



Министерство науки и **ВЫСШЕГО ОБРАЗОВАНИЯ** Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Уральский государственный экономический университет»
(УрГЭУ)

КУРСОВАЯ РАБОТА

по дисциплине «**Объектно-ориентированное программирование**»

на тему: **Разработка программного обеспечения с помощью
графической библиотеки JavaFX**

Институт непрерывного и
дистанционного образования

Направление подготовки
*09.03.01 Информатика и
вычислительная техника*

Направленность (профиль)
*Программное обеспечение
автоматизированных систем*
Кафедра
*Кафедра информационных
технологий и статистики*

Студент
*Разбойников Сергей
Андреевич*

Группа *ИНО ОЗБ ПОАС-
23*

Руководитель
*Панов Михаил
Александрович, к.э.н.,
доцент кафедры
информационных
технологий и статистики*

Екатеринбург
2024 г.

ВВЕДЕНИЕ.....	3
1. АНАЛИТИЧЕСКАЯ ЧАСТЬ.....	5
1.1. Описание предметной области	5
1.2. Словарь предметной области.....	8
1.3. Анализ существующих программных решений	12
1.4. Техническое задание	18
1.4.1. Функциональные требования	18
1.4.2. Требование к экранным формам	21
1.4.3. Модель данных.....	21
1.4.4. Нефункциональные требования	22
2. ПРОЕКТНАЯ ЧАСТЬ	24
2.1. Проектирование и разработка классов	24
2.2. Проектирование пользовательского интерфейса.....	25
2.3. Описание разработанных алгоритмов и программных модулей ..	28
2.3.1. Иерархическая структура приложения.....	28
2.3.2. Алгоритмы	29
2.3.3. Описание основных программных модулей	31
2.4. Тестирование программного комплекса	33
2.4.1. Описание методики тестирования	33
2.4.2. Результаты тестов и анализ	33
2.4. Руководство пользователя	37
ЗАКЛЮЧЕНИЕ	39
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ	40
ПРИЛОЖЕНИЕ А	43
ПРИЛОЖЕНИЕ Б.....	46
ПРИЛОЖЕНИЕ В	52
ПРИЛОЖЕНИЕ Г	53

ВВЕДЕНИЕ

Цель курсовой работы: получить опыт использования графической библиотеки JavaFX для создания программного обеспечения с использованием языка программирования Java.

Актуальность темы: тема получения данных о погоде в реальном времени является актуальной и важной в современном мире, где погода оказывает значительное влияние на повседневную жизнь, бизнес и туризм, точность и своевременность прогнозов погоды играют ключевую роль. Прогнозирование погоды на основе данных в реальном времени позволяет более точно определять тенденции, узнавать об атмосферных процессах и делать более обоснованные прогнозы. Это особенно важно для синоптиков, которые используют данные о текущей погоде и математическое моделирование для предсказания изменений погоды в будущем.

Тема о нахождении основных данных погоды в реальном времени остается актуальной благодаря постоянному развитию технологий и необходимости точных прогнозов для обеспечения безопасности и эффективности деятельности в различных сферах жизни.

Задачи:

- Исследовать и анализировать существующие решения в области разработки программного обеспечения.
- Определить требования и спецификации для программы.
- Разработать алгоритмы и логику работы для нового приложения.
- Спроектировать и реализовать пользовательский интерфейс для нового программного приложения.
- Провести тестирование программного продукта для обеспечения корректной работы программы.

Объектом исследования: библиотека JavaFX.

Предметом исследования: программа для получения данных о погоде.

Краткое изложение структуры курсовой работы:

Первая часть работы посвящена исследованию и анализу предметной области, определения требований, которым должна соответствовать программа. Вторая часть работы посвящена разработке алгоритмов, интерфейса и проектированию приложения, предназначенного для получения данных о погоде в разных городах мира.

Решаемые задачи:

Сбор данных о погоде: программа собирает актуальную информацию о погоде, включая температуру, её максимальные и минимальные значения, влажность, атмосферное давление, скорость ветра.

Предоставление данных в удобном формате: данные о погоде представлены в текстовом формате, что позволяет пользователям легко интерпретировать информацию.

Обновление данных в реальном времени: программа обеспечивает актуальность данных, обновляя информацию о погоде в реальном времени.

1. АНАЛИТИЧЕСКАЯ ЧАСТЬ

1.1. Описание предметной области

До начала разработки и написания кода важно тщательно изучить предметную область, которая будет играть ключевую роль в вашем проекте. Под предметной областью программирования можно понимать создание программного обеспечения, направленного на решение задач в определенной сфере деятельности, при этом используя общие наборы данных и знаний.

В данной курсовой работе, основной предметной областью является библиотека JavaFX. JavaFX – библиотека, которая предоставляет инструменты для разработки кроссплатформенных графических приложений на Java. Эта библиотека оборудована мощными инструментами и дополнительными библиотеками, что позволяет разработчикам создавать более читаемый и компактный исходный код.

Библиотека JavaFX доступна в виде общедоступного интерфейса программирования приложений Java (API)

JavaFX обладает рядом особенностей, делающих его идеальным выбором для создания сложных клиентских приложений:

JavaFX разработан на Java, что позволяет вам воспользоваться всеми преимуществами Java, включая многопоточность, обобщения и лямбда-выражения. Вы можете использовать любой Java-интерфейс. Выбор IDE, например, NetBeans или Eclipse, для разработки, компиляции, запуска, отладки и упаковки вашего приложения JavaFX остается за вами.

JavaFX поддерживает привязку данных с помощью своих библиотек.

Код JavaFX может быть написан на любом языке сценариев, поддерживаемом виртуальной машиной Java (JVM), таком как Kotlin, Groovy и Scala.

Интеграция с Swing: Текущие приложения, разработанные на Swing, могут быть модернизированы, используя возможности JavaFX, включая продвинутые графические эффекты для воспроизведения медиа и встроенные веб-элементы. Введение класса `SwingNode` в JavaFX 8 позволяет разработчикам интегрировать компоненты Swing в приложения JavaFX, обеспечивая более широкие возможности для разработки.

FXML: FXML в JavaFX служит для создания пользовательского интерфейса UI приложения. Этот подход упрощает разработку и обслуживание приложений JavaFX. FXML представляет собой язык разметки на основе XML, который позволяет описывать структуру и визуальное оформление интерфейса пользователя в виде иерархической структуры XML. Этот метод идеально подходит для определения пользовательского интерфейса JavaFX-приложения, так как иерархическая структура XML-документа тесно соответствует структуре сцены JavaFX.

Scene Builder: в JavaFX Scene Builder представляет собой мощный инструмент для визуального проектирования интерфейсов в JavaFX, позволяющий разработчикам создавать графические интерфейсы пользователя (GUI) с помощью перетаскивания компонентов. После завершения проектирования, Scene Builder автоматически генерирует исходный код FXML, который затем можно интегрировать в приложение. Этот процесс обеспечивает удобство и скорость разработки, позволяя разработчикам сосредоточиться на логике приложения, а не на деталях реализации интерфейса.

Встроенные элементы управления пользовательского интерфейса: в JavaFX представляют собой набор компонентов, предоставляемых JavaFX API для создания графического интерфейса пользователя. Эти элементы позволяют разработчикам создавать интерактивные приложения с различными функциями, включая ввод данных, выбор из списка, отображение текста и многое другое. Вот перечисление некоторых встроенных элементов управления пользовательского интерфейса JavaFX: Label, Button, Text Field.

JavaFX представляет собой мощный инструмент для разработки кроссплатформенных приложений, благодаря своей способности работать в различных операционных системах. Это означает, что приложения, созданные с использованием JavaFX, могут быть доступны для пользователей на любой платформе, поддерживающей Java, что значительно расширяет потенциальную аудиторию. Кроме того, JavaFX обеспечивает интеграцию с другими технологиями, такими как облачные вычисления и искусственный интеллект, что позволяет разработчикам создавать более продвинутые и интерактивные приложения, открывая новые горизонты для инноваций и сотрудничества.

Область Применения:

В эпоху цифровизации, когда многие процессы перешли из традиционных компьютерных систем в облачные и веб-решения, десктопные приложения продолжают играть важную роль. Они отличаются от своих онлайн-аналогов более развитым графическим интерфейсом, высокой эффективностью и надежностью, и в некоторых случаях просто не имеют замены. Сфера применения десктопных приложений охватывает широкий спектр:

- Интегрированные среды разработки (IDE).
- Инструменты для работы с графикой и текстом.
- Редакторы и проигрыватели аудио и видео контента.

— Некоторые компьютерные игры.

Также стоит отметить, что многие популярные мобильные и веб-приложения, такие как Skype, WhatsApp, Telegram и Slack, имеют версии для персонального компьютера. Это подчеркивает, что десктопные приложения остаются актуальными и важными в современном мире. JavaFX, как инструмент для разработки, продолжает быть ценным ресурсом для разработчиков.

Однако JavaFX не ограничивается только разработкой для персонального компьютера. Например, приложения, созданные с использованием JavaFX, могут быть адаптированы для работы на мобильных устройствах. Компания Gluon предоставляет необходимые инструменты для этого, а Axiom NIK предлагает удобные решения для интеграции технологии нативных образов в проекты на Java. JavaFX также поддерживается встроенными системами, например такими как Raspberry Pi, что делает его идеальным выбором для разработки приложений с графическим интерфейсом для этих устройств.

1.2. Словарь предметной области

Словарь предметной области – это совокупность терминов, понятий, сущностей и процессов, которые характерны для конкретной области знаний или деятельности. Этот словарь служит для точного и полного описания предметной области, что важно при разработке программного обеспечения, систем автоматизации и других проектов, где требуется точное понимание и отображение специфики предметной области.

Java – это объектно-ориентированный язык программирования, комбинирующий данные и определенные функции обработки этих данных. Java сконструирован таким образом, что программист может создавать апплеты (applets) – крохотные Java-программки, каждая из которых выполняет одну небольшую функцию.

Объектно-ориентированное программирование (ООП) - подход к созданию программ, основанный на использовании классов и объектов, взаимодействующих между собой.

Объект – это мыслимая или реальная сущность, обладающая характерным поведением и отличительными характеристиками и являющаяся важной в предметной области.[2]

Класс – это модель для создания объектов определённого типа. Класс описывает структуру объектов и определяет алгоритмы для работы с этими объектами. В ООП класс служит средством для введения абстрактных типов данных в программный проект, определяя одновременно как интерфейс, так и реализацию для всех своих экземпляров (т.е. объектов).

Интерфейс – это абстрактный тип, который определяет набор методов, которые должен реализовать класс.

Абстракция – это ключевой принцип, позволяющий скрыть детали реализации объектов и системы, представляя их через абстрактные классы и интерфейсы.

Инкапсуляция – это принцип, который подразумевает сокрытие внутренних данных компонента и деталей его реализации от других компонентов приложения, предоставляя набор методов для взаимодействия с ним.

Полиморфизм – это принцип, который позволяет вызывать переопределённый метод через переменную класса-родителя, получая поведение, соответствующее реальному классу-потомку. Он тесно связан с наследованием, которое помогает реализовать полиморфизм.

Наследование – это принцип, которое позволяет создавать новые классы на основе существующих, наследуя их свойства и методы. Это упрощает разработку и поддержку программ, позволяя избежать дублирования кода.

Модификаторы доступа – это ключевые слова в программировании, которые определяют уровень доступности членов класса или типов. Они позволяют контролировать, кто и как может использовать определенные части кода.

Public – члены класса, объявленные как `public`, доступны из любого места программы, где видимо определение класса. Это означает, что они могут быть доступны извне класса, а также из производных классов.

Private – члены класса, объявленные как `private`, доступны только внутри класса, который их объявляет. Это означает, что они не могут быть доступны извне класса или из производных классов.

Protected – члены класса, объявленные как `protected`, доступны внутри класса, который их объявляет, а также в его производных классах. Это означает, что они не могут быть доступны извне класса, но могут быть доступны из производных классов.

Scene Builder – представляет собой мощный инструмент для визуального проектирования пользовательских интерфейсов (UI) в приложениях на JavaFX, доступный бесплатно и с открытым исходным кодом. Этот инструмент позволяет разработчикам создавать интерфейсы с помощью FXML, формата разметки, основанного на XML, что обеспечивает гибкость и удобство в разработке. Scene Builder облегчает процесс разработки, позволяя отделять дизайн пользовательского интерфейса от логики приложения, что упрощает интеграцию и связывание интерфейса с кодом приложения.

Fork eXtensible Markup Language – это формат, основанный на XML, который используется для определения пользовательского интерфейса в приложениях JavaFX. Представляет собой альтернативу разработке пользовательских интерфейсов с использованием процедурного кода, позволяя разработчикам отделить дизайн пользовательского интерфейса от его логики.

Контейнеры – это панели компоновки в JavaFX, которые используются для организации и размещения элементов управления в приложении. Все панели компоновки наследуются от класса `javafx.scene.layout.Pane`, который, в свою очередь, наследуется от `Region`.

Stage – класс в JavaFX, является основным контейнером для создания графического интерфейса пользователя (GUI). Он представляет окно приложения и служит контейнером для всех компонентов интерфейса. Класс Stage наследуется от класса `Window`, который определяет базовые возможности окна приложения.

VBox – является компонентом макета, который располагает все свои дочерние узлы (компоненты) в вертикальном столбце, одно за другим. Это означает, что элементы внутри VBox будут расположены вертикально друг под другом.

BorderPane – контейнер, который позволяет размещать компоненты в пяти различных областях: верхней, нижней, левой, правой и центральной.

Элементы управления – в JavaFX элементы управления представляют собой разнообразные компоненты, служащие для создания графического интерфейса пользователя (GUI). Эти элементы управления являются объектами и используются для взаимодействия с пользователем, обработки событий и отображения данных.

Кнопки и метки (Buttons and Labels) – кнопки (Button) используются для выполнения действий при нажатии, а метки (Label) для отображения текста.

Текстовые элементы управления – включают TextField для однострочного ввода текста и TextArea для многострочного ввода.

ListView – Отображает список элементов, которые могут быть выбраны.

1.3. Анализ существующих программных решений

Анализ существующих программных решений в программировании – это процесс изучения и оценки различных программных продуктов, инструментов и технологий, используемых в разработке программного обеспечения. Этот процесс включает в себя изучение функциональности, производительности, надежности, удобства использования и других аспектов программных продуктов, чтобы определить, какие из них наиболее подходят для конкретных задач и требований проекта. Можно сказать, что анализ существующих программных решений является важным этапом в процессе разработки программного обеспечения. Это помогает командам разработчиков выбрать наиболее подходящие инструменты и технологии, которые могут ускорить процесс разработки, улучшить качество конечного продукта и сделать его более эффективным и надежным.

Abstract Window Toolkit – представляет собой первоначальный инструментарий для разработки графических интерфейсов в Java, предшествующий Swing. Этот набор инструментов, входящий в состав Java Foundation Classes (JFC), обеспечивает базовый API для создания графических интерфейсов в Java-приложениях. AWT также находит применение в определенных профилях Java ME, таких как Connected Device Configuration, что подчеркивает его роль в поддержке Java на мобильных устройствах.

Преимущества AWT:

- Интеграция с нативными элементами ОС: AWT обеспечивает доступ к нативным элементам пользовательского интерфейса, что позволяет разработчикам создавать приложения, соответствующие специфике операционной системы.
- Мощная система обработки событий: AWT предлагает эффективную систему обработки событий, упрощая разработку интерактивных приложений.
- Инструменты для работы с графикой и изображениями: Включает в себя классы для работы с формами, цветами и шрифтами, что способствует созданию визуально привлекательных интерфейсов.
- Менеджеры компоновки: AWT предоставляет инструменты для гибкой организации окон, не зависящие от размера окна или разрешения экрана.

Недостатки AWT:

- Тяжеловесность компонентов: Компоненты AWT могут быть "тяжёлыми", так как они тесно связаны с элементами интерфейса операционной системы, что может вызвать различия в внешнем виде и функционале приложений на разных платформах.
- Зависимость от ОС: AWT зависит от нативных компонентов операционной системы, что может привести к проблемам совместимости и внешнему виду приложений.
- Ограниченность интерфейса: в сравнении с более современными инструментариями, такими как Swing, AWT может предлагать более простой и ограниченный набор функций для создания интерфейса.

Сегодняшний мир программирования на Java практически игнорирует AWT, оставляя его на пороге истории, где он встречается лишь в устаревших проектах и апплетах. Oracle активно отталкивает разработчиков от использования

AWT, предлагая вместо этого более современные и безопасные альтернативы, такие как Swing.

Swing – после успешного запуска AWT, Sun Microsystems представила инновационный набор графических элементов, известный как Swing. Эти компоненты полностью разработаны на языке Java, что обеспечивает их кросс-платформенную совместимость. Для визуализации используется 2D-графика, что открывает широкие возможности для создания визуально привлекательных интерфейсов.

Swing значительно расширяет функциональность и разнообразие стандартных компонентов по сравнению с AWT, позволяя разработчикам легко создавать новые элементы, наследуясь от существующих и применяя любые изображения. Одной из ключевых особенностей Swing является поддержка различных стилей и скинов, что делает его идеальным выбором для создания современных и адаптивных пользовательских интерфейсов. Однако, несмотря на все преимущества, первые версии Swing были известны своей медленной производительностью, что могло привести к зависанию системы при неправильной реализации программы.

Преимущества Swing:

— Многоплатформенность: Swing обеспечивает кроссплатформенность, позволяя разработчикам создавать приложения, работающие на различных ОС без необходимости изменения кода. Это дает возможность сосредоточиться на логике приложения, а не на специфике платформы.

— Глубокая интеграция с Java: swing тесно связан с Java, что позволяет использовать все преимущества языка, включая его мощные возможности для работы с объектами и коллекциями.

— Расширенный набор компонентов: swing предлагает обширный выбор компонентов для создания пользовательского интерфейса, что способствует разработке сложных и функциональных приложений.

Недостатки Swing:

— Производительность: swing может быть менее эффективным по сравнению с некоторыми другими фреймворками для создания GUI, особенно при работе с большими объемами данных или сложными графическими элементами.

— Сложности в стилизации: swing может представлять большую сложность в стилизации по сравнению с другими фреймворками из-за использования нативных компонентов ОС, которые могут отличаться по внешнему виду и поведению на разных платформах.

— Обновления и поддержка: несмотря на то, что Swing является частью стандартной библиотеки Java, его развитие и поддержка могут быть не такими активными, как у некоторых других фреймворков.

В итоге, Swing остается мощным и гибким инструментом для создания GUI в Java, но разработчики должны учитывать его потенциальные недостатки при выборе фреймворка для своих проектов.

Standard Widget Toolkit – SWT, созданный в IBM, был запущен, когда Swing еще не достиг своего пика, и его основная цель - продвижение среды разработки Eclipse. В отличие от AWT, SWT использует нативные компоненты операционной системы, что позволяет ему адаптироваться к различным платформам, каждая из которых требует своей версии JAR-библиотеки. Это обеспечивает более глубокую интеграцию с операционной системой, но также увеличивает сложность разработки, так как требуется предоставлять отдельные реализации для каждой платформы.

Достоинства SWT:

- Производительность: SWT использует нативные элементы управления, что обеспечивает более высокую производительность по сравнению с Swing.

- Нативный вид: Приложения, созданные с использованием SWT, имеют более нативный вид на различных платформах, что улучшает пользовательский опыт.

- Поддержка Eclipse: SWT тесно интегрирована с Eclipse, что делает его предпочтительным выбором для разработки приложений, которые должны быть частью Eclipse.

Недостатки SWT:

- Отсутствие поддержки некоторых компонентов: В отличие от Swing, SWT не поддерживает все компоненты, доступные в Swing, что может ограничивать разработчиков в создании некоторых типов интерфейсов.

- Сложность в разработке: SWT может быть сложнее в использовании по сравнению с Swing, особенно для новичков, из-за его более низкоуровневого подхода к созданию GUI.

- Отсутствие активной поддержки: В отличие от Swing, который активно поддерживается и развивается в рамках проекта OpenJDK, SWT может не получать такого же уровня поддержки и обновлений.

Выбор между SWT и Swing зависит от конкретных требований проекта, предпочтений разработчика и требований к производительности и внешнему виду приложения.

JavaFX – представляет собой революционный инструмент в мире разработки программного обеспечения, обеспечивающий высокопроизводительную отрисовку с использованием графического конвейера, что значительно ускоряет выполнение приложений. Эта платформа обладает богатым набором встроенных компонентов, включая специализированные

элементы для визуализации данных, поддерживает мультимедийные контент и анимацию. Стилизация внешнего вида компонентов достигается с помощью CSS, что делает процесс разработки более гибким и эффективным.

Достоинства JavaFX:

- Высокая производительность благодаря использованию графического конвейера;
- Широкий выбор компонентов для разработки;
- Возможность применения стилей для настройки внешнего вида приложения;
- Инструменты для создания установщиков, упрощающие процесс развертывания;
- Гибкость в использовании: приложение может быть запущено как настольное, так и в веб-браузере как часть веб-страницы.

Недостатки JavaFX:

- Поскольку JavaFX активно развивается, в нем могут возникать сбои.
- JavaFX не получил широкого признания у разработчиков.

В целом, JavaFX представляет собой мощный инструмент для разработки насыщенных клиентских приложений с современным пользовательским интерфейсом, поддерживаемый активным сообществом разработчиков. Однако для новичков или разработчиков, привыкших к Swing, может потребоваться время на освоение нового API и подходов к разработке GUI.

Выбор между JavaFX, Swing, AWT и SWT зависит от конкретных требований к проекту, включая производительность, богатство компонентов пользовательского интерфейса, простоту использования, совместимость с современными стандартами Java, поддержку сообщества и совместимость с платформами. Понимание сильных и слабых сторон каждого фреймворка

поможет сделать обоснованный выбор, обеспечивая высококачественный пользовательский интерфейс и оптимальную производительность.

1.4. Техническое задание

1.4.1. Функциональные требования

Функциональные требования – это детальное описание того, как система должна работать, включая взаимодействие пользователя с системой и ожидаемые результаты этих взаимодействий. Они важны для понимания целей и ожиданий от разработки программного обеспечения.

Список функциональных требований, которые следует учитывать при разработке программы:

- Интуитивно понятное взаимодействие с программой: создание интерфейса, который легок в понимании и ориентировании, позволяя пользователям без особых усилий взаимодействовать с программой.

- Визуализация иконок: не смотря на текст, в котором будет написана основная информация, пользователь должен интуитивно понимать, где именно перед ним представлены определённые данные.

- Удобное взаимодействие с программой: у пользователя должна быть возможно закрыть программу, свернуть её и переместить меню запущенной программы на любой участок экрана Windows.

- Обновление данных: данные, которые будет получать пользователь, должны быть актуальными. Для достижения этой цели необходимо регулярно обновлять информацию, используя получения данных из сети за счет REST API запросов.

Схема интерфейса приложения представлена на рисунке 1.

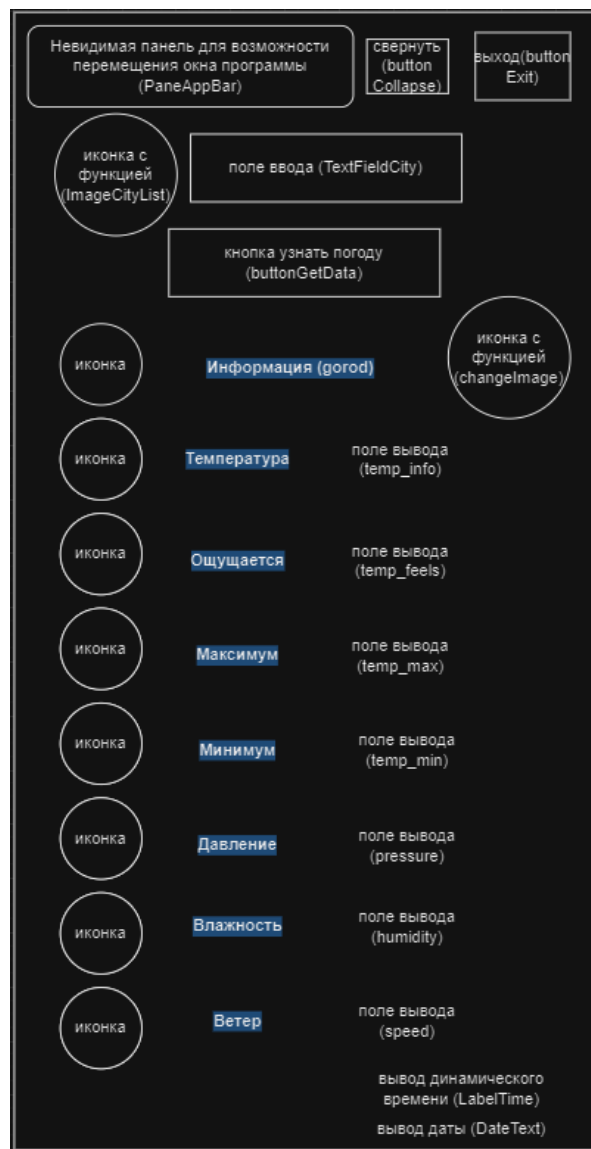


Рисунок 1 – Схема интерфейса приложения¹

Описание основных элементов интерфейса:

PaneAppBar – невидимая панель для возможности перемещения окна программы.

button Collapse – кнопка для возможности сворачивания программы.

¹ Составлено автором по: [15]

button Exit – кнопка для выхода из программы.

ImageCityList – иконка, которая содержит в себе следующую функцию: при нажатии на иконку в поле для ввода (TextFieldCity) появляется название города, которое задано в списке.

TextFieldCity – поле для ввода названия любого города.

buttonGetData – кнопка, в которой прописан основной функционал программы, при нажатии на кнопку, происходит запрос данных, если название города, введенного в поле (TextFieldCity), соответствует с названием из сервиса, тогда приложение отображает данные о погоде в полях для вывода.

changeImage – иконка, которая меняется с определенным погодным условием в городе.

gorod – поле для вывода названия города.

temp_info – поле для вывода данных о температуре.

temp_feels – поле для вывода данных о ощущении температуры.

temp_max – поле для вывода данных о максимальной температуре.

temp_min – поле для вывода данных о минимальной температуре.

pressure – поле для вывода данных о давлении.

humidity – поле для вывода данных о влажности.

speed – поле для вывода данных о скорости ветра.

LabelTime – поле для вывода текущего времени в котором каждую секунду обновляется данные.

DateText – поле для вывода текущей даты.

1.4.2. Требование к экранным формам

Требования к экранным формам обычно относятся к стандартам и правилам, которые должны соблюдаться при разработке и использовании экранных форм в различных контекстах, таких как веб-разработка, мобильная разработка и разработка программного обеспечения.

Экранные формы — это интерфейс пользователя, который позволяет пользователям взаимодействовать с программным обеспечением или веб-сайтом, вводя данные, выбирая опции и выполняя другие действия.

Экранные формы пользовательского интерфейса программы должны быть выполнены в едином графическом дизайне, с одинаковым расположением основных элементов управления и навигации. Элементы управления сворачивания и закрытия приложения должны быть в левом верхнем углу программы, основные элементы управления расположены по центру.

Для обозначения данных, должен использоваться не только текст, но и графические иконки.

Общий цвет, как информационной части, так и фона, должен быть спокойных тонов, не вызывающих усталости пользователя при работе с ним.

Текущие данные о времени и дате должны проставляться автоматически и быть расположены в правом нижнем углу программы.

1.4.3. Модель данных

Модель данных — это абстрактное, самодостаточное, логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактную машину доступа к данным, с которой взаимодействует пользователь.

В данной курсовой работе используется формат передачи данных JSON

JSON (JavaScript Object Notation) — это текстовый формат обмена данными, основанный на JavaScript, но при этом независимый от него и используемый в любом языке программирования. JSON часто используется в REST API, поскольку он легко читается как человеком, так и машиной, и обычно меньше весит, чем XML [12]

Основные элементы JSON включают:

- JSON-объект: Неупорядоченное множество пар "ключ: значение", заключённое в фигурные скобки {}. Ключи в объекте являются строками, а значения могут быть различных типов: числами, булевыми значениями (true или false), null, другими объектами, массивами или строками.
- Массив: Упорядоченный набор значений, разделённых запятыми, и заключённый в квадратные скобки []. Массивы могут содержать значения любого типа, включая другие объекты и массивы.
- Числа: Целые или вещественные числа.
- Литералы: true, false, и null.
- Строки: Текстовые значения, заключённые в двойные кавычки "".

1.4.4. Нефункциональные требования

Нефункциональные требования к программному обеспечению — это требования, которые описывают характеристики системы, не связанные напрямую с конкретными функциями, которые она выполняет. Они включают в себя различные аспекты, такие как производительность, удобство использования, внешние интерфейсы и другие параметры, которые определяют качество и эффективность программного обеспечения.

Нефункциональные требования к программе:

- Доступность и удобство в применении.

- Программа должна предоставлять данные о погоде в удобном для пользователя формате.
- Программа должна обеспечивать быстрый доступ к данным о погоде.
- Программа должна обеспечивать стабильную работу, минимизируя время простоя и обеспечивая доступность данных о погоде в любое время.

2. ПРОЕКТНАЯ ЧАСТЬ

2.1. Проектирование и разработка классов

Данное приложение состоит из классов *HelloApplication.java* и *HelloController.java*

Основным классом является *HelloController.java*, в нём находятся два метода, а именно *start* и *ShowModal*. В методе *ShowModal* создается загрузочный экран приложения, настраивается его местоположение. В методе *start* является входной точкой приложения, этот метод загружает загрузочный экран приложения (*ShowModal*), после и саму XML-структуру, которая была разработана в SceneBuilder.

Так как использовался JavaFX, то вся функциональность приложения находится в классе-контроллере *HelloController*. В этом классе:

- содержатся переменные, которые ссылаются на объекты интерфейса по определённому идентификатору из SceneBuilder.
- происходит обращение по URL адресу и получение данных с сайта OpenWeatherMap – онлайн сервис, который предоставляет платный API для доступа к данным о текущей погоде.
- содержатся методы для вывода даты, времени, иконок с функционалом и т.д.

Таким образом класс *HelloController* используется для управления поведением объектов, определенных в FXML документе. Этот класс ассоциируется с FXML документом через атрибут *fx:controller* и отвечает за координацию действий пользовательских интерфейсных элементов. Контроллер может содержать методы обработки событий, которые вызываются при определенных действиях пользователя, например, при нажатии кнопки.

2.2. Проектирование пользовательского интерфейса

Проектирование пользовательского интерфейса включает в себя несколько ключевых этапов, начиная от концептуального проектирования и заканчивая физическим дизайном. Этот процесс направлен на создание эффективного и удобного взаимодействия пользователя с приложением или сервисом.

Требования к внешнему виду пользовательского интерфейса:

- Пользовательский интерфейс должен быть интуитивно понятным. Используя стандартные элементы управления и иконки, которые пользователи уже знают и ожидают увидеть в различных приложениях и веб-сайтах. Это помогает ускорить процесс адаптации и уменьшить время, необходимое для изучения нового интерфейса
- Использование графики и визуальных элементов может улучшить внешний вид и восприятие пользовательского интерфейса.
- Спокойные и нейтральные цвета иконок и самого фона приложения, не отвлекающие пользователя.
- Стилль текста должен быть единым. Это включает в себя не только текст, но и изображения, цвета, шрифты и другие элементы дизайна.



Рисунок 2 – иконка приложения [14]



Рисунок 3 – экран загрузки приложения²

² Составлено автором

Для создания интерфейса программы был использован инструмент SceneBuilder. SceneBuilder представляет собой программу, позволяющую легко перетаскивать компоненты пользовательского интерфейса JavaFX и настраивать их визуальное представление с помощью файлов fxml.



Рисунок 4 – интерфейс приложения³

³ Составлено автором

2.3. Описание разработанных алгоритмов и программных модулей

2.3.1. Иерархическая структура приложения

Так как при разработке приложения использовался только два класса `HelloApplication` и `HelloController`, работающий с FXML формами, иерархическая структура приложения довольно проста. Базовая структура:

`HelloApplication`: это основной класс приложения, который расширяет `Application` из `JavaFX`. Он отвечает за жизненный цикл приложения, включая инициализацию и отображение главного окна. В методе `start` загружается FXML файл, который определяет пользовательский интерфейс приложения, используя `FXMLLoader`.

`HelloController`: класс связан с FXML файлом и содержит логику обработки событий пользовательского интерфейса. Он содержит методы, которые реагируют на действия пользователя, такие как нажатия кнопок. В данном случае, `HelloApplication` является единственным классом, который работает с FXML, логика обработки событий находится именно в этом контроллере.

FXML файлы: это XML файлы, которые определяют структуру пользовательского интерфейса приложения. Они содержат различные элементы пользовательского интерфейса, такие как кнопки, текстовые поля и иконки. В данном курсовом проекте использовался только один FXML файл, он содержит все элементы пользовательского интерфейса, которые необходимо было отобразить.

2.3.2. Алгоритмы

В программе для мониторинга погоды в разных городах используются следующие алгоритмы:

Обращение по URL - позволяет получить содержимое веб-страницы по заданному URL и вернуть его в виде строки. Код демонстрирует базовый подход к чтению содержимого веб-страницы по заданному URL и обработке возможных исключений, используя классы URL, URLConnection, BufferedReader и InputStreamReader из пакета java.net и java.io.

Листинг 1. метод getUrlContent класса HelloController:

```
public String getUrlContent(String urlAddress){
    StringBuffer content = new StringBuffer();
    try { // Обработка исключений используется для обработки все
возможных исключений
        URL url = new URL(urlAddress);
        URLConnection urlConn = url.openConnection(); // Открытие
соединения
        BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(urlConn.getInputStream())); // Чтение данных
        String line;
        while ((line = bufferedReader.readLine()) != null) //
считывание строк
        {
            // Каждая считанная строка добавляется в StringBuffer
content с добавлением символа новой строки
            content.append(line + "\n");
        }
        bufferedReader.close(); // Закрытие потока
    } catch (Exception e)
    {
        gorod.setText("Такого Города Нет!"); // Конструкция для
проверки на корректный ввод города
    }
    return content.toString(); // возвращается строковое
представление содержимого StringBuffer content, полученного из URL
}
```

Формирование строки запроса: для получения информации о погоде используется API OpenWeatherMap. Объект JSONObject, позволяет легко работать с данными в формате JSON. Строка запроса формируется с использованием параметров, таких как название города которое ввел или выбрал пользователь(getUserCity), температура (temp), ощущаемая температура (feels_like), максимальная и минимальная температура (temp_max и temp_min), давление (pressure), влажность (humidity), скорость ветра (speed) и функции GetCurrentWeatherIcon для получения иконки погоды.

Листинг 2. Обработчик событий setOnActioninitialize метода initialize() класса HelloController:

```
void initialize() {
    getData.setOnAction(event -> {
        String getUserCity = city.getText().trim();
        gorod.setText(getUserCity);
        if(!getUserCity.equals("")) { // Получение Данных с сайта
            String output =
getUrlContent("https://api.openweathermap.org/data/2.5/weather?q="
+ getUserCity +
"&appid=80442090e9bad59438b86babdd86afd8&units=metric");
            if (!output.isEmpty()) {
                JSONObject obj = new JSONObject(output); //
преобразование данных JSON формата
                double receivedTemp =
obj.getJSONObject("main").getDouble("temp"); // вывод необходимых
данных (температура)
                temp_info.setText("" + receivedTemp);
                temp_feels.setText("" +
obj.getJSONObject("main").getDouble("feels_like")); // Как ощущается
температура
                temp_max.setText("" +
obj.getJSONObject("main").getDouble("temp_max")); // Максимальная
температура
                temp_min.setText("" +
obj.getJSONObject("main").getDouble("temp_min")); // Минимальная
температура
                pressure.setText("" +
obj.getJSONObject("main").getDouble("pressure")); // Давление
```

```

        humidity.setText(" " +
obj.getJSONObject("main").getDouble("humidity")); // Влажность
        speed.setText(" " +
obj.getJSONObject("wind").getDouble("speed")); // Скорость ветра
        GetCurrentWeathrIcon(obj); //погодная иконка
        (дождь, снег, солнце, туман)
        city.clear();
    }
}
});

```

Важно отметить, что для работы с API OpenWeatherMap требуется ключ API, который вставляется в URL запроса. В приведенном коде используется API 80442090e9bad59438b86babdd86afd8, который, является ключом, обрабатывающий 1000 запросов в сутки.

2.3.3. Описание основных программных модулей

Основные программные модули представляют собой независимые и самодостаточные единицы программного кода, содержащие определенный набор функций, классов, переменных и других элементов программы. Они служат для организации и управления кодом, обеспечивая модульность и повторное использование кода. Модули разделяют программу на логические блоки, каждый из которых выполняет определенную функцию, упрощая понимание и поддержку кода.

Для создания программы использовались такие классы как HelloApplication в котором прописан метод на загрузку приложения и HelloController в котором прописаны основные методы.

Основными методами класса HelloController являются getUrlContent и initialize. Эти методы подробно были разобраны в пункте 2.3.2.

Немаловажным методом является `GetCurrentWeathrIcon`, расположенный в `HelloController`. Благодаря нему происходит получение идентификатора иконки погоды из JSON-объекта, создается URL для этой иконки, загружается изображение по URL и устанавливается в качестве текущего изображения для компонента пользовательского интерфейса.

Листинг 3. Метода `GetCurrentWeathrIcon` класса `HelloController`:

```
private void GetCurrentWeathrIcon(JSONObject obj) { // получение
иконки с погодой
    String icon =
obj.getJSONArray("weather").getJSONObject(0).getString("icon");
//пременная в которой хранится название иконки
    String iconURL = "https://openweathermap.org/img/wn/" + icon
+"@2x.png"; //пременная в которой хранится иконка, которая была
выбрана исходя из названия в icon
    Image image = new Image(iconURL); //помещаем иконку в image
    changeImage.setImage(image); //помещаем иконку в changeImage
созданную в SceneBuilder
}
```

Остальными методами класса `HelloController` являются следующие:

- `InitAppBar` – возможность передвигать программу.
- `Exit` – кнопка выхода из программы.
- `Collapse` – кнопка предоставляет возможность свернуть окно программы.
- `chooseCity` – при нажатии на кнопку появляется название города из списка, всего городов 19, этот метод создан, чтобы можно было быстро протестировать программу, не вводя с клавиатуры название определенного города.
- `Time` – выводит в приложении текущую дату.
- `DynamicTime` – выводит в приложении текущее время.

2.4. Тестирование программного комплекса

2.4.1. Описание методики тестирования

В приложении были применены следующие методы тестирования:

- Функциональное.
- Тестирование удобства пользователя.
- Тестирование совместимости.

Функциональное тестирование проверяет соответствие программного обеспечения исходным функциональным требованиям, а тестирование удобства пользователя оценивает удобство использования, обучаемость и привлекательность продукта для пользователей.

Тестирование оценило приложение по производительности, эффективности, правильности, активизации в памяти и эмоциональной реакции пользователя. Проверялась совместимость с операционными системами, отличными от Windows.

2.4.2. Результаты тестов и анализ

Способность открытия приложения:

После загрузочного экрана приложение открывается и отображает все элементы управления, это можно понять, посмотрев на рисунок 5.

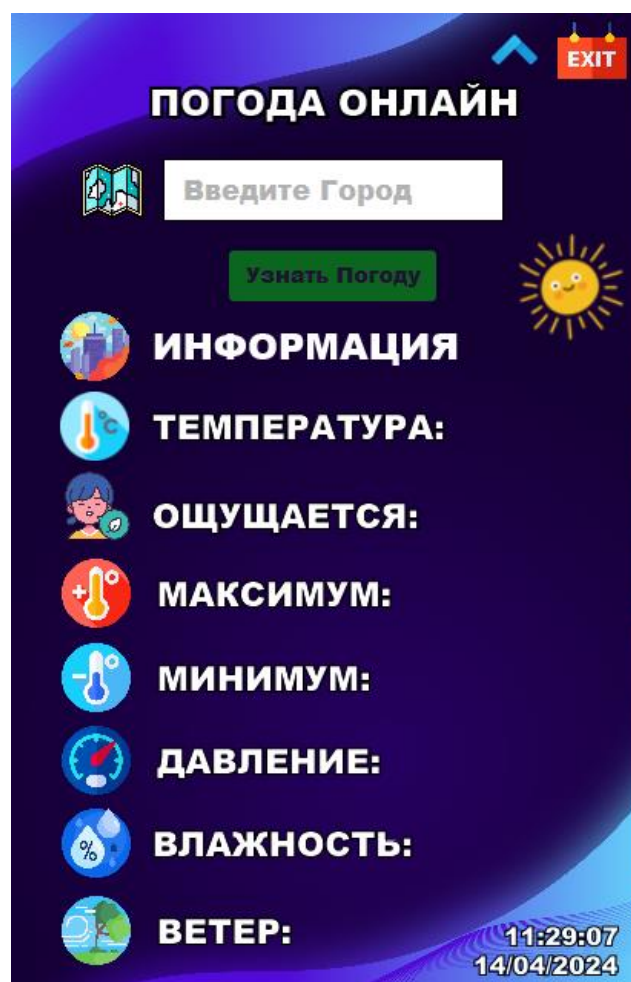


Рисунок 5 – открытое приложение⁴

Проверка работы основного функционала приложения, на картинке 6 можно увидеть, как пользователь вводит название города и после нажатия на кнопку “узнать погоду”, программа выводит нужную информацию.

⁴ Составлено автором

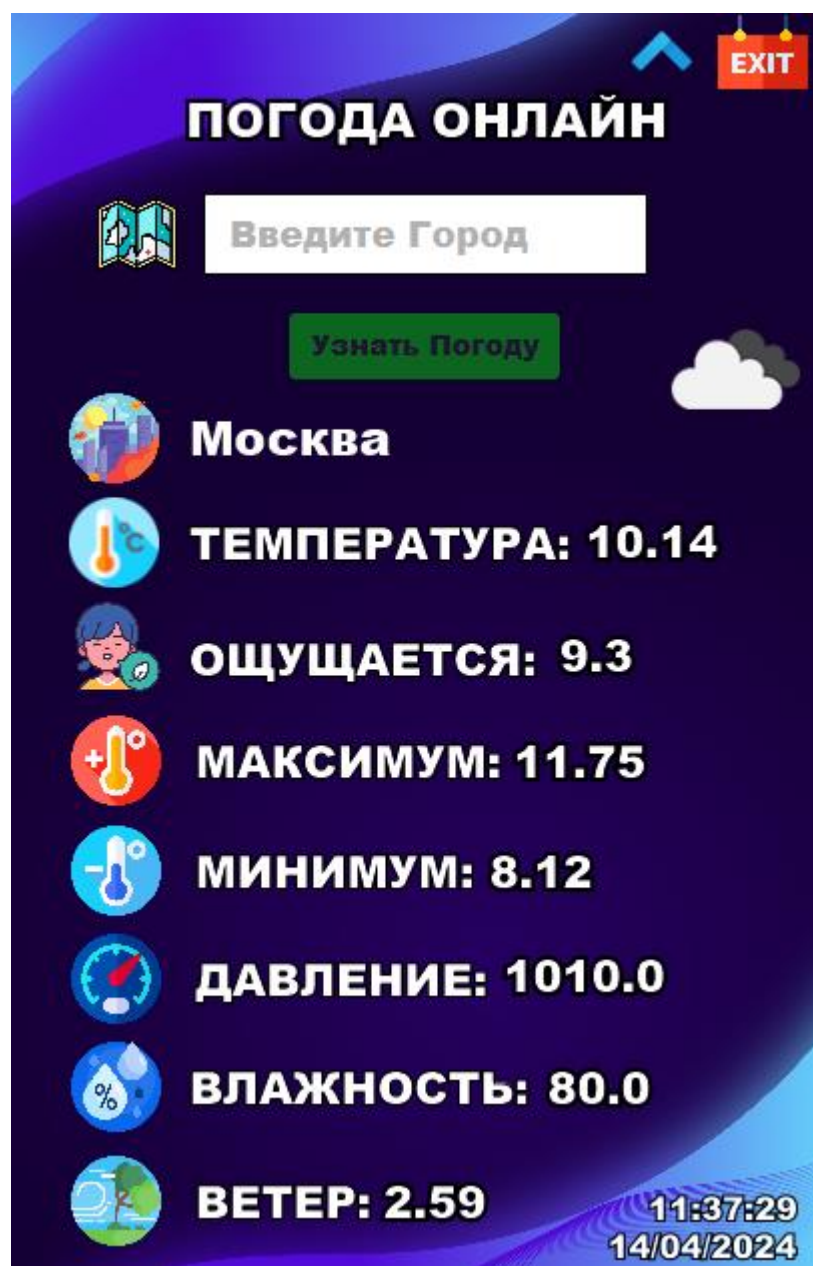


Рисунок 6 – пример работы программы⁵

Проверка работы исключения если пользователь ввел неверное название города или несуществующий город.

⁵ Составлено автором

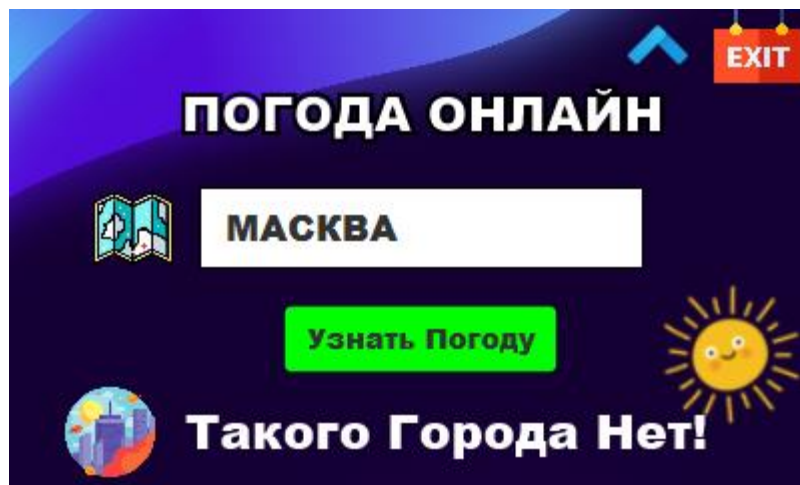


Рисунок 7 – пример работы исключения ⁶

Отдельный функционал программы в виде вывода даты, времени, кнопок выхода и сворачивания, а как же функционал перемещения программы полностью соответствует требованиям и работает исправно!

По результатам теста следует понять, что программа работает исправно.

Производительность и эффективность приложения:

Для начала работы с приложением пользователю понадобилось выполнить четыре шага: запустить приложение, нажать панель ввода текста, ввести название города, нажать на кнопку узнать погоду. Для этого новому пользователю понадобилось около двадцати секунд. Так же пользователь может обойтись и тремя шагами не вводя название города, а просто щелкнув на картинку справа от поля ввода.

⁶ Составлено автором

2.4. Руководство пользователя

Погода онлайн – программа для мониторинга погоды. С её помощью пользователь может узнать основную информацию о погоде в городах всего мира.

Инструкция по использованию:

Для того чтобы узнать информацию о погоде пользователю достаточно ввести название любого интересующего его города на русском или английском языке в поле для ввода, пример:

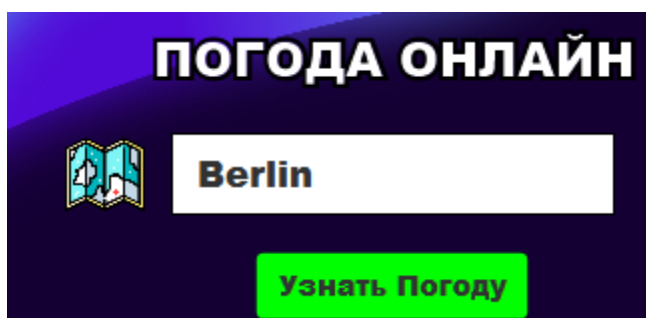


Рисунок 7 – пример ввода города⁷

После ввода города, пользователю необходимо нажать на кнопку, если город указан верно, то данные отобразятся, пример:

⁷ Составлено автором



Рисунок 8 – пример отображения данных⁸

Для передвижения приложения необходимо потянуть за верхний участок. Чтобы свернуть программу, нужно нажать на кнопку слева от выхода. Чтобы выйти из программы необходимо нажать на соответствующую кнопку «exit».

⁸ Составлено автором

ЗАКЛЮЧЕНИЕ

На данный момент, библиотека JavaFX выделяется своей удобностью и богатым функционалом по сравнению с другими аналогами. В целом, JavaFX является мощным инструментом для создания современных и отзывчивых Java приложений, что подтверждается его активным развитием и обновлениями, а также широким применением в разработке GUI приложений что подтверждается заявлениями разработчиков о её перспективности. В процессе достижения поставленной цели были выполнены следующие задачи:

Были проведены исследования существующих программных решений, в результате которых было установлено, что JavaFX обладает значительными преимуществами перед альтернативами, включая обширный набор компонентов для графического интерфейса, совместимость с CSS и возможность использования FXML для создания GUI.

Было разработано техническое задание, разработанное приложение полностью соответствует этим требованиям. В нём реализованы все необходимые функциональные возможности.

Нефункциональные требования также были выполнены, программа разработана с использованием JavaFX, FXML, CSS, JSON что подтверждает успешное выполнение поставленных задач и достижение цели.

В будущем программу можно доработать, добавить больше функций и возможностей.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

Книги, статьи

1. Хорстманн, Кей С. Биг Java: ранние объекты. — Москва: Питер, 2020. — 1200 с.
2. Жеребцов, А. С. Объектно-ориентированный анализ и программирование [Текст] : учебное пособие / А. С. Жеребцов, С. Ф. Молодецкая ; М-во образования и науки Рос. Федерации, Урал. гос. экон. ун-т. - Екатеринбург : [Издательство УрГЭУ], 2014. - 126 с. hЕккель, Брюс. Философия Java. — Санкт-Петербург: Питер, 2019. — 1152 с. Гонсалвес, Энтони Изучаем Java EE 7 / Энтони Гонсалвес. - М.: Питер, 2016. - 640 с.
3. Селдер, Рауль-Урма, Фурньо, Марио, Алан Майер. Java 8 и 9 в действии. — Москва: Манн, Иванов и Фербер, 2020. — 784 с.
4. Орельский, Владимир. Изучаем Java с нуля. — Москва: БХВ-Петербург, 2020. — 512 с.
5. Шибайкин С.Д. Цифровая обработка изображений Java и OpenCV [Текст]руководство для начинающих / Шибайкин С.Д. , 2019. - 456 с.
6. Седжвик, Р. Алгоритмы на Java [Текст] : научное издание / Роберт Седжвик, Кевин Уэйн ; [пер. с англ. А. А. Моргунова; под ред. Ю. Н. Артеменко]. - 4-е изд. - Москва : Вильямс, 2017. - 843 с
7. Прохоренок Н.А. JavaFX [Текст] : руководство для начинающи Прохоренок Н.А. , 2020. - 687 с.

Иностранная литература:

8. Mastering JavaFX 10 Build Advanced and Visually Stunning Java Applications
Sergey Grinev [Text]. - 2018. 218 p.

9. Ralph Lecessi Functional Interfaces in Java: Fundamentals and Examples [Text].
- 2019. 445 p.

10. Josh Juneau Java EE 8 Recipes: A Problem-Solution Approach, Second
Edition[Text]. - 2018. 298 p.

Интернет-источники:

11. Разработка графического пользовательского интерфейса (GUI) -
Режим доступа: <https://xakep.ru/2014/09/10/java-gui/>

12. Основных фреймворка для создания графических пользовательских
интерфейсов (GUI): [Электронный ресурс] - Режим доступа:
<https://www.geeksforgeeks.org/java-awt-vs-java-swing-vs-java-fx/>

13. Полезный ресурс для изучения Java : [Электронный ресурс] - Режим
доступа: <https://habr.com/ru/articles/554274/>

14. Создания диаграмм: [Электронный ресурс] - Режим доступа:
<https://app.diagrams.net/>

15. Ресурс с иконками Freepik: [Электронный ресурс] - Режим доступа:
https://ru.freepik.com/icon/weather-news_648198

16. Официальный сайт JavaFX: [Электронный ресурс] - Режим доступа:
<https://openjfx.io/>

17. Обучающие материалы по Java: [Электронный ресурс] - Режим
доступа: <https://vertex-academy.com/tutorials/ru/list-java-primer/>

18. Подборкой полезных практических и обучающих материалов по Java Tproger : [Электронный ресурс] - Режим доступа: <https://tproger.ru/digest/java>

19. Официальная документация по Java от Oracle: [Электронный ресурс] - Режим доступа: <https://docs.oracle.com/en/java/>

20. Платформа для хранения и совместной работы над проектами на GitHub: [Электронный ресурс] - Режим доступа: <https://github.com/search?q=language%3Ajava&type=repositories>

ПРИЛОЖЕНИЕ А

Листинг А.1. класса HelloApplication

```
package com.example.weather;
import javafx.animation.PauseTransition;
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.Priority;
import javafx.scene.layout.StackPane;
import javafx.scene.layout.VBox;
import javafx.stage.Modality;
import javafx.stage.Stage;
import javafx.stage.StageStyle;
import javafx.util.Duration;
import java.io.IOException;
import java.net.URISyntaxException;
public class HelloApplication extends Application {
    static final int width = 500;
    static final int height = 500;
    @Override
    public void start(Stage stage) throws IOException,
    URISyntaxException {
        ShowModal(stage); // Экран Загрузки

        Image image = new
Image(getClass().getResource("/com/example/weather/iconApp.png").to
ExternalForm());
        stage.getIcons().add(image);
        FXMLLoader fxmlLoader = new
FXMLLoader(HelloApplication.class.getResource("hello-view.fxml"));
// XML-структура, определяет элементы пользовательского интерфейса
спроектированного в SceneBuilder
        Scene scene = new Scene(fxmlLoader.load(), 410, 638);
        stage.setResizable(false);
        stage.initStyle(StageStyle.UNDECORATED);
        stage.setScene(scene);
```

```

        stage.show();
    }
    public void ShowModal(Stage stage) throws URISyntaxException {
//модуль загрузочного экрана
        Stage modalStage = new Stage();
        modalStage.initModality(Modality.WINDOW_MODAL);
        modalStage.initOwner(stage);
        modalStage.initStyle(StageStyle.UNDECORATED);
        StackPane stackPane = new StackPane();
        stackPane.setAlignment(Pos.CENTER);
        ImageView backView = new ImageView();
        Image backImage = new
Image(getClass().getResource("/com/example/weather/Loading.jpg").to
ExternalForm()); //Картинка Загрузочного Экрана
        backView.setImage(backImage);
        backView.setFitHeight(height);
        backView.setFitWidth(widht);
        stackPane.getChildren().add(backView);
        VBox vbox = new VBox();
        vbox.setAlignment(Pos.CENTER);
        vbox.setPadding(new Insets(150,150,150,150));
//Расположение гифки
        VBox innerVBox = new VBox();
        VBox.setVgrow(innerVBox, Priority.ALWAYS);
        vbox.getChildren().add(innerVBox);
        ImageView frontView = new ImageView();
        Image frontImage = new
Image(getClass().getResource("/com/example/weather/gif.gif").toExte
rnalForm()); //Гифка Загрузочного Экрана
        frontView.setImage(frontImage);
        frontView.setFitHeight((double) height / 8); //Размер гифки
        frontView.setFitWidth((double) widht / 8);
        VBox.setVgrow(frontView, Priority.ALWAYS);
        vbox.getChildren().add(frontView);
        stackPane.getChildren().add(vbox);
        Scene modalScene = new Scene(stackPane, widht, height);
        PauseTransition timer = new
PauseTransition(Duration.seconds(3)); //Таймер для загрузочного
экрана
        timer.setOnFinished( event -> {
            modalStage.close();
        });
    }

```

```
        timer.playFromStart();  
        modalStage.setScene(modalScene);  
        modalStage.showAndWait();  
    }  
  
    public static void main(String[] args) {  
        launch();  
    }  
}
```

ПРИЛОЖЕНИЕ Б

Листинг А.1. класса HelloController

```
package com.example.weather;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.URL;
import java.net.URLConnection;
import java.text.SimpleDateFormat;
import java.util.Arrays;
import java.util.Date;
import java.util.List;
import java.util.ResourceBundle;
import javafx.fxml.FXML;
import javafx.scene.Node;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.Pane;
import javafx.scene.text.Text;
import javafx.stage.Stage;
import org.json.JSONObject;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import javafx.animation.KeyFrame;
import javafx.animation.Timeline;
import javafx.util.Duration;

public class HelloController { // здесь содержаться переменные
    // которые ссылаются на объекты интерфейса (дизайна)
    @FXML
    private ResourceBundle resources;
    @FXML
    private URL location;
    @FXML
    private TextField city;
    @FXML
    private Button getData;
    @FXML
```

```

private Text pressure;
@FXML
private Text temp_feels;
@FXML
private Text temp_info;
@FXML
private Text temp_max;
@FXML
private Text temp_min;
@FXML
private Text DateText;
@FXML
private Text LabelTime;
@FXML
private ImageView changeImage;
@FXML
private Text humidity;
@FXML
private Text speed;
@FXML
private Text gorod;
@FXML
private ImageView exit;
@FXML
private Pane appBar;
@FXML
private ImageView collapse;
@FXML
private ImageView CityList;
private double xOffset = 0;
private double yOffset = 0;
private void InitCloseButton() {
    exit.setOnMouseClicked(event -> {Exit(event);});
}
private void CollapseButton() {
    collapse.setOnMouseClicked(event -> {
        Collapse(event);
    });
}
@FXML
void initialize() { //методы
    InitAppBar(); //возможность передвигать приложение

```

```

DynamicTime(); // Вывод Динамического Времени
InitCloseButton(); //кнопка (картинка) выхода
Time(); //вывод даты
chooseCity(); //возможность выбора города
CollapseButton(); // возможность свернуть приложение
getData.setDisable(true);
getData.setAction(event -> {
    String getUserCity = city.getText().trim();
    gorod.setText(getUserCity);
    if(!getUserCity.equals("")) { // Получение Данных с
сайта
        String output =
getUrlContent("https://api.openweathermap.org/data/2.5/weather?q="
+ getUserCity +
"&appid=80442090e9bad59438b86babdd86afd8&units=metric");
        if (!output.isEmpty()) {
            JSONObject obj = new JSONObject(output); //
преобразование данных JSON формата
            double receivedTemp =
obj.getJSONObject("main").getDouble("temp"); // вывод необходимых
данных (температура)
            temp_info.setText("" + receivedTemp);
            temp_feels.setText("" +
obj.getJSONObject("main").getDouble("feels_like")); // Как ощущается
температура
            temp_max.setText("" +
obj.getJSONObject("main").getDouble("temp_max")); // Максимальная
температура
            temp_min.setText("" +
obj.getJSONObject("main").getDouble("temp_min")); // Минимальная
температура
            pressure.setText("" +
obj.getJSONObject("main").getDouble("pressure")); // Давление
            humidity.setText("" +
obj.getJSONObject("main").getDouble("humidity")); // Влажность
            speed.setText("" +
obj.getJSONObject("wind").getDouble("speed")); // Скорость ветра
            GetCurrentWeathrIcon(obj); //погодная иконка
(дождь, снег, солнце, туман)
            city.clear();
        }
    }
}

```



```

    });
    city.textProperty().addListener((observable, oldValue,
newValue) -> {
        getData.setDisable(newValue.isEmpty());
    });
}

private void GetCurrentWeathrIcon(JSONObject obj) { //
получение иконки с погодой
    String icon =
obj.getJSONArray("weather").getJSONObject(0).getString("icon");
//пременная в которой хранится название иконки
    String iconURL = "https://openweathermap.org/img/wn/" +
icon + "@2x.png"; //пременная в которой хранится иконка, которая
была выбрана исходя из названия в icon
    Image image = new Image(iconURL); //помещаем иконку в image
    changeImage.setImage(image); //помещаем иконку в
changeImage созданную в SceneBuilder
}

private void InitAppBar() { //возможность передвигать
приложение
    appBar.setOnMousePressed(mouseEvent -> {
        xOffset = mouseEvent.getSceneX();
        yOffset = mouseEvent.getSceneY();
    });
    appBar.setOnMouseDragged(mouseEvent -> {
        Node node = (Node) mouseEvent.getSource();
        Stage thisStage = (Stage) node.getScene().getWindow();
        thisStage.setX(mouseEvent.getScreenX() - xOffset);
        thisStage.setY(mouseEvent.getScreenY() - yOffset);
    });
}

public void Exit (MouseEvent event) //кнопка (картинка) выхода
{
    Node node = (Node) event.getSource();
    Stage thisStage = (Stage) node.getScene().getWindow();
    thisStage.close();
}

public void Collapse (MouseEvent event) //сворачивание
приложения
{
    Node node = (Node) event.getSource();
    Stage thisStage = (Stage) node.getScene().getWindow();

```

```

        thisStage.setIconified(true);
    }
    private List<String> texts = Arrays.asList("Москва",
        "Норильск", "Екатеринбург", "Berlin", "Paris", "Tokyo", "London",
        "New York", "Sydney", "Rome", "Toronto", "Melbourne", "Dubai",
        "Hong Kong", "Toronto", "Sao Paulo", "Cairo", "Istanbul",
        "Tokyo");
    private int currentIndex = 0;
    public void chooseCity () //при нажатии на картинку появляется
название города
    {
        CityList.setOnMouseClicked(event -> { currentIndex =
        (currentIndex + 1) % texts.size(); // Циклическое изменение индекса
        city.setText(texts.get(currentIndex));
        });
    }
    public void Time () // Вывод даты
    {
        TextField textField = new TextField();
        SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
        Date currentDate = new Date();
        String formattedDate = sdf.format(currentDate);
        textField.setText(formattedDate);
        DateText.setText(formattedDate);
    }
    public void DynamicTime() { // Вывод Динамического Времени
        updateTimeLabel();
        Timeline timeline = new Timeline(new
        KeyFrame(Duration.seconds(1), event -> updateTimeLabel()));
        timeline.setCycleCount(Timeline.INDEFINITE);
        timeline.play();
    }
    public void updateTimeLabel() {
        LocalTime currentTime = LocalTime.now();
        DateTimeFormatter formatter =
        DateTimeFormatter.ofPattern("HH:mm:ss");
        LabelTime.setText(currentTime.format(formatter));
    }
    public String getUrlContent(String urlAdress){ // Обращение по
URL - позволяет получить содержимое веб-страницы по заданному URL и
вернуть его в виде строки
        StringBuffer content = new StringBuffer();

```

```


        try { // Обработка исключений используется для обработки
все возможных исключений
            URL url = new URL(urlAdress);
            URLConnection urlConn = url.openConnection(); //
Открытие соединения
            BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(urlConn.getInputStream())); // Чтение данных
            String line;
            while ((line = bufferedReader.readLine()) != null) //
считывание строк
            {
                content.append(line + "\n"); // Каждая считанная
строка добавляется в StringBuffer content с добавлением символа
новой строки
            }
            bufferedReader.close(); // Закрытие потока
        } catch (Exception e)
        {
            gorod.setText("Такого Города Нет!"); // Конструкция для
проверки на корректный ввод города
        }
        return content.toString(); // возвращается строковое
представление содержимого StringBuffer content, полученного из URL
    }
}

```

ПРИЛОЖЕНИЕ В

Ссылка на репозиторий в GitHub: <https://github.com/NoPlay2024/java-coursework.git>

ПРИЛОЖЕНИЕ Г

**АНТИПЛАГИАТ**
ОБНАРУЖЕНИЕ ЗАИМСТВОВАНИЙ

ТАРИФ
Demo ⚠
ИЗМЕНИТЬ

ПРОВЕРКИ
1 в 6 минут ⌚
ПРОВЕРИТЬ ДОКУМЕНТ

ПОЛЬЗОВАТЕЛЬ
porlay2024@yandex.ru
ВОЙТИ В КАБИНЕТ

МЕНЮ

ru ▾

главная / кабинет / результаты проверки

Оригинальность97,13%

Совпадения2,87%

Цитирование0%

Самоцитирование0%

ПОЛНЫЙ ОТЧЕТ

КРАТКИЙ ОТЧЕТ

ИСТОРИЯ ОТЧЕТОВ

РАСПЕЧАТАТЬ ▾

ВЫГРУЗИТЬ ▾

СОЗДАТЬ ССЫЛКУ ▾

Свойства документа

Структура документа

Текстовые метрики

Параметры проверки

Статистика по документу

Авторы документа ?

Разбойников

Сергей Андреевич

Имя исходного файла

Курсовая-Работа-po-Java.pdf

Название документа

Курсовая-Работа-po-Java

Тип документа

Не указано

РЕДАКТИРОВАТЬ СВОЙСТВА

главнаяистория обновленийпомощьвебинарыконтакты

Сайт для корпоративных клиентовПользовательское соглашениеСоглашение об обработке персональных данных

АО "Антиплагиат" 2005-2024 © Все права защищены

YouTube

VK

13:42

14.04.2024