سرویس keycloak یک پلتفرم آماده برای مدیریت کاربران سرویس های مختلف است که چندین صفحه ورود رو در یک صفحه فراهم میکند مستندات این پلتفرم در این <https://www.keycloak.org/documentation> و برای مطالعه سند restapi ان در <https://www.keycloak.org/docs-api/21.0.1/rest-api/index.html> مطالعه کنید

ما با استفاده از کتابخانه پایتونی python-keycloak جهت کدنویسی و کار با این پلتفرم استفاده میکنیم داکیومنت این کتابخانه در این <https://python-keycloak.readthedocs.io/en/latest/> مطالعه کنید

**راه اندازی keycloak**

در داخل دایرکتوری root پروژه یک دایرکتوری با نام keycloak شده که شامل یک فایل yaml برای راه اندازی و بالا آوردن ئلتفرم و اتصال به یک دیتابیس postgres میباشد با مطالعه فایل README.md سرویس رو بالا بیاورید و وارد محیط کنسول آن شوید

با استفاده از دو مقداری که در فایل .env مقداردهی کردید (KEYCLOAK\_ADMIN , KEYCLOAK\_ADMIN\_PASSWORD) وارد محط کنسول شوید

در قسمت چپ بالا بر روی منوی کشویی زده و ابتدا یک realm برای پروژه بسازید realm name نباید تکراری باشد و Enabled رو فعال بزارید و سپس دکمه create رو بزنید بعد از ساخته شدن شما به این realm جابجا میشوید اکنون در منوی سمت چپ به قسمت User رفته و شروع به ساخت یک کاربر کنید تمامی فیلدهای username, email, first name, last name و همچنین email verified فعال کنید و دکمه create رو بزنید بعد ازاتمام به قسمت تنظیمات این کاربر وارد میشوید در تب credentials برای این یوزر پسورد ست کنید (در هنگام پسورد دادن Temporary غیرفعال کنید ) به تب role mapping رفته و برای این کاربر Assign Role کنید با استفاده از filter by realm roles دو مقدار role مورد نیاز ما manage-users , manage-realm رو assign کنید با استفاده از اطلاعات این یوزر اکنون میتوانید وارد محیط کنسول مختص خود این یوزر شوید (domain/admin/REALM/consol/) مقدار REALM با realm که ساختید تغییر دهید

در همان منوی سمت چپ کنسول به بخش Client بروید و یک کلاینت جدید بسازید در مرحله اول برای آن یک client id غیر تکراری بگذارید مقدار client type نیز بر روی OpenID Connect بگذارید با زدن دکمه next به مرحله بعد رفته گزینه های client authentication رو فعال و گزینه service account roles رو اضافه کنید به مرحله بعد رفته و در قسمت valid redirect URIs مقدار \* بگذارید و کلاینت رو ذخیره کنید به قسمت تنظیمات این کلاینت جابجا شده به تب service account roles رفته و به ان assign role نسبت بدهید با استفاده از filter by realm roles مقدار role با عنوان realm-admin نیازمندیم و assign کنید به تب credentials رفته و client secret رو میتونید مشاهده کنید

با استفاده از مقادیر یوزر و کلاینت و نام realm که ساختیم اکنون ما یک محیط کپسوله برای کار با کی کلاک و استفاده آن در پروژه داریم این مقادیر رو در .env روت پروژه وارد کنید و قرار دهید

KEYCLOAK\_SERVER\_URL='http://localhost:8080/'

KEYCLOAK\_CLIENT\_ID='test'

KEYCLOAK\_REALM\_NAME='test'

KEYCLOAK\_CLIENT\_SECRET\_KEY='QXYGmiB7gdvRURsR9WBMdHvLeo4sr9A3'

KEYCLOAK\_USER\_REALM\_NAME='test'

KEYCLOAK\_USERNAME='test'

KEYCLOAK\_PASSWORD='test'

در دایرکتوری روت پروژه در دایرکتوری accounts و فایل service.py کدهای مورد نیاز ما برای ارتباط بر قرار کردن با keycloak وجود دارد سه کلاس با نام basekeycloak ( که برای مقدار دهی اولیه و هندل کردن ارتباط ) و کلاس userkeycloak جهت مدیریت کاربران و کلاس tokenkeycloak جهت کار با توکن و مدیریت آن نوشته شده است

**کلاس basekeycloak**

    STATUS\_OK = 200

    STATUS\_CREATED = 201

    STATUS\_NO\_CONTENT = 204

    STATUS\_NOT\_FOUND = 404

    STATUS\_FORBIDDEN = 403

    STATUS\_SERVER\_ERROR = 500

مقدار دهی اولیه و ثابت برای ریسناس ها و خروجی که با توچه به api نویسی پروژه جهت یکدست نگه داشتن پروژه مطابق با استانداردهای status مقدار دهی گشته است

از این مقدارهای ثابت خروجی هر درخواست از keycloak بر اساس نیاز ما مقدار دهی گشته تا کنترل آن در سرتاسر پروژه یکدست و هماهنگ گردد و بتوان از این سرویس در تمامی بخش های پروژه بدون دغدغه خاصی استفاده کرد و بتوان آنرا به دیگر پروژه ها نیز انتقال داد

---------------------------------------------------------------------

    def \_\_init\_\_(self):

        self.\_username = None

        self.\_password = None

        self.keycloak\_admin = self.admin\_connect()

        self.keycloak\_openid = self.openid\_connect()

    @property

    def username(self):

        return self.\_username

    @username.setter

    def username(self, value):

        if not value:

            raise ValueError('username cannot be empty')

        self.\_username = value

    @property

    def password(self):

        return self.\_password

    @password.setter

    def password(self, value):

        if not value:

            raise ValueError('password cannot be empty')

        self.\_password = value

مقدار دهی متغییرهای داینامیک و کنترل کردن با استفاده از setter مطابق موردنیاز هر بخش از پروژه این چهار متغییر بسته به نیاز هر اندئوینت ممکن است فراخوانی و مورد استفاده قرار گیرد

---------------------------------------------------------------------

    def admin\_connect(self):

        try:

            keycloak\_connection = KeycloakOpenIDConnection(

                server\_url=settings.KEYCLOAK\_SERVER\_URL,

                username=settings.KEYCLOAK\_USERNAME,

                password=settings.KEYCLOAK\_PASSWORD,

                realm\_name=settings.KEYCLOAK\_REALM\_NAME,

                user\_realm\_name=settings.KEYCLOAK\_USER\_REALM\_NAME,

                client\_id=settings.KEYCLOAK\_CLIENT\_ID,

                client\_secret\_key=settings.KEYCLOAK\_CLIENT\_SECRET\_KEY,

                verify=True

            )

            keycloak\_admin = KeycloakAdmin(connection=keycloak\_connection)

            return keycloak\_admin

        except Exception as e:

            logging.error(f"Error connecting to Keycloak: {e}")

            return None

    def openid\_connect(self):

        try:

            keycloak\_openid = KeycloakOpenID(

                server\_url=f"{settings.KEYCLOAK\_SERVER\_URL}/auth",

                client\_id=settings.KEYCLOAK\_CLIENT\_ID,

                realm\_name=settings.KEYCLOAK\_REALM\_NAME,

                client\_secret\_key=settings.KEYCLOAK\_CLIENT\_SECRET\_KEY

            )

            keycloak\_openid.well\_known()

            return keycloak\_openid

        except Exception as e:

            logging.error(f"Error connecting to Keycloak: {e}")

            return self.STATUS\_SERVER\_ERROR

    def check\_connect(self):

        try:

            self.keycloak\_admin.get\_realms()

            return self.STATUS\_OK

        except Exception as e:

            logging.error(f"Error getting server: {e}")

            return self.STATUS\_SERVER\_ERROR

کنترل کردن کانکت شدن به keycloak در تمام بخش سرویس و متد check\_connect جهت بررسی ارتباط با کی کلاک در صورت نیاز در سرتاسر پروژه

**کلاس UserKeycloak**

این کلاس جهت مدیریت کاربران از جمله گرفتن اطلاعات و بررسی ماهیت دسترسی و ساخت و آپدیت اطلاعات آن می باشد

---------------------------------------------------------

گرفتن یوزر و دریافت اطلاعات یوزر

    def get\_user\_id(self):

        user\_id = self.keycloak\_admin.get\_user\_id(self.username)

        if user\_id is not None:

            return user\_id

        else:

            return self.STATUS\_NOT\_FOUND

    def get\_user(self):

        user\_id = self.get\_user\_id()

        if user\_id == self.STATUS\_NOT\_FOUND:

            return self.STATUS\_NOT\_FOUND

        try:

            user = self.keycloak\_admin.get\_user(user\_id)

            return user

        except Exception as e:

            logging.error(f"Error getting user: {e}")

            return self.STATUS\_NOT\_FOUND

---------------------------------------------------------

بررسی فعال بودن یوزر و تایید شدن ایمیل آن

چک کردن فعال بودن یوزر در صورت غیرفعال بودن یوزر این کاربر از تمامی دسترسی های ممکن محروم میگردد

    def check\_enable(self):

        user = self.get\_user()

        if user == self.STATUS\_NOT\_FOUND:

            return self.STATUS\_NOT\_FOUND

        elif user['enabled']:

            return self.STATUS\_OK

        else:

            return self.STATUS\_NOT\_FOUND

تغییر فعال و غیر فعال بودن یوزر

    def enable(self):

        user = self.get\_user()

        if user == self.STATUS\_NOT\_FOUND:

            return self.STATUS\_NOT\_FOUND

        else:

            try:

                user['enabled'] = True

                self.keycloak\_admin.update\_user(user\_id=user['id'], payload=user)

                return self.STATUS\_OK

            except Exception as e:

                logging.error(f"Error enabling user: {e}")

                return self.STATUS\_NOT\_FOUND

    def disable(self):

        user = self.get\_user()

        if user == self.STATUS\_NOT\_FOUND:

            return self.STATUS\_NOT\_FOUND

        else:

            try:

                user['enabled'] = False

                self.keycloak\_admin.update\_user(user\_id=user['id'], payload=user)

                return self.STATUS\_OK

            except Exception as e:

                logging.error(f"Error disabling user: {e}")

                return self.STATUS\_NOT\_FOUND

بررسی تایید شدن ایمیل کاربر

    def check\_email\_verify(self):

        user = self.get\_user()

        if user == self.STATUS\_NOT\_FOUND:

            return self.STATUS\_NOT\_FOUND

        elif user['emailVerified']:

            return self.STATUS\_OK

        else:

            return self.STATUS\_NOT\_FOUND

تایید ایمیل

    def email\_verified(self):

        user = self.get\_user()

        if user == self.STATUS\_NOT\_FOUND:

            return self.STATUS\_NOT\_FOUND

        try:

            user['emailVerified'] = True

            self.keycloak\_admin.update\_user(user\_id=user['id'], payload=user)

            return self.STATUS\_NO\_CONTENT

        except Exception as e:

            logging.error(f"Error updating email verification status: {e}")

            return self.STATUS\_NOT\_FOUND

--------------------------------------------------

ساخت کاربر

با توجه به امکان ثبت نام کردن کاربر با دو مقدار ایمیل و شماره تلفن برای هرکدوم یک متد جداگانه نوشته شده است تمامی فسلدهای مورد نیاز است و باید نوشته شود

    def create\_email(self):

        user\_id = self.get\_user\_id()

        if user\_id == self.STATUS\_NOT\_FOUND:

            try:

                self.keycloak\_admin.create\_user(

                    {

                        "email": self.username,

                        "username": self.username,

                        "enabled": True,

                        "firstName": self.username,

                        "lastName": self.username,

                        "credentials": [

                            {

                                "value": self.password,

                                "type": "password",

                                "temporary": False

                            }

                        ]

                    }

                )

                return self.STATUS\_CREATED

            except Exception as e:

                logging.error(f"Error creating user: {e}")

                return self.STATUS\_NOT\_FOUND

        else:

            try:

                self.keycloak\_admin.set\_user\_password(user\_id=user\_id, password=self.password, temporary=False)

                return self.STATUS\_NO\_CONTENT

            except Exception as e:

                logging.error(f"Error setting user password: {e}")

                return self.STATUS\_NOT\_FOUND

    def create\_phone(self):

        user\_id = self.get\_user\_id()

        if user\_id == self.STATUS\_NOT\_FOUND:

            try:

                self.keycloak\_admin.create\_user(

                    {

                        "email": f"{self.username}@gmail.com",

                        "username": self.username,

                        "enabled": True,

                        "firstName": self.username,

                        "lastName": self.username,

                        "credentials": [

                            {

                                "value": self.password,

                                "type": "password",

                                "temporary": False

                            }

                        ]

                    }

                )

                return self.STATUS\_CREATED

            except Exception as e:

                logging.error(f"Error creating user: {e}")

                return self.STATUS\_NOT\_FOUND

        else:

            try:

                self.keycloak\_admin.set\_user\_password(user\_id=user\_id, password=self.password, temporary=False)

                return self.STATUS\_NO\_CONTENT

            except Exception as e:

                logging.error(f"Error setting user password: {e}")

                return self.STATUS\_NOT\_FOUND

-----------------------------------------------------------

تغییر پسورد یوزر

    def change\_password(self):

        user\_id = self.get\_user\_id()

        if user\_id == self.STATUS\_NOT\_FOUND:

            return self.STATUS\_NOT\_FOUND

        else:

            try:

                self.keycloak\_admin.set\_user\_password(user\_id=user\_id, password=self.password, temporary=False)

                return self.STATUS\_CREATED

            except Exception as e:

                logging.error(f"Error creating user: {e}")

                return self.STATUS\_NOT\_FOUND

**کلاس TokenKeycloak**

این کلاس جهت کار با توکن و مدیریتی آن میباشد

مقداردهی اولیه توکن

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.\_token = None

    @property

    def token(self):

        return self.\_token

    @token.setter

    def token(self, value):

        if not value:

            raise ValueError('username cannot be empty')

        self.\_token = value

----------------------------------------

گرفتن و ایجاد توکن برای کاربر

    def get\_token(self):

        try:

            token = self.keycloak\_openid.token(self.username, self.password)

            return token

        except Exception as e:

            logging.error(f"Error for get user token: {e}")

            return self.STATUS\_SERVER\_ERROR

-----------------------------------

تمدید توکن دسترسی کاربر

    def refresh\_token(self):

        try:

            token = self.keycloak\_openid.refresh\_token(self.token)

            return token

        except Exception as e:

            logging.error(f"refresh token error: {e}")

            return self.STATUS\_SERVER\_ERROR

---------------------------------

گرفتن اطلاعات کلی توکن همراه کاربران آن

    def decode\_token(self):

        try:

            token\_info = self.keycloak\_openid.decode\_token(self.token)

            return token\_info

        except Exception as e:

            logging.error(f"decode token error: {e}")

            return self.STATUS\_SERVER\_ERROR

-------------------------------------

گرفتن اطلاعات کاربر از توکن

    def user\_info(self):

        try:

            userinfo = self.keycloak\_openid.userinfo(self.token)

            return userinfo

        except Exception as e:

            logging.error(f"user information error: {e}")

            return self.STATUS\_NOT\_FOUND

--------------------------------------

خروج کاربر از حساب خود و حذف توکن دسترسی

    def signout(self):

        try:

            self.keycloak\_openid.logout(self.token)

            return self.STATUS\_OK

        except Exception as e:

            logging.error(f"user sign out error: {e}")

            return self.STATUS\_NOT\_FOUND

از این سرویس ما در بخش سریالایزر پروژه استفاده میکنیم تا اندپوینتها و صفحه نهایی خودمون رو به کاربر نمایش دهیم ما فیلدهای کنترلی و خروجی خود نهایی نمایش داده شده به کاربر رو کاستومایز کرده و کنترل آن را در سریالایزر انچام داده تا انتعطاف پذیری بیشتری بر روند کلی پروژه داشته باشیم

در فایل serializers.py به دو بخش جهت مار با مدیریت کاربران و مدیریت توکن تقسیم میشود

سریالایزر های مدیریت کاربر شامل

class SignupSerializer(serializers.Serializer):

class OTPRequestSeriailizer(serializers.Serializer):

class OTPSigninSerializer(serializers.Serializer):

class OTPPasswordChangeVerifySerializer(OTPSigninSerializer):

class OTPSingnupVerifySerializer(serializers.Serializer):

class PasswordChangeSerializer(serializers.Serializer):

class PasswordSinginSerializer(serializers.Serializer):

و سریالایزرهای مدیریت توکن و دسترسی

class TokenBaseSerializer(serializers.Serializer):

class SignoutSerializer(TokenBaseSerializer):

class UserInfoSerializer(serializers.Serializer):

class RefreshTokenSerializer(serializers.Serializer):

class DecodeTokenSerializer(serializers.Serializer):

هر کدام از این سریالایزر متناسب با نام کلاس view خود در فایل views.py نام گذاری شده که بصورت زیر میباشد

class SignupView(APIView):

class OTPSingnupVerifyView(APIView):

class OTPRequestView(APIView):

class OTPPasswordChangeVerifyView(APIView):

class PasswordChangeView(APIView):

class PasswordSigninView(APIView):

class SignoutView(APIView):

class UserinfoView(APIView):

class RefreshTokenView(APIView):

class DecodeTokenView(APIView):

در طی تمامی پروژه در خصوص otp بصورت سشن و رمزنگاری یکطرفه صورت گرفته که امنیت حفظ گردد در هر درخواست otp به مدت پنج دقیقه و برای پنج بار و هر otp مختص خود دستگاه و سشن آن مورد استفاده و تااید میباشد در غیر اینصورت otp فاقد اعتبار خواهد شد از هر otp نیز تنها برای یکبار ورود معتبر می باشد رمزگذاری otp با مقدار کلید رمز 32 بیتی در بخش تنظیمات صورت میگیرد

VALUE\_HASH = pyotp.random\_base32()

ارسال otp به کاربران توسط سلری و ردیس صورت میگیرد تا در ارسال بالا بتوان انسجام برنامه را حفظ نمود سرویس ها توسط داکر بالا می آیند و تنظیمات آن در settings.py به این شکل است

KEYCLOAK\_ROLES\_TO\_DJANGO\_IS\_STAFF = "is\_staff"

KEYCLOAK\_SERVER\_URL=config("KEYCLOAK\_SERVER\_URL", cast=str)

KEYCLOAK\_CLIENT\_ID=config("KEYCLOAK\_CLIENT\_ID", cast=str)

KEYCLOAK\_REALM\_NAME=config("KEYCLOAK\_REALM\_NAME", cast=str)

KEYCLOAK\_CLIENT\_SECRET\_KEY=config("KEYCLOAK\_CLIENT\_SECRET\_KEY", cast=str)

KEYCLOAK\_USER\_REALM\_NAME=config("KEYCLOAK\_USER\_REALM\_NAME", cast=str)

KEYCLOAK\_USERNAME=config("KEYCLOAK\_USERNAME", cast=str)

KEYCLOAK\_PASSWORD=config("KEYCLOAK\_PASSWORD", cast=str)

*#CELERY*

CELERY\_BROKER\_URL = f"redis://:{config('REDIS\_PASSWORD', cast=str)}@{config('REDIS\_HOST', cast=str)}:{config('REDIS\_PORT', cast=int)}/0"

CELERY\_RESULT\_BACKEND = CELERY\_BROKER\_URL

CELERY\_ACCEPT\_CONTENT = ['json']

CELERY\_TASK\_SERIALIZER = 'json'

CELERY\_RESULT\_SERIALIZER = 'json'

CELERY\_TIMEZONE = 'Asia/Tehran'

مقادیر آن نیز از فایل .env گرفته میشد

مقدار دهی اولیه متنی آن در دایرکتوری utils و در روت و در فایل OTP\_sender میباشد

from django.conf import settings

from django.core.mail import send\_mail

class OTPSender:

    def \_\_init\_\_(self, otp:int):

        self.otp = otp

        self.\_phone = None

        self.\_email =  None

    @property

    def phone(self):

        return self.\_phone

    @phone.setter

    def phone(self, value):

        if not value:

            raise ValueError('phone cannot be empty')

        self.\_phone = value

    @property

    def email(self):

        return self.\_email

    @phone.setter

    def phone(self, value):

        if not value:

            raise ValueError('email cannot be empty')

        self.\_email = value

    def phone\_sender(self):

        pass

    def email\_sender(self):

        subject = 'Sender message form ITS company'

        message = f'Hi. your OTP is {self.otp}, thank you for authorization'

        email\_from = settings.EMAIL\_HOST\_USER

        recipient\_list = [self.email, ]

        send\_mail( subject, message, email\_from, recipient\_list )