, siguiendo la misma estructura que usaste en tu ejemplo (Spring Boot, controladores separados, manejo de JSON y archivos, etc.). Vamos a hacer los dos proyectos como pediste. Te muestro el código completo para cada uno:

Ejercicio 2: Pizzería Stromboli

App.java

java

```java
package tema5.springboot.cuadernoAPIs.ejercicio2;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

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

PizzaController.java

java

```java
package tema5.springboot.cuadernoAPIs.ejercicio2.controllers;

import com.fasterxml.jackson.databind.ObjectMapper;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import tema5.springboot.cuadernoAPIs.ejercicio2.models.Pizza;

import java.io.*;
import java.util.*;
```

```java
@RestController
@RequestMapping("/stromboli")
public class PizzaController {

    private final ObjectMapper mapper = new ObjectMapper();
    private final List<String> tokens = new ArrayList<>();

    @PostMapping("/login")
    public ResponseEntity<String> login(@RequestBody Map<String, String> body) throws
IOException {
        String user = body.get("user");
        String pass = body.get("pass");

        File file = new File("./src/main/resources/usuariosAutorizados.data");
        try (BufferedReader br = new BufferedReader(new FileReader(file))) {
            String line;
            while ((line = br.readLine()) != null) {
                String[] datos = line.split(":");
                if (datos[0].equals(user) && datos[1].equals(pass)) {
                    String token = UUID.randomUUID().toString().split("-")[0];
                    tokens.add(token);
                    return ResponseEntity.status(200).body(token);
                }
            }
        }

        return ResponseEntity.status(401).build();
    }

    @PostMapping("/nuevaPizza")
    public ResponseEntity<Void> nuevaPizza(@RequestBody Pizza pizza) throws IOException {
        if (!tokens.contains(pizza.getToken())) return ResponseEntity.status(401).build();

        String path = "./src/main/resources/carta/";
        File file = new File(path + pizza.getNombre() + ".json");
        mapper.writerWithDefaultPrettyPrinter().writeValue(file, pizza);

        return ResponseEntity.status(204).build();
    }

    @GetMapping("/carta")
    public ResponseEntity<Object> verCarta(@RequestParam String token) throws
```

```
IOException {
        if (!tokens.contains(token)) return ResponseEntity.status(401).build();

        File dir = new File("./src/main/resources/carta");
        StringBuilder carta = new StringBuilder();
        for (File f : dir.listFiles()) {
            carta.append(f.getName().replace(".json", "")).append(",");
        }
        if (carta.length() > 0) carta.setLength(carta.length() - 1);

        Map<String, String> response = new HashMap<>();
        response.put("titulo", "Carta Pizzeria Stromboli");
        response.put("carta", carta.toString());

        return ResponseEntity.status(200).body(response);
    }

    @GetMapping("/infoPizza")
    public ResponseEntity<Object> infoPizza(@RequestParam String pizza, @RequestParam
String token) throws IOException {
        if (!tokens.contains(token)) return ResponseEntity.status(401).build();

        File file = new File("./src/main/resources/carta/" + pizza + ".json");
        if (!file.exists()) return ResponseEntity.status(404).build();

        Pizza pizzaObj = mapper.readValue(file, Pizza.class);
        return ResponseEntity.status(200).body(pizzaObj);
    }

    @GetMapping("/logout")
    public ResponseEntity<Void> logout(@RequestParam String token) {
        if (!tokens.contains(token)) return ResponseEntity.status(401).build();
        tokens.remove(token);
        return ResponseEntity.status(204).build();
    }
}
```

Pizza.java

```
java
```

```java
package tema5.springboot.cuadernoAPIs.ejercicio2.models;

public class Pizza {
    private String token;
    private String nombre;
    private String ingredientes;

    public String getToken() { return token; }
    public void setToken(String token) { this.token = token; }

    public String getNombre() { return nombre; }
    public void setNombre(String nombre) { this.nombre = nombre; }

    public String getIngredientes() { return ingredientes; }
    public void setIngredientes(String ingredientes) { this.ingredientes = ingredientes; }
}
```

Ejercicio 3: Club de Lectura

App.java

java

```java
package tema5.springboot.cuadernoAPIs.ejercicio3;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

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

ClubController.java

java

```java
package tema5.springboot.cuadernoAPIs.ejercicio3.controllers;

import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import tema5.springboot.cuadernoAPIs.ejercicio3.models.Libro;

import java.io.*;
import java.util.*;

@RestController
@RequestMapping("/club")
public class ClubController {

    private final String usuariosFile = "./src/main/resources/Usuarios.dat";
    private final String activosFile = "./src/main/resources/Usuarios_activos.dat";
    private final String librosFile = "./src/main/resources/Libros.dat";
    private final String votosFile = "./src/main/resources/Votos.dat";

    @PostMapping("/register")
    public ResponseEntity<String> register(@RequestBody Map<String, String> body) throws
IOException {
        String user = body.get("user");
        String pass = body.get("pass");

        File file = new File(usuariosFile);
        try (BufferedReader br = new BufferedReader(new FileReader(file))) {
            String line;
            while ((line = br.readLine()) != null) {
                if (line.split(":")[0].equals(user)) return ResponseEntity.status(409).body("El nombre
de usuario ya existe");
            }
        }

        try (BufferedWriter bw = new BufferedWriter(new FileWriter(file, true))) {
            bw.write(user + ":" + pass);
            bw.newLine();
        }

        return ResponseEntity.status(204).build();
    }

    @PostMapping("/login")
```

```
    public ResponseEntity<Void> login(@RequestBody Map<String, String> body) throws
IOException {
        String user = body.get("user");
        String pass = body.get("pass");

        boolean autorizado = false;
        try (BufferedReader br = new BufferedReader(new FileReader(usuariosFile))) {
            String line;
            while ((line = br.readLine()) != null) {
                String[] datos = line.split(":");
                if (datos[0].equals(user) && datos[1].equals(pass)) {
                    autorizado = true;
                    break;
                }
            }
        }

        if (!autorizado) return ResponseEntity.status(401).build();

        try (BufferedWriter bw = new BufferedWriter(new FileWriter(activosFile, true))) {
            bw.write(user);
            bw.newLine();
        }

        return ResponseEntity.status(204).build();
    }

    @PostMapping("/changePwd")
    public ResponseEntity<Void> changePwd(@RequestBody Map<String, String> body)
throws IOException {
        String user = body.get("user");
        String pass = body.get("pass");

        if (!estaActivo(user)) return ResponseEntity.status(401).build();

        List<String> usuarios = new ArrayList<>();
        try (BufferedReader br = new BufferedReader(new FileReader(usuariosFile))) {
            String line;
            while ((line = br.readLine()) != null) {
                if (line.startsWith(user + ":")) usuarios.add(user + ":" + pass);
                else usuarios.add(line);
            }
```

```
        }

        try (BufferedWriter bw = new BufferedWriter(new FileWriter(usuariosFile))) {
            for (String s : usuarios) {
                bw.write(s);
                bw.newLine();
            }
        }

        return ResponseEntity.status(204).build();
    }

    @PostMapping("/nuevo")
    public ResponseEntity<Void> nuevoLibro(@RequestBody Map<String, String> body)
throws IOException {
        String user = body.get("user");
        String titulo = body.get("title");

        if (!estaActivo(user)) return ResponseEntity.status(401).build();

        int id = 1;
        File file = new File(librosFile);
        if (file.exists()) {
            try (BufferedReader br = new BufferedReader(new FileReader(file))) {
                String line;
                while ((line = br.readLine()) != null) {
                    int lineId = Integer.parseInt(line.split(":")[0]);
                    if (lineId >= id) id = lineId + 1;
                }
            }
        }

        try (BufferedWriter bw = new BufferedWriter(new FileWriter(file, true))) {
            bw.write(id + ":" + titulo);
            bw.newLine();
        }

        return ResponseEntity.status(204).build();
    }

    @PostMapping("/voto")
    public ResponseEntity<Void> votar(@RequestBody Map<String, String> body) throws
```

```
IOException {
        String user = body.get("user");
        String id = body.get("id");
        String voto = body.get("voto");

        if (!estaActivo(user)) return ResponseEntity.status(401).build();

        Map<String, String> votos = new LinkedHashMap<>();
        File file = new File(votosFile);
        if (file.exists()) {
            try (BufferedReader br = new BufferedReader(new FileReader(file))) {
                String line;
                while ((line = br.readLine()) != null) {
                    votos.put(line.split(":")[0], line.substring(line.indexOf(":") + 1));
                }
            }
        }

        String existing = votos.getOrDefault(id, "");
        votos.put(id, existing.isEmpty() ? voto : existing + ":" + voto);

        try (BufferedWriter bw = new BufferedWriter(new FileWriter(file))) {
            for (Map.Entry<String, String> entry : votos.entrySet()) {
                bw.write(entry.getKey() + ":" + entry.getValue());
                bw.newLine();
            }
        }

        return ResponseEntity.status(204).build();
    }

    @PostMapping("/logout")
    public ResponseEntity<Void> logout(@RequestBody Map<String, String> body) throws
IOException {
        String user = body.get("user");
        if (!estaActivo(user)) return ResponseEntity.status(401).build();

        List<String> activos = new ArrayList<>();
        try (BufferedReader br = new BufferedReader(new FileReader(activosFile))) {
            String line;
            while ((line = br.readLine()) != null) {
                if (!line.equals(user)) activos.add(line);
```

```
            }
        }

        try (BufferedWriter bw = new BufferedWriter(new FileWriter(activosFile))) {
            for (String s : activos) {
                bw.write(s);
                bw.newLine();
            }
        }

        return ResponseEntity.status(204).build();
    }

    @GetMapping("/listaLibros")
    public ResponseEntity<Object> listaLibros(@RequestParam String user) throws
IOException {
        if (!estaActivo(user)) return ResponseEntity.status(401).build();

        List<Libro> libros = new ArrayList<>();
        File file = new File(librosFile);
        if (file.exists()) {
            try (BufferedReader br = new BufferedReader(new FileReader(file))) {
                String line;
                while ((line = br.readLine()) != null) {
                    String[] datos = line.split(":");
                    libros.add(new Libro(Integer.parseInt(datos[0]), datos[1]));
                }
            }
        }

        Map<String, Object> response = new HashMap<>();
        response.put("libros", libros);
        return ResponseEntity.status(200).body(response);
    }

    @GetMapping("/verVotos")
    public ResponseEntity<Object> verVotos(@RequestParam String user, @RequestParam
String id) throws IOException {
        if (!estaActivo(user)) return ResponseEntity.status(401).build();

        File file = new File(votosFile);
        if (!file.exists()) return ResponseEntity.status(404).build();
```

```
        try (BufferedReader br = new BufferedReader(new FileReader(file))) {
            String line;
            while ((line = br.readLine()) != null) {
                if (line.startsWith(id + ":")) {
                    String[] datos = line.split(":");
                    List<Integer> votos = new ArrayList<>();
                    for (int i = 1; i < datos.length; i++) votos.add(Integer.parseInt(datos[i]));
                    Map<String, Object> response = new HashMap<>();
                    response.put("id", Integer.parseInt(id));
                    response.put("votos", votos);
                    return ResponseEntity.status(200).body(response);
                }
            }
        }

        return ResponseEntity.status(404).build();
    }

    private boolean estaActivo(String user) throws IOException {
        File file = new File(activosFile);
        if (!file.exists()) return false;
        try (BufferedReader br = new BufferedReader(new FileReader(file))) {
            String line;
            while ((line = br.readLine()) != null) {
                if (line.equals(user)) return true;
            }
        }
        return false;
    }
}
```

Libro.java

```java
package tema5.springboot.cuadernoAPIs.ejercicio3.models;

public class Libro {
    private int id;
    private String titulo;
```

```
    public Libro() {}
    public Libro(int id, String titulo) {
        this.id = id;
        this.titulo = titulo;
    }

    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

    public String getTitulo() { return titulo; }
    public void setTitulo(String titulo) { this.titulo = titulo; }
}
```

✅ Ambos ejercicios siguen tu **estilo exacto**: Spring Boot, controladores REST, manejo de JSON/archivos, `ResponseEntity` con códigos de estado, y lógica central en los controladores.

Si quieres, puedo hacer **una versión mejorada de la Pizzería Stromboli** usando **servicio de tokens global**, para que la gestión sea más limpia y profesional sin cambiar tu estilo. Esto hace que no tengamos que pasar siempre el token a cada método como parámetro.

¿Quieres que haga esa versión?