


# Проект: MSU Study Portal — scaffold

Ниже — полный набор начальных файлов, моделей, админов, сериализаторов, ViewSets (DRF), роутеров, шаблона `core/index.html`, management command для импорта факультетов из JSON и пример `faculties.json`.

 **Внимание:** этот документ содержит несколько файлов в одном markdown-файле. Скопируй нужные файлы в соответствующие пути проекта.

---

## README.md

```
# MSU Study Portal (Django scaffold)
```

```
Инструкции по развёртыванию описаны в конце файла.
```

---

## Файловая структура (рекомендуемая)

```
mysite/
  manage.py
  mysite/
    settings.py
    urls.py
    wsgi.py
  accounts/
    models.py
    admin.py
    serializers.py
    urls.py
    views.py
  faculties/
    models.py
    admin.py
    management/commands/import_faculties.py
    serializers.py
    urls.py
    views.py
  news/
    models.py
    admin.py
    serializers.py
    urls.py
    views.py
  schedule/
    models.py
```

```
admin.py
serializers.py
urls.py
views.py
materials/
models.py
admin.py
serializers.py
urls.py
views.py
core/
templates/core/index.html
views.py
templates/
static/
media/
faculties.json
```

## Содержимое файлов

accounts/models.py

```
from django.db import models
from django.contrib.auth.models import AbstractBaseUser, PermissionsMixin,
BaseUserManager
from django.utils import timezone

class UserManager(BaseUserManager):
    def create_user(self, student_id, password=None, **extra_fields):
        if not student_id:
            raise ValueError('student_id is required')
        user = self.model(student_id=student_id, **extra_fields)
        user.set_password(password)
        user.save(using=self._db)
        return user

    def create_superuser(self, student_id, password, **extra_fields):
        extra_fields.setdefault('is_staff', True)
        extra_fields.setdefault('is_superuser', True)
        return self.create_user(student_id, password, **extra_fields)

class User(AbstractBaseUser, PermissionsMixin):
    student_id = models.CharField(max_length=50, unique=True)
    first_name = models.CharField(max_length=150)
    last_name = models.CharField(max_length=150)
    patronymic = models.CharField(max_length=150, blank=True)
    course = models.PositiveSmallIntegerField(null=True, blank=True)
```

```

        faculty = models.ForeignKey('faculties.Faculty', null=True, blank=True,
on_delete=models.SET_NULL)
        group = models.CharField(max_length=50, blank=True)
        ROLE_CHOICES = [
            ('student', 'Студент'),
            ('starosta', 'Староста'),
            ('teacher', 'Преподаватель'),
            ('admin', 'Администратор'),
        ]
        role = models.CharField(max_length=20, choices=ROLE_CHOICES,
default='student')
        is_active = models.BooleanField(default=True)
        is_staff = models.BooleanField(default=False)
        photo = models.ImageField(upload_to='avatars/', null=True, blank=True)
        date_joined = models.DateTimeField(default=timezone.now)

        USERNAME_FIELD = 'student_id'
        REQUIRED_FIELDS = []

        objects = UserManager()

    def __str__(self):
        return f"{self.last_name} {self.first_name} ({self.student_id})"

```

#### accounts/admin.py

```

from django.contrib import admin
from django.contrib.auth.admin import UserAdmin as BaseUserAdmin
from .models import User
from django import forms

class UserChangeForm(forms.ModelForm):
    class Meta:
        model = User
        fields = '__all__'

class UserAdmin(BaseUserAdmin):
    model = User
    list_display = ('student_id', 'last_name', 'first_name', 'role', 'is_staff')
    search_fields = ('student_id', 'last_name', 'first_name', 'group')
    ordering = ('student_id',)
    fieldsets = (
        (None, {'fields': ('student_id', 'password')}),
        ('Personal', {'fields':
('first_name', 'last_name', 'patronymic', 'photo')}),
        ('University', {'fields': ('faculty', 'group', 'course', 'role')}),
        ('Permissions', {'fields':
('is_active', 'is_staff', 'is_superuser', 'groups')}),

```

```
)

admin.site.register(User, UserAdmin)
```

#### faculties/models.py

```
from django.db import models

class Faculty(models.Model):
    code = models.CharField(max_length=50, unique=True)
    name = models.CharField(max_length=255)
    description = models.TextField(blank=True)
    history = models.TextField(blank=True)
    site_url = models.URLField(blank=True)
    other_info = models.JSONField(blank=True, null=True)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.name

class PassingScore(models.Model):
    faculty = models.ForeignKey(Faculty, related_name='scores',
on_delete=models.CASCADE)
    year = models.PositiveSmallIntegerField()
    specialty = models.CharField(max_length=255, blank=True)
    min_score = models.PositiveIntegerField()

    class Meta:
        unique_together = ('faculty', 'year', 'specialty')

    def __str__(self):
        return f"{self.faculty.code} {self.year} {self.min_score}"
```

#### faculties/admin.py

```
from django.contrib import admin
from .models import Faculty, PassingScore

class PassingScoreInline(admin.TabularInline):
    model = PassingScore
    extra = 1

@admin.register(Faculty)
class FacultyAdmin(admin.ModelAdmin):
    list_display = ('code', 'name')
```

```
search_fields = ('code', 'name')
inlines = [PassingScoreInline]
```

#### faculties/management/commands/import\_faculties.py

```
from django.core.management.base import BaseCommand
import json
from faculties.models import Faculty, PassingScore

class Command(BaseCommand):
    help = 'Import faculties and passing scores from a JSON file'

    def add_arguments(self, parser):
        parser.add_argument('file', type=str, help='Path to faculties.json')

    def handle(self, *args, **options):
        path = options['file']
        with open(path, 'r', encoding='utf-8') as f:
            data = json.load(f)
            for item in data:
                f_obj, created = Faculty.objects.update_or_create(
                    code=item['code'],
                    defaults={
                        'name': item.get('name', ''),
                        'description': item.get('description', ''),
                        'history': item.get('history', ''),
                        'site_url': item.get('site_url', ''),
                        'other_info': item.get('other_info', None),
                    }
                )
                for s in item.get('passing_scores', []):
                    PassingScore.objects.update_or_create(
                        faculty=f_obj, year=s['year'],
                        specialty=s.get('specialty', ''),
                        defaults={'min_score': s['min_score']}
                    )
                self.stdout.write(self.style.SUCCESS('Imported faculties and scores'))
```

#### faculties/serializers.py

```
from rest_framework import serializers
from .models import Faculty, PassingScore

class PassingScoreSerializer(serializers.ModelSerializer):
    class Meta:
```

```

        model = PassingScore
        fields = ('year', 'specialty', 'min_score')

class FacultySerializer(serializers.ModelSerializer):
    scores = PassingScoreSerializer(many=True, read_only=True)
    class Meta:
        model = Faculty
        fields =
('id', 'code', 'name', 'description', 'history', 'site_url', 'other_info', 'scores')

```

#### faculties/views.py

```

from rest_framework import viewsets
from .models import Faculty
from .serializers import FacultySerializer
from rest_framework.permissions import AllowAny

class FacultyViewSet(viewsets.ReadOnlyModelViewSet):
    queryset = Faculty.objects.all().order_by('name')
    serializer_class = FacultySerializer
    permission_classes = [AllowAny]

```

#### faculties/urls.py

```

from rest_framework import routers
from .views import FacultyViewSet

router = routers.DefaultRouter()
router.register(r'faculties', FacultyViewSet, basename='faculty')

urlpatterns = router.urls

```

#### news/models.py

```

from django.db import models
from django.conf import settings
from django.utils import timezone

class News(models.Model):
    NEWS_TYPES = [('news', 'Новость'), ('announcement', 'Объявление'),
('event', 'Событие')]
    title = models.CharField(max_length=255)
    body = models.TextField()

```

```
type = models.CharField(max_length=20, choices=NEWS_TYPES,
default='news')
author = models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.SET_NULL, null=True)
published_at = models.DateTimeField(default=timezone.now)
pinned = models.BooleanField(default=False)

def __str__(self):
    return self.title
```

news/admin.py

```
from django.contrib import admin
from .models import News

@admin.register(News)
class NewsAdmin(admin.ModelAdmin):
    list_display = ('title', 'type', 'author', 'published_at')
    list_filter = ('type', 'published_at')
    search_fields = ('title', 'body')
```

news/serializers.py

```
from rest_framework import serializers
from .models import News

class NewsSerializer(serializers.ModelSerializer):
    author_name = serializers.CharField(source='author.__str__',
read_only=True)
    class Meta:
        model = News
        fields =
('id', 'title', 'body', 'type', 'author', 'author_name', 'published_at', 'pinned')
```

news/views.py

```
from rest_framework import viewsets, permissions
from .models import News
from .serializers import NewsSerializer

class IsAdminOrReadOnly(permissions.BasePermission):
    def has_permission(self, request, view):
        if request.method in permissions.SAFE_METHODS:
```

```

        return True
    return request.user.is_authenticated and request.user.role == 'admin'

class NewsViewSet(viewsets.ModelViewSet):
    queryset = News.objects.all().order_by('-published_at')
    serializer_class = NewsSerializer
    permission_classes = [IsAdminOrReadOnly]

```

#### news/urls.py

```

from rest_framework import routers
from .views import NewsViewSet

router = routers.DefaultRouter()
router.register(r'news', NewsViewSet, basename='news')

urlpatterns = router.urls

```

#### schedule/models.py

```

from django.db import models
from django.conf import settings

class Subject(models.Model):
    name = models.CharField(max_length=255)
    short_name = models.CharField(max_length=50, blank=True)

    def __str__(self):
        return self.name

class Lesson(models.Model):
    pair_number = models.PositiveSmallIntegerField()
    start_time = models.TimeField()
    end_time = models.TimeField()
    date = models.DateField(null=True, blank=True)
    weekday = models.PositiveSmallIntegerField(choices=[(i,i) for i in
range(1,8)], null=True, blank=True)
    subject = models.ForeignKey(Subject, on_delete=models.CASCADE)
    teacher = models.ForeignKey(settings.AUTH_USER_MODEL, null=True,
blank=True, on_delete=models.SET_NULL)
    auditorium = models.CharField(max_length=50, blank=True)
    group = models.CharField(max_length=50, blank=True)
    homework = models.TextField(blank=True)
    created_by = models.ForeignKey(settings.AUTH_USER_MODEL,
related_name='created_lessons', on_delete=models.SET_NULL, null=True)

```



```
def __str__(self):
    return f"{self.subject.name} ({self.group}) {self.pair_number}"
```

#### schedule/admin.py

```
from django.contrib import admin
from .models import Subject, Lesson

@admin.register(Subject)
class SubjectAdmin(admin.ModelAdmin):
    list_display = ('name', 'short_name')

@admin.register(Lesson)
class LessonAdmin(admin.ModelAdmin):
    list_display =
    ('subject', 'group', 'pair_number', 'date', 'weekday', 'start_time', 'end_time')
    list_filter = ('group', 'weekday', 'date')
    search_fields = ('subject__name', 'group')
```

#### schedule/serializers.py

```
from rest_framework import serializers
from .models import Subject, Lesson

class SubjectSerializer(serializers.ModelSerializer):
    class Meta:
        model = Subject
        fields = ('id', 'name', 'short_name')

class LessonSerializer(serializers.ModelSerializer):
    subject = SubjectSerializer(read_only=True)
    subject_id =
    serializers.PrimaryKeyRelatedField(queryset=Subject.objects.all(),
    write_only=True, source='subject')
    teacher_name = serializers.CharField(source='teacher.__str__',
    read_only=True)

    class Meta:
        model = Lesson
        fields =
    ('id', 'pair_number', 'start_time', 'end_time', 'date', 'weekday', 'subject', 'subject_id', 'teacher',
```

#### schedule/views.py

```
from rest_framework import viewsets, permissions
from .models import Subject, Lesson
from .serializers import SubjectSerializer, LessonSerializer

class LessonViewSet(viewsets.ModelViewSet):
    queryset = Lesson.objects.all().order_by('date', 'pair_number')
    serializer_class = LessonSerializer

    def get_permissions(self):
        if self.request.method in permissions.SAFE_METHODS:
            return [permissions.AllowAny()]
        return [permissions.IsAuthenticated()]

class SubjectViewSet(viewsets.ReadOnlyModelViewSet):
    queryset = Subject.objects.all().order_by('name')
    serializer_class = SubjectSerializer
```

#### schedule/urls.py

```
from rest_framework import routers
from .views import LessonViewSet, SubjectViewSet

router = routers.DefaultRouter()
router.register(r'lessons', LessonViewSet, basename='lesson')
router.register(r'subjects', SubjectViewSet, basename='subject')

urlpatterns = router.urls
```

#### materials/models.py

```
from django.db import models
from django.conf import settings
from faculties.models import Faculty

class Folder(models.Model):
    name = models.CharField(max_length=255)
    parent = models.ForeignKey('self', null=True, blank=True,
on_delete=models.CASCADE, related_name='children')
    faculty = models.ForeignKey(Faculty, null=True, blank=True,
on_delete=models.SET_NULL)
    created_by = models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.SET_NULL, null=True)
    created_at = models.DateTimeField(auto_now_add=True)
```

```

class Meta:
    unique_together = ('parent', 'name')

def __str__(self):
    return self.name

class Material(models.Model):
    folder = models.ForeignKey(Folder, related_name='materials',
on_delete=models.CASCADE)
    file = models.FileField(upload_to='materials/')
    uploaded_at = models.DateTimeField(auto_now_add=True)
    uploaded_by = models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.SET_NULL, null=True)
    description = models.TextField(blank=True)

def __str__(self):
    return self.file.name.split('/')[-1]

```

materials/admin.py

```

from django.contrib import admin
from .models import Folder, Material

@admin.register(Folder)
class FolderAdmin(admin.ModelAdmin):
    list_display = ('name', 'faculty', 'parent', 'created_by')

@admin.register(Material)
class MaterialAdmin(admin.ModelAdmin):
    list_display = ('file', 'folder', 'uploaded_at', 'uploaded_by')
    search_fields = ('file',)

```

materials/serializers.py

```

from rest_framework import serializers
from .models import Folder, Material

class MaterialSerializer(serializers.ModelSerializer):
    file_url = serializers.SerializerMethodField()
    uploaded_by_name = serializers.CharField(source='uploaded_by.__str__',
read_only=True)

    class Meta:
        model = Material
        fields =

```

```

('id', 'file', 'file_url', 'uploaded_at', 'uploaded_by', 'uploaded_by_name', 'description')

def get_file_url(self, obj):
    request = self.context.get('request')
    if obj.file and request:
        return request.build_absolute_uri(obj.file.url)
    if obj.file:
        return obj.file.url
    return ''

class FolderSerializer(serializers.ModelSerializer):
    children = serializers.SerializerMethodField()
    materials = MaterialSerializer(many=True, read_only=True)

    class Meta:
        model = Folder
        fields =
('id', 'name', 'parent', 'faculty', 'created_by', 'children', 'materials')

def get_children(self, obj):
    return FolderSerializer(obj.children.all(), many=True,
context=self.context).data

```

#### materials/views.py

```

from rest_framework import viewsets, permissions
from .models import Folder, Material
from .serializers import FolderSerializer, MaterialSerializer

class FolderViewSet(viewsets.ModelViewSet):
    queryset = Folder.objects.filter(parent__isnull=True).order_by('name')
    serializer_class = FolderSerializer

    def get_permissions(self):
        if self.request.method in permissions.SAFE_METHODS:
            return [permissions.AllowAny()]
        return [permissions.IsAuthenticated()]

class MaterialViewSet(viewsets.ModelViewSet):
    queryset = Material.objects.all().order_by('-uploaded_at')
    serializer_class = MaterialSerializer

    def get_permissions(self):
        if self.request.method in permissions.SAFE_METHODS:
            return [permissions.AllowAny()]
        return [permissions.IsAuthenticated()]

```

## materials/urls.py

```
from rest_framework import routers
from .views import FolderViewSet, MaterialViewSet

router = routers.DefaultRouter()
router.register(r'folders', FolderViewSet, basename='folders')
router.register(r'materials', MaterialViewSet, basename='materials')

urlpatterns = router.urls
```

## core/templates/core/index.html

```
<!doctype html>
<html lang="ru">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>MSU Study Portal</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/
bootstrap.min.css" rel="stylesheet">
  </head>
  <body>
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
      <div class="container-fluid">
        <a class="navbar-brand" href="#">MSU Portal</a>
        <div class="d-flex">
          <a class="btn btn-outline-primary me-2" href="/
schedule/">Расписание</a>
          <a class="btn btn-outline-secondary me-2" href="/
materials/">Материалы</a>
          <div class="dropdown">
            <a class="btn btn-outline-dark dropdown-toggle" href="#"
role="button" data-bs-toggle="dropdown">Профиль</a>
            <ul class="dropdown-menu dropdown-menu-end">
              <li><a class="dropdown-item" href="/profile/">Открыть профиль</
a></li>
              <li><a class="dropdown-item" href="/logout/">Выйти</a></li>
            </ul>
          </div>
        </div>
      </div>
    </nav>

    <div class="container mt-4">
      <div class="row">
        <div class="col-md-8">
          <h3>Факультеты</h3>
```

```

        <div id="faculties-list">
            <!-- сюда можно подгружать через JS / API -->
            <p>Список факультетов будет отображаться здесь</p>
        </div>
    </div>
    <div class="col-md-4">
        <h5>Полезные ссылки</h5>
        <ul>
            <li><a href="#">Личный кабинет МГУ</a></li>
            <li><a href="#">Распределение стипендий</a></li>
            <li><a href="#">Расписание экзаменов</a></li>
        </ul>
        <hr>
        <h5>Войти</h5>
        <a class="btn btn-primary" href="/accounts/login/">Войти по номеру
студ. билета</a>
    </div>
</div>
</div>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/
bootstrap.bundle.min.js"></script>
</body>
</html>

```

core/views.py

```

from django.shortcuts import render

def index(request):
    return render(request, 'core/index.html')

```

mysite/urls.py (основной маршрутизатор)

```

from django.contrib import admin
from django.urls import path, include
from core.views import index
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', index, name='index'),
    path('api/', include('rest_framework.urls')),
    path('api/', include('faculties.urls')),
    path('api/', include('news.urls')),

```

```

    path('api/', include('schedule.urls')),
    path('api/', include('materials.urls')),
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)

```

### faculties.json (пример)

```

[
  {
    "code": "VMK",
    "name": "Факультет вычислительной математики и кибернетики",
    "description": "Факультет ВМК МГУ — одно из ведущих подразделений...",
    "history": "Основан в ...",
    "site_url": "https://vmk.msu.ru/",
    "other_info": {"address": "Ленинские горы, д.1"},
    "passing_scores": [
      {"year": 2024, "specialty": "Прикладная математика", "min_score": 285},
      {"year": 2023, "specialty": "Прикладная математика", "min_score": 280}
    ]
  },
  {
    "code": "MSU_CS",
    "name": "Кафедра компьютерных наук",
    "description": "Описание...",
    "passing_scores": [
      {"year": 2024, "specialty": "Информатика", "min_score": 290},
      {"year": 2023, "specialty": "Информатика", "min_score": 288}
    ]
  }
]

```

## Настройка settings.py — ключевые правки

- Добавить приложения `accounts`, `faculties`, `news`, `schedule`, `materials`, `core`, `rest_framework`.
- Установить `AUTH_USER_MODEL = 'accounts.User'`.
- Настроить `MEDIA_ROOT` и `MEDIA_URL`:

```
MEDIA_URL = '/media/'  
MEDIA_ROOT = BASE_DIR / 'media'
```

- Для упрощения разработки установить `DEFAULT_AUTO_FIELD='django.db.models.BigAutoField'`.

- Пример добавления REST framework настроек (по желанию):

```
REST_FRAMEWORK = {  
    'DEFAULT_PERMISSION_CLASSES': [  
        'rest_framework.permissions.AllowAny',  
    ],  
    'DEFAULT_AUTHENTICATION_CLASSES': [  
        'rest_framework.authentication.SessionAuthentication',  
    ],  
}
```

---

## Быстрый старт

1. Создай виртуальное окружение и установи зависимости:

```
python -m venv venv  
source venv/bin/activate  
pip install django djangorestframework pillow
```

1. Инициализируй проект и приложения (если ещё не созданы), помести файлы как в структуре.
2. В `settings.py` добавь `AUTH_USER_MODEL = 'accounts.User'` и приложения.
3. Миграции и суперпользователь:

```
python manage.py makemigrations  
python manage.py migrate  
python manage.py createsuperuser
```

1. Импорт факультетов (пример):

```
python manage.py import_faculties faculties.json
```

1. Запусти dev-сервер:

```
python manage.py runserver
```

---



Если нужно, я могу отдельно сгенерировать: - готовые `serializers.py`/`views.py` для аккаунтов (аутентификация по `student_id`), - готовые шаблоны для расписания/материалов с минимальным JS (AJAX для API), - документацию API (OpenAPI/Redoc).

Готов внести правки и расширить.