Санкт-Петербургский Государственный Университет

Математико-механический факультет

Кафедра системного программирования

Разработка редактора диаграмм для облачной технологии создания мобильных приложений

Курсовая работа студента 361 группы

Белокурова Дмитрия Николаевича

Научный руководитель _____ Т. А. Брыксин

ст. преподаватель /подпись/

Санкт-Петербург

2012

Оглавление

1 Введение

За последние несколько лет мобильные устройства набрали огромную популярность. Этому во многом способствует наличие множества приложений для подобных устройств. Но многие пользователи хотели бы иметь возможность получить нужное им приложение, которые имеет некую специфику и, возможно, представляет собой ценность только для небольшой группы пользователей. Кроме того, эти пользователи могут не иметь навыков программирования.

Помимо роста числа смартфонов и тому подобных устройств сейчас наблюдается тенденция по переходу от стандартных приложений, устанавливаемых на устройство пользователя, к онлайн-сервисам, которые позволяют пользователю решать свою задачу, не устанавливая каких-либо приложений. Пользователю необходимо только наличие доступа в интернет и наличие интернет-браузера на своём устройстве.

Ещё одним заметным фактом является повышенный интерес к предметно-ориентированным языкам. Предметно-ориентированные языки максимально приближены к предметной области и, как правило, имеют понятную нотацию. Конструкции языка отражают сущности из предметной области, что помогает разработчику сконцентрироваться на решении задачи. Подобные предметно-ориентированные языки могут иметь и графическую нотацию, что ещё упрощает разработку средств решения задач в некой области.

Стоит заметить, что в данный момент на рынке присутствуют решения для быстрого создания мобильных приложений, но они, как правило, не позволяют разрабатывать приложения с нетривиальной бизнес-логикой, что

ограничивает сферу применимости подобных сервисов только разработкой статичных приложений, которые скорее носят информационный характер.

Сопоставив эти факты, можно прийти к выводу, что разработка сервиса для создания приложений под мобильные устройства, позволяющего разрабатывать приложения с нетривиальной логикой, является актуальной задачей. А описанные выше визуальные предметно-ориентированные языки могут найти применение в описании логики работы приложения.

На данный момент развивается проект, целью которого является создание онлайн-сервиса для быстрой разработки мобильных приложений с нетривиальной логикой под популярные платформы. При этом от пользователя не ожидается владение какими-либо навыками программирования. Для разработки приложений предполагается использование графических предметно-ориентированных языков.

2 Постановка задачи

В рамках проекта разрабатываются генераторы кода под популярные платформы, дизайнер пользовательских интерфейсов и визуальный редактор диаграмм.

Редактор диаграмм служит пользователю для описания логики работы приложения при помощи графической нотации. При этом пользователь способен конструировать схему логики при помощи палитры инструментов, содержащей логические элементы работы приложения.

Так как разработка будет вестись исключительно в интернет-браузере, то желательными свойствами редактора являются кроссбраузерность и отсутствие необходимости в установке программного обеспечения, такого как Microsoft Silverlight, Adobe Flash, Java Web Start.

В рамках данной работы передо мной была поставлена задача по разработке редактора диаграмм, удовлетворяющего следующим требованиям:

- Кроссбраузерность. Редактор должен одинаково работать во всех современных популярных интернет-браузерах.
- Отсутствие требования в наличии каких-либо дополнений к браузеру или иного программного обеспечения, кроме самого браузера.
- Возможность описания логики работы разрабатываемого приложения в графической нотации.
- Присутствие таких компонент пользовательского интерфейса как палитра инструментов, основная рабочая область (сцена) и редактор свойств. Основная рабочая область содержит разрабатываемую схему логики приложения. Палитра инструментов содержит компоненты, которые пользователь может добавить на рабочую область. Редактор

свойств позволяет изменять некоторые свойства объектов на основной рабочей области.

3 Обзор существующих решений

На данный момент существуют популярные сервисы по разработке приложений для мобильных телефонов, которые поддерживают популярные платформы смартфонов. В своей работе я решил рассмотреть некоторые из них.

- **IBuildApp**

  Первым рассматриваемым решением стал онлайн-сервис iBuildApp.

  Сервис позволяет частным лицам и кампаниям разрабатывать приложения для iPhone, iPad и Android. Сервис обладает web-интерфейсом, написанным с использованием языка JavaScript. Сервис не требует от пользователя навыков программирования, что позволяет использовать его для разработки широкому кругу лиц. Обладает возможностью предоставлять созданное пользователем приложение через облачную инфраструктуру.

  Недостатком сервиса является отсутствие возможности разработки приложения, обладающего какой-либо бизнес логикой. Таким образом, при помощи сервиса можно разработать только статичное приложение.

- **App.co**

  В качестве второго рассматриваемого сервиса я выбрал App.co.

  Так же, как и iBuildApp является сервисом по разработке мобильных приложений для популярных платформ. Обладает web-интерфейсом. Как и предыдущий рассмотренный сервис не требует от пользователя навыков программирования. При этом обладает тем же самым недостатком – невозможностью разработки

приложения со сложной логикой работы. На момент написания отчёта по данной работе сервис оказался недоступен для пользователей.

- **ViziApps (MobiFlex App Studio)**

Визуальная система для разработки нативных и web-приложений для мобильных телефонов. Позволяет создавать приложение на основе одного из множества шаблонов приложений. Элементы создаваемого пользовательского интерфейса приложения могут быть связаны с такими функциями, как отправка электронной почты, SMS-сообщения, отправка на сервисы Facebook и Twitter.

Поддерживает создание приложений под такие платформы, как iPhone, iPad, Android. Как и предыдущие сервисы, не требует от пользователя навыков программирования.

Тем не менее, создать приложение с более сложной логикой исполнения не представляется возможным.

- **MakeMeDroid**

Последним из рассмотренных мной сервисов является сервис MakeMeDroid. Это сервис для создания приложений для мобильных телефонов на платформе Google Android. Другие платформы не поддерживаются.

Как и остальные сервисы, позволяет разрабатывать приложение не обладая навыками в программировании. Предоставляет возможность разместить приложение на Android PlayMarket.

Среди недостатков стоит отметить невозможность создания приложений с нетривиальной логикой и ограниченность сервиса только платформой Android.

Таким образом, все рассмотренные решения задачи не позволяют разрабатывать приложения со сложной логикой работы, что послужило причиной для развития проекта.

4 Реализация решения

4.1 Инструменты

Так как среди требований к редактору диаграмм присутствуют кроссбраузерность и отсутствие необходимости в установке стороннего программного обеспечения, то в качестве средств для разработки было принято решение использовать HTML5 и JavaScript.

HTML5 в последнее время набирает всё большую популярность. Почти все современные браузеры могут похвастаться поддержкой HTML5 на приличном уровне.

Среди важных нововведений HTML5 стоит отметить наличие элемента <canvas>, позволяющего вести отрисовку сложных графических элементов на web-странице, и наличие встроенной возможности drag’n’drop, что значительно упрощает процедуру перемещения объектов на сцене перетягиванием указателем мыши.

JavaScript является наиболее распространённым языком для web-программирования на стороне клиента. На данный момент язык хорошо стандартизован, поддерживается всеми современными популярными интернет-браузерами. Это делает язык лучшим вариантом для реализации клиентской части сервиса.

Хоть связка технологий HTML5 + JavaScript и представляет из себя мощный инструмент, эти технологии не избавляют разработчика от некоторых проблем.

В качестве первой проблемы стоит отметить, что хотелось бы при разработке воспользоваться некоторыми шаблонами проектирования, в том числе MVC. Но объектная модель языка JavaScript к этому не располагает, что

во много затрудняет следование обозначенной архитектуре приложения при написании кода.

Второй проблемой оказалась сложность создания приятного и дружественного пользователю пользовательского интерфейса стандартными средствами HTML5.

Решением этих двух проблем стало использование JavaScript-библиотеки Dojo Toolkit. Эта библиотека одновременно предоставляет средства для реализации шаблона MVC и для создания дружественного пользовательского интерфейса. Кроме того, библиотека очень хорошо документирована, что сокращает время, затрачиваемое на разработку с её помощью.

Не смотря на все преимущества использования библиотеки Dojo Toolkit, работа с объектами, отрисованными на элементе <canvas> по-прежнему представляет сложность. Работа с каждым отдельным элементом только средствами JavaScript утомительна и требует очень большой аккуратности. Как и в предыдущем случае, решение проблемы представилось в виде использования библиотеки KineticJS, которая разработана для работы со сложными графическими объектами на элементе <canvas>. Из недостатков библиотеки можно отметить полное отсутствие документации, кроме комментариев в исходном коде. Но тем не менее, библиотека после детального изучения оправдала своё использование, существенно упростив работу с множеством элементов, находящихся на <canvas>.

Таким образом, помимо HTML5 и JavaScript для решения поставленной задачи, было принято использовать две JavaScript-библиотеки. Обе распространяются по открытой лицензии.

4.2 Архитектура редактора

Общая архитектура сервиса изображена на схеме 1.

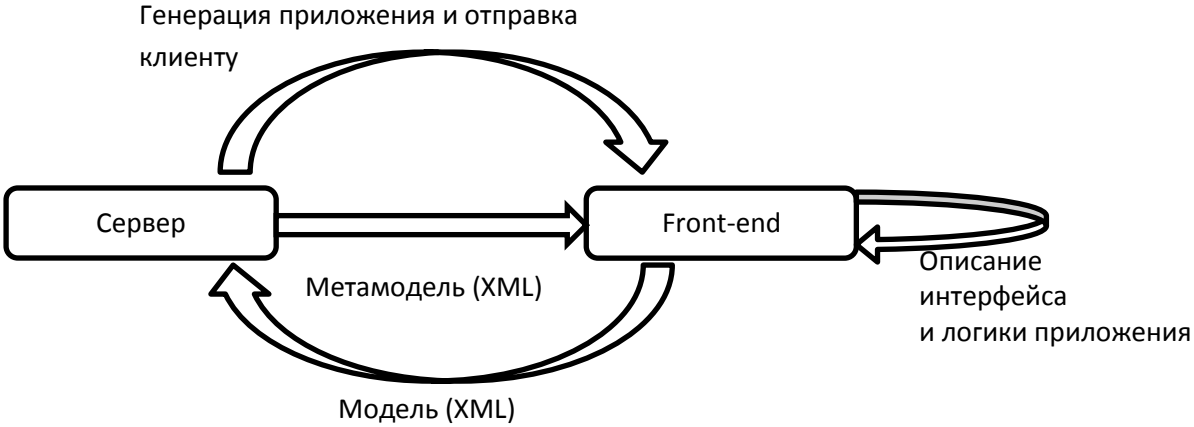

Схема 1. Общая архитектура сервиса.

Сервер генерирует метамодель, в которой описано, какие элементы могут присутствовать на диаграмме или в пользовательском интерфейсе. Затем эта метамодель в формате XML отправляется клиенту. На клиенте при помощи этой метамодели создаётся палитра и определяются свойства элементов. Пользователь рисует интерфейс и логику приложения. После этого нарисованная модель преобразуется в XML и отправляется на сервер, где генерируется приложение. Приложение может быть размещено на Play Market, AppStore и тому подобных сервисах. Так же клиенту может быть отправлена ссылка на скачивание готового приложения.

Форматы метамодели и модели описаны в работе Надежды Чижовой. Метамодель представляет собой специально структурированный XML, в котором описываются элементы палитры инструментов, множество графических примитивов для отрисовки объекта, соответствующего этому элементу, на сцене, а также свойства таких элементов. Модель, как и метамодель, представляет собой специальным образом структурированный XML, где описан каждый элемент сцены, его положение и свойства.

Архитектура разработанного мной редактора диаграмм показана на схеме 2.

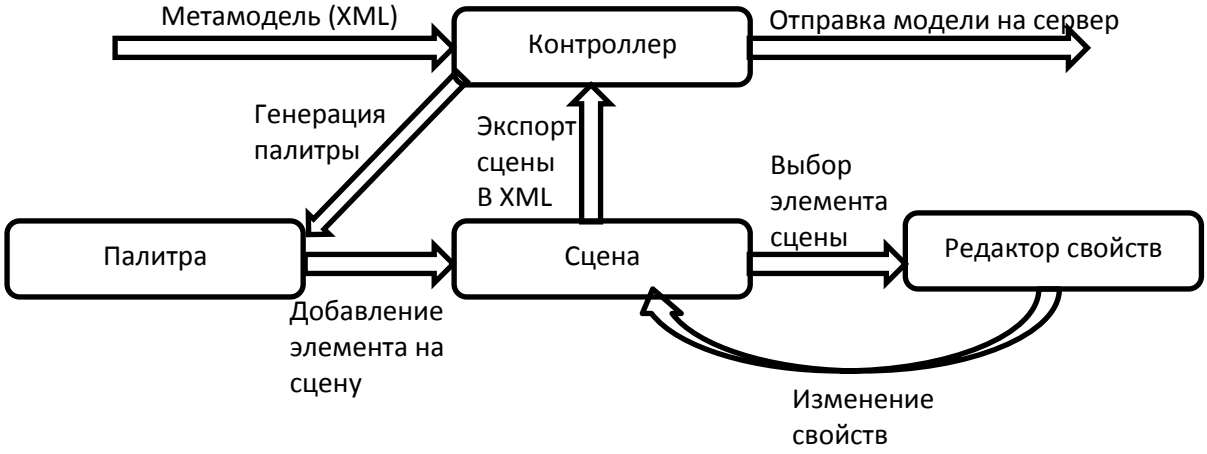

Схема 2. Архитектура редактора диаграмм.

Логически редактор состоит из трёх компонент.

Контроллер отвечает за взаимодействие с сервером, разбор метамодели, составление палитры по метамодели и за экспорт нарисованной схемы в XML и отправку её на сервер для генерации приложения.

Палитра генерируется контроллером. Она содержит все описанные в метамодели элементы, которые могут быть добавлены на сцену. Для каждого такого элемента она содержит информацию о свойствах, которыми обладают объекты, соответствующие этому элементу.

Основная рабочая область (сцена) содержит в себе текущую схему логики приложения, информацию о текущих значениях свойств этих элементов.

Редактор свойств отвечает за изменение свойств выбранного элемента сцены.

Сценарий работы редактора выглядит следующим образом. Контроллер получает XML от сервера, разбирает её и генерирует палитру элементов. Затем пользователь, выбрав нужный элемент палитры, добавляет его на сцену. На сцене пользователь может перетаскивать элементы, выбирать какой-либо из элементов. При выборе элемента в редакторе свойство отображаются свойства этого элемента, которые пользователь может изменить. По завершении работы контроллер преобразует текущую нарисованную схему в XML и отправляет на сервер, где генерируется приложение.

Интерфейс созданного прототипа редактора представлен на рисунке 1.

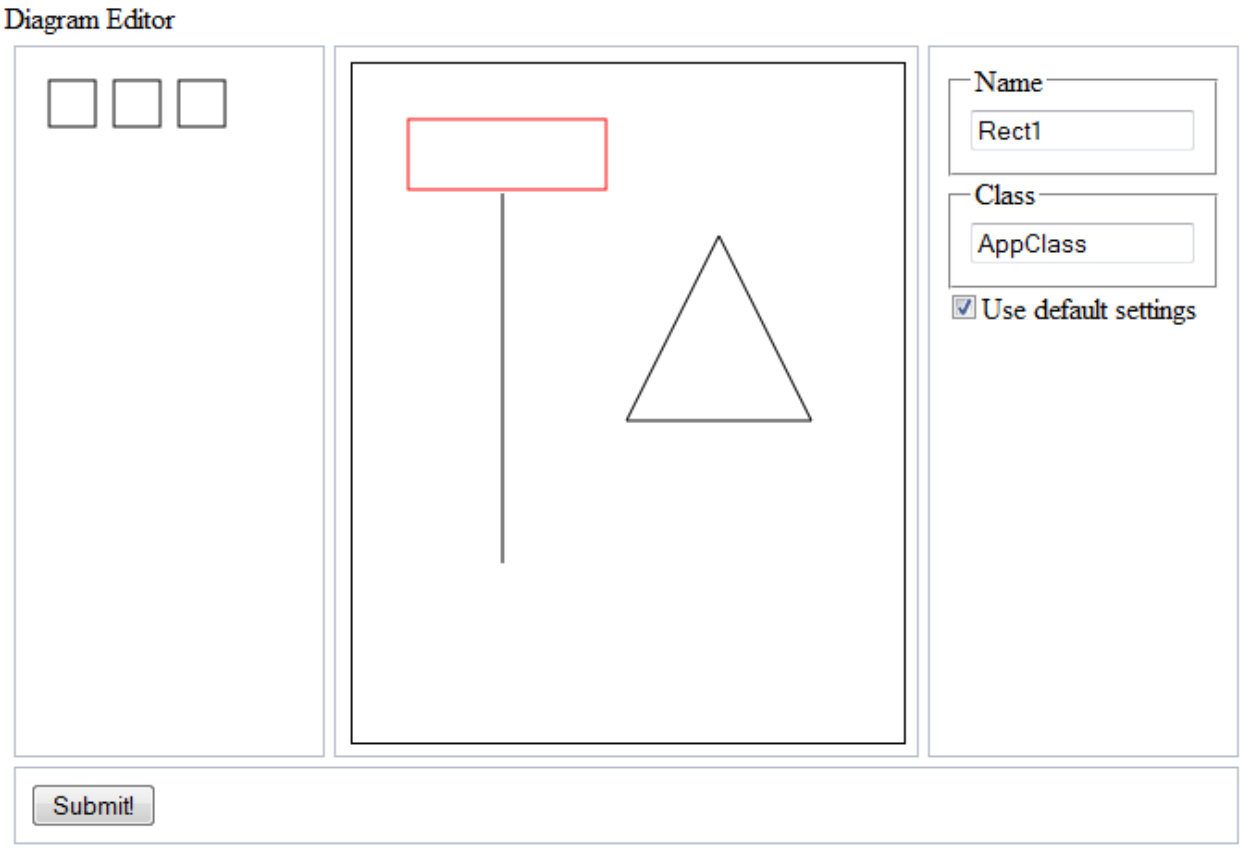

Рисунок 1. Пользовательский интерфейс прототипа редактора.

5 Результаты

В результате проделанной работы были получены следующие результаты:

- Составлены требования к редактору диаграмм.
- Изучены и выбраны библиотеки и средства, подходящие для решения данной задачи.
- Создан прототип редактора диаграмм, удовлетворяющий поставленным требованиям, состоящий из:
    - Палитры инструментов;
    - Редактора свойств;
    - Основной рабочей области.

Таким образом, выбранные средства и библиотеки можно признать подходящими для использования в данной предметной области.

6 Список используемых источников

[1] Документация по библиотеке Dojo Toolkit.

Дата обращения: 13.05.2012. URL:

http://dojotoolkit.org/documentation/

[2] Описание API библиотеки KineticJS.

Дата обращения: 14.05.2012. URL:

http://kineticjs.com/api-docs.php

[3] Спецификация HTML5.

Дата обращения: 20.05.2012. URL:

http://dev.w3.org/html5/spec/

[4] Статья о реализации шаблона проектирования MVC средствами JavaScript.

Дата обращения: 22.05.2012. URL:

http://designformasters.info/posts/mvc-javascript/

[5] Ресурс, посвящённый метамоделированию.

Дата обращения: 22.05.2012. URL:

http://metamodel.org/

[6] Руководство по использованию HTML5 Canvas.

Дата обращения: 22.05.2012. URL:

https://developer.mozilla.org/en/Drawing_Graphics_with_Canvas

[7] Статья, посвящённая анализу визуальных языков моделирования.

Дата обращения: 22.05.2012. URL:

http://www.science-education.ru/102-5655