Google, X.com ve Amazon OAuth 2.0 Entegrasyon Rehberi

gistify.pro için Kapsamlı Uygulama Kılavuzu

React Frontend + Python (FastAPI) Backend

İçindekiler

1. Genel Bakış ve Mimari

Bu rehber, gistify.pro uygulamanıza Google, X.com (Twitter) ve Amazon sosyal giriş özelliklerini eklemek için adım adım talimatlar sunmaktadır. OAuth 2.0 protokolü kullanılarak güvenli kimlik doğrulama sağlanacaktır.

1.1 OAuth 2.0 Akışı

Authorization Code Flow with PKCE kullanılacaktır:

1. Kullanıcı "Google/X/Amazon ile Giriş Yap" butonuna tıklar
2. Frontend, kullanıcıyı OAuth sağlayıcısına yönlendirir
3. Kullanıcı izin verir ve authorization code ile geri döner
4. Backend, code'u access token ile değiştirir
5. Backend, kullanıcı bilgilerini alır ve JWT oluşturur

1.2 Gerekli Bağımlılıklar

Backend (Python)

```
pip install fastapi uvicorn python-jose[cryptography] httpx
pip install python-dotenv pydantic sqlalchemy
```

Frontend (React)

```
npm install @react-oauth/google axios react-router-dom
```

2. Google OAuth 2.0 Entegrasyonu

2.1 Google Cloud Console Kurulumu

1. Google Cloud Console'a gidin: console.cloud.google.com
2. Yeni bir proje oluşturun veya mevcut projeyi seçin
3. APIs & Services > OAuth consent screen'e gidin
4. User Type olarak "External" seçin
5. Credentials > Create Credentials > OAuth client ID seçin
6. Application type: "Web application" seçin

Authorized JavaScript origins:

- http://localhost:3000 (development)
- https://gistify.pro (production)

Authorized redirect URIs:

- http://localhost:8000/api/auth/google/callback
- https://gistify.pro/api/auth/google/callback

□ *Client ID ve Client Secret'ı güvenli bir yerde saklayın!*

2.2 Backend - auth/google.py

```
import httpx
from fastapi import APIRouter, HTTPException
from fastapi.responses import RedirectResponse
from urllib.parse import urlencode
from config import settings
from auth.jwt_handler import create_access_token

router = APIRouter(prefix="/api/auth/google", tags=["Google Auth"])

GOOGLE_AUTH_URL = "https://accounts.google.com/o/oauth2/v2/auth"
GOOGLE_TOKEN_URL = "https://oauth2.googleapis.com/token"
GOOGLE_USERINFO_URL = "https://www.googleapis.com/oauth2/v2/userinfo"

@router.get("/login")
async def google_login():
    params = {
        "client_id": settings.GOOGLE_CLIENT_ID,
        "redirect_uri": settings.GOOGLE_REDIRECT_URI,
        "response_type": "code",
        "scope": "openid email profile",
        "access_type": "offline",
        "prompt": "consent"
    }
    return RedirectResponse(url=f"{GOOGLE_AUTH_URL}?{urlencode(params)}")

@router.get("/callback")
async def google_callback(code: str = None, error: str = None):
    if error:
        raise HTTPException(status_code=400, detail=f"OAuth error: {error}")

    async with httpx.AsyncClient() as client:
        # Token al
        token_resp = await client.post(GOOGLE_TOKEN_URL, data={
            "code": code,
            "client_id": settings.GOOGLE_CLIENT_ID,
```

```
            "client_secret": settings.GOOGLE_CLIENT_SECRET,
            "redirect_uri": settings.GOOGLE_REDIRECT_URI,
            "grant_type": "authorization_code"
        })
        tokens = token_resp.json()

        # User info al
        user_resp = await client.get(GOOGLE_USERINFO_URL,
            headers={"Authorization": f"Bearer {tokens['access_token']}"})
        user_info = user_resp.json()

    jwt_token = create_access_token({
        "sub": user_info["id"],
        "email": user_info["email"],
        "name": user_info.get("name", ""),
        "provider": "google"
    })

    return RedirectResponse(
        url=f"{settings.FRONTEND_URL}/auth/callback?token={jwt_token}"
    )
```

2.3 Frontend - React Google Login

```
// App.jsx
import { GoogleOAuthProvider } from '@react-oauth/google';
import { BrowserRouter, Routes, Route } from 'react-router-dom';

function App() {
  return (
    <GoogleOAuthProvider clientId={import.meta.env.VITE_GOOGLE_CLIENT_ID}>
      <BrowserRouter>
        <Routes>
          <Route path="/login" element={<Login />} />
          <Route path="/auth/callback" element={<AuthCallback />} />
        </Routes>
      </BrowserRouter>
    </GoogleOAuthProvider>
  );
}

// Login.jsx - Google Button
const handleGoogleLogin = () => {
  window.location.href = `${API_URL}/api/auth/google/login`;
};

<button onClick={handleGoogleLogin} className="google-btn">
  <img src="/google-icon.svg" alt="Google" width="20" />
  Google ile Giriş Yap
</button>
```

3. X.com (Twitter) OAuth 2.0 Entegrasyonu

3.1 X Developer Portal Kurulumu

1. X Developer Portal'a gidin: developer.twitter.com
2. Yeni bir proje ve uygulama oluşturun
3. User authentication settings > OAuth 2.0'ı etkinleştirin
4. App permissions: Read (veya ihtiyaca göre)
5. Type of App: Web App, Automated App or Bot

Callback URI / Redirect URL:

- http://localhost:8000/api/auth/twitter/callback
- https://gistify.pro/api/auth/twitter/callback

 X API Free tier ile sadece Read erişimi alabilirsiniz. Tweet post etmek için Basic ($100/ay) gereklidir.

3.2 Backend - auth/twitter.py

```
import httpx
import secrets
import hashlib
import base64
from fastapi import APIRouter, HTTPException
from fastapi.responses import RedirectResponse
from urllib.parse import urlencode
from config import settings
from auth.jwt_handler import create_access_token

router = APIRouter(prefix="/api/auth/twitter", tags=["Twitter Auth"])

TWITTER_AUTH_URL = "https://twitter.com/i/oauth2/authorize"
TWITTER_TOKEN_URL = "https://api.twitter.com/2/oauth2/token"
TWITTER_USERINFO_URL = "https://api.twitter.com/2/users/me"

# PKCE state cache (production'da Redis kullanın)
auth_states = {}

def generate_pkce():
    code_verifier = secrets.token_urlsafe(32)
    code_challenge = base64.urlsafe_b64encode(
        hashlib.sha256(code_verifier.encode()).digest()
    ).decode().rstrip("=")
    return code_verifier, code_challenge

@router.get("/login")
async def twitter_login():
    state = secrets.token_urlsafe(16)
    code_verifier, code_challenge = generate_pkce()
    auth_states[state] = code_verifier

    params = {
        "response_type": "code",
        "client_id": settings.TWITTER_CLIENT_ID,
        "redirect_uri": settings.TWITTER_REDIRECT_URI,
        "scope": "tweet.read users.read offline.access",
        "state": state,
        "code_challenge": code_challenge,
        "code_challenge_method": "S256"
```

```
    }
    return RedirectResponse(url=f"{TWITTER_AUTH_URL}?{urlencode(params)}")

@router.get("/callback")
async def twitter_callback(code: str, state: str):
    code_verifier = auth_states.pop(state, None)
    if not code_verifier:
        raise HTTPException(status_code=400, detail="Invalid state")

    credentials = f"{settings.TWITTER_CLIENT_ID}:{settings.TWITTER_CLIENT_SECRET}"
    basic_auth = base64.b64encode(credentials.encode()).decode()

    async with httpx.AsyncClient() as client:
        token_resp = await client.post(TWITTER_TOKEN_URL,
            headers={
                "Content-Type": "application/x-www-form-urlencoded",
                "Authorization": f"Basic {basic_auth}"
            },
            data={
                "code": code,
                "grant_type": "authorization_code",
                "redirect_uri": settings.TWITTER_REDIRECT_URI,
                "code_verifier": code_verifier
            })
        tokens = token_resp.json()

        user_resp = await client.get(TWITTER_USERINFO_URL,
            headers={"Authorization": f"Bearer {tokens['access_token']}"},
            params={"user.fields": "id,name,username,profile_image_url"})
        user_data = user_resp.json()["data"]

    jwt_token = create_access_token({
        "sub": user_data["id"],
        "name": user_data["name"],
        "username": user_data["username"],
        "provider": "twitter"
    })

    return RedirectResponse(
        url=f"{settings.FRONTEND_URL}/auth/callback?token={jwt_token}"
    )
```

3.3 Frontend - X Login Button

```
// components/TwitterLoginButton.jsx
const API_URL = import.meta.env.VITE_API_URL;

function TwitterLoginButton() {
  const handleTwitterLogin = () => {
    window.location.href = `${API_URL}/api/auth/twitter/login`;
  };

  return (
    <button onClick={handleTwitterLogin} className="twitter-btn">
      <svg viewBox="0 0 24 24" width="20" height="20" fill="currentColor">
        <path d="M18.244 2.25h3.308l-7.227 8.26 8.502 11.24H16.17l-5.214-6.817L4.99
21.75H1.68l7.73-8.835L1.254 2.25H8.08l4.713 6.231zm-1.161 17.52h1.833L7.084
4.126H5.117z"/>
      </svg>
      X ile Giriş Yap
    </button>
  );
}
```

4. Amazon (Login with Amazon) OAuth 2.0 Entegrasyonu

4.1 Amazon Developer Console Kurulumu

1. Amazon Developer Portal'a gidin: developer.amazon.com
2. Apps & Services > Login with Amazon seçin
3. Create a New Security Profile tıklayın
4. Security Profile Name ve Description girin
5. Consent Privacy Notice URL: https://gistify.pro/privacy
6. Security Profile oluşturduktan sonra gear icon > Web Settings

Allowed Origins:

- http://localhost:3000
- https://gistify.pro

Allowed Return URLs:

- http://localhost:8000/api/auth/amazon/callback
- https://gistify.pro/api/auth/amazon/callback

□ *Client ID ve Client Secret'ı Web Settings sayfasından alabilirsiniz.*

4.2 Amazon OAuth Endpoints

Login with Amazon aşağıdaki endpoint'leri kullanır:

- Authorization: https://www.amazon.com/ap/oa
- Token: https://api.amazon.com/auth/o2/token
- User Profile: https://api.amazon.com/user/profile

Bölgesel endpoint'ler:

- North America (NA): api.amazon.com
- European Union (EU): api.amazon.co.uk
- Far East (FE): api.amazon.co.jp

4.3 Backend - auth/amazon.py

```
import httpx
import secrets
from fastapi import APIRouter, HTTPException
from fastapi.responses import RedirectResponse
from urllib.parse import urlencode
from config import settings
from auth.jwt_handler import create_access_token

router = APIRouter(prefix="/api/auth/amazon", tags=["Amazon Auth"])

AMAZON_AUTH_URL = "https://www.amazon.com/ap/oa"
AMAZON_TOKEN_URL = "https://api.amazon.com/auth/o2/token"
AMAZON_PROFILE_URL = "https://api.amazon.com/user/profile"

# State cache (production'da Redis kullanın)
auth_states = {}

@router.get("/login")
async def amazon_login():
```

```
    """Amazon OAuth login başlatır"""
    state = secrets.token_urlsafe(16)
    auth_states[state] = True

    params = {
        "client_id": settings.AMAZON_CLIENT_ID,
        "redirect_uri": settings.AMAZON_REDIRECT_URI,
        "response_type": "code",
        "scope": "profile",
        "state": state
    }
    return RedirectResponse(url=f"{AMAZON_AUTH_URL}?{urlencode(params)}")

@router.get("/callback")
async def amazon_callback(code: str = None, state: str = None, error: str = None):
    """Amazon OAuth callback işler"""
    if error:
        raise HTTPException(status_code=400, detail=f"OAuth error: {error}")

    if not code or not state:
        raise HTTPException(status_code=400, detail="Missing code or state")

    # State doğrula
    if not auth_states.pop(state, None):
        raise HTTPException(status_code=400, detail="Invalid state")

    async with httpx.AsyncClient() as client:
        # Authorization code'u access token ile değiştir
        token_response = await client.post(
            AMAZON_TOKEN_URL,
            headers={"Content-Type": "application/x-www-form-urlencoded"},
            data={
                "grant_type": "authorization_code",
                "code": code,
                "redirect_uri": settings.AMAZON_REDIRECT_URI,
                "client_id": settings.AMAZON_CLIENT_ID,
                "client_secret": settings.AMAZON_CLIENT_SECRET
            }
        )

        if token_response.status_code != 200:
            raise HTTPException(
                status_code=400,
                detail=f"Token exchange failed: {token_response.text}"
            )

        tokens = token_response.json()
        access_token = tokens.get("access_token")

        # Kullanıcı profil bilgilerini al
        profile_response = await client.get(
            AMAZON_PROFILE_URL,
            headers={"Authorization": f"Bearer {access_token}"}
        )

        if profile_response.status_code != 200:
            raise HTTPException(status_code=400, detail="Failed to get user
profile")

        user_profile = profile_response.json()

    # JWT token oluştur
    jwt_token = create_access_token({
        "sub": user_profile["user_id"],
```

```
        "email": user_profile.get("email", ""),
        "name": user_profile.get("name", ""),
        "provider": "amazon"
    })

    # Frontend'e redirect et
    redirect_url = f"{settings.FRONTEND_URL}/auth/callback?token={jwt_token}"
    return RedirectResponse(url=redirect_url)
```

4.4 Frontend - Amazon Login Button

```
// components/AmazonLoginButton.jsx
import React from 'react';

const API_URL = import.meta.env.VITE_API_URL || 'http://localhost:8000';

function AmazonLoginButton() {
  const handleAmazonLogin = () => {
    window.location.href = `${API_URL}/api/auth/amazon/login`;
  };

  return (
    <button onClick={handleAmazonLogin} className="amazon-btn">
      <svg viewBox="0 0 24 24" width="20" height="20" fill="currentColor">
        <path d="M13.958 10.09c0 1.232.029 2.256-.591 3.351-.502.891-1.301 1.44-
2.186 1.44-1.214 0-1.922-.924-1.922-2.292 0-2.692 2.415-3.182 4.7-3.182v.683zm3.186
7.705c-.209.189-.512.201-.745.074-1.052-.872-1.238-1.276-1.814-2.106-1.734 1.767-
2.962 2.297-5.209 2.297-2.66 0-4.731-1.641-4.731-4.925 0-2.565 1.391-4.309 3.37-
5.164 1.715-.754 4.11-.891 5.942-1.095v-.41c0-.753.06-1.642-.383-2.294-.385-.579-
1.124-.82-1.775-.82-1.205 0-2.277.618-2.54 1.897-.054.285-.261.567-.549.582l-3.061-
.333c-.259-.056-.548-.266-.472-.66C6.057 1.926 9.311 1 12.152 1c1.435 0 3.313.383
4.444 1.473 1.435 1.335 1.298 3.115 1.298 5.054v4.58c0 1.378.572 1.981 1.11
2.726.187.265.229.582-.009.778-.595.5-1.651 1.425-2.234 1.94l-.617.244z"/>
        <path d="M20.176 19.006c-2.171 1.685-5.314 2.586-8.024 2.586-3.797 0-7.216-
1.519-9.802-4.047-.204-.195-.022-.461.222-.311 2.792 1.761 6.244 2.82 9.808 2.82
2.405 0 5.049-.539 7.481-1.656.367-.171.675.26.315.608z"/>
        <path d="M21.211 17.707c-.277-.378-1.832-.179-2.532-.09-.212.027-.245-.171-
.054-.314 1.241-.924 3.278-.658 3.516-.348.238.313-.063 2.476-1.227 3.509-.179.159-
.349.074-.27-.136.262-.695.849-2.244.567-2.621z"/>
      </svg>
      Amazon ile Giriş Yap
    </button>
  );
}

export default AmazonLoginButton;
```

4.5 Amazon Login Button Stili

```
/* styles/auth-buttons.css */
.amazon-btn {
  display: flex;
  align-items: center;
  justify-content: center;
  gap: 12px;
  width: 100%;
  padding: 12px 24px;
  background: linear-gradient(to bottom, #f7d066 0%, #f0c14b 100%);
  border: 1px solid #a88734;
  border-radius: 4px;
  color: #111;
  font-size: 14px;
  font-weight: 500;
  cursor: pointer;
```

```
  transition: all 0.2s ease;
}

.amazon-btn:hover {
  background: linear-gradient(to bottom, #f5c645 0%, #e5a735 100%);
}

.amazon-btn svg {
  color: #111;
}
```

5. Güvenlik ve En İyi Uygulamalar

5.1 Environment Variables

Backend .env

```
# Google OAuth
GOOGLE_CLIENT_ID=your-google-client-id.apps.googleusercontent.com
GOOGLE_CLIENT_SECRET=your-google-client-secret
GOOGLE_REDIRECT_URI=http://localhost:8000/api/auth/google/callback

# Twitter/X OAuth
TWITTER_CLIENT_ID=your-twitter-client-id
TWITTER_CLIENT_SECRET=your-twitter-client-secret
TWITTER_REDIRECT_URI=http://localhost:8000/api/auth/twitter/callback

# Amazon OAuth
AMAZON_CLIENT_ID=amzn1.application-oa2-client.xxxxx
AMAZON_CLIENT_SECRET=your-amazon-client-secret
AMAZON_REDIRECT_URI=http://localhost:8000/api/auth/amazon/callback

# JWT
JWT_SECRET_KEY=your-super-secret-key-min-32-characters
JWT_ALGORITHM=HS256

# App
FRONTEND_URL=http://localhost:3000
```

Frontend .env

```
VITE_GOOGLE_CLIENT_ID=your-google-client-id.apps.googleusercontent.com
VITE_API_URL=http://localhost:8000
```

5.2 Config Dosyası

```
# config.py
from pydantic_settings import BaseSettings

class Settings(BaseSettings):
    # Google
    GOOGLE_CLIENT_ID: str
    GOOGLE_CLIENT_SECRET: str
    GOOGLE_REDIRECT_URI: str

    # Twitter/X
    TWITTER_CLIENT_ID: str
    TWITTER_CLIENT_SECRET: str
    TWITTER_REDIRECT_URI: str

    # Amazon
    AMAZON_CLIENT_ID: str
    AMAZON_CLIENT_SECRET: str
    AMAZON_REDIRECT_URI: str

    # JWT
    JWT_SECRET_KEY: str
    JWT_ALGORITHM: str = "HS256"
    JWT_EXPIRE_MINUTES: int = 60 * 24

    # App
    FRONTEND_URL: str
```

```
    class Config:
        env_file = ".env"

settings = Settings()
```

5.3 Main Application

```
# main.py
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from auth.google import router as google_router
from auth.twitter import router as twitter_router
from auth.amazon import router as amazon_router

app = FastAPI(title="gistify.pro API")

origins = [
    "http://localhost:3000",
    "https://gistify.pro",
]

app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

app.include_router(google_router)
app.include_router(twitter_router)
app.include_router(amazon_router)
```

5.4 JWT Handler

```
# auth/jwt_handler.py
from datetime import datetime, timedelta
from jose import jwt, JWTError
from config import settings

def create_access_token(data: dict) -> str:
    to_encode = data.copy()
    expire = datetime.utcnow() + timedelta(minutes=settings.JWT_EXPIRE_MINUTES)
    to_encode.update({"exp": expire})
    return jwt.encode(to_encode, settings.JWT_SECRET_KEY,
                      algorithm=settings.JWT_ALGORITHM)

def verify_token(token: str) -> dict:
    try:
        return jwt.decode(token, settings.JWT_SECRET_KEY,
                          algorithms=[settings.JWT_ALGORITHM])
    except JWTError:
        return None
```

6. Sorun Giderme

6.1 Google OAuth Hataları

- Error 400: redirect_uri_mismatch → Redirect URI'yi Google Console'da kontrol edin
- Error 401: invalid_client → Client ID/Secret doğru mu?
- Error 403: access_denied → OAuth consent screen'i yayınlayın

6.2 X.com OAuth Hataları

- Error 401: Unauthorized → Client credentials'ı kontrol edin
- Error 403: Forbidden → API erişim seviyenizi kontrol edin
- Invalid state → Redis kullanarak state yönetimini geliştirin
- PKCE hataları → code_verifier/code_challenge eşleşmesini kontrol edin

6.3 Amazon OAuth Hataları

- invalid_grant → Authorization code süresi dolmuş olabilir
- unauthorized_client → Security Profile ayarlarını kontrol edin
- invalid_request → Redirect URI tam olarak eşleşmeli
- access_denied → Kullanıcı izni reddetti

6.4 Faydalı Linkler

Google: developers.google.com/identity/protocols/oauth2

X.com: developer.twitter.com/en/docs/authentication/oauth-2-0

Amazon: developer.amazon.com/docs/login-with-amazon

Sonuç

Bu rehberi takip ederek gistify.pro uygulamanıza Google, X.com ve Amazon sosyal giriş özelliklerini başarıyla entegre edebilirsiniz.

 Production'a geçmeden önce tüm güvenlik önlemlerini uyguladığınızdan emin olun!

— Rehber Sonu —