Gereksinim Analizi Belgesi

Proje Adı : GitHub Kullanıcı Analizi

Sürüm : 1.0

Tarih : [1.11.2024]

Hazırlayan : [Emirhan Sevimli/Satellite Team]

1. Giriş

1.1 Amaç

GitHub Kullanıcı Analizi projesinin amacı, GitHub kullanıcılarının kodlama alışkanlıklarını analiz etmek için veri odaklı bir platform oluşturmaktır. Blitz.gg gibi analitik araçlardan ilham alan bu proje, kullanıcılara en çok kullandıkları programlama dilleri, teknoloji kullanımındaki değişimler ve GitHub topluluğu genelindeki eğilimler hakkında içgörüler sağlayacaktır. Nihai hedef, GitHub kullanıcılarının kodlama stillerini, dil popülerliğini ve çerçeve tercihlerini anlamalarına yardımcı olmaktır.

1.2 Kapsam

Bu platform GitHub'ın API'sinden alınan verileri analiz ederek kullanıcıların şunları yapmasına olanak tanıyacak:

- Dil tercihleri ve çerçeve kullanımı gibi kişisel kodlama eğilimlerini görüntüleyin.
- Kullanıcıların diller, çerçeveler veya son etkinlik değişiklikleri konusundaki uzmanlıklarını vurgulayan rozetlere erişin.
- Popüler diller ve yaygın olarak kullanılan çerçeveler gibi topluluk eğilimlerini keşfedin.
- Zaman içinde kodlama dili tercihlerindeki bireysel ve küresel değişimleri izleyin.

1.3 Tanımlar, Kısaltmalar ve Kısaltmalar

- **API** : Uygulama Programlama Arayüzü
- **GitHub** : Sürüm kontrolü ve işbirliği için web tabanlı bir platform

- **Blitz.gg** : Çevrimiçi oyunlardaki oyuncu istatistiklerini izlemek için bir veri analitiği platformu

2. Sistem Genel Bakışı

Sistem, kullanıcıların analizlerini görüntüleyebilecekleri bir ön uç arayüzünden ve GitHub verilerini getiren, işleyen ve depolayan bir arka uçtan oluşacaktır. Temel bileşenler şunlardır:

- **Veri Alma Modülü** : Kullanıcı verilerini almak için GitHub'ın REST ve GraphQL API'leriyle arayüz oluşturur.
- **Veri Analiz Motoru** : Kodlama alışkanlıkları ve trendleri hakkında anlamlı içgörüler oluşturmak için ham verileri işler.
- **Veritabanı** : Kullanıcı verilerini ve analiz sonuçlarını verimli sorgulama ve trend takibi için PostgreSQL'de depolar.
- **Önyüz Kullanıcı Arayüzü** : Analizleri görüntülemek için etkileşimli ve kullanıcı dostu bir deneyim sağlamak amacıyla React ile oluşturuldu.

3. İşlevsel Gereksinimler

3.1 Veri Toplama ve Depolama

- **3.1.1 GitHub API Entegrasyonu** :
    - Sistemin kullanıcı verilerini (depolar, kullanılan diller, commit geçmişi) GitHub REST ve GraphQL API'lerinden alması gerekiyor.
    - **Girdiler** : GitHub kullanıcı adı, kimlik doğrulama belirteci.
    - **Çıktılar** : Kullanıcı depolarını, dil ayrıntılarını ve onaylama geçmişini içeren JSON nesneleri.
    - **Sıklık** : Kullanıcı verileri için günlük güncellemeler; yeni istekler için gerçek zamanlı getirme.
- **3.1.2 PostgreSQL'de Veri Depolama** :
    - Sistem kullanıcı profillerini, veri havuzunu, dil kullanımını ve trend bilgilerini depolamalıdır.
    - Verilerin verimli sorgulama için indekslenmesi ve tarih aralığına, dile ve depoya göre filtrelemeye izin vermesi gerekir.

3.2 Kullanıcı Profili Analizi

- **3.2.1 Dil ve Çerçeve Kullanımı** :
    - Sistem, her kullanıcının veri havuzunu analiz ederek en sık kullanılan dilleri ve çerçeveleri belirlemelidir.
    - **Çıktılar** : Baskın diller veya çerçeveler için rozetler, kullanım ölçümleri (örneğin, %40 Python, %35 JavaScript).
- **3.2.2 Trend Değişimleri** :
    - Sistem, kullanıcının dilinde veya çerçeve kullanımında son zamanlarda meydana gelen değişiklikleri tespit edebilmelidir.
    - **Çıktılar** : Her kullanıcı için son kodlama eğilimlerinin analizi (örneğin, son iki ayda TypeScript kullanımının artması).
- **3.2.3 Rozet Atama** :
    - Rozetleri kullanıcının birincil dillerine veya son geçişlerine göre atayın (örneğin, "Python Uzmanı", "JavaScript'te Yeni").

3.3 Küresel Dil ve Çerçeve Analizi

- **3.3.1 Dil Popülarite Sıralaması** :
    - Saklanan verileri analiz ederek en iyi programlama dillerini ve popülerlik sıralamalarını görüntüleyin.
    - **Çıktılar** : Gerçek zamanlı popülerlik sıralaması (örneğin, “JavaScript kullanıcılar arasında #1”).
- **3.3.2 Yaygın Olarak Kullanılan Çerçeveler** :
    - Her dil için en sık kullanılan çerçeveleri belirleyin.
    - **Çıktılar** : Popüler framework listeleri (örneğin, React JavaScript ile en çok kullanılanıdır).

3.4 Raporlama ve İçgörüler

- **3.4.1 Kullanıcı Raporları** :
    - Kullanıcının kodlama etkinliğini, dil kullanımını ve son değişikliklerini özetleyen kişiselleştirilmiş raporlar oluşturun.
- **3.4.2 Topluluk Eğilimleri** :
    - Topluluk eğilimlerini (örneğin, son 3 ayda en popüler diller) gösteren raporlar oluşturun.
- **3.4.3 Görsel Sunum** :
    - Yorumlamayı kolaylaştırmak için verileri çizelgeler, grafikler ve tablolar halinde gösterin.

4. İşlevsel Olmayan Gereksinimler

4.1 Performans

- Sistemin kullanıcı sorgularına 2 saniye içerisinde cevap vermesi gerekiyor.
- Veri analiz işlemleri 5 saniyeyi geçmemelidir.

4.2 Ölçeklenebilirlik

- Arka uç, aynı anda birden fazla API isteğini karşılayabilmeli ve en az 10.000 kullanıcıyı desteklemelidir.

4.3 Güvenlik

- **API Kimlik Doğrulaması** : Kullanıcı kimlik doğrulama belirteçleriyle güvenli GitHub API çağrıları.
- **Veri Gizliliği** : Kullanıcı verilerini şifrelemeyle koruyun ve veri gizliliği düzenlemelerine uyun.

4.4 Kullanılabilirlik

- Kullanıcı arayüzü sezgisel olmalı, kişisel ve topluluk eğilimlerini görüntülemek için net bir gezinme olanağı sağlamalıdır.

4.5 Bakım kolaylığı

- Kolay güncelleme veya özellik eklemeyi kolaylaştırmak için kod tabanı modüler olmalıdır.

5. Hedef Kitle

- **GitHub Kullanıcıları** : Kodlama alışkanlıklarını analiz etmekle ilgileniyorlar.
- **İşe alım uzmanları** : Teknoloji trendlerini ve geliştirici becerilerini değerlendirmek.
- **Eğitimciler ve Öğrenciler** : Popüler dilleri ve çerçeveleri incelemek.

6. Kısıtlamalar

- **API Oran Sınırlaması** : GitHub API'sinin oran sınırlarına uyun.
- **Veri Güncelliği** : Güncel eğilimleri yansıtacak şekilde verilerin düzenli olarak güncellendiğinden emin olun.