Оглавление

**Введение**

**Основная часть**

**Этап 1. Проработка идеи:**

1.1. Краткое описание идеи  
1.2. Анализ рынка  
1.3. Анализ конкурентов  
1.4. Анализ внешней среды  
1.5. Выделение целевой аудитории  
1.6. Описание продукта  
1.7. Построение стратегии развития продукта  
1.8. MVP и карта жизненного цикла продукта  
1.9. Расходы  
1.10. Unit-экономика  
1.11. Собираем все Lean Canvas

**Этап 2. UX/UI:**

2.1. Построение навигации приложения  
2.2. Наброски экранов  
2.3. Дизайн приложения

**Этап 3. Разработка:**

3.1. Определение стека и архитектуры приложения  
3.2. Выделение сервисов в приложении

3.3. Написание кода

**Этап 4. Аналитика приложения:**

4.1. Падения приложения  
4.2. Аналитические ивенты

**Этап 5. Продвижение:**

5.1. Построение клиентского пути (Customer Journey Map - CJM)  
5.2. Написание коммуникационной стратегии

**Этап 6. Развитие:**

**Заключение**

**Введение**

Сегодня математика составляет основу выражения современных научных знаний, а в основе математики лежат числа и операции сними, попросту говоря, вычисления. По-английски computing. Соответственно, устройство способное производить вычисления, звучит как computer. Отсюда и происходит знакомое нам русское слово **компьютер**. То есть, компьютер – это попросту вычислительная машина.

Первые попытки создать механические вычислительные машины предпринимались с незапамятных времен. Первой документально зафиксированной в истории считается машина Вильгельма Шиккарда, созданная в 1623 году. Однако есть предположения, что еще до этого в 16 веке известный изобретатель Леонардо да Винчи предпринимал попытки создания подобных механизмов. Удалось ли ему это или нет – доподлинно неизвестно. Но, наверное, самой известной счетной машиной является арифмометр Блеза Паскаля, созданный в 1645 году. И более совершенная конструкция, сделанная Готфридом Вильгельмом Лейбницем. Его счетная машина была способна не только складывать и вычитать, но также делить и умножать.

Однако все эти механические счетные машины автоматизировали одно-два действия с двумя числами: складывали, умножали, делили. Более продвинутую идею в 19 веке предложил Чарльз Беббидж сначала в виде **разностной машины** (1822 год), а позже – **аналитической**. Именно в аналитической машине у него появляется идея программного управления вычислениями. Правда, дальше идеи дело не пошло. Сложность построения его машины превосходила технологии того времени. Но с ней связан один интересный исторический факт. Когда леди Августа Ада Лавлейс переводила конспект лекций Беббиджа по устройству его аналитической машины на английский язык, то сопроводила перевод своими многочисленными комментариями, среди которых оказался полный набор команд для вычисления чисел Бернулли с помощью аналитической машины. Этот набор команд считается первой документально зафиксированной программой в истории человечества, а Ада Лавлейс – первым программистом.

Но все эти механические устройства, в большинстве своем, были громоздкими, неудобными и ненадежными. Поэтому оставались на уровне идей или единичных воплощений. Новый виток развития вычислительных машин начался в XX веке благодаря зарождению электроники. Одним из основоположников современной вычислительной техники стал английский математик Алан Тьюринг (1912 – 1954 гг.). В 1936 году он представил общую идею абстрактной универсальной вычислительной машины, которая выполняла различные задачи в зависимости от загруженных в нее данных. Вскоре, эта идея получила название **универсальной машины Тьюринга**. Затем эту идею развил венгерско-американский математик Джон фон Нейман (1903 – 1957 гг.). Он предложил команды и данные хранить в единой, однородной памяти компьютера. До этого команды следовало записывать в один тип памяти, а данные независимо в другую. Это приводило к заметным неудобствам. Поэтому идея фон Неймана по объединению команд и данных в единой памяти оказалась прорывной для того времени. Она стала настолько удачной, что сейчас практически вся вычислительная техника построена по этому принципу – по архитектуре, предложенной Джоном фон Нейманом.

Первое же поколение компьютеров (с 1944 года), построенных на электронных лампах, использовали принцип однородности памяти для данных и команд. Уже тогда сразу была принята на вооружение архитектура фон Неймана. И вычислительная техника, созданная по этому принципу, получила название **машины фон Неймана**.

Второе поколение компьютеров (с середины 1950-х) было выполнено на транзисторах. Схемотехнически, это те же лампы, но реализованные на уровне полупроводников. В результате вычислительная техника стала гораздо меньших объемов, потребляла меньше энергии, стала надежнее и с большим быстродействием. Внутреннее устройство компьютеров этого и всех последующих поколений уже базировалось на архитектуре фон Неймана.

Третье поколение компьютеров (с 1960-х годов) были разработаны на базе интегральных схем (микросхем), которые объединяли внутри себя схемы из большого числа транзисторов. Это привело к еще большему сокращению размеров компьютерной техники, повышению надежности и быстродействию.

Четвертое поколение вычислительной техники (с 1970-х годов) ознаменовано появлением специального типа микросхем, известных, как микропроцессор. Эта микросхема выполняла внутри себя все арифметические, логические операции и операции управления, записанные в машинном коде. Это дало новый скачок в развитии компьютерной техники. Благодаря миниатюризации и энергоэффективности компьютеры стали доступны массовому пользователю.

Практически всю современную вычислительную технику можно отнести к четвертому поколению. Конечно, она стала намного быстрее, с более сложным внутренним устройством, разнообразнее, но ничего принципиально нового в ней не появилось. Также присутствует центральный процессор, несколько вспомогательных процессоров, память, различные внешние и внутренние устройства, и все это объединено по архитектуре, предложенной Джоном фон Нейманом.

**Архитектура современного компьютера**

Если внимательнее посмотреть на общее устройство компьютера, то мы увидим:

* центральный процессор (CPU – central processing unit), который непрерывно выполняет поступающие в него команды;
* оперативное запоминающее устройство (ОЗУ, англ. RAM – random access memory), в котором можно хранить данные и команды для процессора;
* шину, соединяющую центральный процессор, память и другие дополнительные устройства, соединенные через контроллеры.

Как видите, процессор взаимодействует с другими компонентами через контроллеры. Почему это так сделано? Почему бы все эти устройства напрямую не подключать к шине? Зачем потребовался посредник в виде контроллера? Дело в том, что процессор понятия не имеет обо всех этих устройствах: ни о клавиатуре, ни о мониторе, ни о жестких дисках. У него нет инструкций, как, например, управлять считыванием информации с клавиатуры или с жесткого диска, или как выводить что-либо на монитор. Все, что умеет делать процессор – это выдавать определенные данные в шину и читать их оттуда. Именно так происходит взаимодействие с контроллерами: центральный процессор указывает адрес обращения к контроллеру, данные, и команды для контроллера. На основе принятых данных и команд контроллер выполняет определенные действия, например, указывает монитору вывести определенную информацию. И так с каждым устройством: клавиатурой, жестким диском и т.д.

Так как в шину процессор передает разные типы информации: данные для контроллера, его адрес и команды, то шина имеет соответствующие отдельные каналы, которые так и называют: **шина данных**, **шина адресов**, **шина управления**. Благодаря подключению устройств через контроллеры центральный процессор получает возможность взаимодействовать с ними на универсальном уровне. Если в будущем появится новый тип внешних носителей, например, какие-нибудь квантовые флэшки, или био-принтеры или что-либо еще, то это не потребует схемотехнического изменения самого процессора. Через контроллеры все новые устройства смогут, по-прежнему, обмениваться информацией с центральным процессором и полноценно выполнять свои функции.

Правда, из-за этого появляется еще один нюанс: процессор должен знать, какие команды, какие данные и в какой последовательности следует передавать на контроллеры, чтобы внешние устройства корректно ее принимали и обрабатывали. Для этого разработчики устройств совместно с ними поставляют специальные программы, которые называются **драйверами**. Центральный процессор для взаимодействия с тем или иным устройством просто выполняет соответствующую программу-драйвер и, таким образом, передает нужному контроллеру корректные управляющие команды. Так, в целом, происходит взаимодействие между процессором и любым внешним устройством.

**Память и выполнение команд центральным процессором**

Давайте теперь посмотрим, что из себя представляет память. Практически во всей современной вычислительной технике, память можно представить в виде последовательностей упорядоченных ячеек, каждая из которых имеет свой физический номер (**физический адрес**). А сама ячейка, почти во всех схемотехнических реализациях, представляет собой восемь бит информации. Напомню, что один бит кодирует информацию двумя состояниями, которые, обычно, обозначаются числами 0 и 1. Число 0 – бит выключен; 1 – бит включен. Соответственно группа из восьми таких бит способна кодировать:



различных вариантов. Например, целые числа в диапазоне от 0 до 255. Такая неделимая единица информации из 8 бит получила название **байт**. То есть, каждую ячейку памяти можно интерпретировать как один байт, в которой хранится целое число от 0 до 255.

Центральный процессор способен через шину заносить в любую ячейку оперативной памяти некоторое значение в этом диапазоне [0; 255], либо осуществлять запрос на получение значения также из любой существующей ячейки памяти. Но что в итоге нам это дает? Очевидно то, что в такой памяти можно хранить и команды для центрального процессора и данные. Причем, как команды, так и данные кодируются обычными числами.

Для определенности предположим, что в оперативной памяти, начиная с 1000-й ячейки, хранятся следующие числа (приведены в шестнадцатиричной записи):

B8 22 11 00 FF 01 CA 31 F6 53 8B 5C 24 04 8D 34 48 39 C3 72 EB C3

И пусть компьютер оснащен 32-разрядным процессором архитектуры x86. Чтобы на вход процессора поступило первое число B8, расположенное по адресу 1000, **счетчик команд** этого процессора должен быть равен 1000. Да, внутри каждого процессора есть специальные внутренние хранилища данных, которые называются **регистрами**. В этих регистрах сохраняется промежуточная, вспомогательная информация для обработки текущей информации. Доступ к регистрам для чтения или записи (если это допустимо) выполняется очень быстро (быстрее, чем обращение к оперативной памяти). Так вот, один из регистров процессора – это счетчик команд (program counter) или его еще называют **указателем инструкции**(instruction pointer). Он содержит адрес ячейки памяти, в которой хранится следующая для выполнения команда. В нашем примере – это число

После этого счетчик команд увеличивается на 2 и ссылается на следующую команду 31. И так последовательно, такт за тактом происходит выполнение команд процессором в бесконечном цикле. При этом все команды могут быть представлены обычными числами и записаны в оперативной памяти, наряду с данными. Но, благодаря регистру счетчику команд (или указателя инструкций), процессор точно знает откуда брать следующую команду на выполнение и не путает команды с данными. Мало того, центральный процессор никогда не отдыхает, не останавливается, а работает постоянно по непрерывному **циклу обработки команд**, выполняя все новые и новые инструкции, пока компьютер не будет выключен. Скорость обработки команд определяется **тактовой частотой** процессора. Например, тактовая частота 1 ГГц означает выполнение одного миллиарда команд в секунду. А размер порции данных, обрабатываемых процессором за одну операцию, называют **машинным словом**. Как правило, размер машинного слова совпадает с разрядностью процессора. Например, для 32-разрядного процессора машинное слово обычно 32 бита; для 64-разрядного – 64 бита.

**Все данные в памяти компьютера – это числа**

Итак, получается, что в каждой ячейке оперативной памяти можно хранить только числа, например, в диапазоне от 0 до 255. И центральный процессор способен обрабатывать исключительно числовые данные. Спрашивается, как же тогда с помощью вычислительной машины работать с текстами, изображениями, звуками, да и любыми другими не числовыми данными? Выход только один – преобразовать их в набор чисел. Например, процессор компьютера понятия не имеет, что такое символ или буква. Вместо нее в памяти сохраняется ее кодовое представление в соответствии с используемой кодовой таблицей. В соответствии с ней, например, строка «I love C» будет представлена последовательностью чисел:

73 32 108 111 118 101 32 67

Это ее представление в памяти, и для процессора, а на экране, при использовании кодовой таблицы ASCII, мы увидим соответствующую строку.

Подобные преобразования выполняются и для всех других типов данных. В частности, изображения – это наборы отдельных точек (пикселей), цвет которых определяется числовым значением в соответствии с цветовой палитрой:

И вообще, любые данные, предназначенные для компьютерной обработки, предварительно нужно перевести в набор чисел (говорят **оцифровать**) и только после этого использовать в вычислительной технике. И, наоборот, такие устройства, как мониторы, принтеры, звуковые колонки и т.д. преобразовывают цифровую (то есть, числовую) информацию обратно в исходный, привычный для нас вид. Так происходит круговорот цифровой информации в вычислительной технике.

**Операционная система**

Но вернемся к вопросу взаимодействия между процессором и другими узлами компьютера (оперативной памятью, внешними устройствами). Как мы увидели, это происходит не таким уж и тривиальным способом. И если бы перед нами поставили задачу написать, например, программу для вычисления обратных матриц с выводом результата на экран, имея только «железо» в виде центрального процессора, памяти, клавиатуры, дисплея, жесткого диска, нам было бы очень непросто ее выполнить. Пришлось бы вначале прописывать на программном уровне взаимодействие процессора с дисплеем, клавиатурой, жестким диском. Затем, порядок запуска и выполнения программы вычисления матрицы, контролировать цикл обработки команд процессора, так как он работает непрерывно, и решать еще очень и очень много других вспомогательных задач, не относящихся непосредственно к поставленной перед нами начальной цели. Это жутко неудобно! Но выход из этой ситуации есть и он очевиден. Нужно написать программную оболочку, которая бы брала на себя управление всеми периферийными устройствами, памятью и пользовательскими программами. И такая оболочка представляет собой не что иное, как **операционная система**. Как правило, она автоматически загружается при включении компьютера и предоставляет простой, универсальный доступ ко всем имеющимся его ресурсам посредством специальных программ, называемых драйверами. А нам, как прикладным программистам, остается только обратиться к установленной операционной системе и запросить, например, нужный объем памяти для хранения данных, или «попросить» вывести некоторую информацию на экран, или открыть файл на внешнем носителе и занести туда какие-либо данные. И так далее. Все эти операции берет на себя операционная система. Нас не интересует, как конкретно она это делает, главное, чтобы предоставляла доступ к нужным ресурсам и максимально быстро выполняла указанный запрос. Такие запросы к ОС со стороны пользовательских программ называются **системными вызовами**. Именно через системные вызовы происходит взаимодействие со всем многообразием внешних устройств, подключенных к компьютеру, а также выполняются некоторые внутренние инструкции, за которые отвечает операционная система, например, выделение памяти под нужные нам данные.

Помимо системных задач ОС предоставляет набор команд для управления состоянием компьютера для оператора – человека. Например, мы можем посмотреть список файлов на жестком диске, или скопировать информацию на флэшку, или установить пароль доступа к компьютеру, и многое другое. Однако не стоит полагать, что основное назначение ОС – это взаимодействие между человеком и компьютером. Совсем нет. Она в первую очередь ориентирована на корректное выполнение пользовательских программ и их взаимодействие с ресурсами вычислительной техники. Например, в современных дата-центрах можно увидеть ряды стоек с серверами (компьютерами), у которых нет ни клавиатуры, ни дисплея, так как не предполагается постоянная работа с ними человека. При необходимости, можно установить удаленное взаимодействие по сети Ethernet и через командную строку выполнить требуемые действия.

**История создания языка Си**

Вообще, первые операционные системы появились еще в 1960-х годах. Вначале они не имели графического интерфейса и все взаимодействие выполнялось на уровне команд. Позже стали появляться графические пользовательские интерфейсы, но вплоть до середины 1990-х они не были массовым явлением. Пока в 1995 году не вышла известная ОС Windows 95 со встроенным графическим интерфейсом, которая завоевала очень большую популярность. С тех пор вся линейка ОС Windows поставлялась со встроенным графическим интерфейсом и популярна по сей день.

Другое распространенное семейство ОС известно под общим названием Unix. К нему относятся различные вариации Linux, такие как Debian, Ubuntu, Fedora, Gentoo и многие другие, а также различные семейства BSD: FreeBSD, OpenBSD и так далее. Интересно, что графический интерфейс в ОС Unix реализован не на уровне ядра системы (как это сделано в ОС Windows), а на уровне пользовательской программы, которую можно поменять, если потребовалась другая графическая оболочка. Также открытый исходный код дистрибутивов Unix позволяют на их основе относительно просто создавать свои вариации операционных систем. На практике этим довольно часто пользуются. Например, так появилась первая версия ОС Android от компании Google.

Но для нас намного важнее, что именно зарождение Unix способствовало появлению первой версии языка Си. Все начиналось в далеких 1960-х годах. На тот момент некто Кен Томпсон сотрудник фирмы Bell Laboratories принимал участи в создании ОС под названием MULTICS. Проект оказался неудачным, но опыт был получен. Несколько позже, работая с довольно устаревшей даже на тот момент машиной PDP-7, Кен Томпсон решил улучшить ее системное программное обеспечение и, немного-немало, написать свою версию ОС. И он это сделал. В шутку Брайан Керниган назвал эту систему UNICS. Название пошло в народ, только последние две буквы изменились на X, получилось UNIX. Так появилась новая на тот момент ОС Unix для компьютера PDP-7. Но это была устаревшая техника, поэтому Кен Томпсон уже совместно с Деннисом Ритчи решили ее перенести на более совершенную машину PDP-11. Но для этого требовалось переписать ОС. Чтобы не выполнять всю эту кропотливую работу на уровне языка Ассемблер, то есть, по сути, на уровне машинных команд, Кен Томпсон решил воспользоваться языком более высокого уровня под названием «B» - усеченный вариант другого языка BCPL. Но этот язык оказался слишком неудобным для создания ОС. Тогда Деннис Ритчи предложил расширить (усовершенствовать) язык «B» и, недолго думая, назвал новый язык следующей буквой английского алфавита «C». Так в 1972 году появилась первая версия языка Си. Ее автором считается Деннис Ритчи – сотрудник Bell Laboratories. Именно на этом новом языке программирования «С» в 1973 году Кен Томпсон написал ОС Unix для компьютера PDP-11. А в 1974 году вышла совместная статья Кена Томпсона и Денниса Ритчи, где они подробно рассказали о своих разработках. Благодаря тому, что машина PDP-11 на тот момент была довольно распространена, то желающих попробовать новую ОС, а также язык Си, нашлось немало. С этого началось торжественное шествие и становление языка программирования Си, который остается популярным и востребованным до наших дней.

Особенность этого языка в том, что он был создан программистами для программистов в первую очередь для того, чтобы уйти с низкого ассемблерного уровня и позволить писать программы на более высоком абстрактном уровне. Но, при этом, сохраняя все преимущества низкоуровневого программирования, то есть, предоставляя программисту полный контроль и полную свободу действий. Поэтому язык Си можно отнести к самым низкоуровневым из высокоуровневых языков программирования. Благодаря этому программы, написанные на нем, эффективно переводятся в машинный код, максимально быстро работают и могут выполнять все допустимые операции, ограниченные только конкретной архитектурой компьютера. Этот язык удобен для написания ОС, игровых приложений, используется в системах дополненной реальности, искусственном интеллекте, управлении различным оборудованием, на нем реализованы автопилоты самолетов, кораблей. Также на языке Си, а точнее, на С++ написаны различные интерпретаторы для таких языков, как Python, Java, C#. То есть, везде, где нам важна скорость и/или полный контроль за процессом выполнения программы.

**Этап 1. Проработка идеи**

**1.1. Краткое описание идеи**

Если у вас уже есть идея, то важно ее зафиксировать в виде 1-2 предложений, без деталей – они будут проработаны позже. В процессе анализа идея может трансформироваться во что-то иное, и это совершенно нормально. Главное – определить начальную точку, от которой будем отталкиваться.

Мобильное приложение служит разным целям. Оно может быть основным продуктом бизнеса или дополнительным каналом привлечения клиентов. Например, мессенджер Telegram — это самостоятельное приложение-сервис, которое закрывает потребность пользователей в общении. А у супермаркета «Перекрёсток» другая ситуация: их приложение доставки второстепенно, потому что основной продукт — это розничные офлайн магазины. Однако при любой стратегии мобильное приложение приносит пользу: помогает развивать бизнес и увеличивать продажи.

Если идеи нет, то вот небольшая подсказка, где можно ее взять:

* Обратите внимание на свои или чужие проблемы, "болевые точки", недостатки в определенных аспектах жизни, которые может решить или упростить приложение.
* Устройте с друзьями или знакомыми мозговой штурм, чтобы вместе придумать что-то интересное.
* Посмотрите на самые популярные приложения и подумайте, что в них можно улучшить или какие функции добавить.
* Найдите приложения с низкими оценками, но без альтернатив в категории. Можно создать аналогичное, но лучшее.
* Попробуйте объединить функции двух или более успешных приложений.
* Найдите нишевые сообщества или специализированные отрасли, у которых нет подходящего приложения, и разработайте приложение для них.
* Посетите хакатоны и мероприятия для стартапов, там можно найти, придумать классные идеи, а еще найти единомышленников и начать работу над проектом вместе.
* Следите за общественными трендами и появлением новых технологий – это может стать основой для интересных идей.
* Наблюдайте за повседневной жизнью – идея приложения может возникнуть из простого наблюдения.

**1.2. Анализ рынка**

Прежде чем определить, будет ли идея приложения пользоваться спросом, важно определить сегмент, к которому оно относится и изучить объем и тренды рынка, а также специфику выбранного сегмента. Для получения данных лучше обратиться к компаниям, которые специализируются на аналитике приложений, таким как sensortower.com, data.ai, prioridata.com, apptopia.com и другим. Если нет возможности взять данные у них, используйте различные источники в интернете. Чем более качественные данные вы соберете, тем лучше сможете сделать анализ и скорректировать идею и сделать приложение ближе к пользователю.

**362 миллиарда долларов на мобильную рекламу**

По прогнозам ожидается, что реклама на мобильных устройствах будет занимать всё больше и больше времени пользователей. На Android за прошлом год реклама уже прокрутилась 4 триллиона часов.

Приложения с короткими видеороликами двигают рекламную индустрию в мобайле. Этот тренд переходит и в следующем год.

**Экономический кризис и изменения правил конфиденциальности повлияют на снижение расходов на рынке мобильных приложений**

В прошлом году расходы на мобильные игры сократились на 5%, в следующем ожидается снижение еще на 3%. Обычно траты пользователей в этой вертикали никак не зависели от экономических проблем. Но сейчас сильное влияние оказывают изменения в правилах конфиденциальности. Теперь разработчикам будет еще сложнее таргетироваться на пользователей, которые приносили больше всего прибыли.

**14 приложений преодолеют отметку в 2 миллиарда долларов по расходам пользователей**

11 из 14 наименований — это игры. Uma Musume Pretty Derby станет игрой, которая сделает 3 миллиарда долларов всего через два года после запуска. При этом игра доступна только в 5 странах: Япония, Южная Корея, Тайвань, Гонконг и Макао.

Потребители всё чаще обращаются к развлекательному видеоконтенту. В следующем году пользователи потратят 3 миллиарда долларов еще и в HBO Max и iQIYI. Они присоединятся к топу, где уже есть Disney+, Netflix, Youtube и TikTok.

**Пользователи готовы платить за экспириенс: путешествия, события, медитации и спорт**

Shopify выяснила, что на мобильные устройства приходится 73% продаж небольших компаний и брендов, использующих прямые продажи (англ. direct to consumer, DTC).

Путешествия пользуются популярностью после ковидного застоя. Концерты и шоу популярнее, чем покупки для дома. Онлайн-трансляции спортивных событий будут привлекать внимание пользователей и дальше. Юзеры готовы тратить деньги на все эти три категории.

А приложения для медитации становятся «модными» из-за желания людей отвлечься от стресса в этот сложный период.

**К 2028 году время, проведенное в приложениях, достигнет 6 триллионов часов**

Этому способствуют различные факторы: внедрение 5G, распространение игр, потребность в диджитал-общении, персонализация приложений.

Стриминговые сервисы и пользовательский контент продолжат набирать популярность в следующие 6 лет.

**1.3. Анализ конкурентов**

Важно определить основных конкурентов и не основных. Далее проработка идеи будет делаться с оглядкой на то, что они делают. Это поможет выявить те аспекты, которые пользователи ценят у конкурентов, и выделить уникальные черты вашего приложения, которые помогут выделить его на рынке.

Основные конкуренты — это те, кто предлагает схожие продукты или услуги, направленные на одну и ту же целевую аудиторию, и имеют сопоставимый рыночный долю.

Неосновные конкуренты — это те, кто предлагает продукты или услуги, отличающиеся от вашего приложения, но могут быть альтернативой для вашей целевой аудитории или имеют меньший рыночный долю. Также стоит учитывать новые стартапы, которые могут потенциально стать конкурентами в будущем.

Как подобрать конкурентов? Составляем базу

**1.** Определите приложения из ТОП-10 поисковой выдачи по самым популярным запросам из вашей тематики. Например, вводим запрос «Доставка зоотоваров в Москве» — и анализируем приложения из первой десятки. Рекламную выдачу не учитываем.

**2.** Изучайте «Похожие приложения», рекомендуемые сторами. Там могут быть релевантные приложения. Но нужно изучать каждое детально. Магазины могут подбирать их исходя не из сходства тематики, а пересечения аудитории.

**3.**Алгоритмы Checkaso подбирают конкурентов автоматически, основываясь на сходстве семантики приложений – ключевых запросов, по которым вас ищут юзеры.

По каким параметрам нужно сравнивать конкурентов?

* **Качество ASO и видимость приложения в магазине**

**ASO** — это комплекс мер по продвижению мобильного приложения в App Store и Google Play. ASO состоит из текстовой и графической оптимизации. Именно ASO влияет на видимость приложения в магазине. Схема простая. Работаете над ASO — выше ранжируетесь по ключевым запросам. Чтобы ваше приложение скачали, его для начала должны найти. Пока все логично. Для этого нужно правильно оптимизировать текстовые метаданные. Также важно, чтобы юзера заинтересовала графика: на этом этапе происходит конверсия из просмотра в установку. Здесь в работу вступает графическое ASO.

Быстро оценить качество ASO можно с помощью **ASOindex**. Это показатель от 0 до 10, берущий в расчет все основные ASO-параметры: количество знаков текста, число скриншотов, наличие или отсутствие видео, качество работы и видимость приложения. Также ASOindex отслеживает частоту обновлений, рейтинг, число отзывов и количество установок. Он основной и более общий. Показатель растет? Вы все делаете верно – продолжайте эксперименты.

Есть более узконаправленный показатель — **Search Visibility Index**. Он позволяет оценить именно видимость в сторе: то, насколько вы доступны для пользователей, которые ищут приложения по ключевым запросам. При низком проценте стоит проанализировать свои позиции по релевантным и серьезнее заняться текстовым ASO. Также изучите показатель конкурентов. После этого можно перейти к анализу ключей, по которым они ранжируются и позициям по ним.

* **Текстовая оптимизация**

Текстовая оптимизация — это процесс включения релевантных ключевых запросов в индексируемые текстовые элементы: название, подзаголовок, описание и другие. Они отличаются в зависимости от стора. Обязательно следите за текстовой оптимизацией конкурентов: по каким ключам и как они ранжируются, насколько успешно двигаются по позициям, как часто делают новые ASO-итерации, используют ли рекламный трафик.

* **Графическая оптимизация**

Графическая оптимизация – комплексная работа над всеми элементами графики: иконками, скриншотами, видео. Если вкратце, иконки должны быть цепляющими и понятными, а скриншоты — информативными. Обязательно отслеживайте работу конкурентов над графикой: как часто они оптимизируют визуал? Используют ли ситуативный маркетинг? Отличается ли графика в разных сторах?

Чтобы сделать качественное графическое ASO важна насмотренность и тестирования. Чем больше вы смотрите чужой графики, тем проще понять тренды и найти решения, как выделиться. Но личного вкуса не всегда достаточно. Важно проводить тестирования гипотез, чтобы выбрать самую конвертящую графику.

После определения конкурентов делается 5 сил Портера. Анализ по модели Портера позволяет оценить степень конкуренции в отрасли и выявить возможности для развития.

И заключительным этапом анализа конкурентов это построение таблицы, в которой будут представлены основные характеристики приложения конкурентов. Можно включить такие параметры, как функционал, стоимость, монетизационная модель, пользовательский интерфейс, отзывы пользователей и т.д. Это позволит сравнить преимущества и недостатки каждого продукта и выявить возможности для дифференциации своего приложения

**1.4. Анализ внешней среды**

Внешние факторы такие как экономическая ситуация, политический климат, социальные изменения, технологические тренды, пандемия и т.д. помогут прогнозировать возможные изменения в спросе и адаптировать стратегию разработки и продвижения приложения.

Для анализа тенденций изменения внешней среды фирмы используется техника анализа ПЭСТ (PEST), позволяющая более широко взглянуть на то, как влияет окружение на конкретную фирму.

PEST - Анализ - это инструмент, предназначенный для выявления политических (Policy), экономических (Economy), социальных (Society) и технологических (Technology) аспектов внешней среды, которые могут повлиять на стратегию компании. Политика изучается потому, что она регулирует власть, которая в свою очередь определяет среду компании и получение ключевых ресурсов для её деятельности. Основная причина изучения экономики это создание картины распределения ресурсов на уровне государства, которая является важнейшим условием деятельности предприятия. Не менее важные потребительские предпочтения определяются с помощью социальной компоненты PEST - Анализа. Последним фактором является технологическая компонента. Целью её исследования принято считать выявление тенденций в технологическом развитии, которые зачастую являются причинами изменений и потерь рынка, а также появления новых продуктов.

Основные положения PEST - Анализа:

“Стратегический анализ каждой из четырех указанных компонент должен быть достаточно системным, так как все эти компоненты тесным и сложным образом взаимосвязаны”

Нельзя полагаться только на эти компоненты внешней среды, так как реальная жизнь значительно шире и многообразнее.

PEST - Анализ не является общим для всех организаций, так как для каждой из них существует свой особый набор ключевых факторов.

Анализ по методике ПЭСТ должен проводиться систематически. Частота его проведения зависит от сферы бизнеса и уровня нестабильности внешней среды.

Работа над стратегией - это сложный и трудоемкий процесс. Фундаментом для выбора программы развития любого предприятия, не только коммерческого, но и общественного, муниципального, становится анализ внешней среды организации. Результативность данного исследования определяет успех перспективного планирования. В статье рассмотрены методы и инструменты анализа внешнего окружения.

Внешняя среда

Ни одна организация не может существовать в вакууме. На нее воздействуют десятки факторов извне. Особенностями «хамелеона» внешней среды, которые стоит всегда принимать во внимание, являются динамизм, непрерывное движение и изменение. Структура рынка переходит в другое состояние, у потребителей появляются новые предпочтения, «скачут» курсы валют, издаются новые законодательные акты. Если предприятие проявляет гибкость, быстроту реагирования и справляется с переменами, то оно на пути к успеху.

Анализ внешней среды организации можно условно поделить на два направления: исследование «ближнего» окружения и «дальнего».

**1.5. Выделение целевой аудитории**

Приложение делается, в первую очередь, для целевой аудитории, а во второую уже для себя. Важно понимать кто ваша целевая аудитория, чем живут и какие у них проблемы.

По прогнозам, с 2022 по 2030 год объем мирового рынка мобильных приложений вырастет на 13,4%. Более того, первым Этапом к определению вашей целевой аудитории является установление контакта путем выборки различных демографических данных, местоположений, поведения, отношения и других факторов.

Это поможет вам быть на высоте в исследовании вашей целевой аудитории и добиться большего успеха среди определенных пользователей. Следовательно, чтобы узнать свою клиентскую базу, крайне важно использовать различные инструменты и следующие Этапи, чтобы понять свою часть аудитории и спланировать лучшие маркетинговые стратегии для вашего приложения.

Определяем ядро целевой аудитории. Ядро целевой аудитории — это пользователи, которые наиболее вероятно заинтересуются вашим приложением и будут его активно использовать. Следом определяем дополнительные целевые группы, которые могут быть заинтересованы в вашем приложении, но в меньшей степени, чем ядро целевой аудитории. Это позволит расширить охват вашего приложения и увеличить его потенциальную аудиторию. Можно проанализировать смежные возрастные группы, интересы и потребности, которые могут быть удовлетворены вашим приложением.

Чтобы выделить ядро и дополнительные целевые группы, стоит учитывать такие факторы, как возраст, пол, образование, доход, географическое положение, интересы и поведенческие характеристики и др.

Во время исследования целевой аудитории важно определить сегмент аудитории, который вы принимаете во внимание. Это поможет создать точные маркетинговые планы по увеличению количества загрузок вашего приложения.

Сегменты могут быть разделены на различные типы в соответствии с вашими потребностями в адаптации

Весь рынок (когда весь рынок - это ваши предполагаемые пользователи)

Более широкий рынок (когда вашей целевой аудиторией является нечто большее, чем отдельный рынок)

Определенный сегмент (когда предполагаемый пользователь представляет собой отдельный сегмент аудитории) и

Маркетинг с более узкой аудиторией (также называемый нишевым маркетингом).

Кроме того, постарайтесь узнать свой тип аудитории и выстроить маркетинговую стратегию вокруг вашего целевого рынка, чтобы сохранить интерес вашей аудитории. И в результате вы можете увеличить количество загрузок мобильных приложений, их использование и удержание вашими предполагаемыми пользователями.

Поведение пользователей напрямую влияет на их покупку или использование; следовательно, знание их различных атрибутов может помочь вам лучше понять ответ на вопрос "как охватить вашу целевую аудиторию?".

Тактика пользователя должна учитывать различные интересы при использовании вашего приложения. Это будет основано на плане подписки. Сообщите им о проблемах, которые ваше мобильное приложение может решить с помощью ваших услуг или продукта.

Узнайте, используют ли конечные пользователи приложение для личного и профессионального использования. Таким образом, создайте свой целевой маркетинговый план, касающийся их.

Нужно изучить и описать основные проблемы/"боли" вашей целевой аудитории, которые приложение может решить. Это могут быть неудобства, с которыми сталкиваются пользователи при использовании существующих приложений, нерешенные потребности или ожидания. Выявление и понимание этих проблем позволит создать приложение, который будет пользоваться спросом у аудитории и сможет конкурировать с существующими решениями на рынке.

**1.6. Описание продукта**

Мы уже знаем наших конкурентов, и свою целевую аудиторию. Сейчас самое время дорабатывать идею, которая была записана в самом начале. Приложение должно решать "боли"/проблемы целевой аудитории. Задача описать, как функционал вашего приложения поможет решить эти проблемы и удовлетворить потребности целевой аудитории. Можно привести конкретные примеры использования продукта, как он может улучшить опыт пользователей, сравнивая с альтернативами, доступными на рынке.

Эта творческая задача и нет универсального решения как лучше делать, могу предложить один из вариантов описания продукта через описание характеристик и выгод:

* Базовые характеристики: основные функции и возможности приложения, которые обеспечивают его работоспособность и удовлетворяют базовые потребности пользователей.
* Дополнительные характеристики: дополнительные функции (не относятся к базовым или основным) или возможности, которые могут привлечь внимание пользователей и улучшить их опыт взаимодействия с приложением.
* Потребительские характеристики: функционал, направленный на удовлетворение потребностей пользователей, таких как удобство использования, интуитивный интерфейс, персонализация и т.д.
* Потребительские выгоды: преимущества, которые пользователи получают от использования вашего продукта, такие как экономия времени, улучшение качества жизни, удовлетворение социальных потребностей, решение определенных проблем и т.д.

Дополнительно можно описать выгоды и преимущества вашего приложения. Может пересекаться с методом выше, в этом ничего страшного нет, оно еще дополняет предыдущий метод:

* Потребительская ценность: это основная функциональность, которая решает главную потребность или проблему, которую призвано решить приложение.
* Ключевые ценности/выгоды: наиболее важные преимущества продукта, которые делают его привлекательным для пользователей.
* Источники ценностей: могут включать технические аспекты (например, используемые алгоритмы или производительность приложения), а также нетехнические факторы, такие как вовлечение сообщества или партнерство с другими компаниями.
* Отличительные преимущества: уникальные особенности и преимущества продукта, которые выделяют его среди конкурентов и делают его предпочтительным для пользователей.

По структуре:

Первые 1–3 предложения в описании должны максимально четко описывать идею приложения и рассказывать, какую проблему оно решает. Максимальная длина этой части 255 символов.

Если у приложения есть особые заслуги (featured on TechCrunch), о них нужно говорить.

Основной текст описания может иметь 2–3 абзаца. Здесь мы расписываем характеристики и детали.

В конце должен быть список главных функций с их четким описанием.

В самый конец описания мы помещаем секцию «что нового?» Исправили баги, добавили фичи, поменяли звездочку на сердечко — все это здесь.

Первый скрин — самый важный, здесь нужно описать основную функцию приложения. В описании к скриншотам лучше всего использовать глаголы. Именно они могут наиболее качественно объяснить пользователю, что нужно делать на конкретном экране и подталкивают его к действию. Глагол — самая сильная часть речи.

Стараясь описать приложение для апп стора как можно лучше, необходимо соблюдать следующие правила:

* SLAP — Stop, Look, Act, Purchase. Другими словами, захвати внимание пользователя используя односложные предложения с подлежащими и глаголами с самого начала. Передавая смысл просто и ясно, ты подтолкнешь пользователя к действию.
* KISS — Keep it simple stupid. Вырежь все лишние слова, в которых нет никакого смысла. Не используй жаргон, это может отпугнуть.
* WIIFM — What’s in it for me? Что пользователь получит, узнает, ощутит, скачав приложение? Какой у продукта value proposition?

Описание желательно писать от второго лица, с точки зрения того, как пользователь будет использовать продукт.

**1.7. Построение стратегии развития продукта**

Продолжаем прорабатывать идею. В этом процессе нам очень сильно поможет SWOT-анализ. Он поможет выявить сильные и слабые стороны продукта, возможности для развития, а также различные внешние угрозы, которые могут повлиять на успех приложения.

Стратегия развития продукта является его нулевым этапом жизни. Еще ничего не существует, но уже закладывается его будущее. Создание стратегии состоит из трех важных этапов:

1. Формирование видения продукта;
2. Определение целей;
3. Планирование продуктовых инициатив, т.е. формирование концепции по его развитию.

Исследовательский процесс предваряет создание любой стратегии. И продуктовая — не является исключением. Для формирования видения продукта важны любые данные о внешней среде, включая качественные и количественные исследования о потребительских аудиториях, конкурентных предложениях и рыночных трендах. Опираясь на полученные таким образом данные, маркетинговому отделу следует ответить на следующие вопросы:

• На каком географическом рынке будет продаваться продукт? Какова его целевая аудитория потребителей?

• Каким потребительским свойствам должен соответствовать продукт? Какие функции будет выполнять?

• Какой потребительской ценностью он будет обладать? Какие потребительские «боли» будет решать?

• В каком ценовом сегменте будет продаваться?

• Как будет распространяться продукт? В каких каналах распределения? Какие сопутствующие услуги будут оказываться? Какие ключевые особенности продвижения?

Совокупность ответов на эти вопросы в наложении на внешние обстоятельства рынка, полученные из исследований, образуют видение будущего продукта. Исходя из него, следующим этапом нужно определить цели, т.е. обозначить достижения по распространению продукта в определенные временные периоды. Эти цели должны быть развернуты маркетингом компании в план действий по совершенствованию продукта, модификации, оптимизации и реализации прочих инициатив.

**О цикле развития продукта**

Как было отмечено выше, цикл развития продукта начинается с определения его стратегии. Само превращение стратегических наработок в набор физических и эмоциональных характеристик товара выглядит следующим образом:

1. генерация идей нового продукта;
2. селекция идей на основе соответствия продуктовой стратегии;
3. анализ экономических перспектив продукта;
4. разработка самого продукта;
5. тестирование продукта;
6. коммерциализация, т.е. вывод его на рынок.

В реалиях данный цикл развития продукта может быть нарушен, особенно, если речь идет об инновационной разработке, когда параметры спроса и потребительских требований не могут быть определены с достоверной точностью. В этом случае отдел, ответственный за совершенствование продукции и наращивание ее функционала, может чередовать этапы тестирования, экономического анализа и разработки самого продукта между собой и неоднократно возвращаться к каждому из них в процессе доводки до стадии готовности к коммерциализации.

Надо отметить, что соответствующая стратегия, которая создается на начальном этапе, делает перечисленную последовательность цикла развития продукта взаимосвязанной между собой. Генерация идей, их отбор, анализ бизнес-перспектив и непосредственная разработка продукта — все фазы должны соответствовать заложенным требованиям.

Если ограничиться только лишь написанием пунктов в графах Strengths, Weaknesses, Opportunities, Threats, SWOT-анализ не принесет пользы, и время будет потрачено зря. После составления списка сильных и слабых сторон, важно провести анализ, сопоставляя Strengths с Weakness и Opportunities с Threats.

Следующим Этапом является написание стратегии развития приложения на основе результатов SWOT-анализа, подводя итоги анализа. При разработке стратегии стоит учитывать следующие рекомендации:

* Сосредоточиться на ключевых аспектах, которые могут принести наибольшую ценность для вашего продукта. Это поможет оптимизировать использование ресурсов и увеличить шансы на успех.
* Оценивать свои возможности, чтобы избежать чрезмерного оптимизма или пессимизма. При разработке стратегии учитывайте доступные ресурсы для реализации планов и принимайте решения, основанные на реальных возможностях.
* Учитывать изменения на рынке. Рынок постоянно меняется, и стратегия развития продукта должна быть достаточно гибкой, чтобы адаптироваться к этим изменениям и использовать новые возможности.

**1.8. MVP и карта жизненного цикла продукта**

MVP (Minimum Viable Product) представляет собой версию приложения с минимальным набором функций, которые достаточны для удовлетворения потребностей ранних пользователей и получения обратной связи для дальнейшего его развития.

Задача описать важные функции приложения, которые пойдут в первую версию приложения.

Здесь важно продумать, каким образом ваше приложение будет приносить прибыль. Стоит определиться, будет ли приложение платным или оно будет бесплатным для скачивания, но со встроенными покупками или подпиской. Если ваше приложение предполагает использование более полугода, то стоит обратить внимание на модель подписки. Кроме того, продажа эмоций также является успешным подходом: если вы сможете предложить пользователям эмоциональный опыт, это выгодно выделит ваше приложение среди конкурентов.

Следующий этап

В этом Этапе это описать жизненный цикл продукта, чтобы понять, как будет развиваться приложение после MVP.

Жизненный цикл продукта (Product Life Cycle, PLC) – это последовательность этапов, через которые проходит приложение с момента его создания и выхода на рынок до его исчезновения.

Основные этапы жизненного цикла продукта:

* Внедрение (Introduction): Продукт впервые появляется на рынке, активно проводится маркетинг и продажи. На этом этапе прибыль может быть низкой из-за начальных затрат на разработку и продвижение.
* Рост (Growth): Продукт начинает активно завоевывать рынок, увеличивается численность пользователей, и прибыль начинает расти. На этом этапе важно оптимизировать процессы, расширять рынок и улучшать продукт.
* Зрелость (Maturity): Рост продаж замедляется, продукт достигает пика своей популярности и прибыли. На этом этапе необходимо оптимизировать затраты, удерживать клиентов и искать новые возможности для роста, например, путем модификации продукта или расширения ассортимента.
* Насыщение (Saturation): на этом этапе продажи стабилизируются, и рост начинает замедляться из-за насыщения рынка, усиления конкуренции или изменения потребностей пользователей. Важно мониторить изменения во внешней среде, адаптировать маркетинговые стратегии и искать новые ниши или возможности для продления жизненного цикла продукта.
* Спад (Decline): Продажи и прибыль продукта начинают снижаться из-за появления новых конкурентов, устаревания продукта или изменения потребностей пользователей. На этом этапе компания должна принять решение о том, стоит ли инвестировать в обновление или модификацию продукта, или же сосредоточиться на разработке новых продуктов и постепенно выводить текущий продукт с рынка.

**1.9. Расходы**

Расчет расходов поможет понять какие затраты потребуются для реализации приложения. Далее я опишу общими словами какие виды расходов бывают, чтобы было понимание как заполнять таблицу расходов.

* Переменные расходы: это расходы, которые прямо зависят от объема производства товаров или услуг.
* Постоянные расходы: это расходы, которые не зависят от объема производства и остаются постоянными вне зависимости от того, сколько продукции было произведено.
* Прямые расходы: это расходы, которые прямо относятся к себестоимости каждого конкретного изделия или услуги.
* Косвенные расходы: это расходы, которые нельзя напрямую отнести к конкретному продукту.
* Переменные прямые расходы: это расходы, которые непосредственно связаны с производством и имеют прямую зависимость от объема произведенной продукции, они изменяются в зависимости от объема производства и непосредственно влияют на себестоимость каждой единицы продукции. К этим расходам можно отнести: сырье и материалы, сдельная оплата труда и т.д.
* Переменные косвенные расходы: это те расходы, которые связаны с производственным процессом, но их нельзя напрямую причислить к себестоимости конкретного продукта. Они изменяются в зависимости от объема производства, но распределяются между всеми единицами продукции и они распределяются между всеми производимыми единицами. К этим расходам можно отнести: комиссионные продаж, стоимость упаковки, транспортировки и хранения товаров, рекламные расходы, расходы на аутсорсинг, аренда сервера и т.д.
* Постоянные прямые расходы: это расходы, которые непосредственно связаны с производством и не изменяются в зависимости от объема произведенной продукции. Они фиксированы и имеют прямое влияние на себестоимость каждой единицы продукции. Однако стоит отметить, что классическое деление на прямые и косвенные расходы, как правило, не применяется к постоянным расходам, поскольку они относятся к косвенным издержкам. К этим расходам можно отнести: фиксированные зарплаты линейного персонала, аренда специфического оборудования и т.д.
* Постоянные косвенные расходы: это расходы, которые не зависят от объема производства и не могут быть напрямую причислены к себестоимости конкретного продукта. Они фиксированы вне зависимости от количества производимых единиц и распределяются между всеми продуктами и они не могут быть напрямую причислены к себестоимости каждой единицы продукции. К этим расходам можно отнести: амортизация и износ оборудования, аренда производственных помещений, зарплата административного персонала, страховые взносы, налоги и сборы, расходы на связь, интернет и офисные принадлежности.

**1.10. Unit-экономика**

Unit economics (единичная экономика) является важным инструментом для анализа и предсказания финансовых результатов бизнеса, даже если продукт еще не существует. Она поможет спрогнозировать какая должна быть прибыль и затраты чтобы приложение окупалась.

AC (Acquisition Cost) - Маркетинговые затраты  
UA (Units Acquisition) - Количество привлеченных пользователей  
C1 (% Conversion rate) - Процент пользователей доходят до покупки  
AOV (Average Order Value) - Средний чек  
APC (Average Payments Count) - Среднее число покупок на одного пользователя  
COGS (Cost of Good Sold) - Издержки на момент покупки  
1sCOGS (First Sale Cost of Good Sold) - Издержки на момент первой покупки  
fix COGS (Fix Sale Cost of Good Sold) - Постоянные издержки  
CAC (Customer Acquisition Cost) - Стоимость привлечения пользователя  
CLTV (Customer Lifetime Value) - Валовая прибыль на пользователя  
LTV (Lifetime Value) - Валовая прибыль на юнит масштабирования (прибыль, которую компания получает от одного пользователя за весь период его взаимоотношений с бизнесом)  
PPPU (Profit Per Paying User) - Прибыль, полученная от каждого платящего пользователя  
ROMI (Return on Marketing Investment) - Прибыль которую получили в результате маркетинговых инвестиций  
CM (Contribution Margin) - Маржинальная прибыль  
EBITDA (Earnings Before Interest, Taxes, Depreciation and Amortizationn) - Чистая прибыль

Заполняем своими данными AC, UA, C1, AOV, APC, COGS, 1sCOGS, fix COGS, остальное высчитывается по формулам:

CAC = AC / UA  
CLTV = (AOV - COGS) × APC - 1sCOGS  
LTV = CLTV × C1  
PPPU = (CAC / C1) + (AOV - COGS) \* APC  
ROMI = (LTV - CAC) / CAC x 100%  
CM = UA × (CLTV × C1 - LTC)  
EBITDA = UA \* (C1 \* (AOV - COGS) \* APC - CAC) - (UA \* C1 \* 1sGOGS) - fix COGS

Unit-экономику можно считать по-разному, и это один из вариантов.

**1.11. Собираем все Lean Canvas**

Завершающим этапом 1 Этапа, собрать все что мы сделали в одном месте. Для этого очень хорошо подходит Lean Canvas. Это адаптированная версия бизнес-модели Canvas, созданная Эшем Маурьей (Ash Maurya) для использования стартапами и предпринимателями.

**Lean Canvas** является одностраничным шаблоном, который помогает лаконично описать идею и скомпоновать всю важную информацию о ней.

Прототип шаблона был разработан предпринимателем и аналитиком Александром Остервальдером. Он предположил, что любая бизнес-модель имеет 9 параметров. Благодаря этому шаблону стало возможным визуализировать будущее продукта и фазы работы над ним. Однако в нем не учитывались потребности и проблемы клиента, которые должен был решать создаваемый продукт.

В 2010 году предприниматель и бизнес-стратег Эш Маурья доработал шаблон Остервальдера, приняв за основу принципы метода Lean Startup. Маурья выделил 9 блоков:

1. Сегменты клиентов.

2. Проблема и альтернативы ее решения, существующие на рынке.

3. Уникальное ценностное предложение.

4. Решение.

5. Пути продвижения.

6. Источники доходов.

7. Структура расходов, издержек.

8. Ключевые метрики.

9. Скрытое преимущество.

Лучше всего выделить время и сконцентрироваться на заполнении всех 9 разделов шаблона, не отвлекаясь на другие задачи. Работать над описанием всех пунктов следует вместе с командой.

Для полного понимания идеи продукта рекомендуем следовать нумерации блоков шаблона. Шаблон можно заполнять на компьютере или в бумажном виде — как удобнее вам.

Пример последовательности заполнения шаблона Lean Canvas

**1. Выделяем сегменты клиентов**

При разработке продукта важно четко определить сегмент аудитории, для которого он предназначается. Для этого ответьте на такие вопросы:

* Кто является вашим основным клиентом?
* Каков портрет предполагаемого покупателя вашего продукта?
* Продукт будет использоваться покупателем или его будут применять для других пользователей?
* Есть ли у вас группа клиентов, которые смогут протестировать MVP?

Если продукт все же предназначен для нескольких сегментов клиентов, стоит создать отдельный шаблон для лучшего понимания потребностей каждого из них.

В качестве примера поразмышляем над сегментами ЦА приложения вызова такси. В этом случае легко выделить такие сегменты:

1. Жители города без собственного автомобиля.

2. Путешественники, туристы, временно пребывающие в городе.

Первый сегмент можно разделить на небольшие группы, например, тех, кто совершает короткие поездки и дальние, по городу и за город. Далее будет удобно проанализировать каждую из групп.

Вы также можете сегментировать аудиторию по географическому признаку. Например, разработка приложения вызова такси потребует изучения статистики заказов в разных городах или районах одного города.

Точная сегментация ЦА позволит легче найти проблемы клиентов.

**2. Определяем проблему**

Любой сегмент ЦА имеет общую проблему, «болевые точки». Обращаясь к тому же примеру с приложением вызова такси, можем проиллюстрировать проблемы людей, которым сложно передвигаться по Москве. Это может происходить по следующим причинам:

1. Долго ехать на автомобиле из-за пробок.

2. Сложно найти парковку в центре города.

3. Сложно спланировать маршрут и рассчитать время при передвижении на общественном транспорте.

4. Многие люди испытывают дискомфорт, когда в автобусе или метро находится большое количество пассажиров.

Для решения этих проблем клиенты могут использовать аренду велосипеда или самоката от ваших конкурентов. Поэтому стоит подумать об альтернативных решениях этих проблем.

Тому или иному сегменту аудитории не нужен ваш продукт, так как у них нет этих болевых точек.

**3. Разрабатываем уникальное ценностное предложение — UVP**

UVP поможет уточнить, для какого сегмента аудитории создан ваш продукт и почему клиентам стоит выбрать именно вас.

Поразмышляйте о том, что выгодно может выделить вас среди конкурентов и альтернатив. Для этого ответьте на вопросы:

* В чем заключается выгода от приобретения продукта?
* Почему покупатели должны предпочесть ваш продукт, а не конкурентов?
* Достаточно ли ценен продукт, чтобы за него заплатить?

**4. Предлагаем решение**

Напротив каждой найденной проблемы аудитории укажите возможные варианты ее решения. При отсутствии идей привлекайте коллег. Вы также можете узнать это у самих клиентов путем проведения опросов и исследований.

**5. Ищем каналы продвижения**

Определение эффективных каналов продвижения позволит клиентам узнать о вас и вашем продукте. Для этого выясните, какими способами ваш идеальный клиент получает новую информацию и как может познакомиться с вашим предложением:

* Какие социальные сети он предпочитает?
* Какие сайты он регулярно просматривает?
* Смотрит ли телевизор?
* Открывает ли рассылки на электронной почте?
* Какие места посещает в реальной жизни, принимает ли участие в различных событиях?

В этом вопросе может помочь разделение каналов на 3 вида:

1. Потребление контента перед приобретением (рекомендации знакомых, мессенджеры, PR-статьи).

2. Изучение продукта во время приобретения (информация на вашем сайте, презентации).

3. Возможность дать обратную связь после приобретения (форма отзыва на сайте, проведение опросов, взаимодействие в социальных сетях).

**6. Определяем источники финансирования и прибыли**

Для любого проекта важно определить источники финансирования и получения дохода. Для этого нужно найти ответы на следующие вопросы:

* Какие факторы повлияют на стоимость продукта?
* Откуда будет поступать прибыль при бесплатном распространении продукта?
* Как установить оптимальную цену продукта при платном распространении?
* Готовы ли вы сотрудничать с другими компаниями для их продвижения в вашем продукте?

**7. Оцениваем издержки**

В шаблоне также важно указать переменные и постоянные расходы. К переменным затратам относятся разработка сайта, рекламные компании и др. В постоянные расходы может входить аренда, налоги и др.

**8. Перечисляем ключевые метрики**

Ключевые метрики помогут определить, насколько идея успешна. Они выражаются в определенных показателях, которые свидетельствуют о положительных результатах за конкретный период. В качестве ключевых метрик могут выступать количество посещений сайта, заказов, скачиваний и др.

Метрики позволяют наметить точки роста компании, а также увидеть, что существуют определенные проблемы, которые стоит решить.

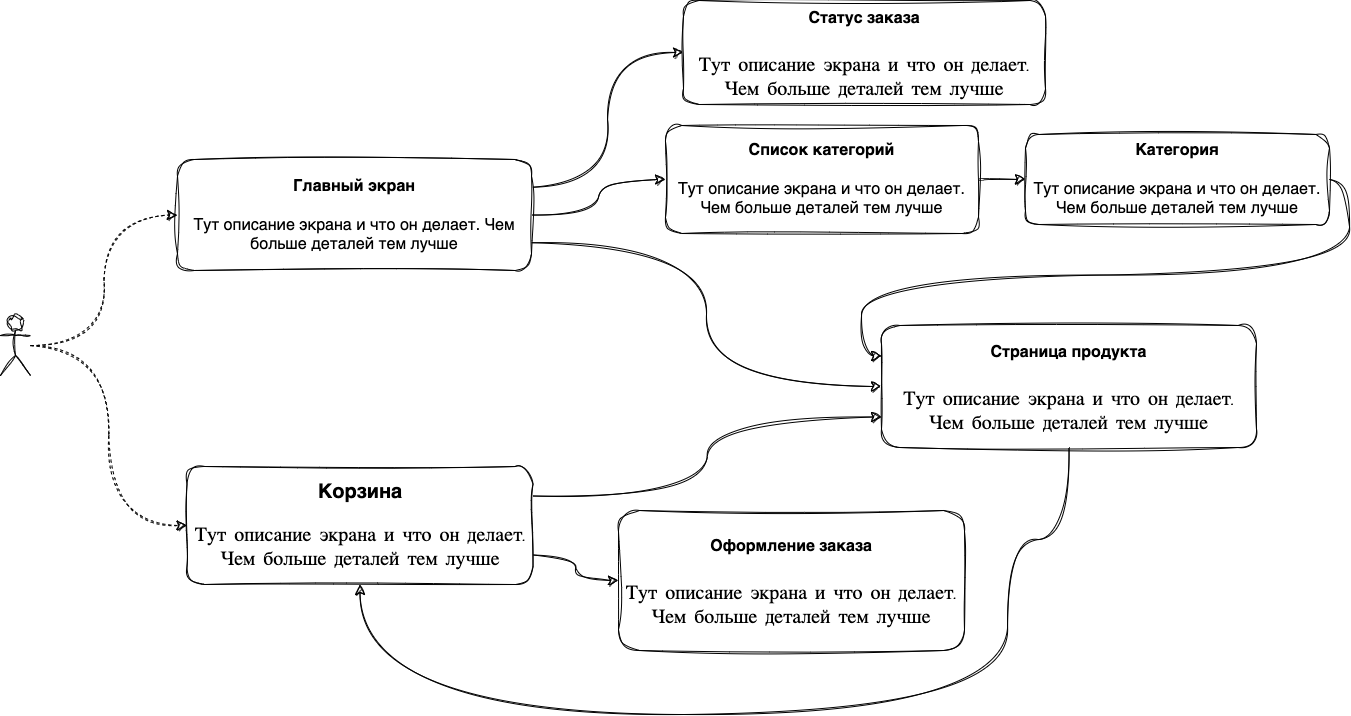
**9. Ищем скрытое преимущество**

Для продукта важно определить уникальное преимущество, которое может выражаться в необычном названии, дизайне, оригинальной функции. Это сильное преимущество, которое делает вас узнаваемыми и неповторимыми в сравнении с конкурентами.

**Этап 2. UX/UI**

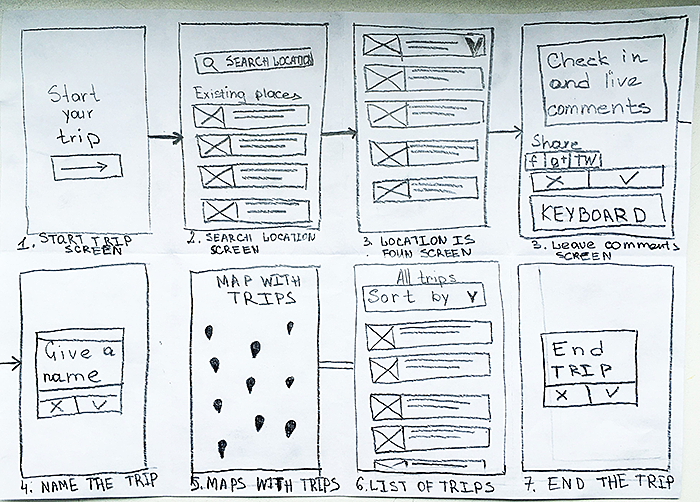
**2.1. Построение навигации приложения**

В этом Этапе определяем экраны и их навигацию между собой. На этом этапе не нужно создавать дизайн или скетчи экранов, достаточно будет описания экрана и указания стрелочками, с какого на какой экран происходит переход. Вы можете использовать любой удобный и доступный вам инструмент, такой как figma.com, miro.com, app.diagrams.net или даже листок бумаги.



**2.2. Наброски экранов**

Приступаем к скетчам UX экрана. Процесс создания скетчей экранов включает максимально простые наброски элементов на экране. В этом этапе не стоит углубляться в детали дизайна, а лишь определить, где и какие элементы будут находиться на каждом из экранов. Также как и предыдущий этап, можно делать где удобно.



**2.3. Дизайн приложения**

Дизайн приложения является критически важным аспектом создание приложения. Интерфейс влияет на первое впечатление и удержание пользователей. Если дизайн не привлекателен или сложен в использовании, пользователи могут перестать использовать приложение. Если у вас нет опыта в дизайне мобильных приложений, лучше обратиться к профессионалу или использовать ресурсы с готовыми дизайн-системами.

Вот два основных подхода к созданию дизайна приложения:

1. Делаем сами (если очень хочется)

Тут важна насмотренность. Прежде чем приступить к разработке дизайна приложения, изучите существующие популярные приложения и определите, какие элементы и стили могут быть применимы для вашего. Можно "набивать" насмотренность на сайте pinterest.com, behance.net или в любых других аналогах, искать примеры и вдохновляться ими.

Существует множество сайтов и платформ, предлагающих готовые дизайн-системы для мобильных приложений. Вы можете использовать эти ресурсы и собрать из готовых элементов свое приложение. К таким платформам можно отнести ui8.net, creativemarket.com, figmaresource.com, uistore.design и др.

1. Делаем через фрилансера или знакомого дизайнера

Самый оптимальный вариант, выйдет дороже, но качественнее.

Этап 3. Разработка

**3.1. Определение стека и архитектуры приложения**

Самый не простой для меня Этап в описании. Разработчики очень трепетно относятся к тем технологиям, которые знают и лишнее слово может вызвать негативную реакцию. Попробую очень осторожно пройтись по этому тонкому льду.

Выбор правильного технологического стека и архитектуры для мобильного приложения может существенно повлиять на его производительность, масштабируемость и дальнейшую его поддержку. Я выделил несколько простых рекомендаций:

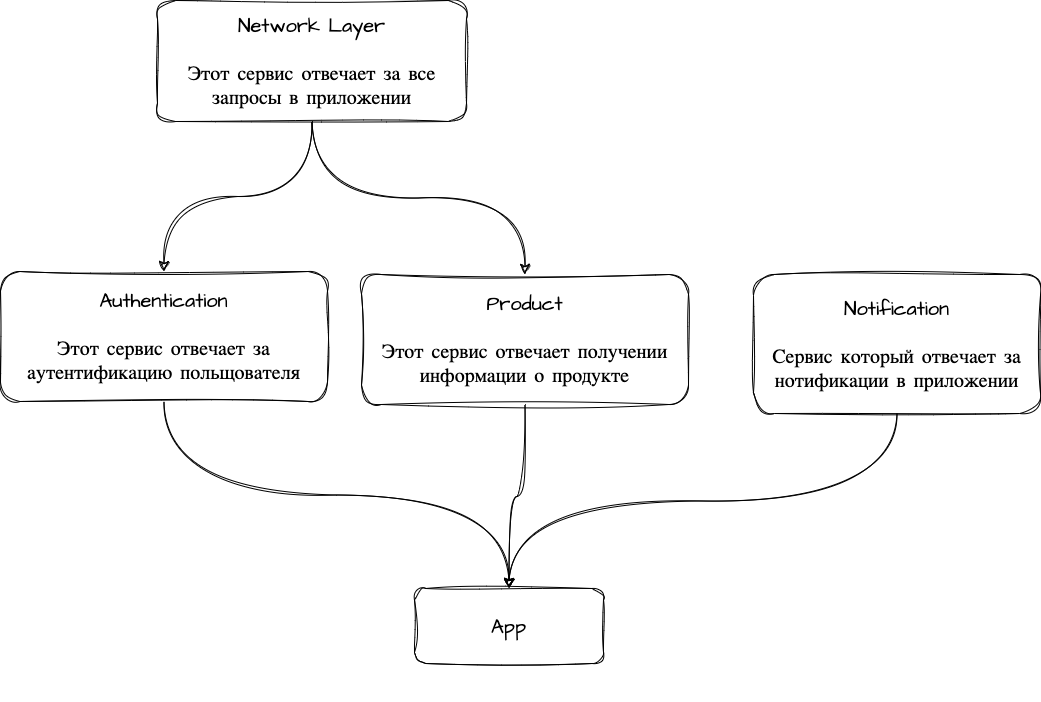
* При выборе стека хочется попробовать что-то новое вместо надежного старого. Если технология будет актуальна через год или два, то это отличный выбор. Однако, если судьба технологии в будущем неясна, то в долгосрочной перспективе такую технологию лучше не использовать.
* Определите платформу, на которую вы хотите ориентироваться (iOS, Android или обе). Хотите ли вы, чтобы ваше приложение в будущем появилось на другой платформе? В этом случае можно обратить внимание на Flutter, который позволяет одновременно охватить обе платформы. Однако, следует иметь в виду, что при использовании Flutter вы можете потерять уникальный стиль дизайна iOS и Android, а также не сможете использовать специфические возможности платформ, такие как AR, Live activity, виджеты и др. Если ваши силы и возможности позволяют разрабатывать на KMM, это также отличный вариант для разработки на обе платформы.
* Если вам нужно разработать серверную часть и вы никогда ранее с этим не сталкивались, обратите внимание на языки, которые вам будет легче освоить или самые популярные. У популярных языков большое сообщество, и вы сможете найти ответы на возникшие вопросы.
* Для разработки на iOS лучше использовать Swift и SwiftUI, а для Android - Kotlin.

С архитектурой все проще. Существует два основных типа архитектур у которых разнонаправленные и однонаправленные потоки данных. Для небольших команд и проектов лучше использовать разнонаправленные архитектуры (MVVM, MVC, MVP и др.), так как они обычно проще для понимания и реализации. В то же время, для больших проектов лучше подойдут однонаправленные архитектуры (Clean Swift, MVI и др.), которые обеспечивают более строгое разделение обязанностей и упрощают управление сложными экранами.

**3.2. Выделение сервисов в приложении**

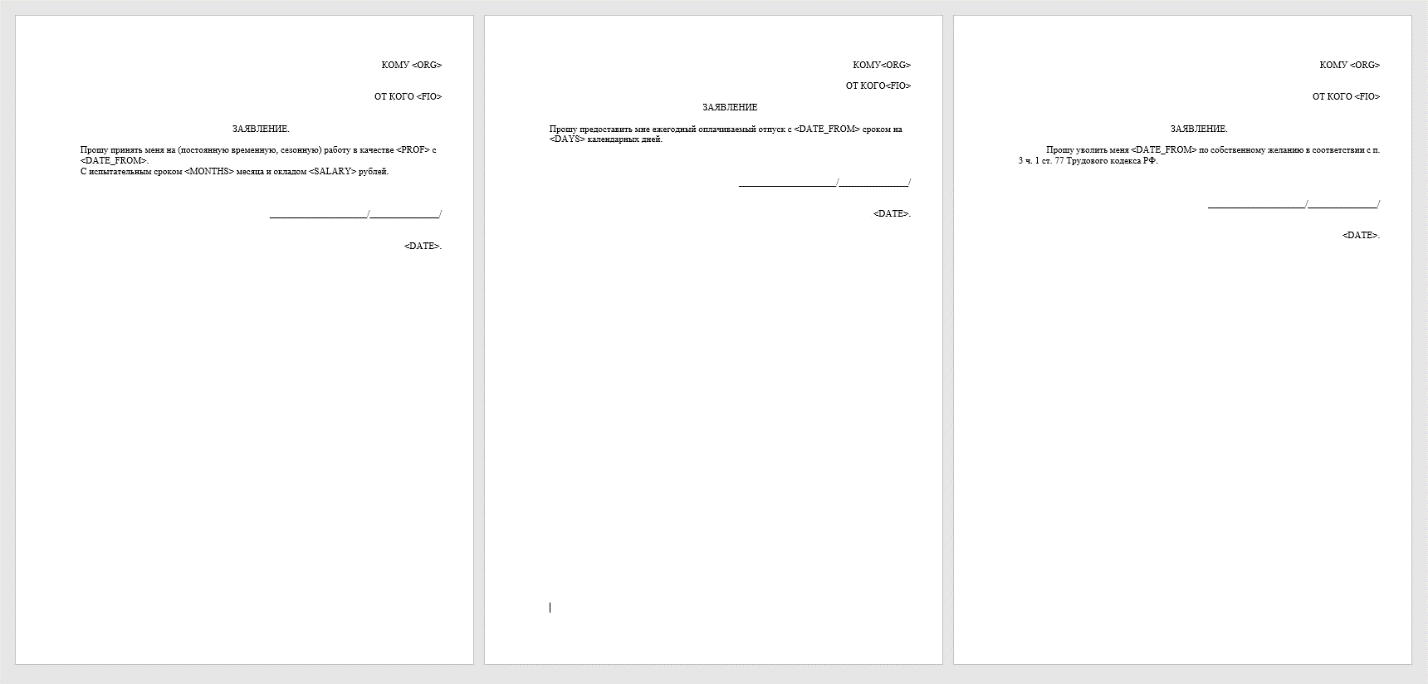
Этот этап не обязательный, но полезный. Он сильно упрощает понимание как структурно будет выглядеть приложение в коде. Такой подход ускоряет разработку.

Квадратиками и стрелочками обозначаем какой сервис за что будет отвечать и от какого сервиса зависит.



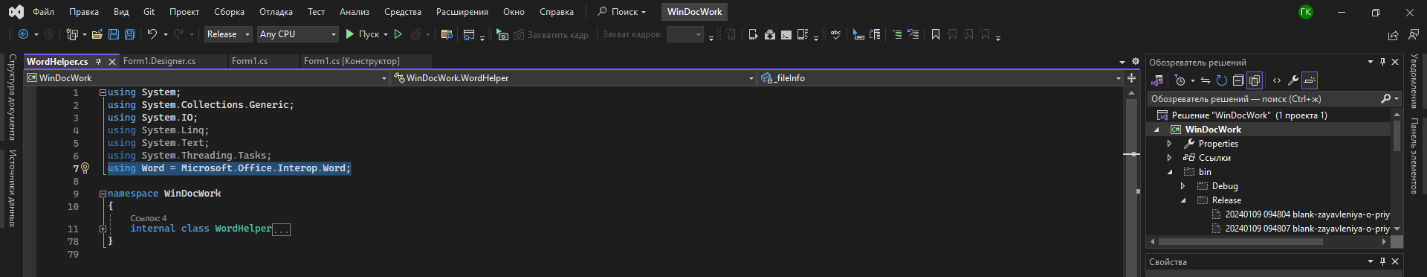
**3.3. Написание кода**

В данном проекте рассмотрим работу с Microsoft Office Word замена слов выражений. О чём пойдёт речь. Ну например у нас есть такой документ, в который регулярно приходится вписывать руками. Какие-то слова которые скажем можно автоматизировать да и брать откуда из какой-то базы данных. Либо из другого места для того чтобы подменять слова мы, заменим эти слова Вот такими <DATE> тегами. Они могут быть абсолютно любые. То есть вы можете придумать их формат сами. А смысл в том что они должны быть в документе чтобы мы могли бы их найти и заменить. На нужное нам выражение. В Visual Studio добавим новый проект, Windows Forms App, назовём его WinDocWork, сделаем сразу же его стартовым, итак что мы хотим сделать. Мы хотим добавить в программу ввод слов которые мы будем менять и соответственно будем менять их в документе. В нашем случае это три документа: заявление о приеме на работу, заявление на отпуск и заявление об увольнении. В Windows forms - добавляю TabControl, создаю 3 вкладки. Назовем их: Прием на работу, Отпуск и Увольнение. Во все вкладки добавим monthCalendar(для красоты), tableLayoutPanel и button. этого должно быть достаточно для выполнения проекта. Далее поработаем с нашими шаблонами документов, а именно заменим нашими тегами заполняемые поля во всех документах как на Рис1.

Рис. 1

Вернемся в Visual Studio, посчитав количество разных тегов в документах, в свойствах tableLayoutPanel выставляем: ColumnCount - 2, RowCount - количество тегов соответствующего документа. На tableLayoutPanel кликаем правой кнопкой мыши, выбираем в выпадающем меню - Правка строк и столбцов… В «Показать» выбираем «Строки» и каждой, кроме нижней, присваиваем тип размера AutoSize, а нижней присваиваем тип размера «Процент». Во вкладке «Прием на работу». В левый столбец верхней строки вставляем label. В его свойствах устанавливаем: Font(шрифт) - Times New Roman; 12pt, Text(название замещяющего текста) - Организация, Anchor(положение в ячейке) - Top,Right, Padding All(расстояние от краев ячейки) - 5. Размножаем label на остальные строки меняя название замещающего текста. В правый столбец первой строки устанавливаем textBox. В его свойствах устанавливаем: Font(шрифт) - Times New Roman; 12pt, Text(замещающий текст) - Организация, Anchor(положение в ячейке) - Top,Left,Right. Размножаем textbox на вторую, третью и шестую строки меняя замещающий текст. В правый столбец четвертой строки устанавливаем dateTimePicker. Его свойств менять не нужно. Размножаем dateTimePicker на седьмую строку. В правый столбец пятой строки устанавливаем numericUpDown. Его свойств менять не нужно. Во вкладке «Отпуск» делаем все тоже самое, разница только в количестве строк tableLayoutPanel, их будет пять. textBox ставим в первую и вторую строку, dateTimePicker в третью и пятую, numericUpDown в четвертую. Во вкладке «Увольнение» делаем все тоже самое, разница только в количестве строк tableLayoutPanel, их будет четыре. textBox ставим в первую и вторую строку, dateTimePicker в третью и четвертую.

Теперь приступим к коду. Прожимаем Shift+Alt+C, выбираем Class. Назовём его WordHelper. В открывшемся окне добавим using Word = Microsoft.Office.Interop.Word;, как на Рис 2.

Рис. 2

В классе WordHelper создаем публичный конструктор WordHelper, то есть функцию с именем класса в котором мы будем получать параметр fileName. Для этого проведем проверку, что у нас такой файл существует. Что бы корректно работал File.Exists подключим using System.IO;, для чего правой кнопкой мыши нажмем на лампочку и из выпадающей подсказки выберем этот пункт, как на Рис 3. Если такой файл существует то мы \_fileInfo присваиваем значение fileName, через лампочку создаем поле \_fileInfo (Рис 4), если не существует то вызываем throw new ArgumentException("File not found");. Открываем вкладку Form1.cs[Конструктор], tabControl1 и двойным кликом нажимаем на кнопку button1, открывается вкладка Form1.cs, где уже появилась функция private void button1\_Click(object sender, EventArgs e). В этой функции мы создадим объект helper куда и передадим fileName, в данном случае это «Заявление о приеме на работу». Чтобы не возникло никаких эксцессов сохраним шаблоны заявлений в папке с нашим проектом.

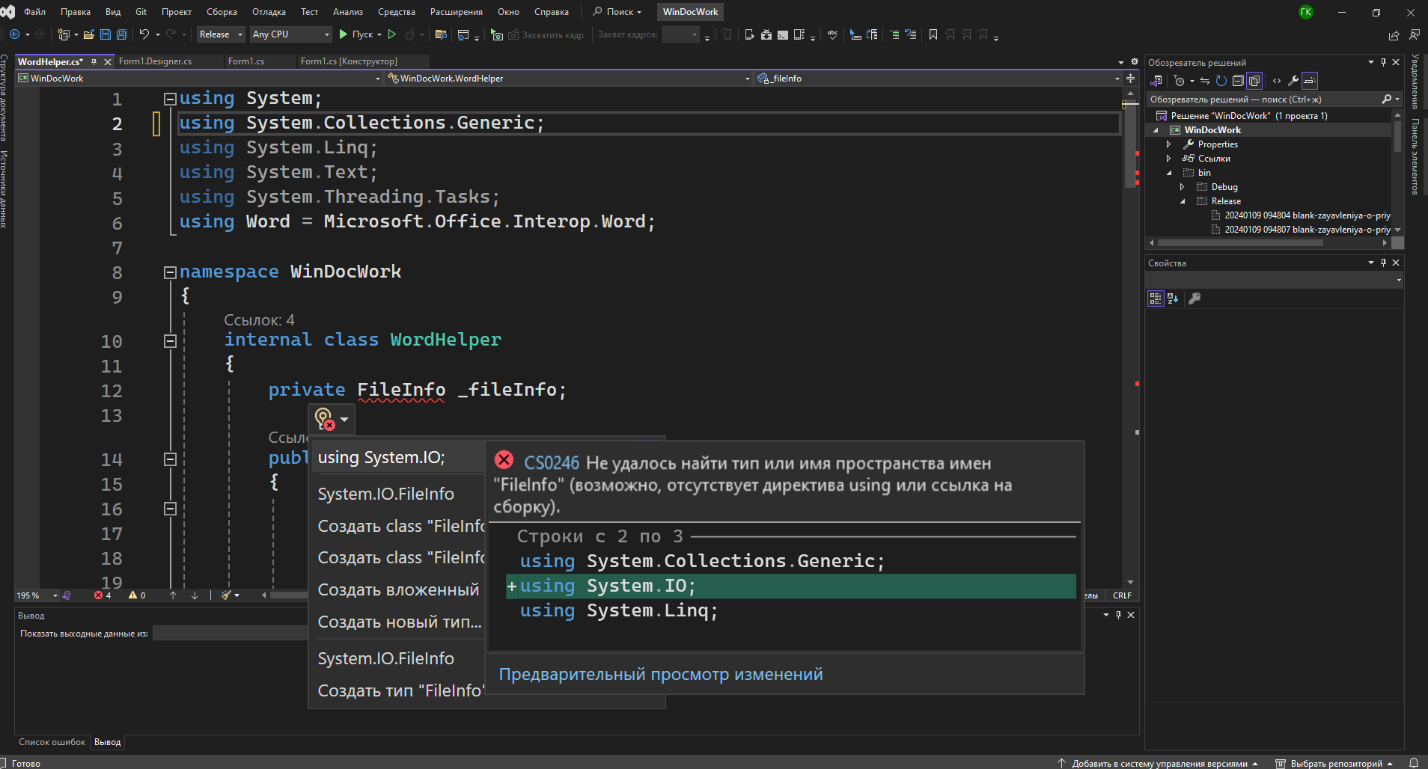
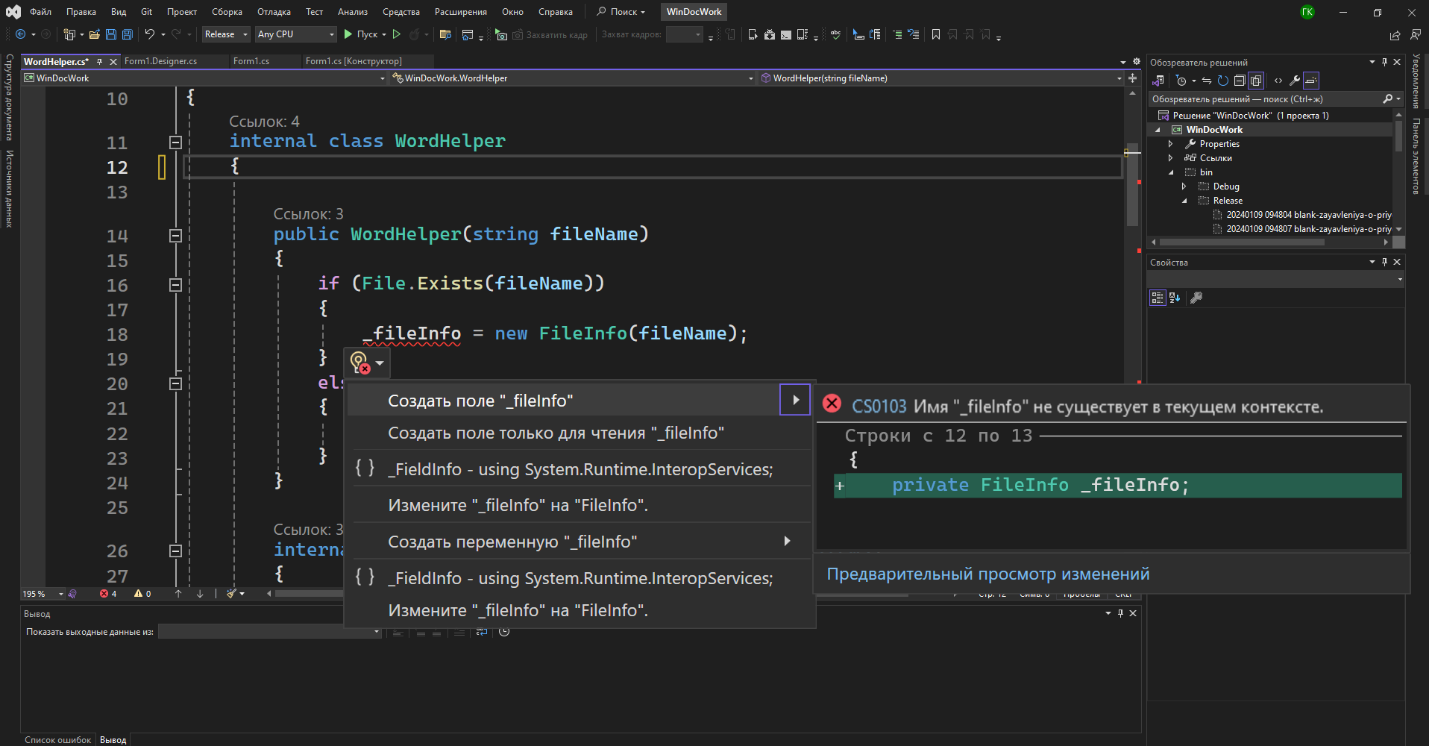
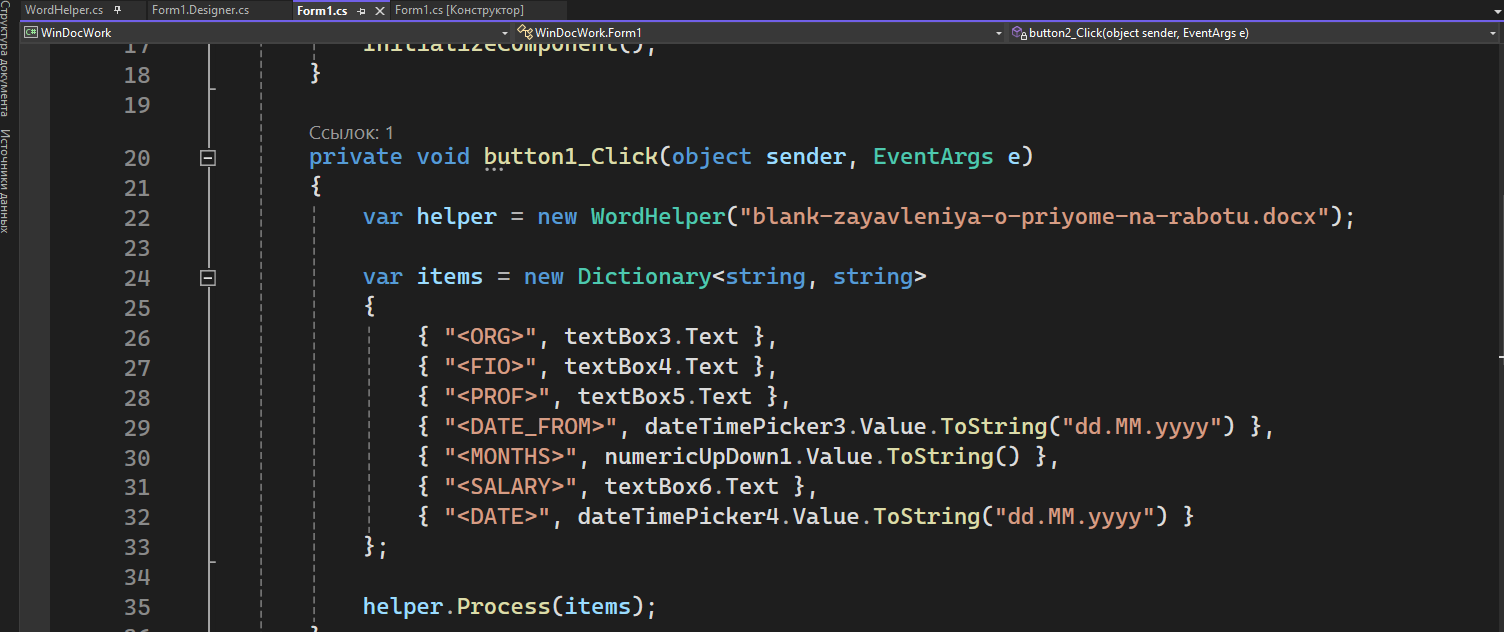
Рис.3

Рис. 4

В обозревателе решений делаем видимыми все файлы и кликаем правой кнопкой мыши на шаблоны, в выпадающем меню выбираем добавить в проект. Далее в свойствах каждого шаблона меняем параметр «Копировать в выходной каталог» на «Копировать более позднюю версию», то есть будет копироваться если произошли изменения в файле. Копируем название шаблона вместе с расширением и вставляем в объект var helper = new WordHelper("blank-zayavleniya-o-priyome-na-rabotu.docx");. Теперь необходимо собрать данные которые мы собираемся заменить в документе, для этого создадим список который будет отображать пару Key - Value, у нас это тег из документа и значение соответствующего поля нашей формы(Рис 5). Далее мы попросим наш helper обработать (Process) эти значения

Рис. 5

для чего передадим ему эти значения(items). Далее через лампочку генерируем метод Process и перейдем к нему, он находится во вкладке WordHelper. Убираем прерывание (throw new NotImplementException()), используем конструкцию try-catch так как это критическое место программы и если что то пойдёт не так то процесс прервется но сама программа продолжит работу. Установим тип метода bool соответственно метод будет возвращать true если все прошло хорошо и false если что то пошло не так. В блоке try создаем конструктор Word.Application(), затем создаем объект, для передачи данных в Word, это будет наш \_fileInfo.FullName. FullName в данном случае означает полный путь к нашему файлу. Создаём объект Type.Missing который представляет отсутствующее значение в данных объекта, он нужен при работе с программой MicrosoftWord. Далее открываем наш файл app.Documents.Open(file);. После чего создадим цикл для каждого item наших items, в котором создадим объект Word.Find find = app.Selection.Find; с помощью которого и будем искать в документе теги для замены. Далее объекту присваиваем тег который будем искать find.Text = item.Key; и текст на который будем менять соответствующий тег find.Replacement.Text = item.Value;. После создадим объекты необходимые в поиске Object wrap = Word.WdFindWrap.wdFindContinue; и замене Object replace = Word.WdReplace.wdReplaceAll; текста в Word. WdFindWrap - Указывает поведение оболочки, если для операции поиска указан выделенный фрагмент или диапазон, а текст поиска не найден в выделенном или диапазоне. wdFindContinue - после поиска по выделенному фрагменту или диапазону в Microsoft Word отображается сообщение о том, следует ли искать оставшуюся часть документа. Операция поиска продолжается, если достигнуто начало или конец диапазона. WdReplace - Указывает количество замен, которые необходимо выполнить при использовании функции поиска и замены. wdReplaceAll - Заменить все вхождения.

**Этап 4. Аналитика приложения**

Два небольших пункта

**4.1. Падения приложения**

Если пользователь сталкивается с тем, что приложение неожиданно закрывается и не работает должным образом, это может привести к тому, что он перейдет к конкурентам. Поэтому необходимо интегрировать в ваше приложение сторонний сервис для мониторинга и отслеживания падений приложения. Самые популярные, но не единственные, на сегодня Firebase Crashlytics и Google Crashlytics

**4.2. Аналитические ивенты**

Аналитика — это ваши глаза и уши. Она помогает понять, на каких этапах пользователи теряют интерес или сталкиваются с проблемами.

На начальном этапе достаточно отслеживать базовые события, такие как открытие экрана, и отправлять соответствующие события в аналитический сервис. В некоторых сервисах также предусмотрена возможность отправки ошибок и логов, что может быть полезно для выявления и устранения проблем в приложении.

Среди самых популярных, но не единственных, сервисов на сегодняшний день можно выделить Firebase Analytics и Google Analytics. Однако, при выборе подходящего сервиса, обратите внимание на стоимость их использования.

**Этап 5. Продвижение**

**5.1. Построение клиентского пути (Customer Journey Map - CJM)**

Чтобы эффективно продвигать приложение, важно разобраться в том, как пользователи взаимодействуют с ним. Для этого можно составить карту клиентского пути (Customer Journey Map, CJM), которая отражает опыт и взаимодействие пользователей с продуктом на различных этапах. CJM поможет выявить слабые стороны клиентского пути и исправить до продвижения.

CJM — это визуальное представление различных этапов, через которые проходит пользователь при взаимодействии с продуктом. Начиная от момента, когда пользователь увидел рекламу или когда у него появилась потребность в товаре, до покупки приложения, подписки, или покупки внутри приложения, можно также включить и постпродажное взаимодействие с приложением.

Существует множество методов создания CJM. Каждый метод имеет свои особенности:

* Графическое отображение. Этот метод предполагает визуализацию пользовательского пути на диаграмме, где каждый этап представлен отдельным блоком или ячейкой, которые соединены стрелочками.
* Таблицы и списки. Это подход заключается в составлении таблиц или списков, где каждый этап пользовательского пути описан подробно.
* Сценарии и истории. Описание сценариев и историй, основанных на реальном опыте пользователей.

**5.2. Написание коммуникационной стратегии**

После анализа карты клиентского пути следует составить коммуникационный план, который определит наиболее эффективные способы продвижения вашего приложения.

Этот план может состоять из следующих пунктов:

Цели: чётко определите цели, которых нужно достичь с помощью коммуникационных усилий. Это может быть повышение осведомленности, увеличение продаж, увеличение количества установок, повышение вовлеченности пользователей или позиционирование вашего приложения как лидера рынка и др.

Целевая аудитория: определите идеальных пользователей вашего приложения, включая их демографические, психографические характеристики, интересы и предпочтения. Это поможет вам адаптировать ваши сообщения таким образом, чтобы они нашли отклик у пользователей.

Ключевые сообщения: Разработайте набор ключевых сообщений для выбранной целевой аудитории, которые передают уникальные торговые преимущества (УТП) вашего приложения. Эти сообщения должны быть последовательными во всех каналах коммуникации и подчеркивать преимущества и особенности приложения.

Каналы и тактика: определите наиболее подходящие каналы коммуникации для достижения целевой аудитории, такие как социальные сети, маркетинг электронной почты, контент-маркетинг, партнерство с влиятельными лицами, платная реклама и др. Выберите тактику, которая лучше всего соответствует вашим целям и предпочтениям целевой аудитории. Если у вас приложение только на одну платформу, это тоже стоит учитывать, например реклама у блогеров не подойдет, так как она охватывает обе платформы.

Контент-стратегия: создайте контент-план, включающий типы контента, который вы будете создавать (например, посты в блоге, видео, посты в социальных сетях), темы, которые вы будете освещать, и частоту публикации или обмена.

Временные рамки: определите сроки реализации коммуникационной стратегии с учетом любых рекламных мероприятий, обновлений или сезонных тенденций, которые могут повлиять на ваши сообщения.

Бюджет: выделите бюджет на коммуникационные усилия, чтобы обеспечить наличие необходимых ресурсов для эффективной реализации стратегии.

Измерение и оценка: определите ключевые показатели эффективности (KPI), которые вы будете использовать для отслеживания успеха вашей коммуникационной стратегии. Регулярно анализируйте результаты и корректируйте стратегию по мере необходимости, чтобы оптимизировать эффективность.

План коммуникации в кризисных ситуациях: Разработайте план действий в негативных ситуациях или потенциальных кризисах, которые могут повлиять на репутацию вашего бренда.

**Этап 6. Развитие**

В 2022 году рынок мобильных приложений изменился: часть инструментов для продвижения заблокирована, стало сложнее монетизировать приложения из-за ограничений платёжных систем.

Но есть среди всего этого и хорошее:  
— появились новые сторы, или магазины приложений;  
— нашлись дополнительные инструменты и каналы продвижения;  
— приложения стали использовать не только для удержания клиентов, но и для привлечения новых.

Спрос на создание приложений продолжает расти, поэтому в будущем конкуренция будет усиливаться. В течение 15 лет я работала в сфере маркетинга, работая с органикой в SEO и ASO, SERM и Acquisition & Retention Marketing. В статье, опираясь на свой опыт, я рассказываю, как продвигать своё приложение и на какие современные тренды стоит обратить внимание.

Особенности мобильного маркетинга

Прежде чем говорить о том, как продвигаться в сторах, важно понять, чем это отличается от продвижения в поисковых системах. Все различия кроются в сценариях их использования. В интернете человек ищет информацию, как удовлетворить конкретную потребность: где купить товар, какую модель выбрать и т. д. Кроме этого, он непосредственно покупает, оставляет отзывы и возвращается за покупкой снова.

Когда человек приходит в стор, то решение у него уже есть, поэтому он ищет конкретный бренд или название продукта или услуги. Это подтверждает тот факт, что 70% запросов в сторах приходится на бренды.

Такое поведение людей в сторах легко объяснить, если посмотреть, какое место в процессе покупки занимают приложения. Весь процесс принятия решения о покупке можно разложить на пять этапов:

1. осознание потребности,  
2. поиск информации,  
3. оценка и выбор вариантов,  
4. выбор источника и покупка,  
5. реакция на покупку.

Поиск в интернете охватывает последние четыре этапа принятия решения о покупке из пяти. В то время как поиск в сторе — только последние два.

Сегодня мобильный маркетинг — один из основных инструментов ретеншена

Итак, мы видим, что люди используют приложения в основном, чтобы выбрать, где купить продукт или услугу и оставить обратную связь о покупке. Отсюда можно сделать вывод, на что они обращают внимание при выборе приложения, и учесть это в общей стратегии продвижения приложения.

Мы выделяем основные факторы: видимость приложения в поиске, рейтинг и отзывы о нём и удобство использования самого приложения. Дальше в статье я расскажу, с помощью каких доступных инструментов можно улучшать эти факторы.

Каналы продвижения мобильных приложений, доступные в России

В агентстве направление ASO работает с 2021 года. За это время на рынке многое изменилось, и все решения, о которых я рассказываю, мы успели проверить на практике и получить положительные результаты.

Главное изменение на мобильном рынке — это уход ASA (Apple Search Ads) из России. С 2022 года реклама в сторе Apple недоступна. Из-за этого главный фокус продвижения мобильных приложений сместился на ASO, или продвижение по ключевым запросам.

Ниже я привожу каналы, которые доступны в России в 2023 году. Из этого списка видно, что в нашем арсенале достаточно инструментов для продвижения приложений.

Сейчас мы продвигаем приложения с помощью ASO, рекламных кабинетов и работаем с рейтингами приложений. Ниже я рассказываю о работе с каждым из инструментом подробнее.

ASO: выводим приложения в топ выдачи

В 2022 году с российского рынка ушли многие популярные бренды, поэтому сейчас хорошее время, чтобы занять лидирующее положение в освободившихся нишах. Например, запрос «ikea» именно в таком написании вбивают 1838 раз в день. Это означает, что спрос на бренд есть, и мебельные магазины могут продвигаться по этому запросу. Для этого надо:  
— Добавить ключи в разных вариациях написания в индексацию.

— Вывести своё приложение в индексе в топ.

Чтобы составить стратегию и понять, сколько трафика недополучает клиент, мы тщательно анализируем разделение рынка по конкурентам и проверяем спрос. Так для оценки потенциала бренда мы подбираем бренды с близкой целевой аудиторией и собираем семантическое ядро. После этого рассчитываем потенциальный суточный прирост новых установок.

На основе аналитики строим стратегию продвижения приложения, чтобы вывести продукт в топ по целевым запросам и улучшить видимость в сторе. Такой подход даёт результаты, и в итоге ASO помогает привлекать новых клиентов и пользователей.

Работа с рекламными кабинетами: общий алгоритм подготовки и продвижения

Сейчас для рекламы мобильных приложений доступны кабинеты VK, MyTarget, а также контекстная реклама в «Яндекс Директе». При продвижении мобильных приложений важно учитывать, что установка приложения требует от пользователя больше усилий, чем переход на сайт или в соцсети бренда. Клиенту нужно перейти в стор, подтвердить скачивание приложения, вбить персональные данные. Всё это означает, что необдуманных установок не так уж и много, а вот конверсии в целевое действие могут быть выше, чем на сайте.

В работе с рекламными кабинетами мы придерживаемся определённого алгоритма, благодаря которому кампании, которые мы запускаем, работают эффективно. Сам алгоритм выглядит так:

1. Интеграция приложения с платформой аналитики до старта кампании. Мы пользуемся аналитикой AppMetrica. С помощью этого или подобного сервиса можно отслеживать ключевые показатели приложения, а также оптимизировать стратегию продвижения.

2. Определение цели рекламной кампании. У каждой кампании должна быть своя цель, например, установка приложения или выполнение целевого действия. При этом неважно, какой именно инструмент для продвижения мы используем. Такой подход помогает оценивать не просто стоимость клика, но и стоимость целевого действия.

3. Эксперименты с форматами и объявлениями. Мы придерживаемся системного запуска рекламных кампаний и тестирования гипотез. Так удаётся понять, какие форматы, тексты и графические изображения работают эффективно и подходят по стоимости за целевое действие.

4. Работа со страницей приложения в сторе. Страница привлекает внимание новых пользователей и рассказывает о новых возможностях нынешним. Для обеих целей работают качественные скриншоты из приложения и подробная информация о его функциях и задачах пользователя, которые оно решает.

5. A/B-тесты. В сторах можно проверить конверсию в открытие карточки приложения и в установку приложения. Так, например, в одном из проектов мы проверяли, влияет ли иконка на эти конверсии.

Кроме продвижения приложения в топе и рекламы, важно работать и с его рейтингом. Дальше расскажу, как рейтинг влияет на привлекательность приложения.

Работа с рейтингом и отзывами: поиск точек роста и повышение привлекательности приложения

Пользователь может попасть в карточку приложения из поиска в сторах, стандартных поисковых системах или из рекламы. В любом случае один из ключевых факторов, который влияет на решение об установке приложения, — это его рейтинг и отзывы.

Анализ отзывов помогает выделить точки роста приложения, потому что пользователи охотно пишут о проблемах, с которыми сталкиваются во время работы с приложением. Чтобы понять, какие ошибки исправить в первую очередь, мы анализируем отзывы о приложении клиента и конкурентов.

Рейтинг приложения складывается не только из отзывов, но и в том числе из клиентского опыта. То есть, важно улучшать сценарии взаимодействия, дизайн, работоспособность приложения. Вот несколько приёмов, которые мы используем на проектах, чтобы улучшить рейтинг:

**Дорабатываем приложение так, чтобы пользователям было максимально удобно в нём работать**. В этом как раз помогает анализ обратной связи от клиентов. С его помощью мы улучшаем клиентский путь внутри приложения, UX-дизайн, тексты, систему монетизации.

**Отвечаем на все отзывы клиентов.** Одно из важнейших правил работы с отзывами — отвечать на всё — негативное и позитивное. Этот подход показывает потенциальным и текущим клиентам, что их мнение учитывают и критические ошибки исправляют.

**Рассказываем об обновлениях и об исправленных ошибках.** Писать об исправлении ошибок стоит не только в карточке приложения и отзывах, но и в описании обновления внутри стора. При этом мы не перечисляем коды ошибок и багов, а используем простые формулировки. Например, «ускорили загрузку первого экрана» или «добавили на главную страницу перевод через систему быстрых платежей». Такая забота о клиенте среди прочего делает приложение более привлекательным.

Напоследок хочется сказать, что несмотря на усиливающуюся конкуренцию в мобильном маркетинге и на наличие эффективных инструментов для системного продвижения приложения, мы продолжаем искать дополнительные способы для продвижения приложений. Пробуем новые форматы, например, запуск рекламы через блогеров в «Телеграме» и загрузку приложений не только App Store и Google Market, но и в новые сторы.

**Заключение**

Когда мы разрабатываем приложение для себя, чаще всего не задумываемся о монетизации, в этом случае весь процесс очень простой. Однако, как только возникает желание заработать, процесс разработки сразу усложняется в разы. Требуется анализировать рынок, конкурентов, выдялять целевую аудиторию и много еще чего.

Существует еще одна стратегия заработка на приложениях, которую стоит упомянуть. Эта стратегия заключается в том, чтобы копировать самое популярное приложение, выкладывать в store и забывать о нем, а затем переходить к следующему популярному, копировать его и тд. Для такой тактики все эти Этапи не нужны.

Однако, если хочется создать что-то качественное и уникальное, как уже писал, процесс становится гораздо сложнее. Каждый этап разработки требует внимательного подхода. Учитывая все эти Этапи, можно создать очень крутые приложения, которое будут полезны и интересны вашим пользователям и вам.