**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

Специальность 1-40 04 01 «Информатика и технологии программирования»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по дисциплине «Объектно-ориентированное программирование»

на тему: «**РАСЧЕТ ЗАРАБОТНОЙ ПЛАТЫ РАБОТНИКОВ БЮДЖЕТНОЙ СФЕРЫ»**

Исполнитель: студентка гр. ИП-21 М. А. Процкая

Руководитель: преподаватель В. Н. Шибеко

Дата проверки: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата допуска к защите: ­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Подписи членов комиссии

по защите курсовой работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гомель 2019

Установа адукацыі «Гомельскі дзяржаўны   
тэхнічны ўніверсітэт імя П.В. Сухога»

Факультэт аўтаматызаваных і інфармацыйных сістэм

Кафедра \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РЭЦЭНЗІЯ**

**на курсавы праект (работу)**

па дысцыпліне \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ,

выканана студэнтам \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

групы \_\_\_\_\_\_\_\_\_\_

I. Пералік заўваг па тэксту курсавога праекта (работы)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

II. Агульная характарыстыка работы

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

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

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

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

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

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

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

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

III. Складаючыя агульнай адзнакі па праекту (рабоце)

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

1. Своечасовасць выканання

|  |  |  |  |
| --- | --- | --- | --- |
| + |  |  | – |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| нізкі ўзровень | | | |  |  | высокі ўзровень | | | |
|  |  |  |  |  |  |  |  |  |  |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

2. Адпаведнасць заданню

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| нізкі ўзровень | | | |  |  | высокі ўзровень | | | |
|  |  |  |  |  |  |  |  |  |  |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

3. Рацыянальнасць праектных рашэнняў

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| нізкі ўзровень | | | |  |  | высокі ўзровень | | | |
|  |  |  |  |  |  |  |  |  |  |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

4. Правільнасць выканання разлікаў

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| нізкі ўзровень | | | |  |  | высокі ўзровень | | | |
|  |  |  |  |  |  |  |  |  |  |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

5. Поўнасць выканання і якасць

афармлення тлумачальнай запіскі

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| нізкі ўзровень | | | |  |  | высокі ўзровень | | | |
|  |  |  |  |  |  |  |  |  |  |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

6. Поўнасць выканання і якасць

афармлення графічнай часткі (пры наяўнасці)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| нізкі ўзровень | | | |  |  | высокі ўзровень | | | |
|  |  |  |  |  |  |  |  |  |  |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

7. Поўнасць і абгрунтаванасць

высноў па праекту (рабоце)

Адзнака пра допуск

праекта (работы) да абароны \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата допуску, подпіс аднаго з чальцоў камісіі па праверцы)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| нізкі ўзровень | | | |  |  | высокі ўзровень | | | |
|  |  |  |  |  |  |  |  |  |  |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

8. Абарона курсавога праекта (работы)

**Агульная адзнака праекта (работы)** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(выстаўляецца з улікам усіх складнікаў па п.п. 1…8)

« \_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_.

(дата)

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

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

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

(подпісі, ініцыялы, П., І., І.п.б. чальцоў камісіі па праверцы)

**СОДЕРЖАНИЕ**

Введение………………………………………………………………....……….3

1. Аналитический обзор существующих методов и средств решения….........4

1.1 Обзор существующих методов и средств автоматизации ...............4

1.2 Требования к проектируемому программному обеспечению…......7

1.3 Используемые средства……………………………….……….……..8

2. Архитектура программного обспечения………………..……..….…………9

2.1 Анализ задания на проектирование………………………………….9

2.2 Модель предметной области………………………..………………10

3. Структура и основные алгоритмы создаваемого программного обеспечения………………………………………………………..……………12

3.1 Структурная схема приложения и ее описание……..……………..12

3.2 Описание основных форм ведения диалогов………………………13

3.2.1 Начало работы в системе……………………………………13

3.2.2 Работа с сотрудниками………………………..…………….15

3.2.3 Предоставление отчетов…………………………………….18

3.3 Руководство пользователя…………………………………………..22

Тестирование……………………………………………………………………23

Заключение…………………………….……………………...……….…....…..24

Список использованных источников………………..……………….…..........25

Приложение А – Листинг программы……….………..…….………..........…..26

**ВВЕДЕНИЕ**

Долгое время людям приходилось записывать необходимые данные на бумаге, например, в архивах, что являлось неудобным, так как поиск нужного занимал довольно долгое время.

С развитием электроники и информационных технологий появились электронные базы данных, которые хранили всю накопленную информацию.

Однако вскоре появилась необходимость в базах данных, которые имеют некоторые специфические свойства.

В настоящее время в современном мире электронных технологий практически невозможно представить компанию, в которой не требуется обработка некоторого объема информации. Информацию требуется где-то хранить. Информация может динамически изменяться. Регулярно требуется выборка данных по определенным критериям из всего массива.

При автоматизации бизнес-процессов очень часто возникают задачи, которые не решают уже готовые программы и базы данных.

В ходе выполнения данного курсового проекта было создано приложение, позволяющее автоматизировать процесс вычисления заработной платы для работников бюджетной сферы.

В ходе работы с программой был использован язык C#, так как C# широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений (игр). Так же была использована интегрированная среда программирования Microsoft Visual Studio 2019 для лучшей компиляции программы и своевременного нахождения ошибок.

**1 АНАЛИТИЧЕСКИЙ ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ И СРЕДСТВ РЕШЕНИЯ**

* 1. **Обзор существующих методов и средств автоматизации**

Заработная плата – основной вид вознаграждения за проделанную работу, которое выплачивается наемному работнику работодателем. Обычно она представляет собой фиксированный оклад за определенное количество отработанных дней или выпущенных единиц продукции. Кроме этого, к ней плюсуются денежные поощрения и доплаты, а также вычитаются налоговые отчисления в государственные фонды.

Информация о том, как выполняется расчет зарплаты, необходима не только бухгалтеру или кассиру. Любой работник должен разбираться в тонкостях ее начисления. Используя формулы, вы сможете проверить проведенные расчеты и воспользоваться определенными льготами по налогам, предлагаемыми государством.

Даже в одной компании расчет вознаграждения за труд представляет собой определенные сложности уже только из-за использования различных систем оплаты труда для разных категорий сотрудников. Для группы компаний дополнительно необходимо обеспечить прозрачность расчетов и сопоставимость больших объемов информации из каждой дочерней компании, а также разработать единую для всей группы форму организации процесса расчета. В большой степени успешное решение этой задачи зависит от имеющейся у компании информационной системы, которая способна и функционально, и технологически поддерживать все расчеты.

В настоящее время существует множество программ для расчета заработной платы, например, TimeSheet, Кадры Плюс 7.1.0, Ирбис-Зарплата, ARUS Зарплата и Кадры, 1С: Зарплата и другие.

Различные автоматизированные программы для расчёта зарплаты сейчас все активнее внедряются в нашу жизнь. Сегодня сложно представить, как на предприятии заработная плата считается бухгалтерами вручную. В подсчете им помогают современные системы, созданные для учета рабочего времени, расчитывания заработной платы, составления отчетов.

Популярной программой для расчета заработной платы является «Многобюджетный расчет заработной платы». Сложная внутри и простая снаружи — так можно охарактеризовать эту **программу начисления заработной платы**. Интерфейс и общие принципы работы программы заимствованы от ее предшественницы, работавшей в среде ДОС. Интерфейс, разработанный еще в 90-е годы, оказался настолько удачным, что не изменялся с момента создания.

Программа-предшественница стала буквально рекордсменом по продолжительности использования. Она проработала с 1993 до 2017 года. В 2017 программа была переписана для работы в среде Windows и к ее функционалу была добавлена возможность многобюджетного учета начислений и удержаний. Пользование программой интуитивно просто, этим и объясняется столь длительная жизнь ее предшественницы. Интерфейс программы построен таким образом, чтобы бухгалтер мог выполнять максимально возможное количество операций без использования мыши. Все основные функции программа позволяет выполнять путем нажатия клавиш на клавиатуре. Это позволяет уменьшить утомляемость с одновременным ускорением работы.

Еще одной из популярных программ является «**Автоматизация расчета заработной платы 1С».** Ее назначение— дать возможность бухгалтеру, который использует типовую конфигурацию «Бухгалтерия 2.0» 1С 8.2, автоматически получать суммы заработной платы сотрудников, используя различные методы расчета этих сумм (оклад за отработанное время, больничный, отпуск, оплата за ночные часы и др.).

Также популярной программой можно считать «Табель учета рабочего времени». Это удобная и функциональная программа для отдела кадров и руководителей подразделений (отделов, служб), которая позволяет вести учет времени работы сотрудников, а также печатать табели и графики дежурств.

Основные возможности программы:

1. Наличие справочника сотрудников организации;
2. Учет рабочего времени сотрудников (используются официально утвержденные условные обозначения и коды);
3. Возможность печати табелей по формам Т-13, а также по форме программы с детальным подсчетом количества отработанных часов и дней по различным кодам;
4. Учет нерабочих праздничных дней РФ;
5. Экспорт форм в форматы DOC, XLS, OpenOffice;
6. Автоматическая архивация базы данных.
   1. **Требования к проектированному программному обеспечению**

В данной курсовой работе необходимо было разработать программное обеспечение для автоматизации службы расчета заработной платы рабтников бюджетной сферы. Оно должно иметь следующий функционал:

1. ведение справочников,
2. расчет заработной платы,
3. учет рабочего времени,
4. формирование надбавок;

Данные функции работают на основе следующих справочников:

1. должности,
2. отделения,
3. работники;

Программа должна предоставлять следующие отчеты:

1. расчетный листок,
2. ведомость выдачи заработной платы,
3. ведомость перевода на карточку,
4. сводный отчет по предприятию;

**1.3 Используемые средства**

В данной курсовой работе выделяются главными частями являютя создание приложения под Windows, создание сервера для взаимодействия с базой данных, и создание базы данных.

Для создания клиентского приложения был выбран объектно-ориентированный язык программирования C#, который является языком разработки приложений для платформы Microsoft .NET Framework и основой для использования технологии Windows Forms.

Данная технология имеет достаточно удобный инструмент для разработки Windows приложений.

Для хранения данных и управления ими используется реляционная система управления базами данных MS Access 2019. Эта система максимально упрощает процесс работы с данными, имеет понятный даже неопытному пользователю интерфейс и представляет большое количество возможностей для разработки.

Разработка этого проекта велась в среде Visual Studio 2019.

1. **АРХИТЕКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

## **2.1 Анализ задания на проектирование**

Данное приложение предназначено для использования бухгалтером предприятия бюджетной сферы, в нашем случае – больницы.

При помощи этого приложения бухгалтер сможет рассчитывать заработную плату сотрудникам предприятия, выводить необходимые отчеты, а также производить кадровый учет.

Приложение имеет один уровень доступа, так как оно рассчитано на использование только бухгалтерами.

Данная программа содержит работу с сотрудниками предприятия, а это означает, что легко можно получить доступ к личным данным (серия и номер паспорта, размер заработной платы и т.д.).

Чтобы этого не произошло, в приложении отсутствует такая роль, как гость. То есть незарегистрированный в системе пользователь не сможет получить и использовать в корыстных целях конфиденциальные данные.

Данные решено хранить в базе данных. Это позволяет упорядочить, систематизировать и облегчить работу с большими объемами данных.

**2.2 Модель предметной области**

Рассмотрим модель предметной области, а также сущности и связи между ними.

База данных спроектирована на основе классов моделей, приведенных ниже.

Приведем диаграмму сущностей:

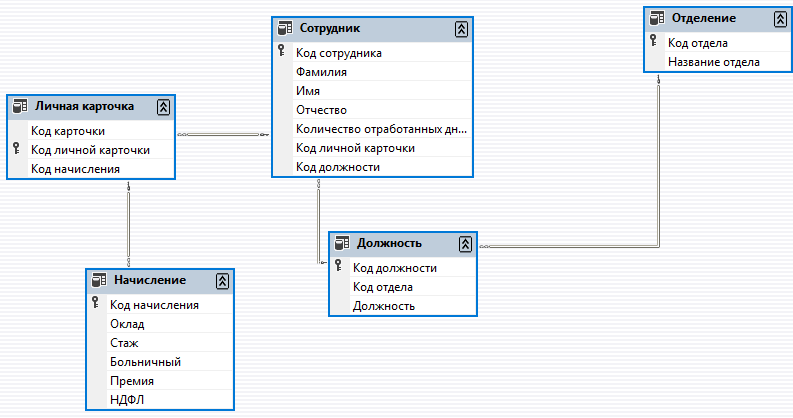


Рисунок 3.1 – Диаграмма сущностей

Данная модель имеет 5 сущностей: Сотрудник, Должность, Отделение, Личная карточка и Начисление.

1. Сотрудник – сущность, отражающая данные о сотрудниках. Имеет следующие атрибуты:
2. Код сотрудника
3. Фамилия
4. Имя
5. Отчество
6. Код отделения
7. Номер карточки (номер личной карточки)
8. Код должности
9. Должность – сущность отражает данные о должностях. Атрибуты сущности:
10. Код должности
11. Код отдела
12. Должность
13. Отделение – отражает данные об отделениях. Имеет следующие атрибуты:
14. Код отдела
15. Название отдела
16. Личная карточка – сущность, отражающая данные о личной карточке сотрудника. Атрибуты данной сущности:
17. Код карточки (код карточки для перевода заработной платы)
18. Код личной карточки
19. Код начисления
20. Начисление – отражает информацию о начислениях. Начисления делятся на вычисляемые и вводимые. Вычисляемые:
21. Премия (10% начислений за стаж)
22. НДФЛ (13% оклада)
23. Стаж (5% оклада)

Вводимые:

1. Больничный
2. Оклад

Рассмотрим виды связей между сущностями в базе данных (Рисунок 1). В проекте используются связи «один ко многим» и «многие к одному».

Сущность Сотрудник соединена связью «многие к одному» с сущностью Должность, так как один сотрудник может занимать одну должность, но также одинаковую должность могут занимать несколько сотрудников.

Сущность Должность соединена связью «один ко многим» с сущностью Отделение, так как данная должность существует только в одном отделении, однако в отделении содержится много должностей.

Сущность Личная карточка соединена связью «многие к одному» с сущностью Сотрудник, так как у одного сотрудника может быть несколько личных карточек.

Сущность Начисления соединена связью «многие к одному» с сущностью Личная карточка, так как у личной карточки могут быть одни начисления (в единственном экземпляре), но одинаковые начисления могут быть у разных карточек.

**3. СТРУКТУРА И ОСНОВНЫЕ АЛГОРИТМЫ СОЗДАВАЕМОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

**3.1 Структурная схема приложения и ее описание**

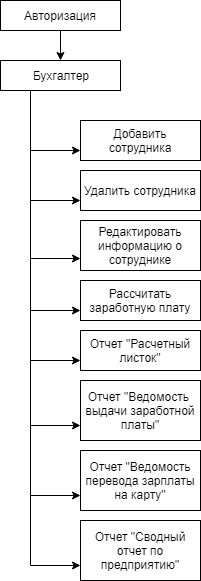


Рисунок 3.1 – Структурная схема приложения

В данной программе подразумевается одна роль – бухгалтер.

Бухгалтер ведет учет сотрудников. Рассмотрим отдельно каждую функцию:

1. Добавить сотрудника. Эта функция позволяет добавить нового сотрудника в базу данных для дальнейшего расчёта заработной платы сотрудника и кадрового учета сотрудников.
2. Удалить сотрудника. Данная функция дает возможность удалить

сотрудника из базы данных, если информация о нем по каким-то причинам больше не нужна (к примеру, увольнение).

1. Редактировать информацию о сотруднике. Функция позволяет отредактировать нужную информацию о выбранном работнике, если она была введена неверно.

Также бухгалтер рассчитывает заработную плату согласно проработанным за месяц дням и формирует отчеты. Рассмотрим отчеты:

1. Отчет «Расчетный листок». Отчет позволяет просмотреть информацию о начислениях для выбранного сотрудника, а также высчитанную заработную плату для него.
2. Отчет «Ведомость выдачи заработной платы». Данный отчет отражает размер выданной заработной платы для необходимого сотрудника.
3. Отчет «Ведомость перевода на карту». Отчет показывает размер заработной платы и номер карты, на которую она была переведена.
4. Отчет «Сводный отчет по предприятию». Данный отчет предоставляет информацию о сумме, которая была выдана сотрудникам на всем предприятии.
   1. **Описание основных форм ведения диалогов**
      1. **Начало работы в системе**

При запуске программы пользователь видит следующее окно:

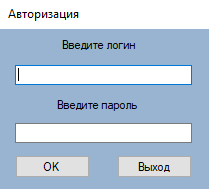


Рисунок 3.2 – Форма авторизации

Здесь пользователь может войти в систему либо выйти из программы. Пароль при вводе скрывается специальными символами.

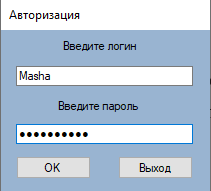


Рисунок 3.3 – Демонстрация ввода пароля

После успешного ввода данных для авторизации пользователь (бухгалтер) получает доступ к основной форме программы (Рисунок 3.4).

Основная форма включает в себя все основные функции системы:

1. начисление заработной платы,
2. отображение данных о сотрудниках,
3. демонстрация отчетов;

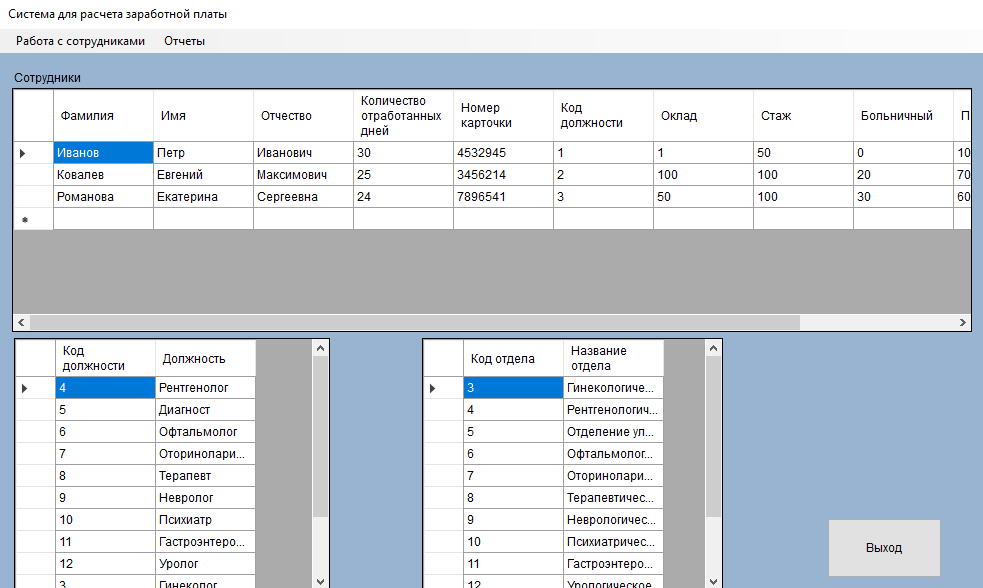
****

Рисунок 3.4 – Основная форма

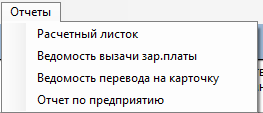
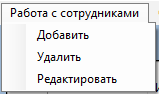
Форма отображает данные из трех основных таблиц:

1. Сотрудники,
2. Должности,
3. Отделения;

На ней можно увидеть кнопку для выхода из системы. При нажатии на нее мы снова попадаем на форму для авторизации, чтобы при необходимости снова войти в систему для дальнейшей работы.

Также вверху на форме находятся два выпадающих меню:

1. для работы с сотрудниками,
2. для отображения отчетов;



Рисунки 3.5, 3.6 – Выпадающие меню

* + 1. **Работа с сотрудниками**

Разберем выпадающее меню «Работа с сотрудниками». При нажатии на первый пункт «Добавить» нам открывается форма «Личная карточка», при помощи которой мы можем добавить нового сотрудника.



Рисунок 3.7 – Форма «Личная карточка. Добавление сотрудника»

Внизу расположены кнопки «Добавить» и «Выход». При нажатии на «Добавить» мы добавляем в базу нового сотрудника, а при нажатии на «Выход» - возвращаемся на основную форму.

Следующий пункт меню – «Удалить». При нажатии на него открывается форма для удаления.

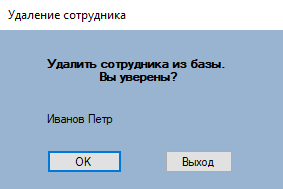


Рисунок 3.8 – Форма «Удаление сотрудника»

Внизу расположены кнопки «ОК» и «Выход». При нажатии на «ОК» выбранный сотрудник удаляется из базы для расчета заработной платы, а при нажатии на «Выход» - возвращаемся на основную форму.

Далее рассмотрим пункт «Редактировать». При нажатии на него открывается форма редактирования личной карточки работника (Рисунок 3.9).

В ней по умолчанию сохранены данные выбранного сотрудника для более удобного редактирования: бухгалтер может изменить необходимые поля и не переписывать остальные повторно.

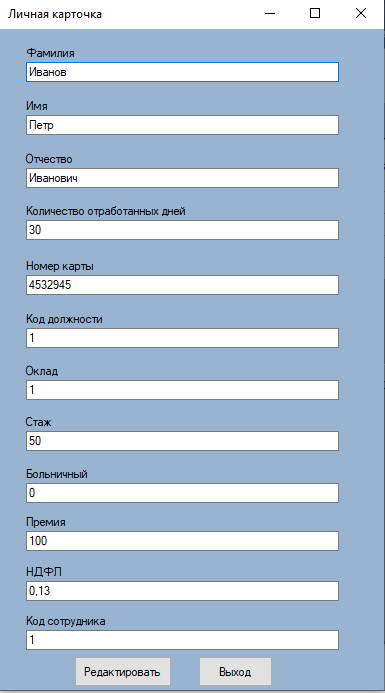
****

Рисунок 3.9 – Форма для редактирования данных о сотруднике

Внизу расположены также кнопки «Редактировать» и «Выход». При нажатии на «Редактировать» изменяются данные о выбранном сотруднике, а при нажатии на «Выход» происходит возврат на основную форму.

* + 1. **Предоставление отчетов**

Следующим выпадающим меню является меню «Отчеты». Оно включает в себя создание следующих отчетов:

1. Расчетный листок,
2. Ведомость выдачи заработной платы,
3. Ведомость перевода заработной платы на карту
4. Сводный отчет по предприятию;

При нажатии на первый пункт меню «Расчетный листок» открывается форма «Расчетный листок», которая отображает таблицу, содержащую данные о различных начислениях и удержаниях.

Начисления включают в себя:

1. Оклад,
2. Начисления за стаж,
3. Начисления за нахождение на больничном,
4. Премиальные начисления;

Удержания включают в себя НДФЛ (подоходный налог).

Итоговая сумма, которой и является заработной платой, посчитана с учетом отработанных за месяц дней. Для подсчета используются специальный коэффициент, отображающий отношение рабочих дней к дням в месяце. При этом, если месяц был отработан целиком, коэффициент равен единице.

Форма представлена на рисунке 3.10

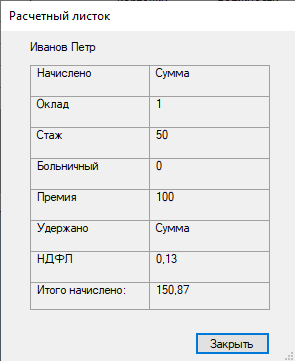


Рисунок 3.10 – Форма «Расчетный листок»

Внизу формы располагается кнопка «Закрыть», при нажатии на которую мы попадаем на основную форму.

Следующий пункт из меню «Отчеты» - «Ведомость выдачи заработной платы».

Ведомость представлена на рисунке 3.11

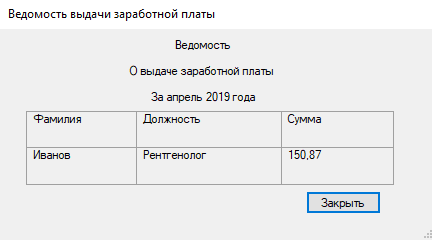


Рисунок 3.11 – Форма «Ведомость выдачи заработной платы»

Форма представляет собой формальную шапку и данные о выбранном сотруднике, заключенные в таблицу.

Таблица состоит из трех столбцов:

1. Фамилия,
2. Должность,
3. Сумма;

Сумма (заработная плата) посчитана с учетом рабочих дней в месяце.

Внизу формы располагается кнопка «Закрыть», при нажатии на которую мы попадаем на основную форму.

Следующим пунктом меню является «Ведомость перевода заработной платы на карту». Нажав на этот пункт, мы видим форму, аналогичную предыдущей. Она также состоит из шапки и таблицы с данными о выбранном сотруднике. Ведомость представлена на рисунке 3.12.

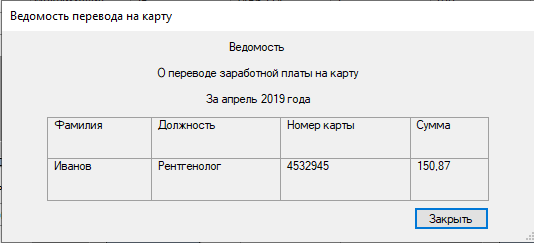
****

Рисунок 3.12 – Форма «Ведомость перевода заработной платы на карту»

Внизу формы располагается кнопка «Закрыть», при нажатии на которую мы попадаем на основную форму.

Следующим и заключительным отчетом является «Сводный отчет по предприятию».

Отчет представляет собой форму, которая содержит сводную таблицу по всему предприятию.

В таблице значения получены путем суммирования всех показателей (Оклад, Начисления за стаж, Начисления в связи с пребыванием на больничном, Премия, НДФЛ) с учетом отработанных за месяц дней по всем сотрудникам.

Внизу на данной форме находится кнопка «Закрыть», при нажатии на которую происходит возврат на основную форму для последующей работы с информационной системой.

Данный отчет представлен на рисунке 3.13.

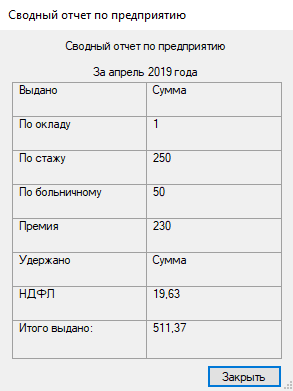
****

Рисунок 3.13 – Форма «Сводный отчет по предприятию»

* 1. **Руководство пользователя**

Для запуска программы запустите файл WindowsFormsApp1.exe.

Когда файл запущен, следующим пунктом является авторизация. Если вы бухгалтер, введите:

Логин: *admin*

Пароль: *admin*

После ввода данных нажмите на кнопку «ОК».

Если же вы открыли файл по ошибке, закройте его. Программа содержит личные данные сотрудников больницы.

Если вы без ошибок ввели логин и пароль, вам откроется основная форма, на которой показаны таблицы с данными о сотрудниках, должностях и отделениях больницы.

Все данные располагаются в системе управления базами данных *MS Access*.

Чтобы ознакомиться непосредственно с ними, откройте файл «*Enterprise.mdb*».

Вверху основной формы находятся два выпадающих меню, наведите на необходимый. Если вы желаете работать с сотрудниками, то вам откроется меню для работы с ними. Вы можете удалить, добавить сотрудника в базу, а также отредактировать информацию о существующем работнике.

Если же вы желаете работать с отчетами, то перед вами меню для создания отчетов.

В таблице с данными сотрудников нажмите на строку, где содержится информация о необходимом работнике. Затем нажмите на нужный отчет. Их четыре:

1. расчетный листок,
2. ведомость выдачи заработной платы,
3. ведомость перевода заработной платы на карту,
4. сводный отчет по предприятию;

После того, как вы выберете отчет, он откроется в новой форме.

Когда вы решите завершить работу в системе, не закрывайте программу, используя панель задач внизу экрана.

Для выхода предусмотрены кнопки «Выход» на каждой из форм. После выхода из основной формы, вы вновь попадете на окно авторизации.

Это сделано для того, чтобы не перезапускать программу заново, если вы вдруг случайно нажали на «Выход».

Если же вы хотите выйти из программы, на форме авторизации нажмите «Выход». Программа закроется.

**ТЕСТИРОВАНИЕ**

Программа прошла проверку на различные ошибки и возникновение различных исключений.

При вводе некорректных данных на моменте авторизации (неверный логин или пароль), пользователю открывается форма, в которой описана причина возникновения ошибки, а также предложение ввести данные заново. Форма представлена на рисунке 4.1

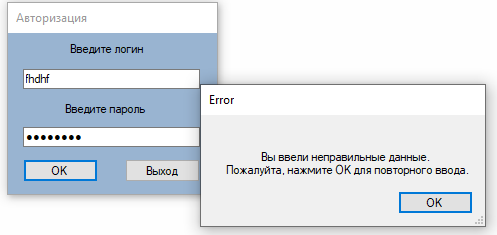


Рисунок 4.1 – Форма с информацией об ошибке при авторизации

При вводе некорректных данных при добавлении или редактировании информации о сотрудниках, пользователь также видит форму, описывающую причину появления исключения, а также предложение ввести информацию заново. Форма представлена на рисунке 4.2.

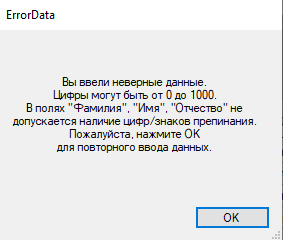


Рисунок 4.2 – Форма с информацией об ошибке при вводе данных

В обоих случаях (и во время авторизации, и во время ввода данных) при нажатии на кнопку ОК пользователь перейдет к повторному вводу данных (исходная форма обновится), либо он может выйти из программы. Пример выхода расположен на рисунке 4.3.

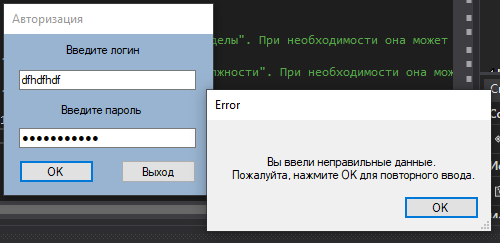
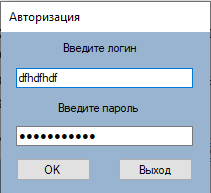
 

Рисунок 4.3 – Действия при нажатии кнопки ОК

В данном случае очищение полей логина и пароля не происходит. Это сделано для того, чтобы пользователь мог посмотреть на введенные данные и исправить ошибки в них для последующего успешного входа в программу.

Весь необходимый функционал приложения был реализован с помощью графических элементов.

Для создания приложения был выбран лаконичный и удобный интерфейс, который не пестрит обилием декоративных элементов и не будет отвлекать от работы конечного пользователя. На всех графических элементах понятно описан принцип работы, который будет ясен даже самым неопытным пользователям. В приложении отсутствуют грамматические ошибки.

Приложение успешно прошло проверку по следующим критериям:

1. правильность написания текста и выравнивание,
2. унификация дизайна (цвет, шрифт, размер),
3. расположение, размер, цвет, ширина, длина элементов; возможность ввода букв или цифр,
4. реализуется ли функционал приложения с помощью графических элементов,
5. читабелен ли использованный шрифт,
6. переходит ли курсор из текстового в поинтер при наведении на активные элементы, выделяются ли выбранные элементы,
7. выравнивание текста и форм,
8. проверить расположение и отображение всех элементов при различных разрешениях экрана,
9. полное понимание интерфейса пользователями и отсутствие неоднозначностей при работе с ним,
10. минимальный объем вводимой пользователями информации,
11. простота интерфейса,
12. минимальное время на выполнение задач пользователем,
13. минимальное количество ошибок, которые допускает пользователь при работе с приложением,
14. четкое разграничение элементов на экране,
15. визуальная привлекательность интерфейса;

Краткие сведения о процессе тестирования представлены в таблице 1.

Таблица 1 - Тестирование интерфейса

|  |  |  |
| --- | --- | --- |
| № | Вид проверки | Результат |
| 1 | Реализуется ли функционал приложения при помощи | 10 |
|  | графических элементов |  |
| 2 | Реализуется ли размещение всех сообщений об ошибках, | 8 |
|  | уведомлений |  |
| 3 | Адаптация под разные размеры экрана | 10 |
| 4 | Отсутствуют ли орфографически, пунктуационные | 9 |
|  | ошибки |  |
| 5 | Читабелен ли используемый шрифт | 9 |
| 6 | Изменяется ли вид курсора при наведении на различные | 8 |
|  | элементы |  |

Минимальные требования для пользователя:

1. Платформа Windows XP/Vista/7/8/10;
2. Архитектура 32/64 бит;
3. Процессор 1 ГГц;
4. Оперативная память 1 Гб;
5. Клавиатура, экран, мышь.

**ЗАКЛЮЧЕНИЕ**

Была создана система для расчета заработной платы работников бюджетной сферы.

Как конкретное предприятие была выбрана больница.

В ходе работы была определена постановка задачи. Поставленная задача была реализована на объектно-ориентированном языке C# на платформе .NET версии 4.7.2 в среде Microsoft Visual Studio 2019 с использованием Windows Forms и реляционной системы управления базами данных MS Access 2019

Проект реализован с целью и возможностью дальнейшего развития.

Поставленные задачи курсового проекта можно считать успешно выполненными.

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Герберт Шилдт. С# 4.0: Полное руководство = с# 4.0 The Complete

Reference - Москва: Вильямс, 2010. – с.1056

1. Павловская Т. А. C#. Программирование на языке высокого уровня / Т. А. Павловская. – Санкт-Петербург: Питер Пресс, 2009. – 432 с.
2. GUI Testing: Complete Guide. [Электронный ресурс]. — Режим доступа: http://www.guru99.com/gui-testing.html - Дата доступа: 16. 05. 2019.

4. .NET Framework 4.6 и 4.5[Электронный ресурс] – 2017 – Режим доступа: https://msdn.microsoft.com/ru-ru/library/w0x726c2(v=vs.110).aspx – Дата доступа: 16. 05. 2019.

**ПРИЛОЖЕНИЕ А - ЛИСТИНГ ПРОГРАММЫ**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.OleDb;

namespace WindowsFormsApp1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void ToolStripMenuItem1\_Click(object sender, EventArgs e)

{

}

private void Form1\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "enterpriseDataSet.Отделы". При необходимости она может быть перемещена или удалена.

this.отделенияTableAdapter1.Fill(this.enterpriseDataSet1.Отделения);

// TODO: данная строка кода позволяет загрузить данные в таблицу "enterpriseDataSet.Должности". При необходимости она может быть перемещена или удалена.

this.должностиTableAdapter1.Fill(this.enterpriseDataSet1.Должности);

// TODO: данная строка кода позволяет загрузить данные в таблицу "enterpriseDataSet.Сотрудники". При необходимости она может быть перемещена или удалена.

this.сотрудникиTableAdapter1.Fill(this.enterpriseDataSet1.Сотрудники);

}

private void addToolStripMenuItem1\_Click(object sender, EventArgs e)

{

FormAddWorker f = new FormAddWorker(); // создать форму

if (f.ShowDialog() == DialogResult.OK) // отобразить форму

{

// если OK, то добавить работника

string workerSurname = f.textBox1.Text;

string workerName = f.textBox2.Text;

string workerPatronymic = f.textBox3.Text;

int numberOfWorkDays = Convert.ToInt32(f.textBox6.Text);

int cardNumber = Convert.ToInt32(f.textBox5.Text);

int positionID = Convert.ToInt32(f.textBox6.Text);

double rate = Convert.ToDouble(f.textBox7.Text);

double experience = Convert.ToDouble(f.textBox4.Text);

double sickLeave = Convert.ToDouble(f.textBox10.Text);

double premium = Convert.ToDouble(f.textBox11.Text);

double tax = Convert.ToDouble(f.textBox12.Text);

int workerID = Convert.ToInt32(f.textBox8.Text);

// работает

сотрудникиTableAdapter1.Insert(rate, experience, sickLeave, premium, workerID, workerSurname, workerName, workerPatronymic, numberOfWorkDays, cardNumber, positionID, tax); // вставка

сотрудникиTableAdapter1.Fill(enterpriseDataSet1.Сотрудники); // отображение

}

}

private void deleteToolStripMenuItem\_Click(object sender, EventArgs e)

{

FormDeleteWorker formDeleteWorker = new FormDeleteWorker(); // создать форму

int index;

// взять номер текущей (выделенной) строки в dataGridView1

index = dataGridView1.CurrentRow.Index;

// заполнить внутренние переменные из текущей строки dataGridView1

int workerID = Convert.ToInt32(dataGridView1[0, index].Value);

string workerSurname = Convert.ToString(dataGridView1[1, index].Value); ;

string workerName = Convert.ToString(dataGridView1[2, index].Value);

string workerPatronymic = Convert.ToString(dataGridView1[3, index].Value);

int numberOfWorkDays = Convert.ToInt32(dataGridView1[4, index].Value);

int cardNumber = Convert.ToInt32(dataGridView1[5, index].Value);

int positionID = Convert.ToInt32(dataGridView1[6, index].Value);

double rate = Convert.ToDouble(dataGridView1[7, index].Value);

double experience = Convert.ToDouble(dataGridView1[8, index].Value);

double sickLeave = Convert.ToDouble(dataGridView1[9, index].Value);

double premium = Convert.ToDouble(dataGridView1[10, index].Value);

double tax = Convert.ToDouble(dataGridView1[11, index].Value);

// сформировать информационную строку

formDeleteWorker.label2.Text = workerSurname + " " + workerName;

if (formDeleteWorker.ShowDialog() == DialogResult.OK)

{

сотрудникиTableAdapter1.Delete(rate, experience, sickLeave, premium, workerID, workerSurname, workerName, workerPatronymic, numberOfWorkDays, cardNumber, positionID, tax); // метод Delete

сотрудникиTableAdapter1.Fill(enterpriseDataSet1.Сотрудники);

}

}

private void editToolStripMenuItem\_Click(object sender, EventArgs e)

{

FormEditWorker formEditWorker = new FormEditWorker(); // создать форму

int index;

if (dataGridView1.RowCount <= 1) return;

// получить позицию выделенной строки в dataGridView1

index = dataGridView1.CurrentRow.Index;

if (index == dataGridView1.RowCount - 1) return; //

// получить данные строки

int workerID = Convert.ToInt32(dataGridView1[0, index].Value);

string workerSurname = Convert.ToString(dataGridView1[1, index].Value); ;

string workerName = Convert.ToString(dataGridView1[2, index].Value);

string workerPatronymic = Convert.ToString(dataGridView1[3, index].Value);

int numberOfWorkDays = Convert.ToInt32(dataGridView1[4, index].Value);

int cardNumber = Convert.ToInt32(dataGridView1[5, index].Value);

int positionID = Convert.ToInt32(dataGridView1[6, index].Value);

double rate = Convert.ToDouble(dataGridView1[7, index].Value);

double experience = Convert.ToDouble(dataGridView1[8, index].Value);

double sickLeave = Convert.ToDouble(dataGridView1[9, index].Value);

double premium = Convert.ToDouble(dataGridView1[10, index].Value);

double tax = Convert.ToDouble(dataGridView1[11, index].Value);

// заполнить поля формы f

formEditWorker.textBox1.Text = workerSurname;

formEditWorker.textBox2.Text = workerName;

formEditWorker.textBox3.Text = workerPatronymic;

formEditWorker.textBox9.Text = numberOfWorkDays.ToString();

formEditWorker.textBox5.Text = cardNumber.ToString();

formEditWorker.textBox6.Text = positionID.ToString();

formEditWorker.textBox7.Text = rate.ToString();

formEditWorker.textBox4.Text = experience.ToString();

formEditWorker.textBox10.Text = sickLeave.ToString();

formEditWorker.textBox11.Text = premium.ToString();

formEditWorker.textBox12.Text = tax.ToString();

formEditWorker.textBox8.Text = workerID.ToString();

if (formEditWorker.ShowDialog() == DialogResult.OK) // вызвать форму FormEditWorker

{

// получить новые (измененные) значения из формы

string newWorkerSurname = formEditWorker.textBox1.Text;

string newWorkerName = formEditWorker.textBox2.Text;

string newWorkerPatronymic = formEditWorker.textBox3.Text;

int newNumberOfWorkDays = Convert.ToInt32(formEditWorker.textBox9.Text);

int newCardNumber = Convert.ToInt32(formEditWorker.textBox5.Text);

int newPositionID = Convert.ToInt32(formEditWorker.textBox6.Text);

double newRate = Convert.ToDouble(formEditWorker.textBox7.Text);

double newExperience = Convert.ToDouble(formEditWorker.textBox4.Text);

double newSickLeave = Convert.ToDouble(formEditWorker.textBox10.Text);

double newPremium = Convert.ToDouble(formEditWorker.textBox11.Text);

double newTax = Convert.ToDouble(formEditWorker.textBox12.Text);

// сделать изменения в адаптере

this.сотрудникиTableAdapter1.Update(newRate, newExperience, newSickLeave, newPremium, workerID, newWorkerSurname, newWorkerName, newWorkerPatronymic, newNumberOfWorkDays, newCardNumber, newPositionID, newTax, rate, experience, sickLeave, premium, workerID, workerSurname, workerName, workerPatronymic, numberOfWorkDays, cardNumber, positionID, tax);

this.сотрудникиTableAdapter1.Fill(this.enterpriseDataSet1.Сотрудники);

}

}

private void paySheetToolStripMenuItem\_Click(object sender, EventArgs e)

{

FormPaySheet formPaySheet = new FormPaySheet();

int index;

// взять номер текущей (выделенной) строки в dataGridView1

index = dataGridView1.CurrentRow.Index;

// заполнить внутренние переменные из текущей строки dataGridView1

string workerSurname = Convert.ToString(dataGridView1[1, index].Value);

string workerName = Convert.ToString(dataGridView1[2, index].Value);

int numberOfWorkDays = Convert.ToInt32(dataGridView1[4, index].Value);

double rate = Convert.ToDouble(dataGridView1[7, index].Value);

double experience = Convert.ToDouble(dataGridView1[8, index].Value);

double sickLeave = Convert.ToDouble(dataGridView1[9, index].Value);

double premium = Convert.ToDouble(dataGridView1[10, index].Value);

double tax = Convert.ToDouble(dataGridView1[11, index].Value);

formPaySheet.label12.Text = rate.ToString();

formPaySheet.label13.Text = experience.ToString();

formPaySheet.label14.Text = sickLeave.ToString();

formPaySheet.label15.Text = premium.ToString();

formPaySheet.label16.Text = tax.ToString();

formPaySheet.label17.Text = (rate \* (numberOfWorkDays / 30) + experience + sickLeave + premium - tax).ToString();

formPaySheet.label6.Text = workerSurname + " " + workerName;

formPaySheet.ShowDialog();

}

private void payrollSheetToolStripMenuItem\_Click(object sender, EventArgs e)

{

FormPayrollSheet formPayrollSheet = new FormPayrollSheet();

int index;

// взять номер текущей (выделенной) строки в dataGridView1

index = dataGridView1.CurrentRow.Index;

// заполнить внутренние переменные из текущей строки dataGridView1

string workerSurname = Convert.ToString(dataGridView1[1, index].Value);

int numberOfWorkDays = Convert.ToInt32(dataGridView1[4, index].Value);

double rate = Convert.ToDouble(dataGridView1[7, index].Value);

double experience = Convert.ToDouble(dataGridView1[8, index].Value);

double sickLeave = Convert.ToDouble(dataGridView1[9, index].Value);

double premium = Convert.ToDouble(dataGridView1[10, index].Value);

double tax = Convert.ToDouble(dataGridView1[11, index].Value);

double salary = rate \* (numberOfWorkDays / 30) + experience + sickLeave + premium - tax;

string position = Convert.ToString(dataGridView2[2, index].Value);

formPayrollSheet.label5.Text = workerSurname;

formPayrollSheet.label6.Text = position;

formPayrollSheet.label7.Text = salary.ToString();

formPayrollSheet.ShowDialog();

}

private void translationListToolStripMenuItem\_Click(object sender, EventArgs e)

{

FormTranslationList formTranslationList = new FormTranslationList();

// взять номер текущей (выделенной) строки в dataGridView1

int index = dataGridView1.CurrentRow.Index;

// заполнить внутренние переменные из текущей строки dataGridView1

string workerSurname = Convert.ToString(dataGridView1[1, index].Value);

int numberOfWorkDays = Convert.ToInt32(dataGridView1[4, index].Value);

int cardNumber = Convert.ToInt32(dataGridView1[5, index].Value);

double rate = Convert.ToDouble(dataGridView1[7, index].Value);

double experience = Convert.ToDouble(dataGridView1[8, index].Value);

double sickLeave = Convert.ToDouble(dataGridView1[9, index].Value);

double premium = Convert.ToDouble(dataGridView1[10, index].Value);

double tax = Convert.ToDouble(dataGridView1[11, index].Value);

double salary = rate\*(numberOfWorkDays/30) + experience + sickLeave + premium - tax;

string position = Convert.ToString(dataGridView2[2, index].Value);

formTranslationList.label6.Text = workerSurname;

formTranslationList.label7.Text = position;

formTranslationList.label8.Text = cardNumber.ToString();

formTranslationList.label9.Text = salary.ToString();

formTranslationList.ShowDialog();

}

private void reportToolStripMenuItem\_Click(object sender, EventArgs e)

{

FormEnterpriceReport formEnterpriceReport = new FormEnterpriceReport(this);

int index = dataGridView1.CurrentRow.Index;

string workerSurname = Convert.ToString(dataGridView1[1, index].Value);

formEnterpriceReport.ShowDialog();

}

}

}

Форма для сводного отчета:

public partial class FormEnterpriceReport : Form

{

private Form1 form1;

public FormEnterpriceReport()

{

InitializeComponent();

}

public FormEnterpriceReport(Form1 form1)

{

InitializeComponent();

this.form1 = form1;

}

private void FormEnterpriceReport\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "enterpriseDataSet.Должности". При необходимости она может быть перемещена или удалена.

this.должностиTableAdapter.Fill(this.enterpriseDataSet.Должности);

// TODO: данная строка кода позволяет загрузить данные в таблицу "enterpriseDataSet.Сотрудники". При необходимости она может быть перемещена или удалена.

this.сотрудникиTableAdapter.Fill(this.enterpriseDataSet.Сотрудники);

double sumRate = 0;

double sumExperience = 0;

double sumSickLeave = 0;

double sumPremium = 0;

double sumTax = 0;

for (int i = 0; i < form1.dataGridView1.RowCount-1; i++)

{

// int index = form1.dataGridView1.CurrentRow.Index;

int numberOfWorkDays = Convert.ToInt32(form1.dataGridView1[4, i].Value);

double rate = Convert.ToDouble(form1.dataGridView1[7, i].Value);

double experience = Convert.ToDouble(form1.dataGridView1[8, i].Value);

double sickLeave = Convert.ToDouble(form1.dataGridView1[9, i].Value);

double premium = Convert.ToDouble(form1.dataGridView1[10, i].Value);

double tax = Convert.ToDouble(form1.dataGridView1[11, i].Value);

sumRate += (rate\*(numberOfWorkDays/30));

sumExperience += experience;

sumSickLeave += sickLeave;

sumPremium += premium;

sumTax += tax;

double salary = sumRate + sumExperience + sumSickLeave + sumPremium - sumTax;

label7.Text = sumRate.ToString();

label8.Text = sumExperience.ToString();

label9.Text = sumSickLeave.ToString();

label10.Text = sumPremium.ToString();

label15.Text = sumTax.ToString();

label16.Text = salary.ToString();

}

}

}

}