PHP ve MySQL ile Güvenli ve Ölçeklenebilir bir QR Kod Giriş Sistemi için Mimari Tasarım

Bu rapor, PHP ve MySQL teknolojileri kullanılarak geliştirilecek optimum bir QR kod tabanlı giriş sisteminin mimarisini, kullanılan teknolojileri ve uygulanması gereken güvenlik önlemlerini kapsamlı bir şekilde ele almaktadır. Rapor, teknik liderler ve deneyimli yazılım geliştiriciler için bir başvuru kaynağı niteliğinde olup, modern, performanslı ve güvenli bir sistemin inşası için gereken tüm bileşenleri ayrıntılı olarak açıklamaktadır.

Bölüm 1: Sistem Mimarisi ve Temel Giriş Akışı

Bu bölüm, QR kod ile giriş sürecinin temelini oluşturan mimariyi, bileşenleri, gereksinimleri ve üst düzey tasarımı tanımlamaktadır. Takip eden teknik tartışmalar için bir plan görevi görür.

1.1. QR Kod ile Girişin Anatomisi: Durum Odaklı Bir Süreç

Sistem, üç ana aktör arasındaki etkileşime dayanır:

1. **Web İstemcisi (Güvenen Taraf - Relying Party)**: Kullanıcının giriş sürecini başlattığı masaüstü/dizüstü bilgisayarındaki tarayıcı.
2. **Arka Uç Sunucusu (Backend Server)**: Tüm akışı yöneten PHP/MySQL tabanlı uygulama.
3. **Mobil İstemci (Doğrulayıcı - Authenticator)**: Kullanıcının önceden kimliğini doğruladığı ve QR kodu taramak için kullandığı güvenilir mobil cihazı.

Giriş akışı, dağıtık bir durum makinesi (distributed state machine) olarak ele alınmalıdır.[1] Bu akış, aşağıdaki olaylar dizisiyle gerçekleşir:

1. **Başlatma**: Web İstemcisi, giriş sayfasını talep eder.

2. **QR Talebi**: Web İstemcisinin JavaScript kodu, Arka Uç Sunucusuna bir AJAX çağrısı yapar (GET /qr-code/request).
3. **Oturum Oluşturma**: Arka Uç Sunucusu, kriptografik olarak güvenli, benzersiz bir session_id ve kısa ömürlü, tek kullanımlık bir nonce (veya token) üretir. Bu bilgileri qr_login_sessions tablosuna pending (beklemede) durumu ve bir expires_at (sona erme) zaman damgası ile kaydeder.[1]
4. **QR Görüntüleme**: Sunucu, session_id ve nonce değerlerini döndürür. Web İstemcisi, bu verileri içeren bir QR kodu oluşturmak için bir JavaScript kütüphanesi kullanır. Eş zamanlı olarak, session_id ile kendini tanıtarak Arka Uç Sunucusuna bir WebSocket bağlantısı açar.
5. **Tarama ve İletme**: Kimliği doğrulanmış Mobil İstemci, QR kodunu tarar, session_id ve nonce'ı çıkarır ve bu bilgileri kullanıcının mobil oturumuna ait kimlik doğrulama belirteci (örneğin bir JWT) ile birlikte güvenli bir API çağrısı (POST /login/scan/confirm) aracılığıyla Arka Uç Sunucusuna gönderir.[1]
6. **Doğrulama**: Arka Uç Sunucusu, session_id, nonce ve mobil kullanıcının kimlik doğrulama belirtecini doğrular. Oturumun hala pending durumunda olduğunu ve süresinin dolmadığını kontrol eder. Başarılı doğrulama üzerine qr_login_sessions tablosunu günceller: durum confirmed (onaylandı) olur, user_id oturumla ilişkilendirilir ve nonce yeniden kullanımı önlemek için geçersiz kılınır.
7. **Gerçek Zamanlı Bildirim**: Arka Uç Sunucusu, session_id ile ilişkili WebSocket bağlantısını kullanarak Web İstemcisine "giriş başarılı" mesajı gönderir. Bu mesaj, yeni ve güvenli bir web oturum belirteci (örneğin, bir JWT veya geleneksel oturum çerezi) içerir.
8. **Tamamlama**: Web İstemcisi mesajı alır, oturum belirtecini saklar ve kullanıcıyı kimliği doğrulanmış kontrol paneline yönlendirir. Böylece kullanıcı, o cihazda hiç şifre yazmadan giriş yapmış olur.[3]

1.2. Sistem Gereksinimlerinin Tanımlanması

Fonksiyonel Gereksinimler [1]:

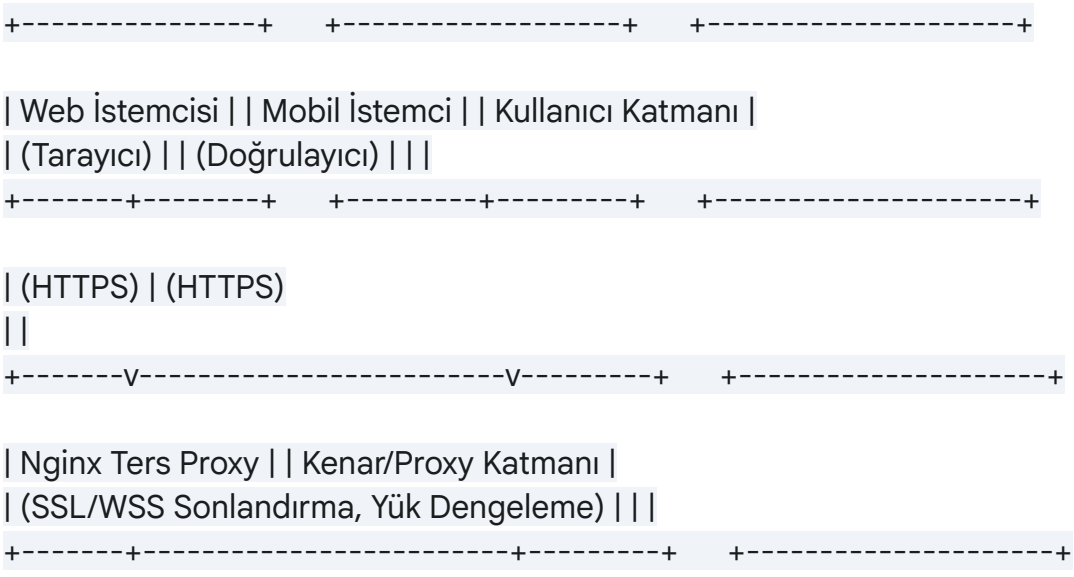
- Kullanıcı, bir web tarayıcısından giriş başlatabilmelidir.
- Sistem, benzersiz ve taranabilir bir QR kodu görüntülemelidir.
- Kullanıcı, kimliği doğrulanmış bir mobil uygulama ile kodu tarayabilmelidir.
- Başarılı tarama ve onay üzerine web oturumu otomatik olarak oluşturulmalıdır.
- QR taraması başarısız olursa, sistem manuel kod girişi veya geleneksel şifre ile giriş gibi net geri dönüş (fallback) seçenekleri sunmalıdır.[5]

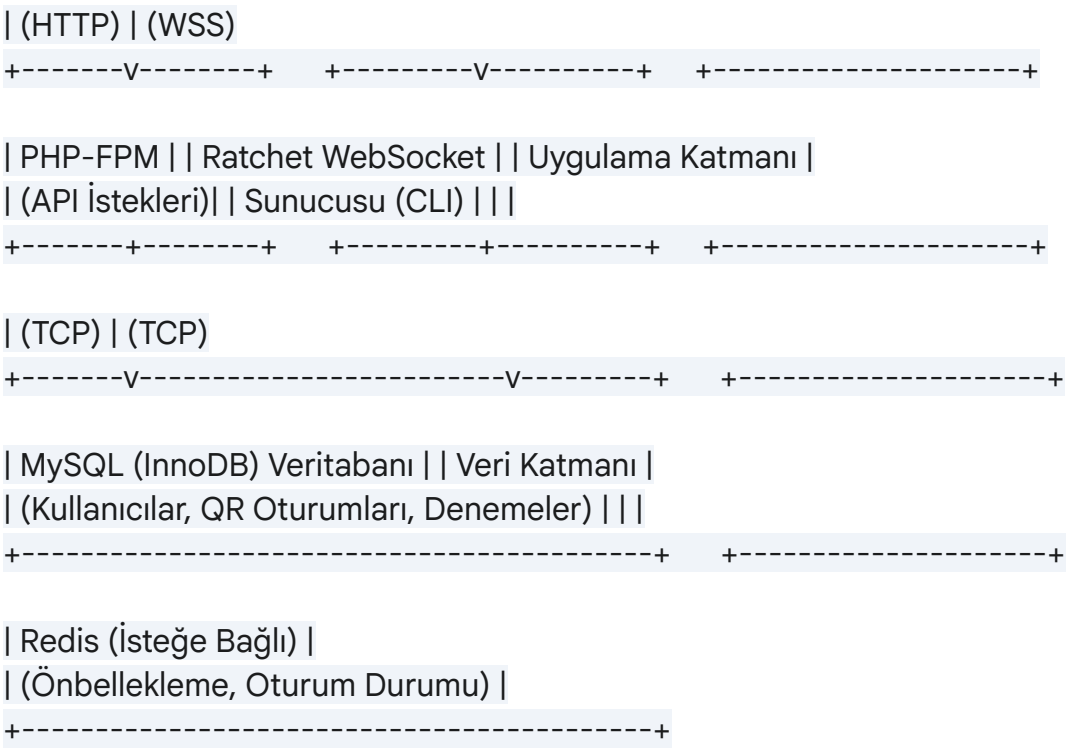
Fonksiyonel Olmayan Gereksinimler [1]:

- **Performans**: Giriş süreci (taramadan web yönlendirmesine kadar) 2 saniyenin altında tamamlanmalıdır. QR kod üretimi anlık olmalıdır.
- **Güvenlik**: Sistem, QRLJacking, yeniden oynatma (replay) saldırıları, oturum ele geçirme ve diğer OWASP Top 10 zafiyetlerine karşı dayanıklı olmalıdır. Tüm iletişim şifrelenmelidir (HTTPS/WSS).
- **Ölçeklenebilirlik**: Mimari, çok sayıda eş zamanlı beklemedeki giriş oturumunu ve WebSocket bağlantısını desteklemelidir. Arka uç hizmetleri mümkün olduğunca durumsuz (stateless) olmalıdır.[1]
- **Güvenilirlik**: Gerçek zamanlı iletişim kanalı, otomatik yeniden bağlanma mekanizmalarıyla sağlam olmalıdır.

1.3. Üst Düzey Mimari Şeması

Aşağıdaki şema, sistem bileşenlerini görselleştirmektedir: Nginx ters proxy, standart istekler için PHP-FPM, PHP WebSocket sunucusu, MySQL veritabanı ve potansiyel bir önbellekleme katmanı olan Redis arasındaki etkileşimi gösterir.





Bu mimarinin temelinde, QR kod ile giriş akışının basit bir kimlik doğrulama mekanizmasından daha fazlası olduğu gerçeği yatar; bu, karmaşık ve dağıtık bir durum yönetimi problemidir. Tüm sistemin güvenliği ve güvenilirliği, qr_login_sessions tablosundaki durumun (pending -> confirmed -> expired) sağlam bir şekilde yönetilmesine bağlıdır. Bir kullanıcının bir cihazda (web) başlattığı bir eylemin başka bir cihazla (mobil) yerine getirilmesi ve ardından ilk cihaza geri yansıtılması gerekliliği, oturumun savunmasız bir pending durumunda olduğu bir zaman aralığı yaratır.[1] Bu durumların doğru bir şekilde yönetilememesi, doğrudan güvenlik açıklarına (yeniden oynatma saldırıları gibi) veya kötü bir kullanıcı deneyimine (eski QR kodları gibi) yol açar. Dolayısıyla, veritabanı şeması (Bölüm 4) ve belirteç geçersizleştirme mantığı (Bölüm 5) sadece uygulama detayları değil, mimari tasarımın merkezindedir.

1.4. API ve WebSocket Sözleşme Tasarımı

REST API Uç Noktaları [1]:

- GET /api/v1/auth/qr/request: Süreci başlatır. Yanıt: $ { "sessionId": "...", "nonce": "...", "expiresIn": 120 } $.
- POST /api/v1/auth/qr/confirm: Mobil uygulama tarafından gönderilir. Gövde: $ {

"sessionId": "...", "nonce": "..." } $. Authorization başlığında mobil kimlik doğrulama belirteci gerektirir. Başarı veya başarısızlık döndürür.

WebSocket İletişimi:

- **Bağlantı URI'si**: wss://alanadiniz.com/ws?sessionId={sessionId}. Oturum ID'si, bağlantıyı bekleyen girişle ilişkilendirmek için kullanılır.
- **Sunucudan İstemciye Mesajlar**: Durumu tanımlayan JSON tabanlı mesajlar.
  - $ { "event": "statusUpdate", "status": "scanned" } $ (İsteğe bağlı, daha iyi UX için).
  - $ { "event": "loginSuccess", "token": "...", "user": {... } } $.
  - $ { "event": "loginFailed", "reason": "expired_qr" | "invalid_scan" } $.

Bölüm 2: Teknoloji Yığını Derinlemesine İnceleme: QR Kod Üretimi

Bu bölümde, hem sunucu hem de istemci tarafında QR kodları oluşturmak için en uygun araçlar değerlendirilmekte ve seçilmektedir.

2.1. Sunucu Taraflı QR Üretimi: Bir PHP Kütüphanesi Seçimi

Bir kütüphane seçimi, sürdürülebilirlik, özellikler ve performansı doğrudan etkiler. Araştırmalara dayanarak en önde gelen üç seçenek karşılaştırılacaktır.[8]

Tablo: PHP QR Kod Kütüphanesi Karşılaştırması

Özellik	endroid/qr-code	chillerlan/php-qrcode	phpqrcode (Eski)
PHP Sürümü	>= 7.4 [8]	>= 8.2 [8]	>= 5.x [8]
Kurulum	Composer [9]	Composer [8]	Manuel/Composer [8]
Temel Özellikler	Akıcı API, logo/etiket gömme, çoklu	Üretim ve okuma, yüksek düzeyde	Saf PHP, GD2 tabanlı, basit PNG/JPEG

	yazıcılar (PNG, SVG, vb.), Symfony entegrasyonu [8]	yapılandırılabilir, birçok çıktı modülü (PDF, EPS), ECI desteği [8]	çıktısı [8]
Bağımlılıklar	Symfony bileşenleri, resim uzantıları [8]	ext-mbstring, isteğe bağlı resim/PDF kütüphaneleri [8]	ext-gd2 [8]
Tavsiye	**Tavsiye Edilir**	Mükemmel Alternatif	Tavsiye Edilmez

Tavsiye Gerekçesi: endroid/qr-code, modern özellikleri (akıcı API, logo desteği), kullanım kolaylığı ve chillerlan/php-qrcode'ın daha katı 8.2+ gereksinimine kıyasla daha geniş PHP sürüm uyumluluğu (7.4+) arasındaki mükemmel denge nedeniyle birincil tavsiye olarak seçilmiştir.[8] Bu, onu daha geniş bir proje yelpazesi için daha erişilebilir kılar. Symfony ve Laravel gibi çerçevelerle güçlü entegrasyonu da önemli bir avantajdır.[8]

endroid/qr-code ile Gelişmiş Kullanım:
Logo, özel renkler, yüksek hata düzeltme seviyesi (logo kodun bir kısmını kapladığında esastır) ve doğrulama özelliklerine sahip bir QR kodu oluşturmak için Builder desenini gösteren ayrıntılı bir kod örneği sunulacaktır.11

PHP

```php
<?php
require 'vendor/autoload.php';

use Endroid\QrCode\Builder\Builder;
use Endroid\QrCode\Encoding\Encoding;
use Endroid\QrCode\ErrorCorrectionLevel;
use Endroid\QrCode\Label\LabelAlignment;
use Endroid\QrCode\Label\Font\NotoSans;
use Endroid\QrCode\RoundBlockSizeMode;
use Endroid\QrCode\Writer\PngWriter;
use Endroid\QrCode\Color\Color;
use Endroid\QrCode\Logo\Logo;
use Endroid\QrCode\Label\Label;
```

```php
$data = '{"sessionId":"abc-123-xyz-789", "nonce":"a1b2c3d4e5f6"}';

$result = Builder::create()
    ->writer(new PngWriter())
    ->writerOptions()
    ->data($data)
    ->encoding(new Encoding('UTF-8'))
    ->errorCorrectionLevel(ErrorCorrectionLevel::High) // Logo için yüksek hata düzeltme
    ->size(400)
    ->margin(10)
    ->roundBlockSizeMode(RoundBlockSizeMode::Margin)
    ->logoPath(__DIR__.'/assets/logo.png')
    ->logoResizeToWidth(100)
    ->logoPunchoutBackground(true) // Logo arkasındaki QR kod modüllerini kaldırır
    ->labelText('Giriş yapmak için tarayın')
    ->labelFont(new NotoSans(20))
    ->labelAlignment(LabelAlignment::Center)
    ->validateResult(true) // Oluşturulan QR kodunun okunabilirliğini doğrula
    ->build();

// Sonucu doğrudan tarayıcıya gönder
header('Content-Type: '.$result->getMimeType());
echo $result->getString();

// Veya bir dosyaya kaydet
// $result->saveToFile(__DIR__.'/qrcode.png');
```

Bir QR kodu kütüphanesi seçimi sadece bir resim oluşturmakla ilgili değildir; üretim hazırlığı ile ilgilidir. endroid/qr-code gibi modern kütüphaneler, yerleşik doğrulama (validateResult) sunar; bu, kritik ancak genellikle göz ardı edilen bir özelliktir.[11] Bir logoyu QR koduna eklemek, taranabilir alanı azaltır ve bu da dağıtıma alınmadan önce taranamaz kodları önlemek için doğrulamayı zorunlu kılar. Bu, üretim için kritik bir risk azaltma adımıdır. Bu nedenle, bir logo gömerken her zaman en yüksek hata düzeltme seviyesini (

ErrorCorrectionLevel::High) kullanmak ve sonucu programatik olarak doğrulamak tavsiye edilir.[15]

2.2. İstemci Taraflı QR Kod Oluşturma

QR kod *verisi* sunucuda oluşturulurken, resmin kendisi istemci tarafında JavaScript kullanılarak dinamik olarak oluşturulmalıdır. Bu, sunucuda resim dosyaları oluşturma ve saklama ihtiyacını ortadan kaldırır.

Tavsiye Edilen Kütüphane: qrcode.js (davidshimjs/qrcodejs), basitliği, bağımlılıklarının olmaması ve HTML5 Canvas veya bir tablo geri dönüşü kullanarak tarayıcılar arası desteği nedeniyle mükemmel bir seçimdir.[16]

Uygulama Akışı:

1. İstemcinin JavaScript'i, sunucudan $ { "sessionId": "...", "nonce": "..." } $ yükünü alır.
2. Bu veriyi tek bir dizeye birleştirir (örneğin, bir JSON dizesi veya özel bir format).
3. Ardından, belirlenmiş bir içinde QR kodunu oluşturmak için new QRCode("element-id", { text: dataString,...options }); çağrısını yapar.[17]
4. QR kodunun yanında, expiresIn değerini görsel olarak temsil eden bir geri sayım sayacı görüntülenmelidir. Bu, kullanıcıyı süresi dolmadan ve yenilenmeden önce taramaya teşvik eder.

Bölüm 3: WebSockets ile Gerçek Zamanlı İletişim Katmanı

Bu bölüm, şifresiz girişin "sihrini" mümkün kılan gerçek zamanlı kanalın uygulanmasını gerekçelendirmekte ve detaylandırmaktadır.

3.1. Protokol Seçimi: WebSockets vs. Long-Polling

Temel sorun, web istemcisinin, mobil uygulama taramayı onayladıktan *hemen sonra* sunucu tarafından bilgilendirilmesi gerekliliğidir. Geleneksel bir istek-yanıt modeli bu senaryo için verimsizdir. İki ana çözüm analiz edilecektir: AJAX Long-Polling ve WebSockets.[20]

Tablo: Gerçek Zamanlı İletişim Protokolü Karşılaştırması

Metrik	WebSockets	AJAX Long-Polling
İletişim	Tam çift yönlü, çift taraflı (Full-duplex) [20]	İstemci tarafından başlatılan, sunucu tarafından tutulan [20]
Bağlantı	Tek, kalıcı TCP bağlantısı [20]	Tekrarlanan HTTP istekleri [20]
Gecikme	Çok düşük [20]	İstek döngüsü nedeniyle daha yüksek [20]
Ek Yük (Overhead)	Düşük (el sıkışmadan sonra minimal başlıklar) [22]	Yüksek (her mesaj için tam HTTP başlıkları) [20]
Sunucu Yükü	Daha düşük, çok sayıda bağlantı için verimli [20]	Daha yüksek, ölçekte kaynak yoğun [20]
Ölçeklenebilirlik	Uygun mimari ile mükemmel [20]	Zayıf, ölçekte maliyetli [20]

Sonuç: WebSockets, modern, ölçeklenebilir ve performanslı bir QR giriş sistemi için şüphesiz üstün bir seçimdir. Düşük gecikme süreleri ve kaynak verimlilikleri, gereken sorunsuz kullanıcı deneyimini sağlamak için kritik öneme sahiptir.[20]

3.2. Ratchet ile PHP WebSocket Sunucusu Uygulaması

Kütüphane Seçimi: Swoole, C eklentisi uygulaması nedeniyle daha yüksek ham performans sunsa da [23],

Ratchet tavsiye edilen kütüphanedir.

Gerekçe: Ratchet, saf bir PHP kütüphanesidir, bu da onu özel PHP eklentileri veya sunucuya kök erişimi gerektirmeden daha geniş bir barındırma ortamı yelpazesinde dağıtmayı çok daha kolay hale getirir. Mimarisi, standart PHP çerçevelerine alışkın geliştiriciler için daha kolay anlaşılırdır.[23] Performans farkı, darboğazın WebSocket işlemesi değil, uygulama mantığının kendisi olduğu gerçek dünya uygulamalarında genellikle ihmal edilebilir düzeydedir.[23]

Sunucunun İnşası:
Aşağıdaki adımlar, QR girişi için uyarlanmış bir sunucu oluşturma sürecini özetlemektedir.25 Sunucu,
Ratchet\MessageComponentInterface arayüzünü uygulayacaktır.

1. **__construct()**: İstemci bağlantılarını tutmak için bir \SplObjectStorage başlatılır.
2. **onOpen(ConnectionInterface $conn)**:
   - Bağlantının sorgu parametrelerinden ($conn->httpRequest->getUri()->getQuery()) sessionId çıkarılır.
   - Bağlantı, sessionId ile anahtarlanmış bir haritada saklanır. Örnek: $this->clients[$sessionId] = $conn;. Bu eşleme, bekleyen bir web girişi ile bir WebSocket bağlantısı arasındaki kritik halkadır.
3. **onMessage(ConnectionInterface $from, $msg)**: Bu yöntem, tetikleyici mobil uygulamanın HTTP POST'undan geldiği için QR giriş akışımızda büyük ölçüde kullanılmayacaktır. Ancak, kalp atışı/ping-pong kontrolleri için kullanılabilir.
4. **onClose(ConnectionInterface $conn)**: Bağlantı, resourceId veya saklanan sessionId ile haritada bulunur ve bellek sızıntılarını önlemek için kaldırılır.
5. **Tetikleme Mantığı**: PHP uygulamasının ayrı bir bölümü (örneğin, POST /login/scan/confirm isteğini işleyen denetleyici), çalışan Ratchet sunucusuyla iletişim kurmalıdır. Mobil tarama doğrulandıktan sonra, hedef bağlantı $this->clients[$sessionId] aracılığıyla bulunur ve $conn->send($message) çağrılır.

3.3. İstemci Taraflı WebSocket Yönetimi: Dayanıklılık için İnşa Etmek

İstemci tarafı JavaScript'in sağlam olması gerekir. Bir bağlantı kopması, kullanıcının giriş akışını kırmamalıdır.

Üstel Geri Çekilme ve Jitter ile Otomatik Yeniden Bağlanma: Bu, sunucunun geçici olarak kullanılamaması durumunda sunucuyu sürekli isteklerle boğmayı önlemek için kritik bir en iyi uygulamadır.[27] Aşağıdaki mantığa sahip bir

ReconnectingWebSocket JavaScript sınıfı sağlanmalıdır [27]:

- onclose olayında, başarısız olmak yerine, setTimeout kullanarak bir yeniden bağlanma denemesi planlanır.
- Zaman aşımı gecikmesi, her başarısız denemede katlanarak artmalıdır (örneğin, 1s, 2s, 4s, 8s...), bir maksimum değerle (örneğin, 30s) sınırlandırılmalıdır. Bu, "üstel geri çekilme" (exponential backoff) olarak bilinir.

- Gecikmeye küçük, rastgele bir "jitter" (örneğin, 0-1s) eklenmelidir. Bu, binlerce istemcinin sunucu genelinde bir kesintiden sonra tam olarak aynı milisaniyede yeniden bağlanmaya çalışmasını önler; bu olgu "thundering herd" problemi olarak bilinir.

PHP'de WebSockets uygulamak, bir PHP uygulamasının dağıtım ve operasyonel modelini temelden değiştirir. Geleneksel olarak durumsuz, kısa ömürlü isteklere dayanan bir ekosisteme, uzun süren, durum bilgisi olan bir süreç ekler. Standart bir PHP-FPM kurulumu her istekten sonra bir betiği sonlandırırken, bir Ratchet sunucusu komut satırından (php server.php) başlatılır ve süresiz olarak çalışır.[25] Bu, web sunucusu (Apache/Nginx) tarafından aynı şekilde yönetilmediği anlamına gelir. Betik çökerse veya sunucu yeniden başlatılırsa, WebSocket sunucusu kaybolur. Bu durum, herhangi bir üretim dağıtımı için

Supervisor gibi bir süreç yöneticisinin "olsa iyi olur" değil, WebSocket sunucusunun her zaman çalışmasını ve başarısızlık durumunda otomatik olarak yeniden başlatılmasını sağlamak için zorunlu bir bileşen olduğu sonucuna götürür.[31]

Bölüm 4: MySQL ile Veritabanı Mimarisi ve Yönetimi

Bu bölüm, sistemin durum yönetiminin kalbi olan MySQL veritabanını detaylandırmaktadır. Güvenli, performanslı ve sürdürülebilir bir şema üzerine odaklanılacaktır.

4.1. Optimize Edilmiş Veritabanı Şema Tasarımı

Sistemi yönetmek için üç temel tablo tanımlanacaktır. Tasarım, kimlik doğrulama sistemleri için en iyi uygulamalara [34] ve QR akışımızın özel ihtiyaçlarına [1] dayanmaktadır.

Tablo: Kapsamlı MySQL Şeması

Tablo	Sütun Adı	Veri Tipi	Nitelikler / Notlar

users	id	BIGINT UNSIGNED	PRIMARY KEY, AUTO_INCREMENT
	uuid	CHAR(36)	UNIQUE, halka açık kimliklendirme için.
	email	VARCHAR(255)	UNIQUE, NOT NULL
	password_hash	VARCHAR(255)	NOT NULL. Şifrelenmiş parolaları saklamak için (geri dönüş girişi).
	created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
	updated_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
qr_login_sessions	id	BIGINT UNSIGNED	PRIMARY KEY, AUTO_INCREMENT
	session_id	VARCHAR(128)	UNIQUE, NOT NULL. Oturum için halka açık ID.
	nonce	VARCHAR(128)	NOT NULL. Tek kullanımlık belirteç.
	status	ENUM('pending', 'scanned', 'confirmed', 'expired')	NOT NULL, DEFAULT 'pending'. Giriş denemesinin durumu.
	user_id	BIGINT UNSIGNED	NULLABLE, FK to users.id. Onaylandığında doldurulur.

	web_socket_id	VARCHAR(255)	NULLABLE. Doğrudan eşleme için Ratchet bağlantı kaynak ID'sini saklar.
	ip_address	VARCHAR(45)	NULLABLE. Başlatan web istemcisinin IP'si.
	user_agent	TEXT	NULLABLE. Başlatan web istemcisinin kullanıcı aracısı.
	expires_at	TIMESTAMP	NOT NULL. Bu QR kodunun geçersiz olacağı zaman.
	created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
login_attempts	id	BIGINT UNSIGNED	PRIMARY KEY, AUTO_INCREMENT
	ip_address	VARBINARY(16)	NOT NULL. IPv4 ve IPv6'yı destekler.
	attempted_at	TIMESTAMP	NOT NULL, DEFAULT CURRENT_TIMESTAMP.

Gerekçe: qr_login_sessions tablosu, durum makinesinin çekirdeğidir. status ENUM, net ve tanımlanmış durumlar sağlar. expires_at, güvenlik ve temizlik için kritiktir. ip_address ve user_agent saklamak, gelişmiş güvenlik kontrollerine olanak tanır. login_attempts tablosu, özellikle oran sınırlaması (rate limiting) uygulamak için tasarlanmıştır.[36]

4.2. Depolama Motoru Analizi: InnoDB vs. MyISAM

MyISAM eski, işlemsel olmayan bir motor iken, InnoDB modern, ACID uyumlu varsayılan

motordur.[37]

qr_login_sessions tablomuz yüksek hacimli INSERT, UPDATE (durum değiştirme) ve DELETE (temizlik) işlemleri görecektir. Bu, yüksek eşzamanlılığa sahip, yazma ağırlıklı bir iş yüküdür.

Karar: **InnoDB tek kabul edilebilir seçimdir.**

- **Satır Düzeyinde Kilitleme**: MyISAM, tablo düzeyinde kilitleme kullanır. Tek bir oturum satırına yapılan bir güncelleme, *tüm tabloyu* kilitleyerek yük altında büyük bir darboğaz yaratır ve sistemi durma noktasına getirir. InnoDB'nin satır düzeyinde kilitlemesi, binlerce eşzamanlı oturum güncellemesini işlemek için esastır.[37]
- **ACID Uyumluluğu ve Çökme Kurtarma**: Giriş, işlemsel bir süreçtir. Kimlik doğrulama sırasında bir sistem çökmesi, bir MyISAM tablosunda bozuk veri bırakabilir. InnoDB'nin işlemsel bütünlüğü ve otomatik çökme kurtarma mekanizmaları, kritik bir kimlik doğrulama sistemi için tartışılamazdır.[38]
- **Yabancı Anahtarlar**: InnoDB, veritabanı düzeyinde veri bütünlüğünü zorlayan yabancı anahtar kısıtlamalarını destekler (örneğin, qr_login_sessions.user_id -> users.id). MyISAM bunu desteklemez.[37]

4.3. Yüksek Performans için İndeksleme Stratejisi

İndeksler, kullanıcı deneyimini düşürecek ve sunucuyu aşırı yükleyecek yavaş sorguları önlemek için hayati önem taşır.[41]

qr_login_sessions Tablosu İndeksleme Stratejisi:

- PRIMARY KEY (id): Örtük olarak oluşturulur.
- UNIQUE INDEX idx_session_id (session_id): Mobil uygulama taramayı onayladığında hızlı aramalar için esastır.
- **Bileşik İndeks**: INDEX idx_status_expires (status, expires_at): Bu en önemli performans optimizasyonudur. Temizlik süreci sık sık WHERE status = 'pending' AND expires_at < NOW() sorgusu yapacaktır. Bu iki sütun üzerindeki bir bileşik indeks, MySQL'in tam tablo taraması yapmadan süresi dolmuş oturumları çok hızlı bir şekilde tanımlamasını ve işlemesini sağlar.[43] Sıra (
status önce) önemlidir çünkü status daha düşük kardinaliteye sahiptir ve ilk filtre olacaktır.

login_attempts Tablosu İndeksleme:

- INDEX idx_ip_time (ip_address, attempted_at): Bu bileşik indeks, belirli bir zaman aralığında belirli bir IP için denemeleri hızlıca COUNT(*) yapması gereken oran sınırlama mantığı için hayati önem taşır.[36]

4.4. Süresi Dolmuş Oturumların Otomatik Temizlenmesi

Veritabanında süresi dolmuş/eski oturumları bırakmak, tablo şişmesine ve performans düşüşüne yol açar. Otomatik bir temizleme mekanizmasına ihtiyaç vardır.

Yöntem 1: MySQL Olay Zamanlayıcısı (Tavsiye Edilir)

- Bu, veritabanı sunucusu içinde doğrudan zamanlanmış SQL sorguları çalıştırabilen yerel bir MySQL özelliğidir.[44]
- **Uygulama**: CREATE EVENT cleanup_expired_qr_sessions ON SCHEDULE EVERY 1 MINUTE DO DELETE FROM qr_login_sessions WHERE expires_at < NOW();
- **Artıları**: PHP betiği veya cron işinin ek yükü olmadığı için son derece verimlidir. Veritabanı içinde kendi kendine yeterlidir.
- **Eksileri**: event_scheduler genel değişkeninin ON olmasını ve kullanıcının EVENT ayrıcalıklarına sahip olmasını gerektirir.[44]

Yöntem 2: PHP Cron İşi

- Sistemin cron daemon'u aracılığıyla zamanlanmış bir DELETE sorgusu çalıştıran bir PHP betiği.[45]
- **Artıları**: Birçok PHP geliştiricisi için daha tanıdıktır; mantık uygulama kod tabanı içindedir.
- **Eksileri**: Daha az verimlidir. PHP yorumlayıcısını başlatmayı, DB'ye bağlanmayı ve ardından sorguyu çalıştırmayı içerir. Başarısız olabilecek daha fazla hareketli parça vardır.

Bir MySQL depolama motoru seçimi küçük bir yapılandırma ayarı değil, sistemin eşzamanlı işlemleri yönetme yeteneğini belirleyen temel bir mimari karardır. Bu tür bir sistem için MyISAM'ı seçmek, performansı öldüren kritik bir hata olur. Giriş akışı, birçok kullanıcı için aynı anda sık ve hızlı durum değişiklikleri gerektirir. MyISAM'ın tablo düzeyinde kilidi [40], bu işlemlerin serileştirileceği anlamına gelir. Bu, kullanıcı trafiğiyle doğrusal olarak büyüyecek bir kuyruk oluşturur ve zaman aşımlarına ve sistem arızasına yol açar. InnoDB'nin satır düzeyinde kilitlemesi [38], bu güncellemelerin paralel

olarak gerçekleşmesine izin verir, bu da gerekli performans ve ölçeklenebilirliği elde etmenin tek yoludur.

Bölüm 5: Çok Katmanlı Bir Güvenlik Çerçevesi

Bu bölüm, hem QR'a özgü tehditleri hem de OWASP ilkelerine dayalı genel web zafiyetlerini ele alarak sistemi güçlendirmek için gereken özel güvenlik önlemlerini detaylandırmaktadır.

5.1. Birincil Tehdit: QRLJacking ve Yeniden Oynatma Saldırılarını Azaltma

Zafiyet: QRLJacking (QR-kod Giriş Ele Geçirme), bir saldırganın kurbanı kendi QR kodunu taramaya ikna ederek, kurbanın kimliği doğrulanmış mobil oturumunu saldırganın web oturumuna bağladığı bir saldırıdır.[48] Bu bir yeniden oynatma saldırısı türüdür.

Temel Önlem: QR kodunun yükü **efemeral (kısa ömürlü) ve tek kullanımlık** olmalıdır. Bu, en önemli tek güvenlik kontrolüdür.[1]

Uygulama Adımları:

1. **Kısa Son Kullanma Tarihi**: qr_login_sessions tablosundaki expires_at alanı kısa bir süreye (örneğin, 60-120 saniye) ayarlanmalıdır. Sunucu, süresi dolmuş oturumlar için yapılan tüm onay denemelerini reddetmelidir.
2. **Tek Kullanımlık Nonce**: Veritabanında saklanan nonce kritiktir. Mobil uygulama onayı gönderdiğinde, sunucu nonce'ı doğrulamalıdır. Başarılı bir doğrulamadan hemen sonra, oturumun durumu confirmed veya expired olarak güncellenmeli ve nonce kullanılmış sayılmalıdır. Aynı session_id ve nonce'ı kullanmaya yönelik sonraki tüm girişimler reddedilmelidir. Bu, ele geçirilen bir QR kodunun ikinci kez kullanılmasını önler.[50]
3. **Bağlamsal Doğrulama (İsteğe Bağlı ama Tavsiye Edilir)**: Sunucu, QR kodunu talep eden web istemcisinin IP adresi/coğrafi konumu ile onu onaylayan mobil istemcinin IP'sini karşılaştırabilir. Eğer coğrafi olarak farklılarsa (örneğin, New York'tan web isteği, Moskova'dan mobil onay), deneme işaretlenebilir veya

engellenebilir.[49]

5.2. WebSocket Kanalının Güvenliğini Sağlama

Güvenli olmayan bir WebSocket, uygulamanıza doğrudan, kalıcı bir arka kapıdır.

- **WSS (Güvenli WebSockets) Zorunluluğu**: Tüm WebSocket bağlantıları, TLS üzerinden WebSocket olan wss:// protokolünü kullanmalıdır. Bu, TLS bağlantısını sonlandıracak ve arka uç Ratchet sunucusuna proxy yapacak olan Nginx ters proxy seviyesinde yapılandırılır.[53] Bu, ortadaki adam (man-in-the-middle) dinlemesini önler.
- **Origin Başlığı Doğrulaması**: Bir tarayıcı, WebSocket el sıkışması sırasında bağlantıyı başlatan alan adını belirten bir Origin başlığını otomatik olarak gönderir. Ratchet sunucusu, bu başlığı kontrol etmek ve yetkisiz alan adlarından gelen bağlantıları reddetmek için *yapılandırılmalıdır*. Bu, kötü niyetli bir web sitesinin (evil.com) kullanıcının tarayıcısında sunucunuza (your-app.com) bir WebSocket açmasını önler; bu zafiyet Siteler Arası WebSocket Ele Geçirme (CSWH) olarak bilinir.[55] Ratchet'in
  OriginCheck bileşeni bu amaç için tasarlanmıştır.[55]
- **WebSocket Kimlik Doğrulaması**: Bağlantının kendisi doğrulanmalıdır. Tarayıcının WebSocket API'sinde özel başlıklar kolayca geçirilemediğinden [58], en pratik yöntem, bağlantı URL'sinin sorgu dizesinde kısa ömürlü, tek kullanımlık bir belirteç geçirmektir.[58] Akış: İlk AJAX isteği (
  /qr-code/request) sessionId, nonce VE geçici bir ws_token döndürür. İstemci daha sonra wss://.../?sessionId={sessionId}&token={ws_token} adresine bağlanır. Ratchet'teki onOpen yöntemi, bağlantıyı kabul etmeden önce bu belirteci doğrular.

5.3. Temel Web Güvenliği (OWASP İlkeleri)

- **A01: Bozuk Erişim Kontrolü**: Tüm uç noktalarda katı yetkilendirme kontrolleri uygulayın. /login/scan/confirm uç noktası, kim olduklarını doğrulamak için mobil kullanıcının kimlik doğrulama belirtecini doğrulamalıdır.[60]
- **A02: Kriptografik Hatalar**: Taşıma sırasındaki tüm veriler için HTTPS/WSS kullanın. Kullanıcı şifrelerini Argon2 veya bcrypt gibi modern, güçlü bir algoritma ile hashleyin.[62]

- **A03: Enjeksiyon**: SQL Enjeksiyonunu önlemek için tüm veritabanı etkileşimlerinde PDO veya MySQLi ile parametreli sorgular (hazırlanmış ifadeler) kullanın. Kullanıcı girdisini asla SQL dizelerine birleştirmeyin.[60]
- **A05: Güvenlik Yanlış Yapılandırması**: Sunucu yapılandırmalarını sıkılaştırın. Ayrıntılı hata mesajlarının kullanıcıya gösterilmediğinden emin olun.
- **A07: Kimlik Tespiti ve Kimlik Doğrulama Hataları**:
  - **Oran Sınırlaması (Rate Limiting)**: Kaba kuvvet ve hizmet reddi saldırılarını önlemek için /qr-code/request uç noktasında ve geleneksel giriş formlarında oran sınırlaması uygulayın. Bu, login_attempts tablosunda IP başına denemelerin izlenmesini içerir.[36]
  - **Hesap Kilitleme**: Belirli sayıda başarısız denemeden sonra hesabı veya IP'yi geçici olarak kilitleyin.[63]
- **A08: Yazılım ve Veri Bütünlüğü Hataları**: Composer gibi bir bağımlılık yöneticisi kullanın ve üçüncü taraf kütüphanelerdeki zafiyetleri düzenli olarak tarayın.
- **CSRF Koruması**: Uygulamadaki herhangi bir geleneksel HTML formu için (örneğin, bir profil güncelleme sayfası), Siteler Arası İstek Sahtekarlığını (CSRF) önlemek için Senkronize Edici Belirteç Desenini (Synchronizer Token Pattern) uygulayın.[66]

Tablo: Güvenlik Tehdidi ve Azaltma Matrisi

Tehdit	Açıklama	Azaltma Stratejisi
QRLJacking/Yeniden Oynatma Saldırısı	Saldırgan, kullanıcının oturumunu ele geçirmek için kötü niyetli bir QR kodunu taraması için onu kandırır.	Kısa ömürlü (60s) QR kodları kullanın. QR verisine tek kullanımlık bir nonce gömün ve ilk kullanımdan sonra sunucuda geçersiz kılın.[49]
Siteler Arası WebSocket Ele Geçirme	Kötü niyetli web sitesi, kullanıcının tarayıcısında sunucunuza bir WebSocket bağlantısı açar.	Ratchet sunucusunda, yalnızca kendi alan adınızdan gelen bağlantılara izin vermek için katı Origin başlığı kontrolü uygulayın.[55]
Gizlice Dinleme (Eavesdropping)	Saldırgan, istemci ve sunucu arasındaki verileri ele geçirir.	Nginx yapılandırması aracılığıyla tüm HTTP trafiği için HTTPS ve tüm WebSocket trafiği için WSS zorunlu kılın.[54]

SQL Enjeksiyonu	Saldırgan, verilere erişmek/değiştirmek için SQL sorgularını manipüle eder.	Tüm veritabanı işlemleri için parametreli sorgular (hazırlanmış ifadeler) kullanın.[60]
Kaba Kuvvet/Hizmet Reddi (DoS)	Saldırgan, giriş/QR isteği uç noktalarını otomatik isteklerle doldurur.	IP adresine dayalı oran sınırlaması uygulayın. Aşırı istekte bulunan IP'lerin denemelerini günlüğe kaydedin ve geçici olarak engelleyin.[36]
Oturum Sabitleme (Session Fixation)	Saldırgan, bir kullanıcıyı bilinen bir oturum ID'sini kullanmaya zorlar.	Başarılı giriş üzerine yeni, güvenli bir oturum belirteci oluşturun ve önceden var olan oturum tanımlayıcılarını yoksayın.

Bir QR kod sistemindeki güvenlik, QR kodunun kendisini "güvenli" yapmakla ilgili değildir (bu sadece verinin görsel bir temsilidir). Güvenlik, QR kodunun *etrafında* inşa edilen protokolün, özellikle zaman ve durum yönetiminin ortaya çıkan bir özelliğidir. Bir saldırgan bir QR kodunu kolayca kopyalayabilir.[2] Bu bir gerçektir. Bu nedenle, QR kodunun gizli olmasına dayanan herhangi bir güvenlik önlemi başarısız olacaktır. Kopyalanmış bir kodu yenmenin tek yolu, onu çok kısa bir süre sonra veya ilk kullanımından sonra işe yaramaz hale getirmektir. Bu,

expires_at zaman damgasının ve tek kullanımlık nonce'ın sadece özellikler değil, güvenlik modelinin mutlak temel taşı olduğu sonucuna götürür.

Bölüm 6: Dağıtım, Ölçeklenebilirlik ve Uyumluluk

Bu son bölüm, sistemi üretime taşıma, büyümesini sağlama ve yasal veri gizliliği yükümlülüklerine uyma konusunda pratik rehberlik sağlar.

6.1. Üretim Dağıtım Stratejisi

Supervisor ile Süreç Yönetimi:

- Ratchet WebSocket sunucusu, uzun süren bir PHP CLI işlemidir ve SSH oturumu kapanırsa veya betik çökerse sonlanır.[31]
- **Supervisor**, Ratchet sunucusunu bir daemon olarak çalıştırmak için kullanılacak bir süreç kontrol sistemidir. Sürecin her zaman çalışmasını sağlar ve başarısız olursa otomatik olarak yeniden başlatır.[31]
- Ratchet süreci için komut, kullanıcı, autostart ve autorestart parametrelerini belirten örnek bir supervisord.conf yapılandırma dosyası sunulmalıdır.[31]

Ini, TOML

```
[program:ratchet-qr-server]
command=php /var/www/your-project/bin/server.php
process_name=%(program_name)s_%(process_num)02d
numprocs=1
autostart=true
autorestart=true
user=www-data
stdout_logfile=/var/log/supervisor/ratchet-qr.log
stderr_logfile=/var/log/supervisor/ratchet-qr.err.log
```

Nginx ile Ters Proxy:

- Nginx, tüm trafik için tek giriş noktası olarak hareket edecektir.
- **Sorumluluklar**:
  1. Tüm HTTPS ve WSS trafiğini yöneterek SSL/TLS'yi sonlandırmak.
  2. Statik varlıkları (CSS, JS, resimler) doğrudan sunmak.
  3. Standart HTTP isteklerini (örneğin, /api/*) PHP-FPM hizmetine proxy'lemek.
  4. WebSocket isteklerini (/ws/*) belirli bir bağlantı noktasında (örneğin, 8080) çalışan arka uç Ratchet sunucusuna proxy'lemek. Bu, proxy_http_version 1.1 ve Upgrade / Connection başlıklarını içeren özel bir location bloğu gerektirir.[32]

Nginx

```
upstream websocket_server {
```

```
    server 127.0.0.1:8080;
}

server {
    listen 443 ssl http2;
    server_name yourdomain.com;

    # SSL/TLS yapılandırması buraya gelir
    ssl_certificate /path/to/your/fullchain.pem;
    ssl_certificate_key /path/to/your/privkey.pem;

    location / {
        # Standart web istekleri için
        try_files $uri $uri/ /index.php?$query_string;
    }

    location ~ \.php$ {
        # PHP-FPM'e proxy
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php8.2-fpm.sock;
    }

    location /ws/ {
        # WebSocket bağlantıları için proxy
        proxy_pass http://websocket_server;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
        proxy_set_header Host $host;
        proxy_read_timeout 86400; # Bağlantıyı açık tut
    }
}
```

6.2. WebSocket Katmanını Ölçeklendirme

Tek bir WebSocket sunucu süreci bir darboğaz olabilir. Daha fazla eşzamanlı kullanıcıyı

yönetmek için birden fazla Ratchet sunucu süreci çalıştırmamız gerekir.

Zorluk: WebSocket bağlantıları durum bilgisi içerir. Bir yük dengeleyici, tek bir bağlantı için sonraki paketleri farklı sunuculara dağıtırsa, bağlantı kopar.

Çözüm 1: Yapışkan Oturumlar (Basit)

- Bu yaklaşım, belirli bir istemciden gelen tüm isteklerin her zaman aynı arka uç sunucu sürecine yönlendirilmesini sağlar.
- **Uygulama**: Nginx'in upstream bloğunda ip_hash yönergesini kullanın.[71] Bu, hedef sunucuyu belirlemek için istemcinin IP adresini hashler.
- **Sınırlamalar**: Bu yöntem, birçok kullanıcı aynı kurumsal NAT veya proxy arkasındaysa dengesiz yüklere yol açabilir, çünkü hepsi aynı sunucuya yönlendirilecektir.[71]

Çözüm 2: Paylaşılan Durum Mimarisi (Gelişmiş)

- Daha sağlam ancak daha karmaşık bir çözüm, oturum durumunu dışsallaştırmayı içerir.
- **Uygulama**: Her PHP sürecinin belleğindeki istemci bağlantı haritasını ($this->clients) saklamak yerine, **Redis** gibi merkezi, hızlı bir veri deposu kullanın.
- Bir bağlantı Sunucu A'da açıldığında, bağlantı ayrıntılarını Redis'e kaydeder. Mobil uygulama girişi onayladığında, HTTP işleyicisi (herhangi bir sunucuda olabilir) Redis'teki bağlantı ayrıntılarını arar ve bir Redis Pub/Sub kanalına bir mesaj yayınlar. Tüm WebSocket sunucuları bu kanala abone olur ve doğru sunucu (Sunucu A) mesajı alır ve uygun istemciye iletir.
- **Fayda**: Bu, sunucuları birbirinden ayırır ve herhangi bir sunucu sürecin herhangi bir bölümünü işleyebileceği için gerçek yük dengelemeye (örneğin, Round Robin, En Az Bağlantı) olanak tanır.

6.3. GDPR ve Veri Gizliliği Hususları

İşlenen Veriler: Sistem, IP adresleri, kullanıcı aracıları ve potansiyel olarak oturumlarla bağlantılı kullanıcı kimlikleri ve e-postalar dahil olmak üzere kişisel verileri işler.[75] Bu, GDPR kapsamına girer.

Uyulması Gereken Temel GDPR İlkeleri:

1. **Hukuka Uygunluk, Dürüstlük ve Şeffaflık**: Gizlilik politikasında kullanıcıya QR

giriş süreci için hangi verilerin neden toplandığını açıkça bildirin.[75]
2. **Amaç Sınırlaması**: Toplanan verileri (örneğin, IP adresi) yalnızca girişi güvence altına almak ve kolaylaştırmak amacıyla kullanın, ilgisiz izleme için değil.
3. **Veri Minimizasyonu**: Yalnızca kesinlikle gerekli olan verileri toplayın. Oturum günlüklerinde gereksiz ayrıntıları saklamayın.[75]
4. **Depolama Sınırlaması (Unutulma Hakkı)**: Oturum günlüklerini süresiz olarak tutmayın. Otomatik temizleme mekanizması (Bölüm 4.4), artık ihtiyaç duyulmayan verileri silerek bu ilkeye uymaya yardımcı olur. Kullanıcılar hesaplarının ve ilişkili verilerinin silinmesini talep edebilmelidir.[77]
5. **Bütünlük ve Gizlilik**: Tüm veriler, Bölüm 5'te detaylandırıldığı gibi şifreleme (HTTPS/WSS) ve güçlü erişim kontrolleri ile güvence altına alınmalıdır.[79]

Rıza: Giriş için oturum çerezleri genellikle "kesinlikle gerekli" kabul edilse ve açık rıza bannerları gerektirmese de, şeffaflık hala anahtardır. Giriş için QR kodlarının kullanımı açıkça açıklanmalıdır.[76] Anlık giriş işlevinin ötesindeki herhangi bir izleme, açık ve bilgilendirilmiş rıza gerektirir.

WebSocket tabanlı bir sistem için PHP kullanma kararı, geleneksel bir PHP web uygulamasında bulunmayan benzersiz dağıtım ve ölçeklendirme zorlukları ortaya çıkarır. Mimari, uzun süren süreçleri ve durum bilgisi olan bağlantıları hesaba katacak şekilde sıfırdan tasarlanmalıdır. Ölçeklendirmenin sadece daha fazla sunucu eklemek kadar basit olduğu düşünülebilir. Ancak, WebSockets'in durum bilgisi olan doğası bu varsayımı bozar. ip_hash yapışkan oturumu en doğrudan çözümdür [71], ancak bu bir "sızıntılı soyutlamadır". İşe yarayana kadar çalışır (örneğin, tüm kullanıcıların tek bir IP'den göründüğü bir üniversite kampüsü). Bu, gerçek, sağlam ölçeklenebilirliğin, süreç yerel durumundan Redis gibi bir araç kullanarak dağıtık bir durum modeline temel bir geçiş gerektirdiğini ortaya koyar.

Sonuç

Bu rapor, PHP ve MySQL kullanılarak modern, güvenli ve ölçeklenebilir bir QR kod giriş sisteminin inşası için kapsamlı bir mimari plan sunmuştur. Başarılı bir uygulamanın temel taşları aşağıdaki gibi özetlenebilir:

1. **Mimari Yaklaşım**: QR kod ile giriş, basit bir kimlik doğrulama adımından ziyade, web istemcisi, mobil doğrulayıcı ve arka uç arasında dikkatle yönetilmesi gereken **dağıtık bir durum makinesi** olarak ele alınmalıdır. Sistemin kalbi, bu geçici

oturumların durumunu (pending, confirmed, expired) güvenilir bir şekilde yöneten qr_login_sessions tablosudur.

2. **Teknoloji Seçimi**:
    - **QR Kod Üretimi**: endroid/qr-code kütüphanesi, akıcı API'si, logo gömme gibi gelişmiş özellikleri ve yerleşik doğrulama yetenekleri ile üretim ortamları için en uygun sunucu taraflı çözümdür.
    - **Gerçek Zamanlı İletişim**: **WebSockets**, düşük gecikme süresi ve kaynak verimliliği nedeniyle AJAX long-polling'e göre kesinlikle üstündür. **Ratchet**, saf PHP tabanlı olması ve geniş barındırma uyumluluğu sayesinde, özel sunucu eklentileri gerektiren Swoole gibi alternatiflere göre daha pragmatik bir seçimdir.
3. **Veritabanı Tasarımı**:
    - **InnoDB**, yüksek eşzamanlı yazma işlemleri gerektiren oturum tablosu için tek geçerli depolama motorudur. Satır düzeyinde kilitleme ve ACID uyumluluğu, sistemin ölçeklenebilirliği ve veri bütünlüğü için pazarlık edilemez gereksinimlerdir.
    - Süresi dolmuş oturumların periyodik olarak temizlenmesi için **MySQL Event Scheduler** kullanmak, PHP tabanlı bir cron işine göre daha verimli ve sağlam bir yöntemdir.
4. **Güvenlik Stratejisi**:
    - Sistemin en büyük zafiyeti olan **QRLJacking** ve yeniden oynatma saldırılarına karşı en etkili savunma, QR kodlarının **kısa ömürlü (60-120 saniye) ve tek kullanımlık bir nonce içermesi** ve sunucunun bu kuralları katı bir şekilde uygulamasıdır.
    - WebSocket kanalı, **WSS** ile şifrelenmeli, **Origin başlığı** kontrolü ile yetkisiz kaynaklardan gelen bağlantılar engellenmeli ve bağlantı anında geçici bir belirteç ile doğrulanmalıdır.
    - SQL enjeksiyonuna karşı parametreli sorgular ve kaba kuvvet saldırılarına karşı oran sınırlaması gibi temel OWASP ilkeleri eksiksiz uygulanmalıdır.
5. **Dağıtım ve Ölçeklendirme**:
    - Ratchet gibi uzun süren PHP süreçleri, üretim ortamında **Supervisor** gibi bir süreç yöneticisi ile daemon olarak çalıştırılmalıdır.
    - **Nginx**, SSL/WSS sonlandırma, statik içerik sunumu ve istekleri PHP-FPM ile Ratchet sunucusuna doğru şekilde yönlendiren bir ters proxy olarak yapılandırılmalıdır.
    - Ölçeklendirme için en basit yaklaşım, Nginx'in ip_hash yönergesi ile **yapışkan oturumlar** sağlamaktır. Daha büyük sistemler için ise, Redis gibi bir araçla **paylaşılan durum mimarisine** geçmek, gerçek yük dengeleme ve daha

yüksek kullanılabilirlik sağlar.

Özetle, önerilen bu mimari, performansı, güvenliği ve ölçeklenebilirliği ön planda tutarak, PHP ekosisteminin geleneksel sınırlarını aşan modern bir çözüm sunmaktadır. Geliştirme sürecinde bu ilkelere bağlı kalmak, son kullanıcıya sorunsuz ve güvenli bir deneyim sunan, aynı zamanda gelecekteki büyümeye hazır, sağlam bir sistemin ortaya çıkmasını sağlayacaktır.

Alıntılanan çalışmalar

1. System Design : Log in on a website by scanning a QR code with ..., erişim tarihi Temmuz 25, 2025, https://medium.com/@ys.yogendra22/system-design-log-in-on-a-website-by-scanning-a-qr-code-with-mobile-0aab147b5132
2. QR Code Authentication Design : r/SoftwareEngineering - Reddit, erişim tarihi Temmuz 25, 2025, https://www.reddit.com/r/SoftwareEngineering/comments/10474m9/qr_code_authentication_design/
3. How to Implement a QR Code Authentication System: Scan to Login - OwnID, erişim tarihi Temmuz 25, 2025, https://www.ownid.com/blog/how-to-implement-qr-code-authentication-a-walkthrough
4. Getting Started With QR Code Authentication | Ping Identity, erişim tarihi Temmuz 25, 2025, https://www.pingidentity.com/en/resources/blog/post/qr-code-login.html
5. QR code authentication method in Microsoft Entra ID, erişim tarihi Temmuz 25, 2025, https://learn.microsoft.com/en-us/entra/identity/authentication/concept-authentication-qr-code
6. Cross-Device Sign-In - Passkey Central, erişim tarihi Temmuz 25, 2025, https://www.passkeycentral.org/design-guidelines/optional-patterns/cross-device-sign-in
7. Manual QR Entry: Users | HYPR Identity Assurance Platform, erişim tarihi Temmuz 25, 2025, https://docs.hypr.com/docs/auth/authUserExp/authUserExpHma/auth-user-exp-hma-qr-fallback/
8. Top 5 JavaScript and PHP QR Code Packages for Your Projects - Codeboxr, erişim tarihi Temmuz 25, 2025, https://codeboxr.com/top-5-javascript-and-php-qr-code-packages-for-your-projects/
9. Dynamically generating a QR code with PHP [closed] - Stack Overflow, erişim tarihi Temmuz 25, 2025, https://stackoverflow.com/questions/5943368/dynamically-generating-a-qr-code-with-php

10. A curated list of awesome QR code libraries, software and resources - GitHub, erişim tarihi Temmuz 25, 2025, https://github.com/make-github-pseudonymous-again/awesome-qr-code
11. endroid/qr-code: QR Code Generator - GitHub, erişim tarihi Temmuz 25, 2025, https://github.com/endroid/qr-code
12. chillerlan/php-qrcode - Packagist, erişim tarihi Temmuz 25, 2025, https://packagist.org/packages/chillerlan/php-qrcode
13. chillerlan/php-qrcode: A PHP QR Code generator and reader with a user-friendly API., erişim tarihi Temmuz 25, 2025, https://github.com/chillerlan/php-qrcode
14. endroid/qr-code-bundle - Packagist, erişim tarihi Temmuz 25, 2025, https://packagist.org/packages/endroid/qr-code-bundle
15. zoujingli/qrcode - Packagist, erişim tarihi Temmuz 25, 2025, https://packagist.org/packages/zoujingli/qrcode
16. QR Code Login System In PHP MYSQL - YouTube, erişim tarihi Temmuz 25, 2025, https://www.youtube.com/watch?v=-9mdaTOkC7g
17. qrcode.js, erişim tarihi Temmuz 25, 2025, https://davidshimjs.github.io/qrcodejs/
18. How to make a QR Code generator using qrcode.js ? | GeeksforGeeks, erişim tarihi Temmuz 25, 2025, https://www.geeksforgeeks.org/how-to-make-a-qr-code-generator-using-qrcode-js/
19. How To Generate a QR Code In JavaScript - DEV Community, erişim tarihi Temmuz 25, 2025, https://dev.to/theudemezue/how-to-generate-a-qr-code-in-javascript-fhk
20. Long Polling vs WebSockets: What's best for realtime at scale? - Ably, erişim tarihi Temmuz 25, 2025, https://ably.com/blog/websockets-vs-long-polling
21. Evaluating Long Polling vs WebSockets - PubNub, erişim tarihi Temmuz 25, 2025, https://www.pubnub.com/blog/evaluating-long-polling-vs-websockets/
22. In what situations would AJAX long/short polling be preferred over HTML5 WebSockets?, erişim tarihi Temmuz 25, 2025, https://stackoverflow.com/questions/10028770/in-what-situations-would-ajax-long-short-polling-be-preferred-over-html5-websock
23. WebSocket in PHP – when & how to use it, examples | TSH.io - The Software House, erişim tarihi Temmuz 25, 2025, https://tsh.io/blog/php-websocket/
24. Swoole: Is that Node in PHP or am I crazy? | TSH.io - The Software House, erişim tarihi Temmuz 25, 2025, https://tsh.io/blog/swoole-is-it-node-in-php-or-am-i-wrong/
25. How to Create a WebSocket Server in PHP with Ratchet for Real-Time Applications | Twilio, erişim tarihi Temmuz 25, 2025, https://www.twilio.com/en-us/blog/developers/tutorials/building-blocks/create-php-websocket-server-build-real-time-even-driven-application
26. WebSocket with PHP: Building Real-Time Applications | by Yunus Emre Adas - Medium, erişim tarihi Temmuz 25, 2025, https://medium.com/write-a-catalyst/websocket-with-php-building-real-time-applications-3df6dcec1aa5
27. Robust WebSocket Reconnection Strategies in JavaScript With Exponential

Backoff, erişim tarihi Temmuz 25, 2025, https://dev.to/hexshift/robust-websocket-reconnection-strategies-in-javascript-with-exponential-backoff-40n1
28. How to implement a random exponential backoff algorithm in Javascript - Which Dev, erişim tarihi Temmuz 25, 2025, https://whichdev.com/how-to-implement-an-exponential-random-backoff-algorithm-in-javascript/
29. WebSocket: How to automatically reconnect after it dies - Stack Overflow, erişim tarihi Temmuz 25, 2025, https://stackoverflow.com/questions/22431751/websocket-how-to-automatically-reconnect-after-it-dies
30. ReconnectingWebsocket.js - a Websocket with an automatic reconnect — The CodeChat System 0.2.22 documentation, erişim tarihi Temmuz 25, 2025, https://codechat-system.readthedocs.io/en/latest/CodeChat_Server/CodeChat_Server/CodeChat_Client/static/ReconnectingWebsocket.js.html
31. How do I keep a websocket server running, even after I close the SSH terminal?, erişim tarihi Temmuz 25, 2025, https://stackoverflow.com/questions/19487664/how-do-i-keep-a-websocket-server-running-even-after-i-close-the-ssh-terminal
32. Tutorial: Installation - Ratchet, erişim tarihi Temmuz 25, 2025, http://socketo.me/docs/deploy
33. Using Supervisor to run ratchet websockets and queue in laravel - Stack Overflow, erişim tarihi Temmuz 25, 2025, https://stackoverflow.com/questions/43106480/using-supervisor-to-run-ratchet-websockets-and-queue-in-laravel
34. Database | Better Auth, erişim tarihi Temmuz 25, 2025, https://www.better-auth.com/docs/concepts/database
35. Secure Login System with PHP and MySQL - CodeShack, erişim tarihi Temmuz 25, 2025, https://codeshack.io/secure-login-system-php-mysql/
36. How to limit login attempt Using PHP and MySQL - PHPGurukul, erişim tarihi Temmuz 25, 2025, https://phpgurukul.com/how-to-limit-login-attempt-using-php-and-mysql/
37. MyISAM vs InnoDB: Key Differences and Performance Comparison - Devart Blog, erişim tarihi Temmuz 25, 2025, https://blog.devart.com/myisam-vs-innodb.html
38. MyISAM vs. InnoDB: Differences Explained - phoenixNAP, erişim tarihi Temmuz 25, 2025, https://phoenixnap.com/kb/myisam-vs-innodb
39. MyISAM vs InnoDB: 7 Critical Differences - Hevo Data, erişim tarihi Temmuz 25, 2025, https://hevodata.com/learn/myisam-vs-innodb/
40. MySQL Performance: MyISAM vs InnoDB - Liquid Web, erişim tarihi Temmuz 25, 2025, https://www.liquidweb.com/blog/mysql-performance-myisam-vs-innodb/
41. MySQL Query Optimization: Mastering Indexing for Faster Queries | by Yashodha Ranawaka, erişim tarihi Temmuz 25, 2025, https://yashodharanawaka.medium.com/mysql-query-optimization-mastering-indexing-for-faster-queries-e6bcdb5ccdff
42. Using indexes to improve MySQL query performance - hosting.com, erişim tarihi

Temmuz 25, 2025, https://kb.hosting.com/docs/using-indexes-to-improve-mysql-query-performance
43. Best indexing strategy for time series in a mysql database with timestamps - Stack Overflow, erişim tarihi Temmuz 25, 2025, https://stackoverflow.com/questions/22101954/best-indexing-strategy-for-time-series-in-a-mysql-database-with-timestamps
44. How To Delete Old MySQL Entries Using An Event Scheduler? - Uptimia, erişim tarihi Temmuz 25, 2025, https://www.uptimia.com/questions/how-to-delete-old-mysql-entries-using-an-event-scheduler
45. How to delete expired data with cron job - mysql - Stack Overflow, erişim tarihi Temmuz 25, 2025, https://stackoverflow.com/questions/29576947/how-to-delete-expired-data-with-cron-job
46. Auto delete Expired Data Row in MySQL - Stack Overflow, erişim tarihi Temmuz 25, 2025, https://stackoverflow.com/questions/24568399/auto-delete-expired-data-row-in-mysql
47. Php mysql delete data every 10mins - SitePoint, erişim tarihi Temmuz 25, 2025, https://www.sitepoint.com/community/t/php-mysql-delete-data-every-10mins/47968
48. Qrljacking - OWASP Foundation, erişim tarihi Temmuz 25, 2025, https://owasp.org/www-community/attacks/Qrljacking
49. Introduction of QR code attacks and countermeasures - HKCert, erişim tarihi Temmuz 25, 2025, https://www.hkcert.org/blog/introduction-of-qr-code-attacks-and-countermeasures
50. What is a Replay Attack and How to Prevent it - Kaspersky, erişim tarihi Temmuz 25, 2025, https://usa.kaspersky.com/resource-center/definitions/replay-attack
51. What is a replay attack, and how can you prevent them? - TeamPassword, erişim tarihi Temmuz 25, 2025, https://teampassword.com/blog/replay-attack-prevention
52. What is Qrljacking and how can you prevent it? - Comparitech, erişim tarihi Temmuz 25, 2025, https://www.comparitech.com/blog/information-security/what-is-qrljacking/
53. ratchetphp/Ratchet: Asynchronous WebSocket server - GitHub, erişim tarihi Temmuz 25, 2025, https://github.com/ratchetphp/Ratchet
54. PHP WebSockets With Ratchet - Medium, erişim tarihi Temmuz 25, 2025, https://medium.com/@winni4eva/php-websockets-with-ratchet-5e76bacd7548
55. Component: OriginCheck | CORS - Ratchet, erişim tarihi Temmuz 25, 2025, http://socketo.me/docs/origin
56. Is the Origin header really useful for securing a WebSocket?, erişim tarihi Temmuz 25, 2025, https://security.stackexchange.com/questions/115716/is-the-origin-header-really-

useful-for-securing-a-websocket
57. Ratchet - How to prevent other websites from connect to my websocket server?, erişim tarihi Temmuz 25, 2025, https://stackoverflow.com/questions/57682382/ratchet-how-to-prevent-other-websites-from-connect-to-my-websocket-server
58. Essential guide to WebSocket authentication - Ably, erişim tarihi Temmuz 25, 2025, https://ably.com/blog/websocket-authentication
59. Authentication - websockets 15.0.1 documentation, erişim tarihi Temmuz 25, 2025, https://websockets.readthedocs.io/en/stable/topics/authentication.html
60. Secure Coding Practices Checklist - OWASP Foundation, erişim tarihi Temmuz 25, 2025, https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/stable-en/02-checklist/05-checklist
61. OWASP Explained: Secure Coding Best Practices - Codacy | Blog, erişim tarihi Temmuz 25, 2025, https://blog.codacy.com/owasp-top-10
62. Authentication - OWASP Cheat Sheet Series, erişim tarihi Temmuz 25, 2025, https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html
63. How can I throttle user login attempts in PHP - Stack Overflow, erişim tarihi Temmuz 25, 2025, https://stackoverflow.com/questions/2090910/how-can-i-throttle-user-login-attempts-in-php
64. Authentication - Laravel 12.x - The PHP Framework For Web Artisans, erişim tarihi Temmuz 25, 2025, https://laravel.com/docs/12.x/authentication
65. limit-login-attempts.php - GitHub, erişim tarihi Temmuz 25, 2025, https://github.com/JPry/limit-login-attempts/blob/master/limit-login-attempts.php
66. PHP CSRF - PHP Tutorial, erişim tarihi Temmuz 25, 2025, https://www.phptutorial.net/php-tutorial/php-csrf/
67. Understanding and Implementing CSRF Tokens in PHP Forms - Domain India, erişim tarihi Temmuz 25, 2025, https://www.domainindia.com/login/knowledgebase/554/Understanding-and-Implementing-CSRF-Tokens-in-PHP-Forms.html
68. Cross-Site Request Forgery Prevention - OWASP Cheat Sheet Series, erişim tarihi Temmuz 25, 2025, https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
69. QR codes within enterprise security: Key considerations and best practices | Yubico, erişim tarihi Temmuz 25, 2025, https://www.yubico.com/blog/qr-codes-within-enterprise-security-key-considerations-and-best-practices/
70. Deploy with Supervisor - websockets 15.0.1 documentation, erişim tarihi Temmuz 25, 2025, https://websockets.readthedocs.io/en/stable/deploy/supervisor.html
71. How to Handle WebSocket Load Balancing Without Losing the Connection Thread, erişim tarihi Temmuz 25, 2025, https://blog.thnkandgrow.com/how-to-handle-websocket-load-balancing-witho

ut-losing-connection-thread/
72. Sticky sessions with Nginx proxy - load balancing - Server Fault, erişim tarihi Temmuz 25, 2025, https://serverfault.com/questions/832790/sticky-sessions-with-nginx-proxy
73. Load balance https and websockets wth sticky sessions and NGINX - Server Fault, erişim tarihi Temmuz 25, 2025, https://serverfault.com/questions/503205/load-balance-https-and-websockets-wth-sticky-sessions-and-nginx
74. Enhancing WebSocket Load Balancing: A Deep Dive into Sticky IPs and Session ID Routing | by Hoài Nhớ (Nick) | Medium, erişim tarihi Temmuz 25, 2025, https://medium.com/@hoainho.work/enhancing-websocket-load-balancing-a-deep-dive-into-sticky-ips-and-session-id-routing-41aebe017f8e
75. GDPR Compliance for Businesses Using QR Codes, erişim tarihi Temmuz 25, 2025, https://qrcodeveloper.com/blog/gdpr-compliance-qr-codes/
76. Session Cookies and GDPR: Key Facts You Should Know for Compliance, erişim tarihi Temmuz 25, 2025, https://pandectes.io/blog/session-cookies-and-gdpr-key-facts-you-should-know/
77. EU General Data Protection Regulation (GDPR) - QR Planet, erişim tarihi Temmuz 25, 2025, https://qrplanet.com/gdpr
78. Pursuing Usable and Useful Data Downloads Under GDPR/CCPA Access Rights via Co-Design - USENIX, erişim tarihi Temmuz 25, 2025, https://www.usenix.org/system/files/soups2021-veys.pdf
79. Practice guide GDPR - Security of personal data 2024 - CNIL, erişim tarihi Temmuz 25, 2025, https://www.cnil.fr/sites/cnil/files/2024-03/cnil_guide_securite_personnelle_ven_0.pdf