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

Кряжев С. А.

**Принял:**

Тимофеев А. А.

Владимир 2016

Схема веб интерфейса системы

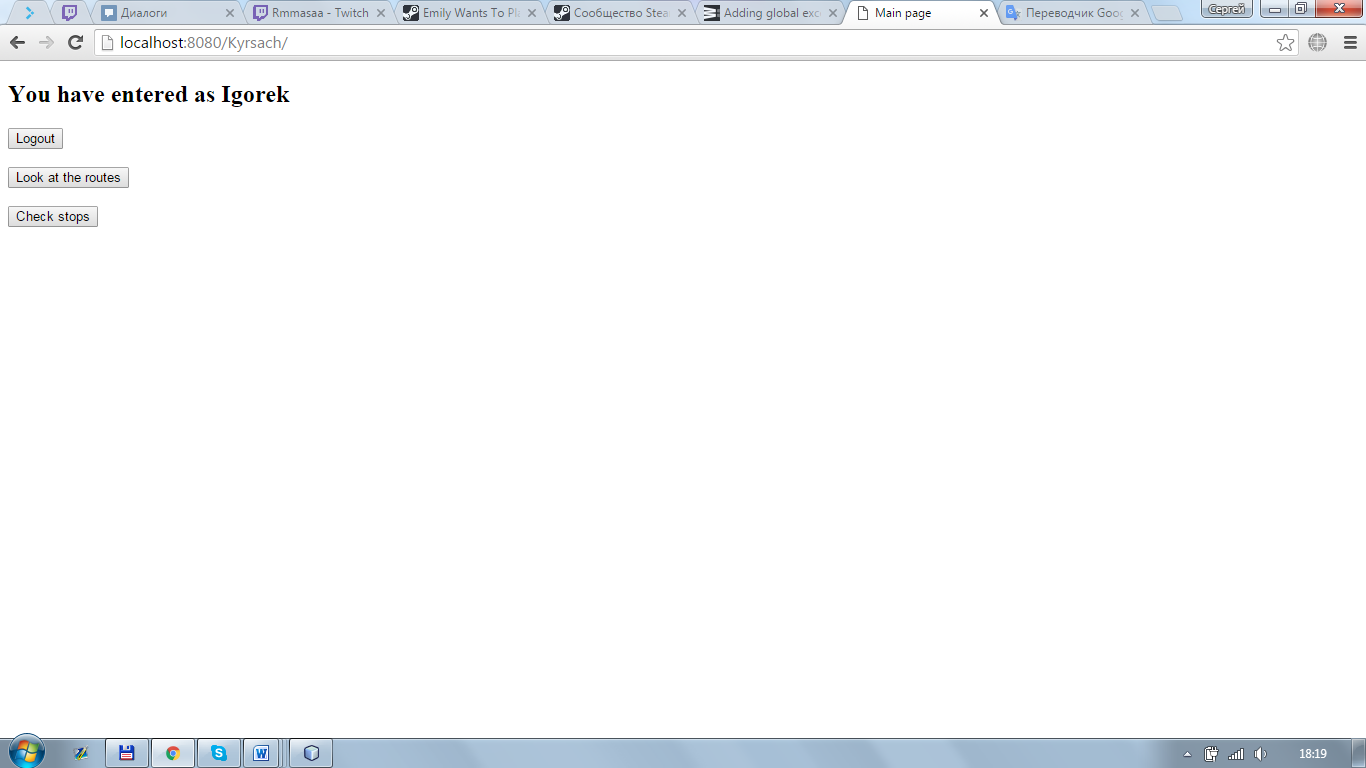
Современные концепции проектирования распределенных информационных систем корпоративного уровня предполагают общение пользователя системы с базой данных посредством браузера и технологий Internet. Для обеспечения возможности взаимодействия пользователя с браузером применяется Веб-интерфейс — это совокупность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

Веб-интерфейс моего курсового проекта по теме «Городской автотранспорт» приведен на рисунке 1.

Рисунок1. Схема веб-интерфейса системы

**Описание веб-страниц.**

1. index.xhtml – стартовая страница приложения. На ней располагаются приветственное сообщение, а также ссылки на страницу маршрутов, остановок, зарегистрированных пользователей системы, отчетов, регистрацию и авторизацию. Пользователь попадает на нее при открытии приложения, а так же при переходе на нее с других страниц, нажав соответствующую ссылку.



Add report

Рисунок2. Главная страница приложения

1. login.xhtml – страница авторизации. На ней расположены поля для ввода логина и пароля и ссылка на главную страницу. Пользователь попадает на эту страницу с главной при нажатии на кнопку «Войти в систему».

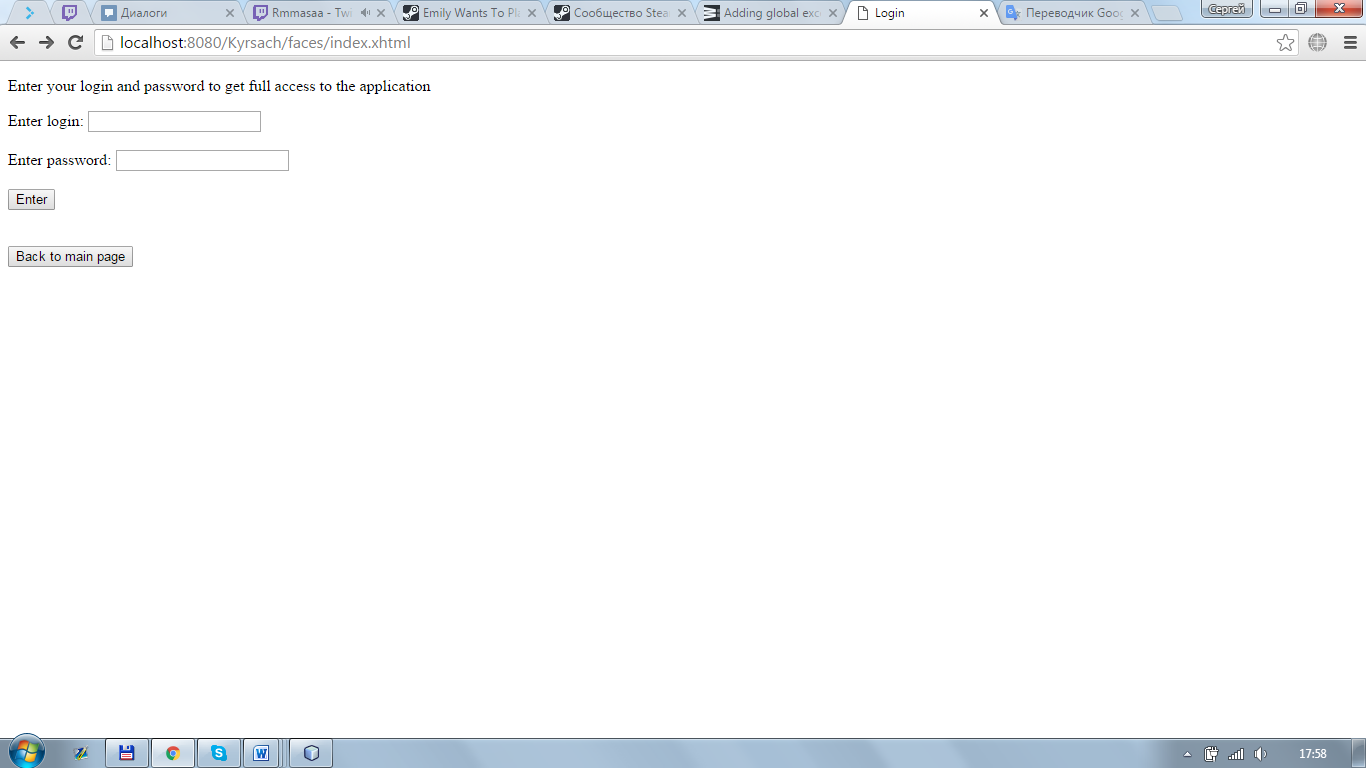


Рисунок3. Страница входа в систему

1. loginerror.xhtml – страница ошибки авторизации. На ней расположено сообщение о неверно указанной пользователем комбинации логина и пароля, а также ссылка на страницу авторизации. Пользователь попадает на эту страницу, если введенной им комбинации пользователя и пароля не обнаружилось в базе данных.
2. allUsers.xhtml – страница зарегистрированных пользователей системы. На ней находится таблица, в которой содержится информация о логинах пользователей и их ролях в системе, а также ссылка на страницу добавления нового пользователя. На данную страницу пользователь попадает, нажав на ссылку на главной странице, находясь в системе под ролью администратора.

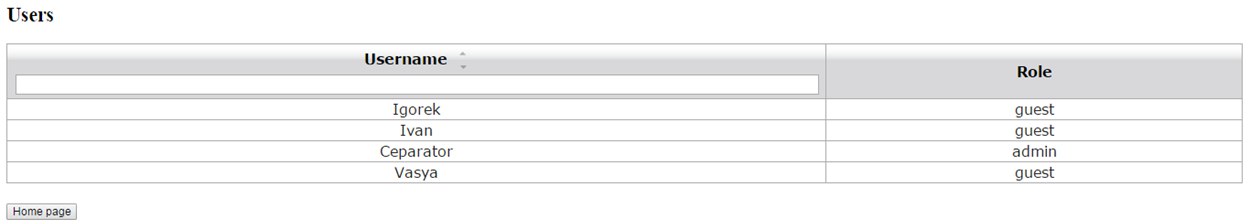


Рисунок4. Страница всех пользователей

1. addUser.xhtml – страница регистрации нового пользователя. На ней находятся поля для ввода информации о новом пользователе, а также ссылки на переход на главную страницу и на страницу со списком пользователя. Незарегистрированный пользователь попадает на данную страницу путем нажатия кнопки «Зарегистрироваться» на главной странице, администратор может попасть на эту страницу со страницы всех пользователей для добавления нового пользователя с нужной ему ролью.



Рисунок5. Страница регистрации нового пользователя

1. allReports.xhtml – страница отчетов о доходах. На ней располагается таблица со сведениями о доходах маршрута. На эту страницу попадает пользователь с ролью «администратор» и «сотрудник» с главной страницы путем нажатия на ссылку.

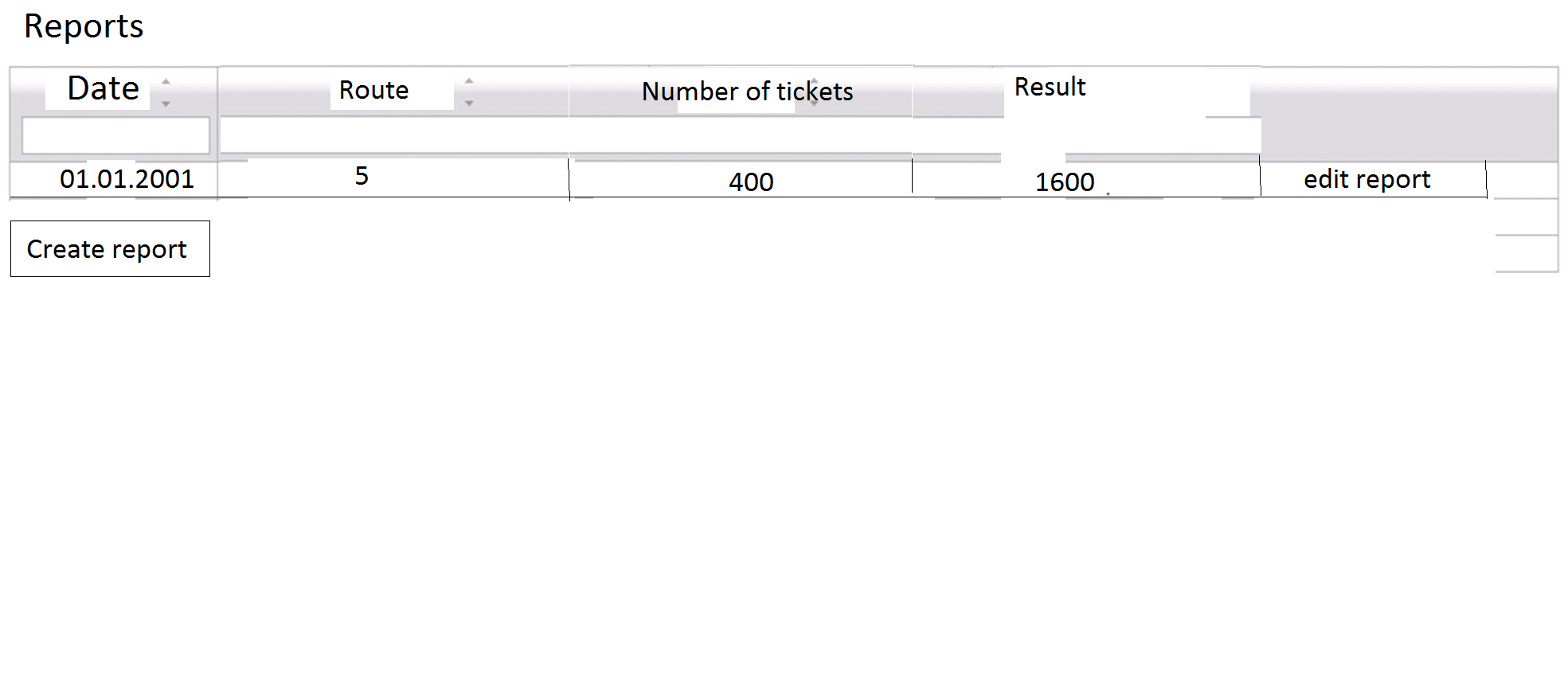


Рисунок6. Страница всех отчетов о доходах

1. addReport.xhtml – страница добавления отчета. На ней находятся поля для ввода информации о добавляемом отчете и кнопка перехода на страницу всех отчетов. Пользователь попадает на эту страницу ссылкой со страницы всех отчетов, желая добавить новый отчет.

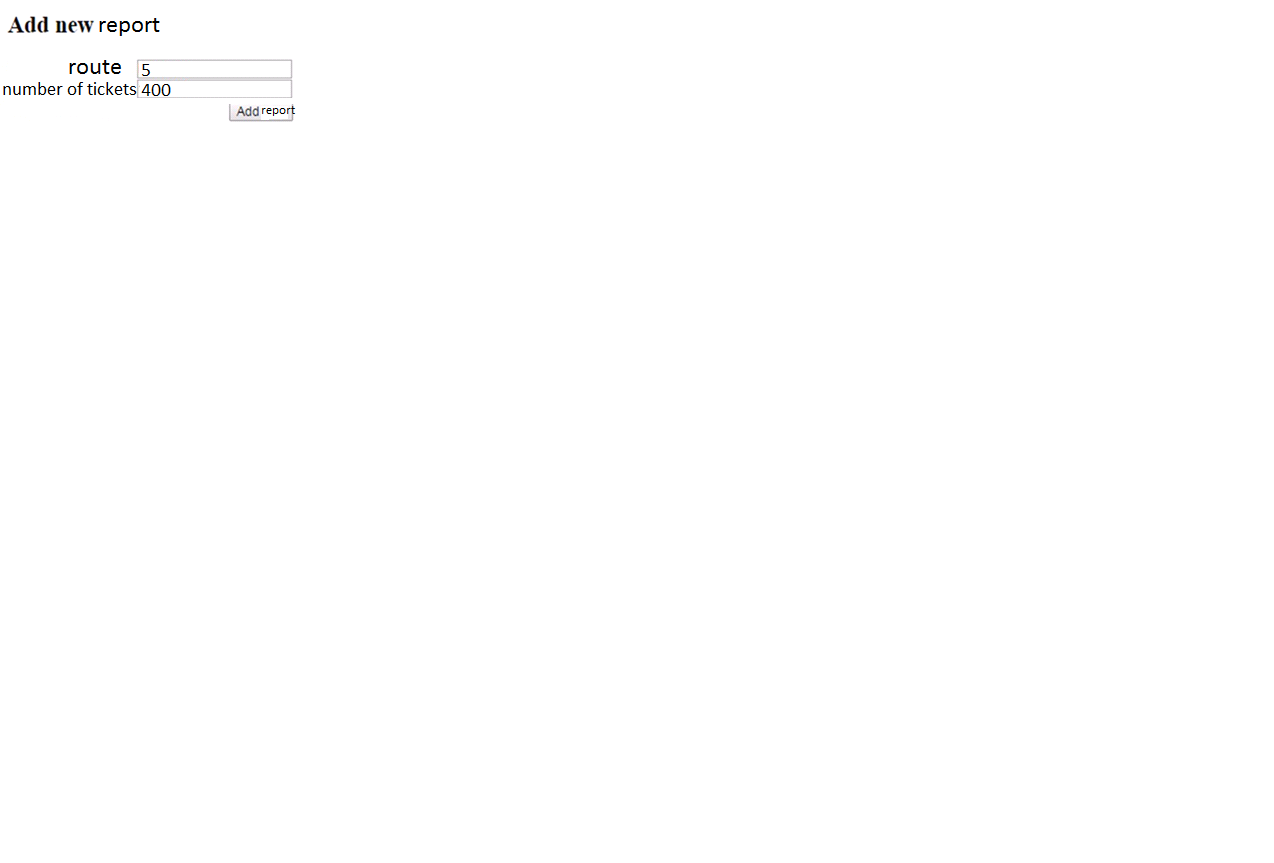


Рисунок7. Страница добавления нового отчета

1. editReport.xhtml – страница редактирования ответа. На ней находятся поля для ввода информации о редактируемом отчете и кнопка перехода на страницу всех отчетов. Пользователь попадает на эту страницу ссылкой со страницы всех отчетов, желая отредактировать существующий отчет.

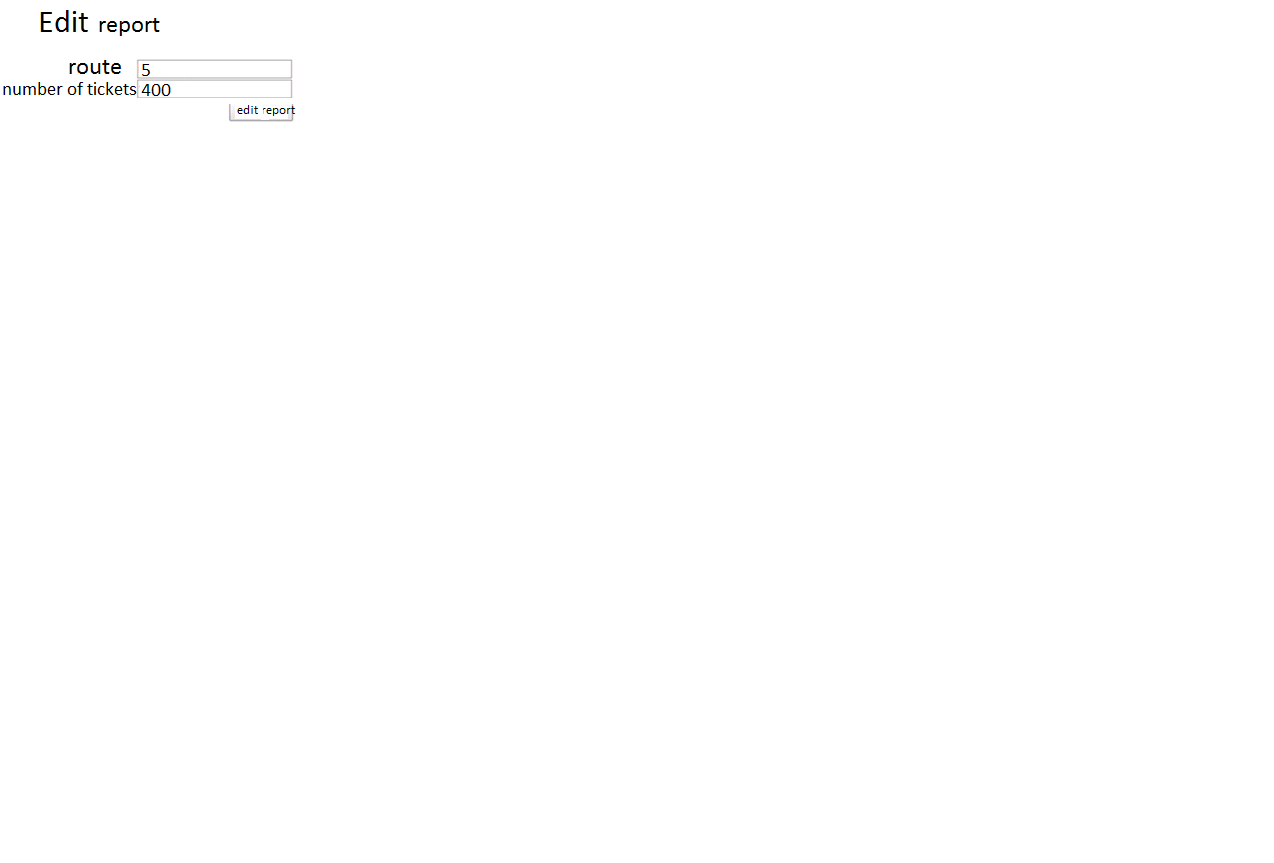


Рисунок8. Страница редактирования отчета

1. allRoutes.xhtml – страница маршрутов. На ней находится таблица, содержащая номера маршрутов, их первую и последнюю остановки, а так же ссылки навигации по сайту. Пользователь попадает на эту страницу, нажав нужную ссылку на других страницах.

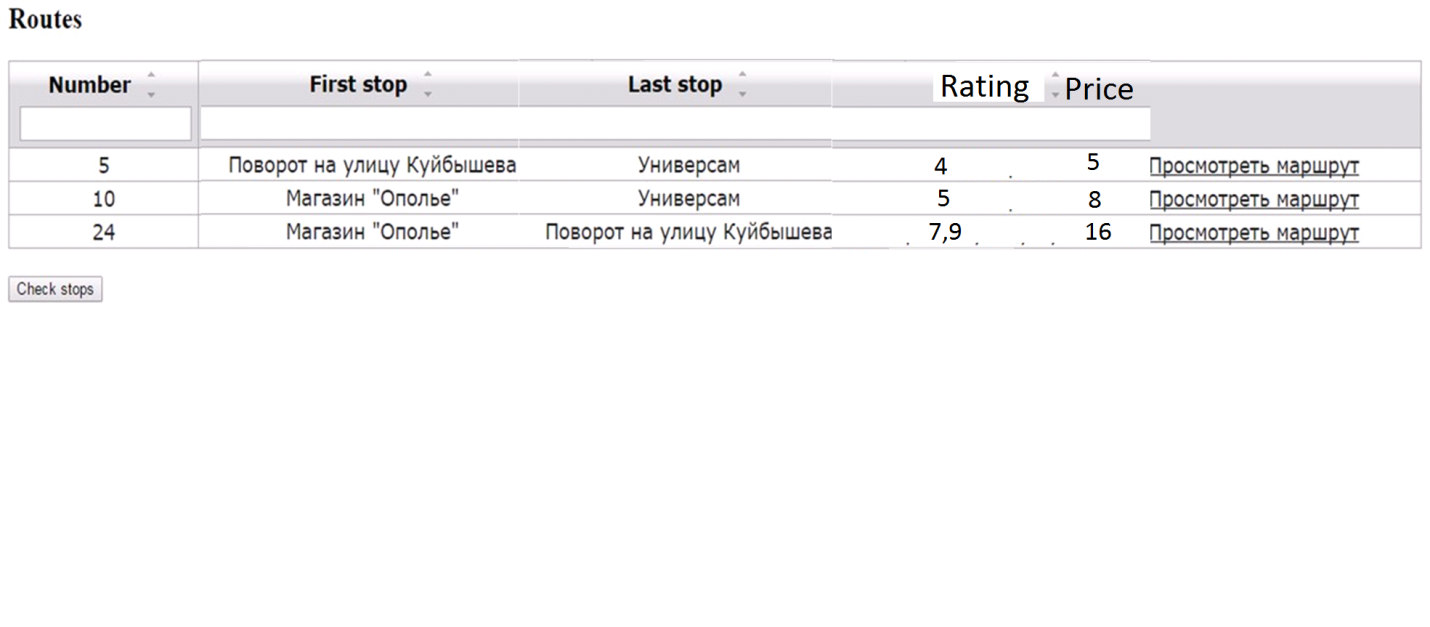
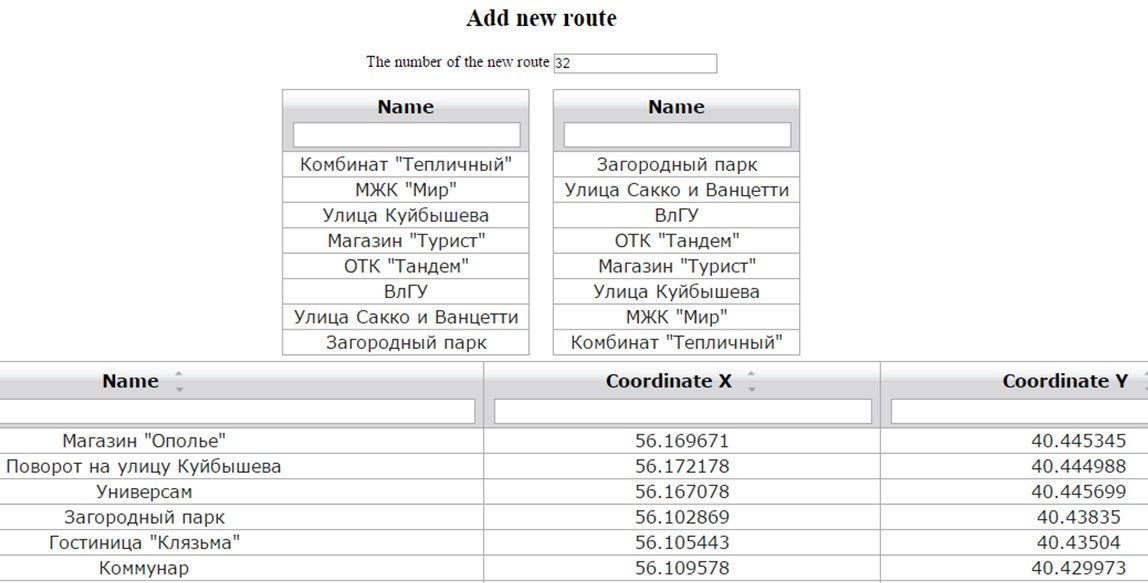


Рисунок10. Страница всех маршрутов

1. addRoute.xhtml – страница добавления маршрута. На ней находятся таблицы, описывающие схему маршрута в одну и другую стороны, поле для ввода номера маршрута, список остановок, из которых составляется схема маршрута и ссылка перехода на обратную страницу. Пользователь попадает на эту страницу, находясь в системе под ролью администратора, переходя по ссылке со страницы всех маршрутов.



Price of the route

Рисунок11. Страница создания нового маршрута

1. routeCircuit.xhtml – страница схемы маршрута. На ней располагается карта с остановками, на которых останавливается конкретный маршрут, а так же табличный вариант его схем движения. Пользователь может попасть на эту страницу по ссылке со страницы всех маршрутов.

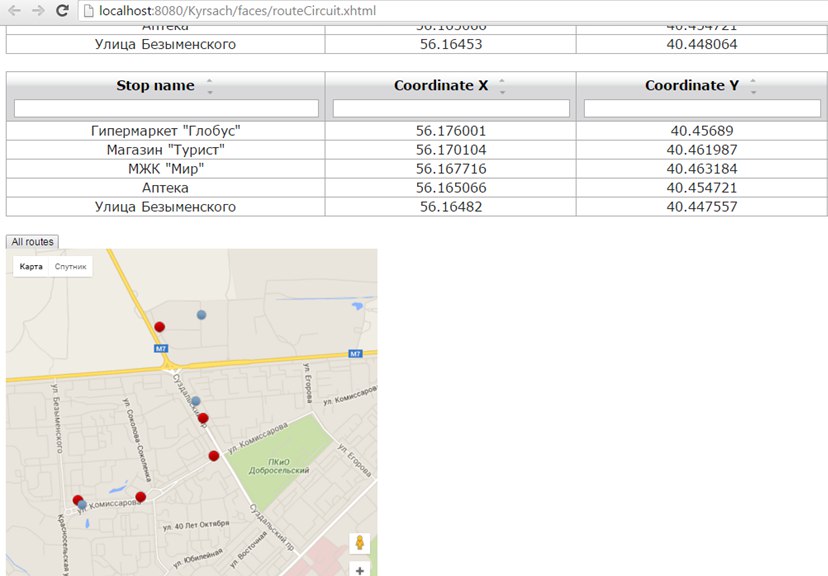


Рисунок12. Страница просмотра схемы маршрута

1. allStops.xhtml – страница остановок. Содержит список всех остановок, которые могут использоваться в маршрутах. Пользователь может попасть на эту страницу по ссылке с главной страницы или со страницы всех маршрутов.

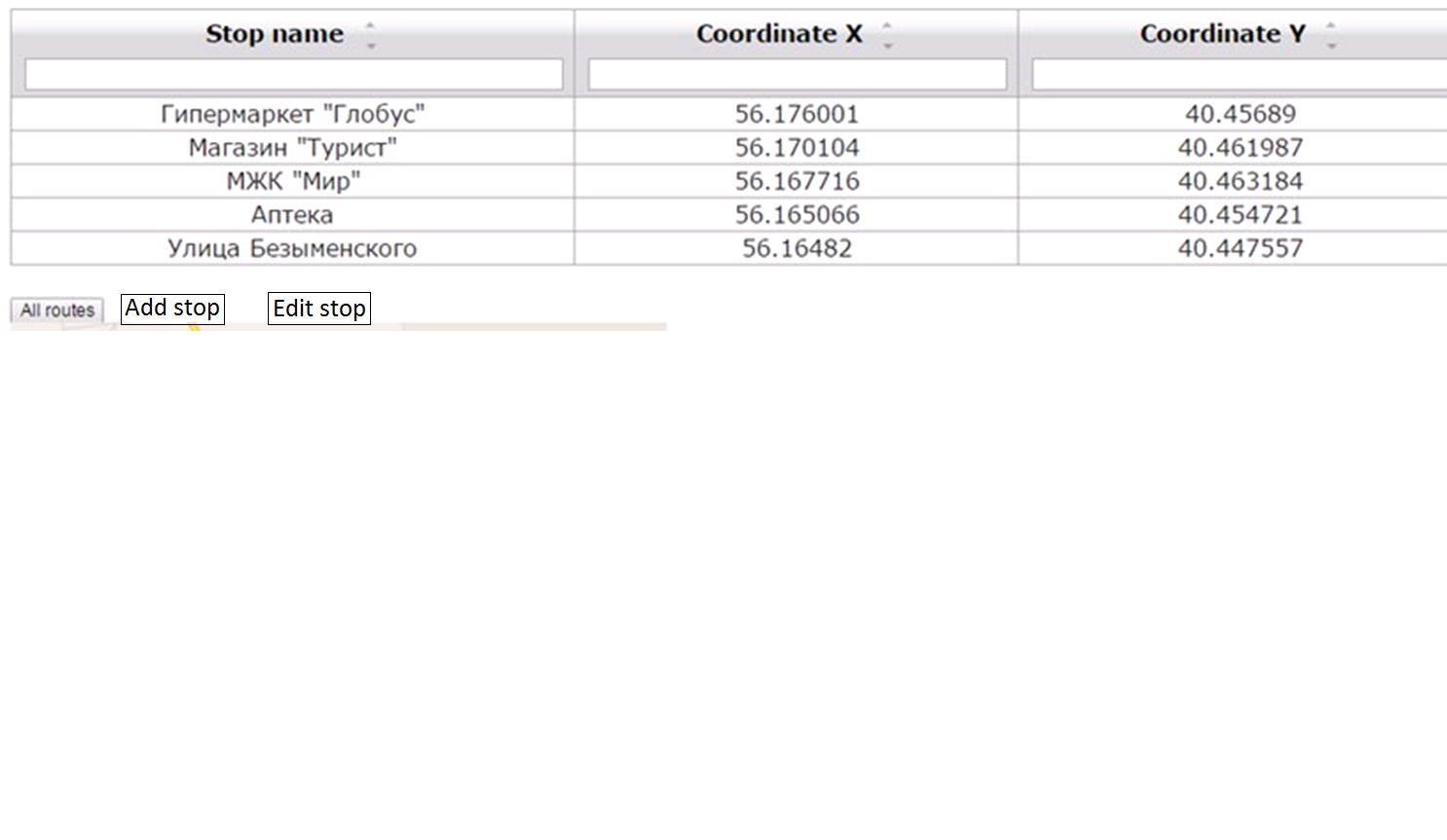


Рисунок13. Страница просмотра всех остановок.

1. addStop.xhtml – страница добавления остановки. На ней находятся поля для ввода информации о добавляемой остановке и кнопка перехода на страницу всех остановок. Пользователь попадает на эту страницу ссылкой со страницы всех остановок, желая добавить новый остановку.

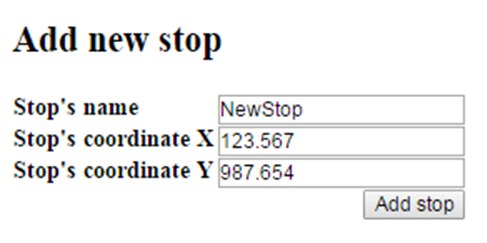


Рисунок14. Страница добавления новой остановки

1. editStop.xhtml – страница редактирования остановки. На ней находятся поля для ввода информации о редактируемой остановке и кнопка перехода на страницу всех остановок. Пользователь попадает на эту страницу ссылкой со страницы всех остановок, желая отредактировать существующую остановку.

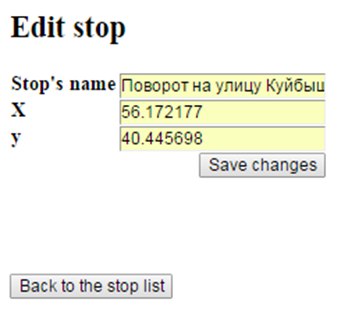


Рисунок15. Страница редактирования остановки

1. errorPage.xhtml – страница ошибок. Содержит сообщение о возникновении ошибки, дыб пользователь не был в неведении. На данную страницу пользователь будет попадать в случае возникновения в приложении ошибок рода 404, 500 и т.д.

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 «Городской автотранспорт» приведена на рисунке 16.



Рисунок16. Диаграмма классов сущностей.

**Описание классов.**

1. Класс Пользователь (myuser) определяет основные атрибуты, которыми обладает пользователь системы:

1) Логин (String username);

2) Пароль (String password);

3) Маршрут пользователя (int route) определяет, к какому маршруту относится сотрудник предприятия. Определяет его права на создание отчетов о доходе.

1. Класс роли пользователей (myuser\_roles) определяет соответствие пользователей с их ролями в системе:

1) Логин (String username);

2) Роль (String role);

1. Класс Остановки (Stops) представляет собой набор остановок, по которым совершается передвижение. Набор остановок представляет собой маршрут. Класс содержит следующие атрибуты:

1) Номер остановки (idStop);

2) Название (String name) – название остановки;

3) Координата по x (Double coordX) – расположение остановки на карте;

4) Координата по y (Double coordY).

1. Класс Цепь (Circuit) определяет схемы маршрутов и обладает следующими атрибутами:

1) Номер схемы (int idCircuit);

2) Остановку (Stops currentStop) – название текущей остановки маршрута;

3) Следующую остановку (Circuit nextStop) – ссылка на схему, в которой содержится следующая остановка в схеме маршрута.

1. Класс Маршрут (Route) маршрут движения транспорта, представляет собой набор остановок. Содержит атрибуты:

1) Идентификатор маршрута (int idRoute);

2) Номер маршрута (int number);

3) Первая остановка (Stops firstStop);

4) Последняя остановка (Stops lastStop);

5) Схема маршрута в одну сторону(Circuit firstCircuit) представляет собой список остановок;

6) Схема маршрута в другую сторону(Circuit secondCircuit).

6) Стоимость билета за проезд на данном маршруте(int price).

1. Класс Доход (Income) – форма отчета о доходе маршрута. Содержит атрибуты:

1) Идентификатор отчета о доходе (int idIncome);

2) Идентификатор маршрута (int idRoute) какому маршруту принадлежит доход;

3) Количество билетов (int tickets) – сколько билетов было куплено;

4) Дата отчета (date time) – время создания отчета, необходим для контроля прибыли за промежутки времени;

5) Величина прибыли (int sum) – сколько денег заработал маршрут за продажу билетов;

Диаграмма последовательностей.

**Расширенное описание прецедента «Составить отчет о доходах»**

* Название: «**Составить отчет о доходах**»
* Предусловие: Пользователь авторизован в системе как сотрудник автотранспортного предприятия и находится на странице с маршрутами
* Действующее лицо: Сотрудник
* Основной поток: Составление отчета. Сотрудник нажимает кнопку составить отчет о доходе, после чего он попадает на форму создания отчета. Система предлагает сотруднику ввести количество билетов и номер маршрута, на который были куплены билеты. После чего сотрудник нажимает кнопку «Готово» и система заносит данные в базу данных.
* Альтернативный поток: Сотрудник не может составлять отчеты о маршруте, к которому он не принадлежит. В случае попытки составить отчет не о своем маршруте, сотрудник будет оповещен системой о невозможности внести данные по причине недостатка прав.
* Постусловие: Система обрабатывает данные и вносит изменения в таблицу маршрутов. Администратор видит обновленную информацию на странице доходов маршрутов.

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 «Составить отчет о доходах» приведена на рисунке 17.



Рисунок17. Диаграмма последовательности для прецедента «Составить отчет о доходах»

 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 завимодействия

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.