МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА SVG РЕДАКТОРОВ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Январева Алексея Алексеевича

Научный руководитель
зав. каф. техн. прогр., к. ф.-м. н. _____ И. А. Батраева

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2021

ВВЕДЕНИЕ

В современном мире веб-разработчики работают с большим количеством форматов изображения: JPEG, PNG, GIF. Однако в последнее время все большее внимание акцентируется на таком формате как SVG.

SVG — это язык для описания двумерной графики XML. SVG разрешает рисовать три типа графических объектов: векторные графические фигуры (например, контуры, состоящие из прямых линий и кривых), изображения и текст. Графические объекты можно группировать, стилизовать, преобразовывать и объединять в ранее нарисованные объекты.

Актуальность выбранной темы дипломной работы обусловлена значимостью разработки таких редакторов, которые способны рисовать SVG элементы, так как SVG формат имеет большой ряд преимуществ по сравнению с остальными форматами.

- SVG имеет небольшой вес, что позволяет веб-разработчикам с легкостью интегрировать такие изображения в свой продукт, тем самым не думая о скорости загрузки изображения пользователем;
- SVG — векторный формат, что является плюсом при создании адаптивных веб-приложений;
- SVG работает с открытыми веб-стандартами;
- SVG легко управлять, то есть менять цвет самого объекта, менять цвет контура или использовать различные фильтры.

Целью данной работы является разработка веб-приложения, которое способно работать с SVG элементами.

На пути к достижению поставленной цели представляется целесообразным решить следующие задачи:

- Рассмотреть, что такое SVG формат;
- Рассмотреть, какие элементы существуют у SVG формата;
- Рассмотреть документацию библиотеки для работы с SVG;
- Разработать веб-приложение по работе с SVG элементами.

В настоящее время существует мало веб-приложений по работе с SVG изображениями, которые не только реализуют функции, необходимые для рисования SVG элементов, но и удобны для использования пользователем. Для того чтобы использовать все функции приложения по рисованию SVG, во всех приложениях требуется регистрация, что является проблемным вопро-

сом. Поэтому реализуемое приложение должно быть удобно для использования пользователем, то есть должны быть доступны все функции на раннем этапе, а также должно использовать регистрацию только для сохранения SVG изображений.

Поскольку выбранная тема дипломной работы актуальна, то реализуемое приложение имеет ряд особенностей по сравнению с другими веб-приложениями:

- возможность добавлять несколько полотен для рисования SVG изображения;
- авторизация и регистрация пользователя;
- возможность авторизованному пользователю сохранять SVG изображения;
- возможность авторизованному пользователю выгружать сохраненные SVG изображения на полотно.

Структура и объем работы. Для решения поставленных задач выполнена дипломная работа, включающая в себя введение, 2 основные главы, заключение, список источников из 20 наименований и 1 приложения. Работа изложена на 70 страницах, содержит 3 таблицы и 33 рисунка.

Первая глава имеет название «Теоретическая часть» и содержит информацию о SVG формате, его основных элементах и важных концепциях, а также о графических системах.

Вторая глава имеет название «Практическая часть» и содержит общую информацию о приложении, его клиентской и серверной частях.

Дипломная работа заканчивается заключением, списком использованных источников, а также приложением с кодом А.

Основное содержание работы

SVG формат. В данном разделе описывается, что такое SVG формат, его основные особенности, а также приводится пример кода SVG, в результате которого рисуется зеленый круг.

Графические системы. В данном разделе рассматривается две основные системы — растровая графика и векторная графика, а также описывается использование этих двух систем.

В растровой графике изображение представлено как прямоугольный массив элементов изображения или пикселей, расположенных в виде сетки для представления изображения. Каждый пиксель представлен либо своими цветовыми значениями RGB, либо индексом в списке цветов. Эта серия пикселей, также называемая растровым изображением, часто сохраняется в сжатом формате.

В системе векторной графики изображение описывается как серия геометрических фигур. Векторная графика использует математические уравнения для рисования ваших дизайнов. Эти математические уравнения преобразуются в точки, которые соединяются линиями или кривыми, также известными как векторные пути, и составляют все различные формы, которые вы видите на векторной графике.

Растровая графика наиболее подходит для использования с фотографиями, которые редко собираются в четкие линии и кривые.

Векторная графика используется в следующих целях:

- Программы компьютерного черчения (CAD), где точное измерение и возможность увеличивать рисунок, чтобы увидеть детали, очень важна.
- Программы для создания графики, которая будет напечатана на принтерах с высоким разрешением (например, Adobe Illustrator).
- Язык печати и обработки изображений Adobe PostScript; каждый напечатанный вами символ описывается в виде линий и кривых.
- Векторная система Macromedia Flash для создания анимаций, презентаций, и веб-сайты.

Важные концепции SVG. В данном разделе дается информация о важных концепциях SVG, таких как масштабируемость, графические объекты, растровые эффекты, шрифты и анимация.

Одно и то же изображение SVG может быть размещено в разных раз-

мерах на одной той же веб-странице и повторно использоваться в разных размерах на разных страницах. Графика SVG масштабируема, потому что одно и то же содержимое SVG может быть автономной графикой или может ссылаться или включать в другую графику SVG, тем самым позволяя сложной иллюстрации строиться по частям, возможно, несколькими людьми.

SVG изображение моделирует графику на уровне графических объектов, а не отдельных точек.

Графически насыщенный материал часто сильно зависит от конкретного используемого шрифта и точного расстояния между символическими знаками. Во многих случаях дизайнеры преобразуют текст в контуры, чтобы избежать любые проблемы с заменой шрифтов. Это означает, что исходный текст отсутствует и, следовательно, возможность поиска и доступность страдает. В ответ на отзывы дизайнеров SVG включает элементы шрифта, чтобы сохранять как текст, так и графический вид.

Анимация может быть произведена посредством манипулирования документов на основе сценариев, но сценарии труднее редактировать и обмениваться между собой инструментами разработки. Опять же в ответ на мнение дизайнерского сообщества, SVG включает декларативные элементы анимации, которые были разработаны совместно с рабочими группами SVG и SYYM. Это позволяет анимированным эффектам, общими для существующей веб-графики, быть выраженными в SVG.

Основные элементы SVG изображения. В данном разделе описываются такие основные элементы SVG формата, как path, rect, ellipse, line и polyline элементы, а также их атрибуты.

Элемент *path* является общим элементов для описания фигуры. Все базовые фигуры могут быть созданы с помощью элемента *path*. Элемент *path* представляет собой контур фигуры, которую можно заливать, обводить и тому подобное.

Элемент *ellipse* — базовая SVG фигура, используемая для создания эллипсов с помощью координат центра и обоих радиусов.

Элемент *line* — базовая SVG фигура, используемая для создания линии, связующей две точки.

Элемент *polyline* — базовая SVG фигура, используемая для создания прямых линий, соединяющие несколько точек. Обычно используется для создания

открытых фигур, поскольку последняя точка не обязательно должна быть соединена с первой точкой.

Общая информация. В данном разделе описывается информация о веб-приложении SVG редактор.

Клиентская часть данного приложения реализована с помощью скриптового языка программирования JavaScript, который интенсивно применяется для разработки веб-приложений, языка разметки HTML и метаязыка, основанного на CSS, SASS, который предназначен для увеличения уровня абстракции CSS-кода и упрощения файлов каскадных таблиц стилей.

Серверная часть реализована на скриптовом языке программирования JavaScript с использованием фреймворка Express.js, который предназначен для создания API. В основе хранении информации, приходящей с клиентской стороны, используется база данных PostgreSQL. Для того чтобы была возможность клиентскому приложению взаимодействовать с серверным приложением, оно было выложено на хостинг Heroku.

Клиентская и серверная часть веб-приложения взаимосвязана посредством регистрации и авторизации пользователя. При регистрации пользователь может создать свою учетную запись, которая будет использоваться в тот момент, когда пользователь будет входить в свою учетную запись. В то время как пользователь будет авторизирован в системе, он сможет сохранять свои SVG изображения в базе данных, а также выгружать эти сохраненные изображения на полотно, что является одной из особенностью реализуемого приложения.

Реализация клиентской части.

Архитектура приложения. SVG редактор реализован с использованием библиотеки svg.js— легковесной JavaScript библиотеки для управления и анимации векторной графики в формате SVG.

В качестве архитектуры веб-приложения была выбрана MVC архитектура. Она подразумевает разделение на Model — компонент, отвечающий за данные, а также определяет структуру приложения, View — компонент, отвечающий за взаимодействие с пользователем, и Controller — компонент, отвечающий за связь между Model и View и определяющий, как приложение реагирует на действия пользователя. В представленном клиентском веб-приложении за Model отвечает несколько классов `MainViewModel` и `SVGAreaModel`. Компоненты View и Controller в приложении разбиты на несколько классов, отвечающие

за разные элементы приложения.

Важно отметить, что все HTML элементы создавались через JavaScript, поэтому для более удобного создания таких элементов была создана функция `createElement`, которой на вход в качестве параметров подается названия тега, названия классов в виде массива, атрибуты в виде объекта и текстовая составляющая элемента.

Для сборки и минимизации JavaScript файлов используется конфигурационный файл Webpack. В нем прописаны настройки и пути, в которые нужно разместить собранные и минимизированные файлы.

Для осуществления часто используемых задач, таких как запуск сервера и сборка проекта, используется GULP — таск-менеджер для автоматического выполнения описанных задач.

Дизайн приложения. Дизайн приложения изначально был создан в приложении Figma — онлайн-сервисе для разработки интерфейсов и прототипирования с возможностью организации совместной работы в режиме реального времени. Предполагалось, что экран пользователя будет разбит на несколько частей:

- шапка приложения, в котором будет написано большими буквами *SVG EDITOR*;
- первая верхняя панель инструментов, в котором будут размещены элементы взаимодействия с полотном и файлами, а также размещен переключатель, который меняет язык приложения;
- вторая верхняя панель инструментов, которая появляется только при выделении SVG элемента, и с помощью которой существует возможность взаимодействия с SVG элементом;
- левое боковое меню инструментов, в котором будут размещены элементы по рисованию основных SVG элементов, а также заливки и выделения контура этих элементов;
- нижняя панель инструментов, предназначенная для создания новых полотен для рисования с сохранением старых полотен и переключения между ними;
- правое боковое меню, отвечающее за появление модального окна по регистрации, авторизации, и списку сохраненных SVG файлов конкретного пользователя;

— полотно, на котором можно рисовать SVG элементы из левого бокового меню инструментов.

Для реализации веб-приложения в едином стиле были выбраны темно-синие тона. Также для осуществления адаптивной верстки прорабатывался дизайн от 1024 пикселей в ширину и более.

Шапка приложения. Верстка шапки приложения реализована с помощью функции `createHeader`, которая создает контейнер *header* для заголовка *h1*. Шапка приложения никак не взаимодействует с пользователем, она только отображает заголовок приложения, поэтому компоненты, отвечающие за данные и связь между представлением и данными, не требуются.

Первая верхняя панель инструментов. Верстка данной панели реализована с помощью функции `createMenuContainer`, которая создает контейнер для кнопок данного меню.

Представленное выше меню полностью взаимодействует с пользователем. Кнопки данной панели выполняют следующие функции:

— кнопка «Create» создает нового чистого полотна;
— кнопка «Save» сохраняет полотно в качестве изображения в *.svg* формате;
— кнопка «Import» импортирует выбранное из компьютера SVG изображение на полотно;
— кнопка «Properties» устанавливает ширину и высоту представленного в приложении полотна в пикселях;
— кнопка «Get the code» показывает код SVG изображения;
— кнопка в виде «стрелки назад» возвращает действия назад, проделанные над полотном;
— кнопка в виде «стрелки вперед» возвращает действия вперед, проделанные над полотном.

Кнопка «Create», «Save», «Properties», «Get the code» вызывают модальные окна для взаимодействия с пользователем.

При нажатии на кнопку «Create» вызывается модальное окно, которое спрашивает, действительно ли пользователь хочет очистить полотно.

При нажатии на кнопку «Save» появляется модальное окно с текстовым полем, в которое можно написать имя SVG файла при сохранении на компьютер.

При нажатии на кнопку «Properties» вызывается модальное окно, в ко-

тором можно изменить размеры полотна в пикселях.

При нажатии на кнопку «Get the code» появляется модальное окно с кодом SVG изображения.

Вторая верхняя панель инструментов. Верстка данной панели реализована с помощью функции `createFunctionalAreaElements`, которая в зависимости от выбранного элемента создает контейнер для кнопок данного меню. На вход подается контейнер `containerPanel`, в котором будут кнопки и список кнопок `arrayBtn` для этого контейнера.

В зависимости от выбранного элемента в данной панели показываются соответствующие атрибуты. При изменении какого-либо поля из этой панели сразу же меняется вид элемента. При двух или более выбранных элементах появляется другая панель инструментов, которая выравнивает эти элементы относительно полотна.

Левое боковое меню инструментов. Верстка данного меню реализована с помощью функции `createToolsLeft`, которая создает контейнер для кнопок данного меню.

Кнопки данной панели выполняют следующие функции:

— кнопка в виде «указателя» позволяет выделять элементы;
— кнопка в виде «прямоугольника» позволяет рисовать прямоугольник (*rect* элемент) на полотне;
— кнопка в виде «эллипса» позволяет рисовать эллипс (*ellipse* элемент) на полотне;
— кнопка в виде «линии» позволяет рисовать линии (*line* элемент) на полотне;
— кнопка в виде «текста» позволяет рисовать поле (*text* элемент) на полотне, в которое можно написать текст;
— кнопка в виде «карандаша» позволяет рисовать «карандашом» (*path* элемент);
— кнопка в виде «ломаной линии» позволяет рисовать (*polyline* элемент);
— кнопка в виде «баночки с краской» позволяет заливать выбранным цветом содержимое внутри элемента;
— кнопка в виде «прямоугольников с контурами» позволяет заливать выбранным цветом контур элемента.

Нижняя панель инструментов. Верстка данной панели реализована с

помощью функции `renderToolsBottom`, которая создает контейнер для листов полотен. Функция `renderTabControl` создает кнопки для добавления и удаления SVG листа.

При нажатии на кнопку создания нового листа вызывается функция `createNewTab`, которая удаляет текущий активный лист с помощью функции `removeActiveConditionTab`, создает новый активный лист полотна и вызывает функцию `callNewController`. Данная функция удаляет все слушатели события с предыдущего активного листа, в список контроллеров добавляет старые контроллеры вместе с новым классом контроллера нового листа, и происходит инициализация последнего добавленного контроллера, то есть создаются новые классы контроллеров, таких как *SwitcherLanguageController*, *TabsController* и *LoadingController*, которые отвечает за смену языка, контроля изменения листов полотен и загрузки содержимого SVG изображения из *localStorage* после перезагрузки веб-браузера. То есть при каждом действии все содержимое SVG изображения сохраняется в *localStorage*, что позволяет не терять информацию о содержимом SVG изображении после перезагрузки веб-браузера.

При нажатии на кнопку удаления листа вызывается функция `closeTab`, которая удаляет контроллеры выбранного листа из массива контроллеров, удаляет выбранный лист полотна из списка листов и устанавливает текущий активный лист.

Правое боковое меню инструментов. Верстка данного меню реализована с помощью функции `createElement`, которая создает контейнер для кнопки входа. Функция `renderProfile` заменяет кнопку входу на другой набор кнопок, когда пользователь авторизован.

При нажатии на кнопку входа с помощью функции `openModalSignIn` появляется модальное окно входа, в котором находятся два поля для обязательного заполнения E-mail и пароля пользователя. Если одно из полей не заполнено, то выводится под данными полями сообщение о том, что нужно ввести E-mail и пароль. Если оба поля заполнены, то при нажатии на кнопку «Sign In» вызывается функция с помощью слушателя события `click` вызывается функция-обработчик `onSignInModalClick`, которая отправляет POST запрос на авторизацию пользователя. Если сервер отвечает кодом 200, то кнопка заменяется на набор кнопок, и выполняется вход в систему.

При нажатии на кнопку «Sign Up» появляется новое окно с помощью функции `openModalSignUp`, в котором находятся три поля для обязательного заполнения имени пользователя E-mail и пароля пользователя. Если все три поля заполнены, то при нажатии на кнопку «Sign Up» вызывается функция с помощью слушателя события `click` вызывается функция-обработчик `onSignUpModalClick`, которая отправляет другой POST запрос на регистрацию пользователя. Если сервер отвечает кодом 200, пользователь успешно зарегистрирован. Пользователю необходимо будет войти в систему для сохранения своих SVG изображений на сервер.

При нажатии на кнопку в виде «папки» с помощью функции `renderModalOpen`, которой на вход подается имя файлов, открывается модальное окно, состоящее из списков файлов.

При нажатии на кнопку в виде «дискеты» с помощью функции `openModalSave` открывается модальное окно сохранения файла, в котором можно указать имя файла.

Контекстное меню. Верстка данного меню реализована с помощью функции `createContextMenuModal`, которая создает контейнер для кнопок данного меню.

При нажатии на полотно правой кнопкой мыши с помощью события `contextmenu` вызывается функция-обработчик `onContextMenuClick`, при которой появляется контекстное меню.

В данном меню доступность представленных выше кнопок зависит того, выделен ли SVG элемент. Данные кнопки реализуют следующие функции:

- кнопка «Delete» позволяет удалять выделенные элементы;
- кнопка «Copy» позволяет скопировать выделенные элементы;
- кнопка «Paste» позволяет вставить скопированные элементы;
- кнопка «Bring to Front» переводит элемент наверх относительно полотна;
- кнопка «Send to Back» переводит элемент вниз относительно полотна.

Переключатель языков. Верстка данного переключателя реализована с помощью функции `createSwitcherContainer`, которая создает контейнер для HTML элемента *input* типа *checkbox*. При клике на переключатель с помощью слушателя события `click` вызывается функция-обработчик `onSwitcherLanguageClic` которая в зависимости от состояния переключателя переводит приложение либо на английский язык, либо на русский язык. Данный

переключатель переводит не только главный экран, но также подсказки к кнопкам в левом боков меню инструментов (*tooltips*), все модальные меню и контекстное меню.

Реализация серверной части. Серверное приложение реализовано с использованием Node.js, поэтому для начала разработки данного приложение необходимо скачать Node.js на официальном сайте.

В данном приложение были задействованы следующие пакеты, которые устанавливались с помощью команды `npm install название_пакета`:

- *express* — минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений. Он спроектирован для создания веб-приложений и API.
- *cors* — пакет, который предоставляет возможность на получение ответа на сгенерированный AJAX запрос к другому веб-сайту, находящемуся по другому адресу.
- *knex* — пакет, который предоставляет возможность строить SQL запросы более гибко и удобно с помощью асинхронных вызовов через «точку» в JavaScript.
- *pg* — пакет, позволяющий подключиться к базе данных PostgreSQL.
- *uuid* — пакет, генерирующий id в виде стандарта идентификации UUID.

Для хранения информации о пользователе в pgAdmin для PostgreSQL была создана база данных с таблицей *users* со следующими колонками:

- id типа uuid для хранения id пользователя;
- username типа character varying(255) для хранения имен пользователя;
- email типа character varying(255) для хранения email пользователя;
- passwordHash типа character varying(255) для хранения хэша пароля пользователя;
- filenames типа text[] для хранения имен файлов SVG изображений;
- projects типа text[] для хранения структуры SVG изображения;

Для того чтобы пользователь смог получать информацию с помощью запросов к серверу, был создан роутер с помощью вызова `Router`. К этому роутеру были прописаны HTTP методы GET, POST и PUT.

Для того чтобы зарегистрироваться, к роутеру был прописан HTTP метод POST, к которому можно обратиться по адресу */register*. Тело данного запроса имеет следующие поля:

— username, которое содержит имя пользователя;
— email, которое содержит email пользователя;
— password, которое содержит пароль пользователя;

В описанном выше выше POST методе генерируется id пользователя с помощью функции `uuid` вида UUID. Далее извлекаются все описанные выше поля из тела данного запроса и проверяется, зарегистрирован ли указанный в запросе пользователь, то есть в базу данных совершается асинхронный запрос SELECT с помощью выборки по полям email и passwordHash, которое является хэшом пароля. Далее проверяется ответ от базы данных: если ответ от базы данных не является нулевым значением, то данный пользователь уже зарегистрирован, поэтому в ответе отправляется JSON объект с HTTP кодом 400 и причиной ошибки «Email already exists», иначе отправляется JSON объект с HTTP кодом 200, что является успешной регистрацией.

Для того чтобы пользователь смог авторизоваться, был создан HTTP метод POST, к которому можно обратиться по адресу */login*. Тело данного запроса имеет следующие поля:

— email, которое содержит email пользователя;
— password, которое содержит пароль пользователя;

В данном POST методе, предназначенным для авторизации, совершается SQL запрос SELECT с помощью выборки по полям email и passwordHash. Далее проверяется ответ от базы данных: если ответ от базы данных не является нулевым значением, то данный пользователь зарегистрирован в системе, поэтому в ответе отправляется JSON объект с HTTP кодом 200, что является успешной авторизацией, иначе отправляется JSON объект с HTTP кодом 403 и причиной ошибки «Invalid password».

Для того чтобы пользователь имел возможность хранить информацию о SVG изображениях, был создан HTTP метод PUT, к которому можно обратиться по адресу */save*. В данном методе сначала извлекаются все поля из тела запроса, которые были переданы клиентом, после чего проверяется актуальность информации о SVG изображениях. После всех проверок в ответ на запрос отправляется JSON объект с полями filenames и projects.

Для того чтобы была возможность просматривать всех пользователей или конкретного пользователя с помощью id, было создано два HTTP метода GET. В данном методе делается асинхронный запрос к базе данных с помощью

SELECT, после чего возвращается JSON объект, который является ответом от базы данных.

ЗАКЛЮЧЕНИЕ

В настоящей дипломной был рассмотрен SVG как формат изображения. Были рассмотрены не только основные элементы SVG изображений, но и их важные концепции, такие как:

- Масштабируемость;
- Графические объекты;
- Растровые эффекты;
- Шрифты;
- Анимация.

В ходе проведенной работы были полностью решены вышеописанные задачи:

- Рассмотреть, что такое SVG формат;
- Рассмотреть, какие элементы существуют у SVG формата;
- Рассмотреть документацию библиотеки для работы с SVG;
- Разработать веб-приложение по работе с SVG элементами.

В ходе рассмотрения SVG формата было создано веб-приложения (SVG редактор), который имеет следующие уникальные особенности:

- возможность добавлять несколько полотен для рисования SVG изображения;
- авторизация и регистрация пользователя;
- возможность авторизованному пользователю сохранять SVG изображения;
- возможность авторизованному пользователю выгружать сохраненные SVG изображения на полотно.