Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ЛЕСОСИБИРСКИЙ ПЕДАГОГИЧЕСКИЙ ИНСТИТУТ –
филиал Сибирского федерального университета

Высшей математики, информатики, экономики и естествознания
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
Л.Н. Храмова
подпись инициалы, фамилия
«14» 06 2024 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 Информационные системы и технологии
код-наименование направления

АНАЛИЗ И ОБРАБОТКА ДАННЫХ В СФЕРЕ ЭНЕРГЕТИКИ СРЕДСТВАМИ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON

Руководитель	14.06.24	доцент, канд. пед. наук	А.В. Фирер
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	14.06.2024		А.В. Морозов
	подпись, дата		инициалы, фамилия
Нормоконтролер	14.06.2024		Е.В. Киргизова
	подпись, дата		инициалы, фамилия

Лесосибирск 2024

РЕФЕРАТ

Выпускная квалификационная работа по теме «Анализ и обработка данных в сфере энергетики средствами языка программирования Python» содержит 68 страниц текстового документа, 17 иллюстраций, 44 использованных источника.

PYTHON, АНАЛИЗ ДАННЫХ, ОБРАБОТКА ДАННЫХ, ПРОГРАММНОЕ ПРИЛОЖЕНИЕ

Цель исследования – теоретически обосновать и разработать программное приложение для анализа и обработки данных интеллектуальных приборов учета электроэнергии средствами языка программирования Python на примере ПАО «Красноярскэнергосбыт».

Объект исследования – процесс мониторинга и анализа данных, получаемых с интеллектуальных приборов учёта.

Предмет исследования – проектирование и разработка программных приложений для обработки и анализа данных в сфере энергетики средствами языка программирования Python.

Для достижения поставленной цели необходимо решить следующие задачи:

− на основе анализа предметной области, а также учебной и технической литературы по теме исследования, выявить требования к программному приложению для обработки и анализа данных интеллектуальных приборов учёта электроэнергии, определить инструментальные средства разработки;

− изучить средства языка программирования Python для обработки, анализа и визуализации данных на примере интеллектуальных приборов учёта электроэнергии ПАО «Красноярскэнергосбыт»;

− разработать программное приложение для анализа и обработки данных интеллектуальных приборов учёта электроэнергии средствами языка программирования Python.

В результате выполнения выпускной квалификационной работы разработано программное приложение для обработки и анализа данных в сфере энергетики.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире энергетика является основой для развития основных отраслей, определяющих прогресс общественного производства. Энергетика всегда играла особую роль в жизни человечества. Все виды деятельности человека связаны с затратами энергии. Во всех промышленно развитых странах темпы развития энергетики опережают темпы развития других отраслей. Уровень её развития отражает уровень развития производительных сил общества, возможности научно–технического прогресса и уровень жизни населения и экономики государства.

XX век принес человечеству настоящую революцию в развитии методов получения и использования энергии: были построены тепловые, гидравлические и атомные электростанции огромной мощности, построены высоковольтные линии электропередач, разработаны новые методы получения, преобразования и передачи электроэнергии (управляемая термоядерная реакция, магнитогидродинамический генератор, сверхпроводниковая технология, турбогенераторы), были созданы мощные энергетические системы. В то же время появились мощные системы нефте и газоснабжения.

Федеральный закон от 23.11.2009 N 261-ФЗ «Об энергосбережении и о повышении энергоэффективности в РФ» [39] (далее – Закон N 261-ФЗ) устанавливает, что используемые энергоресурсы подлежат обязательному учету с применением приборов учета.

В связи с ростом энергопотребления и увеличением числа электроприборов, приборов учета тепловой энергии расхода воды, возникает потребность в переходе от устаревших систем учета энергии, способных измерять только объем потребления, к интеллектуальным системам учета. Такие системы включают в себя приборы учета, линии связи, устройства для сбора и передачи данных и информационно-вычислительные комплексы [32].

Интеллектуальные системы учета объединяют в себе функционально связанные устройства, предназначенные для дистанционного сбора, обработки и передачи показаний приборов учета электроэнергии, обеспечивают информационный обмен и хранение данных, позволяют дистанционно управлять компонентами системы и интеллектуальными приборами учета, а также предоставляют информацию о результатах измерений и других параметрах электроэнергии.

В этом контексте проектирование и разработка программного приложения для обработки, анализа и визуализации данных является актуальной задачей, решение которой позволит повысить качество мониторинга потребления электроэнергии, оптимизировать взаимодействие между клиентами и энергосбытовой компанией, а также упростить процессы учета и контроля за выполнением работ. В качестве примера, для данного исследования, были взяты данные, получаемые с ИПУ, энергосбытовой компании ПАО «Красноярскэнергосбыт», которая представляет собой типичный пример организации, работающей в данной сфере.

Цель исследования данной выпускной квалификационной работы – теоретически обосновать и разработать программное приложение для анализа и обработки данных интеллектуальных приборов учета электроэнергии средствами языка программирования Python на примере ПАО «Красноярскэнергосбыт».

Объект исследования – процесс мониторинга и анализа данных в сфере.

Предмет исследования – проектирование и разработка программных приложений для обработки и анализа данных в сфере энергетики средствами языка программирования Python.

Для достижения поставленной цели необходимо решить следующие задачи:

– на основе анализа предметной области, а также учебной и технической литературы по теме исследования, выявить требования к программному приложению для обработки и анализа данных интеллектуальных приборов учёта электроэнергии, определить инструментальные средства разработки;

– изучить средства языка программирования Python для обработки, анализа и визуализации данных на примере интеллектуальных приборов учёта электроэнергии ПАО «Красноярскэнергосбыт»;

– разработать программное приложение для анализа и обработки данных интеллектуальных приборов учёта электроэнергии средствами языка программирования Python.

Актуальность исследования обусловлена необходимостью повышения энергетической эффективности и снижения потерь электроэнергии в ПАО «Красноярскэнергосбыт», а также в целом в российской электроэнергетике, что непосредственно связано с качеством установки и мониторинга интеллектуальных приборов учета электроэнергии. В ходе исследования будут использованы различные методы, включая анализ научно–технической литературы, а также анализ данных и статистическая обработка результатов.

Полученные результаты могут быть использованы для совершенствования процессов мониторинга интеллектуальных приборов учета электроэнергии в ПАО «Красноярскэнергосбыт» и других компаниях.

Также результаты исследования могут быть полезны для научных работников, занимающихся вопросами энергоэффективности, энергосбережения и мониторинга энергосистем, а также для студентов и преподавателей соответствующих специальностей.

Работа состоит из двух основных разделов, включая теоретический обзор, описание процесса разработки приложения, тестирование и выводы.

Практическая значимость исследования – это разработка программного приложения для обработки, анализа и визуализации данных с помощью языка Python.

Результаты исследования докладывались и обсуждались на конференциях:

1.VII Всероссийской научно–практической конференции «Актуальные проблемы преподавания дисциплин естественнонаучного цикла», ЛПИ – филиал СФУ.

2. III Всероссийском молодежном научном форуме «Современное педагогическое образование: теоретический и прикладной аспекты», ЛПИ – филиал СФУ.

По результатам исследования принята к публикации статья:

1. Морозов А.В. АНАЛИЗ И ВИЗУАЛИЗАЦИЯ ДАННЫХ СРЕДСТВАМИ ЯЗЫКА PYTHON / А.В. Морозов // III Всероссийском молодежном научном форуме «Современное педагогическое образование: теоретический и прикладной аспекты», ЛПИ – филиал СФУ – Лесосибирск

1 Проектирование программного приложения для анализа и обработки данных в сфере энергетики средствами языка Python

1.1 Анализ предметной области

ПАО «Красноярскэнергосбыт» является гарантирующим поставщиком электрической энергии на территории Красноярского края и одной из крупнейших энергосбытовых организаций в России, которая входит в группу РусГидро. Статус гарантирующего поставщика, в соответствии с пунктом 36а, постановления правительства РФ от 31 августа 2006 г. N 530 «Об утверждении основных положений функционирования розничных рынков электрической энергии», присвоен компании на основании Приказа Региональной Энергетической Комиссии Красноярского края от 12.10.2006 №37-пр/06.

Исполнительным директором ПАО «Красноярскэнергосбыт» в настоящее время является Олег Владимирович Дьяченко [35].

В результате реформирования ОАО «Красноярскэнерго», 1 октября 2005 года, было образовано ПАО «Красноярскэнергосбыт», а с 2008 года компания вошла в структуру АО «ЭСК «РусГидро» (51,75% ДЗО ПАО «РусГидро»).

Сейчас в составе ПАО «Красноярскэнергосбыт» 5 территориальных управлений, 39 постоянно действующих и 21 выездной офис обслуживания клиентов, которые охватывают территорию от Ермаковского района Красноярского края до Кежемского района.

Структурная схема организации представлена в Приложении В.

Компания обслуживает около 1 миллиона частных и свыше 32 тысяч корпоративных клиентов. Основными принципами взаимодействия с клиентами и широкой общественностью являются высокое качество обслуживания, широкий охват территории, организованность, доступность

информации, социальная ответственность и постоянное расширение спектра предоставляемых услуг.

Помимо продажи электроэнергии, компания предоставляет сопутствующие услуги, такие как продажа и техническое обслуживание электросчетчиков, высоковольтные испытания электрооборудования и масштабный ремонт внутридомовых инженерных систем электроснабжения.

ПАО «Красноярскэнергосбыт» является основным работодателем в Красноярском крае. Штат сотрудников компании насчитывает более тысячи, человек.

Качество, надежность и доступность всех видов энергии зависит в первую очередь от менеджмента, который заключается в формировании процесса строительства энергетических объектов, их обслуживании своевременной модернизации. Государственные и муниципальные органы заинтересованы в стабильном и постоянно развивающемся энергетическом секторе, для чего необходимо организовать мониторинг качества потребляемой электроэнергии [32].

В ст. 3 Федерального закона от 26.03.2003 № 35-ФЗ «Об электроэнергетике» [38] введено понятие ИСУЭЭ, под которой понимается совокупность функционально объединенных компонентов и устройств, предназначенная для удаленного сбора, обработки, передачи показаний приборов учета электрической энергии, обеспечивающая информационный обмен, хранение показаний приборов учета электрической энергии, удаленное управление ее компонентами, устройствами и приборами учета электрической энергии, не влияющее на результаты измерений, выполняемых приборами учета электрической энергии, а также предоставление информации о результатах измерений, данных о количестве и иных параметрах электрической энергии в соответствии с правилами предоставления доступа к минимальному набору функций интеллектуальных

систем учета электрической энергии (мощности), утвержденными Правительством Российской Федерации.

ИСУЭЭ:

− включает одновременно и ИПУ, и ОДПУ (а также радиомодули и программное обеспечение для передачи данных);

− позволяет дистанционно снимать показания ИПУ и ОДПУ;

− позволяет ограничивать подачу коммунального ресурса в многоквартирный дом (частично эти возможности уже реализованы на базе автоматизированной системы коммерческого учёта электроэнергии (АСКУЭ);

− хранит архив данных, срабатывает при нарушении целостности пломб и клейм.

Полный перечень всех возможностей ИСУЭЭ приведен разделе II «Перечень функций интеллектуальной системы учета и требования к ним» [30, п.9] Правил предоставления доступа к минимальному набору функций интеллектуальных систем учета электрической энергии (мощности), которые утверждены постановлением Правительства Российской Федерации от 19 июня 2020 года № 890.

Автоматизированная, информационно–измерительная система, коммерческого учета электроэнергии ПАО «Красноярскэнергосбыт» (далее АСКУЭ) предназначена для измерения активной и реактивной электрической энергии, потребленной за установленные интервалы времени в точках поставки, зарегистрированных за ПАО «Красноярскэнергосбыт» на оптовом рынке электроэнергии и мощности, сбора, хранения и обработки полученной информации.

АСКУЭ представляет собой многофункциональную, многоуровневую автоматизированную измерительную систему с централизованным управлением и распределенной функцией измерений.

Структурная схема АСКУЭ представлена в приложении Б.

Измерительные каналы состоят из двух уровней АСКУЭ:

1-й уровень – измерительно-информационный комплекс (ИИК), включающий в себя измерительные трансформаторы напряжения (ТН), измерительные трансформаторы тока (ТТ), многофункциональные счетчики активной и реактивной электрической энергии (счетчики), вторичные измерительные цепи и технические средства приема-передачи данных;

2-й уровень – информационно-вычислительный комплекс (ИВК), включающий в себя каналообразующую аппаратуру, сервер ИВК, сервер баз данных, устройство синхронизации времени УСВ-2, автоматизированные рабочие места (АРМ) и программное обеспечение (ПО) «Пирамида–2000».

Первичные токи и напряжения трансформируются измерительными трансформаторами в аналоговые сигналы низкого уровня, которые по проводным линиям измерительных цепей поступают на соответствующие входы электронного счетчика электрической энергии. В счетчике мгновенные значения аналоговых сигналов преобразуются в цифровой сигнал. По мгновенным значениям силы электрического тока и напряжения в микропроцессоре счетчика вычисляются мгновенные значения активной, реактивной и полной мощности, которые усредняются за период 0,02 с. Измерительная информация на выходе счетчика без учета коэффициента трансформации:

– активная и реактивная электрическая энергия, как интеграл по времени от средней за период 0,02 с активной и реактивной мощности, соответственно, вычисляемая для интервалов времени 30 мин;

– средняя на интервале времени 30 мин активная (реактивная) электрическая мощность.

Далее данные со счетчиков с помощью каналообразующей аппаратуры передаются на уровень ИВК. Каналообразующая аппаратура состоит из GSM модемов с поддержкой передачи данных по CSD и GPRS каналам местных операторов GSM сети и сети Интернет.

На верхнем уровне системы выполняется сбор данных со счетчиков электроэнергии и дальнейшая обработка измерительной информации, в частности:

– вычисление электроэнергии и мощности с учетом коэффициентов трансформации ТТ и ТН;

– хранение поступающей информации и результатов измерений на сервере БД;

– визуальный просмотр результатов измерений из БД, оформление справочных и отчетных документов.

Сервер ИВК также обеспечивает прием измерительной информации от АСКУЭ утвержденного типа третьих лиц, получаемой в формате XML-макетов в соответствии с регламентами оптового рынка электроэнергии и мощности (ОРЭМ) в автоматизированном режиме посредством электронной почты сети Internet.

Формирование и передача данных прочим участникам и инфраструктурным организациям оптового и розничного рынков электроэнергии и мощности за электронной–цифровой подписью в формате XML-макетов в соответствии с регламентами ОРЭМ осуществляется сервером ИВК по коммутируемым телефонным линиям, каналу связи Internet через интернет–провайдера или сотовой связи.

АСКУЭ оснащена системой обеспечения единого времени (СОЕВ). Для синхронизации по сигналам точного времени от системы глобального позиционирования GPS используется приемник сигналов точного времени УСВ-2. Время сервера ИВК синхронизировано с временем УСВ-2, синхронизация осуществляется при расхождении показания часов УСВ-2 и сервера ИВК на 1 с. Сервер ИВК осуществляет синхронизацию времени серверу БД и счетчикам. При каждом опросе счетчика происходит сличение времени часов счетчиков с временем часов сервера ИВК, корректировка

времени часов счетчиков выполняется при достижении расхождения со временем часов сервера ИВК ± 2 с.

Журналы событий счетчиков, сервера ИВК и сервера БД отображают факты коррекции времени с обязательной фиксацией времени до и после коррекции и (или) величины коррекции времени, на которую было скорректировано устройство.

В АСКУЭ используется ПО «Пирамида–2000». Программное обеспечение имеет защиту от непреднамеренных и преднамеренных изменений, соответствующую уровню «средний» по Р 50.2.077-2014.

Стоит отметить, что в ПО «Пирамида–2000», происходит структурирование данных, которые можно извлечь в формате xlsx, но аналитический инструментарий недостаточно представлен. Поэтому – в организации дальнейшее составление отчётности реализуется в электронных таблицах MS Excel. Это не всегда удобно, так как загружаемый файл содержит большие данные, что требует значительных технических и человеческих ресурсов. Более того, ввиду программы по импортозамещению, возникла необходимость в разработке приложения, которое бы автоматизировало анализ данных ИПУ, выгружаемых из ПО «Пирамида–2000»

1.2 Требования к приложению для анализа и обработки данных

Для создания приложения для анализа и обработки данных необходимо учитывать ряд специфических требований. Перечислим основные функциональные требования, которым должно удовлетворять приложение «Анализ данных ИПУ»

1.2.1 Функциональные требования

1. Загрузка и импорт данных;

− поддержка различных форматов файлов (CSV, XLSX);

− возможность загрузки данных из удаленных источников (API, FTP–сервера, облачные хранилища).

2. Обработка и очистка данных:

− функции для очистки данных (удаление дубликатов, обработка пропусков, преобразование данных);

− возможность фильтрации, группировки и агрегирования данных.

3. Анализ данных:

− статистический анализ (вычисление среднего, медианы, стандартного отклонения и т.д.);

4. Визуализация данных:

− построение различных типов графиков и диаграмм (линейные графики, гистограммы, круговые, scatter, plots и т.д.);

− интерактивные визуализации с возможностью настройки;

− создание дашбордов и отчетов;

5. Сохранение и экспорт результатов:

− возможность сохранения промежуточных и финальных результатов анализа;

− экспорт данных и визуализаций в различные форматы (CSV, XLSX, PDF, изображения).

1.2.2 Нефункциональные требования

1. Производительность:

− оптимизация обработки больших объемов данных;

− эффективное использование памяти и процессорных ресурсов.

2. Масштабируемость:

− возможность обработки увеличивающихся объемов данных;

− поддержка распределенных вычислений и параллельной обработки.

3. Операционные требования

− кроссплатформенность (поддержка Windows, macOS, Linux);

− возможность развертывания на локальных серверах и в облаке.

4. Обновление и сопровождение:

− регулярные обновления и исправления ошибок;

− техническая поддержка пользователей.

Соблюдение вышеописанных требований позволит создать эффективное и удобное приложение для анализа и обработки данных, получаемых с интеллектуальных приборов учёта, которое удовлетворит потребности пользователей и обеспечит качественный анализ данных, полученных с ПО «Пирамида–2000».

1.3 Выбор инструментальных средств для анализа и обработки данных в сфере энергетики

Визуализация аналитических данных – это представление информации в виде рисунков, графиков и диаграмм, с использованием интерактивных возможностей и анимации как для получения результатов, так и для использования в качестве исходных данных для дальнейшего анализа. Важный этап анализа данных, позволяющий представить самые важные результаты анализа в наиболее удобном для восприятия виде. Как визуализация аналитических данных помогает в анализе больших объёмов данных и в получении наглядной информации для мониторинга данных ИПУ. Методы визуализации предлагают новый подход к аналитике. Они позволяют комбинировать базовые знания и креативность с огромными

возможностями хранения и обработки в современных компьютерах, чтобы получить представление о сложных проблемах.

Аналитическая визуализация является наукой аналитического мышления, поддерживаемой интерактивными визуальными интерфейсами. Они сочетают статистические методы и модели с передовыми методами интерактивной визуализации, чтобы помочь скрыть сложность больших массивов данных и принять обоснованные решения.

По данным исследования П. Морозовой [29], был произведён сравнительный анализ языков программирования для анализа и визуализации данных.

1. Язык программирования Python

Python — это высокоуровневый язык программирования общего назначения, известный своей простотой и читаемостью. Он был создан Гвидо ван Россумом и впервые выпущен в 1991 году. Название языка было взято из названия британского комедийного шоу «Monty Python's Flying Circus», популярного в те годы.

По мнению У.Маккинни [25], из всех интерпретируемых языков программирования Python выделяется крупным и активным сообществом научных расчетов и анализа данных.

За последние два десятилетия язык программирования Python прошел путь от инструмента для научных расчетов до одного из ключевых языков, используемых в науке о данных, машинном обучении и разработке программного обеспечения в академических и промышленных кругах. На протяжении десятилетий Python приобрел репутацию одного из самых популярных языков благодаря своей простоте и эффективности.

В настоящее время практически каждый второй ИТ-специалист обладает базовыми знаниями разработки на Python, и эта тенденция, несомненно, будет усиливаться [6]. Будучи интерпретируемым и

динамически типизируемым языком, Python не требует компиляции перед выполнением кода, что делает его удобным и доступным для использования.

Особая привлекательность Python для начинающих программистов обусловлена его простым и понятным синтаксисом. С помощью всего нескольких строк кода можно создавать интуитивно понятные графические интерфейсы пользователя (GUI) и игровые графические интерфейсы (GGI).

2. Язык программирования R

Одним из старейших языков в анализе и обработке данных является язык программирования К. Он широко применяется для организации данных в таблицы, их обработки, проведения статистических тестов и создания графических отчетов. Как и язык программирования Python, язык R является универсальным языком, используемым в научных исследованиях в различных областях, таких как маркетинг, а также для задач машинного обучения и статистического анализа данных.

Для эффективного использования R рекомендуется знание математического анализа, теории вероятностей и статистических методов, что делает его особенно популярным в научной среде. R также считается одним из ключевых языков программирования в области Data Science.

Для R существует множество библиотек и расширений, которые облегчают визуализацию данных, выполнение быстрых статистических операций и распознавание текстов. В отличие от Python, который используется для более широкого спектра задач, включая веб-разработку, искусственный интеллект и проекты машинного обучения, R нацелен в первую очередь на статистическое моделирование, глубокое обучение и анализ данных.

Хотя Python является более универсальным и обладает понятным синтаксисом, функциональные возможности R специально разработаны для статистических приложений, что обеспечивает значительные преимущества для визуализации данных и других связанных процессов. Если говорить о

сильных сторонах каждого языка, Python предпочтительнее для широкого круга задач, включая искусственный интеллект, машинное обучение, веб-разработку и многие другие области, благодаря большому сообществу специалистов.

Тем не менее, R может быть более эффективным инструментом для задач обучения и оценки алгоритмов машинного обучения и глубокого обучения, простого управления данными, визуализации данных и их анализа, предлагая более конкретный подход к обработке данных по сравнению с Python.

3.Язык программирования Java

Язык программирования Java, как и Python, также является универсальным языком, на котором работают с большими данными и создают приложения. Кроссплатформенность языка позволяет написанный на нём код запускать на смартфонах, консолях, серверах и устройствах умного дома. Это происходит благодаря чему код Java запускается на специальной виртуальной машине, поддержка которой есть у большинства современных устройств. На Java написаны многие инструменты для работы с большими данными, например почти вся экосистема Hadoop.

Язык имеет строгую типизацию в отличие от языка Python, в Java необходимо явно указывать типы переменных при создании. Ошибки в данных из-за типизации не будет. Большинство приложений на Java работают быстрее, чем на других языках программирования. Язык программирования Java полностью основан на объектно-ориентированных моделях программирования и моделях на основе

Краткий синтаксис Python делает его гораздо лучшим вариантом для специалистов, которые хотят использовать язык программирования для интеллектуального анализа данных, нейронной обработки, машинного обучения или статистического анализа по сравнению с синтаксисом Java,

который длинный и трудночитаемый. Для написания той же программы на Java требуется больше строк кода, например, чем на Python.

Ещё одним из недостатков языка Java, является то, что для крупномасштабной разработки приложений может потребоваться платная лицензия.

По сравнению с Java, которая сначала компилируется в байтовый код, затем должна быть скомпилирована в машинный код с помощью виртуальной машины Java (JVM), код Python требует меньше ресурсов для запуска, поскольку он напрямую компилируется в машинный код [3].

2. Язык программирования Scala

Scala, разработанный Мартином Одерски, является объектно-ориентированным языком программирования, который также поддерживает функциональный стиль программирования в значительном масштабе. Его название происходит от сочетания слов «масштабируемый» и «язык», что указывает на его способность масштабироваться в зависимости от потребностей пользователей. Со временем Scala стала одной из самых популярных технологий среди разработчиков.

Scala особенно востребована в области анализа и визуализации данных и идеально подходит для работы с большими данными благодаря своей высокой производительности. Крупные компании, такие как Twitter, Netflix и Тинькофф, активно используют этот язык. Поскольку Scala работает на виртуальной машине Java (JVM), она обладает отличной совместимостью с Java и может выполняться на любых устройствах.

Одним из ключевых инструментов для работы с большими данными, Apache Spark, является фреймворком, написанным на Scala. Scala имеет много общего с Java, включая возможность писать кроссплатформенный код, что позволяет интегрировать инструменты анализа данных в различные приложения. Поддержка объектно-ориентированного и функционального

программирования делает Scala гибким языком, сочетающим лучшие черты обоих подходов.

Scala является предпочтительным языком для обработки больших данных, так как обеспечивает высокую скорость работы через параллельные вычисления. Тем не менее, её синтаксис сложнее, чем у Python и Java, что усложняет процесс изучения и чтения кода. Медленная компиляция также является недостатком Scala, особенно в средах, где необходимы частые обновления. Однако для задач анализа данных это не критично.

Найти информацию о Scala может быть сложнее по сравнению с другими языками, что добавляет дополнительных трудностей для разработчиков.

3. Язык программирования GO

Golang, также известный как Go – статически типизированный, компилируемый многопоточный язык с открытым исходным кодом. В основном его применяют в веб-сервисах и клиент-серверных приложениях [34].

Этот язык был разработан Google специально для анализа и обработки больших данных. Язык создали Роб Пайк и Кен Томпсон. Оба – культовые личности в computer science и в прошлом сотрудники легендарной Bell Labs. А Томпсон к тому же один из создателей ОС UNIX и языка B (предшественника C).

Ориентирован прежде всего на извлечение и анализ информации из БД, хотя также применяется для работы с искусственным интеллектом и веб-разработки.

Отличает этот язык от других это максимальная простота. Язык очень дружелюбный к новичкам, у него низкий порог вхождения и простой синтаксис. С его изучением точно не будет проблем. Язык имеет множество стандартных библиотек. В том числе для анализа данных и работы с базами. Ведущие специалисты корпорации Кен Томпсон и Роберт Гризер создали

более удобный язык, способный упростить работу по поддержке программного обеспечения в современной вычислительной среде. Golang выразителен, лаконичен, чист, эффективен и быстро компилируется в машинный код. В целом он похож на C, например, по части синтаксиса и объектно-ориентированной терминологии. Программы на этом языке быстро компилируются и так же быстро работают. У языка нет устаревших конструкций, он создан с учётом современных стандартов программирования.

Язык Go плохо подходит для написания масштабных и сложных проектов – у него не хватает для этого возможностей. Чаще всего на нём пишут отдельные микросервисы. Из-за новизны и малой распространённости сложно бывает найти ответы на вопросы.

4. Язык программирования Julia

Это новый язык, который разрабатывали специально для работы с данными. Его используют в машинном обучении, создании компьютерных симуляций и для написания бэкенд части приложений.

Язык Julia был разработан для высокой производительности. Программы Julia автоматически компилируются в эффективный машинный код через LLVM и поддерживают несколько платформ.

Он имеет динамическую типизацию, похож на язык сценариев и имеет хорошую поддержку для интерактивного использования, но также может быть скомпилирован отдельно [42].

Функции языка определяются на основе различных комбинаций типов аргументов и применяются путем отправки к наиболее конкретному соответствующему определению. Эта модель хорошо подходит для математического программирования, где неестественно, чтобы первый аргумент «владел» операцией, как в традиционной объектно-ориентированной диспетчеризации. Операторы – это просто функции со специальными обозначениями - чтобы расширить добавление к новым

пользовательским типам данных, необходимоопределить новые методы для функции. Затем существующий код легко применяется к новым типам данных.

Частично из-за вывода типов во время выполнения (дополненного необязательными аннотациями типов), а частично из-за сильного внимания к производительности с самого начала проекта вычислительная эффективность Julia превосходит эффективность других динамических языков и даже конкурирует со статически скомпилированными языками. Для крупномасштабных числовых задач скорость всегда была, продолжает быть и, вероятно, всегда будет иметь решающее значение: объем обрабатываемых данных на протяжении последних десятилетий легко соответствовал Закону Мура.

Воспроизводимые среды позволяют воссоздавать одну и ту же среду Julia каждый раз на разных платформах с помощью готовых двоичных файлов. Julia использует множественную диспетчеризацию в качестве парадигмы, упрощающей выражение многих объектно-ориентированных и функциональных шаблонов программирования.

Недостаток языка в том, что у него мало библиотек, они плохо структурированы, часто несовместимы и нередко имеют плохую документацию. Работать с ними сложно, и многое приходится писать с нуля.

5. Язык программирования C++

Это универсальный язык общего назначения, на нём пишут различное программное обеспечение, не только прикладное, но и системное. Однако, в сфере больших данных его применяют редко. Это обусловлено отсутствием большого количества библиотек. Но благодаря тому, что это один из самых быстрых языков программирования, его применяют в случаях, когда нужно принимать решения быстро. В основном на C++ пишут инструменты для анализа данных.

C++ существует давно и он очень популярен из-за высокой скорости, поэтому у него большая аудитория активных пользователей.

У C++ непростой синтаксис и много нюансов. Порог входа высокий, особенно если нужно решать такие специфические задачи, как анализ данных.

У языка много преимуществ, но и также есть недостатки. Отсутствие готовых инструментов для анализа данных большинство инструментов придётся писать с нуля, что отнимет много времени.

Опираясь на вышеописанный обзор, был сделан краткий анализ языков программирования в контексте обработки и анализа данных.

Python – самый популярный и универсальный, но работает медленнее некоторых других и может вызывать ошибки в данных из-за динамической типизации.

R идеально подходит для комплексной аналитики и содержит тысячи готовых функций, но очень непрост в освоении и почти неприменим для других задач программирования.

Java быстрый и ещё более универсальный, чем Python, но сложнее в освоении и имеет меньше встроенных инструментов для анализа данных.

Scala быстро работает с большими данными, но сложен в освоении и не очень популярен.

Go очень простой и содержит много стандартных библиотек, но пока слишком молодой и не подходит для масштабных проектов.

Julia разработана специально для работы с данными. Язык быстрый и удобный. Но из-за новизны пока содержит мало готовых функций и библиотек.

C++, скорее, язык общего назначения, и для повседневной работы аналитика данных не подходит. Но он очень быстрый, поэтому выручит там, где нужна максимальная скорость работы программы.

2 Разработка программного приложения для анализа и обработки данных в сфере энергетики средствами языка Python

2.1 Средства языка программирования Python для обработки, анализа и визуализации данных

Опираясь на исследования в статье «Рейтинг самых используемых языков программирования за 2023 год по версии крупнейшей социальной сети для разработчиков» [36], а также проведя сравнительный анализ языков программирования, был сделан выбор в пользу языка программирования Python, как самого распространённого и легко изучаемого языка.

Выбор языка Python в качестве первого языка программирования имеет ряд преимуществ:

Простой синтаксис: код легко читается и понимается благодаря операторам, завершающимся концом строки, и структуре блока, определяемой отступами.

Динамическая типизация: не нужно объявлять переменные, что делает код лаконичным и позволяет избежать распространённых ошибок.

Обширная сфера применения: Python используется в науке, веб-разработке, Machine Learning, создании игр и сложных визуальных эффектов.

Python — популярный язык, востребованный на рынке труда, у него есть огромное сообщество разработчиков и доступ к обширной документации и ресурсам.

Лёгкость в изучении: Python прост в освоении, с понятным и лёгким синтаксисом.

Таким образом, Python является отличным выбором для новичков, так как обладает простым синтаксисом, высокой степенью универсальности и востребованностью на рынке труда. Python отлично работает на всех этапах. В первую очередь в этом помогают различные библиотеки. Поиск, обработка

и моделирование, включая визуализацию, это три самых популярных сценария использования языка для анализа данных.

Некоторые из библиотек крайне узкоспециализированы, а какие-то из них могут использоваться почти для любой задачи. Ниже представлены наиболее популярные Python библиотеки для обработки и анализа данных.

Библиотека Pandas — это программная библиотека на языке Python для обработки и анализа данных. Она предоставляет специальные структуры данных и операции для манипулирования числовыми таблицами и временными рядами. Название библиотеки происходит от эконометрического термина «панельные данные». pandas активно используется в среде Python для сбора, очистки, анализа и моделирования данных [11].

С помощью библиотеки Pandas можно загружать и обрабатывать данные из различных источников, переименовывать, сортировать, объединять фрейма данных, а также обновлять, добавлять, удалять столбцы из фреймов данных, восстанавливать недостающие данные типа NaN.

NumPy – это универсальная библиотека для обработки массивов, которая предоставляет собой высокопроизводительные объекты и инструменты для работы с массивами. С этой библиотекой можно реализовать множество задач, для работы с массивами данных.

SciPy — это библиотека для языка программирования Python, предназначенная для выполнения научных и инженерных расчётов. Она была создана в 2001 году Трэвисом Олифантом, Эриком Джонсом и Пиару Петерсоном на основе пакета Numeric.

Возможности SciPy включают поиск минимумов и максимумов функций, вычисление интегралов функций, поддержку специальных функций, работу с генетическими алгоритмами и решение обыкновенных дифференциальных уравнений.

Основной структурой данных в SciPy является многомерный массив, реализованный модулем NumPy. Также доступны различные модули, такие как integrate, interpolate, signal и другие.

Для визуализации результатов расчётов часто применяется библиотека Matplotlib, являющаяся аналогом средств вывода графики MATLAB.

Matplotlib – это ещё одна библиотека, позволяющая строить двумерные графики, позволяет создавать диаграммы с данными.

Библиотека позволяет работать с данными на нескольких уровнях, с помощью модуля Pyplot, который рассматривает график как единое целое и через объектно-ориентированный интерфейс, где каждая фигура или её часть является отдельным объектом, – это позволяет выборочно менять их свойства и отображение.

Matplotlib используют для визуализации данных любой сложности. Библиотека позволяет строить разные варианты графиков: линейные, трёхмерные, диаграммы рассеяния и другие, а также комбинировать их.

Широкий спектр графиков и диаграмм превращают набор данных в доступную и наглядную визуальную информацию.

Ниже представлены примеры графиков с листингом кода.

На рисунке 1 представлен пример линейной диаграммы библиотеки Matplotlib.

```python
# Группировка данных по дате установки ПУ и подсчет количества ПУ
pu_counts = df.groupby('Type').size().reset_index(name='Количество ПУ')
# Построение диаграммы Радар
plt.figure(figsize=(10, 6))

dates = pu_counts['Type']
counts = pu_counts['Количество ПУ']
plt.plot(dates, counts, marker='o')
plt.title('Количество установленных ПУ')
plt.xlabel('Типы ПУ')
plt.ylabel('Количество установленных ПУ')
plt.xticks(rotation=45)  # Поворот дат на 45 градусов для лучшей читаемости
plt.grid(True)
plt.tight_layout()
plt.show()
```

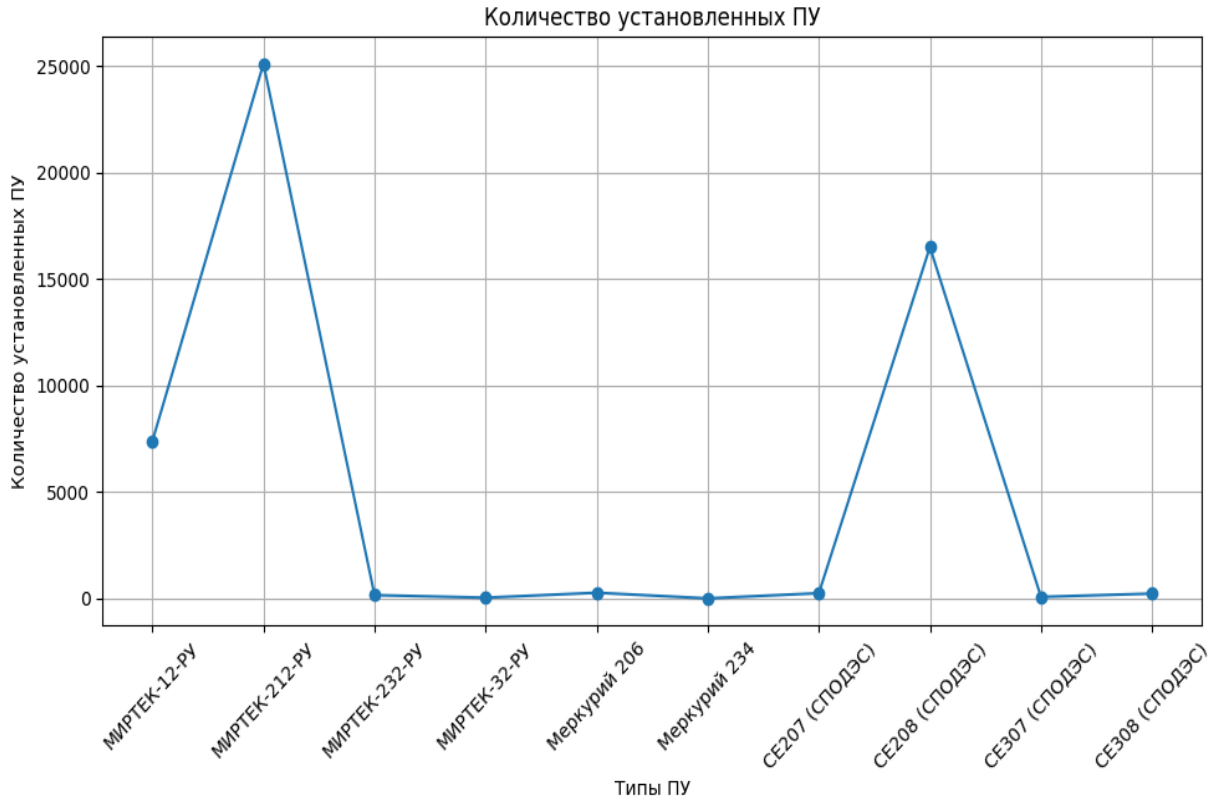


Рисунок 1 – Листинг программы и линейная диаграмма, построенная средствами библиотеки Mathplotlib

Классический график столбчатой диаграммы, наглядно показывающий сравнение по количеству типов приборов учёта, представлен на рисунке 2.

```python
type_counts = df['Type'].value_counts()

# Построение столбиковой диаграммы (гистограммы)
plt.figure(figsize=(10, 6))
type_counts.plot(kind='bar')
plt.title('Столбиковая диаграмма количества типов ПУ')
plt.xlabel('Тип ПУ')
plt.ylabel('Количество')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```

✓ 0.5s

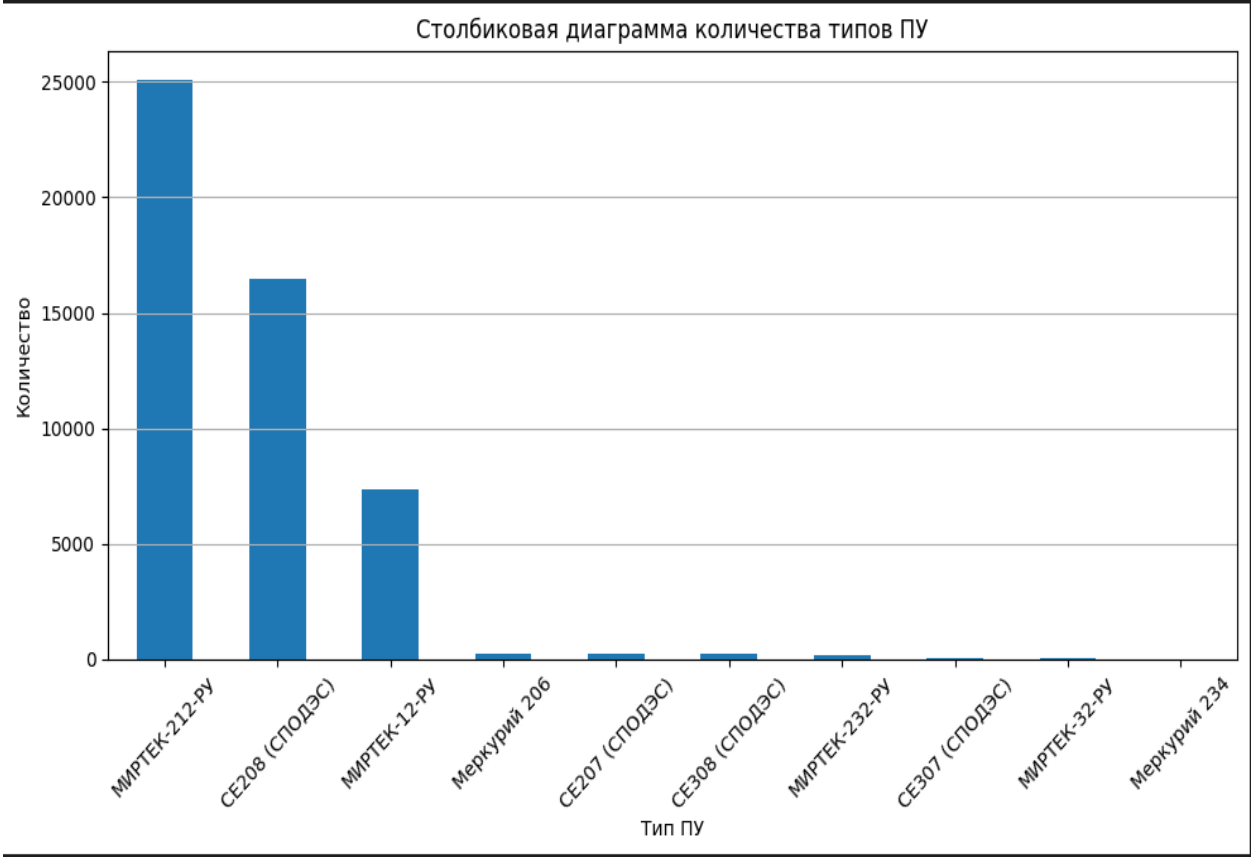


Рисунок 2 – Листинг программы и столбчатая диаграмма, построенная средствами библиотек Mathplotlib

Точечный график наглядным образом показывает плотность зависимости от даты установки. Результаты показаны на рисунке 3.

```python

# Построение графика по столбцу "Дата"
plt.figure(figsize=(10, 6))
plt.plot(df['Data'], df['Type'], marker='o', linestyle=' ')
plt.title('График зависимости типа ПУ от даты')
plt.xlabel('Дата установки ПУ')
plt.ylabel('Тип ПУ')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()

```

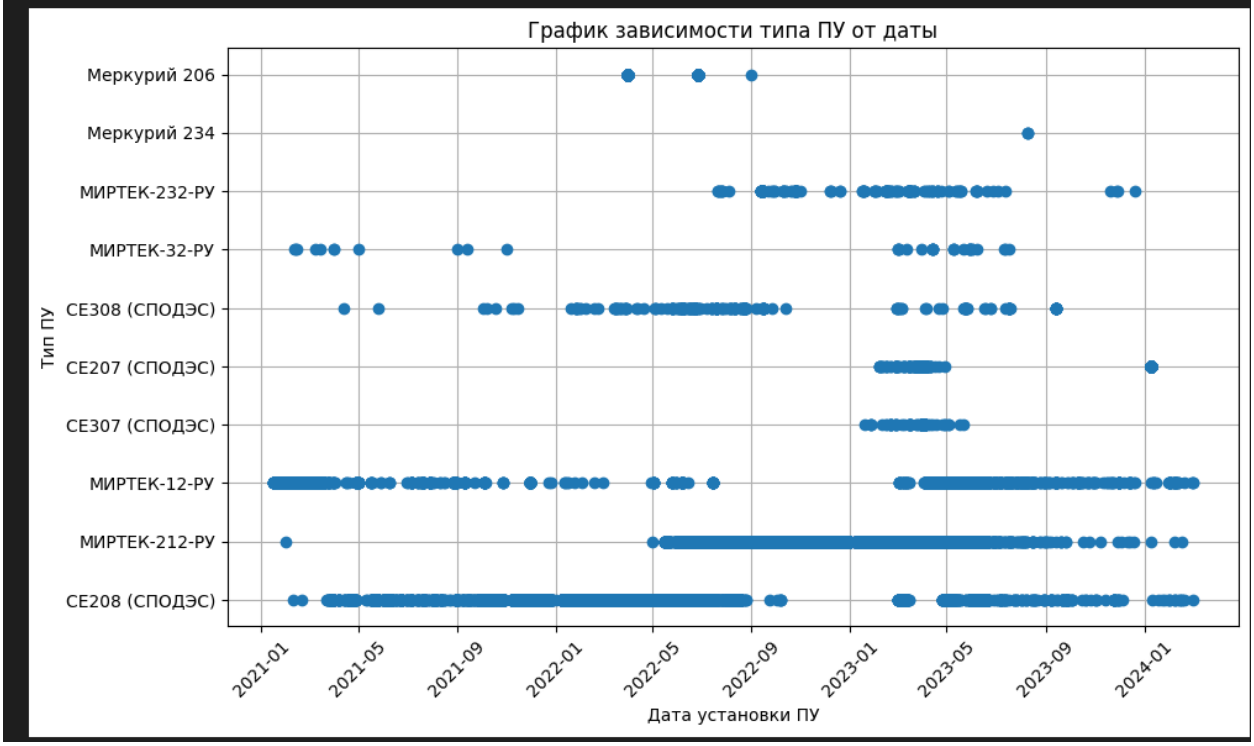


Рисунок 3 – Листинг программы и точечный график, построенный средствами библиотеки Matplotlib(Plotly)

Seaborn функционально очень похож с Matplotlib, однако Seaborn предоставляет множество шаблонов визуализации с меньшим количеством синтаксических правил, причем более простых. Seaborn предоставляет следующие возможности:

– определение отношения между несколькими переменными (корреляция);

– соблюдение качественных переменных для агрегированных статистических данных;

– анализ одномерных или двумерных распределений с последующим сравнением их между различными подмножествами данных;

– построение модели линейной регрессии для зависимых переменных;

– обеспечение многоуровневых абстракций, многосюжетных сеток.

На рисунке 4 изображен график плотности – наглядная карта, позволяющая оценить одновременную зависимость, благодаря цветным перекрытиям.

```python
# Подсчет количества уникальных типов ПУ
type_counts = df['Type'].value_counts()

# Построение графика плотности в зависимости от даты для каждого типа ПУ
plt.figure(figsize=(12, 8))
for pu_type in type_counts.index:
    sns.kdeplot(df[df['Type'] == pu_type]['Data'], label=pu_type, cmap='CMRmap_r', fill=True, thresh=1, warn_singular=False)

plt.title('График плотности в зависимости от даты установки для различных типов ПУ')
plt.xlabel('Дата')
plt.ylabel('Плотность')
plt.legend()
plt.tight_layout()
plt.show()
```

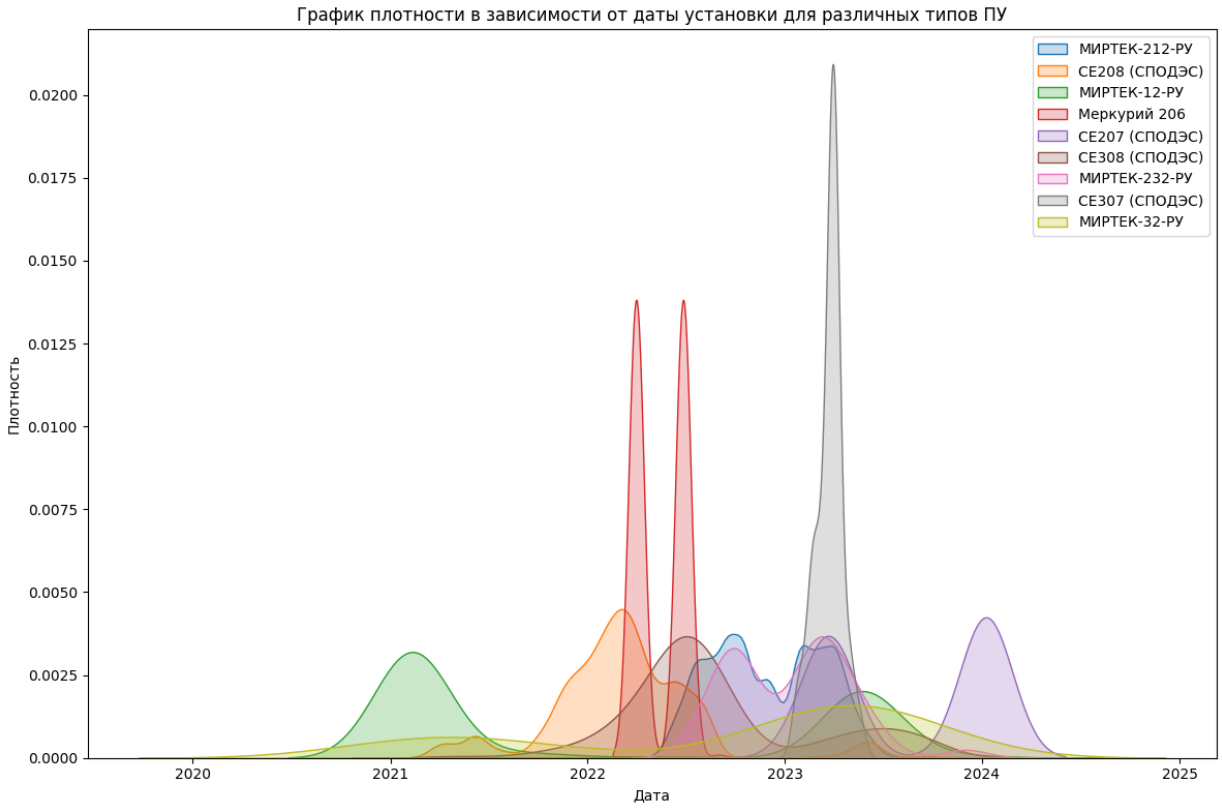


Рисунок 4 – Листинг программы и график плотности, построенный средствами библиотек Mathplotlib/Seaborn

На рисунке 5 представлена тепловая карта. График показывает распределение по «температурному» принципу и показывает цветной градиент, в зависимости от количества.

```python
# Подсчет количества уникальных типов ПУ
type_counts = df['Type'].value_counts()

# Построение графика плотности в зависимости от даты для каждого типа ПУ
plt.figure(figsize=(12, 8))
for pu_type in type_counts.index:
    sns.kdeplot(df[df['Type'] == pu_type]['Data'], label=pu_type, cmap='CMRmap_r', fill=True, thresh=1, warn_singular=False)

plt.title('График плотности в зависимости от даты установки для различных типов ПУ')
plt.xlabel('Дата')
plt.ylabel('Плотность')
plt.legend()
plt.tight_layout()
plt.show()
```

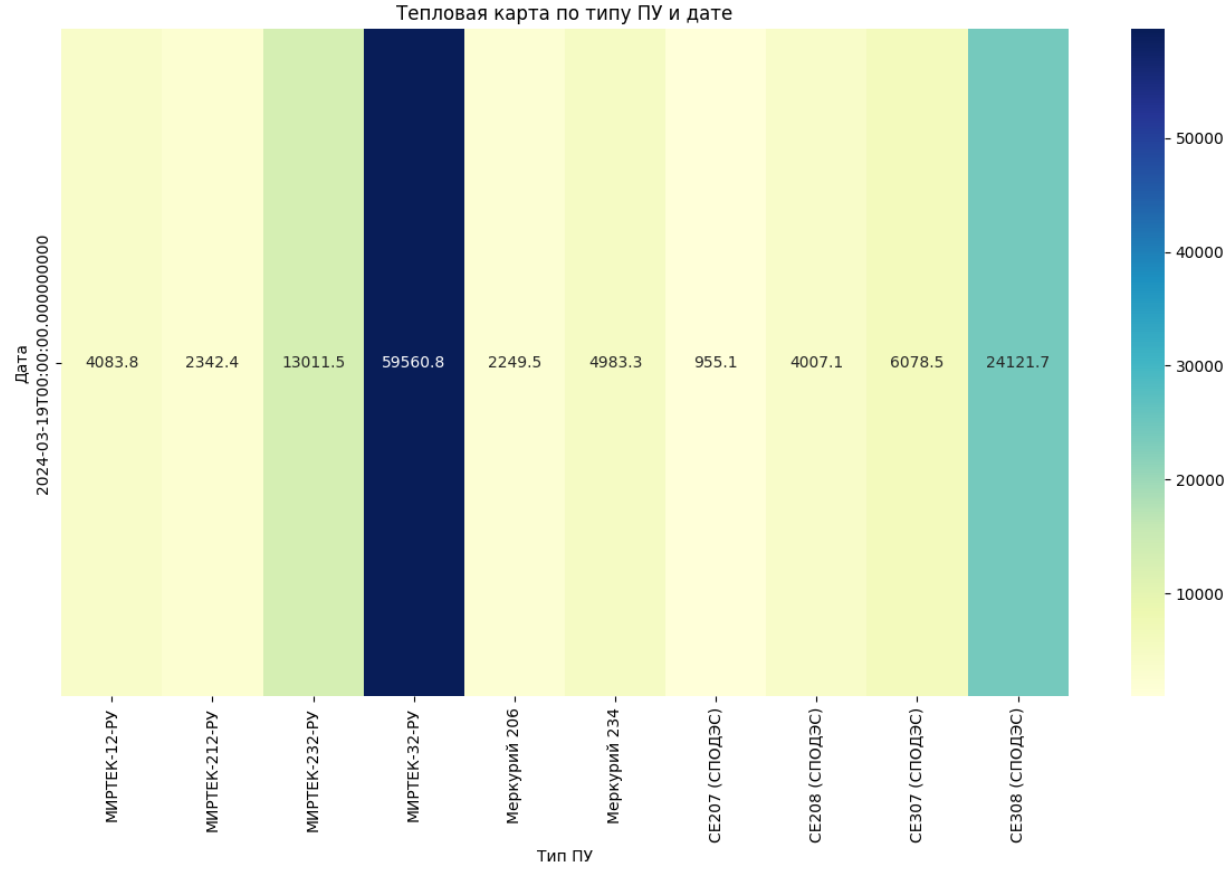


Рисунок 5 – Изображение тепловой карты, с использованием функции Seaborn sns.heatmap

На рисунке 6 показана круговая диаграмма, построенная с помощью библиотеки Seaborn. График отображает наиболее наглядно количество установленных приборов учёта в процентном соотношении. С помощью

библиотеки NumPy были произведены расчёты и отображены на круговой диаграмме

```python
#Подсчет количества уникальных значений в столбце "Тип ПУ"
pu_counts = df['Type'].value_counts()
# Подсчет общего количества записей
total_count = df.shape[0]
# Расчет процента для каждого типа ПУ
pu_percentages = pu_counts / total_count * 100
# Построение круговой диаграммы
plt.figure(figsize=(15, 8))
pu_counts.plot(kind='pie', autopct='%1.1f%%', startangle=140, labels=None)
plt.title('Процентное распределение типов ПУ')
plt.axis('equal')
# Вывод типов ПУ и их процентного соотношения
for i, (pu, count) in enumerate(zip(pu_counts.index, pu_counts)):
    plt.text(-2, 1 - i * 0.1, f'{pu}: {count} ({pu_percentages[i]:.1f}%)', fontsize=10)
plt.tight_layout()
plt.legend(pu_counts.index, loc='upper right')
plt.show()
```

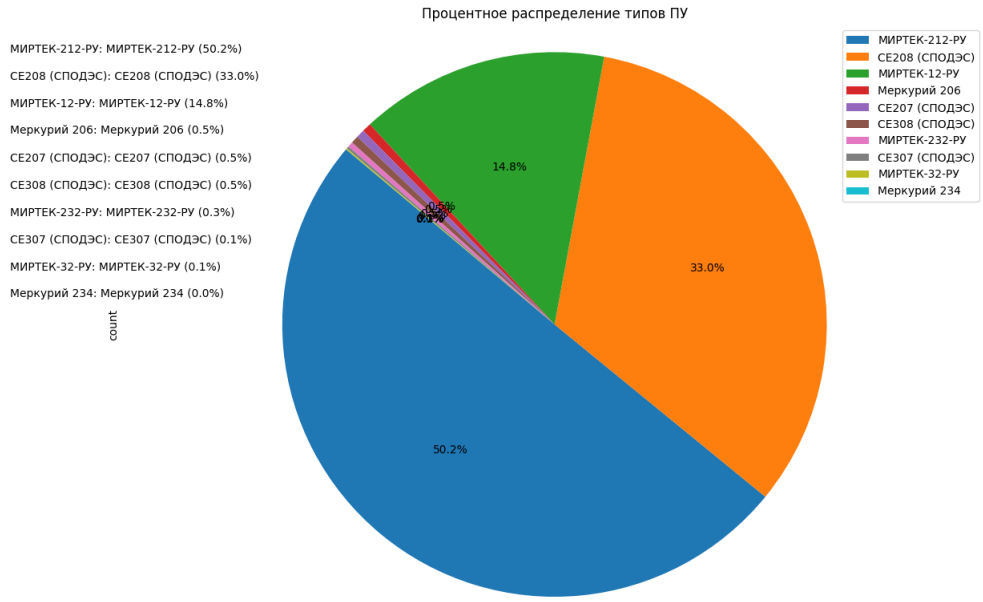


Рисунок 6 – Круговая диаграмма процентного соотношения по типам приборов, с использованием средств библиотеки NumPy и Seaborn

Для вывода всех графиков и диаграмм было решено использовать фреймворк Streamlit. Это так называемый «Дашборд» (Dashboard), который служит для показа различных элементов, графиков, таблиц в web-браузере.

Streamlit – это фреймворк для языка программирования Python. Он содержит набор программных инструментов, которые помогают перенести приложения в web-браузер.

У фреймворка открытый исходный код, так что посмотреть, как он устроен, может любой разработчик. Язык Python, для которого предназначен Streamlit, активно используется в машинном обучении, анализе данных и автоматизации: это именно те сферы, где может пригодиться такой фреймворк [8].

Благодаря этому фреймворку можно развернуть модель в веб-приложение, визуализировать и привести к удобному для восприятия виду можно в несколько строчек кода. Не нужно тратить время на написание веб-интерфейса вручную – за человека это делает фреймворк. Так можно сэкономить время и потратить его на более профильные задачи: сконцентрироваться на анализе данных, а не на интерфейсе для модели.

Для визуализации интерактивных карт была использована библиотека Folium.

Folium – это мощная библиотека визуализации данных в Python, которая была создана в первую очередь для того, чтобы визуализировать геопространственные данные. Простая в использовании и производительная библиотека folium сочетает в себе мощь leaflet и простоту Python, и это делает ее отличным инструментом для создания интерактивных карт на Python. Она позволяет использовать возможности leaflet из Jupyter и легко получать доступ к структурам данных Python (например, pandas.DataFrame).

С помощью Folium можно создать карту и наложить маркеры, а также кластеры маркеров поверх карты [9].

Folium имеет восемь встроенных подложек из OpenStreetMap, Mapbox и Stamen для придания картам особого стиля, а также поддерживает настраиваемые наборы тайлов с API Mapbox или Cloudmade. С Folium можно работать с файлами GeoJSON и TopoJSON, создавать фоновые картограммы

с палитрами от color-brewer, настраивать всплывающие подсказки и интерактивные карты-врезки.

На рисунке 7 представлен пример карты с изображёнными на ней маркерами.

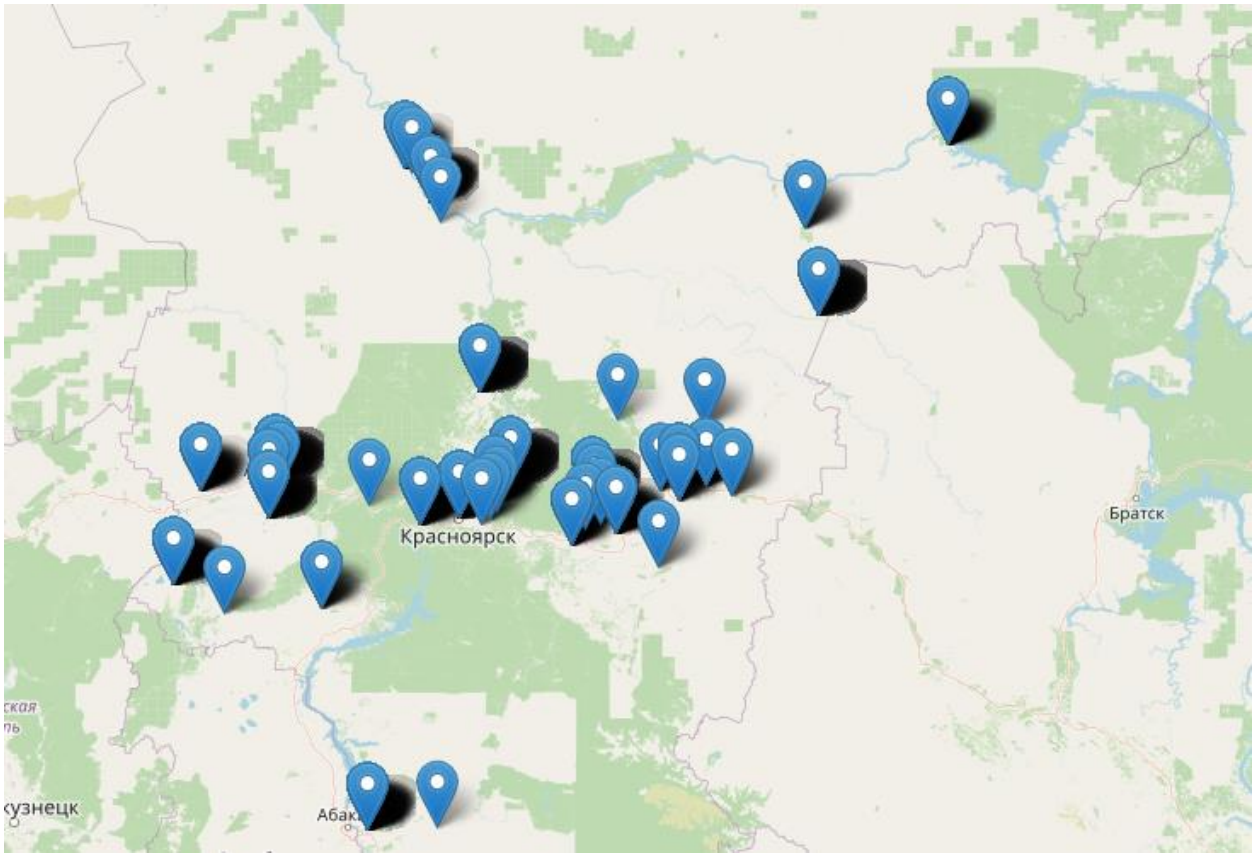


Рисунок 7 – Пример карты библиотеки Folium с маркерами

В данном примере, для показа карты используется открытый проект OpenStreetMap – дословно «открытая карта улиц» – некоммерческий веб-картографический проект по созданию силами сообщества участников – пользователей интернета подробной свободной и бесплатной географической карты мира.

Для создания карт используются:

– данные с персональных GPS-трекеров;

– аэрофотографии;

– видеозаписи;

– спутниковые снимки;

– панорамы улиц, предоставленные некоторыми компаниями;

– знания человека, рисующего карту.

В OpenStreetMap при создании карты используется принцип вики. Каждый зарегистрированный пользователь может вносить изменения в карту.

Проект поддерживается некоммерческой организацией OpenStreetMap Foundation [5].

2.2 Разработка модулей для анализа и обработки данных средствами языка Python в сфере энергетики

Для анализа и обработки данных было принято решение создать web–приложение. Причина выбора – это возможность запуска приложения на различных платформах, а также возможность одновременного просмотра множествами пользователей.

Для создания приложения использовалась среда разработки Visual Studio Code. Visual Studio Code (VS Code) – это кроссплатформенный редактор кода от компании Microsoft, разработанный на базе фреймворка Electron. С его помощью можно разрабатывать кроссплатформенные десктопные приложения, используя веб-технологии.

Выбор среды разработки обусловлен универсальностью и богатым выбором расширений.

На языке Phyton было разработано два отдельных web-приложения: для анализа и обработки и для визуализации на карте в web-браузере.

Основными файлами являются analyses.py и map.py

Каждое web-приложение состоит из модулей. Разработка программного кода модулей начинается с определения функций, которые должно реализовывать web-приложение. Основные функции, который должен реализовывать программный код представлены на рисунке 8.

Рисунок 8 – Схема выполнения кода программы

Данные для обработки и анализа берутся из выгрузки отчета из облачного сервиса «Пирамида–2000» ПАО «Красноярскэнергосбыт» в виде файла Excel. Отчёт представляет собой структурированные данные приборов учёта, типов приборов, адреса, личные счета и т. п.

Структура файла представлена в таблице 1 Приложения Г.

2.2.1 Приложение для обработки, анализа, визуализации данных

Приложение содержит в себе несколько модулей:

– модуль загрузки и обработки датасета из таблицы excel;

– модуль создания таблиц, графиков и диаграмм;

– модуль инициализации фреймворка streamlit.

Чтобы запустить приложение, необходимо ввести команду «streamlit run f:\Py\excel_file_analysis\analysis.py» в терминале и нажать клавишу «Enter»:

На рисунке 9 показан фрагмент окна запуска.

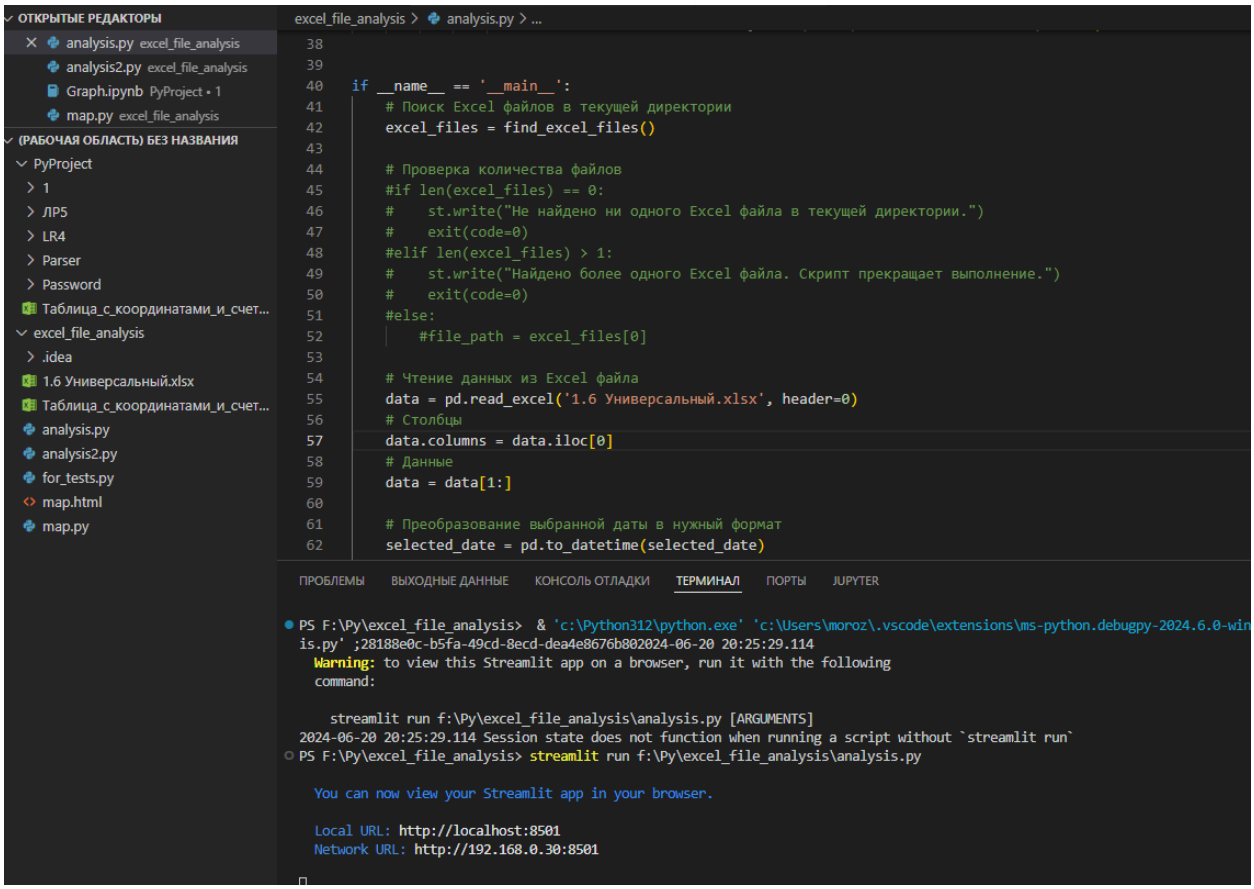


Рисунок 9 – Фрагмент окна запуска приложения

На рисунке 10 показано окно web-браузера со страницей дашборда, открывшееся после запуска команды.

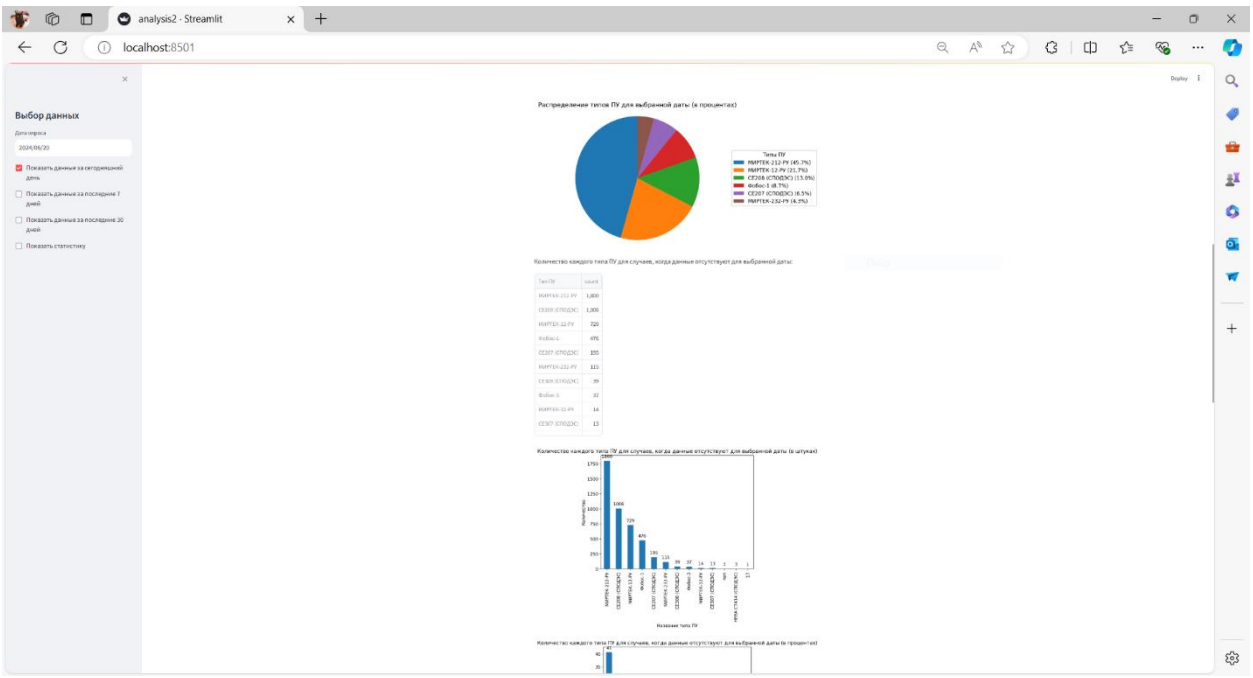


Рисунок 10 – Окно web-браузера с таблицами и диаграммами

2.2.2 Приложение для визуализации данных на карте

Приложение содержит четыре модуля:

– модуль загрузки и обработки датасета из таблицы excel;

– модуль кодирования координат с помощью сервиса геокодирования;

– модуль создания датасета для координат;

– модуль создания HTML файла для отображения в web-браузере.

Запуск программы происходит с помощью командной строки:

f:\Py\excel_file_analysis\python map.py.

После запуска запускается сервис геокодирования и начинает работать скрипт для создания списка адресов для маркеров на карте.

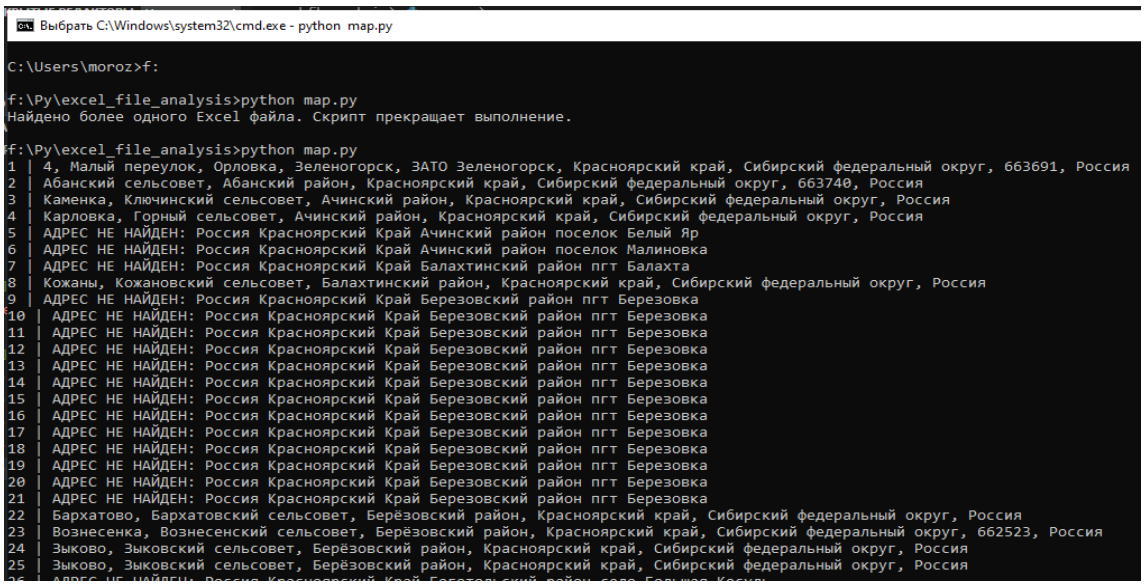


Рисунок 11 – Работа скрипта для геокодирования

После окончания работы скрипта создается HTML файл – map.html, который можно запустить либо в проводнике, либо также, набрав в командной строке map.html.

После этого запустится web-браузер с картой, на которой обозначены маркеры с типами и номерами наших приборов учёта.

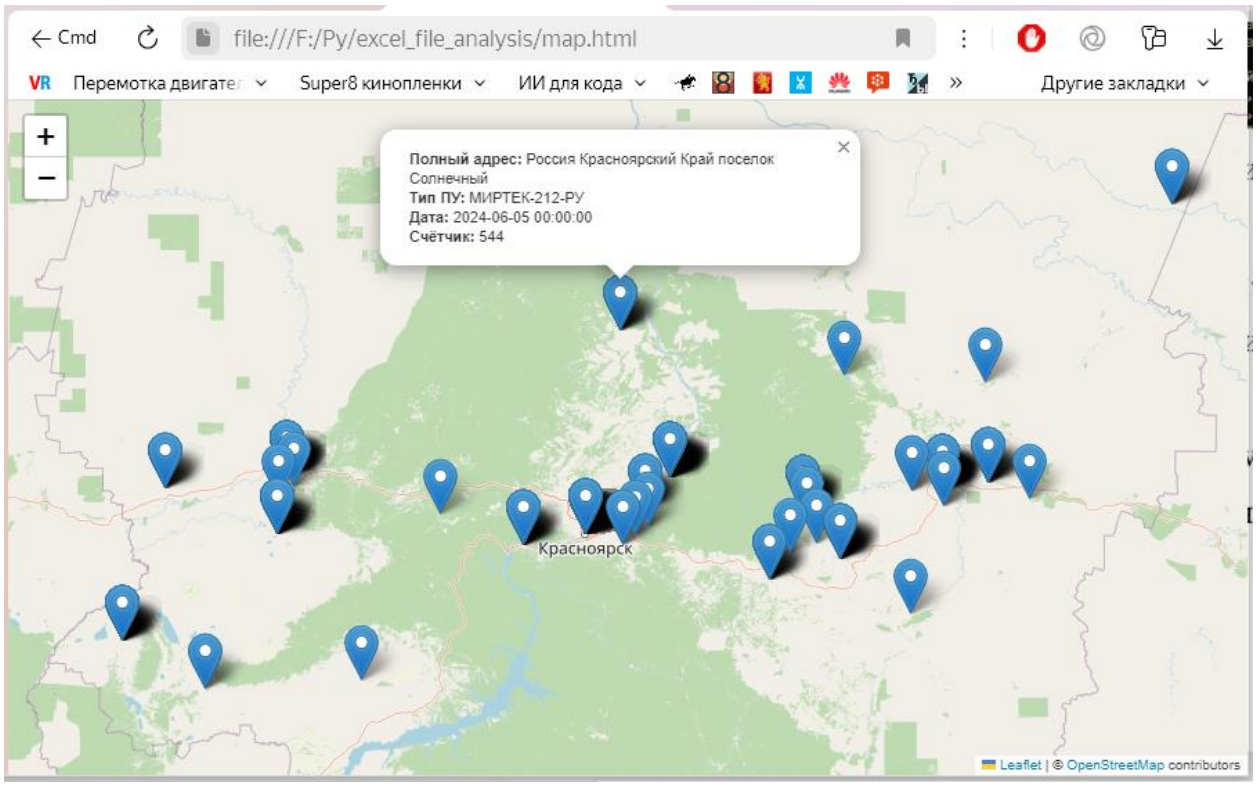


Рисунок 17 – Web-браузер с картой OpenStreetMap и маркерами

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы были реализованы все поставленные задачи, а именно: проведен анализ предметной области, были изучены и проанализированы текущие решения анализа и обработки данных, за счет этого были сформулированы требования к разрабатываемого продукту.

Разработка приложения для анализа и обработки данных в сфере энергетики, средствами языка программирования Phyton, позволила по-новому взглянуть на проблему анализа и обработки данных интеллектуальных приборов учета электроэнергии, а также позволит повысить качество мониторинга потребления электроэнергии в ПАО «Красноярскэнергосбыт» и повысит качество и скорость обслуживания потребителей.

Внедрение программного обеспечения для анализа и обработки данных интеллектуальных систем учета позволит улучшить контроль и учет потребления электроэнергии, оптимизировать работу энергосистемы и снизить потери электроэнергии, а также, частично решить задачу программы по импортозамещению.

Процесс установки интеллектуальных систем учета представляет собой последовательность этапов, начиная от подачи заявки на установку приборов учета до их настройки и программирования. Важным аспектом является привлечение подрядных организаций для выполнения работ по установке интеллектуальных систем учета, что позволяет сконцентрироваться на основной деятельности компании и повысить качество выполняемых работ.

Кроме того, включение работ по установке интеллектуальных систем учета в инвестиционную программу на 2023 год свидетельствует о том, что компания уделяет значительное внимание развитию и модернизации системы учета электроэнергии. Это позволит достичь более высокого уровня

энергоэффективности, обеспечить экономию энергоресурсов и улучшить экологическую ситуацию в регионе.

Проведен анализ проверки показателей качества электроэнергии, поставляемой в многоквартирные дома, с использованием приборов учета, присоединенных к интеллектуальной системе учета электрической энергии.

Таким образом, проведенное исследование на начальном этапе позволяет сделать вывод о том, что реализация разработанного приложения для анализа и обработки данных средствами языка программирования Python, в ходе процесса мониторинга потребления электроэнергии в ПАО «Красноярскэнергосбыт» будет способствовать решению актуальных задач в области энергетики и повышению качества предоставляемых услуг потребителям.

СПИСОК СОКРАЩЕНИЙ

ИСУЭЭ – Интеллектуальная система учета электрической энергии;

АСКУЭ – Автоматизированная система коммерческого учёта электроэнергии;

ИПУ – Интеллектуальный прибор учёта;

ОДПУ – Общедомовой прибор учёта;

ИИК – измерительно – информационный комплекс;

АРМ – Автоматизированные рабочие места

ИВК – Информационно – вычислительный комплекс;

ОРЭМ – Оптовый рынок электроэнергии и мощности;

БД – База данных;

ПО – Программное обеспечение;

GSM – Global System for Mobile Communications (Глобальные Мобильные Системы Связи);

CSD – Circuit Switched Data (Цепь Коммутируемых Данных).

GPRS – General Packet Radio Service (пакетная радиосвязь общего пользования)

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Basic Slicing and Advanced Indexing in NumPy : сайт / GeeksforGeeks A computer science portal for geeks – 2024. – URL: https://www.geeksforgeeks.org/numpy–slicing–and–indexing/ (дата обращения: 20.06.2024).
2. Comparing Python to Other Languages: официальный сайт / python.org – 2024. – URL: https://www.python.org/doc/essays/comparisons/ (дата обращения: 19.06.2024).
3. Comparison of Python with Other Programming Languages: сайт / GeeksforGeeks A computer science portal for geeks – 2024. – URL: https://www.geeksforgeeks.org/comparison–of–python–with–other–programming–languages/ (дата обращения: 19.06.2024).
4. Matplotlib: сайт / Библиотека Matplotlib – 2024. – URL: https://matplotlib.org (дата обращения: 20.06.2024).
5. OpenStreetMap: сайт / РУВИКИ – 2024. – URL: https://ru.ruwiki.ru/wiki/OpenStreetMap (дата обращения: 20.06.2024).
6. Francis Paul : Python vs Other Programming Languages in 2024: Detailed Comparison / P. Francis. uvik.net – 2024. – URL: https://uvik.net/blog/how–python–is–different–from–other–languages/ (дата обращения: 19.06.2024).
7. Python Vs Scala : сайт / Knowledgehut – 2024. – URL: https://www.knowledgehut.com/blog/programming/python–vs–scala–comparison (дата обращения: 20.06.2024).
8. Streamlit. A faster way to build and share data apps: сайт / streamlit.io Официальный сайт – 2024. – URL: https://streamlit.io/ (дата обращения: 20.06.2024).
9. Visualizing Geospatial Data using Folium in Python : сайт / GeeksforGeeks A computer science portal for geeks – 2024. – URL:

https://www.geeksforgeeks.org/visualizing–geospatial–data–using–folium–in–python/ (дата обращения: 20.06.2024).

10. Антипко, А. В. Инструменты для анализа данных: сравнение Python, R и других популярных платформ / А. В. Антипко. – Молодой ученый. – 2023. – № 33 (480). – С. 14–16. – URL: https://moluch.ru/archive/480/105529/ (дата обращения: 18.06.2024).
11. Бурнашев, Р.А. Анализ данных на языке программирования Python: Библиотека Pandas / Р.А. Бурнашев. – Казань: Казан. ун–т, 2022. – 25 с.
12. Гвоздева, В.А. Информатика, автоматизированные информационные технологии и системы. / В.А. Гвоздева. – М.: ИНФРА–М, 2018. – 544 с.
13. Голицына, О. Л. Информационные системы: учебное пособие / О. Л. Голицына, Н. В. Максимов, И. И. Попов. – 2–е изд. – М.: ФОРУМ ИНФРА–М, 2016. – 448 с.
14. Гончаров, А. Самоучитель HTML / А. Гончаров. – Санкт–Петербург: Питер, 2018. – 240 с.
15. ГОСТ 13109–97. Электрическая энергия. Совместимость технических средств электромагнитная. Нормы качества электрической энергии в электрических сетях общего назначения: Принят межгосударственным советом по стандартизации, метрологии и сертификации (протокол № 12—97 от 21 ноября 1997 г.): дата введения 01.01.1999 года. – Текст: электронный // Электронный фонд правовых и нормативно-технических документов. – URL: https://docs.cntd.ru/document/1200006034?ysclid=lxvieiovow315103243 (дата обращения 29.05.2024).
16. ГОСТ 32144–2013. Электрическая энергия. Совместимость технических средств электромагнитная. Нормы качества электрической энергии в системах электроснабжения общего назначения: Принят Межгосударственным советом по стандартизации, метрологии и сертификации (протокол № 55-П от 25 марта 2013 г.): дата введения

01.07.2014. – Текст: электронный // Электронный фонд правовых и нормативно-технических документов. – URL: https://docs.cntd.ru/document/1200006034?ysclid=lxvhzp7pjo750217990 (дата обращения 29.05.2024).

17. ГОСТ Р 58940-2020. Системы учета электрической энергии. Приборы учета. Общие технические требования: Утверждён и введен в действие приказом Федерального агентства по техническому регулированию и метрологии от 28 июля 2020 г. № 415-ст): Дата введения – 01.01.2021 г. – Текст: электронный // Электронный фонд правовых и нормативно-технических документов. – URL: https://docs.cntd.ru/document/1200174430 (дата обращения 29.05.2024).
18. Евлоев, А.З. Основы управления бизнес–процессами в организации / Евлоев А.З. Инновации и инвестиции. – 2017. – № 9. – С. 64–66.
19. Ефимова, М. 8 самых популярных языков программирования для работы с Big Data / М. Ефимова, Л. Шпрингер. Яндекс Образование – 2022. – URL: https://practicum.yandex.ru/blog/yazyki–programmirovaniya–dlya–big–data/ (дата обращения: 19.06.2024).
20. Кириченко, А. Web на практике. CSS, HTML, JavaScript, MySQL, PHP для fullstack–разработчиков. /А. Кириченко. – СПб.: Наука и Техника, 2021. – 432 с.
21. Копытова, М. А. Актуальность языка программирования Python / М. А. Копытова // Экономика и социум. – 2016. – № 10 (29). – С. 943–945. – URL: https://cyberleninka.ru/article/n/aktualnost–yazyka–programmirovaniya–python (дата обращения: 18.06.2024).
22. Красочкин, С. Г. Изображения и визуализация данных в Python / С. Г. Красочкин // Научный журнал. – 2022. – № 2 (64). – URL: https://cyberleninka.ru/article/n/izobrazheniya–i–vizualizatsiya–dannyh–v–python (дата обращения: 18.06.2024).

23. Куликов, С. С. Тестирование программного обеспечения. Базовый курс / С. С. Куликов. – Беларусь: Минск, 2020. – 314с.
24. МакГрат, М. JavaScript для начинающих / М. МакГрат. – М.: Эксмо, 2022. – 232 с.
25. Маккинни, Уэс Python и анализ данных: Первичная обработка данных с применением pandas, NumPy и Jupiter / пер. с англ. А.А. Слинкина. 3–е изд. – М.: МК Пресс, 2023. – 536с.: ил.
26. Мезенцев, К. Н. Автоматизированные информационные системы. / К.Н. Мезенцев. – М.: ИЦ Академия, 2018. – 176 с.
27. Мокеев, В. В. Бизнес–информатика / В. В. Мокеев, Е. В. Бунова, О. С. Буслаева. – Челябинск: издательский центр ЮУрГУ, 2018. – 67 с.
28. Морозов, А. В. Анализ и визуализация данных средствами языка PYTHON / А.В. Морозов // III Всероссийский молодежный научный форум «Современное педагогическое образование: теоретический и прикладной аспекты», ЛПИ – филиал СФУ – Лесосибирск, 2024.
29. Морозова, П. Преимущества и недостатки Python / П. Морозова // Онлайн школа Skysmart: [сайт]. – 2024. – URL: https://skysmart.ru/articles/programming/preimushestva-i-nedostatki-python (дата обращения: 17.06.2024).
30. О порядке предоставления доступа к минимальному набору функций интеллектуальных систем учета электрической энергии (мощности): Постановление Правительства РФ от 19 июня 2020 г. № 890. // Официальный интернет-портал правовой информации. – URL: https://base.garant.ru/74292774/ (дата обращения: 17.06.2024).
31. О предоставлении коммунальных услуг собственникам и пользователям помещений в многоквартирных домах и жилых домов (вместе с «Правилами предоставления коммунальных услуг собственникам и пользователям помещений в многоквартирных домах и жилых домов»): Постановление Правительства РФ от 06.05.2011 № 354. // Официальный

интернет-портал правовой информации. – URL: https://base.garant.ru/12186043/?ysclid=lxvikss9l3473423917/ (дата обращения: 17.06.2024).

32. О функционировании розничных рынков электрической энергии, полном и (или) частичном ограничении режима потребления электрической энергии: Постановление Правительства РФ от 04.05.2012 № 442 (ред. от 03.05.2024). // Официальный интернет-портал правовой информации. – URL: https://base.garant.ru/70183216/?ysclid=lxvin51ur3742704209/ (дата обращения: 17.06.2024).
33. Попов, А. А. Эргономика пользовательских интерфейсов в информационных системах / А. А. Попов. – М.: Кнорус, 2023. – 304 с.
34. Почему Go актуален в современной разработке и зачем его изучать : сайт. / Яндекс Практикум : – 2024. – URL: https://practicum.yandex.ru/blog/zachem–uchit–yazyk–go/ (дата обращения: 20.06.2024).
35. Публичное акционерное общество «Красноярскэнергосбыт»: официальный сайт. – 2024. – URL: https://www.krsk–sbit.ru/ (дата обращения 10.06.2024).
36. Рейтинг самых используемых языков программирования за 2023 год по версии крупнейшей социальной сети для разработчиков: сайт / Github. – 2024. – URL: https://github.blog/2023–11–08–the–state–of–open–source–and–ai/ (дата обращения: 11.06.2024).
37. Российская Федерация. Законы. О внесении изменений в отдельные законодательные акты Российской Федерации в связи с развитием систем учета электрической энергии (мощности) Федеральный закон: № 522-ФЗ: [Принят Государственной Думой 27 декабря 2018 года] // Официальный интернет-портал правовой информации. – URL: http://publication.pravo.gov.ru/Document/View/0001201812280018?ysclid=lxvgkp4baq447741315

38. Российская Федерация. Законы. Об электроэнергетике Федеральный закон №35-ФЗ: [Принят Государственной Думой 26 марта 2003 года] // Официальный интернет-портал правовой информации. – URL: http://pravo.gov.ru/proxy/ips/?docbody=&nd=102080839&ysclid=lxvgn4h89w527849004
39. Российская Федерация. Законы. Об энергосбережении и о повышении энергоэффективности в РФ № 261-ФЗ: [Принят Государственной Думой 23 ноября 2009] // Официальный интернет-портал правовой информации. – URL: http://pravo.gov.ru/proxy/ips/?docbody=&nd=102133970&ysclid=lxvgo59ii2786849530
40. Рындина, Светлана Валентиновна. Базовые возможности языка Python для анализа данных :учеб.–метод. пособие / С. В. Рындина. – Пенза : Изд–во ПГУ,2022. – 72 с.
41. Симпсон, К. Вы пока еще не знаете JS. Познакомьтесь, JavaScript. – СПб.: Питер, 2022. – 192 с.
42. Энгхейм, Э. Julia в качестве второго языка / пер. с англ. А. В. Логунова. – М.: ДМК Пресс, 2023. – 446 с.: ил.
43. Язык программирования Julia // The Julia Programming Language: [сайт]. – 2024. – URL: https://julialang.org/ (дата обращения: 19.06.2024).
44. Язык программирования Python // Python: [сайт]. – 2024. – URL: https://www.python.org/ (Дата обращения: 17.06.2024).

ПРИЛОЖЕНИЕ А

Листинг кода приложения analysis.py

```
import glob
import pandas as pd
import streamlit as st
import matplotlib.pyplot as plt

# Создание интерфейса для выбора данных
st.sidebar.title("Выбор данных")
selected_date = st.sidebar.date_input("Дата опроса")
# Добавление кнопки для вычисления данных за сегодняшний день
show_current_day = st.sidebar.checkbox("Показать данные за сегодняшний день")
# Добавление кнопки для вычисления данных за последние 7 дней
show_last_7_days = st.sidebar.checkbox("Показать данные за последние 7 дней")
# Добавление кнопки для вычисления данных за последние 30 дней
show_last_30_days = st.sidebar.checkbox("Показать данные за последние 30 дней")
# Добавление диапазона дат для выбора
show_date_range = st.sidebar.checkbox("Показать данные за выбранный период")
if show_date_range:
    start_date = st.sidebar.date_input("Начальная дата", selected_date)
    end_date = st.sidebar.date_input("Конечная дата", selected_date)
# Добавление кнопки для отображения статистики
show_statistics = st.sidebar.checkbox("Показать статистику")
# Создание интерфейса для выбора типа диаграммы
chart_type = st.sidebar.selectbox(
    "Выберите тип диаграммы",
    ["Столбчатая диаграмма", "Круговая диаграмма", "Линейная диаграмма"]
)
# Функция для поиска Excel файлов в директории
def find_excel_files() -> list[str]:
    excel_files = glob.glob("*.xlsx")
    return excel_files
```

```
# Функция для подписи значений столбцов диаграммы
def plot_with_labels(ax, data, title, xlabel, ylabel) -> None:
    data.plot(kind='bar', ax=ax)
    ax.set_title(title)
    ax.set_xlabel(xlabel)
    ax.set_ylabel(ylabel)
    for p in ax.patches:
        height = p.get_height()
        ax.annotate(f'{height:.0f}', (p.get_x() + p.get_width() / 2., height),
                    ha='center', va='center', xytext=(0, 10), textcoords='offset points')
# Функция для создания круговой диаграммы с подписями
def plot_pie_chart(data, title) -> None:
    fig, ax = plt.subplots()
    wedges, texts = ax.pie(data, labels=None, startangle=90, 
textprops=dict(color="w"))
    # Создание меток с процентами
    percentages = [f'{data.index[i]} ({round(data[i], 1)}%)' for i in range(len(data))]
    # Настройка меток и подписи клиньев
    ax.legend(wedges, percentages, title="Типы ПУ", loc="center left", 
bbox_to_anchor=(1, 0, 0.5, 1))
    ax.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
    plt.title(title)
    st.pyplot(fig)
# Функция для создания линейной диаграммы
def plot_line_chart(data, title, xlabel, ylabel) -> None:
    fig, ax = plt.subplots()
    data.plot(kind='line', ax=ax)
    ax.set_title(title)
    ax.set_xlabel(xlabel)
    ax.set_ylabel(ylabel)
    plt.xticks(rotation=90)  # Устанавливаем угол наклона меток на оси x
    st.pyplot(fig)
if __name__ == '__main__':
    # Поиск Excel файлов в текущей директории
    excel_files = find_excel_files()
    # Проверка количества файлов
    if len(excel_files) == 0:
        st.write("Не найдено ни одного Excel файла в текущей директории.")
        exit(code=0)
    elif len(excel_files) > 1:
```

```
        st.write("Найдено более одного Excel файла. Скрипт прекращает выполнение.")
        exit(code=0)
    else:
        file_path = excel_files[0]
    # Чтение данных из Excel файла
    data = pd.read_excel(file_path, header=0)
    # Столбцы
    data.columns = data.iloc[0]
    # Данные
    data = data[5:]
    # Преобразование выбранной даты в нужный формат
    selected_date = pd.to_datetime(selected_date)
    # Инициализация переменных
    # Получение текущей даты
    current_date = pd.to_datetime('today')
    # Вычисление даты начала (за последние 7 дней)
    start_date_7_days = current_date - pd.Timedelta(days=7)
    # Вычисление даты начала (за последние 30 дней)
    start_date_30_days = current_date - pd.Timedelta(days=30)
    # ФИЛЬТРЫ
        # Фильтрация данных на основе выбранной даты
    filtered_data_by_date = data[data['Дата'] == selected_date]
    # Фильтрация данных для случаев, когда данные отсутствуют для выбранной даты
    filtered_data_no_selected_date = data[data['Дата'] != selected_date]
    # Фильтрация данных на основе выбранной даты
    filtered_data_current_date = data[data['Дата'] == current_date]
    # Фильтрация данных для случаев, когда данные отсутствуют для выбранной даты
    filtered_data_no_current_date = data[data['Дата'] != current_date]
    # Фильтрация данных за последние 7 дней
    filtered_data_last_7_days = data[(data['Дата'] >= start_date_7_days) & (data['Дата'] <= current_date)]
    # Фильтрация данных для случаев, когда данные отсутствуют за последние 30 дней
    filtered_data_no_last_7_days = data[data['Дата'] < start_date_30_days]
    # Фильтрация данных за последние 30 дней
    filtered_data_last_30_days = data[(data['Дата'] >= start_date_30_days) & (data['Дата'] <= current_date)]
```

```
    # Фильтрация данных для случаев, когда данные отсутствуют за последние 30 дней
    filtered_data_no_last_30_days = data[data['Дата'] < start_date_30_days]
    # Фильтрация данных на основе выбранного диапазона дат
    if show_date_range:
        start_date = pd.to_datetime(start_date)
        end_date = pd.to_datetime(end_date)
        filtered_data_date_range = data[(data['Дата'] >= start_date) & (data['Дата'] <= end_date)]
        filtered_data_no_date_range = data[(data['Дата'] < start_date) | (data['Дата'] > end_date)]
    pu_counts_selected_date = filtered_data_by_date['Тип ПУ'].astype(str).value_counts()
    pu_counts_by_selected_date_percent = pu_counts_selected_date / pu_counts_selected_date.sum() * 100
    pu_counts_no_selected_date = filtered_data_no_selected_date['Тип ПУ'].astype(str).value_counts()
    pu_counts_no_selected_date_percent = pu_counts_no_selected_date / pu_counts_no_selected_date.sum() * 100
    pu_counts_current_date = filtered_data_current_date['Тип ПУ'].astype(str).value_counts()
    pu_counts_current_date_percent = pu_counts_current_date / pu_counts_current_date.sum() * 100
    pu_counts_no_current_date = filtered_data_no_current_date['Тип ПУ'].astype(str).value_counts()
    pu_counts_no_current_date_percent = pu_counts_no_current_date / pu_counts_no_current_date.sum() * 100
    pu_counts_last_7_days = filtered_data_last_7_days.groupby('Тип ПУ').size()
    pu_counts_last_7_days_percent = pu_counts_last_7_days / pu_counts_last_7_days.sum() * 100
    pu_counts_no_last_7_days = filtered_data_no_last_7_days['Тип ПУ'].astype(str).value_counts()
    pu_counts_no_last_7_days_percent = pu_counts_no_last_7_days / pu_counts_no_last_7_days.sum() * 100
    pu_counts_last_30_days = filtered_data_last_30_days.groupby('Тип ПУ').size()
    pu_counts_last_30_days_percent = pu_counts_last_30_days / pu_counts_last_30_days.sum() * 100
    pu_counts_no_last_30_days = filtered_data_no_last_30_days['Тип ПУ'].astype(str).value_counts()
    pu_counts_no_last_30_days_percent = pu_counts_no_last_30_days /
```

```
pu_counts_no_last_30_days.sum() * 100
    if show_date_range:
        pu_counts_date_range = filtered_data_date_range['Тип ПУ'].astype(str).value_counts()
        pu_counts_date_range_percent = pu_counts_date_range / pu_counts_date_range.sum() * 100
        pu_counts_no_data_range = filtered_data_no_date_range['Тип ПУ'].astype(str).value_counts()
        pu_counts_date_no_range_percent = pu_counts_no_data_range / pu_counts_no_data_range.sum() * 100
    if show_statistics:
        # Объединение всех индексов в один
        all_indices = pu_counts_current_date.index.union(pu_counts_last_7_days.index).union(
            pu_counts_last_30_days.index).union(pu_counts_no_last_30_days.index)
        # Создание таблицы статистики
        statistics_df = pd.DataFrame({
            'Тип ПУ': all_indices,
            'Опрошено на начало суток, шт': pu_counts_current_date.reindex(all_indices, fill_value=0),
            'Опрошено за последние 7 дней, шт': pu_counts_last_7_days.reindex(all_indices, fill_value=0),
            'Опрошено за последние 30 дней, шт': pu_counts_last_30_days.reindex(all_indices, fill_value=0),
            'Не опросились за последние 30 суток': pu_counts_no_last_30_days.reindex(all_indices, fill_value=0),
            'Опрос на начала суток, %': pu_counts_current_date_percent.reindex(all_indices, fill_value=0),
            'Опрос за 7 суток, %': pu_counts_last_7_days_percent.reindex(all_indices, fill_value=0),
            'Опрос за 30 суток, %': pu_counts_last_30_days_percent.reindex(all_indices, fill_value=0)
        })
        st.markdown("## Статистика:")
        st.write(statistics_df)
    else:
        st.markdown("## Дата для анализа: ")
        # HTML и CSS для увеличения шрифта до уровня заголовка ## (обычно это 24px)
        st.markdown(
```

```
        """
        <style>
        .big-font {
            font-size:24px !important;
            font-weight: bold;
        }
        </style>
        """,
        unsafe_allow_html=True,
    )
    # Вывод даты с увеличенным шрифтом
    st.markdown(f'<p class="big-font">{selected_date}</p>', unsafe_allow_html=True)
    st.markdown("## Количество каждого типа ПУ для выбранной даты:")
    # Подсчет количества каждого типа ПУ для выбранной даты
    if not filtered_data_by_date.empty:
        # Вывод результатов
        st.write(pu_counts_selected_date)
        if chart_type == 'Столбчатая диаграмма':
            # Создание столбчатой диаграммы в штуках
            fig, ax = plt.subplots()
            plot_with_labels(ax, pu_counts_selected_date, 'Количество каждого типа ПУ для выбранной даты (в штуках)', 'Название типа ПУ', 'Количество')
            st.pyplot(fig)
            # Создание столбчатой диаграммы в процентах
            fig, ax = plt.subplots()
            plot_with_labels(ax, pu_counts_by_selected_date_percent, 'Количество каждого типа ПУ для выбранной даты (в процентах)', 'Название типа ПУ', 'Проценты')
            st.pyplot(fig)
        if chart_type == 'Круговая диаграмма':
            # Создание круговой диаграммы в процентах
            plot_pie_chart(pu_counts_by_selected_date_percent, 'Распределение типов ПУ для выбранной даты (в процентах)')
    else:
        st.markdown("## Для выбранной даты отсутствуют данные.")
    # Подсчет количества каждого типа ПУ для случаев, когда данные отсутствуют для выбранной даты
    if not filtered_data_no_selected_date.empty:
        # Вывод результатов
```

```
        st.write("## Количество каждого типа ПУ для случаев, когда данные отсутствуют для выбранной даты:")
        st.write(pu_counts_no_selected_date)
        if chart_type == 'Столбчатая диаграмма':
            # Создание столбчатой диаграммы в штуках
            fig, ax = plt.subplots()
            plot_with_labels(ax, pu_counts_no_selected_date, 'Количество каждого типа ПУ для случаев, когда данные отсутствуют для выбранной даты (в штуках)', 'Название типа ПУ', 'Количество')
            st.pyplot(fig)
            # Создание столбчатой диаграммы в процентах
            fig, ax = plt.subplots()
            plot_with_labels(ax, pu_counts_no_selected_date_percent, 'Количество каждого типа ПУ для случаев, когда данные отсутствуют для выбранной даты (в процентах)', 'Название типа ПУ', 'Проценты')
            st.pyplot(fig)
        if chart_type == 'Круговая диаграмма':
            # Создание круговой диаграммы в процентах
            plot_pie_chart(pu_counts_no_selected_date_percent, 'Распределение типов ПУ для случаев, когда данные отсутствуют для выбранной даты (в процентах)')
    else:
        st.markdown("## Данных, не совпадающих с выбранной датой, отсутствуют.")
    if show_current_day:
        st.markdown("## Количество каждого типа ПУ для сегодняшней даты:")
        # Подсчет количества каждого типа ПУ для выбранной даты
        if not filtered_data_current_date.empty:
            # Вывод результатов
            st.write(pu_counts_current_date)

            if chart_type == 'Столбчатая диаграмма':
                # Создание столбчатой диаграммы в штуках
                fig, ax = plt.subplots()
                plot_with_labels(ax, pu_counts_current_date, 'Количество каждого типа ПУ для сегодняшней даты (в штуках)',
                                 'Название типа ПУ', 'Количество')
                st.pyplot(fig)
```

```
            # Создание столбчатой диаграммы в процентах
            fig, ax = plt.subplots()
            plot_with_labels(ax, pu_counts_current_date_percent,
                             'Количество каждого типа ПУ для сегодняшней даты (в процентах)', 'Название типа ПУ',
                             'Проценты')
            st.pyplot(fig)

        if chart_type == 'Круговая диаграмма':
            # Создание круговой диаграммы в процентах
            plot_pie_chart(pu_counts_current_date_percent, 'Распределение типов ПУ для сегодняшней даты (в процентах)')

    else:
        st.markdown("## Для сегодняшней даты отсутствуют данные.")

    # Подсчет количества каждого типа ПУ для случаев, когда данные отсутствуют для выбранной даты
    if not filtered_data_no_current_date.empty:
        # Вывод результатов
        st.markdown("## Количество каждого типа ПУ для случаев, когда данные отсутствуют для сегодняшней даты:")
        st.write(pu_counts_no_current_date)

        if chart_type == 'Столбчатая диаграмма':
            # Создание столбчатой диаграммы в штуках
            fig, ax = plt.subplots()
            plot_with_labels(ax, pu_counts_no_current_date,
                             'Количество каждого типа ПУ для случаев, когда данные отсутствуют для сегодняшней даты (в штуках)',
                             'Название типа ПУ', 'Количество')
            st.pyplot(fig)

            # Создание столбчатой диаграммы в процентах
            fig, ax = plt.subplots()
            plot_with_labels(ax, pu_counts_no_current_date_percent,
                             'Количество каждого типа ПУ для случаев, когда данные отсутствуют для сегодняшней даты (в процентах)',
                             'Название типа ПУ', 'Проценты')
            st.pyplot(fig)
```

```
            if chart_type == 'Круговая диаграмма':
                # Создание круговой диаграммы в процентах
                plot_pie_chart(pu_counts_no_current_date_percent,
                               'Распределение типов ПУ для случаев, когда данные отсутствуют для сегодняшней даты (в процентах)')

            else:
                st.markdown("## Данных, не совпадающих с сегодняшней датой, отсутствуют.")
        if show_last_7_days:
            st.markdown("## Количество каждого типа ПУ за последние 7 дней:")
            if not filtered_data_last_7_days.empty:
                # Вывод результатов
                st.write(pu_counts_last_7_days)

                if chart_type == 'Столбчатая диаграмма':
                    # Создание столбчатой диаграммы в штуках
                    fig, ax = plt.subplots()
                    plot_with_labels(ax, pu_counts_last_7_days, 'Количество каждого типа ПУ за последние 7 дней (в штуках)',
                                     'Название типа ПУ', 'Количество')
                    st.pyplot(fig)

                if chart_type == 'Линейная диаграмма':
                    # Создание линейной диаграммы в штуках
                    plot_line_chart(pu_counts_last_7_days, 'Количество каждого типа ПУ за последние 7 дней (в штуках)',
                                    'Название типа ПУ', 'Количество')

                if chart_type == 'Столбчатая диаграмма':
                    # Создание столбчатой диаграммы в процентах
                    fig, ax = plt.subplots()
                    plot_with_labels(ax, pu_counts_last_7_days_percent,
                                     'Количество каждого типа ПУ за последние 7 дней (в процентах)', 'Название типа ПУ',
                                     'Проценты')
                    st.pyplot(fig)

                if chart_type == 'Круговая диаграмма':
```

```
            # Создание круговой диаграммы в процентах
            plot_pie_chart(pu_counts_last_7_days_percent,
                           'Распределение типов ПУ за последние 7 дней (в процентах)')

        else:
            st.markdown("## За последние 7 дней отсутствуют данные.")

        # Подсчет количества каждого типа ПУ для случаев, когда данные отсутствуют за последние 30 дней
        if not filtered_data_no_last_7_days.empty:
            # Вывод результатов
            st.markdown("## Количество каждого типа ПУ для случаев, когда данные отсутствуют за последние 7 дней:")
            st.write(pu_counts_no_last_7_days)

            if chart_type == 'Столбчатая диаграмма':
                # Создание столбчатой диаграммы в штуках
                fig, ax = plt.subplots()
                plot_with_labels(ax, pu_counts_no_last_7_days,
                                 'Количество каждого типа ПУ для случаев, когда данные отсутствуют за последние 7 дней (в штуках)',
                                 'Название типа ПУ', 'Количество')
                st.pyplot(fig)

            if chart_type == 'Линейная диаграмма':
                # Создание линейной диаграммы в штуках
                plot_line_chart(pu_counts_no_last_7_days, 'Количество каждого типа ПУ за последние 7 дней (в штуках)',
                                'Название типа ПУ', 'Количество')

            if chart_type == 'Столбчатая диаграмма':
                # Создание столбчатой диаграммы в процентах
                fig, ax = plt.subplots()
                plot_with_labels(ax, pu_counts_no_last_7_days_percent,
                                 'Количество каждого типа ПУ для случаев, когда данные отсутствуют за последние 7 дней (в процентах)',
                                 'Название типа ПУ', 'Проценты')
                st.pyplot(fig)
```

```
            if chart_type == 'Круговая диаграмма':
                # Создание круговой диаграммы в процентах
                plot_pie_chart(pu_counts_no_last_7_days_percent,
                               'Распределение типов ПУ для последних 7 дней (в процентах)')

        else:
            st.markdown("## Данных, отсутствующих за последние 7 дней, нет.")
    if show_last_30_days:
        st.markdown("## Количество каждого типа ПУ за последние 30 дней:")
        if not filtered_data_last_30_days.empty:
            # Вывод результатов
            st.write(pu_counts_last_30_days)

            if chart_type == 'Столбчатая диаграмма':
                # Создание столбчатой диаграммы в штуках
                fig, ax = plt.subplots()
                plot_with_labels(ax, pu_counts_last_30_days, 'Количество каждого типа ПУ за последние 30 дней (в штуках)','Название типа ПУ', 'Количество')
                st.pyplot(fig)

            if chart_type == 'Линейная диаграмма':
                # Создание линейной диаграммы в штуках
                plot_line_chart(pu_counts_last_30_days, 'Количество каждого типа ПУ за последние 30 дней (в штуках)','Название типа ПУ', 'Количество')

            if chart_type == 'Столбчатая диаграмма':
                # Создание столбчатой диаграммы в процентах
                fig, ax = plt.subplots()
                plot_with_labels(ax, pu_counts_last_30_days_percent,
                                 'Количество каждого типа ПУ за последние 30 дней (в процентах)', 'Название типа ПУ',
                                 'Проценты')
                st.pyplot(fig)

            if chart_type == 'Круговая диаграмма':
                # Создание круговой диаграммы в процентах
                plot_pie_chart(pu_counts_last_30_days_percent,
                               'Распределение типов ПУ за последние 30 дней (в
```

```
процентах)')
        else:
            st.markdown("## За последние 30 дней отсутствуют данные.")

        # Подсчет количества каждого типа ПУ для случаев, когда данные отсутствуют за последние 30 дней
        if not filtered_data_no_last_30_days.empty:
            # Вывод результатов
            st.markdown("## Количество каждого типа ПУ для случаев, когда данные отсутствуют за последние 30 дней:")
            st.write(pu_counts_no_last_30_days)

            if chart_type == 'Столбчатая диаграмма':
                # Создание столбчатой диаграммы в штуках
                fig, ax = plt.subplots()
                plot_with_labels(ax, pu_counts_no_last_30_days,
                                 'Количество каждого типа ПУ для случаев, когда данные отсутствуют за последние 30 дней (в штуках)',
                                 'Название типа ПУ', 'Количество')
                st.pyplot(fig)

            if chart_type == 'Линейная диаграмма':
                # Создание линейной диаграммы в штуках
                plot_line_chart(pu_counts_no_last_30_days, 'Количество каждого типа ПУ за последние 30 дней (в штуках)',
                                'Название типа ПУ', 'Количество')

            if chart_type == 'Столбчатая диаграмма':
                # Создание столбчатой диаграммы в процентах
                fig, ax = plt.subplots()
                plot_with_labels(ax, pu_counts_no_last_30_days_percent,
                                 'Количество каждого типа ПУ для случаев, когда данные отсутствуют за последние 30 дней (в процентах)',
                                 'Название типа ПУ', 'Проценты')
                st.pyplot(fig)

            if chart_type == 'Круговая диаграмма':
                # Создание круговой диаграммы в процентах
                plot_pie_chart(pu_counts_no_last_30_days_percent,
                               'Распределение типов ПУ для последних 30 дней (в
```

```
процентах)')
        else:
            st.markdown("## Данных, отсутствующих за последние 30 дней, нет.")

    # Показ данных за выбранный период
    if show_date_range:
        st.markdown("## Количество каждого типа ПУ за выбранный период:")
        if not filtered_data_date_range.empty:              # Вывод результатов
            st.write(pu_counts_date_range)

            if chart_type == 'Столбчатая диаграмма':
                # Создание столбчатой диаграммы в штуках
                fig, ax = plt.subplots()
                plot_with_labels(ax, pu_counts_date_range, 'Количество каждого типа ПУ за выбранный период', 'Название типа ПУ', 'Количество')
                st.pyplot(fig)

            if chart_type == 'Линейная диаграмма':
                # Создание линейной диаграммы в штуках
                plot_line_chart(pu_counts_date_range, 'Количество каждого типа ПУ за выбранный период (в штуках)',
                                'Название типа ПУ', 'Количество')

            if chart_type == 'Столбчатая диаграмма':
                # Создание столбчатой диаграммы в процентах
                fig, ax = plt.subplots()
                plot_with_labels(ax, pu_counts_date_range_percent,
                                 'Количество каждого типа ПУ для случаев за выбранный период (в процентах)',
                                 'Название типа ПУ', 'Проценты')
                st.pyplot(fig)

            if chart_type == 'Круговая диаграмма':
                # Круговая диаграмма процентного соотношения ПУ за выбранный период
                plot_pie_chart(pu_counts_date_range_percent, 'Процентное соотношение ПУ за выбранный период')
        else:
            st.markdown("## За выбранный период отсутствуют данные.")
```

```
        if not filtered_data_no_date_range.empty:
            # Вывод результатов
            st.markdown("## Количество каждого типа ПУ для случаев, когда данные отсутствуют за выбранный период:")
            st.write(pu_counts_no_data_range)

            if chart_type == 'Столбчатая диаграмма':
                # Создание столбчатой диаграммы в штуках
                fig, ax = plt.subplots()
                plot_with_labels(ax, pu_counts_no_data_range, 'Количество каждого типа ПУ за выбранный период', 'Название типа ПУ', 'Количество')
                st.pyplot(fig)

            if chart_type == 'Линейная диаграмма':
                # Создание линейной диаграммы в штуках
                plot_line_chart(pu_counts_no_data_range, 'Количество каждого типа ПУ за выбранный период (в штуках)',
                                'Название типа ПУ', 'Количество')

            if chart_type == 'Столбчатая диаграмма':
                # Создание столбчатой диаграммы в процентах
                fig, ax = plt.subplots()
                plot_with_labels(ax, pu_counts_date_no_range_percent,
                                 'Количество каждого типа ПУ для случаев за выбранный период (в процентах)',
                                 'Название типа ПУ', 'Проценты')
                st.pyplot(fig)

            if chart_type == 'Круговая диаграмма':
                # Круговая диаграмма процентного соотношения ПУ за выбранный период
                plot_pie_chart(pu_counts_date_no_range_percent, 'Процентное соотношение ПУ за выбранный период')
        else: st.markdown("## Данных, отсутствующих за выбранный период, нет.")
```

ПРИЛОЖЕНИЕ Б

Листинг кода приложения map.py

```
import pandas as pd
import folium
import glob
import os
from geopy.geocoders import Nominatim
from typing import Tuple, Union

GEOLOCATOR = Nominatim(user_agent="Tester")
COUNTER = 0

def find_excel_files() -> list[str]:
    excel_files = glob.glob("*.xlsx")
    return excel_files
def process_address(row) -> str:
    district = row.get('Район', '')
    locality = row['Населенный пункт']
    # street = row['Улица']
    # Убираем "р-н" и заменяем на "район"
    if district:
        if type(district) is str:
            if 'р-н' in district:
                district = district.replace('р-н', 'район').strip()
                # street = str(street).replace('ул', '') + ' улица'
                address = f"Россия Красноярский Край {district} {locality}"
                return address
        else:
            address = f"Россия Красноярский Край {locality}"
            return address
def get_coordinates(address: str) -> Union[Tuple[None, None, int], Tuple[float,
float, int]]:
    global COUNTER
    global GEOLOCATOR
    try:
        COUNTER += 1
        location = GEOLOCATOR.geocode(address)
```

```
        if location is not None:
            print(f'{COUNTER} | {location}')
        else:
            print(f'{COUNTER} | АДРЕС НЕ НАЙДЕН: {address}')
        if location:
            return location.latitude, location.longitude, COUNTER
        else:
            return None, None, COUNTER
    except Exception as e:
        print(f"Error getting coordinates for address '{address}': {e}")
        return None, None, COUNTER
def create_map(df: pd.DataFrame) -> None:
    # Средние координаты для начального положения карты
    avg_lat = df['Широта'].mean()
    avg_lon = df['Долгота'].mean()
    m = folium.Map(location=[avg_lat, avg_lon], zoom_start=6)
    for idx, row in df.iterrows():
        popup_html = f"<b>Полный адрес:</b> {row['Полный адрес']}<br>" \
                     f"<b>Тип ПУ:</b> {row['Тип ПУ']}<br>" \
                     f"<b>Дата:</b> {row['Дата']}<br>" \
                     f"<b>Счётчик:</b> {row['Серийный номер ПУ']}"
        folium.Marker(
            location=[row['Широта'], row['Долгота']],
            popup=folium.Popup(popup_html, max_width=300),
        ).add_to(m)
    current_path = os.getcwd() + '/map.html'
    m.save(current_path)
if __name__ == '__main__':
    # Поиск Excel файлов в текущей директории
    excel_files = find_excel_files()
    # Проверка количества файлов
    if len(excel_files) == 0:
        print("Не найдено ни одного Excel файла в текущей директории.")
        exit(code=0)
    elif len(excel_files) > 1:
        print("Найдено более одного Excel файла. Скрипт прекращает
выполнение.")
        exit(code=0)
    else:
```

```
        file_path = excel_files[0]
    df = pd.read_excel(file_path)
    # Столбцы
    df.columns = df.iloc[0]
    # Данные
    df = df[1:]
    # Обработка адресов
    df['Полный адрес'] = df.apply(process_address, axis=1)
    df = df[~df['Полный адрес'].str.contains('nan', na=False)]
    # Получение координат
    df['Координаты'] = df['Полный адрес'].apply(get_coordinates)
    df[['Широта', 'Долгота', 'Серийный номер ПУ']] =
pd.DataFrame(df['Координаты'].tolist(), index=df.index)

    # Удаление строк, где координаты не найдены
    df.dropna(subset=['Широта', 'Долгота'], inplace=True)
    df.to_excel('Таблица_с_координатами_и_счетчиком.xlsx', index=False)
    create_map(df)
    print(f"Map has been saved")
```

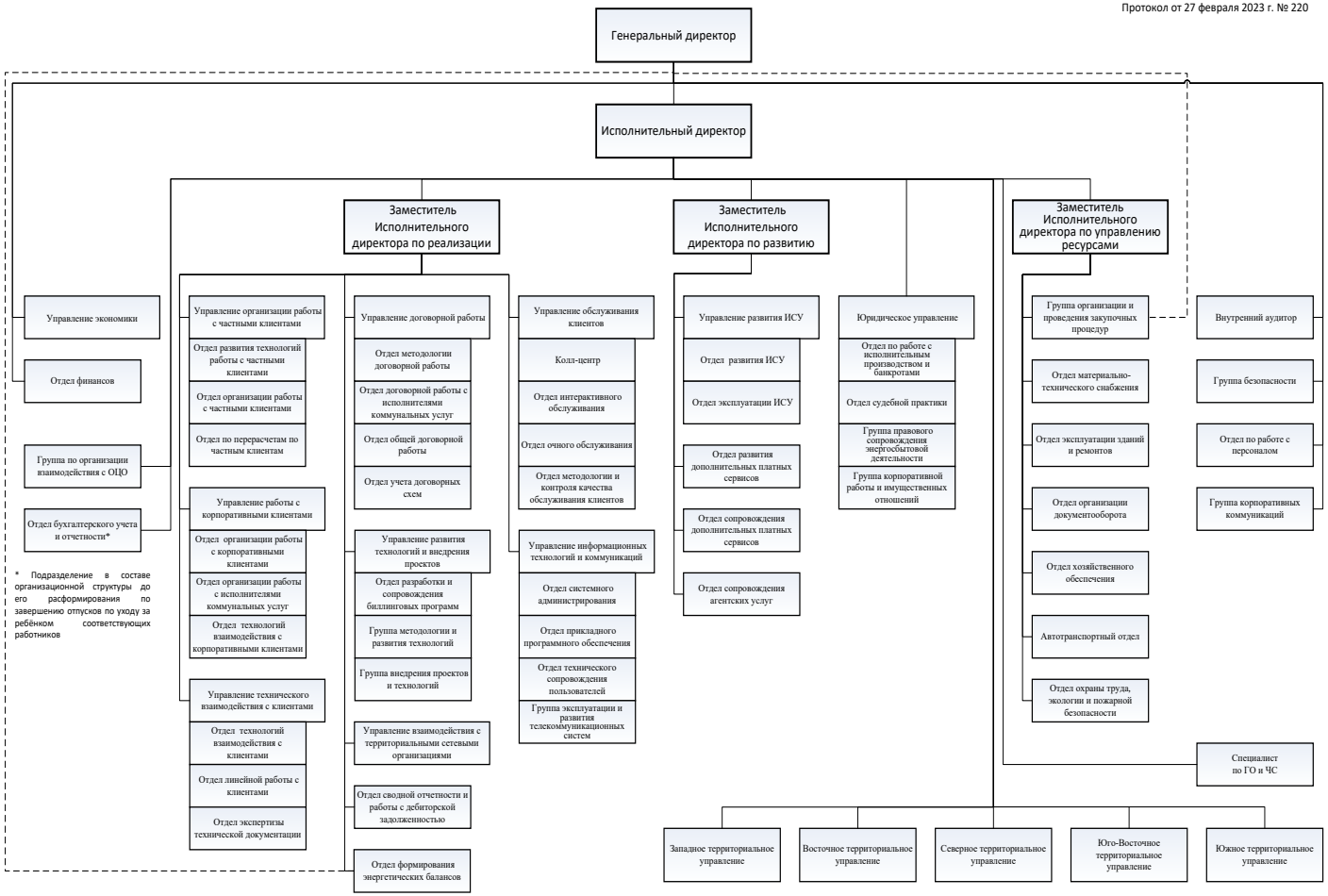
ПРИЛОЖЕНИЕ В

Структурная схема ПАО «Красноярскэнергосбыт»

Приложение № 1
к приказу «Об утверждении организационной структуры
ПАО «Красноярскэнергосбыт» с 01.03.2023 г.

УТВЕРЖДЕНО:
Решением Совета директоров
ПАО «Красноярскэнергосбыт»
Протокол от 27 февраля 2023 г. № 220

Организационная структура ПАО «Красноярскэнергосбыт»
(действует с 01 марта 2023 г.)



ПРИЛОЖЕНИЕ Г

Таблица Г.1 – Фрагмент структуры файла Excel

					Идентификация ПУ			Идентификация маршрутов опроса		Идентификация Абонента		
Район	**Населенный пункт**	**Улица**	**Дом**	**Квартира**	**Тип ПУ**	**Серийный номер ПУ**	**Дата установки ПУ**	**Прямой опрос TCP/IP**		**Номер лицевого счета абонента**	**Дата**	**Энергия А+ на начало суток**
								IP адрес	**Порт**			
3	**4**	**5**	**6**	**7**	**17**	**18**	**19**	**21**	**22**	**27**	**33**	**34**
Абанский р–н	поселок Абан	ул Просвещения	д 3	кв 18	МИРТЕК–12–РУ	8230160011906	04.01.2024	10.98.174.224	2020	115630150350	10.06.2024	3327
Абанский р–н	поселок Абан	ул Просвещения	д 5	кв 17	МИРТЕК–12–РУ	8230160053203	04.01.2024	10.98.173.180	2020	115630150730	17.06.2024	1106,42
Ачинский р–н	деревня Каменка	ул Лесная	д 6	кв 2	МИРТЕК–212–РУ	122T155718637	13.11.2023	10.96.108.38	2020	162002090540	02.06.2024	440,5

ПРИЛОЖЕНИЕ Д

Структурная схема АСКУЭ