Desafio #5.1
Sistema de Controle de Bicicletário
Planejamento de Testes

Endpoint: POST /ciclista ou PUT ciclista/{idCiclista}

- Regras de inserção dos dados:
    - Nome: String de 4 a 60 caracteres.
    - CPF: cpf válido (somente dígitos)
    - Data de nascimento: formato DD/MM/AAAA onde o cliente deve ter ao menos 18 anos na data corrente.
    - Passaporte: Se o cliente for estrangeiro, deve apresentar um passaporte
        - Número: String de um número de passaporte válido.
        - Validade: No mesmo formato da data de nascimento, mas a data de vencimento deve ser pelo menos um dia após a data corrente.
        - País: String com o código do país onde o ciclista reside
    - Nacionalidade: String com a nacionalidade do país onde o ciclista nasceu. Se não for brasileiro, o ciclista deve ter um passaporte registrado.
    - Email: Email válido no formato aaaa@bbbb.cccc
    - URL da foto do documento: String de uma url que possua uma foto do ciclista, deve estar no formato “http://url.exemplo” ou “https://url.exemplo”
    - Senha: Uma string.
    - Meio de pagamento (Cartão de crédito):
        - Nome do titular: Uma string de 2 a 60 caracteres
        - Número do cartão: Uma string com o número do cartão válido
        - Validade: No mesmo formato da data de nascimento, mas a data de vencimento deve ser pelo menos um mês após a data corrente.
        - CVV: String com exatamente três dígitos
- Outras regras:
    - Não pode haver dois ciclistas com o mesmo cpf, e-mail e número de passaporte
    - Os dados são enviados via json e retornados no mesmo formato.
    - O endpoint retorna 201 em caso de sucesso com os dados do ciclista.
    - Retorna 404 para uma requisição mal formada, mostrando o código e mensagem.
    - Retorna 422 para dados inválidos, retornando o código e mensagem.
- Casos de Teste válidos:

CT	Método	Entrada JSON	Código HTTP	Saída JSON
1	POST	JSON CT1	201	JSON CT1
2	POST	JSON CT2	201	JSON CT2
3	PUT ciclista/1	JSON CT1	201	JSON CT1

- Casos de Teste inválidos:
  - Ainda considerando o método POST e o formato do JSON apresentado acima, a separação em classes inválidas, com exemplos de entradas, é dada nas seguintes tabelas:
  - A saída é um json no formato:

```
[
  {
    "codigo": 422,
    "mensagem": "Dado inválido: (Dado da coluna inválido)"
  }
]
```

CT	Entrada (parte de um JSON)	Inválido	Código HTTP
1	“nome”: (5 espaços) ou “Ze” ou (vazio)	Tamanho < 4	422
2	“nome”: exemplo	Tamanho > 60	422
3	“cpf”: 12345678900	CPF inválido	422
4	“cpf”: 123456789000	Tamanho > 11	422
5	“nascimento”: “19940809”	Formato != DD/MM/AAAA	422
6	“nascimento”: “22/09/2023”	Menor que 18 anos	422
7	“passaporte.numero”: “a” ou “ “ ou	Número de passaporte	422

	“1234450411a11b”	inválido	
8	“passaporte.pais”: “ “ ou “A” ou “ABCDEFG” ou “AAA” ou “BB”	Código de país inválido	422
9	“nacionalidade”: “ “ ou “barsileiro”	Nacionalidade inválida	422
10	“email”: “ “ ou “jose” ou “jose.com”	Email inválido	422
11	“urlFotoDocumento”: “ “ ou “aaaa” ou “http://”	Foto inválida	422

No método PUT, temos as mesmas considerações, mas também deve entrar em conta o parâmetro id, que, dependendo da entrada, pode gerar dois tipos de código:

```
  {
    "codigo": 422,
    "mensagem": "Ciclista não encontrado"
  }
]
[
  {
    "codigo": 404,
    "mensagem": "Requisição mal formada"
  }
]
```

CT	Entrada (parte de um JSON)	Código HTTP
1	PUT ciclista	404
2	PUT ciclista/a	404
3	PUT ciclista/494829428024	422

Endpoint: GET /ciclista/{idCiclista}

Recebe o id do ciclista como parâmetro http, como “ciclista/1”, e retorna os dados do ciclista no formato:

```
{
  "id": 0,
  "status": "string",
  "nome": "string",
  "nascimento": "2023-09-21",
  "cpf": "07353072400",
  "passaporte": {
    "numero": "string",
    "validade": "2023-09-21",
    "pais": "SL"
  },
  "nacionalidade": "string",
  "email": "user@example.com",
  "urlFotoDocumento": "string"
}
```

- Regras para funcionamento
  - O id do ciclista deve ser inteiro e deve existir no banco de dados
  - O parâmetro id deve existir na requisição
- Padrão de requisição inválida:

```
[
  {
    "codigo": 422,
    "mensagem": "Ciclista não encontrado”
  }
]
[
  {
    "codigo": 404,
    "mensagem": “Requisição mal formada”
  }
]
```

- Casos de teste válidos

CT	Requisição	Código HTTP	Saída JSON
1	GET ciclista/1	200	JSON CT1

- Casos de teste inválidos

CT	Requisição	Código HTTP
1	GET ciclista	404
2	GET ciclista/a	404
3	GET ciclista/494829428024	422

Endpoint: POST /ciclista/{idCiclista}/ativar

Casos de Teste válidos

CT	Entrada JSON	Código HTTP	Saída JSON
1	{ "idCiclista": 1, "x-id-requisicao": 1 }	200	JSON CT1

Casos de Teste Inválidos:

CT	Entrada JSON	Código HTTP	Saída JSON
1	{ "idCiclista": , "x-id-requisicao": }	422	{ "codigo": 422, "mensagem": "Dados inválidos" }
2	{ "idCiclista": ”a”, "x-id-requisicao":”b” }	422	{ "codigo": 422, "mensagem": "Dados inválidos" }
3	{ "idCiclista":	404	{ "codigo":

	121212, "x-id-requisicao":1 }		404, "mensagem": "Ciclista não encontrado" }

Endpoint: GET /ciclista/{idCiclista}/permiteAluguel

Casos de teste válidos:

CT	Requisição	Código HTTP	Saída JSON

1	GET /ciclista/1/permiteAluguel	200	{
			true
			}

2	GET /ciclista/2/permiteAluguel	200	{
			false
			}

Casos de teste inválidos:

CT	Requisição	Código HTTP	Saída JSON

1	GET /ciclista/14465/permiteAluguel	404	{
			"codigo":
			404,
			"mensagem":
			"Ciclista não encontrado"
			}

Endpoint: GET /ciclista/{idCiclista}/bicicletaAlugada

Casos de teste válidos:

CT	Requisição	Código HTTP	Saída JSON

1	GET /ciclista/1/bicicletaAlugada	200	<pre>{
			"id": 1,
			"marca": "Caloi",
			"modelo": "Caloi 12A",
			"ano": "2014",
			"numero": 1,
			"status": "EM_USO"
			}</pre>

2	GET /ciclista/2/permiteAluguel	200	<pre>{
			
			}</pre>

Casos de teste inválidos:

CT	Requisição	Código HTTP	Saída JSON

1	GET /ciclista/14465/bicicletaAlugada	404	<pre>{
			"codigo": 404,
			"mensagem": "Ciclista não encontrado"
			}</pre>

Endpoint: GET /ciclista/existeEmail/{email}

Casos de teste válidos:

CT	Requisição	Código HTTP	Saída JSON

1	GET /ciclista/existeEmail/jose@email.c	200	<pre>{
			true</pre>

	om		}

2	GET /ciclista/existeEmail/josefino@email.com	200	{
			false
			}

Casos de teste inválidos:

CT	Requisição	Código HTTP	Saída JSON

1	GET /ciclista/existeEmail/	400	{
			"codigo": 400,
			"mensagem": "Email não enviado como parâmetro"
			}