

Projeto: Gestão de Restaurantes

Kelly Cristina Pereira dos Santos – RM361715

Jeferson Herculano de Matos – RM365011

Gustavo Guterres Leite - RM361802

Felipe Matos de Lima - RM362291

Michael Munhoz Pinheiro Barroso - RM363592

1. Introdução

Descrição do problema

O desafio propõe o desenvolvimento de um sistema para que restaurantes possam gerenciar suas operações e os clientes possam consultar informações, deixar avaliações e realizar pedidos em uma aplicação de maneira centralizada, facilitando as ações e diminuindo os custos devido a utilização de sistemas individuais.

Objetivo do projeto

Desenvolver um backend robusto utilizando Spring Boot para gerenciar usuários e atender aos requisitos definidos. As implementações serão feitas por etapas. Na primeira etapa foi entregue funcionalidades com foco no usuário permitindo o cadastro, autenticação, atualização, deleção e consulta. Nesta segunda etapa foram implementadas as funcionalidades referentes aos cadastros de Tipo de Usuário, Restaurante e Itens do Cardápio de um restaurante. Também foram implementados testes unitários.

2. Arquitetura do Sistema

Descrição da Arquitetura

O sistema segue uma arquitetura limpa, promovendo organização, desacoplamento e facilidade de manutenção separando as responsabilidades nas seguintes camadas:

- **Controller:** Recebe as requisições do cliente, expõe os endpoints REST e encaminha para a camada de negócio.
- **Usecase:** Implementa as regras de negócio, validações e orquestra as operações.
- **Gateway:** Abstrai o acesso ao banco de dados, servindo de ponte entre a lógica de negócio e a persistência.
- **Repository:** Responsável pela persistência dos dados, utilizando Spring Data JPA para interagir com o banco.

- **Domain:** Entidades de domínio;
- **Exception:** Tratamento centralizado de exceções;
- **Banco de dados:** PostgreSQL;
- **Docker:** Utilizado para orquestrar a aplicação e o banco de dados.

Diagrama da Arquitetura



3. Descrição dos Endpoints da API Tabela de Endpoints

Endpoint	Método	Descrição
/usuarios	POST	Cadastro de novo usuário
/usuarios/{id}	GET	Consulta de usuário por ID
/usuarios/{id}	DELETE	Exclusão de usuário
/usuarios/{id}	PUT	Atualização de dados do usuário
/usuarios/{id}/senha	PATCH	Troca de senha do usuário
/login	POST	Autenticação de usuário

Exemplos de requisição e resposta

Cenário - Inclusão de usuários preenchendo os dados corretamente:

POST: localhost:8080/usuarios

Body:

```

{
  "cpf": "987.654.321-77",
  "tipoUsuario": 0,
  "nome": "Maria Silva",
  "email": "maria.silva@email.com",
  "login": "mariasilva.br",
  "senha": "senhaSegura123!",
  "endereco": {

```

```
        "logradouro": "Avenida Brasil",
        "numero": "1234",
        "complemento": "apto 101",
        "bairro": "Centro",
        "cidade": "Rio de Janeiro",
        "estado": "RJ",
        "cep": "20000-000"
    }
}
```

Resposta: 201 OK

Cenário - Inclusão de usuários preenchendo login já utilizado:

POST: localhost:8080/usuarios

Body:

```
{
    "cpf": "987.654.321-22",
    "tipoUsuario": 0,
    "nome": "Maria Silva",
    "email": "maria.silva@email.com",
    "login": "mariasilva.br",
    "senha": "senhaSegura123!",
    "endereco": {
        "logradouro": "Avenida Brasil",
        "numero": "1234",
        "complemento": "apto 101",
        "bairro": "Centro",
        "cidade": "Rio de Janeiro",
        "estado": "RJ",
        "cep": "20000-000"
    }
}
```

Resposta: 422 Unprocessable Entity

Body:

```
{
    "code": "usuario.usuarioJaExiste",
    "message": "Usuário já existe"
}
```

Cenário - Consulta de usuário passando um código previamente cadastrado:

GET: localhost:8080/usuarios/1

Resposta: 200 OK

Body:

```
{
  "id": 1,
  "cpf": "987.654.321-22",
  "tipoUsuario": 0,
  "nome": "Maria Silva",
  "email": "maria.silva@email.com",
  "login": "mariasilva.br",
  "senha": "senhaSegura123!",
  "dataUltimaAlteracao": "2025-06-01T20:57:26.549756",
  "endereco": {
    "id": 1,
    "logradouro": "Avenida Brasil",
    "numero": "1234",
    "complemento": "apto 101",
    "bairro": "Centro",
    "cidade": "Rio de Janeiro",
    "estado": "RJ",
    "cep": "20000-000"
  }
}
```

Cenário - Consulta de usuário passando um código inexistente:

GET: localhost:8080/usuarios/100

Resposta: 404 Not Found

Body:

```
{
  "code": "usuario.naoEncontrado",
  "message": "Usuário não encontrado!"
}
```

Cenário - Atualizar usuário passando um código previamente cadastrado:

PUT: localhost:8080/usuarios/1

Body:

```
{
  "cpf": "987.654.321-22",
  "tipoUsuario": 1,
  "nome": "Maria Silva ATUALIZAR",
  "email": "maria.silva.atualizado@email.com",
  "login": "mariasilva.br",
  "senha": "senhaSegura1236!",
  "endereco": {
    "logradouro": "Avenida Brasil ATUALIZADO",

```

```
        "numero": "1234",
        "complemento": "apto 101",
        "bairro": "Centro",
        "cidade": "Rio de Janeiro",
        "estado": "RJ",
        "cep": "20000-000"
    }
}
```

Resposta: 204 No Content

Cenário - Atualizar usuário passando um código de usuário inexistente:

Resposta: 404 Not Found

Body:

```
{
  "code": "usuario.naoEncontrado",
  "message": "Usuário não encontrado!"
}
```

Cenário - Autenticar usuário passando as credenciais corretamente:

POST: localhost:8080/login

```
{
  "login": "mariasilva.br",
  "senha": "senhaSegura1236!"
}
```

Resposta: 200 OK

Body:

Usuário autenticado com sucesso!

Cenário - Autenticar usuário passando as credenciais incorretamente:

POST: localhost:8080/login

```
{
  "login": "mariasilva.br",
  "senha": "senhaSegura12"
}
```

Resposta: 401 Unauthorized

Body:

```
{
  "code": "usuario.senhaIncorreta",

```

```
"message": "A senha informada está incorreta!"
}
```

Cenário - Atualizar senha do usuário passando os dados corretamente:

PATCH: localhost:8080/usuarios/1/senha
Body:
{
 "novaSenha": "senhaSegura12!"
}

Resposta: 204 No Content

Cenário - Deletar usuário passando um id previamente cadastrado:

DELETE: localhost:8080/usuarios/1
Resultado: 204 No Content

Cenário - Deletar usuário passando um id inexistente:

DELETE: localhost:8080/usuarios/2
Resultado: 404 Not Found
Body:
{
 "code": "usuario.naoEncontrado",
 "message": "Usuário não encontrado!"
}

Cenário - Deletar usuário passando um id de usuário utilizado como proprietário de um restaurante:

DELETE: localhost:8080/usuarios/1
Resultado: 400 Bad Request
Body:
{
 "code": "usuario.usuarioUtilizado",
 "message": "Este usuário está vinculado a um ou mais restaurantes!"
}

Endpoint	Método	Descrição
/tipo_usuario	POST	Cadastro de novo tipo de usuário
/tipo_usuario/{id}	GET	Consulta de tipo de usuário por ID

/tipo_usuario /{id}	DELETE	Exclusão de tipo usuário
/tipo_usuario /{id}	PUT	Atualização de dados do tipo de usuário

Exemplos de requisição e resposta

Cenário - Inclusão de tipo de usuários preenchendo os dados corretamente:

POST: localhost:8080/tipo_usuario

Body:

```
{
  "nome": "Proprietário"
}
```

Resposta: 201 OK

Cenário - Inclusão de tipo de usuários deixando campo obrigatório vazio:

POST: localhost:8080/tipo_usuario

Body:

```
{
  "nome": ""
}
```

Resposta: 400 Bad Request

Body

```
{
  "code": "argumentNotValid",
  "message": "nome:não deve estar em branco;"
}
```

Cenário - Inclusão de tipo de usuários preenchendo um nome já utilizado:

POST: localhost:8080/tipo_usuario

Body:

```
{
  "nome": "Proprietário"
}
```

Resposta: 400 Bad Request

Body:

```
{
  "code": "tipoUsuario.tipoUsuarioMesmoNomeExistente",
  "message": "Já existe um tipo de usuário com este nome!"
}
```

Cenário - Consulta de tipo de usuário passando um código previamente cadastrado:

GET: localhost:8080/tipo_usuario/1

Resposta: 200 OK

Body:

```
{
  "id": 1,
  "nome": "Proprietário"
}
```

Cenário - Consulta de tipo de usuário passando um código não existente:

GET: localhost:8080/tipo_usuario/100

Resposta: 404 Not Found

Body:

```
{
  "code": "tipoUsuario.tipoUsuarioNaoEncontrado",
  "message": "Tipo de Usuário não encontrado!"
}
```

Cenário - Atualizar um tipo de usuário passando um código previamente cadastrado e com dados corretos:

PUT: localhost:8080/tipo_usuario/1

Body:

```
{
  "nome": "Dono"
}
```

Resposta: 204 No Content

Cenário - Atualizar um tipo de usuário passando um código inexistente:

PUT: localhost:8080/tipo_usuario/100

Resposta: 404 Not Found

Body:

```
{
  "code": "tipoUsuario.tipoUsuarioNaoEncontrado",

```



```
"message": "Tipo de Usuário não encontrado!"
}
```

Cenário - Deletar um tipo de usuário utilizado no cadastro de um usuário:

DELETE: localhost:8080/tipo_usuario/1
Resultado: 400 Bad Request
Body:

```
{
  "code": "tipoUsuario.tipoUsuarioUtilizado",
  "message": "Este tipo de usuário está vinculado a um ou mais usuários!"
}
```

Cenário - Deletar um tipo de usuário passando um id previamente cadastrado:

DELETE: localhost:8080/tipo_usuario/1
Resultado: 204 No Content

Cenário - Deletar um tipo de usuário passando um id inexistente:

DELETE: localhost:8080/tipo_usuario/200
Resultado: 404 Not Found
Body:

```
{
  "code": "tipoUsuario.tipoUsuarioNaoEncontrado",
  "message": "Tipo de Usuário não encontrado!"
}
```

Endpoint	Método	Descrição
/restaurante	POST	Cadastro de novo restaurante
/restaurante/{id}	GET	Consulta de um restaurante por ID
/restaurante/{id}	DELETE	Exclusão de restaurante
/restaurante/{id}	PUT	Atualização de dados do restaurante

Exemplos de requisição e resposta

Cenário - Inclusão de restaurante preenchendo os dados corretamente:

POST: localhost:8080/restaurante
Body:

```
{
  "nome": "MC Donalds",
  "tipoCozinha": "Fast food",
  "horarioFuncionamento": "Seg à sábado, das 09h às 22h",
  "usuario": {
    "id": 1
  },
  "endereco":{
    "logradouro": "Avenida Ana Costa",
    "numero": "1234",
    "complemento": "",
    "bairro": "Gonzaga",
    "cidade": "Santos",
    "estado": "SP",
    "cep": "11086-100"
  }
}
```

Resposta: 201 OK

Cenário - Inclusão de restaurante deixando campos obrigatórios vazios:

POST: localhost:8080/restaurante

Body:

```
{
  "nome": "",
  "tipoCozinha": "",
  "horarioFuncionamento": "",
  "usuario": {
    "id": 1
  },
  "endereco":{
    "logradouro": "Avenida Ana Costa",
    "numero": "1234",
    "complemento": "",
    "bairro": "Gonzaga",
    "cidade": "Santos",
    "estado": "SP",
    "cep": "11086-100"
  }
}
```

Resposta: 400 Bad Request

Body

```
{
```

```
    "code": "argumentNotValid",
    "message": "tipoCozinha:não deve estar em branco;nome:não deve estar em branco;horarioFuncionamento:não deve estar em branco;"
}
```

Cenário - Inclusão de restaurante preenchendo um nome já utilizado:

POST: localhost:8080/restaurante

Body:

```
{
  "nome": "MC Donalds",
  "tipoCozinha": "Fast food",
  "horarioFuncionamento": "Seg à sábado, das 09h às 22h",
  "usuario": {
    "id": 1
  },
  "endereco": {
    "logradouro": "Avenida Ana Costa",
    "numero": "1234",
    "complemento": "",
    "bairro": "Gonzaga",
    "cidade": "Santos",
    "estado": "SP",
    "cep": "11086-100"
  }
}
```

Resposta: 422 Unprocessable Entity

Body:

```
{
  "code": "restaurante.restauranteJaExiste",
  "message": "Restaurante já existe"
}
```

Cenário - Inclusão de restaurante preenchendo um código de usuário inexistente:

POST: localhost:8080/restaurante

Body:

```
{
  "nome": "MC Donalds",
  "tipoCozinha": "Fast food",
  "horarioFuncionamento": "Seg à sábado, das 09h às 22h",
  "usuario": {
    "id": 100
  },
}
```

```
    "endereco":{
      "logradouro": "Avenida Ana Costa",
      "numero": "1234",
      "complemento": "",
      "bairro": "Gonzaga",
      "cidade": "Santos",
      "estado": "SP",
      "cep": "11086-100"
    }
  }
}
```

Resposta: 404 Not Found

Body

```
{
  "code": "usuario.naoEncontrado",
  "message": "Usuário não encontrado!"
}
```

Cenário - Consulta de restaurante passando um código previamente cadastrado:

GET: localhost:8080/restaurante/1

Resposta: 200 OK

Body:

```
{
  "id": 1,
  "nome": "MC Donalds",
  "tipoCozinha": "Fast food",
  "horarioFuncionamento": "Seg à sábado, das 09h às 22h",
  "usuario": {
    "id": 1,
    "nome": "Maria Silva"
  },
  "endereco": {
    "id": 2,
    "logradouro": "Avenida Ana Costa",
    "numero": "1234",
    "complemento": "",
    "bairro": "Gonzaga",
    "cidade": "Santos",
    "estado": "SP",
    "cep": "11086-100"
  }
}
```

Cenário - Consulta de restaurante passando um código não existente:

GET: localhost:8080/restaurante/100

Resposta: 404 Not Found

Body:

```
{
  "code": "restaurante.restauranteNaoEncontrado",
  "message": "Restaurante não encontrado!"
}
```

Cenário - Atualizar um restaurante passando um código previamente cadastrado e com dados corretos:

PUT: localhost:8080/restaurante/1

Body:

```
{
  "nome": "MC Donalds Atualizado",
  "tipoCozinha": "Fast food",
  "horarioFuncionamento": "Seg à sábado, das 09h às 22h",
  "usuario": {
    "id": 1
  },
  "endereco": {
    "logradouro": "Avenida Paulista",
    "numero": "222",
    "complemento": "",
    "bairro": "Gonzaga",
    "cidade": "Santos",
    "estado": "SP",
    "cep": "11086-100"
  }
}
```

Resposta: 204 No Content

Cenário - Atualizar um restaurante passando um código inexistente:

PUT: localhost:8080/restaurante/100

Resposta: 404 Not Found

Body:

```
{
  "code": "restaurante.restauranteNaoEncontrado",
  "message": "Restaurante não encontrado!"
}
```

Cenário - Deletar um restaurante passando um id previamente cadastrado:

DELETE: localhost:8080/restaurante/1

Resultado: 204 No Content

Cenário - Deletar um restaurante passando um id inexistente:

DELETE: localhost:8080/restaurante/100

Resultado: 404 Not Found

Body:

```
{
  "code": "restaurante.restauranteNaoEncontrado",
  "message": "Restaurante não encontrado!"
}
```

Endpoint	Método	Descrição
/item-cardapio	POST	Cadastro de um item do cardápio
/item-cardapio/{id}	GET	Consulta de um item do cardápio por ID
/item-cardapio/restaurante/{id}	GET	Consulta de todos os itens do cardápio por ID do restaurante
/item-cardapio/{id}	DELETE	Exclusão de um item do cardápio
/item-cardapio/{id}	PUT	Atualização de dados do item do cardápio

Exemplos de requisição e resposta

Cenário - Inclusão de itens do cardápio preenchendo os dados corretamente:

POST: localhost:8080/item-cardapio

Body:

```
{
  "nome": "Pizza Quatro queijos",
  "descricao": "Pizza com massa leve, molho artesanal de tomate, quatro queijos selecionados (mussarela, gorgonzola, parmesão e provolone), borda recheada com catupiry e toque final de manjeriço fresco.",
  "preco": 64.90,
  "disponivelApenasNoRestaurante": true,
```

```
    "foto": "https://alimentos10.com.br/wp-content/uploads/2023/06/imeretian-  
    khachapuri-cheese-lemon-side-view-1.jpg",  
    "idRestaurante": 1  
}
```

Resposta: 201 Created

Cenário - Inclusão de item do cardápio deixando campos obrigatórios vazios:

POST: localhost:8080/item-cardapio

Body:

```
{  
  "nome": "",  
  "descricao": "",  
  "preco": -1,  
  "disponivelApenasNoRestaurante": true,  
  "foto": "https://alimentos10.com.br/wp-content/uploads/2023/06/imeretian-khachapuri-  
  cheese-lemon-side-view-1.jpg",  
  "idRestaurante": 1  
}
```

Resposta: 400 Bad Request

Body

```
{  
  "code": "argumentNotValid",  
  "message": "nome:não deve estar em branco;descricao:não deve estar em  
branco;preco:deve ser maior que 0;"  
}
```

Cenário - Inclusão de um item do cardápio preenchendo um código de restaurante inexistente:

POST: localhost:8080/item-cardapio

Body:

```
{  
  "nome": "Pizza Quatro queijos",  
  "descricao": "Pizza com massa leve, molho artesanal de tomate, quatro queijos  
  selecionados (mussarela, gorgonzola, parmesão e provolone), borda recheada com  
  catupiry e toque final de manjeriço fresco.",  
  "preco": 64.90,  
  "disponivelApenasNoRestaurante": true,  
  "foto": "https://alimentos10.com.br/wp-content/uploads/2023/06/imeretian-  
  khachapuri-cheese-lemon-side-view-1.jpg",  
  "idRestaurante": 200  
}
```

```
}
```

Resposta: 404 Not Found

Body

```
{
  "code": "restaurante.restauranteNaoEncontrado",
  "message": "Restaurante não encontrado!"
}
```

Cenário - Consulta de item do cardápio passando um código previamente cadastrado:

GET: localhost:8080/item-cardapio/1

Resposta: 200 OK

Body:

```
{
  "id": 1,
  "nome": "Pizza Quatro queijos",
  "descricao": "Pizza com massa leve, molho artesanal de tomate, quatro queijos selecionados (mussarela, gorgonzola, parmesão e provolone), borda recheada com catupiry e toque final de manjeriço fresco.",
  "preco": 64.90,
  "disponivelApenasNoRestaurante": true,
  "foto": "https://alimentos10.com.br/wp-content/uploads/2023/06/imeretian-khachapuri-cheese-lemon-side-view-1.jpg",
  "restaurante": {
    "id": 1,
    "nome": "MC Donalds",
    "tipoCozinha": "Fast food",
    "horarioFuncionamento": "Seg à sábado, das 09h às 22h",
    "usuario": {
      "id": 1,
      "nome": "Maria Silva"
    },
  },
  "endereco": {
    "id": 2,
    "logradouro": "Avenida Ana Costa",
    "numero": "1234",
    "complemento": "",
    "bairro": "Gonzaga",
    "cidade": "Santos",
    "estado": "SP",
    "cep": "11086-100"
  }
}
```


Cenário - Consulta de item do cardápio passando um código não existente:

GET: localhost:8080/item-cardapio/100

Resposta: 404 Not Found

Body:

```
{
  "code": "itemCardapio.naoEncontrado",
  "message": "Item de cardápio não encontrado!"
}
```

Cenário - Atualizar um item do cardápio passando um código previamente cadastrado e com dados corretos:

PUT: localhost:8080/item-cardapio/1

Body:

```
{
  "nome": "Pizza de 4 queijos",
  "descricao": "Descrição atualizada",
  "preco": 54.90,
  "disponivelApenasNoRestaurante": false,
  "foto": "https://alimentos10.com.br/wp-content/uploads/2023/06/imeretian-khachapuri-cheese-lemon-side-view-1.jpg",
  "idRestaurante": 1
}
```

Resposta: 204 No Content

Cenário - Atualizar um item do cardápio passando um código inexistente:

PUT: localhost:8080/item-cardapio/100

Resposta: 404 Not Found

Body:

```
{
  "code": "itemCardapio.naoEncontrado",
  "message": "Item de cardápio não encontrado!"
}
```

Cenário - Deletar um item do cardápio passando um id previamente cadastrado:

DELETE: localhost:8080/item-cardapio/1

Resultado: 204 No Content

Cenário - Deletar um item do cardápio passando um id inexistente:

DELETE: localhost:8080/item-cardapio/100

Resultado: 404 Not Found

Body:

```
{  
  "code": "itemCardapio.naoEncontrado",  
  "message": "Item de cardápio não encontrado!"  
}
```

4. Configuração do Projeto

- Clone o repositório:

```
git clone https://github.com/FIAP-7/TECH-CHALLENGE.git  
cd TECH-CHALLENGE
```

- Execute com o Docker Compose:

```
docker-compose up --build
```

- A aplicação estará disponível em:

<http://localhost:8080>

5. Qualidade do Código

Boas Práticas Utilizadas

- Separação de camadas (Controller, Usecase, Gateway, Domain);
- Uso de interfaces e injeção de dependências (princípios SOLID);
- Validações centralizadas e tratamento de exceções customizadas;
- Convenções do Spring Boot e uso de Lombok;
- Organização dos DTOs e entidades;

6. Collections para Teste

Link para a Collection do Postman: /TECH-CHALLENGE/documentacao/postman-collections/Tech-Challenge.postman_collection.json

Descrição dos Testes Manuais

- Importe a collection no Postman.
- Execute as requisições de cadastro, login, consulta, atualização e deleção de usuário para validar os endpoints.
- Altere os dados conforme necessário para testar diferentes cenários (sugestões supracitadas).

7. Cobertura de testes

gestao-restaurantes

gestao-restaurantes

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
br.com.fiap.postech.gestao.restaurantes.core.usecase.itemCardapio		65%		35%	13	23	16	50	6	16	1	5
br.com.fiap.postech.gestao.restaurantes.infra.repositories		94%		60%	13	53	18	348	2	38	0	4
br.com.fiap.postech.gestao.restaurantes.core.exception		79%		n/a	4	20	15	77	4	20	3	19
br.com.fiap.postech.gestao.restaurantes.core.usecase.restaurante.handler		75%		50%	9	17	7	32	0	8	0	4
br.com.fiap.postech.gestao.restaurantes.core.exception.itemCardapio		66%		n/a	2	6	8	24	2	6	2	6
br.com.fiap.postech.gestao.restaurantes.core.usecase.restaurante		88%		50%	5	16	5	68	3	14	0	4
br.com.fiap.postech.gestao.restaurantes.core.usecase.usuario.handler		88%		68%	7	25	5	50	0	14	0	7
br.com.fiap.postech.gestao.restaurantes.core.entities		97%		97%	2	78	5	197	1	42	0	6
br.com.fiap.postech.gestao.restaurantes.core.gateway		100%		81%	6	50	0	102	0	34	0	4
br.com.fiap.postech.gestao.restaurantes.core.usecase.usuario		100%		100%	0	20	0	92	0	18	0	5
br.com.fiap.postech.gestao.restaurantes.core.usecase.tipoUsuario		100%		100%	0	17	0	37	0	15	0	4
br.com.fiap.postech.gestao.restaurantes.core.usecase.tipoUsuario.handler		100%		100%	0	9	0	20	0	6	0	3
br.com.fiap.postech.gestao.restaurantes.core.usecase.validarLogin.handler		100%		100%	0	7	0	15	0	6	0	3
br.com.fiap.postech.gestao.restaurantes.core.usecase.validarLogin		100%		n/a	0	3	0	9	0	3	0	1
Total	291 of 3.670	92%	47 of 208	77%	61	344	79	1.121	18	240	6	75

Para rodar os testes utilize o comando `mvn clean verify`, isso irá gerar um arquivo com os dados da cobertura no diretório `\target\site\jacoco\index.html`

8. Repositório do Código

URL do Repositório: <https://github.com/FIAP-7/TECH-CHALLENGE>

9. Vídeo do projeto

O vídeo está disponível no diretório:

<https://drive.google.com/drive/folders/1f5OaAjPCdZP8mzyeGjK7oGIR01uXnIsX>