

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное  
образовательное учреждение высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ

## ПРОЕКТИРОВАНИЕ ПРОГРАММНЫХ СИСТЕМ

Методические указания  
к выполнению лабораторных работ № 1-4

Санкт-Петербург 2018

Составитель Е. В. Павлов

Рецензент

В методические указания включены краткие теоретические сведения, которые необходимы для выполнения лабораторных работ, требования к содержанию отчетов и примеры выполнения, а также варианты индивидуальных заданий.

Методические указания предназначены для выполнения лабораторных работ по курсу «Проектирование программных систем» для студентов всех форм обучения, проходящих подготовку по направлениям 02.03.03 «Математическое обеспечение и администрирование информационных систем» и 09.03.04 «Программная инженерия».

## СОДЕРЖАНИЕ

1. ЛАБОРАТОРНАЯ РАБОТА «СТРУКТУРНЫЙ АНАЛИЗ СИСТЕМЫ. РАЗРАБОТКА ДИАГРАММЫ ПОТОКОВ ДАННЫХ И СОСТАВЛЕНИЕ СПЕЦИФИКАЦИИ ПРОЦЕССОВ».....	4
2. ЛАБОРАТОРНАЯ РАБОТА «РАЗРАБОТКА СПЕЦИФИКАЦИИ ПРОГРАММНЫХ ТРЕБОВАНИЙ».....	25
3. ЛАБОРАТОРНАЯ РАБОТА «ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ К ДАННЫМ. ЛОГИЧЕСКАЯ МОДЕЛЬ И СЛОВАРЬ ДАННЫХ».....	45
4. ЛАБОРАТОРНАЯ РАБОТА «ТЕСТИРОВАНИЕ ТРЕБОВАНИЙ. СОСТАВЛЕНИЕ СЦЕНАРИЕВ ТЕСТИРОВАНИЯ».....	62
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	73
ПРИЛОЖЕНИЕ.....	74

**1. ЛАБОРАТОРНАЯ РАБОТА**  
**«СТРУКТУРНЫЙ АНАЛИЗ СИСТЕМЫ.**  
**РАЗРАБОТКА ДИАГРАММЫ ПОТОКОВ ДАННЫХ**  
**И СОСТАВЛЕНИЕ СПЕЦИФИКАЦИИ ПРОЦЕССОВ»**

**1.1 Цель работы**

Целью данной работы является изучение методологии графического структурного анализа, построение структурной модели на основе диаграмм потоков данных и уточнение данной модели посредством спецификации процессов.

**1.2 Задание на лабораторную работу**

Разработать структурную модель программной системы или её функционально законченной части.

1. Построить контекстную диаграмму потоков данных;
2. Осуществить декомпозицию контекстной диаграммы (построить диаграмму потоков данных 1-го уровня);
3. Осуществить декомпозицию диаграммы 1-го уровня (построить диаграмму потоков данных 2-го уровня);
4. Составить спецификации всех процессов диаграммы потоков данных 1-го уровня с учетом подпроцессов при декомпозиции.

При составлении спецификаций все процессы должны быть разбиты на равные по количеству процессов группы.

Лабораторная работа будет оцениваться в соответствии со следующими критериями:

	Удовл.	Хорошо	Отлично
Количество процессов на диаграмме потоков данных 1-го уровня	10	12	15
Диаграмма потоков данных 2-го уровня должна представлять собой декомпозицию любых $N$ процессов диаграммы 1-го уровня	$N = 3$	$N = 5$	$N = 7$
Количество способов представления спецификации процессов (структурированный естественный язык, псевдокод, блок-схема) <sup>1</sup>	1	2	3

---

<sup>1</sup> Выбор способа представления спецификации процессов остается за студентом, кроме задания на «отлично», в данном случае необходимо представить спецификацию процессов всеми тремя предложенными способами.

### 1.3 Порядок выполнения работы

1. Получить вариант задания (тему) у преподавателя<sup>2</sup>;
2. Изучить теоретический материал, изложенный в подразделе 1.4;
3. Разработать структурную модель программной системы и составить спецификацию всех процессов диаграммы потоков данных 1-го уровня;
4. Ознакомиться с требованиями по содержанию отчета в подразделе 1.5;
5. Написать отчет о работе.

Для выполнения лабораторных работы можно воспользоваться любой средой моделирования или CASE-средством, которое поддерживает соответствующие структурные нотации (Yourdon & Coad, Gane & Sarson) диаграммы потоков данных. Рекомендуется использовать бесплатное для некоммерческого использования CASE-средство [Software Ideas Modeler](#).

### 1.4 Теоретический материал

Диаграммы потоков данных (Data Flow Diagrams, DFD) — методология графического структурного анализа, описывающая внешние по отношению к системе источники и адресаты данных, логические функции, потоки и хранилища данных, к которым осуществляется доступ.

При использовании структурной модели для анализа и проектирования систему представляют в виде иерархии диаграмм потоков данных. На каждом следующем уровне иерархий происходит декомпозиция, то есть разбиение на структурные составляющие процессов, которые преобразуют входную информацию с заданным алгоритмом в выходную. Когда достигается требуемая глубина декомпозиции, процессы нижнего уровня сопровождаются спецификацией (например, псевдокодом, который является по сути нижним уровнем декомпозиции процесса).

В основе структурной модели лежат понятия внешней сущности, процесса, хранилища (накопителя) данных и потока данных.

*Внешняя сущность* — материальный объект или физическое лицо, выступающие в качестве источников или приемников информации, например, заказчик, персонал, пользователь (тот, кто использует продукт), клиент (тот, кто приобретает продукт), склад, дата-центр, etc. Определение некоторого объекта или системы в качестве внешней сущности указывает на то, что объект или система находятся за пределами границ анализируемой информационной системы. В процессе анализа некоторые внешние сущности могут быть перенесены внутрь контура (диаграммы) анализируемой системы, если это необходимо, или, наоборот, часть процессов системы может быть вынесена за пределы диаграммы и представлена как внешняя сущность.

*Процесс* — преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом. Физически процесс может быть реализован различными способами: это может быть подразделение организации (отдел), выполняющее обработку входных документов и составление отчетов, программа, аппаратно-реализованное логическое устройство, скрипт для реализации функций взаимодействия пользователя с веб-сайтом, etc. Таким образом, физически преобразование может осуществляться программными средствами, вручную или

---

<sup>2</sup> Список индивидуальных заданий представлен в приложении.

специальными устройствами. Каждый процесс в системе имеет свой номер и связан с исполнителем, который осуществляет данное преобразование.

На верхних уровнях иерархии, когда процессы ещё не определены, вместо понятия «процесс» используют понятия «система» и «подсистема», которые обозначают соответственно систему в целом или её функционально законченную часть в виде отдельного модуля или элемента системы.

*Накопитель данных* представляет собой абстрактное устройство для хранения информации. Тип устройства и способы помещения, извлечения и хранения для такого устройства не детализируют. Физически это может быть база данных, файл, таблица в оперативной памяти, картотека на бумаге, etc.

Идентификатор накопителя данных, который может быть произвольным числом, предваряет буква «D». Имя накопителя выбирается из соображения наибольшей информативности при проектировании информационной системы.

Накопитель данных в общем случае является прообразом будущей базы данных. Описание самих данных, хранение которых обеспечивается в накопителе, должно быть отражено в семантической модели в виде диаграммы сущность-связь.

*Поток данных* — процесс передачи некоторой информации от источника к приемнику. Физически процесс передачи информации может происходить по кабелям под управлением программы или программной системы или вручную при участии устройств или людей вне проектируемой системы.

Таким образом, диаграмма иллюстрирует как потоки данных, порожденные некоторыми внешними сущностями, преобразуются соответствующими процессами (или подсистемами), сохраняются накопителями данных и передаются другим внешним сущностям — приемникам информации. В результате мы получаем структурную модель хранения (обработки) информации.







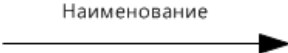
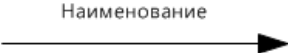
Построение иерархии диаграмм потоков данных начинают с диаграммы особого вида — *контекстной диаграммы*, которая определяет наиболее общий вид системы или определяет границы проекта. На такой диаграмме показывают, как разрабатываемая система будет взаимодействовать с приемниками и источниками информации без указания исполнителей (процессов), иными словами описывают интерфейс между системой и внешним миром.

Если проектируемая система содержит большое количество внешних сущностей (более 10-ти), имеет распределенную природу или включает уже существующие подсистемы, то строят иерархии контекстных диаграмм. Иными словами структурную модель системы допустимо разбивать на несколько диаграмм потоков данных 1-го уровня.

Для графического описания диаграмм потоков данных в равной степени используют две нотации — Йордана-Коуда (Yourdon-Coad) и Гейна-Сарсона (Gane-Sarson), которые отличаются синтаксисом. В настоящих методических указаниях используется нотация Гейна-Сарсона, однако при выполнении лабораторной работы студент вправе выбрать любую из представленных (табл.1.4).

Диаграммы потоков данных в нотации Йордана-Коуда обычно используют для системного анализа и проектирования, в то время как нотация Гейна-Сарсона чаще применяется для визуализации информационных систем.

Таблица 1.4 — Элементы диаграммы потоков данных

Понятие	Нотация Йордана-Коуда	Нотация Гейна-Сарсона
Внешняя сущность		
Система, подсистема или процесс		
Накопитель данных		
Поток данных		

Над линией потока, направление которого обозначают стрелкой, указывают, какая конкретно информация в данном случае передается (рис. 1.4).

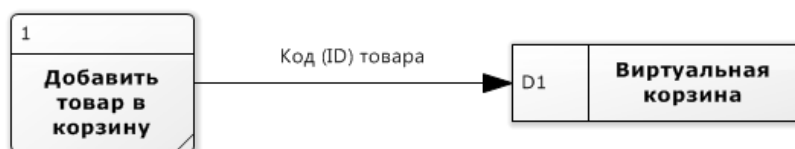


Рисунок 1.4 — Пример потока данных  
(спецификация Гейна-Сарсона)

В нотации Гейна-Сарсона у процессов иногда добавляют дополнительное поле после наименования процесса — поле физической реализации, в котором указывают, какое подразделение организации, аппаратное устройство, программа или скрипт выполняют данный процесс (исполнитель). Данное поле не является обязательным для заполнения и во многих CASE-средствах в нотации Гейна-Сарсона оно отсутствует.

*Спецификации процессов* представляют собой описания алгоритмов задач, выполняемых процессами. Они содержат номер и/или имя процесса и описание логики (функции) данного процесса с указанием входных и выходных данных. Спецификацию процесса обычно представляют в виде структурированного естественного языка, псевдокодов, схем алгоритмов (например, блок-схемы или диаграммы деятельности), деревьев или таблиц решений, Flow-форм или диаграмм Насси-Шнейдермана, etc.

*Структурированный естественный язык* представляет собой комбинацию формализма языка программирования и доступности естественного языка, включает в себя подмножество ключевых слов, организованных в определенные логические структуры. Ключевые слова в структурированном естественном языке выделяются заглавными буквами, а логика процессов выражается в виде комбинации последовательных конструкций выполнения команд, выбора и итераций.

#### 1. Конструкции выполнения

ВЫПОЛНИТЬ «операция» // двойной слеш может служить для указания комментария  
«другая операция»

Для разделения операций используется новая строка, однако можно задать и другие разделители, например точку с запятой. Возможен и такой вариант:

ВЫПОЛНИТЬ «операция»  
ВЫПОЛНИТЬ «другая операция»

Но он более избыточен, чем предыдущий. Операции не обязательно заключать в кавычки, в настоящих методических указаниях это сделано для визуального комфорта.

#### 2. Конструкции выбора

ЕСЛИ «условие» ТО  
ВЫПОЛНИТЬ «операция»  
ИНАЧЕ  
ВЫПОЛНИТЬ «другая операция»  
КОНЕЦ ЕСЛИ

В конструкциях выбора, как и во всем структурированном естественном языке допускается изменять нотацию написания, например, можно записать часть ключевых слов в одну строку. Кроме того, допускается выделять отступами вложенные конструкции для большей наглядности.

ИНАЧЕ ВЫПОЛНИТЬ «другая операция» КОНЕЦ ЕСЛИ

#### 3. Конструкции итераций

ПОКА «условие»  
ВЫПОЛНИТЬ «операция»  
КОНЕЦ ПОКА

Набор ключевых слов также может быть дополнен в зависимости от структуры описываемой схемы алгоритма.



Пример спецификации процесса «Выполнение лабораторных работ» на структурированном естественном языке:

ВХОДНЫЕ ДАННЫЕ: событие «сессия» (true / false), важные дела (true / false)

ВЫХОДНЫЕ ДАННЫЕ: рейтинг за практические занятия

ЕСЛИ «сессия»

ВЫПОЛНИТЬ «переход в режим супергерой Человек-студент»,

«сделать все лабораторные работы за 24 часа»,

«получить минимальный рейтинг за практику (и осуждающий взгляд преподавателя)»,

«сократить продолжительность своей жизни за счет стресса от режима супергероя»

ИНАЧЕ

ЕСЛИ «есть более важные дела (чем выполнение ЛР)»

ВЫПОЛНИТЬ «просматривать картинки в социальных сетях; отвечать в

Интернете тем, кто не прав; ругать тиммейтов за то, что у них физиологические дефекты (руки не из того места, рак мозга в терминальной стадии, отсутствие признаков эволюции со времен кайнозойской эры и многие другие определяющие характеристики); etc.»

ИНАЧЕ ВЫПОЛНИТЬ «сделать лабораторные работы с соблюдением сроков»,

«получить максимальный рейтинг за практику (и хорошее настроение)»

КОНЕЦ ЕСЛИ

КОНЕЦ ЕСЛИ

Псевдокод представляет собой компактный язык описания алгоритмов, использующий ключевые слова языков программирования, но опускающий несущественные подробности и специфический синтаксис. Псевдокод обычно опускает детали, несущественные для понимания алгоритма человеком. Такими несущественными деталями могут быть описания переменных, системно-зависимый код и подпрограммы. Таким образом, главная цель использования псевдокода — обеспечить понимание алгоритма человеком и сделать описание более воспринимаемым, чем исходный код на языке программирования. Синтаксис псевдокода может быть определен привязкой к какому-либо языку программирования, например, Pascal, C, PHP etc. В псевдокоде математические выражения обычно представляются в том виде, как их принято записывать в математике, а не в языках программирования, и некоторые фрагменты псевдокода могут быть фразами естественного языка (русского, английского etc.).

Пример спецификации процесса «Выполнение ЛР» на псевдокоде:

ВХОДНЫЕ ДАННЫЕ: событие «сессия» (true / false), важные дела (true / false)

ВЫХОДНЫЕ ДАННЫЕ: рейтинг за практические занятия

```
if(exam) {
    set superman_student; do lab_work; get lose_lifetime; get min_rating;
} else {
    if(important_task) {
        do task;
    } else {
        do lab_work; get max_rating;
    }
}
```

Псевдокод можно представить и в нотации русского языка:

```
если (сессия) {  
    супергерой_человек_студент = true;  
    выполнить лаб_работа;  
    продолжительность_жизни = продолжительность_жизни - RAND();  
    // RAND() — функция псевдослучайных чисел  
    получить мин_рейтинг;  
} иначе {  
    если(важная_задача) {  
        выполнить важная_задача;  
    } иначе {  
        выполнить лаб_работа;  
        получить макс_рейтинг;  
    }  
}
```

Базовые управляющие структуры псевдокода аналогичны соответствующим конструкциям в структурированном естественном языке с тем лишь отличием, что их синтаксис определяется за счет привязки к конкретному языку программирования.

В примере выполнения работы спецификация процессов на псевдокоде выполнена с привязкой к языку PHP с комментариями на русском языке и приведением упрощенной формы запросов к БД на языке SQL. Детализация и сложность описания процесса (вне зависимости от формы представления, будь то псевдокод или дерево решений) определяется исходя из соображений доступности (понятности) тому, для кого она предназначена. Например, процесс 2 «Регистрация» можно представить в следующей форме псевдокода:

```
if(существуют данные форм регистрации) {  
    if(данные корректны) {  
        result = запрос к БД на совпадение с указанными логином и паролем;  
        // если совпадения не найдены, вернет false, в противном случае true  
        if(result) {  
            сообщение('Данные логин или почта уже заняты');  
        } else { // если данные уникальны  
            произвести запрос к БД на добавление нового пользователя;  
            // и задать статус «ожидает подтверждения регистрации»  
            отправить письмо пользователю для подтверждения регистрации;  
            сообщение('Для подтверждения регистрации перейдите по ссылке в письме');  
        }  
    } else { сообщение('Проверьте введенные данные'); }  
}  
  
if(пользователь перешел по ссылке в письме) {  
    обновить данные в БД; // изменить статус пользователя на «зарегистрирован»  
    сообщение('Ваша учетная запись активирована');  
}
```

Блок-схемы можно рассматривать как графическую альтернативу псевдокоду. В настоящих методических указаниях к работе подробно не рассматриваются элементы блок-схемы, так как подразумевается, что данный материал является базовым и был изучен ранее.

Пример спецификации процесса «Выполнение ЛР» в виде блок-схемы:

ВХОДНЫЕ ДАННЫЕ: событие «сессия» (true / false), важные дела (true / false)  
ВЫХОДНЫЕ ДАННЫЕ: рейтинг за практические занятия

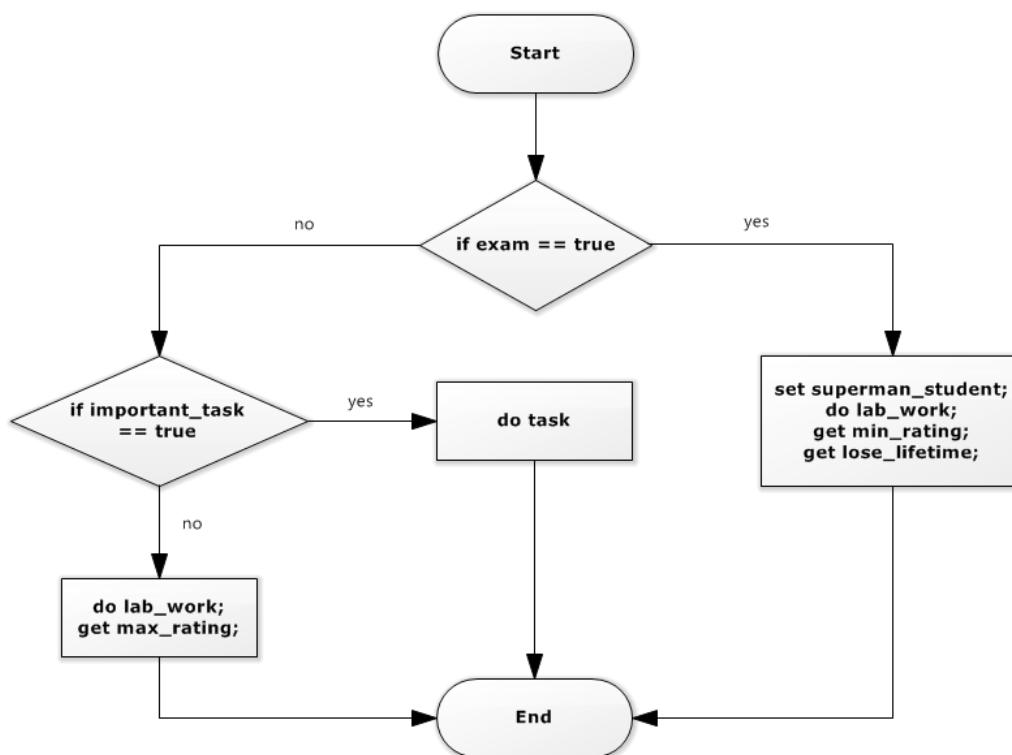


Рисунок 1.5 — Блок-схема процесса «Выполнение лабораторных работ»

Каждый из способов представления спецификации имеет свое определенное предназначение, которое зависит от пользователя данной спецификации (программист, аналитик, специалист по тестированию, etc.), его квалификации, типов решаемых задач, специфики конкретных процессов и самой программной системы.

Во время выполнения работы при выборе, к какой группе способа представления спецификации отнести конкретный процесс (для оценок «хорошо» и «отлично»), следует руководствоваться соображениями удобства и простоты реализации.

## 1.5 Содержание отчета

Отчет о работе должен содержать:

1. Титульный лист
2. Цель работы
3. Задание на лабораторную работу и вариант
4. Структурная модель информационной системы «Наименование»
  - 4.1 Контекстная диаграмма
  - 4.2 Диаграмма потоков данных 1-го уровня
  - 4.3 Диаграмма потоков данных 2-го уровня
5. Спецификация процессов структурной модели
6. Выводы по работе
7. Используемые источники (2-4 источника)

## 1.6 Пример выполнения работы

Структурная модель информационной системы «Интернет-магазин»

Контекстная диаграмма потоков данных

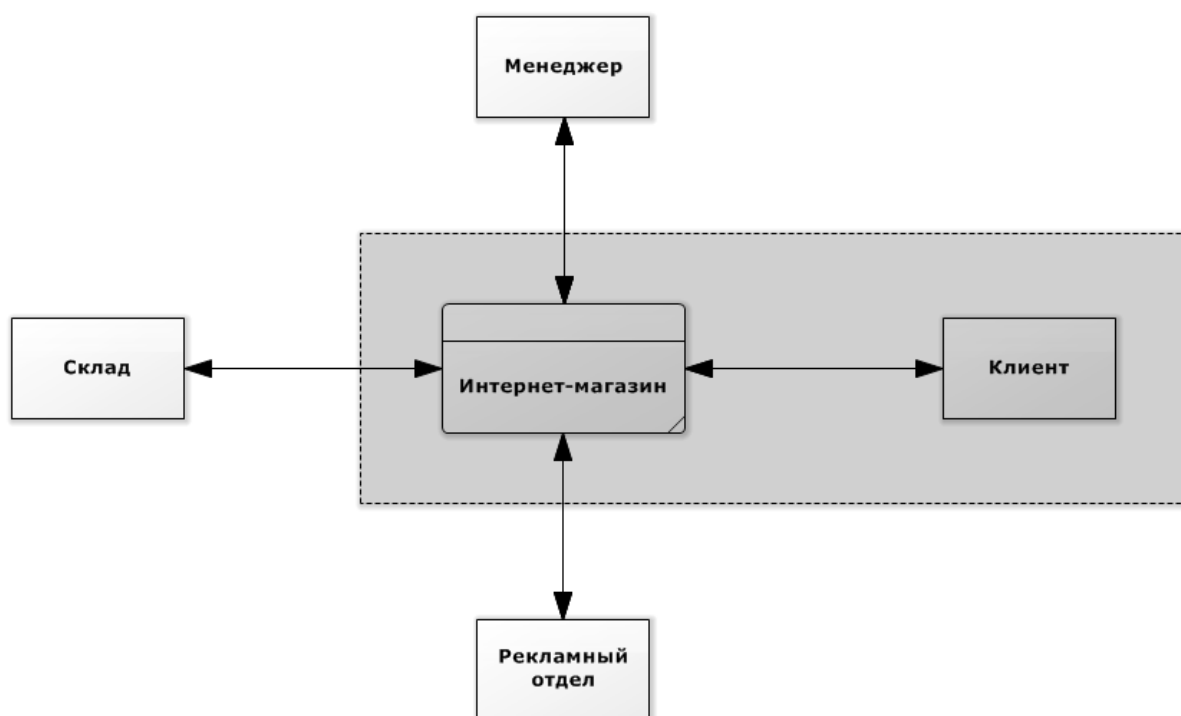


Рисунок 1.6.1 — Контекстная диаграмма системы «Интернет-магазин»

На рис. 1.6.1 серым контуром обозначен фрагмент контекстной диаграммы потоков данных, декомпозиция которого будет осуществлена. На данной контекстной диаграмме не указаны наименования потоков данных, это сделано для демонстрации общих отношений между системой и сущностями, но в целом для большей информативности контекстной диаграммы желательно выделить наименования основных потоков данных, как это сделано на рис. 1.6.2, кроме того, такой вид диаграммы используется при составлении спецификации программных требований, однако первый вариант диаграммы также допустим в работе.

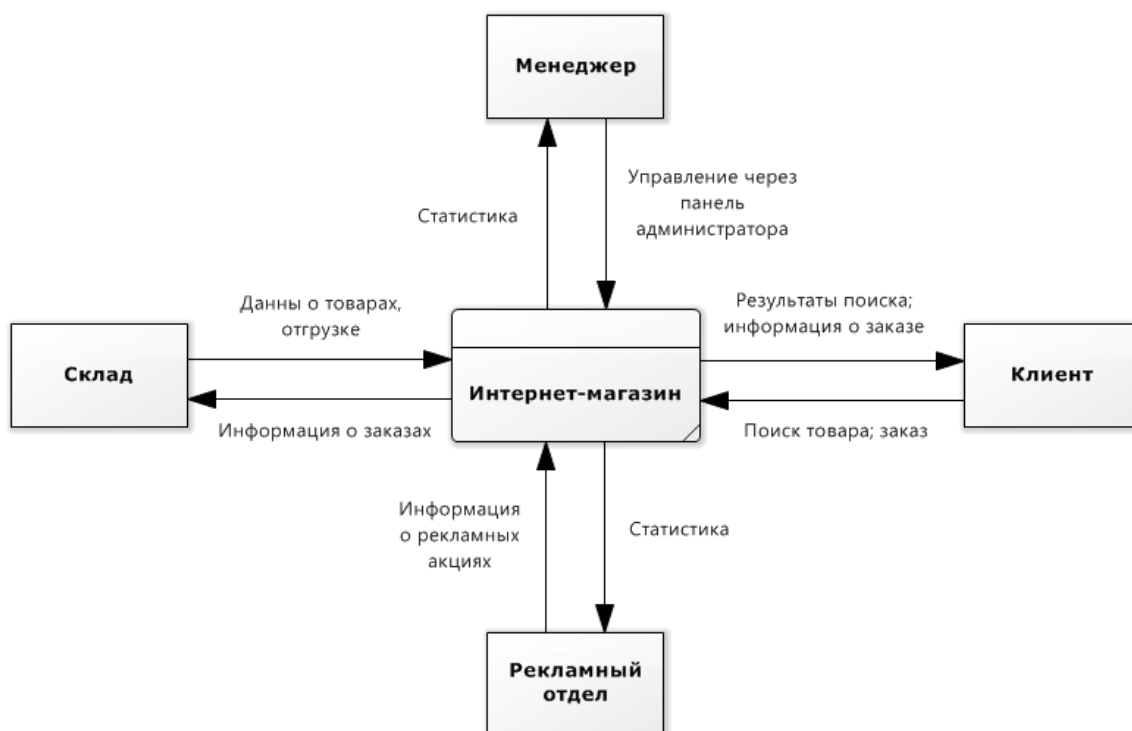


Рисунок 1.6.2 — Контекстная диаграмма системы «Интернет-магазин» с указанием наименований основных потоков данных

Если на контекстной диаграмме фигурирует одна система, то номер системы можно не указывать.

## Диаграмма потоков данных 1-го уровня

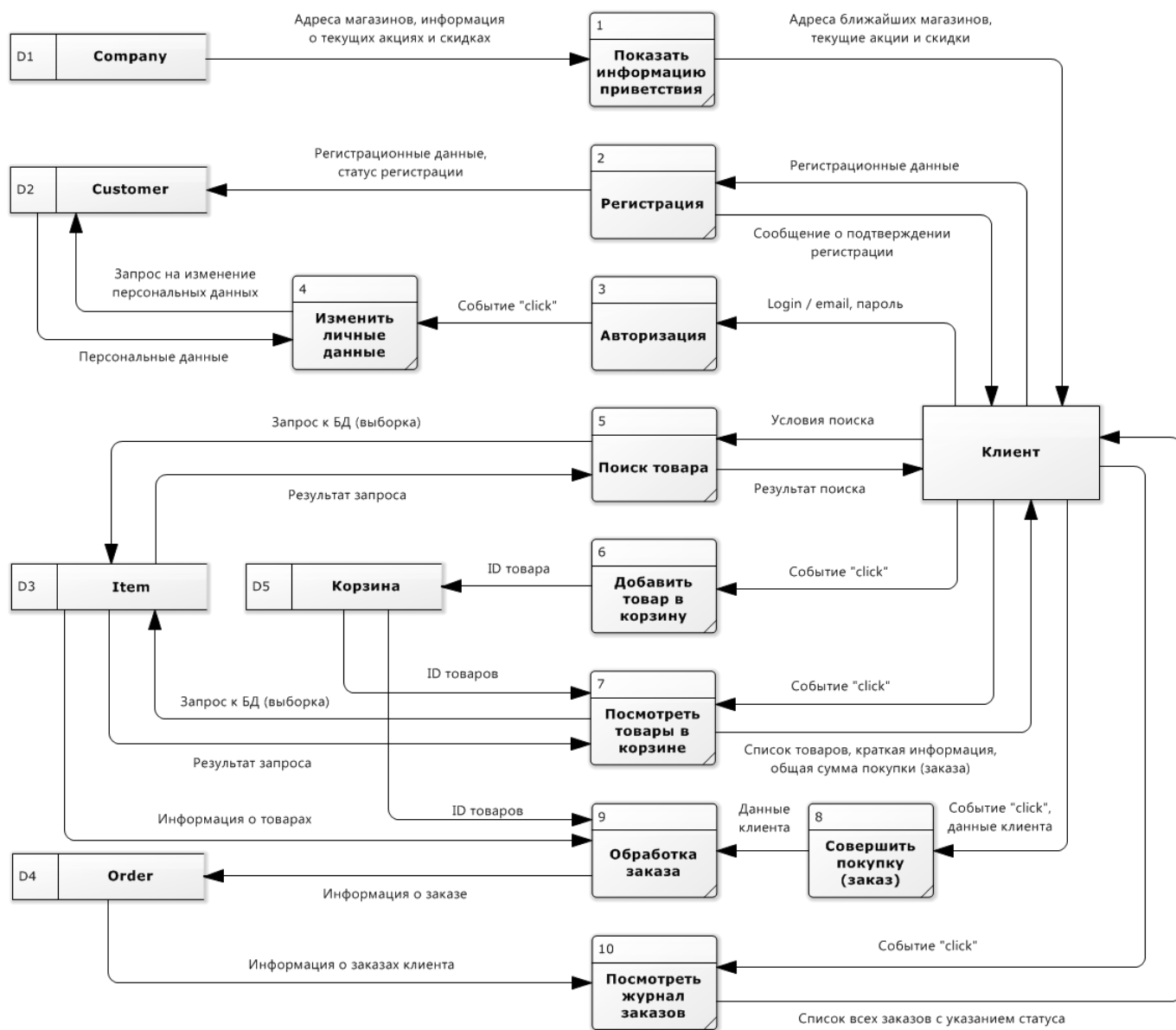


Рисунок 1.6.3 — Диаграмма потоков данных 1-го уровня для информационной системы «Интернет-магазин»

В Software Ideas Modeler:

1. Изменение типа линии:левой кнопкой мыши (ЛКМ) на поток → Line Style → Rectangular;
2. Изменение направления потока: ЛКМ на поток: Relationship → Reverse Relationship;
3. Экспорт диаграммы в изображение: Diagram → Export → Image.

Приведенную диаграмму потоков данных 1-го уровня (рис. 1.6.3) можно семантически структурировать, то есть разбить на ряд диаграмм потоков данных 1-го уровня:

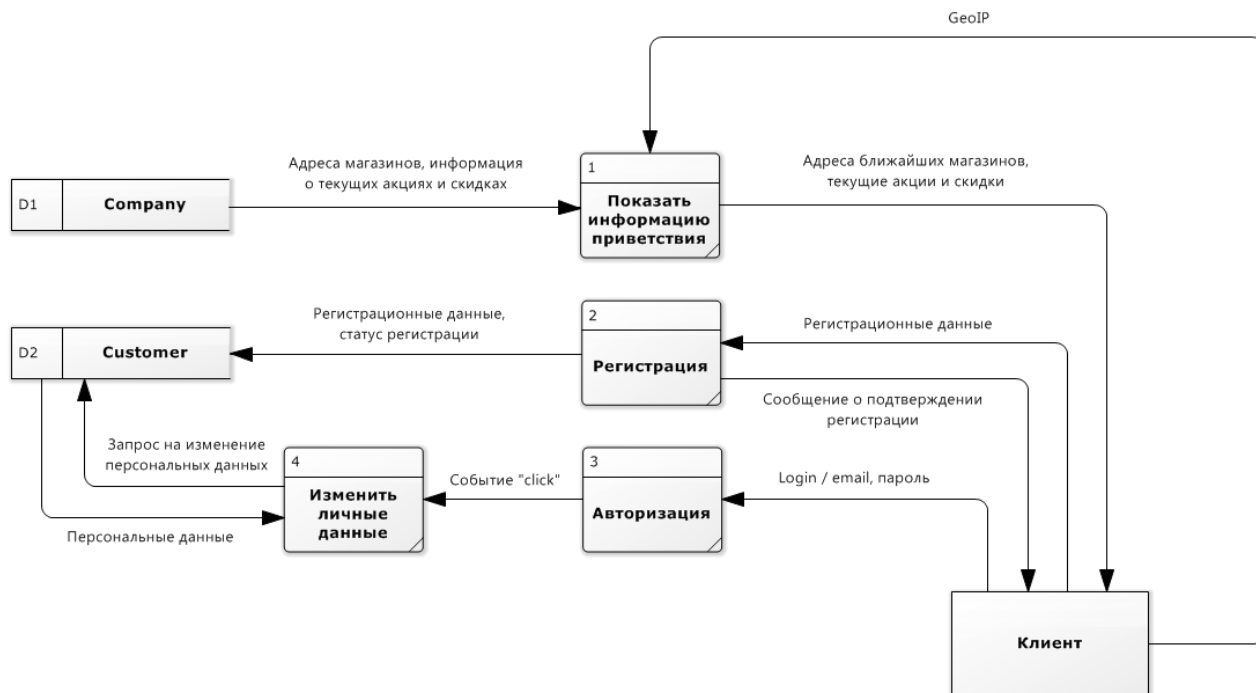


Рисунок 1.6.4 — Диаграмма потоков данных для общих процессов

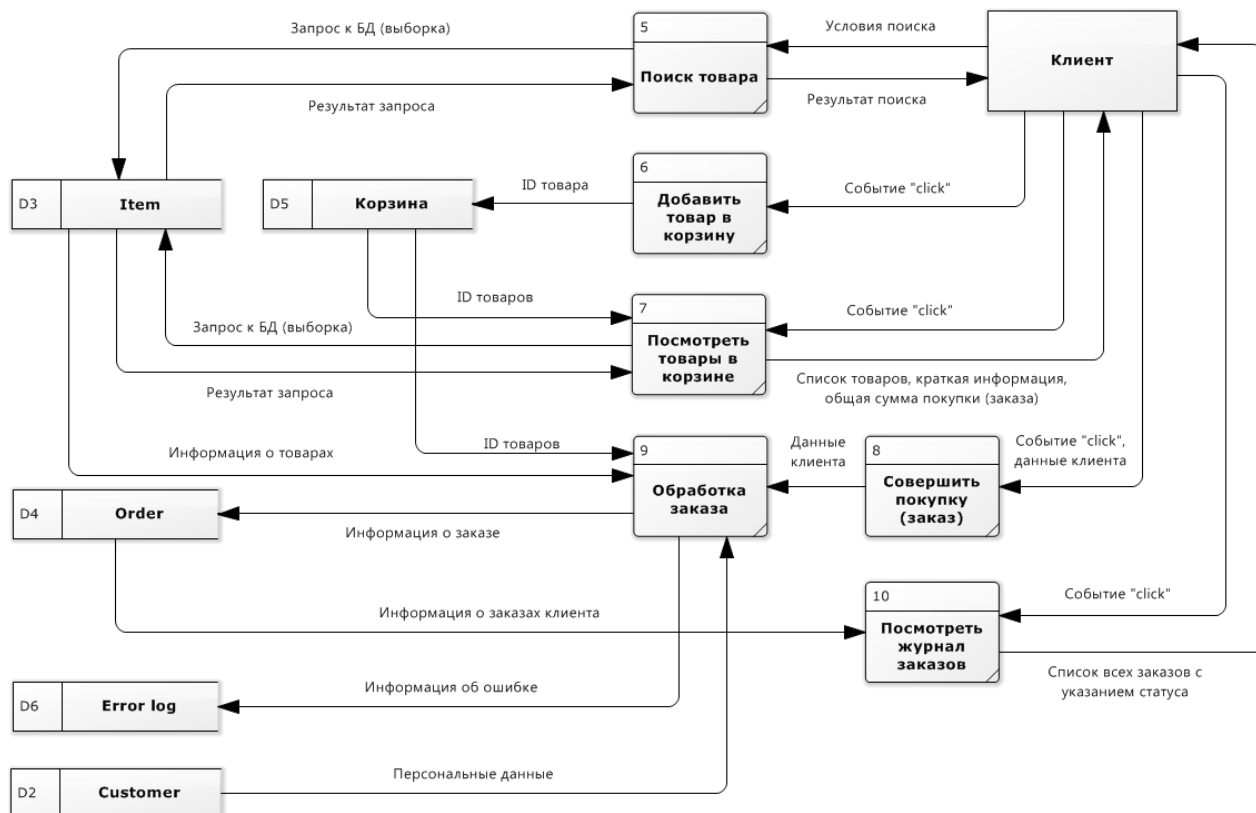


Рисунок 1.6.5 — Диаграмма потоков данных для процессов,

имеющих прямое или косвенное отношение к покупке товара  
 Диаграмма потоков данных 2-го уровня

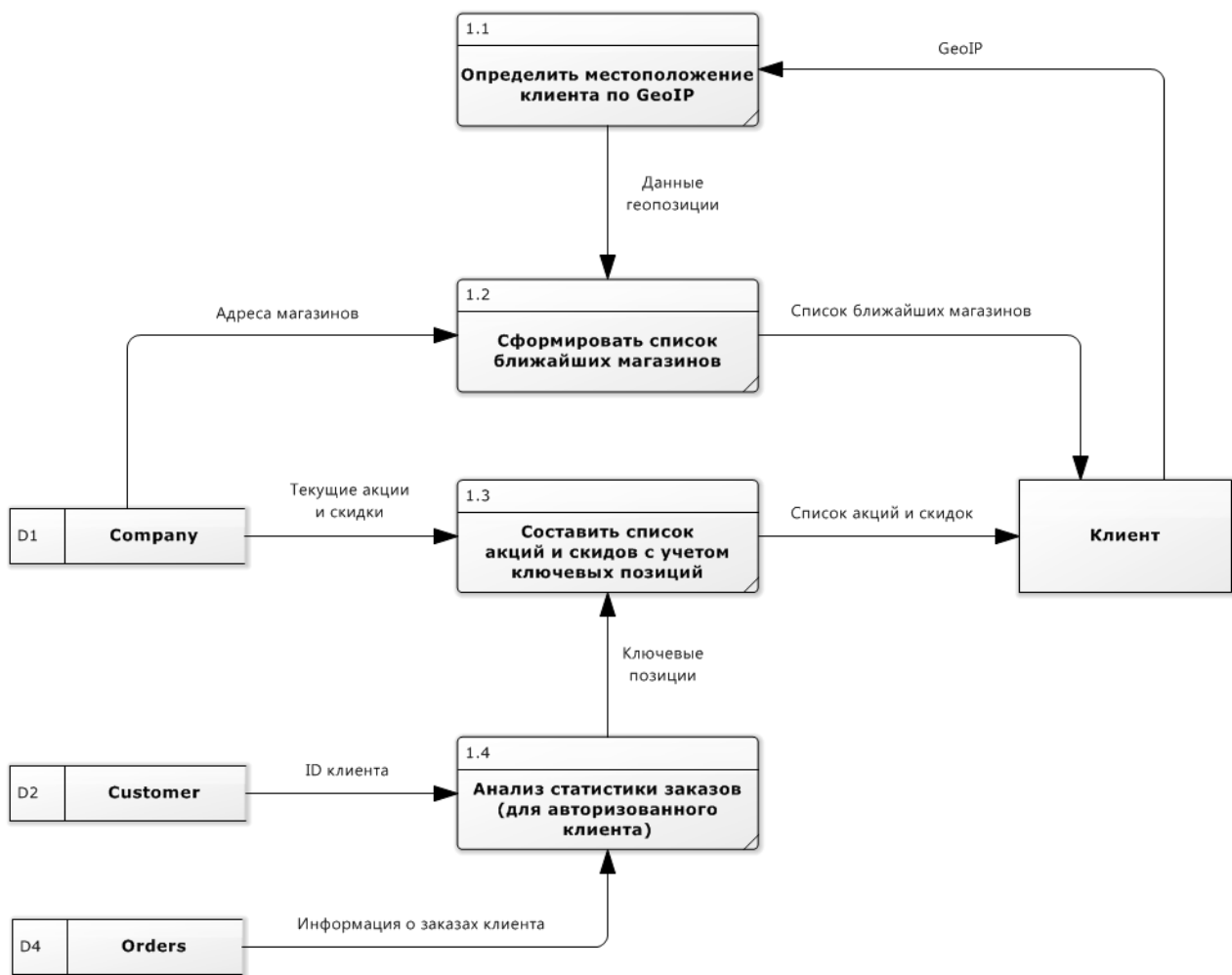


Рисунок 1.6.6 — Декомпозиция процесса №1 «Показать информацию приветствия»

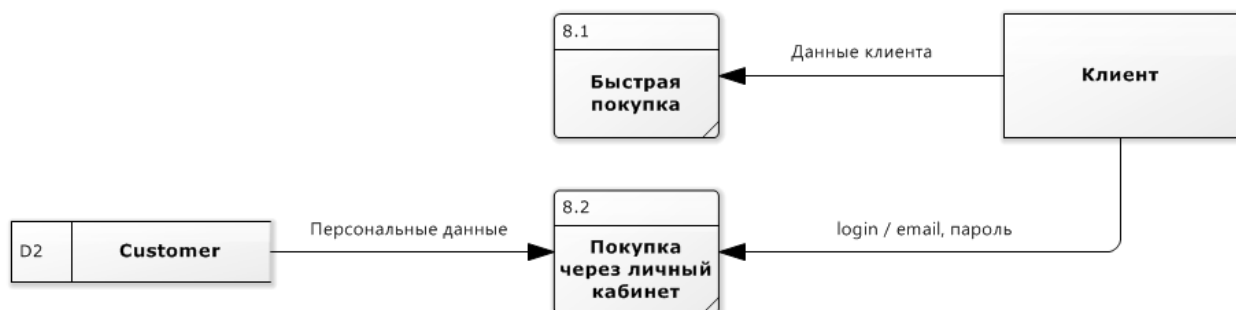


Рисунок 1.6.7 — Декомпозиция процесса №8 «Совершить покупку (заказ)»



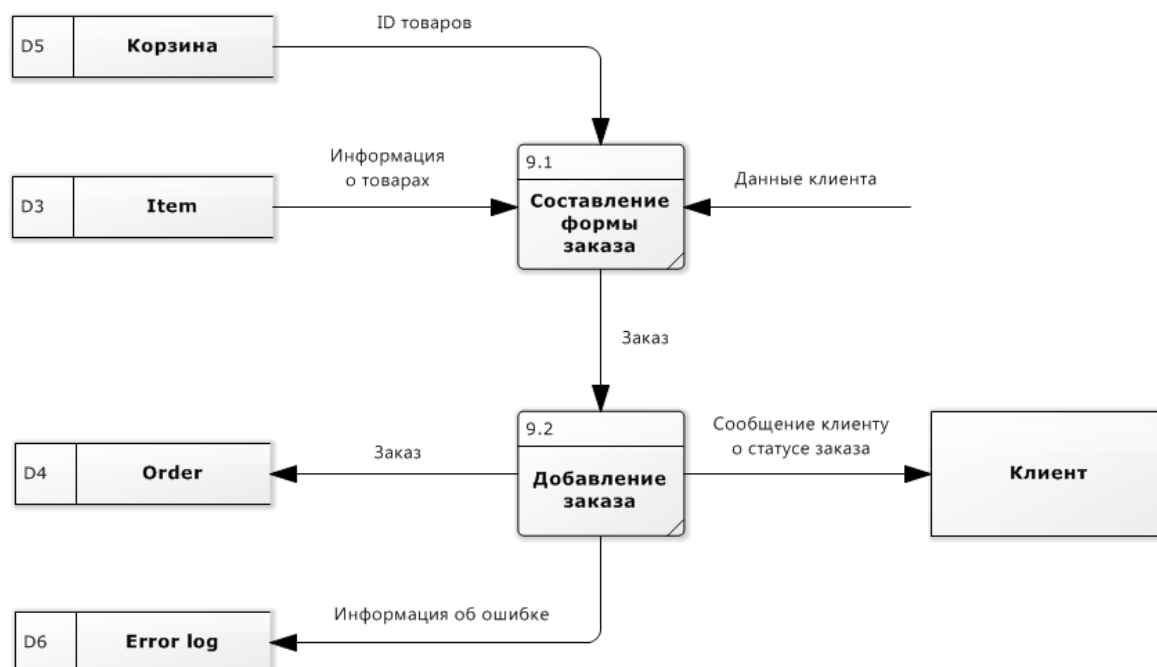


Рисунок 1.6.8 — Декомпозиция процесса №9 «Обработка заказа»

На диаграмме потоков данных 2-го уровня должны быть отражены сущности и накопители данных, с которыми осуществляется взаимодействие исходного процесса на 1-ом уровне иерархии.

Обратите внимание, что некоторые потоки данных не отражены на диаграмме потоков данных 1-го уровня, например поток «Персональные данные», связывающий накопитель D2 и подпроцесс 8.2 (рис. 1.6.7) или «Сообщение клиенту о статусе заказа» (рис. 1.6.8). Диаграмма потоков данных не ставит перед собой задачу, обозначить совершенно все потоки данных, которые присутствуют в анализируемой или проектируемой системе, кроме того, это ухудшит ее читаемость, поэтому указывают только те потоки, которые существенны на данном уровне иерархии диаграммы и стадии проектирования системы.

При декомпозиции также могут появляться новые хранилища данных (рис. 1.6.8), накопитель D6 на диаграмме 1-го уровня не отражен, так как запись в журнал регистрации ошибок происходит только при возникновении ошибки во время добавления заказа и данный накопитель не является принципиальным для понимания функциональных требований к проектируемой системе.

## Спецификация процессов для структурной модели системы «Интернет-магазин»

Таблица 1.6 — Способы представления спецификации:

Структурированный естественный язык	Псевдокод
Процесс 1 – Информация приветствия Процесс 5 – Поиск товара Процесс 6 – Добавить товар в корзину Процесс 7 – Посмотреть товары в корзине Процесс 10 – Посмотреть журнал заказов	Процесс 2 – Регистрация Процесс 3 – Авторизация Процесс 4 – Изменить личные данные Процесс 8 – Совершить покупку (заказ) Процесс 9 – Обработка заказа

Спецификация процессов на структурированном естественном языке:

*Процесс 1 «Показать информацию приветствия»*

*Подпроцесс 1.1 «Определить местоположение клиента по IP»*

*Подпроцесс 1.2 «Сформировать список ближайших магазинов»*

*Подпроцесс 1.3 «Составить список акций и скидок с учетом ключевых позиций»*

*Подпроцесс 1.4 «Анализ статистики заказов (для авторизованного клиента)»*

ВХОДНЫЕ ДАННЫЕ: IP клиента; статистические данные

ВЫХОДНЫЕ ДАННЫЕ: список ближайших магазинов, список акций и скидок

ВЫПОЛНИТЬ «определить местоположение клиента по GeoIP»,

«сообщить клиенту его предполагаемое местоположение»,

«попросить подтвердить (или уточнить) местоположение»

ЕСЛИ «клиент подтвердил местоположение» ТО

ВЫПОЛНИТЬ «вывести список ближайших магазинов»

ИНАЧЕ ВЫПОЛНИТЬ «вывести список ближайших магазинов для предполагаемого местоположения через N секунд»

КОНЕЦ ЕСЛИ

ВЫПОЛНИТЬ «получить данные о текущих акциях и скидках»

ЕСЛИ «клиент авторизован» ТО

ВЫПОЛНИТЬ «анализ статистики заказов клиента за последние N (дней)»,

ЕСЛИ «найжены пересечения статистики заказов клиента и текущих акций» ТО

«вывести соответствующие акции и скидки первыми в списке»,

«вывести прочие текущие акции и скидки по дате»

ИНАЧЕ ВЫПОЛНИТЬ «вывести текущие акции и скидки по дате»

КОНЕЦ ЕСЛИ

ИНАЧЕ ВЫПОЛНИТЬ «вывести текущие акции и скидки по дате»

КОНЕЦ ЕСЛИ

*Процесс 5 «Поиск товара»*

ВХОДНЫЕ ДАННЫЕ: наименование товара; условия фильтра

ВЫХОДНЫЕ ДАННЫЕ: код товара, наименование, описание, характеристики, цена

ЕСЛИ «заполнено поле наименования товара / заданы условия фильтра» ТО

ВЫПОЛНИТЬ «запрос к БД на поиск соответствующей позиции»,

«вывести результат поиска»

ИНАЧЕ «предложить пользователю заполнить формы поиска»

КОНЕЦ ЕСЛИ

*Процесс 6 «Добавить товар в корзину»*

ВХОДНЫЕ ДАННЫЕ: событие «click»; ID товара

ВЫХОДНЫЕ ДАННЫЕ: список или (обновленные) пиктограммы товаров в корзине

ЕСЛИ «нажата кнопка «добавить в корзину» ТО

    ВЫПОЛНИТЬ «записать код (ID) товара в cookie»,

    «обновить содержание виртуальной корзины с учетом новых покупок»

КОНЕЦ ЕСЛИ

*Процесс 7 «Посмотреть товары в корзине»*

ВХОДНЫЕ ДАННЫЕ: событие «click»; ID товаров в корзине

ВЫХОДНЫЕ ДАННЫЕ: наименование товаров, описание, цена; общая сумма покупки

ЕСЛИ «нажата кнопка виртуальной корзины» ТО

    ВЫПОЛНИТЬ «запрос к БД по коду (ID) товара из cookie»

    «вывести описание каждой позиции с указанием общей суммы заказа»

КОНЕЦ ЕСЛИ

*Процесс 10 «Посмотреть журнал заказов»*

ВХОДНЫЕ ДАННЫЕ: событие «click»; ID клиента

ВЫХОДНЫЕ ДАННЫЕ: список заказов, статус заказов

ЕСЛИ «нажата кнопка «журнал заказов» ТО

    ВЫПОЛНИТЬ «запрос к БД по ID клиента»

    «вывести список всех заказов клиента с указанием их статуса»

КОНЕЦ ЕСЛИ

## Спецификация процессов на псевдокоде:

### Процесс 2 «Регистрация»

ВХОДНЫЕ ДАННЫЕ: login, password, email

ВЫХОДНЫЕ ДАННЫЕ: сообщение о статусе регистрации

```
if(is_set(registration)) {  
    if(filter(entered_login, VALIDATE_LOGIN) AND // проверка корректности логина  
    filter(entered_email, VALIDATE_EMAIL) AND // проверка корректности почты  
    filter(entered_pass, VALIDATE_PASS)) { // проверка корректности пароля  
        // запрос к БД на совпадение с указанными логином и почтой  
        result = mysql_query(SELECT rows FROM users WHERE  
            login = entered_login AND email = entered_email);  
        if(result) { // если найдены совпадения  
            message('Данный логин или почта уже заняты');  
        } else {  
            // запрос на добавление нового пользователя в БД  
            mysql_query(INSERT INTO users (  
                login = entered_login,  
                email = entered_email,  
                pass = entered_pass);  
            send_registration_mail(entered_email); // отправить письмо пользователю  
            message('На указанный вами электронный адрес выслано письмо  
                для активации учетной записи');  
        }  
    } else { message('Проверьте введенные данные'); }  
}  
// если пользователь перешел на страницу активации по уникальной ссылке  
if(activate_page AND user_hash_link) {  
    // запрос к БД на изменение статуса пользователя  
    mysql_query(UPDATE users SET status = enable WHERE hash = user_hash_link);  
    message('Ваша учетная запись активирована');  
}
```

### Процесс 3 «Авторизация»

ВХОДНЫЕ ДАННЫЕ: login / email, password

ВЫХОДНЫЕ ДАННЫЕ: сообщение о статусе авторизации

```
if(is_set(authentication)) { // если пользователь ввел данные для авторизации  
    // запрос к БД на совпадение с указанными логином и паролем  
    result = mysql_query(SELECT rows FROM users WHERE  
        email = entered_email AND pass = entered_pass);  
    if(result) { // если найдены совпадения  
        authentication = true; // записываем в cookie, что пользователь авторизован  
    } else {  
        message('Совпадений не найдено.');
```

#### *Процесс 4 «Изменить личные данные»*

ВХОДНЫЕ ДАННЫЕ: событие «click»; имя, дата рождения, пол, email, телефон, адрес, аватар, дополнительная информация  
ВЫХОДНЫЕ ДАННЫЕ: новые данные в профиле

```
if(is_set(entered_data)) { // если пользователь ввел новые данные
    if(filter(entered_data, VALIDATE_DATA)) { // и они корректны
        // запрос к БД на изменение персональных данных
        mysql_query(UPDATE users SET
            name      = entered_data[name],
            birth_day  = entered_data[birth_day],
            sex        = entered_data[sex],
            email      = entered_data[email],
            phone      = entered_data[phone],
            address     = entered_data[address],
            avatar      = file_name WHERE id = user_id),
            add_info    = entered_data[add_info];
        } else { message('Проверьте корректность введенных данных'); }
    }
}
```

#### *Процесс 8 «Совершить покупку (заказ)»*

##### *Подпроцесс 8.1 «Быстра покупка»*

##### *Подпроцесс 8.2 «Покупка через личный кабинет»*

ВХОДНЫЕ ДАННЫЕ: событие «click»; ID товаров

ВЫХОДНЫЕ ДАННЫЕ: данные клиента

```
if(authentication) { // если пользователь авторизован
    // обработать заказ с использованием персональных данных пользователя
    order_processing(order = mysql_query(SELECT data FROM customer AND item));
} else {
    // если пользователь не авторизован,
    // но он ввел персональные данные для быстрой покупки
    if(is_set(entered_data)) {
        if(filter(entered_data, VALIDATE_DATA)) { // и данные корректны
            item_name = mysql_query(SELECT item_name FROM items WHERE
            item_id = shopping_cart(item_id));
            create_array order(
                customer,
                address,
                phone,
                email,
                comment,
                item_id,
                item_name);
            order_processing(order); // обработка заказа
        } else { message('Проверьте корректность введенных данных'); }
    }
}
```

## Процесс 9 «Обработка заказа»

### Подпроцесс 9.1 «Составление формы заказа»

### Подпроцесс 9.2 «Добавление заказа»

ВХОДНЫЕ ДАННЫЕ: ID товаров, наименование, цена; данные клиента  
ВЫХОДНЫЕ ДАННЫЕ: сообщение о статусе заказа; запись в журнал ошибки (if exception)

```
order_processing(order) {  
    // запрос к БД на добавление нового заказа  
    mysql_query(INSERT INTO orders (  
        date          = get_date(),  
        customer      = order[customer],  
        address       = order[address],  
        phone         = order[phone],  
        email         = order[email],  
        comment       = order[comment],  
        item_id       = order[item_id],  
        item_name     = order[item_name],  
        status        = "pending");  
    unset(shopping_cart); // удалить данные из корзины  
    redirect(home_page); // перенаправить пользователя на домашнюю страницу  
    message('Ваш заказ принят, ожидайте звонка');  
}  
if(exception) { // если произошло исключение (ошибка)  
    exception_handling; // обработать исключение (ошибку)  
    if(error_code) { // если есть код ошибки  
        file_recording(error_log, error_code, add_info); // записать в журнал ошибок  
    }  
}
```

## Выводы по работе

В результате выполнения данной лабораторной работы была изучена методология и один из основных инструментов графического структурного анализа — диаграммы потоков данных. Также рассмотрены способы уточнения структурной модели системы (в данном случае диаграммы потоков данных) и получены навыки составления спецификации процессов.

Разработана структурная модель системы «Интернет-магазин» на основе диаграмм потоков данных. Данная модель не описывает весь функционал анализируемой системы, в модели рассматривается только клиентская часть и не затрагивается административная (панель администратора). Также помимо «менеджера (администратора сайта)» за пределы модели вынесены такие сущности, как «склад товаров» и «рекламный отдел», соответственно все процессы, связанные с данными сущностями, в модели не представлены. Для полной декомпозиции контекстной диаграммы потребуется 4 отдельные, связанные между собой диаграммы потоков данных 1-го уровня, что выходит за рамки задания.

Спецификация процессов выполнена двумя способами: на структурированном естественном языке и псевдокоде подобном PНР, стиль псевдокода охарактеризован спецификой информационной системы. Обращения к БД показаны в упрощенной (неточной) форме SQL запросов. В комментариях к псевдокоду между пользователем и клиентом не делается различий и они являются взаимозаменяемыми словами синонимами.

### **1.7 Контрольные вопросы**

- 1) Каково назначение диаграммы потоков данных?
- 2) Что представляет собой внешняя сущность?
- 3) Что по отношению к системе определяют сущности?
- 4) Что представляет собой процесс?
- 5) Является ли процесс атомарным (неделимым) объектом?
- 6) Что представляет собой поток данных?
- 7) Является ли поток данных атомарным (неделимым) объектом?
- 8) Что представляет собой накопитель данных?
- 9) Для чего нужна контекстная диаграмма?
- 10) На каком уровне мы можем прекратить декомпозицию модели?
- 11) Чем отличаются нотации Гейна-Сарсона и Йордана-Коуда?
- 12) Насколько диаграмма потоков данных должна быть детализирована?
- 13) Что представляет собой спецификация процессов?
- 14) Каково назначение спецификации процессов в структурной модели системы?
- 15) Какие способы представления спецификации процессов допустимы?
- 16) Приведите пример спецификации процесса.
- 17) Какой способ представления спецификации предпочтителен?
- 18) Когда (при каких условиях) составляется спецификация процесса?



## 2. ЛАБОРАТОРНАЯ РАБОТА

### «РАЗРАБОТКА СПЕЦИФИКАЦИИ ПРОГРАММНЫХ ТРЕБОВАНИЙ»

#### 2.1 Цель работы

Целью данной работы является изучение способов описания законченного поведения проектируемой программной системы.

#### 2.2 Задание на лабораторную работу

Разработать спецификацию требований программного обеспечения к проектируемой программной системе или её функционально законченной части.

Спецификация требований программного обеспечения должна соответствовать структуре, представленной в теоретическом материале методических указаний к лабораторной работе.

В описании программного обеспечения необходимо обозначить границы проектируемой системы через модель предметной области.

Лабораторная работа будет оцениваться в соответствии со следующими критериями:

	Удовл.	Хорошо	Отлично
Минимальное количество функциональных требований	10	12	15
Минимальное суммарное количество нефункциональных требований (к удобству использования, надежности, безопасности, etc.)	5	8	10
Количество (текстовых) описаний вариантов использования системы	3	5	7
Диаграмма прецедентов (use-case diagram) для действующих лиц системы <sup>3</sup>	–	–	+

#### 2.3 Порядок выполнения работы

1. Изучить теоретический материал, изложенный в подразделе 2.4;
2. Разработать с спецификацию требований программного обеспечения;
3. Ознакомиться с требованиями по содержанию отчета в подразделе 2.5;
4. Написать отчет о работе.

Для выполнения лабораторной работы в части составления диаграммы прецедентов можно воспользоваться любой средой моделирования или CASE-средством, которое поддерживает нотацию UML 2х. Рекомендуется использовать бесплатное для некоммерческого использования CASE-средство [Software Ideas Modeler](#)

---

<sup>3</sup> Диаграмма прецедентов должна описывать всех пользователей системы, представленных в подразделе 1.3, и соответствующие им функциональные требования из подраздела 1.2.

## 2.4 Теоретический материал

### 2.4.1 Спецификация требований программного обеспечения

Процесс проектирования архитектуры программного обеспечения включает в себя сбор требований клиентов, их анализ и создание проекта для компонента программного обеспечения в соответствии с требованиями. Одним из инструментов представления требований к программному обеспечению является спецификация требований программного обеспечения (Software Requirements Specification, SRS).

SRS представляет собой полное и четкое описание разрабатываемого программного обеспечения.

Основными вопросами, которые должны рассматривать составитель (-ли) SRS, являются:

- а) Функциональные возможности. Каковы предполагаемые функции программного обеспечения?
- б) Внешние интерфейсы. Как программное обеспечение взаимодействует с пользователями, аппаратными средствами системы, другими аппаратными средствами и другим программным обеспечением?
- в) Рабочие характеристики. Каково быстродействие, доступность, время отклика, время восстановления различных функций программного обеспечения, etc.?
- г) Атрибуты. Каковы мобильность, правильность, удобство сопровождения, защищенность программного обеспечения и другие критерии?
- д) Проектные ограничения, налагаемые на реализацию изделия. Существуют ли требуемые стандарты на эффективном языке реализации, политика по сохранению целостности баз данных, ограничения ресурсов, операционная среда(-ы), etc.?

Так как SRS играет определенную роль в процессе разработки программного обеспечения, составителю (-ям) SRS следует проявлять осторожность, чтобы не выйти за пределы этой роли. Это означает, что SRS:

- а) Должна правильно определять все требования к программному обеспечению. Причиной существования какого-либо требования к программному обеспечению может являться характер решаемой задачи или особая характеристика проекта.
- б) Не должна описывать детали разработки или реализации. Они должны быть описаны на этапе разработки проекта.
- в) Не должна налагать дополнительные ограничения на программное обеспечение. Эти ограничения надлежащим образом определяются в других документах, таких как план обеспечения качества программных средств.

SRS является корректной только в том случае, если каждое требование, изложенное в ней, является требованием, которому должно удовлетворять программное обеспечение.

SRS является однозначной, если каждое изложенное в ней требование может интерпретироваться однозначно. Как минимум, для этого требуется, чтобы каждая характеристика конечного продукта была описана с использованием одного уникального термина. В тех случаях, когда термин, используемый в специфическом контексте, может иметь множественные значения, этот термин должен быть включен в глоссарий, в котором его значение описывается более конкретно.

#### Основное назначение SRS:

- Получение точной оценки стоимости, рисков и затрат времени;
- Помогает клиенту четко сформировать собственное видение проекта;
- Предоставляет возможность одинакового представления о продукте Заказчику и Исполнителю;
- Помогает выявить оптимальный набор функций;
- Служит основой для формирования другой технической документации;
- Помогает оптимизировать процесс разработки за счет минимизации затрат времени и ресурсов;
- Помогает исключить дублирования задач;
- Позволяет структурировать проблемы, что упрощает и ускоряет процесс их решения;
- Помогает понять, какие именно результаты считаются оптимальными при тестировании.

#### Проблемы, возникающие при разработке проекта без SRS:

- Невозможно получить точную оценку стоимости, рисков и затрат времени;
- Заказчик и Исполнитель могут иметь абсолютно разное представление о продукте;
- Разработчики не могут быть уверены, что создаваемое ими программное обеспечение полностью соответствует требованиям заказчика;
- Усложняется процесс написания руководства для пользователей;
- Увеличивается вероятность того, что потребуется переработать некоторые части проекта заново.

Рекомендации к структуре и методам описания SRS содержатся в стандарте IEEE 830.

Есть ряд шаблонов SRS для работы над большими новыми проектами, для маленьких веб-сайтов и для проектов доработки. Ниже приведен упрощенный вариант (шаблон) SRS, который не является шаблоном какого-либо определенного класса проектов и представляет собой исключительно учебную формацию, в соответствии с которой необходимо выполнить лабораторную работу.

Спецификация требований к программному обеспечению  
«Наименование программного обеспечения/системы»  
Номер версии

Дата создания документа

История изменений

Дата	Версия	Описание	Автор

**ВВЕДЕНИЕ**

Введение к документу «Спецификация требований к программному обеспечению» должно дать его полный обзор. В него необходимо включить назначение, область применения, определения, акронимы, сокращения, ссылки и обзор структуры рассматриваемой спецификации.

Назначение:

В назначении необходимо указать цели создания данного документа и определить аудиторию, для которой он предназначен.

Область применения:

Данный подраздел должен идентифицировать программное обеспечение, которое будет разработано, и обозначить наименование (например, «Система информационной поддержки научных конференций», «Серверная СУБД» etc.). Объяснить, какие действия должны осуществляться данным программным обеспечением и, если необходимо, какие действия выходят за рамки системы или являются теми требованиями, которые система не должна выполнять. Описать применения определяемого программного обеспечения, включая важные преимущества, объекты и цели.

Определения, акронимы (аббревиатуры) и сокращения:

Данный подраздел должен обеспечить определения всех терминов, акронимов (аббревиатур) и сокращений, которые необходимы для интерпретации SRS должным образом. Любые непонятные или неоднозначные термины в спецификации не должны быть определены через другие непонятные и неоднозначные термины.

Ссылки:

Данный подраздел должен обеспечить полный список всех документов, упомянутых в каком либо месте в SRS, идентифицировать каждый документ наименованием, номером (если периодическое издание), датой, издающей организацией, определить источники, в которых ссылки можно получить.

## Краткий обзор:

Данный подраздел должен описать, что содержит остальная часть SRS и объяснить, как SRS структурирована.

### 1. Общее описание

Данный раздел SRS должен описать общие факторы, которые затрагивают систему и требования к ней. Этот раздел не определяет специфические требования. Вместо этого, он обеспечивает основу для их подробного определения в разделе 2, и делает их более легкими для понимания.

#### 1.1 Описание программного обеспечения

В данном подразделе SRS программное обеспечение необходимо рассмотреть как структуру взаимосвязанных компонентов и с точки зрения связанных с ним систем. Если программное обеспечение независимо и полностью автономно, то это должно быть четко обозначено. Если SRS определяет программное обеспечение, которое является компонентом большей системы, то этот подраздел должен связать требования основной системы с функциональными возможностями данного программного обеспечения.

В рамках ЛР в данном разделе необходимо обозначить границы проектируемой системы через модель предметной области (*см. пример выполнения*).

#### 1.2 Функции программного обеспечения

Данный подраздел SRS должен содержать резюме основных функций, которые программное обеспечение выполняет.

Функции должны быть организованы в группы, которые делают список функций понятными клиенту или кому-либо, кто читает документ впервые. Могут использоваться текстовые или графические методы (например, диаграммы последовательности), чтобы показать различные функции и их отношения.

#### 1.3 Характеристики пользователей

Данный подраздел SRS должен описывать общие характеристики пользователей, для которых предназначено программное обеспечение, включая образовательный уровень, опыт, и техническую квалификацию. Не следует использовать этот раздел для того, чтобы сформулировать детальные требования, здесь скорее необходимо объяснить причины, по которым эти требования появятся позже в разделе 2 SRS.

Выделить классы пользователей системы можно, сгруппировав их по следующим признакам:

- По привилегиям доступа и уровню безопасности (рядовой пользователь, пользователь-гость, администратор, модератор);
- По используемым функциям;
- По частоте использования продукта;
- По опыту в предметной области;
- По используемой платформе (ноутбуки, планшеты, смартфоны, ...);
- По виду доступа к системе — прямой или косвенный;
- etc.

Косвенный вид доступа подразумевает, что некоторые пользователи обращаются к системе не напрямую, а работают с её данными и сервисами через другие приложения или отчёты. Например, это могут быть программные агенты, которые имитируют живого пользователя и выполняют проверку веб-сайта на предмет уязвимостей или генерируют спам.

Для таких видов пользователей также создают специальные требования с целью предотвращения их нежелательной деятельности по отношению к системе.

#### 1.4 Интерфейсы пользователя

Необходимо определить логические характеристики каждого интерфейса между изделием программного обеспечения и пользователями. В описание необходимо включить те характеристики конфигурации (например, требуемые форматы экрана, расположение страниц или окон, содержание любых сообщений или меню, наличие программируемых функциональных клавиш), которые требуются для выполнения требований программного обеспечения.

В данном разделе также необходимо определить нормальные и специальные действия, необходимые пользователям.

#### 1.5 Ограничения проекта

Данный подраздел SRS должен обеспечить общее описание любых других факторов, которые ограничивают выбор разработчика. Они включают:

- a) Требования к языку программирования, базе данных;
- b) Стандарты кодирования;
- c) Стандарты обмена данными;
- d) Интерфейсы с другими приложениями;
- e) Требования надежности;
- f) Соображения безопасности и секретности etc.

#### 1.6 Распределение требований

Этот подраздел SRS должен определить требования, которые могут быть отсрочены до будущих версий системы (в данной работе можно не рассматривать).

### 2. Детальные требования

Данный раздел SRS должен содержать все требования программного обеспечения с достаточной степенью детализации, чтобы позволить разработчикам спроектировать и специалистам, отвечающим за тестирование, проверить систему, удовлетворяющую этим требованиям. Каждое требование в данном разделе должно обеспечивать взаимодействие с пользователями, операторами или другими внешними системами. Эти требования должны включать, как минимум, описание всех входов системы, всех выходов системы и всех функций, которые выполняются системой в ответ на вход или для получения результатов на выходе.

В настоящей лабораторной работе данный раздел носит упрощенный характер.

#### 2.1 Требования к базе данных

В настоящей ЛР не рассматривается, необходимо привести ссылку на отчет о лабораторной работе №3.

## 2.2 Варианты использования Системы

В данном подразделе необходимо привести перечень функций системы в виде вариантов использования для различных категорий пользователей и дать их детальное описание.

Для оценки «отлично» в данном подразделе необходимо привести диаграммы прецедентов (use case diagram) для демонстрации пользователей системы (подраздел 1.3) и соответствующих им функциональных возможностей (подраздел 1.2)

Описание вариантов использования необходимо выполнить в соответствии с шаблоном, представленным ниже.

ID и наименование:	У каждого варианта использования должен быть свой уникальный идентификатор, в качестве которого может выступать порядковый номер или условное обозначение, например, UC-01 (от Use Case 01)
Требование:	Здесь необходимо привести идентификатор функционального требования, которому соответствует данный вариант использования
Действующее лицо:	Тот, кто инициирует выполнение варианта использования
Описание:	Краткое текстовое описание цели варианта использования
Условие-триггер:	Событие, которое информирует о желании пользователя выполнить данный вариант использования
Предусловия: (ноль или больше)	Определяют, что должно присутствовать в системе до выполнения варианта использования. Система должна иметь возможность проверить все предварительные условия, чтобы определить, возможно ли выполнять вариант использования
Постусловия: (одно или больше)	Описывают состояние системы после успешного выполнения варианта использования. Выходные условия могут описывать: <ul style="list-style-type: none"><li>• Что-то, что пользователь может наблюдать сам (сообщение, что он теперь является участником конференции);</li><li>• Физические результаты (кофейный автомат выдал приготовленный кофе);</li><li>• Изменение внутреннего состояния системы (транзакции в базе данных).</li></ul>
Основной поток:	Один сценарий развития варианта использования, который приводит к успешному выполнению задания. С точки зрения пользователя, это то, что должно происходить по умолчанию.
Альтернативный поток:	Представляют менее популярные или менее приоритетные вариации самой задачи или способа её выполнения, они также могут привести к успешному выполнению задания и удовлетворяют выходным условиям варианта использования
Исключения:	Условия, препятствующие успешному выполнению варианта использования, описывают ожидаемое ошибочное условие, которое может сложиться во время выполнения варианта использования, и как его обрабатывать

## **2.5 Содержание отчета**

Отчет о работе должен содержать:

1. Титульный лист
2. Цель работы
3. Задание на лабораторную работу и вариант
4. Спецификация требований к программному обеспечению
  - 4.1 ВВЕДЕНИЕ
  - 4.2 Общее описание
    - 4.2.1 Описание программного обеспечения (описание предметной области)
    - 4.2.2 Функции программного обеспечения
    - 4.2.3 Характеристики пользователей
    - 4.2.4 Интерфейсы пользователя
    - 4.2.5 Ограничения проекта
  - 4.3 Детальные требования
    - 4.3.1 Логические требования к базе данных (ссылка на отчет ЛР 3)
    - 4.3.2 Варианты использования Системы
5. Выводы по работе
6. Используемые источники (2-4 источника)



## 2.6 Пример выполнения работы

### История изменений

Дата	Версия	Описание	Автор
24.11.2018	1.0	В данной версии документа рассматриваются основные функциональные требования и ограничения к проектируемой системе	Е. В. Павлов

### ВВЕДЕНИЕ

#### Назначение:

Настоящий документ направлен на описание требований и ограничений, налагаемых на проектируемую программную систему.

#### Область применения:

Проектируемая Система информационной поддержки научных конференций предназначена для информационного сопровождения и обеспечения взаимодействия всех участников научных и образовательных мероприятий. Данная Система предполагает содействие в решении институциональной и межличностной разобщенности учёных, а также в создании дополнительных возможностей для контактов и сотрудничества учёных с иностранными коллегами и потенциальными спонсорами научной деятельности.

Все указанные цели и задачи Системы реализуются посредством веб-сайта и на текущем этапе проектирования не предполагают наличие других архитектурных решений, включая мобильные приложения.

#### Сокращения и обозначения:

Система – Система информационной поддержки научных конференций;

Научная конференция – форма организации научной деятельности, при которой исследователи (не обязательно учёные или студенты) представляют и обсуждают свои работы.

Базовый функционал (Системы) – возможность входа (в личный кабинет) и регистрации на сайте, просмотр общих информационных страниц («Список конференций», «Архив», «О системе», «Новости», etc.). При этом вместо функциональных кнопок для пользователя отображается оповещение, что для дальнейших действий необходимо войти в систему.

ID – Идентификатор;

FUN – Сокращение от Functional Requirement (функциональное требование);

USA – Сокращение от Usability Requirement (требование к удобству использования);

REL – Сокращение от Reliability Requirement (требование к надежности);

SUP – Сокращение от Supportability Requirement (требование к обслуживанию);

SEC – Сокращение от Security Requirement (требование к безопасности).

В настоящем документе содержатся ссылки на следующие документы:

[1] Отчет о ЛР №1 «Структурный анализ системы. Разработка диаграммы потоков данных и составление спецификации процессов»

[2] Отчет о ЛР №3 «Разработка требований к базе данных. Построение логической модели данных»

Структура документа:

В первой главе описаны функциональные и нефункциональные требования к проектируемой Системе. Во второй главе рассмотрены варианты использования Системы.

### 3.1 Общее описание

#### 3.1.1 Описание программного обеспечения

Проектируемая Система представляет собой главным образом веб-сайт; целевая аудитория и основные потребности пользователей (функционал Системы) показаны в виде структурной модели [1].

Поскольку модель и архитектура приложения взаимно определяют друг друга, то в настоящем документе границы Системы будут определены посредством описания модели предметной области, основными объектами которой являются:

- конференции;
- тематика (исследования);
- участники конференции (за каждым участником должен быть закреплен определенный статус, в качестве кого он принимает участие на конференции, данный атрибут можно рассматривать отдельным объектом);
- создатель конференции;
- администратор;
- заявки (как на участие, так и на отказ).

Данная система имеет основную ориентацию на студентов высших учебных заведений, поэтому дополнительно могут быть выделены следующие объекты:

- студенты;
- преподаватели;
- университет.

Система должна быть реализована по следующим правилам:

- Конференции могут проводиться в разных университетах (городах);
- Для каждой конференции задана определенная тематика (участники с неподходящими темами докладов редактируются организатором конференции);
- У каждой конференции в Системе может быть только один организатор;
- Права для редактирования данных и одобрения заявок имеет только администратор.

### 3.1.2 Функции программного обеспечения

Функциональные требования охватывают предполагаемое поведение системы, определяя действия, которые система способна выполнять.

Перечень функциональных требований представлен в таблице 1. Каждое функциональное требование имеет уникальный идентификатор.

Таблица 1 – Функциональные требования.

FUN-ID	Описание
FUN-01	Система должна обеспечивать регистрацию пользователей (создание личного кабинета)
FUN-02	Система должна позволять редактировать личную информацию пользователей (через личный кабинет)
FUN-03	Система должна обеспечить добавление новой конференции (мероприятия)
FUN-04	Система должна предоставлять просмотр подробного описания конференции
FUN-05	Система должна обеспечивать участие в конференции (заполнение формы участника)
FUN-06	Система должна предоставлять возможность покинуть конференцию
FUN-07	Система должна предоставить возможность организатору конференции исключать участников из конференции
FUN-08	Система должна обеспечить возможность удаления конференции (после одобрения заявки удаление конференции возможно только при согласовании данного действия с администратором)
FUN-09	Система должна обеспечивать добавление / удаление материалов конференции
FUN-10	Система должна сохранять в архив все материалы по конференции (мероприятию), после того как конференция (мероприятие) была проведена
FUN-11	Система должна предоставить возможность поиска события (конференции) с использованием фильтра: <ul style="list-style-type: none"><li>- полное наименование или частичное совпадение отдельных слов;</li><li>- тип мероприятия (конференции);</li><li>- страна проведения;</li><li>- город проведения;</li><li>- сроки проведения;</li><li>- статус заявок (открыт / закрыт).</li></ul>
FUN-12	Система должна предоставлять просмотр каталога мероприятий (конференций) с указанием способа сортировки: <ul style="list-style-type: none"><li>- научные конференции;</li><li>- тематики;</li><li>- города;</li><li>- год.</li></ul>

FUN-13	Система должна обеспечивать возможность просмотра списка всех предстоящих мероприятий
FUN-14	Система должна обеспечивать возможность просмотра справки с актуальными вопросами по сайту и участию в мероприятиях (конференциях)
FUN-15	Система должна осуществлять почтовую рассылку информационных сообщений (или оповещений) для пользователей
FUN-16	В системе должен быть реализован интерфейс панели администратора с возможностью редактирования всех сущностей Системы
FUN-17	Система должна отображать заявки на создание конференций в панели администратора (по умолчанию заявки должны быть отсортированы по статусу)
FUN-18	Система должна предоставлять возможность отклонить или одобрить конференцию в панели администратора

Нефункциональные требования к Системе представлены в таблицах 2-5.

Таблица 2 – Требования к удобству использования

USA-ID	Описание
USA-01	Система должна иметь интуитивно понятный интерфейс
USA-02	Система должна быть легка в освоении
USA-03	Система должна облегчить исправление ошибок, в том числе с помощью вывода подсказок при вводе (заполнение форм, поиск) и предупреждений об ошибках в формах
USA-04	Система должна обеспечивать поддержку адаптивности экрана
USA-05	Система должна обеспечивать быстрое выполнение скриптов и незаметную (для пользователя) загрузку страницы

Таблица 3 – Требования к надежности

REL-ID	Описание
REL-01	Система должна допускать возможные ошибки оператора, которые не должны нарушать функциональную надежность самой Системы

Таблица 4 – Требования к безопасности

SEC-ID	Описание
SEC-01	Система должна обеспечить разграничение прав доступа: пользователи должны иметь возможность редактирования сущностей системы, которые имеют непосредственное отношение только к их категории (участники конференции, организаторы); администратор сайта имеет права для редактирования всех сущностей системы.
SEC-02	Система должна хранить пароли пользователей в защищенном виде

Таблица 5 – Требования к обслуживанию

SUP-ID	Описание
SUP-01	Система должна быть спроектирована так, чтобы изменения функциональных возможностей и графической составляющей Системы выполнялись с минимальным изменением программного кода
SUP-02	Система должна предоставлять решения или правила относительно проблем кодирования, таких как поддержка различных наборов символов, правила усечения имён, совпадение имён в случае орфографической ошибки, ...

### 3.1.3 Характеристики пользователей

Общие характеристики пользователей, для которых предназначена Система, представлены в таблице 6.

Таблица 6 – Характеристики пользователей системы.

Наименование	Описание
Пользователь	Посетитель сайта, имеющий доступ к базовому функционалу системы
Участник конференции	Зарегистрированный пользователь, который заполнил форму участника конференции
Организатор конференции	Зарегистрированный пользователь, который заполнил форму создателя конференции. Осуществляет контроль над участниками своей конференции. В его полномочия входит удаление участников с неподходящими докладами.
Администратор	Владелец Системы или сотрудник владельца Системы, который имеет возможность управлять всеми сущностями Системы.

### 3.1.4 Интерфейсы пользователя



Рисунок 1 – Главная (домашняя) страница сайта



Рисунок 2 – Страница новостей

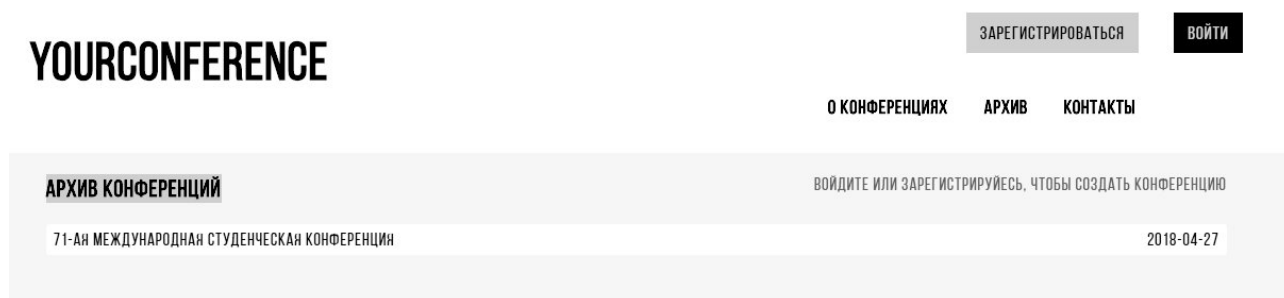


Рисунок 3 – Страница архива конференций

### 3.1.5 Ограничения проекта

- Все программные коды управляющей логики системы должны быть написаны на языке PHP;
- Весь код PHP должен соответствовать стандарту PHP 7.0;
- Весь код PHP должен быть написан в соответствии со стандартом PSR-2;
- Весь код HTML должен соответствовать стандарту HTML 5.0;
- Язык стилей должен быть не ниже CSS 3;
- Система должна использовать СУБД MySQL 5.7;
- HTML и CSS-шаблоны оформления не ниже Bootstrap 4.

## 3.2 Детальные требования

### 3.2.1 Логические требования к базе данных

Требования к данным сформулированы в документе [2].

### 3.2.2 Варианты использования Системы

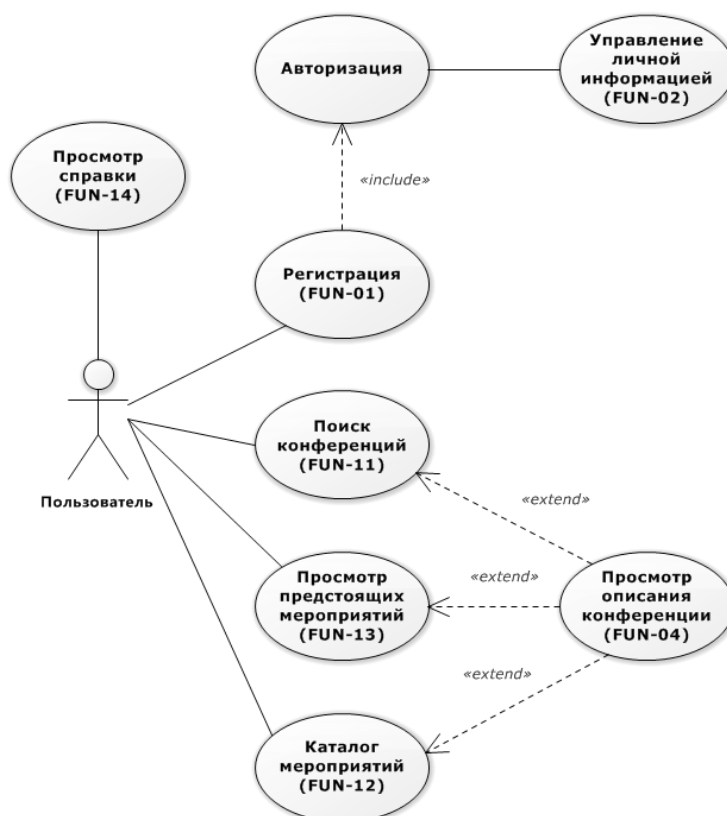


Рисунок 4 – Диаграмма прецедентов для пользователя

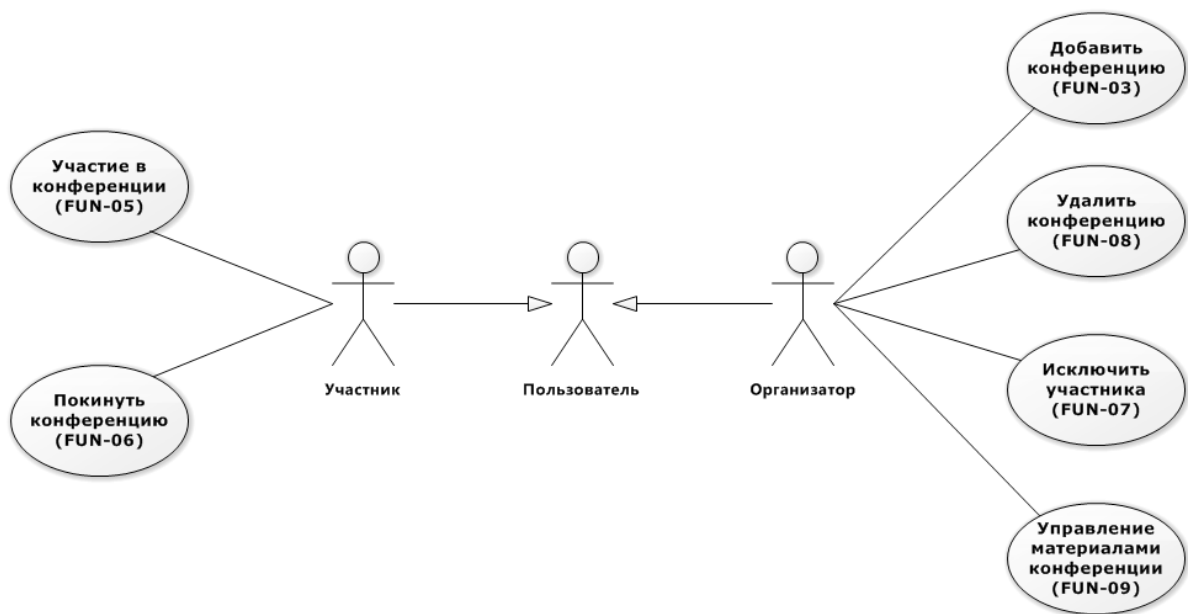


Рисунок 5 – Диаграмма прецедентов для участника и организатора конференции

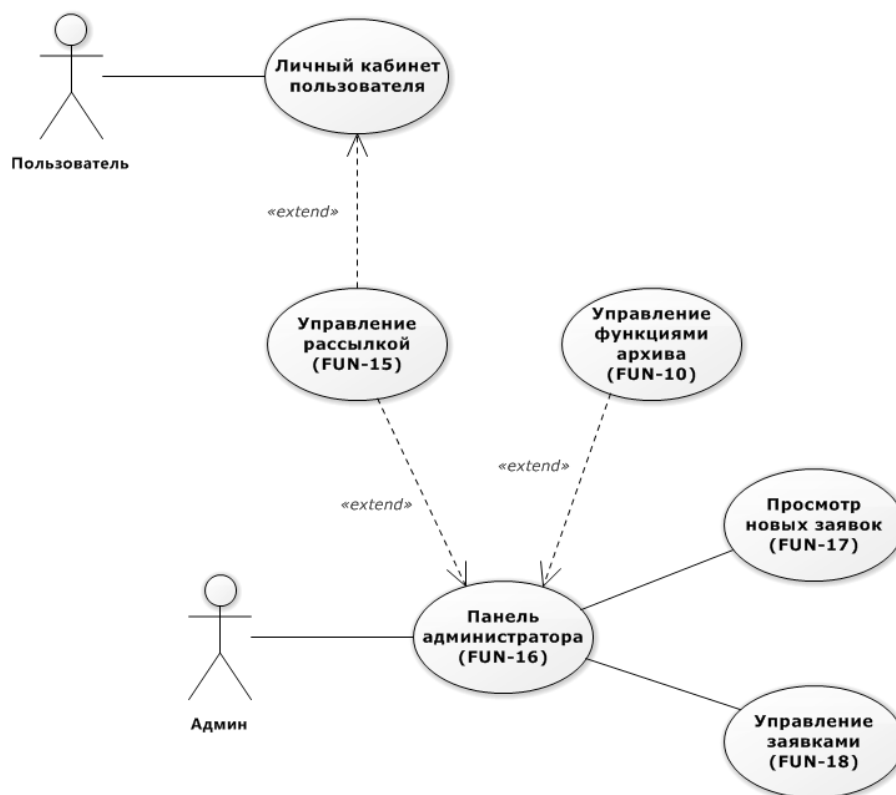


Рисунок 6 – Диаграмма прецедентов для администратора



Описание вариантов использования:

ID и наименование:	UC-01 Выход из участия в конференции
Требование:	FUN-06
Действующее лицо:	Пользователь, который участвует в конференции
Описание:	Пользователь, который ранее подал заявку на участие в конференции, решает покинуть список участников
Условие-триггер:	Пользователь указывает, что хочет покинуть конференцию на вкладке «Мои конференции»
Предусловия:	1. Пользователь аутентифицирован; 2. Конференция, которую пользователь желает покинуть, активна.
Постусловия:	1. Система исключает пользователя из списка участников конференции и сохраняет его идентификатор в списке покинувших конференцию; 2. Происходит переадресация пользователя на вкладку «Мои конференции» с обновленным списком и информационным сообщением, что пользователь покинул конференцию.
Основной поток:	1. Пользователь нажимает «Покинуть конференцию» во вкладке «Мои конференции»; 2. Система просит подтвердить действие пользователя; 3. После подтверждения, система отправляет запрос на исключение пользователя из списка участников.
Альтернативный поток:	Не предусмотрено
Исключения:	Не определены

ID и наименование:	UC-02 Удаление участника конференции
Требование:	FUN-07
Действующее лицо:	Пользователь, который является организатором конференции
Описание:	Пользователь, который организовал конференцию, решает исключить одного или более её участников
Условие-триггер:	Организатор конференции указывает, что хочет уменьшить или очистить список участников конференции
Предусловия:	1. Пользователь (организатор) аутентифицирован; 2. Конференция организатора активна; 3. Количество участвующих в конференции пользователей больше одного (не считая организатора).
Постусловия:	1. Пользователь (или пользователи) исключается из списка участников конференции; 2. Список участников конференции будет обновлен с указанием информационного сообщения об исключенном участнике или участниках.
Основной поток:	1. Организатор конференции в списке участников отмечает одного или более пользователей через check box; 2. Нажимает кнопку «Исключить»; 3. Система просит подтвердить действие пользователя; 4. После подтверждения, система отправляет запрос на исключение участников из списка.
Альтернативный поток:	Очистка списка участников 1. Организатор конференции в списке участников выбирает «Очистить список участников»; 2. Система просит подтвердить действие пользователя; 3. После подтверждения, система отправляет запрос на полную очистку списка участников конференции.
Исключения:	Нельзя исключить участника конференции 1. При попытке исключить участника или очистить список участников появляется информационное сообщение о невозможности выполнить данную операцию (из-за привилегированного статуса участника); 2. Система предлагает направить запрос администратору на выполнение данного действия с указанием причины.

ID и наименование:	UC-03 Удаление конференции
Требование:	FUN-08
Действующее лицо:	Пользователь, который является организатором конференции
Описание:	Пользователь, который организовал конференцию, решает удалить данную конференцию из списка активных мероприятий
Условие-триггер:	Организатор конференции указывает, что хочет удалить созданную им конференцию
Предусловия:	1. Пользователь (организатор) аутентифицирован; 2. Конференция организатора активна.
Постусловия:	1. Запрос отправлен на рассмотрение администратору и сохранён в журнале событий системы; 2. После удовлетворения запроса, конференция удаляется из списка активных и сохраняется в архив со статусом «отменена организатором».
Основной поток:	1. Пользователь в списке своих конференций выбирает «Удаление конференции»; 2. Система просит подтвердить действие пользователя; 3. После подтверждения, система представляет форму запроса на удаление конференции, где предлагает указывает причину удаления и повторно подтвердить действие; 4. После заполнения формы и подтверждения удаления, запрос направляется администратору.
Альтернативный поток:	Не предусмотрено
Исключения:	Не определены

## Выводы по работе

В результате выполнения данной лабораторной работы был изучен один из способов описания законченного поведения проектируемой системы. Получены навыки составления спецификации требований к программному обеспечению (SRS). Структура разработанной SRS соответствует стандарту IEEE 830.

В настоящей работе можно выделить ряд недостатков:

- В подразделе 3.1.2 проработан не исчерпывающий список функциональных требований к Системе. Представленные функциональные требования охватывают основной функционал, который необходимо представить на оценку заказчику (преподавателю) в определенный срок, что накладывает соответствующие ограничения на проработку функций Системы. Однако список функциональных требований отвечает достаточным требованиям текущей проектной документации;
- В подразделе 3.1.4 представлены примеры макетов пользовательского интерфейса без указания конкретных характеристик (требований), таких как размер и конфигурация экрана или ограничения разрешения, стандарты отображения и текста сообщений, стандарты проверки данных (ограничения на вводимые значения) и другие. Данный подход может привести к тому, что визуальный дизайн будет определять требования, что часто ведёт к пропуску функций системы. Указанный недостаток работы также характеризуется достаточными требованиями к текущей проектной документации.
- В подразделе 3.2.2 в описании варианта использования UC-02 указан привилегированный статус участника конференции, однако данная категория пользователей не указана в таблице характеристик пользователей системы и в тексте спецификации не приведены разъяснения на счет характера данных привилегий. Приведенный недостаток связан с планами следующего выпуска и указанный класс пользователей не реализуется в текущем выпуске системы.

## 2.7 Контрольные вопросы

- 1) Что представляет собой SRS?
- 2) Каково назначение SRS?
- 3) Каковы основные правила составления SRS?
- 4) Перечислите основные преимущества SRS;
- 5) Перечислите основные структурные элементы SRS;
- 6) Что представляют собой функциональные и нефункциональные требования?
- 7) Опишите такой тип требований как ограничения проекта;
- 8) Какая информация указывается в описании интерфейсов пользователя?
- 9) Что представляет собой вариант использования? Приведите пример;
- 10) Какие проблемы при разработке могут возникнуть без использования SRS?

### 3. ЛАБОРАТОРНАЯ РАБОТА

#### «ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ К ДАННЫМ. ЛОГИЧЕСКАЯ МОДЕЛЬ И СЛОВАРЬ ДАННЫХ»

#### 3.1 Цель работы

Целью данной работы является изучение способов определения требований к данным и построение логической модели отношений данных.

#### 3.2 Задание на лабораторную работу

Разработать логическую модель отношений данных проектируемой системы или её функционально законченной части на основе диаграммы «сущность-связь» (entity-relationship diagram).

Диаграмма должна быть исполнена в графической нотации, представленной в методических указаниях к работе. Все отношения «многие-ко-многим» должны быть реструктурированы, а сама диаграмма должна отвечать 3-й нормальной форме.

Информация об используемых в системе сущностях данных должна быть представлена в виде словаря данных (data dictionary).

В диаграмме «сущность-связь» и в словаре данных должны быть явно выделены обязательные (*NOT NULL*) и необязательные (*NULL*) атрибуты (элементы данных).

Лабораторная работа будет оцениваться в соответствии со следующими критериями:

	Удовл.	Хорошо	Отлично
Минимальное количество сущностей на диаграмме «сущность-связь» (связующие таблицы при реструктуризации отношений «многие-ко-многим» в это число не входят, они не учитываются)	4	4	6
Описание количества структур (сущностей) в словаре данных (связующие таблицы описывать не нужно)	2	4	6

#### 3.3 Порядок выполнения работы

1. Изучить теоретический материал, изложенный в подразделе 3.4;
2. На основе анализа предметной области системы, выделенных классов пользователей и функциональных требований из ЛР 2, определить хранение какой информации будет осуществлено в базе данных системы;
3. Разработать нормализованную диаграмму «сущность-связь» и выполнить реструктуризацию отношений «многие-ко-многим»;
4. Составить словарь данных или его фрагмент;
5. Ознакомиться с требованиями по содержанию отчета в подразделе 2.5;
6. Написать отчет о работе.

При разработке диаграммы «сущность-связь» и составлении словаря данных необходимо опираться на русские наименования сущностей, атрибутов и элементов данных, так как на текущем этапе осуществляется определение и анализ требований к данным. Разрешается дублирование наименований на английские эквиваленты, однако выбор наименований переменных системы не является задачей данной работы.

Для выполнения лабораторных работы можно воспользоваться любой средой моделирования или CASE-средством, которое поддерживает нотацию Баркера для диаграммы «сущность-связь» (ERD). Рекомендуется использовать бесплатное для некоммерческого использования CASE-средство [Software Ideas Modeler](#)

### 3.4 Теоретический материал

Наиболее известным представителем класса семантических (логических) моделей является модель «сущность-связь» или ER-модель (Entity-Relationship Model, ERM). ER-модель используется при концептуальном проектировании базы данных. Во время проектирования базы данных происходит преобразование ER-модели в конкретную схему базы данных на основе выбранной модели (реляционной, сетевой etc.). Стандартной графической нотацией, с помощью которой можно визуализировать ER-модель, является диаграмма сущность-связь или ER-диаграмма (Entity-Relationship Diagram, ERD).

При выполнении лабораторной работы и построении диаграммы сущность-связь необходимо использовать нотацию Баркера, как самую распространенную. Элементы и примеры данной нотации будут описаны ниже.

Базовыми понятиями ER-модели являются: сущность, атрибут и связь (отношение).

*Сущность* — реальный или абстрактный объект предметной области, имеющий атрибуты. Каждая сущность должна:

- иметь уникальное имя;
- обладать одним или несколькими атрибутами, которые либо принадлежат сущности, либо наследуются через связь;
- обладать одним или несколькими атрибутами, которые однозначно идентифицируют каждый экземпляр сущности.

Сущность представляет собой множество экземпляров реальных или абстрактных объектов (людей, событий, состояний, предметов etc.). Имя сущности должно отражать тип или класс объекта, а не его конкретный экземпляр. Например, «Аудитория» будет сущностью, в то время как «23-10» экземпляром данной сущности.

На ER-диаграмме в нотации Баркера сущность изображается прямоугольником, иногда с закругленными углами (рис. 3.1, а).

*Атрибут* — любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности (рис. 3.1, б). Каждая сущность обладает одним или несколькими атрибутами. Атрибуты делятся на ключевые, являющиеся частью уникального идентификатора, который называется первичным ключом, и описательные — прочие атрибуты.



Рисунок 3.1 — Обозначение сущности в нотации Баркера

- а) — без атрибутов; б) — с указанием атрибутов;  
 с) — с уточнением атрибутов и их типов;

В ER-модели атрибуты ассоциируются с конкретными сущностями и, соответственно, экземпляр сущности должен обладать единственным определенным значением для ассоциированного атрибута. Данное значение будет экземпляром атрибута (определенной характеристикой конкретного экземпляра сущности). Например, если сущность представлена человеком, то одним из атрибутов данной сущности будет его имя, а конкретный экземпляр атрибута, например, имя Евгений соответственно будет характеристикой сущности.

*Первичный ключ* — это атрибут или совокупность атрибутов и/или связей, предназначенная для уникальной идентификации каждого экземпляра сущности. В случае совокупности атрибутов говорят о составном первичном ключе, примером такого ключа может служить серия и номер паспорта. Номер паспорта, как и серия могут повторяться, но вместе они образуют уникальный идентификатор, который однозначно идентифицирует экземпляр сущности, в качестве которой в данном случае выступает человек. Ключевые атрибуты помещают в начало списка и помечают символом «#» (рис. 3.1, с). В Software Ideas Modeler для ключевых атрибутов зарезервирован символ «+», а «#» — для внешних ключей. В настоящих методических указаниях для первичного ключа используется «#». Понятие внешнего ключа выходит за рамки данной ЛР, поэтому здесь оно затрагиваться не будет.

В свою очередь описательные атрибуты бывают обязательными или необязательными. Обязательные атрибуты для каждой сущности всегда имеют конкретное значение, необязательные — могут быть не определены. Обязательные описательные атрибуты помечают символом «\*». В Software Ideas Modeler используется ключевое слово «NOT NULL» для обязательных атрибутов и соответственно «NULL» для необязательных, значения которых могут быть не заданы (например, человек может не указать свой пол).

*Связь* — поименованная ассоциация между двумя или более сущностями, значимая для рассматриваемой предметной области. Связь, таким образом, означает, что каждый экземпляр одной сущности ассоциирован с произвольным (в том числе и нулевым) количеством экземпляров второй сущности и наоборот. Если любой экземпляр одной сущности связан хотя бы с одним экземпляром другой сущности, то связь является обязательной (рис. 3.2, а). Необязательная связь представляет собой условное отношение между сущностями (рис. 3.2, б).

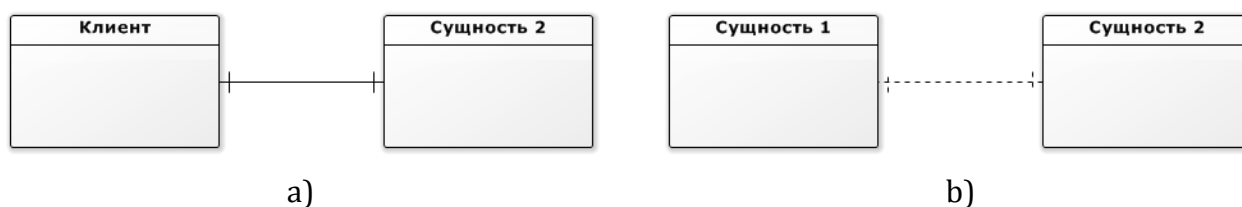


Рисунок 3.2— Обозначение связи в нотации Баркера  
а) — обязательная связь б) — необязательная

Связь предполагает некоторое отношение сущностей, которое характеризуется количеством экземпляров сущности, участвующих в связи с каждой стороны.

Различают три типа отношений (рис. 3.3):

1:1 — «один к одному» — одному экземпляру первой сущности соответствует один экземпляр второй;

1:M — «один ко многим» — одному экземпляру первой сущности соответствуют несколько экземпляров второй;

M:M — «многие ко многим» — каждому экземпляру первой сущности может соответствовать несколько экземпляров второй и, наоборот, каждому экземпляру второй сущности может соответствовать несколько экземпляров первой.

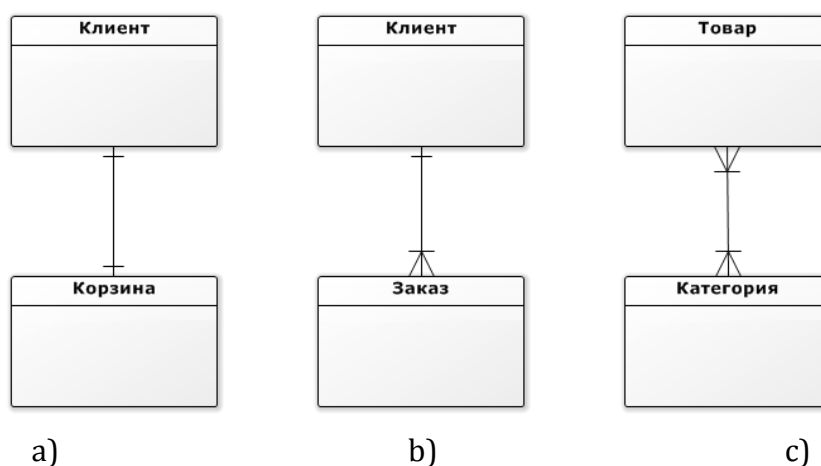


Рисунок 3.3 — Обозначение отношений в нотации Баркера  
а) — «один к одному» б) — «один ко многим» в) — «многие ко многим»

Отношение «многие ко многим» в базе данных реализуется с помощью трех таблиц. Две таблицы – исходных сущностей (рис. 3.4) и одна связующая таблица. Данное представление необходимо в связи с тем, что при отношении «многие ко многим» между двумя сущностями возникает неоднозначность, какие экземпляры одной сущности относятся к экземплярам другой (например, какой товар был произведен каким изготовителем, и наоборот).



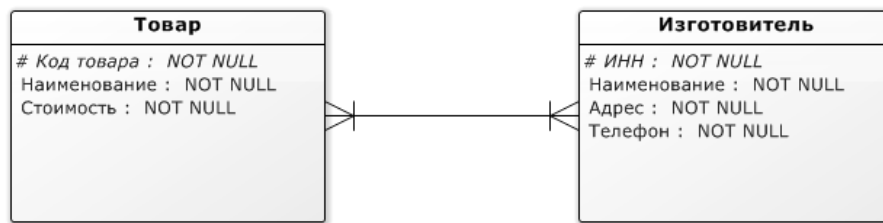


Рисунок 3.4 — Отношение «многие ко многим»

Первичный ключ связующей таблицы всегда будет составным, то есть состоять из двух внешних ключей, которые ссылаются на первичные ключи исходных таблиц (рис. 3.5).

Таким образом, при добавлении связующей таблицы, отношение «многие ко многим» будет состоять из двух отношений «один ко многим». Связующая таблица должна находиться на стороне «многих» обоих отношений. Имя для связующей таблицы, как правило, состоит из имен исходных таблиц, в данном случае ТоварИзготовитель (ItemMaker).

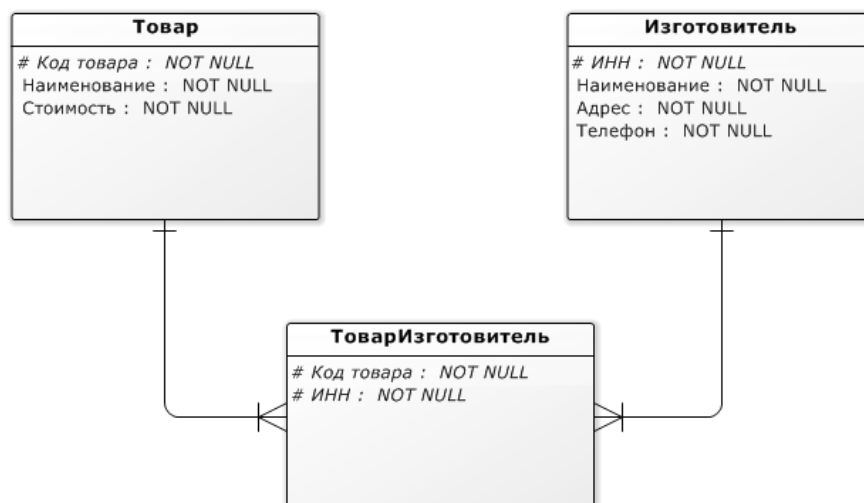


Рисунок 3.5 — Представление отношения «многие ко многим» в базе данных

*Нормальная форма* — свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение.

Процесс преобразования отношений базы данных к виду, отвечающему нормальным формам, называется *нормализацией*. Нормализация предназначена для приведения структуры БД к виду, обеспечивающему минимальную логическую избыточность, и не имеет целью уменьшение или увеличение производительности работы или же уменьшение или увеличение физического объёма базы данных. Конечной целью нормализации является уменьшение потенциальной противоречивости хранимой в базе данных информации.

Общее назначение процесса нормализации заключается в исключении некоторых типов избыточности, устранении определенных аномалий обновления и разработке проекта базы данных, который является достаточно «качественным» представлением реального мира, интуитивно понятен и может служить хорошей основой для последующего расширения;

Устранение избыточности производится, как правило, за счёт декомпозиции отношений таким образом, чтобы в каждом отношении хранились только первичные факты (то есть факты, не выводимые из других хранимых фактов).

#### *Первая нормальная форма (1NF).*

Переменная отношения находится в первой нормальной форме тогда и только тогда, когда в любом допустимом значении отношения каждая строка содержит только одно значение для каждого из атрибутов.

В реляционной модели отношение всегда находится в первой нормальной форме по определению понятия отношение.

Однако сами таблицы могут не быть правильными представлениями отношений и, соответственно, могут не находиться в 1NF.

Например, исходная ненормализованная таблица в БД:

<u>id</u>	customer	email
1	Архангельская Е.А.	archangel95@mail.net
2	Павлов Е.В.	evpavlov@mail.net forspamfromguap@mail.net

Таблица в БД, приведённая к 1NF, будет выглядеть следующим образом:

<u>id</u>	customer	email
1	Архангельская Е.А.	archangel95@mail.net
2	Павлов Е.В.	evpavlov@mail.net
3	Павлов Е.В.	forspamfromguap@mail.net

При реализации БД стараются избегать составных и многозначных атрибутов, чтобы не усложнять написание кода, не перегружать его структуру и не запутывать пользователей. Из этих соображений логически и вытекает определение первой нормальной формы.

Если у нас есть повторяющиеся атрибуты (рис. 3.6), такая сущность также не будет отвечать требованиям 1NF.

Товар
# Код товара : NOT NULL
Наименование : NOT NULL
Стоимость : NOT NULL
Категория1 : NOT NULL
Категория2 : NULL
Категория3 : NULL

Рисунок 3.6 — Сущность с повторяющимися атрибутами

В данном случае повторяющиеся атрибуты выделяют в отдельную сущность:

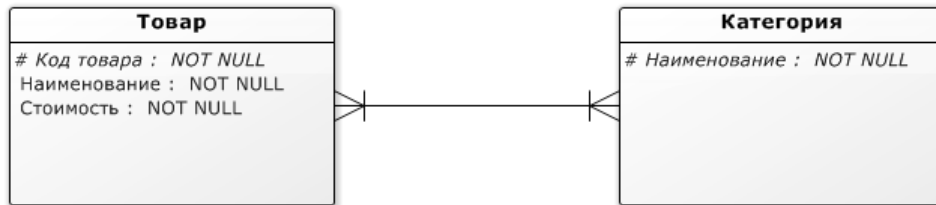


Рисунок 3.7 — Приведение отношения к 1NF

### Вторая нормальная форма (2NF).

Переменная отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме и каждый неключевой атрибут неприводимо зависит от её потенциального ключа.

Неприводимость означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, от которого можно также вывести данную функциональную зависимость. Для неприводимой функциональной зависимости часто используется эквивалентное понятие «полная функциональная зависимость».

Если потенциальный ключ является простым, то есть состоит из единственного атрибута, то любая функциональная зависимость от него является неприводимой (полной). Если потенциальный ключ является *составным*, то согласно определению второй нормальной формы в отношении не должно быть неключевых атрибутов, зависящих от части составного потенциального ключа.

На рис. 3.8 представлен пример неполной функциональной зависимости неключевых атрибутов от составного ключа. Наименование и стоимость зависят от кода товара, но адрес изготовителя и телефон зависят только от изготовителя, что очевидно нарушает 2NF.



Рисунок 3.8 — Пример отношения не отвечающего 2NF

Для приведения ко 2NF необходимо осуществить декомпозицию исходного отношения на два:

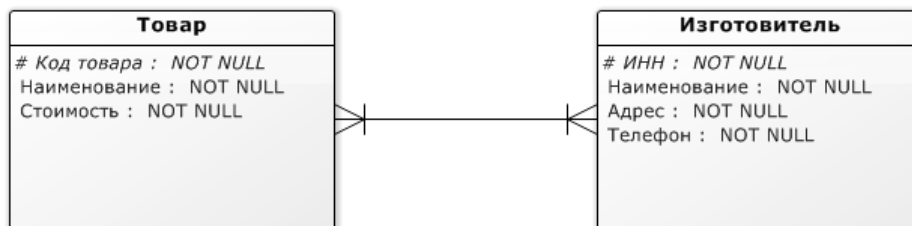


Рисунок 3.9 — Приведение отношения ко 2NF

### Третья нормальная форма (3NF).

Отношение находится в 3NF тогда и только тогда, когда выполняются следующие условия:

- отношение находится во второй нормальной форме.
- ни один неключевой атрибут отношения не находится в транзитивной функциональной зависимости от потенциального ключа отношения.

Под транзитивной зависимостью понимают отношение типа:  $A \rightarrow B$  ( $A$  определяет  $B$ ),  $B \rightarrow C$ , следовательно  $A \rightarrow C$  (зависимость  $C$  от  $A$  является транзитивной).

Иными словами, 3NF можно сформулировать так: каждый атрибут должен предоставлять информацию о ключе, полном ключе и ни о чём, кроме ключа. Данная формулировка проще и достаточно точно отражает условия 3NF.

Например, у нас есть отношение, представленное на рис. 3.10. Данное отношение находится во 2NF так как оно имеет простой первичный ключ (состоящий из одного атрибута). Время, статус и оператор в данном случае определяются кодом заказа, в то время, как телефон оператора полностью зависит от самого оператора. Таким образом, код заказа  $\rightarrow$  оператор, оператор  $\rightarrow$  телефон оператора, соответственно зависимость код заказа  $\rightarrow$  телефон оператора является транзитивной, следовательно, отношение не находится в третьей нормальной форме.

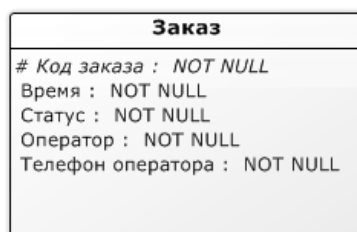


Рисунок 3.10 — Пример транзитивной зависимости атрибутов

Для приведения к 3NF необходимо осуществить декомпозицию исходного отношения на два:

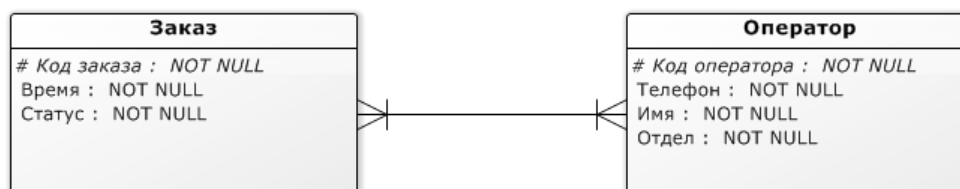


Рисунок 3.11 — Приведение отношения к 3NF

Иногда правила нормализации специально нарушают (в этом случае говорят о денормализации), это связано с двумя основными причинами: удобство и быстроедействие. Меньшим число таблиц проще управлять, чем большим. Кроме того, из-за более сложного характера, нормализованные таблицы более медленные для обновления, изменения и выборки данных.

В тоже время необходимо учитывать, что существует достаточно способов для улучшения производительность базы данных, но не так много способов, чтобы исправить повреждённые данные, возникшие из-за некорректно спроектированной структуры БД, именно поэтому нормализация важна с точки зрения надежности и стабильности работы БД.

Зачастую при проектировании БД вместо естественных первичных ключей (например, код заказа или серия и номер паспорта) используется суррогатный ключ — это дополнительное служебное поле, добавленное к уже имеющимся информационным полям таблицы, единственное предназначение которого — служить первичным ключом. Значение суррогатного ключа не образуется на основе каких-либо других данных из БД, а генерируется искусственно и представляет собой числовое поле, в которое заносятся значения из возрастающей числовой последовательности. Типичным примером суррогатного ключа является автоинкрементный ключ (ID).

Суррогатный ключ используют по следующим причинам:

- **Неизменность.** Главное достоинство суррогатного ключа состоит в том, что он практически никогда не меняется, поскольку не несёт никакой информации из предметной области и, следовательно, в минимальной степени зависит от изменений, происходящих в ней. Атрибуты естественного ключа время от времени могут меняться — например, человек может изменить имя или фамилию, получить новый паспорт взамен потерянного;
- **Гарантированная уникальность.** Далеко не всегда можно гарантировать то, что уникальность естественного ключа не будет скомпрометирована с течением времени. Различные внешние факторы могут приводить к тому, что естественный ключ, ранее уникальный, в новых обстоятельствах может утратить уникальность. Суррогатный ключ свободен от этого недостатка;
- **Гибкость.** Поскольку суррогатный ключ неинформативен, его можно свободно заменять;
- **Эффективность.** Ссылки удобнее хранить в виде целых чисел, чем в виде громоздких естественных ключей. К тому же многие запросы к БД будут компактнее и быстрее, чем при использовании естественных ключей.

Однако у суррогатного ключа есть и свои недостатки:

- **Уязвимости генераторов ключей.** Например, по номерам ключей можно узнать, сколько записей появилось в БД за некоторый период;
- **Неинформативность.** Усложняется ручная проверка БД, в запросах появляются INNER JOIN там, где без них можно обойтись. По этой причине в полях перечисляемого типа часто используют ключи в виде коротких строк.
- **Склоняет администратора пропустить нормализацию.** Добавить суррогатные ключи проще, чем правильно, с учётом дублирования и соотношений «один ко многим» разбить БД на таблицы и проставить уникальные индексы;
- **Вопросы оптимизации.** СУБД приходится поддерживать два индекса, суррогатный и естественный;
- **Невольная привязка разработчика к поведению генератора ключей в конкретной СУБД.** Например, разработчик может предполагать, что сообщение с меньшим ключом появилось раньше, что в некоторых обстоятельствах может быть не так.

*Словарь данных* (data dictionary) представляет набор подробной информации об используемых в приложении сущностях данных.

Сбор информации о составе, типах данных, разрешенных значениях, etc. в виде единого ресурса, служащего для определения критериев проверки данных, помогает разработчикам правильно писать программы и избавляет от проблем с интеграцией.

Во время анализа требований информация словаря данных представляет элементы и структуры данных предметной области. Эта информация попадает в дизайн в форме схем баз данных, таблиц и атрибутов, что в конечном итоге определяет имена переменных в программах. Словарь данных позволяет избежать ошибок, связанных с тем, что участники проекта по-разному понимают ключевые данные. Регулярное обновление словаря данных позволяет сделать его ценным средством как при обслуживании системы, так и при разработке схожих систем. В противном случае он может вводить в заблуждение, предоставляя устаревшую информацию, и члены команды перестают доверять ему.

Словарь данных по сути является одним из атрибутом повышения качества разработки. Определения данных часто повторно используются в других приложениях, особенно в одном семействе продуктов. Использование единообразных определений данных в компании снижает вероятность возникновения ошибок интеграции и интерфейса.

Записи словаря данных могут представлять следующие типы элементов данных:

- Простейшие элементы данных;
- Структуры.

Простейшим элементом данных называется тот, дальнейшая декомпозиция или упрощение которого невозможно или не нужно (идентификатор пользователя, возраст, полное имя, количество опубликованных статей, etc.).

В описании простейших элементов данных указывают тип, дину, диапазон числовых значений, список разрешенных значений и другие уместные атрибуты.

Структура данных (или запись) содержит несколько элементов данных (пользователь, конференции, участник конференции, etc.).

В столбце «Структура или тип данных» в словаре данных перечисляются элементы, из которых состоит структура, а элементы отделяются знаком «плюс» (+). Каждый элемент данных в структуре должен быть определен в словаре данных.

Если элемент в структуре данных необязателен (значение, которое не должно предоставляться пользователем или системой), его заключают в скобки. В примере ниже таким элементов является «категория», данный элемент не является обязательным для заполнения.

Структуры могут содержать другие структуры: структура «Научная статья» включает структуру «Выходные данные», которая в свою очередь может содержать: фамилию и инициалы автора (авторов), полное название статьи, журнал (сборник), в котором опубликована, место выпуска, год, месяц и номер выпуска, страницы, на которых размещена статья.

Ниже представлен фрагмент примера словаря данных, при выполнении работы, поле «Описание» можно опустить.

Таблица 3.4 — Фрагмент словаря данных

Элемент данных	Описание	Структура или тип данных	Длина	Значение
Научная статья	Информация о научной статье	Идентификатор статьи		
		+ Автор		
		+ Выходные данные		
		+ Заголовок		
		+ Аннотация		
		+ Ключевые слова		
		+ Текст статьи		
		+ (Категория)		
Идентификатор статьи	Уникальный идентификатор статьи	Целое число	8	Генерируемый системой порядковый номер, начиная с 1
Автор	Полное имя автора или авторов статьи	Буквенные символы	300	Может содержать пробелы, дефисы, точки и апострофы; для разделения авторов используется запятая; формат: фамилия имя отчество
Выходные данные		Буквенные символы переменной длины		
Заголовок	Наименование статьи	Буквенные символы	300	
Аннотация	Характеристика статьи	Буквенные символы переменной длины		
Ключевые слова	Выражают основное смысловое содержание статьи	Буквенные символы	120	Может содержать от 4 до 10 слов
Текст статьи	Полный текст статьи	Буквенные символы переменной длины		
(Категория)		Перечисляемый тип данных		<ul style="list-style-type: none"> <li>• Международная система цитирования</li> <li>• Список журналов ВАК</li> <li>• Система цитирования РИНЦ</li> <li>• Другое</li> </ul>

### 3.5 Содержание отчета

Отчет о работе должен содержать:

1. Титульный лист
2. Цель работы
3. Задание на лабораторную работу и вариант
4. Требования к данным системы
  - 4.1 Моделирование отношений к данным (диаграмма «сущность-связь»)
  - 4.2 Словарь данных
5. Выводы по работе
6. Используемые источники (2-4 источника)

### 3.6 Пример выполнения работы

#### 3.6.1 Моделирование отношений данных

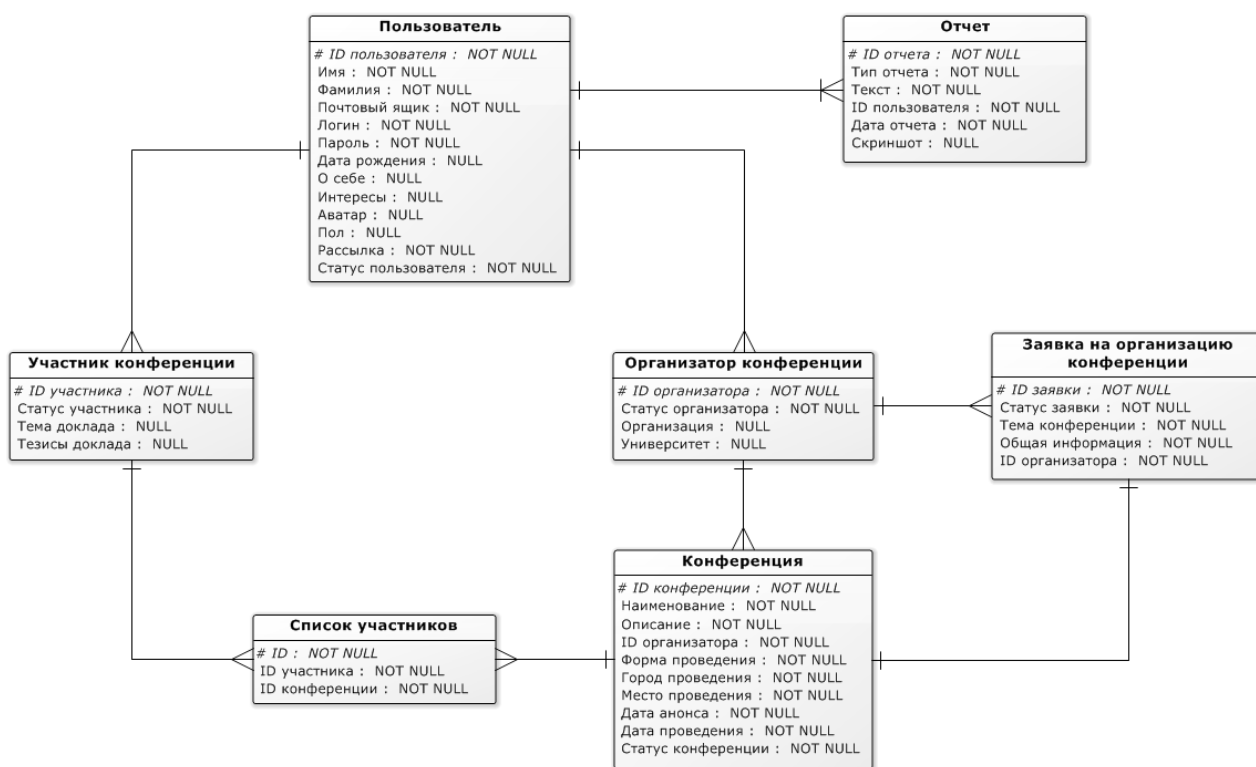


Рисунок 3.12 – Диаграмма «сущность-связь» для системы информационной поддержки научных конференций

Сущность «Список участников» является связующей таблицей в БД между сущностями «Участник конференции» и «Конференция» так как в данном случае имеет место отношение «многие-ко-многим».

Перед тем как будет создана запись в БД о конференции, пользователь создает заявку и если заявка не будет одобрена, то соответствующая запись в таблице «Конференция» не появится. Однако информация о заявке будет сохранена в любом случае.



### 3.6.2 Словарь данных

Таблица 1 – Принятые сокращения и типы данных

1	DATE	Дата в формате ГГГГ-ММ-ДД
2	DATETIME	Дата и время в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС
3	ENUM	Перечисляемый тип данных, позволяет вводить список допустимых значений
4	INT	Сокращение от INTEGER
5	INTEGER	Целочисленные значения в диапазоне от $-2^{32}$ до $2^{32}$
6	TEXT	Данные переменной длины от 1 до 65535 байт
7	VARCHAR	Строковые данные переменной длины от 1 до 8000 байт

Таблица 2 – Фрагмент словаря данных для проектируемой системы

Элемент данных	Структура или тип данных	Длина	Значение
Пользователь	ID пользователя		
	+ Имя		
	+ Фамилия		
	+ Почтовый ящик		
	+ Логин		
	+ Пароль		
	+ Дата рождения		
	+ О себе		
	+ Интересы		
	+ Аватар		
	+ Пол		
	+ Рассылка		
	+ Статус пользователя		
ID пользователя	INT		Генерируемый системой порядковый номер, начиная с 1
Имя	VARCHAR	40	Может содержать пробелы, дефисы, точки и апострофы
Фамилия	VARCHAR	40	Может содержать пробелы, дефисы, точки и апострофы
Почтовый ящик	VARCHAR	40	
Логин	VARCHAR	25	Может содержать символы латинского алфавита, подчеркивание и цифры
Пароль	VARCHAR	25	Может содержать символы латинского алфавита, числа и символы из следующего после двоеточия списка: ! @ # \$ % ^ & ? * _ Остальные символы, включая пробел, запрещены
(Дата рождения)	DATE		
(О себе)	VARCHAR		
(Интересы)	VARCHAR		
(Аватар)	VARCHAR		Может содержать символы, доступные в имени файла
(Пол)	ENUM		Мужской Женский Неопределённый (мы же политкорректные) Не указан (по умолчанию)
Рассылка	ENUM		Включена Выключена (по умолчанию)

Статус пользователя	ENUM		5 – Пользователь-гость (если пользователь не подтвердил регистрацию) 4 - Обычный пользователь 3 - Участник конференции 2 - Организатор конференции (если пользователь одновременно и участник, и организатор, то ему присваивается статус организатора как более высокий) 1 - Администратор
Отчет	ID отчета		
	+ Тип отчета		
	+ Текст		
	+ ID пользователя		
	+ Дата отчета		
	+ Скриншот		
ID отчета	INT		Содержит начальную букву типа отчета и генерируемый системой порядковый номер для данного типа, начиная с 1
Тип отчета	ENUM		R - Отчет об ошибке D - Пожелание Q - Вопрос
Текст отчета	TEXT		
Дата отчета	DATETIME		
(Скриншот)	VARCHAR		Может содержать символы, доступные в имени файла
Участник конференции	ID участника		
	+ Статус участника		
	+ Тема доклада		
	+ Тезисы доклада		
ID участника	INT		Генерируемый системой порядковый номер, начиная с 1
Статус участника	ENUM		Слушатель Докладчик
(Тема доклада)	VARCHAR	300	
(Тезисы доклада)	VARCHAR		
Организатор конференции	ID организатора		
	+ Статус организатора		
	+ Организация		
	+ Университет		
ID организатора	INT		Генерируемый системой порядковый номер, начиная с 1
Статус организатора	ENUM		Частное лицо Представитель коммерческой организации Представитель общественной организации Представитель университета Представитель исследовательского института
(Организация)	VARCHAR	300	
(Университет)	VARCHAR	300	
Конференция	ID конференции		
	+ Наименование		
	+ Описание		
	+ ID организатора		
	+ Форма проведения		
	+ Город проведения		
	+ Место проведения		

	+ Дата анонса		
	+ Дата проведения		
	+ Статус		
ID конференции	INT		Генерируемый системой порядковый номер, начиная с 1
Наименование	VARCHAR	300	
Описание	VARCHAR		
Форма проведения	ENUM		Очная Заочная (дистанционная) Очно-заочная
Город проведения	VARCHAR	50	
Место проведения	VARCHAR	140	
Дата анонса	DATE		
Дата проведения	DATE		
Статус	ENUM		Активна Закрыта Удалена Отложена
Заявка на организацию конференции	ID заявки		
	+ Статус заявки		
	+ Тема конференции		
	+ Общая информация		
	+ ID организатора		
ID заявки	INT		Генерируемый системой порядковый номер, начиная с 1
Статус заявки	ENUM		Ожидает Принята Отклонена Отложена
Тема конференции	VARCHAR	300	
Общая информация	VARCHAR		

## Выводы по работе

В результате выполнения данной лабораторной работы изучены способы определения требований к данным и представления логической модели данных.

Разработана ER-диаграмма и составлен фрагмент словаря данных для системы информационной поддержки научных конференций. В разработанной модели рассмотрен не исчерпывающий список элементов данных проектируемой системы, именно поэтому речь идёт о фрагменте словаря данных. Указанное замечание не является недостатком работы, так как разработанная модель отвечает функциональным требованиям, представленным в спецификации требований к ПО. Однако по ходу выполнения работы стало ясно, что есть некоторые рассогласования в требованиях к системе, в частности:

- В разработанной модели данных, одной из структурных единиц является «Конференция», однако в спецификации (SRS) речь шла о мероприятиях, который могут включать в себя не только конференции, а семинары, симпозиумы и другие события научной направленности;

- Перед тем как в базе данных системы будет создана запись о конференции, пользователю необходимо направить соответствующую заявку, которая требует одобрения администратора системы. До тех пор, пока заявка не будет одобрена, пользователь не может заполнить форму создания конференции, так как у него нет прав на осуществления данного действия. В спецификации нет информации в отношении этого требования.

### 3.7 Контрольные вопросы

- 1) Что представляет собой диаграмма сущность-связь?
- 2) В чем отличие диаграммы сущность-связь от диаграммы потоков данных?
- 3) Что представляет собой сущность?
- 4) В чем отличие атрибута от сущности?
- 5) В каком случае атрибут можно представить как отдельную сущность?
- 6) Что такое экземпляр сущности?
- 7) Чем отличаются ключевой атрибут от обязательного?
- 8) Может ли быть ключевой атрибут необязательным?
- 9) Что такое составной ключ? Приведите пример.
- 10) Что показывает отношение (связь) на диаграмме сущность-связь?
- 11) Что такое кратность связи и какие виды кратности различают?
- 12) Что представляют собой обязательные и необязательные связи?
- 13) Что такое нормальная форма?
- 14) Что такое нормализация и для чего применяется?
- 15) Как нормализация связана с увеличением производительности БД?
- 16) Позволяет ли нормализация уменьшить физический объем БД?
- 17) Опишите первую нормальную форму. Приведите пример.
- 18) Опишите вторую нормальную форму. Приведите пример.
- 19) Опишите третью нормальную форму. Приведите пример.
- 20) Каким образом представляются в БД отношения «многие ко многим»?
- 21) Почему отношение «многие ко многим» нуждается в реструктурировании?
- 22) Что такое денормализация?
- 23) Каково назначение денормализации?
- 24) Что такое суррогатный ключ и чем он отличается от естественного?
- 25) Каковы преимущества и недостатки суррогатного ключа?
- 26) Чем опасно игнорирование нормализации БД?
- 27) Что представляет собой словарь данных и каково его назначение?
- 28) Какую информацию включают в словарь данных?

## **4. ЛАБОРАТОРНАЯ РАБОТА**

### **«ТЕСТИРОВАНИЕ ТРЕБОВАНИЙ.**

### **СОСТАВЛЕНИЕ СЦЕНАРИЕВ ТЕСТИРОВАНИЯ»**

#### **4.1 Цель работы**

Целью данной работы является изучение способов уточнения и тестирования требований к проектируемой программной системе.

#### **4.2 Задание на лабораторную работу**

На основе функциональных требований и описания вариантов использования из ЛР 2 составить сценарии тестирования (test case) и выполнить анализ покрытия функциональных требований с использованием матрицы соответствия.

Тест-кейсы могут описывать как основной и альтернативный потоки, так и исключения. К одному требованию разрешается не более трёх тест-кейсов.

Лабораторная работа будет оцениваться в соответствии со следующими критериями:

	Удовл.	Хорошо	Отлично
Минимальное количество тест-кейсов	8	12	20
Минимальное количество требований, которые должны быть покрыты тест-кейсами	4	8	12

#### **4.3 Порядок выполнения работы**

1. Изучить теоретический материал, изложенный в подразделе 4.4;
2. Проанализировать спецификацию SRS (ЛР 2) и выделить критерии оценки правильной реализации требований;
3. Составить тест-кейсы и выполнить анализ покрытия требований при помощи матрицы соответствия;
4. Ознакомиться с требованиями по содержанию отчета в подразделе 4.5;
5. Написать отчет о работе.

Для построения матрицы соответствия требований рекомендуется использовать любое приложение для работы с электронными таблицами, например, MS Excel или LibreOffice Calc.

#### 4.4 Теоретический материал

Тесты составляют альтернативное представление требований к системе и создаются на основе пользовательских требований для документирования ожидаемого поведения системы в определенных условиях.

Если варианты использования написаны полно, аккуратно и ясно, то процесс составления тестов становится достаточно простым, в противном случае может сложиться ситуация, когда разработчик и специалист по тестированию по-разному интерпретируют правильную реализацию функциональности системы (типичная ситуация: это не баг, это фича). Поэтому при составлении тестов необходимо отладить разработанные варианты использования, если они описаны недостаточно полно, чтобы лучше понять требования к системе.

Тесты должны охватывать нормальные и альтернативные направления всех вариантов использования, а также исключения, определенные в ходе выявления требований и анализа.

*Тест-кейс (test case) или сценарий тестирования* – это тестовый артефакт, суть которого заключается в выполнении некоторого количества действий и/или условий, необходимых для проверки определенной функциональности разрабатываемой программной системы.

Тест-кейс (сценарий тестирования) имеет следующую структуру:

- Выполняемое действие (action);
- Ожидаемый результат (expected result);
- Фактический результат (test result).

При составлении тестов на этапе проектирования ограничиваются только выполняемыми действиями и ожидаемым результатом.

Непосредственно сам сценарий тестирования состоит из 3 частей:

- PreConditions (предусловия) – это может быть как список шагов, которые приводят проверяемую систему в состояние, пригодное для тестирования, так и список проверок условий того, что система уже находится в необходимом состоянии;
- Test Case Description (описание тестового случая) – список действий, с помощью которых осуществляется основная проверка функционала (после которой и сверяется фактический результат с ожидаемым);
- PostConditions (постусловия) – список действий, которые возвращают систему в исходное состояние.

В настоящей работе основная задача заключается в анализе и тестировании требований, поэтому постусловия тест-кейса не рассматриваются, если на этот счет нет особых указаний в спецификации SRS.

Ниже представлен шаблон тест-кейса, который необходимо взять за основу при выполнении работы. Данной шаблон отличается от типового шаблона, так как из него исключены поля, которые не существенны в рамках данной работы.

Шаблон тест-кейса:

Уникальный номер (ID) текст-кейса / Краткое описание тест-кейса (одна или несколько фраз, из которых понятно, что проверяется данным сценарием)	
Ссылка на требование (в данной работе необходимо указать идентификатор требования)	
Дата создания тест-кейса и его автор	
Предварительная настройка окружения / системы (подготовка необходимой аппаратной части и/или выполнение программных настроек)	
Шаги теста (краткое и четкое описание элементарного действия, необходимого для проверки) 1. ... 2. ...	Ожидаемый результат (описание того, что ожидается после выполнения данного действия) 1. ... 2. ...

Тест-кейсы необходимо связывать с функциональными требованиями, чтобы убедиться, что ни одно требование не было упущено и что каждому соответствует определенный тест. Одним из артефактов проверки покрытия требований тестами является матрица соответствия требований (requirements traceability matrix).

*Матрица соответствия требований* представляет собой двумерную таблицу, которая содержит функциональные требования системы и соответствующие им подготовленные тест-кейсы.

В заголовках колонок таблицы расположены требования, а в заголовках строк — тестовые сценарии. На пересечении — отметка, означающая, что требование текущей колонки покрыто тестовым сценарием текущей строки.

Во второй строке указывается количество составленных тест-кейсов для данного требования.

Матрица соответствия требований имеет следующий вид:

Требование	FR-01	FR-02	FR-03	FR-04	FR-05	FR-06	FR-07	FR-08	FR-09
Тест-кейсы	3	1	0	1	1	1	0	2	1
ТС-1.1	×								
ТС-1.2	×								
ТС-1.3	×								
ТС-2		×							
ТС-4				×					
ТС-5					×				
ТС-6						×			
ТС-8.1								×	
ТС-8.2								×	
ТС-9									×

Таким образом, из данной матрицы следует, что для требования FR-01 составлено 3 тест-кейса, в то время как для требований FR-03 и FR-07 ни одного.



## 4.5 Содержание отчета

Отчет о работе должен содержать:

1. Титульный лист
2. Цель работы
3. Задание на лабораторную работу и вариант
4. Тест-кейсы для функциональных требований системы
5. Анализ покрытия требований тестами
6. Выводы по работе
7. Используемые источники (2-4 источника)

## 4.6 Пример выполнения работы

ID и наименование: TC-1.1 Проверка регистрации пользователя	
Ссылка на требование: FUN-01	
Дата создания / Автор: 08.12.2018 / Евгений	
Предусловия: предварительная настройка среды или системы не требуется	
Шаги теста:	Ожидаемый результат:
1. Нажимаем кнопку «Регистрация»	1. Система выводит на экран форму регистрации
2. Вводим корректные пользовательские данные: - логин (символы латинского алфавита, подчеркивание и цифры, от 4 до 25 символов); - пароль (символы латинского алфавита, числа и некоторые специальные символы, от 8 до 25 символов); - почтовый адрес (до 40 символов).	2.1 При вводе система оповещает, соответствуют ли введенные данные требованиям для регистрации 2.2 Если включен CAPS LOCK или русская раскладка клавиатуры при вводе и подтверждении пароля, система предупреждает об этом 2.3 Если пользовательские данные корректны, кнопка «Зарегистрироваться» становится активна
3. Нажимаем «Зарегистрироваться»	3.1 Система закрывает форму регистрации и выводит сообщение о необходимости подтвердить регистрацию через ссылку в письме 3.2 Система отправляет письмо с ссылкой на почтовый ящик, который указал пользователь
4. Переходим по ссылке в письме	4. Система выводит сообщение об успешной регистрации и предлагает авторизоваться
5. Нажимаем «Вход в систему»	5. Система выводит на экран форму авторизации
6. Вводим данные для авторизации	6. Если включен CAPS LOCK или русская раскладка клавиатуры при вводе пароля, система предупреждает об этом
7. Нажимаем «Войти»	7. Вместо функциональных кнопок «Вход» и «Регистрация» отображается панель с логином и иконкой пользователя

ID и наименование:	ТС-1.2 Проверка авторизации, если пользователь не подтвердил регистрацию	
Ссылка на требование:	FUN-01	
Дата создания / Автор:	08.12.2018 / Евгений	
Предусловия:	выполняем шаги 1-3 теста ТС-1.1 и пропускаем шаг 4	
Шаги теста:	Ожидаемый результат:	
1. Нажимаем «Вход в систему»	1. Система выводит на экран форму авторизации	
2. Вводим данные для авторизации	2. Если включен CAPS LOCK или русская раскладка клавиатуры при вводе пароля, система предупреждает об этом	
3. Нажимаем «Войти»	3. Система запрещает вход и предупреждает о том, что необходимо подтвердить регистрацию, перейдя по ссылке в письме	

ID и наименование:	ТС-1.3 Проверка регистрации при некорректных пользовательских данных	
Ссылка на требование:	FUN-01	
Дата создания / Автор:	08.12.2018 / Евгений	
Предусловия:	предварительная настройка среды или системы не требуется	
Шаги теста:	Ожидаемый результат:	
1. Нажимаем кнопку «Регистрация»	1. Система выводит на экран форму регистрации	
2. Вводим некорректные пользовательские данные: - логин (до 3 включительно или более 25 символов); - пароль (до 7 включительно или более 25 символов); - почтовый адрес (более 40 символов).	2.1 Система предупреждает о некорректных данных при регистрации 2.2 Отображается сообщение о допустимой длине данных регистрации 2.3 Кнопка «Зарегистрироваться» не активна	
3. Вводим некорректные пользовательские данные: - логин (символы русского алфавита, специальные символы и пробел, от 4 до 25 символов); - пароль (символы русского алфавита, пробел и специальные символы из набора: / \   ~ ` " { } [ ] ( ) , - + = ; : < > от 8 до 25 символов); - почтовый адрес (неправильный формат, например, email@com, emailgmail.com)	3.1 Система предупреждает о некорректных данных при регистрации 3.2 Отображается сообщение о допустимом формате данных и разрешенных символах 3.3 Кнопка «Зарегистрироваться» не активна	

ID и наименование:	ТС-2 Проверка редактирования личных данных пользователя
Ссылка на требование:	FUN-02
Дата создания / Автор:	08.12.2018 / Евгений
Предусловия: пользователь авторизован	
Шаги теста:	Ожидаемый результат:
1. Нажимаем на иконку пользователя в правом верхнем углу	1. Система выводит всплывающий список
2. Выбираем «Профиль»	2. Система перенаправляет на страницу профиля пользователя
3. Нажимаем «Редактировать профиль»	3.1 Система выводит поля с личной информацией пользователя с возможностью редактирования 3.2 Появляется кнопка «Отмена» для отмены всех изменений
4. Вносим изменения в поля: - Имя - Фамилия - Дата рождения - О себе - Интересы - Пол	4.1 При заполнении поля «О себе» система выдает шаблон того, что можно указать в данном поле 4.2 При заполнении поля «Интересы» система подсказывает возможные варианты 4.3 Пол можно выбрать из всплывающего списка, содержащего четыре позиции (по умолчанию: «Не указан»)
5. Нажимаем «Сохранить изменения»	5. Система запрашивает ввод пароля для подтверждения действия
6. Вводим пароль	6. Если включен CAPS LOCK или русская раскладка клавиатуры при вводе пароля, система предупреждает об этом
7. Зажимаем ввод или «Принять»	7. Отображается страница профиля пользователя с измененными личными данными

ID и наименование: TC-6 Проверка выхода из участия в конференции	
Ссылка на требование: FUN-06	
Дата создания / Автор: 08.12.2018 / Евгений	
Предусловия: пользователь авторизован и состоит в списке участников конференции	
Шаги теста:	Ожидаемый результат:
1. Выбираем вкладку «Мои конференции»	1. Система перенаправляет на страницу конференций пользователя
2. Нажимаем на значок «Покинуть конференцию» напротив конференции, из которой выходим	2. Система выводит сообщение с просьбой подтвердить действие или отменить
3. Нажимаем «Выйти»	3.1 Система обновляет список конференций, в которых участвует пользователь и выводит в нижнем правом углу сообщение, что пользователь вышел из конференции; в сообщении есть кнопка «Отмена», которая отменяет выход 3.2 Пользователь может закрыть сообщение, либо система закроет его через восемь секунд

ID и наименование: TC-7.1 Проверка удаления участника конференции	
Ссылка на требование: FUN-07	
Дата создания / Автор: 08.12.2018 / Евгений	
Предусловия: пользователь авторизован и является организатором конференции	
Шаги теста:	Ожидаемый результат:
1. Выбираем вкладку «Мои конференции»	1.1 Система перенаправляет на страницу конференций пользователя 1.2 Сверху списка идут конференции, которые создал пользователь
2. Нажимаем на конференцию, либо на значок «Подробнее»	2. Открывается информация о конференции
3. Нажимаем на «Список участников»	3. Открывается окно списка участников
4. Отмечаем любого участника	4. Напротив выбранной фамилии ставится галочка
5. Нажимаем «Исключить»	5. Система выводит сообщение с просьбой подтвердить действие или отменить
6. Нажимаем «Подтвердить»	6. Система обновляет список участников конференции и выводит в нижнем правом углу сообщение, что участник «имя» был исключен из списка участников; в сообщении есть кнопка «Отмена», которая отменяет исключение 6.1 Пользователь может закрыть сообщение, либо система закроет его через восемь секунд

ID и наименование:	ТС-7.2 Проверка удаления всех участников конференции через очистку списка участников	
Ссылка на требование:	FUN-07	
Дата создания / Автор:	08.12.2018 / Евгений	
Предусловия:	пользователь авторизован и является организатором конференции	
Шаги теста:	Ожидаемый результат:	
1. Выбираем вкладку «Мои конференции»	1.1 Система перенаправляет на страницу конференций пользователя 1.2 Сверху списка идут конференции, которые создал пользователь	
2. Нажимаем на конференцию, либо на значок «Подробнее»	2. Открывается информация о конференции	
3. Нажимаем на «Список участников»	3. Открывается окно списка участников	
4. Нажимаем «Очистить список участников»	4. Система выводит сообщение с просьбой подтвердить действие или отменить	
5. Нажимаем «Подтвердить»	5.1 Система обновляет список участников конференции и выводит в нижнем правом углу сообщение, что все участники конференции были исключены; в сообщении есть кнопка «Отмена», которая отменяет действие 5.1 Пользователь может закрыть сообщение, либо система закроет его через восемь секунд	

ID и наименование: TC-8 Проверка удаления конференции	
Ссылка на требование: FUN-07	
Дата создания / Автор: 08.12.2018 / Евгений	
Предусловия: пользователь авторизован и является организатором конференции	
Шаги теста:	Ожидаемый результат:
1. Выбираем вкладку «Мои конференции»	1.1 Система перенаправляет на страницу конференций пользователя 1.2 Сверху списка идут конференции, которые создал пользователь
2. Нажимаем на конференцию, либо на значок «Подробнее»	2. Открывается информация о конференции
3. Нажимаем на «Удаление конференции»	3. Система выводит сообщение с просьбой подтвердить действие или отменить
4. Нажимаем «Подтвердить»	4. Система выводит форму для заполнения и просит указать причину удаления конференции; в форме отображены две кнопки «Отправить запрос на удаление» и «Отменить» удаление конференции
5. Нажимаем «Отправить запрос на удаление»	5.1 Система перенаправляет на вкладку «Мои конференции», удаленная конференция помечается серым цветом 5.2 Любые действия с удаленной конференцией, кроме просмотра информации, запрещены для пользователя 5.3 Все участники конференции исключены из списка участников 5.4 После рассмотрения заявки администратором, конференция будет удалена из списка конференций пользователя

## Анализ покрытия требований тестами

Для анализа покрытия построим матрицу соответствия требований:

Requirement ID	FUN-01	FUN-02	FUN-03	FUN-04	FUN-05	FUN-06	FUN-07	FUN-08	FUN-09
Test Cases	3	1	0	0	0	1	2	1	0
TC-1.1									
TC-1.2									
TC-1.3									
TC-2									
TC-6									
TC-7.1									
TC-7.2									
TC-8									

FUN-10	FUN-11	FUN-12	FUN-13	FUN-14	FUN-15	FUN-16	FUN-17	FUN-18
0	0	0	0	0	0	0	0	0

Рисунок 4.1 – Матрица соответствия требований

Как видно из данной матрицы, покрытие требований тестами составляет 27% (5 из 18). При этом на требования FUN-01 и FUN-07 приходится по 3 и 2 тест-кейса соответственно. Большая часть требований осталась не покрыта тестами (13 из 18).

### Выводы по работе

В результате выполнения данной лабораторной работы изучены способы уточнения и тестирования требований к проектируемой системе, а именно анализ функциональных требований с последующим их представлением в виде сценариев тестирования.

Составлены 8 тест-кейсов и построена матрица соответствия требований.

Анализ покрытия показывает, что составленные тесты покрывают лишь 27% функциональных требований, что в конечном итоге уменьшает надежность системы и увеличивает риски недостатков требований, включая ситуации некорректных или пропущенных требований.

Если сопоставить описания вариантов использования из ЛР 2 с тест-кейсами, то видно, что некоторые требования нуждаются в уточнении, в частности FUN-08: постусловия варианта использования UC-03 предполагают, что удаленная конференция сохраняется в архив, однако архив предназначен только для завершенных мероприятий, поэтому информация об удаленных конференциях не должна сохраняться где-либо, что и отражено в тест-кейсе ТС-8.

#### **4.7 Контрольные вопросы**

- 1) Для чего тестируют требования?
- 2) Каким образом можно протестировать требования?
- 3) Что представляет собой тест-кейс?
- 4) Как связаны вариант использования и тест-кейс?
- 5) Какова структура тест-кейса?
- 6) Приведите шаблон описания тест-кейса.
- 7) Что представляет собой матрица соответствия требований?
- 8) Что представляет собой покрытие требований тестами?



## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Виггерс, Карл. Разработка требований к программному обеспечению = Software Requirements: пер. с англ.; 3-е изд., дополненное / Карл Виггерс, Джой Битти. - М.: Издательство «Русская редакция», 2014. - 736 с.: ил.
2. Data Flow Diagram [Электронный ресурс]. - SmartDraw, 1994-2018. - URL: <https://www.smartdraw.com/data-flow-diagram/> (дата обращения: 10.11.2018)
3. Эванс, Эрик. Предметно-ориентированное проектирование: структуризация сложных программных систем = Domain-Driven Design: tackling complexity the heart of software: пер. с англ. / Эрик Эванс. - М.: ИД Вильямс, 2011. - 448 с.: ил.
4. Дейт, К. Дж. Введение в системы баз данных = An Introduction to Database System: пер. с англ.; 8-е изд. / Дж. К. Дейт. - М.: ИД «Вильямс», 2018. - 1328 с.: ил.
5. Куликов, С. С. Тестирование программного обеспечения. Базовый курс / С. С. Куликов. - Минск: Четыре четверти, 2017 - 312 с.
6. Traceability matrix [Электронный ресурс]: From Wikipedia, the free encyclopedia / Wikipedia. 2018. URL: [https://en.wikipedia.org/wiki/Traceability\\_matrix](https://en.wikipedia.org/wiki/Traceability_matrix) (дата обращения: 12.11.2018)
7. 830-1998 - IEEE Recommended Practice for Software Requirements Specifications. - NY: The Institute of Electrical and Electronics Engineers, 2009. - 31 с.
8. Software Ideas Modeler [Электронный ресурс]: CASE tool for software design & analysis. - Dušan Rodina, 2009-2018. - URL: <https://www.softwareideas.net/> (дата обращения: 10.11.2018)

## ПРИЛОЖЕНИЕ

Перечень вариантов (программные системы) для выполнения лабораторных работ и курсового проектирования

	Наименование варианта
1	Онлайн-бронирование билетов (железнодорожных, авиа, автобусных etc.)
2	Онлайн-бронирование номеров в гостиницах
3	Онлайн-бронирование театральных билетов (театральная касса)
4	Онлайн-бронирование столов в ресторанах и ночных клубах
5	Онлайн-бронирование туров
6	Онлайн аренда автомобилей
7	Онлайн-путеводитель по городам и странам
8	Планировщик маршрутов общественного транспорта
9	Управление аэропортом
10	Управление железнодорожным вокзалом
11	Управление грузовыми перевозками
12	Географическая информационная система
13	Управление больницей
14	Электронная регистратура
15	Медицинский информационно-аналитический портал
16	Сервис для изучения иностранных языков
17	Биржа труда (Интернет-рекрутмент)
18	Система федеральной налоговой службы
19	Система миграционного учета (включая сервисы по вопросам миграции)
20	Система дактилоскопического учёта и хранения криминальных досье
21	Система поиска пропавших без вести людей
22	Мониторинг и учет преступлений в сфере информационных технологий

23	Центральная избирательная комиссия
24	Оплата государственных услуг
25	Веб-сервис мониторинга информации о дорожных пробках
26	Умное здание (Smart Building)
27	Умный дом (Smart Home)
28	Автоматизация процессов на предприятиях общественного питания
29	Система контроля кассовых операций для магазина
30	Управление образовательным процессом в школе
31	Управление образовательным процессом в высшем учебном заведении
32	Дистанционное обучение (виртуальная школа)
33	Обучающий портал (курсы и вебинары)
34	Онлайн курсы подготовки к ЕГЭ и ОГЭ
35	Музыкальная школа онлайн
36	Управление студенческим общежитием
37	Управление музеем
38	Виртуальные экскурсии по музеям
39	Художественная онлайн-галерея
40	Система контроля и управления доступом (СКУД)
41	Управление процессами охранного предприятия
42	Веб-портал для отдела кадров
43	Онлайн-страхование (прямое страхование)
44	Юридические консультации онлайн
45	Платформа для организации работы компании (виртуальный офис)
46	Управление взаимоотношениями с клиентами (клиентская база и call-центр)
47	Сервис для сбора клиентских отзывов
48	Родительский контроль
49	Агентство недвижимости

50	Благотворительный фонд
51	Краудфандинговая платформа
52	Букмекерская контора
53	Торговля акциями на бирже
54	Сервис для проведения Интернет-аукционов
55	Сервис для проведения конкурса красоты
56	Виртуальная служба знакомств
57	Интернет-ателье
58	Научно-популярный портал
59	Проведение научных конференций
60	Астрономическая Интернет-обсерватория
61	Астрофизическая интерактивная база данных
62	Видеостриминговый сервис
63	Киберспортивный портал
64	Игровой портал
65	Клиент-серверное приложение для MMORPG
66	База знаний для MMORPG
67	Сервис покупки и продажи игровой валюты (биржа игровых ценностей)
68	Клиент-серверное приложение для фарма игровой валюты (MMORPG)
69	Клиент-серверное приложение для MOBA
70	Проведение чемпионата по MOBA
71	Клиент-серверное приложение для MMO-Action
72	Игровой сервис для совместной игры через Интернет
73	Игровая соцсеть с обеспечением многопользовательской игры и услуг связи
74	Онлайн-кинотеатр
75	База данных о фильмах
76	Корпоративная социальная сеть

77	Веб-сервис социальных закладок
78	Сервис вопросов и ответов с элементами социальной сети
79	Сервис для обмена фотографиями и видеозаписями с элементами соцсети
80	Сервис для создания онлайн дневников (блогов)
81	Сервис для мгновенного обмена сообщениями
82	Веб-сервис для тайм-менеджмента
83	Сервис поиска и прослушивания музыки онлайн (служба потокового аудио)
84	Сервис мониторинга музыкальных новинок
85	Онлайн платформа для продвижения музыкантов и исполнителей
86	Платформа для онлайн-трансляции концертов
87	Творческий портал
88	Социальная сеть для дизайнеров
89	Торговая площадка для покупки и продажи авторских хендмейд работ
90	Аниме портал
91	Система управления массированным ответным ядерным ударом (Dead Hand)
92	Система контроля и управления атомной станцией
93	Интернет-казино
94	Видеохостинг
95	Облачное хранилище данных
96	Веб-сервис для хостинга IT-проектов и их совместной разработки
97	Управление марсоходом (планетоходом)
98	Фильтрация содержимого Интернета
99	Углубленное наблюдение за Интернет-трафиком
100	Веб-сервис по организации платного троллинга
101	Сервис для владельцев домашних питомцев
102	Сервис для продвижения в социальных сетях (SMM)
103	Площадка для обеспечения прямых поставок (Drop Shipping System)