Министерство образования и науки Республики Башкортостан

Государственное автономное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и вычислительной техники

УТВЕРЖДАЮ

Заместитель директора

по учебной работе

\_\_\_\_\_\_\_\_\_\_\_\_З.З. Курмашева

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024г.

ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ И РАЗРАБОТКА ПРИЛОЖЕНИЯ

ДЛЯ ФОРМИРОВАНИЯ СВЕДЕНИЙ ПО ОТРАБОТАННЫМ ЧАСАМ В СИСТЕМЕ БЫСТРОГО ПИТАНИЯ

Пояснительная записка к курсовому проекту

МДК.11.01 Технология разработки и защиты баз данных

Руководитель проекта

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Р.Ф. Каримова

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Студент гр. 21П-1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.А. КУРБАНОВ

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

2024

Министерство образования и науки Республики Башкортостан

Государственное автономное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и вычислительной техники

УТВЕРЖДАЮ

Заместитель директора

по учебной работе

\_\_\_\_\_\_\_\_\_\_\_\_\_З.З.Курмашева

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

ЗАДАНИЕ

на курсовой проект студенту дневного отделения, группы 21П-1, специальности 09.02.07 Информационные системы и программирование.

Фамилия, имя, отчество: Курбанов Айдар Артурович

Тема курсового проекта: «Проектирование баз данных и разработка приложения для формирования сведений по отработанным часам в системе быстрого питания».

Текст задания:

при выполнении курсового проекта должны быть решены следующие задачи:

а) спроектирована база данных;

б) разработана структура программы;

в) реализованы функции авторизация сотрудника и менеджера, фиксация времени входа и выхода сотрудника, вывод времени входа и выхода, вывод сотрудников, вывод должностей, добавление, удаление, редактирование должностей и сотрудников, фильтрация по дате входа сотрудника.

В результате выполнения курсового проекта должны быть представлены:

а) пояснительная записка, состоящая из следующих разделов:

Введение

1 Постановка задачи

2 Экспериментальный раздел

Заключение

Приложения

Список сокращений

Список источников

б) электронный носитель, содержащий разработанный программный продукт;

в) презентация курсового проекта в электронном виде.

Список рекомендуемых источников:

1. ГОСТ 34.321-96 Информационные технологии. Система стандартов по базам данных. Эталонная модель управления данными: межгосударственный стандарт: дата введения 2001-07-01 - <https://internet-law.ru/gosts/gost/6808/?ysclid=led2xw9tta320329480> - Текст : электронный.
2. Основы проектирования баз данных : учебник для студ. учреждений сред. проф. Образования Г. Н. Федорова. - М.: Издательский центр «Академия», 2017. - 224 с.
3. Документация по MySQL. URL :<https://dev.mysql.com/doc/>
4. Metanit: Сайт о программировании. URL: <https://metanit.com/>

Задание к выполнению получил «31» января 2024 г.

Студент Курбанов Айдар Артурович

Срок окончания «31» мая 2024 г.

Руководитель курсового проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Р.Ф. Каримова

Задание рассмотрено на заседании цикловой комиссии информатики

«11» января 2024г.

Председатель цикловой комиссии информатики \_\_\_\_\_\_\_ О.В. Фатхулова

Министерство образования и науки Республики Башкортостан

Государственное бюджетное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и вычислительной техники

ЗАКЛЮЧЕНИЕ

на курсовой проект

Студент Курбанов Айдар Артурович

Группа 21П-1

Специальность 09.02.07 Информационные системы и программирование

Тема Проектирование базы данных для формирования сведений по отработанным часам в системе быстрого питания

Объем курсового проекта:

количество листов пояснительной записки

количество листов графической части

Заключение о степени соответствия заданию на курсовое проектирование

Характеристика качеств, проявленных студентом при работе над проектом: самостоятельность, дисциплинированность, умение планировать работу и пользоваться литературным материалом и т.д.

Положительные стороны курсового проекта

Недостатки курсового проекта

Характеристика общетехнической и специальной подготовки студента

Заключение и предлагаемая оценка за курсовой проект

Руководитель курсового проекта Каримова Резида Флюновна

«\_\_\_\_» 2024 г.

Подпись

АННОТАЦИЯ

Пояснительная записка к курсовому проекту содержит постановку и программу решения задачи «Проектирование базы данных для сведеней по отработанным часам в системе быстрого питания».

Программа WorkingHours.exe написана на языке C# в среде программирования VisualStudio 2019 с использованием системы управления базой данных MySQL Workbench, предназначена для работы в операционной системе MSWindows7 и выше, отлажена на данных контрольного примера.

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 3](#_Toc169851404)

[ВВЕДЕНИЕ 4](#_Toc169851405)

[1. Постановка задачи 5](#_Toc169851406)

[1.1Описание предметной области 5](#_Toc169851407)

[1.2 Описание входной информации: 6](#_Toc169851408)

[1.3 Описание выходной информации: 6](#_Toc169851409)

[1.4. Концептуальное моделирование 6](#_Toc169851410)

[1.5. Логическое моделирование 7](#_Toc169851411)

[1.6. Описание структуры базы данных 9](#_Toc169851412)

[2. Экспериментальный раздел 13](#_Toc169851413)

[2.1 Описание программы 13](#_Toc169851414)

[2.2 Протокол тестирования программного продукта 19](#_Toc169851415)

[2.3 Руководство пользователя 29](#_Toc169851416)

[ЗАКЛЮЧЕНИЕ 38](#_Toc169851417)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 3](#_Toc169851418)

[Приложение А 3](#_Toc169851419)

# ВВЕДЕНИЕ

В современном мире системы быстрого питания становятся все более популярными, что приводит к увеличению количества данных, связанных с отработанными часами сотрудников. Проектирование базы данных и разработка приложения для формирования сведений по отработанным часам в системе быстрого питания играют важную роль в оптимизации управления персоналом и ресурсами. Это позволяет эффективно отслеживать рабочее время, управлять графиками смен и оплатой труда, а также создавать отчеты для анализа производительности. В данном контексте проектирование базы данных и разработка приложения являются ключевыми шагами для автоматизации учета отработанных часов и обеспечения эффективного управления персоналом в системе быстрого питания. Задачами курсового проекта являются:

* описать предметную область;
* разработать структуру базы данных;
* разработать приложение;
* провести тестирование приложения.

# 1. Постановка задачи

# 1.1Описание предметной области

Требуется разработать информационную систему для автоматизации учета входа и выхода человека на работу, так же учета заработной платы и количество часов проведенных на работе.

Данная информационная система предполагает наличие двух групп пользователей: работник и менеджер.

Работник имеет возможность авторизоваться для учета времени входа на работу и выхода с работы.

Для того что бы авторизоваться ему надо заполнить следующие данные:

* Логин
* Пароль

Все данные берутся и хранятся в базе данных

Администратор может просматривать все входы и выходы рабочих, добавлять новых рабочих, удалять и изменять старых, также фильтровать рабочих по дате прихода, добавлять новые должности, удалять и изменять старые.

Добавление рабочих:

* ID
* Пароль
* Фамилия
* Имя
* Отчество
* Должность

В базе данных должны храниться следующие данные: рабочие, время на работе, должности, роль.

В таблице «Работник» хранятся следующие данные:

* ID – номер работника первичный ключ
* Логин - по нему происходит авторизация
* Пароль – по нему происходит авторизация
* Фамилия
* Имя
* Отчество
* Должность

В таблице «Время» хранятся следующие данные:

* ID – номер работника вторичный ключ
* Время прихода
* Время ухода

В таблице «Должность» хранятся следующие данные:

* ID – номер должности
* Название должности

# 1.2 Описание входной информации:

Входной информацией является добавление рабочих и должностей

# 1.3 Описание выходной информации:

* Информация о работнике
* Время на работе

# 1.4. Концептуальное моделирование

Концептуальная модель базы данных - это некая наглядная диаграмма, нарисованная в принятых обозначениях и подробно показывающая связь между объектами и их характеристиками. Создается концептуальная модель для дальнейшего проектирования базы данных и перевод ее, например, в реляционную базу данных. На концептуальной модели в визуально удобном виде прописываются связи между объектами данных и их характеристиками.

В концептуальной модели есть принятые обозначения элементов. Сущность или объект обозначать прямоугольником, отношения обозначать ромбом, атрибуты объектов, обозначаются овалом. Если сущность связана с отношением, то их связь обозначается прямой линией со стрелкой. Необязательная связь обозначается пунктирной линией. Мощная связь обозначается двойной линией.

Концептуальная модель базы данных представлена в рисунке 1.4.1

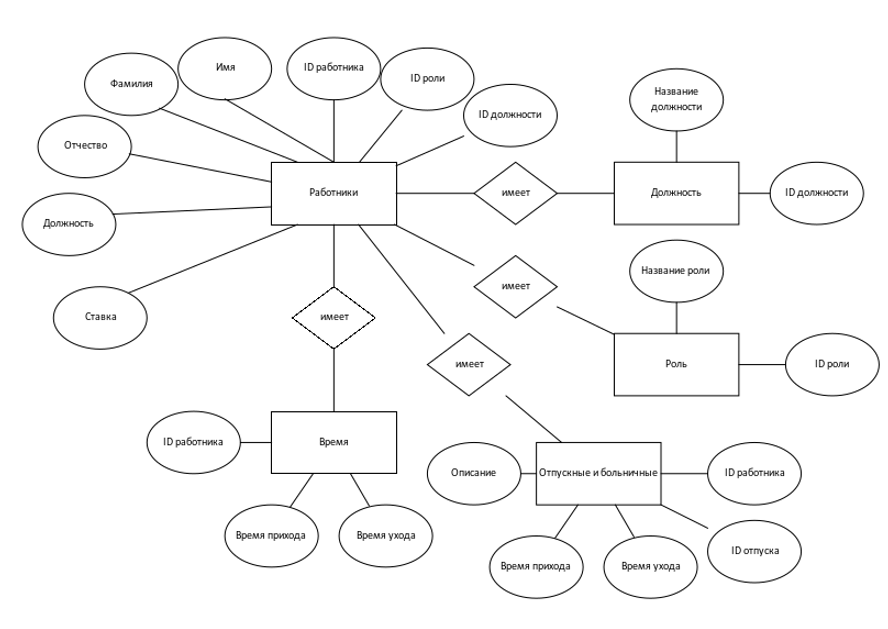


Рисунок 1.4.1 – Концептуальная модель БД

# 1.5. Логическое моделирование

При логическом моделировании происходит окончательное определение структуры данных, определяются ограничения, накладываемые на эти данные, целью которых является обеспечить целостность данных. Наиболее распространенной моделью данных является реляционная модель. В этой модели данных каждая сущность представляется в виде таблицы.

Логическое моделирование заключается в переходе от концептуальной модели к взаимосвязанным таблицам. Этот переход состоит из следующих шагов:

1. Преобразование сущностей:

- каждая простая сущность становится таблицей.

- каждый атрибут становится столбцом таблицы.

- уникальный идентификатор сущности становится ключом таблицы.

2. Преобразование связи:

- сущности, связанные обязательной связью один к одному можно объединить в одну таблицу.

- связи типа один к одному возможные и связи типа один ко многим реализуются путем переноса ключевых атрибутов таблиц, соответствующих сущностей, стоящих со стороны один в таблице соответствующих сущностей, стоящих со стороны многие.

- связи типа многие ко многим реализуются при помощи промежуточных таблиц, содержащих ключевые атрибуты связываемых таблиц в качестве внешних ключей.

Схема данных – это структура базы данных, описанная на формальном языке, поддерживаемом СУБД (системой управления базы данных). В реляционных базах данных схема определяет таблицы, поля в каждой таблице и ограничения целостности, такие как первичный и внешний ключи.

Схема данных представлена на рисунке 1.5.1.

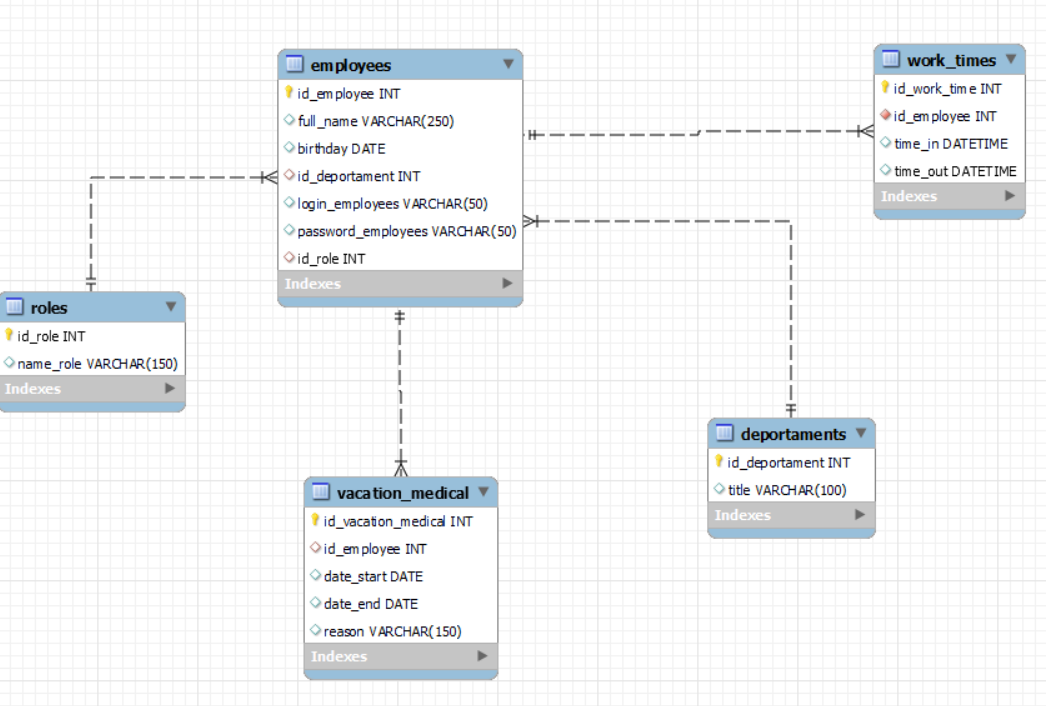


Рисунок 1.5.1 – Схема данных БД

# 1.6. Описание структуры базы данных

Описание структуры базы данных представлено в таблице 1.6.1.

Таблица 1.6.1 - Описание структуры базы данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Размер поля | Тип ключа[[1]](#footnote-2) |
| 1 | 2 | 3 | 4 | 5 |
| Employees(Работник) | | | | |
| id\_employee | IDработника | INT |  | PK |
| full\_name | ФИО рабочего | VARCHAR | 250 |  |
| birthday | Дата рождения | DATE |  |  |
| id\_deportament | ID должности | INTEGER |  | FK |
| login\_employees | Логин работника | VARCHAR | 50 |  |
| password\_employees | Пароль работника | VARCHAR | 50 |  |
| Id\_role | ID роли | INTEGER |  | FK |
| deportaments (Должности) | | | | |
| id\_deportament | ID должности | INT |  | PK |
| Title | Название должности | VARCHAR | 100 |  |
| roles (Роли) | | | | |
| id\_role | ID роли | INTEGER |  | PK |
| name\_role | Название роли | VARCHAR | 150 |  |

Продолжение таблицы 1.6.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| vacation\_medical(Отпускные больнычные) | | | | |
| id\_vacation\_medical | IDОтпускных | INT |  | PK |
| id\_employee | ID Работника | INT |  | FK |
| date\_start | Начало отпуска | DATE |  |  |
| date\_end | Конец отпуска | DATE |  |  |
| birthday | Дата рождения | DATE |  |  |
| reason | Статус (отпуск или больничный) | VARCHAR | 150 |  |
| work\_times(Рабочее время) | | | | |
| id\_work\_time | ID времени | INT |  | PK |
| Id\_emoployee | ID работника | INT |  | FK |
| time\_in | Время входа | DATATIME |  |  |
| time\_out | Время выхода | DATATIME |  |  |

1.7. Контрольный пример

Контрольный пример является ручным подсчётом задачи. По составленной программе обрабатываются исходные данные контрольного примера. Полученные результаты сравниваются с известными результатами контрольного примера. При несовпадении результатов производится поиск, исправление ошибок, и снова производится выполнение программы.

1.8 Общие требования к программному продукту

Пользователи должны иметь базовые навыки пользования персональным компьютером.

Минимальные требования к техническому обеспечению программного продукта следующие:

* ОС:  Windows 7 Service Pack1/ Windows 7 64Bit Service Pack1/ Windows 8.1 64Bit / Windows 10 64Bit / Mac OS;
* процессор: 2.40 Ггц (четырехъядерный) / AMDPhenom 9850 (четырехъядерный) @ 2.5 Ггц;
* оперативная память: 512МБ (Win 7/Win8 и выше);
* видеокарта: NVIDIA 9800 GT c 512 МБ видеопамяти/ AMD HD 4870 с 1 Гб видеопамяти (DX 9, 10, 10.1);
* жесткий диск: 10 гигабайт свободного места;
* Microsoft DirectX версия 9.0c;

Функциональные возможности программного продукта:

* приложение должно формировать и отображать выходные данные пользователю;
* в приложении должен быть обеспечен просмотр таблиц (справочников) базы данных с возможность добавления, редактирования, удаления данных.

Требования к надежности:

* приложение должно обрабатывать ошибочные действия пользователя и сообщать ему об этом;
* приложение должно обеспечивать контроль входной и выходной информации.

Требования к информационной и программной совместимости: обеспечить работу приложения с таблицами СУБД MySQL Workbench.

# 2. Экспериментальный раздел

# 2.1 Описание программы

Программа имеет модульную структуру. При ее запуске выполняется проект на WPF WorkingHours.exe. Программа WorkingHours.exe написана на языке C# в среде программирования VisualStudio 2022 с использованием системы управления базой данных MySQL Workbench.

Таблица 2.1.1. - Описание модулей

|  |  |
| --- | --- |
| Методы | Назначение |
| 1 | 2 |
| MainWindow.xaml.cs | |
| publicMainWindow() | Подключение контекста данных |
| EmployeesView.xaml.cs |  |
| public EmployeesView() | Подключение контекста данных и вывод ФИО работника |
| EmployeesEdit.xaml.cs | |
| public EmployeesEdit() | Заполнение ComboBox данными из бд |
| private void cancel\_Click(object sender, RoutedEventArgs e) | Закрытие окна |
| private void save\_Click(object sender, RoutedEventArgs e) | Изменение работника |
| Employees.xaml.cs | |
| Public Employees() | Подключение контекста данных |
| EmployeeDelete.xaml.cs | |
| public EmployeeDelete() | Заполнение ComboBox данными из бд |
| private void Delete\_Click(object sender, RoutedEventArgs e) | Удаление пользователя |
| private void Cansel\_Click(object sender, RoutedEventArgs e) | Закрытие окна |
| EmployeeAdd.xaml.cs | |
| public EmployeeAdd() | Заполнение ComboBox данными из бд |
| private void save\_Click(object sender, RoutedEventArgs e) | Добавление работника |
| private void Cansel\_Click(object sender, RoutedEventArgs e) | Закрытие окна |
| DeportamentsEdit.xaml.cs | |
| public DeportamentsEdit() | Заполнение ComboBox данными из бд |
| private void Save\_Click(object sender, RoutedEventArgs e) | Изменение должности |
| private void Cansel\_Click(object sender, RoutedEventArgs e) | Закрытие окна |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| DeportamentsAdd.xaml.cs | |
| private void Save\_Click(object sender, RoutedEventArgs e) | Добавление должности |
| private void Cansel\_Click(object sender, RoutedEventArgs e) | Закрытие окна |
| Deportaments.xaml.cs | |
| Public Deportaments() | Подключение контекста данных |
| DeleteDeportament.xaml.cs | |
| public DeleteDeportament() | Заполнение ComboBox данными из бд |
| private void Delete\_Click(object sender, RoutedEventArgs e) | Удаление должности |
| private void Cansel\_Click(object sender, RoutedEventArgs e) | Закрытие окна |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| WorkTimesBrowers.cs | |
| private void ExitWindow() | Выход из приложения |
| private void InWorkTime() | Запись времени входа работника |
| private void OutWorkTime () | Запись времени выхода работника |
| public bool CheckType(int idEmployee) | Проверка вошел ли работник |
| protected void OnPropertyChanged(String propertyName) | Реализация интерфейса INotifyPropertyChanged |
| public event PropertyChangedEventHandler PropertyChanged; | Уведомление об изменении свойст программы |
| private List<WorkTime> work = EmployeesService.GetWorkTime(); | Получение данных метода GetWorkTime() |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| public List<WorkTime> Work | Обновление списка времени входа рабочих |
| public static WorkTime? SelectedWork { get; set; } | Получение выброного значения в таблице рабочего времени |
| private void OpenEmployee() | Открытие окна Employee |
| public void OpenDeportaments() | Открытие окна OpenDeportaments |
| internal class DeportamentBrowers : BindableBase, INotifyPropertyChanged | |
| public event PropertyChangedEventHandler PropertyChanged; | Уведомление об изменении свойст программы |
| private List<Deportament> deportaments = EmployeesService.GetDeportament(); | Получение данных метода GetDeportament() |
| public List<Deportament> DeportamentsView | Обновление списка должностей |
| private List<Employee> employee = EmployeesService.GetEmployee(); | Получение данных метода GetEmployee() |
| public List<Employee> Employee | Обновление списка работников |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| public void OpenEditDeportaments() | Открытие окна DeportamentsEdit |
| public void OpenAddDeportaments() | Открытие окна DeportamentsAdd |
| public void OpenDeleteDeportaments() | Открытие окна DeleteDeportament |
| public string[] Deportaments() | Создание списка названия должностей |
| internal class BrowersEmployees : INotifyPropertyChanged | |
| private List<Employee> employee = EmployeesService.GetEmployee(); | Получение данных метода GetEmployee() |
| public List<Employee> Employee | Обновление списка работников |
| Public static Employee? SelectedEmployee { get; set; } | Получение выброного сотрудника в таблице |
| public string Username | Получение введенного логина |
| public string Password | Получение введенного пароля |
| private void Login() | Авторизация работника |
| public void OpenAddEmployees() | Открытие окна Employees |
| public void OpenEditEmployees() | Открытие окна EmployeesEdit |
| public void OpenDeleteEmployees() | Открытие окна EmployeesDelete |
| public string[] Role() | Создание списка названия ролей |
| public string[] Employee() | Создание списка ФИО работников |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| internal class EmployeesService | |
| public static List<Deportament> GetDeportament() | Получение списка должностей из базы данных |
| public static List<Role> GetRole()) | Получение списка роей из базы данных |
| public static List<Employee> GetEmployee() | Получение списка работников из базы данных |
| public static List<WorkTime> GetWorkTime() | Получение списка рабочего времени из базы данных |
| AdminW.xaml.cs | |
| public AdminW() | Подключение контекста данных, отображение ФИО менеджера |
| private void delete\_Click(object sender, RoutedEventArgs e) | Удаление рабочего времени |
| private void Filter\_Click(object sender, RoutedEventArgs e) | Фильтрация рабочего времени |
| private void Revoke\_Click(object sender, RoutedEventArgs e) | Отмена фильтрации рабочего времени |

# 2.2 Протокол тестирования программного продукта

В протоколе тестирования отражаются:

− тестирование на корректных данных;

− тестирование на некорректных данных;

− тестирование продукта на данных контрольного примера.

Тестирование авторизации на основе данных менеджера, ожидаемое сообщение «Здраствуйте, Петрова Ольга Федоровна» (рисунок 2.2.1).

|  |
| --- |
|  |

Рисунок 2.2.1 – Сообщение об успешной авторизации менеджера

Тестирование авторизации на основе данных работника, ожидаемое сообщение «Здраствуйте, Иванов Иван Иванович» (рисунок 2.2.2).

|  |
| --- |
|  |

Рисунок 2.2.2 – Сообщение об успешной авторизации Сотрудника

Тестирование авторизации на основе неккорктных данных, ожидаемое сообщение «Упс, не удалось войти. Проверьте логин и пароль!» (рисунок 2.2.3).

|  |
| --- |
|  |

Рисунок 2.2.3 – Просьба проверить логин или пароль

Тестирование записи времени входа на работу ожидаемое сообщение «Время входа записанно» (рисунок 2.2.4).

|  |
| --- |
|  |

Рисунок 2.2.4 – Сообщение об успешной записи времени входа

|  |
| --- |
|  |

Рисунок 2.2.5 – Подтверждение удаление записи

Тестирование удаления рабочего времени работника ожидаемое сообщение «Время удалено» (рисунок 2.2.6)

|  |
| --- |
|  |

Рисунок 2.2.6 – Уведомление об успешном удалении

Тестирование удаления рабочего времени работника, если работник не выбран ожидаемое сообщение «Выберите сотрудника» (рисунок 2.2.7)

|  |
| --- |
|  |

Рисунок 2.2.7 – Уведомление выберите сотрудника

Тестирование добавления новой должности, ожидаемое сообщение «Должность успешно добавлена» (рисунок 2.2.8)

|  |
| --- |
|  |

Рисунок 2.2.8 – Успешное добавление должности

Тестирование добавления новой должности, при пустом поле ввода, ожидаемое сообщение «Заполните поле» (рисунок 2.2.9)

|  |
| --- |
|  |

Рисунок 2.2.9 – Не успешное добавление должности

Тестирование изменения должности, ожидаемое сообщение «Должность успешно изменена» (рисунок 2.2.10)

|  |
| --- |
|  |

Рисунок 2.2.10 – Успешное изменени должности

Тестирование удаления должности, ожидаемое сообщение «Должность удалена» (рисунок 2.2.11)

|  |
| --- |
|  |

2.2.11 – Успешное удаление должности

Тестирование удаления должности, ожидаемое сообщение «Должность не может быть удалена Существует сотрудник работающей на этой должности» (рисунок 2.2.12)

|  |
| --- |
|  |

2.2.12 – Не успешное удаление должности

Тестирование добавления нового сотрудника, ожидаемое сообщение «Сотрудник успешно добавлен» (рисунок 2.2.13)

|  |
| --- |
|  |

2.2.13 – Успешное добавление сотрудника

Тестирование изменения сотрудника, ожидаемое сообщение «Сотрудник успешно изменена» (рисунок 2.2.14)

|  |
| --- |
|  |

Рисунок 2.2.14 – Успешное изменение сотрудника

Тестирование удаления должности, ожидаемое сообщение «Должность удалена» (рисунок 2.2.15)

|  |
| --- |
|  |

2.2.15 – Успешное удаление сотрудника

Тестирование удаления должности, ожидаемое сообщение «Должность не может быть удалена Существует сотрудник работающей на этой должности» (рисунок 2.2.16)

|  |
| --- |
|  |

2.2.16 – Не успешное удаление должности

# 2.3 Руководство пользователя

Назначение системы

Программа «Учет рабочего времени» предназначена для упрощения просмотра времени прихода и ухода с работы рабочих и удобного поиска по дате.

Основной целью данной информационной системы является формирование каталога времени отработанных за определленый период.

Условия применения системы

Программное обеспечение разрабатывается для персональной вычислительной техники со следующими характеристиками:

* Microsoft Windows 7 / 8 / 10;
* процессор 1 ГГц;
* 128 МБ ОЗУ;
* 60 МБ свободного пространства на диске;
* разрешение экрана монитора не менее 1920 × 1080;

Программа «Учет рабочего времени» предназначена для пользователей, имеющих как минимум первоначальные навыки работы с графической операционной системой, на которой будет запускаться данная программа.

Подготовка системы к работ

Для запуска программы необходимо запустить приложение workingHoursEmployee.exe из каталога, в котором установлен данный программный продукт. После этого открывается окно авторизации для входа в программу (рисунок 2.3.1).

|  |
| --- |
|  |

Рисунок 2.3.1 – Окно авторизации для входа в программу

Описание операций

После ввода правильного логина и пароля осуществляется вход в приложение. Если в систему вошел не менеджер, то не будут доступны вкладки, доступные лишь менеджеру (рисунки 2.3.2-2.3.3).

|  |
| --- |
|  |

Рисунок 2.3.2 – Интерфейс приложения, в случае входа пользователя

|  |
| --- |
|  |

Рисунок 2.3.3 – Интерфейс приложения, в случае входа администратора

При входе в приложение как работник отображается две кнопки

«вход» и «выход» при нажатии на кнопку входа запишится текущее время и окно закроется при нажатии на кнопку выйти произойдет тоже самое.

Для фильтрации времени рабочих нужно ввести в первое поле ввода начальную дату, а во второе конечную дату поиска, затем нажать на кнопку «Приминить», чтобы вернуть все по умолчанию нужно нажать на кнопку с крестиком. (рисунок 2.3.3).

Для того что бы открыть окно с должностями нужно нажать на кнопку справочники и выбрать вкладку должности, затем вам откроется окно со списком должностей. (рисунок 2.3.4)

|  |
| --- |
|  |

Рисунок 2.3.4 – Окно с должностями

Для того что бы добавить должность нужно нажать на кнопку добавить, затем заполнить поле ввода. (рисунок 2.3.5)

|  |
| --- |
|  |

Рисунок 2.3.5 – Окно добавления должности

Для того что бы изменить должность нужно нажать на кнопку изменить, выбрать изменяймую должность, затем заполнить поле ввода. (рисунок 2.3.6)

|  |
| --- |
|  |

Рисунок 2.3.6 – Окно изменение должности

Для того что бы удалить должность нужно нажать на кнопку удалить, выбрать удаляймую должность. (рисунок 2.3.7)

|  |
| --- |
|  |

Рисунок 2.3.7 – Окно удаление должности

Для того что бы открыть окно с сотрудниками нужно нажать на кнопку справочники и выбрать вкладку сотрудники, затем вам откроется окно со списком сотрудников. (рисунок 2.3.8)

|  |
| --- |
|  |

Рисунок 2.3.8 – Окно с сотрудниками

Для того что бы добавить сотрудника нужно нажать на кнопку добавить, затем заполнить все поля ввода. (рисунок 2.3.9)

|  |
| --- |
|  |

Рисунок 2.3.9 – Окно добавления сотрудника

Для того что бы изменить сотрудника нужно нажать на кнопку изменить, выбрать изменяймого сотрудника, затем заполнить все поля ввода. (рисунок 2.3.10)

|  |
| --- |
|  |

Рисунок 2.3.9 – Окно изменение сотрудника

Для того что бы удалить сотрудника нужно нажать на кнопку удалить, выбрать удаляймого сотрудника. (рисунок 2.3.10)

|  |
| --- |
|  |

Рисунок 2.3.10 – Окно удаление сотрудника

Для того что бы удалить время входа и выхода сотрудника, нужно нажать на него в таблице, а затем на кнопку «Удалить входы». (рисунок 2.3.11-2.3.12)

|  |
| --- |
|  |

Рисунок 2.3.11 – Выбор входа сотрудника для удаления

|  |
| --- |
|  |

Рисунок 2.3.12 – Время сотрудника удалено

Код программы

public partial class EmployeesEdit

BrowersEmployees employees = new BrowersEmployees();

DeportamentBrowers deportament = new DeportamentBrowers();

public EmployeesEdit()

{

InitializeComponent();

cbDepartment.ItemsSource = deportament.Deportaments();

cbRole.ItemsSource = employees.Role();

cbEmployee.ItemsSource = employees.Employee();

}

private void cancel\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

private void save\_Click(object sender, RoutedEventArgs e)

{

string currentDeportament = cbDepartment.SelectedItem?.ToString() ?? "";

string currentRole = cbRole.SelectedItem?.ToString() ?? "";

string currentEmployee = cbEmployee.SelectedItem?.ToString() ?? "";

using (DbWorkingHoursContext dbContext = new DbWorkingHoursContext())

{

int idDeportament = 0;

if (!string.IsNullOrEmpty(currentDeportament))

{

var deportament = dbContext.Deportaments.FirstOrDefault(d => d.Title == currentDeportament);

if (deportament != null)

{

idDeportament = deportament.IdDeportament;

}

}

int idEmployee = 0;

if (!string.IsNullOrEmpty(currentEmployee))

{

var employee = dbContext.Employees.FirstOrDefault(d => d.FullName == currentEmployee);

if (employee != null)

{

idEmployee = employee.IdEmployee;

}

}

int idRole = 0;

if (!string.IsNullOrEmpty(currentRole))

{

var role = dbContext.Roles.FirstOrDefault(r => r.NameRole == currentRole);

if (role != null)

{

idRole = role.IdRole;

}

}

var employeeUpdate = dbContext.Employees.FirstOrDefault(e => e.IdEmployee == idEmployee);

if (employeeUpdate != null)

{

if (txtFIO.Text == null && dtBirthday.Text == null && idDeportament == 0 && txtLogin.Text == null && txtPassword.Text == null && idRole == 0)

{

new MessegeBoxWindow("Заполните все поля", MessageButtons.Ok).ShowDialog();

}

else

{

employeeUpdate.FullName = txtFIO.Text;

employeeUpdate.Birthday = DateOnly.Parse(dtBirthday.Text);

employeeUpdate.IdDeportament = idDeportament;

employeeUpdate.LoginEmployees = txtLogin.Text;

employeeUpdate.PasswordEmployees = txtPassword.Text;

employeeUpdate.IdRole = idRole;

new MessegeBoxWindow("Сотрудник успешно изменен", MessageButtons.Ok).ShowDialog();

dbContext.SaveChanges();

}

}

else

{

new MessegeBoxWindow("Работник не найден", MessageButtons.Ok).ShowDialog();

}

}

}

public partial class MessegeBoxWindow

public MessegeBoxWindow(string Message, MessageButtons Buttons)

{

InitializeComponent();

txtMessage.Text = Message;

switch (Buttons)

{

case MessageButtons.Ok:

btnOk.Visibility = Visibility.Visible;

break;

}

}

public enum MessageButtons

{

Ok

}

private void btnOk\_Click(object sender, RoutedEventArgs e)

{

this.DialogResult = true;

this.Close();

}

public partial class EmployeeDelete : Window

{

BrowersEmployees employees = new BrowersEmployees();

public EmployeeDelete()

{

InitializeComponent();

ComboEmployee.ItemsSource = employees.Employee();

}

private void Delete\_Click(object sender, RoutedEventArgs e)

{

string currentEmployee = ComboEmployee.SelectedItem?.ToString() ?? "";

if (!string.IsNullOrEmpty(currentEmployee))

{

using (DbWorkingHoursContext dbContext = new DbWorkingHoursContext())

{

int idEmployee = 0;

if (!string.IsNullOrEmpty(currentEmployee))

{

var employee = dbContext.Employees.FirstOrDefault(d => d.FullName == currentEmployee);

if (employee != null)

{

idEmployee = employee.IdEmployee;

}

}

var employeeDelete = dbContext.Employees.FirstOrDefault(e => e.IdEmployee == idEmployee);

if (MessageBox.Show("Удалить текущую запись?", "Осторожно", MessageBoxButton.YesNo, MessageBoxImage.Warning) == MessageBoxResult.Yes)

{

try

{

dbContext.Employees.Remove(employeeDelete);

new MessegeBoxWindow("Сотрудник удален", MessageButtons.Ok).ShowDialog();

dbContext.SaveChanges();

}

catch (Exception ex)

{

new MessegeBoxWindow("Сначала удалите все входы сотрудника", MessageButtons.Ok).ShowDialog();

}

}

}

}

else

{

new MessegeBoxWindow("Выберите сотрудника", MessageButtons.Ok).ShowDialog();

}

}

private void Cansel\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

}

public partial class EmployeeAdd : Window

{

BrowersEmployees employees = new BrowersEmployees();

DeportamentBrowers deportament = new DeportamentBrowers();

public EmployeeAdd()

{

InitializeComponent();

cbDepartment.ItemsSource = deportament.Deportaments();

cbRole.ItemsSource = employees.Role();

}

private void cancel\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

private void save\_Click(object sender, RoutedEventArgs e)

{

try

{

string currentDeportament = cbDepartment.SelectedItem.ToString();

string currentRole = cbRole.SelectedItem.ToString();

using (DbWorkingHoursContext dbContext = new DbWorkingHoursContext())

{

int idDeportament = 0;

if (!string.IsNullOrEmpty(currentDeportament))

{

var deportament = dbContext.Deportaments.FirstOrDefault(d => d.Title == currentDeportament);

if (deportament != null)

{

idDeportament = deportament.IdDeportament;

}

}

int idRole = 0;

if (!string.IsNullOrEmpty(currentRole))

{

var role = dbContext.Roles.FirstOrDefault(r => r.NameRole == currentRole);

if (role != null)

{

idRole = role.IdRole;

}

}

if (!string.IsNullOrEmpty(txtFIO.Text) && !string.IsNullOrEmpty(dtBirthday.Text) && idDeportament == 0 && !string.IsNullOrEmpty(txtLogin.Text) && !string.IsNullOrEmpty(txtPassword.Text) && idRole == 0)

{

Employee employee = new Employee

{

FullName = txtFIO.Text,

Birthday = DateOnly.Parse(dtBirthday.Text),

IdDeportament = idDeportament,

LoginEmployees = txtLogin.Text,

PasswordEmployees = txtPassword.Text,

IdRole = idRole

};

dbContext.Employees.Add(employee);

new MessegeBoxWindow("Сотрудник успешно добавлен", MessageButtons.Ok).ShowDialog();

dbContext.SaveChanges();

}

else

{

new MessegeBoxWindow("Заполните все поля", MessageButtons.Ok).ShowDialog();

}

}

}

catch

{

new MessegeBoxWindow("Заполните все поля", MessageButtons.Ok).ShowDialog();

}

}

}

public partial class DeportamentsEdit : Window

{

DeportamentBrowers browers = new DeportamentBrowers();

public DeportamentsEdit()

{

InitializeComponent();

ComboDeportaments.ItemsSource = browers.Deportaments();

}

private void Cansel\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

private void Save\_Click(object sender, RoutedEventArgs e)

{

string currentTitle = ComboDeportaments.SelectedItem.ToString();

if (!string.IsNullOrEmpty(txtTitle.Text) && !string.IsNullOrEmpty(currentTitle))

{

using (DbWorkingHoursContext dbContext = new DbWorkingHoursContext())

{

var deportamentUpdate = dbContext.Deportaments.FirstOrDefault(p => p.Title == currentTitle);

if (deportamentUpdate != null)

{

deportamentUpdate.Title = txtTitle.Text;

new MessegeBoxWindow("Должность успешно изменена", MessageButtons.Ok).ShowDialog();

dbContext.SaveChanges();

}

}

}

else

{

new MessegeBoxWindow("Заполните все поля", MessageButtons.Ok).ShowDialog();

}

}

public partial class DeportamentsAdd : Window

{

public DeportamentsAdd()

{

InitializeComponent();

}

private void Cansel\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

private void Save\_Click(object sender, RoutedEventArgs e)

{

string title = txtTitle.Text;

if (!string.IsNullOrEmpty(title))

{

using (DbWorkingHoursContext dbContext = new DbWorkingHoursContext())

{

Deportament deportament = new Deportament

{

Title = title

};

dbContext.Deportaments.Add(deportament);

new MessegeBoxWindow("Должность успешно добавлена", MessageButtons.Ok).ShowDialog();

dbContext.SaveChanges();

}

}

else

{

new MessegeBoxWindow("Заполните поле", MessageButtons.Ok).ShowDialog();

}

}

}

public partial class DeleteDeportament : Window

{

DeportamentBrowers browers = new DeportamentBrowers();

public DeleteDeportament()

{

InitializeComponent();

ComboDeportaments.ItemsSource = browers.Deportaments();

}

private void Delete\_Click(object sender, RoutedEventArgs e)

{

string currentTitle = ComboDeportaments.SelectedItem.ToString();

if (!string.IsNullOrEmpty(currentTitle))

{

using (DbWorkingHoursContext dbContext = new DbWorkingHoursContext())

{

var deportamentDelete = dbContext.Deportaments.FirstOrDefault(e => e.Title == currentTitle.ToString());

if (deportamentDelete != null)

{

new MessegeBoxWindow("Действительно хотите удалить должность?", MessageButtons.Ok).ShowDialog();

try

{

dbContext.Deportaments.Remove(deportamentDelete);

new MessegeBoxWindow("Должность удалена", MessageButtons.Ok).ShowDialog();

dbContext.SaveChanges();

}

catch (Exception ex)

{

new MessegeBoxWindow("Должность не может быть удалена \nСуществует сотрудник работающий на этой должности", MessageButtons.Ok).ShowDialog();

}

}

else

{

new MessegeBoxWindow("Должность не существует", MessageButtons.Ok).ShowDialog();

}

}

}

else

{

new MessegeBoxWindow("Выберите должность", MessageButtons.Ok).ShowDialog();

}

}

private void Cansel\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

}

public partial class AdminW : Window

{

WorkTimesBrowers work = new WorkTimesBrowers();

public AdminW()

{

InitializeComponent();

NameTitle.Text = "Руководитель - " + UserAuth.CurrentUserName;

DataContext = new WorkTimesBrowers();

}

private void delete\_Click(object sender, RoutedEventArgs e)

{

var currentWork = EmployeeWork.SelectedItem as WorkTime;

using(DbWorkingHoursContext dbContext = new DbWorkingHoursContext())

{

if (MessageBox.Show("Удалить текущую запись?", "Осторожно", MessageBoxButton.YesNo, MessageBoxImage.Warning) == MessageBoxResult.Yes)

{

try

{

dbContext.WorkTimes.Remove(currentWork);

new MessegeBoxWindow("Время удалено", MessageButtons.Ok).ShowDialog();

dbContext.SaveChanges();

EmployeeWork.ItemsSource = dbContext.WorkTimes.ToList();

}

catch (Exception ex)

{

new MessegeBoxWindow("Выберите сотрудника", MessageButtons.Ok).ShowDialog();

}

}

}

}

private void Filter\_Click(object sender, RoutedEventArgs e)

{

var currentWorks = EmployeesService.GetWorkTime();

if (dtStart == null)

{

new MessegeBoxWindow("Укажите начальную дату фильтра", MessageButtons.Ok).ShowDialog();

}

if (dtEnd == null)

{

new MessegeBoxWindow("Укажите конечную дату фильтра", MessageButtons.Ok).ShowDialog();

}

if (dtStart != null && dtEnd != null)

{

var filteredWorks = currentWorks.Where(w => w.TimeIn >= dtStart.SelectedDate && w.TimeOut <= dtEnd.SelectedDate);

EmployeeWork.ItemsSource = filteredWorks.ToList(); // Присваиваем отфильтрованный список свойству Work

}

}

private void Revoke\_Click(object sender, RoutedEventArgs e)

{

EmployeeWork.ItemsSource = work.Work;

}

}

internal class WorkTimesBrowers

{

private RelayCommand timeIn;

public RelayCommand TimeIn

{

get

{

return timeIn ?? new RelayCommand(obj =>

{

InWorkTime();

}

);

}

}

private RelayCommand timeOut;

public RelayCommand TimeOut

{

get

{

return timeOut ?? new RelayCommand(obj =>

{

OutWorkTime();

}

);

}

}

private RelayCommand openDeportament;

public RelayCommand OpenDeportament

{

get

{

return openDeportament ?? new RelayCommand(obj =>

{

OpenDeportaments();

});

}

}

public void OpenDeportaments()

{

Deportaments deportament = new Deportaments();

deportament.ShowDialog();

}

private RelayCommand employeeOpen;

public RelayCommand EmployeeOpen

{

get

{

return employeeOpen ?? new RelayCommand(obj =>

{

OpenEmployee();

}

);

}

}

private void OpenEmployee()

{

Employees employees = new Employees();

employees.ShowDialog();

}

private RelayCommand exit;

public RelayCommand Exit

{

get

{

return exit ?? new RelayCommand(obj =>

{

ExitWindow();

}

);

}

}

private void ExitWindow()

{

System.Windows.Application.Current.Shutdown();

}

private void InWorkTime()

{

using (DbWorkingHoursContext dbContext = new DbWorkingHoursContext())

{

CheckType(UserAuth.CurrentUserId);

var workTime = new WorkTime

{

IdEmployee = UserAuth.CurrentUserId,

TimeIn = DateTime.Now,

};

dbContext.WorkTimes.Add(workTime);

dbContext.SaveChanges();

new MessegeBoxWindow($"Время входа записанно", MessageButtons.Ok).ShowDialog();

}

}

private void OutWorkTime()

{

using (DbWorkingHoursContext dbContext = new DbWorkingHoursContext())

{

var workTimeEntry = dbContext.WorkTimes.FirstOrDefault(w => w.IdEmployee == UserAuth.CurrentUserId && w.TimeOut == null);

if (workTimeEntry != null)

{

workTimeEntry.TimeOut = DateTime.Now;

dbContext.SaveChanges();

new MessegeBoxWindow($"Время выхода записанно", MessageButtons.Ok).ShowDialog();

}

}

}

public bool CheckType(int idEmployee)

{

var workTimes = EmployeesService.GetWorkTime();

var workTime = workTimes

.Where(w => w.IdEmployee == idEmployee && w.TimeOut == null)

.Select(w => new { HasWorkTime = w != null })

.FirstOrDefault();

return workTime?.HasWorkTime ?? false;

}

public event PropertyChangedEventHandler PropertyChanged;

private List<WorkTime> work = EmployeesService.GetWorkTime();

public List<WorkTime> Work

{

get { return work; }

set

{

work = value;

OnPropertyChanged("Work");

}

}

public static WorkTime? SelectedWork { get; set; }

protected void OnPropertyChanged(String propertyName)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

}

}

internal class DeportamentBrowers : BindableBase, INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

private List<Deportament> deportaments = EmployeesService.GetDeportament();

public List<Deportament> DeportamentsView

{

get { return deportaments; }

set

{

deportaments = value;

OnPropertyChanged("DeportamentsView");

}

}

private void NotifyPropertyChanged(String propertyName)

{

if (PropertyChanged != null)

{

PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

private List<Employee> employee = EmployeesService.GetEmployee();

public List<Employee> Employee

{

get { return employee; }

set

{

employee = value;

NotifyPropertyChanged("Employee");

}

}

protected void OnPropertyChanged(String propertyName)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

private RelayCommand openEditDeportament;

public RelayCommand OpenEditDeportament

{

get

{

return openEditDeportament ?? new RelayCommand(obj =>

{

OpenEditDeportaments();

});

}

}

public void OpenEditDeportaments()

{

DeportamentsEdit editDeportament = new DeportamentsEdit();

editDeportament.ShowDialog();

}

private RelayCommand openAddDeportament;

public RelayCommand OpenAddDeportament

{

get

{

return openAddDeportament ?? new RelayCommand(obj =>

{

OpenAddDeportaments();

});

}

}

public void OpenAddDeportaments()

{

DeportamentsAdd addDeportament = new DeportamentsAdd();

addDeportament.ShowDialog();

}

private RelayCommand openDeleteDeportament;

public RelayCommand OpenDeleteDeportament

{

get

{

return openDeleteDeportament ?? new RelayCommand(obj =>

{

OpenDeleteDeportaments();

});

}

}

public void OpenDeleteDeportaments()

{

DeleteDeportament deleteDeportament = new DeleteDeportament();

deleteDeportament.ShowDialog();

}

public string[] Deportaments()

{

var deportament = EmployeesService.GetDeportament();

List<string> titleDeportament = new List<string>();

foreach (var d in deportament)

{

titleDeportament.Add(d.Title);

}

return titleDeportament.ToArray();

}

}

}

internal class BrowersEmployees : INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

private List<Employee> employees = EmployeesService.GetEmployee();

public List<Employee> Employees

{

get { return employees; }

set

{

employees = value;

OnPropertyChanged("Employees");

}

}

public static Employee? SelectedEmployee { get; set; }

private string \_username;

private string \_password;

public static string FullUser;

public string Username

{

get => \_username;

set

{

\_username = value;

OnPropertyChanged(nameof(Username));

}

}

public string Password

{

get => \_password;

set

{

\_password = value;

OnPropertyChanged(nameof(Password));

}

}

private RelayCommand auth;

public RelayCommand Auth

{

get

{

return auth ?? new RelayCommand(obj =>

{

Login();

}

);

}

}

private void Login()

{

var users = EmployeesService.GetEmployee();

try

{

if (string.IsNullOrWhiteSpace(Username))

{

new MessegeBoxWindow("Укажите логин!", MessageButtons.Ok).ShowDialog();

return;

}

if (string.IsNullOrWhiteSpace(Password))

{

MessageBox.Show("Укажите пароль!");

return;

}

var user = users.FirstOrDefault(u => u.LoginEmployees == Username && u.PasswordEmployees == Password);

if (user != null)

{

UserAuth.CurrentUserId = user.IdEmployee;

UserAuth.CurrentUserRole = user.IdRole.ToString();

UserAuth.CurrentUserName = user.FullName;

new MessegeBoxWindow($"Здраствуйте, {UserAuth.CurrentUserName}", MessageButtons.Ok).ShowDialog();

if (users.Any(u => UserAuth.CurrentUserRole == "1"))

{

EmployeesView emploess = new EmployeesView();

emploess.Show();

Application.Current.MainWindow.Close();

}

else

{

AdminW adminW = new AdminW();

adminW.Show();

Application.Current.MainWindow.Close();

}

}

else

{

new MessegeBoxWindow("Упс, не удалось войти.\nПроверьте логин или пароль!", MessageButtons.Ok).ShowDialog();

}

}

catch (Exception ex)

{

new MessegeBoxWindow(ex.Message, MessageButtons.Ok).ShowDialog();

}

}

private RelayCommand openAddEmployee;

public RelayCommand OpenAddEmployee

{

get

{

return openAddEmployee ?? new RelayCommand(obj =>

{

OpenAddEmployees();

});

}

}

public void OpenAddEmployees()

{

EmployeeAdd employeeAdd = new EmployeeAdd();

employeeAdd.ShowDialog();

}

private RelayCommand openEditEmployee;

public RelayCommand OpenEditEmployee

{

get

{

return openEditEmployee ?? new RelayCommand(obj =>

{

OpenEditEmployees();

});

}

}

public void OpenEditEmployees()

{

EmployeesEdit employeeEdit = new EmployeesEdit();

employeeEdit.ShowDialog();

}

private RelayCommand openDeleteEmployee;

public RelayCommand OpenDeleteEmployee

{

get

{

return openDeleteEmployee ?? new RelayCommand(obj =>

{

OpenDeleteEmployees();

});

}

}

public void OpenDeleteEmployees()

{

EmployeeDelete employeeDelete = new EmployeeDelete();

employeeDelete.ShowDialog();

}

protected virtual void OnPropertyChanged(String propertyName)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

public string[] Role()

{

var roles = EmployeesService.GetRole();

List<string> nameRole= new List<string>();

foreach (var role in roles)

{

nameRole.Add(role.NameRole);

}

return nameRole.ToArray();

}

public string[] Employee()

{

var employees = EmployeesService.GetEmployee();

List<string> nameEmployees = new List<string>();

foreach (var employee in employees)

{

nameEmployees.Add(employee.FullName);

}

return nameEmployees.ToArray();

}

}

internal class EmployeesService

{

public static List<Deportament> GetDeportament()

{

using (DbWorkingHoursContext db = new DbWorkingHoursContext())

{

var result = db.Deportaments.ToList();

return result;

}

}

public static List<Role> GetRole()

{

using (DbWorkingHoursContext db = new DbWorkingHoursContext())

{

var result = db.Roles.ToList();

return result;

}

}

public static List<Employee> GetEmployee()

{

using (DbWorkingHoursContext db = new DbWorkingHoursContext())

{

db.Deportaments.ToList();

db.Roles.ToList();

var result = db.Employees.ToList();

return result;

}

}

public static List<WorkTime> GetWorkTime()

{

using (DbWorkingHoursContext db = new DbWorkingHoursContext())

{

db.Deportaments.ToList();

db.Employees.ToList();

var result = db.WorkTimes.ToList();

return result;

}

}

# ЗАКЛЮЧЕНИЕ

В процессе выполнения курсового проекта были разработаны структура и алгоритм работы WPF-приложения «Учет рабочего времени».

При этом были изучены особенности реализации компонентов WPF для построения клиентских приложений с визуально привлекательными возможностями взаимодействия с пользователем.

Результатом работы стало создание WPF-приложения учета рабочего времени рабочих на предприятии.

WPF-приложение написано на языке C# в среде разработки Visual Studio 2022 с использованием языка разметки XAML и системы управления базой данных MySQL Workbench 8.0 CE.

Были проведены опытная эксплуатация и отладочное тестирование WPF приложения. По результатам отладочного тестирования были устранены некоторые недостатки, в частности были обнаружены и исправлены неточности в реализации алгоритма: усовершенствован контроль на входные данные и отформатирован вывод документов. После этого было написано руководство пользователя.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. [ГОСТ 28195-89](https://gost.ruscable.ru/Index/11/11212.htm) Оценка качества программных средств. Общие положения: межгосударственный стандарт: дата введения 1990-07-01 - <https://docs.cntd.ru/document/1200009135?ysclid=lroox3amxp882669222> – Текст: электронный.
2. [ГОСТ 28806-90](https://gost.ruscable.ru/Index/10/10605.htm)Качество программных средств. Термины и определения: межгосударственный стандарт: дата введения 1992-01-01 - <https://docs.cntd.ru/document/1200009077?ysclid=lrq6to4u1z728775522>– Текст: электронный.
3. [ГОСТ 34.321-96](https://gost.ruscable.ru/Index/6/6808.htm)Информационные технологии. Система стандартов по базам данных. Эталонная модель управления данными: межгосударственный стандарт: дата введения 2001-07-01 - <https://docs.cntd.ru/document/1200017662?ysclid=lrq6wn65z5753511468>–Текст: электронный.
4. [ГОСТ Р ИСО/МЭК 12207-99. Информационная технология. Процессы жизненного цикла программных средств](https://docs.cntd.ru/document/1200009075#7D20K3): межгосударственный стандарт: дата введения 2000-07-01 -https://docs.cntd.ru/document/1200009075?ysclid=lrool2t1vo124212660. – Текст: электронный.
5. Ботрос, С. MySQLпо максимуму / C. Ботрос, Д. Тинли., – 4-е изд., перераб. и доп. – Питер, 2023. – 432 с. – ISBN978-5-4461-2261-5. – Текст: непосредственный.
6. Доусон, М. Программируем на Python / М. Доусон., перераб. и доп. – Питер, 2022. – 416 с. – ISBN 978-5-4461-1386-6. – Текст: непосредственный.
7. Кириченко, А.В. Web на практике. CSS, HTML, JavaScript, MySQL, PHP для fullstack-разработчиков/ А.В. Кириченко, А.П. Никольский, Е.В. Дубовик., перераб. и доп. – Питер, 2021. – 432 с. – ISBN978-5-94-387271-6. –Текст: непосредственный.
8. Макеев, Г.А. Объектно-ориентированное программирование: с нуля к SOLIDи MVC / Г.А. Макеев., перераб. и доп. – БХВ, 2024. – 272 с. – ISBN978-5-9775-1913-7. –Текст: непосредственный.
9. Постолит, А.В. Python, Djangoи Bootstrapдля начинающих / А.В. Постолит., переаб. и доп. – БХВ, 2023. – 624 с. – ISBN 978-5-9775-1807-9. –Текст: непосредственный
10. Стефанов, С. React. Быстрый старт / С. Стефанов., – 2-е изд., перераб. и доп. – Питер, 2023. – 304 с. – ISBN 978-5-4461-2115-1. –Текст: непосредственный.
11. Тараканов, О.В. Базы данных. Учебник. Студентам ССУЗов/ О.В. Тараканов, Л.И. Шустова., 2023. – 304 с. – ISBN978-5-16-010485-0. - Текст: непосредственный.
12. Трэйси, О. Веб-дизайн для недизайнеров/ О. Трэйси., перераб. и доп. – Питер, 2022. – 176 с. – ISBN978-5-4461-1917-2. – Текст: непосредственный.
13. Документация CSS. – 2024. Текст :электронный. – URL: <https://developer.mozilla.org/ru/docs/Web/CSS/Reference>(дата обращения: 26.01.2024).
14. Документация Django. – 2021. Текст :электронный. – URL: <https://djangodoc.ru/3.2/> (дата обращения: 26.01.2024).
15. Документация HTML. – 2024. Текст :электронный. – URL: <https://developer.mozilla.org/ru/docs/Web/HTML>(дата обращения: 26.01.2024).
16. Документация JavaScript.–2024. Текст :электронный. – URL: <https://developer.mozilla.org/ru/docs/Web/JavaScript>(дата обращения: 26.01.2024).
17. Документация MySQL. – 2024. Текст :электронный. – URL: <https://dev.mysql.com/doc/>(дата обращения: 26.01.2024).
18. Документация Python. – 2024. Текст :электронный. – URL: <https://docs.python.org/3/index.html>(дата обращения: 26.01.2024).
19. Документация React. – 2023. Текст :электронный. – URL: <https://react.dev/learn>(дата обращения: 26.01.2024).
20. Документация Яндекс Карты API. Текст :электронный. – URL: <https://yandex.ru/maps-api/docs?ysclid=lrq7c8dsns226257630>(дата обращения: 26.01.2024).

# Приложение А

Пример описания структуры базы данных

Таблица Ж.1- Departs (Список отделов)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, длина | Примечания |
| Аббревиатура отдела | D\_IDATE | VARCHAR(12) | первичный ключ |
| Название отдела | D\_NAME | VARCHAR(100) | обязательное поле |

Таблица Ж.2- Rooms (Список комнат)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, длина | Примечания |
| Отдел | R\_DEPART | VARCHAR(12) | внешний ключ (к Departs) |
| Номер комнаты | R\_ROOM | NUMERIC(4) |  |
| Телефон | R\_PHONE | VARCHAR(20) |  |

Таблица Ж.3- Posts (Список должностей)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, длина | Примечания |
| Название должности | P\_POST | VARCHAR(30) | первичный ключ |
| Оклад | P\_SAL | NUMERIC (8,2) | обязательное поле, > 4500 руб. |

Таблица Ж.4- Employees (Список сотрудников)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, длина | Примечания |
| Идентификатор сотрудника | E\_IDATE | NUMERIC (4) | суррогатный первичный ключ |
| Фамилия | E\_FNAME | VARCHAR(25) | обязательное поле |
| Имя, отчество | E\_LNAME | VARCHAR(30) | обязательное поле |
| Дата рождения | E\_BORN | DATE | обязательное поле |
| Пол | E\_GENDER | CHAR(1) | обязательное поле |
| Серия и номер паспорта | E\_PASP | CHAR(10) | обязательное уникальное поле |
| Когда выдан паспорт | E\_DATE | DATE | обязательное поле |
| Кем выдан паспорт | E\_GIVEN | VARCHAR(50) | обязательное поле |
| ИНН | E\_INN | CHAR(12) | обязательное уникальное поле |
| Номер пенсионного страхового свидетельства | E\_PENS | CHAR(14) | обязательное уникальное поле |
| Отдел | E\_DEPART | VARCHAR(12) | внешний ключ (к Departs) |
| Должность | E\_POST | VARCHAR(30) | внешний ключ (к Posts) |
| Номер комнаты | E\_ROOM | NUMERIC(4) | составной внешний ключ (к Rooms) |
| Рабочий телефон | E\_PHONE | VARCHAR(20) |  |
| Логин | E\_LOGIN | VARCHAR(30) |  |

1. PK-первичный ключ

   FK-внешний ключ [↑](#footnote-ref-2)