МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

кафедра «Вычислительная техника».

**Лабораторная работа №3**

**тема: «Создание диалоговых форм c помощью шаблонов»**

по дисциплине: «Корпоративные системы»

Выполнил:

студент 4 курса, гр. ИВТВМбд-41

Захарычев Никита Алексеевич.

Проверил:

доцент кафедры ВТ,

кандидат технических наук

Мартынов Антон Иванович.

г. Ульяновск, 2018

**Задание**

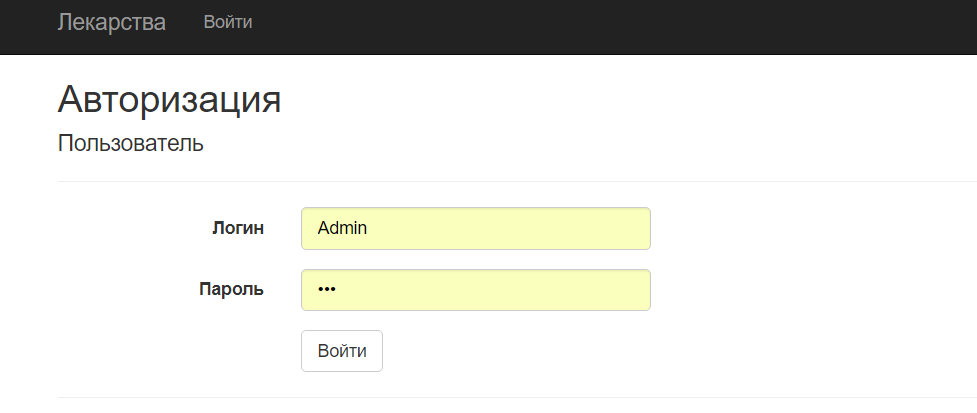
Используя результаты лабораторной работы №1 и №2 расширить функциональность существующей системы, добавив возможность добавления, редактирования и удаления данных, содержащихся в таблицах, то есть осуществлять вход в *режим администрирования*. При этом должны удовлетворяться следующие требования:

1. Для входа в режим администрирования, необходимо реализовать механизм авторизации, с помощью которого система будет запрашивать логин и пароль пользователя, и сравнивать их со значениями, хранящимися в базе данных.
2. В режиме администрирования просмотр таблиц расширен функциональными элементами *добавления новой записи*, *редактирования* и *удаления*, при этом должны сохраняться все возможности, реализованные в лабораторной работе №2. Пример шаблона, реализующего вывод в режиме администрирования, приведен на Рис. 1.
3. При добавлении или редактировании записи должен осуществляться переход в режим формы, которая позволяет добавить/изменить любые данные в таблице. Пример диалога для вышеприведенной таблицы приведен на Рис. 2.
4. При формировании диалоговых форм должны удовлетворяться следующие требования:
   * Поля, значения которых берутся из других таблиц, должны быть реализованы в виде выпадающих списков
   * Для полей типа «Дата» или «Время» используются разделенные выпадающие списки («День»-«Месяц»-«Год» или «Часы»-«Минуты»-«Секунды»).
   * Для полей, представляющих собой значения типа «Да/нет» и т.п. должны использоваться элементы типа «radio».
   * Для полей, которые используются для хранения изображений, должна быть реализована возможность загрузки изображений на сервер.
5. Поля, обязательные для заполнения должны быть отмечены символом «\*».
6. Если пользователь неправильно заполнил форму, то выводится сообщение об ошибке, которое содержит перечень полей, в которых допущены ошибки и причину ошибки. При этом значения тех полей которые заполнены верно, должны сохраняться.
7. После успешного добавления/изменения данных должен осуществляться автоматический переход на тот список, откуда была инициирована операция.
8. При удалении должно выдаваться предупреждение, защищающее от нечаянного удаления. Например, такое:
9. Если пользователь пытается удалить данные, которые ссылаются на другую таблицу, то также должно выводиться соответствующее сообщение.

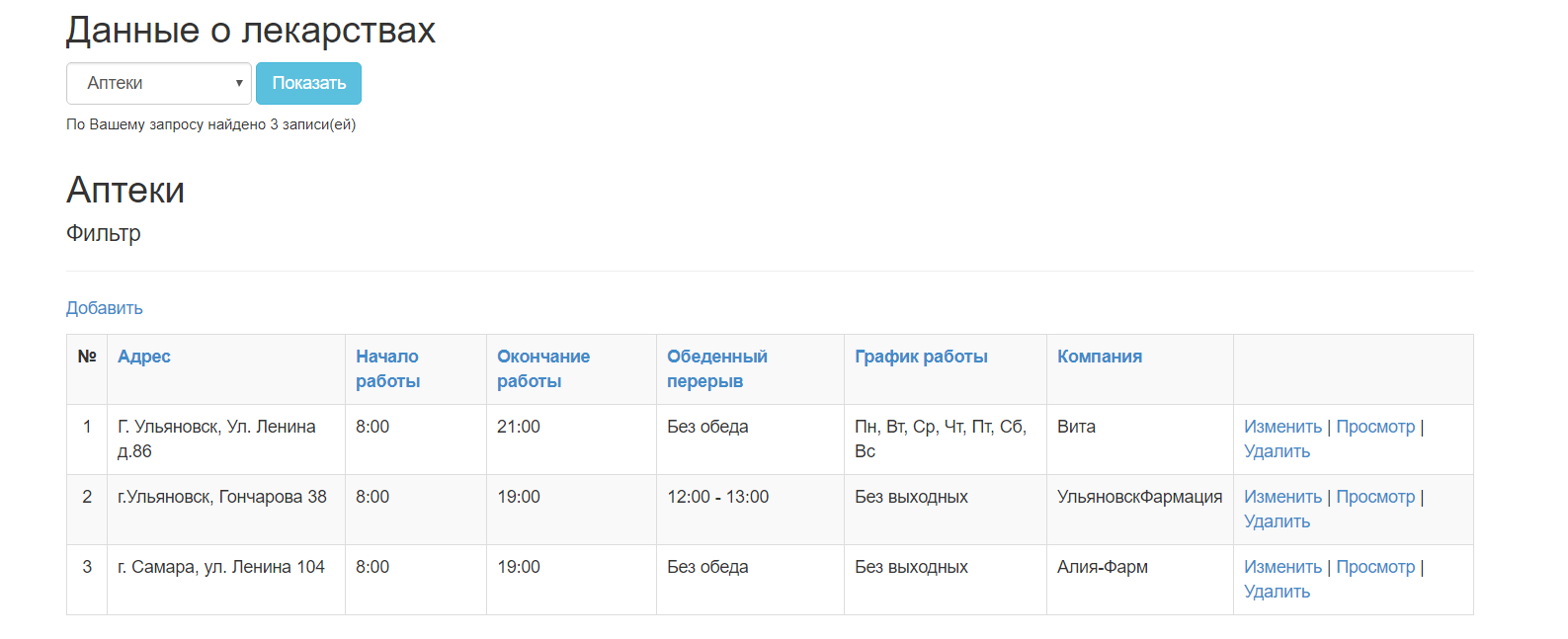
**Ход работы**

1. Для входа в режим администрирования был реализован механизм авторизации суть которого заключалась в проверке введенных данных пользователем с данными соответствующих администратору. В результате успешной проверки пользователь сохраняется в сессии.

Форма для авторизации.

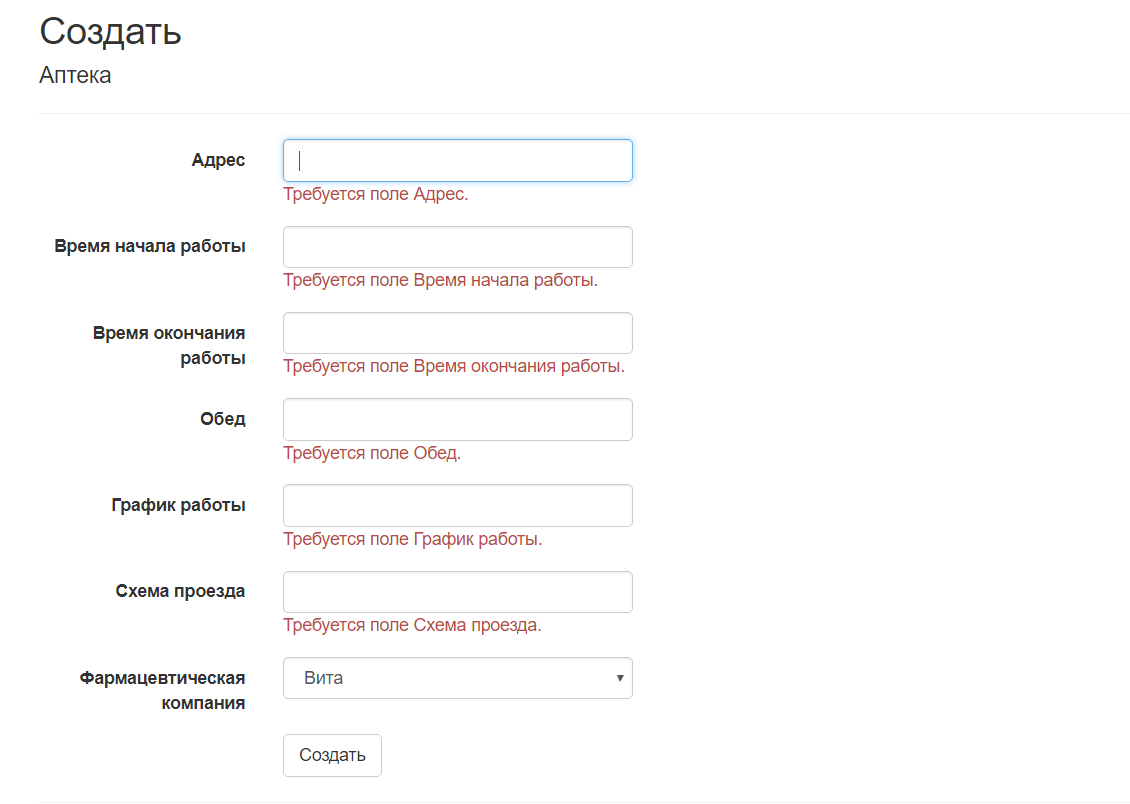


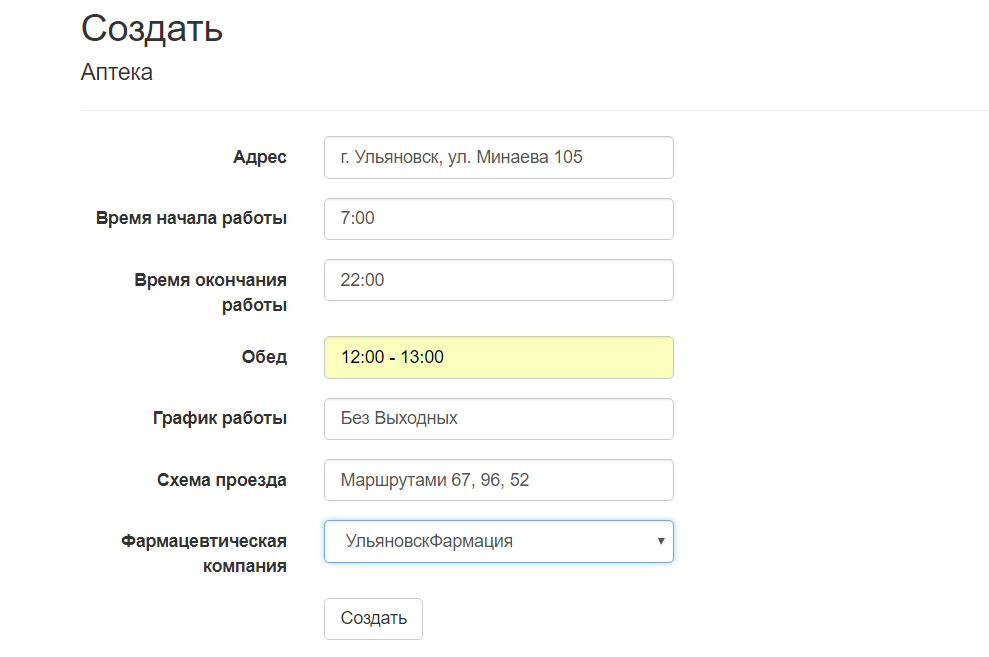
В результате успешной авторизации, администратору становится доступны добавление, просмотр, изменение, удаление данных соответствующих таблиц.



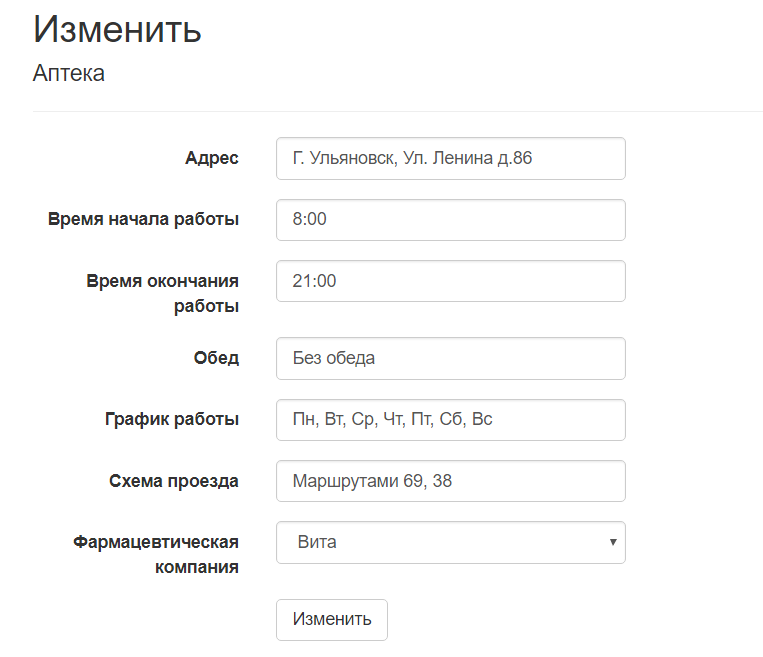
Формы для добавления и редактирования содержат обязательные поля для заполнения, в случае отсутствия значений в этих полях форма становится не валидной и появляются соответствующие сообщения.

Форма для добавления аптеки.

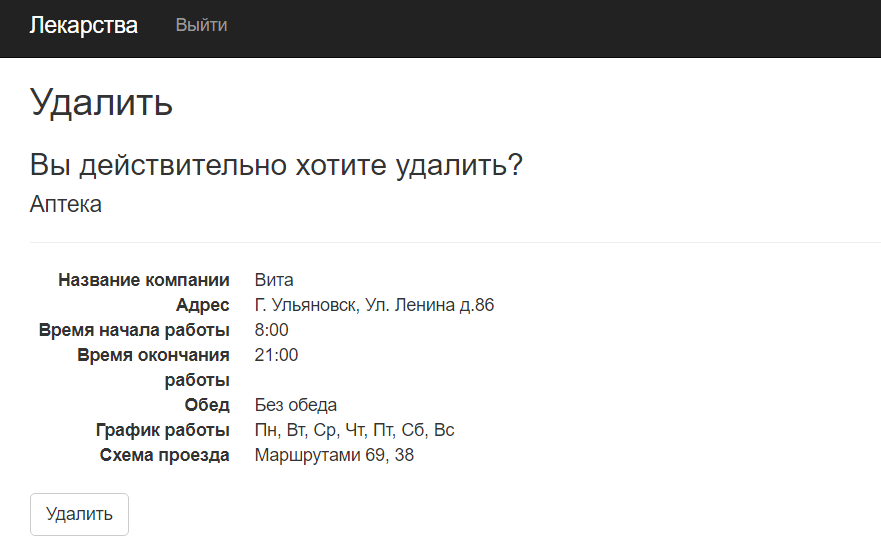




Форма для изменения аптеки.



Диалоговое представление, требующее подтверждения на удаления аптеки.



**Листинг.**

Контроллер реализующий логику авторизации пользователя.

namespace AptekaApteka.Controllers

{

public class AccountsController : Controller

{

private AptekaContext db = new AptekaContext();

public ActionResult Auth()

{

if (Session["Admin"] != null)

{

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

}

return View("Create");

}

[HttpPost]

public ActionResult Auth([Bind(Include = "Login,Password")] Account account)

{

if (Session["Admin"] != null)

{

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

}

if (ModelState.IsValid)

{

if (account.Login == "Admin" && account.Password == "123")

{

Session["Admin"] = new { name = account.Login };

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

}

ModelState.AddModelError("AuthError", new Exception("Не верные логин или пароль"));

}

return View("Create", account);

}

[HttpGet]

public ActionResult Logout()

{

if (Session["Admin"] == null)

{

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

}

Session.Remove("Admin");

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

}

}

}

Представление формы добавления записи в таблицу аптеки.

@model AptekaApteka.Models.Pharmacy

@{

ViewBag.Title = "Create";

}

<h2>Создать</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<h4>Аптека</h4>

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

<div class="form-group">

@Html.LabelFor(model => model.Addres, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Addres, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Addres, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.StartWork, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.StartWork, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.StartWork, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.EndWork, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.EndWork, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.EndWork, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.LunchTime, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.LunchTime, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.LunchTime, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Schedule, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Schedule, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Schedule, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.drivingDirections, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.drivingDirections, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.drivingDirections, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.CompanyID, "Фармацевтическая компания", htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownList("CompanyID", null, htmlAttributes: new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.CompanyID, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Создать" class="btn btn-default" />

</div>

</div>

</div>

}

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

}

Представление формы редактирования записи из таблицы аптеки.

@model AptekaApteka.Models.Pharmacy

@{

ViewBag.Title = "Edit";

}

<h2>Изменить</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<h4>Аптека</h4>

<hr />

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

@Html.HiddenFor(model => model.PharmacyID)

<div class="form-group">

@Html.LabelFor(model => model.Addres, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Addres, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Addres, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.StartWork, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.StartWork, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.StartWork, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.EndWork, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.EndWork, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.EndWork, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.LunchTime, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.LunchTime, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.LunchTime, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Schedule, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Schedule, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Schedule, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.drivingDirections, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.drivingDirections, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.drivingDirections, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.CompanyID, "Фармацевтическая компания", htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownList("CompanyID", null, htmlAttributes: new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.CompanyID, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Изменить" class="btn btn-default" />

</div>

</div>

</div>

}

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

}

Диалоговое представление для подтверждения удаления записи из таблицы аптеки.

@model AptekaApteka.Models.Pharmacy

@{

ViewBag.Title = "Delete";

}

<h2>Удалить</h2>

<h3>Вы действительно хотите удалить?</h3>

<div>

<h4>Аптека</h4>

<hr />

<dl class="dl-horizontal">

<dt>

@Html.DisplayNameFor(model => model.Company.Name)

</dt>

<dd>

@Html.DisplayFor(model => model.Company.Name)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Addres)

</dt>

<dd>

@Html.DisplayFor(model => model.Addres)

</dd>

<dt>

@Html.DisplayNameFor(model => model.StartWork)

</dt>

<dd>

@Html.DisplayFor(model => model.StartWork)

</dd>

<dt>

@Html.DisplayNameFor(model => model.EndWork)

</dt>

<dd>

@Html.DisplayFor(model => model.EndWork)

</dd>

<dt>

@Html.DisplayNameFor(model => model.LunchTime)

</dt>

<dd>

@Html.DisplayFor(model => model.LunchTime)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Schedule)

</dt>

<dd>

@Html.DisplayFor(model => model.Schedule)

</dd>

<dt>

@Html.DisplayNameFor(model => model.drivingDirections)

</dt>

<dd>

@Html.DisplayFor(model => model.drivingDirections)

</dd>

</dl>

@using (Html.BeginForm()) {

@Html.AntiForgeryToken()

<div class="form-actions no-color">

<input type="submit" value="Удалить" class="btn btn-default" />

</div>

}

</div>

Контроллер содержащий реализацию модулей добавления, редактирования, просмотра и удаления конкретной записи из таблицы Аптеки.

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 AptekaApteka.DAL;

using AptekaApteka.Models;

namespace AptekaApteka.Controllers

{

public class PharmaciesController : Controller

{

private AptekaContext db = new AptekaContext();

// GET: Pharmacies

public ActionResult Index()

{

var pharmacies = db.Pharmacies.Include(p => p.Company);

return View(pharmacies.ToList());

}

// GET: Pharmacies/Details/5

public ActionResult Details(int? id)

{

if (Session["Admin"] == null)

{

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

}

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Pharmacy pharmacy = db.Pharmacies.Find(id);

if (pharmacy == null)

{

return HttpNotFound();

}

return View(pharmacy);

}

// GET: Pharmacies/Create

public ActionResult Create()

{

if (Session["Admin"] == null)

{

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

}

ViewBag.CompanyID = new SelectList(db.Companies, "CompanyID", "Name");

return View();

}

// POST: Pharmacies/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 = "PharmacyID,Addres,StartWork,EndWork,LunchTime,Schedule,drivingDirections,CompanyID")] Pharmacy pharmacy)

{

if (ModelState.IsValid)

{

db.Pharmacies.Add(pharmacy);

db.SaveChanges();

return RedirectToAction("Index", "Home", new { table = "Аптеки" });

}

ViewBag.CompanyID = new SelectList(db.Companies, "CompanyID", "Name", pharmacy.CompanyID);

return View(pharmacy);

}

// GET: Pharmacies/Edit/5

public ActionResult Edit(int? id)

{

if (Session["Admin"] == null)

{

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

}

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Pharmacy pharmacy = db.Pharmacies.Find(id);

if (pharmacy == null)

{

return HttpNotFound();

}

ViewBag.CompanyID = new SelectList(db.Companies, "CompanyID", "Name", pharmacy.CompanyID);

return View(pharmacy);

}

// POST: Pharmacies/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 = "PharmacyID,Addres,StartWork,EndWork,LunchTime,Schedule,drivingDirections,CompanyID")] Pharmacy pharmacy)

{

if (ModelState.IsValid)

{

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

db.SaveChanges();

return RedirectToAction("Index", "Home", new { table = "Аптеки" });

}

ViewBag.CompanyID = new SelectList(db.Companies, "CompanyID", "Name", pharmacy.CompanyID);

return View(pharmacy);

}

// GET: Pharmacies/Delete/5

public ActionResult Delete(int? id)

{

if (Session["Admin"] == null)

{

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

}

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Pharmacy pharmacy = db.Pharmacies.Find(id);

if (pharmacy == null)

{

return HttpNotFound();

}

return View(pharmacy);

}

// POST: Pharmacies/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

Pharmacy pharmacy = db.Pharmacies.Find(id);

var preparations = db.Preparations.Where(p => p.PreparationID == id);

foreach(var p in preparations)

{

p.PharmacyID = null;

p.Pharmacy = null;

}

db.Pharmacies.Remove(pharmacy);

db.SaveChanges();

return RedirectToAction("Index", "Home", new { table = "Аптеки" });

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

}

**Вывод**

В ходе выполнения данной лабораторной работы были получены навыки в создании диалоговых форм с помощью шаблонов в Web-приложениях.