

Министерство науки и высшего образования Российской Федерации  
ФГБОУ ВО «Кубанский государственный технологический университет»

Кафедра информационных систем и программирования

## **ПРОЕКТИРОВАНИЕ ПРОГРАММНЫХ КОМПЛЕКСОВ**

Методические указания по выполнению курсового проекта  
для студентов всех форм обучения  
направления подготовки 09.03.04 Программная инженерия

Краснодар  
2019

Составитель: канд. техн. наук, доц. М.В. Янаева

УДК 681.31(031)

**Проектирование программных комплексов:** методические указания по выполнению курсового проекта для студентов всех форм обучения направления подготовки 09.03.04 Программная инженерия / Сост.: М.В. Янаева; Кубан. гос. технол. ун-т. Каф. информационных систем и программирования. – Краснодар: 2019. – 32 с.

Содержат общие положения и порядок выполнения курсового проекта, описание структуры пояснительной записки, а также примеры построения основных типов диаграмм из раздела проектирования программного продукта.

Рецензенты: Руководитель отдела телекоммуникаций Краснодарского регионального информационного центра сети «Консультант Плюс», канд.техн.наук. Н.Ф. Григорьев;  
д-р. техн. наук, профессор каф. ИСП КубГТУ  
В.Н. Марков

## Содержание

1 Нормативные ссылки .....	4
2 Требования к курсовому проекту .....	4
2.1 Общие положения .....	4
2.2 Порядок выполнения курсового проекта.....	5
2.3 Средства программного обеспечения .....	6
2.4 Структура пояснительной записки.....	6
3 Примеры построения диаграмм.....	12
4 Варианты задания на курсовой проект .....	26
Список рекомендуемой литературы.....	28
Приложение А Форма титульного листа курсового проекта .....	29
Приложение Б Форма задания на курсовой проект.....	30
Приложение В Пример оформления реферата.....	31

## **1 Нормативные ссылки**

В данных методических указаниях использованы ссылки на следующие нормативные документы:

- ГОСТ Р 7.0.5-2008 СИБИД. Библиографическая ссылка. Общие требования и правила составления;
- ГОСТ Р 1.5-2004. Стандарты национальные РФ. Правила построения, изложения, оформления и обозначения;
- ГОСТ 2.301-68 ЕСКД. Форматы;
- ГОСТ 7.82-2001 СИБИД. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления;
- ГОСТ 7.12-93 СИБИД. Библиографическая запись. Сокращения слов на русском языке. Общие требования и правила;
- ГОСТ 7.9-95 СИБИД. Реферат и аннотация. Общие требования;
- ГОСТ 34.602-89. Техническое задание на создание автоматизированной системы

## **2 Требования к курсовому проекту**

### **2.1 Общие положения**

Тематика курсовых проектов, как правило, определяется с учетом будущих тем выпускных квалификационных работ студентов. Желательно, чтобы в постановке задачи и проработке проектных решений помимо студента, выполняющего работу на данную тему, принял участие руководитель выпускной квалификационной работы. Тема курсового проекта должна предусматривать разработку новых программных приложений информационных систем или совершенствование существующего программного обеспечения, необходимость которого обоснована требованиями автоматизации рутинных действий пользователей и потребностью более качественного использования информационных ресурсов.

Объектом разработки могут быть информационно-моделирующие системы технологических процессов и объектов, информационные системы различных отраслей, в том числе основанные на базах данных. Содержанием курсового проекта является весь комплекс задач, достаточный для создания программного обеспечения современной информационной системы, в частности разделы по составлению технического задания на разработку системы, описанию проектных решений, программной реализации и функциональных характеристик

созданного программного обеспечения. Формулировка темы курсового проекта осуществляется преподавателем дисциплины после предварительных консультаций со студентом и его научным руководителем. Руководителем курсового проекта совместно со студентом оформляется задание на курсовой проект и утверждается календарный план его выполнения.

Курсовой проект состоит из пояснительной записки и разработанного программного обеспечения.

Пояснительная записка представляет собой совокупность всех текстовых документов (таблицы, спецификации, листинг программного кода, рисунки и др.) и должна в краткой и четкой форме раскрывать содержание работы. В пояснительную записку к курсовому проекту обязательно необходимо включать проектную часть, выполненную с использованием средств UML. Общий объем пояснительной записки не должен превышать 40 листов, в том числе введение – не более 2 листов. К пояснительной записке прилагается носитель с отлаженными приложениями и иной необходимой информацией. Записка иллюстрируется схемами, копиями экрана, выходными документами, листингом программного кода приложения.

Программное обеспечение отражает в компьютерной форме результаты выполнения курсового проекта и включает разработанную прикладную программу и файл справочной помощи.

## **2.2 Порядок выполнения курсового проекта**

В процессе выполнения курсового проекта студенты должны:

1. Разработать развернутое техническое задание на программный продукт.
2. Выполнить анализ задания, выбрать технологию проектирования и разработать проект программного продукта.
3. Выбрать структуры данных для реализации предметной области программного продукта.
4. Выбрать язык и среду программирования, наиболее удовлетворяющие проведенным разработкам.
5. Разработать алгоритмы и реализовать их в выбранной среде разработки.
6. Выполнить тестирование и отладку.
7. Разработать необходимую документацию, указанную в техническом задании.

## **2.3 Средства программного обеспечения**

Рекомендуемые операционные системы: операционные системы семейства Windows, Linux.

Рекомендуемые среды проектирования: Star UML.

Рекомендуемые СУБД: MySQL, MS SQL Server.

Рекомендуемые инструментальные среды создания приложений: Microsoft Visual Studio.

## **2.4 Структура пояснительной записки**

Пояснительная записка на листах формата А4 по ГОСТ 2.301 к курсовому проекту должна содержать следующие обязательные разделы:

- титульный лист (приложение А);
- задание на курсовой проект (приложение Б);
- реферат (приложение В), включающий количество страниц ПЗ, согласно ГОСТ 7.9 количество таблиц, рисунков, программ приложений, ключевые слова (прописными буквами), краткую характеристику и результаты работы;
- содержание;
- введение;
- основную часть;
- заключение;
- список использованных источников;
- приложения (листинг структуры файлов БД, листинг реализованных запросов, листинг компонент реализованных форм, отчетов, меню)

Выполнение пояснительной записки рекомендуется делать на компьютере с последующей распечаткой на принтере. Рекомендуемый объем записки составляет 40–60 страниц. Текст пояснительной записки должен быть кратким, четким, логически последовательным, полностью отвечать всем пунктам задания на курсовой проект, не допускать различных толкований. Листы пояснительной записки должны быть сшиты, пронумерованы и представлены в твердом переплете. Нумерация страниц выполняется арабскими цифрами в правом нижнем углу.

Структурные части пояснительной записки (содержание, реферат, введение, заключение, список использованных источников, приложение) должны начинаться с нового листа. Эти части документа не нумеруются. Заголовки пишут прописными буквами. Допускается написание жирным шрифтом.

Содержание включает наименования всех разделов, подразделов, пунктов, структурных частей с указанием номера страницы начала каждой из перечисленных составных частей курсового проекта.

Реферат представляет собой краткое и точное изложение содержания работы с применением стандартной терминологии. В реферате должны быть:

- тема курсовой работы;
- сведения об объеме текстовой и иллюстрированной частей, количестве страниц, рисунков, таблиц, библиографических названий, приложений;
- перечень ключевых слов и словосочетаний (не более 20), отражающих суть выполненной работы;
- текст реферата.

В тексте реферата указывают:

- цель работы;
- перечень основных проектных решений с краткими комментариями, характеризующими их новизну и эффективность;
- области возможного применения результатов работы.

Оптимальный объем реферата – 1 200 знаков (не более одной страницы).

Введение обосновывает необходимость и направление выполняемой работы. Оно должно содержать:

- оценку современного состояния решаемой в курсовой работе проблемы;
- краткую характеристику предметной области;
- исходные данные для разработки;
- обоснование необходимости выполнения работы;
- указания на актуальность и новизну темы;
- перечень ожидаемых результатов.

Объем введения должен быть не более 2–3 страниц.

Основная часть курсовой работы состоит из ряда разделов и отражает:

- назначение, цели и задачи создания программного обеспечения;
- архитектуру построения информационной системы;
- разработку функциональной модели;
- блок-схему алгоритма обработки данных;
- последовательность разработки прикладной программы;
- функциональные возможности программного приложения.

Рекомендуемый объем основной части 40–50 страниц.

Заключение должно содержать:

- краткие выводы о результатах выполненной работы;
- выводы и предложения по использованию результатов работы на предприятиях или организациях;

– указания на перспективы развития работы.

Рекомендуемый объем заключения 1 страница.

Список использованных источников должен содержать перечень литературных источников, с которыми работал студент в процессе выполнения курсового проекта. Его составляют в соответствии с установленными правилами библиографического описания по ГОСТ Р 7.0.5, ГОСТ 7.82.

Приложение состоит из вспомогательного материала, на который в текстовой части курсового проекта имеются ссылки. В приложении приводят различные схемы, акты, таблицы справочных данных, листинг программ и др. Рекомендуемый объем приложений части 5–10 страниц.

Рекомендуемая структура основной части:

- 1 Нормативные ссылки
- 2 Описание предметной области
  - 2.1 Общее описание предметной области
  - 2.2 Назначение, цели и задачи создания ПО
  - 2.3 Требования к структуре ПО и базе данных
  - 2.4 Описание входных документов и сообщений
  - 2.5 Описание выходных документов и сообщений
  - 2.6 Список ограничений
- 3 Разработка технического задания ПО
  - 3.1 Общие сведения
  - 3.2 Назначение и цели создания (развития) системы
  - 3.3 Характеристика объектов автоматизации
  - 3.4 Требования к системе
  - 3.5 Состав и содержание работ по созданию системы
  - 3.6 Порядок контроля и приемки системы
  - 3.7 Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие
  - 3.8 Требования к документированию
  - 3.9 Источники разработки
- 4 Проектирование структуры базы данных (в случае наличия в проекте)
  - 4.1 Описание сущностей
  - 4.2 Построение концептуальной схемы базы данных
- 5 Реализация базы данных средствами выбранной СУБД (в случае наличия в проекте)
  - 5.1 Выбор СУБД
  - 5.2 Проектирование таблиц
  - 5.3 Построение ER-диаграммы
- 6 Проектирование структуры приложения
  - 6.1 Построение структурной схемы ПО
  - 6.2 Построение архитектуры информационной системы
  - 6.3 Разработка функциональной модели



- 6.4 Диаграмма компонентов
- 6.5 Диаграммы кооперации
- 6.6 Диаграмма размещения
- 6.7 Диаграмма состояний
- 6.8 Диаграммы потоков данных
- 6.9 Диаграмма классов
- 6.10 Диаграмма последовательности
- 6.11 Диаграмма прецедентов
- 7 Описание алгоритмов обработки данных
  - 7.1 Описание алгоритмов обработки данных
  - 7.2 Блок-схемы алгоритмов обработки данных
- 8 Описание реализации программного продукта
  - 8.1 Выбор среды программирования
  - 8.2 Последовательность разработки прикладной программы
  - 8.3 Схема меню приложения
  - 8.4 Описание окон приложения
  - 8.5 Функциональные возможности программного приложения
  - 8.6 Функции и процедуры обработки данных
  - 8.7 Формирование отчетной документации
- 9 Тестирование программного продукта и отладка
- 10 Проектирование технической и справочной информации к ПО

При разработке технического задания следует руководствоваться ГОСТ 34.602-89 Техническое задание на создание автоматизированной системы.

Структурная схема – схема, отражающая состав и взаимодействие частей разрабатываемого продукта. При объектной декомпозиции такими частями являются объекты, при структурной декомпозиции – подпрограммы.

Для тем, связанных с нечисловой обработкой данных, этот раздел должен содержать информационную модель системы, которая может быть представлена функциональной схемой. Функциональная схема – схема взаимодействия частей системы с описанием информационных потоков, состава данных в потоках и указанием используемых файлов и устройств.

Большое значение при разработке структуры программного продукта имеет выбор алгоритмов предметной области и структур данных. Для заданий, при выполнении которых используются сложные математические методы, и заданий, при реализации которых студентом предлагались собственные оригинальные алгоритмы, обязательным является присутствие в записке обоснования выбора метода (по вычислительной сложности или другим соображениям). В этом же разделе приводятся и сами алгоритмы (в виде схем или псевдокодов) и пояснения к ним. Выбор структур данных осуществляется на основе анализа основных процессов

обработки информации (статические или динамические, массивы или другие структуры). При необходимости создаются новые структуры данных или модифицируются уже известные. Обычно при выборе структур учитываются следующие параметры: объем и типы данных, а также основные операции над данными (хранение, поиск, сортировка) и частота обращения к ним в процессе выполнения программы. Если возможны варианты, то производится их оценка по объему требуемой памяти и вычислительной сложности выполнения основных операций. Описание реализации программного продукта для программы, при разработке которой использовалась объектно-ориентированная технология, обязательно должна быть разработана диаграмма классов. Для каждого класса нужно указать необходимые атрибуты и операции, соответственно обосновывая их назначение и функции.

Описание каждого алгоритма должно включать:

- функциональное назначение алгоритма;
- входные и выходные данные (результаты выполнения);
- список формальных параметров и их назначение;
- пример вызова модуля или подпрограммы;
- используемые технические средства;
- таблицу переменных алгоритма;
- рисунок со схемой алгоритма;
- описание процесса обработки данных в соответствии со схемой.

При описании процесса обработки данных в соответствии со схемой алгоритма необходимо пояснить все циклы, каждую альтернативу ветвления, принятое решение по результатам анализа альтернатив и последующие действия. Тексты описания алгоритмов должны быть структурными, предложения короткими. Описание алгоритма должно отражать суть процесса обработки.

Выбор стратегии тестирования и отладка программного средства должны содержать обоснование той или иной стратегии тестирования программного средства, тестовые наборы данных (тесты) по всем частям программного продукта как с использованием правильных входных данных, так и входных данных, не соответствующих принятым ограничениям, а также иллюстрироваться экранными распечатками и комментариями процесса отладки. Отладка включает в себя поиск ошибки в тексте программного модуля (локализация ошибки) и исправление обнаруженной ошибки. Описывается проведенный анализ ошибок, выявленных в ходе написания, трансляции, тестирования и отладки программного средства. Приводятся распечатки экранных форм, отражающие полученные результаты решения поставленной задачи. Делается вывод о соответствии числовых значений результатов, их

точности, форм выдачи и т.д. требованиям поставленной задачи. Можно привести данные статистической отчетности – количество допущенных ошибок (по видам), трудозатраты на разных этапах разработки и отладки модулей программного средства, расход вычислительных ресурсов на отдельных этапах выполнения задания. Описываются обнаруженные некорректные или нерациональные приёмы программирования и программные конструкции, ошибки в программе, ошибки в алгоритме и постановке задачи.

Прикладная программа должна быть разработана с использованием современной среды программирования. В исходном коде программы обязательно должны присутствовать комментарии, поясняющие функциональное назначение каждого программного модуля (процедуры) и объявленные переменные. Сдача прикладной программы осуществляется в двух вариантах: исходных кодах и в виде дистрибутива, подготовленного с использованием современных инсталляционных пакетов.

Файл справочной помощи должен быть представлен в одном из популярных форматов файла-справки (chm, hlp или др.), а также в виде файлов исходного кода среды разработки, например программы Help&Manual.

## **2.5 Защита курсового проекта**

Защита курсового проекта проводится по утвержденному графику, создается специальная комиссия из числа преподавателей кафедры в составе не менее трех человек. Перенос сроков защиты допускается только в особых случаях, возникших не по вине студента. Для доклада на защите курсового проекта студенту отводится 5–10 минут. Доклад следует начинать с названия темы, цели и задач курсового проекта. Далее в краткой форме излагаются суть проектных решений, архитектуры созданной информационной системы, обоснование среды программирования, особенности программной реализации системы и основные функциональные характеристики программного приложения. Доклад сопровождается демонстрацией основных положений в форме мультимедийной компьютерной презентации и заканчивается перечислением основных результатов работы и выводами, сделанными в ходе ее выполнения. После доклада студенту задаются вопросы членами комиссии и всеми присутствующими, на которые он обязан дать полные и аргументированные ответы. Результаты защиты курсового проекта определяются оценками «отлично», «хорошо», «удовлетворительно», «неудовлетворительно» и объявляются в тот же день после оформления зачетных ведомостей в установленном порядке. Курсовые проекты после защиты хранятся в порядке, установленном приказом ректора университета.

### 3 Примеры построения диаграмм

#### Диаграмма прецедентов

Прецеденты – это последовательные истории об использовании системы, которые широко применяются для осмысления и формулировки требований. Стоит отметить, что основной задачей проектировщика является не построение диаграммы, а составление текста для описания сути работы каждого «сценария». Модель диаграммы прецедентов оперирует абстракциями верхнего уровня. В данной модели нет строгого выявления абстракций функционала системы, на данном этапе описывается самый базовый функционал системы. «Системными» исполнителями могут быть как отдельно разработанные классы или пакеты, так и составные из нескольких элементов системной архитектуры. Диаграмма ориентирована на пользователей системы, на данном этапе проектирования выявляются ключевые особенности взаимодействия пользователей с системой, а также описываются базовые требования к реализации системного функционала.

*Исполнителем* (actor) называется сущность, обладающая поведением, например, человека (идентифицируемого по роли, к примеру, кассира), компьютерную систему или организацию.

*Сценарий* (scenario) – это специальная последовательность действий или взаимодействий между исполнителем и системой. Его иногда также называют *экземпляром прецедента* (use case instance). Это один конкретный пример сценария использования системы либо один проход прецедента, например, сценарий успешной регистрации на ресурсе при помощи аккаунтов социальных сетей.

Стоит отметить, каким образом строятся зависимости между сценариями и исполнителями. Обычный сценарий связывается с сущностью исполнителя при помощи обычной прямой линии без стрелочки. Расширение сценария делится на *включаемые прецеденты* (include), *расширяемые* (extend), а также *генерализуемые*.

*Включаемые прецеденты* – это прецеденты, которые наступают одновременно с возникновением определённого сценария. Например, после создания обязательно произойдёт событие по перенаправлению пользователя на страницу комнаты.

*Расширяемые прецеденты* – это прецеденты, которые могут осуществляться при работе определённого сценария, но не являются обязательными к исполнению. Для примера можно привести сценарий «редактирования плейлиста» при выполнении сценария «управления комнатой». Администратор может изменить очередь треков, но это не всегда происходит во время его управления комнатой.

К *генерализуемым прецедентам* относятся те прецеденты, которые иерархически выходят из другого прецедента.

Роли актёров расположены по бокам объектной модели. Исполнители-пользователи, проводящие основное взаимодействие с системой, расположены слева от модели системы. Исполнители, являющиеся системными сущностями программного обеспечения, расположены справа от объектной модели.

*Пользователь* – ключевой исполнитель данной системы. Абсолютно все исполнители всех ролей пользователей (пользователь, администратор и гость) имеют возможность входа в систему по заранее сохранённым персональным данным или как анонимный гость. Каждый пользователь имеет возможность производить авторизацию в систему для осуществления всех остальных действий, поэтому данный прецедент выведен на верхний уровень модели прецедентов. После авторизации пользователь имеет возможность авторизации в привилегированные комнаты, в случае если пользователь обладает данным правом. Находясь в комнате системы, все пользователи имеют доступ к прослушиванию аудио-контента, а привилегированные пользователи имеют право добавлять собственные аудиодорожки в плейлист комнаты.

*Администратор* – пользователь, обладающий правом управления активностью внутри выделенной комнаты. Управление комнатой заключается в возможности добавления или удаления треков из очереди плейлиста комнаты, а также управлением участниками:

- выдача прав пользователям комнаты на добавление треков;
- удаление участников комнаты;
- добавление пользователей в «чёрный список» комнаты.

*Гость* – неавторизованный пользователь системы. Данный исполнитель может производить вход в открытые комнаты, но не имеет права вести любую деятельность в комнате кроме прослушивания аудио-контента. Любой незарегистрированный в системе пользователь имеет возможность произвести регистрацию в системе удобным ему способом – через персональный адрес электронной почты или используя данные аккаунтов социальных сетей.

*Система авторизации* – системный исполнитель, в обязанности которого входит проверка корректности вводимых данных пользователей при авторизации в системе, доступ по паролю в закрытые (привилегированные); выдача анонимным пользователям временных данных для идентификации, а также осуществление регистрации новых пользователей в системе. Для регистрации новых пользователей данный исполнитель производит новые записи в базу данных системы, а также имеет доступ к системным интерфейсам социальных сетей для возможности регистрации новых пользователей, используя персональные данные социальных сетей.

*Менеджер комнат* – к задачам данного системного исполнителя относится реализация программного функционала для ведения администраторами функций управления пользователями комнат.

*Система управления плейлистами* – системный исполнитель, реализующий систему очередей треков в каждой комнате. Также в задачи данного исполнителя входит реализация программного функционала для предоставления возможности пользователям добавлять новые треки в очередь, а администраторам предоставить возможность удалять треки из очереди.

Диаграмма описания прецедентов ставит задачу систематизации требований к первоначальному функционалу системы, а также выделения ролей данной системы, поэтому данную модель строят на первых стадиях проектирования. Для удобства проектировщика текстовое описание прецедентов носит свободный характер, но при этом проектировщик обязуется донести до разработчиков начальное понимание функционирования системы. Главной задачей диаграммы прецедентов являются понятность и читаемость. Не стоит выделять небольшую задачу или функционал системы в отдельный прецедент, в таком случае модель приобретёт избыточность и потеряет свойство читаемости.

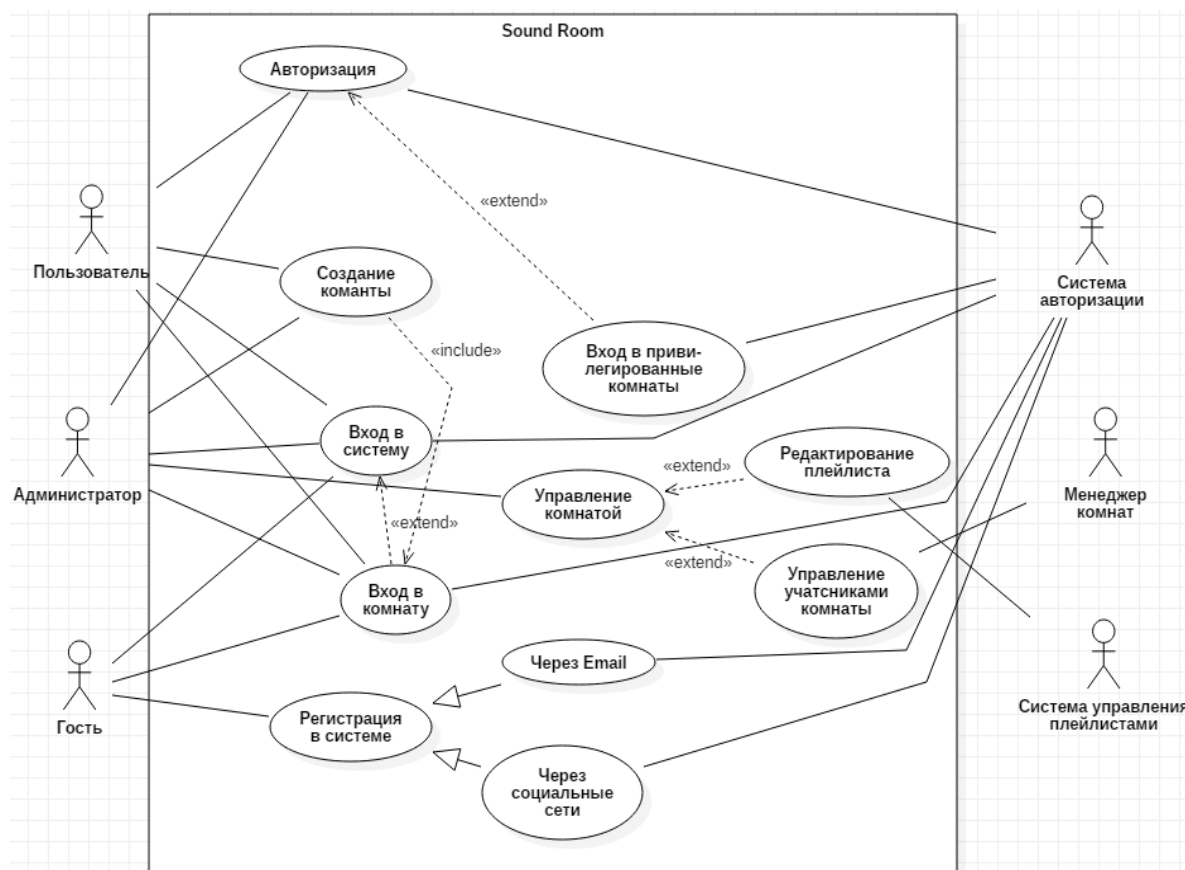


Рис. 1. Диаграмма прецедентов

## Модель предметной области

Модель предметной области – это самая важная модель объектно-ориентированного анализа. Она отображает основные классы понятий (концептуальные классы) предметной области.

Каждой итерации соответствует своя модель предметной области, отражающая реализуемые на данном этапе сценарии прецедентов. Таким образом, модель предметной области эволюционирует в процессе разработки системы. Модель предметной области связана с моделью проектирования, особенно программными объектами, относящимися к уровню предметной области.

На рисунке 2 представлен фрагмент модели приложения Sound Room в виде диаграммы классов. Из рисунка ясно, что с точки зрения предметной области *концептуальными классами* (conceptual class) являются Home (Главная страница), Services (сервисы), User (пользователь), Lobby (страница комнаты размещения треков), Queue (очередь треков) и Track (трек). Как видно из диаграммы, эти понятия связаны между собой, и понятию Lobby соответствуют определения названия комнаты и типа комнаты.

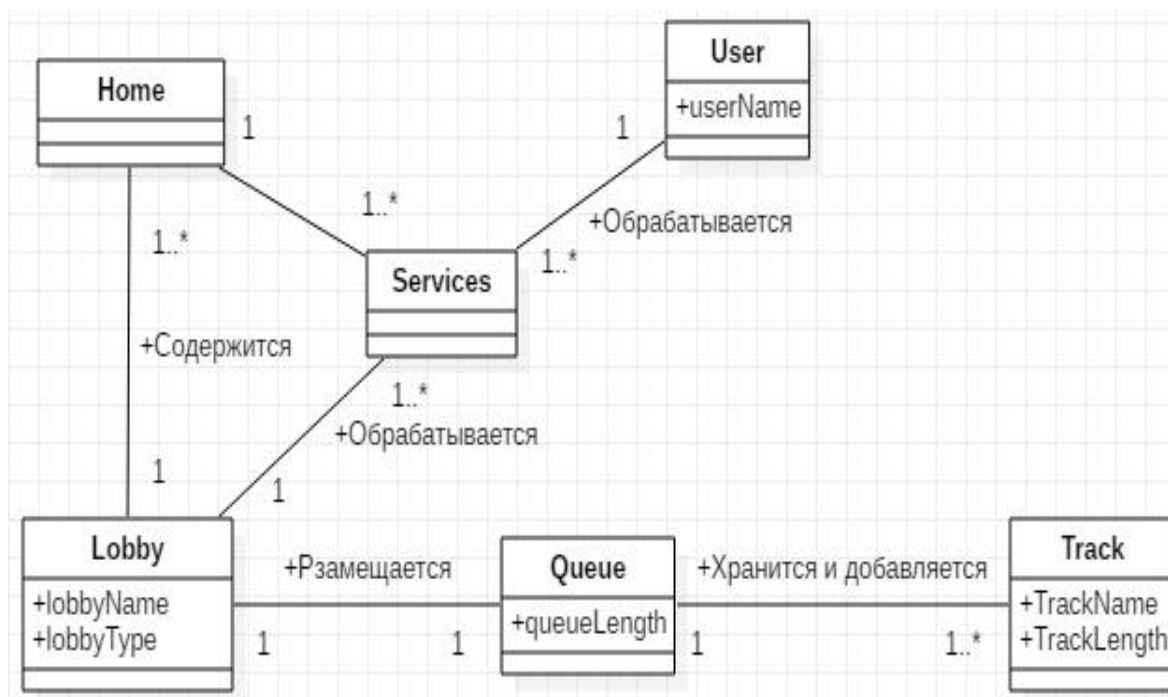


Рис. 2. Фрагмент модели предметной области – визуальный словарь

Диаграмма классов в обозначении UML обеспечивают концептуальную перспективу модели.

Идентификация набора концептуальных классов – основная задача объектно-ориентированного анализа. На начальных этапах построение модели предметной области у разработчика может занять всего несколько часов, но на последующих этапах, когда требования к системе определяются более чётко, уточнение модели предметной области потребует значительно больше времени.

При построении диаграммы модели предметной области избегайте попыток построения полной модели на начальных этапах разработки. Такой стиль присущ каскадному процессу разработки, когда модель предметной области строилась на этапе анализа без учёта обратной связи.

### **Системная диаграмма последовательностей**

Системная диаграмма последовательностей – это схема, которая для определённого сценария прецедента показывает генерируемые внешними исполнителями события, их порядок, а также события, генерируемые внутри самой системы. Диаграмма данного типа приведена на рисунке 3. Данная диаграмма демонстрирует последовательность процессов, протекающих между оператором и системой синхронизации данных системы торговой розничной сети.

На диаграмме (рисунок 3) представлена схема, которая показывает генерируемые оператором события для нескольких сценариев прецедента оператора. Все события имеют строгую последовательность. Стоит выделить фрейм *повтор*, обозначающий цикличность действий, из данной диаграммы. Данная диаграмма демонстрирует поведение системы по принципу «чёрного ящика». Есть описание того, какие действия выполняет система, без определения механизма их реализации. На каждый запрос оператора следует системное событие.

### **Диаграмма видов деятельности**

Диаграммы видов деятельности отображают последовательные и параллельные процессы. Они полезны для моделирования бизнес-процессов, последовательностей выполнения задач, потоков данных и сложных алгоритмов.

На рисунке 4 изображена диаграмма видов деятельности системы синхронизации, а также информационных систем, связанных с системой синхронизации.

На диаграмме представлен пример взаимодействия системы синхронизации между двумя ИС компании. В действительности параллельные процессы протекают между несколькими подразделениями компании.



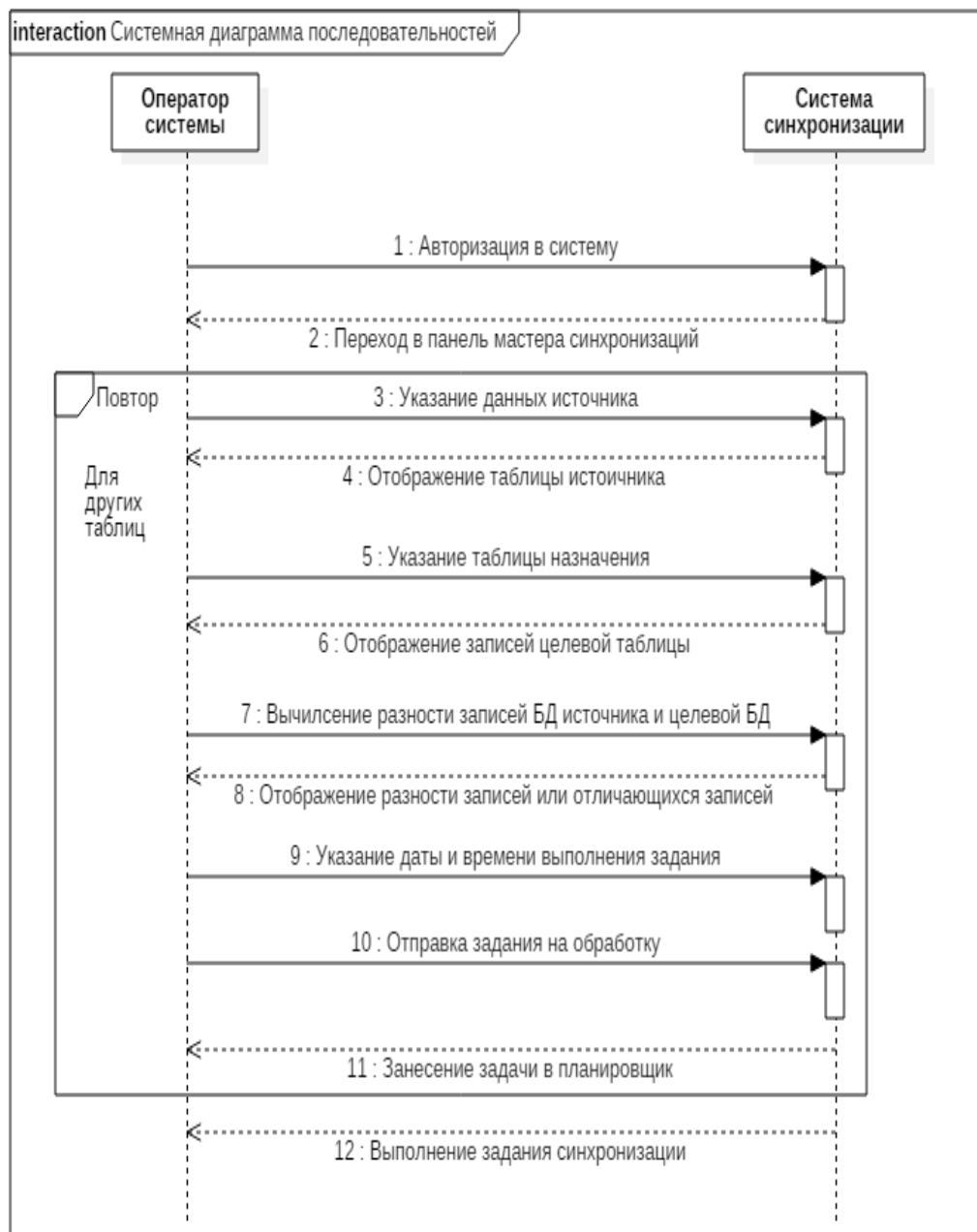


Рис. 3. Диаграмма последовательностей

Таким образом, данная диаграмма будет справедлива для обмена данными между подразделением распределительного центра и главного офиса, или между одним из магазинов и главным офисом.

Точки ветвления сообщают о возможном совершении действия одной из веток сценария или одновременном параллельном выполнении, которые формируют новый объект для системы. На диаграмме мы можем наблюдать такой сценарий у двух информационных систем, формирующих новые данные независимо от работы системы синхронизации. Эти данные оператор системы синхронизации может наблюдать после совершения всех необходимых действий для запуска и авторизации в систему. На моменте действия «сравнения данных» оператор через внутреннюю логику системы отправляет запросы к БД информационных систем на получение

актуальных данных системы источника и данных целевой системы на текущий момент времени.

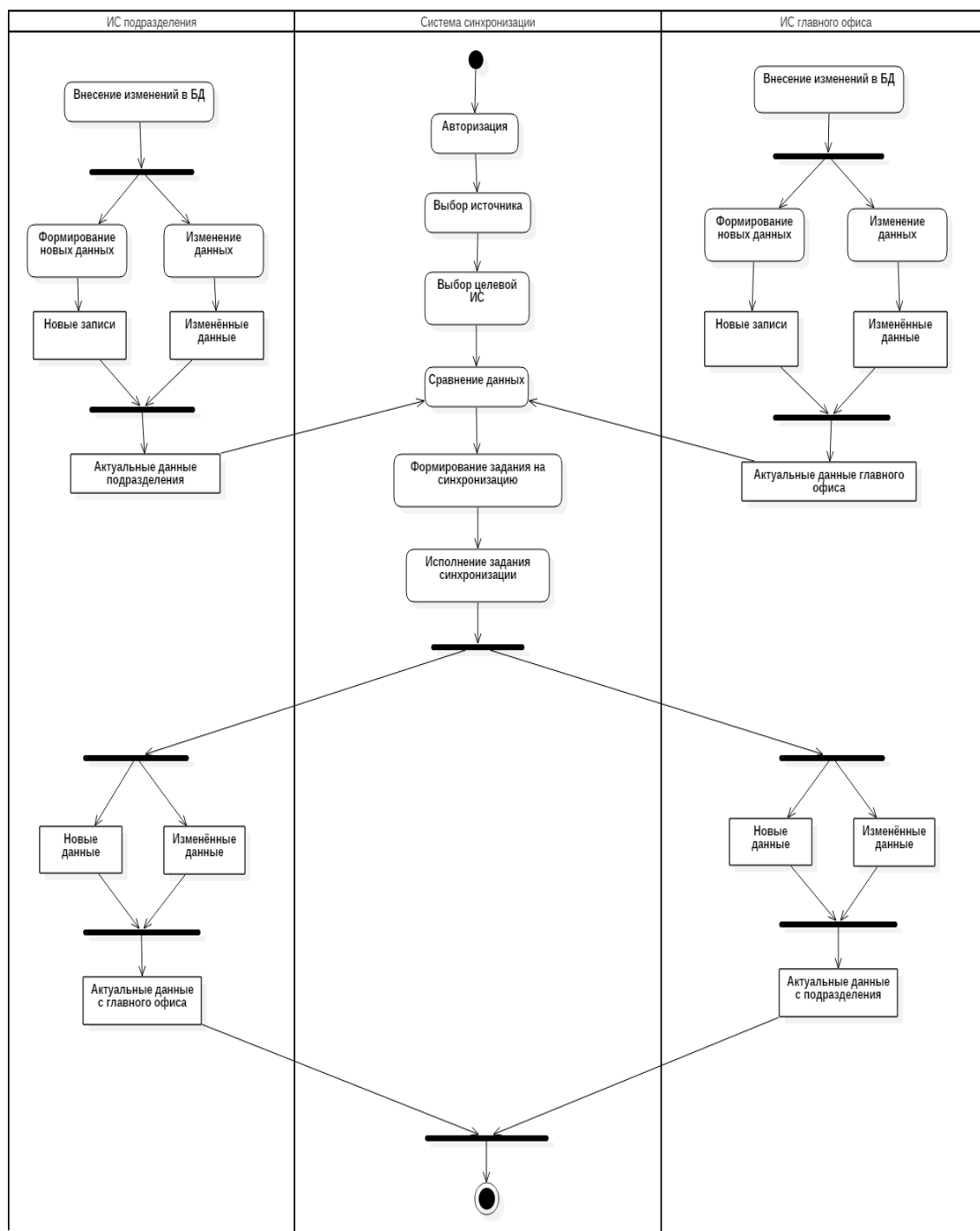


Рис. 4. Диаграмма видов деятельности

В назначенное время после формирования задания происходит отправка новых или изменённых данных в необходимый ресурс. Оператор системы, составивший заявку на актуализацию данных, формирует ответ

об обновлении данных. После получения актуальных данных подразделения продолжают выполнять свои задачи с использованием обновлённых данных. Например, магазин регистрирует новую единицу товара, данные о котором прибыли из главного офиса. После поставки новой единицы товара, магазин накапливает данные о совершённых покупках с учётом нового товара. В назначенное время данные о совершённых покупках в магазине должны быть отправлены в главный офис компании, которые затем распределяются между отделами бухгалтерии и аналитики.

### **Диаграмма компонентов**

При переходе от анализа к проектированию необходимой системы нужно посмотреть на систему в целом. На этом уровне абстракции проектирование типичной объектно-ориентированной системы основывается на нескольких архитектурных уровнях: уровне интерфейса пользователя, логике приложения (или предметной области) и прочих уровнях системы.

Стоит отметить, что диаграмма компонентов может быть представлена на разных уровнях абстракции. Это может быть уровень абстракции всех функциональных компонентов приложений: серверная логика, компонент графического интерфейса, а также сервисы промежуточного ПО.

В данном разделе проектирования будет составлена начальная диаграмма графических компонентов приложения Sound Room. Как указывалось выше, разработка графического интерфейса происходит с помощью библиотеки React вместе с библиотекой Redux. Данный подход позволяет заранее составить компонентную модель проекта, прежде чем приступать к непосредственному написанию кода.

Диаграмма компонентов позволяет декомпонизировать определённый уровень абстракции системы. В данном случае это уровень графического интерфейса приложения, проектируемого по принципу технологии Redux. Рассмотрим подробнее каждый из компонентов.

*app.js* – элемент системы, импортирующий все функциональные библиотеки, а также графические и функциональные компоненты.

*UI* – компонент, содержащий все графические элементы приложения.

*Actions* – компонент, содержащий все функциональные элементы взаимодействия.

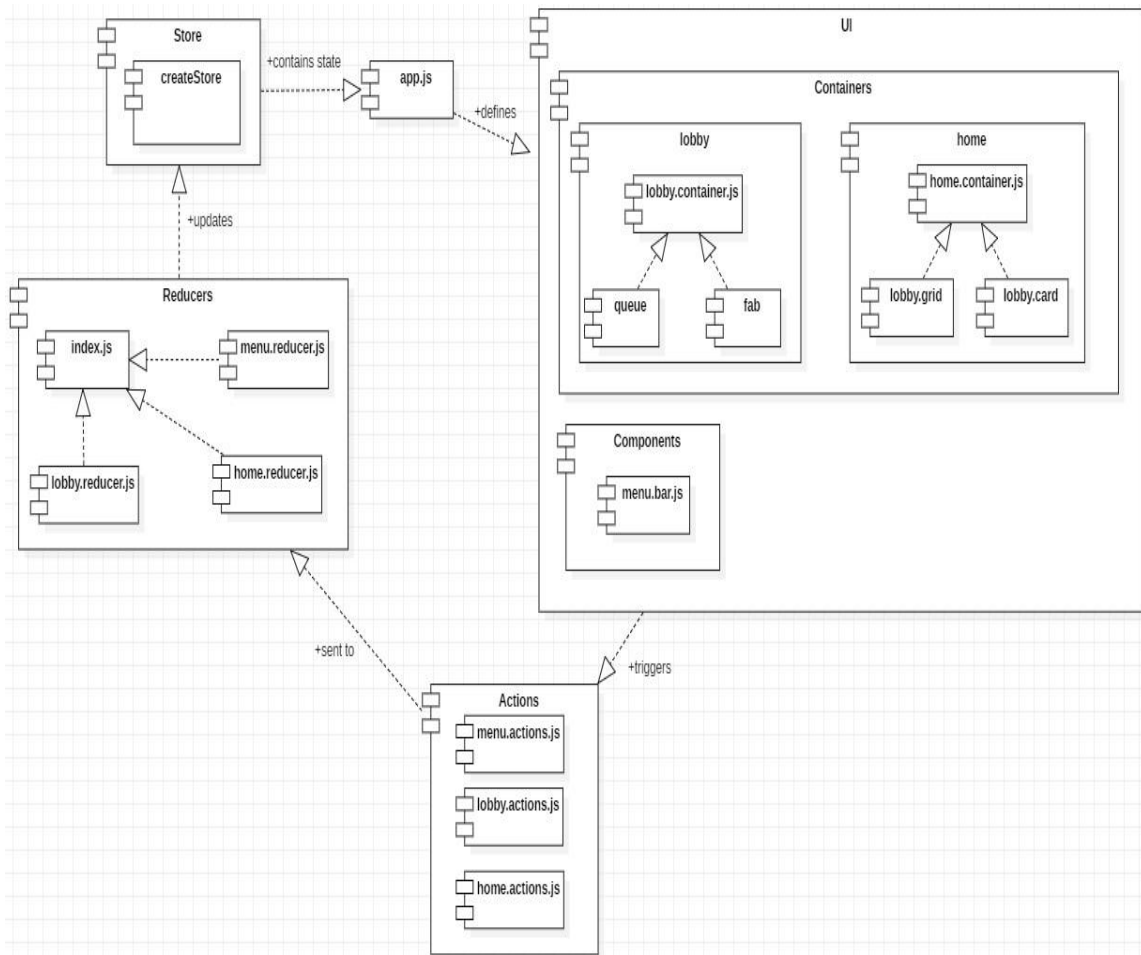


Рис. 5. Диаграмма компонентов

*Reducers* – компонент, содержащий элементы, отслеживающие события, передаваемые функциональными компонентами.

*Store* – компонент, импортирующий компоненты *Reducers* для сохранения изменённого состояния.

При построении диаграммы компонент есть возможность скрыть внутреннюю реализацию компонента или отобразить её на диаграмме. В данном случае отличным примером служит компонент *UI*, который содержит реализации компонентов контейнеров *lobby* и *home*. Компоненты, объединённые в компонент *Components*, являются независимыми, но относятся к самостоятельным объектам графического интерфейса, обладающим собственным состоянием, и способны его изменять. Поэтому данные объекты можно отнести к компоненту *UI*.

Опишем, каким образом взаимодействуют компоненты в системе.

Изменения состояний элементов происходят от взаимодействия пользователя с элементами приложения. Действие, переданное компоненту, активирует действие из компонента *Actions*, который передаёт данное изменение компоненту *Reducer*. *Reducer* передаёт

значение переданного состояния и отправляет данное значение компоненту Store. Компонент Store сохраняет текущее состояние компонента и передаёт обновлённое состояние `app.js`, которое рендерит компонент с обновлённым состоянием.

Данная схема отлично иллюстрирует однонаправленный поток данных, протекающий в React приложении.

### Диаграмма классов

На данном примере представлена диаграмма классов элемента программы торговли.

Как видно из рисунка, диаграмма классов демонстрирует взаимодействие между 4-мя классами:

- *Customer* – класс клиента;
- *Order* – класс заказа;
- *Stock* – класс хранилища склада;
- *Product* – класс продукта.

Средний прямоугольник у каждого класса содержит атрибуты, относящиеся к конкретному классу. «+» означает, что атрибут является публичным, «-» – приватным.

Нижний прямоугольник содержит методы, относящиеся к данному классу. Стоит отметить, если класс имеет get-set методы для доступа к определённому атрибуту, нет необходимости указывать данные методы в поле методов класса на диаграмме. Указание атрибута подразумевает наличие get-set методов в классе.

Из представленного изображения видно, что кратность связи реализована между классами *Customer* и *Order*. Это означает, что объект *Order* содержит в себе объект *Customer*.

Также на данной диаграмме представлена *композиционная агрегация*, реализованная между классами *Stock*, *Order* и *Product*.

### Диаграмма развёртывания

Диаграммы развёртывания отражают соответствие конкретных программных артефактов (например, выполняемых файлов) вычислительным узлам (выполняющим обработку). Они показывают размещение программных элементов в физической архитектуре системы и взаимодействие (обычно сетевое) между физическими элементами. Показанная на рисунке 7 диаграмма развёртывания позволяет лучше понять физическую архитектуру (или архитектуру развёртывания).

Основным элементом диаграммы развёртывания является узел (node), относящийся к одному из двух типов.

*Узел устройства* (device node) (или просто устройство (node)).

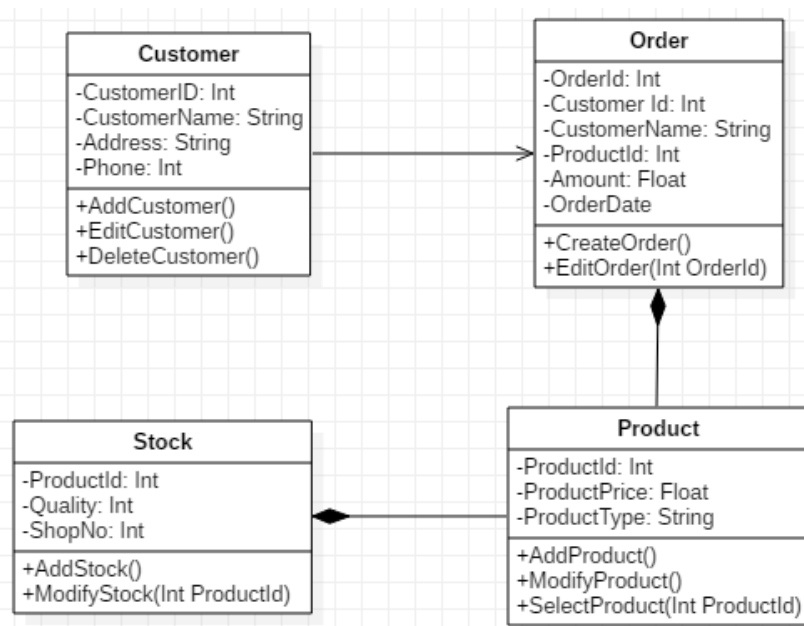


Рис. 6. Диаграмма классов

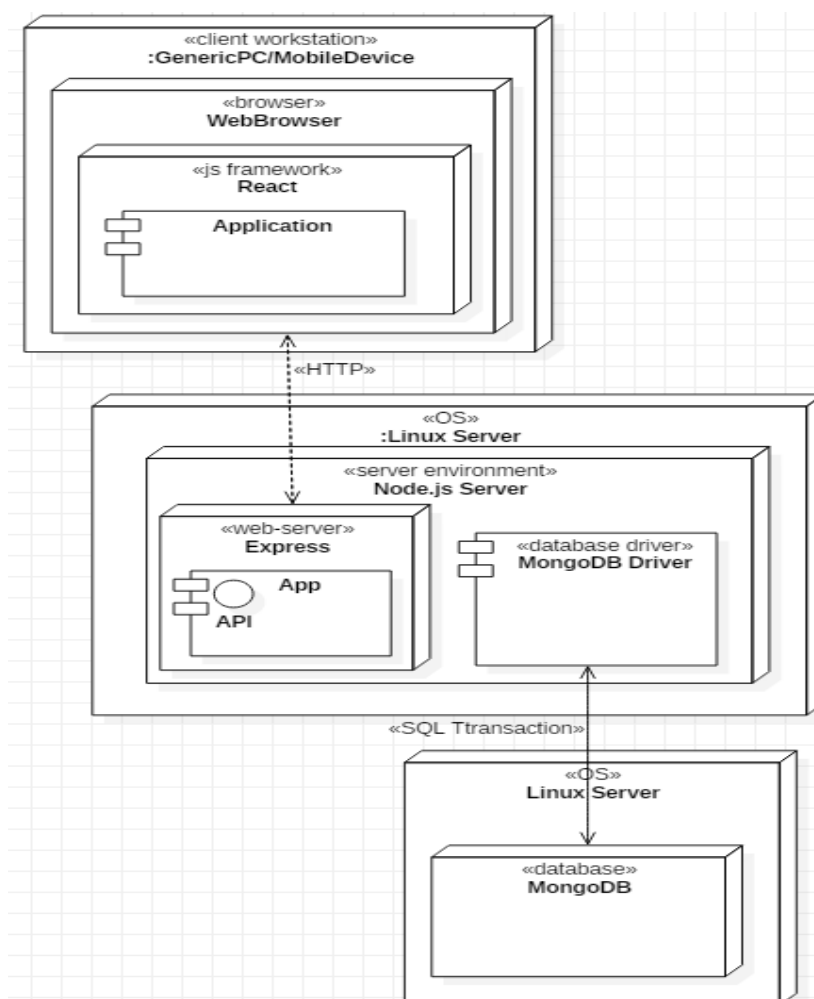


Рис. 7. Диаграмма развёртывания

а) это физический (например, цифровой или электронный) вычислительный ресурс с памятью и процессорным элементом, на котором работает программное обеспечение. В роли устройства может выступать обычный компьютер или смартфон;

б) *исполняющий узел окружения* (execution environment node - EEN) – это программный вычислительный ресурс, работающий в рамках другого узла (например, компьютера) и обеспечивающий выполнение других выполняемых программных элементов. К EEN относятся:

- *операционная система* (operating system) – это программное обеспечение, выполняющее другие прикладные программы;

- *виртуальная машина* (virtual machine), например Java или .NET, отвечающая за выполнение программ;

- *система управления базами данных* (database engine), например PostgreSQL. Она получает и выполняет SQL-запросы, а также хранимые процедуры;

- *Web-браузер*, отвечающий за выполнение сценариев JavaScript, апплетов Java и других активных элементов;

- *механизм управления процессом выполнения задач*;

- *сервлет-контейнер* или *EJB-контейнер*.

Согласно спецификации UML многие типы узлов отображаются с помощью стереотипов, таких как: <<server>>, <<OS>>, <<database>> и <<browser>>.

## **ER-диаграмма**

*Модель* – искусственный объект, представляющий собой отображение (образ) системы и её компонентов.

*Модель данных* (Data Model) – это графическое или текстовое представление анализа, который выявляет данные, необходимые организации с целью достижения ее миссии, функций, целей, стратегий, для управления и оценки деятельности организации. Модель данных выявляет *сущности*, домены (атрибуты) и связи с другими данными, а также предоставляет концептуальное представление данных и связи между данными.

Цель создания модели данных состоит в обеспечении разработчика ИС концептуальной схемой базы данных в форме одной модели или нескольких локальных моделей, которые относительно легко могут быть интегрированы в любую базу данных.

При создании моделей данных используется метод семантического моделирования. Семантическое моделирование основывается на значении структурных компонентов или характеристик данных, что способствует правильности их интерпретации (понимания, разъяснения). В качестве инструмента семантического моделирования используются различные варианты диаграмм сущность-связь (ER – Entity-Relationship) – ERD.

Существуют различные варианты отображения ERD, но все варианты диаграмм сущность-связь исходят из одной идеи – рисунок всегда нагляднее текстового описания. ER-диаграммы используют графическое изображение сущностей предметной области, их свойств (атрибутов) и взаимосвязей между сущностями.

*Сущность* (таблица, отношение) – это представление набора реальных или абстрактных объектов (людей, вещей, мест, событий, идей, комбинаций и т. д.), которые можно выделить в одну группу, потому что они имеют одинаковые характеристики и могут принимать участие в похожих связях. Каждая сущность должна иметь наименование, выраженное существительным в единственном числе. Каждая сущность в модели изображается в виде прямоугольника с наименованием.

Можно сказать, что сущности представляют собой множество реальных или абстрактных вещей (людей, объектов, событий, идей и т. д.), которые имеют общие атрибуты или характеристики.

*Экземпляр сущности* (запись, кортеж) – это конкретный представитель данной сущности.

*Атрибут сущности* (поле, домен) – это именованная характеристика, являющаяся некоторым свойством сущности.







*Связь* – это некоторая ассоциация между двумя сущностями. Одна сущность может быть связана с другой сущностью или сама с собою. Связи позволяют по одной сущности находить другие сущности, связанные с ней.

Связь типа *один-к-одному* означает, что один экземпляр первой сущности связан с одним экземпляром второй сущности. Связь *один-к-одному* чаще всего свидетельствует о том, что на самом деле мы имеем всего одну сущность, неправильно разделенную на две.

Связь типа *один-ко-многим* означает, что один экземпляр первой сущности связан с несколькими экземплярами второй сущности. Это наиболее часто используемый тип связи. Сущность со стороны «один» называется родительской, со стороны «много» – дочерней.

Для обозначения связей на диаграмме используют несколько обозначений типов связи.

Рассмотрим подробнее ER-диаграмму приложения Sound Room

	Один
	Много
	Один и только один
	Ни одного или один
	Один или много
	Ноль или много



Из представленных сущностей основными в данной модели данных являются *Users* и *Rooms*. *Users* содержит данные о пользователях системы. *Rooms* содержит данные о комнатах. Данные сущностей разделяются на подтаблицы для оптимизации хранения данных и безопасности.

*Emails* – содержит данные об электронных адресах пользователей, которые хранятся в отдельной таблице для предоставления базовой безопасности.

*RoomAdmin* – предназначена для хранения записей пользователей администраторов комнат.

*UserLists* – хранит данные о пользователях для каждой комнаты.

*UserRights* – хранит данные о правах пользователей в комнате (только прослушивание или добавление новых треков в очередь).

*UserBannList* – хранит данные о пользователях, добавленных в «чёрный список» определённой комнаты.

*Queues* – хранит данные об очередях треков в комнатах.

*HistoryQueues* – хранит данные истории воспроизведения треков в комнатах.

*Tracks* – хранит данные о записи трека, добавленного в плейлист комнаты.

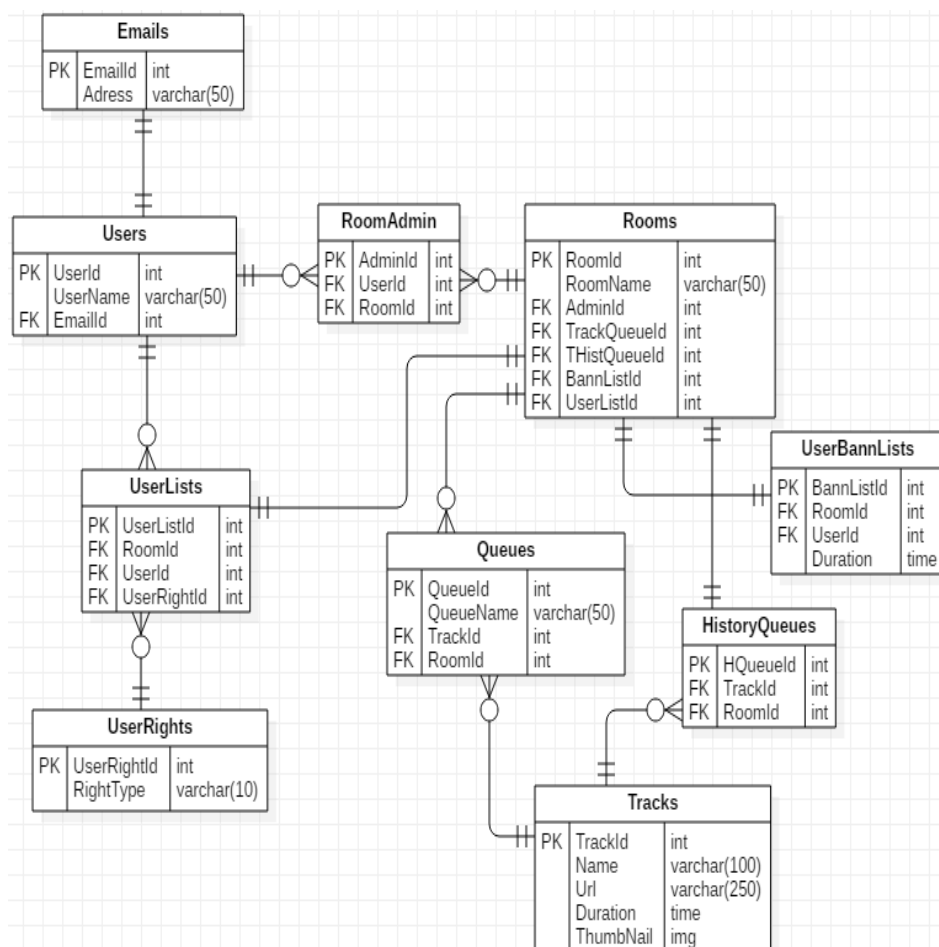


Рис. 8. ER-диаграмма

#### 4 Варианты задания на курсовой проект

1. Моделирование движения транспорта на перекрестке.
2. Информационная система «Автовокзал».
3. Программа для проверки знаний студентов по предмету...
4. Учет успеваемости студентов.
5. Учет аудиторного фонда.
6. Система управления кадрами.
7. Автоматизированный складской учет.
8. Программа для работы пункта обмена валют.
9. Автоматизация учета в торговле.
10. Модель солнечной системы.
11. Система представления табличной информации в графическом виде.
12. Графический редактор «Схемы алгоритмов».
13. Специализированный текстовый редактор.
14. Численные методы линейной алгебры: вычисление определителя, решение системы линейных уравнений, обращение матрицы.
15. Система построения графиков функций.
16. Система «Лотерея».
17. Информационная система «Кинотеатр».
18. Информационная система библиотеки.
19. Информационная система поликлиники.
20. Информационная система деканата.
21. Информационная система «Выставка».
22. Система мгновенного обмена сообщениями.
23. Система учета рабочего времени.
24. Информационная система жилищного агентства.
25. Информационная система технической экспертизы.
26. Система продажи билетов на футбол.
27. Информационная система туристического агентства.
28. Разработка информационной подсистемы по анализу расхода топлива.
29. Разработка информационной подсистемы по анализу финансового состояния предприятия.
30. Разработка информационной подсистемы по учету персонала.
31. Разработка информационной подсистемы по анализу расхода топлива в зависимости от расстояния.
32. Разработка информационной подсистемы по анализу расхода топлива в зависимости от климата.
33. Разработка информационной системы «Оптовая продуктовая база».

34. Разработка информационной подсистемы учета строительно-монтажных работ.

35. Разработка информационной подсистемы по автоматизации процесса подбора запчастей для ремонта автомобилей.

36. Автоматизация работы ресторана.

37. Разработка информационной системы складского учета медицинской аптеки.

38. Разработка информационной системы станции техобслуживания компьютеров.

39. Проектирование информационной подсистемы банковского модуля «Кредитный калькулятор».

40. Разработка информационной системы складского учета ювелирного магазина.

41. Разработка информационной подсистемы по созданию и заполнению календарно-тематического планирования в соответствии с учебным планом.

42. Разработка информационной подсистемы управления заказами клиентов для мебельной фабрики.

43. Разработка информационной подсистемы приема заказов на подключение цифрового телевидения.

44. Разработка информационной подсистемы по оперативно-диспетчерскому управлению автобусного парка.

45. Разработка информационной подсистемы по созданию и заполнению рабочей программы в соответствии с учебным планом.

46. Разработка информационной подсистемы по расчету калькуляции строительства жилого дома.

47. Разработка информационной подсистемы по расчету калькуляции себестоимости готовых блюд в ресторанах и предприятиях общепита.

48. Программная реализация автоматизированной системы складского учета для фирмы, торгующей компьютерами и их комплектующими.

49. Разработка информационной подсистемы по расчету заработной платы (сдельной, повременной).

### Список рекомендуемой литературы

1. Брауде Э. Технология разработки программного обеспечения/ Пер. с англ. - СПб.: Питер, 2004. - 655 с.
2. Одинцов И. О. Профессиональное программирование. Системный подход. - 2-е изд. перераб. и доп. - СПб.: БХВ-Петербург, 2004. - 624 с.
3. Гамма Э., Хелм Р. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Пер. с англ. - СПб.: Питер, 2007. - 368 с.
4. Чарнецки К., Айзенекер У. Порождающее программирование: методы, инструменты, применение. Для профессионалов / Пер. с англ. - СПб.: Питер, 2005. - 731 с.
5. Буч Г., Якобсон А. UML. Классика CS / Пер. с англ. - 2-е изд. перераб. и доп. - СПб.: Питер, 2005. - 736 с.
6. Фаулер М. Архитектура корпоративных программных приложений / Пер. с англ. - М.: Вильямс, 2006. - 544 с.
7. Ларман К. Применение UML 2.0 и шаблонов проектирования/ Пер. с англ. - М.: Вильямс, 2007. - 736 с.
8. Гагарина Л. Г., Кокорева Е. В., Виснадул Б. Д. Технология разработки программного обеспечения. – М., Инфра-М, 2018. - 400 с.
9. Зелковиц М., Шоу А., Гэннон Дж. Принципы разработки программного обеспечения. - М., Мир, 2017. - 364 с.
10. Лукин В.В., Лукин В.Н., Лукин Т.В. Технология разработки программного обеспечения: учеб. пособие. – М., Питер, 2018. - 286 с.
11. Макаровских Т.А. Документирование программного обеспечения. В помощь техническому писателю: учеб. пособие. - М.: Ленанд, 2015. - 266 с.

**Приложение А**  
Форма титульного листа курсового проекта

Министерство науки и высшего образования Российской Федерации  
ФГБОУ ВО «Кубанский государственный технологический университет»  
(ФГБОУ ВО КубГТУ)

Институт \_\_\_\_\_

Кафедра \_\_\_\_\_

Направление подготовки \_\_\_\_\_  
(код и наименование направления)

Профиль \_\_\_\_\_  
(наименование профиля образовательной деятельности)

**КУРСОВОЙ ПРОЕКТ**

по дисциплине \_\_\_\_\_  
(наименование дисциплины)

на тему \_\_\_\_\_  
(тема курсового проекта)

Выполнил(а) студент(ка) \_\_\_\_\_ курса \_\_\_\_\_ группы \_\_\_\_\_  
(фамилия, имя, отчество)

Допущен к защите \_\_\_\_\_  
(дата)

Руководитель (нормоконтролер) работы \_\_\_\_\_ Фамилия И.О.  
(должность, дата)

Защищен \_\_\_\_\_ Оценка \_\_\_\_\_  
(дата)

Члены комиссии: \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

(подпись, дата, расшифровка подписи)

Краснодар  
20\_\_

**Приложение Б**  
Форма задания на курсовой проект

ФГБОУ ВПО «Кубанский государственный технологический университет»  
(ФГБОУ ВО КубГТУ)

Институт \_\_\_\_\_  
Кафедра \_\_\_\_\_  
Направление подготовки \_\_\_\_\_  
Профиль \_\_\_\_\_

УТВЕРЖДАЮ

Зав. кафедрой \_\_\_\_\_ М.В. Янаева.  
« \_\_\_\_ » \_\_\_\_\_ 20\_\_ г.

**ЗАДАНИЕ**  
на курсовой проект

Студенту: \_\_\_\_\_ группы \_\_\_\_\_ курса  
(Ф.И.О.) (№ группы и курса)

Тема проекта: \_\_\_\_\_  
(утверждена указанием директора института № \_\_\_\_ от \_\_\_\_ 20 \_\_\_\_)

План проекта:

1. \_\_\_\_\_
2. \_\_\_\_\_
3. \_\_\_\_\_

Объем работы:

- а) пояснительная записка \_\_\_\_\_ с.  
б) иллюстративная часть \_\_\_\_\_ лист (-ов)

Рекомендуемая литература

1. \_\_\_\_\_
2. \_\_\_\_\_
3. \_\_\_\_\_

Срок выполнения: с « \_\_\_\_ » \_\_\_\_\_ по « \_\_\_\_ » \_\_\_\_\_ 20\_\_ г.

Срок защиты: « \_\_\_\_ » \_\_\_\_\_ 20\_\_ г.

Дата выдачи задания: « \_\_\_\_ » \_\_\_\_\_ 20\_\_ г.

Дата сдачи проекта на кафедру: « \_\_\_\_ » \_\_\_\_\_ 20\_\_ г.

Руководитель работы \_\_\_\_\_ Фамилия И.О.  
(должность, подпись,)

Задание принял студент \_\_\_\_\_ Фамилия И.О.  
(подпись)

## **Приложение В**

### **Пример оформления реферата**

#### **Реферат**

Курсовой проект: ... с., ... рис., ... табл., ... использованных источников, ... приложения, ... иллюстративная часть.

АВТОМАТИЗАЦИЯ УПРАВЛЕНИЯ ВЗАИМООТНОШЕНИЯМИ С КЛИЕНТАМИ, CRM-СИСТЕМА, ERP-СИСТЕМА, MRP-СИСТЕМА, БАЗЫ ДАННЫХ, УПРАВЛЕНИЕ ПРОЕКТАМИ, ДОКУМЕНТООБОРОТ, ТОВАРООБОРОТ, СКЛАД, ФИЛИАЛ, МНОГОПОЛЬЗОВАТЕЛЬСКИЕ СИСТЕМЫ, УПРАВЛЕНИЕ АССОРТИМЕНТОМ, ДИАГРАММА КЛАССОВ, ДИАГРАММА ВЗАИМОДЕЙСТВИЯ ДАННЫХ, АВТОМАТИЗИРОВАННАЯ СИСТЕМА, АВТОМАТИЗАЦИЯ МАЛЫХ ПРЕДПРИЯТИЙ, ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Объектом автоматизации является деятельность салона сотовой связи (до 20 автоматизированных рабочих мест) для организации работы с клиентами сферы розничной торговли и бизнеса.

Целью работы является увеличение эффективности работы с клиентами за счёт обеспечения на техническом уровне следующих условий: стандартизации процессов работы с клиентами; наличия единого хранилища информации; синхронизация управления множественными каналами взаимодействия; моделирования систем управления запасами, доставкой, оперативным и бухгалтерским учетом; единой системы безопасности; обеспечения инструментов анализа динамики продаж; интегрирования различных видов деятельности предприятия; обеспечение взаимодействия между организациями.

Основные полученные результаты:

- проведён анализ ключевых процессов работы с клиентами, позволяющих выполнить автоматизацию в краткосрочной перспективе.
- спроектирован и разработан пакет прикладных программ для автоматизации предприятия.

Об эффективности внедрения можно судить по изменению таких показателей, как: время обработки обращения клиента, время заказа и приемки товара на головной склад и/или удаленный филиал, время на подготовку отчётов о динамике продаж и динамике клиентской базы, а также анализ продаж всего представленного ассортимента.

## ПРОЕКТИРОВАНИЕ ПРОГРАММНЫХ КОМПЛЕКСОВ

Методические указания

Внутри кафедральное издание

Составитель

Янаева Марина Викторовна