### **داکیومنت پروژه**

# مستندات پروژه بک‌اند

## مدل‌ها

### UserProfile

* id: شناسه
* username: نام کاربری
* password: رمز عبور
* email: ایمیل
* created\_at: تاریخ ایجاد
* updated\_at: تاریخ به‌روزرسانی

### Task

* id: شناسه
* user: کاربر مرتبط
* title: عنوان
* description: توضیحات
* completed: وضعیت تکمیل
* created\_at: تاریخ ایجاد
* updated\_at: تاریخ به‌روزرسانی

## سریالایزرها

### UserProfileSerializer

* id
* username
* password (فقط درخواستی)
* email
* created\_at
* updated\_at

### TaskSerializer

* id
* user (فقط خواندنی)
* title
* description
* completed
* created\_at
* updated\_at (فقط خواندنی)

## سیگنال‌ها

### create\_super\_user

در این سیگنال، یک کاربر ادمین با ایمیل "admin@example.com" و رمز "admin" ایجاد می‌شود.

## کلاس‌ها

### UserLogin

یک APIView که به کاربر اجازه ورود به سیستم را می‌دهد.

* POST: ارسال نام کاربری و رمز عبور برای احراز هویت و دریافت توکن.

### UserRegistration

یک APIView برای ثبت نام کاربر جدید.

* POST: ارسال اطلاعات کاربر جدید و دریافت توکن.

### TaskList

یک APIView که لیست وظایف مرتبط با کاربر جاری را باز می‌گرداند.

* GET: دریافت لیست وظایف.
* POST: ایجاد یک وظیفه جدید برای کاربر جاری.

### TaskDetail

یک APIView برای جزئیات و کنترل وظایف خاص.

* GET: دریافت جزئیات وظیفه.
* PUT: به‌روزرسانی یک وظیفه.
* DELETE: حذف یک وظیفه.

## URL‌ها

* register/: ثبت نام کاربر جدید
* login/: ورود به سیستم
* tasks/: لیست وظایف
* tasks/<int:pk>/: جزئیات وظیفه با شناسه مشخص

# **داکیومنت فرانت‌اند پروژه**

## صفحه اصلی (index.html)

* هدف:
  + تعریف ساختار اولیه صفحه وب.
* توضیحات:
  + <!DOCTYPE html>: تعریف نوع مستندات HTML.
  + <html lang="en">: تگ HTML با تعیین زبان صفحه.
  + <head>: قسمت اطلاعات هدر صفحه.
    - charset: تعیین مشخصات کاراکتر.
    - icon: لینک به آیکون صفحه.
    - viewport: تنظیمات مربوط به نمایش روی دستگاه‌های مختلف.
    - theme-color: تعیین رنگ تم.
    - description: توضیحات کوتاه درباره صفحه.
    - manifest: لینک به فایل مانیفست.
    - title: عنوان صفحه.
    - style: استایل‌های مربوط به صفحه.

## اپلیکیشن اصلی (App.js)

* هدف:
  + تعریف کامپوننت اصلی برنامه React.
* توضیحات:
  + useState: استفاده از Hook برای مدیریت وضعیت.
  + getToken, removeToken: توابع مدیریت توکن در localStorage.
  + handleLogin, handleLogout: توابع مدیریت وضعیت توکن.

## سرویس ارتباط با API (services/api.js)

* هدف:
  + تعریف توابع ارتباط با API سمت سرور.
* توضیحات:
  + axios: کتابخانه برای ارسال درخواست‌های HTTP.
  + BASE\_URL: آدرس پایه برنامه سمت سرور.
  + api.create: ایجاد یک نمونه از axios با تنظیمات خاص.
  + توابع مربوط به احراز هویت و وظایف.

## سرویس مدیریت توکن احراز هویت (services/auth.js)

* هدف:
  + تعریف توابع مدیریت توکن احراز هویت.
* توضیحات:
  + TOKEN\_KEY: نام کلید مربوط به توکن در localStorage.
  + getToken: دریافت توکن از localStorage.
  + setToken: ذخیره توکن در localStorage.
  + removeToken: حذف توکن از localStorage.

## صفحه ورود (components/Login.js)

* هدف:
  + تعریف کامپوننت صفحه ورود.
* توضیحات:
  + useState: استفاده از Hook برای مدیریت وضعیت.
  + loginUser: تابع ارسال درخواست ورود به سیستم.

## کامپوننت مدیریت وظایف (components/Tasks.js)

* هدف:
  + تعریف کامپوننت مدیریت وظایف.
* توضیحات:
  + useState, useEffect: استفاده از Hook‌ها برای مدیریت وضعیت و اجرای افعال در لحظه‌ی ایجاد یا به‌روزرسانی کامپوننت.
  + fetchTasks, createTask, updateTask, deleteTask: توابع مربوط به ارتباط با API مربوط به وظایف.

# **داکیومنت Docker**

## توضیحات اولیه

این پروژه با استفاده از Docker برای کانتینریزه‌سازی اجزای مختلف ایجاد شده است. Docker امکان اجرای پروژه در محیط مستقل و ایزوله را فراهم می‌کند.

## نحوه ایجاد و مدیریت Docker Containers

### ساخت تصاویر Docker

1. ابتدا، در ترمینال به مسیر پروژه (در که Dockerfile و docker-compose.yml وجود دارد) بروید.
2. برای ایجاد تصویر Docker از دستور زیر استفاده کنید:

docker build -t نام\_تصویر:تگ .

مثال:

docker build -t my-django-app:latest .

### ایجاد و اجرای Container

1. برای ایجاد و اجرای یک کانتینر از تصویر ایجاد شده، از دستور زیر استفاده کنید:

docker run -p پورت\_ها:پورت\_ها --name نام\_کانتینر نام\_تصویر:تگ

مثال:

docker run -p 8000:8000 --name my-django-container my-django-app:latest

### مدیریت Container

1. برای متوقف کردن یک کانتینر از دستور زیر استفاده کنید:

docker stop نام\_کانتینر

مثال:

* docker stop my-django-container
* برای حذف یک کانتینر از دستور زیر استفاده کنید:

docker rm نام\_کانتینر

مثال:

1. docker rm my-django-container

### نکته:

* می‌توانید پارامترهای بیشتری به دستورات Docker اضافه کنید. به مستندات رسمی Docker مراجعه کنید.

## نحوه اجرای پروژه

1. پیش‌نیازها:
   * Docker و Docker Compose باید بر روی سیستم شما نصب باشند.
2. اجرای پروژه:
   * به مسیر پروژه در ترمینال بروید.
   * از دستور زیر برای اجرای پروژه استفاده کنید:

* docker-compose up --build
* پس از اجرا، پروژه باید در دسترس باشد و از مرورگر خود می‌توانید به http://localhost:8000 بروید.
* متوقف کردن و حذف پروژه:
* برای متوقف کردن پروژه از دستور Ctrl + C در ترمینال استفاده کنید.
* برای حذف کلیه کانتینرها از دستور زیر استفاده کنید:
  + docker-compose down

### نکته:

* برای مشاهده لاگ‌ها و خروجی‌های پروژه از دستور docker-compose logs استفاده کنید.
* می‌توانید پارامترهای بیشتری به فایل docker-compose.yml اضافه کنید.