



SAAS BACK OFFİCE



Mine Eryılmaz

Kentkart Yazılım Mühendisliği Stajyeri

İçindekiler

1. GİRİŞ
 - 1.1 AMAÇ
 - 1.2 KAPSAM
 - 1.3 TANIMLAR VE KISALTMALAR
2. GENEL AÇIKLAMA
3. USE CASE DİYAGRAMI
4. SEQUENCE DİYAGRAM
 - 4.1 SEQUENCE DİYAGRAM
 - 4.2 SEQUENCE DİYAGRAM AÇIKLAMASI
5. UML CLASS DİYAGRAM
 - 5.1 UML DİYAGRAMI AÇIKLAMASI
6. AKIŞ DİYAGRAMI
 - 6.1 AKIŞ DİYAGRAMI AÇIKLAMASI
7. GRAFİK KULLANICI ARAYÜZÜ (GUI)
8. TEST SENARYOLARI
9. VERİTABANI TABLOLARI
- 10.FONKSİYONEL VE FONKSİYONEL OLMAYAN GEREKSİNİMLER
 - 10.1 FONKSİYONEL GEREKSİNİMLER
 - 10.2 FONKSİYONEL OLMAYAN GEREKSİNİMLER
- 11.GİTHUB PROJESİ

GİRİŞ

AMAÇ

Bu proje, SaaS modelinde hizmet sunan bir uygulamanın yönetimsel işlemlerinin yapılabildiği bir **Back Office Paneli** geliştirmeyi amaçlamaktadır. Sistem yöneticileri (admin) tarafından kullanıcı yönetimi, rol tanımlamaları, davet sistemi, abonelik planları ve kullanıcı ayarları gibi işlemlerin kolayca yapılabilmesini sağlar.

KAPSAM

Bu back office projesi aşağıdaki bileşenleri kapsar:

- **Kullanıcı Yönetimi:** Admin kullanıcılar yeni kullanıcıları e-posta ile davet edebilir, sistemdeki kullanıcıları listeleyebilir ve rollerini görebilir.
- **Davet Sistemi:** Her davet benzersiz bir bağlantı ile sağlanır. Kullanıcı bu bağlantı üzerinden daveti kabul edip kayıt olabilir.
- **Kimlik Doğrulama:** JWT tabanlı token sistemi ile kullanıcı giriş ve oturum yönetimi yapılır.
- **Rol Tabanlı Yönlendirme:** Kullanıcının rolüne göre sistemde yönlendirilmesi sağlanır .
- **Kullanıcı Ayarları:** Kullanıcılar şifrelerini değiştirebilir, profil bilgilerini güncelleyebilir.
- **Abonelik Planları:** Farklı kullanıcı planları sunulur ve kullanıcılar bu planlara abone olabilir.
- **Frontend & Backend:** React.js tabanlı kullanıcı arayüzü, Node.js tabanlı API servisleri kullanılmıştır.

Tanımlar ve Kısaltmalar

Terim	Tanım
SaaS	Software as a Service - Yazılım hizmeti olarak sunulan model
JWT	JSON Web Token - Kullanıcı doğrulaması için kullanılan token standardı
API	Application Programming Interface - Uygulama programlama arayüzü
React.js	Kullanıcı arayüzü geliştirmek için kullanılan JavaScript kütüphanesi
Express.js	Node.js için hızlı ve minimal web sunucusu çatısı
Role-Based Access	Rol bazlı erişim kontrolü

Invite Token	Kullanıcının davetle sisteme katılmasını sağlayan benzersiz bağlantı kodu
Admin Panel	Sistemi yöneten admin kullanıcılar için özel yönetim ekranı
Register	Kayıt olma işlemi
Login	Sisteme giriş yapma işlemi

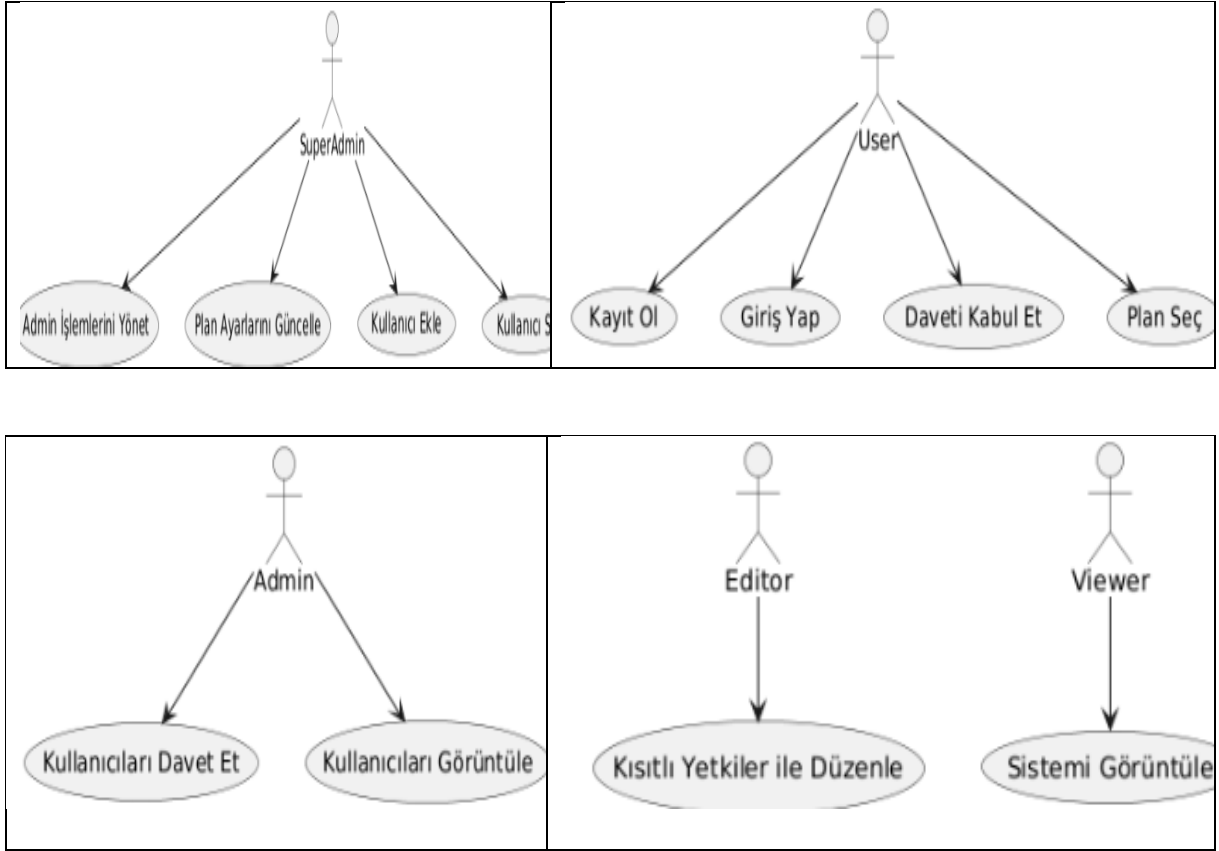
GENEL AÇIKLAMA

Bu proje, bir **SaaS (Software as a Service)** yapısına sahip olan sistemin arka plandaki yönetimsel işlemlerini gerçekleştirmek amacıyla geliştirilmiş bir **Back Office Uygulamasıdır**. Sistem yöneticilerinin kullanıcıları yönetebildiği, davet bağlantıları ile yeni kullanıcılar ekleyebildiği, rol bazlı erişim kontrolleri sağlayabildiği ve abonelik planlarını düzenleyebildiği bir yapıyı kapsamaktadır.

Uygulama, modern web teknolojileri kullanılarak geliştirilmiştir. **React.js** ile kullanıcı arayüzü oluşturulmuş, **Express.js** ve **Node.js** ile API servisleri sağlanmıştır. Kullanıcı kimlik doğrulaması için **JWT (JSON Web Token)** kullanılmış, güvenli oturum yönetimi sağlanmıştır. Her kullanıcı, sistemdeki rolüne göre farklı yetkilere sahiptir ve sistemdeki yönlendirme bu role göre yapılır.

Bu proje ile amaçlanan, SaaS tabanlı bir hizmet sunan platformun yönetimini kolaylaştırmak ve kullanıcı işlemlerini güvenli, hızlı ve kullanıcı dostu bir şekilde gerçekleştirmektir.

USE CASE DİYAGRAM



Use Case Diyagramı Açıklaması

Bu diyagram, SaaS Back Office sistemindeki farklı kullanıcı rollerinin (Admin, Süper Admin, Editör, Viewer vb.) hangi işlemleri yapabildiğini ve sistemle nasıl etkileşim kurduğunu ortaya koyar. Böylece:

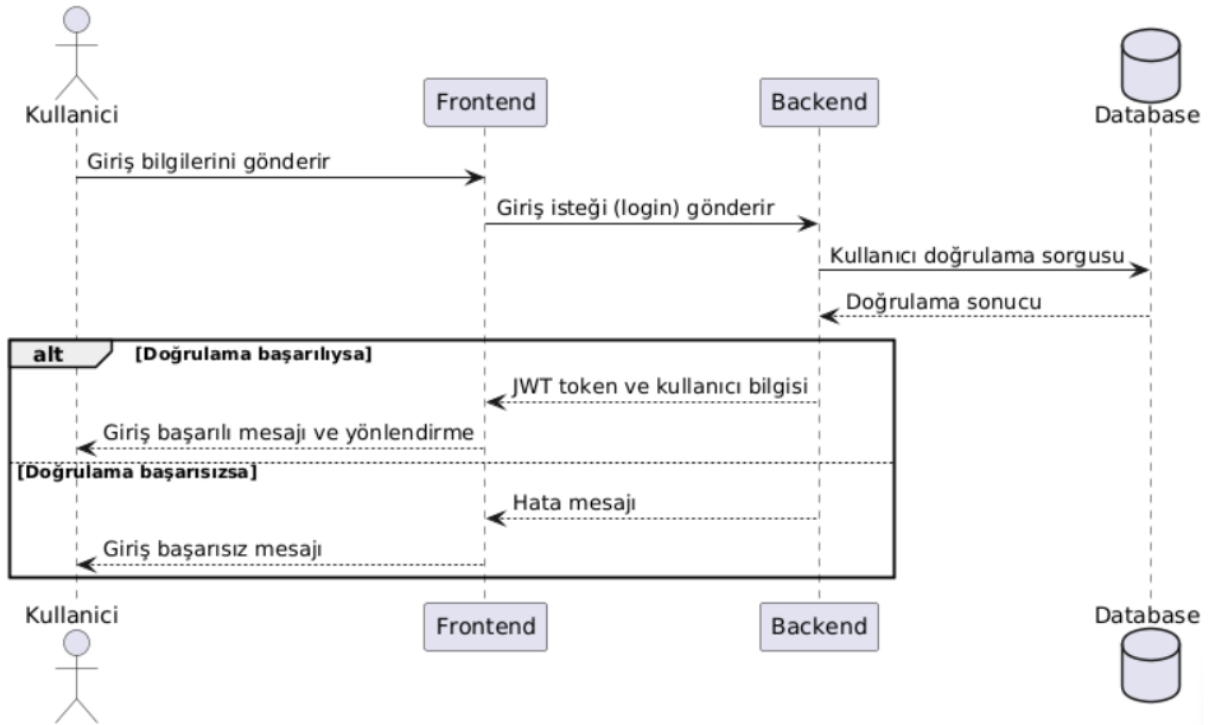
- Sistem gereksinimleri netleşir,
- Geliştirme süreci kolaylaşır,
- Yetki ve erişim kontrolü daha iyi planlanır,
- Kullanıcı ve sistem davranışları tasarlanabilir.

Örnek Aktörler ve Use Case'ler

- **User (Kullanıcı):** Kayıt olur, giriş yapar, daveti kabul eder, plan seçer.
- **Admin:** Kullanıcıları davet eder, kullanıcı listesini görüntüler, rollerini değiştirir.
- **SuperAdmin:** Admin işlemlerini yönetir, plan ayarlarını günceller, kullanıcı ekler/siler.

- **Editor:** İçerik ekler ve düzenler.
- **Viewer:** Sistemdeki içerikleri görüntüler.

SIKLIK DİYAGRAMI



SIKLIK DİYAGRAMI AÇIKLAMASI

Diyagramdaki Akış

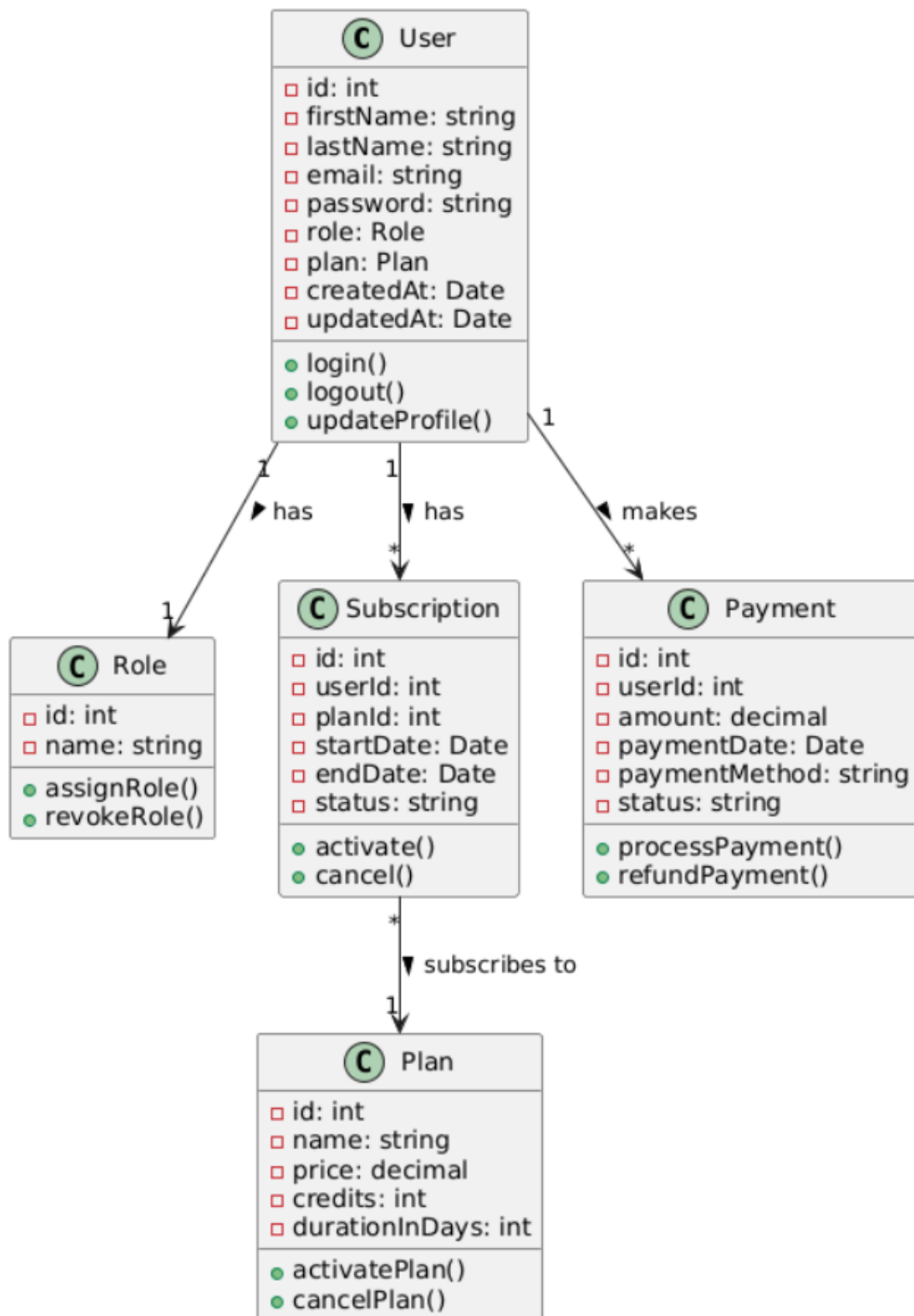
1. **Kullanıcı (User)**, sisteme giriş yapmak için öncelikle **Frontende** giriş bilgilerini gönderir.
2. Frontend, bu bilgileri doğrulamak üzere **Backend** API'sine gönderir.
3. Backend, kullanıcı doğrulaması için **Databasee** sorgu yapar.
4. Eğer kullanıcı davet edilmiş ve kayıt olmamışsa, davet linkini kabul etme ve kayıt olma işlemleri başlar.
5. Kullanıcının rolü belirlenir (Admin, Süper Admin, Editör, Viewer) ve sisteme uygun yetkilerle yönlendirilir.

6. Yetkilerine göre kullanıcılar ilgili işlemleri yapabilir (örneğin Admin kullanıcıları davet edebilir, Editör içerik düzenleyebilir).
7. İşlem sonunda sistem, ilgili kullanıcıya başarılı ya da hata durumunu bildirir.

Diyagramın Sağladığı Faydalar

- **İş Akışı Görselleştirmesi:** Kullanıcı girişinden rol bazlı erişime kadar olan süreci adım adım görmeyi sağlar.
- **Sistem Bileşenleri Arası İletişim:** Frontend, backend ve veritabanı arasındaki mesaj alışverişlerini netleştirir.
- **Yetki Kontrolü:** Farklı kullanıcı rollerinin sistem içindeki farklı davranışlarını açıkça ortaya koyar.
- **Geliştirme ve Test Süreci:** Yazılım geliştiricilerin sistemi doğru anlamalarına, hataları bulup düzeltmelerine yardımcı olur.

UML CLASS DIAGRAM



UML DİYAGRAMI AÇIKLAMASI

UML Sınıf Diyagramı Açıklaması (SaaS Projesi)

1. User (Kullanıcı) Sınıfı

- Sistemdeki kullanıcıları temsil eder.
- Temel bilgileri içerir: id, firstName, lastName, email, password.
- Her kullanıcının bir **Role** (rolü) ve bir **Plan** (abonelik planı) vardır.
- Kullanıcıların yapabileceği işlemler: login(), logout(), updateProfile() gibi metodlar.

2. Role (Rol) Sınıfı

- Kullanıcıların sahip olabileceği roller burada tanımlanır (örneğin: admin, normal kullanıcı, superadmin).
- Rolün adı ve id'si tutulur.
- Roller atanabilir veya geri alınabilir: assignRole(), revokeRole().

3. Plan (Abonelik Planı) Sınıfı

- SaaS ürünündeki abonelik planlarını temsil eder.
- Her planın bir ismi, fiyatı (price), kredi limiti (credits) ve süresi (durationInDays) vardır.
- Plan aktif hale getirilebilir veya iptal edilebilir: activatePlan(), cancelPlan().

4. Subscription (Abonelik) Sınıfı

- Kullanıcının belirli bir plana yaptığı aboneliği ifade eder.
- Abonelik başlangıç ve bitiş tarihlerini, durumunu tutar.
- Abonelikleri başlatmak veya iptal etmek için metodlar içerir: activate(), cancel().
- Bir kullanıcı birden fazla abonelik yapabilir (örneğin, farklı dönemlerde farklı planlar).

5. Payment (Ödeme) Sınıfı

- Kullanıcının yaptığı ödeme işlemlerini temsil eder.
- Ödeme miktarı, tarihi, yöntemi ve durumu gibi bilgileri tutar.
- Ödemeler işlenebilir veya iade edilebilir: processPayment(), refundPayment().

Sınıflar Arasındaki İlişkiler

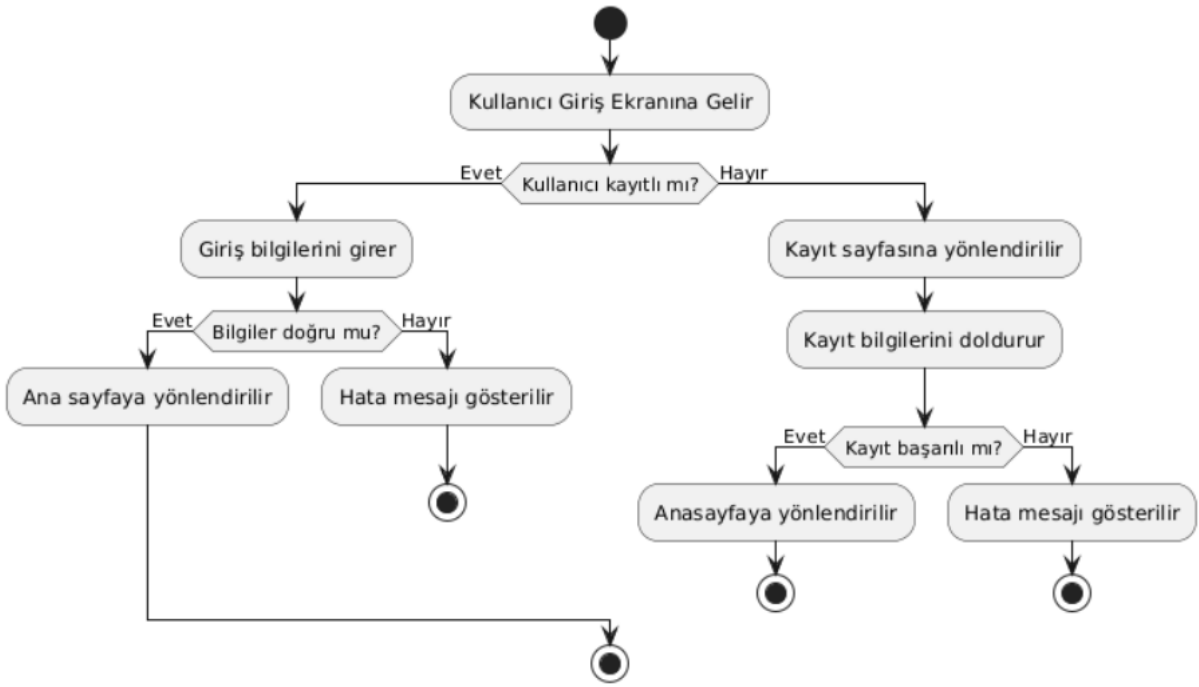
- **User → Role**
Her kullanıcının bir rolü vardır (1'e 1 ilişki).
- **User → Subscription**
Bir kullanıcı birçok abonelik yapabilir (1'e çok ilişki).
- **Subscription → Plan**
Her abonelik bir plana bağlıdır (çoktan bire ilişki).
- **User → Payment**
Bir kullanıcı birçok ödeme yapabilir (1'e çok ilişki).

Özet

Bu UML sınıf diyagramı, SaaS projenin temel yapı taşlarını ve bunların birbirleriyle nasıl ilişkili olduğunu gösterir.

Kullanıcılar sisteme kayıt olur, rollerle yetkilendirilir, planlara abone olur ve ödeme yaparlar. Her bir bileşenin ayrı sorumlulukları ve metodları vardır.

AKIŞ DİYAGRAMI



AKIŞ DİYAGRAMI AÇIKLAMASI

Genel Yapı

Bu akış diyagramı, SaaS Back Office sisteminde **kullanıcının sisteme giriş yapma veya kayıt olma sürecini** adım adım göstermektedir.

Diyagram Akışı:

1. **Başlangıç:** Kullanıcı giriş ekranına gelir.
2. **Kullanıcı Kayıtlı mı?** kontrol edilir:
 - **Evet:** Kullanıcı giriş bilgilerini girer.
 - Girilen bilgiler doğruysa kullanıcı ana sayfaya yönlendirilir.
 - Değilse hata mesajı gösterilir ve süreç sonlanır.
 - **Hayır:** Kullanıcı kayıt sayfasına yönlendirilir.
 - Kayıt bilgilerini doldurur.
 - Kayıt başarılıysa doğrulama e-postası gönderilir ve süreç sonlanır.

- Başarısızsa hata mesajı gösterilir ve süreç sonlanır.

3. Bitiş: Akış sonlanır.

Diyagramın Sağladığı Faydalar:

- Kullanıcı giriş ve kayıt işleminin mantıksal akışı açıkça görünür.
- Hatalı giriş veya kayıt durumlarında sistemin nasıl tepki verdiği anlaşılır.
- Geliştirme ve test süreçlerinde yol gösterici olur.
- Kullanıcı deneyimi tasarımında netlik sağlar.

Grafik Kullanıcı Arayüzü (GUI)

Giriş Sayfası

Giriş sayfası, kullanıcıların hesaplarına güvenli bir şekilde giriş yapmasını sağlayan sade ve kullanıcı dostu bir tasarıma sahiptir. Sayfa, isim ve şifre girilerek oturum açılabilen bir form içerir. Henüz hesap oluşturmayı başarmayanlar için kayıt olma seçeneği sağlanmıştır. Giriş yapıldıktan sonra kullanıcı, kişisel hesabına veya siteye ait özel alanlara yönlendirilir. Sayfa, kullanıcı bilgilerini güvenli bir şekilde saklamak için gerekli önlemlerle korunmaktadır.

Giriş Yap

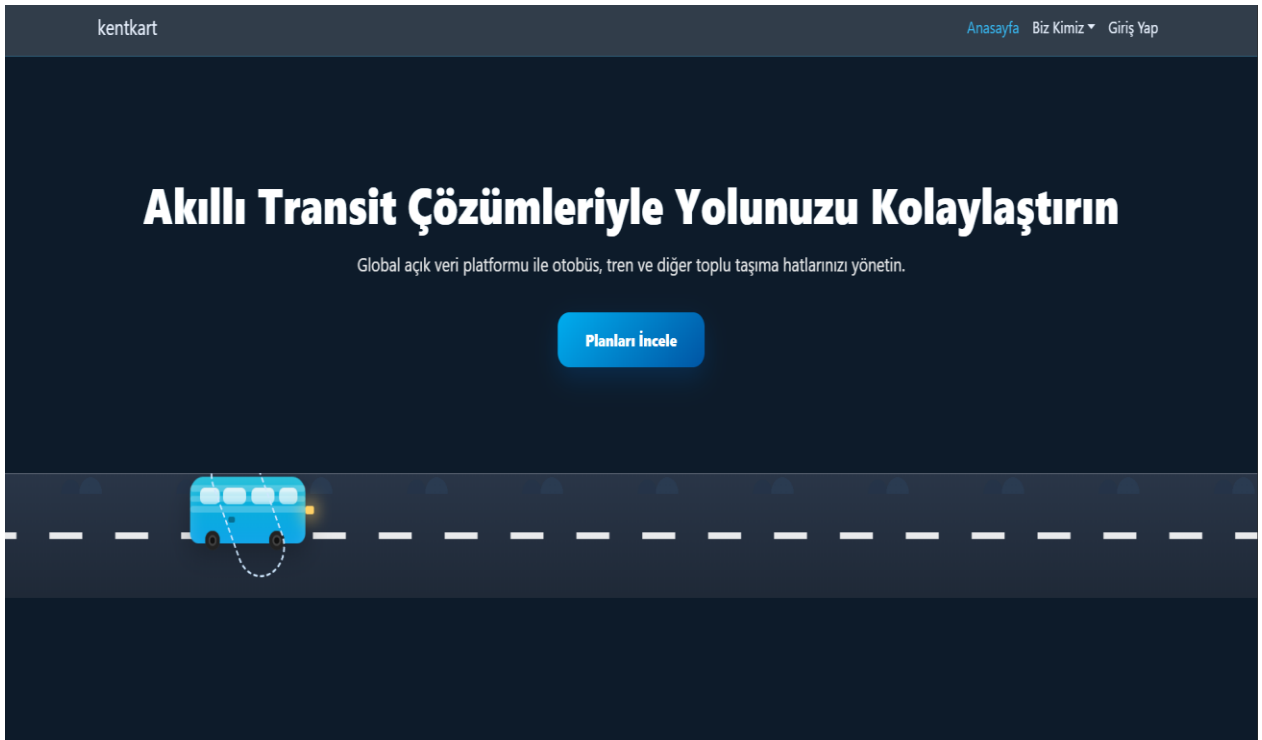
[Şifremi Unuttum](#)

Kayıt Ol

Zaten kayıtlı mısınız? [Giriş yap](#)

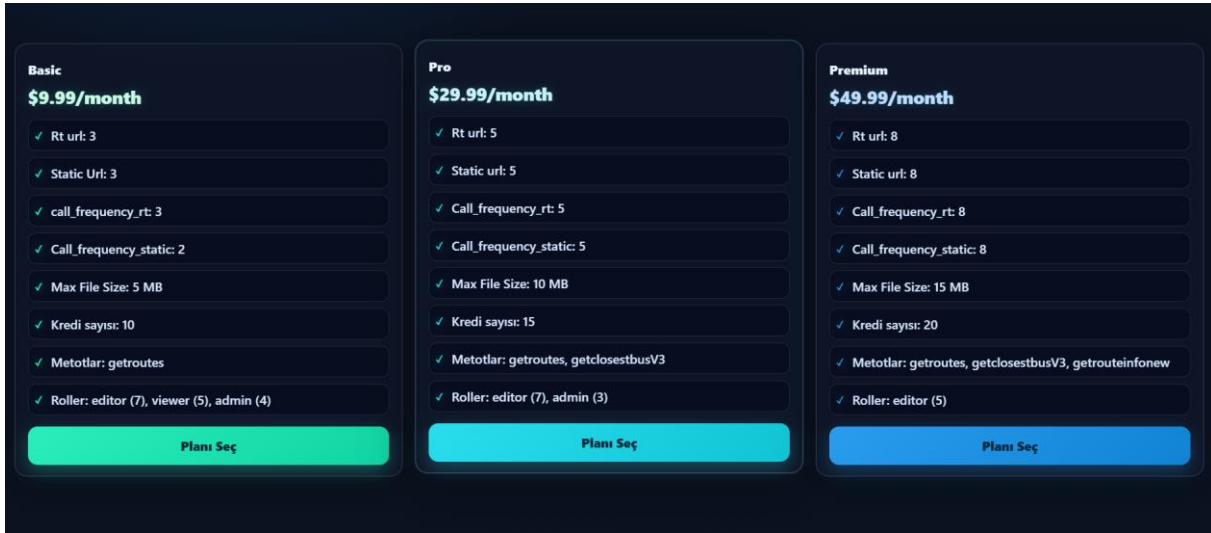
Anasayfa

Ana sayfa, kullanıcıların siteyi hızlıca anlamasına ve gezinmesine yardımcı olacak şekilde tasarlanmıştır. Üst kısımda bir navigasyon çubuğu bulunuyor, burada ‘Biz Kimiz’, ‘Planlar’ ve ‘Giriş Yap’ veya giriş yapıldıysa ‘Ayarlar’ ‘Profil’ gibi bölümlere kolayca ulaşılabilir. Sayfanın orta kısmında planları inceleme butonu sergileniyor, alt kısımda ise otobüs animasyonu görünüyor.



Planlar

Planlar sayfası, kullanıcıların farklı abonelik planlarını karşılaştırıp seçebileceği, her planın sunduğu kredi limitleri ve özelliklerin net biçimde gösterildiği bir arayüzdür. Plan kartları, kullanıcıların hangi planın kendilerine uygun olduğunu kolayca anlayabilmeleri için fiyat, kredi limiti ve ek avantajları detaylı şekilde sunar. Kullanıcılar seçtikleri plana hızlıca geçiş yapabilir, seçim butonları ve görsel vurgu sayesinde deneyim akıcı ve sezgiseldir. Sayfa, responsive tasarımı ile tüm cihazlarda uyumlu çalışır ve kullanıcıların satın alma kararlarını desteklemek üzere sade, anlaşılır ve güven verici bir yapıdadır.



Ayarlar

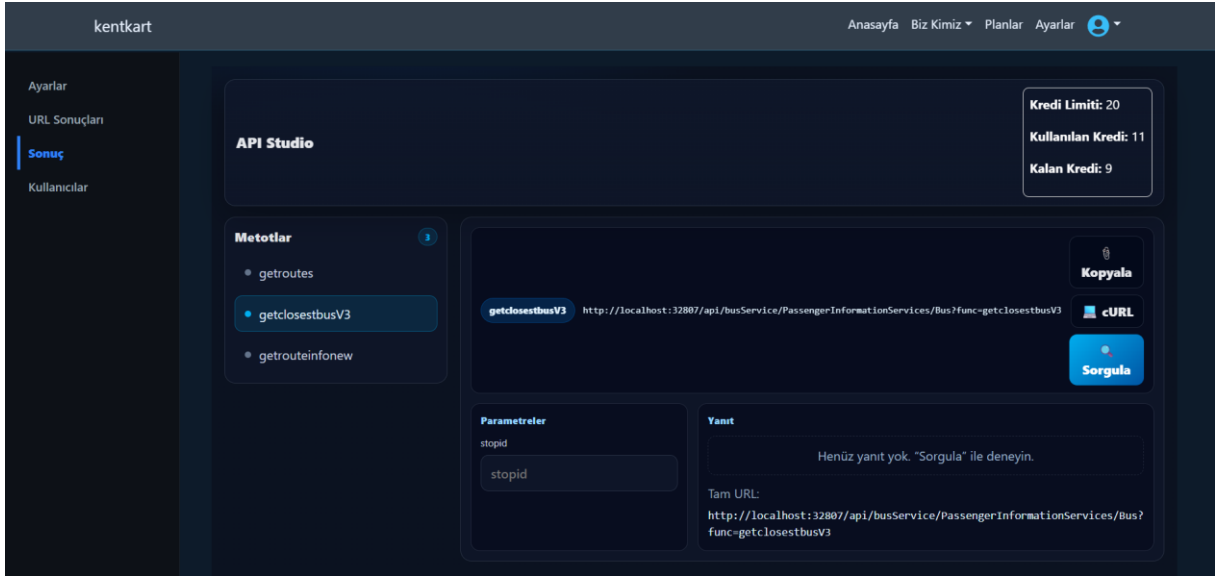
Ayarlar sayfası, kullanıcının hesap yönetimini kolaylaştıran merkezi bir paneldir. Burada kullanıcılar API anahtarlarını görebilir veya yenileyebilir, kendi URL'lerini sisteme ekleyebilirler. Ayrıca, mevcut abonelik planlarının kredi limitlerini ve URL kullanım sınırlarını açık şekilde görüntüleyebilirler. Tüm bu bilgiler, kullanıcının planına ve kullanım durumuna göre dinamik olarak güncellenir. Kullanıcı dostu arayüz sayesinde, ayarların yönetimi hızlı ve sorunsuz gerçekleşir, böylece kullanıcılar hizmetlerini verimli şekilde kontrol edebilir.

Kullanıcının verdiği url'ler mock ile çağırılıyor ve hata varsa durumunu grid yapısında görebiliyor.

Başlık	URL	Çağrı Süresi (ms)	Durum	Hata Detayı	Kontrol Zamanı
RT URL 1	https://example.com/rt1	150	✓ Başarılı	-	05.08.2025 15:00:00
Static URL 1	https://example.com/static1	320	✗ Hata	Timeout	05.08.2025 15:05:00

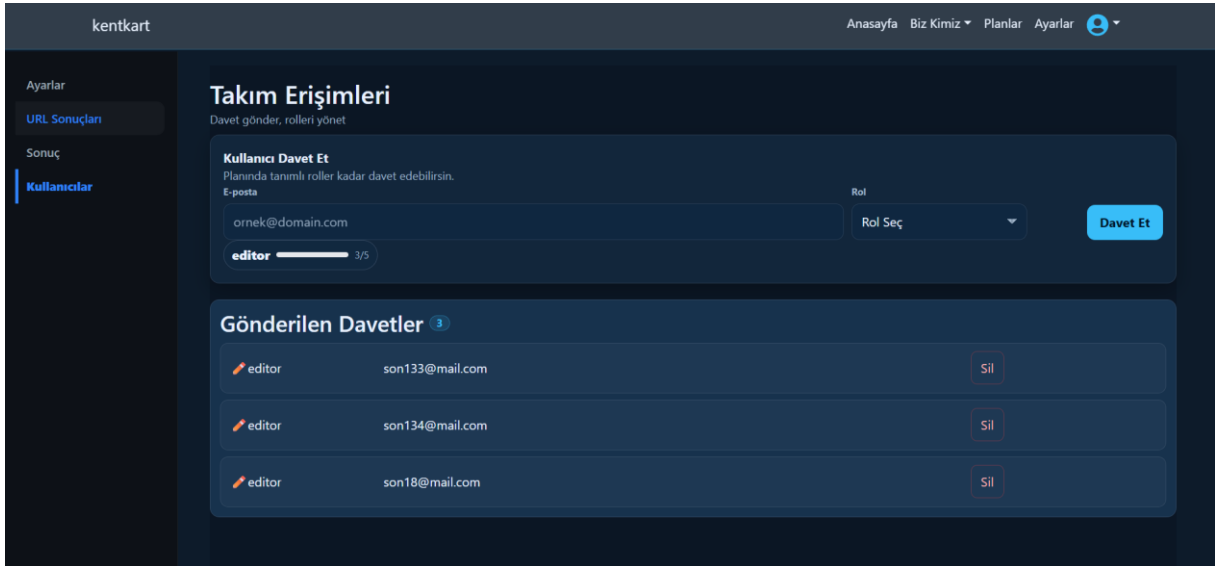
Rows per page: 5 1-2 of 2

İç sistem olarak kullanıcıya planının belirlediği sınırlar dahilinde erişim linki veriliyor. Postman mantığında çalışan bu sistemde sorgula butonuna basınca response'u ,kopyala ve curl butonuna basınca kopyalanması ve gelen isteklerin kontrol edilmesi mümkün kılınıyor.

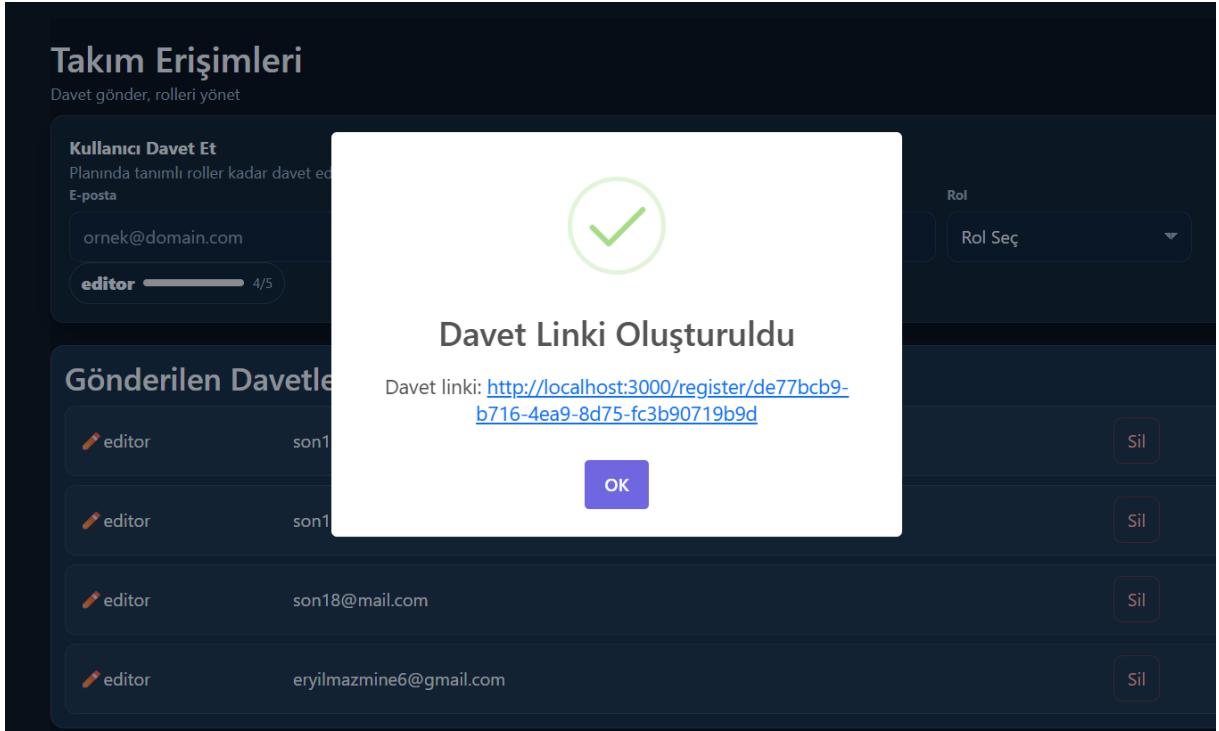


Planın belirlediği rol sınırlarına göre mail atılarak kullanıcı davet ediliyor.

Davet edilme:



Davet linkinin oluşturulması:



Kullanıcının linke tıklayarak kayıt olması:

Davet Edilerek Kayıt Oluyorsunuz!

Atanan Rol: *editor*

Kayıt Ol

İsim

Soyisim

eryilmazmine6@gmail.com

řifre

Kayıt ol

Zaten kayıtlı mısınız? [Giriř yap](#)

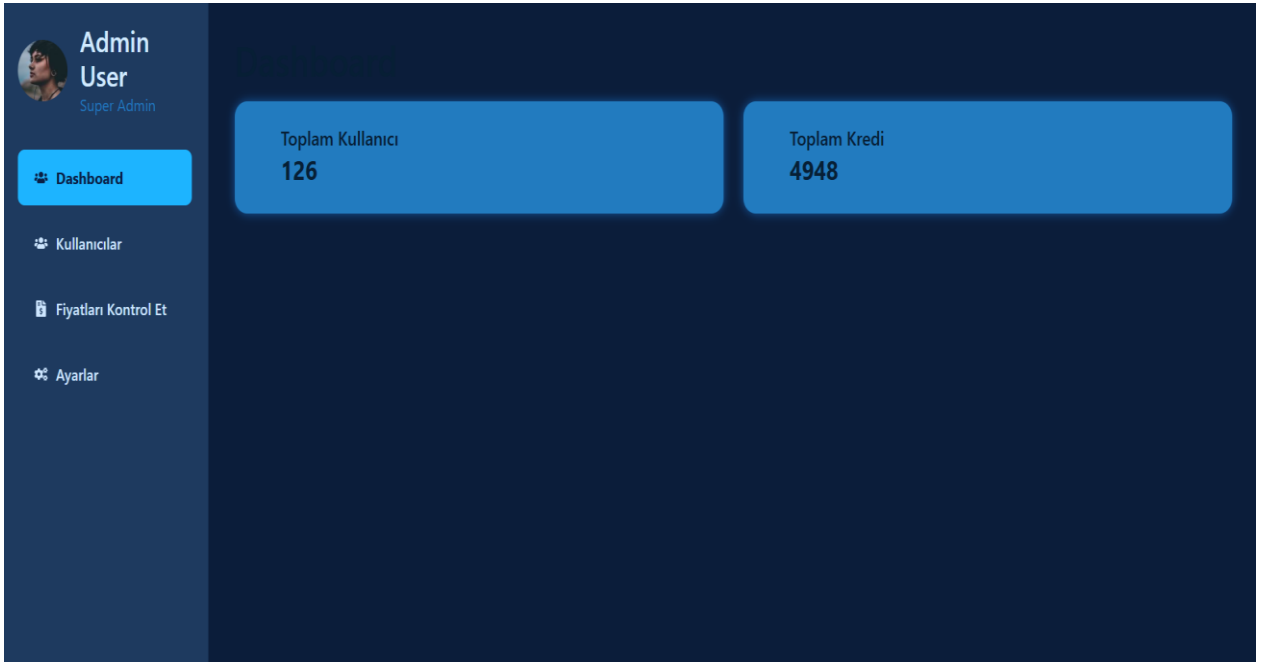
Kullanıcı rolüne göre kısıtlamalar getirilmiştir.Admin olan kullanıcı her ayarı değiştirebilirken,editör ve viewer için ayrı kısıtlamalar getirilmiştir.

Editör tüm ayarlara erişemez:




Admin Paneli

Admin kendi panelinde toplam kullanıcı sayısı ve kredi sayısını görebilir:



Kullanıcıları görebilir ve seçili kullanıcının ayarlarına gidebilir:



Admin User
Super Admin

- Dashboard
- Kullanıcılar**
- Fiyatları Kontrol Et
- Ayarlar

[+ Kullanıcı Ekle](#)

Kullanıcı Listesi

Seçili: 0 Toplam: 228

[AKTİF KULLANICILAR](#) [SILINEN KULLANICILAR](#) [SEÇİLENLERİ SİL](#) [KULLANICI AYARLARINA GİT](#)

<input type="checkbox"/>	ID	Email	Plan	Son Giriş	Oluşturulma Tarihi	Kalan Kredi
<input type="checkbox"/>	4	eryilmazmine6@gmail.com		08.48 18.08.2025	16.20 02.07.2025	0
<input type="checkbox"/>	39	eeryilmazmine6@gmail.com	Pro	13.45 09.07.2025	13.45 09.07.2025	15
<input type="checkbox"/>	48	mne@gmail.com	Premium	14.35 09.07.2025	14.34 09.07.2025	20
<input type="checkbox"/>	55	ornek@gmail.com	Premium	15.25 10.07.2025	15.24 10.07.2025	20
<input type="checkbox"/>	69	ornek10@gmail.com	Pro	13.55 11.07.2025	13.55 11.07.2025	15
<input type="checkbox"/>	72	k@gmail.com	Premium	09.36 17.07.2025	09.36 17.07.2025	20
<input type="checkbox"/>	73	li@gmail.com	Pro	10.06 17.07.2025	10.06 17.07.2025	15
<input type="checkbox"/>	74	ml@gmail.com	Pro	08.53 18.07.2025	08.26 18.07.2025	15

Rows per page: 10 1-10 of 228

Seçili kullanıcının ayarları:

Ayarlar

- URL Sonuçları
- Sonuç
- Kullanıcılar**

Şu an mne@gmail.com ayarlarını görüntülüyorsunuz.

Kullanıcı Ayarları Superadmin mne@gmail.com

[DEĞERLERİ KAYDET](#)

Toplam Alan	Zorunlu	Düzenlenebilir
34	34	0

1. Adım

Dakika * number Değer girin

Durak * number Değer girin

Planları kontrol edebilir,değiştirebilir ve yeni plan ekleyebilir:

Planlar

Toplam 3 plan

[+ Yeni Plan](#)
[Fiyatları Kaydet](#)

#208 **Basic** [Sil](#)

Fiyat: 9,99 Maks. Dosya Boyutu (MB): 5 Kredi: 10

Erişim Linkleri / Metotlar

[getroutes](#) [getclosestbusV3](#)

[getrouteinfo](#)

Plan Limitleri [+ Yeni Limit Key](#)

Rt url: 3 Static Url: 3

call_frequency_rt: 3 Call_frequency_static: 2

Roller [+ Rol Ekle](#)

#209 **Pro** [Sil](#)

Fiyat: 29,99 Maks. Dosya Boyutu (MB): 10 Kredi: 15

Erişim Linkleri / Metotlar

[getroutes](#) [getclosestbusV3](#)

[getrouteinfo](#)

Plan Limitleri [+ Yeni Limit Key](#)

Rt url: 5 Static url: 5

Call_frequency_rt: 5 Call_frequency_static: 5

#210 **Premium** [Sil](#)

Fiyat: 49,99 Maks. Dosya Boyutu (MB): 15 Kredi: 20

Erişim Linkleri / Metotlar

[getroutes](#) [getclosestbusV3](#)

[getrouteinfo](#)

Plan Limitleri [+ Yeni Limit Key](#)

Rt url: 8 Static url: 8

Call_frequency_rt: 8 Call_frequency_static: 8

Key değerlerini düzenleyebilir:

Admin — Key & Tip Yönetimi

Sistem üzerinde kullanılacak key'leri ekleyin, güncelleyin veya kaldırın.

[+ Yeni Key](#)

Zorunlu mu?	Key	Tip	Tekrarlanabilir mi?	Açıklama	İşlem
Evet	Dakika	number	Hayır	Dakika bilgisini gösterir.	Sil
Evet	Rt Url	string	Evet	Rt url bilgisini gösterir.	Sil
Evet	Static Url	string	Evet	Static url bilgisini gösterir.	Sil
Evet	Durak	number	Hayır	Durak sayısını gösterir.	Sil
Evet	Call_frequency_rt	number	Evet	fhkcfyk	Sil
Evet	Call_frequency_static	number	Evet	canım öyle istedi	Sil

Rows per page: 100 1-6 of 6

Test Senaryoları

Frontendde ve backendde bulunan fonksiyonlar için toplam 20 adet test dosyası yazıldı.

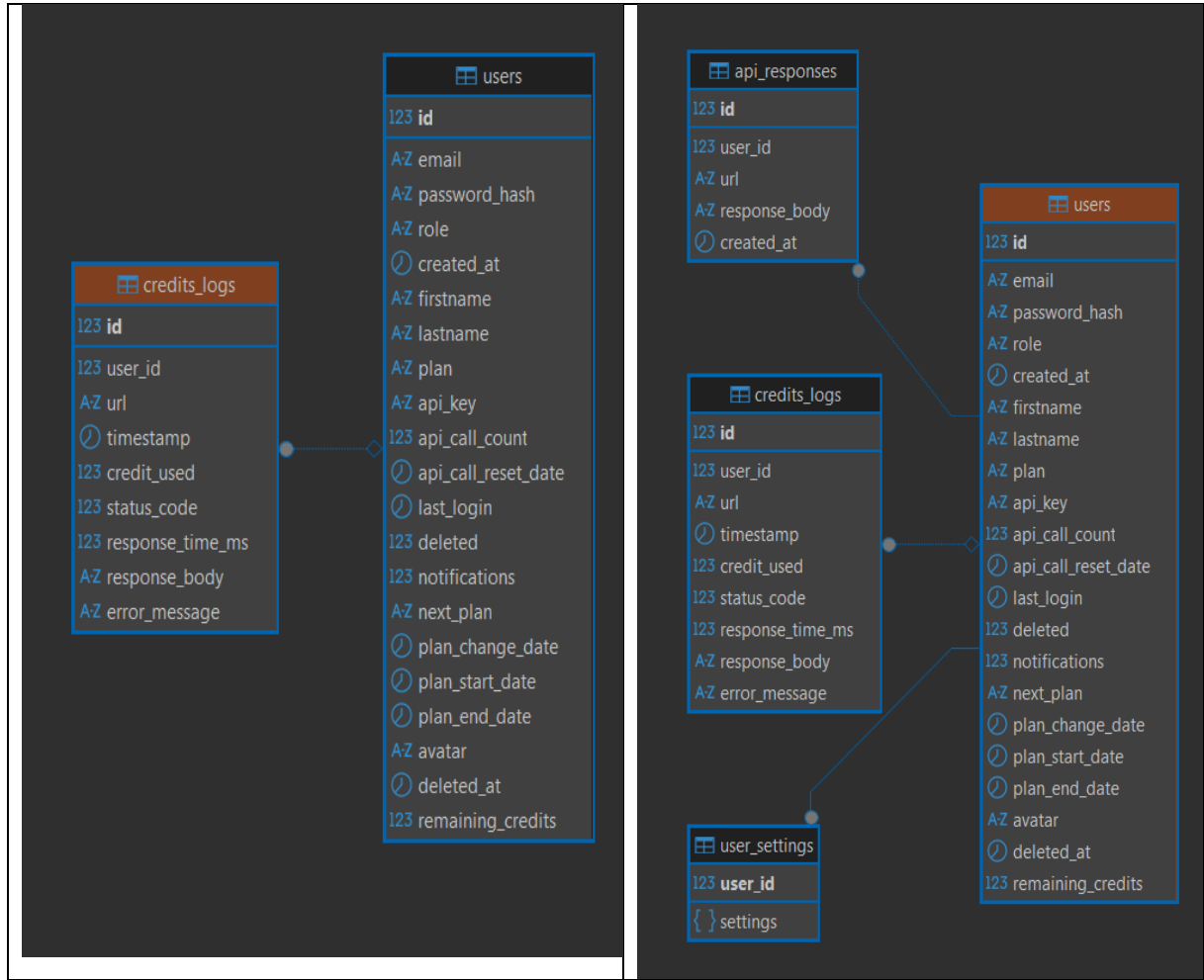
<pre> PASS __tests__/settings.test.js Kullanıcı Ayarları Testleri GET /api/settingkey ✓ ayarları başarıyla getirildi (1 ms) ✓ veritabanı hatası varsa 404 döner (6 ms) POST /api/settingkey ✓ ayarları başarıyla kaydedildi (1 ms) ✓ veritabanı hatası varsa 404 döner (6 ms) Test Suites: 1 passed, 1 total Tests: 4 passed, 4 total Snapshots: 0 total Time: 0.927 s, estimated 1 s </pre>	<pre> PASS src/_tests_/SettingTab.test.js SettingTab Component ✓ İlk yüklemde fetchKeys çağrılır (1 ms) ✓ Yeni key ekleme modalı açılıp kapanır (1 ms) ✓ Boş key eklemeye çalışınca alert çıkar (1 ms) ✓ Başarılı yeni key ekleme isteği (1 ms) Test Suites: 1 passed, 1 total Tests: 4 passed, 4 total Snapshots: 0 total Time: 2.936 s, estimated 3 s </pre>	<pre> PASS src/_tests_/Login.test.js Login bileşeni ✓ Tüm form elemanları ekranda (1 ms) ✓ Boş form gönderiminde Swal.fire çıkar (1 ms) ✓ Geçerli bilgilerle giriş çağrılır (1 ms) ✓ Geçersiz bilgilerle Swal.fire çıkar (1 ms) Test Suites: 1 passed, 1 total Tests: 4 passed, 4 total Snapshots: 0 total Time: 5.973 s </pre>	<pre> PASS __tests__/register.test.js POST /api/register/add-user ✓ Tüm alanlar eksiksiz olunca kullanıcı kaydedilir (1 ms) ✓ Eksik alan varsa 400 döner (6 ms) GET /api/register/list-users ✓ Kullanıcı listesi döner (8 ms) Test Suites: 1 passed, 1 total Tests: 3 passed, 3 total Snapshots: 0 total Time: 1.635 s </pre>
--	--	---	---

Teknoloji Yığını

Katman	Teknoloji
Frontend	React.js
Backend	Node.js
Veritabanı	Mysql

Veritabanı Tabloları

Oluşturduğum tabloların diyagramları:



Fonksiyonel Gereksinimler

Fonksiyonel gereksinimler, sistemin kullanıcılar için sunduğu işlevleri ve özellikleri kapsar.

Kullanıcı Kaydı ve Girişi

- Kullanıcılar e-posta ve şifre ile kayıt olabilir ve sisteme giriş yapabilir.
- Davet linki ile kayıt olan kullanıcılar, ödeme sayfasına yönlendirilmeden doğrudan panele erişebilir.

Abonelik Planı Yönetimi

- Kullanıcılar Basic, Pro gibi planlar arasında seçim yapabilir.
- Her planın kredi ve URL sınırları bulunmaktadır.
- Plan bilgileri kullanıcı profiline yansıtılır.

API Anahtarı Oluşturma ve Görüntüleme

- Kullanıcılar kendilerine özel API anahtarlarını görebilir veya yenileyebilir.

URL Ekleme ve Yönetme

- Kullanıcılar sistemlerine URL tanımlayabilir.
- Planlarına göre belirlenen sayıda URL ekleyebilirler.

Kredi ve Limit Takibi

- Kullanıcılar mevcut kredi miktarını ve kalan URL hakkını görebilir.

Admin Paneli

- Admin, kullanıcıları, planları ve sistem ayarlarını görüntüleyip yönetebilir.

Kullanıcı Ayarları Sayfası

- Kullanıcı kendi bilgilerini ,profil fotoğrafını görüntüleyebilir.

Fonksiyonel Olmayan Gereksinimler

Fonksiyonel olmayan gereksinimler, sistemin performansı, güvenliği, kullanılabilirliği ve diğer teknik özellikleriyle ilgilidir.

Kullanılabilirlik (Usability)

Arayüz sade, sezgisel ve kullanıcı dostudur.

Responsive tasarım sayesinde tüm cihazlarla uyumludur.

Performans

Sayfalar hızlı yüklenir, API istekleri düşük gecikme süresiyle çalışır.

Yüksek trafikte performans düşüşü yaşanmaz.

Güvenlik

Kullanıcı girişleri JWT ile korunur.

Şifreler hashlenmiş şekilde veritabanında tutulur.

API anahtarı yalnızca yetkili kullanıcı tarafından görüntülenebilir.

Erişilebilirlik

Temel erişilebilirlik kurallarına uyumlu olarak geliştirilmiştir.

Bakım ve Genişletilebilirlik

Kod yapısı modülerdir, yeni özellikler kolayca entegre edilebilir.

Admin paneli ve kullanıcı paneli ayrı bileşenler halinde yapılandırılmıştır.

Uyumluluk

Modern tarayıcılarla tam uyumludur (Chrome, Firefox, Edge vs.).

GitHub Projesi

Tüm kaynak kodları, README belgeleri, Issue kayıtları ve güncellemeler aşağıdaki GitHub reposunda tutulmaktadır:

GitHub Repository:

Frontend:

<https://github.com/Mineerylmaz/SaaSBackOfficeFrontend>

Backend:


<https://github.com/Mineerylmaz/SaaSBackOfficeBackend>

Issue Takip ve Yönetim Süreci

Bu projede GitHub Issue özelliği aktif olarak kullanılmıştır. Yaklaşık 100'e yakın issue açılarak aşağıdaki alanlarda görev ve hata takibi yapılmıştır:

- Frontend geliştirme görevleri (sayfa yapıları, tema, bileşenler)
- Backend görevleri (router yapısı, veritabanı sorguları)
- Hata bildirimleri ve çözüm süreçleri
- Test yapıları

Örnek bir Issue bağlantısı:

 [Issue #75 –Url Sonuçları]

(<https://github.com/Mineerylmaz/SaaSBackOfficeFrontend/issues/75>)

README Dosyaları

Proje yapısı içinde her büyük bileşenin (örneğin `frontend`, `backend`) ayrı README belgeleri bulunmaktadır.

Bu belgelerde:

- Kurulum adımları
- Kullanılan teknolojiler
- Komut listeleri (npm start, npm run dev vs.)
- Ortam değişkenleri
- API uç noktaları

...gibi bilgiler yer almaktadır.

Örnek:

- [Frontend README](https://github.com/kullaniciadi/proje-adi/tree/main/frontend/README.md)
- [Backend README](https://github.com/kullaniciadi/proje-adi/tree/main/backend/README.md)

