МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

кафедра Компьютерных технологий

«Допущен к защите»
Заведующий кафедрой _____
_____
(Ф.И.О., ученая степень, звание)
_____ «_____»_____ **20__ г.**
(подпись)

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Технология разработки кроссплатформенных приложений для мобильных устройств

Специальность Вычислительная техника и программное обеспечение

Выполнил (а) Кан М.А. БТ-12-3
(Фамилия и инициалы) группа

Научный руководитель Тургамбаев Е.С., к.ф.м.н.
(Фамилия и инициалы, ученая степень, звание)

Консультанты:

по экономической части:
Бекенова А.Г., к.э.н., доцент
(Фамилия и инициалы, ученая степень, звание)
_____ «26» 05 2016 г.
(подпись)

по безопасности жизнедеятельности:
Прокофьева Н.Г., д.х.н., профессор
(Фамилия и инициалы, ученая степень, звание)
_____ «19» 05 2016 г.
(подпись)

по применению вычислительной техники:
Тургамбаев Е.С.
(Фамилия и инициалы, ученая степень, звание)
_____ «02» 06 2016 г.
(подпись)

Нормоконтролер: Тургамбаев, к.ф.м.н.
(Фамилия и инициалы, ученая степень, звание)
_____ «02» 06 2016 г.
(подпись)

Рецензент: Тургамбаев Е.С.
(Фамилия и инициалы, ученая степень, звание)
_____ «02» 06 2016 г.
(подпись)

Алматы 2016 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Факультет Архитектурных и информационных технологий
Специальность Вычислительная техника и программное обеспечение
Кафедра Компьютерных технологий

ЗАДАНИЕ
на выполнение дипломного проекта

Студент Кан Максим Александрович
(фамилия, имя, отчество)

Тема проекта "Технология разработки кроссплатформенных приложений для мобильных устройств"

утверждена приказом ректора № 21 от «10» марта 2016 г.
Срок сдачи законченной работы «11» июня 2016 г.
Исходные данные к проекту требуемые параметры результатов проектирования (исследования) и исходные данные объекта:
Рынок мобильных приложений растет с каждым годом. Появление все новых мобильных платформ следует разработать приложение, которое поддержит функционал интерфейса и легко взаимодействующее с различными платформами на рынке

Перечень подлежащих разработке дипломного проекта вопросов или краткое содержание дипломного проекта:
1) Анализ рынка мобильных приложений
2) Правила и рекомендации для разработки приложения
3) Анализ предметной области
4) Разработка и оценка интерфейса
5) Анализ современных средств разработки мобильных приложений
6) Этапы разработки приложений
7) Технико-экономическое обоснование
8) Описание безопасности жизнедеятельности

Перечень графического материала (с точным указанием обязательных чертежей)

1) Рисунки архитектуры приложений
2) Рисунки интерфейса инструментов разработки
3) Рисунки описывающие требования к приложению
4) Структурные схемы работы приложения
5) Таблицы для описания технико-экономического обоснования
6) Рисунки и таблицы для информации о безопасности жизнедеятельности

Рекомендуемая основная литература

Cathy Bennett - Objective-C (2010 г)
xamarin.com
appcelerator.com
iOS Programers Guide
github.com

Консультанты по проекту с указанием относящихся к ним разделов

Раздел	Консультант	Сроки	Подпись
БЖД	Приходько Н.Г.	18.03 - 13.05.16	
Экономическая часть	Бекишева А.Ч.	19.04 - 25.05.16	
Нормоконтроль	Гургенбаев Е.С		

Г Р А Ф И К
подготовки дипломного проекта

№ п/п	Наименование разделов, перечень разрабатываемых вопросов	Сроки представления руководителю	Примечание
1	Сбор необходимого материала	01.04.16	
2	Исследование предметной области	03.04.16	
3	Постановка задачи	04.04.16	
4	Выбор платформы и среды разработки	04.04.16	
5	Разработка интерфейса	05.04.16	
6	Анализ форм приложения тестирование и отладка	15.04.16	
7	Оформление отчета	25.04.16	
8	Написание раздела „технико-экономического обоснования"	10.05.16	
9	Написание раздела „Безопасность жизнедеятельности"	26.05.16	

Дата выдачи задания «____»_____20___ г.

Заведующий кафедрой _____ _____
(подпись) (Фамилия и инициалы)

Руководитель _____ _____
(подпись) (Фамилия и инициалы)

Задание принял к исполнению
студент _____ _____
(подпись) (Фамилия и инициалы)

Аңдатпа

Бұл Дипломдық жұмыста кросс-платформалық қолданбалардың технологиясы құрылуы зерделді.

Қосымшалар мен бағдарламалық қамтамасыз салыстырмалы талдау жүргізілді.

Интерфейс жаңа технологияларды пайдалана отырып, әзірленген және бағдарламалық қамтамасыз ету үшін барлық заманауи талаптарға сай келеді.

Сондай-ақ, соңғы тарауларда техникалық-экономикалық негіздемесін қамтамасыз ету және жобаның бағасын есептеу, сондай-ақ қауіпсіздікті қарастырады.

Аннотация

В данной дипломной работе изучается технология разработки кроссплатформенных приложений.

Проведен анализ существующих на рынке мобильных решений, типов приложений и программного обеспечения.

Интерфейс тестового приложения разработан с использованием новейших технологий отвечает всем современным требованиям к программному обеспечению.

Так же в последних главах рассматривается технико-экономическое обоснование и рассчитывается цена разработки проекта и рассматривается вопросы безопасности жизнедеятельности.

Annotation

In this diploma work we study the technology development of cross-platform applications.

The analysis was completed on the market of mobile solutions, the types of applications and software.

The interface of test application is designed using the latest technology and meets all modern requirements to software.

Also in the last chapter provide a feasibility study and calculate the price of the project, and addresses safety.

Содержание

Введение

Сегодня практически у каждого человека под рукой есть мобильный телефон. По всему миру прослеживается тенденция роста использования мобильного интернет-трафика с персональных сотовых устройств человека. Рост вызван тем, что в некоторых развивающихся странах и странах с отстающим развитием наблюдается следующая тенденция: у человека может не быть компьютера и широкополосного интернет-доступа в доме, но у него есть мобильный телефон и хоть какой-то доступ к интернету через него. Кроме того, мобильный телефон – спутник современного человека и практически всегда находится под рукой, поэтому получать информацию с него человеку быстрее и удобнее, чем со стационарного компьютера.

По данным отчета Международного союза коммуникаций к концу 2015 года совершенно более 7 миллиардов активных подключений к сотовой сети. Для сравнения: в 2000 году подключений к сотовой сети было совершено ли 738 миллионов.

Также из отчета Международного союза коммуникаций была сформирована инфографика по пользованию интернетом в домах, подсчет делался в разрезе на 100 жителей/домов. Самый большой скачок с 2000 года показал мобильный интернет, им в 2015 году пользуется 96.8 жителей/домов из 100.

Сегодня на растущем рынке мобильного программного обеспечения, преобладают нативные приложения, т.е разработанные под определенную платформу. Не смотря на свою распространенность, они имеют свои недостатки, основной из них, что они могут быть использованы только на устройствах с определенной платформой. А появление всё нового программного обеспечения и увеличение количества существующих платформ (Android, iOS, Windows Phone, BlackBerry и др.) ставит перед разработчиками новые требования. В качестве решения этой проблемы все популярнее становится использование кроссплатформенной разработки приложений для современных устройств.

Сегодня все чаще вместо традиционного написания скриптов и кодов на популярных языках программирования C++, Java и др., разработчики используют универсальные оболочки, такие как Android SDK, iPhone SDK, Adobe Flex, Java Development Kit, Windows Phone SDK. Однако они чаще всего заточены под одну платформу, данная же работа акцентирует внимание на создании мультиплатформенных мобильных приложений.

1 Предметная область

1.1 Описание основных типов мобильных приложений

1.1.1 Нативные приложения

Компании, имеющие собственные мобильные платформы, как правило, стараются обеспечить разработчиков всем необходимым набором инструментов для создания мобильных приложений.

При этом они пытаются сделать процесс создания мобильного приложения наиболее простым и удобным. Для этого они наделяют свои платформы возможностью решения наиболее часто встречаемых задач, которые встают перед разработчиком во время работы.

Отличие инструментов разных мобильных платформ значительно и заключаются в основном в различии используемых языков программировани: Андроид приложения разрабатываются на языке Java, а приложения для платформы iOS разрабатываются на языке Objective-C, который является надстройкой над классическим языком C, используемым компанией Apple.

Различие также имеется в выборе специалистов, что исходит из выборов языков. Java - один из самых популярных языков программирования, который может обеспечить разработчика огромным количеством функций, что в свою очередь позволяет писать приложения самого разного плана. Поэтому количество специалистов на рынке достаточно большое, чего не скажешь о специалистах по языке Objective-C. Так как язык не является популярным и используется преимущественно только одной компанией, отсюда и недостаток специалистов на рынке. При этом спрос на них большой, так как продукты компании Apple пользуются популярностью среди пользователей, отчего и растет необходимость в создании все большего количества нативных мобильных приложений для них.

Усложняет все не только непопулярность выбранного языка программирования, но и условия, при которых разработчик может им пользоваться. Objective-C требует наличия компьютера под управлением Mac OS X и участия в программе для разработчиков от Apple, подписка на которую стоит 99$, и обновлять ее нужно ежегодно. Это отпугивает разработчиков и привлекает их больше к написанию приложений для Андроида, так как здесь доступ может иметь каждый.

Стоит отметить и различия в быстроте решения той или иной задачи. Objective-C в сравнении с Java менее многословный и более компактный, из-за чего решения задач на нем в среднем на 25% проще и быстрее, чем на Java. На рисунке 1.1 показаны примеры нативного приложения.

Разработка нативных приложений удобнее для разработки и отладки. Конечно, есть разница в том, на какой платформе происходит разработка, но,

в целом, при сравнении с написанием кроссплатформенных и гибридных приложений, нативные явно выигрывают в части удобства. Инструменты разработки нативных приложений более современные и развитые. Не мало важную роль играет и отсутствие лишних прослоек между кодом приложения и устройством.

Нативные приложения также отличаются документацией и обеспечиваемой технической поддержки.

Скорость нативных приложений выделяется не только при самом создании и написании приложения, но и во время его работы.

Также, так как нативные приложения пишутся с использованием специальных китов от разработчиков мобильной платформы, они по умолчанию более адаптированы к внешнему виду и интерфейсу устройства. Таким образом, пользователь чувствует себя наиболее комфортно при его использовании, так как расположение кнопок и в целом внешний вид ему интуитивно знакомы.

Нативные мобильные приложения, как уже говорилось, отличает и сложность в поиске хороших специалистов, в виду узкой специфики разработки под каждую конкретную мобильную платформу.

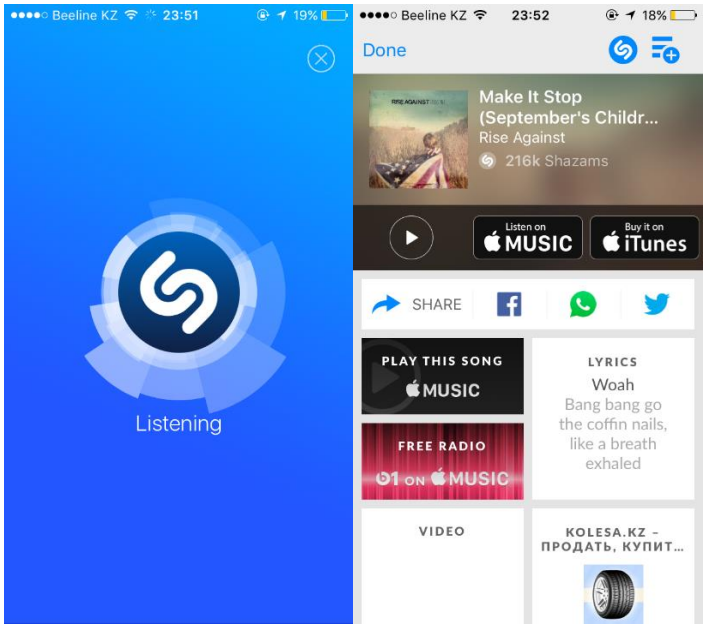

Рисунок 1.1 – Пример нативного приложения Shazam

1.1.2 Веб приложения

Веб-приложения отличаются возможностью использования веб-сервисов. То есть, если нет потребности в работе в офлайн режиме или работе с ресурсами устройства, то пользователь может использовать приложение через браузер посредством простого вбивания адреса в интернете. Это

удобное решение для приложений, которые так и так нуждаются в постоянном доступе к интернету, и при этом не имеют жестких требований в части графики и использования аппаратных средств устройства.

В данном случае применяются вполне стандартные, и можно даже сказать, традиционные инструменты, такие как HTML в части разработки самого интерфейса, и CSS в области создания вижуала и расположения виджетов, контролов и других дополнительных надстроек и инструментов. Также используется JavaScript, который отвечает за логическую составляющую приложения. Так как данные технологии применяются достаточно давно и повсеместно, то они могут смело применяться в создании сложных адаптированных приложений под различные устройства.

Приятным бонусом является и наличие разнообразного выбора инструментов и фреймворков, которые оказывают влияние на скорость разработки приложения. Другими словами, разработчик может сконцентрироваться не на решении типовых задач, а на конкретной проблеме.

Конечно, без минусов и трудностей обойтись невозможно. Разработка приложений под мобильные устройства сталкивается с проблемой ограниченности оперативной памяти и процессора, чего не встретить при разработке приложений для компьютера. Данные особенности накладывают дополнительные требования к знаниям и компетенциям специалиста-разработчика. То есть, не каждый хороший разработчик приложений для ПК может быть настолько же эффективным разработчиком для мобильных устройств. Здесь производительность работы приложения может упасть от неправильного использования памяти, и от этого никто не застрахован. На рисунке 1.2 показан пример веб приложения.

Преимущества веб-приложений заключаются в охвате возможных платформ, и, соответственно, времени разработки. Здесь играет роль возможность охвата сразу нескольких платформ, что освобождает разработчика от необходимости написания приложения под разные платформы отдельно, и в целом простота написания веб-приложений. От этого и количество специалистов по разработке мобильных приложений больше. Нельзя не отметить, документацию и поддержку.

Из минусов можно отметить низкую производительность и скорость работы. Это следствие дополнительной прослойки - веб-браузера. Сомнительный минус, но все же он есть - отладка: на компьютере она достаточно проста и легка, а вот на устройствах могут возникнуть проблемы.

Также стоит отметить ограниченность инструментов и использования аппаратных средств устройств. Как правило, веб-приложения имеют доступ к хранилищу и геолокации, но не имеют доступа к камере или файловой системе.

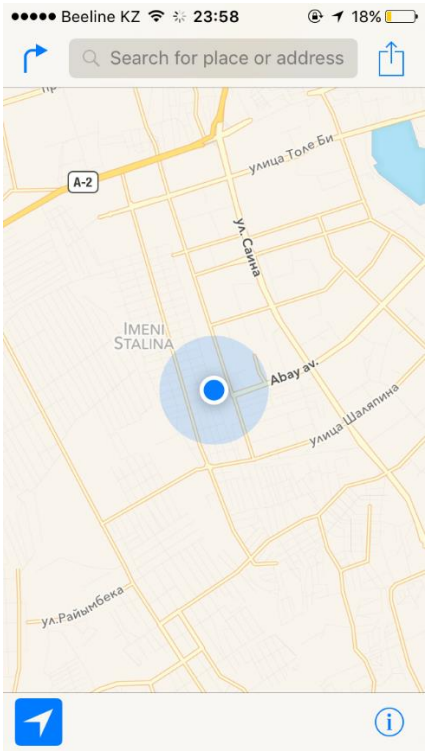

Рисунок 1.2 – Пример веб приложения Google Maps

1.1.3 Гибридные приложения

Идея применения веб-технологий реальна и довольно интересна. Разработав приложение единожды, потенциально можно распространить его на большое количество платформ, поскольку веб-браузеры интегрированы по умолчанию во все прогрессивные мобильные операционные системы. Беря во внимание, собственно, фреймфорки и инструментарий для исследования мобильных HTML5-приложений непрерывно улучшаются, то почти все разработчики программного обеспечения, знакомые с веб-технологиями, имеют все шансы начать проектировать мобильные приложения прямо в настоящий момент.

Но почти всем приложениям нужно читать и хранить файлы на SD-карте, делать снимки через фотокамеру, получать известия о системных событиях и так далее. Веб-приложения, запущенное в простом браузере, таковых способностей не имеет.

Поэтому возникли инструменты, позволяющие проектировать логику и интерфейс на HTML и JavaScript, имея доступ к ресурсам мобильного девайса. Фактически все SDK мобильных платформ предлагают пользователю специализированную составляющую браузера для введения в нативное приложение — такое как WebView в Android, UIWebView в iOS и другие. Точно такой же мобильный браузер, но с функцией передать в веб-приложение (в JavaScript-код) часть из нативного кода и принимать сообщения из JavaScript обратно в нативную часть приложения.

Следовательно, веб-приложения с элементами нативного кода, сможет обрести доступ ко всем аппаратным ресурсам девайса.

На таком методе базирована работа известного фреймворка для проектирования гибридных приложений — Phonegap. С помощью специальной технологии он позволяет объединить функции веб-приложения с нативным, автоматически для нескольких платформ.

Ресурсы Phonegap предлагают пользователям создание плагинов и расширений. Нужный метод или функция может быть добавлена на соответствующем нативном языке удобно и быстро. Фреймворк предлагает большое количество уже разработанных для нужд программистов плагинов и расширений, поэтому создание собственного уникального аддона вам особо не пригодится.

В бизнесе, а особенно в корпоративных центрах, технология разработки гибридных приложений занимает широкое место. Это напрямую зависит от сегодняшней популярности веб-технологий. Несомненное достоинство гибридных приложений, а также большим аргументом в их выборе является их кроссплатформенность. Так как в корпоративных нуждах быстрота работы интерфейса приложения и его непосредственная производительность не является определяющим фактором, то повсеместное применения данной технологии является обычным выбором. На рисунке 1.3 показан пример гибридного приложения.

Плюсы данной технологии очевидны:

—*Кроссплатформенность*. Платформа phonegap предоставляет поддержку более 7 ОС.

— *Большое сообщество разработчиков*

— *Скорость разработки*. Правда бывают случаи отладки приложения под отдельную платформу и это очень замедляет процесс разработки.

Минусы:

— *Отзывчивость приложения*. Специалисты вынуждены постоянно соблюдать баланс между ресурсами устройства и нагрузкой на него.

—*Обслуживание*. Очень сложно совмещать компоненты веб и нативного приложения, соответственно оно будет требовать больших сил в сопровождении и отладке

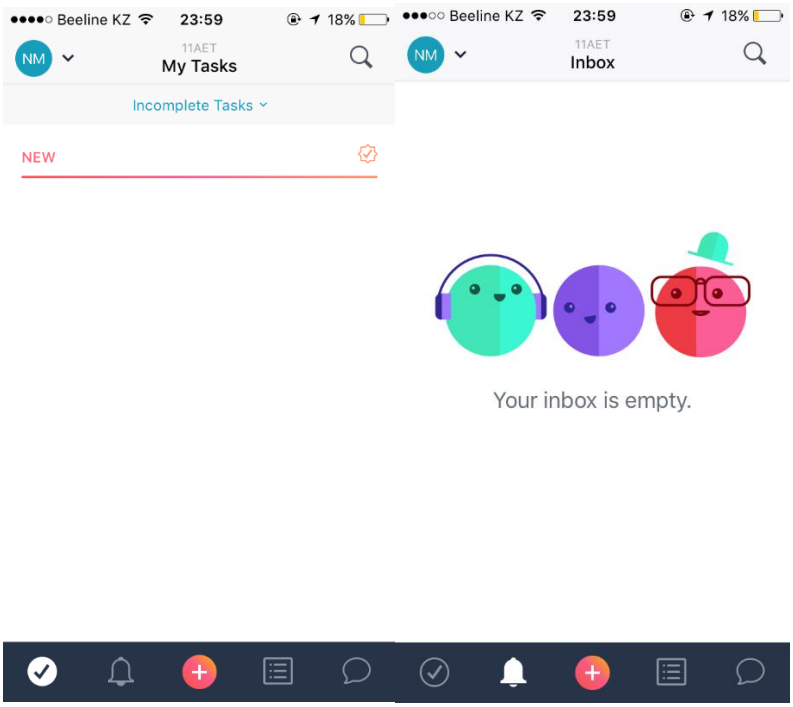

Рисунок 1.3 – Пример гибридного приложения Asana

1.1.4 Кроссплатформенные приложения

Кроссплатформенные приложения отличаются от гибридных довольно специфично и расплывчато. В данной работе этот вид приложения будет приниматься, как тот, в котором компиляция нативного кода происходит непосредственно в выполняемый файл. Также в последующим используется большая часть кода повторно.

Обычно код, ответственный за бизнес-логику, используется приложением повторно. Часть интерфейса для пользователя чаще всего разрабатывается отдельно. Практика показывает, что для отладки такого типа приложения под различные платформы привлекают разных специалистов. Мультиплатформенным приложениям присущи черты нативных – производительность и черты гибридных – повторное использование кода.

В данной категории можно упомянуть два инструмента.

Первый — Appselerator Titanium. Код приложения пишется на JavaScript. Затем он компилируется в нативный код для платформ (поддерживаются всего три — iOS, Android и BlackBerry). Далее, можно собрать этот код в установочный файл приложения для каждой мобильной операционной системы. У Titanium есть собственная среда разработки Titanium Studio с возможностью отладки собственного кода на JavaScript. Titanium, так же как и Phonegap, поддерживает расширения на Java или Objective-C для добавления необходимой функциональности.

Второй набирающий популярность инструмент — MonoTouch от компании Xamarin. Это действительно многообещающий инструмент. В

качестве языка разработки используется язык C#, что может стать решающим фактором при выборе технологии для разработчиков с опытом работы в .NET. MonoTouch компилирует проект на C# сразу в нативное приложение. Кроме того, Mono поддерживает визуальное создание интерфейса iOS-приложений с помощью InterfaceBulder-a, что позволяет использовать файлы nib из нативного редактора X-Code.

MonoTouch может позиционироваться не только как инструмент кроссплатформенной разработки, но и как посто возможность разрабатывать полностью нативные приложения на знакомом языке C#. При этом, однако, появляется возможность переиспользования кода для разных платформ. Например, если приложение содержит большое количество бизнес- и инфраструктурной логики, не относящегося напрямую к интерфейсу пользователя, то, по некоторым данным, переиспользование кода может достигать 80%.

Достоинства данного типа:

— *Быстродействие*

— *Скорость проектирования*

— *Мультиплатформенность*

Недостатки:

—*База специалистов.* Из-за небольшой специфичности программистов с опытом проектирования в, например, MonoTouch найти сложно, даже сложнее чем разработчиков для нативных SDK

—*Удобство отладки.* Отладка кода в Titanium, и невозможно найти доступ в исходниках самого фреймфорка, что затрудняет поиск проблем. Что касается MonoTouch, то, видимо ввиду недостаточной зрелости инструмента, у многих разработчиков возникают постоянны проблемы при отладке и запуске приложения.

1.1.5 Основные преимущества и недостатки

Таблица 1 – Основные преимущества и недостатки разных типов приложений

Нативное	▪ Максимальная функциональность и скорость работы ▪ Не требуется интернет-соединение для использования ▪ Имеет доступ к ПО смартфона (GPS, плеер, камера) Распространение через магазины	▪ Выше стоимость и длиннее сроки разработки ▪ Требует от разработчика знаний определенной среды программирования

	приложений	▪ Работает только с одной платформой ▪ При косметических изменениях необходимо выпускать обновление
Веб(HTML5)	▪ Кроссплатформенность ▪ Не требует загрузки из магазина мобильных приложений ▪ Можно легко адаптировать обычный сайт ▪ Легче найти веб-разработчика нежели разработчика под определенную платформу Простота создания и поддержки	▪ Требует подключения к интернету ▪ Не имеет доступа к ПО смартфона ▪ Не может отправлять push-уведомления ▪ Должен быть запущен интернет-браузер ▪ При продаже требуется использование своей платежной системы
Гибридное	▪ Функциональность нативного приложения на независимой платформе ▪ Запускается не из браузера в отличии	▪ Загружается из магазина мобильных приложений (необходимо соответствовать

	от веб приложения Возможность независимого обновления ▪ Распространение через магазины приложений	требованиям) ▪ Разработчик должен быть знаком с разными API
Кроссплатформенное	▪ Поддержка платформ ▪ Функциональность приложения ▪ Время разработки ▪ Затраты на разработку и поддержку	▪ Удобство отладки ▪ Наличие специалистов

1.2 Правила и рекомендации для разработки приложения

Большинство компаний на рынке мобильных устройств, такие как Google и Apple, выдвигают специальные требования разработчикам, в соответствии с которыми приложение последних публикуется в магазине приложений. Также существуют некоторые требования к дизайну и интерфейсу приложения (User Interface, UI), для того, чтобы оно органично вписывалось в общую картину интерфейса системы. Официально с ними можно ознакомиться на сайтах компаний.

1.2.1 Разработка интерфейса

Важные 6 частей по рекомендации к разработке внешнего интерфейса приложения представлены в разделе «Проектирование».

Такие как – «Начать» (Get Started), «Стиль» (Style), «Шаблоны» (Patterns), «Блоки» (Blocks), «Загрузки» (Downloads), «Видео» (Videos).

Первый раздел «Начать» разделен на три части. В первой части называется «Creative Vision», он не включает в себя особо важную информацию, здесь разработчику подсказывают, чтобы интерфейс приложения интуитивно понятен, гибок и красив. Для его актуальности это необходимо, и чтобы оно могло выполнять свои функции быстро.

Второй раздел «Design Principles» рассказывает о принципах дизайна и этапах его разработки. Также дается пара примеров с актуальными мобильными ОС и пара советов для разработчика.

3-я часть называется «UI Overview» (Обзор интерфейса) и обрисовывает ключевые системные составляющие интерфейса ОС, в том числе панель навигации, шторка уведомлений и т.п.

Информацию по оформлению интерфейса, цветах, шрифтах, моделях и т.д. содержит в себе раздел «Стиль».

Раздел «Шаблоны» включает информацию по оформлению различных составляющих интерфейса, а еще подсказки по навигации в интерфейсе приложения и содействии с ним.

Раздел «Блоки» несет в себе информацию о формах, использующихся в приложении.

Раздел «Загрузки» позволяет загрузить базовые для ОС элементы оформления, а также предоставляет использующуюся в Android палитру.

Часть о «Видео» состоит из ссылок на видео конференции для разработчиков и видео уроки.

Оформление

На современных мобильных ОС Android и iOS существует множество различных мобильных устройств, с совершенно разными форм-факторами и диаметрами экранов. Зная свойства графической подсистемы на современных ОС, можно определиться с дизайном на различных форм-факторах устройств. На рисунке 1.4 представлены основные форм-факторы устройств.

Оформление - алгоритм для приведения приложений к общему стилю. Стиль характеризует визуальные характеристики частей, из которых состоит пользовательский интерфейс, в том числе расцветка, высота и объем шрифта.

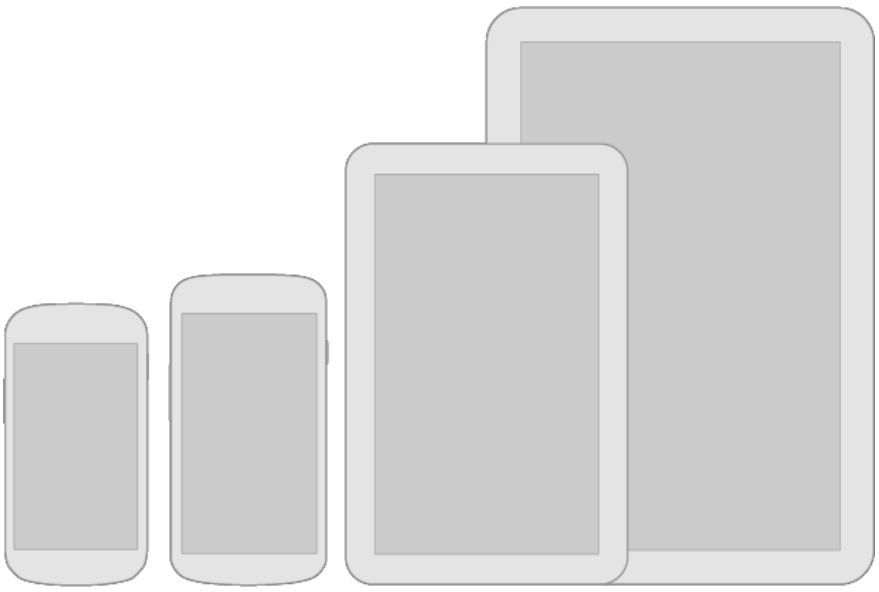

Рисунок 1.4 – Форм-факторы устройств

К общему визуальному стилю можно прийти, используя эти небольшие данные.

Можно выбрать тему приложения, чтобы она соответствовала общему визуальному виду и тематике. Или выбрать из шаблонных, которые

предлагает система, чтобы она служила отправной точкой вашему будущему дизайну.

Для отображения активности, отключённой кнопки или нажатия на нее используйте различные цвета, а также эффекты осветления и затемнения. На рисунке 1.5 представлены стандартные шаблоны кнопок.

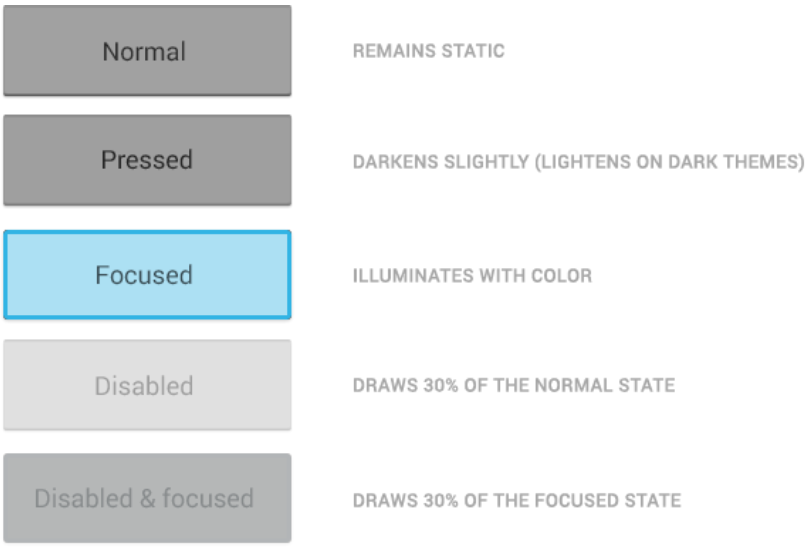

Рисунок 1.5 – Шаблоны кнопок

Используйте жесты любой сложности для навигации в приложении или же совершения каких-либо действий. На рисунке 1.6 изображен пример жеста.

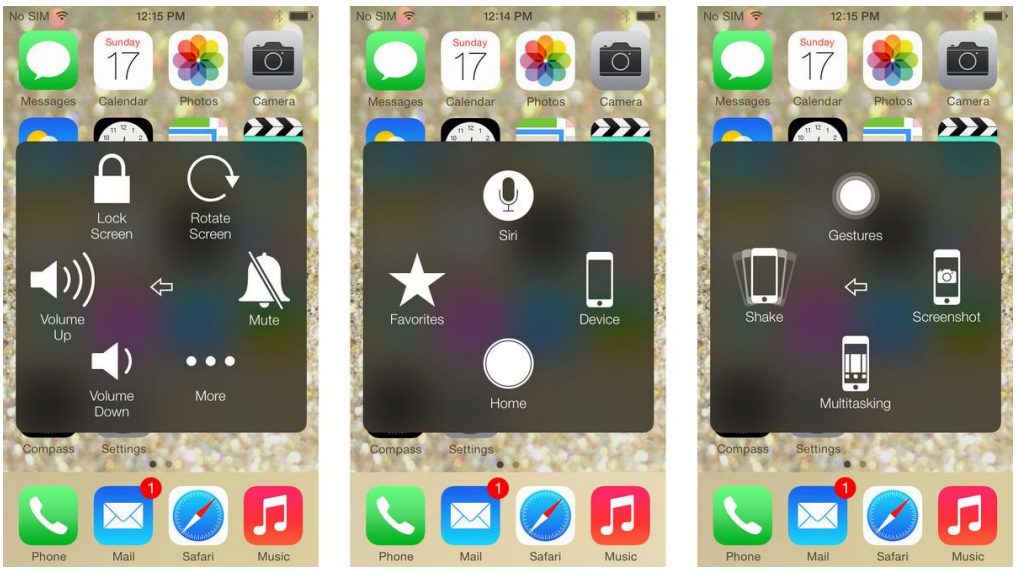

Рисунок 1.6 – Шаблон жестов

При использовании прокручиваемых элементов интерфейса, используйте эффект границы экрана. На рисунке 1.7 продемонстрирован шаблон границы экрана.

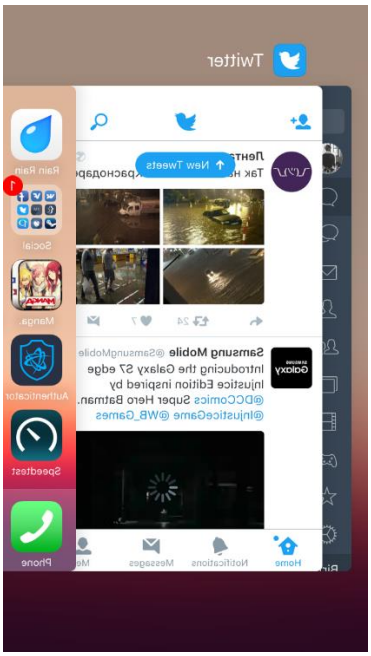

Рисунок 1.7 – Шаблон границы экрана

Приспособления различаются не столько физическими объемами экранов, да и плотностью пикселей на нем (DPI). Для упрощения работы художников есть немного заготовленных образчиков для телефонных аппаратов и планшетов.

Есть DPI всевозможных объемов, для простоты они поименованы LDPI, MDPI, HDPI, XHDPI, XXHDPI и XXXHDPI, и исконно в прошивке любого прибора написан сообразный ему уровень плотности.

Нужно будет постоянно предусматривать эти макеты экрана, потому что при другом развитии событий ваше прибавление имеет возможность исказиться и всецело испортиться интерфейс прибавления, также эмоция о нем. Впрочем данное не составит практически никакого труда, по следующим причинам все шаблоны теснее заложения в инструментарий для разработчиков, который механически приладит интерфейс под необходимую плотность, в случае если вас что-нибудь станет не организовывать, то вы лично в ходе исследования, в графическом видео можете все поправить и организовать так, как вам потребуется. На рисунке 1.8 представлены возможные варианты плотности экрана.

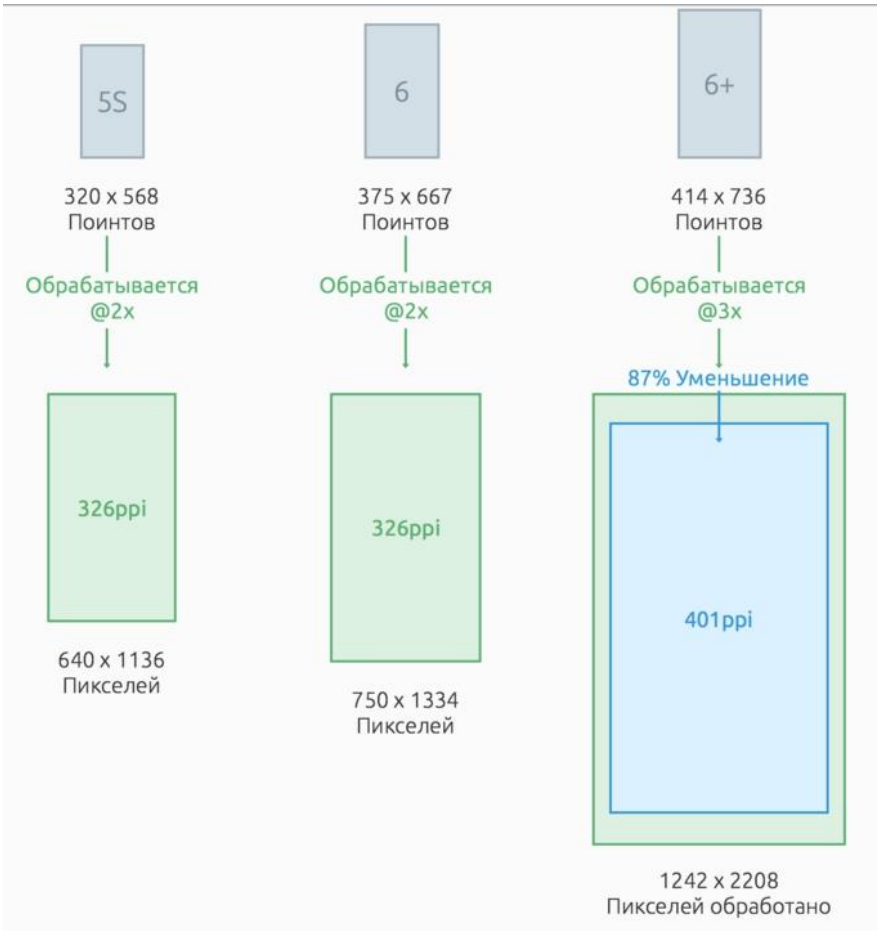

Рисунок 1.8 – Макеты плотности экрана

Все элементы интерфейса выполняются в размере кратном 48dp, то есть от 48x48 dp и выше. Данный размер соответствует примерно 9мм. на экране и является оптимальным по удобству. На рисунках 1.8 и 1.9 отображены рекомендуемые размеры для элементов интерфейса.

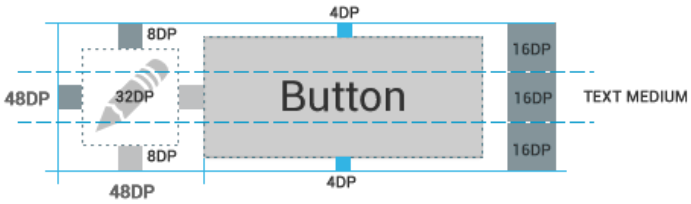

Рисунок 1.8 – Пример значка и кнопки

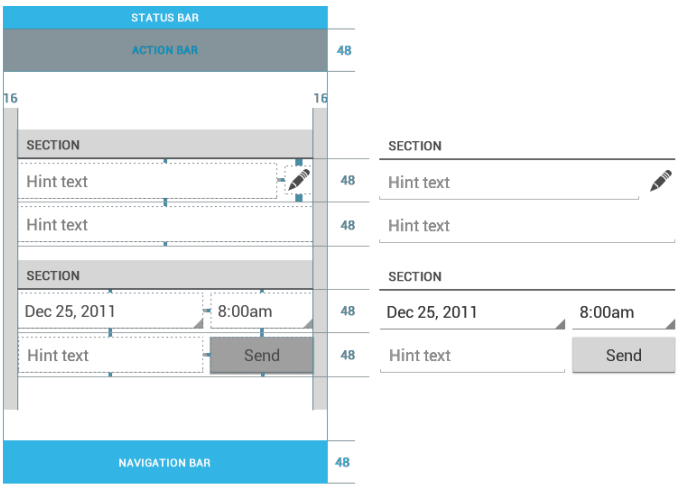

Рисунок 1.9 – Пример элементов размером 48dp в интерфейсе

При названии файлов лучше использовать префиксы, тогда иконки будут группироваться внутри директории и их поиск существенно облегчиться.

В коде программы также лучше разбить все элементы интерфейса на отдельные группы, что снизит затраты времени в случае необходимости что-либо исправить.

Например:

```
res /...
    drawable - ldpi /...
        finished_asset .
        png
    drawable - mdpi /...
        finished_asset .
        png
    drawable - hdpi /...
        finished_asset .
        png
    drawable - xhdpi /...
        finished_asset .
        png
```

При использовании текста в интерфейсе, описании его элементов и прочего, старайтесь максимально сократить его размер и упросить его восприятие. Избегайте использования ненужных слов, несущих не требующуюся информацию, сокращайте размер текста используя цифры вместо букв. Однако ненужно переусердствовать, если данный текст не отражает всю суть информации, которую требуется донести, то лучше сделать его более объемным, но пользователь полностью поймет, что от него хотят.

Шрифты (iOS)

9-ая операционная система поддерживается не на всех устройствах, т.е. рисуя макеты для iPhone 6, используя шрифт SF, необходимо понимать, что у некоторых пользователей будет старая добрая гельветика. (Это нестрашно,

разве что, может помешать в максимальном значении символов в одной строке). На рисунке 1.10 показаны стандартный шрифт для iOS

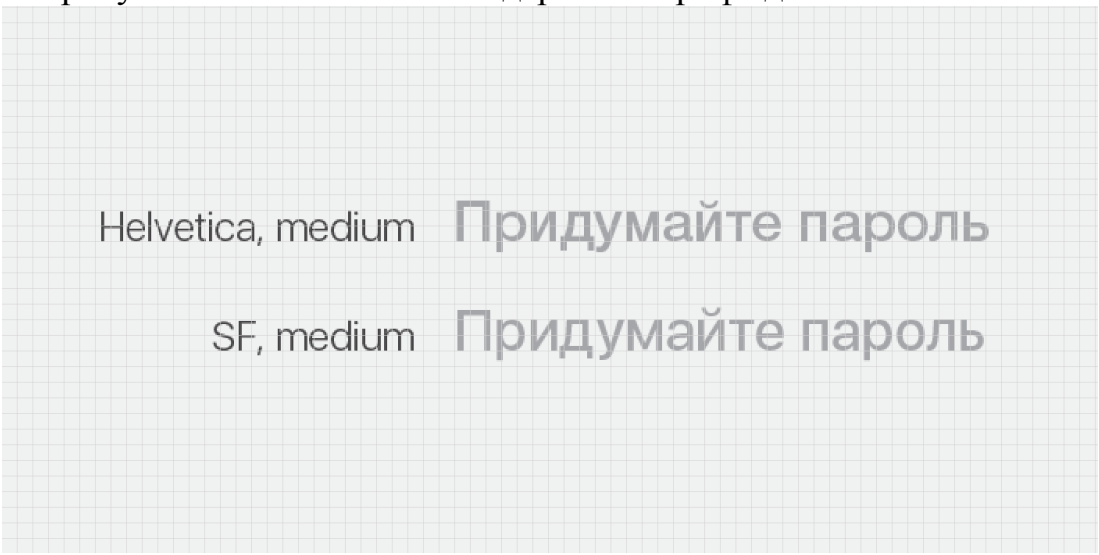

Рисунок 1.10 – Шрифты iOS

Элементы (iOS)

Использовать прилипающую кнопку в дизайне не стоит. Это связано с тем, что она хорошо смотрится на продуманном (прорисованном) макете, но на других экранах перекрывает большую часть вместе с клавиатурой. На рисунке 1.11 отображены макеты приложения

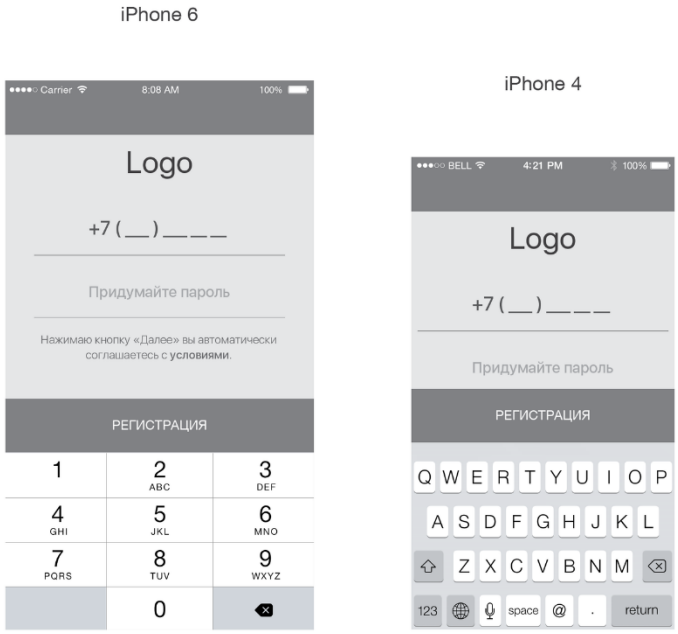

Рисунок 1.11 – Макеты приложения iOS

Иконка мобильного приложения — это небольшая картинка, представляющая приложение на маркете, таком как Google Play или App Store, и устройстве пользователя. Если вскоре вам предстоит создать свой первый дизайн иконки мобильного приложения, прислушайтесь к этим советам. Они помогут вам избежать ошибок, которые подстерегают новичков.

Как дизайнеру, вам важно обратить внимание на приложение: ведь часто пользователи принимают решение о выборе приложения, руководствуясь исключительно внешним видом иконки в маркете. Продуманная иконка ассоциируется с качественным приложением и удобным интерфейсом, а сделанное на скорую руку изображение не внушает доверия.

Поэтому используя макеты плотности экрана создайте несколько иконок разных размеров, для разных экранов. За отправную точку следует брать экраны с MDPI плотностью, с дальнейшим увеличением в пропорция равных 1,5x для HDPI, 2x для XHDPI, 3x для XXHDPI и 4x для XXXHDPI. Исходный размер должен быть равен 48dp, в результате на экранах с плотность MDPI размер значка будет равен 48x48 точек, для экранов HDPI – 72x72 точек, для XHDPI – 96x96 и т.д. На рисунке 1.12 приведены примеры иконок приложений системы iOS

Рисунок 1.12 – Примеры иконок приложений

1.2.2 Принципы навигации

Пользователь управляет смартфоном или планшетом в основном с помощью сенсорного экрана. Поэтому компании придумали специальные жесты, которые упрощают работу со смартфоном. Жесты можно грубо подразделить на два типа: включенные в систему изначально, к примеру, кнопка "Домой", и работающие только при открытии конкретного приложения. Тем не менее, чтобы не нагружать и не путать пользователя, набор жестов заложен в систему стандартный. К ним можно отнести активация приложения или элемента при одинарном нажатии, выбор объекта при длительном нажатии, движение пальца вбок - "свайп", или переключение на соседнюю вкладку или экран, двойное увеличение экрана или картинки при двойном нажатии, и обычное увеличение с использованием двух соединяющихся или разъединяющихся пальца.

При проектировании интерфейса мобильного приложения важно учитывать правильность построения навигации, так как это одна из ключевых задач разработчика. Необходимо обращать внимание на то, чтобы навигация совпадала с типичным поведение ОС, к которой уже привык пользователь, и которая не будет вводить его в заблуждение. К примеру, кнопка "назад" должна действовать определенным образом: с главного экрана приложения она должна позволить пользователю закрыть приложение, с 1 уровня вкладок переносить на главный экран приложения, соответственно, с последующих уровней вкладок переносить на предыдущие.

1.2.3 Компоненты приложений

Обычно приложения состоят из стандартных блоков, компонентов. Все компоненты можно разделить на 4 основных вида:

-Activity
- Intent Receiver
- Service
- Content Provider

Конечно, не всегда используются все четыре типа, но их комбинация присутствует во всех приложениях.

Компоненты помещаются в файл AndroidManifest.xml, где определяется возможности и функции каждого компонента.

Самым распространенным компонентом является Activity, который является единственным экраном приложения. При этом компонент является одним классом, каждый из которых расширяет базовый класс. Он отображает интерфейс и отвечает на события. Интерфейс составляется из Views.

Обычно у каждого приложения есть несколько экранов, каждый из которых является новым Activity, а переход с одного экрана на другой осуществляется путем старта нового Activity и завершения старого. Иногда одно Activity возвращает другое, к примеру, пользователь выбирает фотографию - один компонент, а второй возвращает выбранную фотографию нужной программе.

При открытии нового экрана, старый помещается на стек хронологии и ставится на паузу. Если пользователь желает вернуться назад, то старый экран вновь перемещается из хронологии на главный экран. Также есть возможность удалить экран из стека при его ненадобности.

Intent и его фильтры.

Данный компонент позволяет двигаться от экрана к экрану. Он также описывает действие приложения. Таким образом, двумя частями компонента являются действие и данные к действию. Действиями считаются main, view, pick, edit и др. Данные к действию - Uniform Resource Indicator (URI). То есть, при открытии веб-сайта действием будет VIEW, а данными к действию - адрес сайта.

Также есть класс, который связан с Intent - IntentFilter. Пока основной компонент делает запрос на действие, IntentFilter описывает, что компонент способен к обработке.

С помощью компонента Intent осуществляется навигация экрана. Чтобы переместиться вперед, Activity вызывает startActivity (myIntent). Система тогда смотрит на IntentFilter для всех установленных приложений и выбирает Activity, Intent которого фильтрует myIntent. Новому Activity сообщают о Intent, которое заставляет его начаться. Процесс решения Intent происходит, когда startActivity вызывают.

Данный процесс дает два ключевых момента: благодаря запросу Intent, действия могут использовать функциональные возможности других компонентов; при этом они могут заменяться новыми Activity, вне зависимости от времени, единственное "но" - IntentFilter должен быть эквивалентным.

Intent Receiver.

Данный инструмент позволяет коду действовать согласно реакции на внешнее событие, к примеру, связанное со временем или определенными условиями. Также он может показывать Уведомления. Запуск приложения может в свою очередь контролироваться данным инструментом, выбирая особые условия для этого. К примеру, проигрыватель, запускающий песни из плейлиста. Типичными действиями для него будут выбор и проигрыш песен. Однако пользователь ожидает большего, например, чтобы даже после сворачивания приложения музыка не останавливалась. Данную функцию можно осуществить с помощью данного компонента и кода Service.

Service – код, который долговечен и выполняется без UI. Хороший пример этого – универсальный проигрыватель, запускающий песни из плейлиста. В приложении универсального проигрывателя, вероятно, были бы одно или более Activity, которые позволяют пользователю выбирать песни и запускать их. Однако, воспроизведение самой музыки не должно быть обработано Activity, потому что пользователь будет ожидать, что музыка продолжит играть даже после сворачивания проигрывателя. В этом случае, деятельность универсального проигрывателя могла запустить Service, используя Context.startService(), чтобы работать на заднем плане и сохранить воспроизведение музыки. Тогда система сохранит воспроизведение музыки, пока оно не закроется само. (Вы можете узнать больше о приоритете, данном службам в системе, читая Цикл Жизни Приложения Андроида). Отметьте, что Вы можете соединиться с Service (и запустить его, если он уже не работает) с методом Context.bindService(). Когда есть подключение с Service, Вы можете общаться с этим через интерфейс, выставленный Service. Для Service музыки это могло бы позволить Вам приостанавливать, перематывать, и т.д.

Content Provider.

Данные приложения хранятся в разных ресурсах: в файлах, базе данных SQLite, персональных настройках или любом другом механизме, который

имеет смысл. Инструмент Content Providerи дает возможность разделять данные приложения с другими приложениями.
Класс саомощ стандартных методов позволяет другим приложениям сохранять и восстанавливать тип данных, обработанных другим таким же классом.

2 Инструменты разработки

2.1 Основной инструмент разработки – фреймворк. Концепция MVC

MVC Фреймворк является программной платформой. С помощью него определяется структура программной системы; программное обеспечение, которое позволило бы упростить разработку и объединить все части большого проекта.

Русскоязычным аналогом слова "фреймворк" является слово "каркас", многие даже используют его как основной термин, отказываясь от английского термина.

Здесь же стоит упомянуть каркасный подход, когда конфигурация состоит из двух частей: постоянной, или неизменный каркас с гнездами, и переменной, или сменных модулей, которые и размещаются в гнездах.

Не стоит путать фреймворк с библиотекой. Последняя имеет возможность быть использованной как набор подпрограмм с близкой функциональностью, при этом она не может влиять на архитектуру конечного продукта и не может накладывать на нее ограничения.

Фреймворк в свою очередь напрямую влияет на архитектуру приложения и диктует ей свои правила. Поэтому его еще называют каркасом, так как он задает тон с самого начала разработки, показывает, что нужно расширять и согласно каким требованиям.

Пример фреймворка - CMF (Content Management Framework). Примером библиотеки может служить модуль электронной почты.

Каркасные приложения имеют стандартную структуру, что можно отнести к их преимуществам. Популярность к ним пришла с возникновением графических интерфейсов, которые реализовывались на основе стандартной структуры. Фреймворки облегчили данный процесс, так как структура года известна и понятна заранее.

Каркас строится с помощью объектно-ориентированного программирования и его техник.

Model-view-controller, или MVC — схема использования нескольких шаблонов проектирования, которая разделяет модель приложения, пользовательский интерфейс и взаимодействие с пользователем на три отдельных компонента. Так, модификация одного из компонентов оказывает

минимальное воздействие на остальные. Подобная схема используется в создании архитектурного каркаса при переходе от теории к реализации в выбранной предметной области. Концепция MVC показана на рисунке 2.1

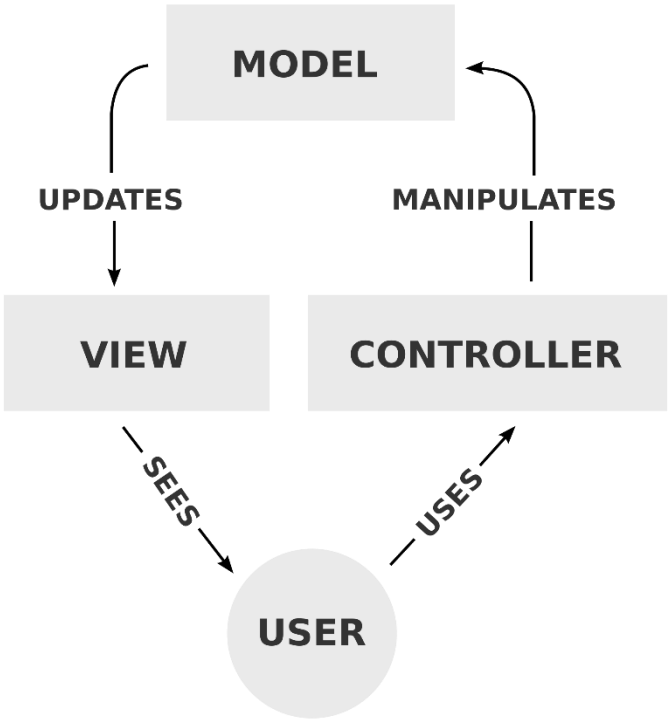

Рисунок 2.1 – Концепция MVC

Контроллер (Controller) управляет запросами пользователя (получаемые в виде запросов HTTP GET или POST, когда пользователь нажимает на элементы интерфейса для выполнения различных действий). Его основная функция — вызывать и координировать действие необходимых ресурсов и объектов, нужных для выполнения действий, задаваемых пользователем. Обычно контроллер вызывает соответствующую модель для задачи и выбирает подходящий вид.

Модель (Model) - это данные и правила, которые используются для работы с данными, которые представляют концепцию управления приложением. В любом приложении вся структура моделируется как данные, которые обрабатываются определённым образом. Что такое пользователь для приложения — сообщение или книга? Только данные, которые должны быть обработаны в соответствии с правилами (дата не может указывать в будущее, e-mail должен быть в определённом формате, имя не может быть длиннее X символов, и так далее).

Вид (View) обеспечивает различные способы представления данных, которые получены из модели. Он может быть шаблоном, который заполняется данными. Может быть несколько различных видов, и контроллер выбирает, какой подходит наилучшим образом для текущей ситуации.

Веб приложение обычно состоит из набора контроллеров, моделей и видов. Контроллер может быть устроен как основной, который получает все запросы и вызывает другие контроллеры для выполнения действий в зависимости от ситуации.

2.2 Анализ существующих на рынке платформ для разработчиков (фреймворков)

Фреймворк Appcelerator Titanium и его свойства:

1) Приложения смотрятся и проявляют себя как нативные, хотя вполне прописаны с применением JavaScript (js код в рантайме транслируется в нативные view).

2) Данный UI можно делать в отдельности для любой платформы с применением фреймворка Alloy (встроенный MVC фреймворк, использующий XML и CSS-подобный синтакс). Не взирая, собственно, на существование отдельного UI для любой ОС усложняет исследование и сильно сокращает размер переиспользуемого кода, вся бизнес-логика, модель и ядро прибавления остаются схожими для каждой платформы.

3) Хранилище плагинов и компонентов (520 компонентов и 204 из них бесплатные) – все, что только может понадобится для приложений: аналитика, реклама, облачные хранилища, социальные сети, работа с графикой и др.

4) Мониторинг данных о приложении в режиме реального времени, мониторинг производительности, крэшей, логов и даже самого процесса создания приложения.

5) Коннекторы встроены к самым популярным enterprise-платформам (Salesforce, SAP, Oracle, Microsoft Dynamics и SharePoint), коннекторы к популярным приложениям (LinkedIn, PayPal, DropBox, Facebook, Twitter и др.). Возможность создавать свои собственные коннекторы к любым сервисам.

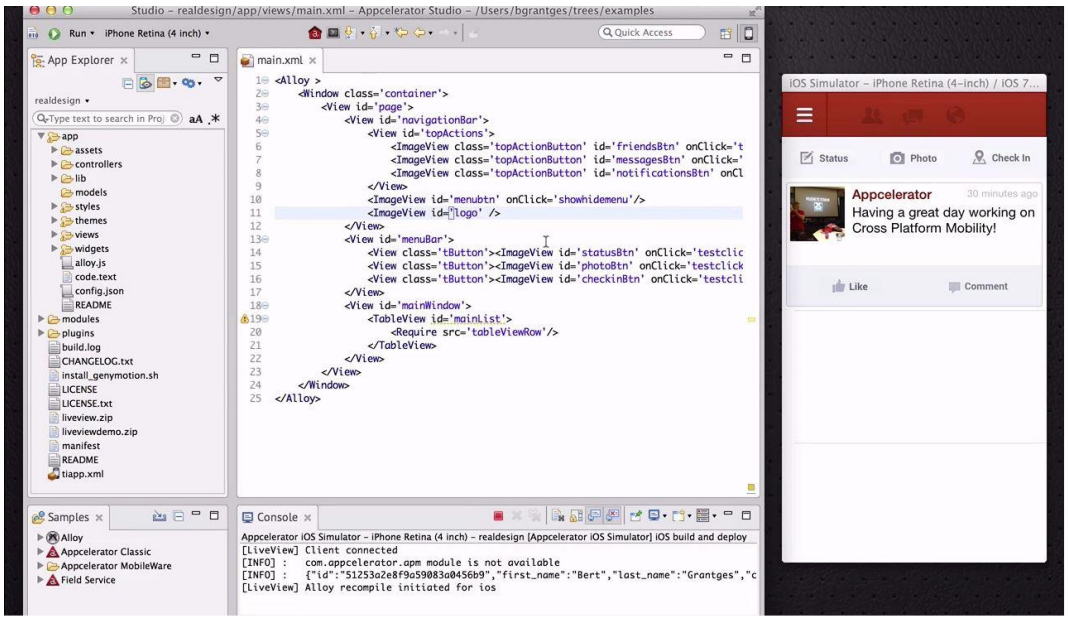

Рисунок 2.2 – Пример создания приложения в Titanium IDE

Фреймворк PhoneGap и его свойства:

1) Приложение работает как обычная веб-страница внутри WebView, соответственно все строится на основе всем знакомых HTML, CSS и JS. Но при этом PhoneGap API дает нам возможность использовать все возможности устройства в приложении: камера, звук, GPS, файловая система, контакты, уведомления и т.д.

2) В PhoneGap приложение строится на основе WebView, его можно внедрить в нативное приложение (получим гибридное приложение).

3) Компиляция под любую существующую мобильную платформу, включая Tizen, Bada, Firefox OS. Причем сделать это можно буквально в пару кликов, используя облачный сервис PhoneGap Build.

4) Небольшой и простой, поэтому порог вхождения в разработку на PhoneGap довольно низок, достаточно знать лишь основные веб-технологии.

5) Бесплатный продукт с открытым кодом.

Рисунок 2.3 – Сайт фреймворка PhoneGap

Фреймворк Xamarin Studio и его свойства:

1) Проектирование нативных iOS, Android, Mac и Windows приложений с помощью языка C#. Причем, в отличие от Appcelerator, код не интерпретируется на стадии выполнения, а компилируется сразу в нативный код. Поэтому поведение, вид и производительность такая же как и у родных приложений!

2) Данный UI создается для каждой платформы с помощью стандартных для этих платформ view.

3) Сервис Xamarin Test Cloud для автоматизированного тестирования приложения на сотнях виртуальных мобильных устройствах.

4) Большой выбор компонентов (плагинов) для расширения возможностей Xamarin.

5) Встроенные покупки для приложений (in-app purchases).

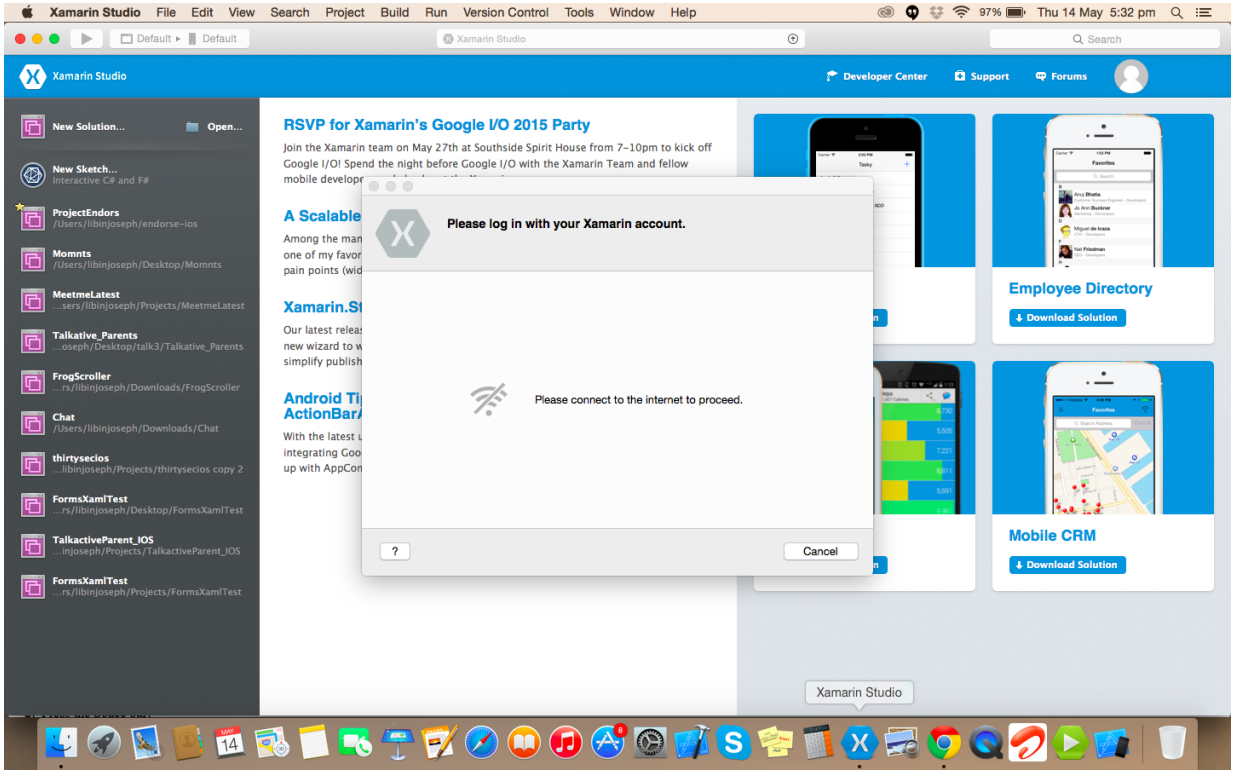

Рисунок 2.4 – Внешний вид Xamarin Studio

2.3 Реляционная база данных для кроссплатформенного мобильного приложения

Совсем недавно вышла новая версия библиотеки, которая будет полезна C# разработчикам, разрабатывающим или планирующим разрабатывать кроссплатформенные мобильные приложения.

SQLitePCL может быть использована для реализации локальной базы данных в приложениях для Windows, Windows Store, Windows Phone, Android (Xamarin) и iOS (Xamarin). Она бесплатна и ее код открыт для всех желающих.

Для экспериментов нам понадобится проект универсального приложения для Windows Phone. На рисунке 2.5 показано создание проекта для Windows Phone

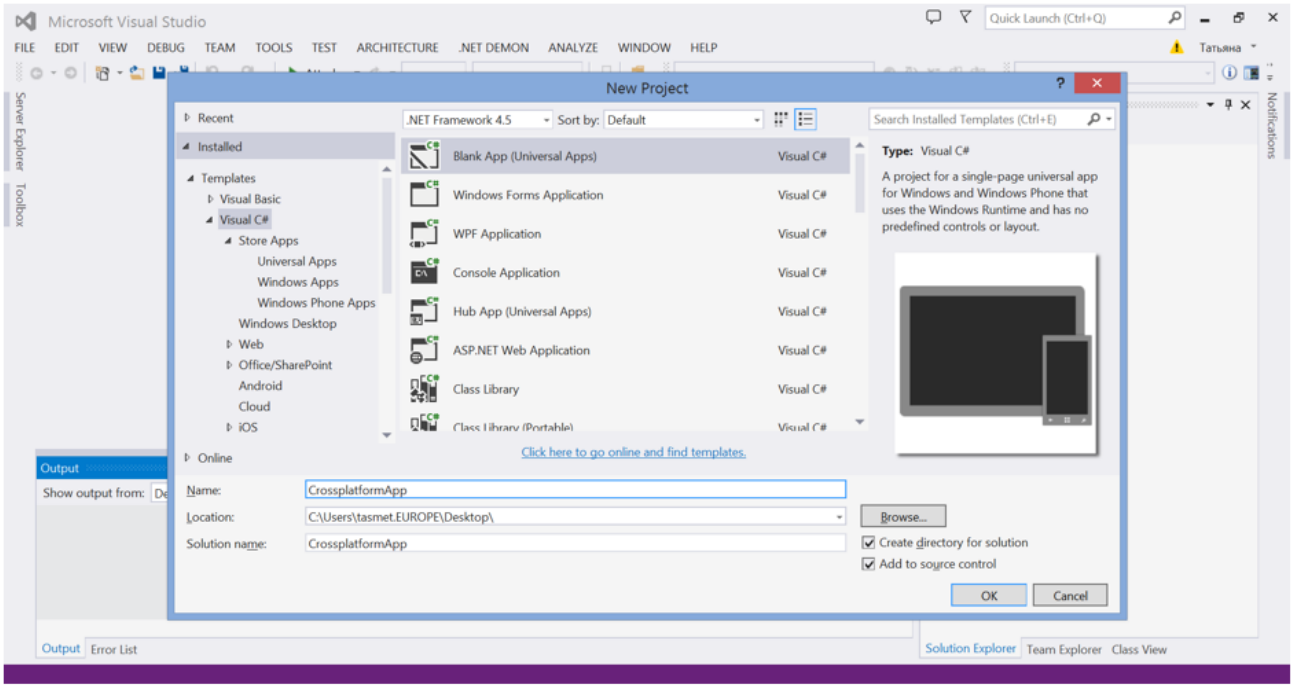

Рисунок 2.5 – Создание проекта для Windows Phone

И проекты Xamarin для Android и iOS приложений на рисунке 2.6

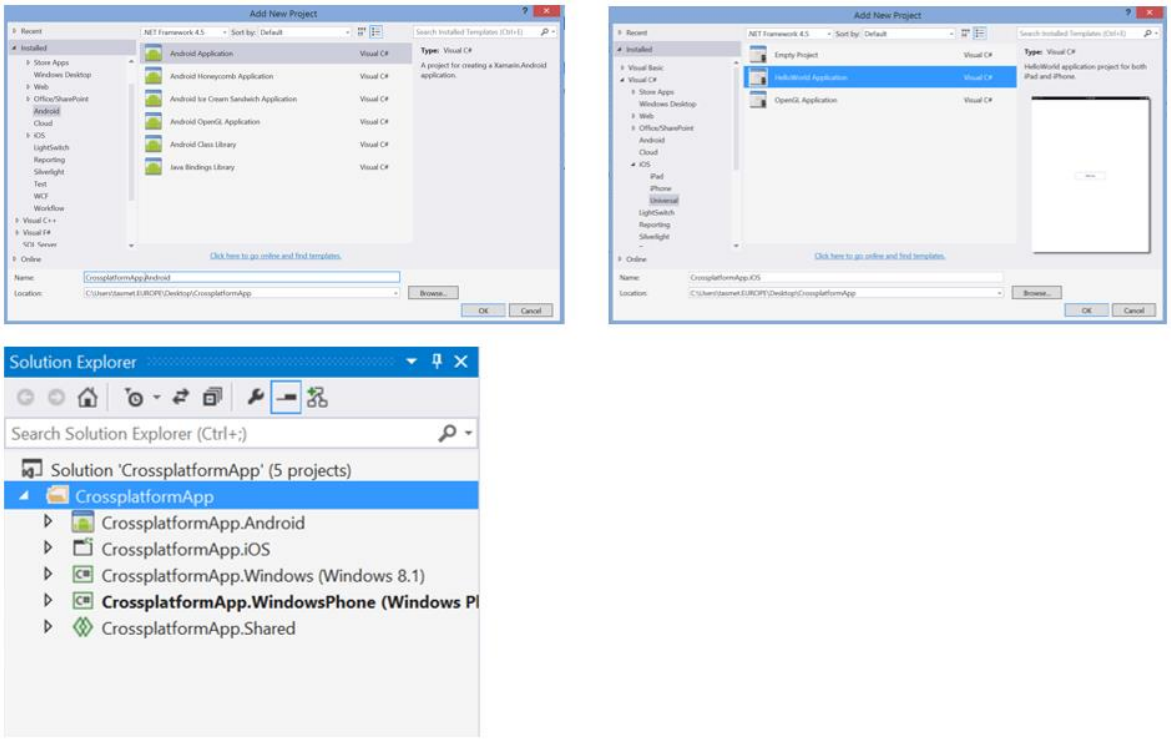

Рисунок 2.6 – Проекты для Andoid и iOS в Xamarin Studio

Для каждого проекта устанавливаем и добавляем в References пакет SQLite-net PCL. Он автоматически установит и SQLitePCL.raw. Это всё, что нам надо будет для использования. Подключаем необходимые библиотеки (Рисунок 2.7)

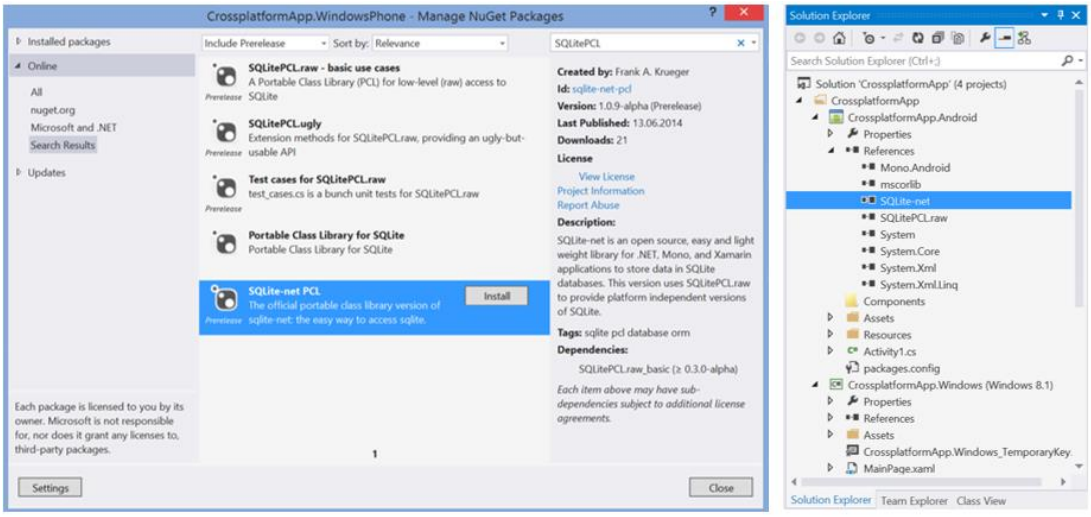

Рисунок 2.7 – Подключение библиотек, добавление в References пакета SQLite-net PCL

Мы будем реализовывать логику работы с данными в уже имеющемся у нас, общем для всех приложений, Shared проекте. Но вы можете делать это и в новой, созданной для этих целей Portable Class Library. Итак, добавим в Shared проект два файла. Один из них MessageItem.cs — будет содержать структуру объекта, который мы будем хранить в базе данных, второй DataBase.cs – реализовывать взаимодействие с БД SQLite. Создаем БД и реализуем CRUD операции. (Рисунок 2.8)

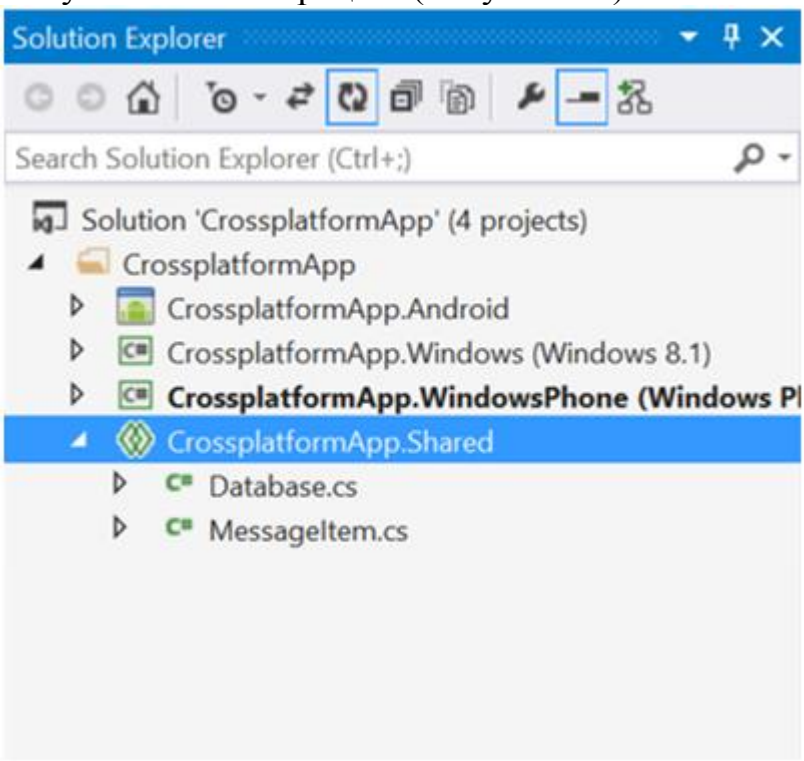

Рисунок 2.8

Заменим весь XAML код в MainPage.xaml, чтобы сделать простой интерфейс приложения и отображать данные из базы пользователю. (Рисунок 2.9)

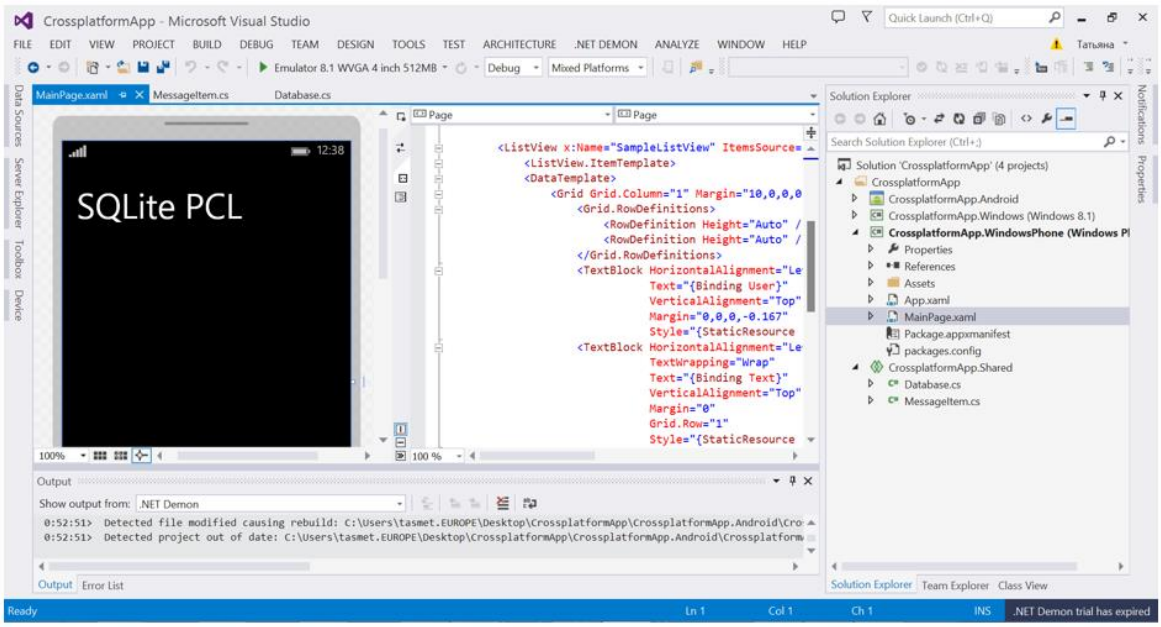

Рисунок 2.9

Напишем код приложения. Создадим БД, наполним данными и выведем их на экран. (Рисунок 2.10)

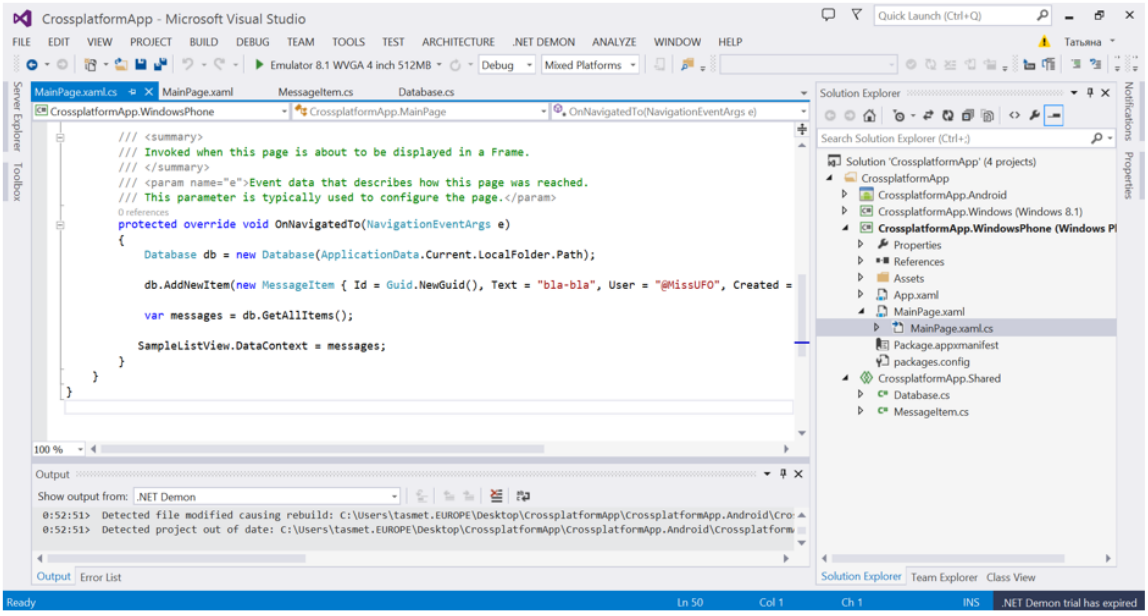

Рисунок 2.10

Запустим проект и посмотрим на результат. БД создалась успешно, данные записываются и читаются. (Рисунок 2.11)

Рисунок 2.11

Для Android и iOS процедура выглядит в точности так же. Вы можете подключить Shared проект в качестве References и пользоваться нашими классами.

3 Этапы разработки приложения

3.1 Проектирование

Проектирование кроссплатформенного мобильного приложения является непростой задачей, выполняется разработчиком самостоятельно и состоит из нескольких этапов. Помимо создания функционала нашего приложения, проектирование включает в себя так же базу данных и пользовательский интерфейс.

Процесс разработки проводится с помощью IDE Xamarin Studio на современном объектно-ориентированном языке программирования C#. Для начала работы требуется установить выбранную технологию (рисунок 3.1):

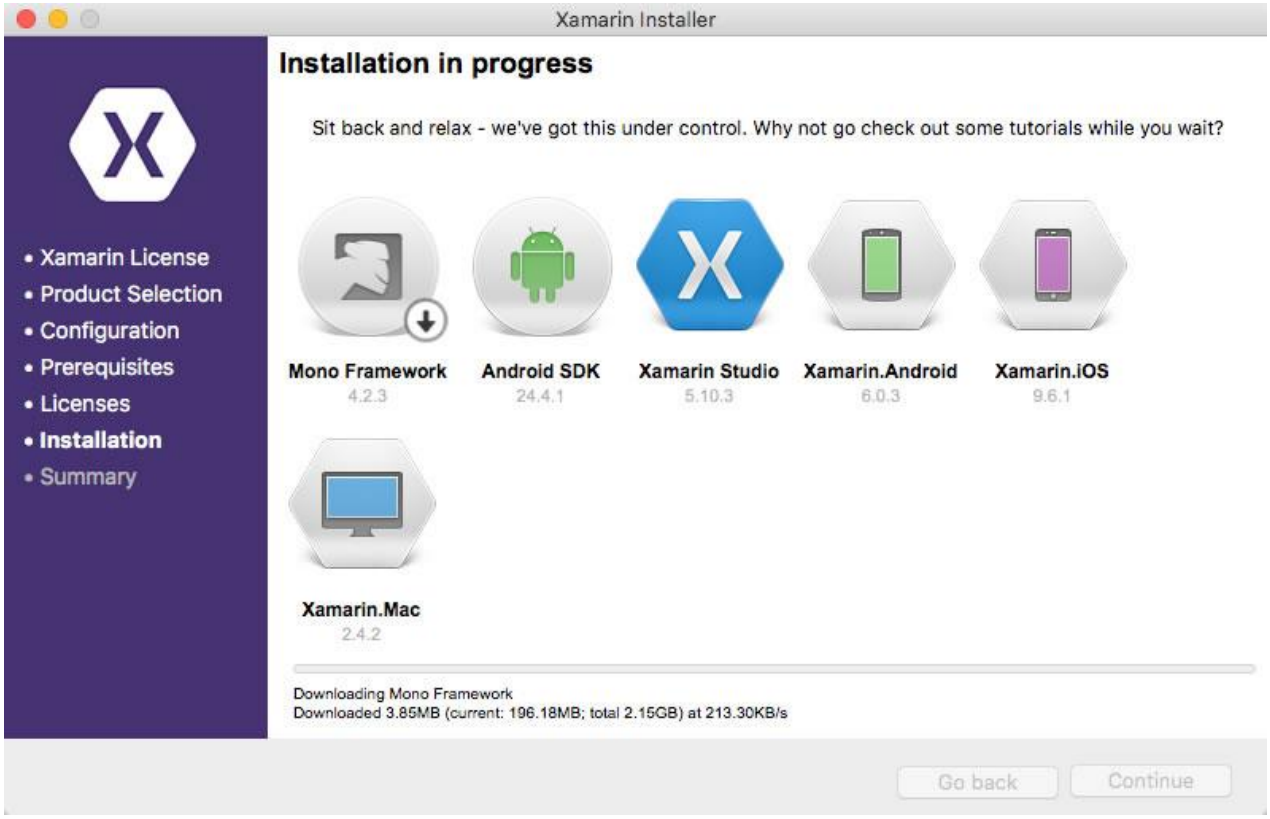

Рисунок 3.1 – Установка Xamarin Studio

Установка состоит из нескольких пунктов, в которых указывается путь установки, выбирается тип установки – полный либо частичный, указываются библиотеки, которые нужны для полнофункциональной работы для создания мобильного приложения. После установки, следует скачать утилиты для корректной работы Xamarin Studio, такие как MonoTouch фреймворк и Visual Studio последней версии.

Следующий пункт – это регистрация на портале GitHub, основная часть изображена на рисунке 3.2. Данное действие необходимо для хранения бэкапов проекта на сервере портала, также на форуме данного портала можно найти множество подсказок и полезных статей по проектированию различных работ для мобильного приложения.

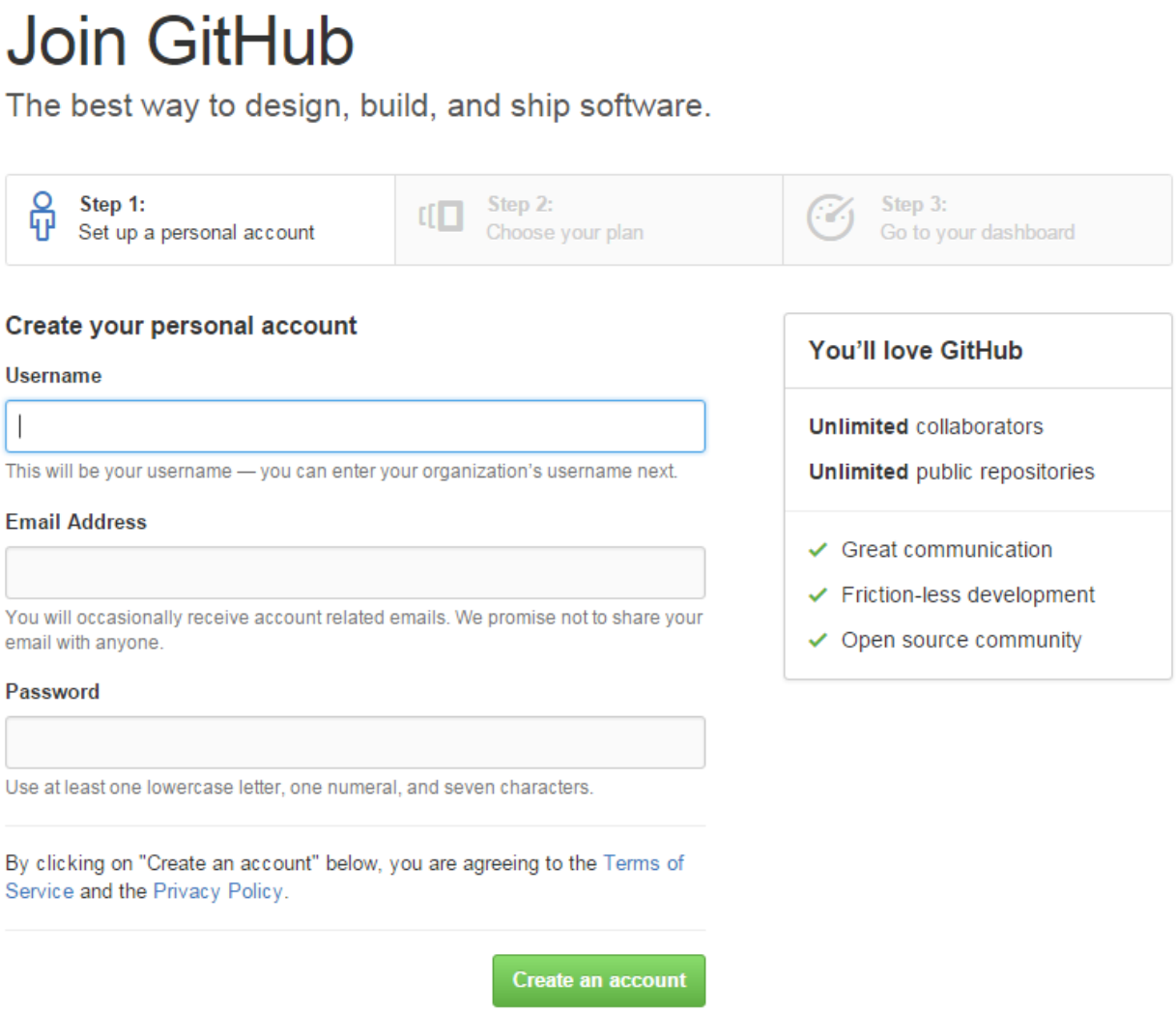

Рисунок 3.2 – Регистрация в GitHub

После регистрации появляется возможность связать свою работу с компьютера разработчика с аккаунтом на GitHub, и проект будет хранится в хранилище, как показано на рисунке 3.3. В нужный момент разработчик может получить доступ к своему проекту на сервере, даже если у него отсутствует личный ПК, разработчик может зайти на аккаунт портала и скачать собственный проект на иной переносной либо персональный компьютер.

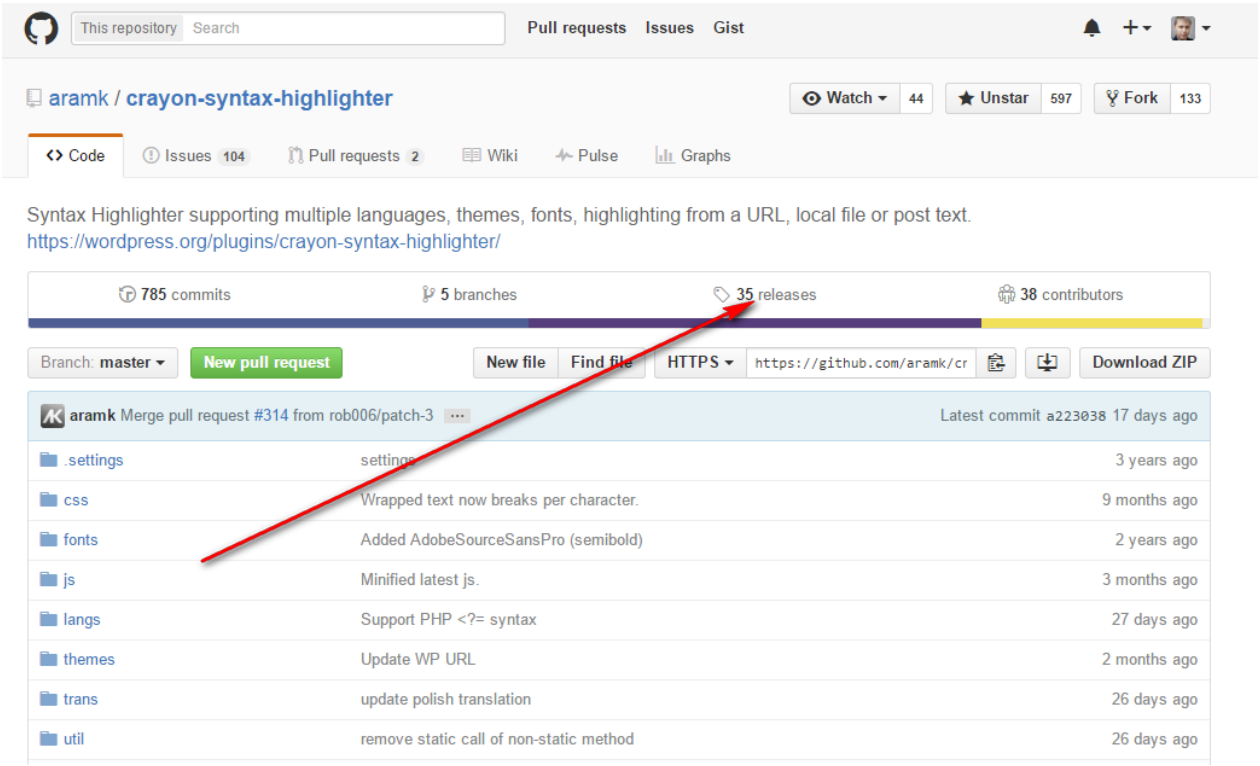

Рисунок 3.3 – Хранилище проектов на GitHub

3.2 Разработка интерфейса приложения

Начало работы в Xamarin Studio, создаем новый проект, даем название и на экране выводится главное окно программы, которое включает в себя панель управления, панель инструментов, панель отладки, панель структуры приложения и эмулятор смартфона по центру экрана. Все вышеперечисленное отображено на рисунках 3.4 и 3.5

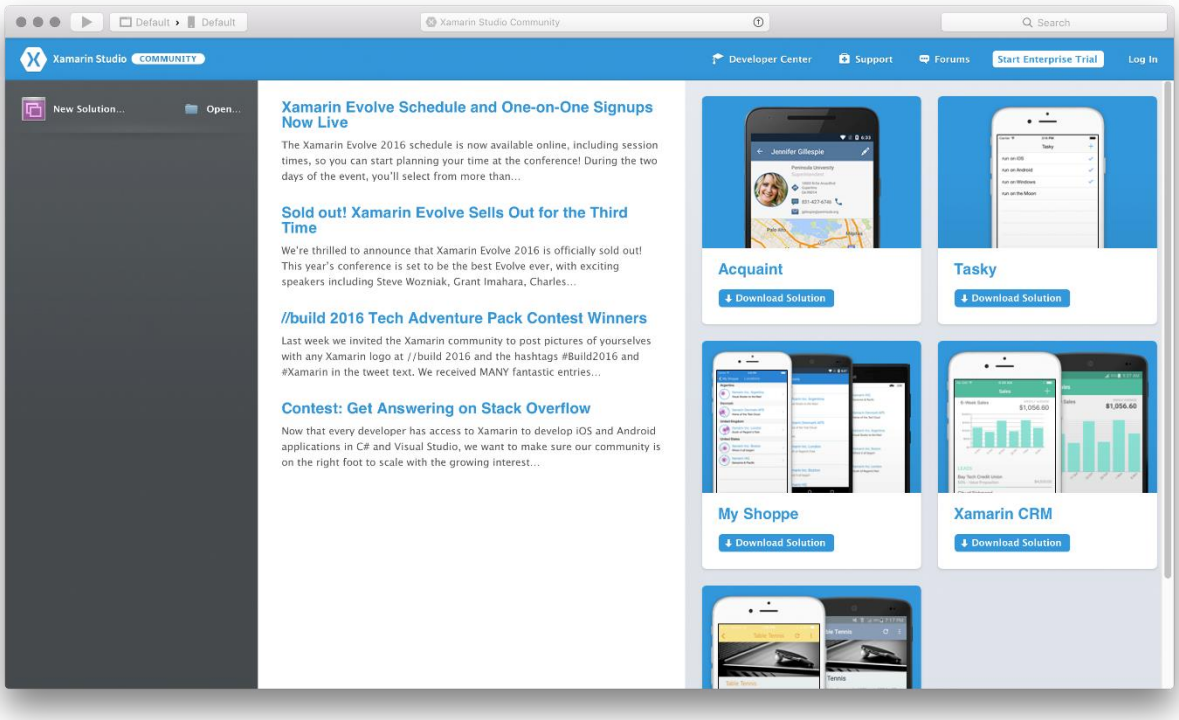

Рисунок 3.4 – Начало работы в Xamarin Studio

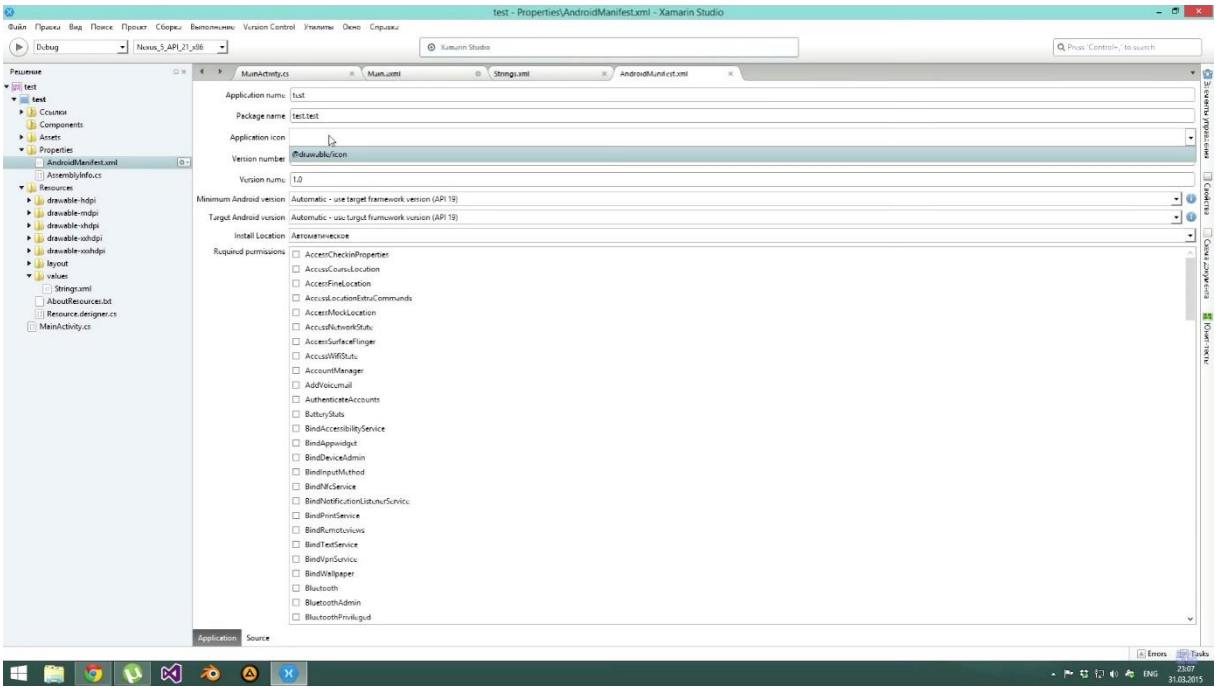

Рисунок 3.5 – Создание тестового приложения

Начало разработки состоит из написания кода тестового приложения и подключения библиотек jdk и sdk;

Все выше перечисленное изображено на рисунке 3.5 и 3.6, на котором показана часть кода

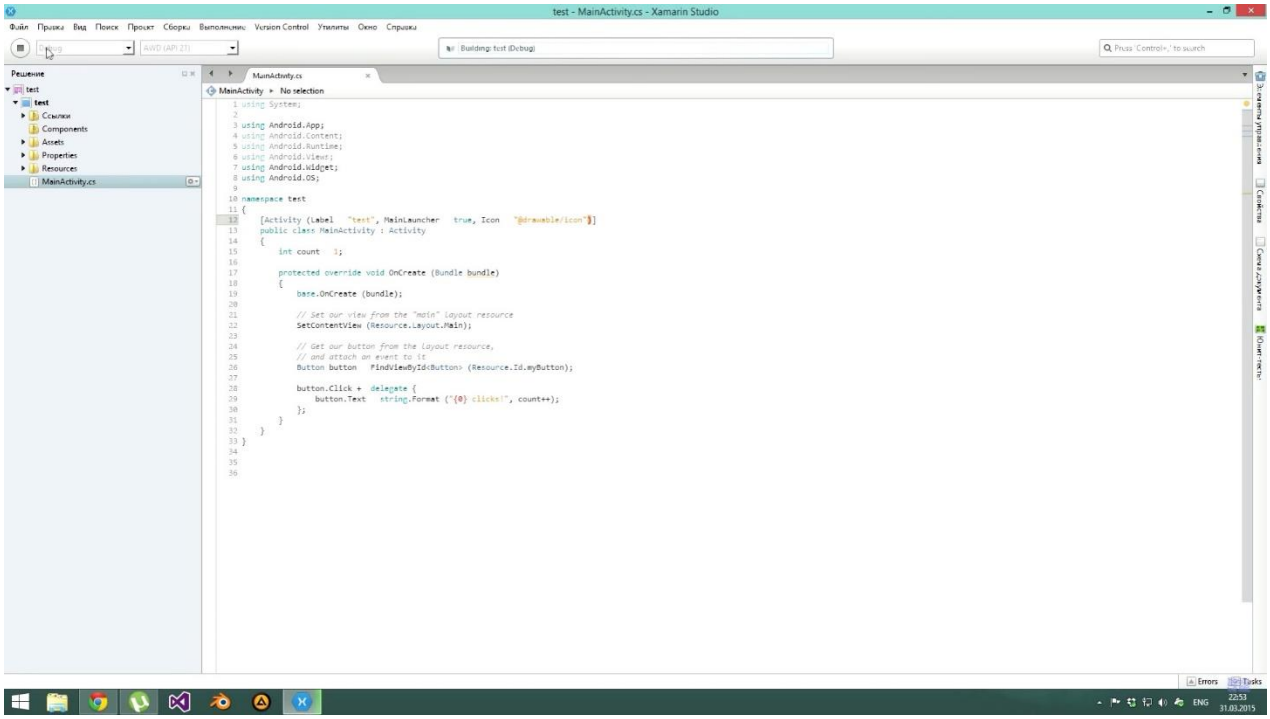

Рисунок 3.5 – Код первоначального приложения

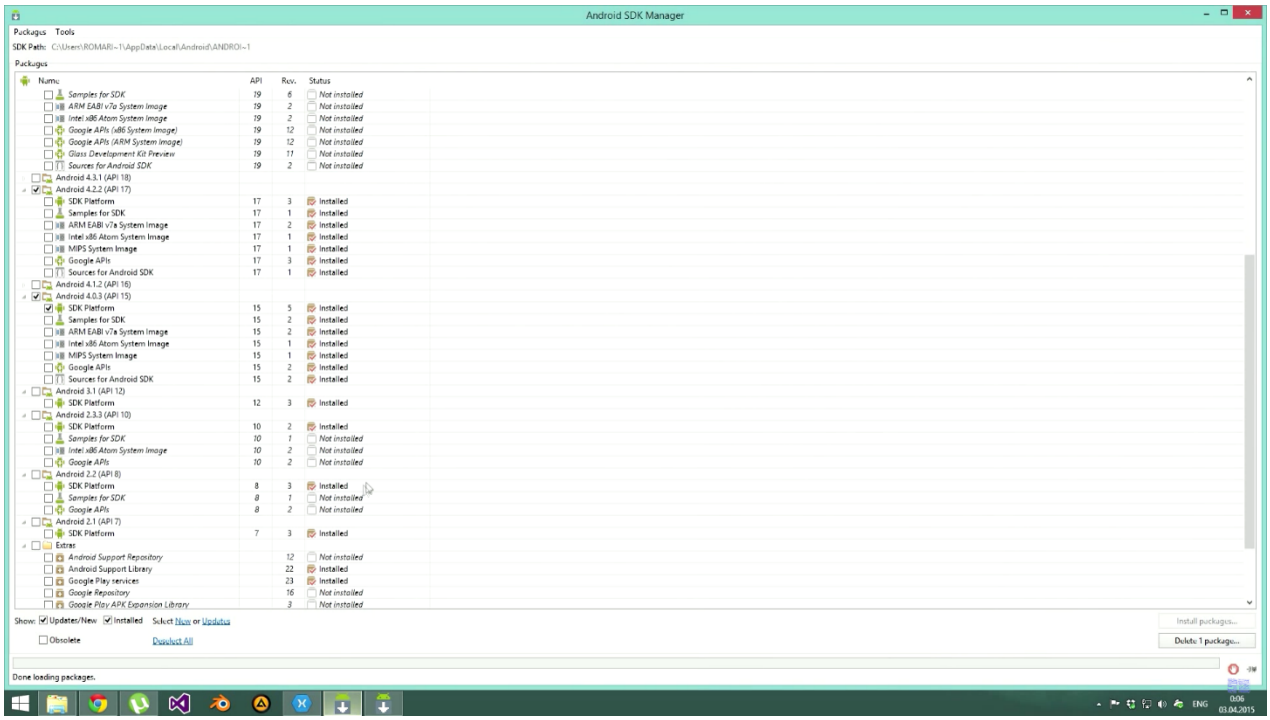

Рисунок 3.6 – Подключение SDK

На приложенном рисунке 3.6 видно, что SDK Manager уже встроен в IDE Xamarin Studio

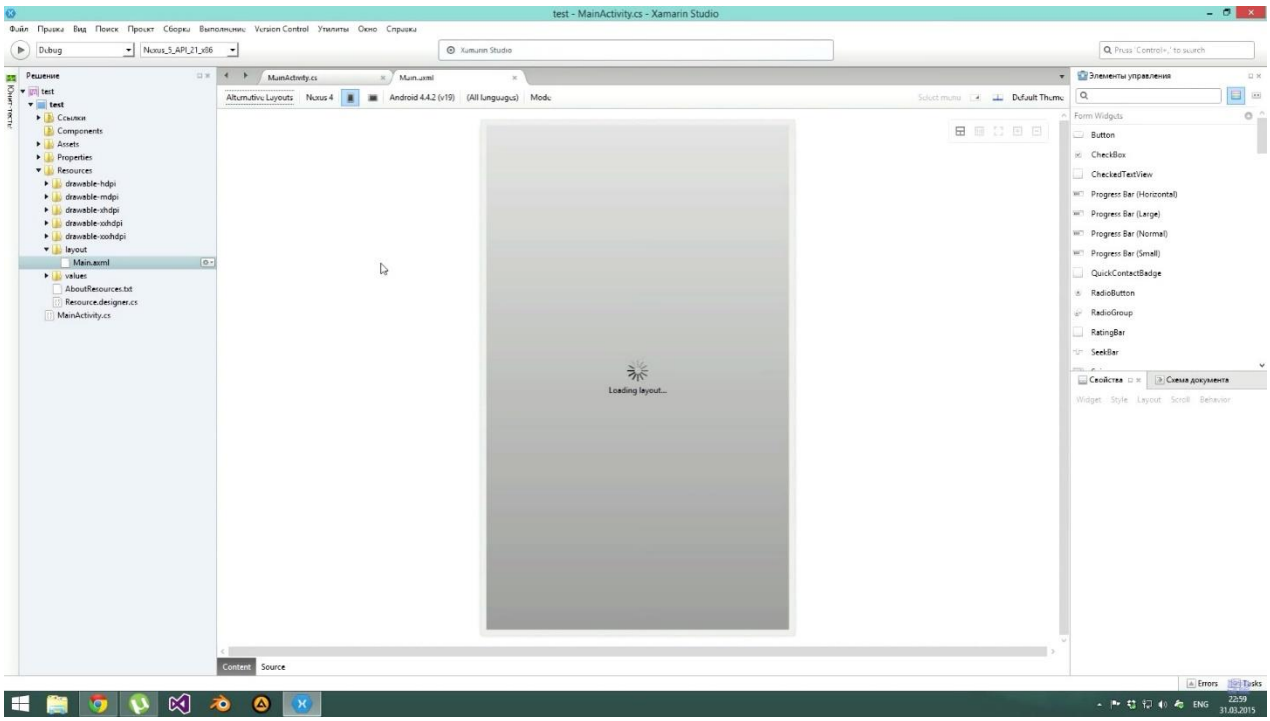

Рисунок 3.7 – Создаем новый layout для стартовой страницы

Рисунок 3.7 демонстрирует создание нового слоя для главного меню будущего приложения

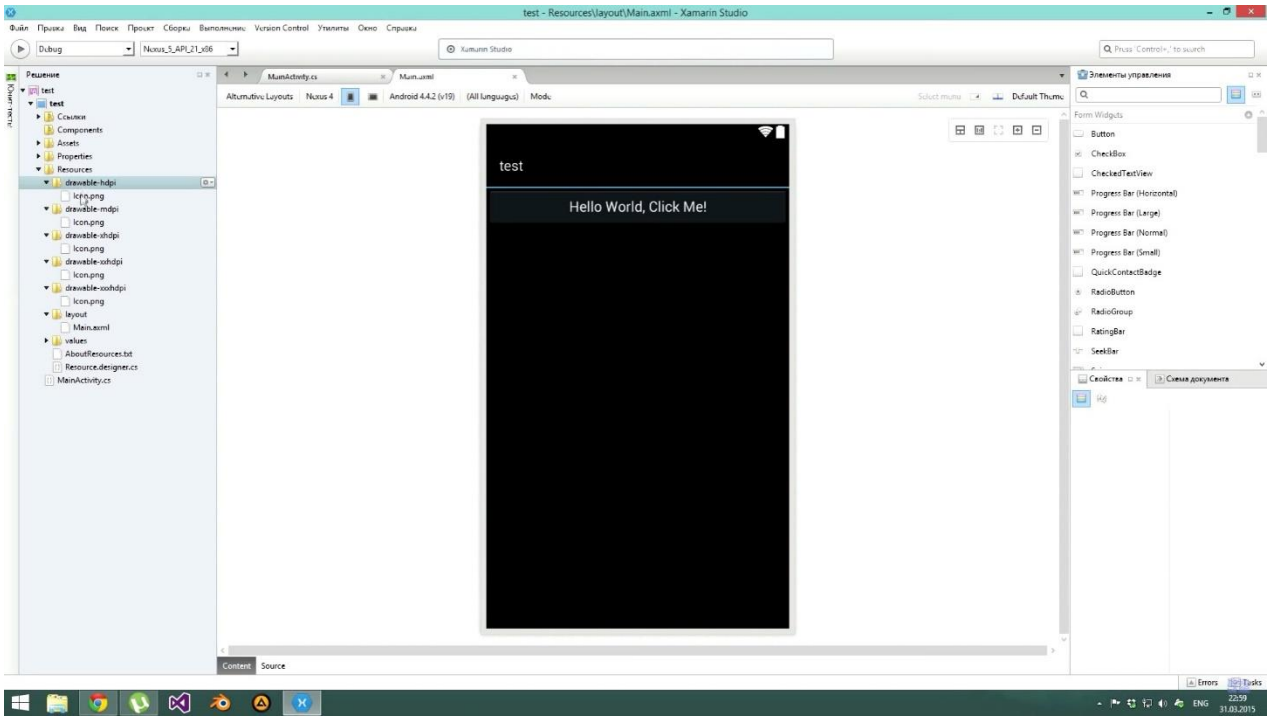

Рисунок 3.8 – Новый слой создан

3.3 Результаты работы приложения

По итогам работы создана часть мобильного приложения, отвечающая за предоставление списка задач, к которым можно добавить новое задание и присвоить ему статус - завершено. Эта часть доступна в готовом проекте, как показано на рисунке 3.8

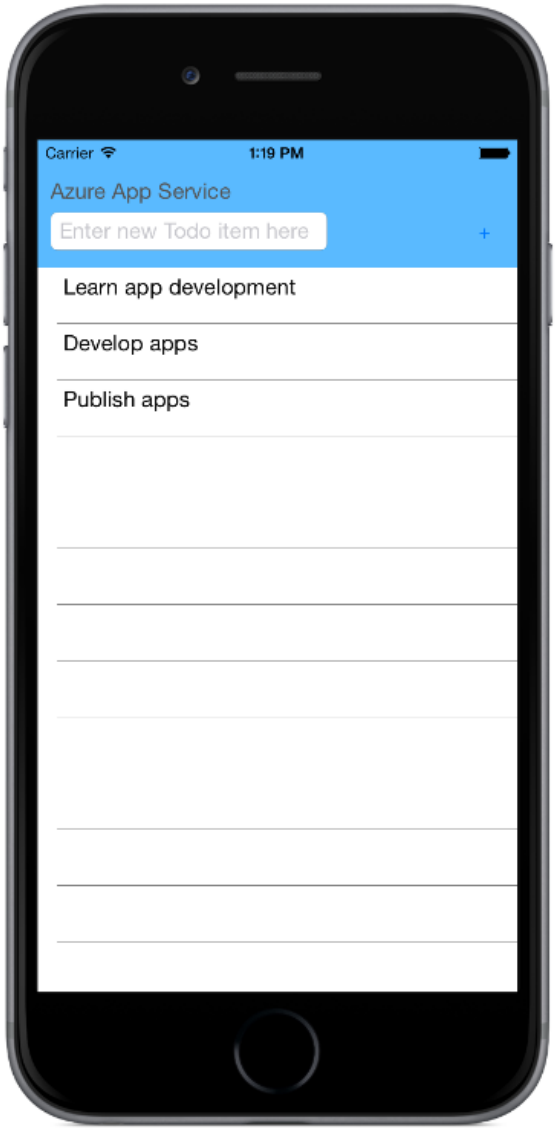

Рисунок 3.8 – Главное меню для ОС iOS

Главное меню включает в себя 5 разделов, которые доступны пользователю. Верхний textbar отвечает за добавление нового задания. Ниже представлены невыполненные еще задания. Ниже можно добавить подробное их описание.

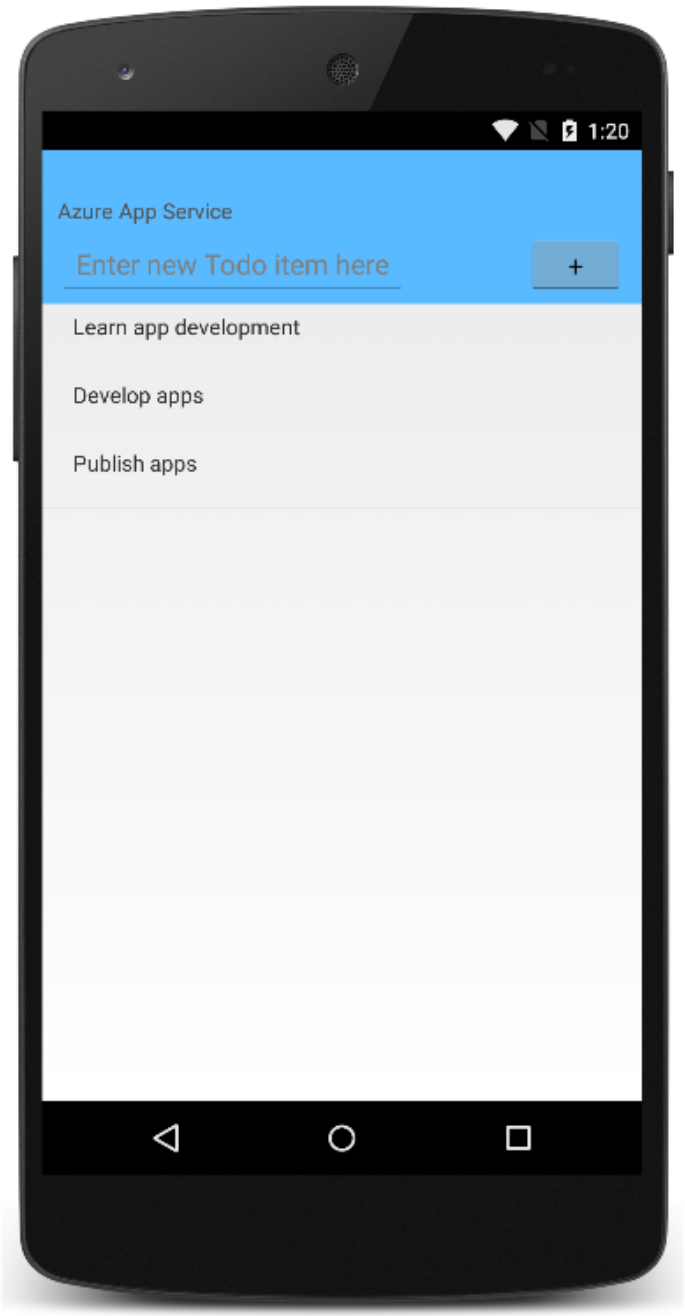

Рисунок 3.9 – Главное меню для ОС Android

Как меняется интерфейс в зависимости от выбранной ОС показано на рисунке 3.9.

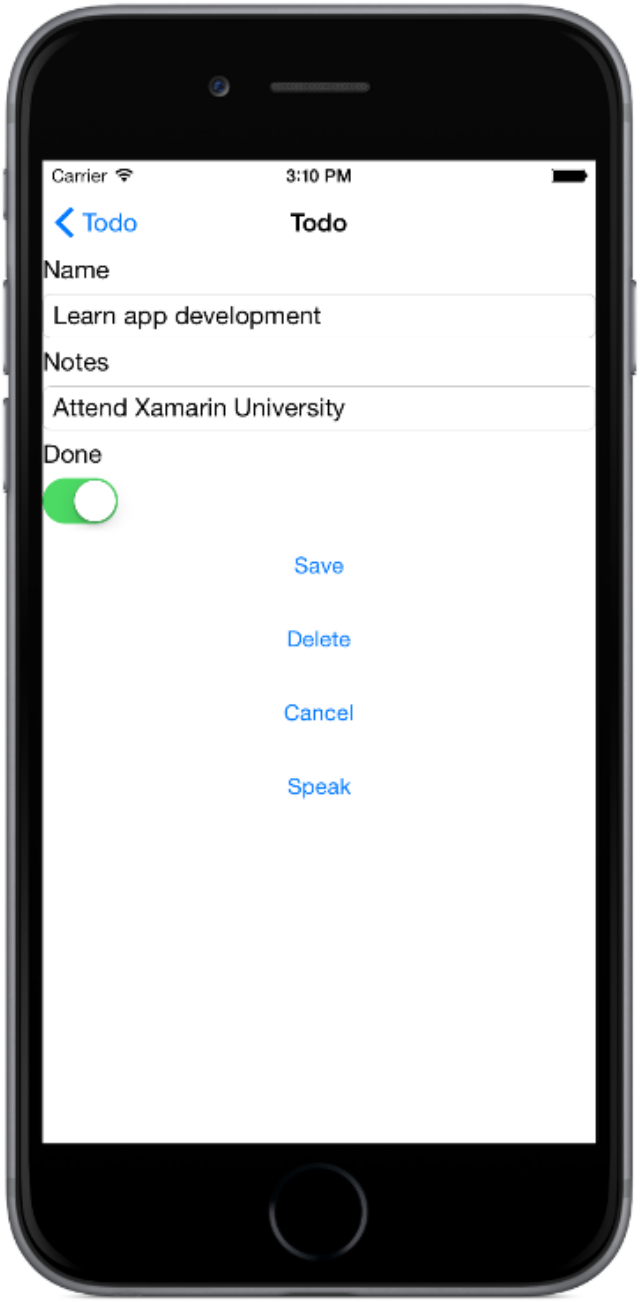

Рисунок 3.10 – Переход на экран создания новой заметки

Как видно на рисунке 3.10, пользователь сам может создавать новые задания, а также редактировать их, сохранять и удалять старые. Также существует смена статуса текущего задания.

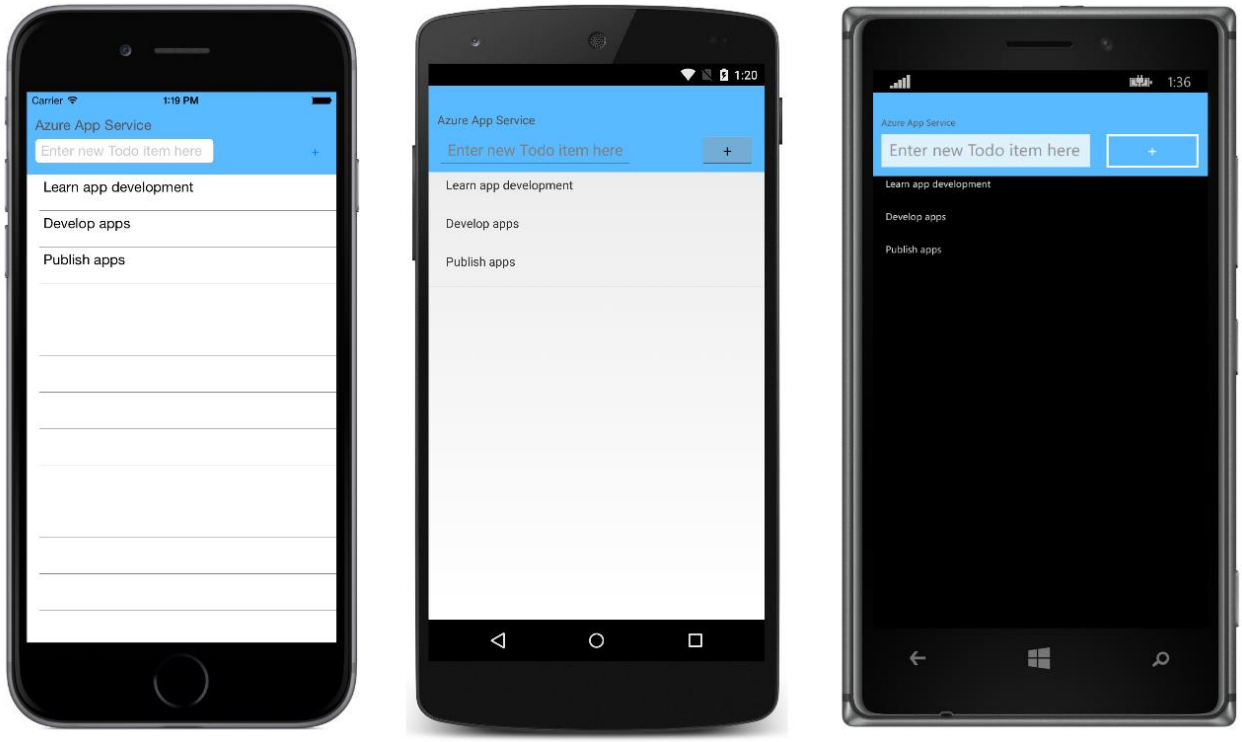

Рисунок 3.11 – Интерфейсы приложения на iOS, Android, Windows Phone

Приложение создано и работает на трех платформах, как и показано на рисунке 3.11.

В этой главе была описана подготовка к разработке, сама разработка и принцип работы кроссплатформенного приложения. Описание подкреплено письменной инструкцией и рисунками с демонстрацией работы мобильного приложения.

4 Технико-экономическая часть

Технико-экономическое обоснование научно-исследовательских работ проводится с целью определения и анализа трудовых и денежных затрат, направленных на их реализацию, а также уровня их научно-технической результативности.

Целью дипломной работы является изучение технологий разработки кроссплатформенных мобильных приложений. В данном разделе приводится рассмотрение экономической составляющей этого проекта, отражающей затраты.

4.1 Определение трудоемкости выполнения НИР

Для определения трудозатратности исполнения НИР сначала был составлен ассортимент всех ключевых рубежей и видов дел, которые обязаны

быть исполнены. При всем при этом специальное внимание было уделено закономерному упорядочению очередности отдельных видов дел и раскрытию вероятностей их параллельного исполнения, что позволило существенно сократить общую длительность проведения НИР.

Трудоемкость проведения работ и исполнитель приведены в таблице 4.1.

Таблица 4.1 - Распределение работ по этапам и видам и оценка их трудоемкости

Этап проведения НИР	Вид работы на данном этапе	Исполнитель	Трудоемкость выполнения НИР, чел.× ч.	Трудоемкость выполнения НИР, дни
1	Постановка задачи	Научный руководитель	20	3
2	Разработка и утверждение технического задания (ТЗ)	Научный руководитель	30	4
3	Подбор и изучение материалов по тематике	Инженер-разработчик	250	32
4	Анализ актуальных технологий	Научный руководитель	40	5
5	Пример приложения	Инженер разработчик	100	13
6	Оформление работы	Инженер-разработчик	60	8
7	Подведение итогов	Научный руководитель	60	8
ИТОГО трудоемкость выполнения проекта			560	73 (3 месяца)

4.2 Расчет затрат на выполнение НИР

Определение затрат на выполнение НИР производится путем составления соответствующей сметы, которая включает следующие статьи:

1) Материальные затраты.
2) Затраты на оплату труда.
3) Отчисления на социальные нужды.
4) Амортизация основных фондов.
5) Прочие затраты.

4.2.1 Расчет материальных затрат

В статью «Материальные затраты» включаются затраты на основные и вспомогательные материалы, энергию, необходимые для выполнения НИР.

Расчет затрат на материальные ресурсы производится по форме, приведенной в таблице 4.2.

Таблица 4.2 - Затраты на материальные ресурсы

Наименование материального ресурса	Единица измерения	Количество израсходованного материала	Цена за единицу, тг	Сумма, тг
Листы А4	упаковка	4	1000	4000
Листы А3	упаковка	1	2 500	2 500
USB накопитель	шт	1	2 000	2 000
ИТОГО затраты на материальные ресурсы				8 500

Расчет затрат на оборудование и программное обеспечение производится по форме, приведенной в таблице 4.3.

Таблица 4.3 - Затраты на оборудование и программное обеспечение

Наименование	Единица измерения	Количество	Цена за единицу, тг	Сумма, тг
Ноутбук HP Pavilion g7	шт	1	140 000	140 000
МФУ HP LaserJet M125a	шт	1	43 000	43 000
Монитор LG	шт	1	100 000	100 000
Microsoft Windows 10	лиц. копия	1	40 000	40 000
Microsoft Office 2016	лиц. копия	1	26 000	26 000
Смартфон Apple Iphone 5s	шт	1	70 000	70 000
Смартфон Samsung Galaxy S4 mini	шт	1	50 000	50 000
ИТОГО затраты на оборудование и ПО				469 000

Общая сумма затрат на материальные ресурсы ($З_М$) определяется по формуле:

$$З_м = \sum_{i=1}^{n} Р_i * Ц_i , \quad (4.1)$$

где $Р_i$ – расход i-го вида материального ресурса, натуральные единицы;

$Ц_i$ – цена за единицу i-го вида материального ресурса, тг;

i – вид материального ресурса;
n – количество видов материальных ресурсов.

4.2.2 Расчет затрат на электроэнергию

Так как при выполнении НИР использовалось электрооборудование, то необходимо было рассчитать затраты на электроэнергию

Расчет затрат на электроэнергию производится по форме, приведенной в таблице 4.2.

Таблица 4.2 - Затраты на электроэнергию

Наименование оборудования	Паспортная мощность, кВт	Коэффициент использования мощности	Время работы оборудования для выполнения НИР, ч	Цена электроэнергии, $\frac{\text{тенге}}{\text{кВт*ч}}$	Сумма, тг
Ноутбук HP Pavilion g7	0,047	0,9	410	27	468
МФУ HP LaserJet M125a	0,465	0,9	80	27	903
Монитор LG	0,046	0,9	410	27	458
Смартфон Iphone 5s	0,015	0,9	100	27	35
Смартфон Galaxy S4 mini	0,02	0,9	100	27	49
ИТОГО затраты на электроэнергию					1 913

Общая сумма затрат на электроэнергию ($З_Э$)рассчитывается по формуле:

$$З_э = \sum_{i=1}^{n} M_i * К_i * Т_i * Ц, \quad (4.2)$$

где M_i – паспортная мощность i-го электрооборудования, кВт;

$К_i$ – коэффициент использования мощности i-го электрооборудования (принимается $К_i$=0.7; 0.9);

$Т_i$ – время работы i-го оборудования за весь период выполнения НИР, ч;

Ц – цена электроэнергии, тг/кВт×ч.;

i – вид электрооборудования;

n – количество электрооборудования.

Приведем расчеты для каждого оборудования:

1) Ноутбук HP Pavilion g7

$$З_э = 0,047*0,9*410*27 = 468 \text{ тг.}$$

3) МФУ HP LaserJet M125a

$$З_э = 0,465*0,9*80*27 = 903 \text{ тг.}$$

4) Монитор LG

$$З_э = 0{,}046*0{,}9*410*27 = 458 \text{ тг.}$$

5) Смартфон Apple Iphone 5s

$$З_э = 0{,}015*0{,}9*100*27 = 35 \text{ тг.}$$

6) Смартфон Samsung Galaxy S4 mini

$$З_э = 0{,}02*0{,}9*100*27 = 49 \text{ тг.}$$

Итого общая сумма амортизационных отчислений составит 1913 тг.

4.2.3 Расчет затрат на оплату труда

В статью «Затраты на оплату труда» включаются расходы по оплате труда всех работников, занятых выполнением НИР (дипломника, руководителей и консультантов дипломной работы, привлеченных лиц).

Затраты на оплату труда рассчитываются по форме, приведенной в таблице 4.4.

Таблица 4.4 - Затраты на оплату труда

Категория работника	Месячная заработная плата, тг	Часовая ставка, тг/ч	Трудоемкость выполнения НИР, чел.×ч	Сумма, тг
Инженер-разработчик	100 000	595	410	243 950
Научный руководитель	120 000	714	150	107 100
ИТОГО затраты на оплату труда				351 050

Общая сумма затрат на оплату труда ($З_{ТР}$) определяется по формуле:

$$З_{тр} = \sum_{i=1}^{n} ЧС_i * Т_i , \quad (4.3)$$

где $ЧС_i$ – часовая ставка i-го работника, тг;

$Т_i$ – трудоемкость выполнения НИР, чел.×ч;

i – категория работника;

n – количество работников, занятых выполнением НИР.

Часовая ставка работника может быть рассчитана по формуле:

$$ЧС_i = \frac{ЗП_i}{ФРВ_i} \quad (4.4)$$

где $ЗП_i$ – месячная заработная плата i-го работника, тг

$ФРВ_i$ – месячный фонд рабочего времени i-го работника, $ФРВ_i$ = 168 ч

Проведем расчеты для сотрудников:

$$ЧС_1 = \frac{100\,000}{168} = 595 \text{ тг/ч} \qquad З_{тр1} = 595*410 = 243\,950 \text{ тг.}$$

$$ЧС_2 = \frac{120\,000}{168} = 714 \text{ тг/ч} \qquad З_{тр2} = 714*150 = 107\,100 \text{ тг.}$$

4.2.4 Расчет отчислений на социальные нужды

Затраты по этой статье составляют отчисления по единому социальному налогу (ЕСН).

Согласно Налоговому кодексу Республики Казахстан социальный налог составляет 11 % от ФОТ (фонда оплаты труда). Следует отметить, что пенсионные отчисления не облагаются социальным налогом. Отчисления по заработной плате определяются по следующей формуле:

$$Ос = (ФОТ- ПО)*0,11, \quad (4.5)$$

где ПО - отчисления в пенсионный фонд, что составляет 10% от ФОТ,

$$ПО = 351\ 050 * 0,1 = 35\ 105 \text{ тг.}$$

Итак, отчисления из заработной платы составили:

$$Ос= (351\ 050 – 35\ 105) * 0,11 = 34\ 754 \text{ тг.}$$

4.2.5 Расчёт амортизационных отчислений

В статью «Амортизация основных фондов» включается сумма амортизационных отчислений от стоимости оборудования и приборов, используемых при выполнении НИР. Амортизационные отчисления рассчитываются по форме, приведенной в таблице 4.5.

Общая сумма амортизационных отчислений определяется по формуле:

$$З_м= \sum_{i=1}^{n} \frac{Ф_i*Н_{Ai}*Т_{НИРi}}{100*Т_{Эфi}}, \quad (4.6)$$

где $Ф_i$ – стоимость i-го оборудования, тг;

$Н_{Ai}$ – годовая норма амортизации i-го оборудования, %;

$Т_{НИРi}$ – время работы i-го оборудования за весь период выполнения НИР, ч;

$Т_{Эфi}$ – эффективный фонд времени работы i-го оборудования за год, ч/год;

i – вид оборудования;

n – количество оборудования.

Приведем расчеты для каждого оборудования:

1) Ноутбук HP Pavilion g7

$$З_м= \frac{140\ 000*20*410}{100*2\ 416}=4\ 752 \text{ тг.}$$

3) МФУ HP LaserJet M125a

$$З_м= \frac{43\ 000*20*80}{100*759}=906 \text{ тг.}$$

4) Монитор LG

$$З_м= \frac{100\ 000*20*410}{100*1\ 518}=5\ 402 \text{ тг.}$$

5) Смартфон Apple Iphone 5s

$$З_м= \frac{70\ 000*20*100}{100*1\ 518}=922 \text{ тг.}$$

6) Смартфон Samsung Galaxy S4 mini

$$З_м = \frac{50\,000*20*100}{100*1\,518} = 659 \text{ тг.}$$

Итого общая сумма амортизационных отчислений составит 12 641 тг.

Таблица 4.5 - Амортизация основных фондов

Наименование оборудования	Стоимость оборудования, тг	Годовая норма амортизации, %	Эффективный фонд времени работы оборудования, ч/год	Время работы оборудования для выполнения НИР, ч	Сумма, тг
Ноутбук HP Pavilion g7	140 000	20	2 416	410	4 752
МФУ HP LaserJet M125a	43 000	20	759	80	906
Монитор LG	100 000	20	1 518	410	5 402
Смартфон Apple Iphone 5s	70 000	20	1 518	100	922
Смартфон Samsung Galaxy S4 mini	50 000	20	1 518	100	659
ИТОГО амортизация основных фондов					12 641

Годовые нормы амортизации оборудования принимаются по налоговому кодексу РК или определяются, исходя из возможного срока полезного использования оборудования:

$$Н_{Ai} = \frac{100}{Т_{Ni}}, \quad (4.7)$$

где $Т_{Ni}$ – возможный срок использования i-го оборудования, год.

Возможный срок полезного использования для всего оборудования составляет 5 лет.

В расчетах принимается максимально возможный фонд времени работы оборудования. эффективный фонд времени работы оборудования (для предприятий вторичных ресурсов с прерывным режимом работы) определяется числом рабочих дней в году и числом часов работы оборудования в сутки по формуле:

$$Т_{эф} = Т_{ном} * (t * С) - Т_{ппр}, \quad (4.8)$$

где $Т_{эф}$ – эффективный фонд времени работы оборудования, ч;

$Т_{ном}$ – номинальный фонд времени, дни;

(t*C) – среднее число часов работы оборудования в сутки, исходя из длительности смены в часах t и числа смен C;

$T_{ппр}$ – время останова оборудования на проведение планово-предупредительных ремонтов (включая время на подготовку к ремонту и пуск оборудования после ремонта), ч.

Проведем расчет для оборудования:

1) Ноутбук HP Pavilion g7
   $T_{эф1}$ = 352*8-400=2416 ч.
2) МФУ HP LaserJet M125a
   $T_{эф2}$ = 352*8-2057=759 ч.
3) Монитор LG
   $T_{эф3}$ = 352*8-1298=1518 ч.
4) Смартфон Apple Iphone 5s
   $T_{эф4}$ = 352*8-1298=1518 ч.
5) Смартфон Samsung Galaxy S4 mini
   $T_{эф5}$ = 352*8-1298=1518 ч.

4.2.6 Расчет прочих затрат

В статью «Прочие затраты» включаются расходы на арендную плату, включая коммунальные платежи, расходы на рекламу, канцелярские и прочие хозяйственные расходы.

Затраты на арендную плату определяются в зависимости от стоимости аренды 1 кв. м занимаемой площади.

Таблица 4.6 – Затраты на арендую плату

Площадь, кв.м.	Цена за кв.м., тг	Цена за месяц, тг	Срок, месяц	Сумма, тг
25	3 000	75 000	3	225 000

Затраты на оплату интернета берутся на основании стоимости услуг, предоставляемых оператором, тарифов и срока использования данного пакета услуг. При выполнении работы был использован тариф ID Net Turbo, его стоимость и расчёт общей суммы указан в таблице 4.7.

Таблица 4.7 – Затраты на пользование интернета

Цена за месяц, тг	Срок, месяц	Сумма, тг
4 600	3	13 800

Итого прочие затраты составляют 238 800 тг.

4.2.7 Составление сметы

На основании полученных данных по отдельным статьям была составлена смета затрат за выполнение НИР по форме, приведенной в таблице 4.8.

Таблица 4.8 - Смета затрат на выполнение НИР

Статьи затрат	Сумма, тг
1. Материальные затраты, в том числе: - оборудование и ПО - материалы - электроэнергия 2. Затраты на оплату труда. 3. Отчисления на социальные нужды. 4. Амортизация основных фондов. 5. Прочие затраты.	 469 000 8 500 1 913 351 050 34 754 12 641 238 800
ИТОГО по смете	1 116 658

4.2.8 Определение возможной (договорной) цены НИР

Величина возможной (договорной) цены НИР должна устанавливаться с учетом эффективности, качества и сроков ее выполнения на уровне, отвечающем экономическим интересам заказчика (потребителя) и исполнителя.

Договорная цена ($Ц_Д$) для прикладных НИР рассчитывается по формуле:

$$Ц_Д = З_{НИР} * \left(1 + \frac{Р}{100}\right), \quad (4.9)$$

где $З_{НИР}$ – затраты на выполнение НИР (из таблицы 4.8), тг;

Р – средний уровень рентабельности НИР, % (принимается в размере 20-30% по согласованию с консультантом по экономической части).

Далее определяется цена реализации с учетом налога на добавленную стоимость (НДС), ставка НДС устанавливается законодательно Налоговым Кодексом РК. На 2016 год ставка НДС установлена в размере 12%.

Цена реализации с учетом НДС рассчитывается по формуле:

$$Цр = Цд + Цд * НДС, \quad (5.0)$$

Приняв во внимание все предыдущие расчеты определим возможную (договорную) цену НИР

$$Ц_Д = 1\ 116\ 658 * \left(1 + \frac{20}{100}\right) = 1\ 339\ 990 \text{ тг.}$$

Цена с учетом НДС

Цр = 1 339 990 + 1 339 990 * 0,12 = 1 500 789 тг.

4.3 Оценка научно-технической результативности и социальной эффективности НИР

Итоговая стоимость изучения технологий и разработки кроссплатформенных мобильных приложений IT инфраструктурой на предприятии составила 1 500 789 тенге, в которую заложены все возможные затраты при разработки программного продукта.

Наибольшую стоимость проекта составляют материальные затраты, более 37%, задействованных в процессе данной разработки.

Разработка мобильного приложения является дорогим проектом, требующих больших интеллектуальных и финансовых затрат. Однако все вложенные средства легко окупаются благодаря наиболее легкому учету оборудования.

Сегодня на растущем рынке мобильного программного обеспечения, преобладают нативные приложения, т.е разработанные под определенную платформу. Не смотря на свою распространенность, они имеют свои недостатки, основной из них, что они могут быть использованы только на устройствах с определенной платформой. А появление всё нового программного обеспечения и увеличение количества существующих платформ (Android, iOS, Windows Phone, BlackBerry и др.) ставит перед разработчиками новые требования. В качестве решения этой проблемы все популярнее становится использование кроссплатформенной разработки приложений для современных устройств.

Оценка технико-экономической эффективности внедрения кроссплатформенной технологии разработки мобильных бизнес-приложений:

- Позволит уменьшить время и затраты на разработку мобильных приложений по сравнению с использованием стандартных средств разработки;
- Позволит уменьшить затраты на системную интеграцию корпоративных приложений по сравнению с использованием интеграционных брокеров приложений;
- Позволит уменьшить стоимость владения мобильных приложений по сравнению с использованием стандартных средств поддержки приложений;
- Позволит увеличить надежность и доступность приложений с помощью балансировки нагрузки на промышленного сервера.

5 Безопасность жизнедеятельности

5.1 Анализ условий труда

Рабочее место – система высокофункционально и пространственно-организованных технических средств и вещей труда, обеспечивающая благосклонные условия для удачного решения человеком-оператором поставленной прежде него задачи.

Правильно санкционированное рабочее пространство позволяет нарастить продуктивность труда на 8-20% и уменьшать вредоносное действие компьютера на самочувствие.

Утомляемость на производстве развивается в следствие неправильной организацией рабочего места, не организованные зоны размещения рабочего оборудования. Далее будут рассмотрены требования к рабочему месту.

Организация рабочего места зависит от способа и характера решаемых проблем, от рабочего оборудования, конкретной цели рабочей деятельности.

Габаритные и компоновочные параметры рабочего места определяются антропологическими характеристиками человека и нормированы в соответствующем документе – ГОСТ 21889-76.

Рабочее место разработчика складывается из:

– пространства, занимаемого оборудованием;

– пространства необходимого для технического обслуживания или ремонта;

– зоны проходов, обеспечивающей нормальное функционирование оборудования;

– сенсомоторного пространства (части пространства рабочего места, в которой осуществляется двигательная и сенсорная работа человека).

Из необходимых для работы устройств:

– монитор;

– стол;

– кресло;

– устройства ввода информации (мышь, клавиатура и т. д.);

– устройства вывода информации (принтер, плоттер).

В первую очередь рабочие места служащих, работающих с ПК, располагают подальше от окон и, следовательно, дабы оконные просветы пребывали сбоку. Экран монитора обращен к оконному просвету, важны защитные экраны. Окна рекомендовано снабжать светорассеивающими шторами, регулируемыми жалюзи либо солнцезащитной пленкой с металлизированным покрытием.

Монитор, документы, клавиатура обязаны находятся так, дабы перепад яркостей их плоскостей, зависящий от их месторасположения источников света, не был выше 1:10 при нужном значении 1:3. При яркости изображения на экране 50-100 кд/м (номинальное значение) освещенность документа

обязана составлять 300-500 лк. Обязаны быть исключены слепящие яркости, блики и отображения от стекла экрана защитной пленкой с металлизированным покрытием.

Для исключения засветки экранов мониторов прямыми световыми потоками электросветильники единого освещения располагают сбоку от рабочего места, вдоль полосы зрения оператора и стене с окнами.

Для обеспечения оптимальных условий работы операторов дисплейных устройств необходима определенная отделка помещений: должны использоваться диффузно – отражающиеся материалы с коэффициентом отражения для потолка – 0,7 – 0,8; для стен – 0,5 – 0,6; для пола – 0,3 – 0,5.

Пол в помещениях эксплуатации компьютеров должен быть ровным, без выбоин, нескользкой, удобной для очистки и для влажной уборки, обладать антистатическими свойствами.

Схемы размещения рабочих мест должны учитывать расстояния между рабочими столами с мониторами (в направлении тыла поверхности одного монитора и экрана другого монитора), которое должно быть не мене 2,0 м, а расстояние между боковыми поверхностями видеомониторов – не менее 1,2 м.

5.2 Требования к микроклимату в рабочей зоне помещений. Расчет системы кондиционирования

Кондиционирование воздуха выполняет задачи вентиляций и создает благоприятные микроклиматы. Кондиционирование выполняется в соответствиях с главой СНиП 11-33-75 "Отопление, вентиляция и кондиционирование воздуха".

Таблица 4.1. Оптимальные нормы температуры, относительной влажности и скорости движения в обслуживаемой зоне жилых, общественных и административно-бытовых помещений

Период года	Категория работ	Температура воздуха,	Относительная влажность воздуха, %	Скорость движения воздуха, м/с
Холодный	Легкая – 1а Легкая – 1б	22-24 21-23	40-60 40-60	0,1 0,1
Теплый	Легкая – 1а Легкая – 1б	23-25 22-24	40-60 40-60	0,1 0,2

Найдем количество приточного воздуха для более точного выбора кондиционера. Для этого используем формулу:

$$L_{пр} = \frac{Q_{изб}}{c * P_{пр} * (t_{вып} - t_{пр})} \quad (5.1)$$

где: $Q_{изб}$ – избыточное выделение явной теплоты, кДж/ч;

с – удельная теплоемкость воздуха при постоянном давлении, равная с= 1 кДж/кг*^{0}С;

$P_{пр}$ – плотность поступающего в помещение воздуха, 1,2кг/м^3

$t_{выт}$ – температура удаляемого из помещения воздуха за пределы рабочего или обслуживаемой зоны, ^{0}С;

$t_{пр}$ – температура приточного воздуха, ^{0}С;

Формула для нахождения температуры удаляемого из помещения воздуха за пределы рабочего или обслуживаемой зоны:

$$t_{выт} = t_{рз}+\Delta t*(h_{вп}-z) \quad (5.2)$$

где:

$t_{рз}$ – температура в рабочей зоне, которые не должна превышать допустимую по нормам ($t_{рз} \leq t_{доп}$), ^{0}С;

$h_{вп}$ – расстояние от пола до центра вытяжных проемов (кондиционера), м;

z – высота рабочей зоны, м

Производим расчёт для теплого периода года, возьмем $t_{РЗ}$ =23^{0}С;

Внутренняя часть кондиционера расположена на высоте $h_{вп}$ = 2 м

$t_{выт}$ =23+1,2*(2-1)=24,2^{0}С

Температура приточного воздуха $t_{пр}$ при наличии избытка явной теплоты должна быть на 5 – 7 ^{0}С ниже температуры воздуха в рабочей зоне

$t_{пр}$ = 24,2 – 6 = 18,2 ^{0}С

Величину избыточного выделения явной теплоты $Q_{изб}$ находят на основании баланса теплоты в помещении по формуле:

$$Q_{изб} = \Sigma Q – \Sigma Q_{ух} \quad (5.3)$$

где ΣQ – суммарное количество поступающей в помещение явной теплоты.

$\Sigma Q_{ух}$ – суммарное количество уходящей из помещения теплоты (за счет тепло потерь ограждениями, нагрева поступающего в помещение воздуха и т.п.)

В помещении источниками избыточного тепла являются человек, источник искусственного освещения, тепловыделения электронного оборудования, излучение солнца. Так как тепловыделения электронного оборудования не значительна им можно пренебречь.

Для расчета тепловыделения от искусственного света Q_1 предположим, что вся затрачиваемая энергия преобразуется в тепло.

$$Q_1 = N*n \quad (5.4)$$

где N – расходуемая мощность светильника, Вт

n – количество светильников.

Q_1 = 100*5 =500Вт

Тепловыделения от людей Q_2 определяется по формуле:

$$Q_2 = n*q \quad (5.5)$$

где n – число работающих,

q – количество тепла, выделяемое одним человеком, представлено в таблице 4.2

Таблица 4.2. Количество тепла, выделяемое одним человеком

Категория работ	Тепло, Вт			
	Полное		Явное	
	При 100^0 С	При 350^0 С	При 100^0 С	При 350^0 С
Легкая	180^0 С	145^0 С	150^0 С	5^0 С

Q_2 = 2*145=290 Вт

Количество тепла от солнечной радиации определяется формулой:

$$Q_{ост.\ рад} = F_{ост} * q_{ост} * A_{ост} \quad (5.6)$$

где $F_{ост}$ – площадь поверхности и покрытия, $м^2$;

$q_{ост}$ – теплопоступления через 1 $м^2$ поверхности остекления и поверхности покрытия, при коэффициенте теплопередачи, равном 1 $Вт/м^2 * {}^0С$

$A_{ост}$ - коэффициент остекления;

$F_{ост}$ = 2*4=8 $м^2$

Окно рабочего помещения направлено на юг поэтому $q_{ост}$ равен 150 $Вт/м^2 * {}^0С$

$A_{ост}$ = 0,5

$Q_{ост.\ рад}$ = 8*150*0,5=600Вт

Среднее значение теплопоступления для покрытия с учетом географической широты примем равным $Q_{п.рад}$ = 8 Вт.

Потери тепла из помещения $Q_{ух}$, кВт через стены двери, окна оценивают ориентировочно по формуле:

$$Q_{ух} = \frac{\lambda * S * (t_{выт} - t_{пр})}{\delta} \quad (5.7)$$

где λ – теплопроводность стен, $Вт/м^2 * {}^0С$.

S – площадь, $м^2$

δ – толщина стен, м.

Стены рабочего помещения изготовлены из кирпичей М350, теплопроводность которого равна 0,7 $Вт/м^2 * {}^0С$. Толщина стен δ = 0,3 м.

$Q_{ух}$ = 0,7*50*(22-16)/0,3 = 700 Вт.

Вычислим суммарное количество поступающей в помещение явной теплоты.

$$\sum Q = Q_1 + Q_2 + Q_{ост.рад} + Q_{пр.р} = 500 + 290 + 600 + 8 = 1398 Вт \quad (5.8)$$

Вычислим количество приточного воздуха;

$L_{пр}$=1398/(1*1,2*(24,2-18,2))=194$м^3$/ч

Выбираем кондиционер с нижней подачей Elenberg CSH-07J с максимальным расходом воздуха L=300 $м^3$/ч, что является оптимальным для обеспечения благоприятного микроклимата.

Таблица 4.3 Характеристика Elenberg CSH-07J

Площадь помещения	21 $м^2$
Мощность охлаждения	1000 Вт
Потребляемая мощность при охлаждении	750 Вт
Мощность обогрева	1000 Вт
Потребляемая мощность при обогреве	750 Вт

Рисунок 4.1 Фото кондиционера Elenberg CSH-07J

5.3 Освещение рабочего места. Расчет искусственного освещения

Высокопроизводительный труд разработчика зависит от качества освещения. Все это из-за того, что человеческий глаз воспринимает 90% информации извне с помощью зрения.

Факторы, которые снижают уровень зрения или являются факторами усталости: низкий уровень освещенности, приводящий к перенапряжению зрения и последующему переутомлению глаз.

Можно воспроизвести нужное освещение искусственно, с помощью ламп дневного света, ламп накаливания и т.д. В рядах помещений используют люминесцентные лампы, из-за ряда причин:

– возможность получить в 2 раза больше освещенности, с тем же расходом электроэнергии

– спектр такой лампы наиболее мягкий и близок к дневному

– срок службы превышает срок службы лампы накаливания в 10-12 раз

К минусам использования данного вида ламп можно отнести:

– изначальная стоимость довольно высока

– температурная зависимость

– значительная пульсация светового потока

Найдем необходимое число ламп при помощи метода коэффициента использования.

Расчёт системы общего освещения производится методом коэффициента использования светового потока, который выражается

отношением светового потока, падающего на расчётную поверхность, к суммарному потоку всех ламп. Его величина зависит от характеристик светильника, размеров помещения, окраски стен и потолка, характеризуемой коэффициентами отражения стен и потолка.

Необходимый световой поток лампы в каждом светильнике рассчитывается по формуле

$$F_0 = \frac{E*k*S*Z}{\eta} \quad (5.9)$$

где *E* – заданная минимальная освещённость, лк (E = 500);

k – коэффициент запаса, учитывающий уменьшение светового потока лампы в результате загрязнения светильников в процессе эксплуатации (для люмин. ламп – 1,5);

S – освещаемая площадь, м^2 (S = 12) ;

Z – отношение средней освещённости к минимальной (обычно принимается равным 1.1-1.2, для люмин. ламп – 1,1);

η – коэффициент использования светового потока в долях единицы (отношение светового потока, падающего на расчётную поверхность, к суммарному потоку всех ламп).

Коэффициент использования η зависит от типа светильника, от коэффициентов отражения потолка $\rho_п$, стен $\rho_с$, расчётной поверхности $\rho_р$, индекса помещения, который рассчитывается по формуле

$$i = \frac{S}{h*(a+b)} \quad (6.0)$$

где *h* – высота светильника над рабочей поверхностью;

a – длина помещения;

b – ширина помещения.

Найдем *h* по формуле

$h = H – h_p – h_c = 3 – 0,7 – 0 = 2,3$ *(м)*

где *H* – высота помещения, м (H=3);

h_p – высота рабочей поверхности от пола, м (h_p = 0,7);

h_c – высота свеса светильника от основного потолка, м (h_c = 0).

$$i = \frac{6 * 2}{2,3 * (6 + 4)} = \frac{12}{18,4} = 0.65$$

Для светлого фона примем: $\rho_п = 70$, $\rho_с = 50$, $\rho_р = 10$ $\eta = 36$ %.

$$F_л = \frac{F_0}{N} \quad (6.1)$$

где Fл – световой поток 1

лампы

Fo – общий световой поток;

N – число ламп;

η – коэффициент использования светового потока.

$$F_{\text{о}} = \frac{500 * 1{,}5 * 12 * 1{,}1}{0{,}36} = 27500 \text{ лм}$$

Количество ламп высчитываем из формулы

$$N = \frac{F_{\text{о}}}{F_{\text{л}}} \tag{6.2}$$

Для освещения выбираем люминесцентные лампы типа ЛХБ65, световой поток которых *Fл* = 4400 Лм.

$$N = \frac{27500}{4400} = 6{,}25$$

Число светильников выбирается в зависимости от размеров освещаемого помещения, при этом количество светильников должно быть таким, чтобы отношение расстояния между ними к высоте их подвеса над поверхностью было равно 1,5 ÷ 2.

При выборе осветительных приборов используем светильники типа ЛСПО 2. Каждый светильник комплектуется двумя лампами. Размещаются светильники тремя рядами, по два в каждом ряду.

Допускается отклонение (ε) светового потока выбранной лампы от расчётного от – 10 % до + 20 %.

$$E_{\text{факт}} = \frac{\Phi_{\text{л}} * N * N_{\text{л.св.}} * \eta}{S * z * k} \tag{6.3}$$

$$E_{\text{факт}} = \frac{4400 * 6 * 2 * 0.36}{20 * 1{,}1 * 1{,}5} = 576 \text{ лк}$$

Отличие от нормированного уровня

$$\frac{E_{\text{факт}} - E_{\text{норм}}}{E_{\text{номр}}} * 100\% \tag{6.4}$$

$$\frac{576 - 500}{500} * 100\% = 15{,}2\%$$

В целях исключения засветки экранов мониторов прямыми световыми потоками электросветильники совокупного освещения располагают сбоку от рабочего места, вдоль полосы зрения оператора и стене с окнами. Данное

расположение источников света позволяет производить их последовательное включение в зависимости от величины естественной освещённости и исключает раздражение глаз чередующимися полосами света и тени, возникающее при поперечном расположении светильников.

Электрическая мощность всей осветительной системы вычисляется по формуле

$$P_{общ} = P_1 * N \quad (6.5)$$

где P_1 – мощность одной лампы = 65 Вт;
N – число ламп = 6.

$$P_{общ} = 65 * 6 = 390 \text{ Вт}$$

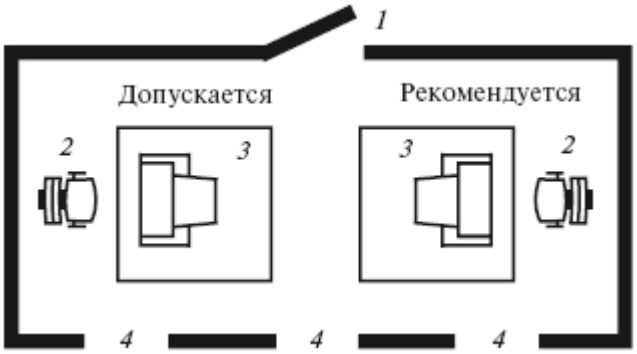

Рисунок 5.1 – План помещения

1 – Дверь; 2 – Рабочее кресло; 3 – Рабочий стол; 4 – Окна

Заключение

Во время выполнения данной дипломной работы были изучены основные принципы разработки мобильных приложений, а также требований, предъявляемых к ним. Изучены принципы построения и работы мобильных приложений для популярных на рынке платформ. Помимо этого, было также установлено, что исследуемая тема является в наивысшей степени актуальной в реалиях современного рынка мобильных устройств.

Также было разработано кроссплатформенное приложение, которое, при желании, может быть перенесено на другие мобильные операционные системы. Данное приложение отвечает всем стандартам и требованиям разных платформ. Предоставляет удобный способ работы с устройством и позволяет гибко изменять его интерфейс под нужды пользователя. Имеет в своем наличии уникальные функции, которые не доступны в стандартных, системных приложениях, а также в аналогичных продуктах сторонних разработчиков.

В технико-экономической части работы была рассчитана общая стоимость программного продукта и количество финансовых затрат на его разработку. В результате расчетов выяснилось, что разработка приложения является экономически эффективным и целесообразным решением.

В разделе безопасности жизнедеятельности был проведен анализ трудовых условий и оценка рабочего помещения, а так же предоставляется расчет систем пожаротушения и кондиционирования.

В процессе выполнения дипломной работы была достигнута цель разработки и выполнены все поставленные задачи.

Список используемой литературы

1. Gary Bennett, Mitchell Fisher Учебник – Objective-C, 2010
2. Обзорная статья: Язык программирования C#
3. А. Аллан Учебник - Программирование для мобильных устройств на iOS, 2013
4. Дэйв Марк, Джек Наттинг - Beginning iOS 6 Development: Exploring the iOS SDK, 2013
5. Armstrong J.S. Forecasting for Marketing // Quantitative Methods in Marketing. London: International Thompson Business Press, 1999. P. 92 – 119.
6. https://www.xamarin.com/
7. Jingfei Yang M. Sc. Power System Short-term Load Forecasting: Thesis for Ph.d degree. Germany, Darmstadt, Elektrotechnik und Informationstechnik der Technischen Universitat, 2006. -139 с.
8. Ретабоуил Сильвен. Android NDK. Создание приложений под Android на C/C++ – Москва: ДМК Пресс, 2012. – 296 с.
9. Э. Бурнет – Привет, Андроид! Разработка мобильных приложений, 2012.
10. http://phonegap.com/
11. http://www.sqlite.org/docs.html
12. http://www.appcelerator.com/
13. Медникс Зигард, Дорнин Лаирд, Мик Блэик, Накамура Масуми Программирование под Android – Питер:Москва, 2013. – 651 с.
14. Android A Programmers Guide - М. ИФРАН, 2011 – 399 с
15. Методические указания к выполнению экономической части дипломных работ для студентов специальности 5В070400 – Вычислительная техника и программное обеспечение. З.Д. Еркешева, Г.Ш. Боканова Алматы: АУЭС, 2013 – 40 с.
16. ГОСТ 12.1.005-88 ОБЩИЕ САНИТАРНО-ГИГИЕНИЧЕСКИЕ ТРЕБОВАНИЯ К ВОЗДУХУ РАБОЧЕЙ ЗОНЫ
17. ГОСТ 30494-96 ПАРАМЕТРЫ МИКРОКЛИМАТА В ПОМЕЩЕНИЯХ
18. habrahabr.ru
19. github.com
20. С. Хашими, С. Коматинени, Д. Маклин Учебник - Про Android, 2011.
21. http://yiiframework.ru/doc/guide/ru/basics.best-practices

Приложение А

Листинг программной части

```
using SQLite;
using System;
using System.Collections.Generic;
using System.Text;
namespace CrossplatformApp
          { public class MessageItem
             { [PrimaryKey, AutoIncrement]
public Guid Id { get; set; }
public string Text { get; set; }
 public string User { get; set; }
public string Created { get; set; } }
          }
```

```
<Page
    x:Class="CrossplatformApp.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:CrossplatformApp"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
        <Grid.RowDefinitions>
            <RowDefinition Height="100*"/>
            <RowDefinition Height="567*"/>
        </Grid.RowDefinitions>

        <ListView x:Name="SampleListView" ItemsSource="{Binding}" Grid.RowSpan="2" Grid.Row="1" Margin="20,20,20,0" >
            <ListView.ItemTemplate>
            <DataTemplate>
                <Grid Grid.Column="1" Margin="10,0,0,0">
                    <Grid.RowDefinitions>
                        <RowDefinition Height="Auto" />
                        <RowDefinition Height="Auto" />
                    </Grid.RowDefinitions>
                    <TextBlock HorizontalAlignment="Left"
                               Text="{Binding User}"
                               VerticalAlignment="Top"
                               Margin="0,0,0,-0.167"
                               Style="{StaticResource ListViewItemTextBlockStyle}"/>
                    <TextBlock HorizontalAlignment="Left"
                               TextWrapping="Wrap"
                               Text="{Binding Text}"
                               VerticalAlignment="Top"
                               Margin="0"
                               Grid.Row="1"
                               Style="{StaticResource ListViewItemSubheaderTextBlockStyle}"/>
                </Grid>
            </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
        <TextBlock HorizontalAlignment="Left" TextWrapping="Wrap" Text="SQLite PCL" VerticalAlignment="Bottom" FontSize="4
8" FontFamily="Global User Interface" Margin="20,0,0,0"/>

    </Grid>
</Page>
```

```
<?xml version="1.0" encoding="utf-8" ?>
<SampleMetadata>
```

```
        <ID>E13B0BD3-B46A-4DFC-8253-4006AE42B7B2</ID>
        <IsFullApplication>true</IsFullApplication>
        <Brief>This sample demonstrates a Todo list application where the data is stored
and accessed from an Azure Mobile App instance.</Brief>
        <Level>Advanced</Level>
        <LicenseRequirement>Indie</LicenseRequirement>
        <Tags>Web Services, Xamarin.Forms</Tags>
        <SupportedPlatforms>Android, iOS, Windows</SupportedPlatforms>
        <Gallery>true</Gallery>
</SampleMetadata>
```

```
using
System;
using System.Collections.Generic;
using System.Linq;


using Foundation;
using UIKit;


namespace TodoAzure.iOS
{
        [Register ("AppDelegate")]
        public partial class AppDelegate :
global::Xamarin.Forms.Platform.iOS.FormsApplicationDelegate
        {
                public override bool FinishedLaunching (UIApplication app, NSDictionary
options)
                {
                        global::Xamarin.Forms.Forms.Init ();


                        Microsoft.WindowsAzure.MobileServices.CurrentPlatform.Init();


            // IMPORTANT: uncomment this code to enable sync on Xamarin.iOS
            // For more information, see:
http://go.microsoft.com/fwlink/?LinkId=620342
            //SQLitePCL.CurrentPlatform.Init();


                        LoadApplication (new App ());


                        return base.FinishedLaunching (app, options);
                }
        }
}
```

```
<?xml version="1.0"
encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
```

```
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
	<key>CFBundleDisplayName</key>
	<string>TodoAzure</string>
	<key>CFBundleIdentifier</key>
	<string>com.xamarin.sample.TodoAzure</string>
	<key>CFBundleShortVersionString</key>
	<string>1.0</string>
	<key>CFBundleVersion</key>
	<string>1.0</string>
	<key>LSRequiresIPhoneOS</key>
	<true/>
	<key>MinimumOSVersion</key>
	<string>7.0</string>
	<key>UIDeviceFamily</key>
	<array>
		<integer>1</integer>
		<integer>2</integer>
	</array>
	<key>UIRequiredDeviceCapabilities</key>
	<array>
		<string>armv7</string>
	</array>
	<key>UISupportedInterfaceOrientations</key>
	<array>
		<string>UIInterfaceOrientationPortrait</string>

	<string>UIInterfaceOrientationLandscapeLeft</string>

	<string>UIInterfaceOrientationLandscapeRight</string>
	</array>
	<key>UISupportedInterfaceOrientations~ipad</key>
	<array>
		<string>UIInterfaceOrientationPortrait</string>

	<string>UIInterfaceOrientationPortraitUpsideDown</string>

	<string>UIInterfaceOrientationLandscapeLeft</string>

	<string>UIInterfaceOrientationLandscapeRight</string>
	</array>
	<key>CFBundleIconFiles</key>
	<array>
		<string>Icon-60@2x</string>
		<string>Icon-60@3x</string>
		<string>Icon-76</string>
		<string>Icon-76@2x</string>
		<string>Default</string>
		<string>Default@2x</string>
```

```
                <string>Default-568h</string>
                <string>Default-568h@2x</string>
                <string>Default-Landscape</string>
                <string>Default-Landscape@2x</string>
                <string>Default-Portrait</string>
                <string>Default-Portrait@2x</string>
                <string>Icon-Small-40</string>
                <string>Icon-Small-40@2x</string>
                <string>Icon-Small-40@3x</string>
                <string>Icon-Small</string>
                <string>Icon-Small@2x</string>
                <string>Icon-Small@3x</string>
        </array>
        <key>UILaunchStoryboardName</key>
        <string>LaunchScreen</string>
</dict>
</plist>
```

```
using System;
using System.Collections.Generic;
using System.Linq;

using Foundation;
using UIKit;

namespace TodoAzure.iOS
{
        public class Application
        {
                // This is the main entry point of the application.
                static void Main (string[] args)
                {
                        // if you want to use a different Application
Delegate class from "AppDelegate"
                        // you can specify it here.
                        UIApplication.Main (args, null, "AppDelegate");
                }
        }
}
```

```
using
System;

using Android.App;
using Android.Content;
using Android.Content.PM;
using Android.Runtime;
using Android.Views;
using Android.Widget;
```

```
using Android.OS;


namespace TodoAzure.Droid
{
	[Activity (Label = "TodoAzure.Droid",
		Icon = "@drawable/icon",
		MainLauncher = true,
		ConfigurationChanges = ConfigChanges.ScreenSize |
ConfigChanges.Orientation,
		Theme = "@android:style/Theme.Holo.Light")]
	public class MainActivity :
global::Xamarin.Forms.Platform.Android.FormsApplicationActivity
	{
		protected override void OnCreate (Bundle bundle)
		{
			base.OnCreate (bundle);


			global::Xamarin.Forms.Forms.Init (this, bundle);

	Microsoft.WindowsAzure.MobileServices.CurrentPlatform.Init();
			LoadApplication (new App ());
		}
	}
}
```

```
<?xml
version="1.0"
encoding="utf-
8"?>
```

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
android:versionCode="1" android:versionName="1.0"
package="com.xamarin.sample.TodoAzure">
	<uses-sdk android:minSdkVersion="15" android:targetSdkVersion="21"
/>
	<application android:label="TodoAzure"
android:icon="@drawable/icon">
	</application>
</manifest>
```