**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра ВТ**

**ОТЧЕТ**

**по лабораторной работе №4**

**по дисциплине «Основы разработки корпоративных систем на платформе .NET»**

**Тема: Разработка программного интерфейса для взаимодействия с приложением**

Студент гр. 6306 Гордиенко М. Е.

Преподаватель Пешехонов К. А.

Санкт-Петербург

2020

**Цель работы**

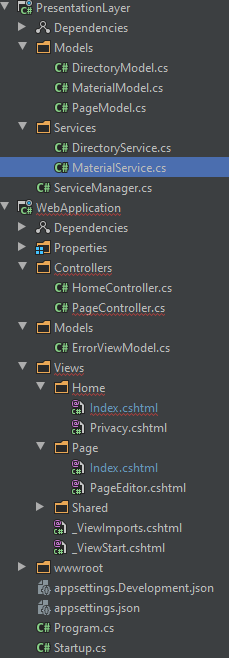
Разработка программного интерфейса Web-приложения.

**Задачи**

Реализация программного интерфейса приложения

**Описание работы и примеры кода**

**Структура слоя программного интерфейса**

****

**Файл DirectoryService.cs**

using System;  
using System.Collections.Generic;  
using System.Diagnostics.CodeAnalysis;  
using System.Linq;  
using BusinessLayer;  
using DataLayer.Entityes;  
using PresentationLayer.Models;  
  
namespace PresentationLayer.Services **{** [SuppressMessage("ReSharper", "InconsistentNaming")]  
 public class DirectoryService {  
 private readonly DataManager \_dataManager;  
 private readonly MaterialService \_materialService;  
  
 public DirectoryService(DataManager dataManager) {  
 \_dataManager = dataManager;  
 \_materialService = new MaterialService(dataManager);  
 }  
  
 public List<DirectoryViewModel> GetDirectoriesList() {  
 var dirs = \_dataManager.DirectoryRepository.GetAllDirectories();  
  
 return dirs.Select(item => DirectoryDBToViewModelById(item.Id)).ToList();  
 }  
  
 public DirectoryViewModel DirectoryDBToViewModelById(int directoryId) {  
 var directory = \_dataManager.DirectoryRepository.GetDirectoryById(directoryId);  
 List<MaterialViewModel> materialViewModels = null;  
 try {  
 materialViewModels = directory  
 .Materials.Select(item => \_materialService.MaterialDBModelToView(item.Id))  
 .ToList();  
 } catch(NullReferenceException){};  
  
 return new DirectoryViewModel { Directory = directory, Materials = materialViewModels};  
 }  
  
 public DirectoryEditModel GetDirectoryEditModel(int directoryId = 0) {  
 if (directoryId == 0) {  
 return new DirectoryEditModel();  
 }  
  
 var directoryDBModel = \_dataManager.DirectoryRepository.GetDirectoryById(directoryId);  
 return new DirectoryEditModel {  
 Id = directoryDBModel.Id,  
 Title = directoryDBModel.Title,  
 Html = directoryDBModel.Html  
 };  
 }  
  
 public DirectoryViewModel SaveDirectoryEditModelToDb(DirectoryEditModel directoryEditModel) {  
 var directoryDbModel = directoryEditModel.Id != 0   
 ? \_dataManager.DirectoryRepository.GetDirectoryById(directoryEditModel.Id)   
 : new Directory();  
 directoryDbModel.Title = directoryEditModel.Title;  
 directoryDbModel.Html = directoryEditModel.Html;  
 directoryDbModel.Id = \_dataManager.DirectoryRepository.SaveDirectory(directoryDbModel);  
 return DirectoryDBToViewModelById(directoryDbModel.Id);  
 }  
  
 public void DeleteDirectoryEditModelFromDb(DirectoryEditModel directoryEditModel) {  
 if (directoryEditModel.Id != 0) {  
 \_dataManager.DirectoryRepository.DeleteDirectory(\_dataManager  
 .DirectoryRepository  
 .GetDirectoryById(directoryEditModel.Id));  
 }  
 }  
 }  
**}**

**Файл PageController.cs**

using BusinessLayer;  
using DataLayer.Enums;  
using Microsoft.AspNetCore.Mvc;  
using PresentationLayer;  
using PresentationLayer.Models;  
  
namespace WebApplication.Controllers **{** public class PageController : Controller {  
 private readonly ServiceManager \_serviceManager;  
  
 public PageController(DataManager dataManager) {  
 \_serviceManager = new ServiceManager(dataManager);  
 }  
  
 // GET  
 public IActionResult Index(int pageId, PageEnums.PageType pageType) {  
 PageViewModel viewModel = pageType switch {  
 PageEnums.PageType.*Directory* => \_serviceManager.DirectoryService.DirectoryDBToViewModelById(pageId),  
 PageEnums.PageType.*Material* => \_serviceManager.MaterialService.MaterialDBModelToView(pageId),  
 \_ => null  
 };  
  
 ViewBag.PageType = pageType;  
 return View(viewModel);  
 }  
  
 [HttpGet]  
 public IActionResult PageEditor(int pageId, PageEnums.PageType pageType, int directoryId) {  
 PageEditModel editModel = null;  
 switch (pageType) {  
 case PageEnums.PageType.*Directory*: {  
 editModel = \_serviceManager.DirectoryService.GetDirectoryEditModel(pageId);  
 break;  
 }  
 case PageEnums.PageType.*Material*: {  
 editModel = \_serviceManager.MaterialService.GetMaterialEditModel(pageId, directoryId);  
 break;  
 }  
 }  
  
 ViewBag.PageType = pageType;  
 return View(editModel);  
 }  
  
 [HttpPost]  
 public IActionResult SaveDirectory(DirectoryEditModel model) {  
 var directoryId = \_serviceManager.DirectoryService.SaveDirectoryEditModelToDb(model).Directory.Id;  
 return RedirectToAction("Index", "Page",  
 new {pageId = directoryId, pageType = PageEnums.PageType.*Directory*});  
 }  
  
 [HttpPost]  
 public IActionResult DeleteDirectory(DirectoryEditModel model) {  
 \_serviceManager.DirectoryService.DeleteDirectoryEditModelFromDb(model);  
 return RedirectToAction("Index", "Home");  
 }  
   
 [HttpPost]  
 public IActionResult SaveMaterial(MaterialEditModel model) {  
 var materialId = \_serviceManager.MaterialService.SaveMaterialEditModelToDb(model).Material.Id;  
 return RedirectToAction("Index", "Page",  
 new {pageId = materialId, pageType = PageEnums.PageType.*Material*});  
 }  
  
 [HttpPost]  
 public IActionResult DeleteMaterial(MaterialEditModel model) {  
 \_serviceManager.MaterialService.DeleteMaterialEditModelFromDb(model);  
 return RedirectToAction("Index", "Home");  
 }  
 }  
**}**

**Файл PageEditor.cshtml**

@using PresentationLayer.Models;  
@model PageEditModel  
  
@{  
 Layout = "\_Layout";  
 ViewData["Title"] = "Edit Page";  
}  
  
@switch (Model) {  
 case DirectoryEditModel \_: {  
 var currentModel = (DirectoryEditModel)Model;  
 *<*form action="/Page/SaveDirectory" method="post"*>  
 <*input type="hidden" value="@ViewBag.PageType" name="PageType" */>  
 <*h1*>*Edit Directory*</*h1*>  
 <*input name="Id" type="hidden" value="@currentModel.Id" */>* Directory Name*<*br*/>  
 <*input class="titleInput" name="Title" value="@currentModel.Title" */><*br*/>* Directory description*<*br*/>  
 <*textarea class="contentInput" name="Html"*>*@currentModel.Html*</*textarea*><*br*/>  
 <*button type="submit"*>*Save Directory*</*button*>  
 </*form*>  
 <*form action="/Page/DeleteDirectory" method="post"*>  
 <*input type="hidden" value="@ViewBag.PageType" name="PageType" */>  
 <*input name="Id" type="hidden" value="@currentModel.Id" */>  
 <*button class="deleteButton" type="submit"*>*Delete Directory*</*button*>  
 </*form*>* break;  
 }  
 case MaterialEditModel \_: {  
 var currentModel = (MaterialEditModel)Model;  
 *<*form action="/Page/SaveMaterial" method="post"*>  
 <*input type="hidden" value="@ViewBag.PageType" name="PageType" */>  
 <*h1*>*Edit Material*</*h1*>  
 <*input name="Id" type="hidden" value="@currentModel.Id" */>  
 <*input name="DirectoryId" type="hidden" value="@currentModel.DirectoryId" */>* Material Name*<*br*/>  
 <*input class="titleInput" name="Title" value="@currentModel.Title" */><*br*/>* Material Content*<*br*/>  
 <*textarea class="contentInput" name="Html"*>*@currentModel.Html*</*textarea*><*br*/>  
 <*button type="submit"*>*Save Material*</*button*>  
 </*form*>  
 <*form action="/Page/DeleteMaterial" method="post"*>  
 <*input type="hidden" value="@ViewBag.PageType" name="PageType" */>  
 <*input name="Id" type="hidden" value="@currentModel.Id" */>  
 <*input name="DirectoryId" type="hidden" value="@currentModel.DirectoryId" */>  
 <*button type="submit"*>*Delete Material*</*button*>  
 </*form*>* break;  
 }  
}

**Вывод**

В ходе выполнения работы был создан программный интерфейс Web-приложения.