Guia de Integração do Supabase Storage com Django

Este guia detalhado irá orientá-lo na configuração do seu projeto Django para utilizar o Supabase Storage para armazenamento e carregamento de imagens, além de configurar o Railway para servir essas imagens.

1. Configuração do Supabase Storage

1.1. Criar um Projeto Supabase

Se você ainda não tem um projeto Supabase, crie um em https://supabase.com/.

1.2. Criar um Bucket no Supabase

1. No painel do seu projeto Supabase, navegue até a seção 'Storage'.
2. Clique em 'New bucket'.
3. Dê um nome ao seu bucket (ex: `django-images`).
4. Defina as políticas de segurança. Para um início rápido, você pode torná-lo público, mas para produção, é crucial configurar políticas de acesso mais restritivas. Vamos detalhar as políticas na próxima seção.

1.3. Configurar Políticas de Acesso (Policies)

As políticas de acesso no Supabase são baseadas em RLS (Row Level Security) e permitem controlar quem pode ler, escrever e listar arquivos em seus buckets. Para acessar as políticas:

1. No painel 'Storage', clique no seu bucket recém-criado.
2. Vá para a aba 'Policies'.

Exemplo de Políticas Comuns:

Política para Leitura Pública (para imagens que devem ser acessíveis a todos):

```sql
CREATE POLICY "Public access for images" ON storage.objects FOR SELECT USING
(bucket_id = 'django-images');
```

Esta política permite que qualquer pessoa leia objetos no bucket 'django-images'.

Política para Upload Autenticado (apenas usuários logados podem fazer upload):

```sql
CREATE POLICY "Authenticated upload for images" ON storage.objects FOR INSERT
WITH CHECK (bucket_id = 'django-images' AND auth.role() = 'authenticated');
```

Esta política permite que apenas usuários autenticados façam upload de arquivos para o bucket 'django-images'.

Política para Exclusão Autenticada (apenas usuários logados podem excluir):

```sql
CREATE POLICY "Authenticated delete for images" ON storage.objects FOR DELETE
USING (bucket_id = 'django-images' AND auth.role() = 'authenticated');
```

Esta política permite que apenas usuários autenticados excluam arquivos do bucket 'django-images'.

Observações Importantes sobre Políticas:

- `bucket_id` : Substitua `'django-images'` pelo nome do seu bucket.
- `auth.role()` : Retorna o papel do usuário autenticado. Outros papéis podem ser usados dependendo da sua configuração de autenticação.
- Para um ambiente de desenvolvimento, você pode começar com políticas mais permissivas e, em seguida, apertá-las para produção.

- Sempre teste suas políticas cuidadosamente para garantir que o acesso esteja configurado conforme o esperado.

2. Configuração do Django

Para integrar o Supabase Storage com o Django, você precisará de um backend de armazenamento personalizado. A biblioteca `django-storages` é uma opção popular, mas ela não tem suporte nativo para Supabase. No entanto, existem projetos que estendem essa funcionalidade ou fornecem uma solução customizada.

Vamos focar na abordagem de usar um backend de armazenamento customizado, que é a mais indicada para o Supabase.

2.1. Instalar Dependências

Você precisará de uma biblioteca para interagir com a API do Supabase Storage. A biblioteca `supabase-py` pode ser útil, mas para o armazenamento de arquivos, uma abordagem mais direta via `requests` ou um cliente S3 compatível pode ser necessária, já que o Supabase Storage é compatível com S3.

No entanto, o projeto `django-storage-supabase` (mencionado nas pesquisas) parece ser a solução mais adequada. Vamos instalá-lo:

Bash

```bash
pip install django-storage-supabase
```

2.2. Configurar `settings.py`

Adicione as seguintes configurações ao seu `settings.py`:

Python

```python
# settings.py

import os

# ... outras configurações ...
```

```
# Supabase Storage Settings
SUPABASE_URL = os.environ.get('SUPABASE_URL')
SUPABASE_KEY = os.environ.get('SUPABASE_KEY') # Sua chave anon ou
service_role
SUPABASE_BUCKET_NAME = 'django-images' # Substitua pelo nome do seu bucket

# Configuração do Default File Storage
DEFAULT_FILE_STORAGE = 'django_storage_supabase.storage.SupabaseStorage'

# Opcional: Configurar o URL base para arquivos de mídia
MEDIA_URL = f"
{SUPABASE_URL}/storage/v1/object/public/{SUPABASE_BUCKET_NAME}/"

# Se você precisar de um backend de armazenamento diferente para arquivos
estáticos
# STATICFILES_STORAGE =
'whitenoise.storage.CompressedManifestStaticFilesStorage'

# ... outras configurações ...
```

Explicação:

- `SUPABASE_URL` e `SUPABASE_KEY`: Obtenha essas informações no painel do seu projeto Supabase, em 'Settings' -> 'API'. É altamente recomendável usar variáveis de ambiente para essas chaves.
- `SUPABASE_BUCKET_NAME`: O nome do bucket que você criou no Supabase.
- `DEFAULT_FILE_STORAGE`: Aponta para o backend de armazenamento customizado fornecido pela biblioteca `django-storage-supabase`.
- `MEDIA_URL`: Define o URL base para seus arquivos de mídia. Isso é crucial para que o Django possa gerar URLs corretas para suas imagens. Certifique-se de que o caminho `storage/v1/object/public/` esteja correto, pois é o padrão para buckets públicos no Supabase.

2.3. Configurar `models.py`

Para usar o Supabase Storage para seus campos de imagem, você simplesmente define seus `ImageField` ou `FileField` como faria normalmente. O Django usará o `DEFAULT_FILE_STORAGE` configurado para lidar com o armazenamento.

Python

```python
# myapp/models.py

from django.db import models

class MyModel(models.Model):
    name = models.CharField(max_length=100)
    image = models.ImageField(upload_to='images/') # 'images/' é um
subdiretório dentro do seu bucket

    def __str__(self):
        return self.name

    def image_url(self):
        if self.image and hasattr(self.image, 'url'):
            return self.image.url
        return ""
```

Explicação:

- `upload_to='images/'` : Isso fará com que as imagens sejam salvas em um subdiretório chamado `images` dentro do seu bucket do Supabase. Por exemplo, se o nome do seu bucket for `django-images` , a imagem será salva em `django-images/images/sua_imagem.jpg` .
- `image_url()` : Um método auxiliar para obter o URL da imagem. O Django, usando o `DEFAULT_FILE_STORAGE` , gerará o URL completo para a imagem no Supabase.

2.4. Configurar `urls.py` (Opcional, para desenvolvimento)

Durante o desenvolvimento, você pode precisar servir arquivos de mídia localmente. No entanto, como estamos usando o Supabase Storage, o Django não servirá essas imagens diretamente. O `MEDIA_URL` configurado em `settings.py` já aponta para o Supabase.

Se você precisar de uma configuração para servir arquivos de mídia localmente em um ambiente de desenvolvimento (por exemplo, antes de configurar o Supabase), você faria isso no seu `urls.py` principal:

Python

```python
# yourproject/urls.py

from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    # ... suas outras URLs ...
]

# APENAS PARA DESENVOLVIMENTO: Não use em produção com Supabase Storage
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)
```

Importante: Esta configuração `static(settings.MEDIA_URL, ...)` é **apenas para desenvolvimento local** quando você não está usando um serviço de armazenamento externo. Com o Supabase Storage, o Django não precisa servir as imagens, pois elas serão acessadas diretamente do Supabase via `MEDIA_URL`.

3. Configuração do Railway

O Railway é uma plataforma de hospedagem que facilita a implantação de aplicações. Para que seu projeto Django no Railway possa interagir corretamente com o Supabase Storage, você precisa garantir que as 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:
    - `SUPABASE_URL`: O URL da sua API Supabase (ex: `https://your-project-ref.supabase.co`)

- `SUPABASE_KEY`: Sua chave `anon` ou `service_role` do Supabase. Para produção, a chave `service_role` é mais segura para operações de backend, mas certifique-se de que ela não seja exposta no frontend.

3.2. Considerações de Segurança

- **Chaves de API:** Nunca exponha sua `SUPABASE_KEY` (especialmente a `service_role`) no código-fonte do seu frontend. Use variáveis de ambiente e acesse-as apenas no backend do Django.
- **Políticas de Acesso:** Revise e ajuste suas políticas de acesso no Supabase para garantir que apenas usuários autorizados possam realizar operações de upload, download e exclusão.
- **CORS:** Se você estiver acessando o Supabase Storage de um domínio diferente (o que é comum em aplicações web), certifique-se de configurar as políticas de CORS (Cross-Origin Resource Sharing) no Supabase para permitir o acesso do seu domínio do Railway.

3.3. Implantação e Teste

Após configurar as variáveis de ambiente no Railway e garantir que seu código Django esteja atualizado com as configurações do Supabase Storage:

1. Implante 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.

Este guia deve fornecer um ponto de partida sólido para integrar o Supabase Storage com seu projeto Django hospedado no Railway. Lembre-se de adaptar as políticas de segurança e as configurações de acordo com as necessidades específicas do seu projeto.