Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

Высшего профессионального образования

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

Кафедра информационных систем и программной

инженерии

**Второй этап курсовой работы**

по дисциплине «Распределенные программные системы»

«Структура бизнес-логики и веб-интерфейса системы.»

**Выполнил:**

Ст. гр. ИСТ-113

Алексеев С.О.

**Принял:**

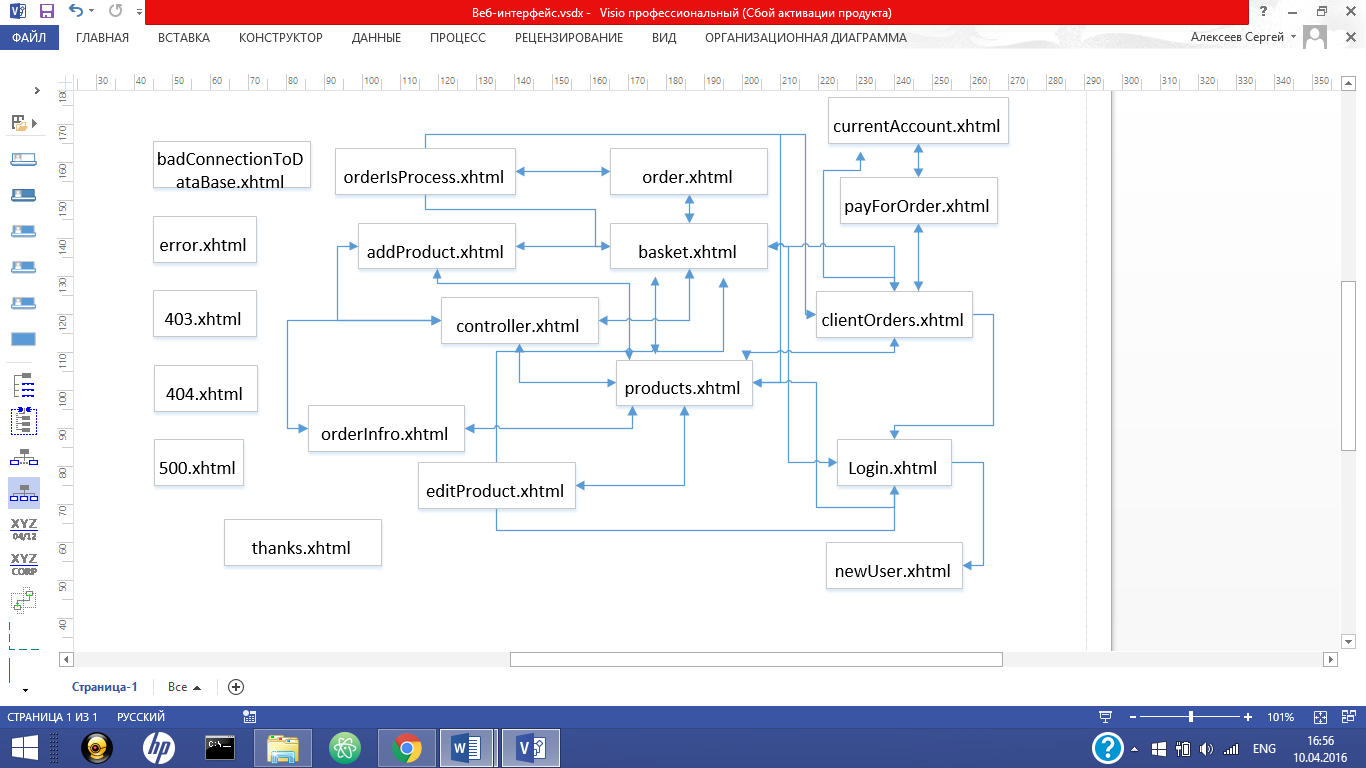
Тимофеев А. А.

Владимир 2016

Схема веб интерфейса системы

Современные концепции проектирования распределенных информационных систем корпоративного уровня предполагают общение пользователя системы с базой данных посредством браузера и технологий Internet. Для обеспечения возможности взаимодействия пользователя с браузером применяется Веб-интерфейс — это совокупность средств, при помощи которых пользователь взаимодействует с веб-сайтом или любым другим приложением через браузер.

Веб-интерфейс моего курсового проекта «Магазин компьютерной техники» приведен на рисунке 1.

Рисунок1. Схема веб-интерфейса системы

**Описание веб-страниц.**

1. JSF-страница thanks.xhtml - в качестве благодарности за покупку создана данная страница.

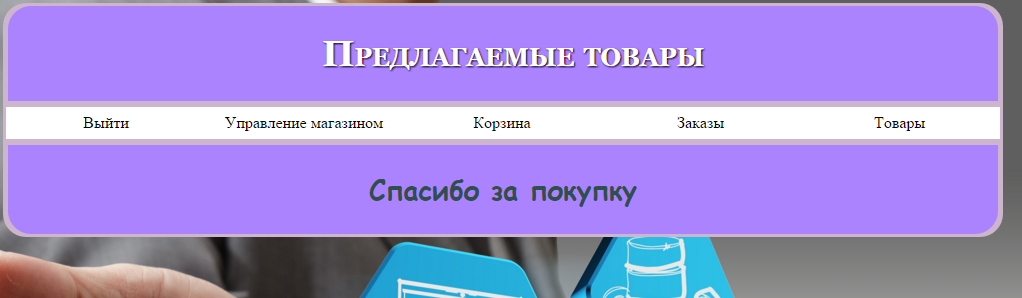


Рисунок 2. Thanks.xhtml

1. JSF-страница products.xhtml – приветствие пользователя и вывод списка товаров. Скриншот выполнен от лица клиента. Продавец может совершать CRUD-операции над товарами.

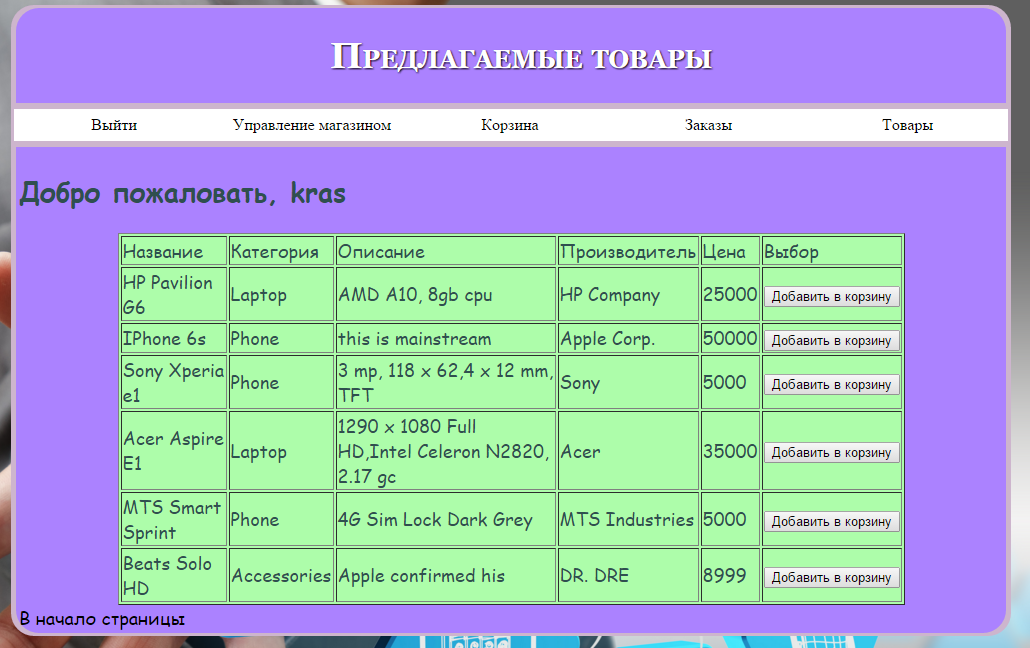


Рисунок 3. Products.xhtml

1. JSF-страница orderIsProcess.xhtml – окно для заполнения данных получателя, если клиент выбрал доставку на дом.

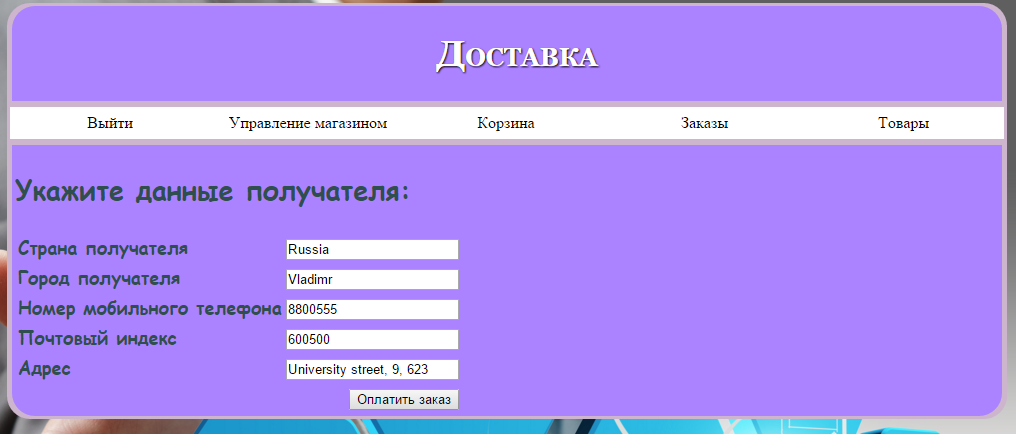


Рисунок 4. orderIsProcess.xhtml

1. JSF-страница orderInfo.xhtml – данная страница будет отображать заказы клиентов. Продавец может изменить состояние заказа. В дальнейшем планируется добавление поиска заказов по имени клиента. И добавление атрибута «Итоговая цена».

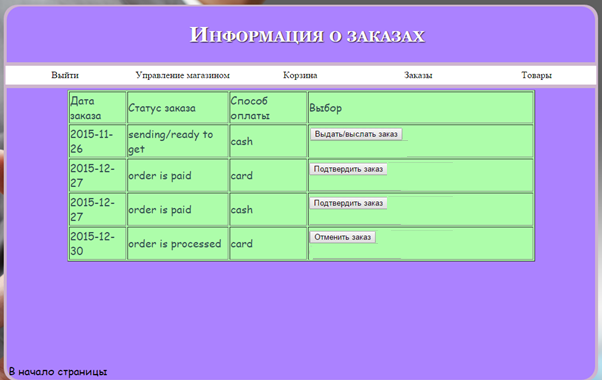


Рисунок 5. orderInfo.xhtml

1. JSF-страница order.xhtml – оформление заказа.

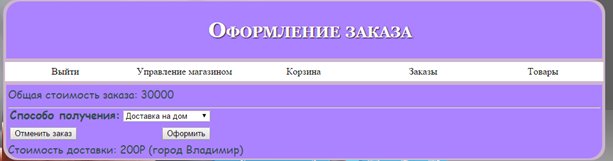


Рисунок 6. order.xhtml

1. JSF-страница payForOrder.xhtml – оплата заказа.



Рисунок 7. order.xhtml

1. JSF-страница newUser.xhtml – регистрация пользователя в системе. По умолчанию присваивается роль клиента.

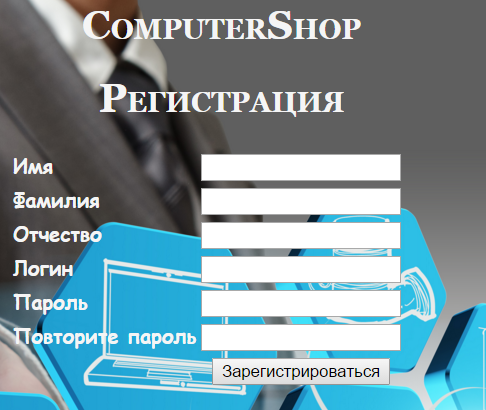


Рисунок 7. newUser.xhtml

1. JSF-страница login.xhtml – вход в систему.



Рисунок 8. Login.xhtml

1. JSF-страница error.xhtml – произошла какая-либо ошибка.

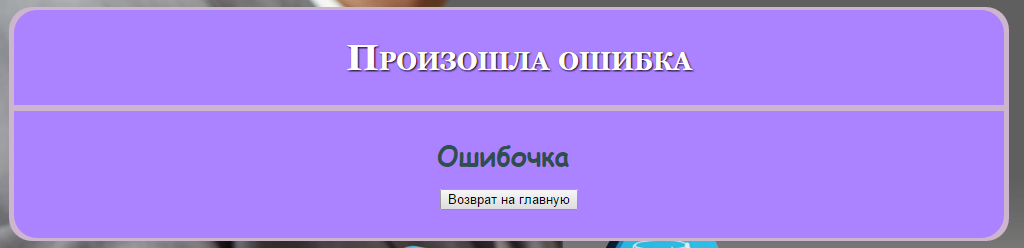


Рисунок 9. Error.xhtml

1. JSF-страница editProduct.xhtml

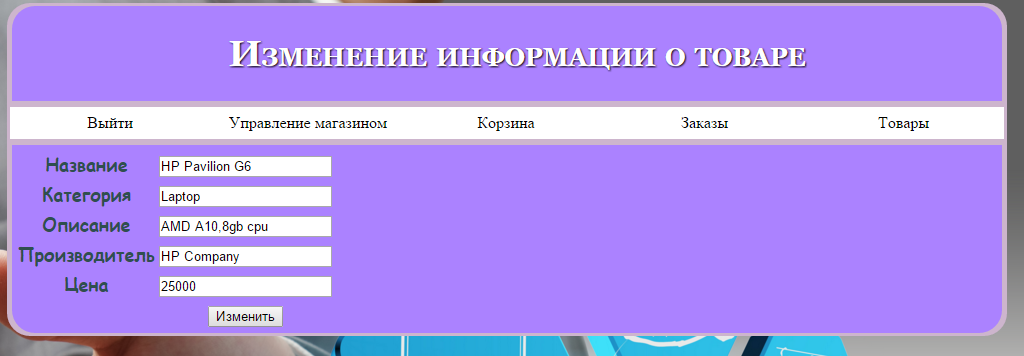


Рисунок 10. editProduct.xhtml

1. JSF-страница controller.xhtml – основной функционал работников магазина.

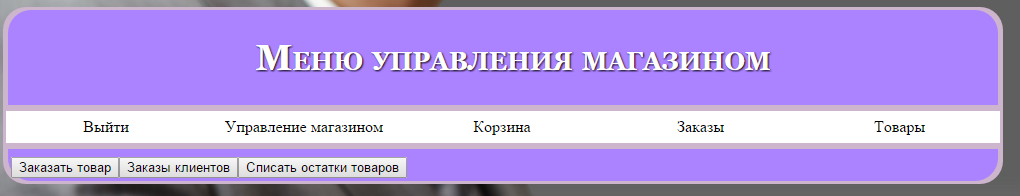


Рисунок 11. Controller.xhtml

1. JSF-страница clientOrders.xhtml – история покупок клиента.

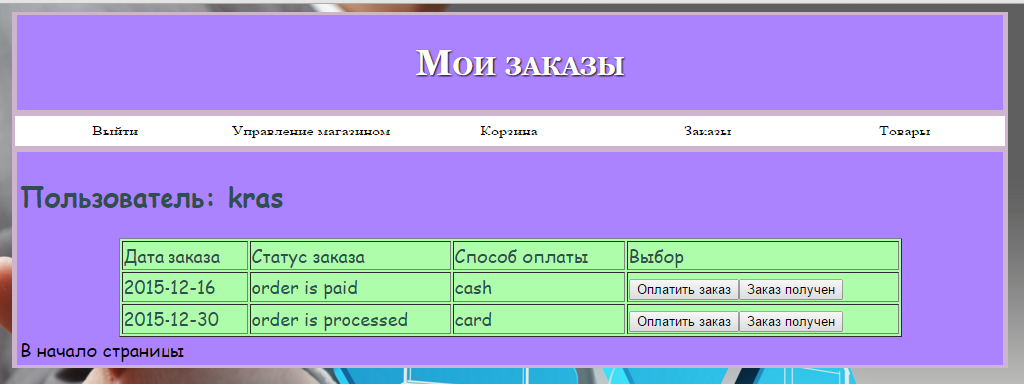


Рисунок 12. clientOrders..xhtml

1. JSF-страница basket.xhtml – корзина клиента.

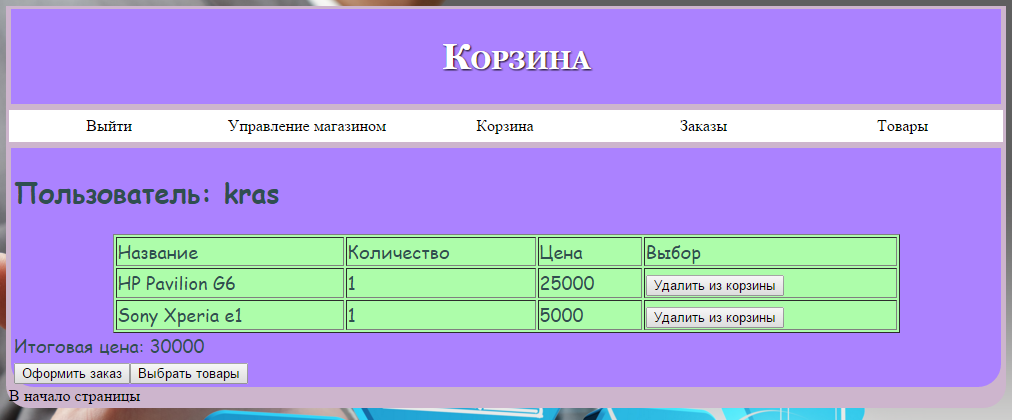


Рисунок 13. basket.xhtml

1. JSF-страница badConnectionToDataBase.xhtml - страница ошибок, связанная с БД.
2. JSF-страница addProduct.xhtml – заказ товара в магазин.

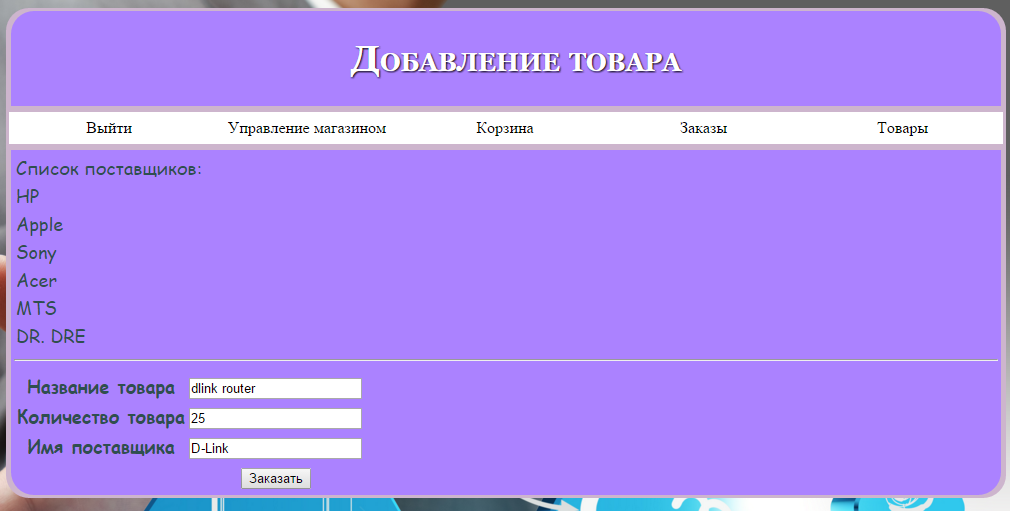


Рисунок 14. addProduct.xhtml

1. JSF-страница 500.xhtml
2. JSF-страница 404.xhtml
3. JSF-страница 403.xhtml
4. JSF-страница currentAccount.xhtml – страница личного счета клиента. Здесь он может его пополнить, либо посмотреть историю покупок. Если пользователь – администратор, то на странице будет показан оборот продаж и капитал магазина.

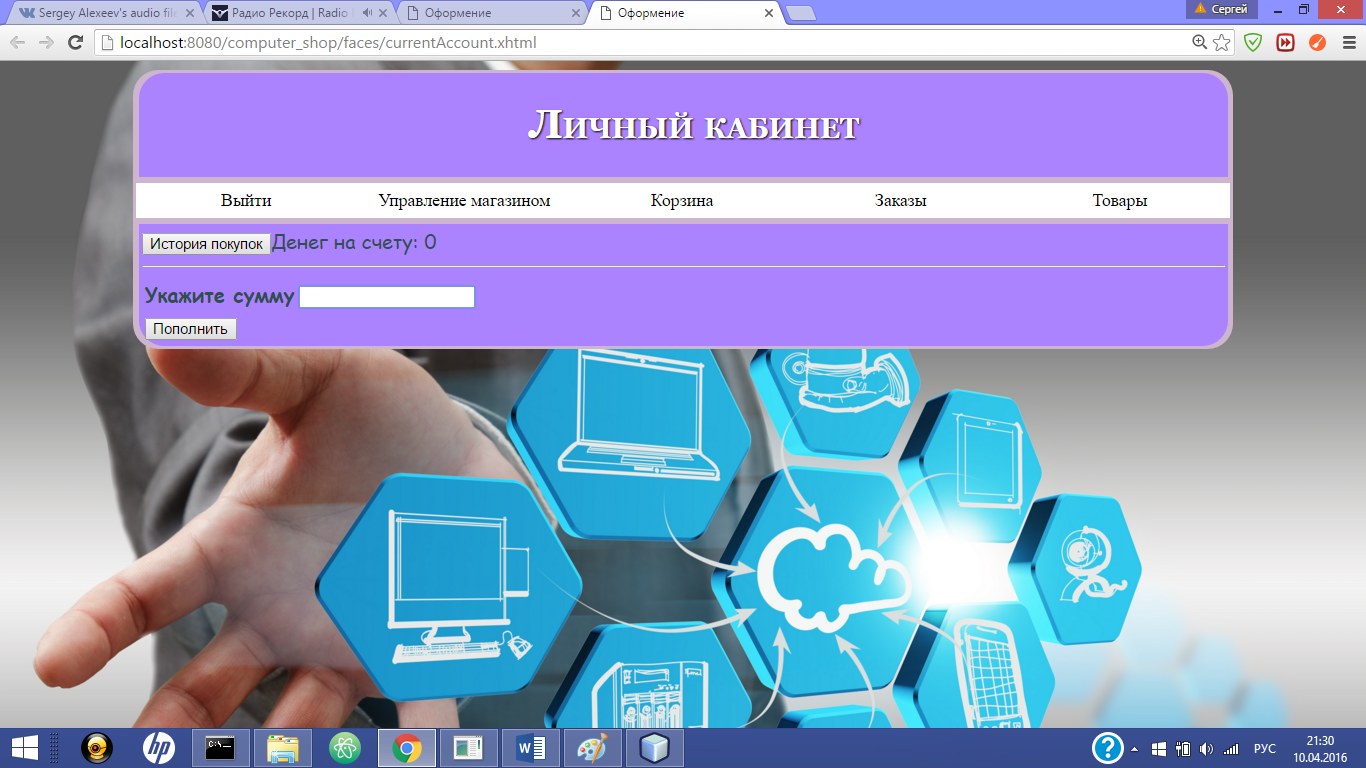


Рисунок 15. currentAccount.xhtml

В ходе разработки системы, будут происходить изменения в дизайне и функционале.

Диаграмма классов сущностей.

Диаграммы классов показывают статическую структуру системы, то есть определяют типы объектов системы и различного рода статические связи и отношения между ними. Диаграммы классов содержат набор статических элементов, как, например, классы, типы, их связи, объединенные в граф.

Диаграмма классов сущностей информационной системы «Магазин компьютерной техники» приведена на рисунке 16.

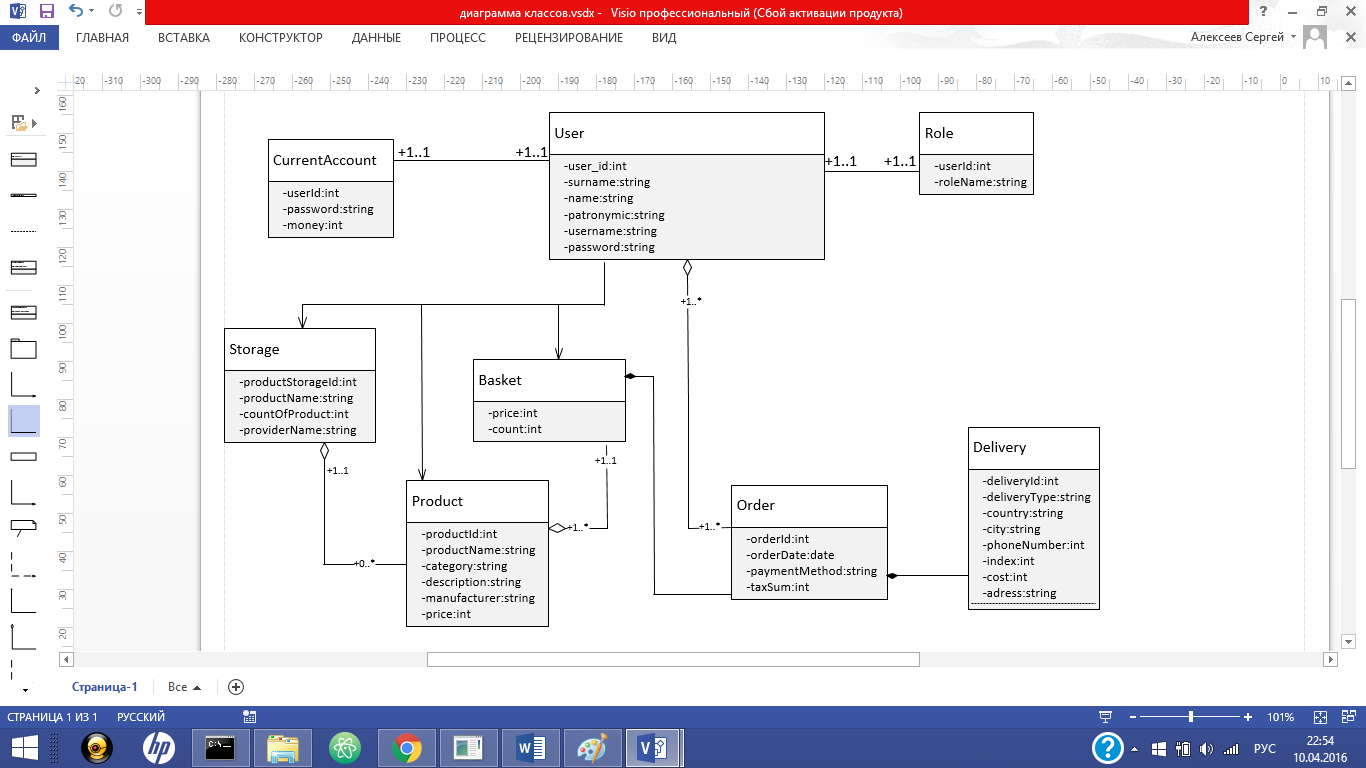


Рисунок16. Диаграмма классов сущностей.

**Описание классов.**

Пользователь (User) – содержит информацию о пользователе.

Атрибуты:

* userName – имя пользователя.
* password – пароль пользователя.
* userId – номер пользователя.
* Name - Имя пользователя.
* Surname - Фамилия пользователя.
* Patronymic - Отчество пользователя.

Роль (Role) – роль пользователя.

Атрибуты:

* userId – номер пользователя.
* roleName – название роли.

Лицевой счет (CurrentAccount) – информация о счете пользователя. У администратора свой счет, который означает капитал магазина.

Атрибуты:

* userId – номер пользователя.
* password – дополнительный пароль.
* Money – количество денежных средств (в рублях).

Товар (Product) – содержит информацию о товарах магазина

Атрибуты:

* productName – название товара
* category – категория товара
* manufacturer – производитель товара
* description – описание товара
* price – цена товара (оптовая)

Корзина (Basket) – содержит информацию о товаре и его количестве.

Атрибуты:

* productID – номер продукта
* count – количество

Заказ (Order) – содержит информацию о заказе.

Атрибуты:

* orderDate – дата совершения заказа.
* paymentMethod – способ оплаты заказа.
* orderStatus – состояние заказа.
* taxSum – итоговая цена заказа.

Доставка (Delivery) – содержит информацию о доставке.

Атрибуты:

* deliveryType – тип получения.
* deliveryStatus – состояние доставки.
* country – страна получателя.
* city – город получателя.
* phoneNumber – номер телефона получателя.
* index – почтовый индекс.
* cost – стоимость доставки.
* adress – адрес доставки.

Storage – информация о заказах на склад.

Атрибуты:

* productId – номер товара.
* countOfProduct – количество товара.
* providerName – имя поставщика.

Диаграмма последовательностей.

**Описание прецедента «Добавить товар в корзину»**

Расширенное описание прецедента «Добавить товар в корзину»

Название: «Добавить товар в корзину»

Предусловие: Клиент должен войти в систему.

Действующее лицо: Клиент

Основной поток: Заполнение корзины

Клиент открывает страницу (окно приложения), отображающую каталог товаров, предоставляемых компьютерным магазином. Клиент добавляет необходимые товары в корзину. После этого он получает возможность оформить заказ.

Альтернативный поток: Данного товара нет в каталоге. Клиент изменяет выбор.

Постусловие: Если товары есть в каталоге, то клиент получает право оформить заказ.

Диаграмма последовательности является одной из разновидности диаграмм взаимодействия и предназначена для моделирования взаимодействия объектов Системы во времени, а также обмена сообщениями между ними.

Диаграмма последовательностей прецедента «Добавить товар в корзину» приведена на рисунке 17.

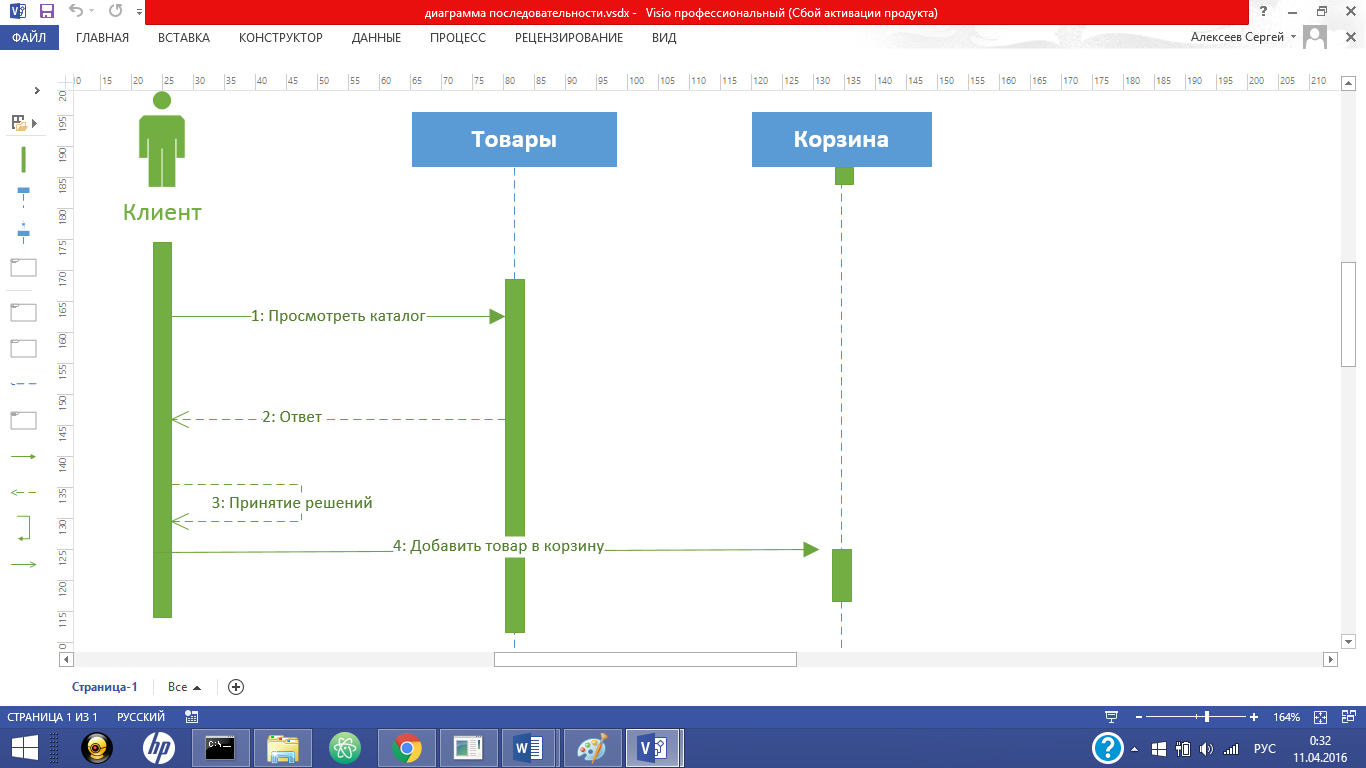


Рисунок17. Диаграмма последовательности для прецедента «Добавить товар в корзину»

 Сводная диаграмма классов/компонентов приложения с указанием связей между компонентами различных слоев.

Слой представления - это единственный слой, непосредственно связанный с пользователем. Слой представления используется для получения данных от пользователя и передачи их слою бизнес-логики для дальнейшей обработки, и когда данные получены в объекте-значении, он отвечает за представление объекта-значения в надлежащей форме, которая понятна пользователю.

Слой бизнес-логики (BUS) работает как мост между уровнем представления и DAO. Все значения пользователя, полученные от уровня представления, передаются в BUS. Слой бизнес-логики содержит всю бизнес-логику программы.

Слой доступа к базе данных (DAO) строит запрос на основе параметров, полученных от слоя бизнес-логики и используя соединение с базой данных, выполняет требуемые операции. Затем происходит возврат результатов из DAO классов в слой бизнес-логики.

Сводная диаграмма классов/компонентов приложения приведена на рисунке 18.



Рисунок18. Диаграмма компонентов и их взавимодействия

Спецификация компонентов системы.

**Слой бизнес-логики.**

1. **CircuitDAO**
   1. void addCircuit(int currentStop) – метод добавления звена схемы маршрута в базу данных, состоящего из одной остановки. Входные параметры – id добавляемой в схему остановки.
   2. void createFirstCircuit() – метод создания первой схемы маршрута исходя из значений «временной» таблицы в базе данных.
   3. void changeCircuit() – метод для обновления предыдущего звена в схеме маршрута при добавлении нового, то есть установка значения следующего звена.
   4. void createSecondCircuit() – метод создания второй схемы маршрута исходя из значений «временной» таблицы в базе данных.
   5. void deleteOneCircuit(int idCircuit, int idRoute) – метод, удаляющий одно звено из схемы маршрута. Входные параметры – номер звена в схеме и номер маршрута, к которому принадлежит схема. Метод использует методы helperDeleter, helperDeleter1, helperDeleter3, helperDeleter5.
   6. int helperDeleter(int idCircuit) – метод, возвращающий значение – ссылку на следующее звено из удаляемого звена схемы. Входные параметры – номер удаляемого звена.
   7. int helperDeleter1(int idCircuit) – метод, возвращающий значение – ссылку на звено, ссылающееся на удаляемой звено схемы. Входные параметры – номер удаляемого звена.
   8. boolean helperDeleter2(int cur, int prev) – метод, который возвращает значение true – если звено – первое, иначе возвращает false и обновляет ссылки в схеме маршрута, исходя из того, является ли удаляемое звено в цепи последним или в середине. Входные параметры – номера удаляемого звена и звена предыдущего.
   9. void helperDeleter5(int idCircuit, int idNext, intIdRoute) – метод, производящий обновлении данных в маршруте, если удаляемое звено – первое в схеме маршрута. Входные параметры – номер удаляемого звена, номер следующего звена и номер маршрута, из которого удаляется звено.
   10. int findRouteFirst(int idRoute) – метод, который находит значение первой схемы маршрута. Входные параметры – номер маршрута.
   11. int findRouteSecond(int idRoute) – метод, который находит значение второй схемы маршрута. Входные параметры – номер маршрута.
   12. boolean routeFirst(int idCircuit, int idRoute) – метод, проверяющий, найденное значение метода findRouteFirst является ли первой схемой маршрута. Входные параметры – номер маршрута, номер удаляемого звена.
   13. boolean routeSecond(int idCircuit, int idRoute) – метод, проверяющий, найденное значение метода findRouteSecond является ли второй схемой маршрута. Входные параметры – номер маршрута, номер удаляемого звена.
   14. void disableForeightKeys() – метод, выключающий проверку соответствия внешних ключей для удаления одного звена.
   15. void enableForeightKeys() – метод, включающий проверку соответствия внешних ключей после удаления одного звена.
   16. void updateRouteStops(int idRoute, int idFirstScheme, int idSecondScheme) – метод, обновляющий информацию о маршруте, в случае удаления его первой или последней остановок в схемах. Входные данные – номер маршрута, номер первой схемы и номер второй схемы. Использует методы idFirstStop и idLastStop.
   17. int idFirstStop(int idFirstScheme) – метод, возвращающий номер остановки из схемы. Входные параметры – номер схемы.
   18. int idLastStop(int idSecondScheme) – метод, возвращающий номер остановки из схемы. Входные параметры – номер схемы.
2. **RouteDAO**
   1. void addRoute(int Number) – метод добавления маршрута в БД. Входные параметры – номер маршрута, который будет ему присвоен. Метод использует методы из CircuitDAO, такие как createFirstCircuit, createSecondSircuit. И методы свеого класса – routeSecondScheme, routeFirstScheme, routeFirstStop, routeLastStop.
   2. int routeFirstScheme() – метод для получения id первой схемы создающегося маршрута.
   3. int routeSecondScheme() – метод для получения id второй схемы маршрута.
   4. int routeFirstStop(int idFirstScheme) – метод для получения первой остановки из схемы маршрута. Входные параметры – номер первой схемы маршрута.
   5. int routeLastStop(int idSecondScheme) – метод для получения последней остановки из схемы маршрута. Входные параметры – номер первой схемы маршрута.
   6. List<Route> getAllRoutes() – метод для вывода всех маршрутов из базы данных на страницу.
   7. Circuit getCircuit(int idCircuit) – метод создания связанного списка по id первого звена в схеме маршрута. Входные параметры – номер звена схемы. Использует метод getStop из класса StopDAO.
   8. List<Circuit> getRouteFirstCircuit(int idRoute) – метод для вывода первой схемы маршрута из БД на страницу. Входные параметры – номер маршрута.
   9. List<Circuit> getRouteSecondCircuit(int idRoute) – метод для вывода второй схемы маршрута из БД на страницу. Входные параметры – номер маршрута.
   10. void deleteRoute(int idRoute, int first, int second) – метод для удаления маршрута. Входные параметры – номер маршрута, номер первой схемы и второй. Использует методы deleteCircuit.
   11. void deleteCircuit (int selectedIdCircuit) – удаление цепочки схемы из БД для всего маршрута. Входные параметры – номер схемы. Использует метод из CircuitDAO для получения ссылки на следующий элемент звена.
   12. void cancelAddRoute() – метод очистки «временных таблиц» для создания маршрута.
3. **StopDAO**
   1. List<Stop> getAllStops() – метод для вывода списка всех остановок.
   2. void addStop(String name, Double coordX, Double coordY) – добавление остановки в базу данных. Входные параметры – название остановки и ее координаты по оси X и Y.
   3. Stop getStop(int idStop) – получение объекта остановки по ее id в БД. Входные параметры – номер остановки.
   4. void deleteStop(List<Stop> selectedList) – удаление выбранных остановок из БД. Входные параметры – список объектов остановок.
   5. void addStopForFirstCircuit(Stop stop) – метод добавления остановки в БД для первой схемы маршрута. Входные параметры – остановка.
   6. void addStopForSecondCircuit(Stop stop) – метод добавления остановки в БД для второй схемы маршрута. Входные параметры – остановка.
   7. List<Stop> getFirstStop() – метод для вывода списка всех остановок первой схемы маршрута.
   8. List<Stop> getFirstStop() – метод для вывода списка всех остановок второй схемы маршрута.
   9. void mirrorRoute() – метод для создания второй схемы маршрута зеркально первой.
   10. void editStop(int id, String name, Double coordX, Double coordY) – метод для редактирования уже существующей остановки. Входные параметры – номер остановки, ее название и координаты.
4. **UserDAO**
   1. List<User> getAllUsers() – метод для вывода всех зарегистрированных пользователей системыиз БД на страницу.
   2. void addUser(String username, String password, String role) – метод для добавления нового пользователя в БД, регистрация. Входные параметры – логин, пароль и роль пользователя. Использует метод addUsername и addRole.
   3. void addUsername(String username, String password) – метод для добавления пользователя в таблицу myuser в БД. Входные параметры – логин и пароль пользователя.
   4. void addRole(String username, String role) – метод для добавления роли пользователю. Входные параметры – логин и роль пользователя.
5. ReportDAO
   1. List<Report> getAllReports() – метод для вывода отчетов о доходах всех маршрутов.
   2. void addReport(int idRoute, int numberOfTickets) – добавление отчета о доходе маршрута в базу данных. Входные параметры – номер маршрута и количество билетов. Использует методы routePrice, userValid.
   3. void deleteReport(List<Report> selectedList) – удаление выбранных отчетов из БД. Входные параметры – список объектов отчетов.
   4. void editReport(int id, int numberOfTickets) – метод для редактирования уже существующего отчета. Входные параметры – номер маршрута и количество билетов.
   5. int routePrice(int idRoute) – метод для получения стоимости одного билета из таблицы маршрутов. Входные параметры – номер маршрута.
   6. boolean userValid(String username, int idRoute) – метод для проверки возможности создания отчета на выбранный маршрут. Входные параметры – логин и номер маршрута.

**Слой представления.**

1. **UserBean**
   1. List<User> getAllUsers() – метод для вывода на страницу списка пользователей. Использует метод UserDAO.getAllUsers. Возвращает список пользователей.
2. **AddUserBean**
   1. String logout() – метод для выхода пользователя из авторизованного состояния. Возвращает ссылку на главную страницу.
   2. String addNewUser() – метод добавления пользователя в БД. Использует методы UserDAO.addUser. Возвращает ссылку на главную страницу.
3. **RouteBean**
   1. void actionListener(ActionEvent event) – метод для удаления выбранного маршрута из БД. Использует методы RouteDAO.getFirst, RouteDAO.getSecond.
   2. List<Route> getAllRoutes() – метод для вывода всех маршрутов на страницу. Использует метод RouteDAO.getAllRoutes.
   3. List<Circuit> getFirstCircuit(int idRoute) – метод для вывода схемы маршрута в первую сторону. Входные параметры – номер маршрута. Использует метод RouteDAO.getRouteFirstCircuit.
   4. List<Circuit> getSecondCircuit(int idRoute) – метод для вывода схемы маршрута во вторую сторону. Входные параметры – номер маршрута. Использует метод RouteDAO.getRouteSecondCircuit.
   5. String redirectRouteInfo(int number) – метод для сохранения значения номера маршрута в сессию. Возвращает ссылку на страницу схемы маршрута.
4. **AddRouteBean**
   1. String AddNewRoute() – метод для добавления нового маршрута. Использует метод RouteDAO.addRoute.
   2. String cancelAddRoute() – метод отмены создания маршрута. Использует метод RouteDAO.cancelAddRoute.
5. **StopBean**
   1. void actionListener(ActionEvent event) – метод для удаления выбранных остановок из БД. Использует метод StopDAO.deleteStop.
   2. List<Stop> getAllStops() – метод для вывода всех остановок на страницу. Использует метод StopDAO.getAllStops.
   3. List<Stop> getFirstStops() – метод для вывода остановок первой схемы маршрута на страницу. Использует метод StopDAO.getFirstStop.
   4. List<Stop> getSecondStops() – метод для вывода остановок второй схемы маршрута на страницу. Использует метод StopDAO.getSecondStop.
   5. String editStop(int idStop) – метод для запоминания id редактируемой остановки в сессии. Возвращает ссылку на страницу редактирования остановки.
6. **AddStopBean**
   1. String addNewStop() – метод для добавления остановки в БД. Использует метод StopDAO.addStop. Возвращает ссылку на страницу со всеми остановками.
   2. String editStop(int idStop) – метод для редактирования остановки. Использует метод StopDAO.editStop. Возвращает ссылку на страницу со всеми остановками.
   3. void addNewFirstStop(ActionEvent event) – метод для добавления выбранных остановок во «временную» таблицу в БД для первой схемы маршрута. Использует метод StopDAO.addStopForFirstCircuit.
   4. void addNewSecondStop(ActionEvent event) – метод для добавления выбранных остановок во «временную» таблицу в БД для второй схемы маршрута. Использует метод StopDAO.addStopForSecondCircuit.
   5. void mirror() – метод для зеркального отображения второй схемы маршрута относительно первой. Использует метод StopDAO.mirrorRoute.
7. **CircuitBean**
   1. void deleteOne(int idCircuit, int idRoute) – метод для удаления одного звена в маршруте. Использует метод CircuitDAO.deleteOneCircuit.
8. **ReportBean**
   1. List<Report> getAllReports() – метод для вывода на страницу списка всех отчетов. Использует метод ReporDAO.getAllReports. Возвращает список отчетов о доходах.
   2. String editReport(int idReport) – метод для запоминания id редактируемого отчета в сессии. Возвращает ссылку на страницу редактирования отчета.
9. **AddReportBean**
   1. String addNewReport() – метод для добавления отчета в БД. Использует метод ReportDAO.addReport. Возвращает ссылку на страницу со всеми отчетами.
   2. String editReport(int idReport) – метод для редактирования отчета. Использует метод ReportDAO.editReport. Возвращает ссылку на страницу со всеми отчетами.