ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Прикладная математика и информатика»

Отчет о программном проекте

на тему Доска для совместной удаленной работы
(промежуточный, этап 1)

Выполнил:

студент группы БПМИ213 _____ _____
Подпись И.О. Фамилия

студент группы БПМИ2110 _____ _____
Подпись И.О. Фамилия

студент группы БПМИ2110 _____ _____
Подпись И.О. Фамилия

студент группы БПМИ2110 _____ _____
Подпись И.О. Фамилия

15.02.2023
Дата

Принял:

руководитель проекта Нужненко Сергей Александрович
Имя, Отчество, Фамилия
Внештатный преподаватель (по ГПХ)
Должность, ученое звание
НИУ ВШЭ, Департамент программной инженерии
Место работы (Компания или подразделение НИУ ВШЭ)

Дата проверки 15.02.2023 _____
Подпись

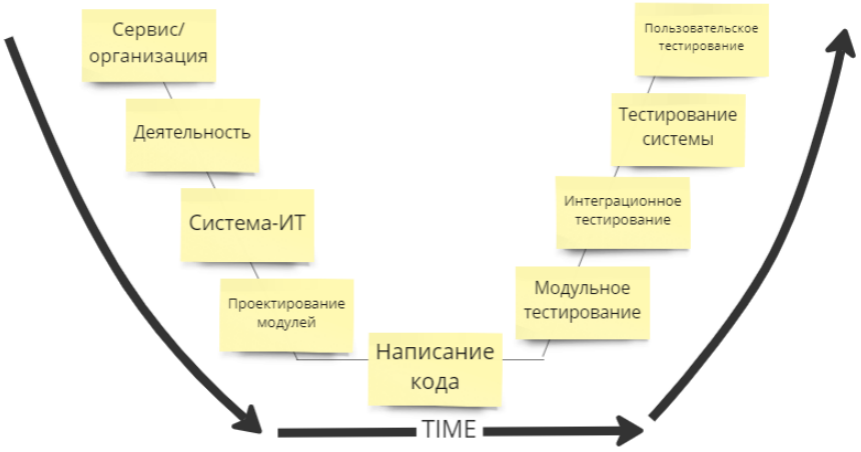
Москва 2023

Содержание

Введение

Краткое описание предметной области

“*Проектирование* – процесс составления описания, необходимого для создания в заданных условиях еще не существующего объекта, на основе первичного описания этого объекта и/или алгоритма его функционирования или алгоритма процесса, преобразованием (в ряде случаев неоднократным) первичного описания, оптимизацией заданных характеристик объекта и алгоритма его функционирования или алгоритма процесса устранением некорректности первичного описания и последовательным представлением (при необходимости) описаний на различных языках .” (ГОСТ 22487-77)

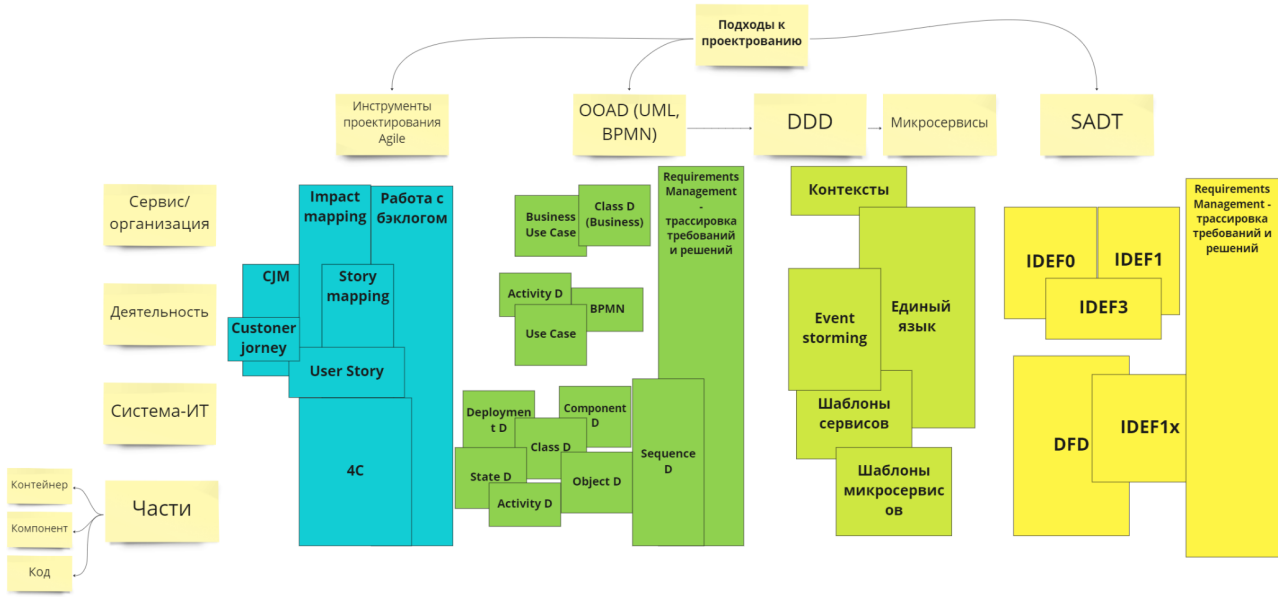


V-модель - модель, которая представляет собой метод описания процесса разработки программного обеспечения от этапов проектирования до полного формирования конечного продукта и направлена на упрощение этого процесса. Основной принцип заключается в постепенной детализации проекта при движении слева направо с течением времени. Рассмотрим процесс проектирования через левую ветку V-модели.

Описание левой ветки, которая отвечает за проектирование:

1. Сервис/организация. На первом этапе следует провести анализ требований к системе. Этот этап направлен на выявление свойств и функций идеальной системы.
2. Деятельность. Второй этап отвечает за анализ уже выставленных требований и оценивание методов и возможностей их реализации.
3. Система информационных технологий. Этот этап можно назвать высокоуровневым проектированием. Создается список модулей с их функциональностью, техническими деталями и зависимостями.
4. Проектирование модулей. Последний этап можно назвать низкоуровневым проектированием, где создается описание конкретных модулей, которое в итоге можно рассматривать как готовое техническое задание для программиста или командам, которым поручат этот модуль.

Предметной областью для нашего инструмента является проектирование.Рассмотрим различные подходы проектирования, для которых используются конкретные приложения автоматизации этого процесса. Во внимание возьмем 4 подхода: SADT, Agile, DDD, OOAD. Далее в диаграмме ниже последует разбиение техник и артефактов этих методологий на проектные решения различных уровней V-модели, отвечающих за проектирование.



Актуальность проблемы

На протяжении предыдущих десятилетий инструменты и подходы проектирования ИТ-систем и ПО постоянно развивались. Был период усложнения инструментов: в них было много различных функций и улучшений. Однако к единому подходу отрасль информационных технологий так и не пришла. На данный момент гибкие практики проектирования и инструменты испытывают откат к доске, фломастеру и стикерами, при этом утрачиваются многие функции, полезные для работы с требованиями и проектными решениями. По этой причине поиски идеального инструмента и метода проектирования следует продолжать.
Одним из представителей доски со стикерами является Miro - его можно считать образцом для подражания. Однако после объявления ухода Miro из России, мы теряем даже такое приложение. Отечественных инструментов проектирования даже предыдущих поколений и подходов в России или мало или нет вообще. Примерами таких являются Business Studio и Elma, однако они узкоспециальные и труднодоступные для среднего и малого бизнеса.

Цели проекта

1. Получение замены дистанционной доски для совместной работы
2. Исследование возможности получения системы проектирования следующего поколения

Задачи проекта

1. Исследовать инструменты проектирования
2. Исследовать возможность получения системы проектирования следующего поколения
3. Получить прототип доски, пригодный для дальнейших экспериментов
4. Определить требования к "инструменту проектирования мечты"

Обзор подходов к проектированию с рассмотрением инструментов для каждого

Object Oriented Analysis and Design

Про OOAD пишет Добрынин А.Э.

UML

UML — унифицированный язык моделирования — это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Используется для визуализации, спецификации, конструирования и документирования программных систем.

Плюсы:

- Возможность посмотреть на задачу с разных точек зрения
- Другим разработчикам легче понять суть задачи и способ ее реализации
- Диаграммы просты для чтения после быстрого ознакомления с их синтаксисом

Минусы:

- Трата времени на моделирование
- Необходимость знания различных диаграмм и их нотаций

Диаграмма вариантов использования — диаграмма, показывающая возможные варианты взаимодействия различных видов пользователей с системой. Участники – множество логически связанных ролей, исполняемых при взаимодействии с прецедентами или сущностями. Участником может быть человек, другая система, подсистема или класс. Прецедент — описание множества последовательных событий, выполняемых системой, которые приводят к наблюдаемому участником результату. Прецедент представляет поведение сущности, описывая взаимодействие между участниками и системой. Прецедент не показывает, каким образом достигается некоторый результат.

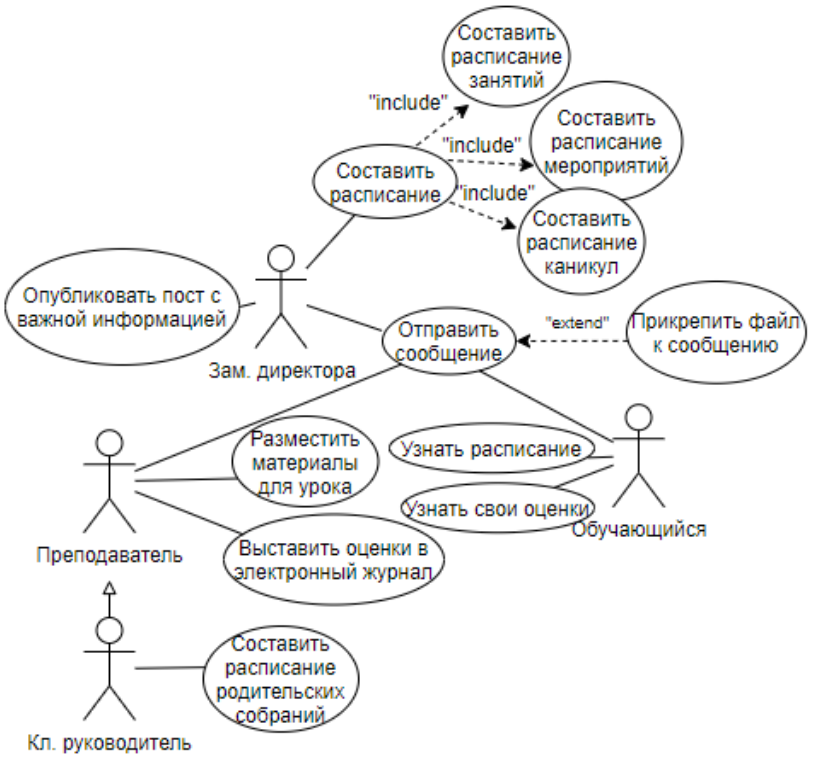


Диаграмма последовательности — диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл объекта и взаимодействие действующих лиц информационной системы в рамках сценария использования. Диаграммы последовательностей используются для уточнения диаграмм вариантов использования, более детального описания логики сценариев использования.

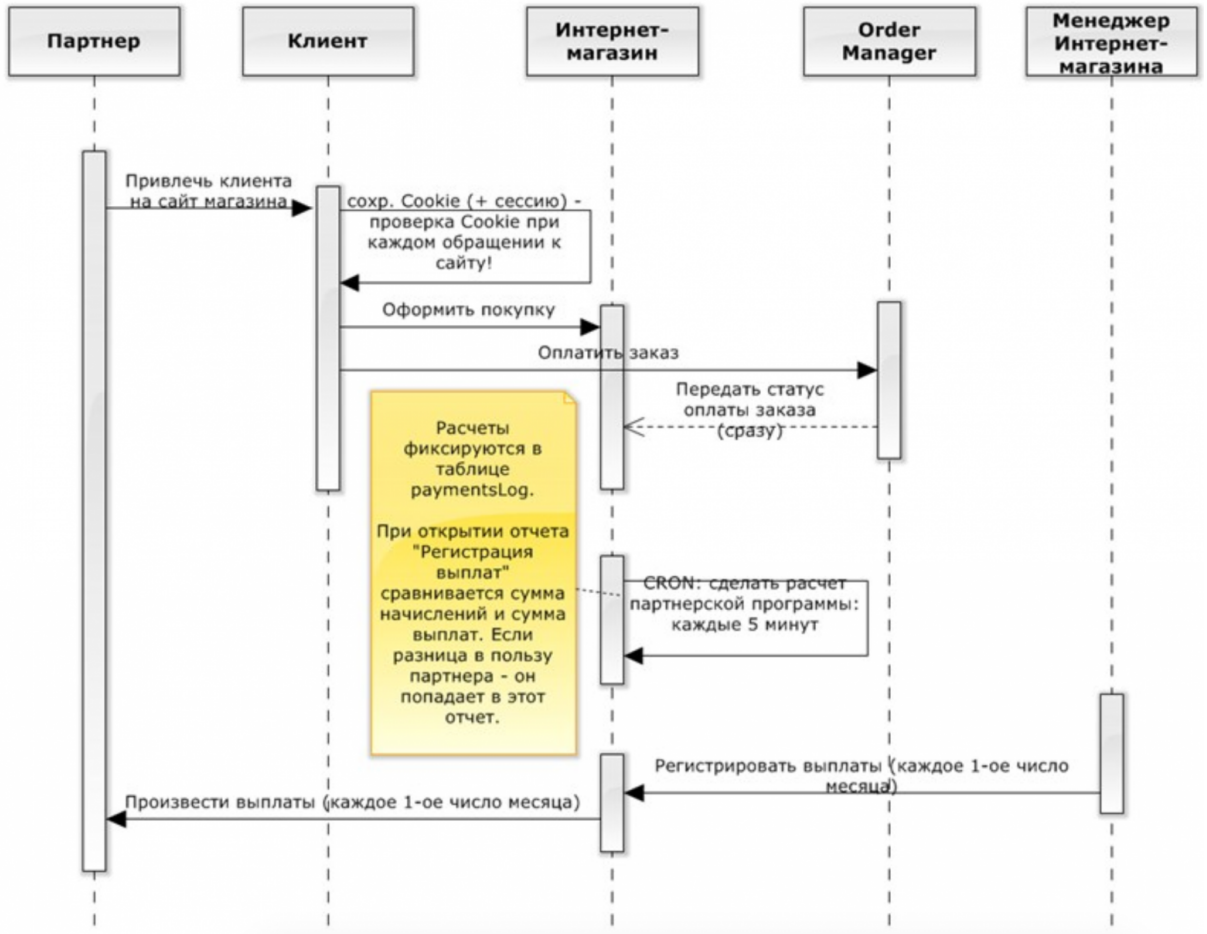


Диаграмма классов — наиболее распространенная разновидность диаграмм UML и фундаментальная база любого объектно-ориентированного решения. Отображает классы внутри системы, а также атрибуты, операции и отношения между классами. Диаграммы классов применяются при схематизации крупных систем, так как позволяют объединять классы в группы.

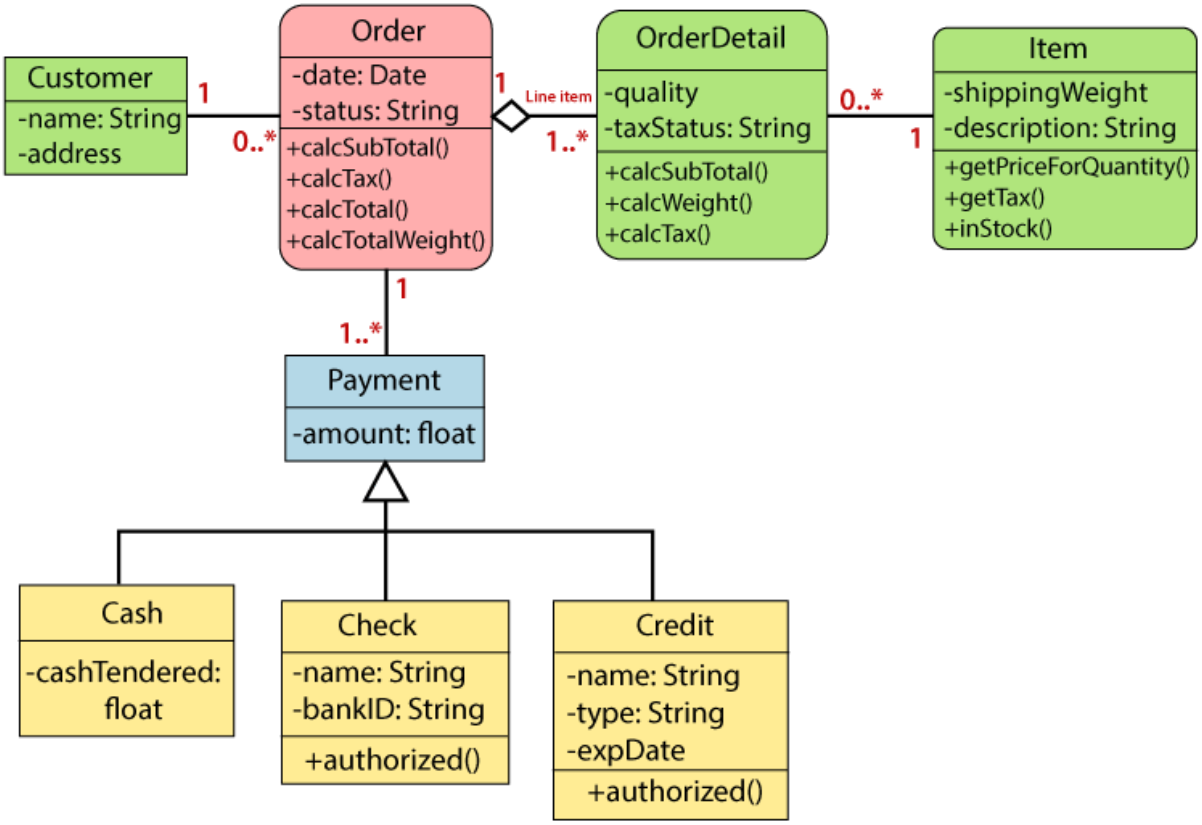


Диаграмма состояний — диаграмма, отражающая состояния и переходы между ними. На диаграмме отображаются события, которые влияют на состояние системы.
События могут иметь параметры, несущие дополнительную информацию. По событию также могут выполняться определенные действия во время смены состояния (например, изменение значения переменной). На диаграмме можно указывать условные выражения. От результата их вычисления будет зависеть то, в какое состояние будет следующий переход.

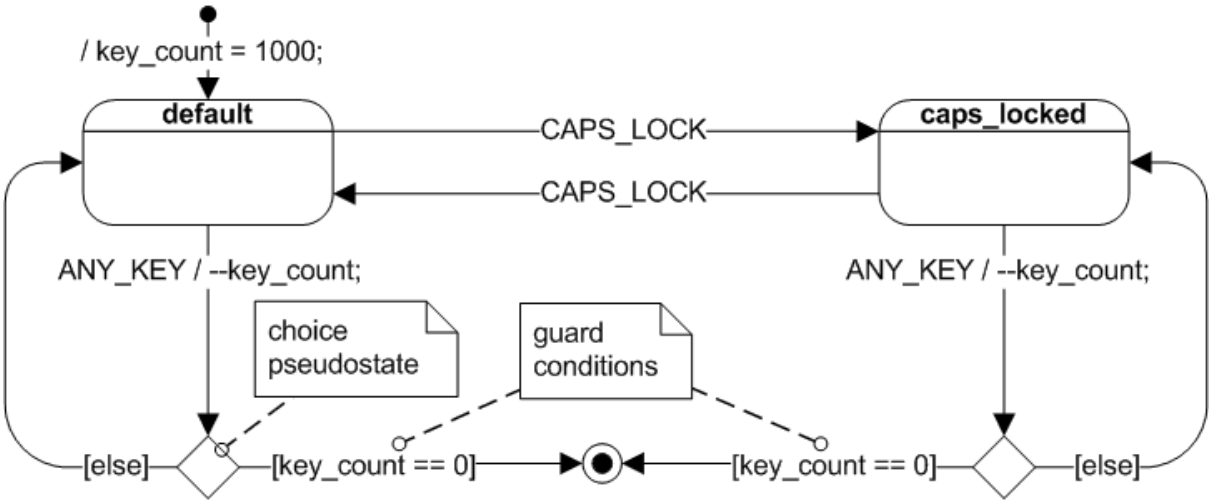


Диаграмма деятельности, как и диаграмма состояний, отражает динамические аспекты поведения системы. По существу, эта диаграмма представляет собой блок-схему, которая наглядно показывает, как поток управления переходит от одной деятельности к другой. Активности на диаграмме распределены по дорожкам, каждая из которых соответствует поведению одного из объектов. Благодаря этому легко определить, каким из объектов выполняется каждая из активностей.

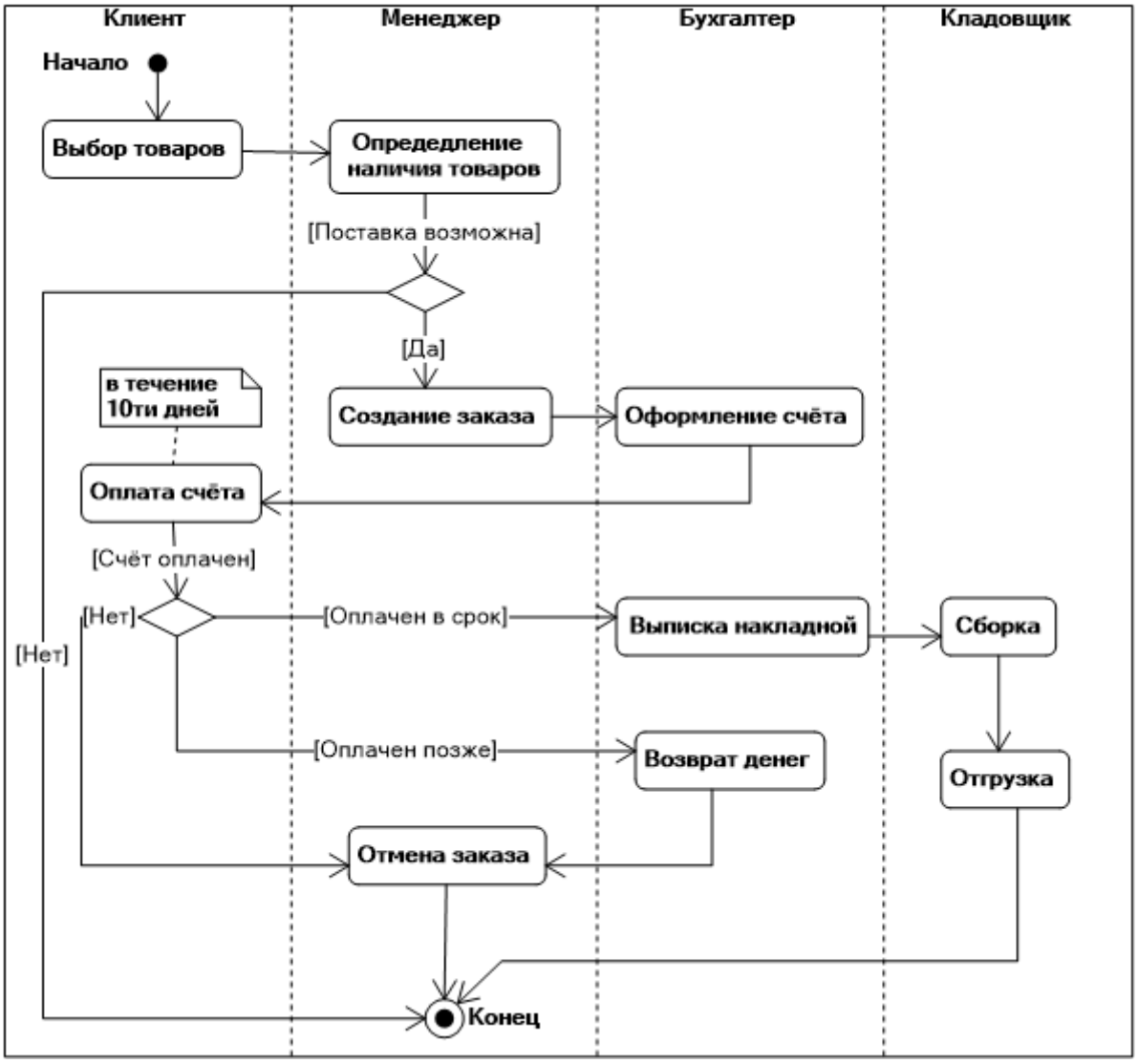


Диаграмма компонентов используется, чтобы показать разбиение программной системы на структурные компоненты и связи между компонентами. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и пр.

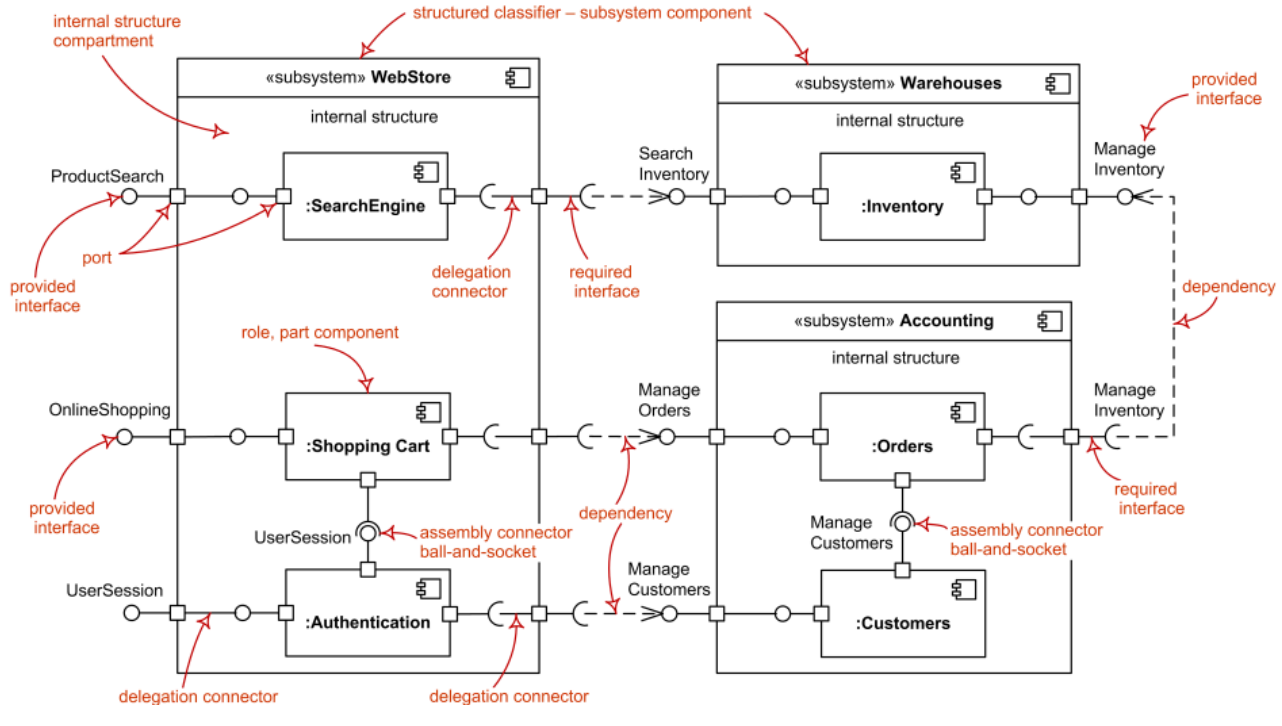
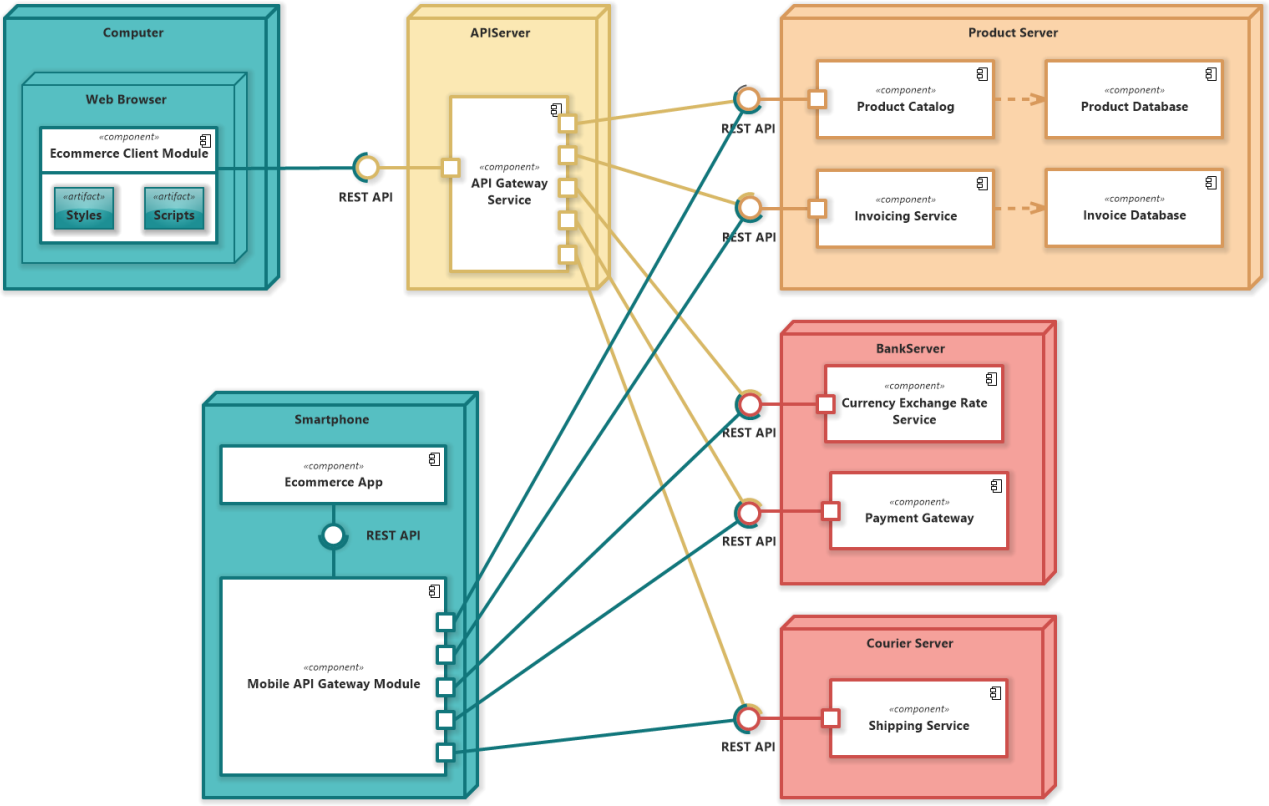


Диаграмма развертывания используется для графического представления инфраструктуры, на которую будет развернуто приложение. Корпоративные приложения часто требуют для своей работы некоторой ИТ-инфраструктуры, хранят информацию в базах данных, расположенных где-то на серверах компании, вызывают веб-сервисы, используют общие ресурсы и т. д. Всё это может быть отражено на диаграмме развертывания.

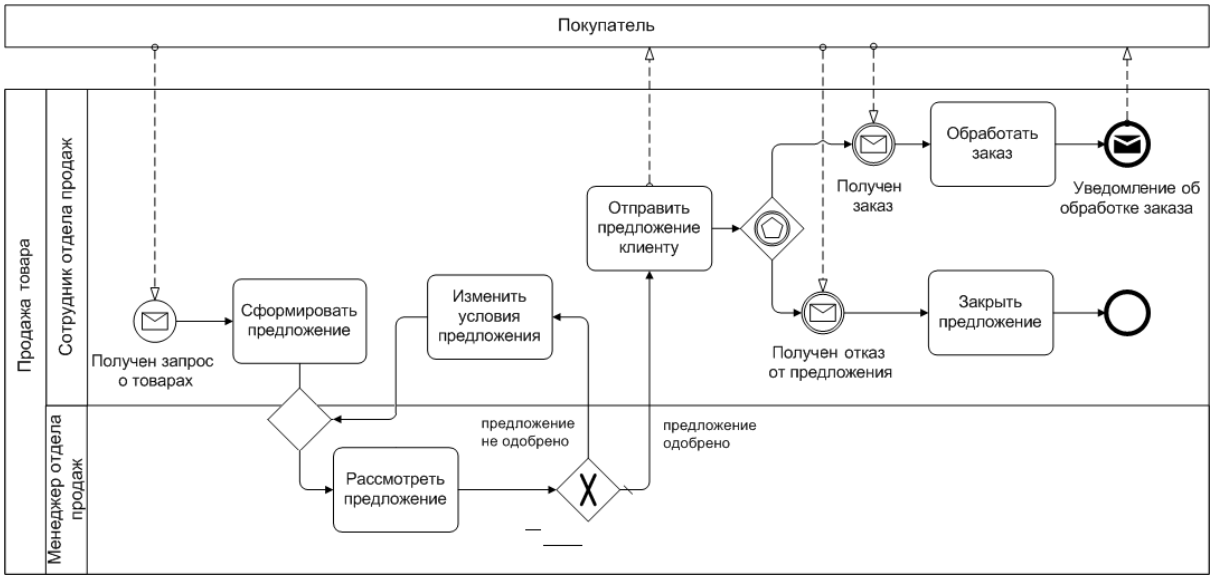


BPMN

Основная цель BPMN – предоставить условные обозначения, которые будут понятны всем участникам бизнеса, от аналитиков, создающих черновые варианты процессов, до разработчиков, реализующих эти процессы, и до тех, кто будет управлять этими процессами и анализировать их. BPMN является "стандартизированным мостиком" между дизайном бизнес-процесса и его реализацией.
Процессы, описанные с помощью BPMN, условно можно поделить на исполняемые и неисполняемые. Отличие в том, что исполняемые процессы могут быть исполнены с помощью ПО (Bizagi, Comundo). Неисполняемые процессы не требуют той строгости описания, которая нужна для выполнения исполняемых.

Плюсы от использования BPMN:

- Более легкое общение и сотрудничество для достижения цели
- Простое визуальное представление этапов
- Выявление проблем в процессах, которые могут нуждаться в решении
- Понимание потенциальных областей для улучшения



Краткое описание основных элементов BPMN

Events – какие-либо происшествия в мире. События инициируют действия или являются их результатами. Согласно расположению в процессе, события могут быть классифицированы на начальные, промежуточные и завершающие.

Activities – задачи, которые необходимо выполнить на определенном этапе бизнес-процесса. Действия могут быть элементарными, т.е. неделимыми на некоторые более простые действия, и не элементарными, т. е. такими, которые при детализации распадаются на последовательность определенных более простых действий.

Gateway – управляющий узел, который появляется при условном разветвлении бизнес-процесса. Шлюзы нужны в тех случаях, когда процедура зависит от определенных факторов. Например, при работе с покупателями шлюз появляется на этапе, когда клиент принимает решение о покупке — «да или нет». При положительном решении необходимо совершить покупку, при отрицательном - выяснить возможные причины отказа, работать с "отказом" и т.д.

Pool – это объект, описывающий один процесс на диаграмме. На одной диаграмме может быть несколько пулов. Пул может быть расширен для просмотра деталей. Пул также может содержать так называемые «дорожки». Они нужны для того, чтобы указать участников процессов, которые скрыты в пуле.

Data – это элемент, указывающий, какие данные и документы необходимы для начала действия или каковы результаты завершенного действия.

Artefact – под артефактами в BPMN понимаются объекты, не являющиеся действиями и не имеющие прямого отношения к действиям. Это могут быть любые документы, данные, информация, не влияющие непосредственно на выполнение процесса.

Инструменты, подходящие для использования OOAD

- **SparX Enterprise Architect**
  - Поддержка UML и BPMN диаграмм
  - Матрица трассировки
  - Дерево связей для навигации по связям
  - Генерация кода и документации по диаграмме классов

    - Отсутствие возможности совместной работы
    - Сложность интерфейса программы
- **PlantUML**
    - Поддержка UML-диаграмм
    - Хорошо подходит для контроля изменений, т.к. диаграммы генерируются из текстового формата
    - Есть дополнения для различных сред разработки и редакторов кода (VSCode, Atom, IntellijIDEA)
    - С готовой диаграммой можно работать лишь как с изображением
    - Нет поддержки совместной работы
    - Необходимость знания языка для описаний диаграмм
- **Miro**
    - В платных планах есть поддержка небольшого кол-ва элементов, которые используются в диаграммах, но их можно эмулировать самому
    - Нет возможности посмотреть связи и карточки в виде списка
    - Возможность совместной работы
- **Camunda**
    - Поддержка BPMN
    - Средство для поиска ошибок и узких мест в бизнес-процессах
    - Веб-приложение, в котором исполнители выполняют задачи, поставленные на них бизнес-процессом
    - Веб-приложение для просмотра состояния процессов
- **ELMA 365**
    - Моделирование и исполнение бизнес-процессов (Поддержка BPMN)
    - Графический конструктор процессов и интерфейсов
    - Инструменты по управлению документооборотом
    - API для использования во внешних программных продуктах
    - Корпоративный мессенджер для совместной работы: лента, каналы и чаты, обмен файлами и постановка задач

Agile

Про AGILE пишет Шарипов С.У.
AGILE - В разработке программного обеспечения включает в себя выявления требований и улучшение продукта за счет совместных усилий разработчиков, бизнес-технологов с их конечными пользователями, адаптивное планирование, ранняя эксплуатация ,постоянное совершенствование и гибкое реагирование на изменения требований.
Плюсы от использования AGILE:

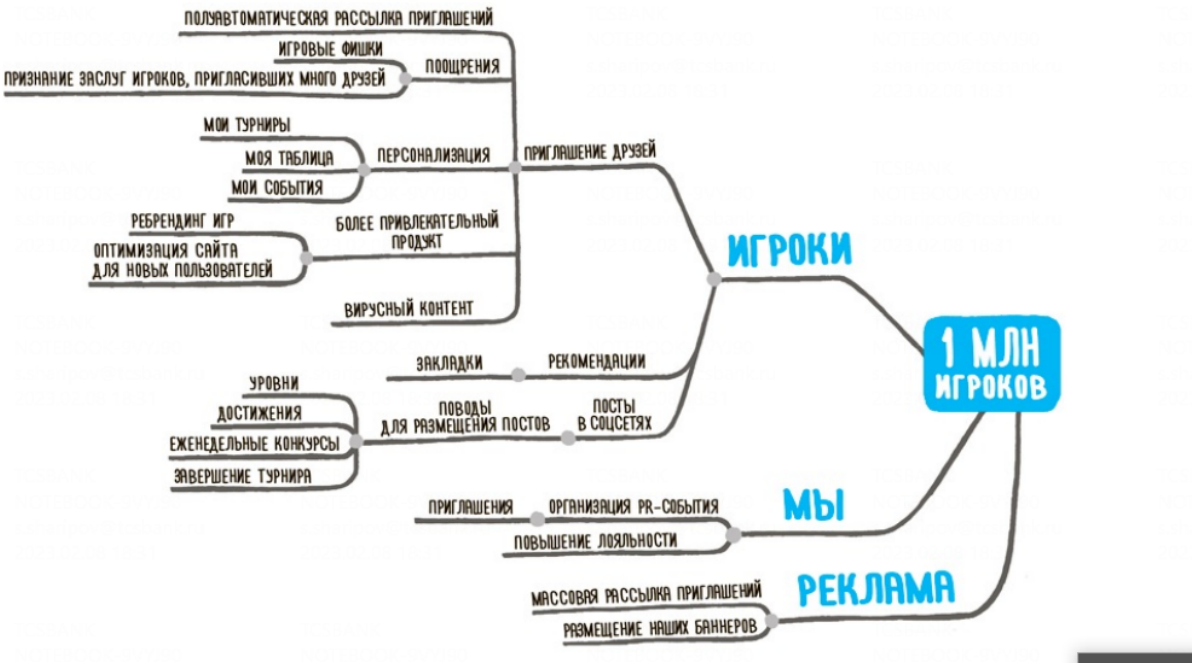
- Из-за того что на каждой итерации проектируется минимальный значимый продукт, то команда может с легкостью получить обратную связь от пользователей и если нужно изменить будущие планы, без потери значительных ресурсов
- Прозрачность
- Вся команда работает, т.е. в каждой итерации найдется задача для всех (разработчики, тестировщики, дизайнеры)

Минусы от использования AGILE:

- Только маленькие команды могут использовать AGILE методологию
- Отсутствие четкого плана разработки
- Риск получить продукт низкого качества

Impact mapping - это составление mind map`ов, которые позволяют визуализировать границы проекта и связывать бизнес процессы с конкретными реализациями.
Правила impact mapping - нужно ответить на 4 вопроса:

1. Зачем? Цель проекта, чего хотим достичь сделав этот проект
2. Кому? Действующие лица - кому мы делаем этот проект, кто будет пользоваться проектом.
3. Как? Как будет воздействовать наш проект действующим лицам и как они могут воздействовать на проект
4. Что сделаем? Что нужно сделать чтобы добится такого проекта



На этом рисунке показан impact mapping онлайн-игры казино :

1. Цель - Привлечь 1 млн онлайн-клиентов

2. Действующие лица - это игроки, реклама и мы
3. Как достичь цель - игроки могут приглашать друзей, мы можем организовать PR-события, а с помощью рекламы массово рассылать приглашения и многое друго
4. Что проект должен иметь чтобы лица могли так вести -
   Для того чтобы игроки хотели приглашать своих друзей, нужно заинтересовать действующих игроков вирусным контентом или же поощрением за 1 го приглашенного друга

Плюсы от использования Impact Mapping:

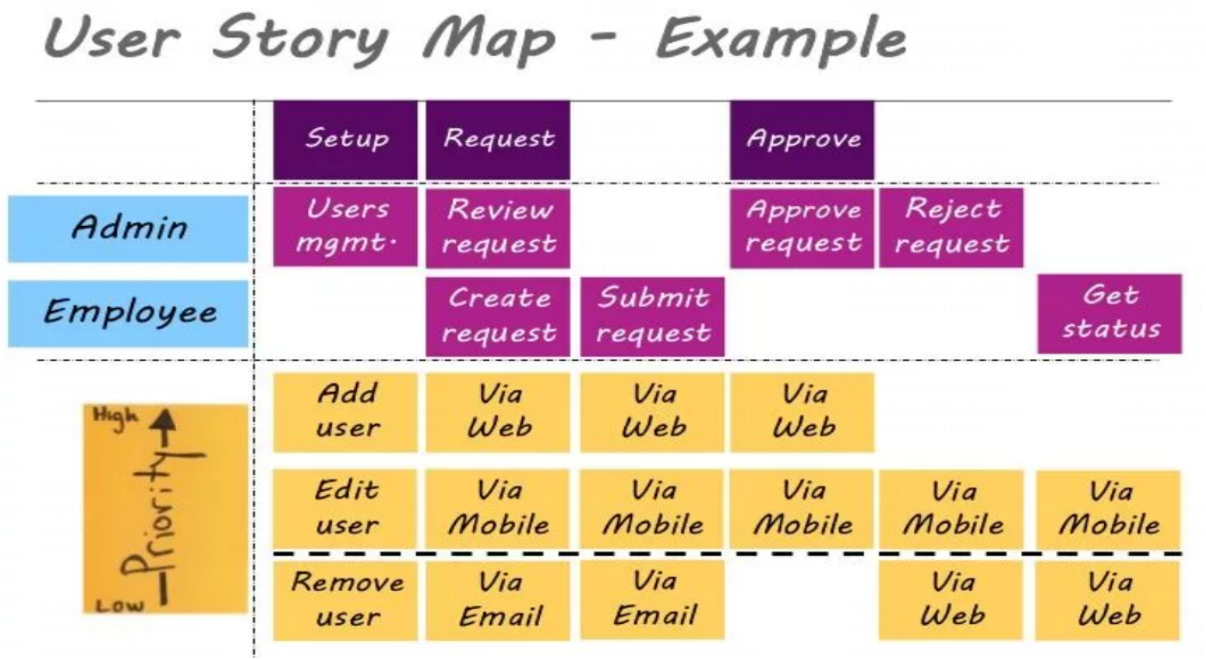
- Легко дорабатывать
- Понимание процессов между бизнесом и разработчиками

Минусы от использования Impact Mapping:

- Короткое время жизни, нужно постоянно актуализировать
- Нет четкого понимания сроков проекта

User story - это короткая формулировка сценария, описывающая что-то, что система должна делать для пользователя.
Story Mapping (пользовательские истории) - это метод, который позволяет дорабатывать или выявлять новые функционалы для усовершенствование продукта Пользовательские истории выделяются в три этапа.

1. Дебют. Базовые сценарии пользователя, которые должен поддерживать проект.
2. Миттельшпиль. Дополняет и прорабатывает функционалы которые есть в дебют которые облегчают работу пользователя
3. Эндшпиль. Наводит блеск проекту, обогащает функционалом.



На этой картине впечатлен пример User Story Map на основе обычного клиент-серверного приложения.

1. Голубые карточки - это типы пользователей
2. Бордовые карточки - это базовое представление приложения
3. Розовые карточки - это то, какие сценарии есть для конкретного пользователя с базового представления

4. Желтые карточки - это функционалы которые должен уметь приложения для поддержания сценария пользователя в приоритетном порядке
5. Пунктирное разделение - это релизы приложения

Плюсы от использования User story mapping:

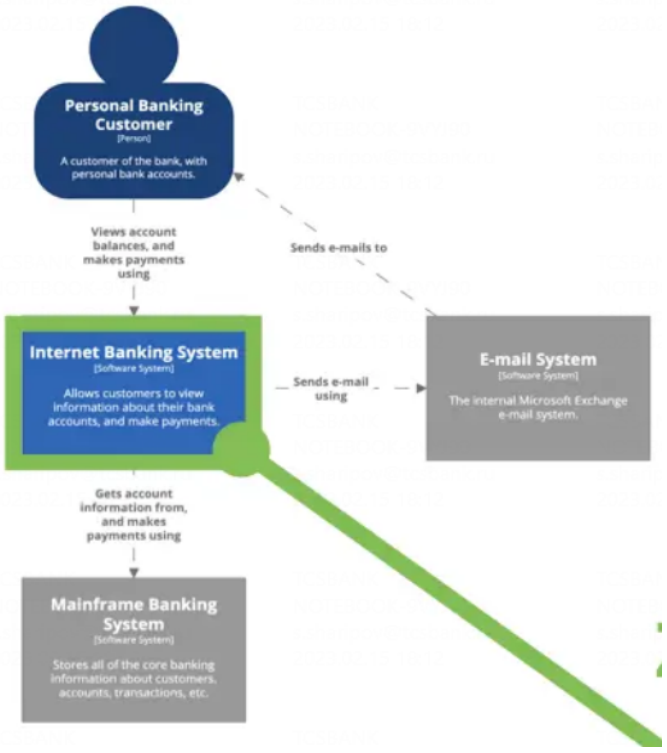
- Прозрачность в смысле наглядности и приоритезации
- Основной фокус на пользователя

Минусы от использования User story mapping:

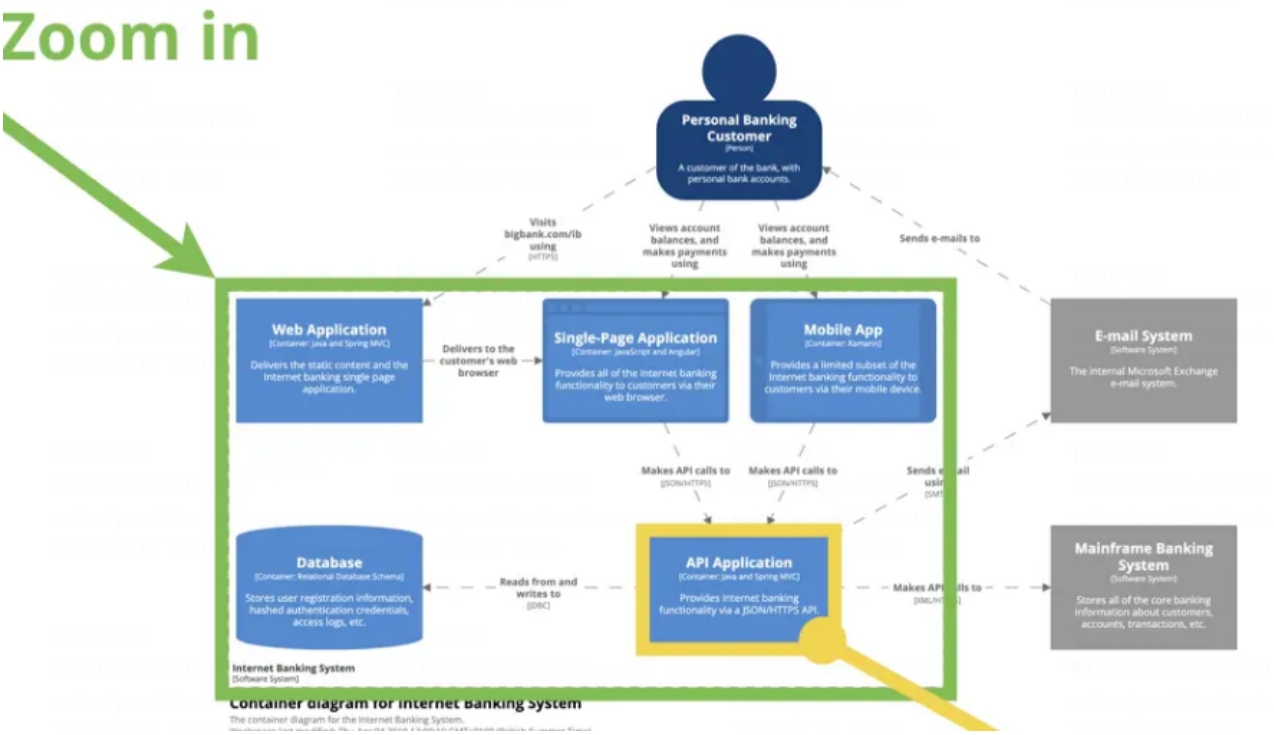
- Нет детальных требований к ресурсу. Повествование делается в общих чертах

4C (context-container-component-code) - метод, который позволяет декомпозировать проект отражая его с разных точек зрения.
Уровни декомпозиции:

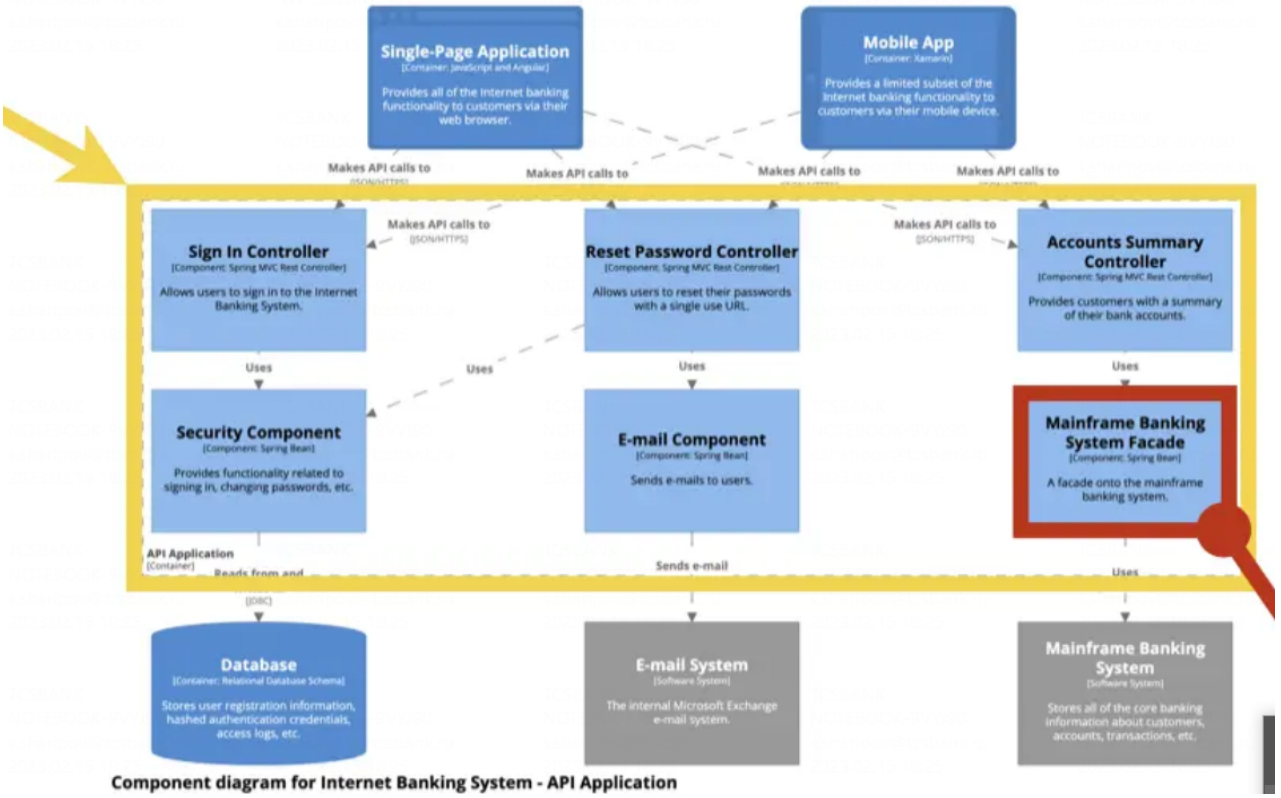
1. Диаграмма **контекста системы** обеспечивает отправную точку, показывая, как программная система по объему вписывается в окружающий мир.
2. схема **контейнера** увеличивает масштаб программной системы, показывая высокоуровневые технические строительные блоки.
3. Диаграмма **компонентов** увеличивает масштаб отдельного контейнера, показывая компоненты внутри него.
4. диаграмма **кода** (например, класса UML) может использоваться для увеличения масштаба отдельного компонента, показывая, как этот компонент реализован.



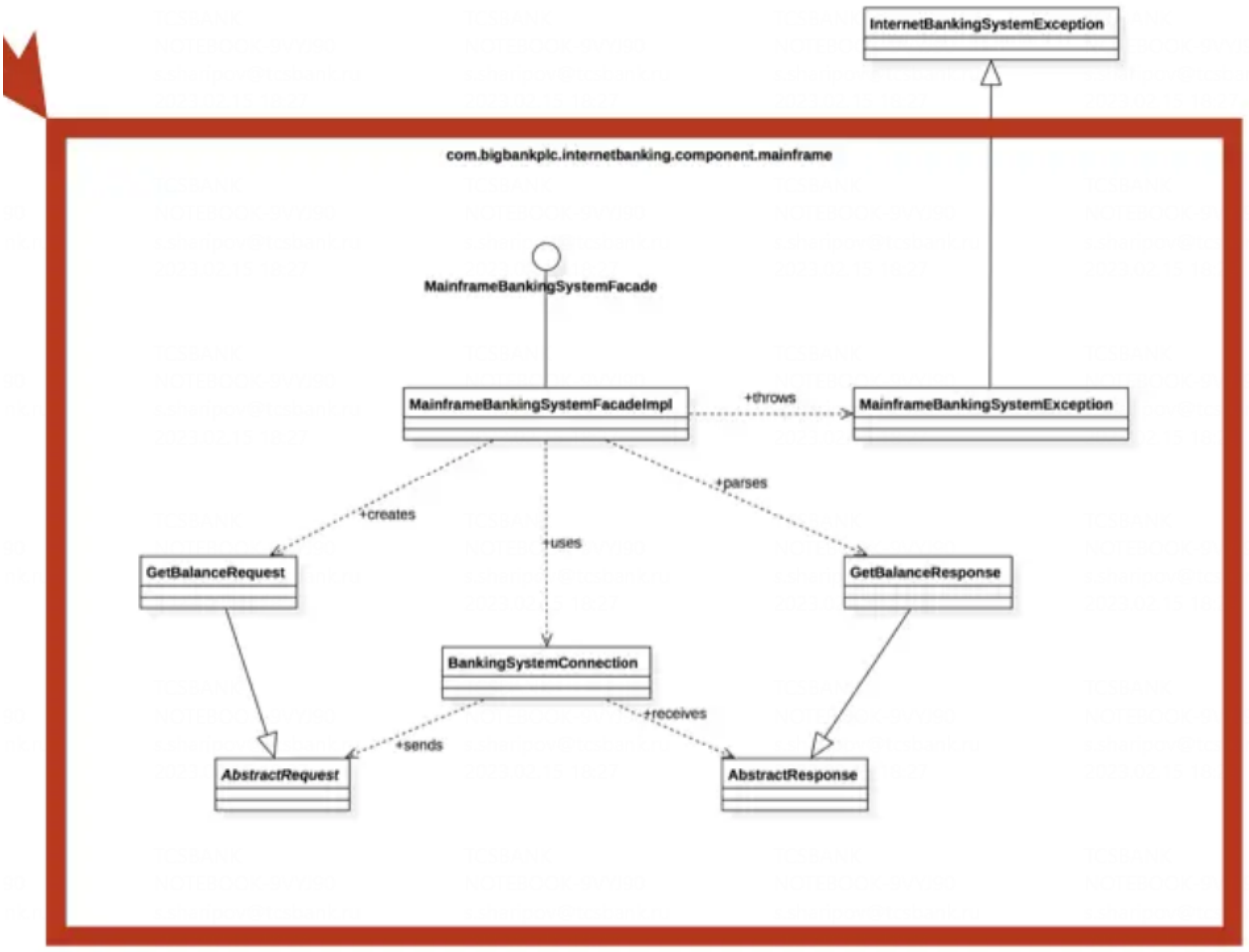
На рисунке показан самый верхний уровень детализации - **контекст системы**
Здесь Банковская система (зеленый прямоугольник) показано в контексты системы. Далее приблизим и контейнерную детализацию



Тут мы разложили банковскую систему на контейнеры (синие фигуры внутри зеленого прямоугольника). Следующий уровень детализации - это компонентная детализация, для примера рассмотрим контейнер - API приложение (желтый квадрат)



Все что в желтом - это компонентная диаграмма контейнера API-приложение



Далее как последний уровень детализации рассмотрим компоненту “Главное меню приложения” и его диаграмму кода

CJ (Customer Journey) - это путь, который проходит клиент: от возникновения потребности в товаре до момента покупки или превращения в фаната бренда.

CJM (Customer Journey Mapping) - это визуализация пути покупателя. Она отображает этапы, которые проходит клиент и какие эмоции при этом испытывает, точки взаимодействия с брендом и сложности, которые не позволяют ему достигать своих целей.

Пример CJM:

1. Триггер - причина почему произошел такой сценарий
2. Голубые карточки показывает что клиент собирается делать
3. Темно желтой карточки показывает - что клиент делает на конкретном этапе
4. В рядах со смайликами показывается то что клиент чувствует на данном этапе.

Так благодаря CJM. мы узнаем точки этапы - где меняются эмоции клиента и найти способы улучшения клиентского опыта.

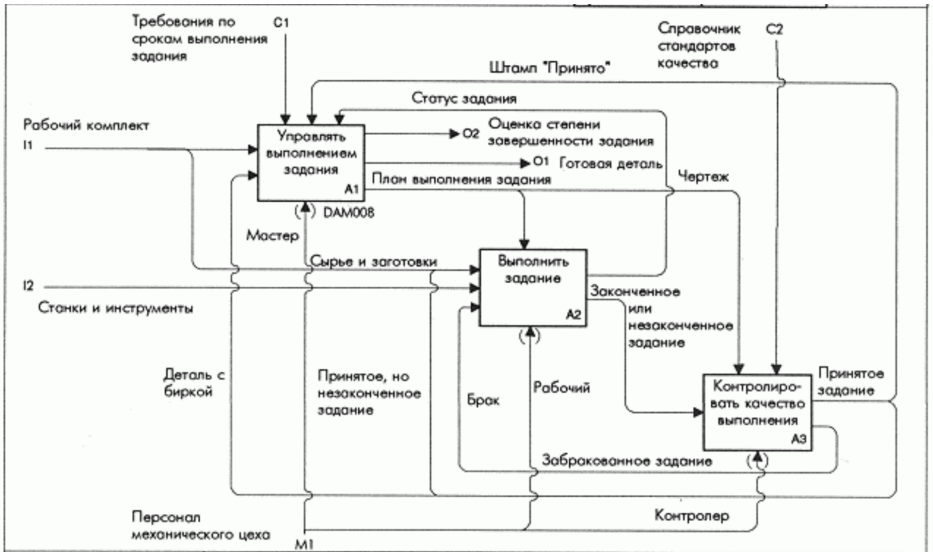
Инструменты, подходящие для использования AGILE

- **SparX Enterprise Architect**
  - Управление бэклогом
  - Иерархические вложенные списки
  - Карточки
  - Настройка планов и задач
- **MIRO**
  - Поддерживает все подходы AGILE
- **Notion**
  - Управление бэклогом
  - Настройка планов и задач
  - Классификация по карточкам
  - Карточки
- **Trello**
  - Управление бэклогом
  - Настройка планов и задач
  - Карточки
- **Google Sheets**
  - Настройка планов и задач
  - С трудом карточки в виде таблиц
- **Confluence**
  - Управление бэклогом
  - Настройка планов и задач
  - Карточки

SADT

Про SADT пишет Туретаева Амина

IDEF0 - Графическая нотация, предназначенная для формализации и описания бизнес-процессов. Модель в этой нотации является некоторым толкованием определенной системы, что значит субъектом моделирования только одна система. У каждой модели только одна цель и точка зрения, где целью является получение ответов на некоторую совокупность вопросов с заданной степенью точности, а точка зрения это позиция, с которой описывается система.



Диаграммы в модели состоят из блоков, описывающих определенное действие (функцию), и двунаправленных стрелок, связывающих блоки и отображающих их взаимодействие посредством объектов. Каждый блок можно детализировать в диаграмме-потомке, чем ниже уровень - тем больше деталей. Единственный блок в контекстной диаграмме обозначает границу системы: все, лежащее внутри него, является частью описываемой системы, а все, лежащее вне него, образует среду системы. На диаграммах нижнего уровня 3-6 блоков

Виды стрелок:

- I - Input - Вход - объекты потребляемое и преобразуемое функцией.
- C - Control - Управление - информация, которая управляет действиями функций
- O - Output - Выход - результат функции, часто является преобразованием вводных объектов
- M - Mechanism - Механизм - объекты используемые, но не потребляемые функцией

IDEF1 - методология моделирования информационных потоков внутри системы, позволяющая отображать и анализировать их структуру и взаимосвязи.

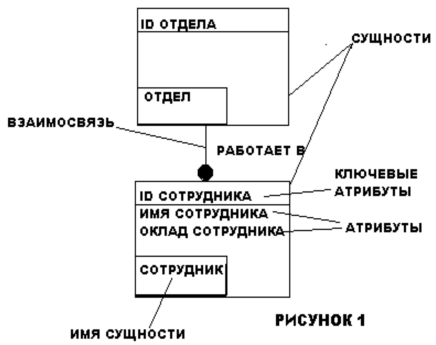


РИСУНОК 1

Блоки в этой нотации представляют сущность, которая соответствует конкретному объекту (или группе) и описывает совокупность информации. У сущностей есть атрибуты, являющиеся их свойствами и признаками. Атрибут состоит из названия атрибута и его значения для этой сущности. Атрибуты с уникальными для каждой сущности значениями называются ключевыми атрибутами.

IDEF1X (IDEF1 extended) - методология позволяющая создавать семантические модели данных, которые могут служить для поддержки управления данными как ресурсами, интеграции информационных систем и построения компьютерных баз данных.

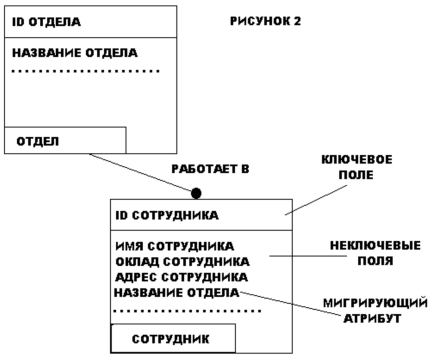
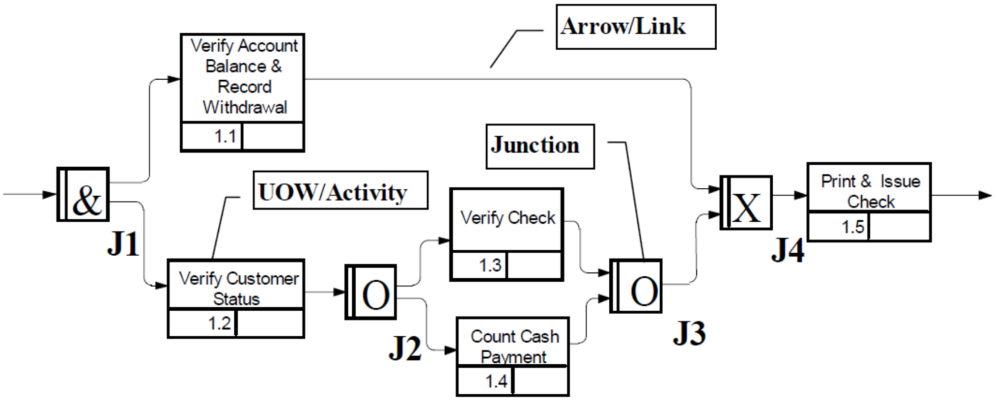


РИСУНОК 2

Блоки в это нотации так же описывают сущности, однако в отличии от IDEF1, тут они описывают конкретный набор объектов, не включающий в себя абстрактный набор информационных отображений реального мира. Первичным ключом называется атрибут или набор атрибутов с уникальными для определенной сущности значениями, по которым можно идентифицировать сущность. Внешние ключи - переданные через связь первичные ключи родительского объекта. Атрибуты мигрируют, если передаются через связь между объектами. Если уникальность дочерней сущности зависит от внешнего ключа, то она зовется зависимой сущностью. В нотации IDEF1X зависимые сущности часто представлены в виде закругленных прямоугольников.

IDEF3 - стандарт для документирования технологических процессов, происходящих на предприятии, который позволяет наглядно изобразить их сценарии.



Issue Cashier's Check

Диаграммы в этой модели состоят из блоков, односторонних стрелок, перекрестков стрелок и объектов ссылок. Блоки представляют собой функции (Unit of Work/activity), перекрестки отображают логику слияния/разветвления стрелок, объект ссылки включает в себя идеи/данные/концепции не покрываемые остальными объектами нотации.
Виды стрелок:

- Старшая - Temporal Precedence - временное предшествие между UOW и UOW. Блок-источник сначала должен завершиться для выполнения блока-цели;
- Стрелка отношений - Relational - нечеткое отношение между двумя блоками, а также между блоками и объектами ссылок (пунктирная стрелка);

- Поток объектов - Object Flow - поток объектов между блоками, пример: объект является выводом одной функции и используется в другой (стрелка с двойным наконечником)

Объектами ссылки являются: объект (Object), заслуживающий отдельного внимания; ссылка (GOTO) для реализации циклов; единица действий (UOB) позволяющая многократно отображать одно и то же действие без цикла; заметка (Note) для документирования важной информации общего характера, уточнение (ELAB) для подробного описания диаграммы.

DFD - Иерархия диаграмм потоковых данных, описывающих асинхронный процесс преобразования информации от ее входа в систему до выдачи пользователю. Диаграммы состоят из внешних сущностей, процессов, хранилищ данных и потоков данных. Внешние сущности представляют объекты или физические лица, которые являются источниками или приемниками информации, к тому же сущность может быть использована многократно. Потоки данных определяют поток объектов между частями системы. Процессы преобразуют входные данные из потоков, а хранилища данных являются устройствами хранения объектов.

Инструменты для реализации SADT

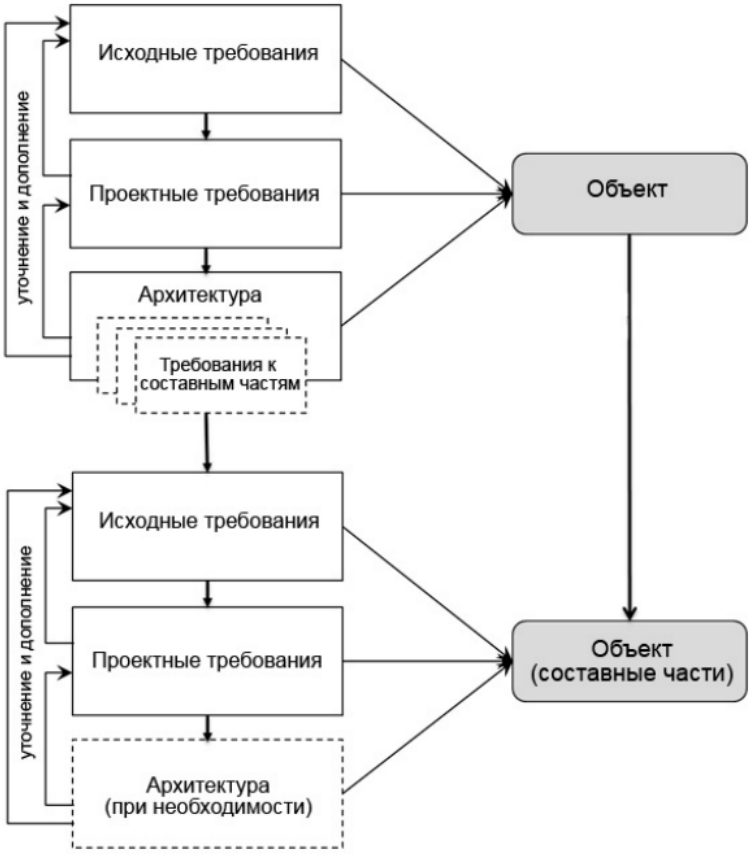
- SparX Enterprise Architect
  - ER-диаграммы
  - DFD
  - Работы с реляционными базами данных.
- Miro
  - Эмуляция диаграмм и связей
- Draw IO и Confluence
  - Диаграммы (с трудом)

Инженерия Требований

Инженерия Требований (Requirements Engineering) - подраздел системной инженерии , занятый выявлением, разработкой, прослеживанием, анализом, проверкой соответствия, установлением взаимосвязей и управлением требованиями, которые определяют систему на последовательных уровнях абстракции.

- Выявление требований - процесс ревью, формулировки и понимания потребностей клиента и ограничений программного обеспечения.
- Анализ требований - анализ потребностей клиента для определения требований
- Уточнение требований - процесс разработки документа (спецификация требований к ПО) в котором четко излагаются требования

- Проверка требований - процесс проверки соответствия требований и спецификации требований к программному обеспечению потребностям клиентов и системы.
- Управление требованиями - планирование и контроль процессов Инженерии Требований.



В результате анализа получают исходные требования: потребности заинтересованных сторон и требования к системе конкретного типа обусловленные документами по стандартизации. На основе исходных требований разрабатываются проектные требования, состоящие из описания функции системы, технических характеристик, описание принципов и режимов работы, условий эксплуатации, характеристики взаимодействия с внешними объектами (интерфейсы) и т.д.. В случае сложных систем разрабатывается архитектура в ходе проектирования, и разрабатываются требования для составных частей системы.

Инструменты для Инженерии Требований

- SparX Enterprise Architect
  - Работа со списками
  - Классификация элементов
  - Трассировочные таблицы
  - API для управления и доступа к информации
  - Хороший импорт/экспорт списков и связей
  - Авторасклaдка элементов со связями на диаграмме
- PlantUML
  - Контроль изменений в GIT
  - Генерация представления кода в различных видах, в том числе как диаграмма
- Miro
  - Ограниченная работа со списками
  - Расширенные функции упорядочивания и перехода от конвергенции к дивергенции
- Notion, Trello, Google Sheets
  - Списки

    - Связи
- Confluence
    - Списки
    - Связи
    - Контроль изменений в GIT
    - Создание документации
    - Трассировки

Предметно-ориентированное проектирование (*domain-driven design*, DDD)

Про DDD пишет Лях Л.А.

DDD – подход к проектированию, направленный на создание высококачественной модели ПО, которая максимально точно соответствует поставленному в основание бизнес-процессу. Подход полезен для правильной и корректной организации структуры модели, когда программист не является специалистом в предметной области поставленной задачи.

Основные направления, которые содержит данный подход:

- **Единый Язык (Ubiquitous Language)**

Единый язык можно представить как совокупность фраз, терминов ми различных понятий, которые использует команда при создании модели. Концепция единый язык помогает найти договоренность между экспертами в предметной области и программистами и привносит более глубокое понимание процессов всем участникам команды. Важно отметить, что DDD стремится четкому следованию архитектуры ПО за строением предметной области. Это несет за собой два полезных свойства: минимальное сопротивление архитектуры перед изменениями; в устоявшихся предметных областях сразу получается более качественная модель. Что стоит за этими двумя свойствами? Мы избегаем избыточных обобщений и привнесенных жестких связей там, где их нет в предметной области. К примеру, можно привести жесткую связь один заказ - один человек в приложении такси. Однако в реальной жизни возникает необходимость заказывать такси не только себе, но и другим людям, что противоречит нашей поставленной связи. Решением такой ситуации является вынесение подобной связи в бизнес-правило, с возможностью его изменить.

Существует множество методов, способных помочь и упростить процесс создания модели. Самым ярким и наиболее важным примером таких методов является Событийный штурм (Event storming).

Так как единый язык можно представить в виде глоссария, то в качестве инструментов, предоставляющих такую возможность хранения отдельных слов и словосочетаний, можно привести Notion и Trello.

- **Стратегическое моделирование**

Ограниченный контекст является одним из шаблонов в этом направлении и представляет собой некую границу, в которой существует модель, с отображением единого языка в ней. Таким образом в любом контексте каждая модель однозначно определена и имеет свои свойства. Например, для описания какой-либо работы организации модуль сотрудника, которая в разных контекстах его работы содержит свои атрибуты.

Работа над задачей в совокупности намного сложнее, чем работа с той же задачей, но разбитой на несколько частей. Таким образом, стратегическое моделирование помогает разбить предметную область на подобласти, ссылаясь на функциональность каждой, с учетом ограниченного контекста. Такое разбиение позволяет выделить смысловое ядро, служебные подобласти, неспециализированные подобласти, правильно распределить специалистов и бюджет на создание каждой.

Рассмотрим подробнее эти подобласти:

- Смысловое ядро - подобласть, которая выделяет весь бизнес и имеет первостепенное значение для организации. Если мыслить стратегически, то основные ресурсы стоит направить именно сюда.
- Служебная подобласть - подобласть, которые являются второстепенными внутренними вещами, которые также важны, но не являются смыслом создания модели предметной области, например формирование заказа в приложении.
- Неспециализированные области - подобласти, без специального назначения, но требуемые для существования. Такие области обычно отдаются на аутсорс(например, оплата через какую-нибудь систему).

· **Тактическое моделирование**

Предоставляет шаблоны программистам как строительные блоки, которые просты в тестировании, помогают допускать меньше ошибок, и доступны в понимании бизнесу. Такие шаблоны имеют технический характер и используются сугубо в реализации решения задач ограниченного контекста. Примеры таких шаблонов: сущности, объекты-значения, службы, события, агрегаты и т.д. Каждый шаблон используется в конкретной ситуации и имеет свои характеристики.Рассмотрим подробнее:

- Сущность - объект, модель которого уникальна и дает возможность однозначно идентифицировать. Например, счет в банке, конкретный человек в системе.
- Объект-значение - чаще неизменяемый объект, который полностью определяется своими атрибутами. Например, {50 долларов}, где атрибуты это число и валюта.
- Служба предметной области - операция, которая выполняется над объектами, но не может быть приписана как метод к одному из них.
- Событие - работает в связке с шаблоном Наблюдатель, который публикует произошедшие действия и передает этот факт в данный шаблон для обработки.
- Модуль - контейнер с определенным именем, содержащий некоторые, тесно связанные между собой объекты, с целью ослабления связи между классами.
- Агрегат - кластер объектов, которые рассматриваются как единое целое. Обращение к происходит через корень(сущность с уникальным идентификатором).
- Фабрика - объект, целью существования которого является создание сложных объектов, таких как агрегаты и сущности, которые почти невозможно задать через конструктор.
- Хранилища - область памяти, которая нужна для безопасного хранения содержащихся в ней элементов, в основном используется для агрегатов.

Плюсы использования:

· Упрощение реализации сложных проектов

· Низкая стоимость дальнейшей разработки в связи с поддержанием модульности

Минусы использования:

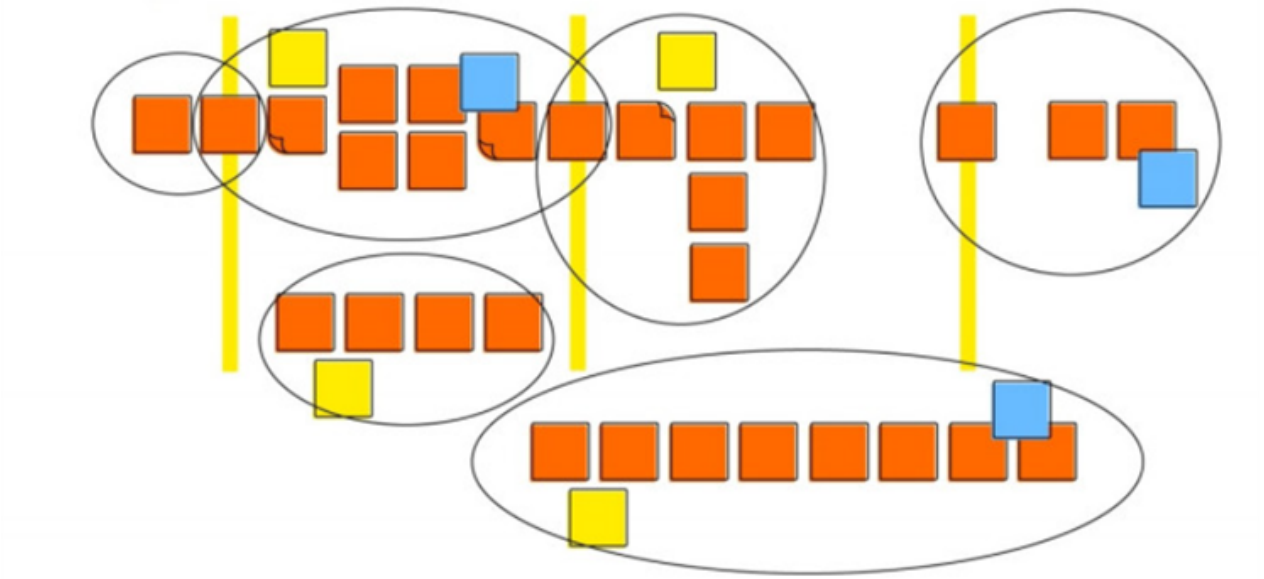
· Сложности коммуникации между экспертами и программистами

· Отсутствие должного опыта при работе с данным подходом

Далее рассмотрим подробнее **метод Событийного штурма**.

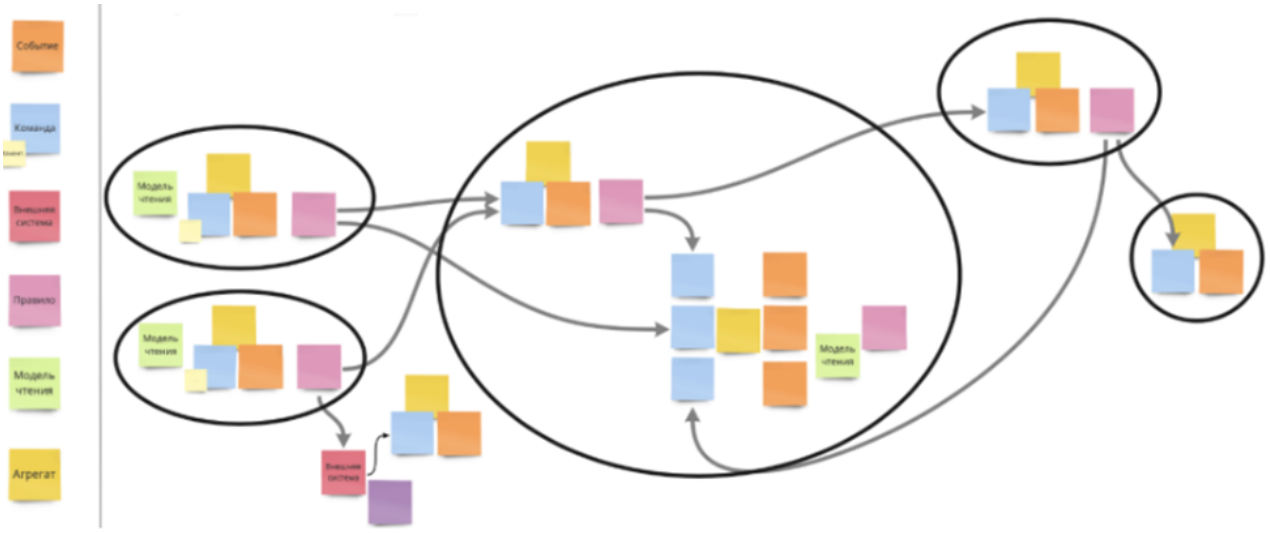
Метод, который позволяет нам создать и организовать устойчивую модульную систему.

Как можно кратко описать? Сессия, при проведении которой нам необходимы большая доска, стикеры, команды разработчиков и экспертов в предметной области, которые могут помочь с ответами на вопросы разработчиков.



При проведении этой сессии в конечном итоге мы получим модель с агрегатами, командами, областями и ограниченным контекстом. Агрегаты и ограниченный контекст - это, в том числе, кандидаты на микросервисы или сервисы. Таким образом шторминг позволяет сразу наметить архитектуру и ответственность отдельных сегментов. Размеры таких досок могут сильно варьироваться в зависимости от количества человек, участвовавших в сессии, и размеров предметной области, которую нужно описать с помощью кода.

Что происходит на этой сессии?



1. Оранжевыми карточками генерируются все события в хронологическом порядке, уточняется смысл каждого события, что помогает программистам лучше понять суть. Как пример: в онлайн-магазине (продукция добавлена в корзину-> … -> заказ оплачен -> …).

2. Голубыми карточками к каждому событию приписывается команда по его воспроизведению. Довольно простой пример: продукция добавлена в корзину (событие) -> добавить в корзину(команда).

3. Сиреневыми карточками описываются правила, которые нужны для обработки события. Также на этом и следующем этапах появляются стрелочки взаимодействие вырисовывающихся модулей.

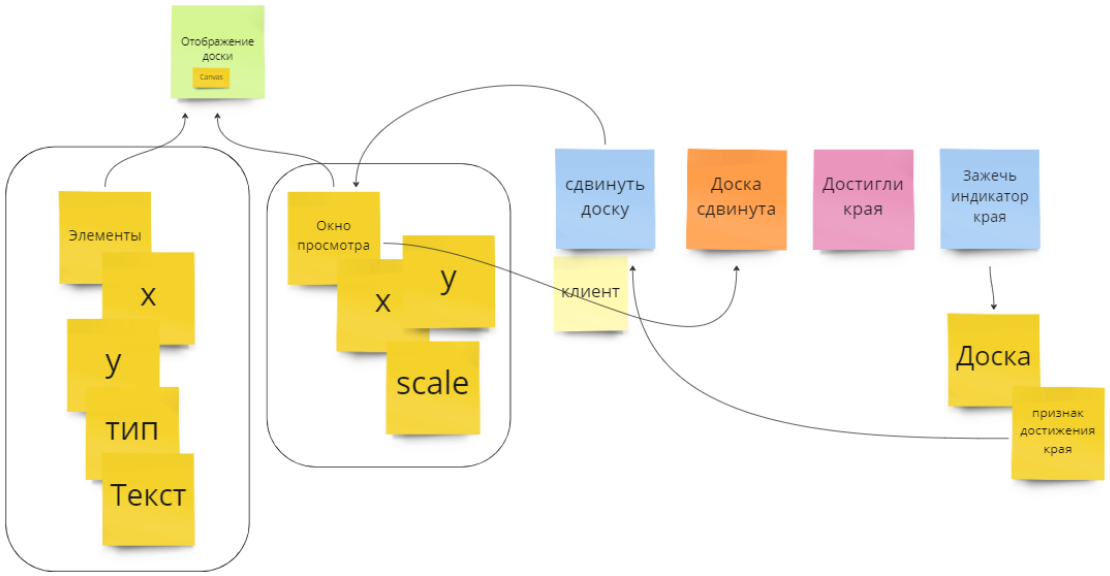
4. Малиновые карточки отвечают за поступление данных к объектам из внешней системы.

5. Зеленые карточки означают модель чтения-представления. Добавляются и обновляются на основе событий с целью помощи в принятии решений клиенту и выполнении следующих команд.

4. Желтыми карточками обозначаются агрегаты (кластер объектов), связанные с остальными карточками. Также агрегаты являются кандидатами на микросервисы и сервисы. Т.е. в процессе шторминга можно выделить основные объекты и контексты. Пример: заказ разделяется на товар, адрес и т.д. При чем при создании доски их можно оставлять пустыми, а уже в конце дополнять.

5. Выделение ограниченных контекстов (показано кружочками), где каждый контекст является автономным и содержит в себе не противоречащие друг другу элементы, то есть сохраняется единый язык, и каждая сущность не имеет разных вариантов существования. Также можно выделить тип взаимодействий этих контекстов, что приводит к созданию такого шаблона стратегического программирования как карта контекстов.

Пример описания отдельного действия на доске в процессе проектирования методикой событийного штурма:



Что может помочь в создании такой доски?

1. Физическая доска

Самым простым примером в данной ситуации является именно физическая доска или даже одноцветная пустая стена, которая предоставляет шанс наклеивать стикеры целой команде.

2. Miro

Среди аналогов к созданию приложения для совместной работы можно выделить Miro. Функционал данного аналога позволяет создавать: карточки различного цвета с

редактируемым текстом; стрелочки, которые помогают устанавливать связь между объектами; различные доски; фигуры для объединения в контекст; а также поддерживает совместную работу большого количества человек. То есть сессия может быть проведена с помощью Miro и любого сервиса для видеоконференций.

3. Confluence/Draw IO

Можно рассмотреть подобные инструменты, в качестве редакторов, предоставляющих возможность рисовать диаграммы, в том числе и что-то близкое к тому, что позволяется в физическом виде. Однако эти инструменты содержат большой минус для команды - неспособность одновременной работы.

Сравнительный анализ функций инструментов

Инструменты	SparX EA	Plant UML	Miro	Notion	Trello	Google sheets	Confluence	DrawIO	Идеал
Диаграммы	Есть	Есть	Есть	Нет	Нет	Нет	С трудом	С трудом	Есть
Карточка	Есть	Нет	Есть	Есть	Есть	Нет	Есть	Нет	Есть
Список карточек	Есть	Нет	Недостаточно хорош	Есть	Есть	Нет	Есть	Нет	Есть
Связи	Есть	Только графические	Только графические	Нет	С помощью тегов	Нет	Нет	Нет	Есть
Простота использования	Нет	Нет	Есть	Есть	Есть	Есть	Есть	Есть	Есть
Совместная работа	Нет	Нет	Есть	Частично	Есть	Есть	Нет	Нет	Есть
Контроль изменений	Есть	Нет	Есть	Есть	Нет	Есть	Есть	Нет	Есть

Описание функциональных и нефункциональных требований к программному проекту

Список требований к инструменту мечты

1. **Диаграмма** - используется для иллюстрации взаимосвязей между объектами, идеями и концепциями. Составляющие диаграммы:
    1.1. **Стикер** - подобие приклеенного куска бумаги на доске, которая помогает помогает делать быстрые заметки во время мозгового штурма, стимулировать дизайн-мышление и сеансы сортировки карточек или визуализировать структуру контента и иерархии информационной архитектуры.
    1.2. **Текст** - инструмент для поддержания текста на доске.
    Текст полезен для вставки дополнительных описаний или представления большого контекста о содержимом на доске.
    1.3. **Фигура** - графический объект, используемый для визуализации идей и решений.
    1.4. **Соединитель** - автоматически настраиваемая стрелка или линия, которая используется для поддержания визуальной связанности разных объектов.
    1.5. **Группа** - объект, позволяющий представить некоторое множество объектов как единое целое.
    1.6. **Фрейм** - объект, предоставляющий возможность визуальной группировки элементов на доске в подобие листа с настраиваемыми размерами.
    1.7. **Таблица** - представление данных в виде стандартной таблицы.
2. **Карточка** - объект диаграммы, созданный по примеру Trello с определенными свойствами:
    2.1. **Имя и описание**
    2.2. **Вложение** - файлы и ссылки находящиеся внутри карточки
    2.3. **Тег -**свойство, позволяющее группировать объекты по какому либо очертанию
    2.4. **Атрибуты** - пара название/ значение атрибута, пример: название - “Статус задачи”, значение - “Выполнено”
3. **Список карточек** – возможность просматривать все карточки в виде списка
4. **Связи** - соединять разные элементы доски под единую логику
5. **Инструменты классификации на диаграмме:**
    5.1. Классификация на основе пространственных связей на диаграмме
    5.2. Расширенные функции упорядочивания, например, для перехода от конвергенции к дивергенции
    5.3. Авторасскладка элементов со связями на диаграмме
6. **Не графические представления данных - альтернативное представление диаграмм**
    6.1. **Иерархический список -** возможность просматривать иерархические связи между данными (как в SparX Enterprise Architect**)**
    6.2. **Список с атрибутами (таблица)** - возможность группировать по атрибутам данные
    6.3. **Двумерная классификация (с иерархическими осями)** - возможность обрисовать таблицу элементов с атрибутами и динамически менять значения атрибутов на элементе
    6.4. **Дерево связей** - навигация по связям
    6.5. **Вычисляемые поля как в Excel -** возможность группировать объекты и получить агрегацию по атрибутам
7. **Легкое расширение на Python/JS + API доступа или формат**
    7.1. **Возможность генерации кода на Python/JS на основе диаграмм**

    7.2. **Возможность генерации документации на основе диаграмм**
    7.3. **API для управления и доступа к информации**
8. **Контроль изменений в GIT**
   Возможность просмотра диаграмм в понятном человеку текстовом формате.
9. **Управление версиями как в GIT:**
    9.1. **Коммиты**
    9.2. **Сравнение**
    9.3. **Ветки**
    9.4. **Слияние**
10. **Совместная удаленная работа на одной доске как в Miro или Google Docs**
11. **Архитектура, позволяющая простое расширение функционала с помощью модулей**

Список требований к текущему проекту

Временный рамки не позволяют выполнить весь задуманный инструмент проектирования, поэтому в текущей курсовой работе мы выбирали только часть элементов:

- **Диаграмма**
    - **Стикер**
    - **Соединитель**
    - **Текст**
    - **Группа**
    - **Карандаш**
    - **Фигура**
    - **Фрейм**
    - **Таблица**
- **Карточка**
    - **Имя**
    - **Описание**
    - **Вложения**
    - **Связи**
    - **Тэги**
    - **Экспорт в таблицу**
- **Список карточек**
    - **Сортировка**
    - **Фильтрация**
- **Сохранение в текст (пригодный для GIT)**
    - **Графический просмотр отличий**

Календарный план выполнения проекта с указанием этапов и сроков выполнения

Задача	Выполнить к	Кто выполняет	Приоритет	Отметка о выполнении
Архитектура	10.01.2023	Все	Обязательно	Выполнена
Модуль “Текст”	17.01.2023	Все	Обязательно	Выполнено
Анализ подходов проектирования	01.02.2023	Все	Обязательно	Выполнено

Сравнение аналогов	08.02.2023	Все	Обязательно	Выполнено
Модуль “Ядро”	05.03.2023	Все	Обязательно	В процессе
Модуль “Стикер”	01.04.2023	Амина	Обязательно	В процессе
Модуль “Соединитель”	01.04.2023	Сардор	Обязательно	В процессе
Модуль “Группа”	01.04.2023	Антон	Обязательно	В процессе
Модуль “Карандаш”	01.04.2023	Лилиана	Обязательно	В процессе
Тестирование	01.05.2023	Все	Обязательно	Не выполнено
Контроль версий	01.05.2023	Все	Обязательно	Не выполнено
Модуль “Карточка”	-	-	В рассмотрении	Не выполнено
Модуль “Список карточек”	-	-	В рассмотрении	Не выполнено
Модуль “Фигуры”	-	-	В рассмотрении	Не выполнено
Модуль “Фрейм”	-	-	В рассмотрении	Не выполнено
Модуль “Таблица”	-	-	В рассмотрении	Не выполнено

Список источников

1. Официальная спецификация BPMN // Object Management Group [Электронный ресурс]. Режим доступа: https://www.omg.org/spec/BPMN/2.0/PDF (дата обращения 12.02.2023).
2. Зачем нам UML? Или как сохранить себе нервы и время // Хабр [Электронный ресурс]. Режим доступа: https://habr.com/en/post/458680/ (дата обращения 12.02.2023).
3. PlantUML – инструмент продуктового разработчика // Блог компании «Qiwi» на портале «Хабр» [Электронный ресурс]. Режим доступа: https://habr.com/en/company/qiwi/blog/577606/ (дата обращения 12.02.2023).
4. Использование диаграммы вариантов использования UML при проектировании программного обеспечения // Хабр [Электронный ресурс]. Режим доступа: https://habr.com/en/post/566218/ (дата обращения 12.02.2023).
5. Краткое описание нотации BPMN // Блог компании «Auriga» на портале «Хабр» [Электронный ресурс]. Режим доступа: https://habr.com/en/company/auriga/blog/667084/ (дата обращения 12.02.2023).
6. Документация PlantUML // Веб-сайт PlantUML [Электронный ресурс]. Режим доступа: https://plantuml.com/ (дата обращения 12.02.2023)
7. Диаграмма активностей // Платформа Flexberry [Электронный ресурс]. Режим доступа: https://flexberry.github.io/ru/fd_activity-diagram.html (дата обращения 12.02.2023)
8. Диаграмма вариантов использования // Платформа Flexberry [Электронный ресурс]. Режим доступа: https://flexberry.github.io/ru/fd_use-case-diagram.html (дата обращения 12.02.2023)
9. UML Class Diagram // Ресурс с туториалами Javatpoint [Электронный ресурс]. Режим доступа: https://www.javatpoint.com/uml-class-diagram (дата обращения 12.02.2023)
10. Deployment Diagram for E-commerce Microservices Architecture // Software Ideas Modeler [Электронный ресурс]. Режим доступа: https://www.softwareideas.net/a/1580/e-commerce-microservices-uml-deployment-diagram- (дата обращения 12.02.2023)
11. Халл Э,, Джексон К., Дик Дж. Инженерия требований / пер. с англ. А. Снастина; под ред. В. К. Баторвина. – М. : ДМК Пресс, 2017.
12. BPWin Methods Guide. Logic Works Inc., Princeton, New Jersey, 1997.
13. Марка Д., МакГоуэн К. Методология структурного анализа и проектирования (SADT) / Пер. с англ. - М: МетаТехнология, 1993.
14. Основы методологии IDEF1 // Корпоративный менеджмент [Электронный ресурс]. Режим доступа: https://www.cfin.ru/vernikov/idef/idef1.shtml (дата обращения 12.02.2023)
15. Основы методологии IDEF1X // Корпоративный менеджмент [Электронный ресурс]. Режим доступа: https://www.cfin.ru/vernikov/idef/idef1x.shtml (дата обращения 12.02.2023)
16. ГОСТ Р 59194–2020. - Издание официальное. - М. : Стандартинформ, 2020. - 29 с.
17. Вон Вернон. Реализация методов предметно-ориентированного проектирования/ пер. с англ. под ред. Д.А. Клюшина, 2016
18. Джефф Паттон. Пользовательские истории
19. Гойко Аджич. IMPACT MAPPING. Как повысить эффективность программных продуктов
20. 4-контекстная диаграмм // [Электронный ресурс]. Режим доступа: https://c4model.com/ (дата обращения 12.02.2023)