# Лабораторная работа: Работа со страницами регистрации и авторизации пользователей в Django

## Цель работы:

Научиться создавать страницы регистрации и авторизации пользователей в Django, используя встроенные механизмы аутентификации, формы и представления для обработки пользовательских данных.

## Описание задания:

Создать Django-приложение с функциональностью регистрации и авторизации пользователей, разработать формы для ввода данных, настроить представления для обработки регистрации и входа, а также реализовать перенаправление и доступ к защищённым страницам.

## Инструменты:

- Установленная среда Python
- Установленный Django
- Текстовый редактор (например, PyCharm, Visual Studio Code)

## Ход работы:

### 1. Создание проекта и приложения Django:

Создайте новый проект Django, используя команду:
```bash
django-admin startproject myproject
cd myproject
```
Создайте новое приложение, например `accounts`, командой:
```bash
python manage.py startapp accounts
```
Добавьте `accounts` в список `INSTALLED\_APPS` в `settings.py`:
```python
# myproject/settings.py
INSTALLED\_APPS = [
 ...,
 'accounts',
]
```

### 2. Настройка URL перенаправлений:

В `settings.py` настройте URL перенаправлений после успешного входа и выхода:
```python
LOGIN\_REDIRECT\_URL = 'home'
LOGOUT\_REDIRECT\_URL = 'login'
```

### 3. Создание форм для регистрации и авторизации:

В приложении `accounts` создайте файл `forms.py` и добавьте форму для регистрации:
```python
# accounts/forms.py
from django import forms
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm

class UserRegisterForm(UserCreationForm):
 email = forms.EmailField()

 class Meta:
 model = User
 fields = ['username', 'email', 'password1', 'password2']
```

### 4. Создание представления для регистрации:

В `views.py` создайте представление `register`, которое обрабатывает регистрацию пользователя:
```python
# accounts/views.py
from django.shortcuts import render, redirect
from django.contrib import messages
from .forms import UserRegisterForm

def register(request):
 if request.method == 'POST':
 form = UserRegisterForm(request.POST)
 if form.is\_valid():
 form.save()
 username = form.cleaned\_data.get('username')
 messages.success(request, f'Аккаунт создан для {username}!')
 return redirect('login')
 else:
 form = UserRegisterForm()
 return render(request, 'accounts/register.html', {'form': form})
```

### 5. Создание шаблона для регистрации:

Создайте директорию `templates/accounts` и добавьте туда файл `register.html` для страницы регистрации:
```html
<!-- templates/accounts/register.html -->
<h2>Регистрация</h2>
<form method="POST">
 {% csrf\_token %}
 {{ form.as\_p }}
 <button type="submit">Зарегистрироваться</button>
</form>
```

### 6. Настройка URL маршрутов:

Создайте файл `urls.py` в приложении `accounts` и добавьте маршруты для регистрации и аутентификации:
```python
# accounts/urls.py
from django.urls import path
from django.contrib.auth import views as auth\_views
from . import views

urlpatterns = [
 path('register/', views.register, name='register'),
 path('login/', auth\_views.LoginView.as\_view(template\_name='accounts/login.html'), name='login'),
 path('logout/', auth\_views.LogoutView.as\_view(template\_name='accounts/logout.html'), name='logout'),
]

# myproject/urls.py
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
 path('admin/', admin.site.urls),
 path('accounts/', include('accounts.urls')),
 path('', include('main.urls')), # Подключение других приложений, если необходимо
]
```

### 7. Создание шаблонов для входа и выхода:

Создайте файл `login.html` для страницы входа:
```html
<!-- templates/accounts/login.html -->
<h2>Вход</h2>
<form method="POST">
 {% csrf\_token %}
 {{ form.as\_p }}
 <button type="submit">Войти</button>
</form>
```

Создайте файл `logout.html` для страницы выхода:
```html
<!-- templates/accounts/logout.html -->
<h2>Вы успешно вышли из аккаунта!</h2>
<a href="{% url 'login' %}">Войти снова</a>
```

### 8. Создание защищенной страницы для авторизованных пользователей:

Создайте представление, доступное только авторизованным пользователям:
```python
# accounts/views.py
from django.contrib.auth.decorators import login\_required

@login\_required
def profile(request):
 return render(request, 'accounts/profile.html')
```
Добавьте маршрут для страницы `profile` и создайте шаблон `profile.html`:
```python
# accounts/urls.py
urlpatterns += [
 path('profile/', views.profile, name='profile'),
]

<!-- templates/accounts/profile.html -->
<h2>Профиль пользователя</h2>
<p>Добро пожаловать, {{ user.username }}!</p>
```

### 9. Проверка работы:

Запустите сервер и откройте страницу `/accounts/register/` для регистрации нового пользователя.
Проверьте, что пользователь может войти через `/accounts/login/` и получить доступ к странице `/accounts/profile/`, которая требует авторизации.

## Контрольные вопросы:

- Какой класс используется для создания формы регистрации в Django?

- Как работает декоратор `login\_required`, и как он защищает доступ к представлению?

- Какие URL перенаправления можно настроить для входа и выхода?

- Как хранить сообщения об успешной регистрации или ошибке на страницах?

## Дополнительное задание:

Добавьте форму для редактирования профиля пользователя, позволяющую изменить его имя и адрес электронной почты, и настройте представление для обновления данных пользователя.