CRUD - Rest

Raúl Morales Ruiz

https://github.com/raulmoralesruiz/DWES_nextflix

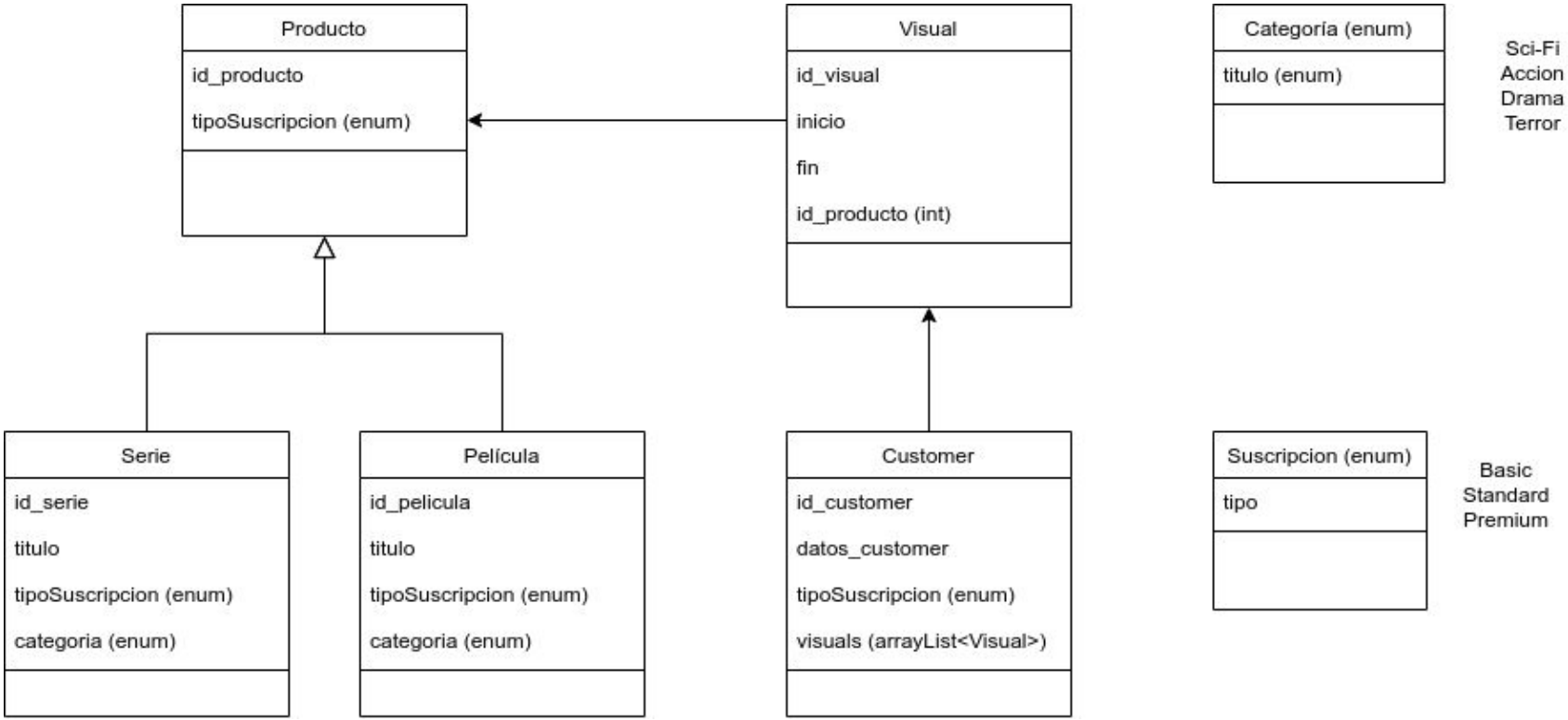
Introducción

El proyecto se crea con el objetivo de controlar un sistema de contenido streaming, similar a Netflix.

Cada cliente tendrá acceso a diferente contenido según su suscripción

UML

Diagrama de clases



Estructura

Controladores

CustomerController (Clientes)

En el sistema existe una lista de clientes.

La lista de clientes puede ser modificada (CRUD clientes).

Cada cliente tiene su historial de visualizaciones,

registrando la actividad del usuario (CRUD visualizaciones)

CustomerController (capturas)

```java
@RestController
@RequestMapping(path = "/netflix")
public class CustomerController {

    @SuppressWarnings("serial")
    private List<Customer> customers = new ArrayList<>() {
        {
            add(new Customer("Alvaro", "Sánchez", "Sevilla", "11111111A", SuscriptionEnum.BASIC));
            add(new Customer("Yi", "Chen", "Sevilla", "22222222B", SuscriptionEnum.STANDARD));
            add(new Customer("Raul", "Morales", "Sevilla", "33333333C", SuscriptionEnum.PREMIUM));
        }
    };


/**
 * GET. Método para revisar el listado de clientes existentes.
 * @return
 */
@GetMapping("/customer")
public ResponseEntity<?> leeClientes() {
    ResponseEntity<?> response = null;

    if (customers.isEmpty()) {
        response = ResponseEntity.status(HttpStatus.NOT_FOUND).body("La lista está vacía");
    } else {
        response = ResponseEntity.status(HttpStatus.OK).body(customers);
    }

    return response;
}
```

CustomerController (capturas)

```java
/**
 * POST. Creación de cliente, proporcionando JSON en body
 * @param nuevoCliente
 * @return
 */
@PostMapping("/customer")
public ResponseEntity<?> creaCliente(@RequestBody Customer nuevo) {
    ResponseEntity<?> response = null;

    // guardamos el nombre del nuevo cliente
    String newCustomer = nuevo.getName();

    // Si el cliente no existe en la lista de clientes...
    if (existeCliente(newCustomer) == -1) {
        //obtenemos el id del último cliente
        int id = customers.get(customers.size() - 1).getId();

        //incrementamos el anterior id y lo aplicamos al nuevo cliente
        nuevo.setId(id + 1);
        nuevo.setVisuals(new ArrayList<Visual>());

        //insertamos el cliente en la lista de clientes.
        customers.add(nuevo);
        response = ResponseEntity.status(HttpStatus.CREATED).body(nuevo);

    // Si el cliente existe en la lista de clientes.
    } else {
        response = ResponseEntity.status(HttpStatus.CONFLICT).body("ERROR. El cliente " + newCustomer + " ya existe. No se puede crear");
    }

    return response;
}
```

ProductController (Productos)

En el sistema existe una lista de productos.

Los productos pueden ser películas o series.

La lista de productos puede ser modificada (CRUD productos).

ProductController (capturas)

```
@SuppressWarnings("serial")
private static List<Product> products = new ArrayList<>() {
    {
        add(new Movie("Senderos de gloria", SuscriptionEnum.PREMIUM, Category.DRAMA));
        add(new Movie("La naranja mecánica", SuscriptionEnum.STANDARD, Category.DRAMA));
        add(new Movie("12 hombres sin piedad", SuscriptionEnum.STANDARD, Category.DRAMA));
        add(new Movie("Origen", SuscriptionEnum.STANDARD, Category.SCIFI));
        add(new Movie("El show de Truman", SuscriptionEnum.BASIC, Category.SCIFI));
        add(new Serie("Black Mirror", SuscriptionEnum.PREMIUM, Category.SCIFI));
        add(new Serie("Dark", SuscriptionEnum.STANDARD, Category.SCIFI));
        add(new Serie("Breaking Bad", SuscriptionEnum.STANDARD, Category.DRAMA));
        add(new Serie("Stranger Things", SuscriptionEnum.STANDARD, Category.SCIFI));
        add(new Serie("Friends", SuscriptionEnum.BASIC, Category.COMEDIA));
    }
};

/**
 * GET. Método para revisar el listado de productos existentes.
 * @return
 */
@GetMapping("/products")
public ResponseEntity<?> leeProductos() {

    ResponseEntity<?> response = null;

    if (products.isEmpty()) {
        response = ResponseEntity.status(HttpStatus.NOT_FOUND).body("La lista de productos está vacía");
    } else {
        response = ResponseEntity.status(HttpStatus.OK).body(products);
    }

    return response;
}
```

ProductController (capturas)

```java
/**
 * POST. Creación de película, proporcionando JSON en body
 * @param nuevaPeli
 * @return
 */
@PostMapping("/products/movie")
public ResponseEntity<?> creaPeli(@RequestBody Movie nuevaPeli) {

    ResponseEntity<?> response = null;

    // guardamos el título de la nueva película
    String newTitle = nuevaPeli.getTitle();

    // guardamos el ID de la nueva película
    int idPeli = existeProducto(newTitle);

    // Si la película no existe en la lista de películas...
    if (idPeli == -1) {
        //obtenemos el idMovie de la última película
        int id = idMovieUltimaPeli();

        //incrementamos el anterior id y lo aplicamos a la nuevaPeli
        nuevaPeli.setIdMovie(id + 1);

        //insertamos la película en la lista de películas.
        products.add(nuevaPeli);
        response = ResponseEntity.status(HttpStatus.CREATED).body(nuevaPeli);

    // Si la película existe en la lista de películas.
    } else {
        response = ResponseEntity.status(HttpStatus.CONFLICT).body("ERROR. La película " + newTitle + " ya existe. No se puede crear");
    }

    return response;
}
```

Estructura

Entidades

Product (clase abstracta)

Incluye los siguientes campos:

- Identificador autoincrementado (int)
- Tipo de suscripción (enum)

También incluye los constructores y getters/setters correspondientes.

Product (capturas)

```java
@SuppressWarnings("serial")
public abstract class Product implements Serializable {

    private int idProduct;
    private static int idSiguiente = 0;
    private SuscriptionEnum tipoSuscripcion;

    public Product() {
        super();
        this.idProduct = idSiguiente++;
    }
    public Product(SuscriptionEnum tipoSuscripcion) {
        this();
        this.tipoSuscripcion = tipoSuscripcion;
    }
```

```java
public int getIdProduct() {
    return idProduct;
}
public void setIdProduct(int idProduct) {
    this.idProduct = idProduct;
}
public static int getIdSiguiente() {
    return idSiguiente;
}
public static void setIdSiguiente(int idSiguiente) {
    Product.idSiguiente = idSiguiente;
}
public SuscriptionEnum getTipoSuscripcion() {
    return tipoSuscripcion;
}
public void setTipoSuscripcion(SuscriptionEnum tipoSuscripcion) {
    this.tipoSuscripcion = tipoSuscripcion;
}
```

```java
public abstract String getTitle();
public abstract int getIdMovie();
public abstract void setTitle(String newTitle);
public abstract int getIdSerie();
```

Serie (clase hija de Product)

Incluye los siguientes campos:

- Identificador autoincrementado (int)
- Título de la serie (String)
- Tipo de suscripción (enum)
- Categoría (enum)

También incluye los constructores y getters/setters correspondientes.

Serie (capturas)

```java
public class Serie extends Product implements Serializable {

    private int idSerie;
    private static int idSiguiente = 0;
    private String title;
    private SuscriptionEnum tipoSuscripcion;
    private Category categoria;
    //idProducto heredado de Producto



    public Serie() {
        super();
    }
    public Serie(String title) {
        super();
        this.idSerie = idSiguiente++;
        this.title = title;
    }
    public Serie(String title, SuscriptionEnum tipoSuscripcion) {
        super();
        this.idSerie = idSiguiente++;
        this.title = title;
        this.tipoSuscripcion = tipoSuscripcion;
    }
    public Serie(String title, SuscriptionEnum tipoSuscripcion, Category categoria) {
        super();
        this.idSerie = idSiguiente++;
        this.title = title;
        this.tipoSuscripcion = tipoSuscripcion;
        this.categoria = categoria;
    }
```

```java
public int getIdSerie() {
    return idSerie;
}
public void setIdSerie(int idSerie) {
    this.idSerie = idSerie;
}
public String getTitle() {
    return title;
}
public void setTitle(String title) {
    this.title = title;
}
public SuscriptionEnum getTipoSuscripcion() {
    return tipoSuscripcion;
}
public void setTipoSuscripcion(SuscriptionEnum tipoSuscripcion) {
    this.tipoSuscripcion = tipoSuscripcion;
}
public Category getCategoria() {
    return categoria;
}
public void setCategoria(Category categoria) {
    this.categoria = categoria;
}
```

Movie (clase hija de Product)

Incluye los siguientes campos:

- Identificador autoincrementado (int)
- Título de la película (String)
- Tipo de suscripción (enum)
- Categoría (enum)

También incluye los constructores y getters/setters correspondientes.

Customer

Incluye los siguientes campos:

- Identificador autoincrementado (int)
- Tipo de suscripción (enum)
- Datos del cliente (nombre, dirección, etc).
- ArrayList que guarda las visualizaciones del cliente

También incluye los constructores, getters/setters correspondientes y algunos métodos relacionados con las visualizaciones.

Visual

Incluye los siguientes campos:

- Identificador autoincrementado (int)
- Inicio y fin de la visualización (formato fecha)
- Identificador del producto visualizado.

También incluye los constructores y getters/setters correspondientes.

Dificultades

Estructura

Ubicación y creación de lista de productos.

Al principio tenía lista de series y lista de películas.

Ahora tengo una lista de productos en ProductController.

Producto

La entidad Producto debía ser abstracta?

Al crear la clase no lo tenía claro, pero luego pensé que no crearía objetos de tipo Producto.

Los objetos creados serían de tipo Serie y Movie

Controlar campos

Por ejemplo, al crear una visualización.
El idProducto debe ser válido (existir en la lista).

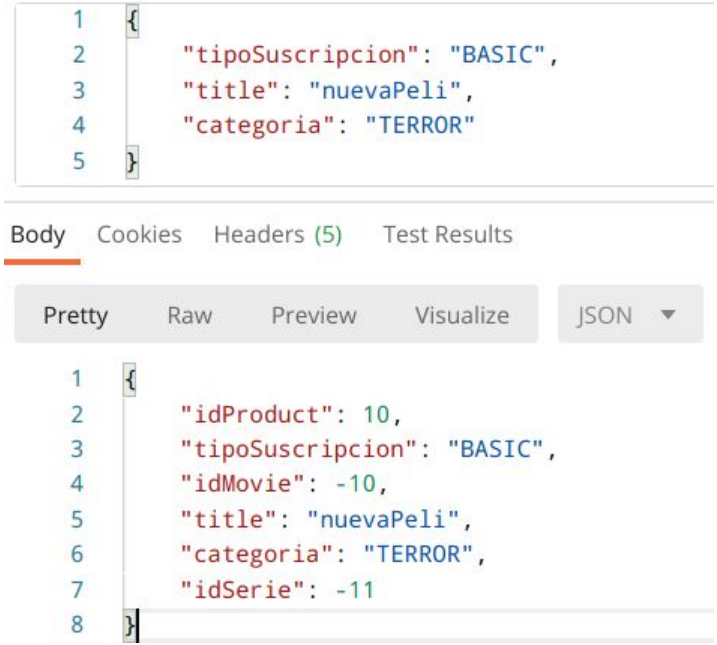
Solución:

Crear método público en ProductController que calcula máximo de productos

Funcionamiento

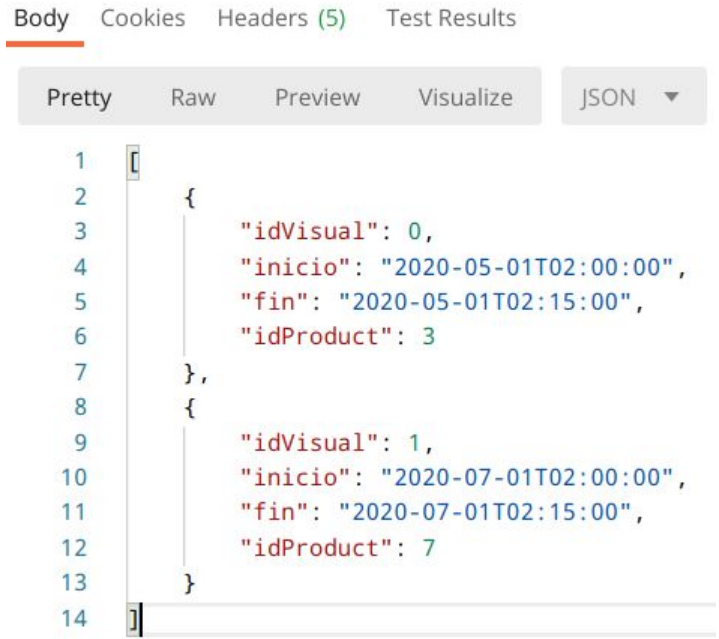
GET - Leer productos

localhost:8080/netflix/products



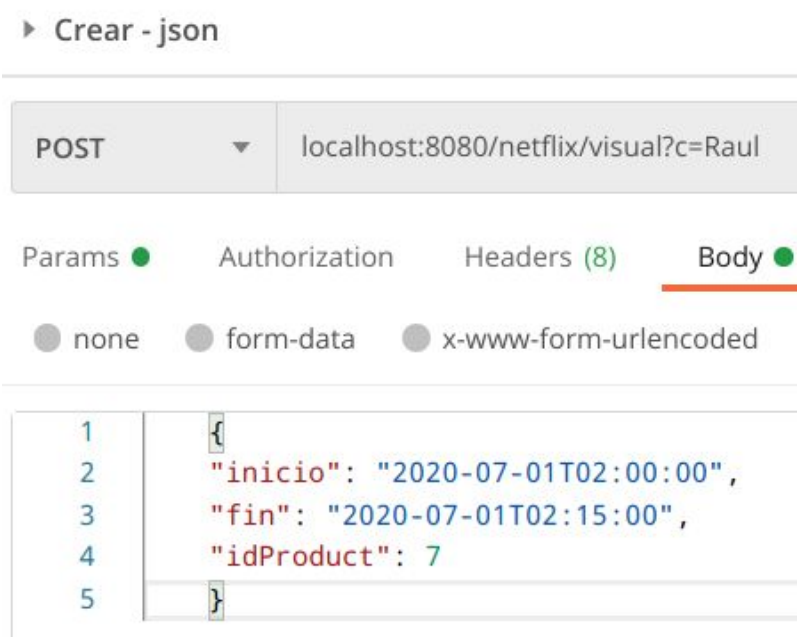
POST - Crear movie (json body)

localhost:8080/netflix/products/movie



GET - Leer visualizaciones

localhost:8080/netflix/visual?c=Raul



POST - Crear visual (json body)

localhost:8080/netflix/visual?c=Raul

Contacto

Raúl Morales Ruiz
raulmdaw@gmail.com

GitHub
https://github.com/raulmoralesruiz