САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Математико-механический факультет

Кафедра системного программирования

Белокуров Дмитрий Николаевич

Онлайн-редактор для разработки мобильных приложений

Бакалаврская работа

Допущена к защите.
Зав. кафедрой:
д. ф.-м. н., профессор А.Н. Терехов

Научный руководитель:
ст. преп. Т.А. Брыксин

Рецензент:
ст. преп. Ю.В. Литвинов

Санкт-Петербург
2013

SAINT-PETERSBURG STATE UNIVERSITY
Mathematics & Mechanics Faculty

Chair of Software Engineering

Dmitriy Belokurov

Online editor for mobile application development

Bachelor's Thesis

Admitted for defence.
Head of the chair:
professor A.N. Terekhov

Scientific supervisor:
assistant T.A. Bryksin

Reviewer:
assistant Y.V. Litvinov

Saint-Petersburg
2013

Оглавление

Введение

В последние годы в мире происходит повсеместное распространение различных мобильных устройств, таких как смартфоны и планшетные компьютеры. Сейчас они позволяют решать многие задачи, выполнение которых несколько лет назад требовало присутствия рядом с персональным компьютером. Такие задачи могут включать в себя работу с электронной почтой, поиск информации в интернете, общение через сервисы мгновенного обмена сообщениями, доступ к социальным сетям и т.д.

Помимо самого распространения мобильных платформ как таковых наблюдается рост рынка мобильных приложений. Приложения разрабатываются для совершенно разных задач: GPS-навигация, работа с текстом, интернет-обозреватели. Даже для самой странной задачи, как например для эмулятора игры на губной гармошке, можно найти приложение.

Данная работа выполняется в рамках проекта QRealWeb по реализации веб-сервиса, позволяющего непрофессиональным пользователям создавать сложные мобильные приложения. Целью моей работы является реализация одной из компонент такого сервиса, а именно дизайнера интерфейса и логики приложения.

0.1. Требования к редактору

К разрабатываемому редактору предъявляются следующие требования:

- отсутствие необходимости пользователю иметь навыки программирования;
- кроссбраузерность;
- возможность задания интерфейса приложения;
- поддержка интеграции с серверной частью веб-сервиса и онлайн-эмулятором.

0.2. Задачи в рамках выпускной работы

Для реализации требований в рамках выпускной работы были поставлены следующие задачи:

- проанализировать существующие решения;
- выбрать подходящие инструменты для реализации требуемого функционала;
- реализовать редактор интерфейса приложения, добавить поддержку основных элементов управления;
- реализовать редактор логики обработчиков событий;
- обеспечить возможность сериализации разрабатываемого приложения в формат, пригодный для взаимодействия с другими элементами QRealWeb;
- апробировать связанные компоненты веб-сервиса на разработке конкретного приложения.

1. Предметная область

Распространение мобильных устройств и рост рынка мобильных приложений связано с некоторыми трудностями. Число различных платформ велико, средства для разработки приложений для них различны: для Android[1] чаще всего используется Java[2], для iOS[3] - Objective C[4] и т.д.

Кроме того, разработка под мобильные платформы по-прежнему требует у пользователя навыки программирования. Таким образом, для пользователя-непрофессионала практически невозможно своими силами создать приложение под свои нужды. Многие компании малого и среднего бизнеса не готовы нанять разработчика для написания мобильного клиента для предоставляемого компанией сервиса.

Одним из решений такой проблемы является использование различных фреймворков для мобильной разработки. Они позволяют писать приложение, практически не задумываясь о различии в платформах. В итоге разработчик получает целый набор приложений под различные мобильные устройства. Но при этом от пользователя подобных фреймворков по-прежнему требуется умение программировать.

С другой стороны к проблеме подходят различные онлайн-сервисы, которые позволяют пользователю, не обладающему нвыками программирования, создавать простые приложения. Для этого они предоставляют пользователю дизайнер интерфейса приложения, позволяют задавать переходы между экранами приложения. Такой подход удобен для непрофессионального пользователя: можно создавать приложения с красивым интерфейсом. Но использование подобных сервисов накладывает на создаваемое приложение ограничения, такие как невозможность создать приложение со сложной логикой работы, агрегацией данных и взаимодействием со сторонними сервисами.

[1]Android. - URL: http://www.android.com . - Дата обращения: 20.05.2013.

[2]What is Java? - URL: http://java.com/en/download/whatis_java.jsp . - Дата обращения: 20.05.2013.

[3]Apple - iPhone 5 - Read all about the amazing iOS. - URL: http://www.apple.com/iphone/ios . - Дата обращения: 20.05.2013.

[4]Objective-C - Wikipedia, the free encyclopedia. - URL: https://en.wikipedia.org/wiki/Objective-C . - Дата обращения: 20.05.2013.

Таким образом, на рынке существуют сервисы, которые позволяют упростить программисту разработку приложений со сложной логикой. Существуют сервисы, которые позволяют пользователю без навыков программирования создавать простые мобильные приложения. Но нет сервиса, который позволил бы пользователю, не владеющему навыками программирования, разрабатывать приложения со сложной логикой.

1.1. QRealWeb

Чтобы решить проблему, описанную в предыдущей главе, на кафедре системного программирования СПбГУ был создан проект QRealWeb. Данный проект ставит своей целью реализацию онлайн-сервиса, ориентированного на непрофессиональных пользователей, который позволил бы создавать приложения со сложной логикой.

Позиционирование QRealWeb как онлайн-сервиса мотивируется тем, что такой подход позволяет избавить пользователей от необходимости устанавливать какой-либо клиент на свой компьютер, а интернет-обозреватель присутствует на абсолютном большинстве пользовательских компьютеров. Помимо позиционирования QRealWeb как веб-сервиса, было принято решение предоставлять сервис как облачное приложение. Использование облачных технологий способно упростить поддержку сервиса.

Конечному пользователю сервис QRealWeb предоставляет следующий набор инструментов.

- Редактор интерфейса и клиентской логики приложения.

  Редактор интерфейса позволяет управлять экранами приложения, добавлять элементы управления на отдельные экраны, редактировать их свойства.

- Онлайн-эмулятор приложения.

  Онлайн-эмулятор позволяет запускать создаваемое приложение прямо в интернет-обозревателе. Это упрощает отладку приложе-

ния, снимая необходимость каждый раз генерировать приложение под нужную платформу и запускать его на реальном устройстве.

- Набор генераторов приложений.

  Генераторы служат для создания по заданному пользователем интерфейсу и логике приложения, пригодного для установки на устройство или расположение его в магазине приложений.

В рамках проекта также реализуется возможность коллаборативной работы нескольких пользователей над одним приложением.

1.2. Анализ требований

В данной части работы более детально будут описаны требования, предъявляемые к редактору.

- Отсутствие требования от пользователя навыков программирования.

  Это требование означает, что задание логики приложение должно происходить без написания пользователем программного кода. Требуется предусмотреть использование более наглядных средств, например, визуальных языков или компоновки элементарных блоков.

- Кроссбраузерность.

  На данный момент существует множество различных интернет-обозревателей. Они по-разному способны исполнять JavaScript[5]-код, по-разному отображать разметку страницы. Таким образом, требуется одинаковая логика исполнения и одинаковое отображение редактора в различных интернет-обозревателях.

- Возможность задания интерфейса приложения.

[5]JavaScript - Wikipedia, the free encyclopedia. - URL: http://en.wikipedia.org/wiki/JavaScript . - Дата обращения: 20.05.2013.

Требуется предоставить возможность пользователю управлять экранами приложения, добавлять на отдельные экраны элементы управления, редактировать свойства элементов управления. Для этого нужно предоставить пользователю палитру с элементами управления, редактор свойств и менеджер экранов.

- Поддержка интеграции с серверной частью веб-сервиса и онлайн-эмулятором.

  Редактор должен поддерживать сериализацию разрабатываемого пользователем приложения в формат, пригодный для взаимодействия с прочими частями веб-сервиса.

1.3. Обзор существующих решений

На данный момент на рынке присутствуют онлайн-сервисы для разработки под мобильные платформы. К таким сервисам можно отнести iBuildApp[6], AppMakr[7], Apps.ru[8] и т.д. Эти сервисы позволяют без применения программирования создавать простые приложения для мобильных платформ. Большинство подобных сервисов ориентируются на платформы Android и iOS.

Данные сервисы позволяют пользователю задавать интерфейс приложения при помощи предоставляемых дизайнеров форм. Кроме того, предоставляется возможность задания перехода между формами. При помощи специальных элементов интерфейса есть возможность отображать на экране приложения веб-страницу с какого-либо внешнего интернет-ресурса.

Описанные выше сервисы позволяют широкому кругу людей создавать простые приложения, так как не требуют навыков программирования. Но стоит заметить, что многие из рассмотренных сервисов

[6]iBuildApp - создайте приложение на базе iPhone и Android, Бесплатно, Без навыков кодирования. - URL: http://russia.ibuildapp.com . - Дата обращения: 20.05.2013.

[7]AppMakr :: iPhone App Maker | Make your own iPhone App | Free iPhone App Maker. - URL: http://www.appmakr.com . - Дата обращения: 20.05.2013.

[8]My-apps.com - Make your own mobile application for your business in 5 minutes. - URL: http://my-apps.com . Дата обращения: 20.05.2013.

реализуют создание приложений исключительно для како-либо одной мобильной платформы.

С другой стороны, подобные сервисы обладают одним общим недостатком: они не позволяют пользователю создать приложение со сложной логикой. Таким образом функциональность указанных сервисов ограничивается созданием информационных приложений, которые не обладают интерактивностью и способностью к взаимодействию с внешними ресурсами, за исключением отображения на экране веб-страницы. Отсутствие выделенного редактора логики не позволяет реализовать какую-либо обработку данных на клиентской и серверной сторонах приложения.

Большинство из подобных сервисов являются проприетарными: с исходными кодами ознакомиться нельзя. Использование подобных сервисов в рамках проекта QRealWeb оказалось невозможным не только в связи с отсутствием возможности задания логики, но и в связи с лицензионными соображениями.

Таким образом, обнаружить какое-либо готовое средство, решающее задачи моей работы, не удалось.

1.4. Обзор используемых инструментов

Разработка требуемого приложения с использованием исключительно языка JavaScript[3] является довольно трудоёмким процессом. JavaScript не обладает статической типизацией, его объектная модель не является привычной для многих разработчиков. Отладка кода представляет собой трудный процесс. Кроме того, различные интернет-обозреватели могут по-разному интерпретировать JavaScript-код. Для борьбы с этим приходится писать отдельный код для различных браузеров. Таким образом, размер кода возрастает, а функциональность нет. В итоге приходится основную часть времени тратить на обработку особенностей выполнения кода различными движками, а не на реализацию продукта. Частично последнюю проблему можно решить использованием специальных библиотек, но проблемы с отсутствием типизации и объектной

моделью остаются.

Для реализации редактора было рассмотренно несколько инструментов и библиотек для веб-разработки: WebSharper[9], TypeScript[10], JQuery[11].

1.4.1. WebSharper

WebSharper является веб-фреймворком, разработанным компанией Intellifactory[12]. Основной идеей фреймворка является написание клиентской и серверной частей приложения на языке F#[13]. Участки кода, помеченные атрибутом, указывающим на принадлежность к клиентской части, транслируются в JavaScript и исполняются на клиентской машине в интернет-обозревателе.

WebSharper поддерживает интеграцию со средой разработки Microsoft Visual Studio[14]. Существуют расширения для поддержки множества популярных JavaScript-библиотек.

Недостатками данного фреймворка являются отсутствие сообщества разработчиков и хорошей обратной связи от разработчиков. Помимо этого, WebSharper более ориентирован на разработку серверной части приложения, разработка же клиентской части по-прежнему остаётся довольно сложным процессом.

1.4.2. TypeScript

TypeScript представляет из себя язык программирования, созданный компанией Microsoft[15]. Данный язык является "синтаксическим сахаром" над JavaScript, для исполнения на клиентской машине транслируется в последний.

Особенностями языка являются наличие статической типизации и более традиционная объектная модель. Изначально этот язык ставил

[9]WebSharper home. - URL: http://www.websharper.com . - Дата обращения: 20.05.2013.

[10]Welcome to TypeScript. - URL: http://www.typescriptlang.org . - Дата обращения: 20.05.2013.

[11]jQuery. - URL: http://jquery.com . - Дата обращения: 20.05.2013.

[12]IntelliFactory Home. - URL: http://www.intellifactory.com . - Дата обращения: 20.05.2013.

[13]The F# Software Foundation. - URL: http://fsharp.org . - Дата обращения: 20.05.2013.

[14]Visual Studio. - URL: http://www.microsoft.com/visualstudio . - Дата обращения: 20.03.2013.

[15]Microsoft Home Page | Devices and Services. - URL: http://www.microsoft.com . - Дата обращения: 20.05.2013.

своей целью предоставить инструмент разработки, расширяющий JavaScript и устраняющий некоторые его недостатки.

Существует плагин для Microsoft Visual Studio, позволяющий вести в этой среде разработку с использованием TypeScript. Помимо плагина для среды разработки, релизованы привязки к многим JavaScript-библиотекам, в том числе JQuery и JQuery Mobile[16].

Исходные коды компилятора распространяются по свободной лицензии Apache License[17].

1.4.3. JQuery

JQuery является популярной JavaScript-библиотекой. Основные преимущества её использования:

- простая реализация AJAX[18];
- удобное взаимодействие с объектной моделью документа;
- решение проблем совместимости между браузерами.

Так как редактор будет использоваться в том числе и для задания интерфейса мобильного приложения, то требуется реализация основных элементов интерфейса мобильных приложений. Реализация каждого элемента в отдельности - процесс, требующий времени и дизайнерских навыков. Поэтому помимо JQuery в данной предметной области разумно рассматривать библиотеку JQuery Mobile, которая содержит множество элементов интерфейса мобильных приложений и обычно используется для разработки веб-приложений, отображаемых на мобильных устройствах.

[16]jQuery Mobile | jQuery Mobile. - URL: http://jquerymobile.com . - Дата обращения: 20.05.2013.

[17]Apache License, Version 2.0. - URL: http://www.apache.org/licenses/LICENSE-2.0.html . - Дата обращения: 20.05.2013.

[18]Ajax (programming) - Wikipedia, the free encyclopedia. - URL: http://en.wikipedia.org/wiki/AJAX . - Дата обращения: 20.05.2013.

1.4.4. Выбранные инструменты разработки

Основываясь на соображениях, описанных в предыдущих пунктах, для разработки редактора интерфейса и логики были выбраны следующие средства разработки:

- TypeScript[2];
- JQuery[1];
- JQuery Mobile.

В качестве среды разработки использовался продукт Microsoft Visual Studio. Для управления исходными текстами продукта использовалась система контроля версий git[19] и сервис GitHub[20].

[19]Git. - URL: http://git-scm.com . - Дата обращения: 20.05.2013.

[20]GitHub · Build software better, together. - URL: https://github.com . - Дата обращения: 20.05.2013.

2. Архитектура сервиса

Общая архитектура сервиса состоит из нескольких заменяемых компонент. [4] Общая структура сервиса представлена на рисунке 1.

Как видно на рисунке, сервис состоит из трёх основных компонент. Кратко опишем каждую из них.

- Редактор интерфейса и логики.

  Данный модуль служит для задания внешнего вида приложения путём перетаскивания из палитры инструментов элементов интерфейса на экран и задания логики приложения. Для управления свойствами элементов интерфейса присутствует редактор свойств. Задание логики приложения ведётся при помощи компоновки элементарных блоков. Для интеграции с прочими компонентами системы поддерживается сериализация создаваемого приложения.

  Реализация выполнена с использованием языка TypeScript и библиотек JQuery, JQueryMobile.

- Онлайн-эмулятор.

  Позволяет запустить создаваемое в редакторе приложение без вызова генератора прямо на странице сервиса. Применяется для упрощения отладки интерфейса и логики приложения. Как и полноценное сгенерированное приложение, поддерживает взаимодействие с внешними ресурсами.

  Как и в случае с редактором, реализация выполнена с использованием языка TypeScript и библиотек JQuery, JQueryMobile в рамках курсовой работы Бумакова Н.

- Сервер.

  Обеспечивает взаимодействие клиентской части сервиса (редактора и онлайн-эмулятора) с набором генераторов приложений и с репозиторием. Также отвечает за авторизацию пользователей сервиса.

Сервер реализован на платформе .NET[21] при помощи языка F# и фреймворка WebSharper в рамках дипломной работы Чижовой Н.

- Набор генераторов.

  Генераторы служат для построения по созданной в редакторе модели готового мобильного приложения, которое может быть установлено на смартфон или размещено в магазине приложений. На данный момент реализованы генераторы под платформы Windows Phone[22] и Android.

  Данная часть сервиса реализована при помощи языка F#, используемого для генерации кода под платформы Windows Phone и Android, в рамках курсовой работы Захарова В.

Взаимодействие между компонентами сервиса осуществляется при помощи сериализованной в XML[23] модели приложения.

[21].NET Downloads, Developer Resources & Case Studies | Microsoft .NET Framework. - URL: http://www.microsoft.com/net . - Дата обращения: 20.05.2013.

[22]The Smartphone Reinvented Around You | Windows Phone. - URL: http://www.windowsphone.com . - Дата обращения: 20.05.2013.

[23]XML - Wikipedia, the free encyclopedia. - URL: http://en.wikipedia.org/wiki/XML . - Дата обращения: 20.05.2013.

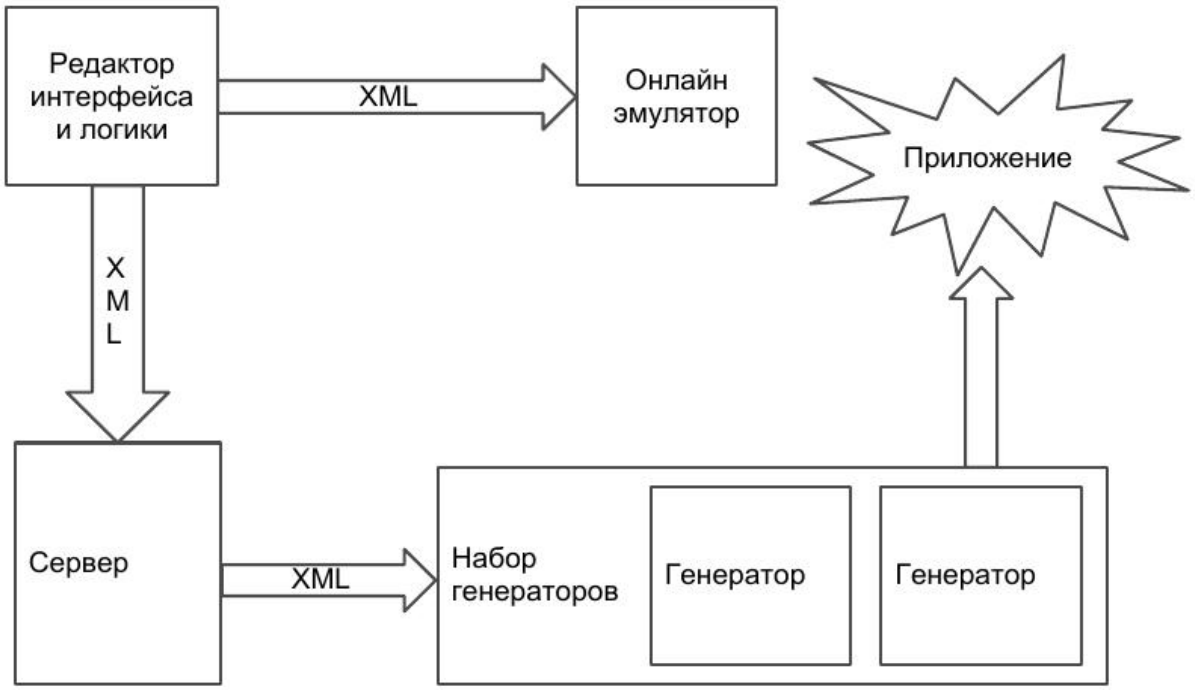

Рис. 1: Общая структура сервиса

3. Редактор интерфейса и логики приложения

Архитектура редактора представлена на рисунке 2. Редактор состоит из следующих компонент:

- набор экранов приложения;
- менеджер экранов;
- палитра элементов;
- редактор свойств элементов интерфейса;
- редактор событий;
- сериализатор.

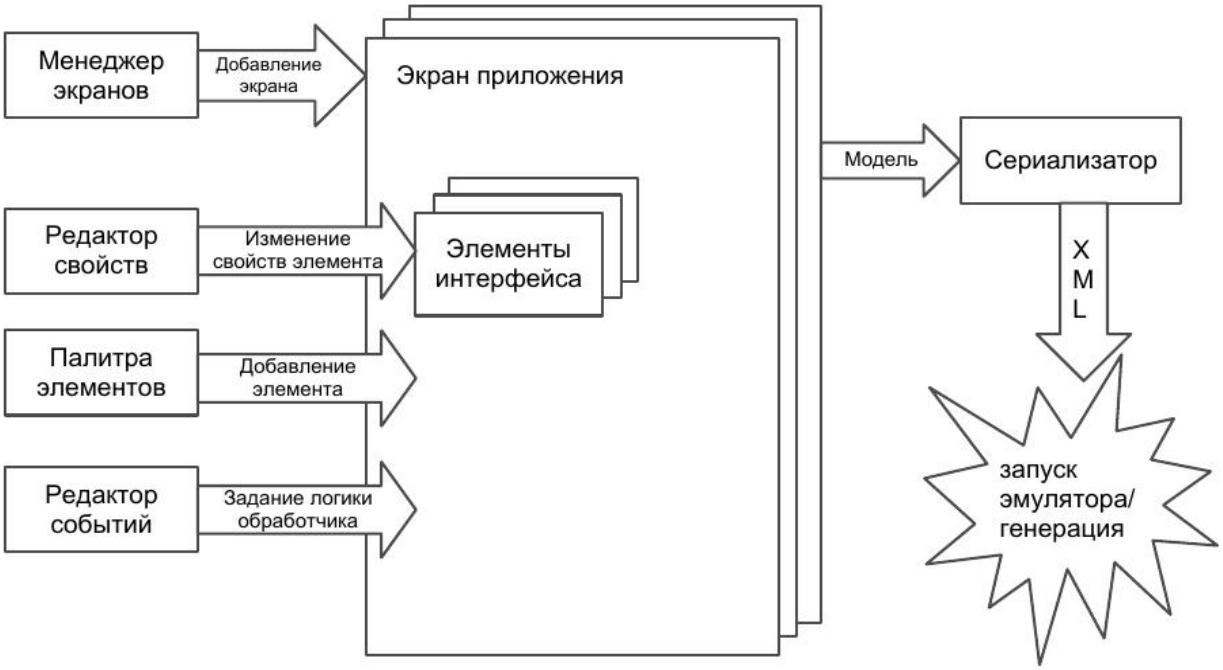

Рис. 2: Архитектура редактора

3.1. Управление экранами приложения

Экран приложения является основной рабочей областью при задании интерфейса приложения. Именно на него добавляются элементы интерфейса. У каждого приложения может быть несколько экранов. В процессе работы приложение может переключать активный экран. К примеру, в типовом приложении с авторизацией может быть выделенный экран с формой авторизации, экран с сообщением об ошибке авторизации и прочие экраны, которые могут отображаться в случае успешной авторизации.

Кроме того, каждому экрану соответствует набор событий, для которых можно задать логику обработки. В том числе таким образом может быть задано переключение экранов приложения.

За управление экранами в редакторе отвечает менеджер экранов. Он позволяет добавлять к приложению новые экраны, переключать текущий редактируемый экран и задавать имя каждого экрана приложения. Один из экранов должен иметь имя "main". Именно он является экраном, который показывается первым при запуске приложения в эму-

ляторе или на реальном устройстве.

3.2. Добавление элементов интерфейса

За возможность добавления элементов интерфейса на экран приложения отвечает палитра элементов. Добавление осуществляется путём перетаскивания нужного элемента интерфейса на текущий редактируемый экран.

В палитре элементов реализованы описанные ниже элементы интерфейса.

- Элемент интерфейса Button.

  Является кнопкой, одним из основных элементов управления приложения. К каждой кнопке привязан обработчик, который позволяет задать логику обработки её нажатия. В том числе, по нажатию на кнопку можно осуществить переход на другой экран приложения.

- Элемент интерфейса TextView.

  Служит для отображения текстовой информации на экране приложения. Сам текст задаётся при помощи редактора свойств.

- Элемент интерфейса ImageView.

  Отображает картинку на экране приложения по заданному URL[24].

- Элемент интерфейса WebView.

  Служит для отображения на экране приложения интернет-ресурса по указанному URL.

- Элемент интерфейса EditText.

  Текстовое поле ввода. Может служить для ввода информации в приложение.

[24]Uniform resource locator - Wikipedia, the free encyclopedia. - URL: http://en.wikipedia.org/wiki/Url . - Дата обращения: 20.05.2013.

- Элемент интерфейса Map.

  Позволяет добавить на экран приложения карту. В качестве карты используется Bing Maps[25].

3.3. Изменение свойств элементов управления

Добавленные элементы интерфейса чаще всего хочется расположить определённым образом или задать им прочие свойства, которые влияют на отображение элемента.

Для этих целей реализован редактор свойств, который позволяет редактировать основные свойства выбранного элемента интерфейса. Выбор элемента на текщем экране осуществляется щелчком мыши.

Часть свойств присутствует у всех элементов интерфейса, например отступ сверху есть у всех элементов. Часть же может быть специфична для какого-то одного элемента интерфейса, как например параметр ”onClick” элемента Button. Некоторыми свойствами обладают несколько типов элементов управления. Например, свойство ”текст” есть у Button и TextView.

3.4. Задание клиентской логики приложения

Для задания логики приложения используются обработчики событий экранов и свойство ”onClick” элемента Button. У каждого экрана приложения есть набор событий, для которых можно задать обработчик. Ниже описан набор событий экрана.

- Событие onShow.

  Данное событие происходит при переключении на соответствующий экран. Для экрана с именем ”main” данное событие происходит и при запуске приложения.

- Событие onTimer.

[25]Bing Maps - Driving Directions, Traffic and Road Conditions. - URL: http://www.bing.com/maps . - Дата обращения: 20.05.2013.

Это событие происходит для текущего экрана с регулярным интервалом времени. Подобное событие полезно использовать для периодического опроса внешнего ресурса или для периодического обновления содержимого текущего экрана.

- Событие onLoginResponse.

  Это событие происходит при получении приложением ответа об авторизации от внешнего источника или сервера приложения. Данное событие может наступить только в том случае, если уже было вызвано действие по авторизации на сервер.

- Событие onDataResponse.

  Это событие происходит при получении приложением ответа на запрос о получении данных от внешнего источника или сервера приложения.

  В дальнейшем подобный вид события будет разбит на несколько типов, которые будут зависеть от типа создаваемого приложения.

В качестве основы для редактора логики был взята среда программирования Scratch[26], которая часто используется для обучения детей программированию. Написание программ в этой среде осуществляется при помощи перетаскивания мышкой элементарных блоков, соответствующих определённым действиям, например условного ветвления и т.д.

Задание логики осуществляется при помощи редактора логики путём компоновки элементарных действий. Редактор логики позволяет добавлять к выбранному обработчику события или свойству "onClick" элемента Button новые действия или удалять уже добавленные.

В рамках выпускной работы были реализованы следующие элементарные действия.

- Действие If.

[26]Scratch - Imagine, Program, Share. - URL: http://scratch.mit.edu . - Дата обращения: 20.05.2013.

Условное ветвление. Используется для задания логики в том случае, если поведение приложения должно зависеть от результата выполнения запроса или от значения какого-либо элемента интерфейса. На данный момент в качестве условия может выступать успешность выполнения запроса об авторизации на сервер. В рамках редактора логики блоки, принадлежащие одной из веток действия "if", выделяются отступом.

- Действие Transition.

  Данное действие задаёт переход на выбранный из списка экран.

- Действие LoginRequest.

  Это действие используется для отправки запроса об авторизации на сервер со значениями логина и пароля, выбранными из соответствующих выпадающих списков, в которых указаны присутствующие на форме элементы типа EditText.

- Действие SaveSession.

  Действие выполняет сохранение активной сессии после успешной авторизации на сервер. Это действие обычно используется в обработчике onLoginResponse.

- Действие DataRequest.

  Выполняет запрос о данных на сервер приложения.

- Действие ShowMap.

  Действие показывает карту, указанную в списке из присутствующих на экране элементов Map. Чаще всего используется в обработчике onDataResponse.

Таким образом, пользователь сервиса для задания логики приложения получает событийно-триггерную систему, позволяющую выполнять элементарные действия и реагировать на наступившие события. При этом часть действий является общей для всех обработчиков, а

часть имеет смысл лишь в контексте выполнения определённого обработчика.

В дальнейшем планируется реализовать задание и клиентской, и серверной логики приложения при помощи специальных графических языков.

3.5. Интеграция с другими компонентами сервиса

За интеграцию с другими компонентами сервиса отвечает сериализатор. Он обходит созданную модель приложения, создавая при этом XML с описанием интерфейса приложения и его логики. В таком сериализованном виде модель создаваемого приложения может быть принята эмулятором для исполнения или же передана на сервер для генерации приложения или сохранения. При этом стоит заметить, что формат XML одинаков для всего сервиса, что позволяет избежать излишней обработки при передаче его между различными компонентами сервиса. За счёт подобного единообразия удалось достичь модульности сервиса, а это в свою очередь позволяет легко заменять один компонент другим или же добавлять новые компоненты.

В сериализованном виде в виде узлов описаны экраны приложения, добавленные на них элементы управления, а также обработчики событий. Свойства элементов управления указываются в атрибутах узлов. При этом важна иерархия узлов: детьми корневого узла являются узел, описывающий логику приложения, и узел, описывающий интерфейс приложения. При этом при описании логики приложения важен вопрос о сохранении порядка выполнения элементарных действий. Поэтому в том случае, если обработчик состоит из нескольких подряд выполняющихся действий, отвечающие им узлы оборачиваются вложенными узлами, где явным образом указана очередность выполнения.

Из окна редактора возможно вызвать эмулятор или же сгенерировать приложение. В каждом из этих случаев, приложение сериализуется и в таком виде передаётся нужному модулю системы.

После отправки сериализованного приложения на сервер для гене-

рации, клиент получает файл с приложением для нужной платформы. Подобный файл можно установить на мобильное устройство или же подать заявку на размещение приложения в магазине. Кроме того, на данный момент реализовано автосохранение разрабатываемого приложения на сервере. Сохранение происходит каждый раз при добавлении нового элемента управления, нового экрана, изменении свойства элемента управления или изменении в обработчике.

3.6. Описание интерфейса редактора

Интерфейс созданного редактора представлен на рисунке 3.

В верхней части располагаются кнопки, вызывающие онлайн-эмулятор и генерацию приложения.

Посередине располагается основная рабочая область, где отображается текущий редактируемый экран и добавленные на него элементы. В данном случае добавлены два TextView, два EditText и один элемент Button.

Слева от основной рабочей области располагаются менеджер экранов и палитра элементов. В менеджере экранов выбран экран с именем ”main”. На палитре располагаются описанные выше элементы управления, которые могут быть добавлены на экран.

Справа от основной рабочей области находятся редактор свойств и редактор логики. Редактор свойств пуст, так как на данный момент на экране не выбран ни один элемент управления. В редакторе логики выбрано событие ”onShow”, при наступлении которого указано поведение, включающее в себя попытку авториазации на сервер приложения, условное ветвление и переход на другой экран.

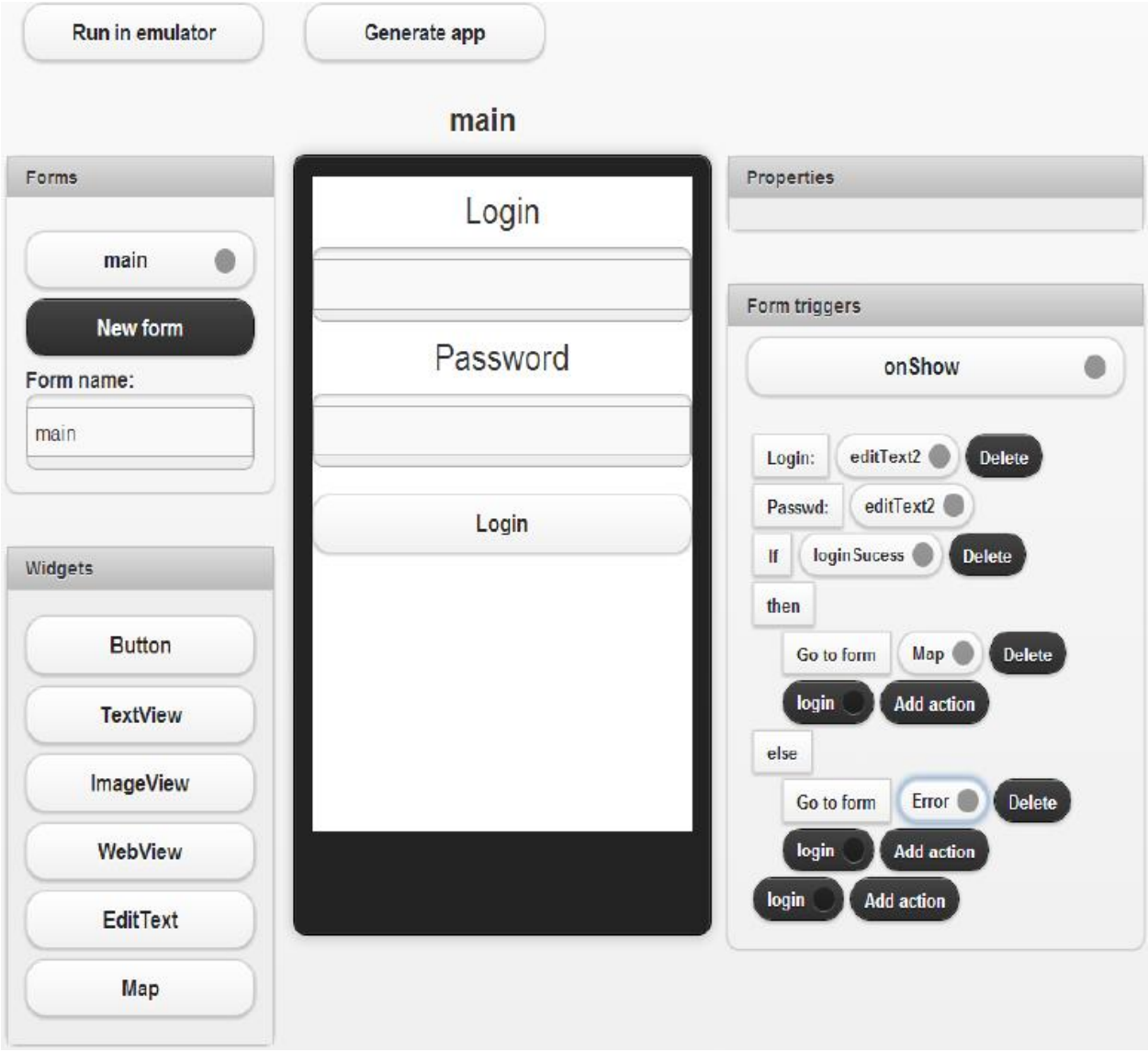

Рис. 3: Интерфейс редактора

4. Апробация

Апробация заключается в реализации при помощи созданного редактора и остальных модулей сервиса реального приложения, которое обладает нетривиальной логикой и имеет какую-либо практическую пользу. Таким образом необходимо проверить пригодность сервиса для реализации востребованных на рынке приложений.

Первым шагом в апробации было создание приложения-визитки. Визитка - чисто информационное приложение, которое, как правило, состоит из небольшого числа экранов, на которых располагается информация о какой-либо компании и предоставляемых ей услугах. Пере-

ходы между экранами задаются при помощи кнопок, которым в редакторе соответствуют элементы интерфейса Button. О более нетривиальной логике в контексте подобного приложения говорить не приходится: вся логика ограничивается заданием переходов между экранами.

При помощи созданного сервиса удалось создать подобное приложение, но это никак не доказывало практическую ценность данной работы, так как многие онлайн-сервисы предоставляют клиентам возможность для создания подобных приложений. Таким образом, необходимо описать более сложное приложение для апробации.

Примером такого приложения, которое имеет практическую ценность и обладает логикой сложнее задания переходов между экранами, является приложение для врачей.

4.1. Приложение для врачей

Представим себе участкового врача, который совершает обход пациентов на своём участке. На первый взгляд, в этом нет ничего нетривиального. Но на самом деле подобная деятельность связана с определёнными сложностями: врач узнаёт о появлении нового пациента звонком из диспетчерской, необходимо записать информацию с нужным адресом. Кроме того, пациенты могут быть сильно территориально разбросаны. Таким образом, врачу будет сложно обойти их всех с минимальными перемещениями по городу. К тому же ориентация в городе тоже является проблемой во многих случаях.

Одним из вариантов решения проблемы является создание приложения, которое для каждого врача показывало бы на карте положение пациентов.

4.1.1. Описание приложения

Одним из экранов приложения должен быть экран с картой, на которой показываются пациенты. Нужно учесть, что подобным сервисом могут одновременно пользоваться несколько врачей. Сервис должен отличать их между собой и отображать на карте лишь тех пациентов,

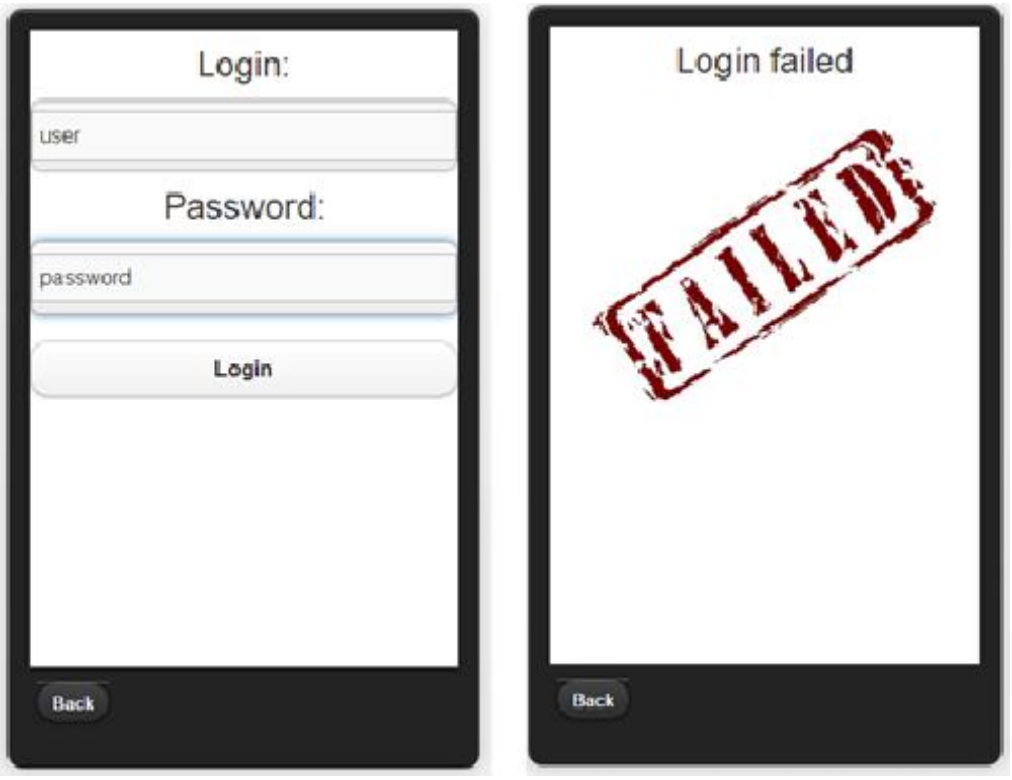

Рис. 4: Экран авторизации и экран сообщения об ошибке

которые находятся в зоне ответственности конкретного врача. Таким образом, необходима авторизация.

Авторизация осуществляется при помощи экрана, на котором вводятся логин и пароль врача, и экрана, который отображается при неудачной авторизации.

Итого мы имеем три экрана: экран с картой, экран авторизации и экран с сообщением об ошибке авторизации.

Интерфейс разрабатываемого приложения показан на рисунках 4 и 5.

По мере появления новых пациентов, их положения должны показываться на карте, таким образом, приложение должно периодически опрашивать сервер приложения о появлении новых пациентов.

4.1.2. Релизация приложения для врачей

При реализации приложения для врачей с требуемым функционалом сначала создаются три экрана. При этом экран с именем ”main”

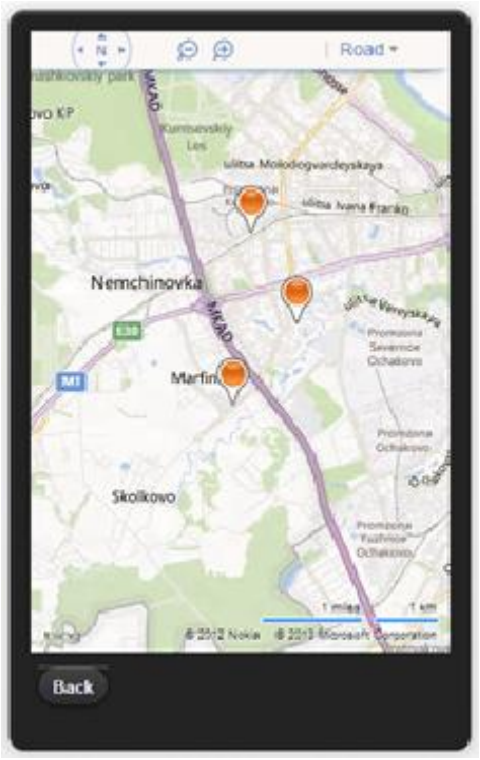

Рис. 5: Экран с картой

соответствует экрану авторизации.

После добавления нужных экранов на них добавляются соответствующие элементы интерфейса: для экрана авторизации это TextView, EditText и Button; для экрана с сообщением об ошибке авторизации - TextView и ImageView; для экрана с картой - Map. При помощи редактора свойств добавляется необходимый текст для элементов типов TextView и Button, задаётся адрес изображения для элемента ImageView. Кроме того, указывается положение элементов на экранах при помощи выставления верхнего отступа.

Задание интерфейса приложения на этом заканчивается. Следующим шагом при реализации приложения является задание логики.

Для задания логики авторизации нужно рассмотреть свойство "onClick" элемента Button. В обработчике события указывается действие "Login", из списков выбираются два элемента типа EditText, которые соответствуют полям логина и пароля соответственно. Затем необходимо указать действие в обработчике "onLoginResponse" текущего экрана. В него нужно добавить действие If, выбрать из списка значение LoginSuccess.

В ветке, соответствующей успеху авторизации указывается переход на экран с картой, а в ветке, соответствующей ошибке авторизации - переход на экран с сообщением об ошибке.

Далее необходимо реализовать отображение положения пациентов на карте и периодический опрос о появлении новых пациентов. Для реализации подобного специфичного для этого приложения функционала были добавлены событие "onPatientsResponse" и действие "patientsRequest". В дальнейшем планируется получать список возможных событий и действий с application-сервера конкретного приложения. Чтоб реализовать отображение пациентов на карте и опрос о появлении новых пациентов нужно перейти на экран с картой и в обработчиках событий onShow и onTimer добавить действие "patientsRequest". В обработчике события onPatientsResponse указывается действие "showMap" с выбранной из списка картой.

На этом создание приложения закончено. Оно успешно запускается в эмуляторе и успешно генерируется на сервере. При этом в результате получается работоспособное приложение, которое успешно работает на реальном устройстве.

Сервер приложения при этом на данный момент реализуется вручную путём написания кода.

4.2. Результат апробации

В результате апробации было выяснено, что при помощи созданного сервиса можно создавать нетривиальные приложения, которые имеют практическую пользу. В дальнейшем необходимо предусмотреть задание серверной логики в редакторе.

Заключение

В результате выполнения данной работы:

- произведён анализ предметной области и существующих решений;
- сформулированы требования к редактору;
- реализован редактор интерфейса и логики, удовлетворяющий поставленным требованиям;
- реализована сериализация модели приложения для интеграции с эмулятором и серверной частью;
- приложение апробировано на примере приложения для врачей.

Список литературы

[1] Bibeault Bear. jQuery in Action, Second Edition. — Manning Publications, 2010. — 475 p.

[2] Maharry Dan. TypeScript Revealed. — Apress, 2013. — 104 p.

[3] Zakas Nicholas C. High Performance JavaScript (Build Faster Web Application Interfaces). — O'Reilly Media, 2010. — 242 p.

[4] Белокуров Д.Н. Бумаков Н.В. Захаров В.А. Чижова Н.А.. Разработка визуального конструктора мобильных приложений // Список-2013: Материалы всероссийской научной конференции по проблемам информатики. — Готовится к публикации.