МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК
Кафедра программной и системной инженерии

РЕКОМЕНДОВАНО К ЗАЩИТЕ В ГЭК
Заведующий кафедрой
Д.т.н., профессор
_____ А.Г. Ивашко
_____ 2021 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
магистерская диссертация

РАЗРАБОТКА И АПРОБАЦИЯ ПРОЦЕССА УПРАВЛЕНИЯ ТРЕБОВАНИЯМИ В ПРОЕКТЕ “МЕТАТЕНДЕР”

09.04.03 Прикладная информатика
Магистерская программа «Информационные системы анализа данных»

Выполнил работу
студент 2 курса магистратуры,
очной формы обучения — *(Подпись)* — Буреш Сергей Владимирович

Научный руководитель
Заведующий кафедрой,
Д.т.н., профессор — *(Подпись)* — Ивашко Александр Григорьевич

Рецензент
Директор,
ООО «Тюмень-Софт» — *(Подпись)* — Лозицкий Андрей Вячеславович

Тюмень, 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современная конкурентная бизнес среда заставляет компании постоянно улучшать потребительскую ценность своих предложений. В маркетинге известна такая фраза: “Если ты не быстрый - значит ты мертвый”. Те преимущества, которые предприятие имело еще 5 лет лет назад, сегодня уже не имеют такой ценности. Или их необходимо постоянно развивать или создавать новые решение. Скорость трансформации предложений и выпуска новаций увеличивается год от года.

Существует много разных показателей, которые могут дать оценку уровня конкурентоспособности на рынке: доля рынка, количество клиентов, прибыль на одного клиента, стоимость кампании, бренда, время вывода на рынок нового продукта, скорость доставки и т.д [1]. Но, лично мне, нравится универсальная фраза-мерило, подходящая для любого рынка или продукта, прекрасно характеризующая уровень конкурентоспособности компании: “пионеры — это те люди, которые получают стрелы в спину” [2]. Если за вами пытаются угнаться конкуренты и вы являетесь примером для подражания (и естественно в совокупности с основными финансовыми показателями), значит Вы законодатель тренда, значит ваше предложение самое передовое и инновационное, значит у вас самый высокий потенциал к дальнейшей работе на рынке.

Как добиться наиболее ценного предложения, инновационного решения или передовой технологии, которая дает то самое преимущество и задел на дальнейший рост компании в высококонкурентной среде? Если мы говорим о конкурентной среде, ответ прост - изучение,

прогнозирование или формирование рынка. Т.е. реализовать функцию маркетинга [3].

Отлично, мы изучили рынок, спрогнозировали динамику развития спроса и предложений, определились с продуктами и решениями, которые необходимо развивать в крупную клетку, посчитали показатели на три - пять лет вперед (отлично, если десять лет), что делать дальше? А дальше - тактика, последовательное ежедневное выполнение стратегической цели. Чтобы наше решение, которые мы выводим на рынок, было максимально конкурентоспособным, мы ~~должны~~ обязаны! постоянно изучать потребности клиентов, на которых оно ориентировано. Сегодня, когда тренды, настроения и потребности меняются с неимоверной скоростью, это особенно важно.

Прекрасным примером может послужить продукт и ситуация, которую мы будем рассматривать в данной выпускной квалификационной работе.

В 2011 году компания “Электронный Эксперт” начала свою деятельность по оказанию услуг для участия в электронных торгах по 44 и 223 Федеральным законам и закупкам на коммерческих электронных торговых площадках для юридических лиц [4] [5]. Для этого компания предоставляла весь комплекс необходимых для участия услуг:

- выдача электронной подписи для участия в электронных торгах для разных электронных торговых площадок;
- сопровождение для юридических лиц процессов закупочной деятельности:
    - аккредитация поставщиков на электронных торговых площадках на которых планируются размещение интересующих клиентов закупок в электронном виде;

    - осуществление процедуры анализа, подачи заявки и сопровождения предприятия в самой закупочной процедуре до заключения контракта;
- продажа сторонних сервисов по поиску тендеров;
- выдача банковской гарантии для заключения контракта;
- продажа услуг обучения поставщиков основам и изменениям законодательства закупочной деятельности по 44 и 223 Федеральным законам.

Основным конкурентным преимуществом на тот момент было сформулировано два предложения:

1. Выпуск электронной подписи за 1 час! В период 2011-2018 год тендеры проводились на более чем 100 электронных торговых площадках, многие из которых предъявляли свои требования к электронной подписи и клиенты постоянно сталкивались с потребностью быстро получить электронную подпись для очередной электронной торговой площадки. Основные конкуренты осуществляли выпуск сертификата электронной подписи за 3-5 рабочих дня.
2. Комплексное сопровождение. Клиенту не надо было обращаться за каждой услугой для победы в электронных торгах в разные организации. В кампании “Электронный эксперт” они получали все: электронную подпись (притом быстрее всех), аккредитацию на необходимой электронной торговой площадке, сопровождение подачи заявки и участия в закупочной процедуре, банковскую гарантию для заключения контракта.

Как и было сказано выше, конкуренты не стояли на месте и постоянно работали над своими предложениями. Как итог, сегодня

скорость выдачи электронной подписи в течении 20 минут считается нормой. Как и комплексное сопровождение.

В ходе стратегической сессии кампанией “Электронный Эксперт” была спрогнозировано снижение интереса и объемов продаж к имеющимся сегодня продуктам и услугам компании в сторону увеличения интереса к сервису по поиску тендеров. А для этого необходима своя информационная система поиска и анализа тендеров.

Поэтому было принято решение о необходимости наличия своей системы поиска и анализа тендеров. В ходе анализа рынка было принято решение о приобретении информационной системы “МетаТендер” [6]. По расчетам, стоимость приобретения данной системы с действующей клиентской базой оказалось более выгодным решением, чем написание своей системы с нуля.

Стратегическая цель и задачи кампании формулируются следующим образом:

Цель: увеличение доли продаж сервиса по поиску тендеров от всего объема продаж в первый год в три раза (с 6,27% до 19%). На второй, до 36%, на третий, до 50%.

Для достижения данной цели необходимо выполнить задачи:

1. *Разработать архитектуру проекта, на которой будет реализовано решение.*
2. *Выбрать метод управления проектом разработки и сформировать команду.*
3. *Разработать процесс управления требованиями в проекте.*
4. *В течении первого года реализовать основные требования пользователей системы поиска и анализа тендеров.*

Следует уточнить, что необходимость команды разработчиков обусловлена следующими факторами:

- на рынке есть подобные системы и они постоянно развиваются, а значит и своей сервис должен развиваться как минимум в темпе конкурентов;
- на рынке реализации сервиса динамически меняются потребности клиентов.

Теперь к тактике. Как было сказано выше, основной разработки наиболее ценного предложения является постоянное изучение потребности клиентов, в том числе и обратная связь. В разработке программного обеспечения (информационная система “МетаТендер” является программным обеспечением) за реализацию данной задачи отвечают “требования” [7].

Смысл требований заключается в разработке того, что в конечном итоге удовлетворит потребность пользователя. И это еще на “четверочку”. Хотим получить от рынка оценку “пять” - нам надо предвосхитить ожидания клиентов. Но, это уже совсем другая история…

Так как разработка программного обеспечения это процесс, то процесс “управления требованиями” в данном проекте (назовем его “Проект ”МетаТендер”) для компании становится стратегическим.

Компания “Электронный Эксперт” до приобретения информационной системы “МетаТендер” не имела опыта разработки программного обеспечения и не имела в своем штате программистов.

Отсюда мы приходим к выводу, что разработка и апробация процесса управления требованиями в проекте “МетаТендер” является отдельной ключевой задачей кампании - и основной темой данной дипломной работы.

Для успешной подготовки и защиты выпускной квалификационной работы обучающимся использовались средства и методы физической культуры и спорта с целью поддержания должного уровня физической подготовленности, обеспечивающую высокую умственную и физической работоспособность. В режим рабочего дня включались различные формы организации занятий физической культурой (физкультпаузы, физкультминутки, занятия избранным видом спорта) с целью профилактики утомления, появления хронических заболеваний и нормализации деятельности различных систем организма.

В рамках подготовки к защите выпускной квалификационной работы автором созданы и поддерживались безопасные условия жизнедеятельности, учитывающие возможность возникновении чрезвычайных ситуаций.

ГЛАВА 1. Разработка архитектуры проекта

1.1 Выбор контура размещения архитектуры проекта

В процессе приобретения сервиса “МетаТендер” одной из ключевых задач являлось определение архитектуры проекта.

Прежде всего были определены критерии выбора размещения серверной архитектуры [8] [9]:

- скорость передачи данных между элементами системы;
- скорость поставки/обновления конфигурации оборудования;
- скорость изменения структуры;
- простота администрирования;
- поддержка контура размещения системы 7/24;
- соответствие 152 Федеральному закону об обработке персональных данных (так как в системе размещаются контактные данные физических лиц - пользователей системы) [10];
- соотношению цена-качество на приобретение о поддержу;

Принято решение о выборе единого контура размещения компонентов [11]. По соотношению цена-качество был выбран облачный сервис Yandex.Cloud [12]:

1. Данный сервис поддерживает все компоненты, что позволяет реализовать скорость передачи данных.
2. Данный облачный сервис позволяет быстро обеспечить проект необходимыми мощностями, гибко их оптимизировать, обновлять конфигурации и администрировать.
3. Поддержка сервиса соответствует требованиям - 7/24.

4. Ресурсы сервиса находятся на территории Российской федерации, что соответствует требованиям 152 Федерального закона об обработке персональных данных.
5. По расчетам и сравнительной таблице сервис оказался самым выгодным в соотношении цены и качества.

В результате проектирования итоговую конфигурацию архитектуры схематично отражена на рисунке 1.

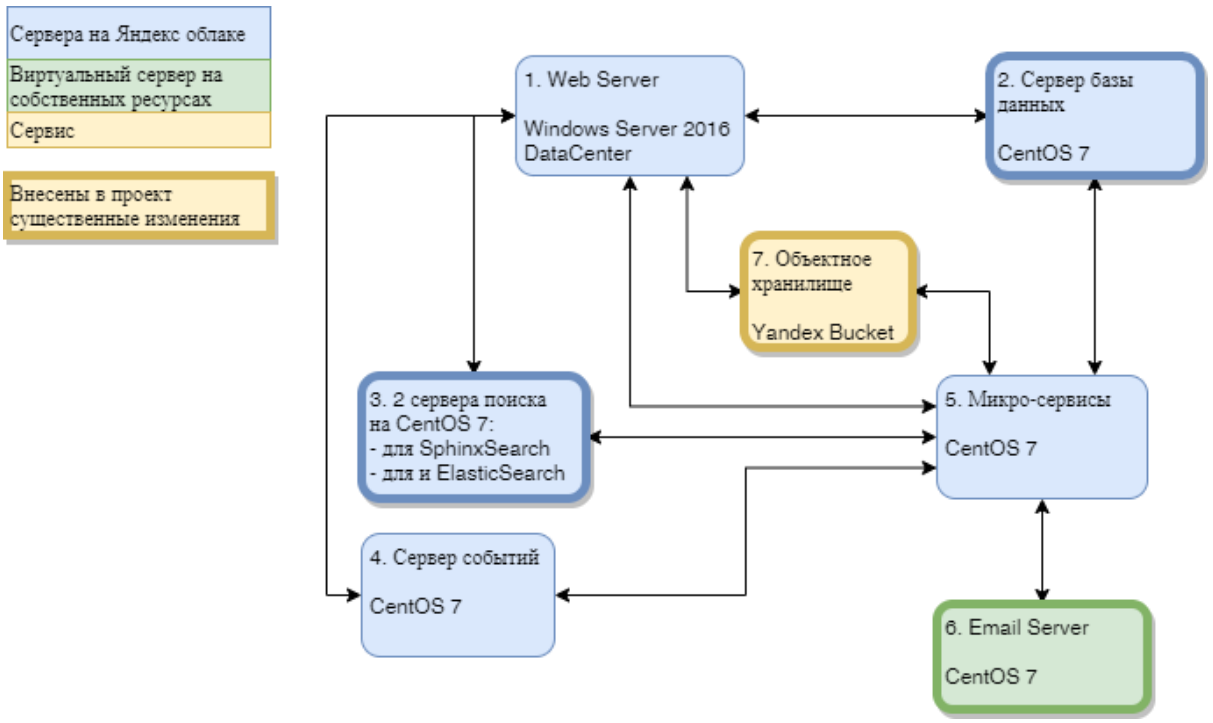


Рис. 1. Итоговая конфигурация архитектуры информационной системы “МетаТендер” .

1.2 Проектирование архитектуры проекта

1. **Web Server**

Прежде всего необходимо спланировать и разместить веб приложение сайта www.metatender.ru [6]. На нем реализуется следующие

решения: Net Framework 4.6.1), MetaTender.API (Net Framework 4.6.1), MetaTender.UserCrm (Net Core).

Так как используется Microsoft Net Framework, который оптимально функционирует на Windows серверах, выбрана следующая конфигурация серверной части: Windows Server 2016 Datacenter, VCPU x 2, RAM 4GB, HDD 60 GB.

Приложения: IIS 7.5+, Net Framework 4.6.1, MySQL Net Connector, Net Core Runtime 2.1, NET Core Windows Server Hosting.

2. **Сервер базы данных**

Шаг два, мы должны ответить на вопрос, где осуществлять хранение данных веб приложения (1) и обработанных (отпарсеных) данных закупочных процедур (все данные). Определяем конфигурацию исходя из объемов и типов данных [13].

Конфигурация: CentOS 7, VCPU x 4, RAM 32GB, HDD 220 GB, SSD 1100 GB. Такое решение выбрано в соответствии с тем, что операционная система бесплатная, имеет открытый исходный код, создана для хостинга больших баз данных, хорошо оптимизирована и справляется с такой задачей.

Приложения: MariaDB 5.5.64. Данное приложение нас не очень устраивает, так как нет поддержки обработки JSON и на практике плохо справляется с существующем объемом данных. Все же, смена базы данных достаточно сложное решение, поэтому было принято решение временно оставить все как есть, накопить необходимой информации для более успешной миграции на новый сервер баз данных.

3. Сервера поиска

С целью оптимизации затрат и ускорения поиска по закупочным процедурам был выделен еще один отдельный сервер. Большие данные могут оставаться и на сервере баз данных (2), но такая информация как контракты и протоколы, по которым и осуществляется поиск и которые предъявляют более серьезные требования к конфигурации, а значит и более дорогой, должна размещаться на отдельном ресурсе. Конфигурация подобрана следующая: CentOS 7, VCPU x 4, RAM 32GB, HDD 100 GB, SSD 100 GB

Приложения: SphinxSearch 2.2.11 [14].

В процессе реализации проекта возникло требование по улучшению релевантности поиска и поиску по документации. SphinxSearch оказалось медлененным, и главное, не справлялся с задачей релевантности поиска. Было принято решение о постепенной миграции на новый сервер с приложением ElasticSearch. Данное решение лучше тем, что в ElasticSearch лучше реализован механизм поиска, в том числе и по документации, больше информации о конфигурировании данного продукта и есть официальные библиотеки, которые значительно упрощают разработку.

Конфигурация выбрана следующая: CentOS 7, VCPU 6, RAM 24 ГБ, Объем дискового пространства HDD 250 ГБ.

Приложения: ElasticSearch 7.11 [15]. В него перенесена прежде всего информация по извещениям и документации. В ближайшее время планируется полностью завершить миграцию на новый сервер и отказаться от предыдущего, на котором остались данные по контрактам и протоколам.

4. **Сервер событий (оперативные данные)**

Данный сервер был принят от предыдущей команды “как есть”, так как функционировал в соответствии с требованиями. Данный сервер

предназначен для работы с кэшем (для скорости) (Redis) и очередью сообщений (хранит очередь и дает возможность их обрабатывать) (RabbitMQ). Связующее звено всем микросервисов (сервера микро-сервисов).

Конфигурация: CentOS 7, VCPU x 2 (20%), RAM 4GB, HDD 50GB.

Приложения: RabbitMQ, Redis.

5. **Сервер микросервисов**

Фактически “Ядро” системы поиска тендеров “МетаТендер” на котором запущены все сервисы и логика MetaTender’a [16].

1. Collector. Сервис по сбору и парсингу закупок с разных источников.
2. Events. Сервис по обработке событий (рассылка сообщений клиентам и управление тикетами)
3. General. Сервис по обработке системных событий и отображения статистики (посещаемость, ошибки, статистика).
4. SearchResource. Сервис по записи и обработке поисковых событий, обновление шаблонов клиентских поиска.
5. Winners. Сервиса поиска победителей на zakupki.gov.ru.
6. TekTorg. Сервис парсинга закупочных процедур ЭТП ТэкТорг.
7. zakupki_mos_ru. Сервис парсинга закупочных процедур Портала Поставщиков.
8. Sphinx 3.0. Новый поиск тендеров.

Конфигурация определена следующая: Centos, VCPU x 4 (50%), RAM 8 GB, HDD 100GB.

Также. содержит приложение: Docker Services Server.

6. **Почтовый сервер**

В сервисе “МетеТендер” есть рассылка, получение и хранение электронной почты пользователям, которая является одним из ключевых функционалов системы. Так как внешние системы достаточно дружественно относятся к функционалу рассылок, во избежании организационных проблем было принято решение данный сервер разместить на своих ресурсах [11].

Конфигурация: CentOS 7, VCPU x 1, RAM 1GB, HDD 60GB.

Приложения: Postfix, Dovecot, Php 5.6, Nginx, Certbot.

7. **Сервис: объектное хранилище**

Для решения задачи по хранению файлов, загруженных клиентами, а также хранения заархивированной документации по закупкам, было принято решение взять в аренду объектное хранилище Yandex.Cloud. Хранилище полностью совместимо с AWS S3 хранилищем. Перечень и характеристики всех серверов отражены в Таблице 1.

Таблица 1. Характеристики серверной архитектуры проекта.

№	Сервер	ОС	vCPU, cores	RAM, Gb	HDD, Gb	SSD, Gb
1	mt-db	Centos7	4	32	220	1100
2	mt-sphinx	Centos7	4	16	100	100
3	mt-iis	Windows 2016 Datacenter	2	4	60	
4	mt-redis	Centos7	2 (20%)	4	50	
5	mt-docker	Centos7	4 (50%)	8	250	
6	mt-mail	Centos7	1	1	60	
7	mt-elastic	Centos7	6	24	250	

- (%) - гарантированная доля процессорного времени
  Object Storage - используется 3 Гб, занято 2 из 25 объектов
- Container Registry - используется 12 реестров из 20.

Спроектированная архитектура соответствует требованиям системы, устойчива и оптимальна. Также, запланировано две миграции в соответствии с меняющимися требования к системе.

Дополнительно в архитектуру проекта на основании требований пользователей были внесены существенные дополнения в виде:

1. Разнесение базы данных на два сервера - для ускорения поиска.
2. Добавление нового сервера поиска осуществляющего более релевантный поиск, в соответствии с требованиями пользователей.
3. Реализовано объектное хранилище на ресурсах нового оператора Yandex.Cloud.
4. Развернут свой почтовый сервер для обеспечения надежности рассылок.

ГЛАВА 2. Выбор метода разработки

2.1. Обзор стандартов и методик управлением разработки

Как уже было сказано во вступительной части дипломной работы, компания - владелец нового информационного сервиса “МетаТендер” никогда не занималась разработкой программного обеспечения и не имела опыта разработки. Первый вопрос, который встал перед командой - “а как это вообще разрабатывать”?, то есть, фактически необходимо выбрать методику разработки.

С чего начать? Первое, с чем предлагается разобраться, это с подбором стандартов и методик разработки, а соответственно и с моделью организации жизненного цикла. Сегодня это следующие стандарты:

- Стандарт ГОСТ Р ИСО/МЭК 12207 (ISO/IEC 12207).
- Стандарт ГОСТ 34.601-90
- Методика Oracle CDM (Custom Development Method)

Сформулируем два вопроса, какой из стандартов и методик выбрать за основу и есть ли смысл жестко придерживаться требований или адаптировать его под свои нужды и использовать как ориентир?

Методика Oracle CDM (Custom Development Method).

Является продолжением Oracle CASE-Method и применяется с помощью программы Designer/2000 [17]. Метод практически жестко привязана к своему конкретному программного продукту от Oracle.

Метод предусматривает следующие этапы: определение требований; анализ; проектирование; реализация; внедрение; эксплуатация и включает в себя следующие процессы:

- RD - Определение производственных требований,

- ES - Исследование существующих систем,
- TA - Определение технической архитектуры,
- DB - Проектирование и построение БД,
- MD - Проектирование и реализация модулей,
- CV - Конвертирование данных,
- DO - Документирование,
- TE - Тестирование,
- TR - Обучение,
- TS - Переход к новой системе,
- PS - Поддержка и сопровождение.

Процессы состоят из задач, которые жестко связаны между собой в программном обеспечении, как на рисунке 2.

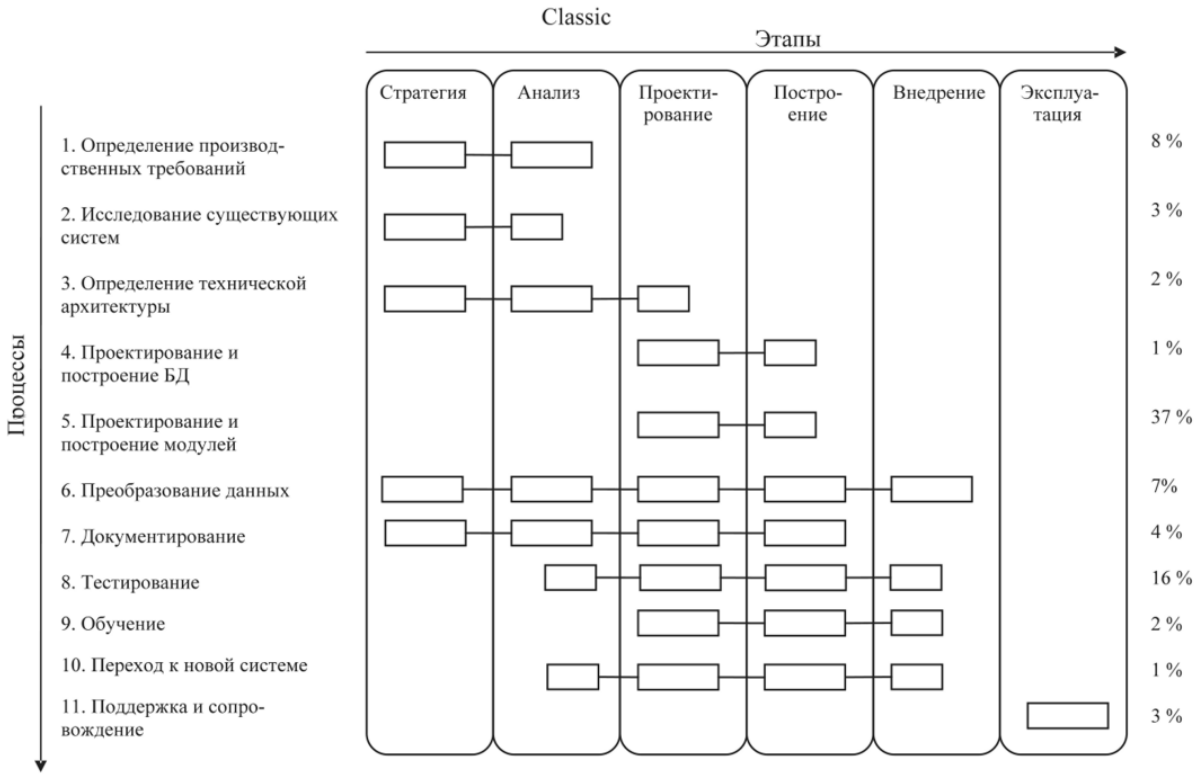


Рис. 2. Процессы жизненного цикла методики Oracle CDM.

Метод предусматривает три варианты жизненного цикла информационной системы: классический, быструю разработку (сильно связана с инструментарием Oracle) и облегченный подход (для малых разработок).

Методика не предусматривает добавления новых задач или удаления старых во всех моделях жизненного цикла. Все три модели жизненного цикла используют “каскадную” (подробнее, далее) модель разработки программного обеспечения.

Международный стандарт ISO/IEC 12207: 1995-08-01.

Универсальный стандарт разработки автоматизированных систем, в том числе и программного обеспечения. Стандарт включает в себя все важные этапы разработки, как стратегию, основные фазы разработки, эксплуатации и завершения жизненного цикла программного продукта [18]. Важными отличиями стандарта ISO/IEC 12207: 1995-08-01 от методики Oracle CDM является:

- стандарт одинаково организует порядок действий как исполнителя, так и заказчика информационной системы;
- стандарт более детально прописывает все процессы жизненного цикла.
- каждый процесс состоит из действие, а действие в свою очередь из набора задач.
- Стандарт имеет высокую степень адаптивности и может применяться для каждого конкретного проекта индивидуально, исключая не используемые процессы. Гибкость стандарта предусматривает взаимосвязь процессов по необходимости их использования, но без потери логики последовательности их выполнения.

Стандарт состоит из пяти основных процессов жизненного цикла, восьми вспомогательных (отвечающих за качество и устранение проблем) и четырех организационных процессов (рисунок 3). Все процессы и их взаимосвязь проиллюстрированы на схеме.

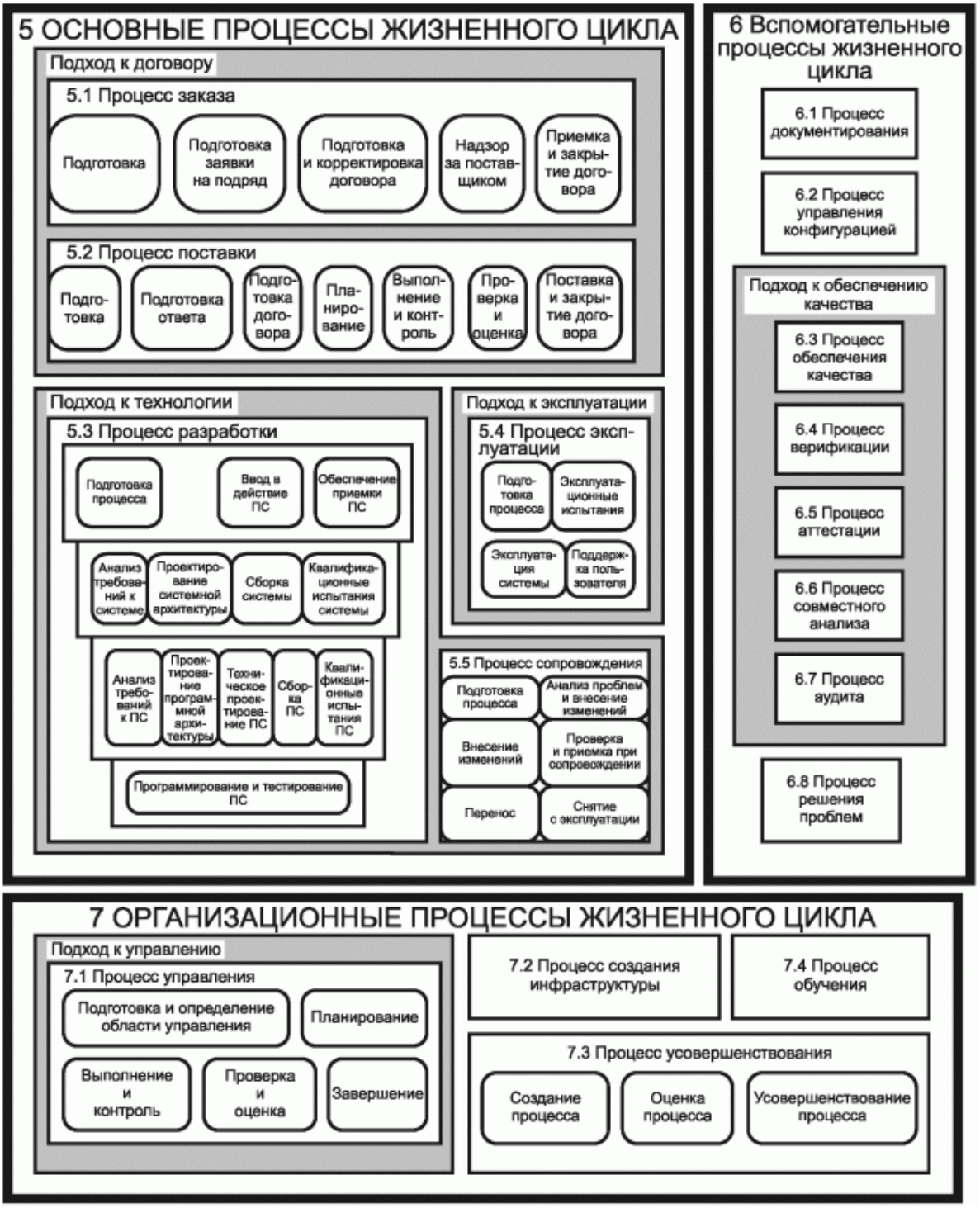


Рис. 3. Процессы жизненного цикла стандарта ISO/IEC 12207.

Стандарт не описывает конкретных действий и не содержит конкретных шаблонов документации. Он описывает структуру и задачи, которые необходимо решить в процессе реализации проекта.

Стандарты комплекса ГОСТ 34.

Также как и Стандарт ISO/IEC 12207, ГОСТ 34 является универсальным стандартом разработки автоматизированных систем, в том числе и программного обеспечения [19].

Еще одной схожей чертой ГОСТ 34 с указанным выше стандартом является описание задач как для поставщика, так и заказчика, но с более детальным содержанием проектных документов, как на рисунке 4.

Рис. 4. Последовательность этапов и процессов указано на рис. 4.

ГОСТ подразумевает индивидуальную адаптивность к конкретным проектам, опускать или добавлять свои разделы, но по большей части ориентируется на классические каскадные методы разработки проектов.

Основным фактором определяющим качество проекта является техническое задание.

2.2. Обзор классических моделей управления разработкой

Как можно понять из стандартов, приведенных выше, основновными стадиями разработки в общем случае можно выделить [20]:

- **планирование** - анализ, что необходимо сделать, в каком порядке, что для этого нужно, какие есть варианты реализации задачи;
- **анализ** - выявление потребности пользователей, описание, требований, понимание результата, формирование критериев результата;
- **дизайн** - определение компонентов системы, требования к безопасности, архитектуру, визуальное представление, формат и тип данных;
- **разработка** - реализация требований и дизайна в виде программного кода, настройка и интеграция изменений с существующей информационной системой;
- **тестирование** - тестирование результата на соответствие заданным критерием при разработке требований, предварительное тестирование разработчиками, тестирование заказчиками;
- **внедрение и поддержка** - обновление системы, информирование и обучение пользователей, поддержка.

Последоставтельнось, довалением или пропуск этапов и зависит от конкретного выбора модели разработки. Существует множество моделей

разработки, еще их называют фреймворки, многие из которых похожи и дополняют другу друга. Проведем обзор основных из них.

Прежде всего отмечу, что по типу модели различаются на классические и гибкие. В классической модели разработка происходит последовательно, промежуточные результаты не демонстрируются заказчику, а следующий этап не начинается, пока не завершится предыдущий. Гибкие модели характеризуется итеративностью, когда сама разработка реализуется небольшими интервалами, после которых можно показывать промежуточный результат заказчику, получить обратную связь и быстро изменить приоритет или глубину разработки.

Классическая модель - Waterfall (каскадная модель, или «водопад»).

Жесткая система - переход на следующий этап разработки в строгой последовательности и только после выполнения предыдущего (рис. 5).

Рис. 5. Этапы жизненного цикла каскадной модели.

Преимущества «водопада» [21]:

- разработку просто контролировать. Всегда известно, какой программист и чем занят, можно легко управлять сроками и стоимостью;
- бюджет проекта определяется на этапе планирования. Все шаги четко запланированы и разработка осуществляется строго по прописанному плану;
- тестирование осуществляется по подробно прописанному изначальному плану.

Недостатки каскадной модели:

- тестирование реализуется только на последней стадии разработки, когда код уже написан. Если в процессе тестирования была выявлена ошибка, то ее исправление сильно скажется на стоимости и сроках;
- так как заказчик увидит результат только после выполнения всех работ, высока вероятность того, что итоговый результат не совпадает с его ожиданиями;
- основная сложность данного метода - очень подробно и качественно расписать требования, чтобы на этапе тестирования или приемки не вскрылись ошибки;
- данный метод подразумевает большое количество документации. Это всегда усложняет процесс внесения изменений в проект и согласования.

V-образная модель (разработка через тестирование).

Это развитие каскадной модели, в которой одновременно составляются требования к системе и описывается, как они будут тестироваться на каждом этапе разработки (рисунок 6). Подходит для

проектов, в которых особенно важна надежность и цена ошибки критически высока.

Рис. 6. Этапы V-образной модели.

Преимущества V-образной модели:

- количество ошибок в разработке сводится к минимуму.

Недостатки V-образной модели:

- как и в “Каскадной” модели, если в процессе проектирования была допущена ошибка, ее исправление окажет существенное влияние на сроки и стоимость;

Incremental Model (инкрементная модель).

Модель разработки по частям (increment в переводе с англ. — приращение). Эта модель подразумевает разработку по частям. Каждая часть представляет собой готовую версию программы. Каждый цикл делится на модули, каждый модуль — на фазы: определение требований, проектирование, написание кода, внедрение, тестирование (рис. 7).

Рис. 7. Схема инкрементной модели.

Данная модель позволяет снизить риски на ранней стадии проекта и значительно быстрее вывести базовую версию продукта на рынок для оценки ее перспективности, без вложений всего бюджета в остальной дополнительный функционал.

Модель предпочтительна для проектов, в которых детально и заранее прописана концепция и техническое задание во всех деталях и важно выпустить продукт на рынок как можно быстрее.

Преимущества инкрементной модели:

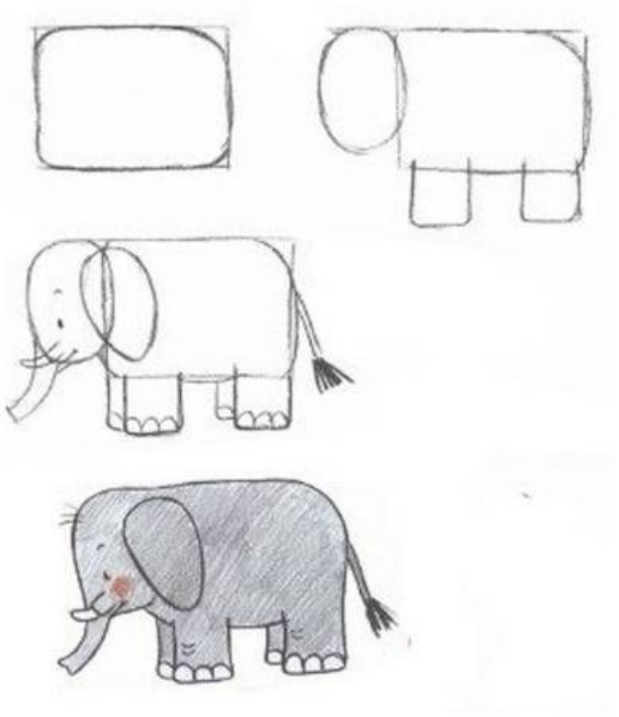
- нет необходимости тратить весь бюджет на старте проекта. Достаточно вложиться в “ядро продукта” и произвести его оценку потребителями (рынком);
- в отличие от “водопадной” модели, можно быстрее получить оценку от заказчика. Исправление ошибок реализуется значительно быстрее и дешевле;

Недостатки инкрементной модели

- есть риск, что каждая команда разработчиков реализует интерфейс продукта по своему, что может привести не к совместимости проекта.
- Разработчики будут оттягивать доработку основной функциональности и «пилить мелочёвку». Чтобы этого не случилось, менеджер проекта должен контролировать, чем занимается каждая команда.

Iterative Model (итеративная модель).

Это модель, при которой заказчик может и не знать что хочет получить в результате, и может не прорабатывать сразу подробное техзадание.



Продукт создается таким образом, чтобы реализовать базовый работающий функционал. Затем с каждой итерацией этот функционал развивается, добавляя с каждой итерацией все новые функцией, как на рисунке 8.

Модель используется если проект большой и нет четких требований, или когда конечный результат не известен.

Рис. 8. Образ итеративной модели.

Преимущества итеративной модели:

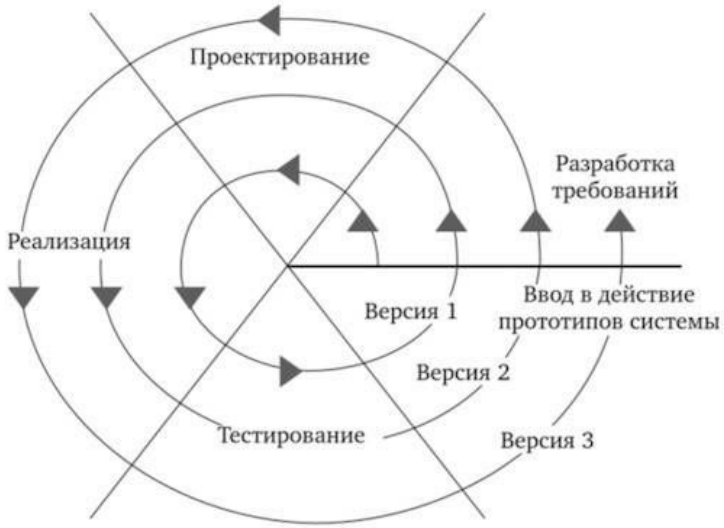
- также как и в инкрементной, нет необходимости тратить весь бюджет на старте проекта. Достаточно вложиться в “ядро продукта” и произвести его оценку потребителями (рынком)
- быстрое обнаружение ошибок.

Недостатки итеративной модели:

- сложно масштабировать.
- не известно как выглядит конечная цель и результат.

Spiral Model (спиральная модель).

В случае, если цена ошибки высока и необходимо серьезно оценивать риски на каждом этапе - используют спиральную модель.



Как и инкрементальная модель, спиральная хорошо реализуется и в стартапах - позволяя в любой момент оценить успешность проекта или провал и завершить проект без весомых потерь.

Рис 9. Схема спиральной модели.

Спиральная модель достаточно затратна и как правило используется в крупных проектах.

Преимущества спиральной модели:

- тщательная проработка рисков

Недостатки спиральной модели:

- разработка длится долго;
- разработка стоит дорого.

2.3 Agile - гибкие методы разработки

Не все проекты могут быть структурированы таким образом, чтобы быть реализованными по классическому проектному подходу [22].

Если привести пример с ремонтом автомобилей: ремонт одного автомобиля идеально ложится на классический (водопадный) подход, а вот быстро и без простоя отремонтировать пять автомобилей, будет практически невозможно, если нельзя приступать к ремонту следующего автомобиля, пока не починили предыдущий (включая простои из-за поставки запчастей).

Agile – семейство гибких итеративно-инкрементальных методов к управлению проектами и продуктами. Согласно данной методике, проект разбивается не на последовательные фазы, а на маленькие подпроекты, которые затем «собираются» в готовый продукт, как на рисунке 10.

Рис. 10. Схема гибкого метода управления проектом.

Таким образом планирование проводятся для всего проекта, а этапы разработки, тестирования и прочие отдельно. Это позволяет достигать

конкретных результатов (инкрементов) быстрее, и вносить изменения небольшими, но ценными партиями.

Сам по себе Agile – не метод управления проектами [23]. Это скорее набор идей и принципов того, как нужно реализовывать проекты. Уже на основе этих принципов и лучших практик были разработаны отдельные гибкие методы или, как их иногда называют, фреймворки (frameworks): Scrum, Kanban, Crystal, и другие. Эти методы могут достаточно сильно отличаться друг от друга, но они следуют одним и тем же принципам.

Сильные стороны Agile:

- гибкость и адаптивность;
- быстрая реакция на изменения;
- прекрасно подходит и для стартапов - когда результат не известен.

Слабые стороны Agile:

- Agile – не является ни методологией, ни стандартом. Agile — это набор принципов и ценностей. Каждой команде придётся самостоятельно составлять свою систему управления, руководствуясь принципами Agile.

Различия между Agile и традиционным подходом в таблице 2.

Таблица 2. Различия традиционного и гибкого подхода управления проектом.

Характеристика проекта	**Традиционные модели**	**Гибкие модели**
Подход:	прогнозирующий	адаптивный
Критерии успеха:	следование плану	ценность для заказчика
Риски:	определены	не определены

Контроль:	легкий	зависит от профессионализма участников
Вовлеченность заказчиков:	низкая	высокая
Документация:	Детальная с самого начала проекта	Дорабатывается, по мере развития, если вообще нужна
Требования:	стабильны, известны заранее	легко изменяемы, не всегда известны заранее
Команда:	легкое вовлечение новых на любом этапе	опытные специалисты, стабильный состав
Рефакторинг:	дорого	не дорого

Далее, я расскажу об основных фрейворках Agile. Я не буду в дипломной работе их описывать, информации о них предостаточно в интернете и литературе. Остановлюсь лишь на основных аспектах.

Scrum

Гибкий фреймворк Scrum сочетает элементы классического процесса разработки и идеи гибкого подхода к управлению проектами [24]. В итоге это сбалансированное сочетание гибкости и структурированности, отображено на рисунке 11.

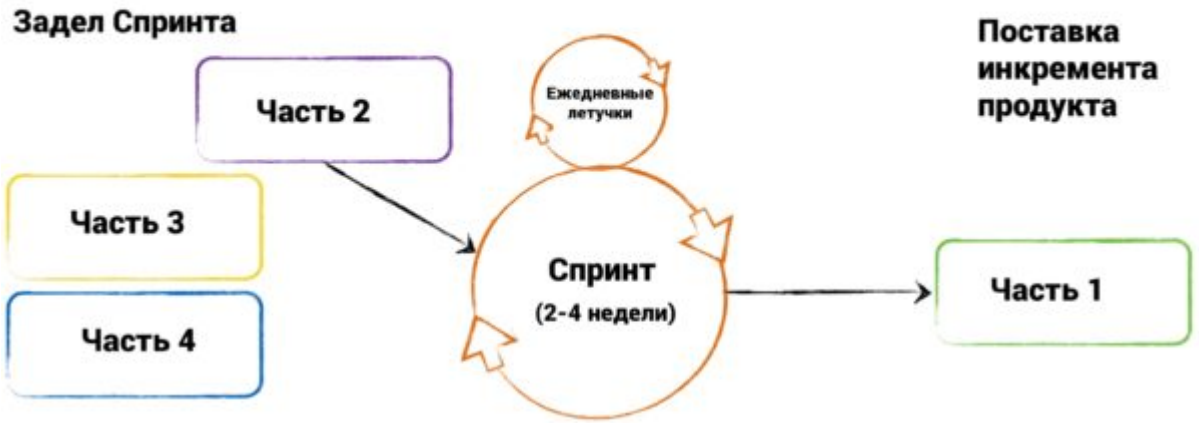


Рис. 11. Схематичное отражение фреймворка Scrum.

Во многом в этом и является основное преимущество Scrum, гибкий, быстрый на результат и притом четкий подход к реализации проектов.

Помимо того, что Scrum прекрасно подходит для проектов, в которых необходим быстрый результат с учетом меняющихся внешних факторов и требований фреймворк подходит для ситуаций, когда не все члены команды имеют достаточный опыт в той сфере, в которой реализуется проект. Дискуссия – постоянное общение между членами проекта позволяют нивелировать недостаток опыта или квалификации за счёт максимальной информации и помощи команды [25].

Так как в Scrum команда проекта является небольшой (5-9 человек) члены проекта должны обладать более чем одной компетенцией-обязанностью. Считается нормой или иной раз и обязанностью, если разработчик должен быть и тестировщиком и даже аналитиком. Принцип взаимозаменяемости позволяет проекту двигаться, если даже часть команды временно отсутствует. Залог успеха, если участники проекта не боятся брать на себя ответственность и помогают другу другу.

Scrum может не подойти для разработки чего то конкретного – например автомобиля или постройки завода.

Lean

Lean добавляет к принципам Agile схему потока операций (workflow) для того, чтобы каждая из итераций выполнялась одинаково качественно (рис. 12).

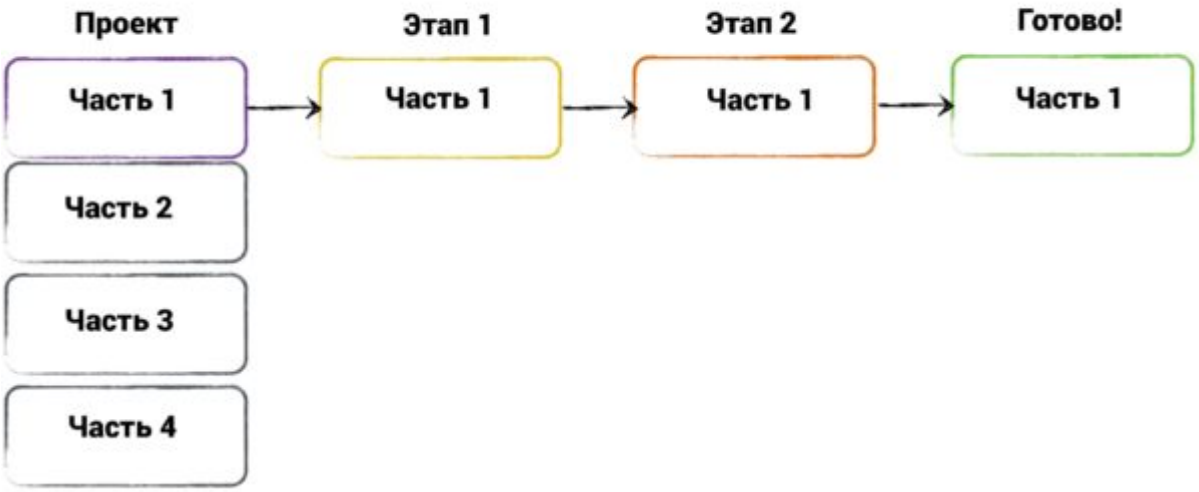


Рис. 12. Схематичное отражение фреймворка Lean.

Каждая часть проекта реализуется так, как требуется. Кроме того, Lean позволяет параллельно выполнять несколько задач на разных этапах, что повышает гибкость и увеличивает скорость исполнения проектов.

Lean позволяет создать систему, удовлетворяющую любым требованиям в управлении проектами.

Сильная сторона - ровное качество и четкое исполнение.

Слабая сторона Lean в том, что каждый этап проекта требует одинаково детальной и скрупулезной проработки и внимания. А также, в отличие от Scrum, в Lean не установлены сроки реализации разработки каждой итерации, что может затянуть процесс получения инкрементов.

Kanban

Kanban, инкремент передается с этапа на этап, а в конце получается готовая ценность.

Kanban позволяет оставить неоконченную задачу на одном из этапов, если её приоритет изменился и есть другие срочные задачи. Нет дедлайнов или четких сроков - процесс реализуется по приоритетам, отражено на рисунке 13.

Рис. 13. Схематичное отражение фреймворка Kanban.

Поэтому Kanban значительно мягче, чем Scrum, как и помимо сроков разработки, нет в нем и ролей. Нет обязанностей по дискуссии и четкости их проведения. Обсуждения и вовсе можно не проводить.

Для Kanban необходимо определить этапы потока операций (workflow). В Kanban они изображаются как столбцы, а задачи обозначают специальные карточки. Карточка перемещается по этапам и на каждом этапе процент завершения становится выше. На выходе мы получаем готовый к поставке заказчику элемент продукта. Доска со столбцами и карточками может быть как настоящей, так и электронной – даже здесь Kanban не накладывает никаких ограничений на пользователей. Команда должна быть дисциплинированной и самомотивированной.

Также как и в Scrum, в Kanban лучше результат достигают команды, в которых, навыки участников пересекаются. То есть обеспечивается взаимозаменяемость и поддержка друг друга. Kanban эффективен, если нет жёстких сроков. Для проектов со сроками лучше подходит Scrum.

6 сигм (Six Sigma)

Six Sigma (рис. 14.) похожа на Kanban, только теперь установлены четкие этапами проекта – планированием, определением целей и тестированием качества. Важным отличием от Kanban то, что появляются по измерению и контролю показателей качества проекта на этапах реализации.

Рис. 14. Схема метода разработки Six Sigma.

6 сигм как правило реализуют в сложных проектах, в которых много новых и сложных операций.

PRINCE2

PRINCE2 - гибрид между классическим и проектным подходом управления разработкой с упором на качество из 6 Sigma, рис. 15.

Рис. 15. Схема основных этапов метода управления проектом PRINCE2.

PRINCE2 концентрируется на управленческих сторонах проекта и рекомендует адаптировать методологию под каждую конкретную организацию. В PRINCE2 более чётко определённая структура команды проекта и ориентирован на масштабные государственные проекты и крупные организации.

PRINCE2 может быть адаптирован для проектов любого масштаба и любой предметной области. Методология предлагает конкретные рекомендации по изменению жизненного цикла проекта, ролевой модели и набора обязательных документов в соответствии с потребностями проекта.

2.4. Выбор стандарта и модели управления разработкой

Так как проекту “Метатендер” необходимо:

- в проекте не определен результат к которому необходимо прийти;
- гибкость изменения требований;

- контроль получения инкрементов в срок;
- легкость управления методологией;
- модель, ориентированная на небольшие команды;
- модель не должна рассыпаться при потере участника команды;
- модель, позволяющая реализовать проект неопытной командой.

Из всего вышеперечисленного, выбор модели разработки проекта “Метатендер” вполне очевиден - за основу выбрана модель гибкой разработки Agile, фрейворк Scrum.

Основной проблемой Scrum является то, что в данном фреймворке четко не определено, как от пользовательской истории перейти к тестированию.

Дискуссия в Scrum - является основным процессом определения всех деталей пользовательской истории [7]. Данные подход позволяет сместить акцент с того, как необходимо выполнить требование на что необходимо сделать - чего достичь. Конечный результат и ценность для конкретного заинтересованного лица в таком подходе в приоритете. Для того, чтобы понять, все ли реализовано верно, создаются проверочные тесты. Дискуссии в этом процессе позволяют выявить все необходимые мелкие детали, дополнительную информацию, найти ошибки и даже посмотреть на требование с критической стороны.

Результатом такого обсуждения для разработки является четкое понимание - как это можно реализовать функционально, фактически - как это сделать, рисунок 16.

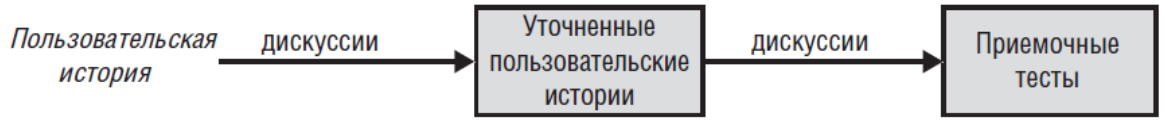


Рис. 16. Как пользовательские требования ведут к функциональным требованиям и тестам при использовании подхода на основе пользовательских историй [7].

По своей сути, это следующие три этапа:

1. Описание пользовательской истории.
2. Обсуждение и детализация пользовательской истории.
3. Определение тестов, как критерием результата.

При необходимости, повторяем пункты 2-3 еще несколько раз.

Приемочные тесты в Scrum:

- используют вместо списка функциональных требований, которые не всегда отражают суть задачи;
- позволяют лучше понять множество деталей, которые стали известны в ходе дискуссий разработчиков и остальных участников проекта;
- позволяют получить четкие критерии оценки и приемки результата;
- пишутся до написания кода разработчики вами и являются неотъемлемой частью процесса разработки, поскольку они представляют собой полезный и эффективный метод, с помощью которого заказчик может сообщить разработчикам о своих пожеланиях в отношении новой функциональности;
- дополняются разработчиками в процессе дискуссии с ответственным и остальной командой проекта.

ГЛАВА 3. Управление требованиями в проекте “МетаТендер”

3.1 Взаимосвязь стратегии и требований

И так, Управление компание приняло стратегическое решение - о приобретении сервиса поиска и анализа тендеров “Метатендер” и запуска проекта. Реализована работа по приобретению прав уже готового решения сторонней командой, реализован процесс заказа и поставки.

Основной целью проекта определено (бизнес-требование) - увеличение доли от всего объема продаж сервиса по поиску тендеров в первый год в три раза (с 6,27% до 19%). На второй, до 36%, на третий, до 50%.

Так как бизнес-требование сформулировано в виде финансовых показателей, реализация системы управления бизнес-требованиями построена на основе системы бюджетирования (таблица 3).

Таблица 3. Часть бизнес-плана проекта, зафиксированного в системе бюджетирования.

ПЛАН	.янв..	.февр..	.мар..	.апр..	.мая.	.июн..
ОБЩИЙ ДОХОД	р.60 000	р.60 000	р.40 000	р. 328 950,0(	р. 333 073,3(	р. 496 415,7
Затраты			р.371 400	р.432 417	р.434 530	р.479 632
Маржа		р.60 000	-р.331 400	р. -103 466,5	р. -101 456,4	р. 16 783,50
Маржа с нак.		р.60 000	-р.271 400	-р.374 867	-р.476 323	-р.459 539
Рентабельность			-829%	-31%	-30%	3%

Структура дохода была спланирована в отдельном плане, с разбивкой на доходную часть от действующих клиентов, новых и продажам от партнеров.

Структура затрат тоже запланирована в отдельном плане по следующим статьям:

- фонд оплаты труда программистов;
- фонд оплаты труда системных администраторов;
- фонд оплаты труда сбытового отдела;
- затраты на инфраструктуру (облако);
- затраты на маркетинг;
- административные затраты.

Система управления показателями верхнего уровня построена на основе системы бюджетов, для удобства контроля ключевых показателей (рис. 17).

Рис. 17. Система финансового управления проектом на основе управления бюджетом.

Планирование бюджета проекта осуществлено помесячно на два года (в рамках выполнения основной цели проекта, его окупаемости за два года).

Контроль показателей осуществляется ежемесячно.

Анализ выполнения плана осуществляется бюджетным комитетом ежеквартально. Если анализ выявляет значительное отклонения основных показателей от запланированных на 20% - то в проект или план могу вносится изменения в виде корректирующих управленческих решений, а соответственно и корректировку плана. Такой подход позволяет быстрее реагировать на изменения внешней среды или корректировать показатели из-за внутренних корпоративных событий, не допуская серьезных финансовых потерь и лучше прогнозировать результат, поддерживая в разработке принципы Agile.

Следующим шагом является формулирование бизнес-правил, определяющие или ограничивающие некоторые стороны бизнес-процессов и разработки в целом. По своей сути они являются как финансовыми, дак и не финансовыми и служат источником нескольких типов требований:

Финансовые бизнес-правила:

1. Проект должен приносить прибыль.
2. Проект должен выполнять план по продажам.
3. Затраты на проект не должны превышать доход.

Не финансовые бизнес-правила:

1. Все пользователи хотят, чтобы доступ к системе был бесперебойный по будням с 02 часов утра до 22 часов по Тюменскому времени.
2. Все пользователи хотят, чтобы доступ к тендерам подключенных (заявленных) площадок был беспрерывным.
3. Все пользователи хотят всегда получать уведомления о новых тендерах и их изменениях:
    - о тендерах, срок подачи заявок у которых не более 2 суток - не более 15 минут с момента опубликования;

- о тендерах, срок подачи заявок у которых более 2 суток - не более 3-х часов с момента публикации, в рабочее время своего часового пояса.

Данные бизнес-правила зафиксированы документально и с ними ознакомлены все участники команды.

3.2 Управление требованиями верхнего уровня

Очевидно, что выполнение финансовых показателей проекта “МетаТендер” зависит от качества и выбора рыночной стратегии, выбора приоритетов требований верхнего уровня, выбранного маркетингом сегмента рынка и усилий сбытовых отделов. Стратегия уже определена. Следующим шагом является выбор и формулирование требований верхнего уровня и работа с пользовательскими историями:

Также к требованиям верхнего уровня отнесены направления разработки, которые тесно связаны с маркетинговой концепцией и обобщенными требованиями конечных пользователей, таблица 4.

Таблица 4. План реализации направлений разработки проекта.

С	по	дней	Напарвление разработки	Результат
16.03.2020	24.04.2020	39	**1. Приемка системы, перевод на наши ресурсы**	готово
24.04.2020	30.06.2020	67	**2. Знакомство с системой**	готово
01.07.2020	31.08.2020	61	**3. Формирование команды**	готово
01.09.2020			**4. Подключение новых ЭТП**	постоянно
01.09.2020	31.12.2020	121	**5. Улучшение системы поиска**	готово
01.01.2021	01.09.2021		**6. Улучшение аналитики**	в работе
01.01.2021			6.1. Аналитика для маркетинга, продаж и партнеров	в работе
			- ЕРУЗ	готово
			- реестр поставщиков	в работе
			- настройка целевой аудитории	в ожидании
			- оценка внешних клиентских баз	в ожидании
01.01.2021			6.2. Аналитики карточки тендера для пользователя	в ожидании
?			- аналитика. Подготовка данных к аналитике	в ожидании
			- аналитика. Доработка карточки тендера	в ожидании

Основным предназначением направления разработки является обеспечение максимальной ценности и актуальности между маркетинговой концепцией проекта, изменениями потребностей рынка (пользователей) с инкрементами разработки.

С целью более точного понимания направлений разработки, полезно их формулировать по правилам пользовательских историй и понимать взаимосвязь в формате mind map (интеллект карта), отражена на рисунке 18.

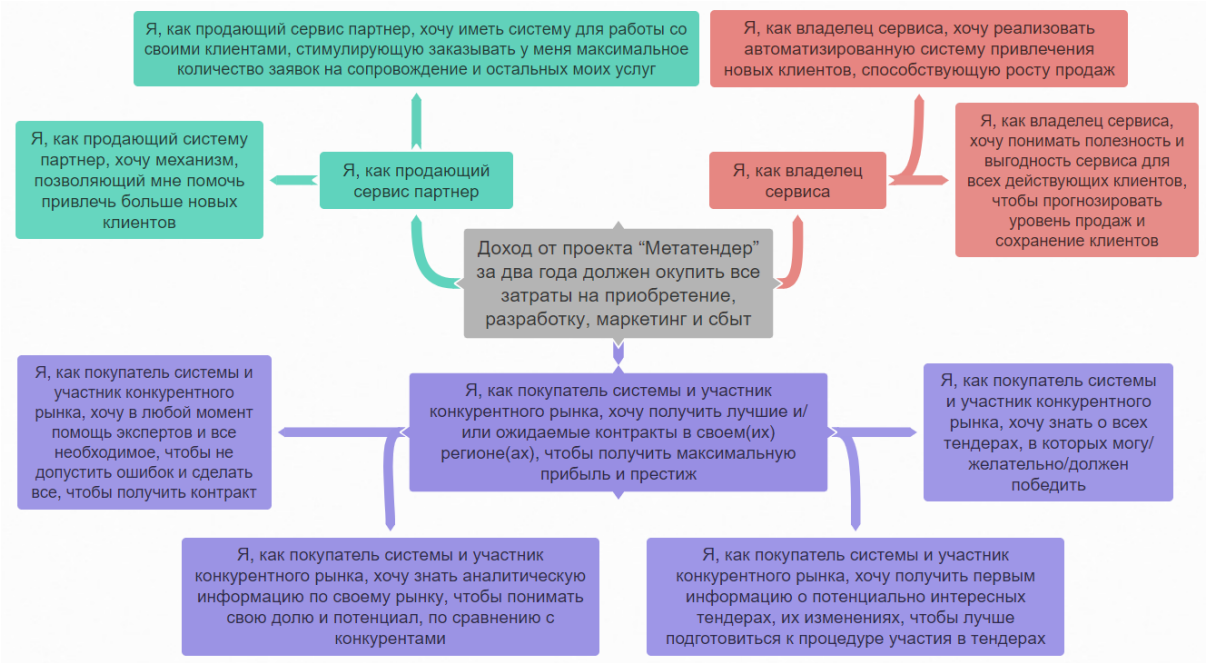


Рис. 18. Ключевые требования к проекту в виде mind map (интеллект карта).

Конечная система управления требования верхнего уровня отражена на рисунке 19:

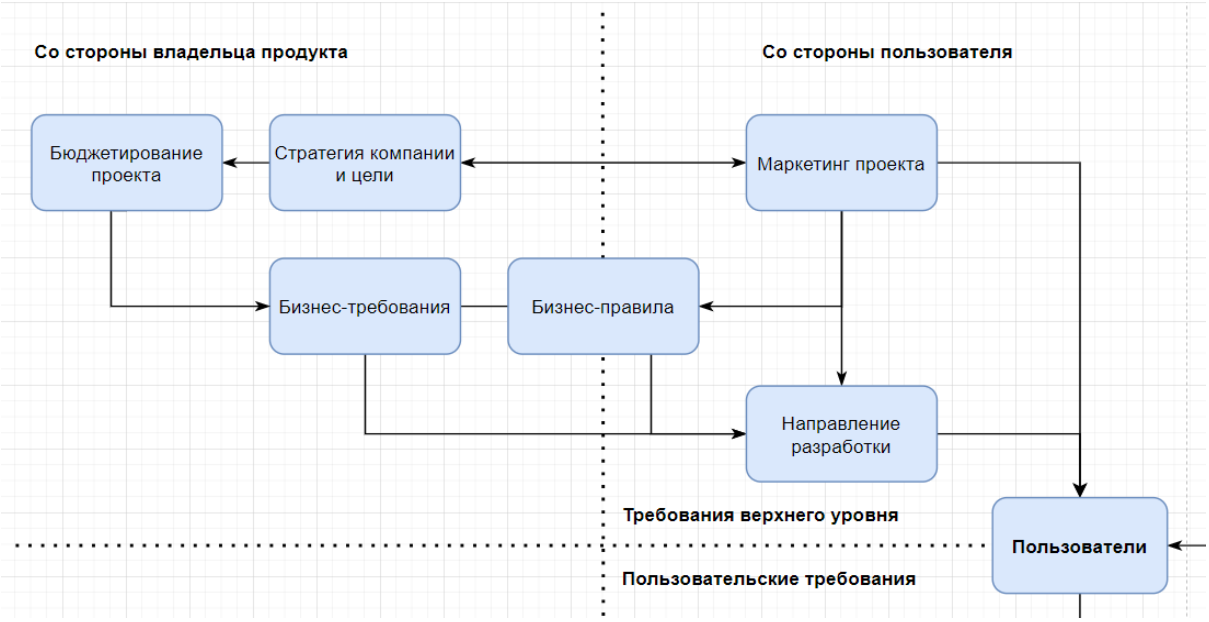


Рис. 19. Система управления требования верхнего уровня проекта “МетаТендер”.

3.3 Управление пользовательскими требованиями

Как было сказано выше, пользовательские требования в проекте “МетаТендер” определяются направлением разработки. Но, направления разработки не являются основным источников разработки требований. Основном источников требований все же являются пользователи продукта.

Пользователи продукта в порядке важности:

- Клиенты - кто платит за использование сервиса (“МетаТендер”) в обмен на ее полезность.
- Потенциальные клиенты - кто еще не платит, но заинтересовался сервисом.
- Партнеры - кто реализует сервис своим клиентам.
- Менеджеры отдела продаж - кто реализует сервис в рамках кампании-владельца сервиса.
- Маркетологи - кто осуществляет маркетинговую функцию сервиса “МетаТендер”.

- Руководство - кто осуществляет управление проектом.

Учет пользовательских требований в системе “Метатендер” в виде резерва (backlog) автоматизирован и осуществляется в системе Trello - сервисе для управления задачами и проектами организации (рис. 20). Trello помогает распределять ресурсы, ставить задачи и отслеживать их выполнение.

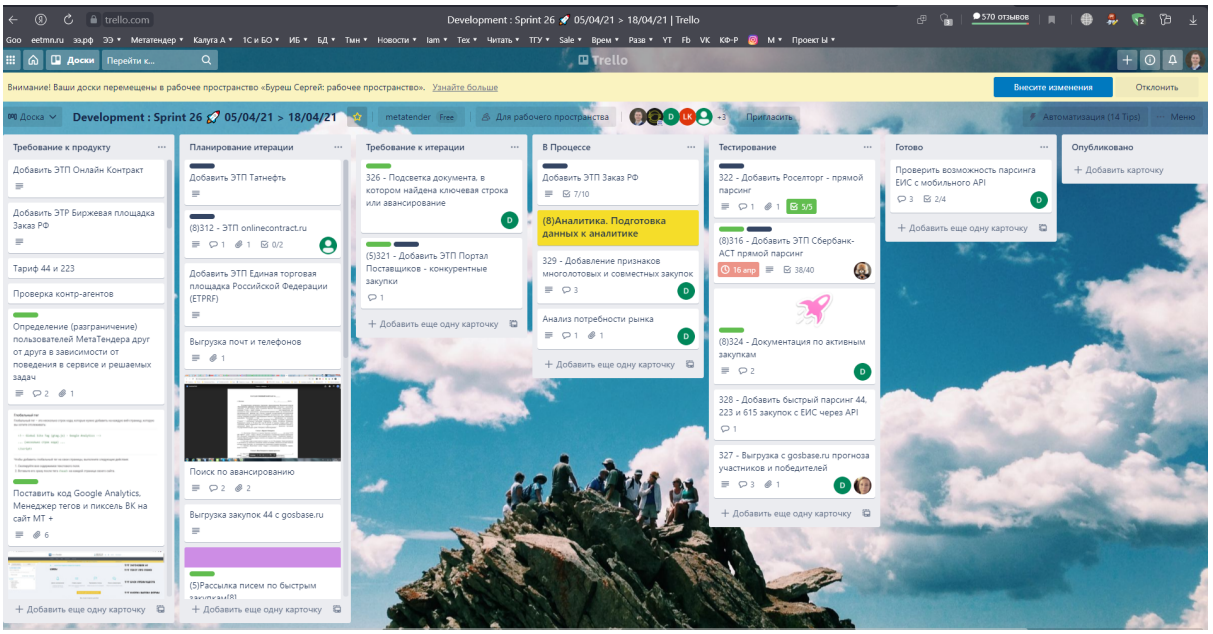


Рис. 20. Скриншот реальной доски резерва (backlog) проекта в Trello.

Процесс распределения приоритетов в разработку (ранжирования) пользовательских историй, осуществляется в первый день спринта в соответствии с рекомендациями фреймворка Scrum. Также, в первый день, после определения приоритетов, команда разработки оцениваем трудозатраты по каждой задачи в порядке очереди, тем самым определяя объем, который она сможет реализовать в течении спринта, с учетом издержек на техническую поддержку.

Доска Srcum состоит из следующих столбцов (этапов):

1. Требования к продукту - наброски, идеи, база пользовательских историй, которые еще не описаны.

2. Планирование итерации - пользовательские истории, которые необходимо обсудить и сформулировать. За каждой пользовательской истории закреплен ответсвенный (эксперт), который отвечает за ее формулирование.
3. Требования итерации - сформулированные пользовательские истории, которые можно брать в работу.
4. В процессе - пользовательские истории, которые взяты командой разработки в работу в спринте.
5. Тестирование - пользовательские истории, разработка по которым завершена и осталось их протестировать.
6. Пользовательские истории - которые прошли тестирование и готовы к опубликованию.
7. Архив - опубликованные пользовательские истории.

Период спринта в проекте по методу Scrum определен на 2 недели. Выбор такого срока связан со следующими факторами:

- скорость разработки и выводом результата разработчиков команды;
- скоростью разработки пользовательских историй участниками проекта;
- маркетинговой эффективностью продвижения икрементов. Раз в неделю - может оказаться более навязчиво. Раз в три недели - это уже более длительный срок для потенциальных клиентов.

Полная команда проекта состоит из следующих ролей и специалистов:

- руководитель проекта - Владелец продукта - Scrum master - автор выпускной квалификационной работы;
- 3 программиста - в балансе в соответствии с бизнес-правилом в части затрат и доходов (проект не должен быть убыточным);

- 2 системных администратора (по четверти ставки - привлекаются вместе очень редко при форс-мажорах);
- руководитель отдела продаж;
- руководитель отдела маркетинга;
- представитель от партнерской сети;
- поклонник сервиса - клиент.

Группа формировалась с учетом представительства от всех групп пользователей, участников разработки и сопровождения инфраструктуры.

Ежедневный scrum - 15-ти минутные дискуссии проводятся в 9-15, не более 30 минут. Время выбрано с учетом:

- утром, команда может получить информацию о происшествиях, которые произошли ночью, принять решение о приоритете их устранения и есть время на их устранение;
- утром, на свежую голову всегда проще генерировать идеи.

Ретроспектива проводится по “классике жанра” - в последний день спринта [24].

Схематично, система управления пользовательскими требованиями отражена на рисунке 21.

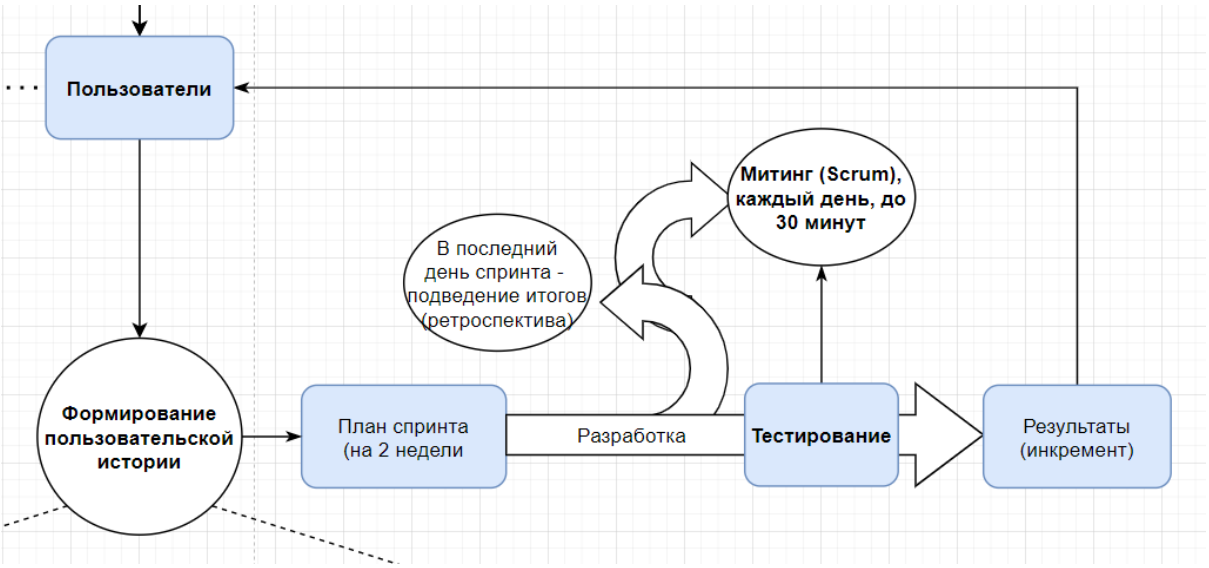


Рис. 21. Система управления пользовательскими требованиями проекта “МетаТендер”.

3.4 Создание пользовательской истории

Создание и первичное описание пользовательской истории начинается с инициатора в любое время. Инициатором может быть любой тип пользователя:

- клиент - кто платит за использование сервиса (“МетаТендер”) в обмен на ее полезность;
- потенциальный клиент - кто еще не платит, но заинтересовался сервисом и готов заплатить, если найдет с помощью нашего решения ответы на свои потребности;
- партнер - кто реализует сервис своим клиентам;
- менеджер отдела продаж - кто реализует сервис в рамках кампании-владельца сервиса;
- Маркетолог - кто осуществляет маркетинговую функцию сервиса “МетаТендер”;
- руководство - кто осуществляет управление проектом.

Мысль или идея инициатора фиксируется кратко в разделе Scrum-доски “Требования к продукту” с такой детализацией, с которой будет понятна идея и полезность требования. Инициатор может указать ответственного за дальнейшее уточнение требования.

В процессе сбора требований участвуют все пользователи, так как каждый из них заинтересован в развитии сервиса.

Владелец продукта (в т.ч. в данном проекте являясь и исполнителем роли Scrum master), за две недели до каждой итерации просматривает данный раздел Scrum-доски на предмет определения приоритета

совместно с группой, отвечающей за конечный финансовый результат проекта. Группа состоит из следующих участников:

1. Владелец продукта (Scrum master).
2. Руководитель отдела продаж.
3. Руководитель отдела маркетинга.

Итоговое решение всегда за владельцем продукта. Для обсуждения приоритетов могут привлекаться и специалисты отдела продаж, партнеры и специалисты отдела маркетинга.

Результатом данного процесса является решение о целесообразности уточнения пользовательской истории, то есть необходимости траты ресурсов компании на дальнейшую проработку требования. Если решение принято о целесообразности реализации данного требования - то карточка перемещается в следующий раздел Scrum доски - “Планирование итерации”. Ошибки и чрезмерное злоупотребление проработкой большого количества требований может повлечь за собой необоснованные траты на отвлечение дорогих ресурсов компании на ненужную работу.

3.5 Первая версия пользовательской истории

Следующим шагом, после того как группа, отвечающая за сбытовой результат проекта, определилась с требованиями которые необходимо описать - владелец продукта определяет первичный порядок детализации всех историй, которые размещены в разделе “Планирование итерации”. В данном разделе могут оказаться истории, которые уже были детально исследованы и описаны. Тогда команда начинает приступать к детализации требования также в порядке ранжирования сверху вниз (по важности). Это сделано на тот случай, если команда не успеет описать все необходимые пользовательские истории - будут готовы хотя бы те, у которых есть наивысший приоритет в разработке.

Следующим этапом является детализация пользовательской истории - ее первой версии, которая будет представлять ценность для пользователя и написание тестов (критериев, по которым будем определяться результат и подтверждение, что разработке завершена. Владелец продукта, совместно со сбытовой группой проекта определяет ответственного за разработку каждой пользовательской истории и тестов и согласовывает с ним срок ее написания. Лучше всего, это должен быть тот. кто инициировал и будет использовать в конечном счете результат реализации требования. Если назначить такого ответственного не представляется возможным - тогда назначается тот, кто лучше всего понимает суть или ближе всех сталкивается с формируемой задачей.

Выбор ответственного для детальной проработки первой версии пользовательской истории обусловлено экономией времени всех участников сбытовой группы. На данном этапе не присутствуют специалисты команды разработки, опять же, для экономии затрат проекта.

В процессе написания истории следует:

- четко выделять для кого она пишется - тип пользователя или роль;
- учесть, что пользовательская история может иметь в своей структуре и ограничения, предписания или правила, которые необходимо учесть;
- пользовательская история не должна быть большой. Одна задача - одна пользовательская история. Срок реализации требования не должны превышать установленные сроки спринта;
- описание интерфейса и как это должно выглядеть необходимо прописывать в последнюю очередь, чтобы не оказаться заложником “красивой картинки” - которая в свою очередь отвлекает от сути истории. Следует помнить, что не каждый результат обязательно следует достигать через интерфейс.

Результатом работы ответственного является сформулированная в письменном виде пользовательская история с детальным описанием тестирования данного требования.

Сама пользовательская история должна быть написана простым доступным текстом, без использования профессионального сленга, чтобы все участники будущей дискуссии могли понять суть однозначно.

3.6 Согласование пользовательской истории

После того, как пользовательская история готова - ответственный инициирует ее обсуждение со следующими участниками:

- сбытовая группа
    - владелец продукта (Scrum master);
    - руководитель отдела продаж;
    - руководитель отдела маркетинга;
- группа разработки

Обязанностями разработчиков является помощь в написании истории, ее максимальной ценности и четких критериев ее приемки - тестов.

При необходимости, могут быть приглашены на дискуссию в зависимости от области использования требования:

- клиенты, являющиеся инициаторами требования или поклонники сервиса;
- представители партнерской сети;
- системные администраторы;
- эксперты по реализуемой области.

До начала встречи, ответственный за пользовательскую историю высылает ссылку на карточку требования с ее описанием. Участники

дискуссии знакомятся с пользовательской историей заранее и готовят вопросы [7].

В процессе самой дискуссии:

- должны быть заданы вопросы, с помощью которых будет очевидна степень проработки требования;
- должно быть выявлено на сколько качество проработано тестирование и можно ли считать история полность соответсвует ожиданиям заказчика;
- должно быть определено, не является ли пользовательская история эпиком, который необходимо разбить на несколько отдельных историй, в виду того, что за одну итерацию реализовать ценность невозможно;
- должна быть произведена предварительная оценка трудоемкости выполнения требования в принятых условных единицах команды разработки;
- постараться посмотреть на описание критически и выявить слабые стороны или как можно ее описать лучше в формате “мозгового штурма”, чтобы увеличилась ценность инкремента.

Результатом самой встречи и дискуссии является четкое понимание разработчиками, что необходимо сделать, чтобы перевести данное требование в код и как ее будут проверять до передачи на этап тестирования [7].

Атрибутами пользовательской истории можно считать:

- независимость требования от других. По возможности следует избегать зависимых друг от друга историй. В случае зависимости или объединять в одну большую или разбить по другому смыслу;
- ценность для пользователя или заказчика;
- возможность четкой оценки, критерий результата;

- компактность - избегать крупных задач, дробить требования на более мелкие и четкие, которые смогут дать максимальный результат;

Схематически разработку и согласование пользовательской истории можно отражена на рисунке 22.

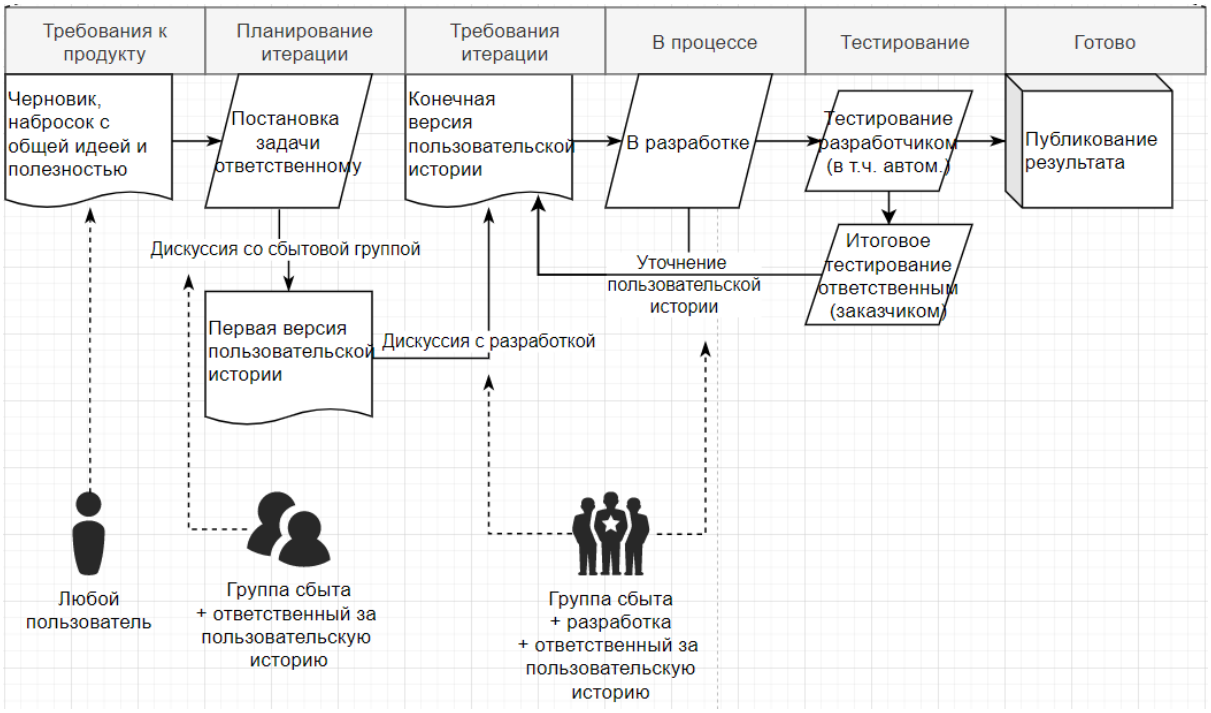


Рис. 22. Схема разработки и согласование пользовательской истории проекта “МетаТендер”.

Важно отметить, что на этом обсуждение пользовательской истории не закончено. Оно возможно, еще в следующих случая:

1. Если в ходе ее разработки у программистов появились дополнительные вопросы, которые были не учтены на предыдущих шагах описания пользовательской истории.
2. В ходе тестирования, если у программистов или ответственного за тестирование появились вопросы, которые также не были учтены ранее.

Обсуждение данных вопросов может привести к срыву сроков реализации и выпуска требования, а значит к финансовым потерям, что

только подтверждает важность и ценность процесса создания пользовательских историй.

Обязательным требованием является то, что истории должны быть написан так, чтобы их можно было протестировать с целью поиска ошибок.

3.7 Тестирование пользовательской истории

Тестирование и приемку пользовательской истории осуществляет тот, кто ее создавал, а значит будет ее использовать так как лучше всех понимает, как это должно работать [25]. В проекте “МетаТендер” это ответственный за пользовательскую историю.

На этапе тестирования обязанности в проекте распределены следующим образом:

- разработчик:
    - отвечает за автоматизацию тестирования системы;
    - находить новые способы тестирования, которые могут выявить дополнительные ошибки или дать ответ, что решение отвечает запросу требования;
- ответственный за пользовательскую историю:
    - создание приемочных тестов. В ходе дискуссии уточняет и развивает тесты совместно с программистом или тестировщиком;
    - выполнение приемочных тестов.

Поэтому ответственному за пользовательскую историю стоит учитывать и время, необходимое на выполнение приемочных тестов, чтобы они не вышли за сроки спринта и были опубликованы в запланированный срок.

В проекте “МетаТендер” различают следующие типы тестов:

- функциональные тесты, которые решают задачи поставленные пользователями;
- тестирование интерфейса - насколько результат соответствует представлению заказчика;
- удобство работы пользователя с решением, насколько ему все понятно, просто и комфортно;
- скорость работы системы, с четкими показателями производительности;
- устойчивость системы - поведение системы в режиме максимальных нагрузок.

Тестирование разбито на два блока. Первоначально, разработчик задачи осуществляет тестирование самостоятельно по тем детализированным требованиям, которые были прописаны в карточке пользовательской истории. Если тестирование предполагает автоматизацию - то реализует автоматизированное тестирование.

Только после того, как разработчик убедился, что его решение полность выполняет условия тестирования и логику пользовательской истории, к процессу приема задачи подключается ответственный за пользовательскую историю. В случае успеха, задача переходит в заключительный блок Scrum доски - “Готов”, для опубликования инкремента.

Если написанная пользовательская история не имеет четких критериев тестирования (например, требования касающиеся удобства использования или дизайна интерфейса) то у команды стоит задача в ходе дискуссии добиться определения хотя бы косвенных критерии приемки таких историй, чтобы приблизится к необходимому результату как можно ближе.

Конечно, дискуссия по уточнению требований возможна и на данном этапе, но как мы понимаем, это самый худший сценарий, четко сигнализирующий, что над пользовательской историей поработали плохо. Такие ситуации должны быть единичны, так как ведут к прямым убыткам в проекта. В проекте “МетаТендер” владелец продукта ведет учет таких ситуаций. Если таких случаев становится много - значит руководителю проекта обязан пересмотреть процесс работы с пользовательскими историями, выявить, на каких этапах чаще всего происходят ошибки и возникают недопонимания в дискуссиях и внести изменения в процесс.

Такой подход позволяет контролировать качество пользовательских историй - а значит и разработку в целом.

ГЛАВА 4. Разработка системы на основе пользовательских историй

В процесс дискуссий с пользователями информационной системы “МетаТендер” были определены 5 основных требования:

1. Выгрузка списка клиентов из ЕРУЗ (Единый реестр участников закупок) для маркетологов [26].
2. Обеспечение возможности выгрузки документов тендеров из самой информационной системы “МетаТендер” для авторизованных пользователей.
3. Отображение прогноза возможных участников тендера, для авторизованных пользователей.
4. Отображение в поиске закупок с авансом, для авторизованных пользователей.
5. Отражение закупок с электронной торговой площадки “Березка”, для авторизованных пользователей [27].

Чтобы не перегружать данную выпускную квалификационную работу, первые две пользовательские истории мы рассмотрим подробнее, остальные отразим справочно в приложении 1.

4.1. Список клиентов из ЕРУЗ

Пользовательская история: Я как маркетолог должен иметь возможность проверить потенциального или действующего клиента в реестре квалифицированных подрядных организация, чтобы определить клиентов к Целевой Аудитории и предложить им актуальные услуги [26].

Тесты:

Ограничения:

- Исторические данные должны быть загружены.

- Реестр должен обновляться ежедневно.

1. Авторизуюсь под пользователем с ролью "Менеджер".
2. Перехожу на страницу: https://metatender.ru/coldcall/export.
3. В "Источнике" мне доступна опция "ЕРУЗ".
4. Выбрав эту опцию в "ЕРУЗ", то становятся доступны следующие фильтры: "Дата регистрации", "Регион", "Статусы лицензии МТ", "Форма регистрации".
5. Если я устанавливаю одни или несколько "Статусы лицензии МТ", то будут отображены только те авторизированные пользователи “МетаТендера”, которые присутствую в ЕРУЗ и в выбранных статусах.
6. Если я устанавливаю опцию "Дата регистрации", то будут отображены только те записи ЕРУЗ, дата регистрации которых входят в выбранный диапазон дат.
7. Если я устанавливаю опцию "Регион", то будут отображены только те записи ЕРУЗ, которые содержат выбранные регионы.
8. Если я устанавливаю опцию "Форма регистрации", то будут отображены только те записи ЕРУЗ, которые содержат выбранные "Форма регистрации".
9. Если не установлена не одна из опций "Статусы лицензии МТ", то отображаем записи независимо от наличия в них клиента МТ.
10. При нажатии на кнопку "Количество", выводится количество найденных записей согласно указанным опциям.
11. Нажимаю кнопку "Экспорт", получаю csv с выбранными опциями и следующими полями:

- Номер в реестре
- Статус в МетаТендер
- Статус регистрации
- Форма
- Полное наименование
- Сокращенное наименование
- ОКВЭД
- ИНН

- КПП
- Код региона
- Регион
- ОГРН
- Лица, имеющие право без доверенности действовать от имени юридического лица
- Является субъектом малого предпринимательства
- Дата регистрации в ЕИС
- Дата окончания срока регистрации в ЕИС
- Дата постановки на учет в налоговом органе
- Дата регистрации ИП/ЮЛ
- Адрес в пределах места нахождения
- Почтовый адрес
- Email
- Телефон
- Площадки

Реализация

Логика серверной части процесса находится в микросервисе MetaTender.Services.General, который запущен в контуре Яндекса в Docker-контейнере. При старте сервиса создается и запускается hosted-сервис DaemonService, в котором инициализируется консьюмер сообщений из RabbitMQ EruzConsumer, а также раз в день отправляется сообщение на забор всех страниц из списка ЕРУЗ в EruzConsumer за последние 2 дня. В EruzConsumer есть обработчики двух очередей: eruz.parse.page - очередь, содержащая сообщения на забор конкретной страницы из списка ЕРУЗ в ЕИС и eruz.parse.card - очередь, содержащая сообщения на забор и парсинг конкретной карточки из ЕРУЗ.

Обработчик очереди eruz.parse.page ParsePageConsume получает сообщение из очереди, содержащее в себе информацию о нужной странице для забора, производит скачивание HTML-документа страницы, извлекает из него карточки участников закупок и для каждой карточки генерирует сообщение в очередь eruz.parse.card, куда передает информацию о ней.

Обработчик очереди eruz.parse.card ParseCardConsume получает сообщение из очереди, содержащее в себе информацию о нужной карточке участника торгов для забора, производит скачивание HTML-документа карточки, извлекает из него нужную информацию, которую потом сохраняет в соответствующую таблицу в клиентскую базу данных.

На рисунке 23 представлена диаграмма классов микросервиса MetaTender.Services.General, ответственных за получение информации из ЕРУЗ и сохранении ее в клиентскую базу данных.

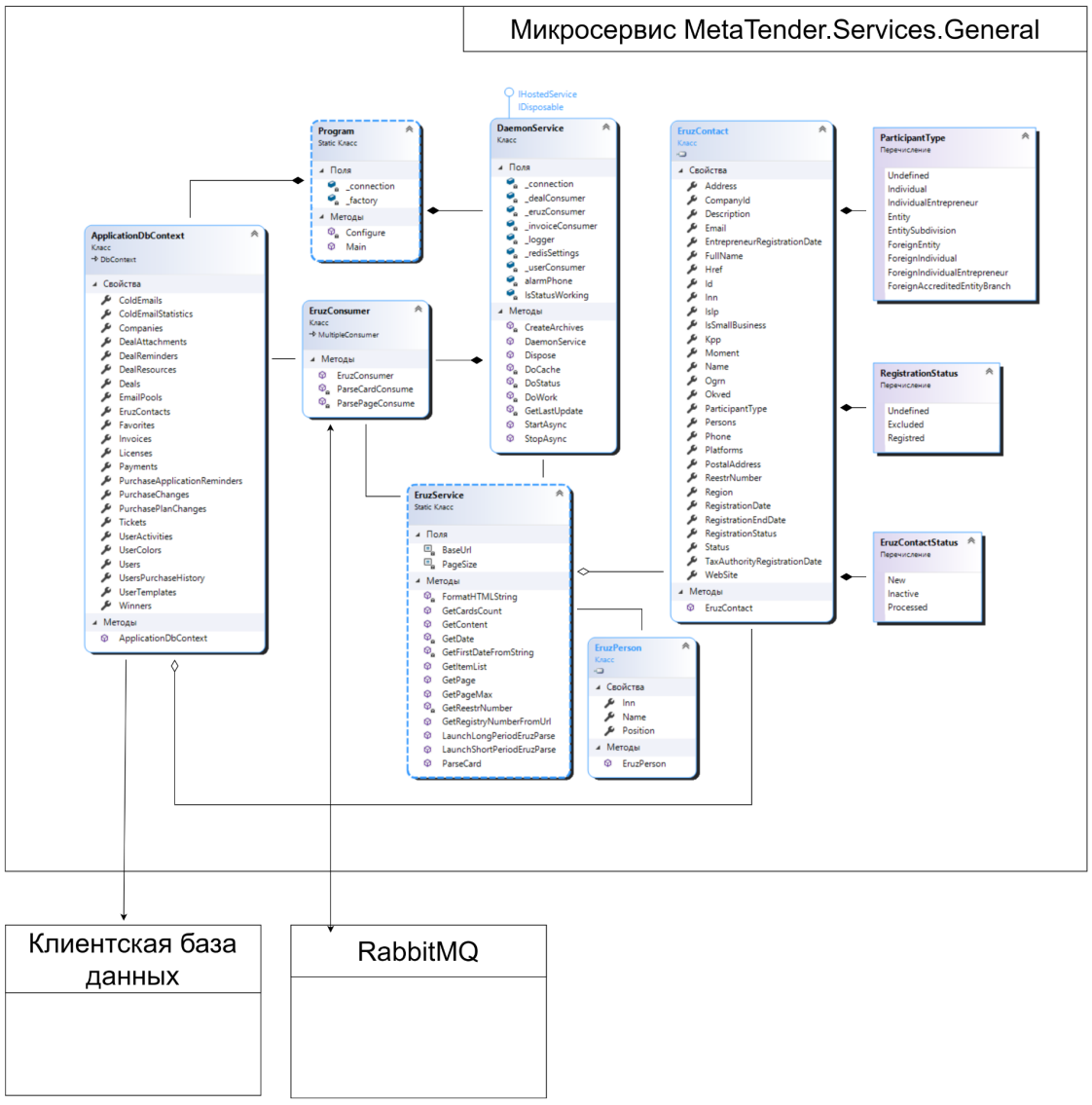


Рис 23. Диагр. классов микросервиса MetaTender.Services.General.

Описание классов:

Program - точка входа приложения, инициализирует различные настройки, контексты базы данных (в данном случае только контекст клиентской БД), запускает hosted-сервис DaemonService.

DaemonService - инициализирует логгер, консьюмеры (для работы с сообщениями из брокера очередей RabbitMQ), выполняет запланированные работы (например, в данном случае отправка команды в EruzConsumer для начала очередного забора данных из ЕИС).

EruzConsumer - консьюмер сообщений, отвечает за сбор и обработку данных по ЕРУЗ из ЕИС с помощью вспомогательных методов из EruzService и записи результатов обработки в клиентскую базу данных с помощью ApplicationDbContext.

EruzService - статический класс, содержащий в себе методы по получению и обработке данных ЕРУЗ.

EruzPerson - вспомогательный класс.

ApplicationDbContext - контекст клиентской базы данных, использующий EntityFrameworkCore, отвечающий за различные операции чтения и записи в нее.

EruzContact - класс, принадлежащий контексту клиентской базы данных, представляющий собой сущность данных ЕРУЗ, хранящихся в соответствующей базе данных.

ParticipantType, RegistrationStatus, EruzContactStatus - вспомогательные перечисления для облегчения записи информации в EruzContact.

Реализация

Логика части веб-сервера, находящегося в контуре Яндекса, заключается в обработке контроллером ColdCallController запросов менеджеров на экспорт информации из ЕРУЗ, хранящейся в клиентской базе данных. При поступлении запроса на экспорт информации от пользователя в ColdCallController, проверяется его статус, если пользователь - менеджер, то из клиентской базы данных запрашиваются данные, согласно критериям, указанным пользователем, формируется excel-документ и отправляется ему для скачивания.

На рисунке 24 представлена диаграмма классов веб-приложения, ответственных за предоставление информации из ЕРУЗ, хранящейся в базе данных менеджерам проекта МетаТендер.

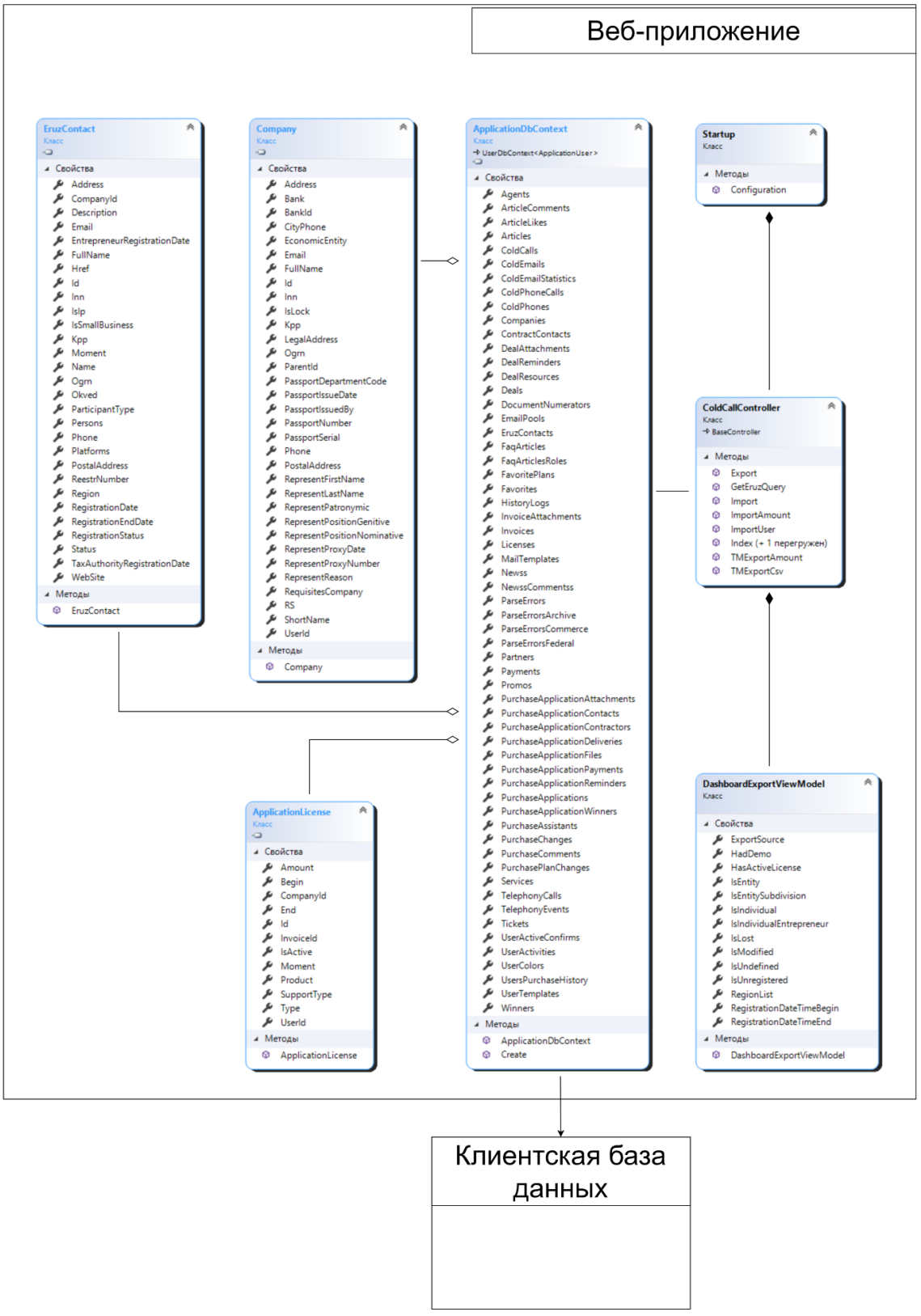


Рис 24. Диаграмма классов веб-приложения данных ЕРУЗ.

Startup - класс, инициализирующий конфигурацию приложения и основные классы.

ColdCallController - контроллер, обрабатывающий запросы на выгрузку данных, в том числе ЕРУЗ.

DashboardExportViewModel - модель представления для страницы выгрузки ЕРУЗ.

ApplicationDbContext - см. выше.

EruzContact - см. выше.

Company - класс, принадлежащий контексту клиентской базы данных, представляющий собой сущность данных компаний-клиентов МетаТендер, используемый в процессе выгрузки.

ApplicationLicense - класс, принадлежащий контексту клиентской базы данных, представляющий собой сущность данных лицензий МетаТендер, используемых в процессе выгрузки.

Итоговый пользовательский интерфейс решения отражен на рисунке 25.

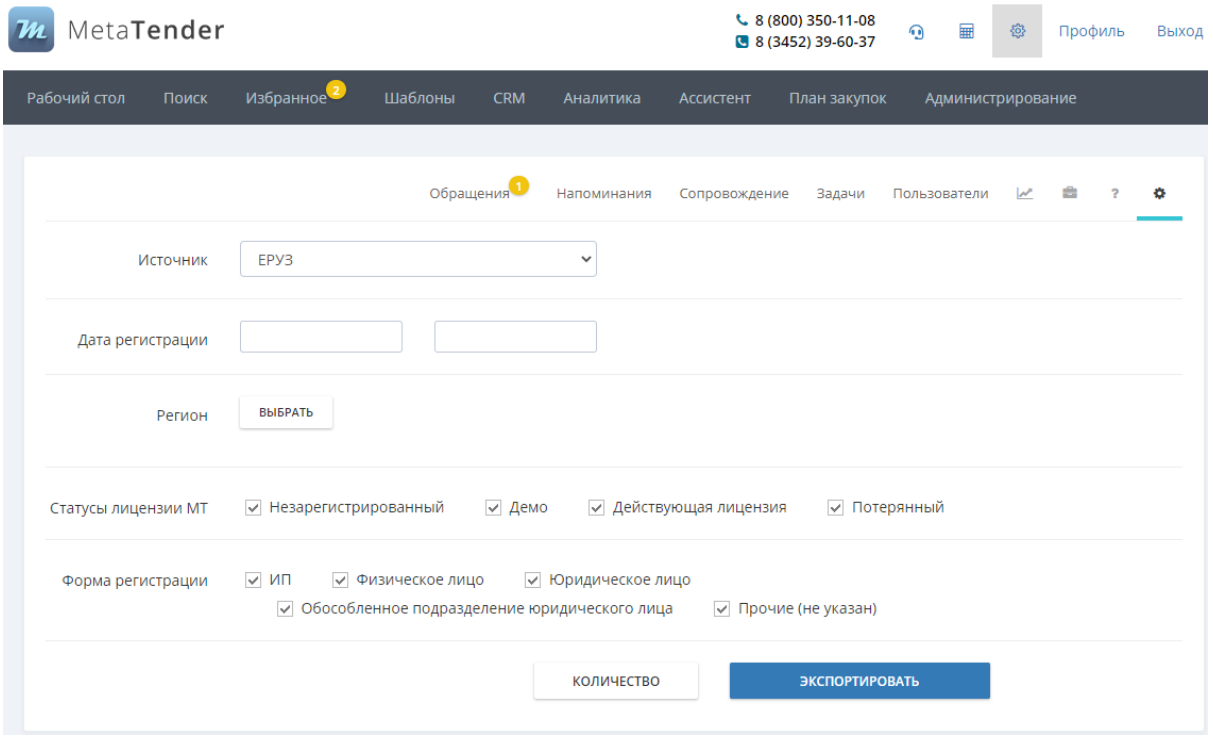


Рис. 25. Итоговый интерфейс пользовательской истории выгрузки списка клиентов из ЕРУЗ.

4.2. Доступ к документации

Пользовательская история: как авторизованный пользователь, я хочу, чтобы в “МетаТендере” хранилась документация по активным закупкам и я мог скачать нужные документы даже если сайт-источник не работает.

Тесты:

1. Храним всю документацию по закупке
2. Закупка должна быть активной или завершенной до 2 месяцев назад.
3. Все документы в закупке должны открываться
4. Касается всех видов закупок: 44-ФЗ, 223-ФЗ и проч [4] [5].
5. Если что-то не можем найти или хранить, должно быть четкое объяснение, почему документации у нас нет.

6. По истечении 2 месяцев должна быть ссылка на ЭТП с документацией (у себя не храним) - как есть сейчас.
7. Проверяем рандомно все виды активных и завершенных не позднее 2 месяцев назад закупок по 10 закупок каждого вида.
8. Смотрим на наличие и открываемость документов.
9. Сопоставляем с документацией в источнике.
10. В идеале - поймаем ситуацию, когда площадка, откуда выгружаем документацию, на техобслуживании и проверим наличие документов у нас.

Реализация

Логика серверной части процесса находится в микросервисе Documentation, который запущен в контуре Яндекса в Docker-контейнере. При старте сервиса создается и запускается hosted-сервис DaemonService, в котором инициализируется консьюмер сообщений из RabbitMQ DocumentationConsumer, а также: раз в день отправляется сообщение на удаление документации неактивных закупок из Yandex Object Storage и каждую минуту отправляется сообщение о необходимости загрузки документации новых активных закупок в Yandex Object Storage. В DocumentationConsumer есть обработчики трех очередей:

1. purchase_documentation.get_period - очередь, обрабатываемая методом GetPurchaseDocumentationPeriodConsume, который получает информацию о периоде, за который необходимо загрузить документацию в Yandex Object Storage для активных закупок, извлекает их из ресурсной базы данных и для каждой закупки генерирует сообщение в очередь purchase_documentation.download_and_send_to_bucket;
2. purchase_documentation.download_and_send_to_bucket - очередь, обрабатываемая методом

DownloadAndSendPurchaseDocumentationToBucketConsume, который получает информацию о закупке, скачивает с сайта ЕИС ее документацию, отправляет ее в Yandex Object Storage и записывает в таблицу PurchaseDocuments ресурсной базы данных информацию о том, что документация к данной закупке есть в наличии в Yandex Object Storage;

3. purchase_documentation.delete_from_bucket - очередь, обрабатываемая методом DeleteDocumentationFromBucketConsume, который получает информацию о том, что необходимо удалить документацию у неактивных закупок. Из ресурсной базы данных извлекается информация о документах завершившихся закупок, находящихся в Yandex Object Storage и выполняется процедура их удаления из него вместе с удалением соответствующих записей из таблицы PurchaseDocuments ресурсной базы данных.

На рисунке 26 представлена диаграмма классов микросервиса Documentation, ответственных за загрузку документации из ЭТП и отправку этой документации в Yandex Object Storage.

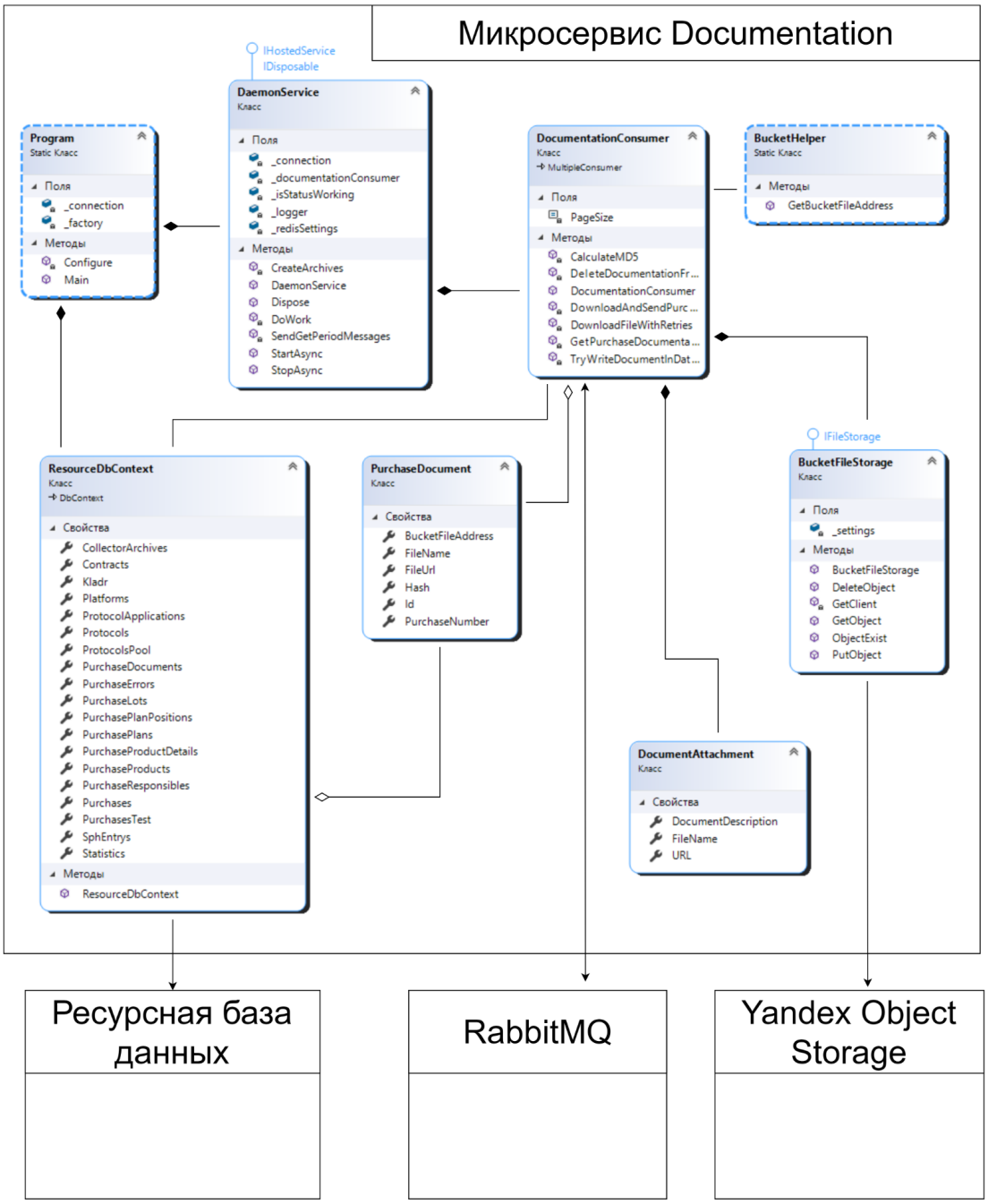


Рис. 26. Диаграмма классов микросервиса Documentation.

Program - см. 4.1. Список клиентов из ЕРУЗ.

DaemonService - см. 4.1. Список клиентов из ЕРУЗ.

DocumentationConsumer - консьюмер сообщений, отвечает за различные операции с документацией закупок (скачивание, загрузку в

Yandex Object Storage, удаление неактуальной документации и т. д.) с помощью BucketFileStorage.

BucketHelper - вспомогательный класс, содержащий в себе логику нахождения корректного адреса документа в Yandex Object Storage.

BucketFileStorage - класс, содержащий логику операций с файлами в Yandex Object Storage.

DocumentAttachment - класс, облегчающий работу с данными о документации закупки.

ResourceDbContext - контекст ресурсной базы данных, использующий EntityFrameworkCore, отвечающий за различные операции чтения и записи в нее.

PurchaseDocument - класс, принадлежащий контексту ресурсной базы данных, представляющий собой сущность данных связи между документации закупки и файлами в Yandex Object Storage.

На рисунке 27 представлена диаграмма классов веб-приложения, ответственных за выдачу пользователям документации по закупкам, загруженной в Yandex Object Storage.

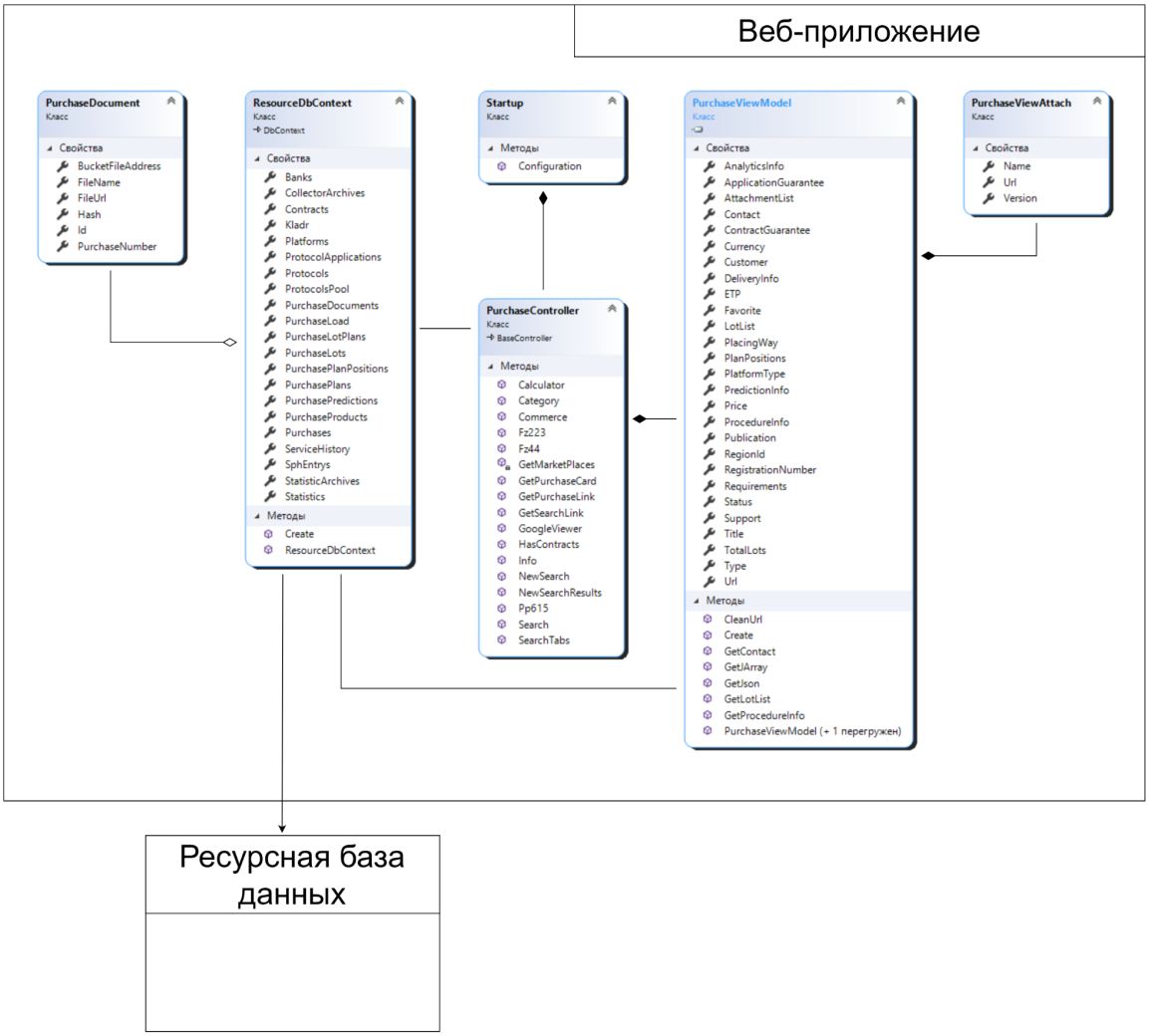


Рис. 27. Диаграмма классов веб-приложения пользовательской истории доступа к документации тендеров.

Startup - см. 4.1. Список клиентов из ЕРУЗ.

PurchaseController - контроллер, отвечающий за выдачу информации по закупкам, включая информацию по документации.

PurchaseViewModel - модель представления для страницы информации по закупке.

PurchaseViewAttach - вспомогательный класс, облегчающий работу с данными о документации закупки.

ResourceDbContext - см. выше.

PurchaseDocument - см. выше.

Реализация

Логика части веб-сервера, находящегося в контуре Яндекса, заключается в обработке контроллером PurchaseController запросов клиентов на скачивание закупочной документации. При поступлении подобного запроса на скачивание документа определенной закупки, система проверяет, есть ли запись о нем в таблице PurchaseDocuments ресурсной базы данных, если есть - то, оттуда извлекается URL-адрес этого документа в Yandex Object Storage пользователь перенаправляется по нему для скачивания. Если такой записи нет - то пользователь перенаправляется по адресу скачивания документации на самой ЭТП.

Итоговый пользовательский интерфейс отражен на рисунке 28.

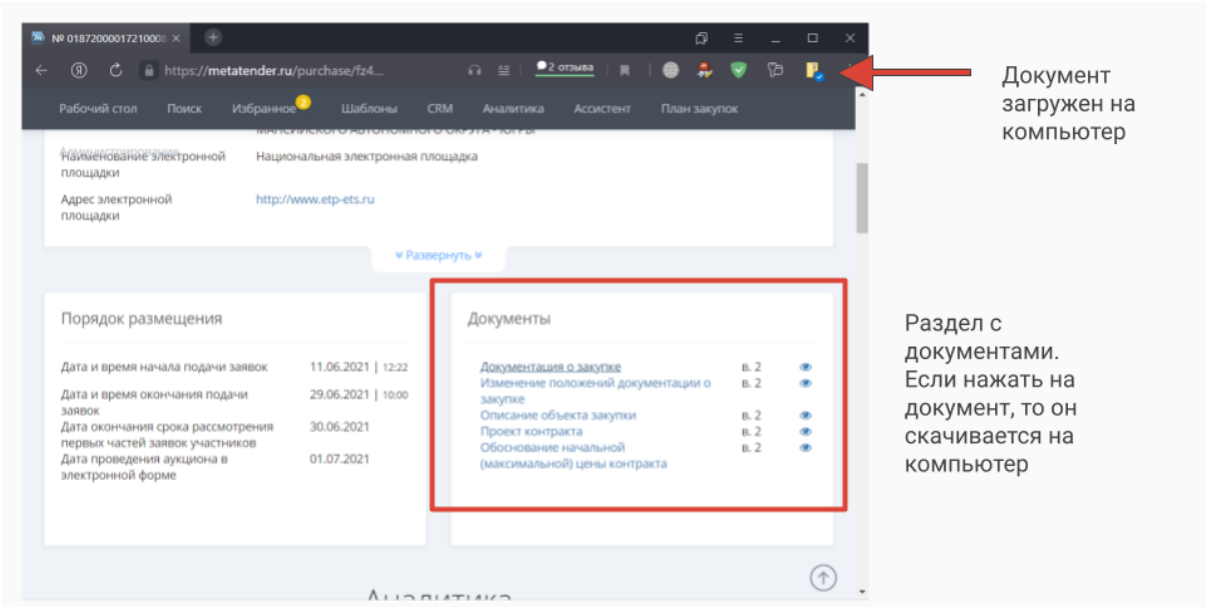


Рис. 28. Итоговый интерфейс пользовательской истории доступа к документации тендеров.

ЗАКЛЮЧЕНИЕ

В данной работе было детально изучен и представлен процесс создания и перевода пользовательской истории от описания к тестированию.

Заявленная стратегическая цель компании на первый год (увеличение доли продаж сервиса по поиску тендеров от всего объема продаж в первый год в три раза (с 6,27% до 19%) была выполнена, факт составил 20,20%.

Также, были реализованы все четыре задачи:

1. *Разработать архитектуру архитектуру проекта, на которой будет реализовано решение* - реализовано, Глава 1;
2. *Выбрать метод управления проектом разработки и сформировать команду* - выбран, Agile-Scrum, Глава 2.
3. *Разработать процесс управления требованиями в проекте* - разработка - Глава 3.
4. *В течении первого года реализовать основные требования пользователей системы поиска и анализа тендеров* - требования определены и реализованы, Глава 4.

Методика Agile и фреймворк Scrum фактически не является ни методологией, ни стандартом, а набором принципов и ценностей. Новацией данной дипломной работы является создание системы управления требованиями проекта “МетаТендер”.

Основной пользовательских историй гибкой разработки является - дискуссия. Именно дискуссия позволяет команде разработки выявить все необходимые детали и полную информацию обсуждаемого требования проекта.

Результатом дискуссии является определение приемочных тестов, которые фактически являются критериями выполнения пользовательской истории. Тесты определяются до написания кода и определяют необходимый функционал разработки.

Тесты, как и пользовательскую историю, описывает заказчик. Заказчик и осуществляет приемку результата на основании сформулированных приемочных тестов.

Разработчики помогают сформулировать пользовательскую историю, уточняют всю необходимую информацию и помогают определить тесты. Если требуется, программируют автоматизированные тесты. Принимают решение о качестве выполненной работы на основе приемочных тестов.

В дипломной работе описана подробная структура управления требованиями до выпуска инкремента.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Портер М. Конкурентное преимущество: как достичь высокого результата и обеспечить его устойчивость: пер. с англ. / М. Портер. М.: Альпина Бизнес Букс, 2005. 715 с.
2. Карл Сьюэлл, Пол Браун. Клиенты на всю жизнь. Перевод: Михаил Иванов, Михаил Фербер. Издательство: МИФ, 2021. 232 с.
3. Филип Котлер , Нильс Бикхофф. Стратегический менеджмент по Котлеру: Лучшие приемы и методы. Переводчик И. Матвеева. ООО «Альпина Паблишер», 2016. 143 с.
4. Закон о контрактной системе (закон о госзакупках). Федеральный закон от 5 апреля 2013 г. N 44-ФЗ "О контрактной системе в сфере закупок товаров, работ, услуг для обеспечения государственных и муниципальных нужд" (с изменениями и дополнениями) — режим доступа: URL: https://base.garant.ru/70353464/
5. Федеральный закон от 18 июля 2011 г. N 223-ФЗ "О закупках товаров, работ, услуг отдельными видами юридических лиц" (с изменениями и дополнениями) — режим доступа: URL: https://base.garant.ru/12188083/
6. Информационная система поиска и анализа тендеров // МетаТендер. URL: https://metatender.ru
7. Карл Вигерс, Джой Битти. Разработка требований к программному обеспечению. Издание третье, дополненное. BHV, 2020 г. 726 с.
8. Нил Форд, Патрик Куа, Ребекка Парсонс. Эволюционная архитектура. Поддержка непрерывных изменений. Переводчик: А. И. Демьяников. Питер, 2018. 272 с.

9. Водяхо А.И., Выговский Л.С., Дубенецкий В.А., Цехановский В. В. Архитектурные решения информационных систем. 2-е изд., перераб. Лань, 2017. 356 с.
10. Закон "О персональных данных" — режим доступа: URL: https://base.garant.ru/12148567/
11. Вадим Алджанов. ИТ-архитектура от А до Я: Теоретические основы. Первое издание. Издательские решения, 2018. 680 с.
12. Документация // Яндекс облако. URL: https://cloud.yandex.ru/docs\\
13. Скотт В. Эмблер, Прамодкумар Дж. Садаладж. Рефакторинг баз данных. Эволюционное проектирование. Вильямс, 2016. 368 с.
14. Open source search server. URL: http://sphinxsearch.com
15. Elastic Enterprise Search. URL: https://www.elastic.co/elasticsearch
16. Ричардсон Крис. Микросервисы. Паттерны разработки и рефакторинга. Питер, 2019 г. 544 с.
17. Зиндер Е.З. Системы управления базами данных. Открытые системы, 1997, выпуск №3.
18. ГОСТ Р ИСО/МЭК 12207-2010. Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств. Режим доступа: URL: https://base.garant.ru/70146140/
19. ГОСТ 34.601-90 Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания. Режим доступа - URL: https://base.garant.ru/187735/
20. И. А. ПЕРЛ, О. В. КАЛЁНОВА. Введение в методологию программной инженерии. Университет ИТМО, 2019. 53 с.
21. Е.А. Кумагина, Е.А. Неймарк. Модели жизненного цикла и технологии проектирования программного обеспечения.

Нижегородский государственный университет им. Н.И. Лобачевского, ННГУ, 2016. 41 с.

22. Эндрю Стеллман и Дженнифер Грин. Постигая Agile. Ценности, принципы, методологии. Манн, Иванов и Фербер ООО, 2017. 448 с.
23. Роберт Мартин. Чистый Agile. Основы гибкости. Переводчик: И. Сигайлюк. Питер 2020. С. 240
24. Майк Кон. Scrum: гибкая разработка ПО. Переводчик: И. В. Диалектика-Вильямс 2011. Красиков С. 575
25. Кен Швабер и Джефф Сазерленд. Авторитетное руководство по Скраму: Правила Игры.
26. Единый реестр участников закупок // Единая информационная система в сфере закупок. Режим доступа - URL: https://zakupki.gov.ru/epz/eruz/search/results.html
27. Реестр закупок // Электронная торговая площадка “Березка. ”Режим доступа - URL: https://agregatoreat.ru/purchases-registry/all

ПРИЛОЖЕНИЕ 1. Прогнозирования участников тендера

Пользовательская история: как авторизованный пользователь, хочу получить список возможных участников тендера, для оценки вероятности победы в нем.

Тесты:

1. В карточке тендера отражается раздел “Прогноз победителей”.
2. В разделе отражены:
    a. Подобные состоявшиеся аукционы того же заказчика с учётом вида деятельности
    b. Подобные состоявшиеся аукционы того же заказчика с учётом контекста ключевых слов
    c. Подобные состоявшиеся аукционы из той же области от других заказчиков
3. По данным аукционам отражена следующая информация:
    - № возможного участника
    - Возможный участник
    - Предмет его состоявшегося тендера
    - Дата его победы
    - ИНН и КПП возможного участника
    - Начальная максимальная цена контракта
    - Цена контракта
    - % снижения

Описание используемых классов и диаграмма классов на рисунке 29.

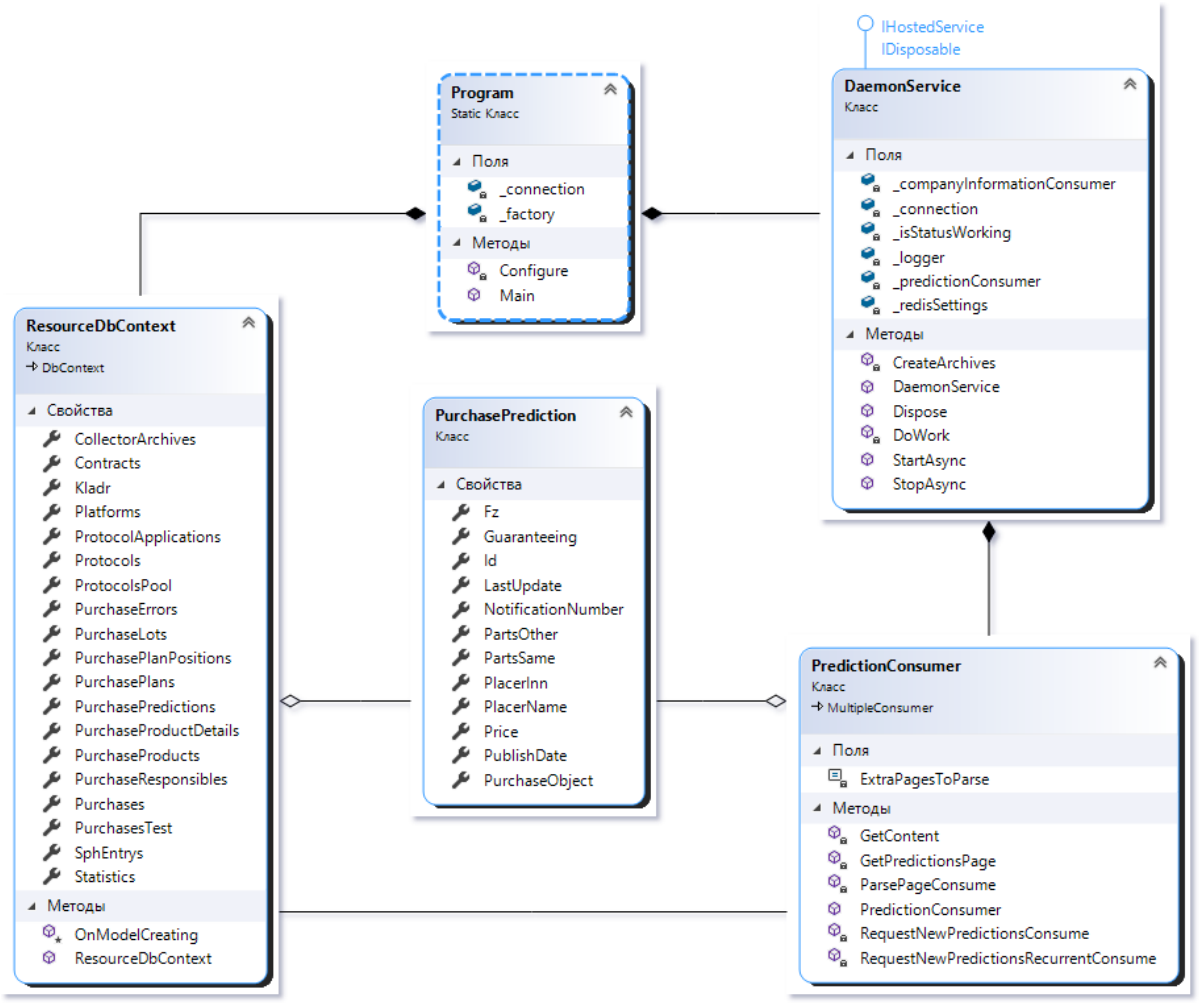


Рис. 29. Диаграмма классов сервисов пользовательской истории отражения прогноза возможных участников тендера.

Program - см. 4.1. Список клиентов из ЕРУЗ.

DaemonService - см. 4.1. Список клиентов из ЕРУЗ.

PredictionConsumer - консьюмер сообщений, отвечает за обращение к API проекта GosBase и забор информации по прогнозированию победителей закупок.

ResourceDbContext - см. 4.2. Доступ к документации.

PurchasePrediction - класс, принадлежащий контексту ресурсной базы данных, представляющий собой сущность данных победителей закупок.

Описание используемых классов веб-приложения и диаграмма классов на рисунке 30.

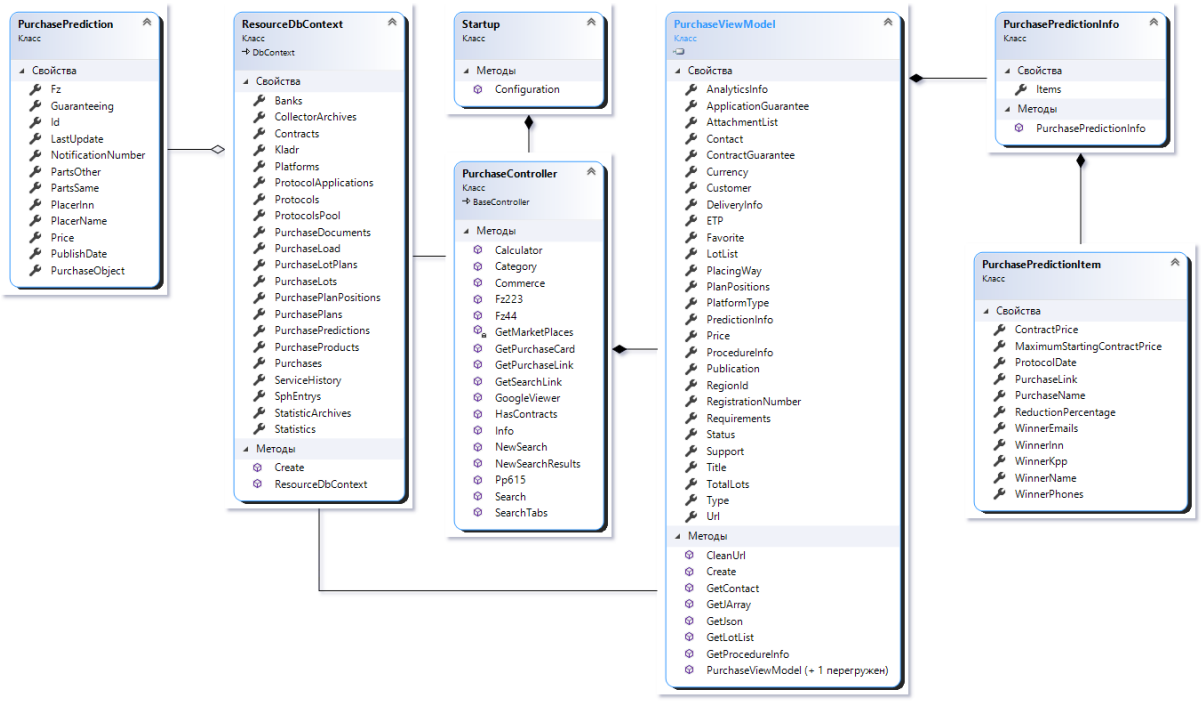


Рис. 30. Диаграмма классов веб-приложения пользовательской истории отражения прогноза возможных участников тендера.

Startup - см. 4.1. Список клиентов из ЕРУЗ..

PurchaseController - см. 4.2. Доступ к документации.

PurchaseViewModel - см. 4.2. Доступ к документации.

PurchasePredictionInfo - класс, содержащий в себе список из PurchasePredictionItem.

PurchasePredictionItem - класс, содержащий в себе информацию о возможных победителях для закупки.

ResourceDbContext - см. 4.2. Доступ к документации.

PurchasePrediction - см. выше.

Итоговый пользовательский интерфейс отражен на рисунке 31.

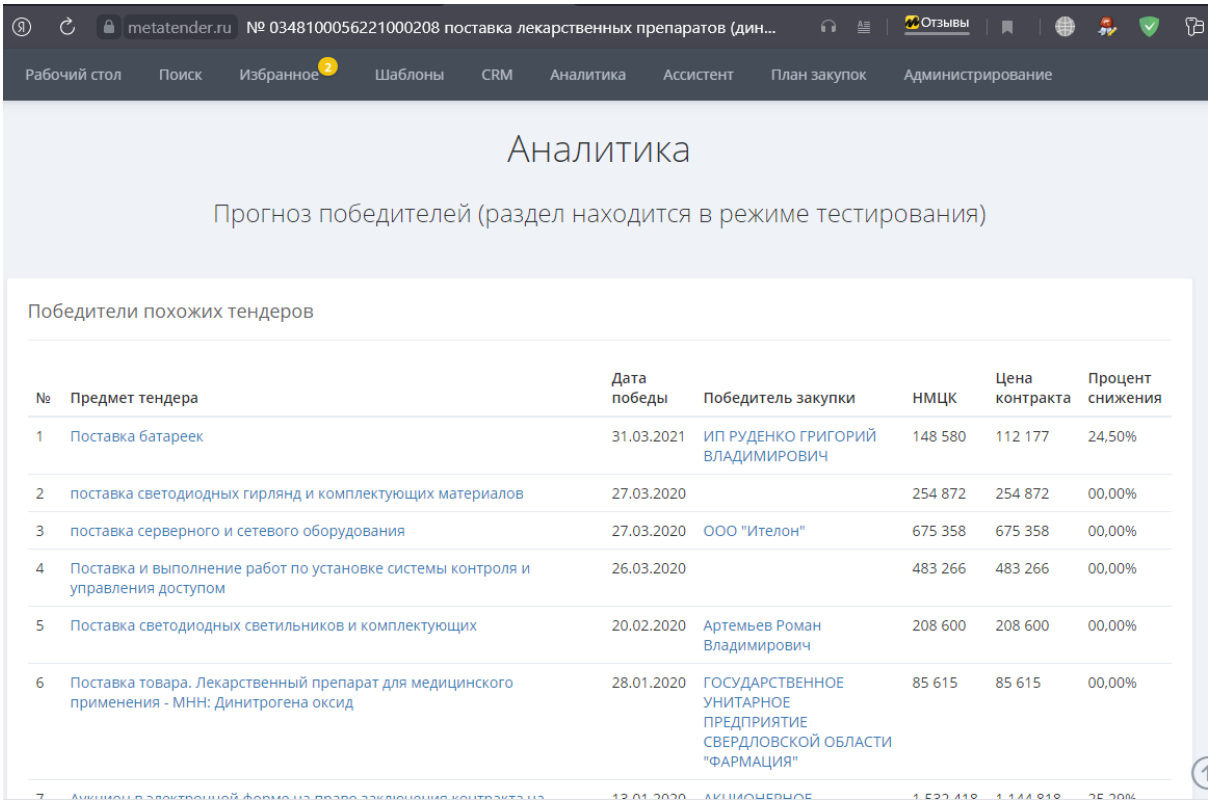


Рис. 31. Итоговый интерфейс пользовательской истории отражения прогноза возможных участников тендера.

ПРИЛОЖЕНИЕ 2. Поиск тендеров с авансом.

Пользовательская история: как авторизованный пользователь, я хочу искать закупки с авансом, чтобы получать только нужные мне закупки

Тесты:

1. В фильтрах поиска закупок есть возможность установить признак "С Авансом".
2. В результатах поиска с признаком "С Авансом" должны отображаться закупки у которых есть авансовые платежи.

Диаграмма классов веб-приложения отражена на рисунке 32.

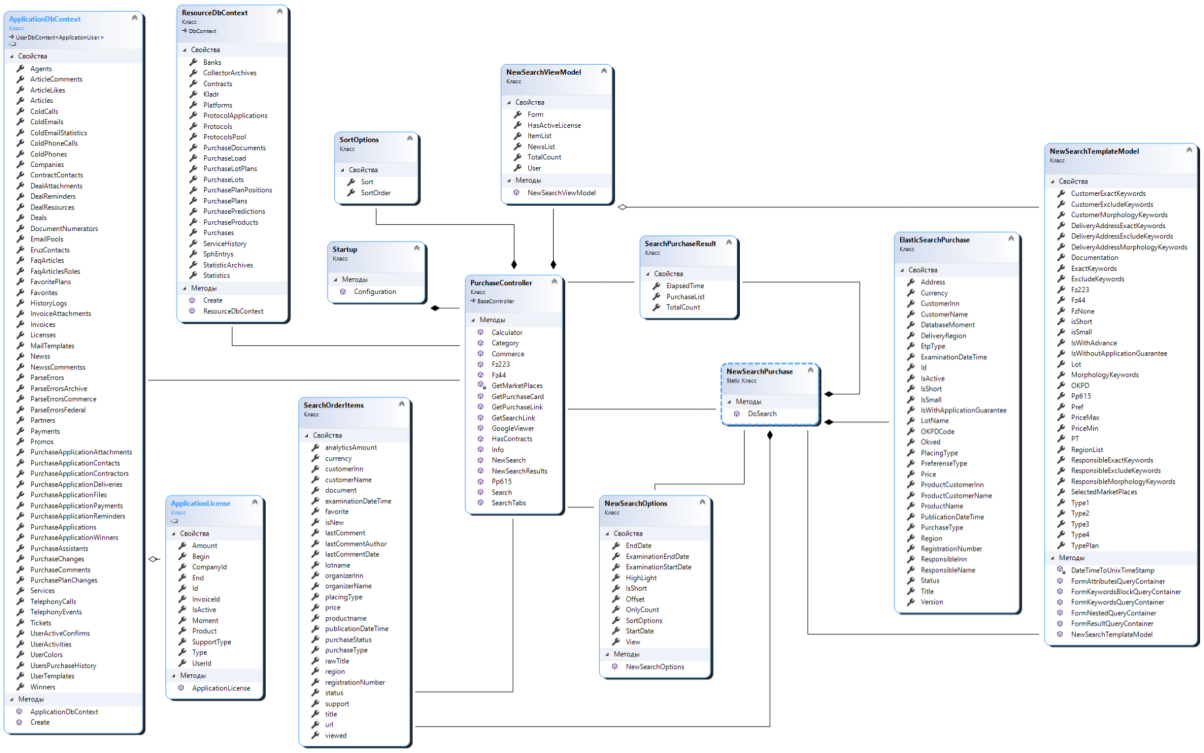


Рис. 32. Диаграмма классов веб-приложения пользовательской истории поиска тендеров с авансом.

Startup - см. 4.1. Список клиентов из ЕРУЗ.

PurchaseController - см. 4.2. Доступ к документации. В данном случае также отвечает за поисковые запросы по закупкам.

NewSearchViewModel - модель представления для страницы поиска по закупкам.

ResourceDbContext - см. 4.2. Доступ к документации.

ApplicaitonDbContext - см. 4.1. Список клиентов из ЕРУЗ.

ApplicationLicense - см. 4.1. Список клиентов из ЕРУЗ.

NewSearchPurchase - класс, отвечающий за поиск закупок по введенным пользователем условиям, хранящимся в NewSearchOptions, а также по данным о сортировке поиска, хранящимся в SortOptions.

SearchPurchaseResult - класс, содержащий данные о результатах поиска.

ElasticSearchPurchase - класс, представляющий модель данных для закупки из поисковой машины ElasticSearch.

SearchOrderItems - класс, содержащий модель для списка закупок, для отображения в представлении.

NewSearchTemplateModel - класс, содержащий информацию о пользовательских шаблонах поиска.

Итоговый пользовательский интерфейс отражен на рисунке 33.

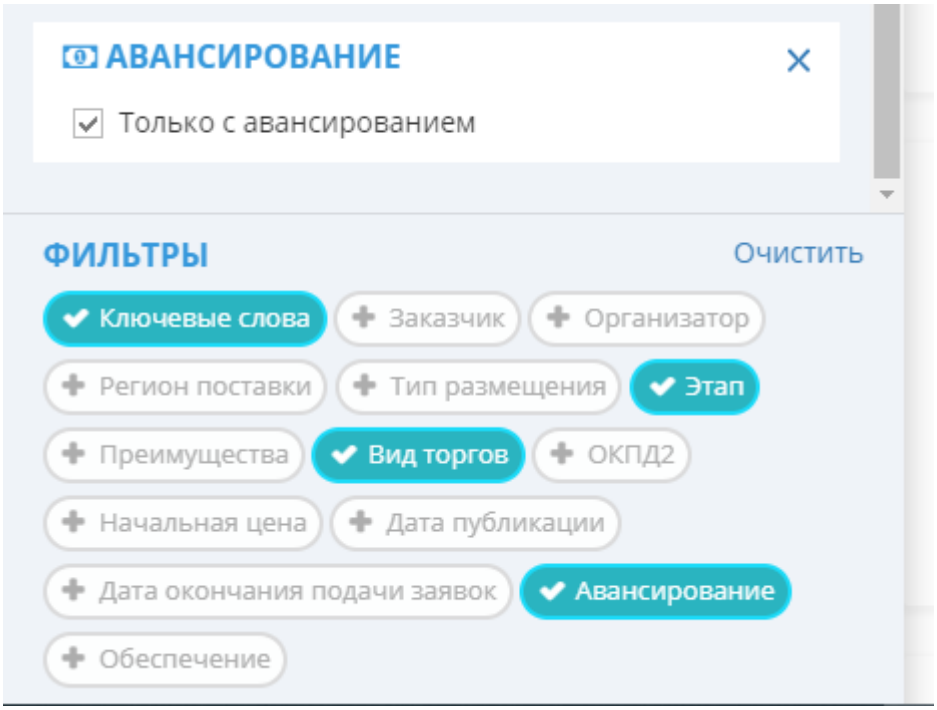


Рис. 33. Пользовательский интерфейс веб-приложения пользовательской истории поиска тендеров с авансом.

ПРИЛОЖЕНИЕ 3. Отражение тендеров площадки “Березка”

Пользовательская история: как авторизованный пользователь, я хочу получать закупки из электронной торговой площадки https://agregatoreat.ru/ для того, чтобы получать нужные мне тендеры с данной площадки [27].

Тесты:

1. В разделе поиска в системе МетаТендер устанавливаем следующий фильтр поиска:
    a. дата публикации, вчерашнее число (на момент тестирования);
    b. этап: подача заявки;
    c. площадка: Березка.
2. В результате поиска, количество опубликованных тендеров должно быть равно аналогичному поиску на самой площадке.
3. В карточке тендера заполнены все поля из тендера по данной площадке.
4. В реестре контрактов загружены все контракты. Количество контрактов прошлого года на площадке равно количеству отображаемых контактов в разделе Контрактов вкладки Аналитики “МетаТендера”.

    Диаграмма классов отражена на рисунке 34.

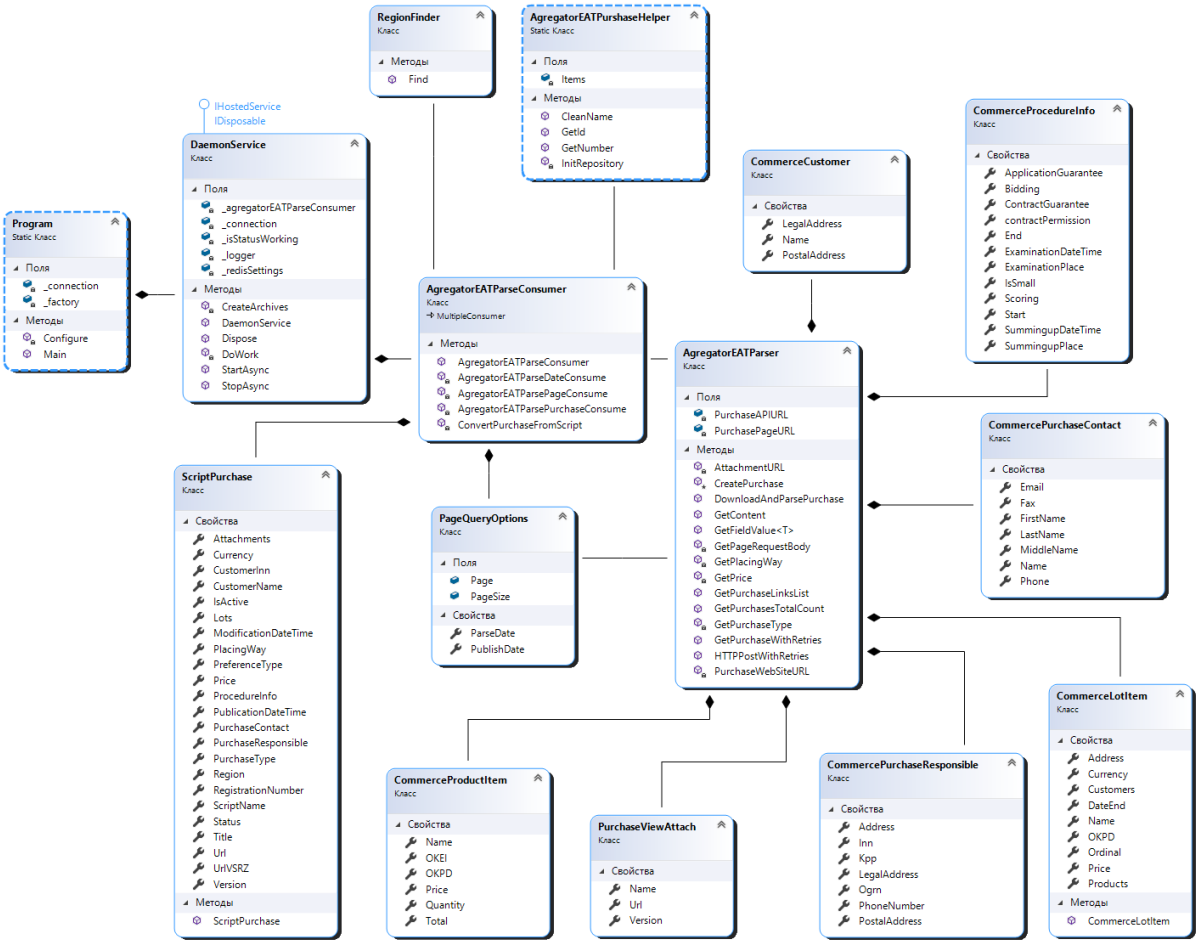


Рис. 34. Диаграмма классов сервисов пользовательской истории отражения тендеров электронной торговой площадки “Березка”.

Program - см. 4.1. Список клиентов из ЕРУЗ.

DaemonService - см. 4.1. Список клиентов из ЕРУЗ.

AgregatorEATParseConsumer - консьюмер сообщений, отвечает за сбор, преобразование и отправку информации о закупках с электронной торговой площадки“Березка” сервису-обработчику.

ScriptPurchase - модель заполненной закупки, отправляемая сервису-обработчику для записи в базу данных.

RegionFinder - вспомогательный класс, используемый для определения кода региона по адресу закупки.

PageQueryOptions - вспомогательный класс для передачи информации для запуска парсинга.

AgregatorEATParser - класс, содержащий в себе логику сбора информации о закупке с электронной торговой площадки “Березка”, ее обработки и генерации ScriptPurchase из нее.

CommerceCustomer - класс, содержащий модель данных заказчика закупок.

CommerceProcedureInfo - класс, содержащий модель данных информации о процедуре закупки.

CommercePurchaseContact - класс, содержащий модель данных контактной информации заказчика закупки.

CommerceLotItem - класс, содержащий модель данных лота закупки.

CommercePurchaseResponsible - класс, содержащий модель данных организатора закупок.

PurchaseViewAttach - класс, содержащий модель данных документации закупки.

CommerceProductItem - класс, содержащий модель данных продукта закупки.

Как результат, в разделе поиска отражаются закупки с электронной торговой площадки “Березка”.