Санкт-Петербургский Государственный университет

Математико-механический факультет

Кафедра системного программирования

Терминальный Android-клиент для распределенных приложений на базе платформы Ubiq Mobile

Курсовая работа студента 345 группы

Бумакова Никиты Вячеславовича

Научный руководитель ……. В.В. Оносовский

Санкт-Петербург
2012

Оглавление

1. Введение 3
2. Обзор платформы Ubiq Mobile 4
2.1 Протокол 4
2.2. Модель интерфейса 4
2.1. Basic 4
2.2. NativeBasic 5
3. Постановка задачи 6
4. Архитектура клиента 7
4.1. Соединение с сервером 8
4.2. Модуль-диспетчер 9
4.2.1. Обработка команд сервера 9
4.2.2. Отправка команд на сервер 10
4.2.3. Аутентификация 10
4.3. Разбор дерева графических объектов 11
4.4. Построение дерева нативных элементов 12
4.5. Обработка действий пользователя 13
5. Заключение 14
5.1. Результаты 14
5.2. Перспективы развития 15
5.2.1. Реализация дополнительных управляющих элементов 15
5.2.2 Вызов функций на клиенте 15
Список литературы 16

1. Введение

На данный момент рынок мобильных сервисов находится в стадии роста и создание мобильных онлайн-сервисов является одним из самых активно развивающихся направлений. Серьезным фактором, ограничивающим их развитие, является высокая сложность разработки. Это связано с платформенной фрагментацией, поэтому разработчики вынуждены создавать несколько приложений под каждую из мобильных платформ (платформо-ориентированный подход).

Решением может стать сервисо-ориентированный подход, когда сервис, реализующий данный подход, может состоять из «тонкого» клиента, разработанного под все мобильные платформы, и единого сервера. Самым известным примером системы, реализующей данный подход, является Opera Mini. Opera Mini обрабатывает весь контент через сервера Opera Software, на которых происходит переформатирование веб-страниц в формат подходящий для устроиства.

На факультете в течение последних ряда лет ведется проект по разработке платформы Ubiq Mobile, основанной на сходных принципах. Она предоставляет широкие возможности для создания пользовательских приложений, оптимизацию Интернет-трафика и эффективность при работе в относительно медленных сетях.

Реализация такого похода позволяет приложениям на платформе Ubiq Mobile полноценно конкурировать с нативными мобильными приложениями. Такие приложения существенно выигрывают в универсальности и простоте разработки, при этом обеспечивая близкие к нативным функциональные возможности.

2. Обзор платформы Ubiq Mobile

2.1 Протокол

Используемый в проекте Ubiq протокол связи клиентской и серверных частей платформы относится к группе протоколов – «тэги и структуры». Данные в протоколе передаются в виде команд. Каждая команда протокола представляет собой дерево, элементами которого являются секции. Данные внутри секций организованы в виде последовательностей полей фиксированной или переменной длины. Числа представляются в little endian формате – т.е. вначале идут младшие байты.

Длина команды	Код	ID клиента	ID соединения	Секции

Длина подсекции 1	Код	Данные подсекции 1	Подсекции

Длина подсекции 2	Код	Данные подсекции 2	Подсекции

2.2. Модель интерфейса

2.1. Basic

Модель интерфейса Basic предполагает отрисовку дерева графических объектов на холст сервера, впоследствии они послойно сжимаются и отправляются клиентскому приложению. Такой подход универсален и удобен прежде всего для слабых телефонов, но на для продвинутых смартфонов с развитыми графическими возможностями такое приложение будет выглядеть неестественным, т.к. будут использоваться не «родные» управляющие элементы.

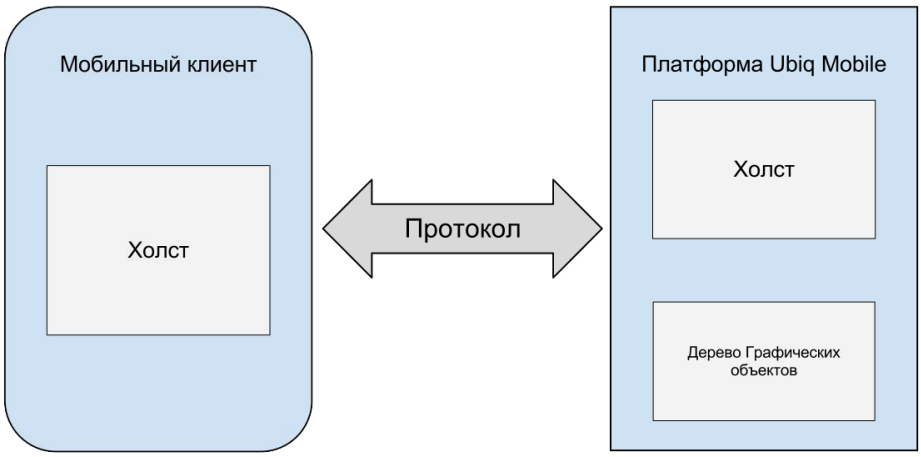


Схема 1. Отрисовка графического дерева с помощью нативного механизма

2.2. NativeBasic

Дерево графических объектов платформы близко к интерфейсам современных мобильных платформ, поэтому естественен подход, при котором на клиент будет передаваться дерево графических объектов, а не их проекция на холст.

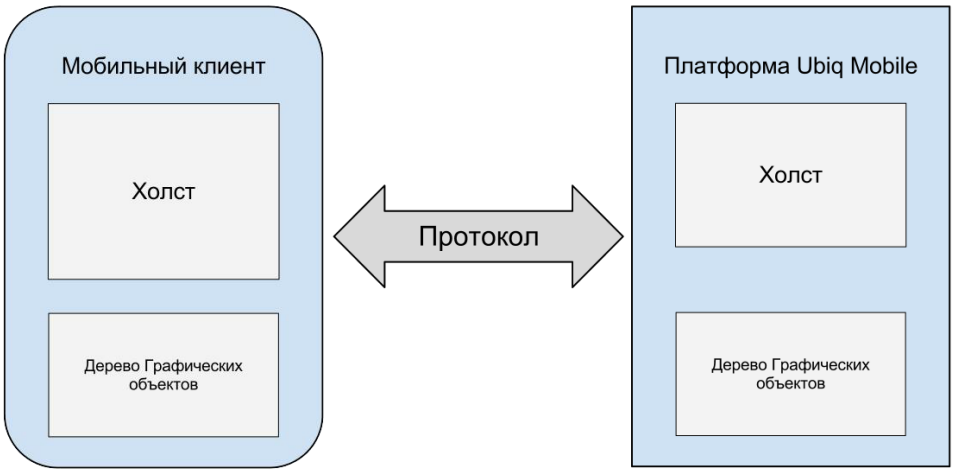


Схема 2 Отрисовка графического дерева на клиенте

3. Постановка задачи

Целью работы является реализация Android-клиента для терминальной платформы Ubiq Mobile. Клиент должен работать с моделью интерфейса NativeBasic.

Основные требования к программе:

- взаимодействие с терминальным сервером Ubiq Mobile по протоколу платформы Ubiq Mobile;
- работа в условиях относительно низкоскоростных мобильных соединений;
- приемлемая производительность.
- использование нативных элементов интерфейса

4. Архитектура клиента

Архитектура клиента включает в себя следующие функциональные модули:

- модуль соединения с сервером
- модуль-диспетчер
    - обработка команд сервера
    - отправка команд на сервер
    - аутентификация
- модуль разбор дерева графических объектов платформы Ubiq, получаемых через протокол
- модуль построение дерева нативных элементов
- модуль обработка действий пользователя

Модули разделены на потоки исполнения таким образом, чтобы задачи, требующие относительно длительного времени, выполнялись параллельно. Такие, как чтение данных из сети, отправка команды на сервер, разбор поступившей команды.

Основная часть клиента создана с использованием шаблона Facade, что позволяет *у*правлять различными модулями системы через один объект. Таким объектом является Controller, через который уже задействуются необходимые менеджеры.

4.1. Соединение с сервером

Модуль поддержки соединения с сервером решает задачи установки и разрыва соединений, а также повторных соединений в случае разрыва.

После Запуска приложения модуль пытается установить соединение с сервером, заданным в настройках приложения. Если сервер отвечает на запрос, то ему посылается информация для аутентификации клиента: номер клиента, данные о клиенте (размер экрана, версия), логин и пароль. При положительном отклике система переводится в состояние «соединение установлено», иначе выводится сообщение об ошибке. В случае разрыва соединения со стороны сервера происходит повторная попытка подключения.

В конце работы пользователь может сам разорвать соединение с сервером, выбрав пункт «logoff»/«logout» в меню устройства, или выйти из приложения, и соединение будет закрыто автоматически.

4.2. Модуль-диспетчер

Основной задачей данного модуля является работа с сервером – получение и отправка сообщений. Он получает сообщения от сервера, формирует команды и инициирует их исполнение.

Непосредственная работа с TCP/IP реализована с помощью стандартных сокетов Java. Поступившие команды помещаются в очередь на отправку или разбор.

Если работа модуля с сетью не производится, то его поток исполнения останавливается, что экономит вычислительные ресурсы и заряд аккумулятора смартфона, на котором функционирует клиент.

4.2.1. Обработка команд сервера

Обработка входящих команд от сервера происходит в параллельном потоке. Это связано с тем, что сервер посылает большие объемы данных (дерево графических объектов, изображения), считывание и обработка которых занимает относительно много времени.

Считывание команды происходит в два этапа: сначала происходит считывание первых трех байт команды, в которых закодирована длина, после чего происходит считывание оставшейся части команды. По полученному массиву формируется объект Command, который передается на обработку.

Для разбора команды требуется последовательно считать каждый ее байт из массива, что эквивалентно обходу дерева в глубину. После считывания длины и кода секции принимается решение, каким из обработчиков следует разбирать секцию. Общая структура обработчиков представлена на схеме 2.

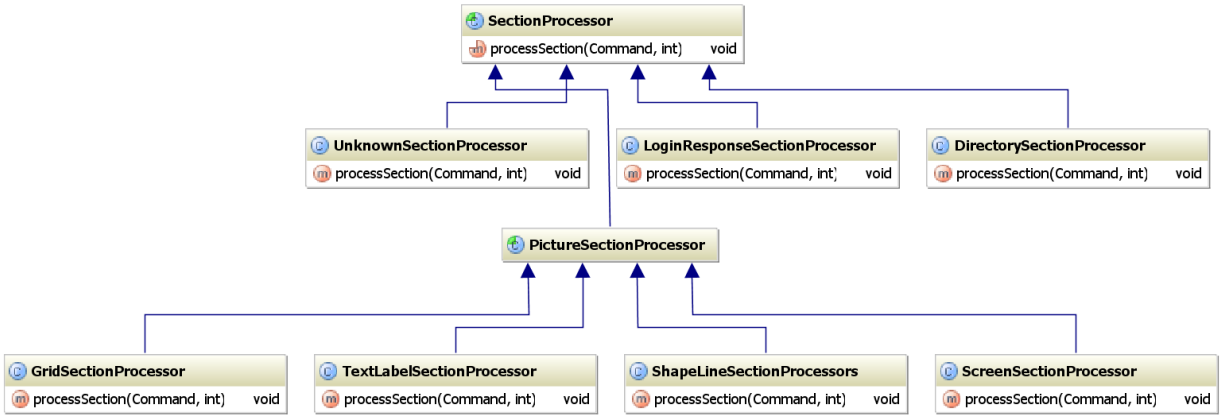


Схема 3 Структура обработчиков секций

Каждый обработчик считывает данные из секции и передает их на обработку соответствующему менеджеру. Если это ответ сервера на аутентификацию, то из массива байт считывается подтверждение или отказ на вход в систему. При положительном ответе присваивается номер соединения, который будет использоваться для запросов. Если это передача графической информации, то на основе команды создаются объекты, которые затем передаются UI менеджеру для создания нативного дерева объектов.

Если на разбор приходит секция, код которой не известен клиенту, то вызывается UnknownSectionProcessor, который пропускает данные секции.

4.2.2. Отправка команд на сервер

Формирование исходящих команд происходит примерно таким же образом, как и разбор входящих команд. Модулю формирования новых команд передаются данные о команде, где соответствующий CommandBuilder формирует команду из секций, которые создаются соответствующими SectionBuilder'ми. Затем результат передается модулю взаимодействия с сервером, который самостоятельно пересылает данные серверу.

4.2.3. Аутентификация

Аутентификация на сервере происходит путем отправки на сервер команды Login, в которой клиент передает свой id и физические параметры

устройства (в частности, разрешение экрана). Кроме того, в этой же команде серверу передается код используемой модели обмена - NB. После этого клиентом обрабатывается ответ, и если аутентификация прошла успешно, то записывается идентификационный номер соединения, и модуль-диспетчер переходит в ожидание новых команд.

4.3. Разбор дерева графических объектов

Разбор команды проходит в отдельном потоке, но в связи с тем, что создание нативных графических элементов может происходить только в UiThread'e, их нельзя создавать напрямую из потока разбора команды. Данные у различных объектов кодируются соответствующими флагами. Поэтому при разборе только считываются флаги этих элементов, которые передаются в отдельный модуль (UI менеджер). Для всех элементов есть общий набор флагов – StandardFlags, от которого наследуются специфичные флаги для конкретных элементов.

Интерфейс приложения создается при использовании паттерна Factory Method, что позволяет UI менеджеру не заботиться о типе создаваемых элементов. Поэтому вместе с флагами передается и соответствующий builder элемента.

Такой подход позволяет обрабатывать поступившие команды, не замедляя работу интерфейса приложения.

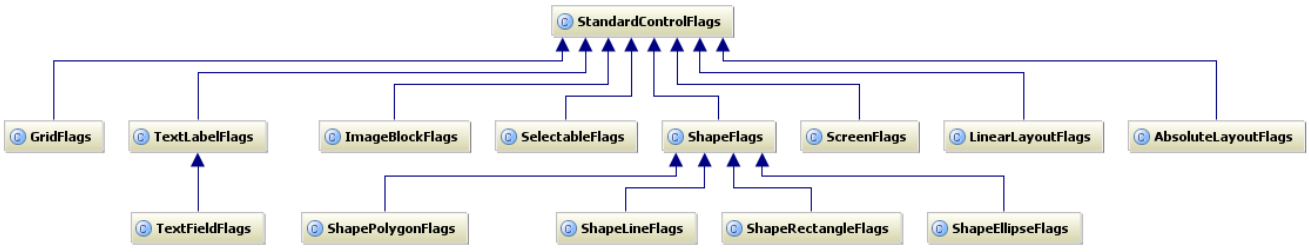


Схема 4 Флаги графических элементов

4.4. Построение дерева нативных элементов

Расширенный графический интерфейс платформы Ubiq Mobile (Extended Graphics Application Programming Interface) близок по структуре к интерфейсу платформы Android, но имеются значительные расхождения. Некоторые элементы специфичны и не имеют прямых аналогов, кроме того существуют отличия в поведении некоторых элементов.

Чтобы обеспечить максимальную совместимость элементов графического интерфейса платформы Ubiq (EGAPI) с нативными элементами Android’a, был создан набор управляющих элементов, расширяющих стандартные элементы системы. В случае если есть нативные элементы с требуемой функциональностью, иначе такие элементы создаются.

На данный момент клиент поддерживает следующие элементы:

- Панели.

Элемент интерфейса Ubiq	Нативный элемент платформы
UbiqScreen	FrameLayout
UbiqLinearLayout	LinearLayout
UbiqAbsoluteLayout	AbsoluteLayout
UbiqGrid	TableLayout
UbiqSelectable	FrameLayout

- Элементы управления

UbiqTextLable	TextView
UbiqTextField	EditText
UbiqImageBlock	ImageView

- Примитивы

UbiqLine	Line
UbiqRectangle	Rectangle
UbiqEllipse	Ellipse
UbiqPolygon	Path

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

4.5. Обработка действий пользователя

Клиент разработан таким образом, чтобы минимизировать загруженность потока пользовательского интерфейса, что обеспечивает быстрый отклик на нажатие клавиш вне зависимости от загруженности всей системы. Такой подход делает клиент максимально интерактивным.

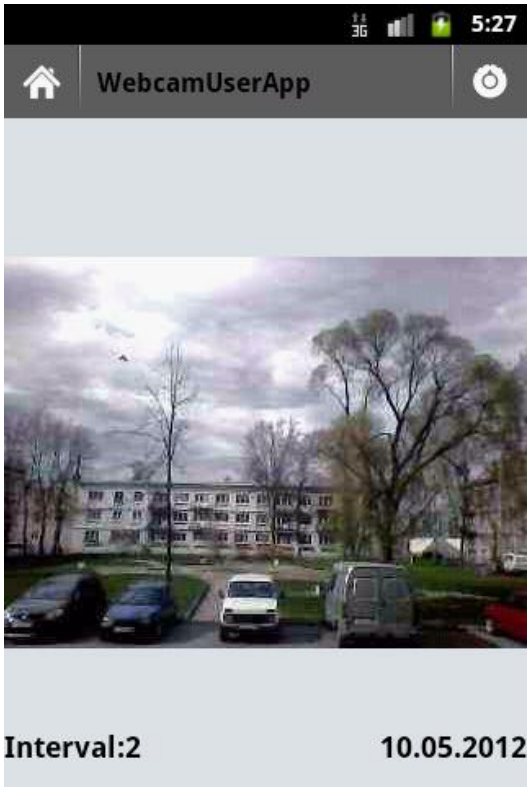
Интерфейс Android’a предусматривает использование клавиатуры только в полях ввода, в то время как в клавишных телефонах кнопки клавиатуры используются и как управляющие элементы (аналог кнопок или элементов меню), поэтому для обеспечения совместимости приложений, разработанных под обычные телефоны, была добавлена виртуальная клавиатура, эмулирующая клавиатуру телефона.

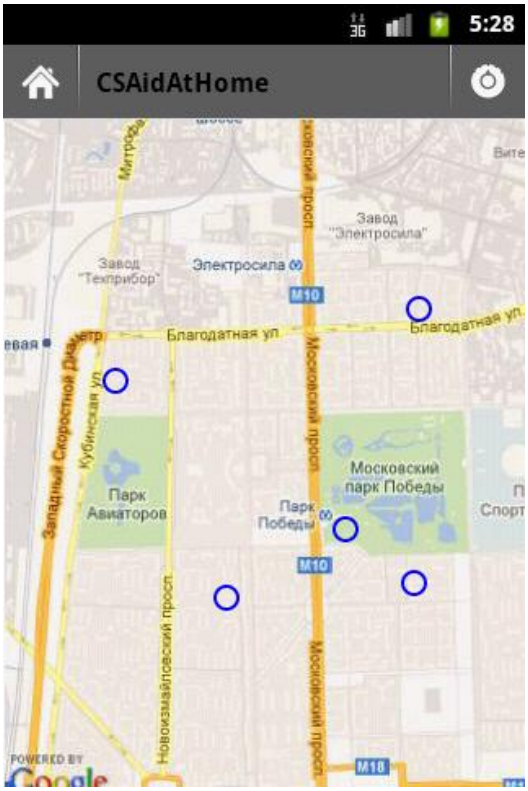
5. Заключение

5.1. Результаты

В ходе выполнения работы было разработано приложение, реализовано и протестировано, являющееся частью программной платформы Ubiq, которое корректно работает с существующими приложениями системы. Обеспечена стабильная работа приложения без задержек при асинхронной подкачке команд с сервера.

Примеры работы тестовых приложений платформы. На первом скриншоте демонстрируется работа приложения, которое считывает данные с web-камеры и с определенном интервалом передает изображения на клиент. На втором показано приложение, демонстрирующее работу с картами google.





5.2. Перспективы развития

5.2.1. Реализация дополнительных управляющих элементов

На данный момент реализованы не все представленные в системе элементы интерфеса. Кроме того, возможно создание библиотеки уникальных элементов для платформы, т.е. не предусмотрены платформой Android.

5.2.2 Вызов функций на клиенте

Для расширений возможной функциональности и интерактивности приложений в протоколе Ubiq Mobile предусмотрен механизм вызова специальных функций клиента. Это может быть получение значений компаса, акселерометра, местоположения, обращение к камере устройства, контактам пользователя, считывание баркодов и qr-кодов и т.п. Реализация данных возможностей является отдельной задачей.

Список литературы

1. Onossovski V., Terekhov A. «Ubiq Mobile – a New Universal Platform for Mobile Online Services» // Proceedings of 6th Seminar of Finish-Russian University Cooperation (FRUCT) Program. Helsinki, 2009.
2. Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес, «Приемы объектно-ориентированного проектирования. Паттерны проектирования»