Областной конкурс

работ исследовательского характера (конференция) учащихся

по учебному предмету

«ИНФОРМАТИКА»

«Математик математика видит издалека»

Автор работы:

Иванов Владислав Владимирович

учащийся 8 «Г» класса

ГУО «Средняяшкола №2

г. Мстиславля»

Руководитель работы:

Виктосенко Ирина Константиновна,

учитель математики и информатики

ГУО «Средняя школа №2

г.Мстиславля»

Мстиславль, 2021

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Среди одноклассников бытует мнение, что неординарные личности уже не существуют. Что победители олимпиад – это «зомбированные» дети, которые много часов в неделю отводят на занятия по тому или иному предмету, на которых учителя «натаскивают» их по определенным темам. У меня возникла идея создать сайт, на котором каждый мог бы разместить свои авторские задания по математике по следующим признакам:

признак 1. Условие задачиучастник придумал сам;

признак 2. Решение задачи, в некоторой степени, не поддается всем известным методам решения.

Как вывод, решить такие задачи сможет только такая же неординарная личность. Как говорится, «рыбак рыбака видит издалека».

Я понимал, что осуществить данную идею я смогу лишь в том случае, если создам такой сайт сам. Поэтому из истинного математика я превратился еще и в истинного информатика.

Объект исследования: создание сайтов.

Предмет исследования: использование сайта с целью выявления математически одаренных учащихся.

Гипотеза: созданный сайт позволит создать сообщество одаренных учащихся, в котором каждый сможет развить свои способности, так как все гениальны, но не все это замечают и развивают.

Цель работы: создание условий для проявления и реализации математических способностей учащихся посредством использования сайта MathPractice.

Задачи:

изучить литературу по данному вопросу;

создать и протестировать сайт в интернете;

получить обратную связь.

Актуальность заключается в том, что нынешнее поколение, постоянно находящееся в виртуальной реальности, не может найти своих единомышленников для самореализации в современном обществе. Я предлагаю нечто новое, нечто несоизмеримое с тем, что на сегодняшний

момент представлено в сети: найти себе подобных по мышлению, по анализу и синтезу, по выводам из сложившейся ситуации.

ГЛАВА 1 ИСТОРИЯ РАЗРАБОТКИ САЙТА MATHPRACTICE

1.1 Возникновение идеи

Возникновение идеи создания сайта для любознательных математиков возникла в конце весны 2021-го года. Тогда мой учитель математики предложил мне интересную идею для исследовательской работы – сделать сайт, в котором люди смогут оставлять свои математические задачи. Мне сразу понравилась эта идея. Я, мои друзья и другие единомышленники смогли придумать много идей для этого сайта.

1.2 Почему я согласился

К тому времени я занимался программированием, а в частности, созданием сайтов уже почти полгода. Я уже понимал принципы работы сайта и уже умел создавать небольшие простые сайты, но без BackEnd'а – программно-аппаратной части проекта. И мне хотелось взяться за создание какого-то серьезного проекта, которым являлся MathPractice.

1.3 Технологии, которые я использовал при создании сайта

Еще с 11 лет меня привлек язык программирования Python [4] Он мне очень нравился своей простотой и доступностью. И у этого языка есть прекрасный фреймворк(программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного кода) – Django [5]. Django является одним из лучших в мире фреймворков для написания сайтов. Поэтому выбор был очевиден. Я выбрал именно его. Для написания FrontEnd'a(клиентская сторона пользовательского интерфейса к программно-аппаратной части сервиса) я использовал языки гипертекстовой разметки HTML и CSS. Весь код я писал в редакторе кода SublimeText 3 [6].

1.4 Первый этап разработки: собираем внешний вид сайта

Приступая к работе, я не понимал, с чего лучше всего начать. И я решил взять листок и карандаш и продумать архитектуру сайта. Дальше я взялся за логотип, который делал с помощью программы Figma [7]. Далее предстояло выбрать основной шрифт и цветовую схему для сайта. Шрифт я выбрал на сайте CoogleFonts[8]. И этим шрифтом стал являться шрифт

Roboto. А цветовую схему я выбрал с сайта ColorHunt[9]. Подготовка к основной разработке была выполнена.

1.5 Второй этап разработки: изучение Django

Сейчас в интернете можно найти огромное множество ресурсов для изучения Django. Но я хотел выбрать такой, чтобы параллельно с изучением я мог разрабатывать сайт. И я выбрал курс вводный курс по Django–[10]. Благодаря этому курсу я начал создавать сайт. На этом этапе у меня не возникло особых сложностей, но я очень заинтересовался данным фреймворком. Он мне понравился не меньше, чем язык, на котором он написан – Python.

1.6 Третий этап разработки: непонимание того, что делать дальше

Во время второго этапа у меня не возникло проблем, так как в курсе всё было четко описано: что, зачем и как делать. Но дальше я отправился в открытое плавание, в котором было огромное множество путей. Это и регистрация/вход, и типы/категории задач, и комментарии по решению к задачам.

1.7 Четвертый этап разработки: неудачная регистрация

После третьего этапа я выбрал путь по созданию регистрации на сайте. Учитывая то, что я плохо знал Django, я воспользовался интернет-ресурсом Django в примерах «Регистрация пользователей» [11]. И сначала, все шло хорошо. Регистрация и вход начинали работать, программа не показывала ошибок. Но потом я понял, что в данном ресурсе используется вторая версия Django(новой версией является Django 3), из-за чего позже могли возникнуть определенные проблемы. Основной из них стал бы перевод Django 2 в Django 3. И мне, как начинающему программисту в этом направлении, было бы очень тяжело это делать. Поэтому я удалил папку с регистрацией из проекта.

1.8 Пятый этап разработки: марафон создания сайта

Уже наступил сентябрь. И для меня это означало лишь одно:довести сайт до той стадии, на которой его можно выкладывать,и это нужно сделать

за 30 дней. И я удвоил свои силы на разработку сайта. Мне нужно было лучше освоить Django. Поэтому каждый день я читал книгу «Django 3. Практика создания веб-сайов на Python», автором которой является Владимир Дронов, который написал огромное множество книг по программированию, таких как «Python 3 и PyQt 5. Разработка приложений», «Laravel 8. Быстрая разработка веб-сайтов на PHP». Также каждый день я пытался добавлять что-то новое на сайт.

1.9 Шестой этап разработки: категории задач

Вначале я решил добавить категории для задач на сайт. Я думал, что сделаю это за пару дней, так как это не выглядело сложным. Но это оказалось совсем не так. У меня возникли сложности из-за того, что я просто писал все сам, без какой-либо помощи. А сам я, повторюсь, знал Django не самым лучшим образом, хотя намного лучше, чем в самом начале. И за всем этим я просидел около недели. Но теперь результат полностью удовлетворял меня.

1.10 Седьмой этап разработки: регистрация, авторизация и идентификация

После мне предстояло сделать то, с чем у меня возникали большие трудности. Мне нужно было сделать для начала простую регистрацию, поэтому я выбрал ролик на youtube-канале «Регистрация и авторизация по Email в Django»[12]. У меня получилась очень простая, но регистрация. Конечно, с ней тоже возникали свои трудности, из-за которых процесс создания растянулся на полторы недели.

1.11 Восьмой этап разработки: комментарии

Все, что мне оставалось сделать для того, чтобы сайт уже можно было выкладывать в интернет, это комментарии. И процесс не занял у меня много времени. Я буквально за день сделал все правильно и красиво, пользуясь роликом на youtube-канале «Django, часть 9: Создаем модуль с комментариями. Связываем две модели»[13].

1.12 Девятый этап: выгрузка сайта в интернет

Я примерно понимал, как нужно выгружать сайт в интернет, но я не знал, как это нужно делать на Django, так как там есть свои тонкости. Я пользовался ресурсом djangogirls «Публикация в интернете»[14]. Этот процесс занял у меня целую неделю из-за неопытности в выгрузке сайтов. Сначала сайт не видел и не читал статические файлы(картинки, CSS). Проблема заключалась в том, что при выгрузке я не указал путь, в котором должны лежать подобные файлы. Это указывается в переменной STATIC_ROOT файла setting.py проекта. После указания пути проект не обновлялся на хостинге. Кстати, хостингом является«Pythonanywhere» [15]. Потом я создал новый аккаунт и снова загрузил туда этот проект, после чего все заработало. И вот, 2 октября любой пользователь уже мог посетить сайт MathPractice по соответствующей ссылке [16].

1.13 Структура сайта

Сайт содержит четыре вкладки:

1. Главная

   1.1 Что такое MathPractice? MathPractice - сайт, в котором ты можешь поделиться с людьми гениальной задачей, которая резко пришла тебе в голову, но ты не можешь ее решить. Здесь тебе обязательно помогут такие любознательные люди, как и ты

2. Задачи

   2.1 Категории задач

   2.2.1 Геометрия

   2.3.2 Дроби

   2.4.3 Множества

   2.5.4 На логику

   2.6.5 Уравнения

3. Регистрация

4. Вход

Вы можете посмотреть абсолютно все файлы и папки проекта по соответствующей ссылке [17].

ГЛАВА 2 СОЗДАНИЕ САЙТА И ЕГО РАБОТА

А теперь давайте глубже рассмотрим работу сайта.

Как и любой Django-проект, MathPractice состоит из приложений. Django-приложение – web-приложение, предоставляющее определенный функционал. Мой проект состоит из 4 приложений: config, main, tasksиaccount.

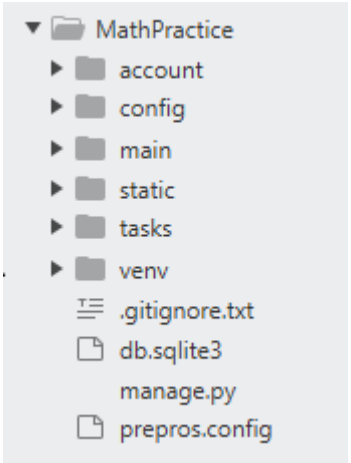

Рисунок 1 – Структура проекта

Наизображениивыможетеувидетьнеупомянутыепапкиstatic, venv, атакжефайлы .gitignore.txt, db.sqlite3, manage.py, prepros.config. Давайтеразбираться.

Папка staticне является приложением, а содержит в себе статические файлы: изображения, файлы css, файлы javascript. Папка venv является папкой виртуального окруженияpython. Виртуальное окружение – изолированное окружение среды, которое позволяет использовать определенные версии приложений. Файл .gitignore.txt является файлом для GitHub и указывает, какие файлы нужно игнорировать. Файл db.sqlite3 хранит в себе таблицы базы данных сайта. Файл manage.pyявляется файлом, который помогает разработчику выполнять и отдавать разные команды сайту(запустить локальный сервер, создать администратора сайта). И наконец, файл prepros.config является файлом, который помогает компилировать SCSS-код в CSS-код. Зачем это нужно? Браузер еще не научился читать код, написанный на SCSS, в отличие от кода CSS, поэтому приходится переводить код SCSSв код CSS. Почему сразу не писать на CSS? Потому что SCSSпредоставляет более простой, понятный и наглядный синтаксис кода, нежели CSS.

Разберем каждое приложение отдельно.

Первым созданным приложением является приложение config, которое создается по умолчанию после создания проекта(это приложение создается под именем проекта по умолчанию, но его можно изменить). Оно отвечает за глобальные настройки проекта. Давайте рассмотрим структуру данного приложения.

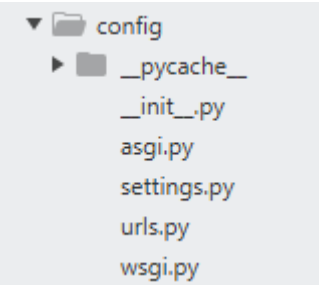

Рисунок 2 – Структура приложенияconfig

Не будем рассматривать файлы __init__.py, asgi.py, wsgi.py, так как они не использовались при разработке. Рассмотрим лишь два файлы: settings.py и urls.py.

По названию файла settings.py можно догадаться, что он отвечает за основные настройки проекта.

```
from pathlib import Path
import os

BASE_DIR = Path(__file__).resolve().parent.parent


SECRET_KEY = '...'

DEBUG = True

ALLOWED_HOSTS = ['mathpractice.pythonanywhere.com', '127.0.0.1']


INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    'main',
    'tasks',
    'account'
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

Рисунок 3 – Файл settings.py приложения config

Также в разработке использовался файл urls.py, который отвечает за обработку url-адресов на сайте.

```python
from django.contrib import admin
from django.urls import path, include

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

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('main.urls')),
    path('tasks/', include('tasks.urls')),
    path('account/', include('account.urls')),
] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

Рисунок 4 – Файл urls.py приложения config

В любом django-приложении должно быть хотя бы одно приложение. Первым мною созданным приложением в данном проекте стало приложение main. Рассмотрим его структуру.

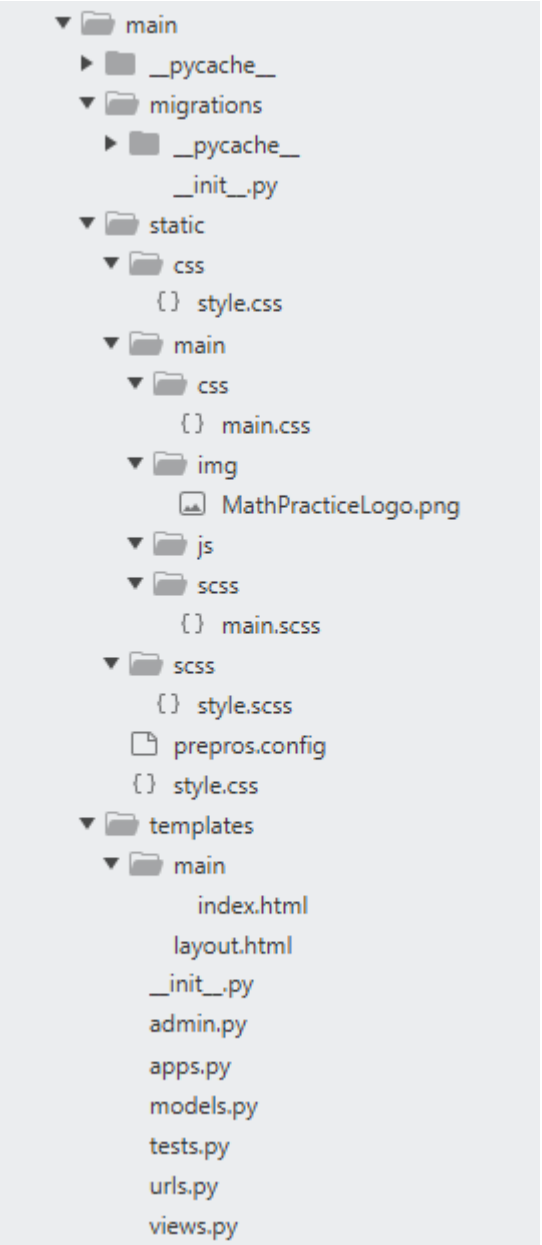

Рисунок 5 – Структура приложения main

Папка static. Уже упоминалась. Хранит в себе статические файлы(изображения, код CSS, код JavaScript).

Папка templates. Хранит в себе HTML-шаблоны страницы, то есть то, что видит пользователь, открывая страницу. Эта папка связана с папкой static. Static делает шаблоны красивыми.

И оставшиеся файлы. В разработке данного приложения использовались лишь urls.py и views.py. Их мы и рассмотрим.

Файл urls.py. Обрабатывает лишь один url-адрес–отображение главной страницы пользователю.

```python
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='home')
]
```

Рисунок 6 – Файл urls.py приложения main

Вы можете увидеть использование функции index файла views.py в этом приложении. Как это работает? Когда пользователь переходит по определенному url-адресу, в нашем случае на главную страницу, вызывается функция indexиз файла views.py. Давайте рассмотрим эту функию.

```python
from django.shortcuts import render

def index(request):
    data = {
        'title': 'Главная страница'
    }
    return render(request, 'main/index.html', data)
```

Рисунок 7 – Файл views.py приложения main

Функция index возвращает шаблон index.html из templates/main/(templates по умолчанию не указывается). И пользователь видит этот шаблон.

Приложение tasks является самым крупным в проекте. Оно отвечает за задачи, категории задач и комментарии. Рассмотрим его структуру.

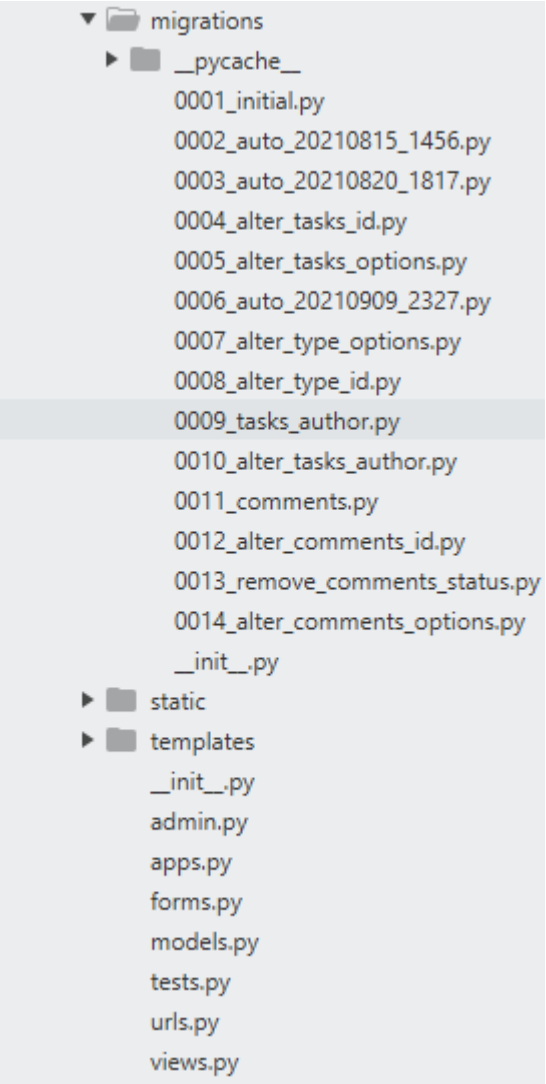

Рисунок 8 – Структура приложения tasks

Папкиstaticitemplatesпохожи во всех приложениях, поэтому мы их рассматривать второй раз не будем. Здесь у нас появляется новая папка, которая использовалась в разработке. Это папка migrations. Она хранит в себе миграции – python-файлы, которые хранят в себе модели базы данных. Подробнее о моделях мы поговорим, когда дойдем до файла models.py.

Рассмотрим файлы admin.py, forms.py, models.py, urls.pyи views.py. Начнем с файла urls.py.

```
from django.urls import path, include
from . import views

urlpatterns = [
    path('', views.tasks_home, name='tasks_home'),
    path('create', views.TaskCreateView.as_view(), name='create'),
    path('<int:type_id>/', views.task_type, name='type'),
    path('<int:type_id>/<int:pk>/', views.TasksDetailView.as_view(), name='task-detail'),
    path('<str:type_id>/<int:pk>/update', views.TasksUpdateView.as_view(), name='task-update'),
    path('<str:type_id>/<int:pk>/delete', views.TasksDeleteView.as_view(), name='task-delete'),
]
```

Рисунок 9 – Структура файла urls.py приложения tasks

Здесь вы можете увидеть непонятный url-адрес(<int:type_id>/ и т.п.). Этот параметр может изменяться в зависимости от задачи, по которой мы переходим, категории и других параметров. И чтобы показать, что этот параметр будет изменяться, используется такая запись. Также вы можете увидеть здесьзапись *имя_класса.as_view()*. Благодаря такой записи мы показываем, что используем не функцию, а класс.

В файле views.py хранятся классы для создания задачи, детального отображения задачи, изменения задачи и удаления задачи, а также функции отображения страницы tasks/ и tasks/*id_категории.*

Теперь давайте рассмотрим файл forms.py. Файл используется, согласно названию, для создания форм.

```python
from .models import Tasks, Type, Comments
from django.forms import ModelForm, TextInput, Textarea, ModelChoiceField


class TasksForm(ModelForm):
    class Meta:
        model = Tasks
        fields = ['title', 'task', 'type']

        type = ModelChoiceField(queryset=Type.objects.all(), to_field_name='type')

        widgets = {
            'title': TextInput(attrs={
                    'class': 'form-input',
                    'tabindex': 1,
                    'placeholder': 'Название',
            }),

            'task': Textarea(attrs={
                    'class': 'form-input',
                    'tabindex': 2,
                    'placeholder': 'Текст задачи',
            }),
        }

class CommentForm(ModelForm):
    class Meta:
        model = Comments
        fields = ('text', )

        widgets = {
            'text': Textarea(attrs={
                    'class': 'form-input comment',
                    'placeholder': 'Текст комментария'
                })
        }
```

Рисунок 10 – Файл forms.py приложения tasks

Здесь есть два класса, которые создают две формы. Форма создания задачи и форма создания комментария.

Теперь разберем файл, в котором хранятся модели базы данных models.py.

```python
from django.db import models
from django.contrib.auth.models import User

class Tasks(models.Model):
    id = models.AutoField(primary_key=True)
    author = models.ForeignKey(User, on_delete=models.CASCADE, null=True, blank=True)
    title = models.CharField('Title', max_length=50)
    task = models.TextField('Task')
    date = models.DateTimeField(auto_now_add=True, null=True, blank=True)
    updated_date = models.DateTimeField(auto_now=True, null=True, blank=True)
    type = models.ForeignKey('Type', on_delete=models.PROTECT, null=True)

    def __str__(self):
        return self.title

    def get_absolute_url(self):
        return f'/tasks/{self.type_id}'

    class Meta:
        verbose_name = 'Задача'
        verbose_name_plural = 'Задачи'
        ordering = ('-date',)

class Type(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=50, db_index=True)

    def __str__(self):
        return self.name

    class Meta:
        verbose_name = 'Категория'
        verbose_name_plural = 'Категории'
        ordering = ('name',)

class Comments(models.Model):
    id = models.AutoField(primary_key=True)
    task = models.ForeignKey(Tasks, on_delete=models.CASCADE, blank=True, null=True, related_name='comments_tasks')
    author = models.ForeignKey(User, on_delete=models.CASCADE, blank=True, null=True)
    created_date = models.DateTimeField(auto_now=True)
    text = models.TextField()

    class Meta:
        verbose_name = 'Комментарий'
        verbose_name_plural = 'Комментарии'
        ordering = ('-created_date',)
```

Рисунок 11 – Файл models.py приложения tasks

Здесь вы можете увидеть три модели базы данных. Для задач, категорий и комментариев. Для каждой модели свои поля. Некоторые поля связываются с другими моделями, такие как поле typeмодели Tasks, поле taskмодели Comments. Они связываются с помощью атрибута ForeignKey. В классе Metaуказываются «человеческие» названия моделей.

И последний файл, который мы рассмотрим в приложении tasks– файл admin.py. Он предназначен для регистрации моделей в базу данных.

```python
from django.contrib import admin
from .models import Tasks, Type, Comments

class TasksAdmin(admin.ModelAdmin):
    list_display = ('title', 'task', 'date', 'updated_date')
    list_display_links = ('title', 'task')
    search_fields = ('title', 'task')

class TypeAdmin(admin.ModelAdmin):
    list_display = ('name',)
    list_display_links = ('name',)
    search_fields = ('name',)

class CommentsAdmin(admin.ModelAdmin):
    list_display = ('author', 'created_date', 'text')
    list_display_links = ('author', 'text')
    search_fields = ('author', 'text')

admin.site.register(Tasks, TasksAdmin)
admin.site.register(Type, TypeAdmin)
admin.site.register(Comments, CommentsAdmin)
```

Рисунок 12 – Файл admin.py приложения tasks

Для чего мы регистрируем модели? Для того чтобы они отображались в панели администратора сайта и их можно было редактировать администратору.

Приложение должно называться так, чтобы, прочитав название, мы поняли, зачем оно создано. По названию приложения accountможно понять, что оно создано для регистрации, авторизации и идентификации пользователей. Давай рассмотрим его структуру.

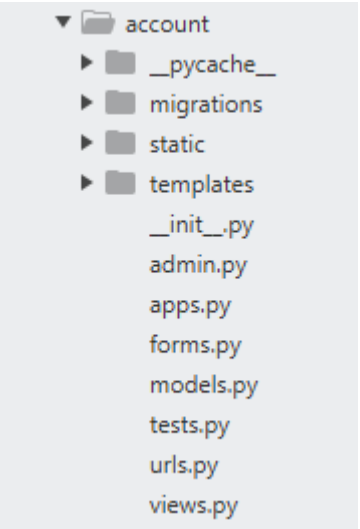

Рисунок 12 – Структура приложения account

В этом приложении мы, как и в предыдущем, не будем рассматривать папки templatesи static. Также мы не будем рассматривать папку migrations,

так как здесь не были использованы миграции. Мы рассмотрим лишь три файла, которые были использованы в разработке. Это файлы urls.py, views.py и forms.py.

Рассмотрим код файла urls.py.

```
from django.urls import path, include
from .views import register
from django.contrib.auth import views as auth_views


urlpatterns = [
    path('register', register, name='register'),
    path('login/', auth_views.LoginView.as_view(template_name='account/login.html'), name='login'),
    path('logout/', auth_views.LogoutView.as_view(template_name='account/logout.html'), name='logout'),
]
```

Рисунок 13 – Файл urls.py приложения account

Здесь мы отслеживаем три url-адреса. Адрес регистрации, входа и выхода. Вход и выход в Django уже заранее реализован, поэтому мы ничего нового не изобретали, а использовали то, что есть. А в Djangoесть два класса: LoginViewи LogoutView.

Регистрацию нужно было писать самому. Для этого используется функция registerиз файла views.py. Давайте рассмотрим ее.

```python
from django.shortcuts import render, redirect
from django.http import HttpResponseRedirect
from .forms import UserRegisterForm
from django.contrib.auth.models import User
from django.contrib import messages
from django.contrib.auth import authenticate, login


def register(request):
    form = None
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            ins = form.save()
            username = form.cleaned_data['username']
            password = form.cleaned_data['password1']
            user = authenticate(username=username, password=password)
            login(request, user)
            form.save_m2m()
            messages.success(request, 'Вы успешно зарегестрировались!')
            return redirect('/')

    else:
        form = UserRegisterForm()

    data = {
        'form': form,
    }
    return render(request, 'account/register.html', data)
```

Рисунок 14 – Файл views.py приложения account

Здесь, в этой функции, мы получаем на вход имя пользователя и пароль, шифруем пароль и сохраняем пользователя.

Давайте рассмотрим, как создавалась форма, с помощью которой, пользователь передает свое имя и пароль. Это описывается в файле forms.py.

```python
from django.contrib.auth.forms import UserCreationForm
from django import forms
from django.contrib.auth.models import User

class UserRegisterForm(UserCreationForm):
    field_order = ['username', 'password1', 'password2']

    class Meta:
        model = User
        fields = ('username', 'password1', 'password2')
```

Рисунок 15 – Файл forms.py приложения account

Вот и все. Вот так устроен сайт MathPractice.

ЗАКЛЮЧЕНИЕ

MathPractice – сайт, который имеет множество идей для доработок, добавлений и обновлений. На сайт будет добавлена намного более продуманная регистрация и вход пользователей. Владельцу задачи будет добавлена возможность выбрать комментарий, который является наилучшим решением задачи. Пользователи смогут оценивать задачу по сложности. Будет добавлен личный кабинет для каждого пользователя, в котором можно посмотреть все созданные и решенные им задачи. Пользователи смогут подписываться друг на друга. Будет добавлен телеграм-бот, который будет сообщать о новых задачах, о решениях вашей задачи и о многом другом. Запланировано создание мобильного приложения MathPractice.

На данном этапе сайт был протестирован среди учащихся нашего района. Получены следующие аналитические сведения: 18 посетителей сайта, 5 зарегистрированных пользователей, 8 размещенных задач от 3-х пользователей, 5 представленных решения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Уитни,Девид. Программирование для детей. Учимся создавать сайты, приложения и игры. HTML, CSS и JavaScript. – СПб.: Питер, 2021. – 208 с.: ил. – (Серия «Вы и ваш ребенок»).
2. [Электронный ресурс]. – 2021. – Режим доступа:. https://alteropt.github.io/hotdogs/
3. [Электронный ресурс]. – 2021. – Режим доступа:. https://alteropt.github.io/activebox/
4. [Электронный ресурс]. – 2021. – Режим доступа:. https://www.python.org/
5. [Электронный ресурс]. – 2021. – Режим доступа:. https://docs.djangoproject.com/en/3.2/
6. [Электронный ресурс]. – 2021. – Режим доступа:. https://www.sublimetext.com/
7. [Электронный ресурс]. – 2021. – Режим доступа:. https://www.figma.com/)
8. [Электронный ресурс]. – 2021. – Режим доступа:. https://fonts.google.com/
9. [Электронный ресурс]. – 2021. – Режим доступа:. https://colorhunt.co/palette/54436b50cb9371efa3acffad
10. [Электронный ресурс]. – 2021. – Режим доступа:. https://www.youtube.com/watch?v=L-FyeHQwo4U&list=PLDyJYA6aTY1nZ9fSGcsK4wqeu-xaJksQQ
11. [Электронный ресурс]. – 2021. – Режим доступа:. https://pocoz.gitbooks.io/django-v-primerah/content/glava-4-sozdanie-social-website/registratsiya-polzovatelei-i-profili-polzovatelei/registratsiya-polzovatelei.html.

12. [Электронный ресурс]. – 2021. – Режим доступа:. https://www.youtube.com/watch?v=YJpcAsNiicg.
13. [Электронный ресурс]. – 2021. – Режим доступа:. https://www.youtube.com/watch?v=fwoPp4v_flE
14. [Электронный ресурс]. – 2021. – Режим доступа:. https://tutorial.djangogirls.org/ru/deploy/.
15. [Электронный ресурс]. – 2021. – Режим доступа:. https://www.pythonanywhere.com/.
16. [Электронный ресурс]. – 2021. – Режим доступа:. https://mathpractice.pythonanywhere.com
17. [Электронный ресурс]. – 2021. – Режим доступа:. https://github.com/alteropt/mathpractice.
18.

ПРИЛОЖЕНИЕ А

ФРАГМЕНТЫ САЙТА

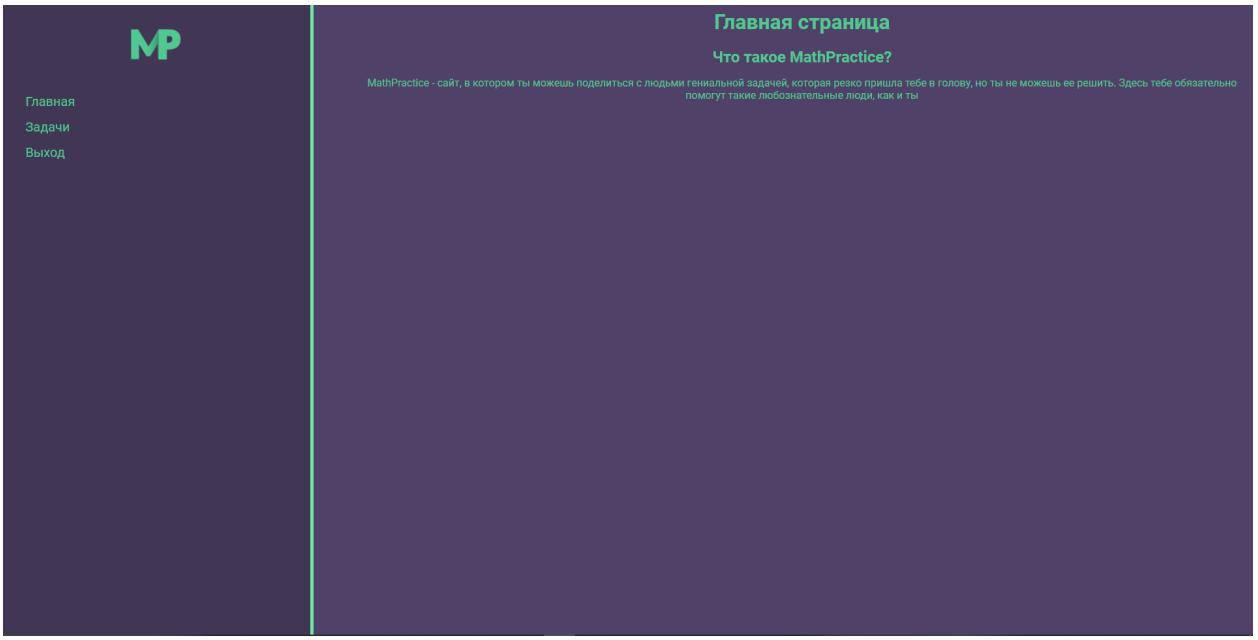

Рисунок А.1 – Главная страница

Рисунок А.2 – Категория задач

Рисунок А.3 – Задачи из категории

Рисунок А.4 – Отсутствие задач в какой-либо категории

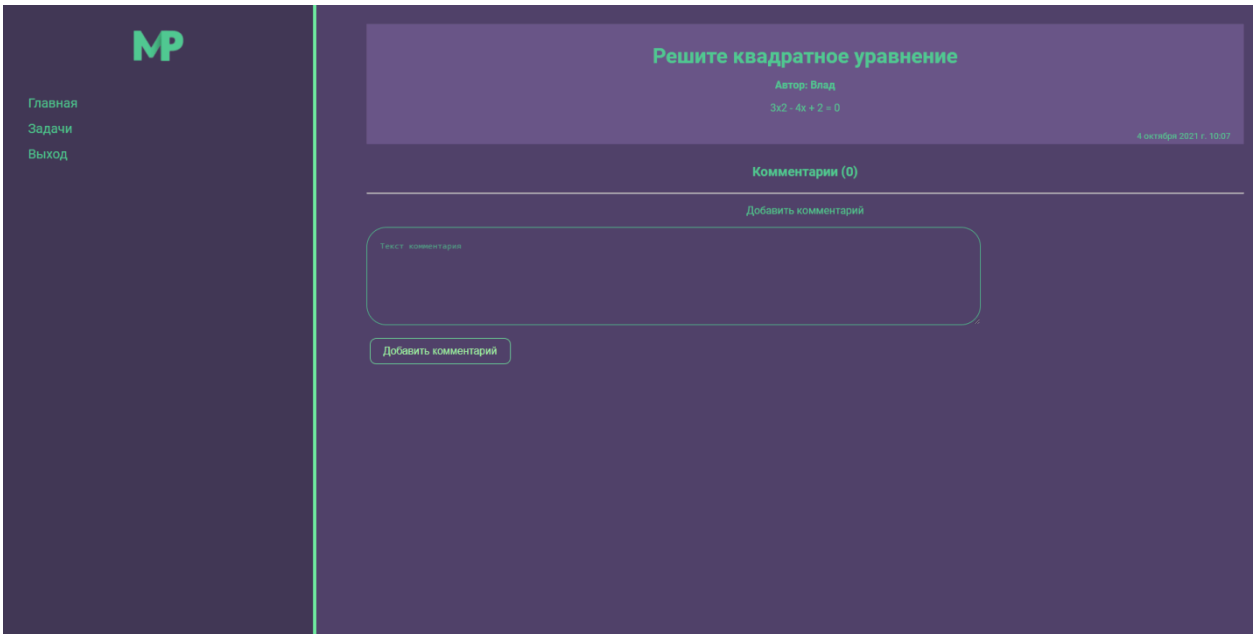

Рисунок А.5 – Детальный просмотр задачи для зарегистрированного пользователя

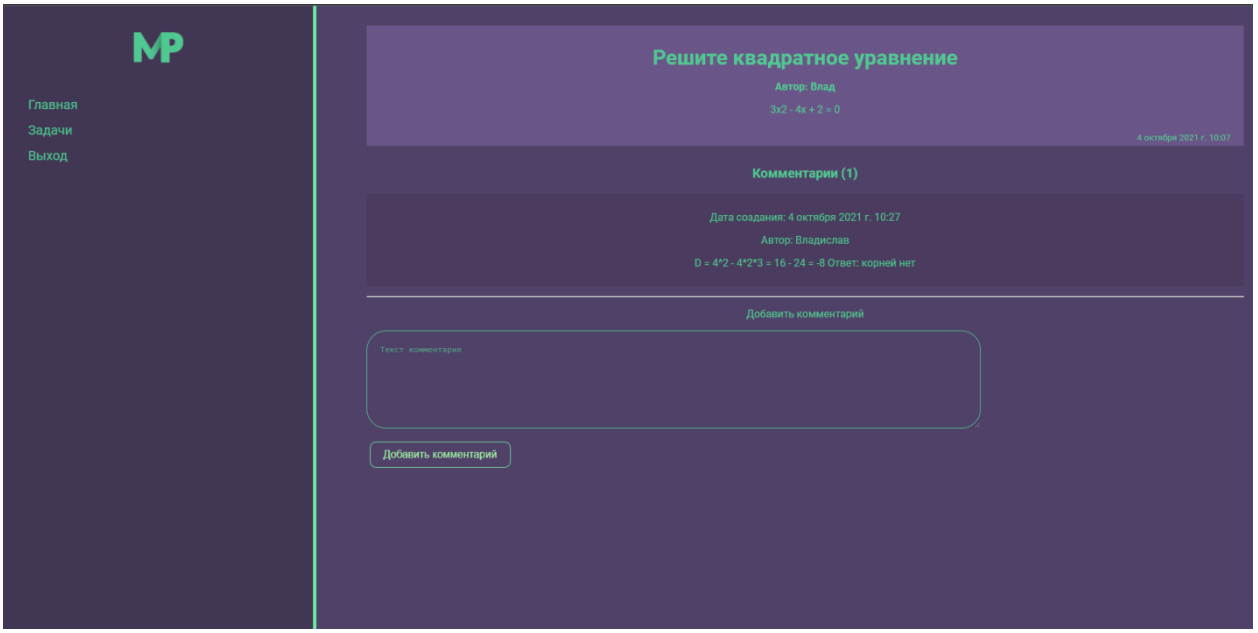

Рисунок А.6 – Отображение комментариев

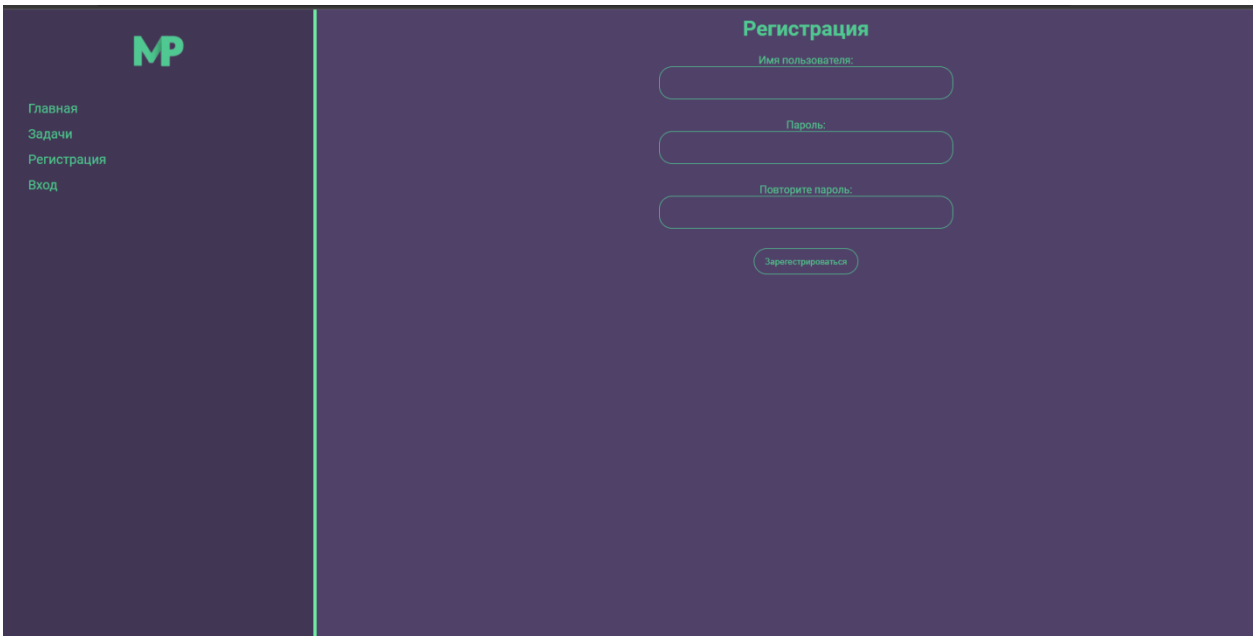

Рисунок А.7 – Регистрация

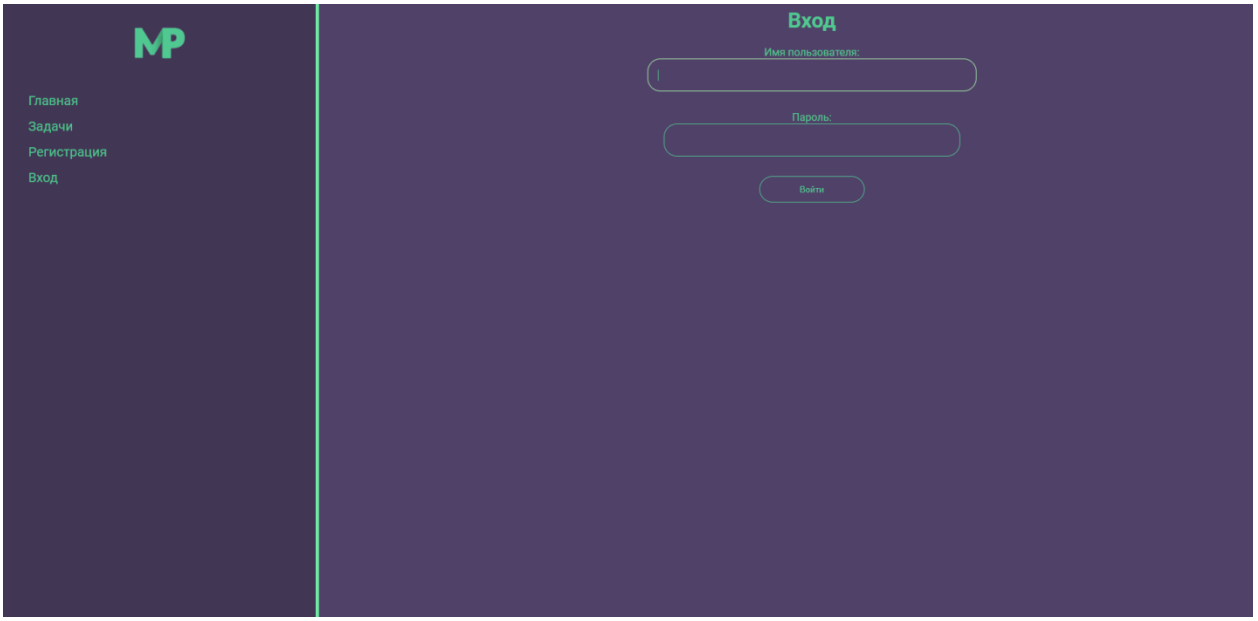

Рисунок А.8 – Вход

Рисунок А.9 – Детальный просмотр задачи для незарегистрированного пользователя

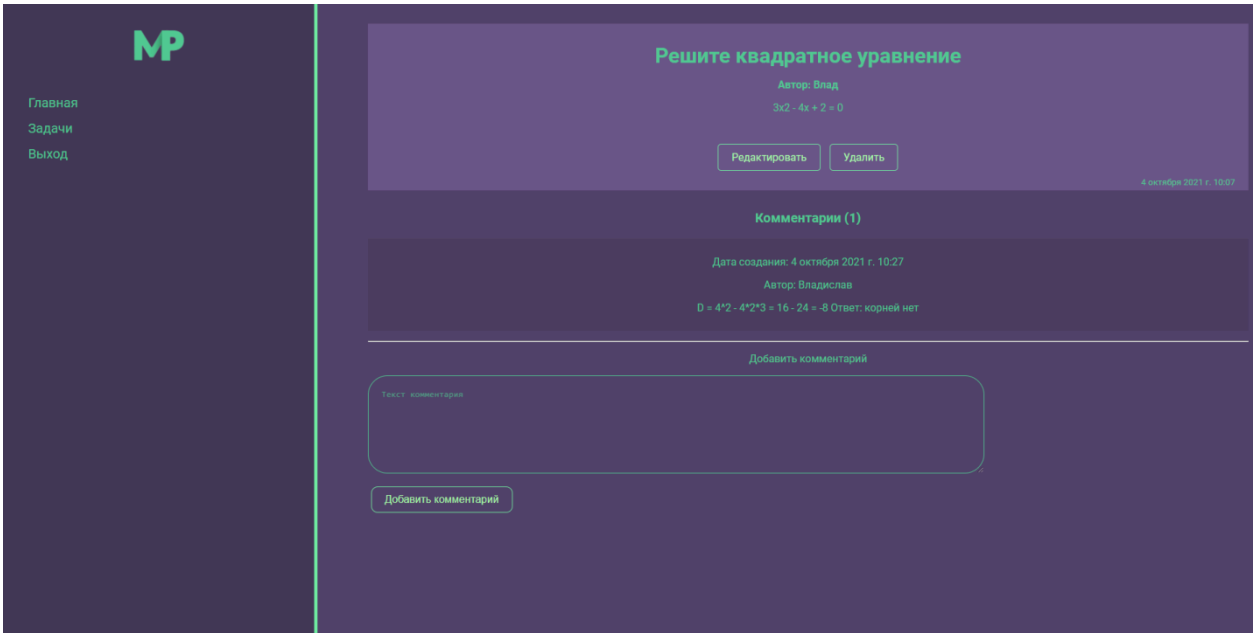

Рисунок А.10 – Детальный просмотр задачи для автора задачи