Документация системы управления пользователями Django

Версия: 1.0
Дата: 2025
Автор: Django User Management System

Содержание

1. Введение
2. Требования к системе
3. Быстрый старт - Клонирование проекта
4. Установка и настройка
5. Архитектура проекта
6. Модели данных
7. Функциональные возможности
8. Руководство пользователя
9. API и представления
10. Безопасность
11. Решение проблем
12. Расширение функционала

Введение

Назначение системы

Система управления пользователями Django - это веб-приложение для управления двумя типами пользователей: администраторами и обычными пользователями. Система обеспечивает:

- Регистрацию и аутентификацию пользователей
- Разделение прав доступа по типам пользователей
- Персональные профили с различными наборами полей
- Управление профилями и настройками

Ключевые особенности

- **Два типа пользователей**: Администраторы и обычные пользователи
- **One-to-One профили**: Автоматическое создание профилей при регистрации
- **Защищенные маршруты**: Доступ к страницам на основе типа пользователя
- **Загрузка файлов**: Возможность загрузки аватаров
- **Адаптивный дизайн**: Современный градиентный интерфейс
- **Система сообщений**: Уведомления об успешных/неуспешных действиях

Быстрый старт - Клонирование проекта

Шаг 1: Клонирование репозитория

Проект находится в GitHub репозитории. Клонируйте ветку `web-part`:

```
# Клонирование конкретной ветки web-part
git clone -b web-part https://github.com/ваш-username/ваш-репозиторий.git

# Или клонировать весь репозиторий и переключиться на ветку
git clone https://github.com/ваш-username/ваш-репозиторий.git
cd ваш-репозиторий
git checkout web-part
```

Шаг 2: Создание виртуального окружения

После клонирования репозитория создайте виртуальное окружение:

```
# Перейдите в директорию проекта
cd ваш-репозиторий

# Создайте виртуальное окружение
python -m venv venv

# Активация виртуального окружения
# Для Windows:
venv\Scripts\activate

# Для Linux/Mac:
source venv/bin/activate
```

После активации вы увидите `(venv)` в начале командной строки.

Шаг 3: Установка зависимостей

Проект содержит файл `requirements.txt` со всеми необходимыми зависимостями:

```
# Установка всех зависимостей
pip install -r requirements.txt
```

Содержимое requirements.txt

```
Django>=4.2,<5.0
Pillow>=10.0.0
```

Основные зависимости:

- **Django** - веб-фреймворк
- **Pillow** - библиотека для работы с изображениями (для загрузки аватаров)

Проверка установки

```
# Проверьте установленные пакеты
pip list

# Проверьте версию Django
python -m django --version
```

Шаг 4: Настройка переменных окружения

Создайте файл `.env` в корне проекта (опционально, для production):

```
# .env
SECRET_KEY=ваш-секретный-ключ-django
DEBUG=True
ALLOWED_HOSTS=localhost,127.0.0.1
DATABASE_URL=sqlite:///db.sqlite3
```

Важно: Не коммитьте файл `.env` в репозиторий! Добавьте его в `.gitignore`.

Шаг 5: Настройка базы данных

Проект использует SQLite по умолчанию. Примените миграции:

```
# Создание таблиц в базе данных
python manage.py migrate
```

Вы увидите вывод подобный этому:

```
Operations to perform:
  Apply all migrations: accounts, admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying accounts.0001_initial... OK
  ...
```

Шаг 6: Создание суперпользователя

Создайте администратора для доступа к Django Admin:

```
python manage.py createsuperuser
```

Введите данные:

- **Username:** admin
- **Email:** admin@example.com
- **Password:** ваш_пароль (минимум 8 символов)

Шаг 7: Создание директорий для медиа-файлов

```
# Создайте директорию для загружаемых файлов
mkdir -p media/avatars
```

В Windows:

```
mkdir media\avatars
```

Шаг 8: Запуск сервера разработки

```
python manage.py runserver
```

Сервер запустится на `http://127.0.0.1:8000/`

Вы увидите:

```
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Шаг 9: Проверка работоспособности

Откройте браузер и перейдите по адресам:

- **Главная страница:** http://127.0.0.1:8000/
- **Вход:** http://127.0.0.1:8000/accounts/login/
- **Регистрация:** http://127.0.0.1:8000/accounts/register/
- **Django Admin:** http://127.0.0.1:8000/admin/

Краткая справка команд

```
# Клонирование проекта
git clone -b web-part https://github.com/username/repo.git
cd repo

# Настройка окружения
python -m venv venv
source venv/bin/activate  # или venv\Scripts\activate для Windows
pip install -r requirements.txt

# Настройка базы данных
python manage.py migrate
python manage.py createsuperuser

# Запуск сервера
python manage.py runserver
```

Возможные проблемы при клонировании

Проблема 1: Git не установлен

Решение:

```
# Скачайте Git с официального сайта
# Windows: https://git-scm.com/download/win
# Linux: sudo apt-get install git
# Mac: brew install git
```

Проблема 2: Ветка web-part не существует

Решение:

```
# Проверьте доступные ветки
git branch -a

# Если ветка называется по-другому, переключитесь на нее
git checkout название-ветки
```

Проблема 3: Ошибка при установке requirements.txt

Решение:

```
# Обновите pip
python -m pip install --upgrade pip

# Попробуйте установить еще раз
pip install -r requirements.txt

# Если проблема с конкретным пакетом:
pip install Django
pip install Pillow
```

Проблема 4: Python не найден

Решение:

```
# Используйте python3 вместо python
python3 -m venv venv
python3 manage.py runserver

# Или установите Python с официального сайта
# https://www.python.org/downloads/
```

Установка и настройка

Программное обеспечение

Компонент	Минимальная версия	Рекомендуемая версия
Python	3.8+	3.11+
Django	4.0+	4.2 LTS
Pillow	9.0+	10.0+
База данных	SQLite 3	PostgreSQL 14+

Системные требования

- **ОЗУ**: минимум 512 МБ
- **Диск**: минимум 100 МБ свободного места
- **Браузер**: Chrome 90+, Firefox 88+, Safari 14+, Edge 90+

Установка и настройка

> **Примечание:** Этот раздел описывает создание проекта с нуля. Если вы клонировали готовый проект из GitHub, используйте раздел Быстрый старт.

Создание проекта с нуля

Шаг 1: Создание виртуального окружения

Создание директории проекта

mkdir myproject cd myproject

Создание виртуального окружения

python -m venv venv

Активация (Windows)

venv\Scripts\activate

Активация (Linux/Mac)

source venv/bin/activate

```
#### Шаг 2: Установка зависимостей

```bash
pip install django pillow
```

Шаг 3: Создание проекта Django

```
 # Создание проекта
django-admin startproject config .

# Создание приложения
python manage.py startapp accounts
```
```

Шаг 4: Конфигурация settings.py

Добавьте в `config/settings.py`:

```
INSTALLED_APPS = [
    'accounts',  # Новое приложение
    'django.contrib.admin',
    'django.contrib.auth',
    # ... остальные приложения
]

# Кастомная модель пользователя
AUTH_USER_MODEL = 'accounts.User'

# Настройки шаблонов
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates'],
        # ...
    },
]

# Медиа файлы
MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR / 'media'

# Перенаправления
LOGIN_URL = 'login'
LOGIN_REDIRECT_URL = 'user_dashboard'
LOGOUT_REDIRECT_URL = 'login'
```

Шаг 5: Применение миграций

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

Шаг 6: Запуск сервера

```
python manage.py runserver
```

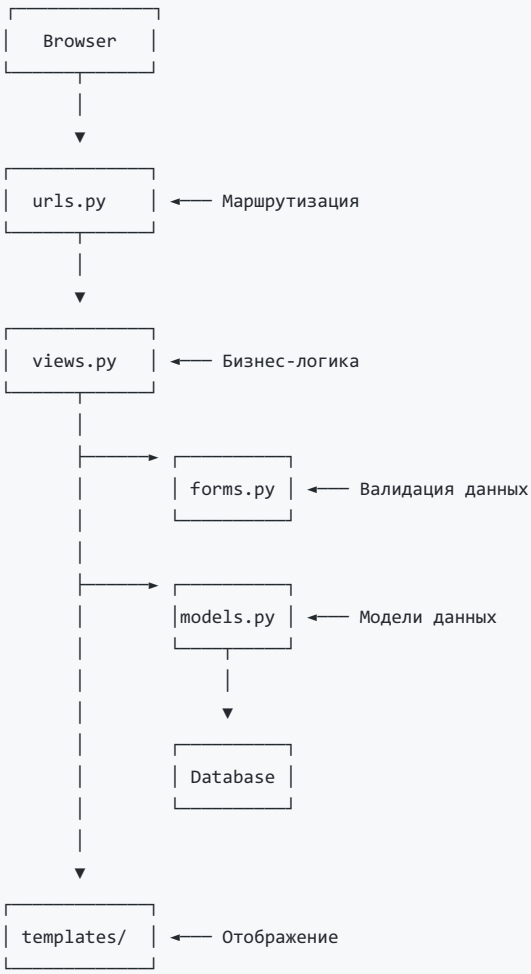
Откройте браузер: `http://127.0.0.1:8000/`

Архитектура проекта

Структура файлов

```
myproject/
├── config/                    # Главный проект
│   ├── __init__.py
│   ├── settings.py            # Настройки проекта
│   ├── urls.py                # Главные URL
│   ├── wsgi.py
│   └── asgi.py
├── accounts/                  # Приложение пользователей
│   ├── migrations/
│   ├── __init__.py
│   ├── admin.py               # Админ-панель
│   ├── apps.py
│   ├── models.py              # Модели данных
│   ├── forms.py               # Формы
│   ├── views.py               # Представления
│   ├── urls.py                # URL приложения
│   └── tests.py
├── templates/                 # HTML шаблоны
│   ├── base.html
│   └── accounts/
│       ├── register.html
│       ├── login.html
│       ├── admin_dashboard.html
│       ├── admin_profile_edit.html
│       ├── user_dashboard.html
│       ├── user_profile_edit.html
│       └── access_denied.html
├── media/                     # Загруженные файлы
│   └── avatars/
├── static/                    # Статические файлы
├── manage.py
└── db.sqlite3                 # База данных
```

Диаграмма взаимодействия компонентов



Модели данных

User (Пользователь)

Расширенная модель пользователя Django.

Поля:

Поле	Тип	Описание
username	CharField(150)	Уникальное имя пользователя
email	EmailField	Уникальный email
user_type	CharField(20)	Тип пользователя (admin/ordinary)
password	CharField(128)	Хешированный пароль
is_active	BooleanField	Активен ли пользователь
date_joined	DateTimeField	Дата регистрации

Методы:

- `__str__()` : Возвращает строковое представление пользователя
- `get_user_type_display()` : Возвращает читаемое название типа

AdminProfile (Профиль администратора)

One-to-One связь с моделью User для администраторов.

Поля:

Поле	Тип	Описание
user	OneToOneField	Связь с User
department	CharField(100)	Отдел
phone	CharField(20)	Телефон
permissions_level	IntegerField	Уровень доступа (1-3)
access_code	CharField(50)	Код доступа
created_at	DateTimeField	Дата создания

Уровни доступа:

1. Базовый
2. Расширенный
3. Полный доступ

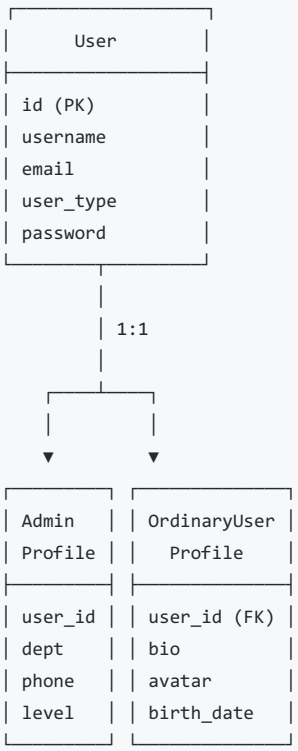
OrdinaryUserProfile (Профиль обычного пользователя)

One-to-One связь с моделью User для обычных пользователей.

Поля:

Поле	Тип	Описание
user	OneToOneField	Связь с User
bio	TextField(500)	Биография
avatar	ImageField	Аватар пользователя
birth_date	DateField	Дата рождения
phone	CharField(20)	Телефон
city	CharField(100)	Город
subscription_active	BooleanField	Активна ли подписка
created_at	DateTimeField	Дата создания

ER-диаграмма



Функциональные возможности

1. Регистрация пользователей

URL: `/accounts/register/`

Функционал:

- Выбор типа пользователя (администратор/обычный)
- Валидация email на уникальность
- Автоматическое создание соответствующего профиля
- Автоматический вход после регистрации

Валидация:

- Имя пользователя: уникальное, 1-150 символов
- Email: уникальный, валидный формат
- Пароль: минимум 8 символов, не полностью цифровой

2. Аутентификация

URL: `/accounts/login/`

Функционал:

- Вход по имени пользователя и паролю
- Автоматическое перенаправление на соответствующую панель
- Сообщения об ошибках при неверных данных

Перенаправления:

- Администратор → `/accounts/admin-dashboard/`
- Обычный пользователь → `/accounts/dashboard/`

3. Панель администратора

URL: `/accounts/admin-dashboard/`
Требования: Авторизация + user_type = 'admin'

Отображаемая информация:

- Общее количество пользователей
- Количество администраторов
- Информация профиля администратора
- Статистика системы

4. Панель пользователя

URL: `/accounts/dashboard/`
Требования: Авторизация + user_type = 'ordinary'

Отображаемая информация:

- Аватар пользователя (или заглушка)
- Личные данные
- Статус подписки
- Биография и контактная информация

5. Редактирование профилей

Администратор: `/accounts/admin-profile/edit/`

Редактируемые поля:

- Отдел
- Телефон
- Уровень доступа
- Код доступа

Обычный пользователь: `/accounts/profile/edit/`

Редактируемые поля:

- Биография
- Аватар
- Дата рождения
- Телефон
- Город
- Статус подписки

6. Выход из системы

URL: `/accounts/logout/`

Функционал:

- Завершение сессии
- Перенаправление на страницу входа
- Уведомление об успешном выходе

Руководство пользователя

Для обычных пользователей

Регистрация

1. Откройте `/accounts/register/`
2. Заполните форму:
    - Имя пользователя
    - Email
    - Выберите "Ordinary User"
    - Введите пароль дважды
3. Нажмите "Зарегистрироваться"
4. Вы будете автоматически перенаправлены в свой профиль

Вход в систему

1. Откройте `/accounts/login/`
2. Введите имя пользователя и пароль
3. Нажмите "Войти"

Просмотр профиля

1. После входа вы окажетесь на странице профиля
2. Здесь вы увидите:
    - Свой аватар
    - Личную информацию
    - Статус подписки

Редактирование профиля

1. На странице профиля нажмите "Редактировать профиль"
2. Измените необходимые поля
3. Для загрузки аватара нажмите "Choose File"

4. Нажмите "Сохранить"

Для администраторов

Регистрация

1. Откройте `/accounts/register/`
2. Выберите тип "Administrator"
3. Заполните остальные поля
4. После регистрации вы попадете на панель администратора

Панель управления

На панели администратора вы увидите:

- **Статистику**: количество пользователей и администраторов
- **Информацию профиля**: отдел, телефон, уровень доступа
- Кнопку для редактирования профиля

Редактирование профиля

1. Нажмите "Редактировать профиль"
2. Измените:
    - Отдел
    - Телефон
    - Уровень доступа (1-3)
    - Код доступа
3. Нажмите "Сохранить"

Общие функции

Навигация

В верхней панели навигации:

- Имя пользователя и тип
- Ссылка на профиль/панель
- Ссылка на редактирование
- Кнопка выхода

Сообщения системы

Система отображает уведомления:

- ✅ **Успех** (зеленый): действие выполнено успешно
- ❌ **Ошибка** (красный): произошла ошибка
- ℹ️ **Информация** (синий): информационное сообщение

API и представления

Список представлений (Views)

register_view

```
def register_view(request)
```

Описание: Регистрация нового пользователя

Метод: GET, POST

Параметры POST:

- username
- email
- user_type
- password1
- password2

Возвращает:

- GET: Форма регистрации
- POST: Перенаправление на dashboard

Права доступа: Публичный доступ

---

login_view

```
def login_view(request)
```

Описание: Аутентификация пользователя

Метод: GET, POST

Параметры POST:

- username
- password

Возвращает:

- GET: Форма входа
- POST: Перенаправление на соответствующий dashboard

Права доступа: Публичный доступ

---

admin_dashboard

```
@login_required
@user_passes_test(is_admin, login_url='/access-denied/')
def admin_dashboard(request)
```

Описание: Панель администратора

Метод: GET

Контекст:

- profile: AdminProfile
- total_users: int
- total_admins: int

Права доступа: Только администраторы

---

user_dashboard

```
@login_required
@user_passes_test(is_ordinary, login_url='/access-denied/')
def user_dashboard(request)
```

Описание: Профиль обычного пользователя

Метод: GET

Контекст:

- profile: OrdinaryUserProfile

Права доступа: Только обычные пользователи

Декораторы и проверки

is_admin

```
def is_admin(user):
    return user.is_authenticated and user.user_type == 'admin'
```

Проверяет, является ли пользователь администратором.

is_ordinary

```
def is_ordinary(user):
    return user.is_authenticated and user.user_type == 'ordinary'
```

Проверяет, является ли пользователь обычным пользователем.

Безопасность

Реализованные меры безопасности

1. Хеширование паролей

Пароли хранятся в виде хешей с использованием PBKDF2 (Django по умолчанию):

```
# Django автоматически хеширует пароли
user.set_password('plain_password')
```

2. CSRF защита

Все формы защищены от CSRF-атак:

```
<form method="post">
    {% csrf_token %}
    <!-- поля формы -->
</form>
```

3. Проверка прав доступа

Использование декораторов для защиты представлений:

```
@login_required  # Требует авторизации
@user_passes_test(is_admin)  # Требует роль администратора
def protected_view(request):
    pass
```

4. Валидация данных

Все данные валидируются на уровне форм:

```
class UserRegistrationForm(UserCreationForm):
    email = forms.EmailField(required=True)
    # Автоматическая валидация Django
```

5. SQL Injection защита

Django ORM автоматически экранирует все запросы:

```
# Безопасно
User.objects.filter(username=user_input)
```

Рекомендации по безопасности

Для production окружения:

1. **Установите SECRET_KEY**

```
# Не используйте стандартный ключ!
SECRET_KEY = 'ваш-уникальный-секретный-ключ'
```

2. **Отключите DEBUG**

```
DEBUG = False
ALLOWED_HOSTS = ['yourdomain.com']
```

3. **Используйте HTTPS**

```
SECURE_SSL_REDIRECT = True
SESSION_COOKIE_SECURE = True
CSRF_COOKIE_SECURE = True
```

4. **Настройте CORS**

```
pip install django-cors-headers
# Настройте в settings.py
```

5. **Регулярно обновляйте зависимости**

```
pip list --outdated
pip install --upgrade django
```

Решение проблем

Частые ошибки и их решение

Ошибка 1: "No such table: accounts_user"

Причина: Миграции не применены

Решение:

```
python manage.py makemigrations accounts
python manage.py migrate
```

Ошибка 2: "AUTH_USER_MODEL не настроен"

Причина: Не указана кастомная модель пользователя

Решение: Добавьте в `settings.py`:

```
AUTH_USER_MODEL = 'accounts.User'
```

Затем удалите базу данных и миграции, пересоздайте:

```
rm db.sqlite3
rm -rf accounts/migrations
python manage.py makemigrations accounts
python manage.py migrate
```

Ошибка 3: "Access Denied" для всех пользователей

Причина: Неправильная проверка прав доступа

Решение: Проверьте, что user_type установлен правильно:

```
user = User.objects.get(username='test')
print(user.user_type)  # Должно быть 'admin' или 'ordinary'
```

Ошибка 4: Изображения не загружаются

Причина: MEDIA настройки не сконфигурированы

Решение:

1. Проверьте `settings.py`:

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

2. Проверьте `urls.py`:

```
from django.conf import settings
from django.conf.urls.static import static

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

Ошибка 5: "RelatedObjectDoesNotExist: User has no admin_profile"

Причина: Профиль не создался автоматически

Решение: Создайте профиль вручную:

```
from accounts.models import User, AdminProfile, OrdinaryUserProfile

user = User.objects.get(username='username')
if user.user_type == 'admin':
    AdminProfile.objects.create(user=user)
else:
    OrdinaryUserProfile.objects.create(user=user)
```

Расширение функционала

Возможные улучшения

1. Восстановление пароля

Добавьте функционал восстановления пароля через email:

```
# В views.py
from django.contrib.auth.views import PasswordResetView

# В urls.py
path('password-reset/', PasswordResetView.as_view(), name='password_reset'),
```

2. Подтверждение email

Используйте django-allauth для верификации email:

```
pip install django-allauth
```

3. API с Django REST Framework

Создайте REST API для мобильных приложений:

```
pip install djangorestframework
```

```
# serializers.py
from rest_framework import serializers

class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = ['id', 'username', 'email', 'user_type']
```

4. Социальная аутентификация

Добавьте вход через Google/Facebook:

```
pip install django-allauth
```

5. Двухфакторная аутентификация

Усильте безопасность с помощью 2FA:

```
pip install django-two-factor-auth
```

6. Расширенные права доступа

Используйте django-guardian для объектных разрешений:

```
pip install django-guardian
```

Примеры кастомизации

Добавление нового поля в профиль

1. Измените модель:

```
class OrdinaryUserProfile(models.Model):
    # ... существующие поля
    company = models.CharField(max_length=200, blank=True)
```

2. Создайте миграцию:

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

3. Обновите форму:

```
class OrdinaryUserProfileForm(forms.ModelForm):
    class Meta:
        fields = (..., 'company')
```

Приложения

Приложение А: Полезные команды Django

```
# Создание нового приложения
python manage.py startapp app_name

# Создание супербользователя
python manage.py createsuperuser

# Запуск shell
python manage.py shell

# Сбор статических файлов
python manage.py collectstatic

# Проверка проекта
python manage.py check

# Создание дампа базы данных
python manage.py dumpdata > backup.json

# Загрузка дампа
python manage.py loaddata backup.json
```

Приложение B: Полезные ссылки

- Django Documentation
- Django Tutorial
- Django Girls Tutorial
- Two Scoops of Django
- Django Packages

Приложение C: Глоссарий

MTV (Model-Template-View) - архитектурный паттерн Django (аналог MVC)

ORM (Object-Relational Mapping) - технология связывания объектов с базой данных

Миграция - файлы, описывающие изменения в схеме базы данных

Middleware - компоненты для обработки запросов/ответов

QuerySet - ленивая коллекция объектов из базы данных

Slug - URL-дружественная версия строки

Заключение

Данная система предоставляет надежную основу для управления пользователями с разными уровнями доступа. Она легко расширяема и может быть адаптирована под различные бизнес-требования.

Контакты поддержки

Для получения помощи:

- Документация Django: https://docs.djangoproject.com/
- Stack Overflow: https://stackoverflow.com/questions/tagged/django
- Django Forum: https://forum.djangoproject.com/

Версия документа: 1.0
Последнее обновление: 2025