# ВВЕДЕНИЕ

На текущий момент для применения в жизни человека существует большое количество разнообразных веб-сайтов. Веб-сайты, упрощая деятельность человека помогают ему. Примером вышеописанных веб-сайтов является информационный портал организации.

Корпоративные информационные порталы — это приложения, которые позволяют компаниям раскрывать информацию, хранящуюся внутри и вне организации, и предоставить каждому пользователю единую точку доступа к предназначенной для него информации, необходимой для принятия обоснованных управленческих решений [1].

В большинстве случаев корпоративный портал полезен достаточно большим компаниям, когда указания руководства проходят несколько инстанций, пока доберутся до адресата. Системы корпоративных порталов просты в конфигурировании и настройке. Их легко адаптировать и сделать оригинальный пользовательский интерфейс для любого звена пользователей [1].

Темой курсовой работы является разработка информационного портала интернет-провайдера.

Актуальность данной темы заключается в том, что применение информационных порталов позволяет повысить скорость и эффективность поиска необходимой информации, а также обеспечить необходимый формат представления полученных результатов.

Целью курсовой работы является применение на практике знаний, полученных в процессе изучения курса «Современные технологии интернет-программирования».

Для достижения поставленной цели необходимо:

* описать предметную область;
* выбрать инструментальные и программные средства разработки информационного портала;
* осуществить программную реализацию информационного портала.

# ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Предметная область — сфера деятельности человека, выделенная и описанная в соответствии с некоторыми целями. Описание предметной области включает:

* предметы (объекты, процессы, явления);
* отношения между выделенными предметами и/или их частями;
* взаимодействия между предметами, их частями и отношениями, возникшие в результате осуществления деятельности человека [2].

Предметной областью курсовой работы является деятельность интернет-провайдера. Всемирная сеть Интернет образована совокупностью сетей операторов и провайдеров фиксированной и мобильной связи. Провайдеры предоставляют доступ в Интернет отдельным пользователям, пользователям, объединенным в локальные сети, а также сетям крупных предприятий [3].

# Классификация веб-ресурсов

Графическое изображение общей классификации представлено на рисунке 1.



Рисунок 1 – общая классификация веб-ресурсов

Согласно общей классификации, существуют следующие виды веб-ресурсов:

* Сайт-визитка. Небольшой по количеству страниц (5–10) сайт, дающий общую информацию о деятельности компании, производимых продуктах и услугах (иногда и их стоимости), а также контактную информацию. Как правило, очень простой по исполнению [4];
* Промосайт. Также небольшой по количеству страниц. Создается в поддержку проводимых рекламных и промоакций, может выступать в качестве самостоятельного продающего инструмента. Посвящен одному продукту, часто содержит интерактивные или игровые элементы [4];
* Корпоративный сайт. Интернет-ресурс, создаваемый с целью обеспечить полноценную двустороннюю коммуникацию базисного субъекта с целевыми группами общественности. Содержит информацию о сфере деятельности компании, о ее структуре, производимых продуктах и тому подобное [4];
* Интернет-магазин. Сайт, содержащий информацию о различных видах товаров или услуг, позволяющий осуществлять их выбор и покупку непосредственно на сайте [4];
* Продающий сайт. Сайт, содержащий полную и всестороннюю информацию о каком-либо продукте или услуге, производимой организацией. Отличие от интернет-магазина заключается в том, что покупку невозможно совершить непосредственно на сайте [4];
* Веб-портал – это крупномасштабный электронный информационный ресурс по какой-либо теме или отрасли, содержащий большой объем информации, предоставляющий различные сервисы и разделенный на отдельные части, предназначенные для разных категорий пользователей; отличается наличием развитых средств навигации и поиска, обеспечивающих различные виды услуг для пользователей [5].

Графическое изображение классификации по назначению представлено на рисунке 2.



Рисунок 2 – классификация веб-ресурсов по целевому назначению

По целевому назначению различают следующие веб-ресурсы:

* Государственный сайт – это сайт, принадлежащий органам государственной власти, государственным учреждениям и организациям, отражающий их деятельность и предоставляющий гражданам различные информационные услуги [5];
* Справочные сайты – сайты, предназначенные для наведения справок или выдачи сведений, содержащие краткие описания и подтверждения тех или иных фактов и событий либо данные о ком-либо или о чем-либо [5];
* Новостные сайты – это сайты, содержащие оперативные информационные сообщения, которые представляют политически, социальный или экономический интерес для аудитории своей новизной, то есть сообщения о событиях в городе, стране, во всем мире, произошедших недавно или происходящих в данный момент [5];
* Образовательный (учебный) сайт – это сайт, содержащий сведения научного или практического характера, предназначенные для обучения [5];
* Сайты – электронные библиотеки – это сайты, предоставляющие доступ к электронным документам литературно-художественного, научного, научно-популярного, справочного характера через сеть Интернет [5];
* Профессиональный сайт – это сайт, содержащий сведения об определенной отрасли деятельности и предназначенный для специалистов [5];
* Научно-популярные сайты – это сайты, содержащие сведения о науке и ученых, технике и технологиях, культуре и искусству, изложенные простым и понятным языком, доступным пользователю-неспециалисту [5];
* Сайты электронной коммерции – это сайты, направленные на реализацию товаров и услуг с использованием информационных технологий, позволяющие совершать покупки или проводить денежные операции онлайн [5];
* Развлекательные сайты и сайты для досуга – это сайты, направленные на развлечение, а также содержащие информацию о том, как провести свое свободное время, заняться творчеством, любимым делом или хобби [5];
* Социальные сайты – это сайты, предназначенные для общения пользователей в сети Интернет [5];
* Сайты-хостинги – это веб-сайты, предоставляющие услуги по размещению в Интернете информации пользователя в любом ее виде: текстовом, изобразительном, видео, аудио и других видах [5].

Разрабатываемый веб-ресурс принадлежит к категории корпоративных сайтов исходя из предметной области данной курсовой работы.

# Анализ целевой аудитории информационного портала

Целевая аудитория представляет собой совокупность конкретных лиц, на которых направлено воздействие. Главным критерием включения отдельных лиц в целевую аудиторию является их интерес к предмету продвижения [6].

Целевая аудитория веб-ресурса — это пользователи, заинтересованные в функционировании сайта или пользователи, преследующие свои цели при работе с сайтом.

Целевой аудиторией информационного портала являются текущие и потенциальные клиенты интернет-провайдера.

Возраст целевой аудитории информационного портала не имеет жёстких ограничений и начинается с 14 лет, когда человек получает законную возможность заключать договор с интернет-провайдером.

Целевая аудитория информационного портала географически вся находится на территории Российской Федерации, что продиктовано возможностью компании оказывать услуги только на территории указанной страны. Исходя из этого информационный портал будет обслуживать пользователей только на русском языке.

У разрабатываемого информационного портала нет строгой принадлежности целевой аудитории к определённому полу. Среди пользователей ресурсов подобных разрабатываемому одинаково распространены оба пола.

Материальный достаток целевой аудитории разнороден в виду ориентированности информационного портала на большое количество слоёв населения. Соответственно этому и услуги, которые предоставляет провайдер ориентированы на клиентов с разным материальным достатком.

Таким образом, была кратко охарактеризована аудитория информационного портала исходя из возраста, географического расположения, принадлежности к полу и материальному достатку.

# Техническое задание

Техническое задание – это основной документ, оформляемый на предпроектной стадии разработки информационного ресурса. Техническое задание отражает требования, предъявляемые заказчиками к разрабатываемой системе [7].

Техническое задание представляет собой исходный материал для создания информационного портала или другого информационного продукта. Поэтому техническое задание в первую очередь должно содержать основные технические требования к продукту, а также отвечать на вопрос, что данная система должна делать, как работать и при каких условиях [7].

Техническое задание должно подробно описывать все функциональные и пользовательские требования к системе. Чем подробнее будет составлено техническое задание, тем меньше спорных ситуаций возникнет между заказчиком и разработчиком во время приемочных испытаний [7].

Конкретное содержание технического задания определяют заказчик и разработчик, а при инициативной разработке – разработчик. Не допускается включать в техническое задание требования, которые противоречат требованиям стандартов и нормативных документов органов, осуществляющих надзор за безопасностью, охраной здоровья и природы [8].

В техническое задание могут быть внесены изменения и дополнения, оформляемые заказчиком и разработчиком в виде протокола разногласий. Те пункты технического задания, которые в конкретном случае не оговариваются, могут опускаться [8].

Таким образом, техническое задание является документом, который позволяет как разработчику, заказчику представить конечный продукт и выполнить проверку на соответствие предъявленным требованиям [7].

Техническое задание для разрабатываемого информационного портала представлено в приложении А.

# Выбор инструментальных и программных средств разработки информационного портала

# Выбор языка и технологии программирования

Языком программирования был выбран java учитывая имеющийся опыт работы с этим языком и реализацией серверной стороны сайта при помощи сервлетов. Java – это объектно-ориентированный язык программирования (ЯП), которым можно управлять через виртуальную машину либо с помощью браузеров [9]. Сервлеты – это маленькие программы, которые выполняются на серверной стороне в контейнере. Являясь приложениями на Java, они импортируют различные пакеты библиотеки Java, могут писать на жесткий диск, обмениваться данными с базами данных [10].

Для страниц, которые будут содержать статическую информацию будет использована связка из HyperText Markup Language и Cascading Style Sheets. HTML – это язык гипертекстовой разметки, который получил очень широкое распространение в сети Интернет [11]. CSS — это набор параметров форматирования, который применяется к элементам документа, чтобы изменить их внешний вид [12].

Для страниц, которые будут использоваться для постоянного ввода и вывода информации будет использована технология Java Server Pages, а также css. Java Server Pages – это технология, позволяющая создавать динамические веб-страницы, обрабатываемые на сервере. JavaServer Pages – это текстовые файлы с обычным HTML-кодом, в которые могут быть вставлены фрагменты Java-кода с помощью специальных тегов JSP [10].

Для реализации капчи на страницах авторизации и регистрации также будет использован язык программирования JavaScript. JavaScript — это объектно-ориентированный язык, предназначенный для создания приложений в Интернете, работающих как на стороне клиента, так и на стороне сервера [13].

Подводя итог можно сказать, что все необходимые языки программирования и технологии для реализации информационного портала были выбраны.

# Выбор системы управления базами данных

Разрабатываемый информационный портал требует хранения данных о текущих тарифах, администраторах, пользователях, вопросов для обратной связи, а также ответов на них и заявок на подключение интернета. Для хранения такого массива данных было решено использовать базу данных, а также систему управления базами данных для удобства обращения с хранимыми данными.

База данных — это информационная модель, позволяющая упорядоченно хранить данные о группе объектов, обладающих одинаковым набором свойств. Иначе говоря, база данных (БД) — это именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области [2].

Система управления базами данных (СУБД) – это специальная программа, необходимая для организации базы данных (хранилища информации) и работы с ней пользователей информационной системы [14].

В ходе выбора было рассмотрено несколько систем управления базами данных: PostgreSQL, MySQL, SQLite.

PostgreSQL — это объектно-реляционная система управления базами данных), одна из наиболее развитых систем управления базами данных в мире. Имеет открытый исходный код и является альтернативой коммерческим базам данных [15]. Преимущество PostgreSQL – это эффективность его центрального алгоритма, что означает, что он превосходит многие базы данных, которые рекламируются как более продвинутые. Это особенно полезно, если происходит работа с большим количеством данных, для которых процессы ввода-вывода могут стать слабым местом. но для небольших баз данных доступны более быстрые инструменты [16].

MySQL — это реляционная система управления базами данных которая является свободным программным обеспечением. Считается одной из наиболее популярных, в виду своей легкости, гибкости, удобства в использовании [17].

SQLite — это лёгкая и быстрая встраиваемая однофайловая система управления базами данных, которая не имеет сервера и позволяет хранить всю базу данных на одном устройстве локально [18]. Благодаря особенностям архитектуры SQLite работает быстро, особенно на чтение. Компоненты СУБД встроены в приложение и вызываются в том же процессе. Поэтому доступ к ним быстрее, чем при взаимодействии между разными процессами [18].

База данных состоит из табличных записей, связей между ними, индексов и других компонентов. В SQLite они хранятся в едином файле, который находится на том же устройстве, что и программа. Из всех возможностей SQL в ней есть наиболее нужные. Поэтому SQLite отличают малый размер, простота решений и легкость администрирования [18].

SQLite отличается высокой степенью надежности. Программный код этой СУБД на сто процентов покрыт тестами. Это означает, что проверен каждый компонент программы. Применение такой СУБД несёт минимальные риски непредсказуемого поведения [18].

SQLite находится в свободном публичном доступе. На использование этой СУБД нет никаких правовых ограничений, а владельцем считается всё общество. Можно без ограничений смотреть и менять под свои цели весь исходный программный код программы [18]. Кроме того, вышеупомянутая СУБД является кроссплатформенной, а значит для UNIX-подобных систем, MacOS и Windows [18].

SQLite полностью независим от любого стороннего программного обеспечения, фреймворков или библиотек. Для работы с базой данных дополнительные компоненты или интернет не обязателен [18].

Таким образом исходя из описанных преимуществ, а также имеющегося опыта работы была выбрана СУБД SQLite.

# Выбор веб-сервера

Веб-сервер представляет собой программу, работающую в операционной системе, которая принимает входящие запросы, формирует ответ и возвращает результат [19].

Был рассмотрено несколько популярных веб-серверов: Nginx, Internet Information Services, Apache Tomcat.

Nginx — второй по популярности веб-сервер после Apache. У этого веб-сервера асинхронная событийно-ориентированная архитектура, которая позволяет добиваться быстрого масштабирования даже при минимальных ресурсах. Хорошо подходит для веб-проектов с высокого нагрузкой. Но при этом веб-сервер не может самостоятельно работать с динамическим контентом [20].

Internet Information Services (IIS) — набор сервисов для создания веб-сервера от компании Microsoft. Распространяется вместе с операционными системами Windows как опционально инсталлируемый компонент. Главное преимущество IIS – в поддержке продуктов Microsoft и глубокой интеграции с ними. Основной недостаток — это жесткая привязка к операционной системе Windows и отсутствие поддержки для других операционных систем [20].

Apache Tomcat — это контейнер сервлетов, который обрабатывает спецификации Java. Например, Java Server Pages (JSP), Java EL Java Servlet и WebSocket. Tomcat может использоваться как самостоятельный веб-сервер, так и вместе с Apache HTTP Server. Веб-сервер написан на языке Java и способен автоматически подгружать Java-библиотеки. Главная направленность этого веб-сервера — веб-проекты с динамическим содержимым [20].

Веб-сервером для разработки информационного портала был выбран Apache Tomcat. В виду того, что этот веб-сервер был изначально разработан для того, чтобы взаимодействовать с содержимым, которое создаётся Java приложениями, а также исходя из наличия опыта работы именно с этим веб-сервером.

# ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОГО ПОРТАЛА

# Структура информационного портала

Структура информационного портала состоит из ряда страниц. Часть страниц будет доступна всем, другая часть будет доступна только зарегистрированным пользователям, а некоторые страницы будут доступны только администратору.

Графическое изображение структуры информационного портала представлено на рисунке 3.

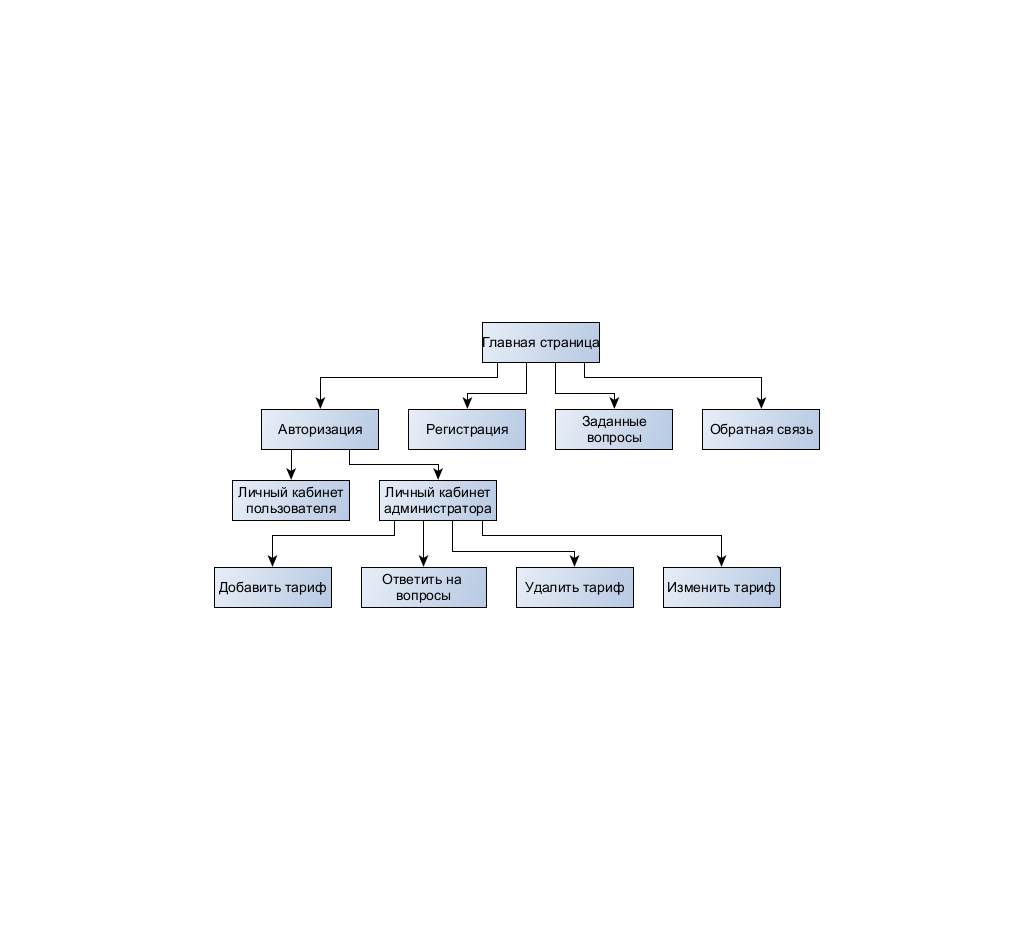


Рисунок 3 – структура информационного портала

# Разработка дизайн-макета информационного портала

Общую структуру макета можно разделить на 3 части: header, content, footer. Header будет содержать блок ссылок для перехода на другие страницы и заголовок веб-сайта. Content будет содержать в себе материал который необходимо представить пользователю иначе говоря главное содержимое страницы. Footer будет содержать в себе знак копирайта, а также email по которому можно будет связаться с автором информационного портала.

Графическое изображение дизайн-макета информационного портала представлено на рисунке 4.

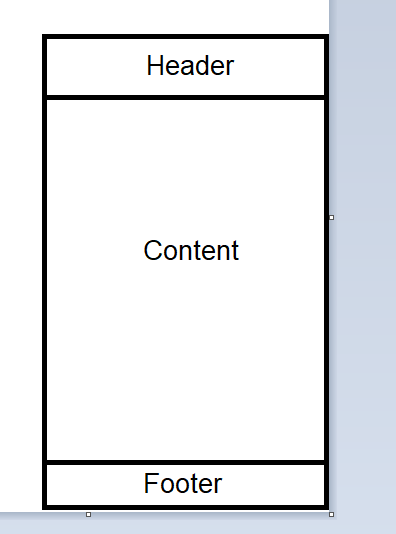


Рисунок 4 – дизайн-макет информационного портала

Для информационного портала был выбран шрифт Roboto. Пункты меню, которые хочет выбрать пользователь будут подсвечиваться зелёным цветом.

# Разработка модулей информационного портала

Модуль регистрации. Пользователь, попав на страницу регистрации должен ввести необходимые данные в поля формы. Пароли будут проверены на совпадение, на наличие одной цифры, на наличие одного спецсимвола, на наличие одной заглавной буквы при помощи скрипта написанного на JavaScript.

Также пользователь должен ввести капчу которая будет также проверена при помощи JavaScript скрипта. В случае прохождения всех проверок данные будут отправлены при помощи POST-метода.

Код проверяющий капчу и пароли представлен ниже.

function getRandomInt(min, max) {

min = Math.ceil(min);

max = Math.floor(max);

return Math.floor(Math.random() \* (max - min)) + min;

}

window.onload = function() {

let number1 = getRandomInt(1, 10);

let number2 = getRandomInt(1, 10);

document.getElementById("captcha").value = number1 + "+" + number2;

document.getElementById("first").onsubmit = function()

{

return isValidForm();

};

function isValidForm() {

if (document.getElementById("captchaInput").value == number1+ number2)

{

if(document.getElementById("pass").value==document.getElementById("pass2").value)

{

if(document.getElementById("pass").value.match(/[A-Z]/)) {if(document.getElementById("pass").value.match(/\W/)) {if(document.getElementById("pass").value.match(/\d/))

{return true;

} else {

alert("Нет ни одной цифры в пароле");

return false;

}

} else {

alert("Нет спецсимвола в пароле");

return false;

}

} else {

alert("Нет заглавной буквы в пароле");

return false;

}

} else {

alert("Пароли не совпали");

return false;

}

} else {

alert("Посчитайте и введите капчу заново");

return false;

}

}

}

Данные отправляются на сервлет RegUser.

Код сервлета представлен ниже.

@WebServlet("/RegUser")

**public** **class** RegUser **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

getServletContext().getRequestDispatcher("/Registration.html").forward(request, response);

}

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

request.setCharacterEncoding("UTF-8");

**try** {

String surname = request.getParameter("surname");

String name = request.getParameter("name");

String adres = request.getParameter("adres");

String telnumb = request.getParameter("telnumb");

String email = request.getParameter("email");

String password = request.getParameter("pass");

users users = **new** users("0",surname,name,adres,email,telnumb,password,"user");

**int** rows = krdb.*insert*(users);

**if**(rows!=0)

{

String role = "user";

request.getSession().setAttribute("email",email );

request.getSession().setAttribute("password",password );

request.getSession().setAttribute("role",role);

Cookie[] cookies = request.getCookies();

String cookieName = "name";

Cookie cookie = **null**;

**boolean** cookieChanged=**false**;

**if**(cookies !=**null**) {

**for**(Cookie c: cookies) {

**if**(cookieName.equals(c.getName())) {

c.setValue(name);

c.setMaxAge(24\*60\*60);

cookieChanged=**true**;

response.addCookie(c);

}

}

}

**if**(cookieChanged==**false**)

{

Cookie cookie2 = **new** Cookie("name",users.getName());

cookie2.setMaxAge(24\*60\*60);

response.addCookie(cookie2);

}

**if** (role.equals("user"))

{ response.sendRedirect(request.getContextPath()+"/UserAccount.jsp");

}

**else** **if**(role.equals("admin"))

{ response.sendRedirect(request.getContextPath()+"/AdminAccount.jsp");

}

}

**else**

{

response.sendRedirect(request.getContextPath()+"/Registration.html");

}

}

**catch**(Exception ex) {

}

}

}

Данные принимаются из POST запроса и записываются в переменные. Далее создаётся экземпляр класса user который содержи в себе все необходимые поля и после этого передаётся в метод krdb.insert(). Этот метод в свою очередь пытается записать этого пользователя в базу данных при помощи запроса insert и возвращает значение отличное от нуля в случае удачи.

Если запись была успешной, то также создаётся сессия и куки и пользователь переадресовывается в тот личный кабинет, который подходит ему согласно роли.

Класс krdb содержит в себе методы для подключения и взаимодействия с базой данных.

Код метода insert представлен ниже.

**public** **static** **int** insert(users users) {

**try** { Class.*forName*("org.sqlite.JDBC").getDeclaredConstructor().newInstance();

**try** (Connection conn = DriverManager.*getConnection*(*url1*))

{

String sql = "INSERT INTO users (Surname, Name,Adres,Email,Telnumb,Password,Role) Values (?,?,?,?,?,?,?)";

**int** rows=0;

**try** (PreparedStatement preparedStatement = conn.prepareStatement(sql)) {

preparedStatement.setString(1,users.getSurname());

preparedStatement.setString(2, users.getName());

preparedStatement.setString(3, users.getAdres());

preparedStatement.setString(4, users.getEmail());

preparedStatement.setString(5, users.getTelnumb());

preparedStatement.setString(6, users.getPassword());

preparedStatement.setString(7, "user");

rows = preparedStatement.executeUpdate();

}

conn.close();

**return** rows;

}

} **catch** (Exception ex) {

System.***out***.println(ex);

**return** 0;

}

}

Поля регистрации пользователя представлены на рисунке 5.

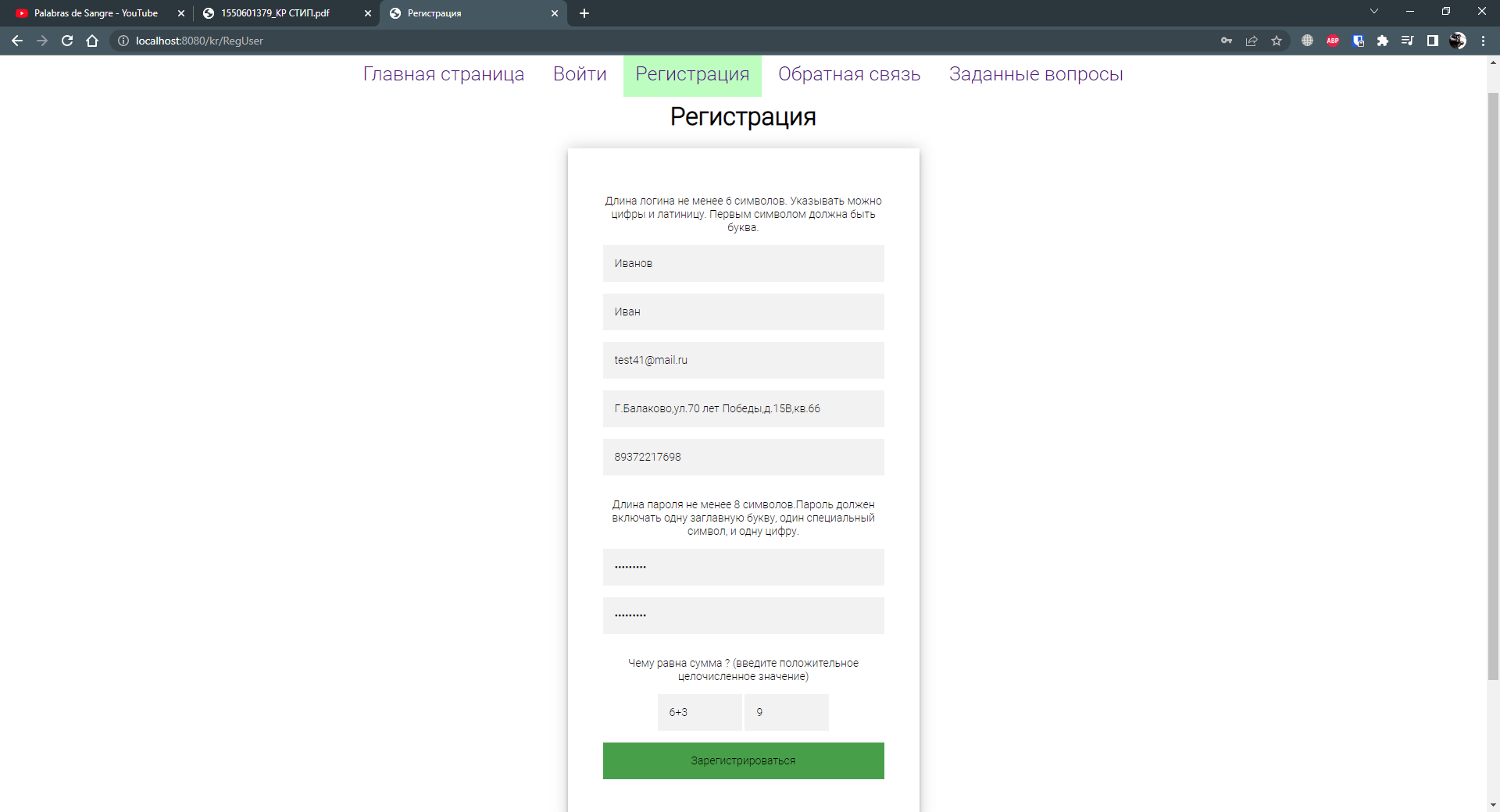


Рисунок 5 – поля регистрации пользователя

Модуль авторизации. Пользователь, перейдя по ссылке на страницу авторизации попадает в сервлет Auth. В get методе сервлета происходит проверка сессии и хранимых куки пользователя. В случае если необходимые данные присутствуют, то пользователь, минуя страницу авторизации сразу попадает в личный кабинет. Если данные отсутствуют, то пользователь перенаправляется на страницу авторизации, где ему нужно будет вводить соответствующие данные.

Код сервлета auth представлен ниже.

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

HttpSession session = request.getSession(); **if**((session.getAttribute("email")!=**null**&&(session.getAttribute("password")!=**null**)) && (session.getAttribute("role")!=**null**))

{

users user = **new** users();

user = krdb.*getUser*(session.getAttribute("email").toString(), session.getAttribute("password").toString());

Cookie[] cookies = request.getCookies();

String cookieName = "name";

**boolean** cookieChanged=**false**;

**if**(cookies !=**null**) {

**for**(Cookie c: cookies) {

**if**(cookieName.equals(c.getName())) {

c.setValue(user.getName());

c.setMaxAge(24\*60\*60);

cookieChanged=**true**;

response.addCookie(c);

}

}

}

**if**(cookieChanged==**false**)

{

Cookie cookie2 = **new** Cookie("name",user.getName());

cookie2.setMaxAge(24\*60\*60);

response.addCookie(cookie2);

}

**if** (user.getRole().equals("user"))

{ response.sendRedirect(request.getContextPath()+"/UserAccount.jsp");

}

**else** **if**(user.getRole().equals("admin"))

{ response.sendRedirect(request.getContextPath()+"/AdminAccount.jsp");

}

}

**else**

{ response.sendRedirect(request.getContextPath()+"/Authorization.html");

}

}

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

request.setCharacterEncoding("UTF-8");

String email = request.getParameter("userEmail");

String password = request.getParameter("userPassword");

users user = **new** users();

user = krdb.*getUser*(email, password);

**if**(user!=**null**)

{

request.getSession().setAttribute("email",email );

request.getSession().setAttribute("password",password);

request.getSession().setAttribute("role",user.getRole());

Cookie[] cookies = request.getCookies();

String cookieName = "name";

**boolean** cookieChanged=**false**;

**if**(cookies !=**null**) {

**for**(Cookie c: cookies) {

**if**(cookieName.equals(c.getName())) {

c.setValue(user.getName());

c.setMaxAge(24\*60\*60);

cookieChanged=**true**;

response.addCookie(c);

}

}

}

**if**(cookieChanged==**false**)

{

Cookie cookie2 = **new** Cookie("name",user.getName());

cookie2.setMaxAge(24\*60\*60);

response.addCookie(cookie2);

}

**if** (user.getRole().equals("user"))

{

response.sendRedirect(request.getContextPath()+"/UserAccount.jsp");

}

**else** **if**(user.getRole().equals("admin"))

{

response.sendRedirect(request.getContextPath()+"/AdminAccount.jsp");

}

}

**else**

{ response.sendRedirect(request.getContextPath()+"/Authorization.html");

}

}

}

Страница авторизации представлена на рисунке 6.

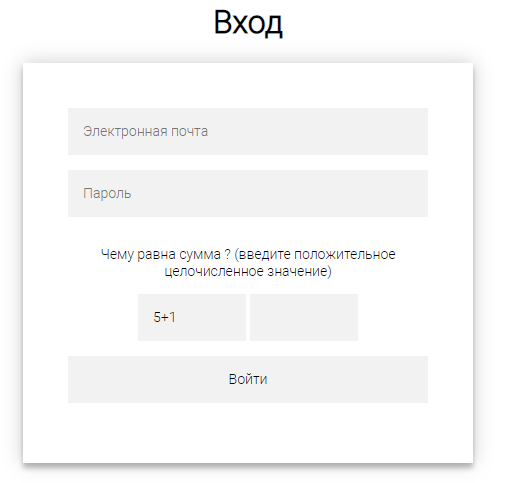


Рисунок 6 – страница авторизации

Дабы избежать несанкционированного перехода на соответствующие страницы, требующие определённых прав, при помощи прямого обращения к странице через адресную строку, была использована проверка. Код встроен во все необходимые jsp страницы.

Код проверки представлен ниже.

<c:if test="${empty email || empty password || role!='user'}">

<c:redirect url="/Auth" />

</c:if>

Происходит проверка существования необходимых данных, записанных при помощи сессии, а также соответствия роли пользователя с уровнем доступа, требуемым для текущей страницы.

Модуль поиска тарифов. Пользователь может искать у себя в личном кабинете существующие тарифы чтобы оформить заявку на подключение интернета.

Данные, вводимые пользователем, принимает сервлет UserSearch.

Код сервлета UserSearch.

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

request.setCharacterEncoding("UTF-8");

String name = request.getParameter("name");

request.setAttribute("TarifList", krdb.tarifSearch(name)); getServletContext().getRequestDispatcher("/UserAccount.jsp").forward(request, response);}

Введённое пользователем название передаётся методу tarifSearch который при помощи SQL оператора LIKE ищет тарифы со схожим названием, а далее формируется объект list куда записываются все подходящие тарифы при помощи добавления в list экземпляра класса tarif который содержит в себе все поля тарифа.

Сформированный таким образом list возвращается в метод doPost() и добавляется в контекст запроса для последующего считывания в jsp файле отвечающем за личный кабинет.

Код метода tarifSearch представлен ниже.

public static List tarifSearch(String name) {

try {

Class.forName("org.sqlite.JDBC").getDeclaredConstructor().newInstance();

try (Connection conn = DriverManager.getConnection(url1)) {

Statement statement = conn.createStatement();

ResultSet resultSet = statement.executeQuery("SELECT \* FROM tarif where name like \"%" + name + "%\"");

List TarifList = new LinkedList();

while (resultSet.next()) {

tarif tf = new tarif();

tf.setID(resultSet.getString("id"));

tf.setName(resultSet.getString("name"));

tf.setPrice(resultSet.getString("speed"));

tf.setSpeed(resultSet.getString("price"));

TarifList.add(tf);

}

conn.close();

return TarifList;

}

} catch (Exception ex) {

System.out.println(ex);

}

return null;

}

Переданный list считывается на jsp странице и в цикле каждую итерацию выводится на экран для пользователя, а также вывод оборачивается в теги input и form для того, чтобы сформировать из всех подходящих тарифов набор форм с помощью, которых можно сразу отправить заявку.

Код вывода тарифов представлен ниже.

<c:forEach items="${TarifList}" var="cell">

<div class="form2">

<form action="/kr/AddUserTarif" method="POST" class="first2" id="first2">

<input type="hidden" name="userEmail" readonly value="${email}" />

<input type="hidden" name="userPassword" readonly value=<c:out value="${password}"/>>

<input type="hidden" name="id" placeholder="ID тарифа" readonly value=<c:out value="${cell.ID}"/>>

<p>Название тарифа</p>

<input type="text" name="name" placeholder="Название нового тарифа" readonly value=<c:out value="${cell.name}" />>

<p>Скорость в мегабитах</p>

<input type="number" name="speed" placeholder="Скорость нового тарифа в мегабитах" readonly min="1" value=<c:out value="${cell.speed}"/>>

<p>Цена тарифа</p>

<input type="number" name="price" placeholder="Цена нового тарифа в рублях" readonly min="1" value=<c:out value="${cell.price}"/>>

<input type="submit" value="отправить заявку" /></form>

</div>

</c:forEach>

Поиск тарифа представлен на рисунке 7.

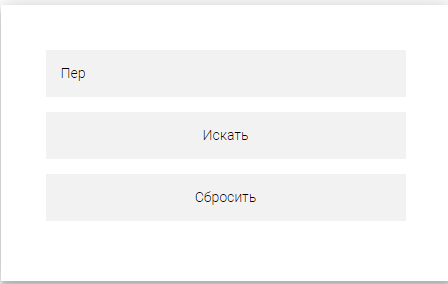


Рисунок 7 – поиск тарифов

Результат поиска представлен на рисунке 8.

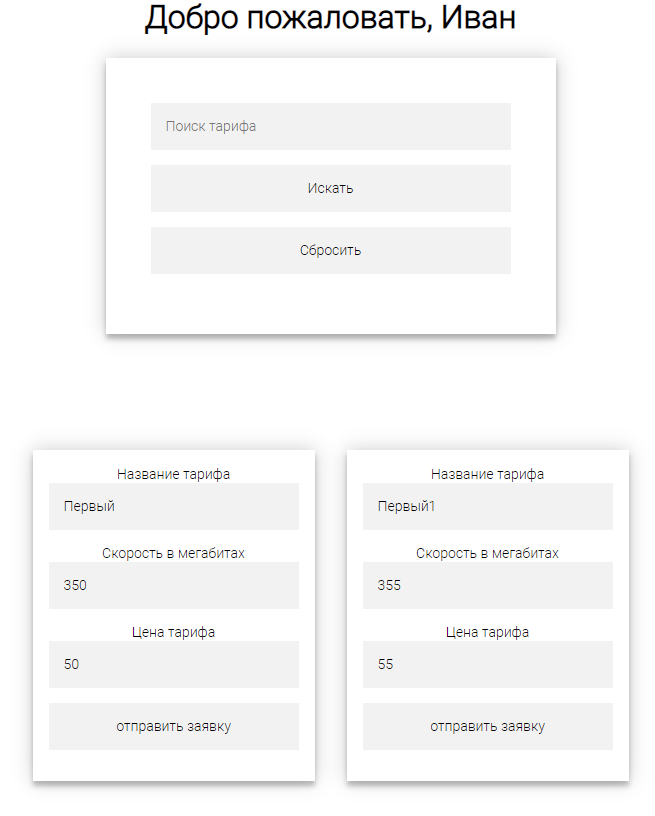


Рисунок 8 – результат поиска

Модуль отправки заявки. Для того чтобы отправить заявку пользователь нажимает на соответствующую кнопку после поиска тарифов. После чего все посланные данные считывает сервлет AddUserTarif. И добавляет их в базу данных обращаясь к методу insertUserTarif передавая полученный id тарифа и пользователя, а id пользователя получает из экземпляра класса user который в свою очередь получает из метода getUser передавая email и password пользователя.

Фрагмент кода сервлета AddUserTarif.

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

request.setCharacterEncoding("UTF-8");

String idTarif = request.getParameter("id");

String userEmail = request.getParameter("userEmail");

String userPassword = request.getParameter("userPassword");

krdb.insertUserTarif(idTarif,krdb.getUser(userEmail, userPassword).getID());

response.sendRedirect(request.getContextPath()+"/UserAccount.jsp");

}

Модуль выхода из личного кабинета.

Модуль выхода из личного кабинета. В случае если пользователю необходимо выйти из личного кабинета, то он может нажать на кнопку «Выйти».

Он будет перенаправлен на сервлет Logout который очистит данные, записанные в сессии и куки, а после этого пользователь будет перенаправлен на главную страницу.

Код сервлета logout.

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

users user = new users();

request.getSession().removeAttribute("email");

request.getSession().removeAttribute("password");

request.getSession().removeAttribute("role");

Cookie[] cookies = request.getCookies();

String cookieName = "name";

Cookie cookie = null;

if(cookies !=null) {

for(Cookie c: cookies) {

if(cookieName.equals(c.getName())) {

c.setMaxAge(0);

response.addCookie(c);

}

}

}

response.sendRedirect(request.getContextPath()+"/MainPage.html");

}

Модуль обратной связи. Попал на страницу для обращения к администрации пользователю необходимо ввести все необходимые данные, а после этого их обработает сервлет Feedback внеся вопрос в базу данных.

Фрагмент кода сервлета Feedback представлен ниже.

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

request.setCharacterEncoding("UTF-8");

String name = request.getParameter("userName");

String email = request.getParameter("userEmail");

String question = request.getParameter("userQuestion");

krdb.insertQuestion(name, email, question);

response.sendRedirect(request.getContextPath()+"/Feedback");

}

Код метода InsertQuestion для вставки в базу данных представлен ниже.

public static void insertQuestion(String name, String email, String question) {try {

Class.forName("org.sqlite.JDBC").getDeclaredConstructor().newInstance();

try (Connection conn = DriverManager.getConnection(url1)) {

String sql = "INSERT INTO feedback (name,email,question) Values (?,?,?)";

try (PreparedStatement preparedStatement = conn.prepareStatement(sql)) {

preparedStatement.setString(1, name);

preparedStatement.setString(2, email);

preparedStatement.setString(3, question);

preparedStatement.executeUpdate();

}

conn.close();

}

} catch (Exception ex) {

System.out.println(ex);

}

}

Страница с обратной связью представлена на рисунке 9.

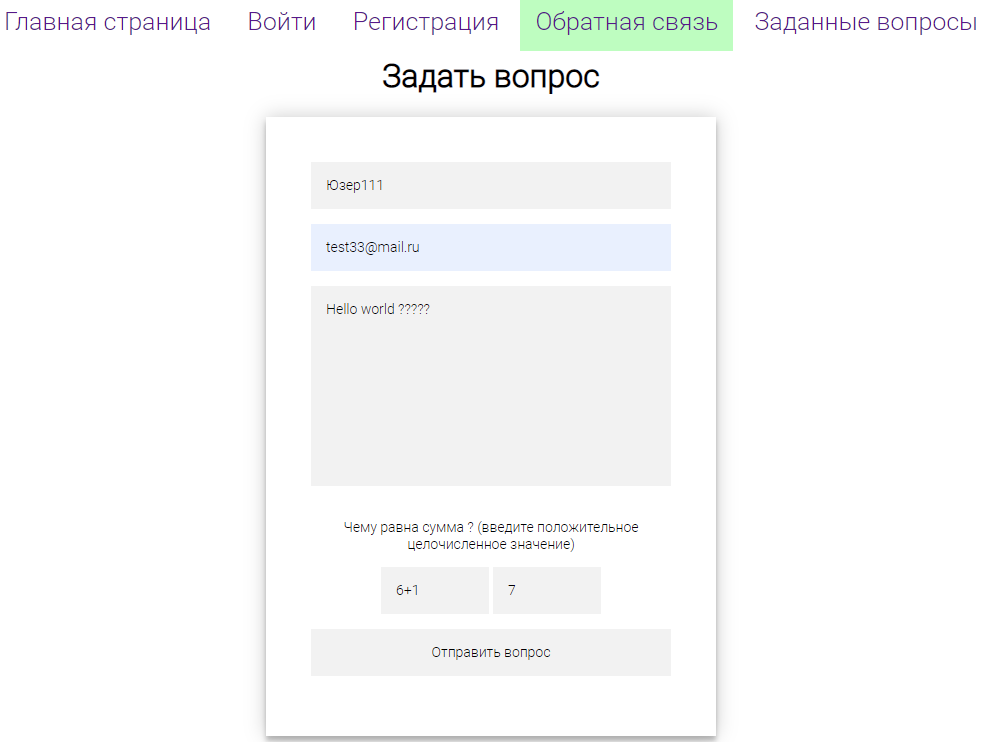


Рисунок 9 – обратная связь

Модуль просмотра обратной связи. Пользователь, зайдя на эту страницу сможет увидеть все заданные вопросы в том числе те на которые ответил администратор.

За работу этой страницы отвечает сервлет FeedbackOutput.

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

request.setAttribute("feedbackList", krdb.FeedbackOutput());

getServletContext().getRequestDispatcher("/FeedbackOutput.jsp").forward(request, response);

}

В методе FeedbackOutput данные запрашиваются из базы данных после чего возвращаются в виде list в котором содержатся объекты типы userFeedback, а после list передаётся в контекст запроса на jsp страницу, где считывается в цикле.

public static List FeedbackOutput() {

try { Class.forName("org.sqlite.JDBC").getDeclaredConstructor().newInstance();

try (Connection conn = DriverManager.getConnection(url1)) {

Statement statement = conn.createStatement();

ResultSet resultSet = statement.executeQuery("SELECT \* FROM feedback");

List feedbackList = new LinkedList();

while (resultSet.next()) {

userFeedback uf = new userFeedback();

uf.setID(resultSet.getString("id"));

uf.setName(resultSet.getString("name"));

uf.setEmail(resultSet.getString("email"));

uf.setQuestion(resultSet.getString("question"));

uf.setAnswer(resultSet.getString("answer"));

feedbackList.add(uf);

}

conn.close();

return feedbackList;

}

} catch (Exception ex) {

System.out.println(ex);

}

return null;

}

Страница с выводом обратной связи представлена на рисунке 10.

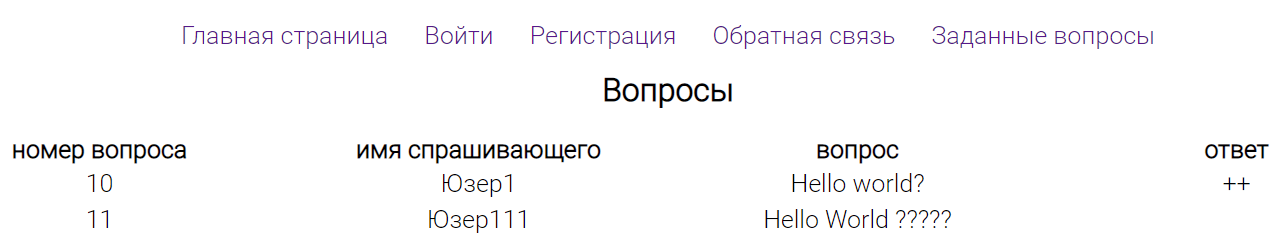


Рисунок 10 – обратная связь

Модуль ответа на вопросы. Доступен администратору.

Администратор на этой странице может ответить на необходимый вопрос введя в поля номер вопроса и ответ. За это отвечает сервлет AdminFeedback.

Код сервлета AdminFeedback представлен ниже.

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

request.setAttribute("feedbackList", krdb.FeedbackOutput());

getServletContext().getRequestDispatcher("/AdminAnswers.jsp").forward(request, response);

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

request.setCharacterEncoding("UTF-8");

String id = request.getParameter("numberQuestion");

String answer = request.getParameter("Answer");

krdb.addAnswer(id, answer);

response.sendRedirect(request.getContextPath()+"/AdminFeedback");

}

}

При POST запросе сервлет считывает данные после этого добавляет их в базу данных при помощи метода AddAnswer. При GET запросе происходит считывание всех вопросов из базы данных, обёртывание их в list, а также передача этого list в контекст запроса и последующее считывание переданного list на jsp странице точно также как это уже описывалось ранее.

Страница для ответов на вопросы представлена на рисунке 11.

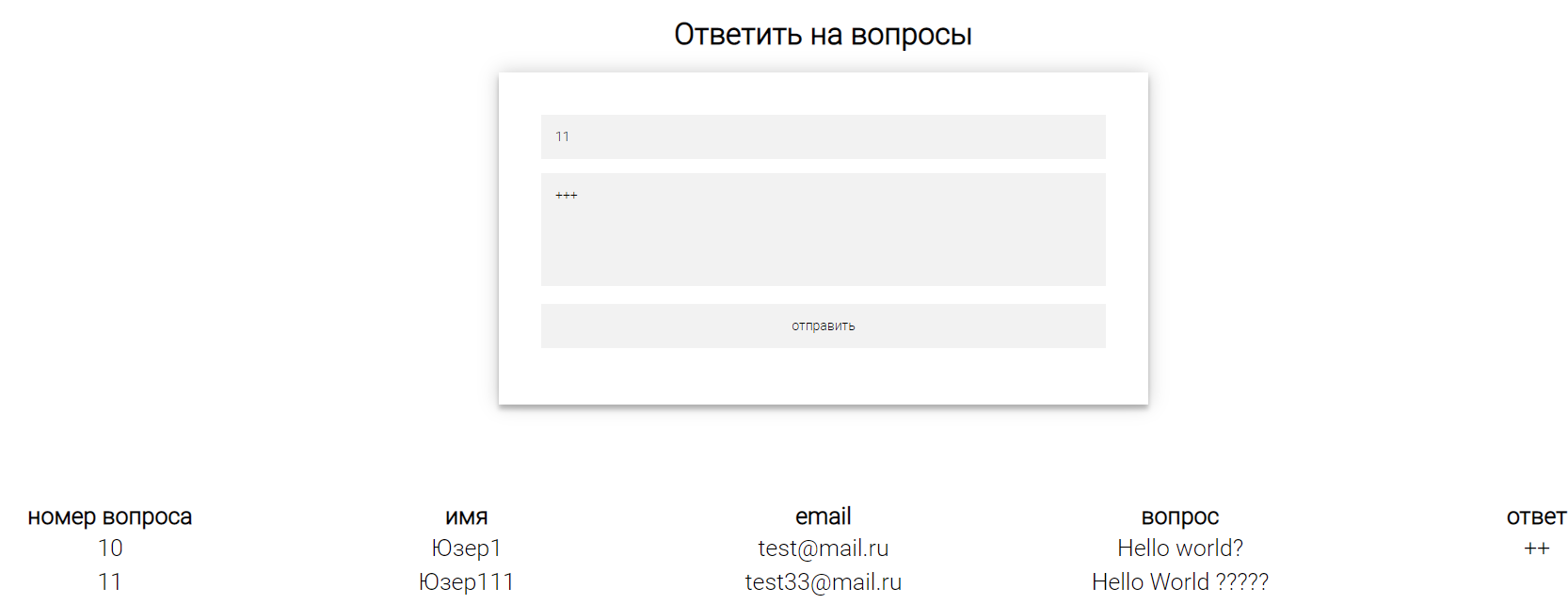


Рисунок 11 – страница для ответа на вопросы

Результат ответа на вопрос представлен на рисунке 12.

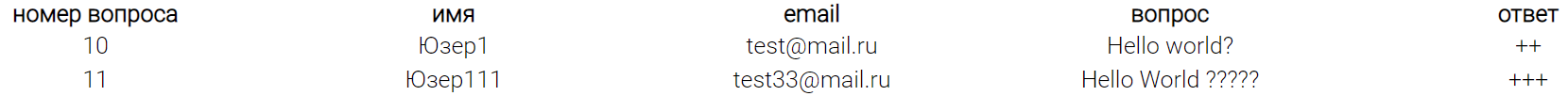


Рисунок 12 – результат ответа

Модуль добавления тарифа. На это странице администратор может добавить новый тариф. За это отвечает сервлет NewTarif.

Код сервлета NewTarif представлен ниже.

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {request.setAttribute("TarifList", krdb.TarifOutput());getServletContext().getRequestDispatcher("/NewTarif.jsp").forward(request, response);

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

request.setCharacterEncoding("UTF-8");

String name = request.getParameter("name");

String speed = request.getParameter("speed");

String price = request.getParameter("price");

krdb.insertTarif(name, speed, price);

response.sendRedirect(request.getContextPath()+"/NewTarif");

}

}

При POST запросе сервлет считывает данные после этого добавляет их в базу данных при помощи метода InsertTarif. При GET запросе происходит считывание всех тарифов из базы данных, обёртывание их в list, а также передача этого list в контекст запроса и последующее считывание переданного list на jsp странице точно также как это уже описывалось ранее.

Страница для добавления тарифа представлена на рисунке 13.

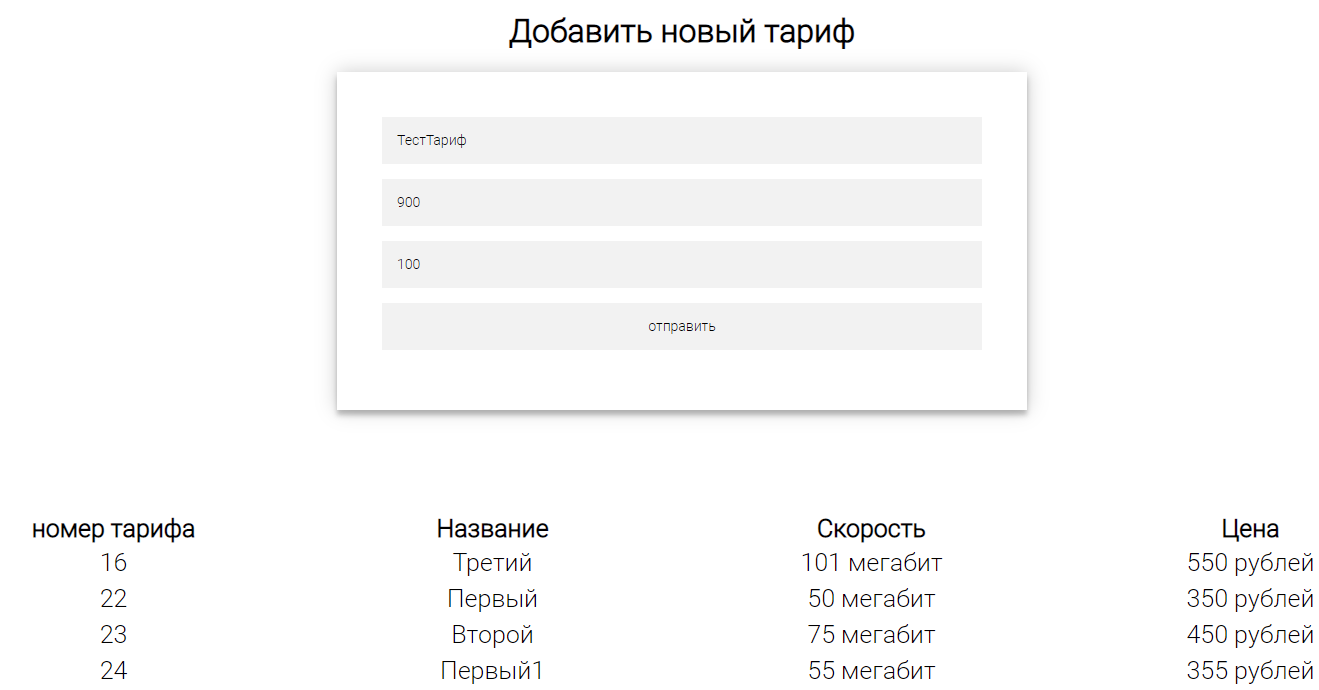


Рисунок 13 – страница для добавления тарифа

Результат добавления тарифа представлен на рисунке 14.



Рисунок 14 – результат добавления тарифа

Модуль изменения тарифа. Доступен администратору.

На этой странице администратор может изменить характеристики необходимого тарифа.

Код сервлета ChangeTarif представлен ниже.

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

request.setAttribute("TarifList", krdb.TarifOutput());

getServletContext().getRequestDispatcher("/ChangeTarif.jsp").forward(request, response);

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

request.setCharacterEncoding("UTF-8");

String id = request.getParameter("id");

String name = request.getParameter("name");

String speed = request.getParameter("speed");

String price = request.getParameter("price");

krdb.updateTarif(id,name, speed, price);

response.sendRedirect(request.getContextPath()+"/ChangeTarif");

}

При POST запросе сервлет считывает данные после этого обновляет запись в базе данных при помощи метода UpdateTarif. При GET запросе происходит считывание всех тарифов из базы данных, обёртывание их в list, а также передача этого list в контекст запроса и последующее считывание переданного list на jsp странице точно также как это уже описывалось ранее.

Страница для обновления тарифа представлена на рисунке 15.

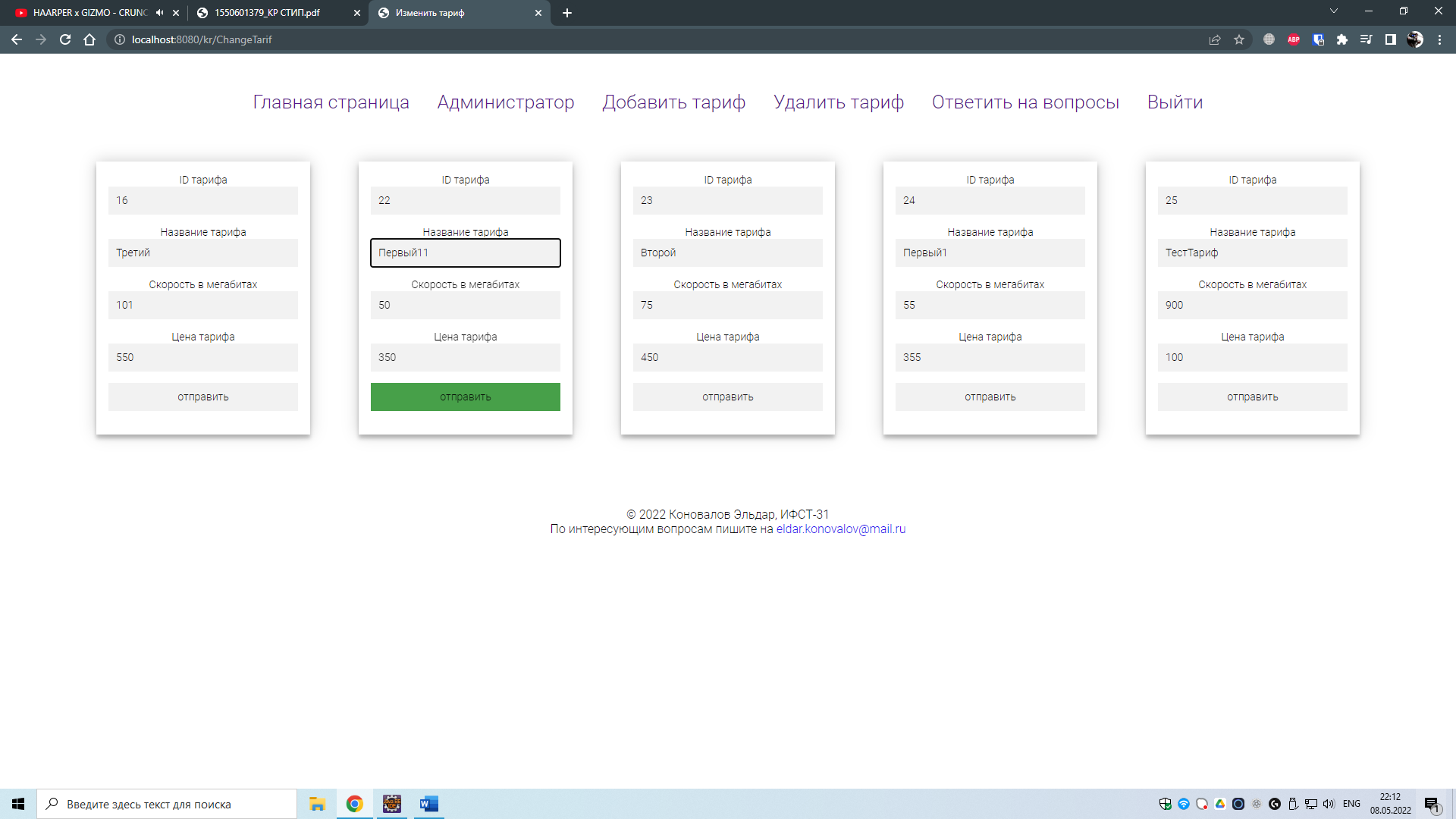


Рисунок 15 – страница для добавления тарифа

Результат обновления тарифа представлен на рисунке 16.

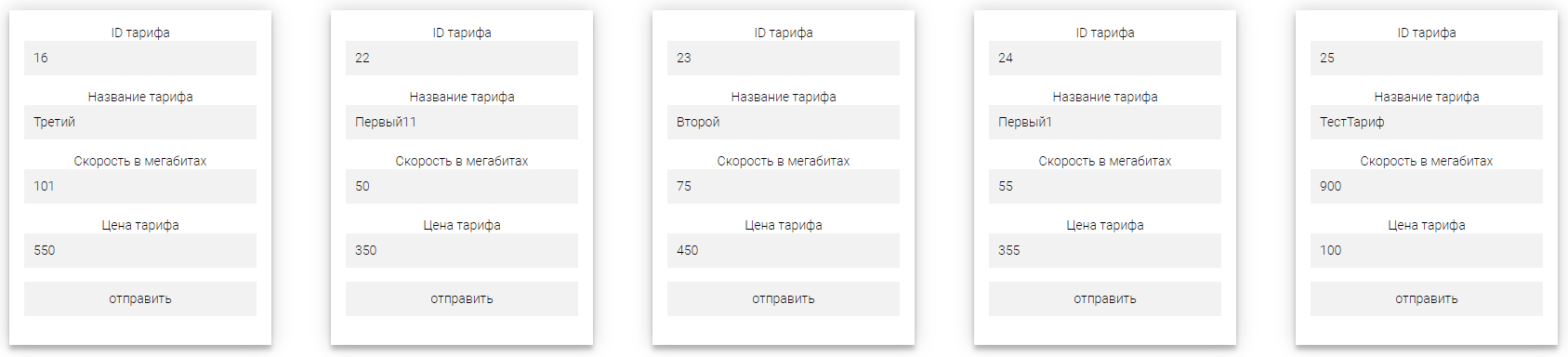


Рисунок 16 – результат обновления тарифа

Модуль удаления тарифа. Доступен администратору.

На этой странице администратор может удалить необходимый тариф написав его номер.

Код сервлета DeleteTarif представлен ниже.

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

request.setAttribute("TarifList", krdb.TarifOutput()); getServletContext().getRequestDispatcher("/DeleteTarif.jsp").forward(request, response);

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

request.setCharacterEncoding("UTF-8");

String id = request.getParameter("id");

krdb.removeTarif(id);

response.sendRedirect(request.getContextPath()+"/DeleteTarif");

}

}

При POST запросе сервлет считывает данные после этого удаляет запись в базе данных при помощи метода removeTarif. При GET запросе происходит считывание всех тарифов из базы данных, обёртывание их в list, а также передача этого list в контекст запроса и последующее считывание переданного list на jsp странице точно также как это уже описывалось ранее.

Код метода removeTarif представлен ниже.

public static void removeTarif(String id) {

try { Class.forName("org.sqlite.JDBC").getDeclaredConstructor().newInstance();

try (Connection conn = DriverManager.getConnection(url1)) {String sql = "DELETE FROM tarif WHERE ID = ?";

try (PreparedStatement preparedStatement = conn.prepareStatement("PRAGMA foreign\_keys = ON")) {

preparedStatement.execute();

}

try (PreparedStatement preparedStatement = conn.prepareStatement(sql)) {

preparedStatement.setString(1, id);

preparedStatement.executeUpdate();

}

conn.close();

}

} catch (Exception ex) {

System.out.println(ex);

}

}

Тариф удаляется при помощи SQL запроса DELETE, а также базе данных передаётся инструкция "PRAGMA foreign\_keys = ON" чтобы включить по умолчанию отключённые связи по внешним ключам чтобы обеспечить работу каскадного удаления.

Страница для удаления тарифа представлена на рисунке 17.

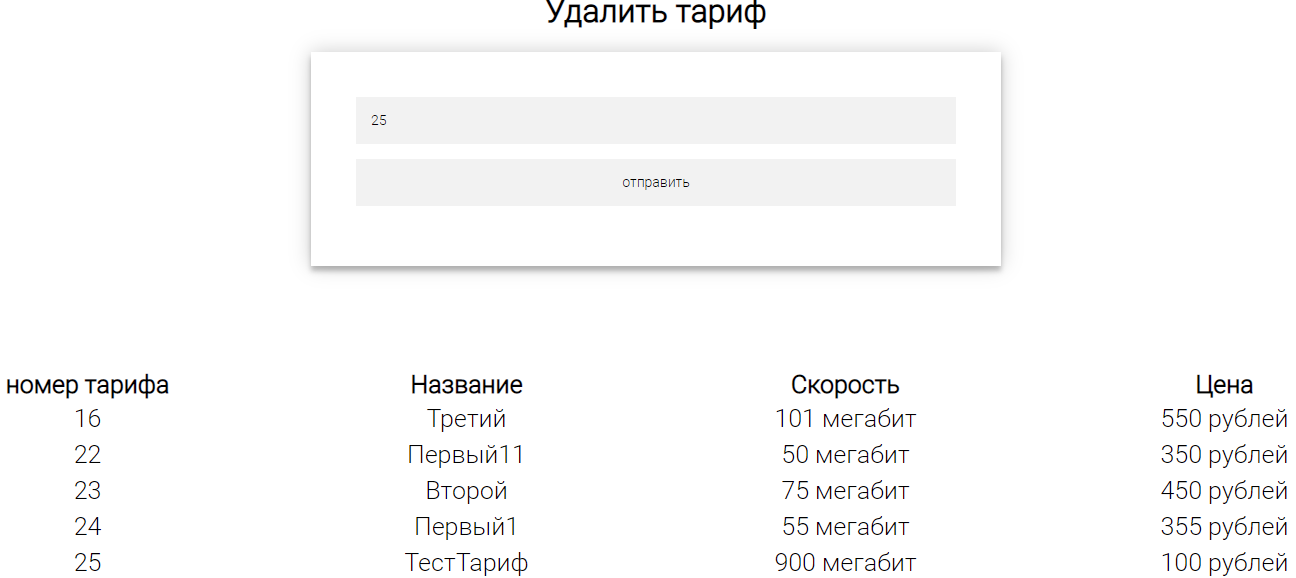


Рисунок 17 – страница для удаления тарифа

Результат удаления тарифа представлен на рисунке 18.

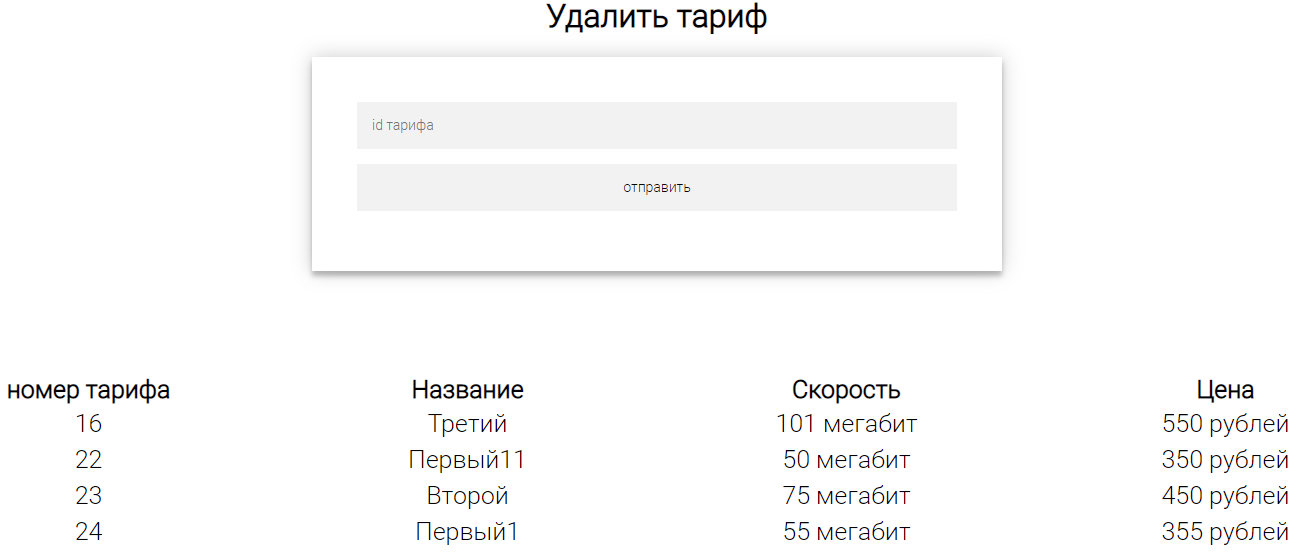


Рисунок 18 – результат удаления тарифа

Информационный портал для интернет-провайдера был успешно реализован.

# ЗАКЛЮЧЕНИЕ

С давних времён различные технологии всё сильнее включаются в жизнь человека. Деятельность многих организаций напрямую зависит от применения различных веб-технологий.

В ходе выполнения данной курсовой работы была дана классификация веб-ресурсов и было определено, что разрабатываемый веб-ресурс принадлежит к категории корпоративных сайтов исходя из предметной области данной курсовой работы.

Также была описана потенциальная аудитория информационного портала интернет-провайдер. Кроме того, было описано назначение и суть технического задания, а также техническое задание было составлено для разрабатываемого информационного портала.

В ходе выполнения курсовой работы были определены используемые языки и технологии программирования, а именно: языки программирования Java, JavaScript, а также технологии html, css, jsp.

Были рассмотрены различные системы управления базами данных и исходя из описанных преимуществ была выбрана SQLite. Также были рассмотрены некоторые веб-сервера и для реализации информационного-портала был выбран Apache Tomcat в виду того, что он был изначально разработан для того, чтобы взаимодействовать с содержимым, которое создаётся Java приложениями.

Был завершён этап программной реализации информационного портала, а именно была описана структура информационного портала, разработан дизайн-макет и все необходимые модули.

Все поставленные в ходе курсового проекта задачи были решены, а цель в виде применения на практике знаний, полученных в процессе изучения курса «Современные технологии интернет-программирования была достигнута.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Крюкова, А. А. ERP-системы и корпоративные порталы: учебное пособие / А. А. Крюкова. — Самара: ПГУТИ, 2019. — 255 с.
2. Москвитин, А. А. Данные, информация, знания: методология, теория, технологии: монография / А. А. Москвитин. — Санкт-Петербург: Лань, 2022. — 236 с.
3. Васин, Н. Н. Технологии пакетной коммутации: учебник / Н. Н. Васин. — Санкт-Петербург: Лань, 2022. — 284 с.
4. Интернет-реклама и pr: методические указания / составитель М. С. Круглова. — Сочи: СГУ, 2020. — 42 с.
5. Гендина, Н. И. Информационная культура личности: технология продуктивной интеллектуальной работы с информацией в условиях интернет-среды: учебное пособие: в 2 томах / Н. И. Гендина. — Кемерово: КемГИК, 2020 — Том 1 — 2020. — 357 с.
6. Москалев, С. М. Связи с общественностью в органах власти: учебное пособие / С. М. Москалев, Т. Г. Виноградова, Я. И. Семилетова. — Санкт-Петербург: СПбГАУ, 2021. — 163 с.
7. Вейцман, В. М. Проектирование информационных систем: учебное пособие / В. М. Вейцман. — Санкт-Петербург: Лань, 2022.
8. Федотов, А. В. Компьютерное управление в производственных системах: учебное пособие для вузов / А. В. Федотов, В. Г. Хомченко. — 2-е изд., стер. — Санкт-Петербург: Лань, 2021. — 620 с.
9. Информационные технологии в процессе подготовки современного специалиста: сборник научных трудов / под редакцией С. В. Мицук. — Липецк: Липецкий ГПУ, 2020
10. Васюткина, И. А. Разработка серверной части web-приложений на Java: учебное пособие / И. А. Васюткина. — Новосибирск: НГТУ, 2021. — 83 с.
11. Даева, С. Г. Основы системного администрирования и администрирования СУБД: учебно-методическое пособие / С. Г. Даева. — Москва: РТУ МИРЭА, 2021. — 75 с.
12. Серова, Е. А. Использование web-технологий при создании информационных систем: учебно-методического пособие / Е. А. Серова. — Москва: МИСИ – МГСУ, 2020. — 55 с.
13. Белугина, С. В. Разработка программных модулей программного обеспечения для компьютерных систем. Прикладное программирование: учебное пособие / С. В. Белугина. — Санкт-Петербург: Лань, 2020. — 312 с.
14. Орлова, И. В. Информатика. Практические задания: учебное пособие / И. В. Орлова. — Санкт-Петербург: Лань, 2022. — 140 с.
15. - PostgreSQL [Электронный ресурс]: Skill Factory. – Режим доступа: https://blog.skillfactory.ru/glossary/postgresql/ (дата обращения: 01.05.2022).
16. - PostgreSQL, MariaDB и SQLite [Электронный ресурс]: Over Coder. – Режим доступа: https://overcoder.net/manuals/postgresql-mariadb-i-sqlite-sravnenie-baz-dannyh (дата обращения: 01.05.2022).
17. - MySQL [Электронный ресурс]: Skill Factory. – Режим доступа: https://blog.skillfactory.ru/glossary/mysql/ (дата обращения: 01.05.2022).
18. - SQLite [Электронный ресурс]: Skill Factory. – Режим доступа: https://blog.skillfactory.ru/glossary/postgresql/ (дата обращения: 01.05.2022).
19. Садыков, А. М. Методы разработки веб-приложений: учебно-методическое пособие / А. М. Садыков. — Иваново: ИГЭУ, 2019. — 72 с.
20. – Что такое Apache [Электронный ресурс]: Eternalhost – Режим доступа: https://eternalhost.net/blog/hosting/web-server-apache (дата обращения: 01.05.2022).

# ПРИЛОЖЕНИЕ А

1. Общие сведения

1.1 Полное наименование веб-ресурса и ее условное обозначение.

Информационный портал интернет-провайдера

1.2 Наименование разработчика веб-ресурса и реквизиты заказчика.

Заказчик – кафедра ИСТ БИТИ НИЯУ МИФИ

Разработчик – студент группы ИФСТ-31 Коновалов Эльдар Наилевич

1.3. Основания для разработки веб-ресурса

Основанием для проведения разработки является задание на выполнение курсовой работы в рамках учебного предмета «Технологии интернет-программирования», в соответствии с учебным планом подготовки бакалавров по направлению «Информационные системы и технологии».

1.4. Плановые сроки начала и окончания работы по веб-ресурсу:

* начало работ по созданию веб-ресурса – январь 2022;
* окончание работ по созданию веб-ресурса – май 2022.

1.5. Источник финансирования работ по созданию веб-ресурса.

Собственные средства разработчика.

1.6. Порядок оформления и предъявления заказчику результатов работ по созданию веб-ресурса:

Проектная документация должна быть разработана в соответствии с ГОСТ 34.201-89 и ГОСТ ЕСПД. Процедуры приемки - передачи результатов работ оформляются актами приемки-передачи.

2. Назначение и цели создания веб-ресурса

2.1 Назначение веб-ресурса.

Информационный портал предназначен для повышения эффективности работы интернет-провайдера.

2.2 Цели создания веб-ресурса.

* Упрощение процесса получения услуг от интернет-провайдера для конечных пользователей;
* Ускорение процесса получения помощи для конечных пользователей;
* Оперативное получение информации о текущих услугах интернет-провайдера для конечных пользователей.

3. Характеристика объекта автоматизации

3.1. Краткие сведения об объекте автоматизации.

Объектом автоматизации является деятельность интернет-провайдера.

3.2. Сведения об условиях эксплуатации объекта автоматизации.

Веб-ресурс используется пользователями и администраторами из организации интернет-провайдера

Функционирование веб-ресурса должно происходить в требуемых условиях:

* при конструктивной температуре, давлении и допустимом уровне запыленности;
* «Гигиенические требования к микроклимату производственных помещений». Специалист выполняет соответствующие ему функции ежедневно (кроме субботы и воскресения) с 9.00 до 18.00 часов.

4. Требования к системе

4.1. Требования к системе в целом.

4.1.1. Требования к структуре и функционированию системы

Информационный портал интернет-провайдера включает в себя подсистемы:

* Загрузки базы данных;
* Регистрации;
* Авторизации;
* Работы с тарифом.

Подсистема загрузки базы данных:

* загружает db-файл базы данных;
* считывает информацию о существующих объектах и связях между ними.

Подсистема регистрации:

* получает данные с заполненных полей;
* вносит данные в базу данных;
* переадресует пользователя в личный кабинет.

Подсистема авторизации:

* получает данные с заполненных полей;
* проверяет данные на совпадение с данными из базы данных;
* переадресует пользователя в личный кабинет.

Подсистема работы с тарифом:

* Добавления тарифа в базу данных;
* Изменения тарифа в базе данных;
* Удаления тарифа из базы данных;
* Вывод тарифов из базы данных.

4.1.2. Перспективы системы, модернизация системы.

При модернизации могут вноситься изменения или осуществляться дополнения в необходимые для функционирования веб-ресурса файлы.

4.1.3. Требования к надежности комплекса.

Необходимо, чтобы веб-ресурс обладал устойчивостью к отказам оборудования и программных систем, а также электропитания. Для надежной работы необходимо высоконадежные аппаратные и программные системы. Требования надежности должны быть регламентированы для следующих аварийных ситуаций:

* выход из строя аппаратных средств системы;
* отсутствие электроэнергии;
* выход из строя программных средств системы;
* неверные действия персонала компании;
* пожар, взрыв и тому подобное.

5. Порядок контроля и приемки системы.

Приемка этапа заключается в рассмотрении и оценке проведенного объема работ и предъявленной технической документации в соответствии с требованиями настоящего технического задания.

Ответственность за организацию и проведение приемки системы должен нести заказчик. Приемка системы должна производиться по завершению приемки всех задач системы. При этом необходимо предоставить обеспечение материальной частью (технические средства), проектной документацией и специально выделенным персоналом.

Заказчик должен предъявлять веб-ресурс ведомственной приемочной комиссии, при этом он обязан обеспечить нормальные условия работы данной комиссии в соответствии с принятой программой приемки.

Завершающим этапом при приемке системы должно быть составление акта приемки.