Министерство науки и высшего образования Российской Федерации

ФГАОУ ВО «УрФУ имени первого Президента России Б.Н. Ельцина»

Институт новых материалов и технологий

Департамент металлургии и металловедения

Кафедра «Теплофизика и информатика в металлургии»

Оценка работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_

К.А. Щипанов

**Разработка программного обеспечения «Система контроля доступа»**

ОТЧЕТ

Вид практики: Производственная практика

Тип практики: Практика по получению профессиональных умений   
и опыта профессиональной деятельности

Направление подготовки 09.03.02 Информационные системы и технологии.

Образовательная программа 09.03.02/03.01 Информационные системы   
и технологии в металлургии

Руководитель практики

от предприятия (организации) К.А. Щипанов

Подпись ФИО руководителя

Студент Е.Ю.Мазлова

Подпись ФИО студента

Группа НМТ-493907

Екатеринбург 2022

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc110866044)

[ВВЕДЕНИЕ 3](#_Toc110866045)

[1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 4](#_Toc110866046)

[1.1 Актуальность работы 4](#_Toc110866047)

[1.2 Описание работы СКУД 4](#_Toc110866048)

[1.3 Обзор существующих программных продуктов 5](#_Toc110866049)

[1.4 Сравнительный анализ программных обеспечений СКУД-систем 8](#_Toc110866050)

[2 ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ 10](#_Toc110866051)

[2.1 Постановка задачи 10](#_Toc110866052)

[2.2 Определение требований пользователей к информационной системе 10](#_Toc110866053)

[3 РАЗРАБОТКА И ОПИСАНИЕ БАЗЫ ДАННЫХ 12](#_Toc110866054)

[3.1 Проектирование концептуальной модели базы данных в сервисе ERDPlus 12](#_Toc110866055)

[3.2 Разработка и описание даталогической модели базы данных на платформе Microsoft SQL Server 2014 13](#_Toc110866056)

[3.3 Автоматизация наполнения базы тестовыми данными с использованием сервиса Microsoft SQL Server Integration Services 18](#_Toc110866057)

[4 РАЗРАБОТКА И ОПИСАНИЕ КЛИЕНТСКОГО ПРИЛОЖЕНИЯ 20](#_Toc110866058)

[4.1 Управление версиями проекта на основе сервиса GitHub 20](#_Toc110866059)

[4.2 Разработка клиентского приложения 20](#_Toc110866060)

[4.3 Описание работы приложения 24](#_Toc110866061)

[ЗАКЛЮЧЕНИЕ 28](#_Toc110866062)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 29](#_Toc110866063)

[ПРИЛОЖЕНИЕ А ФРАГМЕНТ ЛИСТИНГА ПРОГРАММНОГО КОДА ПОЛЬЗОВАТЕЛЬСКОГО ПРИЛОЖЕНИЯ 30](#_Toc110866064)

ВВЕДЕНИЕ

Данный отчет составлен по результатам прохождения производственной практики в период с 4.07.2022 по 31.07.2022.

Место прохождения практики – ФГАОУ ВО УрФУ, кафедра «Теплофизика и информатика в металлургии», г. Екатеринбург.

Целью прохождения производственной практики является получение теоретических знаний, умений и навыков, а также их применений в практической деятельности организации.

Во время прохождения практики была реализована информационная система контроля передвижения сотрудников предприятия. Информационная система предполагает разработку схемы базы данных с помощью сервиса ERDPlus и ее последующую реализацию в пакете SQL Management Studio 2014. Далее было разработано пользовательское WEB-приложение в среде разработки Microsoft Visual Studio 2019 с использованием ASP.NET Core Web Application (MVC) на платформе .Net 5.0, основными потребителями которого являются сотрудники охранного предприятия и их непосредственное руководство, а также администратор системы.

1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

* 1. **Актуальность работы**

В настоящее время наиболее актуальным вопросом всех современных предприятий является безопасность сотрудников и охрана труда. Одним из важнейших составляющих безопасности является контроль передвижения сотрудников по средствам электронно-пропускной системы (СКУД), а также видеонаблюдения на территории предприятия.

Система контроля и управления доступом, СКУД — совокупность программно-аппаратных технических средств контроля и средств управления, имеющих целью ограничение и регистрацию входа-выхода объектов (людей, транспорта) на заданной территории через «точки прохода»: двери, ворота, КПП. Основная задача систем – управление доступом на заданную территорию, включая также:

• ограничение доступа на заданную территорию;

• идентификацию лица, имеющего доступ на заданную территорию.

Дополнительные задачи:

• учёт рабочего времени;

• ведение базы персонала / посетителей;

• интеграция с системой безопасности (с системой видеонаблюдения, пожарно-охранной сигнализацией и системами оповещения).

На данный момент рынок программных обеспечений для СКУД-систем достаточно богат, однако не все программы удовлетворяют функциональным требованиям заказчиков, либо наоборот содержат множество излишних функций, замедляющих работу системы. Поэтому вопрос разработки простой, но при этом функциональной системы остается открытым.

* 1. **Описание работы СКУД**

У входа на объект устанавливается устройство аутентификации, например, считыватель карты доступа, и устройство, которое может по электрическому сигналу блокировать/разблокировать вход (дверь с электромагнитным замком, ворота, шлагбаум и т.д.). Контроллер, к которому они подключены, по локальной сети обменивается данными с управляющим компьютером и сервером, на котором хранится база данных пользователей.

Пользователь подносит свою карту к считывателю, который передает его идентификационные данные на контроллер. Контроллер сверяет полученные идентификационные данные с теми, что хранятся в базе данных, и проверяет права доступа к данному объекту. В зависимости от результатов проверки, владельцу карты либо разрешается доступ на объект - на электромагнитный замок или турникет подается сигнал на открытие, или запрещается - замок остается закрытым. Шлагбаумы и электромеханические замки, автоматические ворота и турникеты, ростовые кабины и приводы ворот предназначены для ограничения доступа к охраняемому объекту. Эти устройства открываются или блокируются по сигналу, приходящему с контроллера.

Для регистрации нового пользователя в биометрической СКУД используется или один из считывателей системы, или сканер, подключенный к рабочему месту сотрудника, отвечающего за ввод новых пользователей. Данные нового пользователя и информация о том, к каким помещениям ему разрешен доступ, сразу же попадают в базу данных СКУД.

Каждый контроллер определяет логику работы одной или нескольких дверей (ворот, шлагбаумов). На вход контроллера поступает сигнал с устройства аутентификации, а также (возможно) сигналы с датчиков ОПС и детекторов движения видеокамер. В соответствии с логикой, которая заложена в контроллер при настройке, он дает исполнительным устройствам сигнал на открытие/блокировку.

В зависимости от сложности системы и требуемого уровня безопасности, можно заложить в контроллер различные алгоритмы работы. Например, можно ввести для определенной группы пользователей ограничение доступа по времени: доступ в помещение будет разрешен только в определенный промежуток времени. Еще один распространенный алгоритм – доступ в помещение только в присутствии одного или более лиц с более высоким уровнем доступа.

* 1. **Обзор существующих программных продуктов**

В обзоре представлены следующие программные продукты:

* ZkTeco
* ControlGate
* PERCo
* RusGuard
* Parsec
* HID Global
* Sigur

**«ZkTeco»**

Китайская компания «ZkTeco Co., Ltd» широко известна своей направленностью на биометрические методы верификации и предоставляет услуги для крупного и среднего бизнеса. Предприятие владеет более 90 филиалами по всему миру, в т.ч. России и странах СНГ.

Программное обеспечение базируется на облачной платформе ZKTeco + Smart Office. Оно поддерживает биометрическую идентификацию, видеонаблюдение, а также интеллектуальную систему входа и выхода «Человек + автомобиль + объект». ПО ZKTime.Net V3.0 предназначено для работы с системой учета рабочего времени компании, включает в себя возможности распознавания лиц, отпечатков и гибридную биометрию. На основе получаемой информации, программа способна создавать отчеты и платежные ведомости. Кроме этого, учитывая реалии времени, в компании разрабатывают методы бесконтактной идентификации COVID-19, что также интегрируется в программное обеспечение через соответствующие модули.

**«ControlGate»**

Российская компания «КонтролГейт» специализируется на программировании и разработке систем безопасности. ПО ControlGate включает в себя полный набор модулей для проведения наблюдения, контроля оборудования, управления доступом, а также анализа и сбора информации. Система безопасности объекта под управлением ПО ControlGate способна работать с широким спектром идентификаторов, в т.ч. и биометрических. При необходимости, есть возможности комбинировать между собой несколько типов идентификаторов. Например, биометрический терминал и считыватель ПИН-кода. Масштабирование системы осуществляется за счет подключения дополнительных модулей.

Основными плюсом софта является его кроссплатформенность, т.е. возможность работать на различных операционных системах (Windows, Linux, MacOs). Это позволяет интегрироваться с оборудованием таких производителей как: Hikvision, Dahua, ZKTeco, IronLogik.

СКУД ControlGate построена на принципах модульности. Существует основной базовый блок, к которому при необходимости подключаются дополнительные (информатор, бюро пропусков, СУРВ, обмен данными). Ярким плюсом служит то, что в базовом модуле предустановлены следующие компоненты: непосредственно контроль доступа, видеосистема, генератор отчетов, среда выполнения пользовательских сценариев, планировщик задач.

**«PERCo»**

Компания «PERCo» входит в число ведущих отечественных производителей оборудования в области безопасности. Комплексные ПО СКУД представлено системами PERCo-Web, PERCo-S-20 и PERCo- S-20 «Школа».

PERCo-Web устанавливается на отдельный компьютер и позволяет работать другим пользователям через веб-интерфейс. Система контролирует доступ лиц в здания, разграничение прав входа в помещения и учет времени работы персонала.

Основным назначением ПО PERCo-S-20 является организация комплексной работы систем безопасности предприятия. ПО координирует работу СКУД с системами пожарно-охранной сигнализации и видеонаблюдения.

Система PERCo-S-20 «Школа» разработана специально для установки в учебных заведениях. Она предназначена для недопущения прохода посторонних, ведет учет посещаемости учащихся с одновременным уведомлением родителей о времени прихода и ухода ребенка.

**«RusGuard»**

Компания «РусГард» специализируется на производстве оборудования для СКУД и охранной сигнализации. Программное обеспечение данной компании – софт для полноценной работы систем доступа. Выделяют три версии софта «Standart», «Premium» и «Enterprise». ПО способно решать задачи, как крупного предприятия, так и небольшого офиса. Связь осуществляется по защищенному протоколу HTTPS.

Важной особенностью является то, что покупателю оборудования передается комплект ПО бесплатно. В дальнейшем осуществляется бесплатная техподдержка. В то же время специалисты, отмечают достаточно высокую стоимость самого оборудования, что нивелирует плюсы от бесплатного программного обеспечения.

**«Parsec»**

Разработчиком и производителем данного программного обеспечения является ООО «Диамант групп». Компания занимается реализацией решений в сфере безопасности и идентификации объектов.

Программное обеспечение компании представлено программными модулями PNSoft и PNOffice, которые имеют множество вариаций для различных комбинаций оборудования. ПО PNSoft применяется на СКУД, устанавливаемых на предприятиях, в то время как PNSoft используется в системах доступа небольших объектов и больше ориентирован на учет и контроль персонала.

**«HID Global»**

Компания «HID Global» являет мировым лидером по производству СКУД. Головной офис компании расположен в г.Остин, штат Техас (США).

Программное обеспечение компании имеет весьма широкую направленность и используется в СКУД для физического и логического доступа персонала, а также в сфере выпуска финансовых инструментов. Софт от HID Global используется для регистрации биометрических параметров и аутентификации, выпуска идентификационных карт для банковского и финансового секторов. Также он широко применяется в операционных системах для микрочипов, а также в сфере Internet of Things (IoT).

Вместе с тем при всей разнонаправленности и плюсах разработок HID, в ПО имеется существенный минус – его цена. Впрочем, высокая стоимость программного обеспечения характерна практически для всех продуктов зарубежного производства.

**«Sigur»**

Владельцем данного бренда является российская компания «ПромАвтоматика», которая работает в сфере создания СКУД с 2006 года. За это время, бренд «Sigur» стал весьма известен на российском рынке систем для безопасности.

Программное обеспечение компании имеет модульную архитектуру и позволяет при необходимости увеличивать возможности системы через установку дополнительных модулей. Доступ ко всем модулям, установленным на сервере, осуществляется удаленно через веб-интерфейс. Решения, созданные компанией, позволяют вести учет рабочего времени, осуществлять защищенную идентификацию по смартфону, вести наблюдение за событиями на охраняемом объекте.

* 1. **Сравнительный анализ программных обеспечений СКУД-систем**

Выбор программного продукта для СКУД-системы напрямую зависит от размеров и режимности охраняемого предприятия – чем масштабнее и опаснее производство, тем сложнее и функциональнее система. Немаловажным фактором выбора ПО является возможность его интегрирования с подобранным оборудованием (контроллеры, идентификаторы, считыватели).

В таблице 1.4.1 представлена сравнительная характеристика выше описанных программных продуктов для систем контроля и управления доступом.

Таблица 1.4.1 – Сравнительная характеристика программных продуктов

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Производитель | ZkTeco | ControlGate | PERCo | RusGuard | Parsec | HID Global | Sigur |
| Название системы | ZKTime.Net V3.0 | ControlGate | PERCo-Web | «Standart» «Premium» «Enterprise» | PNSoft | HID Global | Sigur |
| Количество работников | 100000 | 10000 | 100000 | Не ограничено | 12000 | 150000 | 10000 |
| Количество зон охраны | 1024 | 2064 | 1024 | 1600 | 1024 | 1800 | 1024 |
| Возможность интеграции с БД | + | + | + | + | + | + | + |
| Возможность интеграции со сторонними сервисами | + | + | + | + | + | + | + |
| Доп. пакеты ПО | + | + | + | + | + | + | + |

2 ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

**2.1 Постановка задачи**

Требуется создать программное обеспечение на основе базы данных, которое позволит отслеживать передвижения работников некой организации, а также предоставит пользователю возможность просмотра и редактирования информации о сотрудниках и рабочих помещениях предприятия с последующим сохранением во внешней файл.

**2.2 Определение требований пользователей к информационной системе**

Программное обеспечение позволяет работать со всеми таблицами базы данных, синхронизируя изменения в реальном времени. У пользователей системы есть три роли – сотрудник охраны, начальник охраны и системный администратор. Для авторизации в системе пользователь должен ввести свою электронную почту и пароль, авторизация производится по средствам проверки подлинности логина и пароля, а также роли пользователя на сайте. Функцию регистрации не целесообразно предусматривать в данной системе из соображений политики безопасности пользовательских данных.

У роли «сотрудник охраны» есть возможность просмотра таблицы «Доступ» (данных о передвижении сотрудников).

У роли «начальник охраны» есть возможность просмотра и редактирования записей всех таблиц базы данных, просмотр диаграммы и формирование отчетного документа, а также просмотр авторизованных пользователей с ролью «сотрудник охраны» со временем авторизации, с возможностью формирования отчетного документа.

У роли «системный администратор» есть возможность просмотра, редактирования, добавления и удаления записей всех таблиц базы данных, просмотр диаграммы и формирование отчетного документа, а также просмотр всех авторизованных пользователей со временем авторизации и с возможностью формирования отчетного документа.

Диаграмма носит статистический характер. Она показывает пользователю количество посетителей за каждый день месяца. Отчетный документ представляет собой Excel-файл, в котором содержатся все таблицы базы данных. Это создано во избежание путаницы и непонимания при просмотре дочерних таблиц. Второй отчетный файл представляет собой документ MS Excel, содержащий таблицу авторизованных пользователей с точным временем авторизации.

3 РАЗРАБОТКА И ОПИСАНИЕ БАЗЫ ДАННЫХ

**3.1 Проектирование концептуальной модели базы данных в сервисе ERDPlus**

ERDPlus – современный удобный сервис моделирования баз данных с простым и понятным пользователю интерфейсом и достаточно широким функционалом. Он позволяет создавать диаграммы связей сущностей, реляционные схемы, звездных схемы и инструкции SQL DDL. [13].

Для создания концептуальной модели БД необходимо выбрать тип диаграммы «Реляционная схема», после чего в конструкторе создать все необходимые таблицы, продумать столбцы таблиц с указанием типа данных и первичных ключей. Далее установить связи между таблицами и указать внешние ключи. После чего сохранить диаграмму. Результат разработки концептуальной схемы представлен на рисунке 3.1.1.

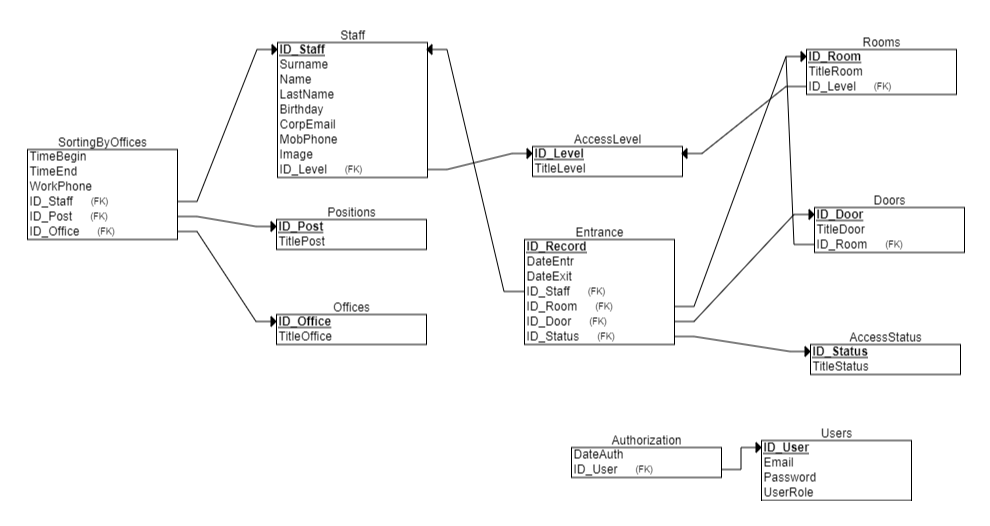


Рисунок 3.1.1 – Концептуальная схема базы данных

Сервис ERDPlus удобен тем, что автоматически генерирует код SQL запроса для создания разработанной БД. После выполнения такого запроса в SQL Management Studio автоматически создается база данных с заданными в схеме типами данных, первичными и внешними ключами и связями между таблицами.

**3.2 Разработка и описание даталогической модели базы данных на платформе Microsoft SQL Server 2014**

SQL Server Management Studio — утилита Microsoft SQL Server 2005 и более поздних версий для конфигурирования, управления и администрирования всех компонентов Microsoft SQL Server. Утилита включает скриптовый редактор и графическую программу, которая работает с объектами и настройками сервера. [12].

Даталогическая схема базы данных представляет собой модель, отражающую логические взаимосвязи между элементами данных безотносительно их содержания и физической организации.

Даталогическая модель формируется непосредственно в SQL Management Studio с помощью функции создания диаграммы. Так как база данных была создана автоматически (см. пункт 3.1), схема базы данных так же сформирована автоматически. Даталогическая схема БД представлена на рисунке 3.2.1.

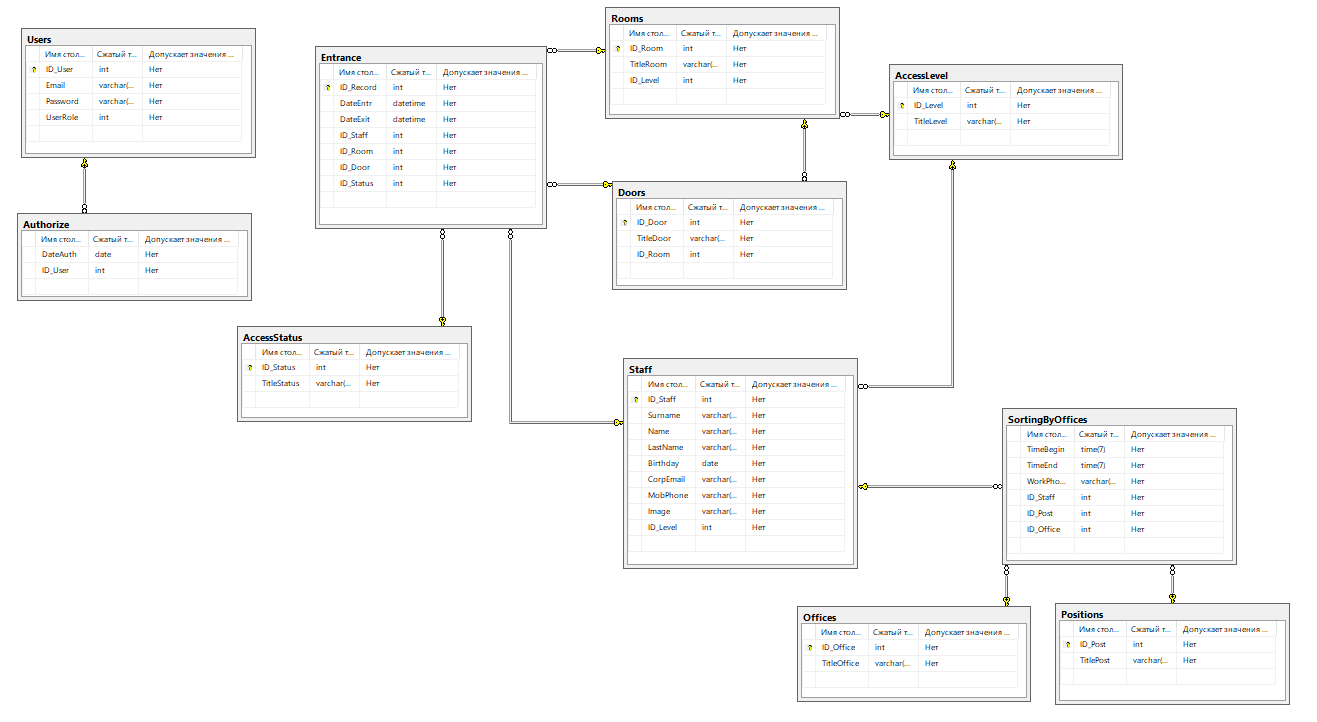


Рисунок 3.2.1 – Даталогическая схема базы данных

Разработанная база данных «EntranceControlWeb» содержит в себе 7 справочных и 2 сводных (дочерних) таблицы данных. Структуры всех таблиц БД представлены в таблицах 3.2.1 – 3.2.9. В каждой таблице предусмотрен первичный ключ, как правило это ID записи. Данные из справочных таблиц в дочерние передаются по средствам внешних ключей и связей.

Таблица 3.2.1 – Структура справочной таблицы «Staff»

|  |  |  |
| --- | --- | --- |
| Наименование столбца | Тип данных | Примечание |
| ID\_Staff | int | PK |
| Surname | varchar(20) |  |
| Name | varchar(20) |  |
| LastName | varchar(20) |  |
| Birthday | date |  |
| CorpEmail | varchar(20) |  |
| MobPhone | char(15) |  |
| Image | varchar(100) |  |
| ID\_Level | int | FK |

Таблица «Staff» (Сотрудники) содержит в себе личную информацию о сотрудниках организации (Фамилия, Имя, Отчество, дата рождения), фото сотрудника, а также контактную информацию – корпоративную почту и номер мобильного телефона. Внешний ключ таблицы («ID\_Level») несет информацию об уровне доступа сотрудника. Первичным ключом таблицы является столбец «ID\_Stuff».

Таблица 3.2.2 – Структура справочной таблицы «Rooms»

|  |  |  |
| --- | --- | --- |
| Наименование столбца | Тип данных | Примечание |
| ID\_Room | int | PK |
| TitleRoom | varchar(20) |  |
| ID\_Level | int | FK |

Справочная таблица «Rooms» (Помещения) содержит в себе информацию обо всех помещениях на территории компании, первичный ключ – «ID\_Room». Внешний ключ таблицы («ID\_Level») несет информацию об уровне доступа соответствующего помещения.

Таблица 3.2.3 – Структура справочной таблицы «AccessLevel»

|  |  |  |
| --- | --- | --- |
| Наименование столбца | Тип данных | Примечание |
| ID\_Level | int | PK |
| TitleLevel | varchar(20) |  |

Справочная таблица «AccessLevel» (Уровни доступа) содержит информацию обо всех возможных уровнях доступа сотрудников и помещений, первичный ключ – «ID\_Level».

Таблица 3.2.4 – Структура справочной таблицы «Doors»

|  |  |  |
| --- | --- | --- |
| Наименование столбца | Тип данных | Примечание |
| ID\_Door | int | PK |
| TitleDoor | varchar(20) |  |
| ID\_Room | int | FK |

Справочная таблица «Doors» (Типы турникетов) содержит информацию обо всех типах турникетов на входах в организацию, первичный ключ – «ID\_Door». Внешний ключ таблицы – «ID\_Room» - несет информацию в каком именно помещении установлен данных турникет.

Таблица 3.2.5 – Структура справочной таблицы «AccessStatus»

|  |  |  |
| --- | --- | --- |
| Наименование столбца | Тип данных | Примечание |
| ID\_Status | int | PK |
| TitleStatus | varchar(20) |  |

Справочная таблица «AccessStatus» (Статусы доступа) содержит в себе информацию о возможных статусах доступа сотрудников, первичный ключ – «ID\_Status».

Таблица 3.2.6 – Структура справочной таблицы «Positions»

|  |  |  |
| --- | --- | --- |
| Наименование столбца | Тип данных | Примечание |
| ID\_Post | int | PK |
| TitlePost | varchar(20) |  |

Справочная таблица «Positions» (Должности) содержит в себе информацию о всех должностях предприятия, первичный ключ – «ID\_Post».

Таблица 3.2.7 – Структура справочной таблицы «Offices»

|  |  |  |
| --- | --- | --- |
| Наименование столбца | Тип данных | Примечание |
| ID\_Office | int | PK |
| TitleOffice | varchar(20) |  |

Справочная таблица «Offices» (Отделы) содержит в себе информацию о всех отдлеах компании, первичный ключ – «ID\_Office».

Таблица 3.2.8 – Структура дочерней таблицы «Entrance»

|  |  |  |
| --- | --- | --- |
| Наименование столбца | Тип данных | Примечание |
| ID\_Record | int | PК |
| DateEntr | date |  |
| DateExit | date |  |
| ID\_Staff | int | FK |
| ID\_Room | int | FK |
| ID\_Door | int | FK |
| ID\_Status | int | FK |

Дочерняя таблица «Entrance» (Досуп) содержит информацию о времени входа и выхода сотрудника, помещении, типе турникета на входе в помещение и статусе доступа. Первичный ключ таблицы – «ID\_Record», внешние ключи: «ID\_Staff», «ID\_Room», «ID\_Door», «ID\_Status».

Таблица 3.2.9 – Структура дочерней таблицы «SortingByOffices»

|  |  |  |
| --- | --- | --- |
| Наименование столбца | Тип данных | Примечание |
| ID\_Item | int | PK |
| TimeBegin | date |  |
| TimeEnd | date |  |
| WorkPhone | char(15) |  |
| Наименование столбца | Тип данных | Примечание |
| ID\_Staff | int | FK |
| ID\_Post | int | FK |
| ID\_Office | int | FK |

Дочерняя таблица «SortingByOffices» (Распределение по отделам) содержит информацию о времени начала и завершения работы сотрудника (в дальнейшем будет добавлена функция редактирования рабочего времени), его ФИО, рабочий телефон, должность и отдел. Первичный ключ таблицы отсутствует, внешние ключи: «ID\_Staff», «ID\_Post», «ID\_Office».

База данных содержит две дополнительные таблицы «Users» и «Authorization» (структуры представлены в таблицах 3.2.10 - 3.2.11). Таблица «Users» нужна для хранения логинов и паролей зарегистрированных пользователей, а таблица «Authorization» – для вывода информации о дате и времени авторизации пользователя на сайте.

Таблица 3.2.10 – Структура таблицы «Users»

|  |  |  |
| --- | --- | --- |
| Наименование столбца | Тип данных | Примечание |
| ID\_User | int | PK |
| Email | varchar(20) |  |
| Password | varchar(20) |  |
| UserRole | int |  |

Таблица «Users» (Пользователи) содержит информацию об Email-адресе зарегистрированного пользователя, его пароль и роль на сайте. Первичный ключ таблицы – «ID\_User».

Таблица 3.2.11 – Структура таблицы «Authorization»

|  |  |  |
| --- | --- | --- |
| Наименование столбца | Тип данных | Примечание |
| DateAuth | date |  |
| ID\_User | int | FK |

Таблица «Authorization» содержит информацию о дате входа на сайт конкретного пользователя. Таблица не содержит первичных ключей, внешний ключ таблицы – «ID\_User».

**3.3 Автоматизация наполнения базы тестовыми данными   
с использованием сервиса Microsoft SQL Server Integration Services**

Для заполнения БД тестовыми данными был создан Excel-файл версии Microsoft Excel 97-2003 (для лучшей совместимости с Microsoft SQL Server Integration Services и SQL Management Studio 2014), в который с помощью задачи экспорта данных были перенесены шаблоны всех таблиц БД. Затем файл был вручную наполнен тестовыми данными. Пример наполнения тестовыми данными таблицы «Сотрудники» представлен на рисунке 3.3.1.

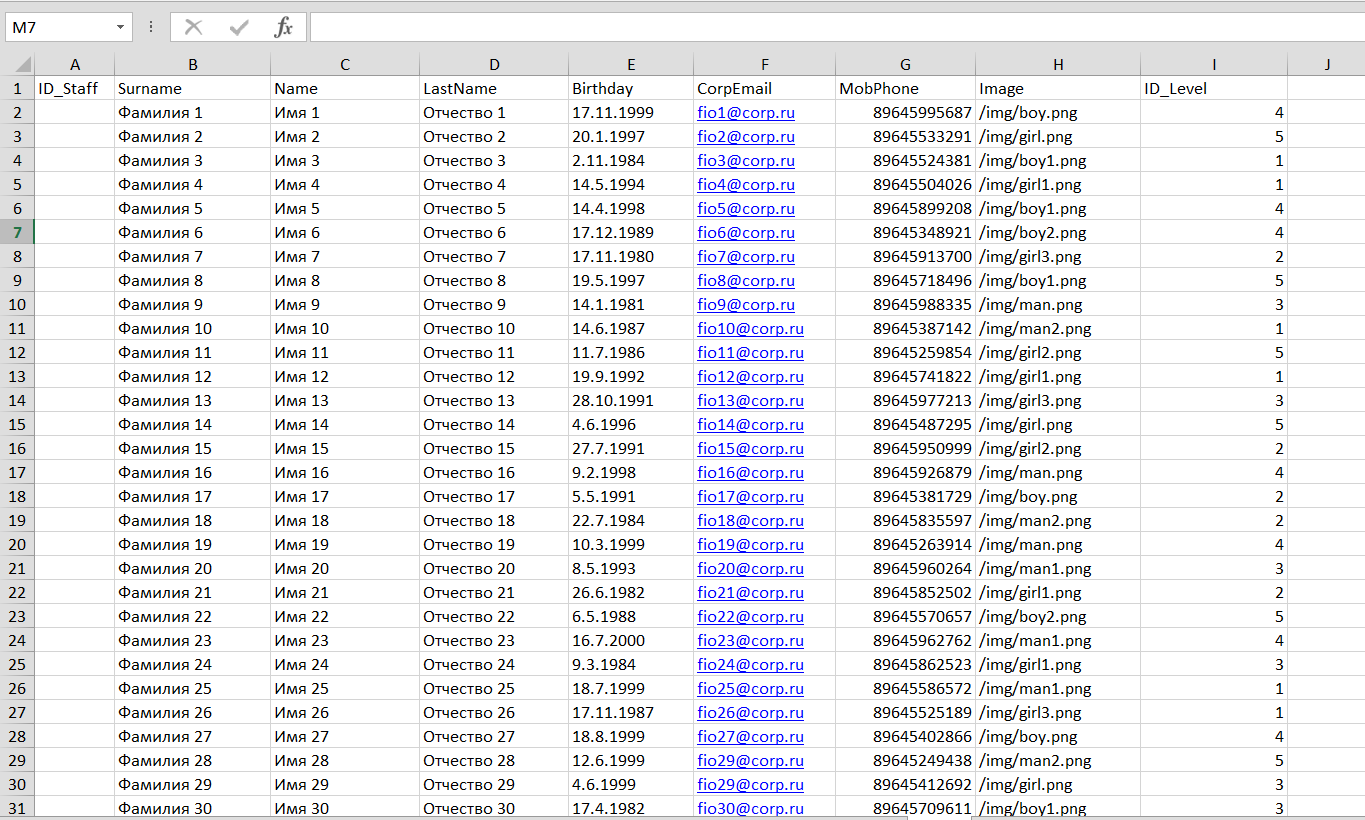


Рисунок 3.3.1 – Наполнение таблицы «Сотрудники» тестовыми данными

Затем в среде разработки Visual Studio 2019 был установлен пакет Microsoft SQL Server Integration Service и создан проект Integration Service Project. Работа в сервисе осуществляется по средствам добавления в окно конструктора соответствующих блоков задач из раздела «Инструменты». Первый блок – «Задача выполнение SQL» – выполняет функцию подключения к серверу и базе данных, а также очистку таблиц базы с помощью специального SQL-скрипта от возможных не нужных записей во избежание возникновения ошибок при переносе данных. Последующие блоки – «Задача потока данных» – отвечают за заполнение данными соответствующих таблиц БД. Блоки «Задача потока данных» состоят из функциональных модулей «Источник Excel» и «Назначение ADO NET». В модуле источника происходит подключение к Excel-файлу с тестовыми данными и выбор необходимых столбцов, а модуль «назначение» отвечает за подключение к соответствующей таблице базы данных и корректный перенос данных. При наличии столбцов с типом данных DateTime, необходимо выполнить промежуточную операцию преобразования данных для корректного переноса их в таблицу базы данных. Данная операция выполняется с помощью дополнительного модуля «Преобразование данных», в котором формат данных файла Excel конвертируется в формат данных SQL Management Studio. Результат работы в сервисе Microsoft SQL Server Integration Service представлен на рисунке 3.3.2.

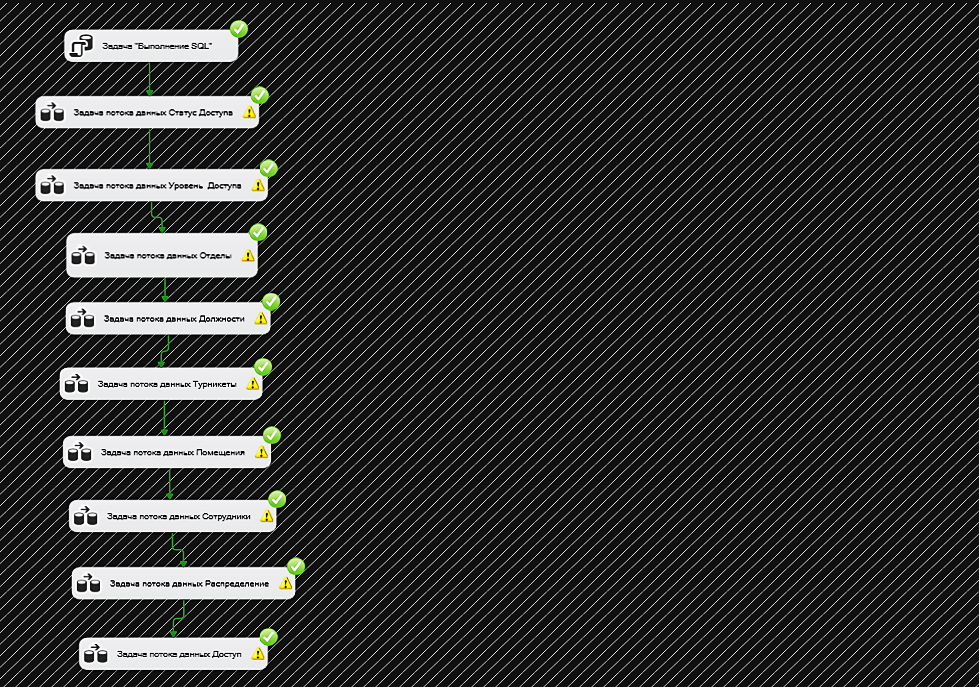


Рисунок 3.3.2 – Успешное выполнение операции наполнения данными   
в среде Microsoft SQL Server Integration Service

4 РАЗРАБОТКА И ОПИСАНИЕ КЛИЕНТСКОГО ПРИЛОЖЕНИЯ

**4.1 Управление версиями проекта на основе сервиса GitHub**

Системы контроля версий позволяют разработчикам сохранять все изменения, внесённые в код. Поэтому в случае возникновения неисправных ошибок кода они могут просто откатить код до рабочего состояния. Системы контроля версий также дают возможность нескольким разработчикам работать над одним проектом и сохранять внесённые изменения, чтобы убедиться, что все могут следить за тем, над чем они работают.

Данные возможности полностью предоставляет система управления версиями GitHub – сервис онлайн-хостинга репозиториев, обладающий всеми функциями распределённого контроля версий и функциональностью управления исходным кодом — всё, что поддерживает Git. Используется вместе с Git и даёт разработчикам возможность сохранять их код онлайн, а затем взаимодействовать с другими разработчиками в разных проектах. Работа с системой осуществлялась посредствам графического интерфейса настольного приложения GitHub Desktop (рисунок 4.1.1) [14].

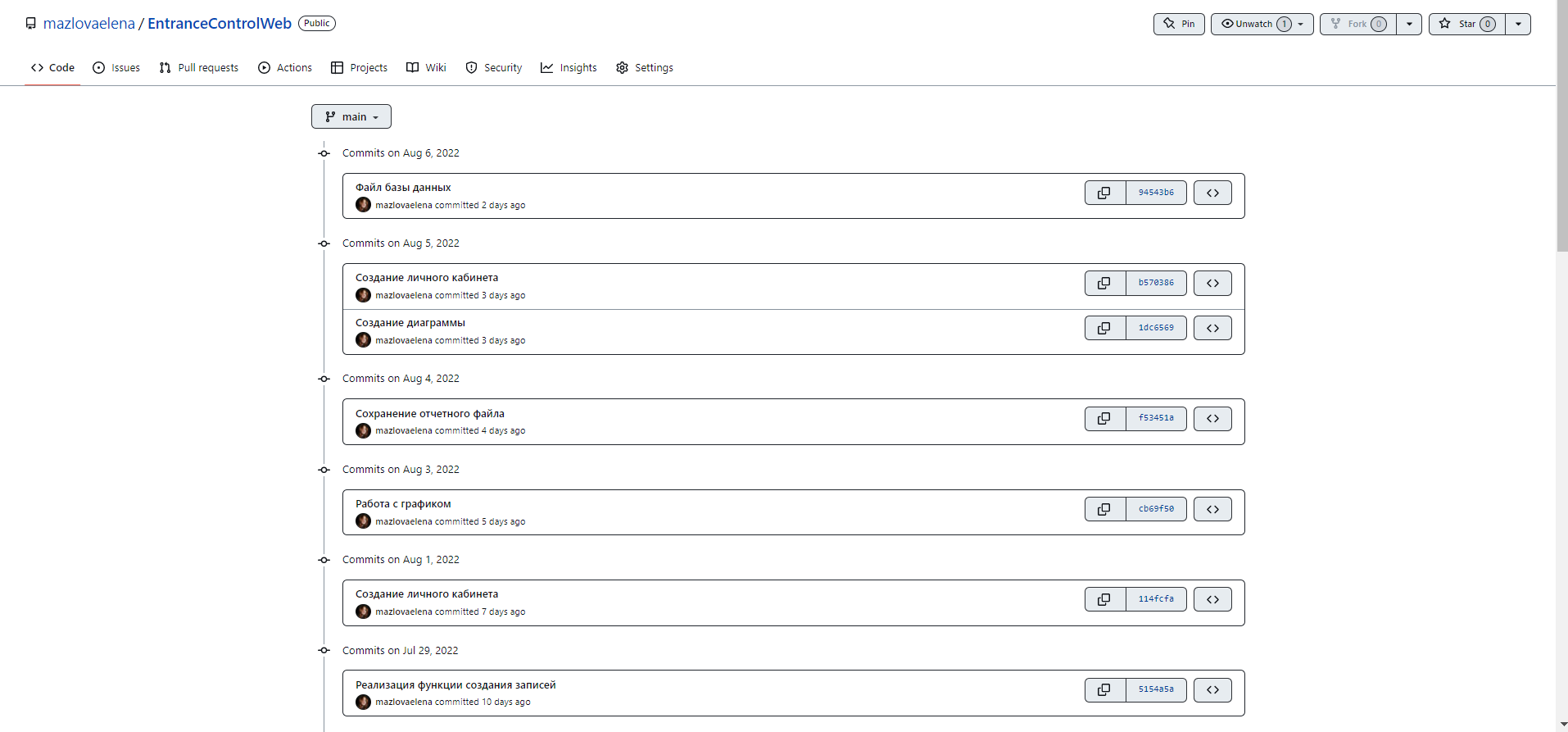


Рисунок 4.1.1 – Репозиторий проекта в сервисе GitHub

Ссылка на репозиторий в системе GitHub: <https://github.com/mazlovaelena/EntranceControlWeb>

**4.2 Разработка клиентского приложения**

Для разработки клиентского приложения был создан проект ASP.NET Core Web Application (MVC) в среде разработки Visual Studio 2019 на платформе .Net 5.0.

Процесс разработки приложения начался с создания дизайна пользовательских страниц. Для этого были использованы дополнительные языки программирования CSS, HTML и JavaScript. CSS отвечает за дизайн страницы, HTML определяет разметку страницы и расположение элементов, а JavaScript необходим для отображения динамических элементов страницы.

Далее была реализована функция авторизации пользователей по ролям. С этой целью необходимо подключить базу данных к проекту для корректного обмена данными. Для этого были использованы расширения Microsoft.EntityFrameworkCore.SqlServer и Microsoft.EntityFrameworkCore.SqlServer.Tools, подключение выполнялось с помощью ввода в консоль диспетчера пакетов соответствующей строки подключения и создания строк подключения в файле конфигурации appsettings.json и файле запуска программы Startup.cs. После подключения БД в проекте появились классы для каждой из таблиц базы и класс-контекст (рисунок 4.2.1), через который происходит обращение к таблицам БД.

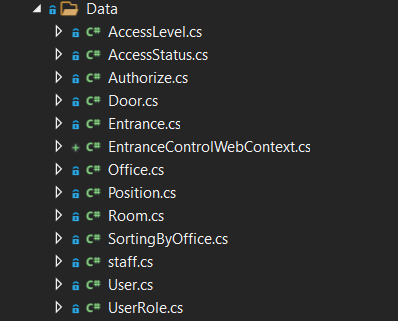


Рисунок 4.2.1 – Классы таблиц базы данных

Реализация ролей пользователей на сайте подразумевает создание дополнительного базового класса UserRole.cs, содержащего в себе определение значений для каждой роли на сайте.

Для корректного вывода данных на пользовательские страницы необходимо преобразовать все записи таблиц в листы моделей представления и затем вывести их на форму. Преобразование данных осуществлялось в контроллере, а функция вывода строк была написана в представлениях с использованием цикла foreach. Фрагменты листинга кода модели и контроллера представлены на рисунке 4.2.2.

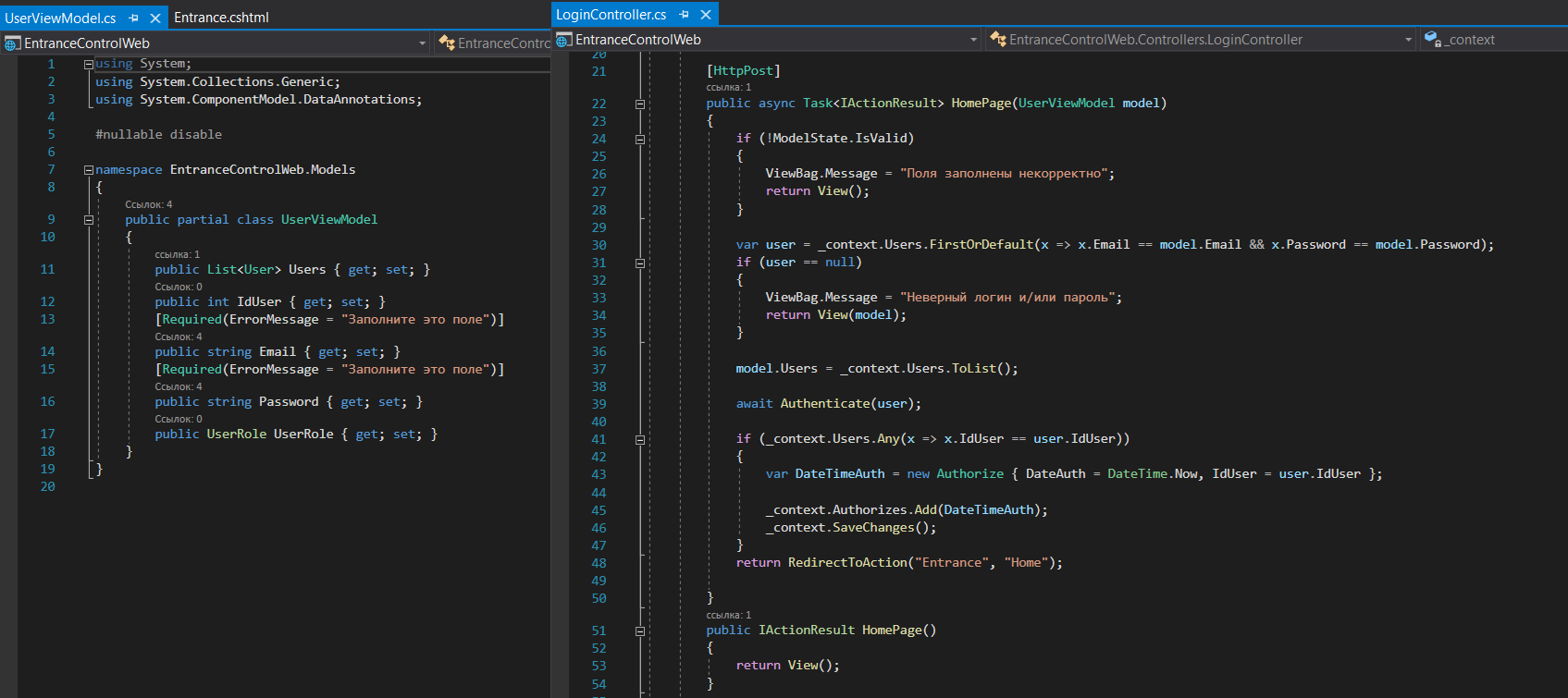


Рисунок 4.2.2 – Фрагменты кода приложения (слева контроллер LoginController.cs, справа модель преставления UserViewModel.cs)

Для пользователей с ролями «начальник охраны» и «системный администратор» предусмотрены возможности редактирования, добавления и удаления информации. Данные функции подразумевают создание дополнительных форм ввода данных. Логика сохранения изменений прописана в контроллере и связана с пользовательскими страницами посредствам моделей. Листинг программного кода контроллера представлен в приложении А.

Во избежание некорректного ввода данных и возможных ошибок была предусмотрена валидация данных, предупреждающая пользователя о некорректном вводе или об отправке пустой формы (рисунок 4.2.3). Функции валидации прописаны в моделях, а вывод – в представлениях.

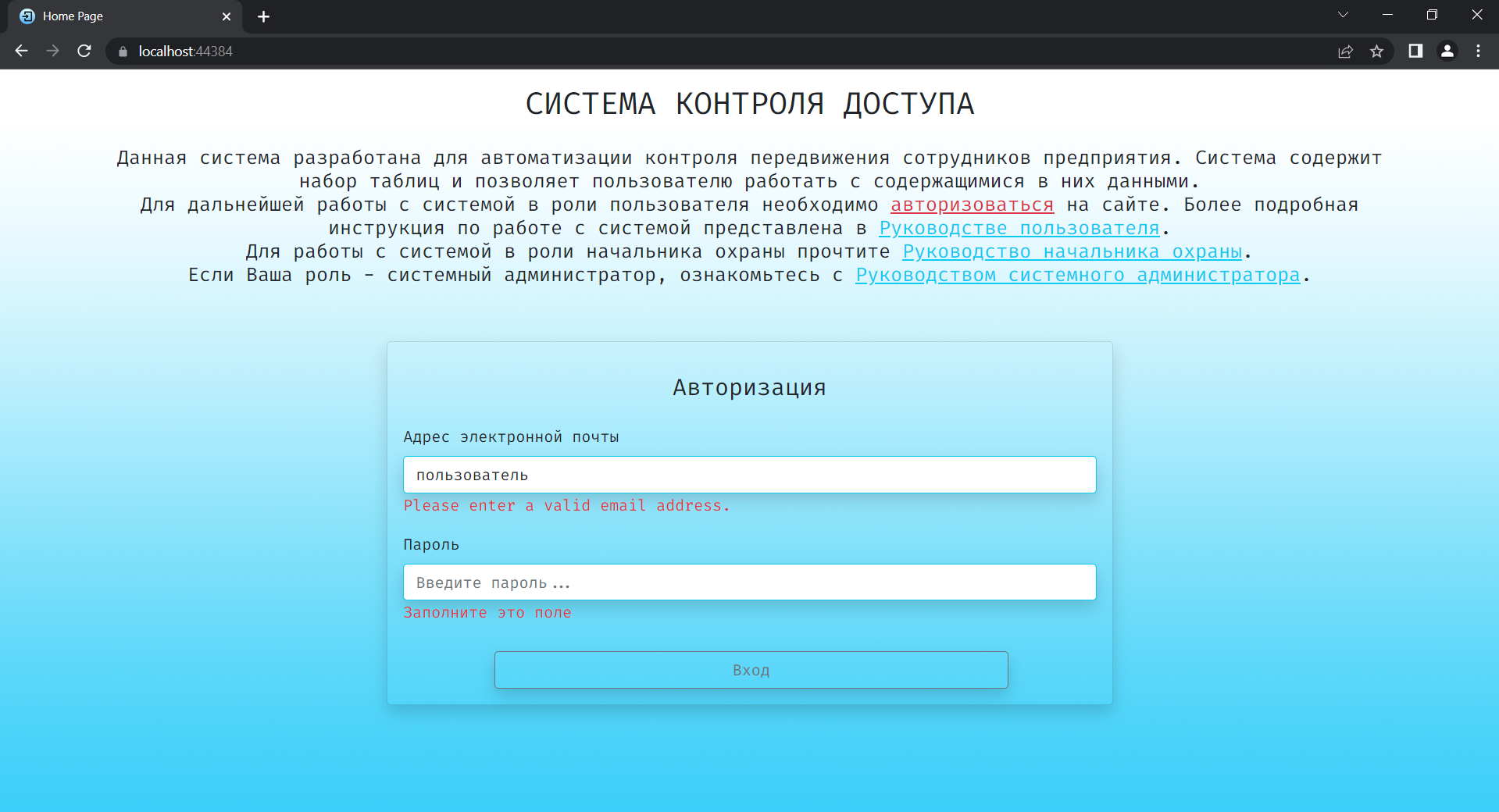


Рисунок 4.2.3 – Пример работы валидации данных

Система содержит статистический график количества посетителей за каждый день месяца, доступный в личном кабинете пользователей с ролями «начальник охраны» и «системный администратор». С этой целью в контроллере был написан LINQ-запрос, подсчитывающий количество записей за каждое число месяца и сохраняющий эти данные в лист для удобства дальнейшей работы с ними. Для создания диаграммы на пользовательской странице была использована библиотека Chart.js. Далее на пользовательскую форму был добавлен элемент canvas и прописан скрипт, содержащий тип диаграммы, дизайн и данные из листа, созданного заранее. Фрагмент кода страницы «Chart.cshtml» представлен на рисунке 4.2.4.

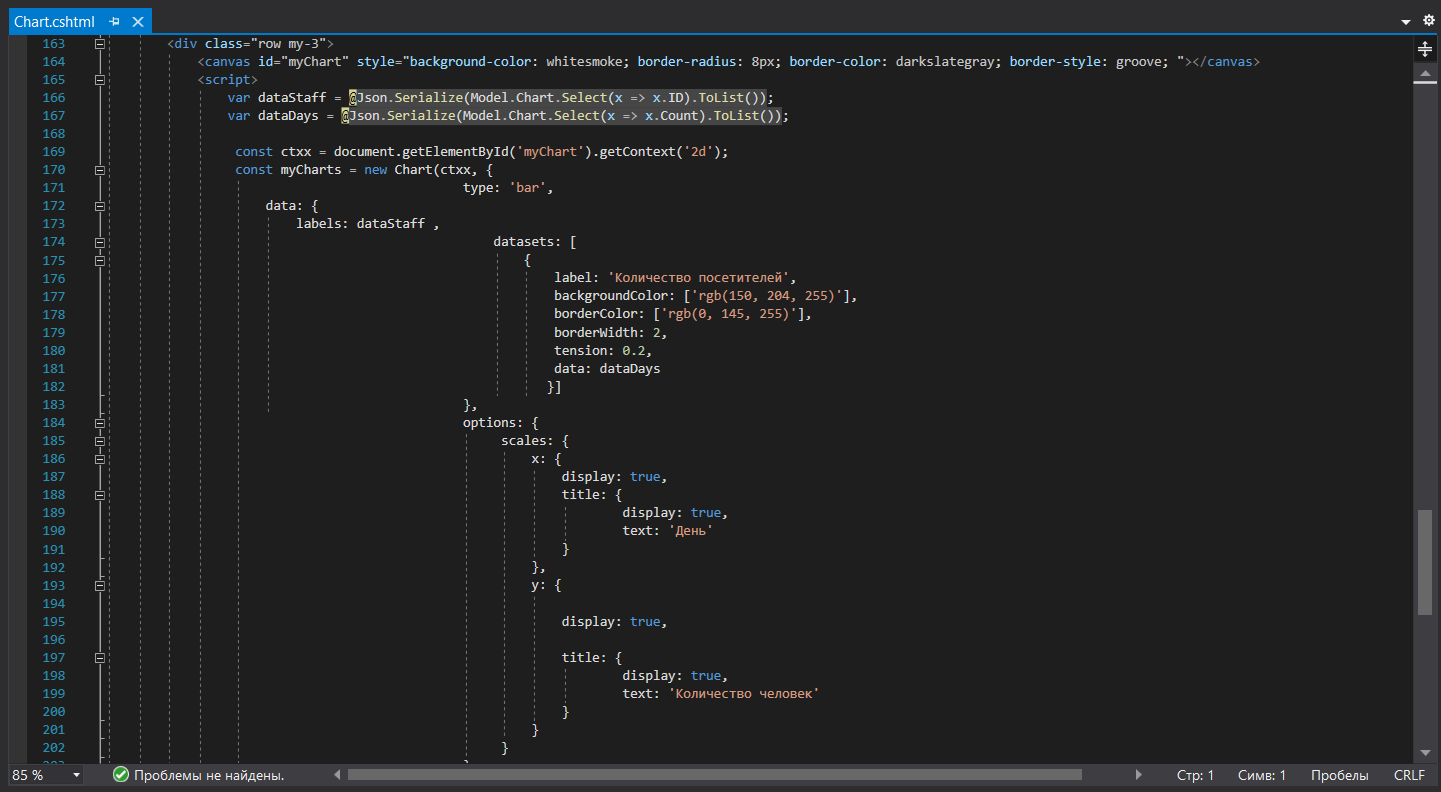


Рисунок 4.2.4 – Код страницы вывода диаграммы «Chart.cshtml»

Одной из функций данной системы является экспорт данных всех таблиц во внешний файл. Табличные данные удобно экспортировать в Excel-файл. Для этого в проект было подключено расширение EPPlus, позволяющее экспортировать данные непосредственно из пользовательского приложения и сохранять их во внешнем файле. В контроллере был прописан функциональный код, передающий данные таблиц в ячейки Excel, для каждой таблицы в файле были созданы отдельные листы. После нажатия пользователем соответствующей кнопки файл автоматически скачивается на ПК пользователя. Таким же образом реализовано сохранение таблицы посетителей сайта. Подобный функционал также доступен только пользователям с ролями «начальник охраны» и «системный администратор». Фрагмент кода контроллера представлен на рисунке 4.2.5.

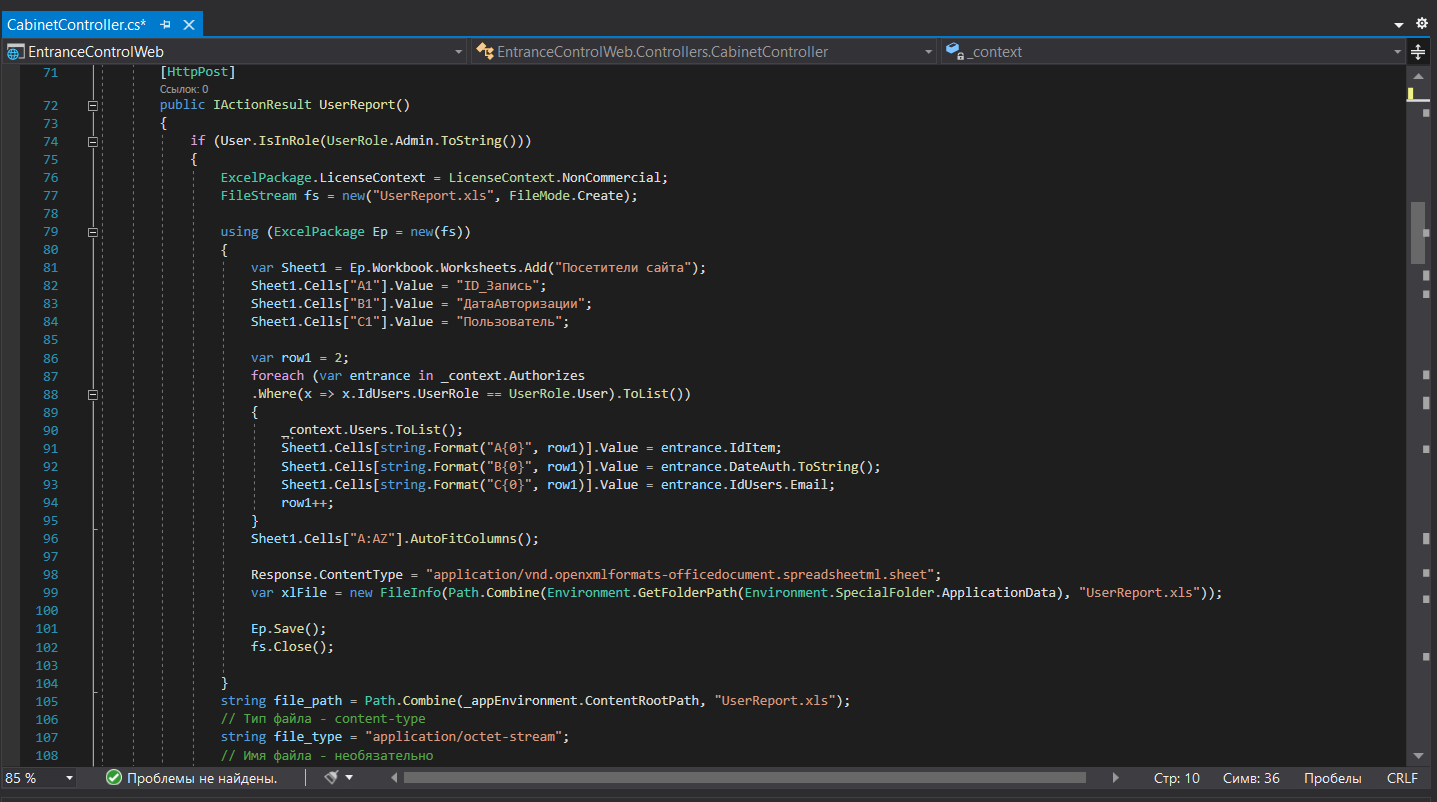


Рисунок 4.2.5 – Фрагмент кода контроллера CabinetContoller.cs

**4.3 Описание работы приложения**

При запуске приложения пользователя встречает главная страница (рисунок 4.3.1) с формой авторизации пользователя. Для дальнейшей работы в системе необходима обязательная авторизация.

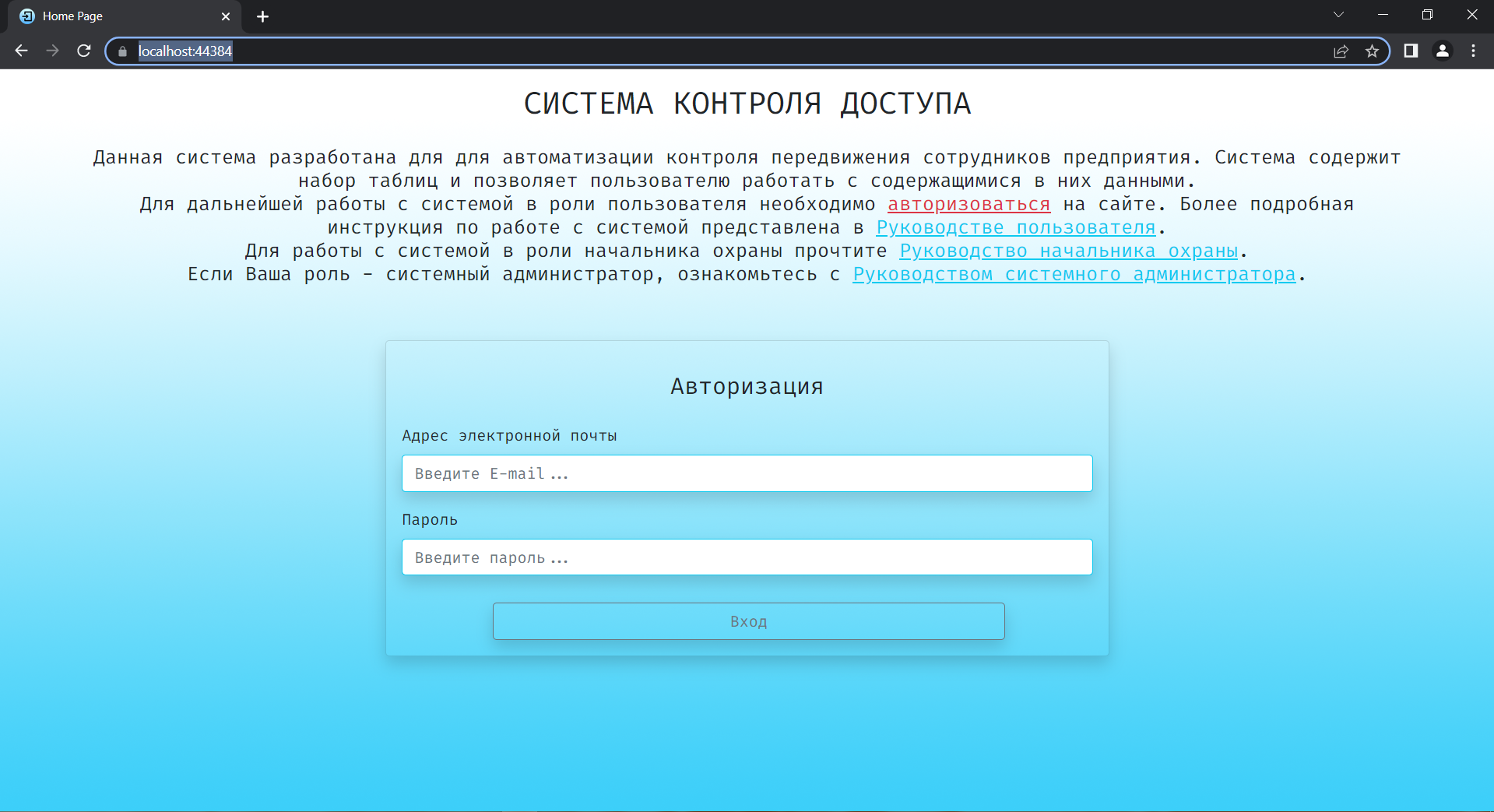


Рисунок 4.3.1 – Главная страница системы

На главной странице есть ссылки на руководства по использованию системы для всех пользователей системы. Пример такого руководства представлен на рисунке 4.3.2.

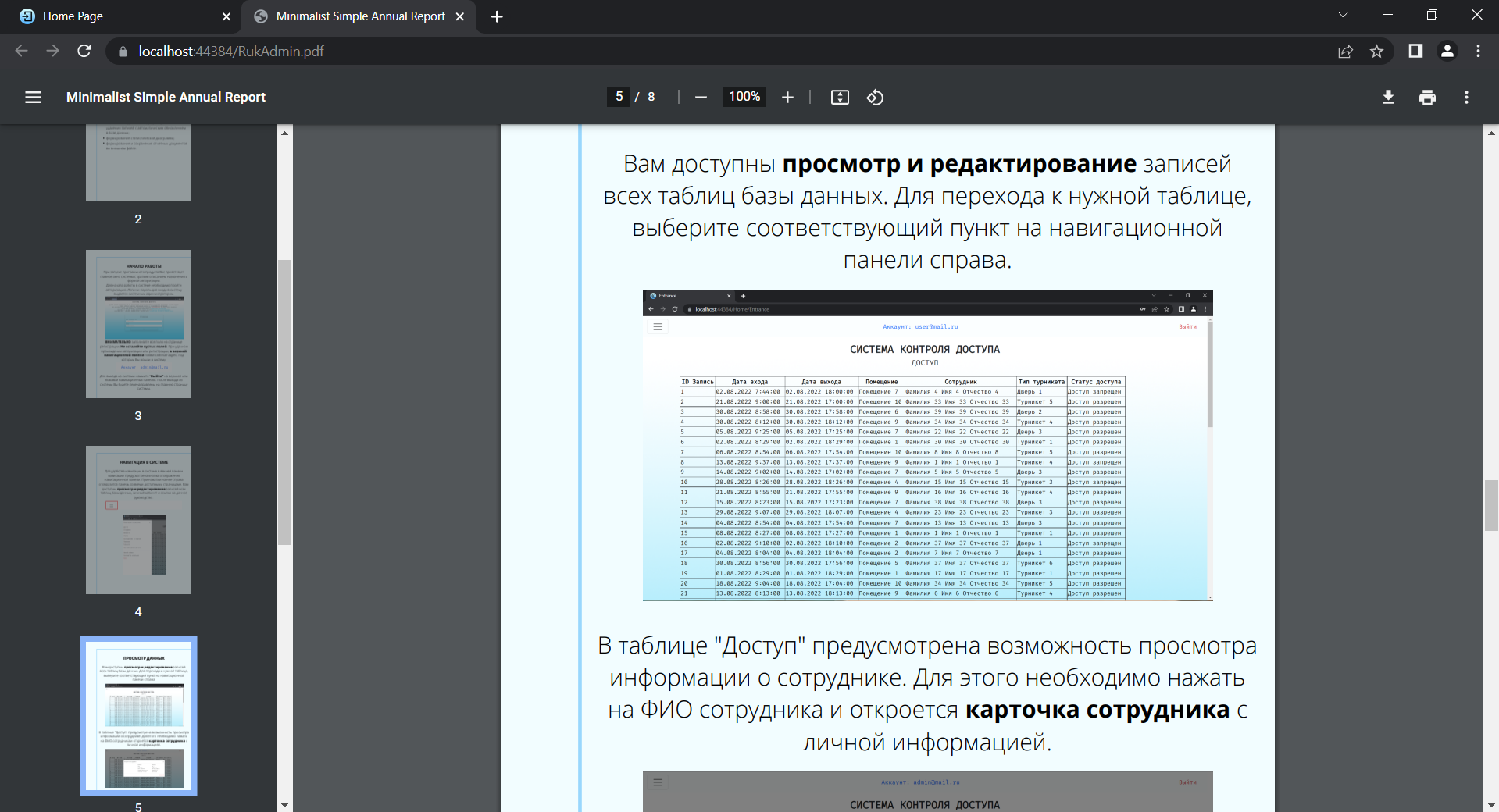


Рисунок 4.3.2 – Руководство начальника охраны

После авторизации в системе на панели навигации вверху страницы появится email, под которым пользователь вошел на сайт. Навигация в системе реализована с помощью удобной динамической навигационной панели (рисунок 4.3.3). Ее содержимое меняется в зависимости от роли в системе.

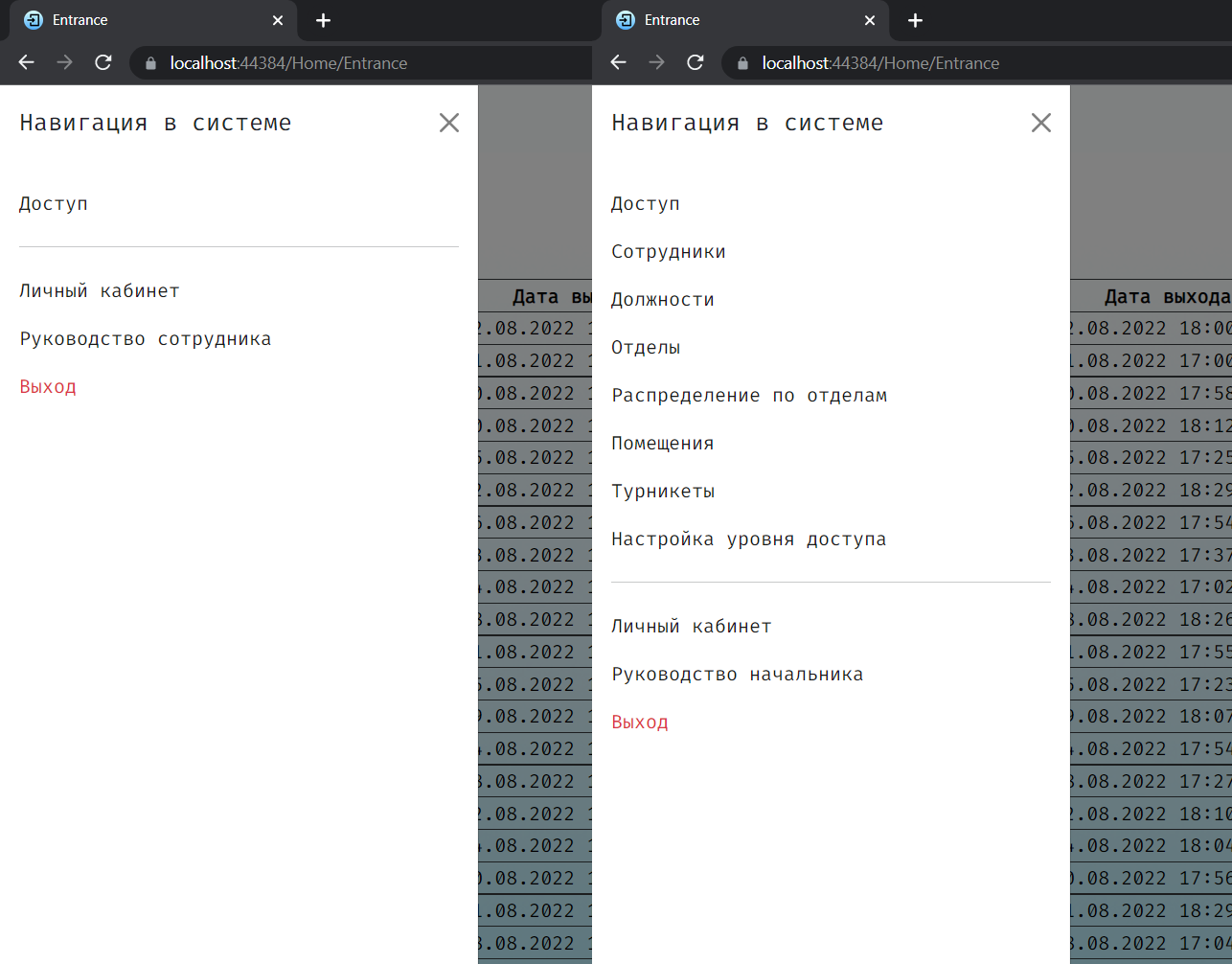


Рисунок 4.3.3 – Навигационная панель системы (справа – для пользователя с ролью «сотрудник охраны», слева – для пользователя с ролью «начальник охраны»)

У пользователя с ролью «сотрудник охраны» ограниченные права на сайте: он может просматривать таблицу «Доступ», имеет доступ к личному кабинету (содержится информация об email и роли на сайте) и руководству пользователя.

При авторизации в системе с ролью «начальник охраны» у пользователя появляется доступ ко всем таблицам и возможность редактирования информации (рисунок 4.3.4), а также доступ к личному кабинету (рисунок 4.3.5) с функциями просмотра диаграммы, просмотра пользователей сайта с ролью «сотрудник охраны» и сохранения данных во внешний файл (рисунок 4.3.6).

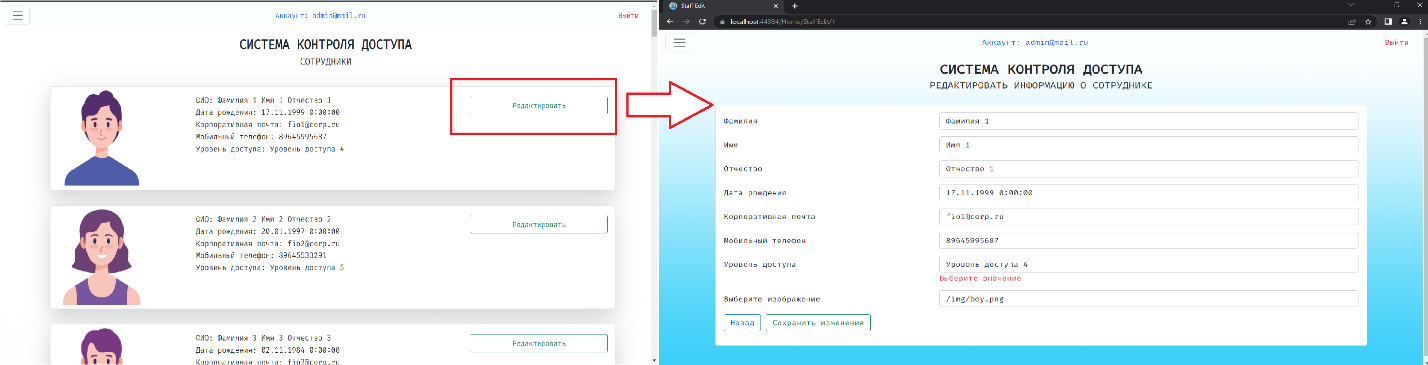


Рисунок 4.3.4 – Редактирование записи таблицы «Сотрудники»

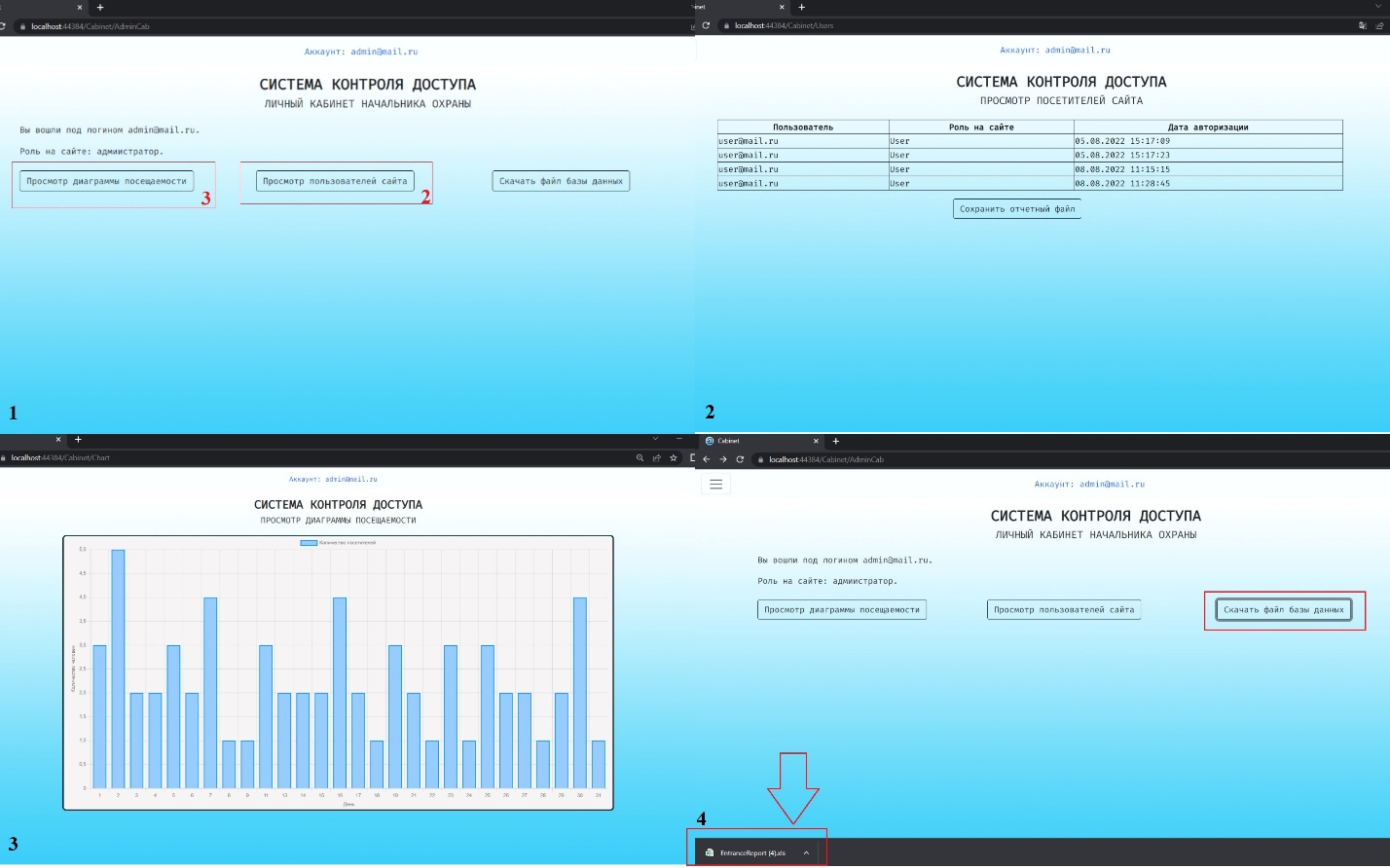


Рисунок 4.3.5 – Личный кабинет начальника охраны: 1 – функции личного кабинета; 2 – просмотр посетителей сайта; 3 – просмотр диаграммы; 4 – сохранение отчетного файла

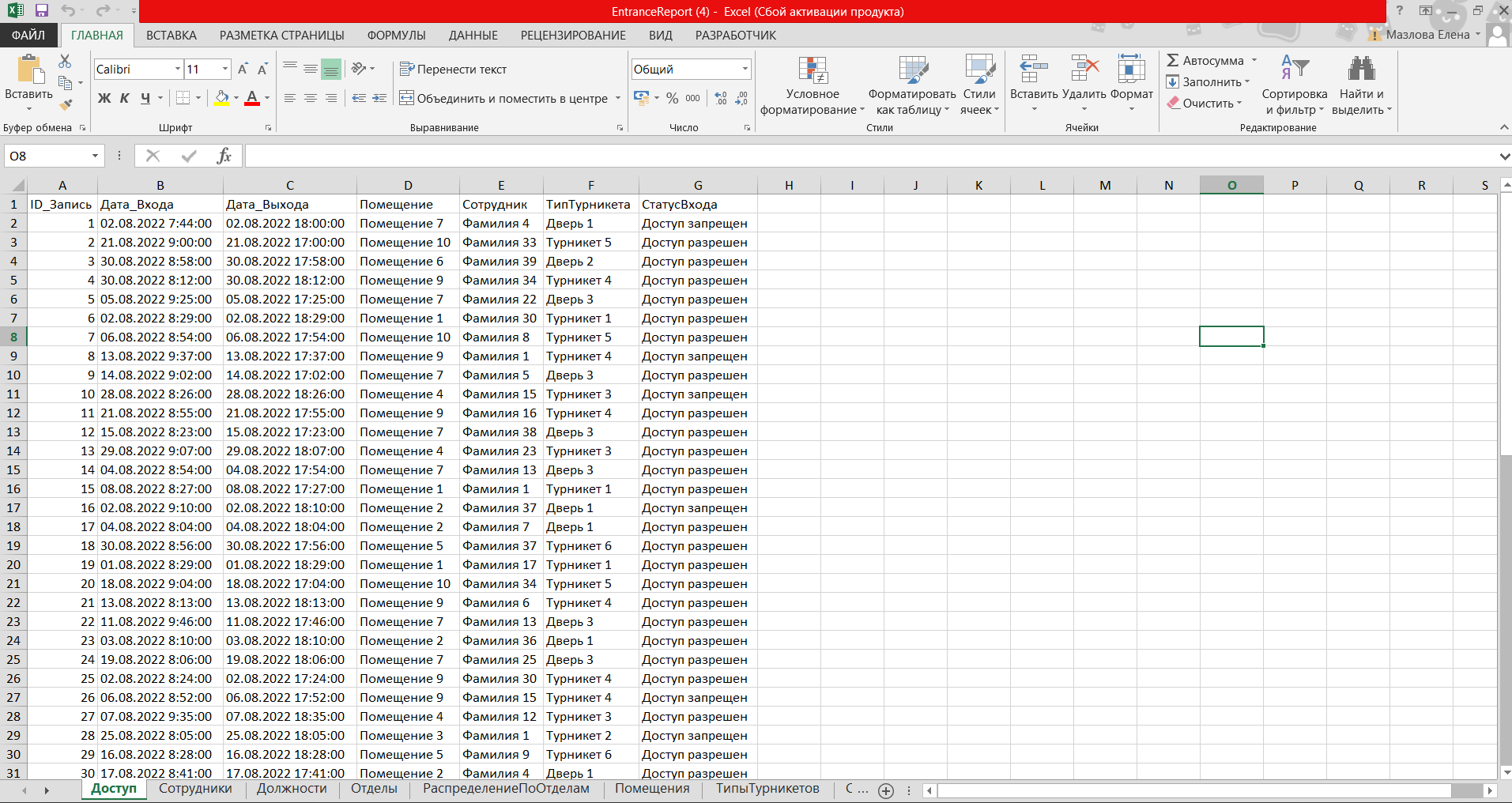


Рисунок 4.3.6 – Сохраненный Excel-файл базы данных

Для пользователя с ролью «системный администратор» в приложении помимо функции редактирования, предусмотрены функции добавления и удаления новых записей во все таблицы (рисунок 4.3.7), а также доступ к личному кабинету с функциями просмотра диаграммы, просмотра всех пользователей и сохранения данных во внешний файл.

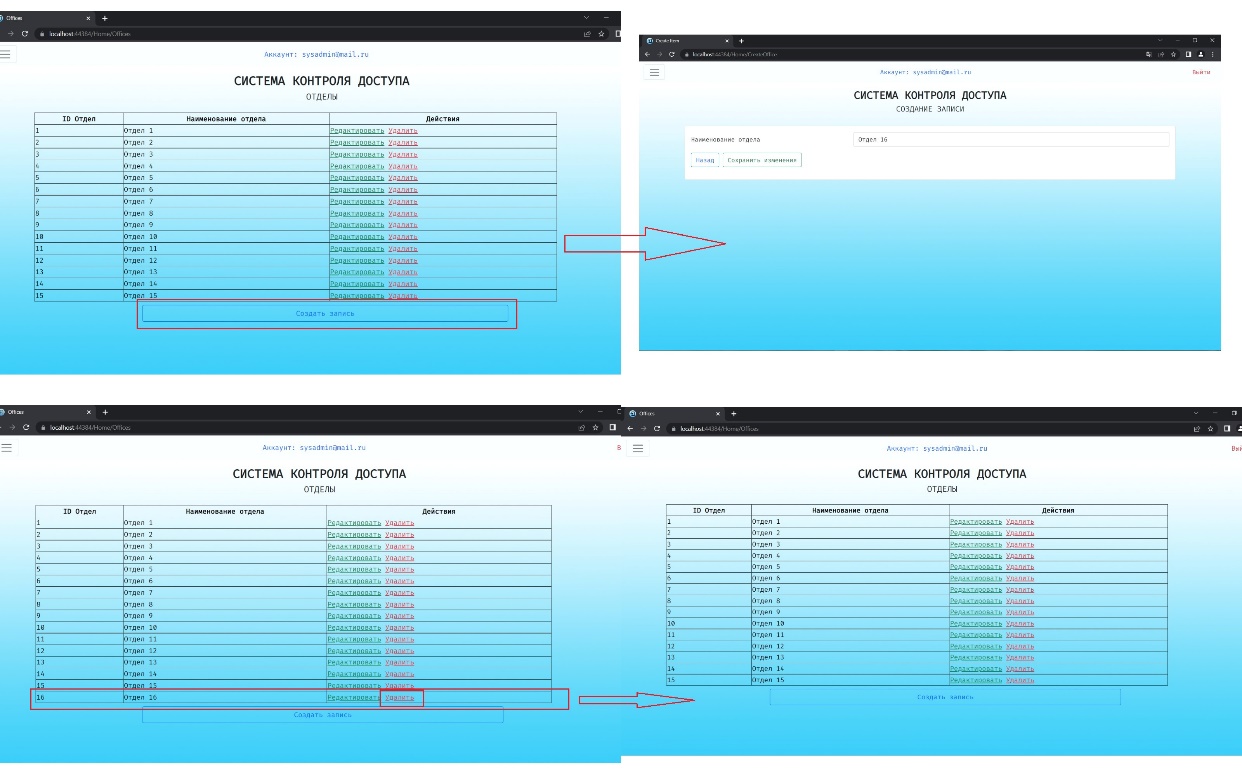


Рисунок 4.3.7 – Функции добавления записей (сверху) и удаления записей (снизу)

ЗАКЛЮЧЕНИЕ

В ходе прохождения производственной практики был разработан и реализован программный продукт «Система контроля доступа», интегрированный с базой данных MS SQL Server 2014 в режиме реального времени. В ходе создания веб-сервиса реализованы основные этапы создания программного обеспечения:

1. изучение предметной области проекта;
2. разработка архитектуры информационной системы;
3. разработка структуры базы данных и автоматизированное наполнение ее тестовыми данными;
4. реализация многопользовательского интерфейса web-приложения с возможностью авторизации и регистрации пользователей;
5. реализация функции вывода данных в табличном виде;
6. реализация функции добавления новых записей, редактирования и удаления существующих;
7. реализация функции вывода данных в графическом виде;
8. реализация функции сохранения данных во внешнем файле;
9. создание интерактивного отчета с результатами визуализации данных из базы в табличной и графической форме для размещения его на web-сервере предприятия.

В ходе реализации проекта использована система контроля и управления версиями программного обеспечения GitHub, организовано централизованное хранение программных версий в удаленном репозитории и настроем обмен с локальным репозиторием на рабочем компьютере.

На данный момент приложение содержит недостаточно полный функционал и планируется его доработка. Однако, при реализованных на данный момент функциях, программный продукт позволит автоматизировать и значительно упростит контроль доступа на любое предприятие, тем самым повысив уровень безопасности организации.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. ГОСТ 24104-85 ЕСС АСУ «Автоматизированные системы управления. Общие требования».
2. ГОСТ 24104-86 ЕСС АСУ «Автоматизированные системы управления. Стадии создания».
3. ГОСТ 34.201-89 «Информационная технология. Комплекс стандартов на автоматизированные системы. Виды, комплектность и обозначение документов при создании автоматизированных систем».
4. ГОСТ 34.603-92 «Информационная технология. Комплекс стандартов на автоматизированные системы. Виды и порядок освоения».
5. Гущин А. Н. Базы данных: учебник. – М.: Директ-Медиа, 2014. – 266 с.   
   <http://biblioclub.ru/index.php?page=book_red&id=222149&sr=1>
6. Гущин А.Н. Базы данных: учебно-методическое пособие. – М., Берлин: Директ-Медиа, 2015. – 311 с.   
   <http://biblioclub.ru/index.php?page=book_red&id=278093&sr=1>
7. Указания к оформлению дипломных и курсовых проектов и работ. Методические указания / Н.Б. Лошкарев, А.Н. Лошкарев, Л.А. Зайнуллин. – Екатеринбург: УГТУ–УПИ, 2007. – 49 с.
8. Википедия: SQL Server Management Studio. Полный текст: <https://ru.wikipedia.org/wiki/SQL_Server_Management_Studio>
9. ERDPlus – официальный сайт. [ЭРДПлюс (erdplus.com)](https://erdplus.com/)
10. Википедия: Azure DevOps Server. Полный текст: <https://ru.wikipedia.org/wiki/Azure_DevOps_Server>
11. Википедия: GitHub. Полный текст: <https://ru.wikipedia.org/wiki/GitHub>
12. Назарова О.Б., Мекешкин Е.Т. АНАЛИЗ СИСТЕМ КОНТРОЛЯ И УПРАВЛЕНИЯ ДОСТУПОМ ДЛЯ ИСПОЛЬЗОВАНИЯ В МУНИЦИПАЛЬНЫХ ОРГАНИЗАЦИЯХ // Научное обозрение. Технические науки. – 2019. – № 4. – С. 50-54; Полный текст: https://science-engineering.ru/ru/article/view?id=1256
13. Алгоритм и основные принципы работы СКУД. Полный текст: <http://asupro.com/building/control/algorithm-basic-principles-skud.html>
14. Метанит: Отправка файлов. Полный текст: <https://metanit.com/sharp/aspnet5/5.7.php>

ПРИЛОЖЕНИЕ А ФРАГМЕНТ ЛИСТИНГА ПРОГРАММНОГО КОДА   
ПОЛЬЗОВАТЕЛЬСКОГО ПРИЛОЖЕНИЯ

Класс-контроллер HomeController.cs

namespace EntranceControlWeb.Controllers

{

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

private EntranceControlWebContext \_context;

public HomeController(ILogger<HomeController> logger, EntranceControlWebContext context)

{

\_logger = logger;

\_context = context;

}

#region ДЕЙСТВИЯ С ТАБЛИЦЕЙ "ТУРНИКЕТЫ"

//Отображение страницы

public IActionResult Doors(DoorViewModel door)

{

door.Doors = \_context.Doors.ToList();

door.Rooms = \_context.Rooms.ToList();

return View(door);

}

//Удаление записи

public IActionResult DelDoor(int id)

{

var data = \_context.Doors.FirstOrDefault(x => x.IdDoor == id);

var entr = \_context.Entrances.FirstOrDefault(x => x.IdDoor == id);

if (entr != null)

{

\_context.Entrances.Remove(entr);

\_context.Doors.Remove(data);

}

\_context.SaveChanges();

return RedirectToAction(nameof(Doors));

}

//Редактирование записи

[HttpPost]

public IActionResult DoorEdit(DoorViewModel door)

{

var edit = \_context.Doors.FirstOrDefault(x => x.IdDoor == door.IdDoor);

edit.IdDoor = door.IdDoor;

edit.TitleDoor = door.TitleDoor;

edit.IdRoom = door.IdRoom;

\_context.SaveChanges();

return RedirectToAction(nameof(Doors));

}

public IActionResult DoorEdit(int id)

{

var viewmodel = new DoorViewModel();

var view = \_context.Doors.FirstOrDefault(x => x.IdDoor == id);

if (view != null)

{

viewmodel.IdDoor = view.IdDoor;

viewmodel.TitleDoor = view.TitleDoor;

viewmodel.IdRoom = view.IdRoom;

}

return Json(viewmodel);

}

//Создание записи

[HttpPost]

public IActionResult CreateDoor (DoorViewModel door)

{

var create = new Door { TitleDoor = door.TitleDoor, IdRoom = door.IdRoom };

\_context.Doors.Add(create);

\_context.SaveChanges();

return RedirectToAction(nameof(Doors));

}

public IActionResult CreateDoor()

{

var door = new DoorViewModel();

door.Doors = \_context.Doors.ToList();

door.Rooms = \_context.Rooms.ToList();

return View(door);

}

#endregion

#region ДЕЙСТВИЯ С ТАБЛИЦЕЙ "УРОВНИ ДОСТУПА"

//Отображение сраницы

public IActionResult Levels(AccessLevelViewModel acclev, AccessStatusViewModel accstat)

{

acclev.Levels = \_context.AccessLevels.ToList();

accstat.Statuses = \_context.AccessStatuses.ToList();

return View(acclev);

}

//Удаление записи

public IActionResult DelLevel(int id)

{

var data = \_context.AccessLevels.FirstOrDefault(x => x.IdLevel == id);

var entr = \_context.staff.FirstOrDefault(x => x.IdLevel == id);

var room = \_context.Rooms.FirstOrDefault(x => x.IdLevel == id);

if(data != null)

{

\_context.staff.Remove(entr);

\_context.Rooms.Remove(room);

\_context.AccessLevels.Remove(data);

}

\_context.SaveChanges();

return RedirectToAction(nameof(Levels));

}

//Редактирование записи

[HttpPost]

public IActionResult LevelEdit(AccessLevelViewModel acclev)

{

var edit = \_context.AccessLevels.FirstOrDefault(x => x.IdLevel == acclev.IdLevel);

edit.IdLevel = acclev.IdLevel;

edit.TitleLevel = acclev.TitleLevel;

\_context.SaveChanges();

return RedirectToAction(nameof(Levels));

}

public IActionResult LevelEdit(int id)

{

var viewmodel = new AccessLevelViewModel();

var view = \_context.AccessLevels.FirstOrDefault(x => x.IdLevel == id);

if (view != null)

{

viewmodel.IdLevel = view.IdLevel;

viewmodel.TitleLevel = view.TitleLevel;

}

return Json(viewmodel);

}

//Создание записи

[HttpPost]

public IActionResult CreateLevel(AccessLevelViewModel lev)

{

var create = new AccessLevel { TitleLevel = lev.TitleLevel };

\_context.AccessLevels.Add(create);

\_context.SaveChanges();

return RedirectToAction(nameof(Levels));

}

public IActionResult CreateLevel()

{

return View();

}

#endregion

#region ДЕЙСТВИЯ С ТАБЛИЦЕЙ "ОТДЕЛЫ"

//Отображение страницы

public IActionResult Offices(OfficeViewModel off)

{

off.Offices = \_context.Offices.ToList();

return View(off);

}

//Удаление записи

public IActionResult DelOffice(int id)

{

var data = \_context.Offices.FirstOrDefault(x => x.IdOffice == id);

var office = \_context.SortingByOffices.FirstOrDefault(x => x.IdOffice == id);

if (data != null)

{

\_context.SortingByOffices.Remove(office);

\_context.Offices.Remove(data);

}

\_context.SaveChanges();

return RedirectToAction(nameof(Offices));

}

//Редактирование записи

[HttpPost]

public IActionResult OfficeEdit(OfficeViewModel off)

{

var edit = \_context.Offices.FirstOrDefault(x => x.IdOffice == off.IdOffice);

edit.IdOffice = off.IdOffice;

edit.TitleOffice = off.TitleOffice;

\_context.SaveChanges();

return RedirectToAction(nameof(Offices));

}

public IActionResult OfficeEdit(int id)

{

var viewmodel = new OfficeViewModel();

var view = \_context.Offices.FirstOrDefault(x => x.IdOffice == id);

if (view != null)

{

viewmodel.IdOffice = view.IdOffice;

viewmodel.TitleOffice = view.TitleOffice;

}

return Json(viewmodel);

}

//Создание записи

[HttpPost]

public IActionResult CreateOffice(OfficeViewModel off)

{

var create = new Office { TitleOffice = off.TitleOffice };

\_context.Offices.Add(create);

\_context.SaveChanges();

return RedirectToAction(nameof(Offices));

}

public IActionResult CreateOffice()

{

return View();

}

#endregion

#region ДЕЙСТВИЯ С ТАБЛИЦЕЙ "ДОЛЖНОСТИ"

//Отображение страницы

public IActionResult Positions(PositionViewModel pos)

{

pos.Positions = \_context.Positions.ToList();

return View(pos);

}

//Удаление записи

public IActionResult DelPos(int id)

{

var data = \_context.Positions.FirstOrDefault(x => x.IdPost == id);

var office = \_context.SortingByOffices.FirstOrDefault(x => x.IdPost == id);

if (data != null)

{

\_context.SortingByOffices.Remove(office);

\_context.Positions.Remove(data);

}

\_context.SaveChanges();

return RedirectToAction(nameof(Positions));

}

//Редактирование записи

[HttpPost]

public IActionResult PosEdit(PositionViewModel pos)

{

var edit = \_context.Positions.FirstOrDefault(x => x.IdPost == pos.IdPost);

edit.IdPost = pos.IdPost;

edit.TitlePost = pos.TitlePost;

\_context.SaveChanges();

return RedirectToAction(nameof(Positions));

}

public IActionResult PosEdit(int id)

{

var viewmodel = new PositionViewModel();

var view = \_context.Positions.FirstOrDefault(x => x.IdPost == id);

if (view != null)

{

viewmodel.IdPost = view.IdPost;

viewmodel.TitlePost = view.TitlePost;

}

return Json(viewmodel);

}

//Создание записи

[HttpPost]

public IActionResult CreatePost(Position pos)

{

var create = new Position { TitlePost = pos.TitlePost };

\_context.Positions.Add(create);

\_context.SaveChanges();

return RedirectToAction(nameof(Positions));

}

public IActionResult CreatePost()

{

return View();

}

#endregion

#region ДЕЙСТВИЯ С ТАБЛИЦЕЙ "ПОМЕЩЕНИЯ"

//Отображение данных

public IActionResult Rooms(RoomViewModel room)

{

room.Rooms = \_context.Rooms.ToList();

room.Levels = \_context.AccessLevels.ToList();

return View(room);

}

//Удаление записи

public IActionResult DelRoom(int id)

{

var data = \_context.Rooms.FirstOrDefault(x => x.IdRoom == id);

var entr = \_context.Entrances.FirstOrDefault(x => x.IdRoom == id);

var door = \_context.Doors.FirstOrDefault(x => x.IdRoom == id);

if (data != null)

{

\_context.Entrances.Remove(entr);

\_context.Doors.Remove(door);

\_context.Rooms.Remove(data);

}

\_context.SaveChanges();

return RedirectToAction(nameof(Rooms));

}

//Редактирование записи

[HttpPost]

public IActionResult RoomEdit(RoomViewModel room)

{

var edit = \_context.Rooms.FirstOrDefault(x => x.IdRoom == room.IdRoom);

edit.IdRoom = room.IdRoom;

edit.TitleRoom = room.TitleRoom;

edit.IdLevel = room.IdLevel;

\_context.SaveChanges();

return RedirectToAction(nameof(Rooms));

}

public IActionResult RoomEdit(int id)

{

var viewmodel = new RoomViewModel();

var view = \_context.Rooms.FirstOrDefault(x => x.IdRoom == id);

if (view != null)

{

viewmodel.IdRoom = view.IdRoom;

viewmodel.TitleRoom = view.TitleRoom;

viewmodel.IdLevel = view.IdLevel;

}

return Json(viewmodel);

}

//Создание записи

[HttpPost]

public IActionResult CreateRoom(RoomViewModel room)

{

var create = new Room { TitleRoom = room.TitleRoom, IdLevel = room.IdLevel };

\_context.Rooms.Add(create);

\_context.SaveChanges();

return RedirectToAction(nameof(Rooms));

}

public IActionResult CreateRoom()

{

var room = new RoomViewModel();

room.Rooms = \_context.Rooms.ToList();

room.Levels = \_context.AccessLevels.ToList();

return View(room);

}

#endregion

#region ДЕЙСТВИЯ С ТАБЛИЦЕЙ "РАСПРЕДЕЛЕНИЕ ПО ОТДЕЛАМ"

//Отображение страницы

public IActionResult SortByOff(SortingByOfficeViewModel sort)

{

sort.Sortings = \_context.SortingByOffices.ToList();

sort.Staffs = \_context.staff.ToList();

sort.Positions = \_context.Positions.ToList();

sort.Offices = \_context.Offices.ToList();

return View(sort);

}

//Удаление записи

public IActionResult DelSort(int id)

{

var data = \_context.SortingByOffices.FirstOrDefault(x => x.IdItem == id);

if (data != null)

{

\_context.SortingByOffices.Remove(data);

}

return RedirectToAction(nameof(SortByOff));

}

//Редактирование записи

[HttpPost]

public IActionResult SortEdit(SortingByOfficeViewModel sort)

{

var edit = \_context.SortingByOffices.FirstOrDefault(x => x.IdItem == sort.IdItem);

edit.IdItem = sort.IdItem;

edit.TimeBegin = sort.TimeBegin;

edit.TimeEnd = sort.TimeEnd;

edit.WorkPhone = sort.WorkPhone;

edit.IdStaff = sort.IdStaff;

edit.IdPost = sort.IdPost;

edit.IdOffice = sort.IdOffice;

\_context.SaveChanges();

return RedirectToAction(nameof(SortByOff));

}

public IActionResult SortEdit(SortingByOfficeViewModel sort, int id)

{

sort.Sortings = \_context.SortingByOffices.ToList();

sort.Staffs = \_context.staff.ToList();

sort.Positions = \_context.Positions.ToList();

sort.Offices = \_context.Offices.ToList();

var view = \_context.SortingByOffices.FirstOrDefault(x => x.IdItem == id);

if (id != 0)

{

var edit = \_context.SortingByOffices.FirstOrDefault(x => x.IdItem == id);

sort.IdItem = edit.IdItem;

sort.TimeBegin = edit.TimeBegin;

sort.TimeEnd = edit.TimeEnd;

sort.WorkPhone = edit.WorkPhone;

sort.IdStaff = edit.IdStaff;

sort.IdPost = edit.IdPost;

sort.IdOffice = edit.IdOffice;

}

return View(sort);

}

//Создание записи

[HttpPost]

public IActionResult CreateSort(SortingByOfficeViewModel sort)

{

var create = new SortingByOffice

{

TimeBegin = sort.TimeBegin,

TimeEnd = sort.TimeEnd,

WorkPhone = sort.WorkPhone,

IdStaff = sort.IdStaff,

IdPost = sort.IdPost,

IdOffice = sort.IdOffice,

};

\_context.SortingByOffices.Add(create);

\_context.SaveChanges();

return RedirectToAction(nameof(SortByOff));

}

public IActionResult CreateSort()

{

var sort = new SortingByOfficeViewModel();

sort.Sortings = \_context.SortingByOffices.ToList();

sort.Staffs = \_context.staff.ToList();

sort.Positions = \_context.Positions.ToList();

sort.Offices = \_context.Offices.ToList();

return View(sort);

}

#endregion

#region ДЕЙСТВИЯ С ТАБЛИЦЕЙ "СОТРУДНИКИ"

//Отображение данных

public IActionResult Staff(StaffViewModel staff)

{

staff.Staffs = \_context.staff.ToList();

staff.Levels = \_context.AccessLevels.ToList();

return View(staff);

}

//Редактирование данных

[HttpPost]

public IActionResult StaffEdit (StaffViewModel staff)

{

var edit = \_context.staff.FirstOrDefault(x => x.IdStaff == staff.IdStaff);

staff.Staffs = \_context.staff.ToList();

staff.Levels = \_context.AccessLevels.ToList();

edit.IdStaff = staff.IdStaff;

edit.Surname = staff.Surname;

edit.Name = staff.Name;

edit.LastName = staff.LastName;

edit.Birthday = staff.Birthday;

edit.CorpEmail = staff.CorpEmail;

edit.MobPhone = staff.MobPhone;

edit.Image = staff.Image;

edit.IdLevel = staff.IdLevel;

\_context.SaveChanges();

return RedirectToAction(nameof(Staff));

}

public IActionResult StaffEdit(StaffViewModel staff, int id)

{

staff.Staffs = \_context.staff.ToList();

staff.Levels = \_context.AccessLevels.ToList();

var view = \_context.staff.FirstOrDefault(x => x.IdStaff == id);

if(id != 0)

{

var edit = \_context.staff.FirstOrDefault(x => x.IdStaff == id);

staff.Staffs = \_context.staff.ToList();

staff.Levels = \_context.AccessLevels.ToList();

staff.IdStaff = edit.IdStaff;

staff.Surname = edit.Surname;

staff.Name = edit.Name;

staff.LastName = edit.LastName;

staff.Birthday = edit.Birthday;

staff.CorpEmail = edit.CorpEmail;

staff.MobPhone = edit.MobPhone;

staff.Image = edit.Image;

staff.IdLevel = edit.IdLevel;

}

return View(staff);

}

//Удаление записи

public IActionResult DelStaff (int id)

{

var data = \_context.staff.FirstOrDefault(x => x.IdStaff == id);

var sort = \_context.SortingByOffices.FirstOrDefault(x => x.IdStaff == id);

var entr = \_context.Entrances.FirstOrDefault(x => x.IdStaff == id);

if (data != null)

{

\_context.Entrances.Remove(entr);

\_context.SortingByOffices.Remove(sort);

\_context.staff.Remove(data);

}

\_context.SaveChanges();

return RedirectToAction(nameof(Staff));

}

//Создание записи

[HttpPost]

public IActionResult CreateStaff (StaffViewModel staff)

{

var create = new staff

{

Surname = staff.Surname,

Name = staff.Name,

LastName = staff.LastName,

Birthday = staff.Birthday,

CorpEmail = staff.CorpEmail,

MobPhone = staff.MobPhone,

Image = staff.Image,

IdLevel = staff.IdLevel,

};

\_context.staff.Add(create);

\_context.SaveChanges();

return RedirectToAction(nameof(Staff));

}

public IActionResult CreateStaff()

{

var staff = new StaffViewModel();

staff.Staffs = \_context.staff.ToList();

staff.Levels = \_context.AccessLevels.ToList();

return View(staff);

}

#endregion

#region ДЕЙСТВИЯ С ТАБЛИЦЕЙ "ДОСТУП"

//Отображение данных

public IActionResult Entrance(EntranceViewModel entr)

{

entr.Entrances = \_context.Entrances.ToList();

entr.Staffs = \_context.staff.ToList();

entr.Rooms = \_context.Rooms.ToList();

entr.Doors = \_context.Doors.ToList();

entr.Statuses = \_context.AccessStatuses.ToList();

return View(entr);

}

public IActionResult ViewStaff(int id)

{

var viewmodel = new StaffViewModel();

var view = \_context.staff.FirstOrDefault(x => x.IdStaff == id);

if(view != null)

{

viewmodel.IdStaff = view.IdStaff;

viewmodel.Surname = view.Surname;

viewmodel.Name = view.Name;

viewmodel.LastName = view.LastName;

viewmodel.Birthday = view.Birthday;

viewmodel.CorpEmail = view.CorpEmail;

viewmodel.MobPhone = view.MobPhone;

viewmodel.Image = view.Image;

}

return Json(viewmodel);

}

#endregion

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error(int code)

{

if (code == 404)

{

return View("Error");

}

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}

Класс-контроллер CabinetController.cs

namespace EntranceControlWeb.Controllers

{

public class CabinetController : Controller

{

private EntranceControlWebContext \_context;

private readonly IWebHostEnvironment \_appEnvironment;

public CabinetController(EntranceControlWebContext context, IWebHostEnvironment appEnvironment)

{

\_appEnvironment = appEnvironment;

\_context = context;

}

[Authorize]

public IActionResult UserCab()

{

return View();

}

[Authorize]

public IActionResult AdminCab(EntranceViewModel entr)

{

entr.Entrances = \_context.Entrances.ToList();

entr.Staffs = \_context.staff.ToList();

entr.Rooms = \_context.Rooms.ToList();

entr.Doors = \_context.Doors.ToList();

entr.Statuses = \_context.AccessStatuses.ToList();

return View(entr);

}

[Authorize]

public IActionResult SysAdminCab(EntranceViewModel entr)

{

entr.Entrances = \_context.Entrances.ToList();

entr.Staffs = \_context.staff.ToList();

entr.Rooms = \_context.Rooms.ToList();

entr.Doors = \_context.Doors.ToList();

entr.Statuses = \_context.AccessStatuses.ToList();

return View(entr);

}

[Authorize]

public IActionResult Users(AuthorizeViewModel auth)

{

auth.Authorizes = \_context.Authorizes.ToList();

auth.Users = \_context.Users.ToList();

if (User.IsInRole(UserRole.Admin.ToString()))

{

auth.Authorizes = \_context.Authorizes

.Where(x => x.IdUsers.UserRole == UserRole.User).ToList();

}

else

{

auth.Authorizes = \_context.Authorizes.ToList();

}

return View(auth);

}

[Authorize]

[HttpPost]

public IActionResult UserReport()

{

if (User.IsInRole(UserRole.Admin.ToString()))

{

ExcelPackage.LicenseContext = LicenseContext.NonCommercial;

FileStream fs = new("UserReport.xls", FileMode.Create);

using (ExcelPackage Ep = new(fs))

{

var Sheet1 = Ep.Workbook.Worksheets.Add("Посетители сайта");

Sheet1.Cells["A1"].Value = "ID\_Запись";

Sheet1.Cells["B1"].Value = "ДатаАвторизации";

Sheet1.Cells["C1"].Value = "Пользователь";

var row1 = 2;

foreach (var entrance in \_context.Authorizes

.Where(x => x.IdUsers.UserRole == UserRole.User).ToList())

{

\_context.Users.ToList();

Sheet1.Cells[string.Format("A{0}", row1)].Value = entrance.IdItem;

Sheet1.Cells[string.Format("B{0}", row1)].Value = entrance.DateAuth.ToString();

Sheet1.Cells[string.Format("C{0}", row1)].Value = entrance.IdUsers.Email;

row1++;

}

Sheet1.Cells["A:AZ"].AutoFitColumns();

Response.ContentType = "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet";

var xlFile = new FileInfo(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), "UserReport.xls"));

Ep.Save();

fs.Close();

}

string file\_path = Path.Combine(\_appEnvironment.ContentRootPath, "UserReport.xls");

// Тип файла - content-type

string file\_type = "application/octet-stream";

// Имя файла - необязательно

string file\_name = "UserReport.xls";

return PhysicalFile(file\_path, file\_type, file\_name);

}

else

{

ExcelPackage.LicenseContext = LicenseContext.NonCommercial;

FileStream fs = new FileStream("UserReport.xls", FileMode.Create);

using (ExcelPackage Ep = new ExcelPackage(fs))

{

var Sheet1 = Ep.Workbook.Worksheets.Add("Посетители сайта");

Sheet1.Cells["A1"].Value = "ID\_Запись";

Sheet1.Cells["B1"].Value = "ДатаАвторизации";

Sheet1.Cells["C1"].Value = "Пользователь";

var row1 = 2;

foreach (var entrance in \_context.Authorizes.ToList())

{

\_context.Users.ToList();

Sheet1.Cells[string.Format("A{0}", row1)].Value = entrance.IdItem;

Sheet1.Cells[string.Format("B{0}", row1)].Value = entrance.DateAuth.ToString();

Sheet1.Cells[string.Format("C{0}", row1)].Value = entrance.IdUsers.Email;

row1++;

}

Sheet1.Cells["A:AZ"].AutoFitColumns();

Response.ContentType = "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet";

var xlFile = new FileInfo(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), "UserReport.xls"));

Ep.Save();

fs.Close();

}

string file\_path = Path.Combine(\_appEnvironment.ContentRootPath, "UserReport.xls");

// Тип файла - content-type

string file\_type = "application/octet-stream";

// Имя файла - необязательно

string file\_name = "UserReport.xls";

return PhysicalFile(file\_path, file\_type, file\_name);

}

}

[Authorize]

public IActionResult Chart()

{

var data = \_context.Entrances

.GroupBy(p => p.DateEntr.Day)

.Select(g => new ChartItemViewModel { ID = g.Key, Count = g.Count() })

.ToList();

var VM = new ChartViewModel

{

Chart = data,

};

return View(VM);

}

[Authorize]

[HttpPost]

public IActionResult EntranceReport()

{

ExcelPackage.LicenseContext = LicenseContext.NonCommercial;

FileStream fs = new FileStream("EntranceReport.xls", FileMode.Create);

using (ExcelPackage Ep = new ExcelPackage(fs))

{

var Sheet1 = Ep.Workbook.Worksheets.Add("Доступ");

Sheet1.Cells["A1"].Value = "ID\_Запись";

Sheet1.Cells["B1"].Value = "Дата\_Входа";

Sheet1.Cells["C1"].Value = "Дата\_Выхода";

Sheet1.Cells["D1"].Value = "Помещение";

Sheet1.Cells["E1"].Value = "Сотрудник";

Sheet1.Cells["F1"].Value = "ТипТурникета";

Sheet1.Cells["G1"].Value = "СтатусВхода";

var row1 = 2;

foreach (var entrance in \_context.Entrances.ToList())

{

\_context.staff.ToList();

\_context.Rooms.ToList();

\_context.Doors.ToList();

\_context.AccessStatuses.ToList();

Sheet1.Cells[string.Format("A{0}", row1)].Value = entrance.IdRecord;

Sheet1.Cells[string.Format("B{0}", row1)].Value = entrance.DateEntr.ToString();

Sheet1.Cells[string.Format("C{0}", row1)].Value = entrance.DateExit.ToString();

Sheet1.Cells[string.Format("D{0}", row1)].Value = entrance.IdRooms.TitleRoom;

Sheet1.Cells[string.Format("E{0}", row1)].Value = entrance.IdStaffs.Surname;

Sheet1.Cells[string.Format("F{0}", row1)].Value = entrance.IdDoors.TitleDoor;

Sheet1.Cells[string.Format("G{0}", row1)].Value = entrance.IdStatusS.TitleStatus;

row1++;

}

Sheet1.Cells["A:AZ"].AutoFitColumns();

var Sheet2 = Ep.Workbook.Worksheets.Add("Сотрудники");

Sheet2.Cells["A1"].Value = "ID\_Сотрудника";

Sheet2.Cells["B1"].Value = "ФамилияСотрудника";

Sheet2.Cells["C1"].Value = "ИмяСотрудника";

Sheet2.Cells["D1"].Value = "ОтчетсвоСотрудника";

Sheet2.Cells["E1"].Value = "ДатаРождения";

Sheet2.Cells["F1"].Value = "КорпоративнаяПочта";

Sheet2.Cells["G1"].Value = "ТелефонМобильный";

Sheet2.Cells["H1"].Value = "Изображение";

Sheet2.Cells["I1"].Value = "ID\_УровеньДоступа";

var row2 = 2;

foreach (var entrance in \_context.staff.ToList())

{

\_context.AccessLevels.ToList();

Sheet2.Cells[string.Format("A{0}", row2)].Value = entrance.IdStaff;

Sheet2.Cells[string.Format("B{0}", row2)].Value = entrance.Surname;

Sheet2.Cells[string.Format("C{0}", row2)].Value = entrance.Name;

Sheet2.Cells[string.Format("D{0}", row2)].Value = entrance.LastName;

Sheet2.Cells[string.Format("E{0}", row2)].Value = entrance.Birthday.ToString();

Sheet2.Cells[string.Format("F{0}", row2)].Value = entrance.CorpEmail;

Sheet2.Cells[string.Format("G{0}", row2)].Value = entrance.MobPhone;

Sheet2.Cells[string.Format("H{0}", row2)].Value = entrance.Image;

Sheet2.Cells[string.Format("I{0}", row2)].Value = entrance.IdLevels.TitleLevel;

row2++;

}

Sheet2.Cells["A:AZ"].AutoFitColumns();

var Sheet3 = Ep.Workbook.Worksheets.Add("Должности");

Sheet3.Cells["A1"].Value = "ID\_Должности";

Sheet3.Cells["B1"].Value = "НаименованиеДолжности";

var row3 = 2;

foreach (var entrance in \_context.Positions.ToList())

{

Sheet3.Cells[string.Format("A{0}", row3)].Value = entrance.IdPost;

Sheet3.Cells[string.Format("B{0}", row3)].Value = entrance.TitlePost;

row3++;

}

Sheet3.Cells["A:AZ"].AutoFitColumns();

var Sheet4 = Ep.Workbook.Worksheets.Add("Отделы");

Sheet4.Cells["A1"].Value = "ID\_Отдела";

Sheet4.Cells["B1"].Value = "НаименованиеОтдела";

var row4 = 2;

foreach (var entrance in \_context.Offices.ToList())

{

Sheet4.Cells[string.Format("A{0}", row4)].Value = entrance.IdOffice;

Sheet4.Cells[string.Format("B{0}", row4)].Value = entrance.TitleOffice;

row4++;

}

Sheet4.Cells["A:AZ"].AutoFitColumns();

var Sheet5 = Ep.Workbook.Worksheets.Add("РаспределениеПоОтделам");

Sheet5.Cells["A1"].Value = "ID\_Запись";

Sheet5.Cells["B1"].Value = "Сотрудник";

Sheet5.Cells["C1"].Value = "Отдел";

Sheet5.Cells["D1"].Value = "Должности";

Sheet5.Cells["E1"].Value = "ВремяНачалаРаботы";

Sheet5.Cells["F1"].Value = "ВремяЗавершенияРаботы";

Sheet5.Cells["G1"].Value = "ТелефонРабочий";

var row5 = 2;

foreach (var entrance in \_context.SortingByOffices.ToList())

{

\_context.staff.ToList();

\_context.Offices.ToList();

\_context.Positions.ToList();

Sheet5.Cells[string.Format("A{0}", row5)].Value = entrance.IdItem;

Sheet5.Cells[string.Format("B{0}", row5)].Value = entrance.IdStaffs.Surname;

Sheet5.Cells[string.Format("C{0}", row5)].Value = entrance.IdOffices.TitleOffice;

Sheet5.Cells[string.Format("D{0}", row5)].Value = entrance.IdPosts.TitlePost;

Sheet5.Cells[string.Format("E{0}", row5)].Value = entrance.TimeBegin.ToString();

Sheet5.Cells[string.Format("F{0}", row5)].Value = entrance.TimeEnd.ToString();

Sheet5.Cells[string.Format("G{0}", row5)].Value = entrance.WorkPhone;

row5++;

}

Sheet5.Cells["A:AZ"].AutoFitColumns();

var Sheet6 = Ep.Workbook.Worksheets.Add("Помещения");

Sheet6.Cells["A1"].Value = "ID\_Помещения";

Sheet6.Cells["B1"].Value = "НаименованиеПомещения";

Sheet6.Cells["C1"].Value = "УровеньДоступа";

var row6 = 2;

foreach (var entrance in \_context.Rooms.ToList())

{

\_context.AccessLevels.ToList();

Sheet6.Cells[string.Format("A{0}", row6)].Value = entrance.IdRoom;

Sheet6.Cells[string.Format("B{0}", row6)].Value = entrance.TitleRoom;

Sheet6.Cells[string.Format("C{0}", row6)].Value = entrance.IdLevels.TitleLevel;

row6++;

}

Sheet6.Cells["A:AZ"].AutoFitColumns();

var Sheet7 = Ep.Workbook.Worksheets.Add("ТипыТурникетов");

Sheet7.Cells["A1"].Value = "ID\_ТипТурникета";

Sheet7.Cells["B1"].Value = "НаименованиеТипаТурникета";

Sheet7.Cells["C1"].Value = "Помещение";

var row7 = 2;

foreach (var entrance in \_context.Doors.ToList())

{

\_context.Rooms.ToList();

Sheet7.Cells[string.Format("A{0}", row7)].Value = entrance.IdDoor;

Sheet7.Cells[string.Format("B{0}", row7)].Value = entrance.TitleDoor;

Sheet7.Cells[string.Format("C{0}", row7)].Value = entrance.IdRooms.TitleRoom;

row7++;

}

Sheet7.Cells["A:AZ"].AutoFitColumns();

var Sheet8 = Ep.Workbook.Worksheets.Add("СтатусДоступа");

Sheet8.Cells["A1"].Value = "ID\_Статус";

Sheet8.Cells["B1"].Value = "НаименованиеСтатус";

var row8 = 2;

foreach (var entrance in \_context.AccessStatuses.ToList())

{

Sheet8.Cells[string.Format("A{0}", row8)].Value = entrance.IdStatus;

Sheet8.Cells[string.Format("B{0}", row8)].Value = entrance.TitleStatus;

row8++;

}

Sheet8.Cells["A:AZ"].AutoFitColumns();

var Sheet9 = Ep.Workbook.Worksheets.Add("УровеньДоступа");

Sheet9.Cells["A1"].Value = "ID\_Уровень";

Sheet9.Cells["B1"].Value = "НаименованиеУровень";

var row9 = 2;

foreach (var entrance in \_context.AccessLevels.ToList())

{

Sheet9.Cells[string.Format("A{0}", row9)].Value = entrance.IdLevel;

Sheet9.Cells[string.Format("B{0}", row9)].Value = entrance.TitleLevel;

row9++;

}

Sheet9.Cells["A:AZ"].AutoFitColumns();

Response.ContentType = "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet";

var xlFile = new FileInfo(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), "EntranceReport.xls"));

Ep.Save();

fs.Close();

}

string file\_path = Path.Combine(\_appEnvironment.ContentRootPath, "EntranceReport.xls");

// Тип файла - content-type

string file\_type = "application/octet-stream";

// Имя файла - необязательно

string file\_name = "EntranceReport.xls";

return PhysicalFile(file\_path, file\_type, file\_name);

}

}

}

Страница представления Room.cshtml

@model EntranceControlWeb.Models.RoomViewModel;

@{

ViewData["Title"] = "Rooms";

}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>@ViewData["Title"]</title>

<link rel="icon" href="/img/icon.ico" sizes="64x64" />

<link rel="stylesheet" href="~/css/bootstrap.min.css" />

<script src="~/js/bootstrap.bundle.min.js"></script>

<script src="~/lib/jquery-validation/dist/jquery.validate.js"></script>

<script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.js"></script>

<link rel="stylesheet" href="~/css/site.css" />

<link rel="preconnect" href="https://fonts.googleapis.com">

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

<link href="https://fonts.googleapis.com/css2?family=Fira+Code&display=swap" rel="stylesheet">

<script src="~/lib/jquery/dist/jquery.min.js"></script>

<script src="~/js/site.js" asp-append-version="true"></script>

@\*@await RenderSectionAsync("Scripts", required: false)\*@

</head>

<body>

<header>

<nav class="navbar navbar-light sticky-top" style="background-color: #fbfeff">

<div class="container-fluid">

<button class="navbar-toggler" type="button" data-bs-toggle="offcanvas" data-bs-target="#offcanvasNavbar" aria-controls="offcanvasNavbar">

<span class="navbar-toggler-icon"></span>

</button>

@if (User.Identity.IsAuthenticated)

{

<a class="nav-link text-primary">Аккаунт: @User.Identity.Name </a>

}

<a class="nav-link text-danger" asp-area="" asp-controller="Login" asp-action="LogOut">Выйти</a>

<div class="offcanvas offcanvas-start" tabindex="-1" id="offcanvasNavbar" aria-labelledby="offcanvasNavbarLabel">

<div class="offcanvas-header">

<h5 class="offcanvas-title" id="offcanvasNavbarLabel">Навигация в системе</h5>

<button type="button" class="btn-close text-reset" data-bs-dismiss="offcanvas" aria-label="Close"></button>

</div>

<div class="offcanvas-body">

<ul class="navbar-nav justify-content-start flex-grow-1 pe-3">

<li class="nav-item">

<a class="nav-link active" asp-controller="Home" asp-action="Entrance">Доступ</a>

</li>

@if (User.Identity.IsAuthenticated)

{

if (User.IsInRole(UserRole.Admin.ToString()))

{

<li class="nav-item">

<a class="nav-link active" asp-controller="Home" asp-action="Staff">Сотрудники</a>

</li>

<li class="nav-item">

<a class="nav-link active" asp-controller="Home" asp-action="Positions">Должности</a>

</li>

<li class="nav-item">

<a class="nav-link active" asp-controller="Home" asp-action="Offices">Отделы</a>

</li>

<li class="nav-item">

<a class="nav-link active" asp-controller="Home" asp-action="SortByOff">Распределение по отделам</a>

</li>

<li class="nav-item">

<a class="nav-link active" asp-controller="Home" asp-action="Rooms">Помещения</a>

</li>

<li class="nav-item">

<a class="nav-link active" asp-controller="Home" asp-action="Doors">Турникеты</a>

</li>

<li class="nav-item">

<a class="nav-link active" asp-controller="Home" asp-action="Levels">Настройка уровня доступа</a>

</li>

}

else if (User.IsInRole(UserRole.SysAdmin.ToString()))

{

<li class="nav-item">

<a class="nav-link active" asp-controller="Home" asp-action="Staff">Сотрудники</a>

</li>

<li class="nav-item">

<a class="nav-link active" asp-controller="Home" asp-action="Positions">Должности</a>

</li>

<li class="nav-item">

<a class="nav-link active" asp-controller="Home" asp-action="Offices">Отделы</a>

</li>

<li class="nav-item">

<a class="nav-link active" asp-controller="Home" asp-action="SortByOff">Распределение по отделам</a>

</li>

<li class="nav-item">

<a class="nav-link active" asp-controller="Home" asp-action="Rooms">Помещения</a>

</li>

<li class="nav-item">

<a class="nav-link active" asp-controller="Home" asp-action="Doors">Турникеты</a>

</li>

<li class="nav-item">

<a class="nav-link active" asp-controller="Home" asp-action="Levels">Настройка уровня доступа</a>

</li>

}

}

</ul>

<hr />

<ul class="navbar-nav justify-content-start flex-grow-1 pe-3">

@if (User.Identity.IsAuthenticated)

{

if (User.IsInRole(UserRole.User.ToString()))

{

<li class="nav-item">

<a class="nav-link active" asp-controller="Cabinet" asp-action="UserCab">Личный кабинет</a>

</li>

}

else if (User.IsInRole(UserRole.Admin.ToString()))

{

<li class="nav-item">

<a class="nav-link active" asp-controller="Cabinet" asp-action="AdminCab">Личный кабинет</a>

</li>

}

else if (User.IsInRole(UserRole.SysAdmin.ToString()))

{

<li class="nav-item">

<a class="nav-link active" asp-controller="Cabinet" asp-action="SysAdminCab">Личный кабинет</a>

</li>

}

}

@if (User.Identity.IsAuthenticated)

{

if (User.IsInRole(UserRole.User.ToString()))

{

<li class="nav-item" style="position: relative; left: 950px">

<a class="nav-link active" href="~/RukUser.pdf" target="\_blank">Руководство пользователя</a>

</li>

}

else if (User.IsInRole(UserRole.Admin.ToString()))

{

<li class="nav-item">

<a class="nav-link active" href="~/RukAdmin.pdf" target="\_blank">Руководство начальника</a>

</li>

}

else if (User.IsInRole(UserRole.SysAdmin.ToString()))

{

<li class="nav-item">

<a class="nav-link active" href="~/RukSysAdmin.pdf" target="\_blank">Руководство администратора</a>

</li>

}

}

<li class="nav-item">

<a class="nav-link active text-danger" asp-controller="Login" asp-action="LogOut">Выход</a>

</li>

</ul>

</div>

</div>

</div>

</nav>

</header>

<div class="container">

<div class="row my-3">

<h3 class="text-center"><strong>CИСТЕМА КОНТРОЛЯ ДОСТУПА</strong></h3>

<h5 class="text-center">ПОМЕЩЕНИЯ</h5>

</div>

<div class="row my-3">

<table class="table-dark table-bordered" style="width:1200px; color:black;">

<thead>

<tr class="text-center">

<th>ID Помещение</th>

<th>Наименование помещения</th>

<th>Уровень доступа</th>

@if (User.Identity.IsAuthenticated)

{

if (User.IsInRole(UserRole.SysAdmin.ToString()))

{

<th>Действия</th>

}

else if (User.IsInRole(UserRole.Admin.ToString()))

{

<th>Действия</th>

}

}

</tr>

</thead>

@foreach (var data in Model.Rooms)

{

<tbody>

<tr class="text-md-start">

<td>@data.IdRoom</td>

<td>@data.TitleRoom</td>

<td>@data.IdLevels.TitleLevel</td>

@if (User.Identity.IsAuthenticated)

{

if (User.IsInRole(UserRole.SysAdmin.ToString()))

{

<td>

<a href="@Url.Action("RoomEdit", "Home", new { id = @data.IdRoom })" class="text-success editClick" data-bs-toggle="modal" data-bs-target="#exampleModal" data-id="@data.IdRoom">Редактировать</a>

<a href="@Url.Action("DelRoom", "Home", new { id = @data.IdRoom })" class="text-danger" data-id="@data.IdRoom">Удалить</a>

</td>

}

else if (User.IsInRole(UserRole.Admin.ToString()))

{

<td>

<a href="@Url.Action("RoomEdit", "Home", new { id = @data.IdRoom })" class="text-success editClick" data-bs-toggle="modal" data-bs-target="#exampleModal" data-id="@data.IdRoom">Редактировать</a>

</td>

}

}

</tr>

</tbody>

}

</table>

@if (User.Identity.IsAuthenticated)

{

if (User.IsInRole(UserRole.SysAdmin.ToString()))

{

<div class="row m-2">

<div class="col-2"></div>

<div class="col-8"> <a href="@Url.Action("CreateRoom", "Home")" class="btn btn-outline-primary" style="width: 100%">Создать запись</a></div>

<div class="col-2"></div>

</div>

}

}

<script>

$('.editClick').on('click', function () {

// ...

console.log('edit clicked');

const roomId = $(this).data('id');

// обращение на сервер

$.get('/Home/RoomEdit/' + roomId, function (result) {

// результат содержит данные о двери

$('#exampleModal #inputTitle').val(result.idRoom);

$('#exampleModal #inputTitle1').val(result.titleRoom);

$('#exampleModal #inputTitle2').val(result.idLevel);

});

});

</script>

<!-- Modal -->

<div class="modal fade" id="exampleModal" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">

<div class="modal-dialog modal-dialog-centered">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalLabel">Редактировать информацию</h5>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

</div>

<form method="post" asp-action="RoomEdit" asp-controller="Home">

<div class="modal-body">

<div class="row mb-3">

<input hidden readonly asp-for="@Model.IdRoom" value="@Model?.IdRoom" type="text" class="form-control" id="inputTitle">

<label for="inputTitle1" class="col-sm-4 col-form-label">Наименование помещения</label>

<div class="col-sm-8">

<input asp-for="@Model.TitleRoom" value="@Model?.TitleRoom" type="text" class="form-control" id="inputTitle1">

<span for="inputTitle" class="text-danger" asp-validation-for="@Model.TitleRoom"></span>

</div>

</div>

<div class="row mb-3">

<label for="#inputTitle2" class="col-sm-4 col-form-label">Уровень доступа</label>

<div class="col-sm-8">

<select name="idLevel" class="form-control" id="#inputTitle2">

@foreach (var data in Model.Rooms)

{

<option value="@data.IdLevel">@data.IdLevels.TitleLevel</option>

}

</select>

<span for="idRoom" class="text-danger" asp-validation-for="@Model.IdLevel"></span>

<span for="idRoom" class="text-danger" asp-validation-for="@Model.TitleLevel"></span>

</div>

</div>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-outline-dark" data-bs-dismiss="modal">Отмена</button>

<button type="submit" class="btn btn-success">Сохранить</button>

</div>

</form>

</div>

</div>

</div>

</div>

</div>

</body>

</html>