**Лабораторная работа № 11**

**ОСНОВЫ РАБОТЫ С ВЕБ-ФРЕЙМВОРКОМ DJANGO**

**Цель работы:** рассмотреть возможности веб-фреймворка Django для построения веб-приложений на языке Python, ознакомиться с понятием ORM и его реализации в данном фрейм-ворке.

**Краткая теория**

Django – это веб-фреймворк для Python, основным преимуществом которого является высокая скорость разработки и «чистая» архитектура проекта. Данный фреймворк является open-source проектом. Архитектура Django предоставляет возможность быстрого и гибкого масштабирования. Данный фреймворк несет в себе множество дополнительного функционала «из коробки», такого как аутентификация пользователей, администрирования контента, панель администратора для работы с базой данных и многое другое. Для работы с базой данных Django, как и многие другие веб-фреймворки используют ORM (Object-Relational Mapping). Это технология, позволяющая связать базы данных с парадигмой ООП. Данная технология позволяет не только не привязываться к определенной базе данных, но и генерировать архитектуру базы данных из написанных классов. В ORM предполагается, что каждая таблица является классом, а каждый атрибут (поле) класса является атрибутом таблицы. Таким образом, ORM представляет собой как бы прослойку между БД и ООП.

**Создание проекта на Django.**

Перед началом создания проекта требуется установить сам веб-фреймворк. Это делают аналогично с установкой любой библиотеки через менеджера pip. Данная команда выполняет скачивание и установку фреймворка: pip install django. После установки фреймворка можно приступить к созданию самого проекта. Для этого в командной строке или терминале требуется перейти в нужную директорию, где будет создаваться проект и выполнить команду django-admin 91 startproject newproject, где newproject наименование создаваемого проекта. Данная команда создаст следующую структуру директорий и файлов: − newproject – корневая директория проекта. − manage.py – утилита командной строки, позволяющая взаимодействовать с проектом различными способами; − newproject – директория, которая является основной для проекте и хранит в себе файл конфигурации (settings.py), менеджер URL(urls.py) и точку входа для WSGI-совместимых вебсерверов (wsgi.py). В отличие от других платформ для разработки вебприложений Django «из коробки» имеет собственный однопоточный веб-сервер для разработки приложения. Запустить на нем созданный проект можно, находясь в корневой директории проекта, с помощью команды python manage.py runserer. После чего при успешном старте сервера веб-приложение будет доступно по адресу http://127.0.0.1:8000/. Для остановки вебсервера достаточно будет нажать комбинацию клавиш Ctrl+C, находясь в командной строке или терминале, где была введена предыдущая команда. Стоит отметить, что данный механизм не следует использовать на этапе работы приложения, а только во время его разработки. Создание приложения в созданном проекте Приложением называется модуль Django проекта. В них заключается мощь «чистой» архитектуры, которую предоставляет данный веб-фреймворк. Приложения удобно можно создавать самостоятельно, а можно заимствовать модули из иных проектов. Следует отметить, что для более правильной работы с предоставляемой архитектурой необходимо разделять создаваемые или заимствованные модули и стараться не допускать зависимостей между ними. Изолированные модули в будущем можно будет легко интегрировать в другие проекты, а ненужные удалять из проекта, не принося ущерба работоспособности приложения. Итак, для того, чтобы создать новое приложение в созданном проекте необходимо, находясь в корневой директории, вы- 92 полнить команду python manage.py startapp newapp, где newapp наименование создаваемого приложения. Данная команда создаст новую директорию newapp, которая содержит следую щую структуру: − migrations – директория, которая в будущем будет содержать информацию о миграциях; − admin.py – файл для настройки отображения созданных моделей на панели администратора; − apps.py – файл для конфигурации самого приложения; − models.py – файл для создания моделей; − tests.py – файл для автоматизированных тестов; − views.py – файл, содержащий бизнес-логику приложения. К выше приведенной структуре следует и отнести файл urls.py, который создается вручную и аналогично одноименному файлу из основной директории проекта представляет собой менеджер URL для созданного приложения. После этого необходимо добавить созданное приложение в проект, добавив его наименование в список INSTALLED\_APPS файла settings.py. Работа с базой данных Ранее было сказано о миграциях и моделях. Под моделями в Django следует понимать классы, описывающие таблицы базы данных. Стандартная модель должна наследоваться от класса models.Model и содержать определенный набор атрибутов, причем каждому атрибуту присваивается определенный тип. Стоит отметить, что атрибут суррогатного ключа (id) устанавливается по умолчанию. Ниже приведены несколько типов для объявления полей в моделях: − CharField() – строковый тип, предназначенный для небольших строк; − TextField() – тип для хранения больших объемов текста; − IntegerField(), FloatField() – типы для хранения целых чисел и числе с плавающей точкой соответственно; − ImageField() – тип для хранения изображений; − EmailField() – тип для хранения адресов электронной почти; 93 − DateField(), DateTimeField() – типы для хранения даты и даты и времени соответственно; − ForeignKey() – тип для связки моделей по внешнему ключу. Требует в качестве аргумента для параметра to наименование модели, с которой происходит связь (для рекурсивной зависимости следует использовать строковое значение ‘self’) и аргумент для параметра on\_delete задающий логику действий при удалении зависимой записи.

**Шаблоны**

Под шаблонами в Django понимаются динамически сгенерированные HTML страницы. Шаблоны содержат как статическую часть, так и специальный синтаксис, описывающий, где и как будет располагаться динамический контент. Для этого используются шаблонизаторы. В Django можно использовать один или несколько шаблонизаторов. Встроенный шаблонизатор в Django называется Django template language (DTL). Настройка поддержки DTL устанавливается при создании проекта. Хранить создаваемые шаблоны необходимо в директориях под названием templates. Желательно хранить шаблоны, используемые приложениями в их директориях, а в самой директории templates создавать директорию с наименованием приложения. Это способствует устранению ошибок, т.к. при рендере шаблона фреймворк ищет его по указанному пути во всех директориях под названием templates. Синтаксис DTL содержит четыре конструкции: − переменные. Вывод значений из контекста, обрамляемый двумя фигурными скобками «{{ }}»; − тэги. Предназначены для обеспечения логики в процессе рендера. Например, использование условий или циклов. Обрамляются фигурной скобкой и знаком процента «{% %}»; − фильтры. Трансформируют значения переменных и аргументы тэгов. Указываются через вертикальную черту после значения, которое необходимо изменить « | » − комментарии. Однострочные обрамляются фигурной скобкой и знаком решетки «{# #}», а многострочные выглядят следующим образом «{% comment %} … {% comment %}»

Представления

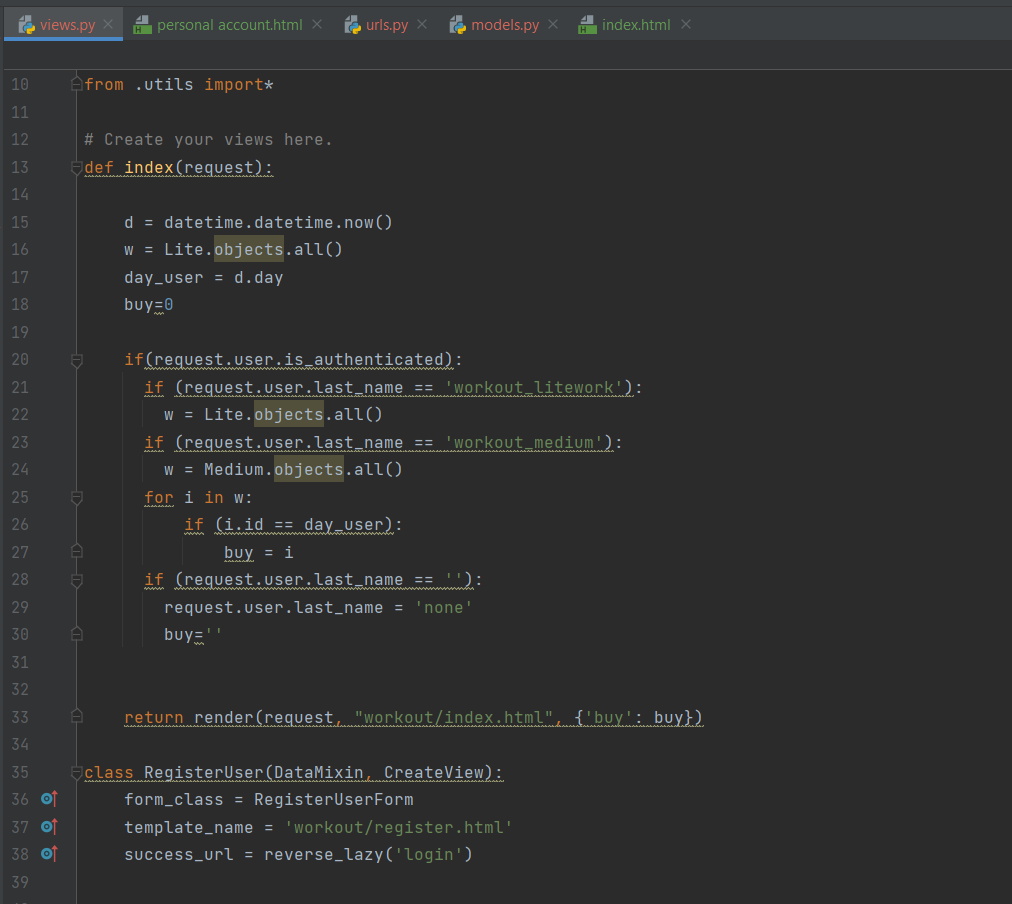
Однако для того, чтобы провести рендер страницы, необходимо вызывать его из функции, находящейся в файле views.py. Функции, которые исполняются при вызове определенного URL, называются представлениями и указываются в файле views.py. Для этого, в файле urls.py, лежащем в корневой директории проекта, требуется указать адрес доступа к urls, находящимся в директории разработанного приложения. Например, в рассматриваемом примере требуется добавить в список urlpatterns строку path('blog/', include('blog.urls')). После этого требуется указать адрес и функцию, которая будет выполняться при обращении к заданному адресу.

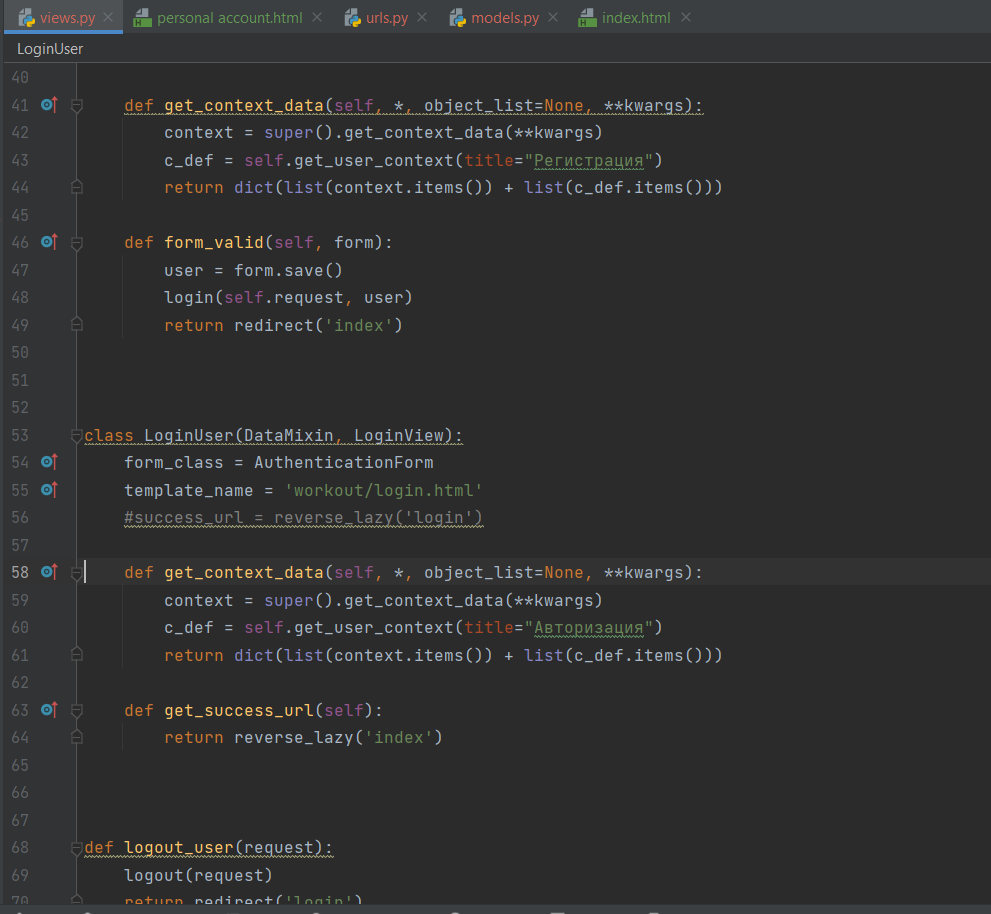
**Задание**

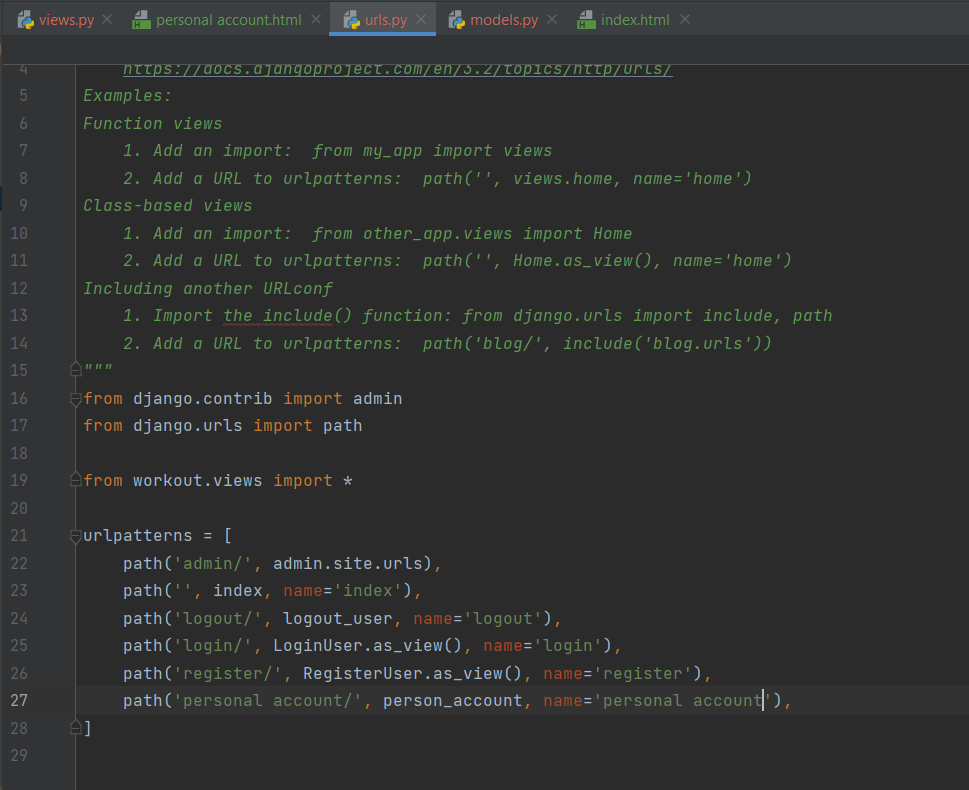
Написать веб-приложение для фитнеса, которое каждый день выдает очередную подборку тренировок для зарегистрированных пользователей. Типы тренировок можно выбирать пользователями в личном кабинете.

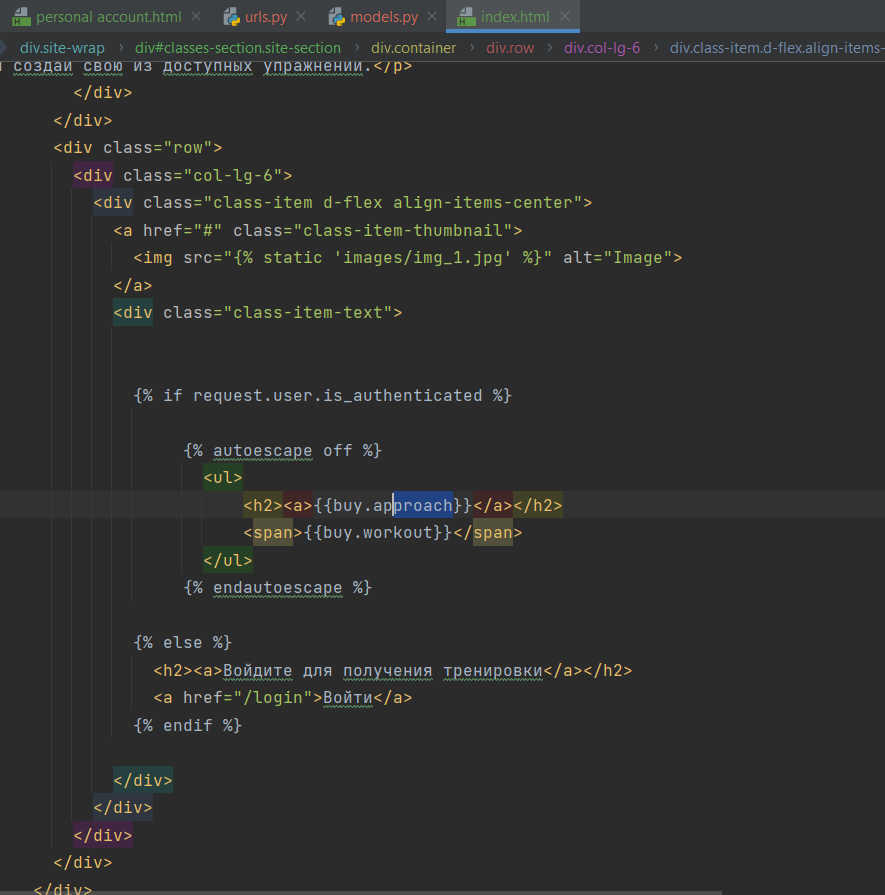
Листинг

from django.db import models  
from django.contrib.auth.models import AbstractUser  
  
# Create your models here.  
class Lite(models.Model):  
 workout = models.CharField(max\_length=50)  
 approach = models.CharField(max\_length=50)  
  
  
class Medium(models.Model):  
 workout = models.CharField(max\_length=50)  
 approach = models.CharField(max\_length=50)









Результат работы программы представлен на рисунке 10.1-10.6

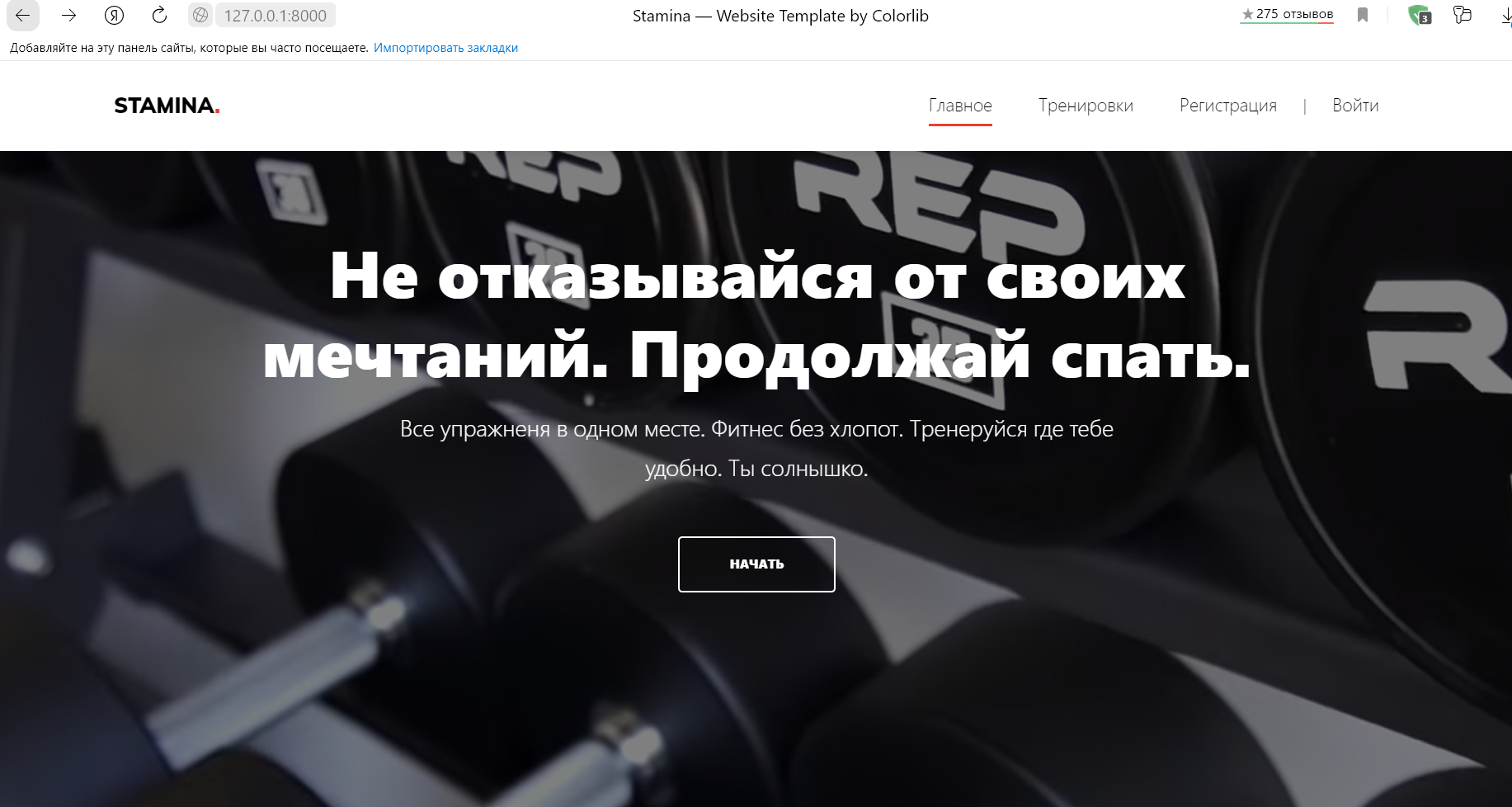


Рисунок 10.1 – результат работы программы.

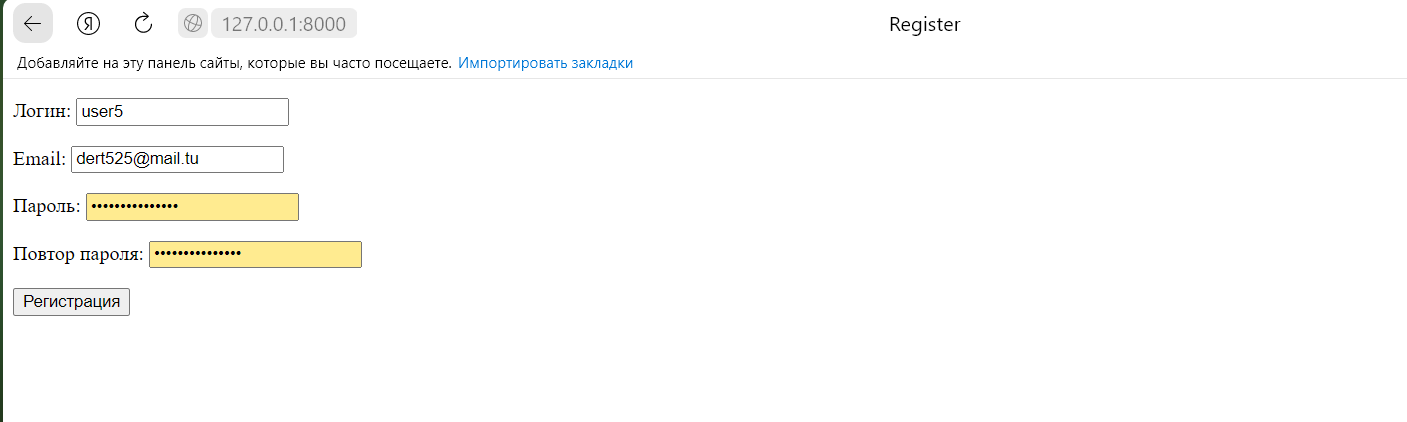


Рисунок 10.2 – результат работы программы.

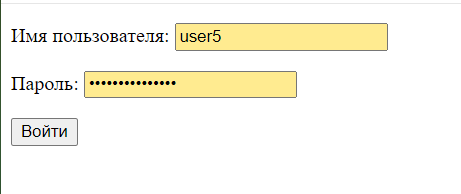


Рисунок 10.3 – результат работы программы.

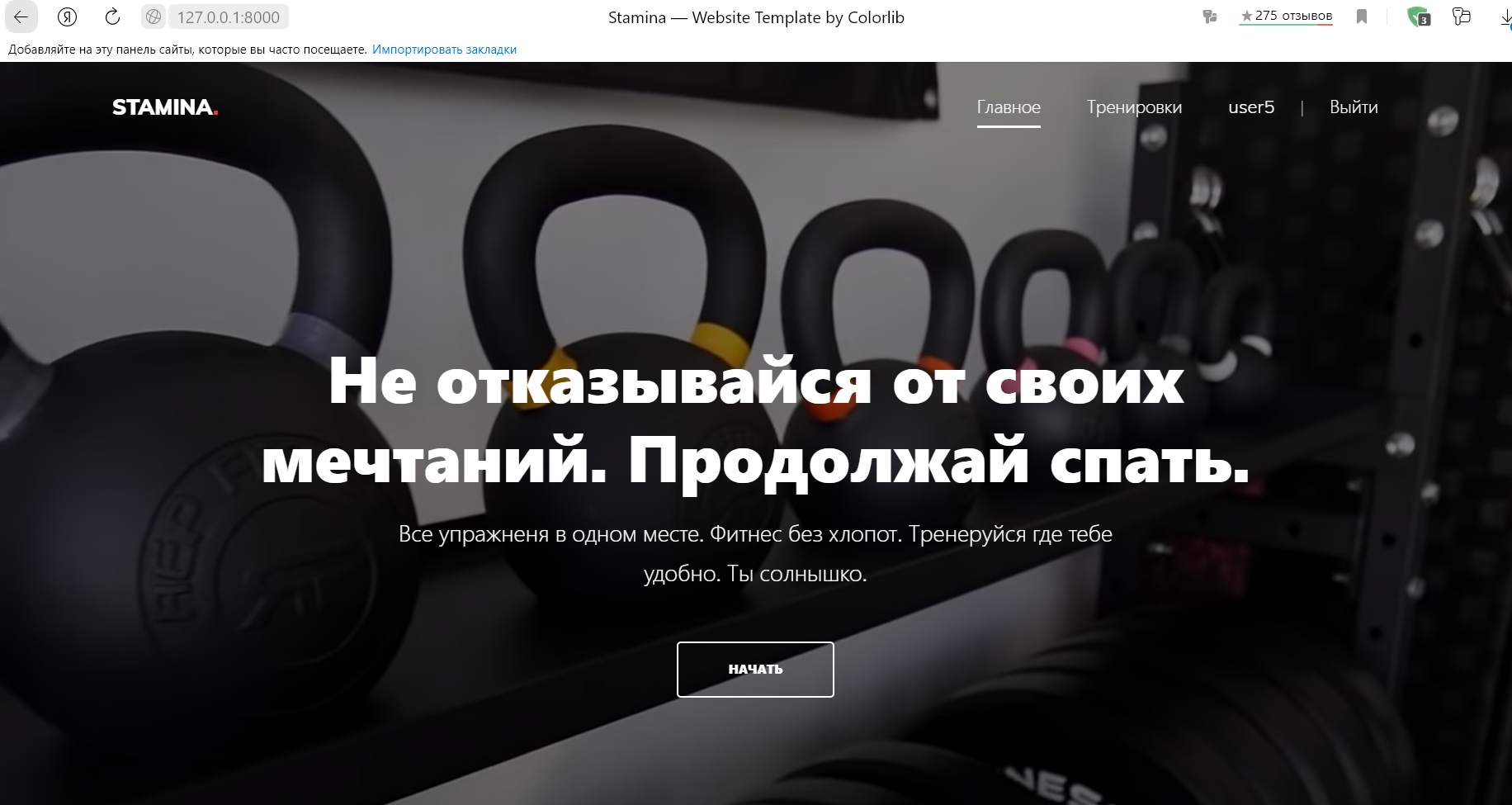


Рисунок 10.4 – результат работы программы.

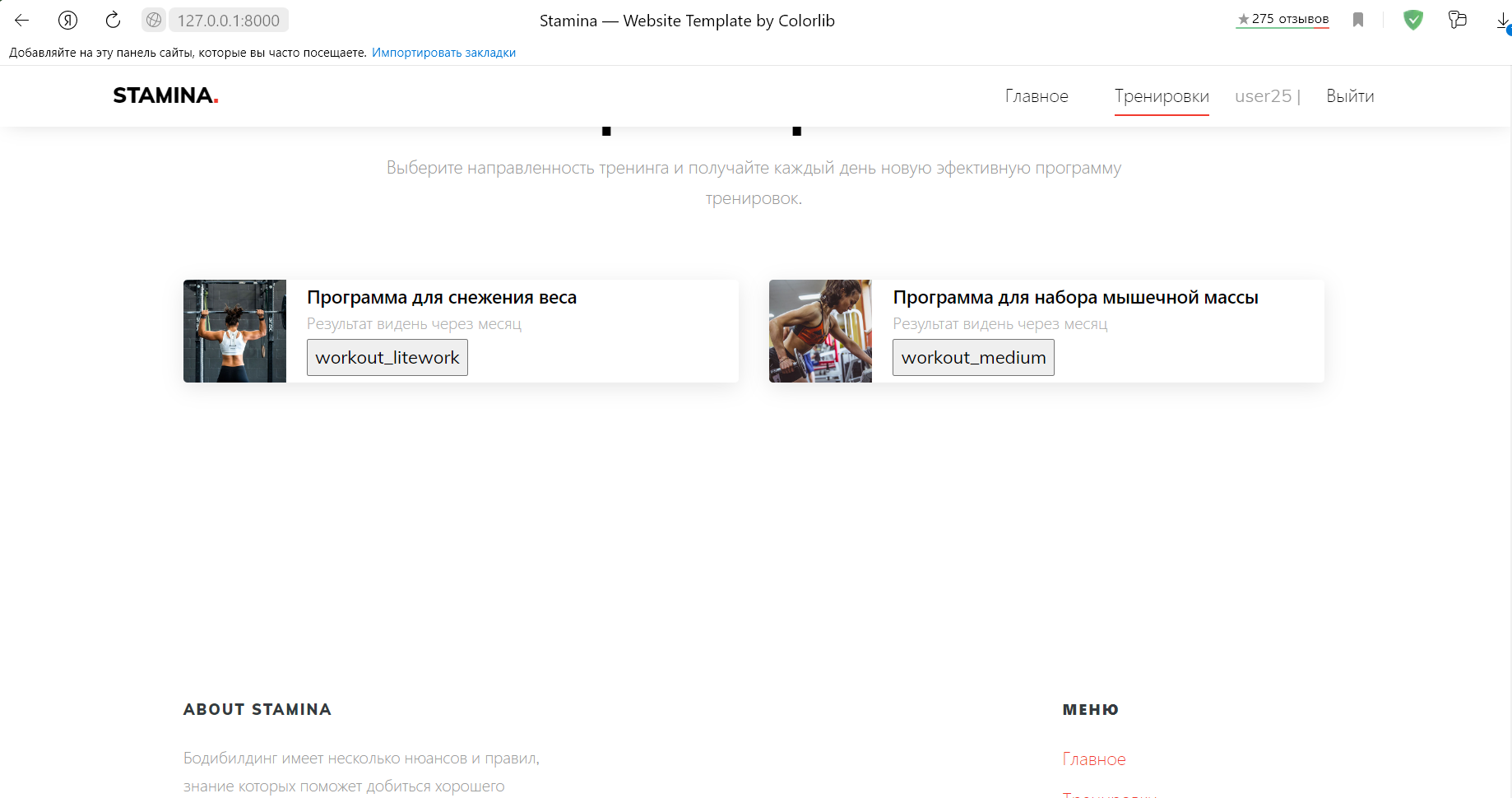


Рисунок 10.5 – результат работы программы.

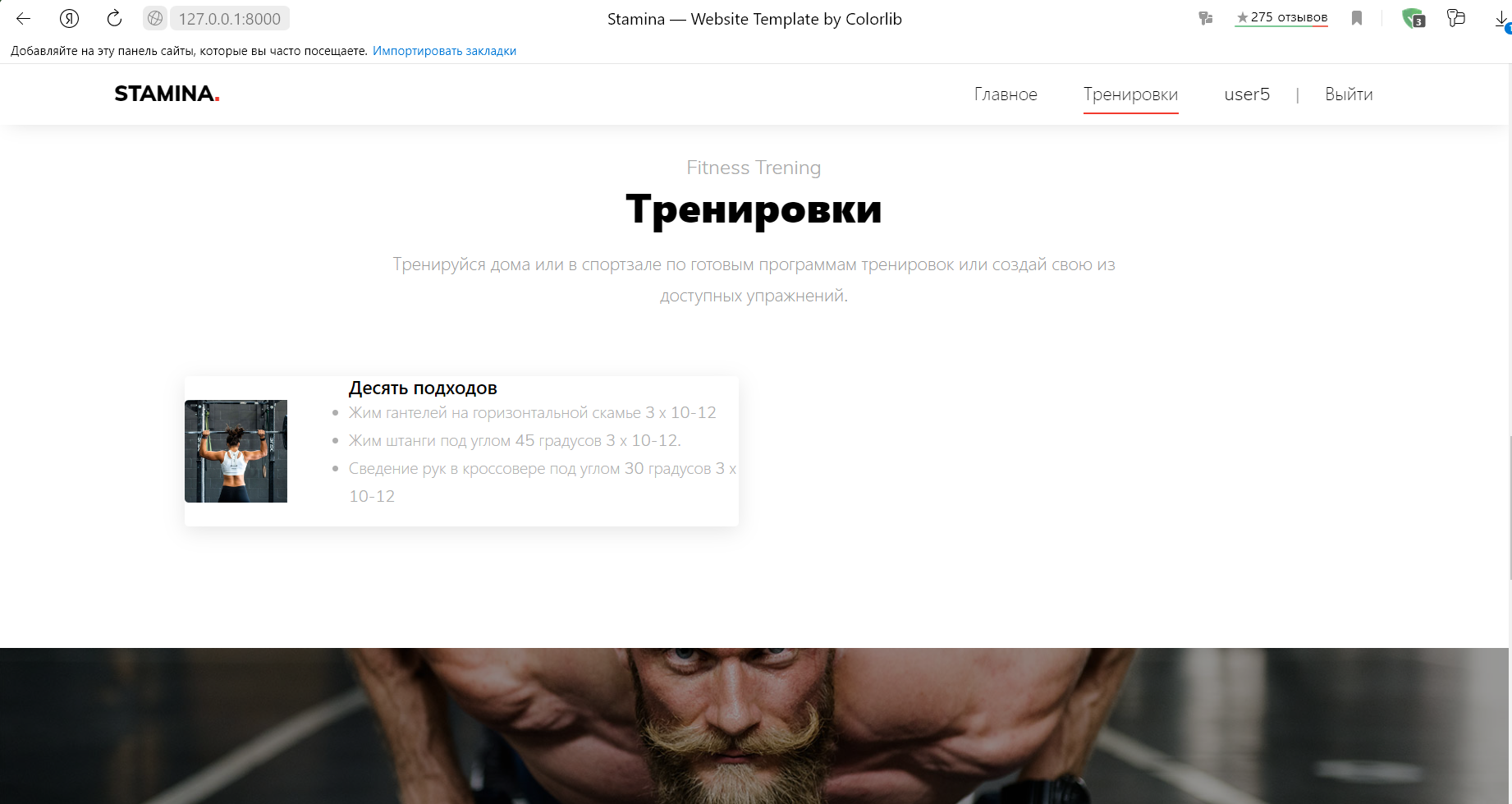


Рисунок 10.6 – результат работы программы.

**Вывод**: были рассмотрены возможности веб-фреймворка Django для построения веб-приложений на языке Python, изучено понятие ORM и его реализации в данном фрейм-ворке.

|  |  |
| --- | --- |
| Выполнили |  |
| Проверил | Елкин Н.С. |