Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

И. Н. Тонкович, А. В. Шелест

ТЕХНОЛОГИИ ПРОЕКТИРОВАНИЯ СЛОЖНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве учебно-методического пособия для специальности 6-05-0611-01 «Информационные системы и технологии»

Минск БГУИР 2025

УДК 004.42(076)
ББК 32.973.3я73
Т57

Р е ц е н з е н т ы:

кафедра программного обеспечения
информационных систем и технологий
Белорусского национального технического университета
(протокол № 4 от 22.11.2023);

генеральный директор ОАО «Планар»
доктор технических наук С. М. Аваков

Тонкович, И. Н.

Т57 Технологии проектирования сложных информационных систем : учеб.-метод. пособие / И. Н. Тонкович, А. В. Шелест. – Минск : БГУИР, 2025. – 167 с. : ил.

ISBN 978-985-543-779-7.

Приведены краткие теоретические сведения и методические указания по выполнению лабораторных работ по проектированию и разработке программных решений.

УДК 004.42(076)
ББК 32.973.3я73

ISBN 978-985-543-779-7

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ

 – важно

 – обратить внимание (заметка)

 – определение

 – задание

 – пример

 – теоретические сведения

 – отсылка к дополнительным сведениям и соответствующему источнику

ВВЕДЕНИЕ

Современные информационные системы обеспечивают оперативность коммуникации и интеграцию участников бизнес-процессов, повышают качество принимаемых решений на всех уровнях управления. Усложнение архитектуры современных информационных систем предопределяет разработку и использование эффективных технологий проектирования, обеспечивающих ускорение создания, внедрения и развития проектов информационных систем, повышение их функциональной и адаптивной надежности.

Цель учебной дисциплины «Технологии проектирования сложных информационных систем» – подготовка специалиста, владеющего теоретическими знаниями в области анализа и проектирования информационных систем и практическими навыками разработки программных решений, их эффективной реализации на мультипарадигменных языках программирования.

Задачи учебной дисциплины:

- овладение методами, технологиями и средствами анализа и моделирования информационных систем;
- формирование базовых научно-теоретических знаний и практических навыков по основам проектирования сложных информационных систем;
- обучение работе в интегрированной среде программирования при разработке программного обеспечения, овладение техникой развертывания, отладки и применения для решения различных практических задач.

Лабораторный практикум по дисциплине «Технологии проектирования сложных информационных систем» включает 11 лабораторных работ (рисунок 1), направленных на выполнение сквозного задания в рамках индивидуального проекта – от разработки требований до получения готового программного средства.

Представлены следующие этапы разработки программного обеспечения:

- разработка требований к программному обеспечению – лабораторные работы № 1–4;
- проектирование программного обеспечения – лабораторные работы № 5–8;
- конструирование программного обеспечения – лабораторная работа № 9;
- тестирование программного обеспечения – лабораторная работа № 10;
- оценка качества программного обеспечения – лабораторная работа № 11.

Все выполняемые лабораторные работы связаны между собой содержательно – последующая использует материал и результаты выполнения предыдущей. В каждой лабораторной работе приводятся название работы, цель, перечень практических заданий, методические рекомендации по их выполнению, теоретические выкладки, требования к содержанию отчета, контрольные вопросы.

Уделяется внимание основным сложностям, связанным с выполнением работы, и типичным ошибкам.

Лабораторная работа № 1	Разработка концепции и границ проектного решения
Лабораторная работа № 2	Экономическое обоснование проектного решения
Лабораторная работа № 3	Анализ и моделирование предметной области
Лабораторная работа № 4	Разработка требований к программному средству
Лабораторная работа № 5	Проектирование архитектурных решений
Лабораторная работа № 6	Проектирование и разработка пользовательского интерфейса программного средства
Лабораторная работа № 7	Организация системы хранения данных
Лабораторная работа № 8	Разработка проектных решений на основе моделирования свойств и линий поведения программных объектов
Лабораторная работа № 9	Реализация программного решения
Лабораторная работа № 10	Тестирование программного средства
Лабораторная работа № 11	Внедрение программного средства. Оценка качества разработанного программного средства

Рисунок 1 – Перечень тем лабораторных работ

Методически лабораторные работы практикума построены на принципах проектного метода обучения, основное назначение которого состоит в предоставлении студентам возможности самостоятельно конструировать свои знания в процессе реализации проектного решения по разработке программного средства, что требует интеграции знаний из различных предметных областей.

Преимуществом такой деятельности является возможность оптимально сочетать исследовательский и практико-ориентированный характер учебной деятельности, что представляется достаточно важным при подготовке специалиста.

При подготовке данного учебно-методического пособия были использованы материалы методических пособий «Разработка программного обеспечения: планирование, анализ и моделирование» и «Разработка программного обеспечения: проектирование, конструирование и внедрение» по дисциплине «Технологии проектирования сложных информационных систем» для студентов очной формы обучения учреждений высшего образования направления специальности 1-40 05 01-10 «Информационные системы и технологии (в бизнес-менеджменте)» [1; 2].

Учебно-методическое пособие предназначено для студентов очной формы получения образования по специальности 6-05-0611-01 «Информационные системы и технологии» (профилизация «Информационные системы и технологии в бизнес-менеджменте»). Может быть использовано студентами заочной и дистанционной форм получения образования и будет полезно всем, чья профессиональная деятельность связана с разработкой программного обеспечения для информационных систем.

ОБЩАЯ ПОСТАНОВКА ЗАДАЧИ НА ВЫПОЛНЕНИЕ ЛАБОРАТОРНЫХ РАБОТ

Реализовать проектное решение по разработке программного средства (ПС) на мультипарадигменном языке программирования с использованием современных технологий, фреймворков и библиотек в соответствии с требованиями, представленными на схеме (рисунок 2).

Программное средство должно выполняться в операционной системе Windows 8 и выше (Android 10 и выше, iOS 13 и выше) с возможной предустановкой библиотек или пакетов выбранной среды программирования и запускаться без использования любых интегрированных средств разработки.

Программная документация представляется в составе руководства по установке (развертыванию) программного средства и руководства по работе для всех категорий пользователей.

Для решения данной задачи выявляются бизнес-потребности заказчика, выполняются анализ и моделирование предметной области с представлением визуальных моделей, построенных с помощью современных case-средств; разрабатывается спецификация требований к программному средству; осуществляется проектирование архитектуры и дизайна; реализуются программные решения; представляются доказательства работоспособности программного средства на основе техник тестирования; оформляется необходимая текстовая и графическая документация.

Варианты тем для выполнения лабораторных работ представлены в приложении А.

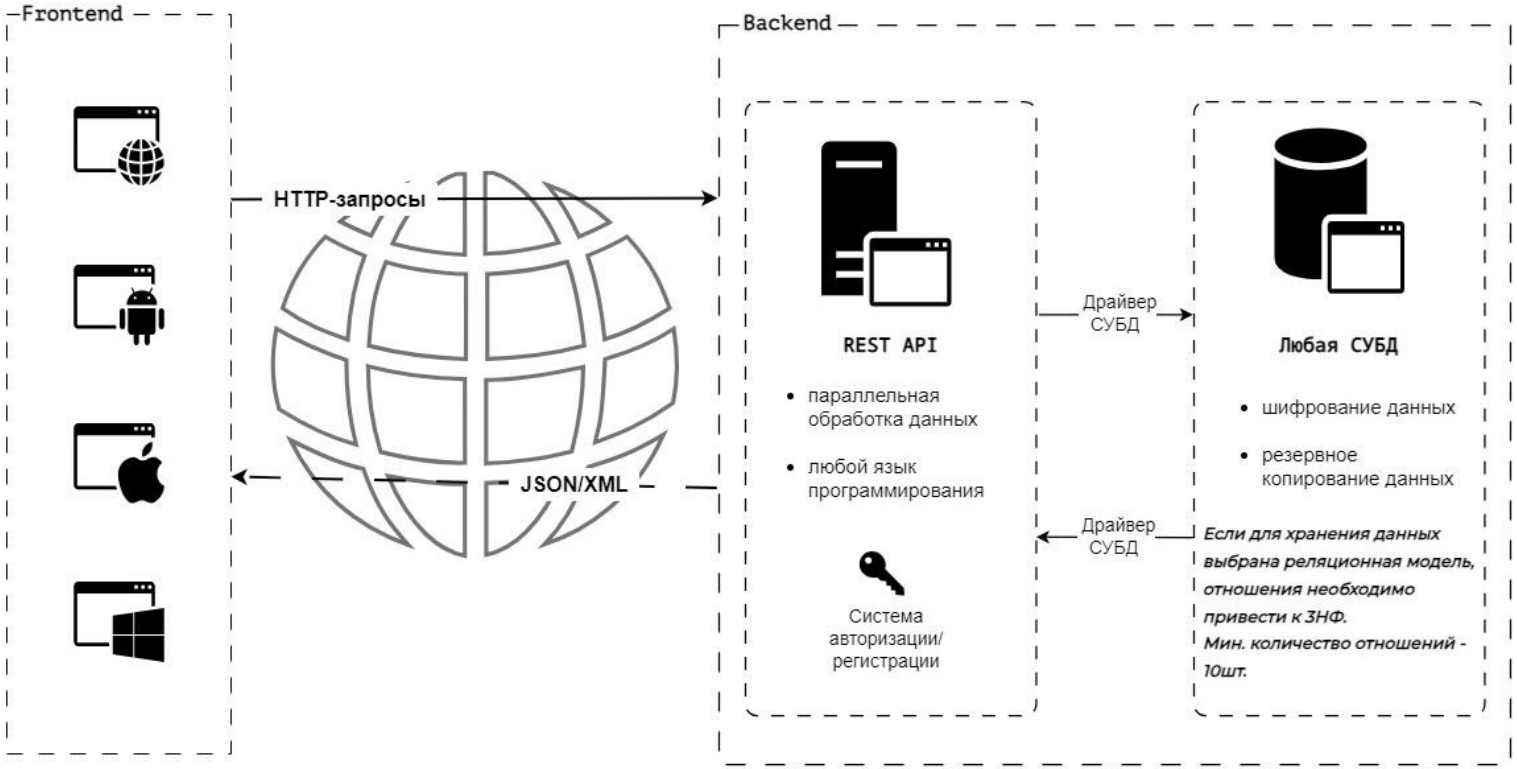


Рисунок 2 – Требования к программному средству

ТРЕБОВАНИЯ К ОРГАНИЗАЦИИ, ПРОВЕДЕНИЮ И ЗАЩИТЕ ЛАБОРАТОРНЫХ РАБОТ

Подготовка, оформление и защита лабораторных работ регламентированы Порядком подготовки, оформления и защиты лабораторных работ студентами I ступени высшего образования от 26.06.2013.

Организация и проведение лабораторных занятий. Подготовка к лабораторным занятиям осуществляется студентами самостоятельно и заблаговременно. В процессе подготовки студент должен усвоить теоретический материал, относящийся к лабораторной работе, изучить и ясно представлять себе содержание и порядок выполнения лабораторной работы.

Выполнение лабораторных работ производится *в течение занятия* в составе подгруппы. После выполнения лабораторной работы студенты предъявляют преподавателю результаты, *оформленные в виде отчета.*

Отработка пропущенных студентом без уважительных причин лабораторных работ осуществляется в соответствии с порядком повторной текущей и итоговой студентов I и II ступеней высшего образования, аспирантов и соискателей ученых степеней БГУИР. Студентам, пропустившим лабораторные занятия по уважительным причинам, *в порядке исключения* преподавателем или деканатом может быть разрешено выполнение лабораторных работ в течение семестра, например, с другой группой.

Студенты, не защитившие лабораторные работы по учебной дисциплине, *к текущей аттестации по данной дисциплине не допускаются* как не выполнившие график образовательного процесса.

Оформление отчета и защита лабораторных работ. Содержание и структура отчета определяются требованиями и методическими рекомендациями к лабораторной работе.

Отчет по выполненной лабораторной работе должен соответствовать следующим требованиям:

- оформление отчета осуществляется согласно требованиям СТП БГУИР 01–2024 «Дипломные проекты (работы)»;
- название файла отчета формируется согласно шаблону: *Лабораторная работа № X,* где X – порядковый номер лабораторной работы;
- титульный лист отчета по лабораторной работе оформляется согласно *приложению Б*;
- *в обязательном порядке* на титульном листе отчета проставляются подпись студента, выполнившего лабораторную работу, и дата ее защиты;
- оформленный отчет по лабораторной работе следует разместить в папке с названием семестра обучения на Google-диске (ссылку уточнять у преподавателя), соблюдая структуру директорий (рисунок 3).

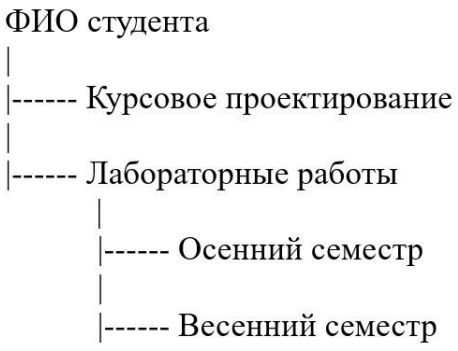


Рисунок 3 – Организация директорий

В случае невыполнения одного из вышеуказанных требований студент не будет допущен к защите лабораторных работ.

Защита лабораторных работ проводится во время *очных занятий* согласно графику защиты лабораторных работ, а также *в сроки, установленные кафедрой.*

Оценка результатов защиты лабораторных работ. По результатам защиты лабораторной работы студенту выставляется отметка по 10-балльной шкале. В случае получения студентом отметки 4 балла и более работа считается защищенной.

Оценка результатов выполнения лабораторной работы осуществляется в соответствии с Постановлением Министерства образования Республики Беларусь № 319 «Об утверждении Правил проведения аттестации студентов, курсантов, слушателей при освоении содержания образовательных программ высшего образования» от 13.10.2023.

Проведение повторной защиты лабораторных работ. В ситуации получения студентом отметки ниже 4 баллов по результатам защиты лабораторной работы ему необходимо подать заявление на имя декана факультета с просьбой разрешить повторно отчитаться по результатам выполнения лабораторной работы. Декан факультета рассматривает поданное заявление, на его основе определяет возможность и стоимость повторной аттестации в соответствии с действующими в университете ценами на платные услуги и сроки ее проведения.

После предъявления квитанции об оплате в деканат студенту выдается разрешение, которое дает право отработать пропущенные лабораторные работы или повторно отчитаться по их результатам.

Преподаватель на основании разрешения деканата назначает время и место выполнения студентом учебной работы, организует ее и осуществляет аттестацию студента. При отработке лабораторных работ преподаватель, ведущий лабораторный практикум, формирует группы из задолжников (5–7 человек) и сообщает им дату, номер аудитории и время отработки лабораторных работ или повторной защиты. Результаты аттестации преподаватель в установленном порядке представляет в деканат.

Порядок принятия решений в частных случаях. Принятие решений в случаях, не предусмотренных Порядком подготовки, оформления и защиты лабораторных работ, осуществляют в пределах своей компетенции учебно-методическое управление университета, деканаты, заведующие кафедрами и преподаватели, ведущие лабораторные занятия на основании Положения об учреждении высшего образования, Устава БГУИР и других нормативных актов.

Организация хранения исходного кода. Перед выполнением лабораторных работ необходимо создать публичный репозиторий, используя любую систему контроля версий (СКВ). Стоит учесть, что выбранная СКВ должна поддерживать возможность обеспечения доступа к хранилищу исходного кода незарегистрированным в ней пользователям (это необходимо для предоставления возможности проверки исходного кода преподавателю).

По мере выполнения лабораторных работ необходимо размещать исходный код программы в выбранной СКВ (*ссылку на аккаунт с исходным кодом необходимо приводить в начале отчета по каждой лабораторной работе*). Название репозитория СКВ должно соответствовать маске: FamiliIO_№группы. Все репозитории, название которых не соответствует маске, не будут учитываться при защите лабораторных работ.

В репозитории нужно создать две ветки: main и develop. В ветке main будет находиться полностью рабочий код разрабатываемого программного средства, а ветке develop – код, находящийся в стадии разработки.

ОБЩИЕ СВЕДЕНИЯ О РАЗРАБОТКЕ ТРЕБОВАНИЙ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

На сегодняшний день не существует единой методологии разработки программного обеспечения. Каждый IT-проект является уникальным и осуществляется в соответствии с различными целями и задачами, отличается масштабом, квалификацией разработчиков и т. д.

Выделяют следующие группы подходов к разработке программного обеспечения [3]:

1 Подходы со слабой формализацией, которые не используют явных технологий и их можно применять только для малых проектов, как правило, завершающихся созданием демонстрационного прототипа.

2 Строгие (классические, жесткие, предсказуемые) подходы, которые рекомендуется применять для средних, крупномасштабных и гигантских проектов с фиксированным объемом работ. Одно из основных требований к таким проектам – предсказуемость.

3 Гибкие (адаптивные, легкие) подходы, которые рекомендуется применять для небольших или средних проектов в случае неясных или изменяющихся требований к системе. При этом команда разработчиков должна быть ответственной и квалифицированной, а заказчики должны принимать участие в разработке.

Практически все существующие проектные методологии разработки программного обеспечения можно отнести к одному из двух типов.

1 Методологии, ориентированные на план. Такие методологии направлены на минимизацию неопределенности решения в целях максимального контроля и минимизации рисков (например, Waterfall). На основе данных методологий реализуются проекты высокой сложности, с большими рисками или проекты, на которых затруднен доступ к заинтересованным лицам.

2 Методологии, ориентированные на изменения. Направлены на быструю поставку бизнесу рабочего продукта за короткие итерации в ситуациях, когда четкого видения будущего решения не существует (Scrum, Kanban, RAD, RUP, XP, MSF и др.). Как правило, на основе данных методологий реализуются проекты с меньшим риском и возможностью прямого участия заинтересованных лиц и сбора регулярной обратной связи.

Неверный выбор методологии разработки может привести к финансовым и временным потерям и тем самым к снижению прибыли компании-разработчика. Важно верно определить методологию разработки.

Используемые в настоящее время IT-специалистами жизненные циклы программного обеспечения в большинстве случаев отличаются от приведенных в существующих стандартах (международных, серий ГОСТ 34 и ГОСТ 19), что обусловлено развитием современных архитектур, внедрением методологий

быстрой разработки программных решений, case-систем, языков четвертого поколения (L4G), появлением концепта фреймворка и языков уже пятого поколения (L5G).

Поэтому целесообразно выбирать и использовать апробированные стандарты и адаптировать их под реализацию конкретного проекта, конкретных условий и конкретной проектной команды.

Успешные системы и продукты начинаются с глубокого понимания потребностей и нужд конечных пользователей.

В этой связи первым и самым важным этапом разработки программного обеспечения является разработка требований. И начинается он с того, что владелец продукта разрабатывает четкое видение того, чего должен достичь программный продукт, какая у него целевая аудитория и как он вписывается в стратегию, бизнес-цели компании. Далее выявляется, анализируется и описывается требуемая функциональность, а в качестве результата представляется спецификация требований.

Требования – отправная точка понимания того, что мы будем проектировать, конструировать и тестировать.

В глоссарии IEEE Standard Glossary of Software Engineering Terminology (1990) дается следующее определение требованиям:

- условия или возможности, необходимые пользователю для решения проблем или достижения целей;
- условия или возможности, которыми должна обладать система или системные компоненты, чтобы выполнить контракт или удовлетворять стандартам, спецификациям или другим формальным документам;
- документированное представление условий или возможностей для вышеприведенных пунктов.

Требования указывают на то, какая функциональность системы и с соблюдением каких условий должна быть реализована, а также какое поведение должна демонстрировать в процессе решения полезной для пользователя задачи, но не говорят о том, каким образом эта функциональность реализуется.

Требования должны быть полные (достаточные), однозначные (недвусмысленные), необходимые (зачем-то кому-то), осуществимые (в данных условиях), приоритизированные (по некоторому критерию), проверяемые (возможно, ли проверить реализацию), модифицируемые, корректные.

Вопросы классификации требований широко представлены в методологиях и стандартах (SWEBOK, BABOK, IEE 830, RUP, ГОСТ 34, FURPS+), в ряде фундаментальных работ К. Вигерса, Д. Леффингуэлла. Однако на сегодняшний день не существует единой системы, объединяющей все варианты классификаций.

Суммируя известные подходы к классификации требований, и в целях наиболее полного описания требований на пути от бизнеса к технической реализации будем использовать модель выявления требований, схематично представленную на рисунке 4.

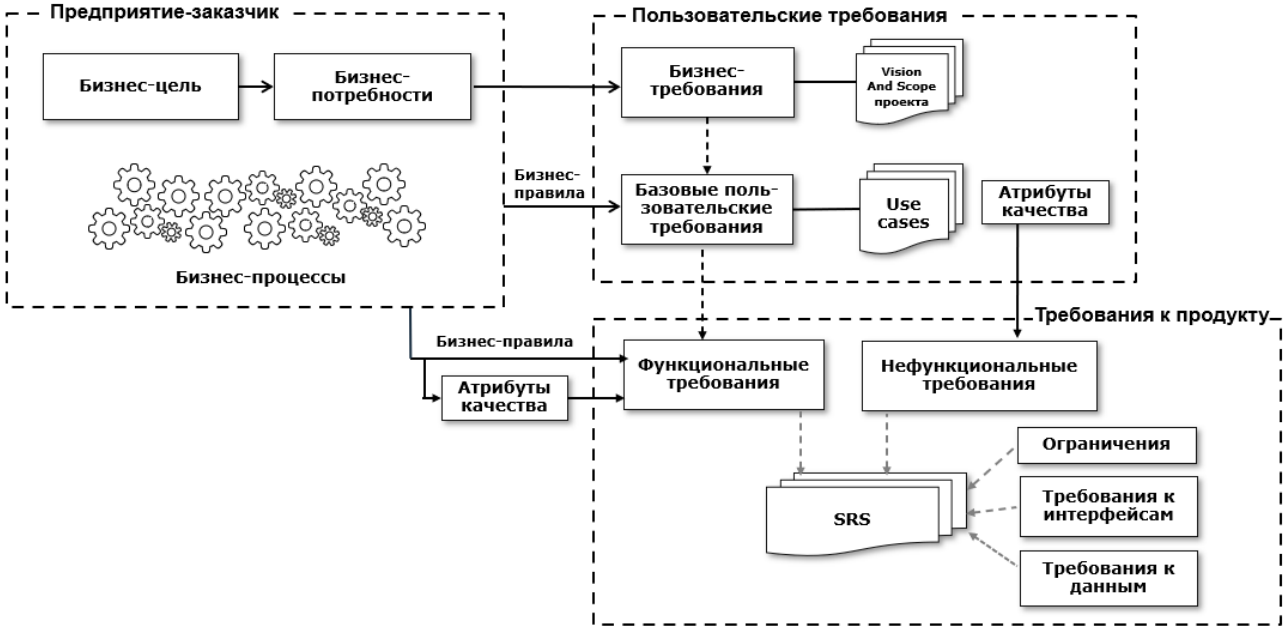


Рисунок 4 – Модель выявления требований к программному продукту

Цель любой разработки – создать продукт, максимально полно соответствующий насущным потребностям и бизнес-целям предприятия-заказчика. Исходя из этого в приведенной выше модели все требования разделены на две категории:

- потребности пользователей – пользовательские требования;
- решения, которые необходимо реализовать для удовлетворения заявленных потребностей – требования к продукту.

Начало разработки программного обеспечения обуславливается четким определением бизнес-целей. Важно понимать, что предприятие-заказчик хочет достичь через внедрение программного продукта. Эти цели могут включать в себя, например, улучшение операционной эффективности, увеличение прибыли или расширение рынка. Ключевым шагом на этапе формирования требований является выявление конкретных потребностей предприятия-заказчика, например, улучшение бизнес-процессов, автоматизация определенных задач или обеспечение новых возможностей для конечных пользователей.

Исходя из бизнес-целей и потребностей предприятия-заказчика формируются бизнес-требования. Эти требования определяют общую картину того, что должно быть достигнуто с помощью программного продукта, например, внедрение системы управления заказами для сокращения времени обработки заказов.

В системе иерархии требований на верхнем уровне находятся именно бизнес-требования, которые отражают цели бизнеса, его потребности и формируют Документ о концепции и границах проекта (Vision and Scope document). Документ о концепции и границах – это дорожная карта, которая определяет будущее программного продукта. Этот документ должен четко формулировать видение проекта, его цели и ограничения, обеспечивая стабильный фундамент для дальнейшей разработки.

Следует отметить, что процесс выявления требований для разработки продуктов на заказ (в рамках аутсорсинговой модели или для внедрения в собственные бизнес-процессы разработчика) и для массового рынка (продуктовая модель) существенно различается.

Продукт на заказ – программное обеспечение, которое разрабатывается индивидуально для конкретной компании (предприятия, организации). В этом случае заказчик определен и проводится полный анализ его требований.

Продукт для массового рынка – заказчиком выступает рынок. Следует определить целевой сегмент рынка и потребности его заказчиков. Цели продукта основываются на конкурентном анализе аналогов.

В целом успешное формирование бизнес-требований требует усилий, направленных на глубокое понимание потребностей заказчика, широкое взаимодействие со стейкхолдерами.

На основе бизнес-требований формируются базовые пользовательские требования. Это следующий уровень в иерархии требований. Данные требования ориентированы на конечного пользователя; определяют, каким образом программный продукт будет использоваться; описывают потребности пользователей и оформляются в виде вариантов использования (use cases). Стоит заметить, применение use cases не всегда оправдано и зависит от стадии проекта, подхода к разработке, договоренностей команды. Например, use cases используют, когда программный продукт разрабатывают с нуля по Waterfall или по Agile, когда над задачами можно работать параллельно. В случае больших проектов пользовательские требования предпочтительнее представлять в виде пользовательских историй (user stories) или сценариев.

Обеспечение удовлетворения пользовательских потребностей становится критическим фактором для успеха программного продукта. Систематический анализ, активное вовлечение стейкхолдеров и использование современных методов проектирования помогают сформулировать четкие и точные пользовательские требования, что, в свою очередь, является фундаментом для разработки успешных программных продуктов.

Требования к программному продукту представляются в документе SRS (Software requirements specification) и, как правило, включают:

- функциональные требования, которые описывают функциональные возможности или поведение программной системы при определенных условиях и формируются на основе пользовательских требований, а также могут проистекать из бизнес-правил и других источников, составляющих основу спецификации требований к программному обеспечению;
- нефункциональные требования, которые определяют качественные характеристики программной системы. Это могут быть требования к производительности, безопасности, масштабируемости и другим аспектам.

ЛАБОРАТОРНАЯ РАБОТА № 1
Разработка концепции и границ проектного решения

Цели выполнения лабораторной работы:

- проанализировать бизнес компании и его продукт;
- разработать Документ о концепции и границах проектного решения.

Задание	**Этапы выполнения задания**
1 Провести анализ бизнеса и его продукта	1 Дать общее представление о компании, представить основные направления деятельности, товары/услуги и целевой рынок, бренд. 2 Определить видение, миссию, стратегические цели, бизнес-стратегию и бизнес-цели компании. Сформировать стратегические цели, следуя формату SMART. 3 Разработать и описать Business Model Canvas. На основе анализа Business Model Canvas определить, какие бизнес-процессы требуют изменений, чтобы достигнуть заданных целей и как решить эту проблему с помощью разработки программного продукта. 4 Описать и продемонстрировать использование другой техники анализа бизнеса и его продукта (на выбор)
2 Разработать Документ о концепции и границах проекта (vision and scope)	1 Определить бизнес-требования к программному средству. 2 Очертить границы проекта. 3 Представить бизнес-контекст
3 Сформировать отчет по лабораторной работе	*Содержание отчета*: Титульный лист (приложение Б) 1 Анализ бизнеса и его продукта 1.1 Характеристика компании 1.2 Видение, миссия, стратегические цели, бизнес-стратегия и бизнес-цели компании 1.3 Business Model Canvas, ее описание и анализ 1.4 Описание выбранной техники анализа бизнеса и его продукта 2 Документ о концепции и границах проектного решения

Краткие теоретические сведения и методические указания к заданию 1

Сегодня предприятия (организации, компании) нуждаются в IT-решениях, которые позволят сосредоточиться на стратегически важных направлениях ведения бизнеса, максимизировать эффективность его работы.

На старте проекта по разработке программного обеспечения необходимо получить максимум информации о заказчике, детально изучить специфику его деятельности и бизнес-задач; выявить проблемы и потребности конечных пользователей в программном продукте; уточнить рамки проектного решения и определить его цель и задачи, т. е. подготовить основу для последующей разработки требований к продукту.

Основная цель анализа бизнеса и его продукта – определить направления развития, а также понять, какие бизнес-процессы требуют изменений, чтобы достигнуть заданных целей, и как решить эту проблему с помощью информационных технологий.

Для того чтобы понять особенности бизнеса и/или потенциального решения, рекомендуем, в первую очередь, сосредоточиться на таких составляющих стратегического управления, как:

- *видение* (заявление о том, каким вы видите будущее компании, что заставляет двигаться вперед);
- *миссия* (главная долгосрочная цель, смысл существования предприятия);
- *ценность* (основные принципы (правила), которыми руководствуется предприятие на пути к выполнению миссии в ежедневной работе);
- *стратегическая цель* (результат, достижение которого представляет ценность для организации);
- *бизнес-стратегия* (способ достижения стратегических целей);
- *бизнес-тактика* (план действий по реализации бизнес-стратегии).

Для выявления ключевых аспектов деятельности предприятия (компании, организации) внимательно изучайте всевозможную документацию.

При формализации стратегических целей необходимо учесть, что они должны удовлетворять *SMART*-критериям.

SMART – это система постановки целей, помогающая понять, что и в какой последовательности необходимо сделать для достижения желаемого результата.

Каждая буква в акрониме SMART фокусируется на различных аспектах в достижении желаемой цели:

- *S* (*Specific*) – конкретность, ясность. Цель должна быть конкретной и четко сформулированной.

- *M* (*Measurable*) – измеримость. Цель должна иметь количественные и качественные параметры, по которым ее можно оценить.

- *A* (*Assignable*) – достижимость. Цель должна быть реалистичной в тех временных и бюджетных рамках, которые отводятся для ее реализации.

- *R* (*Relevant*) – адекватность, актуальность, реалистичность. Цель должна быть адекватной и согласованной с другими целями.

- *T* (*Time-bound*) – ограниченность по времени, обозначение цели во времени. Цель должна быть ограниченна временными рамками и иметь определенный срок достижения.

ПОСТАНОВКА ЦЕЛЕЙ ПО SMART

Пример постановки SMART-целей представлен на рисунке 5.

S	Specific	Снизить затраты на производство продукции
M	Measurable	Снизить затраты на производство продукции на 10 %
A	Achievable	Снизить затраты на производство продукции на 10 % за счет снижения затрат на сырье
R	Relevant	Снизить затраты на производство продукции на 10 % за счет снижения затрат на сырье, если перейти на альтернативные виды сырья, покупаемые по более низкой цене
T	Time-bound	К окончанию третьего квартала следующего года снизить затраты на производство продукции на 10 % за счет снижения затрат на сырье, если перейти на альтернативные виды сырья, покупаемые по более низкой цене

Рисунок 5 – Постановка SMART-целей

Для качественной оценки текущего положения компании и точек роста ее бизнеса необходимы соответствующие методические инструменты.

Наиболее популярные техники анализа бизнеса и его продукта приведены ниже.

Техники анализа бизнеса и его продукта: Business Model Canvas, Diagram Ishikawy, SWOT-анализ, PEST-анализ, MOST-анализ, CATWOE, GAP-анализ, MoSCoW-анализ.

Описание данных техник анализа представлено в таблице 1.

Таблица 1 – Техники анализа бизнеса и его продукта

Техники анализа бизнеса и его продукта	**Описание**	**Когда использовать**
1	**2**	**3**
Business Model Canvas	Инструмент стратегического управления для новых или действующих компаний для структурированного отображения того, как бизнес создает ценность товаров	Для анализа существующей модели с целью нахождения слабых мест или новых точек роста

Продолжение таблицы 1

1	2	3
Diagram Ishikawy	Техника исследования первопричин, также известная как диаграмма Исикавы, диаграмма «рыбьей кости», причинно-следственная диаграмма, диаграмма анализа корневых причин	Когда необходимо визуализировать и оценить наиболее существенные причинно-следственные взаимосвязи между факторами и последствиями в исследуемой ситуации или проблеме
SWOT-анализ	Техника стратегического планирования, позволяющая понять: а) факторы *внутренней среды* объекта анализа (т. е. то, на что сам объект способен повлиять): - *S* (Strengths) – сильные стороны компании или продукта; - *W* (Weaknesses) – слабые стороны компании или продукта; б) факторы *внешней среды* (т. е. то, что может повлиять на объект извне и при этом не контролируется объектом): - *O* (Opportunities) – возможности; - *T* (Threats) – угрозы	Когда необходимо дать начальную оценку текущей ситуации, структурированное описание ситуации, относительно которой нужно принять какое-либо решение (например, оценка рисков перед выводом на рынок нового продукта, актуализация стратегии, принятие решения о расширении бизнеса)
PEST (PESTLE)-анализ	Техника долгосрочного планирования, позволяющая оценить внешнюю среду компании на основе оценки влияния на бизнес следующих факторов: *P* (Policy) – политическая ситуация, *E* (Economy) – состояние экономики, *S* (Society) – социальные факторы (темпы прироста населения, уровень миграции, образования, ценностей, вероисповедания), *T* **(**Technology) – технологические решения и тренды, оказывающие влияние на бизнес или продукт. PESTLE-анализ включает еще две дополнительные группы условий: *L* (Legal) –	- Для построения стратегии компании, чтобы понимать ее внешнее окружение, место на рынке, возможности и угрозы внешней среды; - для построения системы управления макроэкономическими рисками в компании

Продолжение таблицы 1

1	2	3
	правовые особенности, *E* (Ecological) – экологические условия	
VMOST (MOST)-анализ	Техника стратегического анализа, позволяющая контролировать и обеспечивать согласованность пяти внутренних элементов предприятия (видения, миссии, целей, стратегии, тактик) с целью достижения желаемого результата в процессе его деятельности: *M* (Mission) – миссия (сформированное утверждение цели и причины существования предприятия), *O* (Objectives) – цели (конкретные задачи, достижение которых необходимо для реализации миссии), *S* (Strategy) – стратегия (план для достижения целей), *T* (Tactics) – тактика (набор действий, необходимых для выполнения стратегии)	Для проведения детального анализа целей компании и способов их достижения
Метод CATWOE	Метод, позволяющий описать рыночное окружение и его ограничения в контексте планируемого изменения, направленного на решение бизнес-потребности (проблемы или новой возможности): *C* (Clients) – клиенты (как проблема влияет на клиентов?), *A* (Actors) – действующие лица (кто будет участвовать в реализации решений?), *T* (Transformation Process) – процесс трансформации (на какие процессы влияет проблема?), *W* (World View) – взгляд с высоты птичьего полета (какова общая картина и последствия этой проблемы?), *O* (Owner) – владелец (кому принадлежит процесс?), *E* (Environmental	Для выявления и решения реальной бизнес-проблемы

Продолжение таблицы 1

1	2	3
	Constraints) – ограничения среды (каковы ограничения, влияющие на решение?)	
GAP-анализ	GAP-анализ (от англ. gap – «разрыв») – техника, изучающая несоответствия, разрывы межу текущим и желаемым состояниями компании	Когда необходимо выделить проблемные зоны, препятствующие развитию компании и оценить степень ее готовности к выполнению перехода от текущего состояния к желаемому
MoSCoW-анализ	Метод приоритезации задач. Аббревиатура MoSCoW представляет четыре категории инициатив: *Must have* (должно быть), *Should have* (следовало бы иметь), *Could have* (могло бы быть), *Would like* (хотелось бы иметь в будущем)	Когда необходимо расставить приоритеты задач по четырем категориям инициатив в рамках любого ограниченного по времени проекта

Business Model Canvas. Топ-менеджеры компаний в современных рыночных условиях вынуждены постоянно искать новые бизнес-решения. На практике оказывается достаточно сложно быстро сгенерировать новые идеи, перевести любую идею в планы работ. Технология бизнес-моделирования А. Остервальдера и И. Пенье позволяет решить эти ключевые задачи. Бизнес-модель Остервальдера – один из самых популярных инструментов стратегического анализа и планирования.

В настоящее время сложившегося и общепринятого определения бизнес-модели не существует. Встречающиеся в литературе и интернет-источниках трактовки данного термина сводятся к тому, что бизнес-модель – это логическое и наглядное описание того, каким образом организация зарабатывает деньги, получает прибыль. Бизнес-модель служит для описания основных принципов создания, развития и успешной работы компании.

Будем придерживаться следующего определения бизнес-модели, предложенного А. Остервальдером, которое, на наш взгляд, наиболее полно и одновременно просто отражает его суть.

Бизнес-модель – это концептуальная модель бизнеса, которая иллюстрирует логику создания добавленной стоимости (прибыли).

Бизнес-модель компании позволяет ответить на следующие ключевые вопросы:

1 Кто основные потребители?

2 Какие продукты/услуги компания предоставляет потребителям?

3 Что компанию отличает от конкурентов?
4 Каким образом компания генерируем прибыль?
5 Какие проблемы решает продукт/услуга?

Бизнес-модель Остервальдера (Business Model Canvas) – это инструмент стратегического управления для новых или действующих компаний для структурированного отображения того, как бизнес создает ценность товаров.

Бизнес-модель Остервальдера – быстрый и эффективный способ представить бизнес-модель внутренним и внешним заинтересованным сторонам.
Business Model Canvas используется, в первую очередь, для анализа существующей бизнес-модели с целью нахождения слабых мест или новых точек роста. Ключевая особенность модели – фокусировка всех аспектов деятельности предприятия на клиенте и его потребностях.
Бизнес-модель Lean Canvas – это упрощенная модель Остервальдера, используемая стартап-проектами.

Шаблон Business Model Canvas состоит из девяти блоков, имеющих ключевое значение для бизнеса (рисунок 6).

<table>
<tr><td rowspan="2">Ключевые партнеры</td><td>Ключевые деятельности</td><td rowspan="2">Ценностные предложения</td><td>Взаимоотношения с клиентами</td><td rowspan="2">Потребительские сегменты</td></tr>
<tr><td>Ключевые ресурсы</td><td>Каналы сбыта</td></tr>
<tr><td colspan="3">Структура издержек</td><td colspan="2">Потоки доходов</td></tr>
</table>

Рисунок 6 – Шаблон Business Model Canvas

Перечислим данные блоки:

1 *Потребительские сегменты (customer segments)* – это группы потребителей, приносящие доход бизнесу.

2 *Ценностные предложения (value propositions)* – это предложения, которые решают некоторую проблему клиента, удовлетворяют его определенным потребностям. Это те преимущества, которые получит потребитель, воспользовавшись продуктом/услугой компании. Ценностное предложение может предлагать уникальное передовое (инновационное) решение какой-либо проблемы, а может быть адаптированным (аналог уже существующего, но улучшенного решения). Оно может отличаться для разных потребительских сегментов.

3 *Каналы сбыта (channels)* показывают, как компания взаимодействует с потребительскими сегментами и доносит до них свои ценностные предложения.

4 *Взаимоотношения с клиентами (customer relationships)* определяют характер отношений с клиентами в зависимости от решаемых компанией задач: приобретение и удержание клиентов, увеличение продаж.

5 *Потоки доходов (revenue streams)* определяют источники доходов компании. Это материальная прибыль, которую компания получает от каждого потребительского сегмента.

6 *Ключевые ресурсы (key resources)* – это наиболее важные активы, необходимые для функционирования бизнес-модели, позволяющие создавать и доносить до потребителя ценностные предложения.

7 *Ключевые деятельности (key activities)* отражают действия компании, которые необходимы для реализации ее бизнес-модели. Это те виды деятельности, без которых невозможна эффективная работа компании.

8 *Ключевые партнеры (key partners)* – это сеть поставщиков и партнеров, благодаря которым функционирует бизнес-модель.

9 *Структура издержек (cost structure)* – это расходы, связанные с функционированием бизнес-модели.

Все блоки можно разделить на две группы: блоки прибыльной части (потребительские сегменты, ценностные предложения, каналы сбыта, взаимоотношения с клиентами, потоки доходов) и блоки расходной части (ключевые ресурсы, ключевые деятельности, ключевые партнеры, структура издержек).

Описание ключевых блоков Business Model Canvas представлено в таблице 2. Здесь же приведен порядок заполнения блоков.

Таблица 2 – Ключевые блоки Business Model Canvas

Техники анализа бизнеса и его продукта	**Описание**	**Как использовать**
1	**2**	**3**
Потребительские сегменты	- Для каких клиентов создается ценностное предложение? - Какие клиенты наиболее важны? - К каким сегментам можно отнести?	Опишите все сегменты, на которые ориентирована деятельность компании (сегменты, которые в настоящем приносят доход бизнесу или могут принести в будущем). Основные типы потребительских сегментов рынка: массовый рынок, нишевой рынок, дробнонишевой рынок, многонишевой рынок
Ценностные предложения	- Почему потребители должны покупать услугу? - Какие ценности предлагаются потребителю? - Какие проблемы клиентов решаются и какие потребности закрываются?	Опишите ключевые качественные (известный бренд, инновационность, дизайн и др.) и количественные (цена, срок службы и др.) характеристики всех продуктов/ услуг

Продолжение таблицы 2

1	2	3
	- Что можно предложить каждому потребительскому сегменту?	
Каналы сбыта	- С помощью каких каналов охватываются потребительские сегменты? - Какие каналы взаимодействия были бы предпочтительны для потребительских сегментов? - Как связаны каналы между собой? - Какие каналы более бюджетны и эффективны?	- Изучите модели конкурентов, оцените их возможности и выберите наиболее удобный для потребителя канал сбыта; - опишите все возможные каналы сбыта (прямые или через посредников, онлайн или офлайн, розничные или оптовые и др.); - не забывайте, что выделенные каналы сбыта должны повышать осведомленность потребителей о новом продукте/услуге, способствовать их положительной оценке, обеспечивать возможность их легкого и быстрого приобретения; - важно помнить, что каналы взаимодействия должны помогать с решением проблем и после покупки
Взаимоотношения с клиентами	- Отношения какого формата ожидает получить потребитель на каждом этапе взаимодействия? - Соответствуют ли эти отношения выбранной бизнес-модели? - Какие отношения установлены и каких расходов требуют?	- Определите текущие задачи бизнеса и в зависимости от этого выстраивайте типы отношений и взаимодействий с клиентами; - оцените отношения с потребителями: насколько отношения доверительны, качественны и долгосрочны, существует ли обратная связь, оказывается ли им постпродажная поддержка
Потоки доходов	- За какие функции продукта/услуги клиенты готовы платить? - Каким способом клиенты будут оплачивать продукт? Соответствует ли он их ожиданиям?	- Определите все доходы, которые получаете (продажа товаров, активов, плата за использование услуги, оплата подписки, аренда (лизинг), лицензия, реклама и др.);

Продолжение таблицы 2

1	2	3
	- Какой вклад в бизнес вносит каждый поток поступления доходов?	- разбейте их по типам потоков доходов: постоянные доходы с фиксированной ценой и доходы от разовых сделок. Можно, например, делить по группам клиентов или типам продаж
Ключевые ресурсы	Какие ключевые ресурсы необходимы для создания и реализации ценностных предложений?	Составьте список источников ресурсов, которые помогают доносить до клиентов ценностное предложение, выстраивать взаимоотношения и получать прибыль от деятельности
Ключевые деятельности	Какие ключевые деятельности (действия) необходимы для поддержания ценности продукта/услуги?	Выделите главные направления деятельности бизнеса (производство, решение проблем, управление инфраструктурой, технология, продажи и маркетинг), благодаря которым реализуется продажа товара и формируется ценностное предложение
Ключевые партнеры	- Какие партнеры наиболее значимы для бизнеса? - Какие выгоды и преимущества обеспечивают партнерские отношения? - Какие виды деятельности выполняют? Какие ключевые ресурсы поставляют? - Насколько партнеры стабильны? Доверительные ли взаимоотношения?	- Перечислите и опишите партнеров (рекламные агентства, поставщики, консультанты, фрилансеры, аутсорсинговые компании и др.), с которыми взаимодействуете на постоянной основе для создания ценностного предложения, и в каких направлениях сотрудничаете (оптимизация и экономия в сфере производства, снижение риска и неопределенности, поставка ресурсов, совместная деятельность); - типы партнерских отношений: стратегическое партнерство между неконкурирующими компаниями, стратегическое партнерство между конкурентами (соконкуренция), совместные предпри-

Продолжение таблицы 2

1	2	3
		ятия для запуска новых бизнес-проектов, отношения производителя с поставщиками для гарантии получения качественных комплектующих
Структура издержек	- Какие расходы для бизнеса самые значимые? - Какая деятельность требует наибольших расходов?	- Необходимо учесть все значительные издержки (переменные и фиксированные), которые несет бизнес при создании ценностных предложений. При этом учитывайте заполненные ранее блоки по ключевым ресурсам, основным видам деятельности и партнерам; - имеет смысл по структуре издержек разделить бизнес-модели на два класса: с преимущественным вниманием к издержкам и с преимущественным вниманием к ценности

Перед разработкой Business Model Canvas следует определить бизнес, который нужно моделировать. Компания может работать в нескольких направлениях деятельности, что не позволяет создать четкую бизнес-модель.

Подробнее о техниках анализа бизнеса и его продукта читайте в дополнительных источниках:

1 Johnson, Mark W. Rein-venting Your Business Model / Mark W. Johnson, Clayton M. Christensen, Henning Kagermann // Harvard Business Review. – 2008. – № 12.

2 PEST-анализ: что это такое и как его провести на примерах [Электронный ресурс]. – Режим доступа: https://upr.ru/article/pest-analiz-chto-eto-takoe-i-kak-ego-provesti-na-primerah.

3 Анализ рынка: классика стратегического планирования [Электронный ресурс]. – Режим доступа: https://kachestvo.pro/kachestvo-uprav-leniya/instrumenty-menedzhmenta/analiz-rynka-klassika-strategichesko-go-planirovaniya/.

4 Бизнес-модель Остервальдера и Lean Canvas: неклассические подходы планирования [Электронный ресурс]. – Режим доступа: https://kachestvo.pro/kachestvo-upravleniya/instrumenty-menedzhmenta/biznes-model-osterval-dera-i-lean-canvas-neklassicheskie-podkhody-planirovaniya.

5 Бизнес-модель Остервальдера: что это такое? [Электронный ресурс]. – Режим доступа: https://habr.com/ru/companies/productstar/articles/508926/.

6 Гассман, О. Бизнес-модели. 55 лучших шаблонов / О Гассман, К. Франкенбергер, М. Шик. – М. : Альпина Пабли-шер, 2017. – 432 с.
7 Остервальдер А. Построение бизнес-моделей. Настольная книга стратега и новатора / А. Остервальдер, И. Пинье. – М. : Альпина Паблишер, 2016. – 288 с.
8 Построение бизнес-модели. Lean Canvas [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=kbP_DF3q3fE.

Краткие теоретические сведения и методические указания к заданию 2

Бизнес-требования являются краеугольным камнем успешной разработки программного обеспечения, их грамотное определение играет решающую роль в достижении целей проекта.

BABOK®Guide определяет **бизнес-требования (business requirements)** как «представление целей, задач и результатов, объясняющих, зачем было инициировано изменение и как будет оцениваться успех».
Бизнес-требования описывают, *что* является целью проекта, *в каком объеме* и *каким образом* цель проекта связана с достижением бизнес-цели компании.
Бизнес-требования – это высокоуровневая бизнес-цель организации или заказчиков системы.

Бизнес-требования описывают потребности компании в целом, а не отдельных стейкхолдеров.

Как правило, бизнес-требования разрабатываются и выявляются на этапе анализа предприятия. Рекомендации к формированию бизнес-требований представлены на рисунке 7.

Результатом выявления бизнес-требований является Документ о концепции и границах (vision and scope document). **Документ о концепции и границах (vision and scope document)** – это высокоуровневое представление требований. Документ служит для обобщения долгосрочных целей и определения назначения разрабатываемого продукта.

Концепция продукта (product vision) кратко описывает конечный продукт, который достигнет заданных бизнес-целей, т. е. что продукт представляет собой сейчас и каким он станет впоследствии.

Отвечают на вопрос	*Что* мы делаем, *почему* и *зачем*?
Пример	Разработать сайт интернет-магазина по торговле спортивной обувью с целью увеличения объема продаж на 30 % в течение 18 месяцев после его запуска
Источники бизнес-требований (где искать?)	▪ Анализ рынка; ▪ изучение бизнес-процессов; ▪ общение с заказчиком; ▪ документация (внутренняя, по предметной области, нормативная); ▪ продукты конкурентов
Стейкхолдеры (у кого спрашивать?)	▪ Конечные пользователи; ▪ эксперты предметной области; ▪ спонсор; ▪ менеджер проекта; ▪ инициатор проекта
Лайфхаки по разработке бизнес-требований	▪ Систематический анализ: разбивка бизнес-требований на более мелкие компоненты упрощает их понимание и внедрение; ▪ прототипирование: ускоряет процесс визуализации требований и позволяет заказчику лучше представить конечный продукт; ▪ итерационный подход: регулярные обновления и обсуждения с заказчиком позволяют адаптировать бизнес-требования в соответствии с изменяющимися условиями

Рисунок 7 – Рекомендации к формированию бизнес-требований

Рекомендуемый шаблон для описания концепции продукта (видение решения)
Для <целевая аудитория>,
которая <описание потребностей или возможностей>,
этот продукт <имя разрабатываемого продукта>
является <категория продукта>,
который <ключевое преимущество, основная причина для использования>.
В отличие от <основные конкуренты, текущая система или бизнес-процесс>,
наш продукт <перечисление основных отличий и преимуществ разрабатываемого продукта>.

ПРИМЕР ОПИСАНИЯ КОНЦЕПЦИИ ПРОДУКТА

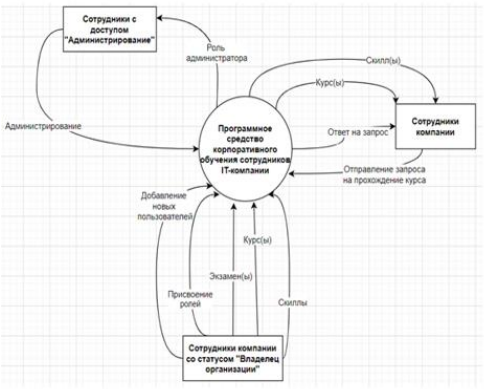
Для сотрудников, **которым** нужно заказывать еду в кафе, **данная** Cafe Ordering System **является** приложением для смартфона, **которое** принимает индивидуальные или групповые заявки на питание, взимает оплату и инициирует доставку готовых блюд к указанной компании. **В отличие от** имеющихся в настоящее время служб заказа по телефону **наш продукт** позволит сотрудникам не приходить в кафетерий, чтобы получать заказанные блюда, что сэкономит им время, и, кроме того, увеличится ассортимент доступных им блюд [4].

Границы проекта (project scope) показывают, на какую часть конечной концепции продукта будет направлен текущий проект или итерация. Определение границ будущей системы является одним из ключевых аспектов формирования пользовательских требований. По сути, границы проекта определяют, что продукт должен делать, а что не должен.

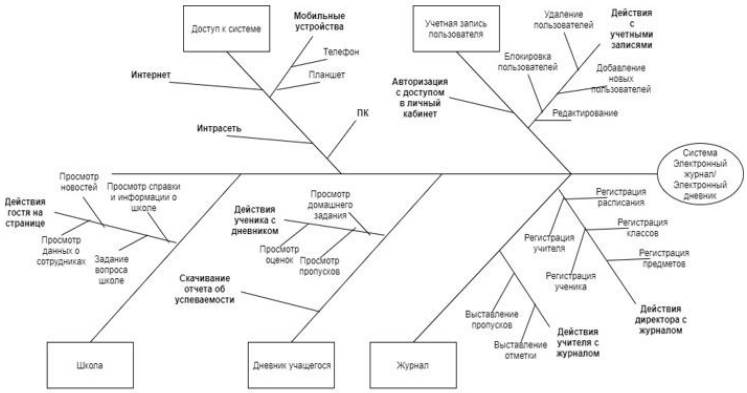
Отсутствие четких границ проекта очень часто приводит к разрастанию функциональности, а следовательно, и к увеличению сроков выполнения проекта.

Для моделирования границ проекта чаще всего используют следующие инструменты (рисунок 8):

- контекстную диаграмму, которая иллюстрирует границу и связи разрабатываемого программного решения с внешним миром (пользователи, устройства, другие системы). Данная диаграмма является наиболее высокоуровневым способом очертить границы решения;
- дерево функций, которое отражает в сжатом виде все запланированные к реализации в проекте функции.



Контекстная диаграмма



Дерево функций

Рисунок 8 – Способы представления границ проекта

Рекомендуемый шаблон документа о концепции и границах проекта из книги К. Вигерса представлен ниже.

1 Бизнес-требования
- 1.1 Исходные данные
- 1.2 Возможности бизнеса
- 1.3 Бизнес-цели
- 1.4 Критерии успеха
- 1.5 Видение решения
- 1.6 Бизнес-риски
- 1.7 Предположения и зависимости

2 Рамки и ограничения проекта
- 2.1 Основные функции
- 2.2 Объем первоначально запланированной версии
- 2.3 Объем последующих версий
- 2.4 Ограничения и исключения

3 Бизнес-контекст
- 3.1 Профили заинтересованных лиц
- 3.2 Приоритеты проекта
- 3.3 Особенности развертывания

Следует заметить, что данный шаблон не является эталонным и зависит от ряда факторов, включая потребности бизнеса заказчика, сложность проекта и объем работы, навыки команды.

В рамках лабораторной работы следует представить **Документ о концепции и границах** следующей структуры:

1 Бизнес-требования
  1.1 Исходные данные
  1.2 Возможности бизнеса
  1.3 Бизнес-цели
  1.4 Видение решения
2 Рамки и ограничения проекта
  2.1 Основные функции
  2.2 Бизнес-правила
3 Бизнес-контекст
  3.1 Профили заинтересованных лиц

При выполнении данного задания следует ознакомиться с пояснительным материалом, представленным на страницах 91–110 источника [4], а также можно воспользоваться примером оформления, приведенным в приложении В данного источника.

Примечание – В подразделе 2.1 «Основные функции» следует описать первоначальный перечень функций программного средства, уделив внимание обзору функциональности конкурирующих аналогов, а также представить частичное дерево функций или контекстную диаграмму. В процессе разработки базовых пользовательских требований (вариантов использования) первоначальный перечень функций будет уточняться.

Обзор функциональности аналогов. Заключается в систематизации функций аналогов с целью разработки программного средства, привлекательного для конечного пользователя.

При выполнении обзора функциональности аналогов необходимо:

- выбрать для обзора не менее трех аналогов;
- желательно сконцентрироваться на актуальных аналогах, входящих в ТОП (на основе материалов аналитических агентств, исследовательских и консалтинговых компаний, специализирующихся на рынках информационных технологий, например, Tadviser, Gartner, IDC и др.);
- представить результаты обзора (для документирования результатов обзора может быть использован шаблон, представленный *таблицей 3*).

Таблица 3 – Аналоги для [указать предметную область] и их функциональность

Характеристика аналога	Значение
Название аналога	
Компания-разработчик	
URL официального сайта компании-разработчика	
Назначение аналога	
URL страницы с представленной функциональностью	
Функциональность [указать название аналога]	

ПРИМЕР ОПИСАНИЯ АНАЛОГА

Пример документирования результатов обзора представлен в таблице 4.

Таблица 4 – Аналоги для управления рекрутингом и их функциональность

Характеристика аналога	Значение
Название аналога	Mirapolis HCM
Компания разработчик	ООО «Мираполис»
URL официального сайта компании-разработчика	https://www.mirapolis.ru
Назначение аналога	Автоматизация полного цикла подбора с момента появления потребности до выхода на работу нового сотрудника
URL страницы с представленной функциональностью	https://www.mirapolis.ru/recruit/
Функциональность Mirapolis HCM	
Единая база и удобный поиск: - единая база для внешнего и внутреннего подбора; - быстрая загрузка резюме из электронной почты или работных сайтов с помощью плагинов; - разграничение доступа для управления видимостью кандидатов и отдельных событий; - гибкий поиск и фильтры (например, по опыту работы, образованию, навыкам и знаниям); - теги, группировки кандидатов по разным критериям	
Распознавание резюме: - резюме с job-сайтов; - e-mail сообщения; - файлы в формате .docx или .pdf; - любой неструктурированный текст, например, из соцсетей или мессенджеров	
Тесты и оценка кандидатов - проведение тестов или практических заданий на любом этапе подбора; - оценка кандидатов по навыкам и компетенциям; - формирование средней оценки несколькими согласующими; - рейтинг кандидатов на основе результатов оценки и тестов; - хранение истории тестов и оценок кандидатов	

Продолжение таблицы 4

Личные кабинеты: - проведение тестов или практических заданий на любом этапе подбора; - кабинет заказчика: список кандидатов с указанием этапов воронки подбора, запланированные и проведенные интервью, форма обратной связи о кандидате, рейтинг кандидатов, комментарии для рекрутера, согласование условий найма; - кабинет рекрутера: настраиваемые фильтры кандидатов по этапам воронки подбора и вакансиям, звонки и отправка сообщений кандидату с основного экрана, комментирование, присвоение тегов, просмотр важной информации без переключения в карточку кандидата, запланированные действия; - кабинет руководителя подбора: согласование и распределение вакансий, индикация сроков по вакансиям и действиям, отображение дедлайнов, KPI и загрузки рекрутеров; - кабинет кандидата: информация о вакансиях, подача заявки, доступ к необходимым материалам, прохождение тестирования, обмен документами (оффер, анкета кандидата, документы для оформления)
Массовый подбор: - автоматический отбор кандидатов по заданным критериям; - массовые операции над кандидатами – приглашения на интервью, проведение тестирования и опросов; - интеграция с голосовыми и чат-ботами, системами видеоинтервью, автодозвона до кандидатов
Настройка воронки подбора: - настройка маршрута согласования заявки на подбор; - последовательное или параллельное согласование; - разные процессы согласования заявок (например, на штатную и внештатную должность, с отклонением от установленной зарплаты и без него); - создание и изменение собственных этапов воронки подбора и сроков их прохождения; - фиксированные этапы воронки или с возможностью выбора следующего при согласовании; - создание и настройка уведомлений к любым этапам и событиям подбора
Аналитика: - библиотека готовых отчетов и возможность создания собственных; - дашборды по ключевым показателям; - наиболее эффективные источники подбора; - эффективность команды подбора и каждого участника; - скорость закрытия вакансий, стоимость подбора
Возможность интеграции: - плагины для парсинга резюме из электронной почты и при поиске на job-сайтах; - интеграция со всеми job-сайтами, интеграция с IP-телефонией и электронной почтой; - интеграция с учетными системами, такими как lC, SAP, Босс Кадровик, Парус; - широкие возможности API позволяют интегрироваться с чат-ботами, системами для проведения видеоинтервью, автодозвонами и другими системами

Бизнес-правила. С бизнес-требованиями тесно связаны бизнес-правила. Бизнес-правила накладывают ограничения на те или иные аспекты бизнеса, которые должны быть реализованы в программном решении и являются основой для разработки требований.

Бизнес-правило (business requirements) – это положение, определяющее или ограничивающее какие-либо стороны бизнеса.

Бизнес-правила не являются требованиями к программному обеспечению. Бизнес-правила описывают особенности принятых в предметной области и/или непосредственно у заказчика процессов, ограничений и иных правил. Данные правила могут относиться к бизнес-процессам, правилам работы сотрудников, особенностям работы программного обеспечения и т. д. Это могут быть корпоративные регламенты, политики, стандарты, законодательные акты, которые подразумевают организацию структуры бизнеса, контролируют ведение бизнеса или влияют на бизнес.

Классификация бизнес-правил приведена в таблице 5.

Таблица 5 – Классификация бизнес-правил

Тип бизнес-правила	**Определение правила**	**Пример**
Факт	Верное утверждение о бизнесе	Интернет-магазин спортивной обуви предлагает доставку заказов по всей Беларуси
Ограничение	Устанавливает, какие операции не могут выполнять система и ее пользователи	Доставка заказов по Беларуси осуществляется с 8:00 до 23:00 ежедневно
Активатор	Инициирует при определенных условиях выполнение определенных действий	Ежемесячно 1-го числа всем подписчикам интернет-магазина рассылается информация о новых поступлениях спортивной обуви
Вычисление	Определяет вычисления с использованием математических формул и алгоритмов	При заказе товара на сумму покупки менее 500 р., к стоимости заказа добавляется стоимость доставки
Вывод	Создает новый факт на основе других фактов	Если интернет-магазин не может осуществить доставку товара в течение пяти дней с момента заказа, заказ аннулируется
Термин	Определяет важные для бизнеса понятия и сокращения	Уведомление о времени прибытия курьера – это уведомление клиентов с помощью SMS

Рекомендации к формированию бизнес-правил представлены на рисунке 9.

Для документирования бизнес-правил может быть использован шаблон, представленный таблицей 6.

Отвечают на вопрос	*Какие* ограничения накладываются на бизнес, на разрабатываемое программное решение?
Пример	Интернет-магазин по торговле спортивной обувью может обрабатывать заказы покупателей только до 21:00
Источники правил (где искать?)	▪ Нормативные и законодательные акты; ▪ регламенты, инструкции, предписания; ▪ документация бизнес-процессов
Стейкхолдеры (у кого спрашивать?)	▪ Конечные пользователи; ▪ эксперты предметной области; ▪ спонсор; ▪ менеджер проекта; ▪ инициатор проекта
Бизнес-правила влияют на	▪ Бизнес-требования; ▪ пользовательские требования; ▪ функциональные требования; ▪ атрибуты качества

Рисунок 9 – Рекомендации к формированию бизнес-правил

Таблица 6 – Шаблон для документирования бизнес-правил

Идентификатор	**Определение правила**	**Тип правила**	**Статическое или динамическое**	**Источник**
BR-1				
BR-2				

Профили заинтересованных лиц. Проект – это временное предприятие с уникальным результатом и большой, как правило, неопределенностью на старте.

Для снижения уровня неопределенности руководитель проекта должен предвидеть возможные риски.

Очень часто источниками проблем становятся пропущенные или неправильно определенные ожидания не только заказчика, но и других заинтересованных сторон (лиц).

Заинтересованная сторона (stakeholders) – лицо, группа или организация, которая может влиять, на которую могут повлиять или которая может воспринимать себя подвергнутой влиянию решения, операции или результата проекта.

Информация об ожиданиях стейкхолдеров, их потребностях, требованиях, об отношении, интересу к проекту влияет на такие аспекты разработки программного обеспечения, как сбор и анализ требований.

Одна из ключевых задач проекта – выявить и проанализировать заинтересованные стороны, чтобы адекватно их вовлечь в проект.

Наиболее распространенные техники выявления и анализа заинтересованных лиц приведены ниже.

Техники выявления и анализа заинтересованных лиц: матрица заинтересованных лиц, Stakeholder map, луковичная диаграмма, матрица RACI, матрица интересов и влияния, матрица оценки уровня вовлечения стейкхолдеров.

Рекомендуемая последовательность действий при описании профилей заинтересованых лиц:

- определить основные заинтересованные лица и их ключевые интересы (заполнить таблицы 7, 8);
- с помощью матрицы интересов и влияния проанализировать интересы и влияние каждого стейкхолдера на проект, а также сформулировать стратегию взаимодействия с каждой группой;
- составить матрицу RACI для заинтересованных лиц проекта;
- на основе проведенных исследований сформировать профили заинтересованных лиц в виде реестра заинтересованных лиц в соответствии с таблицей 9.

Таблица 7 – Заинтересованные лица

Заинтересованное лицо	**Описание**	**В чем заинтересовано**

Таблица 8 – Ключевые подходы к идентификации заинтересованных сторон

Название техники	**Суть техники**	**Недостаток техники**

Таблица 9 – Реестр заинтересованных сторон

Поле	**Алгоритм заполнения**
ID	Уникальный идентификатор требования, например, st-1
Имя	Фамилия и имя заинтересованного лица
Роль в проекте	Проектная роль (пользователь, эксперт, спонсор, член команды и т. п.)
Должность	Занимаемая заинтересованным лицом должность
Отдел/подразделение	Отдел или подразделение, где работает заинтересованное лицо
Непосредственный начальник	Прямой начальник заинтересованного лица
Контактная информация	Телефон, e-mail и пр.
Предпочитаемый вид коммуникаций	Электронная почта/телефон/онлайн-конференция и т. п.
Главные ожидания	Главные ожидания заинтересованного лица по проекту
Главные требования	Главные требования заинтересованного лица по проекту
Влияние на проект	Влияние на проект в баллах по шкале 1–10 (где 1 – минимальное влияние, 10 – максимальное влияние)
Отношение к проекту	Противник/сторонник/нейтрал
Интерес к проекту	Возможно, заинтересованное лицо *хочет* принять участие в проекте как эксперт или в иной роли
Комментарий	Необходимые комментарии

Подробнее о бизнес-требованиях, Документе о концепциях и границах проекта читайте в дополнительных источниках:

1 Анализ бизнес-правил: техника BABOK®Guide для документирования операций и разработки требований [Электронный ресурс]. – Режим доступа: https://babok-school.ru/blogs/business-rules-analysis-babok-technique.

2 Бизнес-правила и требования к системе [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/659045.

3 Бизнес-требования. Назначение [Электронный ресурс]. – Режим доступа: https://systems.education/biz-req.

4 В переводе с клиентского: как аналитик пишет бизнес-требования [Электронный ресурс]. – Режим доступа: https://practicum.yandex.ru/blog/biznes-trebovaniya.

Контрольные вопросы к лабораторной работе № 1

1 Какие существуют уровни требований?

2 В чем заключается анализ бизнеса и его продукта и какие техники анализа используют?

3 Что такое бизнес-модель?

4 Для чего используют Business Model Canvas и как с ней работать?

5 Когда целесообразно использовать метод SMART?

6 Что представляет собой Документ о концепции и границах и когда его следует разрабатывать?

7 Что такое бизнес-контекст?

8 Какие типы бизнес-правил выделяют?

9 В чем отличия бизнес-правил от требований к системе?

10 Кто такие заинтересованные лица и зачем нужен их анализ?

11 Как выявить и проанализировать ключевые заинтересованные стороны, имеющие отношение к проекту по разработке программного обеспечения?

12 Как сформировать профили заинтересованных лиц?

13 Что такое матрица RACI и когда ее следует использовать?

14 На каком этапе определяются бизнес-требования?

ЛАБОРАТОРНАЯ РАБОТА № 2
Экономическое обоснование проектного решения

Цель выполнения лабораторной работы: определить экономическую целесообразность разработки проектного решения.

Задание	**Этапы выполнения задания**
1 Представить экономическое обоснование проектного решения	Выполнить экономическое обоснование проектного решения по разработке программного средства по одному из типовых вариантов: - вариант 1 – по индивидуальному заказу в рамках аутсорсинговой модели; - вариант 2 – в виде продукта для массового рынка (продуктовая модель); - вариант 3 – для внедрения в собственные бизнес-процессы разработчика
2 Сформировать отчет по лабораторной работе	*Содержание отчета*: Титульный лист (приложение Б) 1 Экономическое обоснование разработки программного средства 2 Выводы

Краткие теоретические сведения и методические указания к заданию 1

В рамках лабораторной работы все проектные решения по разработке программного средства сведены к трем типовым вариантам:

Вариант 1. По индивидуальному заказу в рамках аутсорсинговой модели.

Вариант 2. В виде продукта для массового рынка (продуктовая модель).

Вариант 3. Для внедрения в собственные бизнес-процессы разработчика.

Осуществить экономическое обоснование разработки программного средства только **для одного типового варианта**.

Для полноценного экономического обоснования разработки программного средства необходимо четко определить следующие аспекты, важные для коммерциализации проекта [5]:

1 Кто является конечным пользователем программного средства?

2 Какую бизнес-проблему пользователя будет решать программное средство?

3 Какие существуют конкурентные программные аналоги?

4 Почему разрабатываемое программное средство более предпочтительно для пользователя по сравнению с существующими конкурентными программными аналогами?

Методические указания по выполнению экономического обоснования разработки программного средства подробно представлены в источнике [5].

В отчете по лабораторной работе раздел **Экономическое обоснование разработки программного средства** в зависимости от выбранного типового варианта необходимо представить в одной из нижеследующих редакций.

Вариант 1

1 ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ И ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО СРЕДСТВА ПО ИНДИВИДУАЛЬНОМУ ЗАКАЗУ

1.1 Характеристика разработанного по индивидуальному заказу программного средства.

1.2 Расчет затрат на разработку и цена программного средства, созданного по индивидуальному заказу.

1.3 Расчет результата от разработки и использования программного средства, созданного по индивидуальному заказу.

1.4 Расчет показателей экономической эффективности разработки и использования программного средства.

Вариант 2

1 ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ И РЕАЛИЗАЦИИ ПРОГРАММНОГО СРЕДСТВА НА МАССОВОМ РЫНКЕ

1.1 Характеристика программного средства, разрабатываемого для реализации на рынке.

1.2 Расчет инвестиций в разработку программного средства для его реализации на рынке.

1.3 Расчет экономического эффекта от реализации программного средства на рынке.

1.4 Расчет показателей экономической эффективности разработки и реализации программного средства на рынке.

Вариант 3

1 ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ И ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО СРЕДСТВА ДЛЯ СОБСТВЕННЫХ НУЖД

1.1 Характеристика программного средства, разрабатываемого для собственных нужд.

1.2 Расчет инвестиций в разработку программного средства для собственных нужд.

1.3 Расчет экономического эффекта от использования программного средства для собственных нужд.

1.4 Расчет показателей экономической эффективности разработки и использования программного средства в организации.

Особый интерес представляет вариант 2 в случае использования таких моделей монетизации программного средства, как комиссия от продаж и размещение рекламы. Пример такого варианта экономического обоснования разработки и реализации программного продукта на массовом рынке представлен ниже.

ПРИМЕР ЭКОНОМИЧЕСКОГО ОБОСНОВАНИЯ РАЗРАБОТКИ И РЕАЛИЗАЦИИ ПРОГРАММНОГО СРЕДСТВА НА МАССОВОМ РЫНКЕ

1.1 ХАРАКТЕРИСТИКА ПРОГРАММНОГО СРЕДСТВА, РАЗРАБАТЫВАЕМОГО ДЛЯ РЕАЛИЗАЦИИ НА РЫНКЕ

Цель исследования – разработка программного средства для монетизации и обмена знаниями при осуществлении ремесленной деятельности. Программное средство является мобильным приложением для платформ Android и iOS и адресовано такой категории пользователей как ремесленники, чья деятельность связана с изготовлением вязаных изделий.

С внедрением программного средства ремесленники получат:

- возможность расширить или перенести свой бизнес в сеть Интернет с целью получения прибыли от продвижения продуктов ремесленной деятельности;
- удобную среду для осуществления процесса купли-продажи либо бесплатного приобретения результатов ремесленной деятельности без прямого взаимодействия с продавцом.

Основная функциональность программного средства:

- создание и предоставление базы паттернов для их покупки или бесплатного приобретения;
- использование широкого набора опций по управлению паттерном;
- отзывчивый дизайн паттерна;
- формирование сопроводительного листа с информацией о паттерне;
- импортирование паттернов в различные форматы;
- отображение рейтинга паттерна и комментариев к нему;
- возможность публикации паттерна;
- использование настраиваемых фильтров.

Требования к разработчикам программного средства: знание Spring Framework для создания приложений на языке программирования Java, библиотеки React.js, СУБД PostgreSQL.

Пандемия COVID-19 особенно усилила интерес к данной теме, когда многие люди начали изучать новые хобби, в том числе вязание, и переносить свои увлечения в сеть Интернет. Актуальность программного средства обусловлена

ростом заинтересованности пользователей в приобретении паттернов для изготовления вязаных изделий как в качестве хобби, так и с целью развития рукодельного бизнеса.

Популярными конкурентными аналогами программного средства являются такие платформы для творчества, как Craftsy и Etsy. Уникальная особенность разработанного программного средства – возможность использования функции «живого паттерна», которая позволяет зачеркивать уже связанные ряды во время вязания.

Используемые модели монетизации программного средства – комиссия (процент) от продаж и размещения рекламы. Каналы продажи программного продукта – Google Play и App Store. Продвижение программного продукта происходит через рекламу в социальной сети Instagram.

Экономическая целесообразность инвестиций в разработку и реализацию платформы осуществляется на основе расчета и оценки следующих показателей:

- прибыль;
- рентабельность инвестиций;
- срок окупаемости.

Программное средство предназначено для реализации на массовом рынке.

1.2 РАСЧЕТ ИНВЕСТИЦИЙ В РАЗРАБОТКУ ПРОГРАММНОГО СРЕДСТВА ДЛЯ ЕГО РЕАЛИЗАЦИИ НА РЫНКЕ

Инвестициями для организации-разработчиков программного средства являются затраты на его разработку. В затраты входит основная заработная плата разработчиков, которая вычисляется по формуле

$$З_о = К_{пр} \sum_{i=1}^{n} З_{чi} t_i, \quad (1)$$

где $К_{пр}$ – коэффициент премий и иных стимулирующих выплат (по данным предприятия);

n – категории исполнителей, занятых разработкой программного средства;

$З_{чi}$ – часовой оклад исполнителя i-й категории, р.;

t_i – трудоемкость работ, выполняемых исполнителем i-й категории (определяется исходя из сложности разработки программного обеспечения и объема выполняемых им функций), ч.

Чтобы рассчитать зарплаты специалистов, задействованных в разработке данного программного средства, использована статистика медианы зарплат в IT-сфере за месяц, далее, исходя из 21 рабочего дня в месяц и сорокачасовой рабочей недели рассчитан оклад в час.

Трудоемкость работ указана исходя из среднего времени, затраченного на проекты схожей сложности. Чтобы рассчитать полный заработок каждого специалиста за программное средство, необходимо вычислить произведение часового оклада и трудоемкости работ. Суммарная зарплата разработчиков на изго-

товление программного средства составляет сумму зарплат каждого специалиста. Премия и иные стимулирующие выплаты рассчитаны как 20 % от общей итоговой заработной платы.

Таким образом, общее количество затрат на основную заработную плату разработчиков вычислено как сумма итоговой общей заработной платы специалистов, премий и иных стимулирующих выплат и представлено в таблице 10.

Таблица 10 – Расчет затрат на основную заработную плату команды разработчиков

Категория исполнителя	**Месячный оклад, р.**	**Часовой оклад, р.**	**Трудоемкость работ, ч**	**Итого, р.**
Программист (Full-stack Java and React.js)	1428	8.5	252	2142
Дизайнер	1050	6.25	80	500
Бизнес-аналитик	1050	6.25	40	250
Итого				2892
Премия и иные стимулирующие выплаты (по данным предприятия 50 %)				1421
Всего затрат на основную заработную плату разработчиков				4313

Общее количество затрат на дополнительную плату разработчиков вычисляется по формуле

$$З_д = \frac{З_о Н_д}{100}, \quad (2)$$

где $Н_д$ – норматив дополнительной заработной платы (по данным предприятия – 20 %).

Вычислим дополнительную заработную плату:

$$З_д = \frac{4313 \cdot 20}{100} = 862.6 \text{ р.}$$

Отчисления на социальные нужды вычисляются по формуле

$$Р_{соц} = \frac{(З_о + З_д) Н_{соц}}{100}, \quad (3)$$

где $Н_{соц}$ – норматив отчислений в ФСЗН и Белгосстрах (в соответствии с действующим законодательством – 34.6 %).

Вычислим отчисления на социальные нужды:

$$Р_{соц} = \frac{(4313 + 862.6) \cdot 34.6}{100} = 1790.8 \text{ р.}$$

Прочие расходы вычисляются по формуле

$$Р_{пр} = \frac{З_о Н_{пр}}{100}, \quad (4)$$

где $Н_{пр}$ – норматив прочих расходов (по данным предприятия – 30 %).

Вычислим прочие расходы:

$$Р_р = \frac{4313 \cdot 30}{100} = 1293.9 \text{ р.}$$

Расходы на реализацию вычисляются по формуле

$$Р_{пр} = \frac{З_о Н_р}{100}, \quad (5)$$

где $Н_р$ – норматив расходов на реализацию (по данным предприятия или 3 %).

Вычислим расходы на реализацию:

$$Р_р = \frac{4313 \cdot 3}{100} = 129.4 \text{ р.}$$

Общая сумма затрат на разработку и реализацию вычисляется по формуле

$$З_р = З_о + З_д + Р_{соц} + Р_{пр} + Р_р. \quad (6)$$

Вычислим общую сумму затрат на разработку и реализацию:

$$З_р = 4313 + 862.6 + 1790.8 + 1293.9 + 129.4 = 8389.7 \text{ р.}$$

С расчетами затрат на разработку программного средства можно ознакомиться в таблице 11.

Таблица 11 – Расчет затрат на разработку программного средства, предназначенного для продажи

Наименование статьи затрат	**Значение, р.**
Основная заработная плата разработчиков ($З_О$)	4313
Дополнительная заработная плата разработчиков ($З_д$)	862.6
Отчисления на социальные нужды ($Р_{соц}$)	1790.8
Прочие расходы ($Р_{пр}$)	1293.9
Расходы на реализацию ($Р_р$)	129.4
Общая сумма затрат на разработку и реализацию ($З_р$)	8389.7

В результате общая сумма затрат на разработку и реализацию программного средства равна 8389.7 р.

1.3 РАСЧЕТ ЭКОНОМИЧЕСКОГО ЭФФЕКТА ОТ РЕАЛИЗАЦИИ ПРОГРАММНОГО СРЕДСТВА НА РЫНКЕ

Экономический эффект от реализации программного средства представляет собой прирост чистой прибыли от продаж. Его величина зависит от количества продаж и комиссии от средней выручки.

На белорусском рынке продажа паттернов через приложения является распространенным видом бизнеса, однако из-за санкций многие приложения, например, Etsy, стали недоступны. В связи с этим многие ремесленники перешли на социальную сеть Instagram, которая не является платформой для продаж, следовательно, не имеет возможности контролируемого расчета за паттерн (покупка происходит через личные сообщения и перечисление средств на карту), что снижает безопасность во время его покупки. Кроме того, не предусмотрены возможности просмотра покупателями рейтинга и отзывов на паттерны, возможности отслеживания продавцами заработка от продаж. Все вышеперечисленные особенности затрудняют осуществление электронного бизнеса в сфере продажи паттернов. Разработанное программное средство включает указанную функциональность, а также имеет дополнительные функции, например, возможность зачеркивания готовых строк во время вязания, что может сыграть ключевую роль для того, чтобы приложение стало лидирующей платформой электронной коммерции в сфере продаж паттернов.

Для продвижения выбран метод «сарафанного радио»: через личные сообщения в сети Instagram создатели приложения рассказывают о нем ремесленникам. И т. к. регистрация бесплатна, а функциональность приложения предлагает повышение безопасности и удобства в процессе покупки паттернов, такая реклама привлекает большое число пользователей.

Ожидается, что за первый месяц в приложении зарегистрируются 250 пользователей. Увеличение количества пользователей станет рекламой для еще не зарегистрированных пользователей. Прибыль от реализации программного средства получена от комиссии с продаж паттернов. Ожидается, что за первый месяц паттерны будут проданы 400 раз. Средняя цена паттерна определена на основе анализа аналогичных платформ для продажи схем и равна 15 р., комиссия от продажи паттерна равна 12 %.

Средняя выручка от одного паттерна вычисляется по формуле

$$\text{СВ} = \frac{\text{ЦК}}{100}, \tag{7}$$

где СВ – средняя выручка от продажи одного паттерна;
Ц – средняя цена паттерна;
К – комиссия за продажу паттерна.

Вычислим среднюю выручку от продажи одного паттерна:

$$\text{СВ} = \frac{15 \cdot 12}{100} = 1.8 \text{ р.}$$

Доходы, получаемые от оказания услуг через платформу, согласно законодательству облагаются налогом на профессиональный доход, который введен с 1 января 2023 года и составляет 10 % от общей суммы доходов. Так как программное средство уже разработано и не требует поддержки, дополнительные расходы отсутствуют. С учетом налога на профессиональный доход чистую прибыль от продажи одного паттерна рассчитаем по формуле

$$\text{СЧП} = \text{СВ} - \frac{\text{СВН}_{\text{пд}}}{100}, \quad (8)$$

где СЧП – средняя чистая прибыль от продажи одного паттерна;

СВ – средняя выручка от продажи одного паттерна;

$\text{Н}_{\text{пд}}$ – налог на профессиональный доход (в соответствии с действующим законодательством – 10 %).

Вычислим чистую прибыль от продажи одного паттерна:

$$\text{СЧП} = 1.8 - \frac{1.8 \cdot 10}{100} = 1.62 \text{ р.}$$

Ожидается, что с учетом примененной рекламы за первый месяц будет продано 400 паттернов, а за следующие месяцы ожидается повышение среднего количества проданных паттернов до 550 единиц с учетом увеличения количества пользователей, что сделает платформу известной.

Количество проданных паттернов вычисляется по формуле

$$\text{К}_{\text{п}} = \text{К}_{\text{п0}} + \text{К}_{\text{п1}} N, \quad (9)$$

где $\text{К}_{\text{п}}$ – количество проданных за год паттернов;

$\text{К}_{\text{по}}$ – средняя выручка от продажи одного паттерна;

$\text{К}_{\text{п1}}$ – количество проданных паттернов за оставшиеся 11 месяцев;

N – количество месяцев года, не включая первый месяц.

Вычислим количество паттернов, проданных в первый год работы программного средства:

$$\text{К}_{\text{п}} = 400 + 550 \cdot 11 = 6450.$$

Чистую прибыль за год рассчитаем как произведение количества проданных за год паттернов и чистой прибыли от одного паттерна по формуле

$$ЧП = СЧП \cdot К_п, \quad (10)$$

где ЧП – чистая прибыль за год;

СЧП – средняя чистая прибыль за продажу одного паттерна;

$К_п$ – количество проданных за год паттернов.

Вычислим чистую прибыль за год:

$$ЧП = 1.62 \cdot 6450 = 10\,449 \text{ р.}$$

Так как за год будет продано 6450 паттернов со средней чистой прибылью 1.62 р., чистая прибыль за год составит 10 449 р.

1.4 РАСЧЕТ ПОКАЗАТЕЛЕЙ ЭКОНОМИЧЕСКОЙ ЭФФЕКТИВНОСТИ РАЗРАБОТКИ И РЕАЛИЗАЦИИ ПРОГРАММНОГО СРЕДСТВАНА РЫНКЕ

Так как стоимость разработки программного составляет 8389.7 р., а рассчитанная чистая прибыль за год больше стоимости разработки и составляет 10449 р., оценка экономической эффективности инвестиций в разработку программного средства осуществляется с помощью расчета рентабельности инвестиций по формуле

$$П_ч = \frac{П_ч - З_р}{100} \cdot 100\ \%, \quad (11)$$

где $П_ч$ – прирост чистой прибыли от реализации программного средства на рынке, р.;

$З_р$ – затраты на разработку и реализацию программного средства, р.

Таким образом, экономическая эффективность разработки и реализации программного средства составляет

$$П_ч = \frac{10\,449 - 8389.7}{8389.7} \cdot 100\ \% = 24.5\ \%.$$

В ходе экономического обоснования были рассчитаны следующие показатели:

- прирост чистой прибыли составляет 10 449 р.;
- рентабельность составляет 24.5 %;
- срок окупаемости затрат на разработку и реализацию программного средства составляет меньше одного года.

В результате можно сделать вывод: инвестиции в разработку программного средства целесообразны.

Контрольные вопросы к лабораторной работе № 2

1 Какие аспекты важны для коммерциализации проекта по разработке программного средства?

2 В чем заключается экономическое обоснование проектного решения по разработке программного средства по индивидуальному заказу в рамках аутсорсинговой модели?

3 В чем заключается экономическое обоснование проектного решения по разработке программного средства в виде продукта для массового рынка?

4 В чем заключается экономическое обоснование проектного решения по разработке программного средства для внедрения в собственные бизнес-процессы разработчика?

5 Какие классические показатели следует рассчитать для оценки экономической эффективности и целесообразности инвестиций?

6 В чем суть дисконтирования и как рассчитывается коэффициент дисконтирования?

7 В чем отличие дисконтированного срока окупаемости инвестиций от простого?

8 Как определяется норма дисконта?

9 Какова методика расчета чистого дисконтированного дохода?

10 Какие необходимые структурные элементы выделяют в общей схеме экономического обоснования проектных решений?

11 Как рассчитать стоимостную оценку затрат, необходимых для реализации проектного решения по разработке программного средства?

12 Как определяется стоимостная оценка экономического эффекта от использования программного средства?

13 Какова методика расчета затрат на разработку программного средства для собственных нужд?

14 Как определить экономический эффект от реализации программного средства на рынке?

ЛАБОРАТОРНАЯ РАБОТА № 3
Анализ и моделирование предметной области

Цель выполнения лабораторной работы: проанализировать и описать текущее состояние бизнес-процессов, их «узкие места» и будущее состояние.

Задание	**Этапы выполнения задания**
1 Проанализировать модель «AS-IS» текущего состояния бизнес-процессов и выявить их «узкие места». Определить преимущества новых бизнес-процессов и построить модель «TO-BE»	1 Описать предметную область исследования. 2 Определить основные бизнес-процессы предметной области и представить их текстовое описание. Построить для них модель «AS-IS» с использованием нотации BPMN. 3 Проанализировать модель «AS-IS» с целью выявления проблемных бизнес-процессов и подготовить рекомендации по их улучшению. 4 Определить преимущества новых бизнес-процессов и построить модель «TO-BE» в нотации BPMN
2 Сформировать отчет по лабораторной работе	*Содержание отчета*: Титульный лист (приложение Б) 1 Описание предметной области 2 Анализ и моделирование бизнес-процессов предметной области 3 Выводы

Краткие теоретические сведения и методические указания к заданию 1

Пользовательские требования к программному продукту формулируются в терминах предметной области. Понимание предметной области является неотъемлемой частью успешной разработки программного продукта, дает возможность создать решение, которое будет реально полезным и функциональным для конечных пользователей.

Описание предметной области – это представление основных аспектов и характеристик предмета исследования.

Границы предметной области определяются ключевыми словами темы индивидуального задания, выполняемого в рамках лабораторных занятий.

Описание предметной области может включать в себя:

- определение ключевых понятий и терминов, характерных для данной предметной области;
- выявление важных сущностей, концепций, принципов;
- представление правил, методов, методик, математического аппарата;
- описание ключевых процессов и функций и др.

В рамках процессного подхода предприятие рассматривается как совокупность бизнес-процессов, потребляющая ресурсы и создающая продукты/услуги.

Бизнес-процесс – это совокупность различных видов деятельности, в рамках которой «на входе» используется один или более видов ресурсов, и в результате «на выходе» создается продукт, представляющий ценность для клиента.

Важным шагом при описании и структуризации деятельности любого предприятия/организации/компании являются выделение и классификация бизнес-процессов.

На практике выделяют бизнес-процессы уровня:

- предприятия – бизнес-процессы, которыми управляют топ-менеджеры;
- управлений/департаментов – бизнес-процессы крупных функциональных подразделений предприятия;
- подразделений/отделов – функции подразделений и отделов;
- операций на рабочих местах – функции, выполняемые на рабочих местах.

Выделение бизнес-процессов, как правило, проводят на верхнем уровне, что в конечном итоге приведет к увеличению производительности труда, экономии временных ресурсов на вертикальных и горизонтальных коммуникациях.

При описании деятельности компании на верхнем уровне наиболее часто используют следующие классы бизнес-процессов:

- *основные* – создают основные продукты, представляющие ценность для клиента, формируют доходы компании. Отличительная особенность – отражение бизнес-направлений компании, имеют стратегическое значение;
- *обеспечивающие (вспомогательные)* – ориентированы на создание ценности для внутреннего потребителя и предназначены для обеспечения функционирования основных бизнес-процессов. Создают ресурсы для основных процессов. Не имеют стратегического значения;
- *управляющие* – представляют функции управления на уровне каждого бизнес-процесса и предприятия в целом для достижения общих организационных целей.

Для корректного выделения основных бизнес-процессов следует предварительно составить перечень продуктов/услуг или видов деятельности компании.

Алгоритм выделения бизнес-процессов верхнего уровня:

1 Взять за основу стратегию компании (необходимое условие для выделения бизнес-процессов).

2 Определить организационную структуру верхнего уровня компании.

3 Составить перечень продуктов и услуг, производимых (оказываемых) компанией, которые лягут в основу определения основных бизнес-процессов верхнего уровня (т. е. определить вид деятельности).

4 Исходя из пункта 2 определить является компания типовой или нетиповой, целесообразно ли использовать референтную модель. Если да, то следует преобразовать готовую модель и перейти к получению процессов. Если же организация нетиповая, то выбираем подход к выделению бизнес-процессов «с нуля».

5 Выделить основные, обеспечивающие и управляющие бизнес-процессы верхнего уровня. При выделении бизнес-процессов следует учесть три правила:

- *на верхнем уровне должно быть выделено 15–20 бизнес-процессов* (такое количество является оптимальным с точки зрения уровня контроля первым руководителем). Выделенные бизнес-процессы необходимо разделить на три группы: основные, вспомогательные и процессы управления;

- *правило равнозначности.* Бизнес-процессы верхнего уровня должны быть равнозначны с точки зрения важности для достижения стратегии компании. Наиболее важные и проблемные процессы целесообразно поднимать на более высокие уровни процессной модели компании. Однако в различных компаниях есть много похожих бизнес-процессов, которые могут находиться на разных уровнях процессной модели по причине различной важности для стратегии компании, а также из-за их различной степени проблемности;

- *перечень бизнес-процессов верхнего уровня должен быть согласован и утвержден первым руководителем компании.*

Представить выделенные процессы в виде карты процессов верхнего уровня. Пример карты бизнес-процессов верхнего уровня представлен на рисунке 10[1].

6 Сформировать матрицу распределения ответственности за процессы. Пример матрицы представлен на рисунке 11[1].

Для полного описания бизнес-процессов необходимо ответить на следующие вопросы:

1 Какие работы необходимо выполнить для достижения поставленных целей?

2 Кто является ответственным исполнителем (владельцем) бизнес-процесса?

3 Кто является исполнителем работ в рамках бизнес-процесса?

4 Какова последовательность выполнения работ?

5 Какие входящие и исходящие потоки информации и материалов используются, создаются в рамках выполнения каждой работы?

6 Как осуществляется контроль и управление данным процессом?

7 Какими нормативными правовыми, организационно-распорядительными и инструктивными материалами руководствуются исполнители?

8 Какие ресурсы необходимы для выполнения каждой работы?

---

[1] URL: https://upr.ru/article/3-pravila-videleniia-processov-verhnego-urovnia.

Рисунок 10 – Карта процессов верхнего уровня с отображением распределения ответственности за бизнес-процессы производственной компании

	Генеральный директор	Директор по снабжению	Директор по производству	Директор по продажам	Директор по контрактному производству	Директор по транспорту	Директор по общим вопросам	Директор по развитию	Финансовый директор	Директор по ИТ	Директор по персоналу
Основные процессы											
Закупка сырья и материалов		+									
Производство продукции			+								
Продажа продукции				+							
Продажа услуг контрактного производства					+						
Доставка продукции потребителям						+					
Обеспечнвающне процессы											
Административно-хозяйственное обеспечение							+				
IT-обеспечение и связь										+	
Обеспечение безопасности							+				
Юридическое обеспечение									+		
Ремонт и модернизация оборудования			+								
Капитальный ремонт и строительство							+				
Процессы управления											
Стратегическое управление								+			
Управление финансами									+		
Управление маркетингом				+							
Управление бизнес-процессами и качеством								+			
Управление персоналом											+
Управление проектами развития								+			

Рисунок 11 – Матрица распределения ответственности

Отличное описание бизнес-процессов представлено в книге «Настольная книга аналитика. Практическое руководство по проектированию бизнес-процессов и организационной структуры» С. Ковалева, В. Ковалева [6].

Выделяют три основных способа описания бизнес-процессов: текстовый, графический и табличный.

Пример краткого текстового описания бизнес-процесса предметной области приведен в приложении В.

BPM CBOK определяет моделирование бизнес-процессов как набор действий, создающих представление существующего или предполагаемого бизнес-процесса.

Бизнес-моделирование представляет собой разработку модели деятельности предприятия с целью решения задач анализа и совершенствования его деятельности. Бизнес-модель – это формализованное описание аспектов деятельности предприятия.

Перечень наиболее распространенных нотаций моделирования бизнес-процессов (согласно BPM CBOK) представлен на рисунке 12 [7].

НОТАЦИЯ	КРАТКОЕ ОПИСАНИЕ
BPMN2.0	▪ Стандарт, созданный консорциумом Object Management Group (OMG); ▪ содержит 103 символа; ▪ полезен для представления модели разным аудиториям
Дорожки	▪ Является не самостоятельной нотацией, а дополнением ко многим другим нотациям; ▪ помогает изобразить переходы ответственности в ходе процесса
Блок-схемы	▪ Исходно принятый в качестве стандарта ANSI64, включает небольшой и очень простой набор символов; ▪ позволяет быстро понять ход потока работ
EPC	▪ Нотация, разработанная в рамках методологии ARIS, рассматривает входы и выходы шагов процесса как события; ▪ позволяет моделировать сложные комплексы процессов
UML	Поддерживаемое OMG семейство нотаций и методов моделирования, предназначенное в основном для описания требований к информационным системам
IDEF	▪ Стандарт, акцентирующий внимание на входах, выходах, механизмах и средствах управления процессом и явно увязывающий процесс с выше- и нижестоящими в иерархии детализаци; ▪ хорошая отправная точка для составления взгляда на организацию как на единое целое

Рисунок 12 – Наиболее распространенные нотации моделирования бизнес-процессов

Для того чтобы систематизировать текущие бизнес-процессы и описать их состояние, необходимо построить модель «AS-IS», которую называют функциональной и выполняют с использованием популярных графических нотаций.

Для создания верхнего уровня модели бизнес-процессов, как правило, используют нотацию IDEF0, обеспечивающую наиболее общее описание объекта моделирования в виде множества взаимосвязанных функций. Описывает логическую последовательность работ.

Нотация IDEF0 используется для построения структурных моделей, показывающих, как устроен бизнес организации, и раскрывающих информацию о целевых группах бизнес-процессов и их взаимосвязях. Структурная модель не отражает последовательность выполнения процессов во времени.

На практике структурные модели процессов применяются [8]:

- для описания и анализа бизнес-модели организации, а также определения возможных направлений ее реорганизации;
- разработки системы бизнес-процессов в соответствии с принципом декомпозиции «сверху вниз»;
- системного описания бизнес-процессов, которые необходимо автоматизировать (например, в MRPII, ERP-системе);
- описания состава процессов, декомпозированных на следующий уровень.

На нижнем уровне (операционный уровень выполнения бизнес-процесса, модели WorkFlow) используют нотации BPMN, Процесс, Процедура, EPC.

Нотация BPMN широко используется бизнес-аналитиками для моделирования сложных бизнес-процессов (кросс-функциональных) с большим количеством исполнителей, выполняющих разные задачи. BPMN описывает функциональную последовательность работ. С ее помощью можно подробно описать каждый шаг исполнения бизнес-процесса.

Пример BPMN-модели бизнес-процессов предметной области представлен на рисунке 13[1].

Нотации Процесс и Процедура предназначены для описания логики выполнения бизнес-процесса и позволяют задавать как причинно-следственные связи, так и временную последовательность выполнения действий процесса. Данные нотации применяют и для моделирования отдельных процессов, и на нижнем уровне модели, созданной в нотации IDEF0.

Нотация EPC описывает событийную последовательность работ и используется для представления процедур, сценариев и регламентов.

Модель «AS-IS» должна отражать всю полноту информации о бизнес-процессах, существующих на момент обследования предприятия/организации/отдела и позволяет понять, каким образом выполняются бизнес-процессы, а также выявить их узкие места и в конечном итоге разработать мероприятия по их оптимизации/цифровизации.

Модель «AS-IS» должна быть максимально приближена к действительности и основываться на реальных потоках бизнес-процессов, а не на их идеализированном представлении.

При проведении анализа существующей ситуации и разработки модели будущего состояния необходимо опираться, в первую очередь, на результаты стратегического анализа и планирования.

Следует изучить, как работает предприятие/организация/отдел, выделить целевые бизнес-процессы, сделать их описание, и только потом строить модель «AS-IS».

[1] URL: https://optimacons.info/upload/kb-articles/svg/bpmn-01-1-0-02.svg.

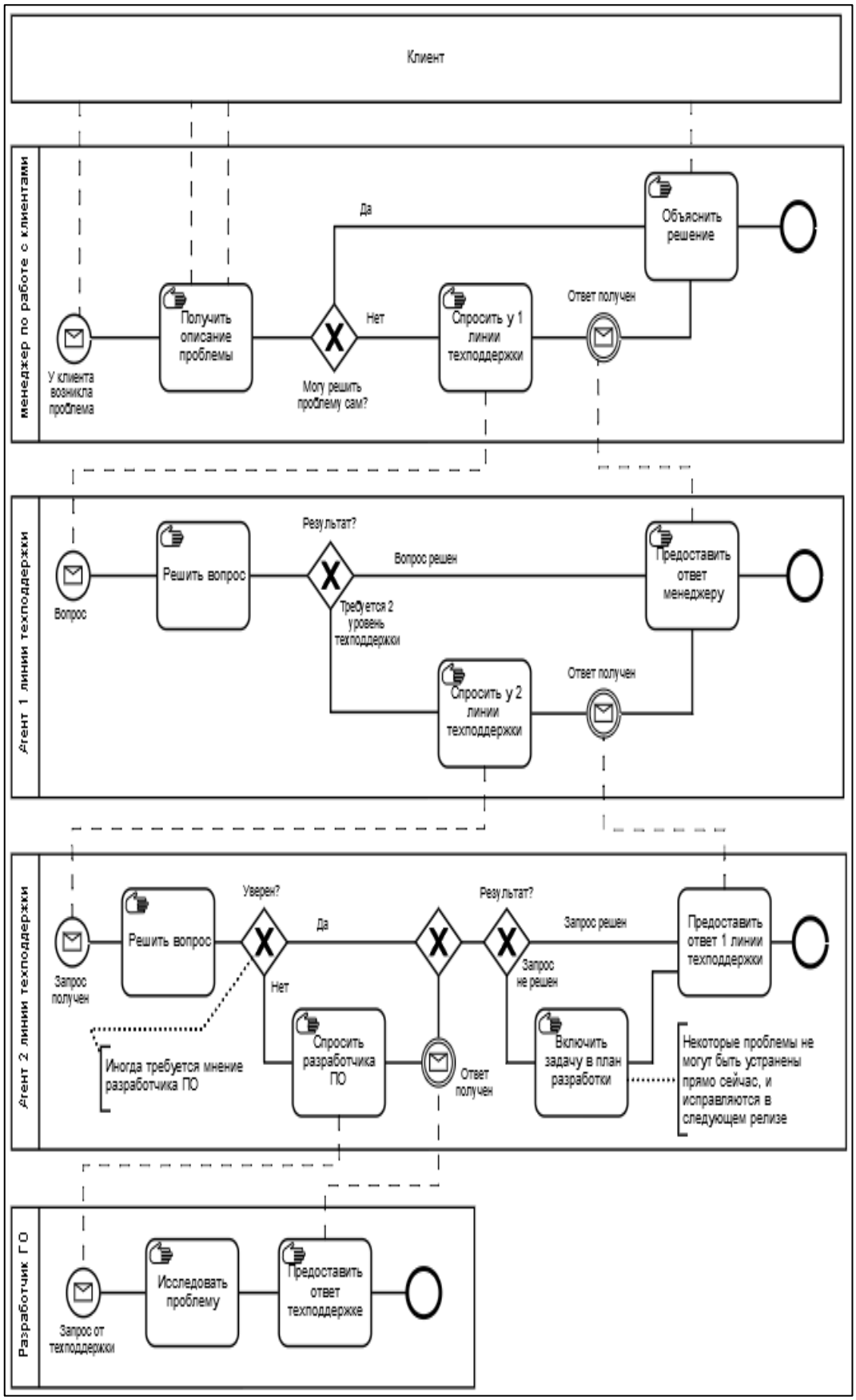


Рисунок 13 – Пример BPMN-модели процессов предметной области

Анализ функциональной модели позволяет выявить проблемные места бизнес-процессов, преимущества новых бизнес-процессов и глубину изменений, которым подвергнется существующая структура организации бизнеса.

Узкие места бизнес-процессов в модели «AS-IS» – работы/функции/операции и связи, которые снижают эффективность процесса, увеличивают его трудоемкость и стоимость.

Признаки «узких мест»[1]:

- бесполезные, неуправляемые и дублирующиеся работы/функции/операции;
- нарушение временных рамок выполнения бизнес-процесса;
- неэффективный документооборот (нужный документ не оказывается в нужном месте в нужное время);
- отсутствие обратных связей по управлению (на проведение работы не оказывает влияния ее результат), входу (объекты или информация используются нерационально) и т. д.

Неплохой пример описания слабых мест бизнес-процессов рассматривается в источнике [9].

Пример модели «AS-IS» процессов предметной области с указанием проблемных мест бизнес-процессов представлен на рисунке 14.

Для снижения затрат (финансовых, материальных, людских, временных), повышения эффективности функций, которые оказались неэффективными и требовали изменений, разрабатывается модель «TO-BE». Модель «TO-BE» строится для новых и модифицированных бизнес-процессов.

Пример модели «TO-BE» процессов предметной области с указанием новых и модифицированных бизнес-процессов представлен на рисунке 15.

[1] URL: https://coollib.com/b/157463-sergey-vladimirovich-maklakov-modelirovanie-biznes-protsessov-s-bpwin-40/read.

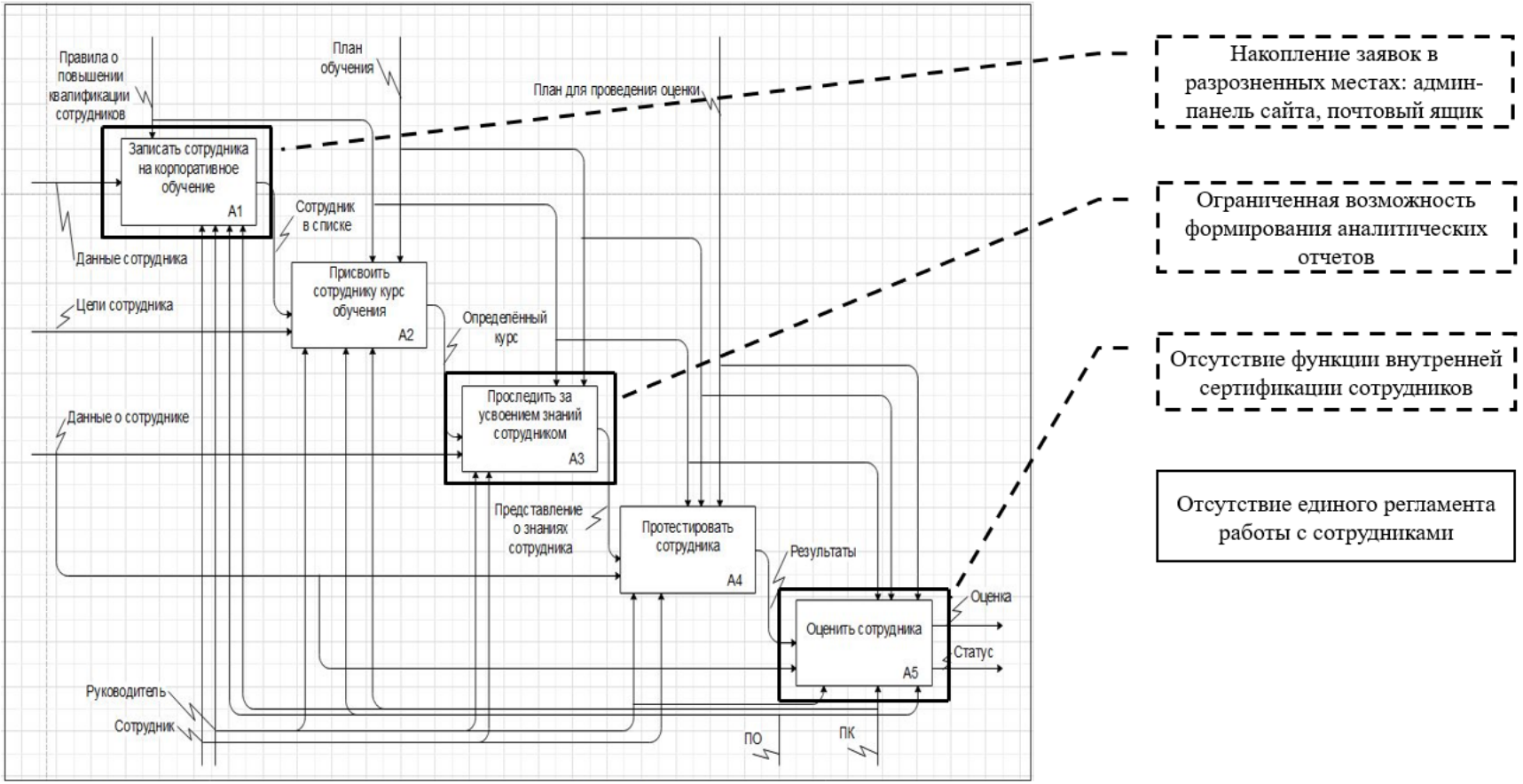


Рисунок 14 – Модель «AS-IS» процессов предметной области с указанием проблемных мест бизнес-процессов

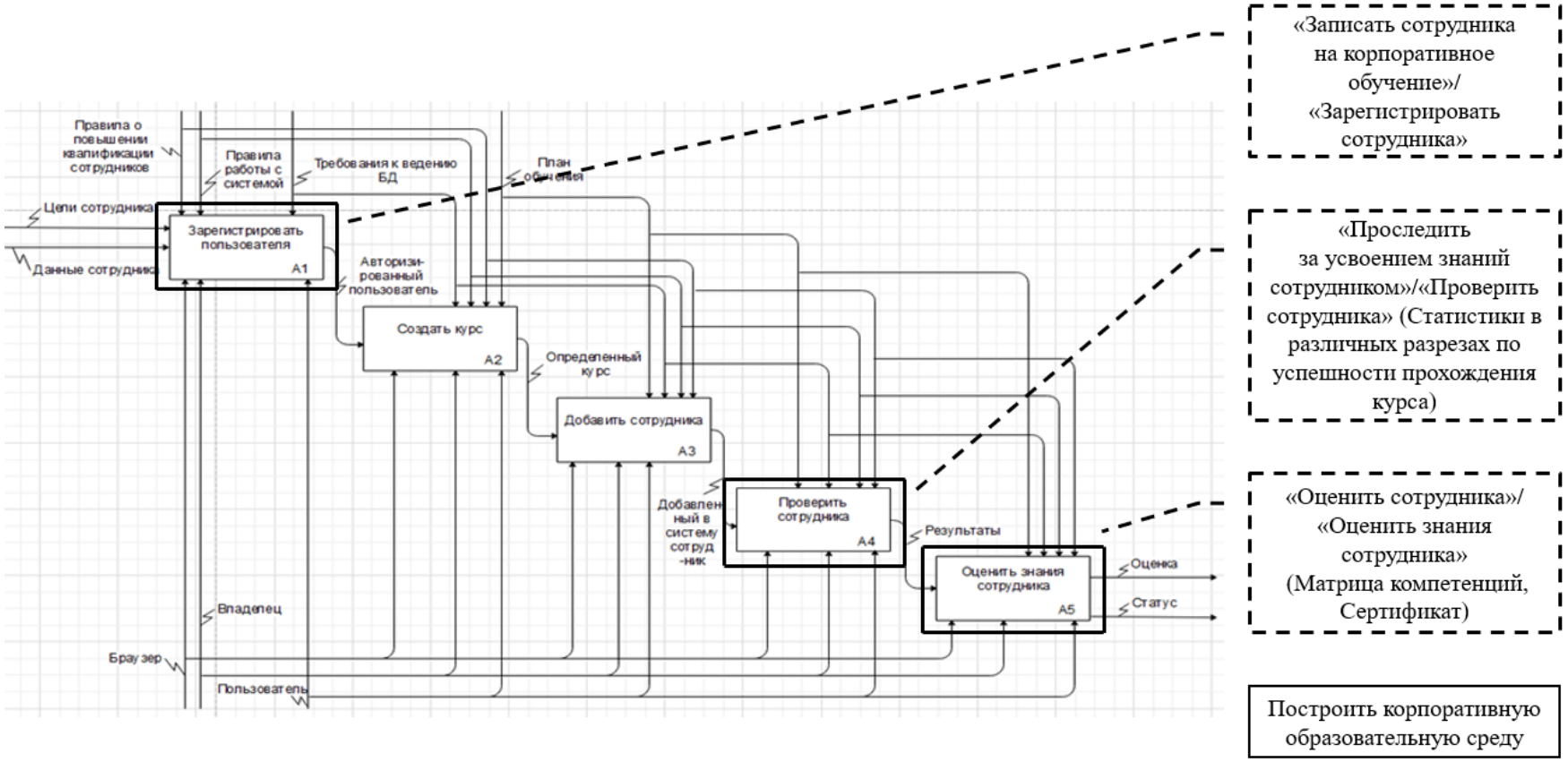


Рисунок 15 – Модель «TO-BE» процессов предметной области с указанием новых и модифицированных бизнес-процессов

Подробнее о нотациях моделирования читайте в дополнительных источниках:

1 DFD: примеры и правила построения диаграмм потоков данных [Электронный ресурс]. – Режим доступа: https://practicum.yandex.ru/blog/diagramma-potokov-dannyh-dfd.

2 Ковалев, С. Технологии процессного управления. Три правила выделения процессов верхнего уровня [Электронный ресурс]. – Режим доступа: https://upr.ru/article/3-pravila-videleniia-processov-verhnego-urovnia.

3 Нотации EPC [Электронный ресурс]. – Режим доступа: https://www.businessstudio.ru/wiki/docs/v4/doku.php/ru/csdesign/bpmodeling/epc_notation.

4 Нотации Процесс и Процедура [Электронный ресурс]. – Режим доступа: https://www.businessstudio.ru/wiki/docs/v4/doku.php/ru/csdesign/bpmodeling/process_procedure.

5 Нотация BPMN 2.0 [Электронный ресурс]. – Режим доступа: https://www.elma-bpm.ru/bpmn2.

6 Нотация BPMN [Электронный ресурс]. – Режим доступа: https://www.businessstudio.ru/wiki/docs/v4/doku.php/ru/csdesign/bpmodeling/bpmn_notation.

7 Нотация IDEF0 [Электронный ресурс]. – Режим доступа: https://www.businessstudio.ru/wiki/docs/v4/doku.php/ru/csdesign/bpmodeling/idef0.

8 Что такое нотация BPMN 2.0 и как она помогает смоделировать бизнес-процесс [Электронный ресурс]. – Режим доступа: https://practicum.yandex.ru/blog/notaciya-bpmn-dlya-biznes-processov.

Контрольные вопросы к лабораторной работе № 3

1 Что такое функциональная модель предметной области?

2 Для чего проводят анализ функциональной модели?

3 Чем различаются модели «AS-IS» и «TO BE»?

4 Что представляют собой узкие места бизнес-процессов в модели «AS-IS» и как их выявляют?

5 Какие нотации моделирования используются для построения процессных моделей?

6 Для чего используется нотация моделирования BPMN?

7 Для каких целей применяются структурные модели бизнес-процессов?

8 Каков алгоритм выделения бизнес-процессов верхнего уровня?

9 Как нотация IDEF0 помогает смоделировать бизнес-процесс?

10 Какие существуют способы описания бизнес-процессов?

11 Что представляет собой карта процессов верхнего уровня?

12 Как классифицируют бизнес-процессы?

ЛАБОРАТОРНАЯ РАБОТА № 4
Разработка требований к программному средству

Цель выполнения лабораторной работы: сформировать спецификацию требований к программному средству.

Задание	**Этапы выполнения задания**
1 Разработать требования к программному средству	1 Идентифицировать базовые пользовательские требования. 2 Построить диаграмму вариантов использования, исключая тривиальные операции работы с БД. Описать варианты использования сценарием. 3 Выявить функциональные и нефункциональные требования и разработать их спецификацию
2 Представить функциональную архитектуру решения	На основе разработанных требований к программному средству представить функциональную архитектуру решения и дать ее текстовое описание
3 Сформировать отчет по лабораторной работе	*Содержание отчета*: Титульный лист (приложение Б) 1 Варианты использования 2 Спецификация функциональных и нефункциональных требований 3 Функциональная архитектура программного средства 4 Выводы

Краткие теоретические сведения и методические указания к заданию 1

На основе бизнес-требований разрабатываются пользовательские требования, формирующие уровень базовых пользовательских требований.

Пользовательские требования (user requirements) представляют собой требования заинтересованных лиц, которые должны быть выполнены для удовлетворения бизнес-требований.
Пользовательские требования обеспечивают соответствующий уровень детализации, позволяющий вести разработку функциональных требований.

Пользовательские требования ориентированы на потребности, ожидания конечных пользователей и описывают задачи, которые можно выполнять с помощью разрабатываемого программного продукта, а также способы взаимодействия пользователя с системой.

Рекомендации к формированию пользовательских требований представлены на рисунке 16.

Отвечают на вопрос	*Что* хотят пользователи, *что* они могут сделать с продуктом и *каким* они хотят его видеть?
Пример	Возможность совершения онлайн-заказа на сайте интернет-магазина спортивной обуви
Источники пользовательских требований (где искать?)	▪ Наблюдение за пользователями; ▪ анализ обратной связи; ▪ проведение опросов; ▪ интервью; ▪ изучение данных об использовании аналогичных продуктов на рынке
Стейкхолдеры (у кого спрашивать?)	▪ Конечные пользователи; ▪ дизайнеры пользовательских интерфейсов; ▪ маркетологи; ▪ менеджеры по продукту
Лайфхаки по разработке пользовательских требований	▪ Прототипирование: создание прототипов позволяет пользователям визуализировать конечный продукт; ▪ тестирование с реальными пользователями: регулярные тесты с участием реальных пользователей на ранних этапах разработки помогают выявить потенциальные проблемы и улучшить пользовательский опыт

Рисунок 16 – Рекомендации по формированию пользовательских требований

Основным подходом к описанию пользовательских требований является создание вариантов использования (use cases), а также могут быть оформлены в виде пользовательских историй и пользовательских сценариев.

Диаграмма вариантов использования (сценариев поведения, прецедентов) – диаграмма, дающая общее представление о функциональном назначении системы с точки зрения получения значимого результата для пользователя. Это диаграмма, на которой изображаются отношения между актерами и вариантами использования.

Цели создания и назначение диаграммы вариантов использования отражены на рисунке 17.

Актер – внешняя сущность, не принадлежащая системе. Актер отвечает на вопрос: *кто* будет пользоваться системой?

Классификация актеров с точки зрения взаимодействия с системой: основные – пользователи программной системы, которые вызывают ее реакцию; вспомогательные – лица, которые обслуживают систему (например, администратор, СУБД); закулисные – лица, которые связаны с исполнением варианта использования, но не являются основными или вспомогательными актерами (например, платежная система).

Цели создания диаграммы вариантов использования	▪ Определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы; ▪ сформулировать общие требования к функциональному поведению проектируемой системы; ▪ разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей; ▪ подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями
Назначение диаграммы вариантов использования	▪ Определяет поведение системы с точки зрения пользователя; ▪ рассматривается как главное средство для первичного моделирования динамики системы; ▪ используется для выяснения требований к системе, фиксации этих требований в форме, которая позволит проводить их дальнейшую разработку
Диаграмма вариантов использования содержит	Конечное множество вариантов использования, которые в целом должны определять все возможное поведение системы

Рисунок 17 – Цели создания и назначение диаграммы вариантов использования

Вариант использования демонстрирует последовательность взаимодействия системы и действующего лица, в результате которого действующее лицо получает полезный результат. Другими словами, вариант использования отражает цель пользователя и описывает сценарий достижения этой цели.

Рекомендации по разработке диаграммы вариантов использования:

▪ необходимо показывать только те варианты использования, которые важны для понимания поведения системы;

▪ не следует злоупотреблять многократным использованием стереотипов «include» и «extend»;

▪ не старайтесь разработать диаграмму с первого раза. Используйте модифицированную диаграмму для улучшений;

▪ используйте границы системы, чтобы отделить внутреннюю часть системы от ее внешнего окружения;

▪ примеры некорректной и корректной диаграмм вариантов использования приведены на рисунке 18.

Для документирования вариантов использования может быть использован шаблон, представленный таблицей 12.

Функциональное требование (functional requirement) – положение о фрагменте требуемой функциональности или поведения системы при определенных условиях.

Функциональные требования определяют реализацию, чтобы пользователи смогли выполнить свои задачи (пользовательские требования) в рамках бизнес-требований.

а *б*

а – некорректная; *б* – корректная

Рисунок 18 – Диаграммы вариантов использования

Таблица 12 – Шаблон описания варианта использования сценарием

Идентификатор требования и название	**UC-1 Зарегистрироваться в системе**
Действующее лицо	
Краткое описание варианта использования	
Входные условия	
Основной поток действий при исполнении варианта использования	
Альтернативный поток действий при исполнении варианта использования	
Выходные условия	
…	

Функциональное требование строится по формуле

ID -> Условие -> Система должна -> действие.

Для документирования функциональных требований может быть использован шаблон, представленный таблицей 13.

Таблица 13 – Шаблон для документирования функциональных требований

Идентификатор требования	**Название варианта**	**Роль**	**Описание**	**Функциональные требования**
UC-1				FR1-1 FR1-2 …
UC-2				FR2-1 FR2-2 …
UC-3				FR2-1 FR2-2 FR2-3 …
…				…

Нефункциональные требования – описание свойств или характеристик, которые система должна демонстрировать, или ограничений, которые она должна соблюдать.
Отвечают на вопросы: не *что* система делает, а *как хорошо* она это делает? *Какие* должны быть границы? *Какое* должно быть качество?

Нефункциональные требования описывают свойства системы (удобство использования, безопасность, надежность, расширяемость и т. д.), которыми она должна обладать при реализации своего поведения.

К группе нефункциональных требований относят **атрибуты качества**, **ограничения**, **внешние интерфейсы**, **требования к данным**.

Атрибуты качества устанавливают параметры, которые будут использоваться для оценки работы системы.

Атрибутов качества большое количество, но для любого проекта реально важными являются лишь некоторые их подмножества: надежность, переносимость, масштабируемость, удобство и др.

Ограничения – условия или требования, которые накладываются на доступные разработчику возможности проектирования или разработки системы.

Ограничения представляют собой факторы, ограничивающие выбор способов и средств (в том числе инструментов) реализации продукта.

Внешние интерфейсы – интерфейсы взаимодействия с внешними системами или операционной средой.

Требования к данным определяют список данных, которые используются для описания элементов в разрабатываемой системе. Часто сюда относят описание базы данных и особенностей ее использования.

Для документирования нефункциональных требований может быть использован шаблон, представленный таблицей 14.

Таблица 14 – Шаблон для документирования нефункциональных требований

Идентификатор требования	**Описание**
Пользовательские интерфейсы	
UI-1	
UI-2	
Интерфейсы программного средства	
SI-1	
SI-2	
…	

Пользовательские, функциональные и нефункциональные требования обычно включаются в Спецификацию требований к программному обеспечению (Software Requirements Specification, SRS). Пример оформления SRS приведен в приложении В источника [4].

Краткие теоретические сведения и методические указания к заданию 2

Под функциональной архитектурой решения будем понимать функциональный состав блоков системы, спроектированный с учетом бизнес-требований, пользовательских требований и обеспечивающий решение бизнес-задач. Пример приведен на рисунке 19.

ПРИМЕР ФУНКЦИОНАЛЬНОЙ АРХИТЕКТУРЫ ПРОГРАММНОГО РЕШЕНИЯ

Рисунок 19 – Функциональная архитектура решения

Подробнее о работе с требованиями читайте в дополнительных источниках:

1 Варианты на все случаи жизни: как написать полезный use case [Электронный ресурс]. – Режим доступа: https://practicum.yandex.ru/blog/chto-takoe-use-case-kak-ih-napisat.

2 Все, что вам нужно знать о моделировании вариантов использования case [Электронный ресурс]. – Режим доступа: https://www.cybermedian.com/ru/all-you-need-to-know-about-use-case-modeling.

3 Коберн, А. Современные методы описания функциональных требований к системам / А. Коберн. – М. : Лори, 2021. – 263 с.

4 Леффингуэлл, Д. Принципы работы с требованиями к программному обеспечению. Унифицированный подход / Д. Леффингуэлл, Д. Уидриг. – М. : Вильямс, 2002. – 448 с.

5 Халл, Э. Инженерия требований / Э. Халл, К. Джексон, Дж. Дик. – М. : ДМК Пресс, 2017. – 223 с.

Контрольные вопросы к лабораторной работе № 4

1 В чем принципиальная разница между требованием и потребностью с точки зрения бизнес-анализа?

2 Что представляют собой пользовательские требования и какие способы их описания существуют?

3 В чем отличие пользовательских требований от функциональных?

4 Какой уровень детализации традиционно применяется при описании варианта использования?

5 Каково назначение вариантов использования?

6 Какую роль могут выполнять актеры по отношению к вариантам использования?

7 Какие требования предъявляют к построению диаграммы вариантов использования?

8 Что такое функциональное требование?

9 Что представляют собой нефункциональные требования и как их фиксировать?

10 Какие артефакты относят к группе нефункциональных требований?

11 Что такое SRS и каковы ее ключевые элементы?

12 Что такое функциональная архитектура решения?

ЛАБОРАТОРНАЯ РАБОТА № 5
Проектирование архитектурных решений

Цели выполнения лабораторной работы:
- описать архитектуру программного средства;
- выбрать стек технологий для реализации проекта.

Задание	**Этапы выполнения задания**
1 Разработать архитектурное решение	Описать архитектурные решения на основании постановки задач для лабораторных работ и анализа литературных источников с использованием нотации C4 model
2 Выбрать стек технологий для разработки проекта	1 Проанализировать технологии, используемые для реализации схожих задач. 2 Выбрать наиболее подходящие технологии для разработки программного средства
3 Сформировать отчет по лабораторной работе	*Содержание отчета*: Титульный лист (приложение Б) 1 Архитектура программного средства 2 Таблицы результатов анализа технологий для разработки программного средства 3 Описание выбранных технологий для разработки с обоснованием их выбора 4 Выводы

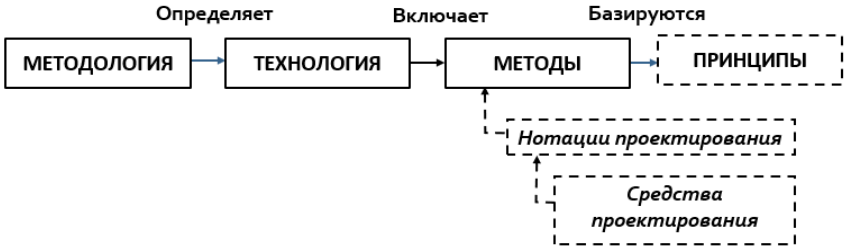
Краткие теоретические сведения и методические указания к заданию 1

Начиная рассматривать вопрос о проектировании архитектуры, сформируем словарь базовых терминов и определений (таблица 15).

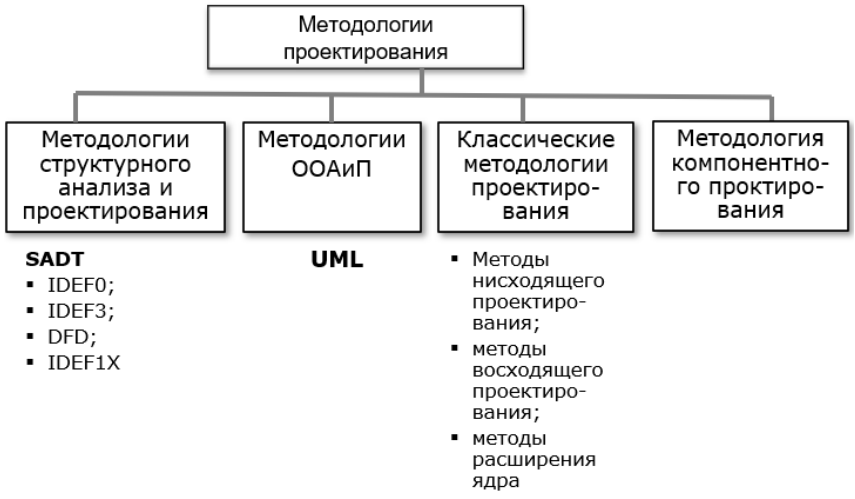
Таблица 15 – Словарь базовых терминов и определений

Термин	**Определение**
1	**2**
Проектирование по ISO / IEC / IEEE 24765	**Проектирование** – часть стадии жизненного цикла системы, процесс определения архитектуры, компонентов, интерфейсов и других характеристик системы или ее части
Проектирование по SWEBoK	**Проектирование** – программный дизайн (Software Design) как результат деятельности по проектированию должен описывать архитектуру ПО, т. е. представлять декомпозицию программной системы в виде организованной структуры компонентов и интерфейсов между компонентами

Продолжение таблицы 15

1	2
Проектирование	- Процесс определения архитектуры, компонентов, интерфейсов, других характеристик системы и конечного результата; - инженерная деятельность в рамках жизненного цикла (ЖЦ) ПО, в которой надлежащим образом анализируются требования для создания описания внутренней структуры ПО, являющейся основой для конструирования программного обеспечения
Ступени (шаги) проектирования	- *Предварительное (архитектурное) проектирование* формирует абстракции архитектурного уровня (архитектура программ и данных). Включает три типа деятельности: структурирование системы; моделирование управления; декомпозиция подсистем на модули; - *детальное проектирование* уточняет абстракции, добавляет подробности алгоритмического уровня (структуры и алгоритмы программ); - *интерфейсное проектирование*
Области проектирования	- Проектирование объектов данных, которые будут реализованы в базе данных; - проектирование программ, экранных форм, отчетов, которые будут обеспечивать выполнение запросов к данным; - учет конкретной среды или технологии, а именно топологии сети, конфигурации аппаратных средств, используемой архитектуры, параллельной и распределенной обработки данных и т. п.
Составляющие проектирования	
Методология проектирования	Предлагает общие принципы проектирования, определяет подходы (концептуальную модель) к оценке и выбору варианта системы, последовательность стадий и этапов проектирования и в конечном итоге позволяет выбрать метод проектирования
Цель методологии проектирования	Регламентировать процесс проектирования: - обеспечить создание системы, отвечающей целям и задачам организации, а также предъявляемым требованиям по автоматизации деловых процессов заказчика; - гарантировать создание системы с заданным качеством в заданные сроки и в рамках установленного бюджета; - поддерживать удобную дисциплину сопровождения, модификации и наращивания системы; - обеспечивать преемственность разработки, т. е. использование в разрабатываемой системе существующей информационной инфраструктуры организации (задела в области информационных технологий)

Продолжение таблицы 15

1	2
	
Технология проектирования	Совокупность технологических операций проектирования в их последовательности и взаимосвязи, приводящая к разработке проекта системы
Требования к технологии проектирования	- Поддерживать полный ЖЦ системы и обеспечивать гарантированное достижение целей ее разработки с заданным качеством и в установленное время; - обеспечивать возможность декомпозиции системы; - обеспечивать возможность ведения работ по проектированию отдельных подсистем небольшими группами; - обеспечивать минимальное время получения работоспособной системы; - предусматривать возможность управления конфигурацией проекта, ведения версий проекта и его составляющих, возможность автоматического выпуска проектной документации и синхронизацию ее версий с версиями проекта; - обеспечивать независимость выполняемых проектных решений от средств реализации системы – СУБД, ОС, языка и системы программирования
Классы технологий проектирования	

Продолжение таблицы 15

1	2
Примеры технологии проектирования	- CASE-технология – совокупность методов проектирования системы, а также набор инструментальных средств, позволяющих в наглядной форме моделировать предметную область, анализировать эту модель на всех стадиях разработки, сопровождения системы и разрабатывать приложения в соответствии с информационными потребностями пользователей; - технология проектирования RUP; - технология проектирования DATARUN
Методы проектирования	Представляют собой совокупность: - концепций и теоретических основ; - нотаций, используемых для построения моделей статической структуры и динамики поведения проектируемой системы; - процедур, определяющих практическое применение метода. Метод конкретизирует порядок разработки отдельных элементов, комплексов задач, подсистем и системы в целом. Выделяют: - функционально-ориентированные (структурные) методы. Базируются на структурном анализе, структурных картах, Dataflow-диаграммах и др. Они ориентированы на идентификацию функций и их уточнение сверху вниз, после чего проводится разработка диаграмм потоков данных и описание процессов; - объектно-ориентированные методы. Базируются на ключевых принципах ООП: наследование, полиморфизм и инкапсуляция, а также на абстрактных структурах данных и отображении объектов; - ориентированные на структуры данных. Базируются на методе Джексона (Jackson) и используются для задания входных и выходных данных; - компонентное проектирование. Ориентировано на использование и интеграцию компонентов (особенно компонентов повторного использования), на их интерфейс, обеспечивающий взаимодействие компонентов; - другие методы: формальные, точные, трансформационные, а также UML для моделирования архитектурных решений с помощью диаграмм
Нотации проектирования	Позволяют представить артефакты ПО, его структуру, а также поведение системы. Выделяют два типа нотаций: структурные и поведенческие (однако существует множество различных их представлений). - *Структурные нотации* используются для представления структурных аспектов проектирования, компонентов и их взаимосвязей, элементов архитектуры и их интерфейсов. К ним относятся формальные языки спецификаций и проектирования: ADL, UML, ERD, IDL, классы и объекты, компоненты и классы, Use Case Driven и др. Нотации включают языки описания архитектуры и интерфейса, диаграммы классов и объектов, диаграммы сущность – связь, компонентов, развертывания, а также структурные диаграммы и схемы;

Продолжение таблицы 15

1	2
	- *поведенческие нотации* отражают динамический аспект поведения систем и их компонентов. Таким нотациям соответствуют диаграммы: Data Flow, Decision Tables, Activity, Colloboration, Pre-Post Conditions, Sequence, таблицы принятия решений, формальные языки спецификации, языки проектирования PDL и др.
Средства проектирования	Инструментальные средства проектирования, поддерживающие метод проектирования. **CASE-средство** – программное средство, поддерживающее процессы ЖЦ ПО, включая анализ требований к системе, проектирование прикладного ПО и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы

Проектирование архитектуры является одним из ключевых этапов разработки любого программного обеспечения. Правильный выбор архитектурных компонентов позволяет создавать «гибкие» программы, которые будут своевременно реагировать на изменяющиеся условия внешней среды.

Архитектурные стили делятся на два типа [10] (рисунок 20):

- монолитный (с единым развертыванием блоком для всего кода);
- распределенный (с несколькими развертываемыми блоками, подключаемыми друг к другу по протоколам удаленного доступа).

Рисунок 20 – Архитектурные стили

Архитектура программного обеспечения – это схема высокого уровня, определяющая структуру, дизайн и поведение программной системы. Она включает в себя организацию компонентов, их взаимодействие и ограничения системы.

Хорошо спроектированная программная архитектура учитывает различные факторы, такие как масштабируемость, производительность, ремонтопригодность и безопасность[1].

Неправильно спроектированная архитектура обуславливает нестабильность ПО, невозможность поддерживать существующие или будущие бизнес-требования, сложности при развертывании или управлении программного обеспечения.

Проектирование архитектуры программного обеспечения следует осуществлять таким образом, чтобы учитывать требования следующих основных областей: пользователи, бизнес и система (IT-инфраструктура).

Например, бизнес-цели и IT-инфраструктура порождают пользовательский интерфейс программного обеспечения, поэтому изменение одного из основополагающих факторов неизбежно окажет влияние на механизмы взаимодействия с пользователем.

Единых методик, подходов или алгоритмов для проектирования архитектуры нет. На практике чаще всего системные архитекторы разрабатывают архитектуру, основываясь на собственном или чужом опыте проектирования. Однако, следующие принципы помогут создать архитектуру, максимально соответствующую требованиям всех заинтересованных сторон:

- определить нужный тип приложения;
- выбрать стратегию развертывания;
- выбрать технологии для реализации;
- определить показатели качества;
- принять решение о путях реализации сквозной функциональности (например, механизмы протоколирования, механизмы аутентификации и авторизации, инфраструктура управления исключениями и т. д.).

Более детально с процессом проектирования архитектуры можно ознакомиться в источниках: «Руководство Microsoft по проектированию архитектуры приложений», М. Ричардс «Фундаментальный подход к программной архитектуре: паттерны, свойства, проверенные методы».

Также стоит четко понимать различия понятий «архитектурный стиль» и «архитектурный паттерн». Нередко между ними ставят знак равно, однако это не в полной мере верно.

[1] URL: https://appmaster.io/ru/blog/kak-vybrat-arkhitekturu-po.

Архитектурный стиль описывает способ построения системы в целом (например, микросервисы, многослойный монолит и пр.), в то время как *архитектурные паттерны* (например, CQRS, 2PC, Saga, RLBS, шардинг и пр.) – это способы описания определенных и специфических архитектурных аспектов этой общей структуры.

Например, архитектурный паттерн CQRS можно применить к любому архитектурному стилю. Это верно и для любого другого архитектурного паттерна – любой из них применим для описания основных деталей общего архитектурного стиля, который используется для разработки программного обеспечения [1].

Для формализации архитектуры ПО можно использовать различные нотации (UML, arc42, C4 model и др.).

В рамках лабораторной работы для проектирования архитектуры программного средства будет использована нотация C4 model[2].

В качестве инструментов для моделирования архитектуры в данной нотации можно воспользоваться следующими:

- app.diagrams (https://app.diagrams.net/);
- lucidchart (https://www.lucidchart.com/pages/);
- plantuml (https://plantuml.com/);
- mermaid (https://mermaid.js.org/);
- ilograph (https://www.ilograph.com/).

Основные элементы данной нотации представлены на рисунке 21.

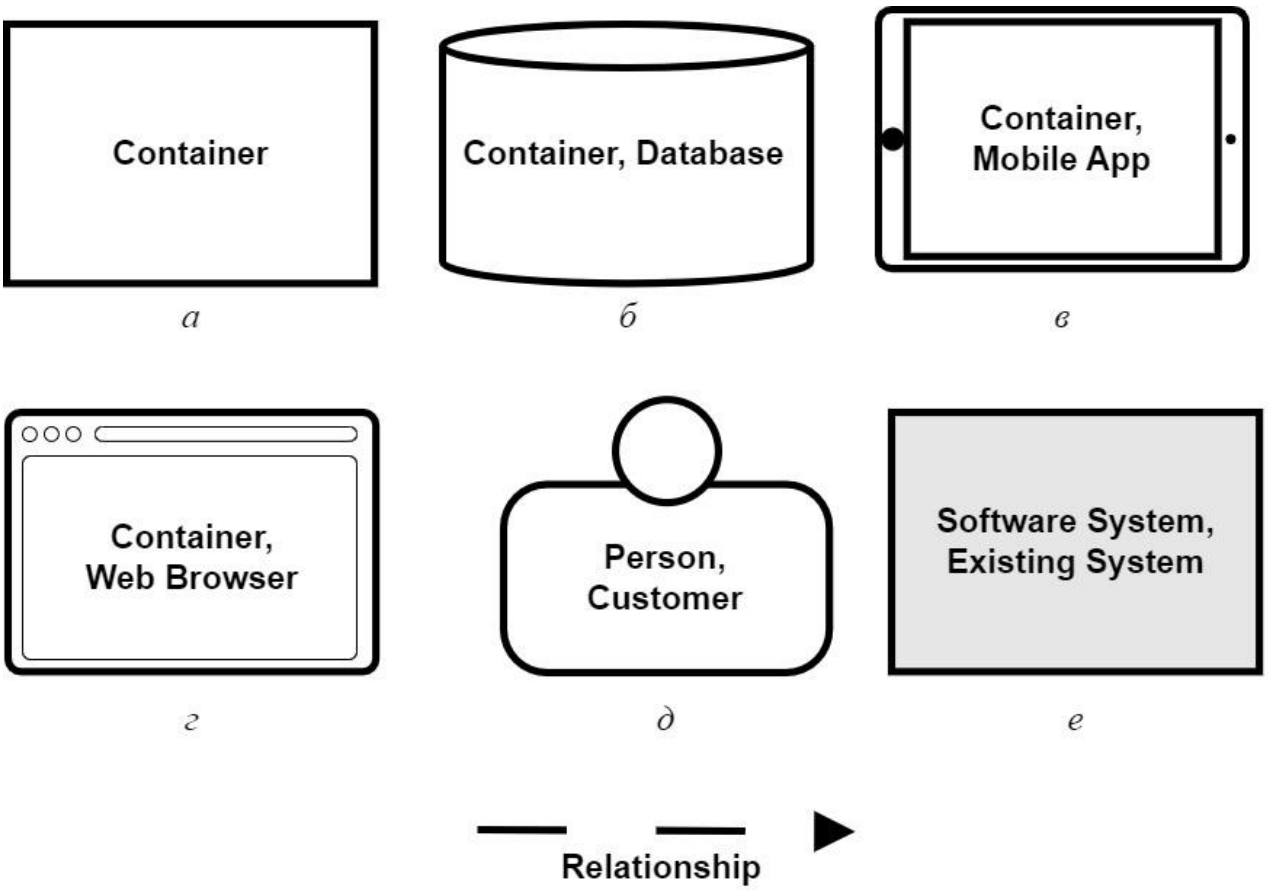
C4 Model – это подход к визуализации архитектуры программного обеспечения, который помогает эффективно описывать системы на различных уровнях абстракции. Он был предложен Саймоном Брауном как упрощенная и последовательная альтернатива традиционным UML-диаграммам. C4 расшифровывается как **Context, Containers, Components, Code** – четыре уровня, которые помогают представлять архитектуру от общего вида до деталей реализации.

Данный подход используется в следующих целях:

- улучшение коммуникации: легко объяснить архитектуру как техническим, так и нетехническим участникам команды;
- документация архитектуры: четко и полно описывает архитектуру системы на всех уровнях;
- совместная работа: упрощает обсуждение архитектуры в команде, помогает выявлять пробелы и улучшать проект;
- планирование и обучение: помогает новым членам команды быстро разобраться в системе, а также полезна для анализа и рефакторинга архитектуры.

---

[1] URL: https://habr.com/ru/companies/piter/articles/730466/.
[2] URL: https://c4model.com/.



а – контейнер; *б* – контейнер БД; *в* – контейнер мобильное приложение;
г – контейнер веб-приложение; *д* – пользователь;
е – внешняя программная система; *ж* – связь

Рисунок 21 – Условные графические обозначения нотации C4 model[1]

Элементы *а–г* представляют собой контейнеры (тип приложения), внутри которых будет исполняться код ПС, а также будет храниться информация (например, веб-приложение, мобильное приложение, СУБД и пр.). При помощи элемента *д* обозначаются пользователи, которые будут взаимодействовать с ПС. Элемент *е* служит для обозначения уже существующего ПС, с которым будет работать проектируемое ПС. Элемент *ж* предназначен для обозначения связи между частями архитектуры. Причем сплошной линией обозначаются синхронные вызовы, а пунктирной – асинхронные.

[1] URL: https://static.structurizr.com/workspace/76749/diagrams/Containers-key.png.

Помимо этого, нотация устанавливает правила именования элементов:

{name}
[{type}]

{description}

где **name** – название элемента (например, Personal Banking Customer, E-mail System и пр.);

type – тип элемента (может принимать следующие значения: Person | Software System | Container: {technology} | Component: {technology});

description – словесное описание элемента.

ПРИМЕР ОПИСАНИЯ ЭЛЕМЕНТОВ АРХИТЕКТУРЫ В НОТАЦИИ C4 MODEL

Personal Banking Customer [Person]	**API Application [Container: Java and spring MVC]**	**E-mail System [Software System]**
A customer of the bank, with personal bank accounts	**Provides Internet banking functionality via a JSON/HTTPS API**	**The internet Microsoft Exchange e-mail system**

Именование связей между частями архитектуры осуществляется несколько иначе (рисунок 22).

Рисунок 22 – Правила и пример именования связей

На рисунке 22 description – название связи (например, отправить идентификационные данные); technology – технология(-и), при помощи которой будет осуществляться отправка или получение данных (например, JSON/HTTPS). Пример именования связей представлен там же (см. рисунок 22) [1]. Суть нотации C4 model заключается в том, что архитектура ПО представляется в виде четырех уровней декомпозиции (рисунок 23), где каждый последующий уровень более детально раскрывает предыдущий.

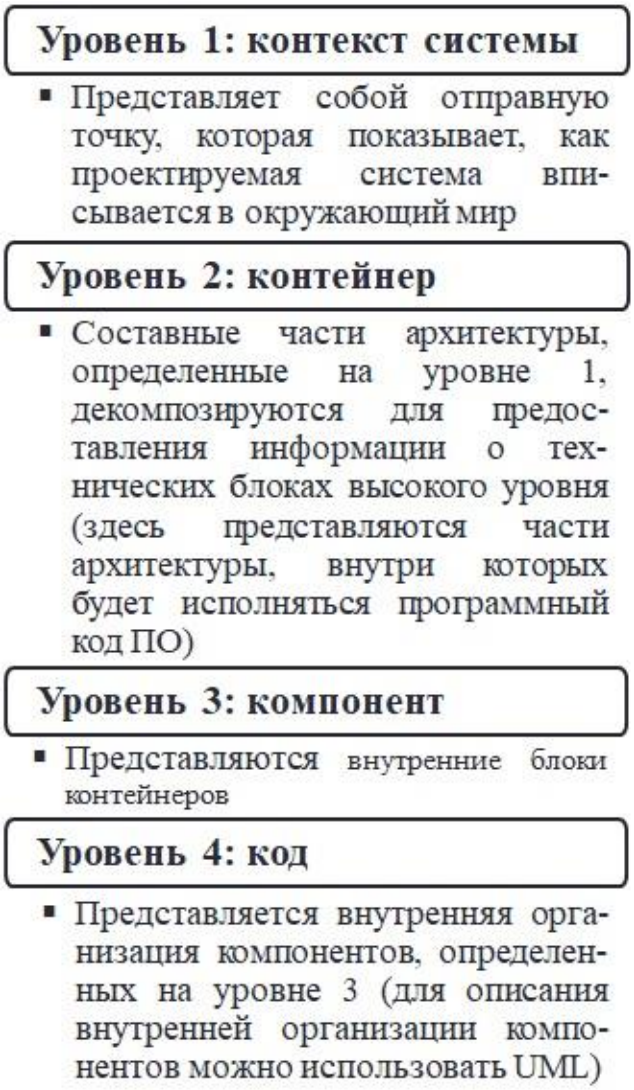


Рисунок 23 – Уровни декомпозиции архитектуры ПО

Контекстный[2], контейнерный[3], компонентный[4] и кодовый[5] уровни декомпозиции архитектуры представлены на рисунках 24–27 соответственно.

[1] URL: https://c4model.com/img/notation-relationship.png.
[2] URL: https://static.structurizr.com/workspace/76749/diagrams/SystemContext.png.
[3] URL: https://static.structurizr.com/workspace/76749/diagrams/Containers.png.
[4] URL: https://static.structurizr.com/workspace/76749/diagrams/Components.png.
[5] URL: https://c4model.com/img/class-diagram.png.

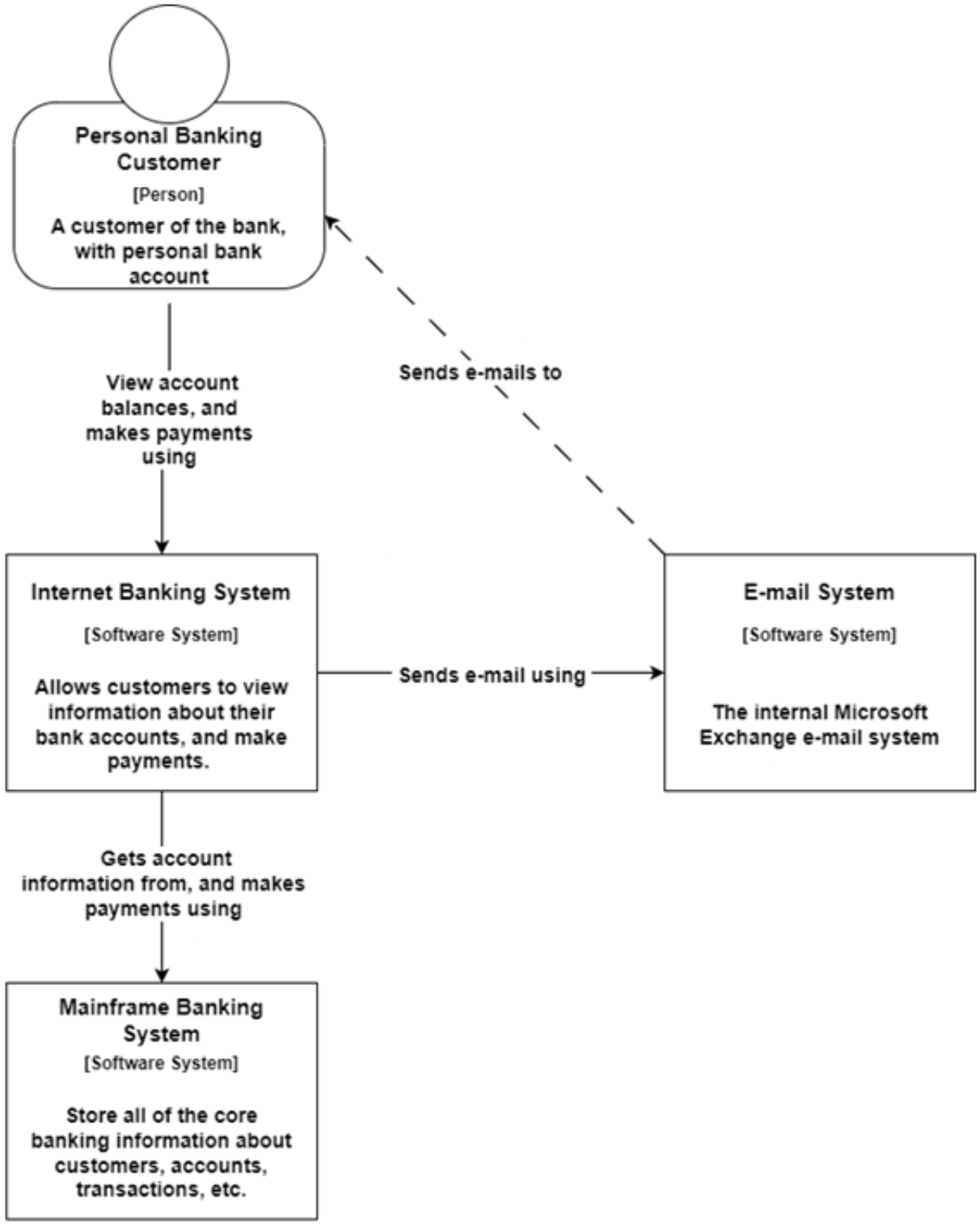
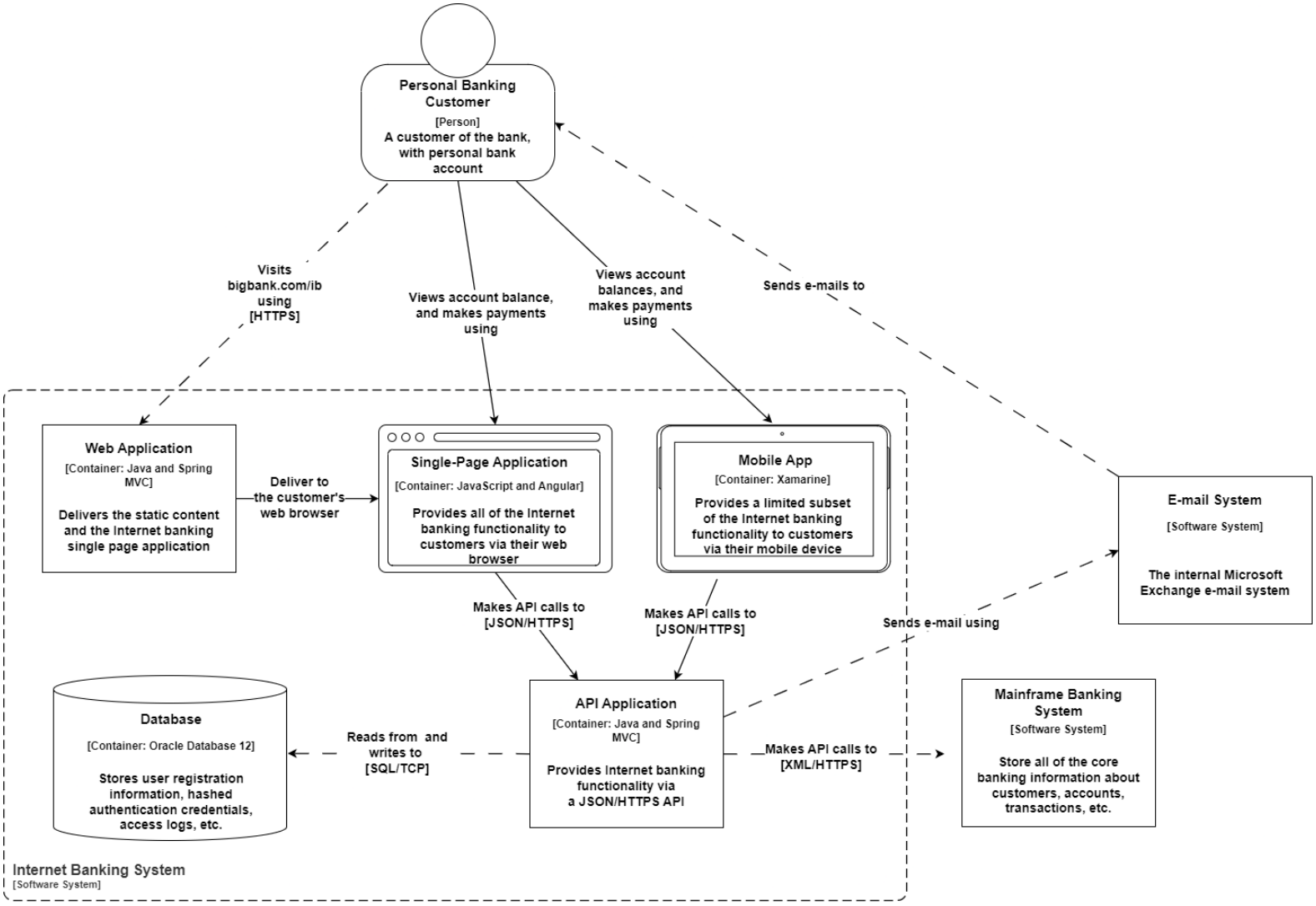


Рисунок 24 – Контекстный уровень представления архитектуры ПО



[Container] Internet Banking System

Рисунок 25 – Контейнерный уровень представления архитектуры ПО

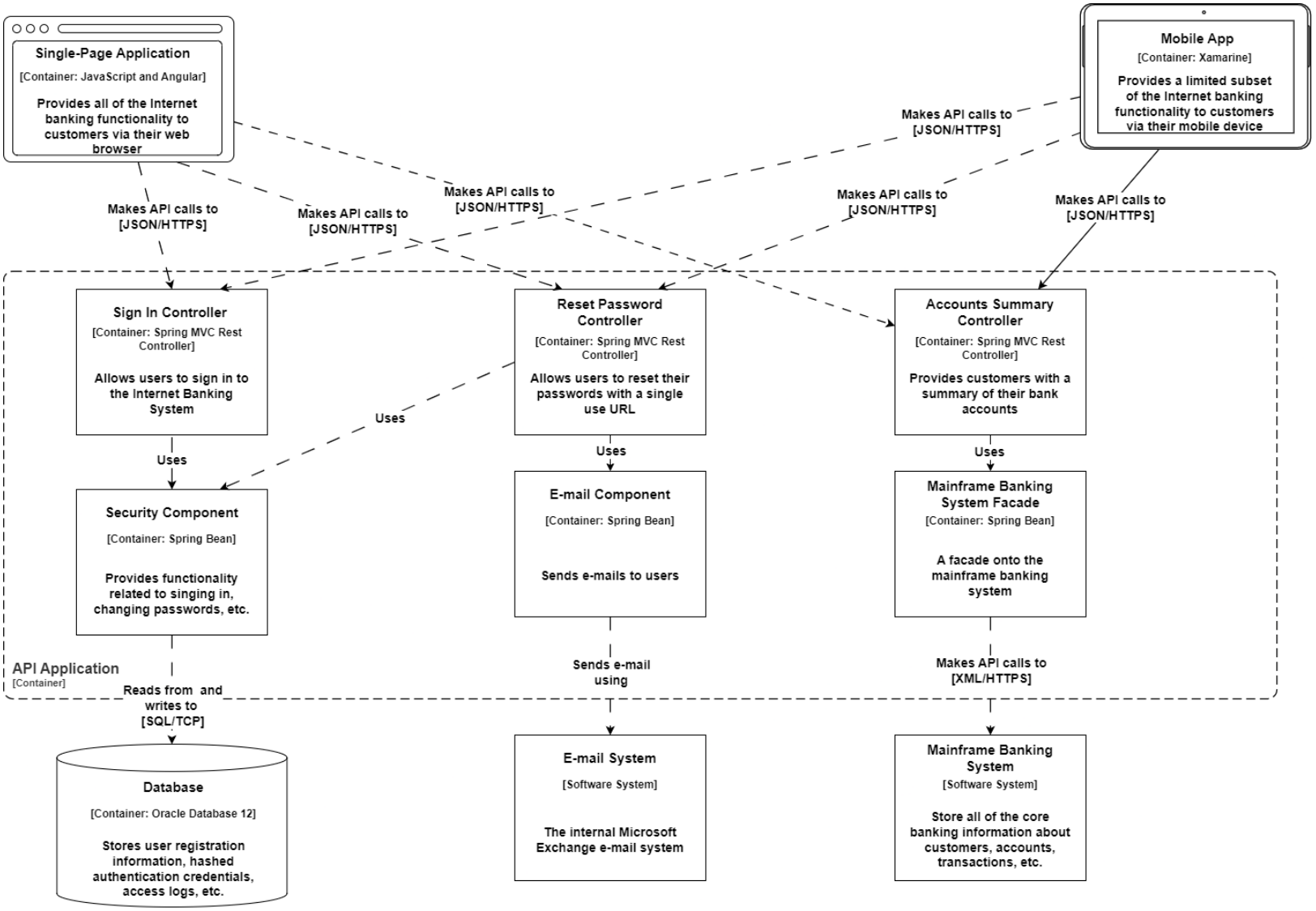


Рисунок 26 – Компонентный уровень представления архитектуры ПО

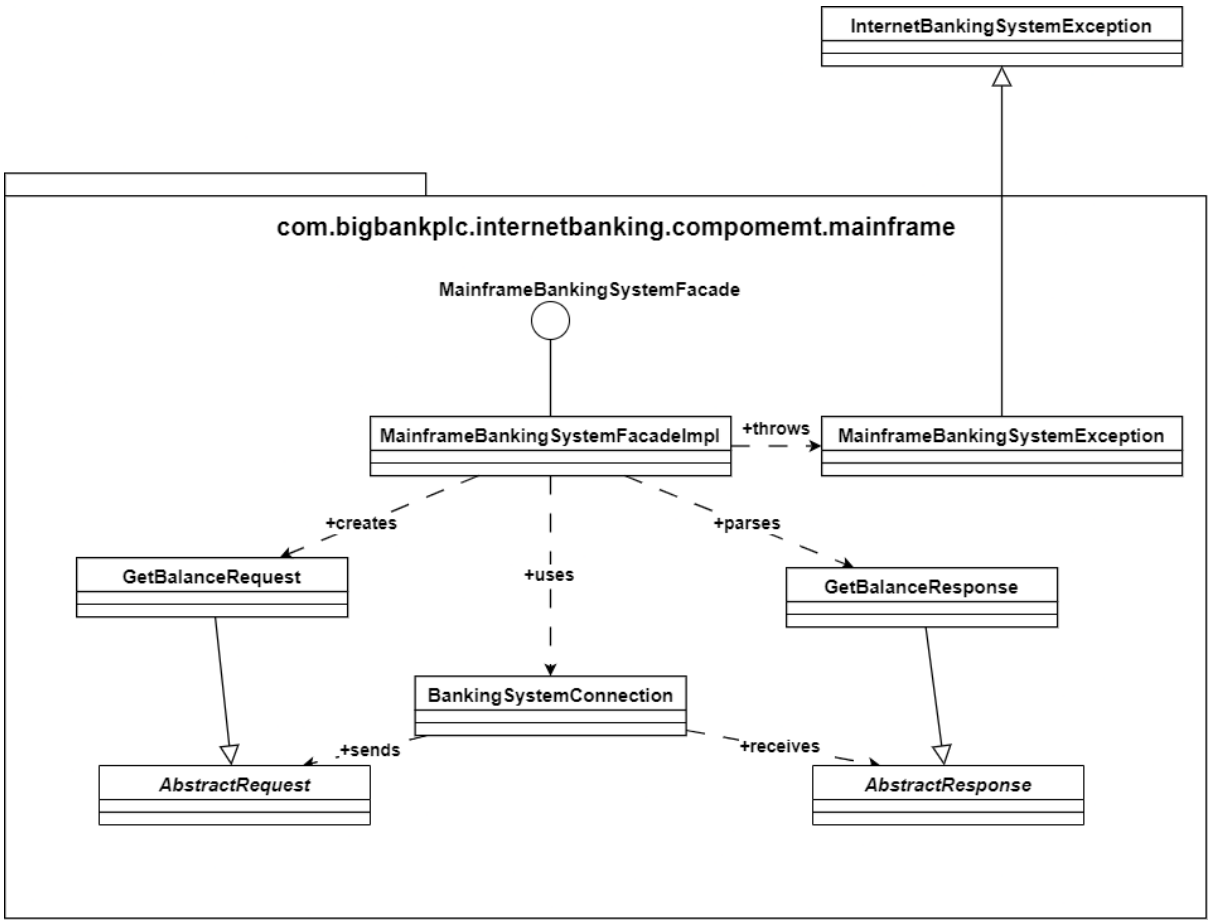


Рисунок 27 – Кодовый уровень представления архитектуры ПО

Краткие теоретические сведения и методические указания к заданию 2

Как было указано в кратких методических указаниях к заданию 1 лабораторной работы № 5, одним из принципов, необходимых для создания требуемой архитектуры ПС, является выбор технологий для реализации. Для того чтобы понять, какие именно технологии подойдут для реализации поставленных задач и нужны ли в принципе готовые решения, необходимо провести *анализ существующих решений* (пример анализа приведен ниже). Использование готовых модулей позволяет получить более качественный код в краткие сроки с меньшими затратами, что влечет за собой увеличение прибыли компании и повышения уровня удовлетворенности заказчиков. Помимо этого, выгоду от использования готовых решений получают и конечные пользователи, т. к. они могут уделить больше времени на реализацию нетривиальных задач заказчика.

После того как проанализированы готовые решения, осуществляется выбор компонентов и технологий для реализации ПС.

При выполнении анализа готовых решений следует представить таблицу сравнения параметров (пример: таблица 16), на основании которых в последующем будет производиться выбор технологий для реализации ПС.

ПРИМЕР АНАЛИЗА ГОТОВЫХ РЕШЕНИЙ

Задача прогнозирования величины урожая сводится к предсказанию значений временного ряда. На основании проведенного анализа литературных источников и интернет-ресурсов был сделан вывод, что на текущий момент для прогнозирования значений временного ряда применяются математические модели под названием «искусственные нейронные сети» (ИНС). Для построения структуры и моделирования работы таких моделей можно использовать большинство высокоуровневых языков программирования. Однако среди множества языков разработчики выделяют язык Python, т. к. именно для него доступны основные версии API различных фреймворков для работы с ИНС.

На основании изучения в сети интернет-ресурсов, посвященных анализу данных и построению ИНС, было принято решение в качестве потенциальных фреймворков для осуществления прогнозирования рассмотреть следующие наиболее распространенные фреймворки: TensorFlow, Keras, PyTorch.

Для того чтобы провести сравнительный анализ и выявить фреймворк, который лучше всего подходит для решения поставленных в рамках лабораторной работы задач, были выбраны следующие характеристики: скорость работы, тестовые модели для обучения, отладка, набор данных, популярность.

Параметр «популярность» был выбран исходя из соображения, что чем более популярен фреймворк, тем большее количество учебных материалов о нем есть в свободном доступе. Результаты сравнения фреймворков представлены в таблице 16.

Таблица 16 – Сравнительный анализ фреймворков для работы с ИНС

Характеристика	**Фреймворк**		
	TensorFlow	**Keras**	**PyTorch**
1	**2**	**3**	**4**
Скорость работы	Высокая	Низкая	Высокая
Тестовые модели для обучения	Да	Да	Да
Отладка	Среди трех выбранных фреймворков обладает наиболее широким набором функций для отладки	Требуется редко ввиду простоты модуля	Присутствует множество функций для отладки кода
Набор данных	Предназначен для работы с большим объемом данных без потери производительности	Оптимален для небольших наборов данных	Предназначен для работы с большим объемом данных без потери производительности

Продолжение таблицы 16

1	2	3	4
Популярность	Среди выбранных фреймворков занимает второе место по популярности	Наиболее популярен среди выбранных фреймворков	Среди выбранных фреймворков занимает третье место по популярности

На основании анализа предметной области и данных таблицы 16 следует вывод, что оптимальным фреймворком для решения задачи прогнозирования величины урожая является TensorFlow.

Далее осуществляется *выбор компонентов и технологий для реализации* ПС. Данный этап является одним из важнейших, т. к. позволяет сократить время разработки, а также повысить качество реализуемого ПС. При обосновании выбора технологий и компонентов для реализации необходимо учитывать особенности функциональности ПС (например, необходимо, чтобы некоторые функции выполнялись параллельно), а также инфраструктуры (например, версия ОС или Android), в которой разработанное ПС будет осуществлять работу.

Краткий пример обоснования компонента приведен ниже.

ПРИМЕР ОБОСНОВАНИЯ ВЫБОРА КОМПОНЕНТОВ

Исходя из требований, определенных в техническом задании, а также результатов анализа готовых решений, было принято решение, что в качестве основного языка разработки будет использоваться объектно-ориентированный язык Java, т. к. он позволяет разрабатывать приложения независимо от конечной пользовательской архитектуры. Это обусловлено тем, что для выполнения кода приложения на Java требуется лишь JVM (Java Virtual Machine – виртуальная машина Java). Байт-код, который получается в результате компиляции приложений, написанных на Java, может быть запущен везде именно благодаря виртуальной машине.

Также стоит описать конкретные особые элементы компонента или технологии, которые отсутствуют в других похожих компонентах и которые будут использоваться в процессе разработки. Например, «*… для реализации полиморфизма в Java не требуется указание дополнительных модификаторов, поскольку Java поддерживает полиморфизм по умолчанию…*»

Описание остальных выбранных компонентов осуществляется аналогичным образом.

Контрольные вопросы к лабораторной работе № 5

1 Что понимают под архитектурой программной системы?

2 Чьи требования необходимо учитывать при проектировании архитектуры?

3 Какими принципами следует руководствоваться при проектировании архитектуры?

4 Что такое архитектурный паттерн?

5 Что такое архитектурный стиль?

6 Какие нотации можно использовать для формализации архитектуры программного обеспечения?

7 Что такое C4 model?

8 Какие уровни проектирования архитектуры выделяют в C4 model?

ЛАБОРАТОРНАЯ РАБОТА № 6
Проектирование и разработка пользовательского интерфейса программного средства

Цели выполнения лабораторной работы:

- спроектировать пользовательский интерфейс с использованием концепций UX и UI;
- построить карту пользовательского интерфейса;
- создать макеты элементов пользовательского интерфейса;
- разработать дизайн элементов пользовательского интерфейса.

Задание	**Этапы выполнения задания**
1 Разработать пользовательский интерфейс программного средства с использованием концепций UX и UI	1 Разработать систему дизайна пользовательского интерфейса. 2 Построить схему логики действий пользователя в интерфейсе программного средства. 3 Разработать макеты всех элементов пользовательского интерфейса. 4 Реализовать пользовательский интерфейс с использованием результатов выполнения предыдущих лабораторных работ
2 Сформировать отчет по лабораторной работе	*Содержание отчета*: Титульный лист (приложение Б) 1 Система дизайна пользовательского интерфейса 2 Схема логики действий пользователя в интерфейсе программного средства 3 Макеты пользовательского интерфейса 4 Дизайн пользовательского интерфейса, включая программный код 5 Выводы

Краткие теоретические сведения и методические указания к заданию 1

Важной частью этапа проектирования в ходе разработки программного обеспечения является проектирование пользовательского интерфейса.

Проектирование пользовательского интерфейса – это создание своеобразной тестовой версии приложения. Это начальный этап разработки, когда выполняется распределение функций приложения по экранам, определяется графическое оформление будущего программного средства, содержимое, элементы

управления и их поведение. Как результат, получается динамичный прототип интерфейса, который можно использовать для тестирования юзабилити или начала разработки приложения. Помочь в проектировании и разработке пользовательского интерфейса может *AI-сервис framer.com.*

В рамках выполнения лабораторной работы будет использоваться следующий *алгоритм проектирования пользовательского интерфейса*:

1 Разработать систему дизайна[1] (англ. design system, style guide, UI-kit, рисунки 28, 29) – набор элементов и их стилей оформления. На данном этапе необходимо выбрать цветовое решение будущего ПС, определить гарнитуры шрифтов и размеры компонентов пользовательского интерфейса, установить размеры сетки (grid), выбрать макет (layout) и пр.

2 Спроектировать логику действий пользователя в ПС. Для представления логики могут быть использованы: карта пользовательского пути (англ. customer journey map, CJM[2]), user flow диаграмма[3], карта разума и т. д.

3 Разработать планы расположения элементов на странице (англ. wireframe). Пример приведен на рисунке 30.

4 Принципы и примеры построения планов расположения элементов на страницах приведены в источниках [11; 12; 13]. Для проектирования планов можно использовать, например, следующее ПО:

- Figma (https://www.figma.com);
- Whimsical (https://whimsical.com);
- inVision (https://www.invisionapp.com/home);
- Adobe XD (https://www.adobe.com/ru/products/xd.html);
- Zeplin (https://zeplin.io).

5 Используя результаты выполнения предыдущих лабораторных работ, реализовать пользовательский интерфейс, используя UI-библиотеки (-фреймворки)[4].

6 Представить результаты реализации пользовательского интерфейса[5] в виде иллюстраций и их описания.

Если ПС предусматривает несколько пользователей, то стоит ознакомиться с примером user flow диаграммы по ссылке https://www.behance.net/gallery/113540759/UIUX-Portfolio.

[1] URL: https://www.behance.net/gallery/-93950011/User-Design-System-Nayi-Disha.
[2] URL: https://www.unisender.com/ru/blog/wp-content/uploads/2020/09/2-cjm.jpg.
[3] URL: https://nicolekendrot.com/wp-content/uploads/2014/07/centz-user-flow.jpg.
[4] URL: https://habr.com/ru/articles/706546.
[5] URL: https://developer.mozilla.org/ru/docs/Learn/CSS/CSS_layout/Responsive_Design.

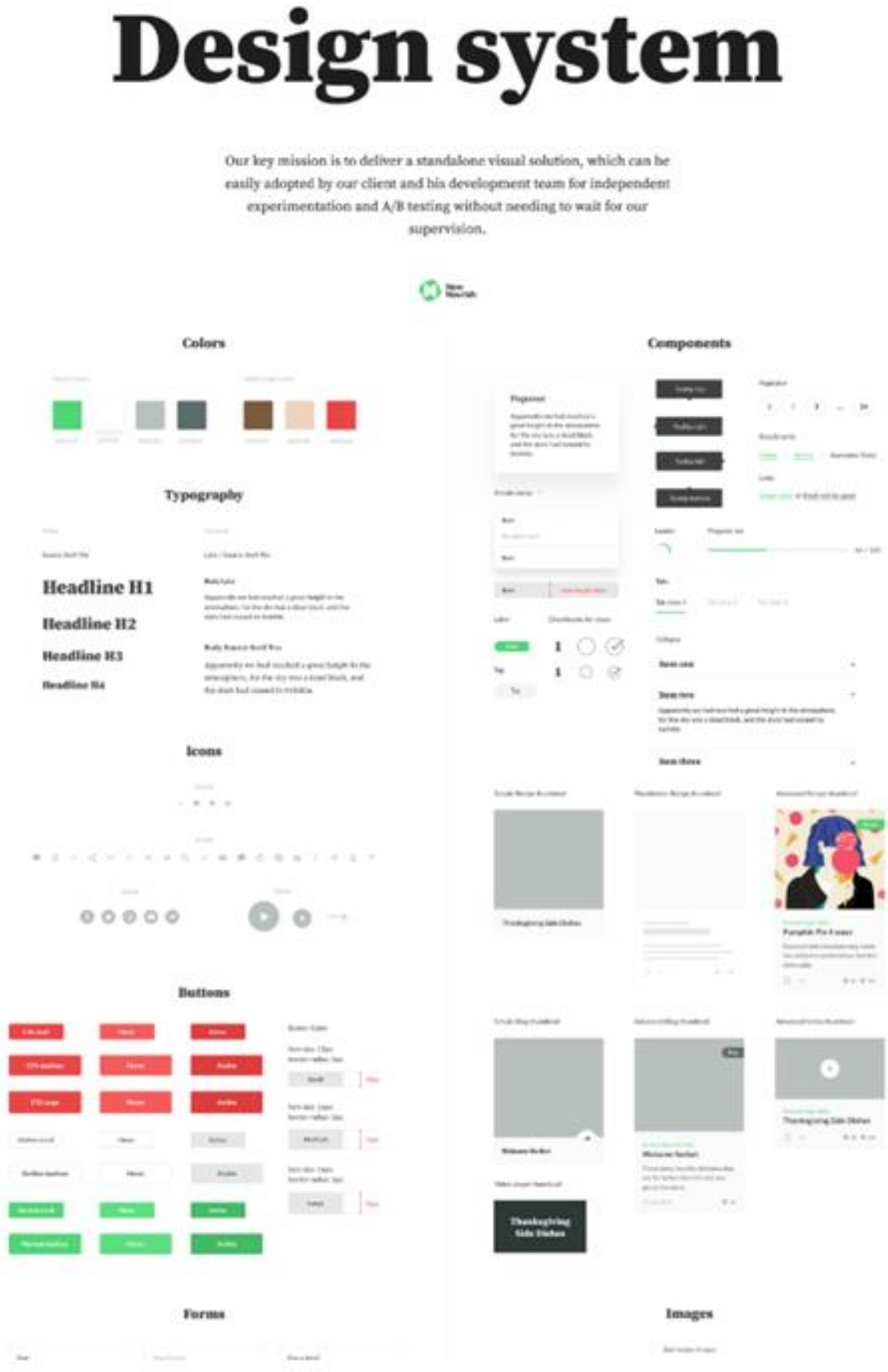


Рисунок 28 – Система дизайна (пример 1)[1]

[1] URL: https://mir-cdn.behance.net/v1/rendition/project_modules/fs/12ca6363768111.5abcc63548335.png.

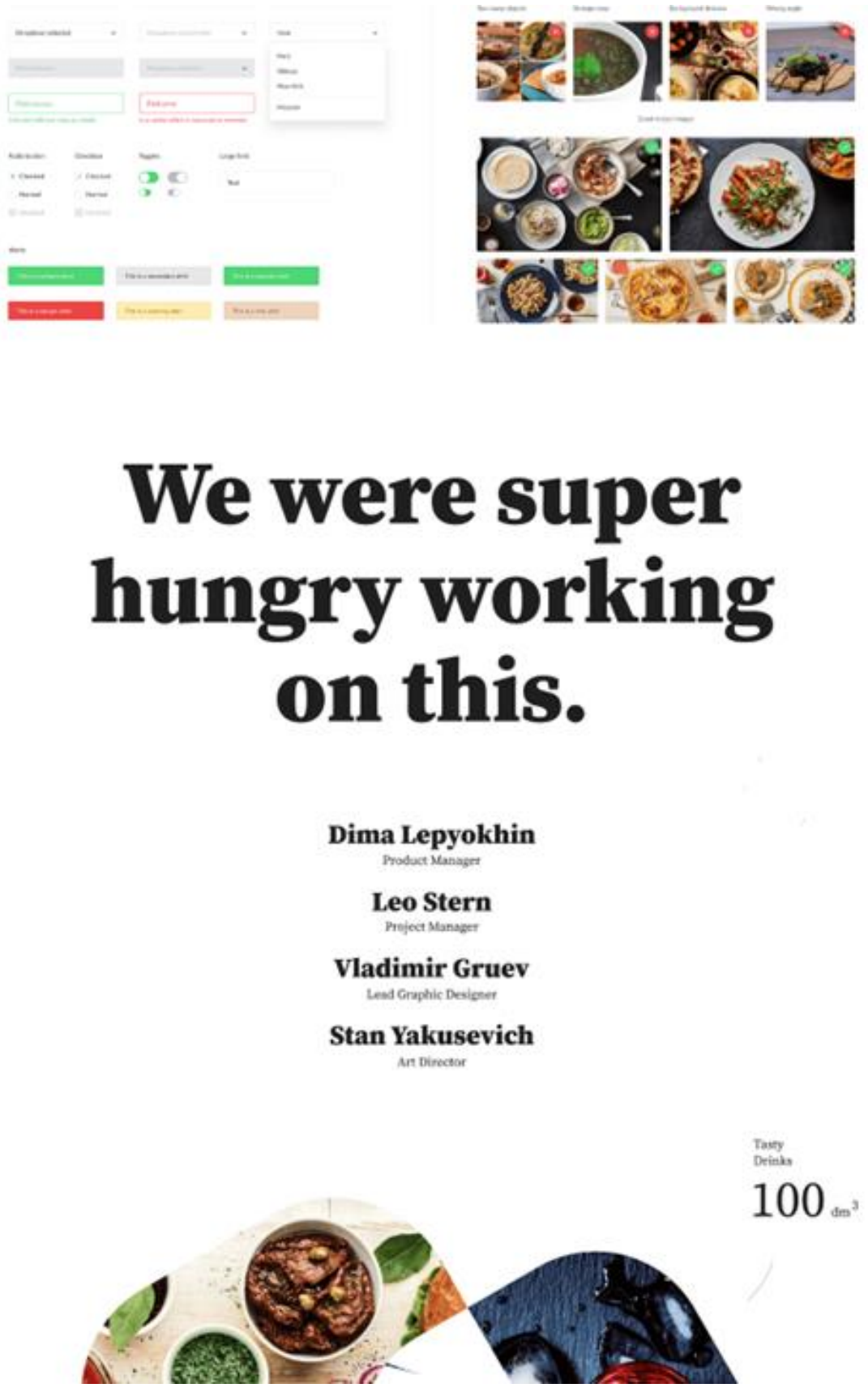


Рисунок 28, лист 2[1]

[1] URL: https://mir-s3-cdn-cf.behance.net/project_modules/fs/5ac1d363768111.5abcc585b81c7.png.

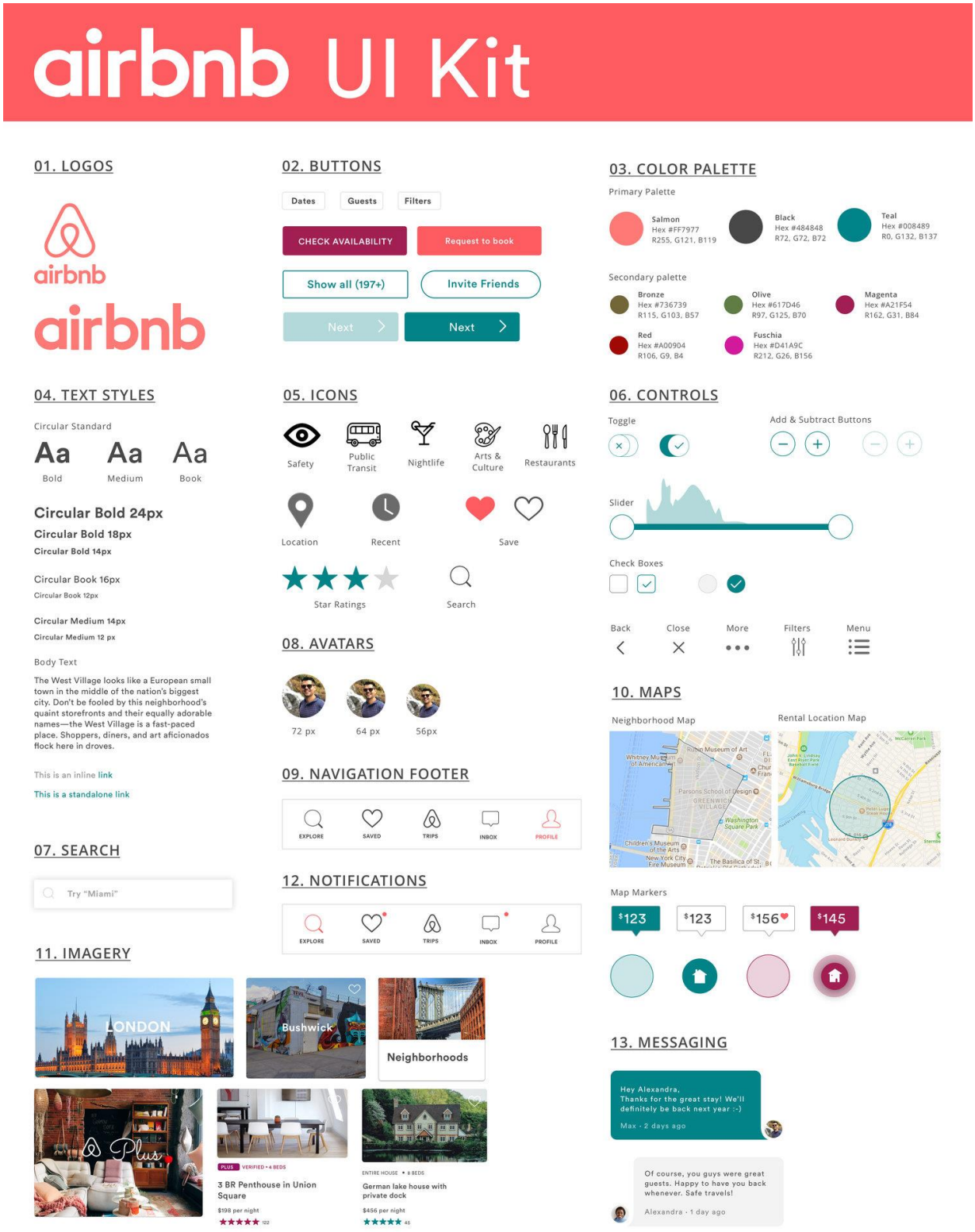


Рисунок 29 – Система дизайна (пример 2)[1]

[1] URL: https://images.squarespace-cdn.com/content/v1/5588c61de4b062bc4c6b34c5/15251093570515RSSDTQPCNBGTWTR1JM5/Airbnb+UI+Kit.jpg?format=1500w.

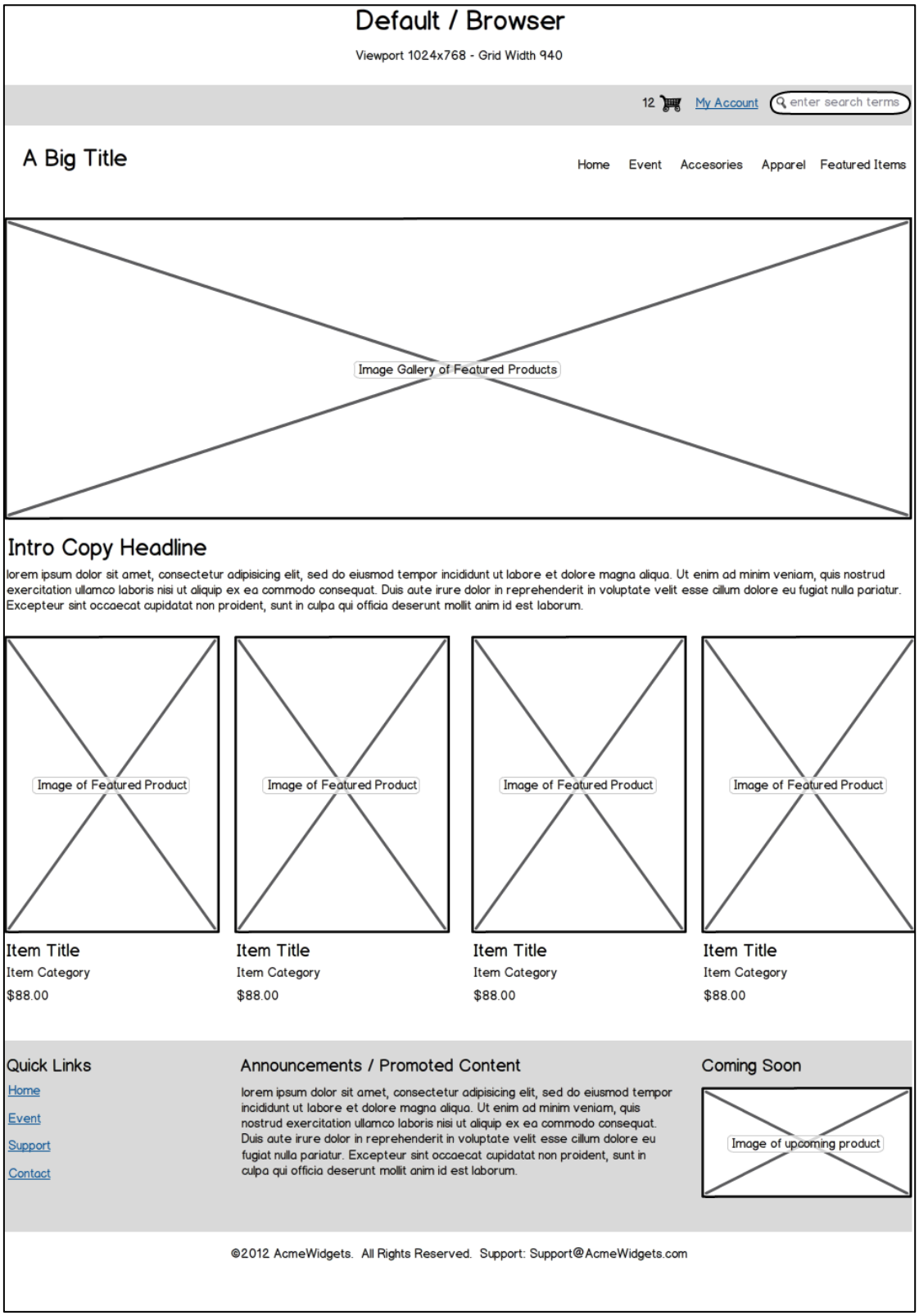


Рисунок 30 – Пример плана расположения элементов для различных типов устройств[1]

[1] URL: https://file.mockplus.com/image/2017/11/e8bbb60f-f2da-49cf-9c3b-64dbefe77b27.png.

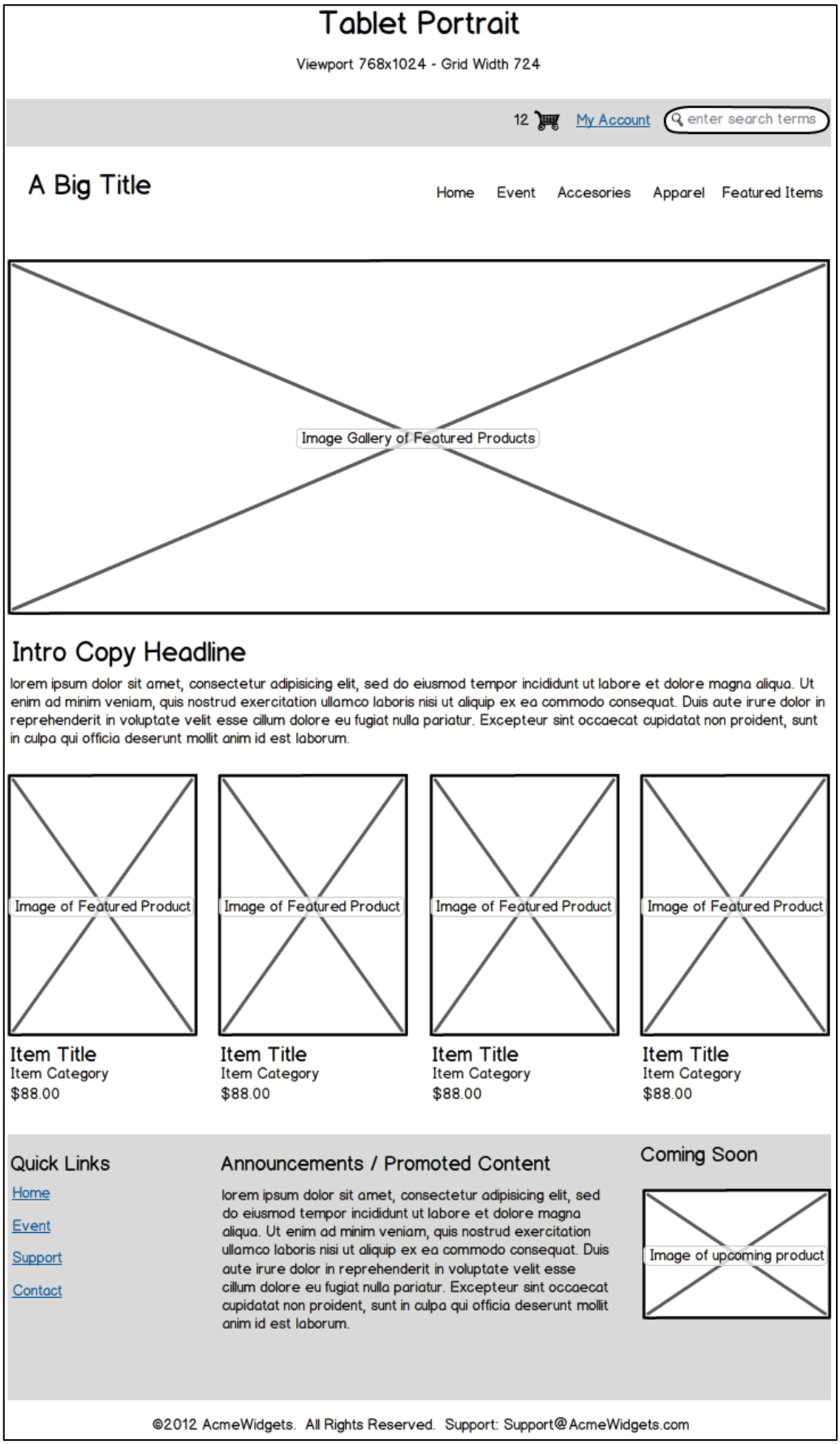


Рисунок 30, лист 2

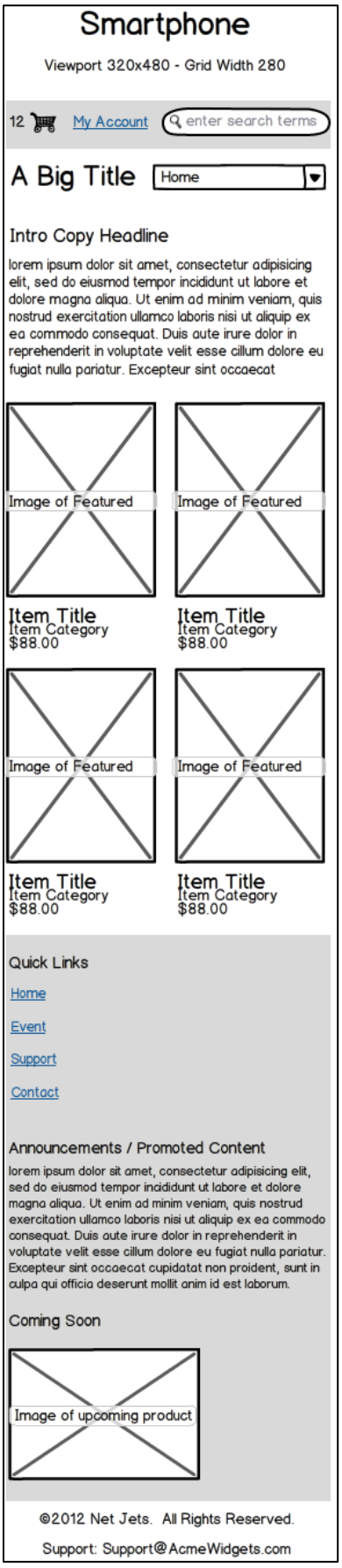


Рисунок 30, лист 3

Контрольные вопросы к лабораторной работе № 6

1 Какие задачи решают на этапе проектирования ПО?

2 Что такое пользовательский интерфейс и какие технологии используются для его реализации?

3 Что представляет собой процесс проектирования пользовательского интерфейса?

4 Какие компоненты включает пользовательский интерфейс?

5 Каковы критерии качественного интерфейса пользователя?

6 Что такое графический интерфейс пользователя и какие элементы он включает?

7 Какие бывают виды графического интерфейса?

8 Какие существуют популярные инструменты для создания прототипов и дизайна интерфейса?

9 Что такое система дизайна и что такое адаптивный дизайн?

10 Каким образом можно представить логику действий пользователя в интерфейсе?

11 Что такое карта пользовательского пути?

ЛАБОРАТОРНАЯ РАБОТА № 7
Организация системы хранения данных

Цели выполнения лабораторной работы:

- выбрать модель данных и организовать хранение данных;
- разработать информационную модель предметной области.

Задание	**Этапы выполнения задания**
1 Организовать хранение данных	1 Разработать и описать структуру данных для хранения информации. Независимо от модели данных должны быть представлена схема организации структуры для хранения данных и описаны использованные типы данных. 2 Построить концептуальную модель данных (только для реляционной модели данных). 3 Построить логическую модель данных и обосновать соответствие ее правилам 3НФ (только для реляционной модели данных). 4 Построить физическую модель данных (только для реляционной модели данных)
2 Сформировать отчет по лабораторной работе	*Содержание отчета:* Титульный лист (приложение Б) 1 Описание структуры данных 2 Концептуальная модель данных 3 Логическая модель данных 4 Физическая модель данных 5 Выводы

Краткие теоретические сведения и методические указания к заданию 1

Выбор модели данных для хранения информации зависит от поставленной задачи, требований заказчика и множества других факторов. На сегодняшний день существует большое количество моделей данных, наиболее популярной среди которых является реляционная.

В контексте данных методических указаний будет рассмотрено моделирование хранения данных с использованием реляционной модели данных.

В отчете по лабораторной работе (независимо от того, какая модель данных была выбрана) в обязательном порядке представить:

- схему организации данных;
- описание выбранных типов данных в табличном виде;
- описание всех дополнительных элементов (под дополнительными элементами подразумеваются любые инструкции языков определения и манипулирования данными для выбранной СУБД).

Информационная модель данных предназначена для представления семантики предметной области в терминах субъективных средств описания – сущностей, атрибутов, идентификаторов сущностей, супертипов, подтипов и т. д.

Информационная модель предметной области базы данных содержит следующие основные конструкции:

- диаграммы «сущность – связь» (Entity-Relationship Diagrams);
- определения сущностей;
- уникальные идентификаторы сущностей;
- определения атрибутов сущностей;
- отношения между сущностями;
- супертипы и подтипы.

Для разработки информационной модели предметной области, основываясь на архитектуре ANSI/SPARC, выделяют следующие уровни абстракции базы данных (БД): концептуальный, логический и физический уровни (рисунок 31).

Дополнительная информация по архитектуре ANSI/SPARC доступна по ссылке: http://andrey-2119163.narod.ru/management_data/lection_02.htm.

Разработка информационной модели включает следующие этапы проектирования БД:

1 *Концептуальное (инфологическое) проектирование (Conceptual design)*. На данном этапе осуществляется анализ предметной области и выявление ее основных сущностей, а также связей между ними.

Итогом является концептуальная модель БД в виде ER-модели.

Обычно концептуальная модель включает в себя:

- описание информационных объектов или понятий предметной области и связей между ними;
- описание ограничений целостности, т. е. требований к допустимым значениям данных и к связям между ними.

2 *Логическое (даталогическое) проектирование (Logical Design*). В ходе данного этапа происходит преобразование концептуальной модели БД в логическую.

Дополнительная информация по ER-модели доступна по ссылке: https://www.bestprog.net/ru/2019/01/24/the-concept-of-er-model-the-concept-of-essence-and-communication-attributes-attribute-types-ru/#q02.

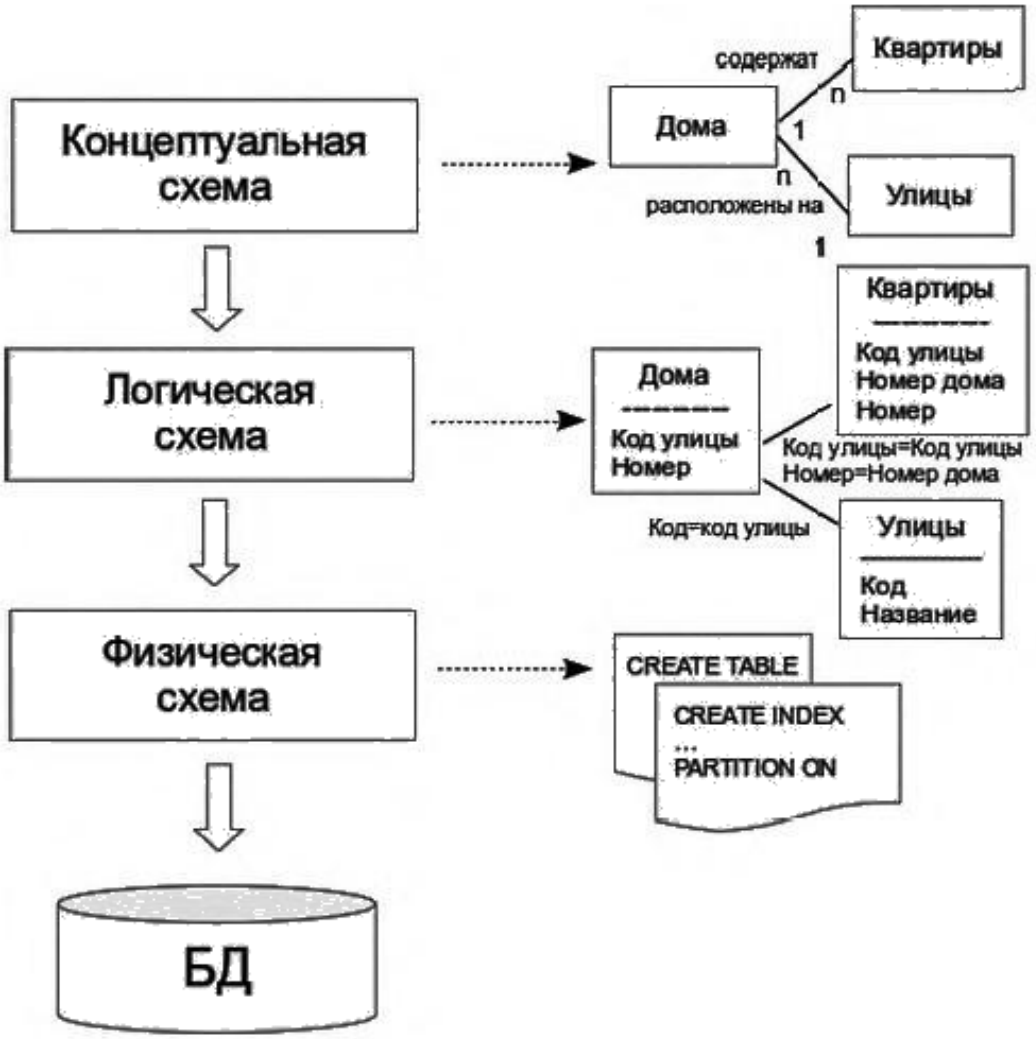


Рисунок 31 – Уровни абстракции БД

Отличительной чертой логической схемы БД является указание атрибутов сущностей и их типа (ключевые, неключевые, многозначные, производные, простые, составные).

Результатом данного этапа проектирования является логическая схема БД, выполненная в соответствии с выбранной нотацией.

Дополнительная информация по существующим нотациям моделирования схем БД доступна по ссылке: https://ru.wikipedia.org/wiki/ER-модель#Графические_нотации_(диаграммы).

Отдельно стоит отметить процесс нормализации данных, который осуществляется перед этапом физического проектирования. Как отмечает К. Дж. Дейт, общее назначение процесса нормализации заключается в следующем [14]:

- исключение некоторых типов избыточности;
- устранение некоторых аномалий обновления;
- разработка проекта базы данных, который является достаточно «качественным» представлением реального мира, интуитивно понятным и может служить хорошей основой для последующего расширения;
- упрощение процедуры применения необходимых ограничений целостности.

На каждом этапе нормализации в БД добавляется новая таблица.

3 *Физическое проектирование (Physical Design).* Это последний этап проектирования БД, на котором осуществляется преобразование логической схемы в схему БД для конкретной СУБД. Различные СУБД могут включать в себя ограничения на именование объектов БД, ограничения на поддерживаемые типы данных и пр. Кроме того, при физическом проектировании специфика СУБД включает выбор решений, связанных с физической средой хранения данных (выбор методов управления дисковой памятью, разделение БД по файлам и устройствам, методов доступа к данным), создание индексов и т. д. Следствием физического проектирования может быть ER-диаграмма БД с указанием конкретных типов данных или скрипт генерации БД.

Процесс нормализации, осуществляющийся перед этапом физического проектирования, является отдельной ступенью в проектировании БД.

Нормализация – это процесс структурирования реляционной базы данных в соответствии с правилами нормальных форм для уменьшения избыточности данных и улучшения целостности данных.

Для лучшего понимания сущности процесса нормализации данных приведем основные определения.

Определение 1. Атомарность – это степень детализации и структурирования информации в базе.

Определение 2. Функциональная зависимость. В отношении R атрибут Y функционально зависит от атрибута X (X и Y могут быть составными) в том и только в том случае, если каждому значению X соответствует в точности одно значение Y: R.X -> R.Y.

Определение 3. Полная функциональная зависимость. Функциональная зависимость R.X -> R.Y называется полной, если атрибут Y не зависит функционально от любого точного подмножества X.

Определение 4. Транзитивная функциональная зависимость. Функциональная зависимость R.X -> R.Y называется транзитивной, если существует такой атрибут Z, что имеются функциональные зависимости R.X -> R.Z и R.Z -> R.Y и отсутствует функциональная зависимость R.Z -> R.X.

Определение 5. Неключевым атрибутом называется любой атрибут отношения, не входящий в состав первичного ключа.

Определение 6. Два или более атрибута **взаимно независимы**, если ни один из этих атрибутов не является функционально зависимым от других.

Определение 7. Отношение находится в **первой нормальной форме (1NF, 1НФ)** при атомарности значений всех его атрибутов.

Определение 8. Отношение находится во **второй нормальной форме (2NF, 2НФ)**, если оно находится в *первой нормальной форме* и каждый неключевой атрибут находится в полной функциональной зависимости от ключа.

Определение 9. Отношение находится в **третьей нормальной форме (3NF, 3НФ)**, если данное отношение находится во *второй нормальной форме* и каждый неключевой атрибут находится в нетранзитивной зависимости от первичного ключа.

Более подробное описание проблем и процесса нормализации данных представлено по ссылке: https://practicum.yandex.ru/blog/chto-takoe-normalizaciya-dannyh.

Рассмотрим процесс проектирования информационной модели на примере аренды помещений.

Арендодатель хранит информацию о помещениях и арендаторах (таблицы 17, 18).

Разработка информационной модели предметной области начинается с этапа концептуального проектирования.

Концептуальное проектирование. На первом этапе проектирования схемы БД необходимо выделить основные сущности предметной области, а также определить связи между ними.

Из таблицы 17 можно выделить сущность *ПОМЕЩЕНИЕ*, а из таблицы 18 – *АРЕНДАТОР* и *ДОГОВОР АРЕНДЫ*. Представим данные сущности в виде ER-модели (рисунок 32).

Таблица 17 – Данные о помещениях

Инвентарный номер помещения	Этаж	Номер помещения[1]	Площадь, м^2	Количество окон	Наличие кондиционера (да/нет)	Мебель	Стоимость аренды в месяц, дол. США
1001256	7	284	85	4	Да	Компьютерные столы – 6 шт., письменные столы – 2 шт., кресла с поворотным механизмом и регулятором высоты –10 шт., шкаф-купе –1 шт.	300
1001278	17	245	175	8	Да	Компьютерные столы – 6 шт., письменные столы – 2 шт., кресла с поворотным механизмом и регулятором высоты – 10 шт., шкаф-купе – 1 шт., диван – 3 шт., журнальный столик – 1 шт., кухонный гарнитур – 1 шт., микроволновка – 1 шт., кофемашина – 1 шт.	800
1001256	1	140	69	4	Нет	Компьютерные столы – 6 шт., письменные столы – 2 шт., кресла с поворотным механизмом и регулятором высоты – 10 шт., шкаф-купе – 1 шт.	275

Таблица 18 – Данные об арендаторах и аренде помещений

Номер договора аренды	Ф. И. О. арендатора	Номер паспорта	E-mail	Номер для связи[2]	Инвентарный номер арендуемого помещения	Дата начала аренды	Срок аренды, сут
24123	Смирнов Александр Валерьевич	AB1870041	austin1988@gmail.com	+375291234567	1001256	02.05.2020	30
55342	Касперович Наталья Игоревна	KH2105396	torrance2006@hotmail.com	+375291876543	1001256	12.06.2020	30
78298	Шиманчук Дмитрий Александрович	MP4007108	margie.rolfs@gmail.com	+375299876543	1001278	02.09.2020	30

[1] Условимся, что номер помещения зависит от этажа.
[2] Номер для связи зависит от номера паспорта.

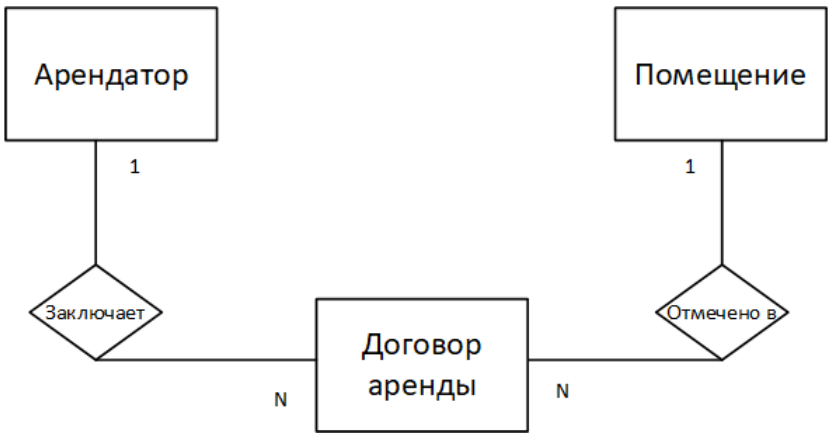


Рисунок 32 – ER-модель предметной области

Логическое проектирование. Следующим этапом является логическое проектирование. В рамках данного этапа нужно выделить атрибуты сущностей. Для представления модели БД воспользуемся нотацией Чена.

Детальнее о нотации Чена по ссылке: https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema7/tema7_5. Также стоит ознакомиться с дополнительной информацией о модели сущность – связь по ссылке: https://foreva.susu.ru/courses/db/lecture3.pdf.
Для построения модели предметной области можно воспользоваться сервисом по ссылке: https://erdplus.com/ или MS Visio, выбрав в качестве шаблона «Нотация Чена для моделирования баз данных».

Рассмотрим сущность *ПОМЕЩЕНИЕ*. Из таблицы 17 видно, что ключевым атрибутом данной сущности является поле *Инвентарный номер помещения*. Остальные атрибуты являются простыми. Отображение сущности *ПОМЕЩЕНИЕ* в нотации Чена представлено на рисунке 33.

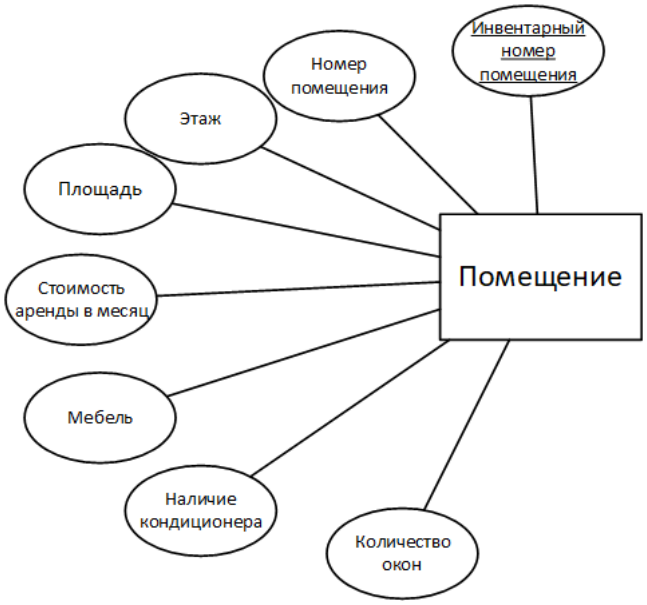


Рисунок 33 – Сущность *ПОМЕЩЕНИЕ*

Теперь представим отображение сущностей *АРЕНДАТОР* и *ДОГОВОР АРЕНДЫ*. Так как таблица 18 одновременно содержит информацию двух сущностей, нужно отделить атрибуты одной сущности от другой. К сущности *АРЕНДАТОР* отнесем поля: *Фамилия, Имя, Отчество, Номер паспорта, Номер для связи, E-mail*. Оставшиеся поля отнесем к сущности *ДОГОВОР АРЕНДЫ*.

Стоит отметить тот факт, что однозначно идентифицировать различные кортежи сущности *АРЕНДАТОР* невозможно, т. к. в ней отсутствует ключевой атрибут. Логичным, на первый взгляд, могло бы быть выделение в качестве первичного ключа поля *Номер паспорта*, однако данный атрибут на самом деле не является уникальным.

Другим вариантом может быть формирование составного первичного ключа из полей *Фамилия, Имя, Отчество*. Но даже эта комбинация полей может оказаться не единственной в своем роде. В таком случае необходимо использовать суррогатный первичный ключ, который получается путем добавления в сущность нового атрибута (в нашем случае поля *ID_Арендатора*). Итоговые представления сущностей *АРЕНДАТОР* и *ДОГОВОР АРЕНДЫ* показаны на рисунках 34 и 35 соответственно.

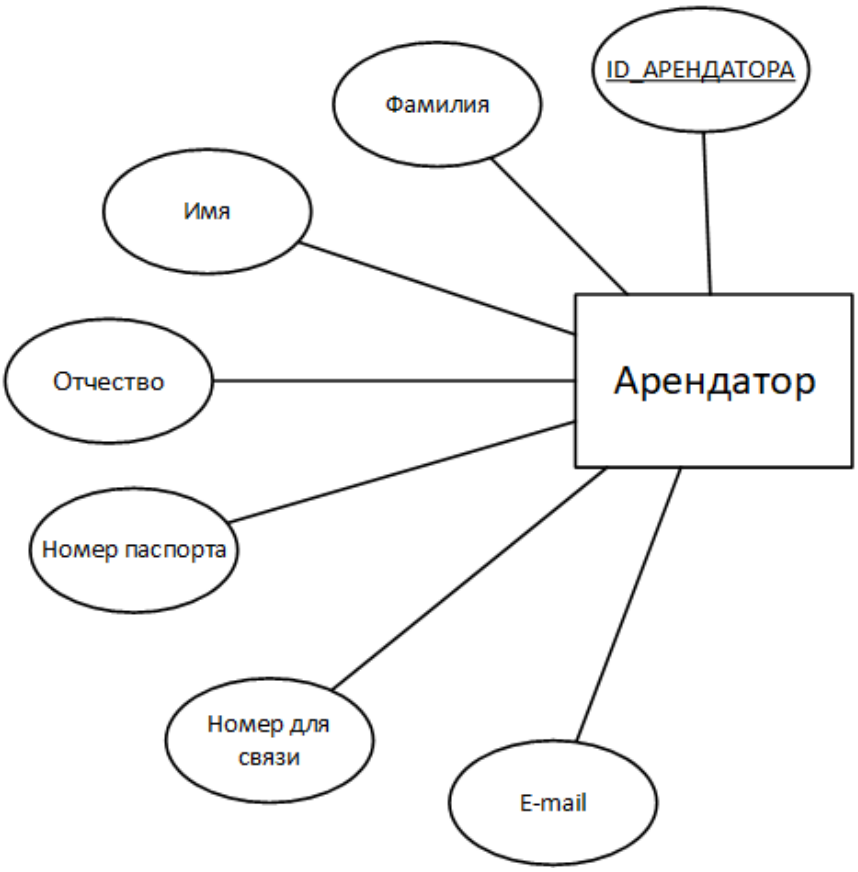


Рисунок 34 – Сущность *АРЕНДАТОР*

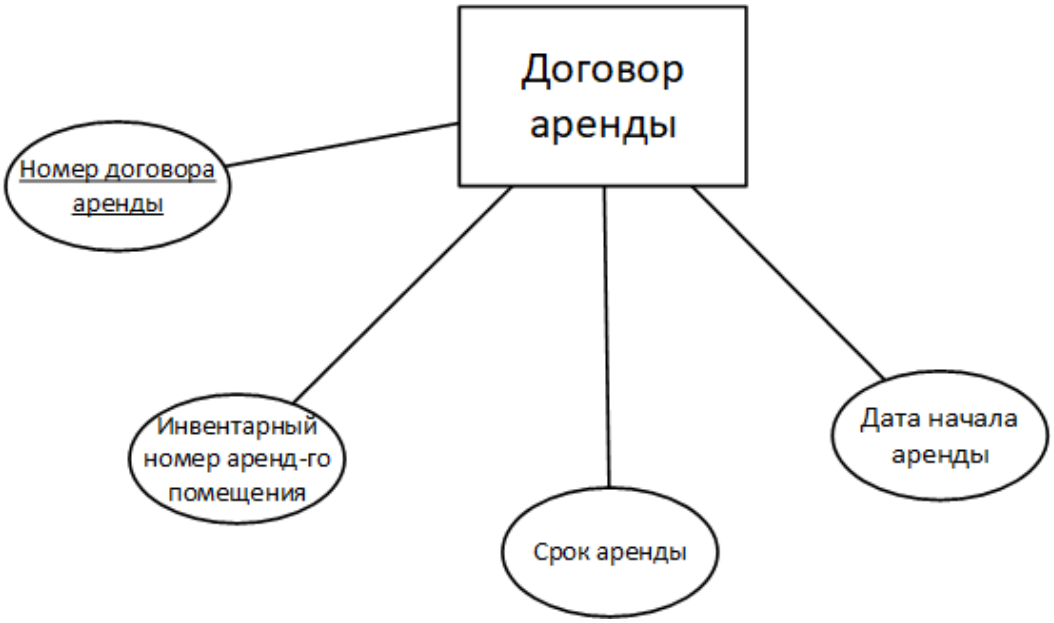


Рисунок 35 – Сущность *ДОГОВОР АРЕНДЫ*

Итоговая логическая модель, включающая связи между указанными ранее сущностями, показана на рисунке 36.

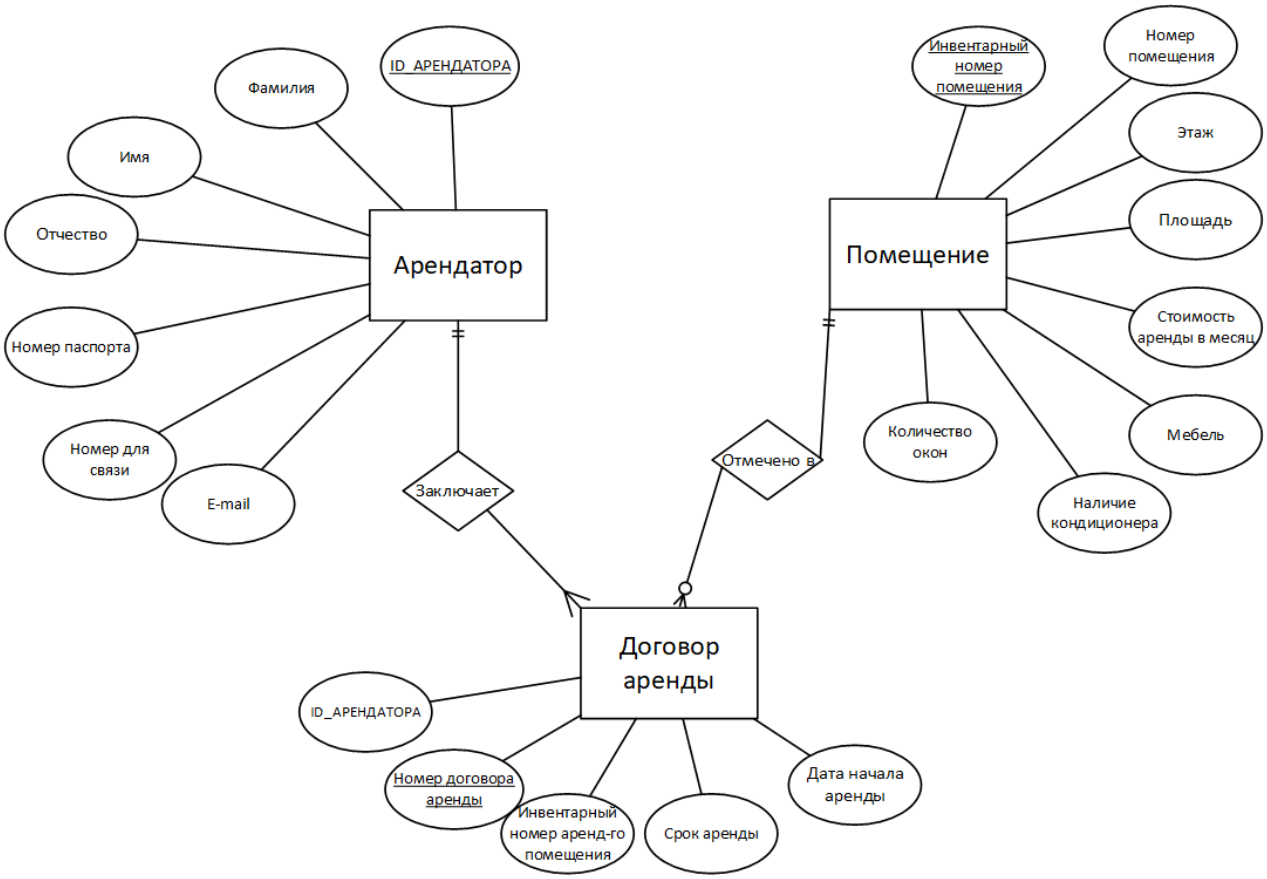


Рисунок 36 – Логическая схема БД

Прежде чем приступить к этапу физического проектирования, оценим степень нормализации спроектированной логической схемы.

Основываясь на определении 7, проанализируем состав атрибутов. Как видно из схемы БД, все атрибуты являются атомарными, поэтому можно сделать вывод, что схема соответствует 1НФ.

Для приведения схемы к 2НФ необходимо, чтобы все неключевые атрибуты зависели только от всего составного первичного ключа, но не зависели ни от одной его части. Так как в сущностях отсутствуют составные первичные ключи, можно говорить о соответствии схемы БД 2НФ.

Теперь можно проверить, соответствует ли схема 3НФ. Для этого необходимо, чтобы все неключевые атрибуты нетранзитивно зависели от первичного ключа.

В сущности *АРЕНДАТОР* присутствует транзитивная зависимость между атрибутами *ID_Арендатора* и *Номер для связи*:

ID_АРЕНДАТОРА → Номер паспорта
Номер паспорта → Номер для связи $\Rightarrow$ **ID_АРЕНДАТОРА → Номер для связи**

Такая же транзитивная зависимость существует между атрибутами *Инвентарный номер помещения* и *Номер помещения* сущности *ПОМЕЩЕНИЕ*:

Инвентарный номер помещения → Этаж
Этаж → Номер помещения $\Rightarrow$ **Инвентарный номер помещения → Номер помещения**

Чтобы устранить данные зависимости, необходимо декомпозировать на две каждую из сущностей *АРЕНДАТОР* и *ПОМЕЩЕНИЕ*. В итоге декомпозиции получим новую схему БД (рисунок 37).

После удаления транзитивных зависимостей схема соответствует 3НФ.

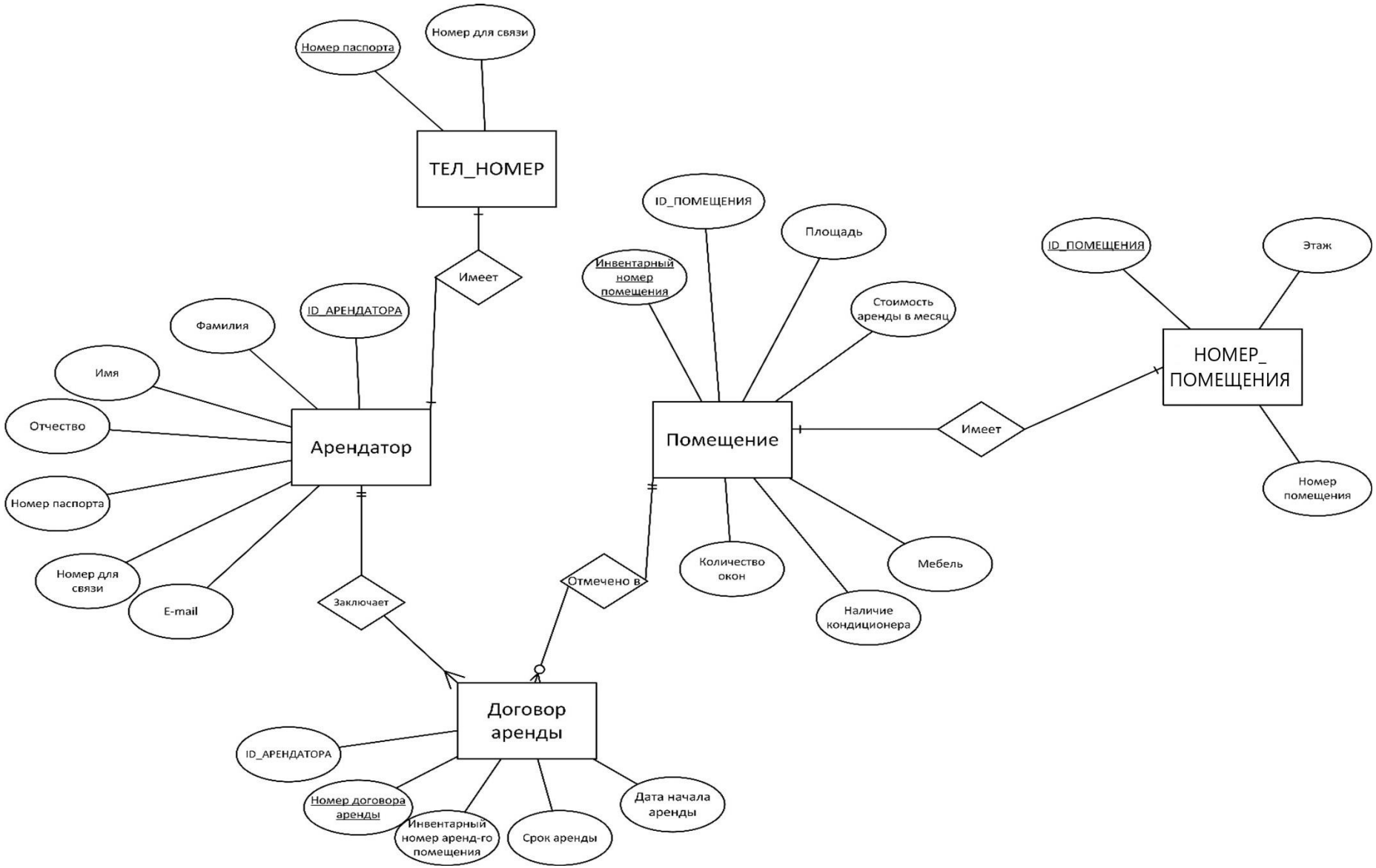


Рисунок 37 – Схема БД после декомпозиции

Физическое проектирование. Теперь перейдем к последнему этапу – физическое проектирование. На данном этапе необходимо выбрать конкретную СУБД и преобразовать логическую схему БД в физическую. Для примера воспользуемся СУБД MySQL, а в качестве визуального средства проектирования будем использовать MySQL Workbench. В итоге преобразования получим физическую схему БД, представленную на рисунке 38.

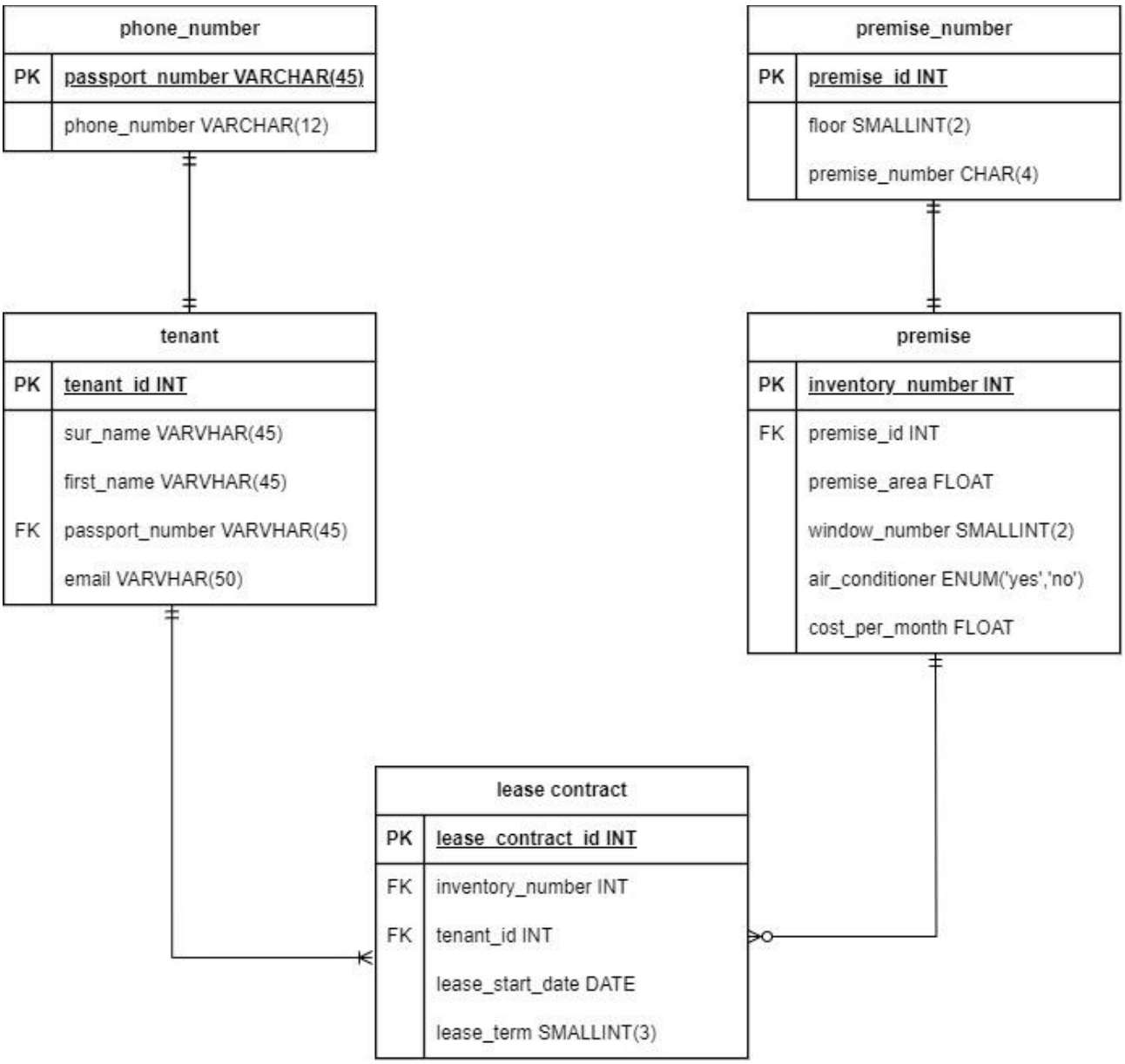


Рисунок 38 – Физическая схема БД

Для демонстрации разработки информационной модели предметной области помимо ER-диаграммы необходимо представить текстовое описание сущностей и их атрибутов, а также связей между сущностями. Текстовое описание должно быть представлено в виде таблицы 19.

Описание остальных сущностей выполняется аналогично и размещается далее в таблице после характеристики первой сущности.

Таблица 19 – Описание сущностей БД

Наименование поля	Назначение атрибута	Тип данных	Примечание
lease_contract (договор аренды)			
lease_contract	Номер договора аренды	Целое число типа INT, принимает значения из диапазона [10000; 99999]	Первичный ключ
inventory_number	Инвентарный номер помещения	Целое число типа INT, принимает значения из диапазона [1001000; 9009999]	Внешний ключ, связывает с таблицей premise, кардинальность связи – M:0
tenant_id	Номер арендатора	Целое беззнаковое число типа INT	Внешний ключ, связывает с таблицей tenant, кардинальность связи – M:1
lease_start_date	Хранит дату начала аренды помещения	Значение, соответствующие маске ДД-ММ-ГГГГ, начальное значение не должно быть ранее даты 17.04.2012	–
lease_term	Хранит количество суток, на которое арендуется помещение	Целое число типа SMALLINT, минимальное значение срока аренды составляет 10 сут, а максимальное – 180 сут	–

Контрольные вопросы к лабораторной работе № 7

1 Что такое модель данных?

2 Какие виды моделей данных выделяют?

3 Какие аспекты рассматриваются при проектировании информационной модели предметной области?

4 Что такое архитектура ANSI/SPARC и какие уровни абстракции выделяют согласно ANSI/SPARC?

5 Какие этапы проектирования БД существуют?

6 Что такое нормализация и о чем гласят определения 1НФ, 2НФ, 3НФ?

7 В чем суть концептуального проектирования?

8 В чем суть логического проектирования?

9 В чем суть физического проектирования?

10 Что такое транзитивная зависимость?

11 Какие виды ключей выделяют?

12 Как привести отношения к 3НФ?

ЛАБОРАТОРНАЯ РАБОТА № 8
Разработка проектных решений на основе моделирования свойств и линий поведения программных объектов

Цели выполнения лабораторной работы:

- разработать алгоритмические и программные реализации бизнес-логики программного средства;
- описать статические и динамические аспекты поведения сложных объектов программной системы.

Задание	**Этапы выполнения задания**
1 Разработать алгоритмы, реализующие бизнес-логику программного средства, построить их схемы и привести фрагменты листинга кода	1 Представить текстовое описание алгоритмов, реализующих бизнес-логику. 2 Построить схемы разработанных алгоритмов (схемы программы или схему работы системы) в соответствии с требованиями ЕСПД и ЕСКД. 3 Представить фрагменты листинга кода программной реализации алгоритмов
2 Выполнить схему алгоритма в виде чертежа	Построить чертеж схемы программы либо схемы работы системы и оформить согласно требованиям ЕСПД и ЕСКД
3 Разработать и описать статические модели поведения сложных объектов программной системы	1 Посредством диаграммы классов показать используемые в архитектуре программного средства паттерны проектирования. 2 При помощи диаграмм компонентов и размещения показать дизайн системы и расположение компонентов по узлам системы (если программное средство обладает множеством классов, узлов или других компонентов, тогда дополнительно строится диаграмма пакетов)
4 Разработать и описать динамические модели поведения сложных объектов программной системы *для конкретного варианта использования*	1 Средствами statechart-диаграммы смоделировать состояния объектов и условия переходов между ними. Представить графическое и текстовое описание полученной модели поведения объектов. 2 С помощью sequence-диаграммы получить отражение во времени процесса обмена сообщениями между объектами. Представить графическое и текстовое описание полученной модели взаимодействия объектов. 3 С помощью activity-диаграммы смоделировать поведение системы в рамках конкретного варианта использования

Задание	**Этапы выполнения задания**
5 Сформировать отчет по лабораторной работе	*Содержание отчета:* Титульный лист (приложение Б) 1 Описание схем алгоритмов 2 Чертежи схем алгоритмов 3 Результаты моделирования структуры объектов программной системы 4 Результаты моделирования поведения объектов программной системы 5 Результаты моделирования взаимодействия объектов программной системы 6 Выводы

Краткие теоретические сведения и методические указания к заданию 1

С точки зрения разработки программной системы **бизнес-логику** можно определить как совокупность правил, принципов, зависимостей поведения объектов предметной области.

В настоящее время существует множество различных подходов к описанию и реализации бизнес-логики. Наиболее распространены следующие три типовых решения, которые детально рассмотрены в книге Матрина Фаулера «Шаблоны корпоративных приложений»:

- *сценарий транзакции (функциональный подход).* В этом случае проектируют функциональную структуру (иерархию функций) программной системы. Каждая функция системы реализует определенную операцию прикладной логики;
- *модель предметной области (объектный подход).* Разрабатывается объектная модель предметной области. В этом случае бизнес-логику можно описывать посредством моделирования отношений и распределения ответственности между объектами;
- *модуль таблицы (смешанный подход)*. Использует сочетание сценария транзакции и модели предметной области.

Неплохой пример упрощенной схемы выбора типового решения для реализации бизнес-логики представлен на рисунке 39.

На стадии анализа и моделирования программной системы бизнес-логика может описываться в виде:

- текста;
- концептуальных аналитических моделей предметной области;
- разнообразных алгоритмов;

- диаграмм UML – диаграмм деятельности, последовательности, состояний;
- моделей бизнес-процессов.

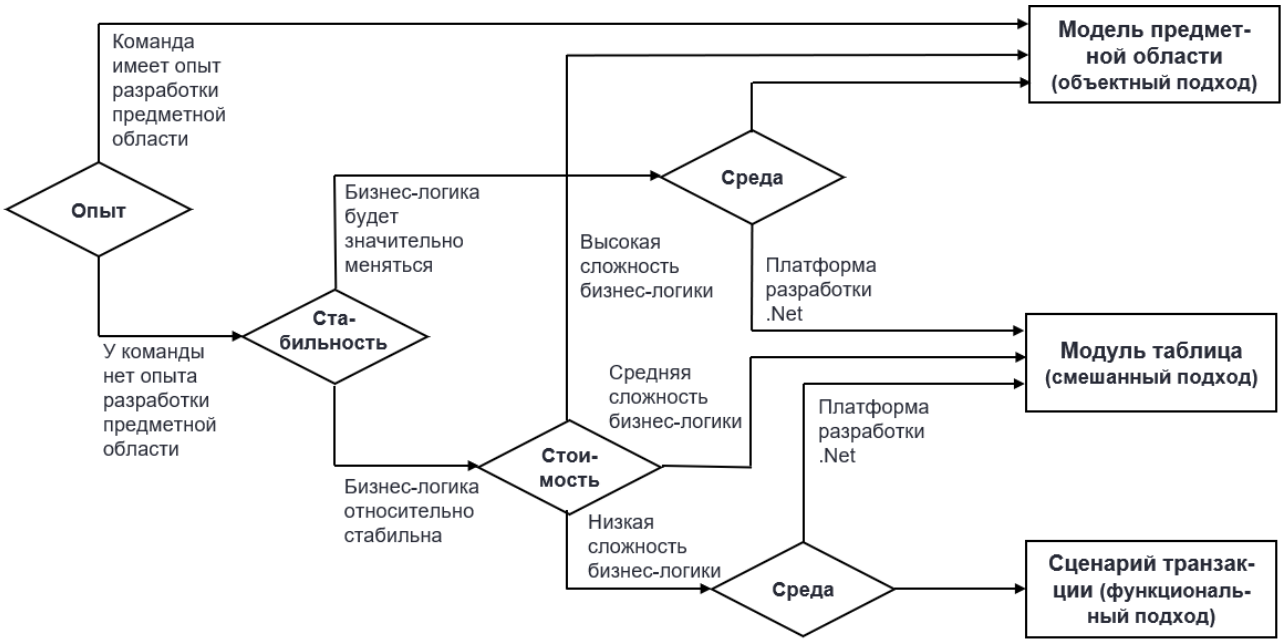


Рисунок 39 – Упрощенная схема выбора типового решения для реализации бизнес-логики[1]

На стадии проектирования программной системы бизнес-логика реализуется в классах и методах классов или процедурах и функциях.

Для детализации алгоритмов бизнес-логики будем использовать текстовое описание и схемы. Схема является универсальной, не зависящей от чего-либо, и отображает принцип работы определенного алгоритма, что дает полное понимание происходящего процесса. Она описывает действие алгоритма так, как его можно реализовать на любом языке программирования.

Следует отметить, что схемы могут использоваться на различных уровнях детализации алгоритмов бизнес-логики.

Согласно ЕСПД **схема** – графическое представление определения, анализа или метода решения задач, в котором используются символы для отображения операций, данных, потока, оборудования и т. д.

---

[1] URL: https://software-testing.ru/library/testing/test-management/ 62-business-logic.

В ГОСТ 19.701–90 [15] выделены следующие типы схем:

- схема данных;
- схема программ;
- схема работы системы;
- схема взаимодействия программ;
- схема ресурсов системы.

Символы, использование которых допускается в такого рода схемах, представлены в таблице «Применение символов» ГОСТ 19.701–90 [15].

В рамках лабораторной работы описание разработанных алгоритмов следует представить в виде текста, а также в виде схем программ (отображают последовательность операций).

Программную реализацию разработанных алгоритмов необходимо представить в виде небольшого фрагмента листинга кода.

ПРИМЕР ОПИСАНИЯ АЛГОРИТМА, РЕАЛИЗУЮЩЕГО БИЗНЕС-ЛОГИКУ ПРОГРАММНОГО СРЕДСТВА

Алгоритм формирования заявки на обучение.

Шаг 1. Пользователь авторизуется в системе. Вводит логин и пароль.

Шаг 2. Осуществляется проверка логина и пароля. При успешной авторизации пользователю отображается список всех доступных курсов. В противном случае выдается сообщение об ошибке, и работа завершается.

Шаг 3. При отображении списка доступных курсов пользователь выбирает интересующий курс и добавляет его в заявку.

Шаг 4. Система проверяет, является ли выбранный пользователем курс новым. Если новый – система формирует заявку на обучение и выводит сообщение о назначении данного курса пользователю. В противном случае пользователю отобразится сообщение об ошибке.

Схема алгоритма формирования заявки на обучение представлена на рисунке 40.

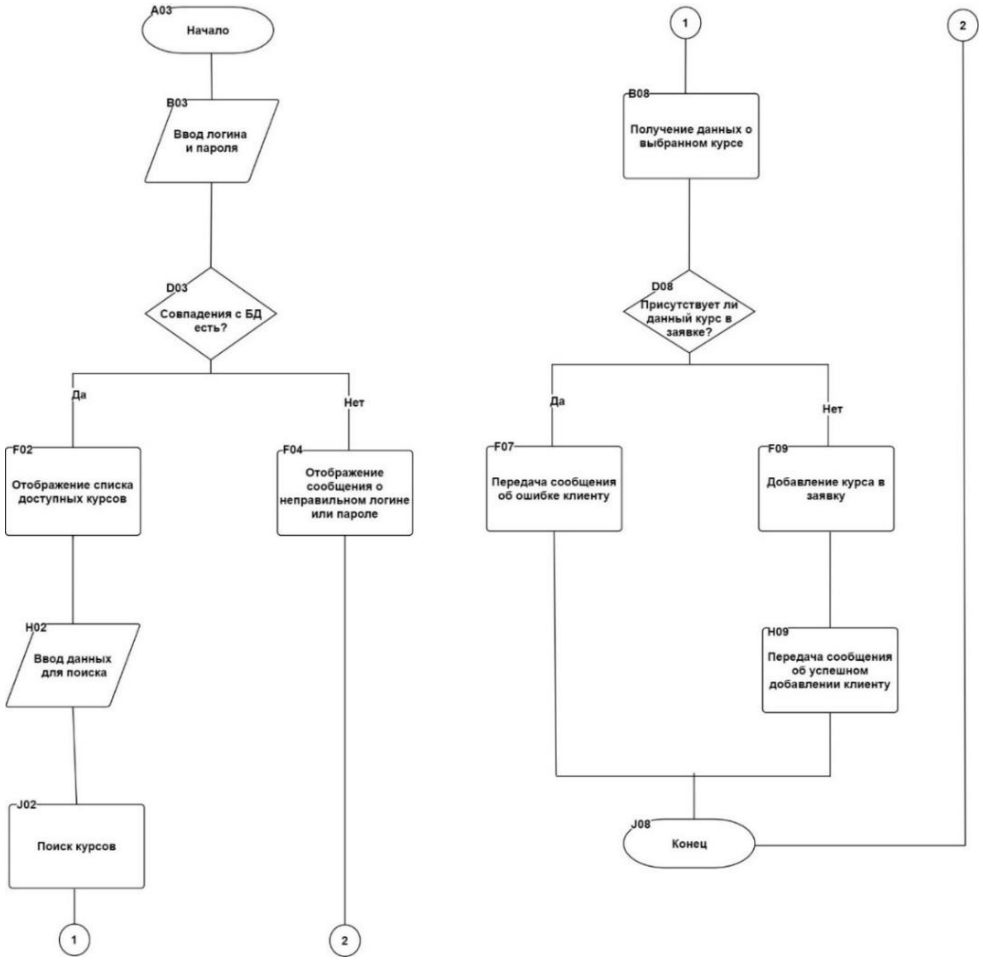


Рисунок 40 – Схема алгоритма формирования заявки на обучение

Для реализации алгоритма формирования заявки на обучение был создан отдельный метод класса OrderTable. Пользователь через форму обучающегося выбирает интересующий его курс. По нажатии кнопки «Добавить» данные о курсе передаются на сервер. При получении от пользователя сообщения, начинающегося с «addToOrder», вызывается метод addToOrder класса OrderTable, код которого представлен ниже.

```
public String addToOrder(String student_id, String course_name, String
course_format, String start_date) {
    boolean isOrderHasEquals = false;
    boolean isUpdatesDone = false;
    try {
        String course_id = courseTable.getCourseParam(course_name, course_for-
mat, start_date, "id");
        ResultSet rs = getUnacceptedOrder(student_id);
        if (!rs.next()) {
            int done1 = stmt.executeUpdate(
                    "INSERT INTO ORDERS(ORD_ID, ST_ID, STATUS) VALUES(OR-
DER_SEQ.nextval, " + student_id + ", 0)");
            int done2 = stmt.executeUpdate(
                    "INSERT INTO STUDY_ORDER(ord_id,st_proc_id) VALUES(OR-
DER_SEQ.currval, " + quotate(course_id) + ")");
            if (done1 > 0 && done2 > 0) {
                isUpdatesDone = true;
            }
```

```
        } else {
            String ord_id = rs.getString("ord_id");
            ResultSet rs1 = getCourseIdFromOrder(ord_id);
            while (rs1.next()) {
                if (rs1.getString("st_proc_id").equals(course_id)) {
                    isOrderHasEquals = true;
                    break;
                }
            }
            if (isOrderHasEquals) {
                return "unique error";
            } else {
                int done1 = stmt.executeUpdate(
                        "INSERT INTO STUDY_ORDER(ord_id, st_proc_id) VALUES(" +
ord_id + "," + course_id + ")");
                if (done1 > 0) {
                    isUpdatesDone = true;
                }
            }
        }
        int done3 = stmt
                .executeUpdate("UPDATE STUDY_PROCESS SET GROUP_SIZE = GROUP_SIZE
- 1 WHERE st_proc_id = " + course_id);
        if (isUpdatesDone && done3 > 0) {
            return "success";
        } else {
            return "fail";
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
        return "fail";
    }
}
```

На сервере производятся все необходимые обработки исключительных ситуаций, которые могут возникнуть при добавлении курса в заявку с помощью оператора try-catch. С помощью цикла if делаются различные логические проверки.

Для взаимодействия с базой данных используется метод executeUpdate, который выполняет такие команды, как INSERT, UPDATE, DELETE, CREATE TABLE, DROP TABLE. В качестве результата возвращает количество строк, затронутых операцией (например, количество добавленных, измененных или удаленных строк), или 0, если ни одна строка не затронута операцией или если команда не изменяет содержимое таблицы.

Алгоритм обработки заявки администратором. *Аналогично описать указанный алгоритм.*

Краткие теоретические сведения и методические указания к заданию 2

Чертежи разрабатываются с целью декомпозиции и пояснения сложных задач проектирования. Наименования и обозначения черчений должны соответствовать требованиям, установленным ГОСТ 2.102–68 [16].

Пример схемы алгоритма обработки данных, оформленной как чертеж, представлен на рисунке 41.

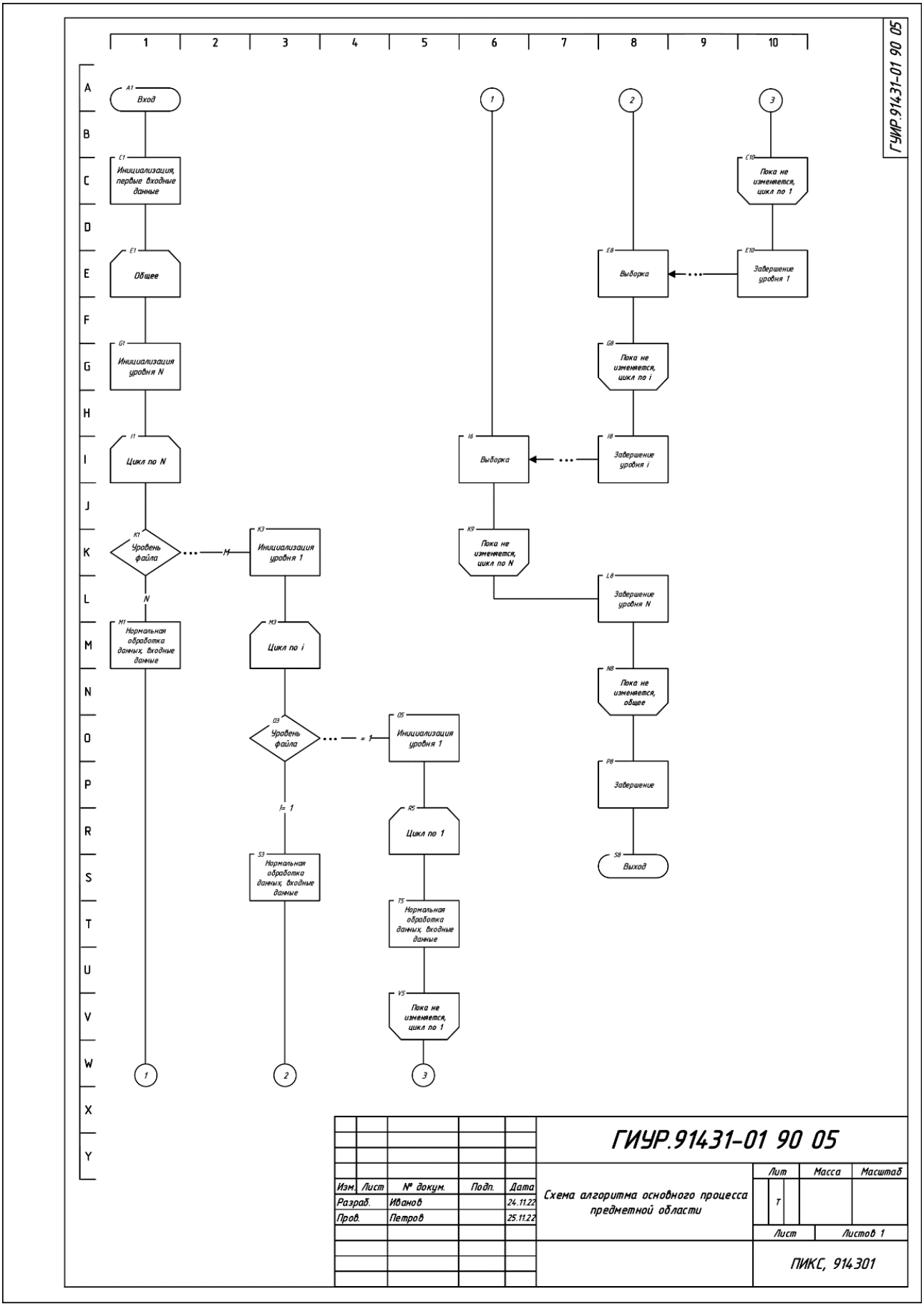


Рисунок 41 – Пример схемы алгоритма обработки данных

При *построении чертежа* необходимо обратить внимание на следующие требования:

1 Каждый элемент схемы должен иметь ширину и длину, равные ширине и длине ячейки сетки (подробнее о том, какими по размеру могут быть ячейки сетки, можно узнать из ГОСТ 19.002–80 [17]).

2 Каждому элементу должны быть присвоены координаты согласно зоне, которую занимает данный элемент (см. пункт 1.4 ГОСТ 19.002–80 [17]), при этом буква и цифра координаты вписывается в левом верхнем углу с прерыванием контура элемента и отступом 5 мм от левого края.

3 Изломы линий выполняются под углом 90°.

Оформление основной надписи и прочих элементов осуществлять в соответствии с требованиями задания 1 лабораторной работы № 3 [1].

Краткие теоретические сведения и методические указания к заданию 3

Все диаграммы, используемые в UML 2.x, могут быть разбиты на две группы (рисунок 42):

- диаграммы для моделирования структуры программной системы – *структурные диаграммы*, показывающие **статическую структуру** системы и ее частей на разных уровнях абстракции и реализации;
- диаграммы для моделирования поведения программной системы – *диаграммы поведения*, показывающие **динамическое поведение** объектов в системе, которое можно описать как серию изменений в системе с течением времени.

В рамках лабораторной работы с позиции структурной организации системы для проектирования архитектуры *будут использоваться UML-диаграммы классов, компонентов, развертывания и пакетов*.

Каждому классу структурных диаграмм соответствует свой уровень абстракции в модели программной системы:

- логический уровень – диаграммы классов;
- уровень реализации – диаграммы компонентов;
- уровень оборудования – диаграммы развертывания.

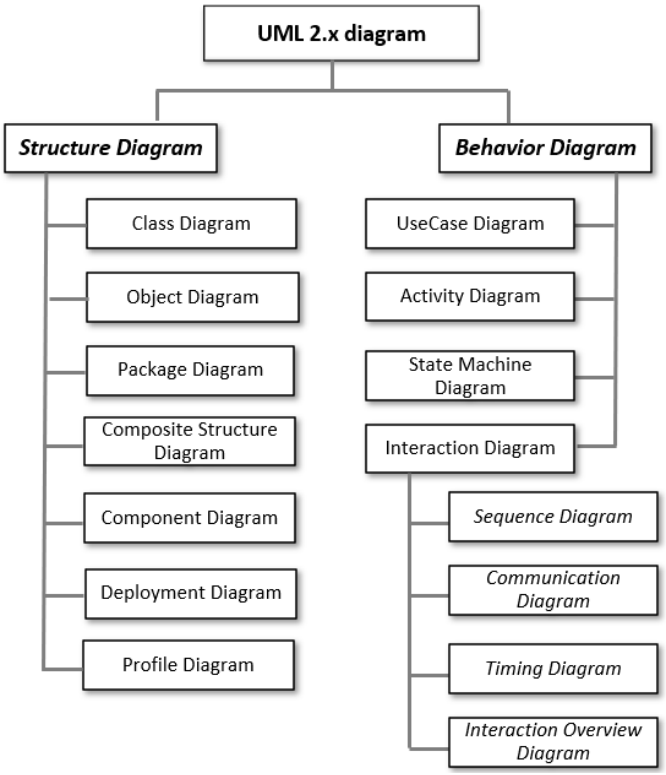


Рисунок 42 – Структурные диаграммы и диаграммы поведения (UML 2.x)

Диаграмма классов. Диаграмма классов – это наиболее часто используемый тип диаграмм, которые создаются при моделировании объектно-ориентированных систем.

Диаграмма классов – это диаграмма, которая показывает набор классов, интерфейсов, кооперaций и их связи (рисунок 43).
Графически представляет собой набор вершин и связывающих их дуг.

На практике диаграммы классов применяют для моделирования статического представления системы (по большей части это моделирование словаря системы, кооперaций или схем). Кроме того, диаграммы данного типа служат основой для целой группы взаимосвязанных диаграмм – диаграмм компонентов и диаграмм размещения.

Диаграммы классов важны не только для визуализации, специфицирования и документирования структурных моделей, но также для конструирования исполняемых систем посредством прямого и обратного проектирования.

Подробнее с примерами диаграмм классов можно ознакомиться в источниках [18; 19].

В отчете по лабораторной работе необходимо представить диаграмму классов, отражающую используемый паттерн проектирования, и краткое описание задачи, которую решает данный паттерн.

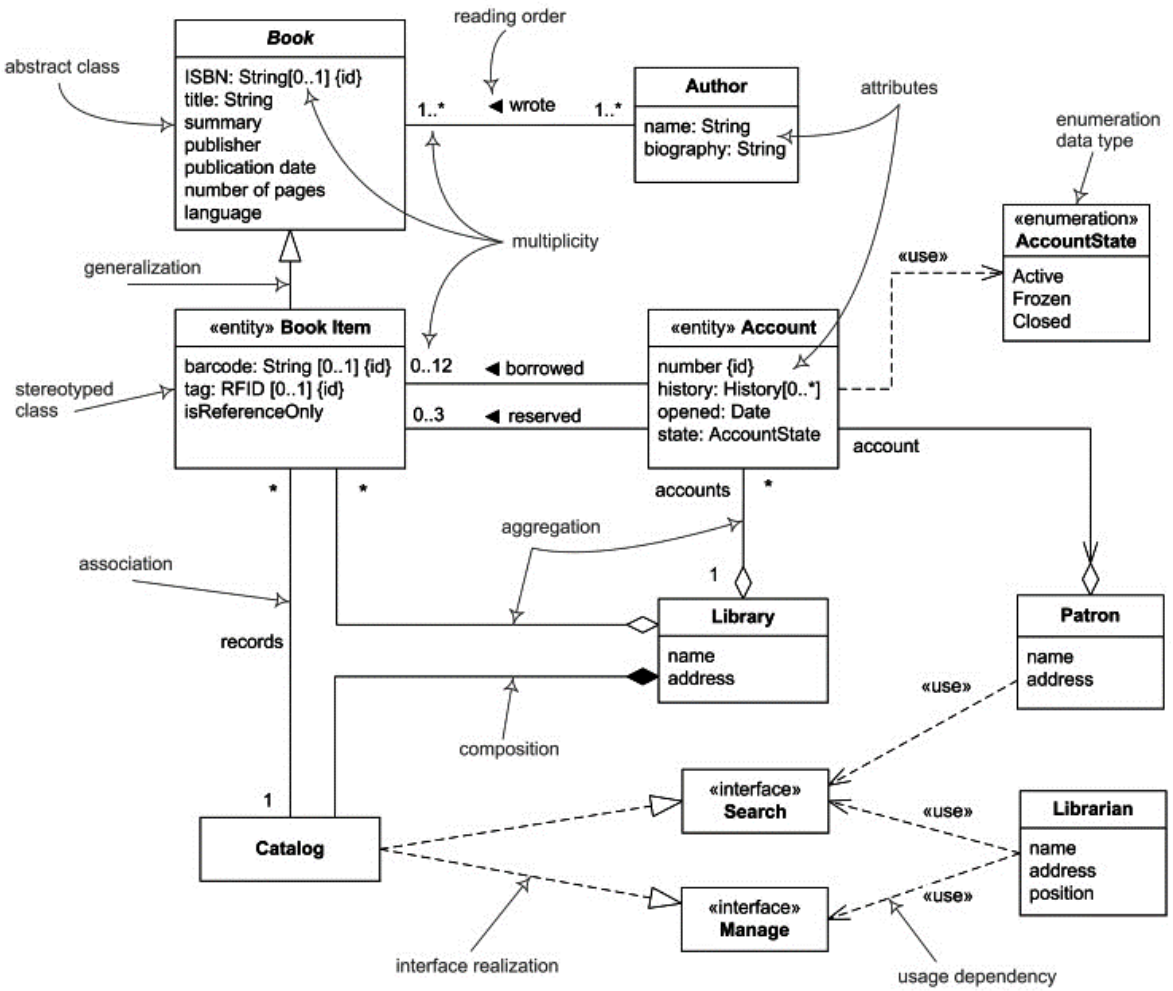


Рисунок 43 – Пример диаграммы классов[1]

[1] URL: https://www.uml-diagrams.org/class-diagrams/class-diagram-domain-overview.png.

Диаграмма компонентов. С точки зрения реализации проектируемая система состоит из компонентов, представленных на диаграммах компонентов, распределенных по вычислительным узлам, представленным на диаграммах размещения.

Диаграмма компонентов (component diagram) демонстрирует инкапсулированные классы и их интерфейсы, порты и внутренние структуры, состоящие из вложенных компонентов и коннекторов рисунок 44).

Диаграммы компонентов описывают статическое представление дизайна системы. Они важны при построении больших систем из мелких частей (UML отличает диаграмму составной структуры (composite structure diagram), применимую к любому классу, от компонентной диаграммы).

Одним из основных элементов данной диаграммы является компонент. Компоненты позволяют инкапсулировать части системы, чтобы уменьшить количество зависимостей, сделать их явными, а также повысить взаимозаменяемость и гибкость на случай, если система должна будет изменяться в будущем.

Хороший компонент наделен следующими характеристиками:

- инкапсулирует сервис с хорошо очерченным интерфейсом и границами;
- имеет внутреннюю структуру, которая допускает возможность ее описания;
- не комбинирует несвязанную функциональность в пределах одной единицы;
- организует внешнее поведение, используя интерфейсы и порты в небольшом количестве;
- взаимодействует только через объявленные порты.

Примеры компонентов: класс или набор классов, реализующих бизнес-логику, веб-сервис (REST API), модуль аутентификации, БД.

Чтобы проще было выделить компонент, можно воспользоваться принципами **высокой связности (high cohesion)** внутри компонента и **слабой связанности (low coupling)** между компонентами.

Так как диаграмма компонентов является высокоуровневой диаграммой, ее не стоит перегружать такими деталями реализации, как внутренние классы и методы.

Также стоит учитывать то, каким образом компоненты будут физически развернуты на сервере или в облаке.

Примеры ситуаций, где целесообразно строить диаграмму компонентов:

- проектирование микросервисной архитектуры (например, показать взаимодействие между сервисами);
- документация сложных систем, чтобы объяснить физическое разделение модулей;
- обсуждение интеграции новых компонентов в разрабатываемую систему.

Более подробное описание других элементов диаграммы компонентов, а также примеры построения представлены в источнике [20].

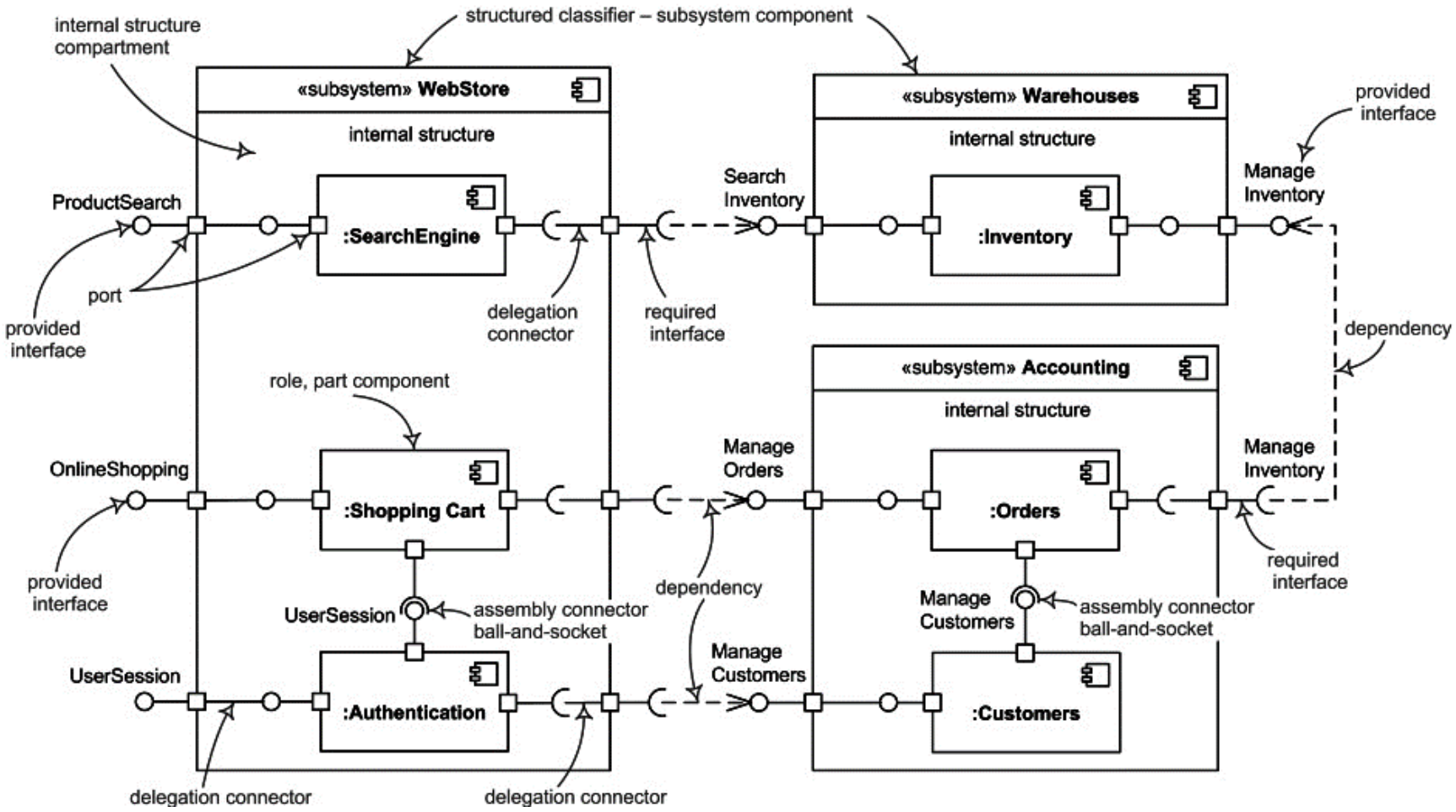


Рисунок 44 – Пример диаграммы компонентов[1]

[1] URL: https://www.uml-diagrams.org/component-diagrams/component-diagram-overview.png.

Диаграмма размещения. Это один из двух видов диаграмм, используемых при моделировании физических аспектов объектно-ориентированной системы. Такая диаграмма представляет конфигурацию узлов, где производится обработка информации, и показывает, какие артефакты размещены на каждом узле.

Диаграмма размещения (deployment diagram) показывает конфигурацию узлов-процессоров, а также размещаемые на них компоненты (рисунок 45). Диаграммы размещения дают статическое представление размещения архитектуры. Узлы, как правило, содержат один или несколько артефактов.

Диаграммы размещения используются для моделирования статического представления системы с точки зрения размещения. В основном под этим понимается моделирование топологии аппаратных средств, на которых работает система. По существу, диаграммы размещения – это просто диаграммы классов, сосредоточенные на системных узлах.

Диаграммы размещения важны не только для визуализации, специфицирования и документирования встроенных, клиент-серверных и распределенных систем, но и для управления исполняемыми системами с использованием прямого и обратного проектирования.

Диаграмму размещения можно проигнорировать только в том случае, если разрабатываемое программное средство будет исполняться *исключительно* на одной машине, обращаясь при этом лишь к стандартным устройствам на этой же машине, управление которыми полностью возложено на ОС.

Подробнее с диаграммами размещения можно ознакомиться в источнике [20].

Уровень детализации диаграммы развертывания зависит от ее цели. Например:

- для DevOps-специалистов можно указать настройки сети и типы оборудования;
- для разработчиков достаточно будет представить узлы и артефакты.

Если при разработке программного средства используются облачные платформы (например, AWS, Azure), узлы могут представлять виртуальные машины, контейнеры или серверы без привязки к конкретному аппаратному обеспечению.

При построении диаграммы размещения стоит обратить внимание на следующие моменты: разделение программного средства на слои, масштабируемость, связь по сети.

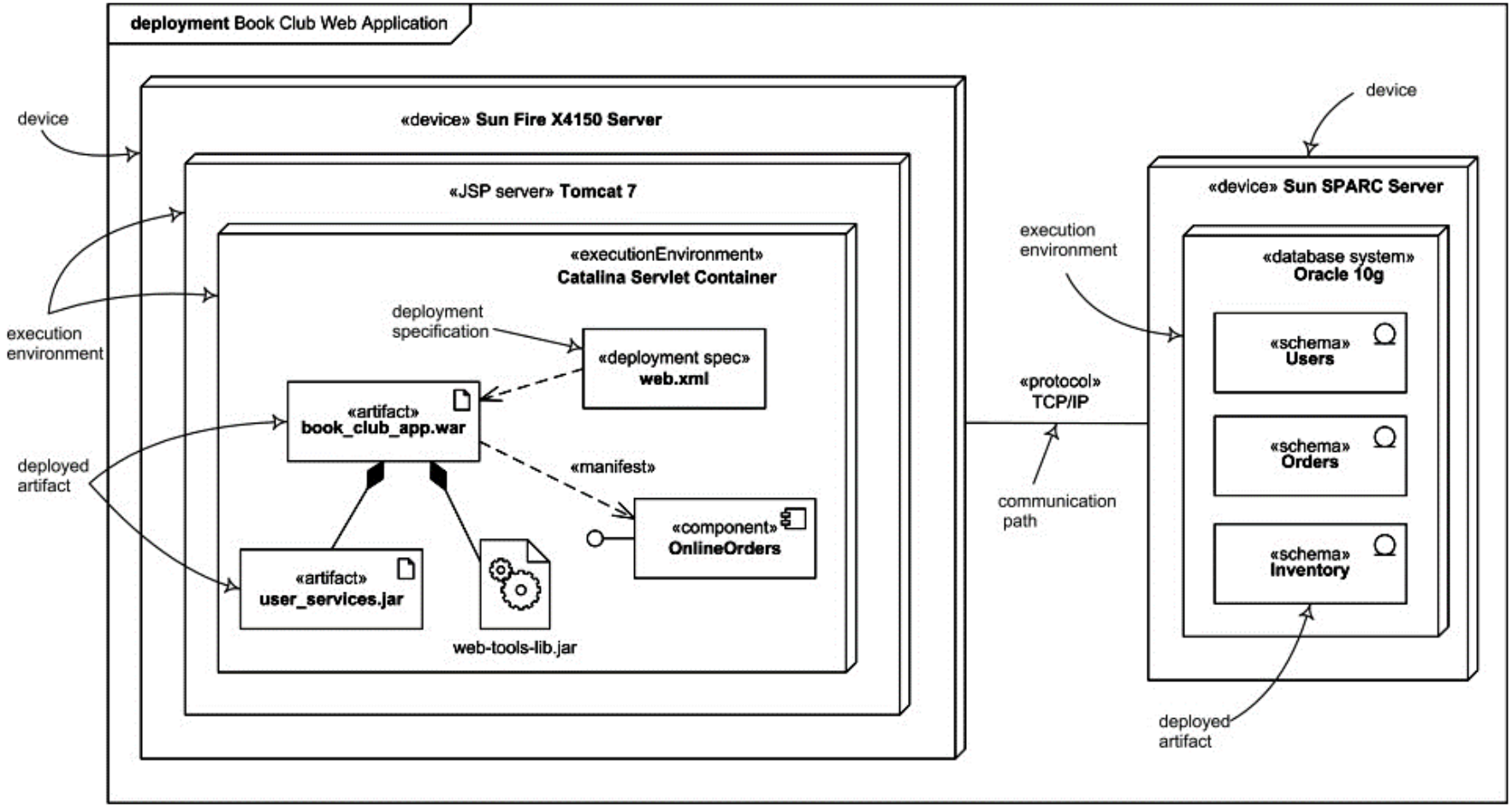


Рисунок 45 – Пример диаграммы размещения[1]

[1] URL: https://www.uml-diagrams.org/deployment-diagrams/deployment-diagram-overview-specification.png.

Диаграмма пакетов. В случае если разрабатываемое программное средство будет иметь большое количество классов, интерфейсов, узлов, компонентов и других элементов, целесообразно построение диаграммы пакетов (рисунок 46).

Диаграмма пакетов (package diagram) показывает декомпозицию самой модели на организационные единицы и их зависимости.

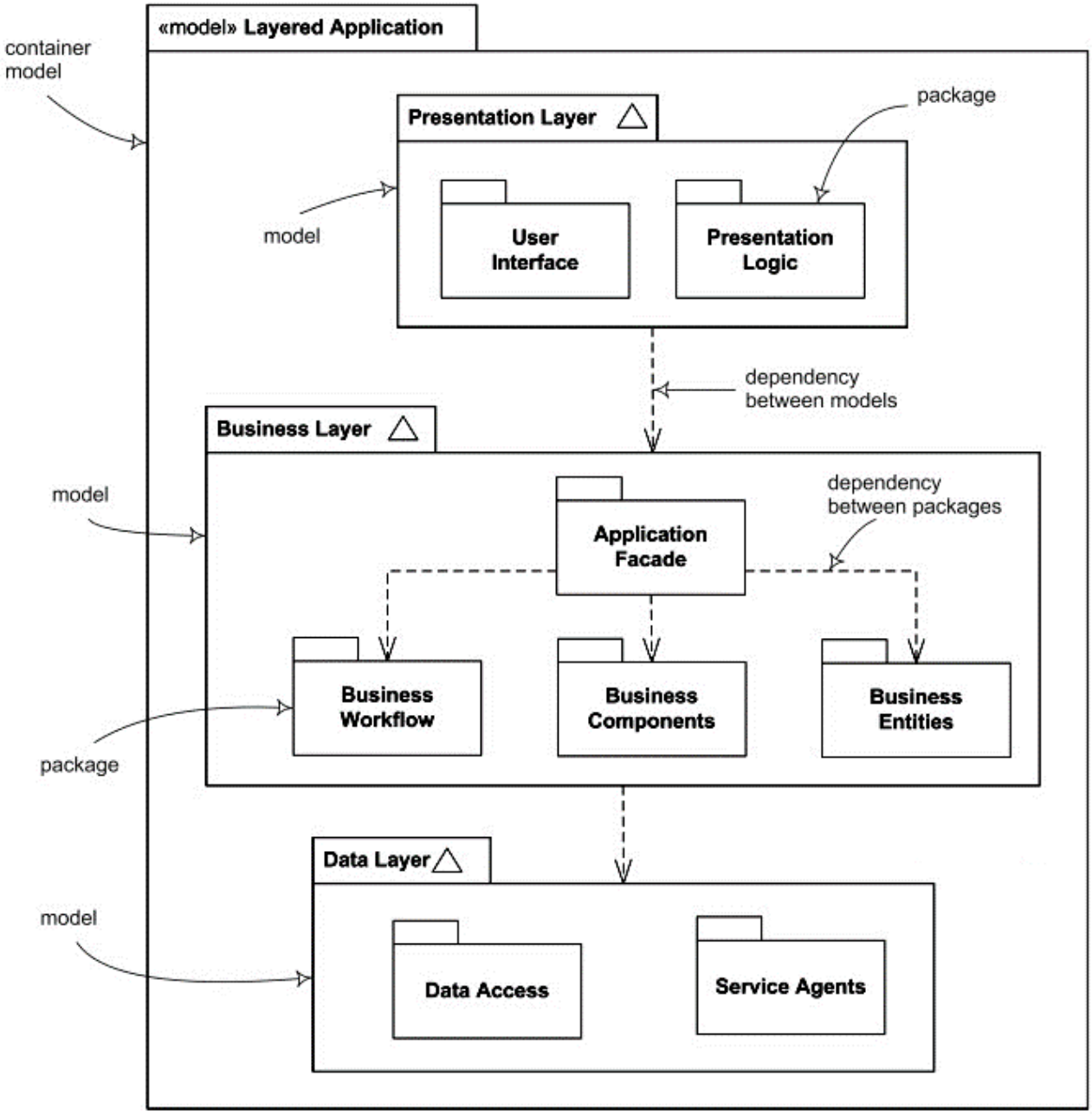


Рисунок 46 – Пример диаграммы пакетов[1]

[1] URL: https://www.uml-diagrams.org/package-diagrams/model-diagram-elements.png.

Основным элементом данной диаграммы является пакет – способ организации элементов модели в блоки, которыми можно распоряжаться как единым целым. Хорошо структурированный пакет объединяет семантически близкие элементы, которые имеют тенденцию изменяться совместно.

Такие пакеты характеризуются слабой связанностью и высокой согласованностью, причем доступ к содержимому пакета строго контролируется. Объединение частей модели в блоки упрощает процесс масштабирования системы в будущем.

Более детальное представление диаграмм компонентов приведено в источнике [20].

Краткие теоретические сведения и методические указания к заданию 4

Представление системы с определенной точки зрения (view point) называется **видом** (**view**). Вид также называют **моделью** системы с определенной точки зрения.

Каждая модель определяет конкретный аспект системы, использует набор диаграмм и документов заданного формата, а также отражает точку зрения и является объектом деятельности различных пользователей с конкретными интересами, ролями или задачами. Модель абстрагирует свойства, поведение и структуру системы.

Программная система может рассматриваться со следующих точек зрения:

- функциональные требования к системе (use case view);
- логическая организация и поведение системы (logical design view);
- функционирование или работа процессов системы (process view);
- компонентная организация и поведение системы (implementation view или component view);
- требования к размещению компонент на аппаратных ресурсах (deployment view).

В зависимости от точки зрения получают различные виды, или модели, представления программной системы:

- Use case view (представление вариантов использования или прецедентов) предназначено для моделирования функциональных требований к системе;
- Logical view (логическое представление) предназначено для моделирования логической структуры и поведения системы;
- Component view (представление компонент) предназначено для моделирования архитектуры системы на уровне компонент;
- Deployment view (представление размещения или развертывания) предназначено для моделирования развертывания компонент системы на аппаратуре.

Иногда Component view и Deployment view объединяют в один вид, который называют Implementation view.

В рамках лабораторной работы ограничимся рассмотрением вида представления Logical view в целях моделирования поведения программной системы – модели поведения и взаимодействия объектов. Модели представления Use case view, Logical view, Component view и Deployment view были рассмотрены ранее.

Для моделирования поведения объектов программной системы широко используются следующие диаграммы:

- диаграмма деятельности (activity diagram), в которой моделируются последовательности действий, выполняемых объектами программной системы;
- диаграмма состояний (statechart diagram, или state machine diagram), в которой моделируются состояния объектов программной системы, а также переходы между этими состояниями.

Для моделирования взаимодействия объектов программной системы используются диаграммы взаимодействия, включающие чаще всего два вида диаграмм:

- диаграмму последовательности (sequence diagram), которая моделирует упорядоченное во времени взаимодействие объектов системы;
- диаграмму коммуникации (communication diagram), которая моделирует общую схему взаимодействия объектов и их роли во взаимодействии.

Диаграммы деятельности. Они используются для моделирования поведения системы в рамках различных вариантов использования или потоков управления, а также для моделирования бизнес-процессов.

Подробнее с примерами диаграмм деятельности можно ознакомиться по ссылке: https://www.uml-diagrams.org/activity-diagrams-examples.html.

Однако применительно к бизнес-процессам желательно выполнение каждого действия ассоциировать с конкретным подразделением компании, т. е. использовать дорожки, позволяющие указать зоны ответственности в рамках моделируемого бизнес-процесса. В качестве имен дорожек используются либо названия подразделений (департаментов) рассматриваемой компании, либо названия отдельных должностей сотрудников тех или иных подразделений.

Диаграммы деятельности также удобны и для описания поведения, включающего большое количество параллельных процессов. Не менее важная область их применения связана с моделированием бизнес-процессов при параллельном программировании, поскольку можно графически изобразить все ветви и определить, когда их необходимо синхронизировать.

Принять во внимание при разработке диаграммы деятельности:

- деятельность (действие) на диаграмме изображается **прямоугольником со скругленными боковыми сторонами (не углами!)**;
- с точки зрения топологии стрелок узлы **decision (ромб)** и **fork (утолщенная линия)** схожи (один входной поток и несколько выходных). Разница между

ними в параллельности: на выходе **decision-узла** появляется только один поток (из нескольких возможных), а на выходе **fork-узла** – несколько параллельных потоков;

- хорошо структурированная диаграмма деятельности сконцентрирована на описании одного аспекта (варианта использования) динамики системы. Должна содержать только те элементы, которые существенны для понимания этого аспекта;
- сложные виды деятельности можно дополнительно детализировать, разбив на действия и изобразив «диаграмму в диаграмме».

Диаграмма состояний. Представлена на рисунке 47 и предназначена для моделирования динамического поведения объектов модели с использованием их состояний и переходов между этими состояниями.

Подробнее с примерами диаграмм деятельности можно ознакомиться по ссылке: https://www.uml-diagrams.org/state-machine-diagrams-examples.html.

Диаграмма состояний определяет все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате наступления некоторых событий. Диаграммы состояний чаще всего используются для описания поведения отдельных объектов, а также спецификации функциональности других компонентов моделей, таких как варианты использования, актеры, подсистемы, системы.

Принять во внимание при разработке диаграммы состояний:

- состояние на диаграмме изображается прямоугольником **со скругленными вершинами (не боковыми сторонами!)**;
- на диаграмме состояний **decision-узлы не используются**;
- диаграммы состояний **не следует создавать для каждого класса**. Если объект класса может существовать в нескольких состояниях и в каждом из них ведет себя по-разному, то для него может потребоваться диаграмма состояний. Диаграммы состояний создаются для описания объектов с высоким уровнем динамического поведения;
- из каждого состояния на диаграмме не может быть самопроизвольного перехода в какое бы то ни было другое состояние. Все переходы должны быть явно специфицированы, в противном случае построенная диаграмма состояний является либо неполной (неадекватной), либо ошибочной с точки зрения нотации языка UML.

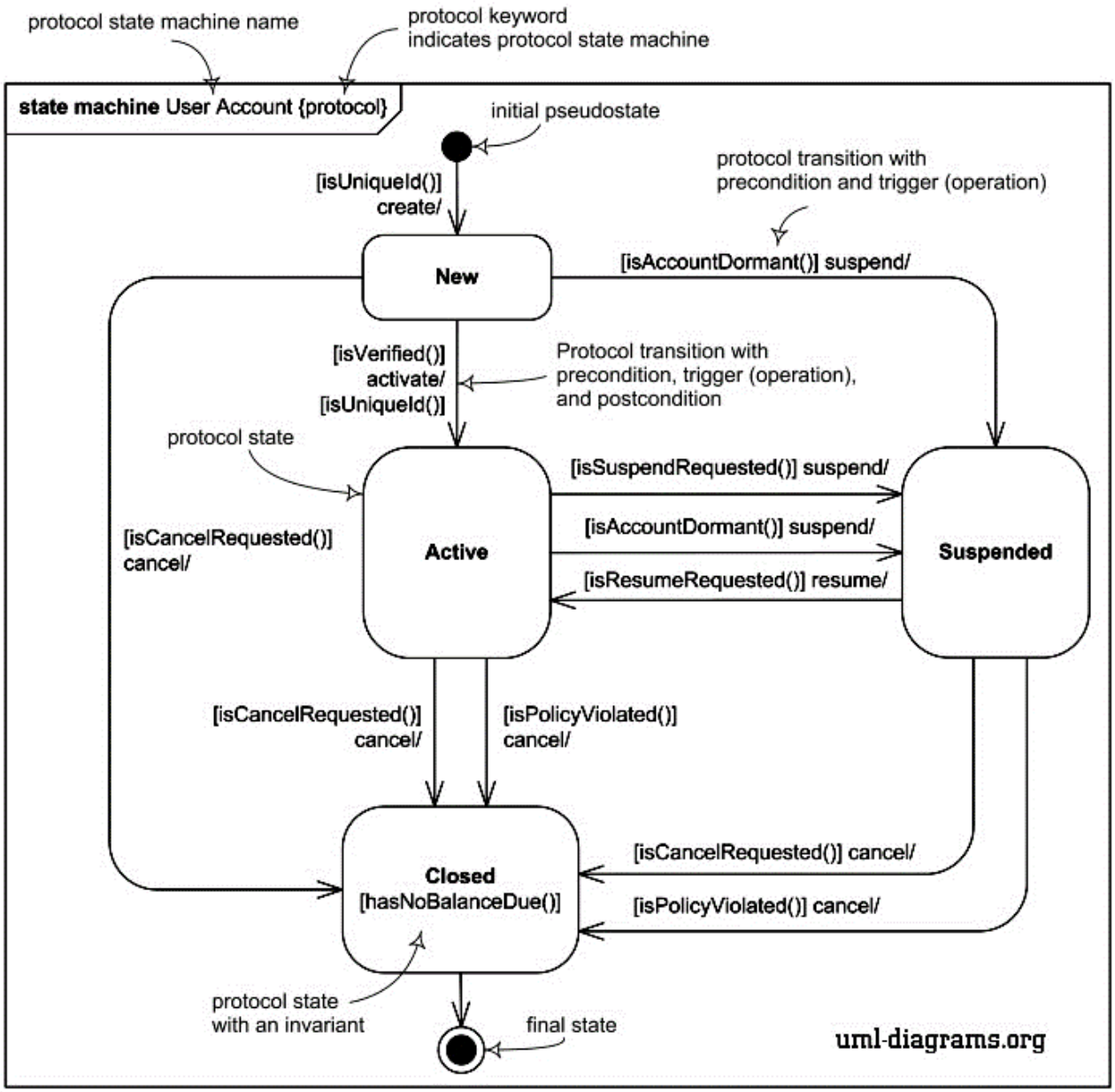


Рисунок 47 – Пример диаграммы состояний[1]

При построении диаграммы необходимо соблюсти требование отсутствия конфликтов у всех переходов, выходящих из одного и того же состояния. Наличие такого конфликта может служить признаком ошибки либо параллельности или ветвления рассматриваемого процесса. Если параллельность по замыслу разработчика отсутствует, то следует ввести дополнительные сторожевые условия либо изменить существующие, чтобы исключить конфликт переходов. При наличии параллельности следует заменить конфликтующие переходы одним параллельным переходом типа ветвления.

[1] URL: https://www.uml-diagrams.org/examples/online-shopping-user-account-state-diagram-example.png.

Диаграммы взаимодействия. Они описывают поведение взаимодействующих групп объектов (в рамках варианта использования или некоторой операции класса). Как правило, диаграмма взаимодействия охватывает поведение объектов в рамках только одного потока событий варианта использования. На такой диаграмме отображается ряд объектов и те сообщения, которыми они обмениваются между собой.

Диаграммы последовательности и коммуникации, по сути, отображают одну и ту же информацию, но представляют ее с различных точек зрения. Подобно диаграммам последовательности кооперативные диаграммы отображают поток событий варианта использования. Диаграммы последовательности упорядочены по времени, а кооперативные диаграммы концентрируют внимание на связях между объектами. Большинство case-средств позволяет после построения одной из диаграмм автоматически получить другую, а также выполнять синхронизацию этих диаграмм между собой.

В рамках лабораторной работы для моделирования **взаимодействия объектов** программного средства следует разработать только **диаграмму последовательности**.

Диаграмма последовательности. Она **представлена на рисунке 48** и описывает поведение взаимодействующих объектов **в рамках реализации конкретного варианта использования**.

Подробнее с примерами диаграмм деятельности можно ознакомиться по ссылке: https://www.uml-diagrams.org/sequence-diagrams-examples.html.

Как правило, ее используют для уточнения варианта использования, более детального описания логики сценариев использования.

Для описания сообщений может быть использован шаблон, представленный таблицей 20.

Таблица 20 – Шаблон описания сообщений

Объект-отправитель	Объект-получатель	Имя сообщения

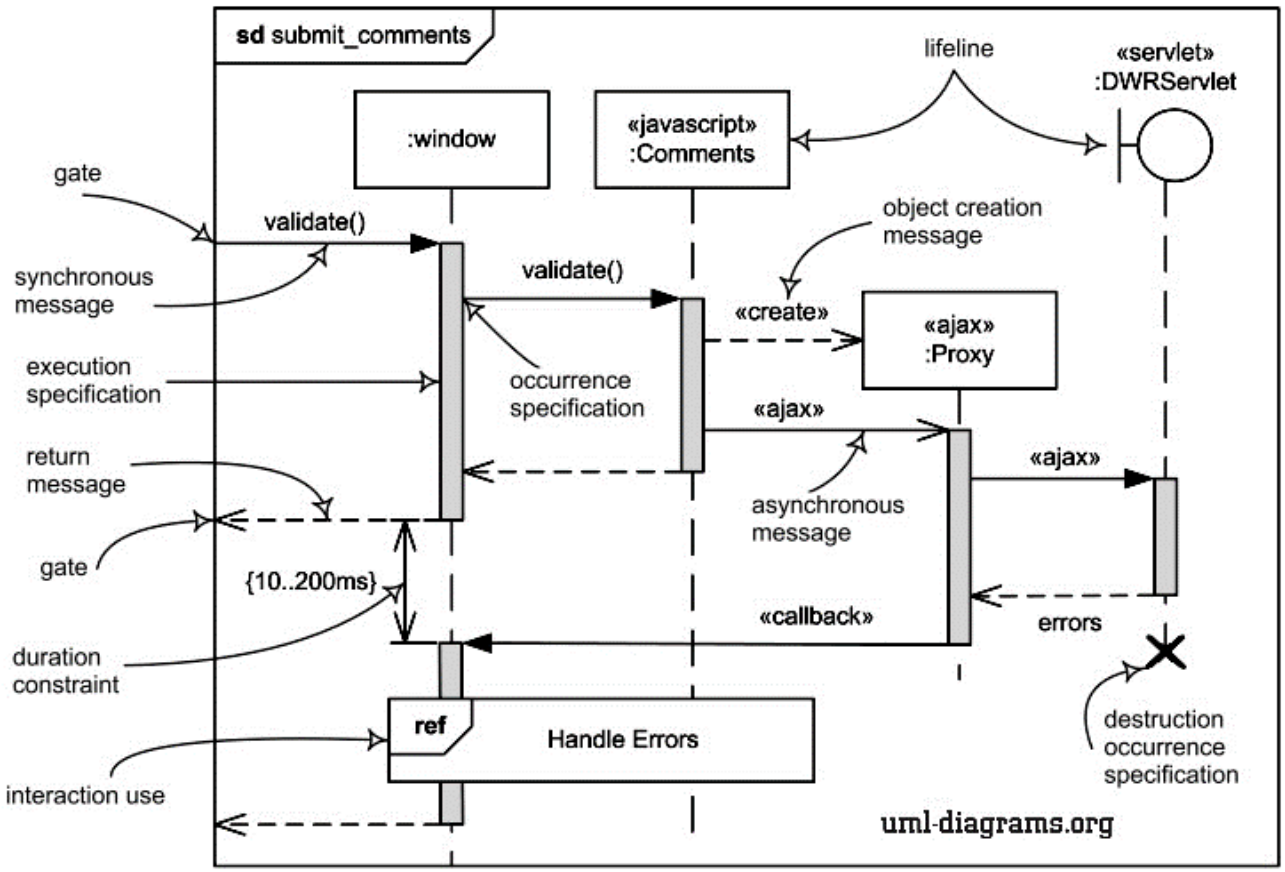


Рисунок 48 – Пример диаграммы последовательности[1]

Принять во внимание при разработке диаграммы последовательности:

- диаграмма последовательности охватывает поведение объектов в рамках **только одного потока событий варианта использования**. Для отображения других вариантов развития событий следует построить несколько диаграмм;
- диаграммы последовательности обязательно опираются на диаграмму классов. Объекты должны принадлежать классам, описанным в диаграмме классов. Если создаваемый объект не может принадлежать ни одному из существующих классов, следует доработать диаграмму классов. Поэтому в диаграммах последовательности нет необходимости создавать новые объекты, их достаточно просто перетащить в рабочую область текущей диаграммы из соответствующей диаграммы классов и определить имя экземпляра данного класса;
- построение диаграммы последовательности целесообразно начинать с выделения из всей совокупности тех и только тех классов, объекты которых участвуют в моделируемом взаимодействии. После этого все объекты наносятся на диаграмму с соблюдением некоторого порядка инициализации сообщений. Когда объекты визуализированы, приступают к спецификации сообщений.

[1] URL: https://www.uml-diagrams.org/sequence-diagrams/sequence-diagram-overview.png.

Более детальное представление рассмотренных диаграмм приведено в источниках [20] и Фаулер, М. UML. Основы / М. Фаулер. – 3-е изд. – СПб. : Символ-Плюс, 2004. – 192 с.

Контрольные вопросы к лабораторной работе № 8

1 В чем суть методологии объектно-ориентированного проектирования программных систем?

2 Какую парадигму проектирования и разработки программного обеспечения поддерживает UML?

3 Как моделируют динамические аспекты программной системы в UML?

4 Как моделируют взаимодействия объектов в UML?

5 В каких случаях целесообразно использование диаграммы состояний и для описания поведения каких компонентов системы используется?

6 Для чего применяется диаграмма последовательности и как ее построить?

7 Что понимают под бизнес-логикой программной системы и каковы ее основные компоненты?

8 Какие существуют типовые подходы к описанию и реализации бизнес-логики?

9 Как описывается бизнес-логика на стадии анализа и моделирования программной системы?

10 Как описывается бизнес-логика на стадии проектирования программной системы?

11 В чем заключаются основные концепции, используемые при разработке бизнес-логики?

12 Какое место занимает бизнес-логика в многоуровневой архитектуре программных систем?

13 Что такое схема алгоритма?

14 Какие выделяют типы схем алгоритма?

15 Каким образом описывается и документируется алгоритм функционирования программной системы?

16 Какие требования устанавливаются к выполнению и оформлению схемы алгоритма в виде чертежа?

ЛАБОРАТОРНАЯ РАБОТА № 9
Реализация программного решения

Цель выполнения лабораторной работы: реализовать функциональность серверной части программного средства в архитектурном стиле REST API с использованием принципов SOLID, DRY, KISS и шаблона DI.

Задание	**Этапы выполнения задания**
1 Разработать серверную часть программного средства	1 Реализовать функциональность серверной части программного средства в архитектурном стиле REST API с использованием SOLID, DRY, KISS, шаблона DI. 2 Составить документацию к разработанному API
2 Сформировать отчет по лабораторной работе	*Содержание отчета:* Титульный лист (приложение Б) 1 Ссылка на публичный репозиторий Github с кодом клиентской части программного средства 2 UML-диаграммы вариантов использования, классов, развертывания 3 Примеры кода, подтверждающие реализацию выбранных принципов и шаблонов программирования 4 Документация к разработанному API 5 Выводы

Краткие теоретические сведения и методические указания к заданию 1

REST – метод взаимодействия компонентов приложения с использованием протокола HTTP для вызова процедуры. При этом необходимые данные передаются в качестве параметров запроса. Этот способ является альтернативой более сложным методам, таким как SOAP, CORBA и RPC.

Веб-сервисы REST (рисунок 49) являются веб-сервисами, реализуемыми с использованием протокола HTTP и принципов REST. Как правило, веб-сервис RESTful определяет URL основного ресурса, поддерживаемые операции и MIME-типы представления/ответа.

Пример запроса к REST веб-сервису размещен ниже.

```
GET /StockPrice/IBM HTTP/1.1
Host: example.org
Accept: text/xml
Accept-Charset: utf-8
```

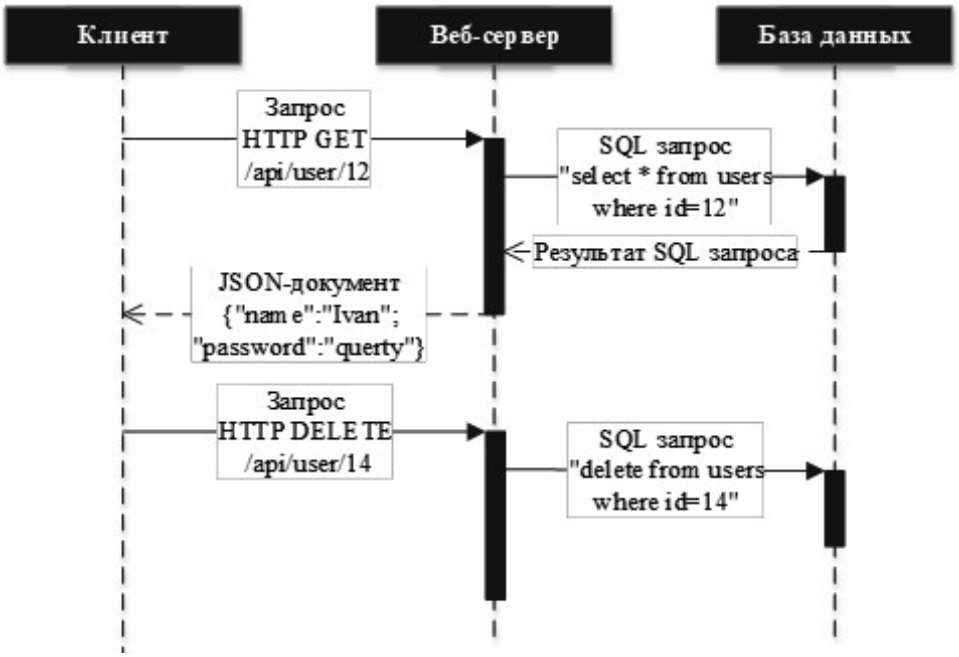


Рисунок 49 – REST веб-сервис

Пример ответа на запрос к REST веб-сервису представлен ниже.

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8;
Content-Length:167;

<?xmlversion="1.0"?>
<s:Quote xmlns:s="http://example.org/stock-service">
<s:TickerSymbol>IBM</s:TickerSymbol>
<s:StockPrice>45.25</s:StockPrice>
</s:Quote>
```

Ресурсом может являться практически любой понятный и значимый адресуемый объект.

Представление ресурса – обычно это документ, отражающий текущее или требуемое состояние ресурса. Ресурсы чаще всего представляются документами в форматах XML или JSON.

JSON (JavaScript object notation) – эффективный текстовый формат кодирования данных, обеспечивающий быстрый обмен небольшими объемами данных между клиентскими браузерами и веб-службами с поддержкой AJAX.

JSON-текст представляет собой (в закодированном виде) одну из двух структур:

1 Набор пар *«ключ: значение»*. В различных языках это реализовано как объект, запись, структура, словарь, хеш-таблица, список с ключом или ассоциативный массив. Ключом может быть только строка, значением – любая форма.

2 Упорядоченный набор значений. Во многих языках это реализовано как массив, вектор, список или последовательность.

REST и горизонтальный подход. Стратегия, опирающаяся на горизонтальный подход к протоколам, наиболее радикальна. Слово «горизонтальный» означает в данном случае сохранение существующего уровня без выстраивания уровней поверх него. Предполагается отказаться от разработки новых протоколов и использовать несколько хорошо проверенных, считая, что для работы с объектами вполне достаточно уметь выполнять четыре типа действий: создание (creation), восстановление (retrieval), изменение (update) и уничтожение (destruction). Из этих действий получается так называемый «шаблон проектирования» CRUD. Протокол Hypertext Transfer Protocol определяет методы GET/PUT/POST/DELETE (таблица 21), которые и реализуют шаблон CRUD. Абревиатура (шаблон) CRUD обозначает перечень основных операций с объектом: create (создать), read (считать, загрузить), update (обновить, изменить, отредактировать) и delete (удалить).

Таблица 21 – HTTP-методы в REST веб-сервисах

HTTP-метод	**CRUD операция**
GET	Read
POST	Create (иногда используется для операций update, delete)
PUT	Create (иногда используется для операции update)
DELETE	Delete

Разработчики SOAP веб-сервисов создают свой собственных перечень имен существительных и глаголов, например, getUsers(), savePurchaseOrder(), для обозначения операций веб-сервиса. В этом смысле SOAP реализуют сервисный принцип работы, в соответствии с которым главную роль при взаимодействии со службами играют их методы.

В основе архитектуры REST веб-сервисов лежит ресурсный (объектный) подход.

REST веб-сервисы разрабатываются согласно следующим принципам:

- возвращайте любые данные по их идентификатору;
- use standard methods – используйте стандарты, т. е. экономьте свои силы и деньги заказчика, используйте стандартные методы HTTP;
- одни и те же данные можно вернуть в различных форматах. Например, в XML или JSON для последующей программной обработки;
- передача данных без сохранения состояния. При обращении к REST-сервису не учитываются результаты ранее выполненных операций. REST-приложение не сохраняет никакого состояния сессии на стороне сервера. Вся информация, необходимая для выполнения запроса, передается в самом запросе.

Часто REST веб-сервисы реализуются с помощью инфраструктуры создания MVC веб-приложений.

Принципы SOLID. В упрощенном варианте: если при написании кода используется несколько принципов вместе, то это значительно облегчает дальнейшую поддержку и развитие программы. Полностью акроним расшифровывается следующим образом:

- ***S**ingle responsibility principle* – принцип единственной обязанности/ ответственности (на каждый класс должна быть возложена одна единственная обязанность);
- ***O**pen/closed principle* – принцип открытости/закрытости (программные сущности должны быть закрыты для изменения, но открыты для расширения);
- ***L**iskov substitution principle* – принцип подстановки Барбары Лисков (функции, которые используют базовый тип, должны иметь возможность использовать подтипы базового типа, не зная об этом; подклассы не могут замещать поведения базовых классов; подтипы должны дополнять базовые типы);
- ***I**nterface segregation principle* – принцип разделения интерфейса (много специализированных интерфейсов лучше, чем один универсальный);
- ***D**ependency inversion principle* – принцип инверсии зависимостей (зависимости внутри системы строятся на основе абстракций; модули верхнего уровня не зависят от модулей нижнего уровня; абстракции не должны зависеть от деталей; детали должны зависеть от абстракций).

Пример реализации принципов SOLID представлен в источнике [21].

DRY (Don’t Repeat Youself). Этот принцип заключается в том, что нужно избегать повторений одного и того же кода. Лучше использовать универсальные свойства и функции.

Пример реализации данного принципа на языке JavaScript представлен ниже.

```
let chips = ['corn', 'pita', 'potato', 'tortilla'];

  // код, не соответствующий принципу DRY
  console.log(chips[0]);
  console.log(chips[1]);
  console.log(chips[2]);

  // код, соответствующий принципу DRY
  for (let i = 0; i < chips.length; i++) {
    console.log(chips[i]);
  }
```

KISS (Keep It Simple, Stupid). Смысл этого принципа программирования заключается в том, что стоит делать максимально простую и понятную архитектуру, применять шаблоны проектирования.

Пример реализации данного принципа представлен ниже.

```
// код, не соответствующий принципу KISS
public String weekday1(int dayOfWeek)
{
    switch (dayOfWeek)
    {
        case 1: return "Monday";
        case 2: return "Tuesday";
```

```
        case 3: return "Wednesday";
        case 4: return "Thursday";
        case 5: return "Friday";
        case 6: return "Saturday";
        case 7: return "Sunday";
        default: throw new IllegalArgumentException("dayOfWeek must be in
range 1..7");
    }
}
// код, соответствующий принципу KISS
public String weekday2(int dayOfWeek)
{
    if ((dayOfWeek < 1) || (dayOfWeek > 7))
        throw new IllegalArgumentException("dayOfWeek must be in range
1..7");

    final String[] weekdays = {
        "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday",
"Sunday"};
    return weekdays[dayOfWeek-1];
}
```

Внедрение зависимости (dependency injection; шаблон DI) – шаблон проектирования, суть которого заключается в следующем: зависимый объект (или функция) получает другой объект (или функцию), от которого он зависит. Данный шаблон позволяет уменьшать степень связанности программных компонентов. Другими словами, он дает возможность сократить риск возникновения непредвиденных изменений в программных элементах ввиду внесения изменений в другие части программы (например, при изменении провайдера данных необходимо изменить бизнес-логику программы). На текущий момент данный шаблон получил широкое распространение среди разработчиков программного обеспечения, т. к. позволяет создавать качественный программный код и строить более эффективные программные архитектуры.

ПРИМЕР РЕАЛИЗАЦИИ REST API-СЕРВИСА

Согласно требованиям заказчика необходимо разработать REST API-сервис, внутренняя архитектура которого представлена на рисунке 50.

Фрагмент диаграммы классов в соответствии с выбранной архитектурой представлен на рисунке 51.

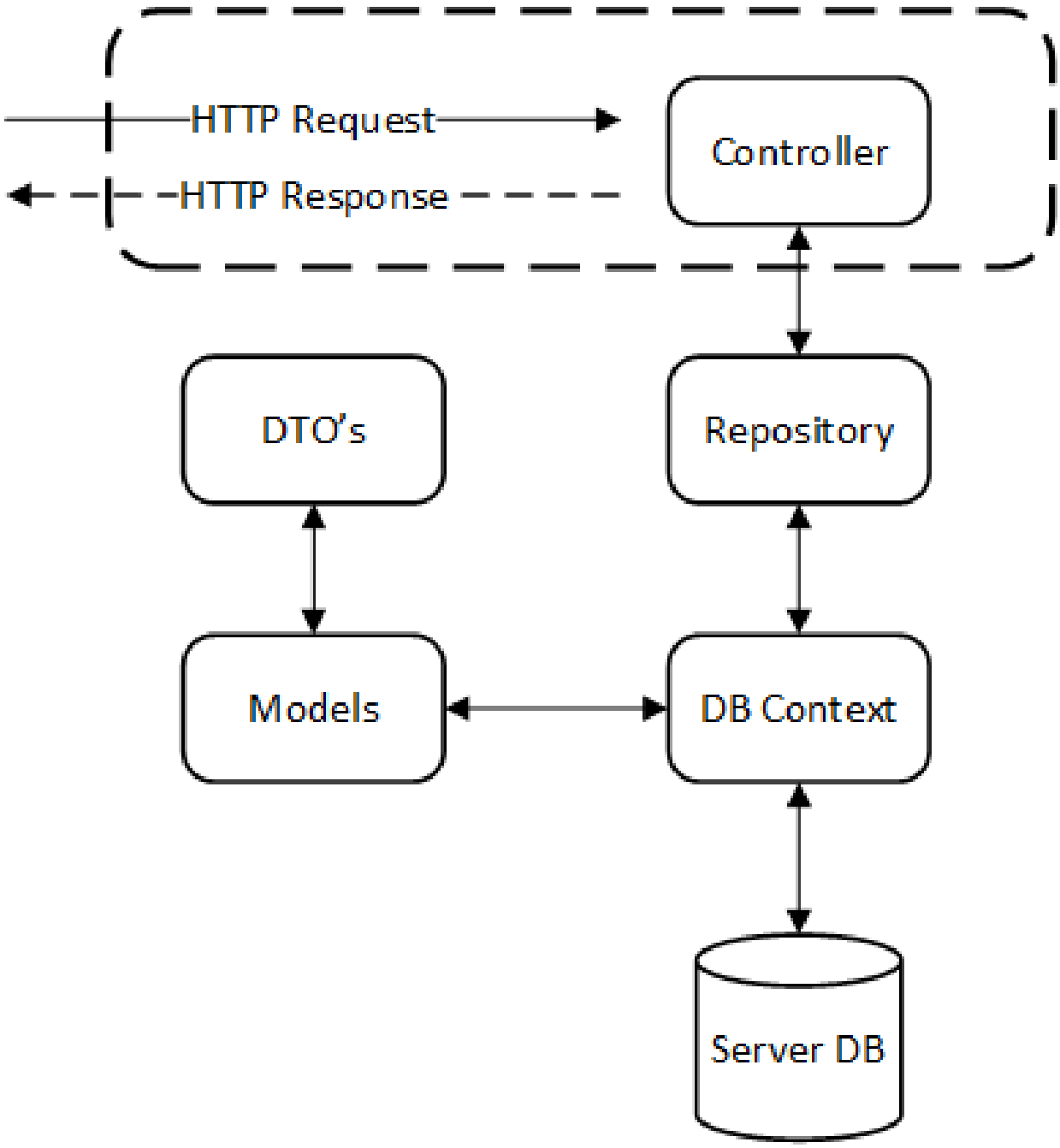


Рисунок 50 – Архитектура REST API-сервиса

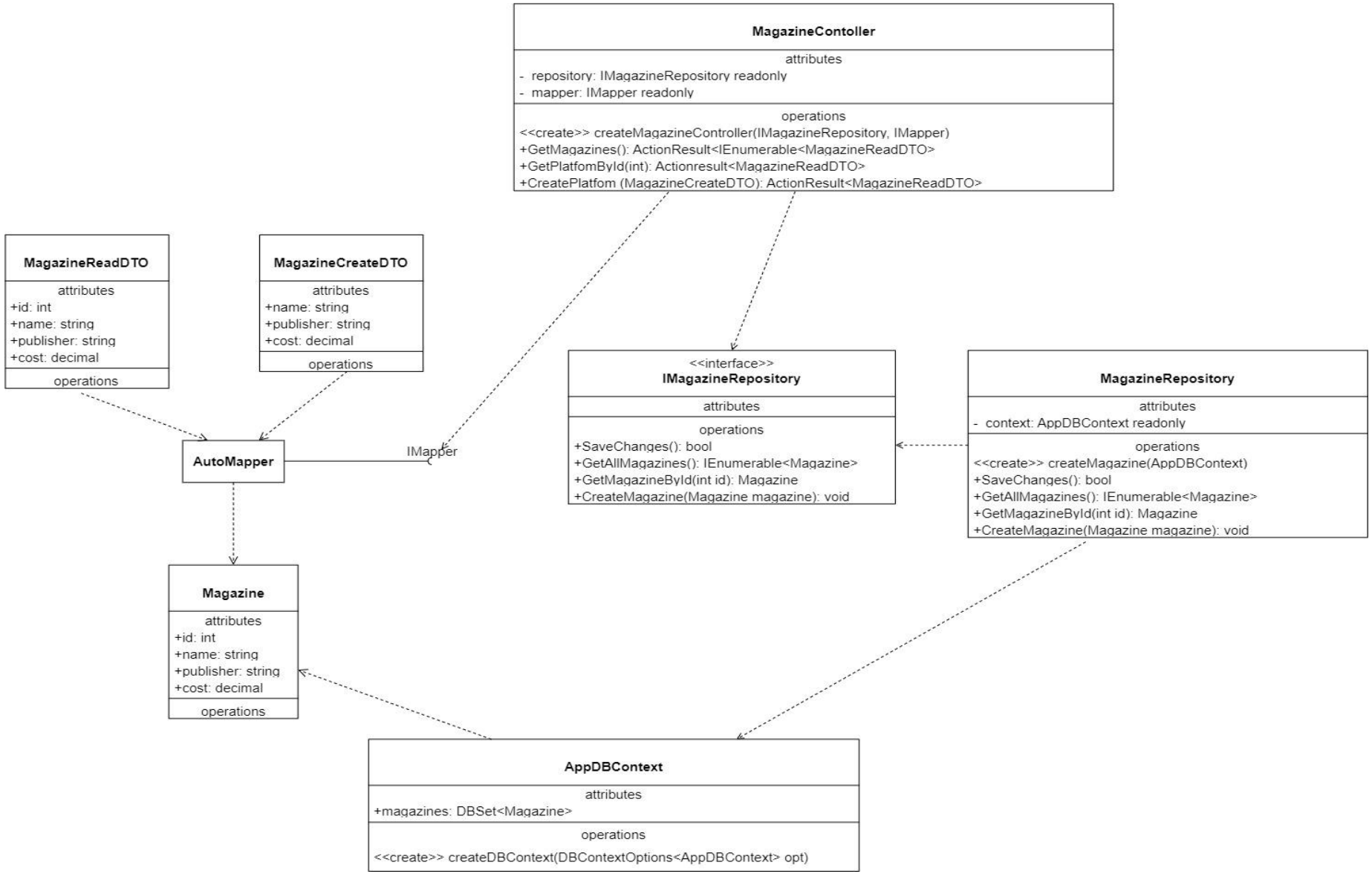


Рисунок 51 – Фрагмент диаграммы классов сервиса

Программный код, реализующий диаграмму классов, представленную на рисунке 51, размещен ниже.

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System.ComponentModel.DataAnnotations;

namespace Magazine.API
{
    public class Magazine
    {
        [Key]
        [Required]
        public int Id { get; set; }

        [Required]
        public string Name { get; set; }

        [Required]
        public string Publisher { get; set; }

        [Required]
        public decimal Cost { get; set; }
    }

    public class MagazineReadDTO
    {

        public int Id { get; set; }
        public string Name { get; set; }
        public string Publisher { get; set; }
        public decimal Cost { get; set; }
    }

    public class MagazineCreateDTO
    {
        [Required]
        public string Name { get; set; }
        [Required]
        public string Publisher { get; set; }
        [Required]
        public decimal Cost { get; set; }
    }

    public class AppDbContext : DbContext
    {
        public AppDbContext(DbContextOptions<AppDbContext> opt) : base(opt)
        {
        }
        public DbSet<Magazine> magazines { get; set; }
    }

    public interface IMagazineRepository
    {
        bool SaveChanges();

        IEnumerable<Magazine> GetAllMagazines();
        Magazine GetMagazineById(int id);
        void CreateMagazine(Magazine magazine);
    }
```

```
    public class MagazineRepository : IMagazineRepository
    {
        private readonly AppDbContext _context;

        public MagazineRepository(AppDbContext context)
        {
            _context = context;
        }

        public void CreateMagazine(Magazine magazine)
        {
            if (magazine == null)
            {
                throw new ArgumentNullException(nameof(magazine));
            }

            _context.Magazines.Add(magazine);
        }

        public IEnumerable<Magazine> GetAllMagazines()
        {
            return _context.Magazines.ToList();
        }

        public Magazine GetMagazineById(int id)
        {
            return _context.Magazines.FirstOrDefault(m => m.Id == id);
        }

        public bool SaveChanges()
        {
            return (_context.SaveChanges() >= 0);
        }
    }

    [Route("api/[controller]")]
    [ApiController]
    public class MagazinesController : ControllerBase
    {
        private readonly IMagazineRepository _repository;
        private readonly IMapper _mapper;

        public MagazinesController(IMagazineRepository repository, IMapper map-
per)
        {
            _repository = repository;
            _mapper = mapper;
        }

        [HttpGet]
        public ActionResult<IEnumerable<MagazineReadDTO>> GetMagazines()
        {
            var magazineItem = _repository.GetAllMagazines();

            return Ok(_mapper.Map<IEnumerable<MagazineReadDTO>>(magazineItem));
        }

        [HttpGet("{id}", Name = "GetMagazineById")]
```

```
        public ActionResult<MagazineReadDTO> GetMagazineById(int id)
        {
            var magazineItem = _repository.GetMagazineById(id);

            if (magazineItem != null)
            {
                return Ok(_mapper.Map<MagazineReadDTO>(magazineItem));
            }

            return NotFound();
        }

        [HttpPost]
        public async Task<ActionResult<MagazineReadDTO>> CreateMagazine(
        MagazineCreateDTO magazineCreateDTO)
        {
            var magazineModel = _mapper.Map<Magazine>(magazineCreateDTO);
            repository.CreateMagazine(magazineModel);
            repository.SaveChanges();

            var magazineReadDTO = _mapper.Map<magazineReadDTO>(magazineModel);

            return CreatedAtRoute(nameof(GetMagazineById),
            new { Id = magazineReadDTO.Id }, magazineReadDTO);
        }
    }

}
```

Представленный код содержит пример реализации:

- принципа единственной ответственности (например, класс AppDBContext);
- принципа DRY (например, метод SaveChanges());
- принципа KISS (например, метод CreateMagazine());
- внедрения зависимости (например, переменная private readonly IMagazineRepository _repository).

---

1 В рамках лабораторной работы необходимо реализовать функциональность серверной части программного средства в виде REST API (тип API можно изменить. Если будет выбран другой тип API, нужно привести обоснование выбора).

2 Основная информация по проектированию API представлена в источнике [22].

3 Результат выполнения лабораторной работы – программный код, соответствующий принципам SOLID, DRY, KISS, а также использующий шаблон DI. В тексте отчета необходимо привести примеры кода с указанием того, какой именно принцип был реализован. Также необходимо представить UML-диаграммы вариантов использования, классов и развертывания.

4 В отчете необходимо представить документацию к разработанному API в виде таблицы (пример описания был взят из источника по ссылке: https://github.com/Medium/medium-api-docs#2-authentication), шаблон которой представлен в таблице 22.

Таблица 22 – Описание запросов и ответов UC-8

Параметр	Описание
Вариант использования	UC-8 Создать публикацию в профиле авторизованного пользователя
URL	`POST https://api.medium.com/v1/users/{{authorId}}/posts`

Пример запроса	`POST /v1/users/5303d74c64f66366f00cb9b2a94f3251bf5/posts HTTP/1.1`
	`Host: api.medium.com`
	`Authorization: Bearer 181d415f34379af07b2c11d144dfbe35d`
	`Content-Type: application/json`
	`Accept: application/json`
	`Accept-Charset: utf-8`
	
	`{`
	`  "title": "Liverpool FC",`
	`  "contentFormat": "html",`
	`  "content": "<h1>Liverpool FC</h1><p>You'll never walk alone.</p>",`
	`  "canonicalUrl": "http://jamietalbot.com/posts/liverpool-fc",`
	`  "tags": ["football", "sport", "Liverpool"],`
	`  "publishStatus": "public"`
	`}`

Пример ответа	`HTTP/1.1 201 OK`
	`Content-Type: application/json; charset=utf-8`
	
	`{`
	`  "data": {`
	`  "id": "e6f36a",`
	`  "title": "Liverpool FC",`
	`  "authorId": "5303d74c64f66366f00cb9b2a94f3251bf5",`
	`  "tags": ["football", "sport", "Liverpool"],`
	`  "url": "https://medium.com/@majelbstoat/liverpool-fc-e6f36a",`
	`  "canonicalUrl": "http://jamietalbot.com/posts/liverpool-fc",`
	`  "publishStatus": "public",`
	`  "publishedAt": 1442286338435,`
	`  "license": "all-rights-reserved",`
	`  "licenseUrl": "https://medium.com/policy/9db0094a1e0f"`
	`  }`
	`}`

Тип и описание параметров запроса и ответа на запрос представлены в таблицах 23, 24.

Таблица 23 – Тип и описание параметров запроса (UC-8)

Параметр	Тип	Обязательный	Описание
1	2	3	4
title	String	Да	Заголовок публикации. Используется для SEO и отображается в случае представления публикаций в виде списка (в фактической публикации заголовок не отображается; чтобы заголовок был отображен в фактической публикации, необходимо указать его в поле content).

Продолжение таблицы 23

1	2	3	4
			Заголовки, длина которых больше 100 символов, будут игнорироваться. В этом случае заголовок будет сгенерирован из текста публикации при ее публикации
contentFormat	String	Да	Формат поля content. Допустимы два возможных типа: html, markdown
content	String	Да	Тело публикации – семантически валидный фрагмент HTML или Markdown
tags	String array	Нет	Теги для классификации публикаций. Использоваться будут только первые три тега. Теги, длина которых более 25 символов, будут игнорироваться
canonicalUrl	String	Нет	URL исходной публикации, если изначально она была опубликована в другом месте
publishStatus	Enum	Нет	Статус публикации. Допустимые значения: public, draft, unlisted. Значение по умолчанию: public
license	Enum	Нет	Лицензия публикации. Допустимые значения: all-rights-reserved, cc-40-by, cc-40-by-sa, cc-40-by-nd, cc-40-by-nc, cc-40-by-nc-nd, cc-40-by-nc-sa, cc-40-zero, public-domain. Значение по умолчанию: all-rights-reserved
notifyFollowers	Bool	Нет	Параметр указывает, необходимо ли уведомлять подписчиков о публикации данной статьи

Таблица 24 – Описание и тип параметров ответа на запрос (UC-8)

Поле	Тип	Описание
id	String	Уникальный идентификатор публикации
title	String	Заголовок публикации
authorId	String	Уникальный идентификатор пользователя, опубликовавшего статью
tags	String array	Теги публикации
url	String	URL публикации
canonicalUrl	String	URL исходной публикации. Если данное поле не было специфицировано в процессе создания публикации, то оно будет отсутствовать
publishStatus	String	Статус публикации
publishedAt	Timestamp	Дата публикации. Если была создана публикация со статусом draft, это поле не будет отображаться
license	Enum	Лицензия публикации
licenseUrl	String	URL лицензии публикации

Перечень ошибок, которые могут возникнуть в процессе выполнения запроса, представлен в таблице 25.

Таблица 25 – Перечень возможных ошибок (UC-8)

Код ошибки	Описание
400 Bad Request	Обязательные поля недействительны или не указаны
401 Unauthorized	Маркер доступа (access token) недействителен или был отозван
403 Forbidden	Пользователь не имеет прав на публикацию или значение поля authorId в пути запроса указывает на неправильного/несуществующего пользователя

Аналогично следует описать функции, реализующие все варианты использования.

При выполнении лабораторной работы можно использовать сторонние библиотеки, фреймворки, шаблоны и прочие дополнительные элементы. Если были использованы сторонние компоненты, на них должна быть приведена ссылка в тексте отчета по лабораторной работе в следующем виде: *<название источника>*: *<URL>*.

Весь код серверной части необходимо разместить в публичном репозитории.

Контрольные вопросы к лабораторной работе № 9

1 Что такое REST?
2 Для чего используют RESTful API и как работает RESTful API?
3 Каковы преимущества и недостатки REST API?
4 Как используется HTTP при создании REST API?
5 Какова структура запроса REST API от клиента к серверу?
6 Какова структура ответа после выполнения REST API запроса?
7 Какие методы HTTP поддерживаются REST и когда они используются?
8 Каковы принципы REST?
9 Что такое коды состояния HTTP?
10 Что такое ресурс в REST?
11 В каком формате передаются данные при взаимодействии с REST вебсервисом?
12 Каким образом кодируются данные в формате JSON?
13 Каковы лучшие ресурсы для изучения REST API?
14 Что такое принципы SOLID и зачем они нужны?
15 Как применять принципы SOLID на практике?
16 Что означает принцип DRY?
17 Что лежит в основе принципа KISS?
18 В чем состоит концепция DI?

ЛАБОРАТОРНАЯ РАБОТА № 10
Тестирование программного средства

Цель выполнения лабораторной работы: разработать тест-кейсы и UNIT-тесты для проверки работоспособности программного средства.

Задание	**Этапы выполнения задания**
1 Провести тестирование разработанного программного средства с использованием тест-кейсов	Разработать тест-кейсы для проверки уровня базовых пользовательских требований и оформить результаты тестирования
2 Провести тестирование разработанного программного средства с использованием UNIT-тестов	1 Разработать UNIT-тесты для проверки работоспособности кода серверной части программного средства и оформить результаты тестирования. 2 Оценить покрытие кода тестами
3 Сформировать отчет по лабораторной работе	*Содержание отчета:* Титульный лист (приложение Б) 1 Ссылка на публичный репозиторий Github 2 Тест-кейсы для проверки уровня базовых пользовательских требований 3 Перечень функциональных требований к программному средству 4 Результаты автоматизированного тестирования функциональности программного кода 5 Листинг кода тестирования функциональных требований 6 Оценка тестового покрытия кода 7 Выводы

Краткие теоретические сведения и методические указания к заданию 1

Тестирование (testing) – процесс, содержащий в себе все активности жизненного цикла, как динамические, так и статические, касающиеся планирования, подготовки и оценки программного продукта и связанных с этим результатом работ с целью определить, что они соответствуют описанным требованиям и показать, что они подходят для заявленных целей и для определения дефектов **[глоссарий RSTQB]**.

Подробная классификация видов тестирования представлена в источнике [23]. В рамках лабораторной работы для проверки уровня базовых пользовательских требований будем использовать тестирование на основе тест-кейсов.

Тестирование на основе тест-кейсов – формализованный подход, в котором тестирование производится на основе заранее подготовленных тест-кейсов, их наборов и иной документации.

Тест-кейс – набор входных данных, условий выполнения и ожидаемых результатов, разработанный с целью проверки того или иного свойства или поведения программного средства.

Общая структура тест-кейса может включать следующие структурные элементы:

- идентификатор;
- связанное с тест-кейсом требование;
- модуль и подмодуль приложения;
- заглавие;
- исходные данные, приготовления к тест-кейсу;
- шаги тест-кейса (сценарий тест-кейса);
- ожидаемые результаты по каждому шагу тест-кейса;
- полученные результаты по каждому шагу тест-кейса.

В рамках лабораторной работы в качестве структурных элементов тест-кейса будем рассматривать:

- идентификатор;
- заглавие;
- шаги;
- ожидаемый результат.

Идентификатор тест-кейса – это уникальное значение, которое позволяет однозначно отличить один тест-кейс от другого.

Например, идентификатор UC-5 показывает связанное с тест-кейсом базовое пользовательское требование, которое приведено в спецификации вариантов использования.

Заглавие необходимо для понимания основной идеи тест-кейса без обращения к его остальным атрибутам. Оно должно быть информативным и уникальным.

Шаги описывают последовательность действий, которые необходимо реализовать в процессе выполнения тест-кейса.

Ожидаемые результаты по каждому шагу описывают реакцию приложения на действия, описанные в поле «шаги тест-кейса». Номер шага соответствует номеру результата. Результат тест-кейса может быть положительным (фактический результат совпадает с ожидаемым), отрицательным (фактический результат отличается от ожидаемого) либо тест-кейс может быть не завершен (в процессе проверки происходит ошибка).

При описании тест-кейсов для проверки уровня базовых пользовательских требований может быть использован шаблон, представленный в таблице 26.

Таблица 26 – Тест-кейсы для проверки уровня базовых пользовательских требований

Идентификатор тест-кейса	Заглавие тест-кейса	Шаги тест-кейса	Ожидаемый результат

Общие требования к написанию шагов тест-кейса:

- заглавие должно быть информативным (понятно, что делает тест-кейс);
- при написании заглавия следует использовать безличную форму глагола;
- один тест-кейс должен проверять только одну функцию;
- шаги должны быть пронумерованы (даже если шаг всего один), описаны понятно и трактоваться однозначно;
- ожидаемый результат должен быть понятным;
- ожидаемые результаты обязательно следует приводить для каждого шага проверки.

В рамках лабораторной работы следует:

- разработать тест-кейсы для проверки уровня **всех базовых пользовательских требований** разработанного программного средства (перечень);
- протестировать все базовые пользовательские требования;
- результаты тестирования оформить в виде таблицы 26.

Краткие теоретические сведения и методические указания к заданию 2

Модульное тестирование (Unit Testing, unit-тестирование) – это тип тестирования программного обеспечения, при котором тестируются отдельные модули или компоненты программного обеспечения. Цель модульного тестирования: проверить, что каждая единица программного кода работает должным образом.

Данный вид тестирования выполняют на этапе кодирования приложения. Модульные тесты изолируют часть кода и проверяют его работоспособность. Единицей для измерения может служить отдельная функция, метод, процедура, модуль или объект.

Отсутствие модульного тестирования при написании кода значительно увеличивает уровень дефектов при дальнейшем (интеграционном, системном и приемочном) тестировании.

Модульное тестирование делят на **ручное** и **автоматизированное**. На практике чаще используется автоматизированное тестирование.

Алгоритм автоматизированного тестирования

1 В приложение записывают единицу кода, чтобы протестировать ее. Делают комментарий и далее удаляют тестовый код при развертывании приложения.

2 Для проведения более качественного тестирования изолируют единицу кода, что подразумевает копирование кода в собственную среду тестирования. Данная процедура позволяет выявить ненужные зависимости между тестируемым кодом и другими компонентами (модулями) или пространствами данных в программном изделии.

3 Для разработки автоматизированных тестовых случаев применяют какой-либо UnitTest Framework. Используя инфраструктуру автоматизации, задаются критерии теста для проверки корректного выполнения кода и в процессе выполнения тестовых случаев фиксируются неудачные. Большинство фреймворков автоматически выделяют и сообщают о неудачных тестах и могут остановить последующее тестирование, опираясь на серьезность сбоя.

4 Проводят модульное тестирование. Алгоритм модульного тестирования:

- создание тестовых случаев;
- просмотр/переработка;
- базовая линия;
- выполнение тестовых случаев.

ПРИМЕР UNIT-ТЕСТА ДЛЯ ПРОВЕРКИ РАБОТОСПОСОБНОСТИ КЛАССА НА ЯЗЫКЕ JAVA С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕКИ JUNIT

```
// тестируемый класс
public class Calculator {

    public int getSum(int x, int y) {
        return x+y;
    }

    public int getDivide(int x, int y) {
        return x/y;
    }

    public int getMultiple(int x, int y) {
        return x*y;
    }

}

// тестирующий класс
public class CalculatorTest {

    private Calculator calculator;

    @BeforeClass
    public static void beforeClass() {
        System.out.println("Before CalculatorTest.class");
    }

    @AfterClass
```

```
    public  static void afterClass() {
        System.out.println("After CalculatorTest.class");
    }

    @Before
    public void initTest() {
        calculator = new Calculator();
    }

    @After
    public void afterTest() {
        calculator = null;
    }

    @Test
    public void testGetSum() throws Exception {
        assertEquals(15, calculator.getSum(7,8));
    }

    @Test
    public void testGetDivide() throws Exception {
        assertEquals(20, calculator.getDivide(100,5));
    }

    @Test
    public void testGetMultiple() throws Exception {

    }

    @Test(expected = ArithmeticException.class)
    public void divisionWithException() {
        calculator.getDivide(15,0);
    }

    @Ignore("Message for ignored test")
    @Test
    public void ignoredTest() {
        System.out.println("will not print it");
    }

    @Test(timeout = 500)
    public void timeStampTest() {
        while (true);
    }
}
```

В рамках лабораторной работы следует:

- разработать UNIT-тесты для проверки работоспособности **всех функциональных возможностей** (перечень см. в задании 1 лабораторной работы № 4 (таблица 12)) разработанного программного средства;
- результаты тестирования свести в таблицу, которая должна содержать следующие графы (таблица 27):

Тестируемый тип – функция, модуль, системы и т. д. (программная единица, подвергаемая тестированию);

Дата проведения теста;

Результаты тестирования – краткая информация о проведенном тесте и его результаты;

Примечания – дополнительная информация, полученная в процессе проведения тестирования программной единицы;

- оценить степень покрытия кода тестами (см. источник [24]).

Таблица 27 – Результаты автоматизированного тестирования функциональности ПС

Тестируемый тип	Дата проведения теста	Результаты тестирования	Примечания

При выполнении лабораторной работы можно использовать сторонние библиотеки, фреймворки и прочие дополнительные элементы. Если были использованы сторонние компоненты, на них должна быть приведена ссылка в тексте отчета по лабораторной работе в следующем виде: <*название источника*>: <*URL*>.

Весь код UNIT-тестов должен быть представлен в публичном репозитории.

Контрольные вопросы к лабораторной работе № 10

1 Что такое тестирование и как его выполняют?
2 Какие виды тестирования бывают?
3 Что такое тест-кейс и каковы основные его атрибуты?
4 Какие категории ошибок выявляет тестирование на основе тест-кейсов?
5 Что представляют собой наборы тест-кейсов?
6 Каковы задачи модульного тестирования?
7 Какие принципы лежат в основе модульного тестирования?
8 Когда целесообразно использовать модульное тестирование?
9 Каковы особенности модульного тестирования?
10 Какие категории ошибок выявляет модульное тестирование?
11 Какие преимущества и недостатки имеет модульное тестирование?
12 Какие существуют типы тестов по покрытию?
13 Как измеряется покрытие кода?

ЛАБОРАТОРНАЯ РАБОТА № 11
Внедрение программного средства. Оценка качества разработанного программного средства

Цели выполнения лабораторной работы:

- составить эксплуатационные документы программного средства;
- оценить качество разработанного программного средства.

Задание	**Этапы выполнения задания**
1 Разработать руководство по установке (развертыванию) программного средства	Составить руководство по установке (развертыванию)
2 Разработать руководство пользователя программного средства	Составить руководство пользователя программного средства
3 Оценить качество разработанного программного средства	1 Изучить серию международных стандартов SQuaRE. 2 Представить количественную оценку характеристик качества программного средства
4 Сформировать отчет по лабораторной работе	*Содержание отчета:* Титульный лист (приложение Б) 1 Руководство по установке (развертыванию) программного средства 2 Руководство пользователя программного средства 3 Таблица результатов количественной оценки характеристик качества программного средства 4 Выводы

Краткие теоретические сведения и методические указания к заданию 1

Руководство по установке (развертыванию) программного средства в большинстве случаев предназначено для сотрудников технического отдела организации-заказчика. В зависимости от масштаба разработанного программного средства, а также масштаба самой организации руководство по установке (развертыванию) может иметь различное содержание.

В рамках выполнения лабораторной работы необходимо представить руководство по установке (развертыванию) в соответствии со следующей структурой (при формировании содержания руководства по установке (развертыванию) были использованы материалы ресурса https://rykovodstvo.ru/exspl/143299/index.html):

1 Необходимые программы и компоненты. Описать все программные и аппаратные компоненты, которые должны быть установлены на ПК конечного пользователя для корректной работы ПС.

ПРИМЕР ОПИСАНИЯ ПРЕДУСТАНОВЛЕННЫХ КОМПОНЕНТОВ

Для успешной установки и запуска программного средства необходимо наличие следующих компонентов:

- операционная система семейства Linux;
- веб-сервер ApacheTomcat;
- Java Development Kit.

2 Последовательность установки. Представить цепочку действий, выполнение которых приведет к установке разработанного ПС на ПК конечного пользователя.

ПРИМЕР ОПИСАНИЯ ПОСЛЕДОВАТЕЛЬНОСТИ ШАГОВ УСТАНОВКИ

Для установки разработанного программного средства необходимо выполнить следующие шаги:

Шаг 1. Назначить права для пользователя, под которым будет проводиться установка и настройка.

Шаг 2. Установить и настроить Java.

Шаг 3. Установить и настроить сервер приложений Apache Tomcat.

Шаг 4. Установить и настроить сервер баз данных Postgre SQL.

Шаг 5. Установить и настроить разработанное ПС.

3 Состав дистрибутива. Указать, какие элементы входят в дистрибутив для установки и развертывания ПС.

ПРИМЕР ОПИСАНИЯ СОСТАВА ДИСТРИБУТИВА

В поставляемый конечному пользователю дистрибутив входят следующие элементы:

- скрипт генерации пустой БД;
- файл ПС с расширением .exe.

4 Распаковка дистрибутива. Распаковка дистрибутива имеет место быть **только в случае**, если установка ПС осуществляется в определенные системные директории ОС. Если распаковка дистрибутива осуществляется в новую или создаваемую при установке дистрибутива директорию, то данный раздел руководства можно опустить.

ПРИМЕР ОПИСАНИЯ РАСПАКОВКИ ДИСТРИБУТИВА

Для распаковки дистрибутива необходимо выполнить следующие шаги:

- удалить содержимое папки /var/apache-tomcat-7.0.67/webapps/ROOT;
- распаковать имеющийся дистрибутив подсистемы в папку /var/apache-tomcat-7.0.67/webapps/ROOT.

5 Восстановление БД из резервной копии. Представить процесс восстановления БД, необходимой для корректной работы ПС, из скрипта БД, поставляемого в составе дистрибутива ПС.

ПРИМЕР ОПИСАНИЯ ПРОЦЕССА ВОССТАНОВЛЕНИЯ БД

Для восстановления БД ПС из резервной копии необходимо выполнить действия, представленные ниже.

Для входа в консоль postgresql необходимо выполнить команду

```
sudo -u postgres psql
```

Для создания пустой БД необходимо выполнить команду

```
CREATE DATABASE tmp_crsmev_kf WITH ENCODING='UTF8' CONNECTION LIMIT=-1;
```

В случае удачного выполнения указанной команды в консоли появится сообщение «CREATE DATABASE».

Для задания прав пользователю tomcat на новую БД необходимо выполнить команду

```
GRANT ALL privileges ON DATABASE tmp_crsmev_kf TO tomcat;
```

В случае удачного выполнения указанной команды в консоли появится сообщение «GRANT».

Для выхода из консоли postgresql необходимо набрать **\q** и нажать клавишу **Enter**.

Для выполнения скрипта из файла db_new_1.txt необходимо выполнить команду (*предполагается, что файл скрипта находится в папке /home/user*)

```
sudo psql –U postgres tmp_crsmev_kf < /home/user/db_new_1.txt
```

Для восстановления БД из резервной копии crsmev_kf.backup необходимо выполнить команду (*предполагается, что файл резервной копии находится в папке /home/user*)

```
sudo pg_restore –U postgres –v –d tmp_crsmev_kf /home/user/crsmev_kf.backup
```

Для выполнения скрипта из файла db_new_2.txt необходимо выполнить команду (*предполагается, что файл скрипта находится в папке /home/user*)

```
sudo psql –U postgres tmp_crsmev_kf < /home/user/db_new_2.txt
```

Для выхода из консоли необходимо набрать **\q** и нажать клавишу **Enter**.

6 Настройка дистрибутива. В данном разделе требуется представить описание всех действий, которые необходимо выполнить в случае, если требуется изменение каких-либо системных файлов или файлов конфигурации (например, необходимо изменить настройки файла конфигурации после восстановления БД из резервной копии).

ПРИМЕР ОПИСАНИЯ НАСТРОЕК ДИСТРИБУТИВА

Для корректной работы ПС необходимо изменить настройки конфигурационных файлов в файле /var/apache-tomcat-7.0.67/conf/server.xml и отредактировать группу параметров host следующим образом:

```
unpackWARs="true" autoDeploy="true"
xmlValidation="false" xmlNamespaceAware="false"
```

В файле /var/apache-tomcat-7.0.67/webapps/ROOT/WEB-INF/sx-config.xml изменить параметры подключения к серверу баз данных путем редактирования секции database следующим образом:

```
driver="org.postgresql.Driver"
url="jdbc:postgresql://127.0.0.1/tmp_crsmev_kf?socketTimeout=60&
loginTimeout=60&connectTimeout=60&tcpKeepAlive=true"
username="tomcat"
password="12345678"
maxActive="100"
testWhileIdle="true"
testOnBorrow="false"
validationQuery="select 1"
minEvictableIdleTimeMillis="600000"
timeBetweenEvictionRunsMillis="20000"
maxWait="21000"
defaultTransactionIsolation="1"
stackTrace="true"
caseInsensitive="true"
```

Проверка работоспособности ПС. В данном разделе необходимо описать последовательность действий, выполнение которых будет сигнализировать об успешной установке ПС на ПК конечного пользователя.

ПРИМЕР ОПИСАНИЯ ПРОВЕРКИ РАБОТОСПОСОБНОСТИ ПС

Для проверки работоспособности ПО необходимо:

- убедиться в том, что сервис PostgreSql запущен;
- убедиться в том, что сервер приложений Tomcat запущен;
- используя любой интернет-браузер, перейти по адресу http://localhost:8080, где 8080 – порт, указанный при установке Tomcat.

В случае успешного перехода в окне браузера откроется домашняя страница ПС.

Краткие теоретические сведения и методические указания к заданию 2

Руководство пользователя ПС – документ, основным назначением которого является предоставление конечным пользователям ПС информации о работе программного обеспечения.

В лабораторной работе необходимо представить описание всех операций обработки данных, которые может осуществлять разработанное ПС.

В начале руководства (при формировании содержания руководства по пользователя были использованы материалы ресурса https://www.prj-exp.ru/patterns/pattern_user_guide.php) необходимо представить таблицу с описанием функций и задач, которые может выполнять разработанное ПС. Пример описания функций и задач, выполняемых ПС, представлен в таблице 28.

Таблица 28 – Пример таблицы с описанием функций и задач

Функции	Задачи	Описание
Обеспечивает многомерный анализ в табличной и графической формах	Визуализация отчетности	В ходе выполнения данной задачи пользователю системы предоставляется возможность работы с выбранным отчетом из состава преднастроенных
	Формирование табличных и графических форм отчетности	В ходе выполнения данной задачи пользователю системы предоставляется возможность формирования собственного отчета в табличном или графическом виде на базе преднастроенных компонентов

Выполнение каждой задачи очень часто разделяется между различными операциями обработки данных. После составления таблицы функций, предоставляемых ПС, необходимо описать каждую операцию по обработке данных в соответствии со следующей структурой:

- наименование операции;
- условия, при соблюдении которых возможно выполнение операции;

- подготовительные действия;
- основные действия в требуемой последовательности;
- заключительные действия (если в них есть необходимость);
- ресурсы, расходуемые на операцию.

Также можно дополнять описание операций иллюстрациями работы ПС, текстом сообщений в журналы логов и прочей дополнительной информацией. Пример описания операции представлен в таблице 29.

Таблица 29 – Пример описания операций, реализуемых ПС

Параметр	**Описание**
Задача «Визуализация отчетности»	
Операция 1	**Регистрация на портале ИАС КХД**
Условия, при соблюдении которых возможно выполнение операции	Компьютер пользователя подключен к корпоративной сети. Портал ИАС КХД доступен. ИАС КХД функционирует в штатном режиме
Подготовительные действия	На ПК пользователя необходимо выполнить дополнительные настройки, приведенные в пункте 3.2 настоящего документа
Основные действия в требуемой последовательности	На иконке «ИАС КХД» рабочего стола произвести двойной щелчок левой кнопкой мыши. В открывшемся окне в поле «Логин» ввести имя пользователя, в поле «Пароль» ввести пароль пользователя. Нажать кнопку «Далее»
Заключительные действия	Не требуются
Ресурсы, расходуемые на операцию	15–30 с
Операция 2	**Выбор отчета**
Условия, при соблюдении которых возможно выполнение операции	Успешная регистрация на Портале ИАС КХД
Подготовительные действия	Не требуются
Основные действия в требуемой последовательности	▪в появившемся окне «мастер создания рабочих книг» поставить точку напротив пункта «открыть существующую рабочую книгу» (рисунок 52); ▪выбрать нужную рабочую книгу и нажать кнопку «Откр.» (рисунок 53)
Заключительные действия	После завершения работы с отчетом необходимо выбрать пункт меню «Файл», далее выбрать пункт «Закрыть»
Ресурсы, расходуемые на операцию	15 с
Задача «Формирование табличных и графических форм отчетности»	

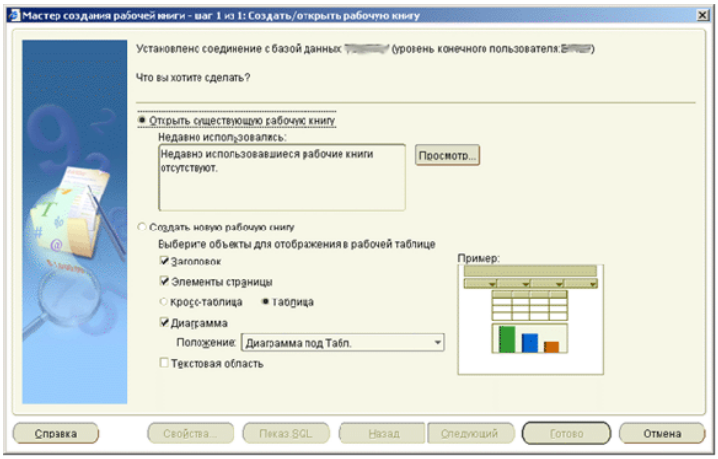


Рисунок 52 – Окно «Мастер создания рабочих книг»[1]

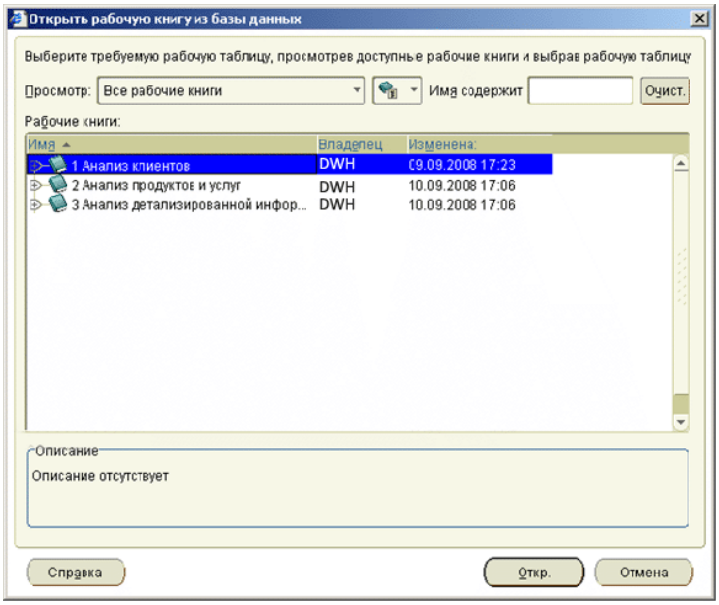


Рисунок 53 – Окно «Открыть рабочую книгу из базы данных»[1]

Краткие теоретические сведения и методические указания к заданию 2

Согласно 1061–1998 IEEE Standard for Software Quality Metrics Methodology **качество программного обеспечения (Software Quality) –** это степень, в которой программное обеспечение обладает требуемой комбинацией свойств.

В соответствии с ISO/IEC 25010:2011(ГОСТ Р ИСО/МЭК 25010–2015) «Системная и программная инженерия. Требования и оценка качества систем и ПО (SQuaRE). Модели качества систем и ПО» **качество системы** – это степень удовлетворения системой *заявленных и подразумеваемых потребностей* различных заинтересованных сторон.

[1] URL: https://www.prj-exp.ru/images/user_guide_1.gif.

Качество реализации ПС представляет особую важность ввиду зависимости от него множества показателей бизнеса. На данный момент существует несколько стандартов, позволяющих специфицировать требования к качеству реализации ПС, а также описать и подготовить сами процедуры оценки качества.

Наиболее часто используемыми на практике являются стандарты: ISO/IEC 9126, ISO/IEC 14598, ГОСТ 28195–99, ГОСТ 28806–90, ИСО/МЭК 9126–2003, серия стандартов ISO/IEC 25000 (SQuaRE).

Указанные стандарты оперируют следующими терминами в области оценки качества [25]:

1 **Атрибут (attribute)** – внутренне присущее свойство объекта, которое может быть распознано количественно или качественно человеком или автоматизированными средствами. Атрибуты могут быть внешними или внутренними.

2 **Качество (quality)** – совокупность характеристик программного продукта, относящаяся к его способности удовлетворять установленные и подразумеваемые потребности.

3 **Отказ (failure)** – прекращение способности продукта выполнять требуемую функцию или его неспособность работать в пределах заданных ограничений.

4 **Оценка качества (quality evaluation)** – систематическое исследование степени, в которой продукт способен к выполнению указанных требований.

5 **Ошибка (fault)** – некорректный шаг, процесс или определение данных в программе.

6 **Подразумеваемые потребности (implied needs)** – потребности, которые не были заданы, но являются реально существующими потребностями при использовании продукта в конкретных условиях. Подразумеваемые потребности включают потребности не заданные, но подразумеваемые другими заданными потребностями, и те незаданные потребности, которые рассматриваются как очевидные. Некоторые потребности становятся очевидными при использовании продукта в конкретных условиях.

7 **Подхарактеристика качества программного средства (software quality subcharacteristic)** – характеристика качества программного средства, входящая в состав другой характеристики качества.

8 **Характеристика качества программного средства (software quality characteristic)** – категория свойств (атрибутов) программного средства, с помощью которых описывается и оценивается его качество. Характеристики качества программных средств могут быть определены с помощью подхарактеристик и в конечном итоге атрибутов качества программного средства.

9 **Шкала (scale)** – набор значений с определенными свойствами.

При оценке качества используются следующие типы шкал (более подробная информация о типах шкал представлена в источнике [25]):

- *номинальная* – соответствует набору категорий; классифицирует программы по признаку наличия или отсутствия некоторого свойства без учета градаций (например, «да» или «нет»);

- *порядковая (упорядоченная)* – соответствует упорядоченному набору делений шкалы; позволяет ранжировать свойства путем сравнения с опорными значениями; имеет небольшое количество делений (например, шкала с четырьмя градациями «отлично», «хорошо», «удовлетворительно», «неудовлетворительно» или с двумя градациями, «удовлетворительно», «неудовлетворительно»);

- *интервальная* – соответствует упорядоченной шкале с равноудаленными делениями; обычно содержит достаточно большое количество делений с количественными значениями (например, шкала с делениями 0, 1, 2, …, 10);

- *отношений* – соответствует упорядоченной шкале с равноудаленными делениями, содержащей значение нуля, представляющего полное отсутствие атрибута.

Два первых типа шкал применяются для оценки качественных атрибутов ПС, которые нельзя измерить количественно, и для ранжирования измеренных значений, третий и четвертый типы – для оценки количественных атрибутов [25].

Вышеописанные стандарты регламентируют оценку качества ПС на основании того или иного вида модели качества.

В настоящий момент показатель качества регулируется международным стандартом ISO/IEC 25010:2011.

Заявленные и подразумеваемые потребности представлены в международных стандартах серии SQuaRE посредством моделей качества:

- модель качества при использовании (рисунок 54);
- модель качества продукта (рисунок 55).

Ввиду сложности, возникающей в ходе оценки качественных характеристик ПС по причине их субъективности, в рамках лабораторной работы необходимо провести оценку качественных характеристик ПС с использованием иерархической модели оценки качества ПС (рисунок 56).

Рисунок 54 – Модель качества при использовании

Рисунок 55 – Модель качества продукта

Функциональность	▪ Пригодность; ▪ правильность; ▪ способность к взаимодействию; ▪ согласованность; ▪ защищенность; ▪ соответствие функциональности
Надежность	▪ Завершенность; ▪ устойчивость к ошибке; ▪ восстанавливаемость; ▪ соответствие надежности
Практичность	▪ Понятность; ▪ обучаемость; ▪ простота использования; ▪ привлекательность; ▪ соответствие практичности
Эффективность	▪ Поведение во времени; ▪ использование ресурсов; ▪ соответствие эффективности
Сопровождаемость	▪ Анализируемость; ▪ изменяемость; ▪ стабильность; ▪ тестируемость ▪ соответствие сопровождаемости
Мобильность	▪ Адаптируемость; ▪ настраиваемость; ▪ совместимость; ▪ взаимозаменяемость; ▪ соответствие мобильности

Рисунок 56 – Иерархическая модель оценки качества ПС

Значения той или иной подхарактеристики целесообразно представлять в относительных единицах, для вычисления которых следует использовать одну из нижепредставленных формул [25]:

$$X = \frac{A}{B},$$

или

$$X = 1 - \frac{A}{B},$$

где X – значение метрики;

A – абсолютное (измеренное) значение некоторого свойства (атрибута) оцениваемого продукта, системы или документации;

B – базовое значение соответствующего свойства.

Результаты вычисления значений подхарактеристик следует свести в таблицу 30.

Таблица 30 – Результаты количественной оценки характеристик качества ПС [25]

Название подхарактеристики	**Относительное значение подхарактеристики**
Функциональность	
Пригодность	0,74
…	…
Надежность	
Завершенность	0,52
…	…

Контрольные вопросы к лабораторной работе № 11

1 В каком стандарте устанавливаются виды программ и программных документов для вычислительных машин, комплексов и систем независимо от их назначения и области применения?

2 Какие существуют виды программных документов?

3 Для чего предназначены эксплуатационные документы?

4 Какие виды эксплуатационных документов выделяют?

5 Какие элементы включает в себя руководство по установке (развертыванию) программных разработок?

6 Что такое руководство пользователя и для чего его создают?

7 Какова структура руководства пользователя?

8 В чем особенности разработки и структуры руководства пользователя для программного обеспечения?

9 Какие популярные инструменты используются для создания качественного руководства?

10 Что такое качество систем и программного обеспечения?

11 Зачем нужны модели качества?

12 Какими тремя главными аспектами характеризуется качество программного обеспечения?

13 Что представляет собой модель качества при использовании?

14 Что представляет собой модель качества продукта?

15 Какова связь модели качества при использовании и модели качества продукта?

16 Какие внутренние и внешние характеристика качества важны для программистов?

17 Как характеристики качества программного обеспечения влияют на процесс проектирования?

ПРИЛОЖЕНИЕ А

Варианты заданий для выполнения лабораторных работ

1 Проектирование и разработка программного средства автоматизации стратегий взаимодействия с заказчиками предприятия.

2 Проектирование и разработка программного средства автоматизации документооборота с реализацией поискового сервиса с возможностью просмотра вложенных и связанных документов.

3 Проектирование и разработка программного средства автоматизации управления гибкими льготами и магазином бенефитов.

4 Проектирование и разработка программного средства прогнозирования пространственного распределения вредных веществ в атмосферном воздухе.

5 Проектирование и разработка программного средства поддержки процессов электронного обучения учащихся в учреждениях общего среднего образования с использованием технологии виртуальной реальности.

6 Проектирование и разработка программного средства реализации интерактивного учебного пособия по учебной дисциплине «Технологии проектирования сложных информационных систем».

7 Проектирование и разработка программного средства автоматизации процессов формирования и распределения учебной нагрузки профессорско-преподавательского состава учреждения высшего образования.

8 Проектирование и разработка программного средства прогнозирования величины урожая на основании метеорологических данных с использованием нейронных сетей.

9 Проектирование и разработка программного средства управления корпоративными рисками на основе данных внутреннего аудита.

10 Проектирование и разработка программного средства поддержки процесса тестирования программного обеспечения на основе тестовых моделей.

11 Проектирование и разработка программного средства детектирования фейковых новостей в социальных сетях.

12 Проектирование и разработка программного средства анализа и прогнозирования конкурентоспособности IT-компании на рынке продуктов и услуг.

13 Проектирование и разработка программного средства поддержки принятия управленческих решений в организациях образовательного профиля.

14 Проектирование и разработка программного средства автоматизации процесса получения новых данных с использованием виртуальных графов знаний.

15 Проектирование и разработка программного средства управления многоуровневой программой лояльности для ритейла.

16 Проектирование и разработка программного средства прогнозирования динамики эпидемиологических показателей заболеваемости COVID-19 в Республике Беларусь.

17 Проектирование и разработка программного средства анализа и прогнозирования объема продаж в промоакциях на основе машинного обучения.

18 Проектирование и разработка программного средства перевода видео- и аудиозвонков в режиме реального времени.

19 Проектирование и разработка программного средства поиска заказов на фриланс-биржах с функцией интеллектуального подбора заданий на основании компетенций исполнителя.

20 Проектирование и разработка программного средства реализации образовательной платформы БГУИР.

21 Проектирование и разработка программного средства управления претензионно-исковой работой госорганов.

22 Проектирование и разработка программного средства автоматизации календарно-сетевого и ресурсного планирования проекта.

23 Проектирование и разработка программного средства управления омникальным сбором клиентской базы.

24 Проектирование и разработка программного средства планирования, учета и анализа выполнения преподавателем нагрузки по руководству практикой.

25 Проектирование и разработка программного средства автоматизации платежного календаря в рамках управления бюджетом и управленческого учета.

26 Проектирование и разработка программного средства автоматизации процесса формирования документов реализации в лизинговой компании.

27 Проектирование и разработка программного средства автоматизации работы со счетами в контексте процессного подхода к управлению предприятием.

28 Проектирование и разработка программного средства управления трудозатратами сотрудников IT-компании.

29 Проектирование и разработка программного средства управления запасами предприятия на основе анализа их движения и формирования рекомендаций.

30 Проектирование и разработка программного средства поддержки интеллектуальной обработки документов компании.

31 Проектирование и разработка программного средства управления командировками и авансовыми отчетами сотрудников.

32 Проектирование и разработка программного средства прогнозирования осадков на основе аппарата нейронных сетей.

33 Проектирование и разработка программного средства ранжирования и подбора новостей по заданной теме на основе методов машинного обучения.

34 Проектирование и разработка программного средства распознавания офтальмологических заболеваний на основе нейросетевых методов.

35 Проектирование и разработка программного средства управления данными об изделии в процессе конструкторско-технологической подготовки производства предприятия.

36 Проектирование и разработка программного средства управления закупками в организации.

37 Проектирование и разработка программного средства технической поддержки по обслуживанию IT-инфраструктуры в формате win-win.

38 Проектирование и разработка программного средства аттестации системы защиты информации.

39 Проектирование и разработка программного средства автоматизации задач производственной безопасности и охраны труда предприятия.

40 Проектирование и разработка программного средства обработки и визуализации данных об использовании кикшеринга.

41 Проектирование и разработка программного средства информационной поддержки принятия решений по реагированию на чрезвычайные ситуации и происшествия.

42 Проектирование и разработка программного средства анализа и формирования статистической отчетности по онлайн- и офлайн-продажам.

43 Проектирование и разработка программного средства автоматизации процессов кадрового планирования предприятия.

44 Проектирование и разработка программного средства автоматизации учета бизнес-процессов в системе корпоративного документооборота.

45 Проектирование и разработка программного средства управления программой лояльности для B2B/B2C коммерческих проектов.

46 Проектирование и разработка программного средства управления бонусной системой организации.

47 Проектирование и разработка программного средства оценки эффективности использования рабочего времени сотрудниками предприятия.

48 Проектирование и разработка программного средства анализа рисков кредитования физических лиц в коммерческом банке.

49 Проектирование и разработка программного средства создания среды разработки приложений в сети Интернет.

50 Проектирование и разработка программного средства анализа политики ценообразования продукции на предприятии и автоматизации расчетной методики.

51 Проектирование и разработка программного средства учета и анализа технического оборудования на балансе предприятия и расчет величины амортизационных отчислений.

52 Проектирование и разработка программного средства учета и анализа расходования материалов для изготовления изделий на предприятии.

53 Проектирование и разработка программного средства учета и анализа снятия денежных средств по пластиковым картам через банкоматы.

54 Проектирование и разработка программного средства учета и анализа движения денежных средств на предприятии.

55 Проектирование и разработка программного средства для ведения электронного документооборота и архива документов с использованием электронно-цифровой подписи.

56 Проектирование и разработка программного средства реализации информационного туристического сервиса Республики Беларусь с возможностью формирования маршрутов путешествий.

57 Проектирование и разработка программного средства реализации электронной записи в учреждения здравоохранения с функцией прогнозирования потока пациентов.

58 Проектирование и разработка программного средства для создания виртуального предприятия с возможностью оценки загруженности работников.

59 Проектирование и разработка программного средства поддержки процессов разработки новых продуктов и управления жизненным циклом программного продукта.

ПРИЛОЖЕНИЕ Б

Пример титульного листа отчета по лабораторной работе

Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования
Кафедра проектирования информационно-компьютерных систем
Технологии проектирования сложных информационных систем

Отчет
по лабораторной работе № X
на тему:
НАЗВАНИЕ ЛАБОРАТОРНОЙ РАБОТЫ

Проверил _____ И.И. Иванов
(подпись)

Выполнил _____ С.С. Сидоров
(подпись)
номер группы

Минск, 202X

ПРИЛОЖЕНИЕ В

Пример описания бизнес-процесса

Описание бизнес-процесса «*Прием товара на склад*»

1 Сначала менеджеру приходит уведомление о необходимости разместить на складе определенное количество товара. Менеджер просматривает справочники НСИ и при необходимости вносит в них данные о новых, ранее не встречаемых на складе товарах. После этого проводится анализ склада. Система предоставляет перечень ячеек, с которыми необходимо будет взаимодействовать в рамках задания. Само задание создается менеджером и содержит информацию о том, куда нужно поместить товар, какой товар и в каком количестве. Также на этом этапе создается задание для кладовщика, которому выделяется команда грузчиков после выбора задания.

2 Далее в работу включается грузчик, т. к. ему на портативное устройство приходит сигнал, свидетельствующий о наличии задания. Работник должен с ним ознакомиться и подтвердить получение. Данное действие предупреждает событие-таймер ожидания машины. По прибытии машины грузчик начинает выгрузку товара в специально отведенное помещение.

3 По прибытии машины кладовщик, который получил свое задание, должен сделать первоначальную проверку товара по накладной («пришло то, что написано») и дать согласие на разгрузку.

4 Грузчик выгружает товар в специально отведенное для этого место и отмечает задание как выполненное.

5 Кладовщик проводит детальную проверку (сверяет количество) и обозначает каждый товар в соответствии с зарезервированным номером в системе. Далее разносит товар по местам и отмечает задание как выполненное.

6 Менеджер видит, что задания по этому товару выполнены, и вносит подтверждение в систему о наличии товаров на складе. Также он отправляет в отдел, из которого ему пришла заявка, уведомление об успешной приемке товара. На этом процесс заканчивается.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Тонкович, И. Н. Разработка программного обеспечения: планирование, анализ и моделирование [Электронный ресурс]: метод. пособие / И. Н. Тонкович, А. В. Шелест. – Электрон. дан. – Минск : БГУИР, 2023 – Режим доступа: https://libeldoc.bsuir.by/handle/123456789/50325.

2 Тонкович, И. Н. Разработка программного обеспечения: проектирование, конструирование и внедрение [Электронный ресурс] : метод. пособие / И. Н. Тонкович, А. В. Шелест. – Электрон. дан. – Минск : БГУИР, 2023 – Режим доступа: https://libeldoc.bsuir.by/handle/123456789/51273.

3 Камаев, В. А. Технологии программирования: учеб. / В. А. Камаев, В. В. Костерин. – М. : Высш. шк., 2006. – 454 с.

4 Вигерс, К. Разработка требований к программному обеспечению / К. Вигерс, Дж. Битти. – М. : Русская редакция; СПб. : БХВ-Петербург, 2014. – 736 с.

5 Экономика проектных решений: методические указания по экономическому обоснованию дипломных проектов : учеб.-метод. пособие / В. Г. Горовой [и др.]. – Минск : БГУИР, 2021. – 107 с.

6 Ковалёв, С. Настольная книга аналитика. Практическое руководство по проектированию бизнес-процессов и организационной структуры / С. Ковалёв, В. Ковалёв. – М. : 1С-Паблишинг, 2021. – 360 с.

7 Свод знаний по управлению бизнес-процессами: BPM CBOK 3.0 / под ред. А. А. Белайчука, В. Г. Елиферова. – М. : Альпина Паблишер, 2016. – 480 с.

8 Репин, В. В. Бизнес-процессы. Моделирование, внедрение, управление / В. Репин, В. Елиферов. – М. : МИФ, 2013. – 512 с.

9 Автоматизация бизнес-процессов и внедрение сквозной аналитики в ИНЖПЛАСТ [Электронный ресурс]. – Режим доступа: https://alexrovich.ru/projects/avtomatizatsiya-otdelov-prodazh/avtomatizatsiya-biznes-protsessov-i-vnedrenie-skvoznoy-analitiki-v-inzhplast

10 Ричардс, М. Фундаментальный подход к программной архитектуре / М. Ричардс, Н. Форд. – СПб. : Питер, 2023. – 448 с.

11 Материал по разработке пользовательского интерфейса [Электронный ресурс]. – Режим доступа: https://livetyping.com/ru/blog/chto-takoe-razrabotka-polzovatelskogo-interfeisa-i- zachem-tt-zakazyvat.

12 Принципы разработки пользовательского интерфейса [Электронный ресурс]. – Режим доступа: https://clck.ru/DcKAK.

13 Этапы разработки пользовательского интерфейса [Электронный ресурс]. – Режим доступа: https://vc.ru/design/58502-etapy-razrabotki-polzovatelskogo-interfeysa-kak-sdelat-tak-chtoby-ui-ne-lishil-vas-pribyli

14 Дейт, К. Дж. Введение в системы баз данных / К. Дж. Дейт. – 8-е изд. – М. : Вильямс, 2005. – 1328 с.

15 ГОСТ 19.701–90 [Электронный ресурс]. – Режим доступа: https://www.swrit.ru/doc/espd/19.701-90.pdf.

16 ГОСТ 2.106–2013 [Электронный ресурс]. – Режим доступа: https://www.baf-psk.ru/doc/new/%D0%93%D0%9E%D0%A1%D0%A2-2_102_2013.pdf.

17 ГОСТ 19.002–80 [Электронный ресурс]. – Режим доступа: https://www.swrit.ru/doc/espd/19.002-80.pdf.

18 Паттерны объектно-ориентированного проектирования / Э. Гамма [и др.]. – СПб. : Питер, 2020. – 448 с.

19 Интернет-ресурс о паттернах проектирования [Электронный ресурс]. – Режим доступа: https://refactoring.guru/ru.

20 Буч, Г. Язык UML. Руководство пользователя / Г. Буч, Дж. Рамбо, И. Якобсон. – М. : ДМК Пресс, 2006. – 496 с.

21 SOLID principles [Электронный ресурс]. – Режим доступа: https://siderite.dev/blog/solid-principles-plus-dry-yagni-kiss.html.

22 Самые важные архитектурные шаблоны [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/alconost/blog/522662.

23 Тестирование программного обеспечения: учеб. пособие / С. С. Куликов [и др.]. – Минск : БГУИР, 2019. – 276 с. : ил.

24 Покрытие кода [Электронный ресурс]. – Режим доступа: https://coderlessons.com/tutorials/kachestvo-programmnogo-obespecheniia/ruchnoe-testirovanie/pokrytie-koda-2.

25 Бахтизин, В. В. Метрология, стандартизация и сертификация в информационных технологиях : учеб. пособие. В 2 ч. Ч. 2 / В. В. Бахтизин, Л. А. Глухова. – Минск : БГУИР, 2016. – С. 141–343.

Св. план 2024, поз. 9

Учебное издание

Тонкович Ирина Николаевна
Шелест Анна Вадимовна

ТЕХНОЛОГИИ ПРОЕКТИРОВАНИЯ СЛОЖНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редакторы *Ю. В. Ляховец, А. Ю. Шурко*
Корректор *Е. Н. Батурчик*
Компьютерная правка, оригинал-макет *А. А. Лущикова*

Подписано в печать 26.02.2025. Формат 60×84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 9,88. Уч.-изд. л. 9,5. Тираж 50 экз. Заказ 42.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014.
Ул. П. Бровки, 6, 220013, г. Минск