МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН
ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

К защите допустить
Зав. кафедрой
_____
_____2015 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

на тему:

Разработка мобильного приложения на OSAndroidдля ресторана Basilic

Выпускник _____ Рубцов М.В.
подпись

Руководитель _____ Керимов К.Ф.
подпись

Консультант по БЖД_____ Абдуллаева С.М.
подпись

Рецензент___ _____ Решитов Э.С.
подпись

Ташкент – 2015

МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН
ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Факультет: Программный инжиниринг

Кафедра: Системное и прикладное программирование

Направление (специальность):

5330200– «Информатика и информационные технологии»

У Т В Е Р Ж Д А Ю

Зав. кафедрой _____

« ___ »_____2015г.

ЗАДАНИЕ

на выпускную квалификационную работу студента Рубцова Максима Владимировича

(фамилия, имя, отчество)

1. Тема работы: Разработка мобильного приложения на OS Android для ресторана Basilic
2. Тема утверждена приказом №80-16 по университету от «22» января 2015 года
3. Срок сдачи законченной работы: 11 июня 2015 года .
4. Исходные данные к работе: Задания по постановке задачи, технические данные, справочник ЖКХ, литература, учебные пособия, диаграммы, Интернет-ресурсы
5. Содержание расчётно-пояснительной записки (перечень подлежащих к разработке вопросов): Анализ предметной области, анализ требований к мобильному приложения, проектирование архитектуры мобильного приложения, реализация мобильного приложения на языке программирования.
6. Перечень графического материала: таблицы, пользовательские интерфейсы (скриншоты), диаграммы, рисунки, презентация.
7. Дата выдачи задания : 5 февраля 2015 года

Руководитель _____
подпись

Задание принял _____
подпись

8. Консультанты по отдельным разделам выпускной работы

Раздел	Ф.И.О руководителя	Подпись, дата	
		Задание выдал	Задание получил
Основная часть	Керимов К.Ф.	05.02.2015	20.05.2015
БЖД	Абдуллаева С.М.	11.03.2015	12.05.2015

9. График выполнения работы

Наименование раздела работы	Срок выполнения	Отметка руководителя о выполнении
1.Введение	24.01.2015-28.01.2015	
2.Анализ предметной области	29.01.2015-26.02.2015	
3.Формирование постановки задачи	28.02.2015-5.03.2015	
4.Проектирование мобильного приложения	9.03.2015-31.03.2015	
5.Реализация мобильного приложения	1.04.2015-30.04.2015	
6.Описание мобильного приложения	1.05.2015-20.05.2015	
7.Безопастность жизнедеятельности	22.05.2015-28.05.2015	
8.Написание доклада	1.06.2015-10.06.2015	

Выпускник _____ «_____» _____ 2015 г.

(подпись)

Руководитель _____ «_____» __ _____ 2015 г.

(подпись)

Данная выпускная квалификационная работа посвящена разработки мобильного приложения для ресторана Basilic. В данной работе были рассмотрены методы проектирования приложения, методы создания мобильного приложения для девайса, работающего на базе операционной системы Андройд.

This Final qualification work is devoted to the development of mobile application for the restaurant Basilic. In this paper we examined the methods of application design, methods for creating mobile applications for device, based on the operating system Android.

Ушбу битирув малакавий иши Basilic ресторани учун мобил илова ишлаб чикишга богишланган.

Ушбу ишда мобил иловани лойихалаштириш усуллари ва Андройд ОТ ассосида амалга ошириладигян «девайс» учун мобил иловалар яратилган усуллари келтирилган.

Содержание

Введение

На сегодня все больше людей используют для обмена информацией не ПК и ноутбуки, а современные мобильные телефоны, которые по функционалу практически сравнялись с упомянутыми устройствами. Как следствие, **разработка мобильных приложений** становится все более и более востребованной – над ними сегодня работает не меньше людей, чем над разработкой ПО для полнофункциональных компьютерных систем.

Глава нашего государства Ислам Каримов, постоянно развивает данную сферу, разрабатывая различные программы и определяя перспективы направления дальнейшей деятельности. Для этого наш Президент принял постановление «О мерах по дальнейшему внедрению и развитию современных информационно – коммуникационных технологий». В данном документе были утверждены программы дальнейшего внедрения и развития информационно – коммуникационных технологий в стране на 2012-2014 годы и перечень ИС органов государственного и хозяйственного управления, органов государственной власти на местах, интегрируемых в Национальную ИС в период 2012-2014 годов.

Мобильные приложения нашего времени – это не wap-сайты и элементарные программки с однотипным интерфейсом, а многоуровневые программы, которые способны обеспечить работу пользователя со сложно структурированной информацией и массивными интерфейсами.

Развитие технологий и потребность пользователей в быстром и удобном доступе к информации привели к тому, что мобильные телефоны, а именно, специально разработанные приложения для них, используются в качестве основного источника доступа к информационным и корпоративным ресурсам, таким как социальные сети, коммуникационные службы, информационные порталы и прочее.

В разработке мобильных приложений наиболее востребованными направлениями являются **разработка под Android**, а также **разработка под iPhone / iPad / iOS**. Именно эти платформы являются наиболее популярными мобильными платформами на сегодняшний день.

На сегодняшний день разработка мобильных приложений из прихоти переросла в реальную необходимость. Качественно спроектированный и отлаженный продукт позволит не только упростить работу отдельного человека или компании в целом, но и решать массу бизнес-задач, находясь практически в любой точке земного шара.

Например, интеграция такого продукта с массивными CRM-системами позволяет полностью взять под контроль работу организации или предприятия, находясь в командировке или деловой поездке, а также в отпуске. Таким образом, разработка мобильных приложений делает возможным то, о чем раньше приходилось только мечтать.

Целью настоящей ВКР является разработка мобильного приложения в качестве удобной системы взаимодействия пользователей данного приложения с рестораном.

Приложение предназначено для:

- просмотра меню ресторана не выходя из дома, меню представлено с полным описанием и иллюстрациями
- просмотра фотогалереи ресторана, тем самым можно будет увидеть вид ресторана изнутри
- возможность забронировать столик в ресторане, не выходя из дома
- просмотр новостей ресторана, контактная информация и многое другое

Во время разработки приложения будет проведено:

- изучение и оценка использования различных методов построения

мобильных приложений

- разработка структуры входных и выходных данных;

Задачи ВКР:

- анализ данной предметной области
- создание приложения
- сопровождение и поддержка данного приложения

1.Анализ предметной области

1.1. Мобильное приложение.

Мобильное приложение – это программа, установленная и запущенная на телефоне, коммуникаторе, смартфоне и т.д.

Самым первыми мобильными устройствами можно считать список контактов в телефоне и сервис для отправки\ приёма сообщений. Сейчас, в связи с развитием сотовой связи и беспроводных технологий (Wi-fi, WiMax, 4G) мобильные приложения ушли далеко вперёд.

Преимущества:

- отсутствия ограничений SMS-рассылок по длине, графической и видеоинформации
- удобство продвижения приложений
- возможность сбора дополнительных данных (местоположения, языка и др.)
- неисчерпаемые возможности по интерактивности

Недостатки:

- недостаточно широкое распространение телефонов, поддерживающих мобильные приложения
- пользователь открывает приложение только тогда, когда он этого захочет, а не когда, например, придет SMS
- не очень широкая аудитория (в основном молодежь)
- пока относительная сложность продвижения (требуются специальные компетенции, чтобы сделать приложение популярным)

Приложения могут быть предустановлены на устройство в процессе производства, загружены пользователем с помощью различных платформ для распространения ПО или существовать в формате веб-приложений.

Основные операционные системы, на базе которых создаются мобильные приложения – Android, iOS, BlackBerry, HP webOS, Symbian OS, Bada от Samsung, и Windows Mobile.

Каналы распространения:

- специализированный портал – AppStore, Android Market
- через sms с порталов сотовых операторов
- самостоятельный поиск и скачивание в Интернете.

Мобильные приложения позволяют не только вовлечь участников в интерактивный процесс, но и могут стать рекламным носителем.

1.2. Мобильный web сайт и мобильное приложение.

1.2.1. Определения

Существует два направления технической реализации проектов для мобильных устройств: мобильный web сайт и мобильное приложение.

Мобильным web сайтом будем считать специализированный сайт, адаптированный для просмотра и функционирования на мобильном устройстве. Сайт может включать в себя интерактивные компоненты с использованием JavaScript, HTML5, новых API браузеров. В этом случае такую реализацию называют web-приложением. Далее, чтобы избежать путаницы любой вариант, работающий внутри браузера мобильного устройства, будем называть web-сайтом.

Мобильное приложение – это специально разработанное приложение под конкретную мобильную платформу (iOS, Android, Windows Phone). Обычно приложение разрабатывается на языке высокого уровня и компилируется в нативный код ОС, дающий максимальную производительность.

Существует еще третий вариант – мобильное приложение, включающее в себя компонент браузера. В этом случае часть мобильного приложения чаще всего используется для навигации и интергации с ОС, а web-компонент – для показа контента. Обычные пользователи не могут зачастую отличить такой вариант от нативного мобильного приложения.

1.2.2. Аргументы

Интерфейс

Одним из первых аргументов, которые приводят сторонники приложений – наиболее близкий к ОС и привычный для пользователей интерфейс. Действительно мобильное приложение наиболее тесно интегрировано с платформой и дает реализовать привычный отзывчивый интерфейс. С другой стороны web сайт с помощью хорошего форматирования и использования JavaScript может дать вполне понятный метод взаимодействия. На текущий момент отзыв web сайта значительно уступает приложению, но мощность мобильных устройств продолжает расти и сами браузеры существенно меняются в лучшую сторону. Кроме того, различные версии мобильных ОС могут диктовать свои стандарты, которых приходится придерживаться. При этом некоторые нововведения могут оказаться не совсем понятны обычным пользователям. Существенным в данном случае является то, что наиболее активными пользователями (теми кто выставляет рейтинг и делает комментарии в магазинах приложений) являются те, кто «фанатеет» от последних новшеств мобильной ОС. На это стоит обратить внимание при продвижении проекта – их можно использовать как союзников, помогающих распространению.

Быстродействие

Web сайт, а особенно интерактивный, существенно уступает приложению с точки зрения быстродействия. Браузеры мобильных устройств пока не могут порадовать высокой производительностью, кроме того, web-разработчики используют не самые оптимизированные версии библиотек (плохая реализация этих библиотек никак не сказывается на «больших» браузерах, поэтому с этим там можно смириться). Однако и приложение не всегда может радовать хорошим быстродействием – излишняя анимация, сложный интерфейс значительно снижают «отклик». Кроме того, для сложной графики и анимации приходится использовать языки более низкого уровня, разрабатывать или покупать отдельные специализированные библиотеки.

Интеграция с платформой

В этой области приложения далеко опережают сайт. В приложении существенно больше возможностей для доступа к устройству. Однако выше упоминался уже третий вариант, когда компонент браузера внедряется в приложение и в этом случае такая разница нивелируется. Кроме того, постоянно растет уровень предоставления доступа к возможностям устройства из браузера через расширяющийся набор API.

Наличие Интернет

Web сайт запускается из браузера, поэтому требует постоянного соединения с сетью. Это не имеет значения, если проект реализуется исключительно как онлайновый. Однако даже в этом случае из-за особенностей мобильного доступа в Интернет переход между частями приложения (навигация) связана с неприятными для пользователя задержками. Возможно, использование API для хранения локальных данных решат эту проблему, но пока примеров такого применения найти не удалось.

Мобильные приложения могут осуществлять работу без подключения, выполняя кеширование и обновление данных, если требуется, при появлении соединения. Но все же и приложению нужно подключение в подавляющем большинстве бизнес-решений.

Фрагментация

Для реализации проекта на всех или каких-то определенных платформах требуется разработать приложение для каждой из платформ отдельно, причем на каждой свои среда и язык разработки, свои стандарты интерфейса. В случае мобильного сайта одна версия должна покрывать потребности всех платформ. Так выглядит в теории. Но на практике оказывается, что браузеры на различных платформах функционируют по-разному. Приходится поддерживать либо несколько версий одного сайта, либо в коде подстраивать выдаваемый контент под текущий запрос. Существенные отличия в размерах экрана также сказываются и на верстке сайта.

Ресурсы

Существует такой аргумент, как наличие специалистов. Считается, что специалиста для разработки мобильных приложений очень трудно найти и требуется очень высокая оплата. Учитываем еще то, что под каждую платформу, скорее всего, потребуется отдельный разработчик. В то время как web разработчиков очень много и их услуги сравнительно меньше стоят. Видимо все зависит от конкретной ситуации и конкретного места. Если в наличие есть web-разработчик, то наиболее выгодным будет разработать именно web сайт, если есть мобильные разработчики, то вполне может оказаться не слишком затратным разработка приложения. Но опять же, зависит от проекта – если потребуется серверная часть (а она скорее всего потребуется), то опять же нужен будет web-разработчик, хотя возможно не

такой высокой квалификации и трудоемкость его части будет существенно ниже.

Публикация

Приложения некоторых платформ «завязаны» на определенный магазин (AppStore, Windows Store). Даже если такой жесткой привязки нет, то пользователи все равно привыкли находить приложения в магазинах (Google Play). Такие магазины накладывают существенные ограничения на функции приложений (в первую очередь в области платных услуг), к тому же требуется значительно время на утверждение каждой новой версии. Со своей стороны web сайт доступен сразу, достаточно только открыть браузер и ввести адрес (хотя если присмотреться, то это довольно трудоемкое действие может оказаться). Новая версия web сайта доступна сразу на момент публикации. Возможность предоставления платных услуг никак не ограничивается. Опять же аргумент весьма своеобразный – с одной стороны ограничение и медленная публикация в магазине, с другой – в магазине уже есть огромное число пользователей и уже готовые системы для оказания платных услуг. Тогда как на сайт пользователей надо привести и оплата через сайт на мобильном устройстве остается очень трудоемкой процедурой.

HTML5

Большое внимание в последнее время уделяется аббревиатуре HTML5. Это понятие существенно отличается, если смотреть на него с маркетинговой или технической точки зрения.

Технологически HTML5 это дальнейшее развитие языка разметки HTML. Однако сделан существенный шаг в сторону большей структуризации представления, нежели формата отображения. В язык добавлены большие мультимедийные возможности для проигрывания аудио и видео. Добавлена возможность работать с графикой. Существенно расширен язык

форматирования CSS. В язык JavaScript добавлено несколько API для работы с графикой, локальными данными, мультимедийным контентом. Сам язык существенно переработан в сторону увеличения быстродействия. Стандарт HTML5 все еще находится в разработке и продолжает дополняться.

С маркетинговой точки зрения HTML5 это гораздо более широкое понятие. Под ним понимают еще много дополнительных API в той или иной степени поддерживаемых различными браузерами, многие интересные расширения CSS (в первую очередь в области интерактивного отображения). Основой понятия является высокая интерактивность сайта, которая позволяет пользователям принимать его за нативные приложения.

С точки зрения мобильной разработки существенно разделять обычный web сайт и сайт с использованием HTML5 не имеет смысла. Фактически стандартом любого сайта становится интерактивность в той или иной мере, реализованная с помощью JavaScript и новых API. Не целесообразно выделять отдельно разработчиков web сайта и разработчиков HTML5 – web разработчик должен свободно владеть технологиями HTML5 и использовать их в случае, если проект удобно реализовать с помощью последних разработок.

Выводы

Как оказывается, ни один из приведенных аргументов не склоняет чашу весов в ту или иную сторону. В каждом аргументе есть как преимущества, так и недостатки обоих вариантов решения. Третий, комбинированный, вариант тоже может решить часть проблем, но при этом порождает новые. Поэтому в каждом конкретном случае надо принимать решение исходя из текущей ситуации.

С точки зрения экономии ресурсов самым предпочтительным вариантом выглядит web разработка. Главное – не погрязнуть в тонкостях реализации, предоставить наиболее полезные функции пользователям. Помнить, что

главное – контент, а «красивости» (анимация, графика) отходят на второй план.

Если планируется онлайн работа проекта как основной вариант взаимодействия с пользователем – безусловно, надо начинать с сайта, который может охватывать не только мобильных клиентов, но и пользователей стационарных компьютеров. В случае успеха можно далее реализовать отдельно мобильные приложения на выбранные платформы. Для большинства бизнес-приложений такой вариант наиболее подходит.

Если проект предусматривает больше оффлайн работу и нацелен на мобильных пользователей, то тут стоит отдать предпочтение приложениям. Однако, как упоминалось выше, возможно web разработчик все равно потребуется.

Для реализации игр и других приложений, требующих высокой производительности интерфейса вероятно дальновиднее реализовать через приложения. Существуют кросс платформенные библиотеки для разработки игр, которые позволяют на одном коде (или с минимальными изменениями) реализовать нативные приложения для различных платформ.

1.3. Краткое описание операционной системы Android

Android («**Андро́ид**») — операционная система для смартфонов, планшетных компьютеров, электронных книг, цифровых проигрывателей, наручных часов, игровых приставок, нетбуков, смартбуков, очков Google, телевизоров и других устройств. В будущем планируется поддержка автомобилей и бытовых роботов. Основана на ядре Linux и собственной реализации виртуальной машины Java от Google. Изначально разрабатывалась компанией Android Inc., которую затем купила Google. Впоследствии Google инициировала создание альянса Open Handset Alliance

(OHA), который сейчас занимается поддержкой и дальнейшим развитием платформы. Android позволяет создавать Java-приложения, управляющие устройством через разработанные Google библиотеки. Android Native Development Kit позволяет портировать (но не отлаживать) библиотеки и компоненты приложений, написанные на Си и других языках.

В 86 % смартфонов, проданных во втором квартале 2014 года, была установлена операционная система Android. При этом за весь 2014 год было продано более 1 миллиарда Android-устройств.

1.4. Среда разработки мобильного приложения

В мае 2013 года Google предложила Android Studio, интегрированную среду разработки Android-приложений, опирающуюся на популярную платформу JetBrains IntelliJ. А сейчас этот продукт подошел к первой стабильной версии — под номером 1.0.

Среда, уводящую в сторону от надоевшей и моральной устаревшей Eclipse, однозначно ускорила и повысила эффективность процесса создания софтверных изделий, сделав его более продуктивным.

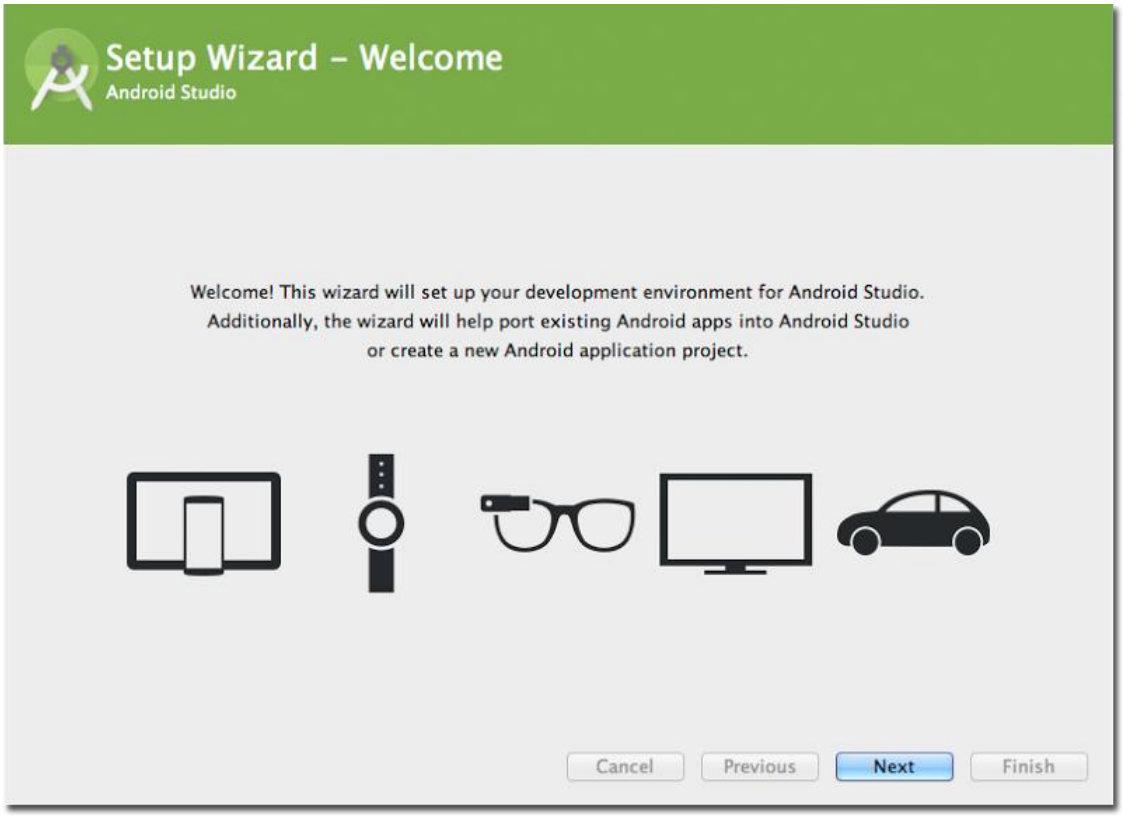


Рис 1.Установка Android Studio

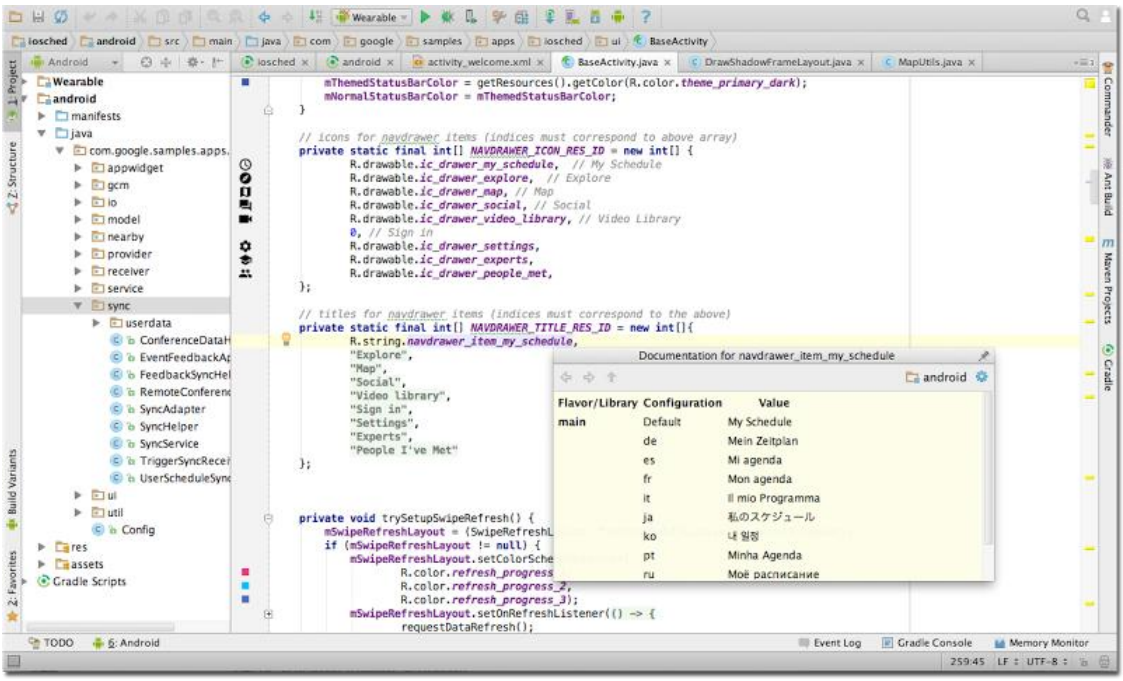


Рис 2.Редактор кода AndroidStudio

Интереснейшей «фишкой» Android Studio выступает вовсе не автоматическое завершение кода, анализ и рефакторинг, а верстка в реальном времени: вы редактируете XML-код, и приложение тут же меняет свой внешний вид. Доступно мгновенное переключение на различные типы верстки и размеры экранов, можно даже менять дизайн интерфейса в самой верстке, и код соответствующим образом это отразит.

Подобное окажется полезным, допустим, в ходе решения вопросов с интернационализацией интерфейса приложения, когда необходимо сразу же видеть, что изменилось при переводе его на другой язык.

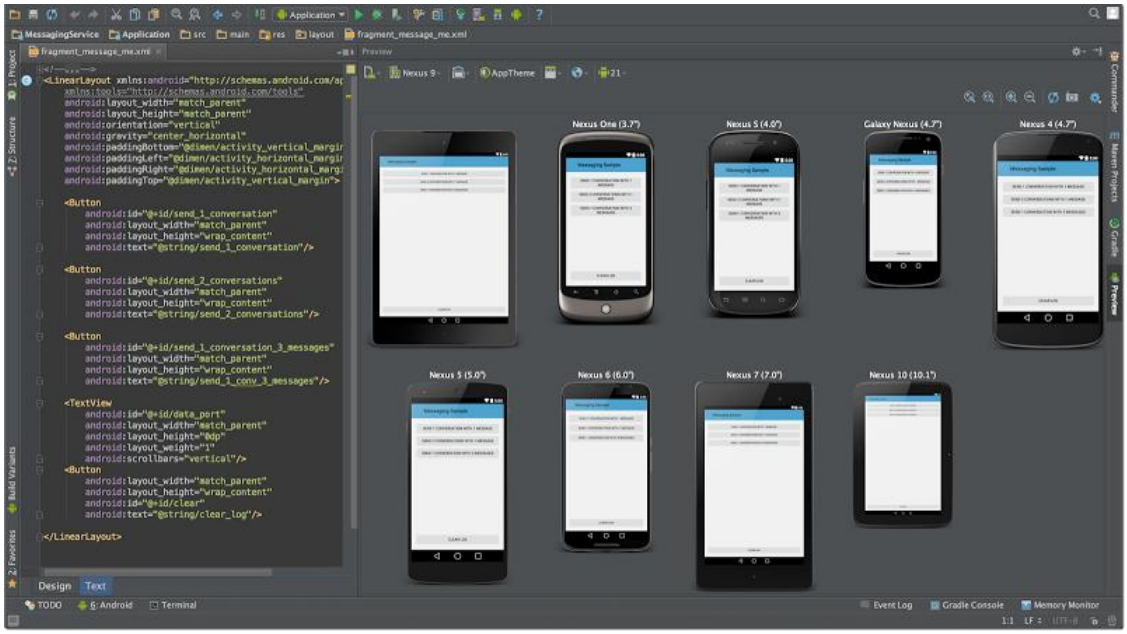


Рис 3.Окно с различными экранами девайсов

Android Studio замечательно подходит для разработки под любые классы устройств, на которых развернут «зеленый робот»: смартфоны, планшеты, смарт-часы, телевизоры, автомобили, смарт-очки.

В Android Studio встроены востребованные инструменты оттачивания качества приложений и выстраивания путей их монетизации. Присутствует раздел с подсказками и советами по оптимизации с тематическими разделами типа «разработка под планшеты». В консоли находится помощник, рекомендующий услуги профессиональных переводчиков — Google в данном случае выступает посредником. Есть способы запуска кампании по продвижению приложения через отслеживание эффективности работы рекламных объявлений: вы узнаете, откуда пришли установившие программу пользователи и что сделали после того, как впервые ее запустили. При желании можно наглядно вывести хронологию доходности приложения.

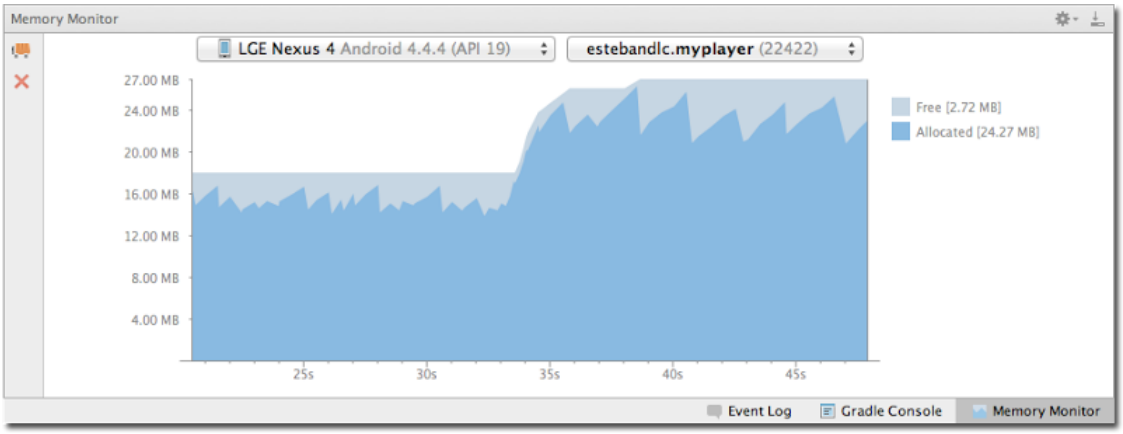


Рис 4. Хронология

Android Studio обогащена расширенной поддержкой шаблонов Google Service, прозрачной интеграцией с GitHub, плюс взаимодействием со службами Google Cloud Platform, такими как Cloud Messaging и App Engine.

Среда разработки приложений Google Android Studio доступна для Windows, OS X и Linux.

2.Проектирование мобильного приложенияBasilic

2.1. Требования к мобильному приложениюBasilic

Общие требования

Приложение должно обеспечивать выполнение следующих основных операций:

- Ввод данных через экранную клавиатуру девайса и их корректировку
- Обработка введенных данных
- Обработка поступающих данных со стороны сервера
- Вывод данных на дисплей девайса

Исходные данные: числовые и текстовые, вводимые через диалоговые окна или взятые из локальной базы данных (БД) приложения.

Выходные данные: результирующий контент приложения, полученные со стороны сервера и сохраненный в локальной БД приложения, выводимый на дисплее девайса.

Надежность

Для корректной работы приложения необходимо реализовать следующие:

- Обработка и контроль входных и выходных данных
- Вывод сообщений об ошибках

Условия эксплуатации

При работе с приложением следует соблюдать условия эксплуатации мобильных устройств, работающих на базе операционной системы Android.

Технические требования

Для функционирования приложения необходимо наличие следующих технических средств:

- Девайс, работающий на базе операционной системы Android, версия которой 4.0 и выше
- Доступ в Интернет

2.2. Стадии разработки ПО (программного обеспечения)

Формирование Технического задания

Разработка Технического задания (ТЗ), как правило, проводится при плотном взаимодействии с Заказчиком. В результате этого этапа (разработка ТЗ) создается документ «Техническое задание». «Техническое задание» может быть достаточно объемным и должно отражать всю функциональность будущего программного приложения или системы. Но, тем не менее, предусмотреть всю функциональность в рамках Технического задания (ТЗ) бывает достаточно сложно. Этот факт обязательно учитывается и предусматривается возможность внесения изменений в Техническое задание на последующих этапах разработки.

Разработка базы данных

Большинство разработок не обходится без создания базы данных (БД). Современные платформы баз данных (MS SQL, ORACLE, DB2)

предназначены не только для хранения информации и выполнения примитивных операций SELECT, UPDATE, DELETE, INSERT, но и способны задавать логику обработки информации. В результате, программное обеспечения клиента (АРМ – автоматизированное рабочее место) несет только функциональную нагрузку по вводу значений (параметров) и отображению данных. Этот подход снижает требования к аппаратной части клиентских ПЭВМ, а все «тяжелые» операции выполняются на сервере баз данных (БД).

В зависимости от качества спроектированной базы данных (таблицы, ключи, уникальные поля, связи, хранимые процедуры, функции, триггеры и т.д.) напрямую зависит будущее программного приложения. При разработке базы данных обязательно учитывается возможность ее вертикального и горизонтального масштабирования и, при не грамотном проектировании БД, вносить изменения в уже существующий программный продукт достаточно трудно.

Так же важно учитывать при разработке БД объемы обрабатываемых данных. При больших объемах данных и сложных логических операциях, обусловленных требованиями бизнес-процессов, выполнение транзакций может занимать неприемлемо большое время (например, при обработке входных данных полученных с использованием автоматизированных устройств таких как, сканер штрих кодов, электронная карта).

Разработка сервера приложений

Иногда целесообразна разработка сервера приложений. Сервер приложений является промежуточным звеном между конечным клиентом и базой данных (БД). Несмотря на использование современных платформ баз данных, часть логической обработки (в основном не данных, а действий оператора) **возлагается на программное приложение**. Перенос этой функциональности со стороны клиента на сервер приложений дает ряд

преимуществ, а сама технология называется «тонкий клиент». Перечислим эти преимущества:

- *внутрипрограммная логика работы заложена централизовано, а значит, при изменении логики не требуется обновления клиентских приложений;*
- *клиент действительно предназначен только для ввода параметров и отображения данных, что фактически сводит требования к аппаратной части ПЭВМ к требованиям функционирования операционной системы (ОС);*
- *серьезно повышается безопасность использования автоматизированного рабочего места (АРМ) использующего приложение (систему, комплекс задач), за счет осуществления работы с назначением определенного порта обмена данными.*
- *серьезно повышается безопасность хранения данных на сервере БД, за счет возможности размещения сервера приложений в демилитаризованной зоне информационной инфраструктуры предприятия.*

К технологии «тонкого» клиента относится и использование WEB доступа. В этом случае, в качестве сервера приложений выступает сервис IIS (Internet Information Services), но функциональность Web-браузера обычно гораздо больше, чем требуется Вашему приложению. К недостаткам можно отнести и меньшую защищенность и зависимость разметки элементов в области отображения браузера от его производителя. Неоспоримым плюсом использования WEB-доступа является возможность подключения с любого ПЭВМ имеющего выход в общедоступную сеть (Internet) и Web-браузер. Однако, эта возможность является и минусом: достаточно знать имя пользователя и пароль что бы воспользоваться доступом к Вашим данным. Для исключения этой ситуации необходимо применение дополнительных

средств защиты от несанкционированного доступа (таких как использование электронных ключей, внедрение центра сертификации и др.).

Разработка приложения (комплекса задач, системы)

Разработке клиентского приложения предшествует проектирование визуального интерфейса приложения. В основу проектирования визуальной части специалисты закладывают два принципа:

- *визуальный интерфейс должен быть интуитивно понятен пользователю;*
- *должен быть осуществлен принцип преемственности;*

Часто, перед принятием решения о ведении собственной разработки программного обеспечения организация уже использует некую автоматизированную среду (построенную с использованием MS Access, разрозненные общедоступные утилиты по различным видам учета и т.д.), а значит, у пользователя уже приобретены некоторые навыки работы с ней. С целью сокращения периода «привыкания» к работе с новой программой иногда имеет смысл использовать уже привычные визуальные формы.

К моменту написания исходных кодов уже имеется:

- *Техническое задание;*
- *Спроектированная база данных (БД);*
- *Определена необходимость разработки сервера приложений, при ее необходимости определен протокол обмена данными между клиентским приложением и сервером приложений;*
- *Разработана визуальная составляющая;*

На основании имеющегося Технического задания (ТЗ) и проекта визуальных форм закладывается логика работы в Ваш программный продукт.

Тестирование

Этот этап работы имеет не менее важное значение, чем остальные. В рамках тестирования определяется:

- *соответствие требованиям Техническому заданию и проекту визуальных форм;*
- *возможность выполнения на различных операционных системах (ОС);*
- *устойчивость программного продукта к внешнему воздействию (попытка задания некорректных условий, подмена программных модулей, удаление модулей, изменение конфигурации и др.);*
- *стабильность работы во времени и другие характеристики;*

В рамках этого этапа формируется карта тестирования, которая определяет проведение всех необходимых тестов программного продукта. По результатам тестирования формируется список замечаний и предложений разработчикам. Практически, по результатам тестирования возникает ряд вопросам по логике работы к Заказчику. Обычно, на этом этапе и выявляются вопросы, не рассмотренные в Техническом задании (ТЗ), осуществляется его корректировка.

Опытная эксплуатация

Опытная эксплуатация – последний этап перед вводом в промышленную эксплуатацию. На этом этапе Исполнитель отрабатывает замечания, полученные в рамках приемо-сдаточных испытаний, и ведет постоянный

контроль над функционированием системы. Обычно, для проведения опытной эксплуатации выбирается «пилотная» зона. Рекомендуется делать выбор пилотной зоны исходя из следующих принципов:

- *возможные ошибки в работе программного обеспечения не приведут к серьезным нарушениям в работе предприятия в целом;*
- *разнообразие программных и аппаратных средств максимально.*

Естественно, что на этапе опытной эксплуатации все выявленные недочеты в программном обеспечении будут устранены, все дополнительные пожелания пользователей будут выполнены. **После окончания опытной эксплуатации осуществляется ввод в промышленную эксплуатацию и тиражирование, теперь уже готового, Вашего программного продукта.**

2.3. Модель UML

Модель UML *(UML model) – это совокупность конечного множества конструкций языка, главные из которых – это сущности и отношения между ними.*

Сами сущности и отношения модели являются экземплярами метаклассов метамодели.

Рассматривая модель UML с наиболее общих позиций, можно сказать, что это граф (точнее, нагруженный мульти-псевдо-гипер-орграф), в котором вершины и ребра нагружены дополнительной информацией и могут иметь сложную внутреннюю структуру. **Вершины этого графа называются сущностями, а ребра – отношениями**. Остальная часть раздела содержит беглый (предварительный), но полный обзор имеющихся типов сущностей и отношений. К счастью, их не слишком много. В последующих главах книги

все сущности и отношения рассматриваются еще раз, более детально и с примерами.

2.3.1. Сущности

Для удобства обзора сущности в UML можно подразделить на четыре группы:

- структурные;
- поведенческие;
- группирующие;
- аннотационные.

Структурные сущности, как нетрудно догадаться, предназначены для описания структуры. Обычно к структурным сущностям относят следующие.

Объект (object) 1 – сущность, обладающая уникальностью и инкапсулирующая в себе состояние и поведение.

Класс (class) 2 – описание множества объектов с общими атрибутами, определяющими состояние, и операциями, определяющими поведение.

Интерфейс (interface) 3 – именованное множество операций, определяющее набор услуг, которые могут быть запрошены потребителем и предоставлены поставщиком услуг.

Кооперация (collaboration) 4 – совокупность объектов, которые взаимодействуют для достижения некоторой цели.

Действующее лицо (actor) 5 – сущность, находящаяся вне моделируемой системы и непосредственно взаимодействующая с ней.

Компонент(component) 6 – модульная часть системы с четко определенным набором требуемых и предоставляемых интерфейсов.

Артефакт (artifact) 7 – элемент информации, который используется или порождается в процессе разработки программного обеспечения. Другими словами, артефакт – это физическая единица реализации, получаемая из элемента модели (например, класса или компонента).

Узел (node) 8 – вычислительный ресурс, на котором размещаются и при необходимости выполняются артефакты.

На следующем рисунке приведена стандартная нотация в минимальном варианте для структурных сущностей.

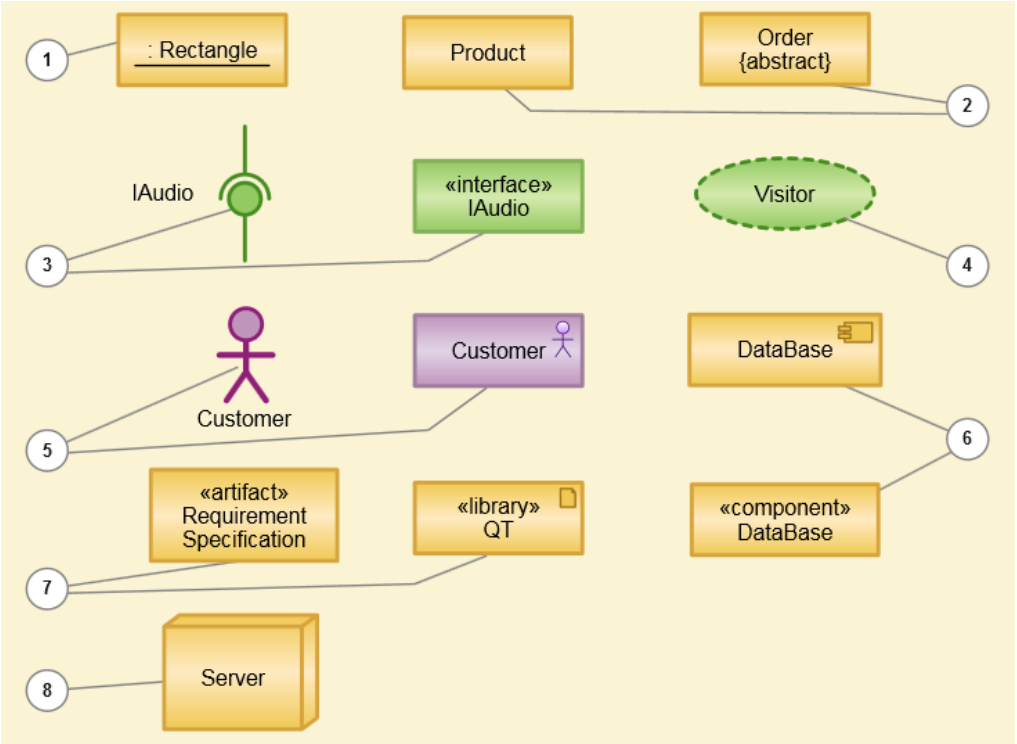


Рис 5. Нотация структурных сущностей

Поведенческие сущности предназначены для описания поведения. Основных поведенческих сущностей всего две: состояние и действие (точнее, две с половиной, потому что иногда употребляется еще и деятельность, которую можно рассматривать как особый случай состояния).

Состояние (state) 1 – период в жизненном цикле объекта, находясь в котором объект удовлетворяет некоторому условию и осуществляет собственную деятельность или ожидает наступления некоторого события.

Деятельность (activity) 2 можно считать частным случаем состояния, который характеризуется продолжительными (по времени) не атомарными вычислениями.

Действие (action) 3 – примитивное атомарное вычисление.

Это только надводная часть айсберга поведенческих сущностей: состояния бывают самые разные. Кроме того, при моделировании поведения используется еще ряд вспомогательных сущностей, которые здесь не перечислены, потому что сосуществуют только вместе с указанными основными.

Несколько особняком стоит сущность – вариант использования.

Вариант использования (use case) 4 – множество сценариев, объединенных по некоторому критерию и описывающих последовательности производимых системой действий, доставляющих значимый для некоторого действующего лица результат.

Ниже приведена стандартная нотация в минимальном варианте для поведенческих сущностей.

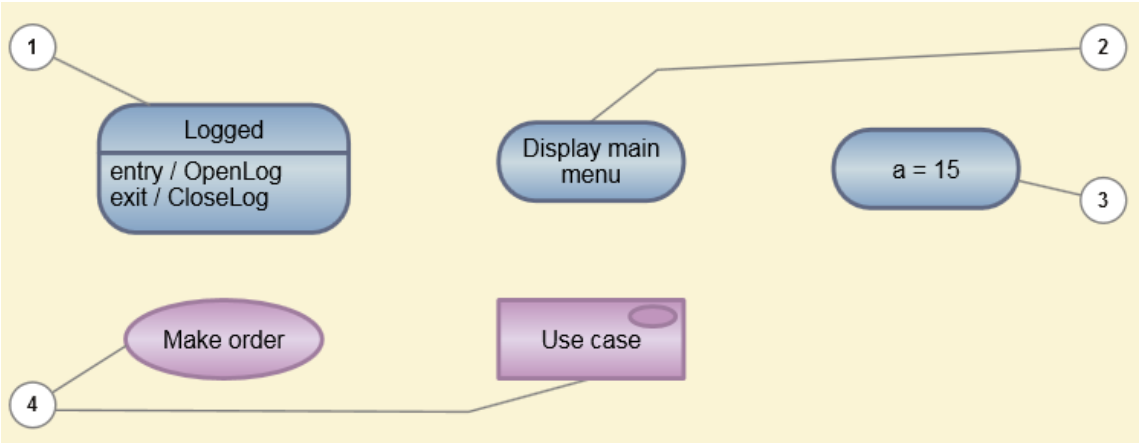


Рис 6. Нотация поведенческих сущностей

Группирующая сущность в UML одна – пакет – зато универсальная.

Пакет (package) 1 – группа элементов модели (в том числе пакетов).

Аннотационная сущность тоже одна – комментарий.

Комментарий (comment) 2 – произвольное по формату и содержанию описание одного или нескольких элементов модели.

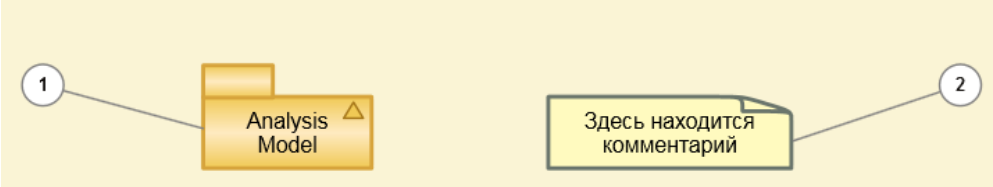


Рис 7. Нотация группирующей и аннотационной сущностей

Приведенная классификация не является исчерпывающей. У каждой из этих сущностей есть различные частные случаи и вариации.

2.3.2. Отношения

В UML используются четыре основных типа отношений:

- зависимость (dependency);
- ассоциация (association);
- обобщение (generalization);
- реализация (realization).

Зависимость – это наиболее общий тип отношения между двумя сущностями.

Отношение зависимости указывает на то, что изменение независимой сущности каким-то образом влияет на зависимую сущность.

Графически отношение зависимости изображается в виде пунктирной линии со стрелкой 1, направленной от зависимой сущности 2 к независимой 3, как показано на следующем рисунке. Как правило, семантика конкретной зависимости уточняется в модели с помощью дополнительной информации. Например, зависимость со стереотипом «use» означает, что зависимая сущность использует (скажем, вызывает операцию) независимую сущность.

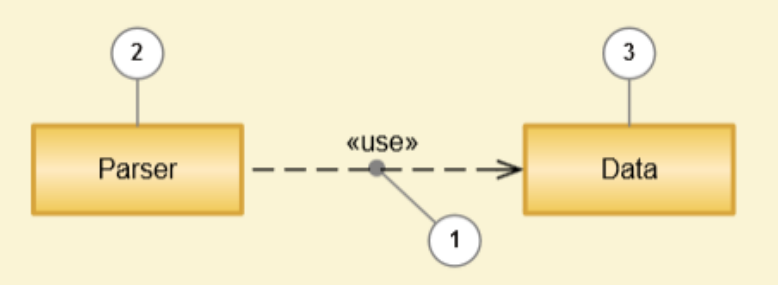


Рис 8. Отношение зависимости

Ассоциация – это наиболее часто используемый тип отношения между сущностями.

Отношение ассоциации имеет место, если одна сущность непосредственно связана с другой (или с другими – ассоциация может быть не только бинарной).

Графически ассоциация изображается в виде сплошной линии 1 с различными дополнениями, соединяющей связанные сущности, как показано на следующем рисунке. На программном уровне непосредственная связь может быть реализована различным образом, главное, что ассоциированные сущности знают друг о друге. Например, отношение часть-целое является частным случаем ассоциации и называется отношением агрегации.

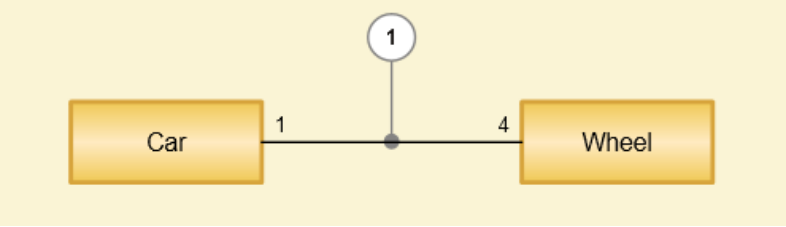


Рис 9. Отношение ассоциации

Обобщение – это отношение между двумя сущностями, одна их которых является частным (специализированным) случаем другой.

Графически обобщение изображается в виде линии с треугольной не закрашенной стрелкой на конце 1, направленной от частного 2 (подкласса) к общему 3 (суперклассу), как показано на следующем рисунке.

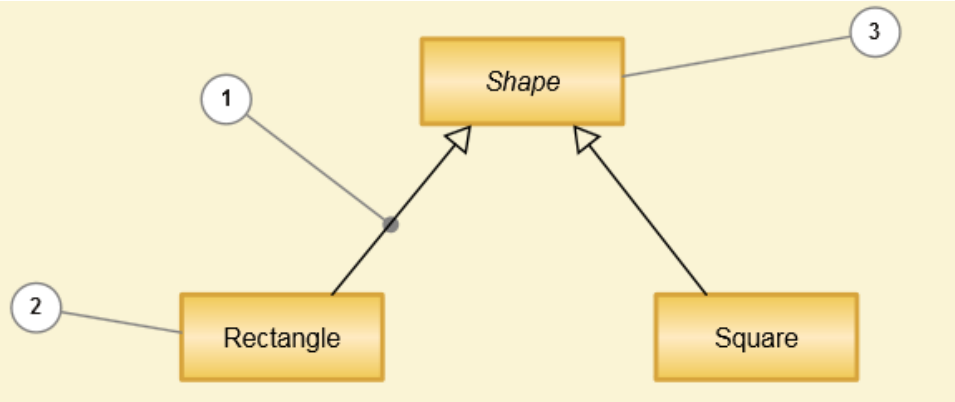


Рис 10. Отношение обобщения

Отношение реализации используется несколько реже, чем предыдущие три типа отношений, поскольку часто подразумеваются по умолчанию.

Отношение реализации указывает, что одна сущность является реализацией другой.

Например, класс является реализацией интерфейса. Графически реализация изображается в виде пунктирной линии с треугольной не закрашенной стрелкой на конце 1, направленной от реализующей сущности 2 к реализуемой 3, как показано на следующем рисунке.

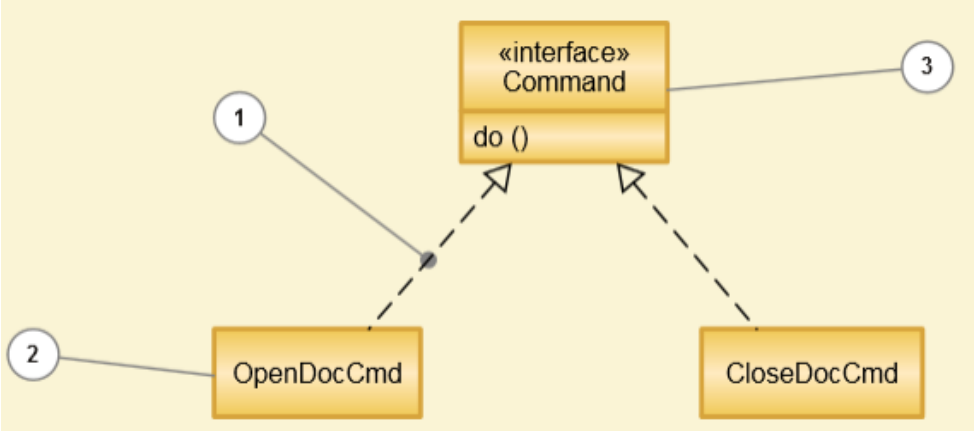


Рис 11. Отношение реализации

2.3.3. Диаграмма использования

Диаграмма использования (use case diagram) – это наиболее общее представление функционального назначения системы.

Диаграмма использования призвана ответить на главный вопрос моделирования: что делает система во внешнем мире?

На диаграмме использования применяются два типа основных сущностей: варианты использования 1 и действующие лица 2, между которыми устанавливаются следующие основные типы отношений:

- ассоциация между действующим лицом и вариантом использования 3;
- обобщение между действующими лицами 4;
- обобщение между вариантами использования 5;
- зависимости (различных типов) между вариантами использования 6.

На диаграмме использования, как и на любой другой, могут присутствовать комментарии 7. Более того, это настоятельно рекомендуется делать для улучшения читаемости диаграмм.

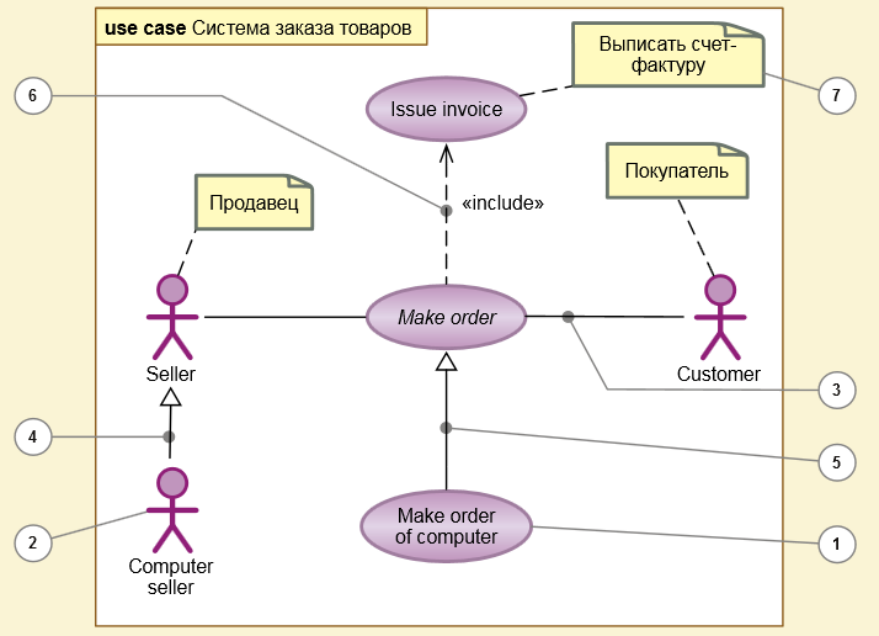
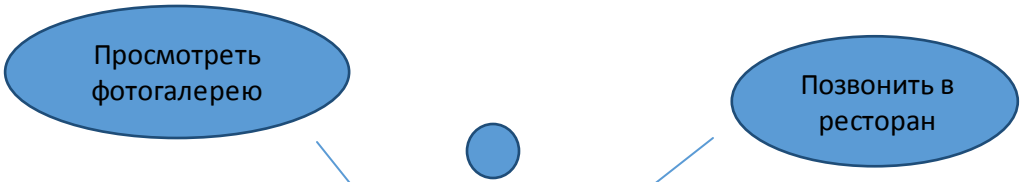
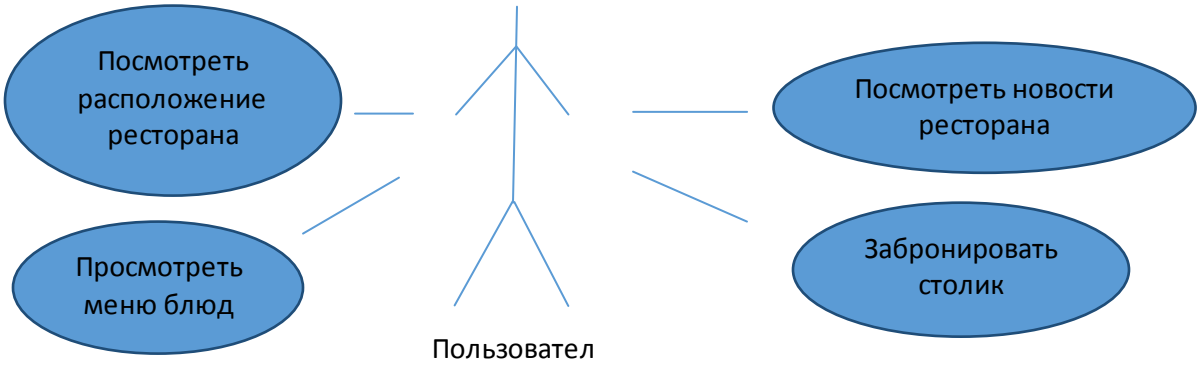


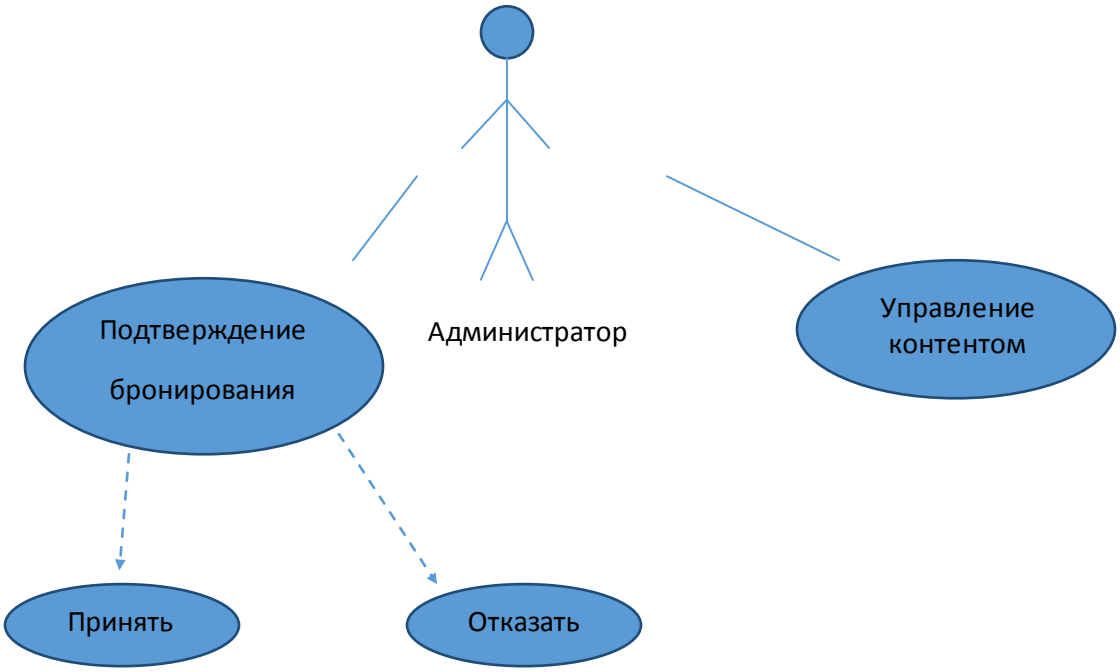
Рис 12. Нотация диаграммы использования

Диаграммы использования приложения «Базилик»

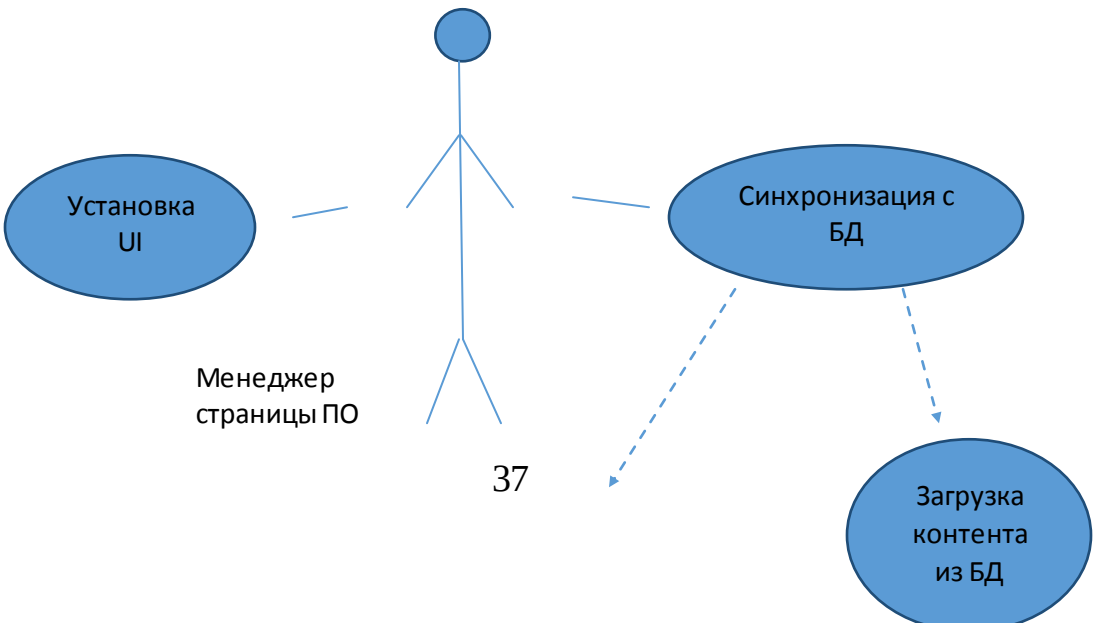


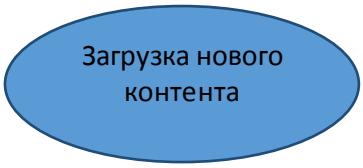


UsecaseСистема пользования приложением



UsecaseСистема администрирования





UsecaseСистема установки UI

2.3.4. Диаграмма классов

Диаграмма классов (class diagram) – основной способ описания структуры системы.

Это не удивительно, поскольку UML в первую очередь объектно-ориентированный язык, и классы являются основным (если не единственным) "строительным материалом".

На диаграмме классов применяется один основной тип сущностей: классы 1 (включая многочисленные частные случаи классов: интерфейсы, примитивные типы, классы-ассоциации и многие другие), между которыми устанавливаются следующие основные типы отношений:

- ассоциация между классами 2 (с множеством дополнительных подробностей);
- обобщение между классами 3;
- зависимости (различных типов) между классами 4 и между классами и интерфейсами.

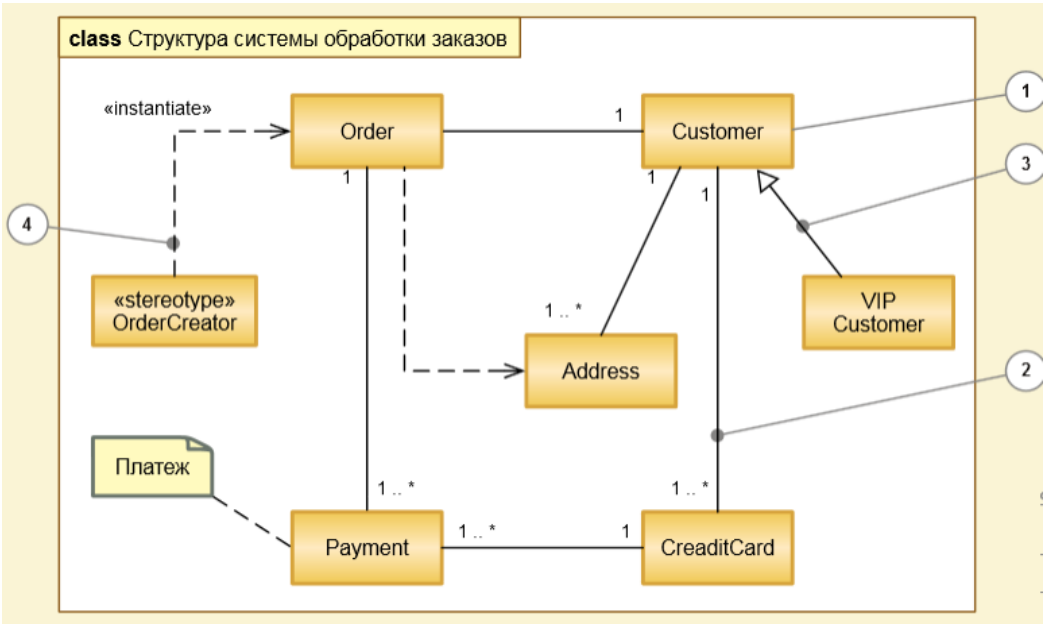
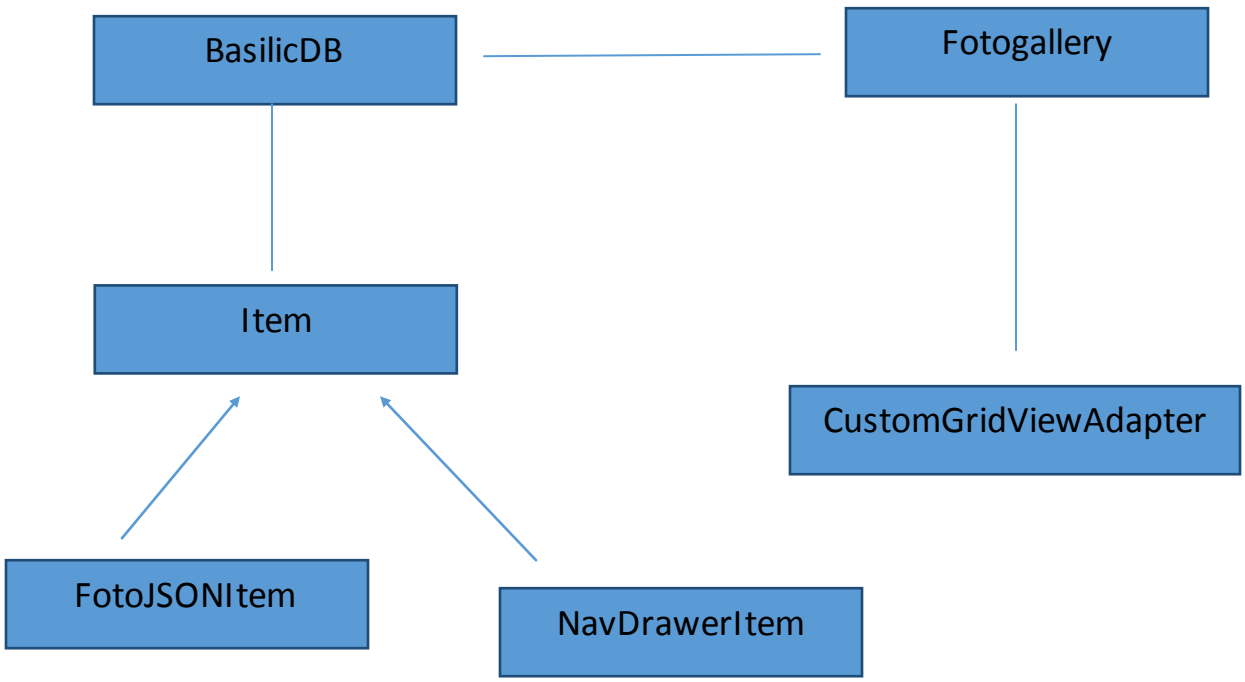


Рис 13. Нотация диаграммы классов

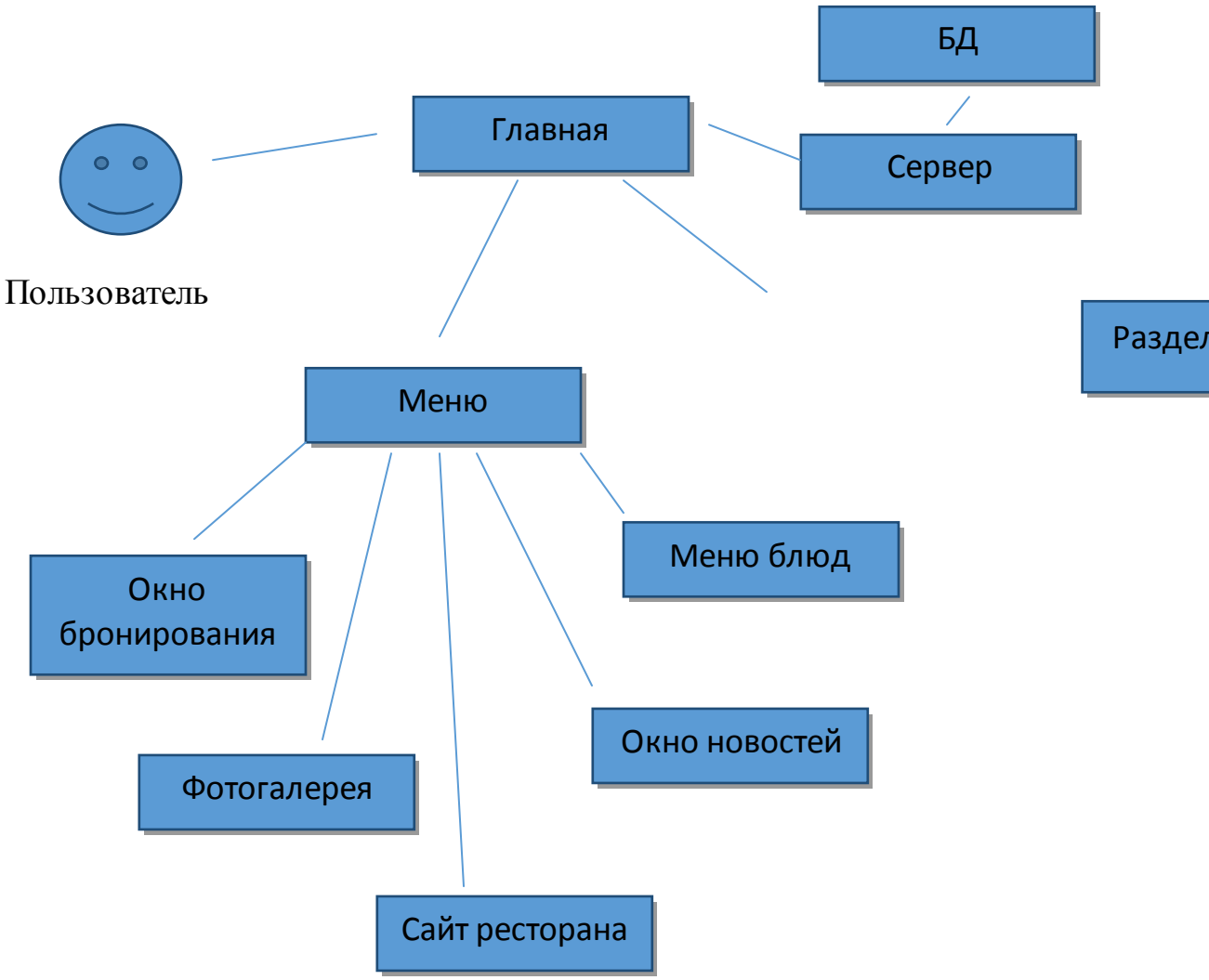
Диаграммы классов приложения «Базилик»



Реализация классов из диаграммы есть в приложение

3.Описание мобильного приложения Basilic

3.1. Архитектура приложения



Обобщенная система приложения «Базилик», представлена в виде иерархической схемы.

3.2. Управление приложением

При запуске данного мобильного приложения пользователь увидит главную страницу (рис 14.), на которой он может перейти в меню приложения (рис 15.), или позвонить в ресторан (рис 16.).

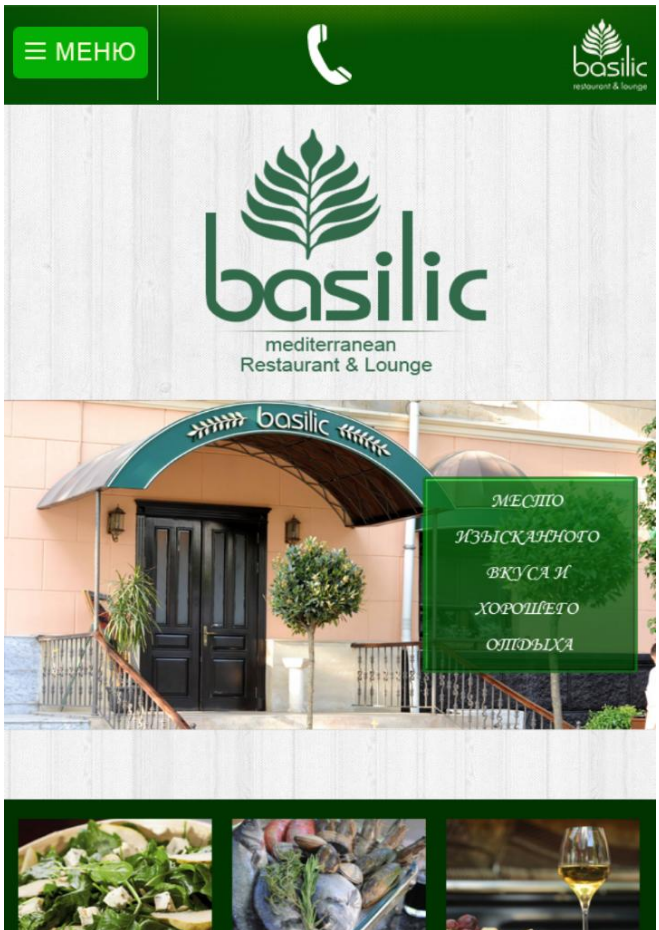


Рис 14. Главное окно

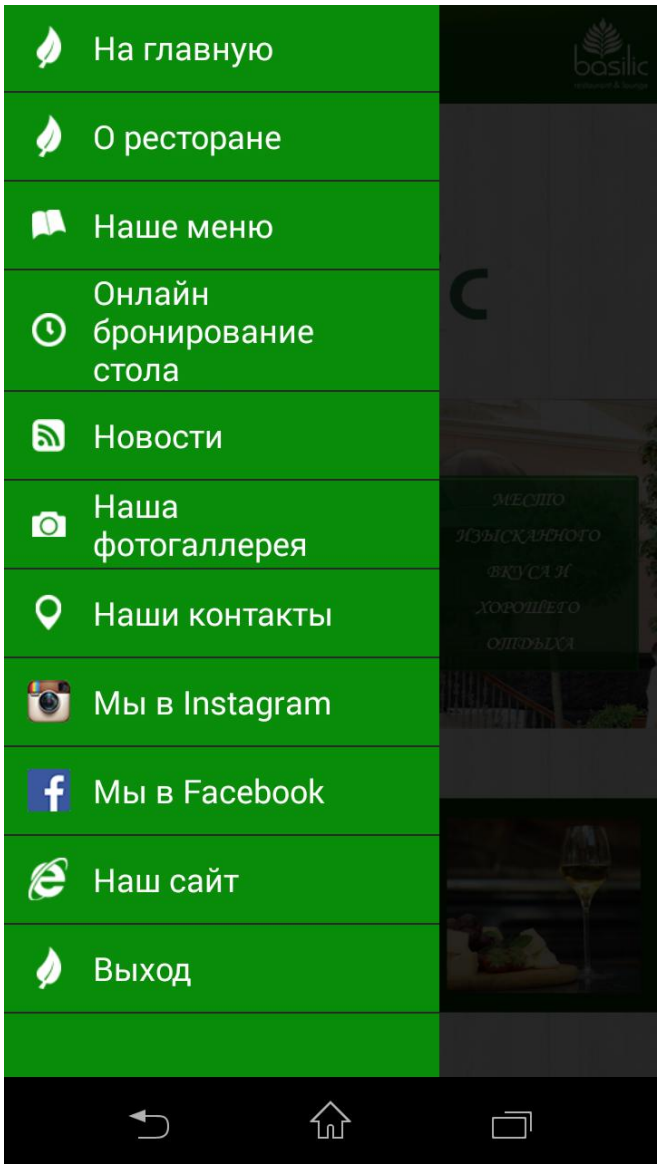


Рис 15. Меню приложения

Рис 16. Кнопка позвонить в ресторан

3.3. Окно «О ресторане»

При нажатие вкладки «О ресторане» пользователь перейдет в окно, где описан сам ресторан, история названия и многое другое (рис 17.). При наличие интернет соединения, сначала будет проделана проверка (рис 18.) нового контента страницы, а далее его загрузка либо из базы данных либо от сервера и сохранение в базе данных приложения на устройстве.

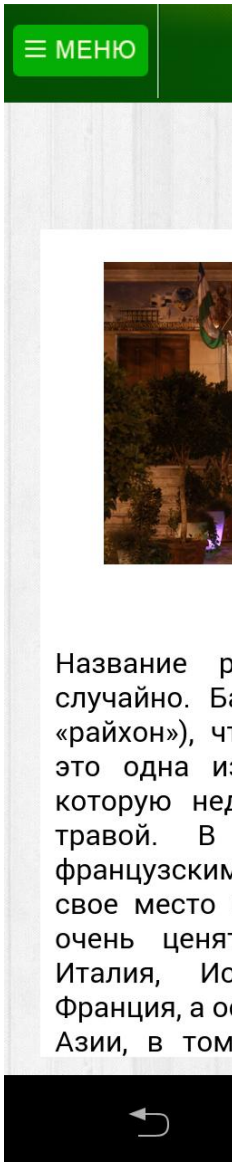


Рис 17. Окно О ресторане

Рис 18. Загрузка страницы

Страницы «Новости», «Фотогалерея» работает аналогично, только отображают свой контент.

3.4. Окно «Меню блюд»

При переходе в меню блюд пользователь попадет на страницу, где может выбрать категорию блюд (рис 19.). При переходе по категориям, пользователь соответственно может просмотреть меню блюд ресторана (рис 20. и рис 21.), включая напитки и блюд от шеф повара (рис 22.).

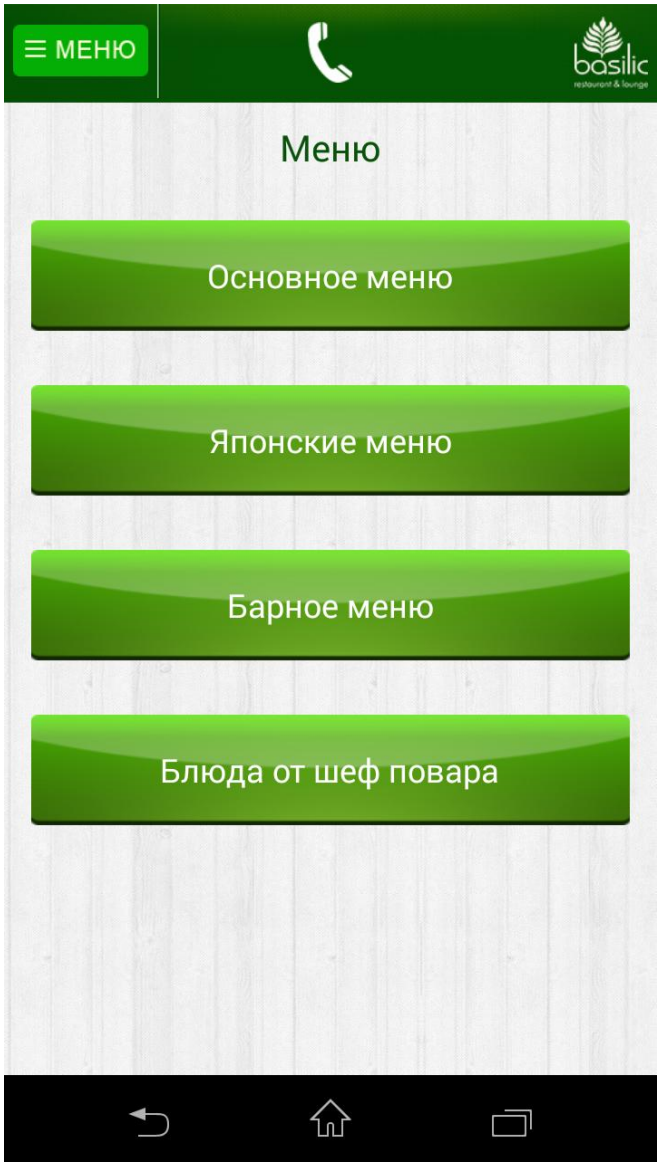


Рис 19. Меню ресторана

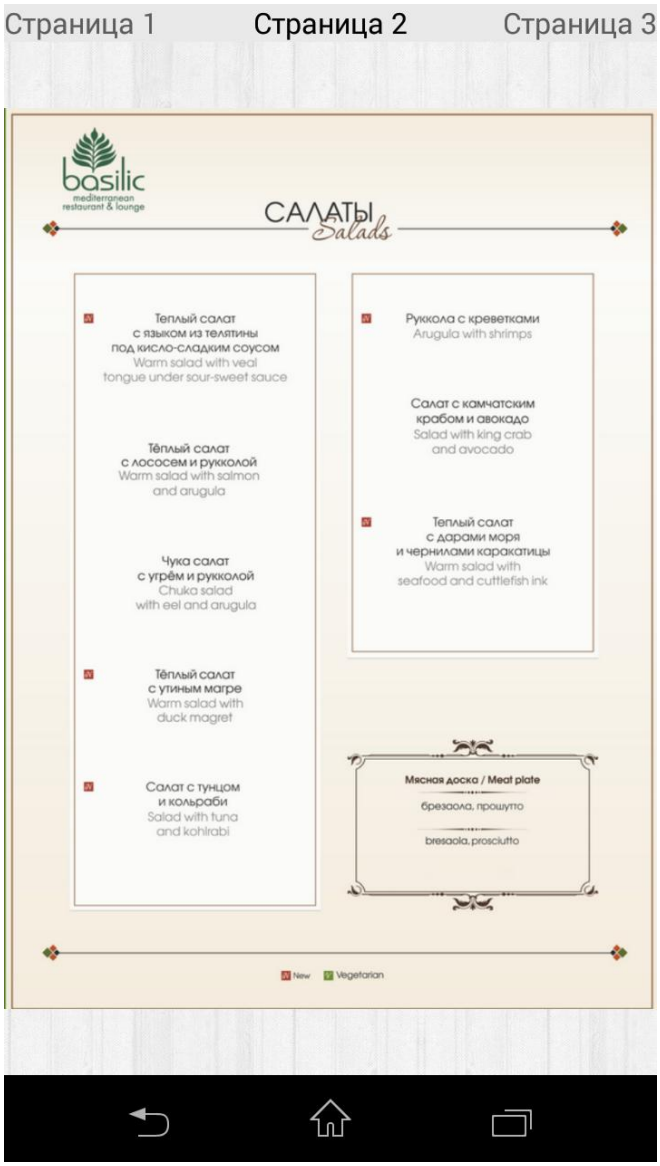


Рис 20. Меню ресторана

С помощью движения двумя пальцами по экрану девайса можно увеличить страницу.

Рис 21.Меню ресторана

На рис 21. изображена страница с японскими блюдами.

Рис 22. Меню ресторана

На рис 22. изображена страница с блюдом от шеф повара.

3.5. Окно «Бронирования»

При нажатие на вкладке меню бронирование, пользователь перейдетна страницу выбора комнат бронирования (рис 23.).

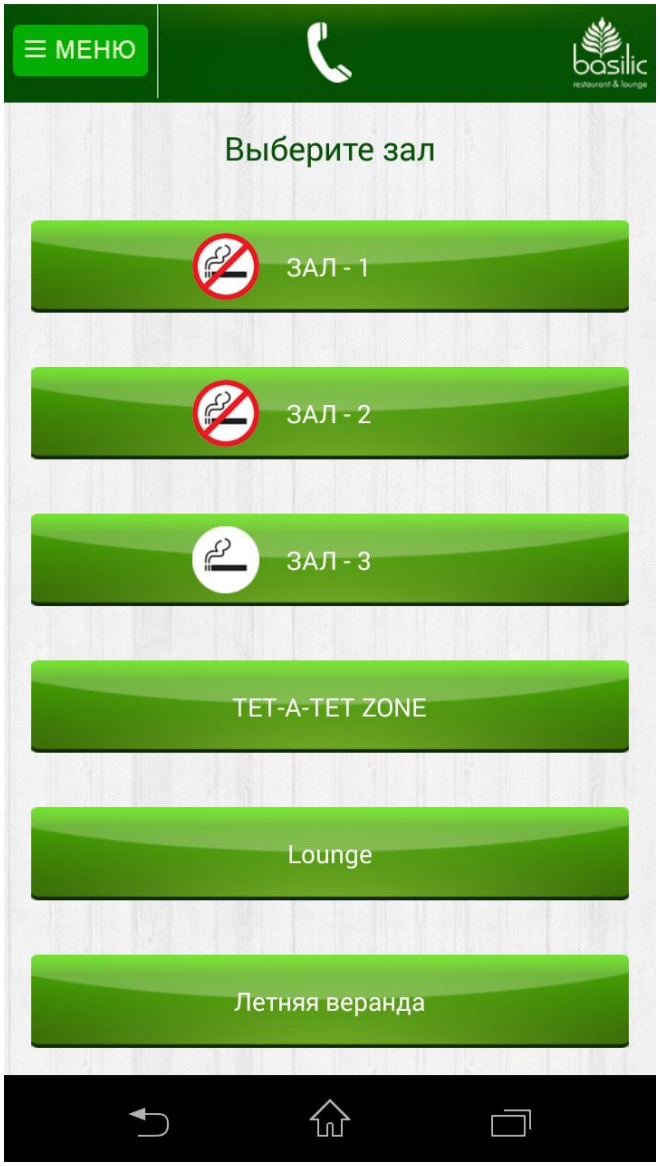


Рис 23. Окно бронирования

При переходе в комнату пользователь может выбрать столик, который хочет забронировать (рис 24.).

Рис 24. Зал ресторана

Нажав на столик он будет подсвечен и можно будет нажить на кнопку забронировать столик онлайн (рис 25.).

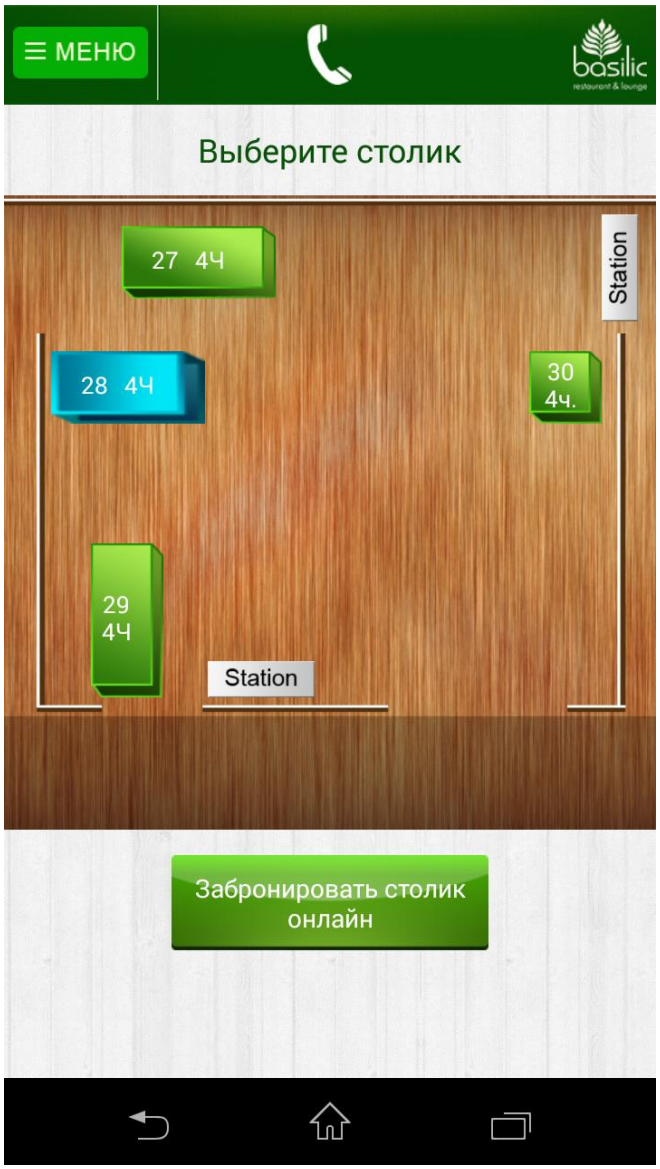


Рис 25. Выбор столика

Далее выйдет диалоговое окно, куда нужно будет вести всю информацию для того чтобы забронировать столик, а после нужно будет подтвердить ваш выбор (рис 26. и рис 27.).

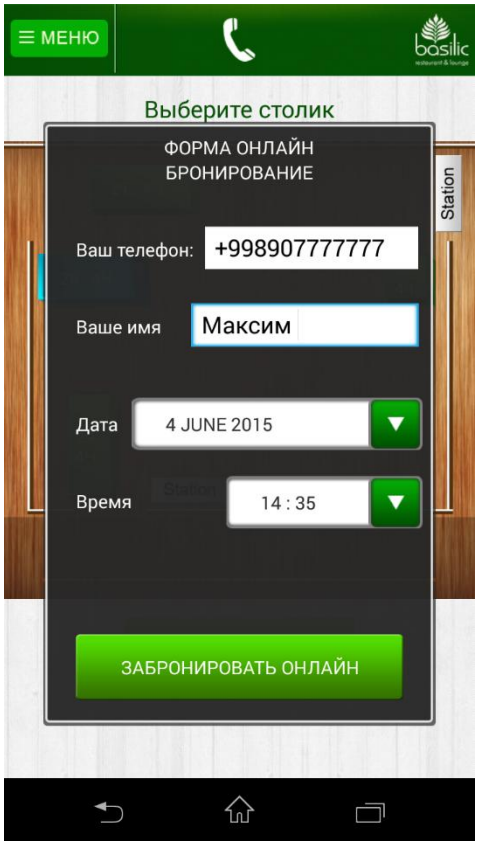


Рис 26. Бронирование

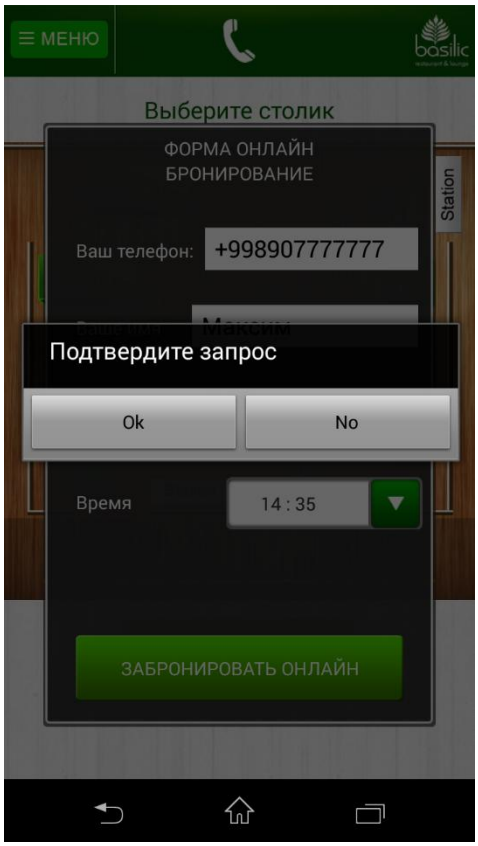


Рис 27. Бронирование

Также хочется отметить, что помимо выше описанных основных функций приложения, пользователь может посмотреть контакты с рестораном (рис 28.), перейти на страницу ресторана в социальных сетях и на сайт ресторана.

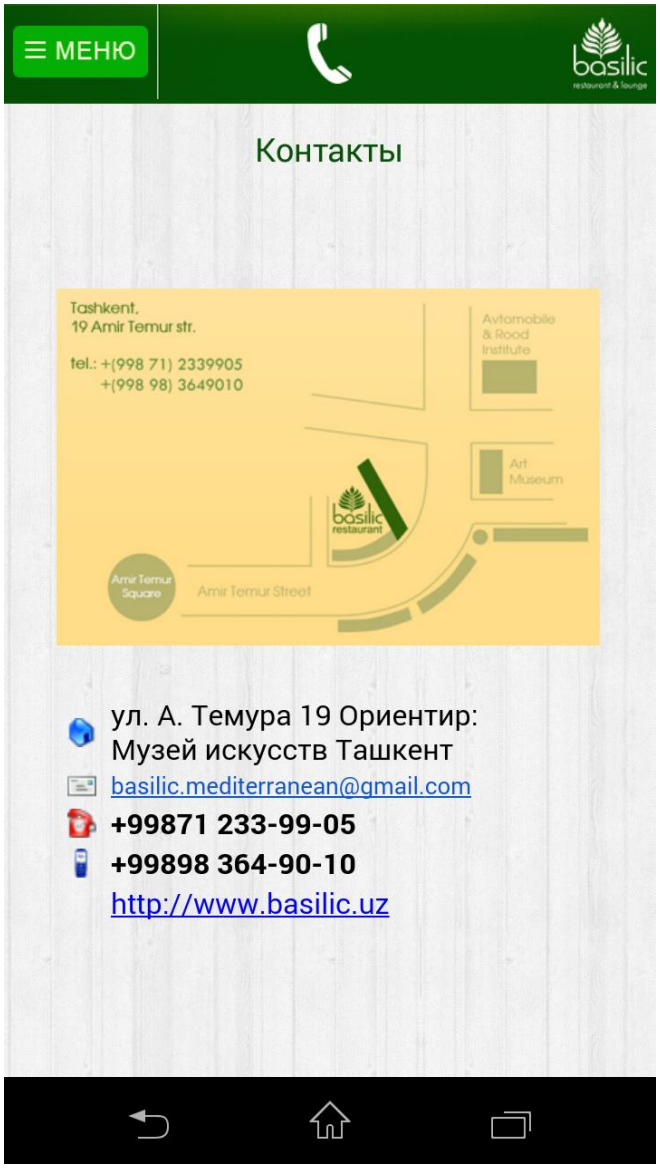


Рис 28. Контакты ресторана

4. Безопасность жизнедеятельности

4.1. Электромагнитные излучения (ЭМИ)

4.1.1. Источники ЭМИ высоких, ультра- и сверхвысоких частот.

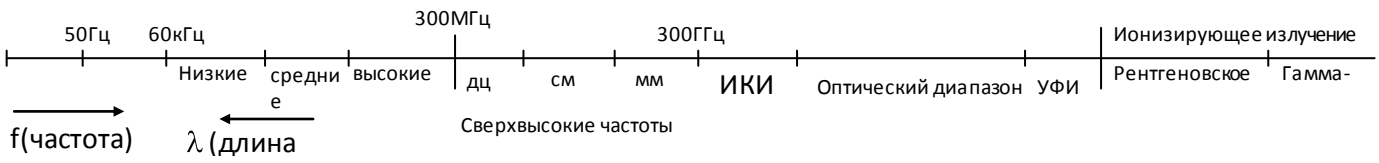


Схема 4.1. Шкала частот

Электромагнитными излучениями пронизано все окружающее пространство. Человек является источником ЭМИ слабой интенсивности. В природе существуют естественные источники ЭМИ.

Природные источники ЭМ полей: 1) атмосферное электричество; 2) радио излучение Солнца и галактик (реликтовое излучение, равномерно распространенное во Вселенной); 3) Электрическое и магнитное поля Земли (грозы - испускание низких ЭМИ).

Проблема вредного воздействия ЭМИ на человека возникла во 2 половине XX века в связи с возросшей ролью техногенных источников ЭМИ. Техногенные источники ЭМИ:

1) на производстве:

- устройства для индукционной и диэлектрической обработки различных материалов (печи, плавильни);
- источники для ионизации газов, поддержания разряда при сварке, получения плазмы;
- устройства для сварки и прессования синтетических материалов;
- линии электропередач, особенно высоковольтные;
- распределительные устройства; е) измерительные устройства и

т.д.;

2) в быту: проводка;

3) радиостанции, ТВ станции, блоки передатчиков, антенные системы и т.д.

4.1.2. Характеристики (параметры) ЭМИ.

Основными характеристиками электромагнитного излучения принято считать частоту, длину волны и поляризацию.

Длина волны прямо связана с частотой через (групповую) скорость распространения излучения. Групповая скорость распространения электромагнитного излучения в вакууме равна скорости света, в других средах эта скорость меньше. Фазовая скорость электромагнитного излучения в вакууме также равна скорости света, в различных средах она может быть как меньше, так и больше скорости света. В большинстве случаев (обычно) скорость — и групповая, и фазовая — распространения электромагнитного излучения в веществе отличается от таковых, В вакууме очень незначительно.

Описанием свойств и параметров электромагнитного излучения в целом занимается электродинамика, хотя свойствами излучения отдельных областей спектра занимаются определенные более специализированные разделы физики (отчасти так сложилось исторически, отчасти обусловлено существенной конкретной спецификой, особенно в отношении взаимодействия излучения разных диапазонов с веществом, отчасти также спецификой прикладных задач). К таким более специализированным разделам относятся оптика (и ее разделы) и радиофизика. Жестким электромагнитным излучением коротковолнового конца спектра занимается физика высоких энергий; в соответствии с современными представлениями (Стандартная модель) при высоких энергиях электродинамика перестает

быть самостоятельной, объединяясь в одной теории со слабыми взаимодействиями, а затем — при еще более высоких энергиях — как ожидается — со всеми остальными калибровочными полями.

Существуют различающиеся в деталях и степени общности теории, позволяющие смоделировать и исследовать свойства и проявления электромагнитного излучения. Наиболее фундаментальной из завершенных и проверенных теорий такого рода является квантовая электродинамика, из которой путём тех или иных упрощений можно в принципе получить все перечисленные ниже теории, имеющие широкое применение в своих областях. Для описания относительно низкочастотного электромагнитного излучения в макроскопической области используют, как правило, классическую электродинамику, основанную на уравнениях Максвелла, причём существуют упрощения в прикладных применениях. Для оптического излучения (вплоть до рентгеновского диапазона) применяют оптику (в частности, волновую оптику, когда размеры некоторых частей оптической системы близки к длинам волн; квантовую оптику, когда существенны процессы поглощения, излучения и рассеяния фотонов; геометрическую оптику — предельный случай волновой оптики, когда длиной волны излучения можно пренебречь). Гамма-излучение чаще всего является предметом ядерной физики, с других — медицинских и биологических — позиций изучается воздействие электромагнитного излучения в радиологии . Существует также ряд областей — фундаментальных и прикладных — таких, как астрофизика, фотохимия, биология фотосинтеза и зрительного восприятия, ряд областей спектрального анализа, для которых электромагнитное излучение (чаще всего — определенного диапазона) и его взаимодействие с веществом играют ключевую роль. Все эти области граничат и даже пересекаются с описанными выше разделами физики.

Некоторые особенности электромагнитных волн с точки зрения теории колебаний и понятий электродинамики:

- наличие трёх взаимно перпендикулярных (в вакууме) векторов: волнового вектора, вектора напряжённости электрического поля E и вектора напряжённости магнитного поля H.
- электромагнитные волны — это поперечные волны, в которых вектора напряжённостей электрического и магнитного полей колеблются *перпендикулярно* направлению распространения волны, но они существенно отличаются от волн на воде и от звука тем, что их можно передать от источника к приёмнику в том числе и через вакуум.

4.1.3. Воздействие ЭМИ на человека.

Зависит от факторов:

1) частота колебаний;

2) значения напряженности электрического и магнитного полей (до 300 МГц) и плотности потока энергии (СВч, ИКИ и тд) - речь о силе воздействия;

3) размеры облучаемой поверхности тела;

4) индивидуальные особенности организма;

5) комбинированные действия с другими факторами среды.

Воздействие ЭМИ 2-х видов:

1) тепловое

2) специфическое.

Тепловое воздействие (механизм) - в электрическом поле молекулы и атомы поляризуются, а полярные молекулы (вода) ориентируются по направлению ЭМ поля; в электролитах возникают ионные токи, тоесть нагрев тканей. Электролиты составляют основной процент от веса человека.

Диэлектрики: сухожилия, хрящи, кости - возможен нагрев за счет поляризации. Чем больше напряженность поля, тем сильнее нагрев. До определенного порога избыточная теплота отводится от тканей за счет

механизма терморегуляции. Начиная с этой величины - возможность организма отводить тепло исчерпывается и начинается нагрев. Слабая терморегуляция (где много жидкости, но слабо развита кровеносная система): хрусталик глаза, глаз, мозг (ткань головного мозга), печень, почки и т.д.

Специфическое воздействие ЭМ полей сказывается при интенсивностях, значительно меньших теплового порога. ЭМ поля изменяют ориентацию белковых молекул, тем самым, ослабляя их биохимическую активность. В результате наблюдается изменение структуры клеток крови, изменения в эндокринной системе, а также ряд трофических заболеваний (нарушение питания тканей: ломкость ногтей, волос и т.д.), нарушение ЦНС, сердечно - сосудистой системы; при низких дозах есть опасность воздействия на иммунитет.

4.1.4. Нормирование ЭМИ.

Осуществляется в зависимости от диапазона частот. При нормировании учитывается:

1) диапазон частот;

2) значения напряженности электрического и магнитного полей и энергетическая нагрузка.

Если в течение рабочего времени человек подвергается воздействию ЭМИ не должна превышать 1 мВт/кв.см. По офиц. данным неблагоприятные воздействия ЭМ поля проявляются при напряженностях магнитного поля, начиная с 160 - 200 Ампер/метр. Токи промежуточных частот не превышают 25 А/м. В зависимости от времени нахождения человека в поле промежуточной частоты, устанавливается предельное значение напряженности электрического поля (8 часов - не > 5 кВ)

4.1.5. Защита от ЭМИ.

Способы защиты:

1) уменьшение мощности источника - уменьшение параметров излучения в самом источнике (защита количеством) - основные поглотители - графит, резина и т.д.;

2) экранирование источника излучения (рабочего места);

3) выделение зоны излучения (зонирование территории);

4) Установление рациональных режимов эксплуатации установок;

5) применение сигнализации;

6) Защита расстоянием (особенно эффективна для СВч) формула

7) Защита временем (от тока пром. частоты)

8) Средства индивидуальной защиты (спец. костюмы).

4.1.6. Диапазоны электромагнитного излучения

Электромагнитное излучение принято делить по частотным диапазонам (см. таблицу). Между диапазонами нет резких переходов, они иногда перекрываются, а границы между ними условны. Поскольку скорость распространения излучения (в вакууме) постоянна, то частота его колебаний жёстко связана с длиной волны в вакууме.

Название диапазона		**Длины волн, λ**	**Частоты, ν**	**Источники**
Радиоволны	Сверхдлинные	более10 км	менее 30 кГц	Атмосферные явления. Переменные токи в проводниках и электронных потоках (колебательные контуры).
	Длинные	10 км — 1 км	30 кГц — 300 кГц	
	Средние	1 км — 100 м	300 кГц — 3	

			МГц	
	Короткие	100 м — 10 м	3 МГц — 30 МГц	
	Ультракороткие	10 м — 1 мм	30 МГц — 300 ГГц[	
Инфракрасное излучение		1 мм — 780 нм	300 ГГц — 429 ТГц	Излучение молекул и атомов при тепловых и электрических воздействиях.
Видимое (оптическое) излучение		780— 380 нм	429 ТГц — 750 ТГц	
Ультрафиолетовое		380 — 10 нм	$7{,}5\times10^{14}$ Гц — 3×10^{16} Гц	Излучение атомов под воздействием ускоренных электронов.
Рентгеновские		10 нм — 5 пм	3×10^{16} — 6×10^{19} Гц	Атомные процессы при воздействии ускоренных заряженных частиц.
Гамма		менее 5 пм	более 6×10^{19} Гц	Ядерные и космические процессы, радиоактивный распад.

4.2. Пожарная безопасность.

Возникновение пожара.

Пожар - это горение вне специального очага, которое не контролируется и может привести к массовому поражению и гибели людей, а также к нанесению экологического, материального и другого вреда.

Горение - это химическая реакция окисления, сопровождающаяся выделением теплоты и света. Для возникновения горения требуется наличие

трех факторов: горючего вещества, окислителя и источника загорания. Окислителями могут быть кислород, хлор, фтор, бром, йод, окиси азота и другие. Кроме того, необходимо чтобы горючее вещество было нагрето до определенной температуры и находилось в определенном количественном соотношении с окислителем, а источник загорания имел определенную энергию.

Наибольшая скорость горения наблюдается в чистом кислороде. При уменьшении содержания кислорода в воздухе горение прекращается. Горение при достаточной и надмерной концентрации окислителя называется полным, а при его нехватке - неполным.

Выделяют три основных вида самоускорения химической реакции при горении: тепловой, цепной и цепочно-тепловой. Тепловой механизм связан с экзотермичностью процесса окисления и возрастанием скорости химической реакции с повышением температуры. Цепное ускорение реакции связано с катализом превращений, которое осуществляют промежуточные продукты превращений. Реальные процессы горения осуществляются, как правило, по комбинированному (цепочно-тепловой) механизму.

Процесс возникновения горения подразделяется на несколько видов.

Вспышка - быстрое сгорание горючей смеси, не сопровождающееся образованием сжатых газов.

Возгорание - возникновение горения под воздействием источника зажигания.

Воспламенение - возгорание, сопровождающееся появлением пламени.

Самовозгорание - явление резкого увеличения скорости экзотермических реакций, приводящее к возникновению горения вещества при отсутствии источника зажигания.

Самовоспламенение - самовозгорание, сопровождается появлением пламени.

Взрыв - чрезвычайно быстрое (взрывчатое) превращение, сопровождающееся выделением энергии с образованием сжатых газов.

Основными показателями пожарной опасности являются температура самовоспламенения и концентрационные пределы воспламенения.

Температура самовоспламенения характеризует минимальную температуру вещества, при которой происходит резкое увеличение скорости экзотермических реакций, заканчивающееся возникновением пламенного горения.

Температура вспышки - самая низкая (в условиях специальных испытаний) температура горючего вещества, при которой над поверхностью образуются пары и газы, способные вспыхивать в воздухе от источника зажигания, но скорость их образования еще недостаточна для последующего горения.

Горючими называются вещества, способные самостоятельно гореть после изъятия источника загорания.

По степени горючести вещества делятся на: горючие (сгораемые), трудногорючие (трудносгораемые) и негорючие (несгораемые).

К трудногорючим относятся такие вещества, которые не способны распространять пламя и горят лишь в месте воздействия источника зажигания.

Негорючими являются вещества, не воспламеняющиеся даже при воздействии достаточно мощных источников зажигания (импульсов).

Горючие вещества могут быть в трех агрегатных состояниях: жидком, твердом и газообразном. Большинство горючих веществ независимо от агрегатного состояния при нагревании образует газообразные продукты, которые при смешении с воздухом, содержащим определенное количество кислорода, образуют горючую среду. Горючая среда может образоваться при тонкодисперсном распылении твердых и жидких веществ.

Из горючих газов и пыли образуются горючие смеси при любой температуре, в то время как твердые вещества и жидкости могут образовать горючие смеси только при определенных температурах.

В производственных условиях может иметь место образование смесей горючих газов или паров в любых количественных соотношениях. Однако взрывоопасными эти смеси могут быть только тогда, когда концентрация горючего газа или пара находится между границами воспламеняемых концентраций.

Минимальная концентрация горючих газов и паров в воздухе, при которой они способны загораться и распространять пламя, называющееся *нижним концентрационным пределом воспламенения* .

Максимальная концентрация горючих газов и паров, при которой еще возможно распространение пламени, называется *верхним концентрационным пределом воспламенения.*

Указанные пределы зависят от температуры газов и паров: при увеличении температуры на 100°С величины нижних пределов воспламенения уменьшаются на 8 -10 %, верхних - увеличиваются на 12 - 15 %.

Пожарная опасность вещества тем больше, чем ниже нижний и выше верхний пределы воспламенения и чем ниже температура самовоспламенения.

Пыли горючих и некоторых не горючих веществ (например алюминий, цинк) могут в смеси с воздухом образовать горючие концентрации.

Наибольшую опасность по взрыву представляет взвешенная в воздухе пыль. Однако и осевшая на конструкциях пыль представляет опасность не только с точки зрения возникновения пожара, но и вторичного взрыва, вызываемого в результате взвихривания пыли при первичном взрыве.

Минимальная концентрация пыли в воздухе, при которой происходит ее загорание, называется *нижним пределом воспламенения пыли* .

Поскольку достижение очень больших концентраций пыли во взвешенном состоянии практически нереально, термин "верхний предел воспламенения" к пылям не применяется.

Воспламенение жидкости может произойти только в том случае, если над ее поверхностью имеется смесь паров с воздухом в определенном количественном соотношении, соответствующим нижнему температурному пределу воспламенения.

Меры по пожарной профилактике.

Мероприятия по пожарной профилактике разделяются на организационные, технические, режимные и эксплуатационные.

Организационные мероприятия: предусматривают правильную эксплуатацию машин и внутризаводского транспорта, правильное содержание зданий, территории, противопожарный инструктаж и тому подобное.

Технические мероприятия: соблюдение противопожарных правил и норм при проектировании зданий, при устройстве электропроводов и оборудования, отопления, вентиляции, освещения, правильное размещение оборудования.

Режимные мероприятия - запрещение курения в неустановленных местах, запрещение сварочных и других огневых работ в пожароопасных помещениях и тому подобное.

Эксплуатационные мероприятия – своевременная профилактика, осмотры, ремонты и практика тушения пожаров наибольшее распространение получили следующие принципы прекращения горения:

1) изоляция очага горения от воздуха или снижение концентрации кислорода путем разбавления воздуха негорючими газами (углеводы CO i<12 - 14 %).

2) охлаждение очага горения ниже определенных температур;

3) интенсивное торможение (ингибирование) скорости химической реакции в пламени;

4) механический срыв пламени струей газа или воды;

5)создание условий огнепреграждения (условий, когда пламя распространяется через узкие каналы).

4.2.1. Техносфера. Ресурсы техносферы.

Объем и состав техносферы.

Мировое хозяйство можно рассматривать как видовую реализованную экологическую нишу человечества. По многим пространственным и потоковым параметрам она совпадает с биосферой, экологическая емкость которой ограниченна. Поэтому неизбежны конкурентные отношения между активными элементами техногенной среды и биосферы, между общественным производством и планетарной биотой. Хотя эти отношения намного сложнее, чем межвидовые взаимоотношения в природе, многие их черты выглядят как конкурентное вытеснение биосферы.

Техносфера - это глобальная совокупность орудий, объектов,

материальных процессов и продуктов общественного производства. Техносферу можно определить также как пространство геосфер Земли, находящееся под воздействием производственной деятельности человека и занятое ее продуктами.

Ресурсы техносферы

Понятие о природных ресурсах.

Это солнечная энергия, свет, пища, вода, тепло, почва, т.е. все то, что необходимо для жизни на Земле. В данном разделе природные ресурсы будут рассмотрены с позиций использования их в общественном производстве.

Природные ресурсы являются основной частью экономических ресурсов, т.е. кроме факторов среды они являются факторами производства.

Ресурсы - это вещества, материалы, силы и потоки вещества, энергии и информации, которые:

- образуют входные звенья природных или хозяйственных циклов, являются их необходимыми участниками и, в связи с этим, носителями функции полезности;
- имеют измеряемое количественное выражение: массу, объем, плотность, концентрацию, интенсивность, мощность, стоимость;
- при изменениях во времени подчиняются фундаментальным законам сохранения.

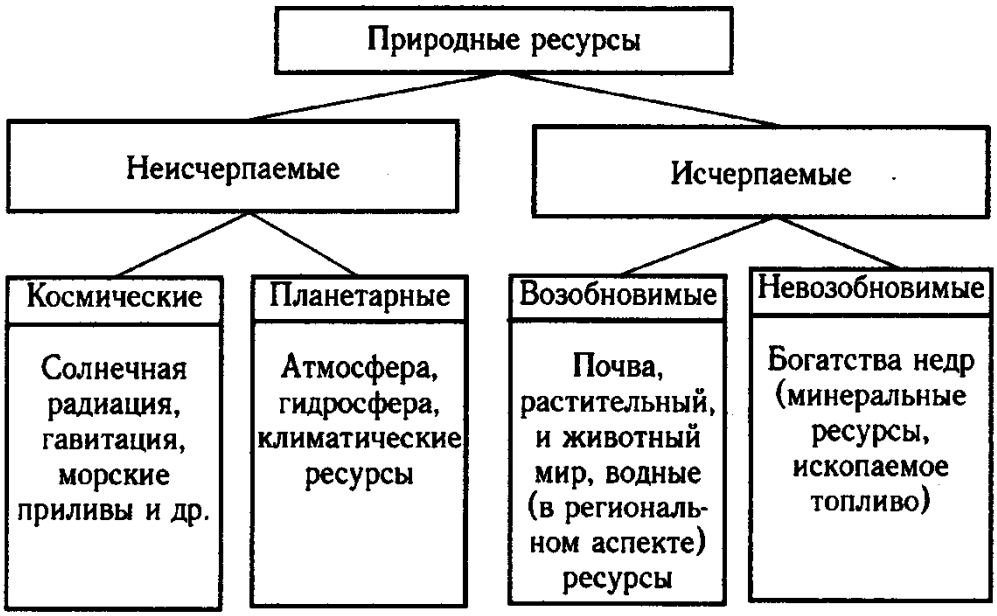


Рис.4.2. Схема классификации природных ресурсов

Все естественные материальные и энергетические ресурсы, используемые человеком, принято называть ***природными ресурсами***. При этом часто забывают, что большинство из них является ресурсами не только для человека, но в основном и в первую очередь ресурсами живой природы.

Классификация ресурсов. Существует несколько классификаций природных ресурсов.

Естественная классификация основана на разделении ресурсов по компонентам природной среды: земельные, минеральные, водные, климатические, растительные, животного мира и т.п.

В хозяйственной классификации ведущее значение имеет отраслевая принадлежность: ресурсы топливно-энергетического комплекса, металлургии, химической промышленности, сельского хозяйства, лесоперерабатывающей промышленности и т.д.

С эколого-экономической точки зрения важна классификация природных ресурсов по признакам исчерпаемости. К практически

неисчерпаемым (в пределах времени существования техносферы) часто относят космические (солнечную радиацию, гравитацию) и планетарные ресурсы (наличие атмосферы, гидросферы, геотермальной энергии). Однако в конкретных земных и, тем более, техносферных условияхдействует закон ограниченности (исчерпаемости) всех природных ресурсов.

Возобновимые ресурсы - это вещества и силы, которые создаются на Земле благодаря текущему потоку солнечной энергии: тепло, атмосферная влага, вода осадков и всех пресных вод, течение рек и гидроэнергия, энергия ветров, волн и течений, почва, все живые организмы, биосфера, наконец, сам человек. Для различных возобновимых, особенно для биологических ресурсов, существуют пределы скорости изъятия и степени исчерпания, после превышения которых уже невозможно возобновление, так как нарушается его естественный режим. Чаще всего это относится к численности популяции или биоразнообразию экосистем. Но это может быть отнесено и к биосфере в целом.

Разумеется, исчерпаемы и все невозобновимые ресурсы. К ним относится подавляющее большинство полезных ископаемых: горные материалы, руды, минералы, осадочные породы, ископаемое топливо. Правда, некоторые минеральные ресурсы и сейчас медленно образуются при геохимических процессах в недрах, глубинах океана или на поверхности земной коры - залежи солей, руды переходных металлов, железомарганцевые конкреции, известняки, продукты выветривания, но не уголь и углеводороды. В отношении полезных ископаемых большое значение имеют доступность и качество ресурса, а также количественное соотношение между оцененными потенциальными, реальными разведанными и эксплуатационными запасами.

Принципиальное отличие техносферы от биосферы заключается в том, что биосфера использует исключительно контролируемые ею возобновимые ресурсы, тогда как человек в техносфере, кроме захвата значительной части биосферных ресурсов, использует и огромную массу невозобновимых

ресурсов, значительная часть которых не нужна биоте биосферы, но влияет на ее функционирование.

Несмотря на указанное отличие ресурсы биосферы и техносферы непрерывно взаимодействуют между собой. Преждевременное изъятие погребенных в литосфере веществ и ввод их в оборот нарушает оптимальный баланс круговорота веществ в природе. Кроме того, использование невозобновимых ресурсов всегда влечет за собой цепь частных последствий, важных для биосферы: преобразование ландшафтов, изъятие площадей природных экосистем, деградацию почв, изменение распределения грунтовых вод и др.

Хотя человечество на протяжении всей своей истории сталкивается с ограниченностью природных ресурсов, оно до сих пор не осознало последствий их бесконтрольного использования. В настоящее время экономика мирового хозяйства чрезвычайно природоемка, что и обусловливает техногенный тип развития и истощение природных ресурсов.

Заключение

В ходе выполнения данной выпускной квалификационной работы, были получены следующие знания:

- Изучена операционная система Андройд
- Изучена среда разработки приложений под данную операционную систему
- Изучены методы взаимодействия данной операционной системы с сетью Интернет
- Подняты знания языка программирования JAVA
- Изучены методы создания приложений под девайсы, работающие на базе операционной системы Андройд

Созданное мобильное приложение является весьма удобным способом узнать новости ресторана, просмотреть меню блюд и забронировать нужный столик в ресторане. Приложение не имеет возрастных ограничений, обладает дружелюбным интерфейсом, что позволяет быстро привыкнуть пользователю к приложению.

Сегодня мобильные устройства получили широкое распространения, программы, которые создаются под них практически не отличаются от программ созданных для персонального компьютера.

Архитектура приложения была спроектирована так, что имеет возможность дальнейшего улучшения и добавления дополнений к приложению. Что в свою очередь может придать ещё больше удобств пользователям приложения.

Список используемой литературы

1. Постановление Президента Республики Узбекистан «О мерах по дальнейшему внедрению и развитию современных информационно-коммуникационных технологий» (Собрание законодательства Республики Узбекистан, 2012 г., № 13, ст. 139; 2013 г., № 44, ст. 578, № 45, ст. 584)
2. Моделирование на UML. Ф.Новиков, Д.Иванов.
3. Безопасность жизнедеятельности. Белов С.В., Ильницкая А.В., Козьяков А.Ф. и др. 2012, 616с.
4. Безопасность жизнедеятельности. Конспект лекций. Алексеев В.С., Жидкова О.И., Ткаченко Н.В. 2008, 160с.
5. Теоретические основы безопасности жизнедеятельности. Айзман Р.И. и др. 2011, 208с.
6. Закон Республики Узбекистан «О ПОЖАРНОЙ БЕЗОПАСНОСТИ» (Собрание законодательства Республики Узбекистан, 2009 г., № 40, ст. 432)
7. Курс лекций по дисциплине «Метрология, стандартизация и сертификация» Полетаева Н.А., Ташкент, 2010 год
8. http://rozumsoft.ru/mobilnie_prilozheniya
9. http://newmediaedu.ru/technology/25/?&tpwf_mode=main
10. http://habrahabr.ru/post/168843
11. https://ru.wikipedia.org/wiki/Android
12. http://techcrunch.com/2013/04/18/larry-page-says-google-glass-runs-on-android/
13. http://www.android.com/tv/
14. http://www.android.com/auto/
15. http://www.sotovik.ru/news/237888-google-android-studio-pervij-stabilnij-reliz.html
16. http://www.m-te.ru/index.php?vib=4
17. http://book.uml3.ru/sec_1_4
18. http://book.uml3.ru/sec_1_5

Приложение

```
packagecom.basilic.fotogallery;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.concurrent.ExecutionException;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.HttpStatus;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.json.JSONException;

import com.basilic.about.About;
import com.basilic.about.AboutRes;
import com.basilic.db.BasilicDB;
import com.basilic.db.tables.BasilicDBItemWithFoto;
import com.basilic.fragments.NavDrawerItem;
import com.basilic.fragments.NavDrawerListAdapter;
import com.basilic.main.MainActivity;
import com.basilic.uz.R;
import com.basilic.main.menu.MainMenu;
import com.basilic.menu.MenuList;
import com.basilic.net.HttpAsyncTask;
import com.basilic.net.JsonUtil;
import com.basilic.news.News;
import com.basilic.reserved.HallList;

import android.annotation.SuppressLint;
import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.content.res.TypedArray;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.support.v4.widget.DrawerLayout;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
```

```
import android.widget.GridView;
import android.widget.ImageButton;
import android.widget.ListView;

public class Fotogallery extends Activity implements OnClickListener {

	GridView gridView;
	public static ArrayList<Item> gridArray = new ArrayList<Item>();
	CustomGridViewAdapter customGridAdapter;
	private ImageButton menu;
	private ProgressDialog dialog;

	Handler handler = new Handler();
	static BasilicDB basilicDB;
	static int index = 0;

	private DrawerLayout mDrawerLayout;
	public static ListView mDrawerList;

	// slide menu items
	private String[] navMenuTitles;
	private TypedArray navMenuIcons;

	private ArrayList<NavDrawerItem> navDrawerItems;
	private NavDrawerListAdapter adapter;

	@SuppressLint("SimpleDateFormat")
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		setContentView(R.layout.fotogallery_new);
		init();

		basilicDB = new BasilicDB(this);

		if (basilicDB.getFotoItem().isEmpty()) {
			if(isOnline()){
				startFotoSyncDB();
			}
		} else {
			ArrayList<BasilicDBItemWithFoto> tableFotoGallery = new
ArrayList<BasilicDBItemWithFoto>();
			if (basilicDB.getFotoItem() != null) {
				tableFotoGallery = basilicDB.getFotoItem();
				for (int k = 0; k < tableFotoGallery.size(); k++) {
					gridArray.add(new
Item(Uri.parse(tableFotoGallery.get(k)
							.getUrl_small_foto()),
tableFotoGallery.get(k)
							.getTitle(),
tableFotoGallery.get(k).getM_date()));
				}

			}

			customGridAdapter = new CustomGridViewAdapter(Fotogallery.this,
					R.layout.row_grid, gridArray);
			gridView.setAdapter(customGridAdapter);
		}
```

```
	}

	public void call(View view) {
		Intent intent = new Intent(Intent.ACTION_CALL);
		intent.setData(Uri.parse("tel:+998712339905"));
		startActivity(intent);
	}

	private void startFotoSyncDB() {
		dialog = new ProgressDialog(Fotogallery.this);
		dialog.setMessage("Загрузка фото...");
		dialog.show();
		dialog.setCancelable(false);
		dialog.setCanceledOnTouchOutside(false);

		new Thread(new Runnable() {

			@Override
			public void run() {

				String myJSON = "";
				ArrayList<FotoJSONItem> arrayList = null;
				boolean check;
				int time = 0;
				int cur_time = 0;

				try {
					myJSON = new HttpAsyncTask().execute(

	"http://basilic.md.uz/api/photo.php").get();
				} catch (InterruptedException e1) {
					// TODO Auto-generated catch block
					e1.printStackTrace();
				} catch (ExecutionException e1) {
					// TODO Auto-generated catch block
					e1.printStackTrace();
				}

				try {
					arrayList = JsonUtil.getFoto(myJSON);
				} catch (InterruptedException e1) {
					// TODO Auto-generated catch block
					e1.printStackTrace();
				} catch (ExecutionException e1) {
					// TODO Auto-generated catch block
					e1.printStackTrace();
				} catch (JSONException e1) {
					// TODO Auto-generated catch block
					e1.printStackTrace();
				}

				try {

					ArrayList<BasilicDBItemWithFoto> tableFotoGallery =
new ArrayList<BasilicDBItemWithFoto>();
					tableFotoGallery = basilicDB.getFotoItem();

					for (int i = 0; i < arrayList.size(); i++) {

						int index_of_table = 0;
```

```
                                        check = false;

                                        while (index_of_table <
tableFotoGallery.size()) {
                                               time =
Integer.valueOf(tableFotoGallery.get(

       index_of_table).getM_date());
                                               cur_time =
Integer.valueOf(arrayList.get(i)
                                                             .getMdate());

                                               if (time == cur_time) {
                                                      check = true;
                                               }

                                               index_of_table++;
                                        }

                                        if (!check) {
                                               Bitmap bitmap = new
ImageDownloader().execute(

       arrayList.get(i).getPhotonamesmall()).get();

                                               Bitmap big_foto = new
ImageDownloader().execute(

       arrayList.get(i).getPhotonamebig()).get();

                                               SimpleDateFormat sdf = new
SimpleDateFormat(
                                                             "yyyyMMdd_HHmmss");
                                               String currentDateandTime =
sdf.format(new Date());

                                               Uri uri_small = saveBitmap(bitmap,
                                                             "small_foto_gallery" +
currentDateandTime);
                                               Uri uri_big = saveBitmap(big_foto,
                                                             "big_foto_gallery" +
currentDateandTime);

                                               basilicDB.addFotoItem(new
BasilicDBItemWithFoto(
                                                             uri_small.getPath(),
uri_big.getPath(),
                                                             arrayList.get(i).getDate(),
arrayList

       .get(i).getPhoto_title(), arrayList

       .get(i).getPhoto_desc(), arrayList

       .get(i).getMdate()));
                                               gridArray.add(new Item(uri_small,
arrayList.get(i)
                                                             .getPhoto_title(),
arrayList.get(i)
                                                             .getMdate()));
                                        }
```

```
                                }

                                if (tableFotoGallery.size() > 0) {
                                      for (int k = 0; k < tableFotoGallery.size();
k++) {
                                            gridArray.add(new
Item(Uri.parse(tableFotoGallery

      .get(k).getUrl_small_foto()),

      tableFotoGallery.get(k).getTitle(),

      tableFotoGallery.get(k).getM_date()));
                                      }
                                }

                          } catch (InterruptedException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                          } catch (ExecutionException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                          }

                          handler.post(new Runnable() {

                                @Override
                                public void run() {
                                      dialog.dismiss();
                                      customGridAdapter = new CustomGridViewAdapter(
                                                  Fotogallery.this,
R.layout.row_grid, gridArray);
                                      gridView.setAdapter(customGridAdapter);
                                }
                          });

                    }
              }).start();
      }

      private void init() {
            menu = (ImageButton) findViewById(R.id.button_menu);
            menu.setOnClickListener(this);
            gridView = (GridView) findViewById(R.id.gallery);

            gridView.setOnItemClickListener(new OnItemClickListener() {

                  @Override
                  public void onItemClick(AdapterView<?> arg0, View arg1,
                              int position, long arg3) {

                          Intent intent = new Intent(Fotogallery.this,
FotoViewBig.class);
                          intent.putExtra("foto",
gridArray.get(position).getMdate());
                          // intent.putExtra("foto", position);
                          startActivity(intent);

                    }
              });
```

```
            mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
            mDrawerList = (ListView) findViewById(R.id.list_slidermenu);

            // load slide menu items
            navMenuTitles = getResources().getStringArray(R.array.arrya_list_menu);

            // nav drawer icons from resources
            navMenuIcons = getResources()
                        .obtainTypedArray(R.array.nav_drawer_icons);

            navDrawerItems = new ArrayList<NavDrawerItem>();

            // adding nav drawer items to array
            // Home
            navDrawerItems.add(new NavDrawerItem(navMenuTitles[0],
navMenuIcons.getResourceId(0, -1)));
            // Find People
            navDrawerItems.add(new NavDrawerItem(navMenuTitles[1],
navMenuIcons.getResourceId(1, -1)));
            // Photos
            navDrawerItems.add(new NavDrawerItem(navMenuTitles[2],
navMenuIcons.getResourceId(2, -1)));
            // Communities, Will add a counter here
            navDrawerItems.add(new NavDrawerItem(navMenuTitles[3],
navMenuIcons.getResourceId(3, -1)));
            // Pages
            navDrawerItems.add(new NavDrawerItem(navMenuTitles[4],
navMenuIcons.getResourceId(4, -1)));
            // What's hot, We  will add a counter here
            navDrawerItems.add(new NavDrawerItem(navMenuTitles[5],
navMenuIcons.getResourceId(5, -1)));
            navDrawerItems.add(new NavDrawerItem(navMenuTitles[6],
navMenuIcons.getResourceId(6, -1)));
            navDrawerItems.add(new NavDrawerItem(navMenuTitles[7],
navMenuIcons.getResourceId(7, -1)));
            navDrawerItems.add(new NavDrawerItem(navMenuTitles[8],
navMenuIcons.getResourceId(8, -1)));
            navDrawerItems.add(new NavDrawerItem(navMenuTitles[9],
navMenuIcons.getResourceId(9, -1)));
            navDrawerItems.add(new NavDrawerItem(navMenuTitles[10],
navMenuIcons.getResourceId(10, -1)));


            // Recycle the typed array
            navMenuIcons.recycle();

            mDrawerList.setOnItemClickListener(new SlideMenuClickListener());

            // setting the nav drawer list adapter
            adapter = new NavDrawerListAdapter(getApplicationContext(),
                        navDrawerItems);
            mDrawerList.setAdapter(adapter);

      }

      /**
       * Slide menu item click listener
       * */
      private class SlideMenuClickListener implements
                  ListView.OnItemClickListener {
```

```
	@Override
	public void onItemClick(AdapterView<?> parent, View view, int position,
			long id) {
		// display view for selected nav drawer item
		displayView(position);
	}
}

/**
 * Diplaying fragment view for selected nav drawer list item
 * */
private void displayView(int position) {
	// update the main content by replacing fragments
	//Fragment fragment = null;
	switch (position) {
	case 0:
		mDrawerLayout.closeDrawer(mDrawerList);
		startActivity(new Intent(this, MainActivity.class));
		overridePendingTransition(R.anim.right_in, R.anim.right_out);
		break;
	case 1:
		mDrawerLayout.closeDrawer(mDrawerList);
		startActivity(new Intent(this, AboutRes.class));
		overridePendingTransition(R.anim.right_in, R.anim.right_out);

		break;
	case 2:
		mDrawerLayout.closeDrawer(mDrawerList);
		startActivity(new Intent(this, MenuList.class));
		//finish();
		overridePendingTransition(R.anim.right_in, R.anim.right_out);
		break;
	case 3:
		mDrawerLayout.closeDrawer(mDrawerList);
		startActivity(new Intent(this, HallList.class));

		overridePendingTransition(R.anim.right_in, R.anim.right_out);
		break;
	case 4:
		mDrawerLayout.closeDrawer(mDrawerList);
		startActivity(new Intent(this, News.class));
		//finish();
		overridePendingTransition(R.anim.right_in, R.anim.right_out);
		break;

	case 5:
		mDrawerLayout.closeDrawer(mDrawerList);
		//startActivity(new Intent(this, Fotogallery.class));
		//finish();
		//overridePendingTransition(R.anim.right_in, R.anim.right_out);
		break;

	case 6:
		mDrawerLayout.closeDrawer(mDrawerList);
		startActivity(new Intent(this, About.class));
		//finish();
		overridePendingTransition(R.anim.right_in, R.anim.right_out);
		break;

	case 7:
		mDrawerLayout.closeDrawer(mDrawerList);
```

```
                Intent instagram = new Intent(Intent.ACTION_VIEW, Uri
                        .parse("http://instagram.com/basilic_rest"));
                startActivity(instagram);

                break;

            case 8:
                mDrawerLayout.closeDrawer(mDrawerList);
                Intent facebook = new Intent(
                        Intent.ACTION_VIEW,
                        Uri.parse("http://www.facebook.com/pages/basilic-
mediterranean-restaurant-lounge/791443420868025?ref=ts&fref=ts"));
                startActivity(facebook);

                break;

            case 9:
                mDrawerLayout.closeDrawer(mDrawerList);
                Intent site = new Intent(Intent.ACTION_VIEW, Uri
                        .parse("http://basilic.uz"));
                startActivity(site);
                break;

            case 10:
                mDrawerLayout.closeDrawer(mDrawerList);
                Intent exit = new Intent(getApplicationContext(),
                        MainActivity.class);
                exit.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                exit.putExtra("EXIT", true);
                startActivity(exit);

                break;

            default:
                break;
            }

        }

        @Override
        protected void onDestroy() {
            // TODO Auto-generated method stub
            super.onDestroy();
            gridArray.clear();
        }

        @Override
        public void onClick(View view) {
            switch (view.getId()) {
            case R.id.button_menu:
                mDrawerLayout.openDrawer(mDrawerList);
                break;

            default:
                break;
            }
        }

        private class ImageDownloader extends AsyncTask<String, Void, Bitmap> {

            // String filename;
```

```
		@Override
		protected Bitmap doInBackground(String... param) {
			// TODO Auto-generated method stub
			// filename = param[0];

			return downloadBitmap(param[0]);
		}

		@Override
		protected void onPreExecute() {
			Log.i("Async-Example", "onPreExecute Called");
			// simpleWaitDialog =
ProgressDialog.show(MainActivity.this,"Wait",
			// "Downloading Image");

		}

		@Override
		protected void onPostExecute(Bitmap result) {

		}
	}

	private Bitmap downloadBitmap(String url) {
		// initilize the default HTTP client object
		/*
		 * @SuppressWarnings("deprecation") HttpClient client = new
		 * DefaultHttpClient(); String username = "latin"; String password =
		 * "5Mm6WbsCZL"; String host = "sab-lab.com";
		 *
		 * ((AbstractHttpClient)
		 * client).getCredentialsProvider().setCredentials(new AuthScope(host,
		 * AuthScope.ANY_PORT), new UsernamePasswordCredentials(username,
		 * password));
		 */
		// forming a HttoGet request
		// String uri = "http://srv0.sab-lab.com/app/Android/lockscreen/" + url;
		String uri = url;
		final HttpGet getRequest = new HttpGet(uri);
		HttpClient client = new DefaultHttpClient();

		try {

			HttpResponse response = client.execute(getRequest);

			// check 200 OK for success
			final int statusCode = response.getStatusLine().getStatusCode();

			if (statusCode != HttpStatus.SC_OK) {
				Log.w("ImageDownloader", "Error " + statusCode
						+ " while retrieving bitmap from " + url);
				return null;

			}

			final HttpEntity entity = response.getEntity();
			if (entity != null) {
				InputStream inputStream = null;
				try {
					// getting contents from the stream
```

```
                                inputStream = entity.getContent();

                                BitmapFactory.Options options = new
BitmapFactory.Options();
                                // options.inSampleSize = 2;
                                Bitmap bitmap =
BitmapFactory.decodeStream(inputStream,
                                                null, options);

                                // final long fSize = entity.getContentLength();
                                // decoding stream data back into image Bitmap that
android
                                // understands
                                // final Bitmap bitmap =
                                // BitmapFactory.decodeStream(inputStream);
                                File cacheDirectory;

                                cacheDirectory =
getApplicationContext().getCacheDir();

                                if (!cacheDirectory.exists()) {
                                      cacheDirectory.mkdirs();
                                      Log.d("AKBAR", "CREATED_CACHE_FOLDER");
                                }

                                // File f = new File(cacheDirectory, url);
                                try {
                                      // FileOutputStream fout = new
FileOutputStream(f);
                                      byte[] buf = new byte[40 * 1024];
                                      @SuppressWarnings("unused")
                                      int len;

                                      while ((len = inputStream.read(buf)) > 0) {
                                            // fout.write(buf,0,len);

                                      }
                                      // fout.flush();

                                      // fout.close();

                                      inputStream.close();

                                } catch (FileNotFoundException e) {
                                      // TODO Auto-generated catch block
                                      e.printStackTrace();
                                } catch (IOException e) {
                                      // TODO Auto-generated catch block
                                      e.printStackTrace();
                                }
                                return bitmap;
                          } finally {
                                if (inputStream != null) {
                                      // inputStream.close();
                                }
                                entity.consumeContent();
                          }
                    }
              } catch (Exception e) {
                    // You Could provide a more explicit error message for
IOException
```

```
                    getRequest.abort();
                    Log.e("ImageDownloader",
                                "Something went wrong while retrieving bitmap from "
+ url
                                                + e.toString());
              }

              return null;
       }

       public static Uri saveBitmap(Bitmap bmp, String fileName) {

              File f = null;

              File directory = new File(Environment.getExternalStorageDirectory()
                           + File.separator + "Basilic");
              directory.mkdirs();
              try {
                    // imgMain.setImageBitmap(bmp);
                    f = new File(Environment.getExternalStorageDirectory()
                                .getAbsolutePath() + "/Basilic/" + fileName +
".png");
                    FileOutputStream fos = new FileOutputStream(f);
                    bmp.compress(Bitmap.CompressFormat.PNG, 90, fos);
              } catch (Exception ex) {
                    ex.printStackTrace();
              }

              Uri imageUri = Uri.fromFile(f);

              return imageUri;
       }

       @Override
       public void onBackPressed() {
              // TODO Auto-generated method stub
              super.onBackPressed();

              this.finish();
              overridePendingTransition(R.anim.left_in, R.anim.left_out);
       }

       protected boolean isOnline() {
              ConnectivityManager cm = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
              NetworkInfo netInfo = cm.getActiveNetworkInfo();
              if (netInfo != null && netInfo.isConnected()) {
                    return true;
              } else {
                    return false;
              }
       }

}

package com.basilic.fotogallery;

import android.net.Uri;

/**
 *
```

```
 * @author manish.s
 *
 */

public class Item {
	Uri image;
	String title;
	String mdate;
	private int picture;

	public Item(Uri image, String title, String mdate) {
		super();
		this.image = image;
		this.title = title;
		this.mdate = mdate;
	}

	public String getMdate(){
		return mdate;
	}

	public Uri getImage() {
		return image;
	}
	public void setImage(Uri image) {
		this.image = image;
	}
	public String getTitle() {
		return title;
	}
	public void setTitle(String title) {
		this.title = title;
	}

	public int getPicture() {
		return picture;
	}

	public void setPicture(int picture) {
		this.picture = picture;
	}


}

package com.basilic.fotogallery;

public class FotoJSONItem {

	private String photo_title;
	private String photo_desc;
	private String photonamesmall;
	private String photonamebig;
	private String date;
	private String mdate;

	public FotoJSONItem() {

	}

	public String getPhoto_title() {
```

```
		return photo_title;
	}

	public void setPhoto_title(String photo_title) {
		this.photo_title = photo_title;
	}

	public String getPhoto_desc() {
		return photo_desc;
	}

	public void setPhoto_desc(String photo_desc) {
		this.photo_desc = photo_desc;
	}

	public String getPhotonamesmall() {
		return photonamesmall;
	}

	public void setPhotonamesmall(String photonamesmall) {
		this.photonamesmall = photonamesmall;
	}

	public String getPhotonamebig() {
		return photonamebig;
	}

	public void setPhotonamebig(String photonamebig) {
		this.photonamebig = photonamebig;
	}

	public String getDate() {
		return date;
	}

	public void setDate(String date) {
		this.date = date;
	}

	public String getMdate() {
		return mdate;
	}

	public void setMdate(String mdate) {
		this.mdate = mdate;
	}

}

package com.basilic.fragments;

public class NavDrawerItem {

	private String title;
	private int icon;
	private String count = "0";
	// boolean to set visiblity of the counter
	private boolean isCounterVisible = false;

	public NavDrawerItem(){}
```

```
	public NavDrawerItem(String title, int icon){
		this.title = title;
		this.icon = icon;
	}

	public NavDrawerItem(String title, int icon, boolean isCounterVisible, String
count){
		this.title = title;
		this.icon = icon;
		this.isCounterVisible = isCounterVisible;
		this.count = count;
	}

	public String getTitle(){
		return this.title;
	}

	public int getIcon(){
		return this.icon;
	}

	public String getCount(){
		return this.count;
	}

	public boolean getCounterVisibility(){
		return this.isCounterVisible;
	}

	public void setTitle(String title){
		this.title = title;
	}

	public void setIcon(int icon){
		this.icon = icon;
	}

	public void setCount(String count){
		this.count = count;
	}

	public void setCounterVisibility(boolean isCounterVisible){
		this.isCounterVisible = isCounterVisible;
	}
}

package com.basilic.fotogallery;

import java.util.ArrayList;
import com.basilic.uz.R;
import android.app.Activity;
import android.content.Context;
import android.text.Html;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;

/**
```

```
 *
 * @author manish.s
 *
 */
public class CustomGridViewAdapter extends ArrayAdapter<Item> {
	Context context;
	int layoutResourceId;
	ArrayList<Item> data = new ArrayList<Item>();

	public CustomGridViewAdapter(Context context, int layoutResourceId,
			ArrayList<Item> data) {
		super(context, layoutResourceId, data);
		this.layoutResourceId = layoutResourceId;
		this.context = context;
		this.data = data;
	}

	@Override
	public View getView(int position, View convertView, ViewGroup parent) {
		View row = convertView;
		RecordHolder holder = null;

		if (row == null) {
			LayoutInflater inflater = ((Activity)
context).getLayoutInflater();
			row = inflater.inflate(layoutResourceId, parent, false);

			holder = new RecordHolder();
			holder.txtTitle = (TextView) row.findViewById(R.id.item_text);
			holder.imageItem = (ImageView) row.findViewById(R.id.item_image);
			row.setTag(holder);
		} else {
			holder = (RecordHolder) row.getTag();
		}

		Item item = data.get(position);
		holder.txtTitle.setText(Html.fromHtml(item.getTitle()));
		holder.imageItem.setImageURI(item.getImage());
		//holder.imageItem.setImageResource(item.getPicture());

		return row;

	}

	static class RecordHolder {
		TextView txtTitle;
		ImageView imageItem;

	}


}

package com.basilic.db;

import java.util.ArrayList;
import com.basilic.db.tablekey.ABOUT_KEY;
import com.basilic.db.tablekey.CHANGES_KEYS;
import com.basilic.db.tablekey.FotoGalleryKey;
import com.basilic.db.tablekey.MenuKey;
import com.basilic.db.tablekey.NewsKey;
```

```
import com.basilic.db.tables.AboutResTable;
import com.basilic.db.tables.BasilicDBItemWithFoto;
import com.basilic.db.tables.CHANGE;
import com.basilic.news.NewsItem;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

public class BasilicDB extends SQLiteOpenHelper {

	// private SQLiteDatabase liteDatabase;

	public static final String DATABASE = "basilic";
	public static final int VERSION = 1;

	private String CREATE_FOTOGALLERY_TABLE = "create table "
			+ TableName.FOTO_GALLERY + " ( " + FotoGalleryKey.KEY_ID
			+ " INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, "
			+ FotoGalleryKey.KEY_URL_SMALL_FOTO + " text,"
			+ FotoGalleryKey.KEY_URL_BIG_FOTO + " text,"
			+ FotoGalleryKey.KEY_DATE + " text," + FotoGalleryKey.KEY_TITLE
			+ " text," + FotoGalleryKey.KEY_DESC + " text,"
			+ FotoGalleryKey.KEY_MDATE + " text );";

	private String CREATE_MENU_TABLE = "create table " + TableName.MENU + " ( "
			+ MenuKey.KEY_ID + " INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
"
			+ MenuKey.TITLE_KEY + " text," + MenuKey.DESC_KEY + " text,"
			+ MenuKey.FOTO_ULR_SMALL_KEY + " text," +
MenuKey.FOTO_URL_BIG_KEY
			+ " text," + MenuKey.DATE_KEY + " text," + MenuKey.M_DATE_KEY
			+ " text );";

	private String CREATE_NEWS_TABLE = "create table " + TableName.NEWS + " ( "
			+ NewsKey.KEY_ID + " INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
"
			+ NewsKey.TITLE_KEY + " text," + NewsKey.SMALL_TEXT_KEY + "
text,"
			+ NewsKey.BIG_TEXT_KEY + " text," + NewsKey.DATE_KEY + " text,"
			+ NewsKey.M_DATE_KEY + " text );";

	private String CREATE_ABOUT = "create table " + TableName.ABOUT_RES + " ( "
			+ ABOUT_KEY.KEY_ID
			+ " INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, "
			+ ABOUT_KEY.KEY_NAME + " text," + ABOUT_KEY.KEY_MDATE + " text,"
			+ ABOUT_KEY.KEY_INFO + " text );";

	private String CREATE_CONTACTS = "create table " + TableName.CONTACTS + " ( "
			+ ABOUT_KEY.KEY_ID
			+ " INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, "
			+ ABOUT_KEY.KEY_NAME + " text," + ABOUT_KEY.KEY_MDATE + " text,"
			+ ABOUT_KEY.KEY_INFO + " text );";

	private String CREATE_CHANGES = "create table " + TableName.CHANGES + " ( "
			+ CHANGES_KEYS.KEY_ID
			+ " INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, "
			+ CHANGES_KEYS.KEY_ABOUT + " text," + CHANGES_KEYS.KEY_CONTACTS
			+ " text," + CHANGES_KEYS.KEY_FOTO + " text,"
```

```
                + CHANGES_KEYS.KEY_MENU + " text," + CHANGES_KEYS.KEY_NEWS
                + " text );";

        public BasilicDB(Context context) {
                super(context, DATABASE, null, VERSION);
        }

        @Override
        public void onCreate(SQLiteDatabase db) {
                db.execSQL(CREATE_FOTOGALLERY_TABLE);
                db.execSQL(CREATE_MENU_TABLE);
                db.execSQL(CREATE_NEWS_TABLE);
                db.execSQL(CREATE_ABOUT);
                db.execSQL(CREATE_CHANGES);
                db.execSQL(CREATE_CONTACTS);
        }

        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
                // Drop older books table if existed
                db.execSQL("DROP TABLE IF EXISTS " + TableName.FOTO_GALLERY);
                db.execSQL("DROP TABLE IF EXISTS " + TableName.MENU);
                db.execSQL("DROP TABLE IF EXISTS " + TableName.NEWS);
                db.execSQL("DROP TABLE IF EXISTS " + TableName.ABOUT_RES);
                db.execSQL("DROP TABLE IF EXISTS " + TableName.CHANGES);
                db.execSQL("DROP TABLE IF EXISTS " + TableName.CONTACTS);

                this.onCreate(db);
        }

        // public boolean Exists(String _id) {
        // SQLiteDatabase mDb = this.getReadableDatabase();
        // Cursor cursor = mDb.rawQuery("select 1 from "
        // + TableNames.TABLE_ACCOUNT + " where " + AccountKey.KEY_ID
        // + "=?", new String[] { _id });
        // boolean exists = (cursor.getCount() > 0);
        // cursor.close();
        // return exists;
        // }

        public boolean isTableExists(String tableName) {

                SQLiteDatabase db = this.getReadableDatabase();

                if (tableName == null || db == null || !db.isOpen()) {
                        return false;
                }
                Cursor cursor = db
                                .rawQuery(
                                                "SELECT COUNT(*) FROM sqlite_master WHERE type
= ? AND name = ?",
                                                new String[] { "table", tableName });
                if (!cursor.moveToFirst()) {
                        return false;
                }
                int count = cursor.getInt(0);
                cursor.close();
                return count > 0;
        }

        public void addFotoItem(BasilicDBItemWithFoto foto_item) {
```

```
        SQLiteDatabase database = this.getWritableDatabase();

        ContentValues contentValues = new ContentValues();
        Log.d("foto", foto_item.getM_date());
        // contentValues.put(FotoGalleryKey.KEY_ID, foto_item.getId());
        contentValues.put(FotoGalleryKey.KEY_URL_SMALL_FOTO,
                    foto_item.getUrl_small_foto());
        contentValues.put(FotoGalleryKey.KEY_URL_BIG_FOTO,
                    foto_item.getUrl_big_foto());
        contentValues.put(FotoGalleryKey.KEY_DATE, foto_item.getDate());
        contentValues.put(FotoGalleryKey.KEY_TITLE, foto_item.getTitle());
        contentValues.put(FotoGalleryKey.KEY_DESC, foto_item.getDesc());
        contentValues.put(FotoGalleryKey.KEY_MDATE, foto_item.getM_date());

        database.insert(TableName.FOTO_GALLERY, null, contentValues);
        database.close();
    }

    public void addAboutInfo(com.basilic.db.tables.AboutResTable aboutRes) {

        SQLiteDatabase database = this.getWritableDatabase();

        ContentValues contentValues = new ContentValues();
        // Log.d("foto", foto_item.getM_date());
        // contentValues.put(FotoGalleryKey.KEY_ID, foto_item.getId());

        contentValues.put(ABOUT_KEY.KEY_NAME, aboutRes.getName());
        contentValues.put(ABOUT_KEY.KEY_INFO, aboutRes.getInfo());
        contentValues.put(ABOUT_KEY.KEY_MDATE, aboutRes.getmDate());

        database.insert(TableName.ABOUT_RES, null, contentValues);
        database.close();

    }

    public void addContactsInfo(com.basilic.db.tables.AboutResTable aboutRes) {

        SQLiteDatabase database = this.getWritableDatabase();

        ContentValues contentValues = new ContentValues();
        // Log.d("foto", foto_item.getM_date());
        // contentValues.put(FotoGalleryKey.KEY_ID, foto_item.getId());

        contentValues.put(ABOUT_KEY.KEY_NAME, aboutRes.getName());
        contentValues.put(ABOUT_KEY.KEY_INFO, aboutRes.getInfo());
        contentValues.put(ABOUT_KEY.KEY_MDATE, aboutRes.getmDate());

        database.insert(TableName.CONTACTS, null, contentValues);
        database.close();

    }

    public void addMenuItem(BasilicDBItemWithFoto foto_item) {

        SQLiteDatabase database = this.getWritableDatabase();

        ContentValues contentValues = new ContentValues();
        Log.d("foto", foto_item.getM_date());
        // contentValues.put(FotoGalleryKey.KEY_ID, foto_item.getId());
        contentValues.put(MenuKey.FOTO_ULR_SMALL_KEY,
```

```
                        foto_item.getUrl_small_foto());
                contentValues
                        .put(MenuKey.FOTO_URL_BIG_KEY,
foto_item.getUrl_big_foto());
                contentValues.put(MenuKey.DATE_KEY, foto_item.getDate());
                contentValues.put(MenuKey.TITLE_KEY, foto_item.getTitle());
                contentValues.put(MenuKey.DESC_KEY, foto_item.getDesc());
                contentValues.put(MenuKey.M_DATE_KEY, foto_item.getM_date());

                database.insert(TableName.MENU, null, contentValues);
                database.close();
        }

        public void addNewsItem(NewsItem news_item) {

                SQLiteDatabase database = this.getWritableDatabase();

                ContentValues contentValues = new ContentValues();

                contentValues.put(NewsKey.TITLE_KEY, news_item.getTitle());
                contentValues.put(NewsKey.SMALL_TEXT_KEY, news_item.getSmall_text());
                contentValues.put(NewsKey.BIG_TEXT_KEY, news_item.getFull_text());
                contentValues.put(NewsKey.DATE_KEY, news_item.getDate());
                contentValues.put(NewsKey.M_DATE_KEY, news_item.getMdate());

                database.insert(TableName.NEWS, null, contentValues);
                database.close();
        }

        public void addChanges(CHANGE change) {

                SQLiteDatabase database = this.getWritableDatabase();

                ContentValues contentValues = new ContentValues();

                contentValues.put(CHANGES_KEYS.KEY_ABOUT, change.getAbout());
                contentValues.put(CHANGES_KEYS.KEY_CONTACTS, change.getContacts());
                contentValues.put(CHANGES_KEYS.KEY_FOTO, change.getPhoto());
                contentValues.put(CHANGES_KEYS.KEY_MENU, change.getMenu());
                contentValues.put(CHANGES_KEYS.KEY_NEWS, change.getNews());
                Log.d("CHANGE", change.getMenu());
                database.insert(TableName.CHANGES, null, contentValues);
                database.close();

        }

        public CHANGE getChanges() {

                CHANGE change = new CHANGE();

                SQLiteDatabase database = this.getReadableDatabase();

                String query = "SELECT  * FROM " + TableName.CHANGES + ";";

                Cursor cursor = database.rawQuery(query, null);

                if (cursor.moveToFirst()) {

                        change.setAbout(cursor.getString(cursor
                                        .getColumnIndex(CHANGES_KEYS.KEY_ABOUT)));
                        change.setContacts(cursor.getString(cursor
```

```
                        .getColumnIndex(CHANGES_KEYS.KEY_CONTACTS)));
                change.setMenu(cursor.getString(cursor
                        .getColumnIndex(CHANGES_KEYS.KEY_MENU)));
                change.setNews(cursor.getString(cursor
                        .getColumnIndex(CHANGES_KEYS.KEY_NEWS)));
                change.setPhoto(cursor.getString(cursor
                        .getColumnIndex(CHANGES_KEYS.KEY_FOTO)));

        }

        database.close();

        return change;

}

public AboutResTable getAbout() {

        AboutResTable aboutRes = new AboutResTable();

        SQLiteDatabase database = this.getReadableDatabase();

        String query = "SELECT  * FROM " + TableName.ABOUT_RES + ";";

        Cursor cursor = database.rawQuery(query, null);

        if (cursor.moveToFirst()) {

                aboutRes.setId(cursor.getInt(cursor
                        .getColumnIndex(ABOUT_KEY.KEY_ID)));
                aboutRes.setmDate(cursor.getString(cursor
                        .getColumnIndex(ABOUT_KEY.KEY_MDATE)));
                aboutRes.setInfo(cursor.getString(cursor
                        .getColumnIndex(ABOUT_KEY.KEY_INFO)));
                aboutRes.setName(cursor.getString(cursor
                        .getColumnIndex(ABOUT_KEY.KEY_NAME)));

        }

        database.close();

        return aboutRes;

}

public AboutResTable getContacts() {

        AboutResTable aboutRes = new AboutResTable();

        SQLiteDatabase database = this.getReadableDatabase();

        String query = "SELECT  * FROM " + TableName.CONTACTS + ";";

        Cursor cursor = database.rawQuery(query, null);

        if (cursor.moveToFirst()) {

                aboutRes.setId(cursor.getInt(cursor
                        .getColumnIndex(ABOUT_KEY.KEY_ID)));
                aboutRes.setmDate(cursor.getString(cursor
                        .getColumnIndex(ABOUT_KEY.KEY_MDATE)));
```

```
                aboutRes.setInfo(cursor.getString(cursor
                        .getColumnIndex(ABOUT_KEY.KEY_INFO)));
                aboutRes.setName(cursor.getString(cursor
                        .getColumnIndex(ABOUT_KEY.KEY_NAME)));

            }

            database.close();

            return aboutRes;

        }

        public ArrayList<BasilicDBItemWithFoto> getFotoItem() {
            SQLiteDatabase database = this.getReadableDatabase();

            String query = "SELECT  * FROM " + TableName.FOTO_GALLERY + ";";

            Cursor cursor = database.rawQuery(query, null);

            ArrayList<BasilicDBItemWithFoto> fotoGalleries = new
ArrayList<BasilicDBItemWithFoto>();

            if (cursor.moveToFirst()) {

                do {

                    fotoGalleries
                            .add(new BasilicDBItemWithFoto(
                                    cursor.getString(cursor

        .getColumnIndex(FotoGalleryKey.KEY_URL_SMALL_FOTO)),
                                    cursor.getString(cursor

        .getColumnIndex(FotoGalleryKey.KEY_URL_BIG_FOTO)),
                                    cursor.getString(cursor

        .getColumnIndex(FotoGalleryKey.KEY_DATE)),
                                    cursor.getString(cursor

        .getColumnIndex(FotoGalleryKey.KEY_TITLE)),
                                    cursor.getString(cursor

        .getColumnIndex(FotoGalleryKey.KEY_DESC)),
                                    cursor.getString(cursor

        .getColumnIndex(FotoGalleryKey.KEY_MDATE))));

                } while (cursor.moveToNext());

            }

            database.close();

            return fotoGalleries;

        }

        public ArrayList<BasilicDBItemWithFoto> getMenuItem() {
            SQLiteDatabase database = this.getReadableDatabase();
```

```
            String query = "SELECT  * FROM " + TableName.MENU + ";";

            Cursor cursor = database.rawQuery(query, null);

            ArrayList<BasilicDBItemWithFoto> fotoGalleries = new
ArrayList<BasilicDBItemWithFoto>();

            if (cursor.moveToFirst()) {

                  do {

                        fotoGalleries.add(new BasilicDBItemWithFoto(cursor
                                    .getString(cursor

      .getColumnIndex(MenuKey.FOTO_ULR_SMALL_KEY)),
                                    cursor.getString(cursor

      .getColumnIndex(MenuKey.FOTO_URL_BIG_KEY)),
                                    cursor.getString(cursor

      .getColumnIndex(MenuKey.DATE_KEY)), cursor
                                                .getString(cursor

      .getColumnIndex(MenuKey.TITLE_KEY)),
                                    cursor.getString(cursor

      .getColumnIndex(MenuKey.DESC_KEY)), cursor
                                                .getString(cursor

      .getColumnIndex(MenuKey.M_DATE_KEY))));

                  } while (cursor.moveToNext());

            }

            database.close();

            return fotoGalleries;

      }

      public ArrayList<NewsItem> getNewsItem() {
            SQLiteDatabase database = this.getReadableDatabase();

            String query = "SELECT  * FROM " + TableName.NEWS + ";";

            Cursor cursor = database.rawQuery(query, null);

            ArrayList<NewsItem> news = new ArrayList<NewsItem>();

            if (cursor.moveToFirst()) {

                  do {

                        NewsItem newsItem = new NewsItem();

                        newsItem.setTitle(cursor.getString(cursor
                                    .getColumnIndex(NewsKey.TITLE_KEY)));
                        newsItem.setSmall_text(cursor.getString(cursor
                                    .getColumnIndex(NewsKey.SMALL_TEXT_KEY)));
                        newsItem.setFull_text(cursor.getString(cursor
```

```
                                .getColumnIndex(NewsKey.BIG_TEXT_KEY)));
                        newsItem.setDate(cursor.getString(cursor
                                .getColumnIndex(NewsKey.DATE_KEY)));
                        newsItem.setMdate(cursor.getString(cursor
                                .getColumnIndex(NewsKey.M_DATE_KEY)));

                        news.add(newsItem);

                } while (cursor.moveToNext());

        }

        database.close();

        return news;

}

public void deleteAllMenu() {

        // 1. get reference to writable DB
        SQLiteDatabase db = this.getWritableDatabase();

        db.delete(TableName.MENU, null, null);

        // 3. close
        db.close();

        // log
        //Log.d("deleteBook", book.toString());

}

public void deleteAllAbout() {

        // 1. get reference to writable DB
        SQLiteDatabase db = this.getWritableDatabase();

        db.delete(TableName.ABOUT_RES, null, null);

        // 3. close
        db.close();

        // log
        //Log.d("deleteBook", book.toString());

}

public void deleteAllContacts() {

        // 1. get reference to writable DB
        SQLiteDatabase db = this.getWritableDatabase();

        db.delete(TableName.CONTACTS, null, null);

        // 3. close
        db.close();

        // log
        //Log.d("deleteBook", book.toString());
```

```
	}

	public void deleteAllChanges() {

		// 1. get reference to writable DB
		SQLiteDatabase db = this.getWritableDatabase();

		db.delete(TableName.CHANGES, null, null);

		// 3. close
		db.close();

		// log
		//Log.d("deleteBook", book.toString());

	}

	public void deleteAllNews() {

		// 1. get reference to writable DB
		SQLiteDatabase db = this.getWritableDatabase();

		db.delete(TableName.NEWS, null, null);

		// 3. close
		db.close();

		// log
		//Log.d("deleteBook", book.toString());

	}

	public void deleteAllFoto() {

		// 1. get reference to writable DB
		SQLiteDatabase db = this.getWritableDatabase();

		db.delete(TableName.FOTO_GALLERY, null, null);

		// 3. close
		db.close();

		// log
		//Log.d("deleteBook", book.toString());

	}
}
```