МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Прикладной информатики

БАКАЛАВРСКАЯ РАБОТА

На тему

ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ-МЕССЕНДЖЕРА ДЛЯ РАССЫЛКИ ИНФОРМАЦИОННЫХ СООБЩЕНИЙ В ПРЕДПРИЯТИЙ С ИЕРАРХИЧНОЙ СТРУКТУРОЙ

Исполнитель
Макаров Илья Алексеевич

Руководитель
кандидат технических наук
Слесарева Людмила Сергеевна

«К защите допускаю»
Заведующий кафедрой

кандидат технических наук,
Слесарева Людмила Сергеевна

«10» 06 2016г.

Санкт–Петербург
2016

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Прикладной информатики

БАКАЛАВРСКАЯ РАБОТА

На тему

ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ-МЕССЕНДЖЕРА ДЛЯ РАССЫЛКИ ИНФОРМАЦИОННЫХ СООБЩЕНИЙ В ПРЕДПРИЯТИЙ С ИЕРАРХИЧНОЙ СТРУКТУРОЙ

Исполнитель
Макаров Илья Алексеевич

Руководитель
кандидат технических наук
Слесарева Людмила Сергеевна

«К защите допускаю»
Заведующий кафедрой

кандидат технических наук,
Слесарева Людмила Сергеевна

«___»_____20__г.

Санкт–Петербург
2016

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Для человека в настоящее время ресурсом, имеющим большое значение, является информация, следовательно от своевременного получения актуальной информации зависит продуктивность при работе. В этой связи использование средств для распространение и донесения информации является актуальным. Данные средства должны быть удобны при использование и выполнять основную функцию – рассылка информационных сообщений.

В ходевыпускной квалификационной работы (ВКР)разрабатываетсяпроект приложения мессенджера, с помощью которого будет осуществляться рассылка информационных сообщений.

Объектом исследования является коммуникативность в группах людей, где подчинённые выполняют или получают информацию от управляющего. Данное приложение рассчитано для предприятий илигрупп людей с иерархичной структурой. Благодаря правильной организации рассылки оповещений и информационных сообщений можно добиться эффективной коммуникативности, что будет благотворно способствовать достижению поставленных целей, что делает данную курсовую работу актуальной на сегодняшний день.

Главной цельюработы являетсяснижение времени на рассылку информационных сообщений и оповещений, в следствии этого повышение эффективности работы предприятия. Главными задачами являются удобство и комфортабельность при использовании, так как каким бы не было полезным и информативным сообщение, при низкой удобочитаемости и неудобстве при работе с приложением снижается эффективность от такого сообщения.

Основным функционалом является рассылка и получение оповещений и информационных сообщений. Также была поставлена задача на проектирование следующих функций:регистрация и аутентификация

пользователей, создание групп пользователей, возможность приглашения пользователей в группы, возможность запроса на публикацию сообщения в группе, если у пользователя недостаточно прав.

В данной ВКР была проделана работа по анализу и изучению предприятий с иерархичной структурой, были отмечены принципы, на которых будет базироваться приложение, был проделан анализ и сравнение аналогов приложений мессенджеров, в ходе работы были спроектированы SADTмодели, а такжеUML-диаграммы, была разработана модель базы данных и был произведен расчет трудозатрат в человеко-днях.

При анализе предметной области можно смело сказать, что данная ВКР не только актуальна на сегодняшний день, но будет актуально еще долгое время. Данная работа выполнялась по инициативе автора. Тема ВКР согласована с кафедрой.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Перед началом работ по проектированию приложения требуется составить план предпроектного обследования. План, условно, можно разделить на две части: первая, которая включает всё, что происходит «за» приложением, то есть внешнюю среду (предприятие, конкретный пользователь, задачи и т.д.) и вторая, где будут включены все аспекты проектирования, внутри приложения, но не углубляясь в непосредственно в программирование, в данном случае эту часть можно понимать, как юзабилити (регистрация, дизайн и т. д.).

Первая часть:

- Цели и задачи;
- Анализ структур предприятий;
- Анализ аналогов;
- Выбор среды разработки;
- Расчет трудозатрат.

Вторая часть:

- Регистрация;
- Функционал приложения;
- Пользовательский интерфейс;
- Концепция приложения.

1.1 Цели и задачи

Перед началом проектирования следует определиться с целью проекта. Определим понятие цели следующим образом, это то как изменится ситуация в результате успешной реализации проекта. В нашем случае целью является сокращение времени на рассылку информационных сообщений и оповещений, а это в следствии приведет к повышению эффективности работы подразделений и предприятия в целом.

Определившись с целью, следует определить задачи проекта. Под задачей надо понимать, конкретные достижения, направленные на решение проблем, которые следует достичь в результате проекта. Можно выделить следующие задачи:

- Простота дизайна и настраиваемый интерфейс;
- Возможность отключение невостребованныхфункций;
- Оптимизация.

1.3 Анализ аналогов

На сегодняшний день на рынке предоставлен огромный выбор мессенджеров с разным функционалом и возможностями, но подобающие большинство из них заточены под общение, поэтому они неудобно при использование в рабочих целях.

Требования, предъявляемые к мессенджерам для командной работы, отличаются от пожеланий пользователей индивидуальных сервисов (вроде Skype или WhatsApp).

Среди наиболее важных параметров мессенджера для корпоративного общения стоит назвать: возможность контроля шума; возможность параллельного общения сразу в нескольких комнатах и организациях; стоимость; удобство форматирования сообщений; количество

интегрированных внешних сервисов и их ассортимент; лёгкость работы в системе; наличие мобильных и нативных клиентов; надежность работы; политику в отношении доступа к данным.

Одновременнаяработа в разных командах. Еще один важный момент — общение в нескольких командах. Довольно часто пользователь приложения-мессенджера должен работать более чем в одной команде (например, внешний консультант или менеджер проектов в большой компании).

В системе Slack поддержка нескольких команд существует, но пользователю каждый раз приходится явным образом выбирать “текущую” команду, чтобы взаимодействовать с всеми ее членами.

Таким образом следует учесть все требования к современным мессенджерам, применяемых при работе, а так же сделать его достаточно удобным при использованием, чтобы при выборе он был предпочтительным для пользователей.

1.6 Выбор среды разработки

Лучшей средой для разработки под Androidявляется AndroidStudio,но для сравнения рассмотрим и EclipseIDE. Сравним их именнокак Java IDE.

Сравним их по нескольким параметрам по которым станет очевидно, что AndroidStudiokуда предпочтительнее:

- Отладка;

Для того чтобы посмотреть значение какого-либо выраженияпри отладке, в Eclipse необходимо обязательно сначала выделить это выражение (рис 1.6). Следует заметить, что выделить требуется точно, если же выделить случайно даже один лишний символ— Eclipse не сможет понять, что вы хотите. После этого нажатием комбинации клавишCtrl+Shift+Iможно увидеть значение выражения.

```
@Test
public void displaysHumanReadableName() {
  Assert.assertEquals("become visible", Condition.visible.toString());
  assertEquals("become hidden", Condition.hidden.toString());
  assertEquals("got attribute lastName=Malkovich", Condition.hasAttribute("lastName", "Malkovich").toString());
}
```

Рисунок 1.6 Отладка в Eclipse

В AndroidStudioвсего лишь необходимо в нужное место поставить курсор, выделять же ничего не надо(на рисунке 1.7 в случае на методе hasAttribute) и нажать Alt+F8. AndroidStudio сама понимает, какое выражение вам, скорей всего, нужно, и сразу же откроется диалог, где вы можно изменять выражение и сразу увидеть его значение (рис 1.7).

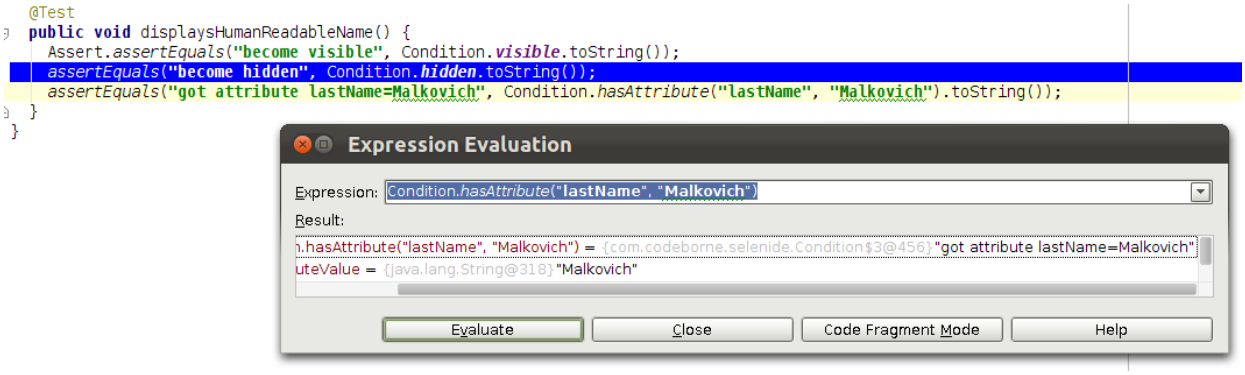


Рисунок 1.7Отладка в AndroidStudio

Такимо образом обе IDE в принципе позволяют делать одно и то же. Но в AndroidStudio это намного удобнее и быстрее. Разница между ними огромная. В этом небольшом окне AndroidStudioможет сделать и автозаполнение, и подсветку синтаксиса, и множество других функций.

- Автозаполнение (autocomplete);

Автозаполнение — это то, что отличает любую IDE от notepad в лучшем свете. AndroidStudiodаёт качественное преимуществов области «понимание контекста». Предположим, сначала написали строчку кода«assertElement(By.id("errorMessage"), vi». Потом, хотим узнать какие варианты есть, и что там может начинаться с букв «vi».

AndroidStudio,не дожидаясь нажатий клавиш, она сразу же понимает контекст, и понимает что метод assertElement хочет получить вторым параметром объект класса Condition, а в этом классе как раз есть статическая переменная типа Condition с именем visible. И предлагает единственный возможный вариант(рис 1.8).

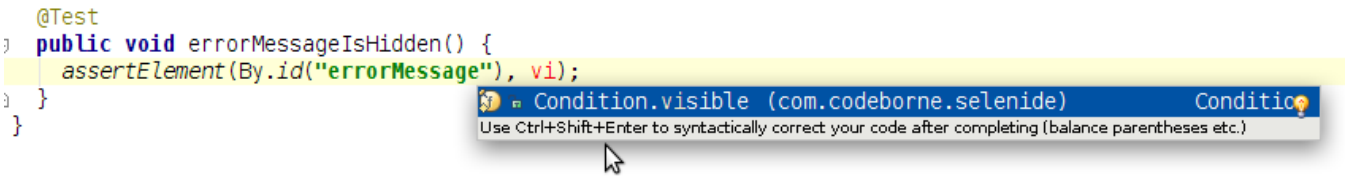


Рисунок 1.8Автозаполнение вAndroidStudio

В случае другая ситуация Eclipse, он не понимает контекста. Он не знает, что курсор находится на месте второго параметра метода assertElement. Именно поэтому, когда вы нажимаете заветные Ctrl+Space, Eclipseпоказывает всё, что начинается на буквы «vi» (рис 1.9).

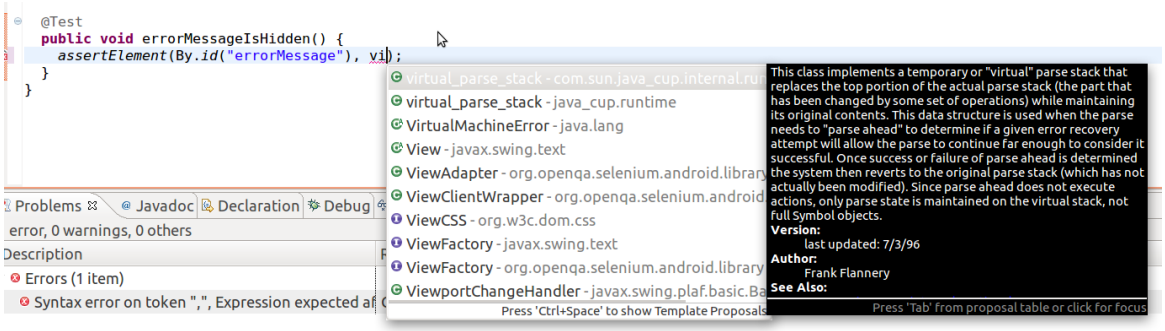


Рисунок 1.9Автозаполнение вEclipse

В этом всплывающем окне можно увидеть много подсвеченной хорошо задокументированной, но бесполезнойинформации.

- Рефакторинг;

Профессиональные программисты для того чтобы быть продуктивными, используют для изменения кода те рефакторинги, которые

их IDE предлагает им. Все современные IDE в принципе предлагают весьма впечатляющий набор рефакторингов, наверное, даже большинство программистов не знает и не использует все из них. Но опять же, рефакторинги в AndroidStudio интеллектуальные. Они догадываются, чего вы хотите, и предлагают разные варианты, которые в большинстве ситуаций подходят.

Например, естьунасметодassertErrorMessageIsHidden инадо сделать так, чтобы строка «errorMessage» была в методе как параметр.

Начнём с AndroidStudio. Поставим курсор в любое место в строке «errorMessage», после нажатием клавиш Ctrl+Alt+P (от «parameter»), и AndroidStudio подсказывает, какое выражение можно было бы вынести в параметр (рис 1.10).

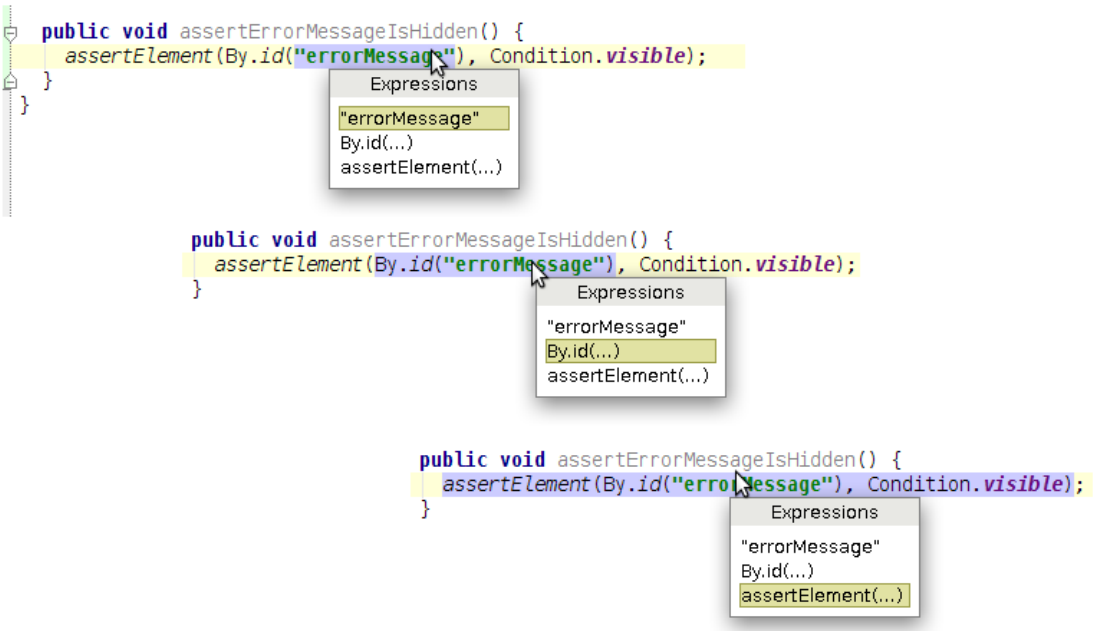


Рисунок 1.10Рефакторинг вEclipse

Сразу после как выражение «errorMessage» выбрано, AndroidStudio подсказывает несколько возможных имён для этого параметра(рис 1.10).

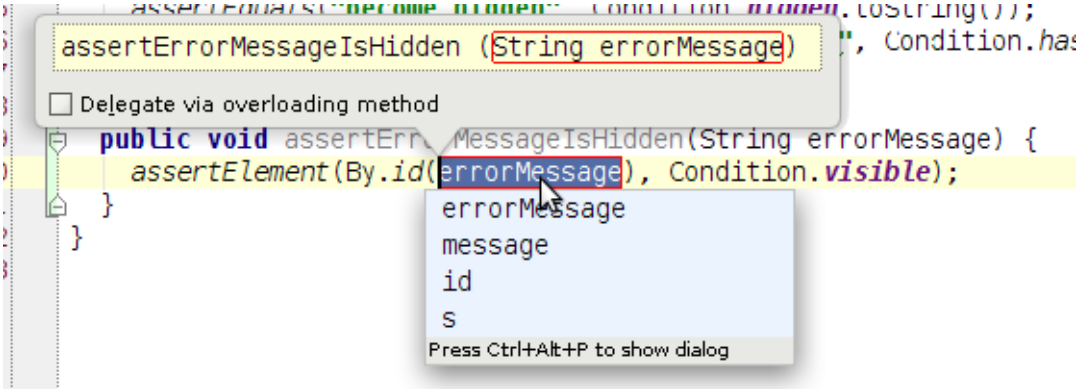


Рисунок 1.11РефакторингAndroidStudio

AndroidStudioучитывает, и тип переменной, и значение, и названия подобных переменных в других местах, и те названия, которые вы давали подобным переменным раньше,и название метода.

Eclipse в свою очередь предлагает нам следующее — Следует не забыть выделить выражение «errorMessage» (непременно с кавычками, если не хотите получить сообщение об ошибке «Anexpressionmustbeselecttoactivatethisrefactoring»), дальше надо выбрать рефакторинг «Introduceparameter» (так как горячей клавиши для этого нет, то выбираем из меню), и получатся тот же результат. А вариантов имени параметра Eclipse не предлагает никаких (рис 1.11).

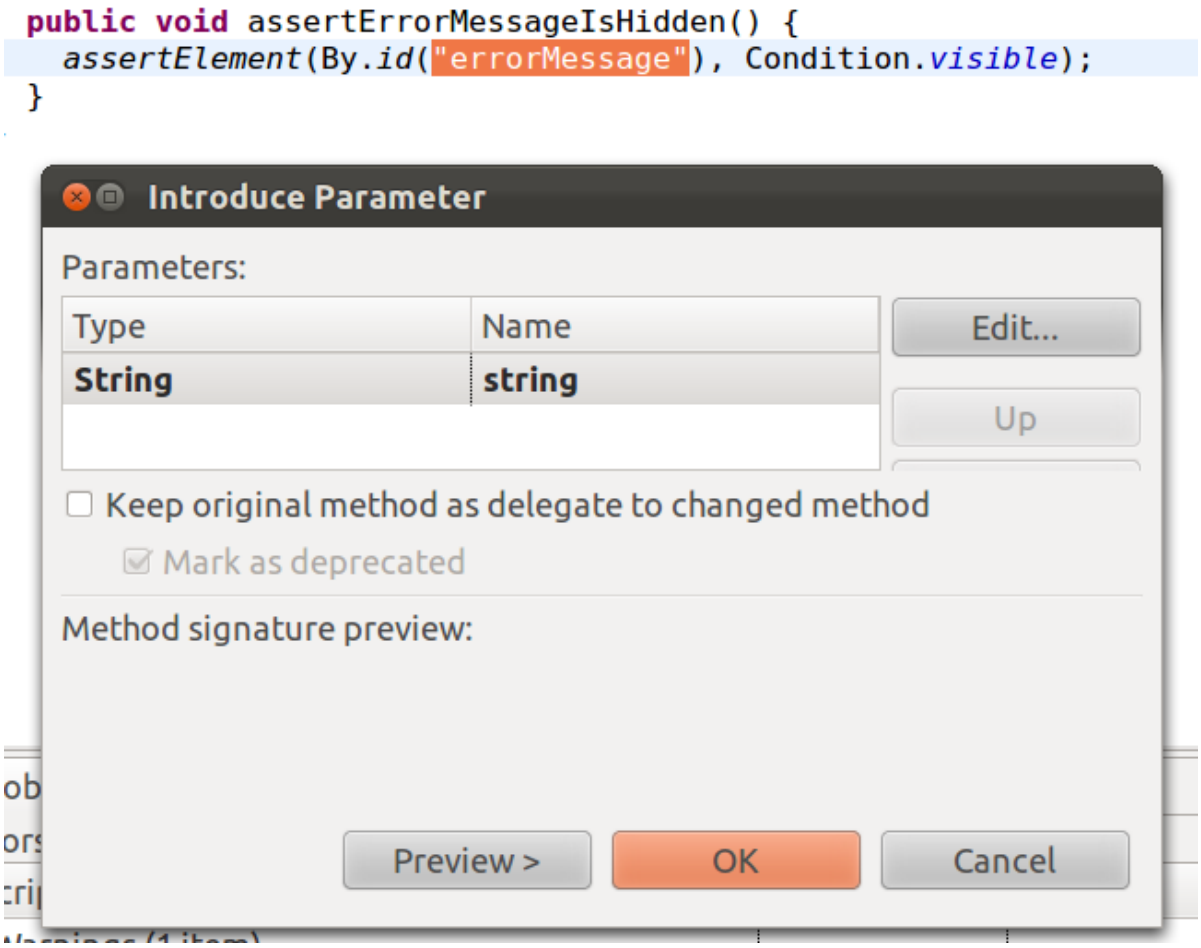


Рисунок 1.11РефакторингEclipse

Следовательно, можно сделать вывод, если говорить о Java IDEдля Android, тоEclipseпроигрывает AndroidStudio. AndroidStudio объективно и справедливо лучше. Она подсказывает подходящие имена, позволяет быстрее и качественнее писать и менять код, находит нужные в данный момент методы. Предугадывает, по тому месту, где вы находитесь курсором, что вы хотели сделать и как хотели это назвать, ане требует точно выделить это выражение, а AndroidStudio предугадывает и подсказывает.

1.7Об Android

Независимо от того, опытный вы инженер в области мобильных приложений, разработчик компьютерных программ, веб-программист или

любитель, Android предоставляет отличную возможность по написанию инновационных приложений для мобильных устройств. Несмотря на название, Android не предназначен для создания несокрушимой армии хладнокровных роботов-солдат с целью очистить Землю от гнета человечества. Android представляет собой набор программ с открытым исходным кодом, который включает операционную систему, подпрограммное обеспечение и ключевые мобильные приложения вместе с библиотеками API, предназначенными для написания новых программ, определяющих визуальное представление и функционал мобильных устройств. Самые разнообразные компактные стильные мобильные устройства со временем снабжались такими мощными инструментами, как камера, медиаплеер, навигатор, сенсорный дисплей. С внедрением новых технологий мобильный телефон превратился в нечто большее, чем просто устройство для звонков. При этом программная платформа и среда разработки отставали от бешеного темпа развития технологий. До сегодняшнего дня мобильные телефоны работали под управлением закрытых платформ, построенных на основе сильно фрагментированных запатентованных операционных систем, для чего требовались запатентованные инструменты разработки. Сами же телефоны функционировали с оригинальным программным обеспечением гораздо лучше, чем со сторонним.

Это создавало искусственные препятствия для программистов, которые рассчитывали на использование более мощного аппаратного обеспечения мобильных устройств. В случае с Android встроенное ПО написано на том же самом API, что и программы сторонних разработчиков, при этом время для исполнения и тех, и других одинаково. Данное API позволяет получить доступ к сенсорному управлению, навигационным сервисам, фоновым и картографическим процессам, реляционным базам данных, двух- и трехмерной графике, к функциям видеозаписи, межпрограммного взаимодействия. В данной книге вы познакомитесь с функционалом API, что

позволит разрабатывать приложения для платформы Android. В настоящей главе мы рассмотрим общие принципы создания мобильных программ, а также изучим основные возможности среды разработки. В распоряжении разработчика приложений для Android достаточно мощное API и качественная справочная документация. Он может стать членом огромного сообщества, ему не нужно тратиться на программное обеспечение или рекламу своего продукта. С ростом популярности мобильных устройств открываются великолепные перспективы разрабатывать инновационные мобильные приложения, причем с любым опытом программирования.

Небольшая предыстория. Задолго до сетей Twitter и Facebook, когда Google был всего лишь идеей в головах его создателей, а по Земле бродили динозавры, мобильный телефон представлял собой переносное устройство связи, достаточно компактное, чтобы поместиться в чемодане, а его батарейки хватало на несколько часов работы. Тем не менее он давал достаточно свободы, чтобы совершать звонки без физического подключения к телефонной линии. Теперь в нашу жизнь прочно вошли компактные, стильные и мощные мобильные телефоны, став незаменимой вещью. Благодаря развитию электроники телефоны стали меньше и функциональнее, а число входящих в них периферийных устройств выросло. После добавления камер и плееров появились телефоны с GPS-навигаторами, акселерометрами и сенсорными дисплеями. Казалось, что рост аппаратных возможностей подготовил благодатную почву для разработчиков программного обеспечения, однако на деле приложения для мобильных телефонов в своем развитии сильно отстали от их аппаратной части.

Не такое далекое прошлое. Исторически сложилось, что программисты, работающие на низкоуровневом C или C++, должны были разбираться в специфике устройств, для которых они создавали программное обеспечение (будь то одно устройство или целая их серия любого производителя). С развитием аппаратных возможностей и расширением доступа к мобильному Интернету данный подход стал неактуальным. В последнее время созданы

платформы, например Symbian, обеспечивающие разработчикам доступ к более широкой аудитории потребителей ПО. Эти системы подталкивали программистов к написанию большого количества приложений, которые эффективно использовали имеющиеся аппаратные средства. Созданные платформы открыли доступ к некоторым устройствам аппаратной части, но для этого требовалось писать полноценный код C/C++ с использованием запатентованного «тяжелого» API, работа с которым была чрезвычайно сложной. Дополнительные трудности возникали при функционировании приложений на устройствах с различным исполнением аппаратной начинки, в особенности это касалось устройств с GPSнавигаторами. За последние годы настоящим прорывом в развитии ПО для мобильных телефонов стало изобретение мидлетов для платформы Java. Мидлеты исполняются на виртуальной Java-машине, что позволяет абстрагироваться от архитектуры того или иного устройства и создавать приложения, работающие на любом из них, которое поддерживает Java. К сожалению, за такое удобство приходится платить ограниченными возможностями доступа к аппаратной части. В программировании ПО для мобильных устройств совершенно нормально, что приложения сторонних разработчиков получают доступ к аппаратной начинке, при этом им выделяются такие же права на исполнение, как и встроенному ПО, которое было разработано производителем телефона. К сожалению, в случае с мидлетами обе эти возможности ограничены. Изобретение мидлетов для платформы Java привлекло большое количество разработчиков, однако отсутствие низкоуровневого доступа к аппаратной части и ограниченные возможности исполнения кода означали, что большинство мобильных приложений представляют собой стандартные desktop-приложения или специальные веб-сайты, приспособленные к отображению на небольших экранах. Таким образом, мобильные приложения никак не использовали преимущества и возможности портативных платформ.

Будущее.Android — одна из операционных систем нового поколения, созданных для работы с аппаратным обеспечением современных мобильных

устройств. На сегодняшний день WindowsMobile, AppleiPhone и PalmPre предлагают достаточно мощные и более простые в использовании среды разработки мобильных приложений. Однако в отличие от Android это запатентованные операционные системы, в которых в определенных случаях приоритет отдается встроенному ПО, а не приложениям сторонних программистов. Кроме того, эти операционные системы ограничивают возможности взаимодействия приложений с данными телефона, а также ограничивают или контролируют процесс распространения сторонних приложений, созданных для данных платформ. Android дает новые возможности для мобильных приложений, предлагая открытую среду разработки, построенную на открытом ядре Linux. У всех приложений есть доступ к аппаратным средствам устройства, для чего используются специальные серии API-библиотек. Кроме того, здесь включена полная и контролируемая поддержка взаимодействия приложений. На платформе Android все программы имеют одинаковый статус. Сторонние приложения написаны на том же API, что и встроенное ПО, при этом во всех программах одинаковое время исполнения. Пользователи могут удалять или заменять встроенные ПО на альтернативные сторонние разработки, будь то номеронабиратель или рабочий стол.

Android: открытая платформа для разработки мобильных приложений. Вот как описывает Android Энди Рубин из Google: «Первая действительно открытая и всеобъемлющая платформа для мобильных устройств и любого программного обеспечения, предназначенного для работы на мобильном телефоне, при этом без патентных ограничений, которые сдерживали развитие портативных устройств.»

Упрощенно Android можно представить как комбинацию трех компонентов:

- свободной операционной системы с открытыми исходными кодами;

- среды разработки с открытыми исходными кодами для создания мобильных приложений;
- устройств, по большей части мобильных телефонов, на которых установлена операционная система Android вместе с разработанными для нее приложениями. Android включает несколько необходимых и взаимозависимых элементов;
- референс-дизайн аппаратного обеспечения с перечнем требований к мобильным устройствам, чтобы гарантировать совместимость с ПО;
- ядро операционной системы Linux, которое предоставляет низкоуровневый интерфейс для управления аппаратным обеспечением, памятью и процессами, оптимизированными для работы на мобильных устройствах;
- библиотеки с открытыми исходными кодами, предназначенными для разработки приложений SQLite, WebKit, OpenGL и медиаменеджер;
- среду исполнения для приложений, включающую виртуальную машину Dalvik и библиотеки ядра, которые отвечают за функционал Android; среда исполнения отличается небольшим размером, что позволяет эффективно использовать ее на мобильных устройствах;
- набор программных компонентов, обеспечивающих доступ к системным службам на уровне приложений; среди них менеджер окон и менеджер местоположения, контент-провайдеры, возможности работы с телефонией и сенсорным дисплеем; • набор компонентов пользовательского интерфейса для размещения и запуска приложений;
- предустановленные приложения, поставляемые в общем программном наборе;
- комплект программ для разработки приложений, включающий инструменты, плагины и справочную документацию. Особо стоит подчеркнуть, что открытая архитектура Android позволяет исправлять любые ошибки в пользовательском интерфейсе или дизайне встроенных

приложений путем написания расширений или замещений ошибок. Android предоставляет возможность создавать собственные интерфейсы для мобильных телефонов, а также приложения с функционалом и дизайном, максимально отвечающими вашим потребностям[1].

1.7 Фреймворк разработчика

Среда разработки приложений для Android включает все необходимое для создания, тестирования и отладки программ. Итак, что входит в состав скачиваемого комплекта:

• API-платформы Android. Ядро среды разработки — библиотеки API, которые обеспечивают программисту доступ к стеку платформы Android. Эти же самые библиотеки используются компанией Google для написания встроенных в Android приложений.

• Инструменты разработки. Вы можете преобразовать исходный код в исполняемые приложения для Android. В состав среды разработки входят инструменты, которые позволяют компилировать и отлаживать приложения.

• Менеджер виртуальных устройств и эмулятор. Эмулятор Android — это полностью интерактивный эмулятор устройств, включающий несколько альтернативных вариантов интерфейса. Эмулятор запускается внутри виртуального устройства Android, которое моделирует аппаратную конфигурацию определенной модели телефона. С по мощью эмулятора вы можете увидеть, как приложения будут выглядеть и функционировать на реальном устройстве под управлением Android. Все программы здесь запускаются внутри виртуальной машины Dalvik, программный эмулятор создает для них идеальное окружение, которое не зависит от особенностей той или иной аппаратной начинки и представляет собой лучшую независимую среду для тестирования, чем любая конкретная аппаратная реализация.

[1]Рето Майер. Android 4. Программирование приложений для планшетных компьютеров и смартфонов, 2013.

• Полный набор документации. Среда разработки включает расширенную справочную информацию с примерами кода, в которой описывается, что входит в каждый программный пакет и класс и как их можно использовать. В дополнение к этому в справочной документации содержится стартовый курс для начинающих, а также подробное описание основ разработки программ для Android.

• Примеры кода. В среде разработки вы найдете примеры программ, которые демонстрируют некоторые возможности Android, а также несколько простых приложений, посвященных определенным функциям API.

• Онлайн-поддержка. Группы Google по адресу http://developer.android.com/resources/community-groups.html — это форумы разработчиков, на которых часто появляются сообщения от команд инженеров и специалистов компании Google. Ресурс StackOverflow[2] также стал популярным — здесь публикуются вопросы, посвященные Android.

Программный стек Android состоит из элементов, показанных на рис. 1.12. Их подробное описание приводится ниже. Упрощенно их можно представить, как комбинацию ядра Linux и набора библиотек C/C++, которые доступны в фреймворке приложения. Последний обеспечивает управление и функционирование рабочей среды и приложений.

[2] Ссылка на ресурс - http://www.stackoverflow.com/questions/tagged/android

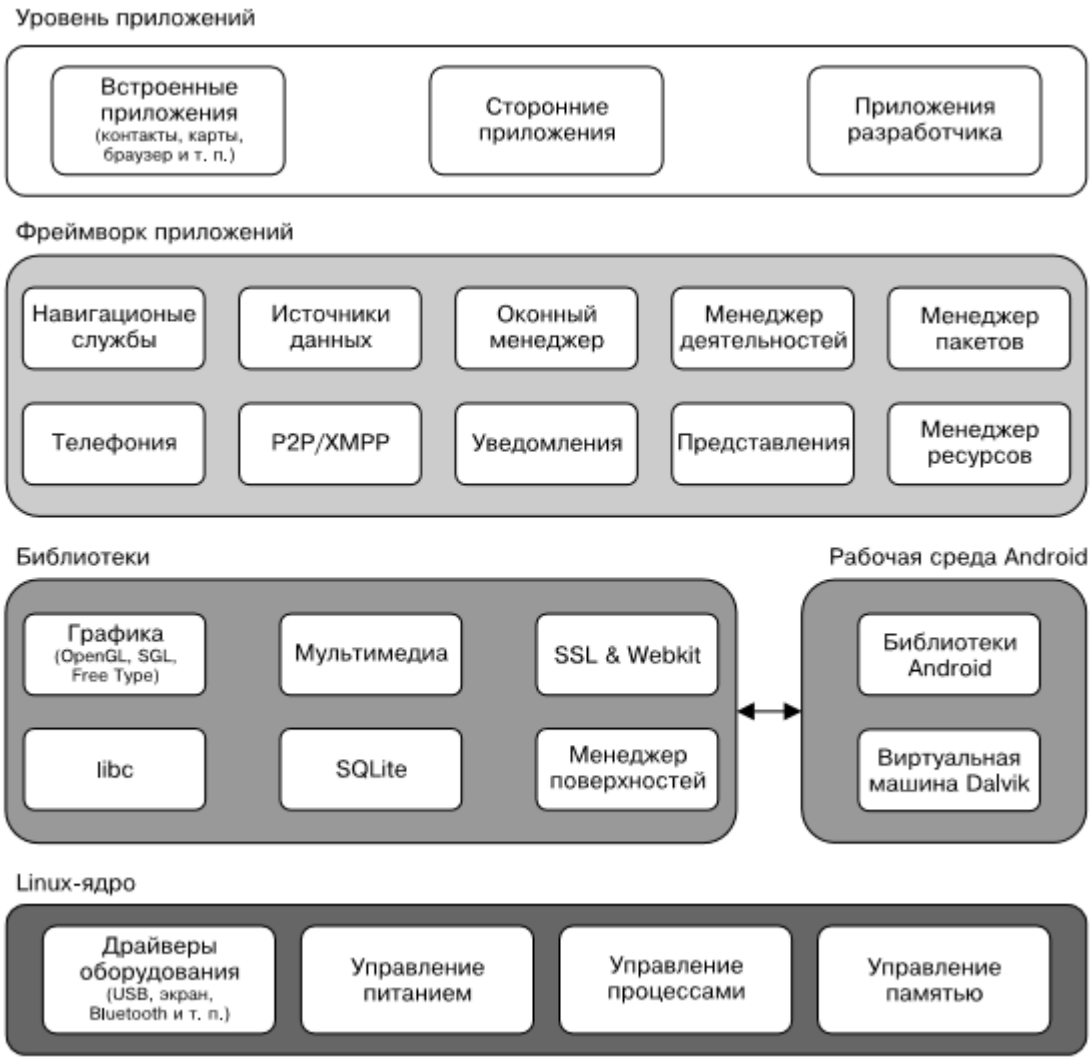


Рисунок 1.12Программный стек Android

• Ядро Linux. Работу системных служб (драйверы устройств, управление процессами и памятью, питанием, безопасность, сетевые службы) обеспечивает ядро Linux версии 2.6. Оно также отвечает за уровень абстракции между аппаратной начинкой и остальной частью программного стека.

Библиотеки. Android включает разнообразные системные библиотеки C/C++ (например, SSL и libc), которые работают поверх ядра. Среди них можно выделить:

- библиотеку для работы с мультимедиа, которая обеспечивает проигрывание аудио- и видеофайлов;
- менеджер интерфейса, отвечающий за управление отображением;

- графические библиотеки, такие как SQL и OpenGL, для работы с 2D- и 3D-графикой;
- библиотеку SQLite, обеспечивающую работу встроенных баз данных;
- SSL и WebKit для работы встроенного веб-браузера и обеспечения интернет-безопасности.

Рабочая среда Android. Особенным телефон на платформе Android делает не столько мобильная версия ОС Linux, сколько рабочая среда Android. Она включает в себя библиотеки ядра и виртуальную машину Dalvik и обеспечивает функционирование программ, а вместе с библиотеками формирует основу фреймворка приложений:

- Библиотеки ядра. Хотя приложения для Android разрабатываются на языке Java, Dalvik — это не виртуальная Java-машина. Библиотеки ядра Android обеспечивают основную функциональность библиотек ядра Java, а также присущийAndroid уникальный функционал;
- Виртуальная машина Dalvik. Dalvik — это виртуальная машина на основе регистров, которая оптимизирована таким образом, чтобы на устройстве можно было запускать несколько приложений одновременно. В ее основе ядро Linux, которое обеспечивает работу потоков и низкоуровневое управление памятью;
- Фреймворк приложений. Фреймворк включает набор классов, которые используются для разработки приложений. Он также предоставляет обобщенные абстрактные классы для доступа к оборудованию и обеспечивает управление пользовательским интерфейсом и ресурсами приложения;
- Уровень приложений. Все программы, как встроенные, так и сторонние, разрабатываются на уровне приложений с использованием одних и тех же библиотек API. Уровень приложений функционирует внутри рабочей среды Android, используя классы и службы, открытые для доступа на этом уровне.

Виртуальная машина Dalvik. Один из ключевых компонентов Android — виртуальная машина (ВМ) Dalvik. Вместо классической виртуальной Java-машины, такой как Java ME (JavaMobileEdition), Android использует собственную ВМ, разработанную для обеспечения эффективной работы нескольких приложений на одном устройстве. В основе ВМ Dalvik ядро Linux, которое обеспечивает работу таких низкоуровневых функций, как безопасность, потоки, управление процессами и памятью. Вы можете также писать приложения C/C++, которые будут работать непосредственно на базовом уровне ОС Linux. Хотя такая возможность и существует, необходимости в этом нет никакой. Если для приложения важны присущие C/C++ скорость и эффективность работы, Android предоставляет доступ к нативной среде разработки (NDK). Она позволяет разрабатывать библиотеки C++ с использованием библиотек libc и libm, а также обеспечивает нативный доступ к OpenGL.

Доступ к устройствам и системным службам Android осуществляется через виртуальную машину Dalvik, которая считается промежуточным слоем. Благодаря использованию ВМ для выполнения кода программы управляющей системы разработчики получают в свое распоряжение уровень абстракции, который позволяет им не беспокоиться об особенностях конструкции того или иного устройства. ВМ Dalvik запускает исполняемые файлы, формат которых оптимизирован под минимальное использование памяти. Вы создаете исполняемый файл с расширением .dex путем трансформирования компилированных классов, написанных на языке Java, используя для этого инструменты, входящие в состав среды разработки. В следующей главе вы узнаете о создании исполняемых файлов формата Dalvik.

Архитектура Android-приложений. В основе архитектуры Android идея многократного использования компонентов, благодаря чему вы можете публиковать и делиться Активностями, Сервисами (Services) и данными с другими приложениями с возможностью управления доступом с помощью

политик безопасности, которые вы назначаете прямо на месте. Аналогичный механизм, с помощью которого вы можете создать собственный менеджер контактов или программу для набора номера, позволяет предоставлять другим программистам свои компоненты — на их основе они смогут создавать новые пользовательские интерфейсы или функциональные расширения, а также строить свои приложения.

Перечислим службы приложений, которые можно назвать базовыми составляющими архитектуры всех приложений для Android, а также основой фреймворка, который вы будете использовать в собственных программах.

- Менеджер Активностей. Контроль за жизненным циклом Активностей (Activity), включая управление стеком Активностей
- Представления. Используются при создании пользовательских интерфейсов для ваших Активностей (глава 4).
- Менеджер уведомлений. Обеспечивает работу унифицированных ненавязчивых уведомлений для пользователей.
- Источники данных. Позволяют приложениям открывать доступ к данным.
- Менеджер ресурсов. Обеспечивает отображение некодированных ресурсов, таких как текстовые строки или изображения.

Android предоставляет набор API-библиотек для разработки приложений. Мы не будем приводить их полный список. По адресу: http://developer.android.com/reference/packages.html, можно найти полный список пакетов, включенных в состав среды разработки Android. Android задуман для работы на множестве разнообразных мобильных устройств, поэтому примите во внимание, что совместимость и функционирование некоторых продвинутых или дополнительных API-функций может значительно варьироваться на том или ином устройстве.

2 КОНЦЕПЦИЯ ПРОЕКТА

Модели SADT дают полное, определённое, верное и адекватное описание системы, которая имееткакое-либо конкретное назначение. Под этим назначением понимают то, что называется целью модели, которое в свою очередь вытекает из определения модели в SADT. В моделях SADT применяются как естественный, так и графический языки. Чтобы передать информацию о конкретной системе источником естественного языка служат люди, которые описывают систему, а источником графического языка – является методология SADT. Для естественного языка модели графический язык SADT создает структуру и точную семантику. Графический язык SADT организует естественный язык вполне определённым и однозначным образом.

Рассматриваяс точки зрения SADT, модель может описывать либо функции системы, либо объекты этой системы. SADT-модели, которые ориентированы на функции, положено называть функциональными моделями, а модели, которые ориентированы на объекты системы - моделями данных, функциональная модель показывает с необходимой степенью детализации систему функций, которые, в свою очередь, отображают свои взаимоотношения через объекты системы. Модели данных дуальны к функциональным моделям и представляют собой описание объектов системы, которые связаны системными функциями. Полная методология SADT поддерживает создание множества моделей для более точного и детализированного описания сложной системы[3].

Методология IDEF0 основана на следующих концептуальных положениях:

- Модель – это искусственный объект, которые представляет из себя отображениеили образ компонентов системы и системы в целом. Модель разрабатывается для понимания, анализа и принятия решений для

[3] Марка Дэвид А., МакГоуэнКлемент. Методология структурного анализа и проектирования SADT, 1999

проектировании новой системы, либо о реконструкции уже существующей, или её замене. Система представляет из себя совокупность связанных и взаимодействующих частей между собой, которые выполняют какую-либо полезную работу. Частями или элементами системы могут быть любые комбинации различных сущностей, которые могут включать людей, информацию, программное обеспечение, оборудование, изделия, сырье или энергию. Модель показывает, что происходит в системе, как ею управляют, какие сущности она преобразует, какие средства используютсяпри выполнения своих функций и что производит.

- Блочное моделирование и его графическое представление. Основной концептуальный принцип методологии IDEF – представление любой системы, которую изучают, в виде набора взаимодействующих и взаимосвязанных блоков, где могут быть отображены процессы, операции, действия, существующие в изучаемой системе. Под функциями в IDEF0 следует понимать все, что происходит в системе и ее элементах. Для каждой функции ставится блок. IDEF0–диаграмма - основной документ при анализе и проектировании систем. НаIDEF0–диаграмме блок представляет собой прямоугольник. Интерфейсы, с помощью которых блок взаимодействует с другими блоками или с внешней по отношению к моделируемой системе средой, представляются стрелками, входящими в блок или выходящими из него. Входящие стрелки указывают, что за условия должны быть одновременно сделаны, чтобы функция, которую описывает блок, была выполнена.

- Лаконичность и точность. Многословные характеристики, изложенные в форме традиционных текстов, неудовлетворительны. Графический язык дает возможность лаконично, верно и точно показать все блоки системы, отношения и связи между ними, показать ошибочные лишние.

- Передача информации. Средства IDEF0 весомооблегчают передачу информации от одного участника разработки модели к

другому.Участником разработки может быть, как отдельный разработчика, так и рабочая группа. К этим средствам относятся: диаграммы, которые основаны на простой графике блоков и стрелок, легкочитаемые и понимаемые; метки на естественном языке, чтобы с их помощью описывать блоки и стрелки, а такжеглоссарий и сопроводительный текст для уточнения смысла элементов диаграммы;последовательная декомпозиция диаграмм, которая строится по иерархическому принципу, при котором на верхнем уровне показываются основные функции, а затем происходит их детализация и уточнение;древовидные схемы иерархии диаграмм и блоков, обеспечивающиеобозримость модели в целом и входящих в нее деталей.

- Строгость и формализм. Чтобы обеспечить преимущества методологии в отношении однозначности, точности и целостности сложных многоуровневых моделей, при разработке моделей IDEF0 необходимособлюдать ряд строгих формальных правил.Отметим только основные из них: все стадии и этапы разработки и корректировки модели должны строго, формально документироваться с тем,чтобы при ее эксплуатации не возникало вопросов, связанных с неполнотой или некорректностью документации.
- Итеративное моделирование. Разработка модели в IDEF0 представляетсобой пошаговую, итеративную процедуру. На каждом новом шаге итерацииразработчик предлагает вариант модели, который подвергают обсуждению, рецензированию и последующему редактированию, а затем циклповторяется. Именно такая организация работы способствует оптимальному использованию знаний системного аналитика, который владеет методологией итехникой IDEF0, и знаний специалистов – экспертов в предметной области, к которой относится объект моделирования.
- Отделение «организации» от «функций». При разработке моделейнеобходимо избегать начальной привязки функций исследуемой системык существующей организационной структуре моделируемого объекта. Это помогает избавиться от субъективной точки зрения, которую

навязывает организация и её руководство. Организационнаяструктура должна быть результатом применения модели. Сравнивая результат, с существующей структурой, появляется возможность, не только оценить адекватность модели, но и предложить решения, которые могут быть направлены на совершенствование этой структуры[4].

2.1Разработка SADT-модели(IDEF0) «как-есть»

На данный момент в различных предприятиях существует множество способов информирования сотрудников, однако большинство из них неудобны. Среди них: оповещение по телефону, рассылка смс, социальные сети, электронная почта. Каждый из этих способов имеет свои положительные и отрицательные моменты, однако перед нами ставится задача создать универсальное средство, которые было бы максимально удобно и эффективно. Так как этих способов достаточно много была создана общая SADT-модель (IDEF0) «как-есть» (рис. 2.1).

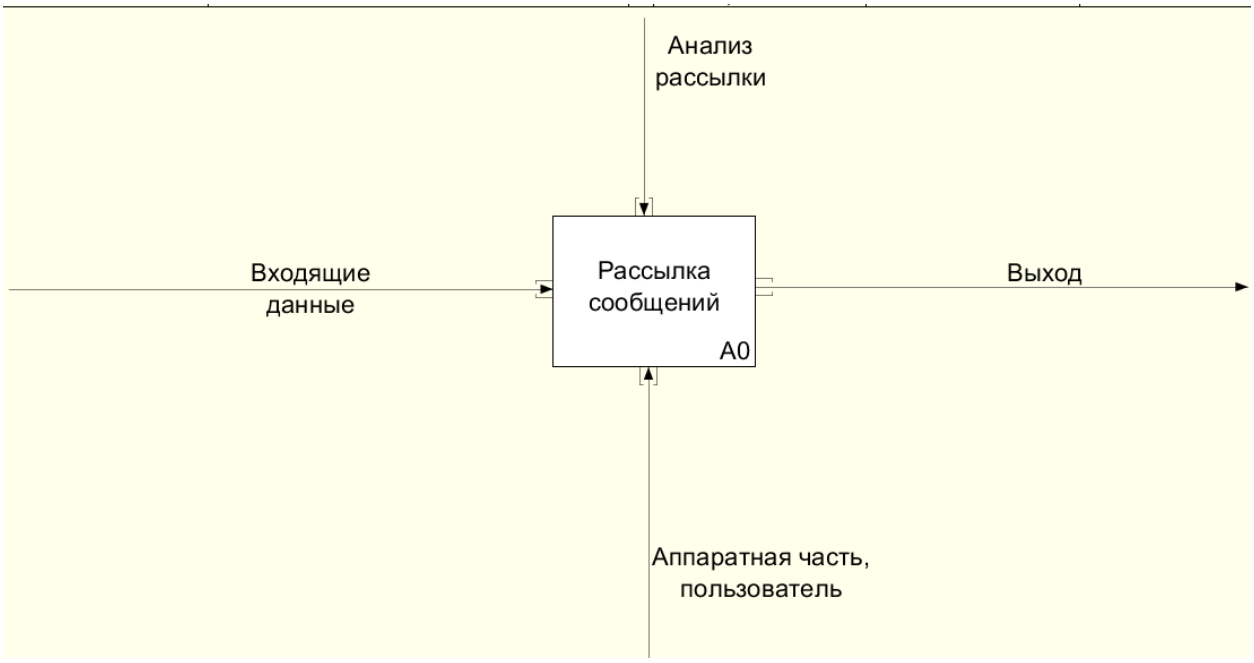


Рисунок 2.1 Родительская диаграмма SADT-модели(IDEF0) «как-есть»

[4]А. Ф. Похилько, И. В. Горбачев, С. В. Рябов, Моделирование процессов и данных с использованием CASE-технологий, 2014

На дочерней диаграмме (рис. 2.2) видно, что модель довольно проста, но из этого следуют и минусы. Например, неверный выбор адресата или в случае использование электронной почты нет полной гарантии, что письмо дойдет или не попадет в папку со спамом.

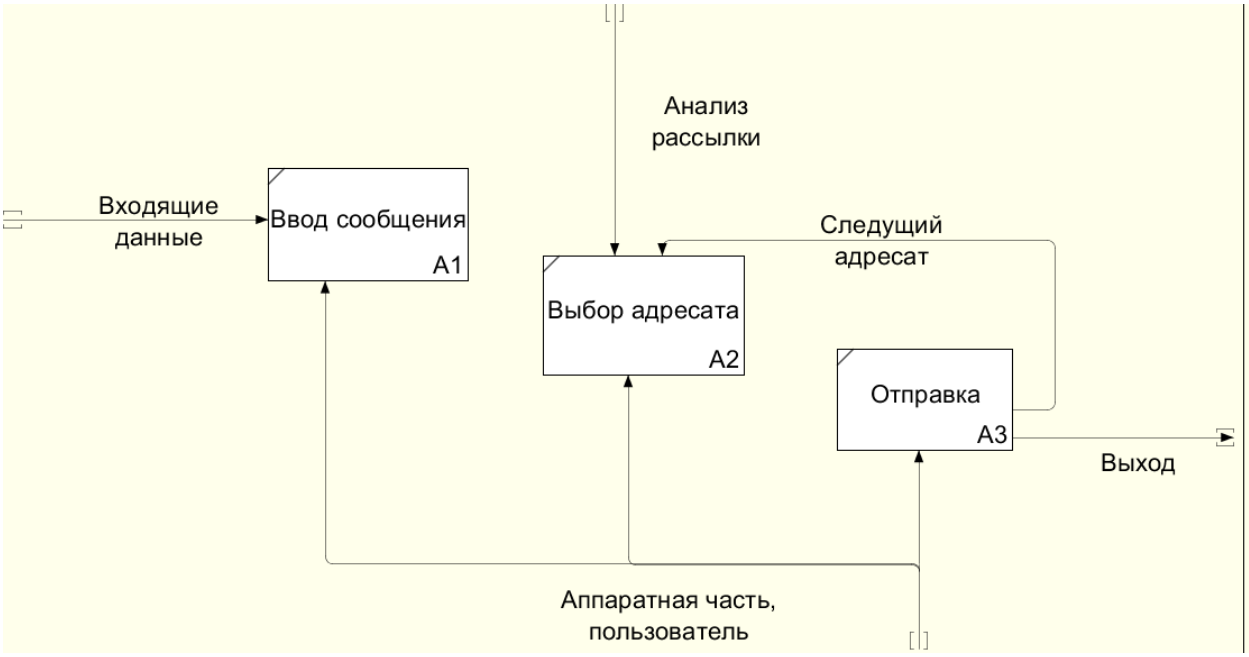


Рисунок 2.2 Дочерняя диаграмма SADT-модели(IDEF0) «как-есть»

2.2Разработка SADT-модели(IDEF0) «как-должно-быть».

На данной функциональной модели (рис 2.3, 2.4) наглядно показано, как осуществляется основная функция отправка сообщения. Пользователь работает с приложением: входит под своей учетной записью, либо регистрируется при необходимости, у него есть возможность создать группы, пригласить в нее нужных пользователей, отправить сообщение.

На родительской диаграмме (рис 2.3) бизнес-функцией является оправление сообщения, управление осуществляется с помощью анализа доступных возможностей, а механизмом в данном случае является пользователь и аппаратная часть. На входе и выходе — запуск приложения и завершение работы, соответственно.

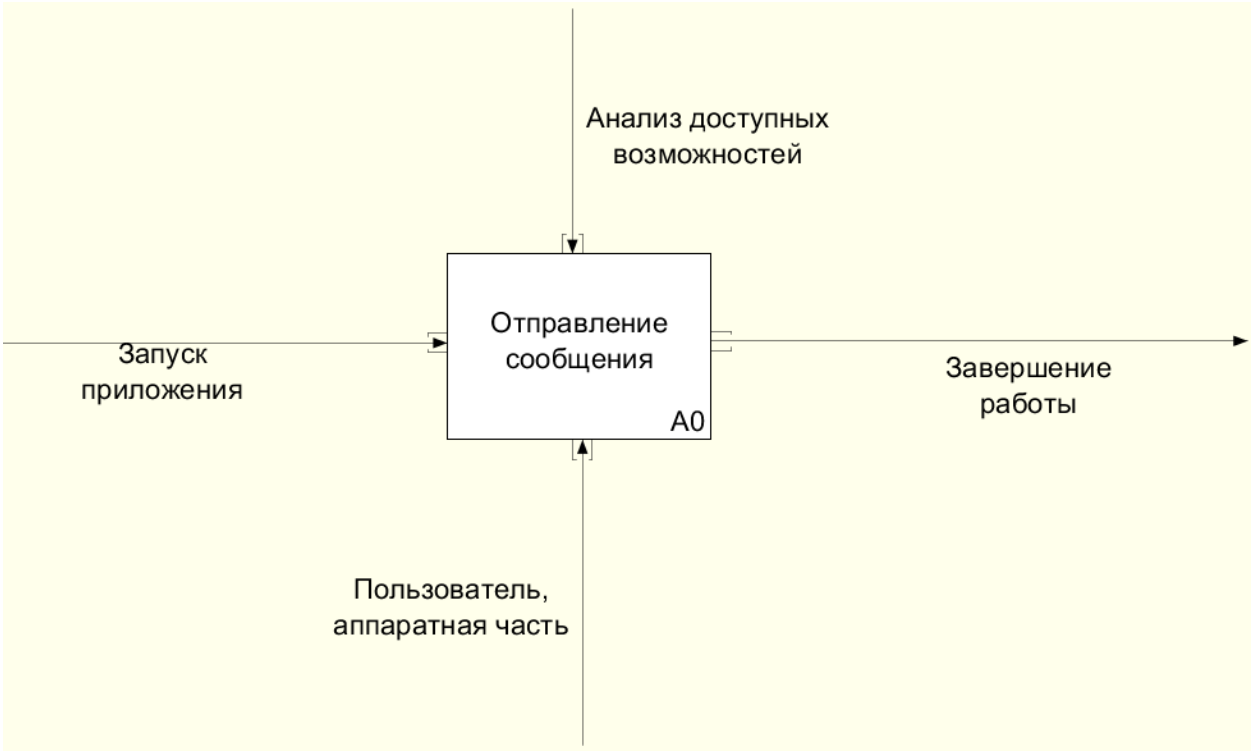


Рисунок 2.3 Родительская диаграмма SADT-модели(IDEF0) «как-должно быть»

На дочерней диаграмме (рис 2.4) указаны все функции приложения, а так же возможные действия в проектируемом приложение.

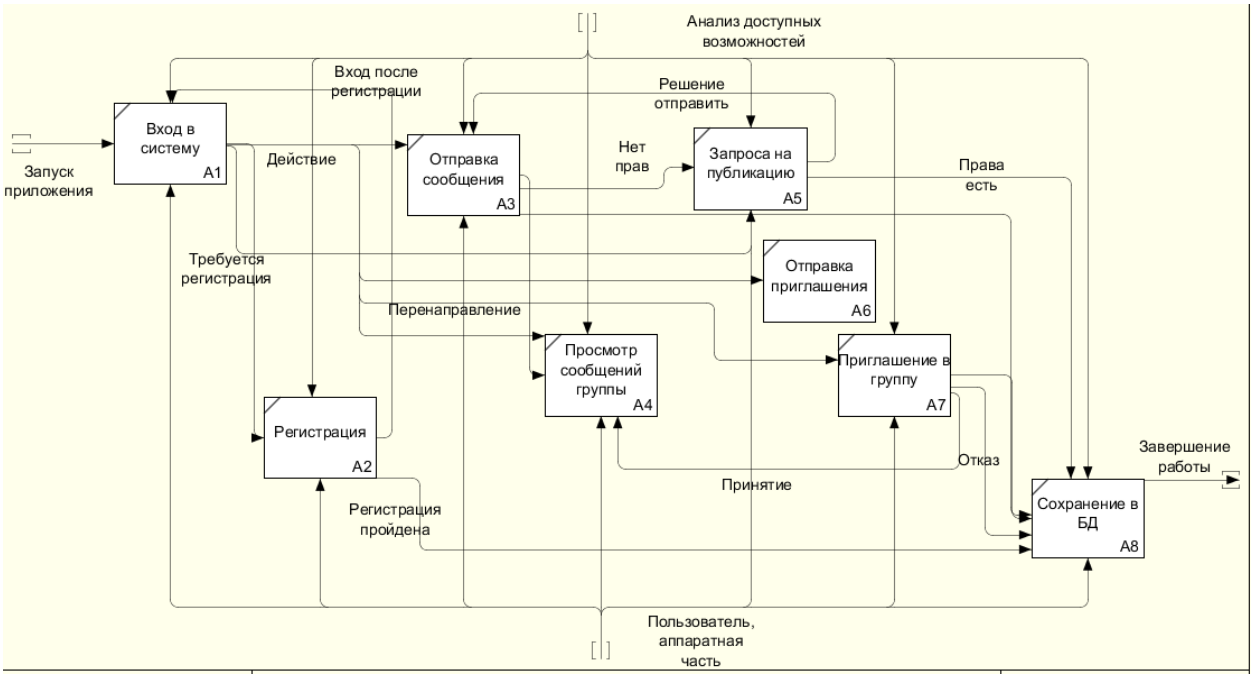


Рисунок 2.4Дочерняя диаграмма SADT-модели(IDEF0) «как-должно быть»

2.3Разработка ER-диаграммы (схема базы данных)

Схема базы данных представлена на рисунке 2.5. Для хранения данных в проектируемом приложении были созданы следующие таблицы: пользователи, общая информация, сообщения, группы, пользователи в группе N и пользователи в группе N. Она Отображает таблицы, хранящие данные проектируемого приложения. База данных приведена к 3ей нормальной форме.

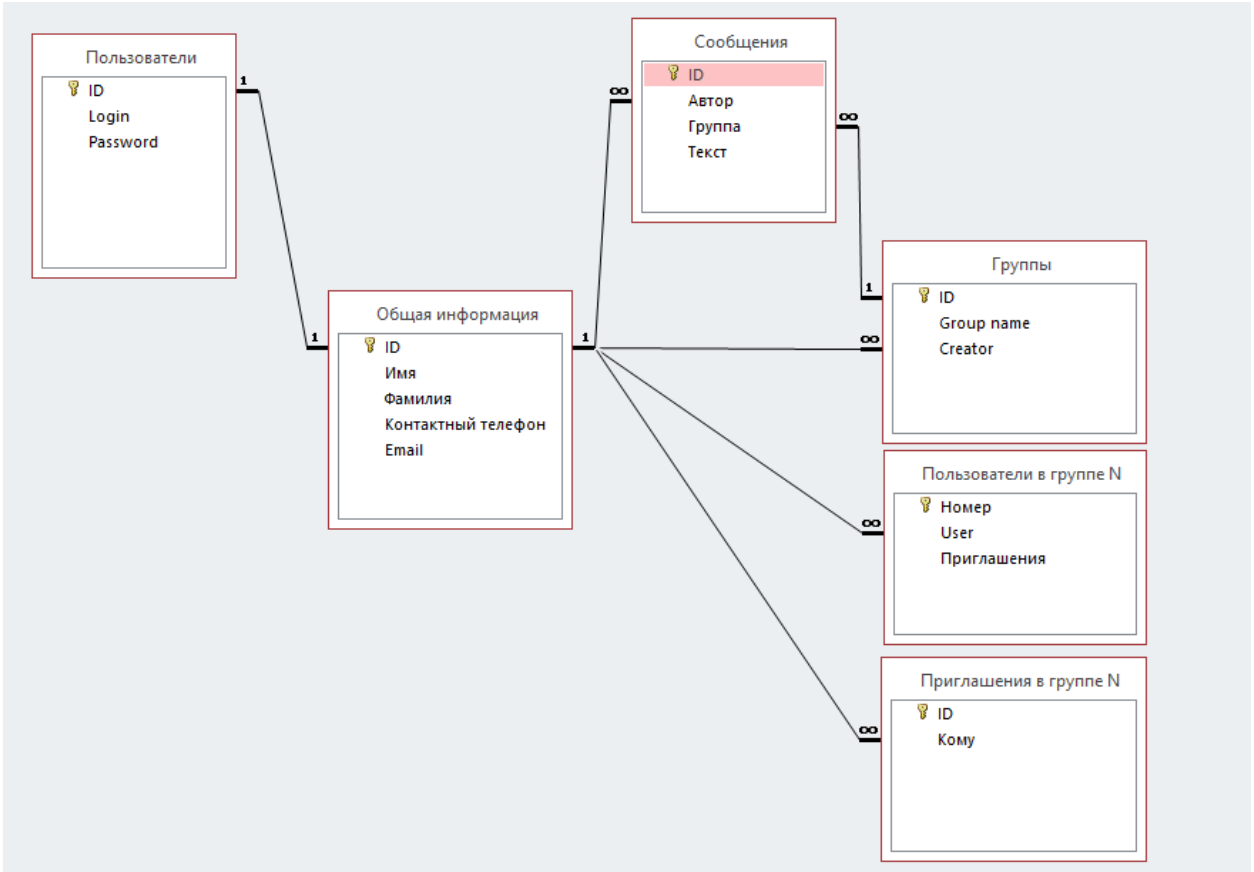


Рисунок 2.5 Схема базы данных

2.5 Разработка форм проекта

Были спроектированы основные формы и интерфейс приложения.Рисунки с 2.6 по 2.9.

Следует отметить, что все формы спроектированы без излишеств. Минимальное количество полей и кнопок, отсутствует перегрузка экрана.

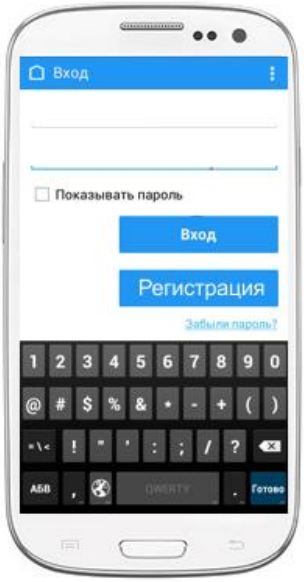


Рисунок 2.6 Форма входа

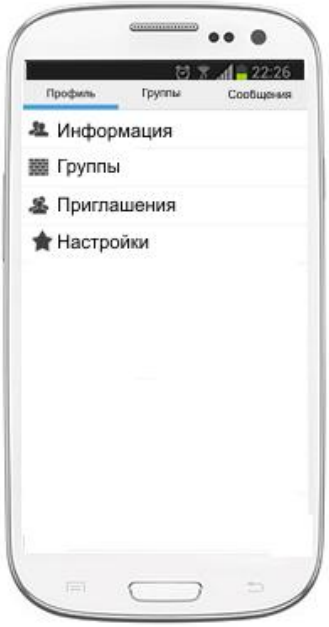


Рисунок 2.7 Форма профиля

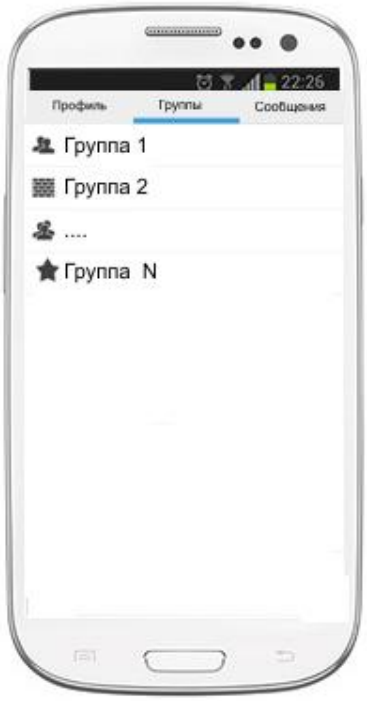


Рисунок 2.8 Форма просмотра групп

Рисунок 2.9 Форма отправки сообщения

3 СИСТЕМНАЯ АРХИТЕКТУРА ПРОЕКТА

3.1 Диаграмма вариантов использования

Диаграмма вариантов использования содержит конечное множество вариантов использования, которые в целом должны определять все возможные стороны ожидаемого поведения системы. Для удобства множество вариантов использования может рассматриваться как отдельный пакет. Применение вариантов использования на всех этапах работы над проектом позволяет не только достичь требуемого уровня унификации обозначений для представления функциональности подсистем и системы в целом, но и является мощным средством последовательного уточнения требований к проектируемой системе на основе их итеративного обсуждения со всеми заинтересованными специалистами.

На диаграмме вариантов использования (рис 3.1) изображены отношения между актерами и вариантами использования.

Актером на данной диаграмме является пользователь приложением на рисунке изображен в виде "человечка", варианты использования изображены в виде эллипсов (среди них – регистрация, запрос на публикацию сообщения, публикация сообщения, приглашение в группе, удаление сообщения и др). На диаграмме указаны все возможные варианты использования приложения пользователем. Отношения между пользователем и вариантами использования показаны ассоциациями (стрелки на рисунке).

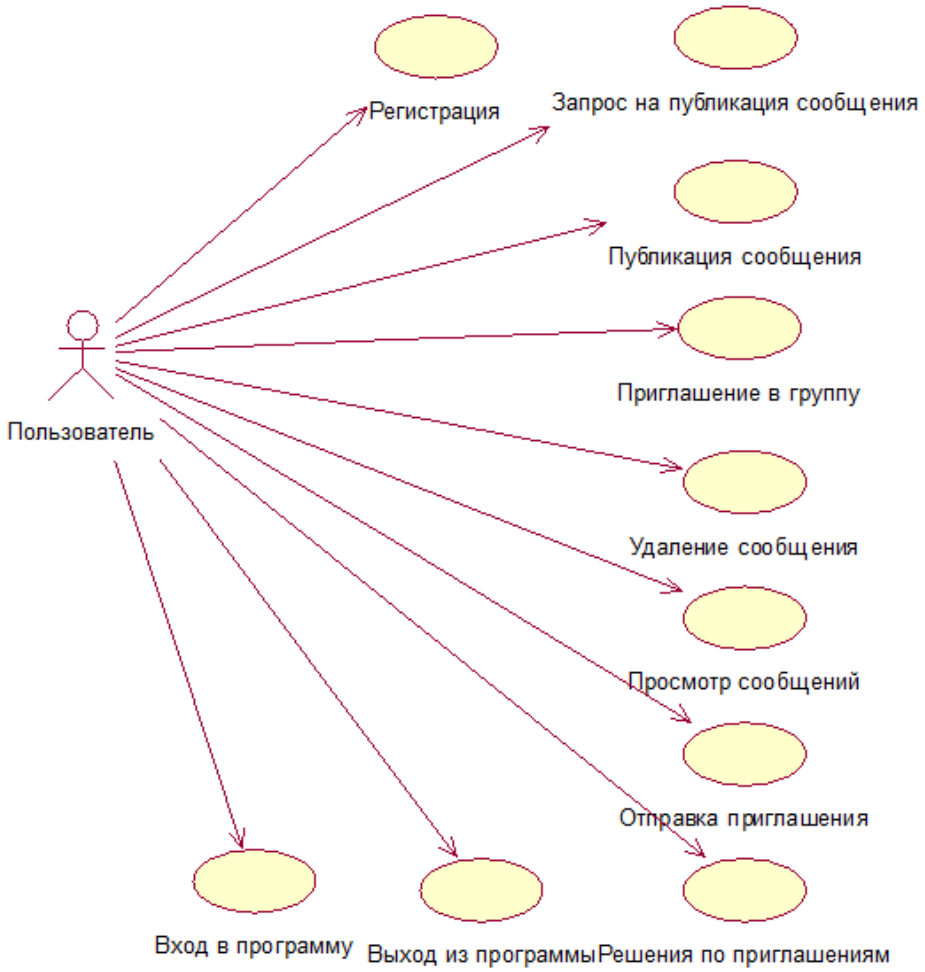


Рисунок 3.1 Диаграмма вариантов использования

3.2 Диаграмма классов

Диаграмма классов —диаграмма, на которой представлена совокупность декларативных или статических элементов модели, таких как классы с атрибутами и операциями, а также связывающие их отношения.

На диаграмме классов (рис. 3.2) изображены следующие классы: Процесс обработки – управляющий класс, который координирует все остальные классы, имеет операцию – взаимодействие, а также атрибут – Алгоритм, т.е способ обработки;

Интерфейс, имеет операцию – взаимодействие путем вывода информации (предоставление информации пользователю системы).

БД – класс-сущность, информация о которых должна храниться постоянно и не уничтожаться с выключением системы. БД имеет атрибуты Пользователи, общая информация, группы, сообщения, приглашения в группе N и пользователи в группе N, которые являются таблицами базы данных, операцией является хранение данных.

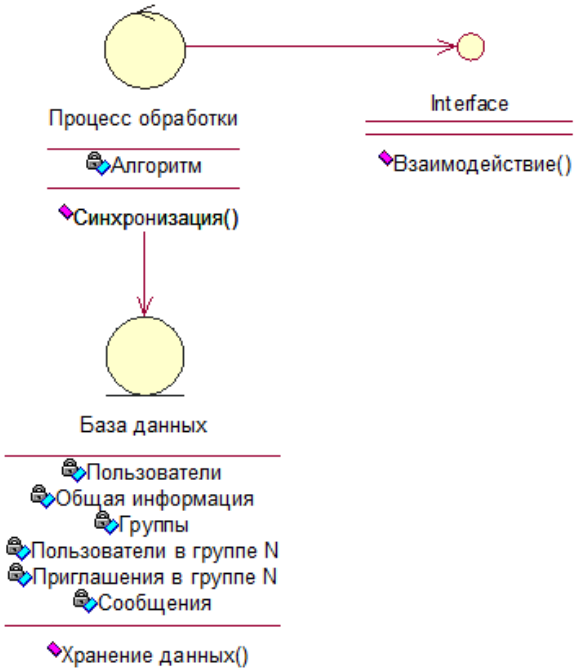


Рисунок 3.2 Диаграмма классов

3.3 Диаграмма деятельности

Для удобства было сделано две диаграммы деятельности (для регистрации и для использования функций приложения).

Диаграмм деятельности визуализирует особенности реализации операций классов, когда необходимо представить алгоритмы их выполнения. Диаграмма деятельности (рис 3.3) выполнена с помощью дорожек для наглядного представления в каком состояние находится система на разных этапах и какое подразделение за него отвечает.

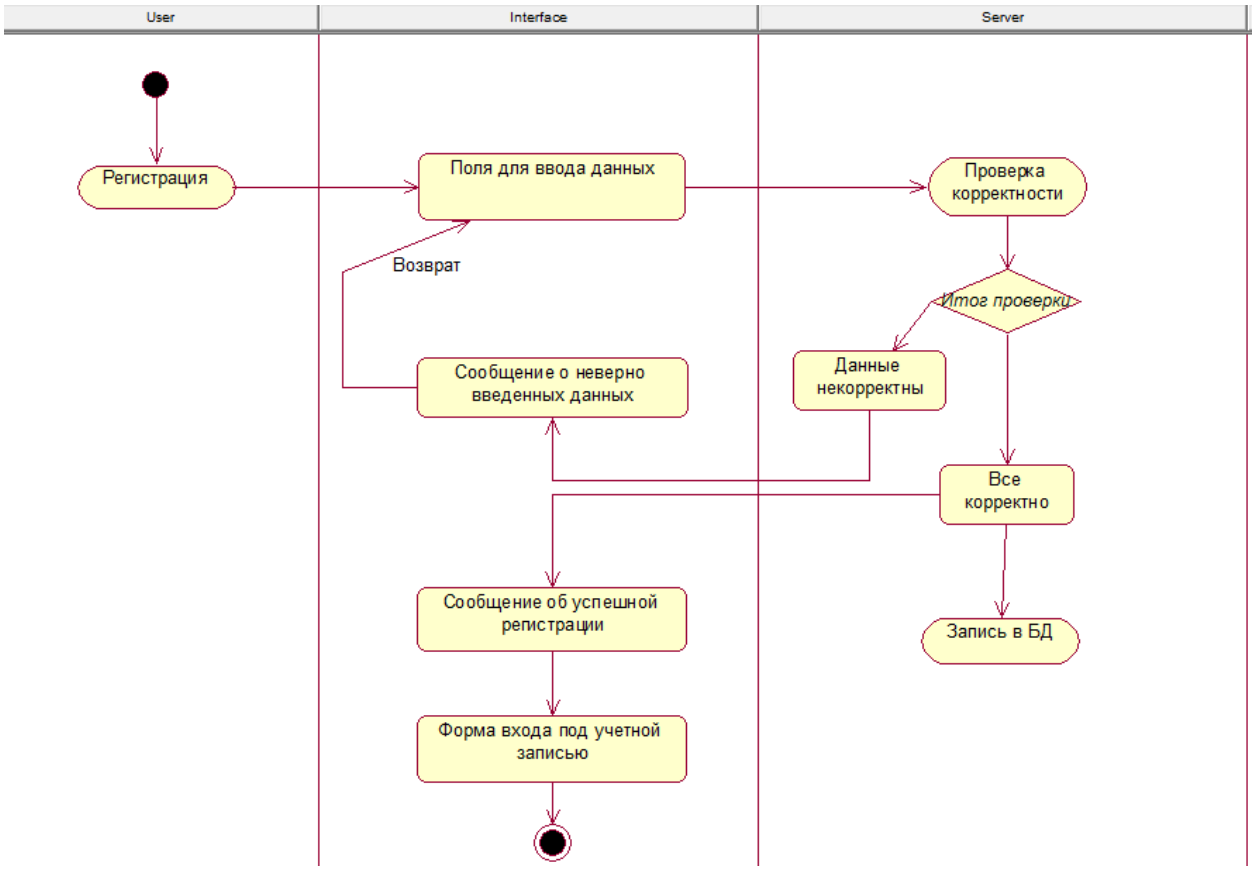


Рисунок 3.3 Диаграмма деятельности (Регистрация)

Пользователю при регистрации предлагается ввести свои данные, программа проверяет их корректность (проверка обязательных полей, запрет специальных символов) и доступность (доступность логина, регистрации на вводимую почту), если данные введены корректно, то данные записываются в БД и пользователь видит сообщение об успешной регистрации, с последующей возможностью войти под своей учетной записью, в противном случае, его возвращает на форму регистрации с сообщением об ошибке в введенных данных.

На данной диаграмме (рис. 3.4) указаны возможные варианты взаимодействия пользователя с приложением, благодаря диаграмме деятельности наглядно продемонстрированы состояния, в которых находится система.

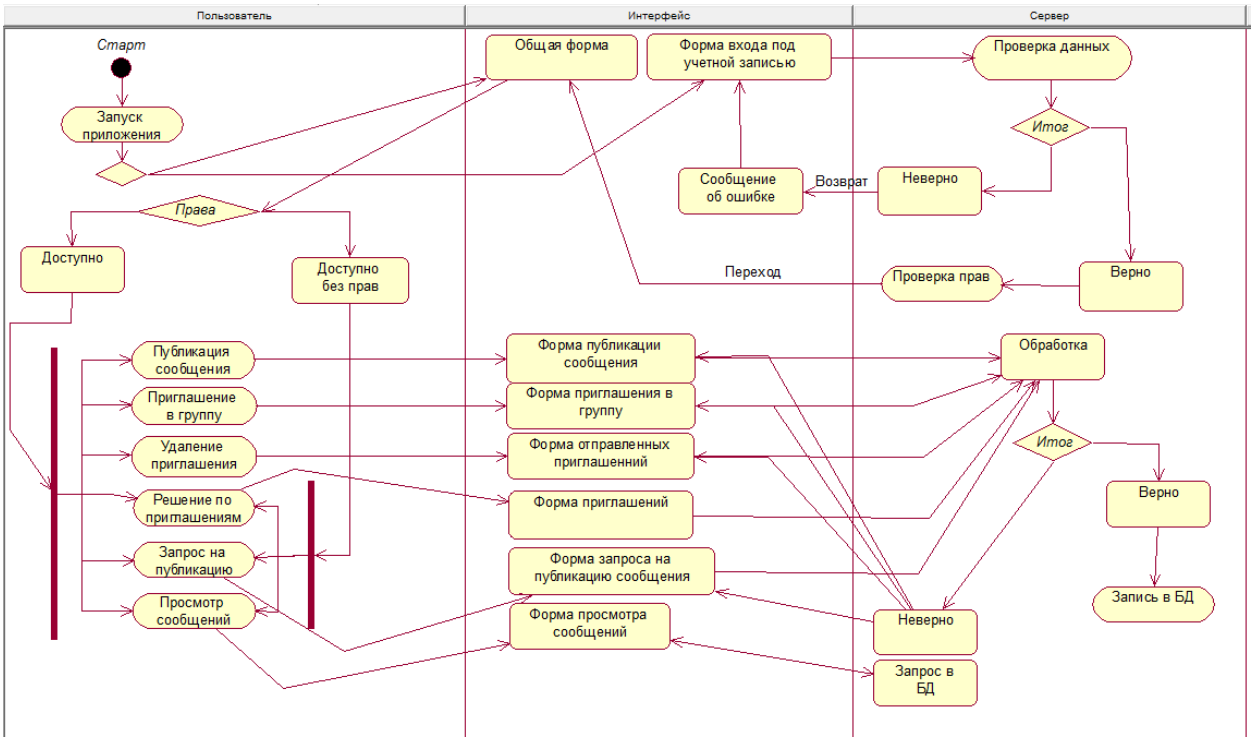


Рисунок 3.4 Диаграмма деятельности (работа с приложением)

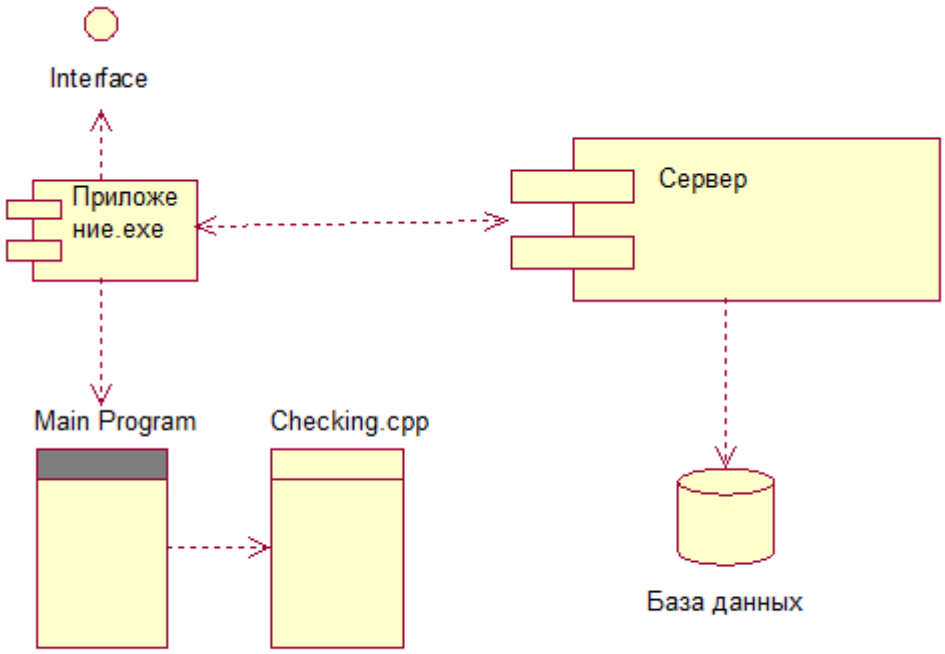
Рассмотрим один из вариантов использования приложения пользователем. Предположим, что ему требуется проверить наличие новых сообщений в группе. Таким образом пользователь запускает приложение, далее он входит под своей учетной записью, если не сделал этого до этого, далее на общей на общей форме (дорожка интерфейс) он видит список доступных ему действий, которые предварительно были проверены (дорожка сервер), пользователь выбирает просмотр сообщений группы и попадает на форму с её сообщениями.

3.4 Диаграмма компонентов

Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными

компонентами, в роли которых может выступать исходный, бинарный и исполняемый код.

На данной диаграмме (рис 3.5) показано, что приложение включает в себя модуль главной программы (MainProgram), которая реализуется с помощью модуля подпрограммы проверки (Checking.cpp), и связываются с сервером, который в свою очередь взаимодействует сБазой данных,



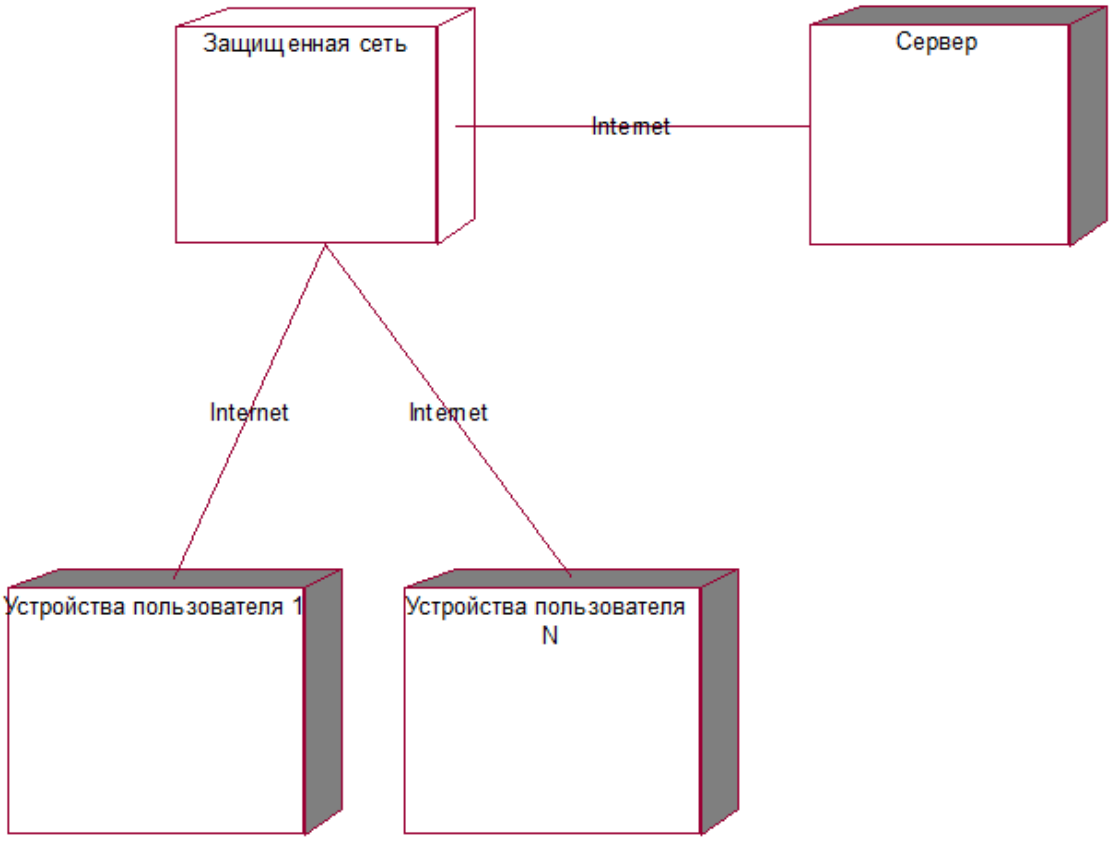
информация при этом будет показывается через интерфейс.

Рисунок 3.5 Диаграмма компонентов

3.5 Диаграмма развертывания

Диаграмма развертывания (рис. 3.6) показывает на какой платформе и каких вычислительных средствах реализована система, что дает полное физическое представление о проектируемой системе.

На данной диаграмме показано, что устройства пользователей посредством сети интернет входят в защищенную сеть и взаимодействует с сервером. Сервер аналогично взаимодействует с устройствами



пользователей.

Рисунок 3.6 Диаграмма развертывания

4 ОЦЕНКА ТРУДОЗАТРАТ

Оценка трудозатрат производилась в человеко-днях из-за нестабильности курса и зарплат.

Таблица 1. - Расчет трудозатрат

Вид работы	Норма времени		Формула расчета	Трудоемкость Чел.-дн
	№ таблиц	Значение		
1	2	3	4	5
Подготовка процесса				
Подготовка процесса	5.27	9	Тпп = Ксл · Кан · Куч · Нвр.пп	9
			Итого на работу	9
Анализ проблем и изменений				
Анализ сообщения о проблеме или заявки на внесение изменений	5.28	4,8	Тан = Ксл ·Кхв · Куч ·Нвр.ан	4,8
Верификациявозникшей проблемы	5.29	9,6	Твер = Ксл · Кте · Куч · Нвр.вер	16,32
Разработка вариантовреализации изменений	5.30	5,2	Твар = Кхв · Куч ·Ксл· Нвр.вар	5,2
Документальное оформление сообщения о проблеме или заявки на внесение изменения; результатов их анализа и вариантов реализации изменений и получение согласования выбранного варианта реализации изменения в соответствии с договором			Тсогл = 5 чел.-дн.	5
			Итого на работу	31,32
Внесение изменений				
Анализ и определение перечней программ и документов, требующих изменения; оформление результата	5.31	4,2	Тдр = Куч ·Ксл· Нвр.раз	4,2
Реализация процесса разработки для внесения изменений	5.32	17	Траз = Краз · Куч ·Ксл· Нвр.раз	17
			Итого на работу	21,2

Проверка и приёмка при сопровождении				
Проверка внесенного изменения в целях подтверждения работоспособности измененного ПС	5.33	3,5	Тпи = Кте · Кхв · Ксл · Нвр.пи	5,95
Перенос				
Получение подтверждения правильности внесенного изменения от организации-заказчика			Тпод = 5 чел.-дн.	5
Проверка соответствия, переносимого ПС стандарту ИСО/МЭК 12207-99» и «Разработка плана переноса	5.34	10,4	Тпп = Кхп · Нвр.п	10,4
Уведомление пользователей о планах и работах по переносу			Туп=1,0 чел.-дн.	1
Обучение специалистов пользователя работе в новой среде	5.35	15,8	Тоб = Ксл · Нвр.об	15,8
Архивация прежних программ и документации	5.36	8,4	Тар = Нвр.ар	8,4
Анализ влияния перехода к новой среде	5.37	5,8	Тап = Ксл · Нвр.ап	5,8
			Итого на работу	52,35
Снятие с эксплуатации				
Разработка и оформление плана снятия с эксплуатации	5.38	10,4	Тпсэ = Кхп · Нвр.псэ	10,4
Уведомление пользователя о планах и работах по снятию с эксплуатации			Туп = 1,0 чел.-дн.	1
Обучение пользователей в течение периода параллельной эксплуатации прежнего и нового программных средств	5.39	18,5	Тобн = Ксл · Нвр.обн	18,5
Архивация связанной с прежним объектом	5.26	8,4	Тар = Нвр.ар	8,4

документации разработки, журналов регистрации и программ				
			Итого на работу	38,3
			ИТОГО	152,17

Таким образом проектирование и разработка данного приложения займет 152 человека-дня.

ЗАКЛЮЧЕНИЕ

Целью данной выпускной квалификационной работы является проектирование приложения-мессенджера для рассылки информационных сообщений в предприятий с иерархичной структурой

В ходе проведения исследования были рассмотрены теоретические аспекты организации управления в предприятиях с иерархичной организацией, а также разработана платформа на принципе иерархии, с помощью которой нет необходимости зацикливаться на отдельных видах организационных структур.

В работе был проведен анализ приложений-мессенджеров и текущих средств отправки сообщений. Полностью проанализировав коммуникативность в структурах с иерархичной структурой автору проекта, помимо главных задач таких рассылка и просмотр информационных сообщений,удалось спроектировать следующие задачи:

- аутентификацию пользователей,
- создание групп пользователей,
- возможность приглашения пользователей в группы,
- возможность запроса на публикацию сообщения в группе,если у пользователя недостаточно прав.

Была спроектирована система регистрации, чтобы она была эффективной, были определены следующие столбы, которым нужно уделить внимание: человечность, потраченное время, подписи, корректность.Безопасность, пароль,соцсети.

При выборе среды разработки были проанализированы для сравнения две IDE: Androidstudio и Eclipse. Сравнение было произведено по следующим критериям: отладка, автозаполнение, рефакторинг. В ходе сравнения было выявлено, что Androidstudio объективно выиграла у Eclipse,

но стоит заметить, что сравнивали их именно какJavaIDE для разработки под Android.

Проектирование приложения-мессенджера выполнено полностью в соответствии с заданием. В работе использовались методы системного анализа, структурного, объектного и информационного моделирования, математической статистики, математического моделирования, идентификации и прогнозирования.

В итоге, основным выводом данной ВКР следует считать тот факт, что для повышения эффективности работы предприятия и подразделений на нем, прежде всего, необходимо разработать эффективную систему для информирования персонала. В результате работы был получен готовый проект приложения, полностью готовый к разработки и реализации. Автор полагает, что полностью справился с поставленной ему задачей.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. С.В. Маклаков. Создание информационных систем с ALL FusionModellingSuite. М.,2003.

2. С.В. Маклаков. ERwin и Bpwin. CASE-средства разработки информационных систем.М.,1999.

3. Боггс У., Боггс М. UML и RationalRose: Пер. с англ. –М.: Лори, 2000.

4. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++. 2-е изд.: Пер. с англ. – М.: Издательство Бином, СПб.: Невский диалект, 1999.

5. Буч Г., Рамбо Дж., Джекобсон А. Язык UML. Руководство пользователя: Пер. с англ. – М.: ДМК, 2000.

6. Вендров А. М. CASE-технологии. Современные методы и средства проектирования информационных систем. – М.: Финансы и статистика, 1998.

7. Вендров А. М. Проектирование программного обеспечения экономических информационных систем. – М.: Финансы и статистика, 2000.

8. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования: Пер. с англ. – М.: ДМК, 2000.

9. Коноков Д.Г., Рожков К.Л., Организационная структура предприятий, 1999.

10. Марка Дэвид А., МакГоуэнКлемент. Методология структурного анализа и проектирования SADT, 1999.

11. Похилько А. Ф., Горбачев И. В., Рябов С. В., Моделирование процессов и данных с использованием CASE-технологий, 2014

12. Рето Майер. Android 4. Программирование приложений для планшетных компьютеров и смартфонов. Пер. с англ – СПб: Эксмо, 2013.

13. Якобсон А., Буч Г., Рамбо Дж. Унифицированный процесс разработки программного обеспечения. Пер. с англ. – СПб: Питер, 2002.

14. Интерфейсы будущего: изучаем materialdesign от google на практике URL - https://xakep.ru/2015/12/28/material-design/ (дата обращения: 15.03.2015).

15. Лучший интерфейс — отсутствие интерфейса URL - https://habrahabr.ru/post/156473/ (дата обращения: 30.03.2015).

16. Пишем приложения с поддержкой плагинов для android. Часть 1. URL - https://xakep.ru/2016/06/02/android-plugins-binder/ (дата обращения: 5.03.2015).

17. Почему IDEA лучше EclipseURL - https://habrahabr.ru/post/112749/ (дата обращения: 27.03.2015).

18. Почему ваш любимый мессенджер должен умереть URL - https://habrahabr.ru/post/272937/ (дата обращения: 16.04.2015).

19. Национальный открытый университет URL [электронный ресурс] - http://www.intuit.ru/ (дата обращения: 28.04.2015).