Teste para Desenvolvedor Junior

Objetivo: Desenvolver uma API utilizando Python com Django para gerenciar agendamentos de pagamentos. A API deve permitir a criação, listagem, consulta e exclusão de agendamentos.

Requisitos Técnicos:

- Utilizar Python com Django.
- Utilize o banco de dados padrão do Django “SQLite”.
- Retornar dados no formato JSON.
- Receber campo de valor em Decimal, mas converter para inteiro antes de salvar no banco de dados.

Endpoints Requeridos:

1 - Criar Agendamento de Pagamento

- **Método:** POST
- **URL:** `/api/agendamentos/`
- **Campos Requeridos no Corpo da Requisição:**
  - `data_pagamento` (data do pagamento no formato YYYY-MM-DD)
  - `permite_recorrencia` (booleano, indica se permite recorrência)
  - `quantidade_recorrencia` (inteiro, número de recorrências)
  - `intervalo_recorrencia` (inteiro, intervalo em dias entre as recorrências)
  - `status_recorrencia` (string, status da recorrência)
  - `agencia` (inteiro, número da agência)
  - `conta` (inteiro, número da conta)
  - `valor_pagamento` (decimal, valor do pagamento, convertido para inteiro no banco de dados)
- **Resposta Esperada:**
  - Código HTTP: 201
  - Corpo da Resposta:

```
{
  "id": <id do agendamento>,
  "data_pagamento": "<data do pagamento>",
  "permite_recorrencia": <booleano>,
  "quantidade_recorrencia": <inteiro>,
  "intervalo_recorrencia": <inteiro>,
  "status_recorrencia": "<status>",
  "agencia": <inteiro>,
```

```
  "conta": <inteiro>,
  "valor_pagamento": <valor em inteiro>
}
```

2 - Listar Todos os Agendamentos de Pagamento

- **Método:** GET
- **URL:** `/api/agendamentos/`
- **Resposta Esperada:**
    - Código HTTP: 200
    - Corpo da Resposta:

```
[
  {
    "id": <id do agendamento>,
    "data_pagamento": "<data do pagamento>",
    "permite_recorrencia": <booleano>,
    "quantidade_recorrencia": <inteiro>,
    "intervalo_recorrencia": <inteiro>,
    "status_recorrencia": "<status>",
    "agencia": <inteiro>,
    "conta": <inteiro>,
    "valor_pagamento": <valor em inteiro>
  },
  ...
]
```

3 - Consultar Agendamento de Pagamento

- **Método:** GET
- **URL:** `/api/agendamentos/<id>/`
- **Parâmetros da URL:**
    - `id` (inteiro, identificador do agendamento)
- **Resposta Esperada:**
    - Código HTTP: 200
    - Corpo da Resposta:

```
{
  "id": <id do agendamento>,
```

```
  "data_pagamento": "<data do pagamento>",
  "permite_recorrencia": <booleano>,
  "quantidade_recorrencia": <inteiro>,
  "intervalo_recorrencia": <inteiro>,
  "status_recorrencia": "<status>",
  "agencia": <inteiro>,
  "conta": <inteiro>,
  "valor_pagamento": <valor em inteiro>
}
```

4 - Deletar Agendamento de Pagamento

- **Método:** DELETE
- **URL:** `/api/agendamentos/<id>/`
- **Parâmetros da URL:**
  - `id` (inteiro, identificador do agendamento)
- **Resposta Esperada:**
  - Código HTTP: 204 (sem conteúdo)

Observações:

- Não é necessário implementar todos os endpoints se o tempo for curto.
- Faça o seu melhor para garantir que os endpoints implementados funcionem corretamente e que a API retorne os dados no formato JSON conforme especificado.

5 - Passos para Implementação:

1. **Configuração do Projeto Django:**
   - Crie um novo projeto Django e um novo app para os agendamentos.
2. **Modelagem:**
   - Crie um modelo para os agendamentos de pagamento com os campos especificados.
   - Converta o valor do pagamento de decimal para inteiro antes de salvar no banco de dados.
3. **Serializers:**
   - Implementar o response em JSON.
4. **Views e URLs:**

    - Implemente as views para os endpoints de criação, listagem, consulta e exclusão de agendamentos.
    - Configure as URLs correspondentes.
5. **Testes:**
    - Teste os endpoints utilizando qualquer ferramenta como Postman, Insomnia ou cURL para garantir que a API está funcionando conforme esperado.

Instruções Finais:

- Faça um commit do código em um repositório Git e envie o link para revisão.