

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Рязанский государственный радиотехнический университет
имени В.Ф. Уткина»
Кафедра САПР ВС

К защите
Руководитель работы:

дата, подпись

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОЙ РАБОТЕ**

по дисциплине
«Компьютерная графика»
Тема:
«Использование библиотеки JavaFX для разработки графических
приложений»

Выполнил студент группы 235
Гусев К.Е.

дата сдачи на проверку, подпись

Руководитель работы
доцент каф. САПР ВС
Митрошин А.А.

оценка

дата защиты, подпись

Рязань 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования

Рязанский государственный радиотехнический университет
имени В. Ф. Уткина

Кафедра САПР ВС

З А Д А Н И Е

на курсовую работу по дисциплине
"Компьютерная графика"

Студент — _____ Гусев Константин Евгеньевич _____ группа – 235

1. Тема: Использование библиотеки JavaFX для разработки графических приложений

2. Срок представления работы к защите - «__» _____ 2024г.

3. Исходные данные для разработки:

3.1. Языки программирования – Java, JavaScript.

3.2. Технология для создания графического интерфейса — JavaFX.

3.3. Формат хранения данных – JSON.

4. Содержание курсовой работы

Введение.

4.1. Описание библиотек.

4.1.1. JavaFX.

4.1.2. Chromium Embedded Framework.

4.1.3. JSON-Java (org.json).

4.2. Постановка задачи.

4.3. Подключение библиотек к проекту.

4.4. Разработка ПО.

4.4.1. Браузер.

4.4.2. Интерфейс.

4.4.3. Сохранение проектов.

4.5. Тестирование.

Заключение.

Список использованных источников.

Руководитель работы _____ / Митрошин А.А.

(подпись)

Задание принял к исполнению _____ «__» _____ 2024г.

(подпись)

Содержание

ВВЕДЕНИЕ.....	4
1 Описание библиотек	5
1.1 JavaFX.....	5
1.2 Chromium Embedded Framework.....	6
1.3 JSON-Java (org.json)	6
2 Постановка задачи.....	7
3 Подключение библиотек к проекту.....	8
4 Разработка ПО	11
4.1 Браузер	12
4.2 Интерфейс	15
4.3 Сохранение проектов.....	21
5 Тестирование приложения	25
Заключение	33
Список использованных источников	34

ВВЕДЕНИЕ

Современные цифровые технологии значительно облегчают работу в различных сферах жизни, заменяя устаревшие и неудобные решения. Одной из таких областей является создание и редактирование сайтов, где традиционные методы разработки уступают место более эффективным и динамичным инструментам. Веб-разработка сегодня активно использует мощные средства для создания интерактивных и адаптивных интерфейсов, что позволяет значительно повысить удобство и скорость работы с информацией. Одним из таких инструментов является разработка конструкторов сайтов, которые обеспечивают возможность быстрого и интуитивно понятного создания веб-страниц без необходимости углубленного знания программирования.

Данная курсовая работа посвящена разработке конструктора сайтов с использованием технологий JavaFX и Java CEF (Chromium Embedded Framework). Использование JavaFX в сочетании с CEF позволяет создать мощное и удобное графическое приложение для создания и редактирования сайтов, с богатым набором возможностей для визуализации, а также с высокой гибкостью в настройке интерфейса. Этот инструмент будет полезен как профессиональным веб-разработчикам, так и пользователям, не обладающим глубокими знаниями в области программирования, но нуждающимся в создании качественных и функциональных сайтов.

1 Описание библиотек

Java Development Kit (JDK) предоставляет обширный набор библиотек для решения широкого круга задач, таких как работа с базами данных, сетевые коммуникации и создание графических интерфейсов. Однако для более специфичных задач, например, интеграции веб-контента или работы с мультимедийными данными, стандартных возможностей JDK часто недостаточно. В таких случаях для расширения функциональности необходимо подключать дополнительные библиотеки.

1.1 JavaFX

JavaFX[1] — это библиотека с открытым исходным кодом, предназначенная для создания графических приложений с использованием языка программирования Java. Она предоставляет широкий набор инструментов для разработки настольных, мобильных и встраиваемых приложений.

Среди преимуществ JavaFX можно выделить:

1. Графический движок Prism, который поддерживает аппаратное ускорение графики.
2. Мультимедийный движок, поддерживающий современные форматы видео и аудио, такие как H.264 для видео и AAC для аудио.
3. Интеграция с другими библиотеками Java, такими как Swing, что дает возможность использовать JavaFX в существующих приложениях.
4. FXML — язык разметки, основанный на XML, который позволяет разделять логику приложения и проектирование интерфейса.
5. Поддержка таблиц стилей CSS, что упрощает настройку внешнего вида и стилизацию элементов интерфейса.
6. Поддержка мультитач-сенсорных экранов, что делает JavaFX идеальным инструментом для создания приложений для мобильных устройств и сенсорных панелей.

7. Кроссплатформенность, что позволяет запускать JavaFX-приложений на различных операционных системах без изменений в коде.

1.2 Chromium Embedded Framework

Chromium Embedded Framework (CEF)[2] - открытая библиотека, предоставляющая механизм для встраивания веб-браузера на основе Chromium в десктопные приложения. CEF позволяет интегрировать современные веб-технологии, такие как HTML5, CSS, JavaScript, прямо в Java-приложения, используя возможности браузера Chromium.

Преимущества использования CEF:

1. Поддержка новейших веб-стандартов, включая HTML5, CSS3, WebGL и JavaScript.
2. Высокая производительность благодаря использованию движка Chromium.
3. Возможность взаимодействия между Java-приложением и веб-контентом через API.
4. Легкость встраивания веб-интерфейсов и расширений без необходимости разрабатывать собственные браузерные компоненты.
5. Кроссплатформенность, что позволяет использовать CEF на различных операционных системах.

1.3 JSON-Java (org.json)

JSON (JavaScript Object Notation)[3] — это легковесный формат обмена данными, основанный на текстовой строке и использующий синтаксис, аналогичный JavaScript-объектам. JSON используется для передачи данных между Java и встроенным веб-браузером, а также для хранения настроек и конфигурации веб-приложений.

2 Постановка задачи

Цель курсовой работы – разработать графическое приложение, позволяющее создавать и редактировать веб-страницы.

Задачи для выполнения цели:

- импорт HTML документа в качестве шаблона для создания веб-страниц;
- добавление элементов на страницу, таких как текстовые блоки, изображения, кнопки, формы;
- редактирование элементов на странице: изменение их размеров, положения, стилей (цвета, шрифты, отступы) и замена контента (например, текста и изображений);
- удаление элементов с веб-страницы;
- сохранение проекта в редактируемом формате и возможность открытия ранее созданных страниц для дальнейшего редактирования;
- экспорт страницы в стандартный HTML-код с возможностью сохранения в файлы формата .html.

3 Подключение библиотек к проекту

Для установки JavaFX нужно скачать SDK[4], затем в среде разработки Eclipse установить плагин, предназначенный для работы с данной библиотекой – e(fx)clipse [5]. Для этого в Eclipse нужно перейти по пунктам меню Help – Install New Software – Add и в поле Add Repository указать «<https://download.eclipse.org/efxclipse/updates-nightly/site/>» (рисунок 1).

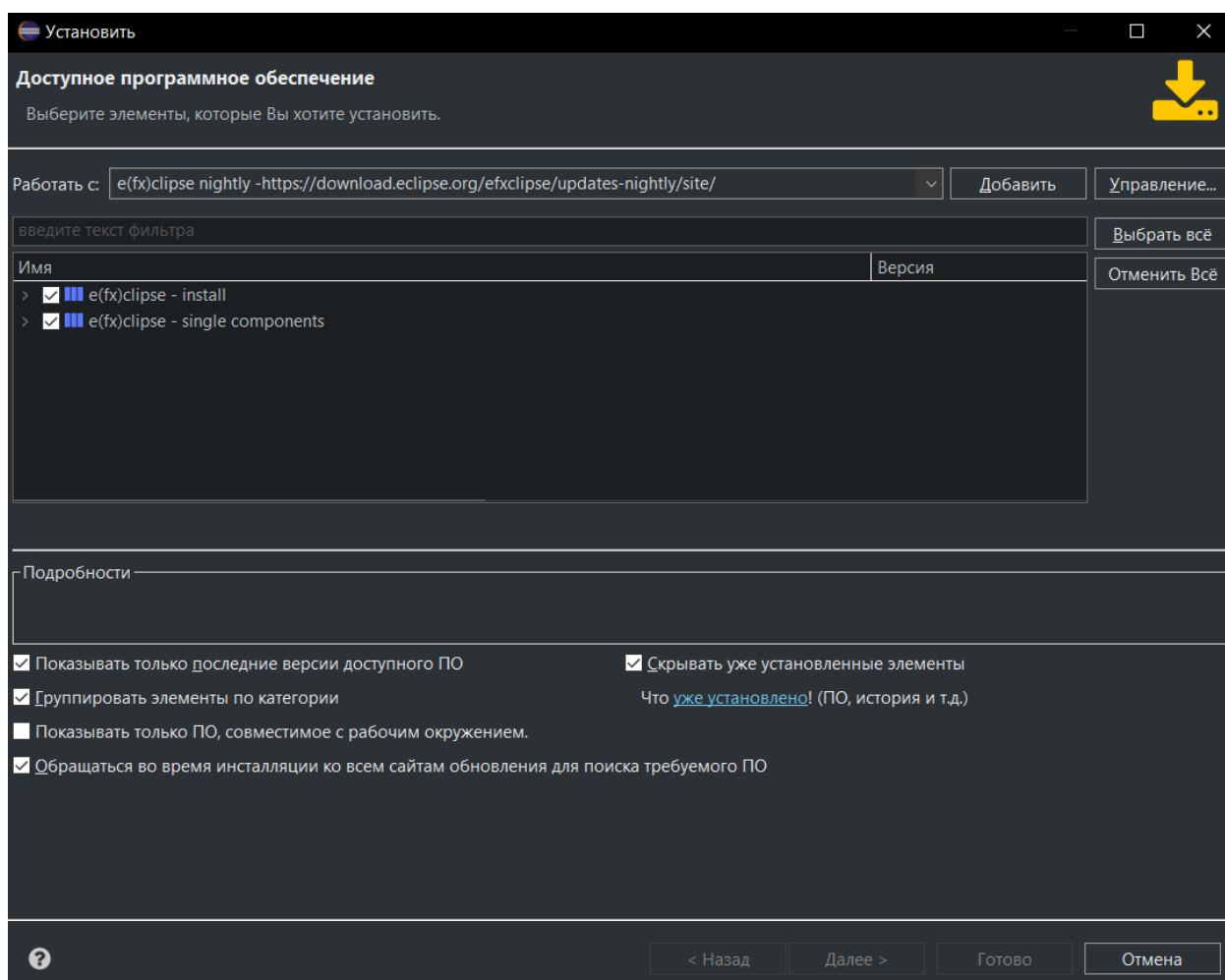


Рисунок 1 – Установка e(fx)clipse

После установки плагина и перезагрузки среды разработки в нужно перейти по пунктам меню Windows – Preferences – JavaFX. Где нужно указать директорию с jar файлами скачанного SDK (рисунок 2).

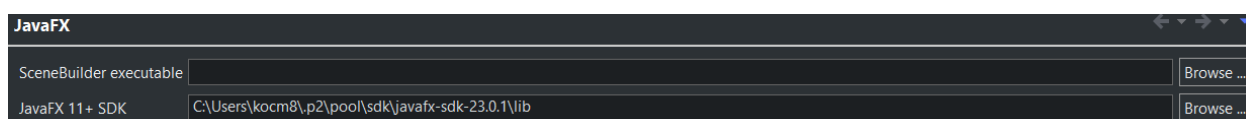


Рисунок 2 – Настройка плагина

v1.0.10-92.0.25+gd15cfa8+chromium-92.0.4515.131

Changes summary:

- [chromiumembedded/java-cef@ chromiumembedded:e08c67e...chromiumembedded:926b08f](#)

▼ Assets 6

win32.zip	89.1 MB	Aug 18, 2021
win32.zip.sha256	64 Bytes	Aug 18, 2021
win64.zip	94.7 MB	Aug 18, 2021
win64.zip.sha256	64 Bytes	Aug 18, 2021
Source code (zip)		Sep 3, 2020
Source code (tar.gz)		Sep 3, 2020

Рисунок 4 – Скачивание CEF

После распаковки нужно указать в системную переменную среды директорию, в данном случае win64, с библиотеками dll (рисунок 5).

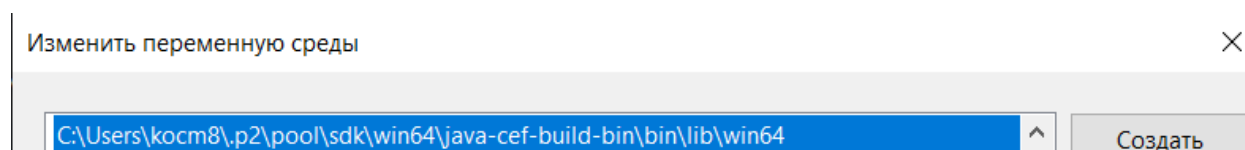


Рисунок 5 – Изменение переменной среды

Затем добавить jar файлы в проект.

Для установки org.json нужно скачать с репозитория maven [7] jar файл и подключить его к проекту.

4 Разработка ПО

Структура Java проекта показана на рисунке 6.

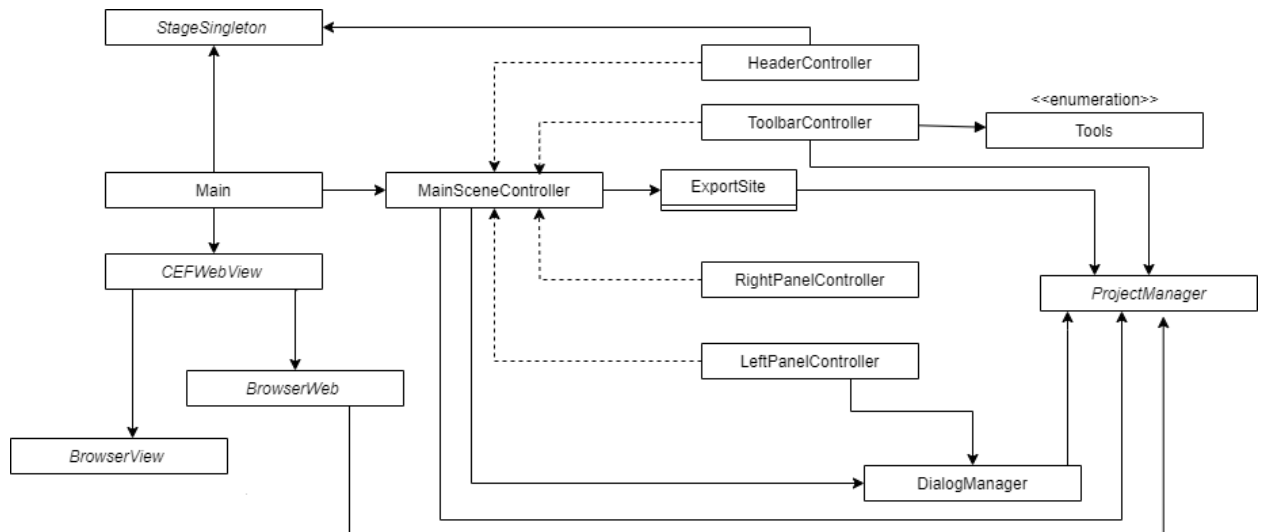


Рисунок 6 – Структура проекта

Класс Main является точкой входа в приложение.

Таблица 1 – Описание класса Main

Поле или заголовок метода	Пояснение
<code>public static void main(String[] args)</code>	Вызывает метод <code>launch()</code> , который инициализирует запуск JavaFX-приложения.
<code>@Override</code> <code>public void start(Stage stage)</code>	Настраивает окно JavaFX-приложения, подключает <code>mainScene.fxml</code> , подключает <code>MainSceneController</code> к <code>fxml</code> , запускает браузер, а также передает ссылку на <code>Stage</code> .

4.1 Браузер

CEFWebView отвечает за работу встроенного браузера, использует методы классов BrowserWeb для настройки браузера и BrowserView для его отображения.

Таблица 2 – Описание класса CEFWebView

Поле или заголовок метода	Пояснение
<code>private static Stage _stage;</code>	Окно приложения.
<code>private static JFrame frame;</code>	Окно браузера.
<code>private static CefBrowser browser;</code>	браузер CEF, который предоставляет API для работы с веб-контентом.
<code>private static String URL = "templates/newproject/index.html";</code>	Ссылка на загружаемый html файл при запуске браузера.
<code>public static void start(Stage stage)</code>	Запускает браузер, получая окно приложения и вызывая методы Web(), View().
<code>public static void startOver()</code>	Перезапускает браузер, не вызывая Web().
<code>private static void Web()</code>	Создает и настраивает браузер, помещает его в поле browser.
<code>private static void View()</code>	Создает и настраивает окно браузера, помещает в него browser.
<code>public static void Reload()</code>	Перезагружает страницу браузера.
<code>public static void ExecuteJS(String filePath)</code>	Выполняет JavaScript код из файла в браузере.
<code>public static void ExecuteJSOneString(String jsCode)</code>	Выполняет JavaScript код в браузере.

Продолжение таблицы 2

<code>public static void ChangeElemets(String id, String json)</code>	Применяет изменения к web-элементу по id .
<code>public static void LoadElements(String json)</code>	Загружает элементы на веб-страницу.

BrowserWeb создает и настраивает встроенный в приложение браузер, также настраивает маршрутизацию запросов между Java и браузером.

Таблица 3 – Описание класса BrowserWeb

Поле или заголовок метода	Пояснение
<code>static final boolean OFFSCREEN = false</code>	Флаг, определяющий, будет ли браузер работать в оффскрин-режиме (без отображения на экране).
<code>static final boolean TRANSPARENT = false</code>	Флаг, который указывает, будет ли окно браузера прозрачным.
<code>static CefBrowser browser</code>	Объект браузера CefBrowser, который используется для рендеринга контента (веб-страниц).
<code>static CefClient client</code>	Клиент CEF (CefClient), который управляет браузером и обрабатывает взаимодействие с веб-страницей.
<code>static CefSettings settings</code>	Настройки для CEF. Содержит параметры, которые конфигурируют поведение браузера.
<code>static CefMessageRouter messageRouter</code>	Маршрутизатор сообщений CEF, который управляет обменом данными между JavaScript на веб-странице и Java-приложением.

Продолжение таблицы 3

<code>public static void createBrowser(String URL, Stage _stage)</code>	Основной метод для инициализации браузера, вызывает методы для настройки приведенные ниже.
<code>private static void setSetting()</code>	Настраивает параметры для CEF. Включает или отключает оффскрин-рендеринг, используя флаг OFFSCREEN.
<code>private static void initClient()</code>	Создает экземпляр клиента CEF с использованием настроек.
<code>private static void initMessageRouter()</code>	Инициализирует маршрутизатор сообщений для обработки запросов от JavaScript.
<code>public static void runBrowser(String URL)</code>	Запускает браузер с заданным URL.

Браузер CEF рассчитан на использование с библиотекой Swing, в окно JavaFX не удалось загрузить, поэтому `BrowserView` создает отдельное окно и настраивает и «соединяет» с окном, созданным JavaFX.

Таблица 4 – Описание класса `BrowserView`

Поле или заголовок метода	Пояснение
<code>private static JFrame frame</code>	Окно, которое будет отображать содержимое браузера.
<code>public static JFrame createBrowserFrame(Stage _stage)</code>	Основной метод для создания окна браузера, используя методы <code>createFrame</code> и <code>connectToStage</code> .
<code>private static void createFrame()</code>	Создает и настраивает окно, в котором будет браузер.
<code>private static void connectToStage(Stage _stage)</code>	Связывает окно браузера с JavaFX окном.

При запуске браузера загружается стартовая HTML страница, к которой подключены JavaScript файлы:

elementSelector.js – по клику на элемент выделить его рамкой и отправить информацию о нем (id, содержимое тега, стили) в Java приложение.


changeSize.js – изменять размер выбранного блока и передвигать его курсором.

createStartSection.js – создать секцию, в которой будет блок с классом container, в котором будут создавать элементы.

loaderFromJava.js – применяет измененные стили к элементу.

В styles.css устанавливается размер container и центрируется на странице.

Работа elementSelector и createStartSection представлена на рисунке 7.



Hello, I am a new h1!

Рисунок 7 – Выделение элемента, границы которого находятся в контейнере

Полученный веб-элемент из браузера:

```
{ "styles":  
  {  
    "outline": "rgb(0, 123, 255) solid 2px"  
  },  
  "id": "351cf815-be89-4d5e-ad21-b9950c57f23a",  
  "text": "Hello, I am a new h1!",  
  "tagName": "H1"  
}
```

4.2 Интерфейс

Main загружает mainScene.fxml, который включает в себя другие fxml файлы, отвечающие за основные области графического интерфейса (рисунок 8), к каждой подключается свой контроллер. Небольшие области: меню и строка состояния остались в этом файле и реализованы в MainSceneController.

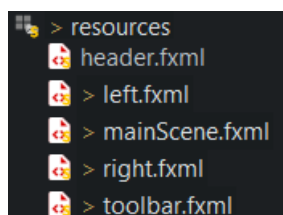


Рисунок 8 – Установка e(fx)clipse

Таблица 5 – Описание класса MainSceneController

Поле или заголовок метода	Пояснение
@FXML private HBox header;	Блок из header.fxml.
@FXML private HBox toolbar;	Блок из toolbar.fxml.
@FXML private VBox left;	Блок из left.fxml.
@FXML private VBox right;	Блок из right.fxml.
@FXML private Label projectName;	Текущий проект.
@FXML private Label projectSaveLabel;	Сохранен ли проект.
@FXML private Label curentPageLabel;	Текущая страница проекта.
private DialogManager dialogManager	Объект, для управления диалогами.
@FXML private void initialize()	Создает dialogManager, для сценария создания проекта.
@FXML private void create()	Создать проект.
@FXML private void save()	Сохранить проект.
@FXML private void load()	Загрузить проект.
@FXML private void export()	Экспортировать проект в HTML.
@FXML private void createNewPage()	Создать страницу проекта.
@FXML private void goToProjectPage()	Перейти на страницу проекта.
@FXML private void deletePage()	Удалить страницу проекта.
@FXML private void openDoc()	Открыть документацию приложения – эту пояснительную записку.
@FXML private void showAbout()	Открыть окно «О программе».

При запуске приложение откроется диалог, который предложит создать или выбрать проект (рисунки 9-10), в случае создание будет предложено назвать проект, в случае выбора проекта откроется окно выбора директории проектов.

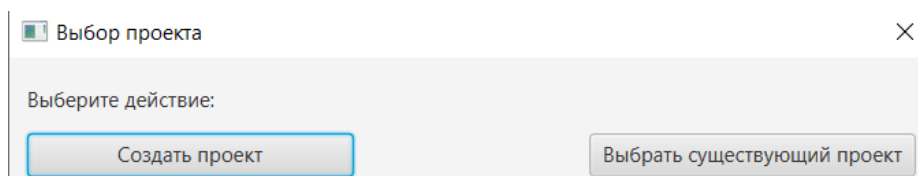


Рисунок 9 – Стартовый диалог

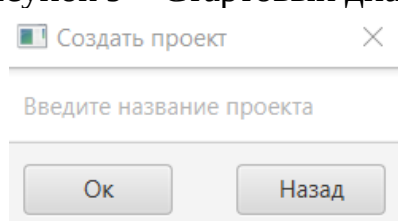


Рисунок 10 – Диалог создания проекта

Строка состояния отображает текущий проект, состояние сохранения и текущую страницу (рисунки 11-12).

Проект: | Состояние: | Страница:

Рисунок 11 – Строка состояния на момент запуска приложения

Проект: test | Состояние: Не сохранен | Страница: index

Рисунок 12 – Строка состояния после создания проекта

Класс `ToolbarController` реализует панель инструментов в виде иконок-кнопок. Нажатие кнопки, создающей элемент, вызывает выполнение JavaScript кода: создать элемент, назначить ему `id` (чтобы в будущем можно было выбрать этот элемент), добавить элемент в выбранный элемент-контейнер, если такой отсутствует, то добавить в `container`. Создание абзаца показано в листинге 1. Аналогичным образом создаются все элементы на веб-странице.

Листинг 1 - CreateP.js

```
(function() {  
    const id = crypto.randomUUID();  
    var newP = document.createElement('p');  
    newP.id = id;  
    newP.textContent = 'Hello, I am a new p!';  
    if (selectedElementId) {  
        const container = document.getElementById(selectedElementId);  
        container.appendChild(newP);  
    } else {  
        const container = document.querySelector('.container');  
        container.appendChild(newP);  
    }  
})();
```

Создание элементов:

 - создать блок;

 - создать текстовый элемент (абзац, заголовок h1/h2/h3) ;

 - создать блок с n-ым количеством колонок;

 - создать кнопку;






 - создать картинку;

 - создать форму;

 - создать поле ввода;

 - сгенерировать шапку сайта с n-ым количеством элементов.

Для работы с веб-страницей используются следующие инструменты:

-  - удалить выбранный элемент;
-  - загрузить изображение в директорию проекта;
-  - выполнить JavaScript код;
-  - очистить страницу;
-  - перезагрузить браузер;

При клике на веб-элемент данные, сформированные `elementSelector.js`, отправляются в панель редактирования элементов, контролируемая классом `RightPanelController`. В зависимости от тега элемента, показывается соответствующие поля для стилизации (рисунки 13-15).

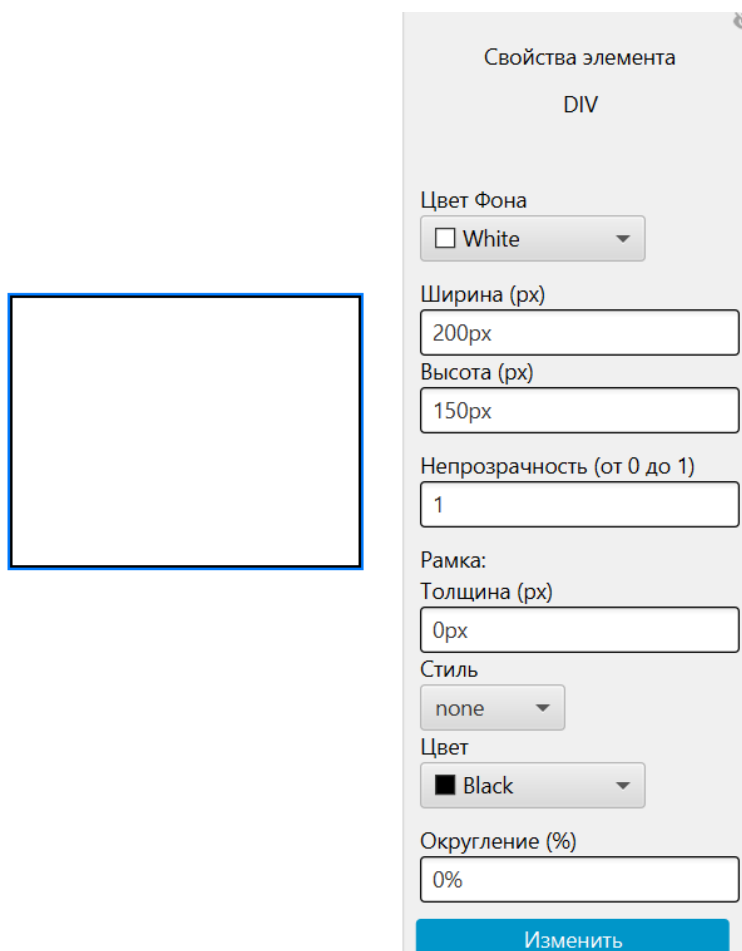


Рисунок 13 – Редактирование блочных элементов

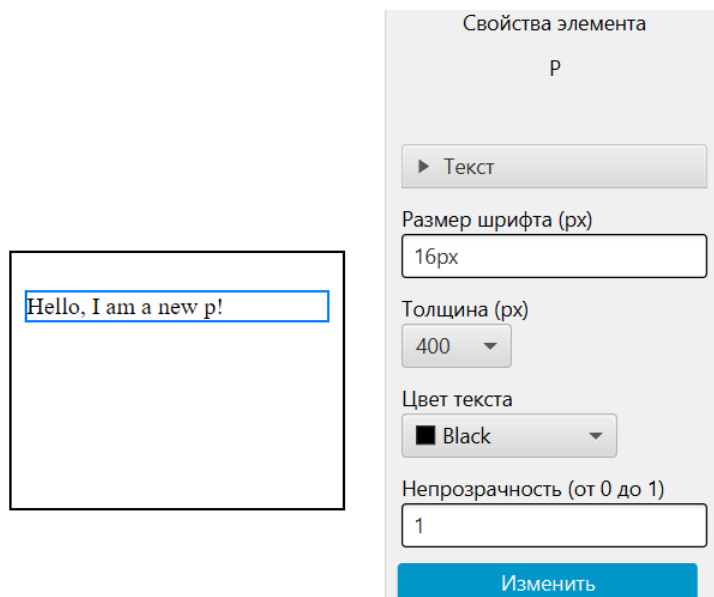


Рисунок 14 – Редактирование строчных элементов

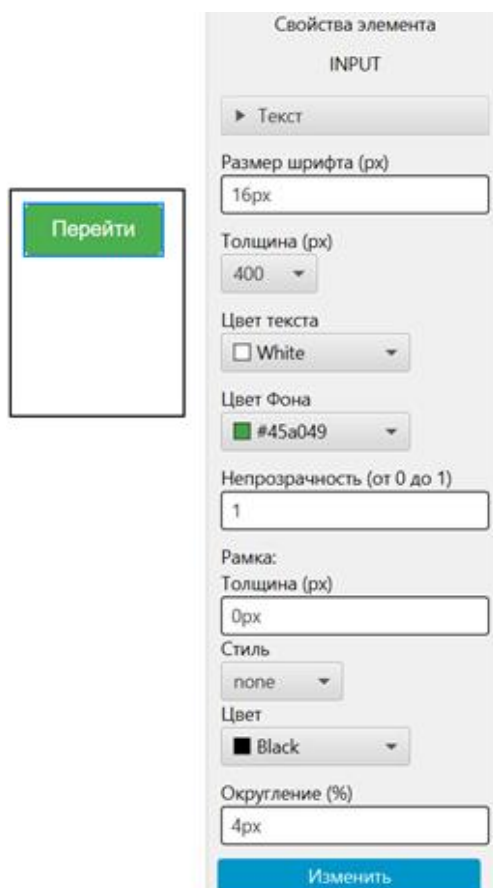


Рисунок 15 - Редактирование кнопочных элементов

При нажатии на кнопку «Изменить» применяются заданные свойства стилизации.

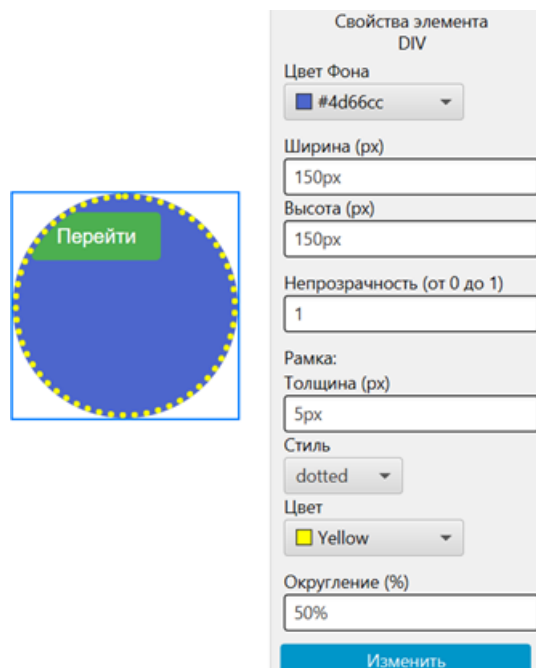


Рисунок 16 – Изменение элементов

При запуске боковые панели скрыты, что их увидеть нужно нажать соответствующие кнопки (рисунок 17).



Рисунок 17 – Иконки для открытия панелей редактирования и модулей

Панель использования модулей реализует LeftPanelController, она используется чтобы добавлять сложные элементы, использующие JavaScript код и специфичные css свойства (например анимации). Реализованные модули показаны на рисунках 18-21

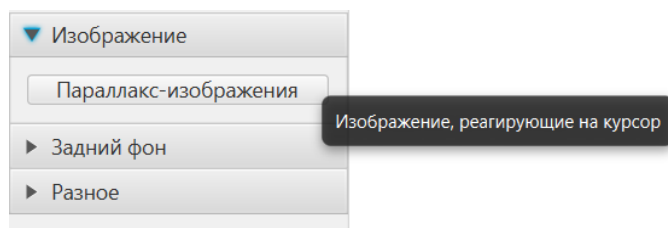


Рисунок 18 – Модуль «Параллакс-изображение»

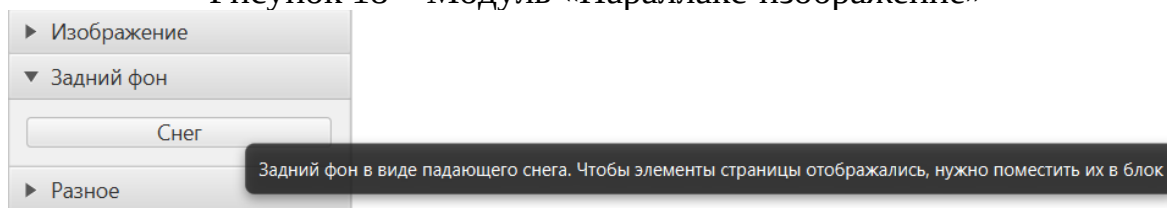


Рисунок 19 – Модуль «Снег»

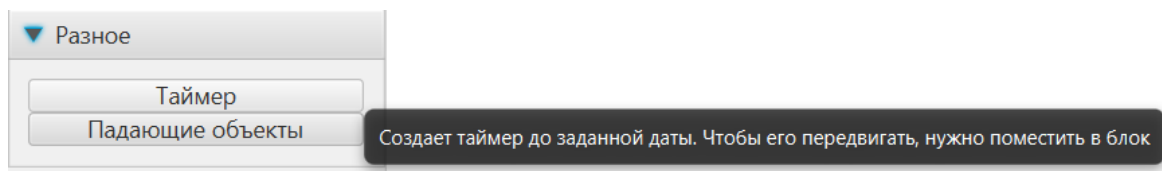


Рисунок 20 – Модуль «Таймер»

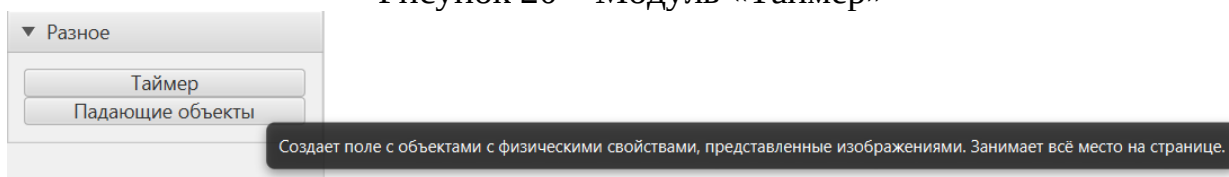


Рисунок 21 – Модуль «Падающие предметы»

Модули могут требовать задания параметров (рисунок 22).

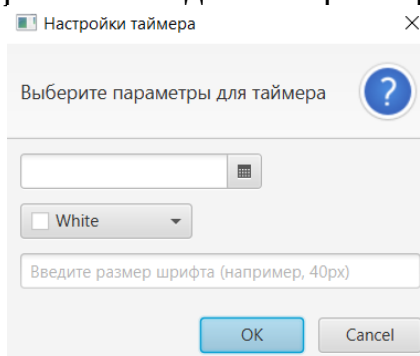


Рисунок 22 – Параметра для таймера

4.3 Сохранение проектов

Класс ProjectManager следит за директорией проекта, он реализует сохранение и загрузку проектов и их страниц.

Таблица 6 – Описание класса ProjectManager

Поле или заголовок метода	Пояснение
<code>static String projectName</code>	Название текущего проекта.
<code>static String jsonFile</code>	Путь к странице проекта.
<code>static String <u>directory</u></code>	Директория проекта.
<code>static String <u>directoryImg</u></code>	Директория с картинками в проекте.
<code>static boolean isSaved = false</code>	Был ли сохранен проект.
<code>public static void setProjectName (String projectName)</code>	Устанавливает название проекта, создает соответствующие директории и файл конфигурации JSON для проекта.
<code>public static void createJsonForNewProject ()</code>	Создает файлы для хранения страницы и конфигурации страниц (при создании проекта).

Продолжение таблицы 6

<code>public static boolean saveCurrentPage()</code>	Сохраняет текущую страницу проекта.
<code>public static void createNewPageJson(String pageName)</code>	Создает новый JSON файл страницы с именем <code>pageName</code> и обновляет конфигурацию страниц.
<code>public static void deletePageJson(String pageJsonFile)</code>	Удаляет страницу из конфигурации <code>pages.cfg</code> , если такая страница существует.
<code>public static boolean checkIfImgFolderIsEmpty()</code>	Пуста ли папка для изображений.
<code>public static void loadPagesConfig()</code>	Загружает <code>pages.cfg</code> , выполняя соответствующие скрипты для модулей страницы.

При создании проекта приложение требует ввести название проекта, которое будет названием директории проекта, а также создается стартовая страница проекта `index`, с которой будет загрузка проекта.

При создании страницы формируется файл, хранящий информацию о страницах проекта `pages.cfg` (листинг 2) и пустой `index.json`, который будет хранить все созданные элементы на странице.

Листинг 2 – `pages.cfg` при создании проекта

```
{
  "pages": [
    {
      "title": "index",
      "url": "index.json",
      "modules": null
    }
  ]
}
```

При сохранении (пункт меню Проект – Сохранить) сохраняются все добавленные элементы на страницу, путем сериализации их скриптом `serialization.js` и записи в json файл текущей страницы (листинг 3).

Листинг 3 – `index.json` после сохранения

```
[
  {
    "children": [
      {
        "children": [
          {
            "children": [

```

```

        "classes": [],
        "styles": "left: 217.6px; top: 16px; outline: rgb(0, 123, 255)
solid 2px; font-size: 16px; color: rgb(0, 0, 0); font-weight: 400;",
        "id": "49c202c8-7401-49f3-8c7a-d230efc6ff2e",
        "tag": "p",
        "value": null,
        "content": "Сохраненный тег p"
    }},
    "classes": ["container"],
    "styles": null,
    "id": null,
    "tag": "div",
    "value": null,
    "content": null
}},
"classes": [],
"styles": null,
"id": "f391be76-1302-40da-86a8-9699ad1ec09c",
"tag": "section",
"value": null,
"content": null
}}

```

При создании страницы (пункт меню Страница – Создать) создается json файл с введенным названием, в pages.cfg добавится запись об этой странице (листинг 4).

Листинг 4 – pages.cfg после создания страницы

```

{"pages": [
  {
    "title": "index",
    "url": "index.json",
    "modules": null
  },
  {
    "title": "page2",
    "url": "page2.json",
    "modules": null
  }
]}

```

Использование модулей требует описание сохранения каждого случая, поэтому в pages записывается в поле modules записывается название модуля, скрипты, которые нужно выполнить при загрузке страницы и дополнительные данные для запуска модуля (листинг 5).

Листинг 5 – pages.cfg после использования модуля

```

{"pages": [
  {
    "title": "index",
    "url": "index.json",
    "modules": null
  },
  {
    "title": "page2",
    "url": "page2.json",
    "modules": [{
      "initscripts": ["templates/modules/parallax/parallax.js"],
      "startup": null,
      "name": "parallax"
    }]
  }
]}

```

```

    }
  }
}

```

При переходе между страницами или загрузке проекта (переход на index.json соответствующей директории) данные из json файла используется в качестве аргумента функции из loaderElements.js. Также проверяется наличие модулей в pages.cfg для их запуска.

Экспорт сохраненной страницы реализует класс ExportSite. При экспорте происходит генерация HTML элементов на основе элементов, записанных в сохранённом файле страницы, использованные модули не экспортирует.

Таблица 7 – Описание класса ExportSite

Поле или заголовок метода	Пояснение
<code>public static void generate()</code>	Чтение конфигурационного файла, создание выходной директории, копирование изображений, генерация HTML для каждой страницы.
<code>public static String generateHtmlFromJson(JSONArray jsonArray)</code>	Генерирует HTML-контент из JSON-массива, для каждого объекта в массиве. Все элементы собираются в строку и возвращаются в виде единого HTML-контента.
<code>public static String createHtmlElement(JSONObject obj)</code>	Генерирует один HTML-элемент на основе данных, переданных в объекте JSON.

5 Тестирование приложения

Проверим работу основных функций на примере создания сайта на праздничную тематику.

При запуске приложения откроется окно выбора проекта (рисунок 23).

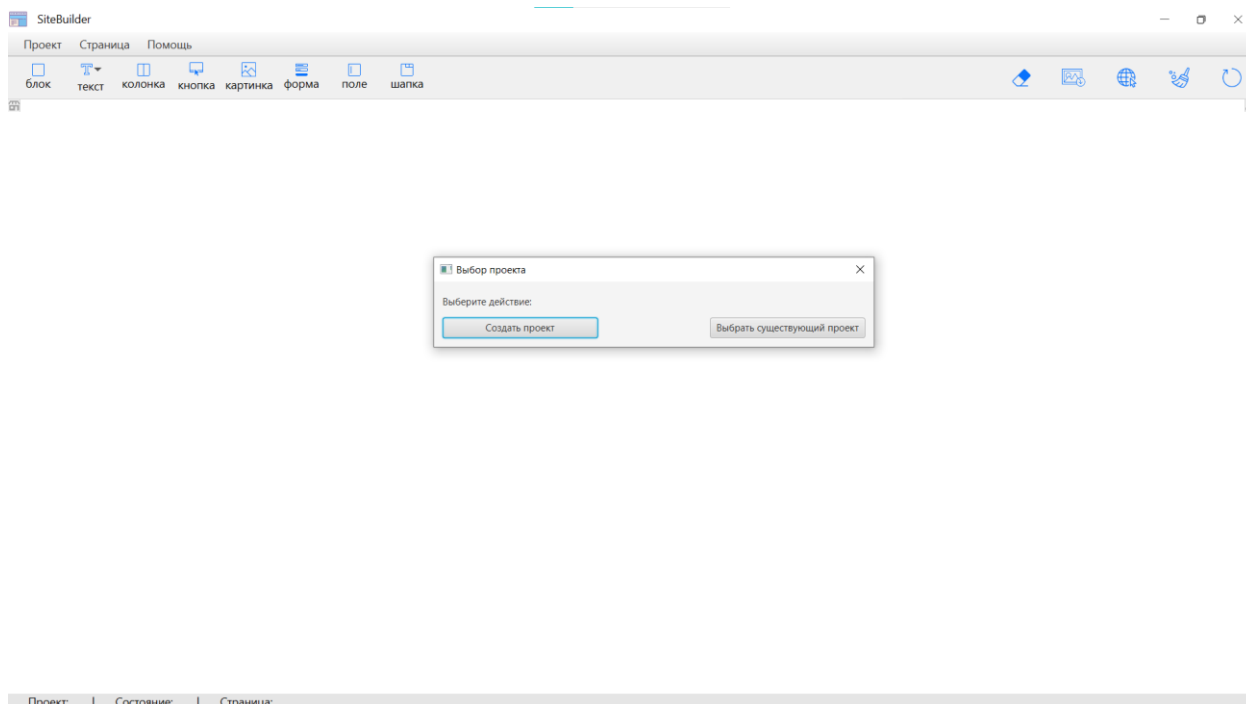


Рисунок 23 – Запуск приложения

Создадим проект (рисунок 24).

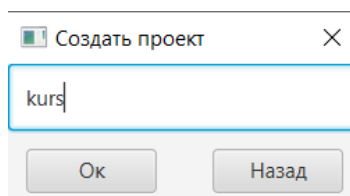


Рисунок 24 – Создание проекта

Начнем с оформления шапки, для этого нажмем на кнопку «Шапка» и введем количество отображаемых страниц (рисунки 25-26).

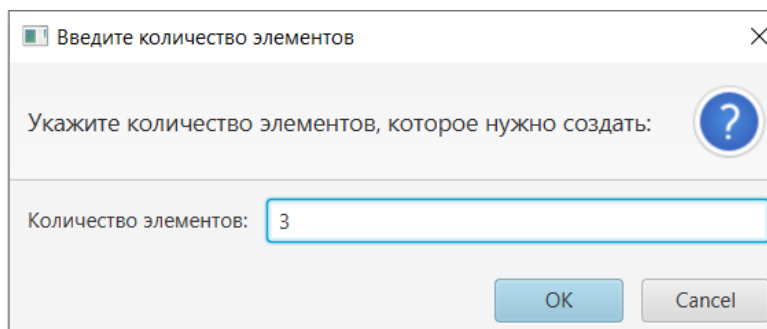


Рисунок 25 – Создание шапки

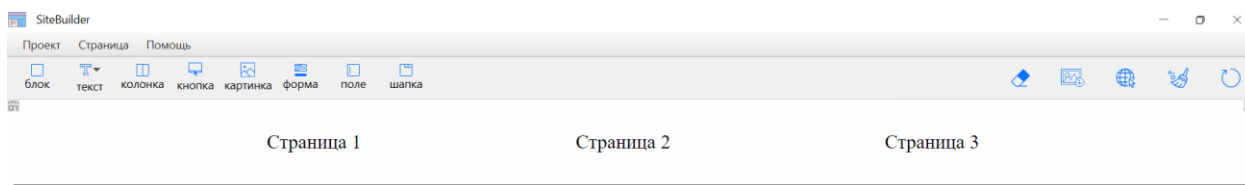


Рисунок 26 – Результат создания

Чтобы задать стилизовать подпись, нужно кликнуть по ней и задать параметры в панели справа (рисунок 27).

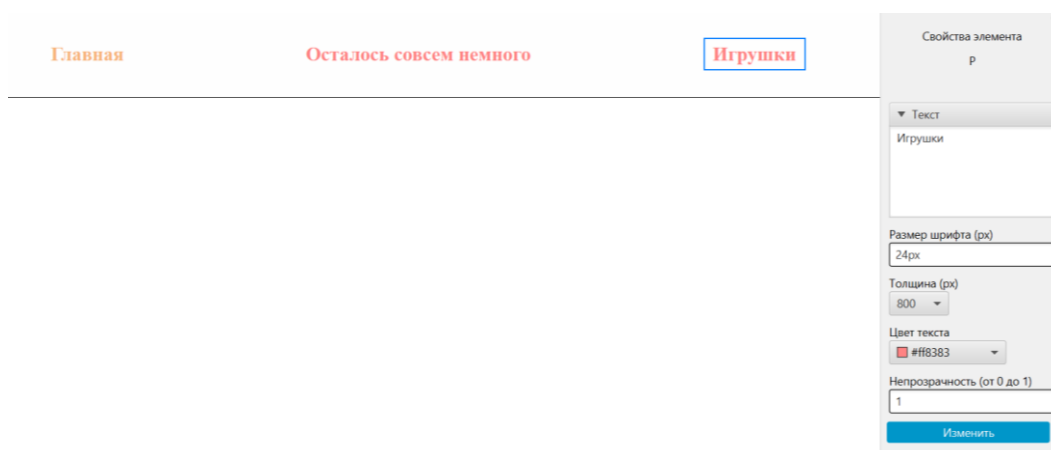


Рисунок 27 – Стилизация подписей

Создадим подпись в центре страницы нужно создать блок и убрать рамку из него, в панели редактирования по умолчанию стоит стиль none, поэтому достаточно просто нажать «Изменить» (рисунки 28-29), затем нужно выбрав блок, создать заголовок и стилизовать.

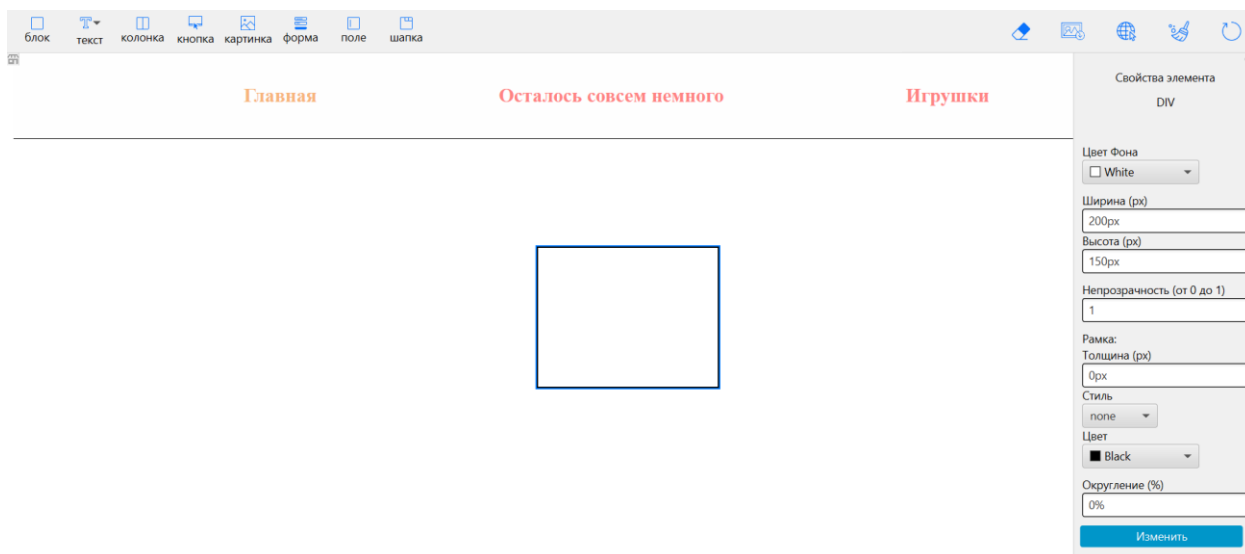


Рисунок 28 – Создание блока

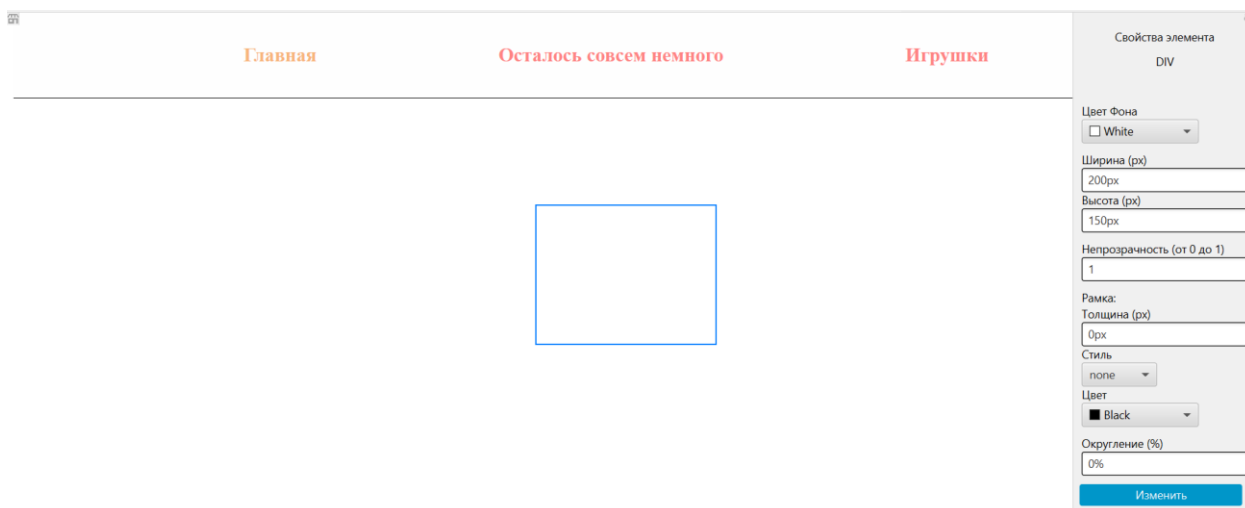


Рисунок 29 – Удаление рамок у блока

Добавим картинки в проект, для этого нужно сохранить проект, если ранее не был сохранен и нажать кнопку загрузки изображений. После выбрать загружаемые изображения (рисунки 30-31).



Рисунок 30 – Загрузить изображения

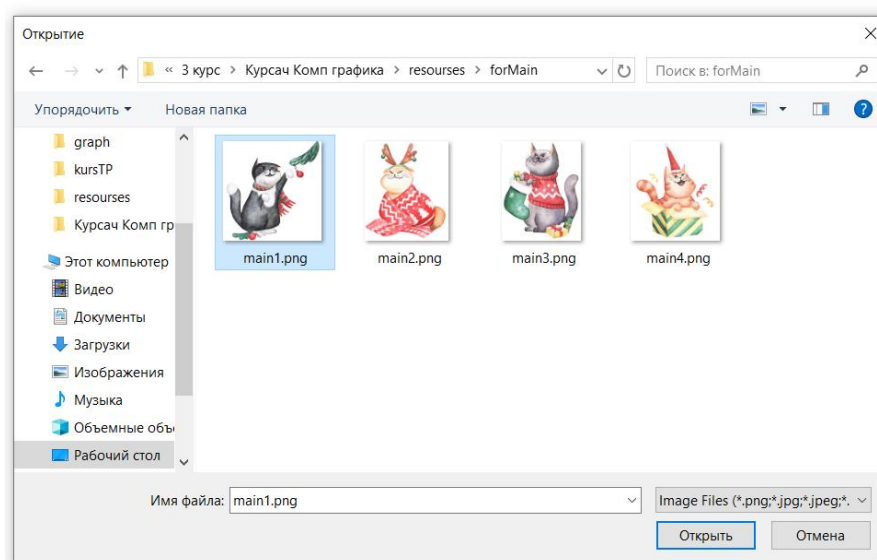


Рисунок 31 – Выбор изображений

Добавим изображения с параллакс эффектом, для этого нужно сначала кликнуть, где оно будет создаваться, затем откроется папка с изображениями проекта, нужно выбрать нужное (рисунок 32).

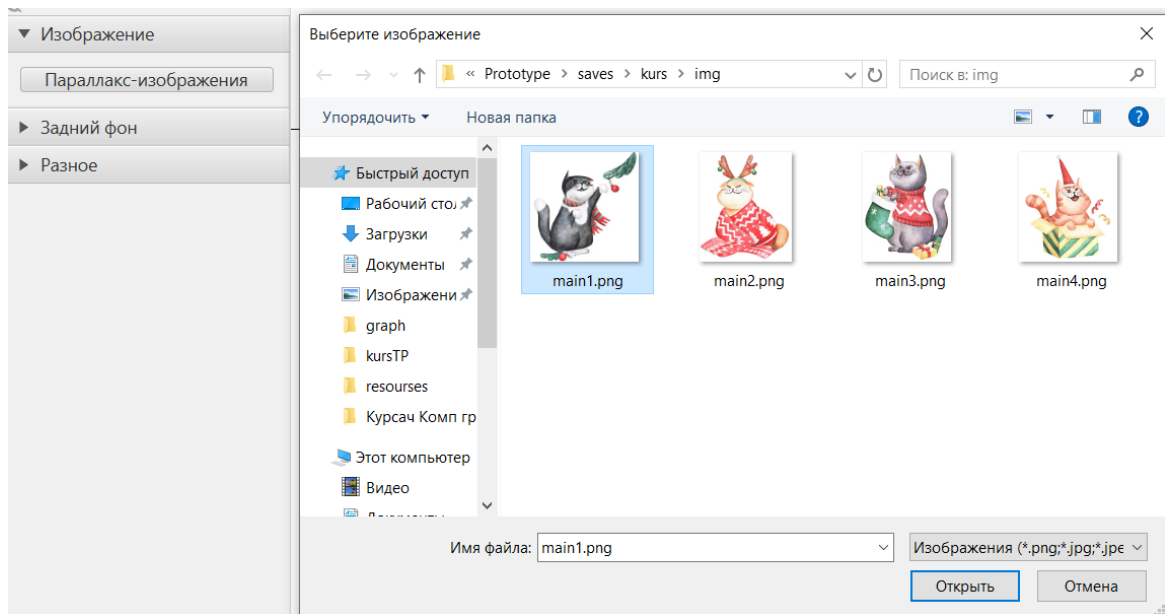


Рисунок 32 – Выбор изображения из проекта

Внесем силу смещения для изображений: 2, 1, -1, -2 (рисунок 33).

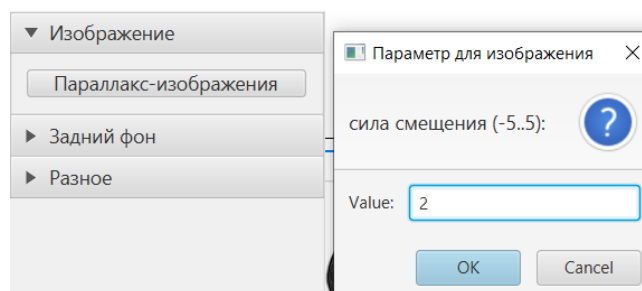


Рисунок 33 – Параметр смещения от курсора

Созданное изображение можно перемещать, а размер корректировать за его края (рисунок 34).

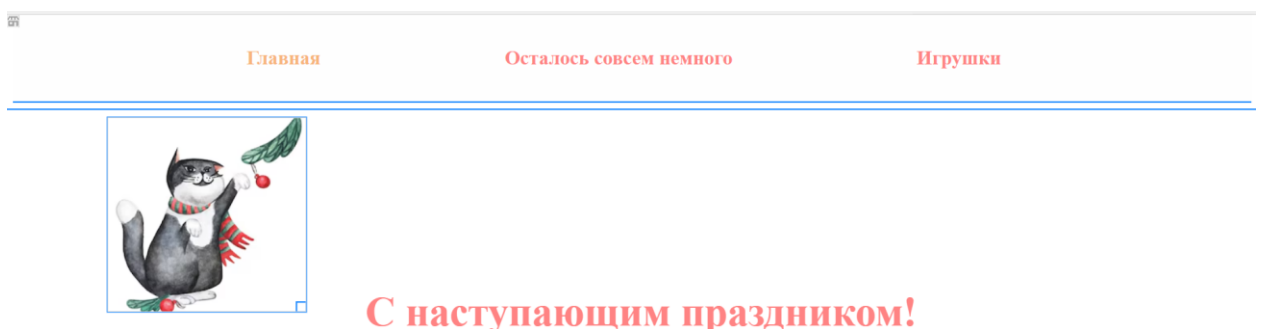


Рисунок 34 – Созданное изображение

Разместив аналогичным образом остальные изображения, завершим создание первой страницы (рисунок 35).

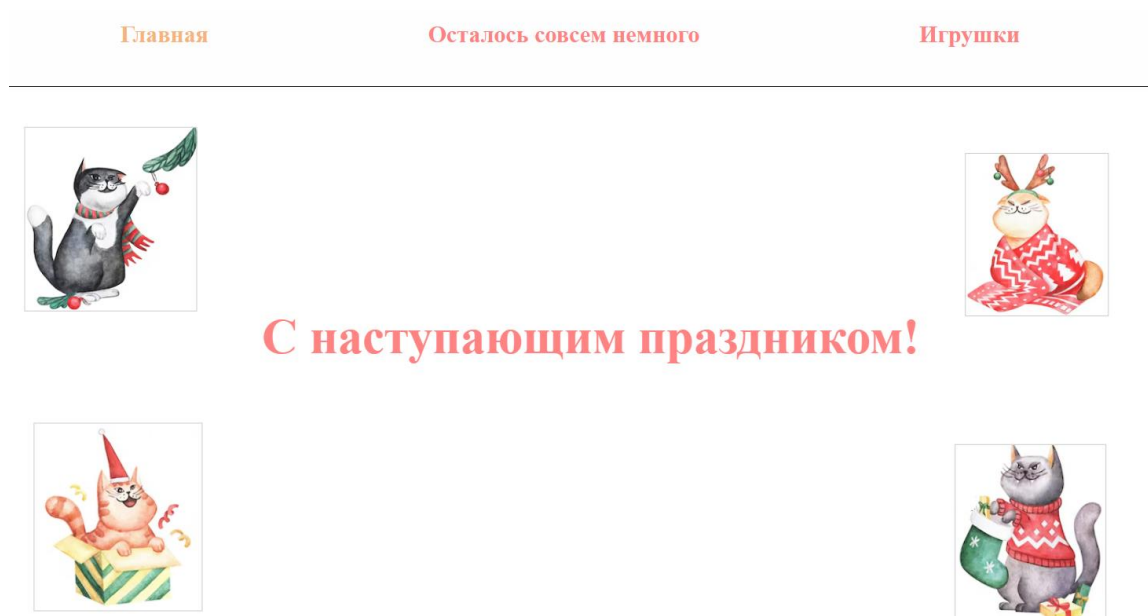


Рисунок 35 – Первая страница сайта

Затем создадим следующую страницу, выбрав пункт меню Страница – Создать. Окно создание страницы показано на рисунке 36.

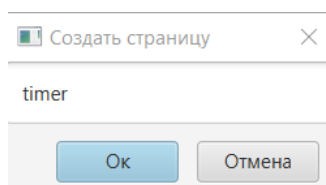


Рисунок 36 – Создание страницы

Повторив создание шапки, можно приступить к заполнению страницы. Выберем фон с снегопадом и создадим блок для отображения создаваемых элементов (рисунок 37).



Рисунок 37 – Использование зимнего фона

В блоке создадим таймер, предварительно настроив отсчет и отображение (рисунок 38) и разместим его в центре страницы.

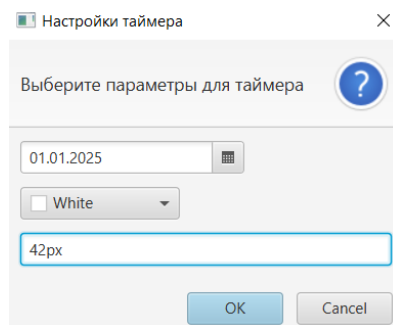


Рисунок 38 – Созданное изображение

Также создадим подпись для таймера и стилизуем её. Вторая страница сайта изображена на рисунке 39. Не забываем сохранить.

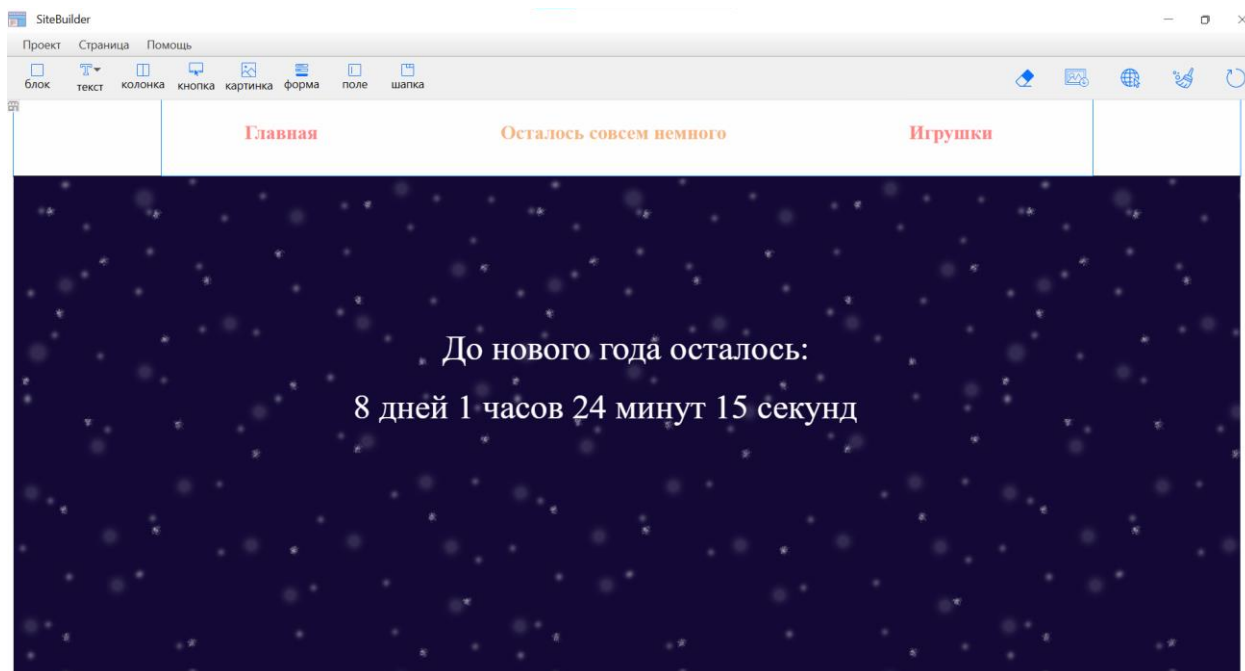


Рисунок 39 – Вторая страница сайта

Создадим третью страницу и после оформления шапки, загрузим изображения в проект чтобы создать блок с падающими картинками, указав количество каждой создаваемой картинки и выбрав их (рисунки 40-41).

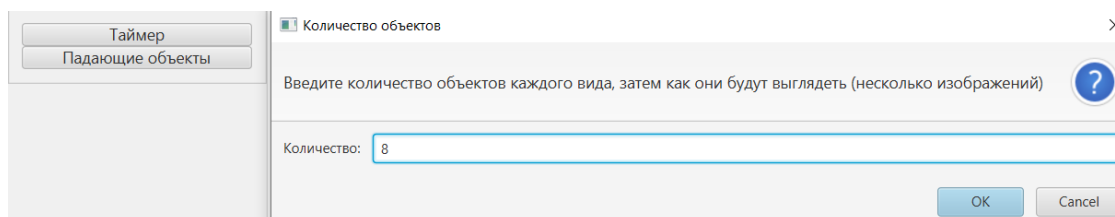


Рисунок 40 – Созданное изображение

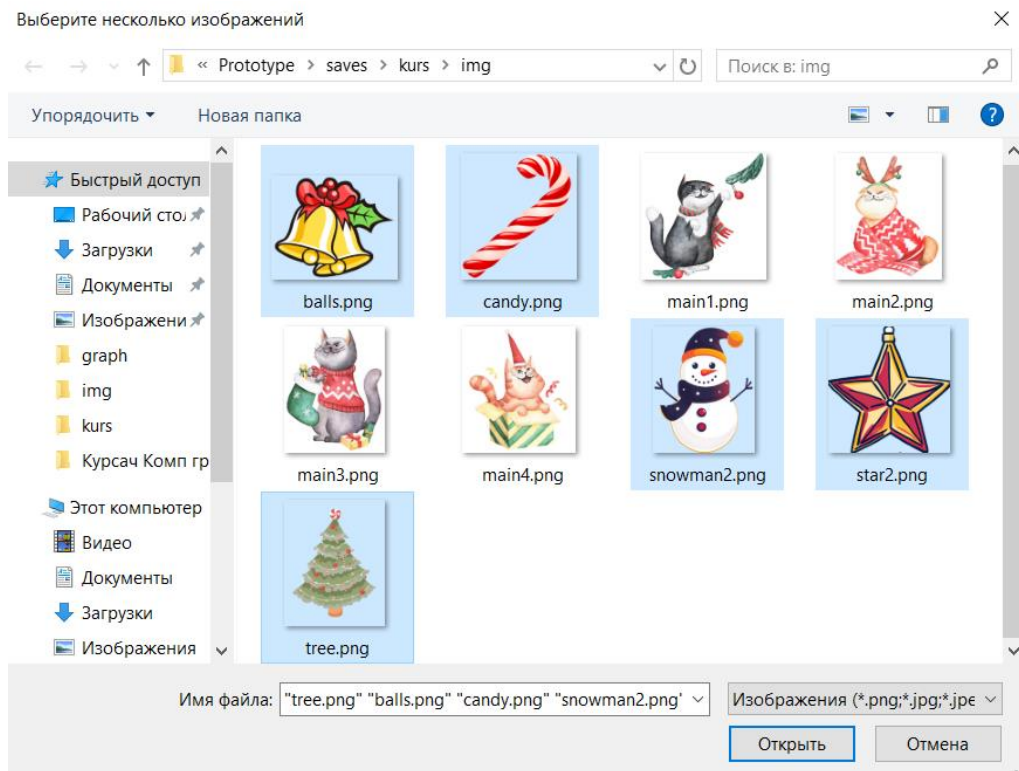


Рисунок 41 – Выбираем изображения из папки проекта

В результате созданные картинки упадут вниз страницы, их можно поднимать мышью (рисунок 42). Сохраняем страницу.



Рисунок 42 – Созданное изображение

Переключаясь между страницами или после создания нового проекта, загрузившись на ранее созданный можно убедиться, что страницы сохранились.

Нажав пункт меню Помощь – О программе откроется окно (рисунок 43).

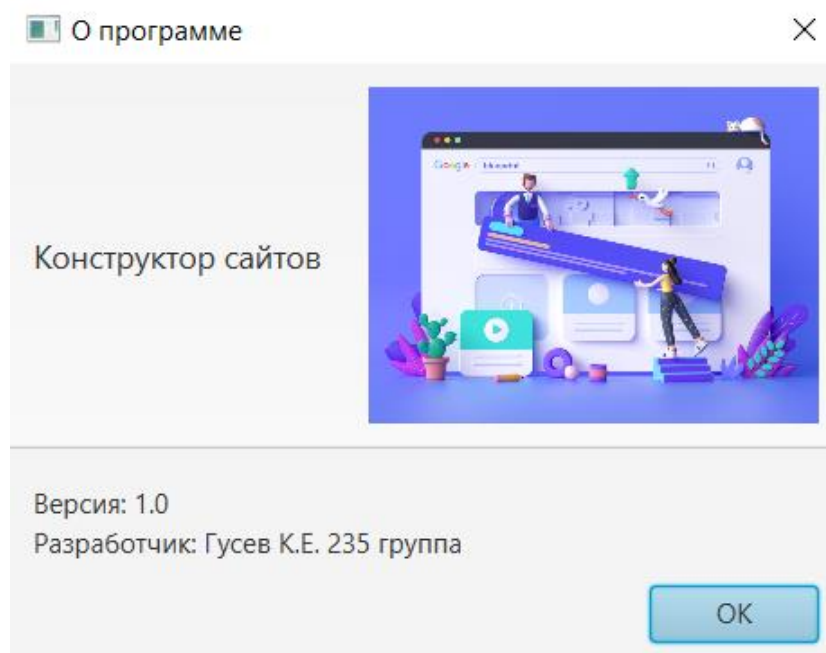


Рисунок 43 – Окно «О программе» вызываемое из меню

Заключение

В рамках данной курсовой работы было разработано приложение конструктор сайтов, использующий технологии JavaFX, Java CEF и реализующий возможность сохранения проекта сайта в формате JSON.

В процессе выполнения работы был разработан основной функционал конструктора сайтов, интегрирован веб-браузер на основе Java CEF для отображения страниц, реализована возможность сохранения проекта, что позволяет сохранять и загружать все страницы сайта.

В дальнейшем функционал конструктора сайтов можно значительно расширить, добавив новые возможности для создания и редактирования веб-страниц:

- увеличить количество доступных создаваемых элементов;
- добавление шаблонов заранее созданных частей веб-страницы;
- расширить функционал редактирования веб-элементов;
- поддержка адаптивного дизайна под разные устройства;
- поддержка различных типов контента (видео, аудио, слайдеры и т.д.);
- улучшить систему сохранения проекта и экспорта в HTML;
- реализовать возможность создавать HTTP запросы.

Список использованных источников

1. JavaFx // openjfx.io [Электронный ресурс] URL: <https://openjfx.io/> (дата обращения: 10.05.2024).
2. Chromium Embedded Framework (CEF) // CEF [Электронный ресурс] URL: <https://bitbucket.org/chromiumembedded/cef/src/master/> (дата обращения: 10.05.2024).
3. JSON // json [Электронный ресурс] URL: <https://www.json.org/> (дата обращения: 1.03.2023).
4. SDK для JavaFX // gluon [Электронный ресурс] URL: <https://gluonhq.com/products/javafx> (дата обращения: 10.05.2024).
5. e(fx)clipse // eclipse.dev [Электронный ресурс] URL: <https://eclipse.dev/efxclipse/index.html> (дата обращения: 10.05.2024).
6. Репозиторий CEF // github.com [Электронный ресурс] URL: <https://github.com/jcefbuild/jcefbuild/releases> (дата обращения: 10.05.2024).
7. Репозиторий org.json // mvnrepository.com [Электронный ресурс] URL: <https://mvnrepository.com/artifact/org.json/json> (дата обращения: 10.05.2024).