Tutorial: Configurando Supabase Storage (S3) com Django e Railway

Este tutorial irá guiá-lo na reconfiguração do seu projeto Django para utilizar o Supabase Storage através da sua compatibilidade com a API S3, usando a biblioteca `django-storages`. Esta é a abordagem mais robusta e recomendada, dado que a integração direta via `django-storage-supabase` está desatualizada.

1. Configuração do Supabase Storage para Compatibilidade S3

Para usar o Supabase Storage como um serviço S3, você precisará de credenciais S3 (Access Key ID e Secret Access Key) e um endpoint S3.

1.1. Gerar Credenciais S3 no Supabase

1. Acesse o painel do seu projeto Supabase.
2. Navegue até **Project Settings** (ícone de engrenagem no canto inferior esquerdo).
3. No menu lateral, selecione **Storage**.
4. Na seção **S3 Credentials**, clique em **Generate New Key**.
5. Será gerado um **Access Key ID** e um **Secret Access Key**. **Copie-os imediatamente**, pois o Secret Access Key não será exibido novamente.

1.2. Obter Endpoint e Região do Supabase S3

O endpoint S3 do Supabase é derivado do URL da sua API. A região geralmente é a mesma da sua região de projeto Supabase.

- **Endpoint:** O endpoint S3 geralmente segue o formato `https://<project-ref>.supabase.co/storage/v1/s3`.

    - Você pode encontrar o `project-ref` no URL do seu painel Supabase (ex: `https://supabase.com/dashboard/project/<project-ref>/...`).
- **Região:** A região padrão do Supabase é `us-east-1` para compatibilidade S3, a menos que especificado de outra forma na documentação do Supabase ou nas configurações do seu projeto.

Exemplo:

- **Access Key ID:** `supabase_s3_access_key_id_gerado`
- **Secret Access Key:** `supabase_s3_secret_access_key_gerado`
- **Endpoint URL:** `https://<your-project-ref>.supabase.co/storage/v1/s3`
- **Região:** `us-east-1` (ou a região do seu projeto Supabase, se diferente)

1.3. Configurar Políticas de Acesso (Policies) no Supabase

As políticas de RLS (Row Level Security) ainda se aplicam ao acesso S3. Certifique-se de que seu bucket tenha as políticas apropriadas para `SELECT` (leitura) e `INSERT` (escrita/upload). O exemplo abaixo é para um bucket público onde qualquer um pode ler e usuários autenticados podem escrever.

1. No painel Supabase, vá para **Storage** e selecione seu bucket (ex: `canesgril-media`).
2. Vá para a aba **Policies**.

Exemplo de Políticas:

- **Leitura Pública:** `sql CREATE POLICY "Public access for objects" ON storage.objects FOR SELECT USING (bucket_id = 'canesgril-media');`
- **Upload Autenticado:** `sql CREATE POLICY "Authenticated upload for objects" ON storage.objects FOR INSERT WITH CHECK (bucket_id = 'canesgril-media' AND auth.role() = 'authenticated');`

2. Configuração do Django com `django-storages` (S3)

Agora, vamos configurar seu projeto Django para usar o `django-storages` com as credenciais S3 do Supabase.

2.1. Atualizar `requirements.txt`

Remova `django-storage-supabase` e `supabase` (se ainda estiverem lá) e adicione `django-storages` e `boto3` (o cliente AWS SDK para Python, que `django-storages` usa para S3).

```
 # requirements.txt
Django==5.2.4 # Sua versão do Django
Pillow # Se você usa para processamento de imagem
django-storages
boto3
# ... outras dependências ...
```

2.2. Configurar `settings.py`

Remova as configurações antigas do `django-storage-supabase` e adicione as configurações para `django-storages`.

```python
# settings.py

import os
from pathlib import Path
from dotenv import load_dotenv

load_dotenv()

BASE_DIR = Path(__file__).resolve().parent.parent

# ... (suas outras configurações como SECRET_KEY, DEBUG, ALLOWED_HOSTS, etc.)
...

INSTALLED_APPS = [
    # ... suas outras apps ...
    'django.contrib.staticfiles',
    'storages', # Mantenha ou adicione esta linha
]

# ... (seus MIDDLEWARE, ROOT_URLCONF, TEMPLATES, WSGI_APPLICATION, DATABASES,
etc.) ...

# Configurações para django-storages com Supabase S3
AWS_ACCESS_KEY_ID = os.environ.get('SUPABASE_S3_ACCESS_KEY_ID')
AWS_SECRET_ACCESS_KEY = os.environ.get('SUPABASE_S3_SECRET_ACCESS_KEY')
AWS_STORAGE_BUCKET_NAME = os.environ.get('SUPABASE_BUCKET_NAME') # O nome do
seu bucket no Supabase
AWS_S3_ENDPOINT_URL = os.environ.get('SUPABASE_S3_ENDPOINT_URL') # Ex:
https://<project-ref>.supabase.co/storage/v1/s3
AWS_S3_REGION_NAME = os.environ.get('SUPABASE_S3_REGION_NAME', 'us-east-1') #
Região do seu projeto Supabase
AWS_S3_SIGNATURE_VERSION = 's3v4' # Necessário para compatibilidade S3
AWS_S3_FILE_OVERWRITE = False # Evita sobrescrever arquivos com o mesmo nome
AWS_DEFAULT_ACL = None # Define ACLs para objetos (pode ser 'public-read' se o
bucket for público)

# Configuração do Default File Storage para arquivos de mídia
DEFAULT_FILE_STORAGE = 'storages.backends.s3boto3.S3Boto3Storage'

# URL base para arquivos de mídia (opcional, mas recomendado para consistência)
# Esta URL será usada pelo Django para construir URLs para seus arquivos de
mídia.
# Certifique-se de que o caminho 'public' está correto para o seu bucket.
MEDIA_URL = f"{AWS_S3_ENDPOINT_URL}/{AWS_STORAGE_BUCKET_NAME}/public/" # Ajuste
'public/' se necessário

# Configurações para arquivos estáticos (se você também quiser servi-los do S3)
# STATICFILES_STORAGE = 'storages.backends.s3boto3.S3Boto3Storage'
# STATIC_URL = f"{AWS_S3_ENDPOINT_URL}/{AWS_STORAGE_BUCKET_NAME}/static/"

# ... (suas outras configurações como DEFAULT_AUTO_FIELD, CSRF_COOKIE_SECURE,
etc.) ...

# Configuração de Logging (mantenha a que você já adicionou para depuração)
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'formatters': {
        'verbose': {
            'format': '{levelname} {asctime} {module} {process:d} {thread:d}
{message}',
```

```
            'style': '{',
        },
        'simple': {
            'format': '{levelname} {message}',
            'style': '{',
        },
    },
    'handlers': {
        'console': {
            'level': 'DEBUG',
            'class': 'logging.StreamHandler',
            'formatter': 'verbose',
        },
    },
    'loggers': {
        'django': {
            'handlers': ['console'],
            'level': 'INFO',
            'propagate': False,
        },
        'django.request': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'storages': { # Adicione este logger para a biblioteca django-storages
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'boto3': { # Adicione este logger para a biblioteca boto3
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'botocore': { # Adicione este logger para a biblioteca botocore
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        's3transfer': { # Adicione este logger para a biblioteca s3transfer
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
        '': { # Logger raiz, captura tudo que não for capturado pelos outros
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
    },
}
```

2.3. Configurar `models.py`

Seu `models.py` não precisa de alterações. Como você já está usando `ImageField` sem um argumento `storage` explícito, ele continuará usando o

`DEFAULT_FILE_STORAGE` que agora aponta para `S3Boto3Storage`.

```
# myapp/models.py

from django.db import models
import uuid

def get_file_path(_instance, filename):
    ext = filename.split('.')[-1]
    filename = f'{uuid.uuid4()}.{ext}'
    return filename

class MyModel(models.Model):
    name = models.CharField(max_length=100)
    foto_prato = models.ImageField(upload_to=get_file_path, blank=True) # Usará
o S3Boto3Storage

    def __str__(self):
        return self.name

    def foto_prato_url(self):
        if self.foto_prato and hasattr(self.foto_prato, 'url'):
            return self.foto_prato.url
        return ""
```

3. Configuração do Railway

Para que seu projeto Django no Railway possa interagir corretamente com o Supabase Storage via S3, você precisa garantir que as novas variáveis de ambiente necessárias estejam configuradas.

3.1. Adicionar Variáveis de Ambiente no Railway

No painel do seu projeto Railway:

1. Vá para as configurações do seu serviço Django.
2. Navegue até a seção **Variables**.
3. Adicione as seguintes variáveis de ambiente, usando os valores que você obteve do Supabase:

Name	Value
`SUPABASE_S3_ACCESS_KEY_ID`	`supabase_s3_access_key_id_gerado`
`SUPABASE_S3_SECRET_ACCESS_KEY`	`supabase_s3_secret_access_key_gerado`
`SUPABASE_BUCKET_NAME`	`canesgril-media`
`SUPABASE_S3_ENDPOINT_URL`	`https://<your-project-ref>.supabase.co/storage/v1/s3`
`SUPABASE_S3_REGION_NAME`	`us-east-1`

3.2. Implantação e Teste

Após configurar as variáveis de ambiente no Railway e garantir que seu código Django esteja atualizado com as configurações do `django-storages` para S3:

1. **Implante novamente** seu projeto Django no Railway.
2. **Teste o upload de imagens** através do seu aplicativo Django (por exemplo, via Django Admin ou um formulário personalizado).
3. **Verifique no painel do Supabase Storage** se as imagens estão sendo carregadas corretamente no seu bucket.
4. **Acesse as URLs das imagens** no seu navegador para confirmar que elas estão sendo servidas pelo Supabase.
5. **Monitore os logs do Railway** para qualquer `Traceback` ou mensagem de erro detalhada. Com `django-storages` e `boto3`, você deve obter logs mais claros em caso de problemas.

Este guia deve fornecer um caminho claro para integrar o Supabase Storage via S3 com seu projeto Django no Railway. Lembre-se de que a depuração de problemas de ambiente e configuração pode exigir paciência e atenção aos detalhes nos logs.