R&D Разработки проекта “StudyPlease”

07 сентября 2019 г..

Обзор

Текущий документ описывает программные компоненты и методологии разработки, необходимые и применимые в рамках процесса подготовки полного цикла создания программного обеспечения высокого уровня (high level) - абстракции, реализовывающей интерфейсы взаимодействия с пользователем; графический интерфейс; приложение, напрямую связанное с UX-свойствами (описание далее). Полный цикл создания ПО включает в себя процесс подготовки мок-апов, а также графических материалов, паттернов и подходов к проектированию современного UX (пользовательского опыта) - как первичной итерации создания прототипа проекта. Вторая итерация включает в себя услуги разработки приложений для разных платформ, первичное построение архитектуры, проектирование базы данных верхнего и нижнего уровней (см. описание в соответствующем разделе), отладку разработанных приложений, их тестирование и проведение нагрузочных проверок работоспособности ПО.

Завершающая итерация, является побочной и предполагает дальнейшую поддержку приложения, выпуск обновлений и отслеживания статитстки распространения программного обеспечения в рамках разных магазинов приложений. По результатам этой итерации, формируется документ-отчетность с

применением эмпирических и статистических измерений, их анализом и предложениями по внедрению тех или иных изменений.

Текущий документ описывает план разработки полного цикла, программно-аппаратную базу и требования, необходимые для работы на исполняющих устройствах. В отдельную нормативную базу выведен стек используемых технологий и их ресурсозатраты.

Содержание

I. Общий анализ проекта

Концепция разработки

Сервис являет собой пример классического SaaS (software-as-a-service) продукта, в рамках которого программа и набор сопровождающих технологий позиционируется в качестве способа предоставления сервисных услуг.

Основная цель приложения - предоставление услуг покупки клиентом занятий и контроля билетов купленных занятий относительно групп менеджерами, с включённой функцией supervise-контроля администратором.

Программный комплекс поддерживает набор функционала взаимодействия, позволяющий реализовать программу как сервисный продукт. В этот набор входят:

- **OAuth-система подтверждения регистраций и верификации пользователей** - для избежания атак на сервис и компрометации работы приложения (прим. - методы GarbageCan и MassTracing) используется система подтверждения регистрации новых пользователей с помощью текстового сообщения или телефонного звонка.
- **Виджет-система построения графического интерфейса** - в процессе разработки используется распределенный фреймворк React Native, который поддерживает специфическую текстовую разметку и систему настройки графических интерфейсов на основе блоков - виджетов, каждый из которых выполняет определенный функционал.
- **FireBase-система учета пользователей и добавления их данных** - после регистрации каждого нового пользователя, формируется список

обновленных данных, который содержит детали о каждом пользователе в качестве специального дата-файла.

- **SQL-база данных доступных паркомест** - для хранения и обработки всей доступной информации о парковочных местах, их владельцах, стоимостях и прочей инфомрации используется реляционная база данных, которая реализовывает принцип хранения данных в распределенных блоках - чанках - и использует специальный синтаксис обработки, задания и редактирования данных в ней.
- **База учета истории операций и действий приложения** - все действия, выполненные пользователями в рамках программы, включая историю подачи заявок на аренду, сдачи паркомест, добавления новых объектов, а также сумм и дат денежных транзакций - пополнений счета и переводов внутренней валюты между аккаунтами клиентов сервиса. Для учета этих операций и формирования отчетностей о них используетяс веб-интерфейс и предварительно разработанная легковесная CRM-система.
- **Лог-файл с отчетами работы программы** - для отслеживания вероятных проблем работ сервиса, а также обнаружения уязвимостей, кждое действие записывается в специальный файл отчетности расширением .log. Это позволяет определить время возникновения неполадки, точно ее идентифицировать и оперативно внести изменения в код программы.
- **Invoice-сервис приема платежей и перевода на внутренний баланс** - для автоматизации перевода платежей с банковских счетов и обеспечения протокола безопасности, определенного предоставителями банковских услуг, используются внешние invoice-сервисы - шифрованные протоколы подтверждения транзакций.
- **Веб-сервис отправки запросов на выплаты -** Для обеспечения возможностей проведения контрольных выплат предполагается

разработка специализированного веб-сервиса, который в формате истории заказов на выплаты отправляет администратору или управляющему лицу уведомление о заказе новой выплаты.

- **Встроенная система верификации пользователей -** для обеспечения безопасности операций в рамках сервиса, минимизации риска появления “искусственных” данных о платежах и внутренних переводах, а также невозможности скомпрометировать внутренние составляющие работы программы, внедряется система верификации пользователей, которая разбивается на внешний и внутренний слои - первый ориентирован на взаимодействие с клиентом (прим. - ввод капчи), второй - на неявную проверку параметров устройства клиента, например, mac-адрес, история использования приложений и прочее.

Вводные данные

На этапе проектной подготовки получен макет PoC-версии, разработанный с помощью программы Canva.

Предложены базовые варианты цветовой палитры, шрифтов и общего размещения UI/UX-элементов, форма и внешний вид кнопок а также логические варианты переходов между контрольными модулями.

Для старта разработки сформировано текстовое техническое задание с проработкой общей стратегии развития и алгоритмов разработки проекта, требования к используемой материально-технической и программной базе, а также базовая архитектура проекта с учетом перспективы разработки под разные платформы. Детали подготовки проекта описываются в дальнейших разделах и включают примеры макетов, описания и правила создания дизайна в рамках поставленных задач, блок проектирования архитектуры программной

компоненты и базы данных, список технических требований и технологий, предполагаемых к задействованию в процессе разработки, а также предложения по дальнейшему развитию и продвижению сервиса, отслеживанию статистик загрузок, внедрение SaaS-продукта в использование фокус-группами, а также подготовка медиаматерилов и составляющих, необходимых для публикации продукта в магазинах приложений и глобальных маркетплейсах.

Исходящие данные

Согласно проведенным обсуждениям и согласованиям проекта, в исходящие материалы входит следующий набор продуктов, соответственно процессу разработки:

- Research & Development-план на полный цикл разработки сервиса, включающий его информационную поддержку, подготовку контентных материалов и стратегию развития.
- Создание макетов (mockups), брендинга, и интерфейсной составляющей дизайна сервиса - исходящими файлами являются макетные заготовки mockups, презентационный файл, макеты дизайна приложения для Android и iOS, дизайн веб-сервиса, мокап веб-сайта и контентные материалы.
- Проектирование комплексной архитектуры сервиса - документ с планами разработки и используемых технологий (включен в текущий).
- Разработка приложения для Android - исходный файл-инсталлятор.
- Разработка приложения для iOS - исходный пакет файлов-инсталляторов.
- Рекламный веб-сайт - индекс-файлы с полным каталогом загрузочных материалов для публикации на сервере.
- Контентные материалы - пакет сопровождающих материалов, включающих аудио-, видео- и фотоконтент, предназначенные для

распространения в сети и оформления страниц сервиса в магазинах приложений и на маркетплейсах.

- Аналитика распространения - табличные файлы и документы статистики загрузок, оценок пользовательской фокус группы и общий математический рассчет показателей продаж.

Проектная аналитика

R&D Проекта содержит:

- 6 разделов информации
- 47 страниц описания
- 4 таблицы
- 18 изображений
- 6 схем
- 4 выноски уточнений
- 6 диаграмм
- 19 ссылок на внешние источники
- 24 уточняющих блока

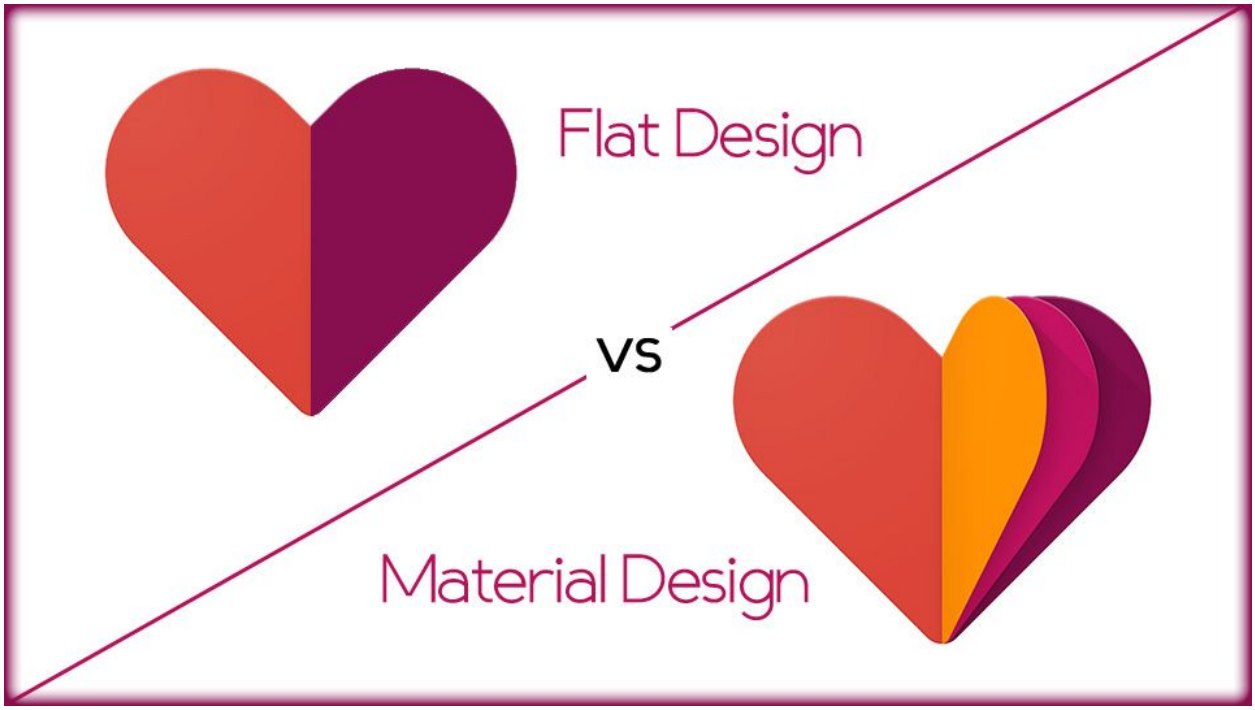
II. Описание способов подготовки решений UI/UX

За время существования различных платформ и операционных систем, зародился принцип “независимости дизайна”, согласно которого компании-собственники каждой системы предлагают собственные подходы и концепции разработки дизайна для программного обеспечения, аргументируя это проработанным подходом и анализом опыта пользовательского удобства использования. Изменения между платформми могут быть как совершенно минимальными (примеры - MacOS и Windows - единый оконный интерфейс с отличиями в размещении; Series40 и Symbian - единый дизайн и отличия в UX), так и предполагать совершенно разную “философию” использования и разработки (пример - Android и iOS). В силу значительного распределения рынка мобильных устойств между двумя упомянутыми выше платформами, данный документ не предполагает анализ других мобильный систем, но рассматривает разработку только под данные виды программного обеспечения.

Компании-владельцы Операционных Систем отдельно занимаются формирование паттернов проектирования и дизайна - документов, описывающих правила подготовки дизайн-решений для конкретно их платформы. С одной стороны, разработка програмнного обеспечения в соответствии с указанными стандартами позволяет унифицировать программы в маркетплейсах и свести их к единому, проработанному дизайну, но с другой - лишает приложение уникальности и приводит или к “смешению” с остальными подобными, или к потери идентифицируемости. Ниже рассмотрены некоторые примеры разработки программ в каждом из аспектов и альтернативные варианты проектной подготовки.

Android Patterns: Material Design 1.0/2.0

В сентябре 2014 года компания Google представила Android версии 5.0, ознаменовав тем самым эру “унифицированного” дизайна и представив подход, получивший название “Material Design”. Исторически, этот стиль вылился из стиля Flat Vector, но обзавелся наличием градиентов, размытых теней, а также правилами комбинирования строго определенного количества цветов - вещей, нехарактерных стилю предшественника.

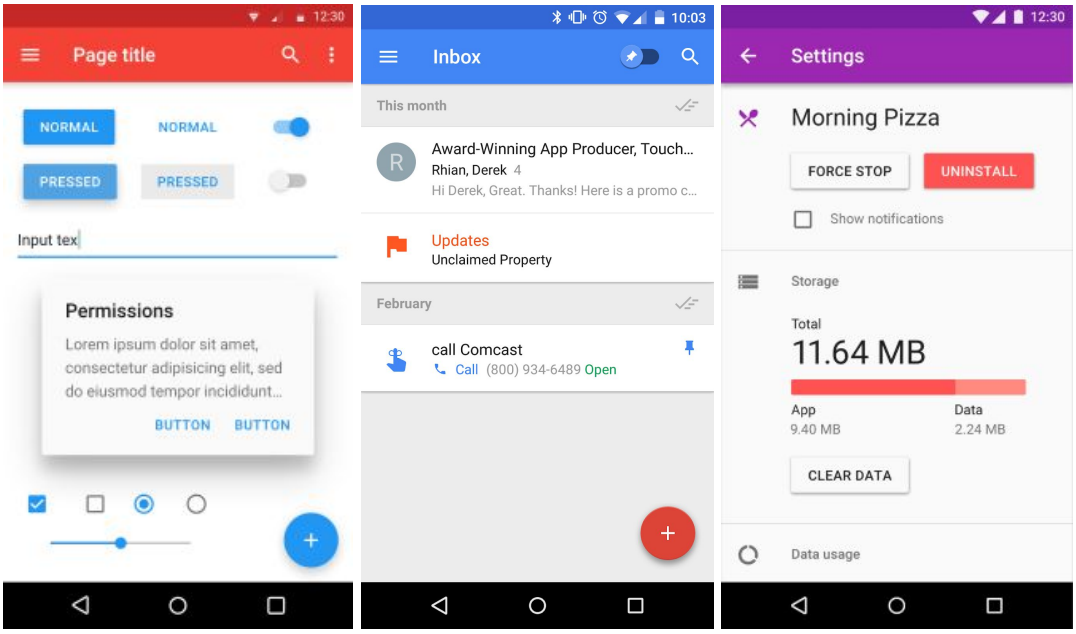


Согласно запросу, отправленному компанией в Google Dublin, на сегодня Material Design определяет следующие “классические” свойства стиля:

- Прямые и простые формы - закругленные углы встречаются крайне редко - есть или круги, или прямоугольники

- Размытые тени - жестких теней в дизайне такого стиля не существует - они все размываются и отводятся на небольшое расстояние от объекта с внедрением полупрозрачности.
- Шрифты не допускают закругленных форм или чрезмерной фривольности - рекомендуется использовать строгие Roboto, Open Sans, Ubuntu.
- В отличие от Flat Vector, Material Design должен иметь многослойность - выржать ее можно как в цветах или тенях, так и в формах и угловых наклонах.
- В анимациях, срок перехода между стартом и завершением интеракции должен составлять от 0.2 до 0.5 секунд, но не более.

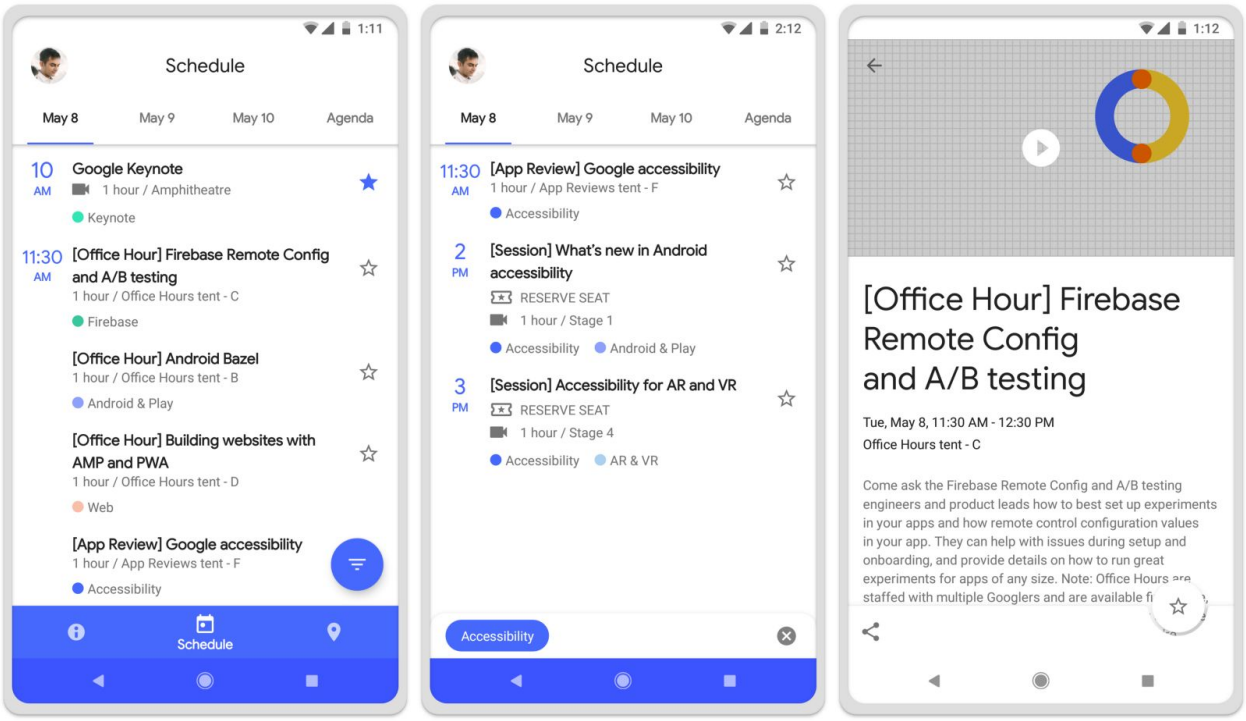
Ниже приведены примеры классических приложений-представителей Material Design:



В мае 2018 года, на выставке Google IO18, был представлен обновленный подход к концепции Material Design, который внедрил больше плавности элементам и увеличил уровень гибкости настроек. Таким образом, пункты 1, 4, 5 предыдущего списка стали иметь скорее рекомендательный характер.

С аналитической точки зрения, внедрение “плавности” форм и опыта взаимодействия в дизайн было результатом появления и массового развития технологий “закругленных” и “безрамочных” дисплеев, и, итог - общего скругления сводящих форм. Изображение приведено ниже.

Первая фотография - digitaltrends.com, 2019 год; вторая - тот же ресурс, 2014 год



Google I/O - Home | Google I/O - Filtered | Google I/O - Event

Рендеры новой версии дизайна с конференции Google IO 2018 на примере приложения выставки - все изображения и мокапы выполнены с рассчетом закругленных граней дисплея.

IOS Patterns: Human Interface

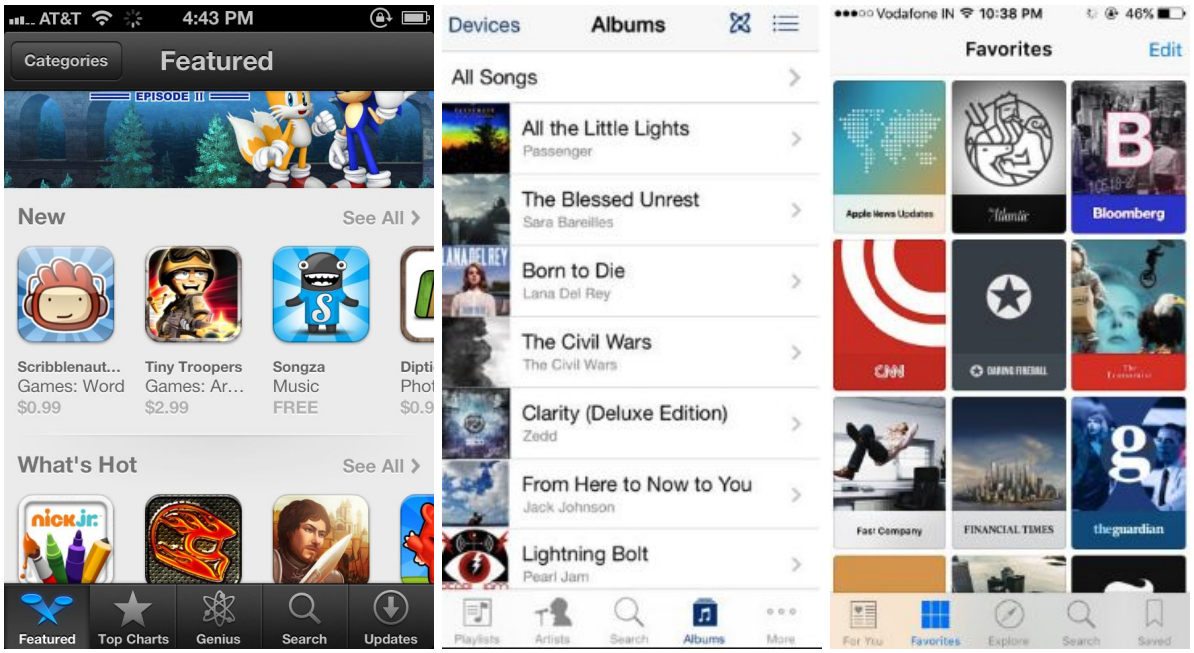
В отличие от паттернов Android, ориентированных на четкое очертание форм, цветов и размерностей теней, правила дизайна Apple характеризуются общими свойствами, которые не привязаны к параметрам цвета или строгости цветов. Причиной такого оформления “общих” концепций является постоянное экспериментирование компании с внешним видом оболочки разных версий iOS, в объеме, информативности, детализации и прочим аспектам.

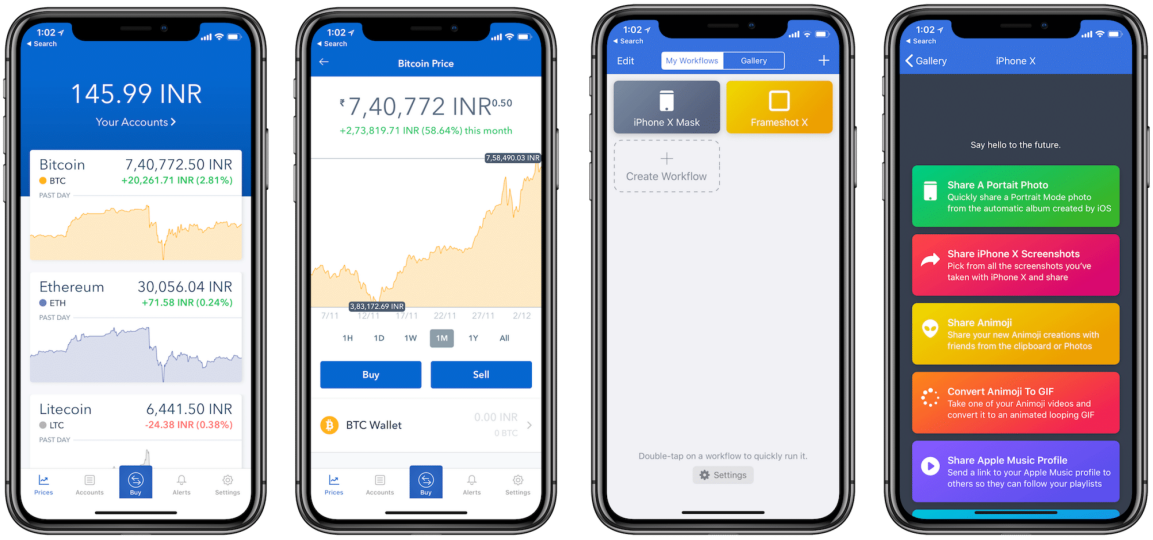
Согласно запросу от 13 февраля 2019 года, разработка под iOS согласно следованию принципов Human Design несет следующие аспекты:

- Ясность - текст и графические элементы равномерно распределяются по экранной области, в том время как важный и первоочередный контент может выделяться с помощью инверсий или свойств интерактивности.
- “Уважение” - для демонстрации расширяемого контента используется полупрозрачность, а градиенты и рамки - табу.
- Глубина - параметр, появившийся по факту внедрения в систему 3D Touch. Предполагает использование теней, а также параллакс-эффекта для отображения и имитации глубины дисплея.

Помимо этого, пользовательский интерфейс предполагает жесткое связывание контента и экранов детализации - так, каждый экран и контекстное меню должен иметь взаимосвязь переход со следующим и предыдущим экраном; следующий экран может быть только один, в то время как к предыдущему можно вернуться из разных интерфейсных компонент приложения.

Также, использование градиентов в программе возможно, невзирая на детали пункта 1: однако, градиент и градиентные переходы используются исключительно в шрифтах, кнопках, а также закрывающих фонах программы.





Комбинирование паттернов. Каноническая импровизация.

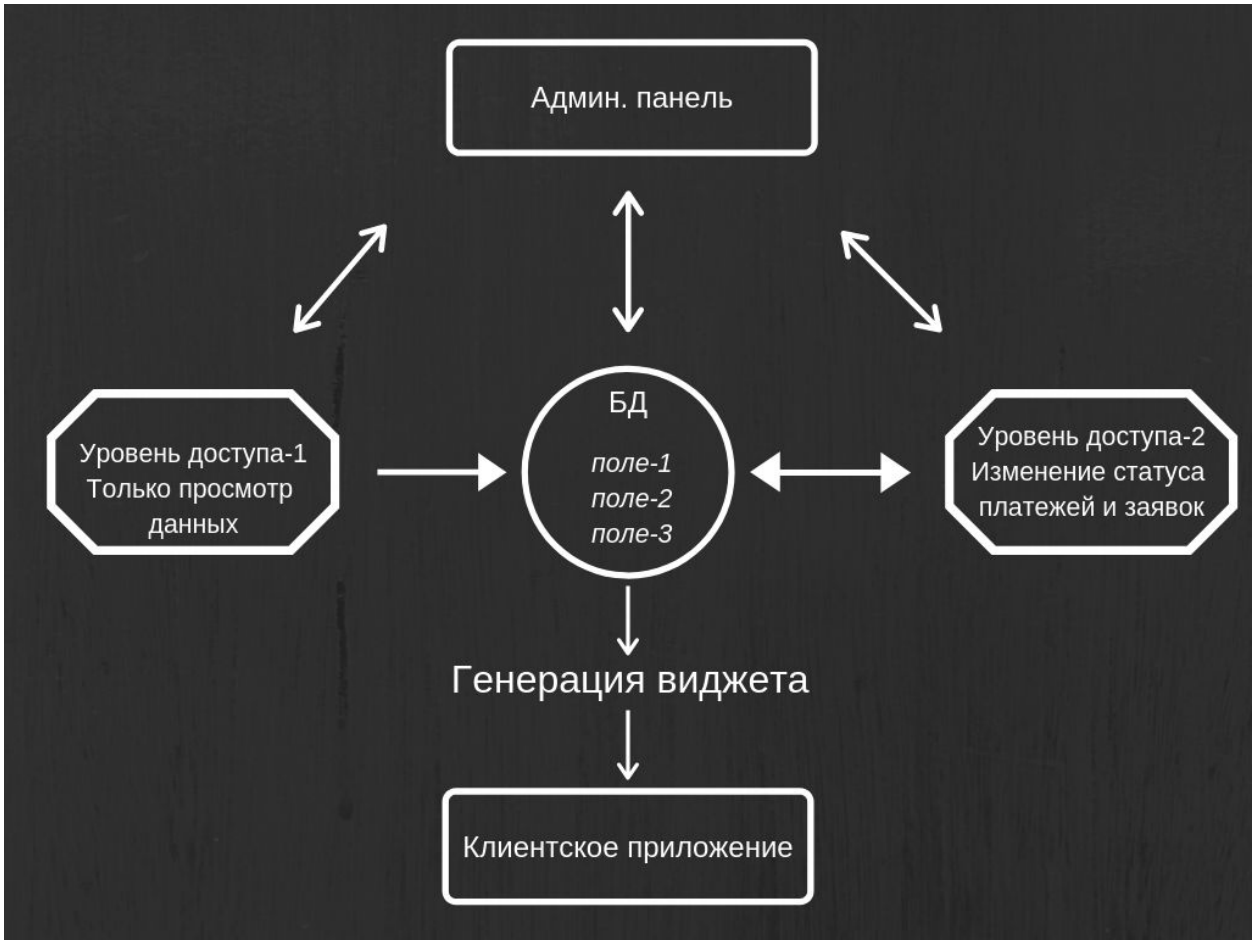
Оба дизайн-подхода, в целом, являются узкоориентированными на конкретный рынок и целевую аудиторию, а потому в контексте использования разных визуальныз решений на разных платформах предполагают значительные затраты на вариативаность разработки в дальнейшем. Как результат, многие компании стали выполнять объединение программных паттернов разных платформ, что позволило привести приложение не только к кроссплатформенносоти в процессе разработки программного кода, но и реализовать один общий дизайн. Обычно для таких программ выделяется исклбчительно общий набор параметров следования из разных guidelines прототипирования. Основным требованием является следование следующим аспектам:

- Соответствие технологическим тенденциям: программа должна учитывать скругленные края, виртуальную кнопку, наличие выносного модуля сверху, и другие особенности современного устройства.
- Соответствие требованиям маркетплейсов: есть обособленный ряд правил магазинов приложений, согласно которого все поступающие на публикацию программы проходят строгую валидацию. Пользовательский интерфейс должен быть разработан согласно этим аспектам.

III. План и процесс разработки

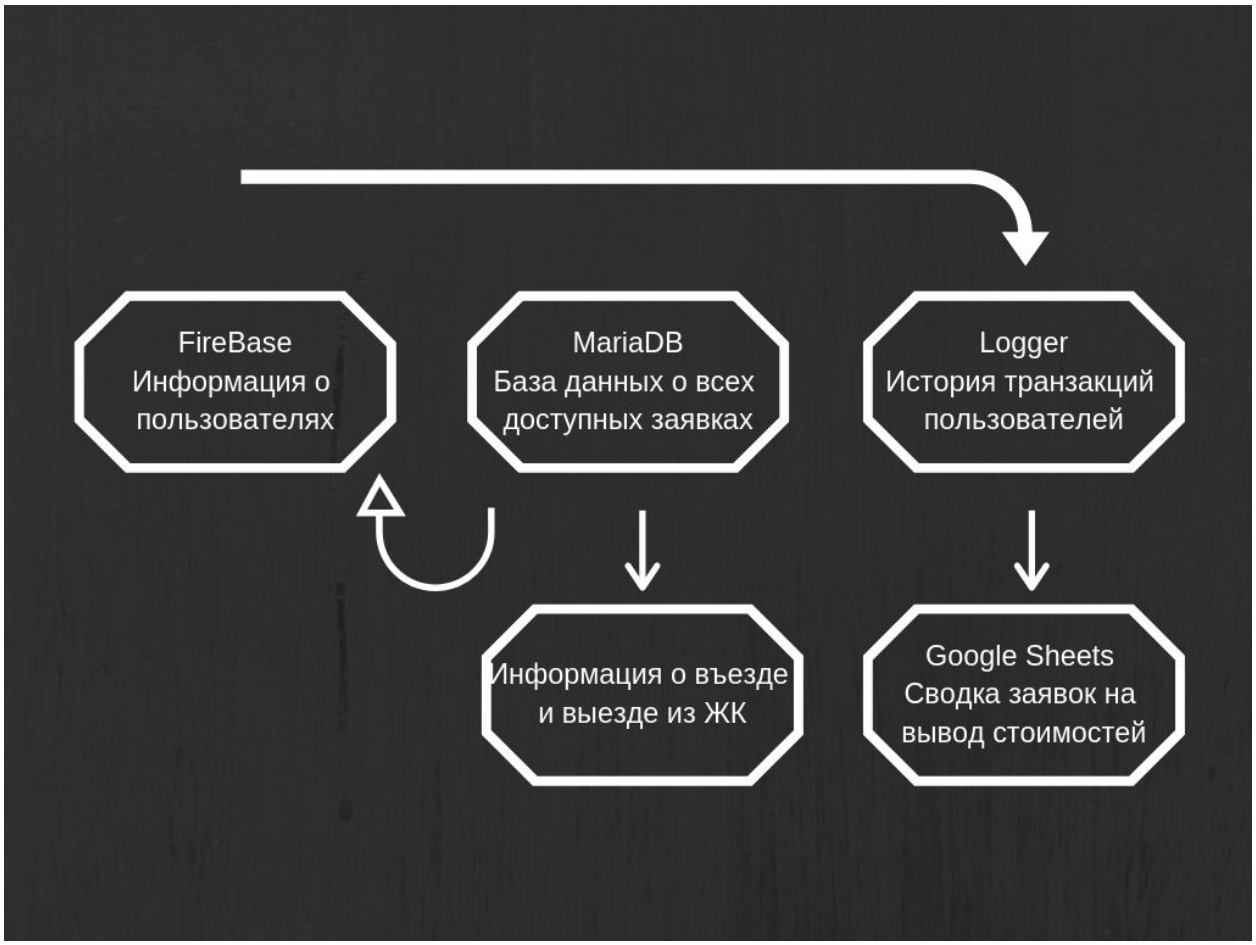
Архитектура проекта

Архитектура взаимодействий и составляющих частей проекта продемнострирована на следующем макете:



“Ядром” построения программы является распределенная реляционная база данных, которая разбивается на три составляющие - информация о

пользователях, информация о предложениях аренды, а также отдельный табличный блок запросов на выплату с историей транзакций пользователей:



- FireBase - система, поддерживающая многопользовательскую систему записи данных и информации о пользователях. Предназначается для хранения записей и деталей о клиентах сервиса.
- MariaDB - форк MySQL, упор которого сделан на ускоренную обработку данных и безопасность. Хранит основной массив данных о заявках на аренду, детали статуса их обработки, время появления и прочие детали.

- Logger - встроенный системный фрейм, предназначенный для хранения данных о работоспособности приложения, деталях транзакций, статистику передачи информации.

По факту наличия спроектированной базы данных, внедряется разработка мобильного приложения, с разбивкой на две составные компоненты - фронтенд и бекенд.

Frontend

Разработка включает в себя построение пользовательских интерфейсов, проектирование виджетов и графических компонент, построение логики UX-взаимодействия и настройку логических переходов взаимодействия. По факту, исходящим результатом является визуальная часть проекта, лишенная функционала, но предоствляющая возможности оценки пользовательской эффективности.

Список технологий:

- React - фреймворк, предназначенный для быстрого построения графических элементов и интерфейсов. Реализовывает принцип непрямой отрисовки (виртуальной) DOM-дерева, что значительно ускоряет процесс написания кода, а также время его отладки.
- React Native - адаптация интерфейсных элементов React под нативные платформы с помощью внутреннего преобразования. Таким образом, использование простого синтаксиса и деталей позволяет, тем не менее, писать адаптированный под любую необходимую платформу, рабочий, код.

- JSX - адаптирующий синтаксис языка JavaScript, который служит полифиллом и изменяет классический синтаксис языка в более простую версию.
- JavaScript - язык, адаптированный под веб-разработку с поддержкой асинхронности, блоков взаимодействия и многопоточности.
- Expo - средство подготовки и генерации проектов, обеспечивающее возможности быстрой сборки и компиляции под любую мобильную платформу.

Backend

Эта компонентная составляющая отвечает за корректную передачу данных, работу функционала программы, а также генерацию корректных ответов на взаимодействие пользователя с интерфейсом. В рамках бекенд-составляющей прописывается логика передачи запросов на базу данных, подключение сторонних API, модулей, а также шифрование всей информации, которая проходит по внешним и внутрненим протоколам подключения.

В список технологий входят сторонние модули подключения, а также языки разработки JavaScript или Python - описание способов разработки бекенд-составляющей представлены ниже.

Вариант 1 - использование чистого JavaScript

Данный способ разработки предполагает использование пакетного менеджера npm, технологий ECMAScript 6+ и синтаксиса языка JavaScript без внедрения сторонних языков и методологий разработки. Помимо этого, применяются

методы binding и state exchange, которые предполагают изменение функций и их привязку к состояниям графического интерфейса.

Как результат, вся разработка ведется в рамках единой экосистемы и этимологии программных языков, что позволяет минимизировать риск возникновения исполнительных ошибок и уменьшить время на отслеживание багов.

Вариант 2 - Написание бриджа технологий

В данном случае, в качестве языка разработки функциональной составляющей используется Python3. Весь программный код разбивается на модули с исходящими параметрами, которые передаются в фоновый процесс сгенерированного терминала. В свою очередь, данные из терминала в качестве команд шестнадцатиричного кода считываются графическим интерфейсом, который спакован с помощью системы Electron.

Общий список технологий с их детальным описанием представлен ниже:

- HTML - Hyper Text Markup Language - язык разметки интерфейсных элементов, используемый при проектировании макета будущего графического дизайна.
- CSS - Cascading Style Sheets - язык визуального оформления разметки, позволяющий добавить сформированному макету стили и визуальное оформление, а также анимации.
- Canvases - система построения блоков интерфейса, на основе которых генерируется интерфейс.

- Flexboxes - система размещения виджетов на дисплее, обеспечивающая общую адаптивность, а также возможность масштабироваться в разных форматах дисплея.
- Bootstrap - технология-поддержка для Flexboxes. Добавляет интерфейсу мобильности.
- Keyframes - система реализации анимаций, которая отвечает за приведение тех или иных элементов в движение.
- JavaScript ES6+ - язык программирования, отвечающий за первый слой интерактивного взаимодействия с системой построения интерфейсов, а также поддерживающий возможности связки между собой интерфейса и программной составляющей.
- NodeJS - интерпретатор языка программирования JavaScript, позволяющий запускать сервис не только в сети, но и на сервере или локальной машине.
- V8 - движок, реализовывающий все необходимые функции языка и принимающий в себя его внешние и внутренние параметры.
- NPM - Node Package Manager - менеджер дополнений, позволяющий расширять функционал языка программирования посредством открытых программных пакетов.
- ElectronJS - настраиваемая система, которая выполняет сборку проекта, написанного для веб-браузера, в десктопное приложение, делая возможность его использования в качестве обычной компьютерной программы.
- ReactJS - вспомогательный фреймворк, упрощающий процесс добавления виджетов и интерфейсных элементов.
- JSX - аналог XML-разметки, предоставляющий возможности написания JavaScript-кода в упрощенной для взаимодействия форме.

- JS2Terminal - пакет-расширение, предоставляющий возможность отправлятьсигналы и команды из языка JavaScript напрямую в терминал.
- PyShell - расширение, позволяющее считывать данные из терминала в язык программирования Python в качестве аргументов тех или иных функций, как способ взаимодействия между двумя языками, или слоями абстракции.
- WebPack - система настройки и отладки проекта, которая формирует набор файлов верстки и пакетных модулей зависимости, а также контентные материалы, в единый файл определенного расширения.
- Git - система контроля версий, позволяющая с удобством отслеживать все детали процесса разработки и контролировать данные, необходимые для взаимодействия с программой.
- GitHub - веб-версия системы контроля версий, позволяющая в ускоренном режиме просмотреть код, выполнить с ним взаимодействие и видоизменить.
- BitBucket - открытая система контроля версий, представленная как аналог Git с расширенным функционалом.
- SketchIt! - встроенная система построения UML-разметки и диаграмм классов для отслеживания и формирования архитектуры проекта.
- UML - Universal Markup Language - язык разметки, предназначенный для формирования архитектуры проектов и текстового описания диаграм классов.
- WebStorm - система подготовки проектов, их отладки, настройки и поиска ошибок запуска.
- brew - пакетный менеджер терминала, позволяющий устанавливать дополнения, программное обеспечение и детали разработки.

- WireFrame.cc - система прототипирования и подготовки мок-апов для отображения базовых элементов будущего дизайна.
- Proto.io - система быстрого прототипирования, предназначенная для формирования UI и UX пользовательских экранов.

Принцип работы

Принцип работы программной составляющей представлен на следующем мок-апе:

Процесс разработки

Ниже приведена таблица используемых команд и их описание, с целью продемонстрировать общую нормативную базу и архитектуру используемых методов и функций. Таблица дана ниже.

Название метода/функции	**Параметры**	**Предназначение**
flush()	-	Очищает все данные, введенные в промежуточный порт, и закрывает его.
app.on	Коллбек-функция	Создает окно программы, и инициализирует стартовую функцию.
BrowserWindow	Ширина, высота окна	Генерация UI-окна с указанными параметрами.
loadURL	Адрес файла	Подключает страницу к созданному графическому окну Electron.
getCurrentWebContents	-	Получает все детали страницы и ее элементы в контексте регулярной загрузки.
addEventListener	Коллбек-функция	Присваивает слушатель события, который выполняет определенный функционал по факту нажатия.

serial.Serial	Строка - адрес порта	Создает новый объект типа Serial, а также проверяет доступность порта по указанному адресу, в случае успеха открывая доступ к нему.
serial.write	Шестнадцатиричная запись	Подает на порт кодированную последовательность сигналов, предвартельно разбирая контентное содержание.
serial.read	Количество битов для считывания	Считывает с открытого порта определенное количество битов, промежуточно варьируя задержку между процессами считывания.
serial.readline	-	Считывает строку, условно разбитую отдельными символами табулирования. Не привязывается к количеству бит, устанавливая его автоматически.
baudrate	Константа	Значение, которое задает частоту обновления процесса подачи сигналов на UART-порт
serial.close	-	Команда закрытия порта, доступного по адресу, а также

		отключения подачи сигналов на него.
hexlify	Двоичные данные	Процесс регенерации данных формата двоичного кода в шестнадцатиричный формат и их выводе в типово-независимой форме.
unhexlify	Шестнадцатиричные данные	Процесс трансляции данных в шестнадцатиричном формате с их приведением в бинарный формат вывода.
rlecode_hqx	Чанк данных	Производит предварительное шифрование данных и их дальнейшую пересылку в зашифрованной форме, а также последующую раскодировку с сохранением шифрования или без него.
process_command	Команда API	Финализированная функция, обеспечивающая передачу данных API между верхним и нижним уровнями устройства. Собирает в себе все описанные выше команды.

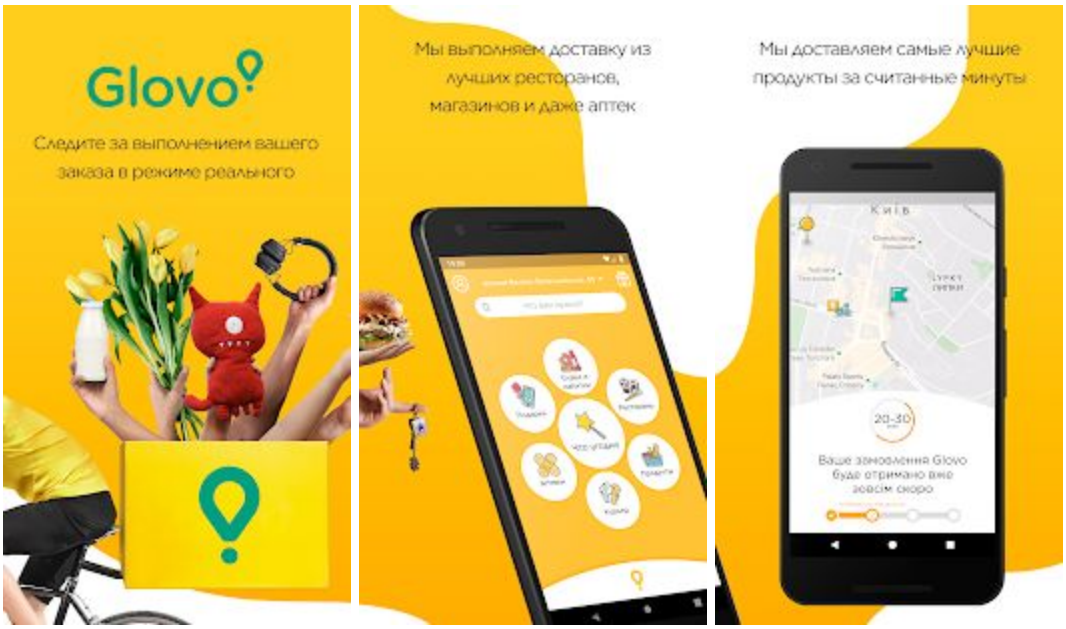
destructor	-	Выполняет очистку ресурсов памяти.

V. Медиаконтент и публикация приложения

Подготовка медиаконтента

Правила публикации приложений в магазинах приложений сегодня сводятся к идентичным наборам задач и планов подготовки подготовки медиаконтентов. Согласно набор UI and Publication Patterns середины 2018 года, корректное оформление дизайна приложений имеет следующий набор требований:

- Видеопрезентация работы и возможностей использования приложения - видеоряд до 30 секунд с демонстрацией использования, в портретном режиме.
- 2 постерных скриншота с основными свойствами
- 4 скриншота примеров использования программы
- Описание приложения с использованием “продающего” текста объемом до 6 предложений
- Сводка данных об обновлении - до 3х подпунктов в неофициальной форме.
- *Поддержка локализации всех языков поддерживаемых регионов (пример - Украина - русский и украинский языки)*



Публикация в Google Play

Запрос для компании Live Typing, в лице получателя/разработчика Александра Мирко, предоставила алгоритм корректной подготовки и публикации приложения в магазине приложений Google Play:

1. Создать аккаунт в Google Play Developer Console для заказчика, если у заказчика такового нет, или предложить произвести публикацию с нашего аккаунта.
2. Оформить privacy policy.
3. Подготовить маркетинговые материалы (иконка, скриншоты, APK, баннер, текст, проморолик).

4. Обеспечить сборку наличием сертификата цифровой подписи.
5. Настроить оплату за пользование приложения.
6. Отправить сборку в Google Play.
7. Отдельно оговариваются кейсы публикации Альфа и Бета-версии, а также внедрение ASO.

Отдельное внимание стоит уделить пользовательскому соглашению Google и его основным пунктам:

- вы полностью отвечаете за ваш продукт и поставляемый в нём контент;
- вы обязуетесь отвечать на вопросы пользователей в течении трёх рабочих дней и на «срочные вопросы согласно определению Google» в течении 24 часов;
- обязуетесь сохранять конфиденциальность и безопасность пользовательских данных;
- вы не пытаетесь обманывать, причинять какой-либо вред или вводить в заблуждение пользователя и компанию Google;
- вы не распространяете запрещённый контент. Все Продукты, распространяемые через Google Play, должны соответствовать Правилам программы для разработчиков;
- вы разрешаете Google возвращать покупателю полную стоимость Продукта или транзакции внутри приложения от вашего имени, если покупатель запрашивает возврат средств в любой момент после покупки.

Удаление продукта не освобождает вас от ответственности перед какого-либо рода выплатами;

- в целом, Google снимает с себя любую ответственность, связанную с вашим продуктом

Требования к тексту:

- название приложения: не более 30 символов;
- короткое описание: не более 80 символов;
- короткое описание: не более 80 символов;
- полное описание не более 4000 символов.

Требование к скриншотам:

- формат JPEG или 24-битный PNG (без альфа-канала);
- не менее 320 пикселей;
- не более 3840 пикселей;
- соотношение сторон не должно превышать 2:1.

Требования к иконке:

- 32-битный PNG (с альфа-каналом) мы делаем всегда без альфа-канала;

- размеры: 512 x 512 пикселей;
- максимальный размер файла: 1024 КБ

Требования к проморолику:

- указывайте URL отдельного видео на YouTube, а не плейлиста или канала;
- не используйте видео с возрастным ограничением в качестве проморолика;
- используйте полную ссылку на видео YouTube вместо сокращенной.

Требования к файлу:

- Размер apk-файла не должен превышать более 100 Мб (и 50 Мб для Android 2.2 и ниже, или для Play Market 5.2 и ниже, но давайте уже про них забудем).
  Бывает, что ваше приложение работает на статическом контенте (не делайте так) или является игрой и его размер больше 100 Мб. Такое приложение можно разбиться на части: основная — до 100 Мб и несколько дополнительных APK Expansion Files до 2 Гб каждый;
- apk-файл не должен быть debuggable;
- apk-файл должен быть подписан файлом цифровой подписи (см. Обеспечение сборки наличием цифровой подписи).

Альфа- и бета-тестирование позволяет опробовать ваше приложение на узком круге пользователей. Существует открытое и закрытое тестирование. В обоих случаях тестовая группа пользователей не сможет оставлять публичные отзывы в Google Play — только личные, которые не являются общедоступными. В любом случае полезно предложить им дополнительный общий канал для обратной связи. Стоит отметить, что неважно, публикуете вы приложение впервые, обновляете существующее или меняете его описание, изменения будут доступны в Google Play не мгновенно, а только через пару часов.

Внедрение версий

- Откройте Google Play Developer Console.
- Выберите приложение.
- В меню слева откройте раздел Управление версиями.
- Рядом с названием нужной версии нажмите Продолжить.
- Просмотрите проект выпуска и при необходимости внесите изменения.
- Выберите Посмотреть. На открывшейся странице можно убедиться, что ничего не мешает выпустить версию приложения для пользователей.
- Просмотрите все предупреждения и сообщения об ошибках.
- Для запущенных продуктов укажите процент внедрения версии. Если вы выпускаете рабочую версию впервые, эта настройка будет недоступна.
- Выберите Подтверждение внедрения версии. Если вы выпускаете приложение впервые, оно будет опубликовано для всех пользователей Google Play в выбранных вами странах.

Публикация в магазине Apple

Отдельное внимание уделяется ASO - AppStore Optimization - отдельная сводка гайдлайнов, которые предполагают публикацию в магазине приложения Apple. Ниже приведен список ключевых аспектов, которые предполагают процесс оптимизации:

Ключевые слова

- Название;
- Описание;
- Локализация.

Внешний вид

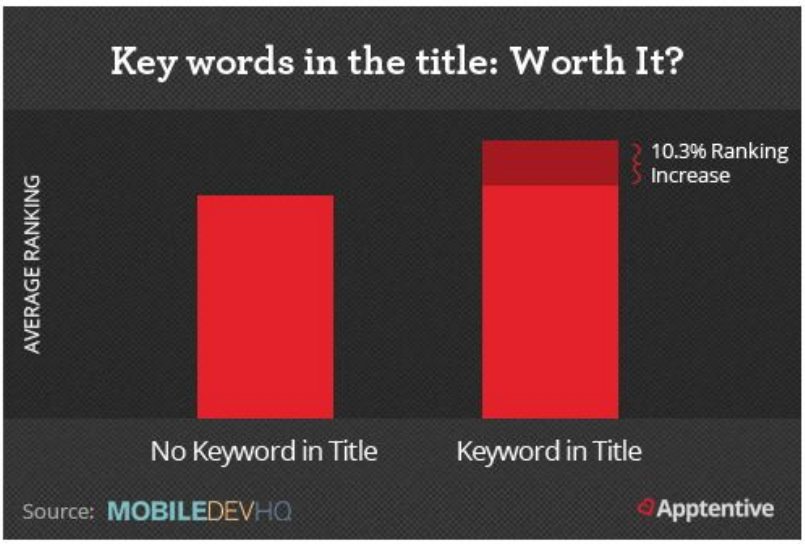
- Иконки;
- Скриншоты.

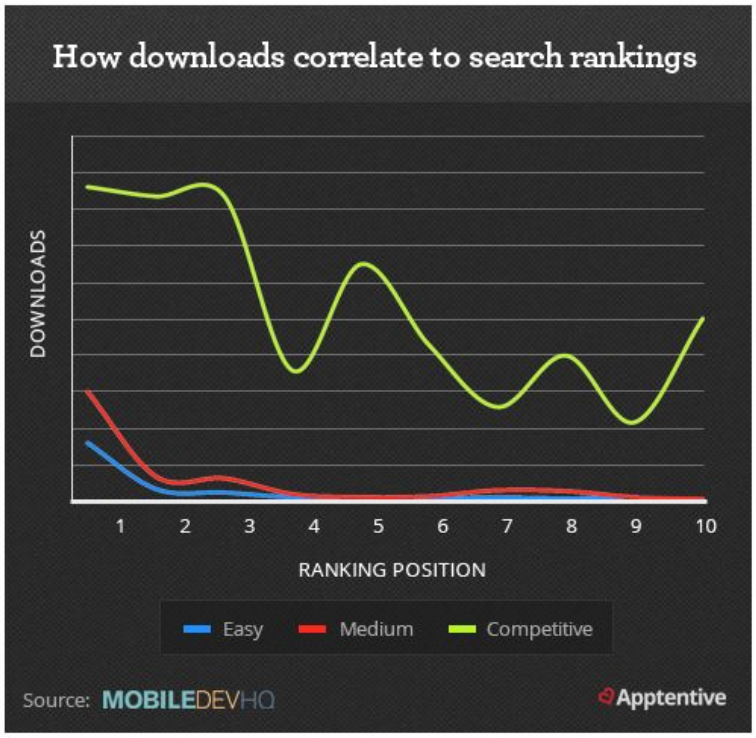
Внешние факторы

- Количество загрузок;
- Рейтинги и отзывы;
- Отслеживание и оптимизация ключевых слов.

Отдельно внедряются правила задействования ключевых слов:

- Не ставьте пробелы после запятых, этот список не увидят конечные пользователи, а движок магазина работает и без пробелов;
- Повторение ключевых слов не имеет смысла;
- Не пишите числительные, указывайте цифру (не «три», а 3);
- Старайтесь использовать короткие ключевые слова в различных комбинациях;
- Не используйте множественное число;
- Не используйте вспомогательные слова вроде “the”, “on”,”at” и их аналоги в других языках;
- Используйте все 100 символов.




How downloads correlate to search rankings
DOWNLOADS
1
2
3
4
5
6
7
8
9
10
RANKING POSITION
Easy
Medium
Competitive
Source: MOBILEDEVHQ
Apptentive


Enjoying Circa News?
Not really
Yes!
How about a rating on the App Store, then?
No, thanks
Ok, sure
Would you mind giving us some feedback?
No, thanks
Ok, sure

Как результат, публикация в Google Play должна выполняться первоочередно - это позволяет отследить тенденции рынка, а также проследить оптимизацию приложения на предмет наличия продающих элементов и соответствие правилам публикации, политики конфиденциальности, пользовательского соглашения. После этого, производится внедрение ASO и повторный аудит приложения на соответствие требованиям рынка. И, наконец, последним этапом является публикация приложения в AppStore - а также дальнейшее отслеживание на предмет сопоставления с ключевыми обновлениями требований пользовательских сервисов.

Контакты

Исследование подготовили: Глушко Б.С., Котик В.В.

Контакты:

Телефон: +380674072130

Skype: cappuccino20001

EMail: bg@z-digital.net, glushko.bohdan@gmail.com