Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
Высшего образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»
(ВлГУ)

Кафедра информационных систем и программной инженерии

Методические указания к курсовому проектированию

по дисциплине:
«Технологии программирования»

Составители:
Вершинин В.В.
Макаров С.А.

Владимир 2016

Оглавление

1. Общие положения

Общая тематика курсового проектирования по дисциплине «Технологии программирования» связана с рассмотрением и изучением вопросов создания, проектирования и разработки программ, имеющих признаки сложных систем и являющихся средством поддержки и автоматизации функций информационных систем (ИС), частью которых они и являются. Процесс выполнения курсового проекта представляется как взаимосвязь последовательных этапов, связанных с реализацией программной системы, программного комплекса или его части в соответствии с общепринятыми стандартами в области разработки software (например, RUP).

Информационная система (ИС) — система обработки информации и соответствующие организационные ресурсы (человеческие, технические, финансовые и т. д.), которые обеспечивают и распространяют информацию.

Информационная система предназначена для своевременного обеспечения надлежащих людей надлежащей информацией, то есть для удовлетворения конкретных информационных потребностей в рамках определенной предметной области, при этом результатом функционирования информационных систем является информационная продукция — документы, информационные массивы, базы данных и информационные услуги.

Достаточно широкое определение понятия «информационная система» подразумевает, что неотъемлемыми компонентами ИС являются данные, техническое и программное обеспечение, а также персонал и организационные мероприятия. Широко трактует понятие «информационной системы» федеральный закон Российской Федерации «Об информации, информационных технологиях и о защите информации», подразумевая под информационной системой совокупность содержащейся в базах данных информации и обеспечивающих её обработку информационных технологий и технических средств. Среди российских ученых в области информатики, наиболее широкое определение ИС дает М. Р. Когаловский, по мнению которого в понятие информационной системы помимо данных, программ, аппаратного обеспечения и людских ресурсов следует также включать коммуникационное оборудование, лингвистические средства и информационные ресурсы, которые в совокупности образуют систему, обеспечивающую «поддержку динамической информационной модели некоторой части реального мира для удовлетворения информационных потребностей пользователей».

2. Цель и задачи курсового проектирования

Целью курсового проектирования является изучение процесса разработки сложных систем на примере конкретной программной системы, определяемой тематикой курсового проекта. Для достижения поставленной цели необходимо решить следующие задачи:

- изучение особенностей конкретной предметной области, относящихся к теме курсового проекта;
- анализ возможных подходов и методов решения с обоснованием выбранного подхода;
- разработка моделей (математической, структурной, информационной, данных) программной системы;
- выбор алгоритмов с учетом их точности и спецификой предметной области;
- разработка программного обеспечения;
- анализ полученных результатов работы ПО.

3. Этапы выполнения проекта и их содержание

3.1 Требования к порядку выполнения курсового проекта

Порядок выполнения курсового проекта определяется общими требованиями к курсовым проектам, принятым в ВлГУ. Конечным результатом курсового проектирования является программная система, выполненная в соответствии требованиями и задачами проектной работы, изложенными в разделе «цель и задачи курсового проектирования». При этом студент обязан в течении семестра докладывать о состоянии дел и об уровне завершенности курсового проекта. Все вопросы решаются в рабочем порядке, на занятиях по курсовому проектированию в соответствии с расписанием занятий. В случае возникновения задержек с выполнением курсового проектирования об этом предупреждается заранее или переносится на более поздние сроки в индивидуальном порядке. В течении семестра предусмотрено 4 рубежных контрольных мероприятия с представлением промежуточных материалов:

1. постановка задачи. Анализ задания. Анализ предметной области. Результаты анализа задания и предметной области (техническое здание на разработку ПС, словарь предметной области с перечислением сущностей (и их атрибутов) и характеристикой связей между ними;
2. участники системы и варианты ее использования (фактически это: диаграммы прецедентов взаимодействия с системой и диаграммы последовательностей, диаграмма классов *концептуального* уровня). Анализ прецедентов и формальные функциональные требования к системе;
3. архитектура разрабатываемой ПС (фактически это диаграмма классов, приближенная к реализации, диаграммы состояний, видов деятельности);

4. реализация программной системы (диаграмма модулей, диаграмма развертывания).

3.2 Календарный план выполнения курсового проекта

Курсовой проект выполняется студентами в самостоятельной форме в течении всего семестра. В рамках дисциплины предусмотрены практические занятия, на которых рассматриваются вопросы выполнения каждого этапа курсового проекта. В процессе работы над курсовым проектом студент самостоятельно изучает предметную область выбранной тематики и необходимые для этого литературные (представлены в разделе 6) источники. Конечным результатом выполнения проекта является отчет (пояснительная записка) по выполненной работе. Примерное распределение видов деятельности по неделям учебного семестра представлено в таблице.

Таблица 1. Примерный календарный план выполнения курсового проектирования

№пп.	**Наименование выполняемой работы**	**Неделя**
1.	Согласование тем курсовых проектов и выдача, и оформление заданий	1 – 2 недели
2.	Изучение предметной области, изучение структуры и функций информационной системы. Предъявление требований к программной системе и выделение функций программной системы и вариантов ее использования. Составление словаря предметной области и синтез диаграммы прецедентов	3 – 5 недели
3.	Моделирование структуры объектов предметной области и их взаимодействия на концептуальном уровне (уровне предметной области). Распределение функций системы по ее структурным элементам. Составление спецификаций прецедентов системы. Синтез диаграмм последовательностей и диаграмм прецедентов	6 – 7 недели
4.	Проектирование структуры программной системы. Формирование диаграммы классов и диаграммы объектов программной системы. Синтез и построение диаграмм состояний и последовательностей	8 – 9 недели
5.	Реализация программной системы, составление тестов. Сохранение результатов работы в репозиторий проекта	10 – 15 недели
6.	Составление пояснительной записки и графического материала	16 неделя
7.	Сдача курсовых работ и их проверка. Защита	16 – 18 неделя

Допускается незначительное отклонение хода выполнения работ от календарного плана.

3.3 Требования оформлению курсового проекта

Общие требования к оформлению пояснительной записки курсового проекта определяются требованиями к порядку выполнения курсового проекта. Пояснительная записка оформляется в соответствии с требованиями (ГОСТ, ЕСКД, ЕСПД) и должна удовлетворять требованиям, изложенным в [8]. Общий объем пояснительной записки курсового проекта составляет 30-40 страниц машинописного текста, включая рисунки, таблицы и содержит обязательные части:

1. титульный лист;
2. лист задания;
3. аннотация (на русском и английском языках);
4. лист содержания (нумеруется как первый лист в составе пояснительной записки);
5. основные разделы записки (введение, анализ, проектирование, разработка, реализация, тестирование; эксплуатационная документация, заключение, список использованных источников);
6. приложения.

В отчете по курсовому проектированию должно быть, как минимум одно приложение, содержащее исходный код разработанной программной системы. Графический материал вычерчивается на чертежах формата А1. Минимальное количество два листа. Структура и содержание графического материала определяется в листе задания на проект и согласовывается с преподавателем.

Примечание: допускается предоставление уменьшенных копий (до формата А4) чертежей, прикладываемых отдельными листами к пояснительной записке, при условии, что защита будет происходить с использованием электронных средств (презентация).

3.4 Лист задания

Для контроля выполнения курсового проекта используется лист задания. Каждый студент должен иметь распечатанный и заполненный экземпляр. При сдаче каждого этапа преподаватель делает соответствующею пометку. После выполнения курсового проекта лист задания включается в состав пояснительной записки.

Шаблон листа задания должен поставляться вместе с данными методическими указаниями.

4. Сроки и порядок защиты курсового проекта

Срок сдачи курсового проекта происходит на последних неделях учебного семестра. После проверки назначается дата защиты проекта. Защита происходит в составе комиссии, состоящей из трех преподавателей. Общая продолжительность 5 минут. Результаты выполнения курсового проекта и его защита оценивается по пятибалльной системе. Оценка проставляется в ведомость и зачетную книжку студента.

5. Перечень тем курсового проектирования

1. Программная система учета знаний учащихся.

2. Программная система автоматизации и учета подвижного состава автобусного предприятия.

3. *Название*: Программная система учета пациентов в информационной системе «Поликлиника» (2)

Краткая характеристика системы: учет и хранение обращений пациентов в поликлинику.

Минимальный набор функций:

– поддержка (авторизация) различных типов пользователей (администратор, врач, пациент);
– поддержка справочников диагнозов (адм.);
– поддержка справочника врачей (адм.);
– выписка рецепта (врач);
– подача заявок и просмотр их состояния (пациент);
– назначение пациенту лечащего врача (система).

4. *Название*: Программная система движения материалов на оптовой базе

Краткая характеристика системы: учет и хранение данных об отпущенных материалах

Минимальный набор функций:

– поддержка (авторизация) различных типов пользователей (администратор, оптовик);
– поддержка справочников материалов и оптовиков (адм.);
– отгрузка/принятие материала (оптовик/система);
– просмотр остатков материалов (оптовик);
– просмотр сведений об операциях отгрузки и принятия товара за выбранный период (оптовик/система).

5. *Название*: Программная система автоматизации учета торговой деятельности компьютерного магазина

Краткая характеристика системы: автоматизация работы компьютерного магазина

Минимальный набор функций:

– поддержка (авторизация) различных типов пользователей (менеджер, продавец);
– поддержка справочника категорий товара (продавец);
– поддержка справочника поставщиков (продавец);
– списание остатков товара (менеджер/система);
– продажа товара продавцом (пациент);
– просмотр оборота продаж продавцом за выбранный период (менеджер).

6. *Название*: Программная система автоматизации документооборота малого предприятия

Краткая характеристика системы: учет поступления документов и их исполнение

Минимальный набор функций:

– поддержка (авторизация) различных типов пользователей (администратор, директор, начальник отдела);
– поддержка справочника отделов, пользователей (адм.);
– поддержка справочника типов документа
– добавление нового документа и его учет по исполнителям (директор/начальник/система);
– отчет по исполненным/неисполненным документам (директор/система);

7. *Название*: Программная система "Туристическое агентство"

Краткая характеристика системы: подбор туров для отдыхающих

Минимальный набор функций:

– поддержка (авторизация) различных типов пользователей (турменеджер, посетитель);
– поддержка справочника туров (турменеджер);
– план отдыха посетителя (аналог корзины);
– вывод статистики по странам отдыха (турменеджер/система);
– автоматическая скидка на «горящие туры» (система/турменеджер);

8. *Название*: Информационно-программная система «Рекламное агентство»

Краткая характеристика системы: автоматизация работы рекламного агентства

Минимальный набор функций:

– поддержка (авторизация) различных типов пользователей (администратор, менеджер, клиент);
– поддержка справочника менеджеров (администратор);
– поддержка справочника категорий рекламы (менеджер);
– поддержка справочника клиентов (менеджер);
– оформление заказа на размещение рекламы (менеджер/система);
– снятие рекламы с оборота по прошествии акции (система/менеджер).

9. *Название*: Программная система автоматизации учета расходных материалов на промышленном предприятии

Краткая характеристика системы: автоматизация учета расходных материалов по различным категориям и по различным структурным единицам

Минимальный набор функций:

– поддержка (авторизация) различных типов пользователей (администратор, начальник подразделения, учетчик);
– поддержка справочника пользователей, (администратор);
– поддержка справочника техники, на которую расходуется материал (начальник подразделения);
– поддержка справочника расходных материалов (учетчик);
– ввод новых расходных материалов (учетчик);
– оценка результата прохождения тестов (система);
– вывод сведений по израсходованным и оставшимся материалам (система/начальник подразделения/учетчик);
– вывод сведений по среднему расходу материала за выбранный период (система/учетчик).

10. *Название*: Программная система «Страховое агентство»

Краткая характеристика системы: организация учета страховой деятельности организации.

Минимальный набор функций:

– поддержка (авторизация) различных типов пользователей (администратор, агент, пользователь);
– поддержка справочника услуг страхования (администратор);
– регистрация договора (агент);
– поддержка справочника юр/физ лиц (агент);
– вывод сведений по страховым договорам за выбранный период (система/агент);
– поиск договоров по критериям (система/агент);
– автоматический вывод сведений по «проблемным» договорам (система).

11. *Название*: Программная система «Деканат университета»

Краткая характеристика системы и минимальный набор функций: обговариваются индивидуально

12. *Название*: Программная система «Дистанционное обучение студента» (тема допускает два исполнителя)

Краткая характеристика системы: организация дистанционного прохождения пользователем системы тестов

Минимальный набор функций:

– поддержка (авторизация) различных типов пользователей (администратор, лектор, студент);
– поддержка справочника лекторов (администратор);
– поддержка справочника предметов (лектор);
– поддержка справочника студентов (администратор);
– прохождение справочника тестов (лектор);
– включение студента в дистанционное прохождение тестов (лектор);
– прохождение тестов (система/студент);

– прохождение тестов (система/студент);
– оценка результата прохождения тестов (система);
– вывод сведений по итогам прохождения теста (система/лектор).

13. *Название*: Программная система «Кадровое агентство»

Краткая характеристика системы: организация работы кадрового агентства;

Минимальный набор функций:

– поддержка (авторизация) различных типов пользователей (администратор, менеджер, пользователь);
– поддержка справочника пользователей (администратор);
– поддержка справочника вакансий (менеджер);
– заполнение анкет от соискателей (менеджер);
– допуск пользователя к просмотру анкет соискателей (менеджер);
– подбор персонала для пользователя (менеджер);
– просмотр базы данных (анкет) соискателей: разовый допуск и допуск на N дней (менеджер/пользователь);

14. *Название*: Программная система «Кадастровое управление»

Краткая характеристика системы: организация работы кадастрового управления;

Минимальный набор функций:

– ввод данных в систему о кадастровых объектах (земельный участок, объект недвижимости и т.д.)
– работа со справочником типов объектов учета в Кадастровом управлении;
– самостоятельная регистрация заявителя в системе;
– подача заявки на регистрацию кадастрового объекта;
– рассмотрение заявок и регистрация (или отказ).
– просмотр сведений о кадастровых объектах и изменение сведений о них

15. *Название*: Регистрационная система «Паспортный стол»

Краткая характеристика системы: организация работы паспортно-визовой службы;

Минимальный набор функций:

– работа со справочником типов документов, регистрируемых в службе(паспорт, регистрация по месту и т.д.);
– регистрация заявителя в системе;
– подача заявки на выдачу/оформление документа;
– рассмотрение заявок на оформление документа (или отказ) и последующая выдача.
– поиск и просмотр сведений о заявках и изменение сведений о них.

16. *Название*: Программная система «База знаний отрасли»

Краткая характеристика системы: организация и поддержка работы системы хранения знания по отраслям;

Минимальный набор функций:

– работа следующих категорий пользователей в системе: администратор, менеджер, пользователи (читатели);

– работа со справочниками отраслей, разделов знаний в отрасли (например: тяжелая промышленность/горное дело/чёрная металлургия и т.п.)

– добавление статей в базу знаний;

– комментирование статей и их модерация (добавление к статьи и отклонение).

17. *Название*: Подсистема учета расходных материалов промышленного предприятия

Краткая характеристика системы: организация учета расходных материалов в процессе работы промышленного предприятия;

Минимальный набор функций:

– работа следующих категорий пользователей в системе: администратор, заведующий склада, учетчики;

– работа со справочниками категорий материалов (например: инструмент, доска, бумага для принтера и т.п.);

– ввод поступлений материалов;

– создание, открытие и закрытие табеля оперативного расхода материала;

– вывод сведений об остатке материала на основании данных по оперативному расходу в виде табели рас.

18. *Название*: Программная система учета пассажирских перевозок информационной системы «Автотранспортное предприятие»

Краткая характеристика системы: поддержка работы автотранспортного предприятия и автоматизация деятельности;

Минимальный набор функций:

– работа следующих категорий пользователей в системе: администратор, диспетчер, водитель, пользователь (покупатель);

– работа со справочниками маршрутов, автотранспортных единиц, персонала;

– ввод маршрутов и распределение персонала (водитель + кондуктор) по ним;

– вход (+саморегистрация) покупателей и бронирование/покупка билета на рейс;

– подтверждение брони/покупки диспетчером

– просмотр актуальных сведений о маршрутах.

19. *Название*: Программная система муниципальной образовательной информационной сети

Краткая характеристика системы: организация работы кадрового агентства;

Минимальный набор функций:

– работа следующих категорий пользователей в системе: администратор, представитель информационной сети, сотрудник образовательного учреждения;

– работа со справочниками образовательных учреждений;

– добавление годовых отчетов от муниципального образовательного учреждения;

– утверждение отчета представителем сети (или отправка на доработку);

– фильтрация и вывод сведений по образовательным учреждениям.

6. Список литературы

1. Липаев В.В. Проектирование программных систем. М.: Высш. шк, 1990.

2. Буч Г. Объектно-ориентированное проектирование / Пер. с анг. Конкорд, 1996.

3. Майерс Г. Надежность программного обеспечения. М.: Мир, 1980

4. Калянов Г.Н. CASE структурный системный анализ (автоматизация и применение). – М.: «ЛОРИ», 1996.

5. Шмуллер Дж. Освой самостоятельно UML. 2-е изд. М.: изд. дом «Вильямс», 2002

6. Калберстон Р., Браун К., Кобб Г. Быстрое тестирование М.: изд. дом «Вильямс», 2002

7. Кармайкл Э., Хейвуд Д. Быстрая и качественная разработка программного обеспечения. М.: изд. дом «Вильямс», 2003

8. Буланкин В.Б. «Стандарт предприятия» 2004 год

7. Порядок проведения курсового проектирования

1. Постановка задачи и анализ предметной области

Цель: провести анализ предметной области для проектирования архитектуры приложения и перехода к следующим этапам разработки системы.

Задачи: исследовать предметную область, выделить основные сущности и их атрибуты; определить условия функционирования разрабатываемой системы и ее основные функции.

Результат: техническое задание на разработку системы; словарь предметной области; перечень требований к разрабатываемой программной системе и варианты ее использования (оформленные в виде диаграммы UseCase), концептуальная модель предметной области.

2. Участники системы и варианты ее использования (фактически это: диаграммы прецедентов взаимодействия с системой и диаграммы последовательностей, диаграмма классов *концептуального* уровня). Анализ прецедентов и формальные функциональные требования к системе.

При построении диаграмм вариантов использования нужно придерживаться следующих принципов (правил):

- не моделируйте связи между действующими лицами. По определению действующие лица находятся вне сферы действия системы. Это означает, что связи между ними также не относятся к ее компетенции.
- не соединяйте стрелкой два варианта использования непосредственно. Диаграммы данного типа описывают только сами варианты использования, а не порядок их выполнения. Для отображения порядка выполнения вариантов использования применяют диаграммы деятельности.

- каждый вариант использования должен быть инициирован действующим лицом. Это означает, что всегда должна быть стрелка, начинающая на действующем лице и заканчивающаяся на варианте использования.

3. Архитектура разрабатываемой ПС (фактически это диаграмма классов, приближенная к реализации, диаграммы состояний, видов деятельности).

4. Реализация программной системы (диаграмма модулей, диаграмма развертывания).

8. Состав и описание основных этапов разработки программной системы

8.1. Анализ

Каждая предметная область оперирует со своими понятиями и терминами, а также объектами и связями между ними. Для того, чтобы разрабатываемая система была адекватна предметной области, была понятна тем, кто с ней будет работать необходим качественный и всесторонний анализ предметной области.

В промышленной разработке ПС разработчику всегда приходится тесно общаться с заказчиком, для наиболее полного выявления фактов предметной области, выделения объектов системы и установления отношений между ними. Кроме этого на данном этапе нужно определить границы системы и выделить основные функции, которые необходимо будет реализовывать. Все эти моменты определяются результатами:

- обследования объекта заказчика, для которого и разрабатывается ПС;
- переговоров заказчика и исполнителя по конкретным вопросам;
- конкретных требований, предъявляемых заказчиком к системе.

Итогом анализа является техническое задание на разрабатываемую систему. Формально этот документ описывает общее видение системы с описанием требований к ней и основных функций системы. Дополнительно там же оговариваются вопросы коммерческой составляющей для разработки системы, основные этапы и календарные сроки. В некоторых случаях по общей договоренности сторон в задание вносятся дополнительные пункты.

При выполнении анализа происходит формирование перечня понятий предметной области, основных сущностей и функций будущей системы.

В курсовой работе в качестве заказчика выступает преподаватель. Задача студента на данном этапе провести анализ предметной области, основываясь на своих собственных знаниях, литературных источников и в ходе общения с преподавателем.

Результат должен быть оформлен в виде краткой характеристики предметной области (которое является частью введения пояснительной записки). За описанием должен следовать словарь предметной области, состоящий из списка объектов и их характеристик (как совокупности свойства/атрибутов и операций над ним/поведения). Объект и его свойства определяются как существительные, а его поведения как глагольные формы русского языка. Словарь логически связан с описанием предметной области. Всякие противоречия должны быть исключены Словарь является отправным моментом для последующих этапов проектирования и разработки системы.

8.2 Проектирование структуры системы

Все задачи, решаемые на данном этапе соотносятся с проработкой и синтезом архитектурных решений и решений, которые в дальнейшем будут использоваться при реализации системы. Из предыдущего этапа извлекается перечень понятий и перекладывается в систему классов и связей между ними. Как правило, существительные из словаря переходят в категорию классов, глагольные формы в операции классов (над классами). Все упоминания одного класса в контексте другого класса выражаются в виде связей между классами типа ассоциация.

8.2.1 Проектирование диаграммы прецедентов

На данном этапе происходит локализаций основных функций системы. Результат оформляется в виде диаграммы прецедентов и спецификаций на каждый прецедент. Согласно методологии UML, каждый прецедент будет тождественен бизнес-функции системы, выявленной на этапе анализа или же она описана в техническом задании. Диаграмма прецедентов дополняется спецификацией прецедентов и таблицей распределения функций системы по прецедентам, из которой четко и ясно следует, что все функции будут реализованы и учтены. Шаблон спецификации прецедента представлен в таблице 2.

Таблица 2. Шаблон спецификации прецедента программной системы

Название раздела спецификации прецедента	Назначение
Название	Название прецедента, которое совпадает с именованием прецедента из диаграммы
Актер	Актер, запускающий прецедент или получающий результаты выполнения прецедента
Предусловия	Условия, которые должны быть выполнены, чтобы прецедент мог выполняться
Постусловия	Особые условия, которые должны быть удовлетворены

	после выполнения основного потока событий
Основной поток событий	Описание набора шагов, выполнение которых полностью реализует функцию (прецедент)
Альтернативные потоки событий	Набор альтернативных потоков событий, в которых происходит обработка особых ситуаций, не предусмотренных основным потоком событий

Формат описания разделов «Основной поток событий» и «Альтернативные потоки событий» представляется в виде последовательности пар: «действие пользователя – реакция системы» или «действие системы – реакция пользователя», в зависимости от того, кто инициирует взаимодействие в рамках описываемого прецедента.

8.2.2 Проектирование диаграммы классов.

Проектирование диаграммы классов начинается с анализа содержимого словаря предметной области, ТЗ и разработанной ранее диаграммы прецедентов. Результатом анализа является синтезированная диаграмма классов, которая ложится в основу разрабатываемой системы на этапе реализации. Для облегчения работ на данном этом этапе иногда представляют двухуровневое представление диаграммы классов. Верхний уровень соответствует первоначальному понимаю структуры будущего приложения, выраженному непосредственно в бизнес-понятиях предметной области (которые в свою очередь были отражены в словаре).

Уровень реализации соответствует представлению системы близком к уровню реализации системы в исходном коде. Тем не менее возможны определенные расхождения, т.к. при реализации могут быть добавлены вспомогательные классы (методы, атрибуты), не учтенные в проектировании. Однако, скорее всего это будут системные атрибуты, методы иногда даже вспомогательные классы и связи между ними. Все проектирование и документирование данного этапа представляется в соответствии с нотацией UML.

Диаграмма классов с комментариями к ней является результатом данного этапа работы.

8.2.3 Проектирование диаграмм видов деятельности

Данный тип диаграмм отображает как пользователи взаимодействуют с системой и между собой. Каждая диаграмма показывает какие действия совершает каждый участник во времени и с какой последовательностью они идут.

8.2.4 Проектирование диаграмм последовательностей

Данный тип диаграмм частично соотносится с этапом реализации. Однако может формироваться и на данном этапе работы над проектом и уточняться позднее. Нотация описания диаграмм стандартная и основана на UML. Основное назначение данных диаграмм показать работу системы во времени. Выносимые модели не должны противоречить ранее созданным диаграммам прецедентов и классов. Степень детализации диаграмм должна соответствовать последовательности вызовов методов (передачи управления) классов, которые взаимодействуют в работе Взаимодействующие классы должны быть соединены соответствующими типами связей на диаграмме классов. По отдельно взятой диаграмме последовательностей (сотрудничества) должна четко прослеживаться логика реализации одного прецедента. Таким образом минимальное число диаграмм последовательностей (сотрудничеств) должно соответствовать числу прецедентов системы.

8.2.5 Проектирование диаграмм состояний

Проектирование данного типа диаграмм должно начинаться с понимания того, что такое состояние и как оно ассоциируется с целевым объектом, для которого она создается. Важность этого следует из того, что не для всех объектов можно разработать данную диаграмму. Диаграмма состояний не проектируется для объектов, состояния которых не различимо в системе или которые могут находится только в дихотомическом состоянии, т.е. в состоянии работает (создан) / не работает (уничтожен, не используется). Явным признаком наличия состояний у объекта является наличие некоторого знакового атрибута (или совокупности атрибутов), который может находиться в некотором значении из определенного диапазона. Каждое значение характеризует объект и отличает конкретным значение состояние от состояния на протяжении всего времени его жизни. Например, атрибут "Статус" объекта "Покупка" будет характеризовать текущее состояние заказа на всем протяжении жизни объекта и может принимать следующие значения: "Оформлена", "Оплачена", "Формируется", "Доставлена" и т.д. Полный перечень состояний объекта определяет все его возможные поведения и закладывается на данном этапе. В тоже время ошибочным следует понимать ситуации, когда в качестве состояния выделяют промежуточные состояния. По этой причине частой ошибкой является выделение состояния для объекта, которая по сути является переходом от состояния к состоянию. Такой переход может быть сколь угодно длительным может даже оставить объект в прежнем состоянии. Для вышеприведенного примера неправильным пониманием состояния будет состояние "Редактируется", т.к. оно зачастую сводится к пониманию того, что существует некоторая экранная форме, в которой пользователь редактирует

состав покупки (добавляет туда товары). Скорее всего, если не оговорено отдельно, такое состояние будет соответствовать переходу объекта из начального состояния, когда пользователь начал формирование покупки путем добавления к ней товаров и нажатия в интерфейсе кнопки "Оформить", в результате такого события (а оно в действительности связывается с переходом объекта в новое состояние и даже отмечается на диаграмме рядом с переходом) объект действительно переходит в состояние "Оформлена".

8.2.6 Проектирование диаграмм развёртывания

Диаграммы развертывания, или применения, - это один из двух видов диаграмм, используемых при моделировании физических аспектов объектно-ориентированной системы. Такая диаграмма показывает конфигурацию узлов, где производится обработка информации, и то, какие компоненты размещены на каждом узле.

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

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

8.3 Реализация системы

Согласно классическому представлению о жизненном цикле разработки ПС реализация идет после этапа проектирования. Однако практика показывает, что календарные сроки данного этап могут быть сдвинуты ближе к началу этапа проектирования, а иногда и до его полного завершения. Некоторые практики разработки ПС даже приветствуют это (XP, Agile и т.п.). Реализация опирается на выбранные до этого стандарты программирования, технологии и платформы. Если система дорабатывается новым функционалом, то как правило при отсутствии какого-либо рефакторинга принимаются стандарты, с применением которых разработка велась ранее.

Ключевым моментом данного этапа является командная разработка функционала системы с дроблением функций системы для реализации ее отдельным участником (программистом). Координацию работ осуществляет старший программист или какой-либо руководитель проекта по части разработки. Эти особенности свойственны сложным проектам, с большим функционалом.

Для данного этапа проектируются (разрабатываются) следующие виды диаграмм: модулей, развертывания.

9. **Пример разработки программной системы**

Исходный код примера доступен на GitHub по адресу:

https://github.com/MAKAR1031/BookStore

9.1 Постановка задачи

Целью данного курсового проекта является разработка программной информационной системы «Интернет магазин книг».

Условно требования к системе разделяются на два вида: функциональные и нефункциональные. Функциональные требования включают в себя список функций, которые должна уметь выполнять система. Нефункциональные требования описывают некоторые особенности будущей системы, к примеру, это может быть указание языка программирования, целевая платформа или платформы, используемые технологии и другие нефункциональные возможности.

Функциональные требования:

- Зарегистрироваться
- Авторизоваться и аутентифицироваться
- Выйти из системы
- Посмотреть каталог книг
- Посмотреть информацию о книге
- Найти книгу
- Посмотреть содержимое корзины
- Добавить книгу в корзину
- Убрать книгу из корзины
- Очистить корзину
- Посмотреть список заказов
- Разместить заказ
- Отметить заказ
- Оплатить заказ
- Отложить заказ
- Добавить книгу в каталог
- Редактировать информацию о книге
- Удалить книгу из каталога
- Подтвердить заказ
- Посмотреть список пользователей
- Добавить пользователя
- Удалить пользователя

- Изменить роль пользователя

Нефункциональные требования:

- Реализация на платформе .NET
- Использование технологию ASP.NET
- Использование шаблона проектирования MVC
- База данных MS SQL
- Использование Entity Framework для работы с данными

9.2 Описание предметной области

Интернет-магазин книг (Book Store) представляет собой сервис для совершения покупок книг с помощью браузера.

Книга (Book) – один из видов печатной продукции: непериодическое издание, состоящее из сброшюрованных или отдельных бумажных листов (страниц) или тетрадей, на которых нанесена типографским или рукописным способом текстовая и графическая (иллюстрации) информация, имеющая объём более сорока восьми страниц и, как правило, твёрдый переплёт. Основными характеристиками являются: название, автор, цена, жанр и количество страниц.

Позиция заказа (Order item) – это информация о том, какая книга и в каком количестве она заказана.

Заказ (Order) объединяет в себе одну и более позиций заказа, а также содержит информацию о клиенте, дате заказа, адресе доставки и текущем состоянии.

Статус заказа (Condition) определяет состояние объекта «заказ», в котором он находится. Значения статуса соотносятся с реальными физическими значениями.

9.3 Сценарий взаимодействия пользователей с системой

Взаимодействие с системой осуществляется посредствам интернет браузера.

Каждый пользователь имеет одну из нескольких ролей, которые определяют, какими возможностями он обладает. Можно выделить следующие роли и их возможности:

- Клиент
  - Посмотреть каталог книг
  - Найти книги
  - Посмотреть информацию о книге
  - Добавить книгу в корзину

- Убрать книгу из корзины
- Очистить корзину
- Посмотреть список заказов
- Разместить заказ
- Отменить заказ
- Отложить заказ
- Оплатить заказ

§ Менеджер по заказам
- Посмотреть список заказов
- Отменить заказ
- Подтвердить заказ

§ Менеджер по книгам
- Посмотреть каталог книг
- Найти книги
- Добавить книгу в каталог
- Посмотреть информацию о книге
- Редактировать информацию о книге
- Удалить книгу из каталога

§ Администратор
- Посмотреть список пользователей
- Создать пользователя
- Удалить пользователя
- Изменить роль пользователя

Также, каждый зарегистрированный пользователь может авторизоваться и аутентифицироваться в системе и выйти из своего аккаунта.

Незарегистрированные пользователи могут просматривать каталог, смотреть информацию о книгах и выполнять поиск книг, а также зарегистрироваться. При регистрации пользователь автоматически получает роль клиента.

9.4 Создание диаграмм UML

9.4.1 Диаграмма прецедентов

На данной диаграмме отображены все прецеденты (бизнес-функции системы) и её актеры (пользователи системы). Каждый прецедент имеет своего инициатора в виде актера. Некоторые прецеденты входят в отношение включения. К примеру, для того чтобы добавить книгу в корзину, нужно сначала открыть каталог с книгами. Прецедент «Оплатить банковской картой» расширяет прецедент «Оплатить заказ», так как они имеют общее поведение, но один является более специфичным по отношению к другому. Актеры могут входить в отношения обобщения. Например, актеры «Сотрудник» и «Администратор» входят в отношения обобщения, администратор может выполнять все прецеденты сотрудника, но также имеет собственные, уникальные.

Типичной ошибкой проектирования данной диаграммы является неверная ассоциация актеров и прецедентов. Например, в данной диаграмме было бы неверным не ассоциировать незарегистрированного пользователя с прецедентом «Авторизоваться и аутентифицироваться», так как пользователь не имеет роли и, соответственно, прав до того, как авторизуется и аутентифицируется.

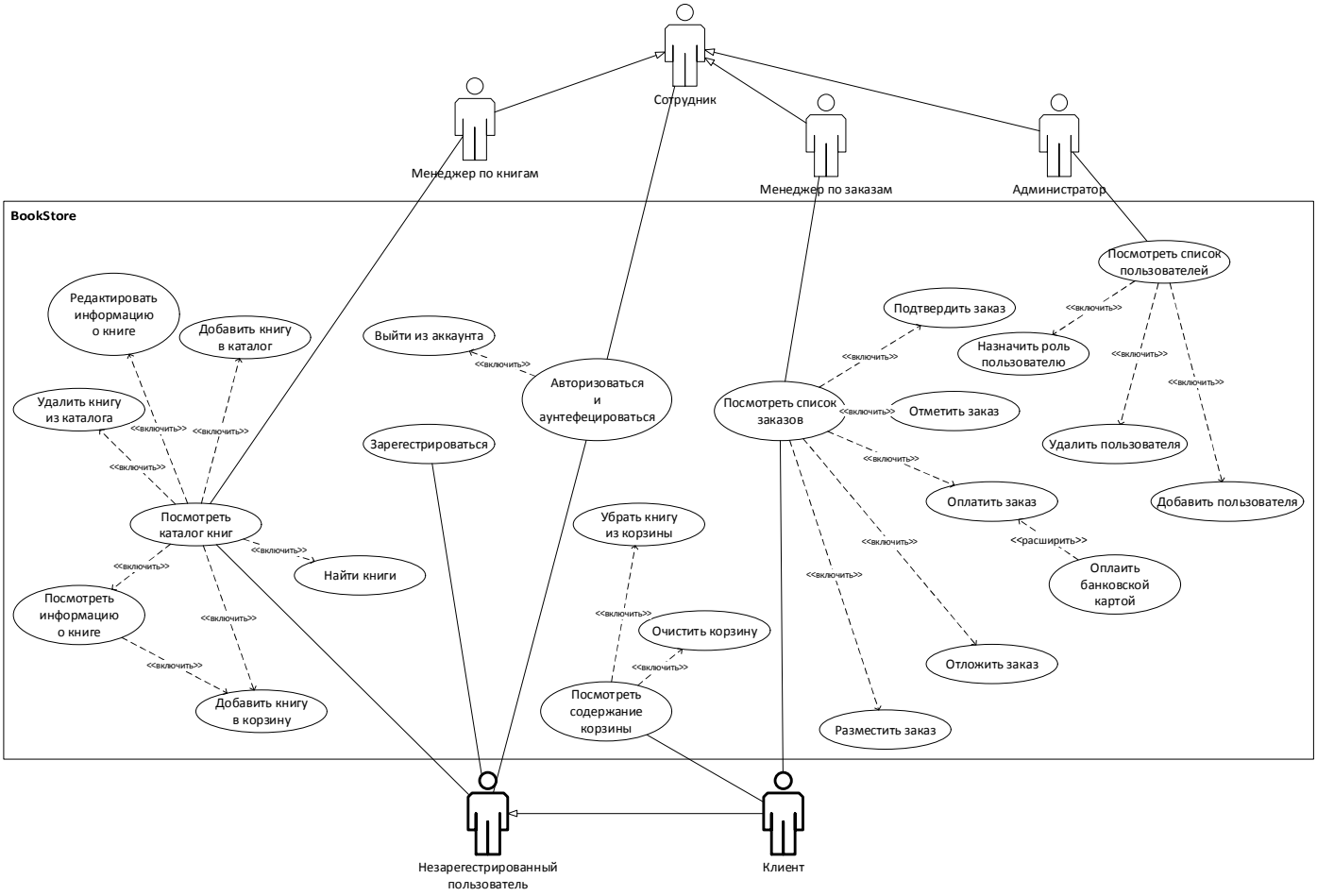

Рисунок 1. Диаграмма прецедентов.

9.4.2 Диаграммы последовательностей

На диаграмме последовательностей система изображается в динамике. Она используется в том случае, если прецедент имеет неочевидное поведение. На ней изображается взаимодействие объектов во времени, время жизни и активности.

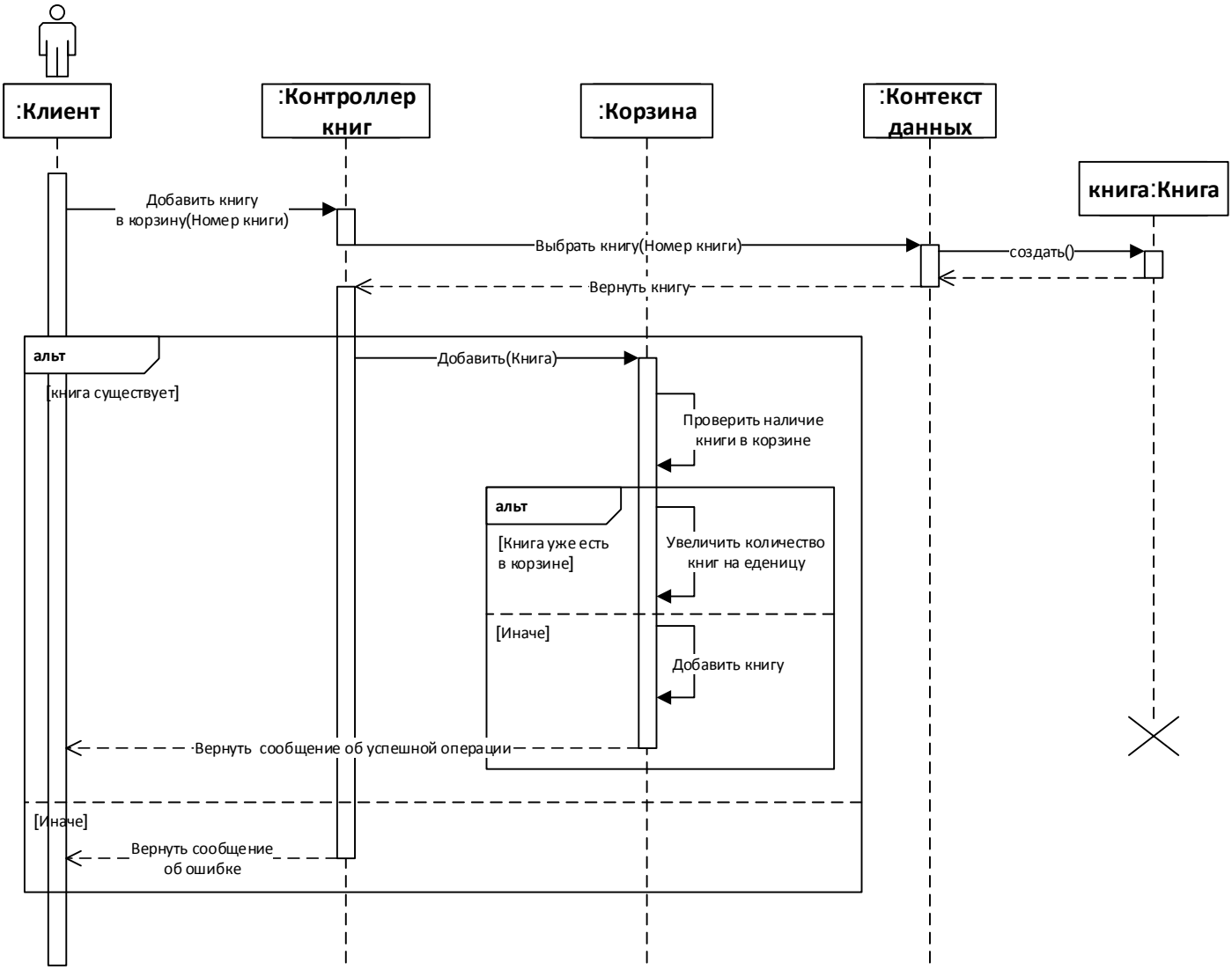

Рисунок 2. Диаграмма последовательностей для прецедента «Добавить книгу в корзину»

9.4.3 Диаграммы видов деятельности

Диаграмма видов деятельности похожа на диаграмму последовательностей, но делает акцент на взаимодействии участников. Здесь также отображается динамическая сторона системы.

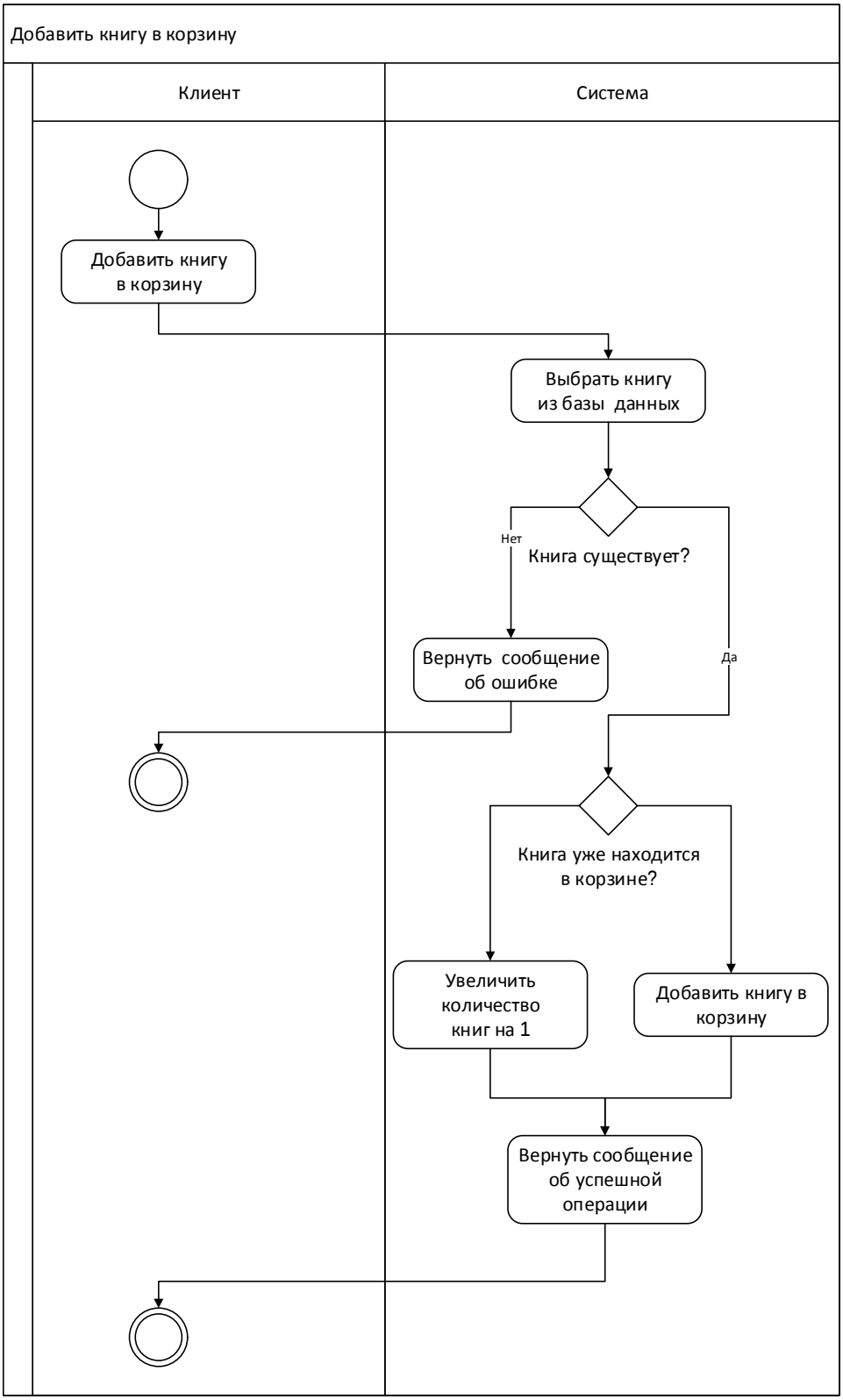

Рисунок 3. Диаграмма видов деятельности для прецедента «Добавить книгу в корзину»

9.4.4 Диаграммы классов

Диаграммы классов могут быть двух уровней. Первый – концептуальный; используется для изображения модели данных будущей системы. Второй – уровень реализации приближен к структуре классов готовой системы. На нем изображаются классы системы, которые тем или иным образом должны реализовывать функциональность системы, прецеденты, модель данных и работу с данными.

Рисунок 4. Диаграмма классов концептуального уровня.

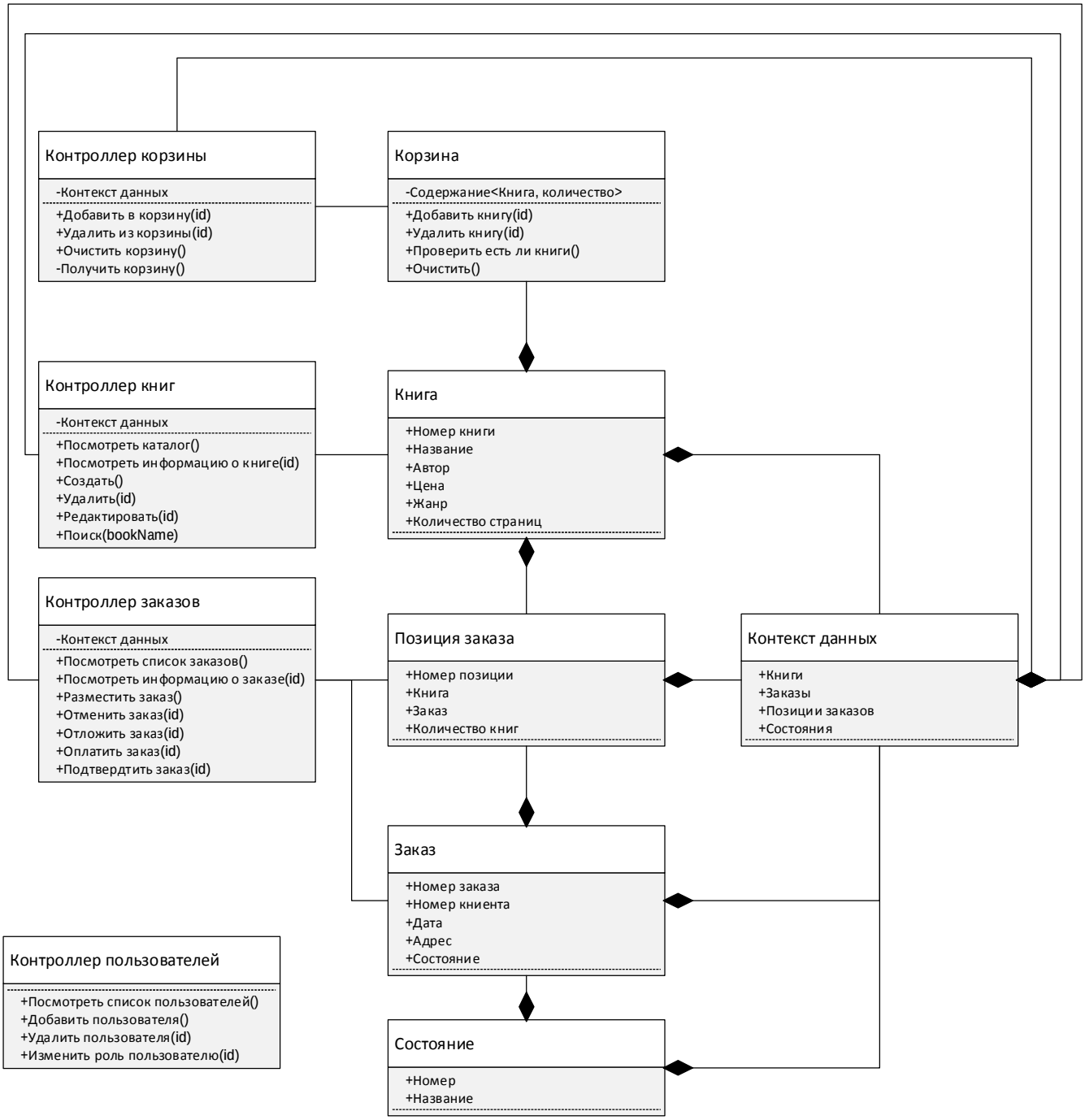

Рисунок 5. Диаграмма классов уровня реализации

9.4.5 Диаграмма состояний

Диаграмма состояний нужна для изображения состояний, в котором может находится бизнес-объект системы.

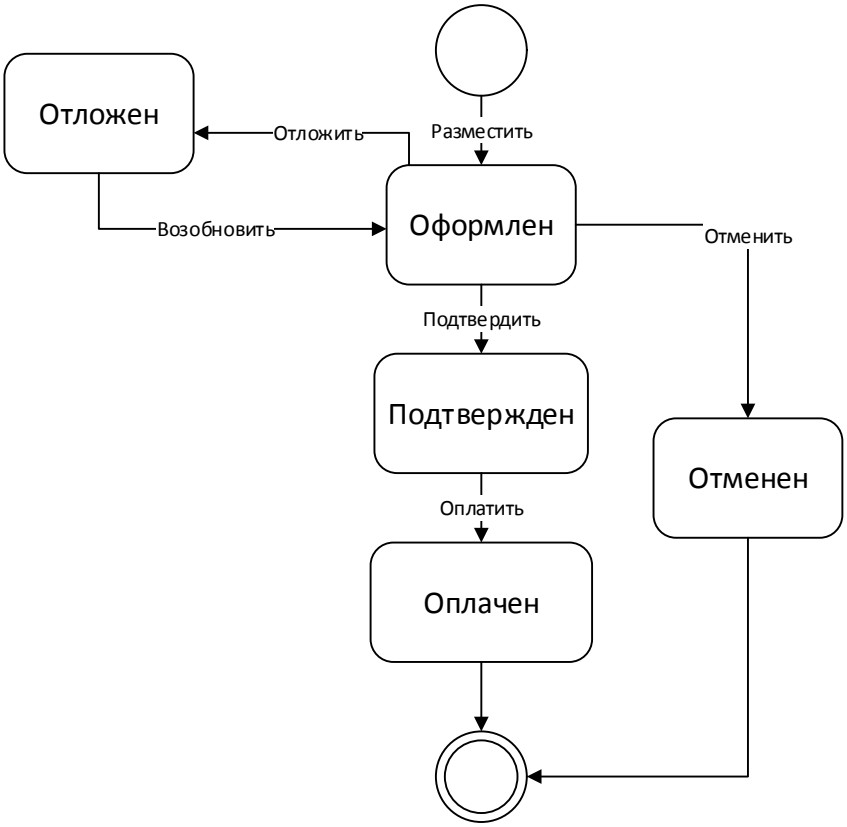

Рисунок 6. Диаграмма состояний заказа.

9.5 Общие принципы организации системы

Разрабатываемая система представляет собой веб-приложение, построенное на базе архитектуры ASP.NET MVC 4 по шаблону «Интернет-приложение».

Как и в любом приложении, реализующее шаблон MVC - ASP.NET MVC 4, содержит всё необходимое для реализации всех трёх компонентов: контроллера, модели и представления.

Каждый контроллер относится к своему классу модели и содержит необходимые методы, которые реализуют бизнес-функции и в результате своей работы возвращают пользователю представление, которое отображает результат выполнения операций.

Представления построены на движке Razor. Razor – это способ преобразования кода на странице в HTML. Преимущество данного способа заключается в удобстве использования, весь код на странице представления начинается с символа «@» и конец блока кода определяется автоматически. Страницы представления Razor имеют расширение «.cshtml».

Для реализации модели будет использован Entity Framework.

9.6 Реализация модели

Entity Framework позволяет использовать любой из трех способов разработки модели данных: Code-First, Model-First, Database-First.

Code-First. Создание модели начинается с написания кода классов-сущностей, а база данных генерируется фреймворком.

Database-First. Сначала создается база данных, а код сущностей генерируется фреймворком.

Model-First.С помощью встроенного редактора диаграмм создается диаграмма сущностей, которая не привязана ни к коду, ни к базе данных. Код и базу данных генерирует фреймворк.

Для реализации данной системы был выбран подход Database-First. Была создана база данных следующего вида:

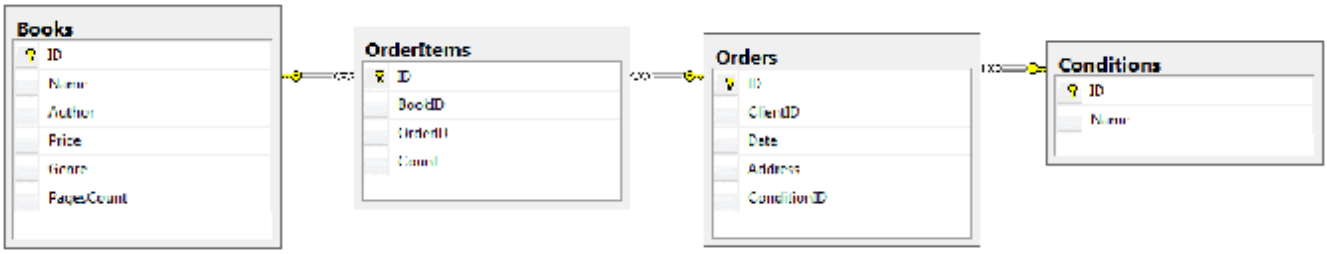

Рисунок 7. Диаграмма базы данных

После её создания в проекте была создана модель ADO.NET EDM на основе этой базы данных. После того, как код был сгенерирован, его необходимо дополнить нотациями, которые нужны для поддержания валидного состояния модели и отображения её свойств.

К примеру, нотация [Required] – означает, что данное свойство обязательно. [Display(Name="")] – определяет название свойства, которое будет видеть пользователь на страницах представления.

Для работы с данными используется экземпляр класса BookStoreContext, наследника класса DbContext, который содержит в себе все сущности. Все данные, при необходимости, подгружаются с помощью фреймворка из базы данных. Все сущности представлены в виде свойств с типом DbSet<Имя класса сущности>.

9.7 Особенности бизнес-логики

При проектировании и разработке системы очень важно учитывать особенности бизнес-логики предметной области.

Для примера рассмотрим один нюанс данной предметной области. Что происходит с заказом после его отклонения? Если данные о заказе будут удаляться из базы данных, то пользователь, заказ которого отменил менеджер по заказам, будет слегка удивлен. Чтобы избежать подобных ситуаций, достаточно менять состояние заказа. Пользователь увидит свой заказ, а также поймет, что он сделал что-то не так и сможет исправить свою ошибку.

9.8 Реализация корзины

Для начала необходимо подготовить модель корзины – класс, объект которого будет хранить состояние корзины. После реализации модели, для взаимодействия с ней, нужен контроллер, который будет предоставлять доступ к интерфейсу корзины. Пример реализации представлен в классе Cart пакета Models и контроллере CartController.

9.9 Реализация механизма регистрации, авторизации и аутентификации

Для реализации работы с пользователями был использован SimpleMembership – стандартный механизм для решения вопросов безопасности приложения.

При создании проекта ASP.NET MVC 4 по шаблону «Интернет-приложение» в него уже входит набор файлов для работы с SimpleMembership:

- /Controllers/AccountController.cs – контроллер работы с учетными записями
- /Filters/InitializeSimpleMembershipAttribute.cs – фильтр инициализации SimpleMembership
- /Models/AccountModels.cs – содержит модели аккаунтов

SimpleMembership использует Entity Framework для работы с базой данных.

Для настройки нужно добавить строку подключения в файле Web.config и в файле InitializeSimpleMembershipAttribute.cs найти строку следующего вида:

```
WebSecurity.InitializeDatabaseConnection("<Название вашей строки подключения>",
"<Имя таблицы с пользователями>",
"<Первичный ключ таблицы с пользователями>",
"<Название поля, в котором хранится имя пользователя(логин)>",
autoCreateTables: true|false);
```

После настройки фильтра необходимо производить его инициализацию. Для этого каждый контроллер, который будет использовать механизм безопасности должен быть аннотирован, перед строкой декларации класса контроллера помещается следующая запись [InitializeSimpleMembership]. После этого уже можно заняться вопросом предоставления доступа. Аннотация [Authorize], которая может быть использована, как для аннотирования целого контроллера, так и отдельного метода. С помощью [AllowAnonymous] можно разрешить доступ неавторизованным пользователям.

После первого запуска в указанной базе данных должны появиться таблицы

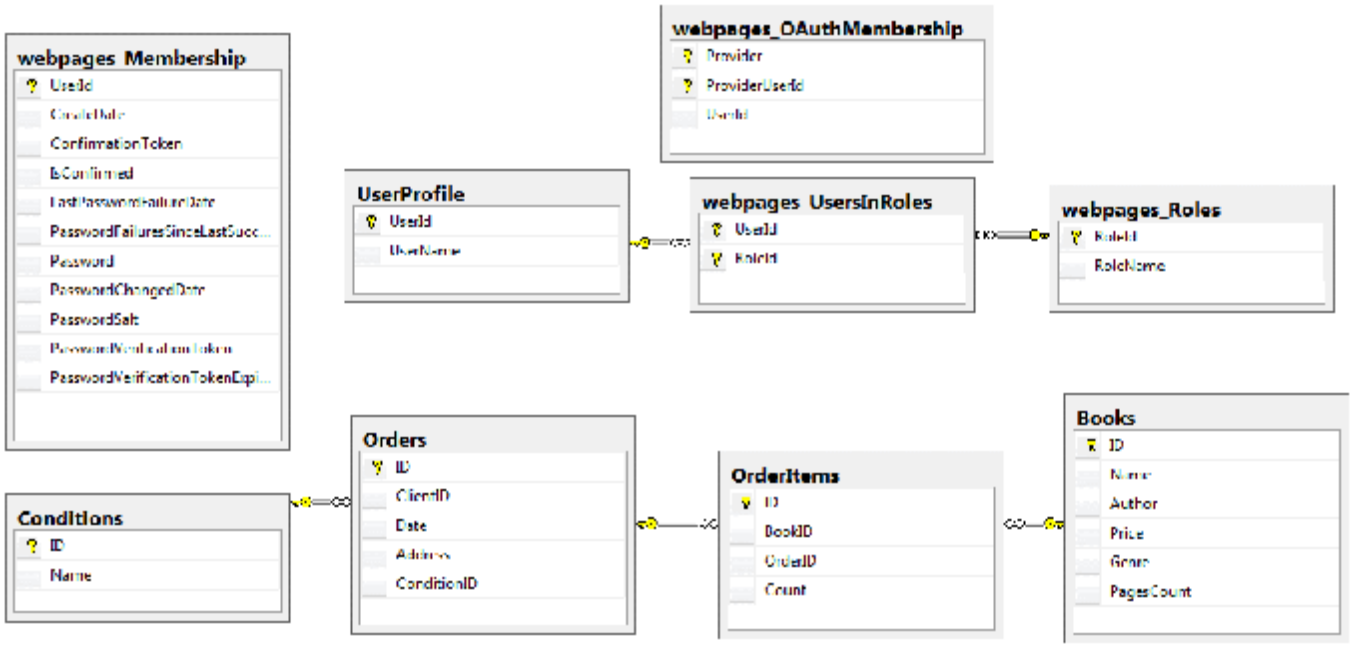

Рисунок 8. Диаграмма базы данных после автоматического создания таблиц.

9.10 AJAX

AJAX – технология для асинхронных запросов к серверу. Позволяет не перезагружать страницу полностью, а обновлять/изменять лишь её часть. Преимуществами данного подхода являются: экономия трафика за счет уменьшения количества загружаемых данных; уменьшение нагрузки на сервер и увеличение отзывчивости интерфейса.

Для подключения AJAX нужно добавить следующие строки на страницу представления или на мастер-страницу, если AJAX будет использоваться почти на каждом представлении:

```
<script type="text/javascript" src="~/Scripts/jquery-[версия].min.js"></script>

<script type="text/javascript" src="~/Scripts/jquery.unobtrusive-ajax.min.js"></script>
```

После этого можно будет пользоваться хелпером Ajax и всеми его методами. Самые часто используемые методы это ActionLink и BeginForm для встраивания Ajax ссылок и форм соответственно.

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

Кафедра информационных систем и программной инженерии

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту по дисциплине
"Технологии программирования"
на тему

Проектирование и разработка программной системы
информационной системы «Магазин»

Выполнил:

Принял: *доц. кафедры ИСПИ*
Верешинин В.В.

Владимир, 2016

<УТВЕРЖДЕННЫЙ ЛИСТ ЗАДАНИЯ НА КУРСОВОЙ ПРОЕКТ>

АННОТАЦИЯ

Аннотация составляется на русском и английском языках. Это краткая характеристика работы, ее содержимого. В конце указывается свойства документа (пояснительной записки).

Курсовой проект представлен на 56 страницах, рисунков – 5, таблиц – 11, использованных источников – 4, приложений – 2, иллюстрационный материал на 3 листах формата А1.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Текст раздела оформляется шрифтом Times New Roman, 14 пунктом, полуторный интервал.

1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1.Словарь предметной области

Ссылки на рисунок, представленный на рисунке 1, оформляется в виде ссылки в тексте и подписи под рисунком. Те же требования и для таблиц. Текст рисунков/таблиц должен быть четким и ясным, легко читаемым.

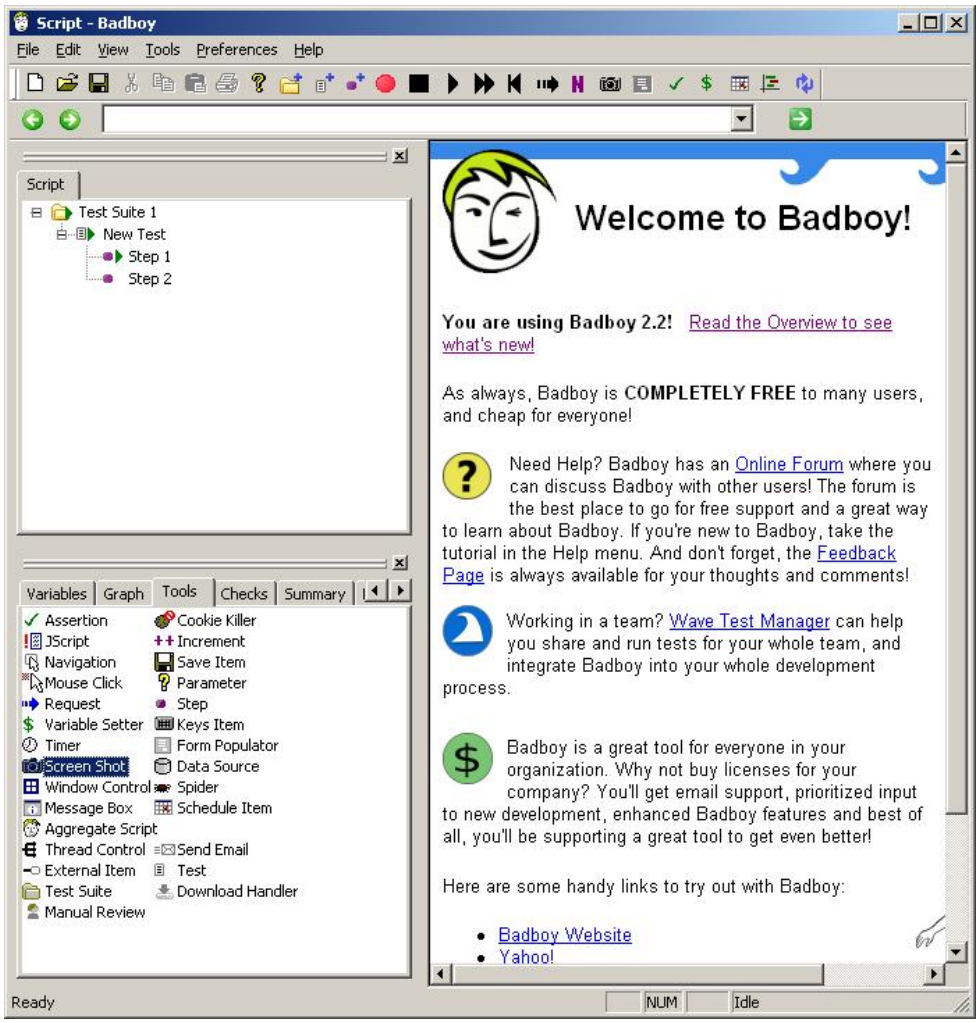

Рисунок 1. Внешний вид окна приложения

Списки в тексте должны быть нумерованными:

1. после номера пункта текст начинается с маленькой буквы;
2. в конце ставится точка с запятой, если пункт не последний;
3. в конце ставится точка, если пункт последний.

2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ

Описание всех решений, использованных в проекте. Это не должен быть реферат, а описание конкретных проектных решений. На все то, что используется в работе должны быть ссылки на соответствующие литературные источники. Ссылки на источник оформляются в прямоугольных скобках внутри которых номер источника. Например, вот так [5].

ЗАКЛЮЧЕНИЕ

Основные выводы по работе и достигнутые результаты. Достаточно трех четырех абзацев текста.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Липаев В.В. Проектирование программных систем. М.: Высш. шк, 1990.

2. Буч Г. Объектно-ориентированное проектирование / Пер. с анг. Конкорд, 1996.

3. Майерс Г. Надежность программного обеспечения. М.: Мир, 1980

4. <название статьи/книги/материала> [Электронный ресурс]: <автор>. – Режим доступа: <полный URL>

ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММНОЙ СИСТЕМЫ

```
package ru.home.lt.cf;

import lrapi.lr;
import org.apache.commons.configuration.ConfigurationException;
import org.apache.commons.configuration.XMLConfiguration;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class UC01CreateApplication extends UseCase{

    final static private Logger logger =
LoggerFactory.getLogger(UC01CreateApplication.class);


    public static Double createILSFPercentage = 82.82;
    public static Double createCWFPPercentage = 10.80;
    public static Double createLLQ1Percentage = 5.21;
    public static Double createCORNPercentage = 0.78;
    public static Double createCCLPPercentage = 0.39;

    private UC01CreateILSF ilsf;
    private UC01CreateCWFP cwfp;
    private UC01CreateLLQ1 llq1;
    private UC01CreateCORN corn;
    private UC01CreateCCLP cclp;

    static {
        XMLConfiguration config = null;
        try {
            config = new XMLConfiguration("../load-test.xml");
        } catch (ConfigurationException e) {
            logger.error(e.getLocalizedMessage());
        }


        createILSFPercentage =
config.getDouble(LoadTestConfiguration.CREATE_ILSF_APPLICATION_PERCENTAGE,
82.82);
        createCWFPPercentage =
config.getDouble(LoadTestConfiguration.CREATE_CWFP_APPLICATION_PERCENTAGE,
10.80);
        createLLQ1Percentage =
config.getDouble(LoadTestConfiguration.CREATE_LLQ1_APPLICATION_PERCENTAGE, 5.21);
        createCORNPercentage =
config.getDouble(LoadTestConfiguration.CREATE_CORN_APPLICATION_PERCENTAGE, 0.78);
        createCCLPPercentage =
config.getDouble(LoadTestConfiguration.CREATE_CCLP_APPLICATION_PERCENTAGE, 0.39);

        LogProductPercentages();
    }
```

ПРИЛОЖЕНИЕ Б СТРУКТУРА ТАБЛИЦ БД

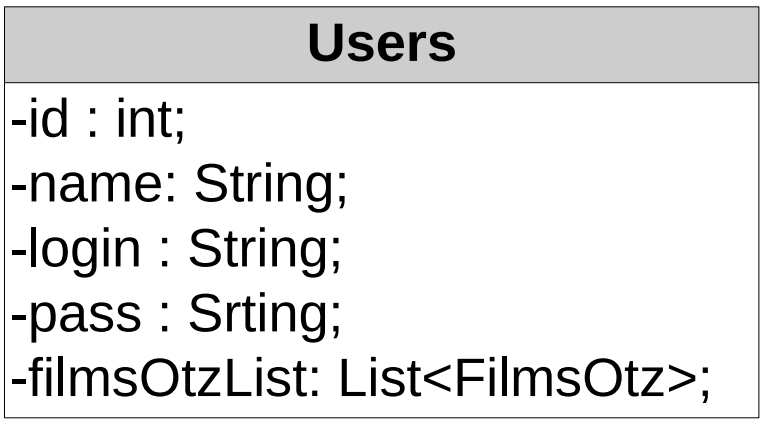

Рисунок Б.1. Структура таблицы "Пользователи"

КАРМАН С ДИСКОМ НА КОТОРОМ ЗАПИСАНЫ МАТЕРИАЛЫ КУРСОВОГО ПРОЕКТИРОВАНИЯ:

1. пояснительная записка
2. проект системы с исходными кодами
3. дополнительные компоненты системы, необходимые для ее развертывания
4. прочие вспомогательные источники
5. презентация по которой будет проходить защита работы

ПАПКА(ФАЙЛИК) ДЛЯ ГРАФИЧЕСКОГО МАТЕРИАЛА К ПОЯСНИТЕЛЬНОЙ ЗАПИСКЕ И САМ ГРАФИЧЕСКИЙ МАТЕРИАЛ В НЕЙ