МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет ФИСТ

Кафедра ИВК

Отчет по практике

защищен с оценкой\_\_\_\_\_\_\_

Руководитель

практики от университета

ст.преподаватель

Новиков А.А.

\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (Ф.И.О.)

ОТЧЕТ по производственной практике

(вид практики)

Студента Иваново Анастасии Петровны

(Ф.И.О.)

Направление (специальность,

профиль) подготовки 09.03.02 Информационные системы и

технологии, профиль «Информационные системы и технологии»

Группа ИСТбд-41

Место прохождения практики кафедра ИВК

(наименование профильной организации, подразделение университета)

Ульяновск

2020г.

## **Постановка задачи**

Разработать информационную систему работы курьерской службы. Система должна представлять собой удобный и простой в использовании веб-интерфейс, с помощью которого заказчик может подать заявку на доставку. Также у заказчика должен быть личный кабинет, в котором он может посмотреть свою историю заказов. Написать ТЗ.

**1. Техническое задание**

## **1.1 Назначение и цели создания системы**

Разрабатываемая система предназначена для автоматизации работы курьерской службы, специализированная на грузоперевозку, а именно для автоматизации процесса отправки и получения заказа.

Данная система может использоваться в организациях, где администратор будет просматривать таблицы с заказами, с возможностью удалять заказы и получать отчет по всем заказам. Курьер будет проводит движение заказа, меняя его статус. Клиенты могут заказать доставку груза на сайте компании и получить печатную форму заказа.

Цели создания системы:

* уменьшить сроки доставки грузов;
* упростить взаимодействие заказчика и курьера.

## **1.2 Характеристика объекта автоматизации**

### *1.2.1 Общее описание*

Объектом автоматизации информационной системы (далее ИС) является деятельность курьерской службы, нацеленная на повышение эффективности их работы благодаря использованию веб-сайта – «Ульяновская служба доставки», где осуществлена возможность заказать доставку. Сайт имеет простой интерфейс, и не возникнет сложностей для совершения заказа. Данный сайт значительно упрощает отправку и получение посылок, освобождая грузовладельца от всех забот по организации перевозки, выбору вида транспорта и схемы доставки груза.

### *1.2.2 Состав и порядок функционирования*

Курьерская служба — это выгодная и полезная услуга, которой можно воспользоваться как в пределах деловой, так и повседневной жизни. Основные функции: оформление заказа, подбор автомобиля, возможность отследить статус заказа, доставка груза.

Основные операции по сбору и обработке информации:

* Заполнение бланка заказа;
* Поиск свободных курьеров с помощью телефонных звонков;
* Распределение заказов между курьерами;
* Заполнение накладной после доставки;
* Уведомление о выполнении заказа;
* Ведение отчета о заказах.

Все операции курьерской службы будут автоматизированы путем создания веб-сайта, обеспечивая доступ с любого устройства.

## **1.3 Общие требования к системе**

### *1.3.1 Требования к структуре и функционированию системы*

Система должна представлять собой интернет-портал, на котором клиенты (зарегистрированные и незарегистрированные), курьер и администратор работают в трех разных ролях, разделенных механизмами авторизации.

В состав системы должны входить следующие подсистемы:

* подсистема авторизации, регистрации и разграничения доступа; авторизация пользователей под индивидуальными почтами и паролями; за каждой учетной записью закреплена своя роль;
* подсистема рейтинга заказчиков и курьеров;
* подсистема распределения заказов между курьерами;
* подсистема формирования отчета о заказах за определенный период;
* подсистема учета заказчиков;
* подсистема учета курьеров;
* подсистема расчета стоимости доставки груза и совершения заказа предназначена для расчета стоимости доставки пользователем, а затем

для предоставления печатной формы сделанного заказ;

* подсистема движения заказа, предназначенная для отслеживания заказа и установления его статуса курьером.

Система должна состоять из двух частей: сервер управления веб-интерфейсом и клиентская часть в виде веб-сайта, которое используется обычными пользователями.

Перспективы развития системы предполагают добавление к вышеописанной системе функционала отслеживания груза.

### *1.3.2 Дополнительные требования*

#### 1.3.2.1 Требования к персоналу

Численность пользователей может быть неограниченная.

Администратор должен владеть навыками пользования ПК на базовом уровне: важно уметь авторизовываться, заполнять формы, также иметь навыки работы с \*.xls файлами. Квалификация других сотрудников не имеет значения.

Режим работы персонала не влияет на использование системы.

Порядок подготовки администратора системы – знакомство и подготовка к работе с базой данных системы и ее основным функционалом, доступным для администратора.

#### 1.3.2.2 Требования к надежности

Система должна сохранять работоспособность и обеспечивать восстановление своих функций при возникновении следующих внештатных ситуаций:

* при сбоях в системе электроснабжения аппаратной части, приводящих к перезагрузке ОС, восстановление программы должно происходить после перезапуска ОС и запуска исполняемого файла системы;
* при ошибках в работе аппаратных средств (кроме носителей данных и программ) восстановление функции системы возлагается на ОС;
* при ошибках, связанных с программным обеспечением (ОС и драйверы устройств), восстановление работоспособности возлагается на ОС.

Для защиты аппаратуры от бросков напряжения и коммутационных помех должны применяться сетевые фильтры.

В системе должна быть обеспечена возможность восстановления данных с внешнего накопителя после восстановления активного накопителя. БД системы необходимо резервировать минимум 1 раз в месяц.

Специальные дополнительные требования по составу и количественным значениям показателей надежности для подсистем модернизируемых модулей и, соответственно, к создаваемой системе в целом не предъявляются.

#### 1.3.2.3 Требования к патентной чистоте

Система должна использовать только лицензионное программное обеспечение. Установка системы в целом, как и установка отдельных частей системы не должна предъявлять дополнительных требований к покупке лицензий на программное обеспечение сторонних производителей.

#### 1.3.2.4 Требования по стандартизации, унификации и тиражированию

Разработка системы должна проводиться с соблюдением требований действующих государственных стандартов в соответствии с областью их распространения, одним из которых является использование современных технологий, протоколов и стандартов для взаимодействия с внешними системами.

Унификации подлежат все составляющие пользовательского интерфейса: система отчетности, формы отображения информации на экране, типы и формы сообщений и запросов пользователю.

## **1.4 Требования к функциям, выполняемым системой**

Реализация функций каждой из задач должна быть закончена в указанные сроки. Функции учета эффективности следует реализовать после реализации остальных задач.

### *1.4.1 Регистрация пользователя в системе*

Данная функция осуществляет регистрацию с разделением уровней доступа разрешений к управлению системы.

Входные данные: Почта, Пароль.

После успешной авторизации пользователю системы должно отобразится окно работы.

Необходимо учесть, что в окно регистрации можно перейти от окна авторизации. В БД почта служит в качестве логина. Пароль должен хэшироваться для большей защиты.

Для регистрации нового пользователя требуется заполнить поля: почта, пароль и повторить пароль. Почта должна быть не менее 7 символов. Пароль должен быть не менее 3х символов.

### *1.4.2 Авторизация и аутентификация в системе*

Данная функция позволяет авторизоваться пользователю под своими учетными данными с разделением уровней доступа разрешений к управлению системой.

Входные данные: Почта, Пароль.

При успешности переход осуществляется аналогично регистрации.

### *1.4.3 Движение заказа*

Данная функция позволяет определять статус заказа и его стоимость.

Входные данные: статус и информация о грузе (адрес, вес, длина, высота, ширина) передаются из БД.

Смена статуса курьером и окончательная стоимость доставки отображается в интерфейсе пользователя.

### *1.4.4 Печать отчета о совершенном заказе*

Функция позволяет заказчику получить печатную форму отчета о совершенном заказе.

Входные данные: информация о грузе (фио, номер телефона, адрес, дата, время, доступность отчета, статус, вес, длина, высота, ширина, оплата отправителем, цена) передаются из БД.

После смены статуса заказа курьером появляется возможность совершить печать отчета о заказе.

Необходимо учесть, что шаблон формы не зависит от данных и постоянен. Динамически изменяются только данные о пользователе и информация о заказе.

### *1.4.5 Подача заявки на доставку*

Функция позволяет заказчику подать заявку на доставку груза.

Входные данные: информация о грузе (фио, номер телефона, адрес, дата, время, вес, длина, высота, ширина, оплата отправителем).

После успешной подачи заявки совершенный заказ отображается в личном кабинете заказчика.

### *1.4.6 Распределение заказов между курьерами*

Функция позволяет распределять заказы между курьерами.

Входные данные: информация о грузе (адрес, район), и информация о курьере (район).

При условии, что к конкретному району прикреплен один курьер, заказ автоматически распределяется данному курьеру. В противном случае, администратор вручную распределяет заказы по курьерам. Если же груз является негабаритным, он распределяется на курьера с газелью, вне зависимости от района.

### *1.4.7 Рейтинг заказчиков и курьеров*

Функция позволяет заказчику оценивать работу курьера, а также курьеру оценивать заказчика.

Входные данные: оценка (от 1 до 5)

После доставки груза и заказчик, и курьер могут оценить услуги и добросовестность соответственно. Для заказчиков рейтинг также влияет на окончательную стоимость последующих заказов.

## **1.5 Требования к видам обеспечения**

### *1.5.1 Информационное обеспечение*

Информационное обеспечение системы должно базироваться на базах данных предприятия «Ульяновская служба доставки». Для функционирования системы необходимо следующее информационное обеспечение:

* каталог заказов;
* справочник курьеров;
* каталог заказчиков.

Информационное обеспечение системы должно являться совокупностью информационных массивов в БД всей системы, используемы системой управления базой данных (СУБД) MS SQL.

Обеспечить контроль ввода в системе. Процедура не предусматривает придание юридической силы электронным документам.

В системе должен быть реализован многопользовательский режим доступа к данным. При этом должна осуществляться проверка на доступ к данным в информационной системе.

Время генерации страницы не должно превышать 5 секунд. Время загрузки страницы должно совпадать с временем генерации.

Пользователям запрещается получать данные с помощью языка запросов без использования прикладного программного обеспечения.

Необходимо использовать резервное копирование базы данных раз в день. При этом должны сохраняться резервные копии за предыдущие два дня, а также каждая копия, созданная в воскресенье, в течение последнего месяц.

### *1.5.2 Требования к математическому обеспечению*

Необходимо разработать алгоритмы, реализующие следующие возможности со стороны администратора:

* управление пользователями: удаление, просмотр деталей;
* формирование отчетов;
* авторизация;
* управление заказами: удаление, просмотр деталей;
* распределение заказов между курьерами.

Со стороны курьера:

* управление заказами: удаление, изменение, просмотр деталей;
* смена статуса заказа;
* расчет стоимости доставки;
* получение отчета;
* авторизация и регистрация;
* оценивание заказчиков.

Со стороны заказчика:

* подача заявки на доставку;
* печать информации о заказе;
* предварительный расчет стоимости доставки;
* оценивание курьеров;
* авторизация и регистрация.

Система должна содержать математические методы расчета:

* стоимости доставки исходя из параметров груза и расстояния отправки;
* рейтинга курьеров и заказчиков, исходя из суммы и количества всех голосов;
* расстояния от города отправки до города получения.

При разработке информационной системы необходимо использовать  
возможности языка и платформы для реализации алгоритмов. При  
возможности использовать библиотеки, разрабатываемые по открытой лицензии. При отсутствии такой возможности необходимо самостоятельно реализовать функциональность.

### *1.5.3 Требования к программному обеспечению*

Требования к используемому программному обеспечению системы определяются на стадии разработки технического проекта. При выборе программного обеспечения необходимо учитывать требования к патентной чистоте. В состав программных средств должны входить:

* серверная платформа, включающая БД и программная среда для веб-разработчиков;
* любой редактор электронных таблиц, работающий с расширением xls;
* браузер Yandex.

Клиентская часть разрабатываемой информационной системы должна корректно отображаться и функционировать на данном браузере.

Серверная часть информационной системы должна корректно разворачиваться на платформе, выполняя все принятые ограничения в БД.

Экранные формы должны соответствовать правилам стилизации вебориентированных информационных систем, принятым на момент разработки технического задания (далее ТЗ).

К обеспечению качества ПС предъявляются следующие требования:

* функциональность должна обеспечиваться выполнением подсистемами всех их функций;
* надежность должна обеспечиваться за счет предупреждения ошибок – не допущения ошибок в готовых ПС;
* легкость применения должна обеспечиваться за счет применения покупных программных средств;
* эффективность должна обеспечиваться за счет принятия подходящих, верных решений на разных этапах разработки ПС и системы в целом;
* сопровождаемость должна обеспечиваться за счет высокого качества документации по сопровождению, а также за счет использования в программном тексте описания объектов и комментариев; использованием осмысленных и устойчиво различимых имен объектов; размещением не больше одного оператора в строке текста программы; избеганием создания фрагментов текстов программ с неочевидным смыслом;
* также на каждом этапе в разработке ПС должна проводится проверка правильности принятых решений по разработке и применению готовых ПС.

## **2. Проект решения**

Чтобы авторизоваться в системе как заказчик, нужно в правом верхнем углу главной страницы выбрать «Войти», а для того чтобы зарегистрироваться, нужно выбрать «Зарегистрироваться» (рисунок 1).

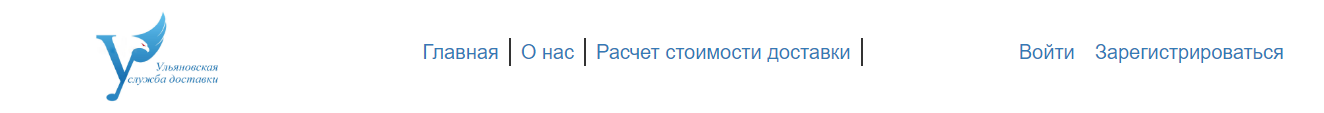


Рисунок 1 – Кнопки авторизации и регистрации

Нажав на «Войти» перед заказчиком откроется поле ввода информации для входа, где нужно ввести почту и пароль для авторизации (рисунок 2).

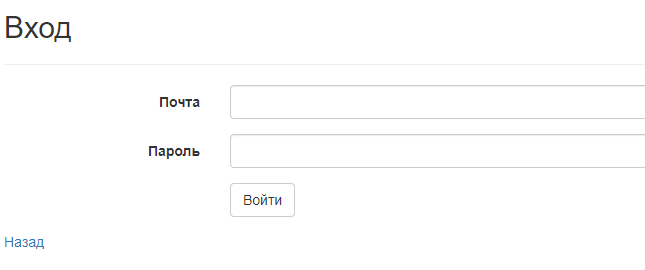


Рисунок 2 – Поле авторизации

Авторизовавшись, заказчик может в своем личном кабинете поменять свою почту или пароль, нажав на кнопку изменить «Изменить почту или пароль»; перед ним откроется окно, в котором нужно ввести, что он хочет изменить (рисунок 3).

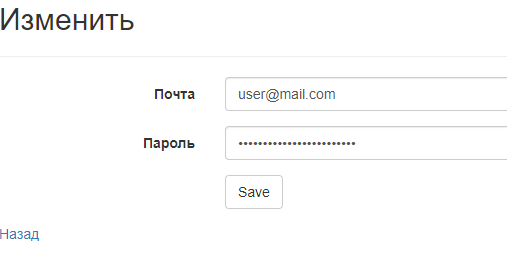


Рисунок 3 – Страница изменения почты и пароля

Если заказчик не авторизован, то нажав на кнопку «Зарегистрироваться» перед ним откроется поле ввода информации для регистрации, где нужно ввести почту, пароль, и повторить пароль для регистрации (рисунок 4). При вводе полей без ошибок, заказчик успешно зарегистрируется.

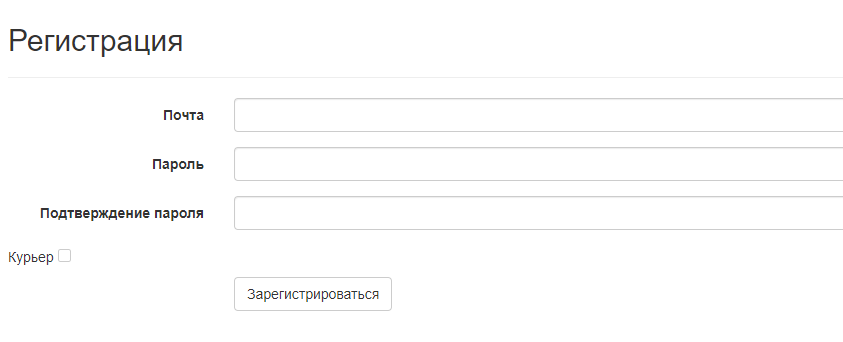


Рисунок 4 – Страница регистрации

Оказавшись на главной странице системы, заказчик может сразу перейти в личный кабинет по ссылке в верхнем правом углу (рисунок 5).

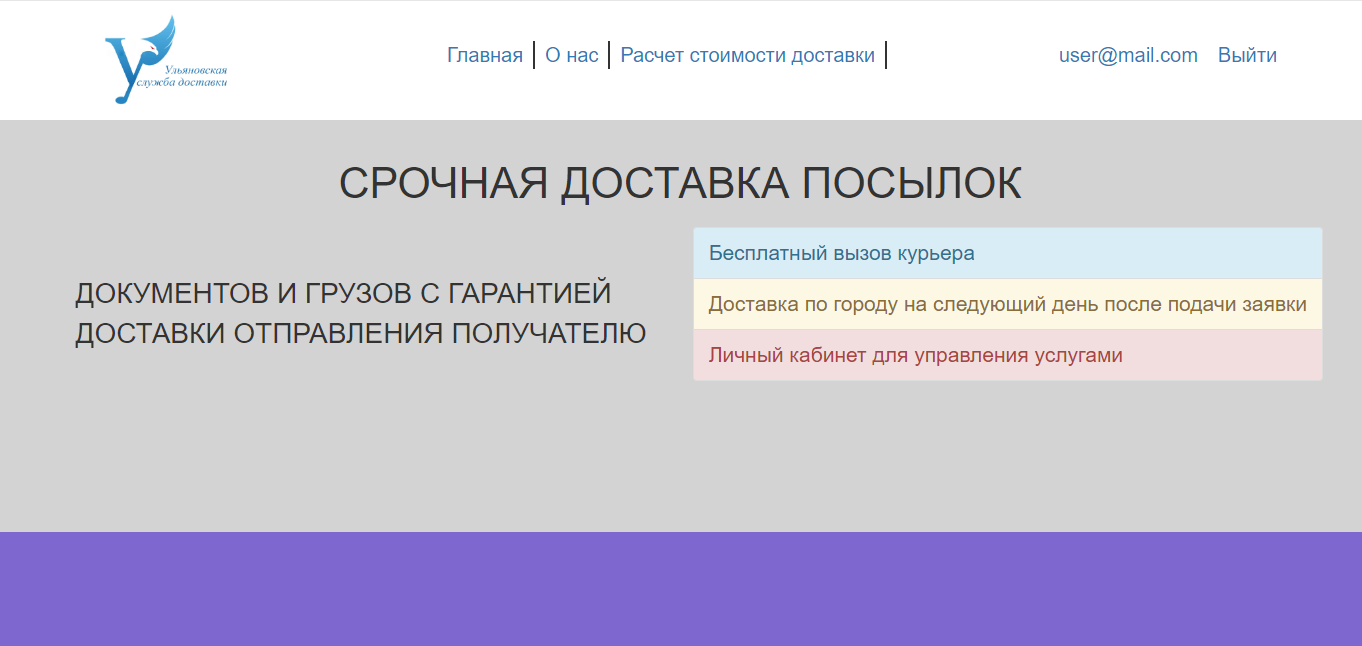


Рисунок 5 – Главная страница

В личном кабинете отображаются все заявки заказчика. Также есть возможность подачи новых заявок (рисунок 6).

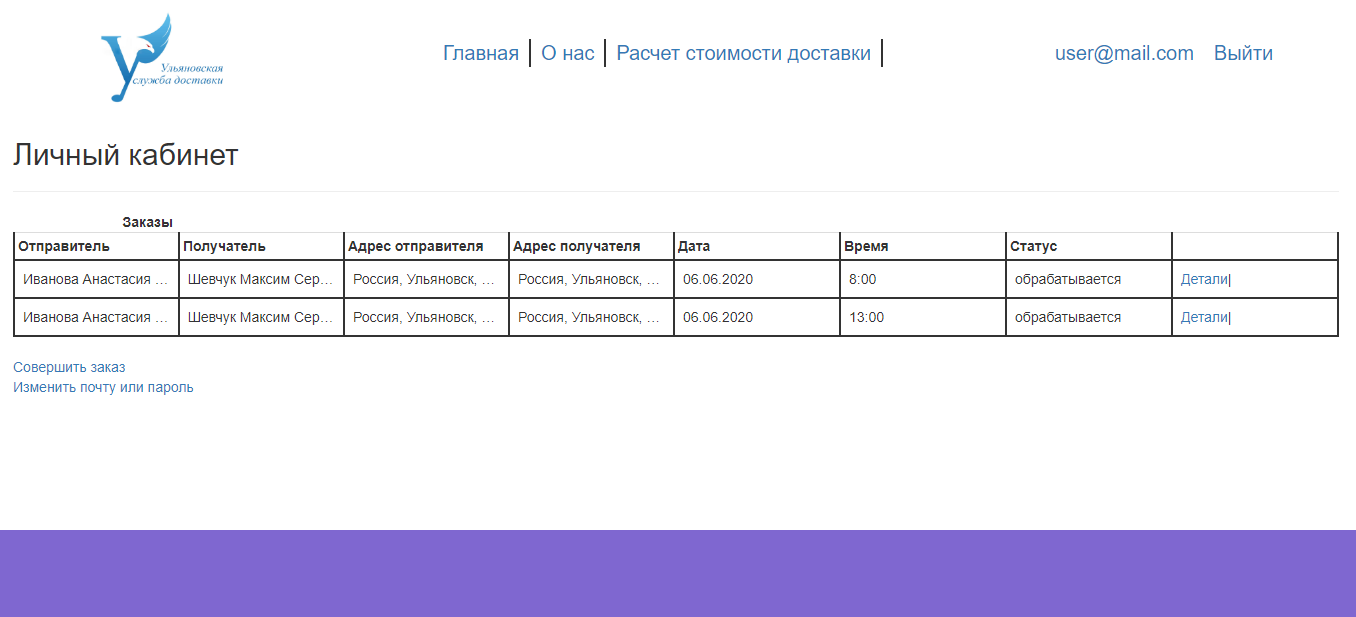


Рисунок 6 – Личный кабинет

Для этого нужно нажать «Совершить заказ» и в открывшейся форме ввести данные о заказе и грузе, как показано на рисунках 7-8.

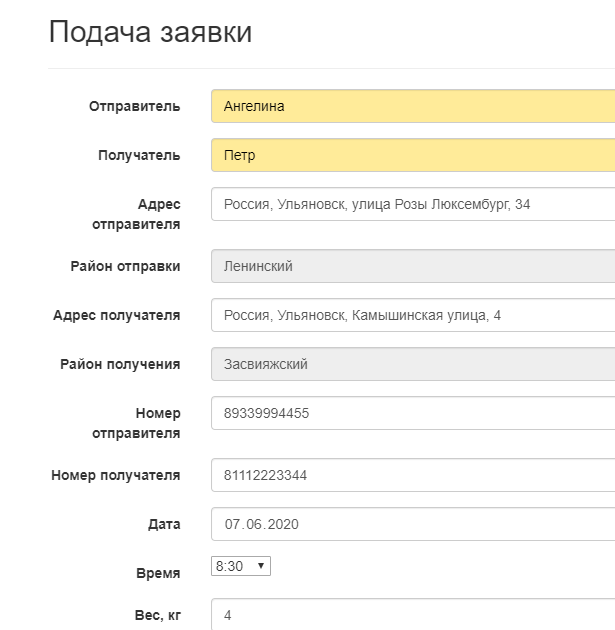


Рисунок 7 – Страница подачи заявки на доставку

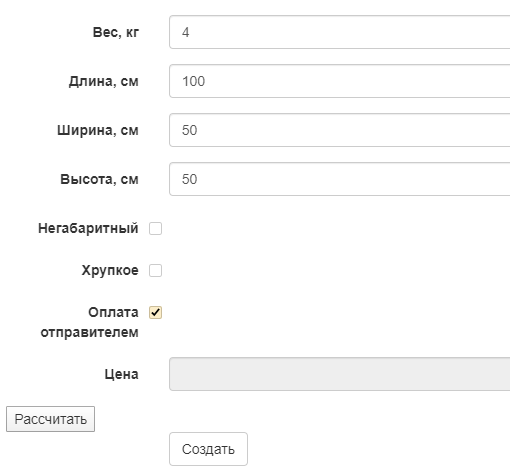


Рисунок 8 – Страница подачи заявки на доставку(продолжение)

После того, как пользователь введет необходимую информацию, по нажатию кнопки «Рассчитать» в поле «Цена» появляется примерная стоимость доставки (рисунок 9)

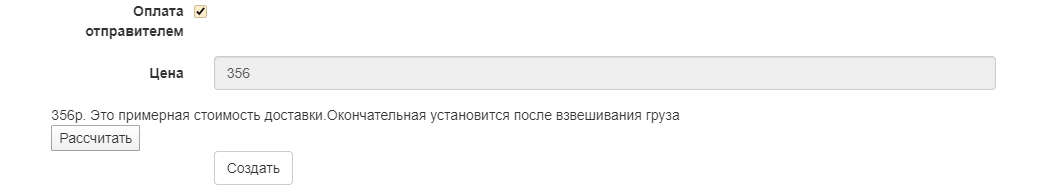


Рисунок 9 – Расчет стоимости

Для завершения подачи заявки, заказчику необходимо нажать кнопку «Создать». Перед ним откроется страница личного кабинета, где отобразиться новая заявка на доставку (рисунок 10).

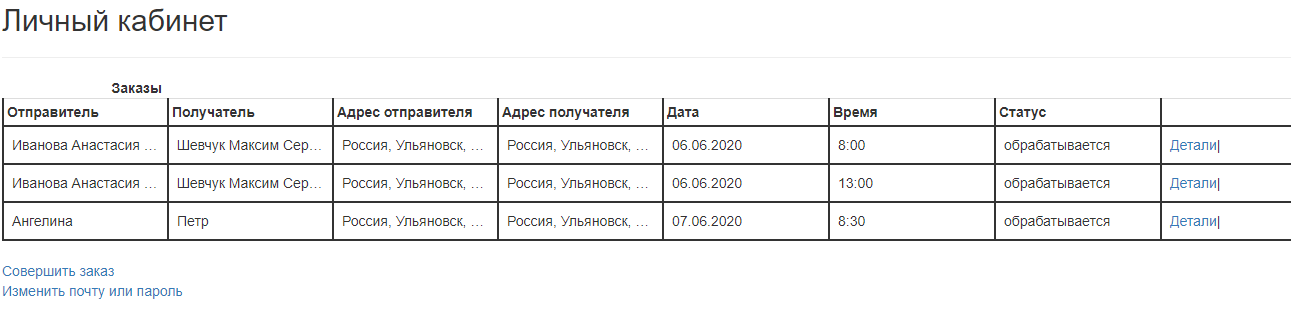


Рисунок 10 – Страница личного кабинета после совершения заказа

# **Код проекта**

Файл AccountController.cs (Авторизация)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Web;

using System.Web.Mvc;

using System.Web.Security;

using UlskDel.Models;

namespace UlskDel.Controllers

{

public class AccountController : Controller

{

OrderContext db = new OrderContext();

public ActionResult Login()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Login(LoginModel model)

{

if (ModelState.IsValid)

{

string pwd = GetHash(model.Password);

// поиск пользователя в бд

User user = db.Users.FirstOrDefault(u => u.Email == model.Name && u.Password == pwd);

if (user != null)

{

FormsAuthentication.SetAuthCookie(model.Name, true);

return RedirectToAction("Index", "Home");

}

else

{

ModelState.AddModelError("", "Пользователя с таким логином и паролем нет");

}

}

return View(model);

}

public ActionResult Register()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Register(RegisterModel model, bool isCourier = false)

{

if (ModelState.IsValid)

{

string pwd = GetHash(model.Password);

User user = db.Users.FirstOrDefault(u => u.Email == model.Name && u.Password == pwd);

int role = 1;

if (isCourier)

{

role = 3;

}

if (user == null)

{

User x = db.Users.Add(new User { Email = model.Name, Password = pwd, RoleId = role });

if (!isCourier)

{

db.Customers.Add(new Customer { Id = x.Id, sumVotes = 0, totalVotes = 0 });

}

db.SaveChanges();

user = db.Users.Where(u => u.Email == model.Name && u.Password == pwd).FirstOrDefault();

if (user != null)

{

FormsAuthentication.SetAuthCookie(model.Name, true);

if (isCourier)

{

return RedirectToAction("Create", "Couriers");

} else

{

return RedirectToAction("Index", "Home");

}

}

}

else

ModelState.AddModelError("", "Пользователь с таким логином уже существует");

}

return View(model);

}

public ActionResult Logoff()

{

FormsAuthentication.SignOut();

return RedirectToAction("Index", "Home");

}

private string GetHash(string input)

{

var md5 = MD5.Create();

var hash = md5.ComputeHash(Encoding.UTF8.GetBytes(input));

return Convert.ToBase64String(hash);

}

}

}

Файл CustomersController.cs (Добавление, редактирование, удаление информации о аказчиках)

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Entity;

using System.Linq;

using System.Net;

using System.Web;

using System.Web.Mvc;

using UlskDel.Models;

namespace UlskDel.Controllers

{

public class CustomersController : Controller

{

private OrderContext db = new OrderContext();

// GET: Customers

public ActionResult Index()

{

var customers = db.Customers.Include(c => c.User);

return View(customers.ToList());

}

public ActionResult Rate(int id)

{

Order c = db.Orders.Include(y => y.Courier).FirstOrDefault(x => x.OrderId == id);

if (c != null)

return PartialView(c);

return HttpNotFound();

}

[HttpPost]

public ActionResult Rate(string Answer, [Bind(Include = "CourierId")] Order order)

{

//Курьер заказа

Courier c = db.Couriers.FirstOrDefault(x => x.Id == order.CourierId);

c.sumVotes = c.sumVotes + Convert.ToInt32(Answer);

c.totalVotes = c.totalVotes + 1;

db.Entry(c).State = EntityState.Modified;

db.SaveChanges();

User user = db.Users.Where(x => x.Email == User.Identity.Name).FirstOrDefault();

int id = user.Id;

return RedirectToAction("Details", "Customers", new { id });

}

// GET: Customers/Details/5

public ActionResult Details(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Customer customer = db.Customers.Include(t => t.Orders).FirstOrDefault(t => t.Id == id);

if (customer == null)

{

return HttpNotFound();

}

return View(customer);

}

// GET: Customers/Create

public ActionResult Create()

{

ViewBag.Id = new SelectList(db.Users, "Id", "Email");

return View();

}

// POST: Customers/Create

// Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойства, для которых следует установить привязку. Дополнительные

// сведения см. в статье https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "Id,sumVotes, totalVotes")] Customer customer)

{

if (ModelState.IsValid)

{

db.Customers.Add(customer);

db.SaveChanges();

return RedirectToAction("Index");

}

ViewBag.Id = new SelectList(db.Users, "Id", "Email", customer.Id);

return View(customer);

}

// GET: Customers/Edit/5

public ActionResult Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Customer customer = db.Customers.Find(id);

if (customer == null)

{

return HttpNotFound();

}

ViewBag.Id = new SelectList(db.Users, "Id", "Email", customer.Id);

return View(customer);

}

// POST: Customers/Edit/5

// Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойства, для которых следует установить привязку. Дополнительные

// сведения см. в статье https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include = "Id,sumVotes,totalVotes")] Customer customer)

{

if (ModelState.IsValid)

{

db.Entry(customer).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

ViewBag.Id = new SelectList(db.Users, "Id", "Email", customer.Id);

return View(customer);

}

// GET: Customers/Delete/5

public ActionResult Delete(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Customer customer = db.Customers.Find(id);

if (customer == null)

{

return HttpNotFound();

}

return View(customer);

}

// POST: Customers/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

Customer customer = db.Customers.Find(id);

db.Customers.Remove(customer);

db.SaveChanges();

return RedirectToAction("Index");

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}

Файл LKController.cs (Личный кабинет)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using UlskDel.Models;

using System.Data.Entity;

namespace UlskDel.Controllers

{

public class LKController : Controller

{

private OrderContext db = new OrderContext();

// GET: LK

public ActionResult Index()

{

User user = db.Users.Where(x => x.Email == User.Identity.Name).FirstOrDefault();

int id = user.Id;

if (User.IsInRole("customer"))

{

return RedirectToAction("Details", "Customers", new { id });

} else

return RedirectToAction("Details", "Couriers", new { id });

}

}

}

Файл OrdersController.cs (Работа с заказами)

using iTextSharp.text;

using iTextSharp.text.pdf;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Entity;

using System.IO;

using System.Linq;

using System.Net;

using System.Web;

using System.Web.Mvc;

using UlskDel.Models;

using iTextSharp.text.html;

using iTextSharp.tool.xml.pipeline.html;

using iTextSharp.tool.xml.html;

using iTextSharp.tool.xml.pipeline.css;

using iTextSharp.tool.xml;

using iTextSharp.tool.xml.pipeline.end;

using System.Xml.Linq;

using iTextSharp.tool.xml.parser;

using System.Text;

using ExcelLibrary.SpreadSheet;

namespace UlskDel.Controllers

{

public class OrdersController : Controller

{

private OrderContext db = new OrderContext();

[HttpGet]

[AllowAnonymous]

public JsonResult CheckAddress(string Address\_Sender)

{

var result = Address\_Sender.Contains("Ульяновск");

return Json(result, JsonRequestBehavior.AllowGet);

}

[HttpGet]

[AllowAnonymous]

public JsonResult CheckDate(DateTime Date)

{

DateTime now = DateTime.Now;

var result = Date.Date.CompareTo(now.Date) > 0 && Date.Date.CompareTo(now.AddYears(1)) < 0;

return Json(result, JsonRequestBehavior.AllowGet);

}

// GET: Orders

public ActionResult Index(int id = 2)

{

var elem = from s in db.Orders

select s;

List<SelectListItem> item = new List<SelectListItem>();

//фильтрация по периоду

item.Add(new SelectListItem { Text = "неделя", Value = "0"});

item.Add(new SelectListItem { Text = "месяц", Value = "1"});

item.Add(new SelectListItem { Text = "все", Value = "2", Selected = true});

DateTime week = DateTime.Now.AddDays(-7);

DateTime month = DateTime.Now.AddMonths(-1);

DateTime now = DateTime.Now;

ViewBag.id = item;

switch (id)

{

case 0: elem = elem.Where(s => s.Date.CompareTo(week)>0

&& s.Date.CompareTo(now)<0);break;

case 1: elem = elem.Where(s => s.Date.CompareTo(month)>0

&& s.Date.CompareTo(now) < 0); break;

case 2: elem = db.Orders;break;

}

return View(elem.ToList());

}

// GET: Orders/Details/5

public ActionResult Details(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Order order = db.Orders.Find(id);

if (order == null)

{

return HttpNotFound();

}

return View(order);

}

// GET: Orders/Create

public ActionResult Create()

{

return View();

}

// POST: Orders/Create

// Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойства, для которых следует установить привязку. Дополнительные

// сведения см. в статье https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "OrderId,Sender,Receiver,Address\_Sender,Address\_Receiver,Area\_Sender,Area\_Receiver,Phone\_Sender,Phone\_Receiver,Date,Time,Weight,Length,Width,Height,Price,Big,Fragile,Who\_pay")] Order order)

{

if (ModelState.IsValid)

{

User user = db.Users.Where(x => x.Email == User.Identity.Name).FirstOrDefault();

int id = user.Id;

order.CustomerId = id;

order.Status = "обрабатывается";

order.Print = false;

//Находим курьера для получения посылки

var elem = from s in db.Couriers

select s;

if (order.Big)

{

elem = elem.Where(x => x.oversize);

} else

{

elem = elem.Where(x => x.Area == order.Area\_Sender);

}

//привязываем курьера к данному заказу

if (elem.Count() == 1)

order.CourierId = elem.FirstOrDefault().Id;

db.Orders.Add(order);

db.SaveChanges();

return RedirectToAction("Index","LK");

}

return View(order);

}

// GET: Orders/Edit/5

public ActionResult Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Order order = db.Orders.Find(id);

if (order == null)

{

return HttpNotFound();

}

IEnumerable < SelectListItem > userTypeList = new SelectList(db.Couriers.ToList(), "Id", "Name");

ViewData["userTypeList"] = userTypeList;

return View(order);

}

// POST: Orders/Edit/5

// Чтобы защититься от атак чрезмерной передачи данных, включите определенные свойства, для которых следует установить привязку. Дополнительные

// сведения см. в статье https://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include = "OrderId,Sender,Receiver,Address\_Sender,Address\_Receiver,Area\_Sender,Area\_Receiver,Phone\_Sender,Phone\_Receiver,Date,Time,Status,Weight,Length,Width,Height,Big,Fragile,Who\_pay,Price,CustomerId,CourierId")] Order order)

{

IEnumerable<SelectListItem> userTypeList = new SelectList(db.Couriers.ToList(), "Id", "Name");

if (!ModelState.IsValid)

{

Console.WriteLine(order.CourierId);

}

if (ModelState.IsValid)

{

db.Entry(order).State = EntityState.Modified;

if (order.Status!="обрабатывается")

{

order.Print = true;

}

//Нашли курьера данного заказа

Courier courier = db.Couriers.FirstOrDefault(a => a.Id == order.CourierId);

if (order.Status != "получен" && courier.Orders.Count == 1)

{

//Курьер завершил все заказы и стал свободен

courier.time = DateTime.Now;

}

//заказчик

Customer customer = db.Customers.FirstOrDefault(a => a.Id == order.CustomerId);

//Стоимость в зависимости от рейтинга

if (customer.rating > 0 && customer.rating < 2.07)

{

order.Price = (int)Math.Round(order.Price \* 1.2);

}

else if (customer.rating >= 3.1)

{

order.Price = (int)Math.Round(order.Price \* 0.8);

}

db.SaveChanges();

return RedirectToAction("Index");

}

return View(order);

}

// GET: Orders/Delete/5

public ActionResult Delete(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Order order = db.Orders.Find(id);

if (order == null)

{

return HttpNotFound();

}

return View(order);

}

// POST: Orders/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

Order order = db.Orders.Find(id);

db.Orders.Remove(order);

db.SaveChanges();

return RedirectToAction("Index");

}

//печать в pdf

public ActionResult Print(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Order order = db.Orders.Find(id);

if (order == null)

{

return HttpNotFound();

}

string Path = Environment.GetFolderPath(Environment.SpecialFolder.Desktop) + "/MyFile2.pdf";

Document DC = new Document();

DC.SetPageSize(PageSize.A4.Rotate());

FileStream FS = System.IO.File.Create(Path);

PdfWriter writer = PdfWriter.GetInstance(DC, FS);

DC.Open();

BaseFont baseFont = BaseFont.CreateFont(@"C:\Windows\Fonts\arial.ttf", BaseFont.IDENTITY\_H, BaseFont.NOT\_EMBEDDED);

iTextSharp.text.Font font = new iTextSharp.text.Font(baseFont, iTextSharp.text.Font.DEFAULTSIZE, iTextSharp.text.Font.NORMAL);

PdfPTable table = new PdfPTable(3);//число столбцов

//Добавим в таблицу общий заголовок

PdfPCell cell = new PdfPCell(new Phrase("Отправитель ", font));

cell.Colspan = 3;

cell.HorizontalAlignment = 1;

//Убираем границу первой ячейки, чтобы балы как заголовок

cell.Border = 0;

table.AddCell(cell);

cell = new PdfPCell(new Phrase(new Phrase("ФИО", font)));

//Фоновый цвет (необязательно, просто сделаем по красивее)

cell.BackgroundColor = iTextSharp.text.BaseColor.LIGHT\_GRAY;

table.AddCell(cell);

cell = new PdfPCell(new Phrase(new Phrase("Телефон", font)));

//Фоновый цвет (необязательно, просто сделаем по красивее)

cell.BackgroundColor = iTextSharp.text.BaseColor.LIGHT\_GRAY;

table.AddCell(cell);

cell = new PdfPCell(new Phrase(new Phrase("Адрес", font)));

//Фоновый цвет (необязательно, просто сделаем по красивее)

cell.BackgroundColor = iTextSharp.text.BaseColor.LIGHT\_GRAY;

table.AddCell(cell);

table.AddCell(new Phrase(order.Sender, font));

table.AddCell(new Phrase(order.Phone\_Sender, font));

table.AddCell(new Phrase(order.Address\_Sender, font));

//Добавляем таблицу в документ

DC.Add(table);

//receiver

PdfPTable table1 = new PdfPTable(3);//число столбцов

//Добавим в таблицу общий заголовок

PdfPCell cell1 = new PdfPCell(new Phrase("Получатель ", font));

cell1.Colspan = 3;

cell1.HorizontalAlignment = 1;

//Убираем границу первой ячейки, чтобы балы как заголовок

cell1.Border = 0;

table1.AddCell(cell1);

cell1 = new PdfPCell(new Phrase(new Phrase("ФИО", font)));

//Фоновый цвет (необязательно, просто сделаем по красивее)

cell1.BackgroundColor = iTextSharp.text.BaseColor.LIGHT\_GRAY;

table1.AddCell(cell1);

cell1 = new PdfPCell(new Phrase(new Phrase("Телефон", font)));

//Фоновый цвет (необязательно, просто сделаем по красивее)

cell1.BackgroundColor = iTextSharp.text.BaseColor.LIGHT\_GRAY;

table1.AddCell(cell1);

cell1 = new PdfPCell(new Phrase(new Phrase("Адрес", font)));

//Фоновый цвет (необязательно, просто сделаем по красивее)

cell1.BackgroundColor = iTextSharp.text.BaseColor.LIGHT\_GRAY;

table1.AddCell(cell1);

table1.AddCell(new Phrase(order.Receiver, font));

table1.AddCell(new Phrase(order.Phone\_Receiver, font));

table1.AddCell(new Phrase(order.Address\_Receiver, font));

//Добавляем таблицу в документ

DC.Add(table1);

//order

PdfPTable table2 = new PdfPTable(6);//число столбцов

//Добавим в таблицу общий заголовок

PdfPCell cell2 = new PdfPCell(new Phrase("Груз ", font));

cell2.Colspan = 6;

cell2.HorizontalAlignment = 1;

//Убираем границу первой ячейки, чтобы балы как заголовок

cell2.Border = 0;

table2.AddCell(cell2);

cell2 = new PdfPCell(new Phrase(new Phrase("Вес", font)));

cell2.BackgroundColor = iTextSharp.text.BaseColor.LIGHT\_GRAY;

table2.AddCell(cell2);

cell2 = new PdfPCell(new Phrase(new Phrase("Длина", font)));

cell2.BackgroundColor = iTextSharp.text.BaseColor.LIGHT\_GRAY;

table2.AddCell(cell2);

cell2 = new PdfPCell(new Phrase(new Phrase("Высота", font)));

cell2.BackgroundColor = iTextSharp.text.BaseColor.LIGHT\_GRAY;

table2.AddCell(cell2);

cell2 = new PdfPCell(new Phrase(new Phrase("Ширина", font)));

cell2.BackgroundColor = iTextSharp.text.BaseColor.LIGHT\_GRAY;

table2.AddCell(cell2);

cell2 = new PdfPCell(new Phrase(new Phrase("Оплата получателем", font)));

cell2.BackgroundColor = iTextSharp.text.BaseColor.LIGHT\_GRAY;

table2.AddCell(cell2);

cell2 = new PdfPCell(new Phrase(new Phrase("Цена", font)));

cell2.BackgroundColor = iTextSharp.text.BaseColor.LIGHT\_GRAY;

table2.AddCell(cell2);

table2.AddCell(new Phrase(order.Weight.ToString(), font));

table2.AddCell(new Phrase(order.Length.ToString(), font));

table2.AddCell(new Phrase(order.Height.ToString(), font));

table2.AddCell(new Phrase(order.Width.ToString(), font));

string str = "нет";

if (order.Who\_pay)

{

str = "да";

}

table2.AddCell(new Phrase(str, font));

table2.AddCell(new Phrase(order.Price.ToString(), font));

//Добавляем таблицу в документ

DC.Add(table2);

DC.Add(new Paragraph("Я подтверждаю, что информация в накладной является полной и точной. С основными условиями пересылки я ознакомлен(а).", font));

DC.SetMargins(100, 10, 10, 10);

DC.Add(new Paragraph("Подпись заказчика\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", font));

DC.SetMargins(100, 10, 10, 10);

DC.Add(new Paragraph("Подпись курьера\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", font));

DC.SetMargins(100, 10, 100, 10);

iTextSharp.text.Image pic = iTextSharp.text.Image.GetInstance(GET("https://api.qrserver.com/v1/create-qr-code/?size=150x150&charset-source=ISO-8859-1", "data=http://192.168.1.8:3000/Orders/Details/" + order.OrderId));

DC.Add(pic);

DC.Close();

return RedirectToAction("Index", "LK");

} private static byte[] GET(string Url, string Data)

{

WebRequest req = WebRequest.Create(Url + "&" + Data);

byte[] bytes;

using (HttpWebResponse res = (HttpWebResponse)req.GetResponse())

using (Stream responseStream = res.GetResponseStream())

using (MemoryStream mstream = new MemoryStream())

{

responseStream.CopyTo(mstream);

bytes = mstream.ToArray();

}

return bytes;

}

//Сохранение отчета в xls

[HttpPost]

public ActionResult Save(List<Order> order )

{

List<Order> model = (List<Order>)TempData["FullModel"];

string file = Environment.GetFolderPath(Environment.SpecialFolder.Desktop) + "/newdoc1.xls";

Workbook workbook = new Workbook();

Worksheet worksheet = new Worksheet("Отчет");

worksheet.Cells[0, 0] = new Cell("Отправитель");

worksheet.Cells[0, 1] = new Cell("Получатель");

worksheet.Cells[0, 2] = new Cell("Адрес отправителя");

worksheet.Cells[0, 3] = new Cell("Район отправителя");

worksheet.Cells[0, 4] = new Cell("Адрес получателя");

worksheet.Cells[0, 5] = new Cell("Район получателя");

worksheet.Cells[0, 6] = new Cell("Номер отправителя");

worksheet.Cells[0, 7] = new Cell("Номер получателя");

worksheet.Cells[0, 8] = new Cell("Дата");

worksheet.Cells[0, 9] = new Cell("Время");

worksheet.Cells[0, 10] = new Cell("Статус");

worksheet.Cells[0, 11] = new Cell("Вес");

worksheet.Cells[0, 12] = new Cell("Длина");

worksheet.Cells[0, 13] = new Cell("Ширина");

worksheet.Cells[0, 14] = new Cell("Высота");

worksheet.Cells[0, 15] = new Cell("Негабаритный");

worksheet.Cells[0, 16] = new Cell("Хрупкий");

worksheet.Cells[0, 17] = new Cell("Оплата отправителем");

worksheet.Cells[0, 18] = new Cell("Цена");

worksheet.Cells[0, 19] = new Cell("ID пользователя");

for (int i = 0; i <= model.Count-1; i++)

{

worksheet.Cells[i + 1, 0] = new Cell(model[i].Sender);

worksheet.Cells[i + 1, 1] = new Cell(model[i].Receiver);

worksheet.Cells[i + 1, 2] = new Cell(model[i].Address\_Sender);

worksheet.Cells[i + 1, 3] = new Cell(model[i].Area\_Sender.ToString());

worksheet.Cells[i + 1, 4] = new Cell(model[i].Address\_Receiver);

worksheet.Cells[i + 1, 5] = new Cell(model[i].Area\_Receiver.ToString());

worksheet.Cells[i + 1, 6] = new Cell(model[i].Phone\_Sender);

worksheet.Cells[i + 1, 7] = new Cell(model[i].Phone\_Receiver);

worksheet.Cells[i + 1, 8] = new Cell(model[i].Date, @"YYYY-MM-DD");

worksheet.Cells[i + 1, 9] = new Cell(model[i].Time, @"hh:mm");

worksheet.Cells[i + 1, 10] = new Cell(model[i].Status);

worksheet.Cells[i + 1, 11] = new Cell((decimal)model[i].Weight);

worksheet.Cells[i + 1, 12] = new Cell((decimal)model[i].Length);

worksheet.Cells[i + 1, 13] = new Cell((decimal)model[i].Width);

worksheet.Cells[i + 1, 14] = new Cell((decimal)model[i].Height);

worksheet.Cells[i + 1, 15] = new Cell((bool)model[i].Big);

worksheet.Cells[i + 1, 16] = new Cell((bool)model[i].Fragile);

worksheet.Cells[i + 1, 17] = new Cell((bool)model[i].Who\_pay);

worksheet.Cells[i + 1, 18] = new Cell((int)model[i].Price);

worksheet.Cells[i + 1, 19] = new Cell((int)model[i].CustomerId);

}

workbook.Worksheets.Add(worksheet);

workbook.Save(file);

return RedirectToAction("Index");

}

public ActionResult Distribute(int id = 4)

{

var elem = from s in db.Orders

select s;

var cour = from s in db.Couriers

select s;

List<SelectListItem> item = new List<SelectListItem>();

//фильтрация по районам

item.Add(new SelectListItem { Text = "Ленинский", Value = "0" });

item.Add(new SelectListItem { Text = "Засвияжский", Value = "1" });

item.Add(new SelectListItem { Text = "Заволжский", Value = "2" });

item.Add(new SelectListItem { Text = "Железнодорожный", Value = "3" });

item.Add(new SelectListItem { Text = "все", Value = "4", Selected = true });

DateTime week = DateTime.Now.AddDays(-7);

DateTime month = DateTime.Now.AddMonths(-1);

DateTime now = DateTime.Now;

ViewBag.id = item;

switch (id)

{

case 0:

elem = elem.Where(s => s.Area\_Sender == Areas.Ленинский);

cour = cour.Where(x => x.Area == Areas.Ленинский);

break;

case 1:

elem = elem.Where(s => s.Area\_Sender == Areas.Засвияжский);

cour = cour.Where(x => x.Area == Areas.Засвияжский);

break;

case 2:

elem = elem.Where(s => s.Area\_Sender == Areas.Заволжский);

cour = cour.Where(x => x.Area == Areas.Заволжский);

break;

case 3:

elem = elem.Where(s => s.Area\_Sender == Areas.Железнодорожный);

cour = cour.Where(x => x.Area == Areas.Железнодорожный);

break;

case 4: elem = db.Orders; break;

}

ViewData["MyList"] = cour.ToList();

elem = elem.Where(x => x.CourierId == null);

return View(elem.ToList());

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}