МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
(МИНОБРНАУКИ)

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»
(СПбГУ ИТМО)

УТВЕРЖДАЮ
Ректор СПбГУ ИТМО,
докт. техн. наук, профессор
В. Н. Васильев

____ _____ 2006 г.

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл.	

ТЕХНОЛОГИЯ АВТОМАТНОГО ПРОГРАММИРОВАНИЯ: ПРИМЕНЕНИЕ И ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ И УКАЗАНИЯ ПО РАЗРАБОТКЕ, БАЗИРУЮЩИЕСЯ НА ПРИНЦИПАХ АВТОМАТНОГО ПРОГРАММИРОВАНИЯ,

ЧАСТЬ 5. ИНТЕРНЕТ-СИСТЕМЫ

Листов 43

Декан факультета «Информационные технологии и программирования»
докт. техн. наук, профессор
_____В.Г. Парфенов

Руководитель темы
заведующий кафедрой «Технологии программирования», докт. техн. наук, профессор
_____ А.А. Шалыто

Ответственный исполнитель
доцент кафедры «Технологии программирования», канд. физ.-мат. наук
_____Ф.А. Новиков

2006

СПИСОК ОСНОВНЫХ ИСПОЛНИТЕЛЕЙ

Руководитель темы докт. техн. наук, профессор	А.А. Шалыто
Ответственный исполнитель канд. физ.-мат. наук, ст. научн. сотр.	Ф.А. Новиков
Ведущие исполнители:	
Магистрант	М.А. Коротков
Магистрант	А.П. Лукьянова
Ведущий научный сотрудник, д.т.н., профессор	В.В. Антипов
Ведущий научный сотрудник, к.т.н.	В.В. Киселев
Ведущий научный сотрудник, к.т.н.	Р.Н. Котляр
Ведущий научный сотрудник, к.т.н.	Ю.П. Московцев
Ведущий научный сотрудник, к.т.н., доцент	В.А. Третьяков
Ведущий научный сотрудник, к.т.н.	Г.М. Файкин
Ассистент, к.т.н.	Д.Г. Шопырин
Аспирант	И.М. Аничкин
Аспирант	В.С. Гуров
Аспирант	М.А. Казаков
Аспирант	Г.А. Корнеев
Аспирант	П.Г. Лобанов
Магистрант	М.А. Мазин
Студент	Н.И. Поликарпова
Студент	Б.М. Ярцев

АННОТАЦИЯ

Описывается методика создания интернет-систем, разработанная в процессе проведения опытно-конструкторских работ по **теме (Шифр ИТ-13.4/004): «Технология автоматного программирования: применение и инструментальные средства» (VI очередь),** выполняемой в рамках Федеральной целевой научно-технической программы «Исследования и разработки по приоритетным направлениям развития науки и техники» на 2002-2006 годы по государственному контракту № 02.435.11.1009 от 01.08.2005, заключенному между Федеральным агентством по науке и инновациям и Государственным образовательным учреждением высшего профессионального образования «Санкт-Петербургский государственный университет информационных технологий, механики и оптики» на основании решения Конкурсной комиссии Роснауки № 3 (протокол от 01 августа 2005 г. № 17).

Приведено общее описание методики, описаны основные понятия, элементы и положения автоматного программирования для рассматриваемого класса задач. Описаны десять этапов методики: написание технического задания; моделирование вариантов использования системы; выделение сущностей предметной области; создание функционального прототипа; выбор программной платформы; проектирование; разработка; развертывание приложения; тестирование производительности; создание инструкции по использованию, а при необходимости и проектной документации.

Применение методики описано по шагам на примере создания интернет-системы для автоматизации заказов книжного склада.

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

Настоящая методика разработана в процессе проведения опытно-конструкторских работ по **теме (Шифр ИТ-13.4/004): «Технология автоматного программирования: применение и инструментальные средства» (VI очередь),** выполняемой в рамках Федеральной целевой научно-технической программы «Исследования и разработки по приоритетным направлениям развития науки и техники» на 2002-2006 годы по государственному контракту № 02.435.11.1009 от 01.08.2005, заключенному между Федеральным агентством по науке и инновациям и Государственным образовательным учреждением высшего профессионального образования «Санкт-Петербургский государственный университет информационных технологий, механики и оптики» на основании решения Конкурсной комиссии Роснауки № 3 (протокол от 01 августа 2005 г. № 17).

В рамках указанных опытно-конструкторских работ подготовлены методические рекомендации и указания по разработке, базирующиеся на принципах автоматного программирования, для создания программных комплексов в следующих областях приложений:

a) системы управления с повышенными требованиями к надежности функционирования;
b) встроенные системы;
c) мобильные системы;
d) клиент-серверные системы;
e) интернет-системы;
f) визуализаторы алгоритмов;
g) тренажеры;
h) симуляторы.

В данном документе изложена пятая часть методических рекомендаций и указаний, а именно, методические рекомендации и указания по разработке для интернет-систем. Применение методики иллюстрируется на примере создания интернет-системы для автоматизации книжного склада.

2. НАЗНАЧЕНИЕ, ОБЛАСТЬ ПРИМЕНЕНИЯ И ОСНОВНЫЕ ПОЛОЖЕНИЯ МЕТОДИКИ

Описываемая методика предназначена для разработки интернет-систем. Она базируется на принципах автоматного программирования и является расширением методики разработки интернет-систем на базе архитектур, управляемых моделями [1, 2].

В данной методике автоматное программирование применяется для описания поведения приложения.

2.1. Основные положения методики

Разработка и реализация интернет-системы в рамках предлагаемой методики состоит из следующих этапов:

- написание технического задания;
- моделирование вариантов использования системы;
- выделение сущностей предметной области;
- создание функционального прототипа;
- выбор программной платформы;
- проектирование и разработка
  - базы данных;
  - логики реализации;
  - бизнес-логики;
- развертывание приложения;
- тестирование производительности;
- создание инструкции по использованию, а при необходимости и проектной документации;

Решать указанные подзадачи предлагается на основе различных технологий, о которых будет сказано ниже.

Написание технического задания. Техническое задание в текстовой форме считается подготовленным и согласованным.

Моделирование вариантов использования системы (*Use Cases*). По текстовой форме технического задания исследуются варианты использования системы [3, 4]. В качестве одного из подходов возможна подготовка сценариев использования (*user stories*) [5, 6]. Основным артефактом, подготавливаемым на этой фазе, является диаграмма вариантов использования системы.

Выделение сущностей предметной области. По текстовой форме технического задания строится диаграмма сущностей предметной области. На данной фазе возможно принятие решений о том, какие сущности хранятся в базе данных, а какие – на сервере приложений [7].

Создание функционального прототипа. На данной фазе, базируясь на имеющейся информации, строится функциональный прототип, содержащий следующую информацию:

- основные интернет-страницы приложения;
- основные элементы управления для каждой страницы;
- комментарии к поведению сложных элементов управления.

Функциональный прототип может строиться либо в среде разработки *web*-страниц, либо в среде, приспособленной для описания интерфейсов.

Выбор программной платформы. Учитывая информацию о масштабе системы, потребностях в дальнейшем масштабировании и нагрузке на систему производится выбор программной платформы. Требования к программной платформе могут также диктоваться заказчиком.

Проектирование. На данном этапе производится проектирование системы. По каждому из направлений производится разработка соответствующего компонента, являющегося частью системы. Строится диаграмма компонентов, отражающая разбиение системы на слои.

Проектирования базы данных. На основе модели предметной области разрабатывается модель базы данных. Выбирается механизм задания связей и реализации наследования [8].

Проектирования логики приложения. При проектировании логики приложения применяется автоматный подход. Конечные автоматы используются для управления приложением и обрабатывают события от браузера и других поставщиков событий. Применение конечных автоматов в этом случае позволяет значительно ускорить процесс проектирования логики приложения и увеличить наглядность документации, описывающей эту логику. Применения средств разработки, реализующих принцип автоматного программирования [9, 10] позволяет также сократить зазор между проектированием и разработкой логики приложения.

На данном этапе необходимо:

- выделить множество конечных автоматов, осуществляющих управление независимыми частями приложения (если таковых несколько);
- выделить поставщики событий;
- выделить множества событий;
- выделить объекты управления;
- описать поведение каждой независимой части приложения в виде графа переходов.

Проектирования бизнес-логики. Здесь осуществляется разработка бизнес-логики приложения. Наиболее обоснованным кажется применение на этом уровне технологий *EJB* (в случае выбора *Java*-платформы) [11]. Строятся диаграммы классов, осуществляющих доступ к данным и их модификацию. Описываются нетривиальные механизмы и алгоритмы, если таковые присутствуют.

Развертывание приложения. Разработанное приложение разворачивается в тестовом окружении для его тестирования. Строится диаграмма развертывания.

Тестирование производительности. Производится тестирование производительности системы в зависимости от нагрузки на нее [12, 13]. Для интернет-систем нагрузка обычно определяется как количество одновременных запросов к системе или как количество пользователей, использующих систему одновременно. Строятся графики зависимости времени отклика и нагрузки на сеть от количества одновременных запросов.

Создание инструкции по использованию. Разрабатывается инструкция по эксплуатации системы, а при необходимости и проектная документация.

3. ПРИМЕР ПРИМЕНЕНИЯ МЕТОДИКИ

В качестве примера применения методики рассмотрим разработку интернет-системы для автоматизации заказов книжного склада. Данный пример является показательным, поскольку задача принадлежит к классу типичных интернет-систем.

3.1. Техническое задание

Требуется разработать систему заказов для книжного склада, на котором хранятся книги различных издательств. Предполагается, что на удаленном сервере находится база данных с информацией о книгах (название, автор, издательство, стоимость и т. д.), которая может изменяться со временем.

Задачами системы являются:

- поддержка возможностей обновления и коррекции информации в базе данных (внесение новых книг, изменение цен, и.т.п.). Для этого необходимо разработать специальное приложение администратора;
- выполнение заказов клиентов. Клиенту предлагается отобрать интересующие его издания в специальную “корзину”. После этого он может оформить заказ на отобранные издания.

Система должна быть разработана с учетом приведенных ниже масштабов данных.

В табл. 1 указаны используемые сущности.

Таблица 1

№	**Сущность**	**Масштаб**
1	Издательства	10-100
2	Книги (издания)	1000-100000
3	Книги (экземпляры издания)	10-10000
4	Заказы (необработанные)	10-100
5	Виды книг в заказе	10-100
6	Экземпляры книг каждого вида в заказе	100-1000

Замечание: обработанные заказы рассматриваются как история системы.

В табл. 2 указаны пользователи подсистем

Таблица 2

#	Подсистема	Масштаб	Комментарии
1	Приложение администратора	1	Не требуются системы авторизации и разделения прав доступа для администраторов. Такой контроль будет производится средствами операционной системы.
2	Приложение клиента	10-1000	

Кроме этого, необходимо заложить в систему возможность масштабирования на большие объемы данных и нагрузки.

3.2. Моделирование вариантов использования системы

Выделим две категории пользователей системы:

- внешний пользователь (*Internet User*);
- внутренний пользователь – администратор (*Intranet User*).

Построим диаграмму вариантов использования системы, отметив на ней основные действия, которые могут совершать эти категории пользователей (рис. 1).

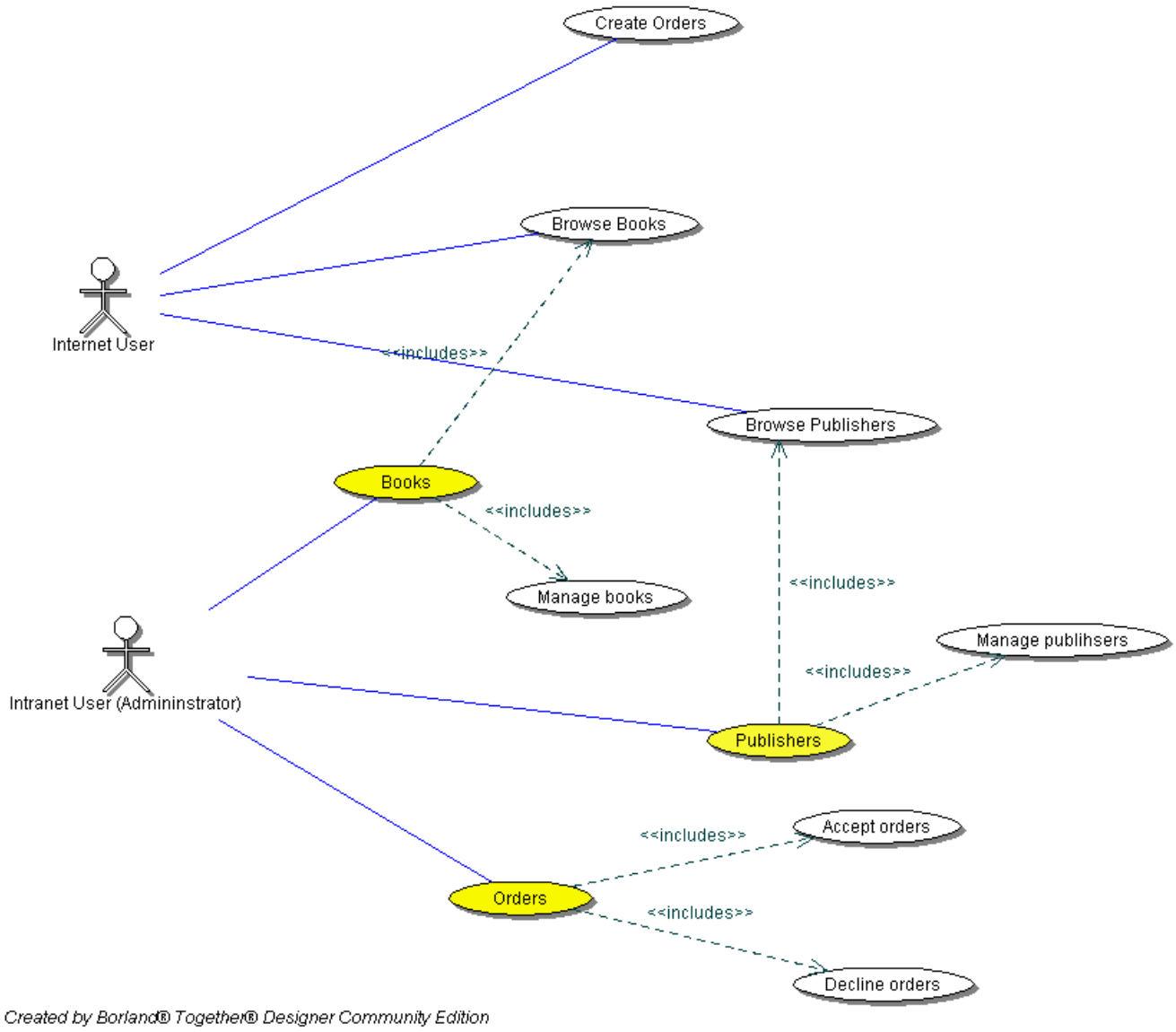


Рис. 1. Варианты использования

3.3. Выделение сущностей предметной области

Построим диаграмму сущностей предметной области. Для этого выделим указанные сущности и связи между ними (рис.2).

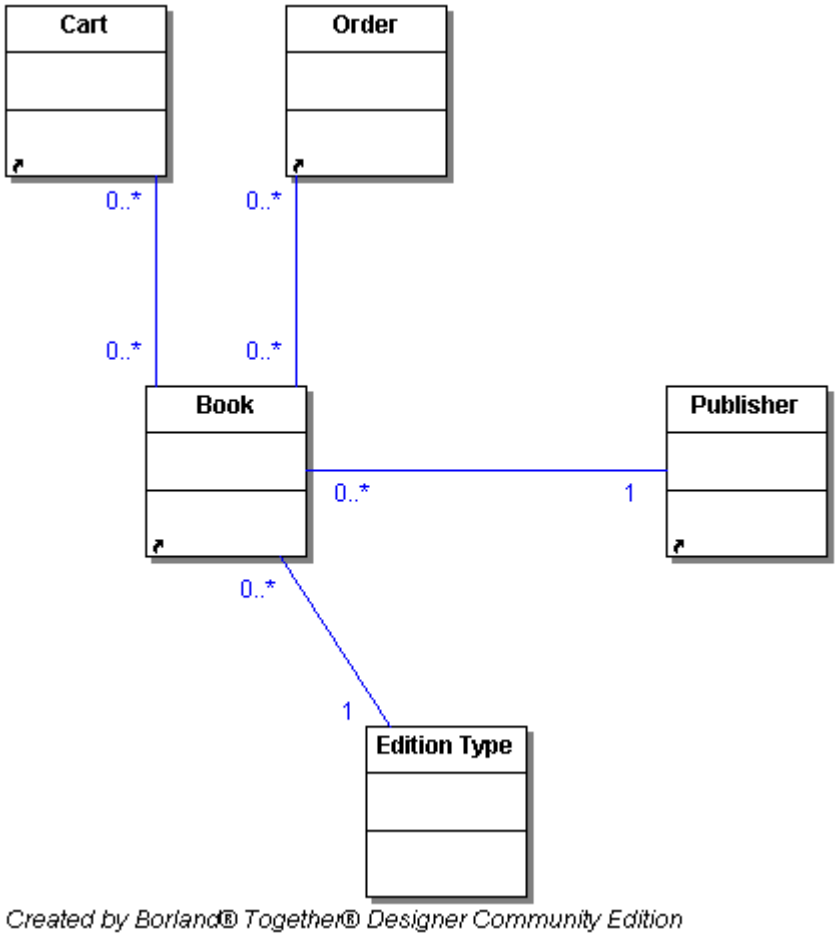


Рис. 2. Диаграмма сущностей предметной области

Отметим некоторые особенности данной диаграммы.

Во-первых, на основании знаний о предметной области появилась новая сущность – тип издания (*edition type*). Эта информация не отражена в техническом задании. В случае, если бы указанная информация была априори неизвестна, эту сущность, возможно, пришлось бы позднее ввести для нормализации базы данных [8].

Во-вторых, на диаграмме имеются две сущности, отвечающие за заказ – корзина (*Cart*) и заказ (*Order*). Это необходимо для наглядности. Заказ представляет собой корзину, сохраненную в базе данных (корзина и заказ содержат одну и ту же информацию). Корзина является единственной сущностью на диаграмме, которая не будет сохранена в базе данных.

Таким образом, стало известно, с какими сущностями должна работать разрабатываемая система.

3.4. Создание функционального прототипа

Для разработки прототипа воспользуемся возможностями среды *Microsoft Visio* [14]. Прототип данного проекта не будет отражать дизайн, но будет содержать функциональные макеты всех страниц приложения с описаниями особенностей поведения этих страниц.

3.4.1. Приложение клиента

На стартовой странице приводится приглашение, список ссылок и общая информация о назначении системы (рис. 3).

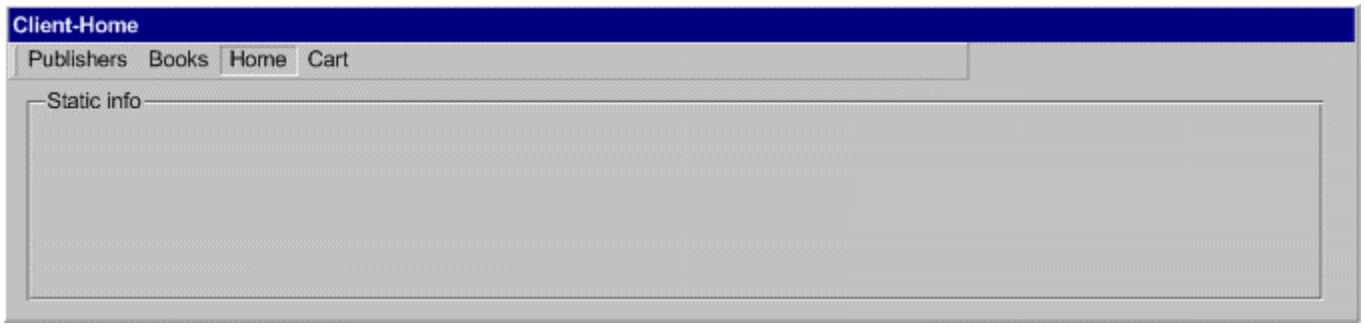


Рис. 3. Стартовая страница приложения клиента

Список издательств позволяет получить информацию об издательствах и перейти к списку книг конкретного издательства (рис. 4). Также возможна фильтрация списка по названию и сортировка по полям.

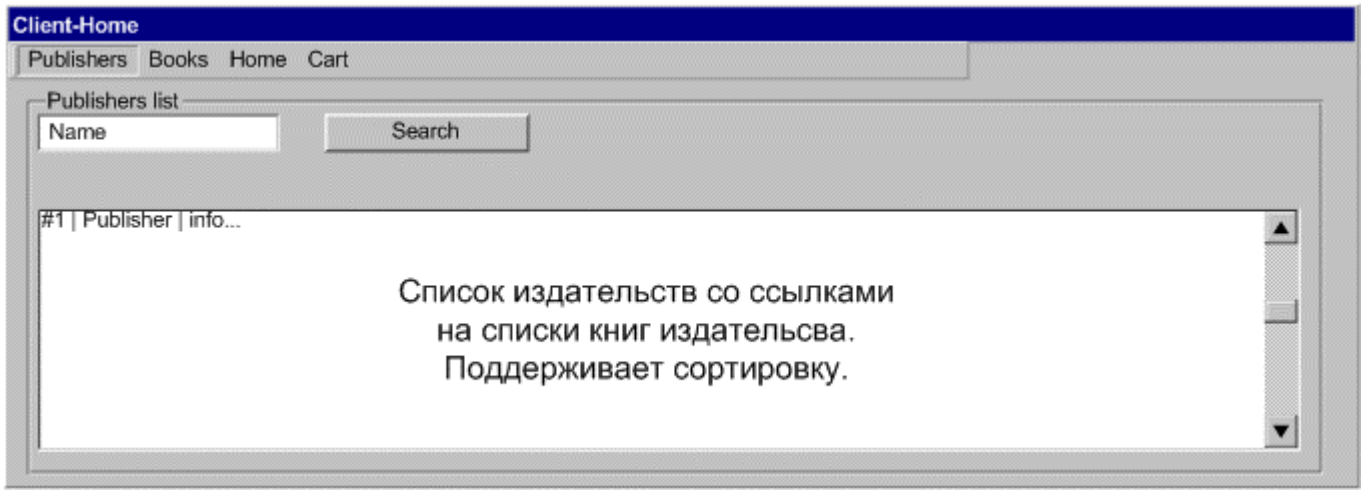


Рис. 4. Список издательств

Список книг позволяет добавить к заказу произвольное количество книг, ограниченное только наличием доступных (еще не заказанных) книг на складе (рис. 5). Возможна фильтрация (поиск) по издательствам, названию, автору, а также сортировка по полям.

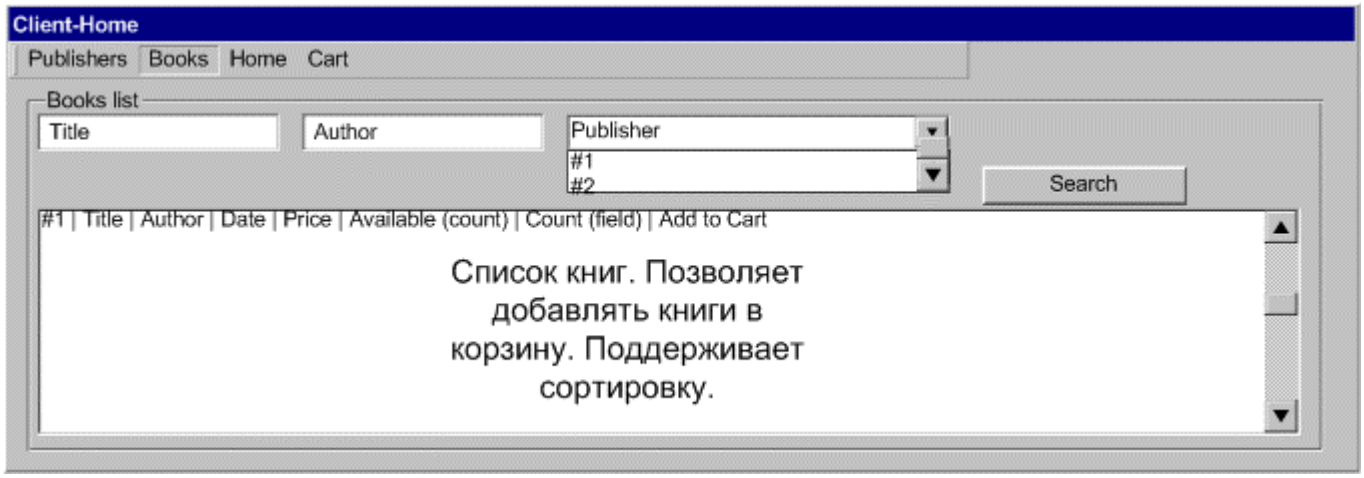


Рис. 5. Список книг

Страница заказа позволяет скорректировать количество экземпляров каждой книги, ввести контактную информацию и подать заказ (рис. 6). При подаче заказа книги блокируются.

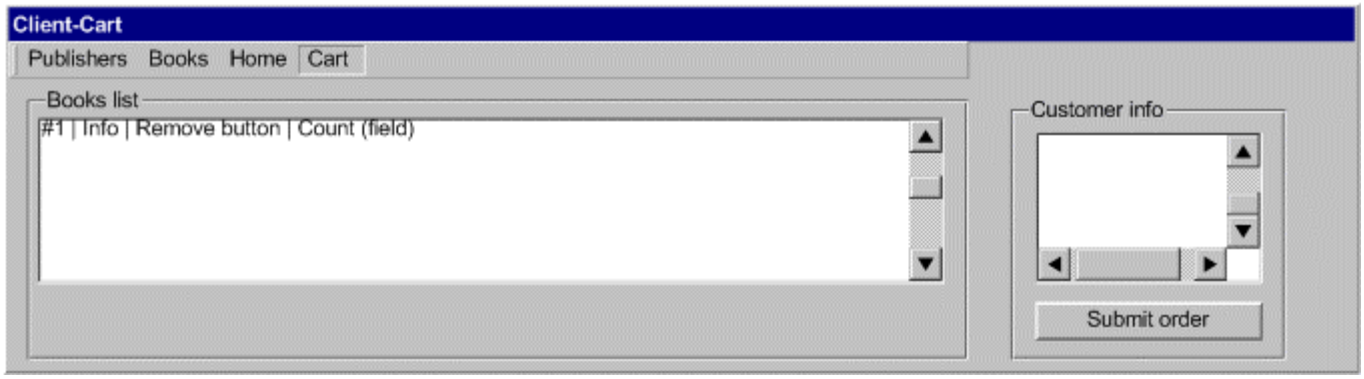


Рис. 6. Клиент. Корзина

3.4.2. Приложение администратора

На стартовой странице администратора приводится приглашение, список ссылок и общая информация о назначении системы (рис. 7).

Список издательств, кроме функций клиента, позволяет удалять, добавлять и редактировать информацию об издательствах.

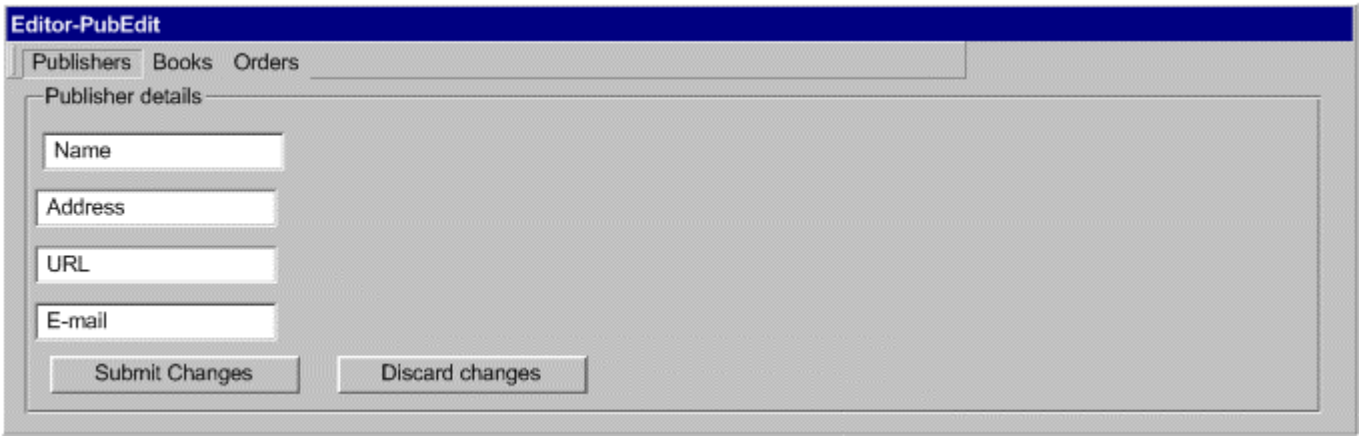


Рис. 7. Администратор. Редактирование издательства

Список книг, кроме функций клиента, позволяет удалять, добавлять и редактировать информацию о книгах (рис. 8).

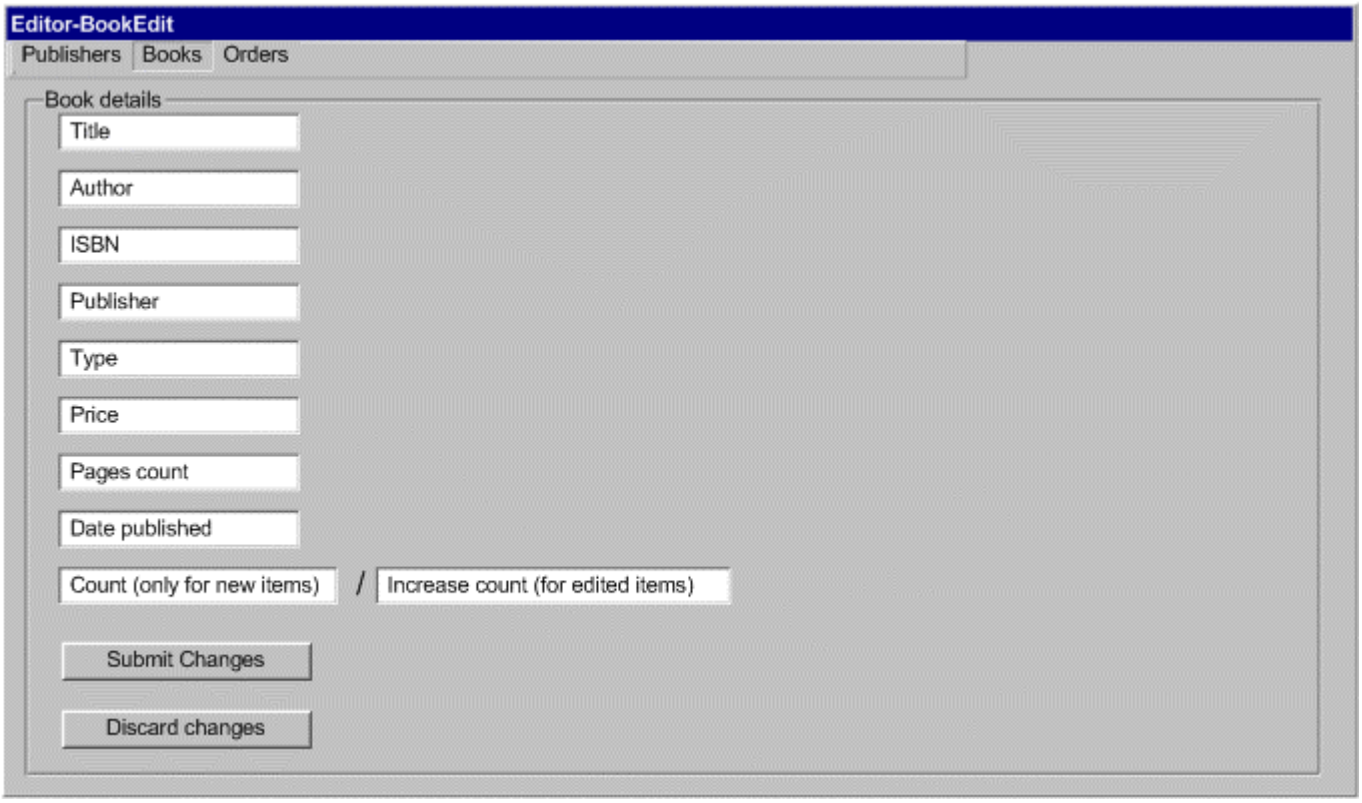


Рис. 8. Администратор. Редактирование информации о книгах

При редактировании информации о книге для новых книг указывается количество, а для существующих – поступление. Уменьшение количества книг производится только посредством заказа (рис. 9). Списание книг выполняется посредством фиктивного заказа (в информации о покупателе указывается причина списания, отдельный интерфейс для списания не предусмотрен). Список заказов приводится совместно с информацией о заказчике, возможностью сортировки и возможностью перехода на страницу принятия или отклонения заказа.

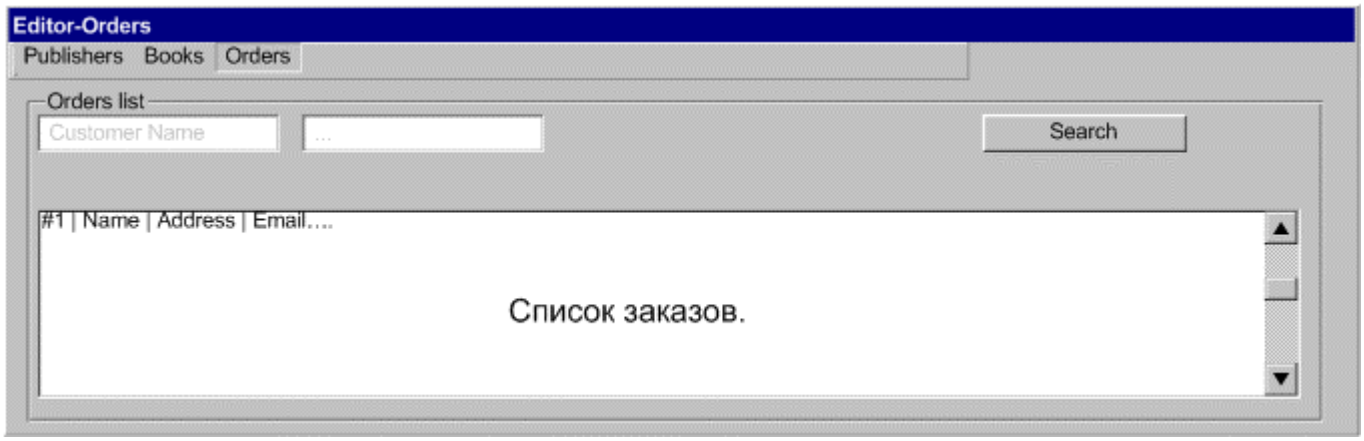


Рис. 9. Администратор. Список заказов

На странице заказа у администратора есть возможность прочитать контактную информацию заказчика, при необходимости связаться с ним, а затем принять или отклонить заказ (рис. 10).

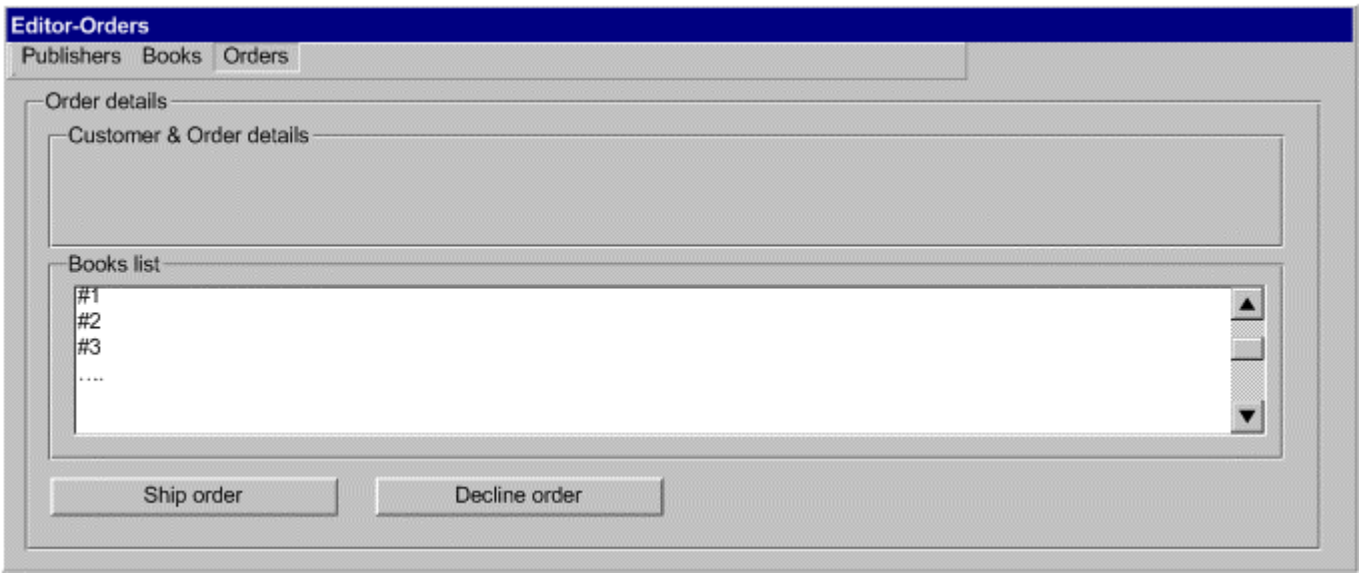


Рис. 10. Администратор. Заказ

3.5. Выбор программной платформы

Для данного проекта выбирается следующая программная платформа:

- база данных – *Oracle* [8]; это объясняется тем, что необходимо обеспечить возможность масштабирования системы в широких пределах;
- язык разработки – *Java* [15];
- технология реализации бизнес-логики, которая в данном случае не является автоматной – *EJB* [16];
- технология обработки запросов – *Java Servlets* [17];
- технология создания шаблонов страниц – *JSP* [18].

Выбор языка разработки *Java* и упомянутых выше технологий обоснован тем, что они обеспечивают возможность быстрой и удобной разработки надежных приложений.

3.6. Проектирование и разработка

Рассмотрим разбиение системы на слои (уровни).

В системе можно выделить три слоя:

1. База данных;
2. Сервер приложений;
3. Клиент.

На уровне базы данных будет развернуто приложение базы данных, обеспечивающее надежное хранение информации и контроль целостности.

На уровне сервера приложений будут развернуты два приложения: первое – отвечающее за бизнес-логику (взаимодействие с базой данных, управление данными), и второе – за логику приложения и создание (рендеринг) интернет-страниц. Второе приложение логически (но не физически) разделено на интерфейс клиента и интерфейс администратора.

В качестве клиента выступает браузер.

Данная архитектура носит название трехуровневой [19] и является одной из наиболее популярных при построении интернет-систем.

Приведем диаграмму компонентов системы (рис. 11).

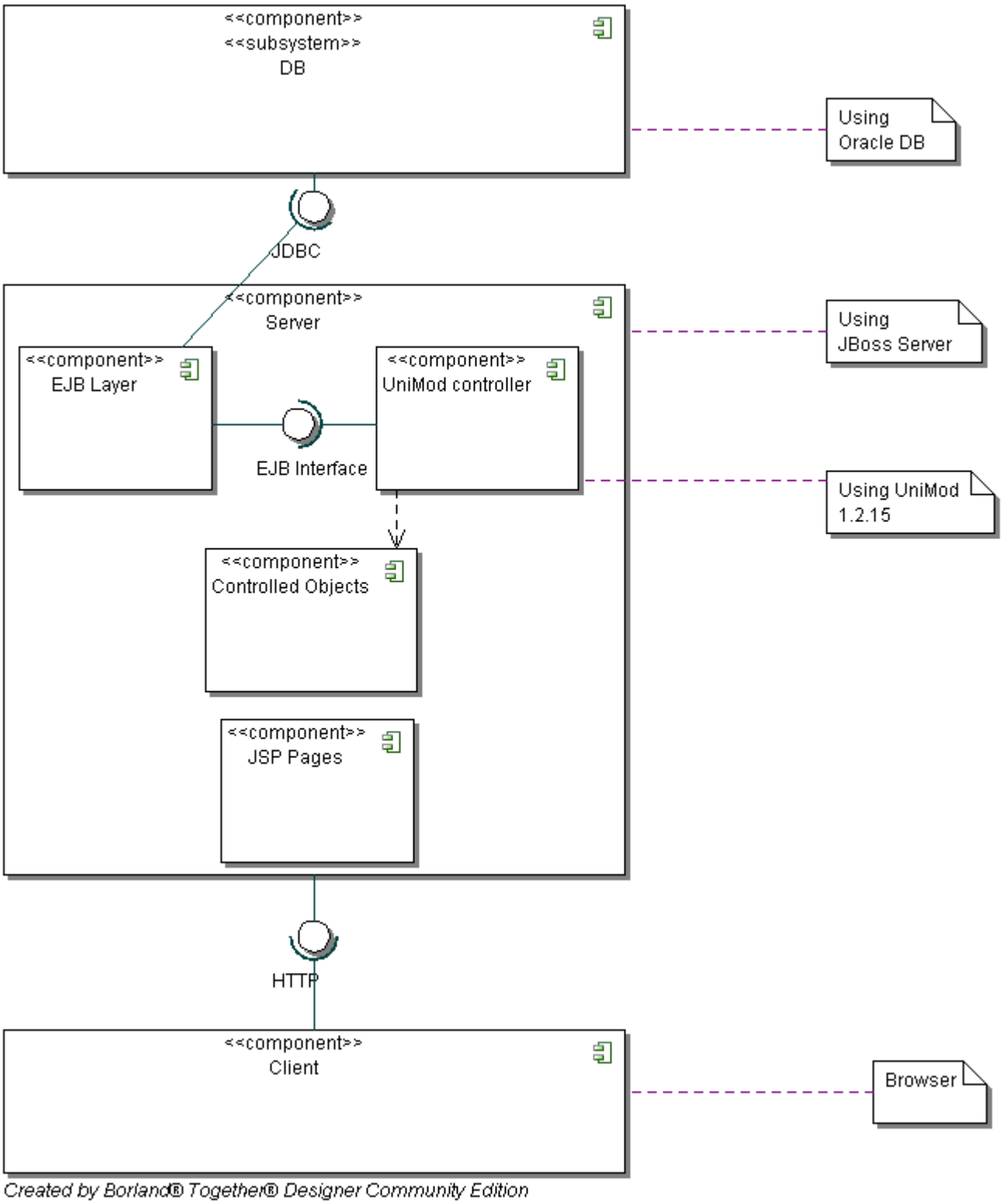


Рис. 11. Диаграмма компонентов

3.6.1. Проектирование базы данных

Построим схему базы данных (рис. 12). Для хранения связи между книгами и заказами воспользуемся вспомогательной таблицей ORDER_PART.

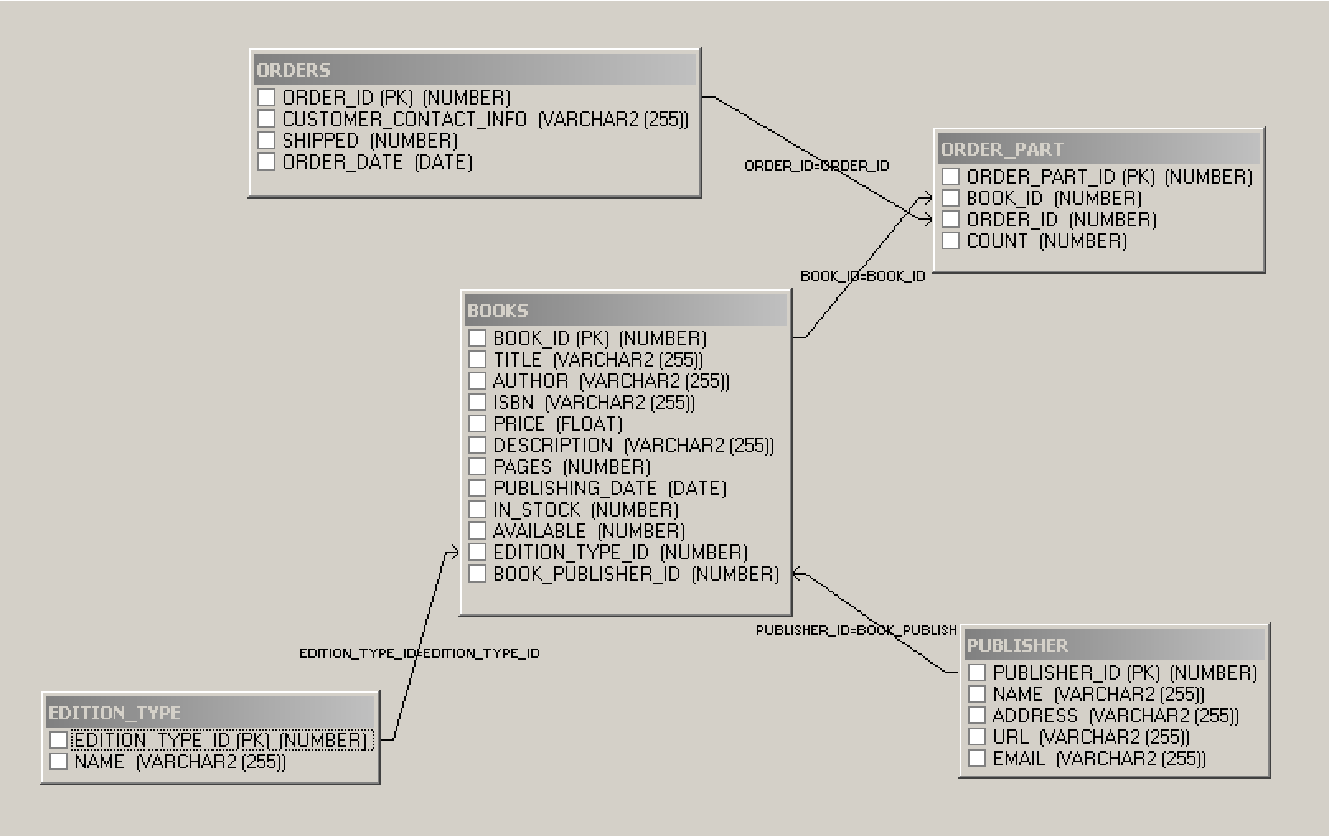


Рис. 12. Схема базы данных

Таким образом, в базе данных присутствует пять таблиц:

1. BOOKS – информация о книгах (изданиях);
2. EDITION_TYPE – типы изданий;
3. PUBLISHER – издательства;
4. ORDERS – заказы;
5. ORDER_PART – экземпляры одного издания, добавленные в заказ.

3.6.2. Проектирование логики реализации

Для проектирования и разработки логики реализации применяется инструментальное средство для поддержки автоматного программирования *UniMod* [20]. В рассматриваемой методике поведение приложения задается с помощью схемы связей и графов переходов конечных автоматов.

Разрабатываемая система разбивается на два логически независимых приложения:

- приложение клиента;
- приложение администратора.

Поставщиками событий для обоих приложений являются:

1) Браузер – сообщает о произошедших событиях пользователя (переходах по ссылке, нажатиях на кнопки, передаче форм);
2) Поставщик информации об ошибках – сообщает о необработанных ошибках (например: разрыв соединения с базой данных и т.д.).

Наборы событий приведены на соответствующих схемах связей и, в основном, соответствуют действиям пользователя.

Объекты управления для этих двух приложений различны.

Начнем с приложения клиента и приведем список его объектов управления:

1) `HomeCO` – утилиты.

   Отвечает за демонстрацию стартовой страницы, вывод сообщения об ошибке или информационного сообщения.

2) `BookCO` – книги.

   Обеспечивает вывод списка книг (с поддержкой фильтрации, сортировки и постраничного вывода).

3) `PublisherCO` – издательства.

   Обеспечивает вывод списка издательств (с поддержкой фильтрации, сортировки и постраничного вывода).

4) `CartCO` – корзина.

   Обеспечивает управление корзиной, хранящейся в сессии пользователя (добавление книг в корзину, отображение информации о корзине) и формирование заказа с передачей его на *EJB*-слой для сохранения в базу данных.

Объекты управления и поставщики событий приведены на схеме связей конечного автомата (Рис. 13). Граф переходов автомата – на Рис. 14.

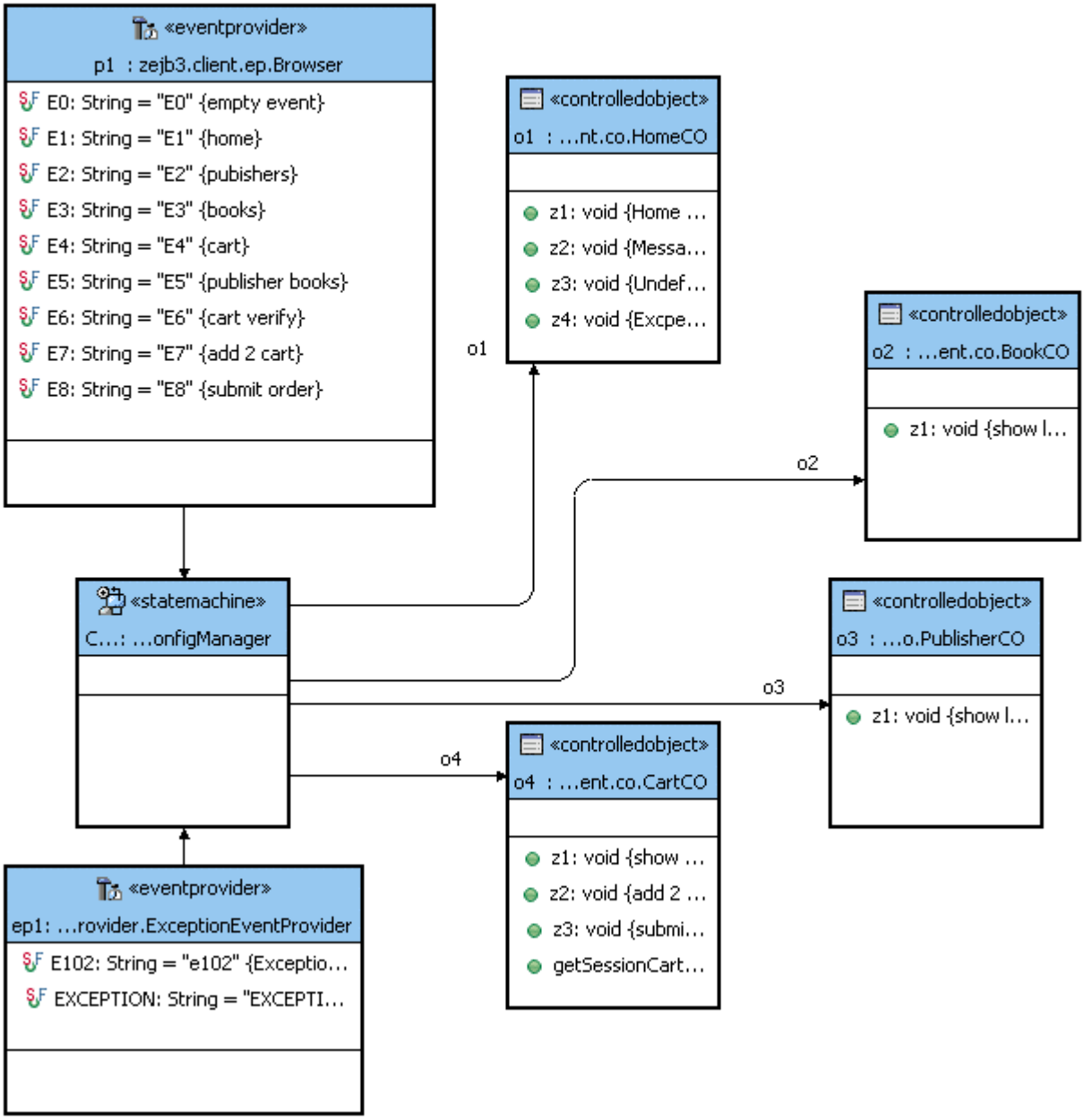


Рис. 13. Приложение клиента. Схема связей

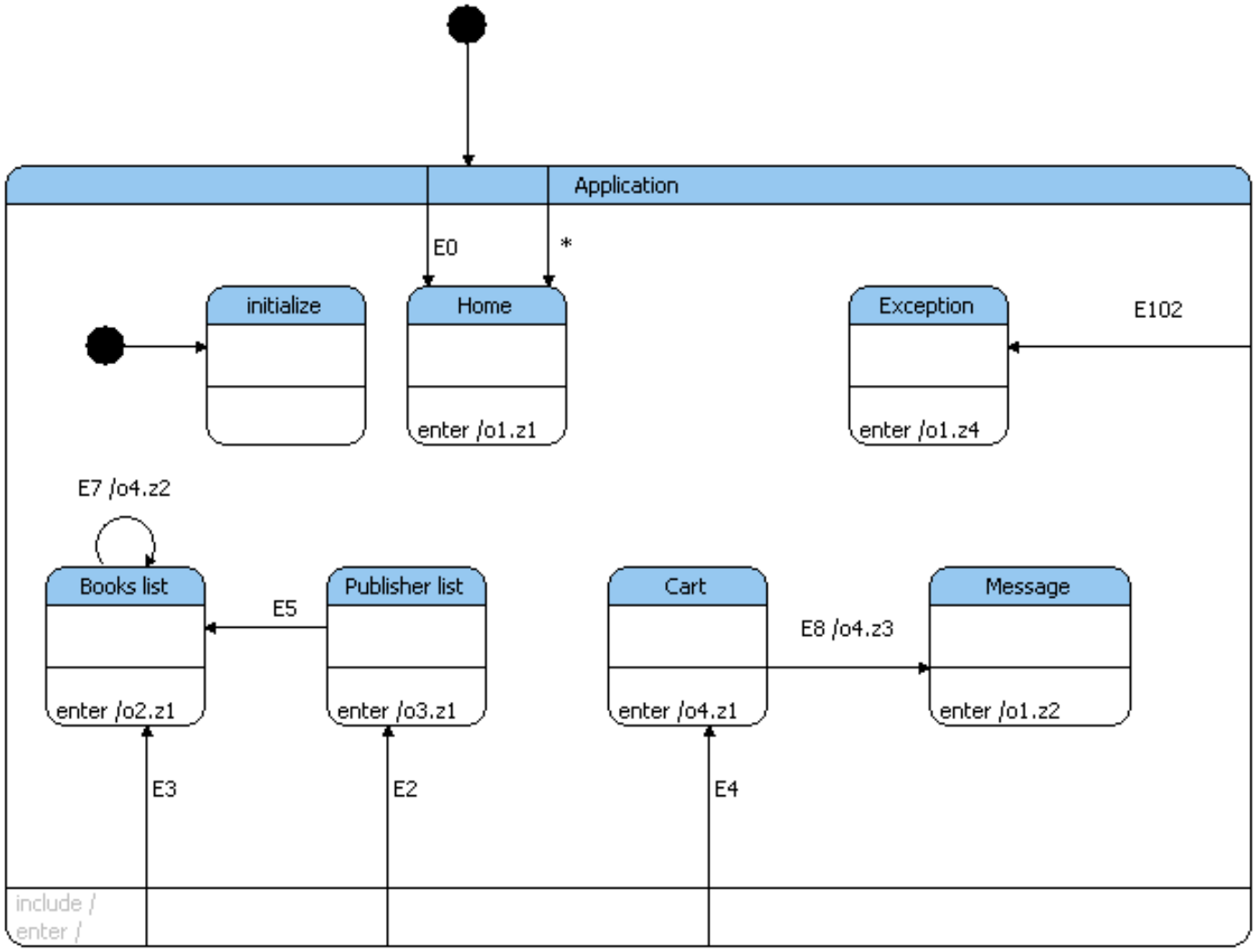


Рис. 14. Приложение клиента. Граф переходов

Приложение администратора использует следующие объекты управления:

- `HomeCO` – утилиты.

  Отвечает за демонстрацию стартовой страницы, вывод сообщения об ошибке или информационного сообщения.

- `BookCO` – книги.

  Обеспечивает вывод списка книг (с поддержкой фильтрации, сортировки и постраничного вывода), создание, удаление и модификацию книг.

- `PublisherCO` – издательства.

  Обеспечивает вывод списка издательств (с поддержкой фильтрации, сортировки и постраничного вывода), создание, удаление и модификацию издательств.

- `OrderCO` – заказы.

Обеспечивает вывод списка заказов (с поддержкой сортировки и постраничного вывода), принятие или отклонение заказов.

Объекты управления и поставщики событий приведены на схеме связей конечного автомата (Рис. 15). Граф переходов приведен на Рис. 16.

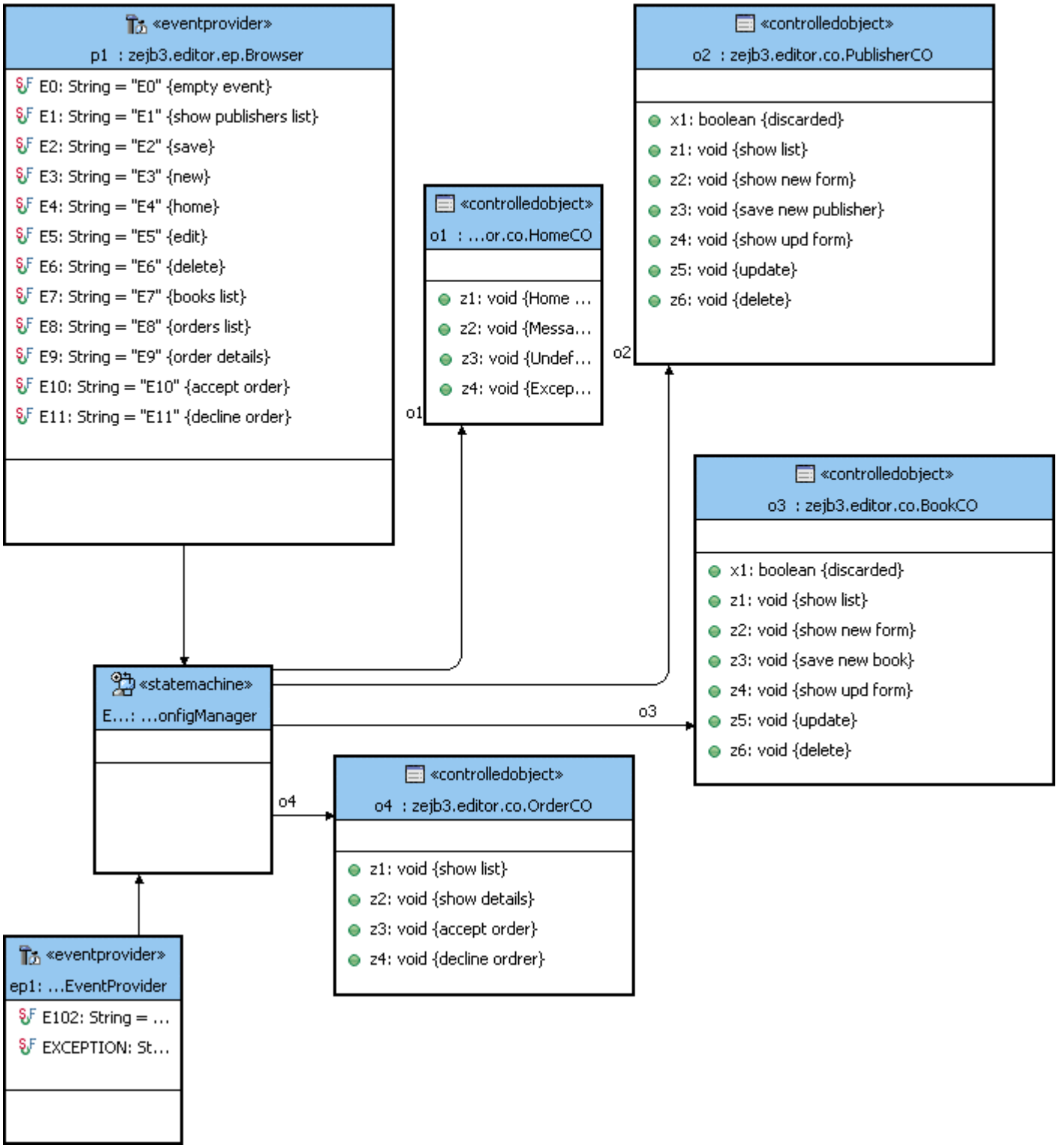


Рис. 15. Приложение администратора. Граф переходов

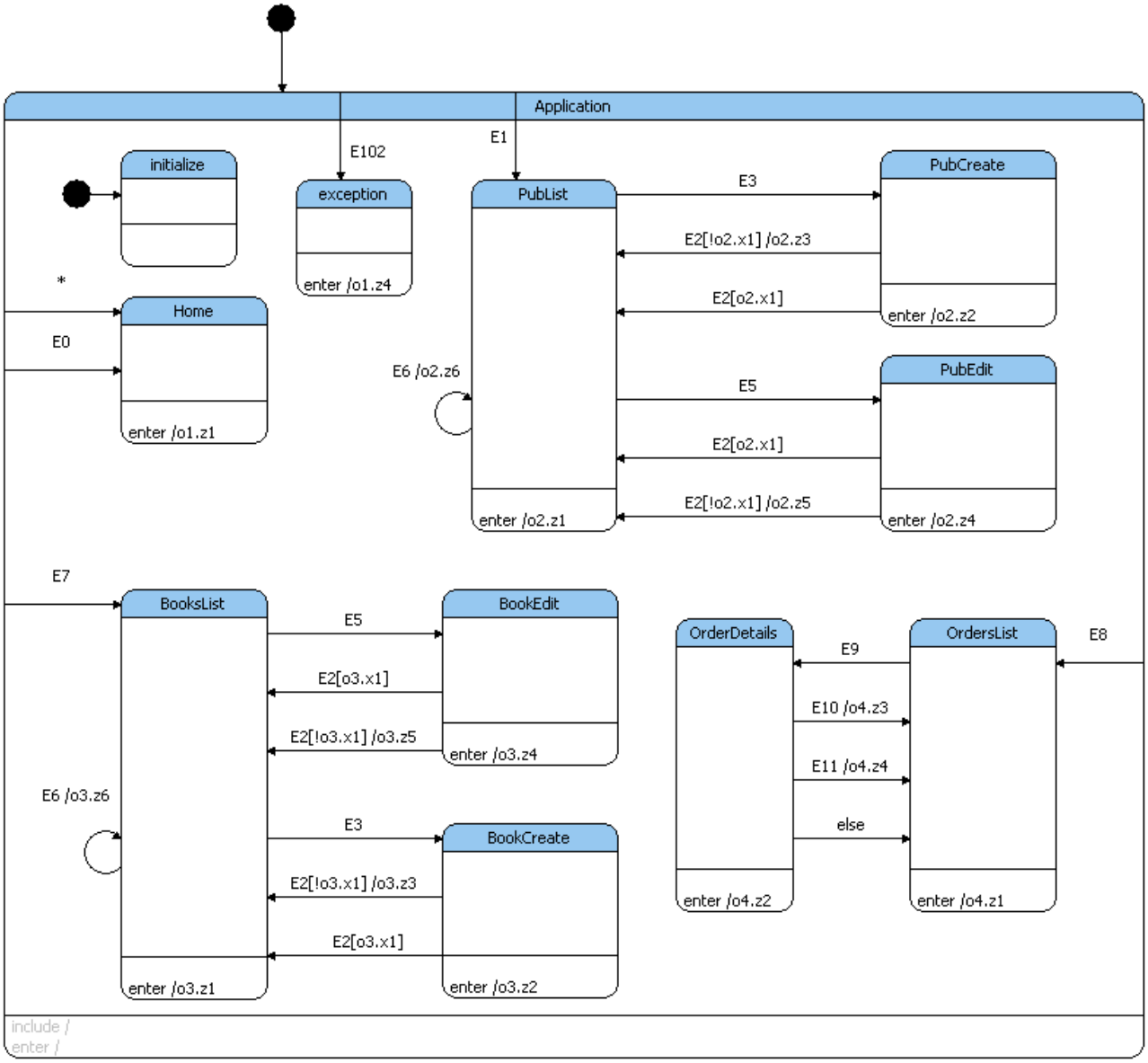


Рис. 16. Приложение администратора. Граф переходов

3.6.3. Проектирование бизнес-логики

Под бизнес-логикой системы понимается связь между сущностями предметной области. Во многих случаях целесообразно использование последних также для реализации бизнес-логики.

Автоматное описание бизнес-логики

Для описания поведения сущности «книга» целесообразно применить конечный автомат. Схема связей приведена на рис. 17, а граф переходов – на рис. 18.

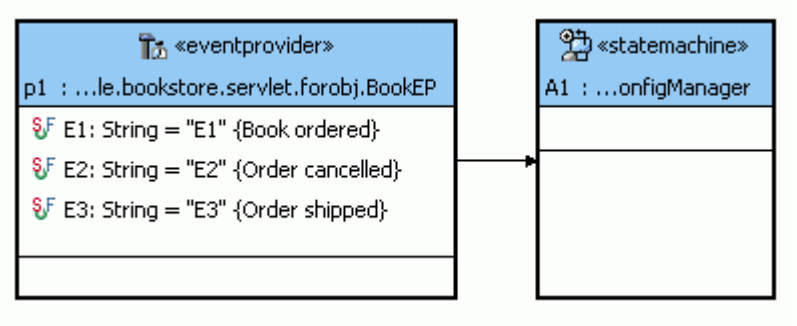


Рис. 17. Сущность «Книга». Схема связей

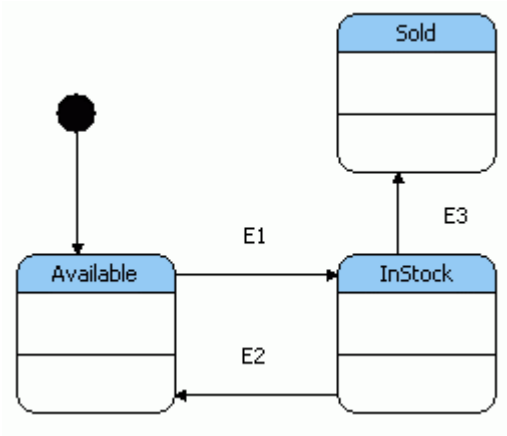


Рис. 18. Сущность «Книга». Граф переходов

Приведенный выше конечный автомат помогает нам понять поведение сущности, но решение хранить по одной записи для каждого экземпляра книги на складе, и управлять каждым экземпляром отдельно было бы ошибочным.

В контексте поставленной задачи правильным решением является объединение группы книг в сущность «Издание» и хранение в базе записи, вид которых приведен в табл. 3.

Таблица 3

Издание	Доступно	Отложено	Прочие поля…
	<количество>	<количество>	…

Об управлении сущностью «Издание» при операциях с заказами будет также сказано ниже.

Автоматная реализация бизнес-логики

В тех же ситуациях, в которых поведение сущности является сложным, и, что важно, операции производятся над конкретной сущностью, а не с их множеством, целесообразно применять автоматный подход для реализации бизнес-логики.

Например, в отличие от книжного склада, в библиотеке осуществляется управление конкретными сущностями, поэтому для описания и реализации упрощенной бизнес-логики управления экземпляром книги в программе управления библиотекой целесообразно применять автоматный подход. Схема связей такого автомата приведена на рис. 19, а граф переходов – на рис. 20.

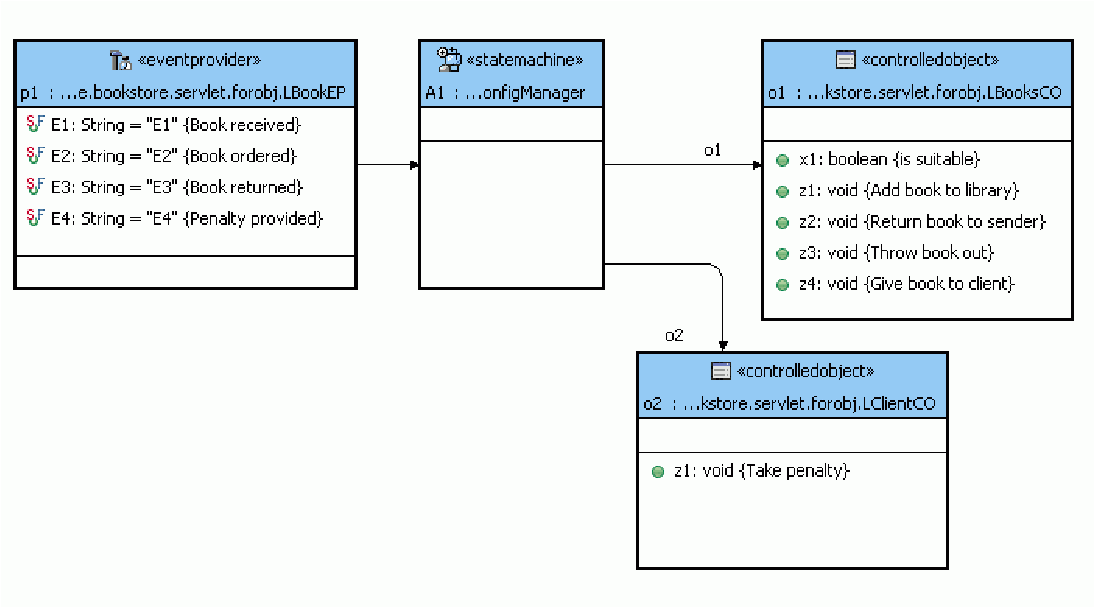


Рис. 19. Сущность «Книга» (для библиотеки). Схема связей

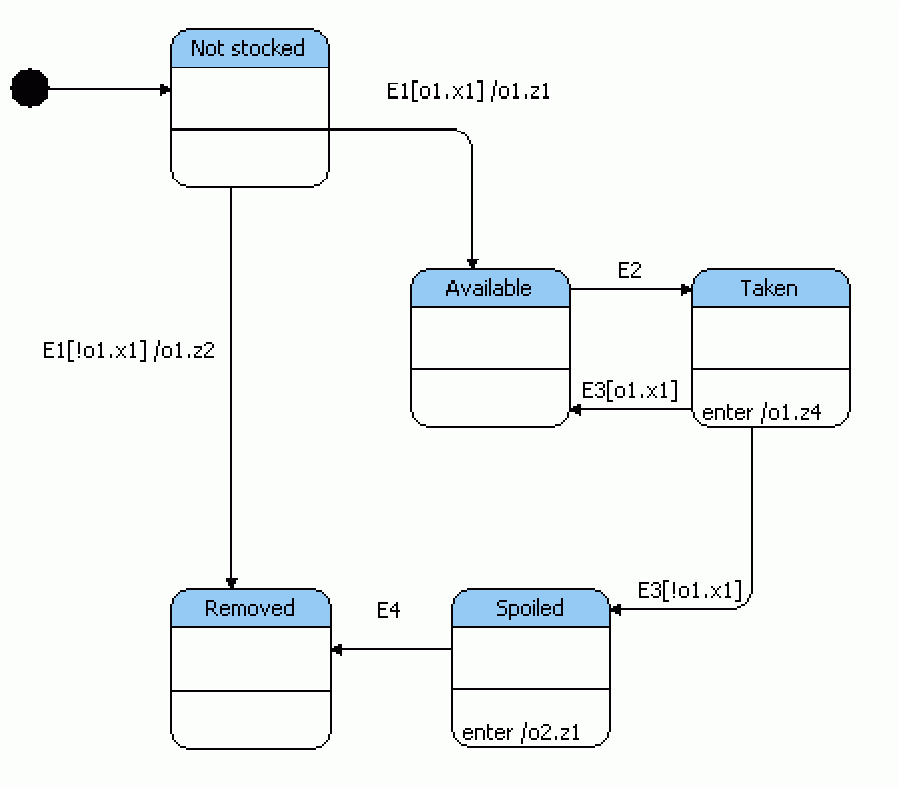


Рис. 20. Сущность «Книга» (для библиотеки). Граф переходов

Из изложенного следует, что если бизнес-логика является автоматной, то она должна быть реализована соотвествующим образом. В рассматриваемом примере книжного склада бизнес-логика не является автоматной, и поэтому она реализована с применением технологии *EJB* без использования автоматов. Перейдем к описанию ее реализации.

Сущности системы

Бизнес-уровень системы оперирует следующими видами сущностей.

- *Book* – издание. Экземпляры издания могут быть доступными для продажи (*available*) и отложенными по заказу на склад (*in stock*).
- *Publisher* – издательство. При добавлении издания в систему необходимо указать его издательство.
- *Edition Type* – тип издания. Один из трех – книга в бумажной обложке, книга в твердом переплете и книга на аудио диске.
- *Order* – оформленный заказ клиента системы. Содержит контактную информацию и список заказанных изданий в виде сущностей *Order Part*.
- *Order Part* – набор заказанных экземпляров одного издания, часть заказа.

Все сущности обладают уникальными числовыми идентификаторами.

Чтение сущностей

Чтение сущностей из базы данных реализовано классами, наследующими интерфейс `RecordReader`. Основной метод `read` из этого интерфейса читает одну запись из полученного от базы `ResultSet` и возвращает соответствующий объект *DTO* (*Data Transfer Object*), заполнив его прочитанными данными (рис. 21).

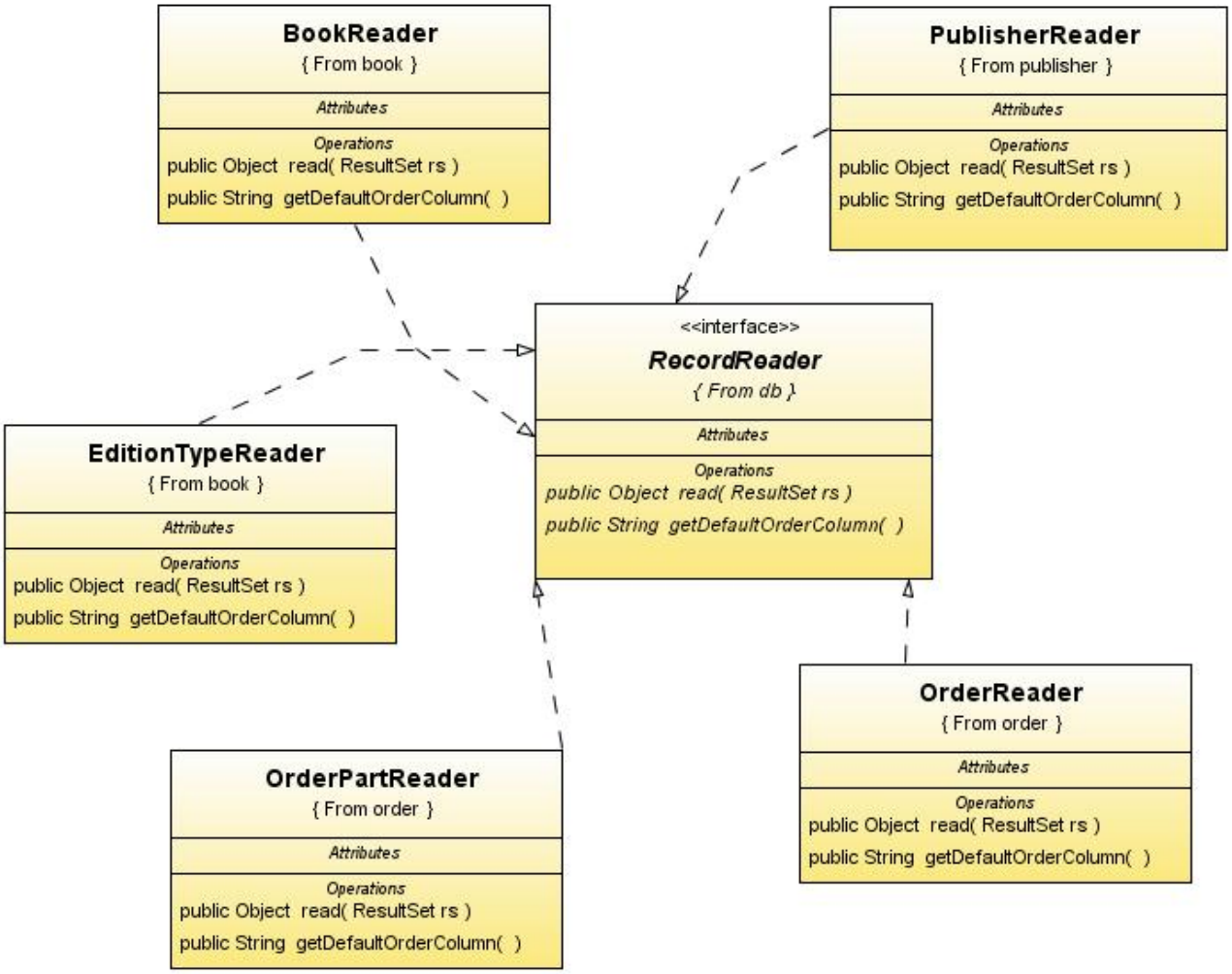


Рис. 21. Классы для чтения сущностей

Запись сущностей

Запись данных сущностей в *SQL*-запросы `INSERT\UPDATE\DELETE` производят объекты классов, реализующих интерфейс `RecordWriter` (рис. 22).

Методы интерфейса `RecordWriter`:

- `Write` – запись значений полей объекта в строку, для операций *`INSERT`* или `UPDATE`, в зависимости от второго параметра метода;
- `GetFields` – запись имен полей объекта в строку, для операций `INSERT` или `UPDATE`, в зависимости от второго параметра метода;

- `GetId` – получение идентификатора (для всех видов сущностей идентификаторами являются поля типа `number\int`);
- `GetIdColumn` – возвращает имя поля идентификатора в базе.

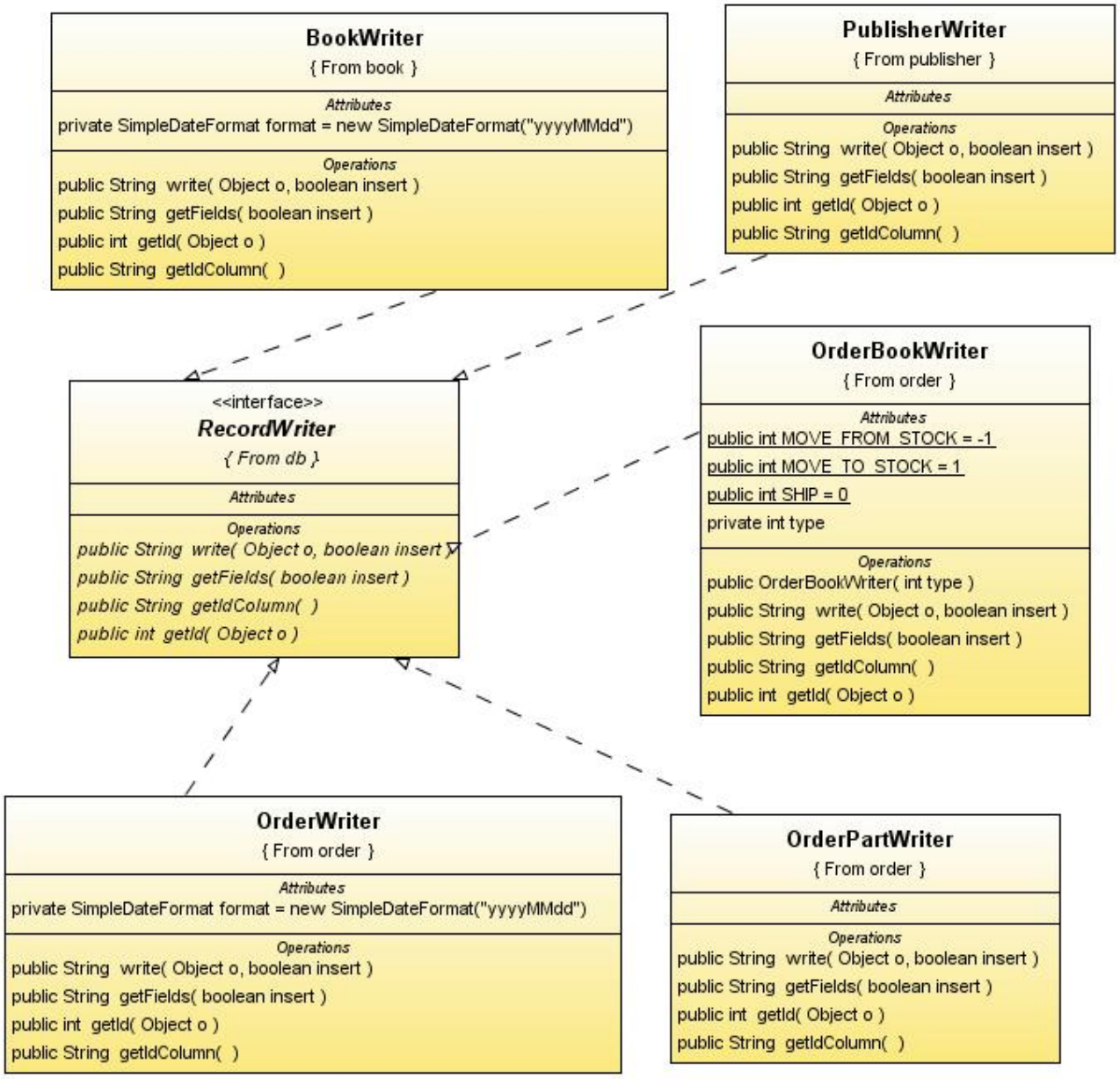


Рис. 22. Классы для записи сущностей

Бизнес-компоненты

Операции бизнес уровня реализуются компонентами *EJB* и разделены по типам оперируемых сущностей (рис. 23).

Общий базовый класс `BaseEntityBean` реализует общие методы добавления, модификации, удаления, получения объекта и метод получения списка объектов с сортировкой, фильтрацией и раз-

биением по страницам. Конкретные бизнес компоненты используют эти методы, передавая в них имя таблицы и объекты для чтения или записи соответствующей сущности.

Кроме доступа к базе данных, бизнес компоненты реализуют бизнес-логику системы. А именно:

- При оформлении заказа соответствующее число экземпляров выбранных изданий переносится из доступных экземпляров (`available`) на склад (`in stock`);
- При отмене заказа экземпляры издания возвращаются в свободное состояние `available;`
- При доставке заказов экземпляры издания удаляются из системы.

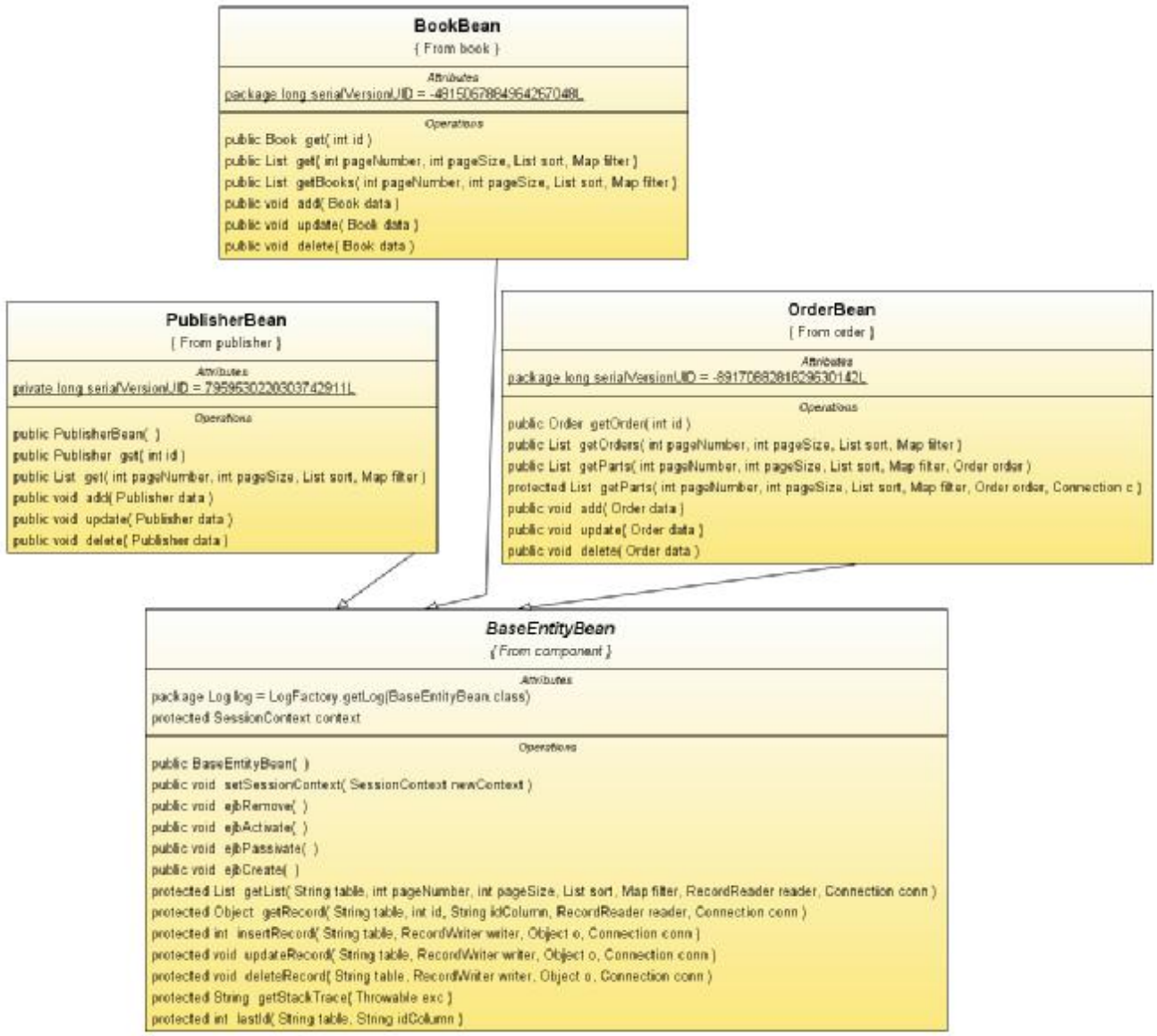


Рис. 23. Бизнес-компоненты

3.7. Развертывание приложения

Разработанное приложение развертывается на двух логических серверах (желательно установленных на различных физических серверах):

1. Сервер баз данных *Oracle 8i* (использовалась версия 8.1.7).
2. Сервер приложений *JBoss 4* (использовалась версия 4.0.2).

На сервер баз данных устанавливается схема базы данных.

На сервер приложений устанавливаются два приложения:

- приложение, отвечающее за бизнес-логику, и реализованное в виде *EJB* (*EJB Application* [16]);

- приложение, отвечающее за и отображение страниц и логику переходов, и реализованное в виде Сервлета (*Servlet Application* [17]).

Приведем диаграмму развертывания системы (Рис. 24).

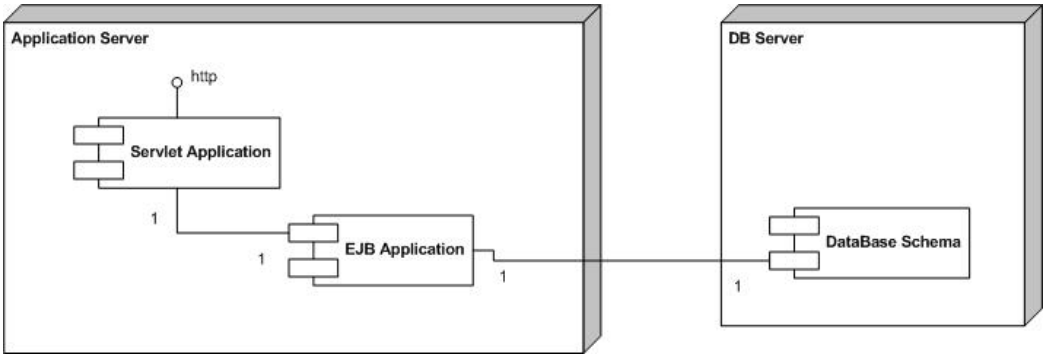


Рис. 24. Диаграмма развертывания

3.8. Тестирование производительности

Далее, необходимо исследовать производительность разработанной системы. Для исследования производительности применялся пакет *Jakarta JMeter* версии 2.1.1 [21]. Тестированию подвергался интерфейс клиента (из технического задания следует, что приложение администратора не подвергается пиковым нагрузкам)

Краткое описание конфигурации тестируемой системы:

- сервер баз данных – 2-х процессорный *Pentium III – 500 Mhz, 1GB RAM;*
- сервер приложений –2-х процессорный *Pentium III – 750 Mhz, 1GB RAM;*
- сервер *Jmeter* – однопроцессорный *Pentium IV – 2,04 Ghz, 1GB RAM*.

Тестирование не являлось чистым. На всех серверах одновременно были запущены другие приложения, обрабатывающие запросы и выполняющие другую деятельность.

3.8.1. Общее описание процесса тестирования

Скрипт *JMeter* выбирает один из 10 предложенных сценариев поведения пользователя случайным образом. Между каждыми двумя вызовами происходит задержка по таймеру. Время задержки задается формулой $a+Gaussian(b)$, где a – значение постоянной составляющей задержки в секундах, а b – среднее значение переменной составляющей, имеющей распределение Гаусса. Таким образом, имитируется поведение реального пользователя (который не осуществляет мгновенных переходов между страницами и паузы между переходами у которого имеют различное значение). Под периодом подключения потоков понимается отрезок времени, в течении которого все потоки данного типа будут запущены. При тестировании данного проекта использовались потоки, эмулирующие пользователей двух типов (табл. 4).

Таблица 4

Потоки	Период подключения	Задержка	Цикл
С высокой активностью (Active)	60	3+Gaussian(3)	До ручной остановки
С низкой активностью (Delayed)	60	6+Gaussian(3)	До ручной остановки

Рассмотрим результаты тестирования производительности (зависимость времени отклика от количества одновременно запущенных потоков) (рис. 25).

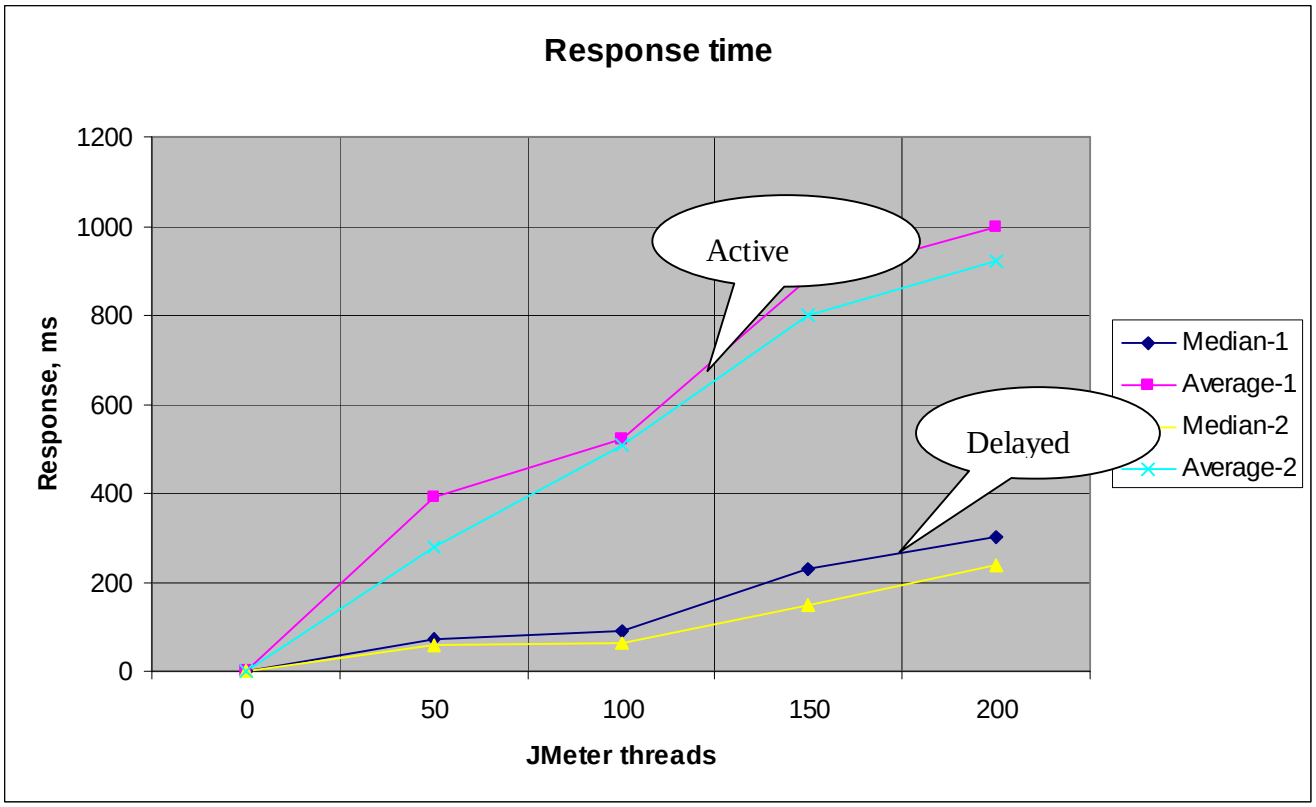


Рис. 25. Тестирование производительности

Таким образом, система в данной конфигурации показывает приемлемое время отклика (одна секунда) вплоть до 200 условно-одновременных потоков.

Время отклика в пределах одной секунды означает время ожидания отклика пользователем – порядка двух секунд в зависимости от настроек сети, маршрута и быстродействия компьютера пользователя. Также обратим внимание на то, что тестирование проводилось в ситуации, когда все компьютеры включены в единую локальную (*100 Mbit/s*) сеть.

3.8.2. Тестирование нагрузки на сеть

Приведем график зависимости нагрузки на сеть (объема передаваемых данных) от количества потоков (рис. 26).

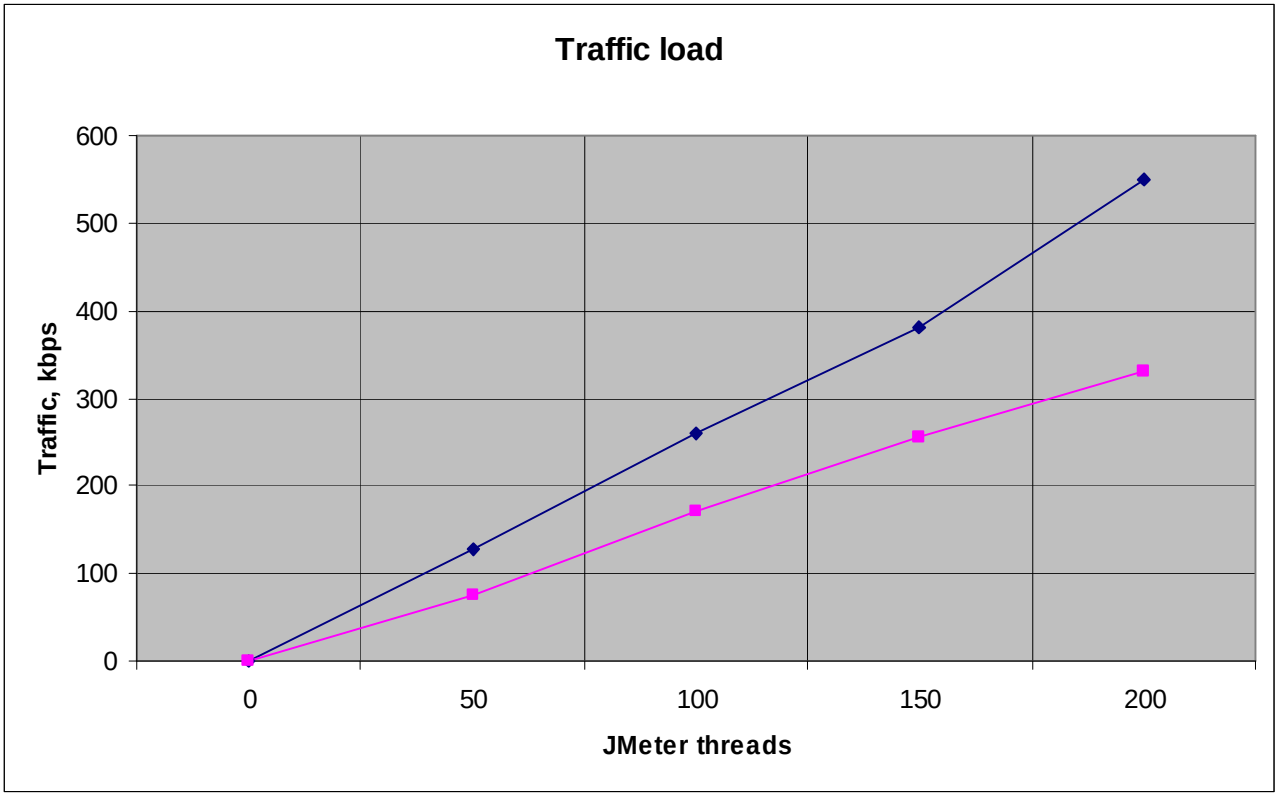


Рис. 26. Тестирование нагрузки на сеть

Средний объем страницы составляет порядка 12 килобайт. При тестировании:

- исследовался суммарный трафик, зафиксированный *Jakarta Jmeter* отнесенный к времени выполнения сценарии;
- не моделировалась одновременная многоточечная загрузка кэшируемого содержимого (*css, png, js*). Последний вопрос относится к области конфигурации *JBoss Application Server* [22, 23] и сетей, в которых расположены сервер и клиент.

3.8.3. Эффект стабилизации времени отклика системы

Рассмотрим работу системы при предельной нагрузке (500 потоков). На рис. 27 приведен график зависимости медианы значений времени отклика от времени тестирования.

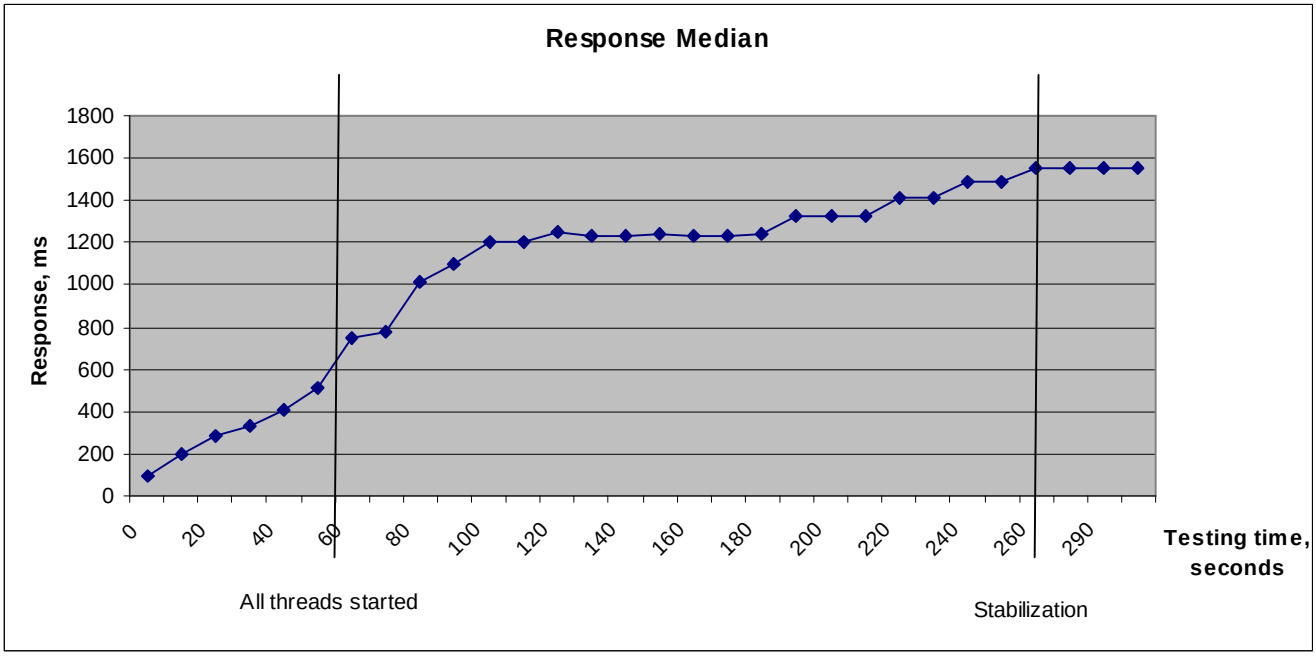


Рис. 27. Эффект стабилизации времени отклика

На графике наблюдается первая область стабилизации в районе 100–260 секунд с начала тестирования. Вторая область стабилизации начинается в районе 280 с. Области стабилизации времени отклика связаны с ограничением физической памяти доступной процессам серверов баз данных и сервера приложений.

3.9. Создание инструкции по использованию

В инструкции по использованию необходимо привести:

- общее описание функциональности, ориентированное на пользователя системы
- список экранов (страниц) системы с изображениями этих экранов и комментариями к возможным действиям.

Замечание. Следующий далее текст является фрагментом реальной документации к конкретной системе, фактически переданной заказчику. Способ оформления документации определялся заказчиком. В частности, в соответствии с пожеланиями заказчика иллюстрации в документации не нумеруются и не подписываются.

3.9.1. Общее описание функциональности

Система предназначена для обработки заказов для книжного склада, на котором хранятся книги различных издательств. Предоставлены два рабочих интерфейса системы.

Интерфейс администратора позволяет:

- просматривать и модифицировать информацию о книгах на складе;
- просматривать информацию о заказах, принимать или отклонять заказы.

Интерфейс пользователя позволяет:

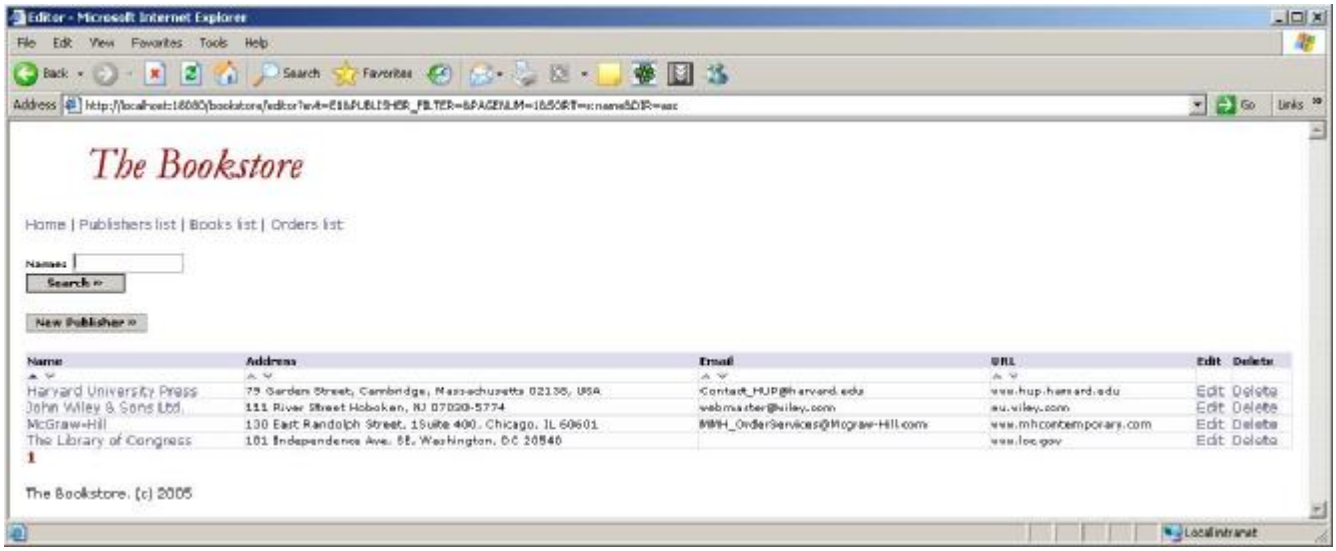
- просматривать информацию об имеющихся книгах;
- производить заказы книг.

Оба интерфейса включают в себя меню, расположенное всегда вверху окна.

3.9.2. Страницы интернет-приложения

Приложение администратора

Список издательств

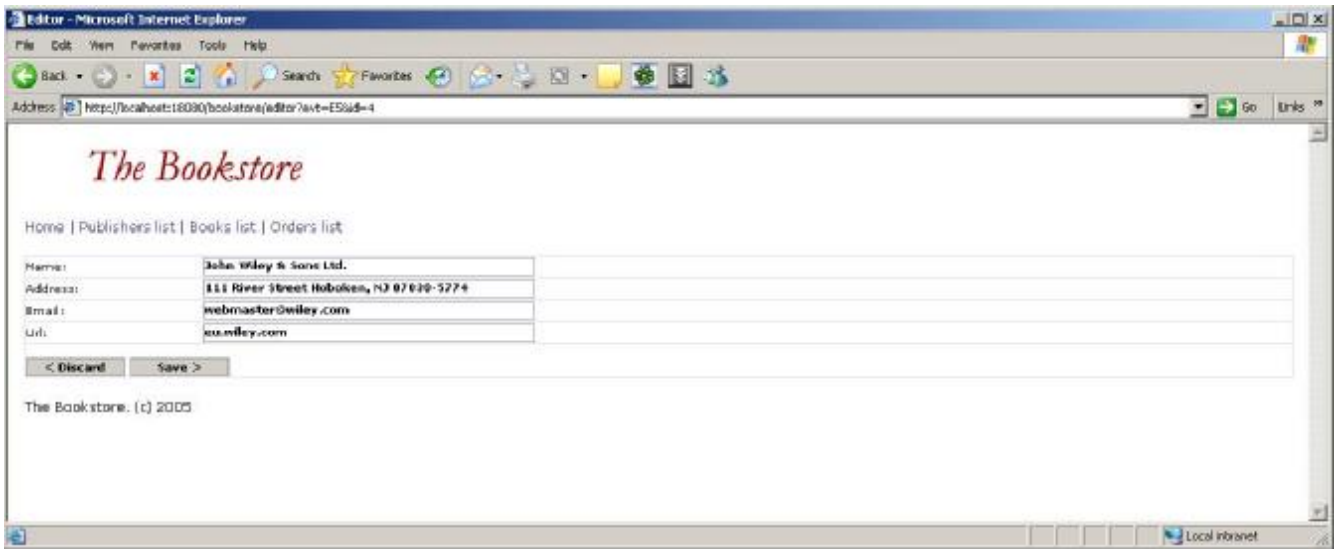


Функциональность:

Edit	Переход на страницу модификации издательства
Delete	Удаление издательства из системы (с удалением всех изданных у него книг)
New Publisher	Переход на страницу добавления издательства
Search	Обновление страницы с поиском по введенному полю Name (имя издательства)
Стрелки	Сортировка списка издательств
Счетчик страниц	Переход по страницам списка (8 записей на странице)
Имя издательства	Переход на страницу изданий с фильтрацией по издательству

Добавление и модификация издательства

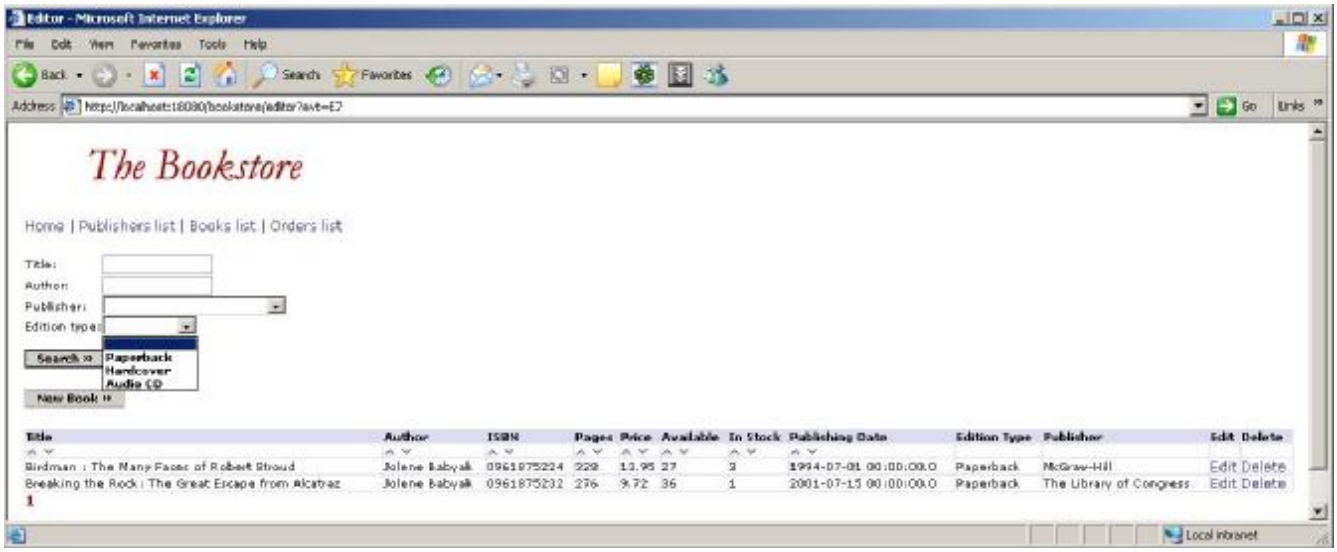
Переход на данную страницу происходит со списка издательств либо по ссылке *Edit*, либо по кнопке *New Publisher*. В первом случае данная страница позволяет редактировать издательство, во втором – добавить новое.



Функциональность:

Discard\Cancel	Возвращение на страницу списка издательств
Save	Сохранение изменений или добавление нового издательства

Список изданий

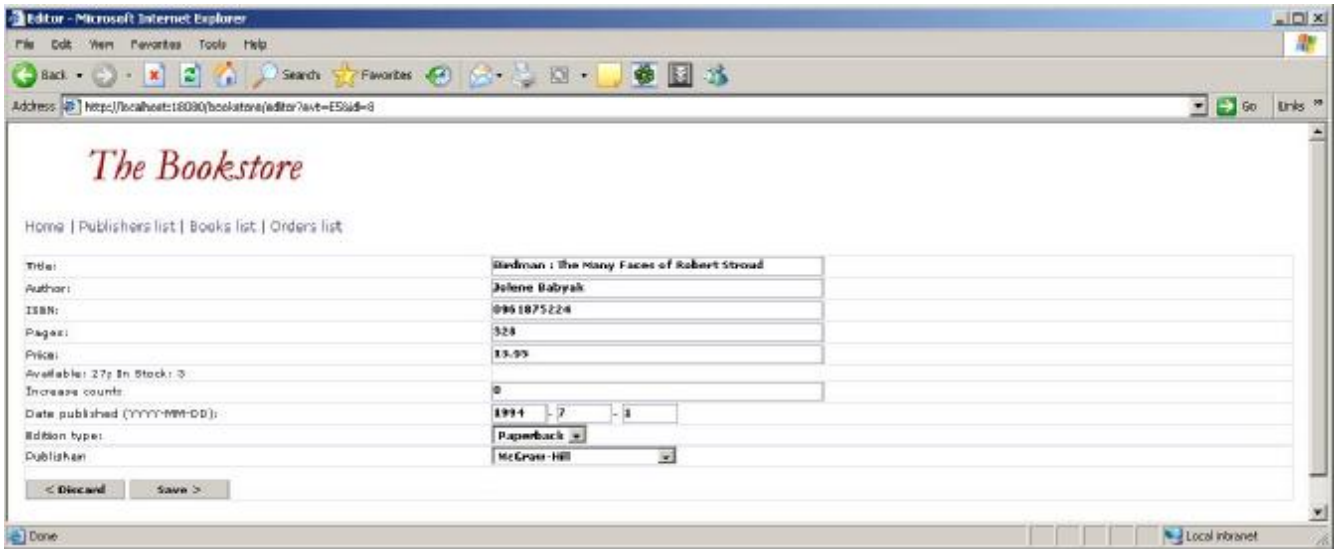


Функциональность:

Edit	Переход на страницу модификации издания
Delete	Удаление издания из системы
New Book	Переход на страницу добавления издания
Search	Обновление страницы с фильтрацией списка по введенным в форме полям
Стрелки	Сортировка списка изданий
Счетчик страниц	Переход по страницам списка (8 записей на странице)

Добавление и модификация изданий

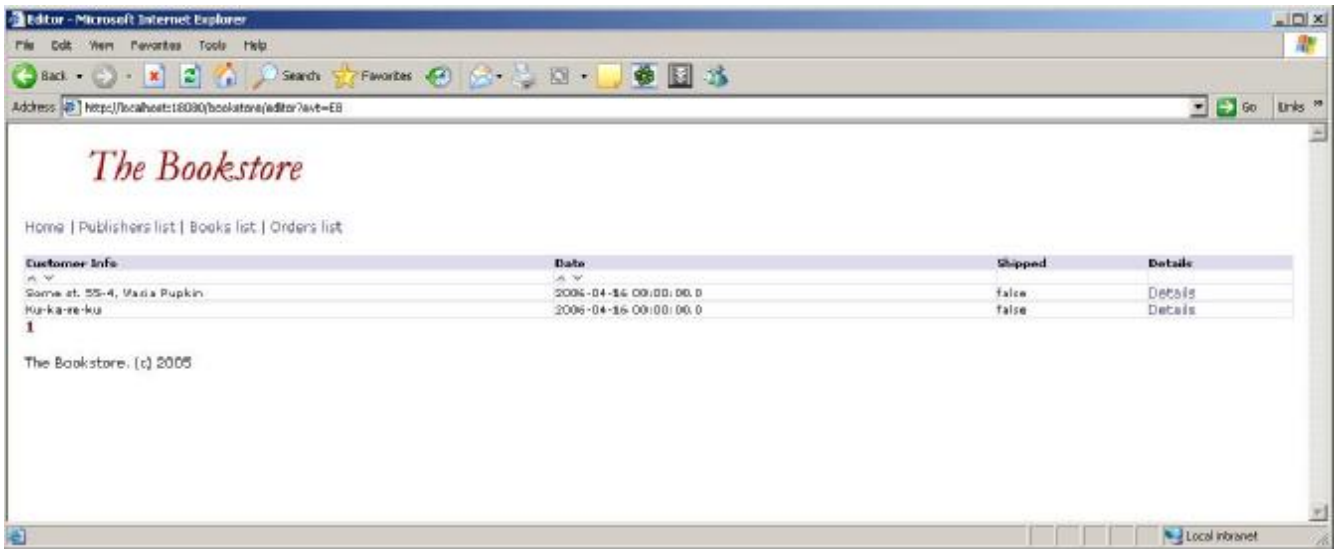
Переход на данную страницу происходит со списка изданий либо по ссылке *Edit*, либо по кнопке *New Book*. В первом случае данная страница позволяет редактировать существующую запись об издании, во втором – добавить информацию о новом издании.



Функциональность:

Discard\Cancel	Возвращение на страницу списка изданий
Save	Сохранение изменений или добавление нового издания

Список заказов

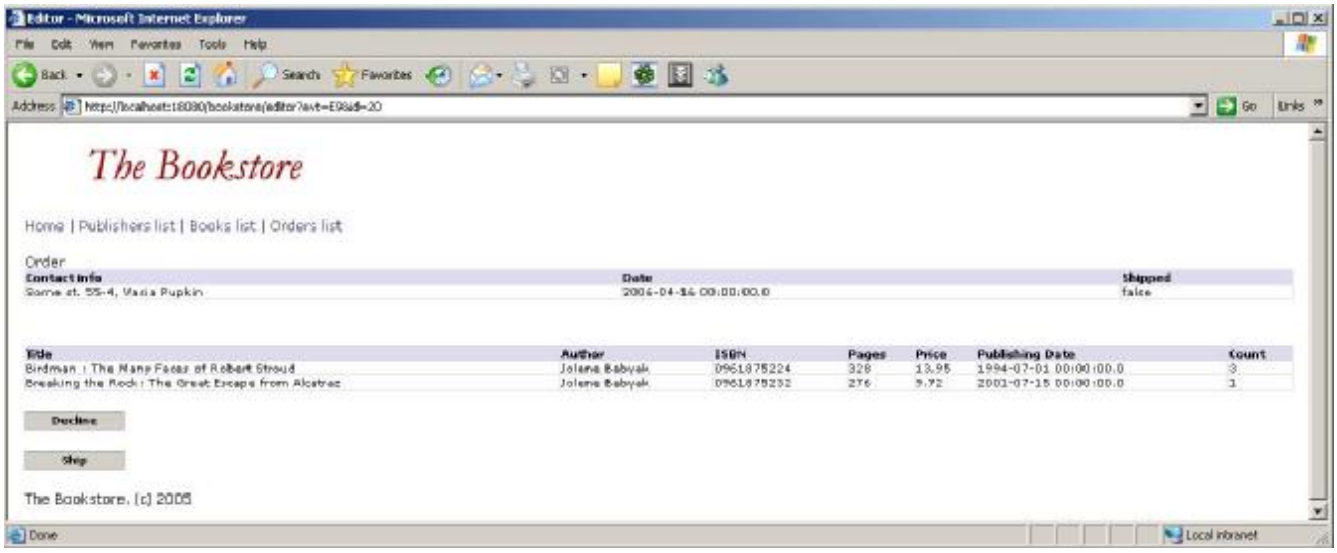


Функциональность:

Details	Переход на страницу манипуляции заказом
Стрелки	Сортировка списка заказов
Счетчик страниц	Переход по страницам списка (8 записей на странице)

Подтверждение и отклонение заказов

Переход на данную страницу происходит со списка заказов по ссылке *Details*.



Функциональность:

Decline	Отклонить заказ
Ship	Пометить заказ как доставленный

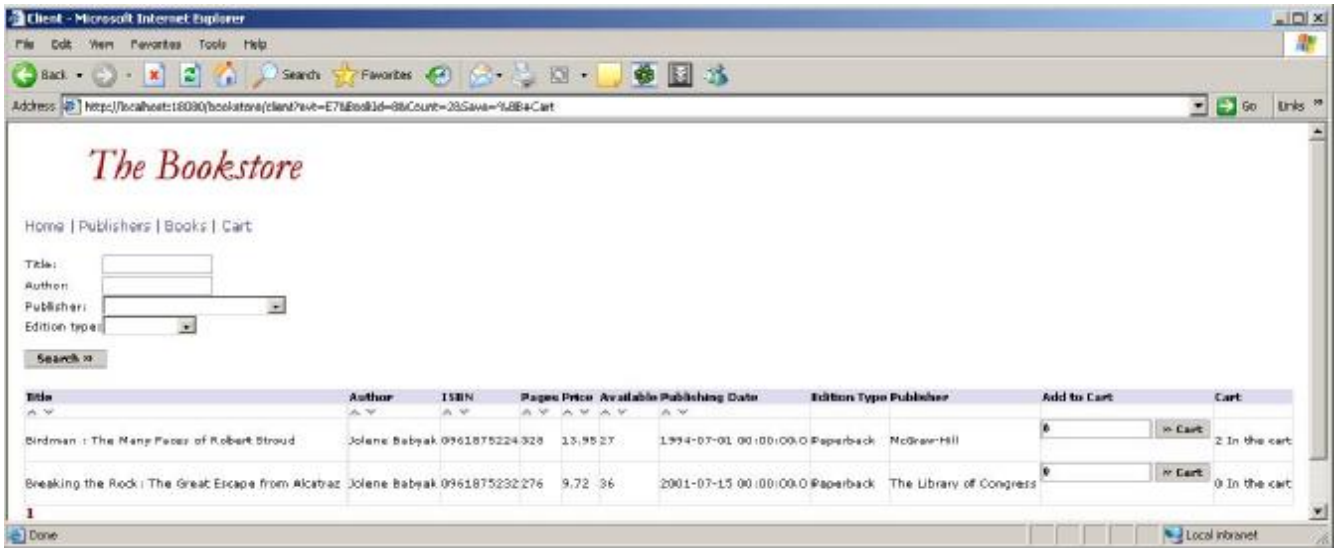
Приложение пользователя

Список издательств

Функциональность для пользовательского интерфейса совпадает с функциональностью интерфейса администратора, за исключением возможности модифицировать данные системы. Клиенту недоступны пункты *New Publisher, Edit* и *Delete*.

Список изданий

Клиенту предлагается отобрать интересующие его издания в специальную корзину *Cart*, после чего он может оформить заказ на отобранные издания.

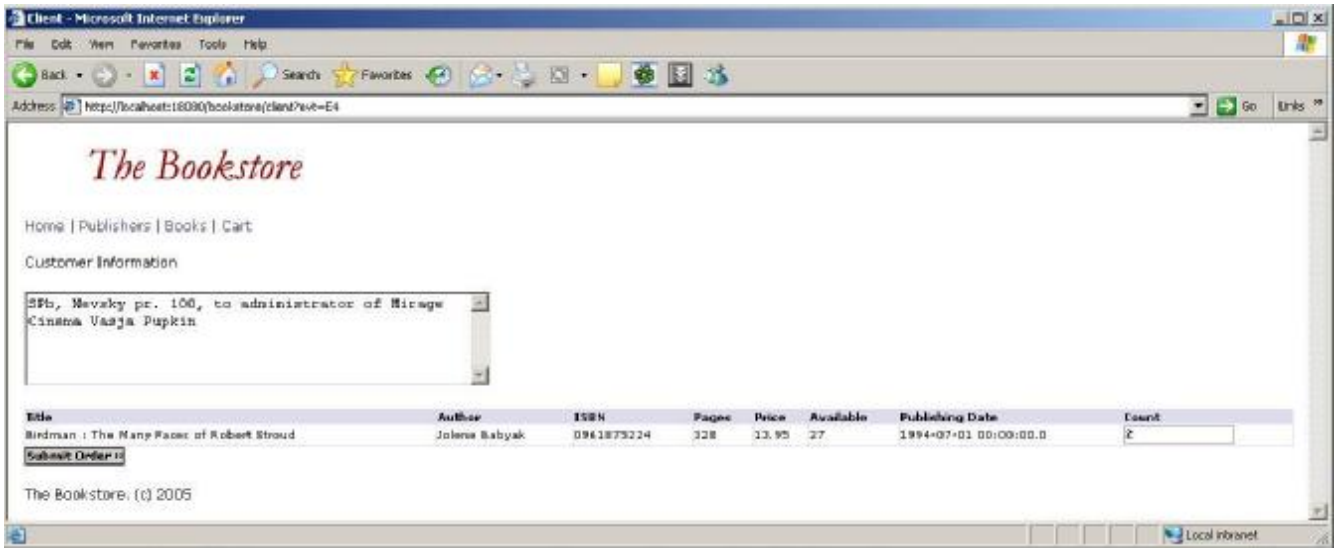


Функциональность:

Cart	Положить в корзину введенное в поле слева количество экземпляров данного издания.
остальное	Возможность управления списком совпадает с функциональностью, описанной в разделе об интерфейсе администратора – поиск с фильтрацией, постраничная выборка, сортировка списков.

Оформление заказа

В данном окне клиент может просматривать содержимое своей корзины, модифицировать количество требуемых экземпляров выбранных изданий. После определения контактной информации, достаточной для успешной доставки заказа, клиент должен воспользоваться кнопкой *Submit Order* для оформления заказа.



Функциональность:

Submit Order	Оформить заказ.

4. ЗАКЛЮЧЕНИЕ

Предложенная методика состоит из десяти этапов, выполнение которых позволяет спроектировать разработать, протестировать и поставить готовую Интернет-систему. Отметим, что проектирование и разработку логики реализации в данной методике предлагается выполнять, руководствуясь принципами автоматного программирования, что обеспечивает высокие показатели качества разработки этого компонента системы.

5. ЛИТЕРАТУРА

1. *Conallen J.* Building Web Applications with UML. Addison-Wesley, 2000.
2. *Frankel D.* Model Driven Architecture. Applying MDA to Enterprise Computing. Indianapolis: Web Publishing Inc., 2003.
3. *Буч Г., Рамбо Дж., Якобсон И.* UML. СПб.: Питер, 2005.
4. *Фаулер М.* UML. Основы. СПб.: Символ-Плюс, 2005.
5. *Cohn M.* User Stories Applied For Agile Software Development. Addison-Wesley, 2004.
6. *Jacobson I.* Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley, 1994.
7. *Philips R., Mohseni P.* Professional Java Server Programming. NY: Springer-Verlag, 1999.
8. *McCullough-Dieter C.* Oracle 8 Bible. Hungry Minds, 1998.
9. *Шалыто А.А.* SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука. 1998. http://is.ifmo.ru/books/switch/1
10. *Шалыто А.А., Туккель Н.И.* Танки и автоматы // BYTE/Россия. 2003. № 2, с. 69–73. http://is.ifmo.ru/works/tanks_new/
11. *Alur D., Crupi J., Malks D.* Core J2EE Patterns: Best Practices and Design Strategies. Palo Alto: Prentice Hall. 2003.
12. *Kaner C., Falk J., Nguyen H.Q.* Testing Computer Software. Wiley, 1999.
13. *Schulmeyer G.G., Mcmanus J.I.* The Handbook of Software Quality Assurance. Prentice Hall, 1999.
14. *Grimm T.* Rapid Prototyping. Society of Manufacturing Engineers, 2004.
15. *Ноутон П., Шилдт Г.* Java 2. СПб.: БХВ-Петербург, 2001.
16. *Roman E.* Mastering Enterprise JavaBeans and the Java 2 Platform. John Wiley & Sons, 1999.
17. *Hunter J.* Java Servlet Programming. O'Reilly, 1998.
18. *Сью Ш.* JSTL: практическое руководство для JSP-программистов. М.: Кудиц-Образ, 2004.
19. *Sessions R.* Software Fortresses: Modeling Enterprise Architectures. Addison-Wesley, 2003.
20. *Гуров В.С., Мазин М.А.* Веб-сайт проекта *UniMod*. http://unimod.sourceforge.net/
21. *Pekowsky L.* Apache Jakarta and Beyond: A Java Programmer's Introduction. Addison-Wesley, 2005.
22. *Marrs T., Davis S.* JBoss at Work: A Practical Guide. O'Reilly, 2005.
23. *Richards N., Griffith S.* JBoss: A Developer's Notebook. O'Reilly, 2005.

Лист регистрации изменений

Номера листов (страниц)					*Всего листов (страниц) в докум.*	*№ документа*	*Входящий № сопроводительного докум. и дата*	*Подп.*	*Дата*
Изм.	*Измененных*	*Замененных*	*Новых*	*Аннулированных*					