**HOMEWORK**

## **1-Dart’ta “assert” ifadesi nedir ve ne zaman kullanılır?**

“assert” operatörü ,Dart dilinde bir koşulun doğru olduğunu test etmek için kullanılır. Eğer koşul doğru değilse , bir “AssertionError” atar ve kodun çalışması durur. “assert” operatörü , genellikle test ve debug amaçlı kullanılır ve çalışma zamanında kapatılabilir.

# **2-Dart’ta “async” ve “await” anahtar kelimelerinin kullanımı hakkında ne biliyorsunuz?**

Programda belirli bir fonksiyonun işini bitirmesini bekle demek istediğimizde await kullandığımız her yerde async kullanmak zorundayız aksi halde hata verir.

# **3-Dart’ta “extension” lar nedir ve nasıl kullanılır?**

Dart dilinin büyük zenginliklerden biri ,extension’ların sağladığı esnekliktir. Extension’lar, mevcut sınıflara veya türlere istediğimiz ek özellikleri eklememize olanak tanır.

# **4-API nedir yazılım geliştirmede hangi amaçlarla kullanılır?**

* Uygulama programlama arayüzü, bir yazılımın başka bir yazılımda tanımlanmış işlevlerini kullanabilmesi için oluşturulmuş bir tanım bütünüdür.
* Backend ve frontend’i birbirine bağlayan kısımdır.
* Farklı uygulamalar arasında veri iletişimini ve paylaşımını sağlayan bilgi işlem bütünüdür.

**Yazılım geliştirmede hangi amaçlarla kullanılır;**

* API güvenlik duvarı oluşturur.
* Tüm verileri ortak bir sunucu ile paylaşmak yerine yalnızca API ile sunucuya gönderildiği için verileri güvenli bir şekilde iletir.
* Pek çok iş API sayesinde hızlı ve kolay halledilebilir.
* Program geliştirmeyi kolaylaştırır.
* Tekrardan uygulama geliştirmeye gerek kalmadığı için zamandan ve paradan tasarruf sağlar. Aynı zamanda daha az bakım maliyetide vardır.
* Web sitesi ve mobil uygulamaların trafiğini ve görünürlüğü arttırır.

# **5-Yazılım geliştirme sürecinde API’ler hangi rolleri üstlenir?**

* Kimlik bilgileri kullanan uygulamalarda oturum açma izni verildiğinde,
* Sunuculardan kişilere ait verileri çekerken,
* E-ticaret işlemlerinde ödeme yaparken banka ile iletişime geçildiğinde,
* Hava durumu uygulamalarında,
* Anlık borsa verilerinin takip edildiği uygulamalarda,
* Uygulamalardaki yazılım güncellemelerinin dağıtılmasında,
* Konum verilerinin sağlanmasında,
* Genel olarak API uygulama yazarken herhangi bir yerden veri çekeceğimiz zaman kullanılır.

1. Entegrasyon

API’ler ,yeni uygulamaların mevcut yazılım sistemlerine entegre edilmesi için kullanılır.Her bir işlevi sıfırdan yazmaya gerek duyulmadığından geliştirme hızı artar . Mevcut kodlardan yararlanmaya devam etmek için API’leri kullanabilirsiniz.

1. İnovasyon

Yeni bir uygulamanın gelişi, bütün bir sektörü değişime zorlayabilir. İşletmelerin hızlı yanıt vermesi ve yenilikçi hizmetlerini hızlı dağıtımını desteklemesi gerekir. Bunu , tüm kodu yeniden yazmak yerine API düzeyinde değişiklikler yaparak gerçekleştirebilirler.

1. Genişleme

API’ler, müşterilerinin ihtiyaçlarını birden fazla platformda karşılamak isteyen işletmelere benzersiz bir fırsat sunar. Örneğin, haritalar API’si harita bilgileri entegrasyonunun web siteleri, Android, iOS vb. aracılığıyla yapabilmesini sağlar. Her işletme, ücretsiz veya ücretli API’leri kullanarak kendi dahili veritabanlarına benzer şekilde erişim verebilir.

1. Bakım kolaylığı

API, iki sistem arasında bir ağ geçidi vazifesi görür. Her bir sistem, API’nin etkilenmemesi için dahili değişiklikler yapma gereği duyar. Bu sayede, taraflardan birinin gelecekte yapacağı kod değişiklikleri, değer tarafı etkilemez.

# **6-RESTful API ve GraphQL gibi farklı farklı API türleri arasındaki farklar nelerdir?**

**REST API’lerinin ana fikirleri aşağıdakileri içerir:**

* İstekler arasında sunucu tarafından geçici durum tutulmaz (dolayısıyla durum aktarımı).
* Uç noktalar açıktır.
* HEAD, GET, POST, PUT, DELETE ve PATCH gibi http “fiilleri” verilerie ne olduğunu açıklar.

**REST’in Avantajları**

* Herşey URL’de,
* Standart tabanlıdır,
* Önbelleğe alma,

**REST’in Dezavantajları**

* Aşırı veya yetersiz veri getirme**,**
* Sürüm oluşturma,

**GraphQL Nedir?**

Yeni başlayanlar için, RESTful API’ler ihtiyaç duyulandan daha fazla veri döndürüyordu, bu da daha fazla API çağrısı anlamına geliyordu. Her uç nokta yalnızca belirtilen verileri döndürebildiğinden, geliştiriciler uç noktaları ön uç görünümlerini akılda tutarak tasarlamak zorundaydı.

Tüm bunlar daha az esneklikle sonuçlandı ve Facebook mühendisleri REST’e bir alternatif oluşturmaya karar verdi. GraphQL, 2012 yılında Facebook tarafından oluşturuldu ve 2015 yılında açık kaynak haline getirildi.

GraphQL, bir sunucu ile nasıl iletişim kurabileceğinizi ve bir istemciden bir sunucuya nasıl veri aktarabileceğinizi açıklar. GraphQL, istemcinin hangi verilere ihtiyaç duyduğunu özellikle sorabilmesi nedeniyle REST’ten temel olarak farklıdır.

GraphQL, birden fazla kaynaktan veri almayı ve toplamayı kolaylaştırır. Birden fazla uç nokta yerine, karmaşık sorguları toplayabilen ve çıktıyı istemcinin istediği formatta sunabilen “akıllı” bir uç noktanız vardı.

Örneğin, kafenizden yemek sipariş etmeniz, marketten alışveriş yapmanız ve bir kütüphaneden kitap ödünç almanız gerektiğini düşünün. REST ile, üç istek olan üç farklı telefon görüşmesi yaparsınız.

GraphQL ile bu üç telefon görüşmesi, Siri ile konuşmak gibi tek bir görüşme ile değiştirilebilir. Tek yapmanız gereken bu yerlerin adresini söylemek ve ardından siparişinizi vermek.

**GraphQL’in Avantajları**

* Esneklik
* Tek istek
* Evrimleşme

**GraphQL’in Dezavantajları**

* Karmaşıklı
* Önbelleğe alma

# **7-Bir web API’si kullanarak nasıl veri alınır ve gönderilir?**

* Başlarken ilk yapmamız gereken pub.dev’den http paketini kopyalamak .
* Eğer çektiğimiz http paketinde kod sayfasında httpler hata verir ise import ettiğimiz yerde yazdığımız paketin sonuna as http yazarsak bu sorun ortadan kalkar.
* Bu kopyaladığımız paketi pubspec.yaml dosyasına ekleriz.
* Daha sonra veriyi yönetebilmek için .dartlı bir dosyası açılır.
* Sonrasında veriyi çekeceğimiz URL’I response değişkeni üzerinden veriyi çekip gelen verinin “response.body” ile decode etmek ve son olarak return de bunu json olarak servis ediyoruz.
* Veriyi tarayıcından çektiğimiz için tarayıcıya json yiklememiz gereklidir ki veriyi bu şekilde görüp kodlamaları o jsona göre çağırmamız gerekiyor yoksa veri çekerken hata alırız.
* Bu işemi ücretsiz apı bulabilevceğimiz sitelerdende yapabiliriz.
* Çekeceğimiz veriler için model oluşturmamız gerekir.Modelide internetten veri çekeceğimiz urlden yardım alarak yaparız.

# **8-API belgeleri neden önemlidir ve geliştiriciler için ne gibi bilgiler sağlar?**

API belgeleri, bir API’nin nasıl etkili bir şekilde kullanılacağına ve API ile entegre edileceğine ilişkin talimatlar içeren, teknik bir içerik sunumudur.İşlevler, sınıflar, dönüş türleri, argümanlar ve daha fazlası hakkında ayrıntılar içeren, API ile çalışmak için gereken tüm bilgileri içeren bir tür kılavuzudur.

Neden Yazıyoruz?

* API’mızı kullanan geliştiricilerin deneyimini geliştirir.
* Yeni kullanıcıların katılımı için harcanan süreyi azaltır. Bu sayede yeni kullanıcılar daha erken sürede üretken olurlar.
* İyi bir bakıma ve daha hızlı güncellemelere sebep olurlar.
* API’yi daha iyi anlamaya ve neler yapacağını bilmeye sebep olur.
* Tüm ekip üyelerinin API’nin ne olduğunu ve API kaynaklarının nasıl ortaya çıktığı konusunda uyum içerisinde olmasını sağlayan merkezi bir referans görevi görebilir.
* Farklı taraflardaki (ön uç/arka uç /mobil geliştirme) geliştirme engellerini kaldırır.
* Ekiple birlikte tanımlarken API mimarisinin hatalarını ve sorunlarının belirlenmesine olanak tanır.
* API’nin nasıl çalıştığını anlamak için ve onu kullanırken beklenmeyen hataların çözmek için harcanan süreyi kısaltır.

# **9-Bir yazılım projesinde harici bir API kullanırken nelere dikkat etmek gerekir?**

Harici API’leri kullanmak , dahili olarak ek işlevsellik sağlamanın en iyi yollarından biridir.

* Harici API’ler size kaynak koduna erişim sunmaz;bu nedenle ,nasıl çalıştığını anlamamıza olanak tanıyan iyi belgelere sahip olmak önemlidir.
* Kod kalitesinin yüksek olduğunu bilmeniz için iyi bir API’nin tutarlı olması gerekir. Bu ,tüm uç noktaların ve niteliklerin anlamlı olduğuna emin olmak, tüm çıktıların aynı şekilde biçimlendirildiğini kontrol etmek gibi şeyleri içerir.
* Kaliteli teknik desteğe sahip olmak şarttır.
* Verilerinizin nasıl kullanılacağını anlamak için her zaman veri gizliliği politikasına göz atın.
* API fiyatlandırmasını karşılaştırırken iyi düşünün bazı ucuz yollu API’ler daha yüksek maliyetlere sebep olabilir.

# **10-Bir mobil uygulama için API kullanımı nasıl optimize edilir?**

* Mobil ihtiyaçları karşılamak için benzer parametreleri birleştirin.
* Veri dönüş boyutunu azaltmak için ayrıntılı seçenekler sunun.
* Minimum veri alt kümelerini sağlayın.
* Verileri gZIP kullanarak sıkıştırın.
* Gönderilen verileri yalnızca görüntülenenlerle sınırlandırın.
* Mobil bir cephe yapılandırın.

# **11-API versiyonlaması neden önemlidir?**

API'nin özel veya herkese açık olmasına bakılmaksızın, API'nin üreticileri ve tüketicilerinin API geliştikçe senkronize kalması çok önemlidir. Etkili bir API sürüm oluşturma stratejisi, yalnızca API üreticilerinin son değişikliklerin tüketiciye yönelik etkilerini en aza indirecek şekilde yineleme yapmalarına olanak sağlamakla kalmaz, aynı zamanda bu değişiklikleri tüketicilere etkili bir şekilde iletmek için bir çerçeve de sağlar. Bu şeffaflık güven oluşturur ve genel API'ler söz konusu olduğunda kuruluşun itibarını güçlendirir; bu da API'nin benimsenme ve elde tutulma oranlarını artırabilir.

# **12-Oauth ve JWT gibi kimlik doğrulama yöntemleri API’lerde nasıl kullanılır?**

**Neden Önemlidir?**

* Veri güvenliği
* Kullanıcı gizliliği
* Kaynak koruması
* Uyumluluk
* API güvenliğini iyileştirme

**JWT**

JWT'ler (JSON Web Belirteçleri), taraflar arasında aktarılacak talepleri temsil etmenin kompakt ve URL açısından güvenli bir yoludur. JWT'ler noktalarla ayrılmış üç bölümden oluşur: başlık, veri ve imza. Başlık, jetonu imzalamak için kullanılan algoritmayı belirtir, yük, talepleri içerir ve imza, jetonun bütünlüğünü doğrulamak için kullanılır.

JWT'ler genellikle web uygulamalarında kimlik doğrulama ve yetkilendirme için kullanılır. Bir kullanıcı oturum açtığında sunucu, kullanıcı kimliği ve izinler gibi kullanıcı hakkındaki bilgileri içeren bir JWT oluşturur. JWT daha sonra yalnızca sunucunun bildiği gizli bir anahtar kullanılarak imzalanır. Sunucu, JWT'yi istemciye gönderir ve istemci daha sonra bunu sunucudaki korunan kaynaklara erişmek için kullanabilir.

İstemci sunucuya bir istek gönderdiğinde, JWT'yi yetkilendirme başlığına dahil eder. Sunucu, gizli anahtarı kullanarak JWT'nin imzasını doğrular ve imza geçerliyse talepleri çıkarır ve bunları isteği yetkilendirmek için kullanır.

JWT'ler, istemci ile sunucu arasında kimlik doğrulama ve yetkilendirme verilerinin durum bilgisi olmadan iletilmesine izin verdikleri için REST API'lerinde yaygın olarak kullanılır. Ayrıca farklı hizmetler ve sistemler arasında kolayca paylaşılabildiklerinden taşınabilirdirler.

**Oauth**

OAuth 2.0, kullanıcıların veya uygulamaların, API'nin kullanıcı adı ve şifre gibi kimlik bilgilerine erişmesine izin vermeden bir API'den kaynaklara erişmesine olanak tanıyan bir yetkilendirme çerçevesidir. Kullanıcının kimlik bilgilerini üçüncü tarafa açıklamasına gerek kalmadan, üçüncü taraf uygulamalarının kullanıcı adına kaynaklara erişmesine izin vermek için yaygın olarak kullanılır.

OAuth 2.0, kaynak sahibi, istemci, yetkilendirme sunucusu ve kaynak sunucusu dahil olmak üzere çeşitli roller içerir. İstemci, yetkilendirme sunucusu tarafından kimliği doğrulanan kaynak sunucusundaki kaynaklara erişmek için kaynak sahibinden yetki ister. Yetki verilirse istemci, kaynaklara erişmek için kullanabileceği bir erişim belirteci alır.

OAuth 2.0, kaynaklara erişim izni vermenin güvenli ve standartlaştırılmış bir yolunu sağlamak için sosyal medya platformları, bulut hizmet sağlayıcıları ve diğer web uygulamaları tarafından yaygın olarak kullanılmaktadır. Belirli bir uygulamaya veya kullanıcıya erişimi iptal etme yeteneği, belirli kaynaklara sınırlı erişim verme yeteneği ve kimlik doğrulamayı güvenilir bir üçüncü tarafa devretme yeteneği de dahil olmak üzere, geleneksel kimlik doğrulama şemalarına göre çeşitli güvenlik avantajları sunar.

# **13-API’lerin güvenliği nasıl sağlanır ve yaygın güvenlik tehditleri nelerdir?**

**API’leri daha güvenli hale getirmek çeşitli stratejiler içerir:**

* Güçlü kimlik doğrulama ve yetkilendirme protokolleri uygulayın.
* Aktarım sırasında verileri korumak için şifreleme kullanın.
* Potansiyel saldırı vektörlerini azaltmak için API uç noktası maruziyetini sınırlayın.
* Düzenli güvenlik denetimleri ve güvenlik açığı değerlendirmelerini yapın.
* Varsayılan olarak hiçbir isteğe güvenmeyerek sıfır güven modelini izleyin.

**Yaygın Güvenlik Tehditler;**

* Zayıf şifrelemeden kaynaklanan veri ihlaleri.
* Zayıf kimlik doğrulama önlemleri nedeniyle yetkisiz erişim.
* Aşırı izinlere yol açan zayıf erişim kontrolleri.
* Veri aktarımı sırasında şifreleme eksikliği.
* Sistem açıklarına yol açan yetersiz uç nokta koruması.