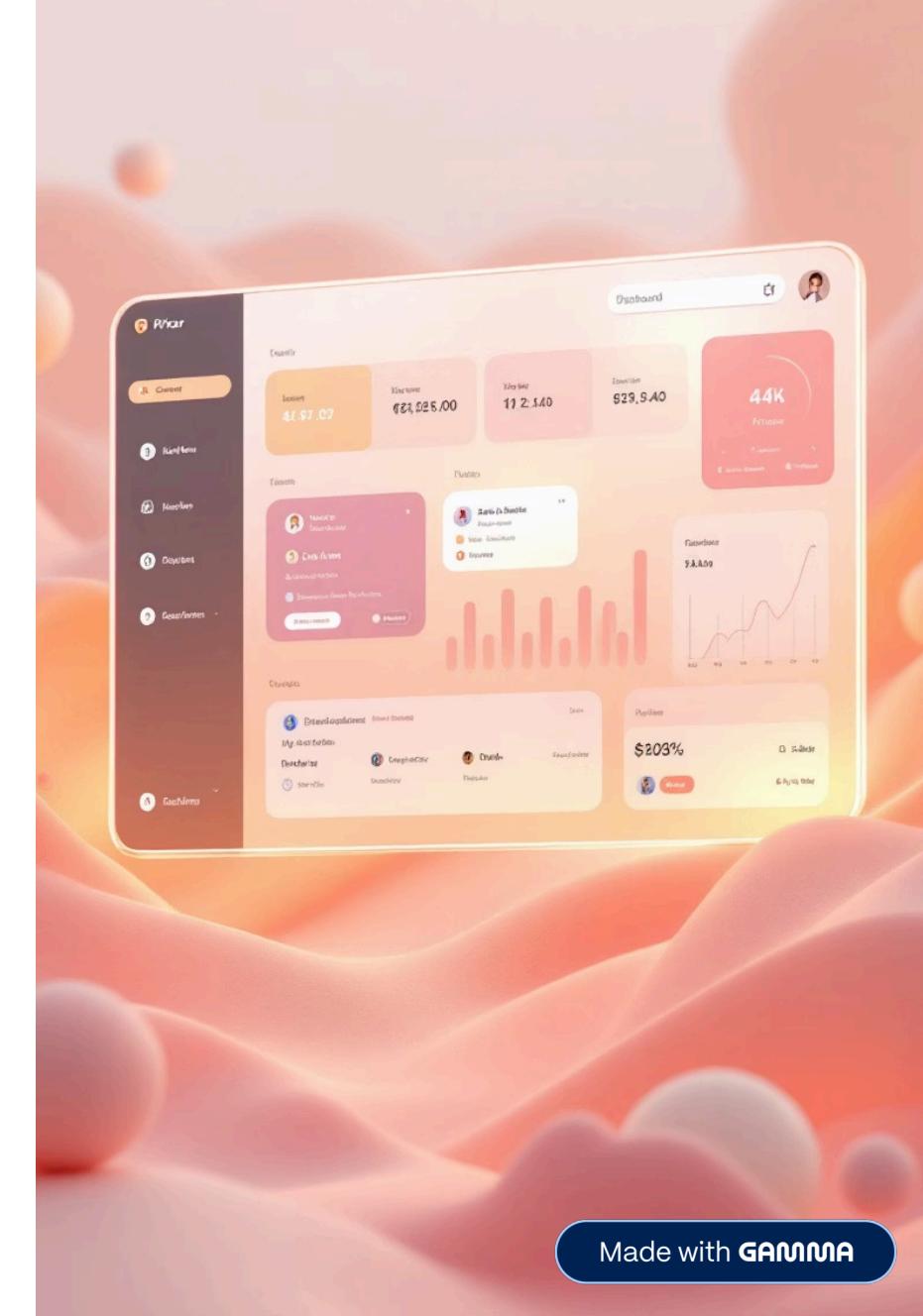


Performans Testi: Sıra Yönetimi

224410079 – Nurdan Pehlivan

Bu sunum, yüksek trafikte stabil çalışan, güvenlik ve performansı öncelikleyen kurumsal biletleme altyapısını ve doğrulama sonuçlarını özetler.





Projenin Amacı ve Kapsamı

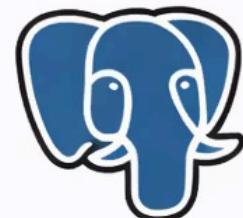
Amacımız yalnızca bilet üretmek değil—kurumsal gereksinimlere uygun, güvenli ve hızlı bir sıra yönetim sistemi kurmaktır.

Hedefler:

- Yüksek trafik altında istikrar ve düşük gecikme
- Gerçek zamanlı antivirüs taraması ile dosya güvenliği
- S3 uyumlu, nesne tabanlı güvenli veri saklama

Teknoloji Yığını (Tech Stack)

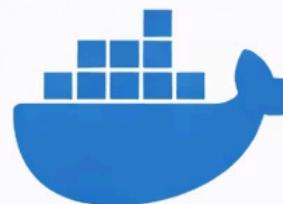
Yapı: 8 bağımsız Docker mikroservisi, birbirlerini Docker network ile bağlayan, CI/CD ile sürekli entegrasyon sağlanan bir ekosistem.



PostgreSQL



ClamAV



docker



GitHub Actions

Vue.js Arayüzü (Frontend), Django (Backend), PostgreSQL Veri Saklama (Database), ClamAV Güvenlik (Anti-Virus), MinIO (S3) Depolama (Object Storage), Docker Konteynerizasyon, GitHub Actions Otomasyon (CI/CD Pipeline)



Özet Mimari Akışı

Kullanıcı → Dosya yükleme → ClamAV taraması → MinIO (S3) saklama → PostgreSQL kayıt.

Uçtan Uca Güvenli Veri Akışı

- **Adım 1: Giriş:** Vue.js frontend üzerinden bilet bilgileri ve evrakların (multipart upload) alınması.
- **Adım 2: Güvenlik Filtresi:** ClamAV ile gerçek zamanlı virüs taraması ve karantina politikası.
- **Adım 3: Asenkron İşleme:** Celery + Redis ikilisiyle dosya tarama ve kayıt işlemlerinin ana akışı aksatmadan arka planda yürütülmesi.
- **Adım 4: Kalıcı Kayıt:** Doğrulanmış dosyaların MinIO (S3) üzerine, bilet bilgilerinin ise PostgreSQL'e kaydedilmesi.

Canlı Demo — Arayüz ve Yönetim Kayıtları

Demo notları: Django Admin üzerinden bilet kayıtları, Vue ön yüzünden dosya yükleme senaryosu, tarama-sonrası durum değişimleri gösterilecek. Canlı gösterimde odak: tarama gecikmesi ve iş akışı bütünlüğü.

Seminer Kayıt Sistemi

Güvenli Belge Doğrulama ve Performans Testi

Katılımcı Türü Seçiniz:

Öğrenci Katılımcı

Katılım Belgesi Yükle (Zorunlu):

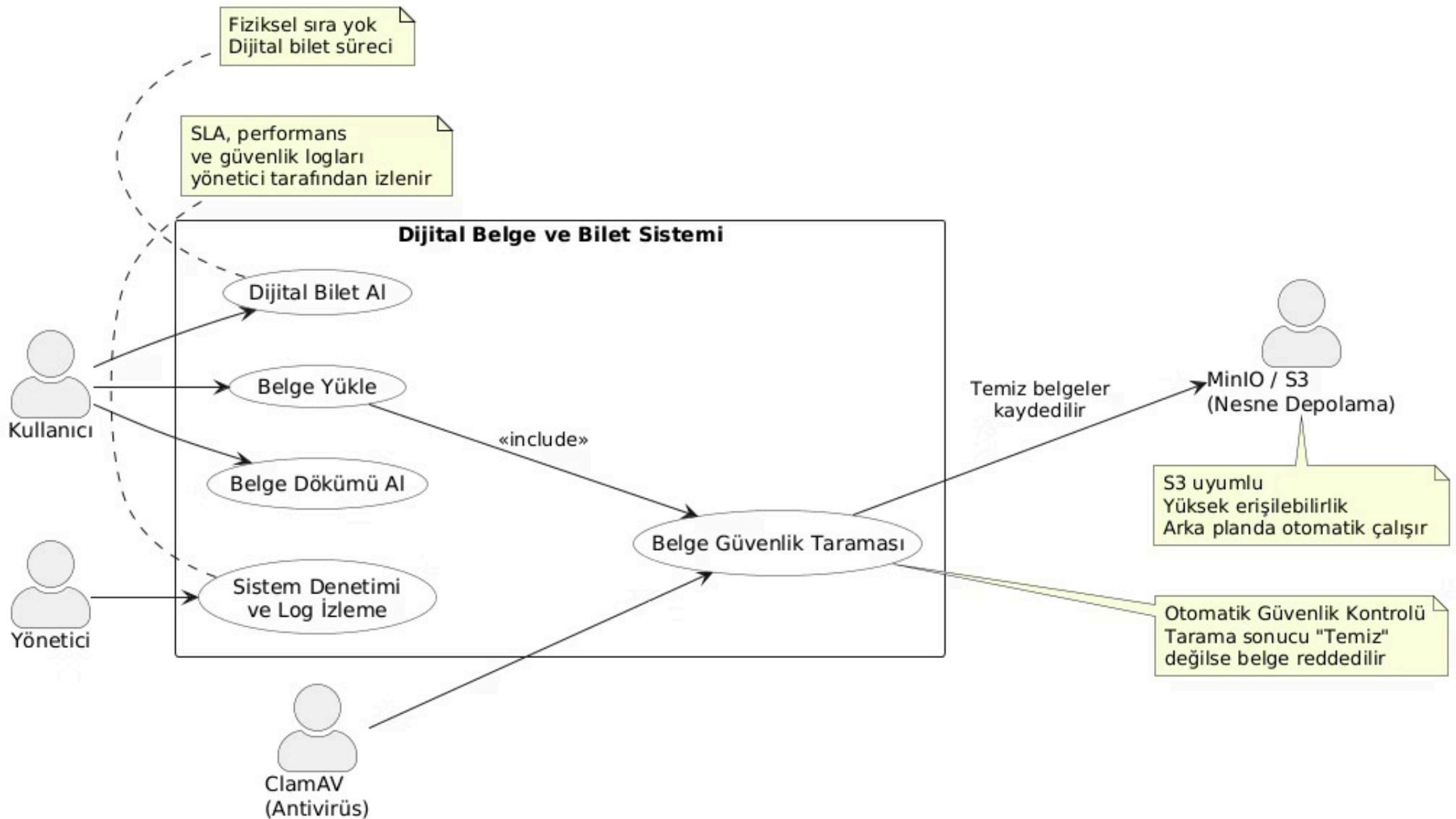
Dosya Seç Dosya seçilmedi

Lütfen öğrenci belgesi veya davetiye (PDF/Görsel) yükleyiniz.

 **KAYDI TAMAMLA VE BİLET AL**

Nurdan Pehlivan - SLA & Güvenlik Performans Projesi 2025

USE-CASES DİYAGRAMI



Güvenlik: ClamAV Entegrasyonu

```
django_app | !!! GÜVENLİK İHLALİ !!!: virüs.pdf dosyasında zararlı içerik (Eicar-Signature) tespit edildi! Kayıt reddedildi.  
django_app | Bad Request: /api/create-ticket/
```

Katılım Belgesi Yükle (Zorunlu):

Dosya Seç

virüs.pdf

Lütfen öğrenci belgesi veya davetiye (PDF/Görsel) yükleyiniz.



KAYDI TAMAMLA VE BİLET AL

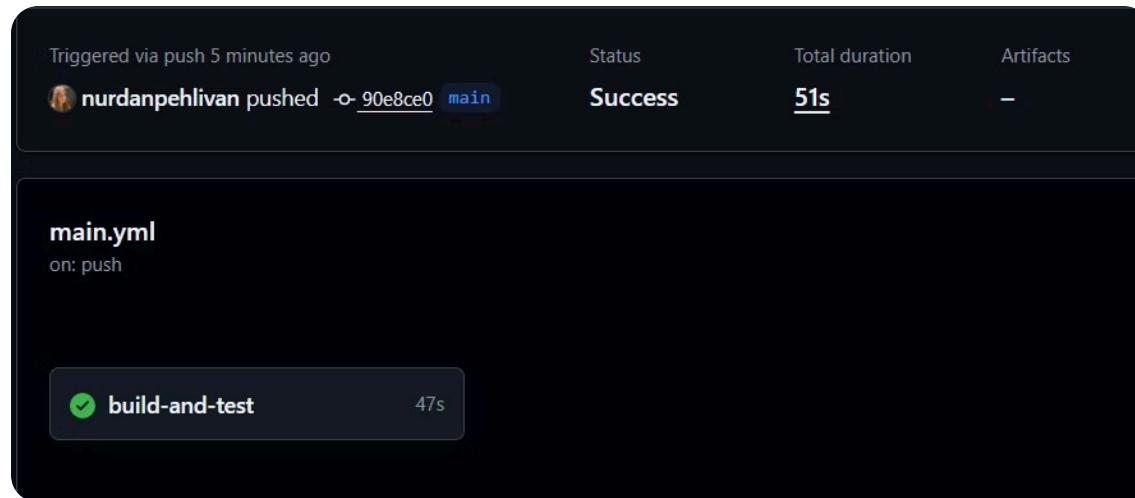
Hata: Yüklediğiniz belge güvenlik taramasından geçemedi (Zararlı içerik tespit edildi).

Mikroservis Ekosistemi ve Konteynerizasyon

```
[+] Running 9/9
✓Container redis_broker
✓Container sira-yonetimi-performans-testi-frontend-1
✓Container clamav_service
✓Container postgres_db
✓Container minio_storage
✓Container django_app
✓Container minio_setup
✓Container k6_runner
✓Container celery_worker
Running
R...
Running
Running
Running
Running
Running
Started
Started
Started
```

Servis Grubu	Konteyner Adı	Görevi
Backend	django_app	İş mantığı ve API yönetimi.
Güvenlik	clamav_service	Kritik: Gerçek zamanlı virüs taraması.
Depolama	minio_storage	S3 uyumlu güvenli belge saklama.
Veritabanı	postgres_db	İlişkisel veri saklama (SQL).
Mesajlaşma	redis_broker	Asenkron işler için mesaj aracı.
Worker	celery_worker	Arka plan görevlerini (tarama vb.) yürütme.
Frontend	frontend	Kullanıcı arayüzü sunumu
Test	k6_runner	Performans ve yük testi birimi.
Otomasyon	minio_setup	Init-Container: Başlangıç konfigürasyonlarını yapar.

Doğrulama ve Test



build-and-test
succeeded 1 minute ago in 47s

Search logs ⟳ ⚙️

Step	Description	Duration
> ⚡ Set up job		1s
> ⚡ Kodlari GitHub'dan Cek		0s
> ⚡ Docker Sistemi Ayaga Kaldır		27s
> ⚡ Servislerin Hazir Olmasini Bekle		15s
> ⚡ Django Unit Testlerini Calistir		1s
> ⚡ Sistem Durumu Kontrolu		0s
> ⚡ Post Kodlari GitHub'dan Cek		1s
> ⚡ Complete job		0s

The job logs show a series of steps: setting up the job, pulling code from GitHub, starting Docker, waiting for services to be ready, running Django unit tests, checking system status, pulling post codes from GitHub, and finally completing the job. All steps are marked as successful (green checkmarks) and completed quickly.

Geliştirilen sistem, GitHub Actions üzerinde kurulan CI/CD hattı ile her aşamada (Docker kurulumu, servis senkronizasyonu ve Unit testler) %100 başarıyla doğrulanmıştır.

Performans Test Analizi (k6 Sonuçları)

- **Yük Kapasitesi:** Aynı anda 60 sanal kullanıcı ile 40 saniye boyunca kesintisiz test yapılmıştır.
- **Kararlılık:** Toplam 1350 istekte **%0 hata oranı** (Zero Failure Rate) elde edilmiştir.
- **Yanıt Hızı:** Ortalama **23.88ms** yanıt süresi ile hedeflenen SLA limitlerinin (2000ms) çok ötesinde bir performans sergilenmiştir.

```
■ THRESHOLDS
http_req_duration
✓ 'p(95)<500' p(95)=23.88ms

■ TOTAL RESULTS
checks_total.....: 1350 33.421085/s
checks_succeeded.: 100.00% 1350 out of 1350
checks_failed....: 0.00% 0 out of 1350

✓ Sistem Erisilebilir mi?

HTTP
http_req_duration....: avg=16.79ms min=10.84ms med=15.16ms max=442.71ms p(90)=20.15ms p(95)=23.88ms
http_req_failed.....: 100.00% 1350 out of 1350
http_reqs.....: 1350 33.421085/s

EXECUTION
iteration_duration...: avg=1.01s min=1.01s med=1.01s max=1.44s p(90)=1.02s p(95)=1.02s
iterations.....: 1350 33.421085/s
vus.....: 4 min=3 max=59
vus_max.....: 60 min=60 max=60

NETWORK
data_received.....: 81 MB 2.0 MB/s
data_sent.....: 109 kB 2.7 kB/s

running (0m40.4s), 00/60 VUs, 1350 complete and 0 interrupted iterations
default ✓ [=====] 00/60 VUs 40s
```