SEO (Search Engine Optimization, поисковое продвижение, organic, органика) — инструмент и канал продвижения, цель которого — вывести сайт на первые строчки поисковой выдачи поисковых систем, как правило, Google и Яндекс. Достигается это за счёт того, что сайт приводится в надлежащее состояние — в соответствии с требованиями и ожиданиями поисковых роботов и реальных пользователей.

SEO — это особенный канал продвижения, работа над которым требует совместного участия специалистов разных профилей. Дизайнеры и аналитики должны проработать удобные интерфейсы, маркетологи — правильно определить спрос и наполнить сайт релевантным контентом, разработчики — удовлетворить технические требования поисковиков и сделать сайт производительным.

Несмотря на то, что поисковая оптимизация — это трудоёмкий процесс, для многих проектов он оказывается спасательным кругом, так как является ключевым или даже единственным источником пользователей и клиентов. Вот некоторые выдержки из статистики о поисковом спросе и этом канале:

* 68% опыта взаимодействия с сайтами начинается с поисковых систем (BrightEdge);
* 0,78% пользователей, осуществляющих поиск в Google, открывают результаты на второй странице (Backlinko);
* 53,3% всего трафика сайтов приходится на органический поиск. (BrightEdge);
* 92,96% мирового трафика приходится на поиск в Google, Картинках Google и Картах Google (Sparktoro);
* на SEO приходится на 1000%+ больше трафика, чем обычные социальные сети (BrightEdge);
* средний показатель эффективности топового результата органического поиска Google составляет 31,7% (Backlinko);
* количество поисковых запросов в Google составляет примерно 3,5 млрд в день (Internet Live Stats);
* 39% покупателей руководствуются результатами поиска (Think With Google).

**Основные преимущества SEO как канала продвижения**

**Со временем канал всё больше выходит в плюс**

Самое сложное – на старте вывести проект в топ выдачи. Но если это получилось, канал будет работать длительное время, приносить трафик и заявки, а затраты на него будут не такими масштабными, как на старте. Это значит, что каждый последующий посетитель, заявка, продажа будут доставаться вашему проекту дешевле, и во многих проектах со временем этот канал становится самым выгодным и даже образующим бизнес.

**Нет рекламных ограничений**

В контекстной и таргетированной рекламе есть рекламные ограничения. Например, если ваш проект связан с медицинскими услугами, для продвижения в платной рекламе вам нужно дополнительно прикладывать сертификаты, проходить модерации, объявления могут отклоняться. В некоторых тематиках платную рекламу запустить нельзя совсем. При этом в SEO можно продвигать всё, что люди ищут в поисковиках: если человек вбивает запрос на любую тему Google и Яндекс дадут ему максимально релевантный ответ, и это может быть ваш сайт.

**Растёт коэффициент конверсии, усиливаются другие каналы**

Внедрение рекомендаций, актуальных для SEO, неизбежно сказывается на коэффициенте конверсий сайта. Иными словами, среди тех, кто приходит на ваш сайт, увеличивается процент тех, кто покупает. Дело в том, что удобные интерфейсы, грамотная навигация, хорошие изображения и исчерпывающие тексты — всё это необходимо как для поисковых роботов, так и для пользователей — поэтому чаще покупают даже те пользователи, которые приходят с других рекламных каналов.

**Позволяет в дальнейшем отказаться от контекстной рекламы**

Многие проекты первым каналом продвижения подключают платный трафик, например, контекстную рекламу в Яндексе и Google. И это часто действительно оправдано. Но если со временем сайт выйдет в топ в органике, чаще всего продолжать платить за рекламный поисковый трафик нет смысла, и высвободившиеся деньги можно распределить на другие каналы продвижения или улучшение продукта.

**Когда этот канал подходит плохо:**

* на ваш продукт нет сформированного спроса и его просто не ищут в поисковиках;
* на продукт низкий или падающий спрос;
* когда нет ресурсов дорабатывать сайт;
* продажи нужны сразу;
* спрос съедают агрегаторы и маркетплейсы.

В интернете находятся миллиарды **документов** (среди них и сайты). Из них поисковые системы за доли секунд должны найти те, которые точнее всего отвечают **запросу** и ожиданиям пользователя. Для этого **поисковые роботы** исследуют интернет-пространство, формируя **поисковую базу,** которая периодически переживает **апдейты**, то есть обновления. Документы, содержащие одни и те же слова и фразы, попадают в **индекс** — список документов, которые выдаются в ответ на запросы с этими словами. Эти слова называют **ключевыми словами**. При этом поисковые системы заинтересованы в том, чтобы пользователь получил исчерпывающий и максимально точный ответ на свой запрос. Чтобы отдавать пользователю сайт, который отвечает наилучшим образом, поисковые системы **ранжируют** сайты — выше в выдаче показывают наиболее подходящие, то есть имеющие наибольшую **релевантность**. Сайт становится релевантным, если удовлетворяет определённым требованиям со стороны поисковиков и пользователей — их называют **факторами ранжирования**. Задачи команды SEO в том, чтобы сайт соответствовал этим требованиям.

Как понятно из текста выше, чтобы пользователь нашёл сайт по какому-то запросу, на сайте должны использоваться ключевые слова, содержащиеся в этом запросе. Список таких ключевых запросов называют **семантикой сайта**. Семантику сайта всегда нужно учитывать при проектировании сайта, потому что не может быть документа, который одинаково релевантен для всех товаров и услуг. Это значит, что для каждого товара или услуги должна быть своя страница со своими ключами. Но в таком случае как, например, быть с запросами «чёрная куртка купить» и «чёрная куртка купить в интернет-магазине»? Кажется, что в рамках интернет-магазина ответом и на один запрос, и на другой будет одна и та же страница. Когда так происходит, запросы объединяют в **кластеры** — то есть группы запросов, для которых подразумевается одна посадочная. При этом сами эти запросы отличаются друг от друга тем, что один, более широкий. «Чёрная куртка купить» — будет более **частотным**, так как он родительский как для запроса «чёрная куртка купить в интернет-магазине», так и, например, для запроса «чёрная куртка купить недорого». Так, запросы делятся на высокочастотные, среднечастотные и низкочастотные, при этом определить, к какой категории относится запрос можно чаще всего только при анализе конкретного рынка, так как на некоторых рынках 500 запросов в месяц может быть высокочастотным показателем, а для других — очень низкочастотным.

Оценка сайтов происходит в основном с помощью двух инструментов:

1. **Асессоров.** То есть людей, которые вручную проверяют сайты и формулируют критерии качественных сайтов.
2. **Алгоритмов.** Это механизм, в основе которого нейронная сеть, постоянно обучающаяся и улучшающая собственные навыки ранжирования — то есть выдачи в первую очередь самых адекватных запросу документов.

Важно отметить, что именно из-за самообучающихся алгоритмов даже сами сотрудники Google и Яндекса не до конца знают, какие сайты попадут в топ и согласно какой формуле. Это значит, что рекомендации специалистов по поисковому продвижению основаны скорее на успешных практиках, чем на точных гарантированных расчётах (хотя попытки поиска таких возникают регулярно), и в работе над этим каналом всегда будут элементы случайности и неопределённости.

Также формулы ранжирования и алгоритмы отличаются у разных поисковых систем, например, у Google и Яндекса. Однако, есть принцип, который объединяет все поисковые системы и о котором всегда стоит помнить команде, занимающейся SEO. Цель поисковой системы — с первого раза удовлетворить запрос пользователя самым релевантным ответом. Это значит, что если пользователи будут приходить на ваш сайт, удовлетворять свою потребность и не возвращаться обратно в поиск, чтобы найти кого-то ещё по тому же запросу — ваше продвижение в топ будет успешным.

Таким образом, поисковая оптимизация — это точка пересечения интересов пользователя, владельца сайта и поисковой системы. Все заинтересованы в том, чтобы пользователь получил наилучший пользовательский опыт и наиболее точный ответ на свой запрос. В следующем разделе мы подробнее разберём факторы ранжирования.

**Выливать сайт на продакшен с закрытым robots.txt ( директива Disallow: /).**

Эта директива не позволяет роботам индексировать сайт, соответственно, сайт не попадает в выдачу совсем. Директива используется для релиза сайта на тестовый домен, но иногда команда разработки забывает поправить robots после релиза.

**Одинаковые и неинформативные метатеги Title и Description на всех страницах.**

Бывает так, что про метатеги забывают, и они остаются одинаковыми для всех страниц или группы страниц. Лучше реализовать шаблон, чтобы в Title и Description по умолчанию уходил заголовок и первый абзац текста страницы, но с возможностью исправить их в дальнейшем.

**Не делать канонизацию слеша (/) в конце.**

В таком случае открывается как страница *site/catalog,* так и *site/catalog/*. Для поисковых систем это будут две разные страницы с одинаковым контентом. Необходимо настраивать 301-редирект с версии без слеша на версию со слешем. То же самое касается главного зеркала сайта: необходимо, чтобы был настроем 301-редирект на <https://www.site.ru/> со страниц:

- [https://www.site.ru](https://www.site.ru/),

- [https://site.ru](https://site.ru/),

- <https://www.site.ru/index.php>,

- <https://site.ru/index.php>,

- <https://www.site.ru/index.html>,

- <https://site.ru/index.html>.

**Создавать одинаковые сайты по разным адресам.**

Иногда заказчик желает заполонить своим сайтом весь интернет и размещает один и тот же сайт по разным доменам. Однако это не работает, потому что негативно влияет на все копии, и в топ не выходит ни одна. Также существенной проблемой для продвижения становятся тестовые копии сайта, не закрытые от индексации.

**Релиз нового сайта без 301-редиректов со страниц старого сайта.**

Часто заказчик делает редизайн и меняет структуры, но не делает редиректы со старых страниц. Это негативно влияет на позиции сайта и накопленное доверие со стороны поисковых систем. Плюс, это негативно влияет и на пользовательский опыт: пользователь переходит по сохраненной ссылке и попадает на 404-страницу с ошибкой, потому что страница сайта переехала, но при этом переадресация была не настроена.

**Несуществующая страница ведет на главную или страницу каталога.**

Для несуществующих страниц необходимо отдавать 404 статус и создавать отдельный шаблон страницы, чтобы пользователь понимал, что попал на несуществующий адрес. В некоторых проектах вместо открытия такого шаблона при попадании на несуществующую страницу пользователь сразу переадресуется на главную страницу или в каталог. Это путает пользователя и негативно сказывается на поисковом продвижении.

**Страницы без текстового блока.**

Бывает так, что заказчик приходит с запросом на минималистичное наполнение, без лишнего текста. Особенно так часто происходит с главной страницей. Но зачастую это негативная практика, так как главная страница идеальна для высокочастотных запросов, и скорее всего на ней необходимо будет разместить текст с ключевыми словами. Лучше на этапе проектировки и дизайна предусматривать один или несколько текстовых блоков, которые можно скрыть, пока текстов нет, и добавить, как только понадобится разместить тексты.

Коммуникация — ключ к успеху. В разработке программного обеспечения ясно и хорошо изложенные требования помогают командам разработчиков создать качественный продукт. Наверняка вы уже составили свой первый документ — видение или, другими словами, концепцию.

:info:**Концепция** раскрывает цели и суть продукта, содержит описание пользователей, основные возможности системы и часто даже планы по продвижению этого продукта.

Другой документ — **спецификация,** онауточняет, конкретизирует, детализирует описание системы, которую вы будете создавать. Когда вы описываете требования, то опираетесь на сведения, полученные ранее в ходе общения с заказчиком и пользователями. У вас уже есть понимание ожидаемых функций вашего продукта и, возможно, первые прототипы интерфейса.

Но для разработки реального продукта этого недостаточно. Как правило, заказчики и разработчики говорят на разных языках. Заказчики и пользователи чаще всего представляют поведение системы — что она будет делать и как с ней можно взаимодействовать. Программисты же думают о продукте с точки зрения его внутренних характеристик. Понять друг друга им помогает в первую очередь аналитик. Он превращает потребности заказчика и пользователя в требования, а требования в задачи для разработчиков и других членов команды.

Первоначально это делается путем составления спецификации требований к программному обеспечению:

* функциональные требования;
* системные требования;
* бизнес-правила;
* атрибуты качества;
* ограничения и проч.

Самая известная книга в области подготовки спецификации и требований — «Разработка требований к программному обеспечению» Карла Вигерса. Эта книга уже на протяжении многих лет является своего рода библией для аналитиков. Это универсальный учебник по работе с требованиями. Многие концепции, описанные в этой книге, являются общепринятыми в индустрии разработки программ.

В этой книге даются следующие определения:

1. **Требование** —спецификация того, что должно быть реализовано в системе. В требованиях может быть описано поведение системы, ее атрибуты; требования могут служить ограничениями в процессе разработки системы.
2. **Спецификация** — документ, целью которого является предоставление исчерпывающего описания разрабатываемого программного продукта, включая его назначение, основные бизнес-процессы, которые будут поддерживаться, функции, ключевые параметры производительности и поведение.

Структура требований в таком документе изменяется в зависимости от проекта, но всегда включает функциональные и нефункциональные требования. О них поговорим немного позже.

Существуют шаблоны, по которым составляется структура спецификации требований к ПО, но нет строгих правил. Поэтому для стандартных шаблонов изменения скорее необходимы. Каждая компания и даже команда оформляет этот документ по-разному, но тем не менее можно выделить некие важные разделы, которые нужны всегда. Об этом мы также поговорим на предстоящих встречах.

На самом деле, как только вы начали задавать вопросы стейкхолдером и фиксировать идеи для своего продукта, вы уже занимались выявлением требований. Но когда вы фиксируете мысли, пожелания, идеи стейкхолдеров, то это еще не выявление требований — это сбор требований. Процесс выявления требований включает в себя анализ полученных сведений.

Не ждите, что заказчик сам сформулирует вам требования по нагрузке системы, а пользователь расскажет, какую навигацию реализовать в системе. Задавайте косвенные вопросы, которые позволят вам сделать выводы. Например, уточните у заказчика, сколько сотрудников компании будут использовать новую программу; а у пользователя можно узнать, какие другие сайты и приложения он считает удобными.

Для получения полной спецификации мы рекомендуем использовать различные способы сбора и выявления требований.

1. **Интервью и фокус-группы**

Этот метод позволяет получить полезные сведения из первых уст. Вы можете уточнить пожелания по ходу разговора, проверить гипотезу и услышать детали, которые даже не предполагали. Это очень трудоемкий метод, и для получения реально полезной информации требуются определенные навыки. Не забывайте, что вы также можете попросить стейкхолдеров прислать вам материалы уже после общения. Возможно, это будут наброски интерфейса от заказчика или проблемы, которые пользователь сначала не озвучил.

1. **Опросы и анкетирование**

Метод анкетирования позволяет собрать больше информации от большего числа респондентов, но он не даст вам сведения о чем-то неявном, о том, что можно обнаружить лишь в устном разговоре.

1. **Наблюдение и анализ деятельности**

Работа в поле состоит из наблюдения за деятельностью и процессами будущих пользователей системы, а также в определении требований, основанных на этом наблюдении. Если говорить проще, то это наблюдение за тем, как работают пользователи, и документирование процесса задач и результатов их деятельности. Такой метод позволит отстраниться от личных предпочтений конкретного пользователя, но при этом вы можете пропустить какие-то особые случаи, которые сложно заметить в ежедневной работе.

1. **Изучение существующей системы и анализ документов**

Изучение существующей документации может быть полезно для понимания процесса, правил и ограничений, которые важно реализовать в системе. Это могут быть регламенты, описание процессов, структура организации, различные процедуры, стандарты, инструкции, шаблоны документов и так далее.

Если у вас работающая система или хотя бы прототип, вы можете провести юзабилити-тестирование и получить обратную связь от пользователей. В некоторых случаях очень хорошие идеи приходят при анализе данных — при веб-аналитике.

Собранную информацию необходимо проанализировать, систематизировать, структурировать:

* Создайте набросок или прототип, чтобы понять, достаточно ли у вас сведений, чтобы проектировать интерфейс, что вы могли упустить при общении;
* Опишите функции с помощью пользовательских историй;
* Создайте диаграмму вариантов использования;
* Напишите сценарий.

Очень часто аналитики создают артефакты анализа, например, диаграммы и наброски — будто бы для других. Но помните, что в процессе такой работы каждый из нас проверяет свои результаты работы. Для аналитика это хороший метод дополнить спецификацию потерянными и упущенными требованиями.

Каждый продукт по-своему уникален. При этом существует множество похожих продуктов: реализация может повторять отдельные кусочки других систем, собираться как пазл в единое решение. Поэтому вы также можете использовать свой предыдущий опыт и опыт своих коллег. Иногда полезно пересмотреть ваши прошлые спецификации, учесть ошибки и взять хорошие примеры. Ваши коллеги могут также подсказать неявные моменты, с которыми они уже сталкивались. Например, если вы уже делали интеграцию с Госуслугами на одном проекте, то в следующий раз вам будет проще. Если вы знаете, как лучше отображать товары в интернет-магазине, то в следующий раз сделаете лучше.

С осторожностью копируйте документы, так как вы можете перенести ошибки и требования, которые нерелевантны для новой системы. Прошлый опыт для вас — это опора и способ выработать свой стиль оформления спецификации.

Выявление требований — непрерывный процесс. Конечно, для нового продукта наиболее интенсивно проводится сбор требований на старте проекта или при описании детальных требований. Но не пренебрегайте исследованиями и в ходе разработки. Даже при наличии исследований и обоснованных решений мы не можем сделать однозначный прогноз об успехе проекта. Иногда случай или вовремя собранная обратная связь могут круто повернуть путь проекта в новое русло и открыть новые возможности.

При разработке программного обеспечения **функциональные требования** определяют функции, которые должно выполнять все приложения или только один из его компонентов.

Функция состоит из трех шагов:

1. ввод данных;
2. поведение системы;
3. вывод данных.

Она может вычислять, манипулировать данными, выполнять бизнес-процессы, устанавливать взаимодействие с пользователем или выполнять любые другие задачи. Другими словами, функциональное требование — это то, что приложение должно или не должно делать после ввода некоторых данных.

Функциональные требования важны, поскольку они показывают разработчикам программного обеспечения то, как должна вести себя система. Если система не соответствует функциональным требованиям, значит она не работает должным образом.

Функциональные требования можно разделить на три группы:

1. **Бизнес-требования**

Эта группа подразумевает описание целей и ожиданий проекта, преимуществ, которые может принести проект, возможных ограничений проекта и его объем.

1. **Требования пользователя**

Включают потребности пользователя и действия, которые пользователь сможет выполнять в системе.

1. **Системные требования**

Подразумевают системные действия, спецификации программного и аппаратного обеспечения и так далее.

**Нефункциональные требования** определяют стандарты производительности и атрибуты качества программного обеспечения. Например, удобство использования системы, эффективность, безопасность, масштабируемость и так далее.

Проще говоря, функциональные требования определяют, **что** делает система, а нефункциональные требования описывают, **как** система это делает.

Например, веб-приложение должно обрабатывать более 15 миллионов пользователей без какого-либо снижения производительности, или веб-сайт не должен прогружаться более трех секунд. Если приложение не соответствует нефункциональным требованиям, оно продолжает выполнять свои основные функции, однако не может обеспечить удобство для пользователя.

Нефункциональные требования важны, поскольку они помогают разработчикам программного обеспечения определять возможности и ограничения системы, которые необходимы для разработки высококачественного программного обеспечения. Следовательно, не функциональные требования так же важны, как и функциональные, для успешного внедрения продукта.

Нефункциональные требования подпадают под различные категории:

* **Удобство пользования** определяет, насколько легко пользователь может взаимодействовать с интерфейсом приложения. Сюда входит цвет экрана, размер кнопок и так далее.
* **Доступность** гарантирует, что приложение будет стабильно работать в течение определенного периода времени. Например, редкие простои в течение года 24/7.
* **Надежность** определяет, что приложение будет работать в определенной среде или в течение определенного периода времени без сбоев.
* **Восстанавливаемость** гарантирует, что приложение сможет восстановить все данные после сбоя системы или восстановить систему до определенных параметров.
* **Масштабируемость** определяет, что приложение будет продолжать работать должным образом после изменения его размера или объема.
* **Производительность** оценивает, насколько быстро работает приложение.
* **Возможность поддержки** определяет, легко ли поддерживать приложение на протяжении всего его жизненного цикла и какая поддержка ему требуется — например, собственная команда или удаленная поддержка.
* **Безопасность** определяет, насколько безопасным должно быть приложение. Например, FinTech и банковские приложения должны соответствовать международным и региональным стандартам безопасности.
* **Емкость** оценивает объем данных или служб, которые может обрабатывать приложение.

Четко определенные функциональные и нефункциональные требования помогают разработчикам программного обеспечения создавать продукт, точно соответствующий потребностям клиента.

Однако действительно ли необходимо знать разницу между функциональными и нефункциональными требованиями? Дело в том, что они определяют объем работ по проекту. Разработчики программного обеспечения должны идти в ногу с этим объемом, чтобы разработать приложения в рамках своих сроков и бюджета. Если объем работ постоянно меняется, команде разработчиков приходится продлевать сроки — затраты на разработку возрастают, а это может привести к неблагоприятным последствиям для проекта.

Вы уже знаете про стейкхолдеров проекта. Теперь давайте взглянем чуть шире на **заинтересованные стороны**. Это люди или их группы, для которых ваш проект имеет значение или которые могут на него повлиять. Обычно не возникает проблем с выявлением очевидных заинтересованных сторон — заказчик проекта, команды его создателей, будущие пользователи. Но есть и менее очевидные, например, конкуренты. Чтобы вы могли выявить полный перечень заинтересованных сторон, рекомендуем поискать людей среди групп, перечисленных на экране.



Вероятно, многие из тех, кого вы вспомните, незначительно связаны с проектом, но есть и наоборот, те, кто имеет максимальное влияние на проект — заказчик или инвестор. Они проект финансируют, заинтересованы в его коммерческом успехе. Они, как правило, имеют возможность сильного влияния на проект, то есть их требования должны учитываться с высоким приоритетом или даже в первую очередь.

Для успешного выполнения проекта важно изучить требования и ожидания всех заинтересованных сторон. Фреймворк для управления проектами PMBoK предлагает такой **список возможных заинтересованных лиц**:

* спонсор или заказчик проекта;
* пользователи;
* поставщики и подрядчики;
* деловые партнеры;
* подразделение компании-заказчика и их руководителя;
* финансовые институты;
* правительственные регулирующие органы;
* эксперты по предметной области.

Используйте этот список для проверки тех направлений, в которых стоило бы поискать заинтересованных лиц и для вашего проекта. Некоторые требования могут оказаться противоречивыми. Здесь речь даже не о том, что конкуренты хотят провала, а заказчик успеха для проекта. Даже у положительно заинтересованных сторон требования и ожидания могут не совпадать. Например, из пользовательских историй вы можете узнать, что посетители будущего интернет-магазина ждут лаконичного современного светлого интерфейса, но при этом заказчик из-за индивидуальных предпочтений настаивает на тёмном интерфейсе с большим количеством графики.

**Что делать в такой ситуации?**

Для начала — искать компромисс, пытаться скорректировать требования сторон. Вы представите заказчику концепцию, в ней покажете конкретные пользовательские истории, подтвердите, что они репрезентативны. Возможно, после этого заказчик изменит свои требования. В то же время вы можете нарисовать прототип или пример дизайна с темным оформлением и обсудить его с некоторыми пользователями повторно. Возможно и такой вариант им тоже покажется удобным.

Но что, если никто не скорректирует требования? В данном примере можно на сайте реализовать две цветовые темы — темную и светлую — и сделать возможность переключения между ними. Тогда заказчик сможет смотреть сайт так, как ему нравится — с тёмной темой, а пользователи в том виде, как удобнее им — светлая тема. Но такая реализация потребует дополнительных расходов — часов работы программистов и дизайнеров. Хорошо, если заказчик на эти расходы согласится. Но если нет, то вам все-таки придется делать тяжелый выбор между требованиями заказчика и ожиданиями пользователей.

Чтобы выстроить отношения с заинтересованными сторонами, недостаточно составить их список. Нужно решить, с какой периодичностью, в каком объеме, каким способом вы будете взаимодействовать. Например, с конкурентами вы можете напрямую не взаимодействовать вообще, но можете следить за их реакцией после запуска проекта — усилят ли они рекламу, начнут ли они модернизировать свой сайт в ответ на новую версию вашего. С приоритетными заинтересованными лицами, например, заказчиком, взаимодействовать надо проактивно — предлагать решение регулярно и в достаточном объеме. Ресурс менеджера проекта тоже ограничен, именно поэтому менеджер должен заранее намечать взаимодействия, закладывать на них время, обеспечивать для них возможность.

# **Опасные и неочевидные заинтересованные стороны**

Как вы уже знаете не все заинтересованные стороны хотят, чтобы проект получился таким, как вы задумали и представили в концепции.

По разным причинам, в том числе из благих побуждений, проект могут саботировать:

* **Отдельные члены вашей команды.** Например, разработчик хочет попробовать новую технологию и настаивает на её применении, но никакие требования к этой технологии не располагают;
* **Отдельные представители заказчика.** Например, системный администратор заказчика знает, что сайт придётся обслуживать ему, но для этого придётся изучать документацию, менять какие-то технические процессы. Администратору проще оставить старый сайт;
* **Другие неочевидные заинтересованные стороны.** Например, вы создаёте новый сайт для производителя пластиковых изделий, чтобы он с помощью сайта и интернет-рекламы стал продавать больше продукции и нарастил производство. Но местному экологическому объединению не нравится всё, что связано с ростом пластикового производства, потому что предприятие заказчика не использует вторсырьё и, по мнению экологов, загрязняет природу.

Полезно для каждой заинтересованной стороны проанализировать **интересы** и **влияние**.

Рассмотрим подробнее пример с системным администратором:

* **Интересы:** не хочет получить дополнительных задач, связанных с переходом компании на новый сайт
* **Влияние:** значительное (много и напрямую общается с генеральным директором заказчика, может его переубедить), будет участвовать в приёмке сайта, может затягивать процесс

Даже такой базовый анализ поможет вам, в большинстве случаев, прийти к решению.

Например, с системным администратором бесполезно спорить и ссориться, но можно предложить дополнительную поддержку сайта как отдельную услугу (тогда новый сайт не станет задачей администратора) или заранее показать администратору примеры инструкций, которые вы создавали для других клиентов, чтобы он понял, что получит документированный продукт.

# **Инвесторы и соинвесторы как заинтересованные стороны**

Эталонная история получения инвестиций выглядит так: некая ИТ-компания создаёт или развивает востребованный продукт (полезное веб-приложение, посещаемый сайт), для существенного прогресса ей надо больше ресурсов (например, вдвое увеличить штат), но прибыли от продукта недостаточно, чтобы эти ресурсы приобрести. Поэтому компания приглашает инвестора, который, за долю в уставном капитале, предоставляет большой объём средств на развитие, в котором, как совладелец компании, теперь и сам заинтересован.

На практике существует масса причин, по которым такой путь невозможен или затруднителен:

* Продукт не кажется инвесторам таким уж востребованным, как считала компания;
* Инвесторы требуют настолько большую долю, что фактически смогут контролировать компанию;
* Инвесторам в целом нравится продукт, но они хотят его изменить и доработать в том направлении, которое не разделяет команда проекта;
* Инвесторы требуют аналитики и отчётности, а у компании таких данных с нужной детализацией ещё нет.

И только в том случае, если компания с помощью своего продукта уже генерирует выручку (а в идеале — и какую-то прибыль), она может привлекать портфельные (финансовые) инвестиции через широкий рынок акций, облигаций, других ценных бумаг. Такие инвестиции позволят команде сохранить контроль над компанией и продуктом. Но обычно это не вариант для стартапов.

Стартапам важно понять, что в современных условиях мало кто из инвесторов готов вкладываться в идею. Ведь даже хорошую идею кто-то должен качественно воплотить. Поэтому инвестору комфортнее выбирать не идею, а **команду** с готовым **MVP**.

Инвестор может вложиться не только деньгами, но и материальными ценностями, экспертизой, даже своими связями. Например, если в компанию приходит инвестор, имеющий налаженные контакты с потенциальными заказчиками, то он может для продукта обеспечить сбыт.

Люди из команды проекта — могут быть соинвесторами. Например, работая бесплатно или за небольшую зарплату, они вкладывают в стартап свой труд, в надежде окупить эти вложения, когда и если стартап, что называется, взлетит. При таком подходе важно, чтобы ожидания и договоренности были зафиксированы заранее.

Инвестор может быть не только частным лицом, но и фондом (в том числе государственным), а также другой компанией (даже конкурирующей, которая может инвестировать с целью дальнейшего поглощения).

Существуют гранты, региональные и федеральные целевые программы из которых тоже можно получить инвестиции. Как правило, больше шансов на такие инвестиции у социально значимых веб-проектов. Работа с госинвестициями требует объёмной отчётности.

Любого коммерческого инвестора будут интересовать:

* Вложения в проект со стороны создателей. Если люди верят в свой проект, то они вложатся в него сами;
* Полноценная команда, которая может и хочет развивать продукт;
* Планы и возможности для роста продукта;
* Наличие рынка для продукта и исследования со стороны команды, которые бы подтверждали потребности и ёмкость рынка сбыта;
* Презентация команды и продукта.

# **Устав проекта как базис отношений со стейкхолдерами**

Определение устава согласно PMBoK («Своду знаний по управлению проектами» от международного Института управления проектами) таково:

**Устав проекта** (англ. Project Charter) — это документ, выпускаемый инициатором или спонсором проекта, который формально авторизует существование проекта и предоставляет руководителю проекта полномочия использовать ресурсы организации в операциях проекта.

По факту и в практике небольших и даже средних проектов, устав — это общий обзор проекта **с фиксацией основных параметров**, но в более формальной форме, чем это делает концепция.

Если концепция подходит для целей презентации проекта, то устав — больше подходит для включения в договор в роли приложения.

В уставе не требуются иллюстрации (наброски, CJM и прочее), в устав не включаются пользовательские истории.

Зато там обязательно содержатся существенные договоренности между спонсором и менеджером проекта. Под спонсором здесь понимается лицо или организация, которая оплачивает проект. Обычно спонсор совпадает с заказчиком.

## **Какие пункты обычно включаются в устав:**

* **Обоснование проекта**
  + Какие проблемы решает проект
  + В чём основная идея проекта или продукта
  + Какие цели у проекта
  + Каковы критерии достижения этих целей
  + Заинтересованные лица (включая клиентов). Краткое описание (скорее даже перечисление), без пользовательских историй
* **Определение проекта**
  + Какие результаты получим (какие основные фичи будут у продукта)
  + Стоимость проекта, порядок оплаты
  + Сроки проекта, основные вехи (контрольные точки в виде дат и кратко сформулированных результатов, которые выдаст к этим датам команда проекта)
* **Оргструктура**
  + Кто заказывает
  + Кто руководит (менеджер с контактами)
  + Кто исполняет (команда исполнителей, роли или даже конкретные лица с контактами)
  + Кто принимает (ответственные со стороны заказчика, роли или даже конкретные лица с контактами)
* **Допущения и ограничения**
  + В каких условиях будет эксплуатироваться продукт
  + Какие есть риски
  + Как на наступление этих рисков готовы реагировать стороны
  + За что отвечает и за что НЕ отвечает исполнитель

В идеале, устав проекта не должен изменяться по ходу реализации проекта.

Достаточно важным является последний раздел «Допущения и ограничения», поскольку он регулирует развитие проекта в неблагоприятных условиях.

# **Матрица распределения ответственности в проекте**

Матрица распределения ответственности (англ. RACI) — это распространенный способ определения ответственности по любой задаче, фиче, результату проекта.

* **Исполнитель задачи** или **Ответственное лицо (R — Responsible).** Это человек, который непосредственно выполняет задачу.
* **Постановщик задачи (A — Accountable).** Это человек, который принимает результат, утверждает задачу как выполненную или возвращает на доработку.
* **Консультант** или **Соисполнитель (C — Consulted).** Это человек, который помогает выполнять задачу. У задачи таких людей может быть несколько, а может не быть совсем. Тогда над задачей будет работать только исполнитель.
* **Информируемое лицо (I — Informed).** Это человек или группа лиц, которых информируют о ходе и завершении задачи. Таких специалистов может быть несколько, а может не быть совсем. Эти люди не выполняют задачу, но должны быть в курсе. Например, потому далее им самим предстоит делать другую задачу на основании результатов текущей.

Не все элементы матрицы обязательны, но у любой задачи есть постановщик и исполнитель.

Если вы имеете дело не с конкретной задачей, а с целой фичей или большой функцией продукта, над которой поработают сразу несколько участников команды, то матрицу RACI надо тоже определять на уровне отделов или команд, а не на уровне конкретных людей. Иначе вы нарушимте важный принцип о том, что ответственный (R) должен быть один.

Для простоты рассмотрим пример конкретной задачи:

В интернет-магазине игрушек при оформлении заказа есть выбор варианта оплаты, включая вариант «Оплата курьеру при получении». Необходимо при выборе этого варианта пользователем показывать уведомление с текстом «Пожалуйста, приготовьте для оплаты карту или сумму наличными без сдачи». Уведомление должно показываться только при первом выборе этого варианта (при повторном выборе снова показывать уведомление не нужно).

По такой задаче роли будут иметь следующую ответственность:

Исполнителем (R) по этой задаче будет программист, но ему потребуется небольшая консультация (C) дизайнера, чтобы тот порекомендовал какой из вариантов оформления уведомлений лучше использовать в корзине. Менеджер эту задачу поставил и будет принимать (A). Аналитик и маркетолог должны быть просто в курсе (I), что мы предупреждаем клиентов заранее об особенностях оплаты наличными.

Матрица RACI может составляться сразу для несколько задач в виде таблицы. Изучите пример с несколькими задачами, одна из которых вам уже известна:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Менеджер** | **Аналитик** | **Дизайнер** | **Разработчик** | **Маркетолог** |
| Добавить уведомление при оплате курьеру | A | I | C | R | I |
| Нарисовать слайд про новогоднюю акцию | A |  | R |  | C |
| Исправить баг: в корзине можно изменить кол-во товара на отрицательное | I | A | R |  |  |
| Запустить поисковую кампанию в Директе по товарам из новогодней акции | A |  | C |  | R |

# **Информационные каналы для коммуникации**

В любом проекте важна коммуникация:

* Между менеджером и заказчиком. При этом со стороны заказчика может быть несколько представителей с разными компетенциями, полномочиями, влиянием на проект
* Между менеджером и командой
* Между всеми членами команды напрямую (минуя менеджера)
* Между аналитиками, маркетологами и пользователями, клиентами продукта
* Между конкретными представителями исполнителя и конкретными представителями заказчика по узкоспециализированным вопросам (например, разработчику проще пообщаться напрямую с сисадмином заказчика, чтобы решить какие-то вопросы по интеграции систем)

Одна из задач эффективного управления проектом состоит в том, чтобы все эти коммуникации подготавливать и контролировать.

Очевидное, но плохое решение: собрать всех в один громадный общий чат и пытаться там устраивать общие созвоны по любому вопросу. Никогда так не делайте!

**Менеджер должен выбирать удобные средства коммуникации для разных участников проекта**

Предположим, генеральный директор заказчика человек старой закалки, не любит видеоконференции, ценит личный контакт. Возможно, менеджеру стоит общаться с ним по телефону, а для решения самых ответственных вопросов (например, чтобы презентовать и утвердит концепцию) — даже приезжать офис.

Другой пример: сисадмин заказчика — интроверт, к тому же всё время занят текущими делами. Его тяжело затащить даже в Zoom. Поэтому лучше будет, чтобы разработчик из вашей команды написал сисадмину по email все вопросы об интеграции.

**Менеджер должен выстраивать последовательность коммуникаций**

На следующей неделе надо ехать к гендиру заказчика и обсуждать сроки проекта? Значит уже на этой (заранее) надо, чтобы разработчик списался с сисадмином и выяснил тонкости интеграции, иначе потом из-за неё придётся сдвигать сроки.

**Менеджер должен изолировать коммуникации**

Если команда горячо между собой обсуждает текущие баги, то этот диалог не надо видеть заказчику. Баги вы пофиксите за пару дней. А заказчику может показаться, что весь проект на грани провала.

# **Оценка требуемых ресурсов**

В развитие темы управления ресурсами процессов производства Информационных Систем (далее ИС), рассмотрим подробнее вопрос об оценке самого смысла «браться за ИТ-проект».

Начнем с осознания того, какие впринципе вызовы перед нами возникают при рассмотрении возможности стартовать новому ИТ-проекту.

* Нет четкого представления о незакрытых проблемах заказчика;
* Нет понимания, кто может быть заинтересован в автоматизации и какие потребности у заинтересованных лиц;
* Нет общего видения о сплетениях деловых активностей, которые могут рассматриваться, как область автоматизации;
* Нет понимания, как должна поменяться организация и ее текущие деловые активности, при автоматизации.

И во главе всего стоит самая насущная проблема:

* Нет понимания, какие ресурсы понадобятся на реализацию проекта автоматизации;

Для ответа на эти вызовы обычно организуют 3 этапа подготовки проекта:

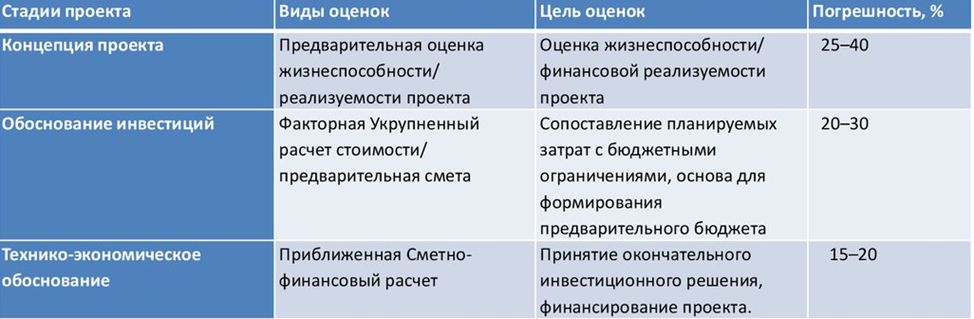
·       Этап № 1 – предпроектное исследование и анализ потребностей клиента

·       Этап № 2 – pre sale. оценка и планирование будущего проекта для создания выгодного предложения как для клиента, так и самой компании.

·       Этап № 3 – подготовка коммерческого предложения

На первом этапе оценка проводится без технического задания.

В этом случае аналитики, используя полученную в ходе обсуждения с клиентом информацию и свой практический опыт, составляют **Видение** проекта, и техническую **Концепцию** с разбивкой по фичам для оценки. При этом, чем более абстрактны требования, тем выше погрешность оценки ресурсоемкости. См.Таб.1.



Понятно, что чем детальнее на этапе 1 проработаны Видение и Концепция проекта, на которых основана оценка, тем точнее может быть дан ответ: во что обойдется проект автоматизации. Но тут возникает дилемма: с одной стороны, мы можем затратить очень много ресурсов на эти работы, при отсутствии гарантий того, что проект вообще будет реализовываться. С другой, если сэкономим ресурсы и сильно ошибемся в прогнозах затрат, то решение о его старте может стать фатальным. Решение подобной проблемы на практике часто решают привлечением специалистов с очень большим опытом, способных, основываясь на прошлых решениях, принимать максимально оперативно верные решения.

Когда выявлен предварительный перечень задач исполнителям и потребности в оборудовании, начинается Этап 2. Цель этих работ – заключение сделки на условиях максимально выгодных для обеих сторон: заказчика (клиента) и поставщика (IT-компания).

К работе подключаются пресейлы направлений – специалисты, которые будут делать оценку ИТ-проекта по своей части. Создается коммуникационный канал, к которому подключаются все необходимые эксперты. Для этих расчетов могут быть привлечены следующие специалисты;

Код 06.022   Системный аналитик

Код 08.037   Бизнес-аналитик

Код 07.007   Специалист по процессному управлению

Код 06.012   Менеджер продуктов в области информационных технологий

Код 06.003   Архитектор программного обеспечения

Код 06.032   Специалист по безопасности компьютерных систем и сетей

Код 06.041   Специалист по интеграции прикладных решений

После того как все оценщики завершили свою часть работы, проводится проверка оценки. Далее готовится описание и на основе сметы – примерный состав будущей команды ИТ-проекта, сроки реализации или roadmap.

Как это может происходить?

На рис.1 и 2 представлен пример, как может выглядеть концептуальное представление требований к ИС.

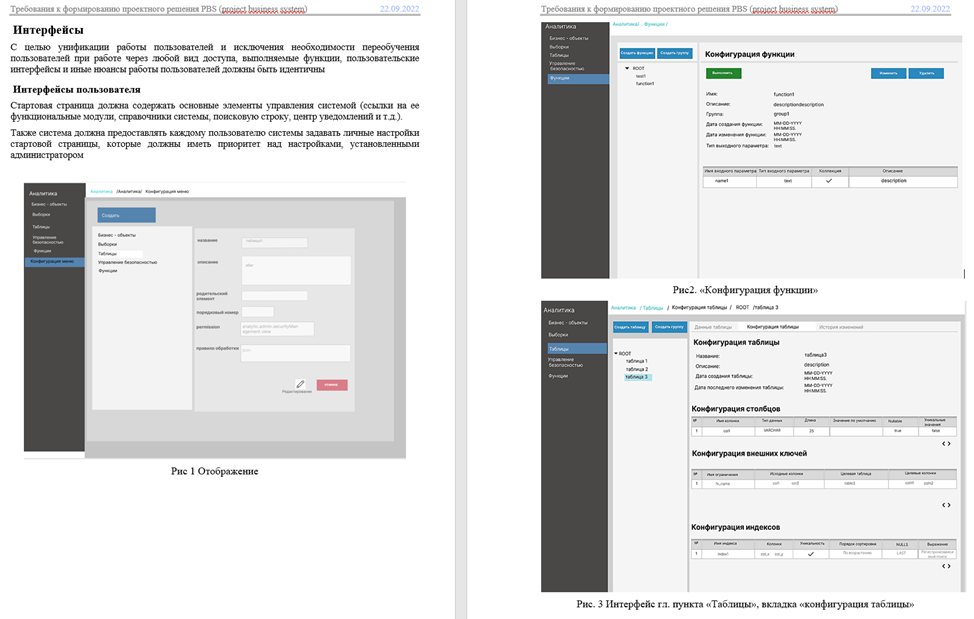


Рисунок 1 – высокоуровневые требования к формам ИС

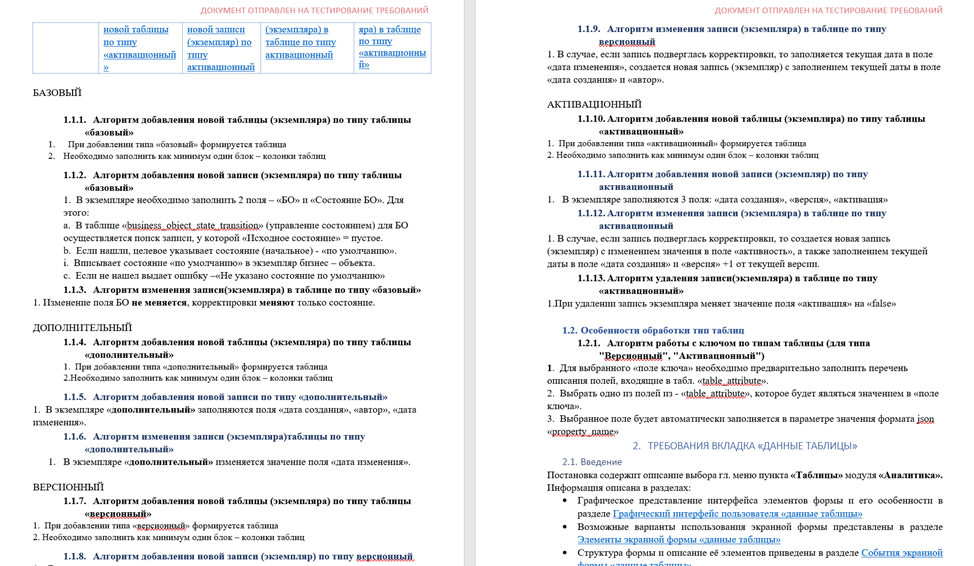


Рисунок 2 – высокоуровневые требования к алгоритмам и процедурам ИС

В таком документе, например, могут быть описаны представления визуальных форм и событий- реакции на взаимодействие с пользователем, описаны бизнес-процессы, внедренные на предприятии, приложены формы требуемых отчетов, указаны ограничения в доступе к данным и прочее.

В результате можно формализовать предварительный перечень объектов ИС, который необходимо реализовать, а также уровень сложности реализации.

Например:

1. **Экранных форм**: 5 сложных, 3 средних, 10 простых;
2. **Алгоритмов** (функций): 3 сложных, 2 простых;
3. **Бизнес-процессов**: 2 средних;
4. **Сущностей** (предметной области): 5 сложных, 12 средних, 8 простых;
5. **Отчетов**: 12 средних;
6. **Ролей** (набора прав доступа пользователей): 5.
7. **Оборудование**: Сервера – 2 компл., датчики – 8 шт.
8. …

Выигрышно, когда ИТ-предприятие разработало свои нормативы выполнения работ, где перечислены основные проектные работы ИТ-команды, квалификация специалиста, способного их выполнить и плановое время реализации (активности, воспроизводимые из проекта в проект). см. Таблица 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **П/№** | **Тип работы** | **Наименование работы** | **Квалификация исполнителя** | **Трудоём.** |
| 1 | Автоматизация шаблонного процесса | Инжениринг и реинжениринг бизнес-процесса. | Senior системный аналитик | 48 ч. |
| 2 | Создание BPMN диаграммы | Midle системный аналитик | 48 ч. |
| 3 | Документирование бизнес-процесса | Junior системный аналитик | 8 ч. |
| 4 | Разработка GUI | Создание мокапа формы | Junior системный аналитик | 6 ч. |
| 5 | Реализация модальной формы | Junior Front-end разработчик | 4 ч. |
| 6 | Реализация Блока  модуля иерархического меню | Midle Front-end разработчик | 6 ч. |
| 7 | Реализация страницы с таблицей, блоком поиска и фильтраций, пейджингом. | Midle Front-end разработчик | 16 ч. |
| 8 | Реализация страницы с дашбордом элементов системы | Senior Front-end разработчик | 24 ч. |
| 9 | Реализация API | разработка интерфейса поодержки GUI карточки бизнес-объекта | Midle Back-end разработчик | 12 ч |
| 10 | Реализация алгоритма | Создание процедуры обработки данных | Midle Back-end разработчик | 6 ч |

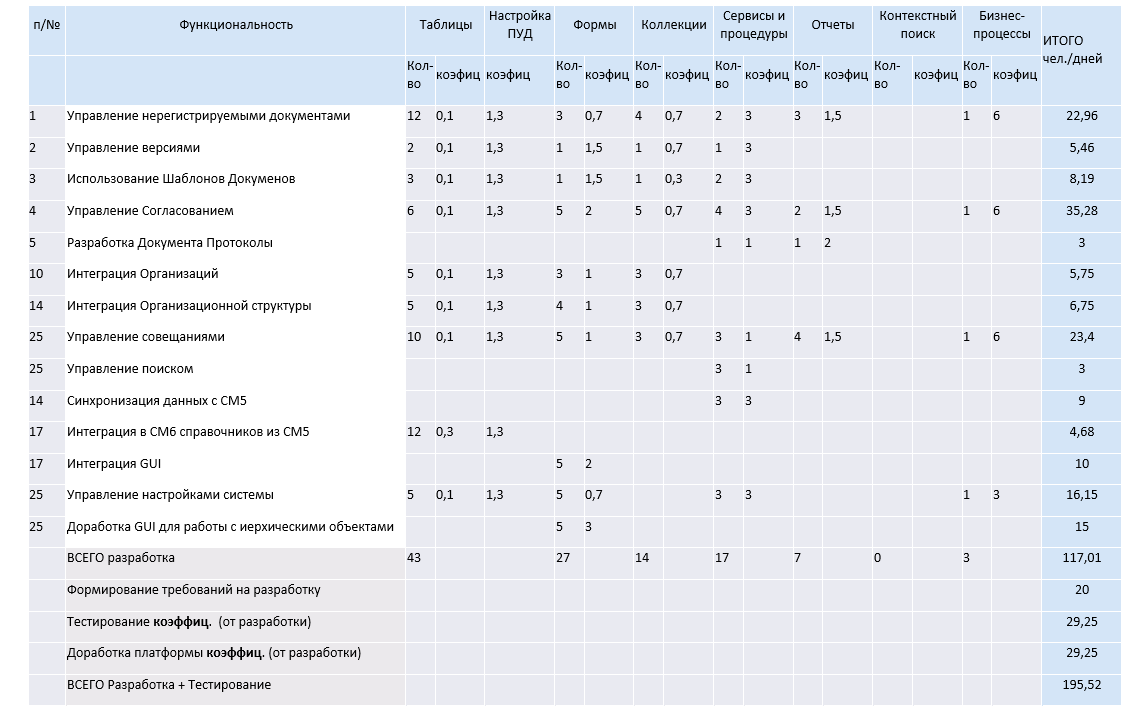
Накладывая на перечень выявленных работ, предполагаемое время их реализации специалистом определенной квалификации из норматива, можно получить предварительный план затрат.

Ниже приведен пример формирования предварительной сметы работ по реализации проекта. В колонке «Функции» перечислены все подсистемы (модули), которые необходимо реализовать. Далее следуют колонки с типовыми программными объектами, которые необходимо реализовать в коде, для обеспечения функционирования: Таблицы (БД), Настройки ПУД (Подсистемы Управления Доступом), Формы (GUI), Коллекции (выборки данных из БД), Сервисы и процедуры, Отчеты, Контекстный поиск (организации системы поиска), Бизнес-процессы. При необходимости колонки можно добавить.

Каждая колонка в свою очередь делится на 2 столбца -количественное представление и коэффициент сложности реализации, для пересчета количества программных объектов в человеко-часы (при умножении количества на коэффициент должны получиться человеко-часы трудозатрат).

Таким образом мы можем вычислить приблизительное количество чел./часов, которое потребуется для реализации каждого модуля.

Дальше добавляются строки корректировок, для учета ресурсов на проектирование, тестирование, доработку системных элементов (платформы разработки), разворачивание серверов, взаимодействие с заказчиком и прочее.  Причем их можно вычислять по заданному коэффициенту от рассчитанной трудоемкости объема работ на разработку.



**Коэффиц**. Тестирования (от разработки)          0,25

**Коэффиц.** Доработки платформы (от разработки)       0,25

Коэффициенты для **Таблиц**: 0,1 - простой; 0,3 - средний; 0,5 - сложный

Коэффициенты для **ПУД**: 1,3 - простой; 1,7 - средний; 2 - сложный

Коэффициенты для **Форм**: 0,7 - простой; 1 - средний; 1,5 - сложный; 2 - очень сложная

Коэффициенты для **Сервисов**: 1 - простой; 3 - средний; 6 - сложный; 12 - очень сложный

Коэффициенты для **Отчетов**: 0,7 - простой; 1,5 - средний; 2 – сложный

По итогам проведенной оценки 2-ого этапа готовится коммерческое предложение и затем направляем клиенту. При необходимости группа оценщиков проводит защиту оценки и презентацию коммерческого предложения клиенту и его потенциальным инвесторам.

В этом процессе могут участвовать специалисты:

* Код 08.029   Менеджер по продажам информационно-коммуникационных систем;

Бывают кейсы, когда после ознакомления с оценкой клиент вносит корректировки: или изменились требования, или надо выделить MVP, уложиться в рамки изменившегося бюджета или сроки.

Перед стартом работ подобранная на проект команда вместе с клиентом еще раз проверяет оценку фич.

Для планирования требуемых ресурсов составляют Ресурсный план.

*:info:***Ресурсный план** — это перечень ресурсов проекта, распределенных по плану-графику проекта в зависимости от задач.

Используется для управления ресурсами проекта. В ресурсный план входит только те ресурсы, которыми может управлять руководитель проекта. К данным ресурсам относится: материальные ресурсы, трудовые ресурсы, оборудование.

Для удобства необходимо классифицировать ресурсы по характеристикам, накладывающим ограничения на их использование. В ИТ-сфере их принято делить на:

* Трудовые;
* Материальные;
* Затратные.

**Трудовые ресурсы**

*:info:***Стафинг** — это процесс подбора и согласование ресурсов на вакантные позиции ресурсного плана. Проще говоря, стафинг — это подбор команды на проект.

Очень редко бывает, что руководитель проекта получает на свой проект сотрудников, полностью соответствующих сформулированным требованиям. Также не всегда удается получить людей к нужному сроку или на нужное время. Поэтому план может видоизменяться в ходе стафинга. Если качество ресурсов хуже того, что закладывалось при оценке проекта, то, очевидно, увеличится количество часов для выполнения тех же работ.

Трудовые ресурсы являются воспроизводимыми, то есть они не складируемые, не накапливаемые, в ходе работы сохраняют свою натурально-вещественную форму и по мере высвобождения могут использоваться на других работах.

**Проблемы стафинга**

Необходимо учитывать, что проект развивается поэтапно и на каждом этапе нужны не все специалисты команды. Поэтому важно исключить простои специалистов в ходе проекта, когда они ожидают своей очереди или уже освободились от максимальной загрузки в проекте. Также специалисты могут быть задействованы на нескольких проектах и важно понимать объем их загрузки на каждом из проектов.

**Материальные ресурсы**

Материальные ресурсы, в свою очередь, рассчитываются исходя из количества, они, как правило, не возобновляемы.В данном случае это не только сырье и материалы, но и любые ресурсы, исчисляемые количественно.

В случае аутсорсинга какого-либо вида работ или найма подрядчика, хотя основной услугой и является труд (возобновляемый ресурс), менеджера проекта не интересует количество человек в бригаде и их ставки. Если используется модель Fixed Price, по которой клиенты оплачивают конечный результат, и стоимость оговорена заранее, целесообразно относить такие ресурсы к материальным.

Планирование материальных ресурсов коренным образом отличается от человеческих. Данный процесс производится на уровне нескольких подсистем управления проектами, а именно: управлении закупками, управлении поставками, а также логистики и управлении запасами. Невоспроизводимые ресурсы являются складируемыми, накапливаемыми и в процессе выполнения работ расходуются полностью, не допуская повторного использования.

**Затратные ресурсы**

Выделение затратных ресурсов связано с необходимостью моделирования затрат на те ресурсы, которые не связаны прямо с выполнением задач проекта, но должны быть учтены. Как правило, это потребность задачи в каком-либо финансовом обеспечении (стоимость авиабилета, представительские расходы и т.д.).

Ресурсы, необходимые для покрытия любых издержек, связанных с проектом, принято относить к финансовым.

Менеджер проекта, согласно профессиональному стандарту, должен выполнять:

* **планирование управления персоналом** в проектах любого уровня сложности в области ИТ;
* **привлечение (набор) персонала** в проектах любого уровня сложности в области ИТ;
* **управление эффективностью работы команды** в проектах любого уровня сложности в области ИТ;
* **планирование закупок** в проектах любого уровня сложности в области ИТ;
* **подбор субподрядчиков** в проектах любого уровня сложности в области ИТ;
* **выбор поставщиков** в проектах любого уровня сложности в области ИТ.

# **Обсуждение с заинтересованными сторонами**

При взаимодействии с заинтересованными лицами проекта могут возникнуть следующие проблемы:

* Начальный запрос клиента не соответствует его реальным потребностям и подготовленному проектному решению;
* Ограниченный бюджет заказчика;
* Технические и технологические ограничения заказчика;
* Не полное понимание клиентом предлагаемого решения.

Важно решить все эти проблемы еще до старта проекта, внеся поправки в планы его реализации при необходимости.

В этом процессе могут участвовать специалисты:

Код 08.029   Менеджер по продажам информационно-коммуникационных систем;

Код 06.012   Менеджер продуктов в области информационных технологий

Код 07.007   Специалист по процессному управлению

# **Принятие решения о запуске или остановке проекта**

**Утверждение проекта заказчиком**

Сложность согласования уже подготовленного предложения с заказчиком обуславливается целым рядом причин:

1. Отсутствие у заказчика ясного и четкого понимания того, что описано в документации (недостаточный технический уровень подготовки);
2. Боязнь представителей заказчика, возложить на себя ответственность за согласование решения, в котором они сомневаются (не до конца понимают);
3. Вероятность изменения требований в ходе производства продукта, способных повлечь за собой значительное, критическое удорожание утвержденной сметы проекта;

Необходимо учитывать эти факторы и нивелировать их воздействие.

**Подтверждение условий проекта командой исполнителя**

Выполняя анализ заинтересованных лиц проекта и их цели, команда, реализующая проект, должна четко учитывать свои цели и закладывать свои интересы в планы реализации проекта. При принятии решения о запуске проекта, ИТ-команда должна сверить соблюдение своих объявленных интересов.

Эти интересы могут включать не только оплату работ за реализацию проекта, согласно договорам, но и продукты или полуфабрикаты, полученные в ходе его исполнения, которые могут быть переиспользованы в других проектах или, например, имиджевая узнаваемость и прочее.

На этом занятии мы разберем с вами, как именно может происходить процесс инициации ИТ-проекта и как его организовать так, чтобы и заказчики остались довольны и исполнители сыты.

**Этап 1. Подготовка**

Для начала давайте рассмотрим обстоятельства, при которых он может зарождаться, варианты старта создания продукта.

1. **Потребность в программном продукте возникает у самого потенциального пользователя**.

У заказчика возникает потребность в автоматизации некоторых функций своей деятельности, при этом он неспособен самостоятельно и вразумительно выразить свои пожелания к программному продукту. И часто даже определиться с тем, какую пользу он должен ему принести. В такой ситуации необходимо прежде всего подчинить сумбурные и разрозненные «хотелки» клиента значимой цели, достижение которой принесёт ощутимый эффект от автоматизации деятельности клиента. При этом иногда необходимо применять приемы манипуляции с сознанием заказчика, чтобы вытащить его потаенные ощущения о продукте, которые он не может озвучить, но которые реально могут повысить эффективность его работы.

1. **Потенциальный заказчик ощущает потребность в своем развитии и имеет ресурсы для этого, но не понимает, какие конкретно действия ему необходимо предпринять.**

В подобных ситуациях либо клиента «обрабатывают» специалисты отдела продаж и убеждают, что ему необходим именно продаваемый ими продукт, либо с ним работает бизнес-аналитик и пытается выявить направление деятельности, автоматизация которого принесет ощутимую выгоду от внедрения.

Чаще всего бывает смешанный вариант: сначала сейлз продает какое-то ИТ-решение клиенту, а уже потом бизнес-аналитик натягивает проданное на действительные потребности клиента.

1. **Разработка больших программных продуктов с участием штатных сотрудников предприятия.**

Здесь само предприятие выступает одновременно и заказчиком, и исполнителем, и «кошельком». В этом случае разработка чаще всего ведется силами штатных программистов-могостаночников, которые выступают и в роли аналитика, и исполнителя, и внедренца. А в качестве тестировщика зачастую используются пользователи продукта — такой эконом-вариант. В подобной разновидности проекта качество автоматизации критично зависит от профподготовки ИТ-команды. Создавать быстро новые решения такая команда обычно не мотивирована.

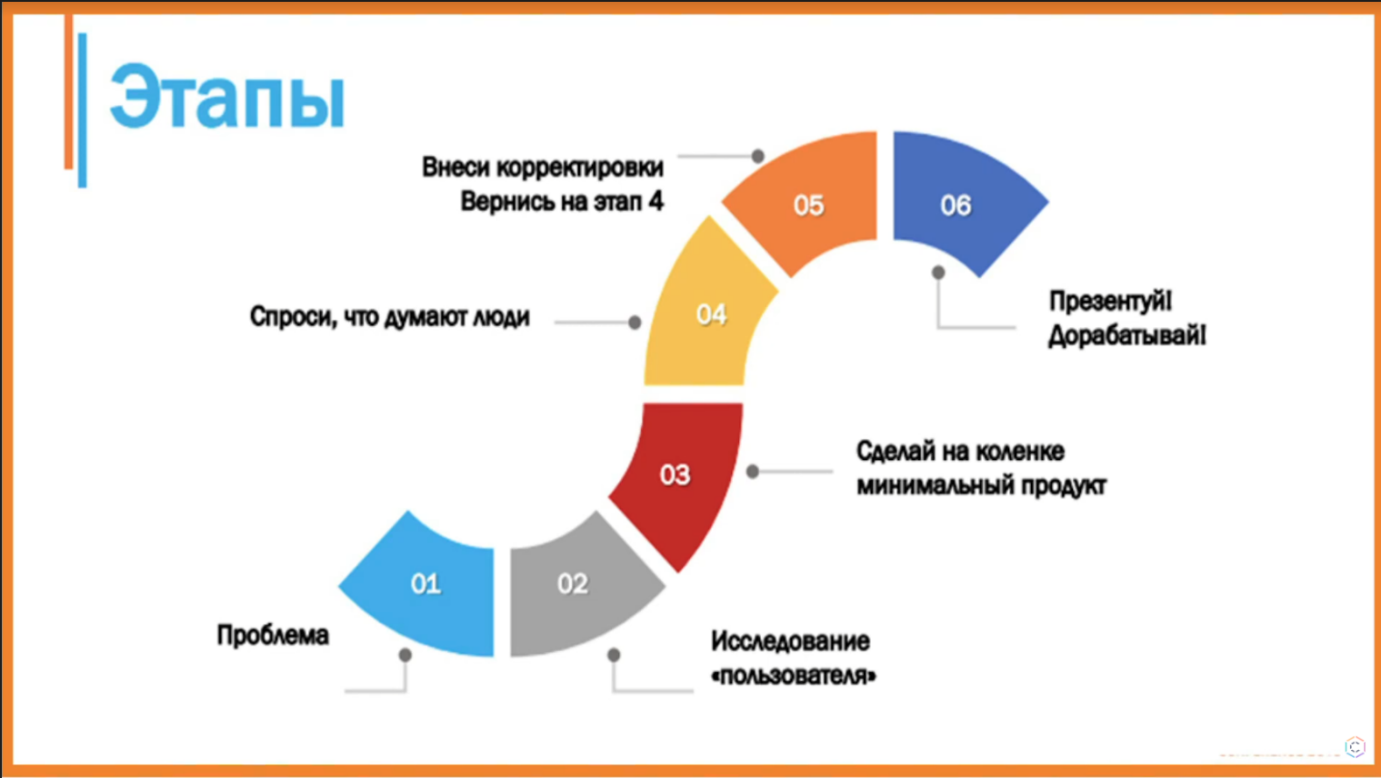
1. **Кто-то формулирует потребность в ПО от имени потенциальных пользователей.**

То есть креативный ИТ-предприниматель полагает, что он четко представляет, какой программный продукт не сегодня-завтра будет востребован каким-то сегментом сообщества, и они будут готовы за это будут платить. Развитие подобной альтернативы чаще всего называют **стартапом**. Этот вариант требует детальной оценки, поскольку он обычно устремлен на максимально большую аудиторию и ошибки в предположениях могут вызвать значительные отклонения от реальных потребностей большинства вероятных пользователей, а также в отклонения в оценке масштабов работ.

1. **Предложение закрывает задачи, боли, потребности целевой аудитории.**

Этот вариант похож предыдущий, но здесь действует уже не одинокий предприниматель, а профессиональная продуктовая команда, которая подгоняет разрабатываемый продукт под рынок, а не под контракт. При этом начальная концепция может существенно отличаться от финального решения. Команда проводит собственное исследование рынка, в этом случае формируется гипотеза о продукте.

Важная обязанность менеджера по продукту — превращение в гипотезы всех передовых идей, которые возникают в головах руководства, специалистов и своей собственной. Далее выполняется проверка гипотезы, путем сбора максимального количества отзывов потенциальных клиентов о востребованности продукта. Для этого создают сайт-лендинг, проводят онлайн-опросы, запускают телефонные опросы, реализует MVP — продукт с минимальными возможностями. В итоге на основании обратной связи о продукте производится анализ и новая оценка, корректируется решение и запускается новый цикл для дальнейшего совершенствования изделия. Так, небольшими шагами, продукт развивается в угоду рынку, а не амбиций исполнителей.



[Screenshot-86-29-12-22.png](https://drive.google.com/file/d/1C23D40sWBwUotJrKhz-x4UD2TxQmm69e/view?usp=drivesdk)

**Особенности подхода в зависимости от типологии внедряемых производственных изменений**

Что же заставляет ИТ-команды выбрать тот или иной путь? Принять такое решение можно на основании:

* оценки характеристик объекта автоматизации;
* готовности предприятия к изменениям;
* финансовых и материально-технических условий;
* прочих факторов.

Например, если цикл принятия решений на предприятии длинный: анализ, согласования, комиссии, утверждения, связанные с высокой ценой ошибки, а скорость получения обратной связи о продукте низкая, то необходимо четко проработать требования с высокой детализацией, составить подробный план-график и получить точную оценку его реализации. Ракету по Agile не построить.

Если же наоборот, необходимо реализовать инновационный продукт молодой энергичной команде с небольшим опытом и в короткие сроки вывести его на скромный благосклонный сегмент рынка, то стоит применить продуктовый подход с короткими циклами выпуска продукта с ограниченным, но продуктивным функционалом. При этом часть требований может составляться уже на этапе реализации, а оценка производится поэтапно.

Еще на выбор способа организации проекта влияет **величина требуемых инвестиций**. Например, можно начинать со стартапа «на своей кровные» или организовывать серию продуктовых проектов на средства заинтересованных инвесторов.

Итак, как это может происходить на деле? Вот мы ощутили потребность разработать новую информационную систему. У нас пока еще нет требований к ней, мы пока даже не представляем себе весь набор функционала, который должен решить проблемы заинтересованных лиц. Чаще всего мы даже не в состоянии эти проблемы четко сформулировать.

В ответ на этот вызов команда должна организовать работы, в результате которых будет сформировано предложение для заказчика, предполагающие при помощи разрабатываемой информационной системы решить его проблемы.

Также необходимо рассчитать, во сколько это предложение должно обойтись клиенту и какие затраты могут понести исполнители в течение периода реализации этого решения.

Есть три этапа решения:

1. **Предпроектное исследование и анализ потребностей клиента.**
2. **Pre-sale** — оценка и планирование будущего проекта для создания выгодного предложения как для клиента, так и для самой ИТ-компании.
3. **Подготовка коммерческого предложения.** Аналитики, используя полученную в ходе обсуждения с клиентами информацию и свой практический опыт, составляет видение проекта и техническую концепцию, с разбивкой по фичам для удобства дальнейшей оценки.

Понятно, что чем детальнее на шаге один проработаны видение и концепция, на которых основана оценка, тем точнее может быть дан ответ, во сколько обойдется проект автоматизации.

Но все не так однозначно, как кажется на первый взгляд. Возникает дилемма: с одной стороны, мы можем затратить очень много ресурсов на эти работы при отсутствии гарантии того, что проект вообще «взлетит». С другой стороны, если мы сэкономим свои затраты и из-за этого сильно ошибемся в оценке предстоящих расходов, то решение о старте проекта может стать фатальным и привести к убыткам.

Разрешение подобной проблемы на практике часто решается привлечением специалистов с очень большим опытом, способных на основании прошлых прецедентов принимать максимально сбалансированные верные решения.

**Этап 2. Инициация проекта**

Когда выявлен предварительный перечень задач и потребности в оборудовании, переходим к шагу два — инициации проекта. Цель этих работ — заключение сделки на условиях, максимально выгодных для обеих сторон, заказчика — клиента и поставщика — ИТ-компании.

К работе подключается пресейлы направлений — специалисты, которые будут давать оценку ИТ-проекта по своей части. Создается коммуникационный канал, к которому подключаются все необходимые эксперты.

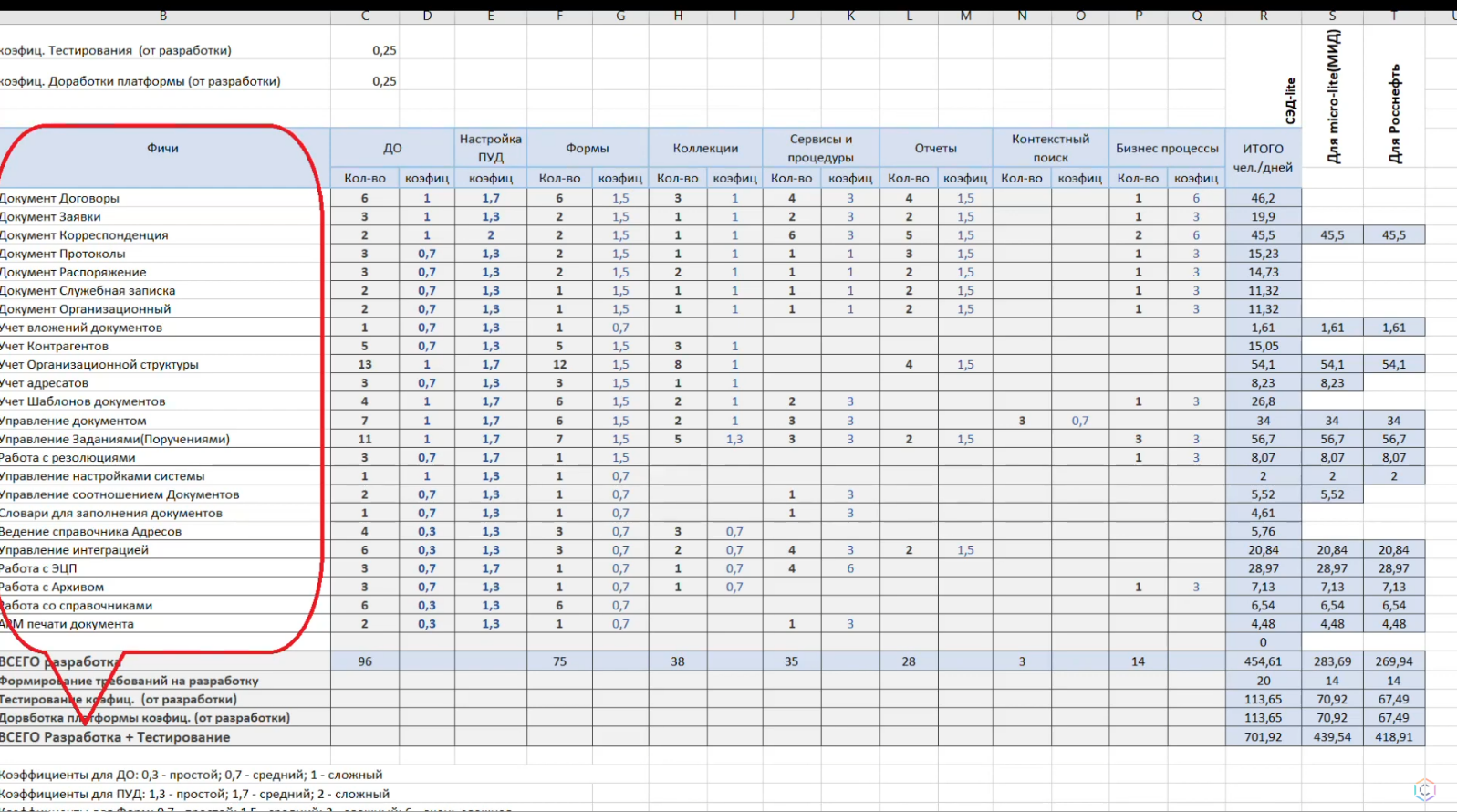
После того как все таксировщики завершат свою часть работы, проводится проверка оценки. Далее готовится описание на основе сметы: примерный состав будущей команды проекта, сроки реализации или road map.

Как это может происходить: если документ видения и концепция подготовлены качественно, то по ним можно определить примерный перечень объектов информационной системы, которые необходимо реализовать, а также уровень сложности их воплощения. В таком документе могут быть описаны представления визуальных форм и событий, реакции на взаимодействие с пользователем, описаны бизнес-процессы, внедряемые на предприятии, приложены формы требуемых отчетов, указаны ограничения в доступе к данным и прочее.

В результате, используя эту подробную информацию, можно выделить **предварительный перечень объектов информационной системы**, которые необходимо реализовать, а также уровень сложности их разработки. Например, экранных форм 5 сложных, 3 средних, 10 простых. Алгоритмов функций 3 сложных, 2 простых. Бизнес-процессов: два средних. Сущностей (предметной области): 5 сложных, 12 средних, 8 простых и так далее.

Теперь, чтобы **переложить количество элементов системы в трудозатраты** на их разработку, необходимо применить нормативы выполнения работ, принятые для команды. Нормативы должны устанавливать квалификацию специалиста, способного их выполнить, и плановое время реализации, уточняемое из проекта в проект.

Накладывая на перечень выявленных работ предполагаемое время их реализации специалистом определенной квалификации, можно получить примерный план затрат. Предварительная смета работ по реализации проекта может выглядеть так:



В колонке «Функции» или «Фичи» перечислены все возможности, которые необходимо воплотить. Далее следуют колонки с типовыми программными объектами, которые необходимо реализовать в коде для обеспечения функциональности: таблицы БД, настройки подсистемы управления доступом, формы выборки данных из БД, сервисы и процедуры, отчеты, ограничения систем поиска, бизнес-процессы. При необходимости колонки можно добавить.

Каждая колонка, в свою очередь, делится на два столбца: количественное представление и коэффициент сложности реализации, для пересчета количества программных объектов в человеко-часы. При умножении количества на коэффициент должны получиться человеко-часы трудозатрат. Таким образом мы можем вычислить приблизительное количество человеко-часов, которое потребуется для реализации каждой фичи.

Дальше добавляются строки корректировок, для учета ресурсов на проектирование, тестирование, доработку системных элементов, разворачивание серверов, взаимодействия заказчиком и прочее. Их можно вычислить по заданному коэффициенту от рассчитанной трудоемкости объема работ.

В результате мы получим предварительную оценку трудовых ресурсов для составления ресурсного плана. Добавив материальные затраты, которые должны быть понесены командой для реализации проектного решения, а также прочие затраты — на командировки, постоянные расходы для поддержания команды, и прочее — мы можем получить примерную оценку стоимости реализации проекта.

На практике, поскольку мы смогли охватить своим пониманием далеко не все активности, **эту цифру нужно удвоить**. Теперь мы можем сформировать предложение клиенту: с одной стороны, ви́дение презентует контекст решения, то, что мы хотим предложить, с другой, ресурсный план показывает, какие затраты необходимо понести для его реализации.

# **Физические серверы. Виртуальные серверы. Хостинг**

С точки зрения аппаратного обеспечения **сервером** называется выделенный компьютер с постоянным подключением к сети, на котором выполняются сервисные приложения (например, СУДБ) и к которому имеют доступ сразу несколько клиентов. Клиентом в данном случае может выступать как пользователь, так и какое-то другое приложение (или даже другой сервер).

С программной точки зрения сервисные приложения тоже часто называют серверами. Например: веб-сервер, FTP-сервер. Также сервисное ПО могут называть сервисами, а в операционной системе FreeBSD вообще принято называть их демонами.

Получается, что сервер — это двойственное понятие. Когда говорят веб-сервер, то могут иметь в виду как приложение (например, Nginx или Apache), так и весь компьютер, на котором это приложение запущено. Но общему смыслу понятия это не противоречит, ведь сервер — это то, что обслуживает клиентов (а этим занимается и компьютер, и приложение).

Когда нужно уточнить, что речь идёт не о программе, а о компьютере — на помощь приходит термин **физический сервер**. Вот это точно про компьютер.

Функции сервера может выполнять обычный настольный компьютер. Например, во многих небольших офисах файловым сервером является один из компьютеров, на котором открыта общая папка для всей локальной сети. Все сотрудники хранят в ней рабочие документы. В данном случае сервер обслуживает небольшое количество клиентов (например, десятерых сотрудников). Поэтому требования к нему невелики.

Для промышленной эксплуатации применяются специальные компьютеры. В целом, там имеются те же компоненты, что и в обычных компьютерах: центральные процессоры, оперативная память, накопители, блоки питания, но в серверных версиях. Например:

* Центральные процессоры Intel Xeon или AMD Epic, в которых ядер и потоков больше, чем в настольных аналогах
* Система питания дублируется: на один сервер может приходиться сразу два блока питания
* Твердотельные накопители (SSD) рассчитаны на большее количество перезаписей, жесткие диски повышенной надежности
* Накопители и блоки питания поддерживают «горячую» замену: можно заменить деталь, не останавливая сервер
* Оперативная память используется с коррекцией ошибок (ECC). Для памяти предоставлено больше слотов, чем на настольных ПК (где обычно имеется 2 или 4 слота). Объёмы памяти также увеличены: 128ГБ ОЗУ для сервера — нормальное явление
* Система охлаждения более надёжная, мощная и шумная, чем на настольных ПК
* Серверы собираются в корпусах специального формата, которые затем удобно монтировать в стойку

Для разработчиков веб-приложений обычно не требуется отдельный промышленный сервер, пока сайт или приложение не наберёт миллионную популярность.

В процессе разработки программист часто пользуется локальными серверами — это набор сервисного ПО (веб-стек), установленный непосредственно на компьютере разработчика.

А вот при размещении тестовой или финальной версии сайта или приложения уже требуется площадка, доступная всем через интернет. Обычно такую площадку арендуют.

**Хостинг** – это услуга по предоставлению ресурсов для размещения сайта или веб-приложения. Компания, которая предоставляет подобные услуги, называется **хостером**.

Ресурсы могут предоставляться в разном виде и объёме, поэтому существуют разные виды хостингов.

# **Выделенный сервер (DS, Dedicated Server)**

Самый дорогой вариант: предоставление в аренду выделенного физического сервера – отдельного компьютера, установленного в дата-центре и подключенного к сети. Как правило, для доступа к этому компьютеру имеется удаленная консоль (эмулятор дисплея, мыши, клавиатуры), через которую по сети можно перезагрузить компьютер и даже переустановить на нем операционную систему при необходимости.

Все заботы по настройке сервера в этом случае ложатся на того, кто его арендует. Но при этом сервер предоставляется в монопольное использование (все ресурсы сервера всегда ваши).

Аренда выделенного сервера стоит от нескольких тысяч до десятков тысяч рублей в месяц, в зависимости от выбранного сервера и дата-центра. Если арендованный сервер ломается, то сотрудники дата-центра оперативно заменяют неисправные узлы (обычно, в течение часа). Но при этом дата-центр не отвечает за сохранность данных: резервные копии и их восстановление – тоже ваша забота.

Отдельно отметим, что в дата-центрах предлагается услуга колокейшн – это размещение сервера, принадлежащего клиенту.

Вы сами покупаете физический сервер, передаёте его дата-центру, там его размещают, обеспечивают питанием, охлаждением и скоростной надёжной сетью. Ежемесячная оплата за колокейшн меньше, чем за аренду выделенного сервера. Но такой вариант требует первоначальных инвестиций (сервер вполне может стоить несколько сотен тысяч рублей). Если сервер на колокейшене ломается, то сотрудники дата-центра могут провести его ремонт, но только за отдельную плату и при наличии у них совместимых комплектующих. В ином случае вам придётся забирать сервер, чинить самостоятельно, возвращать в дата-центр снова.

# **Виртуальный сервер (VPS – Virtual Private Server, VDS – Virtual Dedicated Server)**

На одном физическом сервере хостинговая компания создаёт несколько виртуальных машин, между которыми делит физические ресурсы сервера (потоки процессора, оперативную память, дисковое пространство) и вам в аренду сдает одну из таких виртуальных машин.

В этом случае виртуальный сервер для вас выглядит как выделенный: вы можете его перезагружать или даже на время выключать, переустанавливать операционную систему. И вы также обязаны выполнять все настройки самостоятельно и заботиться о резервном копировании (хотя некоторые хостеры производят резервное копирование виртуальных машин, но вам за доступ к таким резервным копиям придётся платить отдельно).

При этом виртуальный сервер может быть достаточно мощным (например, для одной виртуальной машины может быть выделена половина всех ресурсов физического сервера).Надо учесть, что часть выделенных ресурсов расходуется вашей виртуальной машиной: например, часть места на диске нужно, чтобы установить отдельную копию ОС для виртуальной машины, поэтому не все ресурсы достанутся вашим веб-приложениям.

Также некоторые технологии виртуализации, например, OpenVZ, допускают оверселлинг ресурсов -- когда суммарное количество по всем виртуальным машинам превышает физические ресурсы сервера. Расчет со стороны хостера делается на то, что одновременно все виртуальные машины (выданные для разных целей разным клиентам) не будут загружены по всем ресурсам на полную.

Подобным оверселлингом занимаются не только хостеры. Например, билетов на авиарейс лоукостеры могу продавать тоже на несколько штук больше, чем мест в салоне самолёта, потому что всегда находится кто-то из пассажиров, кто опоздал или передумал лететь.

В случае с виртуальными серверами  рекомендуется выбирать VDS с более строгими технологиями виртуализации: Xen, KVM, Hyper-V.

# **Облачные сервисы**

Вы наверняка уже использовали облачное хранилище файлов, например, Яндекс.Диск или Google Drive. Файлы, загруженные туда, хранятся на надёжных серверах с резервным копированием, к файлам вы можете получить доступ в любой момент через интернет, можете делиться этими файлами по ссылкам. Облако предоставляет вам удобный способ хранения и даже распространения файлов.

Файлы в облаке раскиданы по тысячам физических серверов. Например, половина файлов из вашего аккаунта может храниться на сервере в московском дата-центре, а вторая половина файлов — в Рязани, резервная копия — во Владимире. Распределенное хранение — это нормальный и правильный подход. Провайдер облака кладёт файлы на те серверы, которые в данный момент в наименьшей степени загружены. Сохраняет резервные копии в другой географический регион, чтобы в случае катастрофы в одном дата-центре, данные можно было достать из другого.

Конечно же никто не перемещает ваши файлы вручную. Огромное количество серверов работают как единый механизм под управление специального ПО.

Задача облака в том, чтобы избавить вас (или вашу компанию) от необходимости решать целый набор технических проблем.

## **Облачный хостинг**

Представим, что для своего проекта вы выбрали виртуальный сервер. Настроили его и использовали какое-то время. Но проект начал расти. И в какой-то момент ресурсов, выделенных изначально для виртуального сервера — перестало хватать. Скорее всего, вам придётся заказать и настроить новый более мощный VDS, перенести на него базу и приложение, затем переключить домен на новый IP-адрес, всё протестировать и только потом отказаться от старого виртуального сервера. Хлопотный процесс, согласитесь. К тому же, какое-то время вам придётся одновременно оплачивать оба виртуальных сервера.

А ещё у вашего проекта может быть такая специфика, что пользователи в больших объемах на него приходят в определенные моменты. Например, если ваш сервис показывает информацию о погоде в регионе, то резкий наплыв посетителей у вас, скорее всего будет в тот период, когда люди собираются на работу (по утрам). Возможно, вам хватило бы и старого сервера на все часы, кроме утренних.

Облачный хостинг позволяет **динамически** управлять ресурсами. Вы можете добавить процессорных потоков или оперативной памяти именно в те моменты, когда это необходимо вашему приложению.

Получается, что за повышенную нагрузку вам придётся платить только в течение пары утренних часов, остальное время ваше приложение сможет работать на минимальных ресурсах.

Однако, облачный хостинг дороже обычных виртуальных серверов. Поэтому в каждом конкретном случае следует считать, какой же вариант будет более выгоден.

## **Масштабирование**

Пока мы остаёмся в пределах одного сервера (не важно: виртуального или физического) расти не так уж сложно. Даже если у нас не будет облачного хостинга с динамическим выделением ресурсов, то мы сможем иногда переезжать на более мощный сервер. Добавление ресурсов — это **вертикальное масштабирование**. Обычно небольшие проекты начинают именно с него.

Но возможен и другой подход.

Допустим, что ваш второй успешный веб-проект хранит какие-то записи пользователей (например, о доходах и расходах). В какой-то момент данных становится так много, что на один сервер они уже никак не помещаются.

Тогда вы решаете: данные всех пользователей, зарегистрированных на сайте до 1 января — будем хранить на существующем сервере, но данные новых пользователей — начнём записывать на второй новый сервер. В этот момент вы **масштабируетесь горизонтально**.

При горизонтальном масштабировании возникает **кластер** из виртуальных серверов (хотя сам подход применим и к физическим серверам). И вместе с этим кластером возникают проблемы управления: например, потребуется где-то проверять, в какой момент зарегистрировался пользователь, чтобы отправлять его за данными на один из серверов.

Эту задачу можно доверить третьему серверу, он будет заниматься авторизацией пользователя, а затем отправлять на нужный хост из первой пары. Третий сервер в таком случае называется **балансировщик**.

Заметили, как усложнилась наша инфраструктура?

Её можно было бы реализовать и на наборе обычных виртуальных серверов, но существуют целые облачные платформы, такие как Amazon Web Services (AMS) или Яндекс Cloud — там многие задачи построения сложных инфраструктур уже решены специалистами, а мы, как клиенты облака, может эти решения арендовать в виде различных услуг.

## **Облачные сервисы**

Пионером облачных сервисов стала компания Amazon. Все последователи (например, Google, Яндекс, VK) строят свои облака так, чтобы на уровне протоколов они были по возможности совместимы с API сервисов Amazon. Это удобно для разработчиков: они могут переносить свои приложения из одного облака в другое, не меняя существенно кода. Но на практике всё-таки могут проявляться различия в реализации. В сегодняшних гео-политических условиях русскоязычному веб-проекту лучше ориентироваться на отечественные облака (в том числе с учётом требований ФЗ-152 о персональных данных).

### Объектное хранилище (Object Storage, Amazon S3)

Это инструмент для хранения данных в облаке корпоративного уровня. Данные могут автоматически реплицироваться на несколько географических зон. Хранилище может расширяться до тысяч терабайт (чем оно больше, тем больше денег вам придётся платить облачному провайдеру, но зато вам не потребуется докупать жесткие диски или целые серверы). Возможен вариант «холодного» хранения, когда данные размещаются на длительный срок (например, как резервные копии) и к ним не требуется регулярный доступ — такое хранилище дешевле.

Объектное хранилище при этом не похоже на привычные вам файлы и папки: туда можно складывать файлы и у каждого файла после добавления появляется уникальный идентификатор. По нему далее можно будет скачать или удалить файл. Никакой иерархии у файлов нет, как и имён. Они все лежат как бы в одной большой папке под случайными именами. Это непривычно, но поскольку у разработчиков обычно существует база данных для приложения, то в ней можно без проблем хранить те самые идентификаторы вместо путей к файлам.

### Облачные функции (Cloud Functions, Amazon Lambda)

Это инструмент, так называемых, бессерверных вычислений, когда вы в облаке размещаете даже не целый сайт, а конкретную небольшую функцию (грубо говоря, один файл написанный на популярном языке программирования). И эта функция выполняет конкретную задачу: например, генерирует отчёт или преобразует данные. Такая функция может обращаться к остальной инфраструктуре облака (например, она может сохранить отчёт в объектном хранилище). А вызываться она может по специальной ссылке (при этом по HTTP в функцию могут передаваться какие-то параметры или она может их сама забирать через API).

Если эта функция нужна вам только раз в неделю (например, для того же недельного отчёта), то нет необходимости постоянно держать сервер (даже виртуальный) ради редких запусков функции. В облаке вы будете платить только за фактически потребленные ресурсы в момент запуска.

У Яндекса (да и в других облаках) даже есть бесплатный лимит (Free Tier) для выполнения облачных функций. Так что, возможно, для вас периодический запуск нескольких облачных функций вообще окажется бесплатным.

### Сервисы очередей (Message Queue, Amazon SQS)

Очереди используются для обмена сообщениями о каких-то событиях между компонентами приложений или микросервисами.

Представьте, что ваше веб-приложение должно разослать миллион push-уведомлений (да, такое можно делать и через браузер). При этом для каждого сообщения перед отправкой надо проверить, подходит ли пользователь под условия акции.

Допустим, у вас даже есть облачная функция, которая умеет делать такую проверку. Тогда в очередь можно помещать огромные объемы небольших заданий (сообщений) последовательно или одномоментно, а за счёт интеграции между облачными сервисами она будет разбираться по мере возможности. Обработанные сообщения из очереди будут пропадать. Сообщения, которые не удалось обработать, наоборот можно будет отследить и проверить.

### И многое другое

Конечно, в облаках Яндекса и других провайдеров доступны виртуальные серверы с динамическим масштабированием, виртуальные сети для связывания таких серверов, балансировщики, сервисы мониторинга, облачные СУБД (их не надо устанавливать на сервер, вы сразу получаете доступ к базе с практически неограниченными ресурсами и платите только за потребление: количество хранимых записей, количество запросов) и многое другое.

Для примера вы можете ознакомиться с полным перечнем облачных сервисов Яндекса: <https://cloud.yandex.ru/services#_all>

Но запомните, что облако — не панацея. В каждом случае нужно оценивать риски, учитывать требования, просчитывать наиболее выгодный вариант.

Прежде чем говорить о безопасности ИТ-структуры, напомним, что в IT-инфраструктуру входят:

* сети и сетевые устройства (включая IoT: различные датчики и умные устройства);
* физические и виртуальные серверы;
* пользовательские устройства (в частности, ПК и смартфоны);
* целые центры обработки данных и дата-центры.

Все эти компоненты должны исправно работать и быть максимально защищенными от атак. Чем больше элементов инфраструктуры взаимодействует между собой, тем выше риски и тем больше мер предосторожности должно быть принято. Ведь если часть инфраструктуры будет взломана или выйдет из строя, под угрозой окажутся и другие связанные компоненты.

*:info:***Информационная безопасность** — это практика защиты данных, информации и IT-инфраструктуры от случайного или умышленного повреждения, искажения, несанкционированного доступа, ненадлежащего использования.

**Цель информационной безопасности** в том, чтоб чтобы защитить данные и инфраструктуру от атак злоумышленников, снизить вероятность технического сбоя, а также минимизировать финансовые последствия для бизнеса в случае атаки или сбоя.

В современном мире безопасность инфраструктуры — обязательное условие успешного существования любой компании. Количество атак увеличивается с каждым годом, наверняка вы слышали, как у крупных компаний происходят множественные утечки. При этом владельцы небольших компаний могут думать, что их это не коснется, но это не так. Ведь более 70% атак нацелены именно на малый и средний бизнес.

Часть атак являются неизбирательными, массовыми. Такие атаки эксплуатируют обнаруженную уязвимость в оборудовании или популярном программном обеспечении. Их жертвой часто становятся как раз небольшие компании, где на обеспечение безопасности выделен небольшой ресурс. Поскольку внедрение технологий без обеспечения информационной безопасности влечет за собой финансовые и репутационные риски для компании, тема эта крайне важна. С другой стороны, важно, чтобы информационная безопасность своими ограничениями не препятствовала развитию бизнеса.

В безопасности ИТ-инфраструктуры не получится однажды достичь требуемого уровня и оставить систему в таком состоянии. Постоянно появляются новые угрозы, обнаруживаются свежие уязвимости. Стратегия защиты должна совершенствоваться, быть гибкой и меняться в соответствии с тенденциями атак и угроз.

Нужно понимать, что лишь системный и комплексный подход к защите может обеспечить информационную безопасность. Для этого необходим непрерывный контроль в реальном времени. Контроль должен производиться 24/7 и охватывать весь жизненный цикл информации — от момента, когда она поступает в организацию, и до ее уничтожения или потери актуальности.

Для успешного внедрения систем информационной безопасности необходимо **придерживаться трех главных принципов:**

1. **Конфиденциальность** — означает введение контроля для обеспечения достаточного уровня безопасности при работе с данными, активами и информацией на разных этапах операций, для предотвращения нежелательного или несанкционированного раскрытия.

1. **Целостность** — тесно связана с элементами управления, направленными на то, чтобы корпоративная информация была внутренне и внешне последовательной и лишенной искажений.

1. **Доступность** — обеспечивает надежный и эффективный доступ к информации уполномоченных лиц. Сетевая среда должна вести себя предсказуемым образом, с целью предоставления доступа к информации и данным, когда это необходимо. Восстановление системы по причине сбоя является важным фактором, когда речь идет о доступности информации, и такое восстановление также должно быть обеспечено оперативным и надежным образом.

Выбор и внедрение подходящих видов контроля безопасности помогает организации снизить риски. Рассмотрим **основные виды контроля:**

* **Административный**. Состоит из утвержденных процедур, стандартов и принципов. Он формирует рамки для ведения бизнеса и управления людьми. Законы и нормативные акты, созданные государственными органами, также являются одним из видов административного контроля. Другие примеры административного контроля включают политику корпоративной безопасности, паролей, найма и дисциплинарные меры.

* **Логический**. Логические средства управления (еще называемые техническими средствами контроля) базируются на защите доступа к информационным системам, программном обеспечении, паролях, информации для мониторинга и контроле доступа к системам информации.

* **Физический**. Это контроль среды рабочего места и вычислительных средств (кондиционирование, дымовые и пожарные сигнализации, противопожарные системы, камеры, ограждения, замки, двери).

Информационная безопасность в Российской Федерации регулируется следующими типами нормативно-правовых документов:

1. Акты федерального законодательства. К ним относятся:

федеральные законы (кодексы);

указы Президента РФ;

постановления Правительства РФ;

нормативные правовые акты федеральных министерств и ведомств;

нормативные правовые акты субъектов РФ, органов местного самоуправления и т.д.

1. Нормативно-методические документы государственных органов.

1. Стандарты информационной безопасности. Бывают двух видов: международные стандарты и государственные (национальные) стандарты РФ.

Рассмотрим основные законы об информационной безопасности в Российской Федерации:

**Федеральный закон от 27 июля 2006 г. N 149-ФЗ «Об информации, информатизации и защите информации»** — регулирует отношения, возникающие при осуществлении права на поиск, получение, передачу, производство и распространение информации, при применении информационных технологий, а также при обеспечении защиты информации, за исключением отношений, возникающих при охране результатов интеллектуальной деятельности и приравненных к ним средств индивидуализации.

**Федеральный закон от 27 июля 2006 г. N 152-ФЗ «О персональных данных»** — создает правовую основу обращения с персональными данными физических лиц в целях реализации конституционных прав человека, в том числе права на неприкосновенность частной жизни, личную и семейную тайну.

Персональными данными являются любые сведения о физическом лице, такие как его фамилия, имя, отчество, год, месяц, дата и место рождения, адрес, семейное, социальное, имущественное положение, образование, профессия, доходы, другая информация.

**Федеральный закон от 29 июля 2004 г. N 98-ФЗ «О коммерческой тайне»** — регулирует отношения, связанные с отнесением информации к коммерческой тайне, передачей такой информации, охраной ее конфиденциальности и предупреждением недобросовестной конкуренции. Действие закона распространяется на информацию, составляющую коммерческую тайну, независимо от вида носителя, на котором она зафиксирована.

Под коммерческой тайной понимается конфиденциальность информации, позволяющая ее обладателю увеличить доходы, избежать неоправданных расходов, сохранить положение на рынке товаров, работ, услуг или получить другую коммерческую выгоду.

**Федеральный закон от 6 апреля 2011 г. N 63-ФЗ «Об электронной подписи»** — расширяет сферу использования и допустимые виды электронной подписи.

Электронной подписью называется официальный утверждённый законом аналог собственноручной подписи, который применяется для подписания электронных и заверения бумажных документов, преобразованных в электронный формат. Иными словами, это уникальная цифровая информация в виде комбинации символов.

**Федеральный закон от 26 июля 2017 г. №187-ФЗ «О безопасности критической информационной инфраструктуры Российской Федерации» (вступил в силу 01 января 2018 г.)** регулирует отношения в области обеспечения безопасности критической информационной инфраструктуры (КИИ) России в целях ее устойчивого функционирования при проведении в отношении нее компьютерных атак.



Перейдем к определению **стандартов информационной безопасности**.

*:info:***Стандарты информационной безопасности** — это обязательные или рекомендуемые к выполнению документы, в которых определены подходы к оценке уровня информационной безопасности и установлены требования к безопасности информационных систем.

Исторически первым таким стандартом, который позднее получил широкое распространение и оказал огромное влияние на базу стандартизации информационной безопасности во многих странах, стал стандарт Министерства обороны США «Критерии оценки доверенных компьютерных систем» (Trusted Computer System Evaluation Criteria).

Позже появились и другие международные стандарты информационной безопасности, такие как стандарты ISO, International Organization for Standardization, Международной организации по стандартизации. На текущий момент ей разработано более двадцати тысяч стандартов в самых разных областях. Процессы обеспечения информационной безопасности организации регулируются семнадцатью стандартами, объединенными в группу ISO 27000.

Есть и государственные стандарты Российской Федерации, содержащие требования к процессам и способам обеспечения информационной безопасности. Чаще всего они носят рекомендательный характер (например, ГОСТ Р ИСО/МЭК 18044 или ГОСТ Р ИСО/МЭК 27001-2006), однако некоторые являются обязательными (например, ГОСТ Р 57580.1-2017 и ГОСТ Р 57580.2-2018 для финансовых организаций).

# **Модели разграничения доступа**

Для начала рассмотрим важные для этой темы понятия объекта, субъекта и непосредственно самой модели управления доступом:

* **Объект** – это различные информационные сущности, например: файлы, приложения, базы данных, устройства.
* **Субъект** – это пользователь или приложение, которое обращается к объекту.
* **Метод** (или операция) – это вариант обращения субъекта к объекту. Например: создание, чтение, изменение, удаление файла; запуск приложения.
* **Модель безопасности** (модель управления доступом, модель политики безопасности) – это набор правил и принципов разграничения доступа субъектов к объектам, регламентирующий какие методы и кем при этом смогут применяться.

Рассмотрим основные модели управления доступом.

## **Модель дискреционного доступа, избирательное управление доступом (англ. discretionary access control, DAC)**

В данной модели для каждого объекта указывается перечень разрешенных методов, которые может применять субъект.

В частности, для каждого объекта существует субъект-владелец, который сам определяет тех, кто имеет доступ к объекту, а также разрешенные операции доступа. Основными операциями доступа являются:

* **READ** (чтение)
* **WRITE** (запись)
* **EXECUTE** (выполнение, имеет смысл для программ).

Таким образом, в модели дискреционного доступа для каждой пары субъект-объект устанавливается набор разрешенных операций доступа.

При запросе доступа к объекту, система ищет субъекта в списке прав доступа объекта и разрешает доступ, при условии если субъект присутствует в списке и разрешенный тип доступа соответствует требуемому типу. Иначе доступ не предоставляется.

Классическая система дискреционного контроля доступа является «закрытой», т.е. изначально объект не доступен никому, а далее в списке прав доступа прописывается набор разрешений.

Также существуют «открытые» системы, в которых по умолчанию у всех есть полный доступ к объектам, а в списке доступа прописывается набор ограничений.

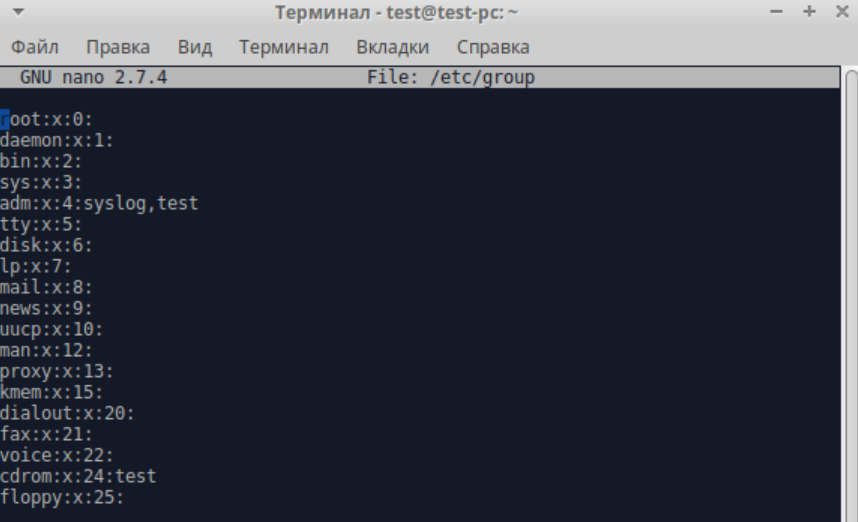
Пример дискреционной модели — система прав в ОС Linux.

В Linux для каждого файла устанавливаются разрешения доступа для трех видов субъектов:

* владелец файла
* группа файла
* остальные пользователи

Таким образом, у файла есть один владелец (пользователь) и одна группа. Но в группу можно добавить сразу нескольких пользователей. Смысл такого объединения в группы проистекает как раз из необходимости предоставлять права доступа к файлам и папкам разным пользователям системы. В этом случае пользователей объединяют в группу и дают нужные права группе.

Все существующие в ОС Linux группы хранятся в текстовом файле /etc/group:



Для редактирования этого файла нужны права администратора.

Для каждого вида субъекта (владелец, группа, остальные) могут устанавливаться права на чтение (r – read), запись (w – write) и выполнение (x - execute).

Набор прав доступа к объекту может быть представлен в виде символьной строки (например, «rwxrw-r--»). Каждая «троица» символов в этой строке отвечает, соответственно, за владельца, группу и остальных. То есть в представленном примере:

* Владелец сможет производить с файлом все три действия rwx
* Группа сможет читать и записывать файл rw- (но запускать как программу – не сможет)
* Остальные смогут только читать файл r-- (записывать и изменять – не смогут)

Есть и другое популярное представление в виде трёхзначного числа (например: 764)

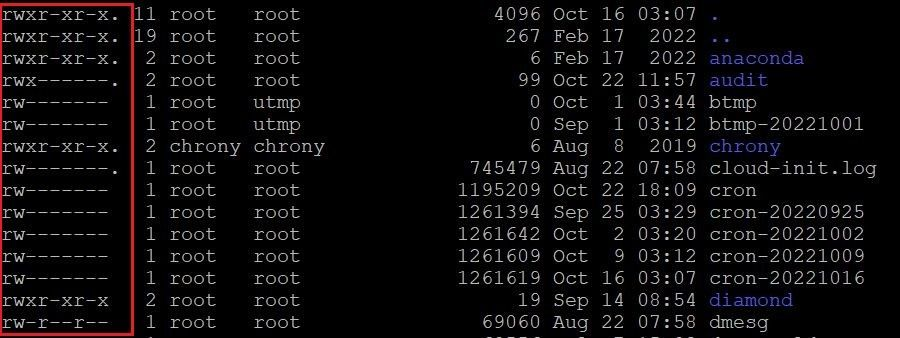
Каждый разряд этого числа получается как сумма трёх действий:

* выполнение — 1
* запись — 2
* чтение — 4

Таким образом, для примера 764 получаем, что:

* Владелец сможет производить с файлом все три действия 1+2+4=7
* Группа сможет читать и записывать файл 2+4=6 (но запускать как программу – не сможет)
* Остальные смогут только читать файл 4 (записывать и изменять – не смогут)

Приведем пример, как в Linux отображаются права в папке с логами базы данных :



Недостаток модели DAC заключается в том, что субъект, имеющий право на чтение информации, может передать ее другим субъектам, которые этого права не имеют, без уведомления владельца объекта. Таким образом, нет гарантии, что информация не станет доступна субъектам, не имеющим к ней доступа. Также, не во всех системах каждому объекту можно назначить владельца (во многих случаях данные принадлежат не отдельным субъектам, а всей системе).

## **Модель мандатного доступа, принудительное управление доступом (англ. Mandatory access control, MAC)**

Ещё одна популярная модель безопасности – модель Белла-ЛаПадулы (классическая модель мандатного управления доступом).

В данной модели есть два основных правила безопасности: одно относится к чтению, а другое – к записи данных.

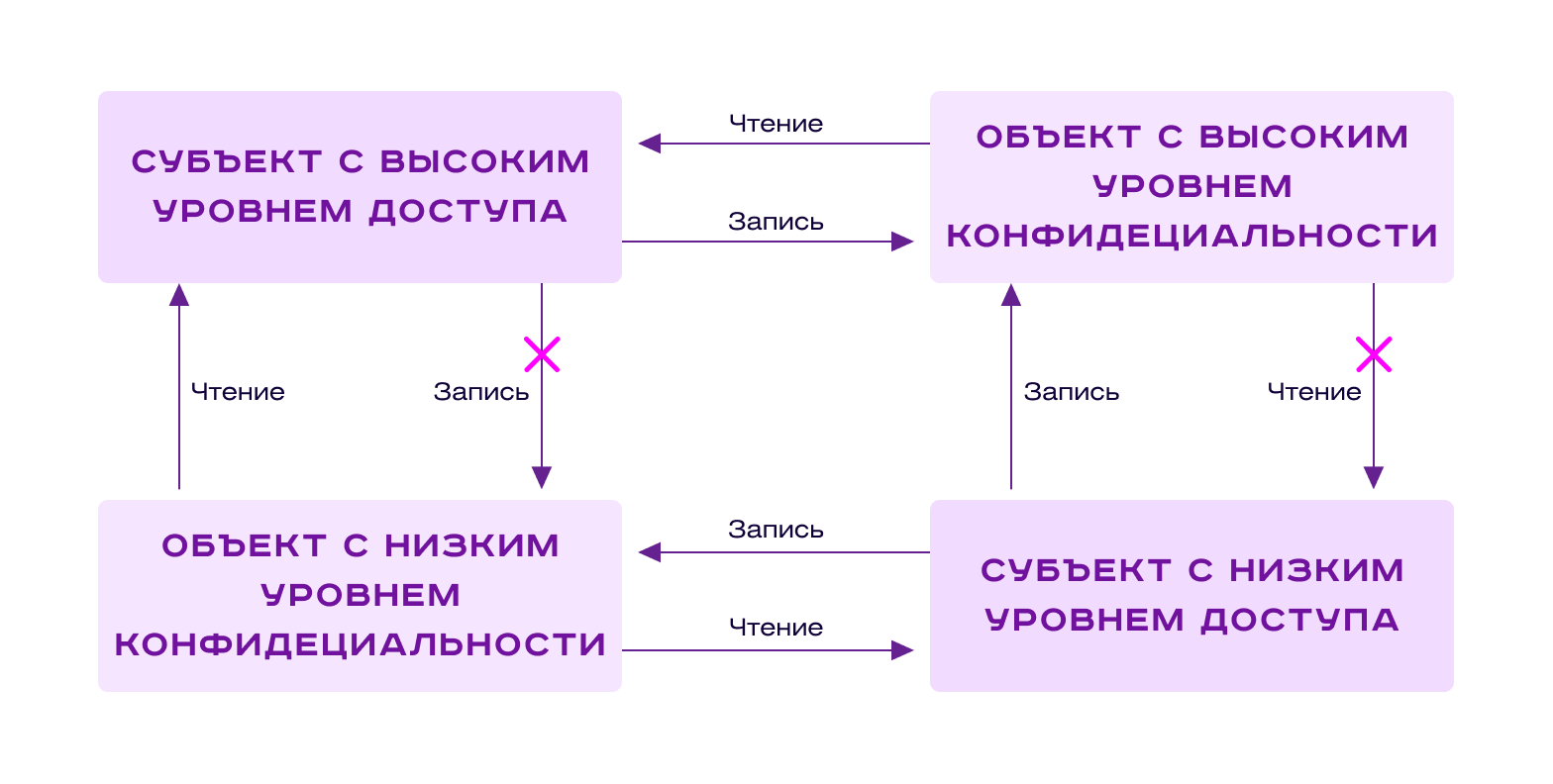
Пусть в системе имеются данные двух видов: с низким уровнем конфиденциальности (несекретные) и высоким уровнем конфиденциальности (секретные), а пользователи этой системы также относятся к двум категориям – с уровнем допуска к несекретным данным (низкий уровень доступа) и с уровнем допуска к секретным данным (высокий уровень доступа).

Для простоты может считать, что все конфиденциальные данные хранятся в секретной папке, а все не конфиденциальные данные в несекретной папке.

Правила таковы:

* Пользователь с высоким уровнем доступа может читать любые данные (из любой папки), и записывать данные только в папку с секретными данными
* Пользователь с низким уровнем доступа может читать только несекретные данные, и записывать любые (в любую папку, в том числе в секретную)

Наглядно модель можно представить на схеме:



Получается, что пользователь с высоким уровнем доступа не сможет распространить (умышленно или случайно) секретные данные — не запишет их в несекретную папку.

В то же время пользователь с низким уровнем доступа, хоть и не может прочитать секретные данные, но может передать их пользователю с высоким уровнем доступа (записать в секретную папку).

В последствии модель получила развитие и в современных системах имеет более сложную организацию.

Модернизированная модель широко используется различными Ведомствами в РФ, документация и данные которых используются в соответствии с системой классификации секретной информации, действующей в РФ согласно Закону «О государственной тайне». Там представлены следующие уровни секретности:

1. Особой важности (ОВ)
2. Совершенно секретные (СС)
3. Секретные (С)
4. ДСП (Для служебного пользования) – ограниченный доступ (потенциально конфиденциальная), но несекретная информация
5. Несекретные (Н)

Общее правило данной модели звучит так: пользователи могут читать только документы, уровень секретности которых не превышает их допуска, и не могут создавать документы ниже уровня своего допуска. То есть теоретически пользователи могут создавать документы, прочесть которые они потом уже не смогут.

Модель мандатного доступа реализована в виде подсистемы ядра Linux под названием SELinux, применяется, например, в дистрибутивах CentOS. В дистрибутиве Ubuntu есть подсистема мандатного контроля доступа под названием AppArmor. Подсистемы мандатного доступа (MAC), как правило, можно отключить и использовать только классический дискреционный доступ (DAC).

В сертифицированных Минобороны России и ФСТЭК операционных системах (например, МСВС, Astra Linux) механизм мандатного разграничения доступа реализован также в ядре ОС, как и механизм дискреционного разграничения доступа,

## **Ролевая модель контроля доступа, управление доступом на основе ролей (англ. Role Based Access Control, RBAC)**

В ролевой модели доступ пользователей к объектам регулируется на основе предопределенных сценариев активности в системе (ролей).

Ролью называется совокупность действий и обязанностей, связанных с определенным видом деятельности. Примеры ролей: администратор базы данных, менеджер, начальник отдела, аналитик.

В ролевой модели с каждым объектом сопоставлен набор разрешенных операций для каждой роли, а не для каждого пользователя.

В зависимости от реализации системы каждому пользователю может быть присвоена одна или сразу несколько ролей.

В некоторых системах есть обязательное ограничение на то, чтобы роли не противоречили друг другу: если одна роль разрешает некую операцию, а другая — запрещает эту же операцию, то одновременно эти две роли одному пользователю присвоить нельзя. Но есть и системы, где разрешено как угодно комбинировать роли: тогда пользователь может выполнять действие, если оно дозволено хотя бы одной из его ролей.

Основное достоинство ролевой модели — удобство администрирования. В отличие от модели DAC не нужно прописывать разрешения для каждой пары «объект-пользователь». Вместо этого прописываются разрешения для пар «объект-роль». И затем роль может быть назначена сразу многим пользователям (или отозвана у них).

RBAC широко используется для управления пользовательскими привилегиями в пределах единой системы или приложения. Например, в Microsoft Active Directory или СУБД PostgreSQL.

Также ролевая модель RBAC применяется в популярных CMS для веб-приложений: например, Drupal и Wordpress.

Как раз на примере Wordpress мы с вами рассмотрим в скринкасте практическую реализацию ролевой модели.

# **Путь от пользовательской истории через требования к новой функциональности**

В гибких методологиях процесс добавления новой функциональности начинается с **пользовательской истории** (user story). Вы уже знаете, что это такое, поэтому ограничимся простым примером:

Я, как постоянный покупатель интернет-магазина детских игрушек, хочу добавлять товары в Избранное, чтобы оттуда добавлять в корзину товары, которые покупаю регулярно, или покупку которых откладывал на потом.

Проводя анализ и декомпозицию пользовательской истории, мы получаем набор требований. Например таких:

* Сайт должен обеспечивать возможность добавления товара в Избранное.
* Сайт должен обеспечивать просмотр товаров в Избранном.
* Сайт должен обеспечивать возможность товар из Избранного добавить в корзину.

Всё перечисленное — это **функциональные требования**. Они определяют, **что система сможет** делать, какие возможности предоставит.

Кроме требований, явно следующих из истории, могут возникать и дополнительные (явно в истории не представленные, но уместные, например, с позиции общепринятых решений). Пример такого требования:

* Сайт должен обеспечивать возможность удаления товара из Избранного.

В пользовательской истории упоминается явно про добавление в Избранное, просмотр списка, добавление из списка в корзину, но ничего не говорится про удаление. Однако, мы понимаем, что даже не озвучив такого намерения заранее, пользователь в какой-то момент пожелает навести порядок в своём Избранном, убрав оттуда товары, которые больше не интересны.

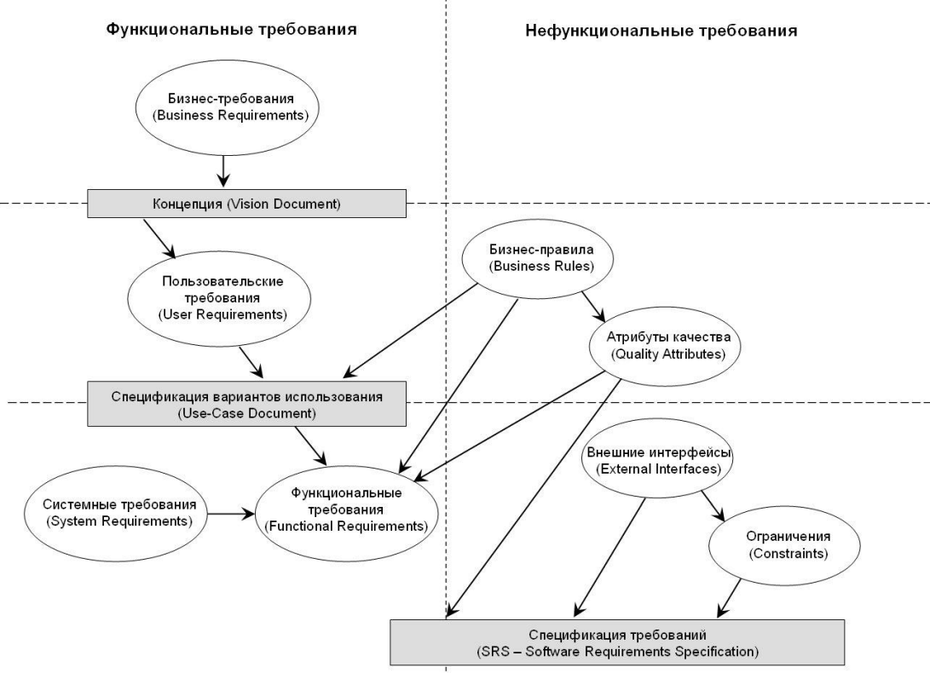
Кроме требований, уже извлеченных из пользовательской истории, могут возникать требования связанные с ограничениями нашей существующей системы, предложений нашей команды, развитием пользовательской истории силами команды.

Например, могут добавиться такие требования:

* В Избранное можно добавить не более 100 товаров (ограничение).
* Если товар снят с продажи, то он удаляется из избранного (бизнес-правило).
* Избранные товары в каталоге отмечаются звёздочкой (дизайн интерфейса).

Эти требования — **нефункциональные**. Они определяют, **как должна быть реализована** та или иная возможность.

Современный классик системного анализа Карл Вигерс разделял требования на следующие уровни:



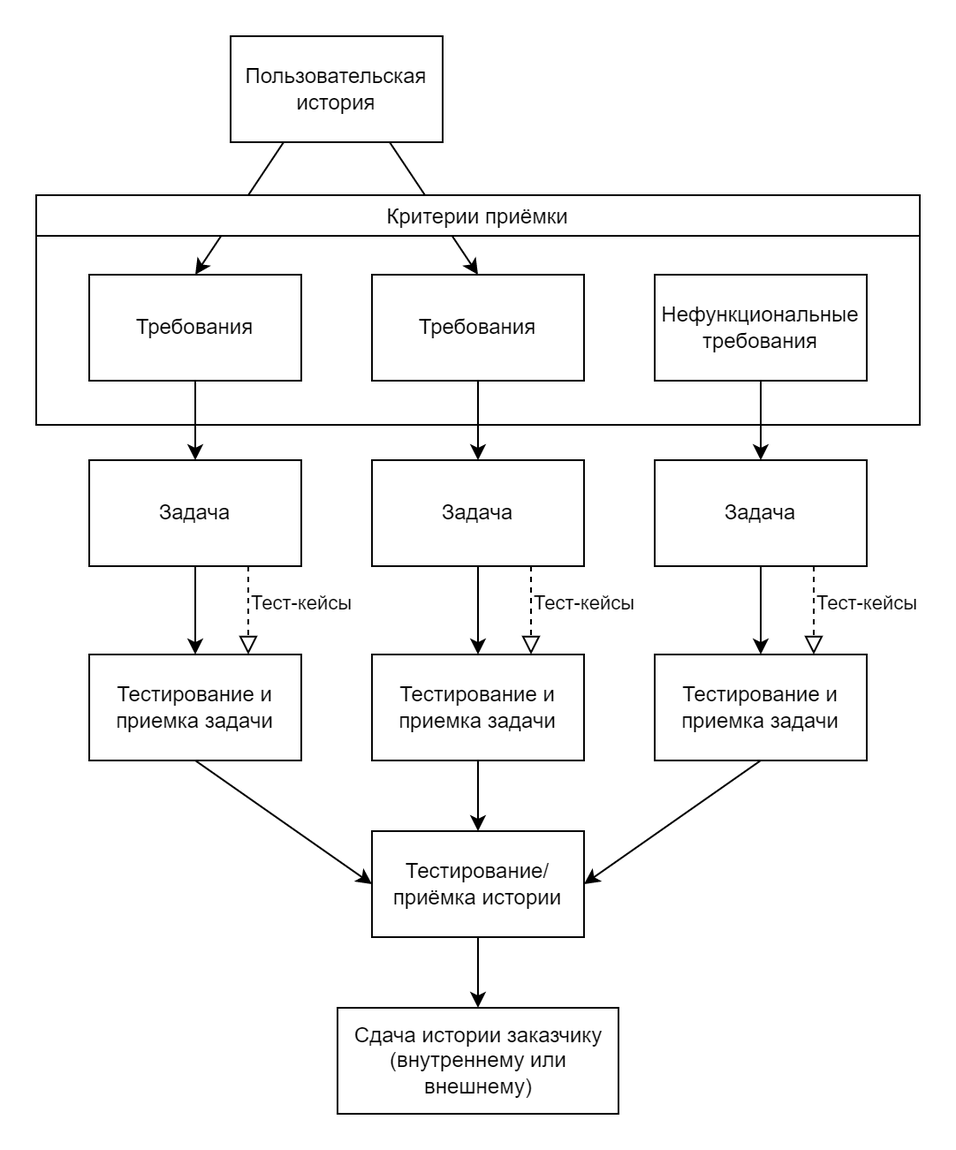
Из всех требований (функциональных и нефункциональных) создаются **критерии приёмки** (англ. Acceptance Criteria).

Каждое отдельное требование или группа требований превращаются далее в задачи, которые будут делать разработчики, дизайнеры или другие специалисты.

Вместе с описанием задач мы также описываем **тест-кейсы** (англ. Test Case) — это сценарии, с помощью которых можно будет проверить готовность каждой задачи в отдельности.

Выполнив все задачи, мы получаем готовую пользовательскую историю. Однако необходимо проверить и результат в целом, опираясь на известные критерии приёмки. И только если результат им удовлетворяет, то мы можем считать историю реализованной и сдавать результат заказчику (внутреннему или внешнему).

Процесс реализации пользовательской истории можно представить в виде такой схемы:



Придерживаясь такой схемы, вы сможете силами команды проходить от очередной пользовательской истории к готовой функциональности. И в рамках гибкой методологии повторять этот путь вновь и вновь на каждом спринте. На всякий случай уточним, что в рамках одного спринта можно реализовать несколько пользовательских историй.