ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ФАКУЛЬТЕТ МАТЕМАТИКИ И ЕСТЕСТВЕННОНАУЧНОГО ОБРАЗОВАНИЯ

КАФЕДРА ИНФОРМАТИКИ, ЕСТЕСТВЕННОНАУЧНЫХ ДИСЦИПЛИН И МЕТОДИК ПРЕПОДАВАНИЯ

Разработка приложения "Помощник учителя" под операционную систему Android

Выпускная квалификационная работа
обучающегося по направлению подготовки 44.03.05 Педагогическое образование, профиль Информатика и иностранный язык (английский)
очной формы обучения, группы 02041205
Демченко Василия Джумбериевича

Научный руководитель:
старший преподаватель
Рядинская Л.В

БЕЛГОРОД 2017

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Трудно переоценить значение системы образования в современном обществе. Человек попадает в эту систему буквально с первых дней его жизни. К примеру, ясельная группа в детском саду начинается в возрасте 1-2 лет, последующие группы охватывают период с 2 до 6 лет. Затем наступает период школьного возраста. Который продолжает формировать ребёнка, как личность, ежедневно готовя его к настоящей жизни.

Современная система предполагает активное участие как учителя, так и родителей и близких родственников, в формировании жизни ученика. Им вместе, предстоит обсудить множество вопросов, принять важные решения и быть всегда на связи друг с другом. Это позволяет иметь актуальную информацию о жизни ребёнка, что в свою очередь позволяет своевременно реагировать на изменения в его школьной жизни.

Современные средства связи, такие как мобильная связь и интернет, предлагают множество способов доставки мгновенных сообщений. Можно с уверенностью сказать, что сейчас интернет вытесняет мобильную связь, потому что позволяет совершать групповые звонки, создавать общие диалоги, чаты, для обмена актуальной информацией.

Система школьного образования так же следует этим современным трендам, из последних нововведений мы увидели такие проекты как: «Электронная школа», «Электронный журнал» и «Электронный дневник».

Всё это представлено на электронном ресурсе школы, обычно это сайт. Так как для школ не определён общий дизайн и инструмент в создании сайта, представляющего школу, зачастую они могут сильно различаться по своему техническому функционалу, быть сложно организованными или иметь простую структуры. Это позволяет в большей или меньшей степени представить актуальную информацию о школьной жизни.

Цель: создание платформы для эффективной связи системы

образования и родителей, которая позволила бы вести активный диалог между современной системой образования и родителями.

Объект: разработка мобильного приложения, с использованием современных средств и достижений информационных технологий.

Предмет: мобильное приложение «Помощник классного руководителя».

В нашей работе я постараюсь доказать необходимость создания мобильного приложения, как современного и эффективного средства обмена информацией между учителем и родителями.

1 ВЫБОР СРЕДСТВ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

1.1 Понятие и классификации современных средств организации связи между учителем и родителями

Давайте рассмотрим современные средства, для организации связи между учителем и родителями. Сотовую связь мы сразу определим как устаревшую и второстепенную, подходящую больше для решения конкретных, трудных вопросов в личном диалоге. Поэтому будем рассматривать средства направленные на активное использование современных технологий.

Мы изучили самые известные и подходящие нам по параметрам приложения. Был изучен их функционал, а так же была проведена проверка практикой. Это было сделано что бы учесть все плюсы и минусы, уже представленных приложений в этой сфере.

Первое в нашем списке – социальные сети. Давайте дадим определение.

Социальная сеть — платформа, онлайн-сервис и веб-сайт, предназначенные для построения, отражения и организации социальных взаимоотношений в Интернете.

Это явление активно вписалось в нашу современную жизнь, с уверенностью можно сказать, что все родители следующего поколения детей будут иметь аккаунты в социальных сетях. Рассмотрим социальные сети со стороны практической ценности для нас, как инструмента создания связи между учителем и родителями.

Социальные сети позволяют нам реализовать множество возможностей, там же мы можем создать и общие диалоги, закрепить необходимый нам материал, обмениваться информацией как в текстовом, так и в других форматах, например изображения или видео.

Самая популярная социальная сеть в России – «ВКонтакте». Её функционал почти полностью соответствует нашим требованиям. Но есть и большой минус, социальная сеть предполагает собой использование персонального аккаунта, со всеми его вытекающими. А именно на своей странице люди представляют о себе личную информацию, из мирового опыта пользования социальными сетями мы видим, что многие люди предпочитают использование «фейковых», ненастоящих аккаунтов, что может создать почву для конфликта. Не говоря уже о том, что некоторым людям по долгу службу запрещено иметь аккаунт в социальных сетях.

Несмотря на то, что самая популярная социальная сеть в Россий это «ВКонткте», некоторые предпочитают другие, такие как «Facebook» и прочие. К сожалению в таком случае, создать общий диалог использую социальную сеть как платформу, не получится.

Следующим шагом мы рассмотрим так называемые «мессенджеры». Дадим определение.

Messenger - от английского "курьер" или "связной". Это программы для мгновенного обмена сообщениями между пользователями. Именно в скорости состоит их главное преимущество перед обычной электронной почтой. Здесь послание передается молниеносно, тогда как обновление почтового ящика происходит раз в несколько минут. Говоря о том, что такое мессенджер, следует уточнить важную особенность - он является клиентской программой. Это значит, что самостоятельно программа работать не может, для ее использования необходимо подключение к серверу (центральному компьютеру сети).

Сразу обозначим конкретные примеры, такие как «Viber» и «WhatsApp». Будем рассматривать их вместе, потому что функционал у них практически одинаковый. Обе платформы позволяют нам создавать групповые диалоги, прикреплять изображения и видео. Сообщения приходят мгновенно и позволяют удобно пользоваться списком контактов и диалогов.

По сравнению с социальной сетью, отсутствует минус, о котором мы

говорили выше. В этих приложениях нет никакой кастомизации профиля, за исключением фото, представляющего контакт. Но есть другой минус, в этих приложениях нельзя создавать диалоги со сложной структурой и как следствие нельзя разграничивать доступ, разделяя категории например на учителей и родителей. Нет настроек видимости сообщений, а так же необходимо иметь контакты, записанные в своём телефоне, для реализации возможностей группового диалога.

Можно сказать, что сейчас существующего функционала вполне хватает, что бы удовлетворить современные задачи в определённой нами проблеме. Тем не менее, в нашей работе мы хотим реализовать такие функции как создание расписания, редактирование его по дням, обзор и восстановление его из архива, оповещение о последних событиях класса, рассылка смс оповещения и уведомлений на смартфон пользователя, а так же разграничение уровня доступа в приложении. Все выше описанные технологии не позволят реализовать нам задуманное, что подтверждает актуальность разработки приложения. Так же, давайте не забывать, что сейчас мы используем приложения, стороннего разработчика, которое не может быть изменено под наши нужды, мы можем пользоваться только тем, что есть, не в силах поменять что то в функционале самого приложения. Так же, как мы видим из современных реалий, даже такая крупная сеть как «ВКонтакте» может быть заблокирована. Эту причину можно отнести к основной, почему стоит иметь специальное приложение, которое могло бы быть настроена под нужды системы образования.

1.2 Обзор средств разработки

При выборе средств разработки был проанализирован достаточно большой спектр всевозможных решений.

В настоящий момент как такого приложения даже частично реализующего идеи поставленные в данной работе – нету.

Для разработки приложения прежде всего нужно учитывать, что клиент программы будет установлен или на компьютер или смартфон пользователя. В связи с этим выбор был между C#(Visual Studio) и Java(Android studio).

Изначально язык назывался Oak («Дуб») разрабатывался Джеймсом Гослингом для программирования бытовых электронных устройств. Впоследствии он был переименован в Java и стал использоваться для написания клиентских приложений и серверного программного обеспечения. Назван в честь марки кофе Java, которая, в свою очередь, получила наименование одноимённого острова (Ява), поэтому на официальной эмблеме языка изображена чашка с горячим кофе. Существует и другая версия происхождения названия языка, связанная с аллюзией на кофе-машину как пример бытового устройства, для программирования которого изначально язык создавался. В соответствии с этимологией, в русскоязычной литературе с конца двадцатого и до первых лет двадцать первого века название языка нередко переводилось как Ява, а не транскрибировалось, как это стало общепринятым позднее.

Ключевой особенностью языка Java является то, что его код сначала транслируется в специальный байт-код, независимый от платформы. А затем этот байт-код выполняется виртуальной машиной JVM (Java Virtual Machine). В этом плане Java отличается от стандартных интерпретируемых языков как PHP или Perl, код которых сразу же выполняется интерпретатором. В то же время Java не является и чисто компилируемым языком, как C или C++.

Подобная архитектура обеспечивает кроссплатформенность и аппаратную переносимость программ на Java, благодаря чему подобные

программы без перекомпиляции могут выполняться на различных платформах - Windows, Linux, Solaris и т.д. Для каждой из платформ может быть своя реализация виртуальной машины JVM, но каждая из них может выполнять один и тот же код.

Java является языком с Си-подобным синтаксисом и близок в этом отношении к C/C++ и C#. Поэтому, если вы знакомы с одним из этих языков, то овладеть Java будет легче.

Еще одной ключевой особенностью Java является то, что она поддерживает автоматическую сборку мусора. А это значит, что вам не надо освобождать вручную память от ранее использовавшихся объектов, как в C++, так как сборщик мусора это сделает автоматически за вас.

Java является объектно-ориентированным языком. Он поддерживает полиморфизм, наследование, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений.

Качества, которые делают Java столь привлекательным, имеются и в других языках программирования. Многие языки идеально подходят для определенных типов приложений. Даже больше, чем Java. Но Java собрал все эти качества вместе, в одном языке. Для индустрии разработки программного обеспечения — это революционный скачок вперед.

Давайте подробнее взглянем на некоторые из свойств этого языка:

- объектно-ориентированность
- переносимость
- поддержка многопоточности
- автоматическая «сборка мусора»
- надежность
- поддержка работы с сетью и «Интернетом»
- простота и лёгкость в использовании

Объектно-ориентированность

Многие языки, возникшие до Java, такие как C и Pascal, являлись процедурными языками. Процедуры (или функции) — это блоки кода, которые являлись частью модуля или приложения. В процедуры передавались параметры (примитивные типы данных: целые числа, символы, строки и числа с плавающей запятой). Код обрабатывался отдельно от данных. Вам приходилось передавать структуры данных, а процедуры могли с легкостью изменять их содержимое. Из-за этого возникало много проблем, поскольку использование одних частей программы в других ее частях могло давать непредсказумые результаты. На поиски неисправной процедуры уходило огромное количество времени и усилий. Особенно, когда дело касалось больших программ.

Некоторые из процедурных языков позволяли даже получить адрес данных в памяти. Зная этот адрес, можно было считывать или дописывать данные спустя некоторое время, либо случайно записать новую информацию поверх старой.

Java — это объектно-ориентированный язык. Объектно-ориентированный язык работает с объектами. Объекты содержат данные (поля) и код (методы). Каждый объект принадлежит определенному классу, который является «чертежом», описывающим поля и методы, которые предлагает объект. В Java практически любая переменная является объектом того или иного вида — даже строки. Объектно-ориентированное программирование требует иного типа мышления, но такой способ разработки ПО лучше, чем процедурное программирование.

На сегодняшний день существует множество популярных объектно-ориентированных языков. Некоторые из них были изначально разработаны как объектно-ориентированные, например, Java и Smalltalk. Другие, такие как C++, являются частично объектно-ориентированными и частично процедурными. В C++ вы по-прежнему можете перезаписать содержимое структур данных и объектов, что приведет к сбою приложения. К счастью, Java запрещает прямой доступ к содержимому памяти, тем самым создавая

более надежную систему.

Переносимость

Большинство языков программирования спроектированы под конкретную операционную систему и процессор. При компиляции исходный код (инструкции, из которых строится программа) превращается в машинный код, который может быть исполнен только на устройствах определенного типа. Этот процесс порождает «внутренний код», работающий невероятно быстро.

Есть и другие типы языков — интерпретируемые. Интерпретируемый код считывается программным приложением (интерпретатором), которое выполняет указанные действия. Интерпретируемый код чаще всего компилировать не нужно — он транслируется по мере выполнения. Из-за этого интерпретируемый код работает довольно медленно, но он позволяет переносить программы между различными операционными системами и процессорами с разной архитектурой.

Надежность

Безопасность играет в Java очень важную роль. Поскольку апплеты Java загружаются удаленно и выполняются в браузере, безопасности уделяется много внимания. Мы бы не хотели, чтобы апплеты получали доступ к нашим личным документам, могли удалять наши файлы или наносить какой-либо вред. На уровне API существуют строгие ограничения по безопасности, когда речь идет о доступе апплетов к файлам и к сети. Кроме того, для проверки целостности загружаемого кода имеется поддержка цифровых подписей. На уровне байткода проверяются очевидные взломы, такие как манипуляция стеком или неверный байткод. Мощные механизмы обеспечения безопасности в Java помогают защититься от неумышленных или преднамеренных нарушений безопасности, но важно не забывать, что идеальных систем не бывает. Самым слабым звеном в этой цепи является Java Virtual Machine, в которой всё и работает — JVM может быть подвержена атакам, поскольку у нее есть известные слабые стороны.

Стоит отметить, что хотя в JVM и было найдено несколько уязвимостей, такое происходит очень редко и обычно быстро исправляется.

Поддержка работы с сетью и «Интернетом»

Java создавался с оглядкой на «Internet» и на поддержку сетевого программирования. API Java предоставляет обширную поддержку сетевых функций, от сокетов и IP-адресов до URL и HTTP. Нет ничего проще, чем написать на Java сетевое приложение. При этом получившийся код можно будет перенести на любую платформу. В таких языках как C/C++ код, работающий с сетью, нужно переписывать заново для каждой операционной системы, и обычно он имеет более сложную структуру. Поддержка работы с сетью в Java позволяет сэкономить много времени и сил.

Java поддерживает и более экзотические виды сетевого программирования, например, удаленный вызов методов (RMI), обобщённую архитектуру обработчика объектных запросов (CORBA) и архитектуру распределенных систем Jini. Эти технологии распределенных систем делают Java привлекательным языком для масштабных проектов.

Простота и лёгкость в использовании

Язык Java своими корнями уходит в язык C++. C++ очень популярен и широко распространен. И, тем не менее, он считается сложным языком, имеющим такие функциии, как множественное наследование, шаблоны и указатели, которые являются контр-продуктивными. В свою очередь, Java — это скорее «чисто» объекто-ориентированный язык. Здесь нет доступа к указателям памяти, а вместо них используются ссылки на объекты. Поддержку множественного наследования тоже убрали. Это позволило добиться более понятных и простых схем классов. Библиотеки ввода/вывода и работы с сетью очень просты в использовании. API Java предоставляет разработчикам большое количество кода для экономии их времени (функции работы с сетью и структуры данных). Поработав с Java некоторое время, большинство разработчиков неохотно возвращаются к другим языкам из-за простоты и элегантности Java.

Следующий инструмент разработки это C#.

История си шарп — является недавней. Язык появился на свет в июне 2000 г. в результате кропотливой работы большой группы разработчиков компании Microsoft, возглавляемой Андерсом Хейлсбергом (Anders Hejlsberg). Этот человек известен как автор одного из первых компилируемых языков программирования для персональных компьютеров IBM — Turbo Pascal. Наверное, на территории бывшего Советского Союза многие разработчики со стажем, да и просто люди, обучавшиеся в той или иной форме программированию в вузах, испытали на себе очарование и удобство использования этого продукта. Кроме того, во время работы в корпорации Borland Андерс Хейлсберг прославился созданием интегрированной среды Delphi (он руководил этим проектом вплоть до выхода версии 4.0).

Появление языка си шарп и инициативы .NET отнюдь не случайно пришлось на начало лета 2000 г. Именно к этому моменту компания Microsoft подготовила промышленные версии новых компонентных технологий и решений в области обмена сообщениями и данными, а также создания Интернет-приложений (COM+, ASP+, ADO+, SOAP, Biztalk Framework). Несомненно, лучшим способом продвижения этих новинок является создание инструментария для разработчиков с их полноценной поддержкой. В этом и заключается одна из главных задач нового языка. Кроме того Microsoft не могла больше расширять одни и те же инструменты и языки разработки, делая их все более и более сложными для удовлетворения конфликтующих между собой требований поддержки современного оборудования и обеспечения обратной совместимости с теми продуктами, которые были созданы в начале 1990-х гг. во время первого появления Windows. Наступает момент, когда необходимо начать с чистого листа для того, чтобы создать простой, но имеющий сложную структуру набор языков, сред и средств разработки, которые позволят разработчику легко создавать современные программные продукты.

C# и .NET являются той самой отправной точкой. Если говорить упрощенно, то .NET представляет собой новую платформу, новый API для программирования в Windows, а C# ее новый язык, созданный с нуля, для работы с этой платформой, а также для извлечения всех выгод из прогресса сред разработки и нашего понимания принципов объектно-ориентированного программирования в течение последних 20 лет.

Необходимо отметить, что обратная совместимость не потеряна. Существующие программы будут выполняться, а платформа .NET была спроектирована таким образом, чтобы она могла работать с имеющимся программным обеспечением. Связь между компонентами в Windows сейчас почти целиком осуществляется при помощи COM. С учетом этого .NET обладает способностью создавать оболочки (wrappers) вокруг существующих компонентов COM, так что компоненты .NET могут общаться с ними, и создавать оболочки вокруг компонентов .NET, что позволяет им выглядеть как обычные COM-компоненты.

Авторы C# стремились создать язык, сочетающий простоту и выразительность современных объектно-ориентированных языков (вроде Java) с богатством возможностей и мощью C++. По словам Андерса Хейлсберга, C# позаимствовал большинство своих синтаксических конструкций из C++. В частности, в нем присутствуют такие удобные типы данных, как структуры и перечисления (другой потомок C++ — Java лишен этих элементов, что создает определенные неудобства при программировании). Синтаксические конструкции C# унаследованы не только от C++, но и от Visual Basic. Например, в C#, как и в Visual Basic, используются свойства классов. Как и C++ позволяет производить перегрузку операторов для созданных вами типов, Java не поддерживает ни ту, ни другую возможность. C# — это фактически гибрид разных языков, при этом синтаксически не менее (если не более) чист чем Java, так же прост как Visual Basic, и обладает практически той же мощью и гибкостью, что и C++.

Особенности C#:

1. Полная поддержка классов и объектно-ориентированного программирования, включая наследование интерфейсов и реализаций, виртуальных функций и перегрузки операторов.

2. Полный и хорошо определенный набор основных типов.

3. Встроенная поддержка автоматической генерации XML-документации.

4. Автоматическое освобождение динамически распределенной памяти.

5. Возможность отметки классов и методов атрибутами, определяемыми пользователем. Это может быть полезно при документировании и способно воздействовать на процесс компиляции (например, можно пометить методы, которые должны компилироваться только в отладочном режиме).

6. Полный доступ к библиотеке базовых классов .NET, а также легкий доступ к Windows API (если это действительно необходимо).

7. Указатели и прямой доступ к памяти, если они необходимы. Однако язык разработан таким образом, что практически во всех случаях можно обойтись и без этого.

8. Поддержка свойств и событий в стиле VB.

9. Простое изменение ключей компиляции. Позволяет получать исполняемые файлы или библиотеки компонентов .NET, которые могут быть вызваны другим кодом так же, как элементы управления ActiveX (компоненты COM).

10. Возможность использования C# для написания динамических web-страниц ASP.NET.

Как мы видим, на сегодняшний день у нас действительно есть из чего выбрать, но так как наше приложение будет ориентировано на мобильные платформы, нам больше подойдёт Java, а следовательно Android Studio. В дальнейшем развитии приложения, его платформу можно так же будет перенести и на персональные компьютеры, но актуальность такой разработки

ещё предстоит изучить, так как Android Studio предоставляет прекрасные возможности для разработки приложения, которые не будут уступать по функционалу разработкам для персонального компьютера.

1.3 Обзор выбранного средства разработки

Android Studio — это интегрированная среда разработки (IDE) для работы с платформой Android, анонсированная 16 мая 2013 года на конференции Google I/O.

IDE находилась в свободном доступе начиная с версии 0.1, опубликованной в мае 2013, а затем перешла в стадию бета-тестирования, начиная с версии 0.8, которая была выпущена в июне 2014 года. Первая стабильная версия 1.0 была выпущена в декабре 2014 года, тогда же прекратилась поддержка плагина Android Development Tools (ADT) для Eclipse.

Android Studio, основанная на программном обеспечении IntelliJ IDEA от компании JetBrains, официальное средство разработки Android приложений. Данная среда разработки доступна для Windows, OS X и Linux. 17 мая 2017 на ежегодной конференции Google I/O, Google анонсировал язык Kotlin используемый в Android Studio официальным языком программирования для платформы Android в добавление к Java и C++.

Android Studio – мощная среда разработки мобильных приложений под Android, которая являет собой набор интегрированных инструментов для эффективной разработки, отладки и тестирования программ. Это позволит нам реализовать все задуманные функции, а так же добавить что то новое уже по ходу разработки. Благодарю выбору такой гибкой платформы, мы сможем создать приложение, которое в дальнейшем сможешь масштабировать. Что это значит? Это означает, что в первой его версии мы увидим достаточный функционал для ведения диалога между учителем и родителями, но в основе концепции приложения мы сможем заложить дальнейшее развитие, что позволит создать диалог между учителями различных предметов в рамках одной школы, руководства школы с учителями, для объявления и закрепления новостей.

В конечном счёте, приложение предполагает под собой объединение всех школа города в одну живую систему, которая будет разделена на категории по праву доступа, учителя, директора, родители и так далее, позволит своевременно оповещать большое количество людей актуальной и достоверной информацией. Это поможет решить множество проблем и подготовить базу, для создания мощного инструмента помощи современной образовательной системе.

2 РАЗРАБОТКА ПРИЛОЖЕНИЯ "ПОМОЩНИК УЧИТЕЛЯ"

2.1 Структура приложения

В структуре приложения описано основное меню, которые представляется пользователю после запуска программы.

Категории меню разделены на две группы, а так же представлены на рисунке (Рисунок 1):

- Группа активного использования
- Группа служебная для управления приложением

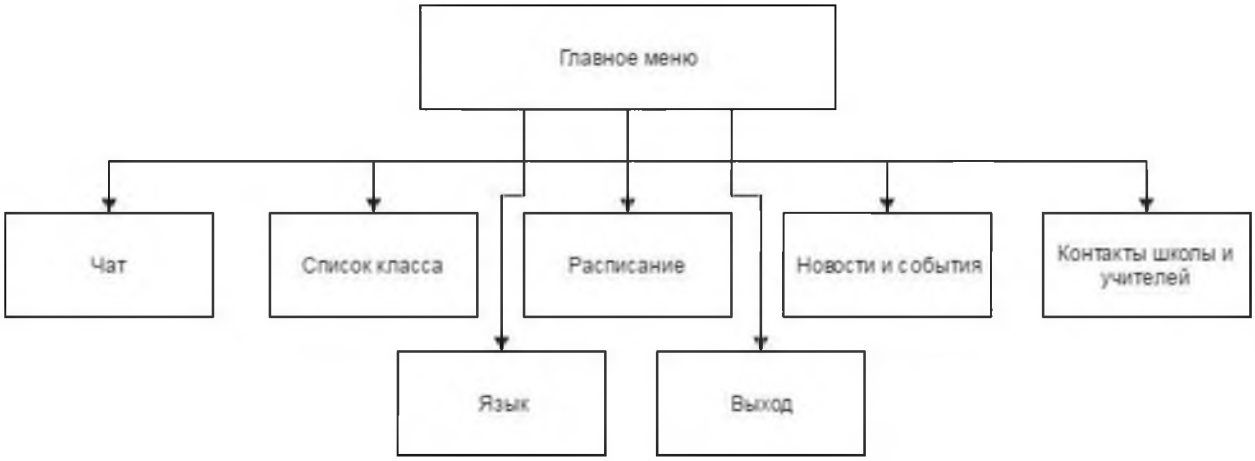

Рисунок 1 Структура меню

Как мы видим, структура меню достаточно простая, но она многофункциональна и сделана в виде модулей, которые мы можем подключать к уже созданному списку навигации. Это сделано для того, что бы в последующем можно было легко расширить функционал приложения, добавив в него то, что нам нужно. Так же в дальнейшем в служебную группу меню планируется добавить пункт с настройками, в котором пользователь сможет выбрать размер, и тип шрифтов, а так же настроить цветовую гамму приложения.

2.2 Возможности приложения

Давайте поближе рассмотрим, что на в настоящий момент включает в себе реализованная платформа.

При открытии главное окна приложения мы попадаем в навигационный интерфейс называемый «Navdrawer», в третей главе мы рассмотрим поближе с помощью чего он реализован.

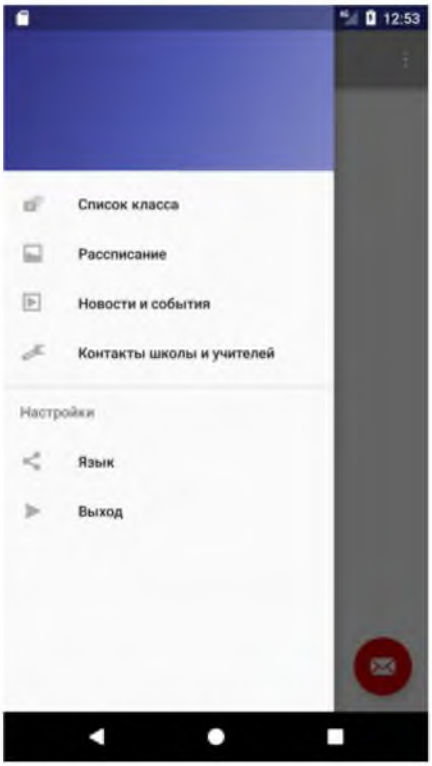

Рисунок 2 Навигационное меню "Navdrawer"

Из главного окна мы можем попасть в интересующий нас модуль, а так же произвести настройку приложения.

Такой навигационный интерфейс предполагает последующее развитие приложения путём добавления в него новых модулей.

Давайте теперь попробуем перейти в пункт меню «Новости и события».

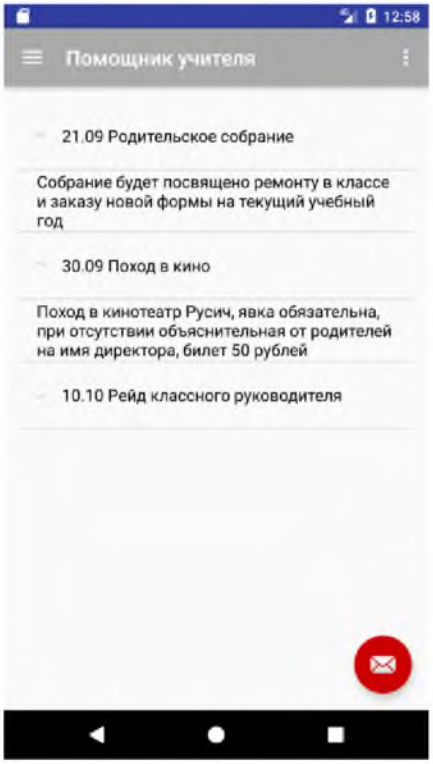

Рисунок 3 Модуль "Новости и события"

Как мы видим, мы попадаем в модуль, который представляет из себя новостную ленту с прошедшими, настоящими и будущими событиями классной жизни. События отсортированы по дате проведения. Сейчас мы видим реализованный многоуровневый, выпадающий список с текстом внутри. Дальнейшее развитие предполагает возможность закрепления графической информации, представленной картинками и фотографиями.

Так же планируется добавление фильтров для сортировки событий, по дате проведения, а так же возможность скрывать прошедшие мероприятия, возможность выделять цветом прошедшие и будущие события и закреплять срочные новости вверху ленты.

Следующий пункт меню, который мы рассмотрим - «Расписание»

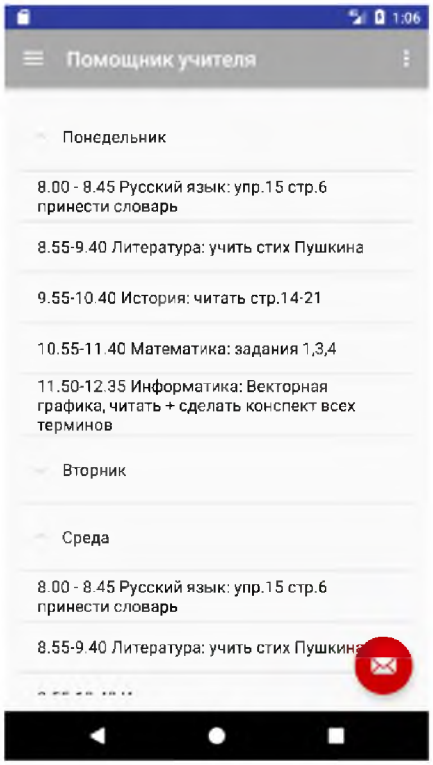

Рисунок 4 Модуль "Расписание"

По своей реализации этот модуль похож на предыдущий, с тем различием, что здесь у нас идёт не новостная лента, а закреплённый формат расписания, в котором меняется содержимое выпадающего списка. Домашнее задание, а так же возможные изменения в расписании.

2.3 Реализация чата

Рассмотрим подробнее создание модуля чат на андроид, используя сервис Firebase. Это backend service от Google.

Чат будет использовать авторизацию по email. После авторизации открывается экран с полем ввода, кнопкой отправки и списком сообщений. В этом списке отображаются все отправленные сообщения на всех устройствах, где установлено данное приложение.

Нам нужно связать проект с сервисом Firebase. Для этого перейдем в меню Tools/Firebase. Выберем вкладку Cloud Messaging. Здесь нужно выполнить 2 первых пункта.

Нажатие первой кнопки свяжет наш проект с сервисом Firebase. При этом нам будет предложено авторизоваться с помощью учетной записи Google.

В случае успеха вместо кнопки появится зеленый значок «connected».

А в консоли разработчика по адресу https://console.firebase.google.com мы увидим новое приложение.

Теперь нужно добавить в проект необходимые зависимости. Второй пункт добавит в файлы сборки проекта ссылки на библиотеки google-services и firebase-messaging.

А в папке модуля app появится файл google-services.json с параметрами, необходимыми для работы проекта с Firebase.

Проект мы подключили, но библиотека firebase-messaging — не совсем то, что нам нужно. Идем в файл сборки пакета build.gradle и заменим ее на библиотеку firebase-ui. Минимальный уровень API, с которым работает эта библиотека — API 16. Изменим соответствующую директиву и синхронизируем с gradle.

Фрагмент кода build.gradle (app)

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 25
    buildToolsVersion "25.0.1"
    defaultConfig {
        applicationId "info.fandroid.firebasechat"
        minSdkVersion 16
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner
"android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-
core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:25.0.1'
```

```
    testCompile 'junit:junit:4.12'

    //Add Library
    compile 'com.android.support:design:25.0.1'
    compile 'com.firebaseui:firebase-ui:0.6.2'
}
apply plugin:  'com.google.gms.google-services'
```

Теперь перейдем к описанию процедур.

Для начала создадим макет разметки главного экрана, нам поможет поле ввода, кнопка отправки сообщений и виджет списка ListView.

Фрагмент кода окна регистрации в чате activity-main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/activity_main"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context="info.fandroid.firebasechat.MainActivity">


   <ListView
      android:id="@+id/listView"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:layout_below="@+id/button2"
```

```
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"  />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPersonName"
        android:ems="10"
        android:id="@+id/editText"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_above="@+id/listView"
        android:layout_toLeftOf="@+id/button2"
        android:layout_toStartOf="@+id/button2" />

    <Button
        android:text="Send"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button2"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"  />
</RelativeLayout>
```

Теперь создадим макет разметки пункта списка.

Фрагмент кода главного окна чата item.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:layout_width="match_parent"  android:layout_height="match_parent">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:id="@+id/tvUser" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/tvTime"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/tvMessage"
        android:layout_below="@+id/tvUser"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />
</RelativeLayout>
```

Здесь мы видим три поля Textview для имени автора, времени и текста

сообщения. Нам также необходимо в папке res создать папку menu и в ней описать пункт меню для выхода из учетной записи.

Фрагмент кода menu.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item android:icon="@drawable/logout"
        app:showAsAction="always"
        android:id="@+id/menu_signout"/>


</menu>
```

Зададим чату иконку из стандартного инструментала Android studio. Также пропишем способ отображения в тулбаре. Атрибут showAsAction берем из пространства имен app, добавим соответствующую декларацию для этого комбинацией Alt+Enter.

Теперь в основном пакете создадим новый класс Message. Это будет макет, или модель сообщения.

Фрагмент кода Message.java.

```
package info.fandroid.firebasechat;


import java.util.Date;

public class Message {

    private String textMessage;
    private String autor;
```

```
private long timeMessage;

public Message(String textMessage, String autor) {
    this.textMessage = textMessage;
    this.autor = autor;

    timeMessage = new Date().getTime();
}

public Message() {
}

public String getTextMessage() {
    return textMessage;
}

public void setTextMessage(String textMessage) {
    this.textMessage = textMessage;
}

public String getAutor() {
    return autor;
}

public void setAutor(String autor) {
    this.autor = autor;
}

public long getTimeMessage() {
    return timeMessage;
```

```
    }

    public void setTimeMessage(long timeMessage) {
        this.timeMessage = timeMessage;
    }
}
```

Создадим переменные textMesage, autorMessage и timeMessage. Не трудно догадаться, это текст, автор и время сообщения.

Создадим конструктор с первыми двумя переменными. Используется комбинация Alt+Insert. В этом же конструкторе будем сохранять в переменную timeMessage - текущее время.

Также создадим пустой конструктор, а также геттеры и сеттеры для всех полей класса.

Основной код будет написан в MainActivity.

Фрагмент кода MainActivity.java.

```
package info.fandroid.firebasechat;

import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.text.format.DateFormat;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
```

```
import android.widget.RelativeLayout;
import android.widget.TextView;

import com.firebase.ui.auth.AuthUI;
import com.firebase.ui.database.FirebaseListAdapter;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.FirebaseDatabase;

public class MainActivity  extends AppCompatActivity {

    private static int SIGN_IN_REQUEST_CODE = 1;
    private FirebaseListAdapter<Message> adapter;
    RelativeLayout activity_main;
    Button button;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        activity_main  = (RelativeLayout)findViewById(R.id.activity_main);
        button = (Button)findViewById(R.id.button2);
        button.setOnClickListener(new  View.OnClickListener()  {
            @Override
            public void onClick(View  view) {
                EditText input = (EditText)findViewById(R.id.editText);
                FirebaseDatabase.getInstance().getReference().push()
                        .setValue(new  Message(input.getText().toString(),
```

```
                    FirebaseAuth.getInstance().getCurrentUser().getEmail()));
            input.setText("");
        }
    });

    if (FirebaseAuth.getInstance().getCurrentUser() == null) {
        startActivityForResult(AuthUI.getInstance()
            .createSignInIntentBuilder()
            .build(), SIGN_IN_REQUEST_CODE);
    } else {
        displayChat();
    }
}

private void displayChat() {

    ListView listMessages = (ListView)findViewById(R.id.listView);
    adapter = new FirebaseListAdapter<Message>(this, Message.class,
R.layout.item, FirebaseDatabase.getInstance().getReference()) {
        @Override
        protected void populateView(View v, Message model, int position) {

            TextView textMessage, autor, timeMessage;
            textMessage = (TextView)v.findViewById(R.id.tvMessage);
            autor = (TextView)v.findViewById(R.id.tvUser);
            timeMessage = (TextView)v.findViewById(R.id.tvTime);

            textMessage.setText(model.getTextMessage());
            autor.setText(model.getAutor());
```

```
            timeMessage.setText(DateFormat.format("dd-MM-yyyy (HH:mm:ss)",
model.getTimeMessage()));
        }
    };
    listMessages.setAdapter(adapter);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == SIGN_IN_REQUEST_CODE)
    {
        if (resultCode == RESULT_OK)
        {
            Snackbar.make(activity_main, "Вход выполнен",
Snackbar.LENGTH_SHORT).show();
            displayChat();
        } else {
            Snackbar.make(activity_main, "Вход не выполнен",
Snackbar.LENGTH_SHORT).show();
            finish();
        }
    }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu, menu);
    return true;
}
```

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == R.id.menu_signout)
        {
            AuthUI.getInstance().signOut(this)
                    .addOnCompleteListener(new OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {

                            Snackbar.make(activity_main, "Выход выполнен",
Snackbar.LENGTH_SHORT).show();
                            finish();

                        }
                    });
        }
        return true;
    }
}
```

Для начала создадим константу SIGN_IN_REQUEST_CODE со значением 1. Далее создаем переменную класса FirebaseListAdapter — это дженерик(шаблон), который обеспечивает поддержку списка сообщений. В качестве параметризированного типа у него будет наш класс Message. Далее объявляем корневой макет экрана и кнопку.

В методе onCreate находим кнопку и корневой RelativeLayout по ID и присваиваем кнопке обработчик нажатия.

В методе onClick определяем поле ввода.

Далее считываем текст из поля ввода и отправляем новый экземпляр сообщения в базу данных Firebase. Но, прежде чем отправить сообщение. пользователь должен авторизоваться. А если пользователь не авторизован, то ему нужно показать форму авторизации, а не экран чата.

Для этого создадим экран авторизации с помощью метода startActivityForResult, которому мы передаем интент, создающий и настраивающий окно авторизации, а также константу, хранящую код авторизации.

Создавать окно авторизации мы будем через проверку авторизации пользователя. Обернем этот метод в блок if... else комбинацией Ctrl+Alt+T и пропишем соответствующую проверку. Если же пользователь авторизован, будем показывать ему экран чата со списком сообщений.

Для этого мы создадим метод displayChat и будем вызывать его здесь.

В методе displayChat создаем список сообщений. Также создаем адаптер списка, используя класс FirebaseListAdapter. Передаем ему контекст, класс модели сообщения, макет пункта списка и экземпляр базы данных Firebase. Далее в автоматически созданном методе populateView, заполняем пункты списка.

Сначала определяем поля пункта списка по ID. Затем прописываем текст сообщения, имя пользователя. Также устанавливаем формат даты и отображаем ее. Обратите внимание — нужно использовать именно этот класс DateFormat. И наконец, передаем адаптер списку.

Также нам нужно будет показать окно чата после окна авторизации в случае ее успеха. Для этого мы переопределим метод onActivityResult.

В двух словах, в метод onActivityResult приходит результат вызова Activity методом startActivityForResult, которым мы вызываем здесь окно авторизации. Сначала вызываем метод суперкласса. затем проверям, что значение requestCode равно константе SIGN_IN_REQUEST_CODE, которую мы передаем в методе startActivityForResult. Затем мы проверяем, что вызов активити прошел успешно, и отображаем окно чата после оповещения

пользователя об удачном входе. В противном случае показываем уведомление о неудаче пользователю.

И теперь нам осталось реализовать выход пользователя из чата. Сделаем это через меню. Создаем меню в методе onCreateOptionsMenu и переопределяем метод onOptionsItemSelected, где проверяем выбранный пользователем пункт и реализуем выход пользователя из учетной записи чата. В случае успеха отображаем снекбар с уведомлением.

В нашем приложении будет использоваться авторизация по email. Ее нужно активировать в консоли Firebase.

Что бы запуск прошёл успешно, последнее что нам нужно сделать это включить Identity Toolkit API в консоли Google разработчика.

После этого наш модуль чата готов к подключению в основную платформу.

3 ДЕМОНСТРАЦИЯ И ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЯ

3.1 Демонстрация работы

Работа с программой достаточно проста, достаточно произвести стандартную процедуру установки .apk файла на android устройство. После установки, в главном меню телефона появится ярлык приложения. Кликнув по нему, произойдёт открытие главного меню приложения.

Нажатие на кнопку меню, на главном экране приложения, откроет перед нами окно навигации по модулям (Рисунок 5).

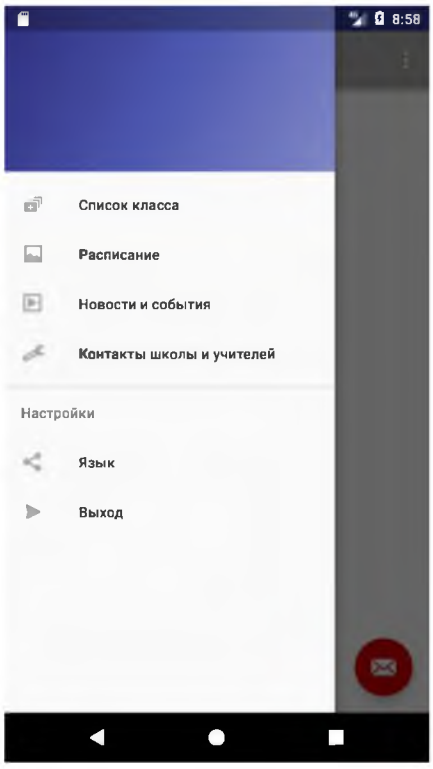

Рисунок 5 Главное меню выбора модуля

Нажатие на любой пункт меню переведёт пользователя в выбранный модуль. Что бы выйти из выбранного модуля, достаточно кликнуть копку меню, она доступна из любого модуля, на рисунке ниже, выделена красным цветом (Рисунок 6).

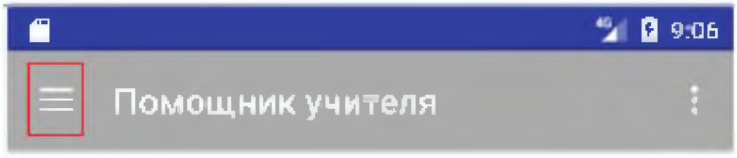

Рисунок 6 Внешний вид кнопки "Меню навигации"

Для выхода из приложения можно воспользоваться пунктом выход, в меню навигации.

3.2 Инструкция пользователя

Рассмотри пункты меню навигации поближе.

Список класса.

В этом меню пользователь может ознакомиться с списком класса. По желанию к каждому контакту может быть добавлен пункт с номером телефона, местом проживания, а так же личной информацией.

Расписание.

Пункт меню «Расписание» предлагает пользователю неделю выбрать день недели для просмотра и ознакомления с предложенной информацией. В выпадающем списке дня недели описано расписание уроков, время их начала и окончания, а так же актуальное домашнее задание, на каждый день недели.

Новости и события.

Пункт меню «Новости и события» отвечает за предоставление информации по прошедшим, текущим и предстоящим мероприятиям. Пользователь может увидеть название мероприятия, дату его проведения, а так же описание события.

Контакты школы и учителей.

Пункт меню «Контакты школы и учителей» позволяет пользователю получить информацию о преподавательском составе того или иного заведения, получить номер интересующего сотрудника, будь то классный руководитель, завуч, школьный психолог или директор.

Настройки.

Пункт меню «Язык» позволяет произвести переключение между русским и английским языком

Выход.

Пункт меню «Выход» позволяет выйти из приложения.

ЗАКЛЮЧЕНИЕ

В заключении хотелось бы сказать, что в данной дипломной работе мы ставили перед собой целью создание платформы для эффективной связи системы образования и родителей, которая позволила бы вести активный диалог между современной системой образования и родителями.

Конечно в существующем виде приложение не может стать соперником таких современных существующих систем как «Viber» или «ВКонтакте». Тем не менее, плюсы созданного приложения могут дать задел на будущее, тот факт, что платформа может быть переработана и дополнена под современные нужды образования, а так же в теории может быть подкреплена законодательно, то есть не может быть запрещена, могут положительно повлиять на дальнейшее развитие.

Нет сомнений в том, что если современная система образования хочет развиваться, ей необходимо развивать себя в сторону взаимодействия пользователя через мобильные приложения. На это указывают все современные тенденции.

Данное приложение может выступить платформой для шага в этот сегмент. Такое решение поможет улучшить взаимодействие учителей с родителями и позволит сэкономить время и силы, необходимые для оповещения и связи с родителями. Это непременно повлияет на качество образования и поможет соответствовать современным требованиям и реалиям.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Блог о разработке под Android [Электронный ресурс]// URL: http://habrahabr.ru/blogs/android_development/

2. Видеоуроки по разработке на Android [Электронный ресурс]// URL: https://www.youtube.com/channel/UCORRUYUmW1pffMgLPzU0XCA

3. Гуриков С. Р. Введение в программирование на языке Visual C#; Форум, Инфра-М, 2013. - 448 с.

4. Джошуа Блох. Java. Эффективное программирование = Effective Java. — М.: «Лори», 2002. — С. 224. — ISBN 5-85582-169-2

5. Кей С. Хорстманн, Гари Корнелл. Java 2. Библиотека профессионала, том 1. Основы = Core Java™ 2, Volume I--Fundamentals. — 7-е изд. — М.: «Вильямс», 2007. — С. 896. — ISBN 0-13-148202-5

6. Мартин Р. С., Мартин М. Принципы, паттерны и методики гибкой разработки на языке C#; Символ-Плюс, 2011. - 768 с.

7. Монахов Вадим Язык программирования Java и среда NetBeans, 2-е издание. — СПб.: «БХВ-Петербург», 2009. — С. 720. — ISBN 978-5-9775-0424-9

8. Официальная справка для Android разработчиков [Электронный ресурс] // URL http://developer.android.com/index.html

9. Официальная справка по среде программирования [Электронный ресурс]// URL: http://www.jetbrains.com

10. Программирование под Android / Блэйк Мик . – СПб.: Санкт-Петербург, 2012. – 496 с.

11. Программирование для Android. Самоучитель / Денис Колиснеченко . – СПб.: Санкт-Петербург, 2011. – 272 с.

12. Программирование приложений для планшетных компьютеров и смартфонов / Рето Майер . – СПб.: Санкт-Петербург, 2011. – 672 с.

13. Программирование приложений для планшетных компьютеров и смартфонов (Рето Майер, Эксмо, 2011)

14. Программирование для мобильных устройств (Голощапов А.Л., 2011, BHV Санкт-Петербург)

15. Пугачев С., Шериев А., Кичинский К. Разработка приложений для Windows 8 на языке C#; БХВ-Петербург, 2013. - 416 с.

16. Разработка приложений для Android (С. Хашими, С. Коматинени, Д. Маклинг, 2011)

17. Смартфоны Android без напряга. Руководство пользователя / Андрей Жвалевский . – СПб.: Санкт-Петербург, 2012. – 224 с.

18. Статьи о программировании для Android [Электронный ресурс]//URL: http://flashbot.ru/android-dev

19. Фленов Михаил Библия C#; БХВ-Петербург, 2009. - 560 с.

20. Программирование для Android. Самоучитель / Колисниченко Д. – СПб.: Санкт-Петербург, 2011. – 736 с.

21. Форум о программировании для Android [Электронный ресурс]// URL: http://www.cyberforum.ru/android-dev/

22. 8 Форум о программировании для мобильных устройств [Электронный ресурс]// URL: http://www.4pda.ru