
بسم الله الرحمن الرحيم

گزارش کارآموزی شایا

حامد خسروی

شماره دانشجویی: ۹۹۲۰۲۳۰۱۰

تابستان ۱۴۰۲

فهرست

هفته اول و دوم (۱۴۰۲/۰۴/۱۵ – ۱۴۰۲/۰۴/۳۱)

پیاده‌سازی API برای ثبت‌نام کاربران با استفاده از JWT

این کدها به یک سیستم ثبت‌نام کاربران در یک وبسایت یا برنامه مبتنی بر API اشاره دارند، که از Django و Django REST Framework برای ساخت API استفاده می‌کنند. در این کد، یک ویو (view) و یک سریالایزر (serializer) برای ثبت‌نام کاربران ایجاد شده‌اند.

```python
# user/serializers.py
from rest_framework import serializers
from django.contrib.auth.models import User
from rest_framework_simplejwt.tokens import RefreshToken

class UserSerializer(serializers.ModelSerializer):
    password = serializers.CharField(write_only=True)

    def create(self, validated_data):
        user = User.objects.create_user(
            username=validated_data['username'],
            password=validated_data['password']
        )
        return user

    class Meta:
        model = User
        fields = ('id', 'username', 'password')
```

در این فایل، یک سریالایزر برای کاربران ساخته شده است. این سریالایزر از `serializers.ModelSerializer`که از `rest_framework` وارد شده استفاده می‌کند. سریالایزر اطلاعات کاربر را از ورودی دریافت می‌کند و یک کاربر جدید با استفاده از اطلاعات داده

شده ایجاد می‌کند. فیلدهای `id`، `username` و `password` انتخاب شده‌اند و فقط `username` و `password` به عنوان فیلدهای قابل نمایش تنظیم شده‌اند.

```python
# user/views.py
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import status
from rest_framework_simplejwt.tokens import RefreshToken
from .serializers import UserSerializer

class RegisterUser(APIView):
    def post(self, request):
        serializer = UserSerializer(data=request.data)
        if serializer.is_valid():
            user = serializer.save()
            refresh = RefreshToken.for_user(user)
            return Response(
                {
                    'user_id': user.id,
                    'username': user.username,
                    'refresh': str(refresh),
                    'access': str(refresh.access_token),
                },
                status=status.HTTP_201_CREATED
            )
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

در این فایل، یک کلاس `RegisterUser` برای ویو ایجاد شده است که یک `post` متد را پیاده‌سازی می‌کند. این متد برای درخواست‌های POST به این ویو اجازه می‌دهد تا اطلاعات یک کاربر جدید را دریافت کند و با استفاده از سریالایزر، این اطلاعات را بررسی و در صورت صحت، یک کاربر جدید ایجاد کند. اگر اطلاعات ورودی معتبر نباشند، خطاهای مربوطه برگردانده می‌شوند. در صورت موفقیت آمیز بودن ثبت‌نام، جوابی شامل اطلاعات کاربر (مانند `user_id` و `username`) و توکن‌های JWT (Refresh و Access) به عنوان پاسخ برمی‌گرداند.

هفته سوم و چهارم (۱۴۰۲/۰۵/۰۱ – ۱۴۰۲/۰۵/۱۵)

پیاده‌سازی API مربوط به محاسبه نمره طبق امتیاز گزینه‌های هر سوال

```python
# test/models.py
from django.db import models

class Question(models.Model):
    text = models.CharField(max_length=128)
    parent = models.ForeignKey("self", on_delete=models.CASCADE, null=True, blank=True)
    has_table = models.BooleanField(default=False)
    has_sub_detail = models.BooleanField(default=False)

class Answer(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    text = models.TextField()
    score = models.IntegerField(default=0)
    is_checked = models.BooleanField(default=None, null=True, blank=True)
    self_input = models.CharField(max_length=128, default=None, null=True, blank=True)
```

در اینجا مدل‌های دو جدول در پایگاه‌داده تعریف شده‌اند:

مدل `Question` دارای ویژگی‌هایی مانند (`text` یک متن با حداکثر طول ۱۲۸ کاراکتر)، (`parent` که یک Question دیگر را به عنوان پدر می‌پذیرد، با اشاره به خودش)، (`has_table` یک فیلد بولین که به صورت پیش‌فرض `False` است) و (`has_sub_detail` یک فیلد بولین که به صورت پیش‌فرض `False` است) می‌باشد.

مدل `Answer` دارای ویژگی‌هایی مانند (`question` ارتباط با مدل Question به عنوان کلید خارجی)، (`user` ارتباط با مدل User به عنوان کلید خارجی)، (`text` متن پاسخ به صورت متن طولانی)، (`score` یک عدد صحیح که به صورت پیش‌فرض صفر است)، (`is_checked` یک فیلد بولین که می‌تواند هم`True` ، هم `False` یا `None` باشد) و (`self_input` یک متن با حداکثر طول ۱۲۸ کاراکتر که به صورت پیش‌فرض `None` است) می‌باشد.

```python
# test/serializers.py
class AnswerSerializer(serializers.ModelSerializer):
    class Meta:
        model = Answer
        fields = '__all__'


class QuestionSerializer(serializers.ModelSerializer):
    class Meta:
        model = Question
        fields = '__all__'
```

همچنین، سه فایل serializer برای این مدل‌ها تعریف شده است:

سریالایزر `AnswerSerializer` که مدل `Answer` را سریالایز می‌کند و تمام فیلدهای آن را شامل می‌شود.

سریالایزر `QuestionSerializer` که مدل `Question` را سریالایز می‌کند و تمام فیلدهای آن را شامل می‌شود.

```python
# test/views.py
class CalculateScore(APIView):
    def post(self, request):
        user = request.user
        answers = request.data.get('answers')
        total_score = 0
        for answer_data in answers:
            answer = Answer.objects.get(id=answer_data['id'])
            # Calculate score based on answer and update the Answer object ...
        return Response({'total_score': total_score}, status=status.HTTP_200_OK)
```

در فایل`views.py` ، یک کلاس به نام `CalculateScore` تعریف شده است که از `APIView` ارث‌بری می‌کند. این کلاس یک متد `post` دارد که اطلاعات درخواست را دریافت می‌کند. این متد ابتدا کاربر را از درخواست دریافت می‌کند و سپس اطلاعات مربوط به پاسخ‌ها را از داده‌های درخواست استخراج می‌کند.

سپس برای هر داده‌ی پاسخ، پاسخ متناظر با شناسه آن را از پایگاه داده با استفاده از `Answer.objects.get(id=answer_data['id'])` دریافت می‌کند. در نهایت، مجموع امتیازات را طبق یک الگوریتمی محاسبه کرده و آن را با استفاده از `Response({'total_score': total_score}, status=status.HTTP_200_OK)` به عنوان پاسخ به کلاینت ارسال می‌کند.

*** به دلیل محرمانه بودن قسمت محاسبه امتیازات، ارائه الگوریتم محاسبه امتیازات امکان‌پذیر نمی‌باشد.**

هفته پنجم و ششم (۱۴۰۲/۰۵/۱۵ – ۱۴۰۲/۰۵/۳۱)

داکرایزکردن پروژه با Docker

این سیستم شامل سه سرویس است: وب اپلیکیشن Django، دیتابیس PostgreSQL و سرویس Nginx به عنوان وب سرور معین استفاده می‌شود تا به وب اپلیکیشن Django وصل شود.

```
# Dockerfile
FROM python:3.10-slim

ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1
WORKDIR /app

COPY requirements.txt /app/

RUN pip install --no-cache-dir -r requirements.txt

COPY . /app/
```

`FROM python:3.10-slim`: این داکر فایل از تصویر داکر رسمی Python با نسخه ۳.۱۰ استفاده می‌کند.

`ENV PYTHONDONTWRITEBYTECODE 1` و `ENV PYTHONUNBUFFERED 1`: تنظیم متغیرهای محیطی برای Python به منظور جلوگیری از ایجاد فایل‌های `.pyc` و تنظیم ورودی و خروجی غیر بافر.

`WORKDIR /app`: تعیین مسیر کاری داخل کانتینر به `/app`.

`COPY requirements.txt /app/`: کپی کردن فایل `requirements.txt` به داخل کانتینر.

`RUN pip install --no-cache-dir -r requirements.txt`: نصب پکیج‌های مورد نیاز برنامه با استفاده از `pip`.

```yaml
# docker-compose.yaml
version: '3'

services:
  web:
    build: .
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - .:/app
    ports:
      - "8000:8000"
    depends_on:
      - db

  db:
    image: postgres:14
    environment:
      POSTGRES_DB: myproject
      POSTGRES_USER: myuser
      POSTGRES_PASSWORD: mypassword

  nginx:
    image: nginx:1.25.2
    ports:
      - "80:80"
    volumes:
      - ./nginx:/etc/nginx/conf.d
    depends_on:
      - web
```

`version: '3'`: نسخه‌ی استفاده شده از فرمت `docker-compose`.

`web`: سرویس وب اپلیکیشن Django که از تنظیمات داخل فایل Dockerfile برای ساخت تصویر استفاده می‌کند. این سرویس به دیتابیس PostgreSQL وابسته است.

`db`: سرویس دیتابیس PostgreSQL با تنظیمات محیطی مشخص برای نام دیتابیس، نام کاربری و رمز عبور.

`nginx`: سرویس Nginx که به عنوان وب سرور به وب اپلیکیشن Django متصل می‌شود. این سرویس به وب اپلیکیشن وابسته است و تنظیمات Nginx را از داخل پوشه `nginx` در سطح بالایی پروژه می‌گیرد.

```
upstream sampleapp1 {
    server django:8000;
}

server {
    server_name talentcentral.ir;
    listen 80;
    client_max_body_size 0;
    root /usr/share/nginx/html;

    location / {
        proxy_pass http://sampleapp1;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $host;
        proxy_redirect off;
    }
    location /media {
        autoindex on;
    }
    location /static {
        autoindex on;
    }

}
```

`upstream sampleapp1`: تعریف گروه سرور `sampleapp1` که به سرویس Django با آدرس `django:8000` (نام سرویس درDocker Compose) متصل می‌شود.

`server`: تنظیمات سرور Nginx برای دریافت و ارسال ترافیک.

`server_name talentcentral.ir`: تنظیم نام سرور برای دسترسی به این وب سرور.

`listen 80`: گوش دادن به درخواست‌ها بر روی پورت ۸۰.

`location /`: تنظیمات مسیر اصلی وب‌سرور با استفاده از `proxy_pass` برای ارسال ترافیک به `sampleapp1`.

`location /media` و `location /static`: تنظیمات برای ارسال فایل‌های `media` و `static` اپلیکیشن به صورت `autoindex on` که به معنای نمایش فایل‌های موجود در این مسیرها می‌باشد.

هفته هفتم و هشتم (۱۴۰۲/۰۶/۰۱ – ۱۴۰۲/۰۶/۱۵)

وارد کردن سوالات آزمون با استفاده از فایل اکسل

```python
class ImportStandardTestQuestionsApi(GeneralCustomClassMixin):
    @transaction.atomic
    def post(self, request, *args, **kwargs):
        try:
            excel_file = self.request.FILES.get('excel_file')
            if excel_file is None:
                return bad_request_response()

            excel_handler = ExcelHandler()
            excel_data = excel_handler.excel_to_dict(excel_file=excel_file)

            test_items_list = []
            test_item_option_list = []

            for row in excel_data:
                test_item_data = {
                    'name': row.get('Item Name', None),
                    'item_type': constants.ITEM_TYPE_MULTIPLE_CHOICE_STANDARD,
                    'question': row.get('Question Text', None),
                    'multiple_answer': False if row.get('is multiple_answer?') is None else bool(row.get('is multiple_answer?')),
                }
                test_item = TestItemModel(**test_item_data)
                test_items_list.append(test_item)

                for i in range(1, 9):
                    option_text = row.get(f"Option text {i}", None)
                    option_score = row.get(f"Option score {i}", None)
                    if option_text is not None and option_score is not None:
                        option_data = {
                            'item': test_item,
                            'option': option_text,
                            'score': option_score,
                        }
                        if i == int(row.get('Correct answer')):
                            option_data['is_correct_option'] = True
                        else:
                            option_data['is_correct_option'] = False

                        test_item_option = TestItemOptionModel(**option_data)
                        test_item_option_list.append(test_item_option)

            TestItemModel.objects.bulk_create(test_items_list, ignore_conflicts=True)
            TestItemOptionModel.objects.bulk_create(test_item_option_list, ignore_conflicts=True)

            result = {
                'data': TestItemDetailSerializer(test_items_list, many=True).data
            }

            return success_response(result)
        except Exception as e:
            logger.error(e)
            return error_response()
```

۱. در تابع `post` اطلاعات فایل اکسل از درخواست دریافت شده و سپس توسط `ExcelHandler`به دیکشنری تبدیل می‌شود.

۲. سپس دو لیست تهیه می‌شوند: یکی برای ذخیره مدل‌های `TestItemModel` و دیگری برای ذخیره مدل‌های `TestItemOptionModel`.

۳. سپس برای هر ردیف از داده‌های اکسل، اطلاعات مربوط به آیتم آزمون (`TestItemModel`) و گزینه‌های مرتبط (`TestItemOptionModel`) آماده می‌شوند.

۴. اطلاعات آیتم آزمون و گزینه‌های آن سپس با استفاده از `bulk_create` به صورت گروهی در پایگاه داده ذخیره می‌شوند.

۵. سپس نتیجه‌ی عملیات ذخیره‌سازی به عنوان پاسخ به درخواست ارسال می‌شود.

خطاها (exceptionها) نیز با استفاده از `logger.error(e)` ثبت می‌شوند و یک پاسخ خطایی به درخواست ارسال می‌شود.

این کد از تراکنش‌های پایگاه داده (`transaction.atomic`) برای اطمینان از اینکه تغییرات در پایگاه داده فقط در صورت بدون خطا بودن همه مراحل انجام شوند، استفاده می‌کند.