Санкт-Петербургский Государственный университет

Математико-механический факультет

Кафедра системного программирования

Разработка эмулятора мобильных устройств в браузере

Курсовая работа студента 445 группы

Бумакова Никиты Вячеславовича

Научный руководитель ……. Тимофей Брыксин

Санкт-Петербург
2013

Оглавление

Введение

На текущий момент в мире наблюдается рост рынка мобильных устройств. Особенно рост рынка связан с ростом продаж более сложных устройств, прежде всего, смартфонов и планшетных компьютеров, в то время как продажи обычных сотовых телефонов сократились по сравнению аналогичным периодом прошлого года. В связи с этим растет рынок мобильных приложений.

Серьезным фактором, ограничивающим их развитие, является высокая сложность разработки. В связи с этим появилась идея создания конструктора мобильных приложений, для упрощения их разработки. С его помощью можно создавать приложения, не имея навыков программирования. Такая технология востребована как у обычных пользователей смартфонов, нуждающихся в уникальных приложениях, так и у программистов, позволяя быстрее создавать приложения.

В рамках проекта QReal:Web создается браузерная среда разработки мобильных приложений. В этой среде разработки пользователь имеет возможность спроектировать приложение, протестировать его на эмуляторе и собрать пакет для конкретной платформы. Целью данной работы является разработка части системы - эмулятора приложений.

1. Обзор существующих решений

Рынок онлайн-сервисов, позволяющих создавать мобильные приложения, уже существует. Наиболее известными среди них являются такие сервисы, как iBuildApp, App Inventor, Apps.ru и т.д. Подходы к созданию мобильных приложений отличаются у разных сервисов, но в основном такие сервисы предоставляют инструмент для задания внешнего вида приложений и простейшей логики (переходы между экранами и т.п.).

Такой подход позволяет создавать за небольшое время красивые, но малофункциональные приложения. Нет возможности создания приложений с нетривиальной логикой, например: работа с внешними сервисами (авторизация, агрегация данных и т.п.), продвинутое взаимодействие с внутренними сервисами телефона (gps, accelerometer и т.д.). Поэтому большинство таких сервисов позволяют создавать красивые информационные приложения, но не способные на взаимодействие с внешними сервисами, либо с функционалом телефона.

Сейчас существует несколько мобильных операционных систем, а многие сервисы предоставляют возможность создания приложений под какую-то одну конкретную платформу, что затрудняет задачу написания приложений под различные мобильные платформы.

Рассмотрим наиболее известные сервисы подробнее.

1.1 iBuildApp

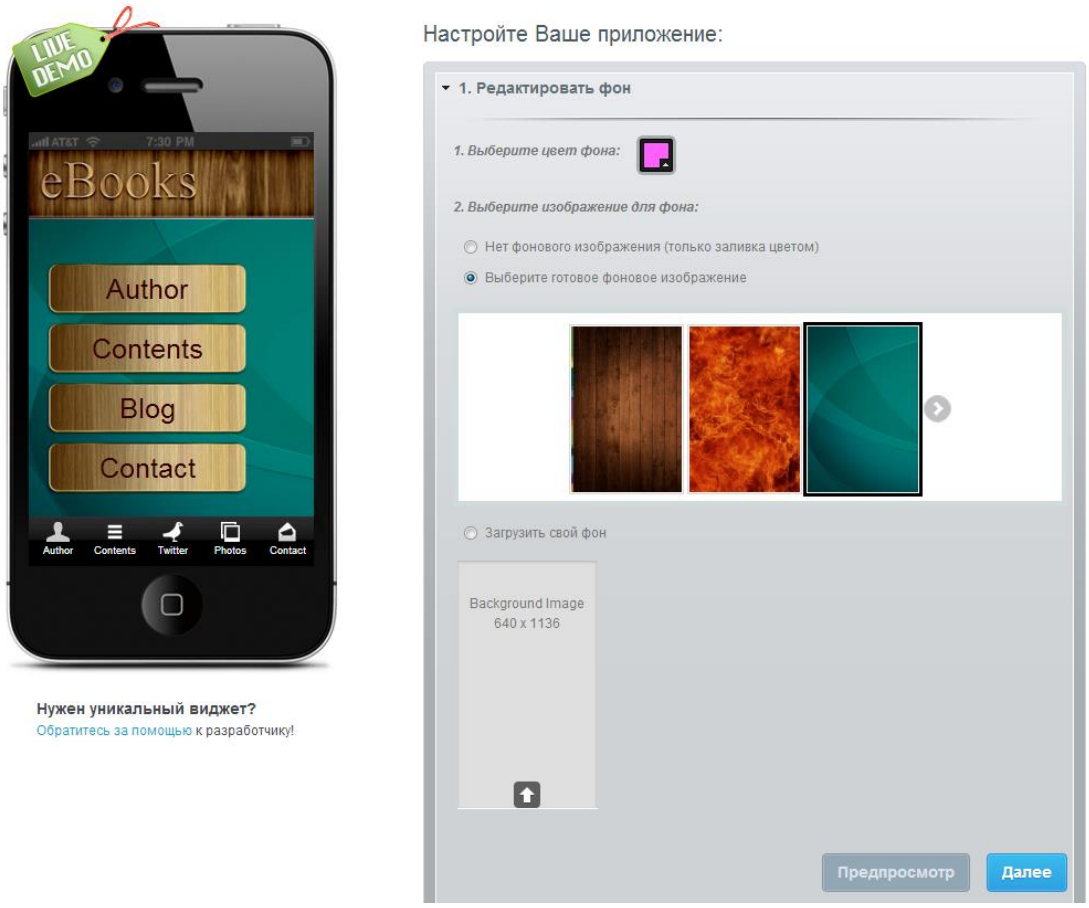

Рис 1. iBuildApp

iBuildApp[1] - это конструктор мобильных приложений под платформы Android и iOS. Отличительной особенностью данного сервиса является то, что он позволяет создавать приложения на основе шаблонов и заранее написанных виджетов. Так существуют элементы для показа rss, проигрывания аудиопотока, просмотра фотографий, общения в соц. сетях и многое другое. Такой подход позволяет решать многие задачи, но не позволяет создавать приложения за рамками существующих шаблонов, без непосредственного программирования.

[1] http://russia.ibuildapp.com/

1.2 AppInventor

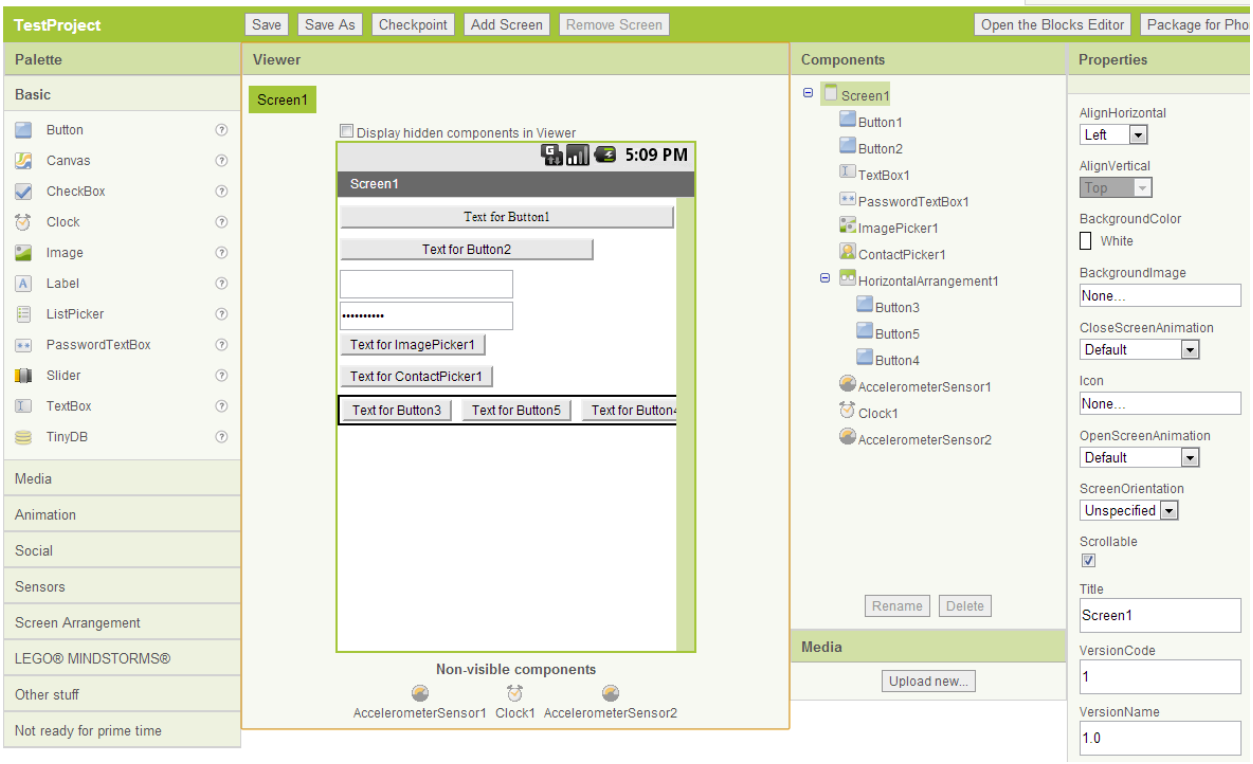

Рис. 2 AppInventor

AppInventor[2] – среда разработки мобильных приложений под платформу Android, разработанная в Google Labs. В среде для задания визуального представления приложения используется классический интерфейс с палитрой элементов. Кроме задания внешнего вида приложений среда позволяет задавать логику, для этого используется визуальный язык программирования подобный языку Scratch[3].

Среда разработки состоит из двух частей: визуального редактора и редактора логики, причем редактор логики – это отдельное приложение, скачиваемое на компьютер разработчика. Такой подход вносит свои трудности в разработку приложений.

[2] http://appinventor.mit.edu/
[3] http://scratch.mit.edu/

2. QReal:Web

Несмотря на то, что уже существуют сервисы создания мобильных приложений, они имеют ряд серьезных недостатков, не позволяющих создавать полноценные приложения:

- проектирование приложений только для одной-двух мобильной платформы;
- невозможность создавать приложения с нетривиальной логикой;
- отсутствие возможностей тестирования разрабатываемых приложений;

Для решения описанных проблем на кафедре системного программирования СПБГУ был создан проект QReal:Web. В рамках этого проекта создается онлайн-среда визуальной разработки мобильных приложений, позволяющая создавать приложения с нетривиальной логикой. QReal:Web основан на облачных технологиях – это облегчает пользователем использование ПО, т.к. нет необходимости в скачивании дополнительных программ, а также позволяет разработчикам контролировать доступ пользователей к среде, что избавляет от пиратства и облегчает монетизацию.

Среда состоит из нескольких частей (см. рис. 3):

- визуального дизайнера и дизайнера логики, позволяющего визуально задать внешний вид разрабатываемого мобильного приложения, а также задавать логику поведения приложения;
- эмулятора мобильных приложений, позволяющего протестировать в браузере работу создаваемого приложения;
- генераторов кода в различные мобильные платформы;
- сервера приложений, на котором работают используемые приложением сервисы;

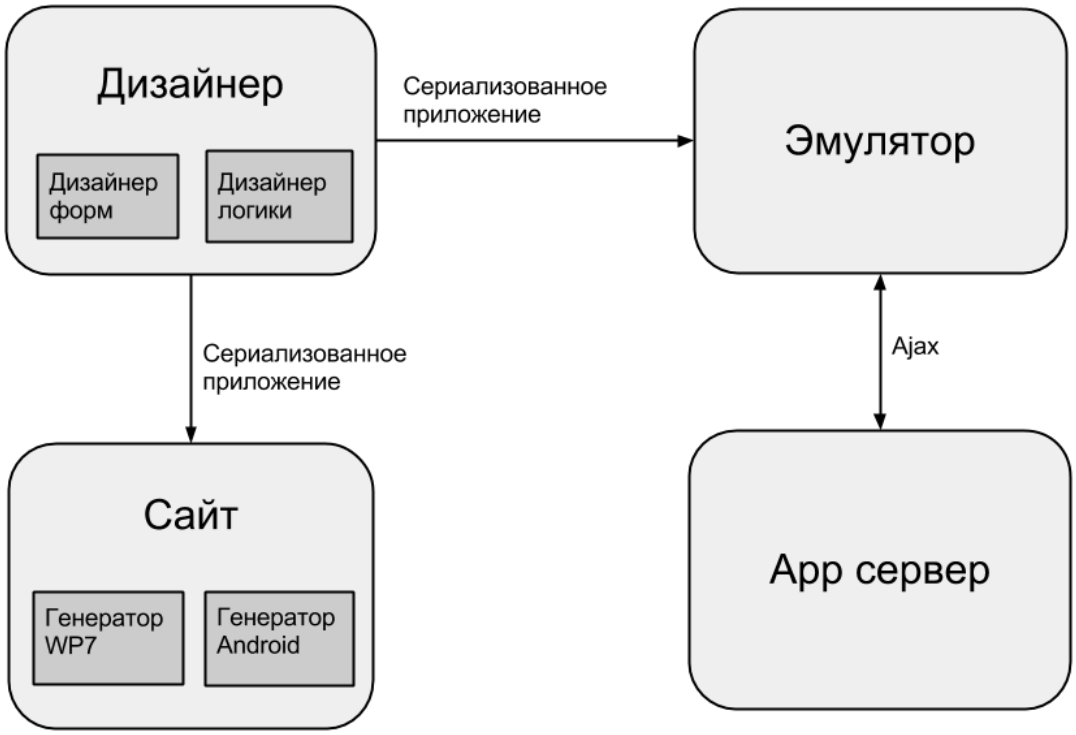

Рис. 4 Архитектура QReal:Web

При помощи визуального дизайнера проектируется приложение, затем оно сериализуется во внутренний формат. Приложение в таком формате можно протестировать на эмуляторе или сгенерировать нативное приложение под конкретную платформу.

3. Постановка задачи

Цель работы — реализация эмулятора мобильного приложения, работающего в браузере. По данным, полученным от визуального дизайнера, должно эмулироваться визуальное представление и логика взаимодействия с внешними сервисами.

Основные требования:

- Генерация визуального представления мобильного приложения.
- Генерация логики.
- Взаимодействие с сервером.
- Работа в браузере.

4. Реализация

4.1. Выбор инструментов разработки

Разработка браузерного приложения возможна на языке JavaScript, но этот язык не слишком удобен для программирования. JavaScript – это прототипно-ориентированный язык с динамической типизацией, что является непривычным для разработчиков, привыкшим к объектно-ориентированному подходу. Также различные браузеры имеют различные интерпретаторы JavaScript, что приводит к различным поведениям приложения в разных браузерах.

Чтобы решить данные проблемы существуют библиотеки, облегчающие кросбраузерную разработку, предоставляя набор функционала, который работает одинаково в различных браузерах. Но это не решает проблем с отсутствием типизации и неудобством прототипной модели. Решением этой проблемы является ряд языков, для которых существуют компиляторы в JavaScript, что позволяет писать исходный код в более традиционном стиле.

Примером таких решений является WebSharper[4], TypeScript[5], JQuery[6].

4.1.1. jQuery

jQuery – это JavaScript-библиотека, позволяющая более гибко взаимодействовать с DOM моделью, предоставляет более удобный API по работе с AJAX, а также решает некоторые проблемы совместимости между браузерами.

Библиотека jQuery имеет обширное сообщество, а также обладает большим набором плагинов, в том числе есть плагин, позволяющий

[4] http://www.websharper.com
[5] http://www.typescriptlang.org
[6] http://jquery.com

создавать графические элементы, схожие с нативными элементами интерфейсов мобильных телефонов – jQueryMobile.

4.1.2. WebSharper

WebSharper – это веб фреймворк, позволяющий создавать веб-приложения на языке программирования F#. При этом серверный код компилируется в байткод платформы .net и выполняется на сервере, а клиентский код компилируется в JavaScript и интерпретируется в браузере.

Разработка кода осуществляется с помощью Microsoft Visual Studio. WebSharper поддерживает ряд популярных JavaScript-библиотек, в том числе и jQuery/jQueryMobile.

Но данный фреймворк имеет существенные недостатки:

- маленькое сообщество разработчиков;
- довольно сложные механизмы для разработки клиентской части.

4.1.3. TypeScript

TypeScript является языком программирования, созданным компанией Microsoft в 2012 году. Разработчики Microsoft’а решили не создавать уникальный язык, а лишь дополнили JavaScript недостающими конструкциями. Таким образом TypeScript – это JavaScript с статической типизацией и традиционной объектно-ориентированной моделью.

Так как TypeScript является расширением языка JavaScript, то любой код на JavaScript является рабочим кодом и на TypeScript. Это позволяет переиспользовать уже написанный код JavaScript, а также свободно использовать любые JavaScript библиотеки Разработка кода также может осуществляется с помощью Microsoft Visual Studio.

4.2. Первая реализация

Первый прототип системы решено было писать с использованием технологии WebSharper. Такой подход позволил бы использовать единственный язык программирования для всей системы – F#. На этой технологии был написан прототип визуального дизайнера, но из-за малого сообщества у WebSharper’а и неработоспособности некоторых плагинов (в частности jQueryMobile) было решено отказаться от этого решения.

4.3. Вторая реализация

Для второй реализации было решено использовать другие технологии. Для написания клиентской части был выбран язык TypeScript, а для отображения графических элементов библиотека jQueryMobile. Также для упрощения работы с dom-элементами и с работой с сервером используется библиотека jQuery. В качестве среды разработки использовался Microsoft Visual Studio.

Для проверки этих технологий также был написан прототип визуального дизайнера, который показал возможность написания системы на этих технологиях.

5. Архитектура эмулятора

В системе приложения представляются в виде xml. По этому формату эмулятор создает визуальное представление приложения и его логику обработки команд пользователя и взаимодействия с внешними сервисами. Таким образом, архитектура эмулятора включает в себя следующие функциональные модули:

- разбора внутреннего представления приложения;
- генерация интерфейса мобильного приложения;
- генерация логики мобильного приложения;
- эмулятор приложения;

5.1. Разбора внутреннего представления приложения

Формат представления приложения – это xml-дерево, корнем которого является тег <application>, внутри которого находятся два тега: <logic> и <forms>. Внутри этих тегов описывается логика поведения приложения и набор его графических элементов. Таким образом, для создания внешнего вида приложения нужно обработать данные внутри тега form, внутри содержатся информация о страницах приложения и об элементах, которые на этих страницах находятся. Данный формат был разработан в рамках дипломной работы Надежды Чижовой.

Логика поведения приложения задается с помощью триггерно-событийной системы. Каждому событию в приложении соответствует соответствующий тег, в котором описывается поведение приложения при наступлении такого события.

5.2. Генерация интерфейса мобильного приложения

Чтобы обеспечить максимальную совместимость элементов графического интерфейса различных мобильных платформ, был создан набор

управляющих элементов, расширяющих стандартные элементы библиотеки jQueryMobile. В случае если в библиотеке есть элементы с требуемой функциональностью, то они используются, иначе создаются свои элементы с требуемой функциональностью.

На данный момент клиент поддерживает следующие элементы:

Элемент интерфейса QReal:Web	Элемент jQueryMobile
Button	Button
EditText	TextField
ImageView	отсутствует
Input	Input
LinearLayout	отсутствует
Map	отсутствует
WebView	отсутствует

При разборе команды данные об элементах отправляются в UI менеджер, в котором формируется дерево. По полученному дереву создается его графическое представление.

5.2. Генерация логики мобильного приложения

Генерация логики приложения происходит по принципу, схожему с генерацией интерфейса – в системе существует набор базовых действий, которые используются для написания приложения. Для каждого такого действия заранее написана их реализация. На данный момент в системе есть поддержка следующих действий:

Действие	Описание действия
If	Условный оператор. Используется для задания логики, если поведение приложение должно зависеть от какого-то условия. В зависимости от условия будет выполнена одна из

	веток
Transition	Действие перехода между экранами приложения
LoginRequest	Отправка запроса авторизации на сервисе. Параметрами этого действия должны быть два поля ввода для логина и пароля соответственно.
SaveSession	Сохранении текущий сессии.
DataRequest	Запрос данных у текущего сервиса.
ShowMap	Показ данных на карте, присланных сервисом.

С помощью комбинации этих действий создаются логика поведения приложения при наступлении события. Примером таких событий может быть нажатие пользователем на кнопку. Сейчас системой поддерживается пять типов событий:

Событие	Описание события
onClick	Нажатие пользователя на кнопку
onShow	Событие, происходящее при переключении на новую страницу приложения.
onTimer	К каждой странице можно добавить таймер, по которому будет происходить заданное действие. Например будут обновляться данные на карте
onLoginResponse	Получение ответа от сервиса авторизации
onDataResponse	Ответ от сервиса на запрос получения данных

5.3. Эмулятор приложения

После того, как сгенерирован внешний вид приложения и функции поведения, данные поступают в менеджер, который управляет текущим состоянием приложения. Пользователь видит в браузере приложение, схожее с нативным мобильным приложением, а также имеет возможность

тестировать в нем как графическое представление, так и взаимодействие с внешними сервисами.

Эмулятор имеет вид, схожий с мобильным телефоном, также реализован back stack, позволяющий эмитировать работу кнопки “back”, которая присутствует практически во всех мобильных платформах.

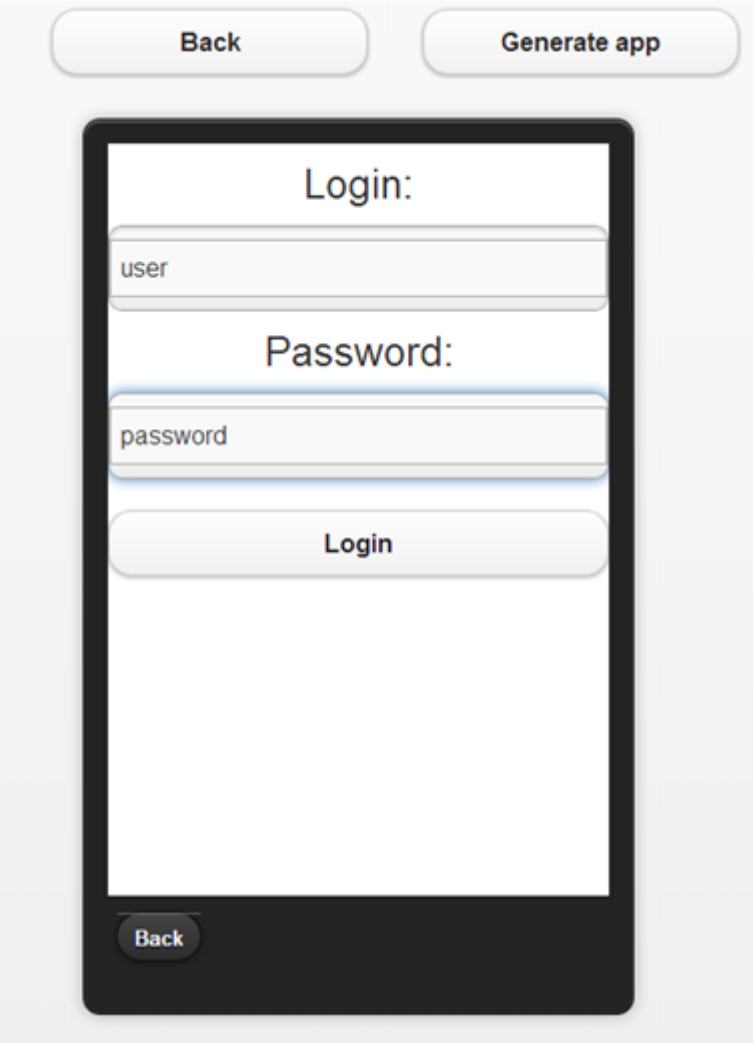

Рис. 5 Эмулятор

6. Апробация

Апробация заключалось в создании приложения с нетривиальной логикой и проверка его работы на эмуляторе. Таким образом проверялось возможность создания таких приложений с помощью QReal:Web.

6.1. Приложение-визитка

Наиболее простым примером создания примером создания приложений стало приложение-визитка. Это приложение, которое содержит информацию о компании, мероприятии и т.п. Обычно такие приложения состоят из небольшого числа экранов, на которых располагается основная информация о событии. Приложение содержит переходы между экранами, осуществляемые по нажатию на кнопки, но более серьезная логика в таких приложениях отсутствует.

С помощью QReal:Web такое приложение создать возможно, но оно не позволяет проверить возможности работы с внешними сервисами и более сложную клиентскую логику.

6.2. Приложение для врачей

Для проверки возможности создания и тестирования приложений с более сложной логикой было создан прототип приложение для врачей, которое бы позволяло участковому врачу видеть на карте координаты своих пациентов. Данные о пациентах могут храниться на сервере в поликлинике, а врач мог видеть актуальную информацию о них на своем телефоне.

Такое приложение может состоять из двух частей: экрана авторизации на сервисе, и экран с данными о пациентах. На рис. 5 показано, как выглядит экран авторизации на эмуляторе.

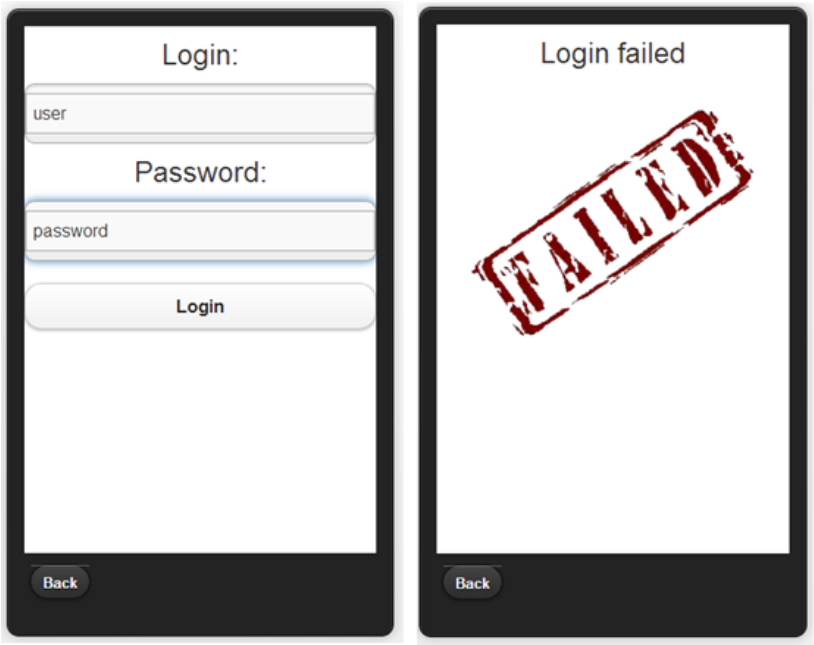

Рис. 6 Эмулятор с экранами авторизации и сообщения об ошибке

Другим экраном приложения является экран с данными о пациентах. В нашем случае это карта, на которой отмечены положения квартир пациентов. По мере появления новых пациентов, их положения должны показываться на карте. Таким образом, приложение должно опрашивать сервер о появлении новых пациентов. На рис. 6 показан эмулятор с картой, отображающей положения пациентов.

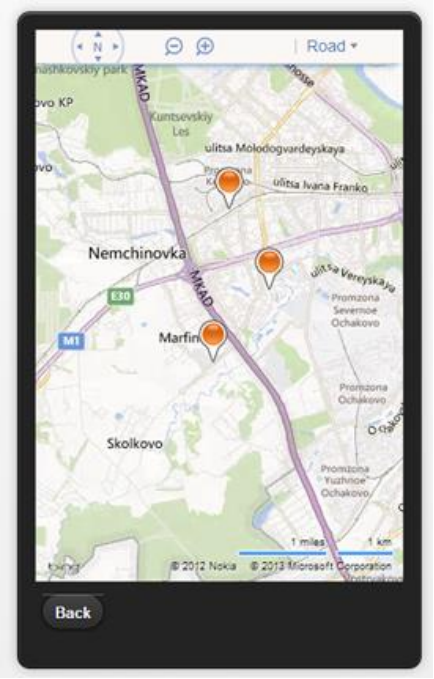

Рис. 7 Экран с данными о пациентах

6.3. Результат апробации

С помощью этих приложений была проверена возможность создания и тестирования приложений с нетривиальной логикой. А также проверено взаимодействие эмулятора с внешними сервисами.

Заключение

В рамках данной работы:

- изучена возможности визуального создания и тестирования мобильных приложений
- изучены возможности создания клиентских веб-приложений
- произведён анализ существующих решений
- реализован эмулятор, позволяющий тестировать создаваемые приложения в браузере
- 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.