Министерство образования и науки Российской Федерации
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой
_____Г.И. Радченко
«___»_____2018 г.

Разработка веб-приложения для проведения олимпиад по программированию

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-090301.2018.534 ПЗ ВКР

Руководитель работы,
доцент каф. «Электронные
Вычислительные машины»
_____И.Л. Надточий
«___»_____2018 г.

Автор работы
студента группы КЭ-484
_____А.С. Шурунов
«___»_____2018 г.

Нормоконтролер, ст. преп. каф.
«Электронные вычислительные
машины»
_____В.В. Лурье
«___»_____2018 г.

Челябинск 2018

АННОТАЦИЯ

Шурунов А.С. Разработка веб-приложения для проведения олимпиад по программированию. – Челябинск: ЮУрГУ, КЭ-484, **48 с., 11 рис., 1 табл., библиогр. список – 14 наим., 1 прил.**

Выпускная квалификационная работа посвящена созданию веб-приложения для проведения олимпиад по программированию. Помимо самой реализации, в рамках работы также был проведен анализ аналогов, выдвинуты требования, спроектирована архитектура, приняты и обоснованы проектные решения.

Данное веб-приложение может быть использовано для проведения спортивных состязаний по программированию, онлайн-олимпиад по программированию или для автоматизации проверки работ студентов вводных курсов языков программирования, фреймворков и библиотек.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Олимпиада по программированию – соревнование, в котором принимают участие, обычно, студенты, в личном или командном зачете. Тур олимпиады происходит следующим образом: каждой команде или участнику выдается компьютер и некоторое количество задач. Для участников задача выглядит как некоторое описание проблемы и поясняющий пример. Участникам необходимо написать программу, решающую данную задачу и послать ответ, где он будет проверен на правильность путем прохождения заранее составленного множества тестов и на удовлетворение дополнительным условиям, таким как время выполнения и затраченная память. Побеждает команда или участник, решивший правильно наибольшее число задач. В качестве дополнительных показателей могут считаться количество неудачных попыток, которое должно быть минимальным, и общее затраченное время.

Можно заметить, что такое же программное обеспечение можно использовать для автоматизации проверки работ студентов вводных курсов языков программирования, фреймворков и библиотек: в обоих случаях часто встречаются задачи с консольным интерфейсом и детерминированными выходными данными. Примерами курсов, для которых выполненные задания смогут проверяться автоматически: вводный курс C++, введение в ROS, операционные системы семейства Unix, Linux и основы распределенной обработки и др.

1. ВВЕДЕНИЕ В ПРЕДМЕТНУЮ ОБЛАСТЬ

1.1 Терминология

Олимпиада по программированию – интеллектуальное соревнование по решению различных задача на ЭВМ, в ходе которых участники олимпиады пишут программы для решения олимпиадных задач;

Администратор олимпиады – лицо, ответственное за проведение олимпиады;

Участник олимпиады – лицо или группа лиц (команда), принимающее участие в олимпиаде и решающее олимпиадные задачи;

Олимпиадная задача – задача с описанием проблемы, ограничениями и возможностью проверки правильности решения, т. е. заданным множеством входных и выходных значений;

Ограничения олимпиадной задачи – ограничения на время выполнения задачи, используемую память или используемые языки;

Компилятор – техническое средство, выполняющее компиляцию – трансляцию программы, составленную на некотором языке, в аналогичную программу низкоуровневого языка;

Интерпретатор – техническое средство, выполняющее интерпретацию – построчный анализ и выполнение исходного кода.

1.2 ACM/ICPC

Для изучения предметной области рассмотрим самую популярную олимпиаду по программированию – Международную студенческую олимпиаду по программированию ACM/ICPC (или просто ICPC).

В каждом туре участвуют команды из трех человек, подавшие заявки, т. е. добавление пользователей происходит администраторами. В назначенное время команды собираются на площадке для проведения олимпиады, где каждой

команде выдается подготовленный организаторами компьютер. В начале командам выдается тестовая задача, которое используется для выявления неполадок. Спустя некоторое время начинается олимпиада: участникам выдается от 8 до 12 задач. С точки зрения участников, каждая задача представляет собой название и текстовое описание. Исходя из описания, команда пишет программу и посылает ее, после чего задача либо считается решенной, либо попытка считается неудачной. Во втором случае на участников налагается штраф и им приходит сообщение об ошибке с ее описанием: на каком тесте программа провалилась, на сколько миллисекунд она работала дольше или насколько килобайт памяти больше заняла [1].

2. ПОСТАНОВКА ЗАДАЧИ

2.1 Цель выпускной квалификационной работы

Целью выпускной квалификационной работы является разработка веб-приложения для проведения олимпиад по программированию.

2.2 Задачи выпускной квалификационной работы

Для достижения цели выпускной квалификационной работы были поставлены следующие задачи:

1. Провести сравнительный анализ аналогов;
2. Составить требования к разрабатываемому продукту;
3. Разработать архитектуру программного комплекса;
4. Реализовать программного обеспечения.

2.3 Конечный результат выпускной квалификационной работы

Конечным результатом выпускной квалификационной работы является веб-приложение для проведения олимпиад по программированию с возможностью дальнейшего расширения функциональных возможностей.

3 АНАЛИЗ АНАЛОГОВ

Аналогами для разрабатываемого программного продукта выступают платформенные приложения, веб-приложения и иные программы, которые включают в себя возможность проведения олимпиад по программированию.

Среди аналогов есть:

1. SPOJ [2];

2. TOPH [3];

3. PC^2 [4];

4. Kattis [5].

3.1 SPOJ

SPOJ – веб-сайт для проведения олимпиад по программированию, на котором провели уже более 4500 олимпиад.

Из плюсов:

1. Поддерживает более 45 языков программирования и компиляторов;

2. Предоставляет более 13000 задач для практики с описанием на разных языках.

Из минусов

1. Цена - 250$ за мероприятие;

2. Необходимость связываться с разработчиками для использования некоторого функционала (например, проведения закрытых олимпиад).

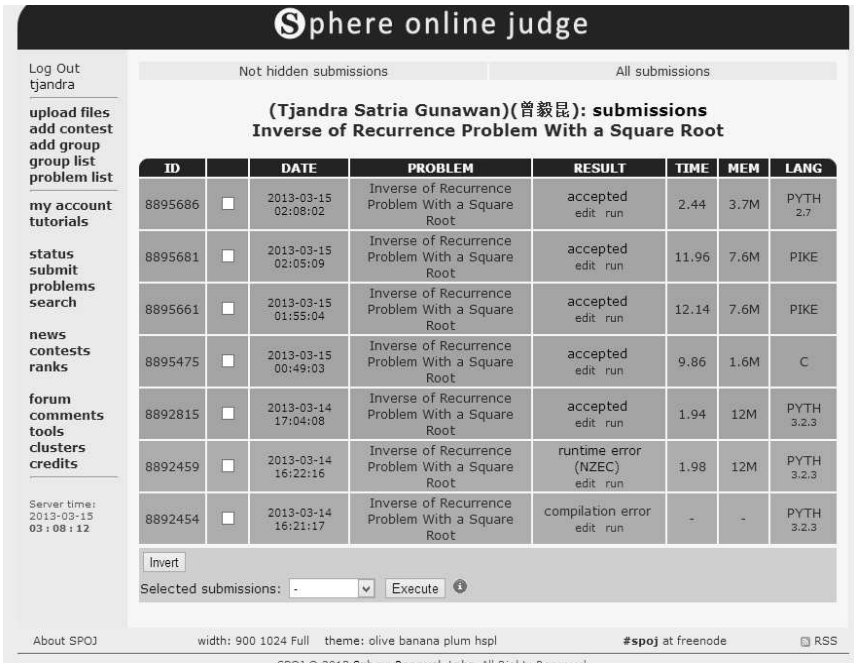

Рисунок 3.1.1 – интерфейс SPOJ

3.2 TOPH

Toph – веб-сайт для проведения олимпиад по программированию. Он менее популярен и главным минусом является неавтоматизированное создание задач.

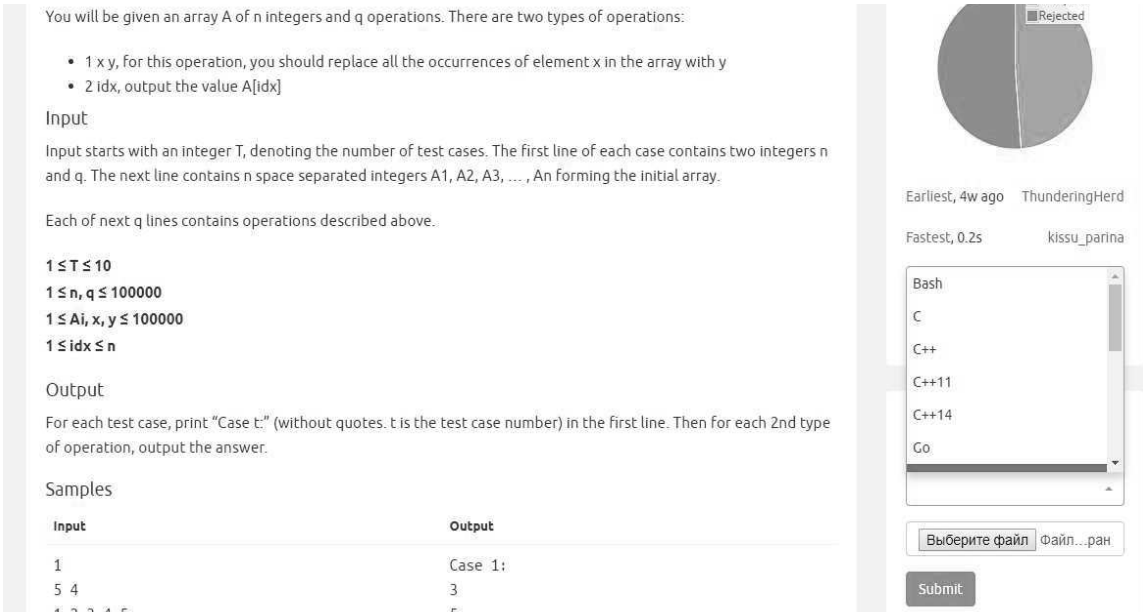

Рисунок 3.2.1 – интерфейс TOPH

3.3 PC^2

PC^2 – программа для проведения олимпиад по программированию, разработанная калифорнийским государственным университетом в Сакраменто и долгое время использовалось в ICPC (международной студенческой олимпиаде по программированию). Работая с этой программой, участники могут отсылать решения и получать оценку. У судей и проверяющих есть возможность перекомпилировать принятую программу, выполнить ее, просмотреть исходный код и результаты выполнения и отослать обратно участникам. Система также поддерживает режим «автоматического судейства», когда оценка выставляется программным обеспечением, а не человеком. Кроме того, система автоматически создает временные отметки и архивирует подтвержденные запуски и позволяет судьям перезапускать приложения. Это дает участникам возможность подавать запросы в спорных ситуациях. Главным минусов является необходимость самостоятельной подготовки к работе и настройки системы.

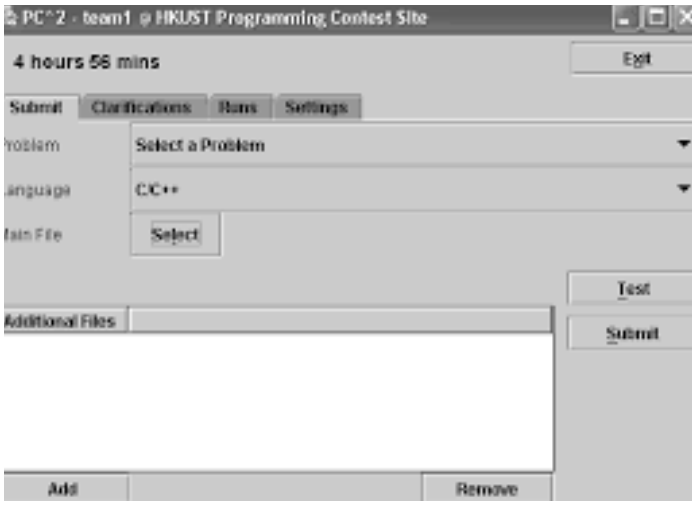

Рисунок 3.3.1 – интерфейс Kattis

3.4 Kattis

Kattis – веб-сайт для проведения олимпиад по программированию. Он стал использоваться ICPC после отказа от PC^2 в 2010 году. Веб-сайт поддерживает 17

языков и предлагает либо использовать веб-форму, либо клиент, который надо сначала скачать и настроить.

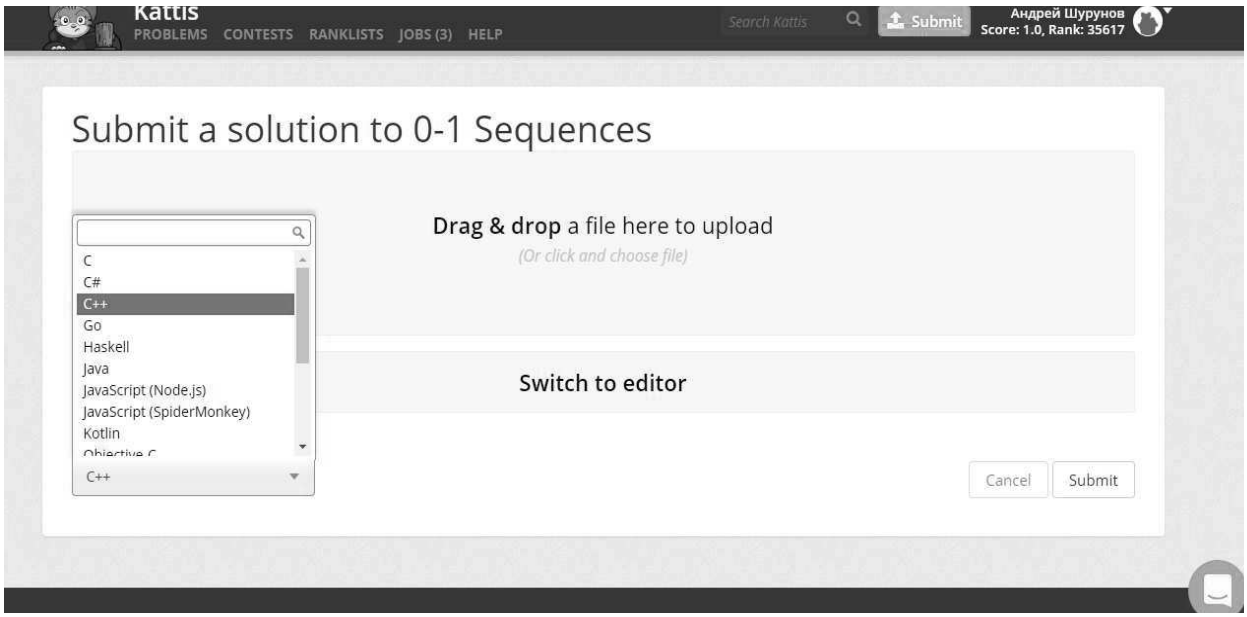

Рисунок 3.4.1 – интерфейс Kattis

3.5 Вывод

Таблица 3.1 – сравнительный анализ аналогов

Критерий сравнения	SPOJ	TOPH	PC^2	Kattis
Автоматизированный	+	-	+	+
Бесплатный	-	+	+	+
Подготовительные работы	-	-	+	±
Веб-сайт	+	+	-	+

Таким образом, суммируя все вышеперечисленное на данный момент отсутствует простое в использовании и автоматизированное бесплатное программное обеспечение.

4. РАЗРАБОТКА ТРЕБОВАНИЙ К ПРОГРАММНОМУ КОМПЛЕКСУ

4.1 Группы пользователей

Для определения необходимого функционала рассмотрим группы пользователей и функции, которыми они должны располагать. Основные группы пользователей проекта:

- Администраторы;
- Участники.

Для администраторов требуется следующие функции:

- Создание олимпиады;
- Создание и просмотр задач;
- Запуск программы;
- Связь с другими пользователями;
- Просмотр прогресса.

Для студентов требуется следующий функционал:

- Просмотр задач;
- Отправка решения;
- Связь с администраторами;
- Запуск программы;
- Просмотр прогресса.

4.2 Требования к функциональным характеристикам

Под созданием олимпиады понимается ввод названия, выбор даты начала, даты окончания и времени сессии (времени после начала участником прохождения олимпиады, после которого тот может отправлять решения). Также необходима возможно выбора открытости олимпиада.

Под созданием задачи понимается ввод описания (необходимое для участников), ввод тестов и ограничений (на время, память, используемые языки. Под тестом понимается описание входных и выходных значений.

При запуске программы проводятся описанные тесты и выдается результат о принятии решения или об ошибке и ее описание. Ошибкой может быть проваленный тест или неудовлетворение ограничениям.

Пользователи должны иметь возможность связи друг с другом. При этом администраторы должны иметь способ написать всем участникам или конкретному пользователю (будь тот администратором или участником), в то время как участники не должны иметь связи друг с другом.

Администраторы должны иметь возможность просмотреть прогресс всех или конкретного участника, а участники должны иметь возможность просмотреть свой прогресс. При этом должно быть показано, сколько задач решено верно и с какой попытки. Также участники должны видеть, на каком они месте, чтобы понять, насколько хорошо они справляются.

Под отправкой решения понимается отправка участником кода с указанием выбранного языка и задачи. После отправки решения проводится автоматический запуск программы, описанный выше.

4.3 Требования к реализации

Для удобства пользования целесообразно реализовать программу в виде веб-приложения. Таким образом, не будет необходимости в специальной настройке каждого рабочего места для обычных олимпиад, а в случае с удаленными олимпиадами пользователям не придется устанавливать дополнительное программное обеспечение.

4.4 Требования к начальному комплекту

Веб-приложение должен поставляться с системой, обеспечивающей хранения данных, а также набором компиляторов и интерпретаторов для используемых языков.

4.5 Требования к составу и параметрам технических средств

Вычислительная машина сервера должна обладать мощностью, соизмеримой с количество пользователей и требования к скорости обработки их запросов. Таким образом, основные ограничения накладываются двумя факторами:

- Реализация хранения данных (что будет рассмотрено далее);
- Количество участников в одной олимпиаде.

Вычислительные машины клиентов должны обладать стандартным набором технических средств:

- Устройства ввода\вывода;
- Сетевая карта.

4.6 Требования к схеме организации данных

Разрабатываемая модель данных должна адекватно описывать предметную области, избегать избыточности и быть расширяемой – т. е. модель должна иметь

возможность развиваться для включения новых требований без потери имеющихся данных и с минимальным влиянием на алгоритмы работы с базой данных.

5. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРИЛОЖЕНИЯ

Для проектирования и разработки программного обеспечения были выделены следующие этапы:

- построение диаграммы использования;
- проектирование базы данных;
- выбор инструментов для разработки приложения;
- создание программного кода.

5.1 Построение диаграммы использования

Ранее (в пунктах 4.1 и 4.2) были описаны функции, который должен предоставлять веб-сервис. На их основании составим основные сценарии использования программы и покажем диаграмму использования на рисунке 5.1.

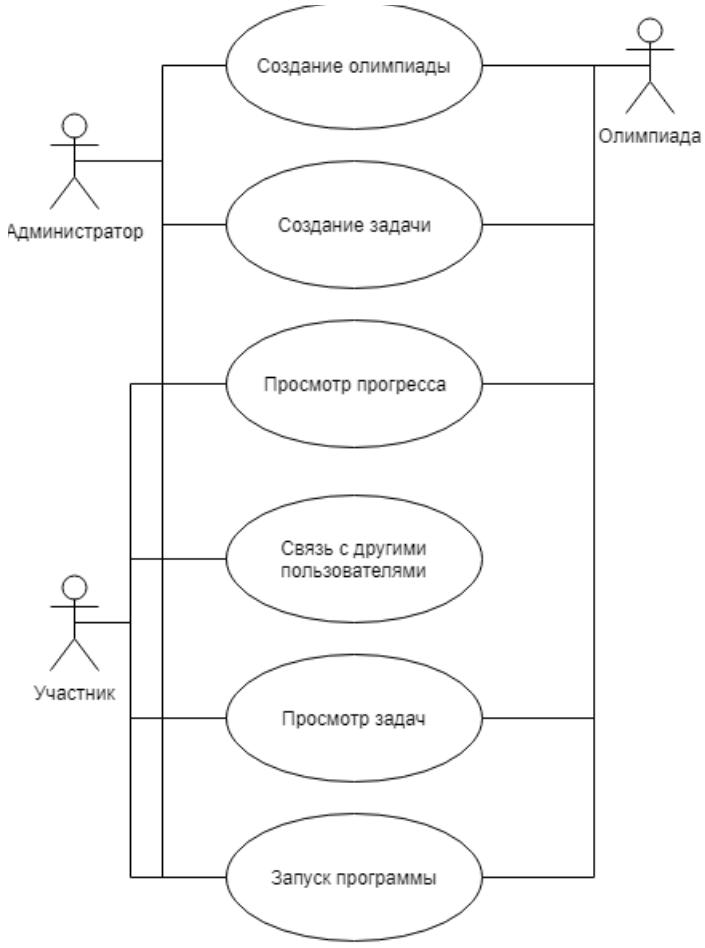

Рисунок 5.1 – диаграмма использования

5.2 Проектирование базы данных

Проектирование базы данных начиняется с представления предметной области в виде отношений, которые постепенно приводятся к необходимому виду. Таким образом, процесс проектирования представляет собой процесс нормализации схем отношений, причем каждая следующая нормальная форма обладает свойствами, в некотором смысле лучшими, чем предыдущая.

Каждой нормальной форме соответствует определенный набор ограничений, и отношение находится в некоторой нормальной форме, если удовлетворяет свойственному ей набору ограничений. Примером может служить ограничение первой нормальной формы – значения всех атрибутов отношения атомарны.

В теории реляционных баз данных обычно выделяется следующая последовательность нормальных форм:

1) первая нормальная форма;

2) вторая нормальная форма;

3) третья нормальная форма;

4) нормальная форма Бойса-Кодда;

5) четвертая нормальная форма;

6) пятая нормальная форма.

Каждая следующая форма предъявляет новые требования, но также включается в себя требования предыдущих форм.

5.2.1 Выделение сущностей и атрибутов

Из диаграммы использования видно, что есть администратор, участник и олимпиада. Администраторов и участников можно хранить как одну сущность «Пользователь». Тогда его атрибутами будет:

- Идентификатор – атрибут с доменом числового типа, служащий ключом;
- Тип пользователя – атрибут с доменом числового типа, показывающий, является администратор или участник. Т. к. типов пользователей всего два и, если и появятся другие, их будет немного, то можно выбрать минимальный по размеру числовой тип данных, доступный в СУБД;
- Логин – уникальный атрибут с доменом строкового типа, являющийся именем Пользователя;
- Хэш пароля – атрибут с доменом строкового типа.

Очевидно, что домены всех атрибутов сущности «Пользователь» не включают в себя неопределенное значение.

Другая сущность – «Олимпиада» – имеет следующие атрибуты:

- Идентификатор – атрибут с доменом числового типа, служащий ключом;
- Название – атрибут с доменом строкового типа;
- Время создания, начала и конца – три атрибута, показывающие время создания олимпиады, время начала и время окончания;
- Продолжительность – атрибут, показывающий время, в течение которого участник, начав проходить олимпиаду, может отправлять решения;
- Тип регистрации – атрибут, показывающий условия для регистрации. На данный момент в олимпиаде могут регистрироваться либо все желающие, либо по приглашению.

Объект «Олимпиада» включает в себя множество решений и задач, которые, в рамках декомпозиции, выделяются в отдельные сущности.

Сущность «Задача» содержит атрибуты:

- Идентификатор – атрибут с доменом числового типа, служащий ключом;
- Название – атрибут с доменом строкового типа;
- Олимпиада – внешний ключ, показывающий, какой олимпиаде принадлежит задача;

- Описание – атрибут с доменом строкового типа, необходимый для описания условий задачи участникам олимпиады;
- Награда – атрибут с доменом числового типа, показывающий количество условных очков, которые будут даны за решение задачи;
- Время и память – атрибуты с доменом числового типа, показывающие лимиты времени и памяти, которые не должны быть превышены;
- Тесты – атрибут с доменом строкового типа, описывающий тесты для этого решения.

Сущность «Решение» содержит атрибуты:

- Идентификатор – атрибут с доменом числового типа, служащий ключом;
- Код – атрибут с доменом строкового типа, хранящий путь до файла, где записан программный код, являющийся решением для данной задачи и написанный участниками.
- Автор – внешний ключ, показывающий, какой участник (пользователь) является автором этого решения;
- Исполняющий модуль – внешний ключ (см. следующую сущность), необходимый, чтобы показать, какой именно компилятор или интерпретатор должен быть использован;
- Задача – внешний ключ, показывающий, решением какой задачи является данное решение.

Сущность «Исполняющий модуль» имеет следующие атрибуты:

- Идентификатор – атрибут с доменом числового типа, служащий ключом;
- Название – атрибут с доменом строкового типа, используемый для удобства пользования, поэтому он может принимать неопределенное значение;
- Язык – атрибут, показывающий, какому именно языку программирования принадлежит данный компилятор или интерпретатор. Целесообразно выделить отдельный словарь для языков. Тогда, данный атрибут будет являться внешним ключом;

- Исполняемый файл – атрибут с доменом строкового типа, показывающий путь до компилятора или интерпретатора;
- Параметры – атрибут с доменом строкового типа на случай, если компилятору или интерпретатору необходимо подать на вход дополнительные параметры.

Кроме того, необходимо создать две таблицы для следующих связей «многие-ко-многим»:

- «Олимпиада»-«Пользователь» – для хранения данных об участниках олимпиады;
- «Олимпиада»-«Язык» – одно из ограничений, которое накладывается в рамках олимпиады по программированию – на используемые языки программирования.

Были введены следующие каскадные ограничения целостности:

- Для сущности «Решение» при удалении пользователя происходит присвоение неопределенного значения. Тогда удаление пользователя не испортит записи в олимпиадах, где он принимал участие;
- Для сущности «Задача» при удалении Олимпиады происходит каскадное удаление, т. к. в сохранение данной записи больше нет необходимости;
- Для сущности «Решение» при удалении Задачи происходит каскадное удаление, т. к. в сохранение данной записи больше нет необходимости;
- Для сущности «Исполняющий модуль» при удалении языка происходит присвоение неопределенного значения, т. к. на сервере все еще хранится файл (компилятор или интерпретатор).

Таким образом, наша база данных примет следующий вид:

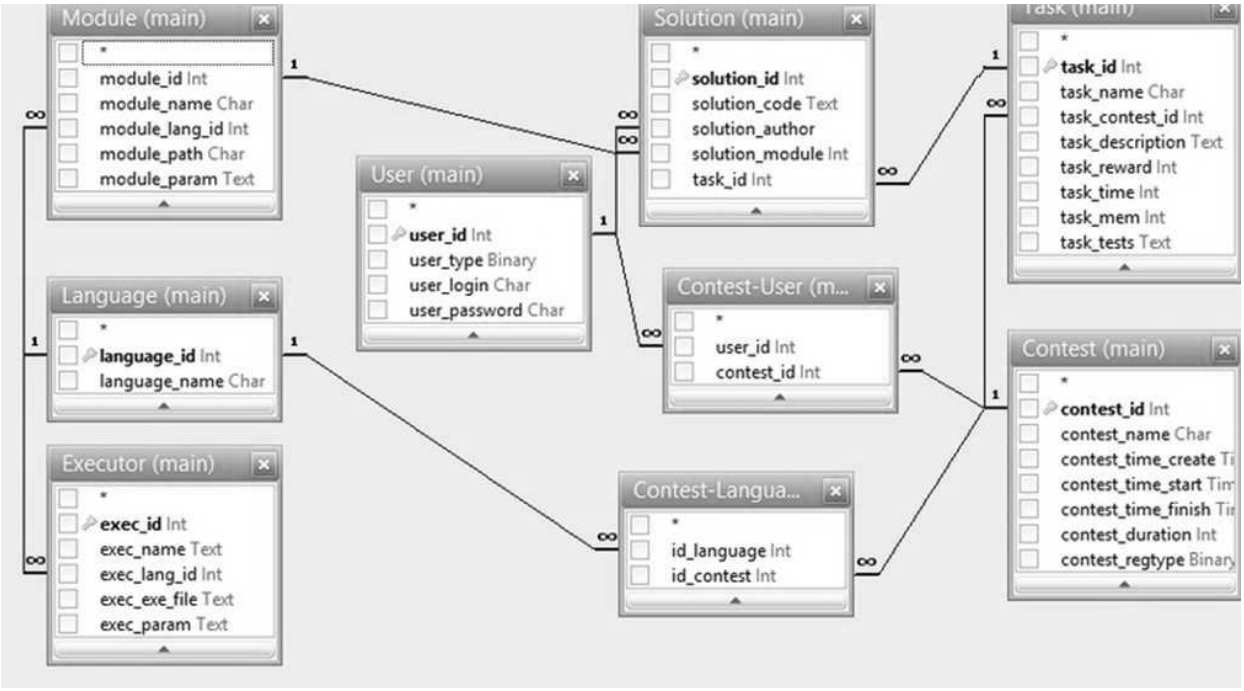

Рисунок 5.2.1 – схема базы данных

5.3 Выбор инструментов для разработки приложения

Веб-приложение – клиент-серверное приложение, в котором клиент взаимодействует с сервером при помощи браузера, а за сервер отвечает – веб-сервер. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются межплатформенными службами.

5.3.1 Классификация платформ

Разработка веб-приложения, как и любого другого ПО, с нуля нецелесообразно, необходимо использовать некую платформу. Платформы предоставляют инструментарий и так же влияют на реализацию. Обычно,

платформа для веб-разработки предполагает использование некоторого шаблона проектирования – паттерна – т. е. повторимой архитектурной конструкции, представляющей собой решение проблемы проектирования в рамках некоторого часто возникающего контекста [6]. Предпочтительней использовать при разработке паттерны, т. к. это снижает сложность разработки за счет готовых абстракций для решения целого класса проблем.

Для разработки веб-приложения существует классификацию шаблонов проектирования Мартина Фаулера [7]. Согласно этой классификации, шаблоны можно разделить по классам:

1. базовые шаблоны;
2. шаблоны веб-представления;
3. шаблоны архитектурных источников данных;
4. шаблоны объектно-реляционной логики;
5. шаблоны объектно-реляционного структурирования;
6. шаблоны логики сущности;
7. шаблоны распределения данных;
8. шаблоны локальной конкуренции.

Одним из самых популярных шаблонов, нашедший применение и в вебе, и в мобильной разработке, является MVC и его производные:

1. MVP (Model – View – Presenter);
2. MVVM (Model – View – View – Model);
3. HMVC (Hierarchical MVC);
4. PAC (Presentation – Abstraction – Control).

Основная функция MVC состоит в отделении бизнес-логики (модели) от пользовательского интерфейса (представления), а связь между ними обеспечивается контроллером. Представление и контроллер зависят от

модели, которая в свою очередь независима, что позволяет строить модель отдельно визуального представления [8].

MVP – это шаблон проектирования пользовательского интерфейса, облегчающий автоматическое модульное тестирование и отделяющий логику от отображения. В MVP Presenter становится посредником, аналогично контроллеру в MVC, но отличие состоит в том, что он также отвечает за управление событиями пользовательского интерфейса, в то время как в MVC их обработка была ответственностью за представлениями (View).

MVVM также позволяет отделить логику приложения от визуальной части (представления), он является архитектурным паттерном и отличается более «тесной» связью между моделью и ее представлением благодаря слою «Представление-Модель», синхронизирующему данные при событии со стороны модели или представления. Такая концепция реализована в WPF и Silverlight [9].

HMVC – расширение MVC, решающее некоторые проблемы масштабируемости приложений. Согласно паттерну, каждая отдельная MVC группа является независимым от других слоем иерархической структуре, способным обратиться к контроллеру другой триады. Такой подход облегчает разработку сложных приложений, их дальнейшую поддержку и масштабирование, а также способствует повторному использованию кода.

5.3.2 ASP.NET Core

ASP.NET Core – это универсальная платформа разработки, которая поддерживается корпорацией Майкрософт и сообществом .NET на сайте GitHub. ASP.NET Core был разработан как следующее поколение ASP.NET и имеет такие преимущества как:

- В отличие от ASP.NET, он работает так же на .NET CORE, что делает его кроссплатформеным и позволяет запускать на любой платформе, имеющей .NET Framework или .NET Core: Windows, Mac OS, Linux, облака, внедренные системы и в сценариях Интернета вещей;
- Больший контроль над сборкой без привязки к IIS и System.Web.dll;
- Встроенный функционал для внедрения зависимостей;
- Открытый исходный код.

Кроме того, Core приложение теперь унифицировано с другими типами приложений, т. е., оно таким же образом включает метод Main класса Program, который вызывается при запуске приложения, а тот в свою очередь запускает сервис [10].

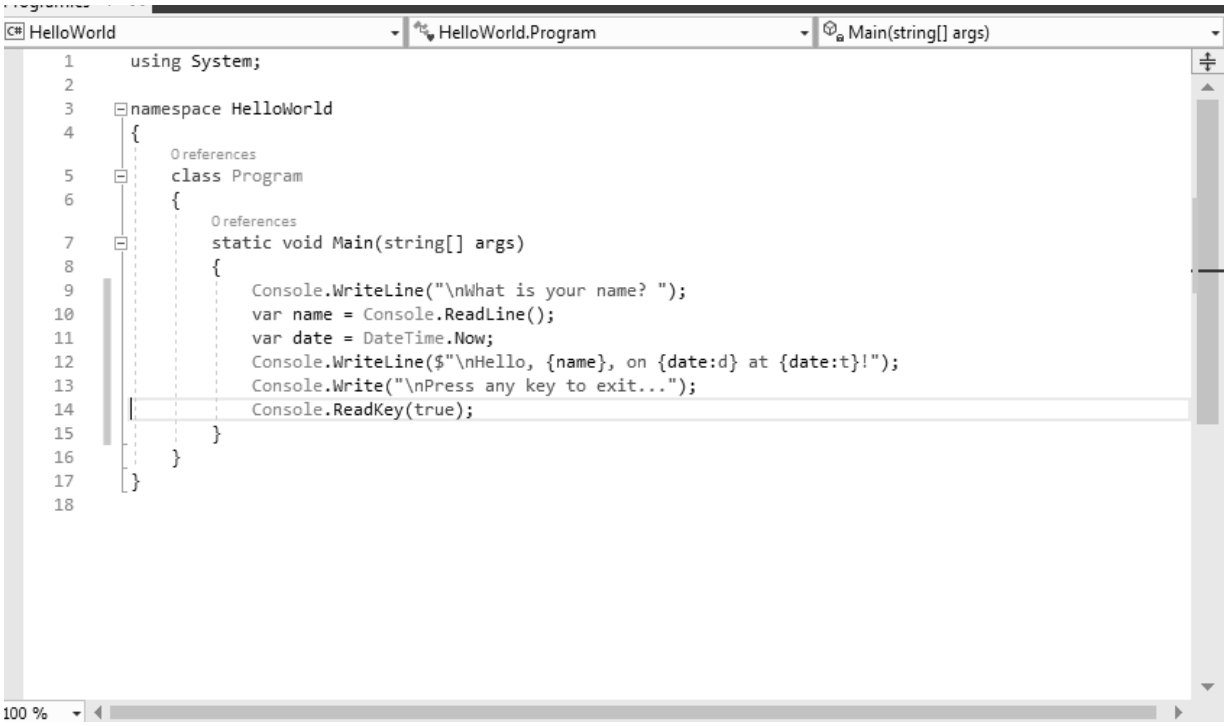

Рисунок 5.3.1 – запуск сервиса на ASP.NET Core

5.3.3 Django

Django – это Open Source фреймворк для создания веб-приложений различной сложности на языке Python. Преимуществами Django являются:

- быстрая разработка высокопроизводительных и полнофункциональных сайтов;
- масштабируемость;
- расширяемость;
- абстрагирование от низкого уровня.

Как и в ASP.NET Core, веб-приложения на Django разрабатываются согласно архитектуре MVC, однако в рамках Django она называется MTV (Model – Template – View).

Django базируется на классе Python django.db.models.Model, который задает данные модели так, чтобы они были пригодны к использованию на Web-сайтах. Эти данные определяются соответствующими атрибутами объектов, которые сохраняются в базе данных в процессе работы. При создании сайта создается подкласс класса Model и добавляется поле членов в класс для задания специфических данных. Кроме того, с помощью библиотек можно автоматически преобразовать такие классы в таблицы в СУБД.

Благодаря всему вышесказанному и наличию встроенного административного интерфейса (который при желании можно переопределить), Django позволяет быстро создать веб-приложение и сконцентрироваться на его логике. На рисунке 5.2.2 показан административный интерфейс [11].

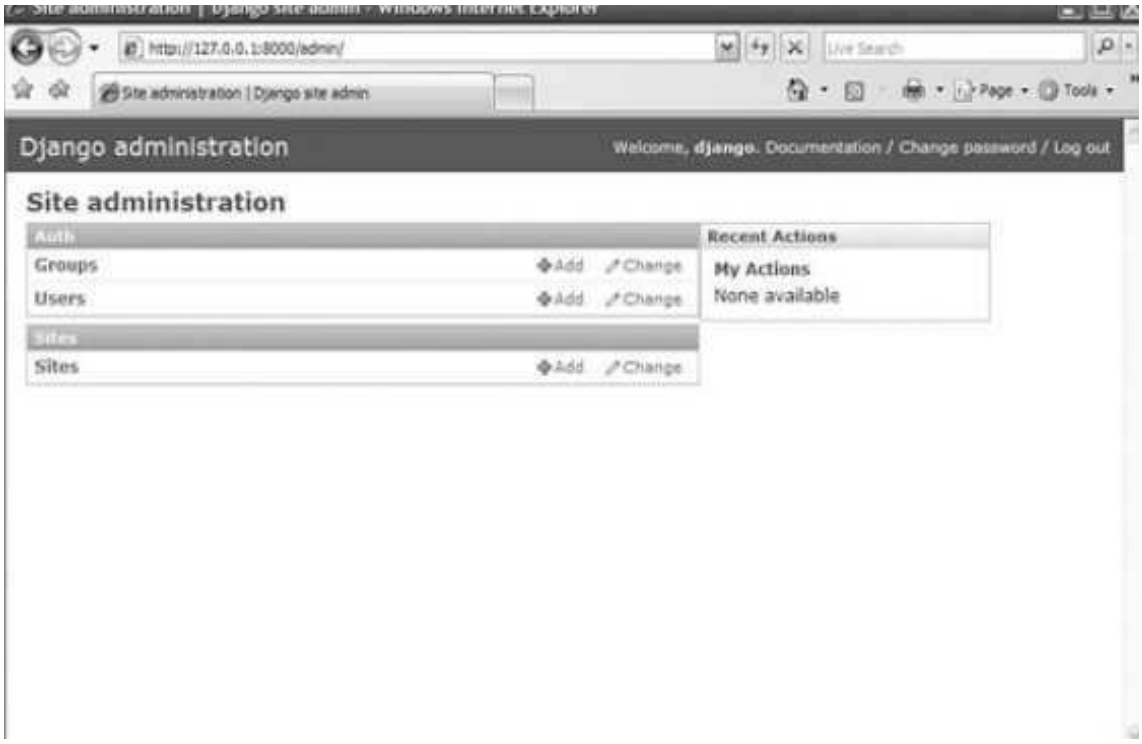

Рисунок 5.3.2 – Окно доступа администрирования

5.3.4 Rails

Rails – полноценный, многоуровневый фреймворк для построения веб-приложений, использующих базы данных, который основан на архитектуре MVC и языке Ruby.

Rails отлично работает со многими веб-серверами и СУБД. В качестве веб-сервера рекомендуется использовать Apache или nginx с модулем Phusion Passenger, но также можно использовать Unicorn, Thin, Mongrel или FastCGI. В качестве СУБД можно использовать MySQL, PostgreSQL, SQLite, Oracle, SQL Server, DB2 или Firebird. Использовать Rails можно на практически любой операционной системе, однако для развертывания рекомендуется использовать системы семейства *nix [12].

Таким образом, к плюсам можно отнести:

- Множество библиотек и расширений;

- Скорость разработки.

К минусам относятся:

- Множество зависимостей;
- Скорость выполнения;
- Проблемы с многопоточностью.

5.3.5 PHP

PHP – скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов. Язык и его интерпретатор (Zend Engine) разрабатываются группой энтузиастов в рамках проекта с открытым кодом. Проект распространяется под собственной лицензией, несовместимой с GNU GPL.

PHP имеет такие преимущества, как:

- Быстрая разработка приложения;
- Большое сообщество и экосистема;
- Хорошая документация;
- Интеграция со средой;
- Хорошая масштабируемость.

К недостаткам относятся:

- Возможные проблемы с кроссплатформенностью;
- Новые версии не поддерживают старые;
- Проблемы с многопоточностью.

5.3.5 Вывод

Таким образом, наилучшим выбором будет фреймворк для Python Django за счет скорости разработки, масштабируемости и расширяемости. Кроме того, на Июнь 2018 Python является третьим по популярности языком, что позволяет при выборе Django быть уверенным в документации и иной поддержки со стороны сообщества [14].

5.4 Организация данных

Существует два способа организации данных:

1. Локальные файлы, поддерживаемые функциональными пакетами прикладных программ;
2. База данных, управляемая системой управления базами данных (СУБД).

Локальные файлы позволяют сократить время для обрабатывания данных, однако недостатками является организация данных при изменении большого объема, а также недостаток гибкости. Поэтому локальные файлы могут использоваться только в специальных приложениях, требующих при вводе необходимой информации требуется немалая реакция.

Второй вариант – база данных – это собрание взаимосвязанных, хранящиеся вместе данных при такой избыточности, которая разрешает их использование нормальным образом. Использование СУБД позволяет обеспечить соединимость этих данных, а также управлять доступом и разграничивать права.

Существуют клиент-серверные, файл-серверные и встраиваемые СУБД.

В файл-серверных СУБД данные централизованно находятся на файл-сервере, а СУБД – на каждом клиентском устройстве. Доступ к данным в таком случае происходит через локальную сеть, а синхронизация чтений и обновлений – через файловые блокировки. За счет этого снижается нагрузка на сервер, но:

- увеличивается нагрузка на локальную сеть;
- усложняется централизованное управление;

- становится труднее обеспечивать надежность и безопасность.

На данный момент данная технология считается устаревшей. Тем не менее, к плюсам технологии можно отнести низкую цену создания и внесения изменений. Примером такой СУБД является Microsoft Access.

В клиент-серверных СУБД сама система расположена на сервере вместе с данными, а запросы от клиентов обрабатываются централизованно. В результате повышаются требования к серверу, но уменьшается нагрузка на локальную сеть и повышается надежность и безопасность. Кроме того, данное решение позволяет распределить функции системы между несколькими вычислительными машинами, в следствие чего можно заменять, ремонтировать или обновлять сервер без отключения всего комплекса. Однако такое решение намного дороже, т. к. требует более мощного оборудования, сложного программного обеспечения и специализированного работника.

Примерами такой СУБД являются Oracle Database, MS SQL Server, PostgreSQL и MySQL.

5.4.1 Oracle Database

Oracle Database – СУБД от Oracle, чье ядро поставляется в одном из четырех вариантов:

1. Enterprise Edition – для систем масштаба крупной организации;
2. Standard Edition – для организации среднего масштаба или подразделения в составе крупной организации;
3. Personal Edition – для персонального использования;
4. Lite – управление реляционными данными для встраиваемых систем и мобильных устройств при наличии или отсутствии сетевого подключения.

Помимо этого, Oracle предлагается дополнительные слуги вроде параллельного сервера, iFS и т.д.

Oracle Database предоставляет множество разнообразных решений, но сложно в освоение и часто дорого.

5.4.2 Microsoft SQL Server

Microsoft SQL Server – это СУБД от Microsoft. Одной из особенности СУБД является язык Transact-SQL, созданный Microsoft вместе Sybase и представляющий собой стандарт ANSI/ISO по SQL с дополнениями. Данная СУБД подходит для баз всех размеров баз данных масштаба компании; конкурирует с другими СУБД на рынке. Еще одним преимуществом является технология SQL Server Always On, позволяющая снизить время нахождения сервера в неисправном состоянии [14].

5.4.3 PostgreSQL

PostgreSQL – свободно распространяемая объектно-реляционная СУБД, позволяющая использовать помимо SQL так же Python и PERL. Имеет большое количество расширений и открытый код. Преимуществами PostgreSQL являются:

- Встроенная поддержка для пользовательских процедур;
- Легкий переход на платные СУБД вроде Oracle Database, что делает PostgreSQL хорошим временным решением;
- Возможность создания сложную структуру данных;

В то время как к недостаткам можно отнести:

- Скорость;
- Сложность настройки.

5.4.4 MySQL

MySQL – свободная СУБД, принадлежащая Oracle Corporation после поглощения этой корпорацией компании Sun Microsystems и распространяется

под GNU General Public License или под своей лицензией. MySQL так же является популярным решением и входит, например, в состав WAMP и в сборки серверов Денвера. MySQL может применяться в качестве сервера, к которому обращаются собственные или удаленные клиенты, или включаться в самостоятельные программы. Другим плюсом является то, что именно работа с MySQL встроена в фреймворк Django, в то время как для работы с другими СУБД необходимо подключать сторонние библиотеки.

5.4.5 Вывод

Наихудшим вариантов является PostgreSQL, в то время как остальные СУБД подходят для поставленной задачи, но за счет простоты и встроенной поддержки в Django целесообразно выбрать MySQL.

6. РЕАЛИЗАЦИЯ

Исходя из вышесказанного можно выделить следующую архитектуру:

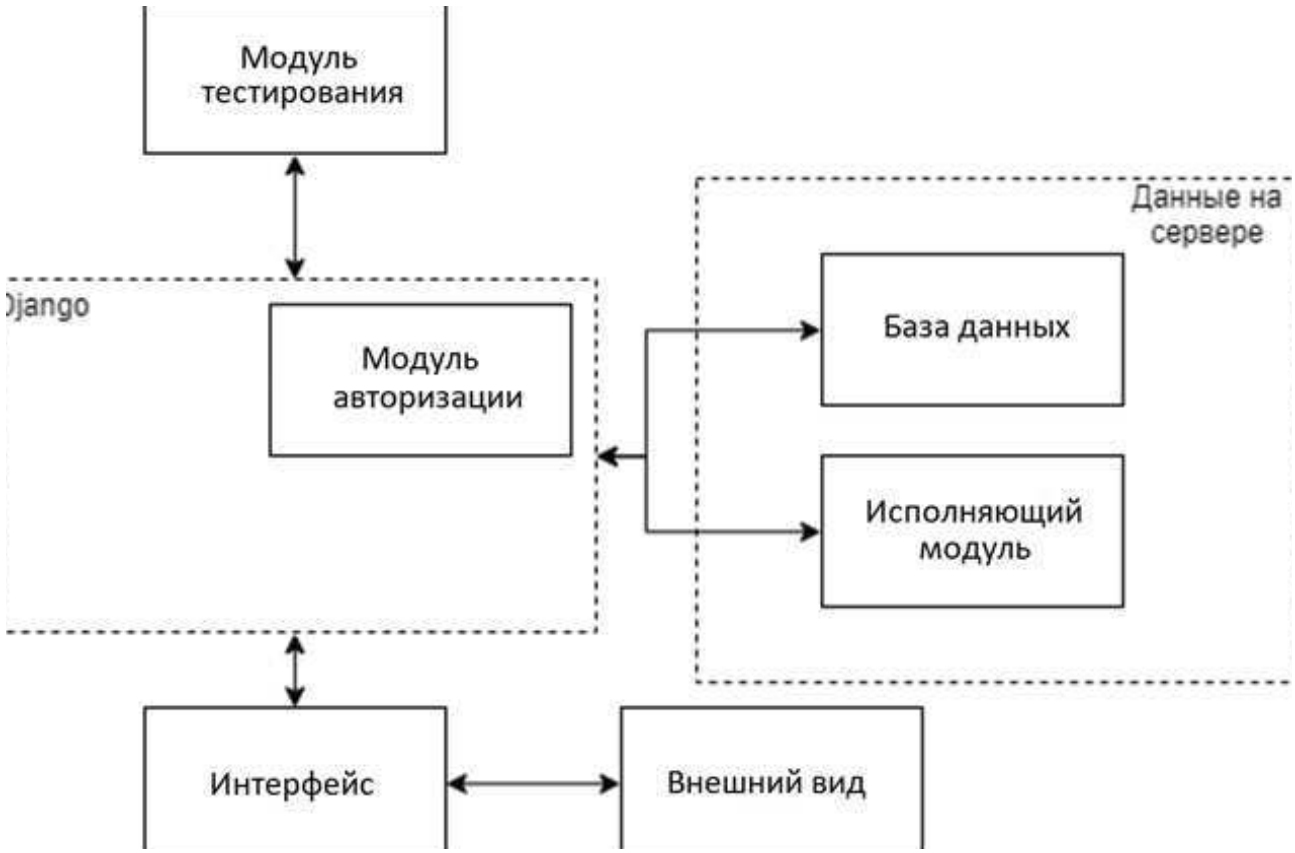

Рисунок 6.1 – архитектура программы

На сервере у нас хранятся данные об олимпиадах, участниках, их действия, а также о параметрах исполняющих модулей, которые хранятся отдельно.

Также есть модуль тестирования, ответственный за запуск программы, получаемой из интерфейса, с помощью исполняющего модуля.

Фреймфорк Django берет на себя связь между компонентами, работу с СУБД, обработку запросов, создание html-страниц и т. д.

6.1 Пример внешнего вида и соответствующего интерфейса

В Django, согласно MTV, программа разделена на логику слой доступа к данным, слой представления данных и слой бизнес-логики. Внешний вид

называется View и описывается с помощью html, а интерфейс называется Form и описывается на Python.

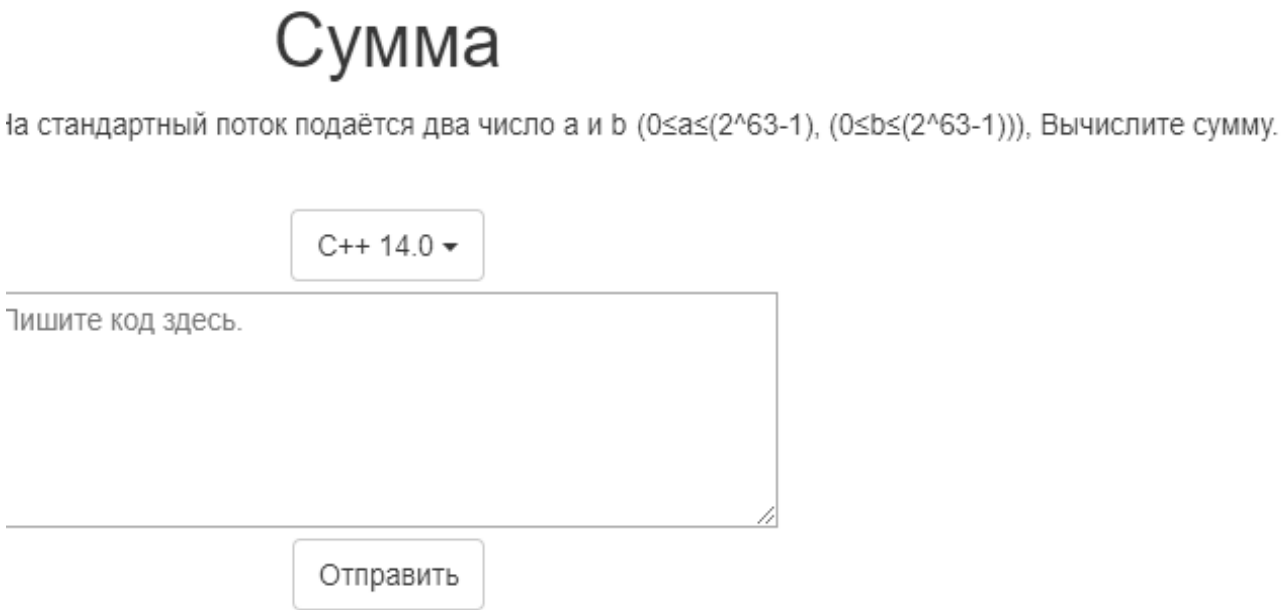

Рисунок 6.1.1 – частичное представление для решения задачи

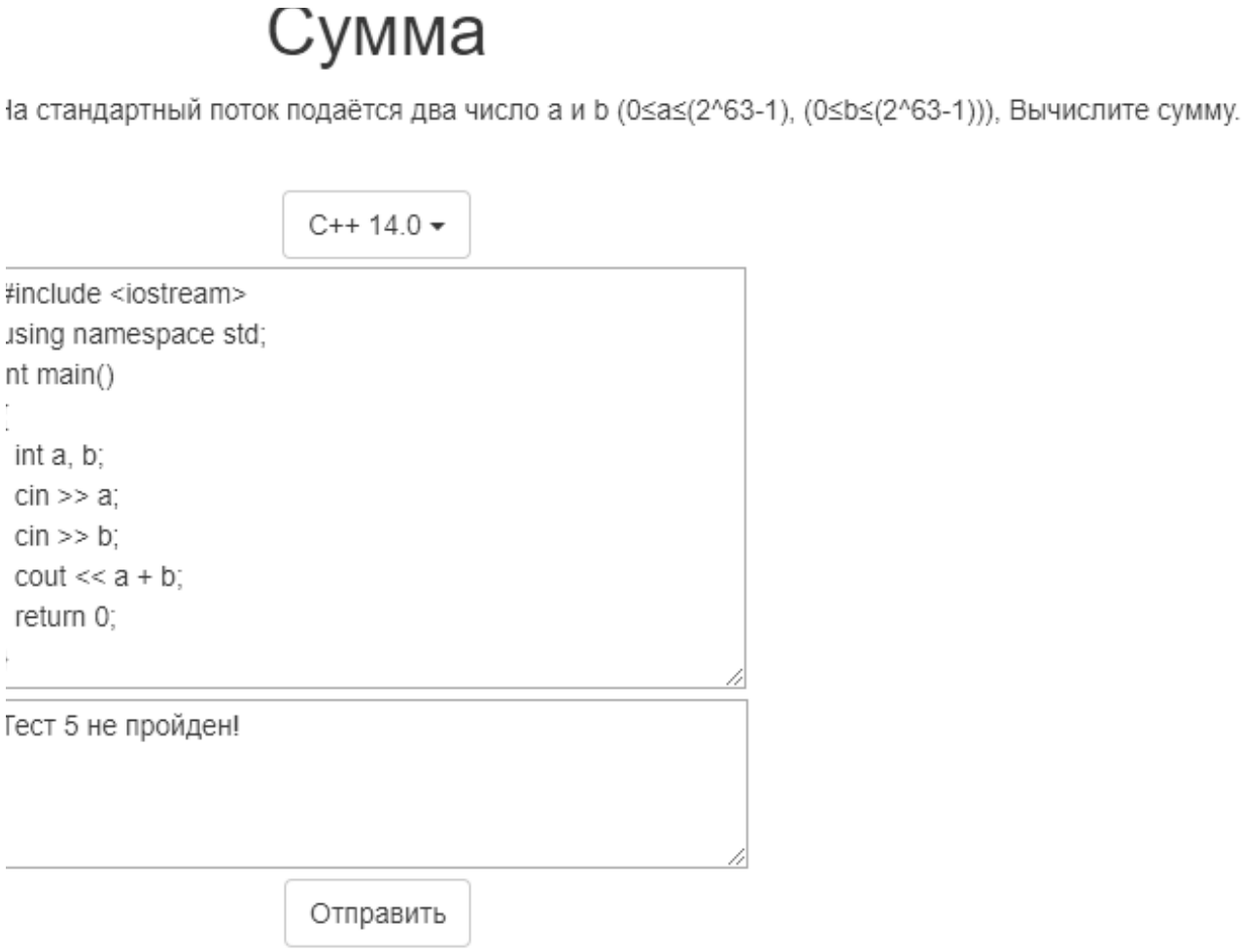

Рисунок 6.1.2 – частичное представление для решения задачи с неверным ответом

В коде данное частичное представление выглядит так («solutions_list.html»):

```
{% extends 'solution/solutions.html' %}

{% block content %}
<form method="POST" class="solution-form">{% csrf_token %}
  <div class="text-center">
    <h1><a href="/solution/{{ solution.pk }}/">{{ solution.name }}</a></h1>
 <p align="left"><a href="/solution/{{ solution.pk }}/">{{ solution.description }}</a></p>
 <button type="button" class="btn btn-default dropdown-toggle ticket-dropdown-menus" data-
toggle="dropdown" aria-haspopup="true" aria-expanded="false">
       <a href="/getlanguage/{{ solution.pk }}/">{{ language.name }}</a>
    <span class="caret" />
 </button>
    <ul class="dropdown-menu">
      {% for language in <a href="/getlanguage/{{ solution.pk }}/"></a> %}
      <li>
        <a href="#">{{ language.name }}</a>
      </li>
      {% endfor %}
    </ul>
  </div>
  <div>
    <div>
```

```
      <textarea placeholder="Пишите код здесь."><a href="/solution/{{ solution.pk }}/">{{
solution.code }}</textarea>
    </div>
    {% if <a href="/solution/send/{{ solution.pk }}/{{ language.pk }}"> is not None %}
      <textarea><a href="/solution/send/{{ solution.pk }}/{{ language.pk }}"></textarea>
    {% endif %}
  </div>
 <button type="button" class="btn btn-default" aria-haspopup="true" aria-
expanded="false">Отправить</button>
</form>
    {% endblock %}
```

Форма:

```
class SolutionForm(forms.ModelForm):

  class Meta:
    model = Solution
    fields = ('name', 'code', 'language',)
```

Шаблон адреса:

```
urlpatterns = [
  url(r'^$', views.solution_list, name='solutions'),
  url(r'^(?P<psolution_id>\d+)/$', views.solution, name='solution'),
]
```

Функции соответствующих запросов:

```
def solution_list(request):
    solutions = solution.objects.all()
    return render(request, 'solution/solution_list.html', {'solutions' : solutions})

def solution(request, solution_id):
    solution = solution.objects.get(pk=solution_id)
        return render(request, 'solution/solution.html', {'solution' : solution})
```

6.2 Модели

Рассмотрим модель, реальзиующую сущность «Решение» функции тестирования:

```
from django.db import models
import subprocess
from subprocess import Popen
import time
import pexpect
from pexpect import popen_spawn, expect
import fileinput

class Solution(models.Model):
    executor    = models.ForeignKey(Executor)
    code        = models.FileField(upload_to='/solutions/"{0}"/{1}'.format(instance.author,
instance.id))
    author      = models.ForeignKey('user.Id',
                                    on_delete=models.SET_NULL)
    task        = models.ForeignKey('task.Id',
```

```
                                    on_delete=models.CASCADE)

    def __str__(self):
        return self.name

    def publish(self):
        self.save()

    def test(self):
        if(instance.executor.type == Executor.Compiler):
            exe_path = '/exe/"{0}"/{1}'.format(instance.author, instance.id)
            subprocess.call([instance.executor.executable.path, instance.code.path,
instance.executor.parameters, exe_path])
        tests = task.parse_inputoutput()
        test_number = 0
        for test in tests:
            if(instance.executor.type == Executor.Compiler):
                p = popen_spawn.PopenSpawn(exe_path)
            else:
                p = popen_spawn.PopenSpawn([instance.executor.executable.path,
instance.code.path, instance.executor.parameters])
            output = ''
            start_time = time.time()
            for input in test[0]:
                p.sendline(input)
                p.expect([r'.+'])
                if(p.after is not None):
                    output.append(p.after)
                if(output != test[1]):
                    raise ValueError('Failed at test {0}. Incorrect
output.'.format(test_number))
                test_number += 1
            elapsed_time = time.time() - start_time
            if(elapsed_time > task.maxtime):
                    raise    ValueError('Failed    at    test    {0}.    Too    much    time
spent.'.format(test_number))
```

В данной функции стоит отметить использование библиотеки pexpect. Pexpect – это модуль Python для порождения дочерних приложений, контроля над ними и реагирования на ожидаемые закономерности. Модуль Pexpect основан на Expect – расширении к скрипт-языку Tcl для автоматизации и тестирования в ОС Unix, которое используется для таких интерактивных приложений telnet, ftp, passwd, fsck, rlogin, tip, ssh, и других.

6.3 Работа с СУБД

Для работы с СУБД необходимо воспользоваться скриптом manage.py с параметром syncdb, т. е.:

$ python manage.py syncdb

Однако, перед этим необходимо в settings.py прописать:

```
DATABASES = {
  'default': {
    'ENGINE': 'django.db.backends.mysql',
    'NAME': os.path.join(BASE_DIR, 'db.mysql'),
  }
}
```

Затем, после инициализации базы данных, необходимо ее заполнить. Для заполнения будет опять использоваться скрипт manage.py, для чего будут написаны модели, описанные ранее, после чего будут созданы соответствующими таблицы следующими двумя команда:

$ python manage.py makemigrations

$ python manage.py migrate

6.4 Работа с пользователями

Мы настроили базу данных, в которой Django будет хранить пользователей сервера в системной таблице. Отметим, что пользователи сервера и пользователи веб-сервиса – не одно и то же. В Django по умолчанию включает соответствующие модули для работы с пользователями:

```
INSTALLED_APPS = [
  'django.contrib.auth',
]


MIDDLEWARE_CLASSES = [
  'django.middleware.security.SecurityMiddleware',
]




AUTH_PASSWORD_VALIDATORS = [
  {
    'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
  },
  {
    'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
  },
  {
    'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
  },
```

```
  {
    'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
  },
]
```

По умолчанию, Django использует PBKDF2 – стандарт формирования ключа на основе пароля, рекомендуемый Национальным институтом стандартов и технологий США – вместе с алгоритмом криптографического хеширования SHA26.

ЗАКЛЮЧЕНИЕ

В ходе выпускной квалификационной работы:

- Были рассмотрены существующие программные решения;
- Были выдвинуты требования к системе;
- Были рассмотрены платформы для реализации веб-приложения;
- Были рассмотрены варианты организации хранения данных;
- Была выбрана архитектура;
- Реализована системе.

Полученное решение веб-приложение может быть использовано для проведения спортивных состязаний по программированию, онлайн-олимпиад по программированию или для автоматизации проверки работ студентов вводных курсов языков соответствующих дисциплин.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Финал чемпионата мира по программированию 2014 [Электронный ресурс] – Режим доступа: http://www.icpc2014.ru/ru. – Заглавие с экрана. - (Дата обращения: 28.05.2018).

2. SPOJ – платформа для онлайн-судейства [Электронный ресурс] – Режим доступа: https://www.spoj.com/. – Заглавие с экрана. - (Дата обращения: 28.05.2018).

3. Toph – платформа для спортивного программирования [Электронный ресурс] – Режим доступа: https://toph.co/. – Заглавие с экрана. - (Дата обращения: 28.05.2018).

4. PC^2 – Система контроля соревнований по программированию [Электронный ресурс] – Режим доступа: https://pc2.ecs.csus.edu/. – Заглавие с экрана. - (Дата обращения: 28.05.2018).

5. Kattis – архив проблем Kattis [Электронный ресурс] – Режим доступа: https://open.kattis.com/. – Заглавие с экрана. - (Дата обращения: 28.05.2018).

6. Фаулер М. Шаблоны корпоративных приложений. – М.: Издательский дом «Вильямс», 2012. – 544 с.

7. Сайт Мартина Фаулера [Электронный ресурс] – Режим доступа: http://martinfowler.com/eaaCatalog/index.html. – Заглавие с экрана. - (Дата обращения: 28.05.2018).

8. Myer T. Professional CodeIgniter. – USA: Wiley Publishing, Inc., 2008. – 339 p.

9. Натан А. WPF 4. Подробное руководство. – СПб.: «СимволПлюс», 2011. – 880 с.

10. Хабрхабр, Путь ASP.NET Core [Электронный ресурс] – Режим доступа: https://habr.com/post/312226/. – Заглавие с экрана. - (Дата обращения: 28.05.2018).

11. Фреймворк Django [Электронный ресурс] – Режим доступа: https://www.djangoproject.com/. – Заглавие с экрана. - (Дата обращения: 28.05.2018).

12. Michael H. Ruby on Rails Tutorial 4th edition. –USA: Softcover, 2012 – 405 p.

13. Индекс TIOBE [Электронный ресурс] – Режим доступа: https://www.tiobe.com/tiobe-index/. – Заглавие с экрана. - (Дата обращения: 28.05.2018).

14. SQL Server 2016 [Электронный ресурс] – Режим доступа: https://www.microsoft.com/ru-ru/sql-server/sql-server-2016. – Заглавие с экрана. - (Дата обращения: 28.05.2018).

ПРИЛОЖЕНИЕ А. Исходный код

Файл «models.py»:

```
from django.db import models
import subprocess
from subprocess import Popen
import time
import pexpect
from pexpect import popen_spawn, expect
import fileinput

class Language(models.Model):
    name = models.TextField(null=True)

    def __str__(self):
        return self.name

    def publish(self):
        self.save()


class Solution(models.Model):
    executor    = models.ForeignKey(Executor)
    code        = models.FileField(upload_to='/solutions/"{0}"/{1}'.format(instance.author,
instance.id))
    author      = models.ForeignKey('user.Id',
                                    on_delete=models.SET_NULL)
    task        = models.ForeignKey('task.Id',
                                    on_delete=models.CASCADE)

    def __str__(self):
        return self.name

    def publish(self):
        self.save()

    def test(self):
        if(instance.executor.type == Executor.Compiler):
            exe_path = '/exe/"{0}"/{1}'.format(instance.author, instance.id)
            subprocess.call([instance.executor.executable.path, instance.code.path,
instance.executor.parameters, exe_path])
        tests = task.parse_inputoutput()
        test_number = 0
        for test in tests:
            if(instance.executor.type == Executor.Compiler):
                p = popen_spawn.PopenSpawn(exe_path)
            else:
                p = popen_spawn.PopenSpawn([instance.executor.executable.path,
instance.code.path, instance.executor.parameters])
            output = ''
            start_time = time.time()
            for input in test[0]:
                p.sendline(input)
                p.expect([r'.+'])
                if(p.after is not None):
                    output.append(p.after)
                if(output != test[1]):
                    raise ValueError('Failed at test {0}. Incorrect
output.'.format(test_number))
                test_number += 1
```

```
            elapsed_time = time.time() - start_time
            if(elapsed_time > task.maxtime):
                raise ValueError('Failed at test {0}. Too much time
spent.'.format(test_number))


class UsersType(Enum):
    Paticipant  = 1
    Admin       = 2

class User(models.Model):
    type        = models.SmallIntegerField(
                                choices=[(tag, tag.value) for tag in UsersType],
                                default=UsersType.Paticipant)
    login       = models.CharField(max_length=20,
                                unique=True)
    password    = models.CharField(max_length=50)

    def __str__(self):
        return self.login

    def publish(self):
        self.save()


class Task(models.Model):
    name            = models.CharField(max_length=50)
    olympiad        = models.ForeignKey('olympiad.Id',
                                        on_delete=models.CASCADE)
    inputoutput     = models.TextField()
    description     = models.TextField()
    reward          = models.IntegerField()
    maxtime         = models.IntegerField()
    maxmemory       = models.IntegerField()

    def __str__(self):
        return self.name

    def publish(self):
        self.save()

    def parse_inputoutput(self):
        try:
            tests = inputoutput.split('\r\r\r')
            io_tests = []
            for test in tests:
                temp_test = test.split('\r\r')
                io_tests.append([temp_test[0], temp_test[1]])
            return io_tests
        except IndexError:
            raise IndexError('Test file is not correct for task with
id={0}'.format(instance.id))


class OlympiadRegistrationMode(Enum):
    Open  = 1
    Close = 2

class Olympiad(models.Model):
    name             = models.CharField(max_length=70)
    describtion      = models.TextField()
    creation_date    = models.DateTimeField(default=datetime.now)
    start_date       = models.DateTimeField(default=datetime.now)
```

```
    end_date         = models.DateTimeField(default=datetime.now)
    duration         = models.TimeField()
    registrationMode = models.SmallIntegerField(choices=[(tag, tag.value) for tag in
UsersType])
    users            = models.ManyToManyField(User)
    languages        = models.ManyToManyField(Language)

    def __str__(self):
        return self.name

    def publish(self):
        self.save()


class Executor(Enum):
    Compiler  = 1
    Interpretator = 2

class Executor(models.Model):
    name       = models.TextField(null=True)
    language   = models.ForeignKey('language.Id',
                                   on_delete=models.SET_NULL)
    executable = models.FileField(upload_to='/executors/"{0}"/{1}'.format(instance.language,
instance.id))
    parameters = models.TextField()
    type = models.SmallIntegerField(choices=[(tag, tag.value) for tag in Executor])


    def __str__(self):
        return self.name is not None if self.name else
'/executors/"{0}"/{1}'.format(instance.language, instance.id)

    def publish(self):
        self.save()
```

Файл «settings.py»:

```
import os
import posixpath

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


SECRET_KEY = [redacted]

DEBUG = True

ALLOWED_HOSTS = []


INSTALLED_APPS = [
    'app',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'program',
]
MIDDLEWARE_CLASSES = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
```

```
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.auth.middleware.SessionAuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'DjangoWebProject1.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'DjangoWebProject1.wsgi.application'


DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}


AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'Europe/Ekaterinburg'
USE_I18N = True
USE_L10N = True
USE_TZ = True
STATIC_URL = '/static/'
STATIC_ROOT = posixpath.join(*(BASE_DIR.split(os.path.sep) + ['static']))
```