“Kitaplık”

BİL372 Veritabanı Sistemleri Projesi

Son Rapor

Ali Doğan Çamur 231101017

Ege Azca 231101057

Yankı Yağız Ozan 231101059

0. GitHub Linki: https://github.com/chivenos/Kitaplik

1. Domain Gerçek Dünya Probleminin Tanımı

Projemiz, insanların sahip olduğu 2. el kitapları tekrar kullanışlı hale getirmeyi hedeflemektedir. Kampüsümüzde pek çok öğrencinin, okudukları veya artık kullanmadıkları ders kitaplarını yurt odalarında ya da evlerinde beklettiğini gözlemliyoruz. Bu durum aynı kitaplara ihtiyaç duyan diğer öğrencilerin bu kaynaklara erişmek için yüksek ücretler ödemek zorunda kalmasına neden olmaktadır. Aynı zamanda basılı kitaplara erişimi kısıtlı olan veya her dönem yeni kitap alacak bütçesi olmayan öğrencilerimiz için de ciddi zorluklar yaratabilmektedir.

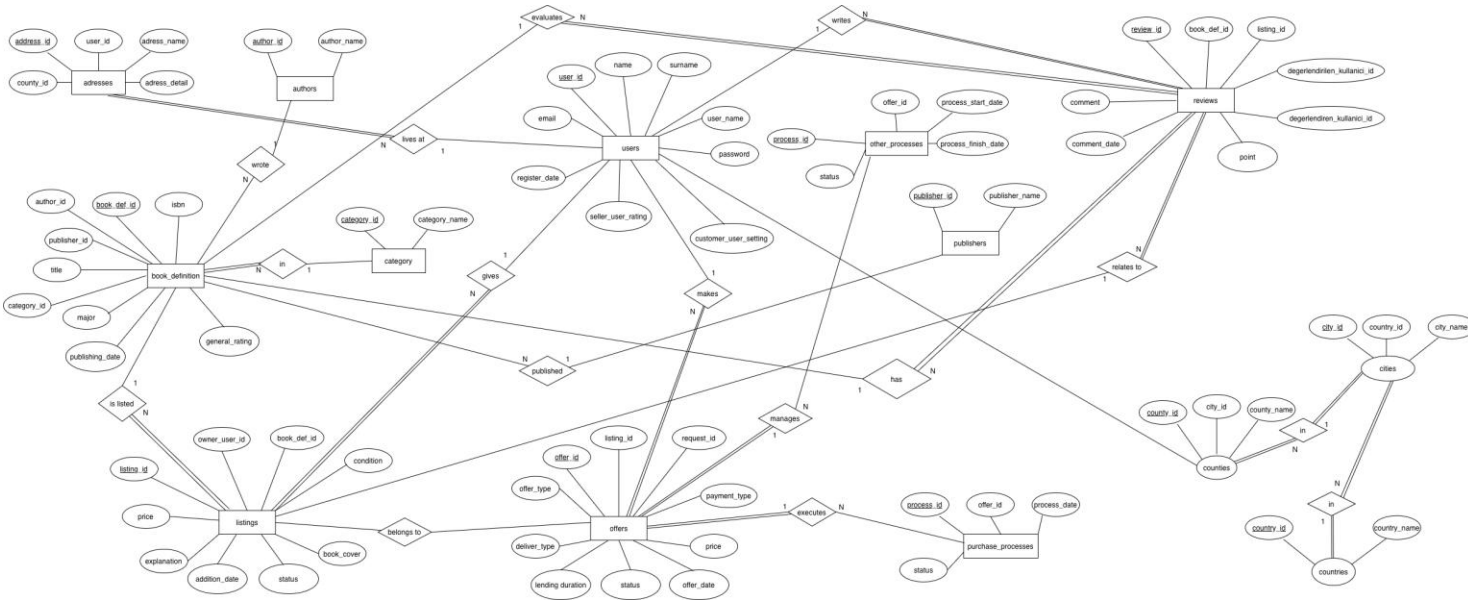
Platformumuz, 2. el kitap satın alım ve kiralama platformudur. Kullanıcılar bu platform üzerinden ihtiyaçları olan kitabı aratabilecek ve dış piyasaya kıyasla çok daha uyguna diğer kullanıcıların 2. el kitaplarını satın alabilecek veya bir süreliğine kiralayabileceklerdir. Sisteme yüklenen kitapların kullanılma derecelerine, ilk yüklenme fiyatına göre göre alıcı satıcıya teklifler sunabilecektir.

2. Gereksinim Analizi

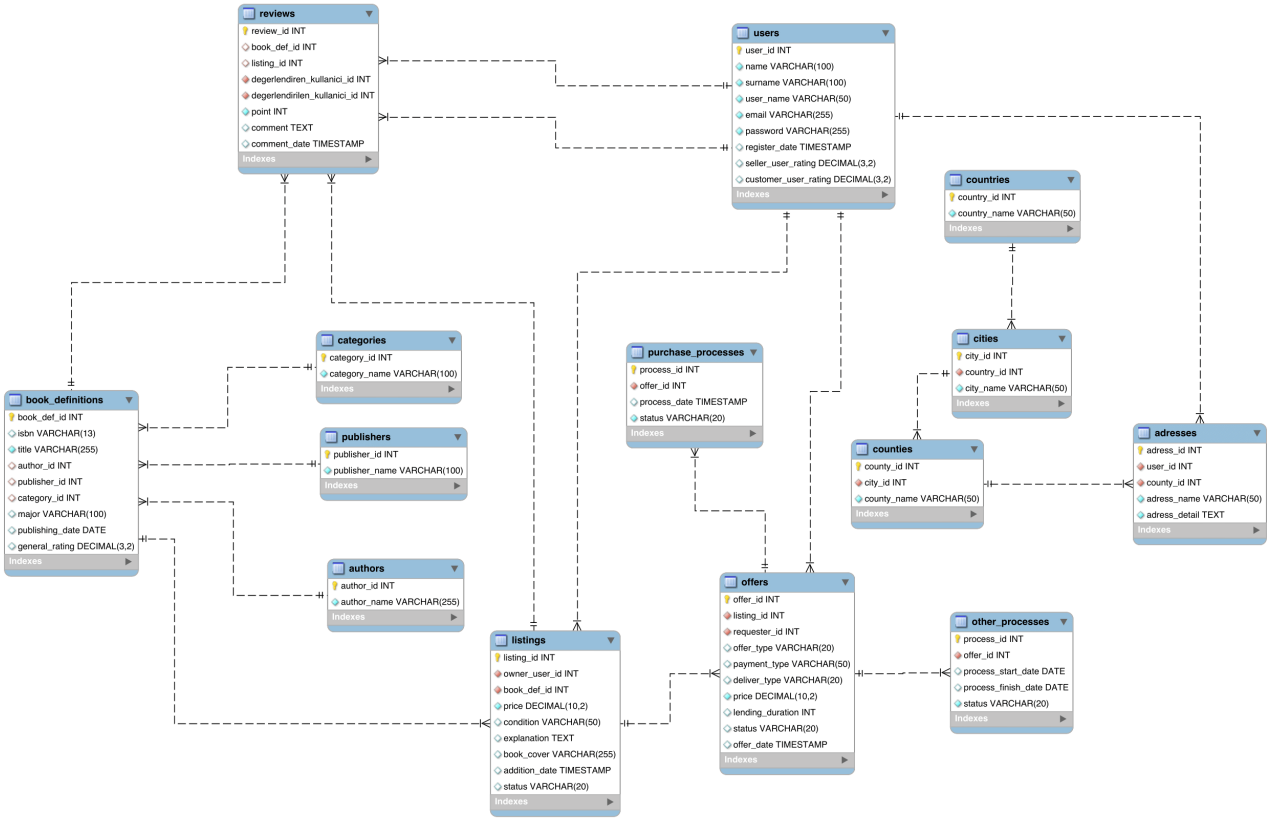
Sistemin temel işlevsel gereksinimleri şunlardır:

- **Kullanıcı Yönetimi:** Öğrenciler sisteme kaydolabilmeli, giriş yapabilmeli ve profil bilgilerini (adres vb.) yönetebilmelidir.
- **Katalog Yapısı:** Kitaplar bir kez tanımlanmalı (ISBN, Yazar, Yayınevi), kullanıcılar bu tanımlar üzerinden ilan açmalıdır.
- **İlan Yönetimi:** Kullanıcılar ellerindeki kitaplar için kondisyon ve fiyat belirterek "Satılık", "Kiralık" veya "Ödünç" ilanı açabilmelidir.
- **Teklif Sistemi:** Alıcılar, ilanlara fiyat teklifi verebilmeli, kiralama süresi talep edebilmelidir.
- **Değerlendirme Sistemi:** İşlem sonunda taraflar birbirini ve kitabı puanlayabilmelidir. Bu puanlar "Satıcı Puanı" ve "Müşteri Puanı" olarak profile yansımalıdır.

3. Kavramsal Tasarım (EER Diyagramı)



4. Mantıksal Tasarım ve Şema Diyagramları



5. Tasarımın Uygulanması

5.1. Kullanılan Teknolojiler ve Frameworkler

Proje 3 Katmanlı Mimari kullanılarak geliştirilmiştir.

1. **Veritabanı (VTYS): MySQL**
2. **Backend: Node.js + Express.js**
3. **Frontend: JavaScript + HTML/CSS**

5.2. Tabloların Oluşturulması

```sql
create table countries (
    country_id int auto_increment primary key,
    country_name varchar(50) not null unique
);
create table cities (
    city_id int auto_increment primary key,
    country_id int not null,
    city_name varchar(50) not null,
    foreign key (country_id) references countries(country_id) on delete cascade
);
create table counties (
    county_id int auto_increment primary key,
    city_id int not null,
    county_name varchar(50) not null,
    foreign key (city_id) references cities(city_id) on delete cascade
);
create table authors (
    author_id int auto_increment primary key,
    author_name varchar(255) not null unique
);

create table publishers (
    publisher_id int auto_increment primary key,
    publisher_name varchar(100) not null unique
);

create table categories (
    category_id int auto_increment primary key,
    category_name varchar(100) not null unique
);
-- 2. kullanicilar tablosu
create table users (
    user_id int auto_increment,
    name varchar(100) not null,
    surname varchar(100) not null,
    user_name varchar(50) not null,
    email varchar(255) not null,
    `password` varchar(255) not null,
    register_date timestamp default current_timestamp,
    seller_user_rating decimal(3, 2) default 0.00,
    customer_user_rating decimal(3, 2) default 0.00,
    primary key (user_id),
    unique (user_name),
    unique (email)
);

-- 3. kitap tanimlari (katalog)
create table book_definitions (
    book_def_id int auto_increment,
    isbn varchar(13) unique,
    title varchar(255) not null,
    author_id int,
    publisher_id int,
    category_id int,
    major varchar(100),
    publishing_date date,
    general_rating decimal(3, 2) default 0.00,
    primary key (book_def_id),
    foreign key (author_id) references authors(author_id),
    foreign key (publisher_id) references publishers(publisher_id),
    foreign key (category_id) references categories(category_id)
);


```

```sql
-- 4. ilanlar
create table listings (
    listing_id int auto_increment,
    owner_user_id int not null,
    book_def_id int not null,
    `condition` varchar(50),
    explanation text,
    book_cover varchar(255),
    addition_date timestamp default current_timestamp,
    status varchar(20) default 'Yayinda',
    price decimal(10, 2) not null default 0.00,
    primary key (listing_id),
    foreign key (owner_user_id) references users(user_id) on delete cascade,
    foreign key (book_def_id) references book_definitions(book_def_id)
);
-- 5. teklifler
create table offers (
    offer_id int auto_increment,
    listing_id int not null,
    requester_id int not null,
    offer_type varchar(20),
    payment_type varchar(50),
    deliver_type varchar(20),
    price decimal(10, 2) not null default 0.00,
    lending_duration int,
    status varchar(20) default 'Aktif',
    offer_date timestamp default current_timestamp,
    primary key (offer_id),
    foreign key (listing_id) references listings(listing_id) on delete cascade,
    foreign key (requester_id) references users(user_id) on delete cascade
);

-- 6. islem tablolari
create table purchase_processes (
    process_id int auto_increment,
    offer_id int not null,
    process_date timestamp default current_timestamp on update current_timestamp,
    status varchar(20) not null default 'Talep Edildi',
    primary key (process_id),
    unique (offer_id),
    foreign key (offer_id) references offers(offer_id) on delete cascade
);

create table other_processes (
    process_id int auto_increment,
    offer_id int not null,
    process_start_date date,
    process_finish_date date,
    status varchar(20) not null default 'Talep Edildi',
    primary key (process_id),
    unique (offer_id),
    foreign key (offer_id) references offers(offer_id) on delete cascade
);

-- 7. adresler
create table adresses (
    adress_id int auto_increment,
    user_id int not null,
    county_id int not null,
    adress_name varchar(50) not null,
    adress_detail text not null,
    primary key (adress_id),
    foreign key (user_id) references users(user_id) on delete cascade,
    foreign key (county_id) references counties(county_id)
);

-- 8. degerlendirmeler
create table reviews (
    review_id int auto_increment,
    book_def_id int,
    listing_id int,
    degerlendiren_kullanici_id int not null,
    degerlendirilen_kullanici_id int not null,
    point int not null check (point >= 1 and point <= 5),
    `comment` text,
    comment_date timestamp default current_timestamp,
    primary key (review_id),
    foreign key (book_def_id) references book_definitions(book_def_id),
    foreign key (listing_id) references listings(listing_id),
    foreign key (degerlendiren_kullanici_id) references users(user_id),
    foreign key (degerlendirilen_kullanici_id) references users(user_id)
);
```

5.3 Viewlar

```
create view view_catalog_summary as
select bd.title, a.author_name, bd.isbn, bd.general_rating, count(b.listing_id) as total_listings_available
from book_definitions bd
left join listings b on bd.book_def_id = b.book_def_id and b.status = 'Yayinda'
left join authors a on bd.author_id = a.author_id
group by bd.book_def_id;
```

- **Mantık**: Kitap tanımlarını ve o kitaba ait satışta kaç adet kopya olduğunu listeler.
- **Neden Left Join?** Eğer WHERE kullansaydık, hiç ilanı olmayan kitaplar sonuçtan elenirdi (çünkü status NULL olurdu). Koşulu JOIN içine alarak, "Sadece yayında olan ilanları bağla, eğer yoksa yine de kitabı listele ama ilan kısmını NULL yap" demiş oluyoruz

```
create view view_active_listings as
select b.listing_id, bd.title, u.user_name as seller, b.condition, b.status
from listings b
join book_definitions bd on b.book_def_id = bd.book_def_id
join users u on b.owner_user_id = u.user_id
where b.status = 'Yayinda';
```

- **Mantık:** Sadece şu an satın alınabilir durumdaki ilanları düz bir liste olarak sunar.

```
create view view_full_addresses as
select a.adress_id, u.user_name, a.adress_name, a.adress_detail, cnt.county_name, ci.city_name,
co.country_name
from adresses a
join users u on a.user_id = u.user_id
join counties cnt on a.county_id = cnt.county_id
join cities ci on cnt.city_id = ci.city_id
join countries co on ci.country_id = co.country_id;
```

- **Mantık:** Normalizasyon gereği parçalanmış coğrafi veriyi (county_id -> city_id-> coutnry_id) tek bir okunabilir satıra çevirir.
- **Tasarım Tercihi** (Zincirleme Inner Join)**:** Adres hiyerarşisinin tam olduğunu garanti eder. Üstelik tek tablo gibi de gözükebilir.

```
create view view_active_offers as
select o.offer_id, bd.title, u.user_name as requester, o.offer_type, o.price, o.status
from offers o
join listings b on o.listing_id = b.listing_id
join book_definitions bd on b.book_def_id = bd.book_def_id
join users u on o.requester_id = u.user_id
where o.status = 'Aktif';
```

- **Mantık:** Sadece üzerinde aksiyon alınması gereken (Bekleyen) teklifleri gösterir. Reddedilen veya onaylananlar elenir**.**

5.4 Veri Yükleme

Mock veriler Mockaroo ve kendimiz yazdığımız queryler ile yüklenmiştir.

5.5 İndisler

```
create index idx_book_title on book_definitions(title);
```

- **Amaç:** Kitap ismine göre yapılan aramaları ve sıralamaları hızlandırmak**.**

```
create index idx_book_isbn on book_definitions(isbn);
```

- **Amaç:** ISBN numarası üzerinden yapılan kesin eşleşme sorgularını optimize etmek.

```
create index idx_user_name on users(name, surname);
```

- **Amaç:** Kullanıcıları isim ve soyisim kombinasyonuyla aramak.

```
create index idx_offer_status on offers(status);
```

- **Amaç:** Belirli bir durumdaki (Örn: 'Aktif') teklifleri filtrelemek.

5.6 Sorgu Tasarımları

1. **GET /api/catalog:**

```
SELECT bd.book_def_id, bd.isbn, bd.title, a.author_name, bd.general_rating,
       COUNT(CASE WHEN b.status = 'Yayinda' THEN 1 END) AS total_listings_available
FROM book_definitions bd
LEFT JOIN listings b ON bd.book_def_id = b.book_def_id
LEFT JOIN authors a ON bd.author_id = a.author_id
WHERE ... GROUP BY bd.book_def_id
```

   - ~ **Amaç:** Kütüphane genelindeki kitap tanımlarını listelemek.
   - ~ **Neden Left Join:** Hiç satıcısı olmayan (stokta olmayan) kitapların da arama sonucunda çıkmasını sağlar. INNER JOIN olsaydı sadece satılan kitaplar listelenirdi.

2. **GET /api/listings:**

```
SELECT l.listing_id, bd.title, u.user_name as seller, u.user_id as seller_user_id, l.condition, l.status,
l.price, bd.general_rating
FROM listings l
JOIN book_definitions bd ON l.book_def_id = bd.book_def_id
JOIN users u ON l.owner_user_id = u.user_id
LEFT JOIN authors a ON bd.author_id = a.author_id
WHERE l.status = 'Yayinda'
```

   - ~ **Amaç:** Satıştaki fiziksel ürünleri listelemek.
   - ~ **Neden Inner Join:** Kitap tanımı (bd) veya satıcısı (u) olmayan bir ilan "bozuk veri"dir. Bunların listelenmesini engellemek için INNER JOIN ile veri bütünlüğü zorlandı.

3. **GET /api/listings/:id:**

```
SELECT l.*, bd.title, u.user_name, u.seller_user_rating as seller_user_rating, u.user_id as owner_user_id
FROM listings l
JOIN book_definitions bd ON l.book_def_id = bd.book_def_id
JOIN users u ON l.owner_user_id = u.user_id
WHERE l.listing_id = ?
```

   - ~ **Amaç:** Tek bir ilanın detay sayfasını göstermek.

4. **POST /api/offers:**

```
SELECT l.*, bd.title, u.user_name, u.seller_user_rating as seller_user_rating, u.user_id as owner_user_id
```

   - ~ **Amaç:** İlanın sahibini bulmak.

5. **POST /api/offers/:id/accept:**

```
SELECT l.*, bd.title, u.user_name, u.seller_user_rating as seller_user_rating, u.user_id as owner_user_id
```

   - ~ **Amaç:** Transaction başlatıldığında, onaylanan teklifin hangi ilana (listing_id) ait olduğunu bulmak.

6. **GET /api/offers/:id:**

```
SELECT o.*, bd.title, u.user_name as requester_name, u.user_id as requester_id
FROM offers o
JOIN listings l ON o.listing_id = l.listing_id
JOIN book_definitions bd ON l.book_def_id = bd.book_def_id
JOIN users u ON o.requester_id = u.user_id
WHERE o.offer_id = ?
```

   - ~ **Amaç:** Bir teklifin özetini göstermek.
   - ~ **Tasarım:** Inner Join zinciri. Teklif -> İlan -> Kitap -> Teklifi Veren Kullanıcı. Bu zincirden biri bile eksikse teklif verisi bozuktur, getirilmez.

7. **GET /api/active-offers:**

```
SELECT * FROM view_active_offers
```

   - ~ **Amaç:** Aktif teklifleri listelemek.

8. **POST /api/login:**

```
SELECT * FROM users WHERE email = ? OR user_name = ?
```

   ~ **Amaç:** Kullanıcıyı bulmak.
   ~ **Tasarım:** OR kullanımı. Kullanıcının hem e-posta hem de kullanıcı adıyla giriş yapabilmesini sağlar.

9. **POST /api/register:**

```
SELECT * FROM users WHERE email = ? OR user_name = ? LIMIT 1
```

   ~ **Amaç:** Email veya kullanıcı adı dolu mu?
   ~ **Tasarım:** *LIMIT 1.* Performans için. İlk eşleşmeyi bulunca aramayı durdurur.

10. **GET /api/user/username/:username:**

```
SELECT * FROM users WHERE user_name = ?
```

   ~ **Amaç:** Kullanıcı adına göre profil bulmak.

11. **GET /api/user/:id:**

```
SELECT * FROM users WHERE user_id = ?
```

   ~ **Amaç:** Kullanıcı idlerine göre profil bulmak.

12. **GET /api/user/:id:**

```
SELECT l.*, bd.title FROM listings l JOIN book_definitions bd ON l.book_def_id = bd.book_def_id WHERE
l.owner_user_id = ?
```

   ~ **Amaç:** Kullanıcının sattığı ürünleri listelemek.

13. **GET /api/user/:id:**

```
SELECT o.*, bd.title, u.user_name as requester_name, u.user_id as requester_id
FROM offers o
JOIN listings l ON o.listing_id = l.listing_id
JOIN book_definitions bd ON l.book_def_id = bd.book_def_id
JOIN users u ON o.requester_id = u.user_id
WHERE o.offer_id = ?
```

   ~ **Amaç:** Bana gelen teklifleri görmek**.**
   ~ **Tasarım:** Teklif bana değil, benim ilanıma gelir. Bu yüzden Teklif -> İlan -> İlan Sahibi (Ben) zinciri kurulmuştur.

14. **GET /api/user/:id/public**

```
SELECT user_id, user_name, name, surname, seller_user_rating, customer_user_rating FROM users WHERE user_id
= ?
```

   ~ **Amaç:** Başkası profili gezerken gösterilecek bilgi.
   ~ **Tasarım:** Özel kolon seçimi (SELECT user_id). Şifre ve email gibi hassas verileri gizlemek için SELECT * kullanılmadı.

15. **GET /api/user/:id/public**

```
SELECT o.*, bd.title, u.user_name as requester_name, u.user_id as requester_id
FROM offers o
JOIN listings l ON o.listing_id = l.listing_id
JOIN book_definitions bd ON l.book_def_id = bd.book_def_id
JOIN users u ON o.requester_id = u.user_id
WHERE o.offer_id = ?
```

   ~ **Amaç:** O kişinin şu an sattığı ürünleri göstermek.

16. **GET /api/user/:id/public**

```
SELECT r.*, u.user_name as reviewer_name, u.user_id as reviewer_id, bd.title as book_title, l.listing_id
FROM reviews r
JOIN users u ON r.degerlendiren_kullanici_id = u.user_id
LEFT JOIN book_definitions bd ON r.book_def_id = bd.book_def_id
LEFT JOIN listings l ON r.listing_id = l.listing_id
WHERE r.degerlendirilen_kullanici_id = ?
ORDER BY r.comment_date DESC
```

- ~ **Amaç:** Kişinin aldığı yorumları göstermek.
- ~ **Neden Left Join:** Eğer yorum yapılan kitap veya ilan silinmişse bile, yorumun (puanın) profilde kalması için.

17. **PUT /api/user/:id**

```
SELECT * FROM users WHERE (email = ? OR user_name = ?) AND user_id != ? LIMIT 1DESC
```

- ~ **Amaç:** Yeni girilen email başkasının mı?

18. **GET /api/user/:id/orders**

```
SELECT o.offer_id, l.listing_id, bd.book_def_id,
       l.owner_user_id AS seller_user_id,
       bd.title AS book_title,
       o.status, o.offer_date AS process_date, o.price
FROM offers o
JOIN listings l ON o.listing_id = l.listing_id
JOIN book_definitions bd ON l.book_def_id = bd.book_def_id
WHERE o.requester_id = ?
ORDER BY o.offer_date DESC
```

- ~ **Amaç:** Bu sorgu, giriş yapmış kullanıcının Alıcı (Requester) rolünde olduğu işlemleri listeler. Kullanıcının profilindeki "Siparişlerim" veya "Yaptığım Teklifler" ekranını dolduran ana sorgudur.
- ~ **Tasarım:** Sistemde "Var olmayan bir ilanın teklifi de geçersizdir" olarak kabul edilmiştir. Eğer bir ilan sistemden fiziksel olarak silinmişse ama kullanıcı o ilana geçmişte teklif vermişse dahi bu listede göremez.

19. **POST /api/books**

```
SELECT * FROM ${table} WHERE ${col} = ? LIMIT 1
```

- ~ **Amaç:** Yazar/Yayınevi/Kategori veritabanında var mı diye bakar (Lookup). Varsa ID'sini döner, yoksa yaratır.

20. **GET /api/user/:id/addresses**

```
SELECT r.*, u.user_name as reviewer_name, u.user_id as reviewer_id, bd.title as book_title, l.listing_id
FROM reviews r
JOIN users u ON r.degerlendiren_kullanici_id = u.user_id
LEFT JOIN book_definitions bd ON r.book_def_id = bd.book_def_id
LEFT JOIN listings l ON r.listing_id = l.listing_id
WHERE r.degerlendirilen_kullanici_id = ?
ORDER BY r.comment_date DESC
```

- ~ **Amaç:** Adres listesini okunabilir formatta getirmek.

21. **GET /api/addresses/:id**

```
SELECT a.*, cnt.county_name, ci.city_name, co.country_name FROM adresses a JOIN counties cnt ... JOIN cities
ci ... JOIN countries co ... WHERE a.adress_id = ?
```

- ~ **Amaç:** Düzenleme ekranı için tek bir adresi getirmek.

22. **POST /api/reviews**

```
SELECT r.*, u.user_name as reviewer_name, u.user_id as reviewer_id, bd.title as book_title, l.listing_id
FROM reviews r
JOIN users u ON r.degerlendiren_kullanici_id = u.user_id
LEFT JOIN book_definitions bd ON r.book_def_id = bd.book_def_id
LEFT JOIN listings l ON r.listing_id = l.listing_id
WHERE r.degerlendirilen_kullanici_id = ?
ORDER BY r.comment_date DESC
```

- ~ **Amaç:** Bu ilana zaten yorum yapmış mı'nın kontrolü.

23. **POST /api/reviews**

```
SELECT * FROM reviews WHERE book_def_id = ? AND degerlendiren_kullanici_id = ? AND listing_id IS NULL LIMIT
1
```

~ **Amaç:** Bu kitaba genel bir yorum yapmış mı?

24. **POST /api/reviews**

```
SELECT AVG(point) AS avgp FROM reviews WHERE book_def_id = ?
```

~ **Amaç:** Kitabın yeni ortalama puanını hesaplamak.

25. **POST /api/reviews**

```
SELECT AVG(point) AS avg_seller FROM reviews WHERE degerlendirilen_kullanici_id = ? AND listing_id IS NOT
NULL
```

~ **Amaç:** Kullanıcının "Satıcı" kimliğiyle aldığı puanların ortalamasını hesaplamak.

26. **POST /api/review**

```
SELECT AVG(point) AS avg_customer FROM reviews WHERE degerlendirilen_kullanici_id = ? AND listing_id IS NULL
```

~ **Amaç:** Kullanıcının "Normal Kullanıcı" (kitap yorumcusu) olarak aldığı puanları hesaplamak.

27. **GET /api/reviews**

```
SELECT * FROM reviews WHERE listing_id = ?
```

~ **Amaç:** Belirli bir satış işleminin yorumlarını getirmek.

28. **GET /api/reviews**

```
SELECT * FROM reviews WHERE book_def_id = ?
```

~ **Amaç:** Belirli bir kitabın yorumlarını getirmek.

29. **GET /api/reviews**

```
SELECT * FROM reviews ORDER BY comment_date DESC LIMIT 100
```

~ **Amaç:** Anasayfa veya akışta son yorumları göstermek.

30. **GET /api/user/:id/reviews**

```
SELECT r.*, u.user_name as reviewer_name, u.user_id as reviewer_id, bd.title as book_title, l.listing_id
FROM reviews r
JOIN users u ON r.degerlendiren_kullanici_id = u.user_id
LEFT JOIN book_definitions bd ON r.book_def_id = bd.book_def_id
LEFT JOIN listings l ON r.listing_id = l.listing_id
WHERE r.degerlendirilen_kullanici_id = ?
ORDER BY r.comment_date DESC
```

~ **Amaç:** Profil sayfasında "Yorumlarım" sekmesi.

~ **Neden Left Join:** Yorumun ait olduğu nesne silinse bile yorumu göstermek için.

31. **GET /api/countries**

```
SELECT * FROM countries
```

~ **Amaç:** Ülke dropdown listesi.

32. **GET /api/cities/:countryId**

```
SELECT * FROM cities WHERE country_id = ?
```

~ **Amaç:** Seçilen ülkeye göre şehirleri getirmek.

33. **GET /api/counties/:cityId**

```
SELECT * FROM cities WHERE city_id = ?
```

~ **Amaç:** Seçilen ülkeye göre ilçeleri getirmek.

34. **GET /api/authors**

```
SELECT * FROM authors ORDER BY author_name
```

~ **Amaç:** Yazar dropdown listesi (Alfabetik).

35. **GET /api/publishers**

```
SELECT * FROM publishers ORDER BY publisher_name
```

~ **Amaç:** Yayınevi dropdown listesi (Alfabetik).

36. **GET /api/categories**

```
SELECT * FROM categories ORDER BY category_name
```

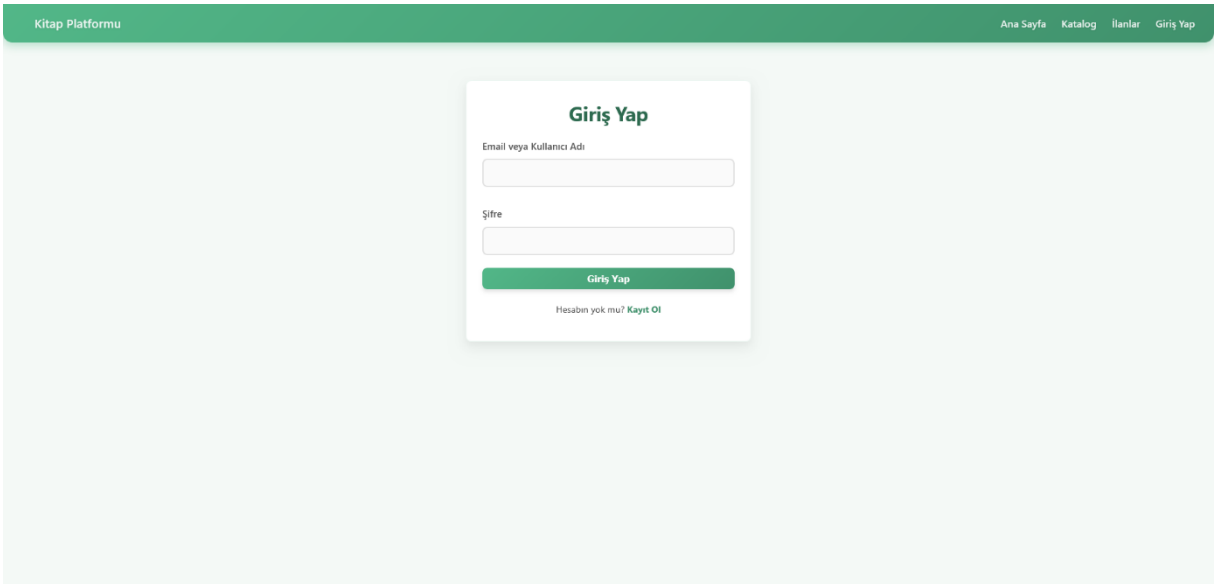
~ **Amaç:** Kategori dropdown listesi (Alfabetik).

37. **GET /api/user/:id/sales**

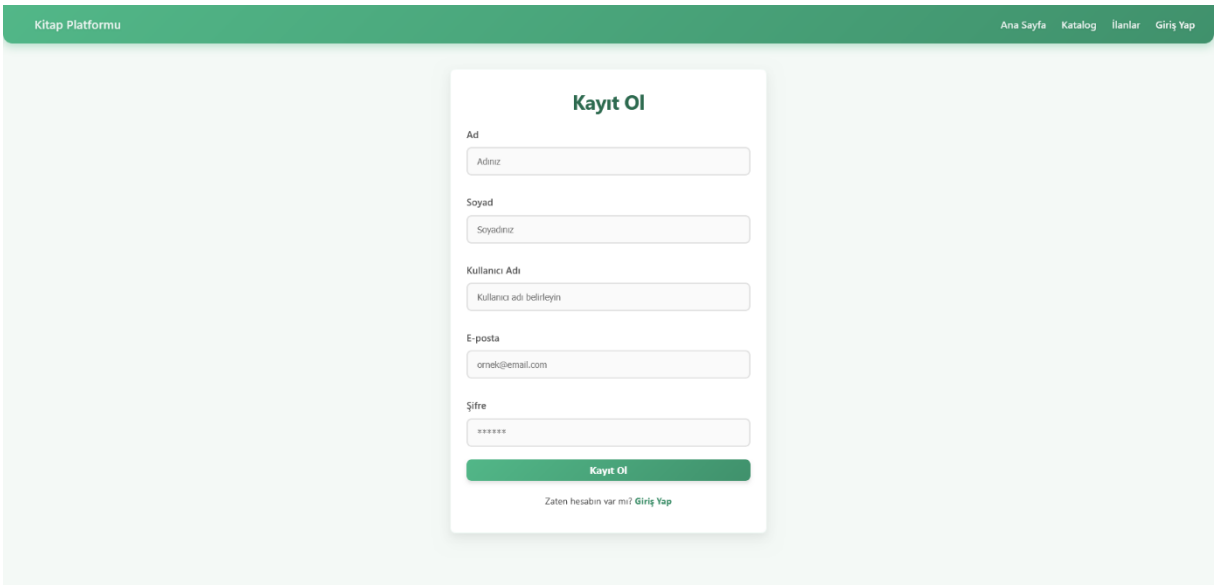
```
SELECT o.offer_id, l.listing_id, bd.book_def_id, bd.title, o.status,
       o.offer_date as process_date, o.price,
       u.user_name as buyer_name, u.user_id as buyer_user_id
FROM offers o
JOIN listings l ON o.listing_id = l.listing_id
JOIN book_definitions bd ON l.book_def_id = bd.book_def_id
JOIN users u ON o.requester_id = u.user_id
WHERE l.owner_user_id = ?
  AND o.status = 'Onaylandı'
  AND o.offer_type LIKE '%sat%'
ORDER BY o.offer_date DESC
```

~ **Amaç:** Bu sorgu, giriş yapmış kullanıcının Satıcı olduğu ve satış işleminin başarıyla tamamlandığı geçmiş kayıtları listeler.

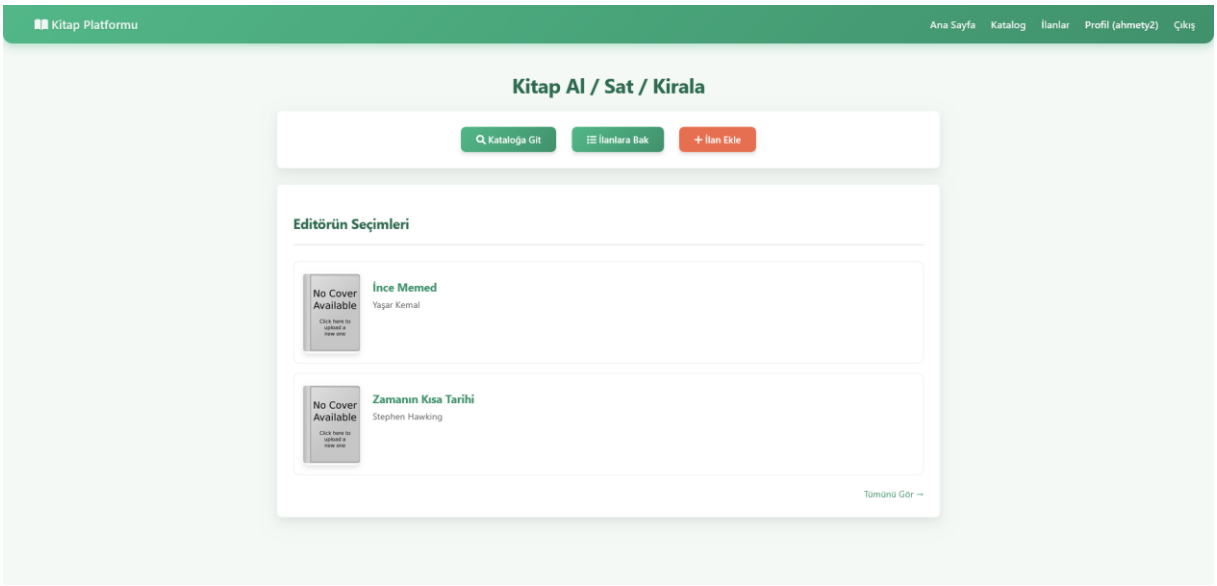
7. Uygulama Program Tanıtımı:



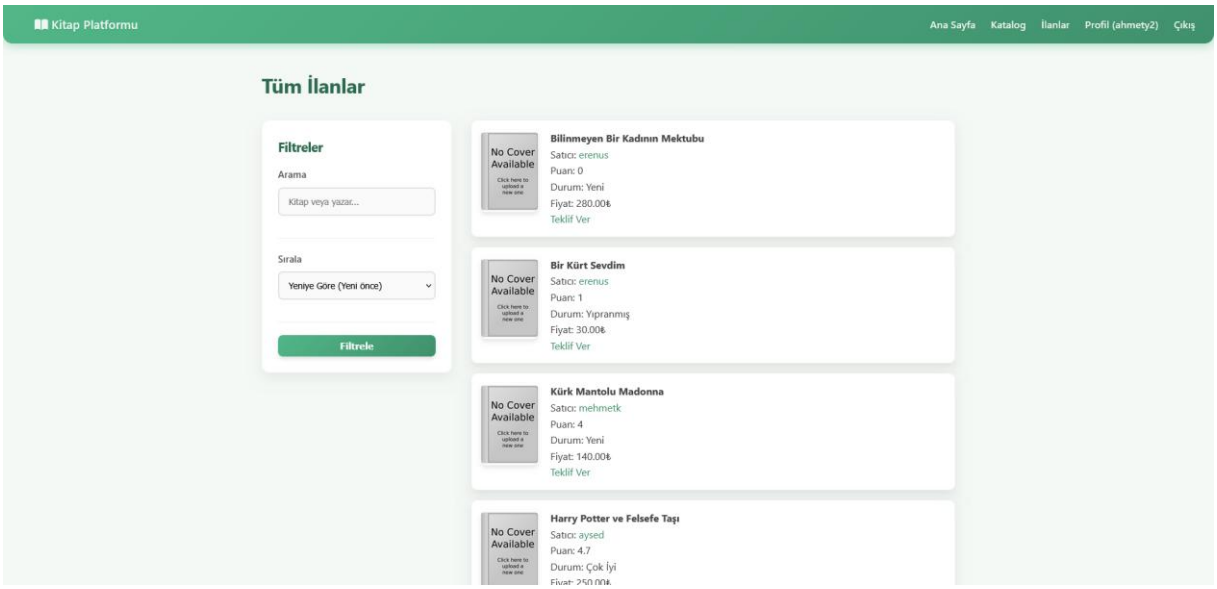
Giriş Ekranı



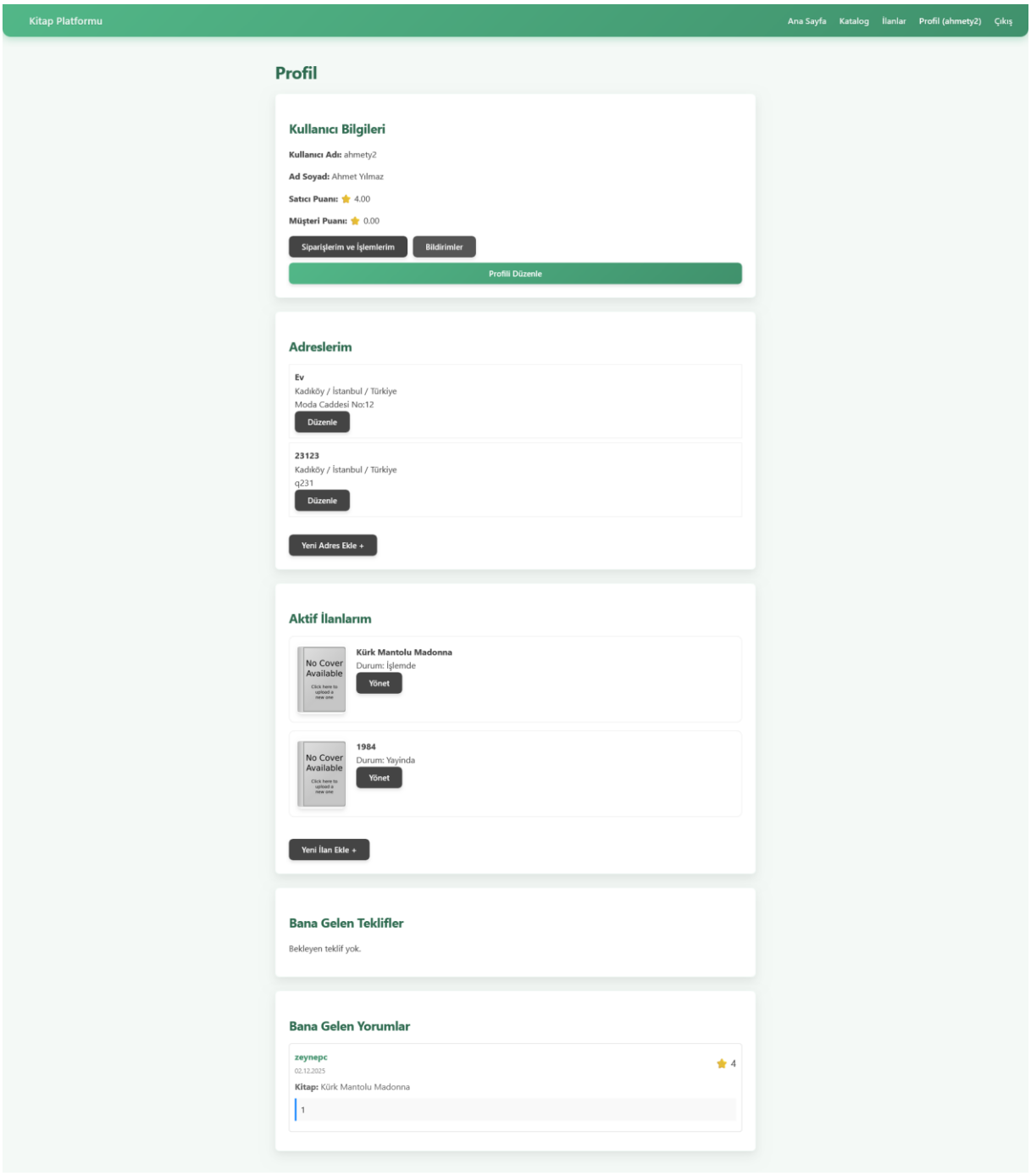
Kayıt Olma Ekranı



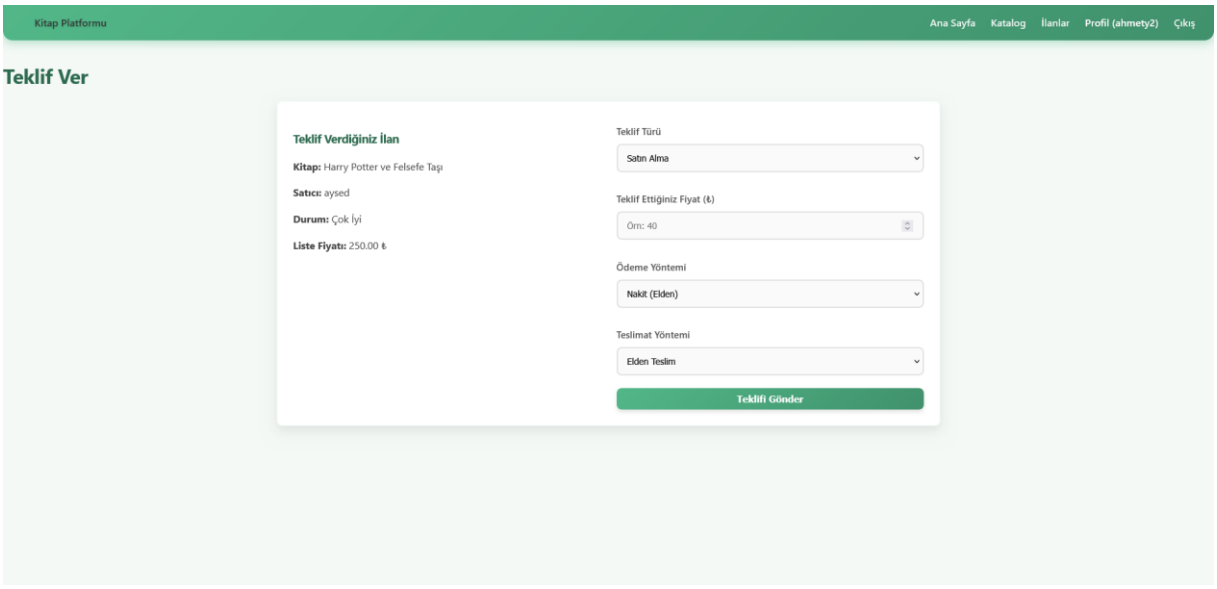
Ana Ekran



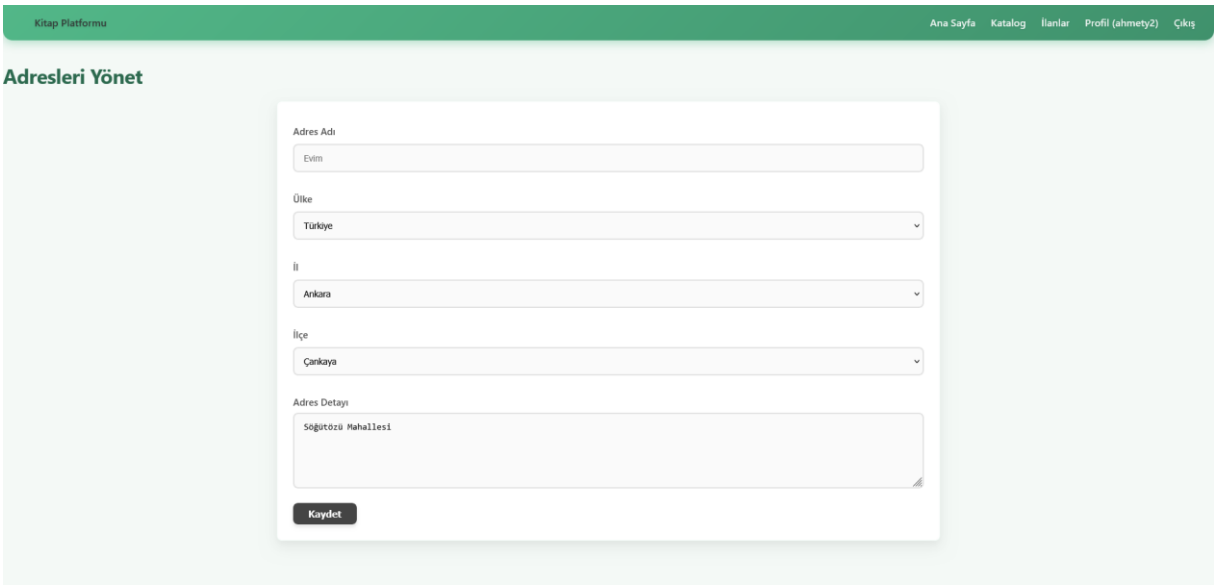
İlanlar



Profil Ekranı



Teklif Verme



Adres Ekleme/Düzenleme Ekranı

7. Sonuç

Bu proje kapsamında, öğrencilerin kitap paylaşım sorununa çözüm getiren, ölçeklenebilir bir web uygulaması geliştirilmiştir. Veritabanı tasarımında 3NF kurallarına sadık kalınmış, performans için indeksleme ve view yapıları kullanılmıştır. Backend tarafında RESTful mimari benimsenmiş,ve gerekli SQL sorgularıyla sistemin istenen isteklere tutarlı ve istendiği şekilde yanıt vermesi sağlanmıştır.