

Web Tasarımı ve Programlama

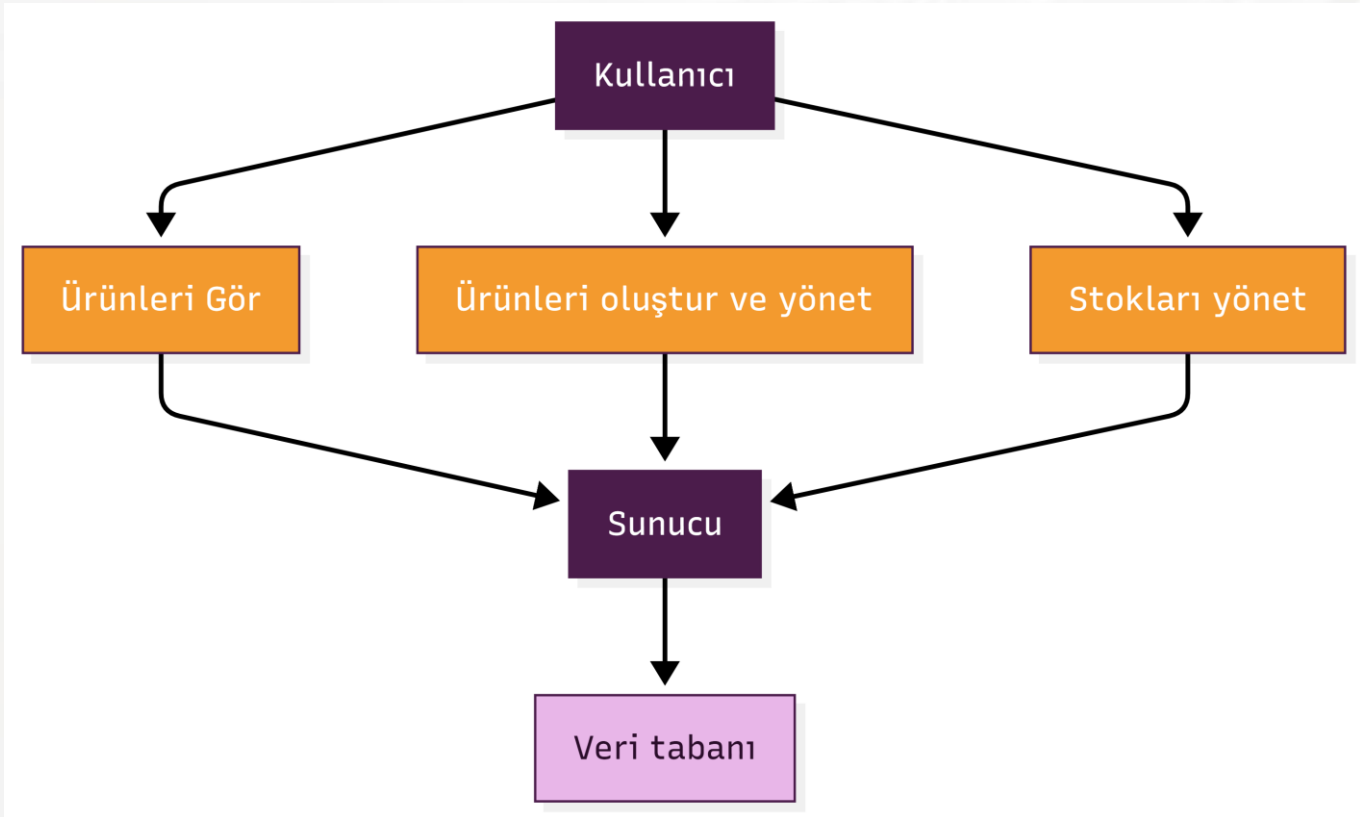
Ders 11: Kimlik Doğrulama (Authentication)

Dr. Öğr. Üyesi Elvan DUMAN

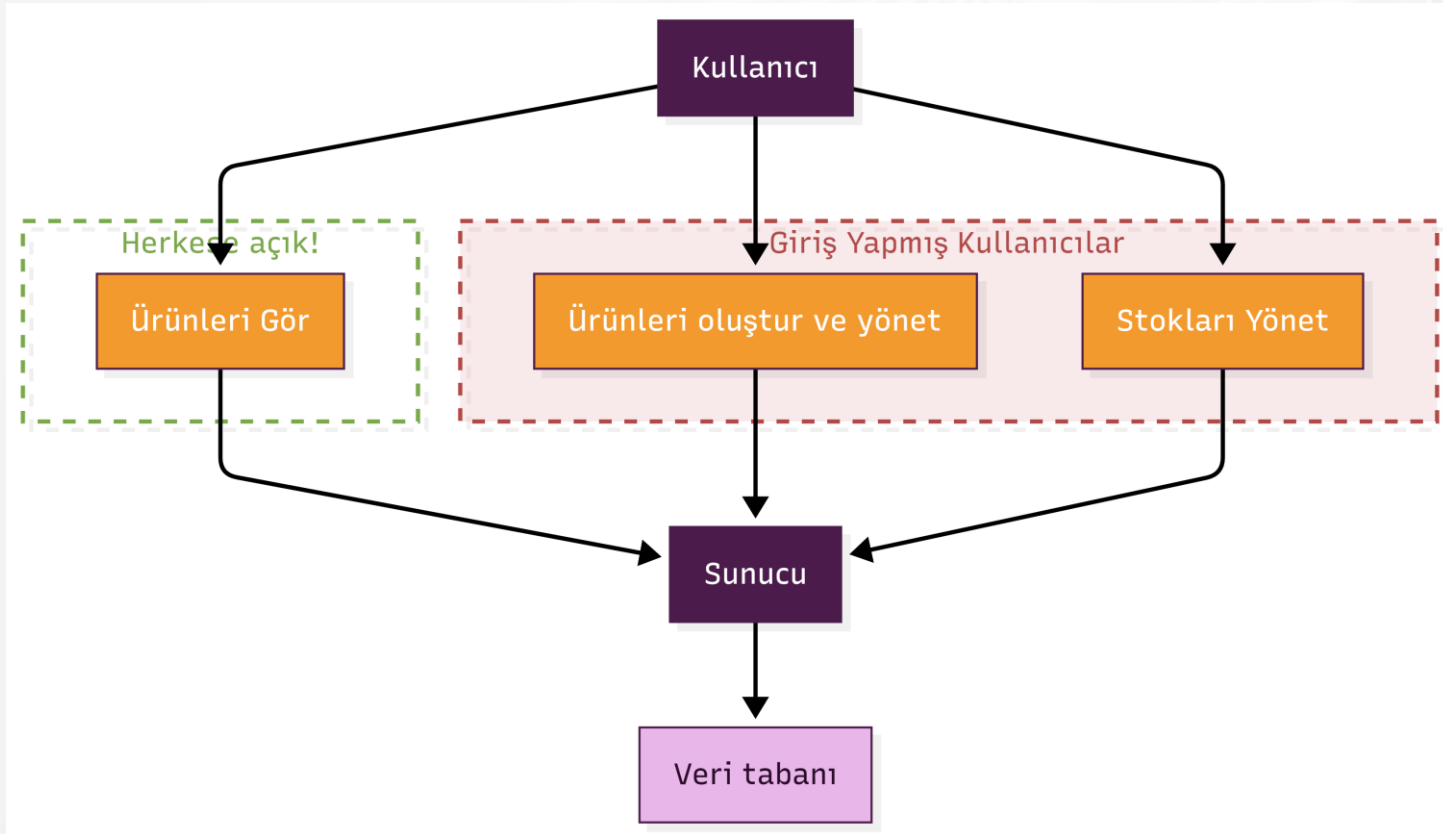
eduman@mehmetakif.edu.tr
Yazılım Mühendisliği Bölümü

- Uygulamada farklı işlemler (routes/actions) bulunur.
- Bu işlemlerin tamamı her kullanıcıya açık olmamalıdır.
- **Kimlik Doğrulama:**
 - Giriş yapmamış (public / anonymous) kullanıcılar ve giriş yapmış kullanıcılar arasındaki farkları belirler.
- Ürün görüntüleme gibi bazı işlemler herkese açıkken ürün oluşturma, yönetme gibi işlemler sadece giriş yapmış belirli kullanıcılara özgü olabilir.
- **Veri doğrulama sayesinde:**
 - Kullanıcı ile yapılan işlemler ilişkilendirilir.
 - Kullanıcının giriş durumu session üzerinden takip edilir.

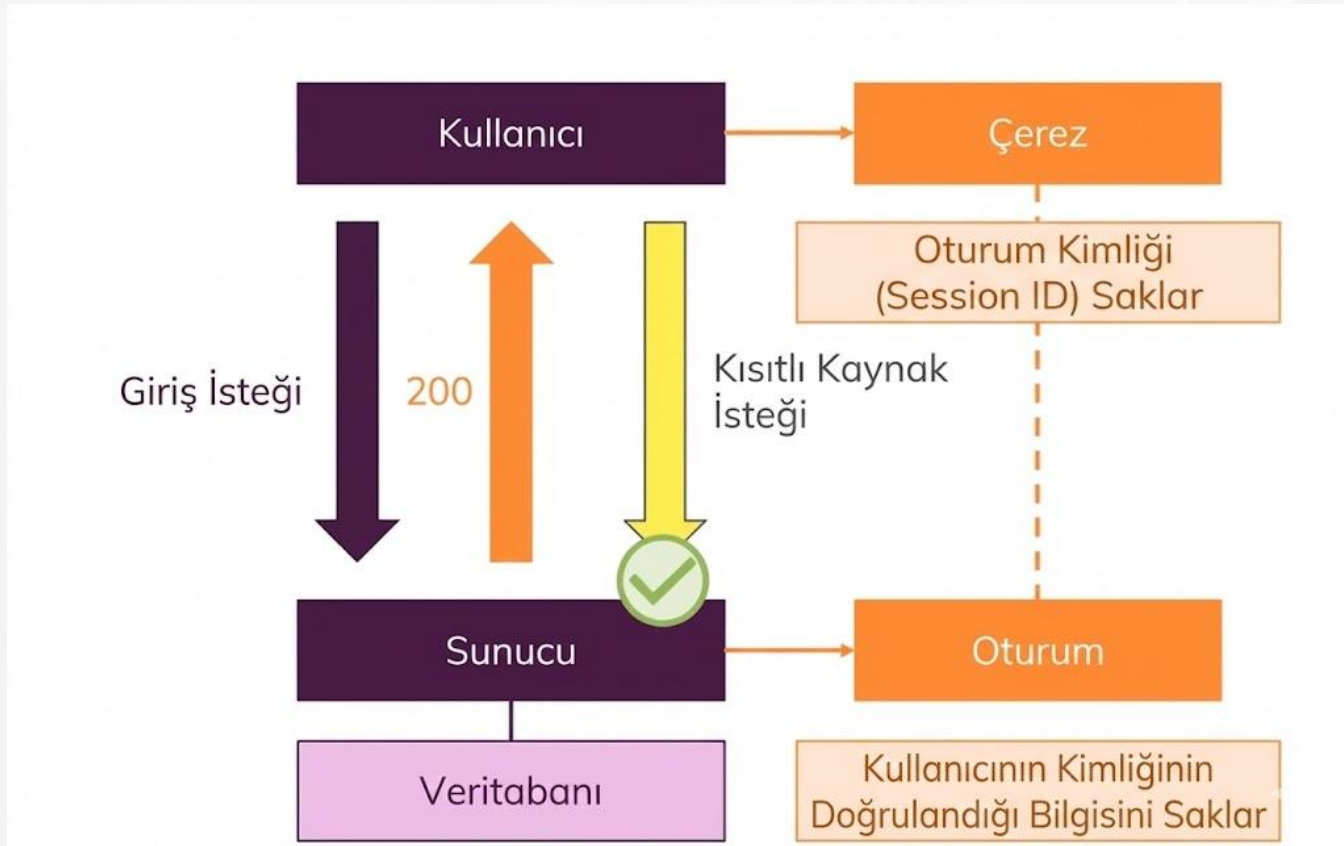
- Kullanıcı, uygulamadaki tüm işlemlere aynı şekilde erişebilir. Sunucu ve veritabanı tarafında, işlemler arasında kullanıcıya bağlı bir erişim ayrımı yapılmaz.



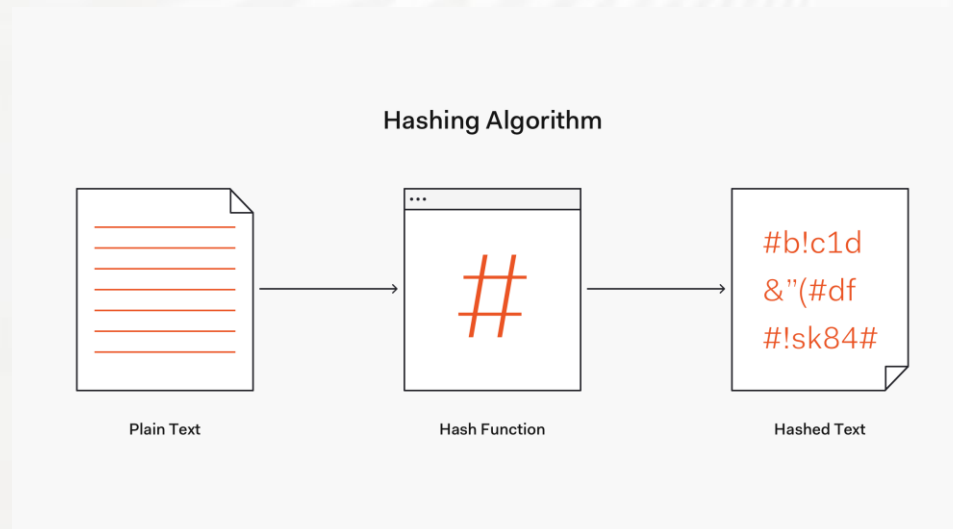
- Authentication ile birlikte, yalnızca giriř yapmıř kullanıcılar belirli iřlemleri gerekleřtirebilir. Herkese aık iřlemler ile korumalı iřlemler sunucu tarafında net biimde ayrılmıřtır.



- Kullanıcı, daha önce kayıt olduğu **eposta ve şifre** ile giriş isteği gönderir.
- Sunucu kullanıcının bilgilerini veri tabanında doğrular.
- Bilgiler geçerliyse başarılı giriş yanıtı ve oturuma ait çerez istemciye yollanır.
- Sonraki isteklerde sunucu cookie üzerinden oturumu bulur.
- Kullanıcının giriş yapmış olup olmadığı kontrol edilir.



- Şifrelerin veritabanında düz metin (plain text) olarak saklanması ciddi bir güvenlik açığıdır. Yetkisiz erişim veya yetkili kişi erişiminde şifreler açığa çıkar.
- Bu nedenle şifreler **okunabilir veya geri döndürülebilir biçimde saklanmamalıdır.**
- Doğru yaklaşım, şifrelerin **geri döndürülemez bir hash algoritması** ile saklanmasıdır.
- Hashleme işlemi sonucunda oluşan değer, gerçek şifrenin kendisi değildir. Hash değerinden orijinal şifre geri elde edilemez.
- Bu yaklaşım, veritabanı ele geçirilse bile **kullanıcı şifrelerinin korunmasını sağlar. Zorunlu bir gereksinimdir.**

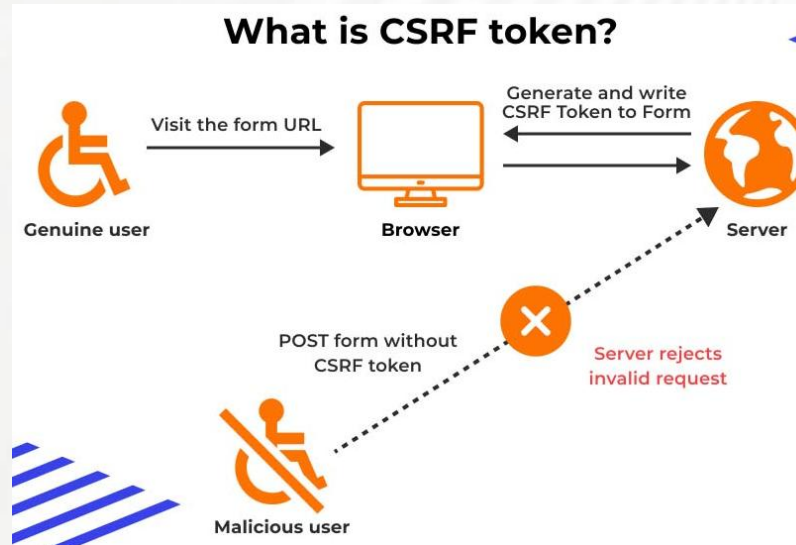


- Şifre, sunucu tarafından özel bir işlemde geçirilerek okunamaz ve anlamsız görünen farklı bir değere dönüştürülür. Bu dönüştürme işlemine hashleme adı verilir ve amaç şifreyi gizlemek değil geri döndürülemez hale getirmektir.
- Hashleme sonucunda ortaya çıkan değer gerçek şifre değildir ve bu değere bakılarak kullanıcının ne yazdığı anlaşılamaz. Hashlenmiş bir değerden orijinal şifreyi tekrar elde etmek teorikte mümkün değildir
- Uygulama, kullanıcıdan gelen şifreyi değil bu şifrenin hashlenmiş halini veritabanında saklar. Böylece veritabanına yetkisiz erişim sağlansa bile kullanıcıların gerçek şifreleri korunmuş olur.
- Modern uygulamalarda **bcrypt** bu amaçla yaygın olarak kullanılır
`npm install --save bcryptjs`
- crypt kullanırken verilen sayı (örneğin 12), hashleme işleminin kaç kez tekrarlandığını (salt rounds) belirler
`bcrypt.hash(sifre, 12); // 8 hızlı ama zayıf, 10 orta, 12 güvenli yaygın`

- Şifre, sunucu tarafından özel bir işlemde geçirilerek okunamaz ve anlamsız görünen farklı bir değere dönüştürülür. Bu dönüştürme işlemine hashleme adı verilir ve amaç şifreyi gizlemek değil geri döndürülemez hale getirmektir.
- Hashleme sonucunda ortaya çıkan değer gerçek şifre değildir ve bu değere bakılarak kullanıcının ne yazdığı anlaşılamaz. Hashlenmiş bir değerden orijinal şifreyi tekrar elde etmek teorikte mümkün değildir
- Uygulama, kullanıcıdan gelen şifreyi değil bu şifrenin hashlenmiş halini veritabanında saklar. Böylece veritabanına yetkisiz erişim sağlansa bile kullanıcıların gerçek şifreleri korunmuş olur.
- Modern uygulamalarda **bcrypt** bu amaçla yaygın olarak kullanılır
`npm install --save bcryptjs`
- crypt kullanırken verilen sayı (örneğin 12), hashleme işleminin kaç kez tekrarlandığını (salt rounds) belirler
`bcrypt.hash(sifre, 12); // 8 hızlı ama zayıf, 10 orta, 12 güvenli yaygın`

- CSRF, **kullanıcının aktif oturumunu kötüye kullanmaya** dayanan bir saldırı türüdür.
- Kullanıcı uygulamada giriş yapmış durumdadır ve geçerli bir session + cookie'ye sahiptir.
- Saldırı senaryosunda kullanıcı **sahte bir siteye yönlendirilir**. Bu sahte site, kullanıcının fark etmediği şekilde **gerçek sunucunuza istek gönderir**.
- İstek, kullanıcının geçerli session'ı ile gönderildiği için sunucu tarafından **meşru kabul edilir**.
- Sonuç olarak kullanıcı:
 - Farkında olmadan sipariş verebilir
 - Para transferi yapabilir.
 - İstenmeyen işlemleri gerçekleştirmiş olur

- CSRF saldırısı çalışır çünkü, tarayıcı, session cookie'yi **otomatik olarak her isteğe ekler**. Sunucu, isteğin gerçekten kendi sayfasından mı geldiğini ayırt edemez.
- Bu yüzden sahte bir sayfadan gelen istek, gerçek kullanıcı isteği gibi görünür.
- Çözüm yaklaşımı: session'ın **sadece uygulamanın kendi sayfaları üzerinden** kullanılmasını sağlamak
- Bunun için her form veya kritik isteğe **CSRF token** eklenir. CSRF token sunucu tarafından üretilir. Her istekte doğrulanır. Sahte sayfalar tarafından üretilemez



Ders 11 Sunum Sonu

İletişim : eduman@mehmetakif.edu.tr

