FilmsBuster

Objetivo

Desenvolver um conjunto de micros serviços para permitir o gerenciamento de um catálogo de filmes para aluguel online.

Micros Serviços

ms-region

Micro serviço para gerenciar as informações de país e cidade

Especificação Country

- Controller: CountryController com path /country
    - POST: nome operação create e receber como parâmetro o domain
    - PUT: nome operação update e receber como parâmetro o domain
    - DELETE path /{id}: nome operação update e receber como parâmetro o Id do domain
    - GET: nome operação findAll sem parâmetros e retornar uma lista do domain
    - GET path /{id}: nome operação findById parâmetro Id domain language e retornar o domain correspondente ao Id informado
- Services: CountryService
    - Método save: receber um domain do POST e incluir no Map em memória
        - Regras
            - se o Id e/ou Nome existirem retornar um erro 400
            - Id deve ser maior que zero
            - Nome deve conter no mínimo 5 caracteres e somente letras, não pode conter números e caracteres especiais e não pode ser vazio
    - Método update: receber um domain do UPDATE e atualizar no Map em memória
        - Regras
            - Nome existir em um Id diferente retornar um erro 400
            - Nome deve conter no mínimo 5 caracteres e somente letras, não pode conter números e caracteres especiai
    - Método delete: remover do Map em memória o id correspondente a partir do DELETE
    - Método getAll: retornar todos os domains do map para o GET
    - Método getById: retornar o domain do map de acordo com o Id informado para o GET
- Domain: Country
    - id: Int -> language_id
    - name: String -> name
    - lastUpdate: LocalDatetime -> last_update

ms-type (Monica)

Micro serviço para gerenciar as informações de idioma e categoria de filmes
Tabelas: language e category

Especificação Language

- Controller: LanguageController com path /language
    - POST: nome operação create e receber como parâmetro o domain
    - PUT: nome operação update e receber como parâmetro o domain
    - DELETE path /{id}: nome operação update e receber como parâmetro o Id do domain
    - GET: nome operação findAll sem parâmetros e retornar uma lista do domain
    - GET path /{id}: nome operação findById parâmetro Id domain language e retornar o domain correspondente ao Id informado
- Services: LanguageService
    - Método save: receber um domain do POST e incluir no Map em memória
        - Regras
            - se o Id e/ou Nome existirem retornar um erro 400
            - Id deve ser maior que zero
            - Nome deve conter no mínimo 5 caracteres e somente letras, não pode conter números e caracteres especiais e não pode ser vazio
    - Método update: receber um domain do UPDATE e atualizar no Map em memória
        - Regras
            - Nome existir em um Id diferente retornar um erro 400
            - Nome deve conter no mínimo 5 caracteres e somente letras, não pode conter números e caracteres especiai
    - Método delete: remover do Map em memória o id correspondente a partir do DELETE
    - Método getAll: retornar todos os domains do map para o GET
    - Método getById: retornar o domain do map de acordo com o Id informado para o GET
- Domain: Language
    - id: Int -> language_id
    - name: String -> name
    - lastUpdate: LocalDatetime -> last_update

Especificação Category

- Controller: CategoryController com path /category
    - POST: nome operação create e receber como parâmetro o domain
    - PUT: nome operação update e receber como parâmetro o domain
    - DELETE path /{id}: nome operação update e receber como parâmetro o Id do domain
    - GET: nome operação findAll sem parâmetros e retornar uma lista do domain
    - GET path /{id}: nome operação findById parâmetro Id domain language e retornar o domain correspondente ao Id informado

- Services: CategoryService
    - Método save: receber um domain do POST e incluir no Map em memória
        - Regras
            - se o Id e/ou Nome existirem retornar um erro 400
            - Id deve ser maior que zero
            - Nome deve conter no mínimo 5 caracteres e somente letras, não pode conter números e caracteres especiais e não pode ser vazio
    - Método update: receber um domain do UPDATE e atualizar no Map em memória
        - Regras
            - Nome existir em um Id diferente retornar um erro 400
            - Nome deve conter no mínimo 5 caracteres e somente letras, não pode conter números e caracteres especiai
    - Método delete: remover do Map em memória o id correspondente a partir do DELETE
    - Método getAll: retornar todos os domains do map para o GET
    - Método getById: retornar o domain do map de acordo com o Id informado para o GET
- Domain: Category
    - id: Int -> language_id
    - name: String -> name
    - lastUpdate: LocalDatetime -> last_update

ms-customer (Giardi)

Micro serviço para gerenciar as de clientes e seus endereços.
Tabelas: customer e address
Carregar dados: city, country, store

Especificação Address

- Controller: AddressController com path /address
    - POST: nome operação create e receber como parâmetro o domain
    - PUT: nome operação update e receber como parâmetro o domain
    - DELETE path /{id}: nome operação update e receber como parâmetro o Id do domain
    - GET: nome operação findAll sem parâmetros e retornar uma lista do domain
    - GET path /{id}: nome operação findById parâmetro Id domain language e retornar o domain correspondente ao Id informado
- Services: AddressService
    - Método save: receber um domain do POST e incluir no Map em memória
        - Regras
            - se o Id e/ou Nome existirem retornar um erro 400
            - Id deve ser maior que zero
            - Endereço deve conter no mínimo 4 caracteres e somente letras, não pode conter números e caracteres especiais e não pode ser vazio
            - Segundo endereço se informado deve conter no mínimo 4 caracteres e somente letras, não pode conter números e caracteres especiais e não pode ser vazio

                - O Id da cidade informado deve existir no Map carregado de cidade
        - Método update: receber um domain do UPDATE e atualizar no Map em memória
            - Regras
                - Nome existir em um Id diferente retornar um erro 400
                - Nome deve conter no mínimo 5 caracteres e somente letras, não pode conter números e caracteres especiai
        - Método delete: remover do Map em memória o id correspondente a partir do DELETE
        - Método getAll: retornar todos os domains do map para o GET
        - Método getById: retornar o domain do map de acordo com o Id informado para o GET
- Domain: Address
    - id: Int -> address_id
    - address: String
    - address2: String (Opcional)
    - district: String
    - cityId: Int
    - postalCode: Int -> postal_code
    - phone: String
    - lastUpdate: LocalDatetime -> last_update
    - Location (não mapear)

Especificação Customer

- Controller: CustomerController com path /customer
    - POST: nome operação create e receber como parâmetro o domain
    - PUT: nome operação update e receber como parâmetro o domain
    - DELETE path /{id}: nome operação update e receber como parâmetro o Id do domain
    - GET: nome operação findAll sem parâmetros e retornar uma lista do domain
    - GET path /{id}: nome operação findById parâmetro Id domain language e retornar o domain correspondente ao Id informado
- Services: CustomerService
    - Método save: receber um domain do POST e incluir no Map em memória
        - Regras
            - se o Id e/ou Nome existirem retornar um erro 400
            - Id deve ser maior que zero
            - Primeiro Nome deve conter no mínimo 3 caracteres e somente letras, não pode conter números e caracteres especiais e não pode ser vazio
            - Segundo Nome deve conter no mínimo 3 caracteres e somente letras, não pode conter números e caracteres especiais e não pode ser vazio
            - O Id da loja (store), deve existir no Map carregado de Store
            - O Id do endereço (address) deve existir no Map carregado de endereço
    - Método update: receber um domain do UPDATE e atualizar no Map em memória
        - Regras
            - Nome existir em um Id diferente retornar um erro 400
            - Nome deve conter no mínimo 5 caracteres e somente letras, não pode conter números e caracteres especiai
    - Método delete: remover do Map em memória o id correspondente a partir do DELETE

        - Método getAll: retornar todos os domains do map para o GET
        - Método getById: retornar o domain do map de acordo com o Id informado para o GET
- Domain: Customer
    - id: Int -> customer_id
    - storeId: Int -> store_id
    - firstName: String -> first_name
    - latName: String -> last_name
    - email: String
    - addressId: Int -> address_id
    - active: Boolean
    - createDate: LocalDatetime -> create_update
    - lastUpdate: LocalDatetime -> last_update

ms-film (Daieny)

Micro serviço para gerenciar as informações de filmes e atores/atrizes.
Tabelas: Ator e filme
Carregar dados: language e category

Especificação Actor

- Controller: ActorController com path /actor
    - POST: nome operação create e receber como parâmetro o domain
    - PUT: nome operação update e receber como parâmetro o domain
    - DELETE path /{id}: nome operação update e receber como parâmetro o Id do domain
    - GET: nome operação findAll sem parâmetros e retornar uma lista do domain
    - GET path /{id}: nome operação findById parâmetro Id domain language e retornar o domain correspondente ao Id informado
- Services: ActorService
    - Método save: receber um domain do POST e incluir no Map em memória
        - Regras
            - se o Id e/ou (firstName + lastName) existirem retornar um erro 400
            - Id deve ser maior que zero
            - firstName e lastName devem conter no mínimo 3 caracteres e somente letras, não pode conter números e caracteres especiais e não pode ser vazio
    - Método update: receber um domain do UPDATE e atualizar no Map em memória
        - Regras
            - (firstName + lastName) existir em um Id diferente retornar um erro 400
            - firstName e lastName devem conter no mínimo 3 caracteres e somente letras, não pode conter números e caracteres especiai
    - Método delete: remover do Map em memória o id correspondente a partir do DELETE
    - Método getAll: retornar todos os domains do map para o GET
    - Método getById: retornar o domain do map de acordo com o Id informado para o GET
- Domain: Actor
    - id: Int -> actor_id
    - firstName: String -> first_name

- lastName: String -> last_name
- lastUpdate: LocalDatetime -> last_update

Especificação Film

- Controller: FilmController com path /film
  - POST: nome operação create e receber como parâmetro o domain
  - PUT: nome operação update e receber como parâmetro o domain
  - DELETE path /{id}: nome operação update e receber como parâmetro o Id do domain
  - GET: nome operação findAll sem parâmetros e retornar uma lista do domain
  - GET path /{id}: nome operação findById parâmetro Id domain language e retornar o domain correspondente ao Id informado
- Services: FilmService
  - Método save: receber um domain do POST e incluir no Map em memória
    - Regras
      - se o Id e/ou Nome existirem retornar um erro 400
      - Id deve ser maior que zero
      - Nome do filme (title) deve conter no mínimo 2 caracteres e somente letras, não pode conter números e caracteres especiais e não pode ser vazio
      - Descrição (description) deve conter no mínimo 10 caracteres e somente letras, não pode conter números e caracteres especiais e não pode ser vazio
      - O Id do idioma (language), deve existir no Map carregado de Language
      - O Id da categoria (category) deve existir no Map carregado de Category
  - Método update: receber um domain do UPDATE e atualizar no Map em memória
    - Regras
      - Nome do filme (title) existir em um Id diferente retornar um erro 400
      - Nome deve conter no mínimo 5 caracteres e somente letras, não pode conter números e caracteres especiais
      - O Id do idioma (language), deve existir no Map carregado de Language
      - O Id da categoria (category) deve existir no Map carregado de Category
  - Método delete: remover do Map em memória o id correspondente a partir do DELETE
  - Método getAll: retornar todos os domains do map para o GET
  - Método getById: retornar o domain do map de acordo com o Id informado para o GET
- Domain: Film
  - id: Int -> film_id
  - title: String
  - description: String
  - releaseYear: Int -> release_year
  - languageId: Int -> language_id
  - categoryId: Int -> category_id
  - lastUpdate: LocalDatetime -> last_update

ms-store (Kate)

Micro serviço para gerenciar as informações de filmes e atores/atrizes.
Tabelas: staff e store
Carregar dados: city, address e country

Especificação Staff

- Controller: StaffController com path /staff
    - POST: nome operação create e receber como parâmetro o domain
    - PUT: nome operação update e receber como parâmetro o domain
    - DELETE path /{id}: nome operação update e receber como parâmetro o Id do domain
    - GET: nome operação findAll sem parâmetros e retornar uma lista do domain
    - GET path /{id}: nome operação findById parâmetro Id domain language e retornar o domain correspondente ao Id informado
- Services: StaffService
    - Método save: receber um domain do POST e incluir no Map em memória
        - Regras
            - se o Id e/ou (firstName + lastName) existirem retornar um erro 400
            - Id deve ser maior que zero
            - firstName e lastName devem conter no mínimo 3 caracteres e somente letras, não pode conter números e caracteres especiais e não pode ser vazio
            - O addressId, deve existir no Map carregado de Address
            - O storeId, deve existir no Map com o Store registrado a partir da API
    - Método update: receber um domain do UPDATE e atualizar no Map em memória
        - Regras
            - (firstName + lastName) existir em um Id diferente retornar um erro 400
            - firstName e lastName devem conter no mínimo 3 caracteres e somente letras, não pode conter números e caracteres especiais
            - O addressId, deve existir no Map carregado de Address
            - O storeId, deve existir no Map com o Store registrado a partir da API
    - Método delete: remover do Map em memória o id correspondente a partir do DELETE
    - Método getAll: retornar todos os domains do map para o GET
    - Método getById: retornar o domain do map de acordo com o Id informado para o GET
- Domain: Staff
    - id: Int -> actor_id
    - firstName: String -> first_name
    - lastName: String -> last_name
    - addressId: Int -> address_id
    - storeId: Int -> store_id
    - lastUpdate: LocalDatetime -> last_update

Especificação Store

- Controller: StoreController com path /store
    - POST: nome operação create e receber como parâmetro o domain
    - PUT: nome operação update e receber como parâmetro o domain
    - DELETE path /{id}: nome operação update e receber como parâmetro o Id do domain
    - GET: nome operação findAll sem parâmetros e retornar uma lista do domain
    - GET path /{id}: nome operação findById parâmetro Id domain language e retornar o domain correspondente ao Id informado
- Services: StoreService
    - Método save: receber um domain do POST e incluir no Map em memória

        - Regras
            - se o Id e/ou Nome existirem retornar um erro 400
            - Id deve ser maior que zero
            - O addressId deve existir no Map carregado de Address
    - Método update: receber um domain do UPDATE e atualizar no Map em memória
        - Regras
            - O addressId deve existir no Map carregado de Address
    - Método delete: remover do Map em memória o id correspondente a partir do DELETE
    - Método getAll: retornar todos os domains do map para o GET
    - Método getById: retornar o domain do map de acordo com o Id informado para o GET
- Domain: Store
    - id: Int -> store_id
    - addressId: Int -> address_id
    - lastUpdate: LocalDatetime -> last_update

Dados

Os dados abaixo devem ser carregados em **memória** de acordo com a necessidade do micro serviço que está sendo desenvolvido utilizando um objeto **Map** usando como **key** o **Id da classe de Domain** e o **Value** a **instância do Domain** em questão. Exemplo:

ms-customer precisa do dados das tabelas de country e city, o map deve ser
Map de Countries, sendo key(campo Id da classe) e value(Instância da classe Country), de acordo com a tabela abaixo
Map de Cities, sendo key(campo Id da classe) e value(Instância da classe City), de acordo com a tabela abaixo

Tabela Country

country_id	country	last_update
1	**Brasil**	**2023-01-01 09:00:00**

Tabela City

city_id	city	country_id	last_update
1	**São Paulo**	**1**	**2023-01-01 09:00:00**
2	**Rio de Janeiro**	**1**	**2023-01-01 10:00:00**
3	**Santa Catarina**	**1**	**2023-01-01 11:00:00**
4	**Minas Gerais**	**1**	**2023-01-01 12:00:00**

Tabela Language

language_id	name	last_update
1	Postuguês	2023-01-01 09:00:00
2	Inglês	2023-01-01 10:00:00
3	Espanhol	2023-01-01 11:00:00

Tabela Category

category_id	name	last_update
1	Ação	2023-01-01 09:00:00
2	Comédia	2023-01-01 10:00:00
3	Ficção	2023-01-01 11:00:00

Tabela Address

address_id	address	address2	district	city_id	postal_code	phone	last_update
1	Av Paulista	null	Cerqueira Cezar	1	12345333	93333-0000	2023-01-01 09:00:00
2	Av Brasil	null	Leblon	2	12345444	94444-0001	2023-01-01 10:00:00
3	Av Brasil	null	Sambaqui	3	12345555	95555-0002	2023-01-01 11:00:00

Tabela Store

store_id	manager_staff_id	address_id	last_update
1	1	1	2023-01-01 09:00:00
2	2	2	2023-01-01 10:00:00
3	3	3	2023-01-01 11:00:00

SUPER IMPORTANTE

Dica: o Map mencionado é o mapOf do kotlin.
Usem ele para simular como se fosse uma tabela em memória!

Não exitem, perguntem!
Vou ajudar em dúvidas, só não vou dar a solução!

Divirtam-se!