### **داکیومنت پروژه کاریابی**

# مستندات پروژه بک‌اند django

از ویژگی های این وبسایت میتوان به ورود با استفاده از jwt، ورود به صورت ادمین و اضافه کردن شغل ها یا حذف آنها، و ورود به صورت کاربر عادی و امکان ارسال رزومه خود برای موقعیت های شغلی موجود اشاره کرد.

توضیحات پروژه وبسایت کاریابی به صورت کامل در زیر اورده شده است.

**1. models.py:**

Job Model:

- title:

- فیلد `CharField` برای ذخیره عنوان شغل با حداکثر طول ۱۰۰ کاراکتر.

- description:

- فیلد `TextField` برای ذخیره توضیحات شغل.

`Request` Model:

- user:

- ارتباط با مدل کاربر (`User`) با استفاده از `ForeignKey`.

- job:

- ارتباط با مدل `Job` با استفاده از `ForeignKey`.

- resume:

- فیلد `FileField` برای آپلود رزومه با ذخیره در مسیر 'resumes/'.

- created\_at:

- فیلد `DateTimeField` برای ذخیره تاریخ ایجاد درخواست با استفاده از `auto\_now\_add`.

**2. serializers.py:**

`JobSerializer`:

- استفاده از `ModelSerializer` برای سریالایز کردن مدل `Job`.

`RequestSerializer`:

- استفاده از `ModelSerializer` برای سریالایز کردن مدل `Request`.

**3. signals.py:**

`create\_super\_user`:

- post\_migrate:

- یک سیگنال برای ایجاد یک کاربر ادمین با ایمیل 'admin@example.com' و رمز عبور 'admin'.

`create\_test\_user`:

- post\_migrate:

- یک سیگنال برای ایجاد یک کاربر تست با ایمیل 'test@example.com' و رمز عبور 'test'، بدون دسترسی ادمین.

**4. app.py:**

`JobConfig`:

- استفاده از `AppConfig` برای تنظیمات اپلیکیشن `job` و فراخوانی `signals` در `ready` method.

**5. admin.py:**

- Register models (`Job` and `Request`) in the admin panel.

نکات

- همواره اطمینان حاصل کنید که از ایمپورت‌ها و نام‌گذاری‌ها به درستی استفاده کرده‌اید.

- استفاده از `receiver` و `post\_migrate` برای اجرای عملیات‌ها پس از مهاجرت دیتابیس.

- تنظیمات اپلیکیشن در `AppConfig` انجام شده است.

### **6**. **views**.**py**:

#### Login Class:

* post method:
  + درخواست ورود کاربر با استفاده از نام کاربری و رمز عبور.
  + ایجاد توکن JWT با افزودن مشخصه is\_superuser به پی‌لود.

#### JobListCreateAPIView Class:

* get method:
  + دریافت لیست تمام شغل‌ها.
* post method:
  + ایجاد یک شغل جدید.

#### JobDetailAPIView Class:

* get method:
  + دریافت جزئیات یک شغل با احتمال استفاده از حافظه‌پنهانی.
* put method:
  + به‌روزرسانی اطلاعات یک شغل.
* delete method:
  + حذف یک شغل.

#### RequestSendAPIView Class:

* post method:
  + ارسال درخواست برای یک شغل با آپلود رزومه.

#### JobRequestsListAPIView Class:

* get\_queryset method:
  + بازیابی لیست درخواست‌ها بر اساس شناسه شغل.

### **7. urls.py:**

* تعریف مسیرهای مربوط به اپلیکیشن job.
* استفاده از کلاس‌های JobListCreateAPIView, Login, JobDetailAPIView, RequestSendAPIView, JobRequestsListAPIView.
* تغییر نام مسیر path('jobs/all/', ...).

نکات:

* استفاده از permission\_classes و authentication\_classes برای تعیین مجوزها و روش احراز هویت.
* استفاده از method\_decorator برای حافظه‌پنهانی اطلاعات در JobDetailAPIView.
* استفاده از RequestSendAPIView برای ارسال درخواست برای یک شغل.
* تعیین مسیرهای API برای ورود (login) و مدیریت شغل‌ها.

**8- توضیحات settings.py:**

#### INSTALLED\_APPS**:**

* + - * اضافه کردن اپلیکیشن‌های مورد نیاز برای پروژه.
      * اضافه کردن 'job' برای اپلیکیشن شغل.
      * اضافه کردن 'rest\_framework' برای استفاده از فریم‌ورک REST.
      * اضافه کردن 'rest\_framework\_jwt' برای احراز هویت با JWT.
      * اضافه کردن 'corsheaders' برای مدیریت درخواست‌های CORS.

#### DATABASES **(پایگاه داده):**

* + - استفاده از پایگاه داده PostgreSQL با مشخصات مشخص.
    - استفاده از نام کاربری 'admin' و رمز عبور '1234'.
    - استفاده از db\_service به عنوان نام هاست پایگاه داده.
    - امکان استفاده از SQLite (خطوط کامنت شده) به عنوان پایگاه داده آزمایشی.

#### AUTH\_PASSWORD\_VALIDATORS **(اعتبارسنجی رمز عبور):**

* + - تنظیم اعتبارسنجی رمز عبور با استفاده از تنوع نشانه‌ها.

#### LANGUAGE\_CODE**,** TIME\_ZONE**,** USE\_I18N**,** USE\_L10N**,** USE\_TZ **(تنظیمات بین‌المللی):**

* + - تنظیمات بین‌المللی برنامه.

#### STATIC\_URL **(فایل‌های استاتیک):**

* + - تنظیمات برای مسیر URL فایل‌های استاتیک.

#### DEFAULT\_AUTO\_FIELD **(نوع پیش‌فرض کلید اصلی):**

* + - تنظیم نوع پیش‌فرض کلید اصلی برای نسخه‌های جدیدتر Django.

#### CACHES **(حافظه‌پنهان Redis):**

* + - تنظیم حافظه‌پنهان با استفاده از Redis.

#### REST\_FRAMEWORK **(تنظیمات REST Framework):**

* + - استفاده از JSONWebTokenAuthentication به عنوان روش احراز هویت پیش‌فرض.

#### JWT\_AUTH **(تنظیمات JWT):**

* + - تنظیمات مربوط به JWT برای احراز هویت.

### **9-** urls.py**:**

#### urlpatterns**:**

* تنظیم مسیرهای مربوط به اپلیکیشن job.
* افزودن مسیرهای API به اپلیکیشن job.

نکات:

* استفاده از CORS برای مدیریت درخواست‌های CORS.
* تنظیمات مربوط به پایگاه داده PostgreSQL.
* تنظیمات مربوط به فایل‌های استاتیک و تنظیمات بین‌المللی.
* تنظیمات مربوط به احراز هویت با JWT.
* تنظیمات مربوط به حافظه‌پنهان Redis.

# مستندات پروژه فرانت اند react

### **1.** components/JobList.jsx**:**

#### useState **و** useEffect**:**

* استفاده از useState برای ایجاد وضعیت‌های متغیرهای مرتبط با کامپوننت.
* استفاده از useEffect برای اجرای عملیات نوشته شده درون آن به موقع.

#### fetchUserJobs **و** useEffect**:**

* تابع fetchUserJobs برای دریافت وظایف کاربر از API.
* استفاده از useEffect برای فراخوانی fetchUserJobs به صورت همزمان با تغییر توکن.

#### **متدهای** handleCreateJob **و** handleDeleteJob **و** handleSendRequest**:**

* متدهای مربوط به افزودن شغل جدید، حذف شغل و ارسال درخواست به API.

#### **نمایش لیست وظایف:**

* نمایش لیست وظایف با استفاده از map.
* بررسی مختصات وظیفه بر اساس نقش کاربر (Superuser یا عادی).

#### **ایجاد شغل توسط ادمین:**

* امکان ایجاد وظیفه جدید توسط ادمین.
* نمایش فرم ایجاد وظیفه در صورتی که کاربر ادمین باشد.

#### **Modal برای ارسال درخواست:**

* استفاده از Modal از react-bootstrap برای نمایش فرم ارسال درخواست به شغل.
* نمایش Modal باز یا بسته شده بر اساس وضعیت selectedJob.

### **2.** components/Login.jsx**:**

#### useState **و** handleLogin**:**

* استفاده از useState برای نگه‌داری وضعیت ورودی‌های فرم.
* تابع handleLogin برای ارسال درخواست ورود به API و ذخیره توکن.

#### **فرم ورود:**

* فرم ساده ورود با دو ورودی برای نام کاربری و رمز عبور.
* امکان ورود با فشردن دکمه.

نکات:

* استفاده از react-bootstrap برای UI و Modal.
* استفاده از توابع API برای ارتباط با سمت سرور.
* نگه‌داری وضعیت‌های کامپوننت با useState.
* استفاده از useEffect برای اجرای عملیات درون آن در زمان‌های خاص.
* مدیریت وضعیت selectedJob برای نمایش و بسته شدن Modal.
* استفاده از setToken برای ذخیره توکن برنامه.

### **3.** pages/api.jsx**:**

#### api **و** BASE\_URL**:**

* ایجاد یک instance از axios به نام api با تنظیمات پیش‌فرض.
* BASE\_URL به عنوان پایه برای تمام درخواست‌های API.

#### **توابع** loginUser **و** fetchJobs **و ...:**

* توابعی برای انجام درخواست‌های مربوط به احراز هویت (وارد شدن) و درخواست‌های کاربر.

### **4.** pages/auth.jsx**:**

#### **توابع** getToken **و** setToken **و** removeToken**:**

* توابعی برای مدیریت توکن در localStorage.

### **5.** App.jsx**:**

#### **استفاده از** useState **و** useEffect**:**

* استفاده از useState برای نگه‌داری وضعیت توکن و نقش کاربر.
* استفاده از useEffect برای بررسی توکن در زمان بارگذاری برنامه.

#### **توابع** handleLogin **و** handleLogout**:**

* توابعی برای ورود و خروج کاربر.

#### **نمایش** Login **یا** JobList **بر اساس وجود توکن:**

* نمایش فرم ورود (Login) در صورت عدم وجود توکن.
* نمایش لیست وظایف (JobList) در صورت وجود توکن.

#### **توابع** parseJwt**:**

* تابعی برای دیکد کردن توکن JWT.

نکات:

* استفاده از localStorage برای ذخیره و بازیابی توکن.
* استفاده از توابع API برای ارتباط با سمت سرور.
* اعتبارسنجی و انتقال توکن در درخواست‌های API با استفاده از headers.
* استفاده از useEffect برای بررسی توکن و نقش کاربر در زمان بارگذاری برنامه.

# مستندات docker

توضیحات برای فایل‌ها و Docker Compose:

### **1.** backend/Dockerfile**:**

* استفاده از تصویر python:3.8 به عنوان پایه.
* مشخص کردن مسیر کاری (WORKDIR) به /app.
* کپی کردن فایل requirements.txt به /app.
* نصب پکیج‌ها با استفاده از pip.
* کپی کل فایل‌های پروژه به /app.
* دستور CMD برای اجرای مهم‌ترین دستورات (ساخت migration، اجرای migrate و اجرای سرور).

### **2.** database/Dockerfile**:**

* استفاده از تصویر postgres:latest به عنوان پایه.
* کپی کردن فایل initial.sql به /docker-initdb.d/.

### **3.** initial.sql**:**

* دستور SQL برای ایجاد دیتابیس jobs\_db.

### **4.** redis/Dockerfile**:**

* استفاده از تصویر redis:latest به عنوان پایه.

### **5.** frontend/Dockerfile**:**

* استفاده از تصویر node:14 به عنوان پایه.
* مشخص کردن مسیر کاری (WORKDIR) به /app.
* کپی کردن فایل‌های package.json و package-lock.json به /app.
* نصب پکیج react-scripts.
* نصب پکیج‌ها با استفاده از npm.
* کپی کل فایل‌های frontend به /app.
* تعیین EXPOSE 3000 برای اطلاع از پورت مورد نیاز توسط Docker.
* مشخص کردن REACT\_APP\_BACKEND\_URL به http://localhost:8000.
* دستور CMD برای اجرای npm start.

### **6.** docker-compose.yml**:**

* services تعیین اجزای سرویس.
* networks و volumes برای تعریف شبکه و ولوم‌ها.
* database سرویس پایگاه داده PostgreSQL با متغیرهای محیطی و ولوم.
* django\_app سرویس اجرای Django با اتصال به پایگاه داده و تعیین پورت.
* react\_app سرویس React با تعیین پورت و اتصال به django\_app.
* redis سرویس Redis.

نکات:

* توجه به مسیرهای کاری و کپی فایل‌ها در Dockerfiles.
* استفاده از Docker Compose برای تعریف و مدیریت چندین سرویس.
* تنظیمات محیطی و شبکه برای ارتباط میان سرویس‌ها.
* استفاده از volumes برای ذخیره دائمی داده‌های PostgreSQL.
* depends\_on برای تعیین وابستگی‌ها بین سرویس‌ها.