

Mart 2024

# Proje Dokümantasyonu

Django ile İHA Kiralama Projesi Dokümantasyonu

## Önsöz

Bu belge, Baykar şirketinin Web Yazılım Uzmanı / Back-End Developer pozisyonu için sunduğum başvurum kapsamında gerçekleştirdiğim ve sizin değerli incelemenize sunmak için hazırladığım Django ile İHA Kiralama Projesi'nin dokümantasyonunu içermektedir.

Dokümantasyonda projenin genel yapısı hakkında kısa bir özet de sunulmaktadır. Ardından, projenin temel fonksiyonallikleri ve kullanıcıların beklediği özellikler açıklanmaktadır.

Projeyi oluştururken, geliştirme sürecinde karşılaşılan zorlukları aşmak ve en iyi uygulamaları takip etmek için çaba gösterdim. Kullanıcıların ihtiyaçlarına odaklanarak, kullanımı kolay, güvenilir ve esnek bir platform oluşturmayı amaçladım.

Umarım bu dokümantasyon, projenin detaylarını anlamanıza ve değerlendirmenize yardımcı olur.

Saygılarımla,

Müberra BÜLBÜL

## Proje Tanımı ve Amaç

Bu proje, geliştirilmiş bir İnsansız Hava Aracı (İHA) kiralama sistemini temsil etmektedir.

Proje, kullanıcıların İHA kiralama sürecini yönetmelerine olanak tanıyan bir platformun tasarımını ve geliştirilmesini amaçlamaktadır. Bu platform, kullanıcıların kayıt olması, giriş yapması ve kiralama işlemlerini gerçekleştirmesi için gerekli olan temel işlevsellikleri içermektedir.

İHA Kiralama Projesi'nin temel fonksiyonallikleri arasında üyelik yönetimi, İHA'ların eklenmesi, silinmesi, güncellenmesi ve listelenmesi, kiralama işlemlerinin gerçekleştirilmesi, üyelerin kiralama geçmişlerinin takibi ve tüm listeleme sayfalarında filtreleme ve arama özellikleri bulunmaktadır.

## Kullanılan Teknolojiler:

- **Python:** Proje geliştirme sürecinde ana programlama dili olarak tercih edilmiştir. Python'un kolay okunabilirliği ve geniş kütüphane desteği, projenin hızlı bir şekilde geliştirilmesine olanak tanımıştır.
- **Django:** Django, projenin temel altyapısını oluşturan bir Python web framework'üdür. Otomatik URL yönlendirme, veritabanı yönetimi ve güvenlik önlemleri gibi özellikleriyle projenin geliştirilme sürecini optimize etmiştir.
- **PostgreSQL:** PostgreSQL, projenin veritabanı yönetim sistemi olarak seçilmiştir. Güvenilirlik ve performansı ile bilinen PostgreSQL, projenin karmaşık veri yapılarına uygun bir şekilde genişlemesine imkan sağlamıştır.
- **Django Rest Framework (DRF):** DRF, projenin RESTful API'lerini oluşturmak ve yönetmek için kullanılmıştır. Serileştirme işlevleri ve yetkilendirme mekanizmaları gibi özellikleriyle projenin API katmanını etkili bir şekilde geliştirmiştir.
- **Swagger:** Swagger, projenin API'lerinin dökümantasyonunu otomatikleştirmek için entegre edilmiştir. API kullanıcılarına API'leri daha iyi anlamaları için detaylı bilgi sunarak geliştiricilere kolaylık sağlamıştır.
- **Docker:** Docker, projenin konteynerleştirilmesi ve dağıtılmasını sağlamak için kullanılmıştır. Docker konteynerleri, projenin bağımsız birimler halinde çalışmasını sağlayarak, geliştirme ve dağıtım süreçlerini standartlaştırmış ve kolaylaştırmıştır.

Bu teknolojilerin bir araya gelmesi, İHA Kiralama Projesi'nin güvenilir, performanslı ve kullanıcı dostu bir yapıya sahip olmasını sağlamıştır.

## Proje Yapısı ve Modüller

### *Kullanıcı Yönetimi Modülü:*

- Kullanıcıların kayıt olma, giriş yapma ve hesap yönetimi işlemlerini içerir.
- Django'nun kullanıcı yetkilendirme sistemiyle entegre çalışır.

### *İHA Modülü:*

- İHA'ların eklenmesi, silinmesi, güncellenmesi ve listelenmesi gibi işlemleri içerir.
- İHA'ların marka, model, ağırlık, kategori gibi özelliklerini yönetir.

### *Kiralama Modülü:*

- İHA kiralama işlemlerinin yönetimini sağlar.
- Kiralanan İHA'ların kayıtlarını tutar.
- Kiralama işlemlerinin eklenmesi, silinmesi, güncellenmesi ve listelenmesi gibi işlemleri içerir.

### *Veritabanı Modülü:*

- PostgreSQL veritabanı ile etkileşimi sağlar.
- Projenin veri modelini oluşturur ve yönetir.

### *REST API Modülü:*

- Django Rest Framework (DRF) kullanılarak geliştirilmiş RESTful API'leri içerir.
- Kullanıcılar ve istemciler arasında veri alışverişini sağlar.

### *Arayüz Modülü:*

- Kullanıcı arayüzü (frontend) için gerekli olan HTML, CSS dosyalarını içerir.
- Kullanıcıların web arayüzü üzerinden etkileşimde bulunmasını sağlar.

### *Filtreleme ve Arama Modülü:*

- Tüm listeleme sayfalarında kullanılacak olan filtreleme ve arama özelliklerini içerir.
- Kullanıcıların kolaylıkla istedikleri verilere erişmelerini sağlar.

Proje yapısı, modüler bir şekilde tasarlanmıştır, bu da her bir modülün bağımsız olarak geliştirilip test edilebilmesini sağlar. Her bir modül, belirli bir işlevselliği yönetir ve projenin genel yapısını oluşturur. Bu modüler yapı, proje üzerindeki değişiklikleri ve güncellemeleri kolaylaştırır ve bakımını daha yönetilebilir hale getirir.

## Kurulum ve Başlangıç

### *Yerel Makineye Kurulum*

- Bu proje, Python 3.9 sürümünü gerektirir. Bağımlılıklar Poetry aracılığıyla yönetilir.

### *Gereksinimleri Kurma*

- Projeyi çalıştırmak için gereksinimleri kurmak için aşağıdaki adımları izleyin:

```
docker install
```

### *PostgreSQL ve Uygulamayı Yapılandırma*

- Projeyi çalıştırmak için PostgreSQL ve uygulamayı yapılandırmanız gerekir. Docker ve docker-compose kullanarak kolayca yapılandırabilirsiniz:

```
docker compose up -d
```

*Eğer de-anonimleştirilmiş bir veritabanı görüntüsüne erişiminiz yoksa, docker-compose.yml dosyasında postgres:latest kullanmalısınız:*

```
postgres:
  image: postgres:latest
  ...
```

### *Sunucuyu Çalıştırma*

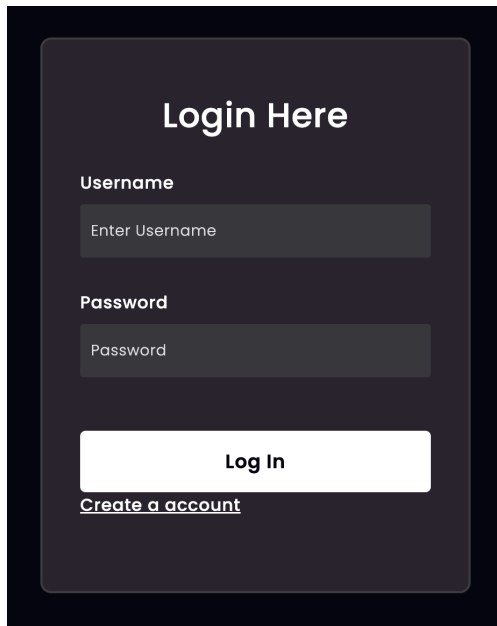
```
docker compose up baykarcase
```

## Kullanım Kılavuzu & API Belgesi

Bu kısım API'lerin temel kullanımını açıklar ve projenin işlevlerinin nasıl kullanıldığı hakkında bilgi verir.

- Proje başlatıldığında öncelikli olarak kullanıcı kayıtlı ise kayıt olması, kayıtlı değilse önce kaydolup ardından giriş yapması gerekmektedir.

*<http://localhost:8000/login/>*

A dark-themed login form titled "Login Here". It features two input fields: "Username" with a placeholder "Enter Username" and "Password" with a placeholder "Password". Below the fields is a white "Log In" button. At the bottom, there is a link that says "Create a account".

**Login Here**

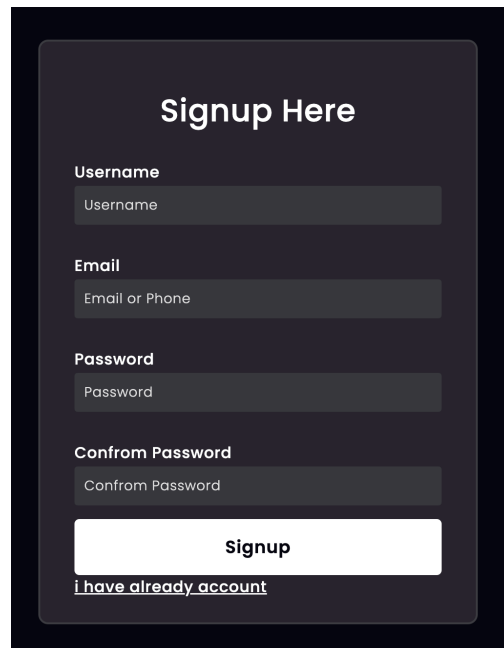
**Username**  
Enter Username

**Password**  
Password

**Log In**

[Create a account](#)

*<http://localhost:8000/register/>*

A dark-themed signup form titled "Signup Here". It features four input fields: "Username" with a placeholder "Username", "Email" with a placeholder "Email or Phone", "Password" with a placeholder "Password", and "Confrom Password" with a placeholder "Confrom Password". Below the fields is a white "Signup" button. At the bottom, there is a link that says "i have already account".

**Signup Here**

**Username**  
Username

**Email**  
Email or Phone

**Password**  
Password

**Confrom Password**  
Confrom Password

**Signup**

[i have already account](#)

Aşağıdaki bölümde proje isterlerinde belirtildiği gibi Ekleme, Silme, Güncelleme, Listeleme işlemlerini gerçekleştiren endpoint' ler listelenmiştir. Bazı endpoint' lerin JSON formatında input alanlarının görselleri de eklenmiştir. Listelenen endpointlerde Filtreleme, Search ve sayfalama özellikleri sağlanmıştır

### register

GET

/register/

register\_list

⌵

🔒

POST

/register/

register\_create

⌵

🔒

📄

POST

/register/

register\_create

⌵

🔒

Parameters

Cancel

Name	Description
<b>data</b> * required	
object	Edit Value   Model
(body)	<pre>{  "username": "user",  "email": "user@example.com",  "password": "string",  "password2": "string"}</pre>

Cancel

Parameter content type

application/json

Execute

login

GET

/login/

login\_list

lock

POST

/login/

login\_create

lock

POST

/login/refresh/

login\_refresh\_create

lock

POST

/login/

login\_create

lock

Parameters

Cancel

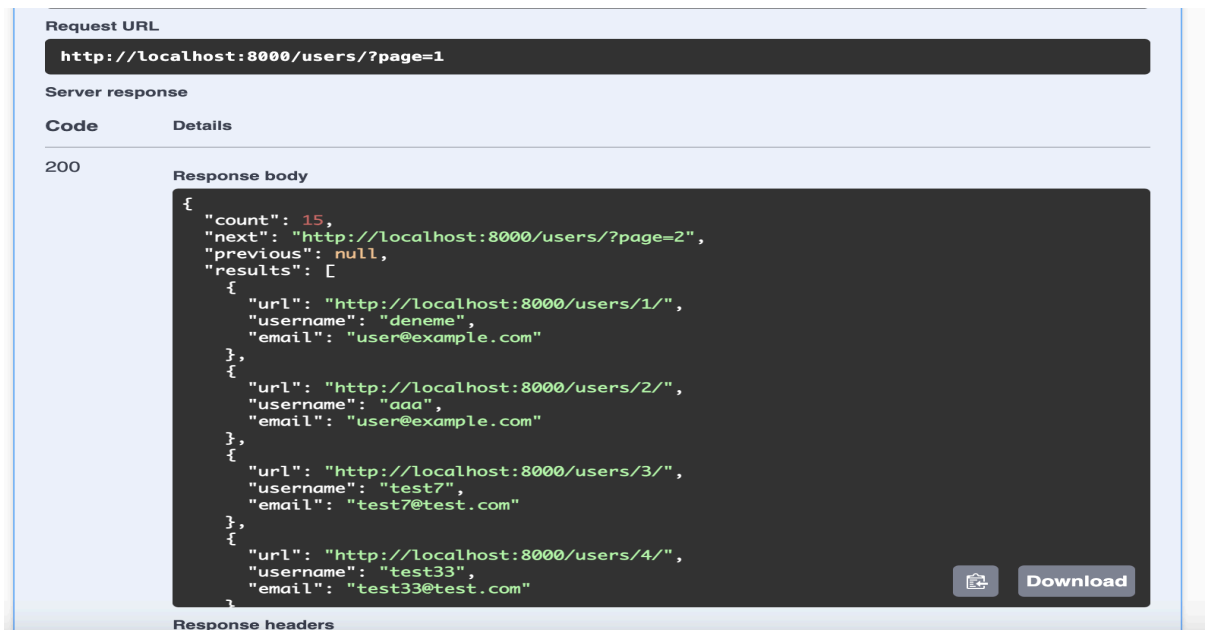
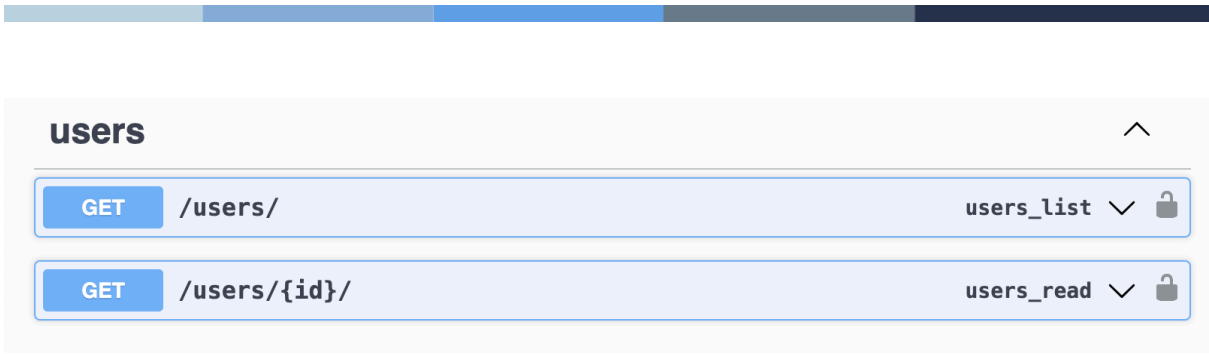
Name	Description
<b>data</b> <small>* required</small> object (body)	<div><div>Edit Value   Model</div><div><pre>{  "username": "string",  "password": "string"}</pre></div></div>

Cancel

Parameter content type  
application/json

Execute





## categories

GET	/categories/	categories_list	✓	🔒
POST	/categories/	categories_create	✓	🔒
GET	/categories/{id}/	categories_read	✓	🔒
PUT	/categories/{id}/	categories_update	✓	🔒
PATCH	/categories/{id}/	categories_partial_update	✓	🔒
DELETE	/categories/{id}/	categories_delete	✓	🔒

POST /categories/

categories\_create ✓ 🔒

This viewset automatically provides **list**, **create**, **retrieve**, **update** and **destroy** actions for staff and 'list', 'retrieve' for any user.

### Parameters

Cancel

Name	Description
------	-------------

**data** ★ required  
object  
(body)

Edit Value | Model

```
{
  "category": "string",
  "text": "string",
  "slug": "L_RlmeNEfILMCKgo-ZVjit2cVAtn5V12khWg0yrpG-GsDrFUQt"
}
```

Cancel

Parameter content type

## categories



GET

/categories/

categories\_list



This viewset automatically provides **list**, **create**, **retrieve**, **update** and **destroy** actions for staff and 'list', 'retrieve' for any user.

### Parameters

Cancel

Name	Description
category	category
string	<input type="text" value="category"/>
(query)	
search	A search term.
string	<input type="text" value="search"/>
(query)	
page	A page number within the paginated result set.
integer	<input type="text" value="page"/>
(query)	

Execute

## ihas

GET /ihas/

ihas\_list ▾ 🔒

POST /ihas/

ihas\_create ▾ 🔒

GET /ihas/{id}/

ihas\_read ▾ 🔒

PUT /ihas/{id}/

ihas\_update ▾ 🔒

PATCH /ihas/{id}/

ihas\_partial\_update ▾ 🔒

DELETE /ihas/{id}/

ihas\_delete ▾ 🔒

POST /ihas/

ihas\_create ▾ 🔒

This viewset automatically provides **list**, **create**, **retrieve**, **update** and **destroy** actions for staff and 'list', 'retrieve' for any user.

### Parameters

Cancel

Name	Description
------	-------------

**data** \* required

object  
(body)

Edit Value | Model

```
{
  "title": "string",
  "brand": "string",
  "category_id": 0,
  "serial_number": "string",
  "model": "string",
  "weight": "string",
  "engine": "string",
  "year": "string",
  "mileage": "string",
  "location": "string",
  "condition": "used",
  "day_price": "string"
}
```

Cancel

ihas

GET

/ihas/

ihas\_list

This viewset automatically provides `list`, `create`, `retrieve`, `update` and `destroy` actions for staff and 'list', 'retrieve' for any user.

Parameters

Try it out

Name	Description
title string (query)	title <input type="text" value="title"/>
brand string (query)	brand <input type="text" value="brand"/>
serial_number string (query)	serial_number <input type="text" value="serial_number"/>
model string (query)	model <input type="text" value="model"/>
category string (query)	category <input type="text" value="category"/>
search string (query)	A search term. <input type="text" value="search"/>
page integer (query)	A page number within the paginated result set. <input type="text" value="page"/>

Server response

Code

Details

200

Response body

```
{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": "c414920c-c2de-4d32-86f5-1c709352bb95",
      "title": "1",
      "url": "http://localhost:8000/ihas/c414920c-c2de-4d32-86f5-1c709352bb95/",
      "brand": "1",
      "serial_number": "1",
      "model": "1",
      "weight": "1.00",
      "engine": "1",
      "year": "1",
      "mileage": "1",
      "location": "1",
      "condition": "new",
      "day_price": "1.00",
      "photos": []
    }
  ]
}
```

Download

## iha-photos



GET	/iha-photos/	iha-photos_list	✓	🔒
POST	/iha-photos/	iha-photos_create	✓	🔒
GET	/iha-photos/{id}/	iha-photos_read	✓	🔒
PUT	/iha-photos/{id}/	iha-photos_update	✓	🔒
PATCH	/iha-photos/{id}/	iha-photos_partial_update	✓	🔒
DELETE	/iha-photos/{id}/	iha-photos_delete	✓	🔒

## rentals



GET	/rentals/	rentals_list	✓	🔒
POST	/rentals/	rentals_create	✓	🔒
GET	/rentals/{id}/	rentals_read	✓	🔒
PUT	/rentals/{id}/	rentals_update	✓	🔒
PATCH	/rentals/{id}/	rentals_partial_update	✓	🔒
DELETE	/rentals/{id}/	rentals_delete	✓	🔒

GET

/rentals/

rentals\_list

This viewset automatically provides `list`, `create`, `retrieve`, `update` and `destroy` actions.  
User have CRUD operations on own objects.  
Staff have CRUD operation on all objects.

Parameters

Try it out

Name	Description
rental_start string (query)	<input type="text" value="rental_start"/>
rental_end string (query)	<input type="text" value="rental_end"/>
created string (query)	<input type="text" value="created"/>
ordering string (query)	<input type="text" value="ordering"/>
page integer (query)	<input type="text" value="page"/>

Request URL

http://localhost:8000/rentals/?page=1

Server response

Code

Details

200

Response body

```
{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
    {
      "url": "http://localhost:8000/rentals/1/",
      "user": 11,
      "rental_start": "2024-03-24",
      "rental_end": "2024-03-24",
      "created": "2024-03-24T04:44:41.157249Z",
      "updated": "2024-03-24T04:44:41.157265Z",
      "rental_duration": "0",
      "total_price": "0.00"
    }
  ]
}
```

Download

PUT

/rentals/{id}/

rentals\_update ^

This viewset automatically provides `list`, `create`, `retrieve`, `update` and `destroy` actions.  
User have CRUD operations on own objects.  
Staff have CRUD operation on all objects.

Parameters

Cancel

Name	Description
<b>data</b> <span>★ required</span> <b>object</b> (body)	<div>Edit Value   Model</div> <pre>{  "iha_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",  "rental_start": "2024-03-24",  "rental_end": "2024-03-24"}  </pre>

Cancel