CRUD - Rest

Raúl Morales Ruiz

https://github.com/raulmoralesruiz/DWES_nextflix

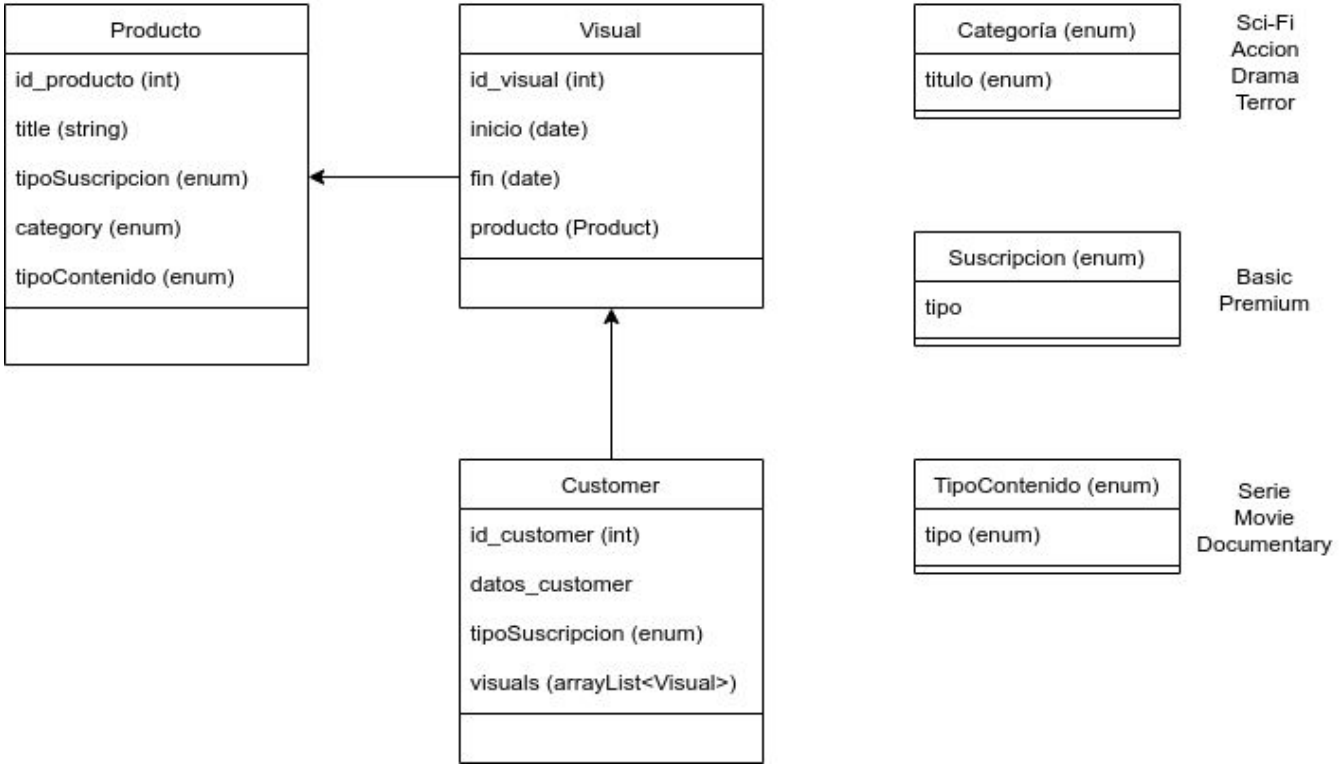
Introducción

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)

```
@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.PREMIUM));
        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, series o documentales.

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

ProductController (capturas)

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

        //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

Incluye los siguientes campos:

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

También incluye los constructores y getters/setters correspondientes.

Product (capturas)

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

    private int idProduct;
    private static int idSiguiente = 0;
    private String title;
    private Category categoria;
    private TipoContenido tipoContenido;
    private SuscriptionEnum tipoSuscripcion;

    public Product() {
        super();
        this.idProduct = idSiguiente++;
    }
    public Product(SuscriptionEnum tipoSuscripcion) {
        this();
        this.tipoSuscripcion = tipoSuscripcion;
    }
    public Product(String title, Category categoria,
            TipoContenido tipoContenido,
            SuscriptionEnum tipoSuscripcion) {
        super();
        this.idProduct = idSiguiente++;
        this.title = title;
        this.categoria = categoria;
        this.tipoContenido = tipoContenido;
        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 String getTitle() {
    return title;
}
public void setTitle(String title) {
    this.title = title;
}
public Category getCategoria() {
    return categoria;
}
public void setCategoria(Category categoria) {
    this.categoria = categoria;
}
public TipoContenido getTipoContenido() {
    return tipoContenido;
}
public void setTipoContenido(TipoContenido tipoContenido) {
    this.tipoContenido = tipoContenido;
}
public SuscriptionEnum getTipoSuscripcion() {
    return tipoSuscripcion;
}
public void setTipoSuscripcion(SuscriptionEnum tipoSuscripcion) {
    this.tipoSuscripcion = tipoSuscripcion;
}
```

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.

Customer (capturas)

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

    private int id;
    private static int idSiguiente = 0;
    private String name;
    private String surname;
    private LocalDate birthdate;
    private String address;
    private String city;
    private String dni;
    private String country;
    private String mobileNumber;
    private String gender;
    private SuscriptionEnum tipoSuscripcion;
    private ArrayList<Visual> visuals;

public Customer(String name, String surname,
        String city, String dni,
        SuscriptionEnum tipoSuscripcion) {
    super();
    this.id = idSiguiente++;
    this.name = name;
    this.surname = surname;
    this.city = city;
    this.dni = dni;
    this.tipoSuscripcion = tipoSuscripcion;
    this.visuals = new ArrayList<Visual>();
}
```

```java
public int getId() {
    return id;
}
public void setId(int id) {
    this.id = id;
}
public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}
public String getSurname() {
    return surname;
}
public void setSurname(String surname) {
    this.surname = surname;
}
public LocalDate getBirthdate() {
    return birthdate;
}
public void setBirthdate(LocalDate birthdate) {
    this.birthdate = birthdate;
}
public String getAddress() {
    return address;
}
public void setAddress(String address) {
    this.address = address;
}
public String getCity() {
    return city;
}
public void setCity(String city) {
    this.city = city;
}
```

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.

Visual (capturas)

```java
private int idVisual;
private static int idSiguiente = 0;
private LocalDateTime inicio;
private LocalDateTime fin;
private Product producto;


public Visual() {
    super();
}
public Visual(LocalDateTime inicio,
        LocalDateTime fin, Product producto) {
    super();
    this.idVisual = idSiguiente++;
    this.inicio = inicio;
    this.fin = fin;
    this.producto = producto;
}
```

```java
public int getIdVisual() {
    return idVisual;
}
public void setIdVisual(int idVisual) {
    this.idVisual = idVisual;
}
public LocalDateTime getInicio() {
    return inicio;
}
public void setInicio(LocalDateTime inicio) {
    this.inicio = inicio;
}
public LocalDateTime getFin() {
    return fin;
}
public void setFin(LocalDateTime fin) {
    this.fin = fin;
}
public Product getProducto() {
    return producto;
}
public void setProducto(Product producto) {
    this.producto = producto;
}
```

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.

Entidad 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

Entidades Serie y Movie

Es posible mover todo el contenido a Producto?

Creado tipoContenido (enum) para distinguir si el producto es
Documental, Serie o Película.

Se eliminan las clases Serie y Movie,
evitando conflictos con los id específicos de cada clase (idSerie y idMovie)

Controlar campos

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

Solución:

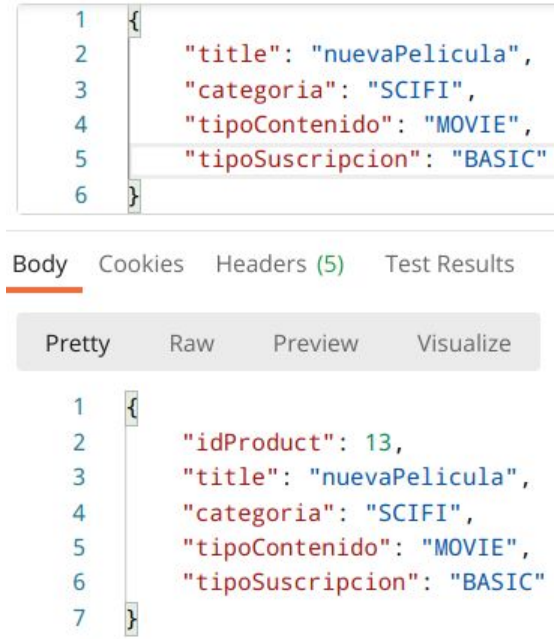
Crear método público en ProductController que calcula máximo de productos

Funcionamiento

```
[
    {
        "idProduct": 0,
        "title": "Senderos de gloria",
        "categoria": "DRAMA",
        "tipoContenido": "MOVIE",
        "tipoSuscripcion": "PREMIUM"
    },
    {
        "idProduct": 1,
        "title": "La naranja mecánica",
        "categoria": "DRAMA",
        "tipoContenido": "MOVIE",
        "tipoSuscripcion": "BASIC"
    },
```

GET - Leer productos

localhost:8080/netflix/products



```
{
    "title": "nuevaPelicula",
    "categoria": "SCIFI",
    "tipoContenido": "MOVIE",
    "tipoSuscripcion": "BASIC"
}
```

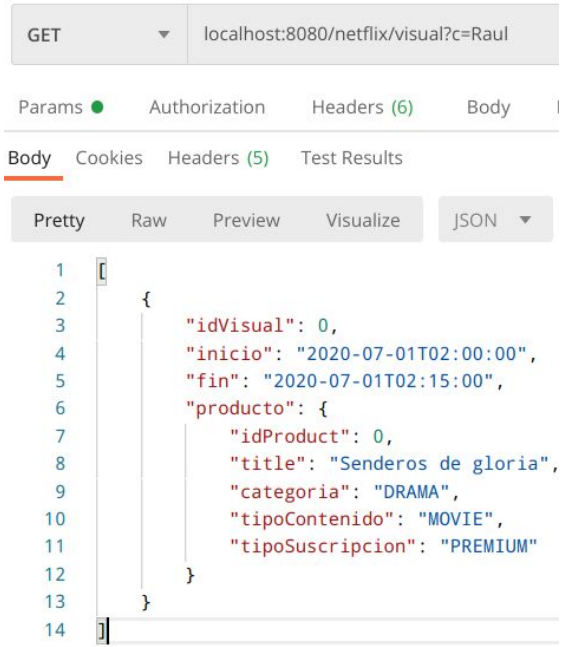
Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize

```
{
    "idProduct": 13,
    "title": "nuevaPelicula",
    "categoria": "SCIFI",
    "tipoContenido": "MOVIE",
    "tipoSuscripcion": "BASIC"
}
```

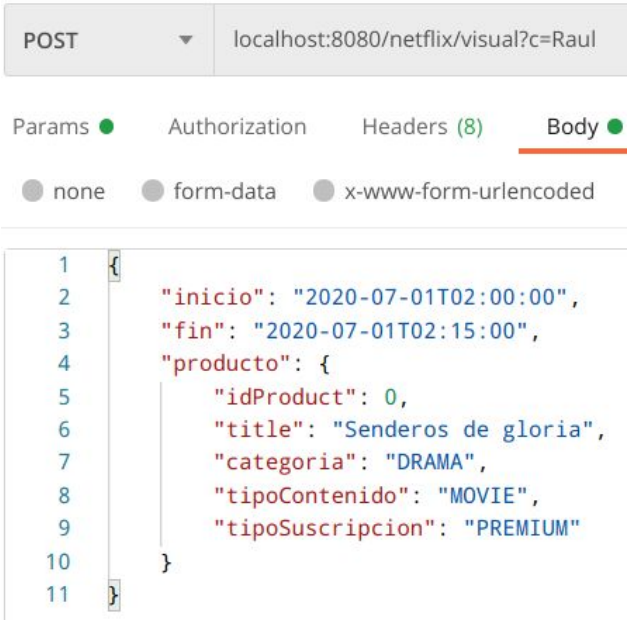
POST - Crear producto (body)

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