

Федеральное государственное образовательное бюджетное
учреждение высшего образования
**«Финансовый университет
при Правительстве Российской Федерации»
(Финансовый университет)**

Департамент анализа данных и машинного обучения

Пояснительная записка к курсовой работе по дисциплине
«Современные технологии программирования»
на тему:

«Нейросетевая система прогнозирования электроэнергии»

Выполнил:
Студент группы ПИ21-7
Пятунин А. И.

Научный руководитель:
к.т.н., ст. преподаватель
Пальчевский Е.В.

Москва – 2023

Оглавление

Введение.....	4
1. Описание программы.....	7
1.1. Алгоритмические решения.....	7
1.1.1. <i>Безопасность</i>	7
1.1.2. <i>Клиент</i>	9
1.2. Описание интерфейса программы.....	13
1.2.1. <i>Навигация и Футер</i>	13
1.2.2. <i>Регистрация и Авторизация</i>	15
1.2.3. <i>Главная страница</i>	17
1.2.4. <i>Прогноз</i>	19
1.2.5. <i>Тренировка</i>	21
1.3. Архитектура приложения.....	22
1.3.1. <i>Зависимости проекта</i>	22
1.3.2. <i>Клиент</i>	25
1.3.3. <i>База данных</i>	26
2. Структура классов и их назначение в рамках проекта.....	28
2.1. Сервер.....	28
2.1.1. <i>Config</i>	29
2.1.2. <i>Controllers</i>	30
2.1.3. <i>DTO</i>	32
2.1.4. <i>Model</i>	32
2.1.5. <i>Network</i>	33
2.1.6. <i>Parse</i>	34
2.1.7. <i>Repositories</i>	35
2.1.8. <i>Security</i>	36
2.1.9. <i>Services</i>	37
2.1.10. <i>Static</i>	38
2.1.1. <i>Util</i>	38

5.2 Клиент.....	39
2.2.1. <i>Template</i>	39
2.2.2. <i>Static</i>	41
Заключение	43
Список использованных источников	44
Приложения	46

Введение

Потребление электроэнергии во всем мире стремительно растет в связи с постоянным увеличением населения, желанием улучшения уровня жизни и расширением индустриализации, что способствует положительному темпу экономического роста [1]. Прогнозирование потребления электроэнергии на средний и долгосрочный период имеет важное значение для планирования инвестиций в энергетику [2]. Например, избыточное использование ископаемого топлива для генерации электроэнергии в конце 20 века привело к истощению ресурсов. С начала 21 века все чаще используются возобновляемые источники энергии, такие как гидроэнергетика, солнечная и ветровая энергия. В отличие от традиционных источников энергии, ветровая, солнечная и гидроэнергетика обладают множеством преимуществ, таких как возможность вторичной переработки и экологическая безопасность, а также большой потенциал развития. Они способствуют установлению более эффективной и чистой энергетической структуры. Однако, из-за неопределенности и взаимосвязи гидро, ветровой и солнечной энергии, прогнозирование потребления электроэнергии становится чрезвычайно важным, но сложным. Точное прогнозирование может содействовать более эффективному использованию возобновляемой энергии. Кроме того, точное прогнозирование потребления электроэнергии может определять правительственные стратегии будущего использования и развития энергетики.

Наиболее часто используемые методы прогнозирования потребления электроэнергии включают регрессионные модели [3], модели временных рядов [4], теорию нечетких данных [5], нейронные сети [6], байесовские сети [7], гибридные методы [8] и др. Регрессионный анализ и модели временных рядов являются наиболее известными методами моделирования в прогнозировании потребления электроэнергии [9]. С развитием современных методов искусственного интеллекта в прогнозирование электроэнергии внедряются искусственные нейронные сети [10]. С точки зрения результатов прогнозирования, существующие методы можно разделить на

детерминистические точечные прогнозы и вероятностные прогнозы, основанные на анализе неопределенностей [12]. Точный точечный прогноз не учитывает колебания потребления электроэнергии. На реальное потребление электроэнергии и рост нагрузки влияют различные факторы, в том числе экономическое развитие, структура промышленности, уровень доходов населения, климатические условия, географическая среда, национальная политика (цена на электроэнергию) и т.д. Все эти факторы взаимодействуют и влияют друг на друга. Кроме того, интеграция информатизации и индустриализации в электроэнергетике приводит к быстрому росту данных об энергетике, что приводит к большому количеству данных с разнообразными источниками, функциями и объемами.

Объектом исследования является процесс прогнозирования значений потребления электроэнергии.

Предметом исследования являются нейросетевые модели для прогнозирования временных рядов.

Таким образом, **целью** проекта является разработка нейросетевой системы прогнозирования потребления электроэнергии.

Для достижения поставленной цели необходимо решить следующие **задачи**:

- разработка клиент-серверного приложения на языке программирования Java, предназначенного для прогнозирования потребления электроэнергии на основе искусственной нейронной сети;

- создание серверной части приложения на основе фреймворка Spring Boot, отвечающей за обработку запросов клиентов и предоставление нейросетевых прогнозных данных;

- разработка клиентской части приложения с графическим интерфейсом с помощью HTML, CSS и JavaScript, обеспечивающей удобный интерфейс для взаимодействия с сервером и получения прогнозных данных;

– применение модели MVC (Model View Controller) для разделения управляющей логики на отдельные компоненты, что позволяет улучшить фундаментальные свойства системы, в том числе и повысить читаемость исходного кода;

– создание облачной кроссплатформенной веб-геоинформационной системы для нейросетевого расчета прогнозных значений потребления электроэнергии.

Для решения выше поставленных задач используются технологии искусственного интеллекта, а также следующие инструментальные средства: Java, Spring Boot [13-15], MySQL, ORM и т.д.

Таким образом, разработка веб-ГИС является актуальным инструментом для энергетических компаний, нуждающихся в прогнозировании электроэнергии для своей деятельности.

1. Описание программы

1.1. Алгоритмические решения

1.1.1. Безопасность

Отдельное внимание стоит уделить безопасности приложения и то, как она реализована (рисунок 1).

Для начала, после запуска проекта, настраивается цепочка фильтров, которая отвечает за основные моменты безопасности, например: какая страница будет отвечать за вход в приложение, на какие страницы будет доступ у пользователей, а на какие нужны будут отдельные разрешения, на какой адрес будет осуществляться выход пользователя из приложения и куда программа будет перенаправлять при каких-либо ошибках.

Затем инициализируется несколько бин компонентов, которые будут отвечать за настройку `AuthenticationManager`. `AuthenticationManager` — это интерфейс в `Spring Security`, который отвечает за процесс аутентификации пользователей.

`AuthenticationManager` принимает объект `Authentication`, который содержит учетные данные пользователя, и возвращает объект `Authentication`, который содержит информацию о пользователе, если аутентификация прошла успешно. Если аутентификация не удалась, `AuthenticationManager` может выбросить исключение `AuthenticationException`, которое указывает на причину ошибки.

`AuthenticationManager` используется в `Spring Security` для проверки учетных данных, полученных от пользователя, и выполнения процесса аутентификации. Он может быть настроен с помощью Java-кода или XML-файлов или как в нашем случае с помощью авто-конфигурации `Spring Boot`.

`AuthenticationManager` работает с различными источниками данных, такими как базы данных, файлы конфигурации, LDAP-серверы и т.д. Также он работает с различными механизмами аутентификации, такими как базовая

аутентификация, форма аутентификации, аутентификация через социальные сети и т.д.

В Spring Security существует несколько реализаций интерфейса `AuthenticationManager`, таких как `ProviderManager`, которая делегирует аутентификацию другим провайдерам, и `DaoAuthenticationProvider`, который использует базу данных для аутентификации пользователей, что я и использовал в своем проекте.

При настройке `DaoAuthenticationProvider` необходимо указать `UserDetailsService` и `PasswordEncoder`.

`UserDetailsService` — это интерфейс в Spring Security, который отвечает за загрузку информации о пользователе из источника данных, такого как база данных.

`UserDetailsService` содержит метод `loadUserByUsername()`, который принимает имя пользователя в качестве параметра и возвращает объект `UserDetails`, который содержит информацию о пользователе, такую как имя пользователя, пароль и список ролей. Если пользователь не найден, `UserDetailsService` может выбросить исключение `UsernameNotFoundException`.

`PasswordEncoder` отвечает за хэширование паролей пользователей для обеспечения безопасности в приложениях.

`PasswordEncoder` принимает пароль пользователя в качестве параметра и возвращает хэш-код, который сохраняется в базе данных или в других источниках данных. При аутентификации пользователя, введенный пользователем пароль сравнивается с сохраненным хэш-кодом. В программе использовался такой способ шифрования как `BCrypt`.

Таким образом, `AuthenticationManager` — это интерфейс в Spring Security, который отвечает за процесс аутентификации пользователей. Он используется для проверки учетных данных пользователей.

В программе используется защита от межсайтовой подделки запросов `CSRF` (`Cross-Site Request Forgery`) для обеспечения дополнительной безопасности.

При регистрации по умолчанию устанавливается роль обычного пользователя, для получения дополнительных прав, можно связаться с автором приложения по номеру телефона указанном в футере сайта.

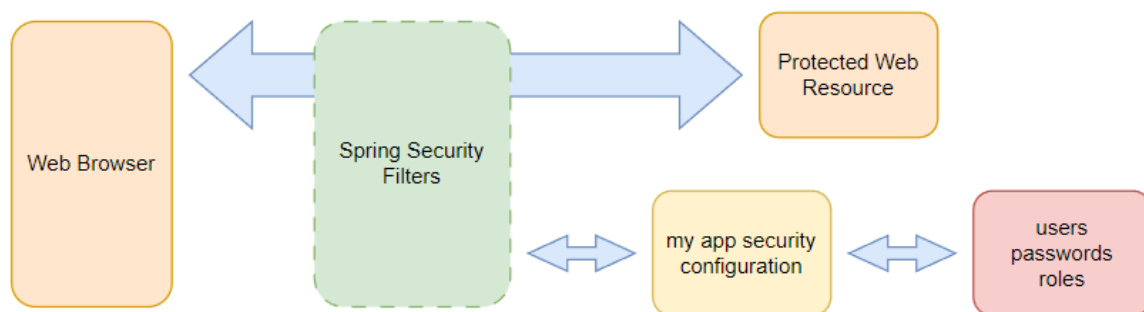


Рисунок 1 – Архитектура безопасности

1.1.2. Клиент

В приложении используется архитектура MVC (Model-View-Controller), которая разделяет приложение на три основных компонента: модель, представление и контроллер. Взаимодействие всех элементов можно рассмотреть на рисунке 2.

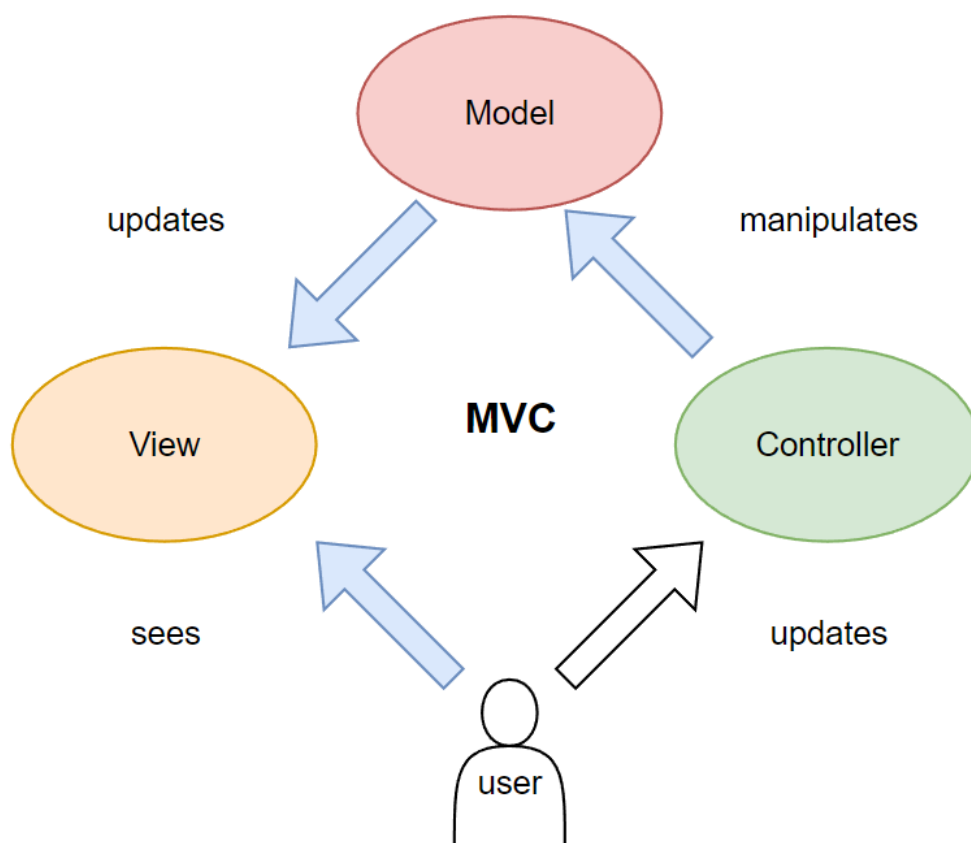


Рисунок 2 – Взаимодействие в паттерне MVC

Модель (Model) — это компонент, отвечающий за обработку данных. Он содержит классы, которые представляют данные и методы для работы с ними. Например, класс User, который представляет информацию о пользователе, и UserRepository, который содержит методы для работы с данными пользователей.

Представление (View) — это компонент, который отображает данные в пользовательском интерфейсе. Он содержит файлы представлений, например HTML-страницы, которые отображают данные, полученные из модели. В Spring Boot для рендеринга представлений использует шаблонизатор Thymeleaf или другие аналогичные решения.

Контроллер (Controller) — это компонент, который связывает модель и представление. Он содержит методы, которые обрабатывают запросы от пользователя и возвращают данные в нужном формате. Например,

UserController, который содержит методы для обработки запросов, связанных с пользователями.

Взаимодействие пользователя и приложения:

- пользователь открывает браузер и вводит URL-адрес приложения;
- сервер принимает запрос от пользователя и сопоставляет URL-адрес с соответствующим методом контроллера;
- метод контроллера обращается к методам JPA-репозитория для извлечения данных из БД их обработки и последующей передачи их в представление;
- представление использует данные, полученные из контроллера, и отображает их в пользовательском интерфейсе;
- пользователь взаимодействует с интерфейсом, например, заполняет форму и отправляет ее;
- сервер принимает запрос от пользователя, сопоставляет URL-адрес с соответствующим методом контроллера и обрабатывает запрос, используя данные, полученные от пользователя;
- метод контроллера обрабатывает запрос от пользователя и возвращает результат;
- представление использует данные, переданные из контроллера, и отображает их в пользовательском интерфейсе;
- пользователь продолжает взаимодействовать с интерфейсом, и процесс повторяется.

Таким образом, пользователь взаимодействует с приложением через браузер.

1.2.3. Нейронная сеть

Нейронная сеть – это алгоритм машинного обучения, который состоит из множества взаимосвязанных узлов, называемых нейронами. Нейроны

объединяются в слои, и каждый слой выполняет определенную функцию в обработке входных данных.

Это приложение использует нейронную сеть для прогнозирования энергетических параметров на основе входных данных, введенных пользователем. Чтобы получить прогноз по энергетическим параметрам, пользователь указывает даты, для которых необходимо получить прогноз.

Администратор может обучать нейронную сеть, передавая параметры, количество дней для обучения и прогнозирования, количество эпох, learning rate, количество скрытых слоев и количество нейронов в каждом скрытом слое, размеры тренировочной, валидационной и тестовой выборок. Когда запрос на обучение отправлен, контроллер обрабатывает его и запускает процесс обучения нейронной сети с заданными параметрами.

Обучение происходит на тренировочных данных и проверяется на валидационных данных. После завершения обучения модель сохраняется и используется для прогнозирования значений на основе входных данных. Использование нейронной сети позволяет улучшить качество прогнозов.

Основные компоненты нейронной сети включают:

- входной слой, который принимает входные данные и передает их на обработку;
- скрытые слои, которые обрабатывают входные данные, используя веса и функции активации;
- выходной слой, который предсказывает результат на основе обработанных данных;
- веса, которые определяют силу связей между нейронами;
- функции активации, которые определяют выходное значение каждого нейрона в зависимости от входного сигнала;
- функция потерь, которая измеряет ошибку прогнозирования и используется для обучения нейронной сети.

Процесс обучения нейронной сети включает в себя оптимизацию весов, чтобы минимизировать функцию потерь. Это достигается путем прямого распространения входных данных через сеть, вычисления ошибки прогнозирования и обратного распространения ошибки через сеть, чтобы обновить веса.

1.2. Описание интерфейса программы

В данном разделе будет описываться интерфейс программы, написанный на языках HTML, CSS и JavaScript.

Основной целью интерфейса является предоставление пользователю удобного и интуитивно понятного способа взаимодействия с программой. Для этого он включает в себя различные элементы управления, такие как кнопки, текстовые поля, выпадающие списки и т.д., которые позволяют пользователю выполнить необходимые действия.

1.2.1. Навигация и Футер

В интерфейсе реализована удобная навигация в верхней части экрана (рисунок 3).

Навигация в браузере выглядит следующим образом:

- о проекте;
- сделать прогноз;
- контакты;
- научная работа.

Если пользователь не авторизован, то также отображаются ссылки на "Войти" и "Регистрацию" в виде кнопок, в мобильной версии (рисунок 4). Если пользователь авторизован, то отображается кнопка "Выйти", которая позволяет пользователю выйти из своей учетной записи.

Также на странице присутствует иконка "login", которая отображается только для пользователей, которые не авторизованы, в веб-версии (рисунок 3). При нажатии на эту иконку пользователь будет перенаправлен на страницу авторизации.



Рисунок 3 – Навигация сайта, веб-версия

Аналогично реализована навигация и для смартфона (рисунок 4).

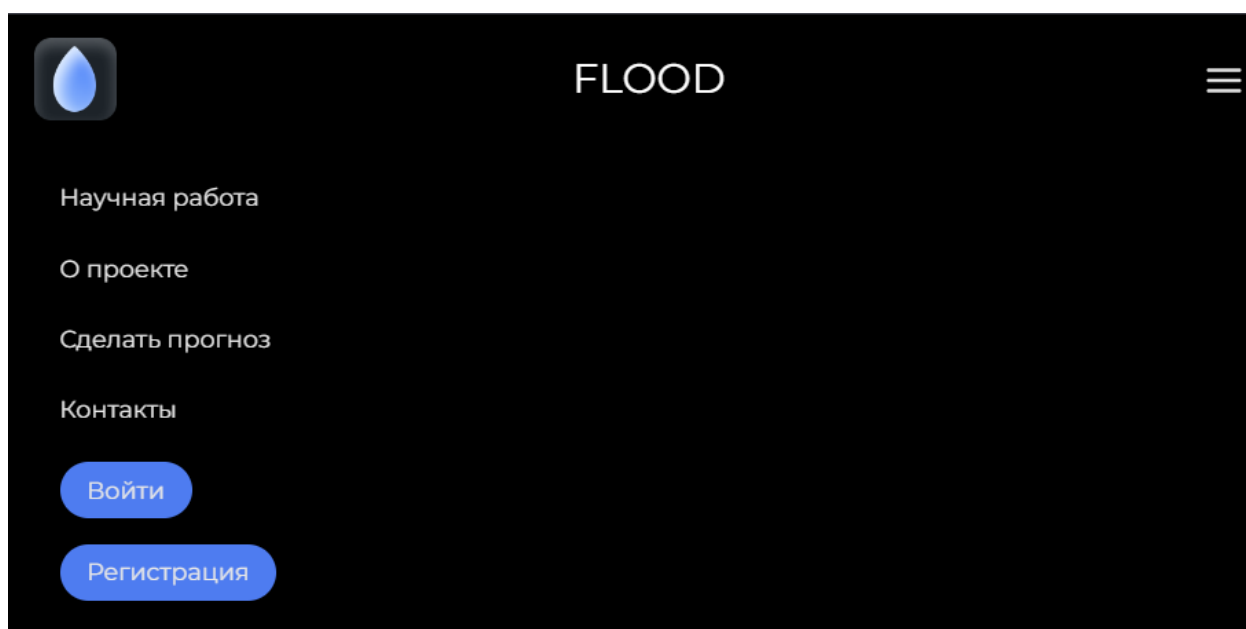


Рисунок 4 – Навигация сайта, мобильная-версия

Футер (англ. footer) — это нижняя часть веб-страницы, которая содержит дополнительную информацию и ссылки на другие страницы или ресурсы, (рисунок 5). В данном проекте футер содержит следующие элементы:

- «О нас» — это небольшой текст с описанием цели проекта;
- «Меню» — это все ссылки, которые доступны пользователю без дополнительных прав;

– «Контакты» — это место, где расположены почта и телефон научного руководителя проекта.



Рисунок 5 – Футер

1.2.2. Регистрация и Авторизация

Регистрация пользователя — это процесс, который позволяет создать учетную запись на сайте или в приложении и получить доступ к его функциональности. В данном проекте, для регистрации пользователя ему необходимо внести свою почту, имя и пароль (рисунок 6).

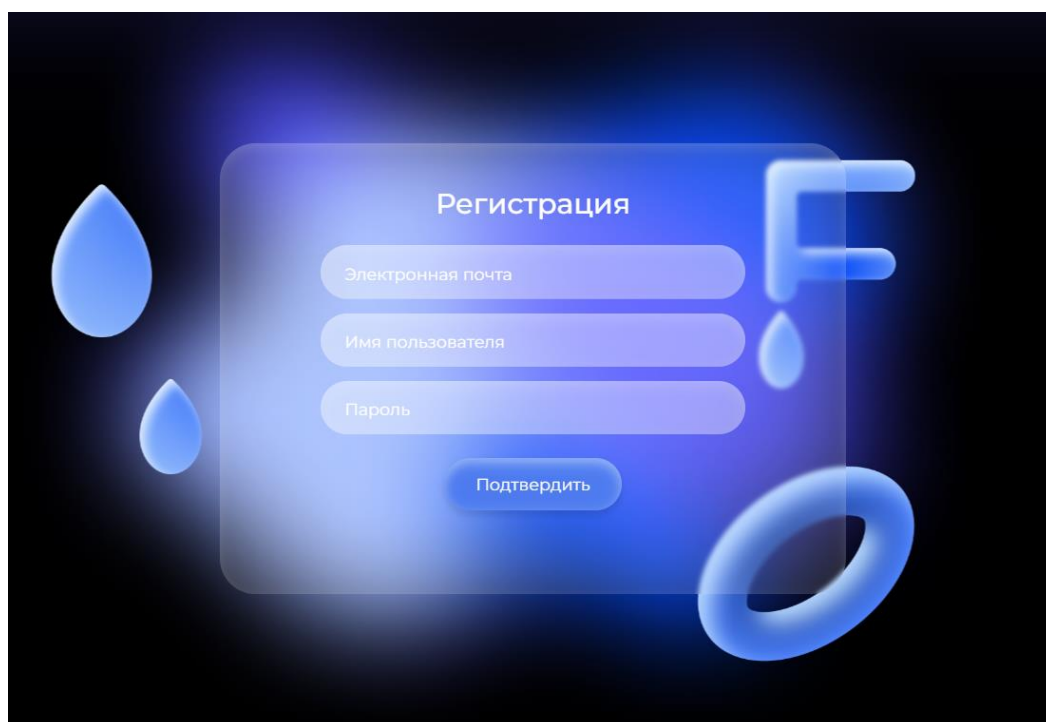


Рисунок 6 – Регистрация

При регистрации пользователь должен заполнить все требуемые поля, которые включают в себя следующую информацию:

– почта — адрес электронной почты, который будет использоваться для связи с пользователем;

– имя — имя пользователя, которое будет отображаться на сайте;

– пароль — пароль, который будет использоваться для входа на сайт.

После заполнения всех полей пользователь должен нажать на кнопку "Зарегистрироваться". При успешной регистрации система перенаправляет пользователя на страницу входа в приложение (рисунок 7).

При регистрации пользователю следует учитывать следующее:

– проверка введенной почты на корректность. Почта должна иметь синтаксис настоящей электронной почты, а просто набор букв;

– пароль должен быть длиннее чем 8 символов;

– если почта уже используется одним из пользователей сайта, то необходимо выбрать другую. Это сделано для избегания путаницы.

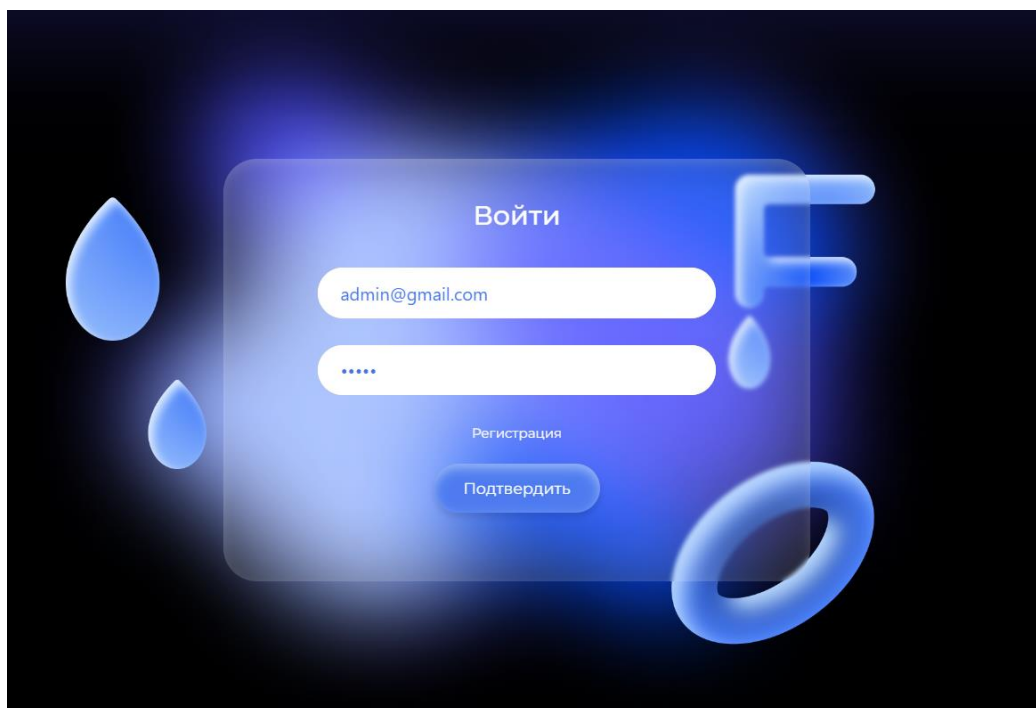


Рисунок 7 – Авторизация

Авторизация — это процесс, который позволяет пользователю войти в свою учетную запись на сайте или в приложении, используя свои учетные данные.

В данном проекте, для авторизации пользователю необходимо ввести только свою почту и пароль. После ввода данных пользователь должен нажать на кнопку "Войти", после чего система проверит правильность введенных данных и, если они корректны, предоставит пользователю доступ к его учетной записи на сайте или в приложении. Если данные оказались неверны, пользователя вернет на страницу авторизации и высветит сообщение «Неправильные почта или пароль».

1.2.3. Главная страница

Главная страница сайта — это первая страница, которую пользователь увидит при заходе на сайт. Ее основная задача — представить проект и привлечь внимание пользователя, а также предоставить ему информацию о доступной функциональности и возможностях проекта (рисунок 8).

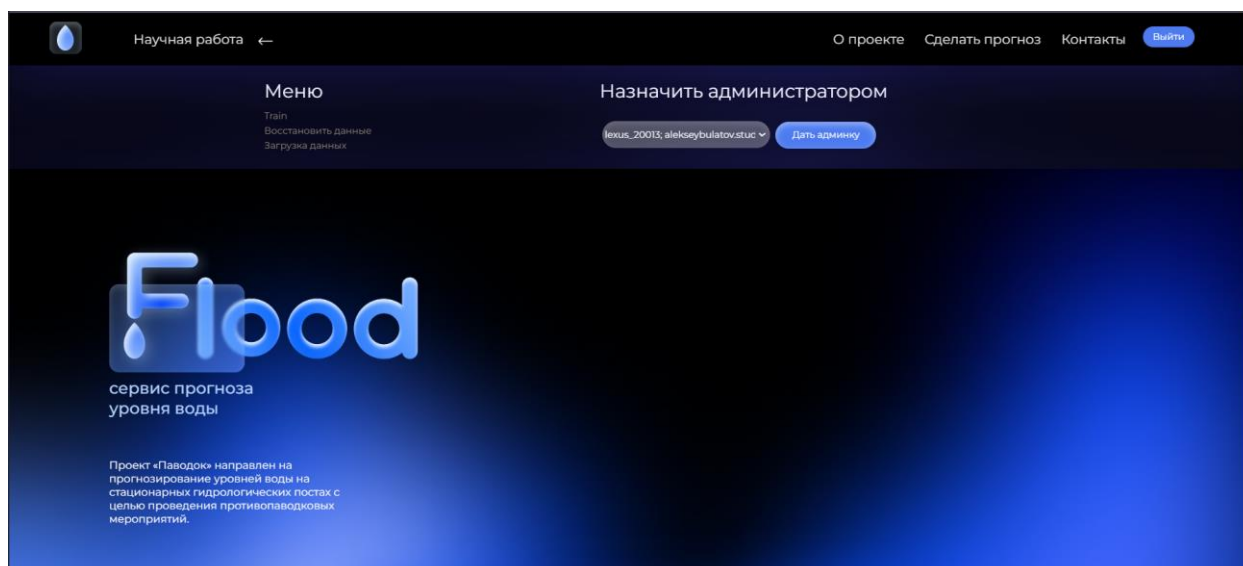


Рисунок 8 – Главная страница

В данном проекте главная страница содержит следующие элементы:

– шапка страницы — это верхняя часть страницы, которая содержит логотип проекта, название и основное меню навигации по сайту;

– крупное изображение — это элемент, который содержит название и основную суть проекта;

– тезисы — это блоки, которые содержат информацию о доступных функциях проекта. В данном проекте тезисы включают в себя, предсказание энергетики, научную работу и контакты (рисунок 9);

– футер (рисунок 5) — это нижняя часть страницы, которая содержит дополнительную информацию и ссылки на другие страницы или ресурсы. Футер содержит ссылки на социальные сети, ссылки на другие страницы, контактную информацию и дополнительную информацию о проекте;

– администраторская панель – это панель, которая будет доступна пользователям с правами «ADMIN», на ней можно получить доступ к назначению других пользователей администраторами, тренировке моделей, парсингу данных и восстановлению данных. Если пользователь не является администратором, то такой панели он видеть не будет.

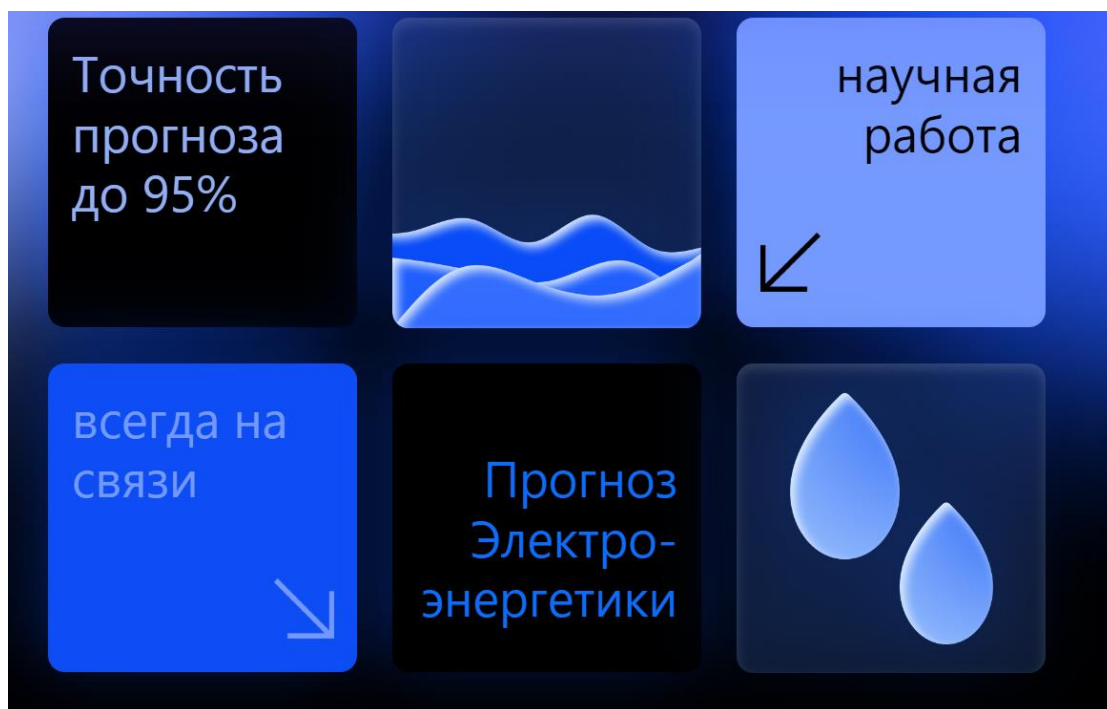


Рисунок 9 – Тезисы

1.2.4. Прогноз

Страница прогноза представляет собой интерфейс, на котором пользователь может произвести прогноз затраты энергии на определенный период времени (рисунок 10).

Для начала, пользователя встречает небольшое обучение, которое рассказывает, как сделать прогноз. Затем, достаточно выбрать дату и нажать кнопку отправить. Все, после загрузки, появится прогноз на запрашиваемые даты в виде таблицы (рисунок 11).

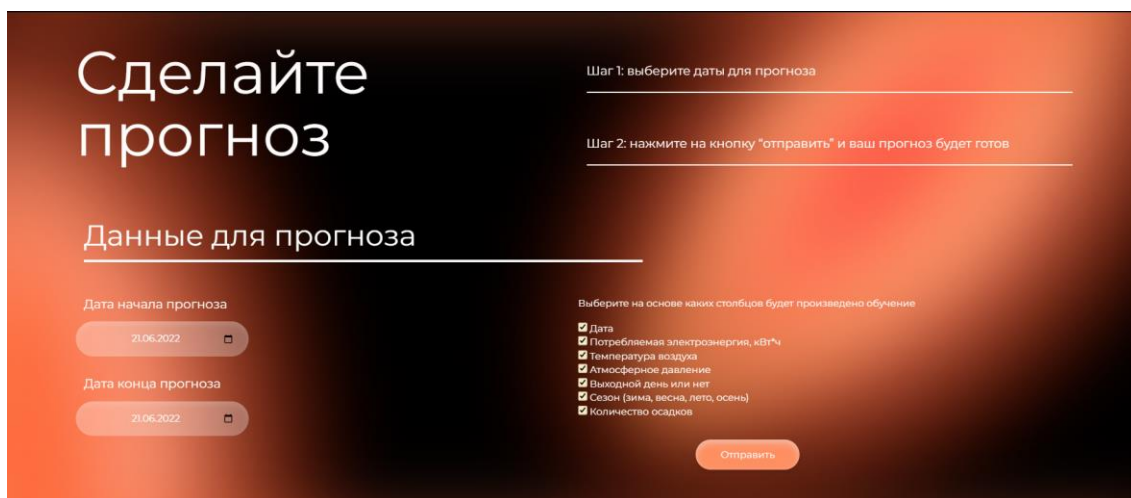


Рисунок 10 – Страница прогноза

Для администратора, в прогнозе появляется небольшое меню, в котором он может выбрать на основе каких столбцов из базы данных необходимо обучать модель, это позволяет сделать настройку прогноза более точной.

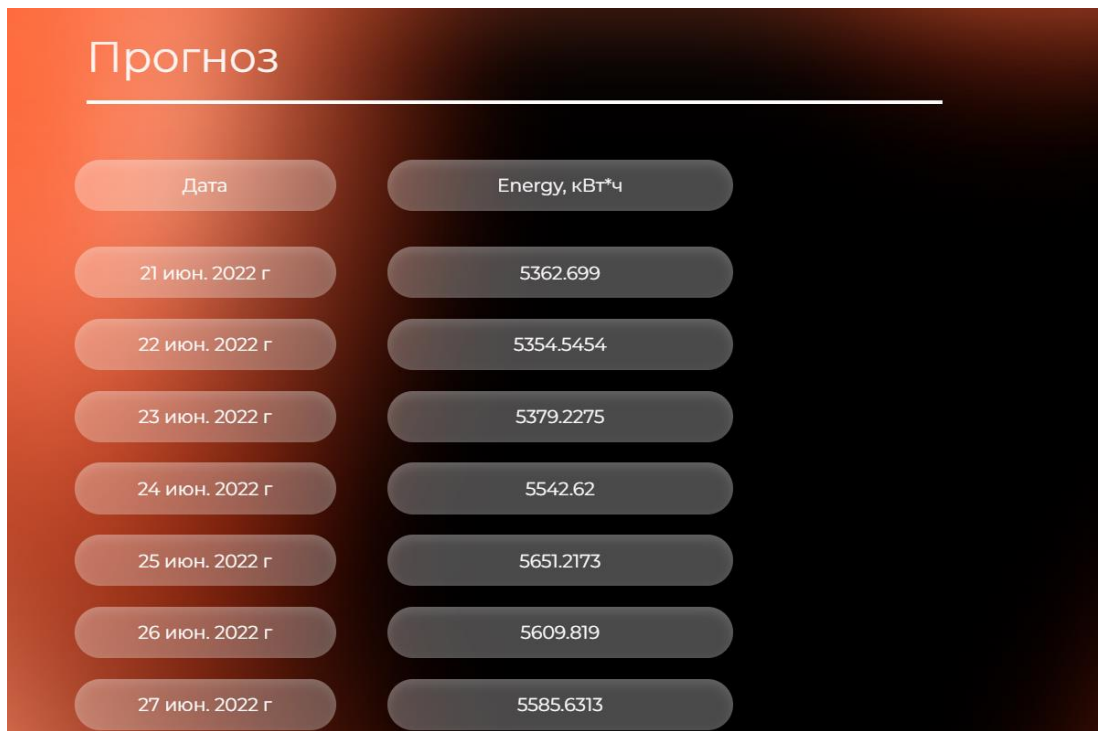


Рисунок 11 – Результат прогноза

После прогноза, у пользователя есть возможность загрузить файл Excel, с результатами и увидеть график, который добавит наглядности и будет удобен при большом количестве данных (рисунок 12).



Рисунок 12 – Дополнительные возможности

1.2.5. Тренировка

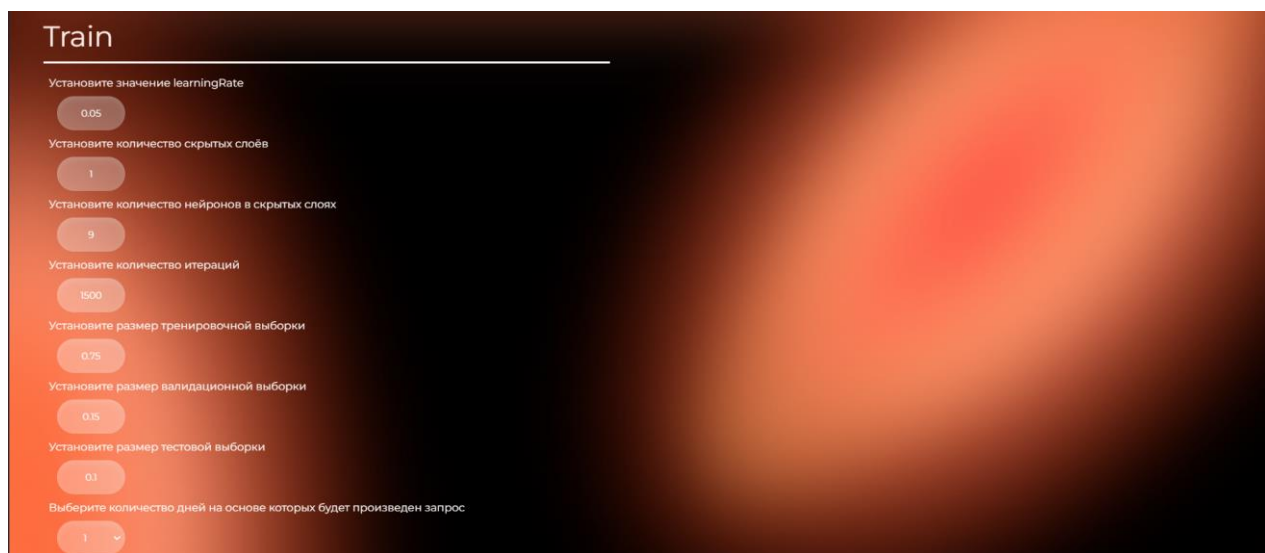
Данная страница представляет собой форму (рисунок 13), на которой администратор может настроить параметры обучения модели машинного обучения.

В верхней части страницы расположен заголовок "Train" и изображение линии, разделяющее заголовок от остальной формы.

Далее администратор может ввести значения параметров обучения в соответствующие поля ввода. Например, можно установить значение learningRate, количество скрытых слоев, количество нейронов в скрытых слоях и т.д.

Кроме того, администратор выбирает количество дней, на основе которых будет произведен запрос, а также количество дней для прогноза. Также можно выбрать на основе каких столбцов будет произведено обучение, отметив соответствующие чекбоксы.

В конце формы расположена кнопка "Submit", при нажатии на которую произойдет отправка данных на сервер для обучения модели.



The image shows a web form titled "Train" with a dark orange background. The form contains several input fields, each with a label and a value:

- Установите значение learningRate: 0.05
- Установите количество скрытых слоёв: 1
- Установите количество нейронов в скрытых слоях: 9
- Установите количество итераций: 1500
- Установите размер тренировочной выборки: 0.75
- Установите размер валидационной выборки: 0.15
- Установите размер тестовой выборки: 0.1
- Выберите количество дней на основе которых будет произведен запрос: 1

Рисунок 13 – Страница тренировки модели

1.3. Архитектура приложения

1.3.1. Зависимости проекта

В файле `pom.xml` представлены зависимости проекта, которые определяют, какие библиотеки и фреймворки будут использоваться в проекте. Ниже приведен список всех зависимостей, которые указаны в данном файле:

- `spring-boot-starter-data-jpa` — зависимость для работы с базой данных с использованием JPA;
- `spring-boot-starter-thymeleaf` — зависимость для работы с шаблонизатором Thymeleaf;
- `selenium-support` — зависимость для работы с Selenium WebDriver, который используется для автоматизации тестирования;
- `spring-boot-devtools` — зависимость для облегчения разработки приложения в режиме разработки;
- `spring-boot-starter-validation` — зависимость для работы с валидацией данных;
- `spring-boot-starter-web` — зависимость для работы с веб-приложением;
- `mysql-connector-j` — зависимость для работы с базой данных MySQL;
- `lombok` — зависимость для использования аннотаций, которые облегчают написание кода;
- `spring-boot-starter-test` — зависимость для написания тестов;
- `poi` — зависимость для работы с форматом документов Microsoft Office;
- `poi-ooxml` — зависимость для работы с форматом документов Microsoft Office в формате OpenXML;
- `spring-boot-starter-security` — зависимость для работы с безопасностью веб-приложения;
- `thymeleaf-extras-springsecurity5` — зависимость для работы с безопасностью веб-приложения в сочетании с Thymeleaf;

- `spring-security-test` — зависимость для написания тестов безопасности веб-приложения;
- `modelmapper` — зависимость для работы с маппингом объектов;
- `json-simple` — зависимость для работы с форматом JSON;
- `json` — зависимость для работы с форматом JSON;
- `jsoup` — зависимость для работы с HTML-документами;
- `selenium-java` — зависимость для работы с Selenium WebDriver;
- `log4j-core` — зависимость для работы с логированием.

Каждая зависимость представляет собой отдельный инструмент или библиотеку, которая используется в проекте для выполнения определенных задач.

Некоторые из самых важных зависимостей в этом списке это:

- `spring-boot-starter-web` — это основная зависимость для создания веб-приложения на Spring. Она включает в себя все необходимые библиотеки и компоненты, такие как Tomcat, Jackson, Spring MVC, которые позволяют создавать веб-приложения на Spring;

- `spring-boot-starter-data-jpa` — это зависимость для работы с базой данных с использованием JPA. Она включает в себя все необходимые библиотеки и компоненты, такие как Hibernate, Spring Data JPA, которые облегчают работу с базами данных и позволяют быстро создавать запросы к базе данных;

- `spring-boot-starter-security` — это зависимость для создания безопасных веб-приложений. Она включает в себя все необходимые библиотеки и компоненты, такие как Spring Security, которые позволяют защитить веб-приложение от несанкционированного доступа, обеспечить аутентификацию и авторизацию пользователей;

- `lombok` — это библиотека, которая облегчает написание кода на Java. Она предоставляет аннотации, которые автоматически генерируют геттеры,

сеттеры, конструкторы и другие методы, что упрощает написание кода и сокращает его объем;

– `jsoup` — это библиотека для работы с HTML-документами. Она позволяет быстро и удобно извлекать данные из HTML-документов, выполнять поиск и обработку данных, а также создавать HTML-документы;

– `selenium-java` — это библиотека для автоматизации тестирования веб-приложений. Она позволяет выполнять автоматические тесты, которые могут проверять работу веб-приложения на различных браузерах и платформах;

Структура файла `pom.xml` следующая (рисунок 14):

– `project` — это корневой элемент файла, определяющий проект. Он содержит информацию о версии проекта, его артефакте, группе и имени. Также он содержит информацию о родительском проекте, если он существует, и описывает зависимости проекта;

– `model Version` — это элемент, который указывает на версию модели проекта, используемой в файле `pom.xml`;

– `properties` — это элемент, который содержит определенные пользователем свойства, которые могут использоваться в других частях файла `pom.xml`;

– `dependencies` — это элемент, который определяет зависимости проекта от других библиотек или проектов. Он содержит информацию о группе, артефакте и версии зависимости;

– `build` — это элемент, который определяет конфигурацию сборки проекта, такую как цели сборки, плагины и профили;

– `plugins` — это элемент, который определяет список плагинов, используемых в проекте. Каждый плагин содержит информацию о группе, артефакте и версии плагина, а также о его конфигурации.


```

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.0.1</version>
    <relativePath/>
  </parent>
  <groupId>org.neuralNetworks</groupId>
  <artifactId>demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>demo</name>
  <description>project for Neural Network</description>
  <properties>
    <java.version>19</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
  </dependencies>

```

Рисунок 14 – Фрагмент кода зависимостей

1.3.2. Клиент

На стороне клиента используется HTML, CSS, JavaScript, Bootstrap и Thymeleaf.

HTML является основой веб-страниц и используется для создания структуры и содержимого интерфейса. CSS используется для стилизации веб-страниц и задания внешнего вида элементов интерфейса. Bootstrap — это CSS-фреймворк, который предоставляет готовые компоненты и стили для создания адаптивного и кроссбраузерного интерфейса.

JS используется для создания интерактивности и динамических элементов интерфейса. Thymeleaf — это шаблонизатор для веб-приложений, который позволяет создавать HTML-шаблоны с использованием Java-кода и вставлять данные из модели Java в HTML-шаблоны.

Таким образом, использование чистого JS, HTML, CSS, Bootstrap и Thymeleaf позволяет создавать красивый, адаптивный и интерактивный веб-интерфейс для проекта.

1.3.3. База данных

Настройки базы данных и таблиц находятся в файле `application.yml` (рисунок 15) в проекте. В них определяются параметры подключения к базе данных, а также настройки таблиц.

```
spring:
  datasource:
    driver-class-name:
    url:
    username:
    password:
  jpa:
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MySQLDialect
        show_sql: true
    open-in-view: false
  mvc:
    hiddenmethod:
      filter:
        enabled: true
  server:
    error:
      path: "/error"
```

Рисунок 15 - Файл настроек подключения

Данный код содержит настройки для подключения к базе данных MySQL и использования JPA (Java Persistence API) для работы с базой данных.

В блоке `spring.datasource` задаются параметры подключения к базе данных, такие как `driver-class-name` (имя драйвера), `url` (URL-адрес базы данных), `username` (имя пользователя базы данных) и `password` (пароль пользователя базы данных).

В блоке `spring.jpa` задаются настройки JPA, такие как `hibernate.dialect` (диалект Hibernate для работы с базой данных), `hibernate.show_sql`

(включение/отключение вывода SQL-запросов в консоль), и `open-in-view` (включение/отключение сеанса Hibernate для каждого HTTP-запроса).

Блок `server.error` содержит путь, по которому выводятся сообщения об ошибках на сервере.

Блок `spring.mvc.hiddenmethod.filter.enabled` включает фильтр для обработки HTTP-запросов с параметром `_method`, который используется для передачи метода HTTP, отличного от GET и POST.

Таким образом, эти настройки позволяют приложению подключаться к базе данных MySQL и использовать JPA для работы с данными в базе данных. Остальные настройки определяют поведение сервера при обработке запросов и отображении ошибок.

2. Структура классов и их назначение в рамках проекта

2.1. Сервер

На стороне сервера написано 42 класса которые разбиты на структуру пакетов (рисунок 16).

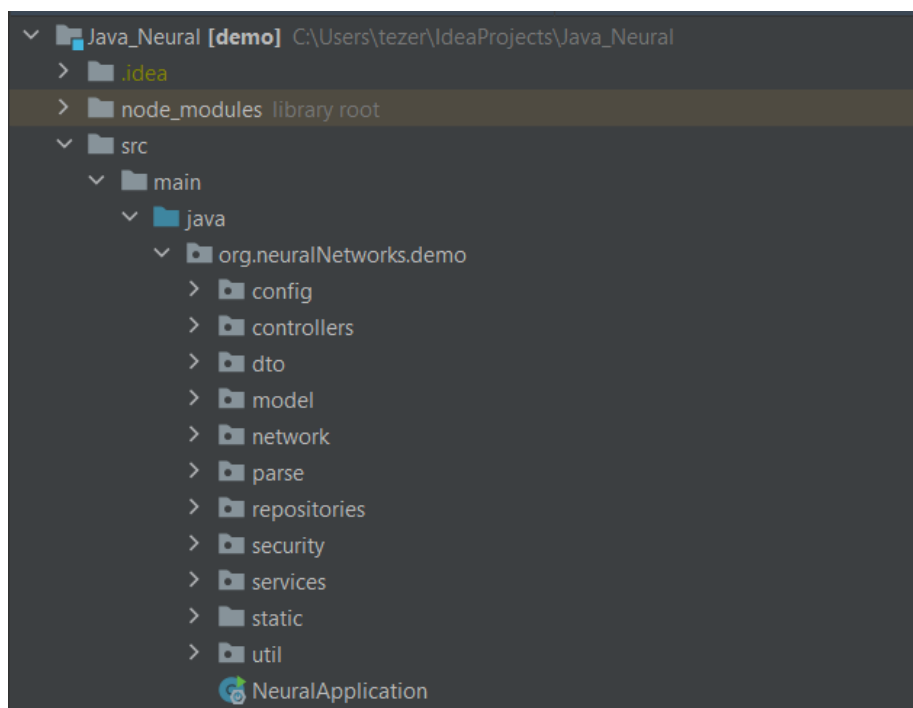


Рисунок 16 – Структура классов

Структура программы создана в удобном формате, где каждый пакет отвечает только за свою логику и свои классы.

– config – пакет, содержащий классы конфигурации приложения, которые определяют различные настройки и параметры, связанные с работой приложения;

– controllers – пакет, содержащий классы контроллеров, которые обрабатывают запросы от клиента и управляют логикой обработки этих запросов;

– `dto` – пакет, содержащий классы DTO (Data Transfer Object), которые используются для передачи данных между слоями приложения или между приложением и клиентом;

– `model` – пакет, содержащий классы моделей, которые представляют сущности и объекты данных в приложении;

– `network` – пакет, содержащий классы, связанные с работой нейронной сети;

– `parse` – пакет, содержащий классы, связанные с обработкой и парсингом данных, классы для работы с форматами данных, такими как JSON или XML;

– `repositories` – пакет, содержащий классы репозитория, которые предоставляют доступ к данным в базе данных;

– `security` – пакет, содержащий классы, связанные с безопасностью приложения, классы для работы с аутентификацией и авторизацией пользователей;

– `services` – пакет, содержащий классы, которые предоставляют бизнес-логику и функциональность приложения, включая обработку данных и выполнение операций;

– `static` – директория, содержащая статические ресурсы, в данном проекте там хранятся модели, которые записываются после каждого прогноза;

– `util` – пакет, содержащий утилитарные классы, которые предоставляют общие функции и методы, используемые в различных частях приложения.

Теперь, разберем каждый пакет по отдельности.

2.1.1. Config

Пакет содержит 3 класса, каждый из которых отвечает за свою часть настроек (рисунок 17).

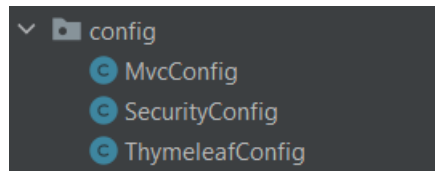


Рисунок 17 – Config

`ThymeleafConfig` – класс конфигурации для Thymeleaf, который определяет два бина: `springTemplateEngine()` и `htmlTemplateResolver()`. `springTemplateEngine()` создает объект шаблонизатора Thymeleaf, а `htmlTemplateResolver()` определяет, как Thymeleaf должен искать HTML-шаблоны.

`SecurityConfig` – класс конфигурации для Spring Security, который определяет настройки безопасности приложения. Он определяет бины для объектов `SecurityFilterChain`, `AuthenticationManager`, `AuthenticationProvider` и `PasswordEncoder`, а также определяет правила авторизации для различных URL-адресов.

`MvcConfig` – класс конфигурации для Spring MVC, который определяет обработчики ресурсов. Он определяет метод `addResourceHandlers()`, который регистрирует обработчик для статических ресурсов, таких как CSS, JS и изображения.

2.1.2. Controllers

`AuthController` – контроллер, который используется для авторизации и регистрации пользователей в веб-приложении. Контроллер содержит методы, которые обрабатывают запросы GET и POST для страниц регистрации и входа в систему. Он использует классы `PersonValidator`, `RegistrationService` и `AuthenticationUtil` для проверки и обработки данных, связанных с регистрацией пользователей. В методе `performRegistration` контроллер проверяет данные пользователя на корректность, затем регистрирует его в системе с помощью сервиса регистрации и перенаправляет на страницу входа в систему.

`CustomErrorController` – контроллер, который обрабатывает ошибки в приложении и отображает пользовательские страницы ошибок. Он содержит методы для обработки различных типов ошибок, таких как ошибки 404 и 500.

`EnergyController` – контроллер, который обрабатывает запросы, связанные с данными о потреблении энергии. Он содержит методы для добавления, обновления и удаления данных, а также для получения статистики и анализа данных (рисунок 18).

`HomeController` – контроллер, обрабатывает запросы GET и PATCH. Контроллер использует классы `ModelMapper`, `PersonDetailsService` и `AuthenticationUtil` для конвертации данных, работы с пользователями и проверки аутентификации. Метод `allPerson` контроллера получает список пользователей с помощью сервиса `personDetailsService` и конвертирует его в список `PersonDTO`. Метод `updateRole` обновляет роль пользователя в системе. Контроллер также имеет методы для конвертирования `Person` в `PersonDTO` и наоборот.

`ParserController` – контроллер, который обрабатывает запросы, связанные с парсингом и обработкой данных. Он содержит методы для загрузки данных из внешних источников, обработки и сохранения данных в базе данных.

`PredictFloodController` – контроллер, который обрабатывает запросы, связанные с предсказанием наводнений. Он содержит методы для получения данных о погоде и водных ресурсах, а также для расчета и отображения прогнозов наводнений.

`RecoveryDataController` – контроллер, который обрабатывает запросы, связанные с восстановлением данных. Он содержит методы для восстановления данных из резервных копий, а также для очистки и обновления данных в базе данных.

`TrainFloodController` – контроллер, который обрабатывает запросы, связанные с обучением моделей для предсказания наводнений. Он содержит методы для загрузки и обработки данных, а также для обучения и сохранения моделей.

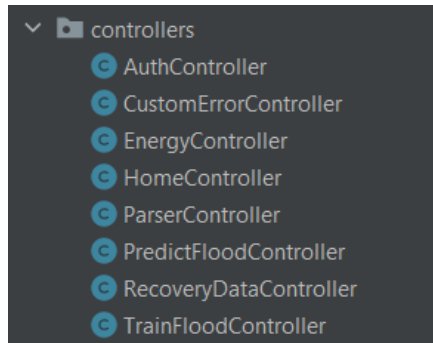


Рисунок 18 – controllers

2.1.3. DTO

В пакете dto (рисунок 19) находится класс, который предназначен для передачи информации о пользователе между различными частями приложения. Класс PersonDTO задает структуру, по которой будет передаваться информация о пользователе, и содержит только те поля, которые необходимы для отображения информации о пользователе, исключая пароль и роль.

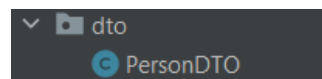


Рисунок 19 – DTO

2.1.4. Model

Complete – класс, который представляет объект, содержащий данные о погодных условиях на определенном посту.

Energy – класс, который представляет объект, содержащий данные о потребляемой электроэнергии и погодных условиях в определенный день. Класс Energy также использует аннотации Lombok и JPA, как и класс Complete. Атрибуты этого класса включают дату, потребляемую электроэнергию, температуру воздуха, количество осадков, выходной день или нет, атмосферное давление и сезон. Класс также имеет атрибут lastDate, который не сохраняется

в базе данных и используется только временно при выполнении определенных операций (рисунок 20).

Person — класс, который представляет объект, содержащий информацию о пользователе системы. Класс Person использует аннотации Lombok и JPA, а также аннотации Jakarta Validation для проверки входных данных. Атрибуты этого класса включают имя пользователя, электронную почту, пароль и роль пользователя в системе. Класс Person имеет несколько конструкторов, в том числе конструкторы для создания объектов с определенными параметрами. Класс Person также имеет пустой конструктор, который используется для создания нового объекта без параметров.

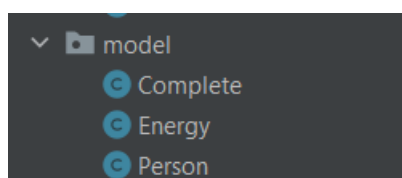


Рисунок 20 – Model

2.1.5. Network

Activation — класс, который содержит методы для активации нейронов в нейронной сети.

ATYPE — перечисление, которое содержит три различных типа функций активации, которые могут использоваться в нейронной сети.

DataActions — класс, который содержит методы для обработки и подготовки данных для использования в нейронной сети.

HiddenLayer — класс, который наследует абстрактный класс Layer и представляет собой скрытый слой нейронной сети. Он содержит массив нейронов, тип активации, массив дельт и методы для активации слоя и вычисления дельты.

InputLayer — класс, который является подклассом класса Layer и предназначен для реализации входного слоя в нейронной сети.

Layer — класс, который представляет собой абстрактный слой нейронной сети. Он содержит массив нейронов, количество нейронов и соединений, тип активации, массив дельт и матрицу приращений весов нейронов (рисунок 21).

MatrixActions — класс, который содержит методы для работы с матрицами, используемыми в нейронной сети.

Neuron — класс, который представляет собой нейрон в нейронной сети

Normalization — класс, который содержит методы для нормализации данных, используемых в нейронной сети.

OutputLayer — класс, который представляет собой выходной слой нейронной сети.

Predict — класс, который содержит методы для предсказания значений на основе обученной нейронной сети.

Train — класс, который содержит методы для обучения нейронной сети на основе обучающих данных.

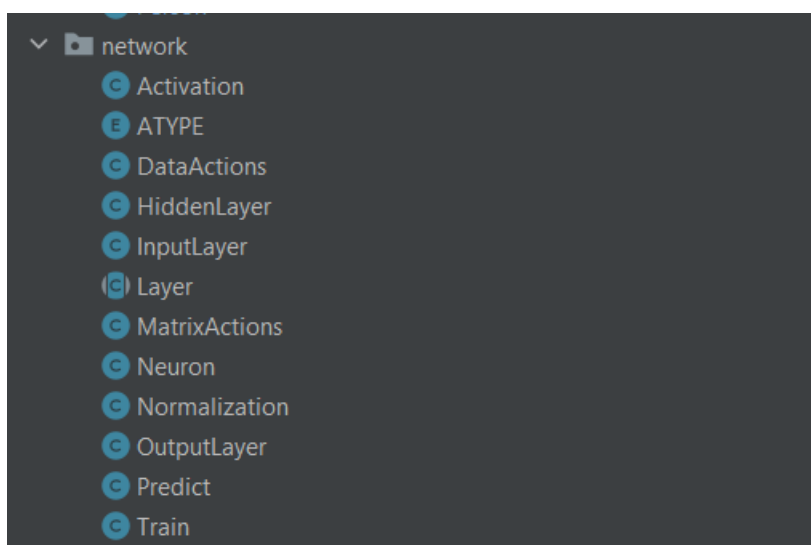


Рисунок 21 – Network

2.1.6. Parse

WaterParser — класс, который предназначен для парсинга данных, связанных с использованием воды.

XmlParserElectro — класс, который предназначен для парсинга данных в формате XML, связанных с использованием электроэнергии (рисунок 22).

XmlParsingWater — класс, который предназначен для парсинга данных в формате XML, связанных с использованием воды.

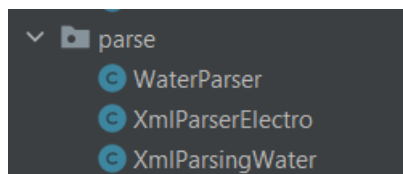


Рисунок 22 – Parse

2.1.7. Repositories

CompleteRepository — интерфейс, который представляет собой репозиторий для хранения и управления данными, связанными с энергетикой. В этом интерфейсе реализованы методы для получения, добавления, обновления и удаления данных, а также для выполнения запросов к данным.

EnergyRepository — интерфейс, который представляет собой репозиторий для хранения и управления данными, связанными с электропотреблением. В этом интерфейсе реализованы методы для получения, добавления, обновления и удаления данных, а также для выполнения запросов к данным (рисунок 23).

PeopleRepository — интерфейс, который представляет собой репозиторий для хранения и управления данными, связанными с пользователями. В этом интерфейсе реализованы методы для получения, добавления, обновления и удаления данных, а также для выполнения запросов к данным.

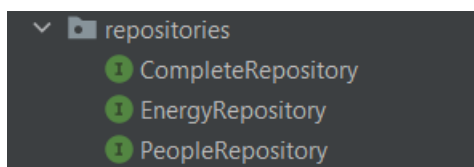


Рисунок 23 – repository

2.1.8. Security

AuthenticationUtil — класс, который предназначен для получения информации о текущем пользователе в системе.

PersonDetails — класс, который реализует интерфейс UserDetails из Spring Security. Класс PersonDetails используется для представления информации о пользователе, который аутентифицировался в системе. Класс получает объект Person, содержащий информацию о пользователе, в качестве параметра конструктора.

– метод getAuthorities возвращает коллекцию GrantedAuthority, которая определяет роли пользователя;

– метод getPassword возвращает пароль пользователя;

– метод getUsername возвращает email пользователя;

– методы isAccountNonExpired, isAccountNonLocked и isCredentialsNonExpired возвращают true, чтобы указать, что учетная запись пользователя не истекла и не заблокирована, а также что учетные данные пользователя не истекли;

– метод isEnabled также возвращает true, чтобы указать, что учетная запись пользователя активна;

Класс PersonDetails также содержит метод getPerson, который возвращает объект Person, связанный с пользователем.

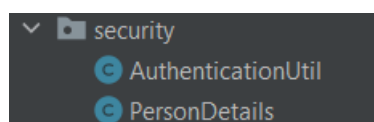


Рисунок 24 – Security

По своей сути это класс «обёртка» над классом Person, который просто расширяет его функционал за счет других методов (рисунок 24).

2.1.9. *Services*

`CompleteService` — класс, который предназначен для выполнения операций и прогнозирования данных. В этом классе реализованы методы для получения, прогнозирования, обновления и удаления данных, а также для выполнения запросов к данным (рисунок 25).

`EnergyServices` — класс, который предназначен для выполнения операций и прогнозирования данных, связанными с потреблением электроэнергии. В этом классе реализованы методы для получения, прогнозирования, обновления и удаления данных, а также для выполнения запросов к данным.

`PersonDetailsService` — класс, который предоставляет сервисы для работы с объектами класса `Person` и реализует интерфейс `UserDetailsService` из фреймворка `Spring Security`.

Класс `PersonDetailsService` содержит методы:

- `personList` — метод, который возвращает список всех объектов `Person`;
- `loadUserByUsername` — метод, который ищет пользователя по имени пользователя (`email`) в репозитории `PeopleRepository` и возвращает объект `PersonDetails`, содержащий информацию о найденном пользователе;
- `findByEmail` — метод, который ищет пользователя по `email` в репозитории `PeopleRepository` и возвращает объект `Person`;
- `update` — метод, который обновляет роль пользователя на `ROLE_ADMIN` в репозитории `PeopleRepository`.

`RegistrationService` — класс, который предоставляет сервисы для регистрации новых пользователей в системе.

Класс `RegistrationService` содержит метод `register`.

`Register` — метод, который принимает объект `Person`, кодирует его пароль с помощью `PasswordEncoder` и сохраняет нового пользователя в репозитории `PeopleRepository`.

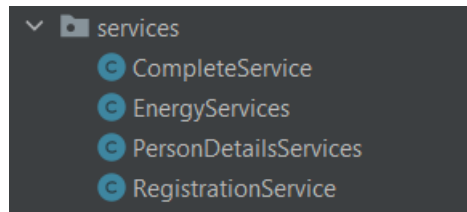


Рисунок 25 – Services

2.1.10. Static

Директория, которая содержит сохраненные модели после обучения нейронной сети, они добавляются динамически при каждом запросе пользователя.

2.1.11. Util

Util — класс, который предназначен для содержания служебных методов, которые могут использоваться в различных частях приложения (рисунок 26).

PersonValidator — класс, который используется для проверки валидности объектов типа Person в приложении. В данном случае класс PersonValidator проверяет наличие пользователя с таким же email в системе.

Класс PersonValidator содержит методы:

– supports — метод, который возвращает true, если данная валидация поддерживается классом Person;

– validate — метод, который получает объект типа Person и объект Errors, и проверяет наличие пользователя с таким же email в системе. Если пользователь с указанным email уже существует, то метод rejectValue вызывается для объекта Errors, чтобы отобразить сообщение об ошибке на форме.

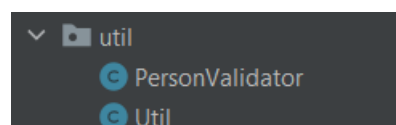


Рисунок 26 – Util

5.2. Клиент

Всего на стороне клиента используется 14 HTML файлов, 10 картинок для интерфейса, 9 файлов формата JSON с координатами городов, 7 файлов с кодом на JavaScript, 12 иконок, 6 файлов CSS (рисунок 27).

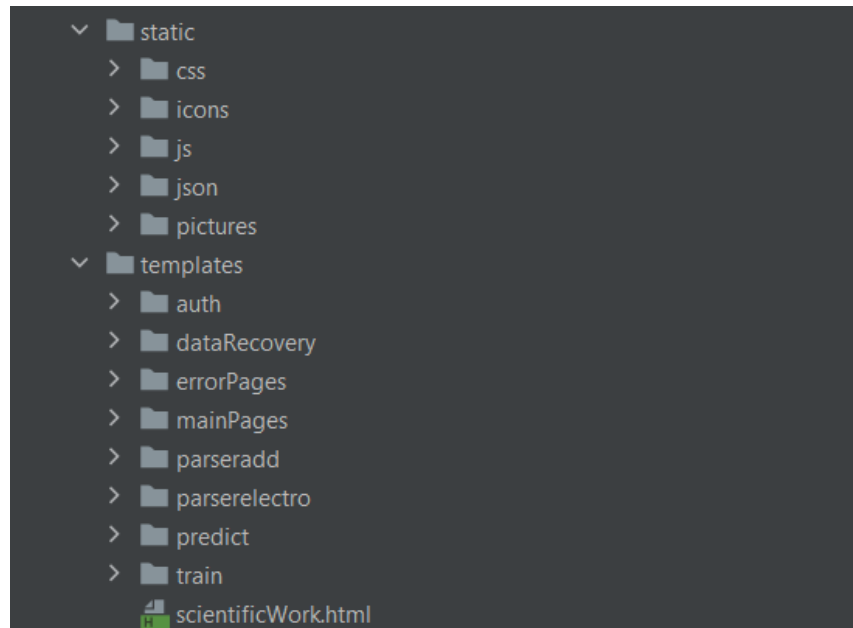


Рисунок 27 - Структура клиентской части

2.2.1. *Template*

Auth: эта папка содержит HTML-страницы, связанные с аутентификацией пользователей в приложении, login.html и registration.html. Эти страницы содержат формы для входа, регистрации и выхода из системы, а также соответствующие скрипты для обработки отправленных данных на сервере. Первая страница содержит форму регистрации, в которой пользователь может ввести свой адрес электронной почты, имя пользователя и пароль, а затем отправить форму для создания новой учетной записи. Вторая страница содержит форму входа, в которой пользователь вводит свой адрес электронной почты и пароль, чтобы получить доступ к своей учетной записи. Обе страницы

имеют сходный макет с заголовком, полями для ввода и кнопкой подтверждения.

DataRecovery: эта папка содержит HTML-страницы, связанные с восстановлением данных в приложении, `dataRecovery.html`. Эта страница содержит формы для восстановления данных, а также соответствующие скрипты для обработки отправленных данных на сервере.

ErrorPages: эта папка содержит HTML-страницы, связанные с обработкой ошибок в приложении. Например, `404.html`, `500.html` и т.д. Эти страницы отображаются, когда происходит ошибка в приложении, и содержат соответствующие сообщения об ошибке и инструкции для пользователя.

MainPages: эта папка содержит HTML-страницу, которая является основной страницей приложения. Код страницы содержит разные элементы, такие как меню, формы, изображения и текстовый контент. Он также содержит ссылки на другие страницы в приложении и информацию о контактах.

Parseradd: эта папка содержит HTML-страницу, связанную с парсингом данных, для добавления новых объектов в приложение. Эта страница содержит формы для добавления данных в приложение, а также соответствующие скрипты для обработки отправленных данных на сервере.

Parseelectro: эта папка содержит HTML-страницу, связанную с парсингом данных, связанных с энергетикой. Эта страница содержит формы для парсинга данных, связанных с электроникой, а также соответствующие скрипты для обработки отправленных данных на сервере.

Predict: эта папка содержит HTML-страницы, связанные с прогнозированием данных или событий. Код страниц позволяет пользователю выбрать даты для прогнозирования потребления. Страница содержит форму с двумя полями для ввода дат, а также поле для выбора столбцов, которые будут использоваться для обучения модели прогнозирования, доступно только для пользователей с ролью "ADMIN". После отправки формы на сервер, на странице отображается таблица с прогнозируемыми значениями потребления электроэнергии для выбранных дат, а также кнопки для скачивания файла Excel

с результатами и отображения графика. Также на странице есть кнопка для возврата на главную страницу приложения.

Train: эта папка содержит HTML-страницы, связанные с обучением моделей. Эти страницы содержат формы для загрузки и обработки данных, на основе которых будет проводиться обучение моделей, а также соответствующие скрипты для обработки отправленных данных на сервере и отображения результатов обучения.

ScientificWork.html: это HTML-страница, содержащая научную работу, связанную с тематикой приложения.

2.2.2. *Static*

В данной папке есть небольшая структура разделения (рисунок 28). Самая интересная часть данной директории заключается в папке js, которая содержит файлы с кодом на JavaScript. Сейчас её и разберем.

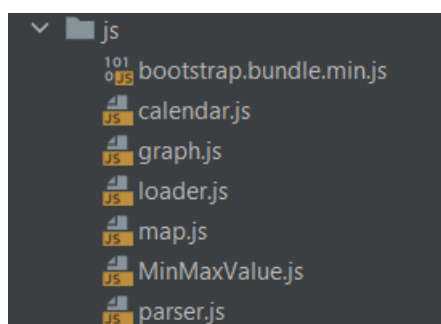


Рисунок 28 – структура файлов JavaScript

Эти файлы являются скриптами, которые используются в веб-приложении для реализации различных функций и возможностей.

Bootstrap.bundle.min.js—содержит минифицированный код библиотеки Bootstrap, которая предоставляет множество готовых компонентов интерфейса для создания адаптивных веб-страниц.

Calendar.js—скрипт, который используется для создания календарей на веб-страницах. В нем прописаны основные ограничения и условия выбора дат.

Graph.js—содержит код, который позволяет рисовать графики и диаграммы на веб-страницах с использованием библиотеки anychart.

Loader.js—содержит код, который используется для создания анимированных загрузчиков на веб-страницах.

Map.js — скрипт, который используется для создания интерактивной карты на веб-странице. Он использует данные из JSON файлов.

MinMaxValue.js — содержит код, который позволяет найти минимальное и максимальное значение в массиве чисел.

Parser.js — содержит код, который используется для страницы парсинга (анализа) данных.

Заключение

В результате выполнения курсовой работы была разработана нейросетевая система прогнозирования электроэнергии с использованием фреймворка Spring и создано соответствующее веб-приложение. Основной целью проекта было разработать систему, позволяющую прогнозировать потребление электроэнергии на основе анализа исторических данных и предоставляющую пользователям доступ к актуальным данным в режиме реального времени.

Для достижения этой цели были выполнены следующие задачи: разработка клиент-серверного приложения на языке Java, создание серверной части приложения на основе фреймворка Spring Boot, разработка клиентской части приложения с графическим интерфейсом с помощью HTML, CSS и JavaScript, применение модели MVC для разделения управляющей логики на отдельные компоненты и создание кроссплатформенного приложения.

В результате работы была получена эффективная и удобная система, позволяющая прогнозировать потребление электроэнергии на основе анализа исторических данных и предоставляющая пользователям доступ к актуальным данным в режиме реального времени. Разработанное приложение может быть использовано в современной энергетике для более точного прогнозирования потребления электроэнергии и оптимизации производственных процессов.

Также, во время выполнения курсовой работы были получены навыки разработки нейросетевых систем и создания веб-приложений на основе фреймворка Spring. Эти навыки могут быть применены в различных областях, где требуется разработка систем прогнозирования на основе анализа исторических данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Brown, R.E. Electricity use in California: past trends and present usage patterns / R.E. Brown // *Energy Policy*. – Vol. 31(9). – Elsevier, 2003. – pp. 849–864.
2. Y. He Probability density forecasting of wind power using quantile regression neural network and kernel density estimation *Energy Convers Manage* / Y. He // *Energy Convers Manage*. – Volume 164. – Elsevier, 2018. – pp. 374-384.
3. Q. Xu Composite quantile regression neural network with applications / Q. Xu // *Expert Syst Appl*. – Volume 76. – Elsevier, 2017. – pp 129-139.
4. A.J. Cannon Quantile regression neural networks: implementation in R and application to precipitation downscaling/ A.J. Cannon // *Comput Geosci*. – Volume 46. – Elsevier, 2012. – p 9.
5. D. Pradeepkumar Forecasting financial time series volatility using particle swarm optimization trained quantile regression neural network / D. Pradeepkumar // *Appl Soft Comput*. – Volume 58. – Elsevier, 2017. – pp 35-52.
6. J.X. Zhang Prognostic and predictive value of a microrna signature in stage ii colon cancer: a microrna expression analysis / J.X. Zhang // *Lancet Oncol*. – Volume 14, Issue 13. – Elsevier, 2013. – pp 1295-1306.
7. G.M. Huebner Explaining domestic energy consumption - the comparative contribution of building factors, socio-demographics, behaviours and attitudes / G.M. Huebner // *Appl Energy*. – Volume 159. – Elsevier, 2015. – pp 589-600.
8. F. Ziel Forecasting wind power - modeling periodic and non-linear effects under conditional heteroscedasticity / F. Ziel // *Appl Energy*. – Volume 177. – Elsevier, 2015. – pp 285-297.
9. Y. He Short-term power load probability density forecasting method using kernel-based support vector quantile regression and copula theory / Y. He // *Appl Energy*. – Volume 185, Part 1. – Elsevier, 2017. – pp 254-266.
10. Y. He Short-term power load probability density forecasting based on quantile regression neural network and triangle kernel function / Y. He // *Energy*. – Volume 114. – Elsevier, 2016. – pp 498-512.

11. V. Bianco Electricity consumption forecasting in Italy using linear regression models / V. Bianco // Energy. – Volume 34, Issue 9 . – Elsevier, 2019. – pp 1413-1421.

12. D. van der Meer Probabilistic forecasting of electricity consumption, photovoltaic power generation and net demand of an individual building using Gaussian processes / D. van der Meer // Appl Energy. – Volume 213 . – Elsevier, 2018. – pp 195-207.

13. Walls, C. Spring в действии / C. Walls. - М.: ДМК Пресс, 2019. - 616 с.

14. Sharma, R., Chopra, K. Spring 5 для профессионалов / R. Sharma, K. Chopra. - М.: Питер, 2018. - 624 с.

15. Long, J. Spring Boot в действии / J. Long. - М.: ДМК Пресс, 2018. - 432 с.

Приложения

Приложение А – Фрагмент исходного класса, отвечающего за предоставление доступа к веб-инструментам

```
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class MvcConfig implements WebMvcConfigurer {

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry
            .addResourceHandler(...pathPatterns: "/static/**")
            .addResourceLocations("classpath:/static/");
    }
}
```

Приложение Б – Фрагмент исходного класса, отвечающего за цепочку безопасности приложения

```
@Configuration
@EnableMethodSecurity
@AllArgsConstructor
public class SecurityConfig {

    1 usage
    private final PersonDetailsService personDetailsService;

    2 Alexandr Pyatunin +1
    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http.authorizeHttpRequests()
            .authorizeHttpRequests(configurer->configurer.requestMatchers(
                ...patterns: "/scientificWork", "/", "/auth/registration", "/auth/login").permitAll()
                .requestMatchers(...patterns: "/css/**", "/js/**", "/fonts/**", "/pictures/**", "/icons/**", "/json/**").permitAll()
                .requestMatchers(...patterns: "/train").hasRole("ADMIN")
                .anyRequest().hasAnyRole(...roles: "USER", "ADMIN")
            )
            .and()
            .formLogin()
            .loginPage("/auth/login")
            .loginProcessingUrl("/process_login")
            .defaultSuccessUrl(...defaultSuccessUrl: "/", alwaysUse: true)
            .failureUrl(...authenticationFailureUrl: "/auth/login?error")
            .and()
            .logout()
            .logoutUrl("/logout")
            .logoutSuccessUrl("/?logout")
            .and()
            .exceptionHandling()
            .accessDeniedPage(...accessDeniedUrl: "/errorPages/error");

        return http.build();
    }
}
```

Приложение Б – Фрагмент исходного класса, отвечающего за предоставление
пользователю доступа к интерфейсу
(продолжение)

```
Alexandr Pyatunin
@Bean
public AuthenticationManager authenticationManager() throws Exception {
    return new ProviderManager(Collections.singletonList(authenticationProvider()));
}

1 usage Alexandr Pyatunin
@Bean
public AuthenticationProvider authenticationProvider() {
    DaoAuthenticationProvider provider = new DaoAuthenticationProvider();
    provider.setUserDetailsService(personDetailsService);
    provider.setPasswordEncoder(passwordEncoder());
    return provider;
}

1 usage Alexandr Pyatunin
@Bean
public PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }
```

Приложение В – Фрагмент исходного класса, отвечающего за предсказание нейронной сети

```
public float[] predict(Date FirstDate, Date LastDate, List<String> selectedName) {
    List<Energy> full_energy = energyRepository.findAll();
    List<Energy> predict_energy = energyRepository.findAllLastDateTimeAndLimit(FirstDate, Util.INPUT_NEURONS);
    System.out.println(predict_energy);
    makeModelName(selectedName);
    Util.error = 600;
    Predict predict = new Predict(fillList(full_energy, selectedName),
        fillList(predict_energy, selectedName));
    float[] predictionDL = predict.makePrediction();
    float maeDL = predict.getMAE();
    float accuracyDL = predict.getAccuracy();
    System.out.println();
    System.out.println("MAE DL");
    System.out.println(maeDL);
    System.out.println("accuracy DL");
    System.out.println(accuracyDL);
    System.out.println(Arrays.toString(predictionDL));
    SimpleDateFormat dateFormat = new SimpleDateFormat(pattern: "yyyy-MM-dd");
    Object obj = predictML(dateFormat.format(FirstDate), dateFormat.format(LastDate));
    if (obj == null){
        return predictionDL;
    }
    float[] MLResults = (float[]) obj;
    float maeML = MLResults[0];
    float accuracyML = MLResults[1];
    float[] predictionML = new float[MLResults.length - 2];
    for (int i = 2; i < MLResults.length; i++) {
        predictionML[i-2] = MLResults[i];
    }
    System.out.println();
    System.out.println("MAE ML");
    System.out.println(maeML);
    System.out.println("accuracy ML");
    System.out.println(accuracyML);
    System.out.println(Arrays.toString(predictionML));
    return accuracyDL > accuracyML ? predictionDL : predictionML;
}
```


Приложение Г – Фрагмент исходного класса, отвечающего за определения роли текущего пользователя

```
@Component
public class AuthenticationUtil {

    7 usages  Alexandr Pyatunin
    public String getCurrentRole() {
        Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
        Object principal = authentication.getPrincipal();
        if (principal instanceof PersonDetails personDetails) {
            System.out.println(personDetails.getPerson());
            return personDetails.getPerson().getRole();
        } else {
            return null;
        }
    }
}
```

Приложение Д – Фрагмент исходного класса, отвечающего за обработку ошибок при регистрации

```
@Component
public class PersonValidator implements Validator {

    2 usages
    private final PersonDetailsService personDetailsService;

    Alexandr Pyatunin
    public PersonValidator(PersonDetailsService personDetailsService) {
        this.personDetailsService = personDetailsService;
    }

    Alexandr Pyatunin
    @Override
    public boolean supports(Class<?> clazz) { return Person.class.equals(clazz); }

    Alexandr Pyatunin
    @Override
    public void validate(Object target, Errors errors) {
        Person person = (Person) target;
        try {
            personDetailsService.loadUserByUsername(person.getEmail());
        } catch (UsernameNotFoundException ignored){
            return;
        }
        errors.rejectValue( field: "email", errorCode: "", defaultMessage: "Человек с таким email существует");
    }
}
```