SISTEMA DE GERENCIAMENTO DE PATRIMÔNIOS - SIS-PAT

1 - Dados Técnicos

ITEM	TECNOLOGIA	VERSÃO
Linguagem	Java	SE-11 (Compliant)
Framework	Quarkus	1.13.7.Final
Repositório de Bibliotecas	Apache Maven	3.6.3
Autenticação	JWT	1.13.7-Final
Interface Visual API	OpenAPI/Swagger	3.0.3
Acesso JPA	Panache	1.13.7.Final
Banco de Dados	PostgreSQL	13.3.2

---

2 - Arquitetura

A arquitetura adotada foi uma variante da **MVC** (*Model-View-Controller*). Nesse projeto, optou-se por utilizar seis camadas de sistema para melhor modularizar e delegar as funcionalidades presentes. Do Endpoint ao Banco, estão caracterizadas:

#	CAMADA	CAMADA-MACRO	SUFIXO DE CLASSE	DESCRIÇÃO
1	MODELO	MODEL	-	Representantes diretas e fiéis das tabelas presentes no banco de dados. São os objetos centrais e manipulados diretamente pela camada de controle
2	ACESSO	CONTROLLER	*DAO*	Classes responsáveis pelo meio de campo do acesso ao banco de dados. Concentram todas as queries relevantes ao projeto. Por causa da implementação do *framework* `Panache`, sua implementação ficou bem simplificada, sendo necessária implementação apenas de queries específicas
3	SERVIÇO	CONTROLLER	*Service*	Camada que gerencia e trafega objetos do tipo*Model* como eles são, aplicando toda e qualquer regra de negócio necessária a eles. Se subdivide em dois tipos essenciais - Operacional e Validação. As classes de validação são as únicas que lidam com parametrização do tipo *VO* diretamente
4	PORTAL/GATEWAY	VIEW	*Gateway*	Grupo de classes encarregada de gerenciar recebimentos de VO's e chamadas controladas às camadas de serviço. Base de comunicação se dá, essencialmente, em mensagens
5	VIEW OBJECT	VIEW	*VO*	Objetos que compõem informações de visualização e tráfico de entrada e saída do sistema
6	REST	VIEW	*Resource*	Camada que expõe os endpoints do sistema. Responsável por regular também o acesso externo e verificação do Token de Autorização

---

3 - Configurações Iniciais - Antes de Rodar o projeto

3.1 - Configuração de Acesso à Banco

Vá ao arquivo `sis-pat/src/main/resources/application.properties` e configure as seguintes linhas:

- `quarkus.datasource.username=[USUARIO_DO_BANCO]` com o usuário do Banco de Dados (sem os colchetes)
- `quarkus.datasource.password=[SENHA_DO_BANCO]` com a senha do Banco de Dados (sem os colchetes)

3.2 - Criação do ambiente dentro do Banco de Dados

Logo na raiz do projeto, existe um arquivo SQL com a inicialização do banco de dados. Só será necessário rodar a criação do *Database*, visto que as tabelas estão mapeadas em detalhe dentro dos seus respectivos *Models*.

Ao iniciar o projeto, o framework *Quarkus* se encarregará de estruturar as tabelas dentro do banco, seguindo as configurações das entidades mapeadas à risca.

3.3 - Configuração de tempo de duração do JWT

Caso deseje aumentar ou diminuir o tempo de duração do Token de autorização, configure as seguintes linhas do arquivo `sis-pat/src/main/resources/application.properties`:

- `smallrye.jwt.time-to-live=[VALOR_EM_SEGUNDOS]` com um número positivo que representará os segundos de duração do token

3.4 - Informações adicionais

Ao iniciar, o sistema cria as tabelas necessárias para trabalho, além de popular as mesmas com dados de teste da API.

Essa criação é sensível à presença de dados nas tabelas que o projeto acessará - pulando a inserção caso existam dados previamente cadastrados nas tabelas.

Caso deseje desligar definitivamente a auto-população de dados para testes, defina a variável `isRodarStartUp` como `false`, dentro da classe `sis-pat/src/main/java/br/com/navita/config/Startup.java`.

---

4 - Iniciando a Aplicação

Ao descompactar a aplicação, rode `mvn quarkus:dev` dentro da raiz do projeto. Depois disso, a aplicação estará disponível na porta 8080 (localhost). Caso deseje uma experiência mais agradável ao navegar pelos endpoints disponíveis, acesse o seguinte endereço no seu navegador: **localhost:8080/q/api-ui**.

Esse link permitirá o acesso à interface Swagger, onde os serviços estarão mapeados de forma mais amigável.

OBS.: Necessário **Aapache Maven 3.6.3** mapeado no seu sistema para executar este comando.

---

5 - Mapeamento dos Serviços

Todos os serviços aqui descritos estão disponíveis, em ordem, na interface *Swagger* da aplicação.

O título dos serviços informam que permissão é necessária para acessá-los.

Ex.: *5.X.Y* - [NOME_SERVIÇO] - ABERTO / USUÁRIO / ADMIN

A descrição de cada serviço inclui um exemplo de requisição e o esqueleto da resposta.

5.1 - USUÁRIO

5.1.1 - Login - ABERTO

Ao invés de gerar um token de autenticação manualmente, utilize este serviço. Ele gerará um token automaticamente para poder navegar pelas funcionalidades do sistema.

REQ.

```
$curl -X 'POST' \ 'http://localhost:8080/publico/login' \ -H 'accept: application/json' \ -H 'Content-Type: application/json' \ -d '{ "email": "admin@email.org.br", "senha": "admin" }'
```

RES.

```
{ "msg": "string", "token": "string", "timestamp": "string" }
```

OBS-1.: O exemplo de requisição possui um login do administrador padrão, que já vem salvo no banco de dados, disponível no momento do *Startup* da aplicação.

OBS-2.: Caso esteja utilizando a interface Swagger, existe um botão **Authorize** no topo-direito da tela, onde pode-se colocar o token gerado por esta operação e garantir acesso de administrador a todos os endpoints do sistema. O acesso durará pelo tempo que o token for válido (vide Item 3.3 para configuração do prazo).

5.1.2 - Cadastro de Usuário - ABERTO

Serviço que cadastra um novo usuário apenas com papel de usuário.

Ao término da operação, será gerado um token de autorização para testar acesso aos outros endpoints, caso deseje.

Observada regra de negócio em que *não pode haver e-mails repetidos dentro da base*

REQ.

```
curl -X 'POST' \ 'http://localhost:8080/publico/novoUsuario' \ -H 'accept: application/json' \ -H 'Content-Type: application/json' \ -d '{ "email": "teste@email.org.br", "nome": "Teste", "senha": "12345" }'
```

RES.

```
{ "msg": "string", "token": "string", "timestamp": "string" }
```

5.1.3 - Consultar todos os Usuários - ADMIN

Consulta todos os usuários registrados no sistema

REQ.

```
curl -X 'GET' \ 'http://localhost:8080/admin/consultar_todos_usuarios' \ -H 'accept: application/json' \ -H 'Authorization: Bearer [TOKEN]'
```

RES.

```
{ "dados": [ { "mensagem": "string", "email": "string", "id": 0, "msg": "string", "nome": "string", "papeis": [ "string" ] } ], "msg": { "msg": "string", "timestampFormat": "string" }, "qntItens": 0 }
```

5.1.4 - Consultar Usuários por E-mail - ADMIN

Recupera uma lista de usuários por um trecho do e-mail

REQ.

```
curl -X 'POST' \ 'http://localhost:8080/admin/consultar_usuarios_por_email' \ -H 'accept: application/json' \ -H 'Authorization: Bearer [TOKEN]' -H 'Content-Type: application/json' \ -d '{ "email": "email" }'
```

RES.

```
{ "dados": [ { "mensagem": "string", "email": "string", "id": 0, "msg": "string", "nome": "string", "papeis": [ "string" ] } ], "msg": { "msg": "string", "timestampFormat": "string" }, "qntItens": 0 }
```

5.1.5 - Consultar Usuários por Nome - ADMIN

Recupera uma lista de usuários por um trecho do nome

REQ.

```
curl -X 'POST' \ 'http://localhost:8080/admin/consultar_usuarios_por_nome' \ -H 'accept: application/json' \ -H 'Authorization: Bearer [TOKEN]' -H 'Content-Type: application/json' \ -d '{ "nome": "admin" }'
```

RES.

```
{ "dados": [ { "mensagem": "string", "email": "string", "id": 0, "msg": "string", "nome": "string", "papeis": [ "string" ] } ], "msg": { "msg": "string", "timestampFormat": "string" }, "qntItens": 0 }
```

5.1.6 - Criar Usuário Administrador - ADMIN

Cria um usuário com papéis de usuário e administrador.

Observada regra de negócio em que *não pode haver e-mails repetidos dentro da base*

REQ.

```
curl -X 'POST' \ 'http://localhost:8080/admin/criar_usuario_admin' \ -H 'accept: application/json' \ -H 'Authorization: Bearer [TOKEN]' -H 'Content-Type: application/json' \ -d '{ "email": "email@admin.org.br", "nome": "Novo Admin", "senha": "12345" }'
```

RES.

```
{ "msg": "string", "token": "string", "timestamp": "string" }
```

5.2 - MARCA DE PATRIMÔNIO

5.2.1 - Consultar todas as Marcas de Patrimônio - USER / ADMIN

Consulta geral das marcas de patrimônio.

REQ.

```
curl -X 'GET' \ 'http://localhost:8080/marca_patrimonio' \ -H 'accept: application/json' \ -H 'Authorization: Bearer [TOKEN]'
```

RES.

```
{ "dados": [ { "id": 0, "nome": "string" } ], "msg": "string", "qntItens": 0 }
```

5.2.2 - Alterar Marca de Patrimônio - USER / ADMIN

Altera os dados (nome) da marca de patrimônio

REQ.

```
curl -X 'POST' \ 'http://localhost:8080/marca_patrimonio/alterar' \ -H 'accept: application/json' \ -H 'Authorization: Bearer [TOKEN]' -H 'Content-Type: application/json' \ -d '{ "id": 5, "novoNome": "Marca Teste" }'
```

RES.

```
{ "msg": "string", "timestampFormat": "string" }
```

5.2.3 - Cadastrar Marca de Patrimônio - USER / ADMIN

Cadastra uma nova marca de patrimônio

Observada regra de negócio que *duas marcas de patrimônio não podem ter o mesmo nome*

REQ.

```
curl -X 'POST' \ 'http://localhost:8080/marca_patrimonio/cadastrar' \ -H 'accept: application/json' \ -H 'Authorization: Bearer [TOKEN]' -H 'Content-Type: application/json' \ -d '{ "nome": "Nova Marca" }'
```

RES.

```
{ "msg": "string", "timestampFormat": "string" }
```

5.2.4 - Consultar Marca de Patrimônio por ID - USER / ADMIN

Recupera uma Marca de Patrimônio por ID

REQ.

```
curl -X 'POST' \ 'http://localhost:8080/marca_patrimonio/consultar/id' \ -H 'accept: application/json' \ -H 'Authorization: Bearer [TOKEN]' -H 'Content-Type: application/json' \ -d '{ "id": 5 }'
```

RES.

```
{ "msg": "string", "timestampFormat": "string" }
```

5.2.5 - Excluir Marca de Patrimônio - ADMIN

Exclui uma Marca de Patrimônio por ID

REQ.

```
curl -X 'POST' \ 'http://localhost:8080/marca_patrimonio/excluir' \ -H 'accept: application/json' \ -H 'Authorization: Bearer [TOKEN]' -H 'Content-Type: application/json' \ -d '{ "idMarca": 5 }'
```

RES.

```
{ "msg": "string", "timestampFormat": "string" }
```

5.3 - PATRIMÔNIO

5.3.1 - Consultar todos os Patrimônio - USER / ADMIN

Consulta geral de patrimônios.

REQ.

```
curl -X 'GET' \ 'http://localhost:8080/patrimonio' \ -H 'accept: application/json' \ -H 'Authorization: Bearer [TOKEN]'
```

RES.

```
{ "dados": [ { "descricao": "string", "idMarca": 0, "msg": "string", "nome": "string", "nomeMarca": "string", "nroTombo": "string" } ], "msg": { "msg": "string", "timestampFormat": "string" }, "qntItens": 0 }
```

5.3.2 - Alterar Patrimônio - USER / ADMIN

Altera os dados de um patrimônio

REQ.

```
curl -X 'POST' \ 'http://localhost:8080/patrimonio/alterar' \ -H 'accept: application/json' \ -H 'Authorization: Bearer [TOKEN]' -H 'Content-Type: application/json' \ -d '{ "descricao": "Desc teste", "idMarca": 4, "nome": "Patrimônio Teste", "nroTombo": "[VALOR_NRO_TOMBO_ADEQUADO]" }'
```

RES.

```
{ "msg": "string", "timestampFormat": "string" }
```

5.3.3 - Cadastrar Patrimônio - USER / ADMIN

Cadastra os dados de um novo patrimônio

Observada regra de negócio em que *o número do tombo do patrimônio deve ser gerado automaticamente, sem interferência do usuário*

REQ.

```
curl -X 'POST' \ 'http://localhost:8080/patrimonio/cadastrar' \ -H 'accept: application/json' \ -H 'Authorization: Bearer [TOKEN]' -H 'Content-Type: application/json' \ -d '{ "descricao": "Novo teste", "idMarca": 11, "nome": "Novo Patrimônio" }'
```

RES.

```
{ "msg": "string", "timestampFormat": "string" }
```

5.3.4 - Consultar Patrimônios por ID da Marca de Patrimônio - USER / ADMIN

Consulta todos os patrimônios pertecentes a uma determinada marca.

REQ.

```
curl -X 'POST' \ 'http://localhost:8080/patrimonio/consultar/idMarca' \ -H 'accept: application/json' \ -H 'Authorization: Bearer [TOKEN]' -H 'Content-Type: application/json' \ -d '{ "idMarca": 11 }'
```

RES.

```
{ "dados": [ { "descricao": "string", "idMarca": 0, "msg": "string", "nome": "string", "nomeMarca": "string", "nroTombo": "string" } ], "msg": { "msg": "string", "timestampFormat": "string" }, "qntItens": 0 }
```

5.3.5 - Consultar Patrimônios por Número do Tombo - USER / ADMIN

Consulta um patrimônio por número do tombo.

REQ.

```
curl -X 'POST' \ 'http://localhost:8080/patrimonio/consultar/nroTombo' \ -H 'accept: application/json' \ -H 'Authorization: Bearer [TOKEN]' -H 'Content-Type: application/json' \ -d '{ "nroTombo": "[NRO_TOMBO_ADEQUADO]" }'
```

RES.

```
{ "descricao": "string", "idMarca": 0, "msg": "string", "nome": "string", "nomeMarca": "string", "nroTombo": "string" }
```

5.3.6 - Excluir Patrimônio - USER / ADMIN

Exclui um patrimônio com base no número do tombo

REQ.

```
curl -X 'POST' \ 'http://localhost:8080/patrimonio/excluir' \ -H 'accept: application/json' \ -H 'Authorization: Bearer [TOKEN]' -H 'Content-Type: application/json' \ -d '{ "nroTombo": "[NRO_TOMBO_ADEQUADO]" }'
```

RES.

```
{ "msg": "string", "timestampFormat": "string" }
```

---

6. Considerações Finais

Embora seja utilizado mais comumente em arquiteturas de micro-serviços para clound, o framework no qual esse projeto foi baseado se provou extremamente versátil, comportando de forma organizada a estrutura de três micro-serviços.

Graças as várias facilidades providas pelo Quarkus, foi possível a criação de uma panacéia de funcionalidades bem heterogênias.