



МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное
учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт
Кафедра

ИВТИ
ВМСС

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(бакалаврская работа)

Направление 09.03.01 – Информатика и вычислительная техника
(код и наименование)

Направленность (профиль) Вычислительные машины, комплексы,
системы и сети

Форма обучения Очная
(очная/очно-заочная/заочная)

Тема: Разработка web-приложения по расчету прихода солнечной
радиации на горизонтальную поверхность

Студент A-12-17 Палагина С.А.
группа подпись фамилия и инициалы

Научный ст. преп. Карвовский Д. А.
руководитель
уч. степень должность подпись фамилия и инициалы

Консультант к.т.н. доцент ГВИЭ НИУ МЭИ Васьков А. Г.
уч. степень должность подпись фамилия и инициалы

«Работа допущена к защите»

Зав. кафедрой к.т.н. доцент Вишняков С. В.
уч. степень звание подпись фамилия и инициалы

Дата _____

Москва 2021

Оглавление

| | |
|--|------------|
| ВВЕДЕНИЕ..... | 3 |
| 1. ПОСТАНОВКА ЗАДАЧИ И ВЫБОР СРЕДСТВ РАЗРАБОТКИ..... | 5 |
| 1.1. Постановка задачи | 5 |
| 1.2. Средства разработки | 7 |
| 2. РАЗРАБОТКА БАЗЫ ДАННЫХ..... | 13 |
| 2.1. Разработка структуры базы данных | 13 |
| 2.2. Разработка таблиц | 14 |
| 2.3. Соединение веб-приложения с базой данных | 22 |
| 2.4. Структура базы данных | 23 |
| 3. РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ..... | 25 |
| 3.1. Архитектура разрабатываемого web-приложения | 25 |
| 3.2. Структура разрабатываемого web-приложения..... | 26 |
| 3.3. Разработка алгоритмов поиска | 30 |
| 3.4. Разработка алгоритмов интерполяции данных | 39 |
| 3.4.1. Интерполяция “День сурка” | 39 |
| 3.4.2. Интерполяция “Линейная”..... | 42 |
| 3.5. Экспорт в csv | 48 |
| 3.6. Разработка пользовательского интерфейса..... | 54 |
| 6. ТЕСТИРОВАНИЕ | 68 |
| 6.1. Тестирование алгоритма экспорта | 69 |
| 6.2. Тестирование поиска данных | 74 |
| ЗАКЛЮЧЕНИЕ | 78 |
| СПИСОК ЛИТЕРАТУРЫ..... | 79 |
| <i>Приложение А. Техническое задание</i> | <i>81</i> |
| <i>Приложение Б. Листинг программы серверной части веб-приложения</i> | <i>85</i> |
| <i>Приложение В. Листинг программы клиентской части веб-приложения</i> | <i>117</i> |
| <i>Приложение Г. Схема алгоритма интерполирования данных</i> | <i>131</i> |
| <i>Приложение Д. Схема алгоритма экспорта данных в формате csv.....</i> | <i>133</i> |

ВВЕДЕНИЕ

Еще в 2004 году на кафедре института ГВИЭ была разработана база данных, хранящая среднегодовые данные о приходе солнечной радиации на горизонтальную поверхность при средних условиях облачности. Данные, содержащиеся в базе, были взяты из «Научно-прикладного справочника по климату СССР» [19]. Данные, публикуемые в научно-прикладном справочнике, были рассчитаны и обобщены по принципу максимальной информативности многолетних наблюдений метеорологических станций. Справочник выпускался в 1990-х годах и состоял из шести частей, каждая из которых освещала какой-либо регион бывшего СССР.

Сотрудники института ГВИЭ проделали огромную работу по анализу и структурированию такого огромного количества данных. База данных была создана в формате excel файла. Компьютерная программа Excel сама по себе является мощным аналитическим инструментом, предназначенным для хранения, организации и анализа данных, а так же для решения многих вычислительных задач. Существующая база данных грамотно организована, имеет удобный функционал и хорошо структурирована. Но тут возникает вопрос о том, как предоставлять для пользования эту базу данных все желающим. Живя в век информационных технологий, уже не совсем рационально распространять подобного рода информацию на электронных носителях, особенно, если это не один или два человека, а, допустим, целый поток студентов. Намного удобнее создать веб-интерфейс по предоставлению информации, чтобы любой желающий смог перейти по ссылке в интернете и найти все нужные данные без особых трудностей.

Институт ГВИЭ предложил решить вышеупомянутую задачу. С этой целью было разработано веб-приложение для поиска данных о приходе солнечной радиации на горизонтальную поверхность, с последующей их обработкой и

возможностью скачивания искомой информации. Данное приложение находится в свободном доступе, то есть каждый желающий сможет просматривать данные. Не было смысла закрывать информацию, которая и так была общедоступна. В рамках данного приложения не было предусмотрено предоставление пользователям разных привилегий доступа, поскольку никакие новые данные в базу не добавлялись, а существующие не будут подвергаться изменению. Такие послабления также исходят и того факта, что исходные данные – это среднесрочные наблюдения, сформированные уже достаточно давно, то есть они являются, своего рода, базовыми.

1. ПОСТАНОВКА ЗАДАЧИ И ВЫБОР СРЕДСТВ РАЗРАБОТКИ

1.1. Постановка задачи

База данных, с которой предстояло работать, содержит среднемноголетние данные о приходе солнечной радиации на горизонтальную поверхность для различных актинометрических станций, которые располагаются в регионах нашей страны и на территории бывшего СССР. Было необходимо эти данные проанализировать и для удобства работы перенести в реляционную базу данных MySQL.

Основные требования к разрабатываемой базе данных MySQL:

- данные должны быть структурированы;
- должна быть исключена избыточность информации, ее дублирование и противоречивость;
- база данных должна быть сформирована таким образом, чтобы запросы для обращения к ней были как можно проще;
- база данных должна быть масштабируемая, то есть она должна быть построена таким образом, чтобы было возможно ее дальнейшее расширение и изменение.

Веб-приложение является, своего рода, механизмом для доступа к вышеупомянутой базе данных. Из этого следует, что функционал самого приложения будет не слишком сложным. Ключевым является тот факт, что отсутствует возможность добавления, удаления и обновления записей в базе данных. Следовательно, задача сводится к разработке удобного интерфейса для поиска данных. Помимо этого данные еще необходимо в некоторых случаях интерполировать, а полученный результат поиска скачивать.

Требования, предъявляемые к разрабатываемому приложению:

- удобство использования – пользовательский интерфейс должен быть интуитивно понятным;
- наглядность – запрашиваемая информация должна корректно и понятно отображаться для просмотра;
- настраиваемость – пользователь должен иметь возможность задать подходящие ему параметры для интерполяции, такие как: тип интерполяции, шаг дискретизации и период, за который необходимо получить данные;
- возможность скачивания – пользователю должна быть предоставлена возможность экспортировать полученные данные.

Формулировка поставленной задачи представлена в следующем виде: необходимо разработать веб-интерфейс для удобного поиска информации по базе данных, содержащей информацию по актинометрическим станциям, с последующим интерполированием полученных данных, с возможностью задания параметров для интерполяции, а также с наличием функционала для экспорта готовых данных в формате csv.

Веб-приложение состоит из трех основных блоков: поиск, интерполяция и экспорт.

1. Поиск - предоставление возможности поиска актинометрической станции по нескольким основным параметрам: по названию станции, по номеру станции и по географическим координатам станции.

2. Интерполяция – предоставление возможности интерполировать среднемесячные значения прихода солнечной радиации на горизонтальную поверхность в зависимости от заданных параметров: тип интерполяции, шаг дискретизации и период, за который необходимо получить данные.

3. Экспорт – предоставление возможности экспортировать либо исходные данные, либо интерполированные в формате csv, с заданной структурой файла.

1.2. Средства разработки

Первое, что необходимо сделать при создании веб-приложения, - это проанализировать все доступные средства, для быстрой и качественной разработки. Этими средствами являются язык программирования, фреймворк и СУБД.

Выбор языка программирования

Сначала предстоит выбрать язык программирования, который поможет создать многофункциональное и удобное веб-приложение. Сравним два наиболее популярных языка, используемых в веб-разработке: Python и PHP.

Несмотря на то, что PHP и Python пользуются большим спросом, каждый из них служит разным целям. В то время как PHP изначально разрабатывался как веб-язык, Python создавался как язык программирования общего назначения. И PHP, и Python являются динамически типизированными и объектно-ориентированными языками, которые совместимы с несколькими операционными системами, поэтому в некотором смысле они похожи [13].

Сравним два языка.

1. Сложность

Python и PHP отличаются в простоте использования.

PHP изначально разрабатывался как инструмент для создания динамических веб-сайтов и веб-приложений. Разработка PHP может быть слишком сложной из-за его жесткого синтаксиса. Например, названия функций могут быть похожи, но выполнять совсем разные операции. Если в коде будет ошибка, то язык позволит использовать ее. Когда недостаток будет очевидным, ошибку будет очень сложно найти [13,14].

Напротив, Python довольно прост для понимания благодаря его высокой читабельности. Кроме того, в Python написанный код можно легко интерпретировать и читать, что упрощает процесс отладки, поскольку в простом синтаксисе легче найти ошибки и уязвимости. Python помогает упростить сложное программирование. Он внутренне обрабатывает адреса памяти, уборщиков мусора [13,14].

2. Безопасность

Python считается одним из самых безопасных языков программирования. Python имеет несколько функций безопасности, которые можно использовать для создания сложных приложений с точными целями и функциями. Например, его фреймворк Django имеет встроенные функции безопасности, которые позволяют разработчикам эффективно бороться с угрозами.

В отличие от Python, большое количество приложений PHP может иметь проблемы с безопасностью из-за старых методов кодирования и плохого кода [13,14].

3. Популярность

И Python, и PHP имеют отличную поддержку сообщества. PHP присутствует на рынке с 1995 года и сумел сформировать огромное сообщество разработчиков, готовых оказать поддержку.

Python был выпущен еще раньше, в 1991 году. Как и его конкурент, Python имеет большое сообщество разработчиков, которые непрерывно разрабатывают веб-приложения, и, следовательно, поддержка сообщества является выдающейся [13,14].

Поэтому, учитывая популярность языков, сложностей в поиске ответов на возникающие вопросы возникнуть не должно в обоих случаях.

Сравнивая и другие параметры языков, такие как: наличие фреймворков, подключение к базе данных, масштабируемость, было решено остановиться на языке программирования Python. Данный язык очень прост в освоении, имеет высокое качество кода, имеет внушительное количество различных библиотек, хоть и немного проигрывает PHP производительности и не поддерживает подключение к базе данных так широко, как PHP, однако, эти недостатки в рамках разработки нашего веб-приложения не вызовут явных неудобств.

Выбор фреймворка

Фреймворк – это рабочая среда, которая помогает разработчику быстро и качественно создавать программный продукт, не отвлекаясь на мелочи. Они оснащены полезными функциями, такими как шаблоны, механизмы управления сессиями и библиотеки доступа к базам данных.

Django и Flask – два очень широко используемых фреймворка для Python. Оба фреймворка имеют открытый исходный код.

Django был создан для упрощения процесса разработки сайта. Он предлагает веб-разработчикам набор компонентов, которые увеличивают скорость разработки и облегчают весь процесс.

Flask – это небольшой веб-фреймворк, который не требует использования определенных библиотек или инструментов. Хотя в нем и нет компонентов, интегрирующих сторонние библиотеки, Flask поддерживает расширения, позволяющие легко подключить недостающие функции [15, 16].

Посмотрим чем они отличаются.

1. ORM

Django предоставляет свой собственный Django ORM (объектно-реляционное отображение) и использует модели данных, в то время

как Flask вообще не имеет моделей данных и в принципе поставляется без ORM, следовательно, придется подключать стороннюю ORM. Легче использовать уже готовое решение.

2. Встроенные пакеты

Flask очень минималистичный, поэтому нам необходимо использовать внешние библиотеки в зависимости от поставленной задачи. Это делает Flask гибким и расширяемым.

Django имеет огромное количество встроенных пакетов, следовательно, поиск пакета для сборки и запуска приложения пройдет с меньшими усилиями.

3. Область использования

Django был разработан для быстрой разработки сложных веб-приложений. Он предоставляет необходимые инструменты для реализации масштабируемой и поддерживаемой функциональности. С другой стороны, простота Flask позволяет быстрее создавать небольшие приложения.

Также стоит учесть, что Django автоматически создает строго структурированную систему директорий при создании приложения. Если же мы используем Flask, то нам необходимо самим продумывать структуру приложения, что может вызвать дополнительные сложности [16].

Основное различие между Django и Flask в том, что Django предоставляет полнофункциональную среду Model-View-Controller («Модель–Представление–Управление»). В Django паттерн MVC чаще обозначают как MTV («Модель–Шаблон–Представление»).

В концепции MTV:

- «Model» — модель, уровень доступа к данным. Сюда относится всё, что связано с данными — как получить к ним доступ, как проверить их, какое у них поведение и отношение друг с другом.
- «Template» — шаблоны. Этот уровень содержит в себе всё, что связано с отображением: как именно что-то должно быть отображено на веб-странице или любом другом документе.
- «View» — представление. Этот уровень связан со всей логикой, связывающей модель и необходимые шаблоны. Вы можете себе представлять это уровень как мост между моделями и шаблонами.

Происходит разделение данных приложения, пользовательского интерфейса и логики на три отдельных компонента, поэтому изменение отдельного компонента может осуществляться независимо [17].

В результате анализа всех параметров для разработки веб-приложения был выбран фреймворк Django.

Выбор СУБД

По умолчанию в настройках указано использование SQLite. SQLite уже включен в Python, следовательно, не нужно дополнительно что-то устанавливать [3].

SQLite — компактная встраиваемая реляционная база данных. Имеет открытый исходный код библиотеки. SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а предоставляет библиотеку, с которой программа компонуется и движок становится составной частью программы. К недостаткам относятся Отсутствие пользовательского управления, то есть отсутствие возможности управлять связями в таблицах в соответствии с привилегиями и невозможность дополнительной настройки, то есть SQLite нельзя сделать более производительной [18].

Поэтому в качестве СУБД была выбрана реляционная база данных MySQL.

MySQL - это самая распространенная полноценная серверная СУБД. MySQL очень функциональная, свободно распространяемая СУБД, которая успешно работает с различными сайтами и веб приложениями.

К преимуществам данной базы данных относятся:

- Простота: MySQL легко устанавливается. Существует много сторонних инструментов, включая визуальные, облегчающих начало работы с БД.
- Много функций: MySQL поддерживает большую часть функционала SQL.
- Безопасность: в MySQL встроено много функций безопасности.
- Мощность и масштабируемость: MySQL может работать с действительно большими объёмами данных, и неплохо подходит для масштабируемых приложений [18].

2. РАЗРАБОТКА БАЗЫ ДАННЫХ

2.1. Разработка структуры базы данных

В данном разделе описан процесс разработки реляционной базы данных в СУБД MySQL. В процессе создания была проанализирована предоставленная институтом база данных, содержащая в себе среднемноголетние данные по активным станциям, такие как:

1. Приход солнечной радиации (суммарной, прямой, диффузной) на горизонтальную поверхность при средних условиях облачности:

- 1) Среднемесячные часовые суммы солнечной радиации
- 2) Среднемесячные суточные суммы солнечной радиации
- 3) Месячные суммы солнечной радиации

2. Среднемесячные коэффициенты отражательной способности поверхности – альбедо.

Всего общее количество наблюдаемых метеорологических станций – 166.

На основе полученной информации была проработана структура таблиц, а также взаимосвязи между ними. Основной целью было устранить дублирование и избыточность информации. Также важно было избежать противоречивости в хранимых данных.

Для достижения данных целей было необходимо провести нормализацию всех таблиц базы данных, то есть привести их к нормальным формам.

Нормализация – это процесс последовательной пошаговой модификации таблиц базы данных, имеющий целью сокращение избыточности и несогласованности. Процесс нормализации устроен таким образом, что по завершении его очередного шага нормализуемая база данных переходит в

определенную форму, называемую нормальной и связанную с этим шагом. Так, реляционная модель определяет три нормальных формы.

- первая нормальная форма – любое значение любого столбца таблицы является элементарным, то есть значение нельзя разделить на части без потери смысла, каждая строка имеет одинаковое количество столбцов, все строки в таблице должны быть разными;

- вторая нормальная форма – таблица должна удовлетворять требованиям первой нормальной формы и все ее поля, не входящие в первичный ключ, должны зависеть от полного первичного ключа, то есть таблица не имеет частичных функциональных зависимостей;

- третья нормальная форма – таблица должна удовлетворять требованиям второй нормальной формы и не должна содержать транзитных зависимостей, то есть все ее столбцы, не входящие в первичный ключ, должны зависеть от первичного ключа и не зависеть друг от друга.

При этом каждая последующая нормальная форма сильнее предыдущей в том смысле, что удовлетворяет критериям предыдущей нормальной формы [10].

Исходя из данных требований, были разработаны таблицы базы данных, которые рассмотрены в следующем разделе.

2.2. Разработка таблиц

Таблица Станции

Первой и основополагающей таблицей является таблица, содержащая необходимые данные по каждой из 166 станций, которые могут потребоваться для её описания.

Естественно, таблица должна иметь свой первичный ключ, а именно уникальный идентификатор `id`, который будет внешним ключом для остальных

таблиц. При помощи этого поля мы связываем все наши таблицы. Столбцу *id* соответствует номер конкретной станции.

Также для полного описания станции мы должны хранить в данной таблице название этой станции – «Name_Station», регион ее расположения – «Region», а также географические координаты ее расположения: широта – «Latitude» и долгота – «Longitude».

В Табл. 1 представлена структура таблицы «Станции».

Табл. 1

Таблица «Станции»

| Название переменной | Тип данных | Описание переменной |
|---------------------|--------------|---|
| <u>id</u> | int | Идентификатор станции, является уникальным (первичный ключ) |
| Name_Station | Varchar(50) | Название станции |
| Region | Varchar(50) | Регион расположения станции |
| Latitude | Decimal(5,2) | Широта |
| Longitude | Decimal(5,2) | Долгота |

Таблицы Среднемесячных часовых сумм суммарной, прямой и диффузной солнечной радиации на горизонтальную поверхность

Проанализируем структуру фрагмента таблицы Среднемесячных часовых сумм суммарной, прямой и диффузной солнечной радиации из предоставленной базы данных для конкретной станции, которая показана на Рис. 1.

| № | Станция | | | | | | | | | | | | Регион | | | | | | | | | | | | φ | ψ |
|------|---|------|------|------|------|------|------|------|------|------|-------|-------|-----------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|------|------|
| 1 | Гасан-Кули | | | | | | | | | | | | Туркмения | | | | | | | | | | | | 37.1 | 54.0 |
| Мес. | Часовой интервал (истинное солнечное время) | | | | | | | | | | | | | | | | | | | | | | | | Σ | |
| | 0-1 | 1-2 | 2-3 | 3-4 | 4-5 | 5-6 | 6-7 | 7-8 | 8-9 | 9-10 | 10-11 | 11-12 | 12-13 | 13-14 | 14-15 | 15-16 | 16-17 | 17-18 | 18-19 | 19-20 | 20-21 | 21-22 | 22-23 | 23-24 | | |
| I | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.03 | 0.15 | 0.26 | 0.35 | 0.40 | 0.40 | 0.35 | 0.26 | 0.15 | 0.03 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 2.4 |
| II | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.01 | 0.10 | 0.23 | 0.35 | 0.46 | 0.51 | 0.51 | 0.46 | 0.35 | 0.23 | 0.10 | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 3.3 |
| III | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.04 | 0.17 | 0.31 | 0.43 | 0.52 | 0.57 | 0.57 | 0.52 | 0.43 | 0.31 | 0.17 | 0.04 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 4.1 |
| IV | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.02 | 0.09 | 0.20 | 0.34 | 0.50 | 0.63 | 0.68 | 0.68 | 0.63 | 0.55 | 0.43 | 0.29 | 0.15 | 0.02 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 5.2 |
| V | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.05 | 0.17 | 0.33 | 0.48 | 0.63 | 0.77 | 0.82 | 0.82 | 0.77 | 0.67 | 0.53 | 0.38 | 0.21 | 0.06 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 6.7 |
| VI | 0.00 | 0.00 | 0.00 | 0.00 | 0.01 | 0.06 | 0.21 | 0.37 | 0.53 | 0.68 | 0.83 | 0.89 | 0.89 | 0.83 | 0.73 | 0.60 | 0.44 | 0.23 | 0.08 | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 7.4 |
| VII | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.06 | 0.18 | 0.33 | 0.48 | 0.64 | 0.78 | 0.84 | 0.84 | 0.80 | 0.70 | 0.57 | 0.39 | 0.23 | 0.08 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 6.9 |
| VIII | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.04 | 0.14 | 0.31 | 0.44 | 0.60 | 0.73 | 0.80 | 0.80 | 0.76 | 0.65 | 0.51 | 0.36 | 0.17 | 0.04 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 6.3 |
| IX | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.07 | 0.22 | 0.38 | 0.53 | 0.67 | 0.73 | 0.73 | 0.67 | 0.57 | 0.43 | 0.27 | 0.10 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 5.4 |
| X | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.01 | 0.14 | 0.31 | 0.42 | 0.55 | 0.60 | 0.60 | 0.55 | 0.42 | 0.31 | 0.14 | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 4.1 |
| XI | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.06 | 0.20 | 0.33 | 0.43 | 0.46 | 0.46 | 0.43 | 0.33 | 0.20 | 0.06 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 3.0 |
| XII | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.02 | 0.13 | 0.23 | 0.33 | 0.38 | 0.38 | 0.33 | 0.23 | 0.13 | 0.02 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 2.2 |
| Σ | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.2 | 0.9 | 2.3 | 4.0 | 5.6 | 7.0 | 7.7 | 7.7 | 7.1 | 5.9 | 4.4 | 2.6 | 1.2 | 0.3 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 57 |

Рис.1. Структура таблицы часовых сумм

Как было замечено, структура хранения данных здесь достаточно сложная. Можно наблюдать повторяющиеся данные, которые относятся к описанию станции. Это приводит к избыточности, так как вся эта информация уже хранится в таблице Stations. Естественно, в будущей таблице необходимо указать только id станции, чтобы идентифицировать наши данные для конкретной станции.

Общая структура исходной таблицы такова, что по строкам идут месяцы в году, а по столбцам часы в сутках. Для дальнейшего удобства работы с данными нам необходимо транспонировать данную таблицу, то есть поменять местами строки и столбцы.

Исходя из специфики хранения исходных данных, была разработана таблица с составным первичным ключом.

Первой частью ключа будет внешний ключ от таблицы Станции – «id», который является уникальным идентификатором станции. Благодаря этой части составного первичного ключа явно указываем, для какой конкретно станции храним данные. Второй частью ключа будет id часа в сутках - «ID-Hour». Полученное сочетание id станции и id часа в сутках даст уникальную комбинацию для хранения значений по месяцам в базе данных MySQL.

Остальные столбцы таблицы соответствуют каждому месяцу в году.

Рассмотрим структуру данных таблиц на примере среднемесячных часовых сумм суммарной солнечной радиации, которая предствалена в Табл. 2.

Табл. 2

Таблица « Среднемесячных часовых сумм суммарной солнечной радиации »

| Название переменной | Тип данных | Описание переменной |
|---------------------|--------------|--|
| <u>ID_Station</u> | int | Идентификатор станции (внешний ключ является первичным ключом) |
| <u>ID_Hour</u> | int | Номер часа в сутках |
| Month_1 | Decimal(5,2) | Здесь будут храниться суммы по всем часам для конкретного месяца |
| Month_2 | Decimal(5,2) | |
| Month_3 | Decimal(5,2) | |
| Month_4 | Decimal(5,2) | |
| Month_5 | Decimal(5,2) | |
| Month_6 | Decimal(5,2) | |
| Month_7 | Decimal(5,2) | |
| Month_8 | Decimal(5,2) | |
| Month_9 | Decimal(5,2) | |
| Month_10 | Decimal(5,2) | |
| Month_11 | Decimal(5,2) | |
| Month_12 | Decimal(5,2) | |

Таблицы Среднемесячных суточных сумм суммарной, прямой и диффузной солнечной радиации на горизонтальную поверхность

Проанализируем структуру фрагмента таблицы Среднемесячных суточных сумм суммарной/прямой/диффузной солнечной радиации из предоставленной базы данных для конкретной станции, которая показана на Рис. 2.

| № | Станция | Регион | φ | ψ | Месяц | | | | | | | | | | | | |
|---|-------------------------------|-----------------------|-------|------|-------|------|------|------|------|------|------|------|------|------|------|------|------|
| | | | град. | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | Сум. |
| 1 | Гасан-Кули | Туркмения | 37.1 | 54.0 | 2.37 | 3.29 | 4.06 | 5.19 | 6.67 | 7.37 | 6.94 | 6.33 | 5.36 | 4.07 | 2.96 | 2.19 | 56.8 |
| 2 | Курган-Тюбе | Курган-Тюбинская | 37.7 | 68.8 | 1.90 | 2.72 | 3.68 | 5.03 | 6.64 | 7.78 | 7.73 | 6.97 | 5.86 | 4.09 | 2.60 | 1.75 | 56.7 |
| 3 | Ашхабад | Ашхабадская | 38.0 | 58.3 | 2.03 | 2.82 | 3.54 | 4.97 | 6.47 | 7.57 | 7.52 | 6.96 | 5.77 | 4.02 | 2.55 | 1.75 | 56.0 |
| 4 | им. академика Н. П. Горбунова | Горно-Бадахшанская АО | 38.4 | 72.2 | 2.58 | 3.41 | 4.77 | 6.40 | 7.77 | 8.37 | 8.13 | 7.46 | 5.82 | 4.14 | 3.05 | 2.25 | 64.1 |
| 5 | Душанбе, АС | Таджикистан | 38.5 | 68.8 | 1.99 | 2.68 | 3.58 | 4.84 | 6.27 | 7.60 | 7.53 | 6.83 | 5.70 | 3.76 | 2.45 | 1.77 | 55.0 |
| 6 | Беквент | Красноводская | 38.7 | 56.0 | 2.42 | 3.32 | 4.02 | 5.17 | 6.47 | 7.46 | 7.08 | 6.66 | 5.58 | 4.11 | 2.79 | 2.19 | 57.3 |

Рис.2. Структура таблицы суточных сумм

Данные здесь так же повторяются, а именно: номер станции, название, регион, широта и долгота. Было не правильно переносить в базу данных таблицу в таком виде, поскольку мы получим огромную избыточность. Поэтому была разработана таблица, в которой первичным ключом является внешний ключ от таблицы «Stations» - «id», который является уникальным идентификатором станции. Остальные столбцы таблицы соответствуют месяцу в году.

Рассмотрим структуру данных таблиц на примере среднемесячных суточных сумм суммарной солнечной радиации, которая представлена в Табл. 3.

Табл. 3

Таблица «Среднемесячных суточных сумм суммарной солнечной радиации»

| Название переменной | Тип данных | Описание переменной |
|---------------------|--------------|--|
| <u>ID_Station</u> | int | Идентификатор станции (внешний ключ является первичным ключом) |
| Month_1 | Decimal(5,2) | Хранит данные для данной станции за январь |
| Month_2 | Decimal(5,2) | Хранит данные для данной станции за февраль |
| Month_3 | Decimal(5,2) | Хранит данные для данной станции за март |
| Month_4 | Decimal(5,2) | Хранит данные для данной станции за апрель |
| Month_5 | Decimal(5,2) | Хранит данные для данной станции за май |
| Month_6 | Decimal(5,2) | Хранит данные для данной станции за июнь |
| Month_7 | Decimal(5,2) | Хранит данные для данной станции за июль |
| Month_8 | Decimal(5,2) | Хранит данные для данной станции за август |
| Month_9 | Decimal(5,2) | Хранит данные для данной станции за сентябрь |
| Month_10 | Decimal(5,2) | Хранит данные для данной станции за октябрь |

Продолжение Табл. 3

| | | |
|----------|--------------|---|
| Month_11 | Decimal(5,2) | Хранит данные для данной станции за ноябрь |
| Month_12 | Decimal(5,2) | Хранит данные для данной станции за декабрь |

Таблицы Среднемесячных месячных сумм суммарной, прямой и диффузной солнечной радиации на горизонтальную поверхность

Проанализируем структуру фрагмента таблицы Среднемесячных месячных сумм суммарной, прямой и диффузной солнечной радиации из предоставленной базы данных для конкретной станции, которая показана на Рис. 3. Поскольку она аналогична таблице, представленной на Рис.2, то и структура формируемой таблицы будет похожей.

| № | Станция | Регион | φ | ψ | Месяц | | | | | | | | | | | | Год |
|---|-------------------------------|-----------------------|-------|------|-------|------|-------|-------|-------|-------|-------|-------|-------|-------|------|------|------|
| | | | град. | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | |
| 1 | Гаспи-Кули | Туркмения | 37.1 | 54.0 | 73.4 | 92.2 | 125.7 | 155.6 | 206.8 | 221.2 | 215.1 | 196.3 | 160.8 | 126.1 | 88.7 | 67.9 | 1730 |
| 2 | Курган-Тюбе | Курган-Тюбинская | 37.7 | 68.8 | 58.8 | 76.1 | 113.9 | 150.9 | 205.7 | 233.4 | 239.7 | 216.1 | 175.8 | 126.8 | 78.0 | 54.3 | 1729 |
| 3 | Ашхабад | Ашхабадская | 38.0 | 58.3 | 62.9 | 79.0 | 109.7 | 149.0 | 200.6 | 227.1 | 233.2 | 215.9 | 173.2 | 124.7 | 76.5 | 54.3 | 1706 |
| 4 | им. академика Н. П. Горбунова | Горно-Бадахшанская АО | 38.4 | 72.2 | 79.8 | 95.4 | 147.9 | 192.1 | 240.8 | 251.1 | 251.9 | 231.3 | 174.6 | 128.3 | 91.6 | 69.8 | 1955 |
| 5 | Душанбе, АС | Таджикистан | 38.5 | 68.8 | 61.6 | 75.1 | 111.1 | 145.2 | 194.4 | 227.9 | 233.4 | 211.6 | 171.1 | 116.4 | 73.6 | 54.8 | 1676 |
| 6 | Бекмбент | Красноводская | 38.7 | 56.0 | 74.9 | 92.9 | 124.7 | 155.2 | 200.5 | 223.8 | 219.4 | 206.4 | 167.3 | 127.3 | 83.8 | 67.9 | 1744 |

Рис.3. Структура таблицы месячных сумм

Была разработана таблица, в которой первичным ключом будет внешний ключ от таблицы «Stations», а именно уникальный идентификатор станции. Остальные столбцы таблицы соответствуют месяцу в году.

Рассмотрим структуру данных таблиц на примере среднемесячных часовых сумм суммарной солнечной радиации, которая представлена в Табл. 4.

Табл. 4

Таблица «Среднемесячных месячных сумм суммарной солнечной радиации»

| Название переменной | Тип данных | Описание переменной |
|---------------------|--------------|--|
| <u>ID_Station</u> | int | Идентификатор станции (внешний ключ является первичным ключом) |
| Month_1 | Decimal(5,2) | Хранит данные для данной станции за январь |
| Month_2 | Decimal(5,2) | Хранит данные для данной станции за февраль |
| Month_3 | Decimal(5,2) | Хранит данные для данной станции за март |
| Month_4 | Decimal(5,2) | Хранит данные для данной станции за апрель |
| Month_5 | Decimal(5,2) | Хранит данные для данной станции за май |
| Month_6 | Decimal(5,2) | Хранит данные для данной станции за июнь |
| Month_7 | Decimal(5,2) | Хранит данные для данной станции за июль |
| Month_8 | Decimal(5,2) | Хранит данные для данной станции за август |
| Month_9 | Decimal(5,2) | Хранит данные для данной станции за сентябрь |
| Month_10 | Decimal(5,2) | Хранит данные для данной станции за октябрь |
| Month_11 | Decimal(5,2) | Хранит данные для данной станции за ноябрь |
| Month_12 | Decimal(5,2) | Хранит данные для данной станции за декабрь |

Таблица Альбедо

Проанализируем структуру фрагмента таблицы из предоставленной базы данных для конкретной станции, которая показана на Рис. 4. Таблица содержит среднемесячные коэффициенты отражающей способности поверхности – альбедо (о.е.) для всех станций.

| № | Станция | Регион | Ф | ψ | Месяц | | | | | | | | | | | | Год |
|---|-------------------------------|-----------------------|-------|------|-------|------|------|------|------|------|------|------|------|------|------|------|------|
| | | | град. | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | |
| 1 | Гасан-Кули | Туркмения | 37,1 | 54,0 | 0.28 | 0.28 | 0.27 | 0.28 | 0.29 | 0.30 | 0.30 | 0.30 | 0.30 | 0.28 | 0.27 | 0.29 | |
| 2 | Курган-Тюбе | Курган-Тюбинская | 37,7 | 68,8 | 0.33 | 0.28 | 0.21 | 0.23 | 0.22 | 0.23 | 0.25 | 0.25 | 0.25 | 0.26 | 0.24 | 0.25 | 0.25 |
| 3 | Ашхабад | Ашхабадская | 38,0 | 58,3 | 0.32 | 0.29 | 0.23 | 0.23 | 0.23 | 0.25 | 0.27 | 0.28 | 0.29 | 0.28 | 0.25 | 0.27 | 0.27 |
| 4 | им. академика Н. П. Горбунова | Горно-Бадахшанская АО | 38,4 | 72,2 | 0.70 | 0.73 | 0.76 | 0.76 | 0.73 | 0.51 | 0.23 | 0.19 | 0.20 | 0.42 | 0.60 | 0.68 | 0.54 |
| 5 | Душанбе, АС | Таджикистан | 38,5 | 68,8 | 0.38 | 0.31 | 0.21 | 0.21 | 0.21 | 0.21 | 0.22 | 0.23 | 0.24 | 0.23 | 0.25 | 0.27 | 0.25 |
| 6 | Бекибевт | Красноводская | 38,7 | 56,0 | 0.33 | 0.31 | 0.27 | 0.28 | 0.29 | 0.29 | 0.29 | 0.30 | 0.31 | 0.31 | 0.31 | 0.30 | 0.30 |

Рис.4. Структура таблицы Альбедо

В разработывной таблице, первичным ключом будет внешний ключ от таблицы «Stations», то есть уникальный идентификатор станции. Остальные столбцы таблицы соответствуют месяцу в году.

В Табл. 5 представлена структура таблицы Альбедо.

Табл. 5

Таблица «Альбедо»

| Название переменной | Тип данных | Описание переменной |
|---------------------|--------------|--|
| <u>ID</u> | int | Идентификатор станции (внешний ключ является первичным ключом) |
| Month_1 | Decimal(5,2) | Хранит данные для данной станции за январь |
| Month_2 | Decimal(5,2) | Хранит данные для данной станции за февраль |
| Month_3 | Decimal(5,2) | Хранит данные для данной станции за март |
| Month_4 | Decimal(5,2) | Хранит данные для данной станции за апрель |
| Month_5 | Decimal(5,2) | Хранит данные для данной станции за май |
| Month_6 | Decimal(5,2) | Хранит данные для данной станции за июнь |
| Month_7 | Decimal(5,2) | Хранит данные для данной станции за июль |
| Month_8 | Decimal(5,2) | Хранит данные для данной станции за август |
| Month_9 | Decimal(5,2) | Хранит данные для данной станции за сентябрь |
| Month_10 | Decimal(5,2) | Хранит данные для данной станции за октябрь |
| Month_11 | Decimal(5,2) | Хранит данные для данной станции за ноябрь |
| Month_12 | Decimal(5,2) | Хранит данные для данной станции за декабрь |

2.3. Соединение веб-приложения с базой данных

Серверное приложение разрабатывается с помощью фреймворка Django 3.1.6 на языке Python 3.9. Поскольку Django по умолчанию работает с СУБД SQLite, то для работы с другими СУБД необходимо прописать настройках проекта (файл «settings.py»), с какой базой данных необходимо работать и как к ней подключаться (наименование базы данных, логин, пароль, хост, порт). После соединения проекта Django с базой данных можно приступить к созданию моделей в проекте и с помощью миграций создавать или изменять реальные таблицы базы данных.

Модели в Django описывают структуру используемых данных. Используемые в программе данные хранятся в базах данных, и с помощью моделей как раз осуществляется взаимодействие с базой данных. Как правило, каждая модель отображается в одну таблицу базы данных [3, 4].

При создании приложения по умолчанию в его каталог добавляется файл «models.py», который применяется для определения моделей. Модель представляет класс, унаследованный от `django.db.models.Model` [4].

В файле «models.py» определили модели, поля которых аналогичны столбцам таблиц, которые были разработаны ранее. Далее необходимо осуществить миграции, то есть преобразовать базу данных в соответствии с определением моделей. Миграции осуществляются при помощи двух команд: `makemigrations` и `migrate`. Первая команда отвечает за создание миграции, а вторая соответственно за само выполнение миграции. После выполнения миграций в базе данных появится ряд таблиц, которые будут иметь название по имени приложения и модели. Помимо этого появятся также еще и служебные таблицы.

2.4. Структура базы данных

В базе данных все таблицы связаны только с основной таблицей «mainapp_stations». Они не связаны между собой, потому что таблицы не содержат взаимосвязанных данных.

Таблицы, содержащие данные о суточных и месячных суммах суммарной, прямой и диффузной солнечной радиации, а также таблица, содержащая среднемесячные коэффициенты отражающей способности поверхности – альбедо, имеют связь с основной таблицей «один к одному», поскольку каждой конкретной станции соответствует одна строка в таблице.

С таблицами, содержащими данные о часовых суммах суммарной, прямой и диффузной солнечной радиации, немного другой случай. Одной станции соответствуют сразу несколько строк в таблице. Данная ситуация складывается из-за первичного ключа, который состоит из id станции и id часа, то есть мы получаем 24 строки для каждой станции для каждого часа в сутках. Поэтому здесь будет связь «один ко многим». Структура полученной базы данных, с указанием связей, представлена на Рис. 5. В структуре не показаны служебные таблицы, поскольку в проекте они не используются.

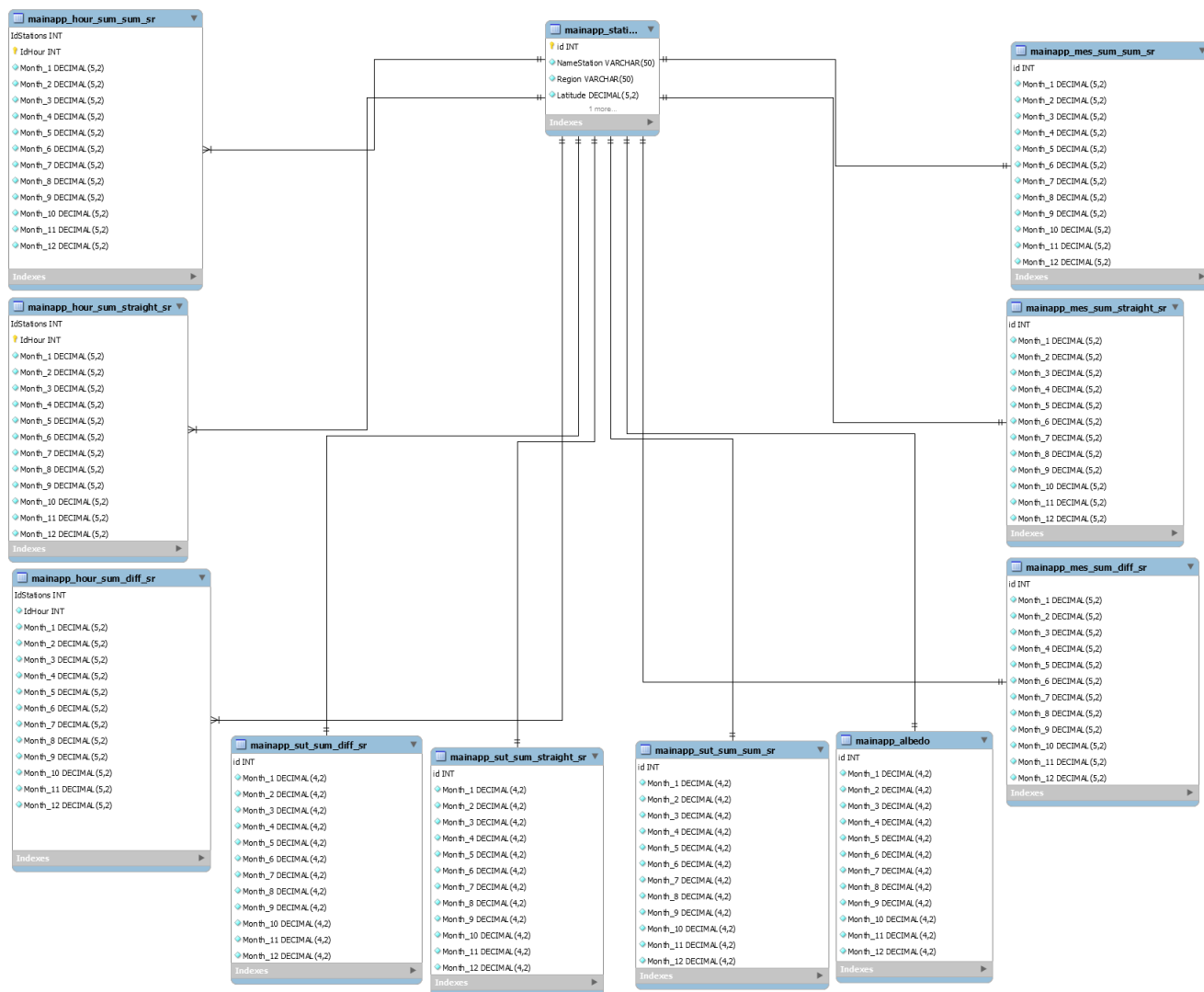


Рис.5. Структура базы данных в СУБД MySQL

3. РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ

3.1. Архитектура разрабатываемого web-приложения

Веб-приложение разрабатывалось по архитектуре «клиент-сервер». Из названия понятно, что в данной концепции участвуют две стороны: клиент и сервер.

Клиент – локальный компьютер на стороне виртуального пользователя, который выполняет отправку запроса к серверу для возможности предоставления данных или выполнения определенной группы системных действий.

Сервер – очень мощный компьютер или специальное системное оборудование, которое предназначается для разрешения определенного круга задач по процессу выполнения программных кодов. Он выполняет работы сервисного обслуживания по клиентским запросам, предоставляет пользователям доступ к определенным системным ресурсам, сохраняет данные или БД [11].

Обычно клиент нужен для обеспечения красивого интерфейса, который был бы удобен для конечного пользователя, а сервер для проведения основных вычислений и хранения информации.

Также стоит заметить, что в основе взаимодействия клиент-сервер лежит принцип того, что такое взаимодействие начинает клиент, сервер лишь отвечает клиенту и сообщает о том, может ли он предоставить услугу клиенту и если может, то на каких условиях. Веб-сайты являются частным случаем такой архитектуры, когда общение между клиентом и сервером происходит через сеть Интернет по протоколу HTTP. Роль клиентской части выполняет браузер, который реализует пользовательский интерфейс, формирует запросы к серверу и обрабатывает ответы от него [7, 12].

Серверная часть получает запрос от клиента, выполняет вычисления, после этого формирует веб-страницу и отправляет её клиенту по сети с использованием протокола HTTP (протокол передачи гипертекста).

Данная концепция взаимодействия была разработана в первую очередь для того, чтобы разделить нагрузку между участниками процесса обмена информацией, а также для того, чтобы разделить программный код поставщика и заказчика [12].

3.2. Структура разрабатываемого web-приложения

Одним из преимуществ фреймворка Django является то, что проект всегда имеет конкретный каркас, то есть имеется некоторый набор папок и файлов, которые содержат в себе все основные составляющие проекта. Рассмотрим ниже файловую структуру разработанного веб-приложения.

MainProject/

 MainApp /

 migrations/

 static/

 templates/

 __init__.py

 admin.py

 apps.py

 forms.py

 models.py

 tests.py

 urls.py

```
view.py
MainProject /
    __init__.py
    asgi.py
    setting.py
    urls.py
    wsgi.py
venv/
manage.py
```

Рассмотрим назначение основных компонент проекта ниже.

MainProject/ – внешний корневой каталог, контейнер для нашего проекта.

MainProject/ – папка проекта.

__init__.py – файл , который сообщает Python, что этот каталог должен рассматриваться как пакет Python'а.

asgi.py - точка входа для ASGI-совместимых веб-серверов для обслуживания проекта.

setting.py – файл содержит в себе все настройки проекта. Здесь регистрируем приложения, задаём размещение статичных файлов, настройки базы данных и так далее.

urls.py – файл, который задаёт ассоциации URL адресов с представлениями. Несмотря на то, что этот файл может содержать *все* настройки url, обычно его делят на части, по одной на приложение

wsgi.py – файл, использующийся для налаживания связи между вашим Django приложением и веб-сервером.

MyApp/ - папка приложения.

migrations/ - папка используется, чтобы хранить «миграции» — файлы, которые позволяют вам автоматически обновлять базу данных по мере изменения моделей.

static/ — папка, содержащая в себе статические файлы, необходимые для работы веб-приложения.

__init__.py — файл указывает, что приложение является пакетом Python.

admin.py — файл, хранящий настройки интерфейса администратора приложения.

apps.py - файл определения информации о приложении. В нем создается класс AppConfig, который используется для определения метаданных, таких как имя приложения.

tests.py — файл используется для проверки файлов кода.

forms.py — файл, используемый для описания форм приложения.

models.py — файл, используемый для определения моделей приложения.

MyApp/ urls.py — файл, который задает ассоциации URL адресов с представлениями в приложении.

view.py — файл, в котором содержатся все выполняемые в приложении представления.

template/ - папка, включающая в себя шаблоны HTML-страниц.

venv/ - виртуальное окружение: все подключаемые внешние модули и библиотеки.

manage.py – скрипт, используемый для взаимодействием с проектом, при его помощи можно создаются новые приложения, работать с базами данных и запускать отладочный сервер [3, 4].

Функции, используемые в веб-приложении, которые являются содержимым файла «views.py», представлены в Табл. 6.

Табл. 6

Описание функций в файле «views.py»

| Название | Входные параметры | Назначение | Отображаемый шаблон |
|-----------------|--------------------------|---|----------------------------|
| index | request | Отображение шаблона главной страницы | Index.html |
| ShowStations | request | Отображение шаблона со списком всех станций | ShowStations.html |
| FindName | request | Отображение шаблона для поиска станции по названию | FindName.html |
| FindNumber | request | Отображение шаблона для поиска станции по номеру | FindNumber.html |
| FindCoord | request | Отображение шаблона для поиска станции по географическим координатам | FindCoord.html |
| SearchResult | request | Формирование шаблона с результатами поиска | SearchResult.html |
| Export_csv | request | Экспорт выбранных данных в формате csv | Export_csv.html |
| Show | request | Отображение шаблона с тремя ближайшими станциями для выбора | Show.html |
| Export_page | request | Отображение шаблона для задания параметров необходимых для интерполирования | Export_page.html |
| Export_csv_intr | request | Экспорт интерполированных данных в формате csv | - |

Параметр «request» - это объект HttpRequest, содержащий данные о запросе.

3.3. Разработка алгоритмов поиска

В разрабатываемом приложении для удобства использования необходимо реализовать несколько видов поиска:

- Поиск по названию
- Поиск по номеру станции
- Поиск по географическим координатам

Поиск по названию

Изначально необходимо считать введенные данные из формы. Форма состоит всего из одного поля, в которое пользователю необходимо ввести название интересующей станции. Для дальнейшего поиска необходимо передать данные на сервер при помощи метода GET. Метод используется по причине того, что нам необходимо осуществить простой поиск по базе данных без внесения изменений в ресурс.

Из формы мы получаем название станции. Изначально нужно проверить, есть ли в базе данных вообще станция с таким названием. Если такая станция есть, то продолжаем работу, если нет, то нужно вывести предупреждение, что такой станции нет и данные не могут быть предоставлены.

Осуществлять поиск по базе данных удобнее по id станции, а не по названию, поскольку только таблица *Stations* хранит название станции, а все остальные только id. Чтобы получить id станции по ее названию используем как раз основную таблицу *Stations*, которая хранит в себе основную информацию по каждой станции. Далее нам необходимо обратиться к базе данных.

Веб-приложения Django получают доступ и управляют данными через объекты Python, называемые моделями. Обращаясь к модели, мы получаем записи из нашей базы данных как объект QuerySet [2 – 4].

QuerySet — это итерируемый объект из базы данных, означающий, что он содержит несколько объектов, которые мы можем перебирать / прокручивать. Объект также может использовать фильтры для ограничения результатов [2 – 4].

Каждая модель Django имеет как минимум один менеджер, а менеджер по умолчанию называется objects. Сделать запрос к объекту (QuerySet) можно с помощью менеджера модели [2 – 4].

Ниже приведен фрагмент кода получения id станции по ее названию.

```
id_st = stations.objects.select_related('id').filter(NameStation=search_query)[:1]
```

search_query – переменная, которая хранит полученное из формы при помощи метода GET название станции.

Применяем метод filter() к QuerySet, чтобы получить объекты для нужной станции. Методом select_related достаем именно id.

Получив id станции, обращаемся к базе данных и получаем данные со всех таблиц по интересующей станции.

Ниже приведен пример кода обращения к таблицам в базе данных, содержащим среднемесячные часовые значения суммарной, прямой и диффузной солнечной радиации.

```
hour_sum = mainapp_hour_sum_sum_sr.objects.filter(IdStations=id_st)
```

```
hour_straight = mainapp_hour_sum_straight_sr.objects.filter(IdStations=id_st)
```

```
hour_diff = mainapp_hour_sum_diff_sr.objects.filter(IdStations=id_st)
```

Получив данные, в конечном итоге, их нужно отобразить HTML - шаблоне.

```
return render(request, 'MainApp/SearchResult.html', context= context)
```

Используем функцию render() для того, чтобы создать HttpResponse, который будет отправлен назад браузеру из нашего представления. Эта функция является «ярлыком», она создаёт HTML-файл, комбинируя указанный HTML-

шаблон и некоторые данные для вставки в шаблон (предоставляется в переменной с именем «context») [4].

Содержание переменной context представлено в Табл. 7.

Табл. 7

Содержимое переменной *context*

| Название поля | Описание поля |
|-------------------|--|
| res | Хранит объект с информацией по станции |
| res_mes_sum | Хранит объект со среднемесячными месячными значениями суммарной слонечной радиации |
| res_mes_straight | Хранит объект со среднемесячными месячными значениями прямой слонечной радиации |
| res_mes_diff | Хранит объект со среднемесячными месячными значениями диффузной слонечной радиации |
| res_sut_sum | Хранит объект со среднемесячными суточными значениями суммарной слонечной радиации |
| res_sut_straight | Хранит объект со среднемесячными суточными значениями прямой слонечной радиации |
| res_sut_diff | Хранит объект со среднемесячными суточными значениями диффузной слонечной радиации |
| res_hour_sum | Хранит объект со среднемесячными часовыми значениями суммарной слонечной радиации |
| res_hour_straight | Хранит объект со среднемесячными часовыми значениями прямой слонечной радиации |
| res_hour_diff | Хранит объект со среднемесячными часовыми значениями диффузной слонечной радиации |
| albedo | Хранит объект со среднемесячными коэффициентами отражающей способности поверхности – альбедо |
| error | Хранит текст ошибки |

Алгоритм поиска по названию станции представлен на Рис.6.

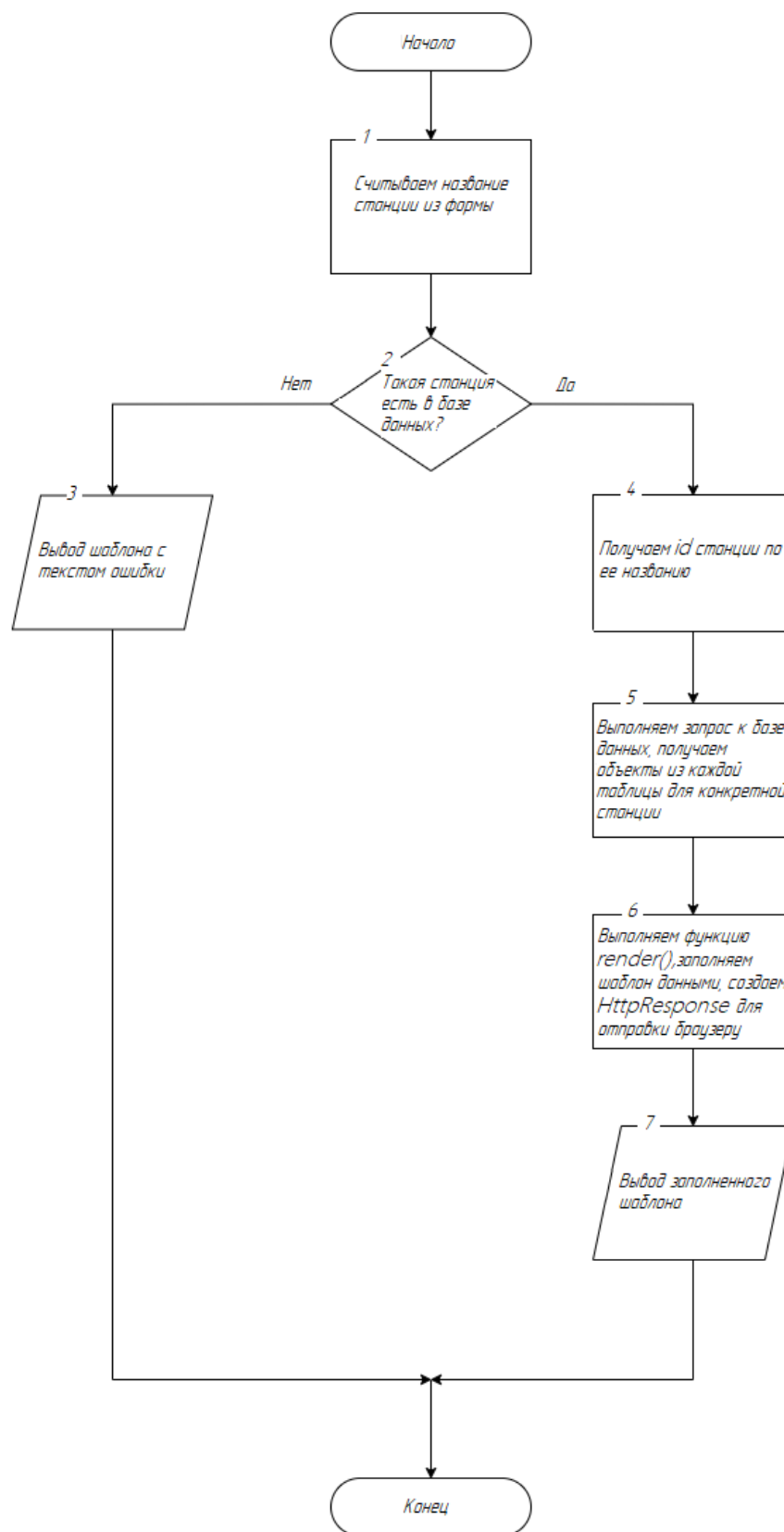


Рис.6. Алгоритм поиска станции по названию

Поиск по номеру

Данный тип поиска практически аналогичен поиску по названию. Отличие состоит в том, что пользователь указывает в поле формы номер станции, что эквивалентно ее id. Таким образом, мы минуем этап поиска id. Также отсутствует проверка на наличие станции в базе данных. Поскольку нам изначально известно общее количество станций, то установив максимальное и минимальное значение для поля ввода формы, мы производим валидацию данных со стороны клиента. Некорректные данные в принципе уйти на сервер не смогут.

Мы также считываем номер станции из формы. Форма состоит из одного поля, в которое пользователю предложено ввести номер станции. Данные передаются на сервер методом GET. На данном этапе мы считаем, что id – это номер введенной станции.

Далее осуществляем запрос к базе данных. Как результат получаем объекты QuerySet со всех таблиц по конкретной станции, отфильтрованные по id.

Полученные данные отображаем в HTML – шаблоне.

Алгоритм поиска по номеру станции представлен на Рис.7.

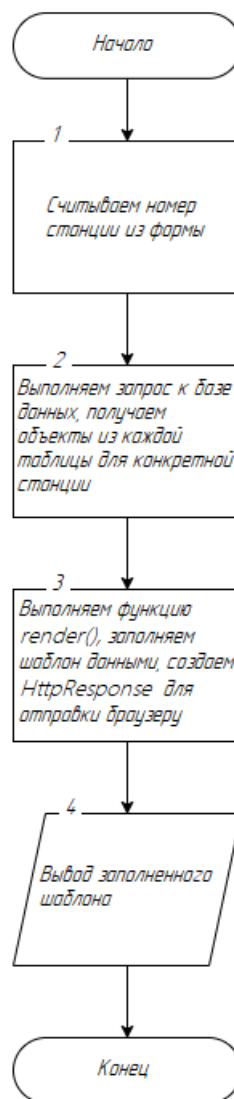


Рис.7. Алгоритм поиска станции по номеру

Поиск по географическим координатам

Поиск по географическим координатам проходит в два этапа: задание пользователем широты и долготы и выбор одной из трех предложенных ближайших к указанной точке станций.

Пользовательская форма для поиска по географическим координатам имеет два поля ввода. Пользователю предложено указать долготу и широту точки расположения станции. Поля ввода также проверяются на валидность со стороны клиента, следовательно, никаких дополнительных проверок со стороны сервера не требуется.

Изначально считываем из формы широту и долготу и передаем на сервер посредством метода GET. Далее рассчитываем расстояния от указанной пользователем точки до каждой станции. Результатом будут три ближайшие станции. Расчет производился при помощи библиотеки «геору» [6]. Пример кода расчета расстояния показан ниже.

```
dist = distance.distance(coords_1, coords_2).km
```

Функция `distance.distance` рассчитывает расстояние между двумя точками, используя геодезическое расстояние. Геодезическое расстояние - это кратчайшее расстояние на поверхности эллипсоидальной модели Земли [6]. Функция принимает два параметра: географические координаты заданной точки (`coords_1`) и географические координаты станции (`coords_2`). Параметр `.km` означает, что расстояния будут выражаться в километрах. При расчете не учитывается высота расположения точек.

Расстояния рассчитываются для каждой станции по порядку, а результат заносится в массив по индексу, который соответствует номеру станции. Далее сортируем полученный массив по значению расстояния в порядке возрастания.

Нас интересуют индексы первых трех элементов массива, поскольку это и будут три ближайшие станции.

Далее отображаем HTML – шаблон, на котором выводим по отобранным трем станциям информацию в виде формы, для предоставления пользователю возможности выбора нужной станции. В зависимости от того, какой выбор сделает пользователь, нажав на кнопку, соответствующую одной из трех представленных станций, на сервер отправится id этой станции. На сервере при помощи GET запроса мы получаем id и делаем запрос к базе данных для отбора информации. Как результат обращения получаем объекты QuerySet, отфильтрованные по id. Полученные данные отображаем в HTML – шаблоне.

Алгоритм поиска по географическим координатам представлен на Рис. 8.

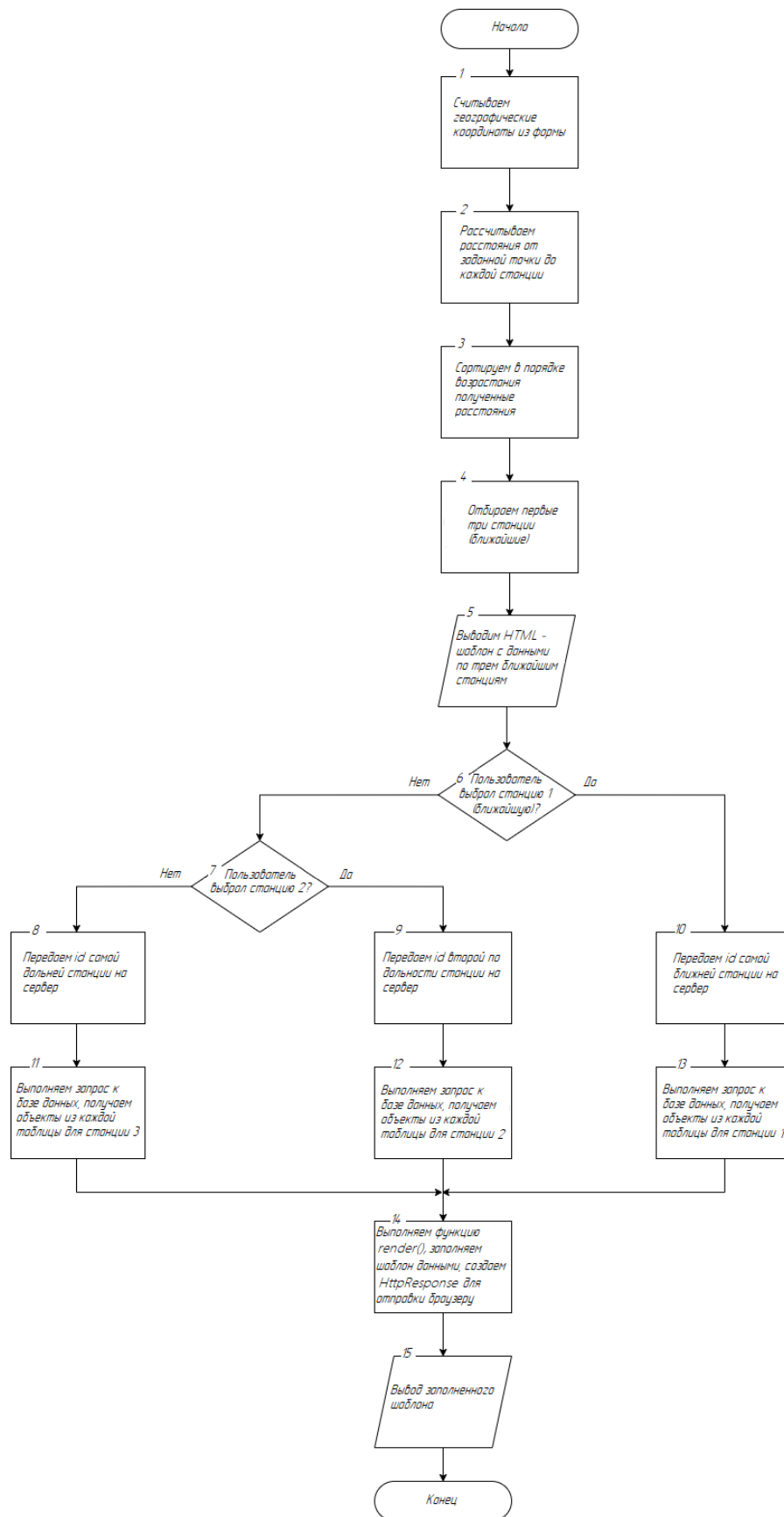


Рис.8. Схема поиска по географическим координатам

3.4. Разработка алгоритмов интерполяции данных

База данных содержит в себе среднемесячные часовые, суточные и месячные суммы прихода солнечной радиации на горизонтальную поверхность. Однако в ряде случаев предоставить данные пользователю необходимо не в виде средних значений за месяц, а за каждый час месяца. Исходя из этого, появляется потребность в интерполировании данных.

Интерполировать данные предстоит двумя способами:

1. Интерполяция по принципу – “День сурка”;
2. Интерполяция по принципу – “Линейная”;

Указанные способы интерполяции были заданы институтом ГВИЭ, поскольку они чаще всего применяются на практике.

3.4.1. Интерполяция “День сурка”

Описание принципа интерполяции типа “День сурка”

Суть интерполяции по принципу «День сурка» состоит в том, что имеющееся среднемесячное значение прихода солнечной радиации для некоторого месяца мы просто повторяем для каждого дня данного месяца. Никакой дополнительной обработки над данными не производится. Считается, что каждый день месяца было именно такое значение. Данный подход достаточно сильно упрощает процесс интерполяции, но при этом точность получаемой информации падает.

Описание алгоритма интерполяции – «День сурка»

На вход алгоритма поступает QuerySet, который мы фильтруем при помощи метода filter(), отбирая данные для конкретной станции. Фильтрация идет по id станции, который мы получили ранее при поиске.

Объект QuerySet будет содержать данные для конкретной станции:

id, Month_1, Month_2, Month_3, Month_4, Month_5, Month_6, Month_7, Month_8, Month_9, Month_10, Month_11, Month_12.

Для перебора данных необходимо сериализовать объект. Сериализация — это преобразование списка или словаря в JSON-строку.

object_list – это объект, содержащий в себе сериализованные данные, содержащиеся в QuerySet, но в другом формате представления.

Сериализация — это преобразование списка или словаря в JSON-строку.

Формат данных, хранящихся в *object_list*:

```
{'model': 'hour_sum_sum_sr', 'pk': 121, 'fields': {'IdStations': 6, 'IdHour': 1, 'Month_1': Decimal('0.00'), 'Month_2': Decimal('0.00'), 'Month_3': Decimal('0.00'), 'Month_4': Decimal('0.00'), 'Month_5': Decimal('0.00'), 'Month_6': Decimal('0.00'), 'Month_7': Decimal('0.00'), 'Month_8': Decimal('0.00'), 'Month_9': Decimal('0.00'), 'Month_10': Decimal('0.00'), 'Month_11': Decimal('0.00'), 'Month_12': Decimal('0.00')}}}
```

Как результат получаем словарь – коллекцию объектов с доступом по ключу, который удобно итерировать.

Далее необходимо заполнить массивы данных, с помощью которых мы будем производить интерполяцию. Тип хранимых данных и назначение каждого массива описаны в Табл. 8.

Табл. 8

Описание массивов данных

| Название переменной | Тип хранимых данных | Описание переменной |
|---------------------|---------------------|---|
| a11 | Float(4,2) | Массив, хранящий значения за 15-ое января, за 24 часа |
| a12 | Float(4,2) | Массив, хранящий значения за 15-ое февраля, за 24 часа |
| a13 | Float(4,2) | Массив, хранящий значения за 15-ое марта, за 24 часа |
| a14 | Float(4,2) | Массив, хранящий значения за 15-ое апреля, за 24 часа |
| a15 | Float(4,2) | Массив, хранящий значения за 15-ое мая, за 24 часа |
| a16 | Float(4,2) | Массив, хранящий значения за 15-ое июня, за 24 часа |
| a17 | Float(4,2) | Массив, хранящий значения за 15-ое июля, за 24 часа |
| a18 | Float(4,2) | Массив, хранящий значения за 15-ое августа, за 24 часа |
| a19 | Float(4,2) | Массив, хранящий значения за 15-ое сентября, за 24 часа |
| a110 | Float(4,2) | Массив, хранящий значения за 15-ое октября, за 24 часа |
| a111 | Float(4,2) | Массив, хранящий значения за 15-ое ноября, за 24 часа |
| a112 | Float(4,2) | Массив, хранящий значения за 15-ое декабря, за 24 часа |
| a | Float(4,2) | Выходной массив, содержащий всю выходную последовательность |

Берем значения из `object_list` по ключу `fields`. Значению по ключу `fields` соответствует также словарь, который хранит в себе данные о станции и месячные значения для этой станции. Основная работа будет производится именно с этими данными.

В цикле перебираем поля каждого месяца по `field_name` (ключ - название поля) и записываем значение `field_value` в массив, соответствующий данному месяцу. Пример кода представлен в Приложении 1.

После того, как массивы заполняются исходными данными, начинается процесс интерполяции. Суть состоит в том, что формируется выходной массив как объединение всех массивов, каждый из которых повторяется некоторое

количество раз, в соответствии с количеством дней в месяце. Пример кода получения выходного массива показан ниже.

$$a = a_{11} \cdot 31 + a_{12} \cdot 28 + a_{13} \cdot 31 + a_{14} \cdot 30 + a_{15} \cdot 31 + a_{16} \cdot 30 + a_{17} \cdot 31 + a_{18} \cdot 31 + a_{19} \cdot 30 + a_{110} \cdot 31 + a_{111} \cdot 30 + a_{112} \cdot 31$$

Общее количество элементов в массиве $a = 8760$, что соответствует количеству часов в году.

3.4.2. Интерполяция “Линейная”

Описание принципа интерполяции типа “Линейная”

«Линейная интерполяция» в данном случае отличается от общепринятого понятия, поскольку выполняется она совсем другим способом. Описанный ниже процесс называется линейной интерполяцией в солнечной энергетике.

Изначально мы имеем только средние за месяц значения прихода солнечной радиации. Затем мы предполагаем, что эти значения приходятся на конкретное число месяца – на 15-ое. Теперь мы имеем точки, приходящиеся на середину каждого месяца, которые нам нужно «соединить».

Для того чтобы соединить две соседние точки, мы должны найти шаг, с которым мы будем идти. Находим мы его по формуле:

$$\frac{(b-a)}{n}, \text{ где} \tag{1}$$

a – значение в месяце;

b – значение в месяце, который идет следующим за « a »;

n – количество дней между 15-ым числом одного месяца и 15-ым числом следующего;

Здесь необходимо учитывать, что от месяца к месяцу n будет различным, поскольку в месяце может быть и 30, и 31, и 28 дней (считается, что год невисокосный). От этого возникают дополнительные условия.

Также необходимо найти разность между этими двумя значениями, чтобы понять, увеличивается ли приход солнечной радиации в следующем месяце или наоборот уменьшается.

- Если разность меньше нуля, то делаем вывод, что приход солнечной радиации уменьшается и, следовательно, необходимо уменьшать значение «а».

- Если разность больше нуля, то приход солнечной радиации, наоборот, увеличивается и, следовательно, нужно увеличивать «а».

- Если разность равна ноль, то приход солнечной радиации не изменился, поэтому значения на протяжении всего интервала будут одинаковыми.

Процесс интерполяции – это не что иное, как последовательное увеличение (уменьшение) значения «а» на полученный шаг.

В идеальном случае при таком последовательном увеличении (уменьшении) величины «а» на шаге n мы должны получить значение равное исходному значению «b».

Такие действия повторяются для каждого часа месяца.

Таким образом, мы «восстанавливаем» неизвестные нам значения, находящиеся в промежутке между 15-ым числом одного месяца и 15-ым числом следующего.

Описание алгоритма интерполяции – «Линейная»

Исходные данные аналогичны интерполяции по принципу «День сурка», то есть работаем с массивами: $a11, a12, a13, a14, a15, a16, a17, a18, a19, a110, a111, a112$. Массивы заполнены среднемесячными значениями прихода солнечной радиации для конкретного месяца.

Для каждого промежутка (от 15-го числа одного месяца до 15 числа следующего месяца) мы находим разность и запоминаем ее в переменной *raz*. Сравниваем ее с нулем, чтобы оценить увеличивается ли приход солнечной радиации или нет. Находим шаг дискретизации по формуле (1) и запоминаем в переменной *delt*. Для каждого промежутка знаменатель в формуле будет меняться, поскольку это зависит от количества дней в месяце. Сохраняем в переменную *nach* - начало промежутка. Далее в зависимости от того, положительна разность или отрицательна, мы в цикле, длительность которого зависит от количества дней между двумя точками, либо прибавляем *delt*, либо вычитаем из переменной *nach*. Если разность равна нулю, то прибавляем *delt* к *nach*, значение от этого не изменится. Каждое полученное значение записываем в выходные массивы. Тип вышеупомянутых переменных и их значения описаны в Табл. 9.

Табл. 9

Описание переменных, используемых при интерполировании

| Название переменной | Тип переменной | Значение переменной |
|---------------------|----------------|---|
| raz | Float(4,2) | Разность между начальным и конечным значением |
| nach | Float(4,2) | Значение начала периода |
| delt | Float(4,2) | Шаг дискретизации |

Процесс повторяется для каждого промежутка 24 раза, по количеству часов в сутках.

Таким образом, заполняются выходные массивы значениями, полученными в результате интерполяции. Тип хранимых данных и назначение массивов описываются в Табл. 10.

Табл. 10

Описание выходных массивов

| Название переменной | Тип хранимых данных | Значение переменной |
|---------------------|---------------------|--|
| a21 | Float(4,2) | Массив, хранящий интерполированные значения для промежутка с 15-го января по 15-ое февраля |
| a22 | Float(4,2) | Массив, хранящий интерполированные значения для промежутка с 15-го февраля по 15-ое марта |
| a23 | Float(4,2) | Массив, хранящий интерполированные значения для промежутка с 15-го марта по 15-ое апреля |
| a24 | Float(4,2) | Массив, хранящий интерполированные значения для промежутка с 15-го апреля по 15-ое мая |
| a25 | Float(4,2) | Массив, хранящий интерполированные значения для промежутка с 15-го мая по 15-ое июня |
| a26 | Float(4,2) | Массив, хранящий интерполированные значения для промежутка с 15-го июня по 15-ое июля |
| a27 | Float(4,2) | Массив, хранящий интерполированные значения для промежутка с 15-го июля по 15-ое августа |
| a28 | Float(4,2) | Массив, хранящий интерполированные значения для промежутка с 15-го августа по 15-ое сентября |
| a29 | Float(4,2) | Массив, хранящий интерполированные значения для промежутка с 15-го сентября по 15-ое октября |
| a210 | Float(4,2) | Массив, хранящий интерполированные значения для промежутка с 15-го октября по 15-ое ноября |
| a211 | Float(4,2) | Массив, хранящий интерполированные значения для промежутка с 15-го ноября по 15-ое декабря |
| a212 | Float(4,2) | Массив, хранящий интерполированные значения для промежутка с 15-го декабря по 15-ое января |

Необходимо также учесть переход от 15-го декабря к 15-му января. Получается, что часть интерполированных значений, соответствующих периоду с 15-го декабря по 31-го декабря, должны находиться в конце выходной последовательности, а часть, соответствующая периоду с 1-го января по 15-ое января, наоборот находиться в начале. Таким образом, мы делим массив *a212* на две части и, при формировании выходного массива, помещаем части на соответствующие места.

Табл. 11 содержит описание переменных, которые будут использоваться для правильного формирования выходной последовательности.

Табл. 11

Описание переменных для формирования выходной последовательности

| Название переменной | Тип переменной | Значение переменной |
|----------------------------|-----------------------|--|
| begin | Float(4,2) | Массив, содержащий часть массива a212, которая соответствует периоду с 1-го января по 15-января |
| end | Float(4,2) | Массив, содержащий часть массива a212, которая соответствует периоду с 15-го декабря по 31-декабря |
| a | Float(4,2) | Выходной массив, в котором мы собираем данные за все месяцы |

На Рис.9. представлена схема алгоритма интерполяции данных (схема алгоритма интерполирования данных по ЕСПД приведена в приложении Г).

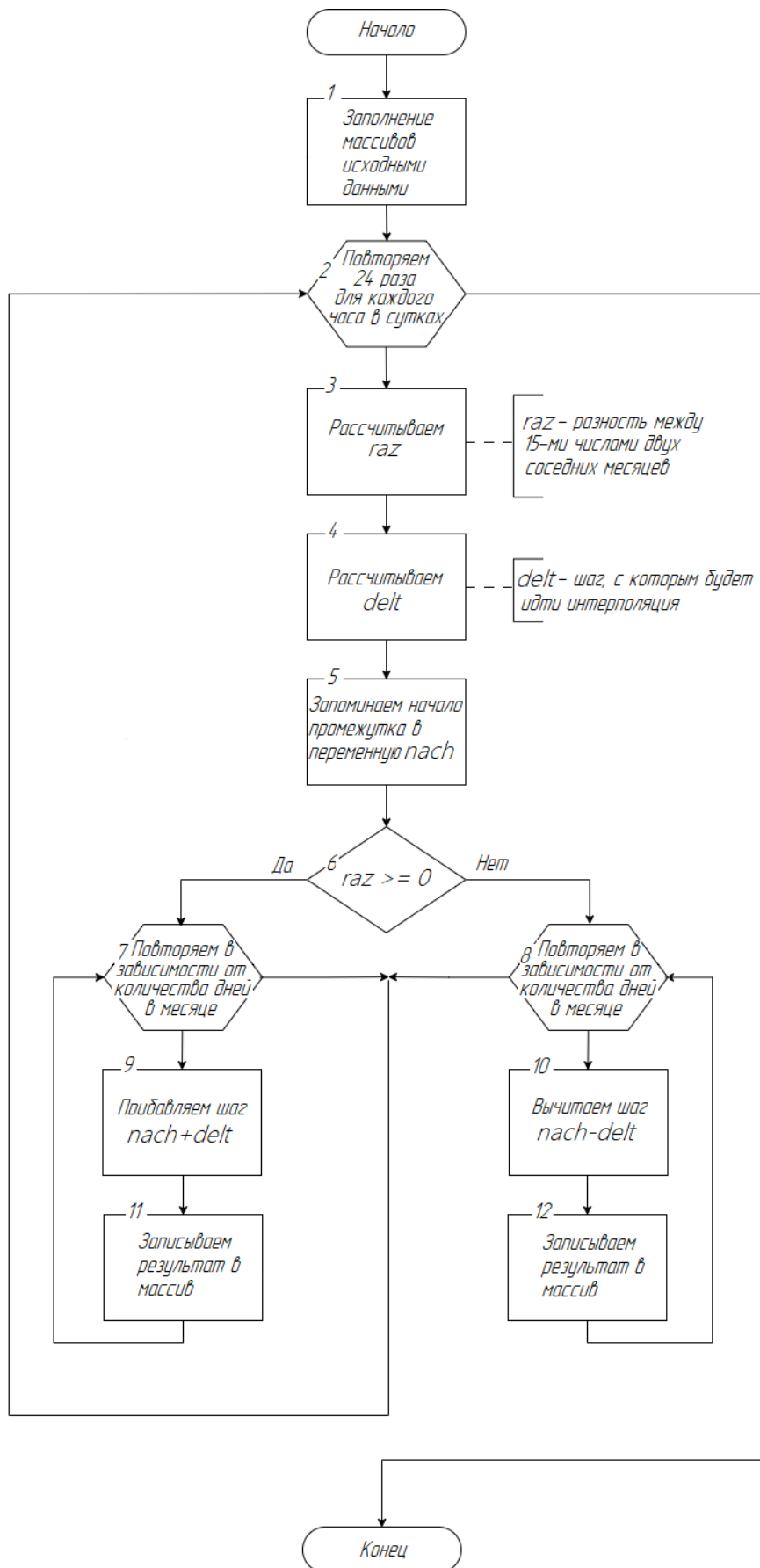


Рис.9. Схема алгоритма интерполяции

3.5. Экспорт в csv

В данном разделе описывается алгоритм экспорта данных для выбранной станции в формате csv. CSV (*Comma-Separated Values* — значения, разделённые запятыми) — текстовый формат, предназначенный для представления табличных данных. Строка таблицы соответствует строке текста, которая содержит одно или несколько полей, разделённых запятыми.

Всего экспортируется 5 таблиц: таблица альбедо, таблицы среднемесячных часовых сумм суммарной, прямой и диффузной солнечной радиации, а так же таблица, формируемая в результате интерполяции.

Первые четыре экспортируются в том же виде, в каком хранятся в базе данных, то есть в виде среднемесячных значений.

Последняя таблица формируется в зависимости от заданных пользователем параметров: период, за который необходимо получить данные, тип интерполяции и шаг дискретизации.

Есть два варианта указания периода:

- за весь период наблюдений;
- за указанный интервал;

Два типа интерполяции:

- День сурка;
- Линейная;

Два варианта шага дискретизации:

- 1 час;
- 2 часа;

В зависимости от сочетания значений этих трех входных параметров мы получаем на выходе некоторую таблицу, которую и направляем на экспорт.

Описание алгоритма экспорта интерполированных данных

На входе алгоритма имеем 3 параметра, которые получаем при помощи GET запроса от пользовательской формы.

1. *Period* – период за который мы хотим получить данные.

Возможны два варианта:

All_period – данные берутся за весь период наблюдения (1 год);

Part_period – данные берутся за указанный интервал;

Для задания интервала указываются 6 параметров:

Mes_begin – месяц начала интервала (обязательный параметр);

Day_begin – день начала интервала;

Hour_begin – час начала интервала;

Mes_end – месяц конца интервала (обязательный параметр);

Day_end – день конца интервала;

Hour_end – час конца интервала;

2. *Type* – тип интерполяции, который мы хотим применить к нашим исходным данным.

Возможны два варианта:

Type_surok – интерполяция по принципу «День сурка»;

Type_lin – интерполяция по принципу «Линейная интерполяция»;

3. *Hour* – шаг дискретизации, с которым мы хотим интерполировать данные.

Возможны два варианта:

1_Hour – шаг дискретизации 1 час, то есть берем каждое значение;

2_Hour – шаг дискретизации 2 часа, то есть берем каждое второе значение;

На Рис.10. представлен вид пользовательской формы приложения.

Выберете период, за который Вы хотите получить данные

☐ За весь период наблюдений
☒ Задать временной интервал

Начиная с:

Заканчивая на:

Месяц

День

Час

Месяц

День

Час

Выберете тип интерполяции

☒ "День сурка"
☐ "Линейная интерполяция"

Выберете шаг дискретизации

☒ 1 час
☐ 2 часа

Экспорт

Рис.10. Вид пользовательской формы

Все три параметра должны быть обязательно выбраны. Это достигается использованием настраиваемой логики проверки для полей. Обязательные для заполнения поля имеют атрибут `required`. Поэтому, если пользователь не выберет период, тип интерполяции или шаг дискретизации, то фреймворк выдаст предупреждение, в котором попросит выбрать один из вариантов.

Обязательно для заполнения поле – Месяц. Если мы не укажем месяц, то получим неопределенность в запрашиваемых данных. Однако день и час можно не указать, тогда просто устанавливается по умолчанию:

– начало периода:

$Day_begin = 1$ (первый день месяца);

$Hour_begin = 1$ (первый час дня);

– конец периода:

$Day_end = 31/30/28$ (последний день месяца, а в месяце может быть либо 31, либо 30, либо 28 дней);

$Hour_end = 24$ (последний час дня);

В соответствии со значением полученного параметра *Type* обрабатываем массивы, содержащие интерполированные данные часовых сумм суммарной, прямой и диффузной солнечной радиации, а также альбедо.

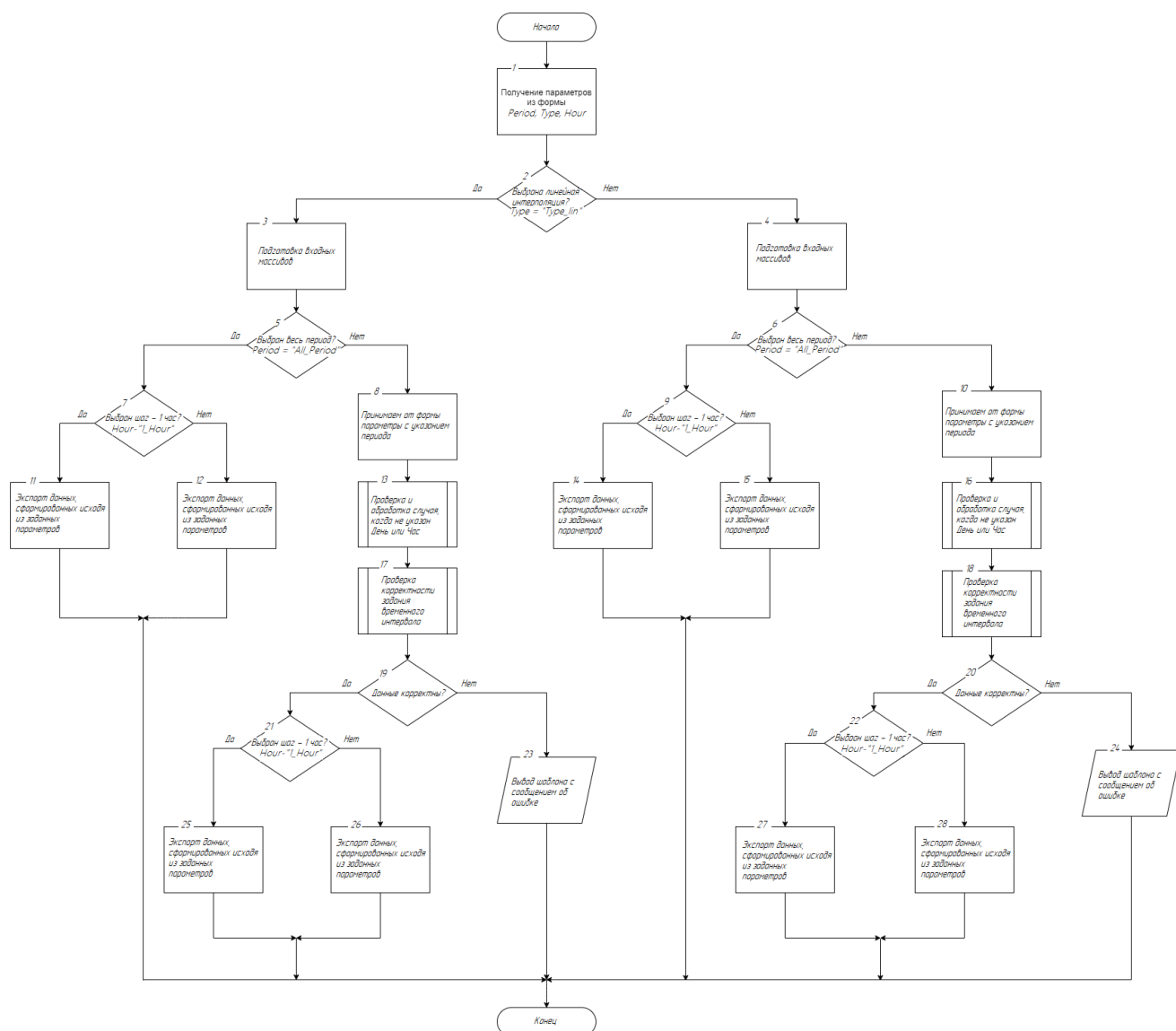
Формируем массивы с индексацией: *id*, *HOY*, *DOY*, *Month*, *Day*, *Hour*, в соответствии с предоставленным институтом ГВИЭ форматом выходного файла.

Далее в цикле последовательно подаем все эти массивы на экспорт. В зависимости от заданных параметров берутся определенные элементы массивов в соответствии с индексом.

Если указан шаг дискретизации 1 час, то берутся все значения подряд, если 2 часа, то каждое второе (кроме *id*, этот столбец должен всегда содержать непрерывную последовательность).

Если указан параметр, указывающий, что нам нужны данные за весь период наблюдений, то мы считываем все массивы целиком. Если нужны данные за какой-то период, то мы, в соответствии с указанными значениями начала и конца периода, высчитываем индекс, с которого нужно начать читать данные, а также индекс, на котором мы должны остановиться.

На Рис.11. представлена структура алгоритма экспорта данных (схема алгоритма экспорта данных в формате csv по ЕСПД приведена в приложении Д).



Для реализации экспорта использовалась библиотека «django-import-export» [1 – 3]. Сначала создается объект `HttpResponse` с соответствующим заголовком CSV. Это сообщает браузерам, что документ является файлом CSV, а не файлом HTML. Далее подключаемся к API генерации CSV. Передаем ответ из начального аргумента в `csv.writer`. Определяем возвращаемую информацию, режим вложения и имя файла. Записываем в файл заголовок, а затем построчно данные из массивов. Пример кода реализации экспорта данных показана ниже.

```

response = HttpResponse(content_type='text/csv')
writer = csv.writer(response)
response['Content-Disposition'] = 'attachment; filename = "Irradiation_8760.csv"'

writer.writerow([' ', 'HOY', 'DOY', 'Month', 'Day', 'Hour', 'GHI', 'DHI', 'DirectHI',
'Albedo'])

for i in range(8760):
    writer.writerow([i, HOY[i], DOY[i], month[i], day[i], hour[i], a1[i], a3[i], a2[i], a[i]])

```

Названиям столбцов в заголовке соответствуют следующие значения:

id – индексация

HOY – час в году;

DOY – день в году;

Month – месяц;

Day – день;

Hour – час;

GHI – часовые суммы суммарной солнечной радиации;

DHI – часовые суммы диффузной солнечной радиации;

DirectHI – часовые суммы прямой солнечной радиации;

Albedo – коэффициенты отражающей способности – альбедо;

3.6. Разработка пользовательского интерфейса

В результате выполнения данной работы был разработан пользовательский интерфейс. Основной задачей было сделать его максимально удобным и интуитивно понятным. Разрабатывался он на языке HTML5 и CSS3 [8, 9], а также Bootstrap, свободного набора инструментов для создания сайтов и веб-приложений. Bootstrap включает в себя HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения [20]. Предусматривалась адаптивность приложения под различные размеры дисплея, такие как планшет (10 - 11 дюймов) и смартфон (5 - 6 дюймов).

Отображаемые шаблоны содержат 4 основные области: заголовочная часть, навигационное меню, контентная часть и футер сайта. Навигационное меню, заголовочная часть и футер остаются постоянными на всех страницах приложения.

Навигационное меню находится в левой части экрана и содержит ссылки на страницы поиска: «Поиск по названию», «Поиск по номеру», «Поиск по координатам». Рядом с навигационным меню находится область контента, в которой отражается информация в зависимости от выбранного раздела. Шапка сайта содержит кнопку «Список станций», при нажатии на неё отображается таблица, содержащая основную информацию по всем станциям, ее интерфейс показан на Рис.13.

Главная страница приложения содержит в контентной части информацию о данных, содержащихся в базе. Интерфейс главной страницы показан на Рис.12.

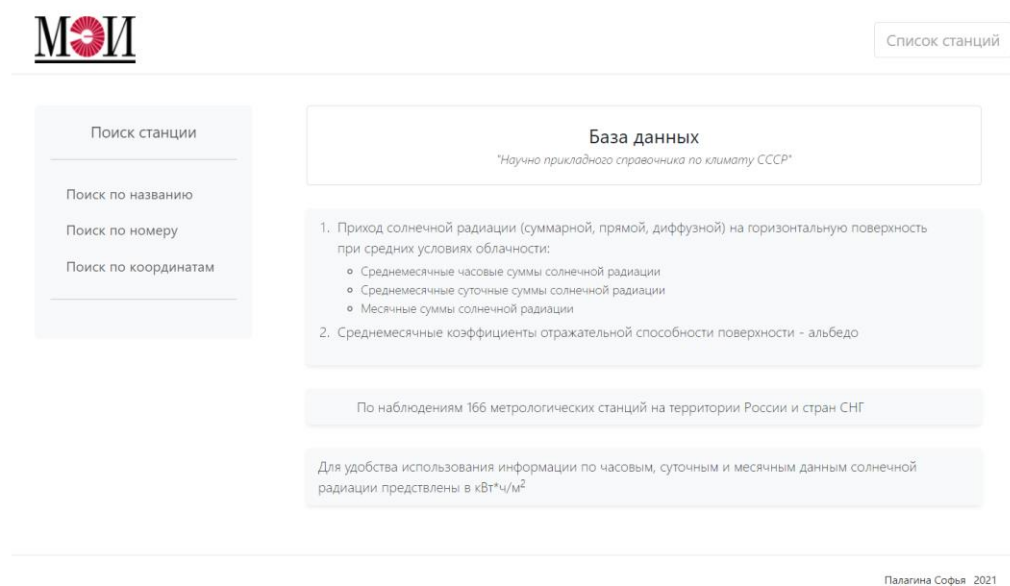


Рис.12. Главная страница приложения

| Номер станции | Название станции | Регион | Широта | Долгота |
|---------------|-------------------------------|-----------------------|--------|---------|
| 1 | Гасан-Кули | Туркмения | 37,10 | 54,00 |
| 2 | Курган-Тюбе | Курган-Тюбинская | 37,70 | 68,80 |
| 3 | Ашхабад | Ашхабадская | 38,00 | 58,30 |
| 4 | им. академика Н. П. Горбунова | Горно-Бадахшанская АО | 38,40 | 72,20 |
| 5 | Душанбе. АС | Таджикистан | 38,50 | 68,80 |
| 6 | Бекибент | Красноводская | 38,70 | 56,00 |
| 7 | Чарджоу | Чарджоуская | 39,10 | 63,80 |
| 8 | Нахичевань | Нахичеванская АССР | 39,20 | 45,30 |
| 9 | Харамкуль | Таджикистан | 39,20 | 68,40 |
| 10 | Акмолла | Ташауская | 39,40 | 60,30 |
| 11 | Ереван. агро | Армения | 40,20 | 44,50 |
| 12 | Мартуни | Армения | 40,20 | 45,20 |
| 13 | Кайракумское водохр. | Ленинабадская | 40,20 | 69,60 |
| 14 | Артём-остров | Дагестан | 40,40 | 50,60 |
| 15 | Севан. озерная ГМО | Армения | 40,60 | 44,80 |

Рис.13. Страница просмотра информации по всем станциям

При выборе «Поиск по названию» происходит переход на страницу, где находится поле ввода. В это поле предложено ввести название интересующей станции. После ввода необходимо нажать на кнопку «Поиск», если поиск пройдет удачно, то произойдет переход на страницу, содержащую данные по выбранной

станции, а если же нет, то на странице выведется сообщение об ошибке. Интерфейс страницы поиска по названию представлен на Рис.14.

The screenshot shows the MOI website's search interface. At the top left is the MOI logo. At the top right is a button labeled "Список станций". Below the logo is a sidebar with the heading "Поиск станции" and three options: "Поиск по названию", "Поиск по номеру", and "Поиск по координатам". The main area features a large input field with the placeholder text "Введите название интересующей станции". Below this input field is a smaller input field with the placeholder text "Введите название". At the bottom of the main area is a dark button labeled "Поиск". At the bottom right of the page is the text "Палагина Софья 2021".

Рис.14. Страница «Поиск по названию»

Интерфейс страницы, которая отображается при выборе «Поиск по номеру», аналогичен предыдущей, только в поле ввода необходимо ввести номер станции. Интерфейс страницы «Поиск по номеру» представлен на Рис. 15, Рис. 16 и на Рис.17.

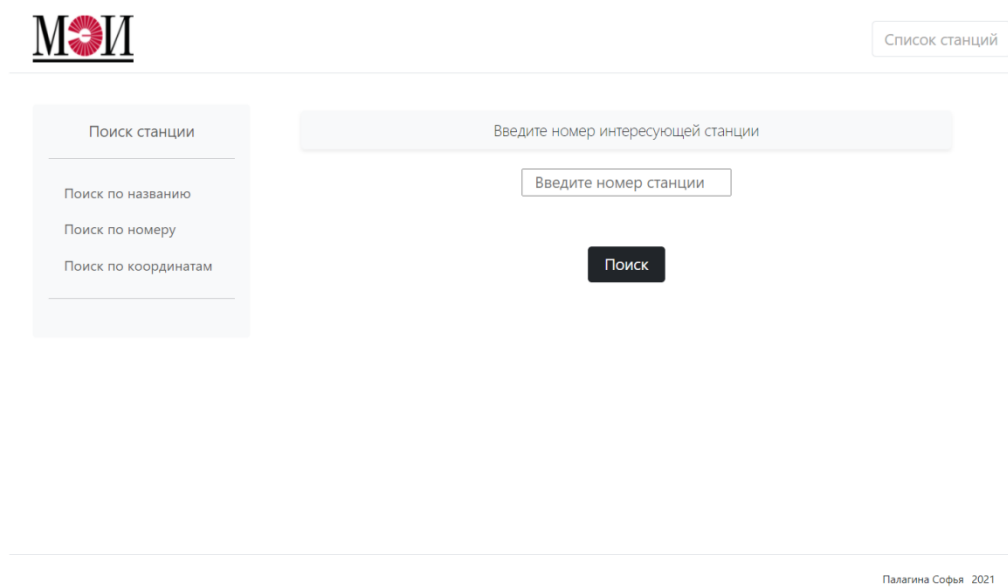


Рис.15. Страница «Поиск по номеру»

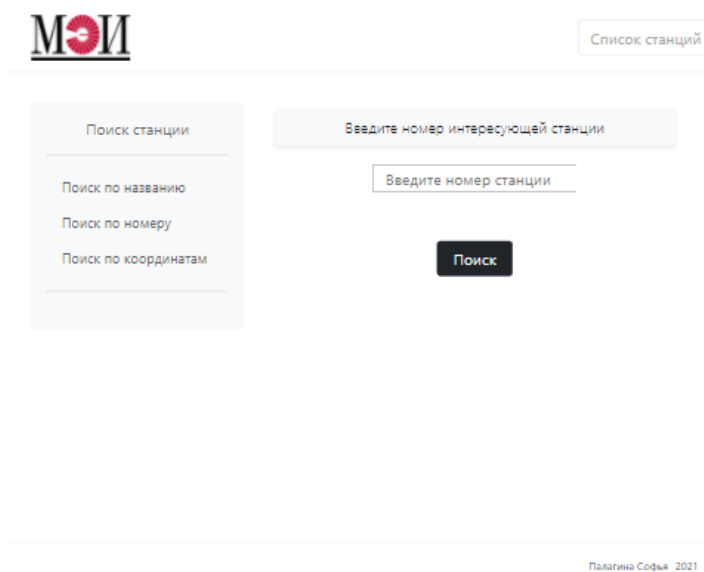



Рис.16. Страница «Поиск по номеру». Формат просмотра - планшет



Список станций

Поиск станции

Поиск по названию

Поиск по номеру

Поиск по координатам

Введите номер
интересующей станции

Введите номер станции

Поиск

Рис.17. Страница «Поиск по номеру». Формат просмотра – мобильное устройство

При выборе «Поиск по координатам» происходит переход на страницу, содержащую в контентной области два поля ввода. В первое поле необходимо ввести широту, во второе долготу искомой точки. После заполнения полей необходимо нажать на кнопку «Поиск». Интерфейс страницы «Поиск по координатам» представлен на Рис.18.

The screenshot shows the 'MOI' logo in the top left corner. In the top right corner, there is a button labeled 'Список станций'. Below the logo, on the left, is a sidebar titled 'Поиск станции' containing three links: 'Поиск по названию', 'Поиск по номеру', and 'Поиск по координатам'. The main content area is titled 'Введите координаты интересующей станции'. It contains two input fields: 'Широта' with the value '30' and 'Долгота' with the value '20'. Below each input field is the text 'в градусах'. At the bottom center of the main area is a dark button labeled 'Поиск'. In the bottom right corner of the page, there is a small footer text: 'Палагина Софья 2021'.

Рис.18. Страница «Поиск по координатам»

В результате поиска по географическим координатам происходит переход на страницу, где отображаются сначала введенные параметры, а затем три блока, содержащие информацию по ближайшим станциям, а именно широту и долготу ближайшей найденной станции, и расстояние от найденной станции до указанной пользователем точки. Исходя из представленной информации, пользователь может сделать выбор, нажав на кнопку с названием станции. В результате выбора произойдет переход на страницу, содержащую данные по выбранной станции. Вид пользовательского интерфейса представлен на Рис.19, Рис.20 и на Рис.21.

Поиск станции

Поиск по названию

Поиск по номеру

Поиск по координатам

Введенные значения

| Широта | Долгота |
|--------|---------|
| 44 | 67 |

Результаты поиска ближайших станций

Кайракумское водохр.

Широта: 40,20
Долгота: 69,60
Расстояние до заданной точки: 473,66

Аральское море

Широта: 46,70
Долгота: 61,60
Расстояние до заданной точки: 518,57

Харамкуль

Широта: 39,20
Долгота: 68,40
Расстояние до заданной точки: 545,72

* Данные станции были подобраны из расчета наименьшего расстояния (в километрах) от указанной Вами точки.

Рис.19. Страница с результатом поиска по географическим координатам

Введенные значения

| Широта | Долгота |
|--------|---------|
| 44 | 44 |

Результаты поиска ближайших станций

Пятигорск

Широта: 44,10
Долгота: 43,00
Расстояние до заданной точки: 80,90

Крестовый перевал

Широта: 42,60
Долгота: 44,30
Расстояние до заданной точки: 157,43

Кизляр

Широта: 43,90
Долгота: 46,60
Расстояние до заданной точки: 209,00

Рис.20. Страница с результатом поиска по географическим координатам. Формат просмотра – мобильное устройство



Список станций

Поиск станции

 Поиск по названию
 Поиск по номеру
 Поиск по координатам

Введенные значения

| Широта | Долгота |
|--------|---------|
| 44 | 44 |

Результаты поиска ближайших станций

Пятигорск

Широта: 44,10
Долгота: 43,00
Расстояние до заданной точки: 80.90

Крестовый перевал

Широта: 42,60
Долгота: 44,30
Расстояние до заданной точки: 157.43

Кизляр

Широта: 43,90
Долгота: 46,60
Расстояние до заданной точки: 209.00

* Данные станции были подобраны из расчета наименьшего расстояния (в километрах) от указанной Вами точки.

Рис.21. Страница с результатом поиска по географическим координатам. Формат просмотра - планшет

После выбора станции пользователь попадает на страницу, содержащую информацию по станции. Сначала отображается название станции и основная информация по ней. Далее располагается кнопка «Экспорт csv», при нажатии на нее появляется выпадающее меню. Данное меню содержит следующие поля: «Альбедо», «Средние часовые суммы суммарной солнечной радиации», «Средние часовые суммы прямой солнечной радиации», «Средние часовые суммы диффузной солнечной радиации» и «Интерполировать данные». При нажатии на первые четырех ссылки происходит скачивание файлов в формате csv, которые содержат исходные необработанные таблицы: альбедо, средние часовые суммы

суммарной солнечной радиации, средние часовые суммы прямой солнечной радиации, средние часовые суммы диффузной солнечной радиации.

При нажатии на кнопку «Интерполировать данные» происходит переход на страницу для задания параметров интерполяции. Далее идет небольшое навигационное меню, при нажатии на соответствующую ссылку происходит прокрутка экрана на нужную область страницы, это, своего рода, закладка для быстрого перемещения по странице с целью поиска, называемая якорь сайта.

Далее непосредственно идут сами данные в виде таблиц, которые пользователь может просматривать. Интерфейс страницы представлен на Рис.22, Рис.23, Рис.24, Рис.25 и на Рис.26.

MOI

Список станций

Поиск станции

Поиск по названию

Поиск по номеру

Поиск по координатам

Информация по станции Чарджоу

| Номер станции | Название станции | Регион | Широта | Долгота |
|---------------|------------------|-------------|--------|---------|
| 7 | Чарджоу | Чарджоуская | 39,10 | 63,80 |

Экспорт csv

Альбедо

Среднемесячные часовые суммы суммарной солнечной радиации

Среднемесячные часовые суммы прямой солнечной радиации

Среднемесячные часовые суммы диффузной солнечной радиации

Интерполировать данные


Альбедо Месячные суммы Суточные суммы Часовые суммы

Рис.22. Страница с данными по станции

| Среднемесячные коэффициенты отражательной способности поверхности - Альбедо | | | | | | | | | | | |
|--|------|------|------|------|------|------|------|------|------|-------|------|
| Янв. | Фев. | Март | Апр. | Май | Июнь | Июль | Авг. | Сен. | Окт. | Нояб. | Дек. |
| 0,28 | 0,26 | 0,22 | 0,21 | 0,22 | 0,23 | 0,23 | 0,24 | 0,24 | 0,24 | 0,25 | 0,25 |

| Месячные суммы солнечной радиации на горизонтальную площадку | | | | | | | | | | | |
|--|-------|--------|--------|--------|--------|--------|--------|--------|--------|-------|-------|
| Суммарная солнечная радиация: | | | | | | | | | | | |
| Янв. | Фев. | Март | Апр. | Май | Июнь | Июль | Авг. | Сен. | Окт. | Нояб. | Дек. |
| 69,20 | 90,10 | 131,80 | 168,50 | 225,40 | 251,00 | 255,60 | 234,70 | 185,70 | 136,10 | 86,00 | 59,10 |
| Прямая солнечная радиация: | | | | | | | | | | | |
| Янв. | Фев. | Март | Апр. | Май | Июнь | Июль | Авг. | Сен. | Окт. | Нояб. | Дек. |
| 34,30 | 48,50 | 68,40 | 93,80 | 152,90 | 187,70 | 195,60 | 181,70 | 140,30 | 95,60 | 51,80 | 29,80 |
| Диффузная солнечная радиация: | | | | | | | | | | | |
| Янв. | Фев. | Март | Апр. | Май | Июнь | Июль | Авг. | Сен. | Окт. | Нояб. | Дек. |
| 35,00 | 41,50 | 63,40 | 74,70 | 72,50 | 63,30 | 59,90 | 53,00 | 45,30 | 40,50 | 34,20 | 29,30 |

Рис.23. Страница с данными по станции. Продолжение Рис. 22



Список станций

Поиск станции

Поиск по названию

Поиск по номеру

Поиск по координатам

Информация по станции Крестовый перевал

| Номер станции | Название станции | Регион | Широта | Долгота |
|---------------|-------------------|--------|--------|---------|
| 27 | Крестовый перевал | Грузия | 42,60 | 44,30 |

Экспорт csv

Альбедо

Месячные суммы

Суточные суммы

Часовые суммы

Среднемесячные коэффициенты отражательной способности поверхности -
Альбедо

| Янв. | Фев. | Март | Апр. | Май | Июнь | Июль | Авг. | Сен. | Окт. | Нояб. | Дек. |
|------|------|------|------|------|------|------|------|------|------|-------|------|
| 0,76 | 0,78 | 0,78 | 0,71 | 0,45 | 0,23 | 0,23 | 0,23 | 0,23 | 0,33 | 0,64 | 0,76 |

Месячные суммы солнечной радиации на горизонтальную площадку

Суммарная солнечная радиация:

| Янв. | Фев. | Март | Апр. | Май | Июнь | Июль | Авг. | Сен. | Окт. | Нояб. | Дек. |
|-------|--------|--------|--------|--------|--------|--------|--------|--------|--------|-------|-------|
| 85,90 | 106,80 | 152,90 | 167,30 | 178,50 | 173,60 | 178,50 | 158,10 | 121,30 | 107,70 | 80,80 | 70,10 |

Рис.24. Страница с данными по станции. Формат просмотра - планшет

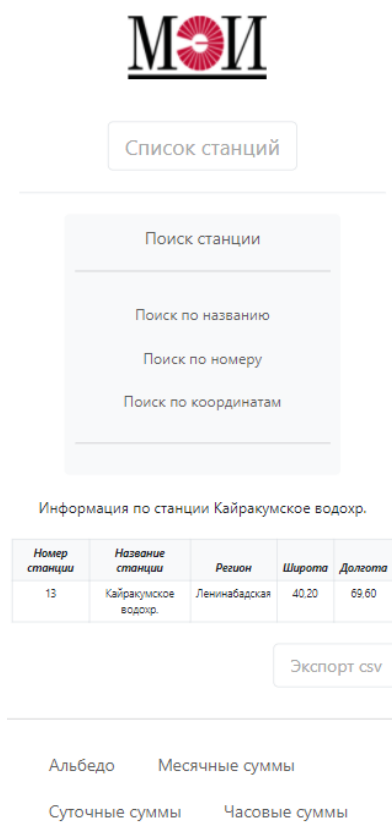


Рис.25. Страница с данными по станции. Формат просмотра – мобильное устройство

Суточные суммы солнечной радиации на горизонтальную площадку

Суммарная солнечная радиация:

| Янв. | Фев. | Март | Апр. | Май | Июнь | Июль | Авг. | Сен. | Окт. | Нояб. | Дек. |
|------|------|------|------|------|------|------|------|------|------|-------|------|
| 2,05 | 2,85 | 3,77 | 5,28 | 6,99 | 8,01 | 7,76 | 7,11 | 5,63 | 3,86 | 2,40 | 1,57 |

Прямая солнечная радиация:

| Янв. | Фев. | Март | Апр. | Май | Июнь | Июль | Авг. | Сен. | Окт. | Нояб. | Дек. |
|------|------|------|------|------|------|------|------|------|------|-------|------|
| 0,95 | 1,21 | 1,85 | 2,97 | 4,71 | 5,99 | 5,74 | 5,29 | 3,93 | 2,47 | 1,33 | 0,63 |

Диффузная солнечная радиация:

| Янв. | Фев. | Март | Апр. | Май | Июнь | Июль | Авг. | Сен. | Окт. | Нояб. | Дек. |
|------|------|------|------|------|------|------|------|------|------|-------|------|
| 1,10 | 1,61 | 1,92 | 2,31 | 2,22 | 2,01 | 2,01 | 1,82 | 1,70 | 1,39 | 1,08 | 0,94 |

Часовые суммы солнечной радиации на горизонтальную площадку

Суммарная солнечная радиация:

| Янв. | Фев. | Март | Апр. | Май | Июнь | Июль | Авг. | Сен. | Окт. | Нояб. | Дек. |
|------|------|------|------|------|------|------|------|------|------|-------|------|
| 1 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |
| 2 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |
| 3 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |
| 4 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 |
| 5 | 0,00 | 0,00 | 0,00 | 0,00 | 0,00 | 0,01 | 0,01 | 0,00 | 0,00 | 0,00 | 0,00 |
| 6 | 0,00 | 0,00 | 0,00 | 0,02 | 0,07 | 0,11 | 0,06 | 0,04 | 0,04 | 0,00 | 0,00 |
| 7 | 0,00 | 0,00 | 0,00 | 0,11 | 0,21 | 0,24 | 0,24 | 0,18 | 0,09 | 0,03 | 0,00 |
| 8 | 0,02 | 0,06 | 0,11 | 0,26 | 0,39 | 0,45 | 0,41 | 0,27 | 0,26 | 0,14 | 0,05 |
| 9 | 0,11 | 0,17 | 0,27 | 0,41 | 0,56 | 0,63 | 0,59 | 0,50 | 0,45 | 0,31 | 0,17 |
| 10 | 0,20 | 0,28 | 0,38 | 0,54 | 0,70 | 0,79 | 0,76 | 0,71 | 0,61 | 0,44 | 0,27 |
| 11 | 0,29 | 0,38 | 0,50 | 0,65 | 0,81 | 0,89 | 0,86 | 0,80 | 0,71 | 0,54 | 0,34 |
| 12 | 0,36 | 0,45 | 0,56 | 0,70 | 0,85 | 0,94 | 0,93 | 0,89 | 0,76 | 0,59 | 0,40 |
| 13 | 0,37 | 0,46 | 0,56 | 0,70 | 0,85 | 0,94 | 0,93 | 0,89 | 0,77 | 0,57 | 0,40 |
| 14 | 0,32 | 0,40 | 0,49 | 0,63 | 0,78 | 0,84 | 0,85 | 0,83 | 0,70 | 0,51 | 0,34 |
| 15 | 0,23 | 0,32 | 0,40 | 0,52 | 0,65 | 0,71 | 0,71 | 0,70 | 0,57 | 0,36 | 0,21 |
| 16 | 0,13 | 0,20 | 0,27 | 0,39 | 0,50 | 0,61 | 0,61 | 0,54 | 0,40 | 0,24 | 0,13 |

Рис.26. Страница с данными по станции. Продолжение Рис. 25. Формат просмотра – мобильное устройство

При переходе по ссылке «Интерполировать данные» пользователь попадает на страницу для задания параметров интерполяции. Страница содержит три группы по две радиокнопки. Первая группа отвечает за задание периода. Есть два варианта: «За весь период наблюдений» и «Задать временной интервал». При активации радиокнопки «Задать временной интервал» появляется выпадающая область, которая содержит 6 полей, три из которых относятся к заданию начала периода (месяц, день и час), а три остальных для задания окончания периода (месяц, день и час).

Вторая группа задает тип интерполяции и содержит два варианта: «День сурка» и «Линейная интерполяция».

Третья группа задает шаг дискретизации, также содержит два варианта: «1 час» и «2 часа».

Пользователь может выбрать только один вариант из двух во всех группах. После выбора всех параметров необходимо нажать на кнопку «Экспорт», при нажатии на которую произойдет скачивание файла в формате csv. Файл будет содержать интерполированные данные в зависимости от заданных параметров. Вид пользовательского интерфейса представлен на Рис.27, Рис.28, Рис.29 и на Рис.30.

The screenshot shows the MOI web application interface. At the top left is the MOI logo. At the top right is a button labeled "Список станций". Below the logo is a sidebar with the heading "Поиск станции" and three links: "Поиск по названию", "Поиск по номеру", and "Поиск по координатам". The main content area contains three stacked sections for parameter selection:

- Выберете период, за который Вы хотите получить данные**
 - ☐ За весь период наблюдений
 - ☐ Задать временной интервал
- Выберете тип интерполяции**
 - ☐ "День сурка"
 - ☐ "Линейная интерполяция"
- Выберете шаг дискретизации**
 - ☐ 1 час
 - ☐ 2 часа

At the bottom center is a dark button labeled "Экспорт". At the bottom right, in small text, is "Палагина Софья 2021".

Рис.27. Страница для задания параметров интерполяции

Поиск станции

Поиск по названию

Поиск по номеру

Поиск по координатам

Выберете период, за который Вы хотите получить данные

☐ За весь период наблюдений
 ☒ Задать временной интервал

Начиная с:

Заканчивая на:

Месяц

День

Час

Месяц

День

Час

Выберете тип интерполяции

☐ "День сурка"
 ☐ "Линейная интерполяция"

Выберете шаг дискретизации

☐ 1 час
 ☐ 2 часа

Экспорт

Рис.28. Страница для задания параметров интерполяции. Вид с раскрытой областью для задания периода

МОИ

Список станций

Поиск станции

Поиск по названию

Поиск по номеру

Поиск по координатам

Выберете период, за который Вы хотите получить данные

☐ За весь период наблюдений
 ☐ Задать временной интервал

Выберете тип интерполяции

☐ "День сурка"
 ☐ "Линейная интерполяция"

Выберете шаг дискретизации

☐ 1 час
 ☐ 2 часа

Экспорт

Палагина София 2021

Рис.29. Страница для задания параметров интерполяции. Формат просмотра - планшет

Поиск станции

Поиск по названию

Поиск по номеру

Поиск по координатам

Выберете период, за который Вы хотите получить данные

☐ За весь период наблюдений
 ☐ Задать временной интервал

Выберете тип интерполяции

☐ "День сурка"
 ☐ "Линейная интерполяция"

Выберете шаг дискретизации

☐ 1 час
 ☐ 2 часа

Экспорт

Рис.30. Страница для задания параметров интерполяции. Формат просмотра – мобильное устройство

Вид выходного файла представлен на Рис.31.

| 1 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W |
|----|-----|-----|-------|-----|------|----|------|---------|--------|------|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | HOY | DOY | Month | Day | Hour | GH | DH | DirectH | Albedo | | | | | | | | | | | | | | |
| 3 | 1 | 2 | 1 | 1 | 1 | 2 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 4 | 2 | 3 | 1 | 1 | 1 | 3 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 5 | 3 | 4 | 1 | 1 | 1 | 4 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 6 | 4 | 5 | 1 | 1 | 1 | 5 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 7 | 5 | 6 | 1 | 1 | 1 | 6 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 8 | 6 | 7 | 1 | 1 | 1 | 7 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 9 | 7 | 8 | 1 | 1 | 1 | 8 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 10 | 8 | 9 | 1 | 1 | 1 | 9 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 11 | 9 | 10 | 1 | 1 | 1 | 10 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 12 | 10 | 11 | 1 | 1 | 1 | 11 | 0.01 | 0.01 | 0 | 0.81 | | | | | | | | | | | | | |
| 13 | 11 | 12 | 1 | 1 | 1 | 12 | 0.03 | 0.03 | 0 | 0.81 | | | | | | | | | | | | | |
| 14 | 12 | 13 | 1 | 1 | 1 | 13 | 0.03 | 0.03 | 0 | 0.81 | | | | | | | | | | | | | |
| 15 | 13 | 14 | 1 | 1 | 1 | 14 | 0.01 | 0.01 | 0 | 0.81 | | | | | | | | | | | | | |
| 16 | 14 | 15 | 1 | 1 | 1 | 15 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 17 | 15 | 16 | 1 | 1 | 1 | 16 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 18 | 16 | 17 | 1 | 1 | 1 | 17 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 19 | 17 | 18 | 1 | 1 | 1 | 18 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 20 | 18 | 19 | 1 | 1 | 1 | 19 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 21 | 19 | 20 | 1 | 1 | 1 | 20 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 22 | 20 | 21 | 1 | 1 | 1 | 21 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 23 | 21 | 22 | 1 | 1 | 1 | 22 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 24 | 22 | 23 | 1 | 1 | 1 | 23 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 25 | 23 | 24 | 1 | 1 | 1 | 24 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 26 | 24 | 25 | 2 | 1 | 2 | 1 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 27 | 25 | 26 | 2 | 1 | 2 | 2 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 28 | 26 | 27 | 2 | 1 | 2 | 3 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 29 | 27 | 28 | 2 | 1 | 2 | 4 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 30 | 28 | 29 | 2 | 1 | 2 | 5 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 31 | 29 | 30 | 2 | 1 | 2 | 6 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |
| 32 | 30 | 31 | 2 | 1 | 2 | 7 | 0 | 0 | 0 | 0.81 | | | | | | | | | | | | | |

Рис.31. Пример выходного файла csv

6. ТЕСТИРОВАНИЕ

Тестирование приложения проводилось, в соответствии с межгосударственным стандартом ГОСТ 19.301-79 – «Программа и методика испытаний». Блок должен содержать следующие разделы:

1) Объект испытаний;

Объект тестирования – веб-приложение для поиска и обработки данных по актинометрическим станциям, которое было разработано в ходе выполнения данной выпускной квалификационной работы.

2) Цель испытаний;

Цель проведения испытаний – проверка работоспособности разработанного приложения, выявление ошибок функционирования, оценка корректности работы.

3) Средства испытаний;

Тестирование производилось на ноутбуке со следующими характеристиками:

Табл. 12

Параметры средства испытаний

| Параметр | Значение |
|----------------------|--|
| Операционная система | Windows 10 |
| Разрядность системы | × 64 |
| Процессор | Intel(R) Core(TM) i7-8565U CPU 1.80GHz, 1992 МГц, ядер: 4, логических процессоров: 8 |
| ОЗУ | 8ГБ DDR3 2133 МГц |
| Разрешение экрана | 1920×1080 |

4) Методы испытаний;

Тесты нацелены на проверку работоспособности веб-приложения в случае ввода некорректных данных. Метод испытаний - функциональное тестирование по входным и выходным данным. Тестирование проводится в нормальных и исключительных условиях.

Все тесты были пройдены успешно, ошибок не выявлено.

6.1. Тестирование алгоритма экспорта

В процессе разработки алгоритма, были учтены все основные исключения.

Фреймворк Django имеет достаточно мощные механизмы валидации данных. То есть на клиентской части приложения будет автоматически идти проверка данных на корректность, прежде чем они отправятся на сервер [3].

Для полей *Period*, *Type*, *Hour* указан атрибут *required*, поэтому в случае, когда пользователь не выберет один из предложенных вариантов, фреймворк выдаст предупреждение. Также атрибут *required* имеет поле *Месяц*.

Для полей *День* и *Час* прописаны обработчики исключений. В случае, если пользователь не укажет *День* или *Час*, просто берутся значения по умолчанию.

Установлены максимальные и минимальные значения вводимых данных.

- для месяца: $\max=12$, $\min=1$;
- для дня: $\max=31$, $\min=1$;
- для часа: $\max=24$, $\min=1$;

Поля *Месяц*, *День* и *Час* принимают только числовые значения, поэтому пользователь просто не сможет ввести символ или букву.

Помимо этого стоит учесть, что данные мы имеем за один год наблюдений, поэтому задавать период мы можем только в порядке возрастания. То есть мы не можем задать начало периода – 5-ый месяц, а конец периода – 2-ой месяц. Это касается и поля День, при условии, что месяцы начала и конца периода одинаковые (пользователь запросил данные за один месяц). Если указаны одинаковые Месяц и День, то час начала периода должен быть больше часа конца периода.

В случае некорректно заданного периода, генерируется исключение, файл не будет сформирован и данные не экспортируются, а пользователю придет сообщение об ошибке.

1. Пользователь ввел некорректный номер месяца, дня или часа (Рис.32)

Рассмотрим данный случай на примере некорректно заданного месяца.

Входные данные: Месяц = 45

Получаем предупреждение, что значение не может быть больше 12.

The screenshot shows the MOI web application interface. On the left is a sidebar with the MOI logo and search options: "Поиск станции", "Поиск по названию", "Поиск по номеру", and "Поиск по координатам". The main content area is titled "Выберите период, за который Вы хотите получить данные". It contains two radio buttons: "За весь период наблюдений" (unselected) and "Задать временной интервал" (selected). Below this is a form for selecting a time interval. It has two columns: "Начиная с:" and "Заканчивая на:". The "Начиная с:" column has a dropdown menu showing "45" and a text input field below it containing "1". A yellow warning box with an exclamation mark icon is positioned over the "45" dropdown, containing the text "Значение должно быть меньше или равно 12.". The "Заканчивая на:" column has three dropdown menus showing "3", "12", and "5". Below the time interval form is a section titled "Выберите тип интерполяции" with two radio buttons: "День сурка" (selected) and "Линейная интерполяция" (unselected). Below that is a section titled "Выберите шаг дискретизации" with two radio buttons: "1 час" (selected) and "2 часа" (unselected). At the bottom of the form is a black button labeled "Экспорт". In the top right corner of the interface is a button labeled "Список станций". In the bottom right corner, there is a small text string: "Палагина Софья 2021".

Рис.32. Пример ввода некорректного числа в поле Месяц

2. Пользователь не выбрал один из вариантов

Рассмотрим случай, что пользователь не задал период, то есть оставил обе радиокнопки в группе задания периода неактивными (Рис.33).

The screenshot shows the MOI web application interface. At the top left is the MOI logo. At the top right is a button labeled "Список станций". Below the logo is a sidebar with the title "Поиск станции" and three links: "Поиск по названию", "Поиск по номеру", and "Поиск по координатам". The main content area has a title "Выберите период, за который Вы хотите получить данные". Below this title are two radio buttons: "За весь период наблюдений" (which is selected) and "За заданный интервал". A yellow warning box with an exclamation mark icon and the text "Выберите один из вариантов." is positioned over the "За заданный интервал" radio button. Below the radio buttons is a section titled "Выберите тип интерполяции" with two radio buttons: "День сурка" (selected) and "Линейная интерполяция". Below this is a section titled "Выберите шаг дискретизации" with two radio buttons: "1 час" (selected) and "2 часа". At the bottom of the main content area is a dark button labeled "Экспорт". At the bottom right of the page is the text "Палагина Софья 2021".

Рис.33. Пример отсутствия выбора

Получаем предупреждение, что нужно выбрать один из предложенных вариантов.

3. Пользователь оставил обязательное поле незаполненным

В случае незаполненного обязательного поля - Месяц получаем предупреждение, что нужно поле заполнить (Рис.34)

Палагина Софья 2021

4. Пользователь неверно задал временной интервал (Рис.35)

4. Пользователь неверно задал временной интервал (Рис.35)

Начало периода:

День = 8

Завершение периода:

День =1

$$\chi_{ac} = -$$

Получается такая ситуация, что данные берутся за один месяц (за 9-ый), но при этом указывается, что начало периода это день – 8, а конец периода день – 1, то есть начало периода больше значения завершения периода.

The screenshot shows the MOI web application interface. At the top left is the MOI logo. At the top right is a button labeled "Список станций". On the left side, there is a sidebar with the title "Поиск станции" and three search options: "Поиск по названию", "Поиск по номеру", and "Поиск по координатам". The main content area contains a form with the following sections:

- Выберите период, за который Вы хотите получить данные**
 - ☐ За весь период наблюдений
 - ☒ Задать временной интервал
- Начиная с:** Three input fields containing the values 9, 8, and 4.
- Заканчивая на:** Three input fields containing the values 9, 1, and 8.
- Выберите тип интерполяции**
 - ☒ "День сурка"
 - ☐ "Линейная интерполяция"
- Выберите шаг дискретизации**
 - ☒ 1 час
 - ☐ 2 часа
- Экспорт** button

At the bottom right of the page, the text "Палагина Софья 2021" is visible.

Рис.35. Пример некорректного интервала

При попытке указать некорректный интервал, получаем сообщение об ошибке (Рис.36).

The screenshot shows the MOI web application interface after an error. The layout is similar to the previous screenshot, but with the following changes:

- The "Экспорт" button is no longer visible.
- A yellow error message box is displayed at the top of the main content area, containing the text: "Некорректно указан временной интервал. Экспорт не может быть осуществлен!".
- The footer text "Палагина Софья 2021" remains at the bottom right.

Рис.36. Выводимое сообщение об ошибке

6.2. Тестирование поиска данных

Рассматриваются случаи неверного ввода данных в поля поиска: «Поиск по названию», «Поиск по номеру» и «Поиск по координатам».

1. Пользователь ввел некорректное название станции

Случай, когда пользователь вводит в поле ввода название станции, информации по которой в базе данных не содержится (Рис.37).

Входные данные:

Название станции – Тамбов;

The screenshot shows the MOI website's search interface. At the top left is the MOI logo. At the top right is a link labeled 'Список станций'. Below the logo is a sidebar with the title 'Поиск станции' and three options: 'Поиск по названию', 'Поиск по номеру', and 'Поиск по координатам'. The main search area has a header 'Введите название интересующей станции' and a text input field containing 'Тамбов'. Below the input field is a dark button labeled 'Поиск'. At the bottom right of the page, there is a small footer that reads 'Палагина Софья 2021'.

Рис.37. Пример задания некорректного названия станции

Станции с таким названием в базе данных не содержится, следовательно, как результат, получим предупреждение, что такой станции нет.

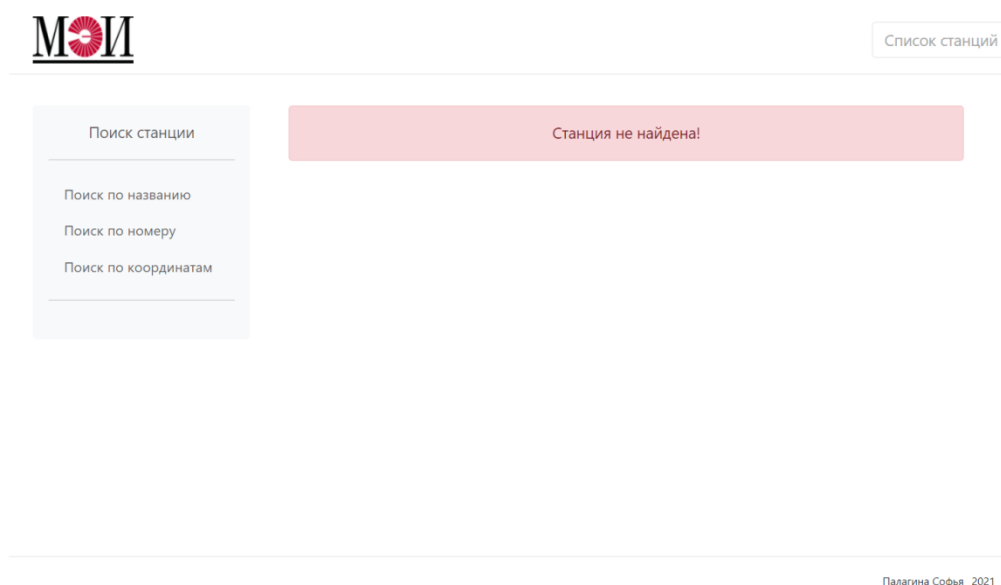


Рис.38. Выводимое сообщение об ошибке

2. Пользователь указал некорректный номер станции

Если пользователь задст значение, равное номеру станции, вне интервала (1, 166), то появится предупреждение, что параметр некорректный (Рис.39).

The screenshot shows the MOI website interface. At the top left is the MOI logo. At the top right is a button labeled "Список станций". Below the logo is a sidebar with the title "Поиск станции" and three options: "Поиск по названию", "Поиск по номеру", and "Поиск по координатам". The main content area has a header "Введите номер интересующей станции" above a text input field containing "197". Below the input field is a red error message: "Значение должно быть меньше или равно 166." and a "Поиск" button. At the bottom right of the page, the text "Палагина Софья 2021" is visible.

Рис.39. Пример задания некорректного номера станции

3. Пользователь указал некорректные географические координаты

Пользователь может ввести не совсем корректные значения для широты и долготы, в результате появится предупреждения о некорректности введенных данных (Рис.40).

Широта: 0° - 90°

Долгота: 20° - 150°

Поиск станции

Поиск по названию

Поиск по номеру


Поиск по координатам

Введите координаты интересующей станции

Широта

Долгота

в градусах

 Значение должно быть меньше или равно 90.

Поиск

Рис.40. Пример задания некорректных географических координат

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы было разработано действующее веб-приложение для поиска данных по актинометрическим станциям, с последующей их обработкой. Разработанное приложение полностью удовлетворяет техническому заданию.

В ходе выполнения работы была разработана структура реляционной базы данных. База данных была приведена к третьей нормальной форме и построена таким образом, чтобы можно было с легкостью ее модифицировать, изменив или добавив новые компоненты. Было создано серверное приложение, позволяющее осуществлять быстрый и удобный поиск по базе данных. Разработаны алгоритмы интерполирования данных в зависимости от заданных параметров. Предусмотрен функционал для экспорта данных в формате csv. Разработанный пользовательский интерфейс интуитивно понятный, простой и отзывчивый к вводимым данным. Страницы приложения корректно отображаются в различных браузерах. Приложение было адаптировано под устройства с разным размером экрана.

Приложение прошло проверку на все основные исключения. Тестирование показало, что приложение работает корректно.

В будущем приложение можно будет с легкостью расширить и дополнить новым функционалом, если предъявляемые требования возрастут.

СПИСОК ЛИТЕРАТУРЫ

1. Official site of Django framework django-import-export [Электронный ресурс]. – Режим доступа: <https://django-import-export.readthedocs.io/en/latest/>
2. Python 3.9.5 documentation [Электронный ресурс]. – Режим доступа: <https://docs.python.org/3/>
3. Official site of Django framework [Электронный ресурс]. – Режим доступа: <https://www.djangoproject.com/start/overview/>
4. Сайт Django.fun. Документация Django 3.1 [Электронный ресурс]. – Режим доступа: <https://django.fun/docs/django/ru/3.1/>
5. Сайт developer.mozilla.org. Веб-технологии для разработчиков [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/ru/docs/>
6. Official site of Django framework geopy [Электронный ресурс]. – Режим доступа: <https://geopy.readthedocs.io/en/stable/>
7. Жданов С.А., Собоелва М.Л., Алфимова А.С.. Информационные системы: учебник студ. учреждений высш. образования – М.: ООО «Прометей», 2015. – 302 с.
8. Дакетт Д. HTML и CSS. Разработка и дизайн веб-сайтов.: Пер. с англ. М.А. Райтман. – М.: Эксмо, 2013. – 480 с.: ил.
9. Сайт htmlbook.ru. Справочник по HTML [Электронный ресурс]. – Режим доступа: <http://htmlbook.ru/html>
10. Фиайли К. SQL: Пер. с англ. – М.: ДМК Пресс. – 456 с.: ил. (Серия «Quick Start»).
11. Сайт ZametkiNaPolyah.ru. О модели взаимодействия клиент-сервер простыми словами [Электронный ресурс]. – Режим доступа: <https://zametkinapolyah.ru/servera-i-protokoly/o-modeli-vzaimodejstviya-klient-server-prostymi-slovami-arxitektura-klient-server-s-primerami.html>
12. Сайт testmatick.com. Основные понятия и особенности клиент-серверной

- архитектуры [Электронный ресурс]. – Режим доступа:
<https://testmattick.com/ru/osnovnye-ponyatiya-i-osobennosti-klient-servernoj-arhitektury/>
13. Сайт bestprogrammer.ru. Какой язык лучше для веб-разработки: PHP или Python? [Электронный ресурс]. – Режим доступа:
<https://bestprogrammer.ru/programmirovanie-i-razrabotka/kakoj-yazyk-luchshe-dlya-veb-razrabotki-php-ili-python>
14. Сайт blog.skillfactory.ru. Что выбрать: PHP или Python? [Электронный ресурс]. – Режим доступа:
<https://blog.skillfactory.ru/php-vs-python-chto-vybrat/>
15. Сайт pythonist.ru. Django vs Flask: что выбрать для своего проекта? [Электронный ресурс]. – Режим доступа:
<https://pythonist.ru/django-vs-flask-chto-vybrat-dlya-svoego-proekta/>
16. Сайт hackr.io. Flask vs Django in 2021: Which Framework to Choose? [Электронный ресурс]. – Режим доступа: <https://hackr.io/blog/flask-vs-django>
17. Сайт rtfm.co.ua. Django Book: концепция разработки MVC – Model, View, Controller [Электронный ресурс]. – Режим доступа:
<https://rtfm.co.ua/django-book-model-razrabotki-mvc-model-view-controller/>
18. Сайт tproger.ru. SQLite, MySQL и PostgreSQL: сравниваем популярные СУБД [Электронный ресурс]. – Режим доступа:
<https://tproger.ru/translations/sqlite-mysql-postgresql-comparison/>
19. Сайт studmed.ru. Научно-прикладной справочник по климату СССР [Электронный ресурс]. – Режим доступа:
https://www.studmed.ru/science/nauki-o-zemle/meteorologiya-i-klimatologiya/nauchnoprikladnoi_spravochnik_po_klimatu_sssr
20. Сайт bootstrap-4.ru. Bootstrap. Документация на русском языке [Электронный ресурс]. – Режим доступа:
<https://bootstrap-4.ru/>

Приложение А. Техническое задание



МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное
учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт
Кафедра

ИВТИ
ВМСС

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
(бакалаврскую работу)

Направление 09.03.01 – Информатика и вычислительная техника
(код и наименование)

Направленность (профиль) Автоматизированные системы обработки
информации и управления

Форма обучения Очная
(очная/очно-заочная/заочная)

Тема: Разработка web-приложения по расчету прихода солнечной
радиации на горизонтальную поверхность

| | | |
|----------------|---------|-------------------|
| Студент | A-12-17 | Палагина С. А. |
| инициалы | группа | подпись фамилия и |

| | | |
|---------------------------------|-----------------------|----------------------------|
| Научный руководитель | ст. преп. | Карвовский Д. А. |
| | уч. степень должность | подпись фамилия и инициалы |

| | | |
|--------------------|----------------------------|----------------------------|
| Консультант | к.т.н. доцент ГВИЭ НИУ МЭИ | Васьков А. Г. |
| | уч. степень должность | подпись фамилия и инициалы |

| | | |
|----------------------|--------------------|----------------------------|
| Зав. кафедрой | к.т.н. доцент | Вишняков С. В. |
| | уч. степень звание | подпись фамилия и инициалы |

СОДЕРЖАНИЕ РАЗДЕЛОВ ЗАДАНИЯ И ИСХОДНЫЕ ДАННЫЕ

Разработать web-приложение по предоставлению данных о приходе солнечной радиации на горизонтальную поверхность. Выполнить данную бакалаврскую работу предстоит совместно с институтом ГВИЭ. Исходную информацию следует получить из предоставленной институтом базы данных. База была разработана сотрудниками кафедры института ГВИЭ на основе актинометрических данных, взятых из "Научно-прикладного справочника по климату СССР". Наблюдения в справочнике были получены от метеорологических станций, расположенных на территории бывшего СССР.

Актинометрические данные, используемые в web-приложении, необходимо импортировать из Excel-файла в реляционную базу данных MySQL. Приложение предоставляет пользователю возможность выбора актинометрической станции по одному из заданных параметров: название станции наблюдения, номер станции наблюдения в списке или географическая координата предполагаемой точки. При поиске по географическим координатам пользователь получит, как результат, выборку из трех ближайших к указанной точке станций. Приложение позволяет пользователю выбрать период времени, за который необходимо получить информацию по нужной станции, тип интерполяции и шаг дискретизации. Необходимость интерполировать возникает из-за того, что в базе данных информация хранится в виде среднемесячных значений. Полученные данные выводятся для просмотра в табличном виде. Результат можно экспортировать в формате CSV. Структура CSV-файла изначально задается сотрудниками института ГВИЭ.

Web-приложение необходимо выполнить по архитектуре клиент-сервер на основе модели представления контроллера (MVC). Данное web-приложение планируется разместить на сервере в ИВИЦ МЭИ. Всю обработку информации следует производить на сервере при помощи скриптов, написанных на языке Python с использованием фреймворка Django. Web-браузер нужен, чтобы сформировать запросы к серверу и отобразить подготовленные сервером HTML страницы. Дизайн страниц необходимо адаптировать к просмотру на устройствах с различными размерами экрана (компьютер, смартфон, планшет).

Исходные данные:

- используемые языки – Python, HTML, CSS, SQL;
- база данных – MySQL;
- веб-фреймворк – Django.

ПЕРЕЧЕНЬ ГРАФИЧЕСКОГО МАТЕРИАЛА

Количество слайдов в презентации

не менее 8

1. Структура базы данных
2. Схема алгоритма экспорта данных в формате CSV (ЕСПД)
3. Схема алгоритма интерполирования данных (ЕСПД)
4. Вид пользовательского интерфейса

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Доусон М. Програмируем на Python. – СПб.: Питер, 2016. – 416 с.: ил.
2. Грубер М. Понимание SQL. : Пер. с англ. В.Н. Лебедева – Москва, 1993. – 291 с.
3. Пауэлл Т.А. Полное руководство по HTML. : Пер. с англ. А.В. Качанов. – Мн. : ООО «Попурри», 2001. – 912 с.: ил.
4. Alchin M. Pro Django. – Apress, 2013. – 295 с.
5. Шелдон. Р., Мойе Дж. MySQL: Базовый курс. : Пер. с англ. – М.: ООО «И.Д. Вильямс», 2007. – 880 с.: ил.

Примечания:

1. Задание брошюруется вместе с выпускной работой после титульного листа (страницы задания имеют номера 2, 3).
2. Отзыв руководителя, рецензия(и), отчет о проверке на объем заимствований и согласие студента на размещение работы в открытом доступе вкладываются в конверт (файловую папку) под обложкой работы.

Приложение Б. Листинг программы серверной части веб-приложения

#Выпускная квалификационная работа студентки Палагиной Софьи Алексеевны

#Группа А-12-17

Листинг модуля "models.py" приложения "MainApp"

```
from django.db import models
```

```
#Создание модели соответствующей таблице - Станции
```

```
class Stations(models.Model):
```

```
    # Fields
```

```
    idStations = models.AutoField('ID'),
```

```
    NameStation = models.CharField('Название', max_length=50)
```

```
    Region = models.CharField('Регион', max_length=50)
```

```
    Latitude = models.DecimalField('Широта', max_digits=5, decimal_places=2)
```

```
    Longitude = models.DecimalField('Долгота', max_digits=5, decimal_places=2)
```

```
    def __str__(self):
```

```
        return self.NameStation
```

```
    class Meta:
```

```
        verbose_name = 'Stations'
```

```
        verbose_name_plural = 'Stations'
```

```
#Создание модели соответствующей таблице - Альбедо
```

```
class Albedo(models.Model):
```

```
    # Fields
```

```
    Month_1 = models.DecimalField('Месяц_1', max_digits=4, decimal_places=2)
```

```
    Month_2 = models.DecimalField('Месяц_2', max_digits=4, decimal_places=2)
```

```
    Month_3 = models.DecimalField('Месяц_3', max_digits=4, decimal_places=2)
```

```
    Month_4 = models.DecimalField('Месяц_4', max_digits=4, decimal_places=2)
```

```
    Month_5 = models.DecimalField('Месяц_5', max_digits=4, decimal_places=2)
```

```
    Month_6 = models.DecimalField('Месяц_6', max_digits=4, decimal_places=2)
```

```
    Month_7 = models.DecimalField('Месяц_7', max_digits=4, decimal_places=2)
```

```
    Month_8 = models.DecimalField('Месяц_8', max_digits=4, decimal_places=2)
```

```
    Month_9 = models.DecimalField('Месяц_9', max_digits=4, decimal_places=2)
```

```
    Month_10 = models.DecimalField('Месяц_10', max_digits=4, decimal_places=2)
```

```
    Month_11 = models.DecimalField('Месяц_11', max_digits=4, decimal_places=2)
```

```
Month_12 = models.DecimalField('Месяц_12', max_digits=4, decimal_places=2)
```

```
def __str__(self):  
    return self.AlbedoNew
```

```
class Meta:  
    verbose_name = 'Albedo'  
    verbose_name_plural = 'Albedo'
```

#Создание модели соответствующей таблице – Среднемесячные суточные суммы диффузной солнечной радиации

```
class sut_sum_diff_sr(models.Model):
```

```
    # Fields
```

```
    Month_1 = models.DecimalField('Месяц_1', max_digits=4, decimal_places=2)  
    Month_2 = models.DecimalField('Месяц_2', max_digits=4, decimal_places=2)  
    Month_3 = models.DecimalField('Месяц_3', max_digits=4, decimal_places=2)  
    Month_4 = models.DecimalField('Месяц_4', max_digits=4, decimal_places=2)  
    Month_5 = models.DecimalField('Месяц_5', max_digits=4, decimal_places=2)  
    Month_6 = models.DecimalField('Месяц_6', max_digits=4, decimal_places=2)  
    Month_7 = models.DecimalField('Месяц_7', max_digits=4, decimal_places=2)  
    Month_8 = models.DecimalField('Месяц_8', max_digits=4, decimal_places=2)  
    Month_9 = models.DecimalField('Месяц_9', max_digits=4, decimal_places=2)  
    Month_10 = models.DecimalField('Месяц_10', max_digits=4, decimal_places=2)  
    Month_11 = models.DecimalField('Месяц_11', max_digits=4, decimal_places=2)  
    Month_12 = models.DecimalField('Месяц_12', max_digits=4, decimal_places=2)
```

```
def __str__(self):  
    return self.sut_sum_diff_sr
```

```
class Meta:  
    verbose_name = 'sut_sum_diff_sr'  
    verbose_name_plural = 'sut_sum_diff_sr'
```

#Создание остальных моделей аналогично

#Выпускная квалификационная работа студентки Палагиной Софьи Алексеевны
#Группа А-12-17
Листинг модуля "apps.py" приложения "MainApp"

```
from django.apps import AppConfig
```

```
class MainAppConfig(AppConfig):  
    name = 'MainApp'
```

#Выпускная квалификационная работа студентки Палагиной Софьи Алексеевны
#Группа А-12-17
#Подключаемые библиотеки из модуля "views.py" приложения "MainApp"

```
import csv  
from django.http import HttpResponse  
from django.shortcuts import render  
from .models import StationsNew, mes_sum_sum_srNew, mes_sum_straight_srNew,  
mes_sum_diff_srNew, AlbedoNew, sut_sum_diff_srNew, sut_sum_sum_srNew,  
sut_sum_straight_srNew, hour_sum_sum_srNew, hour_sum_straight_srNew, hour_sum_diff_srNew  
from django.core.paginator import Paginator  
from django.core import serializers  
from geopy import distance  
import operator
```

#Выпускная квалификационная работа студентки Палагиной Софьи Алексеевны
#Группа А-12-17
Подпрограмма интерполяции по принципу «День сурка» из модуля "views.py"
приложения "MainApp"

```
    #Перебор и сортировка входных данных, заполнение массивов для дальнейшей их  
    обработки
```

```
        for object in object_list:  
            for field_name, field_value in object['fields'].items():
```



```

#Ищем по названию – ключу, значение записываем в массив
if field_name == 'Month_1': #заполнения массива данными за январь

    a11.append(float(field_value))
if field_name == 'Month_2': #заполнения массива данными за февраль

    a12.append(float(field_value))
if field_name == 'Month_3': #заполнения массива данными за март

    a13.append(float(field_value))
if field_name == 'Month_4': #заполнения массива данными за апрель

    a14.append(float(field_value))
if field_name == 'Month_5': #заполнения массива данными за май

    a15.append(float(field_value))
if field_name == 'Month_6': #заполнения массива данными за июнь

    a16.append(float(field_value))
if field_name == 'Month_7': #заполнения массива данными за июль

    a17.append(float(field_value))
if field_name == 'Month_8': #заполнения массива данными за август

    a18.append(float(field_value))
if field_name == 'Month_9': #заполнения массива данными за сентябрь

    a19.append(float(field_value))
if field_name == 'Month_10': #заполнения массива данными за октябрь

    a110.append(float(field_value))
if field_name == 'Month_11': #заполнения массива данными за ноябрь

    a111.append(float(field_value))

```

```

if field_name == 'Month_12': #заполнения массива данными за декабрь

    a112.append(float(field_value))

# Формирование выходной последовательности объединением массивов
    # «Склеиваем» все массивы в один при этом умножая их на коэффициенты,
соответствующе числу дней в месяце
    # Умножение массива на число эквивалентно его повторению указанное число раз
a = a11 * 31 + a12 * 28 + a13 * 31 + a14 * 30 + a15 * 31 + a16 * 30 + a17 * 31 + a18 * 31 + a19 * 30
+ a110 * 31 + a111 * 30 + a112 * 31

#Выпускная квалификационная работа студентки Палагиной Софьи Алексеевны
#Группа А-12-17
#Подпрограмма интерполяции по принципу «Линейная интерполяция» из модуля
"views.py" приложения "MainApp"

# Заолнение выходного массива для промежутка между 15-м января и 15-м февраля

    for i in range(24): #проводим данную операцию для каждого часа
raz = a12[i] - a11[i] #находим разницу между соседними элементами
delt = float(abs(raz) / 31) #находим шаг, знаменатель зависит от числа дней между двумя
точками
nach = a11[i] #запоминаем первую точку, относительно которой будем двигаться
    #сравниваем разность с нулем
if raz < 0: #если разность меньше нуля, значит приход солнечной радиации уменьшается
    for j in range(31):
        nach = nach - delt #следовательно, вычитаем шаг
        a21.append(float("{0:.3f}".format(nach))) #полученным значениями заполняем
выходной массив
elif raz > 0: #если разность больше нуля, значит приход солнечной радиации увеличивается
    for j in range(31):
        nach = nach + delt #следовательно, прибавляем шаг
        a21.append(float("{0:.3f}".format(nach)))
elif raz == 0: #если разность равна нулю, значит приход солнечной радиации не меняется

```

```

    for j in range(31):
        nach = nach + delt #просто прибавляем шаг
        a21.append(float("{0:.3f}".format(nach)))
    # процессы аналогичны
# Заолнение выходного массива для промежутка между 15-м февраля и 15-м марта

for i in range(24):
    raz = a13[i] - a12[i]
    delt = float(abs(raz) / 28)
    nach = a12[i]
    if raz < 0:
        for j in range(28):
            nach = nach - delt
            a22.append(float("{0:.3f}".format(nach)))
    elif raz > 0:
        for j in range(28):
            nach = nach + delt
            a22.append(float("{0:.3f}".format(nach)))
    elif raz == 0:
        for j in range(28):
            nach = nach + delt
            a22.append(float("{0:.3f}".format(nach)))
# Заолнение выходного массива для промежутка между 15-м марта и 15-м апреля
for i in range(24):
    raz = a14[i] - a13[i]
    delt = float(abs(raz) / 31)
    nach = a13[i]
    if raz < 0:
        for j in range(31):
            nach = nach - delt
            a23.append(float("{0:.3f}".format(nach)))
    elif raz > 0:
        for j in range(31):
            nach = nach + delt

```

```

        a23.append(float("{0:.3f}".format(nach)))
elif raz == 0:
    for j in range(31):
        nach = nach + delt
        a23.append(float("{0:.3f}".format(nach)))
# Заполнение выходного массива для промежутка между 15-м апреля и 15-м мая
for i in range(24):
    raz = a15[i] - a14[i]
    delt = float(abs(raz) / 30)
    nach = a14[i]
    if raz < 0:
        for j in range(30):
            nach = nach - delt
            a24.append(float("{0:.3f}".format(nach)))
    elif raz > 0:
        for j in range(30):
            nach = nach + delt
            a24.append(float("{0:.3f}".format(nach)))
    elif raz == 0:
        for j in range(30):
            nach = nach + delt
            a24.append(float("{0:.3f}".format(nach)))
# Заполнение выходного массива для промежутка между 15-м мая и 15-м июня
for i in range(24):
    raz = a16[i] - a15[i]
    delt = float(abs(raz) / 31)
    nach = a15[i]
    if raz < 0:
        for j in range(31):
            nach = nach - delt
            a25.append(float("{0:.3f}".format(nach)))
    elif raz > 0:
        for j in range(31):
            nach = nach + delt

```

```

        a25.append(float("{0:.3f}".format(nach)))
elif raz == 0:
    for j in range(31):
        nach = nach + delt
        a25.append(float("{0:.3f}".format(nach)))
# Заполнение выходного массива для промежутка между 15-м июня и 15-м июля
for i in range(24):
    raz = a17[i] - a16[i]
    delt = float(abs(raz) / 30)
    nach = a16[i]
    if raz < 0:
        for j in range(30):
            nach = nach - delt
            a26.append(float("{0:.3f}".format(nach)))
    elif raz > 0:
        for j in range(30):
            nach = nach + delt
            a26.append(float("{0:.3f}".format(nach)))
    elif raz == 0:
        for j in range(30):
            nach = nach + delt
            a26.append(float("{0:.3f}".format(nach)))
# Заполнение выходного массива для промежутка между 15-м июля и 15-м августа
for i in range(24):
    raz = a18[i] - a17[i]
    delt = float(abs(raz) / 31)
    nach = a17[i]
    if raz < 0:
        for j in range(31):
            nach = nach - delt
            a27.append(float("{0:.3f}".format(nach)))
    elif raz > 0:
        for j in range(31):
            nach = nach + delt

```

```

        a27.append(float("{0:.3f}".format(nach)))
elif raz == 0:
    for j in range(31):
        nach = nach + delt
        a27.append(float("{0:.3f}".format(nach)))
# Заполнение выходного массива для промежутка между 15-м августа и 15-м сентября
for i in range(24):
    raz = a19[i] - a18[i]
    delt = float(abs(raz) / 31)
    nach = a18[i]
    if raz < 0:
        for j in range(31):
            nach = nach - delt
            a28.append(float("{0:.3f}".format(nach)))
    elif raz > 0:
        for j in range(31):
            nach = nach + delt
            a28.append(float("{0:.3f}".format(nach)))
    elif raz == 0:
        for j in range(31):
            nach = nach + delt
            a28.append(float("{0:.3f}".format(nach)))
# Заполнение выходного массива для промежутка между 15-м сентября и 15-м октября
for i in range(24):
    raz = a110[i] - a19[i]
    delt = float(abs(raz) / 30)
    nach = a19[i]
    if raz < 0:
        for j in range(30):
            nach = nach - delt
            a29.append(float("{0:.3f}".format(nach)))
    elif raz > 0:
        for j in range(30):
            nach = nach + delt

```

```

        a29.append(float("{0:.3f}".format(nach)))
elif raz == 0:
    for j in range(30):
        nach = nach + delt
        a29.append(float("{0:.3f}".format(nach)))
# Заолнение выходного массива для промежутка между 15-м октября и 15-м ноября

```

```

        for i in range(24):
            raz = a111[i] - a110[i]
            delt = float(abs(raz) / 31)
            nach = a110[i]
            if raz < 0:
                for j in range(31):
                    nach = nach - delt
                    a210.append(float("{0:.3f}".format(nach)))
            elif raz > 0:
                for j in range(31):
                    nach = nach + delt
                    a210.append(float("{0:.3f}".format(nach)))
            elif raz == 0:
                for j in range(31):
                    nach = nach + delt
                    a210.append(float("{0:.3f}".format(nach)))
# Заолнение выходного массива для промежутка между 15-м ноября и 15-м декабря

```

```

for i in range(24):
    raz = a112[i] - a111[i]
    delt = float(abs(raz) / 30)
    nach = a111[i]
    if raz < 0:
        for j in range(30):
            nach = nach - delt
            a211.append(float("{0:.3f}".format(nach)))
    elif raz > 0:

```

```

        for j in range(30):
            nach = nach + delt
            a211.append(float("{0:.3f}".format(nach)))
elif raz == 0:
    for j in range(30):
        nach = nach + delt
        a211.append(float("{0:.3f}".format(nach)))
# Заполнение выходного массива для промежутка между 15-м декабря и 15-м января

```

```

for i in range(24):
    raz = a11[i] - a112[i]
    delt = float(abs(raz) / 31)
    nach = a112[i]
    if raz < 0:
        for j in range(31):
            nach = nach - delt
            a212.append(float("{0:.3f}".format(nach)))
    elif raz > 0:
        for j in range(31):
            nach = nach + delt
            a212.append(float("{0:.3f}".format(nach)))
    elif raz == 0:
        for j in range(31):
            nach = nach + delt
            a212.append(float("{0:.3f}".format(nach)))

```

#разбиваем поледний массив, одна его часть, соответствующая промежутку с 1-го января по 15-е января, ставится в начало выходной последовательности, а вторая, соответствующая промежутку с 15-го декабря по 31-декабря, в конец

```

end = []
for i in range(336):
    end.append(a212[i])

```



```
begin = []  
for i in range(336, 744):  
    begin.append(a212[i])
```

#формирование выходной последовательности посредством объединения всех массивов в нужном порядке

```
a = begin + a21 + a22 + a23 + a24 + a25 + a26 + a27 + a28 + a29 + a210 + a211 + end
```

#Выпускная квалификационная работа студентки Палагиной Софьи Алексеевны

#Группа А-12-17

#Подпрограмма экспорта в формат csv из модуля "views.py" приложения "MainApp"

```
def Export_csv(request):  
  
    #получение параметров от других представлений  
    res0 = request.session.get('res3')  
    res = request.session.get('res2')  
    coord = request.session.get('coord')  
    res1 = request.session.get('res2_num')  
    res2 = request.session.get('res_coord_1')  
    res3 = request.session.get('res_coord_2')  
    res4 = request.session.get('res_coord_3')  
    Export = request.GET.get("export")  
  
    response = HttpResponse(content_type='text/csv') #экспортируем в формате csv  
    writer = csv.writer(response)  
    writer.writerow(['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12'])  
  
    #если пользователь нажал на кнопку экспорт Альбедо  
    if Export == 'Albedo':  
        if res0:
```

```

        response['Content-Disposition'] = 'attachment; filename = "Typical_Albedo12.csv"'
#экспортируем с названием "Typical_Albedo12.csv"

        obj = next(serializers.deserialize("json", res0)).object #сериализуем данные для вывода

        writer.writerow([obj.Month_1, obj.Month_2, obj.Month_3, obj.Month_4, obj.Month_5,
obj.Month_6, obj.Month_7, obj.Month_8, obj.Month_9, obj.Month_10, obj.Month_11,
obj.Month_12])

#получение id
if res: #если пришло в параметре название станции
    r = StationsNew.objects.get(NameStation=res)
    #если пришло в параметре id станции
elif res1:
    r = StationsNew.objects.get(id=res1)
elif coord == '1':
    if res2:
        r = StationsNew.objects.get(id=res2)
elif coord == '2':
    if res3:
        r = StationsNew.objects.get(id=res3)
elif coord == '3':
    if res4:
        r = StationsNew.objects.get(id=res4)

#если пользователь выбрал экспорт среднемесячные суммы суммарной/прямой/диффузной
солнечной радиации
if Export == 'Sum': #если суммарная
    response['Content-Disposition'] = 'attachment; filename = "Typical_GlobalHI24.csv"'
#указываем название
    stat = hour_sum_sum_srNew.objects.filter(IdStations=r.id) #получаем соответствующие
данные

```

```

if Export == 'Straight': #если прямая
    response['Content-Disposition'] = 'attachment; filename = "Typical_DirectHI24.csv"#указываем
название
    stat = hour_sum_straight_srNew.objects.filter(IdStations=r.id)

if Export == 'Diff': #если диффузная
    response['Content-Disposition'] = 'attachment; filename =
"Typical_DiffuseHI24.csv"#указываем название
    stat = hour_sum_diff_srNew.objects.filter(IdStations=r.id)

for row in stat:
    writer.writerow([row.IdHour, row.Month_1, row.Month_2, row.Month_3, row.Month_4,
row.Month_5, row.Month_6, row.Month_7, row.Month_8, row.Month_9, row.Month_10,
row.Month_11, row.Month_12])

return response

```

#Выпускная квалификационная работа студентки Палагиной Софьи Алексеевны

#Группа А-12-17

#Подпрограмма экспорта в формате csv интерполированных данных из модуля "views.py" приложения "MainApp"

```

def Ex(request):
    #получаем данные от других представлений
    res = request.session.get('res2')
    coord = request.session.get('coord')
    res1 = request.session.get('res2_num')
    error = ""
#получение id в зависимости от прешедших параметров
    if res:
        r = StationsNew.objects.get(NameStation=res)

```

```

if res1:
    r = StationsNew.objects.get(id=res1)
if coord == '1':
    res2 = request.session.get('res_coord_1')
    r = StationsNew.objects.get(id=res2)
if coord == '2':
    res3 = request.session.get('res_coord_2')
    r = StationsNew.objects.get(id=res3)
if coord == '3':
    res4 = request.session.get('res_coord_3')
    r = StationsNew.objects.get(id=res4)

    #получаем исходные данные по id от требуемых моделей
stat = AlbedoNew.objects.filter(id=r.id)
stat1 = hour_sum_sum_srNew.objects.filter(IdStations=r.id)
stat2 = hour_sum_straight_srNew.objects.filter(IdStations=r.id)
stat3 = hour_sum_diff_srNew.objects.filter(IdStations=r.id)

#подготовка второстепенных массивов для вывода выходного файла

    #столбец Hour
hour = []
for i in range(365):
    for j in range(1, 25):
        hour.append(j)

#столбец день
day = []
for i in range(1, 13):
    if i in [1, 3, 5, 7, 8, 10, 12]:
        for j in range(1, 32):
            for k in range(24):
                day.append(j)
    elif i in [4, 6, 9, 11]:

```

```

        for j in range(1, 31):
            for k in range(24):
                day.append(j)
    elif i == 2:
        for j in range(1, 29):
            for k in range(24):
                day.append(j)
#столбец месяц
month = []
for i in range(1, 13):
    if i in [1, 3, 5, 7, 8, 10, 12]:
        for j in range(31):
            for k in range(24):
                month.append(i)
    elif i in [4, 6, 9, 11]:
        for j in range(30):
            for k in range(24):
                month.append(i)
    elif i == 2:
        for j in range(28):
            for k in range(24):
                month.append(i)

#столбец день в году
DOY = []
for i in range(1, 366):
    for j in range(24):
        DOY.append(i)

#столбец час в году
HOY = []
for i in range(1, 8761):
    HOY.append(i)

```

```

ind = []
for i in range(4380):
    ind.append(i)

per = []
for i in range(8760):
    per.append(i)
del per[::2]

#сериализуем входные данные для удобного перебора
object_list = serializers.serialize("python", stat)
object_list1 = serializers.serialize("python", stat1)
object_list2 = serializers.serialize("python", stat2)
object_list3 = serializers.serialize("python", stat3)

#получаем входные параметры от пользовательской формы
Period = request.GET.get('Period') #период, за который берем данные
Type = request.GET.get('Type') #тип интерполяции, который будем применять
Hour = request.GET.get('Hour') #шаг дискретизации, с которым будем идти

# Тип интерполяции - "День сурка"
if Type == 'Type_surok':

    # заполнение массивов
    # процесс интерполирования
    # получаем выходные массивы заполненные интерполированными данными

    # Данные за весь период
    if Period == 'All_period':

        # Шаг дискретизации 1 час
        if Hour == 'Hour_1':

            response = HttpResponse(content_type='text/csv')

```

```

writer = csv.writer(response)
response['Content-Disposition'] = 'attachment; filename = "Irradiation_8760.csv"'

writer.writerow([' ', 'HOY', 'DOY', 'Month', 'Day', 'Hour', 'GHI', 'DHI', 'DirectHI', 'Albedo'])

for i in range(8760): #берем все элементы массивов
    writer.writerow([i, HOY[i], DOY[i], month[i], day[i], hour[i], a1[i], a3[i], a2[i], a[i]])

# Шаг дискретизации 2 часа
if Hour == 'Hour_2':

    response = HttpResponse(content_type='text/csv')
    writer = csv.writer(response)
    response['Content-Disposition'] = 'attachment; filename = "Irradiation_8760.csv"'

    writer.writerow([' ', 'HOY', 'DOY', 'Month', 'Day', 'Hour', 'GHI', 'DHI', 'DirectHI', 'Albedo'])

    j = 0
    for i in per: #берем каждый второй элемент из массивов
        writer.writerow([j, HOY[i], DOY[i], month[i], day[i], hour[i], a1[i], a3[i], a2[i], a[i]])
        j = j+1

# Данные за указанный интервал
if Period == 'Part_Period':

    #высчитываем индекс, с которого необходимо начать брать данные
    Mes_begin = request.GET.get('Mes_begin')
    Day_begin = request.GET.get('Day_begin')

    #обработка случая, когда не указан день или часа
    if Day_begin == "":
        Day_begin = 1
    Hour_begin = request.GET.get('Hour_begin')

```

```

if Hour_begin == "":
    Hour_begin = 1

#высчитываем индекс, на котором необходимо завершить брать данные
Mes_end = request.GET.get('Mes_end')
Day_end = request.GET.get('Day_end')

#обработка случая, когда не указан день или часа
#в зависимости от месяца берем последний день

if int(Mes_end) in [1, 3, 5, 7, 8, 10, 12]:
    Day_end = 31
if int(Mes_end) == 2:
    Day_end = 28
if int(Mes_end) in [4, 6, 9, 11]:
    Day_end = 30

Hour_end = request.GET.get('Hour_end')
if Hour_end == "":
    Hour_end = 24

#проверка на корректность указания интервала
if int(Mes_begin) > int(Mes_end):
    return render(request, 'MainApp/Error.html')

if int(Mes_begin) == int(Mes_end):
    if int(Day_begin) > int(Day_end):
        return render(request, 'MainApp/Error.html')

if int(Mes_begin) == int(Mes_end):
    if int(Day_begin) == int(Day_end):
        if int(Hour_begin) > int(Hour_end):
            return render(request, 'MainApp/Error.html')

```



```

sum_begin = 0
sum_end = 0

for i in range(1, int(Mes_begin)):
    if i in [1, 3, 5, 7, 8, 10, 12]:
        sum_begin = sum_begin + 744
    if i == 2:
        sum_begin = sum_begin + 672
    if i in [4, 6, 9, 11]:
        sum_begin = sum_begin + 720

sum_begin = sum_begin + int((int(Day_begin)-1)*24)
sum_begin = sum_begin + int(Hour_begin)

for i in range(1, int(Mes_end)):
    if i in [1, 3, 5, 7, 8, 10, 12]:
        sum_end = sum_end + 744
    if i == 2:
        sum_end = sum_end + 672
    if i in [4, 6, 9, 11]:
        sum_end = sum_end + 720

sum_end = sum_end + int((int(Day_end)-1)*24)
sum_end = sum_end + int(Hour_end)

if sum_begin < sum_end:

    # Шаг дискретизации 1 час
    if Hour == 'Hour_1':

        response = HttpResponse(content_type='text/csv')
        writer = csv.writer(response)

```

```

response['Content-Disposition'] = 'attachment; filename = "Irradiation_8760.csv"'

writer.writerow([' ', 'HOY', 'DOY', 'Month', 'Day', 'Hour', 'GHI', 'DHI', 'DirectHI',
'Albedo'])
j = 0
for i in range(sum_begin-1, sum_end):
    writer.writerow([j, HOY[i], DOY[i], month[i], day[i], hour[i], a1[i], a3[i], a2[i], a[i]])
    j = j+1

# Шаг дискретизации 2 часа
if Hour == 'Hour_2':

    response = HttpResponse(content_type='text/csv')
    writer = csv.writer(response)
    response['Content-Disposition'] = 'attachment; filename = "Irradiation_8760.csv"'

    writer.writerow([' ', 'HOY', 'DOY', 'Month', 'Day', 'Hour', 'GHI', 'DHI', 'DirectHI',
'Albedo'])
    j = 0
    for i in per:
        if i in range(sum_begin-1, sum_end+1):
            writer.writerow([j, HOY[i], DOY[i], month[i], day[i], hour[i], a1[i], a3[i], a2[i], a[i]])
            j = j+1
    else:
        error = 'e'
        return render(request, 'MainApp/Export.html', context={'error':error})
    error = ""

# Тип интерполяции - линейная
if Type == 'Type_lin':

    # заполнение массивов

    # процесс интерполирования

```

```

# получаем выходные массивы заполненные интерполированными данными

# За весь период
if Period == 'All_period':

    # Шаг дискретизации 1 час
    if Hour == 'Hour_1':

        response = HttpResponse(content_type='text/csv')
        writer = csv.writer(response)
        response['Content-Disposition'] = 'attachment; filename = "Irradiation_8760.csv"'

        writer.writerow([' ', 'HOY', 'DOY', 'Month', 'Day', 'Hour', 'GHI', 'DHI', 'DirectHI', 'Albedo'])

        for i in range(8760):
            writer.writerow([i, HOY[i], DOY[i], month[i], day[i], hour[i], a1[i], a3[i], a2[i], a[i]])

    # Шаг дискретизации 2 часа
    if Hour == 'Hour_2':

        response = HttpResponse(content_type='text/csv')
        writer = csv.writer(response)
        response['Content-Disposition'] = 'attachment; filename = "Irradiation_8760.csv"'

        writer.writerow([' ', 'HOY', 'DOY', 'Month', 'Day', 'Hour', 'GHI', 'DHI', 'DirectHI', 'Albedo'])

        j = 0
        for i in per:
            writer.writerow([j, HOY[i], DOY[i], month[i], day[i], hour[i], a1[i], a3[i], a2[i], a[i]])
            j = j + 1

# Данные за указанный интервал
if Period == 'Part_Period':

```

```

        #получение индекса начала
        Mes_begin = request.GET.get('Mes_begin')
        Day_begin = request.GET.get('Day_begin')
        #обработка случая, когда не указан день или час
        if Day_begin == "":
            Day_begin = 1
        Hour_begin = request.GET.get('Hour_begin')
        if Hour_begin == "":
            Hour_begin = 1

        #получение индекса завершения
        Mes_end = request.GET.get('Mes_end')
        Day_end = request.GET.get('Day_end')

        #обработка случая, когда не указан день или час
        if int(Mes_end) in [1, 3, 5, 7, 8, 10, 12]:
            Day_end = 31
        if int(Mes_end) == 2:
            Day_end = 28
        if int(Mes_end) in [4, 6, 9, 11]:
            Day_end = 30
        Hour_end = request.GET.get('Hour_end')
        if Hour_end == "":
            Hour_end = 24

        #проверка на корректность указания интервала
        if int(Mes_begin) > int(Mes_end):
            return render(request, 'MainApp/Error.html')

        if int(Mes_begin) == int(Mes_end):
            if int(Day_begin) > int(Day_end):
                return render(request, 'MainApp/Error.html')

```

```

        if int(Mes_begin) == int(Mes_end):
            if int(Day_begin) == int(Day_end):
                if int(Hour_begin) > int(Hour_end):
                    return render(request, 'MainApp/Error.html')

sum_begin = 0
sum_end = 0

for i in range(1, int(Mes_begin)):
    if i in [1, 3, 5, 7, 8, 10, 12]:
        sum_begin = sum_begin + 744
    if i == 2:
        sum_begin = sum_begin + 672
    if i in [4, 6, 9, 11]:
        sum_begin = sum_begin + 720

sum_begin = sum_begin + int((int(Day_begin) - 1) * 24)
sum_begin = sum_begin + int(Hour_begin)

for i in range(1, int(Mes_end)):
    if i in [1, 3, 5, 7, 8, 10, 12]:
        sum_end = sum_end + 744
    if i == 2:
        sum_end = sum_end + 672
    if i in [4, 6, 9, 11]:
        sum_end = sum_end + 720

sum_end = sum_end + int((int(Day_end) - 1) * 24)
sum_end = sum_end + int(Hour_end)

# Шаг дискретизации 1 час
if Hour == 'Hour_1':

```

```

response = HttpResponse(content_type='text/csv')
writer = csv.writer(response)
response['Content-Disposition'] = 'attachment; filename = "Irradiation_8760.csv"'

writer.writerow([' ', 'HOY', 'DOY', 'Month', 'Day', 'Hour', 'GHI', 'DHI', 'DirectHI', 'Albedo'])
j = 0
for i in range(sum_begin - 1, sum_end):
    writer.writerow([j, HOY[i], DOY[i], month[i], day[i], hour[i], a1[i], a3[i], a2[i], a[i]])
    j = j + 1

# Шаг дискретизации 2 часа
if Hour == 'Hour_2':

    response = HttpResponse(content_type='text/csv')
    writer = csv.writer(response)
    response['Content-Disposition'] = 'attachment; filename = "Irradiation_8760.csv"'

    writer.writerow([' ', 'HOY', 'DOY', 'Month', 'Day', 'Hour', 'GHI', 'DHI', 'DirectHI', 'Albedo'])
    j = 0
    for i in per:
        if i in range(sum_begin - 1, sum_end + 1):
            writer.writerow([j, HOY[i], DOY[i], month[i], day[i], hour[i], a1[i], a3[i], a2[i], a[i]])
            j = j + 1

    return response

```

#Выпускная квалификационная работа студентки Палагиной Софьи Алексеевны

#Группа А-12-17

#Подпрограмма расчета трех ближайших станций из модуля "views.py" приложения "MainApp"

```

search_query_sr = request.GET.get('search_coord_sr', '') #получаем широту из формы
search_query_dl = request.GET.get('search_coord_dl', '') #получаем долготу из формы
coords_1 = (float(search_query_sr), float(search_query_dl)) #получаем координату введенной

```

точки

```
object_list = serializers.serialize("python", StationsNew.objects.all())
```

```
stat = []
```

```
name={}
```

```
name2 = {}
```

```
#Перебираем координаты всех станций
```

```
i=1
```

```
for object in object_list:
```

```
    for field_name, field_value in object['fields'].items():
```

```
        if field_name == 'Latitude':
```

```
            a = field_value
```

```
        if field_name == 'Longitude':
```

```
            b = field_value
```

```
    coords_2 = (float(a), float(b))
```

```
#Рассчитываем расстояние между указанной точкой и текущей станцией
```

```
dist = distance.distance(coords_1, coords_2).km
```

```
#Результат помещаем в массив
```

```
    name[i] = dist
```

```
    i=i+1
```

```
#Сортируем полученный массив по значению расстояния в порядке возрастания и берем первые  
три значения
```

```
name2 = sorted(name.items(), key=operator.itemgetter(1))[:3]
```

```
n1 = (name2[0])[0]
```

```
n2 = (name2[1])[0]
```

```
n3 = (name2[2])[0]
```

```
request.session['res_coord_1'] = n1
```

```
request.session['res_coord_2'] = n2
```

```
request.session['res_coord_3'] = n3
```

```
#Получаем данные по трем станциям
```

```
stat1 = StationsNew.objects.get(id=n1)
```

```
stat2 = StationsNew.objects.get(id=n2)
```

```
stat3 = StationsNew.objects.get(id=n3)
```

```
return render(request, 'MainApp/Show.html', context={ 'n1':format((name2[0])[1], '.2f'),  
'n2':format((name2[1])[1], '.2f'),'n3':format((name2[2])[1], '.2f'), 'res':stat1, 'res2':stat2, 'res3':stat3,  
'l':search_query_dl, 's':search_query_sr})
```

#Выпускная квалификационная работа студентки Палагиной Софьи Алексеевны

#Группа А-12-17

**#Подпрограмма вывода результата поиска данных из модуля "views.py" приложения
"MainApp"**

```
#Получаем информацию из формы
```

```
search_query = request.GET.get('search', "")
```

```
request.session['res2'] = search_query
```

```
error = "
```

```
search_query2 = request.GET.get('search_num', "")
```

```
request.session['res2_num'] = search_query2
```

```
Coord = request.GET.get("search_coord")
```

```
request.session['coord'] = Coord
```

```
object_list = serializers.serialize("python", StationsNew.objects.all())
```

```
Name = []
```

```
for object in object_list:
```

```
    for field_name, field_value in object['fields'].items():
```

```
        if field_name == 'NameStation':
```

```
            Name.append(field_value)
```

```
#Если пришло название станции
```

```
if search_query:
```



```

stat = StationsNew.objects.filter(NameStation=search_query)
ser = serializers.serialize("json", stat)
request.session['res1'] = ser
#По названию получаем id станции
r = StationsNew.objects.select_related('id').filter(NameStation=search_query)[:1]

if search_query in Name:
    error = 'e'
else:
    error = 'Станция не найдена!'

#Если пришел номер станции
elif search_query2:

#Получаем данные по этому номеру
stat = StationsNew.objects.filter(id=search_query2)
ser = serializers.serialize("json", stat)
request.session['res1'] = ser
r = search_query2

if search_query2:
    error = 'e'
else:
    error = 'Станция не найдена!'

#Если пришли координаты станции
#Если пользователь выбрал самую ближайшую станцию, то берем ее координаты
elif Coord == '1':
    search_query_coord_1 = request.session.get('res_coord_1', "")
#По координатам ищем id станции
stat = StationsNew.objects.filter(id=search_query_coord_1)
ser = serializers.serialize("json", stat)
request.session['res1'] = ser
r = search_query_coord_1

```

```

if search_query_coord_1:
    error = 'e'
else:
    error = 'Станция не найдена!'
#Если пользователь выбрал вторую по дальности станцию, то берем ее координаты
elif Coord == '2':
    search_query_coord_2 = request.session.get('res_coord_2', "")

    stat = StationsNew.objects.filter(id=search_query_coord_2)
    ser = serializers.serialize("json", stat)
    request.session['res1'] = ser
    r = search_query_coord_2
    if search_query_coord_2:
        error = 'e'
    else:
        error = 'Станция не найдена!'
#Если пользователь выбрал самую дальнюю станцию, то берем ее координаты
elif Coord == '3':
    search_query_coord_3 = request.session.get('res_coord_3', "")

    stat = StationsNew.objects.filter(id=search_query_coord_3)
    ser = serializers.serialize("json", stat)
    request.session['res1'] = ser
    r = search_query_coord_3
    if search_query_coord_3:
        error = 'e'
    else:
        error = 'Станция не найдена!'
#По найденному id получаем данные по станции сос всех таблиц
mes_sum = mes_sum_sum_srNew.objects.filter(id=r)
mes_straight = mes_sum_straight_srNew.objects.filter(id=r)
mes_diff = mes_sum_diff_srNew.objects.filter(id=r)

```

```

sut_sum = sut_sum_sum_srNew.objects.filter(id=r)
sut_straight = sut_sum_straight_srNew.objects.filter(id=r)
sut_diff = sut_sum_diff_srNew.objects.filter(id=r)

hour_sum = hour_sum_sum_srNew.objects.filter(IdStations=r)
hour_straight = hour_sum_straight_srNew.objects.filter(IdStations=r)
hour_diff = hour_sum_diff_srNew.objects.filter(IdStations=r)

albedo = AlbedoNew.objects.filter(id=r)

ser1 = serializers.serialize("json", albedo)
request.session['res3'] = ser1

context = {
    'res' : stat,
    'res_mes_sum' : mes_sum,
    'res_mes_straight' : mes_straight,
    'res_mes_diff' : mes_diff,
    'res_sut_sum' : sut_sum,
    'res_sut_straight' : sut_straight,
    'res_sut_diff' : sut_diff,
    'res_hour_sum' : hour_sum,
    'res_hour_straight' : hour_straight,
    'res_hour_diff' : hour_diff,
    'albedo': albedo,
    'error': error,
}
return render(request, 'MainApp/SearchResult.html', context= context)

```

#Выпускная квалификационная работа студентки Палагиной Софьи Алексеевны

#Группа А-12-17

#Подпрограмма вывода информации по всем станциям из модуля "views.py" приложения "MainApp"

```
stat = StationsNew.objects.all()
paginator = Paginator(stat, 25)
page_number = request.GET.get('page', 1)
page = paginator.get_page(page_number)

is_paginated = page.has_other_pages()

if page.has_previous():
    prev_url = '?page={}'.format(page.previous_page_number())
else:
    prev_url = ""

if page.has_next():
    next_url = '?page={}'.format(page.next_page_number())
else:
    next_url = ""

context = {
    'page_object': page,
    'is_paginated': is_paginated,
    'next_url': next_url,
    'prev_url': prev_url
}

return render(request, 'MainApp/ShowStations.html', context=context)
```

Приложение В. Листинг программы клиентской части веб-приложения

#Выпускная квалификационная работа студентки Палагиной Софьи Алексеевны

#Группа А-12-17

#Листинг модуля "base.html" приложения "MainApp"

```
<!doctype html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
        content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0,
minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>{% block title %} {% endblock %}</title>
    {% load static %}
    <link rel="stylesheet" href="{% static 'MainApp/css/bootstrap.css' %}">
    <link rel="stylesheet" href="{% static 'MainApp/css2/main.css' %}">
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
integrity="sha384-
UO2eT0CpHqdsJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
crossorigin="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
crossorigin="anonymous"></script>
</head>

<body class="position-relative">
<div class="container " wfd-id="78">
    <header class="d-flex flex-wrap justify-content-center my-3 border-bottom">
        <a href="/" class="d-flex align-items-center mb-3 mb-md-0 me-md-auto text-dark text-decoration-
none py-md-2 mx-4">
            

</a>

<ul class="nav nav-pills my-3 align-content-center" wfd-id="79">

<li class="page-item" wfd-id="84">

<a href="{ % url 'ShowStations' % }" class="page-link btn-light "><font style="vertical-align: inherit;"><font style="vertical-align: inherit;">

Список станций</font></font>

</a>

</li>

</ul>

</header>

<div class="container" style="min-height: 30rem" wfd-id="78">

<div class="mx-1 d-sm-flex flex-wrap justify-content-center">

<aside class="my-3 d-flex justify-content-center align-items-sm-start " style="min-width: 10rem; max-width: 25rem">

<div class="d-flex flex-column px-3 mx-2 py-3 bg-light pb-4 rounded" wfd-id="60">

<a href="/" class="d-flex align-items-center text-center justify-content-center link-dark text-decoration-none">

<span style="font-size: 1rem" wfd-id="69"><font style="vertical-align: inherit;"><font style="vertical-align: inherit;">Поиск станции</font></font></span>

</a>

<hr class="hr">

<ul style="font-size: 0.9rem" class=" nav nav-pills flex-column ms-md-auto me-md-auto w-100" wfd-id="63">

<li wfd-id="67">

<a href="{ % url 'FindName' % }" class="nav-link link-dark" style="width: 12rem">

<font style="vertical-align: inherit;"><font style="vertical-align: inherit;">

Поиск по названию

</font></font></a>

```


<li wfd-id="66">

 Поиск по номеру

 <li wfd-id="65">

 Поиск по координатам

 <hr class="hr" style="height: 0.5px">
</div>
</aside>

<main class="d-flex flex-wrap ms-md-auto me-md-auto align-items-start flex-fill justify-content-
center w-auto" style="min-width: 5rem; max-width: 900px">

 {% block content %}
 {% endblock %}

</main>

</div>
</div>
</div>

<div class="container navbar-fixed-bottom ">
 <div class="navbar-inner">
 <footer class=" modal-footer col-12 bg-white">

```



```

 <p style="font-size: 0.7rem">Палагина Софья</p>

 <p style="font-size: 0.7rem">2021</p>
 </footer>
</div>
</div>
</body>
</html>

```

**#Выпускная квалификационная работа студентки Палагиной Софьи Алексеевны**

**#Группа А-12-17**

**#Листинг модуля "Export.html" приложения "MainApp"**

```

{%extends 'MainApp/base3.html'%}
{%block title%}
Поиск по номеру
{% endblock %}

```

```

{%block content%}

```

```

<form class = "form-control" action="{% url 'Ex' %}" method="GET">
 {% csrf_token %}

 <div class="container align-items-center justify-content-center d-flex flex-column w-100">

 <span class="container shadow-sm p-2 my-3 bg-light rounded justify-content-center lead row
text-center fs-6" >
 Выберите период, за который Вы хотите получить данные

 <div class="container">
 <div class="d-flex flex-wrap justify-content-center ">
 <div class="custom-control custom-radio">
 <input type="radio" id="All_period" value="All_period" name="Period" class="custom-
control-input" required>

```

```
<label class="custom-control-label" style="font-size: 0.9rem" for="All_period">За весь
период наблюдений</label>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<div class="container">
```

```
<div class="d-flex flex-wrap justify-content-center ">
```

```
<p class="mb-2 align-content-end text-muted ">
```

```
<div class="custom-control custom-radio mb-1" >
```

```
<input type="radio" id="Part_Period" value="Part_Period" name="Period"
class="custom-control-input" data-toggle="collapse" data-target="#collapseExample" aria-
expanded="false" aria-controls="collapseExample" required>
```

```
<label class="custom-control-label" style="font-size: 0.9rem" for="Part_Period">Задать
временной интервал</label>
```

```
</div>
```

```
</p>
```

```
<div class="collapse w-100" id="collapseExample">
```

```
<div class="card card-body" >
```

```
<div class="d-flex flex-row justify-content-center mw-100">
```

```
<div class="justify-content-center d-flex flex-column row w-auto mx-4 px-5 my-1 ">
```

```
Начиная с:
```

```
<input type="number" step="any" style="font-size: 0.8rem" class="input-group my-2 justify-
content-center mx-auto text-center" style="border: solid #b1b1b1 1px; color: #777A7A"
placeholder="Месяц" name="Mes_begin" min="1" max="12" required>
```

```
<input type="number" step="any" style="font-size: 0.8rem" class="input-group my-2 justify-
content-center mx-auto text-center" style="border: solid #b1b1b1 1px; color: #777A7A"
placeholder="День" name="Day_begin" min="1" max="31" >
```

```
<input type="number" step="any" style="font-size: 0.8rem" class="input-group my-2 justify-
content-center mx-auto text-center" style="border: solid #b1b1b1 1px; color: #777A7A"
placeholder="Час" name="Hour_begin" min="1" max="24" >
```

```
</div>
```

```

<div class="justify-content-center d-flex flex-column row w-auto mx-4 my-1 px-5">
 Заканчивая на:
 <input type="number" step="any" style="font-size: 0.8rem" class="input-group my-2 justify-
content-center mx-auto text-center" style="border: solid #b1b1b1 1px; color: #777A7A"
placeholder="Месяц" name="Mes_end" min="1" max="12" required>
 <input type="number" step="any" style="font-size: 0.8rem" class="input-group my-2 justify-
content-center mx-auto text-center" style="border: solid #b1b1b1 1px; color: #777A7A"
placeholder="День" name="Day_end" min="1" max="31" >
 <input type="number" step="any" style="font-size: 0.8rem" class="input-group my-2 justify-
content-center mx-auto text-center" style="border: solid #b1b1b1 1px; color: #777A7A"
placeholder="Час" name="Hour_end" min="1" max="24" >
</div>
</div>

```

```

</div>
</div>
</div>
</div>

```

```

<span class="container shadow-sm p-2 my-3 bg-light rounded justify-content-center lead row
text-center fs-6" >

```

```

 Выберите тип интерполяции


```

```

<div class="justify-content-start d-flex flex-column">
 <div class="custom-control custom-radio">
 <input type="radio" id="Type_surok" value="Type_surok" name="Type" class="custom-
control-input" required>
 <label class="custom-control-label" style="font-size: 0.9rem" for="Type_surok">"День
супка"</label>
 </div>
 <div class="custom-control custom-radio">
 <input type="radio" id="Type_lin" value="Type_lin" name="Type" class="custom-control-
input" required>

```

```

 <label class="custom-control-label" style="font-size: 0.9rem" for="Type_lin">"Линейная
интерполяция"</label>
 </div>
</div>

<span class="container shadow-sm p-2 my-3 bg-light rounded justify-content-center lead row
text-center fs-6" >
 Выберите шаг дискретизации

<div class="d-flex flex-column justify-content-start">
 <div class="custom-control custom-radio">
 <input type="radio" id="Hour_1" value="Hour_1" name="Hour" class="custom-control-
input" required>
 <label class="custom-control-label" style="font-size: 0.9rem" for="Hour_1">1 час</label>
 </div>
 <div class="custom-control custom-radio">
 <input type="radio" id="Hour_2" value="Hour_2" name="Hour" class="custom-control-
input" required>
 <label class="custom-control-label" style="font-size: 0.9rem" for="Hour_2">2 часа</label>
 </div>
</div>

<div class=" justify-content-around d-flex flex-row mt-2">
 <div class=" justify-content-center row mx-3 my-3">
 <button type="submit" class="btn btn-dark row justify-content-center mx-auto fs-6"
style="width: 200px;">Экспорт</button>
 </div>
</div>
</div>
</form>
{% endblock %}

```

**#Выпускная квалификационная работа студентки Палагиной Софьи Алексеевны**

**#Группа А-12-17**

**#Листинг модуля "SearchResult.html" приложения "MainApp"**

```
{%extends 'MainApp/base3.html'%}
{%block title%}
Главная страница
{% endblock %}

{%block content%}
{% if error == 'e' %}
<div class="container my-3">
<div class="d-flex flex-column justify-content-center w-100">
<div class="d-flex justify-content-center fs-5 mb-3 text-center">
 Информация по станции {% for el in res %} {{el.NameStation}} {% endfor %}
</div>

<div class="table-responsive mx-3">
 <table class="table table-bordered text-center table-sm " >
 <thead style="font-style: italic; background-color: #f8f9fa">
 <tr style="font-size: 0.8rem">
 <th>Номер станции</th>
 <th>Название станции</th>
 <th>Регион</th>
 <th>Широта</th>
 <th>Долгота</th>
 </tr>
 </thead>
 {% for el in res %}
 <tbody style="font-size: 0.8rem">
 <tr>
 <td>{{ el.id }}</td>
```

```

<td>{{el.NameStation}}</td>
<td>{{el.Region}}</td>
<td>{{el.Latitude}}</td>
<td>{{el.Longitude}}</td>
</tr>

</tbody>
{% endfor %}
</table>
</div>
<div class="container">
<div class="d-flex flex-row justify-content-end ">
<p class="mb-2 align-content-end text-muted">
 <button class="btn page-link" style="font-size: 0.8rem" type="button" data-toggle="collapse" data-
target="#collapseExample" aria-expanded="false" aria-controls="collapseExample">
 Экспорт csv
 </button>
</p>
<div class="collapse container" id="collapseExample" >
 <div class="card card-body " >
 <div class="d-flex flex-column justify-content-around w-100" >

 <form action="{% url 'Export_csv' %}" method="get" >
 <button type="submit" class="dropdown-item text-muted text-wrap" style="font-size: 0.8rem"
name="export" value="Albedo">Альбедо</button>
 <button type="submit" class="dropdown-item text-muted text-wrap" style="font-size: 0.8rem"
name="export" value="Sum">Среднемесячные часовые суммы суммарной солнечной
радиации</button>
 <button type="submit" class="dropdown-item text-muted text-wrap" style="font-size: 0.8rem"
name="export" value="Straight">Среднемесячные часовые суммы прямой солнечной
радиации</button>
 <button type="submit" class="dropdown-item text-muted text-wrap" style="font-size: 0.8rem"
name="export" value="Diff">Среднемесячные часовые суммы диффузной солнечной
радиации</button>

```

```
 Интерполировать данные
```

```
 </form>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<div class="container">
```

```
 <hr>
```

```
</div>
```

```
<div class="container">
```

```
<ul class="nav justify-content-center mx-3 ">
```

```
 <div class="d-flex flex-wrap ">
```

```
 <li class="nav-item">
```

```
 Альбедо
```

```

```

```
 <li class="nav-item">
```

```
 Месячные суммы
```

```

```

```
 <li class="nav-item">
```

```
 Суточные суммы
```

```

```

```
 <li class="nav-item">
```

```
 Часовые суммы
```

```

```

```
</div>
```

```

```

```
</div>
```

```
<div class="container">
```

```
 <hr>
```

```
</div>
```

```
<div class="mb-5 mt-3 rounded px-3 mx-3 " style="border: solid 1px #DDDDDD" id="alb">
```

```
<div class="text-center py-sm-4 fw-normal">
```

```
Среднемесячные коэффициенты отражательной способности поверхности -
Альбедо
```

```
</div>
```

```
<div class="table-responsive ">
```

```
<table class="table table-bordered text-center table-sm " >
```

```
<thead style="font-size: 0.7rem; font-style: italic; background-color: #f8f9fa">
```

```
<tr style="font-size: 0.7rem">
```

```
<td>Янв.</td>
```

```
<td>Фев.</td>
```

```
<td>Март</td>
```

```
<td>Апр.</td>
```

```
<td>Май</td>
```

```
<td>Июнь</td>
```

```
<td>Июль</td>
```

```
<td>Авг.</td>
```

```
<td>Сен.</td>
```

```
<td>Окт.</td>
```

```
<td>Нояб.</td>
```

```
<td>Дек.</td>
```

```
</tr>
```

```
</thead>
```

```
{% for el in albedo %}
```

```
<tbody style="font-size: 0.7rem">
```

```
<tr>
```

```
<td>{{el.Month_1}}</td>
```

```
<td>{{el.Month_2}}</td>
```



```

<td>{{el.Month_3}}</td>
<td>{{el.Month_4}}</td>
<td>{{el.Month_5}}</td>
<td>{{el.Month_6}}</td>
<td>{{el.Month_7}}</td>
<td>{{el.Month_8}}</td>
<td>{{el.Month_9}}</td>
<td>{{el.Month_10}}</td>
<td>{{el.Month_11}}</td>
<td>{{el.Month_12}}</td>
</tr>

```

```

</tbody>

```

```

{% endfor %}

```

```

</table>

```

```

</div>

```

```

</div>

```

#Дальнейшее отображение данных аналогично для всех таблиц

```

{% endblock %}

```

**#Выпускная квалификационная работа студентки Палагиной Софьи Алексеевны**

**#Группа А-12-17**

**#Листинг модуля "FindName.html" приложения "MainApp". Модули "FindNumberhtml" и "FindCoord.html" аналогичны.**

```

{%extends 'MainApp/base3.html'%}

```

```

{%block title%}

```

Поиск по названию

```

{% endblock %}

```

```

{%block content%}

```

```

<form class = "form-control my-3" action="{% url 'SearchResult' %}" method="get">

```

```

 {% csrf_token %}

```

```

<div class="container align-items-center justify-content-center d-flex flex-column w-100">

 <span class="container shadow-sm p-2 mb-3 bg-light rounded justify-content-center lead row
text-center fs-6 " >

 Введите название интересующей станции

 <div class=" justify-content-center flex-column row w-auto fs-6">

 <input type="text" class=" input-group mw-auto my-sm-1 justify-content-center mx-auto text-
center fs-6" placeholder="Введите название" name="search" required>

 </div>

 <div class=" justify-content-center flex-column row w-auto ">

 </div>

 <div class=" justify-content-center d-flex">

 <div class=" justify-content-center flex-column row my-5">

 <button type="submit" class="btn btn-dark row justify-content-center mx-auto fs-6"
style="width: 100px;">Поиск</button>

 </div>

 </div>

</div>

</div>
{% endblock %}

```

***Приложение Г. Схема алгоритма интерполирования данных***

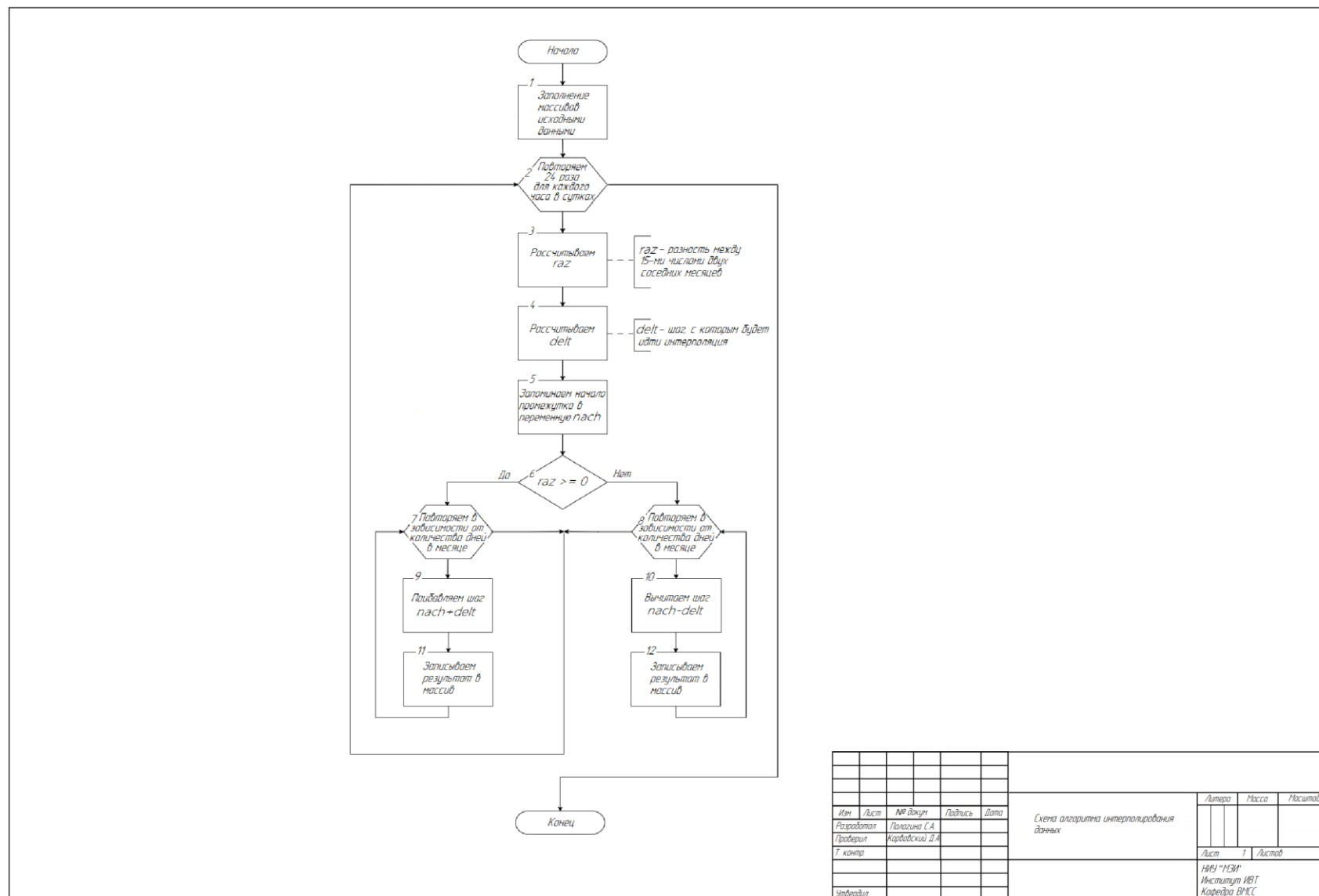


Рис.41. Схема алгоритма интерполирования данных

***Приложение Д. Схема алгоритма экспорта данных в формате csv***

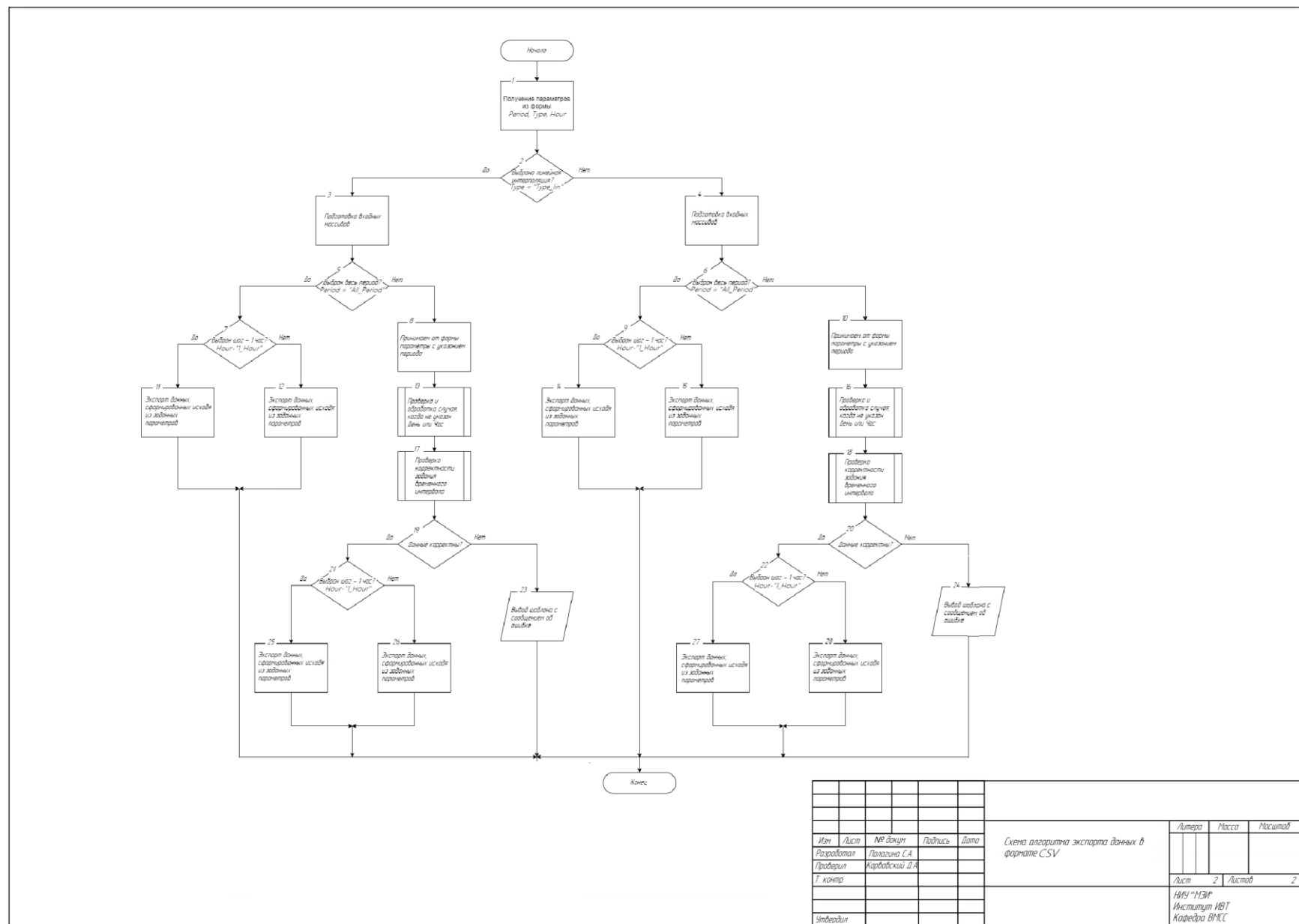


Рис.42. Схема алгоритма экспорта данных в формате csv