🔐 Configuração OAuth Strava - Guia Completo

📋 Visão Geral

Este documento explica em detalhes como configurar a integração OAuth com o Strava API, incluindo:
- Como criar uma aplicação no Strava
- Como obter credenciais (Client ID e Client Secret)
- Como configurar callbacks
- Como implementar o fluxo OAuth 2.0
- Como renovar tokens automaticamente
- Troubleshooting de problemas comuns

🚀 Passo 1: Criar Aplicação no Strava

1.1. Acesse o Strava Developers

1. Vá para: https://www.strava.com/settings/api
2. Faça login com sua conta Strava (crie uma se não tiver)

1.2. Criar Nova Aplicação

1. Clique em **“Create & Manage Your App”** ou **“My API Application”**
2. Preencha o formulário:

Application Name:

```
Maratona Training
```

Category:

```
Training
```

Club (opcional):

```
Deixe em branco
```

Website:

```
https://42maurillio.abacusai.app
```

(Use http://localhost:3000 para desenvolvimento)

Application Description:

```
Plataforma de treinamento de corrida com geração de planos personalizados usando IA.
Integração com Strava para sincronização automática de treinos e análise de perform-
ance.
```

Authorization Callback Domain:

```
42maurillio.abacusai.app
```

(Para desenvolvimento: localhost)

Upload Icon (opcional):
Envie um ícone da sua aplicação (256x256 px mínimo)

1. Leia e aceite os **Termos de Uso da API**
2. Clique em **“Create”**

1.3. Obter Credenciais

Após criar a aplicação, você verá:

```
Client ID: 123456
Client Secret: abc123def456ghi789jkl012mno345pqr678stu
```

⚠️ **IMPORTANTE**:
- **Nunca compartilhe** o Client Secret publicamente
- **Não commite** no Git (use .env)
- Trate como senha

🔧 Passo 2: Configurar Ambiente

2.1. Adicionar Variáveis ao .env

No arquivo `.env` na raiz do projeto:

```
# Strava OAuth
STRAVA_CLIENT_ID="123456"
STRAVA_CLIENT_SECRET="abc123def456ghi789jkl012mno345pqr678stu"
STRAVA_REDIRECT_URI="http://localhost:3000/api/strava/callback"

# Em produção, use:
# STRAVA_REDIRECT_URI="https://42maurillio.abacusai.app/api/strava/callback"
```

2.2. Verificar Callback Route

Certifique-se de que existe o arquivo:
`app/api/strava/callback/route.ts`

🔄 Passo 3: Entender o Fluxo OAuth 2.0

Visão Geral do Fluxo

```
[1] Usuário → Clica "Conectar Strava"
         ↓
[2] App → Redireciona para Strava.com
         ↓
[3] Usuário → Autoriza na página do Strava
         ↓
[4] Strava → Redireciona de volta com 'code'
         ↓
[5] App → Troca 'code' por tokens
         ↓
[6] App → Salva tokens no banco
         ↓
[7] App → Usa access_token para fazer requests
```

Detalhamento de Cada Etapa

Etapa 1: Usuário Clica “Conectar Strava”

Frontend (`components/strava-connect.tsx`):

```
const handleConnect = () => {
  // Redireciona para endpoint de auth
  window.location.href = '/api/strava/auth';
};

return (
  <Button onClick={handleConnect}>
    Conectar Strava
  </Button>
);
```

Etapa 2: App Redireciona para Strava

Backend (`app/api/strava/auth/route.ts`):

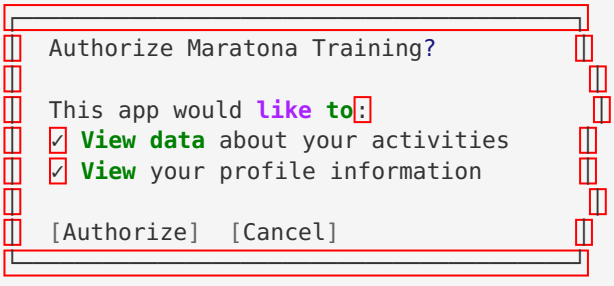
```
import { NextResponse } from 'next/server';
import { getServerSession } from 'next-auth';

export async function GET(req: Request) {
  const session = await getServerSession();

  if (!session) {
    return NextResponse.redirect('/login');
  }

  const stravaAuthUrl = new URL('https://www.strava.com/oauth/authorize');

  stravaAuthUrl.searchParams.append('client_id', process.env.STRAVA_CLIENT_ID!);
  stravaAuthUrl.searchParams.append('redirect_uri', process.env.STRAVA_REDIRECT_URI!);
  stravaAuthUrl.searchParams.append('response_type', 'code');
  stravaAuthUrl.searchParams.append('scope',
'read,activity:read_all,profile:read_all');
  stravaAuthUrl.searchParams.append('state', session.user.id); // Para identificar
usuário depois

  return NextResponse.redirect(stravaAuthUrl.toString());
}
```

URL gerada (exemplo):

```
https://www.strava.com/oauth/authorize?
  client_id=123456&
  redirect_uri=http://localhost:3000/api/strava/callback&
  response_type=code&
  scope=read,activity:read_all,profile:read_all&
  state=user-id-uuid
```

Etapa 3: Usuário Autoriza no Strava

O Strava exibe uma tela de autorização:



Se usuário clicar **Authorize**, Strava redireciona para callback.

Etapa 4: Strava Redireciona com Code

O Strava redireciona para:

```
http://localhost:3000/api/strava/callback?
  state=user-id-uuid&
  code=abc123def456&
  scope=read,activity:read_all,profile:read_all
```

Etapa 5: App Troca Code por Tokens

Backend (`app/api/strava/callback/route.ts`):

```
import { NextResponse } from 'next/server';
import { prisma } from '@/lib/db';

export async function GET(req: Request) {
  const url = new URL(req.url);
  const code = url.searchParams.get('code');
  const state = url.searchParams.get('state'); // user ID

  if (!code || !state) {
    return NextResponse.redirect('/dashboard?error=strava_auth_failed');
  }

  try {
    // Troca code por tokens
    const tokenResponse = await fetch('https://www.strava.com/oauth/token', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify({
        client_id: process.env.STRAVA_CLIENT_ID,
        client_secret: process.env.STRAVA_CLIENT_SECRET,
        code,
        grant_type: 'authorization_code'
      })
    });

    const tokenData = await tokenResponse.json();

    if (tokenData.errors) {
      throw new Error(tokenData.message);
    }

    // Salva no banco
    await prisma.stravaConnection.upsert({
      where: { userId: state },
      update: {
        accessToken: tokenData.access_token,
        refreshToken: tokenData.refresh_token,
        expiresAt: tokenData.expires_at,
        athleteId: tokenData.athlete.id.toString(),
        lastSync: new Date()
      },
      create: {
        userId: state,
        accessToken: tokenData.access_token,
        refreshToken: tokenData.refresh_token,
        expiresAt: tokenData.expires_at,
        athleteId: tokenData.athlete.id.toString()
      }
    });

    return NextResponse.redirect('/dashboard?strava=connected');

  } catch (error) {
    console.error('Strava OAuth error:', error);
    return NextResponse.redirect('/dashboard?error=strava_connection_failed');
  }
}
```

Resposta da API do Strava (tokenData):

```
{
  "token_type": "Bearer",
  "expires_at": 1735689600,
  "expires_in": 21600,
  "refresh_token": "abc123def456...",
  "access_token": "xyz789ghi012...",
  "athlete": {
    "id": 12345678,
    "username": "joaosilva",
    "firstname": "João",
    "lastname": "Silva",
    ...
  }
}
```

Etapa 6: Salvar Tokens no Banco

O schema Prisma (`prisma/schema.prisma`):

```
model StravaConnection {
  id           String   @id @default(uuid())
  userId       String   @unique
  user         User     @relation(fields: [userId], references: [id], onDelete: Cascade)

  accessToken  String   // Token de acesso (válido por ~6h)
  refreshToken String   // Token para renovar access_token
  expiresAt    BigInt   // Timestamp Unix de expiração
  athleteId    String   // ID do atleta no Strava

  lastSync     DateTime? // Última sincronização de atividades

  createdAt    DateTime @default(now())
  updatedAt    DateTime @updatedAt
}
```

Etapa 7: Usar Access Token

Buscar atividades (`lib/strava.ts`):

```
export async function getRecentActivities(userId: string) {
  // Pega token válido (renova se necessário)
  const accessToken = await getValidAccessToken(userId);

  const response = await fetch(
    'https://www.strava.com/api/v3/athlete/activities?page=1&per_page=30',
    {
      headers: {
        'Authorization': `Bearer ${accessToken}`
      }
    }
  );

  if (!response.ok) {
    throw new Error(`Strava API error: ${response.statusText}`);
  }

  return await response.json();
}
```

🔄 Renovação Automática de Tokens

Por que Renovar?

O `access_token` expira em **6 horas**. Após isso, requisições retornam 401 Unauthorized.

Como Renovar?

Função de renovação (`lib/strava.ts`):

```
export async function getValidAccessToken(userId: string): Promise<string> {
  const connection = await prisma.stravaConnection.findUnique({
    where: { userId }
  });

  if (!connection) {
    throw new Error('Strava não conectado');
  }

  const now = Math.floor(Date.now() / 1000); // Unix timestamp
  const expiresAt = Number(connection.expiresAt);

  // Se token ainda válido (com 5 min de margem)
  if (expiresAt > now + 300) {
    return connection.accessToken;
  }

  // Token expirado, renova
  console.log('🔄 Renovando token Strava para usuário:', userId);

  const response = await fetch('https://www.strava.com/oauth/token', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({
      client_id: process.env.STRAVA_CLIENT_ID,
      client_secret: process.env.STRAVA_CLIENT_SECRET,
      refresh_token: connection.refreshToken,
      grant_type: 'refresh_token'
    })
  });

  const data = await response.json();

  if (data.errors) {
    throw new Error(`Falha ao renovar token: ${data.message}`);
  }

  // Atualiza no banco
  await prisma.stravaConnection.update({
    where: { userId },
    data: {
      accessToken: data.access_token,
      refreshToken: data.refresh_token,
      expiresAt: data.expires_at
    }
  });

  return data.access_token;
}
```

Sempre use `getValidAccessToken()`

❌ **Errado**:

```
const connection = await prisma.stravaConnection.findUnique({...});
fetch('https://www.strava.com/api/v3/...', {
  headers: { 'Authorization': `Bearer ${connection.accessToken}` }
});
```

✅ **Correto**:

```
const accessToken = await getValidAccessToken(userId);
fetch('https://www.strava.com/api/v3/...', {
  headers: { 'Authorization': `Bearer ${accessToken}` }
});
```

🔍 Escopos do Strava

Escopos Disponíveis

Escopo	Acesso
`read`	Leitura básica de perfil
`read_all`	Leitura de todas as atividades (incluindo privadas)
`profile:read_all`	Leitura completa de perfil
`profile:write`	Edição de perfil
`activity:read`	Leitura de atividades públicas
`activity:read_all`	Leitura de todas as atividades
`activity:write`	Criação/edição de atividades

Escopos Usados no Projeto

```
scope: 'read,activity:read_all,profile:read_all'
```

Por que estes escopos?
- `read` : Básico para autenticação
- `activity:read_all` : Para buscar todas as corridas (incluindo privadas)
- `profile:read_all` : Para obter nome, foto, etc.

⚠️ **Nota**: Não usamos `write` pois não vamos criar atividades no Strava.

🛠️ Troubleshooting

Problema 1: “Invalid redirect_uri”

Sintoma: Erro ao clicar “Conectar Strava”

Causa: `redirect_uri` não corresponde ao configurado no Strava

Solução:
1. Vá para https://www.strava.com/settings/api
2. Verifique **Authorization Callback Domain**
3. Deve ser apenas o domínio (sem path):
- ✅ `localhost` (dev)
- ✅ `42maurillio.abacusai.app` (prod)
- ❌ `localhost:3000/api/strava/callback`

1. No `.env`, o `STRAVA_REDIRECT_URI` deve incluir o path:
   `bash`
   `STRAVA_REDIRECT_URI="http://localhost:3000/api/strava/callback"`

Problema 2: “Bad Request” ao trocar code

Sintoma: Erro 400 ao callback

Causa: Possíveis:
- `client_id` ou `client_secret` incorretos
- `code` já foi usado (códigos são one-time use)
- `code` expirou (válido por 30 segundos)

Solução:
1. Verifique credenciais no `.env`
2. Não recarregue a página de callback (código expira)
3. Inicie o fluxo novamente

Problema 3: “Unauthorized” ao fazer requests

Sintoma: 401 ao buscar atividades

Causa: Token expirado e não renovado

Solução:
Certifique-se de usar `getValidAccessToken()`:

```
const token = await getValidAccessToken(userId);
```

Problema 4: Escopos insuficientes

Sintoma: API retorna atividades vazias ou erro de permissão

Causa: Escopos não incluem `activity:read_all`

Solução:
1. Verificar escopos no código:
`typescript`
`scope: 'read,activity:read_all,profile:read_all'`
2. Se usuário já autorizou com escopos antigos, precisa re-autorizar
3. Ofereça botão “Reconectar Strava”

Problema 5: Callback não funciona em produção

Sintoma: Funciona em localhost, falha em produção

Causa: HTTPS vs HTTP

Solução:
1. Certifique-se que produção usa HTTPS
2. Atualize `STRAVA_REDIRECT_URI` em produção:
```bash
STRAVA_REDIRECT_URI="https://42maurillio.abacusai.app/api/strava/callback"
```
3. Atualize **Authorization Callback Domain** no Strava:
`42maurillio.abacusai.app`

📡 Testando a Integração

Teste Manual

1. **Iniciar servidor**:
   ```bash
   cd /home/ubuntu/app_maratona/nextjs_space
   yarn dev
   ```
2. **Fazer login**: http://localhost:3000/login
3. **Ir para dashboard**: http://localhost:3000/dashboard
4. **Clicar em "Conectar Strava"**
5. **Autorizar no Strava**
6. **Verificar redirecionamento**:
   - Deve voltar para dashboard
   - Deve ver "Strava conectado"
7. **Verificar no banco**:
   ```bash
   yarn prisma studio
   # Abra StravaConnection
   # Verifique se há registro para seu usuário
   ```
8. **Testar busca de atividades**:
   - No dashboard, veja se aparece opção "Sincronizar Strava"
   - Clique e veja se atividades aparecem

Teste Programático

Script de teste (`test_strava_integration.ts`):

```
import { getRecentActivities, getValidAccessToken } from './lib/strava';

async function testStrava() {
  const userId = 'seu-user-id-aqui';

  try {
    console.log('1. Verificando token...');
    const token = await getValidAccessToken(userId);
    console.log('✅ Token obtido');

    console.log('\n2. Buscando atividades...');
    const activities = await getRecentActivities(userId);
    console.log(`✅ ${activities.length} atividades encontradas`);

    console.log('\n3. Primeira atividade:');
    console.log(JSON.stringify(activities[0], null, 2));

  } catch (error) {
    console.error('❌ Erro:', error.message);
  }
}

testStrava();
```

Executar:

```
yarn ts-node test_strava_integration.ts
```

📊 Endpoints da API Strava Usados

1. Buscar Atividades do Atleta

GET `https://www.strava.com/api/v3/athlete/activities`

Query Params:
- `page`: Número da página (default: 1)
- `per_page`: Atividades por página (max: 200)
- `before`: Unix timestamp (atividades antes desta data)
- `after`: Unix timestamp (atividades depois desta data)

Response:

```
[
  {
    "id": 12345678,
    "name": "Morning Run",
    "distance": 10000.0,
    "moving_time": 3000,
    "elapsed_time": 3100,
    "type": "Run",
    "start_date": "2025-10-27T06:00:00Z",
    "average_speed": 3.33,
    "max_speed": 5.5,
    "average_heartrate": 155.0,
    "max_heartrate": 178.0
  }
]
```


---

2. Buscar Detalhes de Atividade

GET `https://www.strava.com/api/v3/activities/:id`

Response: Inclui todos os campos acima + streams (GPS, HR, etc.)

---

3. Buscar Streams de Atividade

GET `https://www.strava.com/api/v3/activities/:id/streams`

Query Params:
- `keys`: `time,latlng,distance,altitude,heartrate,cadence`
- `key_by_type`: `true`

Response: Arrays de dados por segundo/metro

🔐 Segurança

Boas Práticas

1. **Nunca exponha Client Secret**
   - Mantenha em `.env`
   - Adicione `.env` ao `.gitignore`

2. **Valide State**
   - Use `state` para prevenir CSRF
   - Sempre verifique se `state` corresponde ao usuário esperado

3. **Criptografe Tokens**
   - Em produção, considere criptografar tokens no banco
   - Use lib como `crypto` do Node.js

4. **Rate Limiting**
   - Strava limita a 100 requests / 15 min / usuário

- 1000 requests / dia / usuário
- Implemente retry com backoff

5. **Handle Desconexão**
   - Usuário pode revogar acesso no Strava
   - Trate erros 401 com botão “Reconectar”

📚 Recursos Adicionais

- **Documentação Oficial**: https://developers.strava.com/docs/
- **Guia de Início**: https://developers.strava.com/docs/getting-started/
- **Referência da API**: https://developers.strava.com/docs/reference/
- **Playground**: https://developers.strava.com/playground/
- **Rate Limits**: https://developers.strava.com/docs/rate-limits/
- **Webhooks**: https://developers.strava.com/docs/webhooks/

Última Atualização: 27 de outubro de 2025
Autor: Documentação do Projeto Maratona Training