УДК 004

Разработка геоинформационной системы, предоставляющей метеорологические параметры, связанные с засухой и пожарной опасностью

Вихляев Дмитрий Романович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье рассматривается создание веб-ориентированной геоинформационной системы занимающиеся расчётами индексов засухи и представлением показателей, связанных с засушливостью и пожарной опасностью. Система включает два приложения написанных на Python, C# и JavaScript. Хранилищами данных выступают база данных MySql и файловая система с векторными данными. Результатом исследования станет подробное описание создания описываемой системы, и представление документации для пользователя системы.
Ключевые слова: ГИС, Python, CSV, pandas, C#, ASP.NET, JavaScript, Leaflet.

Development of a geoinformation system providing meteorological parameters related to drought and fire hazards

Vikhlyaev Dmitry Romanovich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article discusses the creation of a web-based geoinformation system dealing with the calculation of drought indices and the presentation of indicators related to aridity and fire danger. The system includes two applications written in Python, C# and JavaScript. The data storages are a MySQL database and a file system with vector data. The result of the study will be a detailed description of the creation of the described system, and the presentation of documentation for the user of the system.
Keywords: GIS, Python, CSV, pandas, C#, ASP.NET, JavaScript, Leaflet.

1 Введение

1.1 Актуальность

В наши дни информационные технологии играют ключевую роль в разных направлениях, одними из которых являются метеорология и управление рисками. Любые метеорологические данные считаются ценной

информацией при прогнозировании погоды и оценки рисков, связанных с природными явлениями.

История использования информационных систем с метеорологическими данными насчитывает десятилетия развития и совершенствования. С появлением компьютеров и сетей передачи данных стало возможным собирать, хранить и анализировать большие объемы метеорологической информации. С развитием технологий обработки данных, искусственного интеллекта и облачных вычислений информационные системы стали более эффективными и точными в предоставлении прогнозов и анализе метеорологических данных.

Особое внимание уделяется индексу засухи, который относится к важным показателям для оценки текущих погодных условий. Этот индекс основан на анализе таких параметров, как количество осадков, значение температуры воздуха и влажность почвы. Процесс расчета индекса засухи позволяют автоматизировать информационные системы и оперативно предоставлять информацию о его состоянии.

Также важно прогнозирование возгораний, особенно в регионах с высоким риском лесных пожаров. Метеорологические данные, включая скорость ветра, температуру и влажность, играют важную роль в определении степени пожарной опасности. Применение информационных систем помогает анализировать и обрабатывать эти данные, а также создавать модели, прогнозирующие возможные пожары, на основе которых можно принимать соответствующие меры по их предотвращению и тушению.

1.2Обзор исследований

А.В.Аргучинцева, Л.В.Голубева исследовали различные индексы пожароопасности [1]. Е.В.Болданова в своей статье указала точку росы как основу оперативного показателя лесопожарной опасности [2]. Е.В.Воропаева, Е.П.Кунаева использовали ГИС и веб-ГИС технологии в разработке собственного электронного ресурса [3]. В.В.Журавель, П.А.Осипов, Я.С.Осипова, М.О.Димухаметов, Д.А.Осипова в своей работе описали использование различных Web-технологий и ГИС на примере геопорталов и web-ГИС-серверов [4]. Р.М.Илякова, С.А.Долгих провели анализ повторяемости почвенной засухи на основе индекса Палмера [5]. Н.К.Кадыркулова, И.И.Абдухалимов представил результат исследования проектирования баз данных в среде ГИС-технологий [6].

1.3Цель исследования

Цель исследования – разработка геоинформационной системы, предоставляющей метеорологические параметры, связанные с засухой и пожарной опасностью.

2 Материалы и методы

В этапы реализации включено проектирование логической и физической модели базы данных, а также функциональных задач информационной системы. В Разработке использовались языки программирования Python, C# и JavaScript, интерфейс построен на фреймворке ASP.NET и библиотеке для работы с цифровыми картами Leaflet. Расчёт индексов засухи основывался на метеорологических показателях станций Дальнего Востока. Для графического представления данных используются векторные данные лесных кварталов.

3 Результаты и обсуждения

В среднем ежегодно в регионе Дальнего Востока возникает до 2.5 тыс. лесных пожаров, а пройденная огнем площадь составляет более 1 млн. га.

Лесные пожары наносят огромный ущерб экосистемам, экономике и человеческим жизням. Частота и интенсивность засух напрямую влияет на пожарную опасность, так как сухие условия создают идеальную среду для распространения лесных пожаров. Вовремя рассчитанные метеорологические индексы засухи помогают лучше прогнозировать и предотвращать пожары, что снижает потенциальные потери.

Набор используемых индексов можно выбрать исходя из доступных метеоданных, территориальной пригодности, времени и простате расчёта. Индексы должны отличаться по способу расчёта, применению и периоду расчёта.

Ежемесячные индексы засухи более эффективны для долгосрочного планирования и управления природными ресурсами. Помогают оценивать и прогнозировать долгосрочные климатические тенденции и их влияние на окружающую среду.

Ежедневный индекс подходят для оперативного мониторинга и реагирования на пожары.

Самым часто используемым индексом засухи в России является гидротермический коэффициент увлажнения (ГТК) или индекс Селянинова.

Применяется в сельском хозяйстве для оценки влажности и условий вегетации растений.

Вычисляется как отношение сумм осадков R (мм), к сумме активных температур T (ºC) уменьшенное в 10 раз, за идентичные периоды времени, где T ≥ 10 ºC.

$$\text{ГТК} = \frac{\sum_{i=0}^{n} Pi}{0.1 * \sum_{i=0}^{n} Ti}$$

Г.Т. Селянинов выделил следующие климатические зоны:

- избыточного увлажнения (ГТК > 1.3);
- обеспеченного увлажнения (1.0 <=ГТК <1.3);

– засушливая (0.7 <=ГТК < 1.0);
– сухого земледелия (0.5 <=ГТК < 0.7);
– ирригации (ГТК < 0.5).

Чем выше показатель, тем больше уровня влагообеспеченности территории. На территории России ГТК чаще показывает более высокие результаты при сравнительном анализе с другими индексами.

Индекс аридности де Мартонна используется для классификации климата различных регионов. Представляет собой отношение среднемесячной суммы осадков к среднемесячной температуре воздуха, где P – количество осадков за месяц в миллиметрах, T – средняя температура за месяц в градусах Цельсия.

$$Im = \frac{12 * \sum_{i=0}^{n} Pi}{T + 10}$$

В таблице 1 Индекс де Мартонна классифицирует 7 климатических условий (табл. 1).

Таблица 1. Классификация климатических условий по индексу де Мартонна

Значения Im	Климат
Im<10	засушливый
10<=Im<20	полузасушливый
20<=Im<24	субтропический
24<=Im<28	умеренный
28<=Im<35	влажный
35<=Im<=55	очень влажный
Im>55	чрезвычайно влажный

Индекс отражает условия, которые влияют на испарение влаги. Наименьшие значения индекса соответствуют наибольшей аридности.

Количественным отражением вероятности пожаров в природе, официально принятым в России, является комплексный показатель пожарной опасности (КПО). КПО представляет собой кумулятивную сумму произведения температуры воздуха на разность температур воздуха и точки росы, вычисляемую для отдельного пункта и конкретного времени.

$$КПО_n = КПО_{n-1} * K + [t(t - t_d)]_n,$$

где, КПО(N) – значение КПО, рассчитываемое на текущий день в °С;

КПО(N-1) – значение КПО, рассчитываемое на предыдущий день КПО(N-1) = t (t-td);

К – коэффициент поправки на осадки (равен единице, если количество осадков менее 3 мм, равен нулю, если количество осадков больше или равно 3 мм);

t – температура воздуха максимальная за день в °С;

td – точка росы максимальная за день в °С;

(t – td) – дефицит точки росы в °С.

За факт выпадения осадков принимаются любые значения, начиная с 3 мм за 24 ч, осадки меньше 3 мм не учитываются.

В 1941 году была разработана «шкала Нестерова». Шкала насчитывает, подразумевает пять классов пожарной опасности, каждый из которых зависит от значения индекса Нестерова (табл. 2).

Таблица 2. Шкала Нестерова

Класс пожарной опасности	Значения индекса (КПО)	Степень пожарной опасности
I	0 – 300	Очень малая
II	301 – 1000	Малая
III	1001 – 4000	Средняя
IV	4001 – 12000	Высокая
V	>12000	Чрезвычайная

Показатель Нестерова отражает баланс иссушающих и увлажняющих факторов. Наибольшее значение индекса соответствуют наибольшей вероятности пожарной опасности.

Все метеорологические индексы вычисляются по данным метеостанций, находящихся в определённых точках. Чтобы определить всю окружающую площадь метеостанции можно воспользоваться интерполяцией методом ближайшего соседа. Метод ближайшего соседа использует значение ближайшего соседа в качестве оценки для точек, которые нужно интерполировать. Для каждой точки (xi,yi) в наборе точек вычисляется евклидово расстояние до целевой точки (x,y).

$$k = \arg \min_{i \in \{1,2,\ldots,n\}} \sqrt{((x_i - x)^2 + ((y_i - y)^2},$$

где, k – индекс ближайшей точки;

n – количество точек.

В качестве исходных данных выступают набор точек и целевая точка, для которой нужно найти ближайшую точку в наборе. Для каждой точки в наборе рассчитывается евклидово расстояние до целевой точки. Из всех рассчитанных расстояний выбирается минимальное, что определяет ближайшую точку.

В разрабатываемой программе вместо точек используются центральные точки полигонов, представляющих территорию лесных кварталов. В каждой отдельно взятой области на территории Дальнего Востока, имеются охраняемые лесные участки. Данные метеостанций берутся от выбранной области. Дополнительно к ним определяются станции из соседних районов. Для каждой станции внутри области рассчитываются ближайшее от неё допустимое расстояние по четырём сторонах в шаге два градуса земных координат, в пределах которых находятся дополнительные

станции, в конце оттесняются все повторы. Таким образом, охватывается всё пространство вокруг.

Для начала необходимо определить, какие физические и программные компоненты входят в систему и как они между собой взаимодействуют. Графическая модель компонентов системы представлена на рисунке 1.

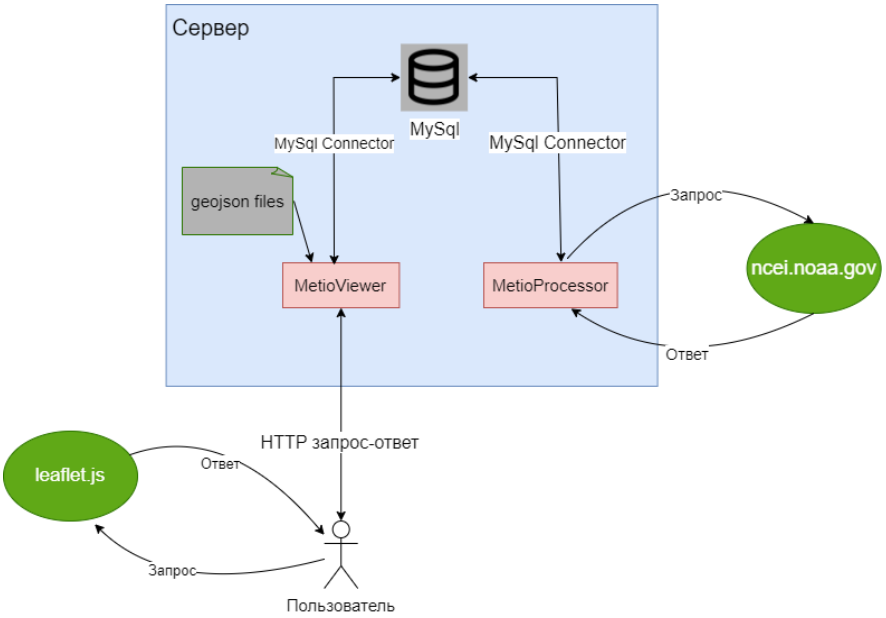

Рис. 1. Диаграмма компонентов системы и их взаимодействия

На стороне сервера расположены хранилища данных и две главные программы MetioViewer и MetioProcessor.

Программа MetioProcessor занимается обработкой метеорологических данных полученных от сервера ncei.noaa.gov. Данные в формате CSV проходят фильтрацию для выделения необходимых показателей, после чего происходит расчёт требуемых индексов засухи и их загрузка в базу данных. Предполагается, что с определённой периодичностью на сервере запускается MetioProcessor, для добавления новых значений в базу данных. Программа реализует sql запрос для получения списка метеостанций. Для каждой станции запрашиваются данные на текущий год. Затем вновь происходит запрос к базе данных для получения последней даты. В случае если в загруженных данных есть более поздний период, чем указан в базе данных, происходит процесс расчёта и заполнения новых значений.

Приложение имеет консольный интерфейс. Набор команд позволяет администраторам системы изменять, добавлять и удалять записи в базе данных, а также вручную запускать процесс получения и расчёта метеорологических показателей, как с локального, так и с удалённого сервера. Каждое действие заносится в файл легирования, что также позволяет просматривать и находить ошибки работы системы.

MetioViewer является основной программой, с которой взаимодействует пользователь. Главная задача приложения состоит в том, чтобы отображать цифровую карту и визуализировать расчётные данные.

Все показатели, хранящиеся в базе данных, интерфейс позволяет представить в табличной форме по конкретной станции и дате.

Приложение предоставляет функцию расчёта интерполяции каждого индекса засухи для каждого лесного квартала, на основе окружающих данных метеостанций.

За визуализацию карты и всех объектов на ней, отвечает библиотека leaflet. В исходный код библиотеки вложена функция запроса мозаичных изображений карты с сервера. При начальном отображении карты, зумировании или перемещении происходит запрос на сервер для получения других частей карты. В список объектов отображаемых на карте входят метки метеостанций и окно подсказка содержащее описание станции. Данные о координатах и описание берутся из соответствующей таблицы в базе данных. Другим типом объектов являются лесоохранные участки, представляемые как полигоны. На сервере хранятся в виде файлов в формате geojson. Помимо отображения, leaflet имеет методы изменения стиля отображаемых объектов. Эта особенность применяется для окрашивания лесных кварталов в цвет характерный определённому значению запрашиваемого индекса.

Обычные пользователи могут взаимодействовать с системой, только через веб-интерфейс MetioViewer. Таким образом, гость может просматривать визуальные данные. Привилегированный пользователь, например, администратор системы, может быть, как гостем и только получать данные, и в тоже время способен создавать новые записи в базе данных через интерфейс команд MetioProcessor. В данной системе под разделением прав доступа, подразумевают разделение программ. Для гостя доступ осуществляется по протоколу HTTP, а у привилегированного пользователя должен быть другой способ взаимодействия с внутри серверной программой, например, через протокол SSH. Визуальное описание доступных функция для разных групп пользователей представлено на рисунке 2.

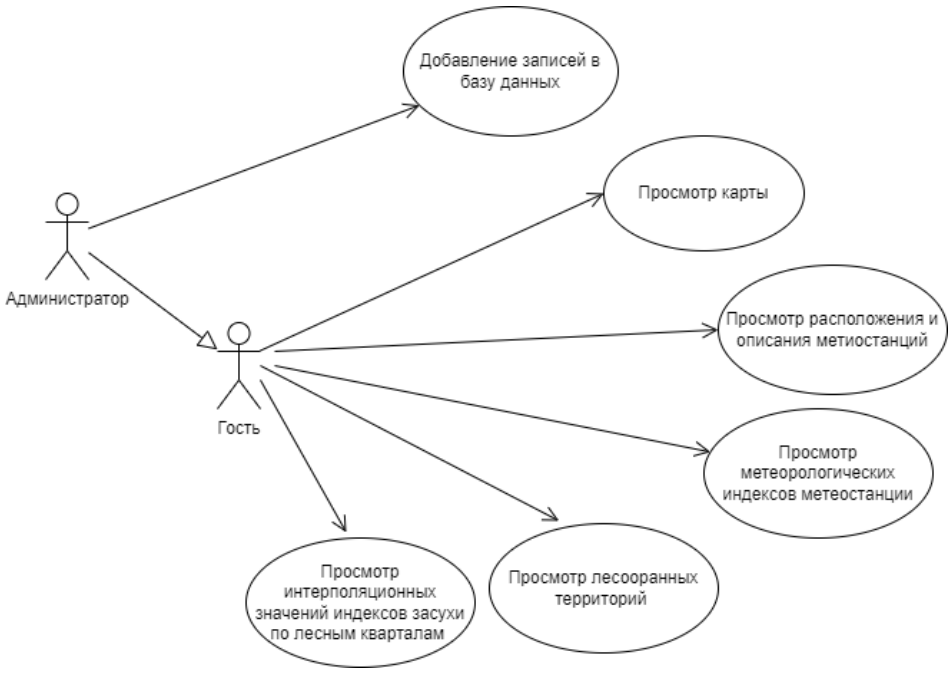

Рис. 2. Диаграмма вариантов использования

Следующей частью разработки системы является проектирование классов. Построение диаграммы классов позволит определить сущности системы и взаимосвязь объектов. Первым делом необходимо определить главные сущности хранения данных в приложении MetioProcessor. Главными объектами системы являются метеостанции и их зафиксированные и расчётные показатели, разделённые на ежедневные и ежемесячные индексы. Помимо этого, есть необходимость разделить все имеющиеся станции по территориальной принадлежности регионов. Каждый из перечисленных объектов должен ассоциироваться с таблицей базы данных, что значит иметь при себе методы взаимодействия с СУБД. Для предотвращения дублирования кода, нужно добавить общий класс, взаимодействующий с базой данных и имеющий доступ к своим методам из перечисленных объектов. По всем вышеописанным требованиям была создана следующая диаграмма классов (рис.3).

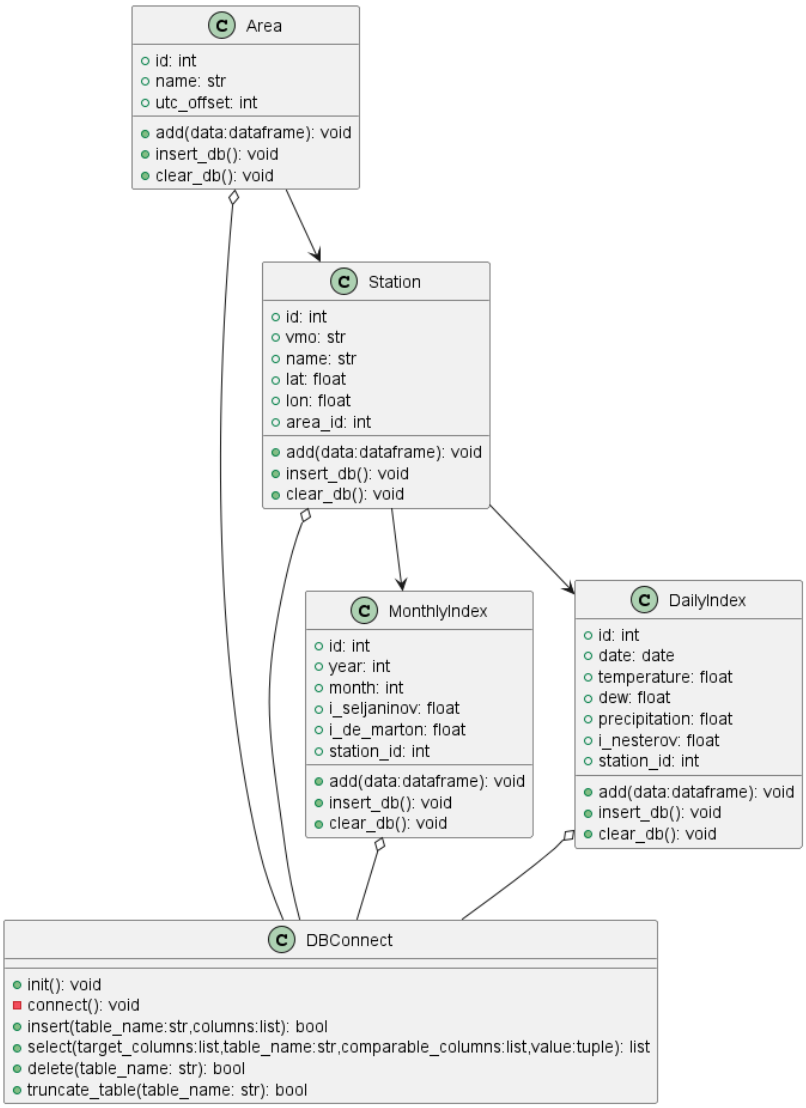

Рис. 3. Диаграмма классов приложения MetioProcessor

В данной программе используются несколько классов для работы с метеорологическими данными: Area, Station, MonthlyIndex, DailyIndex и DBConnect. Каждый класс имеет свои атрибуты и методы, а также определенные взаимосвязи с другими классами.

Методы класса включают собственные, такие как add (data: dataframe), который добавляет данные в экземпляр класса. Также являющиеся оболочкой для вызова общего метода класса DBConnect, такими являются insert_db() и clear_db(), которое соответственно добавляет и очищает данные из одноимённой таблицы.

Класс Area предназначен для представления региона и содержит следующие атрибуты: id (уникальный идентификатор региона), name (название региона) и utc_offset (смещение времени относительно UTC).

Класс Station представляет метеорологическую станцию и имеет следующие атрибуты: id (уникальный идентификатор станции), wmo (код WMO), name (название станции), lat (широта), lon (долгота), area_id (идентификатор региона, к которому относится станция).

Класс MonthlyIndex используется для представления месячных индексов и имеет следующие атрибуты: id (уникальный идентификатор), year (год), month (месяц), i_seljaninov (индекс Селянинова), i_de_marton (индекс де Мартонна), station_id (идентификатор станции, к которой относится индекс).

Класс DailyIndex предназначен для представления ежедневных индексов и имеет следующие атрибуты: id (уникальный идентификатор), date (дата), temperature (температура), dew (точка расы), precipitation (количество осадков), i_nesterov (индекс Нестерова), station_id (идентификатор станции, к которой относится индекс).

Класс DBConnect используется для подключения и взаимодействия с базой данных. Методы класса включают init(), главный метод инициализации, внутри которого вызывается не публичный метод connect(), который устанавливает соединение с базой данных.

Другие методы реализуют удобную обёртку над вызовом sql запросов в стиле объектно-ориентированного программирования.

Имена методов обозначают их действия, например insert() осуществляет запрос на добавление новых записей в базу данных. Также функция select() делает запрос по нужным полям для получения записей. Применяемые в основном с целью тестирования приложения, методы очищения таблиц delete() и truncate_table() различаются тем, что второй сбрасывает значения первичного ключа.

Взаимосвязи между классами строятся следующим образом: каждый экземпляр класса Station связан с экземпляром класса Area через атрибут area_id. Экземпляры классов MonthlyIndex и DailyIndex связаны с экземплярами класса Station через атрибут station_id. Класс DBConnect используется всеми остальными классами для взаимодействия с базой данных, предоставляя методы для вставки, выбора, удаления и очистки данных.

Таким образом, эти классы и методы организованы для упрощения работы с метеорологическими данными, позволяя эффективно добавлять, обновлять и очищать данные в базе данных, а также организовывать и связывать данные между различными уровнями.

Работы программы можно описать при помощи диаграммы последовательностей, показывающей взаимодействие функций. На рисунке 4 представлена диаграмма, описывающая работу основных функций приложения MetioProcessor (рис.4).

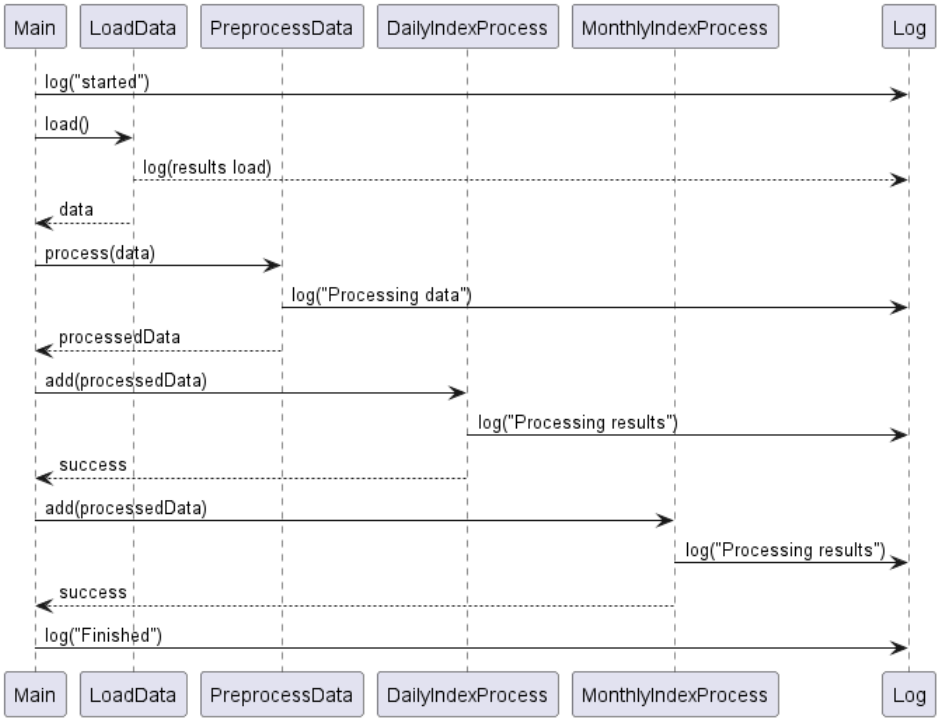

Рис. 4. Диаграмма последовательности главных функций приложения MetioProcessor

Взаимодействие происходит между несколькими компонентами: Main, LoadData, PreprocessData, DailyIndexProcess, MonthlyIndexProcess и Log.

Процесс начинается с того, что компонент Main вызывает метод log("started") у компонента Log, чтобы зафиксировать начало процесса в файле логирования. Затем Main вызывает метод load() у компонента LoadData для загрузки данных с удалённого сервера. При успешной загрузке, данные возвращаются главной функции.

Далее происходит процесс предварительной обработки данных. На этом этапе обрезаются данные до нужных показателей. На каждую строчку записей и по всем индексам реализуется процесс раскодирования данных. Поля даты и переводятся в местный для каждой станции часовой пояс. Другие показатели содержаться в целочисленных форматах, и хранят вместе с собой дополнительную информацию, которую нужно отсечь, а сами значения нужно пересчитать в требуемые единицы измерения.

Обработанные данные передаются функциям DailyIndexProcess и MonthlyIndexProcess, которые рассчитывают соответствующие индексы и заносят их в базу данных.

Каждая функция заносит свои результаты работ в файл логирования, что позволяет следить за работой и отслеживать ошибки, вызванные в программе. Детальный разбор обработки ежедневных индексов описан на рисунке 5.

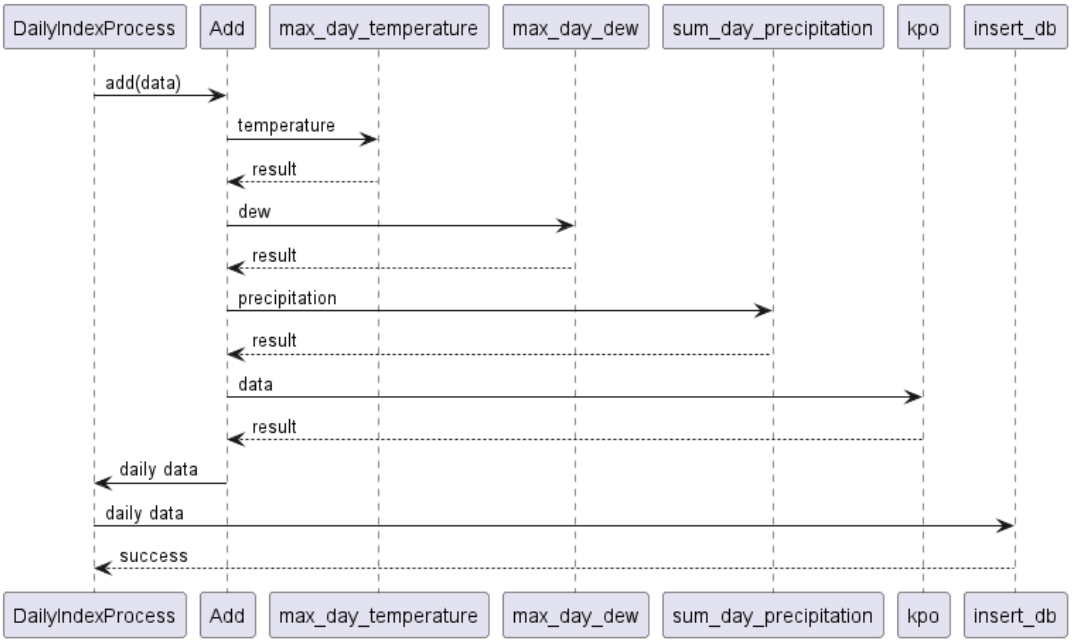

Рис. 5. Диаграмма последовательности обработки ежедневных показателей

На диаграмме последовательностей изображен процесс выполнения операции главной функции "DailyIndexProcess" с участием нескольких других компонентов для выполнения расчетов и взаимодействия с базой данных.

Для получения нужных показателей производятся расчёты максимальных значений температуру и точки росы за сутки. Затем на каждый день рассчитываются суммарное значение осадков. И уже в конце на основе полученных данных вычисляется индекс Нестерова.

Результаты возвращаются в виде одной табличной структуры, после чего записываются в базу данных

Аналогичный процесс происходит при обработке ежемесячных индексов, описанных на рисунке 6.

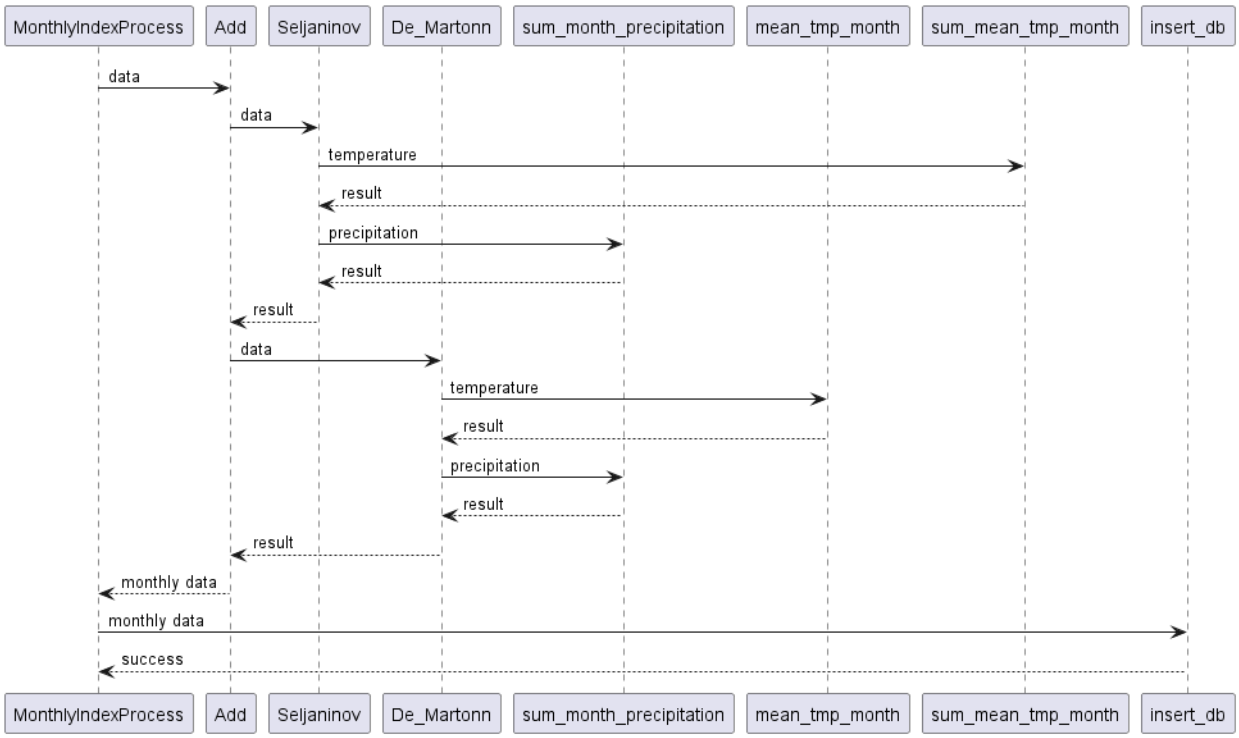

Рис. 6. Диаграмма последовательности обработки ежемесячных показателей

В отличие от предыдущего процесса на этой диаграмме видны более глубокие вызовы функций. Также имеется главная функция MonthlyIndexProcess, которая вызывает метод добавления всех нужных показателей.

Для получения индекса Селянинова необходимо расчитать сумму среднемесячных температур и сумму месячных осадков. Индекс де Мартонна, также вызывает функцию расчёта суммы месячных осадков, но для температуры ему необходима средняя температура за весь месяц. После расчёта данные записываются в базу данных.

Описание классов в программе MetioViewer схоже с предыдущим приложением, но в ней добавляется компонент Forest. Новая структура не относится к базе данных, она описывает содержание полигонов лесных участков, полученных из файлов, находящихся на сервере в формате geojson (рис. 7).

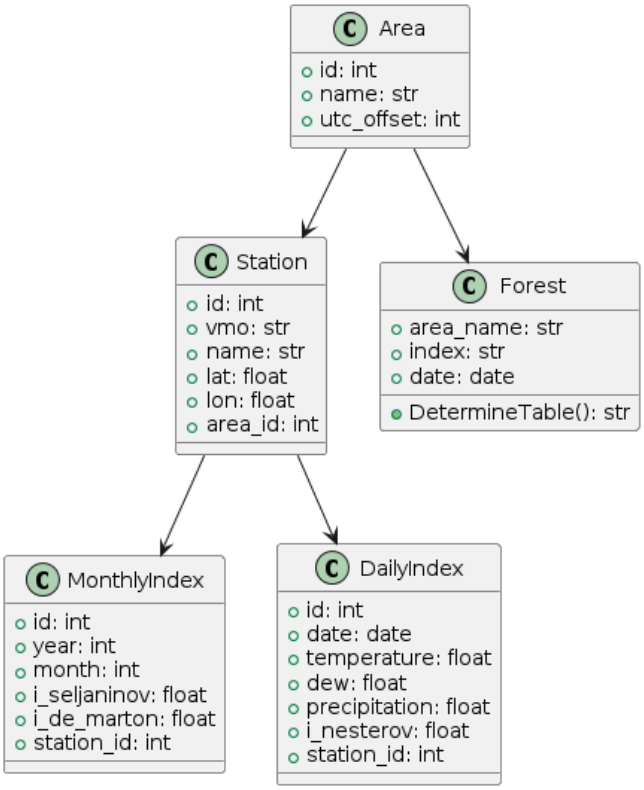

Рис. 7. Диаграмма классов приложения MetioViewer

Внутренне содержимое уже описанных для предыдущего приложения классов не изменяется. В классе Forest, содержится идентификатор области, для которой будут загружаться полигоны. Поле индекс предназначено для хранения строкового названия индекса, для которого производится расчёт интерполяции всех полигонов. В поле данных содержаться сами данные полигонов их координаты и атрибуты. Также имеется метод, определяющий в какой таблице искать нужный индекс.

Обращение к базе данных происходит через дополнительные классы контроллеры (рис. 8).

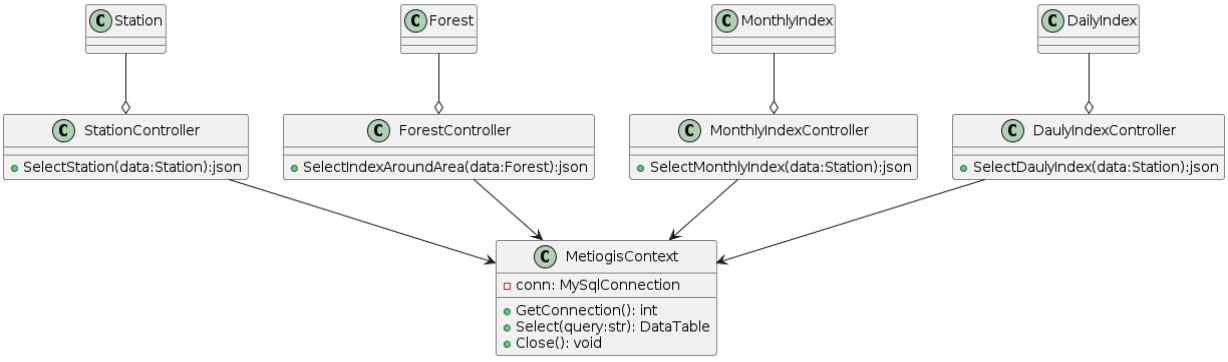

Рис. 8. Диаграмма классов, описывающая сущности для чтения базы данных приложения MetioViewer

Контроллеры используют классы модели для обращения к соответствующим таблицам базы данных. Между этими классами и СУБД

стоит посредник, в котором прописан низкоуровневые обращения, основанные на sql-запросах.

Данные классы описывают лишь серверную часть программы, но на стороне клиента, также реализованы классы и структуры. Всё взаимодействие выполняется вокруг библиотеки Leaflet. Объект L в Leaflet служит основным пространством имен и точкой входа для взаимодействия с библиотекой. Он предоставляет набор методов и классов, которые позволяют разработчику создавать и управлять картами, слоями, маркерами и другими элементами. Диаграмма классов на рисунке 9 представляет взаимодействие объектов библиотеки Leaflet и пользовательских структур, таких как станции и леса. Она демонстрирует, как элементы карты и слои объединены и настроены для отображения географических данных.

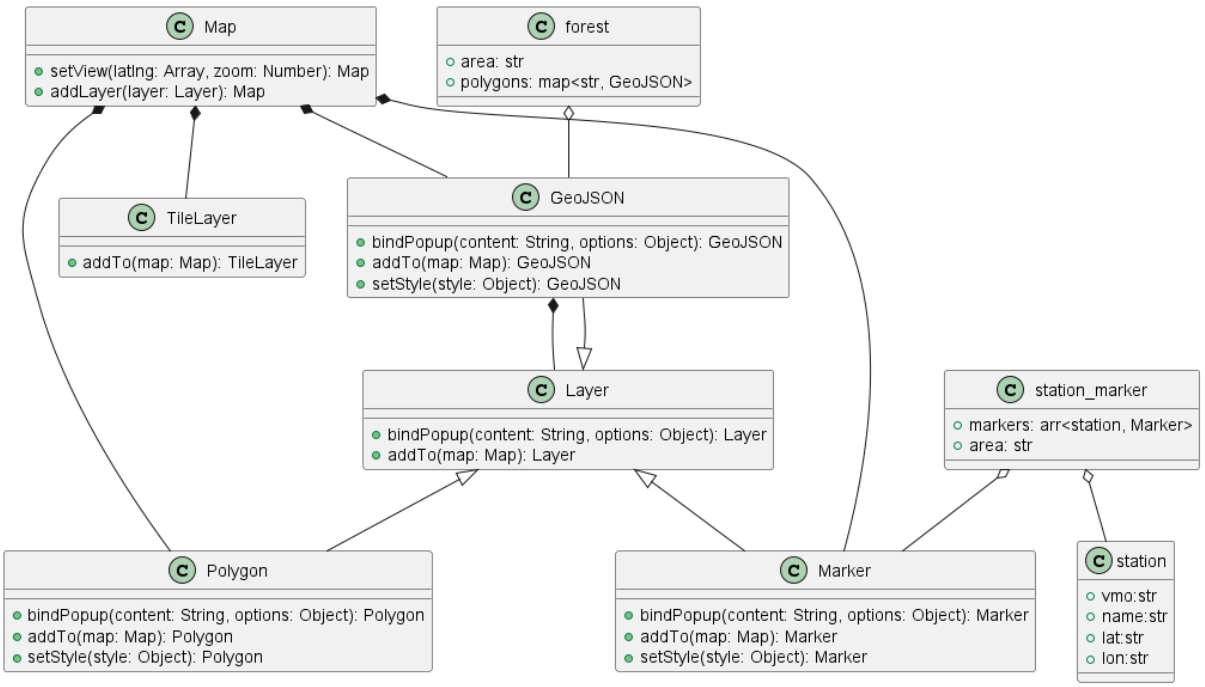

Рис. 9. Диаграмма классов, описывающая взаимодействие объектов Leaflet и пользовательские структуры

Основным классом является Map, который представляет карту и содержит методы для управления картой и добавления слоев. Метод setView устанавливает центр и масштаб карты, а метод addLayer добавляет слой на карту. На карте можно размещать слои различных типов, таких как TileLayer, Marker, Polygon и GeoJSON.

Класс TileLayer представляет слой и имеет метод addTo, который добавляет слой на карту. Это позволяет загружать и отображать карты, состоящие из множества маленьких изображений (тайлов).

Классы Marker, Polygon и GeoJSON представляют различные геометрические объекты на карте. У всех этих классов есть методы bindPopup для привязки всплывающих окон, addTo, для добавления объектов на карту и setStyle для задания стиля объектов. Класс GeoJSON используется

для отображения данных в формате GeoJSON, что позволяет легко интегрировать сложные географические данные.

Класс Layer является абстрактным и наследуется классами Marker, Polygon и GeoJSON. Это позволяет использовать общие методы, такие как bindPopup и addTo, для всех типов слоев.

Пользовательские объекты включают station, station_marker и forest. Класс station представляет станцию с атрибутами vmo (идентификатор станции), name (название станции), lat (широта) и lon (долгота). Класс station_marker представляет маркер станции и содержит массив markers, который состоит из объектов station и Marker, а также строковое поле area, указывающее на область, к которой принадлежит маркер.

Класс forest представляет лесные участки и содержит строковое поле area, а также карту polygons, где ключом является строка, а значением – объект GeoJSON. Это позволяет отображать лесные участки в виде полигонов на карте.

Взаимодействие объектов показано на диаграмме последовательностей функций представленной на рисунке 10.

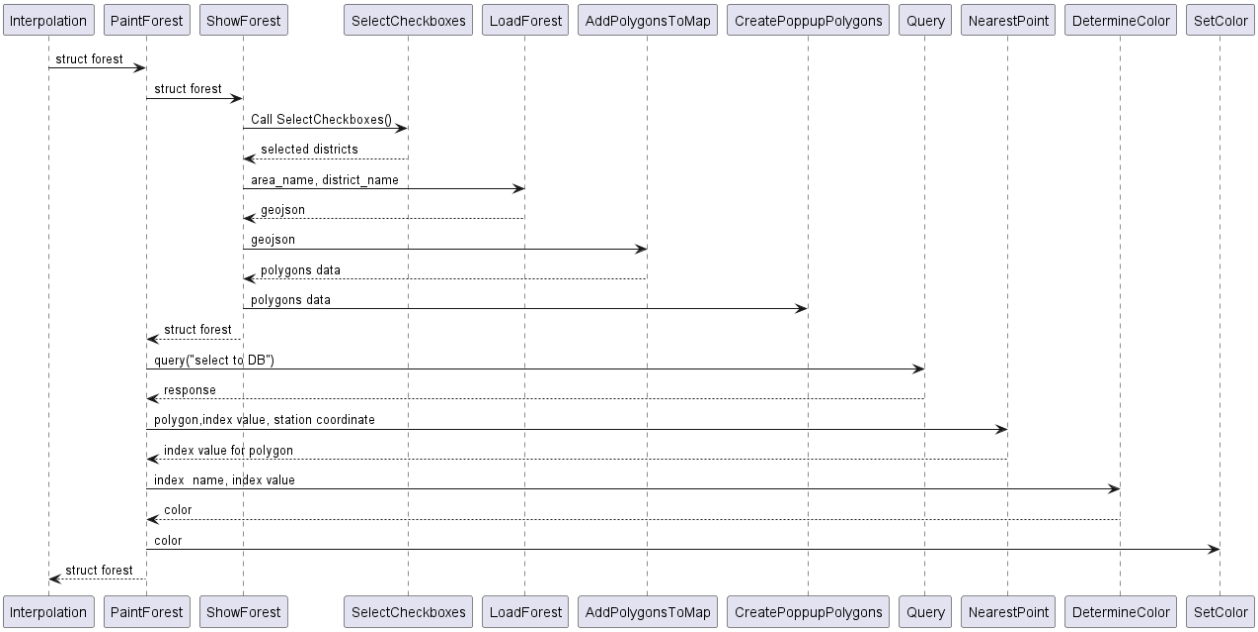

Рис. 10. Диаграмма последовательности функций, интерполяции показателей засухи по лесным кварталам

Функция PaintForest инициирует процесс отображения лесных полигонов на карте. Она вызывает функцию ShowForest, которая отвечает за выбор областей или районов, которые должны быть отображены на карте. В функции ShowForest используется объект Map для взаимодействия с картой. Затем вызывается функция SelectCheckboxes, где пользователь выбирает нужные районы. Эта функция возвращает выбранные районы, имена областей и данные полигонов в формате GeoJSON, используя объекты Map, GeoJSON и forest.

Функция LoadForest загружает данные полигонов для выбранных областей из базы данных. Она отправляет запрос и получает ответ, содержащий координаты станций, индекс значений и данные полигонов. В этой функции используются объекты Map, GeoJSON, forest и station. Затем данные полигонов обрабатываются и создаются объекты GeoJSON для отображения полигонов на карте.

Функция AddPolygonsToMap добавляет загруженные полигоны на карту. Она использует метод addTo(map: Map) для добавления полигонов на карту и присваивает индекс значений каждому полигону. В этой функции используются объекты Map, GeoJSON, forest и Polygon.

Функция CreatePopupPolygons создает всплывающие окна для полигонов. Она использует метод bindPopup для привязки всплывающих окон к полигонам. Всплывающие окна показывают данные индексов и значений, которые были получены ранее. В этой функции используются объекты Polygon, GeoJSON и forest.

Функция Query выполняет запросы к базе данных для получения необходимой информации для полигонов. Она возвращает результаты запроса, которые включают координаты станций и данные индексов. В этой функции используются объекты Map, station и forest.

Функция NearestPoint определяет ближайшую точку или станцию к каждому полигону. Она использует данные координат станций, хранящихся в объектах station и forest.

Функция DetermineColor определяет цвет для каждого полигона на основе его индекса или других критериев. Она возвращает цветовые значения для полигонов. В этой функции используются объекты Polygon, GeoJSON и forest.

Функция SetColor применяет определенные цвета к полигонам на карте. Она использует метод setStyle для задания стиля полигонов, используя данные о цвете для каждого полигона. В этой функции используются объекты Polygon, GeoJSON и forest.

Следующим этапом разработки информационной системы стало создание базы данных. В предыдущем этапе проектирования уже были определены базовые сущности системы, которые будут хранить необходимые данные. На основании построенной диаграммы классов, описывающей основные сущности системы, была построена следующая схема базы данных (рис. 11).

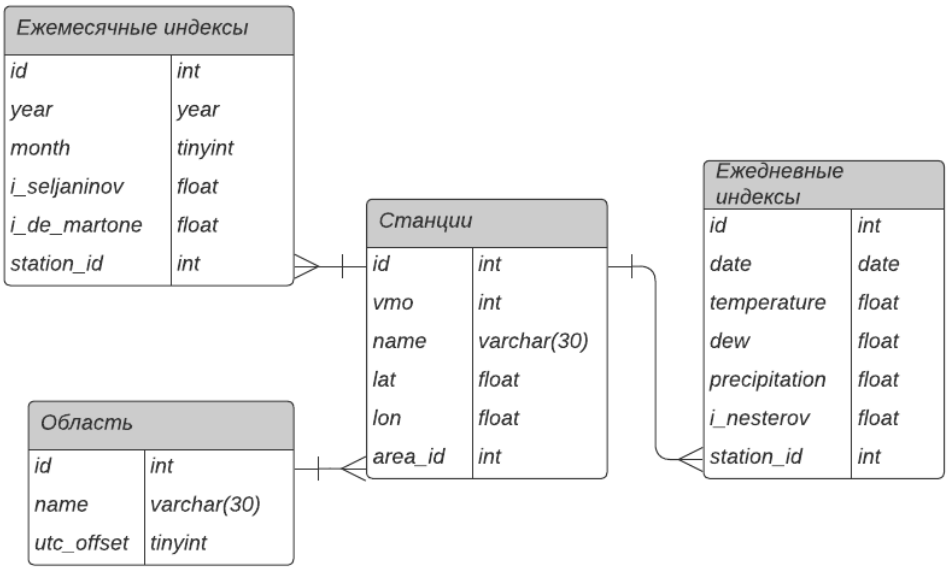

Рис. 11. Структура базы данных

В разработки информационной системы состоящей из двух программ, использовались языки программирования Python и C#.

Для приложения рассчитывающего индексы засухи применялся язык Python и библиотека pandas, разработанная для анализа данных. В качестве основной структуры данных для работы с табличными данными, отлично подходит объект pandas.DataFrame. Данные метеостанций находящиеся на сервере в формате CSV, загружаются непременно в данную структуру, и продолжают в ней находиться до выгрузки в базу данных (рис. 12, 13)

```python
#загрузка таблицы из csv в pandas dataframe
def csv_to_dataframe(filename):

    try:
        df=pd.read_csv(filename)
        return df
    except:
        print(f'Error loadDataFrame. File filename {filename} not founds\n')
        return None

#загрузка метиоданных из локального сервера
def metio_load(year=0,vmo=0,path=None):
    if(path is not None):
        return csv_to_dataframe(path)
    else:
        return csv_to_dataframe(PATH.format(year,vmo))

#загрузка метиоданных из удалённого сервера
def remote_metio_load(year,vmo):
    response = requests.get(URL.format(year,vmo))
    if response.status_code == 200:
        return csv_to_dataframe(StringIO(response.text))
    else:
        print(f"Ошибка: Невозможно получить данные. Код статуса: {response.status_code}")
        return None
```

Рис. 12. Программный код загрузки метеорологических данных станции

STATION	DATE	SOURCE	LATITUDE	LONGITUDE	ELEVATION	NAME	REPORT_TYPE	CALL_SIGN	QUALITY_CONTROL	WND
31713099999	2000-01-01T03:00:00	4	48.7333333	132.95	80.0	BIROBIDZHAN, RS	FM-12	99999	V020	999,9,C,0000,1
31713099999	2000-01-01T06:00:00	4	48.7333333	132.95	80.0	BIROBIDZHAN, RS	FM-12	99999	V020	999,9,C,0000,1
31713099999	2000-01-01T15:00:00	4	48.7333333	132.95	80.0	BIROBIDZHAN, RS	FM-12	99999	V020	020,1,N,0020,1
31713099999	2000-01-01T18:00:00	4	48.7333333	132.95	80.0	BIROBIDZHAN, RS	FM-12	99999	V020	999,9,C,0000,1
31713099999	2000-01-01T21:00:00	4	48.7333333	132.95	80.0	BIROBIDZHAN, RS	FM-12	99999	V020	999,9,C,0000,1
31713099999	2000-01-02T03:00:00	4	48.7333333	132.95	80.0	BIROBIDZHAN, RS	FM-12	99999	V020	360,1,N,0020,1
31713099999	2000-01-02T06:00:00	4	48.7333333	132.95	80.0	BIROBIDZHAN, RS	FM-12	99999	V020	050,1,N,0020,1
31713099999	2000-01-02T15:00:00	4	48.7333333	132.95	80.0	BIROBIDZHAN, RS	FM-12	99999	V020	999,9,C,0000,1
31713099999	2000-01-02T18:00:00	4	48.7333333	132.95	80.0	BIROBIDZHAN, RS	FM-12	99999	V020	999,9,C,0000,1
31713099999	2000-01-02T21:00:00	4	48.7333333	132.95	80.0	BIROBIDZHAN, RS	FM-12	99999	V020	340,1,N,0020,1
31713099999	2000-01-03T03:00:00	4	48.7333333	132.95	80.0	BIROBIDZHAN, RS	FM-12	99999	V020	090,1,N,0020,1
31713099999	2000-01-03T06:00:00	4	48.7333333	132.95	80.0	BIROBIDZHAN, RS	FM-12	99999	V020	090,1,N,0020,1
31713099999	2000-01-03T15:00:00	4	48.7333333	132.95	80.0	BIROBIDZHAN, RS	FM-12	99999	V020	999,9,C,0000,1
31713099999	2000-01-03T18:00:00	4	48.7333333	132.95	80.0	BIROBIDZHAN, RS	FM-12	99999	V020	340,1,N,0020,1
31713099999	2000-01-03T21:00:00	4	48.7333333	132.95	80.0	BIROBIDZHAN, RS	FM-12	99999	V020	340,1,N,0020,1

Рис. 13. Часть данных CSV файла с метеоданными станции

После загрузки данные проходят обработку, для каждого показателя реализована собственная функция преобразования (рис.14).

```
''' главная функия обработки данных'''
def data_translate(df,utc_offset):

    # добавление столбца информации о редактировании, 4 битовых флага
    # DATE TMP DEW AA1 от младшего к старшему соответственно
    df["EDIT"]=0

    # обработка даты
    df = date_traslation(df,utc_offset)

    # коды соответствующие ошибочным данным
    fault_code=["3","7","9"]

    for row in df.itertuples():

        # обработка температуры
        data_tmp=tmp_translatin(row.TMP)
        # если пропущенны или ошибочны
        if(data_tmp==None or data_tmp[1] in fault_code):
            df.at[row.Index,"TMP"]=None
        else:
            df.at[row.Index,"TMP"]=data_tmp[0]


        # обработка точки рассы
        data_dew=dew_translatin(row.DEW)
        # если пропущенны или ошибочны
        if(data_dew==None or data_dew[1] in fault_code):
            df.at[row.Index,"DEW"]=None
        else:
            df.at[row.Index,"DEW"]=data_dew[0]


        # обработка осадков
        data_aa1=aa1_translatin(row.AA1)
        # если пропущенны или ошибочны
        if(data_aa1==None or data_aa1[1] in fault_code):
            df.at[row.Index,"AA1"]=None
        else:
            df.at[row.Index,"AA1"]=data_aa1[0]

    return df
```

Рис. 14. Реализация основной функции предобработки данных

Особенности работы с pandas позволяют делать удобные расчёты с табличными данными, простые операции расчёта не требуют глубоких знаний работы с двумерными массивами, только простые правила описание в документации. Представленные на рисунках 15-17 фрагменты кода, являются функциями для расчёта целевых индексов. Методы используемой структуры данных во многих случаях позволяют не использовать циклы, если один dataframe схож с другим, то математические действия выполняются в одну строчку по соответствующим строкам.

```python
def kpo(df):

    kpo=pd.Series()

    for i, row in df.iterrows():

        if((row['TMP'] is None) or (row['DEW'] is None)):
            kpo[row["DATE"]] = 0
            #print("None")
            continue

        if(pd.isna(row['TMP']) or pd.isna(row['DEW'])):
            kpo[row["DATE"]] = 0
            #print("NaN")
            continue

        # Проверяем условие для K
        K = 1 if row['AA1'] < 3 else 0

        # Выполняем расчеты по формуле
        if row.name == 0:
            # Для первой строки KPO(N-1) принимаем равным 0
            KPO_N_minus_1 = 0
        else:
            # Для остальных строк берем значение KPO(N-1) из предыдущей строки
            KPO_N_minus_1 = kpo.iloc[-1]

        KPO_N = KPO_N_minus_1 * K + (row['TMP'] * (row['TMP'] - row['DEW']))  # КПО(N)

        kpo[row["DATE"]] = KPO_N


    return kpo
```

Рис. 15. Программная реализация расчёта индекса Нестерова

```python
def gtk(df):

    # средняя температура за день
    mtd=mean_tmp_day(df)

    # сумма температур за месяц
    smt=sum_tmp_month_mean_day(mtd)
    #print(smt)

    #сумма осадков в месяц
    sma=sum_month_aa1(df)
    #print(sma)

    #Вычесление индекса засухи Селянинова
    gtk=((sma.div(smt * 0.1)).astype(float)).round(2)


    #проверка условия что среднемесячная температура выше 5 градусов по Цельсию
    #================================================================

    #среднемесячная температура
    mtm=mean_tmp_month(df)
    #print(mtm)

    bad_month=mtm[mtm<5]
```

Рис. 16. Программная реализация расчёта индекса Селянинова

```python
def De_Martonna(df):

    #сумма осадков в месяц
    sma=12*sum_month_aa1(df)
    #print(sma)

    #среднемесячная температура
    mtm=mean_tmp_month(df)
    #print(mtm)

    #Вычесление индекса аридности по Де Мартонну
    iadm =(sma.div(mtm + 10)).astype(int)

    #удаление значений полученных при среднемесечной t < 5 градусов C
    bad_month=mtm[mtm<5]

    iadm=iadm.astype(object)
    iadm.loc[bad_month.keys()]=None

    return iadm
```

Рис. 17. Программная реализация расчёта индекса де Мартонна

Обработанные данные присваиваются методу класса, также в структуру dataframe, однако в данном случае псе колонки соответствуют полям в базе данных. Заполненные и рассчитанные показатели, заносятся в базу данных с помощью одноимённого метода. На рисунке 18 представлены

два основных метода класса MonthlyIndex. Название и логическая работа методов в других классах аналогична, реализована с представленным кодом.

```
def add(self,df=None):
    self.data=pd.DataFrame()
    self.data["seljaninov"]=selaninov.gtk(df)
    self.data["de_marton"]=de_martonn.De_Martonna(df)
    self.data.rename_axis(["month"],inplace=True)
    self.data.reset_index(inplace=True)

    self.data.insert(0, 'year', self.year)
    self.data.insert(0, 'index_id', self.index_id)
    self.data["station_id"]=self.station_id


def insert_db(self):
    dbc=DbConnect()
    for index,row in self.data.iterrows():
        dbc.insert(self.table_name,tuple(row))
    dbc.save()
```

Рис. 18. Программная реализация методов класса ежемесячных индексов

Как и описывалось в разделе проектирования системы, методы каждого класса, представляющие табличные данные, имеют только оболочку к доступу к базе данных. Сам код взаимодействия с СУБД расположен в классе посреднике. С помощью модуля python mysqlconnector, происходит процесс взаимодействия с MySql (рис.19).

```
def connect_to_mysql(self,config):
    try:
        return mysql.connector.connect(**config)
    except mysql.connector.Error as err:
        if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
            self.log.add_error("wrong login or password")
            print("Something is wrong with your user name or password")
        elif err.errno == errorcode.ER_BAD_DB_ERROR:
            self.log.add_error("Database does not exist")
            print("Database does not exist")
        else:
            print(err)
            self.log.add_error(err)
        return None



def insert(self,table_name:str,columns:list):

    self.log.add_header("INSERT TO "+table_name)

    values=','.join(['%s'] * len(columns))
    insert_query="INSERT INTO {} VALUES ({})".format(table_name,values)

    cursor=self.cnx.cursor()

    ret=True
    try:
        cursor.execute(insert_query, columns)
    except mysql.connector.Error as err:
        self.log.add_error(
            "INSERT IN {} VALUES {}:".format(table_name,columns)+" "+err.msg)
        ret=False
    else:
        self.log.add_ok(
            "INSERT IN {} VALUES {}".format(table_name,columns))

    cursor.close()
    return ret
```

Рис. 19. Программная реализация методов подключения и записи данных

Вторая программа, отвечающая за представление данных, написана на языке C# и фреймворке ASP.net. Приложение является веб-ориентированным, обусловленным клиент-серверным взаимодействием.

На стороне сервера главной задачей является получать запросы от клиента и предоставлять необходимые данные из базы данных. Связь между

СУБД и компилятором языка C# поддерживает модуль MySql.Data.MySqlClient. На рисунке 20 представлен метод запроса к базе данных.

```
public DataTable Select(string query)
{
    Console.WriteLine(query);
    string sql = query;
    DataTable dataTable = new();
    try
    {
        MySqlCommand cmd = new(sql, conn);
        using (MySqlDataReader reader = cmd.ExecuteReader())
        {
            dataTable.Load(reader);

        }
        return dataTable;
    }
    catch (Exception ex)
    {
        Console.WriteLine("Произошла ошибка при выполнении операции SELECT: " + ex.Message);
        return new DataTable();
    }
}
```

Рис. 20. Программная реализация метода запроса к базе данных

В стандартных структурах данных используемого языка программирования есть объект способный полностью сымитировать таблицу базы данных. Структура dataTable имеет при себе метод прямой загрузки необработанного результата запроса базы данных, с полным сохранением типов данных для каждого поля. Впоследствии данный тип конвертируется в формат json и уже в текстовой форме отправляется клиенту.

На стороне клиента работают функции и методы, написанные на языке JavaScript. Основной задачей клиентской части программы является отображение карты и объектов на ней. Библиотека leaflet предоставляет простую реализацию создания карты (рис.21).

```
//главная переменная карты leaflet
var map = L.map('map').setView([60, 130], 4);

//главная функция добавления и отображения карты leaflet
L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {
    maxZoom: 19,
    attribution: '&copy; <a href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>'
}).addTo(map);
```

Рис. 21. Программная реализация отображения карты

Контейнером для всех видимых элементов сайта служат HTML теги. Для красивого интерфейса отображения элементов используется библиотека CSS bootstrap. Язык JavaScript объединяет использование двух перечисленных инструментов. При генерации динамических элементов, строящихся впоследствии событий, происходящих при нажатии кнопок пользователем. Одно из событий генерирует радио кнопки, с наименованием муниципальных районов выбранной области. Дополнительно к этому, есть возможность получать значения информацию о событиях по имени тега или присвоенных ему атрибутов, таких как id и class. Всё это возможно благодаря встроенному компоненту DOM (Document Object Model). На рисунке 22

представлена функция, которая используя возможности DOM, реагирует на запрос пользователя и отображает на карте несохранные участки.

```
async function show_forest(forest)
{
    //получение выбранной области из всплывающего списка
    let new_area = document.getElementById('select_area').value;

    // выбранные районы
    let district = processCheckboxes();
    //forest["selectedValues"] = selectedValues;

    //если регион поменялся
    if (forest["current_area"] != new_area) {
        //если до этого был выбран другой регион
        if (forest["current_area"] != null) {
            //удаление предыдущего региона с карты
            for (let polygon of forest["polygons"].values()) {
                map.removeLayer(polygon);
            }
            //очищение памяти
            forest["polygons"].clear();
        }
        //новый регион стал текущим
        forest["current_area"] = new_area;

        for (let i = 0; i < district.length; i++)
        {
            await polygon_map_process(new_area, district[i], forest);
        }
    }
    else {
        //очистка полигонов с карты
        for (let polygon of forest["polygons"].values()) {
            map.removeLayer(polygon);
        }

        for (let i = 0; i < district.length; i++) {
            //если район уже загружен, просто отображаем на карте
            if (forest["polygons"].has(district[i])) {
                forest["polygons"].get(district[i]).addTo(map);
            }
            else
            {
                await polygon_map_process(new_area, district[i], forest);
            }
        }
    }

    return forest;
}
```

Рис. 22. Программная реализация отображения лесных кварталов на карте

Основная страница сайта с названием «Карта» представляет собой два окна. С правой стороны большую часть страницы занимает сама карта. С левой стороны боковое меню (рис. 23).

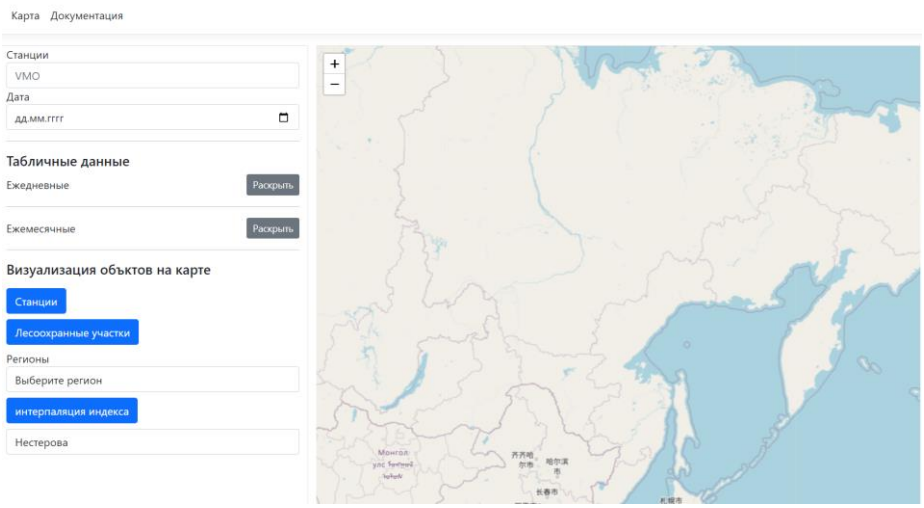

Рис. 23. Страница сайта с картой

Начинать знакомится функционалом лучше всего с кнопки «Станции», при нажатии на карте появляются маркеры, указывающие месторасположения метеостанций. При повторном нажатии кнопки маркеры стираются с карты (рис. 24).

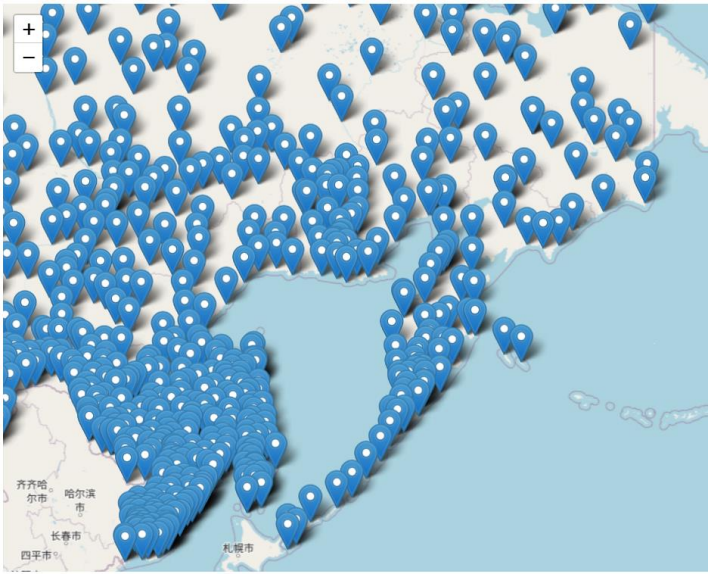

Рис. 24. Отображение всех доступных метеостанций на карте

При нажатии на сам маркер, появляется окно с описанием станции. Нижняя кнопка, предназначена для автоматического заполнения поля «Станции» в боковом меню (рис. 25).

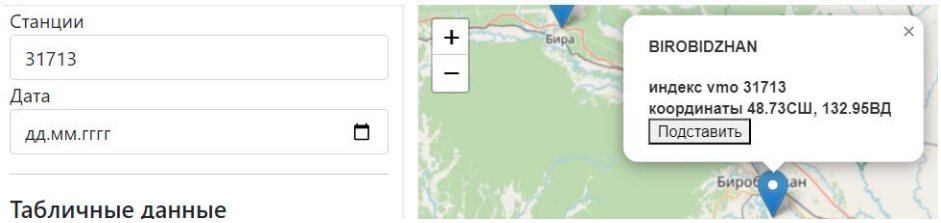

Рис. 25. Отображение всплывающего окна с данными метеостанции

Боковое меню разделено на три логические части. Верхние поля, принимающие номер метеостанции и дату, образуют первую и главную часть, которая относиться ко всем остальным (рис. 26).

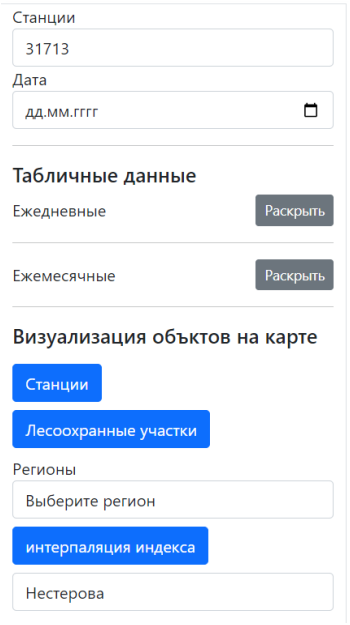

Рис. 26. Боковое меню

Вторая часть предоставляет данные в табличном виде. В качестве полей запроса служит первая часть меню. Для ежемесячных данных, неважно число месяца, внутренний функционал, самостоятельно отделит год и месяц, и отобразит результат (рис. 27).

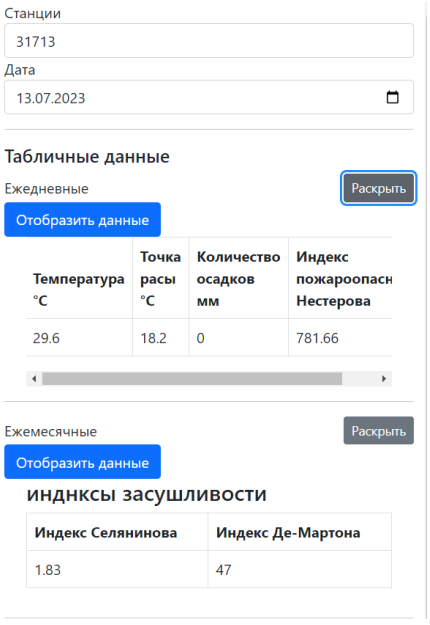

Рис. 27. Часть бокового меню с отображением табличных данных

На третьей части располагаются инструменты отображающие объекты и меняющие их стили на карте (рис. 28).

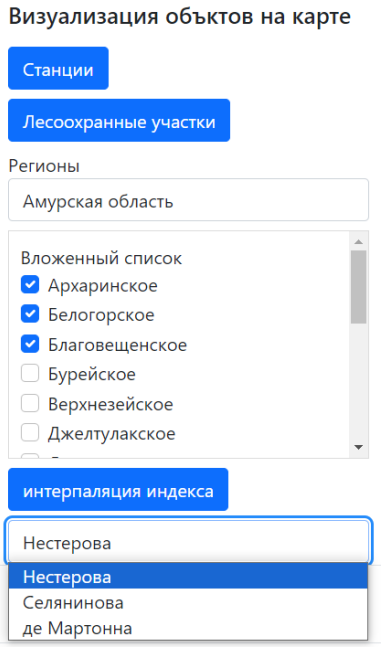

Рис. 28. Часть бокового меню с выбором отображаемых объектов на карте

Чтобы отобразить лесные участки сначала нужно выбрать интересующую область, после чего ниже автоматически появится список радио кнопок, в котором необходимо выбрать муниципальные районы текущей области. В связи с тем, что многие регионы имеют обширную территорию, отображение всех лесных участков области ограничено. После указания шести районов, система блокирует радио кнопки. Каждый лесной квартал, помимо координат, хранит в себе атрибуты, с информацией записанные федеральным агентством лесного хозяйства каждого региона для идентификации местности (рис. 29).

Рис. 29. Лесоохраны участки на территориях разных регионов

На данный момент в открытом доступе имеются векторные данные лесоохранных участков для шести регионов Дальнего Востока. В их число входят: Еврейская автономная область, Хабаровский край, Приморский край, Сахалинская область, Амурская область, Забайкальский край.

Следующее поля изменяет окрас отображаемых участков или при изменении региона, районов, сначала отображает с изменённым окрасом участки. И интерполяция выбранного из списка индекса, производит расчёт на основе ближайшего к той или иной станции с имеющимся индексом на дату указанную в первой части бокового меню. Уровень засухи или пожарной опасности описывается в виде оттенков цветов, от тёмно-синего цвета (менее опасного) до тёмно-красного цвета (чрезвычайно опасного) (рис. 30, 31, 32).

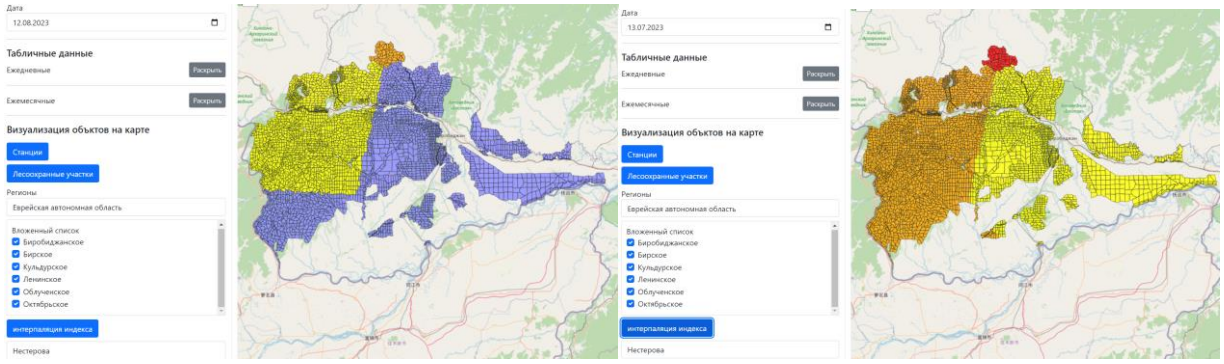

Рис. 30. Интерполяция значений индекса Нестерова в разные даты

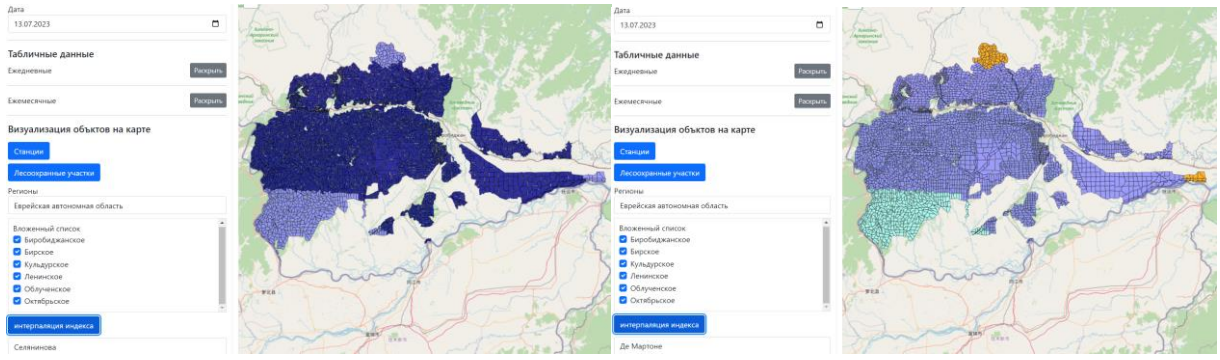

Рис. 31. Интерполяция значений индекса Селянинова и де Мартонна в один период

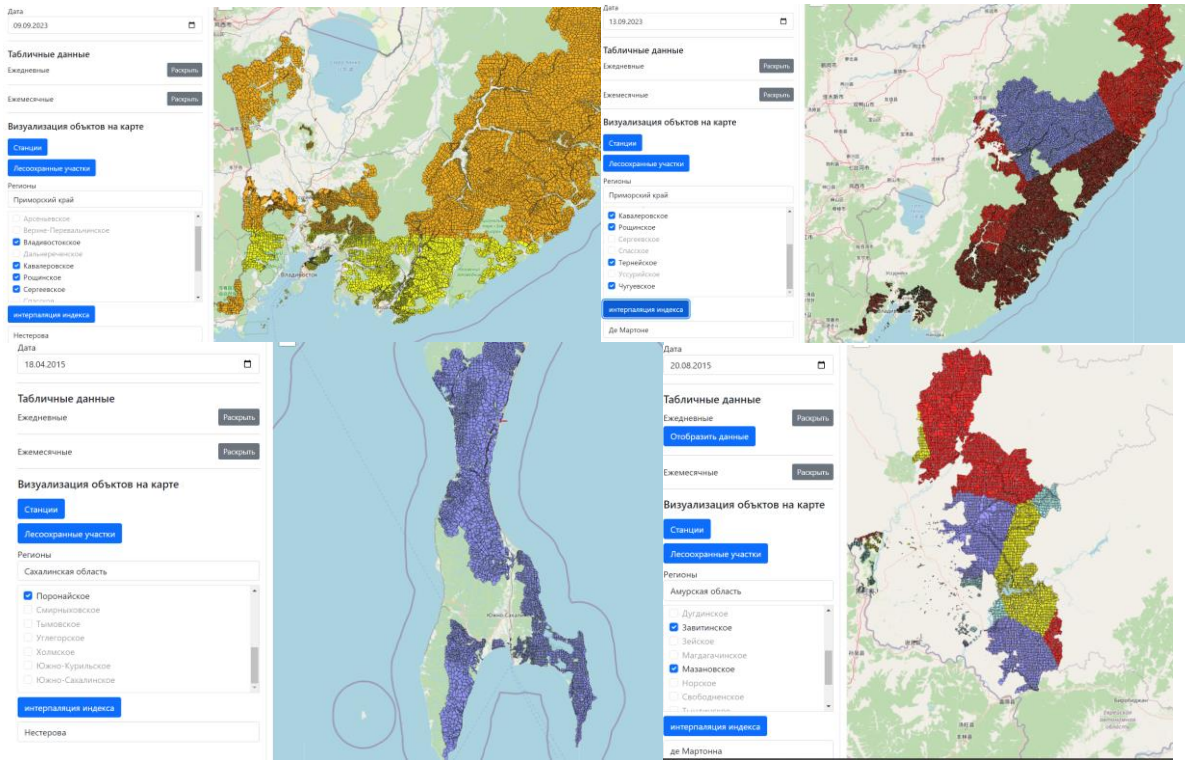

Рис. 32. Интерполяция разных индексов в разных регионах

Вся необходимая информация для пользователей ресурса доступно на странице документации (рис.33).

Показатели данных метеостанций Дальневосточного Федерального Округа РФ

Источник данных

Все доступные показатели взяты или рассчитаны из архивных данных Национального центра экологической информации США (https://www.ncei.noaa.gov/data/global-hourly/), полученных от метеорологических станций, расположенных на территории Далневосточного Федерального Округа РФ.

Показатель количества осадков

Источник: Данные фиксированные метеостанциями

Описание: Сумма выпавших осадков в миллиметрах за день

Пропуски данных:

Если на протяжении всего дня нет информации об осадках, показатель принимает значение 0.

Показатель температура воздуха

Источник: Данные фиксированные метеостанциями

Описание: Максимальное значение температуры воздуха в градусах Цельсия за сутки

Пропуски данных:

Если на протяжении всего дня нет информации о температуре, значение показателья отсутствует.

Индекс пожарной опасности Нестерова (КПО)

Источник: Рассчитан на основе данных метеостанций

Описание:

КПО представляет собой кумулятивную сумму произведения температуры воздуха на разность температур воздуха и точки росы, вычисляемую для отдельного пункта и конкретного времени по формуле:

КПО(N) = КПО(N−1) · K + [t(t - td)](N)

где КПО(N) – значение КПО, рассчитываемое на текущий день, °C; КПО(N-1) – значение КПО, рассчитываемое на предыдущий день, КПО(N-1) = t (t-td), °C; K – коэффициент поправки на осадки (равен единице, если количество осадков менее 3 мм, равен нулю, если количество осадков больше или равно 3 мм); t – максимальная температура воздуха за день, °C; td – максимальная точка росы за день. °C; (t - td) – дефицит точки росы. °C.

Рис. 33. Страница сайта со справочной информацией

В описании к руководству описано необходимая для понимания пользователя информация о том, откуда берутся данные, как рассчитываются

индексы, а также дополнительная информация о доступных возможностях сайта.

Таким образом, была разработана геоинформационная система, выполняющая получение, расчёт, обработку, хранение и представление метеорологических данных связанных с засухой и пожарной опасностью. В процессе разработки были созданы два приложения, взаимодействующие через базу данных. Реализована объектно-ориентированная структура позволяющая работать с данными как с объектами. Построен функционал представления информации для пользователя.

В процессе разработки делался выбор инструментов и технологий разработки, учитывающие особенности проекта. Отбирались различные модули и библиотеки, способствующие ускорению разработки системы.

Была выбрана система управления и спроектирована модель базы данных. В результате исследования предметной области определены основные сущности системы. Описаны модели классов и методов. Разобраны последовательности исполнения основных функций в каждом приложении.

В конечном итоге созданная система будет использована в дальнейшем службами МЧС и фондом охраны лесов, для мониторинга и оценки рисков засухи и пожарной опасности. На основе собранных данных и дополнительной информации о местности в систему будут добавлены инструменты для прогнозирования пожаров на ближайшее время.

Библиографический список

1. Аргучинцева А.В., Голубева Л.В. Индекс пожароопасности // В сборнике: Современные тенденции и перспективы развития гидрометеорологии в России. Материалы II Всероссийской научно-практической конференции, приуроченной к 55-летию кафедры гидрологии и природопользования ИГУ. 2019. С. 303-307.
2. Болданова Е.В. Точка росы как основа оперативного показателя лесопожарной опасности // Известия высших учебных заведений. Лесной журнал. 2024. № 1 (397). С. 114-125.
3. Воропаева Е.В., Кунаева Е.П. Использование ГИС и веб-ГИС технологий для создания электронного образовательного ресурса // В сборнике: Информатизация непрерывного образования - 2018. материалы Международной научной конференции: в 2 т.омах. Под общей редакцией В. В. Гриншкуна. 2018. С. 551-556.
4. Журавель В.В., Осипов П.А., Осипова Я.С., Димухаметов М.О., Осипова Д.А. Web-технологии и ГИС на примере геопорталов и web-ГИС-серверов // В сборнике: Фундаментальные и прикладные научные исследования: актуальные вопросы, достижения и инновации. сборник статей VIII Международной научно-практической конференции: в 4 частях. 2017. С. 115-117.
5. Илякова Р.М., Долгих С.А. Анализ повторяемости почвенной засухи на основе индекса Палмера // Гидрометеорология и экология. 2011. № 3 (62).

С. 50-65.
6. Кадыркулова Н.К., Абдухалимов И.И. База данных в среде ГИС-технологий // Известия Ошского технологического университета. 2022. № 1. С. 84-88.
7. Ларионов А.С., Ботыгин И.А. Построение спектрограмм вейвлет-преобразований показателя индекса засухи Палмера // Научный альманах. 2018. № 4-3 (42). С. 69-72.
8. Максютова Е.В., Макаренко Е.Л., Силаев А.В. Активность лесных пожаров и пожароопасность в байкальском регионе в современных условиях // География и природные ресурсы. 2019. № S5 (159). С. 52-58.
9. Официальный веб-портал Национального управления по аэронавтике и исследованию космического пространства NASA URL: https://www.earthdata.nasa.gov (дата обращения: 18.12.2023).
10. Поляков Д.В., Кужевская И.В. Применение кластерного анализа для оценки температурно-влажностных условий в период активной вегетации на территории юга западной сибири и его связь с гидротермическим коэффициентом Т.Г. Селянинова // Вестник Томского государственного университета. 2012. № 360. С. 188-192.
11. Попов Н.А., Халиуллин Р.Р. Индекс засушливости де Мартонна для республики башкортостан // В сборнике: актуальные проблемы геодезии, картографии, геоинформатики и кадастра. Материалы VI Всероссийской научно-практической конференции, посвященной Дню работников картографии и геодезии. Башкирский государственный университет. 2021. С. 100-104.
12. Потанин В.Г. Усовершенствование ГТК Селянинова для расширения возможностей его применения // Сибирский вестник сельскохозяйственной науки. 2022. Т. 52. № 2. С. 95-104.
13. Смирнов А.А. Оценка формирования пожароопасной обстановки на основе спутниковых и метеорологических данных // Сборник избранных статей научной сессии ТУСУР. 2023. № 1-2. С. 302-305.
14. Федеральное агентство лесного хозяйства URL: https://aviales.ru (дата обращения: 18.12.2023).
15. Черенкова Е.А., Золотокрылин А.Н. О сравнимости некоторых количественных показателей засухи // Фундаментальная и прикладная климатология. 2016. Т. 2. С. 79-94.