РОСЖЕЛДОР

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Ростовский государственный университет путей сообщения»

(ФГБОУ ВО РГУПС)

Н.А. Москат

ОПЕРАЦИОННЫЕ СИСТЕМЫ И КРОССПЛАТФОРМЕННОЕ ПРОГРАМИРОВАНИЕ

Учебно-методическое пособие
для курсовой работы

Ростов-на-Дону
2017

УДК 681.3.06(07) + 06

Рецензент – доктор технических наук, профессор М.А. Бутакова

Москат, Н.А.

Операционные системы и кроссплатформенное программирование: учебно-методическое пособие для курсовой работы / Н.А. Москат; ФГБОУ ВО РГУПС. – Ростов н/Д, 2017. – 23 с.

Рассмотрены основные требования к оформлению исследований студента в рамках курсового проектирования. Содержится обзорный материал, приведены задания и методика выполнения курсовой работы.

Предназначено для студентов направлений «Информатика и вычислительная техника» и «Информационные системы и технологии», изучающих дисциплины «Операционные системы», «Операционные системы и кроссплатформенное программирование», «Операционные системы реального времени», «Операционная система LINUX», «Инструментальные средства информационных систем (Операционные системы)», а также для студентов всех специальностей, изучающих указанные и смежные дисциплины. Изложенный материал будет полезен при курсовом и дипломном проектировании, а также аспирантам и соискателям.

Одобрено к изданию кафедрой «Вычислительная техника и автоматизированные системы управления».

ОГЛАВЛЕНИЕ

1 ВАРИАНТЫ ПОСТРОЕНИЯ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ. НАТИВНЫЙ ПОДХОД. КРОССПЛАТФОРМЕННЫЙ ПОДХОД

В настоящее время перед разработчиками программных приложений зачастую стоит вопрос выбора способа реализации приложения. Разрабатывать ли мобильное приложение независимо для разных платформ, или использовать кроссплатформенные средства? У каждого из вариантов есть свои положительные и отрицательные стороны. Преимущества и недостатки обоих подходов будут совершенно по-разному сказываться в разных ситуациях. Значит, этот вопрос стоит рассматривать не в вакууме, а применительно к конкретным условиям [1]. Нюансы ещё и меняются со временем: например, не так давно Microsoft, купив Xamarin, сделали этот продукт бесплатным. Исходя из этого, поменялись и приоритеты.

Как правило, выход любого бизнеса в интернет протекает по следующему сценарию: сначала компания запускает сайт, затем его адаптируют под мобильные устройства, и если наблюдается прирост трафика, появляется смысл закрепиться среди владельцев мобильных гаджетов, и компания выпускает приложение.

Сравнивать мобильный сайт и приложение нет смысла – второе однозначно выигрывает за счет широты своих возможностей и отзывчивого интерфейса, взаимодействовать с которым через телефон или планшет гораздо комфортнее. Кроме того, приложение может работать без постоянного подключения к интернету.

Вне зависимости от того, на чем построен бизнес – на продажах, предоставлении услуг или просветительской деятельности, сегодня невозможно не учитывать время, которое люди проводят перед экранами мобильных устройств.

1.1 Нативный подход

Если разработчики в процессе написания приложения пользуются принятым для конкретной платформы языком программирования, Objective-C и Swift для iOS или Java для Android, такое приложение будет называться ***нативным*** (англ. native – родной, естественный). Такие приложения могут получать доступ ко всем службам и сервисам телефона: камере, микрофону, геолокатору, акселерометру, календарю, медиафайлам, уведомлениям и так далее .

Нативными приложениями являются те, с которыми пользователь сталкиваетсяс первого дня использования устройства. Это установленные по умолчанию браузер, почтовый клиент, адресная книга, будильник, календарь и другие стандартные программы.

Плюсы нативной разработки[2]

Разработка на родных технологиях и языках под iOS и Android имеет следующие положительные моменты:

1. Скорость работы приложения. Т.к. приложение создается с использованием оригинальных инструментов разработки (XCode, Android Studio), получаемый в результате компиляции проекта код является оптимальным для данной платформы. Приложение получает полную аппаратную поддержку устройства (обработка тех же изображений осуществляется отдельным процессором, специально для этого предназначенным — GPU), используется многопоточность для реализации сложных задачи и загрузки контента в фоне, в процессе разработки программисты могут измерять скорость работы всех участков кода и при необходимости их оптимизировать, в их распоряжении так же есть инструменты по мониторингу использования оперативной памяти, поиску возможных утечек и т.д.

2. Гибкость в реализации. В отличие от ограничений в построении интерфейса и сложности визуальных эффектов, накладываемых фреймворками для кроссплатформенной сборки проектов, в случае нативной разработки реализовать можно все, на что способны технологии той или иной мобильной операционной системы.

3. Использование последних технологий, зависимости от кроссплатформенных фреймворков. Новый программный и аппаратный функционал, предоставленный компаниями-производителями устройства и операционной системы, становится доступен для реализации сразу после выпуска соответствующих обновлений. К примеру, в iOS 9 заложена возможность поиска внутри приложений: в каждом из них должен быть реализован специальный метод, который возвращает результаты по определенному поисковому запросу. В результате для тех приложений, в которых этот функционал реализован, доступна возможность поиска контента через системный раздел поиска в iOS, там же, где осуществляется поиск приложений, контактов, событий и прочей информации. В случае с кроссплатформенной разработкой для реализации подобного функционала придется ждать не только релиза iOS 9, но и обновления соответствующего фреймворка, причем когда появится поддержка тех или иных новых возможностей и появится ли вообще, предсказать невозможно.

4. Легкость и качество тестирования. Помимо упомянутом в п. 1 инструментария для контроля использования приложением аппаратных ресурсов устройства в распоряжении разработчиков и тестировщиков есть целых комплекс технологий. Во-первых, опять же, все параметры системы в процессе работы приложения контролируются автоматически. Если приложение стало использовать больше памяти чем это ожидается или больше ресурсов центрального процессора, это не пройдет незамеченным. Во-вторых, возможности в широком применении юнит-тестов — автоматического тестирования практически каждого метода в приложении. Если какая-то часть приложения перестала работать корректно вследствие каких-либо изменений кода, новая версия просто не соберется, а программист сразу увидит причину. В-третьих, доступны широкие возможности в интеграции систем удаленного мониторинга ошибок: в каждый нативный проект встраивается

соответствующий функционал, который позволяет увидеть ошибку и ее причину, возникшую на устройстве любого пользователя.

5. Полная поддержка со стороны магазинов приложений App Store и Google Play. Обе компании, и Apple, и Google, заинтересованы в том, чтобы пользователи получали максимально положительный опыт при использовании приложений на соответствующих платформах, который возможен на текущий момент. Это означает что приложение должно выглядеть максимально качественно (т.е. если у экрана высокое разрешение, а изображения расплывчаты, в App Store приложение просто не пропустят), должно работать настолько быстро, насколько это возможно (если приложение отображает небольшой список элементов за 20-30 секунд — его так же не пропустят), и вообще все должно быть красиво и удобно. Если какие-то из этих параметров слишком низки или вообще не выполнены, приложение не пропустят в магазин. Если же они не на высоте, чего добиться с кроссплатформенными технологиями крайне сложно, а часто и невозможно в принципе, то ваше приложение никогда не будет рассмотрено соответствующими компаниями для размещения в специальных рекламных разделах (Featured). Среди приложений, находящихся во Featured-разделах и App Store, и Google Play, нет ни одного, сделанного с помощью кроссплатформенных технологий, за исключением игровых проектов, в которых интерфейс не является системным.

1.2 Кроссплатформенный подход

Кроссплатформенный подход предполагает унификацию большей части программного кода приложения, его независимость от последующего выбора платформы. Как правило, кроссплатформенная разработка подразумевает использование специальных утилит (фреймворков) для создания приложения на основе семейства языков JavaScript. Вся структура и логика приложения создается с помощью таких инструментов (PhoneGap, Titanium, Xamarin, Cordova и др.) на JavaScript, а затем оборачивается в нативный запускающий элемент, т.е. интегрируется в базовый проект для XCode или Android Studio, что позволяет создавать сборки проекта с одной и той же логикой под несколько операционных систем сразу.

Ближайшая аналогия в случае с персональными компьютерами: все, что происходит в браузере (сайты, онлайн-редакторы текста и графики, социальные сети, чаты, форумы) – это кроссплатформенные технологии.

Плюсы кроссплатформенной разработки [2]

Кроссплатформенный подход к разработке имеет следующие положительные моменты:

1. Требуется меньше ресурсов для реализации приложения сразу под несколько платформ. В этом собственно и суть кроссплатформенного подхода — один и тот же код работает и на iOS, и на Android. Программистов, занимающихся проектом, нужно ровно в два раза меньше. Дизайнер делает

только один набор графики. Все это снижает количество рабочих часов и как следствие бюджет проекта.

2. Меньшее время на разработку. За счет отсутствия уникальных элементов интерфейса и более простых технологий время на создание простых продуктов как правило меньше.

3. Упрощенный цикл обновления продукта. Если в проект нужно что-то добавить или исправить какую-то ошибку, это делается сразу для всех платформ, на которых распространяется проект.

4. Возможность использования мобильной версии сайта. Большинство кроссплатформенных решений используют семейство JavaScript языков, поэтому если у вас уже есть мобильная версия сайта, значительная часть кода и материалов может быть использована в приложении без изменений.

5. Использование единой логики приложения. Заложенная логика в работу приложения будет работать гарантированно одинаково для всех платформ. Довольно часто это может являться и минусом из-за разной архитектуры операционных систем (яркий пример — кнопка Назад в навигации между экранами: в Android предусмотрена аппаратная кнопка Back для этих целей, в iOS — движение пальцем от левой части экрана или же наличие кнопки в левой части навигационной панели; если кнопку не делать вовсе, пользователи iOS не смогут вернуться назад; если сделать, но не на том месте и выглядящую нестандартно — пользователям iOS будет непривычно и неудобно; если сделать как в iOS, будет непривычно пользователям Android), однако написанная и отлаженная один раз логика содержит потенциально меньшее количество ошибок и расхождений в своей работе: вам не придется проделывать двойную и тройную работу по поиску проблем на каждой платформе.

1.3 Гибридные приложения

Существует класс задач, в которых наиболее эффективно комбинировать указанные подходы. Например, использовать кроссплатформенные преимущества HTML для оформления контента, а требовательные к скорости отзывчивости меню и элементы управления сделать нативными, затратив на это минимум усилий, времени и бюджета. Такие приложения называются ***гибридными.*** В этом случае только объём нативного кода определяет, какому подходу больше соответствует разработка приложения [1].

Какие ситуации приводят к слиянию подходов? Предположим, что клиенту нужна незатейливая новостная лента, где не будет ничего, кроме текста и изображений. Исходя из этой задачи, разработчик принимает решение использовать кроссплатформенный подход. Но если через некоторое время заказчик пожелает, чтобы приложение хранило большое количество данных или обрабатывало звук и графику, задача усложняется. Для этих целей нужно писать нативный код под каждую конкретную платформу, и некогда полностью кроссплатформенное приложение превращается в гибридное.

Распространено заблуждение, что за любой иконкой на рабочем столе пользователя ждёт нативное приложение. Но на рабочий стол можно вывести даже ярлык для сайта, поэтому иконка ничего не гарантирует, и по ту сторону с равной вероятностью может оказаться как нативное приложение, так и любое другое.

1.4 Выводы

К выбору той или иной стратегии всегда приводят индивидуальные обстоятельства. Могло сложиться впечатление, что кроссплатформенному приложению в равной степени комфортно на всех платформах, вплоть до самых непопулярных. Требуется оговорка: чтобы это убеждение соответствовало действительности, под каждую платформу, возможно, придётся писать кусок дополнительного кода. В случае же нативных приложений можно рассчитывать на их отличную работу, но для каждой платформы требуется разрабатывать свою версию.

С технической точки зрения, с точки зрения качества создаваемого интерфейса, нативная разработка имеет гораздо больше плюсов. Однако есть сферы, в которых кроссплатформенные технологии являются оправданными: это игровой сектор и тестовые проекты.Современные игры пишутся в подавляющем большинстве на кроссплатформенных технологиях, это сильно ускоряет разработку без ущерба для качества, т.к. в этом случае используются специальные графические фреймворки .

В рамках реализации курсовой работы (в учебных целях) использование кроссплатформенного подхода является достаточным и вполне оправданным.

2 ОБЗОР КРОССПЛАТФОРМЕННЫХ РЕШЕНИЙ ДЛЯ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

В данной главе, на основе материала [3] представлены 6 решений для кроссплатформенной разработки, которые были популярны в 2016 году. А именно, кроссплатформенные фреймворки PhoneGap, Xamarin, Unity, Qt и Appcelerator Titanium, Telerik Platform , которые на сегодняшний день занимают 80% рынка кроссплатформенной разработки для мобильных устройств.

В таблице 2.1. представлены основные характеристики для каждого из указанных фреймворков.

Таблица 2.1

	PhoneGap	Xamarin	Unity	Qt	Appcelerator Titanium	Telerik AppBuilder
Языки	JavaScript, HTML5, CSS3 и нативные языки (Java, Objective-C, C#)	C#, Xaml	C#, UnityScript, Boo	C++ QML	JavaScript, Python, Ruby, PHP	.Net, JavaScript, HTML5, Java, PHP
Поддерживаемые платформы	Android, iOS, Windows Phone, Blackberry, WebOS, Symbian, Bada, Ubuntu, Firefox OS.	iOS, Android, Windows Phone and Windows 8/RT, Tizen	Android, iOS, Windows Phone, Tizen, PS 4, Xbox One	Android, iOS, WinRT, Windows, Symbian, Linux, QNX	iOS, Android, BlackBerry, Windows, Tizen, Denso	iOS, Android, BlackBerry, Windows, Windows Phone
Цены	Платная версия: от 9.99$ Бесплатная версия: доступна Adobe Creative Cloud Membership: доступно	Xamarin Studio Community: бесплатно Visual Studio Community: бесплатно Visual Studio Professional: доступно Visual Studio Enterprise: доступно	Personal Edition: бесплатно Professional Edition: от 75 $ в месяц	Есть бесплатная версия. Платные версии начинаются от 79$.	Есть бесплатный пробный период Indie: 39$ в месяц Pro: $99 в месяц	Есть бесплатный пробный период Цена от 39$ в месяц
Open source	+	-	-	+	+	-
UI	Web	Native	UI Canvas	Native	Native	Web

2.1 Фреймворк PhoneGap

PhoneGap позволяет создавать мобильные приложения используя стандартные веб технологии (HTML5, JavaScript and CSS3). В результате это привело к быстрому росту популярности фреймворка, с его помощью можно обойтись без разработки на таких языках программирования как :Java for Android, Objective-C for iOS и C#.

PhoneGap Build позволяет делать сборки для iOS, Android и Windows Phone одновременно, без необходимости устанавливать какие-либо SDK tools (при разработке всё равно лучше делать сборку локально, хотя бы на Android, перед отправкой на тестирование). Но что более важно, этот сервис позволяет делать сборки для iOS в облаке без наличия Mac.

PhoneGap представляет возможность использовать нативные функции мобильного устройства по работе с:

- акселерометром,
- камерой,
- компасом,
- контактами,
- файловым хранилищем,
- геолокацией,
- базой данных,
- событиями, уведомлениями,
- медия и др.

Если приложение не выходит за рамки данных пунктов, то скорость разработки с использованием фреймворка PhoneGap будет на порядок выше, чем разработка нативного приложения для каждой из платформ.

Преимущества:

- PhoneGap имеет простое API, что позволит легко начать разработку, для тех кто сталкивался с HTML, CSS и JavaScript.
- Возможность использования любых существующих JavaScript библиотек (JQuery, Prototype, Sencha Touch)
- Поддержка всех мобильных платформ

Недостатки:

- Пользовательский интерфейс визуализируется с помощью встроенного браузера. Это создает трудности в получении обратной связи по сравнению с нативным приложением.
- Часто существующие плагины оказываются устаревшими, поэтому иногда придется писать свои.

2.2 Фреймворк Xamarin

Xamarin позволяет создавать одну единственную логику приложения с применением C# и .NET.

Функционально платформа Xamarin представляет ряд субплатформ. Эти субплатформы играют большую роль — через них приложения могут направлять запросы к прикладным интерфейсам на устройствах. Определяется визуальный интерфейс, привязывается логика на C#, и все это будет работать

на Android, iOS и Windows Phone.

Преимущества:

- Большое и развивающееся сообщество.
- Разработчики могут использовать TestCloud для тестирования приложений автоматически.
- Если вы уже знакомы с C# и .NET то вам не нужно будет тратить много времени на изучение нескольких новых фреймворков.
- Можно повторно использовать уже написанный код.
- Приложения под разными системами будут выглядеть очень похоже.
- Динамическая верстка для iOS в бесконечное число раз проще, чем использование constraints вручную.
- За счет CustomRenderer‘ов стандартные контролы легко дополняются произвольными свойствами (например, сделать градиентную заливку кнопок — дело пары минут, хотя «из коробки» это не работает).

Недостатки:

- Некоторые интерфейсные паттерны тяжело реализовать на monodroid и очень тяжело на monotouch, так как решения по умолчанию для той или иной фитчи опираются на костыли платформы, которые могут попросту не работать в Xamarin.
- Возникают проблемы со стороны платформы mono, monotouch и monodroid. Ваше приложение должно удовлетворять особенным требованиям стабильности.
- Android страницы невозможно расположить как часть уже существующего Activity/Fragment.
- Реализованы не все контролы.

2.3 Фреймворк Telerik AppBuilder

Одной из основных причин использовать AppBuilder является полноценная онлайн IDE. Она позволяет создавать, тестировать и даже публиковать гибридные приложения с любого компьютера или мобильного устройства, без необходимости в его загрузке.

Возможность создавать iOS приложения работая на Windows или Linux еще одно преимущество.Ппринадлежность AppBuilder к Telerik Platform дает возможность пользоваться такими фичами как аналитика, всплывающие уведомления, авторизация пользователей и облачным хранилищем.

Преимущества:

- Telerik предоставляет плагины Visual Studio и Sublime Text для AppBuilder.
- AppBuilder предлагает быстрый способ импорта плагинов Cordova.
- Полноценная онлайн IDE.
- Легок в использовании и изучении

Недостатки:

- Небольшое сообщество

2.4 Фреймворк Unity

Мультиплатформенный инструмент для разработки 2D и 3D приложений и игр Unity, также один из лучших инструментов для демонстрации 3D контента. Созданные с помощью Unity приложения работают под операционными системами Windows, OS X, Linux, Android, Apple iOS, Windows Phone, BlackBerry, а также на игровых приставках Wii, PlayStation 3 и Xbox 360.

Преимущества:

- Отличный вариант для создания мобильных игр для целого ряда устройств
- 3D-движок дает высококачественные результаты без каких-либо сложных конфигураций
- Есть много хороших бесплатных плагинов
- Unity позволяет разработчику сделать свои собственные шейдеры и изменить путь, которым Unity визуализирует игру.

Недостатки:

- UI и сложность в использовании для новичков
- Исходный код недоступен
- Компиляторы Unity не оптимизированы для ARM процессоров на некоторых мобильных устройствах.

2.5 Фреймворк Qt

Qt библиотека для создания кроссплатформенных оконных приложений на C++. Qt стоит рассматривать не столько как набор классов для создания GUI, а скорее как полноценный инструментарий классов на все случаи жизни. Есть возможность разрабатывать программы не только на C++, но и языке QML, сильно схожим с JavaScript. Это особая ветвь развития Qt, направленная на быстрое прототипирование и разработку мобильных приложений.

Преимущества:

- Qt имеет множество хороших инструментов которые помогут в разработке, например: IDE QT Creator, Qt Designer и code profiling.
- Он имеет библиотеки, содержащие интуитивно понятные API интерфейсы для элементов, таких как сети, анимации и многое другое.

Недостатки:

- Qt сложен для начинающих

2.6 Фреймворк Appcelerator Titanium

Titanium — это полностью открытая платформа для разработки, развертывания, распространения, и, в конечном итоге, для исполнения веб-приложений. Позволяет создавать мобильные приложения на JavaScript, HTML и CSS.

Возможность создавать современные, а главное — нативные приложения, используя любую популярную на сегодняшний день операционную систему: Windows, GNU/Linux или MacOS X.

Приложения созданные с помощью данного SDK будут действительно нативными. Контроллер навигации на Андроиде будет выглядеть привычно и

не так как на iOs. Причем не только вид, но и сам код приложения будет тоже нативный. Это не мешает создавать и классический WebView и наполнить его желаемым web контентом.
Преимущества:

- JavaScript позволяет легко разрабатывать приложения без использования языков платформы.
- Appcelerator позволяет делать аналитику в режиме реального времени
- Использование native API даст более высокую производительность для приложений, которые не очень велики.

Недостатки:

- Есть задержки при запуске приложения из-за загрузки библиотеки
- Трудно создавать сложные приложения, так как использование JavaScript отрицательно сказывается на производительности приложений.

2.7 Выводы

В главе представлено комплексное описание наиболее популярных фреймворков, описаны их сильные стороны и недостатки. В процессе разработки приложения студент может использовать любой из описанных фреймворков, либо выбрать другой по своему усмотрению.

3 ТРЕБОВАНИЯ К НАПИСАНИЮ И ОФОРМЛЕНИЮ КУРСОВОЙ РАБОТЫ

Курсовая работа – это более глубокое и объемное исследование избранной проблемы учебного курса, чем реферат, доклад и контрольная работа. Общий объем курсовой работы ***20-25 страниц***. При этом ***приложения не входят в объем работы***, что позволяет исследователю уложиться в установленные рамки.

3.1 Цель курсового проектирования

Целью курсовой работы является формирование у студентов навыков проектирования и разработки кроссплатформенного программного средства на основе теоретических и практических знаний, полученных в курсах «Операционные системы», «Операционные системы и кроссплатформенное программирование» и «Технологии программирования». В процессе выполнения курсовой работы необходимо грамотно выбрать средство разработки (обосновать свой выбор), выполнить проектирование программы, ее реализацию, представить пример использования программы.

3.2 Структура курсовой работы

Курсовая работа должна включать в себя следующие разделы:

а) титульный лист;

б) лист задания;

б) оглавление ;

в) введение;

г) главы основной части;

д) заключение (выводы);

е) список использованных источников;

ж) приложения – при необходимости;

Введение

Обосновывая тему, студент должен определить ее место и значимость изучения. Необходимо обозначить цель своей работы, четко выделить конкретные задачи, с помощью которых будет достигаться цель исследования. Введение не должно превышать 1/10 части общего объема работы (в среднем 1-2 страницы).

Автор вправе переставить местами названные элементы введения, исходя из принципа оптимальной подачи материала курсовой работы.

Введение состоит из следующих элементов:

1) *обоснование (актуальность) темы* – это степень ее важности в определенный момент и в конкретной ситуации для решения данных проблемы, вопроса или задачи. Освещение актуальности не должно быть многословным.

2) *описание степени разработанности проблемы* – перечисление основных точек зрения, подходов и методологических основ исследований;

3) *указание предмета и объекта работы:* объект исследования – это процесс или явление, порождающее проблемную ситуацию и избранное для изучения.

4) *постановка цели и задач исследования.* Цель – это результат, который необходимо получить при проведения исследования, некоторый образ будущего. Задачи исследования – это те исследовательские действия, которые необходимо выполнить для достижения поставленной в работе цели, решения проблемы;

Основная часть

Основная часть курсовой работы обычно состоит из двух теоретических и практических или экспериментальных глав, при этом каждая глава – из двух-трёх параграфов и выводов по главе. Формулировка глав и параграфов должна быть четкой, краткой и в последовательной форме раскрывать содержание работы. После каждой главы делается вывод по рассмотренному материалу (2-3 предложения).

Первая глава представляет собой аналитический (теоретический) обзор по проблеме, рассматриваемой в работе. На основе изучения литературных источников отечественных и зарубежных авторов рассматривается сущность исследуемой проблемы, анализируются различные подходы их решения, дается их критический анализ, излагается собственная позиция исследователя.

Необходимо выбрать среду разработки, описать ее преимущества и недостатки, грамотно, аргументировано обосновать свой выбор. Разработать структуру будущей программы, выделить ее основные части и продумать правила взаимодействия. Определиться с внешним видом будущего программного средства.

Вторая глава посвящена описанию разработанного Вами программного приложения. Здесь приводится интерфейс программы, руководство по использованию программы (возможно, некоторые правила, в случае, например, игрового приложения). Описываются графические составляющие, приводится контрольный пример использования программы.

Заключение

Заключение объемом 1 – 2 страницы должно содержать в концентрированном (тезисном) виде без какой-либо аргументации ранее обоснованные студентом в тексте работы наиболее важные выводы. Автор курсовой работы должен выделить собственный вклад в разработку темы, подчеркнуть значимость своих выводов и наблюдений. Качество работы увеличится, если ее студент сумеет не только грамотно и профессионально подвести итоги, но и определить перспективность направлений дальнейшего исследования темы на новом уровне. Не стоит включать в заключение цитаты и примеры.

Список использованных источников.

В список использованных источников и литературы включаются все изученные или использованные автором книги, статьи, нормативные акты и другие источники, имеющие отношение к избранной теме, независимо от того, цитируются ли они в работе. Обязательно включение в список литературы всех цитируемых либо упомянутых в тексте публикаций.

Список сокращений составляется в алфавитном порядке. Точки между буквами, обозначающие сокращенные слова, не ставятся.

Приложения

В Приложение выносится дополнительный материал, который может нарушить связность изложения основного содержания и препятствовать его целостному восприятию. В контексте данной работы, в приложение выносится текст программы.

3.3 Оформление работы

Оформление – одна из важнейших стадий работы над курсовой работой. Причем определенные элементы оформления нельзя откладывать «на потом» – на то время, когда текст в своей основе уже будет написан.

Работа должна быть оформлена аккуратно с соблюдением ряда требований.

Объем работы зависит от многих факторов: масштабности и сложности темы, хронологических рамок исследования, количества привлеченных источников, стиля изложения. Рекомендуемый объем курсовой работы 20-25 страниц (без приложения).

Общие требования

Текст пояснительной записки оформляют в соответствии с требованиями СТП РГУПС 2-07.

Курсовая работа выполняется на листах белой бумаги плотностью 80 г/см3 и форматом 210×297 мм (ГОСТ 9327). На листах оставляются поля по всем четырём сторонам, вычерчивается внутренняя рамка. В правом нижнем углу проставляется номер страницы. Размер левого поля 30 мм, правого – 10 мм, верхнего – 15 мм, нижнего – 20 мм.

Курсовая работа печатается с применением ЭВМ на одной стороне листа. Для печати основного текста используется шрифт Times New Roman кегля 14. Межстрочный интервал – полуторный. Размер абзацного отступа – 1,25 см. Текст должен быть выровнен ***по ширине*** страницы; каждый абзац рекомендуется начинать с красной строки (устанавливается опцией ***Формат / Абзац / Отступ***). Страницы должны быть пронумерованы , номер проставляют в нижнем поле по центру, при нумерации учтите, что первой страницей является титульный лист, второй – лист задания и оглавление, на которых номер страницы не ставится. Номера страниц проставляются с введения.

Каждая глава, введение, заключение, список используемой литературы, приложение (но не пункты и параграфы) должны начинаться с новой страницы.

Необходимо правильно оформлять общепринятые условные сокращения. После перечисления пишут т.е. (то есть), и т.д. (и так далее), и т.п. (и тому подобное), и др. (и другие), и пр. (и прочие); при ссылках: см. (смотри), ср. (сравни); при цифровом обозначении веков и годов: в. (век), вв. (века), г. (год), гг. (годы).

Текст должен быть написан грамотно, с соблюдением всех требований русского языка. Язык пояснительной записки должен быть сжатым и точным, свойственным научно-техническим документам. Не следует злоупотреблять описаниями устройств или программного обеспечения, известными из литературы. Достаточно коротко перечислить их существенные особенности и дать библиографическую ссылку. Не должны использоваться жаргонные

технические выражения. При необходимости сокращенного обозначения выражений или слов принятые сокращения приводятся в перечне сокращений, символов, специальных терминов.

Изложение текста курсовой работы даётся от первого или третьего лица множественного числа или в безличной форме, например: "значение коэффициента принимаем...", "принимают...", "принято".

Заголовки разделов, подразделов, основной текст курсовой работы

Текст работы должен быть разделён на разделы, которые могут включать в себя подразделы.

Каждый раздел имеет порядковый номер, обозначенный арабской цифрой без точки. Каждый раздел работы следует начинать с нового листа. Реферат, содержание, перечень сокращений, введение, заключение, список использованных источников не нумеруются. Подразделы нумеруются арабскими цифрами в пределах каждого раздела. Номер подраздела состоит из номера раздела и подраздела, разделённых точкой. В конце номера подраздела точка не ставится. Подразделы состоят из пунктов. Номер пункта включает номер раздела, порядковый номер подраздела в разделе и порядковый номер пункта в подразделе, разделенные точками (например, «2.5.3» – третий пункт пятого подраздела второго раздела).

Текст заголовка разделов, подразделов рекомендуется выделять полужирным шрифтом:

Пример: **1.3 (третий подраздел первого раздела).**

Текст заголовков пунктов и подпунктов выделять полужирным шрифтом не следует. Единственный подраздел в разделе (а также единственный пункт в подразделе) не допускается.

Заголовки разделов, подразделов и пунктов записывают строчными буквами (кроме первой прописной) с абзацного отступа. Переносы слов в заголовках не допускаются. Точку в конце заголовков раздела, подраздела или пункта не ставят. Если заголовок состоит из двух предложений, их разделяют точкой.

Заголовки должны быть ясными и четкими, исключающими неопределенность их толкования. Заголовок каждого раздела и подраздела должен отражать не только объект исследования, но и раскрывать содержание изложенного материала в разделе и подразделе.

Иллюстрации

К иллюстрациям относятся рисунки, схемы, фотографии, графики, номограммы, диаграммы, все виды чертежей. Они размещаются сразу после ссылки на них в тексте работы и называются рисунками. Нумеруются арабскими цифрами в пределах соответствующего раздела, например: Рисунок 2.1 (первый рисунок второго раздела).

Если рисунок один, то он обозначается: «Рисунок 1». Иллюстрации каждого приложения обозначают отдельной нумерацией арабскими цифрами с добавлением перед цифрой обозначения приложения. Например: «Рисунок А.5» – пятый рисунок Приложения А.

При ссылках на иллюстрации следует писать:

« … в соответствии с рисунком 2.1» при нумерации в пределах раздела и « … в соответствии с рисунком 2» при сквозной нумерации.

Название иллюстрации помещают под рисунком.

Слово «Рисунок», номер рисунка и название иллюстрации помещают под рисунком через тире, например:

Рисунок 1 – Составные элементы интерфейса пользователя

Располагают симметрично тексту.

В конце номера рисунка и после названия рисунка точки не ставятся, номер от названия отделяется дефисом.

В тексте обязательно должны быть ссылки на все иллюстрации. Иллюстрации размещаются по тексту после первой ссылки на них в тексте. Можно располагать иллюстрации на отдельных листах.

Список использованных источников

Сведения об источниках, включенных в список, приводятся в соответствии с требованиями ГОСТ 7.1 – 2003. Цитирование из теоретических источников должно быть взято в кавычки с указанием источника и страницы. Например: /16, с. 25/.

Нумерация источников даётся в порядке упоминания их в тексте дипломного проекта (работы) или в алфавитном порядке.

Например.

1 **Голицына, О.Л.** Системы управления базами данных: учебное пособие для вузов / О.Л. Голицына, И.И. Попов, Т.Л. Партыка. – М.: Инфра-М, 2006. – 432 с.

2 **Харламов, А.И.** Общая теория статистики: Статистическая методология в изучении коммерческой деятельности: учебник / А.И. Харламов, О.Э. Башина, В.Т. Батурин и [др.] / Под ред. А.А. Спирина, О.Э. Башиной. – М.: Финансы и статистика, 1994. – 295 с.

При библиографическом описании электронного документа, полученного с web-страницы, необходимо включить следующие обязательные элементы:

Автор. Заглавие страницы.[Указание типа документа]. (Электронный адрес (URL)). Дата обращения.

Пример.

1 **Травин, А.** Три поисковика Рунета, не считая Google. [Электронный документ]. (http://www.netoscop.ru/theme/2001/06/21/2662.html). Проверенно 21.06.2017

Использование чужого материала без ссылки на автора и источник заимствования является плагиатом.

Приложения

Приложения оформляют на листах формата А4, допустимы и другие размеры, кратные формату А4.

Каждое приложение следует начинать с новой страницы с указанием наверху посередине страницы слова «Приложение» и его обозначения в виде заглавной буквы русского алфавита.

Приложения обозначают заглавными буквами русского алфавита, начиная с А, за исключением букв Ё, З, И, Й, О, Ч, Щ, Ь, Ы, Ъ, после буквы Я приложения обозначаются арабскими цифрами. Если в проекте одно приложение, оно обозначается Приложение А.

Ниже обозначения в скобках указывается его характеристика: справочное, рекомендуемое или обязательное.

В тексте на все приложения должны быть ссылки. Приложения располагают в порядке ссылок на них в тексте проекта, они должны иметь общую с остальной частью документа сквозную нумерацию страниц.

Ссылки на приложение оформляются следующим образом: «...смотри приложение А».

Чистовой вариант курсовой работы надо тщательно выверить. В нём должны быть исправлены все ошибки, опечатки, внесены необходимые поправки, тщательно сверены фамилии, цитаты, названия.

4 ПРИМЕРНАЯ ТЕМАТИКА КУРСОВОЙ РАБОТЫ

1. Разработка игрового приложения в стиле 3D-шутеры, «бродилки-стрелялки»
2. Разработка игрового приложения в стиле «Аркады»
3. Разработка игрового приложения в стиле «Стелс-экшен»
4. Разработка игрового приложения в стиле «Симуляторы и Менеджеры»
5. Разработка игрового приложения в стиле «Стратегии»
6. Разработка игрового приложения в стиле «Квесты»
7. Разработка игрового приложения в стиле «Головоломки»
8. Разработка музыкального приложения
9. Разработка ролевого игрового приложения
10. Разработка текстового игрового приложения
11. Разработка игрового приложения «Обучающее приложение для малышей»
12. Разработка приложения для интернет-магазина (ассортимент магазина, цены, он-лайн продажи и т.п.)
13. Разработка мобильного приложения для застройщика (этапы строительства, закупка строительных материалов и т.п.)
14. Разработка мобильного фитнес-приложения (подсчет калорий, рекомендации по правильному питанию и т.п)
15. Разработка приложения парка такси. Клиентская часть с возможностью вызова машины, просмотра назначенного ему транспортного средства, ориентировочного времени прибытия и т.д.
16. Разработка приложения парка такси. Разработка приложения водителя (например, с указанием пробок, маршрутов и т.д.)
17. Разработка приложения парка такси. Приложение для диспетчера такси с просмотром свободных транспортных средств, назначением такси на конкретный адрес, расчетом стоимости поездки и.т.п.
18. Разработка инженерного калькулятора
19. Разработка приложения для общения в социальных сетях
20. Разработка приложения для путешествий – с возможностью просмотра достопримечательностей, комментариев, прокладки маршрута и т.п.
21. Разработка приложения для составления заметок/списка покупок
22. Разработка приложения – системы тестирования по различным критериям

– психологические тесты
– обучающие тесты
– тесты проверки уровня знаний

23. Разработка приложения работника библиотеки (классификация, поиск необходимой литературы, отметки о выдаче и т.п)
24. Клиент-серверные приложения (создание сокетов; получение информации от приложений - клиентов; вывод на экран полученных результатов; конвертацию потоков для их дальнейшей обработки; установка соединения с сервером и т.п.)

25. Разработка приложения для автостоянки (время прибытия машины, гос.номер, оплата и т.п.)
26. Разработка приложения для обмена фотографиями
27. Разработка приложения-шагомера, способного подсчитывать и запоминать количество шагов, пройденных за определенный период, вести статистику, строить графики и т.п.
28. Разработка приложения «Фонарик++», позволяющего управлять светодиодной вспышкой смартфона, выставлять период ее мерцания
29. Разработка приложения по «Alarm», для установки напоминания, заданного пользователем. Основные функции: установка даты и времени, дня недели, мелодии уведомления и ее длительность и т.п.
30. Разработка приложения - счета. В основные фнкции приложения входит: внесение общей суммы чека; добавление и удаление персоны;редактирование имени; добавление внесенной персоной суммы; расчет долга и т.п.

Возможна реализация курсовой работы по теме, предложенной студентом и своевременно согласованной с преподавателем.

5 БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Кроссплатформенная мобильная разработка: вопросы взрослым [Электронный документ]. (https://habrahabr.ru/company/jugru/blog/302070/). Проверенно 7.07.2017
2. Кроссплатформенная и нативная разработка. [Электронный документ].(http://appcraft.pro/portal/cross_platform_vs_native_app/). Проверенно 7.07.2017
3. **Зарицкий, Д**.Обзор кросс-платформенных решений для разработки мобильных приложений. [Электронный документ]. (https://habrahabr.ru/post/319348/). Проверенно 7.07.2017

Учебное издание

Москат Наталья Александровна

ОПЕРАЦИОННЫЕ СИСТЕМЫ
И КРОССПЛАТФОРМЕННОЕ ПРОГРАМИРОВАНИЕ

Печатается в авторской редакции

Технический редактор А.В. Артамонов

Подписано в печать 04.09.17. Формат 60×84/16.
Бумага газетная. Ризография. Усл. печ. л. 1,4.
Тираж экз. Изд. № 9019. Заказ .

Редакционно-издательский центр ФГБОУ ВО РГУПС.

Адрес университета: 344038, г. Ростов н/Д, пл. Ростовского Стрелкового Полка Народного Ополчения, д. 2.