Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Программирование сетевых приложений»

|  |  |
| --- | --- |
|  | «К защите допустить»  Руководитель курсовой работы  ассистент кафедры ЭИ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Л. А. Снопок  \_\_\_.\_\_\_.2024 |

**Пояснительная записка**

к курсовой работе

на тему:

**«Разработка автоматизированной системы построения прогноза погоды на основе статистических наблюдений»**

БГУИР КР 1-40 05 01-12 003 ПЗ

|  |  |
| --- | --- |
|  | Выполнил студент группы 224402  Бурая Александра Константиновна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента)  Курсовая работа представлена на проверку \_\_\_.\_\_\_.2024  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2024

**Реферат**

БГУИР КР 1-40 05 01-12 003 ПЗ

**Разработка автоматизированной системы построения прогноза погоды на основе статистических наблюдений** : курсовая работа / А. К. Бурая. – Минск : БГУИР, 2024, – п.з. – 52 с..

Пояснительная записка 52 с., 30 рис., 10 источников, 4 приложения.

**Ключевые слова**: автоматизированная система, прогнозирование погоды, обработка метеорологических данных, численные модели, статистический анализ.

*Целью курсовой работы* является сокращение времени обработки метеорологических данных за счет разработки автоматизированной системы прогнозирования на основе статистических наблюдений. Это позволит уменьшить трудозатраты метеорологов, повысить точность прогнозов и автоматизировать процесс обработки данных.

*Объект исследования* – процесс прогнозирования погоды.

*Предмет исследования* – модели и методы построения прогноза погоды.

*Методология проведения работы*: в процессе разработки системы использованы современные подходы к обработке данных, функциональный анализ процессов, моделирование системы с помощью UML-диаграмм.

*Результаты работы*: были рассмотрены основные модели и методы прогнозирования погоды. Был проведен анализ методов обработки метеорологических данных, разработаны UML-диаграммы и блок-схемы для описания структуры и работы системы, протестирована система с использованием реальных данных, а также создано руководство по развертыванию приложения. Полученная система позволяет сократить трудозатраты, повысить качество прогнозирования и минимизировать ошибки при обработке данных.

**Содержание**

[Введение 6](#_Toc184915190)

[1 Анализ и моделирование процессов автоматизированного прогнозирования погоды 7](#_Toc184915191)

[1.1 Модели и методы прогнозирования погоды 7](#_Toc184915192)

[1.2 Разработка функциональной модели предметной области 8](#_Toc184915193)

[2 Проектирование и конструирование программного средства 13](#_Toc184915194)

[2.1 Постановка задачи и обзор методов ее решения 13](#_Toc184915195)

[2.2 UML-модели представления программного обеспечения 13](#_Toc184915196)

[2.3 Информационная модель системы и её описание 21](#_Toc184915197)

[2.4 Описание алгоритмов, реализующих бизнес-логику серверной части проектируемой системы 22](#_Toc184915198)

[2.5 Разработка схем алгоритмов программы 24](#_Toc184915199)

[3 Тестирование и проверка работоспособности программного средства 28](#_Toc184915200)

[4 Инструкция по развертыванию приложения 32](#_Toc184915201)

[4.1 Авторизация 32](#_Toc184915202)

[4.2 Модуль пользователя 33](#_Toc184915203)

[4.3 Модуль сотрудника 35](#_Toc184915204)

[4.4 Модуль администратора 39](#_Toc184915205)

[Заключение 43](#_Toc184915206)

[Список использованных источников 44](#_Toc184915207)

[Приложение А (обязательное) Отчет о проверке на заимствования в системе «Антиплагиат» 45](#_Toc184915208)

[Приложение Б (обязательное) Листинг кода алгоритмов, реализующих основную бизнес-логику приложения 46](#_Toc184915209)

[Приложение В (обязательное) Листинг основных элементов программы 49](#_Toc184915210)

[Приложение Г (обязательное) Листинг скрипта генерации базы данных 51](#_Toc184915211)

# Введение

Человек сталкивается с необходимостью учитывать погодные условия практически ежедневно: будь то планирование работы, отдыха или повседневной деятельности. Прогнозы погоды позволяют не только узнавать о возможных изменениях климата, но и минимизировать последствия различных природных явлений.

В современном мире всё больше людей обращаются к автоматизированным системам для получения актуальных данных о погоде в любом регионе мира. Развитие технологий позволяет создавать информационные системы, которые собирают, обрабатывают и анализируют метеорологические данные.

Целью данного курсового проекта является сокращение времени обработки метеорологических данных за счет разработки автоматизированной системы построения прогноза на основе статистических наблюдений. Это позволит уменьшить трудозатраты метеорологов на обработку больших массивов данных; повысить качество прогноза за счет использования современных статистических методов и численных моделей; увеличить степень автоматизации процесса, что сократит риск ошибок при ручной обработке данных и их интерпретации.

Для достижения поставленной цели необходимо решить следующие задачи:

* проанализировать особенности обработки метеорологических данных и изучить статистические методы наблюдений;
* разработать UML диаграммы и блок-схемы программы;
* провести тестирование системы на реальных данных;
* предусмотреть инструкцию по развертыванию программы;

Объектом исследования является процесс прогнозирования погоды. Предмет исследования: модели и методы построения прогноза погоды. Таким образом, разработка данной системы является актуальной и важной задачей, направленной на повышение точности и доступности прогнозов погоды.

Курсовая работа выполнена самостоятельно, проверена в системе «Антиплагиат». Процент оригинальности составляет 96,4%. Цитирования обозначены ссылками на публикации, указанными в «Списке использованных источников». Скриншот приведен в приложении А (рисунок А.1).

# 1 Анализ и моделирование процессов автоматизированного прогнозирования погоды

## 1.1 Модели и методы прогнозирования погоды

Прогнозирование погоды включает сбор, анализ и интерпретацию данных о состоянии атмосферы для предсказания изменений погодных условий за определённый период. Предметная область, связанная с прогнозом погоды, охватывает широкий спектр метеорологических дисциплин и инструментов, от сбора данных до применения математических моделей.

Процесс создания прогноза погоды начинается с получения текущих данных о состоянии атмосферы. Они поступают из разных источников, таких как метеорологические станции, которые собирают данные о температуре, влажности, давлении и скорости ветра; спутники, которые отслеживают облачность, осадки, движение воздушных масс и состояние земной поверхности; метеорологические шары (радиозонды): эти устройства запускаются в атмосферу и собирают данные о её состоянии на различных высотах и другие.

Затем собранные данные очищаются и калибруются, чтобы исключить возможные ошибки измерений и пробелы в данных. На этом этапе выполняется интерполяция данных: заполнение пробелов между отдельными точками измерений, чтобы получить более полную картину состояния атмосферы.

После этапа очистки и интерполяции данных важным шагом в метеорологическом прогнозировании становится выбор методов анализа, которые позволяют предсказывать будущие погодные условия. Среди них можно выделить метод аналогов, регрессионный метод и вероятностный подход. Они основываются на обработке исторических данных и построении зависимостей между текущими погодными условиями и их возможными изменениями в будущем [1].

Метод аналогов основан на предположении о том, что если текущие погодные условия похожи на те, которые уже наблюдались в прошлом, то и дальнейшее развитие погоды будет аналогичным. Метод полезен для долгосрочных прогнозов, так как он требует сравнительно небольших вычислительных ресурсов и часто демонстрирует точность, сопоставимую с численными моделями. Однако его ограничивает зависимость от качества и объёма исторических данных, особенно в случаях редких явлений [2].

В регрессионном методе анализируются зависимости между различными погодными параметрами. Он позволяет строить прогнозы температуры, влажности и других характеристик на основе математических моделей. На основе текущих данных оцениваются вероятные изменения каждого параметра в будущем. Преимущество этого подхода заключается в его универсальности: регрессионные модели могут быть адаптированы под разные климатические зоны и типы данных [3].

Вероятностный метод применяется для анализа неопределённых событий, таких как осадки или экстремальные температуры. Он позволяет оценить вероятность того или иного явления, предлагая сценарии развития ситуации. Этот подход особенно ценен при прогнозировании редких или непредсказуемых явлений. Например, с его помощью можно оценить вероятность наводнений или штормов, основываясь на предыдущих случаях и текущих условиях. Вероятностные модели активно используются для уменьшения рисков в таких областях, как сельское хозяйство, авиация и энергетика [4].

Выбор подхода зависит от поставленных целей, доступных данных и временного отрезка прогноза, а комбинация методов позволяет достичь максимальной точности и надёжности прогноза.

На основе расчетов и статистических методов формируется прогноз погоды. Он включает в себя такие параметры, как температура воздуха, влажность, атмосферное давление, скорость и направление ветра, вероятность осадков, облачность и ожидаемые погодные явления (дождь, снег, град, гроза).

После создания прогноза проводится оценка его точности на основе сравнения с фактическими погодными условиями. Этот анализ помогает улучшить модели и алгоритмы прогнозирования, что позволяет со временем повысить их точность.

Таким образом, прогнозирование погоды представляет собой сложный процесс, объединяющий современные технологии, математическое моделирование и анализ больших объемов данных. Использование различных методов и моделей, позволяет адаптировать прогнозирование к различным климатическим условиям и временным рамкам, обеспечивая высокую точность и надёжность прогнозов погоды.

## 1.2 Разработка функциональной модели предметной области

Основным фактором, обеспечивающим разработку качественного программного продукта, является понимание логики процессов обработки и анализа информации. Для этого использована методология функционального моделирования IDEF0.

Далее представлена контекстная модель информационной системы. На контекстной диаграмме представлен один функциональный блок «Составить прогноз погоды». Были определены данные входа, механизма, контроля и выхода.

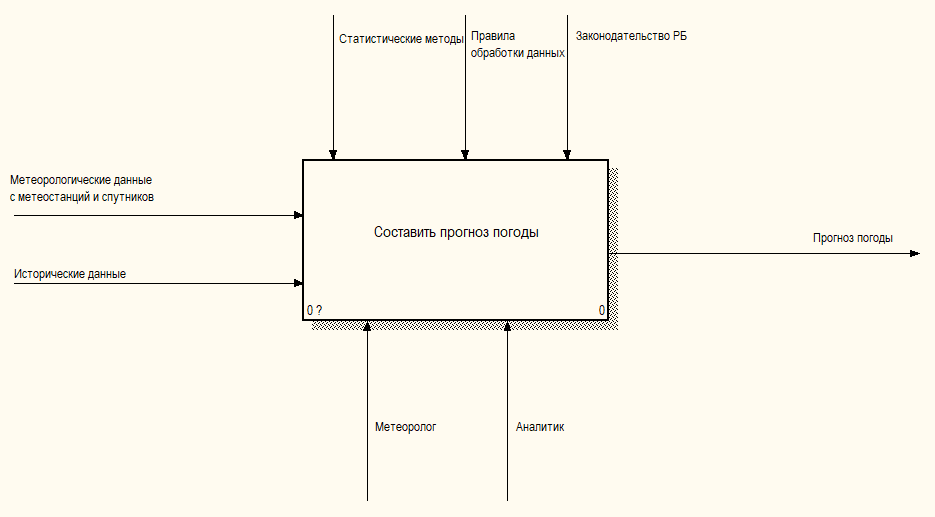


Рисунок 1.1 – Контекстная диаграмма

Контекстная диаграмма содержит следующие параметры:

* входные параметры: метеорологические данные с метеостанций и спутников, исторические данные;
* выходной параметр: прогноз погоды;
* управляющие параметры: законодательство РБ, статистические методы, правила обработки данных;
* исполнительные параметры: метеоролог, аналитик.

Диаграмма декомпозиции процесса «Составить прогноз погоды» представлена на рисунке 1.2.

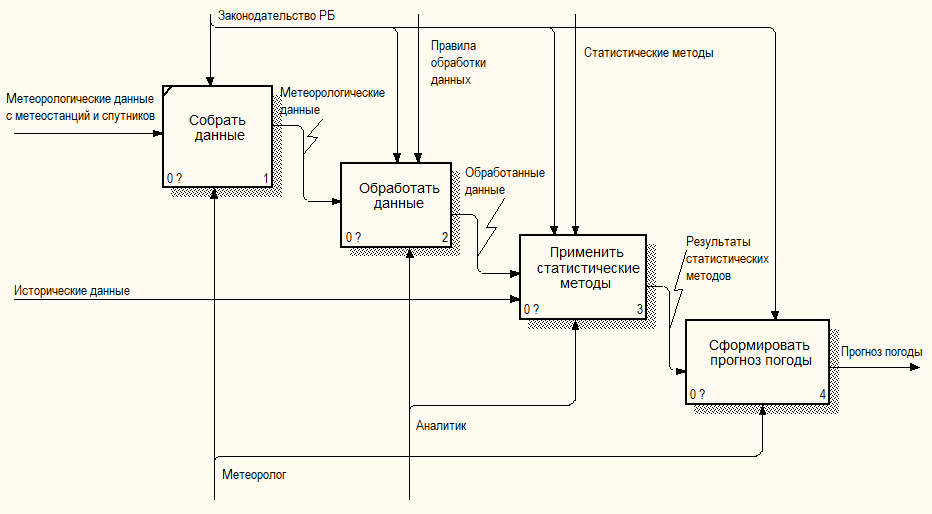


Рисунок 1.2 – Диаграмма декомпозиции

Эта диаграмма состоит из четырех функциональных блоков: «Собрать данные», «Обработать данные», «Применить статистические методы», «Сформировать прогноз погоды».

Чтобы более детально описать процесс «Обработать данные», производится декомпозиция второго уровня (см. рисунок 1.3).

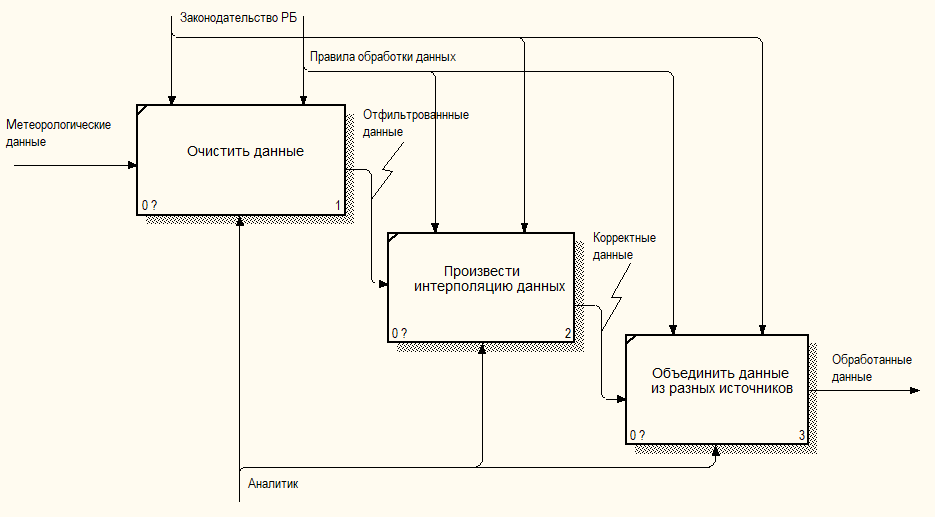


Рисунок 1.3 – Диаграмма декомпозиции процесса «Обработать данные»

Эта диаграмма состоит из трех функциональных блоков: «Очистить данные», «Произвести интерполяцию данных», «Объединить данные из разных источников».

Во время очистки данных происходит процесс фильтрации ошибочных, недостоверных или некорректных данных. Затем в ходе интерполяции заполняются пробелы в массивах данных и рассчитываются значения в промежуточных точках. После этого происходит объединение данных из различных источников в единую информационную базу.

Для более детального представления процесса «Применить статистические методы», происходит декомпозиция второго уровня (см. рисунок 1.4).

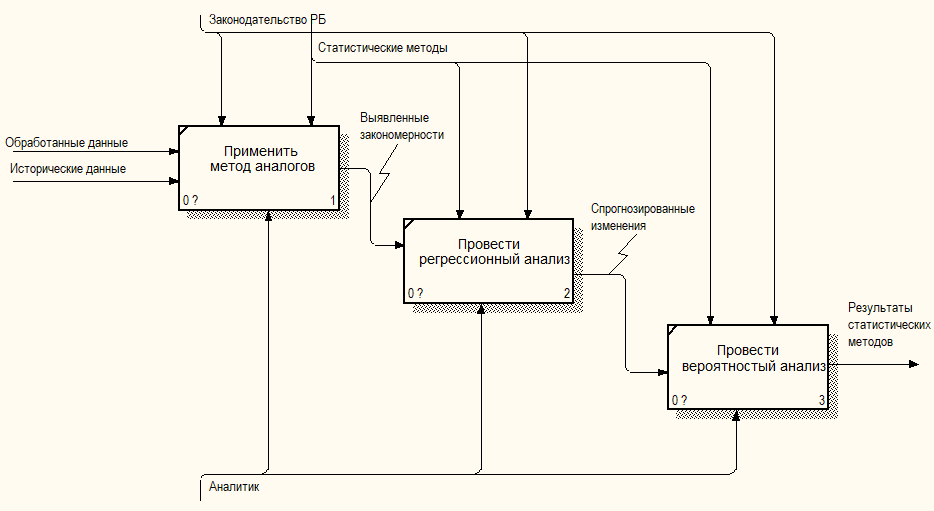


Рисунок 1.4 – Диаграмма декомпозиции процесса «Применить статистические методы»

Эта диаграмма состоит из трех функциональных блоков: «Применить метод аналогов», «Провести регрессионный анализ», «Провести вероятностный анализ».

Метод аналогов позволяет выявить закономерности во время сравнения текущих метеорологических данных с историческими. Затем используется регрессионная модель для прогнозирования изменений параметров погоды на основе их взаимосвязи. После чего вычисляется вероятность наступления определённых погодных явлений (например, осадки, грозы) на основе статистических закономерностей.

Чтобы более детально описать процесс «Сформировать прогноз погоды», происходит декомпозиция второго уровня (см. рисунок 1.5).

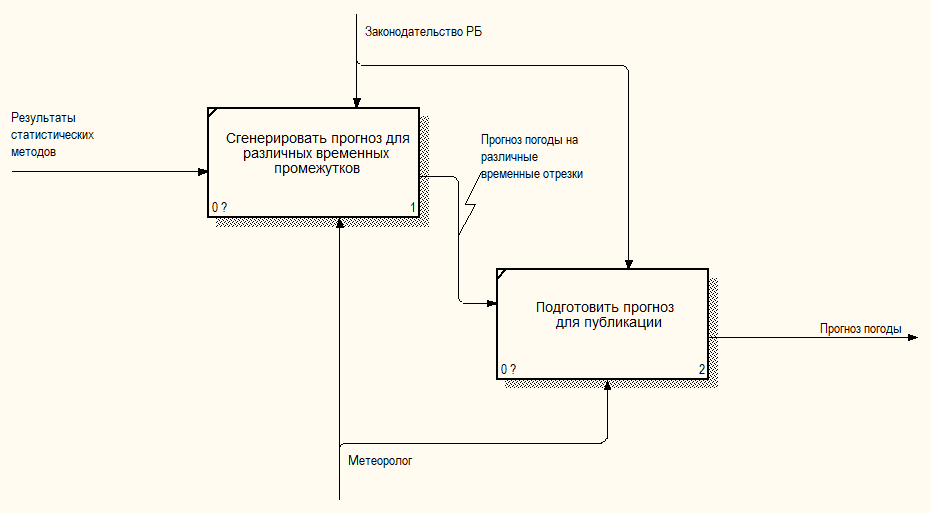


Рисунок 1.5 – Диаграмма декомпозиции процесса «Сформировать прогноз погоды»

Эта диаграмма состоит из двух функциональных блоков: «Сгенерировать прогноз для различных временных промежутков», «Подготовить прогноз для публикации».

После получения результатов статистических методов, создается прогноз на краткосрочные, среднесрочные и долгосрочные периоды (от нескольких часов до недель). Затем он обрабатывается для публикации. В результате этого процесса формируется готовый прогноз погоды на определенный период времени.

Таким образом, применение функционального моделирования способствует эффективному проектированию информационных систем, упрощает понимание их логики и обеспечивает надёжную основу для последующей разработки программного обеспечения.

# 2 Проектирование и конструирование программного средства

## 2.1 Постановка задачи и обзор методов ее решения

Главной задачей была разработка приложения в архитектуре клиент-сервер с многопоточным сервером и организацией взаимодействия с базой данных на объектно-ориентированном языке Java.

С данным программным средством должны работать следующие группы пользователей:

‒ пользователи (пользователи с ограниченными правами);

‒ сотрудники (пользователи с расширенными правами);

‒ администраторы.

Модуль администратора должен включать следующие функции:

‒ редактирование прав доступа пользователей;

‒ просмотр подключенных пользователей.

Модуль сотрудника должен включать следующие функции:

‒ изменение личных данных;

‒ просмотр прогноза погоды на текущую дату;

‒ просмотр календаря;

‒ редактирование БД;

‒ просмотр графиков.

Модуль студента включает в себя следующие подмодули:

‒ изменение личных данных;

‒ просмотр прогноза погоды на текущую дату;

‒ просмотр календаря.

После анализа поставленных задач были выбраны следующие средства их решения. В качестве среды для разработки была выбрана Intelij IDEA Ultimate 2024. Для работы с базами данных – MySQL 8.0. Для построения диаграмм IDEF0 была выбрана программа AllFusion Process Modeler 7, а для построения UML-диаграмм был выбран онлайн-сервис draw.io.

Для разработки приложения выбраны сокеты и протокол TCP/IP для передачи данных между клиентом и сервером, стандартная библиотека пользовательского интерфейса JavaFX для создания клиентского оконного приложения, JDBC для упрощения работы сервера с базой данных, паттерны проектирования, такие как MVC, strategy, singleton, data access object [5].

## 2.2 UML-модели представления программного обеспечения

UML является графическим языком для визуализации, конструирования и документирования систем, в которых большая роль принадлежит программному обеспечению. С его помощью разрабатывается детальный план создаваемой системы, содержащей множество различных элементов, связанных друг с другом [6].

Диаграмма вариантов использования представляет совокупность прецедентов и актеров. Они описывают систему с точки зрения возможностей ее использования.

Система содержит следующие роли пользователей: администратор, пользователь, сотрудник. Общими функциями для всех ролей являются авторизация, регистрация и восстановление пароля.

Администратор имеет следующие возможности использования системы:

* изменение личных данных;
* просмотр подключенных пользователей;
* просмотр данных пользователей;
* управление правами доступа пользователей.

Пользователь имеет следующие возможности использования системы:

* персональные настройки и изменение личных данных;
* просмотр прогноза погоды на сегодня;
* просмотр прогноза погоды на месяц;
* просмотр погоды за выбранный день;
* просмотр графика изменения температуры за выбранный период;
* сохранение данных в БД.

Сотрудник имеет все возможности пользователя. Уникальными функциями являются:

* CRUD операции с БД погоды;
* просмотр данных из БД;
* просмотр графиков изменения температуры, давления, скорости ветра за месяц;
* получение минимальных, максимальных и средних значений параметров погоды;
* сохранение данных в файл;
* просмотр диаграммы прогнозов погоды за месяц;
* поиск по дате, прогнозу и региону.

На рисунке 2.1 представлена диаграмма вариантов использования системы.

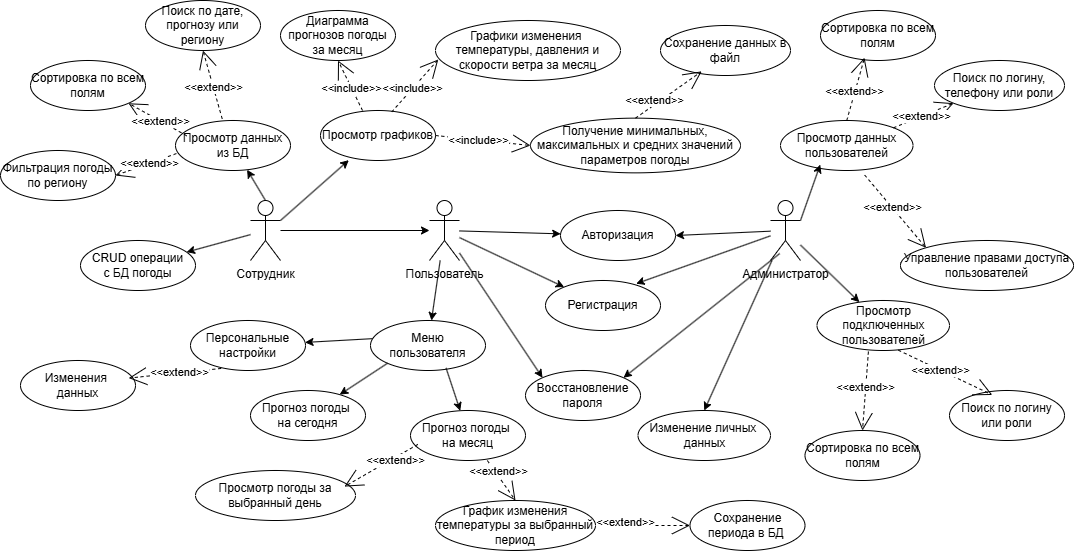


Рисунок 2.1 – Диаграмма вариантов использования

Диаграмма классов является основным логическим представлением модели и содержит детальную информацию об архитектуре программы.

Классы и объекты системы описываются следующим образом. Центральным классом является класс Day. Он содержит следующие данные: дата, регион, дневные параметры погоды, ночные параметры погоды и сам прогноз. Класс Location содержит название города и страны. Класс WeatherParameters содержит параметры погоды для каждого дня, такие как температура, давление, влажность, вероятность осадков и скорость ветра. Класс WeatherName содержит описание прогноза погоды (например, ясно, облачно, мокрый снег).

Класс User содержит данные пользователя: логин, пароль, роль и личные настройки. А класс PersonalSettings отображает настройки единиц измерения для температуры (°C/°F), давления (мм рт. ст./гПа) и скорости ветра (м/с, км/ч); и такие данные пользователя, как телефон и информация о рассылке погоды (включена или выключена).

Класс Dashbord содержит данные о сохраненных периодах просмотра погоды (на 3 дня, неделю, месяц) и объект класса User, чтобы сохранить период для конкретного пользователя. Диаграмма представлена на рисунке 2.2.

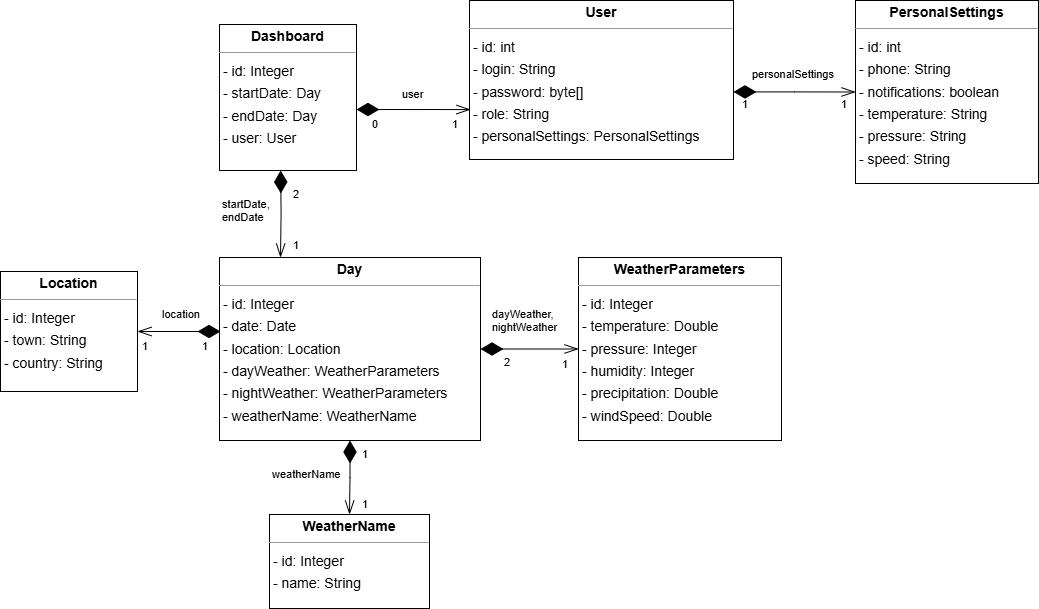


Рисунок 2.2 – Диаграмма классов

Диаграммы последовательностей моделируют взаимодействия между объектами в едином сценарии использования. Они иллюстрируют, как различные части системы взаимодействуют друг с другом для выполнения функции, а также порядок, в котором происходит взаимодействие при выполнении конкретного случая использования.

На рисунке 2.3 представлена диаграмма последовательностей процесса обновления календаря. На данной диаграмме изображены сообщения между используемыми классами (меню пользователя, класс обработки ошибок и интерфейс обработки ошибок). Сначала на выбранный месяц, год и регион создается календарь с подходящим количеством кнопок, равным количеству дней. Затем на каждый день устанавливается текст, отображающий дату и температуру днём и ночью в этот день. При нажатии на кнопку инициализируется поле с подробным описанием всех параметров погоды в этот день. Затем обновляется график температуры, чтобы период изменения показателей начинался с выбранной даты. При возникновении ошибок создается объект класса LabelErrorStrategy, который отображает ошибку на экране пользователя.

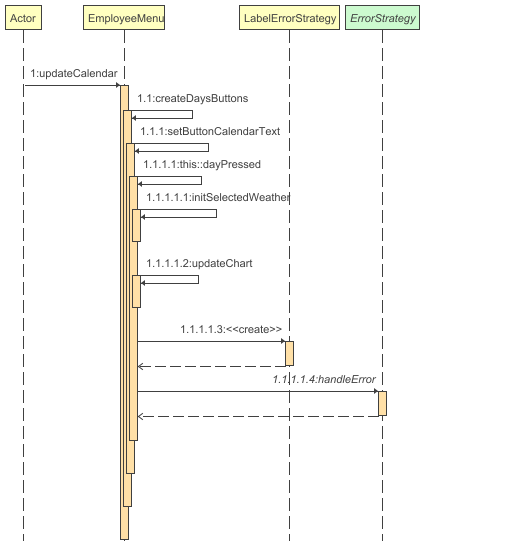


Рисунок 2.3 – Диаграмма последовательностей

Диаграммы состояний используются для моделирования поведения классов. Они акцентируют внимание на поведении объекта, которое зависит от последовательности событий.

На рисунке 2.4 представлена диаграмма состояний в модуле клиента, которая отображает процесс выбора временного отрезка для просмотра графика изменения температуры с последующей возможностью сохранить данные в БД.

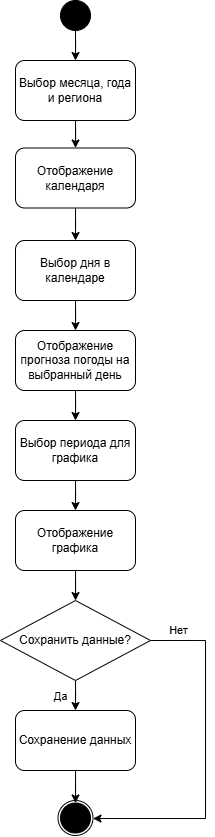


Рисунок 2.4 – Диаграмма состояний

Диаграммы компонентов используются для представления организации совокупности компонентов и существующих между ними зависимостей. Данный тип диаграмм описывает систему с точки зрения реализации.

Программа реализуется на трех компонентах – база данных, программа сервера и программа клиента. На рисунке 2.5 представлена диаграмма компонентов модуля сервера.

Она реализуется в четырёх основных пакетах приложения: Utilities используется для связи с клиентов и БД, Models (TCP – для обработки запросов, Entities – для POJO классов), DAO – для связи с БД, Service – для связи с DAO слоем.

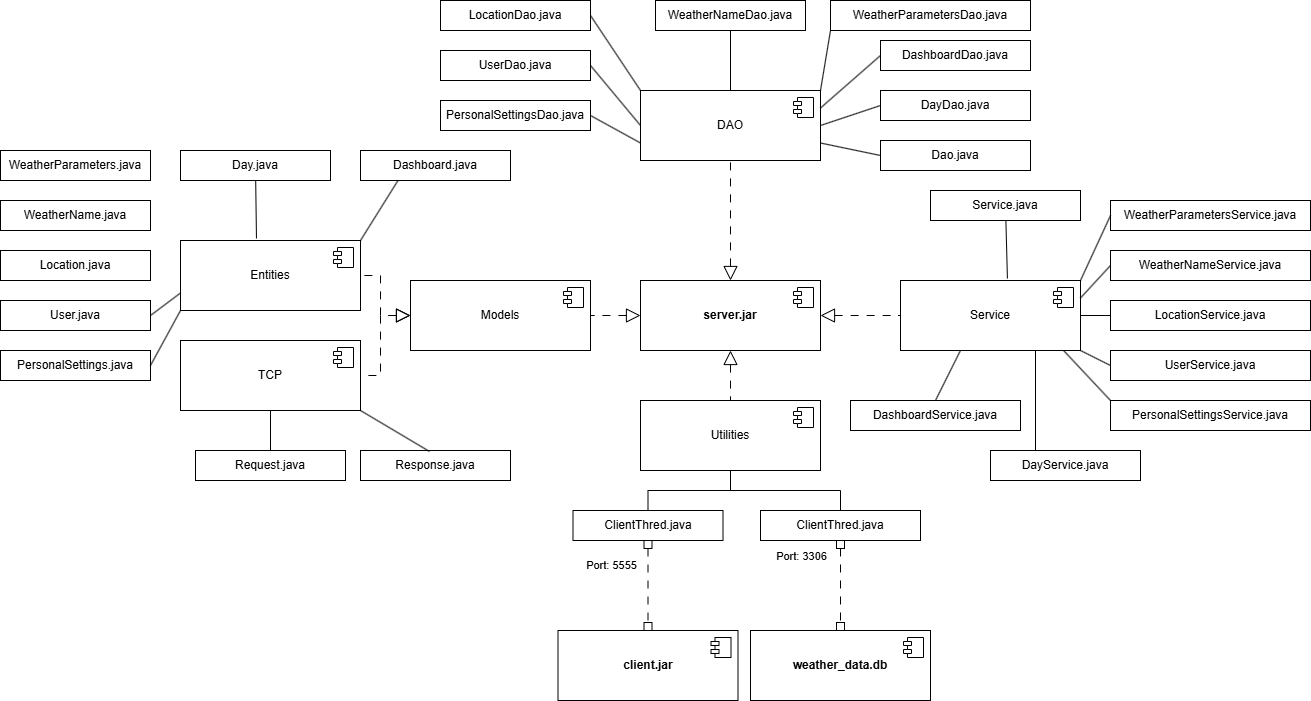


Рисунок 2.5 – Диаграмма компонентов сервера

Модуль клиента взаимодействует с сервером через открытый порт 5555 на ней. А для взаимодействия с базой данных используется локальный порт MySQL 3306. На рисунке 2.6 представлена диаграмма компонентов модуля клиента.

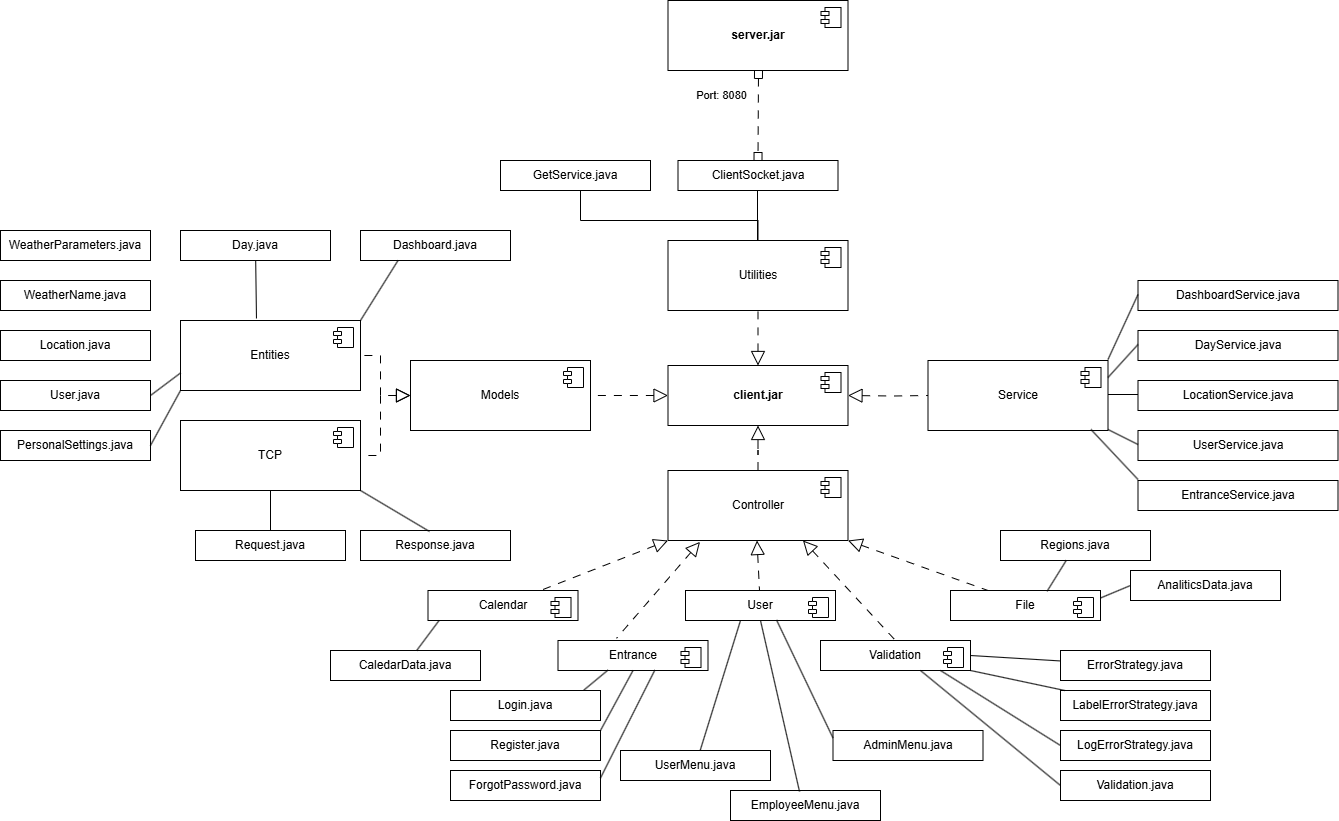


Рисунок 2.6 – Диаграмма компонентов клиента

Паттерн MVC (Model View Controller) – это способ организации кода, который предполагает выделение блоков, отвечающих за решение разных задач. Один блок отвечает за данные приложения, другой отвечает за внешний вид, а третий контролирует работу приложения [7].

Согласно паттерну MVC клиент реализуется на четырёх основных пакетах классов:

* Controller: необходим для обработки пользовательских запросов и запуска служб;
* Service: реализуется основная бизнес-логика приложения, отправка запросов на сервер;
* Model: содержит данные POJO-классов;
* View: пакет папки resources, который содержит все fxml файлы для визуального представления приложения пользователя.

Диаграмма развёртывания предназначена для описания аппаратной части системы. Она показывает на каком физическом оборудовании запускается та или иная составляющая программного обеспечения. На рисунке 2.7 представлена диаграмма развёртывания для разрабатываемого приложения.

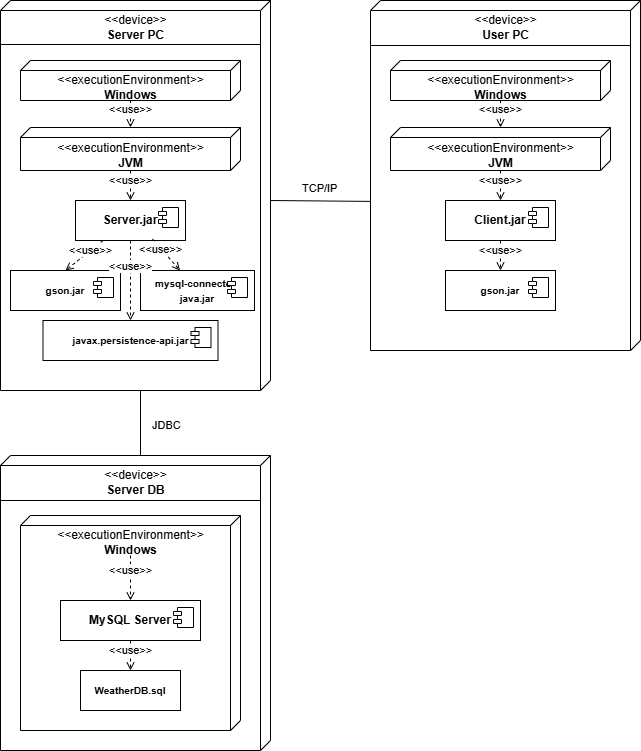


Рисунок 2.7 – Диаграмма развёртывания

Приложение реализуется на трех компонентах, один из которых – это компьютер пользователя «User PC». Узел «Server PC» осуществляет обработку данных поступающих с компьютеров пользователей. На нем установлена серверная часть разрабатываемой системы.

Компонент «Server DB» осуществляет хранение данных пользователей и предоставляет эти данные по запросу сервера приложения

Применение различных типов диаграмм, таких как диаграммы вариантов использования, состояний, последовательностей, классов, компонентов и развёртывания, позволяет всесторонне описать функциональность и архитектуру системы. Каждая диаграмма выполняет свою роль: от описания взаимодействий пользователей с системой до моделирования её структуры и логики работы.

## 2.3 Информационная модель системы и её описание

В качестве средства для хранения данных было выбрано использование базы данных. На рисунке 2.8 изображена схема базы данных weather\_data. Она состоит из 7 таблиц: day (данные о погоде на день), location (данные о местоположении), weather\_name (данные о прогнозе погоды), weather\_parameters (данные о параметрах погоды), user (данные пользователя для авторизации), personal\_settings (данные о пользователе), dashboard (данные о выбранном периоде просмотра погоды). Скрипт базы данных представлен в приложении Г.

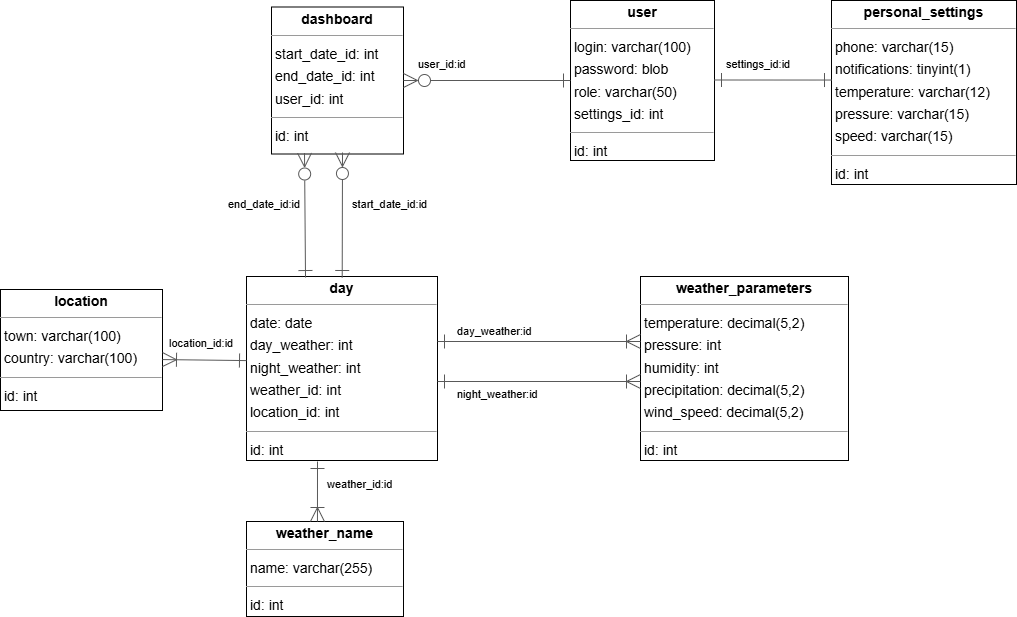


Рисунок 2.8 – Схема базы данных

Приведение БД к третьей нормальной форме. В первой нормальной форме все поля в базе данных должны быть простыми, все используемые множества допустимых значений должны содержать только скалярные значения. Не должно быть повторений строк в таблице. Во второй нормальной форме каждый неключевой атрибут неприводимо (функционально полно) должен зависеть от потенциального ключа. В третьей нормальной форме отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых. Необходимо выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы [8].

Структура базы данных для разрабатываемого приложения полностью соответствует третьей нормальной форме, что подтверждается выполнением следующих условий:

* таблицы находятся в первой нормальной форме: данные атомарны, отсутствуют повторяющиеся строки, используется единый формат данных;
* таблицы находятся во второй нормальной форме: каждый неключевой атрибут функционально полно зависит от первичного ключа;
* таблицы находятся в третьей нормальной форме: отсутствуют транзитивные зависимости между неключевыми атрибутами и ключом, а общие данные вынесены в отдельные таблицы.

Соответствие нормальным формам гарантирует целостность данных, отсутствие избыточности и оптимальную работу базы данных.

## 2.4 Описание алгоритмов, реализующих бизнес-логику серверной части проектируемой системы

Алгоритм в программировании – это набор последовательных инструкций, которые описывают порядок поведения программы для остижения нужной цели [9].

Эта функция представляет собой основную точку входа программы и служит для обработки входящих соединений от клиентов. Её алгоритм можно описать следующим образом. В самом начале создаётся серверный сокет (ServerSocket), который будет прослушивать подключения на указанном порту. Сервер работает в режиме постоянного ожидания. Главный цикл (while (true)) организует его непрерывную работу, обеспечивая возможность обработки как текущих подключений, так и новых.

В каждой итерации цикла сервер проходит по списку активных подключений (currentSockets). Если сокет закрыт, он удаляется из списка. Когда текущие подключения проверены, сервер переходит к ожиданию нового клиента с помощью метода accept(). Этот метод блокирует выполнение программы до тех пор, пока не произойдёт новое подключение. После успешного подключения в список активных подключений добавляется новый сокет. Для него создаётся объект ClientThread, который инкапсулирует всю логику работы с этим клиентом. Запускается новый поток, где выполняются операции с этим клиентом. После завершения обработки сервер возвращается к началу цикла и продолжает работать, ожидая новых клиентов. На рисунке 2.9 изображена блок-схема алгоритма функции main() на сервере. Код функции приведен в приложении Б.



Рисунок 2.9 – Схема алгоритма функции main() на сервере

Таким образом, функция отвечает за управление текущими соединениями и приём новых. Её логика организована так, чтобы сервер мог стабильно работать в условиях высокой нагрузки и большого числа подключений, эффективно управляя ресурсами.

## 2.5 Разработка схем алгоритмов программы

На рисунке 2.10 представлена схема алгоритма всей программы.

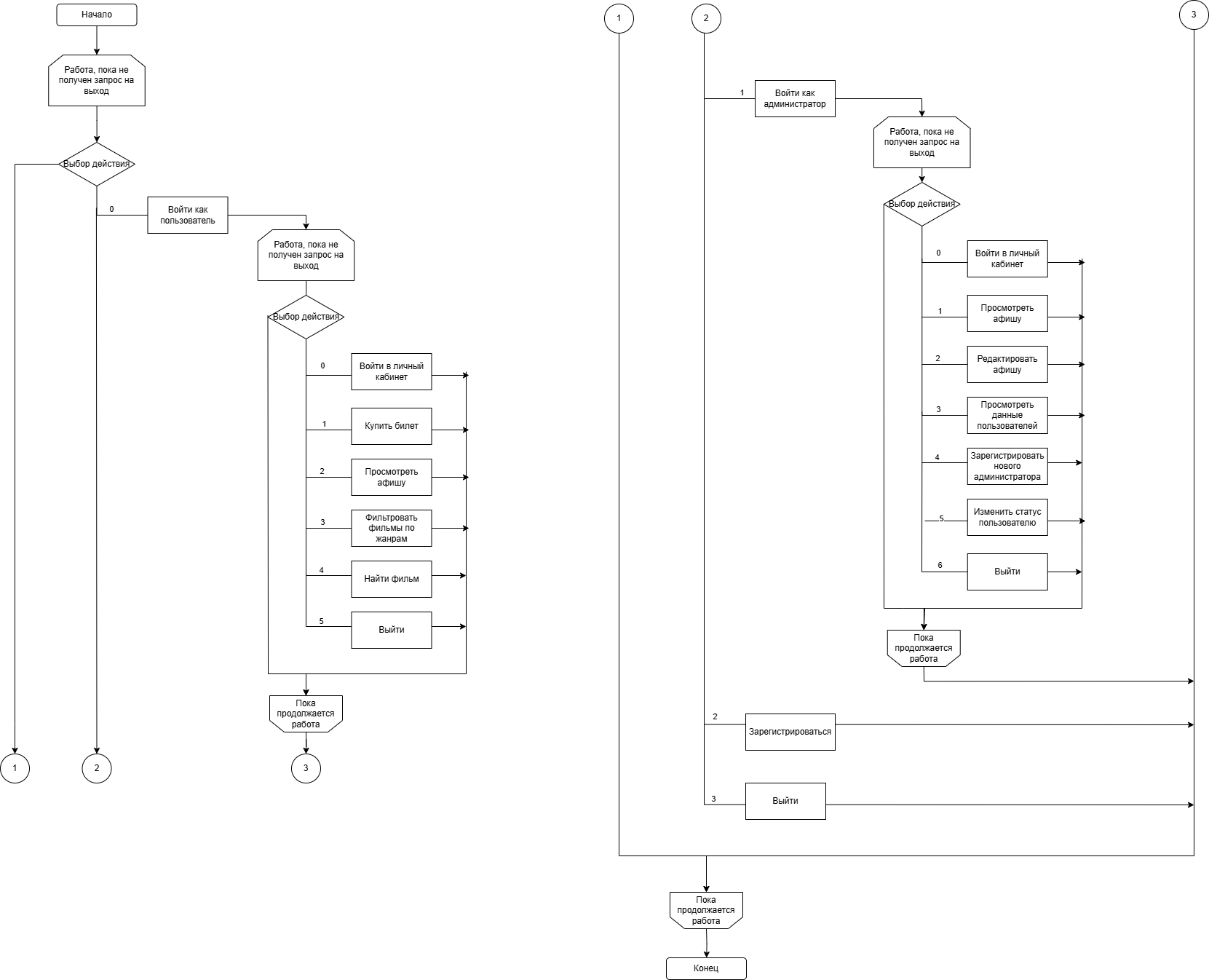


Рисунок 2.10 – Схема алгоритма модуля клиента

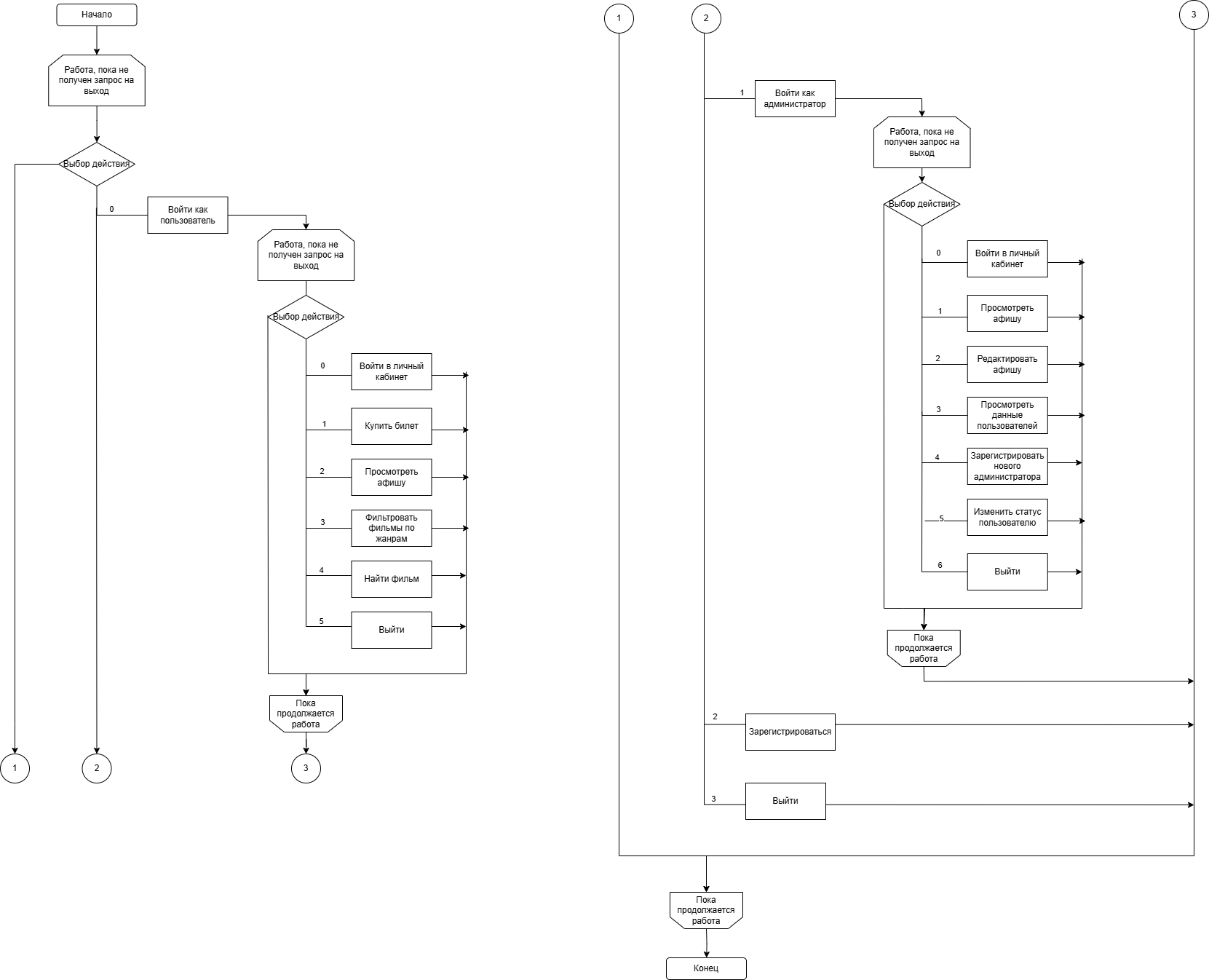


Рисунок 2.10 – Схема алгоритма модуля клиента (продолжение)

Эта блок-схема представляет собой алгоритм работы всего модуля клиента. После загрузки системы пользователь может войти в систему, зарегистрироваться либо восстановить пароль. После выбора входа существует разделение по правам доступа на обычного пользователя, сотрудника или администратора. В зависимости от выбранной роли пользователь имеет различный функционал.

Если человек входит как пользователь, ему доступны изменение личных данных и просмотр прогнозов погоды.

Для сотрудников функционал расширяется, кроме базовых функций, доступных пользователям, сотрудники могут редактировать базу данных, и анализировать графики за месяц.

Администратор, в свою очередь, работает с БД пользователей. Он может управлять правами доступа и просматривать статистику подключений к системе.

Независимо от роли, пользователи возвращаются к алгоритму программы до тех пор, пока не примут решение завершить свою работу. Код функции приведен в приложении Б.

На рисунке 2.11 представлена блок-схема алгоритма редактирования БД погоды.

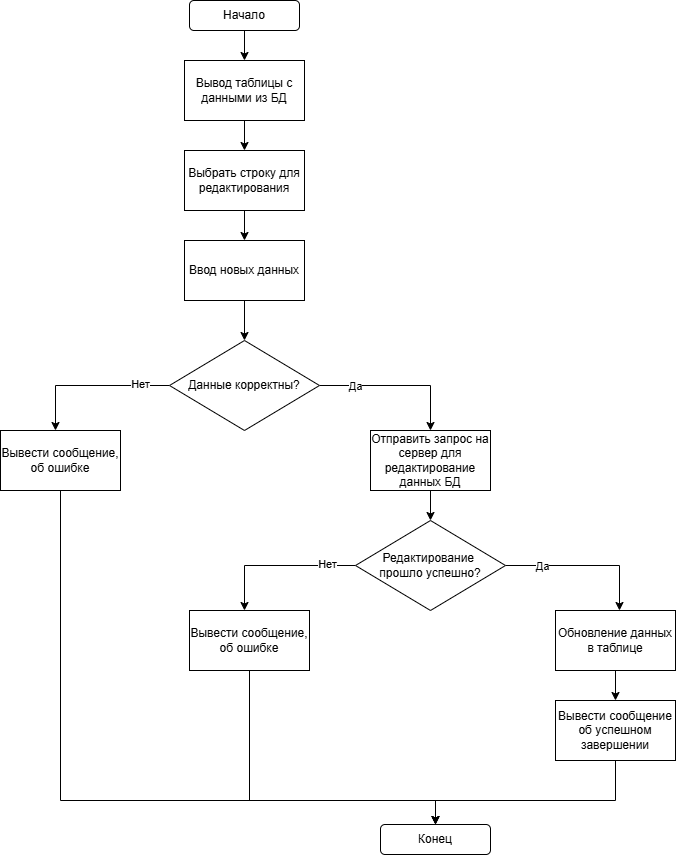


Рисунок 2.11 – Схема алгоритма редактирования

Этот алгоритм отвечает за обновление информации о дне, включая его погодные параметры, местоположение и описание прогноза погоды. Сначала пользователь должен выбрать строку редактируемого дня, затем ввести новые данные. Далее начинается валидация вводимых значений. Если какое-либо из полей содержит ошибку или выходит за допустимые пределы (например, температура ниже -40°C или выше 60°C), процесс немедленно прекращается, а пользователь получает сообщение о некорректных данных.

После ввода корректных данных происходит обновление дня через сервис dayService. Если обновление прошло успешно, таблица, содержащая список дней, синхронизируется: обновлённый объект заменяет старую версию. Если по какой-то причине нужный элемент не найден в таблице, пользователю сообщается об ошибке.

Таким образом, блок-схемы позволяют наглядно представить процесс выполнения программы, выделяя ключевые этапы, точки принятия решений и взаимосвязь между элементами алгоритма. В процессе их анализа можно выявлять возможные ошибки или неоптимальные участки на стадии проектирования, что значительно повышает эффективность разработки и последующего тестирования программного обеспечения.

# 3 Тестирование и проверка работоспособности программного средства

Тестирование является ключевым этапом разработки программного обеспечения, позволяющим убедиться в его корректной работе, стабильности и соответствии заданным требованиям. Этот процесс включает в себя проверку функциональности всех компонентов программы, выявление возможных ошибок и несоответствий, а также оценку её производительности в различных условиях эксплуатации [10].

Были написаны тесты для всех классов DAO слоя, для проверки функций добавления, удаления, редактирования, поиска и получения данных из базы данных. Результаты тестов представлены на рисунке 3.1. Полученные результаты подтвердили корректность работы программы в различных сценариях использования.

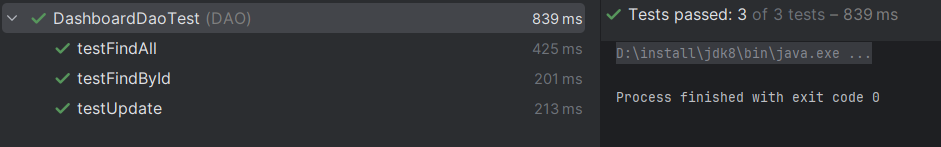
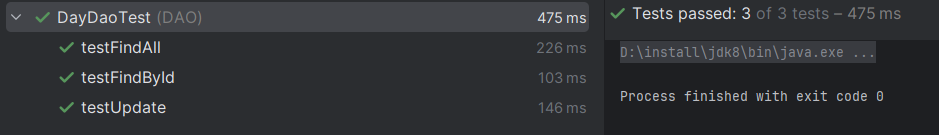
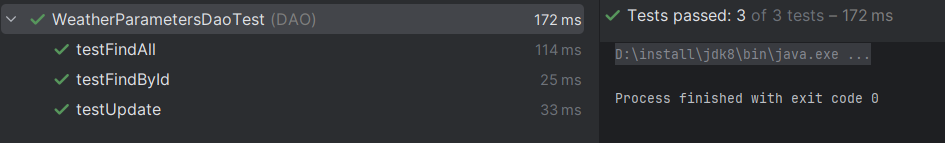
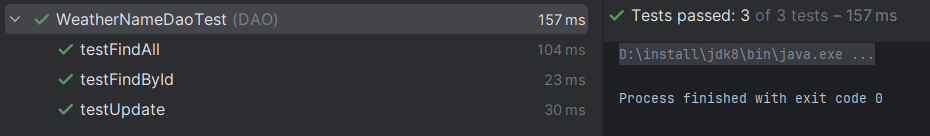
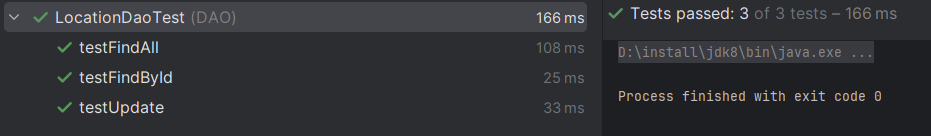
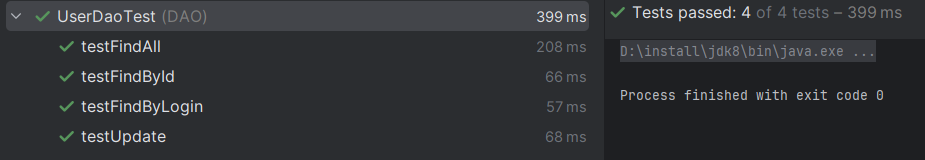
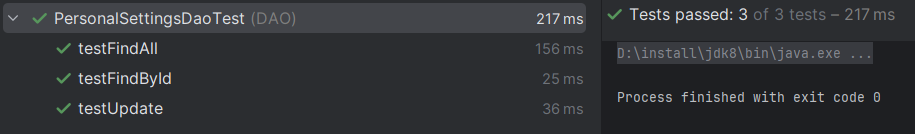


Рисунок 3.1 – Результаты тестирования

При написании кода программы необходимо предусмотреть обработку исключительных ситуаций, таких как:

* введенные пользователем данные не соответствуют формату поля (например, символы в числовом поле);
* введенные пользователем данные нелогичны (например, температура больше 50°C);
* проверки на уникальность введенных данных (если необходимо).

Во время регистрации очень важно вводить корректные данные, поэтому существует проверка на валидность логина (см. рисунок 3.2), он должен содержать только цифры и латинские буквы и быть длиной от трёх до 25 символов.

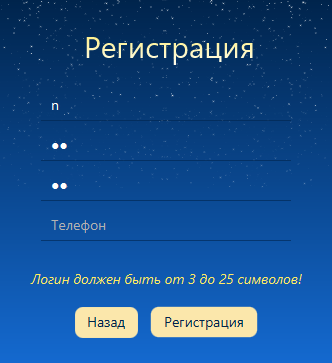


Рисунок 3.2 – Проверка логина

Во время регистрации также необходимо подтвердить введённый пароль (см. рисунок 3.3), при получении разных данных выводится ошибка.

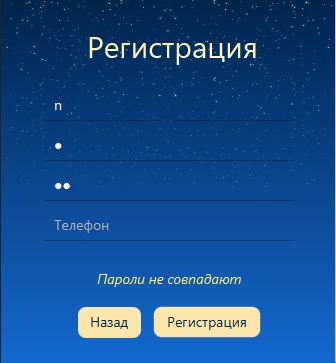


Рисунок 3.3 – Проверка соответствия паролей

Далее происходит проверка на уникальность введённого логина. Если логин уже занят, то выводится ошибка (см. рисунок 3.4).

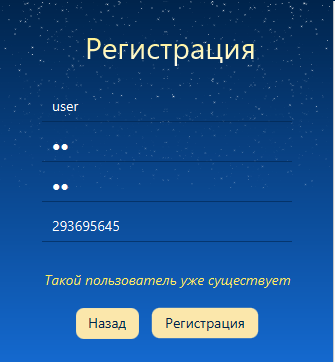


Рисунок 3.4 – Проверка уникальности логина

Номер телефона пользователя должен содержать девять цифр (с кодом), при вводе некорректного номера выводится сообщение об ошибке (см. рисунок 3.5).

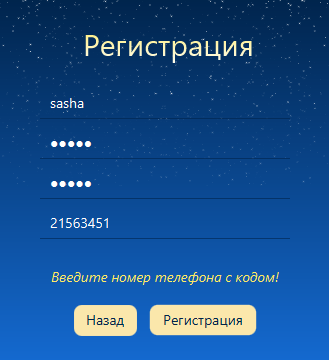


Рисунок 3.5 – Проверка валидности номера телефона

В программе существует большое количество проверок на ввод данных. Например, при изменении данных о погоде и ввода температуры, равной –100°С, вводится сообщение об ошибке (см. рисунок 3.6).

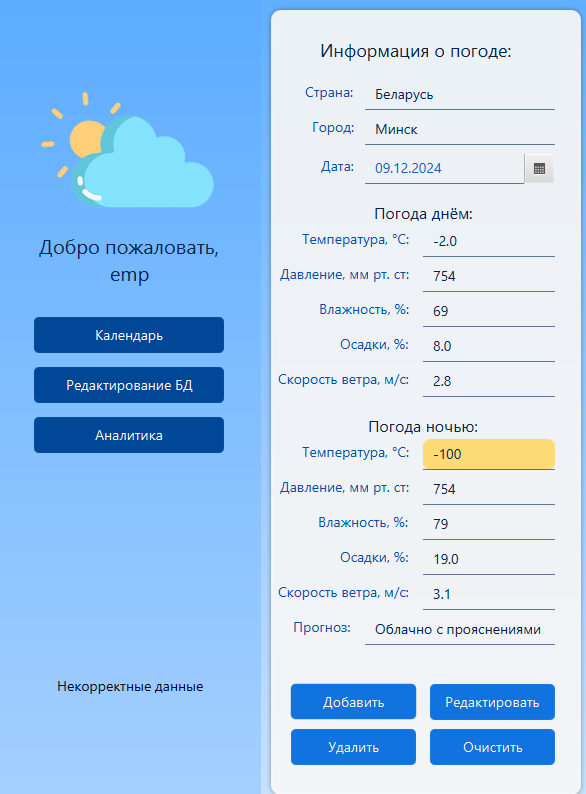


Рисунок 3.6 – Проверка корректности введённых данных

Таким образом, реализованные тесты и механизмы обработки исключений создают прочную основу для успешного функционирования программы, что является важным итогом тестового этапа разработки.

Особое внимание было уделено обработке исключительных ситуаций, что позволило сделать систему более устойчивой к ошибкам, связанным с некорректным вводом данных, нарушением логики или требованиями уникальности. Это не только увеличивает надёжность приложения, но и повышает его удобство для конечного пользователя.

# 4 Инструкция по развертыванию приложения

Этот раздел представляет собой руководство по эксплуатации разработанного приложения, его интерфейс для пользователей с различными правами доступа.

## 4.1 Авторизация

После загрузки программы появляется меню, с помощью которого производится вход в систему. Пользователь может ввести свой логин и пароль для входа, пройти регистрацию или выбрать восстановление пароля (см. рисунок 4.1).

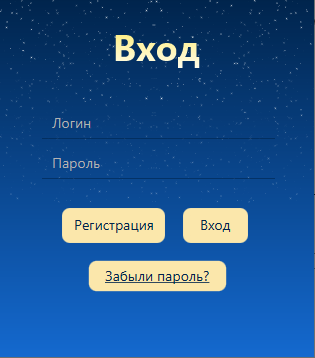


Рисунок 4.1 – Вход в систему

Если пользователь выбрал регистрацию, то появляется окно, представленное на рисунке 4.2. Здесь нужно ввести логин, пароль и телефон, также можно изменить настройки единиц измерения или оставить их по умолчанию.

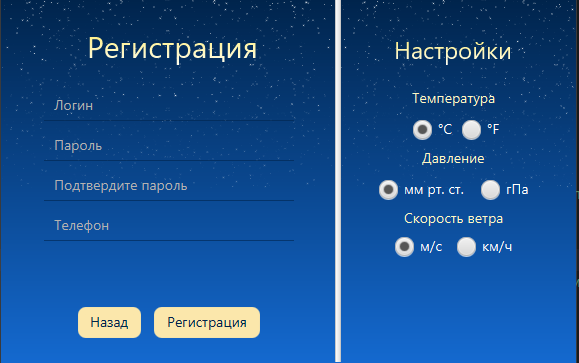


Рисунок 4.2 – Регистрация

Если пользователь выбрал кнопку «Забыли пароль?», то появляется окно восстановления пароля, представленное на рисунке 4.3. Здесь нужно ввести логин и телефон, после чего, если будет найден такой пользователь, можно будет ввести новый пароль.

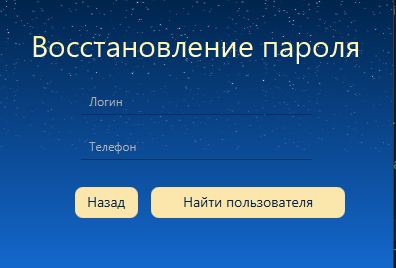


Рисунок 4.3 – Восстановление пароля

## 4.2 Модуль пользователя

При входе в систему с правами доступа обычного пользователя, появляется главное меню пользователя (см. рисунок 4.4), которое включает просмотр погоды на текущую дату, возможность открыть личный кабинет для изменения настроек и возможность открыть календарь, нажав на соответствующую кнопку.

При выборе кнопки «Выйти» пользователь отключается и возвращается на окно входа в систему.

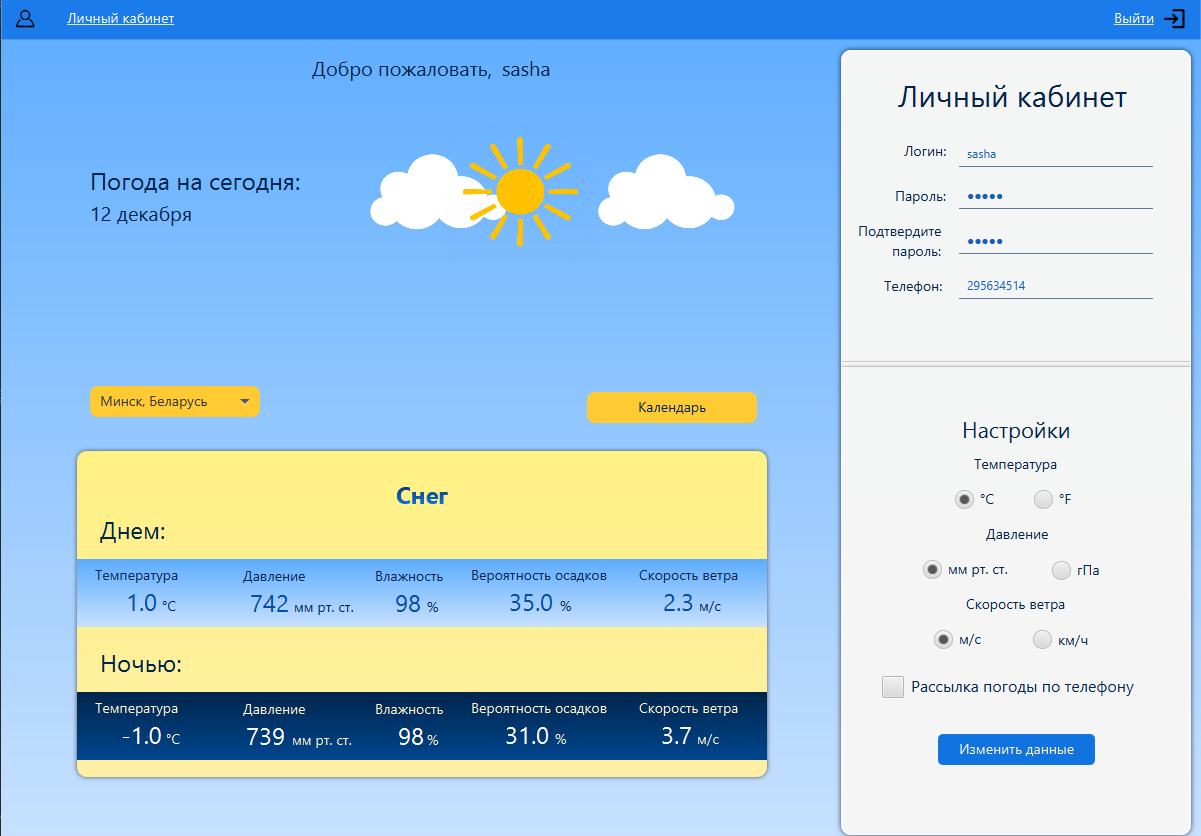


Рисунок 4.4 – Главное меню пользователя

При выборе кнопки «Календарь» пользователь переходит на окно просмотра погоды. Здесь отображен календарь с погодой за выбранный месяц и год в выбранном регионе. При нажатии на день в календаре отображается подробный прогноз погоды на выбранную дату. Также можно выбрать период изменения температуры для создания соответствующего графика. После чего сохранить свои данные, чтобы они автоматически отображались при следующем входе в систему (см. рисунок 4.5).

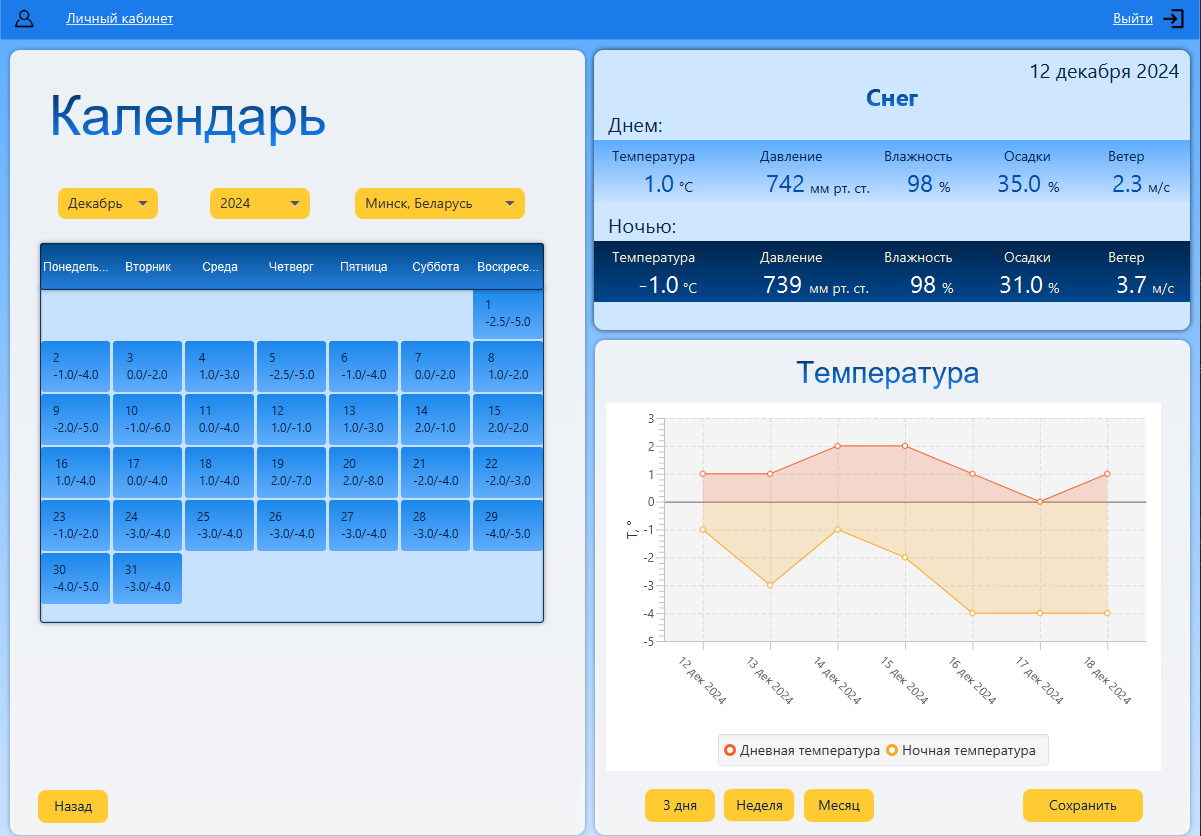


Рисунок 4.5 – Меню «Календарь»

## 4.3 Модуль сотрудника

При входе в систему с правами доступа сотрудника, появляется главное меню сотрудника (см. рисунок 4.6), которое включает просмотр погоды на текущую дату, возможность открыть личный кабинет для изменения настроек и набор функций для просмотра календаря, редактирования БД и просмотра графиков.

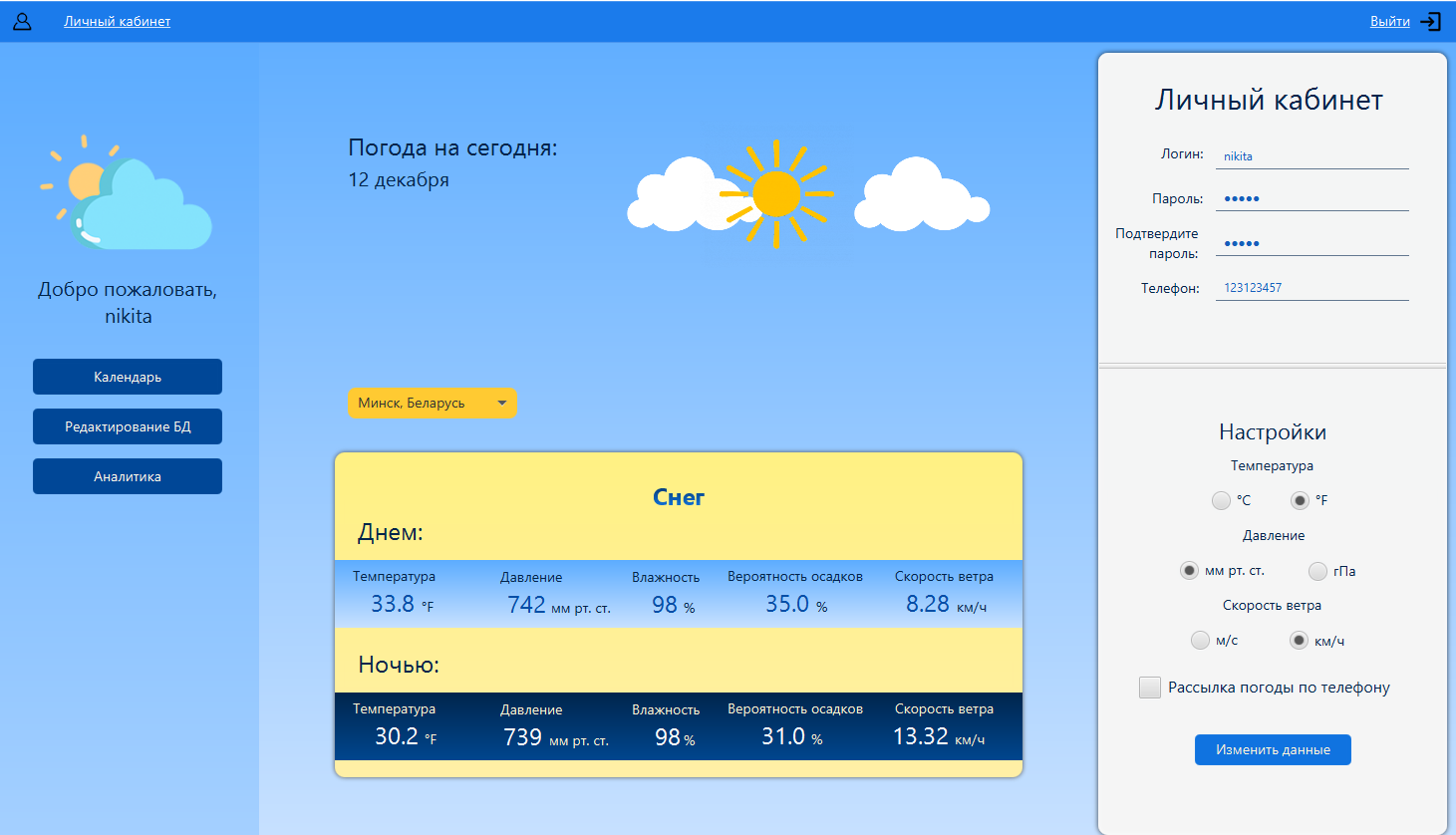


Рисунок 4.6 – Главное меню сотрудника

При выборе кнопки «Календарь» сотрудник получает доступ к функции просмотра погоды на месяц (см. рисунок 4.7). Функционал этого меню аналогичен функционалу меню «Календарь» обычного пользователя.

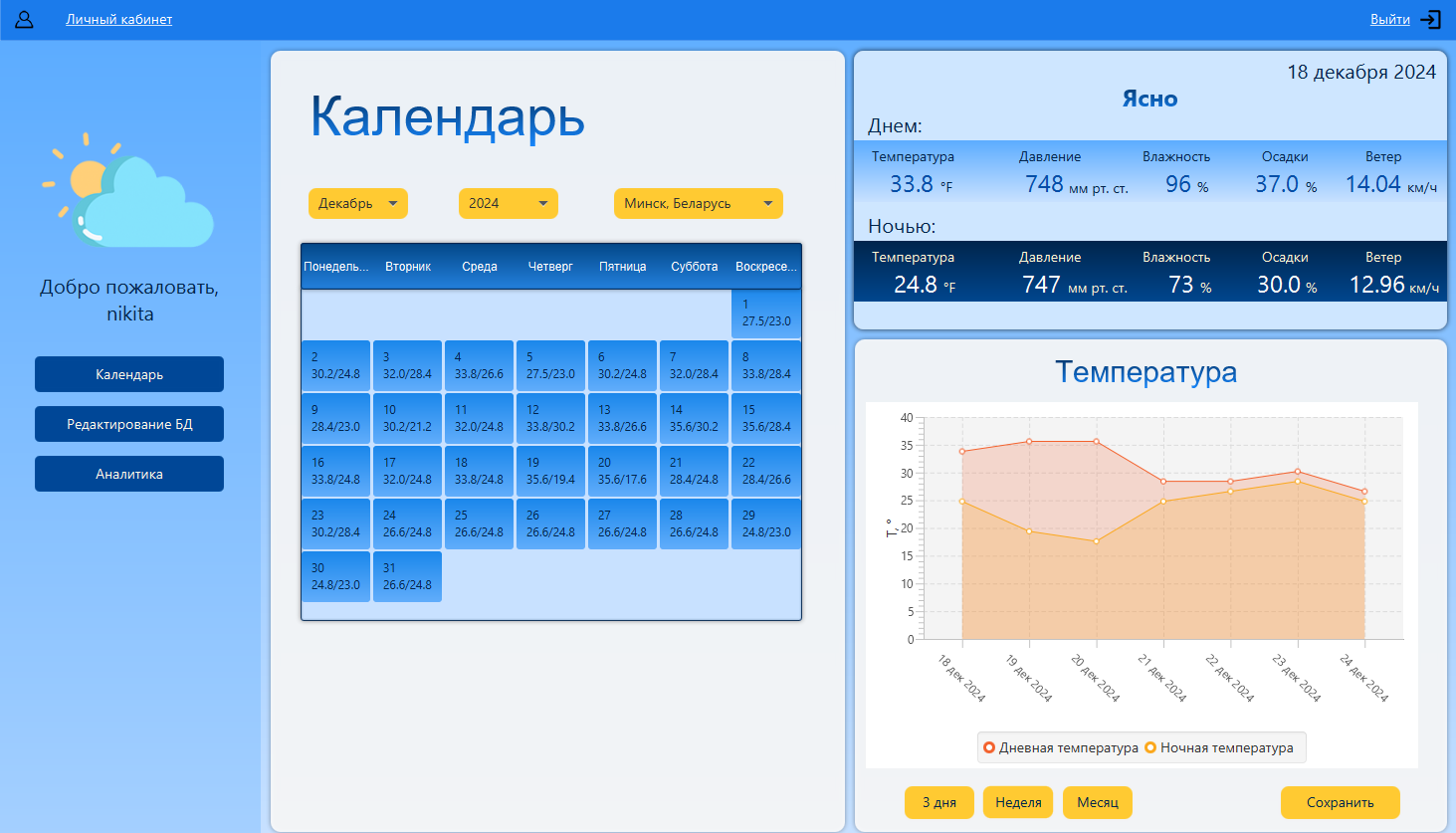


Рисунок 4.7 – Меню «Календарь»

При выборе кнопки «Редактирование БД» сотрудник получает доступ к функциям работы с БД погоды, таким как добавление, редактирование и удаление. Чтобы удалить или отредактировать какие-либо данные, необходимо сначала выбрать нужную строку в таблице просмотра погоды (см. рисунок 4.8). Кнопка «Очистить» позволяет очистить данные из всех полей ввода.

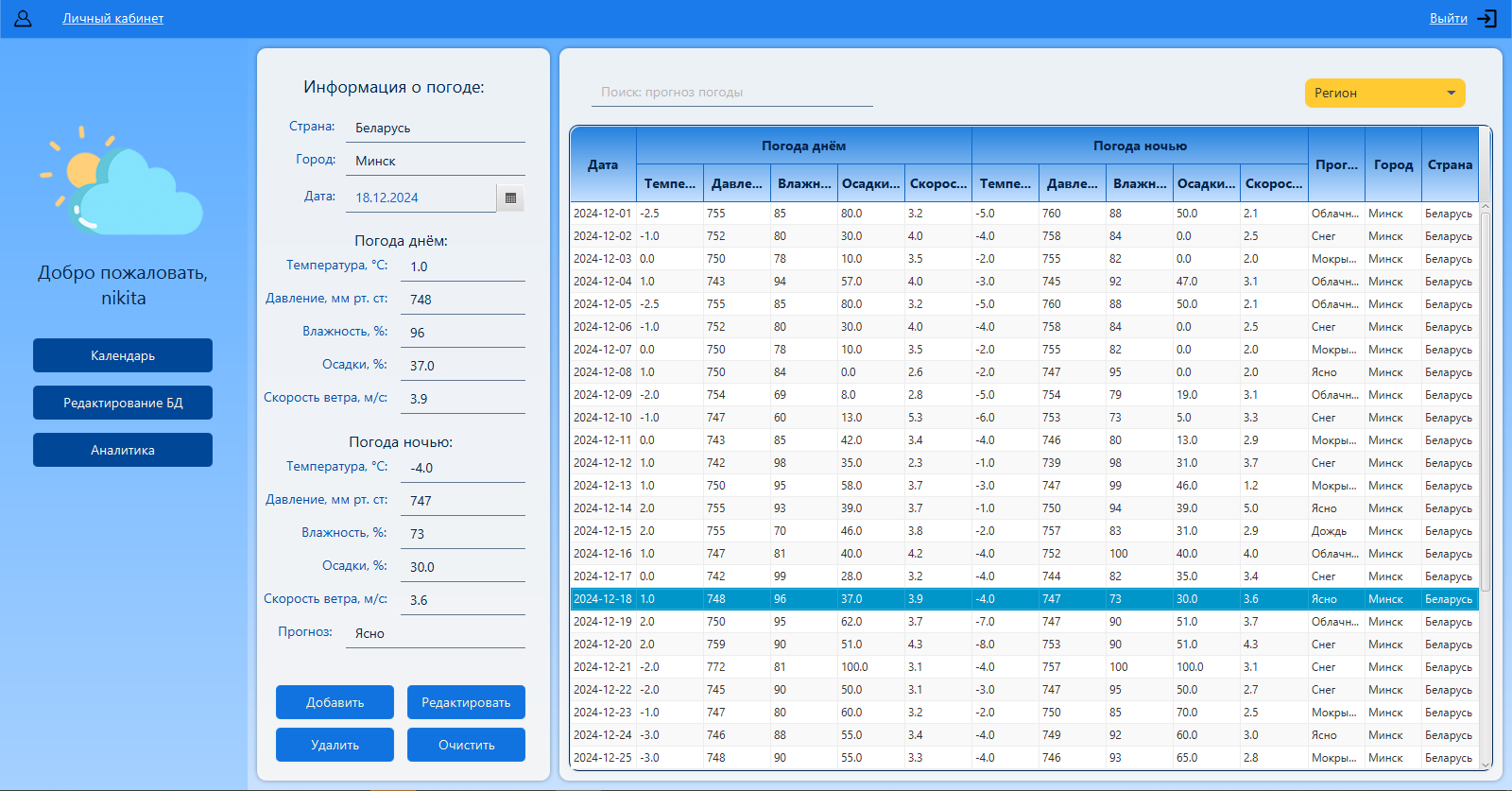


Рисунок 4.8 – Меню «Редактирование БД»

Для удобного просмотра данных можно отфильтровать данные по региону или ввести данные для поиска. Будет сделан поиск по столбцам «Дата», «Прогноз», «Город» и «Страна» (см. рисунок 4.9).

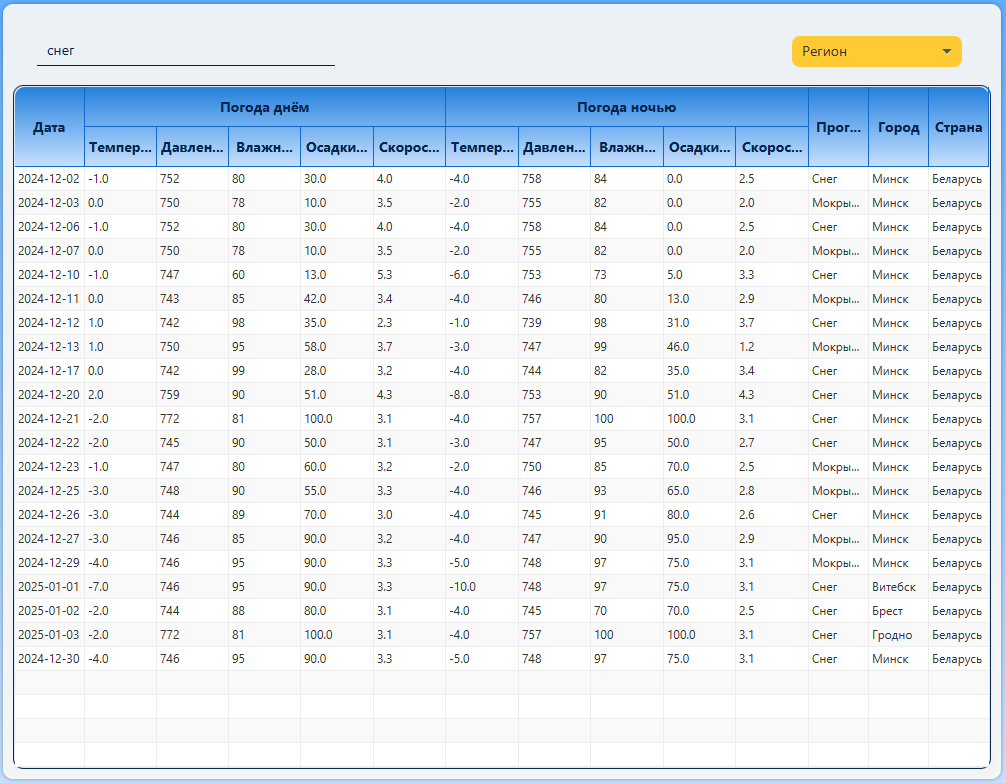


Рисунок 4.9 – Фильтрация погоды по прогнозу

При выборе кнопки «Аналитика» сотрудник получает доступ к просмотру графиков изменения температуры, давления и скорости ветра за выбранный месяц и диаграммы, отображающей разные прогнозы и количество дней с такими прогнозами. Под диаграммой отображаются минимальные, максимальные и средние значения параметров температуры, давления и скорости ветра, их можно сохранить в файл, если нажать на кнопку «Запись в файл» (см. рисунок 4.10).

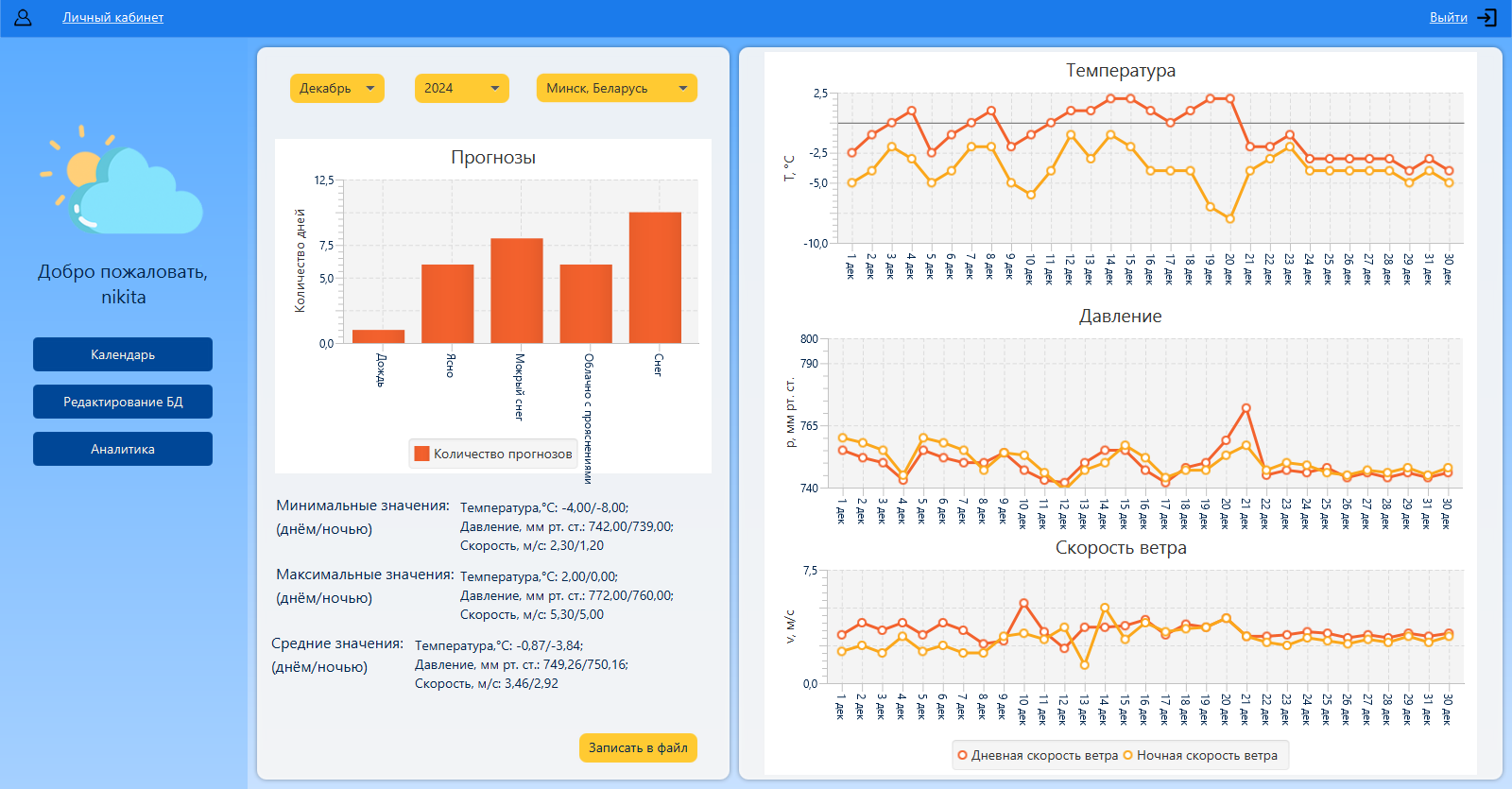


Рисунок 4.10 – Меню «Аналитика»

Чтобы вернуться в главное меню, нужно нажать на прозрачную кнопку, расположенную под надписью «Добро пожаловать, employee».

## 4.4 Модуль администратора

При входе в систему с правами доступа администратор, появляется главное меню администратора (см. рисунок 4.11), которое включает просмотр статистики подключений, возможность открыть личный кабинет для изменения настроек и функцию для управления правами доступа.

Статистика подключений отображает количество текущих пользователей, которые вошли в систему, и таблицу с их логинами и ролями.

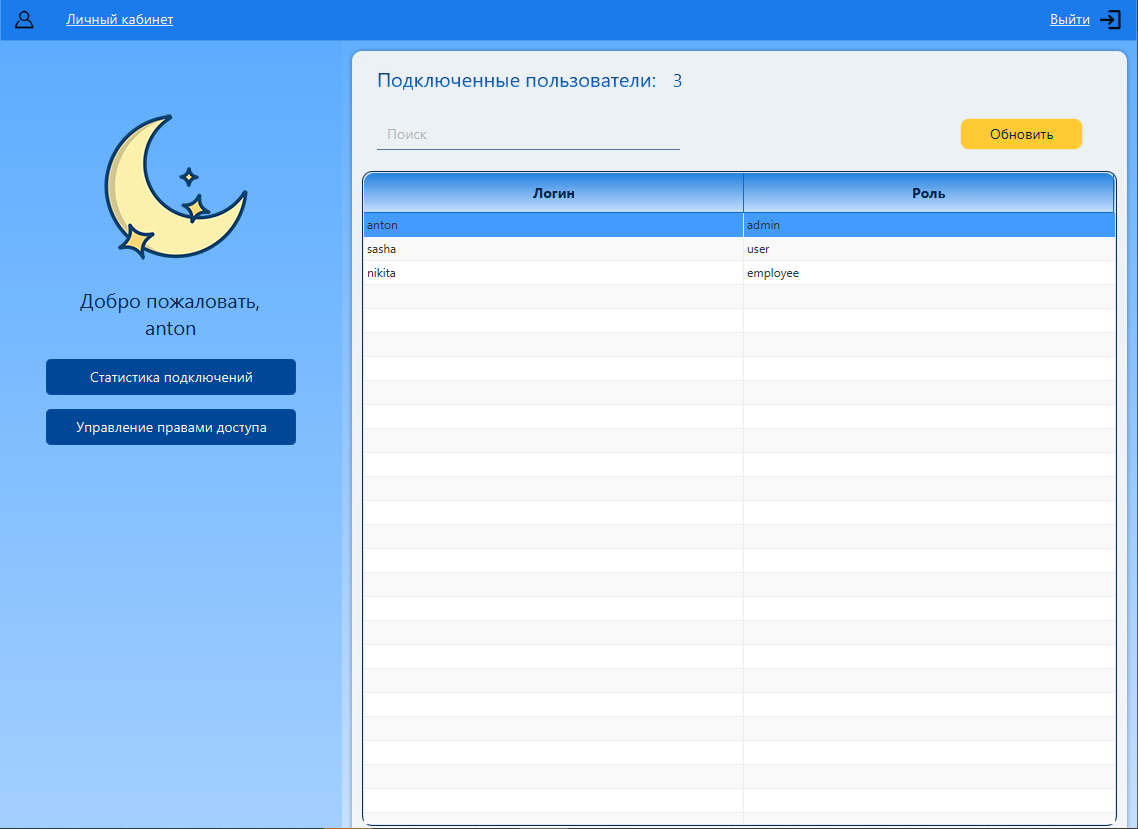


Рисунок 4.11 – Главное меню администратора

При выборе кнопки «Личный кабинет» администратору открывается меню настроек для изменения его логина, пароля и телефона (см. рисунок 4.12).

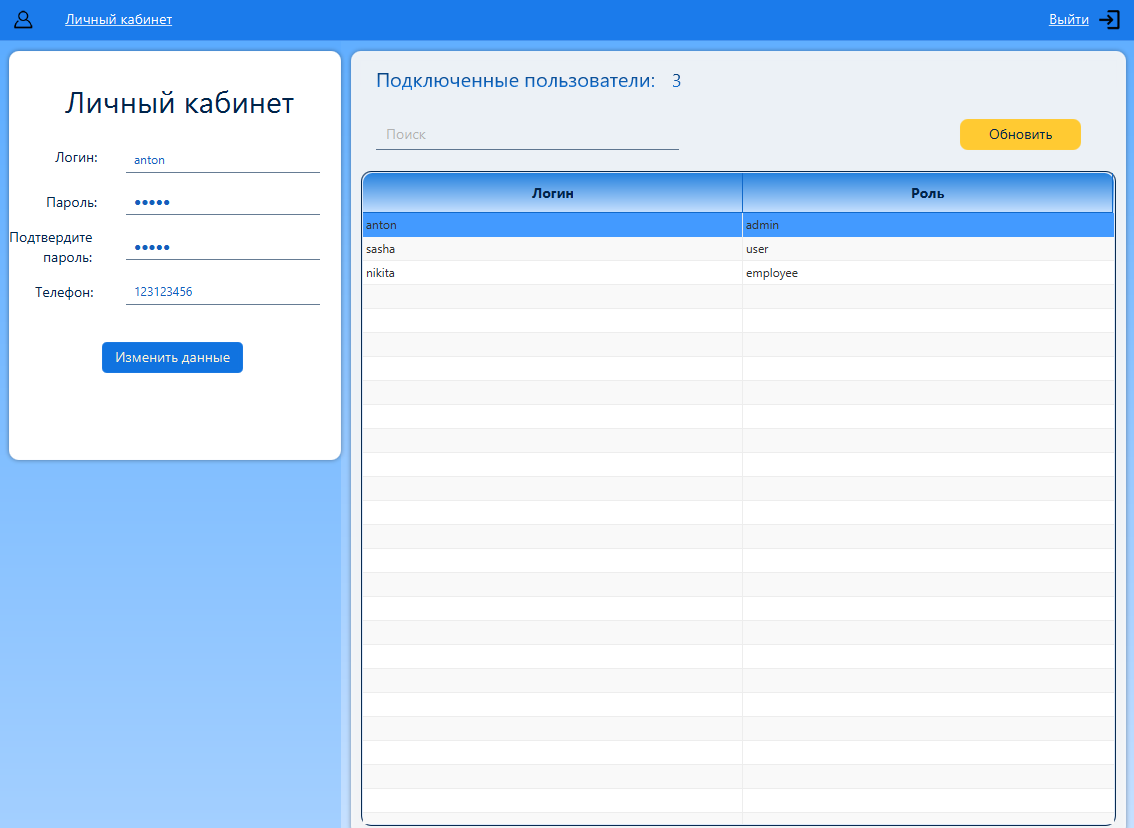


Рисунок 4.12 – Личный кабинет администратора

При выборе кнопки «Управление правами доступа» администратор получает доступ к просмотру таблицы, в которой отображены все данные пользователей кроме их паролей. Выбрав поле в столбце «Роль», администратор может изменить права доступа выбранного пользователя (см. рисунок 4.13).

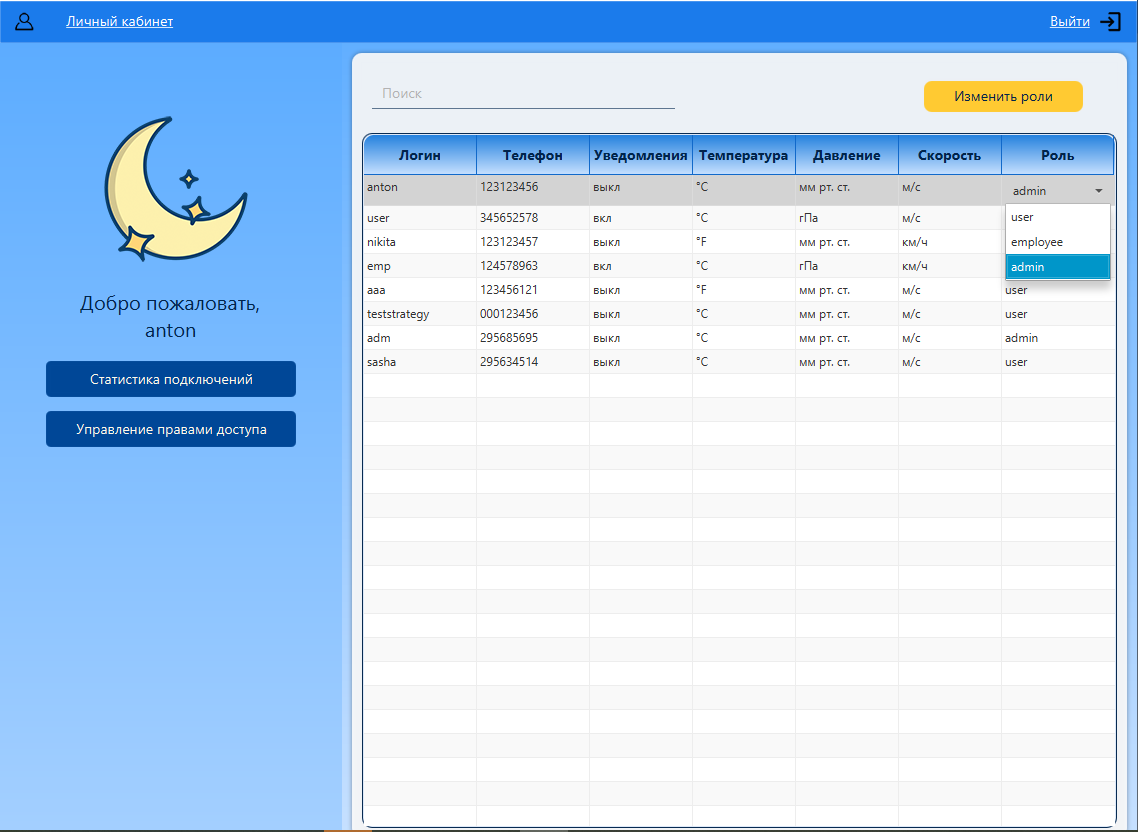


Рисунок 4.13 – Меню «Управление правами доступа»

Таким образом, разработанная программа удовлетворяет потребностям всех категорий пользователей. Они получают доступ к интуитивно понятному интерфейсу, который упрощает выполнение основных операций и обеспечивает комфортное взаимодействие с приложением.

# Заключение

При выполнении курсовой работы был изучен процесс автоматизации прогнозирования погоды. Использование современных статистических методов и автоматизированных систем обработки данных позволяет существенно снизить трудоемкость и повысить точность прогнозов, минимизируя вероятность ошибок, связанных с ручной обработкой данных.

В ходе работы над курсовым проектом был изучен процесс прогнозирования, который представляет собой сложный процесс, объединяющий современные технологии, математическое моделирование и анализ больших объемов данных. Применение функционального моделирования стало важным этапом проектирования информационной системы. Эта методология упростила процесс понимания логики процесса составления прогноза погоды.

Использование UML-диаграмм, позволяет описать основные сценарии использования, структуру и взаимодействие компонентов системы. Такой подход обеспечивает глубокое представление как о взаимодействии пользователей с программой, так и о её внутренней логике и структуре. Разработка блок-схем стала ключевым элементом проектирования, предоставляя наглядное описание алгоритмов работы программы. Благодаря этому удалось выделить ключевые этапы, точки принятия решений и взаимосвязи элементов. Этот шаг способствовал обнаружению возможных ошибок и оптимизации системы ещё на стадии проектирования.

Проведено тестирование программного обеспечения с использованием реальных метеорологических данных. Особое внимание было уделено обработке исключительных ситуаций, что сделало систему устойчивой к ошибкам. Этот аспект значительно повысил удобство использования приложения и обеспечил его стабильную работу.

В итоге разработанная программа полностью удовлетворяет потребностям пользователей с разными уровнями доступа. Благодаря интуитивно понятному интерфейсу пользователи могут легко выполнять основные операции, что делает взаимодействие с системой комфортным и продуктивным.

Цель курсовой работы – сокращение времени обработки метеорологических данных за счет разработки автоматизированной системы – была успешно достигнута. Итогом работы является приложение, которое позволяет собирать, анализировать и предоставлять метеорологические данные для различных категорий пользователей. Разработанное программное средство полностью удовлетворяет функциональным требованиям, необходимым для автоматизации процесса построения прогнозов погоды.

# Список использованных источников

1. Статья «Что такое аналоговое прогнозирование?». [Электронный ресурс]. – Режим доступа:

https://www.worldclimateservice.com/2021/09/02/what-is-analog-forecasting/.

1. Статья «Улучшение аналоговых прогнозов скорости ветра для редких событий». [Электронный ресурс]. – Режим доступа: https://journals.ametsoc.org/view/journals/mwre/147/7/mwr-d-19-0006.1.xml.
2. Статья «Применение регрессионного анализа для прогнозирования погоды». [Электронный ресурс]. – Режим доступа: https://www.researchgate.net/publication/379675299\_Primenenie\_regressionnogo\_analiza\_dla\_prognozirovania\_pogody.
3. Статья «Вероятностное прогнозирование». [Электронный ресурс]. – Режим доступа: <https://fnow.ru/articles/veroyatnostnoe_prognozirovanie>
4. Статья «Паттерны проектирования». [Электронный ресурс]. – Режим доступа: https://refactoring.guru/ru/design-patterns/java.
5. Методическое пособие для выполнения лабораторной роботы «Язык Uml. Диаграммы UML» [Электронный ресурс]. – Режим доступа: \трпо\ТРПО– Лабораторные работы №7-8.docx.
6. Статья «Что такое MVC: рассказываем простыми словами». [Электронный ресурс]. – Режим доступа: <https://ru.hexlet.io/blog/posts/chto-takoe-mvc-rasskazyvaem-prostymi-slovami>.
7. Статья «Нормализация отношений. Шесть нормальных форм». [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/articles/254773/>.
8. Статья «Блок-схема алгоритма: разбираемся в особенностях». [Электронный ресурс]. – Режим доступа: https://gb.ru/blog/blok-shema-algoritma/.
9. Статья «Тестирование программного обеспечения: разбираемся в деталях». [Электронный ресурс]. – Режим доступа: https://gb.ru/blog/testirovanie-programmnogo-obespecheniya/.

# Приложение А

**(обязательное)**

**Отчет о проверке на заимствования в системе «Антиплагиат»**

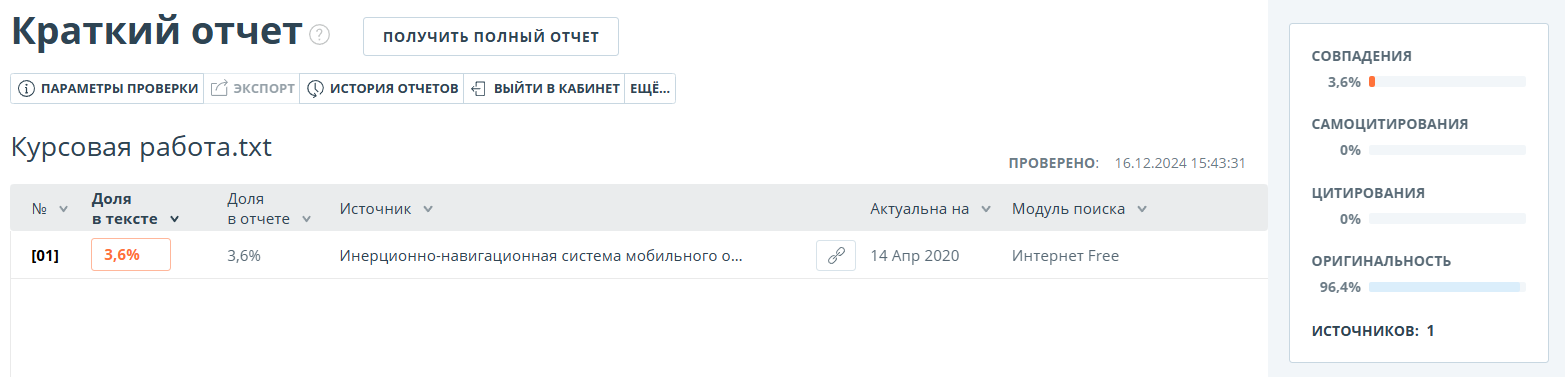


Рисунок А.1 – Отчет о проверке на заимствования в системе «Антиплагиат»

# Приложение Б

**(обязательное)**

**Листинг кода алгоритмов, реализующих основную бизнес-логику приложения**

//Сервер

public class Server {

private static final int PORT\_NUMBER = 5555;

private static ServerSocket serverSocket;

private static ClientThread clientHandler;

private static Thread thread;

private static List<Socket> currentSockets = new ArrayList<>();

public static void main(String[] args) throws IOException, SQLException {

serverSocket = new ServerSocket(PORT\_NUMBER);

while (true) {

for (int i = 0; i < currentSockets.size(); i++) {

Socket socket = currentSockets.get(i);

if (socket.isClosed()) {

// Если сокет закрыт, удаляем его из списка

currentSockets.remove(i);

i--;

continue;

}

}

// Ожидаем нового клиента

Socket socket = serverSocket.accept();

currentSockets.add(socket);

clientHandler = new ClientThread(socket);

thread = new Thread(clientHandler);

thread.start(); // Запускаем новый поток для клиента

System.out.flush();

}

}

}

}

//класс ClientThred

public ClientThread(Socket clientSocket) throws IOException, SQLException {

this.clientSocket = clientSocket;

in = new BufferedReader(new InputStreamReader (clientSocket.getInputStream()));

out = new PrintWriter(clientSocket.getOutputStream(), true);

}

@Override

public void run() {

try {

System.out.println("Клиент " + clientSocket.getInetAddress() + ":" + clientSocket.getPort() + " подключен.");

while (clientSocket.isConnected()) {

String message = in.readLine();

if (message == null) { // Клиент разорвал соединение

break;

}

Request request = gson.fromJson(message, Request.class);//

Response response = handleRequest(message);

out.println(gson.toJson(response)); // Отправка ответа клиенту

if (request != null && request.getRequestType() == RequestType.LOGOUT) {

closeResources();

break;

}

}

Продолжение приложения Б

} catch (IOException e) {

System.err.println("Ошибка связи с клиентом: " + e.getMessage());

} finally {

if (currentId != 0) {

synchronized (loginUsersId) {

loginUsersId.removeIf(id -> id == currentId);

}

}

if (clientSocket != null && !clientSocket.isClosed())

closeResources();

}

}

private Response handleRequest(String message) {

try {

Request request = gson.fromJson(message, Request.class);

if (request == null || request.getRequestType() == null) {

return new Response(ResponseStatus.ERROR, "Некорректный запрос", "");

}

switch (request.getRequestType()) {

case REGISTER:

return handleRegister(request);

case LOGIN:

return handleLogin(request);

case FORGOT\_PASSWORD:

return handleForgotPassword(request);

case UPDATE\_USER:

return handleUpdateUser(request);

case GET\_ALL\_USERS:

return handleGetAllUsers();

case UPDATE\_ALL\_USERS:

return handleUpdateAllUsers(request);

case GET\_CONNECTED\_USERS:

return handleGetConnectedUsers();

case GET\_REGIONS:

return handleGetRegions();

case GET\_ALL\_DAYS:

return handleGetAllDays();

case LOGOUT:

return handleLogout(request);

case TODAY\_WEATHER:

return handleGetTodayWeather(request);

case ADD\_DAY:

return handleAddDay(request);

case UPDATE\_DAY:

return handleUpdateDay(request);

case DELETE\_DAY:

return handleDeleteDay(request);

case GET\_DASHBOARD:

return handleGetDashboard(request);

case ADD\_USER\_DASHBOARD:

return handleAddUserDashboard(request);

default:

return new Response(ResponseStatus.ERROR, "Неизвестный тип запроса", "");

}

} catch (Exception e) {

System.err.println("Ошибка обработки запроса: " + e.getMessage());

return new Response(ResponseStatus.ERROR, "Ошибка обработки запроса", "");

} }

Продолжение приложения Б

//Клиент

//Реализация паттерна Singletone

public class ClientSocket {

private static ClientSocket SINGLE\_INSTANCE;

private User user;

private Socket socket;

private BufferedReader in;

private PrintWriter out;

private ClientSocket() { // Конструктор для установления соединения с сервером

connectToServer();}

// Метод для создания нового соединения

private void connectToServer() {

try {

// Подключаемся к серверу по указанному адресу и порту

socket = new Socket("localhost", 5555);

// Создаём потоки для ввода и вывода данных

in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

out = new PrintWriter(socket.getOutputStream(), true); // Автоматически вызывает flush()

} catch (IOException e) {

System.err.println("Ошибка при подключении к серверу: " + e.getMessage());

e.printStackTrace();

}

}

// Получение экземпляра синглтона

public static ClientSocket getInstance() {

if (SINGLE\_INSTANCE == null) {

SINGLE\_INSTANCE = new ClientSocket();

}

return SINGLE\_INSTANCE;

}

// Метод для закрытия всех ресурсов

public void close() {

try {

if (in != null) {

in.close();

}

if (out != null) {

out.close();

}

if (socket != null && !socket.isClosed()) {

socket.close();

}

} catch (IOException e) {

System.err.println("Ошибка при закрытии ресурсов: " + e.getMessage());

}

}

// Метод для перезапуска соединения

public static void resetConnection() {

if (SINGLE\_INSTANCE != null) {

SINGLE\_INSTANCE.close(); // Закрываем старое соединение

}

SINGLE\_INSTANCE = new ClientSocket(); // Создаем новый экземпляр с новым соединением

}

}

# Приложение В

**(обязательное)**

**Листинг основных элементов программы**

//Реализация паттерна Strategy

interface ErrorStrategy {

void handleError(String message);

}

public class LabelErrorStrategy implements ErrorStrategy {

private final Label labelError;

public LabelErrorStrategy(Label labelError) {

labelError.setVisible(false);

this.labelError = labelError;

}

@Override

public void handleError(String message) {

labelError.setVisible(true);

labelError.setText(message);

}

}

public class LogErrorStrategy implements ErrorStrategy {

@Override

public void handleError(String message) {

System.err.println(message);

}

}

public class Validation {

private ErrorStrategy errorStrategy;

public Validation(ErrorStrategy errorStrategy) {

this.errorStrategy = errorStrategy;

}

public String validateLogin(String login) {

if (!login.matches("^[a-zA-Z0-9]+$")) {

errorStrategy.handleError("Логин должен содержать только латинские буквы или цифры!");

return "error";

}

if (login.length() < 3 || login.length() > 25) {

errorStrategy.handleError("Логин должен быть от 3 до 25 символов!");

return "error";

}

return login;

}

public String validatePhoneNumber(String phoneNumber) {

if (phoneNumber.length() != 9 || !phoneNumber.matches("\\d+")) {

errorStrategy.handleError("Введите номер телефона с кодом!");

return "error";

}

return phoneNumber;

}

public String validateAndFormatString(String input) {

input = input.trim();

if (input.length() <= 3) {

errorStrategy.handleError("Строка должна быть длиннее 3 символов");

return "error";

}

if (!input.matches("[a-zA-Zа-яА-ЯёЁ\\s,]+")) {

Продолжение приложения В

errorStrategy.handleError("Строка должна содержать только буквы, пробелы или запятые");

return "error"; // Возвращаем "error" для указания ошибки

}

return input.substring(0, 1).toUpperCase() + input.substring(1).toLowerCase();

}

public double validateAndParseDouble(String input){

if (!input.matches("-?\\d+(\\.\\d+)?")) {

errorStrategy.handleError("Число должно содержать только цифры");

return -1000;

}

double value = Double.parseDouble(input);

if (value < -1000||value>1000) {

errorStrategy.handleError("Слишком маленькое или большое значение");

return -1000;

}

return value;

}

public int validateAndParseInt(String input) {

// Проверка на наличие букв (ни русских, ни английских)

if (!input.matches("-?\\d+(\\.\\d+)?")) {

errorStrategy.handleError("Число должно содержать только цифры");

return -1000; // Возвращаем ошибочный код

}

return Integer.parseInt(input.trim()); // Убираем пробелы

}

}

# Приложение Г

**(обязательное)**

**Листинг скрипта генерации базы данных**

`user`CREATE DATABASE weather\_data

CHARACTER SET utf8

COLLATE UTF8\_GENERAL\_CI;

USE weather\_data;

CREATE TABLE personal\_settings (

id INT AUTO\_INCREMENT PRIMARY KEY,

phone VARCHAR(9) CHARACTER SET utf8 COLLATE utf8\_general\_ci UNIQUE NOT NULL,

notifications BOOLEAN DEFAULT FALSE,

temperature VARCHAR(3) DEFAULT 'C',

pressure VARCHAR(15) DEFAULT 'мм рт. ст.',

speed VARCHAR(5) DEFAULT 'м/с'

) CHARACTER SET utf8 COLLATE utf8\_general\_ci;

CREATE TABLE user (

id INT AUTO\_INCREMENT PRIMARY KEY,

login VARCHAR(25) CHARACTER SET utf8 COLLATE utf8\_general\_ci UNIQUE NOT NULL,

password BLOB NOT NULL,

role VARCHAR(15) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL DEFAULT 'user',

settings\_id INT,

FOREIGN KEY (settings\_id) REFERENCES personal\_settings(id) ON DELETE CASCADE

) CHARACTER SET utf8 COLLATE utf8\_general\_ci;

CREATE TABLE location (

id INT AUTO\_INCREMENT PRIMARY KEY,

town VARCHAR(100) CHARACTER SET utf8 COLLATE utf8\_general\_ci UNIQUE NOT NULL,

country VARCHAR(100) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL

) CHARACTER SET utf8 COLLATE utf8\_general\_ci;

CREATE TABLE weather\_name (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL

) CHARACTER SET utf8 COLLATE utf8\_general\_ci;

CREATE TABLE weather\_parameters (

id INT AUTO\_INCREMENT PRIMARY KEY,

temperature DECIMAL(5, 2) NOT NULL,

pressure INT NOT NULL,

humidity INT NOT NULL,

precipitation DECIMAL(5, 2),

wind\_speed DECIMAL(5, 2) NOT NULL

) CHARACTER SET utf8 COLLATE utf8\_general\_ci;

CREATE TABLE day (

id INT AUTO\_INCREMENT PRIMARY KEY,

date DATE NOT NULL,

day\_weather INT,

night\_weather INT,

weather\_id INT,

location\_id INT,

Продолжение приложения Г

FOREIGN KEY (day\_weather) REFERENCES weather\_parameters(id) ON DELETE CASCADE,

FOREIGN KEY (night\_weather) REFERENCES weather\_parameters(id) ON DELETE CASCADE,

FOREIGN KEY (weather\_id) REFERENCES weather\_name(id) ON DELETE CASCADE,

FOREIGN KEY (location\_id) REFERENCES location(id) ON DELETE CASCADE

) CHARACTER SET utf8 COLLATE utf8\_general\_ci;

CREATE TABLE dashboard (

id INT AUTO\_INCREMENT PRIMARY KEY,

start\_date\_id INT NOT NULL,

end\_date\_id INT NOT NULL,

user\_id INT,

FOREIGN KEY (start\_date\_id) REFERENCES day(id) ON DELETE CASCADE,

FOREIGN KEY (end\_date\_id) REFERENCES day(id) ON DELETE CASCADE,

FOREIGN KEY (user\_id) REFERENCES user(id) ON DELETE CASCADE

) CHARACTER SET utf8 COLLATE utf8\_general\_ci;

DELIMITER $$

CREATE TRIGGER delete\_personal\_settings\_after\_user

AFTER DELETE ON user

FOR EACH ROW

BEGIN

DELETE FROM personal\_settings WHERE id = OLD.settings\_id;

END $$

DELIMITER ;

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначение | | | | | Наименование | | | Дополнитель-ные сведения | | | |
|  | | | | | Текстовые документы | | |  | | | |
|  | | | | |  | | |  | | | |
| БГУИР КР 1-40 05 01-12 003 ПЗ | | | | | Пояснительная записка | | | 52 с. | | | |
|  | | | | |  | | |  | | | |
|  | | | | |  | | |  | | | |
|  | | | | | Графические документы | | |  | | | |
|  | | | | |  | | |  | | | |
| ГУИР 502119 001 ПД | | | | | Описание процесса «Составить | | | Формат А4 | | | |
|  | | | | | прогноз погоды» | | |  | | | |
|  | | | | |  | | |  | | | |
| ГУИР 502119 002 ПЛ | | | | | Схема алгоритма функции | | | Формат А4 | | | |
|  | | | | | main() | | |  | | | |
|  | | | | |  | | |  | | | |
| ГУИР 502119 003 ПЛ | | | | | Плакат Информационная модель | | | Формат А4 | | | |
|  | | | | | базы данных | | |  | | | |
|  | | | | |  | | |  | | | |
| ГУИР 502119 004 ПЛ | | | | | Плакат Модели представления | | | Формат А4 | | | |
|  | | | | | программного средства | | |  | | | |
|  | | | | |  | | |  | | | |
| ГУИР 502119 005 ПЛ | | | | | Плакат Скриншоты рабочих окон | | | Формат А4 | | | |
|  | | | | | программного средства | | |  | | | |
|  |  |  |  |  | БГУИР КР 1-40 05 01-12 003 Д1 | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм. | Л. | № докум. | Подп. | Дата | Разработка автоматизированной системы построения прогноза погоды на основе статистических наблюдений  Ведомость курсовой  работы | Лит | | | | Лист | Листов |
| Разраб. | | Бурая |  |  |  | Т | |  | 1 | 1 |
| Пров. | | Снопок |  |  | Кафедра ЭИ  гр. 224402 | | | | | |
| Т.контр. | | Снопок |  |  |
| Рец. | |  |  |  |
| Н.контр | |  |  |  |
| Утв. | | Ефремов |  |  |
|  | |  |  |  |