

Факультет Информационных технологий

Кафедра Прикладной информатики и мультимедийных технологий

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

(бакалаврская работа)

на тему: «\_Программная реализация сбора экспертных оценок с применением метода попарного сравнения\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_»

название ВКР

Направление подготовки **09.03.03** – «Прикладная информатика»

(код) (наименование)

Направленность: «Прикладная информатика в психологии»

(наименование)

Студент \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ /Н.И. Верхогляд/

(подпись) (дата)

**Оригинальность текста – \_**89,2**\_ %**

Начальник отдела МКПО \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ /С.Н.Антипова/

(подпись) (дата)

**«Допустить к защите»**

Зав. кафедрой \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ /Л.С.Куравский/

(подпись) (дата)

Руководитель \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ / Л.С.Куравский /

(подпись) (дата)

Консультант (при наличии) \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ / И.С. Колотовкин /

(подпись) (дата)

Рецензент \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ /Е.Д. Куланин/

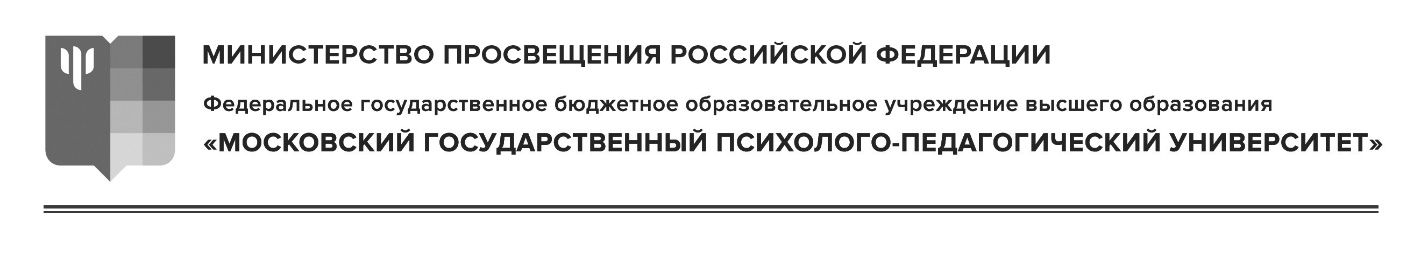
(подпись) (дата)

*Печатная версия ВКР соответствует цифровой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* / Н.И. Верхогляд/

(подпись студента)

*ВКР (в формате adobe PDF) принята   
для размещения в электронной библиотеке \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*/ С.Н.Антипова/

(подпись ответственного по кафедре за размещение ВКР в ЭБ)



Факультет Информационных технологий

Кафедра Прикладной информатики и мультимедийных технологий

УТВЕРЖДАЮ:

Заведующий кафедрой прикладной информатики и мультимедийных технологий Л.С. Куравский

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021г.

**ЗАДАНИЕ**

**НА ВЫПОЛНЕНИЕ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ (ВКР)**

**Студент** \_Верхогляд Никита Игоревич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Тема ВКР** «\_Программная реализация сбора экспертных оценок с применением метода попарного сравнения \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_»

(утверждена Приказом по университету от «\_\_\_» \_\_\_\_\_\_\_\_\_\_202\_г. № 7–03/1333-у)

**Срок сдачи** **ВКР** «\_\_» \_\_\_\_\_\_\_ 202\_г.

Исходные данные к ВКР

1. Балябин В.А. Заславский А.А. Построение правильных матриц парных сравнений. Результаты вычислительного эксперимента // Экономика и математические методы. – 2018;
2. Давид Г. Метод парных сравнений. – Статистика, 1978;
3. Данн Н. Функциональный Python: курс для начинающих программистов на Python. – Webucator, 2020;
4. Джордж Н. Создание сайта с помощью Django 3. – Австралия, Гамильтон: GNW Independent Publishing, 2020

Перечень подлежащих разработке в ВКР вопросов:

1. \_Изучение темы экспертных оценок; \_
2. \_Изучение метода попарных сравнений; \_
3. \_Анализ вариантов программной реализации сбора оценок; \_
4. \_Проектирование программного продукта для сбора оценок; \_
5. \_Разработка спроектированного программного продукта. \_

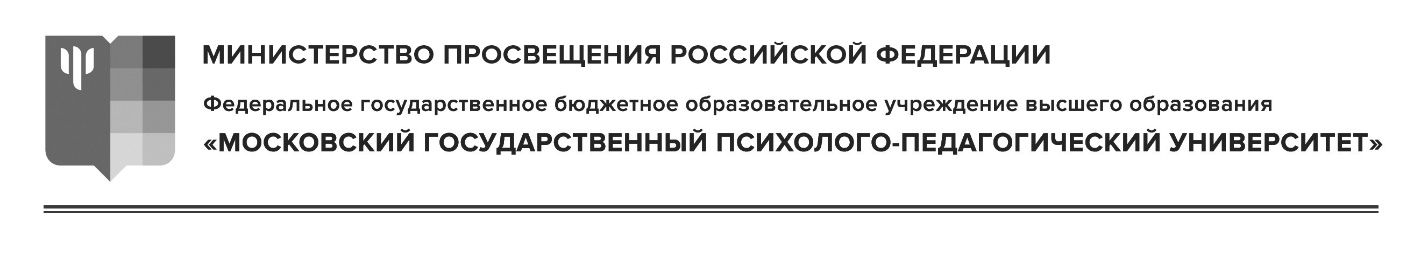
Перечень дополнительного материала, подлежащего к разработке в рамках ВКР \_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_202\_г.

Руководитель Подпись Дата Л.С.Куравский

Задание принял к исполнению Подпись Дата Н.И. Верхогляд



Факультет Информационных технологий

Кафедра Прикладной информатики и мультимедийных технологий

**АННОТАЦИЯ**

**НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ (ВКР)**

Тема: Программная реализация сбора экспертных оценок с применением метода попарного сравнения.

Объем работы: 60 страниц, на которых размещено 8 рисунков, 2 таблицы и 2 формулы. Данная работа состоит из введения, трех глав, заключения и списка литературы.

Ключевые слова: экспертные оценки, парное сравнение, веб-приложение, python, django.

Введении описывается актуальность данной работы, а также поставлена цель работы.

В первой главе «Экспертные оценки» описываются понятие экспертных оценок, процедуры присущие им, методы экспертных оценок, а также их применение.

Во второй главе «Метод попарного сравнения» описываются понятие метода попарных сравнений, его положительные и отрицательные качества, пример его реализации, а также возможные программные реализации сбора экспертных оценок.

В третьей главе «Разработка программного решения» описывается подбор стека технологий для разработки программы, проектирование базы данных, моделирование работы программы, а также описание процесса разработки программы.

В заключении подведен итог проделанной работы, а также возможные варианты улучшения получившейся программы.

Оглавление

[Оглавление 4](#_Toc74613446)

[Введение 5](#_Toc74613447)

[Глава 1. Экспертные оценки 6](#_Toc74613448)

[1.1. Понятия экспертной оценки 6](#_Toc74613449)

[1.2. Классификация методов экспертных оценок 7](#_Toc74613450)

[1.3. Этапы экспертного оценивания 10](#_Toc74613451)

[1.4. Применение экспертных оценок 12](#_Toc74613452)

[Глава 2. Метод попарного сравнения 13](#_Toc74613453)

[2.1. Понятие попарного сравнения 13](#_Toc74613454)

[2.2. Достоинства и недостатки метода попарного сравнения 14](#_Toc74613455)

[2.3. Представление метода попарного сравнения 16](#_Toc74613456)

[2.4. Автоматизация системы попарного сравнения с помощью программных решений 18](#_Toc74613457)

[2.5. Применение метода попарного сравнения 20](#_Toc74613458)

[Глава 3. Разработка программного решения 21](#_Toc74613459)

[3.1. Выбор языка программирования 21](#_Toc74613460)

[3.2. Выбор фреймворка 23](#_Toc74613461)

[3.3. Выбор базы данных 27](#_Toc74613462)

[3.4. Проектирование системы 28](#_Toc74613463)

[3.5. Реализация системы 37](#_Toc74613464)

[Заключение 57](#_Toc74613465)

[Список литературы 58](#_Toc74613466)

Введение

Некоторые задачи, требующие принятия решения, не позволяют дать однозначный ответ. В такой ситуации приходится прибегать к методам экспертных оценок. Метод парного сравнения является одним из них. Его используют для упрощения экспертной оценки, когда сложно сравнивать сразу все варианты. К примеру его используют для принятия решения о составе семантического ядра для сайтов, с целью оптимизации бюджета, так как сравнивать все варианты и сразу является неудобным подходом и влечет за собой ошибки из-за большого количества данных.

Проблема метода парного сравнения является в том, что количество пар для оценок получается большим, даже для относительного небольшого списков альтернатив. Что в свою очередь вызывает проблему отсутствия специализированного программного обеспечения для автоматизации процесса сбора экспертных оценок.

Целью данной квалификационной работы является разработка программного обеспечения для автоматизации сбора экспертных оценок метода парного сравнения для объектов, которые должны восприниматься как одно целое, то есть чьи характеристики не могут быть разделены и имеют значимость лишь в сочетании.

Глава 1. Экспертные оценки

* 1. Понятия экспертной оценки

Для принятия любых решений, приходится проводить анализ вариантов, между которыми происходит выбор. Когда варианты выбора обладают количественными качествами или поддаются математической формализации, принятие решения является более простой и однозначной задачей, чем в случае отсутствия количественных качеств или из-за новизны проблемы, недостаточности имеющейся информации. Для отдельно взятого индивида ошибка при принятии решения в подобной ситуации скорее всего не приведет к серьезным проблемам. Но в случае каких-либо организаций подобные ошибки могут привести к репутационным и материальным потерям, что в худшем варианте может привести к упразднению организации. В такой ситуации верным решение будет прибегнуть к использованию методов экспертных оценок.

К примеру, методы экспертных оценок применяются для сравнения каких-то параметров объектов находящихся в одном классе, одной категории: комфорт самолета, вкус еды, дизайн, качество обслуживания.

Экспертная оценка – это метод поиска и рекомендации, решения, которые выдвигает один человек или группа лиц, наделенная экспертностью в требуемой области, на основании собственного опыта, мнения, а также проведенной экспертизы. Экспертная оценка является основой для принятия решения или выбора.

Эксперт – это компетентное лицо, специалист, имеющее практический опыт в необходимой области, участвующий в требуемом исследовании в качестве источника для получения достоверной информации. Логично, что в качестве экспертов необходимо подбирать тех людей, чей опыт и суждения, будут способствовать принятию адекватного, верного решения. При выборе экспертов, необходимо учитывать возможную опасность личной заинтересованности, предвзятости, эксперта в принятии того или иного решения. Данная проблема может стать серьезной проблемой на пути к получению объективного и верного результата.

Сложность подхода экспертных оценок заключается в том, что от оценки эксперта, группы экспертов, может зависит не только конкретная ситуация, но и в целом состояние организации, которая прибегла к услугам эксперта. Соответственно поиск и подбор экспертов является многогранной и сложной задачей от результата которой напрямую зависит эффективность используемых методов и как следствие качество результата и принимаемого на его основании решения. Поэтому при выборе специалиста к нему выставляется ряд качеств, которыми он должен обладать.

Основные требования при выборе эксперта:

* Компетентность, в требуемой сфере;
* Знания в смежных областях, не относящиеся напрямую к изучаемому вопросу;
* Научными и академически достижениями;
* Опытом практической работы в данной сфере;
* Объективностью и непредвзятостью при проведении исследования и формировании оценки;
* В случаях, когда необходимо, умение мыслить неординарно.
  1. Классификация методов экспертных оценок

Существуют две группы получения экспертных оценок, основанные на составе группы экспертов:

Индивидуальные оценки – основывается на получении независимо друг от друга сформированных оценок отдельных экспертов или же на мнении лидера группы экспертов. Данный способ предполагает предварительный сбор информации от экспертов;  
Методы получения индивидуального мнения:

* Метод интервью – предполагает беседу с экспертом в формате вопрос-ответ;
* Метод доклада – предполагает самостоятельную работу эксперта над анализом, с представлением результата в виде аналитической записки;
* Метод Дельфи – разработка программы для последовательных индивидуальных многоуровневых опросов.

Коллективные оценки – основывается на использовании коллективного мнения экспертов. Результат работы экспертной группы является коллективное мнение группы экспертов, которое было получено в результате прихода всей группы к консенсусу, принятие решения без проведения общего голосования, на основе общего согласия. Данный способ предполагает результата в следствии совместного обсуждения вопроса группой.  
Методы получения коллективной работы группы экспертов:

* Деловая игра – заключается в моделировании функционирования интересующей системы управления при выполнении действий, нацеленных на достижение поставленных целей;
* Мозговой штурм – выступление экспертов с запретом на критику предложений других экспертов;
* Оценка комиссией – основан на составлении специальных комиссий. Заключается в обсуждении группой экспертов заданной проблемы с целью согласования и выработки единого мнения;
* Суд – является аналогией ведения судебного процесса, где в качестве подсудимых выступают предполагаемые варианты решения проблемы;
* Метод 635 – один из вариантов мозгового штурма. Первая цифра в названии обозначает количество участников, вторая цифра обозначает количество идей которые должен предложить каждый участник, последняя цифра показывает время за которое участники должны выдвинуть свои предположения для решения проблемы.

Экспертное оценивание основывается на нескольких методах выработки как индивидуальных, так и коллективных мнений. Выбор наиболее подходящего метода исследования зависит от поставленных целей, в следствии чего выбираются лишь после общения заказчика и эксперта. Наиболее популярными методами являются:

* Метод ассоциаций – основывается на изучении объектов имеющих схожие свойства;
* Метод парных сравнений – основывается на предоставлении экспертам альтернативных вариантов, которые необходимо сопоставить с целью определить и изучить наиболее предпочтительные варианты решения поставленной проблемы;
* Метод векторов предпочтений – основан на анализе всей выборке альтернативных вариантов, из которых эксперту необходимо выбрать наиболее подходящие варианты;
* Метод фокальных объектов – основан на присваивании к исследуемому объекту свойств других случайно выбранных объектов-аналогов;
* Индивидуальный экспертный опрос – заключается в опросе в формате интервью или анализа экспертных оценок. Зачастую является беседой между экспертом и ответственным лицом организации, заказчиком. В процессе беседы заказчик, ставит вопросы перед экспертом, которые предположительно являются значимыми для достижения необходимых целей. В результате опроса эксперт заполняет форму, разработанную заказчиком. На основании данной формы производится анализ целевой ситуации и выявляются пути решения поставленной задачи. При необходимости свое заключение выносит в отдельный документ;
* Метод средней точки – в процессе оценки формулируются два противоположных варианта решения задачи: наиболее и наименее предпочтительные. Предполагается что между ними находится некая «средняя точка», альтернатива представленным вариантам, который необходимо найти эксперту, учитывая особенности первого и второго варианта.
  1. Этапы экспертного оценивания

Все методы экспертных оценок состоят из нескольких стадий исследования, в которых получается необходимая информация касаемая изучаемой проблемы. Результат достигается с помощью следующих шагов:

* Постановка цели исследования. Началом экспертного анализа является определение цели, постановка проблемы, формулирование советующих вопросов для экспертов;
* Выбор формата, проведения исследования. Выделяют две формы исследований: очную и заочную. Очный метод считается более достоверным в следствии того, что позволяет эксперту на месте оценить и адаптировать необходимые тесты. Заочный же формат является менее финансово затратным;
* Подготовка информационных материалов. Экспертам необходимо рассмотреть представленную проблему и на основе информации вынести свои суждения. Для проведения данной процедуры экспертам необходимо описание данной проблемы и информацию по ней. Чем более разнообразна предоставляемая информация, тем больше материалов для аналитики экспертов. Стоит избегать упоминания разработчиков задачи, выделять какие-либо варианты решений, отношения к данной проблеме, а также желаемый результат. Чем более качественная информация будет предоставлена экспертам, тем более вероятно нахождение верного решения поставленной задачи.
* Подбор экспертов. Эксперты должны соответствовать ряду критериев: опыт, квалифицированность. Необходимо избегать выбора эксперта имеющего личную заинтересованность в вопросе решения поставленной задачи. Для подбора эксперта можно использовать методы Шора, суть которых заключается в том, что наиболее авторитетный специалист рекомендует ряд других, они же в свою очередь рекомендуют третьих, и так до тех пор пока не будет собран необходимы коллектив.
* Проведение экспертизы. Исследование не может основываться лишь на предоставленных информационных материалах. Целю этапа является построения экспертами собственных суждений, гипотез, которые должны быть подтверждены практикой. Для этого устанавливается четкие временные рамки, оптимальные для решения необходимой задачи. Необходимо препятствовать давлению авторитета на экспертов.
* Обработка экспертных оценок. Это финальный этап экспертной оценки. На данной этапе происходит анализ и обработка полученных данных в ходе проведения экспертизы. После анализа происходит отбор наиболее удачных вариантов решений, после чего происходит их интеграция в систему.
  1. Применение экспертных оценок

Экспертные оценки применяются на любой стадии разработки, оптимизации, оценке бизнес-процессов. Чаще всего к данным методам прибегают, когда нельзя определить более верный вариант пути, оперируя точными расчетами.

Экспертная оценка помогает найти и определить проблему, а также найти пути ее решения. Своевременно проведенная экспертиза может помочь оперативно решить возникшую проблему, а также избежать каких-либо потерь.

Глава 2. Метод попарного сравнения

2.1. Понятие попарного сравнения

В управленческой практике часто встречаются ситуация, где необходимо принять решение, которое заключается в совершенно противоположных вариантах, альтернативах. По принципу да или нет, одно или другое. Примером таких ситуаций могут служить:

* Прием человека на работу;
* Переход к новому алгоритму работы;
* Выбор целевых запросов для семантического ядра сайта.

Метод попарного (бинарного) сравнения является одним из методов проведения экспертной оценки. Он стоит на ряду с другими методами: ассоциаций, векторов предпочтений, фокальных объектов, средней точки, а также индивидуальным экспертным опросом.

Метод попарного сравнения – используется для принятия выбора и оценки принимаемых решений. Целью данного метода является определение наилучшего варианта, с помощью сравнения варианта с каждой предоставленной альтернативой.

Задумка метода является довольно простой – берется список объектов, которые необходимо оценить, каждый объект сравнивается с каждым (попарно) альтернативным вариантом, от того и образуется название.

При использовании данного метода экспертных оценок задача эксперта заключается в сравнении пары объектов и определении более значимый, верный вариант. Данный метод был создан на основе позиции, что при принятии решения или ранжировании вариантов, гораздо проще провести качественное сравнение опираясь на мнение экспертов, чем разрабатывать и устанавливать количественные оценки для каждого объекта.

Данный метод позволяет решать множество задач, включая разработку оценочной шкалы, построение количественных оценок, для конкретного ряда объектов.

2.2. Достоинства и недостатки метода попарного сравнения

Метод бинарного сравнения является довольно простым для сбора данных, а также для интерпретации результатов. Он позволяет эксперту абстрагироваться от всех имеющихся вариантов и сосредоточиться лишь на паре, что уменьшает время на принятие решения, а также качество результатов. Значимым преимуществом данного подхода в сравнении с альтернативными методами экспертных оценок, как раз является возможность выносить вердикт по очереди для пар альтернатив, а не проводить анализ и ранжирования всего списка альтернатив, которых может насчитываться десятки, сотни или даже тысячи.

Положительные стороны метода попарного сравнения:

* Допускается неравномерно изменяющиеся важность разных свойств при сравнении разных пар.
* Эксперт концентрирует внимание лишь на паре объектов, а не на всем списка альтернатив сразу. Что облегчает работу и принятие решений, а также сокращает время на оценку. В следствии облегчения работы а также увеличении концентрации полученные результаты являются более качественными.
* Возможность получить больше данных. Большее число сравнений каждого объекта с другими. Благодаря этому повышается точность оценок, а также появляется возможность изучить большее количество сторон объекта, нежели при использовании других методов. На основе полученных данных возможно провести анализ взаимосвязей между качествами всех альтернативных объектов.
* Возможность получить помимо средней оценки, каждого эксперта, также и дисперсию данной оценки. Что позволяет провести математический анализ.
* Метод попарного сравнения является одним из самых простых из существующих классификационных тестов, так как предоставляет сравнение лишь двух объектов.
* Метод попарных сравнений дает возможность провести строгий, статистически обоснованный анализ согласованности мнений экспертов, выяснить, были ли полученные оценки случайными или имеется какая-нибудь закономерность.

Минусы метода попарного сравнения:

* Необходимость проводить огромное количество попарных сравнений, при проведении оценок в большом количестве объектов. Уже при сравнении 60 объектов, эксперту необходимо провести 1770 сравнений.

Формула 2.1 Расчет количества парных сравнений

Формула для расчета количества сравнений:

Где – количество сравнений, – количество альтернатив.

* Отсутствие готовых программных решений, для проведения тестирования.

2.3. Представление метода попарного сравнения

Метод попарного сравнения удобно использовать при небольшом количестве альтернативных вариантов, а также сравнивать их лишь по одному параметру. Подобное свойство метода возникает из-за того, что результат исследования представляется в виде сводной таблицы размерностью n\*n (количество альтернатив \* количество альтернатив). При наличии пяти, десяти, или даже пятнадцати альтернатив, таблица результатов все еще имеет адекватный вид и размер. Но в случае большего количества альтернатив возникает проблема размерности, да и в целом становится легко запутаться в полях.

К примеру у нас есть список из пяти целевых запросов для семантического ядра сайта, среди которых необходимо распределить бюджет на их продвижение в поисковых системах:

1. Шкаф купе электросталь;
2. Мебель электросталь Ногинск;
3. Шкафы купе на заказ электросталь Ногинск;
4. Электросталь шкафы;
5. Электросталь шкаф купе.

В результате составления таблицы для экспертной оценки для распределения бюджета будет получена следующая таблица:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Объекты | 1. | 2. | 3. | 4. | 5. | Результат |
| 1. Шкаф купе электросталь |  |  |  |  |  |  |
| 2. Мебель электросталь Ногинск |  |  |  |  |  |  |
| 3. Шкафы купе на заказ электросталь Ногинск |  |  |  |  |  |  |
| 4. Электросталь шкафы |  |  |  |  |  |  |
| 5. Электросталь шкаф купе |  |  |  |  |  |  |

Таблица 2.1. Пустая таблица для попарного сравнения

После проведения экспертизы, таблица заполняется нулями и единицами. Единицы ставятся напротив наилучших вариантов. Диагональ матрицы оставляется пустой либо заполняется нулями.

Формула 2.2. Подсчет результатов попарного сравнения

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Объекты | 1. | 2. | 3. | 4. | 5. | Результат |
| 1. Шкаф купе электросталь |  | -1 | -1 | -1 | 1 | -2 |
| 2. Мебель электросталь Ногинск | 1 |  | 1 | 1 | 1 | 4 |
| 3. Шкафы купе на заказ электросталь Ногинск | 1 | -1 |  | 0 | 1 | 1 |
| 4. Электросталь шкафы | 1 | -1 | 0 |  | 1 | 1 |
| 5. Электросталь шкаф купе | -1 | -1 | -1 | -1 |  | -4 |

Таблица 2.2. Заполненная таблица для попарного сравнения

Результат является сумма всех значений строки:

После подсчета оценок можно провести сортировку альтернатив по их важности. В результате мы получим отсортированный список:

1. Мебель электросталь Ногинск (4);
2. Электросталь шкафы (1);
3. Шкафы купе на заказ электросталь Ногинск (1);
4. Шкаф купе электросталь (-2);
5. Электросталь шкаф купе (-4).

На основании полученного списка можно предположить, владельцу сайта стоит сделать упор на продвижение целевого запроса «Мебель электросталь Ногинск», который получил наибольшую оценку.

2.4. Автоматизация системы попарного сравнения с помощью программных решений

Готовые решения:

Microsoft Excel, Google Таблицы – возможно сразу создать матрицу, сводную таблицу, так же возможно провести анализ полученных данных встроенными средствами программ. Хорошо подходят для проведения сравнений небольшого количества элементов, ведь при увеличении количества элементов, увеличивается и сводная таблица что усложняет ее заполнение и порождает вероятность заполнения неверного поля. Так же появляется проблема, если есть необходимость собрать данные нескольких опрошенных. Респондентам придется вручную отправлять заполненные файлы, для дальнейшей обработки. Так же в данных программах отсутствует возможность изменения визуальной составляющей.

Google Формы – программное обеспечение разработанное специально для проведения опросов и анкетирования. Предоставляет возможность элементы формы. Результаты опросов сразу показываются в диаграммах. Присутствует возможность получить данные в формате таблицы, правда в данном контексте подобную таблицу придется обрабатывать для удобного анализа. Пользователю проходящему опрос нет необходимости что-то устанавливать на свое устройство. Все данные о пройденных опросах сохранятся на Google Диске, для дальнейшей обработки, все равно придется скачать данные. Все пары вариантов приходится прописывать вручную, а для опрашиваемого это начинает выглядеть как бесконечное полотно однообразных пар фраз, что при большом количестве таковых приводит к тому, что большинство людей просто бросят заполнение опросника; в такой ситуации даже уже заполненные данные будут утеряны, так как данные на диск отправляются лишь после того, как пользователь заполнит всю анкету и подтвердит это.

Решения требующие разработки:

Нативные приложения – нативными приложениями называют то программное обеспечение, которое было разработано под определённую операционную систему или устройство. Плюсом подобного приложения является то, что оно хорошо оптимизировано под конкретную платформу. Данный подход позволяет полностью настроить интерфейс опросника, можно показывать пользователю по одной паре запросов, что решает проблему с ошибками при заполнении таблиц. Позволяет сохранять данные на любом из этапов анкетирования. При наличии подключения к сети интернет, позволяет отправлять полученные данные в удобном для обработки виде. Сложность данного подхода заключается в том, что для каждой операционной системы придется разрабатывать отдельное приложение, это заметно увеличивает затраты на разработку, в настоящее время придется разработать как минимум три приложения, а также что бы собирать из них данные нужно будет разработать серверную часть, которая принимала бы данные. Так же после разработки каждого приложения придется зарегистрировать его в магазине приложений каждой операционной системы и дождаться одобрения администрации магазина.

Сайт и веб-приложение – так же как и нативные приложения, позволяет настроить дизайн интерфейса по желанию. Позволяет сохранять данные как только пользователь делает выбор. Доступно для всех устройств имеющих подключение к интернету. Данные сохраняются в необходимом формате. Возможно обрабатывать данные сразу при получении, а также демонстрировать полученные данные пользователям. Как один из вариантов реализации, можно разработать программный интерфейс приложения (API), что позволит совместить данный подход с нативными приложениями.

Данный подход позволяет как получать так и загружать данные в удобном для нас формате. Так же нам представляется возможность не пугать конечного пользователя огромным списком пар целевых запросов, а показывать ему лишь одну пару в один момент времени. Человеку проходящему исследование нет необходимости сравнивать все пары, он может в любой момент начать, закончить, а также продолжить прохождение опроса. В результате данный подход является оптимальным, так как он доступен для всех устройств, а также может работать с удобным для нас форматом данных.

Разработка подобного программного обеспечения позволит упростить процесс сбора семантического ядра для сайтов, а также собрать данные об экспертных оценках, используя которые можно будет разработать оптимальный математический аппарат, который позволил бы автоматизировать подобные задачи.

2.5. Применение метода попарного сравнения

Метод попарных сравнений является довольно простым по сравнению с альтернативными методами экспертных оценок. Он позволяет не проводит анализ сразу над всеми альтернативами. Данный метод хорошо себя показывает при небольшом списке альтернатив. В случае большого списка начинается проблема с масштабируемостью.

Глава 3. Разработка программного решения

3.1. Выбор языка программирования

Вопрос, автоматизации сбора оценок для исследования с применением метода парных сравнений, является довольно важным. Для решения данного вопроса была поставлена цель разработать специализированный программный продукт. В качестве формата данного приложения было выбрано веб-приложение, сайт. Решение о разработке приложения именно в данном формате было принято основываясь на том, что данный формат позволяет избежать необходимости разрабатывать несколько нативных приложений, для каждой операционной системы и заниматься сопутствующими необходимыми требованиями, размещение приложения в магазинах приложений каждой системы. При всем этом, данный формат практически не уступает по функциональности. Он точно покрывает все требования к разрабатываемой системе, так как ей нет необходимости иметь доступ к датчикам устройства.

Первой задачей в процессе разработки является определить инструментарий для разработки приложения. Первым таким инструментом является язык программирования.

Основные языки программирования для веб-разработки:

* Java – строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный в 1998 году. Код приложения написанный на Java транслируется в специальный байт-код, который может исполняться на любом устройстве, на котором установлена виртуальная Java.
* C# – высокоуровневый язык программирования, разработан в 2000 году. Изначально был разработан специально для работы с фреймворком .NET (точка нет), который предназначается для разработки веб-сайтов.
* PHP – гипертекстовый препроцессор, скриптовый язык общего назначения, был разработан в 1995 году. Интенсивно применяется для разработки динамических веб-сайтов, а также веб-приложений.
* Python – высокоуровневый язык программирования общего назначения с динамической строгой типизацией. Особенностями данного языка является простота восприятия синтаксиса, автоматизация многих процессов, что в свою очередь улучшает качество кода, а также увеличивает скорость разработки приложений на данном языке. Оригинальная версия языка была выпущена в 1991 году, но на данный момент она не поддерживается разработчиком, ее заменила 3 версия Python, она была выпущена в 2008 году.

Так как приложение планируется быть небольшим, то скоростью работы кода и объемом итогового приложения можно пожертвовать в угоду большего удобства и скорости разработки. Основываясь на этом тезисе можно откинуть Java и C#, из-за сложности их синтаксиса, в сравнении с другими двумя языками.

Остался выбор между двумя интерпретируемыми языками программирования: PHP и Python. Оба эти языка обладают в некоторыми схожими качествами, к примеру они оба имеют хорошую документацию, в большинстве случаев. Также у них у обоих простой, понятный и удобный синтаксис. Также оба языка являются одними из самых популярных, что гарантирует большое сообщество разработчиков, и соответственно большое количество материалов по решению задач и исправлению проблем. Изъяны PHP заключаются в том, что он страдает от несистемного синтаксиса. Это значит, что почти идентичные функции могут выполнять совершенно разные задачи. Также PHP наследует некоторые наработки от таких языков как Java и C, в свою очередь это привело к тому, что при разработке на данном языке используются разные правила оформления кода. Эти два пункта могут вводить в заблуждение и порождать простые и глупые ошибки. Так же PHP имеет не самую удобную и понятную систему отладки кода.

В итоге в качестве языка будет выбран Python. Он обладает удобным, простым и понятным синтаксисом, хорошей документацией, в большинстве случаев, большим количеством фреймворков, готовых решений, обширным сообществом разработчиков, единым стилем написания кода.

3.2. Выбор фреймворка

После выбора языка появляются два варианта дальнейших событий на пути к готовому приложению. Первый – начать разработку веб-сайта используя чистый язык программирования, не используя дополнительных компонентов. Второй – найти дополнительные инструменты, которые могут упростить процесс разработки приложения. В процессе разработки веб-приложения для автоматизации попарных оценок, было решено прибегнуть к использованию дополнительных инструментов, веб-фреймворков.

Для языка программирования Python было разработано много фреймворков, чья цель упростить разработку веб-приложений. Среди всех инструментов выделяются два самых популярных и распространенных: Flask и Django.

Flask – фреймворк разработанный на языке программирования Python для создания веб-сайтов и веб-приложений. Он относится к группе минималистичных каркасов для веб-приложений, то есть к группе микрофреймворков. Идея данной группы фреймворков заключается в том, что они предоставляют лишь минимальный набор инструментов для разработки веб-сайтов. Выпуск первой версии состоялся в 2010 году.

Django – полнофункциональный фреймворк написанный на языке программирования Python. Содержит инструменты для упрощения реализации разнообразных функций, начиная от собственных инструментов для создания виртуальной объектной базы данных, до инструментов для разработки программного интерфейса приложений.

Разница между Flask и Django:

* Объект Request – Flask использует локальные потоки, а Django передает запрос там, где это нужно.

Объект запроса существует у обоих фреймворков, но он отличается как реализацией, так и использованием. В случае с Django, если есть необходимость получить доступ к данным запроса, то необходимо передать объект в качестве переменной в необходимую область. Положительная сторона такого подхода заключается в том, что бизнес-логика приложения отделена от контекста запроса. В свою очередь Flask локальную переменную потока для хранения запроса. Это значит, что для получения доступа к данным запроса, необходимо лишь импортировать объект request из модуля flask, после этого появляется возможность использовать его в любом месте кода программы. С данным подходом необходимо быть внимательным, так как возможно обратиться к объекту запроса вне контекста самого запроса, что вероятнее всего приведет к непредвиденным ошибкам.

* Формы — Django доступен со встроенными формами, которые интегрируются с ORM и административной панелью сайта. Flask не поддерживает формы по умолчанию, но вы можете использовать библиотеку WTForms, чтобы заполнить этот пробел.

Django предоставляет возможность определять формы в виде классов в языке программирования Python, это позволяет обрабатывать данные в единой форме. Так же упрощают работу с механизмами безопасности и проверки данных. Если форма создается для обновления модели (таблицы) базы данных, то Django предоставляет возможность создавать объекты форм основываясь на уже описанных моделях для базы данных. В свою очередь Flask, по умолчанию не предоставляет инструментарий для работы с формами. Для исправления данной ситуации можно использовать библиотеку WTForms.

* Базы данных – Django доступен со встроенной ORM (объектно-реляционное отображение) и системой миграции, для контроля и управления базой данных. Flask не может этим похвастаться, однако есть инструменты, такие как библиотека SQLAlchemy, которые предоставляют аналогичный функционал (или даже больше).

По умолчанию Django использует собственное реляционное представление объектов для баз данных. Данная технология позволяет описать схему бузы данных в виде классов языка программирования python, а запросы через данные классы к базе, автоматически генерируют и исполняют запросы на языке SQL. Еще одним плюсом является автоматическая генерация миграций базы данных, что упрощает слежение за актуальность формата данных. Как было уже сказано, использование встроенной системы ORM позволяет создавать формы в упрощённом формате. Flask же придерживается позиции свободы выбора, соответственно основываясь на этом он не навязывает никаких инструментов для работы с базами данных. Разработчики могут использовать удобные им инструменты, но задача их интеграции полностью ложится на плечи программиста.

* Аутентификация и привилегии пользователям – Django предоставляет приложение аутентификации, которое предоставляет реализацию по умолчанию для пользовательского управления и привилегий. Flask предоставляет безопасные cookie в качестве инструмента вашей собственной реализации.

Для аутентификации Django предоставляет подключаемое приложение, которое полностью интегрируется с проектом и предоставляет уже готовые решения в виде моделей пользователей, системы доступа и поддержки сессий. При этом фреймворк не накладывает никаких ограничений, при необходимости разработчик может переписать элемент системы требующий коррекции. Со стороны Flask снова свобода выбора и ответственность за качество выбранного инструмента, а также его интеграцию.

* Панель администратора – Django включает в себя полностью интегрированный админ-интерфейс для управления данными приложения. Flask не имеет таких функций, но Flask-Admin — очень популярное расширение, которое можно использовать для создания аналогичного административного инструмента.

Фреймворк Django по умолчанию предоставляет удобный инструмент для управлений информацией из вашей базы данных. Как и в предыдущих пунктах, Flask не предоставляет готового решения. Но существует расширение Flask-Admin, которое позволяет реализовать панель администратора схожую с встроенной в Django. Но для интеграции данного расширения в проект, также необходимо подключить и настроить систему аутентификации.

В результате выбор встает между фреймворком в котором уже интегрирована большая часть инструментов, которые могут понадобиться при разработке веб приложения и фреймворком, который предоставляет свободу выбора, но возлагает ответственность за интеграцию инструментов на разработчика. Для разработки данного приложения будет выбран фреймворк Django, так как его встроенный инструмент для работы с базами данных заметно упрощает и ускоряет разработку приложения, а для работы с объектами запросов нет необходимости следить за контекстом самого запроса, из-за удобного формата настройки URL-адресов, и конечно из-за официальной документации, часть которой переведена на русский язык.

3.3. Выбор базы данных

База данных является основным источником и местом хранения информации в проекте. Естественно что база данных должна соответствовать потребностям разрабатываемого приложения.

Результаты попарного сравнения изображаются в виде сводной таблицы, квадратной матрицы. Таблицы в программировании зачастую представляют собой массив из массивов. Так как для разработки приложения был выбран скриптовый язык общего назначения, Python, стоит отметить, что он не обладает таким типом данных как массивы. В данном языке программирования массивы заменены списками. Таким образом к выбираемой базе данных создается требование, а именно – возможность работать с массивами в их исходном виде.

Конечно можно массивы форматировать в строки и записывать их в базу данных с нефиксированной длинной строки. Так же возможно хранить таблицы в отдельных файлах, по типу CSV, а в базе хранить ссылку на них. Но это не так уж и удобно, если учесть, что существуют не реляционные базы данных, которые позволяют хранить данные практически в любом их виде, а также реляционные базы данных, которые могут работать с данными в формате объектов и массивов.

Не реляционный базы данных хоть и позволяют хранить данные в любом формате, они не всегда удобны для использования. Зачастую их функционал является излишним для не очень больших проектов. На практике, если используемую информацию при работе приложения можно хранить в реляционных базах данных, то выбирают именно их, из-за их устоявшейся позиции и большого количества документации и готовых решений.

В данном приложении будет использоваться реляционная база данных PostgreSQL. Выбор был остановлен на ней, так как она предоставляет возможность работы с массивами в их исходном виде. Имеет инструменты для легкой интеграции в Django-проекты. Входит в группу свободного программного обеспечения.

3.4. Проектирование системы

После выбора стека технологий для разработки приложения, но перед переходом к написанию кода программы, желательно описать схему функционирования приложения и его взаимодействие с пользователями и администраторами. Помимо описания самого приложения, также необходимо разработать схему базы данных. Для этого используются разного рода диаграммы и блок-схемы. Описание приложения до перехода к написанию кода, позволяет примерно представить как должно вести себя приложение, а так же как лучше можно спроектировать его архитектуру и на какие части его лучше разделить.

Для описания алгоритма работы разрабатываемого приложения будет использоваться нотация и модель бизнес-процессов (BPMN - Business Process Model and Notation). На данной диаграмме показывается последовательность действий, алгоритм, которые происходят во время функционирования системы.

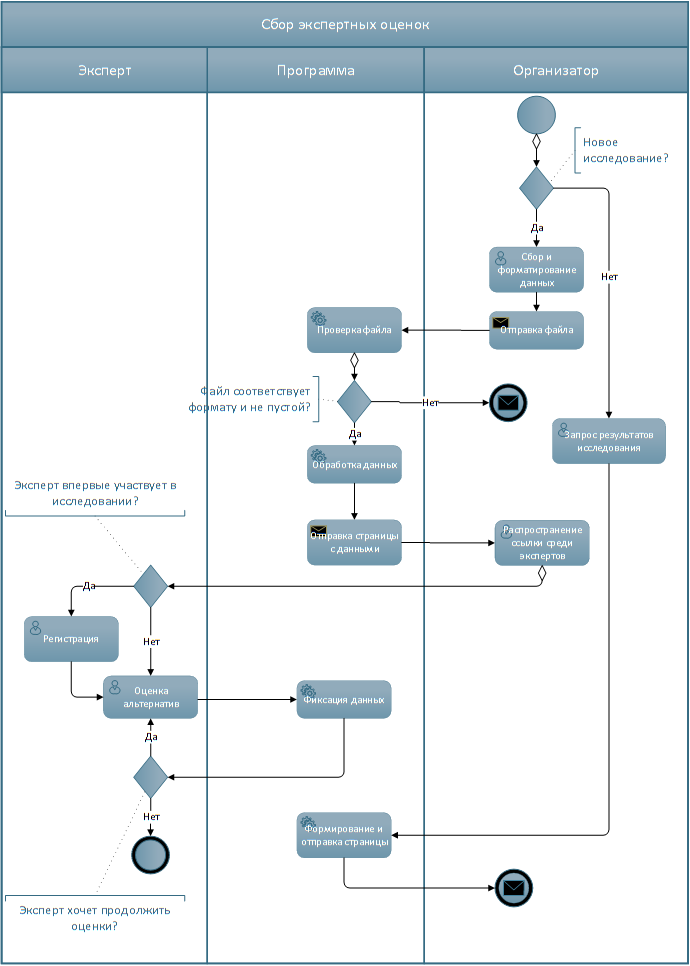


Рисунок 3.1. Диаграмма BPMN

Начальным узлом диаграммы деятельности, для веб-приложения для автоматизации сбора оценок методом попарного сравнения, можно считать чью-то необходимость провести экспертную оценку чего-либо методом попарного сравнения.

Сразу же после начального узла, на диаграмме, находится узел принятия решения. В данном случае задается вопрос: «Организатор пришел получить результат экспертных оценок?»

1. В случае отрицательного ответа, начинается процесс загрузки и обработки данных.
2. После сбора данных и их форматирования. Организатор отправляет их на сервер приложения.
3. В свою очередь сервер читает присланные данные и сверяет их с необходимым форматом. В случае если по какой-то причине данные не соответствуют заданной структуре, это доносится до организатора. После чего цикл деятельности приложения завершается.
4. В случае если данные были присланы в правильном формате, они обрабатываются и записываются в базу данных.
5. После сохранения присланных данных организатору отправляется страница с полученной информацией. А также ссылка для просмотра результатов исследования и ссылка для проведения самого исследования.
6. Далее задача организатора донести ссылку для проведения экспертизы до самих экспертов.
7. В случае если эксперт впервые учувствует в каком-либо исследовании он проходит регистрацию, если же он уже учувствовал, либо прерывал процесс оценки, он сразу направляется на страницу сбора оценок.
8. Результат оценки отправляется на сервер и фиксируется.
9. После этого эксперт может продолжить выносить оценки до тех пор пока не сравнит все доступные варианты, либо прервать процесс сравнивания, после чего цикл деятельности будет закончен.

В случае положительного ответа организатору отправляется страница с данными, которые удалось собрать. После чего логический цикл деятельности приложения заканчивается.

В результате, программу можно разделить на две части по субъекту, с которым связана деятельность. То есть одна часть программы работает с организатором экспертизы, обрабатывает данные, а другая с экспертами, собирает оценки.

Как уже говорилось выше, одна из основных составляющих проекта – это база данных. Заблаговременное проектирование схемы базы данных зачастую является обязательной стадией разработки любого программного продукта, который использует базу данных. Для описания схемы базы данных будет использоваться схема «сущность-связь» или ER-диаграмма.

ER-модель – это модель данных позволяющая описывать как различные объекты, внутри системы, связанны между собой. ER-модель чаще всего применяют для высокоуровневого проектирования реляционных баз данных. Она помогает определить ключевые сущности и обозначить связи, которые могут образовываться между этими сущностями. Во время проектирования баз данных происходит преобразование схемы, созданной на основе ER-модели, в конкретную схему базы данных.



Рисунок 3.2. Схема базы данных

База данных для данного приложения содержит четыре таблицы: таблица пользователей (Users), исследований (Comparisons), объектов для исследования (Objects), данных об оценках пользователей в конкретном исследовании (ComparisonsUsers).

Таблица пользователей (Users) содержит в себе информацию о экспертах, которые принимали участие в оценках. Данная таблица связана с таблицей с результатами исследований (ComparisonsUsers), отношением один ко многим. Таблица содержит следующие поля:

1. Идентификатор пользователя (UUID), является первичным ключом, используется программой, для того, чтобы узнать пользователя и не предлагать ему оценивать те пары, которые он уже оценил. Идентификатор также хранится на устройстве участника исследования, именно поэтому он является не просто порядковым номером, как в других таблицах;
2. Фамилия имя отчество (Name), имеет ограничение на количество символов;
3. Адрес электронной почты (Email) – предоставляется организатору для связи с экспертом в случае необходимости, также как и имя имеет ограничение по количеству символом, проверяется на соответствие формату адреса электронной почты;
4. Время регистрации.

Таблица исследований (Comparisons) содержит в себе информацию о каждом зарегистрированном исследовании. Данная таблица связана с двумя таблицами: с таблицей результатов (ComparisonsUsers), отношением один ко многим, и таблицей исследуемых альтернатив (Objects). Таблица содержит поля:

1. Порядковый номер исследования (ID), является первичным ключом, используется для поиска и сортировки;
2. Название исследования (Title), имеет ограничение на количество символов в строке. В целом организатор может обойтись заполнением лишь одного этого поля;
3. Описание исследования (Description), не имеет ограничений на количество символов. Заполняется по желанию организатора исследования. Предполагается, что данное поле будет использовано для донесения справочной информации до экспертов, так как данный текст всегда находится перед ними, на странице сбора оценок.
4. Названия параметров объектов исследования (ParameterTitles), данное поле является массивом состоящим из текстовых элементов. В случае если объекты исследований имеют какие-то дополнительные параметры, то их названия данных параметров хранятся в этом поле;
5. Адрес страницы с данными об исследовании (URL), а также собранными результатами. Адрес является фиксированной строкой из набора случайных заглавных и прописных букв латинского алфавита, и цифр. Подобный формат используется для того, чтобы усложнить доступ к результатам исследований случайных людей;
6. Дата загрузки файла исследований (UploadDate), она же дата начала проведения исследований. Имеет встроенный в базу данных формат даты и времени. Фиксируется дата часового пояса сервера, а не организатора исследования;
7. Адрес загруженного файла (File), является фиксированной строкой. Используется для хранения пути к файлу, с объектами и их параметрами, который был загружен при создании исследования. Является обязательным полем, так как данный файл должен содержать список альтернатив и их свойств, которые предстоит сравнивать экспертам;
8. Результат (Result), является матрицей, то есть состоит из массивов с массивами, содержащими цифровые значения. База ограничивает такие поля тем, что число элементов в каждой размерности должно быть одинаковым. Данная матрица хранит в себе данные общего рейтинга каждой пары альтернатив, который является суммой оценок всех экспертов.

Таблица альтернатив для сбора оценок (Objects). Данная таблица хранит всю информацию о множестве объектов пары которых будут предоставляться экспертам для их сравнения. Данная таблица связана с таблицей исследований (Comparisons) связью многие-к-одному. Таблица содержит следующие поля:

1. Общий порядковый номер альтернативы (ID), является первичным ключом;
2. Порядковый номер в исследовании (RawID), используется для сортировки внутри исследования. Содержит целочисленные данные;
3. Название альтернативы (Title), является строкой неограниченной длинны;
4. Значения параметров альтернативы (Parameters), содержит одномерный массив строк, каждый элемент массива является отдельным параметром. Данное поле необязательно для заполнения;
5. Количество сравнений (NumberComparisons), содержит целочисленное значений, которое обозначает сколько раз данная альтернатива была оценена экспертами. Используется для более равномерного сравнений альтернатив, в случае когда некоторые эксперты по какой-либо причине не завершили сравнения;
6. Номер сравнения к которому относится данный объект (ComparisonsID), содержит идентификатор сравнения, данное поле отвечает за связь с сравнениями многие-к-одному.

Таблица результатов сравнений, связь конкретных экспертов с конкретными исследованиями (ComparisonsUsers). Данная таблица содержит данные о результатах оценок конкретных экспертов в конкретном исследовании. Она является связующей таблицей между данными о исследованиях и экспертах, а также реализует отношение многие-ко-многим. Она содержит следующие поля:

1. Порядковый номер результата (ID);
2. Дата начала сбора оценок (Start), содержит дату, когда конкретный эксперт впервые посетил страницу с альтернативами для сравнений;
3. Дата окончания сбора оценок (Final), содержит дату, когда конкретный эксперт произвел оценку последнего доступного для него сравнения, до этого момента не заполняется;
4. Результаты сравнения эксперта (Result), содержит матрицу, то есть является массивом массивов состоящих из целочисленных значений, каждая ячейка может иметь 4 состояния: 1, -1, 0 и быть пустой. Если ячейка является пустой, это значит, что оценка данной пары альтернатив еще не была произведена;
5. Порядковый номер исследования (ComparisonID), содержит идентификатор сравнения, данное поле отвечает за связь с исследованием многие-к-одному;
6. Порядковый номер эксперта (UserID), содержит идентификатор эксперта, данное поле отвечает за связь с экспертом многие-к-одному.

Последние два поля в таблице с результатами сравнений (ComparisonsUsers), а именно поля с порядковыми номерами исследований (ComparisonID) и экспертов (UserID), создают связь многие-ко-многим между таблицей с данными об исследованиях (Comparisons) и таблицей с данными об экспертах (Users). Сочетания данных полей являются уникальными для данной таблицы.

3.5. Реализация системы

Для начала работы, сначала, необходимо установить все требуемые библиотеки, в данном случае будут устанавливаться три библиотеки: Django, Pandas и Psycopg2. Для установки библиотек будет использоваться пакет управления виртуальным окружением для Python – Pipenv. Первым шагом необходимо активировать виртуальное окружение, в котором будет происходить работа с проектом, это делается командой «pipenv shell». Чтобы произвести установку в консоли необходимо выполнить команду «pipenv install…»:

(FQW) D:\FQW>pipenv install django pandas psycopg2

Creating a Pipfile for this project…

Installing django…

Adding django to Pipfile's [packages]…

Installation Succeeded

Installing pandas…

Adding pandas to Pipfile's [packages]…

Installation Succeeded

Installing psycopg2…

Adding psycopg2 to Pipfile's [packages]…

Installation Succeeded

Pipfile.lock not found, creating…

Locking [dev-packages] dependencies…

Locking [packages] dependencies…

Success!

Updated Pipfile.lock (15df59)!

Installing dependencies from Pipfile.lock (15df59)…

================================ 9/9 - 00:00:01

Django – это веб-фреймворк, основной инструмент для разработки веб-приложения. Pandas – библиотека для работы с данными, она предоставляет удобный инструмент для чтения файлов формата Microsoft Excel. Psycopg2 – является библиотекой для работы с базами данных PostgreSQL.

Далее необходимо установить сам инструментарий базы данных. Так как разработка ведется на персональном компьютере под управлением операционной системы Windows10, установка самого инструментария выглядит стандартно с несколькими диалоговыми окнами, в которых содержится информация об устанавливаемых программах, разработчиках, пользовательского соглашения, а также диалоговое окно для установки пароля для суперпользователя. После завершения установки, необходимо создать базу данных, а также пользователя, с помощью которого будет производиться подключение к ней. Создать пользователя можно с помощью команды «createuser …»:

(FQW) D:\FQW>createuser -P main\_user

Введите пароль для новой роли:

Повторите его:

Пароль:

Далее необходимо создать саму базу данных и определить владельца, пользователя базы данных. Это делается с помощью команды «createdb -O…»:

(FQW) D:\FQW>createdb -O main\_user fqw\_db

Пароль:

После создания базы данных можно переходить к созданию и первичной настройке самого приложения. Для создания Django-проекта необходимо использовать команду «django-admin startproject…». После чего будет создана директория в которой будут находиться все файлы с кодом приложения. По умолчанию создаются файлы с основными распространенными настройками. Данный каталог выглядит так:

│ manage.py

│

└───FQW

asgi.py

settings.py

urls.py

wsgi.py

\_\_init\_\_.py

После создания проекта, необходимо создать приложение, модуль проекта. В данном случае, так как сайт должен выполнять лишь сбор и отображение данных, разделение на несколько модулей не имеет смысла, поэтому будет создано лишь одно приложение. Для создания приложения необходимо воспользоваться командой «python manage.py startapp…». В результате будет создана еще одна директория файлов с которыми будут происходить дальнейшие манипуляции для создания веб-приложения. Данная директория будет выглядеть так:

│ admin.py

│ apps.py

│ models.py

│ tests.py

│ views.py

│ \_\_init\_\_.py

│

└───migrations

\_\_init\_\_.py

После создания приложения, его необходимо подключить к проекту, а также настроить подключение к базе данных и URL-адрес для приложения. В файле settings.py в переменной INSTALLED\_APPS необходимо оставить приложение для использования статических файлов, файлов содержащих стили страниц (CSS), функциональные файлы (JavaScript), а также подключить созданное приложение. В итоге переменная будет выглядеть так:

INSTALLED\_APPS = [  
 **'paired\_comparison'**,  
 **'django.contrib.staticfiles'**,  
]

По умолчанию Django, в качестве базы данных, использует SQLite, но для данного приложения была выбрана другая база данных, PostgreSQL, соответственно необходимо внести изменения в настройки подключения к базе данных. За это отвечает переменная DATABASES, она содержит указание на библиотеку с помощью которой будут происходить манипуляции с базой данных, название базы данных, данные пользователя с помощью которого будет производиться подключение к базе, а также адрес по которому возможно подключиться к базе. В результате переменная будет содержать следующие данные:

DATABASES = {  
 **'default'**: {  
 **'ENGINE'**: **'django.db.backends.postgresql\_psycopg2'**,  
 **'NAME'**: **'fqw\_db'**,  
 **'USER'**: **'main\_user'**,  
 **'PASSWORD'**: **'mu\_fqw'**,  
 **'HOST'**: **'localhost'**,  
 **'PORT'**: **'5432'**,  
 }  
}

Так же необходимо указать URL-адрес для отображения данных из ранее созданного приложения. Для этого необходимо отредактировать переменную urlpatterns в файле проекта под названием urls.py. Так как в данном случае будет лишь одно приложение, то и в переменную будет записана лишь одна строка, которая сообщит Django, что все обращения необходимо перенаправлять в файл urls.py, принадлежащий приложению, данный файл необходимо создать в директории приложения. В файле проекта будет внесены следующие данные:

urlpatterns = [  
 path(**''**, include(**'paired\_comparison.urls'**))  
]

После данных настроек, все изменения будут вносится в файлы принадлежащие приложению, то есть модулю проекта. В первую очередь стоит создать модели таблиц базы данных. Модели описываются в файле models.py. Каждая модель является отдельным классом.

Описание таблицы с данными об экспертах (Users), содержит четыре поля: идентификатор, являющийся первичным ключом, имя пользователя, адрес электронной почты, он может быть пустым, а также дата создания пользователя, которая принимает значение даты и времени, на момент создания объекта, данной таблицы.

**class** Users(Model):  
 uuid = UUIDField(primary\_key=**True**, default=uuid4)  
 name = CharField(max\_length=200)  
 email = EmailField(max\_length=250, blank=**True**)  
 first\_visit = DateTimeField(auto\_now=**True**)

Модель таблицы с данными об исследованиях (Comparisons), имеет восемь атрибутов, то есть столбцов: название, описание, массив с названиями параметров, адрес по которому располагаются результаты исследования, создается с помощью функции возвращающей строку из пятидесяти символов случайно выбранных из набора состоящего из латинского алфавита и цифр, дата создания исследования, адрес где сохранен файл, который был отправлен организатором исследования, в случае отсутствия папки для сохранения файлов, она создается при запуске приложения, общие результаты, массив массивов содержащие суммы оценок всех экспертов, изначально является пустым, ссылка на экспертов, образует связь многие-ко-многим с помощью промежуточной таблицы ComparisonUser.

**class** Comparisons(Model):  
 title = CharField(max\_length=200)  
 description = TextField(blank=**True**)  
 parameters\_title = ArrayField(TextField(), null=**True**)  
 url = CharField(max\_length=50, default=create\_url)  
 upload\_date = DateTimeField(auto\_now=**True**)  
 file = FileField(upload\_to=**'upload\_file'**)  
 result\_fin = ArrayField(ArrayField(IntegerField()), null=**True**)  
 users = ManyToManyField(Users, through=**'ComparisonUser'**)

Описание таблицы альтернатив (Object) состоит из пяти атрибутов: идентификатор внутри конкретного исследования, названия, массив хранящий информацию о параметрах, количество сравнений, изначально устанавливается как ноль, ссылка на конкретный объект модели описывающей исследования, устанавливает связь многие-к-одному, также задает условие, при котором, в случае удаления исследования, все связанные с ним альтернативы тоже удаляются.

**class** Objects(Model):  
 raw\_id = IntegerField()  
 title = TextField()  
 parameters = ArrayField(TextField())  
 number\_comparisons = IntegerField(default=0)  
 comparison = ForeignKey(Comparisons, on\_delete=CASCADE)

Модель описывающая таблицу результатов оценок экспертов (ComparisonUser), содержит пять атрибутов: объект таблицы пользователей, эксперт, устанавливает связь многие-к-одному, в случае удаления пользователя, данное значение становится пустым, поэтому у него имеется параметр отвечающий, за возможность принимать полем пустого значения, объект таблицы исследований, устанавливает связь многие-к-одному, в случае удаления исследования, все связанные с ним результаты сбора оценок тоже удаляются, дата начала оценок, дата окончания сравнений, является пустым, пока конкретный эксперт не предоставит оценки по каждой из возможных пар, результаты, массив из массивов, образующие сводную таблицу, которая заполняется по мере предоставления оценок экспертом, изначально является пустым.

**class** ComparisonUser(Model):  
 user = ForeignKey(Users, null=**True**, on\_delete=SET\_NULL)  
 comparison = ForeignKey(Comparisons, on\_delete=CASCADE)  
 start = DateTimeField(auto\_now=**True**)  
 final = DateTimeField(null=**True**)  
 result = ArrayField(ArrayField(IntegerField()), null=**True**)

Каждая модель, в которой не указан первичный ключ, имеет атрибут ID, который создается инструментами Django ORM.

После описания моделей, необходимо внести их в базу данных, для этого нужно создать миграцию для базы данных, то есть зафиксировать обновленную структуру базы данных. Для создания миграции необходимо выполнить команду «python manage.py makemigrations»:

(FQW) D:\FQW\FQW>python manage.py makemigrations

Migrations for 'paired\_comparison':

paired\_comparison\migrations\0001\_initial.py

- Create model Comparisons

- Create model Users

- Create model Objects

- Create model ComparisonUser

- Add field users to comparisons

После создания миграции, база данных все еще не соответствует необходимому образу, не содержит описанные таблицы, так как после создания миграции, ее необходимо применить к базе данных. Данный процесс производится с помощью команды «python manage.py migrate»:

(FQW) D:\FQW\FQW>python manage.py migrate

Operations to perform:

Apply all migrations: paired\_comparison

Running migrations:

Applying paired\_comparison.0001\_initial... OK

В результате выполнения этих двух команд в базе данных будут созданы пять таблиц, четыре таблицы описанные выше и одна сервисная таблица, созданная инструментами Django ORM для отслеживания миграций базы данных.

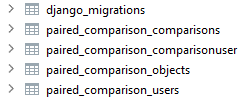


Рисунок 3.3. Список таблиц базы данных

После создания моделей объектно-ориентированного представления базы данных, будет логичным перейти к разработке основных страниц приложения. Первая страница, которая будет разработана будет страница создания исследования, так как именно с нее начинается сбор данных, и именно она является локальным началом исследования. Для отправки данных из браузера организатора исследования, необходимо создать HTML-форму. Для данной цели Django имеет инструменты для создания форм на основе моделей таблиц базы данных.

**class** ComparisonsForm(ModelForm):  
 index\_check = BooleanField(help\_text=**'Является ли первый столбец индексами'**, required=**False**)  
 **class** Meta:  
 model = Comparisons  
 fields = [**'title'**, **'description'**, **'file'**]  
 widgets = {  
 **'title'**: TextInput(attrs={  
 **'placeholder'**: **'Введите название для исследования'**,}),  
 **'description'**: Textarea(attrs={  
 **'placeholder'**: **'Введите описание для исследования'**,})  
 }

Для создания полей формы необходимо создать класс, содержащий нужные поля, а также мета-класс в котором формируются поля формы основанные на модели таблицы. В данном случае создается флажковая кнопка, которая не связана с базой данных, а будет использоваться в процессе чтения данных из файла с альтернативами, присланного организатором исследования. В мета-классе описываются поля основанные на модели таблицы, поля будут основаны на модели таблицы исследований. Организатор заполняет лишь три поля данной таблицы в форме: название исследования, описание, а также файл с списком альтернатив в формате excel-таблицы. Для полей названия и описания, указываются подсказки.

За формирование данных для страницы, то есть создание формы и обработку данных присланных с страницы создания исследования отвечает одна функция отображения. В случае если пользователь зашел впервые на данную страницу, он запрашивает ее с помощью метода GET, тогда сервер просто создает объект формы описанной выше и отправляет его в HTML-шаблон, в котором содержит структуру страницы, после чего отправляет получившуюся HTM-страницу пользователю.

**def** load(request):  
 data = {}  
 **if** request.method == **'POST'**:  
 …  
 **else**:  
 form = ComparisonsForm()  
 data[**'form'**] = form  
 **return** render(request, **'main/load.html'**, data)

Также данная функция обрабатывает и полученные данные из формы. Если организатор исследования заполнил форму и отправил данные, то браузер отправляет запрос типа POST. В таком случае скрипт попытается извлечь из файла данные, в случае успеха, оригинальный файл будет переименован, если же прочесть файл не удастся, на странице регистрации исследования будет выведено предупреждение о том, что файл не соответствует формату.

…

**if** form.is\_valid():  
 raw\_file, file, data[**'errors'**] = **None**, **None,** []  
 **try**:  
 raw\_file, file = read\_file(request)

**except** (ValueError, IOError):  
 data[**'errors'**].append(**'Файл не является таблицей excel'**)  
 **else**:

…

В случае если файл был успешно прочитан, полученные из него данные должны содержать какие-то данные помимо заголовков. Если данная проверка была пройдена, то создается объект модели таблицы информации об исследовании, из полученных из файла данных формируются объекты альтернатив, связываются с объектом исследования и сохраняются, оригинальный файл уже с новым именем сохраняется в соответствующую директорию. После успешной записи полученной информации в базу данных организатор исследования перенаправляется на страницу с информацией о созданном исследовании, так же на этой странице будут отображаться полученные оценки от экспертов.

…  
 **if not** file.empty:  
 comparison = create\_comparison(request.POST, raw\_file, file)  
 **return** redirect(**'compare\_admin'**, id=comparison.id, admin\_token=comparison.url)

**else**:

data[**'errors'**].append(**'Список не распознан. Возможно файл был пустым?'**)

form = ComparisonsForm(request.POST, request.FILES, )  
 …

В случае если извлеченные данные не содержали ничего, или же из файла были получены лишь строка заголовков, то сообщение об этом выводится на страницы регистрации исследования.

Для того, чтобы пользователи могли получить доступ к функции создания исследования, необходимо данную функцию привязать URL-адресу, это делается в файле urls.py данного приложения. Адрес и функцию отображения необходимо вписать в переменную urlpatterns, она содержит список. В данный список необходимо внести строку: «path(**'load'**, load, name=**'load'**)». Данная строка обозначает, что запрос по адресу /load будет обработан функцией load.

После успешной регистрации исследования, организатор перенаправляется на страницу с информацией указанной им, а также результатами исследования, логично, что сразу после регистрации исследования результаты не будут отображены из-за их отсутствия. Адрес данной страницы содержит в себе идентификатор исследования, а также случайно сгенерированную строку, которая должна усложнить получение данных результатов исследования не доверенным лицам. В первую очередь данная функция пытается получить информацию из базы данных о запрашиваемом исследовании. В случае если получить объект необходимого исследования не удалось обнаружить в базе, выводится сообщение о неправильно введённой ссылке.

**def** compare\_admin(request, id, admin\_token):  
 data = {}  
 **try**:  
 comparison = Comparisons.objects.get(id=id, url\_admin=admin\_token)  
 **except** Comparisons.DoesNotExist:  
 data[**'errors'**] = [**'Проверьте правильность введенной ссылки.'**]  
 **else**:

…

Если же в базе нашлось совпадение, по запрашиваемому сравнению, то из полученных данных составляется словарь, который в дальнейшем будет передан в HTML-шаблон, для их отображения. Словарь содержит в себе информацию о самом сравнении, список альтернатив отсортированный по идентификатору внутри исследования, индивидуальные результаты экспертов, информацию о самих экспертах, а также две ссылки, которые организатор должен сохранить, одна ссылка ведет на данную страницу, ссылка на страницу с результатами исследования, а другая выдается экспертам, для проведения процедуры оценки альтернатив.

…  
 data[**'comparison'**] = comparison  
 data[**'objects'**] = comparison.objects\_set.order\_by(**'raw\_id'**).all()  
 data[**'results'**] = comparison.comparisonuser\_set.all()  
 data[**'users'**] = comparison.users.all()  
 host = request.META[**"HTTP\_HOST"**]  
 data[**'urls'**] = {  
 **'main'**: **f'{**host**}{**reverse(compare\_target, args=[comparison.id])**}'**,  
 **'admin'**: **f'{**host**}{**reverse(compare\_admin, args=[comparison.id, comparison.url])**}'** }  
 **return** render(request, **'main/compare\_admin.html'**, data)

В случае первой загрузки данной страницы, вместо результатов исследования, отображается сообщение о том, что ни одной оценки не было получено. Данная функция представления, также закрепляется в файле urls.py строкой «path(**'compare/<int:id>/<str:admin\_token>'**, compare\_admin, name=**'compare\_admin'**)», данная строка обозначает, что запрос по адресу /compare/идентификатор\_сравнения/случайная\_строка, будет обработан функцией compare\_admin. Задача по распространению ссылки для сбора оценок стоит перед самим организатором исследования.

Когда эксперт переходит по предоставленной ему ссылке, сначала происходит попытка получить информацию о исследовании из базы данных. Если исследование с идентификатором из адреса не было найдено в базе, на странице появится уведомления о этом.

**def** compare\_target(request, id):  
 data= {} **try**:  
 comparison = Comparisons.objects.get(id=id)  
 **except** Comparisons.DoesNotExist:  
 data[**'errors'**] = [**'Список для попарного сравнения не найден.'**]

response = render(request, **'main/compare.html'**, data)  
 **else**:

…

Если в базе данных было найдено совпадение по запрашиваемому исследованию, следующим шагом является идентификация эксперта. Для данной цели на устройство пользователя после регистрации записывается данные cookie, содержащие идентификатор пользователя. Эти данные находятся на устройстве пользователя в течении месяца, после последней сессии оценок, после чего удаляется. Если этих данных на устройстве пользователя обнаружить не удалось, считается, что эксперт впервые учувствует в исследованиях и перенаправляется на страницу, где ему необходимо указать данные о себе.

…  
 **try**:  
 user\_id = request.COOKIES[**'uuid'**]  
 **except** KeyError:  
 **return** redirect(**'new\_user'**, id=id)  
 **else**:

…

В случае обнаружения данных на устройстве эксперта, он направляется на страницу для сравнения и обновляется срок жизни cookie.

…  
 data[**'comparison'**] = comparison  
 response = render(request, **'main/compare.html'**, data)  
 response.set\_cookie(**'uuid'**, user\_id, max\_age=2678400)  
 **return** response

Сами альтернативы для оценки запрашиваются с помощью AJAX-запросов уже на самой странице. Это сделано для того, чтобы избежать частого обновления страницы, а также сократить время обновления пар для сравнения. В случае если страница была только загружена, если источником вызова функции оказалась не кнопка, а окно браузера, то просто отправляется запрос на получение пары альтернатив для сравнения по адресу «/compare/идентификатор\_исследования/get\_targets».

async **function** setAndGetData(source){  
 url = window.location.href +**'/get\_targets'  
 if** (!source.window){  
 …

}**else**{  
 targets = await fetch(url)  
 }  
 json = await targets.json()  
 **return** json  
}

Если же источником события вызвавшего функцию является не окно браузера, то есть нажатие кнопки, то происходит формирование JSON объекта для отправки на сервер с результатами сравнения.

…

targetsButtons = targetsBlock.children  
 result = {}  
 **if** (source.id == **'draw'**){

…

Если была нажата кнопка пропуска оценки, отправляемые данные добавляется поле draw, которое указывает, на то, что в данном сравнении фаворита не оказалось и обе альтернативы являются равноценными.

…  
 result.draw = **'true'** result.winner = targetsButtons[0].id  
 result.loser = targetsButtons[1].id  
 }**else**{

…

Если же эксперт все же выбрал одну из альтернатив и нажал соответствующую кнопку в поле winner записывается идентификатор выбранного объекта, а в поле loser записывается идентификатор альтернативного варианта.

…  
 **for** (targetsButton of targetsButtons){  
 **if** (targetsButton.id == source.id){  
 result.winner = targetsButton.id  
 }**else if** (targetsButton.id != **'draw'**){  
 result.loser = targetsButton.id  
 }  
 }}

…

После формирования результатов оценки, составляется AJAX запрос на сервер. Отправляется он методом POST, так как отправленные данные будут записаны в базу данных. В заголовках запроса указывается кодировка отправляемых данных, а также CSRF-токен, который Django обязует отправлять при запросах подобного сипа. Данный токен хранится в данных cookie на стороне клиента. В тело запроса помещаются данные с результатами оценки экперта.

…  
 targets = await fetch(url, {  
 method: **'POST'**,  
 headers: {  
 **'Content-Type'**: **'application/json;charset=utf-8'**,  
 **'X-CSRFToken'**: getCookie(**'csrftoken'**)  
 },  
 body: JSON.stringify(result)  
 })

…

В результате данной функции на сервер отправляется запрос. Он может быть пустым, что означает лишь получение альтернатив для сравнения или же он может нести с собой JSON-файл, что подразумевает обновление результатов исследования. Все запросы обрабатываются по адресу «/compare/идентификатор\_исследования/get\_targets». За обработку данного адреса отвечает функция get\_targets. Для начала функция собирает и формирует данные об исследовании по его идентификатору.

**def** get\_targets(request, id) :  
 data = {}

comparison = Comparisons.objects.get(id=id)  
 data[**'comparison'**] = {  
 **'id'**: comparison.id,  
 **'title'**: comparison.title,  
 **'description'**: comparison.description,  
 **'parameters\_title'**: comparison.parameters\_title  
 }

…

После формирования общих данных о исследовании. Происходит поиск уже существующих оценок данного эксперта. В случае если это была первая оценка, то создается объект модели результатов оценок экспертов. После этого проверяется наличие данных в теле запроса. При наличии таковых, они записываются в индивидуальный результат эксперта, а также учувствует в формировании общего результата в исследовании.

…

User = Users.objects.get(uuid=request.COOKIES[**'uuid'**])  
 comparison\_user = get\_comparison\_user(comparison, user)  
 **if** request.body != **b''**:  
 set\_result(request.body, comparison\_user)  
 data.update(get\_targets\_to\_dict(comparison, comparison\_user))  
 **return** JsonResponse(data)

Последним шагом в данной функции является подбор и отправка пары альтернатив для сравнения эксперту. Данная функция ищет объект исследования с наименьшим количеством сравнений, у которого имеется альтернатива с которой данный эксперт ее еще не сравнивал. Если такая пара альтернатив находится, то она добавляется в словарь и отправляется эксперту. В случае если таких альтернатив не находится, эксперт получает сообщение о том, что он оценил всевозможные варианты.

В случае если эксперт впервые посетил страницу сбора оценок, то есть у него на устройстве не сохранен его идентификатор, приложение воспринимает его как нового пользователя и направляет на страницу, где ему необходимо заполнить форму с данными о себе. Данная форма создается на основе модели пользователей инструментами Django.

**class** UsersForm(ModelForm):  
 **class** Meta:  
 model = Users  
 fields = [**'name'**, **'email'**]  
 widgets = {  
 **'name'**: TextInput(attrs={**'placeholder'**: **'ФИО'**}),  
 **'email'**: TextInput(attrs={**'placeholder'**: **'email'**})  
 }

Эксперту необходимо заполнить всего два поля: свое имя, а также адрес электронной почты. Когда пользователь отправляет свои данные на сервер, создается новый объект модели пользователей, после чего эксперта перенаправляет обратно на страницу сбора оценок исследования, вместе с ответом отправляются также данные cookie для дальнейшей идентификации пользователя.

**def** new\_user(request, id):  
 **if** request.method == **'POST'**:  
 user = create\_user(request.POST)  
 response = redirect(**'compare\_target'**, id=id)  
 response.set\_cookie(**'uuid'**, user.uuid, max\_age=2678400)  
 **else**:  
 form = UsersForm()  
 data= {**'form'**: form}  
 response = render(request, **'main/create\_user.html'**, data)  
 **return** response

Если же эксперт только зашел на данную страницу, то ответ содержит лишь форму, и HTML-шаблон для ее отображения.

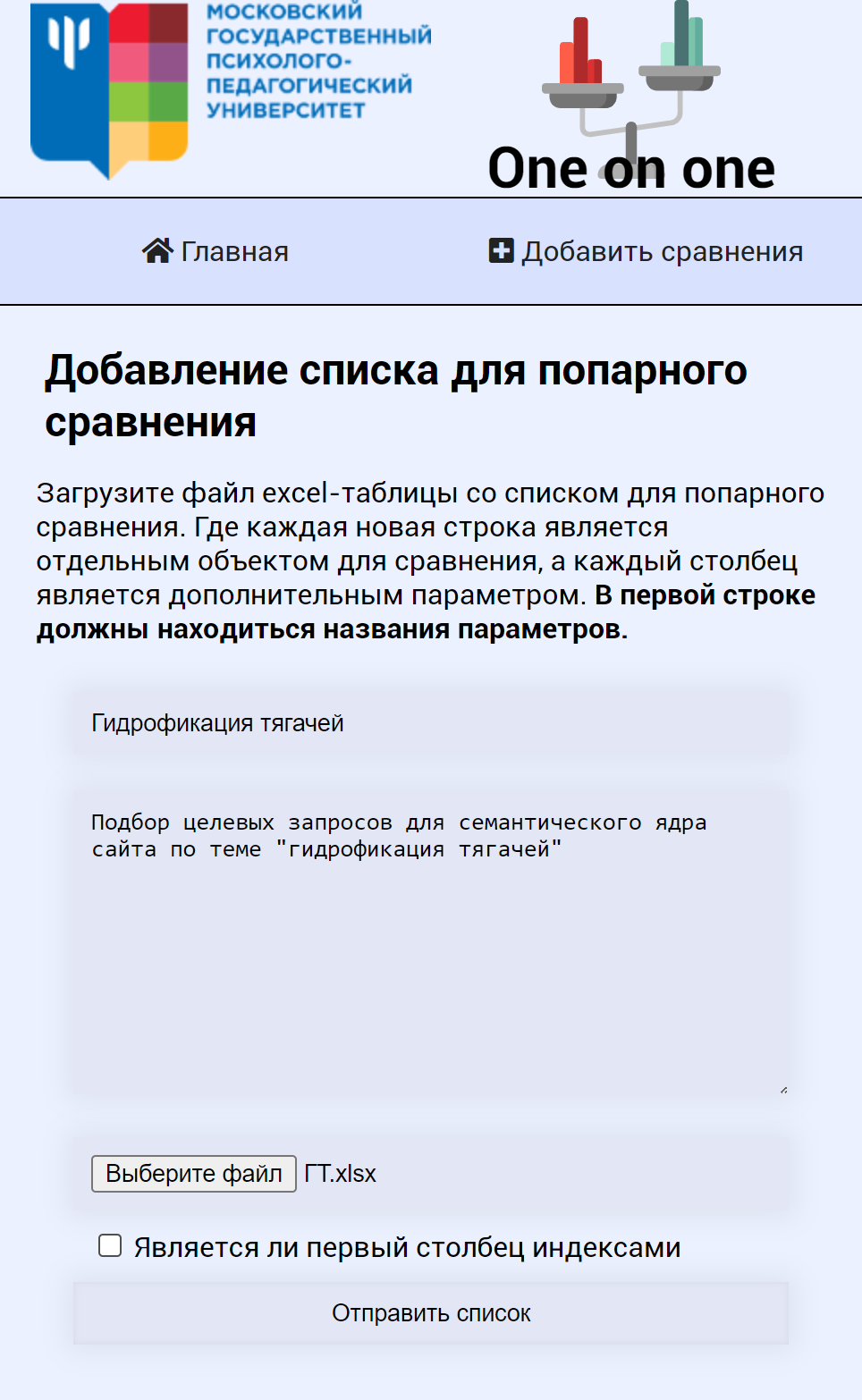


Рисунок 3.4. Форма регистрации исследования

Как уже было сказано выше, первой страницей, с которой начинается исследование для данного приложения, является страница регистрации исследования, где организатор исследования должен указать название исследования, по желанию дать ему какое-то описание, обязательно должен загрузить таблицу, содержащую список альтернатив для сравнений, также организатор должен указать пронумерованы ли альтернативы в предоставленном файле.

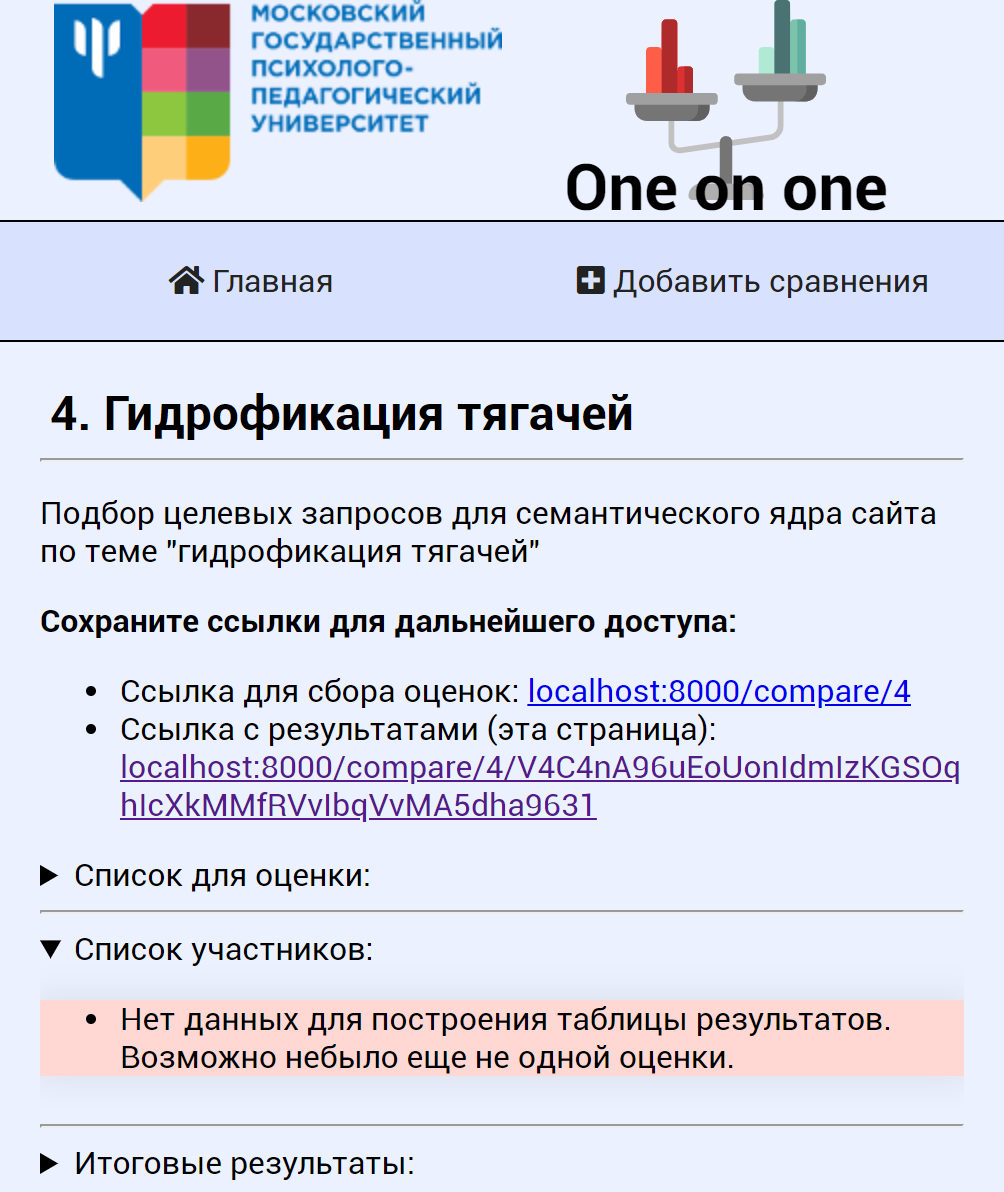


Рисунок 3.5. Страница информации и результатов исследования

После чего организатора перенаправляет на страницу с данными об исследовании, которые были только что предоставлены. Также данная страница содержит результаты исследования. Но в случае регистрации исследования, результатов еще нет. На данной странице находятся две ссылки, одна из которых предназначается приглашаемым экспертам, вторая же дублирует адрес данной страницы, обе эти ссылки организатор должен где-нибудь сохранить, в случае их утери будет необходимо обращаться напрямую к базе данных. Распространение ссылки для сбора оценок экспертов является задачей организатора исследования.

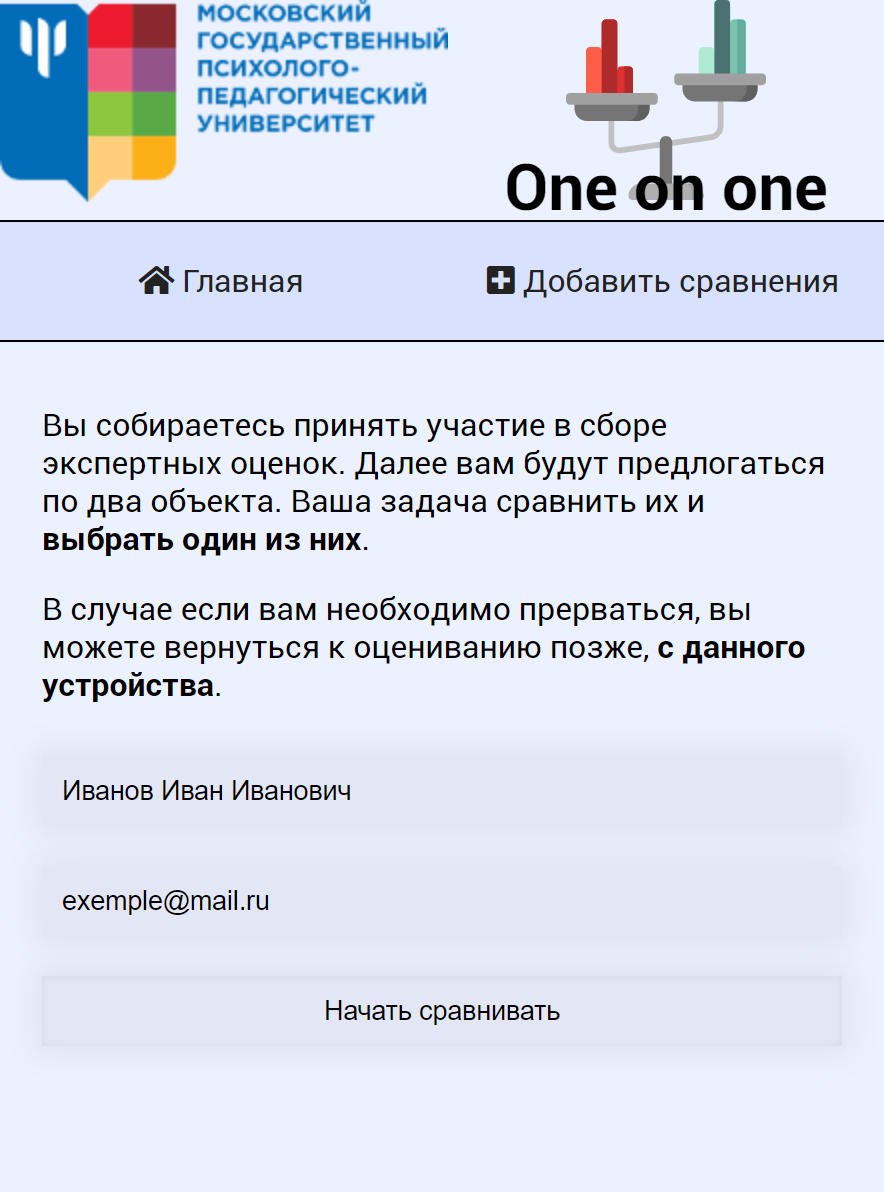


Рисунок 3.6. Страница регистрации эксперта

Когда эксперт впервые переходит по предоставленной ему организатором ссылки, он направляется на страницу регистрации. Небольшая форма с двумя полями: для имени эксперта и электронной почты для связи с ним.

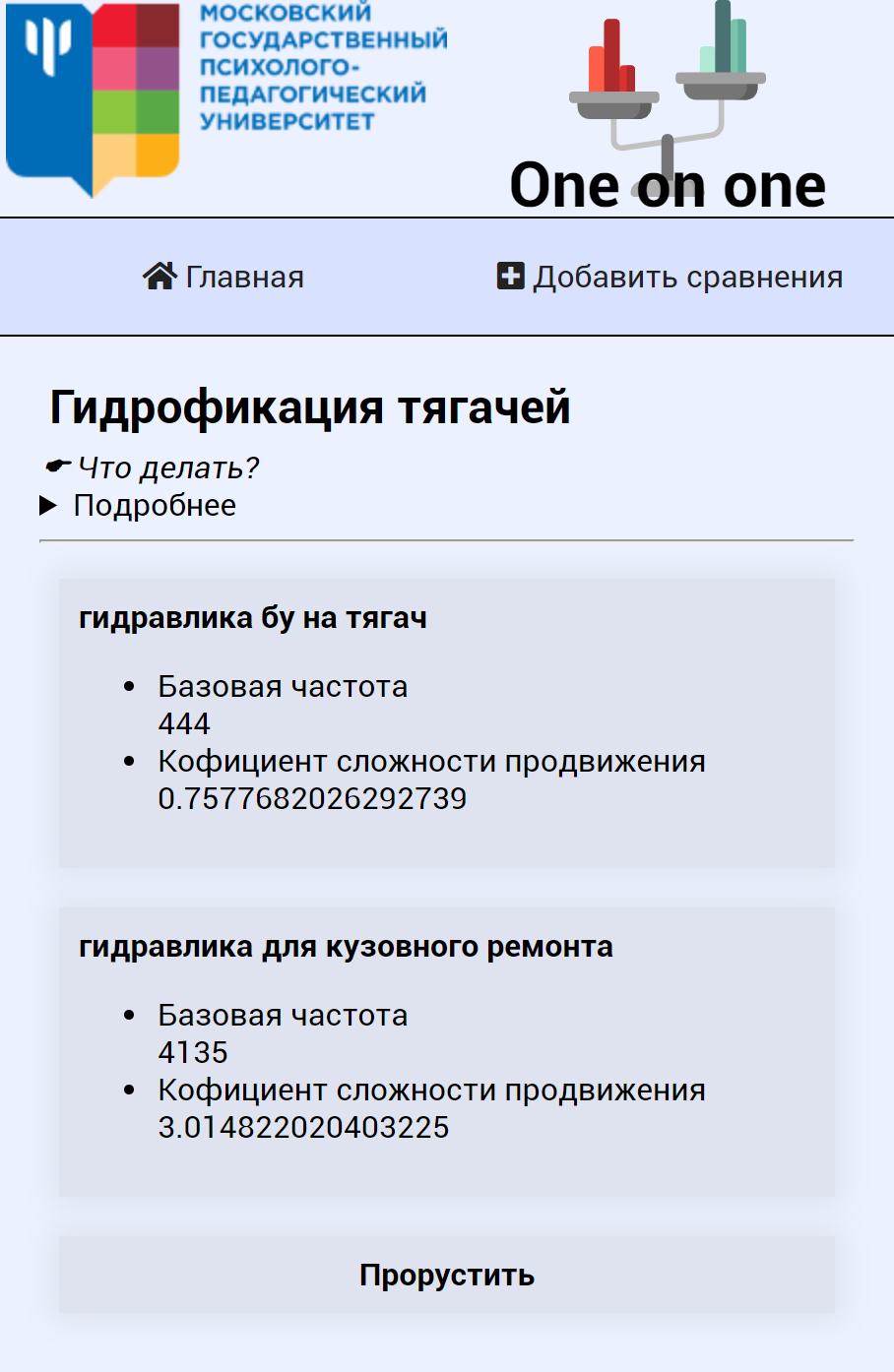


Рисунок 3.7. Страница сбора оценок

После отправки своих данных, эксперт перенаправляется на страницу сбора оценок, где ему будет предоставлен выбор из двух альтернатив. В случае равнозначности данных объектов эксперт имеет возможность пропустить данное сравнение. В результате после всех сравнений эксперту будет показано уведомления, что он произвел все возможные сравнения в данном исследовании.

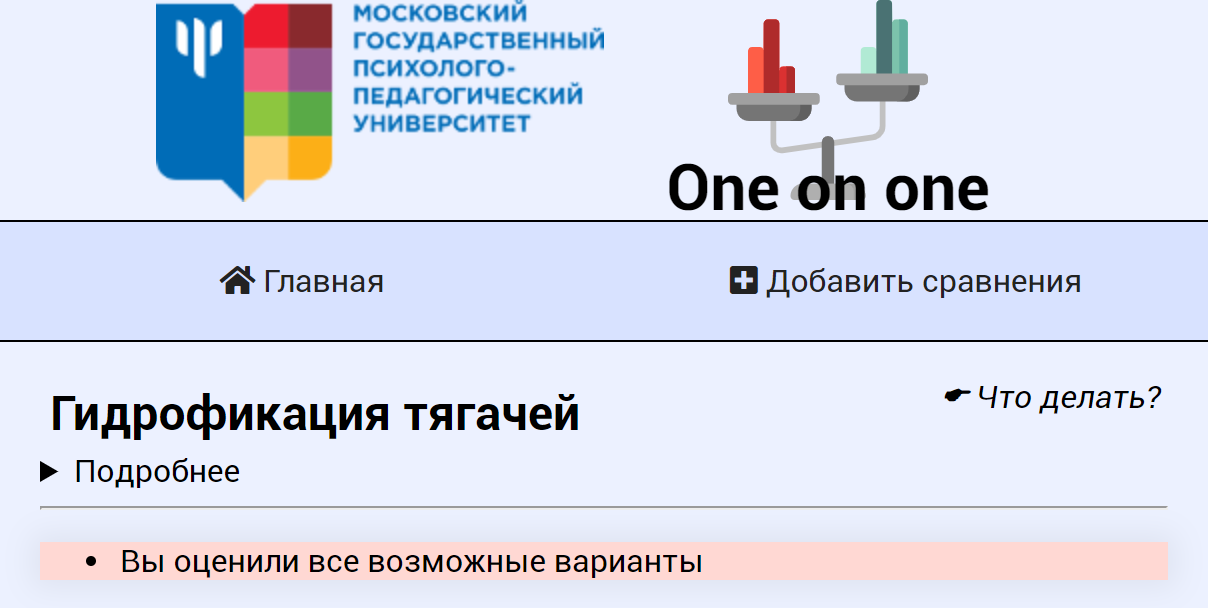


Рисунок 3.8. Пар нет

Заключение

В результате проделанной работы была изучена тема метода экспертных оценок – парного сравнения. Было обнаружено, отсутствие специализированного программного обеспечения, для автоматизации сбора экспертных оценок. В результате чего было принято решение о разработке веб-приложения для сбора экспертных оценок.

Были выбраны и изучены инструменты для разработки веб-приложения. Спроектирована и реализована база данных для хранения собранной информации. Также было смоделировано приложение и реализовано на языке программирования Python.

Получившееся приложение адаптировано под мобильные устройства, а также имеет хорошую поддержку программ экранного доступа. Данное приложение имеет базовый функционал, оно предоставляет возможность собирать экспертные оценки для объектов воспринимаемых как целое, чьи характеристики нежелательно разделять.

Логичным развитием данного программного продукта будет добавление профилей пользователей, для сохранения историй оценок, средств для анализа полученных данных, а также разработка API, для предоставления возможности интегрировать программу в другие приложения.

Список литературы

1. Орлов А.И. Организационно-экономическое моделирование. – Том 2. – Москва : МГТУ им. Н.Э. Баумана, 2011
2. Балябин В.А. Заславский А.А. Построение правильных матриц парных сравнений. Результаты вычислительного эксперимента // Экономика и математические методы. – 2018. – №1.
3. Давид Г. Метод парных сравнений. – Статистика, 1978
4. Данн Н. Функциональный Python: курс для начинающих программистов на Python. – Webucator, 2020
5. Джордж Н. Освоение Django. – Австралия, Гамильтон: GNW Independent Publishing, 2020
6. Джордж Н. Создание сайта с помощью Django 3. – Австралия, Гамильтон: GNW Independent Publishing, 2020
7. Метод парного сравнения [Электронный ресурс] // Актион, Кадры и HR. – 2019. – URL: <https://www.hr-director.ru/article/67496-metod-parnogo-sravneniya-19-m10>
8. Методы попарного сравнения [Электронный ресурс] // Энциклопедия по экономике. – URL: <https://economy-ru.info/info/33282/>
9. Java [Электронный ресурс] // Википедия. – 2021. – URL: <https://ru.wikipedia.org/wiki/Java>
10. Веб-приложение [Электронный ресурс] // Википедия. – 2021. – URL: <https://ru.wikipedia.org/wiki/Веб-приложение>
11. Python [Электронный ресурс] // Python Software Foundation. – 2021. – URL: <https://www.python.org/>
12. The Python Package Index [Электронный ресурс] // PyPI. – 2021. – URL: <https://pypi.org/>
13. Django documentation [Электронный ресурс] // Django Software Foundation. – 2021. – URL: <https://docs.djangoproject.com/>
14. PostgreSQL: The World's Most Advanced Open Source Relational Database [Электронный ресурс] // The PostgreSQL Global Development Group. – 2021. – URL: <https://www.postgresql.org/>
15. Документация к Postgres Pro Standard 13.2.2. [Электронный ресурс] // Постгрес Профессиональный. – 2020. -URL: <https://postgrespro.ru/docs/>
16. Psycopg – PostgreSQL database adapter for Python [Электронный ресурс] // Psycopg. – 2020. – URL: <https://www.psycopg.org/docs/>