Министерство образования и науки Республики Башкортостан

Государственное бюджетное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и вычислительной техники

УТВЕРЖДАЮ

Заместитель директора

по учебной работе

_____ З.З. Курмашева

«___» _____ 2021 г.

ЗАДАНИЕ

на курсовой проект студенту дневного отделения, группы 18П-1

специальности 09.02.03 Программирование в компьютерных системах

Фамилия, имя, отчество: Командов Максим Олегович

Тема курсового проекта: «Проектирование базы данных для учета заработной платы сотрудников»

Текст задания:

при выполнении курсового проекта должны быть решены следующие задачи:

а) спроектирована база данных;

б) разработана структура программы;

в) реализованы функции расчёта зарплаты, налогов, отчислений и налоговых льгот, формирования расчётного листа.

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 Култыгин, О. П. Култыгин, О. П. Администрирование баз данных. СУБД MS SQL Server [Текст] : учеб. пособ. / О. П. Култыгин. - М.: МФПА, 2012. - 232 с.

2 Фуфаев, Э.В. Базы данных [Текст]: учеб. пособ. для студ. учрежд. сред. проф. образования / Э.В. Фуфаев, Д.Э. Фуфаев. - 6-е изд., стер. - М.: Издательский центр «Академия», 2012.- 320 с.- (Среднее профессиональное образование)

3 Википедия [Электронный ресурс] // Свободная энциклопедия. – Режим доступа: http://ru.wikipedia.org/wiki/, свободный

Задание к выполнению получил «29» января 2021 г.

Студент Командов Максим Олегович

Срок окончания «30» мая 2021 г.

Руководитель курсового проекта_____Р.Ф. Каримова

Задание рассмотрено на заседании цикловой комиссии информатики

«11» января 2021 г.

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 Командов Максим Олегович

Группа 18П-1

Специальность 09.02.03 Программирование в компьютерных системах

Тема Проектирование базы данных для учета заработной платы сотрудников

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

«____» _____2021 г.

Подпись_____

АННОТАЦИЯ

Пояснительная записка к курсовому проекту содержит постановку и программу решения задачи «Проектирование базы данных для учета заработной платы сотрудников»

Программа BDZarplata.exe написана на языке C# в среде программирования Visual Studio 2019 с использованием сервера баз данных MS SQL SERVER, предназначена для работы в операционной системе MS Windows 10, отлажена на данных контрольного примера.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Среди наиболее трудоемких участков бухгалтерского учета особое место занимают расчеты с персоналом по оплате труда. В роли объектов учета могут выступать десятки, сотни и даже тысячи человек, по каждому из которых нужно учитывать и обрабатывать достаточно большие объемы данных.

Актуальность данной работы обусловлена тем, что расчет заработной платы сотрудников производится бухгалтерами либо с помощью программы «1С-бухгалтерия», либо вручную. Так как программа «1С-бухгалтерия» очень сложна в применении, и ее может освоить не каждый бухгалтер, то расчет заработной платы производится с помощью электронных таблиц. Это довольно трудоемкий процесс, который занимает очень много времени и ресурсов компьютера. Данная курсовая работа, ориентирована на простое освоение и комфортную работу бухгалтеров.

Цель курсового проекта – разработка приложения для упрощения процесса учета заработной платы бухгалтерам, путем автоматизации расчёта заработной платы, а также проверки корректности вводимых данных.

Задачами курсового проекта являются:

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

1. Постановка задачи

1.1. Описание предметной области

Задача – автоматизировать расчёт зарплаты, содержание работника и суммы налогов, упростить процесс выплаты зарплаты для упрощения работы Бухгалтера

Информационная система должна обеспечивать: ввод, изменение анкетных данных работников, сведения о болезнях, надбавках; ежемесячный перерасчет зарплаты с выдачей ведомости на экран и печать.

С общей суммы зарплаты отчисляется подоходный налог.

1.1.1. Круг пользователей

- Сотрудники Отдела кадров имеют право:
  - добавлять новых работников в БД;
  - изменять ФИО и другие анкетные данные сотрудников;
  - изменять должность работников;
  - изменять статус работника.
- Сотрудники бухгалтерии имеют право:
  - изменять оклад в соответствии с должностью;
  - изменять показатели процента налогов;
  - Изменять константы влияющие на расчет зарплаты;
  - создавать выписки по ЗП;
  - начислять штрафы и надбавки работникам соответствующих отделов.
- Рядовые сотрудники имеют право:
  - получать выписку по ЗП.

1.1.2. Константы

МРОТ 12 792 руб.

Налоги, которые платит предприятие:

Пенсионный фонд. Тариф составляет 22% для работников, доход которых за год не превышает 1,292 млн рублей и 10% — с больших сумм.

Фонд соцстрахования. Платеж составляет 2,9% для работников, которые в ход получают заработную плату в размере до 912 тыс. рублей. Если зарплата больше, на сумму больше предельной платеж не начисляется.

Медицинское страхование. Размер ежемесячного платежа составляет 5,1%

Существует также платеж «на травматизм». Процентная ставка определяется в зависимости от того, в какой сфере деятельности работает компания. Минимальная ставка составляет 0,2%, максимальная 8,5%.

1.1.3. Промежуточные данные

Информация о ЗП:

- Доплата за выход в праздник(выходной);
- Льготы;
- Льготы за детей (НДФЛ);
- Вычет на ребенка (детей) предоставляется до месяца, в котором доход налогоплательщика, облагаемый по ставке 13% и исчисленный нарастающим итогом с начала года, превысил 350 000 рублей. Вычет отменяется с месяца, когда доход сотрудника превысил эту сумму;
- на первого и второго ребенка – 1400 рублей;
- на третьего и каждого последующего ребенка – 3000 рублей;
- на каждого ребенка-инвалида до 18 лет, или учащегося очной формы обучения, аспиранта, ординатора, интерна, студента в возрасте до 24 лет, если он является инвалидом I или II группы – 12 000 рублей родителям и усыновителям (6 000 рублей – опекунам и попечителям);

- Процент налогов на зарплату в валюте (НДФЛ для физ. лиц = 13% *налоговую базу (налоговая база = оклад + надбавки –налоговый вычет))
- Процент пенсионных вычетов;
- Итоговая ЗП работника;
- Стоимость содержания работника для предприятия.

1.1.4. Расчеты

Работник получает ЗП согласно окладу и фактически отработанным дням

Найдем дневной доход = оклад / количество рабочих дней в месяце[1]

Найдем фактический дневной доход сотрудника, учитывая статус дня в месяце:

– Если статус сотрудника = «вышел» а статус дня «рабочий» - начисляется дневной доход без изменений;

– Если статус сотрудника = «вышел» а статус дня «выходной» - начисляется дневной доход в 2х кратном объеме;

– Если статус сотрудника = «болеет» а статус дня «рабочий» или «выходной» - начисляется дневной доход в соответствие с на стр. 8 ниже;

– В ином случае дневной доход не начисляется.

Найдем фактический оклад в месяц сложив все дневные доходы в конкретном месяце.

Найдем фактический доход= фактический оклад – штрафы +надбавки

Найдем налоговую базу (НБ), изначально равную Фактическому Доходу, используя налоговые вычеты, зависящие от указанных данных в таблице сотрудник:

Если сотрудник имеет спец статус 1, то НБ = НБ - 3000 руб

Если сотрудник имеет спец статус 2, то НБ = НБ - 500 руб

[1] «количество рабочих дней в месяце» – количество полей в таблице «график Работы» с определенным табельным номером и статусом дня «рабочий»

Если сотрудник имеет детей не инвалидов[2] :Если детей менее 3 то НБ=НБ-1400*(Число детей), Если детей 3 и более то НБ = 2800 + 3000 * (Число детей-2)

Если сотрудник имеет детей инвалидов НБ = НБ - 12 000 * (число детей инвалидов)

Если сотрудник имеет опекунство над детьми инвалидами НБ = НБ - 6000 * (число детей инвалидов)

Если сотрудник имеет семейный статус «мать-одиночка» или «отец-одиночка», то выше описанные налоговые вычеты удваиваются, соответственно:

НБ = НБ - (1400 * (Число детей) * 2)

НБ = НБ - (5600 + 3000 * (Число детей - 2) * 2)

НБ = НБ - 12 000 * (число детей инвалидов) * 2

НБ = НБ - 6000 * (число детей инвалидов) * 2

Рассчитаем НДФЛ = НБ * 13% или НДФЛ = НБ – (НБ / 100 * 13)

Итоговая зарплата («на руки») = фактический доход – НДФЛ

Расчёт содержания сотрудника = фактический доход + ПФР + Соцстрах + МедСтрах + травматизм

ПФР = фактический доход * 22%

Соцстрах = фактический доход * 2,9%

Медстрах = фактический доход * 5,1%

Травматизм = фактический доход * процент травматизма

Расчёт больничного

Ср. заработок = зарплата за 2 года[3] (24 месяца) / 730 * размер больничного[4]

---

[2] распространяется на родителя, супруга (супругу) родителя, усыновителя, опекуна, попечителя, приемного родителя, супруга (супругу) приемного родителя, на обеспечении которых находится ребенок(Статья 218 НК РФ Пункт 1.4)

[3] Если превышает предельную базу страховых взносов – берётся значение базы

[4] Страховой стаж Размер больничного

менее 5 лет (60 мес.) 60% среднего заработка

от 5 до 8 лет (60-96 мес) 80% среднего заработка

8 лет и более (96 мес) 100% среднего заработка

Если количество больничных дней ≤ 3: Размер пособия (работодатель) = количество больничных дней* Ср. заработок

Если количество больничных дней > 3:

Размер пособия (работодатель) = 3 * Ср. заработок

Размер пособия (ФСС) = (количество больничных дней -3) * Ср. заработок

В случае если:

- в расчетном периоде у работника нет заработка;
- заработок работника за полный календарный месяц(Оклад) ниже МРОТ;
- стаж работника менее 6 месяцев;
- работник нарушил режим, предписанный врачом;
- больничный лист выдан вследствие алкогольного, наркотического или токсического опьянения.

Расчёт больничного идет из МРОТ:

Если количество больничных дней ≤ 3: Размер пособия (работодатель) = количество больничных дней * МРОТ

Если количество больничных дней > 3:

Размер пособия (работодатель) = 3 * МРОТ

Размер пособия (ФСС) = (количество больничных дней - 3) * МРОТ

Итоговый расчет больничного:

НДФЛ(работодатель) = Размер пособия (работодатель) * 13%

НДФЛ(ФСС) = размер пособия * 13%

Итоговый размер пособия = Размер пособия (работодатель) - НДФЛ(работодатель) + Размер пособия (ФСС) - НДФЛ(ФСС)

1.2. Описание входной информации

Входной информацией для выполнения задачи являются данные непосредственно вносимые сотрудниками Бухгалтерии и отдела кадров:

- Предельные базы для исчисления страховых взносов:
    - Год;
    - Значение базы.
- Справочник (хранящий различные константы):
    - Сумма выплаты за 1 ребенка;
    - Сумма выплаты за 3 и более детей;
    - Сумма выплаты за 1 ребенка инвалида;
    - Сумма выплаты за опекунство над 1 ребенком инвалидом;
    - Процент отчислений в пенсионный фонд;
    - Процент отчислений в фонд соц. Страхования;
    - Процент отчислений в фонд мед. Страхования;
    - Процент НДФЛ;
    - МРОТ.
- Информация о сотрудниках:
    - Страховой стаж;
    - Табельный номер[5];
    - Ф.И.О.;
    - семейное положение (не обязательно);
    - число здоровых детей (не обязательно);
    - число детей инвалидов (до 18 лет), или учащихся очной формы обучения, аспирантов, ординаторов, интернов, студентов в возрасте до 24 лет, если они являются инвалидами I или II группы (не обязательно);
    - число опекаемых детей инвалидов (не обязательно)[6];
    - счет зачисления (20 символов);
    - код должности.

[5]Поля, выделенные подчеркиванием, являются ключевыми

[6] распространяется на опекуна, попечителя, приемного родителя, супруга (супругу) приемного родителя, на обеспечении которых находится ребенок

- наличие спец статуса:
  - статус предоставляющий налоговый вычет в размере 3000 ₽;
  - статус предоставляющий налоговый вычет в размере 500 ₽ [7].

- Информация о должностях и окладах:
  - Код должности;
  - Название должности;
  - Оклад (в единой Валюте);
  - Процент травматизма (от 0,2% до 8,5%);
  - Уровень доступа.
- Информация о ЗП:
  - Табельный номер;
  - Дата выплаты;
  - Надбавки [8](сумма всех надбавок) (в единой Валюте);
  - Штрафы (сумма всех штрафов) (в единой Валюте)[9].
- График Работы:
  - Дата;
  - Табельный номер;
  - Статус сотрудника (уволен / вышел / не вышел / болеет (другая уважительная причина));
  - Статус дня (рабочий / выходной).

1.3. Описание выходной информации

[7] *Согласно статье 218 НК РФ:. Налогоплательщикам, имеющим в соответствии с подпунктами 1 и 2 пункта 1 настоящей статьи право более чем на один стандартный налоговый вычет, предоставляется максимальный из соответствующих вычетов.

[8] Денежные поощрения, не связанные с окладом (премии и т.д.)
Региональные доп. Выплаты
Доп. выплаты гос. сотрудникам за стаж

[9] Дисциплинарные, Материальная ответственность и тому подобное

Выходной информацией является расчётный лист

Описание выходных документов представлено в таблице 1.3.1.

Таблица 1.3.1 – Описание выходных документов

Наименование документа (шифр)	Периодичность выдачи документа	Кол-во экз.	Куда передаются	Поля сортировки	Поля группировки	Итоги
1	2	3	4	5	6	7
Расчётный лист	Раз в месяц или по мере необходимости	1	Сотруднику	-	-	-

Шаблоны выходных документов представлены в приложении А

1.4. Концептуальное моделирование

Концептуальная модель – это отражение предметной области, для которой разрабатывается база данных.

Можно сказать, что это некая диаграмма с принятыми обозначениями элементов. Так, все объекты, обозначающие вещи, обозначаются в виде прямоугольника. Атрибуты, характеризующие объект – в виде овала, а связи между объектами – ромбами. Мощность связи обозначается стрелками (в направлении, где мощность равна многим – двойная стрелка, а со стороны, где она равна единице - одинарная).

Концептуальная модель базы данных представлена в схеме 1.4.1

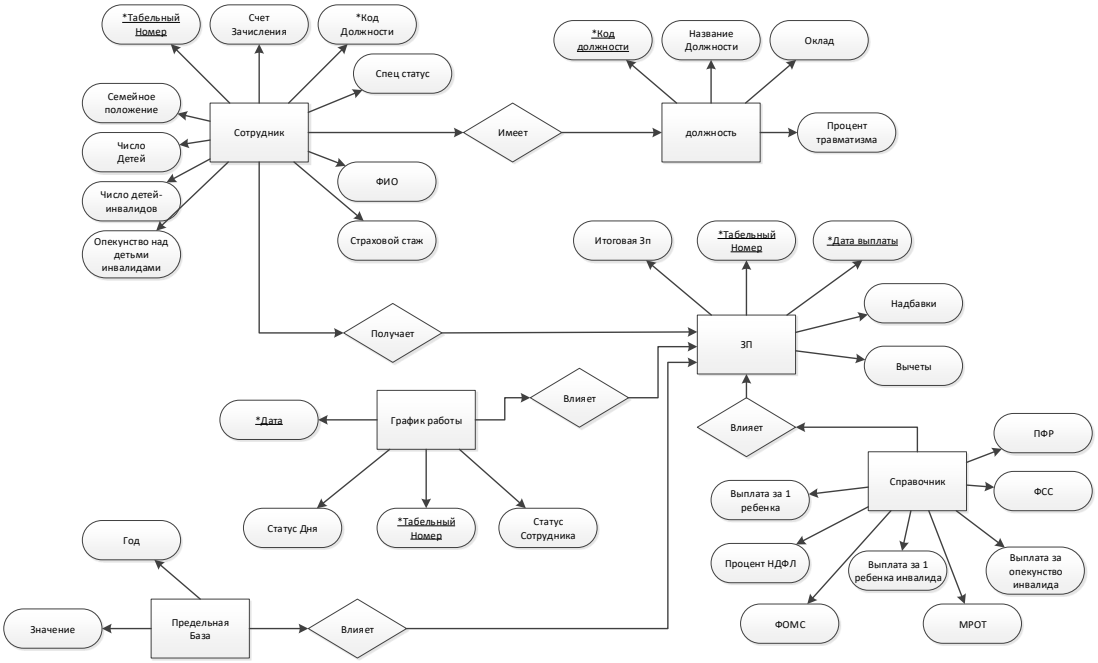


Рисунок 1.4.1 – Концептуальная модель БД

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 каждая сущность представляется в виде таблицы.

Логическое моделирование заключается в переходе от диаграммы «сущность-связь» к взаимосвязанным таблицам. Этот переход состоит из следующих шагов:

- каждая простая сущность становится таблицей;
- каждый атрибут становится столбцом таблицы;
- уникальный идентификатор сущности становится ключом таблицы;
- 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 ключей 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, поддерживаемая 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.

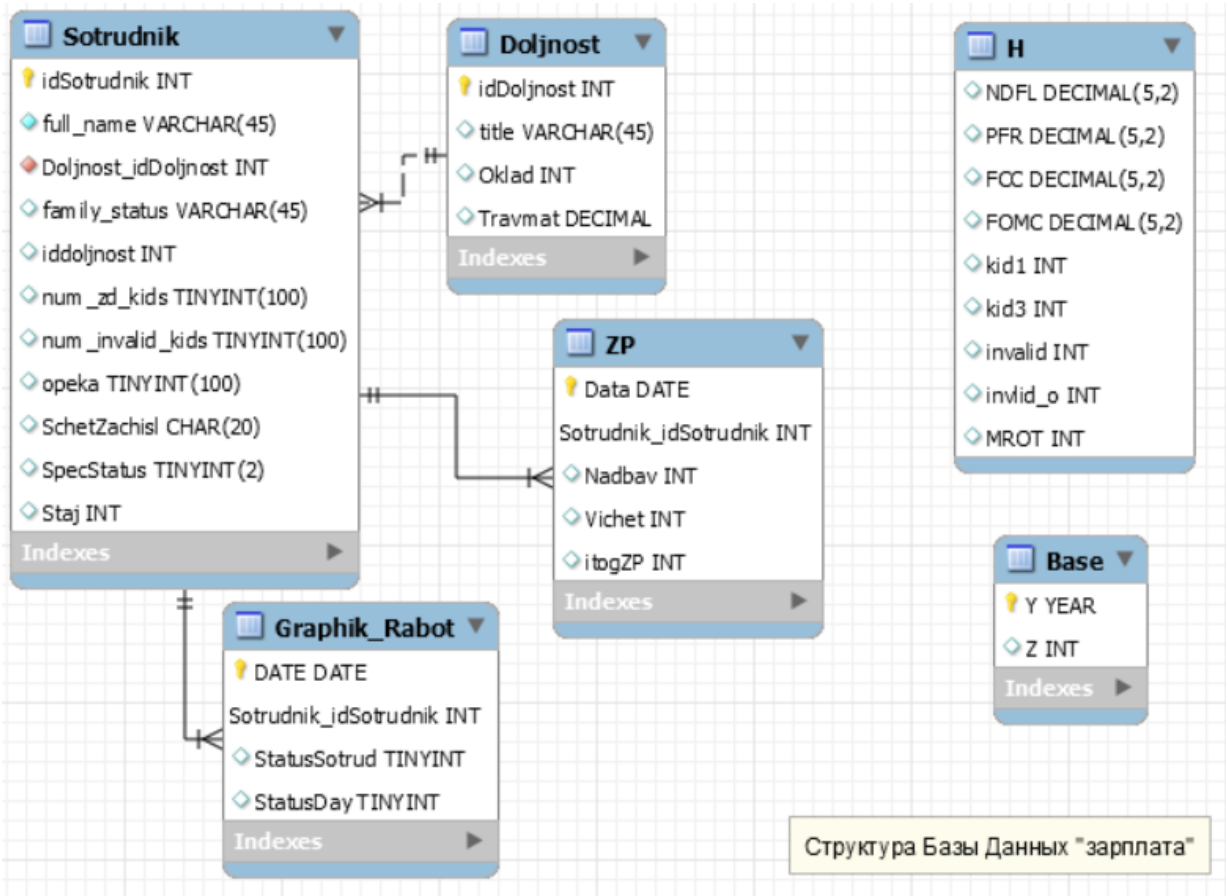


Рисунок 1.5.1 – Схема данных БД

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[10]
1	2	3	4	5
Sotrudnik (Сотрудник)				
idSotrudnik	Табельный номер сотрудника	INT	4	PK
full_name	ФИО сотрудника	Varchar(45)	46	
idDoljnost	ID Должности	INT	4	FK
family_status	Семейный статус сотрудника	VARCHAR(45)	46	
num_zd_kids	Количество здоровых детей сотрудника	TINYINT	1	
num_invalid_kids	Количество детей инвалидов	TINYINT	1	
opeka	Число опекаемых детей инвалидов	TINYINT	1	
SchetZchisl	Счёт зачисления сотрудника	CHAR(20)	21	

[10]PK-первичный ключ

FK-внешний ключ

Продолжение таблицы 1.6.1

1	2	3	4	5
SpecStatus	Наличие или отсутствие одного из спец статусов	TINYINT	1	
Staj	Страховой стаж	INT	4	
Doljnost(Должность)				
idDoljnost	ID Должности	INT	4	PK
title	Название должности	Varchar(45)	46	
Oklad	оклад	INT	4	
Travmat	Процент травматизма	DECIMIAL(2,1)		
AccessLvl	Уровень доступа к данным	TINYINT		
Graphik_Rabot(График работ)				
DATE	Дата дня	DATE	3	PK
idSotrudnik	Табельный номер сотрудника	INT	4	PK
StatusSotrud	Статус сотрудника	CHAR(10)	1	
StatusDay	Статус Дня	CHAR(10)	1	
ZP(Заработная Плата)				
DATE	Дата выплаты	DATE	3	PK
idSotrudnik	Табельный номер сотрудника	INT	4	PK
Nadbav	Сумма всех надбавок	INT	4	
itogZP	Итоговая ЗП сотруднка	INT	4	

Продолжение таблицы 1.6.1

1	2	3	4	5
Base(Предельная база для исчисления страховых взносов)				
Y	год	YEAR	1	PK
Z	Значение базы	INT	4	
H(Справочник)				
NDFL	НДФЛ	DECIMIAL(5,2)		
PFR	ПФР	DECIMIAL(5,2)		
FCC	ФСС	DECIMIAL(5,2)		
FOMC	ФОМС	INT	4	
kid1	Размер льготы за 1 и 2 ребенка	INT	4	
kid3	Размер льготы за 3 и последующих детей	INT	4	
invalid	Размер льготы за ребенка-инвалида	INT	4	
invalid_O	Размер льготы за опеку над ребенком инвалидом	INT	4	
MROT	МРОТ	INT	4	

1.8. 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. Входная информация контрольных примеров представлена в приложении Б. Выходные данные для контрольных примеров показаны в приложении В.

1.9. Общие требования к программному продукту

Пользователи должны иметь базовые навыки пользования персональным компьютером и знать основы бухгалтерского учета.

Минимальные требования к техническому обеспечению программного продукта следующие:

- Windows 10 64bit;
- Процессор 2,3 ГГц (2 ядра, 4 потока) / Intel core i3-7020U;
- Интегрированное графическое ядро Intel HD Graphics 620 или аналогичная дискретная видеокарта;
- Оперативная память 1 ГБ;
- 50 Мб свободного места на жёстком диске;
- .Net Core;
- .NET Framework.

Функциональные возможности программного продукта:

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

Требования к надежности:

– пользователь для входа в свою учетную запись должен использовать логин;
– 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 MS SQL

2. Экспериментальный раздел

2.1 Описание программы

Программа имеет модульную структуру. При ее запуске выполняется проект на BDZarplata.exe. Схема взаимодействия модулей программы представлена на рисунке 2.1.1. Описание модулей и методов представлено в таблице 2.1.1.

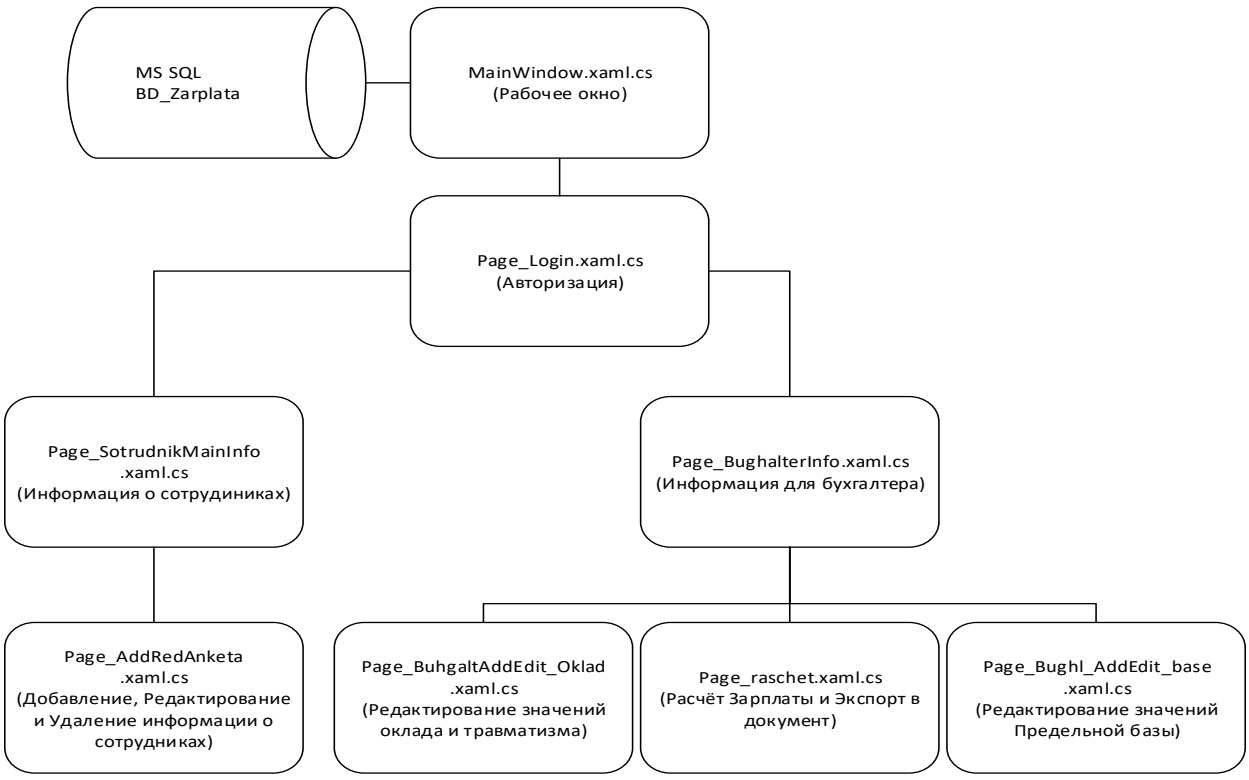


Рисунок 2.1.1– Схема взаимодействия модулей

Таблица 2.1.1. – Описание Модулей

Методы	Назначения
1	2
DB.cs	
void loadDataGrid	Загрузка данных из БД в DataGrid
void LoadDataComboBox	Загрузка данных из БД в список
void LoadDataListBox	Загрузка данных из БД в список
object queryScalar	Выполнение SQL Запроса к БД
string queryScalar	Выполнение запроса к БД
string[] queryScalar	Выполнение запроса к БД
void ReturnTable	Запрос к БД с получением данных из таблицы (таблиц)
int queryData	Выполнение внешнего SQl Файла или запроса с возвратом количества строк
DB_Connect.cs	
void OpenConnection	Создание соединения с БД
void CloseConnection	закрытие соединения с БД
bool OpenClouseConnection	Проверка на возможность установки соединения с БД
DG.cs	
string GetSelectCell	Получение значения выбранной ячейки DataGrid
Manager.cs	
void UpdateLabel	Обновление текста в Label
Procedure.cs	
void UpdateTable	Обновление значений указанной таблицы в БД

Продолжение Таблицы 2.1.1.

1	2
void InsertTable	Добавление значений указанной таблицы в БД
MainWindow.xaml.cs	
public MainWindow	Переключение на страницу входа
void BtnBack_Click	Возврат на предыдущую страницу
Page_AddRedAnketa.xaml.cs	
public Page_AddRedAnketa	Загрузка формы
void CB_Doljnost_SelectionChanged	соединение 2 списков для синхронного выбора
void intOnly_PreviewTextInput	Фильтр целочисленных значений
void Btn_Save_Click	Проверка и внесение данных в таблицу
void Btn_Delete_Click	Удаление записи
Page_BughalterInfo.xaml.cs	
public Page_BughalterInfo()	Загрузка формы
void TabI_LN_Initialized	Подгруздка справочника льгот и налогов
void Btn_Save_Click	сохранение измененных полей таблицы БД
void Btn_Redactir_Click	Переход на страницу редактирования
void intOnly_PreviewTextInput	Фильтр целочисленных значений
void floatOnly_PreviewTextInput	Фильтр дробных значений
void DG_SotridnikOklad_SelectionChanged	запись ID выделенной строки

Продолжение Таблицы 2.1.1.

1	2
void DG_SotridnikOklad_MouseDoubleClick	Переход в режим редактирования оклада
void TabI_OkladSotrud_GotFocus	Смена видимости кнопок , при выборе вкладки
void TabI_Base_GotFocus	
void TabI_LN_GotFocus	
void LB_Sotrud_FIO2_SelectionChanged	Подгрузка данных о надбавках и штрафах выбранного сотрудника в DataGrid
void LB_Sotrud_id2_SelectionChanged	
void DG_NadbavShtraf_SelectionChanged	Вывод в форму даты , надбавки и штрафов
void Btn_Raschet_Click	Переход к странице расчета
void Btn_Red_Nadbav_Click	Внесение изменений в БД
Page_Bughl_AddEdit_base.xaml.cs	
void BtnSave_Click	Сохранение данных
void intOnly_PreviewTextInput	Фильтр целочисленных значений
Page_Login.xaml.cs	
void BtnLogin_Click	Подключение к БД с указынными параметрами
void CB_IPPC_Localhost_Click	Переключение поля для ввода адреса БД
void CB_BD_NAmeDef_Click	Переключение поля для ввода названия БД
void BTN_Raschet_Click	Расчет зарплаты
void BTN_Export_Click	Экспорт в Эксель
void CB_SotrudID_SelectionChanged	Сопоставление ФИО и ID сотрудника

Продолжение Таблицы 2.1.1.

1	2
void CB_FIO_SelectionChanged	Подгрузка информации о сотруднике
void CB_Date_SelectionChanged	Изменение даты на календаре
Page_SotrudnikMainInfo.xaml.cs	
void Btn_Redactir_Click	Переход в режим редактирования
void Btn_Save_Click	сохранение измененных полей таблицы БД
void LB_Sotrud_FIO_SelectionChanged	Сопоставление ФИО и ID сотрудника
void LB_Sotrud_id_SelectionChanged	Подгрузка данных о Расписании выбранного сотрудника
void DG_Sotrud_Anketa_MouseDoubleClick	Переход в режим редактирования
void Btn_Add_Click	Переход в режим добавления нового сотрудника
void DG_Raspisnie_SelectionChanged	Подгрузка данных о выбранном дне
void BTN_RedRaspisan_Click	сохранение измененных полей таблицы БД

2.2. Протокол тестирования программного продукта

Тестирование входа Бухгалтера, при вводе корректных данных (рисунки 2.2.1–2.2.2)

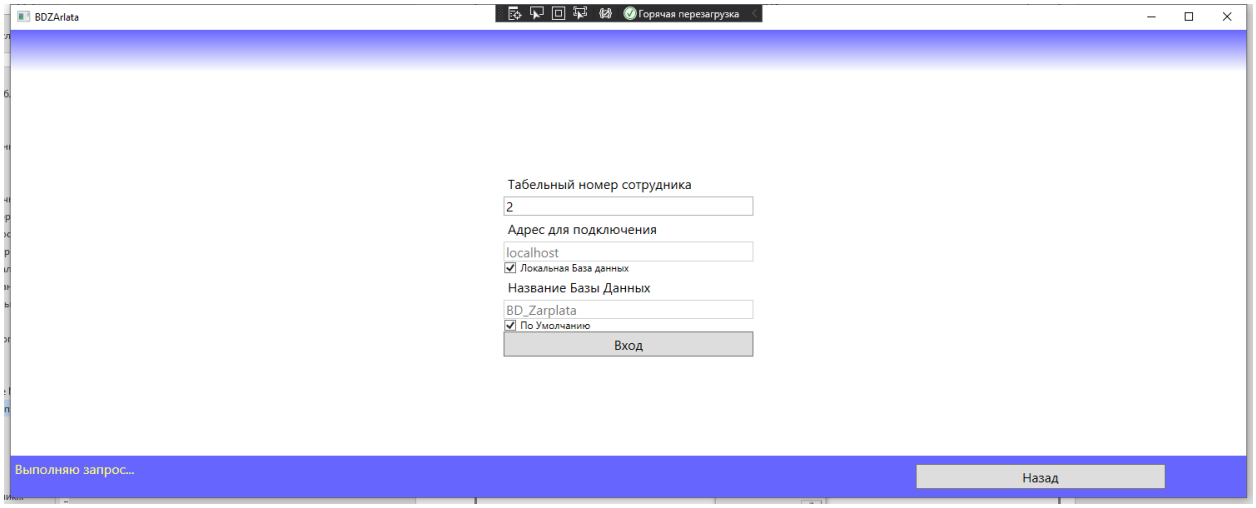


Рисунок 2.2.1– Ввод корректных данных бухгалтера

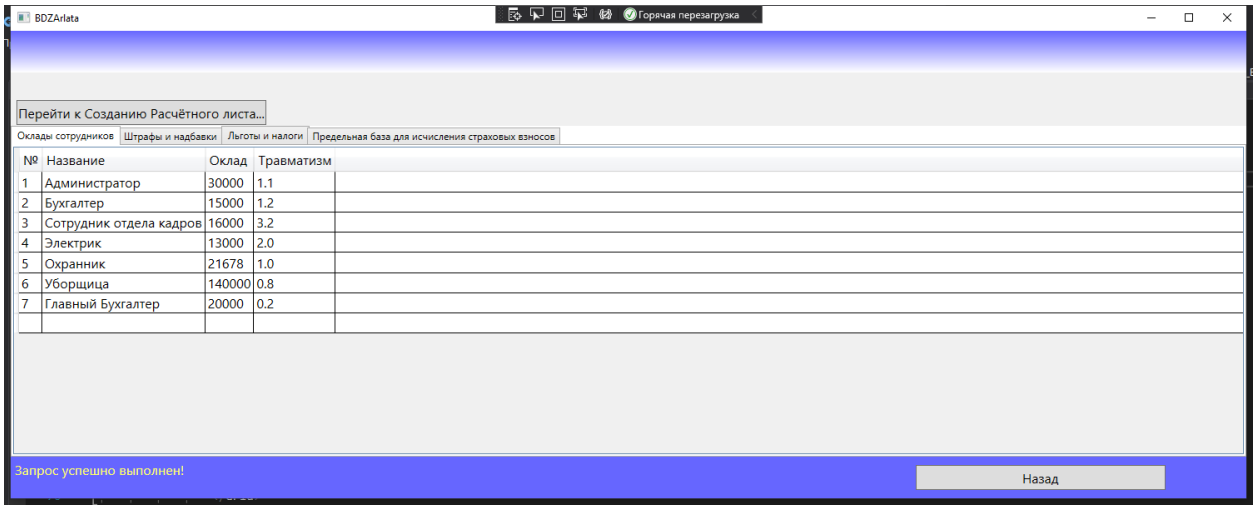


Рисунок 2.2.2– Страница бухгалтера

Тестирование входа кадровика при вводе корректных данных (рисунки 2.2.3–2.2.4)

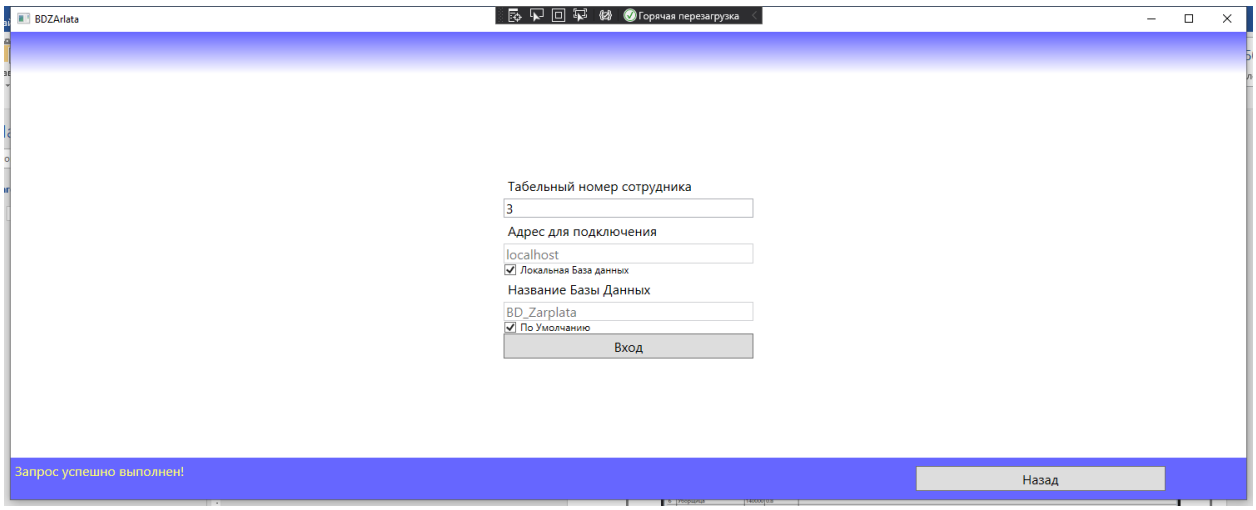
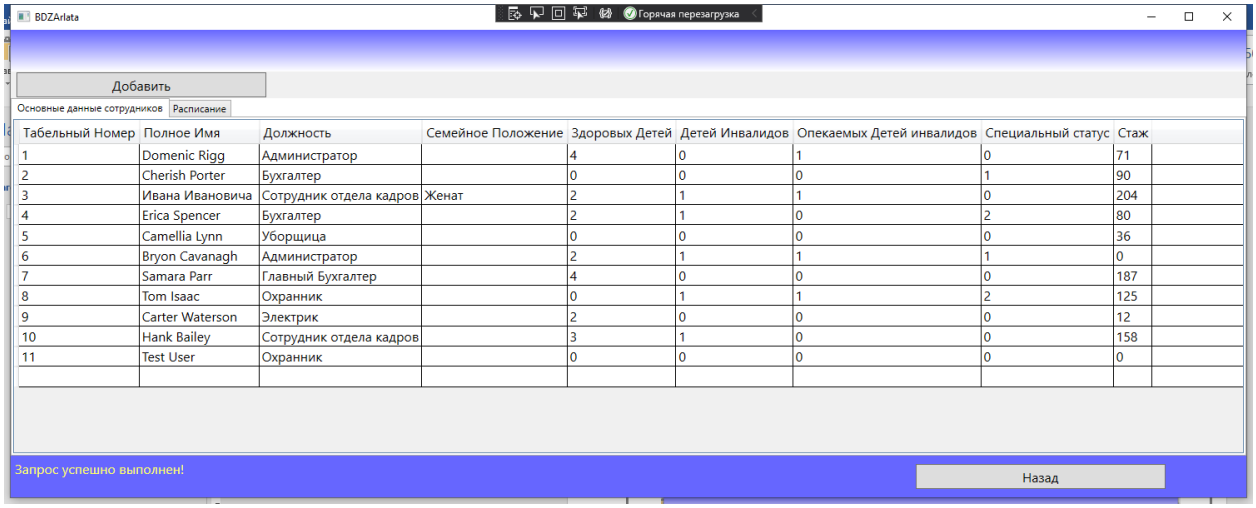


Рисунок 2.2.3– Ввод корректных данных кадровика



Табельный Номер	Полное Имя	Должность	Семейное Положение	Здоровых Детей	Детей Инвалидов	Опекаемых Детей инвалидов	Специальный статус	Стаж
1	Domenic Rigg	Администратор		4	0	1	0	71
2	Cherish Porter	Бухгалтер		0	0	0	1	90
3	Ивана Ивановича	Сотрудник отдела кадров	Женат	2	1	1	0	204
4	Erica Spencer	Бухгалтер		2	1	0	2	80
5	Camellia Lynn	Уборщица		0	0	0	0	36
6	Bryon Cavanagh	Администратор		2	1	1	1	0
7	Samara Parr	Главный Бухгалтер		4	0	0	0	187
8	Tom Isaac	Охранник		0	1	1	2	125
9	Carter Waterson	Электрик		2	0	0	0	12
10	Hank Bailey	Сотрудник отдела кадров		3	1	0	0	158
11	Test User	Охранник		0	0	0	0	0

Рисунок 2.2.4– Страница кадровика

Тестирование входа обычного пользователя при вводе корректных данных (рисунок 2.2.5)

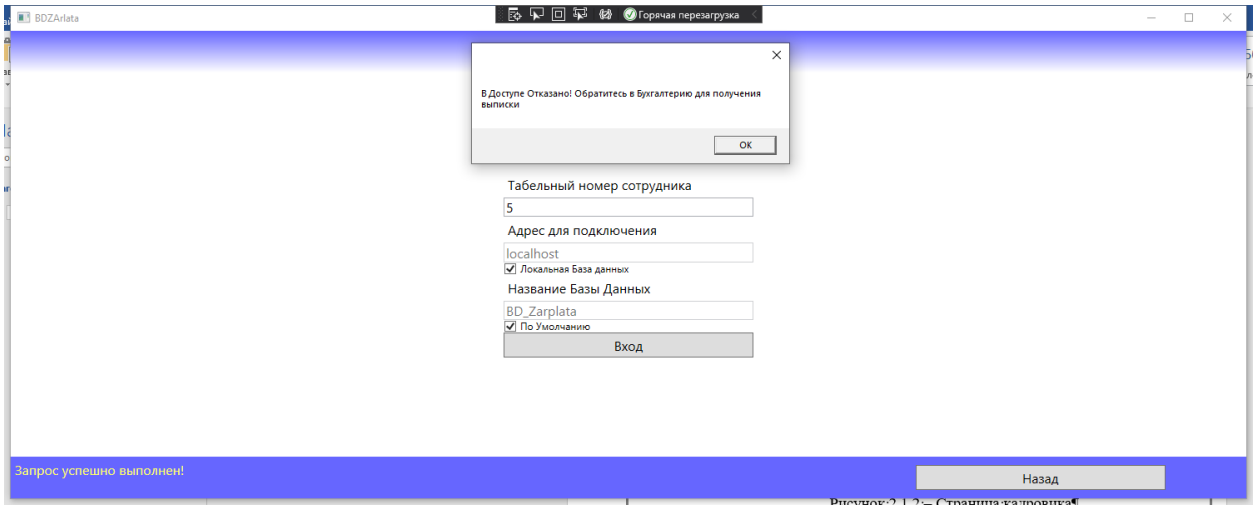


Рисунок 2.2.5– Сообщение об отказанном доступе

Тестирование входа при вводе некорректных данных (рисунки 2.2.6–2.2.10)

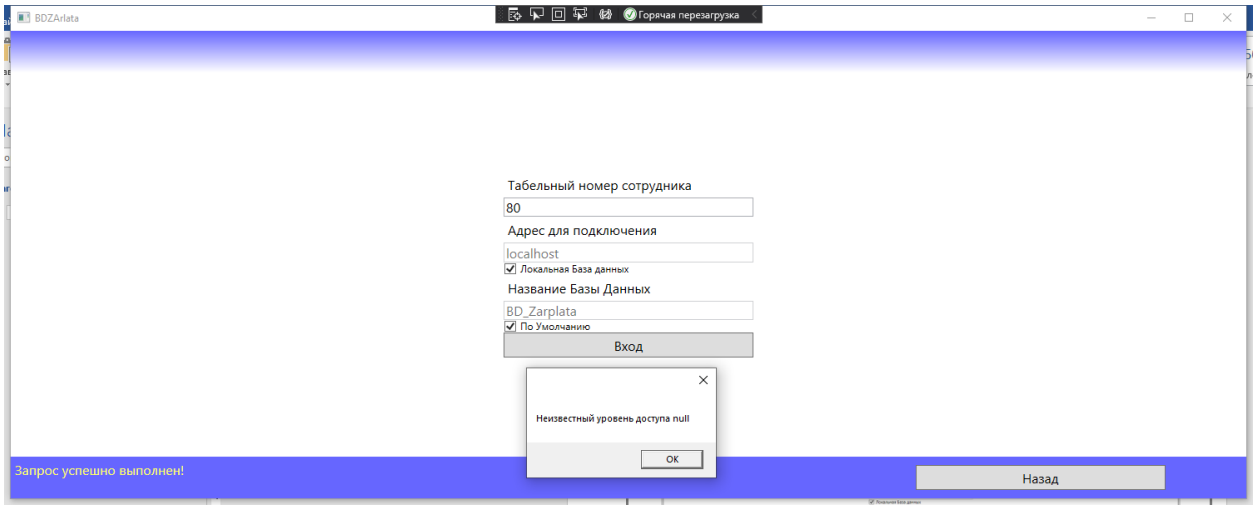


Рисунок 2.2.6– Сообщение об ошибке

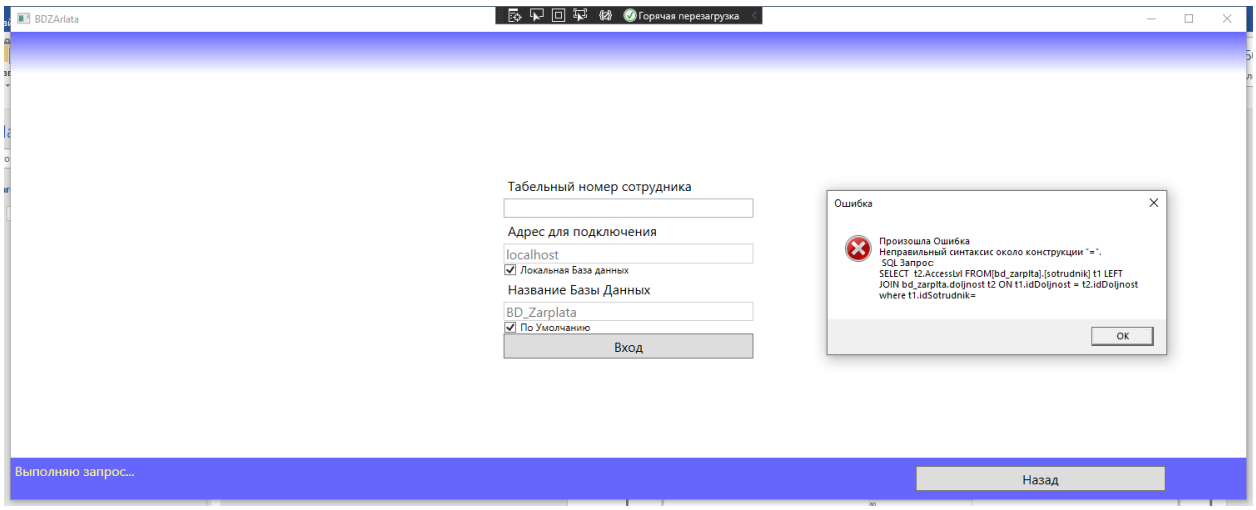


Рисунок 2.2.7– Сообщение об ошибке

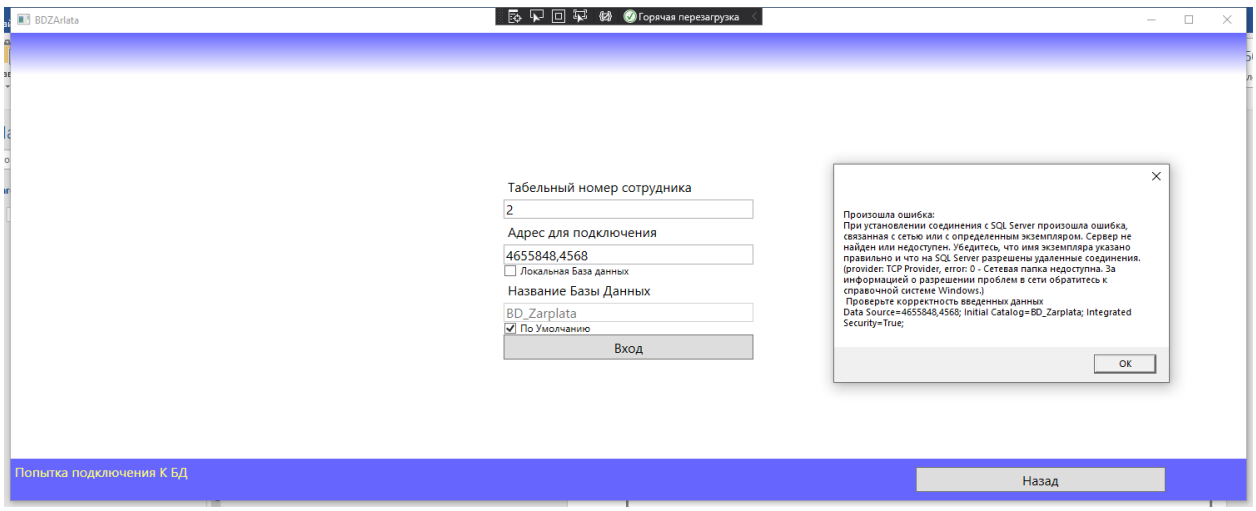


Рисунок 2.2.8– Сообщение об ошибке

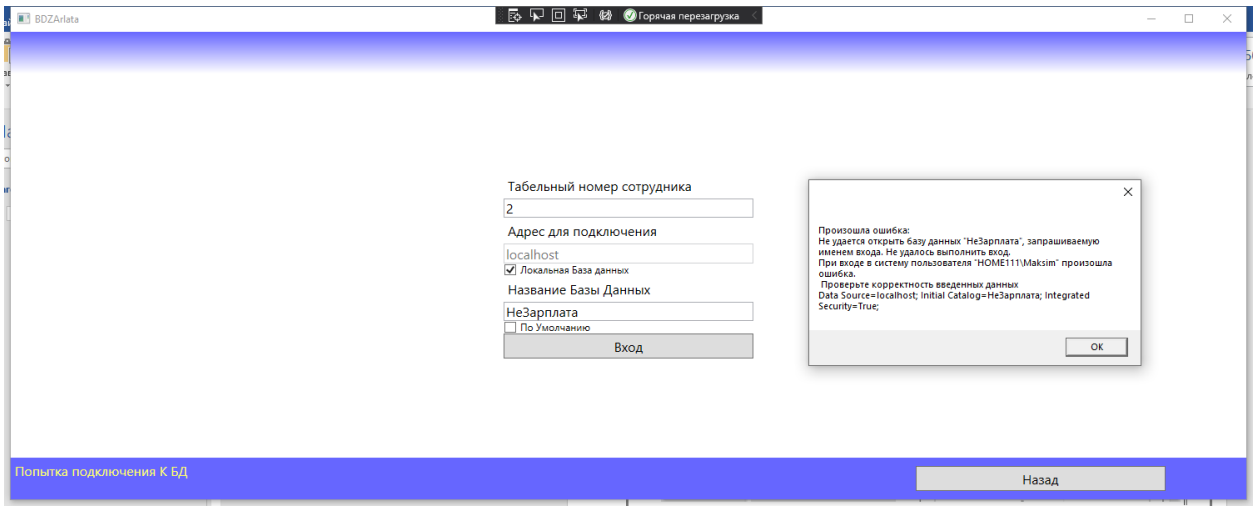


Рисунок 2.2.9– Сообщение об ошибке

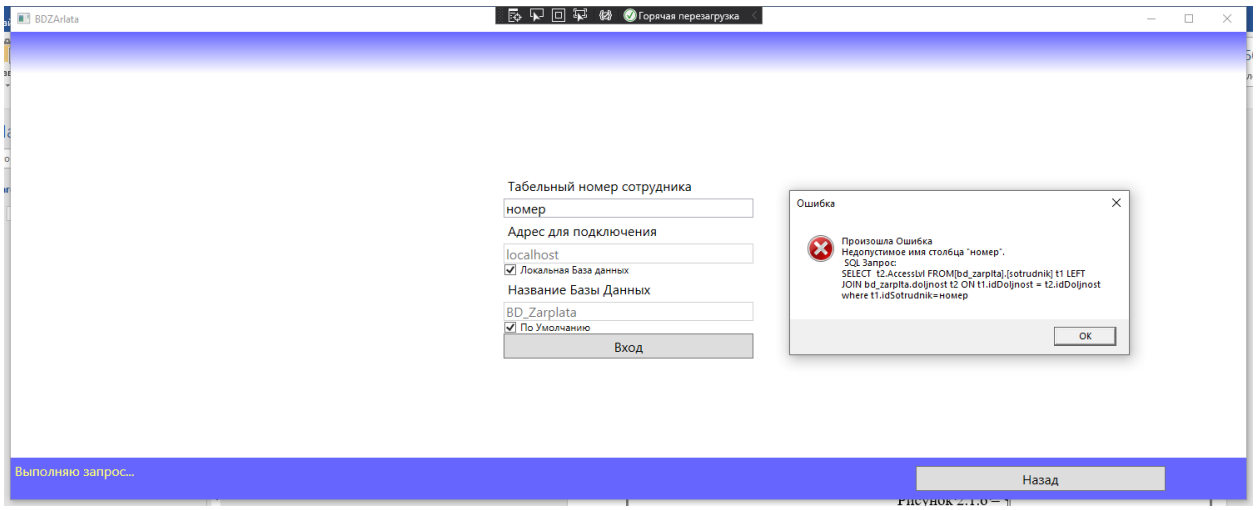


Рисунок 2.2.10– Сообщение об ошибке

Тестирование изменение оклада при вводе корректных данных (рисунки 2.2.11–2.2.12)

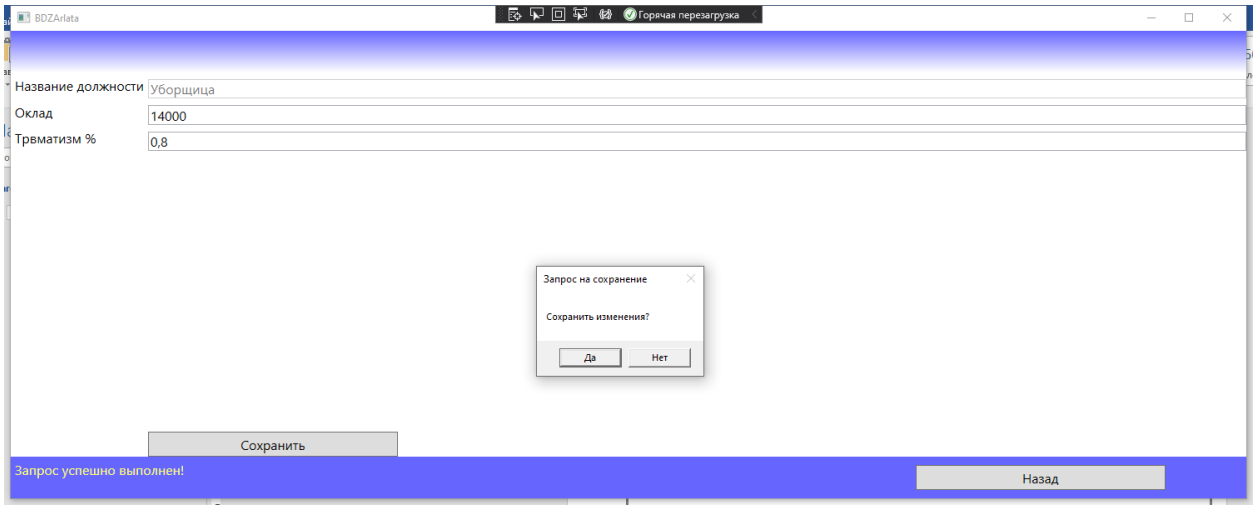


Рисунок 2.2.11– Запрос на сохранение

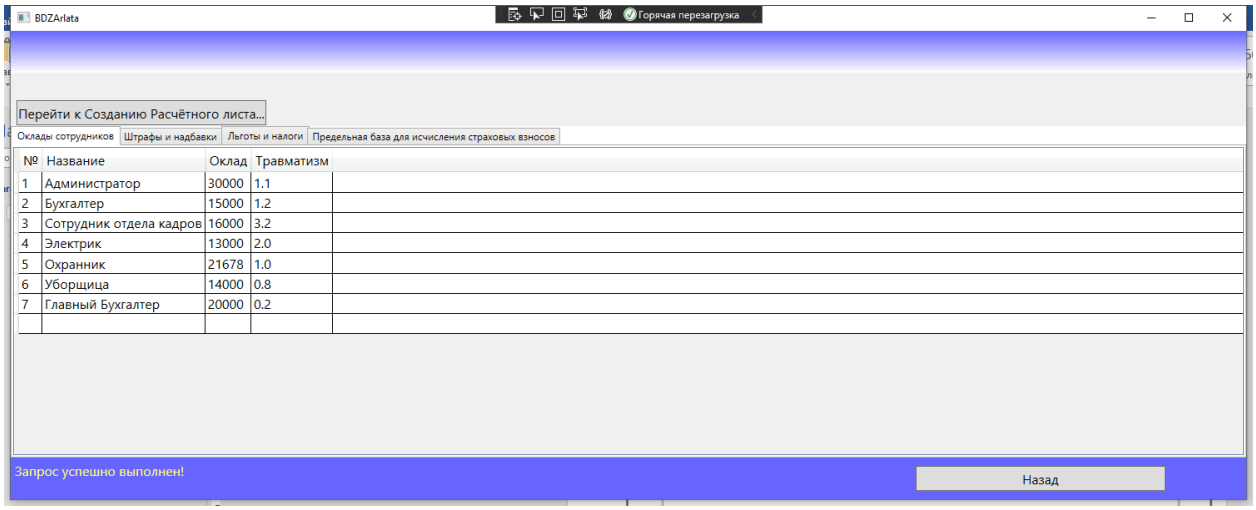


Рисунок 2.2.12– Изменение данных

Тестирование изменения оклада при вводе не корректных данных в поле «Оклад» (рисунок 2.2.13)

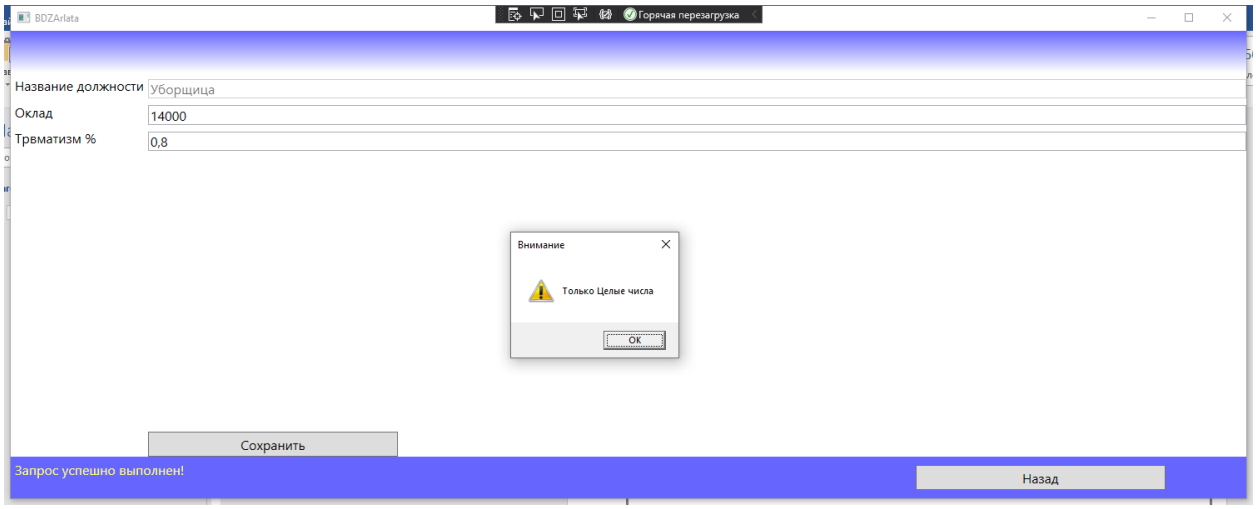


Рисунок 2.2.13– Сообщение об ошибке

Тестирование изменения оклада при вводе не корректных данных в поле «Травматизм%» (рисунок 2.2.14)

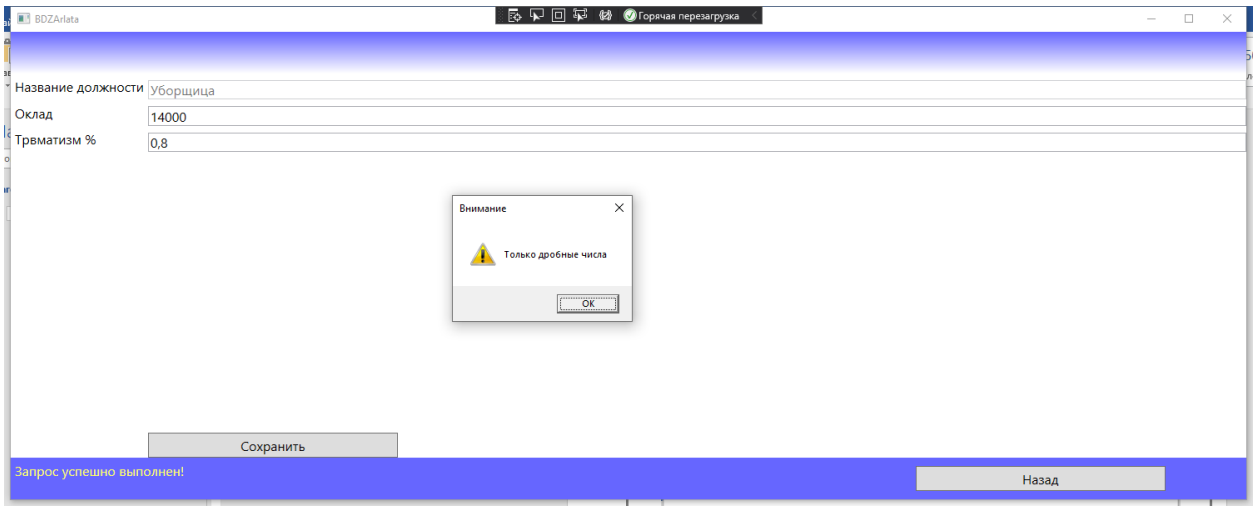


Рисунок 2.2.14– Сообщение об ошибке

Тестирование штрафов и надбавок при вводе корректных данных (рисунок 2.2.15)

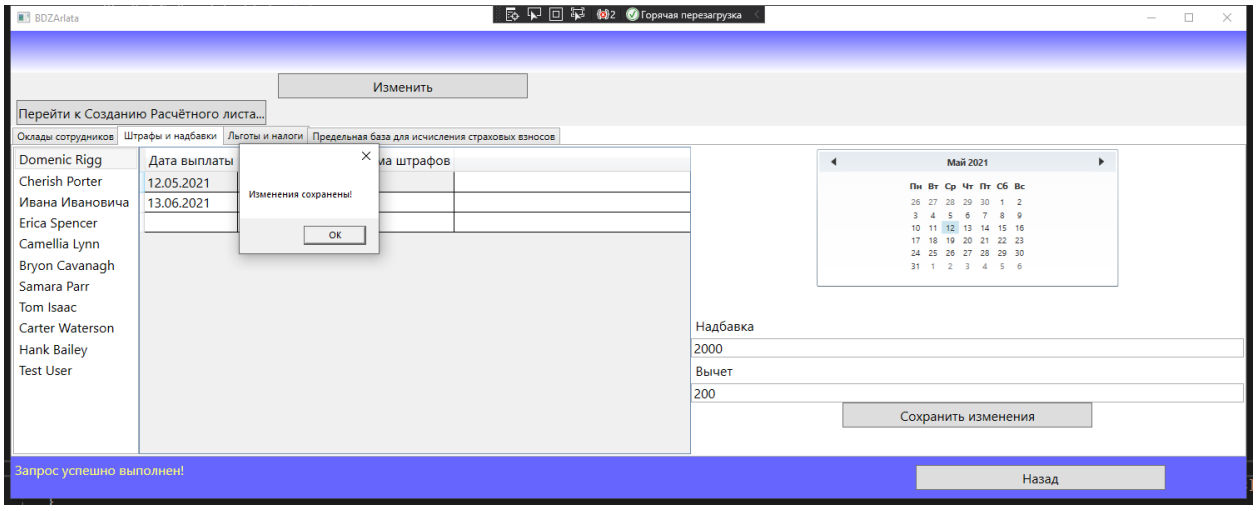


Рисунок 2.2.15– Сообщение об успешном сохранении

Тестирование штрафов и надбавок при вводе не корректных данных (рисунки 2.2.16–2.2.17)

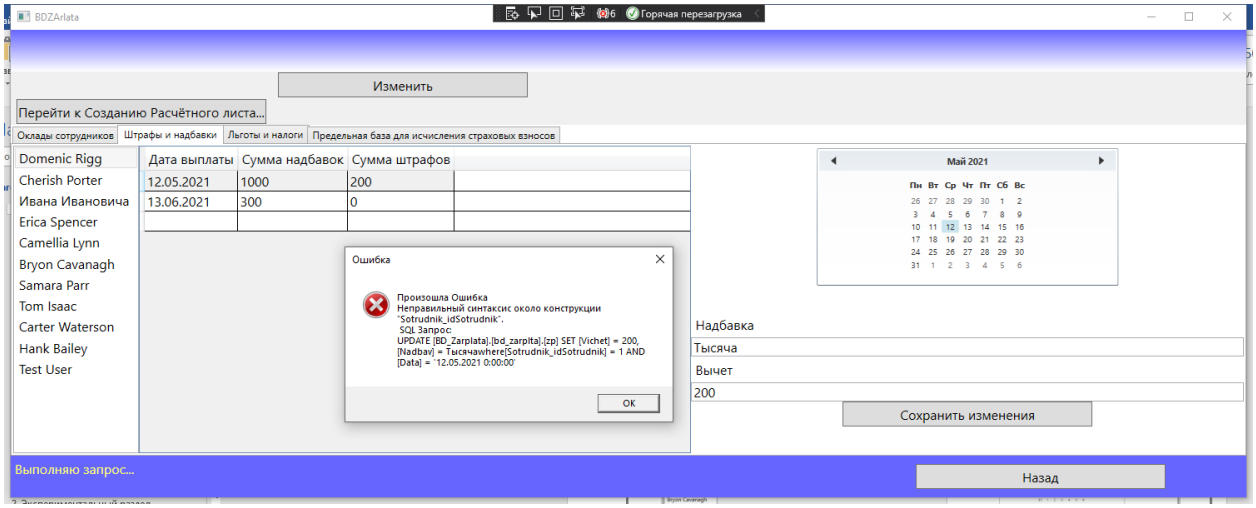


Рисунок 2.2.16 – Сообщение об ошибке

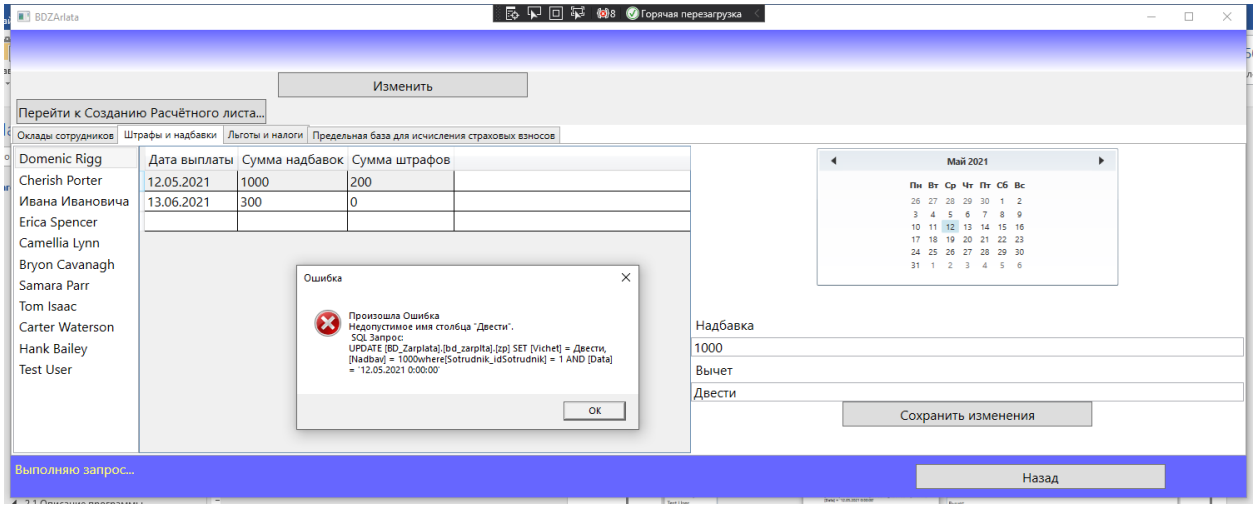


Рисунок 2.2.17 – Сообщение об ошибке

Тестирование льгот и налогов при вводе корректных данных (рисунок 2.2.18)

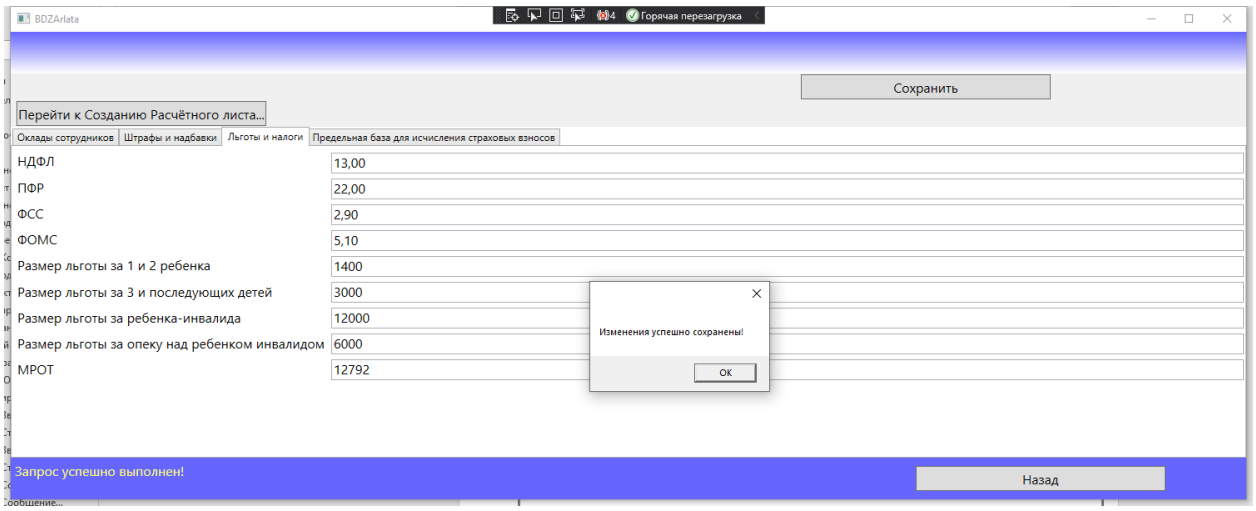


Рисунок 2.2.18 – Сообщение об успешном сохранении

Тестирование льгот и налогов при вводе некорректных данных в поля: «НДФЛ», «ПФР», «ФСС» , «ФОМС» (рисунок 2.2.19)

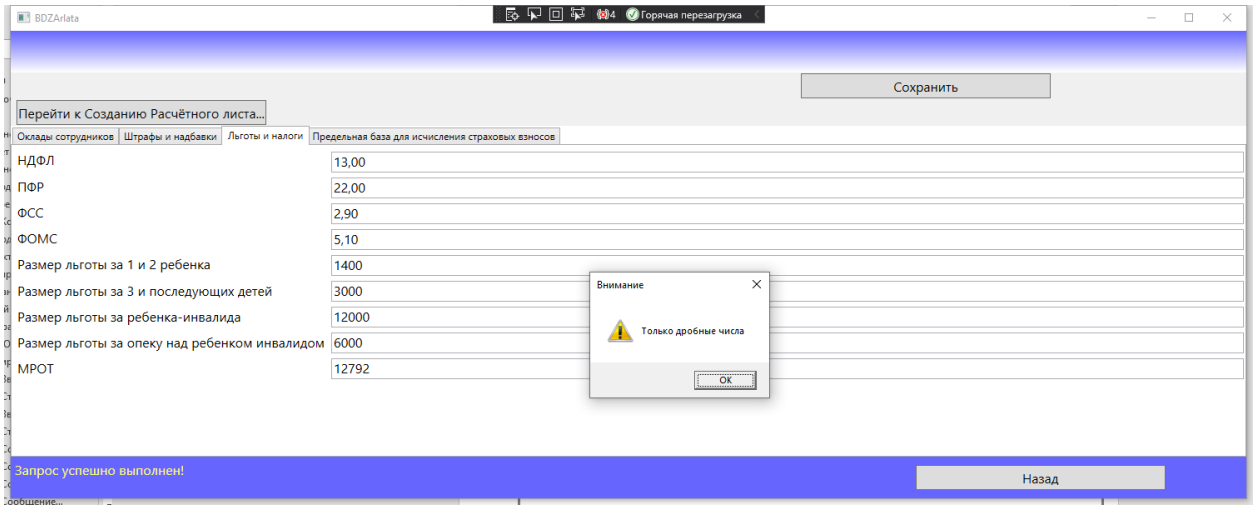


Рисунок 2.2.19 – Сообщение об ошибке

Тестирование льгот и налогов при вводе некорректных данных в поля льгот и «МРОТ» (рисунок 2.2.20)

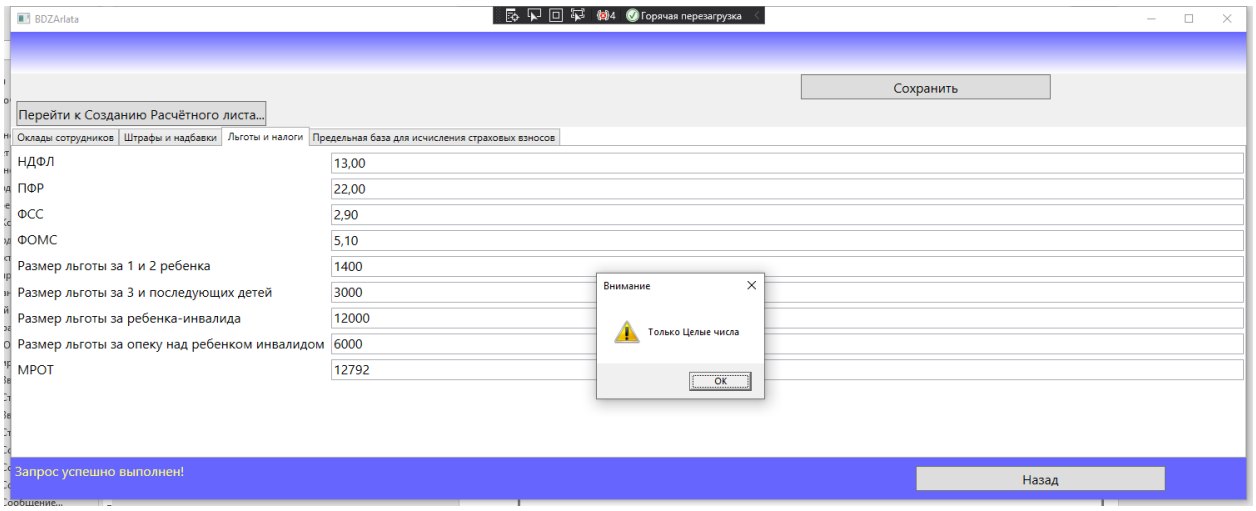


Рисунок 2.2.20 – Сообщение об ошибке

Тестирование добавления нового пользователя при заполнении только обязательных полей корректно (рисунок 2.2.21)

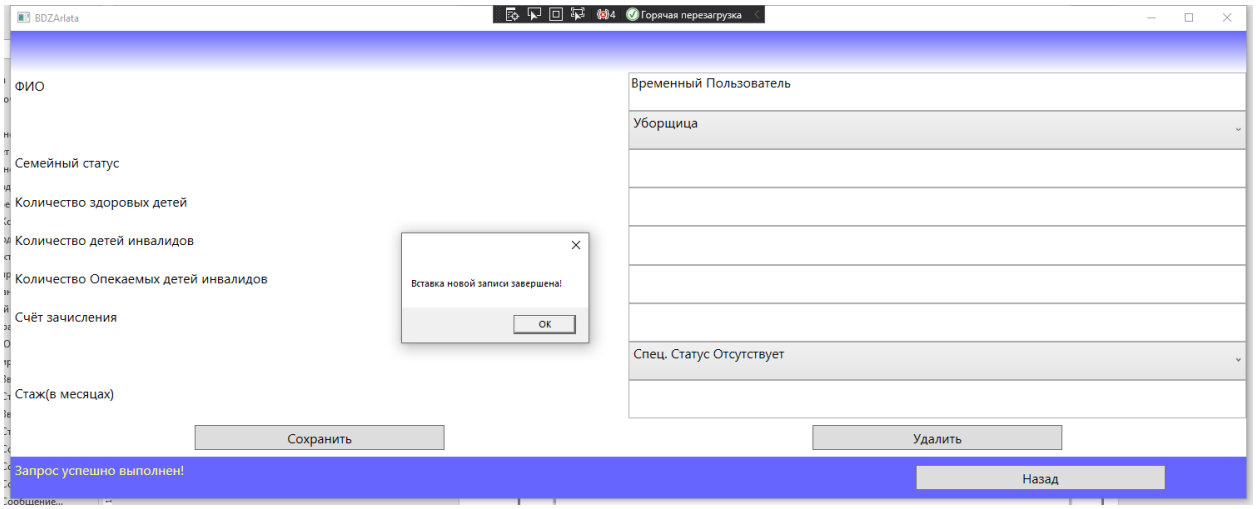


Рисунок 2.2.21– Успешное создание нового пользователя

Тестирование добавления или изменения нового пользователя при некорректном заполнении поля «ФИО» (рисунок 2.2.22)

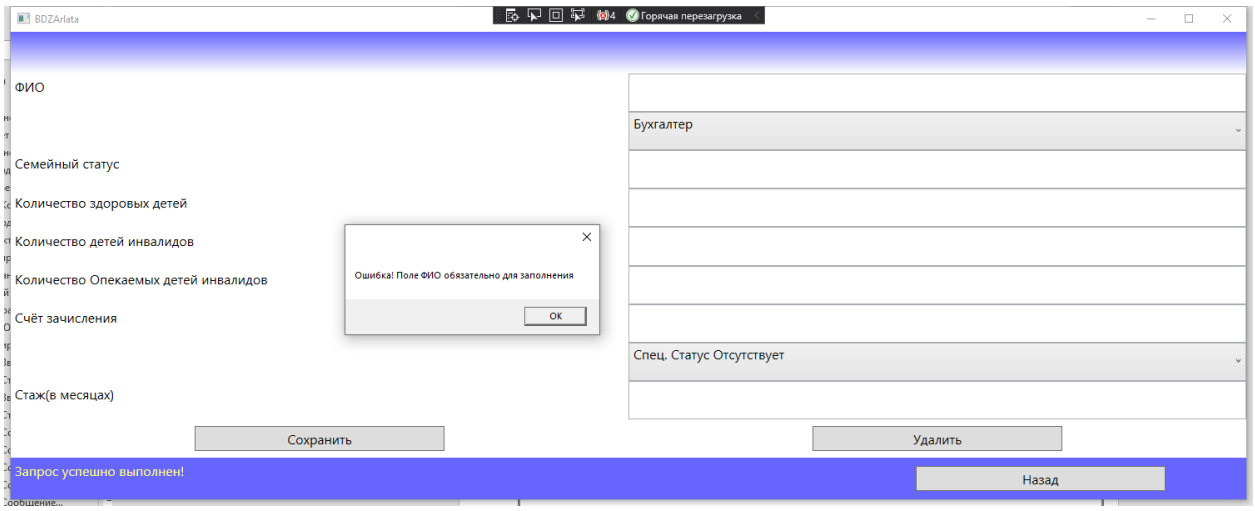


Рисунок 2.2.22 – Сообщение об ошибке

Тестирование добавления или изменения нового пользователя при некорректном заполнении числовых полей (рисунок 2.2.23)

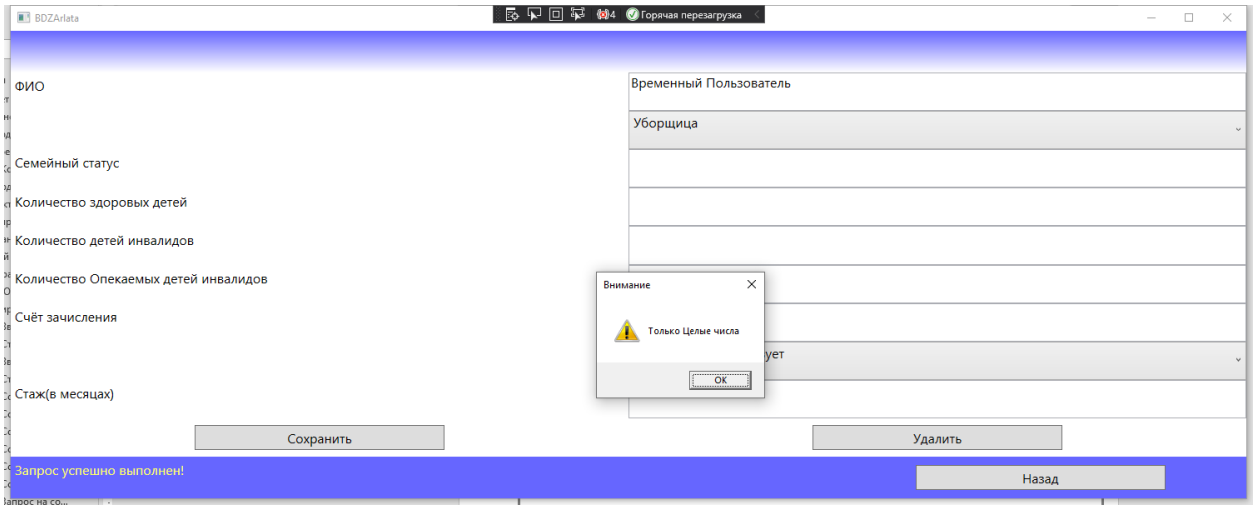


Рисунок 2.2.23 – Сообщение об ошибке

Тестирование удаления пользователя (рисунки 2.2.24–2.2.25)

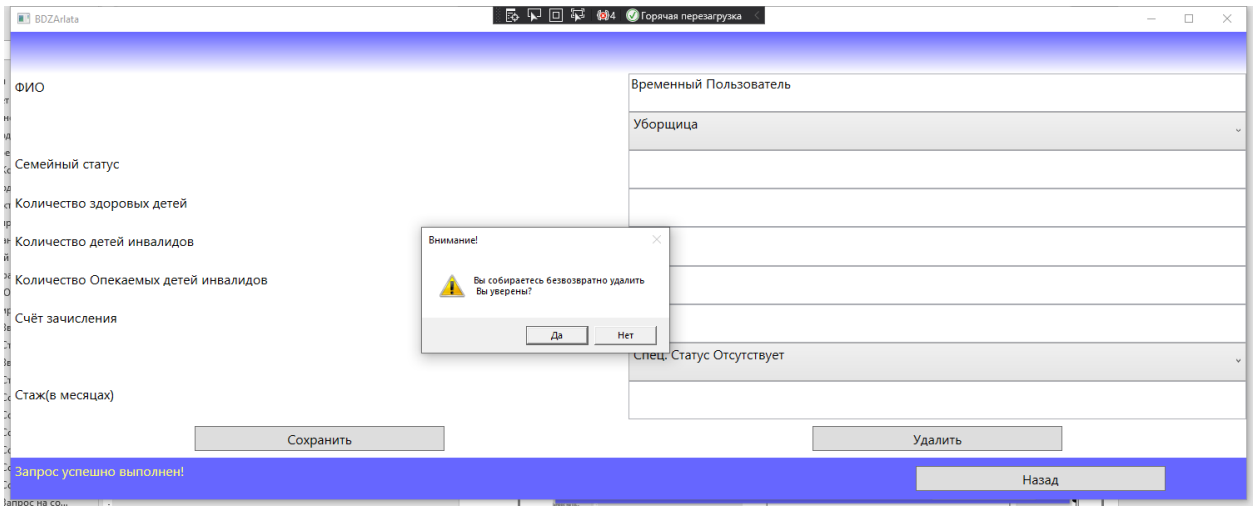


Рисунок 2.2.24– Диалоговое окно

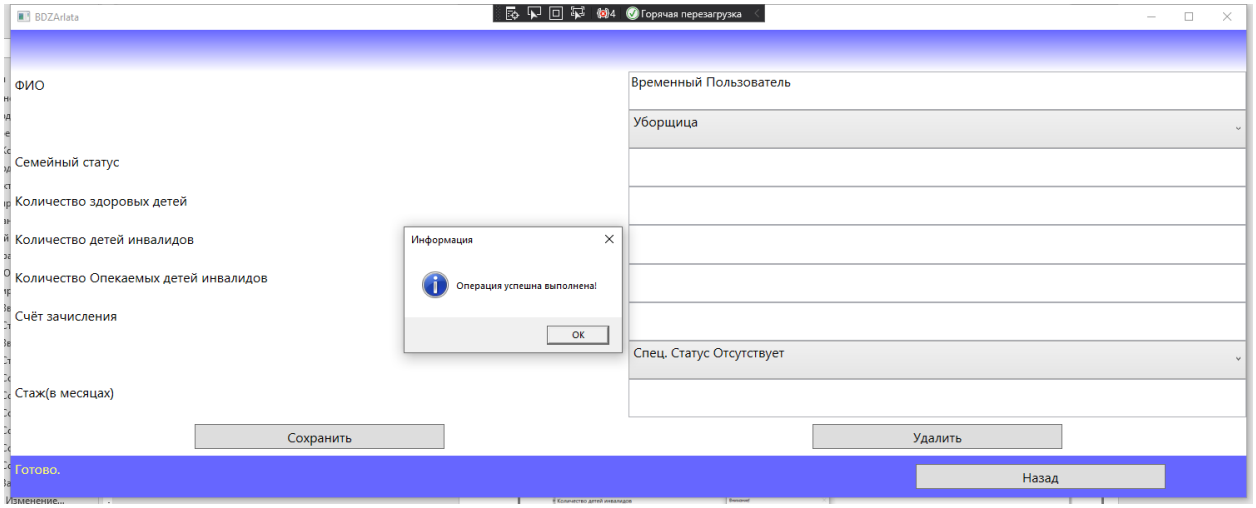


Рисунок 2.2.25– Сообщение об успешном удалении

Тестирование удаления ещё не созданного пользователя (рисунок 2.2.26)

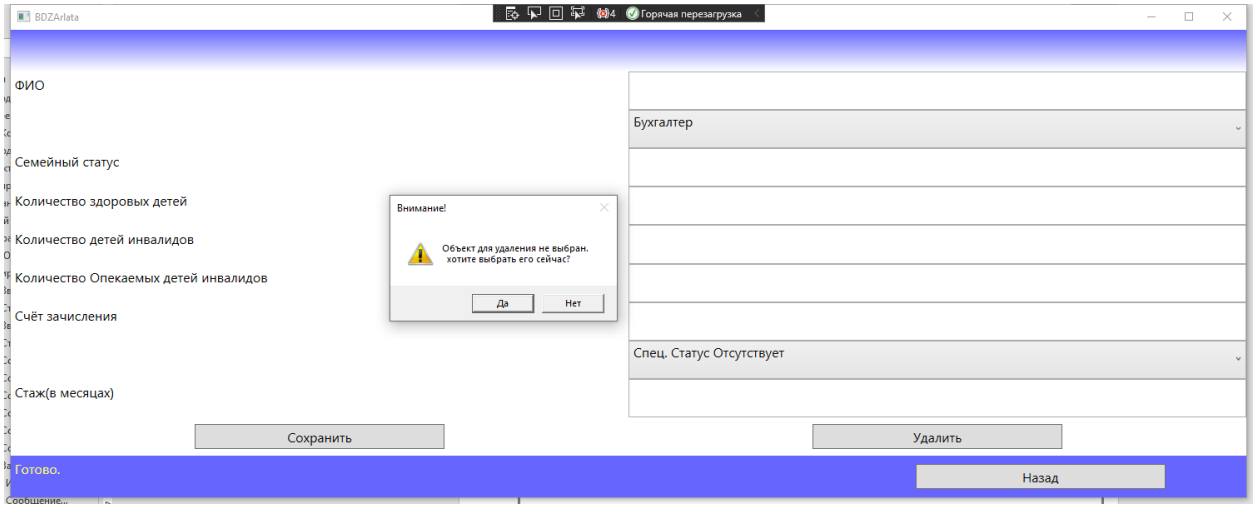


Рисунок 2.2.26– Сообщение о выборе не существующего объекта

2.3. Руководство пользователя

2.3.1. Назначение системы

Программа BDZarplata предназначена автоматизировать расчёт зарплаты, содержание работника и суммы налогов, упростить процесс выплаты зарплаты для упрощения работы Бухгалтера.

2.3.2. Условия применения системы

Программное обеспечение разрабатывается для персональной вычислительной техники со следующими характеристиками:

- Windows 10 64bit
- Процессор 2,3 ГГц (2 ядра, 4 потока) / Intel core i3-7020U
- Интегрированное графическое ядро Intel HD Graphics 620 или аналогичная дискретная видеокарта
- Оперативная память 1 ГБ
- 50 Мб свободного места на Твердотельном накопителе или Жестком диске

Программа «BDZarplata» предназначена для пользователей, имеющих как минимум первоначальные навыки работы с графической операционной системой, на которой будет запускаться данная программа, а также владеющих знаниями по бухгалтерскому учету.

2.3.3. Подготовка системы к работе

Для запуска программы необходимо запустить приложение BDZarplata.exe из каталога, в котором установлен данный программный продукт. После этого открывается окно авторизации для входа в программу (рисунок 2.3.3.1).

Что бы войти в программу необходимо заполнить поля «Табельный номер сотрудника», «Адрес для подключения» и «Название базы данных». Поле «Адрес для подключения» содержит адрес сервера с базой данных в локальной

сети (к примеру 127.0.0.1). что бы его активировать, необходимо снять галочку с флажка «локальная База данных». Название Базы данных содержит внутреннее название базы данных к которому хочет подключиться пользователь. Что бы его изменить необходимо убрать галочку с флажка «По умолчанию». После входа в программу, в зависимости от роли пользователя его будет ожидать страница для сотрудника отдела кадров (рисунок 2.3.4.1) или бухгалтерии

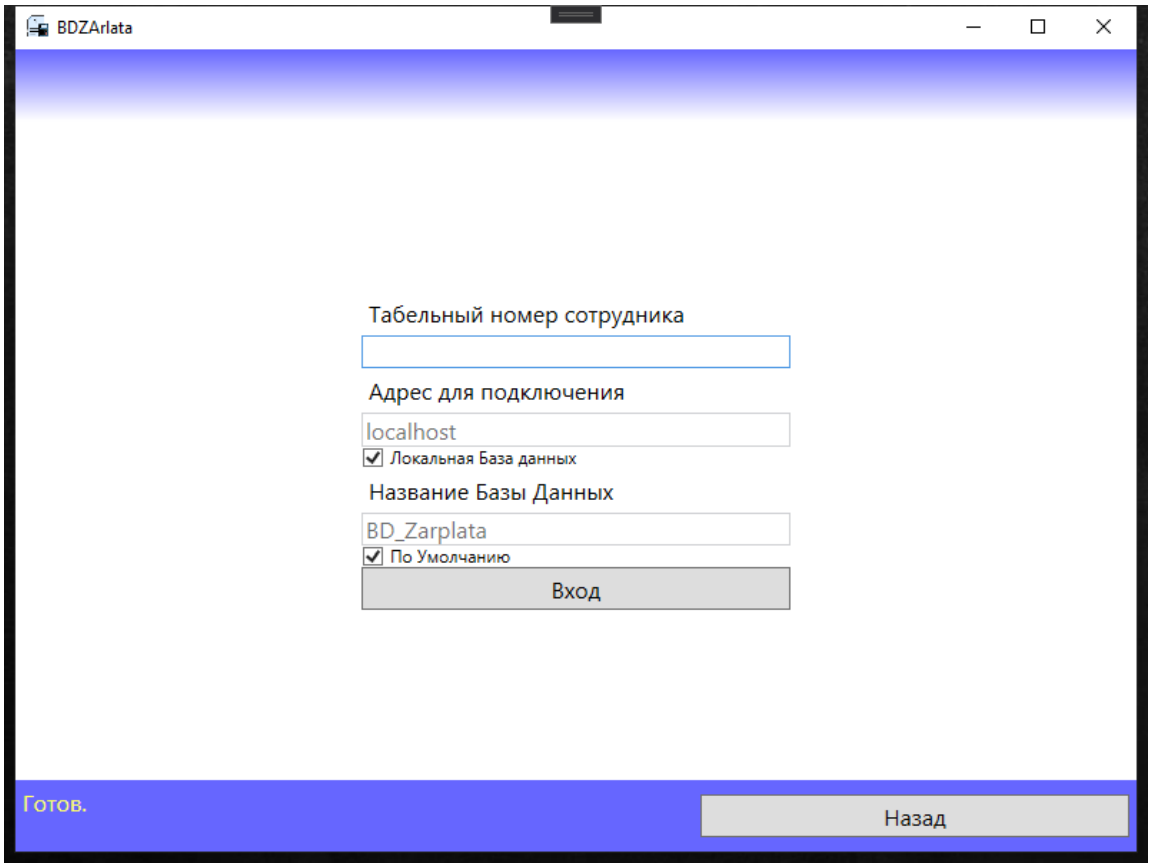


Рисунок 2.3.3.1– Окно авторизации

2.3.4. Отдел кадров

После входа, сотрудник отдела кадров видит перед собой страницу с 2 вкладками: «Основные данные сотрудников» и «Расписание». (рисунок 2.3.4.1)

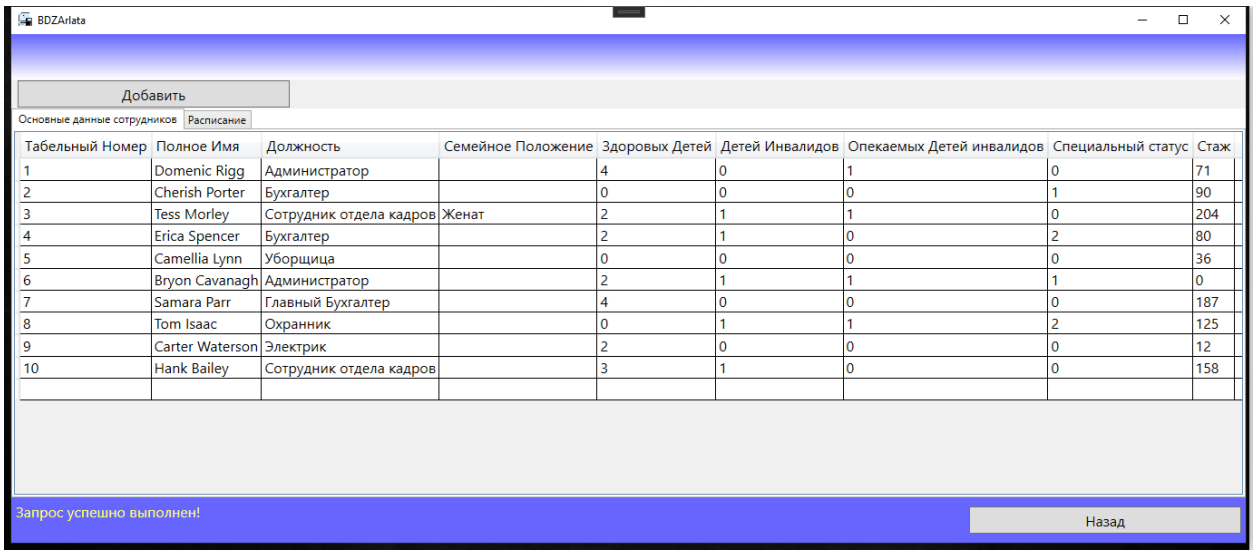


Рисунок 2.3.4.1– Главная страница отдела кадров

Для добавления нового сотрудника необходимо нажать кнопку «Добавить» находясь на вкладке «Основные данные сотрудников»

После чего пользователя перенесет на страницу для заполнения анкеты сотрудника (рисунок 2.3.4.2)

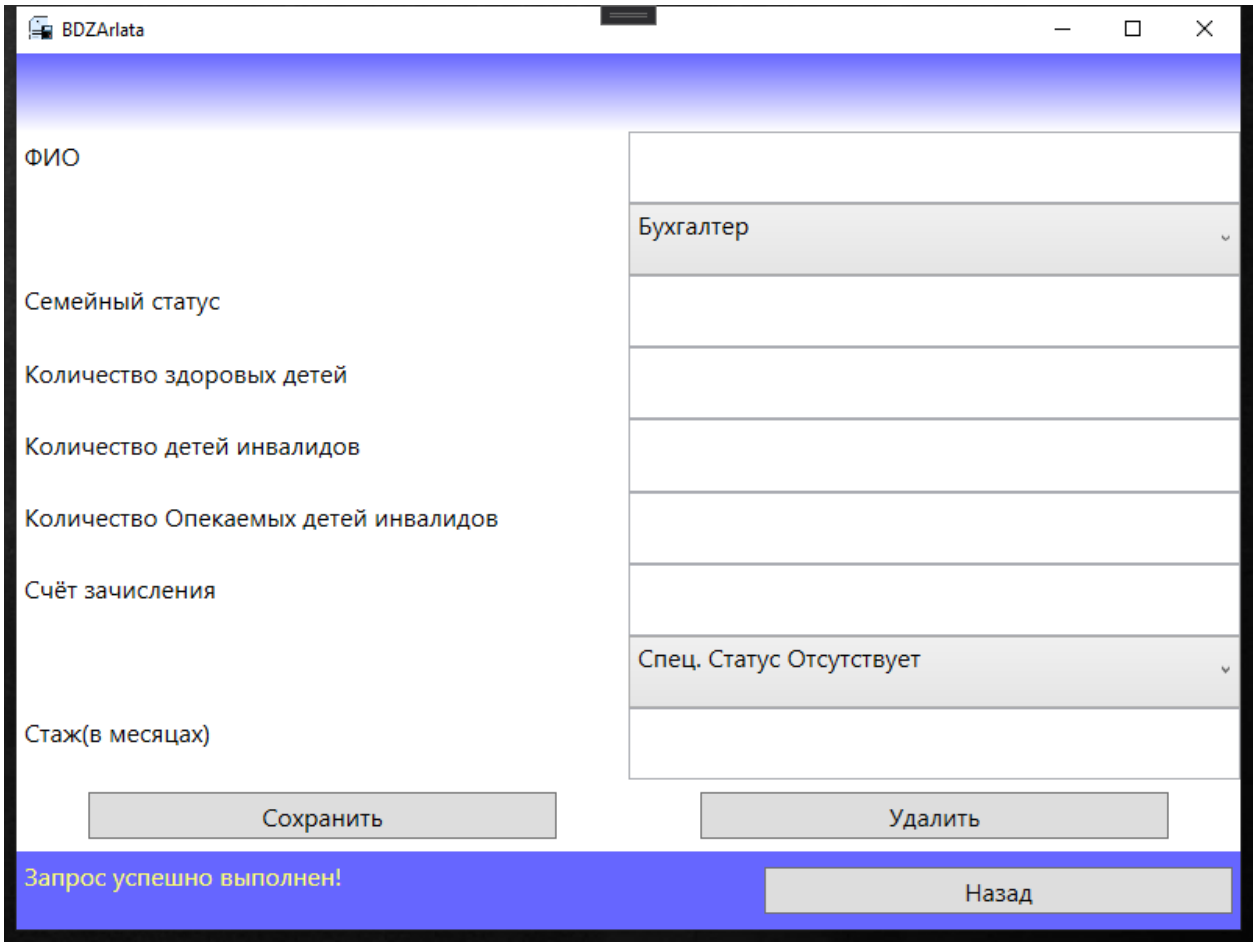


Рисунок 2.3.4.2– Пустая анкета для заполнения

После заполнения всех обязательных данных пользователь может нажать кнопку «Сохранить» для занесения данных в БД. После чего программа сообщит о успешности вставки записи (рисунок 2.3.4.3).

После чего программа вернётся на предыдущую страницу автоматически заполнив недостающие поля анкеты (рисунок 2.3.4.3).

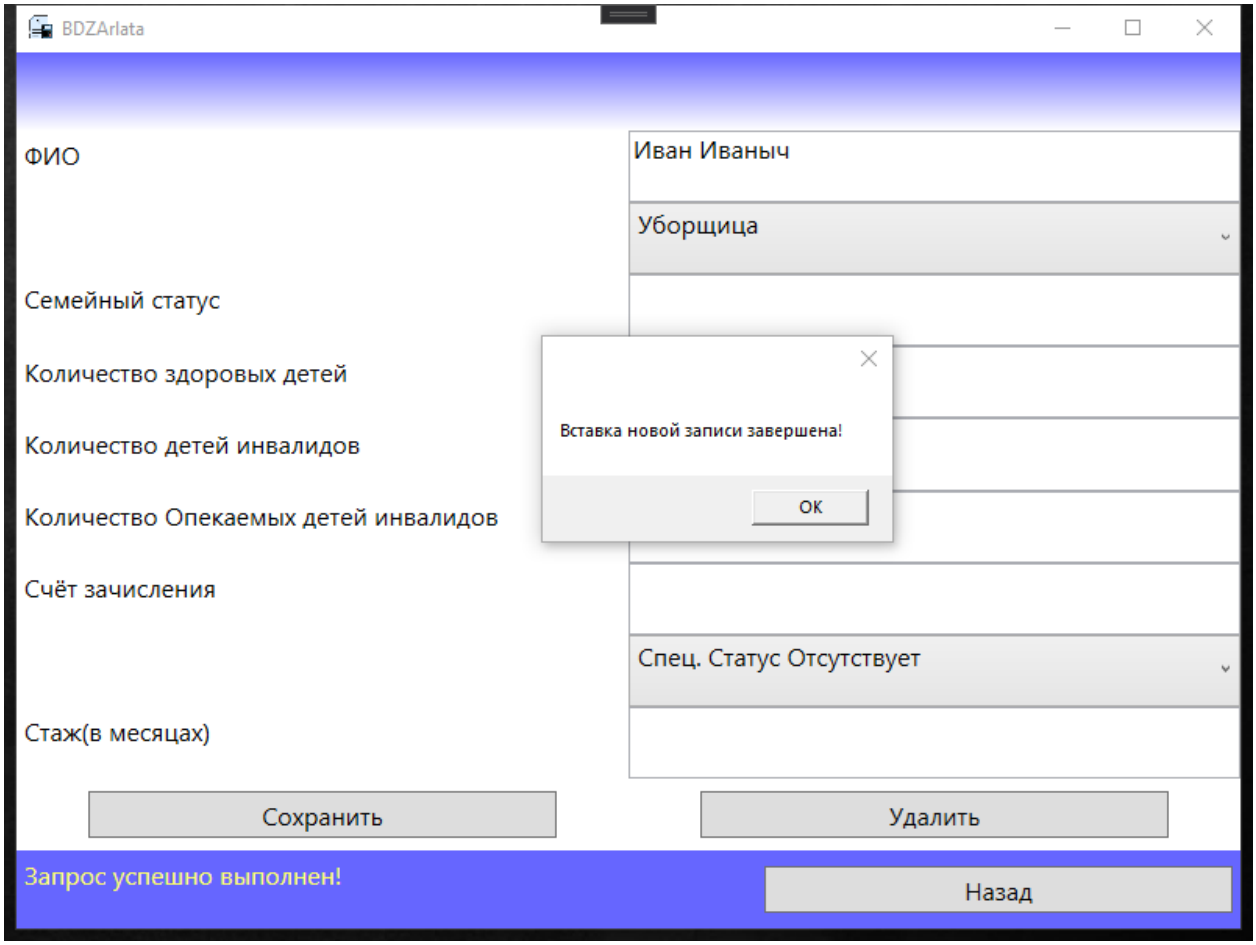
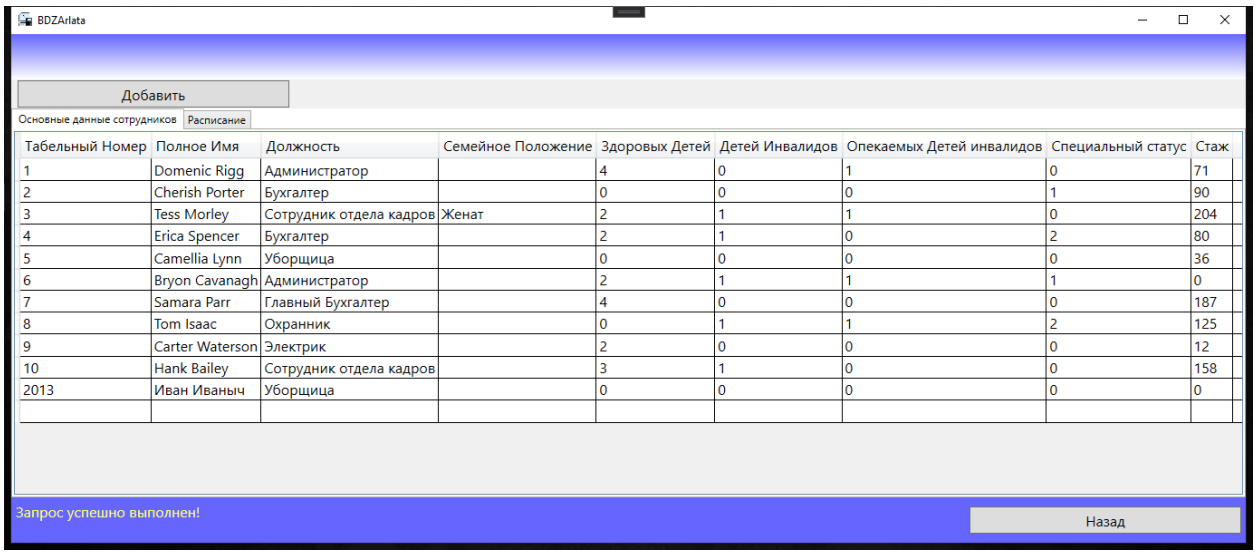


Рисунок 2.3.4.3– Заполненная анкета и сообщение об успешной вставке



Табельный Номер	Полное Имя	Должность	Семейное Положение	Здоровых Детей	Детей Инвалидов	Опекаемых Детей инвалидов	Специальный статус	Стаж
1	Domenic Rigg	Администратор		4	0	1	0	71
2	Cherish Porter	Бухгалтер		0	0	0	1	90
3	Tess Morley	Сотрудник отдела кадров	Женат	2	1	1	0	204
4	Erica Spencer	Бухгалтер		2	1	0	2	80
5	Camellia Lynn	Уборщица		0	0	0	0	36
6	Bryon Cavanagh	Администратор		2	1	1	1	0
7	Samara Parr	Главный Бухгалтер		4	0	0	0	187
8	Tom Isaac	Охранник		0	1	1	2	125
9	Carter Waterson	Электрик		2	0	0	0	12
10	Hank Bailey	Сотрудник отдела кадров		3	1	0	0	158
2013	Иван Иваныч	Уборщица		0	0	0	0	0

Рисунок 2.3.4.4– Новая запись в таблице

Для изменения записи необходимо 2 раза кликнуть левой кнопкой мыши на запись в таблице, которую хочет изменить пользователь, после чего откроется окно для редактирования анкеты (рисунок 2.3.4.5)

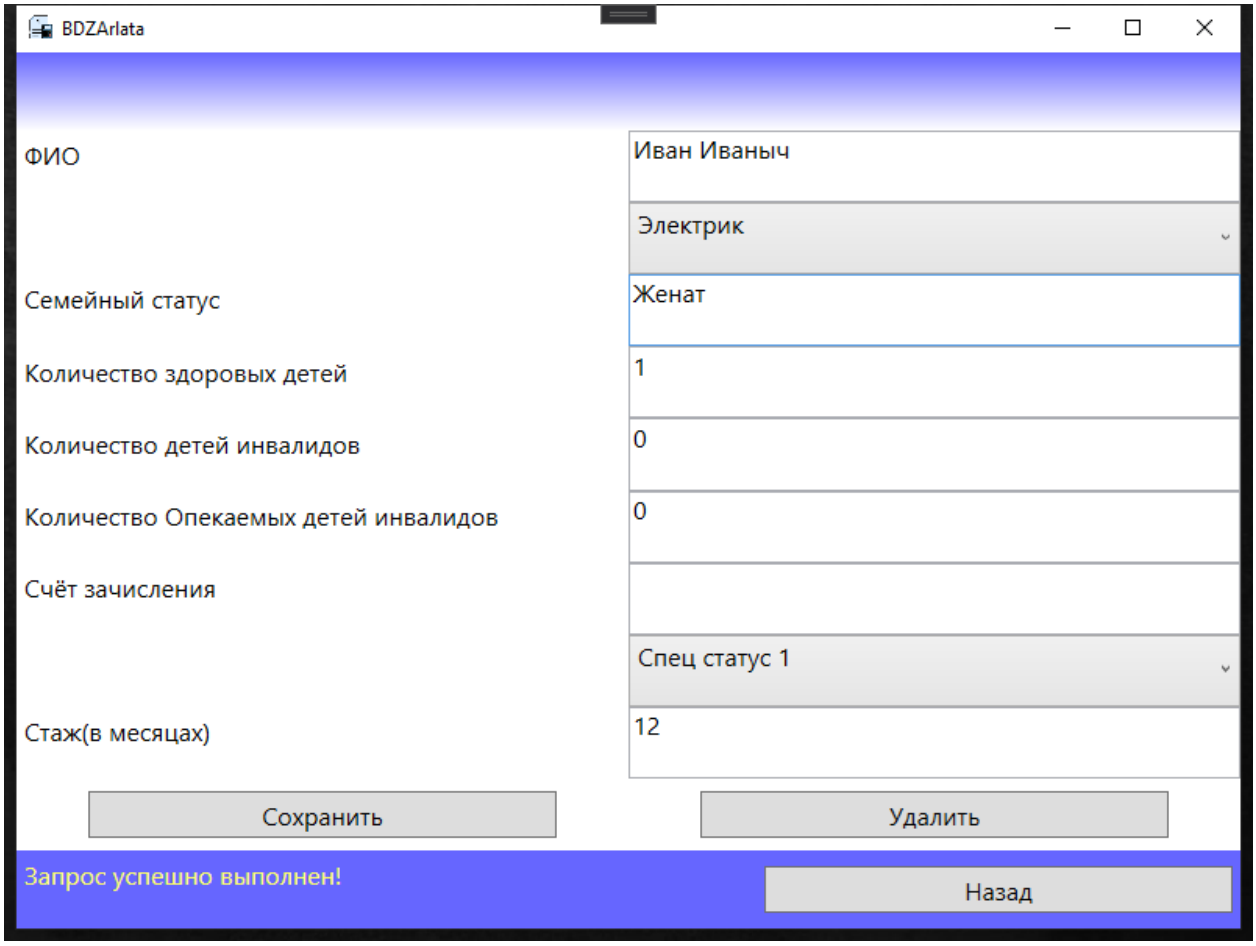
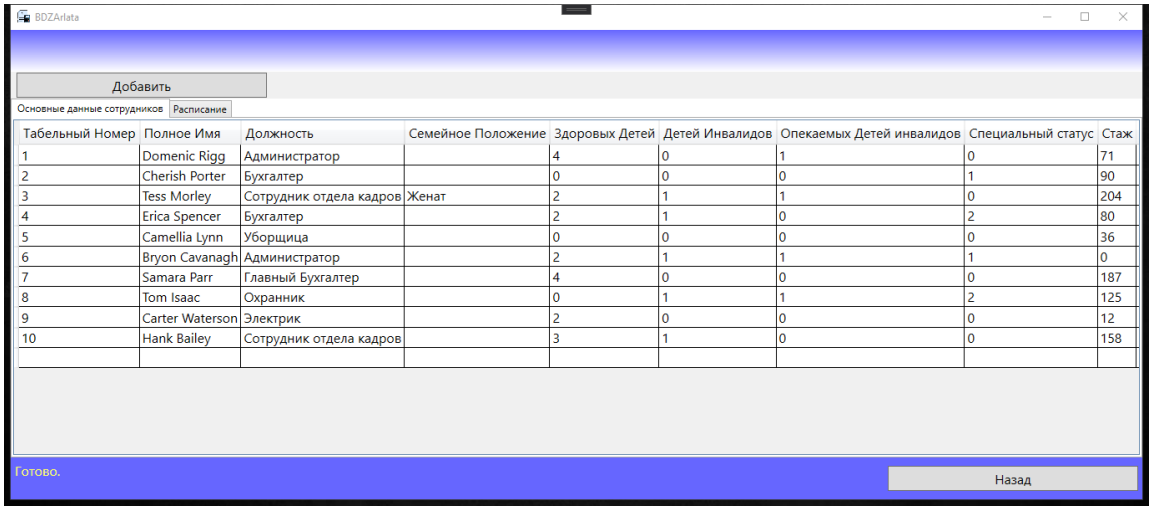


Рисунок 2.3.4.5– Окно редактирования анкеты

После редактирования пользователь может нажать кнопку «Сохранить» для сохранения изменённой записи, «Удалить» для полного удаления записи из БД или «Назад» для возврата к предыдущей странице без сохранения изменений.



Табельный Номер	Полное Имя	Должность	Семейное Положение	Здоровых Детей	Детей Инвалидов	Опекаемых Детей инвалидов	Специальный статус	Стаж
1	Domenic Rigg	Администратор		4	0	1	0	71
2	Cherish Porter	Бухгалтер		0	0	0	1	90
3	Tess Morley	Сотрудник отдела кадров	Женат	2	1	1	0	204
4	Erica Spencer	Бухгалтер		2	1	0	2	80
5	Camellia Lynn	Уборщица		0	0	0	0	36
6	Bryon Cavanagh	Администратор		2	1	1	1	0
7	Samara Parr	Главный Бухгалтер		4	0	0	0	187
8	Tom Isaac	Охранник		0	1	1	2	125
9	Carter Waterson	Электрик		2	0	0	0	12
10	Hank Bailey	Сотрудник отдела кадров		3	1	0	0	158

Рисунок 2.3.4.6– Изменение и удаление записи

Для для просмотра и изменения расписания сотрудника необходимо перейти на вкладку «Расписание», выбрать сотрудника, в левом списке и выбрать дату для изменения в центральной таблице (рисунок 2.3.4.7)

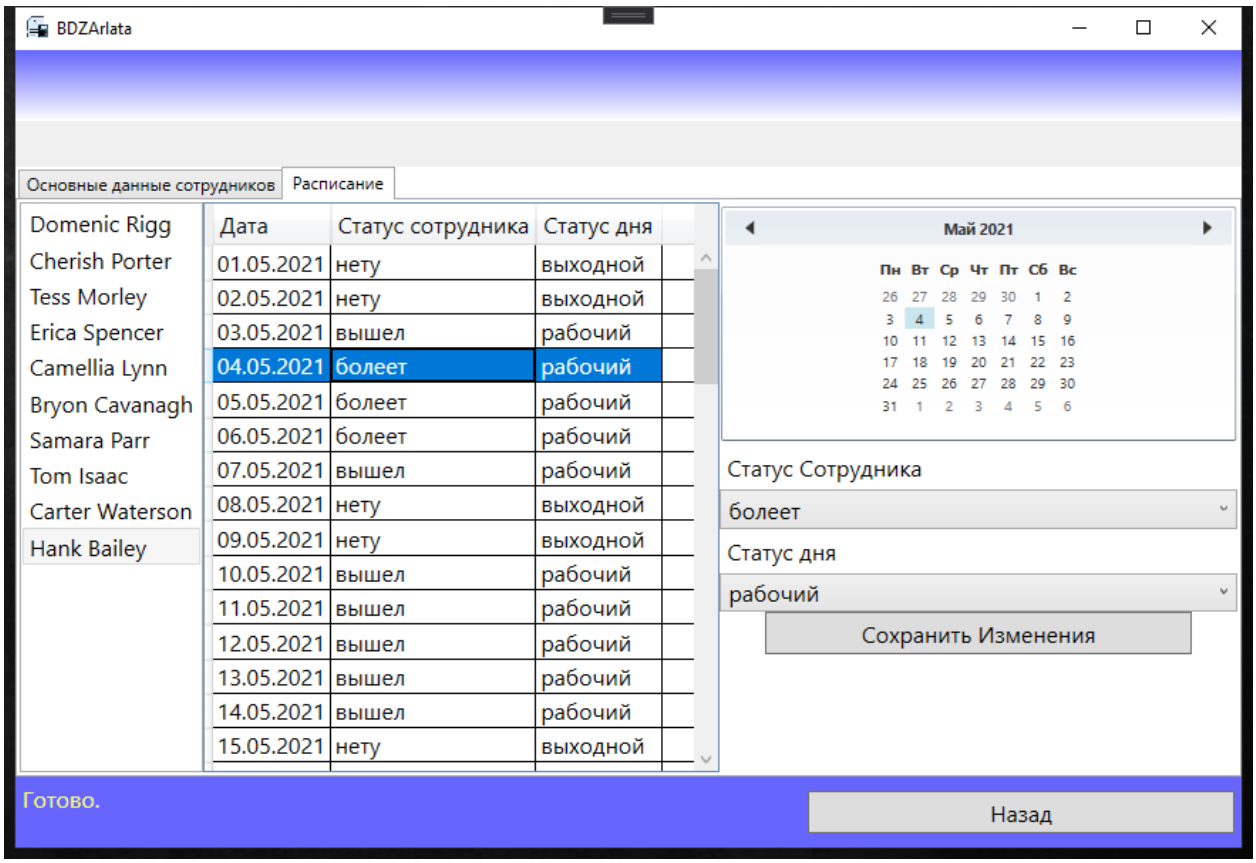


Рисунок 2.3.4.7– просмотр расписания сотрудника

После чего необходимо выбрать в раскрывающихся списках «статус сотрудника», «статус дня» значения и нажать кнопку «Сохранить изменения» (Рисунок 2.3.4.8)

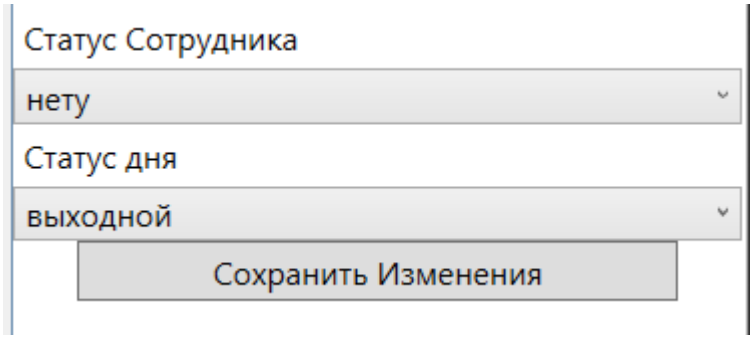


Рисунок 2.3.4.8– Списки и кнопка

ЗАКЛЮЧЕНИЕ

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

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 для базового создания списка работников, их расписания, а также полного расчёта заработной платы.

WPF-приложение написано на языке C# с использованием среды разработки Visual Studio 2019 с использованием языка разметки XAML и системы управления базой данных Microsoft SQL Server

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.

С помощью приложения на основании данных контрольного примера были получены результаты, которые полностью совпадают с выходной информацией контрольного примера.

Приложение А.

Расчетный листок за 13.05.2021

ООО

Работник: Cherish Porter

Табельный номер: 2 **Должность: Бухгалтер**

Вид	*Период*	*Дни*	*Часы*	*Сумма*	*Вид*	*Период*	*Сумма*
1. Начислено					*2.Удержано*		
Оклад					НДФЛ по ставке 13%		
Дополнительные выплаты					Иные удержания		
Больничные пособия							
Всего начислено:					Всего удержано:		
3. Взносы в ПФР					Сумма к выплате		
Страховые взносы в ПФР (страховая часть 22%)					Зачислено на счёт№		
					Выдано наличными		

Рисунок А – Шаблон выходного документа

Приложение Б.

Входные данные контрольного примера

Таблица Б.1.1 - Справочник сотрудников

Табельный номер сотрудника	ФИО сотрудника	ID Должности	Семейный статус сотрудника	Количество здоровых детей сотрудника	Количество детей инвалидов	Число опекаемых детей инвалидов	Счёт зачисления сотрудника	Наличие или отсутствие одного из спец статусов	Страховой стаж
1	2	3	4	5	6	7	8	9	10
1	Domenic Rigg	1		4	0	1		0	71
2	Cherish Porter	2		0	0	0		1	90
3	Tess Morley	3		2	1	1		0	204
4	Erica Spencer	2		2	1	0		2	80
5	Camellia Lynn	6		0	0	0		0	36
6	Bryon Cavanagh	1		2	1	1		1	0
7	Samara Parr	7		4	0	0		0	187
8	Tom Isaac	5		0	1	1		2	125
9	Carter Waterson	4		2	0	0		0	12
10	Hank Bailey	3		3	1	0		0	158

Таблица Б.1.2 - Справочник Финансовых данных

НДФЛ	ПФР	ФСС	ФОМС	Размер льготы за 1 и 2 ребёнка	Размер льготы за 3 и последующих детей	Размер льготы за ребёнка инвалида	Размер льготы за опеку над ребёнком инвалидом	МРОТ
1	2	3	4	5	6	7	8	9
13.00	22.00	2.90	5.10	1400	3000	12000	6000	12792

Таблица Б.1.3 - Справочник Налоговой базы

Год	Значение базы
1	2
2020	1292000
2021	1465000

Таблица Б.1.4 - Справочник Выплат Сотрудникам

Дата выплаты	Табельный номер сотрудника	Сумма всех надбавок	Сумма всех Вычетов
1	2	3	4
12.05.2021	1	1000	100
13.05.2021	2	0	100
10.05.2021	3	500	400
11.05.2021	4	400	200
12.05.2021	5	600	0
11.05.2021	6	0	200
10.05.2021	7	0	0
13.05.2021	8	500	400
12.05.2021	9	0	0
11.05.2021	10	700	0
13.06.2021	1	300	100
10.06.2021	2	500	100
11.06.2021	3	1000	0
11.06.2021	4	0	200
12.06.2021	5	0	300

Продолжение таблицы Б.1.4

1	2	3	4
13.06.2021	6	0	0
10.06.2021	7	500	300
11.06.2021	8	400	200
12.06.2021	9	800	200
10.05.2021	10	1000	0

Таблица Б.1.5 Справочник Должностей

ID Должности	Название должности	Оклад	Процент травматизма	Уровень доступа к данным
1	2	3	4	5
1	Администратор	30000	1.1	3
2	Бухгалтер	15000	1.2	1
3	Сотрудник отдела кадров	16000	3.2	2
4	Электрик	13000	2.0	0
5	Охранник	21678	1.0	0
6	Уборщица	13222	0.8	0
7	Главный бухгалтер	20000	0.2	1

Приложение В.

Выходные данные контрольного примера

Расчетный листок за 12.05.2021							
ООО							
Работник: Domenic Rigg							
Табельный номер: 1					**Должность: Администратор**		
Вид	*Период*	*Дни*	*Часы*	*Сумма*	*Вид*	*Период*	*Сумма*
1. Начислено					*2.Удержано*		
Оклад				30000	НДФЛ по ставке 13%		1522,3
Дополнительные выплаты				1000	Иные удержания		200
Больничные пособия							
Всего начислено:				31000	Всего удержано:		1722,3
3. Взносы в ПФР					Сумма к выплате		24987,7
Страховые взносы в ПФР (страховая часть 22%)				5832,2	Зачислено на счёт№		
					Выдано наличными		24987,7

Рисунок В.1 – Выходные данные контрольного примера

Приложение Г.

Код программы

DB.cs

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Windows;
using System.Windows.Controls;
namespace BDZarplata.Classes
{
    class DB
    {
        /// <summary>
        /// загрузка данных из БД в DataGrid
        /// </summary>
        /// <param name="dataGrid"></param>
        /// <param name="sql">SQL команда для выполнения (типа SELECT) </param>
        public static void loadDataGrid(DataGrid dataGrid, string sql)
        {
            DB_Connect.OpenConnection();
            SqlDataAdapter mysqldataAdapter = new SqlDataAdapter(sql, DB_Connect.connectionString);
            DataTable dataTable = new DataTable();
            mysqldataAdapter.Fill(dataTable);
            dataGrid.ItemsSource = dataTable.DefaultView;
            DB_Connect.CloseConnection();
        }

        /// <summary>
        /// Загрузка данных из БД в список
        /// </summary>
        /// <param name="comboBox">Список куда будет загружены данные из БД</param>
        /// <param name="sql">SQL команда по которой происходит выборка</param>
```

```
/// <param name="numberCol">Номер столбца из выборки (начиная с 0) который присваивается списку  </param>
public static void LoadDataComboBox(ComboBox comboBox, string sql, int numberCol)
{
   DB_Connect.OpenConnection();
   SqlCommand command = new SqlCommand(sql, DB_Connect.myConnection);
   SqlDataReader reader = command.ExecuteReader();
   while (reader.Read())
   { comboBox.Items.Add(reader.GetValue(numberCol).ToString()); }
   DB_Connect.CloseConnection();
}
/// <summary>
///  Загрузка данных из БД в список
/// </summary>
/// <param name="listbox">Список куда будет загружены данные из БД</param>
/// <param name="sql">SQL команда по которой происходит выборка</param>
/// <param name="numberCol">Номер столбца из выборки (начиная с 0) который присваивается списку  </param>
public static void LoadDataListBox(ListBox listbox, string sql, int numberCol)
{
   DB_Connect.OpenConnection();
   SqlCommand command = new SqlCommand(sql, DB_Connect.myConnection);
   SqlDataReader reader = command.ExecuteReader();
   while (reader.Read())
   { listbox.Items.Add(reader.GetValue(numberCol).ToString()); }
   DB_Connect.CloseConnection();
}
/// <summary>
/// Выполнение SQL Запроса к БД
/// </summary>
/// <param name="sql">SQL команда для выполнения</param>
/// <returns>Первый столбец первой строки набора результатов или пустая ссылка</returns>
public static object queryScalar(string sql)
{
   Manager.UpdateLabel("Выполняю запрос...");
   object sqlValue = null;
```

```
            DB_Connect.OpenConnection();
            try
            {
                SqlCommand command = new SqlCommand(sql, DB_Connect.myConnection);
                sqlValue = command.ExecuteScalar();
                Manager.UpdateLabel("Запрос успешно выполнен!");

            }
            catch (Exception ex) { MessageBox.Show("Произошла Ошибка при обработке SQL запроса \n" + ex.Message + "\n SQL Запрос: \n" + sql,
"Ошибка", MessageBoxButton.OK, MessageBoxImage.Error); }
            finally { DB_Connect.CloseConnection(); }
            if (sqlValue != null)
            {
                return sqlValue;
            }
            else
            {
                sqlValue = "null";
                return sqlValue;
            }
        }
        /// <summary>
        /// Выполнение запроса к БД С возвратом указанного столбца
        /// </summary>
        /// <param name="SQL_Comand">SQL команда для выполнения</param>
        /// <param name="Column"> Номер возвращаемого столбца</param>
        /// <returns>Указанный столбец первой строки набора результатов или пустая ссылка</returns>
        public static string queryScalar(string SQL_Comand, int Column)
        {
            try
            {
                Manager.UpdateLabel("Выполняю запрос...");
                DB_Connect.OpenConnection();
                SqlCommand command = new SqlCommand(SQL_Comand, DB_Connect.myConnection);
```

```
            SqlDataReader reader = command.ExecuteReader();
            reader.Read();
            string temp = reader[Column].ToString();

            Manager.UpdateLabel("Запрос успешно выполнен!");
            if (temp != null)
            {
                return temp;
            }
            else
            {
                temp = "null";
                return temp;
            }
        }
        catch (Exception ex) { MessageBox.Show("Произошла Ошибка \n" + ex.Message, "Ошибка", MessageBoxButton.OK,
MessageBoxImage.Error); return ""; }
        finally { DB_Connect.CloseConnection(); }
    }
    /// <summary>
    /// Выполнение запроса к БД
    /// </summary>
    /// <param name="SQL_Comand">SQL команда для выполнения</param>
    /// <param name="Column"> Название возвращаемого столбца</param>
    /// <returns>Указанный столбец первой строки набора результатов или пустая ссылка</returns>
    public static string queryScalar(string SQL_Comand, string Column)
    {
        try
        {
            Manager.UpdateLabel("Выполняю запрос...");
            DB_Connect.OpenConnection();
            SqlCommand command = new SqlCommand(SQL_Comand, DB_Connect.myConnection);
            SqlDataReader reader = command.ExecuteReader();
            reader.Read();
```

```
            string temp = reader[Column].ToString();

            Manager.UpdateLabel("Запрос успешно выполнен!");
            if (temp != null)
            {
                return temp;
            }
            else
            {
                temp = "";
                return temp;
            }
        }
        catch (Exception ex) { MessageBox.Show("Произошла Ошибка \n" + ex.Message, "Ошибка", MessageBoxButton.OK,
MessageBoxImage.Error); return ""; }
        finally { DB_Connect.CloseConnection(); }
    }
    /// <summary>
    /// Выполнение запроса к БД
    /// </summary>
    /// <param name="SQL_Comand">SQL команда для выполнения</param>
    /// <param name="Columns">Массив индексов возвращаемых столбцов (начиная с 0) </param>
    /// <returns> массив значений Указанных столбцов первой строки набора результатов или пустая ссылка</returns>
    public static string[] queryScalar(string SQL_Comand, int[] Columns)
    {
        string[] results = new string[Columns.Length];
        try
        {
            Manager.UpdateLabel("Выполняю запрос...");
            DB_Connect.OpenConnection();
            SqlCommand command = new SqlCommand(SQL_Comand, DB_Connect.myConnection);
            SqlDataReader reader = command.ExecuteReader();
            reader.Read();
```

```
            for (int i = 0; i < Columns.Length; i++)
            {
                results[i] = reader[Columns[i]].ToString();
            }


            Manager.UpdateLabel("Запрос успешно выполнен!");
            return results;
        }
        catch (Exception ex)
        {
            MessageBox.Show("Произошла Ошибка при чтении данных \n" + ex.Message, "Ошибка\n", MessageBoxButton.OK,
MessageBoxImage.Error);
            results = new[] { "" };
            return results;
        }
        finally { DB_Connect.CloseConnection(); }
    }
    /// <summary>
    /// Выполнение запроса к БД
    /// </summary>
    /// <param name="SQL_Comand">SQL команда для выполнения(Не рекомендуется использовать эту перегрузку если в запросе
содержаться столбцы с одинаковым именем)</param>
    /// <param name="Columns">Массив названий возвращаемых столбцов </param>
    /// <returns> массив значений Указанных столбцов первой строки набора результатов или пустая ссылка</returns>
    public static string[] queryScalar(string SQL_Comand, string[] Columns)
    {
        string[] results = new string[Columns.Length];
        try
        {
            Manager.UpdateLabel("Выполняю запрос...");
            DB_Connect.OpenConnection();
            SqlCommand command = new SqlCommand(SQL_Comand, DB_Connect.myConnection);
            SqlDataReader reader = command.ExecuteReader();
```

```
            reader.Read();

            for (int i = 0; i < Columns.Length; i++)
            {
                results[i] = reader[Columns[i]].ToString();
            }

            Manager.UpdateLabel("Запрос успешно выполнен!");
            return results;
        }
        catch (Exception ex)
        {
            MessageBox.Show("Произошла Ошибка при чтении данных \n" + ex.Message, "Ошибка\n", MessageBoxButton.OK,
MessageBoxImage.Error);
            results = new[] { "" };
            return results;
        }
        finally { DB_Connect.CloseConnection(); }
    }

    /// <summary>
    /// Запрос к БД с получением данных из таблицы (таблиц)
    /// </summary>
    /// <param name="SQL_Comand">SQL команда для выполнения(типа Select)</param>
    /// <param name="CountColumn">Количество столбцов для возврата</param>
    ///  <param name="CountString">Количество строк для возврата(возврат начинается с 1 строки возвращаемым запросом</param>
    /// <param name="Results">[c,s]Массив результатов где перебор с - столбца , s- строки</param>
    public static void ReturnTable(string SQL_Comand, out string[,] Results, int CountString = 1, int CountColumn = 1)
    {
        Results = new string[CountColumn, CountString];

        try
        {
            Manager.UpdateLabel("Выполняю запрос...");
```

```
            DB_Connect.OpenConnection();
            SqlCommand command = new SqlCommand(SQL_Comand, DB_Connect.myConnection);
            SqlDataReader reader = command.ExecuteReader();
            int s = 0;
            while (reader.Read() && s < CountString)
            {
                for (int c = 0; c < CountColumn; c++)
                {
                    Results[c, s] = reader[c].ToString();
                }
                s++;
            }
            Manager.UpdateLabel("Запрос успешно выполнен!");
        }
        catch (Exception ex)
        {
            MessageBox.Show("Произошла Ошибка при чтении таблицы \n" + ex.Message, "Ошибка\n", MessageBoxButton.OK,
MessageBoxImage.Error);
        }
        finally { DB_Connect.CloseConnection(); }
    }

    /// <summary>
    /// Выполнение внешнего SQl Файла или запроса с возвратом количества строк
    /// </summary>
    /// <param name="sql"></param>
    /// <returns> количество задействованных в инструкции строк или -1 если при обработке запроса происходит ошибка </returns>
    public static int queryData(string sql)
    {
        Manager.UpdateLabel("Выполняю запрос...");
        DB_Connect.OpenConnection();
        try
        {
            SqlCommand command = new SqlCommand(sql, DB_Connect.myConnection);
```

```
            return command.ExecuteNonQuery();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Произошла Ошибка \n" + ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);
            return -1;
        }
        finally { DB_Connect.CloseConnection(); }
    }
  }
}
```

DB_Connect.cs

```
using System;
using System.Data.SqlClient;
using System.Windows;
namespace BDZarplata.Classes
{
   public class DB_Connect
   {
      public static string connectionString = "Data Source=localhost; Initial Catalog=BD_Zarplata; Integrated Security=true;";
      public static SqlConnection myConnection = new SqlConnection(connectionString);
      /// <summary>
      /// создание соединения с БД
      /// </summary>
      public static void OpenConnection()
      {

         if (myConnection.State == System.Data.ConnectionState.Closed)
         {
            myConnection.Open();

         }

      }
```

```
/// <summary>
/// закрытие соединения с БД
/// </summary>
public static void CloseConnection()
{
    if (myConnection.State == System.Data.ConnectionState.Open)
    {
        myConnection.Close();
    }
}
/// <summary>
/// Проверка на возможность установки соединения с БД
/// </summary>
/// <param name="DataSource">Адрес компьютера на котором храниться БД</param>
/// <param name="InitialCatalog">Название БД</param>
/// <param name="IntegratedSecurity">проверку подлинности</param>
/// <returns></returns>
public static bool OpenClouseConnection(string DataSource = "localhost", string InitialCatalog = "BD_Zarplata", bool IntegratedSecurity = true)
{
    string connectStr = "Data Source=" + DataSource + "; Initial Catalog=" + InitialCatalog + "; Integrated Security=" + IntegratedSecurity + ";";
    try
    {
        SqlConnection userConnection = new SqlConnection(connectStr);

        if (userConnection.State == System.Data.ConnectionState.Closed)
        {
            userConnection.Open();

        }
        if (userConnection.State == System.Data.ConnectionState.Open)
        {
            userConnection.Close();
            myConnection = userConnection;
        }
```

```
            return true;
        }
        catch (Exception ex)
        {
            MessageBox.Show("Произошла ошибка:\n" + ex.Message + "\n Проверьте корректность введенных данных\n" + connectStr);
            return false;
        }

    }
  }
}
```

DG.cs

```
using System;
using System.Windows.Controls;

namespace BDZarplata.Classes
{
  class DG
  {
    /// <summary>
    /// Получение значения выбранной ячейки
    /// </summary>
    /// <param name="dataGrid1">DataGrid в котором находиться нужная ячейка</param>
    /// <param name="selectedColumn">Номер столбца (начиная с 0) в котором находиться ячейка.
    /// Присвоить -1 для автоматического определения столбца </param>
    /// <returns>значение ячейки или ""</returns>
    public static string GetSelectCell(DataGrid dataGrid1, int selectedColumn = -1)
    {
      try
      {
        string result = "";

        if (selectedColumn == -1) { selectedColumn = dataGrid1.CurrentCell.Column.DisplayIndex; }
        var selectedCell = dataGrid1.SelectedCells[selectedColumn];
```

```
            var cellContent = selectedCell.Column.GetCellContent(selectedCell.Item);
            if (cellContent is TextBlock)
            {
                result = (cellContent as TextBlock).Text;
            }
            return result;
        }
        catch (Exception ex) { Manager.UpdateLabel("Ошибка в returnCell, возможно не выбрана ячейка для возврата \n" + ex.Message); return "";
}
    }


  }
}
```

Manager.cs

```
using System.Windows.Controls;
namespace BDZarplata.Classes
{
  class Manager
  {
    public static Frame MainFrame { get; set; }
    public static Label LabelStatus { get; set; }
    public static ProgressBar MainProgressBar { get; set; }
    /// <summary>
    /// отладочный вывод сообщений в отдельной Label
    /// </summary>
    /// <param name="newContent">новое сообщение для вывода</param>
    public static void UpdateLabel(string newContent = "")
    {
      LabelStatus.Content = newContent;
    }
  }
}
```

Procedure.cs

```
using System.Collections.Generic;
using System.Windows;

namespace BDZarplata.Classes
{
    class Procedure
    {
        /// <summary>
        /// Обновление значений указанной таблицы в БД
        /// </summary>
        /// <param name="NameTable">Имя таблицы</param>
        /// <param name="ColumnsName">Массив имен столбцов</param>
        /// <param name="ColumnsData">Массив значений этих столбцов</param>
        /// <param name="Where">Условие отбора</param>
        public static void UpdateTable(string NameTable, List<string> ColumnsName, List<string> ColumnsData, string Where)
        {
            string sqlComand = $"UPDATE {NameTable} SET {ColumnsName[0]} = {ColumnsData[0]}";
            if (ColumnsData.Count == ColumnsName.Count)
            {
                for (int i = 1; i < ColumnsData.Count; i++)
                {
                    sqlComand += $", {ColumnsName[i]} = {ColumnsData[i]}";
                }
                sqlComand += Where;
                Classes.DB.queryScalar(sqlComand);
            }
            else
            {
                MessageBox.Show("err UpdateTable \n Количество столбцов не совпадает с количеством значений");
            }
        }
        /// <summary>
        /// Добавление значений указанной таблицы в БД
        /// </summary>
```

```
        /// <param name="NameTable">Имя таблицы</param>
        /// <param name="ColumnsName">Массив имен столбцов</param>
        /// <param name="ColumnsData">Массив значений этих столбцов</param>
        public static void InsertTable(string NameTable, List<string> ColumnsName, List<string> ColumnsData)
        {

            string sqlComand = $"INSERT INTO {NameTable} ( {ColumnsName[0]}";
            if (ColumnsData.Count == ColumnsName.Count)
            {
                for (int i = 1; i < ColumnsName.Count; i++)
                {
                    sqlComand += $", {ColumnsName[i]}";
                }
                sqlComand += $") VALUES( {ColumnsData[0]} ";
                for (int i = 1; i < ColumnsData.Count; i++)
                {
                    sqlComand += $", {ColumnsData[i]}";
                }
                sqlComand += ")";
                DB.queryScalar(sqlComand);

            }
            else
            {
                MessageBox.Show("err InsertTable \n Количество столбцов не совпадает с количеством значений");
            }
        }

    }
}
```

MainWindow.xaml

```
<Window x:Class="BDZarplata.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:BDZarplata"
    mc:Ignorable="d"
    Title="BDZArlata" Height="600" Width="1500" MinWidth="800" MinHeight="600" FontSize="16">
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="1*"/>
        <ColumnDefinition Width="1*"/>
        <ColumnDefinition Width="310"/>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="50"/>
        <RowDefinition/>
        <RowDefinition Height="50"/>
    </Grid.RowDefinitions>
    <!--создание рамок-->
    <Rectangle Grid.ColumnSpan="3">
        <Rectangle.Fill>
            <LinearGradientBrush EndPoint="0.5,1" MappingMode="RelativeToBoundingBox" StartPoint="0.5,0">
                <GradientStop Color="White" Offset="1"/>
                <GradientStop Color="#FF6666FF"/>
            </LinearGradientBrush>
        </Rectangle.Fill>
    </Rectangle>
    <Rectangle Grid.Row="2" Grid.ColumnSpan="3" Fill="#FF6666FF"/>
    <!--страницы приложения-->
    <Frame
                        Name="MainFrame"
        Grid.Row="1"
        Grid.ColumnSpan="3"
                            NavigationUIVisibility="Hidden"
                    ></Frame>
```

```
        <!--вспомогательные информационные элементы-->
        <Label x:Name="LabelStatus1" Grid.Row="2" Foreground="#FFFFFF66" Content="Готов." />
        <ProgressBar x:Name="ProgressBarStatus"  Grid.Row="2" Grid.Column="0" Margin="0.4,34.6,-0.4,0" Visibility="Hidden"/>
        <Button x:Name="BtnBack" Grid.Row="2" Grid.Column="2" Click="BtnBack_Click">Назад</Button>
    </Grid>
</Window>
```

Page_SotrudnikMainInfo.xaml

```
<Page x:Class="BDZarplata.Pages.Page_SotrudnikMainInfo"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:BDZarplata.Pages"
      mc:Ignorable="d"
      d:DesignHeight="450" d:DesignWidth="800"
      Title="PageSotrudnikMainInfo">

    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition/>

        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"></RowDefinition>
            <RowDefinition/>
            <RowDefinition Height="Auto"></RowDefinition>

        </Grid.RowDefinitions>
        <Menu>
            <Button x:Name="Btn_Add" Cursor="Hand" Content="Добавить" Visibility="Visible" Click="Btn_Add_Click"/>
            <Button x:Name="Btn_Redactir" Cursor="Hand" Content="Изменить" Visibility="Collapsed" Click="Btn_Redactir_Click"/>
            <Button x:Name="Btn_delete" Cursor="Hand" Content="Удалить" Visibility="Collapsed"/>
            <Button x:Name="Btn_Save" Cursor="Hand" Content="Сохранить" Visibility="Collapsed" Click="Btn_Save_Click"/>
```

```
    </Menu>
    <TabControl Grid.Row="1" x:Name="TabC_Main">
      <TabItem x:Name="TabI_MainData" Header="Основные данные сотрудников" GotFocus="TabI_MainData_GotFocus">

        <DataGrid x:Name="DG_Sotrud_Anketa" Grid.ColumnSpan="2" Grid.RowSpan="2"
MouseDoubleClick="DG_Sotrud_Anketa_MouseDoubleClick" AutoGenerateColumns="False">
          <DataGrid.Columns>
            <DataGridTextColumn Binding="{Binding Path=idSotrudnik}" Header="Табельный Номер" />
            <DataGridTextColumn Binding="{Binding Path=full_name}" Header="Полное Имя"/>
            <DataGridTextColumn Binding="{Binding Path=title}" Header="Должность"/>
            <DataGridTextColumn Binding="{Binding Path=family_status}" Header="Семейное Положение"/>
            <DataGridTextColumn Binding="{Binding Path=num_zd_kids}" Header="Здоровых Детей"/>
            <DataGridTextColumn Binding="{Binding Path=num_invalid_kids}" Header="Детей Инвалидов"/>
            <DataGridTextColumn Binding="{Binding Path=opeka}" Header="Опекаемых Детей инвалидов"/>
            <DataGridTextColumn Binding="{Binding Path=SpecStatus}" Header="Специальный статус"/>
            <DataGridTextColumn Binding="{Binding Path=Staj}" Header="Стаж"/>
          </DataGrid.Columns>
        </DataGrid>
      </TabItem>

      <TabItem Header="Расписание" GotFocus="TabItem_GotFocus">
        <Grid>
          <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto"></ColumnDefinition>
            <ColumnDefinition/>
            <ColumnDefinition></ColumnDefinition>
          </Grid.ColumnDefinitions>
          <ListBox x:Name="LB_Sotrud_FIO" Grid.Column="0" SelectionChanged="LB_Sotrud_FIO_SelectionChanged" ></ListBox>
          <ListBox x:Name="LB_Sotrud_id" SelectionChanged="LB_Sotrud_id_SelectionChanged" Visibility="Hidden" ></ListBox>
          <DataGrid x:Name="DG_Raspisnie"
                AutoGenerateColumns="False"
                Grid.Column="1"
                SelectionChanged="DG_Raspisnie_SelectionChanged" >
            <DataGrid.Columns>
```

```
                    <DataGridTextColumn Binding="{Binding Path=Date}" Header="Дата" />
                    <DataGridTextColumn Binding="{Binding Path=StatusSotrud}" Header="Статус сотрудника" />
                    <DataGridTextColumn Binding="{Binding Path=StatusDay}" Header="Статус дня" />
                  </DataGrid.Columns>
                </DataGrid>
                <StackPanel Grid.Column="2">

                  <Calendar x:Name="calendar_raspisan"></Calendar>
                  <Label>Статус Сотрудника</Label>
                  <ComboBox x:Name="CB_StatusSotrud" SelectedIndex="1"></ComboBox>
                  <Label>Статус дня</Label>
                  <ComboBox x:Name="CB_StatusDay" SelectedIndex="1"></ComboBox>
                  <Button x:Name="BTN_RedRaspisan" IsEnabled="False" Click="BTN_RedRaspisan_Click">Сохранить Изменения</Button>

                </StackPanel>
              </Grid>
            </TabItem>
          </TabControl>
        </Grid>
</Page>
```

Page_BughalterInfo.xaml

```
<Page x:Class="BDZarplata.Pages.Page_BughalterInfo"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:BDZarplata.Pages"
      mc:Ignorable="d"
      d:DesignHeight="450" d:DesignWidth="800"
      Title="Page_BughalterInfo">

    <Grid>
      <Grid.ColumnDefinitions>
```

```
    <ColumnDefinition Width="*"></ColumnDefinition>

</Grid.ColumnDefinitions>
<Grid.RowDefinitions>
    <RowDefinition Height="Auto"></RowDefinition>
    <RowDefinition></RowDefinition>
</Grid.RowDefinitions>
<Menu>
    <Button x:Name="Btn_Add" Cursor="Hand" Content="Добавить" Visibility="Collapsed" Click="Btn_Add_Click"/>
    <Button x:Name="Btn_Redactir" Cursor="Hand" Content="Изменить" Visibility="Collapsed" Click="Btn_Redactir_Click"/>
    <Button x:Name="Btn_delete" Cursor="Hand" Content="Удалить" Visibility="Collapsed" Click="Btn_Delete_Click"/>
    <Button x:Name="Btn_Save" Cursor="Hand" Content="Сохранить" Visibility="Collapsed" Click="Btn_Save_Click"/>
    <Button x:Name="Btn_Raschet" Cursor="Hand" Content="Перейти к Созданию Расчётного листа..." Click="Btn_Raschet_Click"></Button>
</Menu>
<TabControl Grid.Row="1" x:Name="TabC_Main">
    <TabItem x:Name="TabI_OkladSotrud" Header="Оклады сотрудников" GotFocus="TabI_OkladSotrud_GotFocus" >
        <DataGrid x:Name="DG_SotridnikOklad"
                  AutoGenerateColumns="False"
                  SelectionChanged="DG_SotridnikOklad_SelectionChanged"  MouseDoubleClick="DG_SotridnikOklad_MouseDoubleClick"  >
            <DataGrid.Columns>
                <DataGridTextColumn Binding="{Binding Path=idDoljnost}" Header="№"/>
                <DataGridTextColumn Binding="{Binding Path=title}" Header="Название"/>
                <DataGridTextColumn Binding="{Binding Path=Oklad}" Header="Оклад"/>
                <DataGridTextColumn Binding="{Binding Path=Travmat}" Header="Травматизм"/>

            </DataGrid.Columns>

        </DataGrid>
    </TabItem>
    <TabItem x:Name="TabI_Nadbav" Header="Штрафы и надбавки" GotFocus="TabI_Nadbav_GotFocus">
        <Grid >
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="Auto"></ColumnDefinition>
                <ColumnDefinition Width="Auto"/>
```

```
            <ColumnDefinition></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <ListBox x:Name="LB_Sotrud_FIO2" Grid.Column="0" SelectionChanged="LB_Sotrud_FIO2_SelectionChanged"></ListBox>
        <ListBox x:Name="LB_Sotrud_id2" SelectionChanged="LB_Sotrud_id2_SelectionChanged" Visibility="Collapsed"></ListBox>
        <DataGrid x:Name="DG_NadbavShtraf"
                  AutoGenerateColumns="False"
                  Grid.Column="1"
                  SelectionChanged="DG_NadbavShtraf_SelectionChanged">
            <DataGrid.Columns>
                <DataGridTextColumn Binding="{Binding Path=Data}" Header="Дата выплаты"/>
                <DataGridTextColumn Binding="{Binding Path=Nadbav}" Header="Сумма надбавок"/>
                <DataGridTextColumn Binding="{Binding Path=Vichet}" Header="Сумма штрафов"/>
            </DataGrid.Columns>

        </DataGrid>

        <StackPanel Grid.Column="2">
            <Calendar x:Name="Calendar2" FontSize="14"/>
            <ComboBox x:Name="CB_Data_Nadbav" SelectedIndex="0" Visibility="Hidden">
                <ComboBoxItem x:Name="CBI_New_Ellement" Background="#FFB1DBF3" Content="&lt;Добавить Новый Эллемент&gt;"
FontWeight="Bold" FontStyle="Italic"/>
            </ComboBox>
            <Label>Надбавка</Label>
            <TextBox x:Name="TB_Nadbav" ></TextBox>
            <Label>Вычет</Label>
            <TextBox x:Name="TB_Vichet" ></TextBox>
            <Button x:Name="Btn_Red_Nadbav"
                    IsEnabled="False"
                    Click="Btn_Red_Nadbav_Click">Сохранить изменения
            </Button>
        </StackPanel>
    </Grid>
</TabItem>
```

```
        <TabItem Header="Льготы и налоги" x:Name="TabI_LN" GotFocus="TabI_LN_GotFocus" Initialized="TabI_LN_Initialized" Margin="-1,-2,-3,0">
            <Grid >
              <Grid.ColumnDefinitions>
                 <ColumnDefinition Width="62"></ColumnDefinition>
                 <ColumnDefinition Width="320"/>
                 <ColumnDefinition></ColumnDefinition>
              </Grid.ColumnDefinitions>
              <StackPanel Grid.Column="0" Grid.ColumnSpan="2">
                 <Label>НДФЛ</Label>
                 <Label>ПФР</Label>
                 <Label>ФСС</Label>
                 <Label>ФОМС</Label>
                 <Label>Размер льготы за 1 и 2 ребенка</Label>
                 <Label>Размер льготы за 3 и последующих детей</Label>
                 <Label>Размер льготы за ребенка-инвалида</Label>
                 <Label>Размер льготы за опеку над ребенком инвалидом</Label>
                 <Label>МРОТ</Label>
              </StackPanel>
              <StackPanel Grid.Column="2">
                 <TextBox x:Name="TB_NDFL" Height="25" Margin="0,6,0,0" PreviewTextInput="floatOnly_PreviewTextInput"></TextBox>
                 <TextBox x:Name="TB_PFR" Height="25" Margin="0,6,0,0" PreviewTextInput="floatOnly_PreviewTextInput"></TextBox>
                 <TextBox x:Name="TB_FCC" Height="25" Margin="0,6,0,0" PreviewTextInput="floatOnly_PreviewTextInput"></TextBox>
                 <TextBox x:Name="TB_FOMC" Height="25" Margin="0,6,0,0" PreviewTextInput="floatOnly_PreviewTextInput"></TextBox>
                 <TextBox x:Name="TB_Kid1" Height="25" Margin="0,6,0,0" PreviewTextInput="intOnly_PreviewTextInput"></TextBox>
                 <TextBox x:Name="TB_Kid3" Height="25" Margin="0,6,0,0" PreviewTextInput="intOnly_PreviewTextInput"></TextBox>
                 <TextBox x:Name="TB_KidInvalid" Height="25" Margin="0,6,0,0" PreviewTextInput="intOnly_PreviewTextInput"></TextBox>
                 <TextBox x:Name="TB_KidInvalid_opek" Height="25" Margin="0,6,0,0" PreviewTextInput="intOnly_PreviewTextInput"></TextBox>
                 <TextBox x:Name="TB_Mrot" Height="25" Margin="0,6,0,0" PreviewTextInput="intOnly_PreviewTextInput" ></TextBox>
              </StackPanel>
            </Grid>
        </TabItem >
        <TabItem x:Name="TabI_Base" Header="Предельная база для исчисления страховых взносов" GotFocus="TabI_Base_GotFocus">
```

```
            <DataGrid x:Name="DG_Base" AutoGenerateColumns="False">
               <DataGrid.Columns>
                  <DataGridTextColumn Binding="{Binding Path=Y}" Header="Год"/>
                  <DataGridTextColumn Binding="{Binding Path=Z}" Header="Значение"/>
               </DataGrid.Columns>
            </DataGrid>
         </TabItem>
      </TabControl>
   </Grid>
</Page>
```

Page_AddRedAnketa.xaml.cs

```
using BDZarplata.Classes;
using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;

namespace BDZarplata.Pages
{
   /// <summary>
   /// Логика взаимодействия для Page_AddRedAnketa.xaml
   /// </summary>
   public partial class Page_AddRedAnketa : Page
   {
      string IDSotrud;
      /// <summary>
      /// Загрузка формы
      /// </summary>
      /// <param name="IdSotrud"></param>
      public Page_AddRedAnketa(string IdSotrud = "")
      {
         IDSotrud = IdSotrud;
         InitializeComponent();
```

```
        DB.LoadDataComboBox(CB_Doljnost, "SELECT [idDoljnost], [title] FROM [BD_Zarplata].[bd_zarplta].[doljnost]", 1);
        DB.LoadDataComboBox(CB_DoljnostID, "SELECT [idDoljnost], [title] FROM [BD_Zarplata].[bd_zarplta].[doljnost]", 0);
        if (IdSotrud != "")
        {
            TB_FIO.Text = DB.queryScalar("SELECT [full_name] FROM [BD_Zarplata].[bd_zarplta].[sotrudnik] WHERE [idSotrudnik]= " + IdSotrud).ToString();
            TB_FamilyStatus.Text = DB.queryScalar("SELECT [family_status] FROM [BD_Zarplata].[bd_zarplta].[sotrudnik] WHERE [idSotrudnik]= " + IdSotrud).ToString();
            TB_KidInvalid.Text = DB.queryScalar("SELECT [num_invalid_kids] FROM [BD_Zarplata].[bd_zarplta].[sotrudnik] WHERE [idSotrudnik]= " + IdSotrud).ToString();
            TB_KidInvalid_opek.Text = DB.queryScalar("SELECT [opeka] FROM [BD_Zarplata].[bd_zarplta].[sotrudnik] WHERE [idSotrudnik]= " + IdSotrud).ToString();
            TB_kidsZd.Text = DB.queryScalar("SELECT [num_zd_kids]  FROM [BD_Zarplata].[bd_zarplta].[sotrudnik] WHERE [idSotrudnik]= " + IdSotrud).ToString();
            TB_SchetZachisl.Text = DB.queryScalar("SELECT [SchetZachisl] [SchetZachisl] FROM [BD_Zarplata].[bd_zarplta].[sotrudnik] WHERE [idSotrudnik]= " + IdSotrud).ToString();
            TB_Staj.Text = DB.queryScalar("SELECT [Staj] FROM [BD_Zarplata].[bd_zarplta].[sotrudnik] WHERE [idSotrudnik]= " + IdSotrud).ToString();
            CB_DoljnostID.SelectedItem = DB.queryScalar("SELECT [idDoljnost] FROM [BD_Zarplata].[bd_zarplta].[sotrudnik] WHERE [idSotrudnik]= " + IdSotrud).ToString();
            CB_Doljnost.SelectedIndex = CB_DoljnostID.SelectedIndex;
            CB_SpecStatus.SelectedIndex = Convert.ToInt32(DB.queryScalar("SELECT [SpecStatus] FROM [BD_Zarplata].[bd_zarplta].[sotrudnik] WHERE [idSotrudnik]= " + IdSotrud));
        }
    }
    /// <summary>
    /// соединение 2 списков для синхронного выбора
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void CB_Doljnost_SelectionChanged(object sender, SelectionChangedEventArgs e)
    {
        CB_DoljnostID.SelectedIndex = CB_Doljnost.SelectedIndex;
    }
```

```
        /// <summary>
        /// Фильтр int значений для TextBox
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void intOnly_PreviewTextInput(object sender, TextCompositionEventArgs e)
        {
            //узнаем Кто вызвал событие
            TextBox textBox = sender as TextBox;
            //проверка что введена цифра
            if (!Char.IsDigit(e.Text, 0))
            {
                e.Handled = true;
                MessageBox.Show("Только Целые числа", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);
            }
            //проверяем выходит ли за предел int
            else if (!int.TryParse(textBox.Text + e.Text, out _))
            {
                MessageBox.Show("Максимальный размер числа не может быть больше 2147483647", "Внимание", MessageBoxButton.OK,
MessageBoxImage.Warning);
                e.Handled = true;

            }
        }
        /// <summary>
        /// Проверка и внесение данных в таблицу
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void Btn_Save_Click(object sender, RoutedEventArgs e)
        {
            List<string> columnNames = new List<string>() { "[full_name]", "[idDoljnost]", "[SpecStatus]" };
```

```
        List<string> RowData = new List<string>() { """ + TB_FIO.Text + """, CB_DoljnostID.SelectedItem.ToString(), CB_SpecStatus.SelectedIndex.ToString() };

        MessageBoxResult Result_SaveOrRed;
        if (TB_FIO.Text.Length < 2)
        {
            MessageBox.Show("Ошибка! Поле ФИО обязательно для заполнения", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);
            return;
        }
        if (TB_FamilyStatus.Text != "")
        {
            columnNames.Add("[family_status]");
            RowData.Add("'" + TB_FamilyStatus.Text + "'");
        }
        if (TB_KidInvalid.Text != "")
        {
            columnNames.Add("[num_invalid_kids]");
            RowData.Add(TB_KidInvalid.Text);
        }
        if (TB_KidInvalid_opek.Text != "")
        {
            columnNames.Add("[opeka]");
            RowData.Add(TB_KidInvalid_opek.Text);
        }
        if (TB_kidsZd.Text != "")
        {
            columnNames.Add("[num_zd_kids]");
            RowData.Add(TB_kidsZd.Text);
        }
        if (TB_SchetZachisl.Text.Length == 20)
        {
            columnNames.Add("[SchetZachisl]");
            RowData.Add("'" + TB_SchetZachisl.Text + "'");
        }
```

```
        else if (TB_SchetZachisl.Text == "")
        { }
        else
        {
            MessageBox.Show("Счёт Зачисления должен состоять из 20 цифр!");
            return;
        }
        if (TB_Staj.Text != "")
        {
            columnNames.Add("[Staj]");
            RowData.Add(TB_Staj.Text);
        }

        if (IDSotrud == "") { Result_SaveOrRed = MessageBoxResult.No; }
        else
        {
            Result_SaveOrRed = MessageBox.Show
                (
                "Сохранить изменения?  \n \n Да - Сохранить изменения  существующей записи \n Нет - Создать новую запись \n Отмена - Возврат
в окно редактирования",
                "Добавить или Изменить?",
                MessageBoxButton.YesNoCancel,
                MessageBoxImage.Question
                );
        }
        switch (Result_SaveOrRed)
        {
            //обновление существующей записи
            case MessageBoxResult.Yes:
                Procedure.UpdateTable("[BD_Zarplata].[bd_zarplta].[sotrudnik]", columnNames, RowData, "WHERE [idSotrudnik]= " + IDSotrud);
                MessageBox.Show("Изменения Сохранены!");
                break;
            case MessageBoxResult.No:
                // вставка новой записи
```

```
                Procedure.InsertTable("[BD_Zarplata].[bd_zarplta].[sotrudnik]", columnNames, RowData);
                MessageBox.Show("Вставка новой записи завершена!");
                break;
            default:
                Manager.UpdateLabel("Отменено пользователем");
                return;
        }
        Manager.MainFrame.Navigate(new Page_SotrudnikMainInfo());
    }

    private void Btn_Delete_Click(object sender, RoutedEventArgs e)
    {
        MessageBoxResult MR;
        if (IDSotrud == "")
        {
            MR = MessageBox.Show("Объект для удаления не выбран. \n  хотите выбрать его сейчас?", "Внимание!", MessageBoxButton.YesNo,
MessageBoxImage.Warning);
            if (MR == MessageBoxResult.Yes)
            {
                Classes.Manager.MainFrame.Navigate(new Page_SotrudnikMainInfo());
            }
            else
            {
                return;
            }
        }
        else if (MessageBoxResult.Yes == MessageBox.Show("Вы собираетесь безвозвратно удалить\n Вы уверены?", "Внимание!",
MessageBoxButton.YesNo, MessageBoxImage.Warning))
        {
            //ТК в БД прописано каскадное удаление ,удаляем лишь из родительской таблицы
            string comand = "DELETE FROM [bd_zarplta].[sotrudnik] WHERE [idSotrudnik] =" + IDSotrud;

            if (Classes.DB.queryData(comand) != -1)
            {
```

```
                Classes.Manager.UpdateLabel("Готово.");
                MessageBox.Show("Операция успешна выполнена!", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);
                Classes.Manager.MainFrame.Navigate(new Page_SotrudnikMainInfo());
            }
            else
            {
                MessageBox.Show("Произошла ошибка при удалении");
            }
        }
    }
}
```

Page_BughalterInfo.xaml.cs

```
using BDZarplata.Classes;
using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;

namespace BDZarplata.Pages
{
    /// <summary>
    /// Логика взаимодействия для Page_BughalterInfo.xaml
    /// </summary>
    public partial class Page_BughalterInfo : Page
    {
        bool Select0Items = true;
        public Page_BughalterInfo()
        {
            Manager.MainProgressBar.Visibility = Visibility.Visible;
            Manager.MainProgressBar.Maximum = 5;

            InitializeComponent();
```

```
    Manager.MainProgressBar.Value = 1;
    DB.loadDataGrid(DG_SotridnikOklad, "SELECT  [idDoljnost],    [title]     ,[Oklad], [Travmat] FROM [BD_Zarplata].[bd_zarplta].[doljnost]");

    TabI_LN_Initialized(null, null);
    DB.loadDataGrid(DG_Base, "SELECT *  FROM [BD_Zarplata].[bd_zarplta].[base]");


    Classes.DB.LoadDataListBox(LB_Sotrud_FIO2, "SELECT  [idSotrudnik],[full_name]  FROM [BD_Zarplata].[bd_zarplta].[sotrudnik]", 1);
    Classes.DB.LoadDataListBox(LB_Sotrud_id2, "SELECT [idSotrudnik]  ,[full_name] FROM [BD_Zarplata].[bd_zarplta].[sotrudnik]", 0);
    Manager.MainProgressBar.Visibility = Visibility.Collapsed;

}
/// <summary>
/// Подгруздка справочника льгот и налогов
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void TabI_LN_Initialized(object sender, EventArgs e)
{
    Manager.MainProgressBar.Value = 2;
    TB_FCC.Text = Classes.DB.queryScalar("SELECT FCC FROM [BD_Zarplata].[bd_zarplta].[h]").ToString();
    TB_FOMC.Text = Classes.DB.queryScalar("SELECT FOMC FROM [BD_Zarplata].[bd_zarplta].[h]").ToString();
    Manager.MainProgressBar.Value = 3;
    TB_Kid1.Text = Classes.DB.queryScalar("SELECT kid1 FROM [BD_Zarplata].[bd_zarplta].[h]").ToString();
    TB_Kid3.Text = Classes.DB.queryScalar("SELECT Kid3 FROM [BD_Zarplata].[bd_zarplta].[h]").ToString();
    TB_KidInvalid.Text = Classes.DB.queryScalar("SELECT invalid FROM [BD_Zarplata].[bd_zarplta].[h]").ToString();
    TB_KidInvalid_opek.Text = Classes.DB.queryScalar("SELECT [invlid_o] FROM [BD_Zarplata].[bd_zarplta].[h]").ToString();
    Manager.MainProgressBar.Value = 5;
    TB_Mrot.Text = Classes.DB.queryScalar("SELECT MROT FROM [BD_Zarplata].[bd_zarplta].[h]").ToString();
    TB_NDFL.Text = Classes.DB.queryScalar("SELECT NDFL FROM [BD_Zarplata].[bd_zarplta].[h]").ToString();
    TB_PFR.Text = Classes.DB.queryScalar("SELECT PFR FROM [BD_Zarplata].[bd_zarplta].[h]").ToString();
    Manager.MainProgressBar.Value = 5;
}
/// <summary>
```

```
        /// сохранение измененных полей таблицы БД
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void Btn_Save_Click(object sender, RoutedEventArgs e)
        {
            switch (TabC_Main.SelectedIndex)
            {
                case 0:
                    break;
                case 2:
                    Console.WriteLine(2);
                    if (MessageBoxResult.OK == MessageBox.Show("Сохранить Изменения? \n Ок -сохранить \n Отмена - Отменить все изменения",
"Запрос на сохранение", MessageBoxButton.OKCancel))
                    {
                        List<string> columnName = new List<string>() { "[NDFL]", "[PFR]", "[FCC]", "[FOMC]", "[kid1]", "[kid3]", "[invalid]", "[invlid_o]",
"[MROT]" };
                        List<string> MasData = new List<string>() { TB_NDFL.Text.Replace(',', '.'), TB_PFR.Text.Replace(',', '.'), TB_FCC.Text.Replace(',', '.'),
TB_FOMC.Text.Replace(',', '.'), TB_Kid1.Text, TB_Kid3.Text, TB_KidInvalid.Text, TB_KidInvalid_opek.Text, TB_Mrot.Text };
                        Classes.Procedure.UpdateTable("[BD_Zarplata].[bd_zarplta].[h]", columnName, MasData, "");
                        MessageBox.Show("Изменения успешно сохранены!");
                    }
                    TabI_LN_Initialized(null, null);
                    break;
                case 1:
                    Btn_Red_Nadbav_Click(sender, e);
                    break;
                default:
                    Console.WriteLine("Дефульт");
                    break;

            }
        }
        /// <summary>
```

```
/// Переход на страницу редактирования
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void Btn_Redactir_Click(object sender, RoutedEventArgs e)
{
   switch (TabC_Main.SelectedIndex)
   {
      case 0:
         Manager.MainFrame.Navigate(new Pages.Page_BuhgaltAddEdit_Oklad(CellID));
         break;
      case 2:
         break;
      case 3:
         string Year = Classes.DG.GetSelectCell(DG_Base, 0);
         string Znach = Classes.DG.GetSelectCell(DG_Base, 1);
         Manager.MainFrame.Navigate(new Pages.Page_Bughl_AddEdit_base(Year, Znach));
         break;
      default:
         break;
   }
}
private void Btn_Add_Click(object sender, RoutedEventArgs e)
{
   Manager.MainFrame.Navigate(new Pages.Page_Bughl_AddEdit_base());
}
/// <summary>
/// Удаление записи из base
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void Btn_Delete_Click(object sender, RoutedEventArgs e)
{
   string YEAR = Classes.DG.GetSelectCell(DG_Base, 0);
```

```
        if (YEAR == "")
        {
            MessageBox.Show("Объект для удаления не выбран. ", "Внимание!", MessageBoxButton.OK, MessageBoxImage.Warning);

        }
        else if (MessageBoxResult.Yes == MessageBox.Show("Вы собираетесь безвозвратно удалить запись\n Вы уверены?", "Внимание!", MessageBoxButton.YesNo, MessageBoxImage.Warning))
        {
            //ТК в БД прописано каскадное удаление ,удаляем лишь из родительской таблицы
            string comand = "DELETE FROM [bd_zarplta].[base] WHERE [Y] =" + YEAR;

            if (Classes.DB.queryData(comand) != -1)
            {
                Classes.Manager.UpdateLabel("Готово.");
                MessageBox.Show("Операция успешна выполнена!", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);
                DB.loadDataGrid(DG_Base, "SELECT *  FROM [BD_Zarplata].[bd_zarplta].[base]");
            }
            else
            {
                MessageBox.Show("Произошла ошибка при удалении");
            }
        }
    }
    private void intOnly_PreviewTextInput(object sender, TextCompositionEventArgs e)
    {
        //узнаем Кто вызвал событие
        TextBox textBox = sender as TextBox;
        //проверка что введена цифра
        if (!Char.IsDigit(e.Text, 0))
        {
            e.Handled = true;
            MessageBox.Show("Только Целые числа", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);
        }
```

```
            //проверяем выходит ли за предел int
            else if (!int.TryParse(textBox.Text + e.Text, out _))
            {
                MessageBox.Show("Максимальный размер числа не может быть больше 2147483647", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);
                e.Handled = true;

            }
        }
        /// <summary>
        /// Ограничение для TextBox для ввода только Дробных данных
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void floatOnly_PreviewTextInput(object sender, TextCompositionEventArgs e)
        {
            //узнаем Кто вызвал событие
            TextBox textBox = sender as TextBox;
            //проверка что в строке лишь 1 .
            string fullText = textBox.Text + e.Text;
            if (e.Text.Contains(",") && (textBox.Text.Contains(",") || textBox.Text.Length == 0))
            {
                e.Handled = true;
                MessageBox.Show("неверно поставлена точка", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

            }
            //проверка что введена цифра или .
            else if (!Char.IsDigit(e.Text, 0) && e.Text != ",")
            {
                e.Handled = true;
                MessageBox.Show("Только дробные числа", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);
            }
            else if (textBox.Text.Length > 4)
            {
```

```
        e.Handled = true;
        MessageBox.Show("Длина строки не может превышать 5 символов", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);
    }

}


/// <summary>
/// запись ID выделенной строки
/// </summary>
string CellID;
private void DG_SotridnikOklad_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    Select0Items = false;
    Btn_Redactir.Visibility = Visibility.Visible;
    CellID = DG.GetSelectCell(DG_SotridnikOklad, 0);
}
/// <summary>
/// Переход в режим редактирования оклада
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void DG_SotridnikOklad_MouseDoubleClick(object sender, MouseButtonEventArgs e)
{
    Manager.MainFrame.Navigate(new Pages.Page_BuhgaltAddEdit_Oklad(CellID));
}
/// <summary>
/// Смена видимости кнопок , при выборе вкладки
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void TabI_OkladSotrud_GotFocus(object sender, RoutedEventArgs e)
{
    Btn_Add.Visibility = Visibility.Collapsed;
```

```
    Btn_delete.Visibility = Visibility.Collapsed;
    if (Select0Items)
    {
        Btn_Redactir.Visibility = Visibility.Collapsed;
    }

    Btn_Save.Visibility = Visibility.Collapsed;
}
/// <summary>
/// Смена видимости кнопок , при выборе вкладки
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void TabI_Base_GotFocus(object sender, RoutedEventArgs e)
{
    Btn_Add.Visibility = Visibility.Visible;
    Btn_delete.Visibility = Visibility.Visible;
    Btn_Redactir.Visibility = Visibility.Visible;
    Btn_Save.Visibility = Visibility.Collapsed;
}
/// <summary>
/// Смена видимости кнопок , при выборе вкладки
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void TabI_LN_GotFocus(object sender, RoutedEventArgs e)
{
    Btn_Add.Visibility = Visibility.Collapsed;
    Btn_delete.Visibility = Visibility.Collapsed;
    Btn_Redactir.Visibility = Visibility.Collapsed;
    Btn_Save.Visibility = Visibility.Visible;
    Btn_Save.IsEnabled = true;

}
```

```
        private void TabI_Nadbav_GotFocus(object sender, RoutedEventArgs e)
        {
            Btn_Add.Visibility = Visibility.Collapsed;
            Btn_delete.Visibility = Visibility.Collapsed;
            Btn_Redactir.Visibility = Visibility.Collapsed;
            Btn_Save.Visibility = Visibility.Visible;
            Btn_Save.IsEnabled = false;
        }
        private void LB_Sotrud_FIO2_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            LB_Sotrud_id2.SelectedIndex = LB_Sotrud_FIO2.SelectedIndex;
        }
        /// <summary>
        /// Подгрузка данных о надбавках и штрафах выбранного сотрудника в DataGrid
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void LB_Sotrud_id2_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            Classes.DB.loadDataGrid(DG_NadbavShtraf, $"SELECT FORMAT([Data],'d') as 'Data',[Nadbav] ,[Vichet] FROM [BD_Zarplata].[bd_zarplta].[zp] Where [Sotrudnik_idSotrudnik]= {LB_Sotrud_id2.SelectedItem}");
            CB_Data_Nadbav.Items.Clear();
            DB.LoadDataComboBox(CB_Data_Nadbav, $"SELECT [Data] FROM [BD_Zarplata].[bd_zarplta].[zp] Where [Sotrudnik_idSotrudnik]= {LB_Sotrud_id2.SelectedItem}", 0);
        }
        /// <summary>
        /// Вывод в форму даты , надбавки и штрафов
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void DG_NadbavShtraf_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            try
            {
```

```
            Calendar2.DisplayDate = Convert.ToDateTime(DG.GetSelectCell(DG_NadbavShtraf, 0));
            Calendar2.SelectedDate = Convert.ToDateTime(DG.GetSelectCell(DG_NadbavShtraf, 0));
            TB_Nadbav.Text = DG.GetSelectCell(DG_NadbavShtraf, 1);
            TB_Vichet.Text = DG.GetSelectCell(DG_NadbavShtraf, 2);
            Btn_Red_Nadbav.IsEnabled = true;
            Btn_Save.IsEnabled = true;

        }
        catch
        {
            Calendar2.DisplayDate = DateTime.Now;
            Calendar2.SelectedDate = DateTime.Now;
            TB_Nadbav.Text = "";
            TB_Vichet.Text = "";
            Btn_Red_Nadbav.IsEnabled = false;
            Btn_Save.IsEnabled = false;
        }
    }
    /// <summary>
    /// Переход к странице расчета
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void Btn_Raschet_Click(object sender, RoutedEventArgs e)
    {
        Manager.MainFrame.Navigate(new Page_raschet());
    }
    /// <summary>
    /// Внесение изменений в БД
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void Btn_Red_Nadbav_Click(object sender, RoutedEventArgs e)
    {
```

```
            Procedure.UpdateTable
                (
                "[BD_Zarplata].[bd_zarplta].[zp]"
                , new List<string>() { "[Vichet]", "[Nadbav]" }
                , new List<string>() { TB_Vichet.Text, TB_Nadbav.Text }
                , $"where[Sotrudnik_idSotrudnik] = {LB_Sotrud_id2.SelectedItem} AND [Data] = '{Calendar2.SelectedDate}'"
                );
            Classes.DB.loadDataGrid(DG_NadbavShtraf,    $"SELECT       FORMAT([Data],'d'),[Nadbav]                ,[Vichet]                FROM
[BD_Zarplata].[bd_zarplta].[zp]  Where [Sotrudnik_idSotrudnik]= {LB_Sotrud_id2.SelectedItem}");
            MessageBox.Show("Изменения сохранены!");
        }
    }
}
```

Page_Bughl_AddEdit_base.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;

namespace BDZarplata.Pages
{
    /// <summary>
    /// Логика взаимодействия для Page_Bughl_AddEdit_base.xaml
    /// </summary>
    public partial class Page_Bughl_AddEdit_base : Page
    {
        /// <summary>
        /// Режим Добавления новых данных
        /// </summary>
        bool AddMode;
        string YEAR;
```

```
public Page_Bughl_AddEdit_base(string year = "", string znach = "")
{
    if (year == "")
    {
        AddMode = true;
    }
    else
    {
        AddMode = false;
    }

    InitializeComponent();
    TB_Year.Text = year;
    TB_Znach.Text = znach;
    YEAR = year;
}
/// <summary>
/// Сохранение данных
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void BtnSave_Click(object sender, RoutedEventArgs e)
{
    if (TB_Year.Text == "")
    {
        MessageBox.Show("Поле год обязательно для заполнения!");
        return;
    }

    if (TB_Znach.Text == "")
    {
        MessageBox.Show("Поле значение обязательно для заполнения!");
        return;
    }
```

```
        List<string> columnNames = new List<string>() { "[Y]", "[Z]" };
        List<string> RowData = new List<string>() { TB_Year.Text, TB_Znach.Text };
        if (AddMode)
        {
            Classes.Procedure.InsertTable("[BD_Zarplata].[bd_zarplta].[base]", columnNames, RowData);
            MessageBox.Show("Вставка новой записи завершена!");
        }
        else
        {
            Classes.Procedure.UpdateTable("[BD_Zarplata].[bd_zarplta].[base]", columnNames, RowData, "WHERE [Y]= " + TB_Year.Text);
            MessageBox.Show("Обновление записи успешно!");

        }
        Classes.Manager.MainFrame.Navigate(new Pages.Page_BughalterInfo());
    }
    private void Btn_Delete_Click(object sender, RoutedEventArgs e)
    {
        MessageBoxResult MR;
        if (YEAR == "")
        {
            MR = MessageBox.Show("Объект для удаления не выбран. \n  хотите выбрать его сейчас?", "Внимание!", MessageBoxButton.YesNo,
MessageBoxImage.Warning);
            if (MR == MessageBoxResult.Yes)
            {
                Classes.Manager.MainFrame.Navigate(new Page_BughalterInfo());
            }
            else
            {
                return;
            }
        }
        else if (MessageBoxResult.Yes == MessageBox.Show("Вы собираетесь безвозвратно удалить\n Вы уверены?", "Внимание!",
MessageBoxButton.YesNo, MessageBoxImage.Warning))
        {
```

```
            //ТК в БД прописано каскадное удаление ,удаляем лишь из родительской таблицы
            string comand = "DELETE FROM [bd_zarplta].[base] WHERE [Y] =" + YEAR;

            if (Classes.DB.queryData(comand) != -1)
            {
                Classes.Manager.UpdateLabel("Готово.");
                MessageBox.Show("Операция успешна выполнена!", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);
                Classes.Manager.MainFrame.Navigate(new Page_BughalterInfo());
            }
            else
            {
                MessageBox.Show("Произошла ошибка при удалении");
            }
        }
    }
    private void intOnly_PreviewTextInput(object sender, TextCompositionEventArgs e)
    {
        //узнаем Кто вызвал событие
        TextBox textBox = sender as TextBox;
        //проверка что введена цифра
        if (!Char.IsDigit(e.Text, 0))
        {
            e.Handled = true;
            MessageBox.Show("Только Целые числа", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);
        }
        //проверяем выходит ли за предел int
        else if (!int.TryParse(textBox.Text + e.Text, out _))
        {
            MessageBox.Show("Максимальный размер числа не может быть больше 2147483647", "Внимание", MessageBoxButton.OK,
MessageBoxImage.Warning);
            e.Handled = true;

        }
    }
```

```
        private void shortOnly_PreviewTextInput(object sender, TextCompositionEventArgs e)
        {
            //узнаем Кто вызвал событие
            TextBox textBox = sender as TextBox;
            //проверка что введена цифра
            if (!Char.IsDigit(e.Text, 0))
            {
                e.Handled = true;
                MessageBox.Show("Только Целые числа", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);
            }
            //проверяем выходит ли за предел int
            else if (!short.TryParse(textBox.Text + e.Text, out _))
            {
                MessageBox.Show("Максимальный размер числа не может быть больше 32000", "Внимание", MessageBoxButton.OK,
MessageBoxImage.Warning);
                e.Handled = true;

            }
        }
    }
}
```

Page_BuhgaltAddEdit_Oklad.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;

namespace BDZarplata.Pages
{
    /// <summary>
    /// Логика взаимодействия для Page_BuhgaltAddEdit_Oklad.xaml
```

```
/// </summary>
public partial class Page_BuhgaltAddEdit_Oklad : Page
{

    public Page_BuhgaltAddEdit_Oklad(string IdDoljnost = "NOT")
    {
        InitializeComponent();
        if (IdDoljnost != "NOT")
        {
            TB_title.Text = Classes.DB.queryScalar($"Select * FROM [BD_Zarplata].[bd_zarplta].[doljnost] where [idDoljnost]={IdDoljnost}", 1);
            TB_Oklad.Text = Classes.DB.queryScalar($"Select * FROM [BD_Zarplata].[bd_zarplta].[doljnost] where [idDoljnost]={IdDoljnost}", 2);
            TB_Travmat.Text = Classes.DB.queryScalar($"Select * FROM [BD_Zarplata].[bd_zarplta].[doljnost] where [idDoljnost]={IdDoljnost}", 3);
            id = IdDoljnost;
        }
    }
    string id;
    int mrot = Convert.ToInt32(Classes.DB.queryScalar($"Select [MROT] FROM[BD_Zarplata].[bd_zarplta].[h]"));
    /// <summary>
    /// Проверка и сохранение при их корректности
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void BtnSave_Click(object sender, RoutedEventArgs e)
    {
        if (Convert.ToInt32(TB_Oklad.Text) < mrot)
        {
            MessageBox.Show($"Оклад не может быть меньше МРОТ! ({mrot} руб) ");
            return;
        }
        if (Convert.ToDouble(TB_Travmat.Text) < 0.2 || Convert.ToDouble(TB_Travmat.Text) > 8.5)
        {
            MessageBox.Show("Процент травматизма должен быть в пределах от 0.2 до 8.5");
            return;
        }
```

```
            List<string> columnname = new List<string>() { "[Oklad]", "[Travmat]" };
            List<string> datacolumn = new List<string>() { TB_Oklad.Text, TB_Travmat.Text.Replace(',', '.') };
            MessageBoxResult boxResult = MessageBox.Show("Сохранить изменения?", "Запрос на сохранение", MessageBoxButton.YesNo);
            if (boxResult == MessageBoxResult.Yes)
            {
                Classes.Procedure.UpdateTable("[bd_zarplta].[doljnost]", columnname, datacolumn, $"WHERE [idDoljnost]={id}");
                Classes.Manager.MainFrame.Navigate(new Pages.Page_BughalterInfo());
            }
        }
        /// <summary>
        /// Фильтр int значений для TextBox
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void intOnly_PreviewTextInput(object sender, TextCompositionEventArgs e)
        {
            //узнаем Кто вызвал событие
            TextBox textBox = sender as TextBox;
            //проверка что введена цифра
            if (!Char.IsDigit(e.Text, 0))
            {
                e.Handled = true;
                MessageBox.Show("Только Целые числа", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);
            }
            //проверяем выходит ли за предел int
            else if (!int.TryParse(textBox.Text + e.Text, out _))
            {
                MessageBox.Show("Максимальный размер числа не может быть больше 2147483647", "Внимание", MessageBoxButton.OK,
MessageBoxImage.Warning);
                e.Handled = true;

            }
        }
```

```
        private void floatOnly_PreviewTextInput(object sender, TextCompositionEventArgs e)
        {

            //узнаем Кто вызвал событие
            TextBox textBox = sender as TextBox;
            //проверка что в строке лишь 1 .
            string fullText = textBox.Text + e.Text;
            if (e.Text.Contains(",") && (textBox.Text.Contains(",") || textBox.Text.Length == 0))
            {
                e.Handled = true;
                MessageBox.Show("неверно поставлена точка", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

            }
            //проверка что введена цифра или .
            else if (!Char.IsDigit(e.Text, 0) && e.Text != ",")
            {
                e.Handled = true;
                MessageBox.Show("Только дробные числа", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);
            }
            else if (textBox.Text.Length > 2)
            {
                e.Handled = true;
                MessageBox.Show("Длина строки не может превышать 3 символов", "Внимание", MessageBoxButton.OK,
MessageBoxImage.Warning);
            }

        }
    }
}
```

Page_Login.xaml.cs

```
using System.Windows;
using System.Windows.Controls;
```

```
namespace BDZarplata.Pages
{
    /// <summary>
    /// Логика взаимодействия для Page_Login.xaml
    /// </summary>
    public partial class Page_Login : Page
    {
        public Page_Login()
        {
            InitializeComponent();
        }
        /// <summary>
        /// Попытка подключения к БД
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void BtnLogin_Click(object sender, RoutedEventArgs e)
        {
            Classes.Manager.UpdateLabel("Попытка подключения К БД");
            if (Classes.DB_Connect.OpenClouseConnection(TB_IpPc.Text, TB_DBName.Text))
            {
                Classes.Manager.UpdateLabel("Подключение прошло успешно");

                object DostupLvl =
                 Classes.DB.queryScalar("SELECT  t2.AccessLvl " +
                    "FROM[bd_zarplta].[sotrudnik] " +
                    "t1 LEFT JOIN bd_zarplta.doljnost t2 ON t1.idDoljnost = t2.idDoljnost " +
                    "where t1.idSotrudnik=" + TB_idSotrudnik.Text).ToString();
                switch (DostupLvl)
                {
                    case "0":
                        MessageBox.Show("В Доступе Отказано! Обратитесь в Бухгалтерию для получения выписки");
                        break;
                    case "1":
```

```
                Classes.Manager.MainFrame.Navigate(new Page_BughalterInfo());
                break;
            case "2":
                Classes.Manager.MainFrame.Navigate(new Pages.Page_SotrudnikMainInfo());
                break;
            default:
                MessageBox.Show("Неизвестный уровень доступа " + DostupLvl);
                break;
        }

    }
    else
    {
        Classes.Manager.UpdateLabel("Ошибка");
    }
}
/// <summary>
/// Переключение поля для ввода адреса
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void CB_IPPC_Localhost_Click(object sender, RoutedEventArgs e)
{
    if (CB_IPPC_Localhost.IsChecked == true)
    {
        TB_IpPc.Text = "localhost";
        TB_IpPc.IsEnabled = false;
    }
    else
    {
        TB_IpPc.Text = "";
        TB_IpPc.IsEnabled = true;
    }
}
```

```
        /// <summary>
        /// Переключение поля для ввода названия БД
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void CB_BD_NAmeDef_Click(object sender, RoutedEventArgs e)
        {
            if (CB_BD_NAmeDef.IsChecked == true)
            {
                TB_DBName.Text = "BD_Zarplata";
                TB_DBName.IsEnabled = false;
            }
            else
            {
                TB_DBName.Text = "";
                TB_DBName.IsEnabled = true;
            }
        }
    }
}
```

Page_raschet.xaml.cs

```
using BDZarplata.Classes;
using System;
using System.Windows;
using System.Windows.Controls;
using Excel = Microsoft.Office.Interop.Excel;
namespace BDZarplata.Pages
{
    /// <summary>
    /// Логика взаимодействия для Page_raschet.xaml
    /// </summary>
    public partial class Page_raschet : Page
    {
```

```
double NDFL_stavka, PFR_stavka, FCC_stavka, FOMC_stavka, TRavmat_stavka, Mrot_stavka;
int Kid1_stavka, Kid3_stavka, Invalid_stavka, Invalid_o_stavka;
int Nadbav, Vichet;



public Page_raschet()
{
   InitializeComponent();
   DB.LoadDataComboBox(CB_FIO, "SELECT [idSotrudnik] ,[full_name] FROM[BD_Zarplata].[bd_zarplta].[sotrudnik]", 1);
   DB.LoadDataComboBox(CB_SotrudID, "SELECT [idSotrudnik] ,[full_name] FROM[BD_Zarplata].[bd_zarplta].[sotrudnik]", 0);
   int[] tempColumns = { 0, 1, 2, 3, 4, 5, 6, 7, 8 };
   string[] temp = DB.queryScalar("Select * FROM [BD_Zarplata].[bd_zarplta].[h]", tempColumns);
   NDFL_stavka = Convert.ToDouble(temp[0]);
   PFR_stavka = Convert.ToDouble(temp[1]);
   FCC_stavka = Convert.ToDouble(temp[2]);
   FOMC_stavka = Convert.ToDouble(temp[3]);
   Kid1_stavka = Convert.ToInt32(temp[4]);
   Kid3_stavka = Convert.ToInt32(temp[5]);
   Invalid_stavka = Convert.ToInt32(temp[6]);
   Invalid_o_stavka = Convert.ToInt32(temp[7]);
   Mrot_stavka = Convert.ToInt32(temp[8]);

}
public double itogZarplata = 0;
/// <summary>
/// Расчет зарплаты
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void BTN_Raschet_Click(object sender, RoutedEventArgs e)
{

   DateTime StrartdateTime;
```

```
//ищем ближайшую предыдущую выплату ЗП
var StartVar = DB.queryScalar($"SELECT [Data]  " +
   $"FROM[BD_Zarplata].[bd_zarplta].[zp]  " +
   $"Where[Sotrudnik_idSotrudnik] = {CB_SotrudID.SelectedItem} AND[Data] < '{CB_Date.SelectedItem}' " +
   $"order by Data desc");
//Если не находим начинаем считать зарплату сначала месяца
if (StartVar == "null")
{

   StrartdateTime = calendar.DisplayDate;
   StrartdateTime = StrartdateTime.AddDays(-StrartdateTime.Day + 1);
}
//иначе считаем с него
else { StrartdateTime = Convert.ToDateTime(StartVar); }
//считаем количество рабочих дней
int KolvoRabDney = Convert.ToInt32(DB.queryScalar($"SELECT COUNT(*) FROM[BD_Zarplata].[bd_zarplta].[graphik_rabot] " +
   $"where[Sotrudnik_idSotrudnik] = {CB_SotrudID.SelectedItem} " +
   $"AND DATE >= '{StrartdateTime}' " +
   $"AND DATE < '{CB_Date.SelectedItem}' " +
   $"AND[StatusDay] = 'рабочий'   "));
double DnevDohod = Convert.ToInt32(L_Oklad.Content) / KolvoRabDney;
double OkladMes = 0;
for (DateTime i = StrartdateTime; i < calendar.DisplayDate; i = i.AddDays(1))
{
   string statusDay = DB.queryScalar($"SELECT [StatusDay] FROM[BD_Zarplata].[bd_zarplta].[graphik_rabot]" +
      $" where[Sotrudnik_idSotrudnik] =  {CB_SotrudID.SelectedItem} AND DATE = '{i}'").ToString();
   string statusSotrud = DB.queryScalar($"SELECT [StatusSotrud] FROM[BD_Zarplata].[bd_zarplta].[graphik_rabot]" +
      $" where[Sotrudnik_idSotrudnik] =  {CB_SotrudID.SelectedItem} AND DATE = '{i}'").ToString();
   Manager.UpdateLabel(i.ToString());
   //начисление зп в обычный день
   if (statusDay.LastIndexOf("рабочий") != -1 && statusSotrud.LastIndexOf("вышел") != -1)
   {
      OkladMes += DnevDohod;
   }
```

```
            //начисление ЗП за выход выходной
            else if (statusDay.LastIndexOf("выходной") != -1 && statusSotrud.LastIndexOf("вышел") != -1)
            {
                OkladMes += DnevDohod * 2;
            }
            else
            {
                //расчет больничного
            }
        }

        Vichet = Convert.ToInt32(DB.queryScalar($"SELECT  [Vichet] FROM[BD_Zarplata].[bd_zarplta].[zp] where[Data] =
'{CB_Date.SelectedItem}' AND [Sotrudnik_idSotrudnik] = {CB_SotrudID.SelectedItem}"));
        Nadbav = Convert.ToInt32(DB.queryScalar($"SELECT [Nadbav] FROM[BD_Zarplata].[bd_zarplta].[zp] where[Data] =
'{CB_Date.SelectedItem}' AND [Sotrudnik_idSotrudnik] = {CB_SotrudID.SelectedItem}"));
        //сумма зарплаты работника до вычета налогов
        double FactDohod = OkladMes + Nadbav - Vichet;

        double nalogBase = FactDohod;

        int CountZdKids = Convert.ToInt32(DB.queryScalar($"SELECT [num_zd_kids]  FROM [BD_Zarplata].[bd_zarplta].[sotrudnik] WHEre
[idSotrudnik]={CB_SotrudID.SelectedItem}"));
        int CountInvalidKids = Convert.ToInt32(DB.queryScalar($"SELECT [num_invalid_kids]  FROM [BD_Zarplata].[bd_zarplta].[sotrudnik]
WHEre [idSotrudnik]={CB_SotrudID.SelectedItem}"));
        int CountInvalidOpekaKids = Convert.ToInt32(DB.queryScalar($"SELECT [opeka]  FROM [BD_Zarplata].[bd_zarplta].[sotrudnik] WHEre
[idSotrudnik]={CB_SotrudID.SelectedItem}"));
        int SpecStatus = Convert.ToInt32(DB.queryScalar($"SELECT [SpecStatus]  FROM [BD_Zarplata].[bd_zarplta].[sotrudnik] WHEre
[idSotrudnik]={CB_SotrudID.SelectedItem}"));

        if (SpecStatus == 1)
        {
            nalogBase = nalogBase - 3000;
        }
        else if (SpecStatus == 2)
```

```
    {
        nalogBase = nalogBase - 500;
    }

    if (CountZdKids <= 2)
    {
        nalogBase = nalogBase - (Kid1_stavka * CountZdKids);
    }
    else if (CountZdKids > 2)
    {
        nalogBase = nalogBase - (Kid1_stavka * 2 + Kid3_stavka * (CountZdKids - 2));
    }

    nalogBase = nalogBase - CountInvalidKids * Invalid_stavka;
    nalogBase = nalogBase - CountInvalidOpekaKids * Invalid_o_stavka;
    //Ндфл которое платит работник
    double NdFl = (nalogBase / 100 * NDFL_stavka);
    L_Ndfl.Content = NdFl;
    //Итоговая зарплата на руки работнику
    itogZarplata = FactDohod - NdFl;
    //налоги  уплачиваемые работодателем
    L_FCC.Content = (FactDohod / 100 * FCC_stavka);
    L_FOMC.Content = (FactDohod / 100 * FOMC_stavka);
    L_pfr.Content = (FactDohod / 100 * PFR_stavka);
    L_Travmatizm.Content = (FactDohod / 100 * TRavmat_stavka);
    BTN_Export.IsEnabled = true;

}
/// <summary>
/// Экспорт в Эксель
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void BTN_Export_Click(object sender, RoutedEventArgs e)
```

```
    {
        Excel.Application ex = new Microsoft.Office.Interop.Excel.Application();
        ex.SheetsInNewWorkbook = 1;
        Excel.Workbook workBook = ex.Workbooks.Add(Type.Missing);
        ex.DisplayAlerts = false;
        Excel.Worksheet sheet = (Excel.Worksheet)ex.Worksheets.get_Item(1);

        // sheet.Range["A1", "K1"].Merge(); - объединение ячеек вы заданном диапазоне
        // sheet.Cells.Range["A1", "A7"].Font.Bold = true; включение жирного текста
        //sheet.Cells[1, 1] = "String" присвоить значение ячейки
        //sheet.Cells.WrapText = true; перенос текста в ячейках
        // sheet.Range["B13", "B13"].BorderAround2(Excel.XlLineStyle.xlContinuous, Excel.XlBorderWeight.xlThin,
Excel.XlColorIndex.xlColorIndexAutomatic);
        //рамка для ячейки

        sheet.Range["A1", "Q2"].VerticalAlignment = Excel.XlVAlign.xlVAlignCenter;
        sheet.Range["A1", "B2"].HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;
        sheet.Cells.Font.Name = "Times New Roman"; //задание шрифта
        sheet.Cells.Font.Size = "12";

        sheet.Columns[1].ColumnWidth = 30;
        sheet.Columns[6].ColumnWidth = 30;
        sheet.Cells.Range["A1", "F4"].Font.Bold = true;

        sheet.Cells[1, 1] = "Расчетный листок за " + Convert.ToDateTime(CB_Date.SelectedItem).ToShortDateString();
        sheet.Cells[2, 1] = "ООО";
        sheet.Cells[3, 1] = "Работник: " + CB_FIO.SelectedItem;
        sheet.Cells[4, 1] = "Табельный номер: " + CB_SotrudID.SelectedItem;
        sheet.Cells[4, 6] = "Должность: " + L_DoljnostTitle.Content;
        sheet.Cells.Range["A5", "H6"].Font.Italic = true;
        sheet.Cells.Range["A5", "H5"].BorderAround2(Excel.XlLineStyle.xlContinuous, Excel.XlBorderWeight.xlMedium,
Excel.XlColorIndex.xlColorIndexAutomatic);

        sheet.Cells[5, 1] = "Вид";
```

```
        sheet.Cells[5, 2] = "Период";
        sheet.Cells[5, 3] = "Дни";
        sheet.Cells[5, 4] = "Часы";
        sheet.Cells[5, 5] = "Сумма";
        sheet.Cells[5, 6] = "Вид";
        sheet.Cells[5, 7] = "Период";
        sheet.Cells[5, 8] = "Сумма";

        sheet.Range["A5", "H15"].BorderAround2(Excel.XlLineStyle.xlContinuous, Excel.XlBorderWeight.xlMedium,
Excel.XlColorIndex.xlColorIndexAutomatic);

        sheet.Range["A1", "H1"].Merge();
        sheet.Range["A2", "H2"].Merge();
        sheet.Range["A3", "E3"].Merge();
        sheet.Range["F4", "H4"].Merge();

        sheet.Cells[6, 1] = "1. Начислено";
        sheet.Cells[7, 1] = "Оклад";
        sheet.Cells[7, 5] = L_Oklad.Content;
        sheet.Cells[8, 1] = "Дополнительные выплаты";
        sheet.Cells[8, 5] = Nadbav;
        sheet.Cells[9, 1] = "Больничные пособия";
        sheet.Cells[11, 1] = "Всего начислено:";
        sheet.Cells[11, 5] = "=СУММ(E7:E10)";

        sheet.Cells[12, 1] = "3. Взносы в ПФР";
        sheet.Cells[13, 1] = "Страховые взносы в ПФР (страховая часть 22%)";
        sheet.Cells[13, 5] = L_pfr.Content;

        sheet.Cells[6, 6] = "2.Удержано";
        sheet.Cells[7, 6] = "НДФЛ по ставке 13%";
        sheet.Cells[7, 8] = L_Ndfl.Content;
        sheet.Cells[8, 6] = "Иные удержания";
        sheet.Cells[8, 8] = Vichet;
```

```
            sheet.Cells[11, 6] = "Всего удержано:";
            sheet.Cells[11, 8] = "=СУММ(H7:H10)";
            sheet.Cells[12, 6] = "Сумма к выплате";
            sheet.Cells[12, 8] = itogZarplata;
            sheet.Cells[13, 6] = "Зачислено на счёт№" + DB.queryScalar("SELECT [SchetZachisl] FROM [BD_Zarplata].[bd_zarplta].[sotrudnik]
WHERE" + CB_SotrudID.SelectedItem);
            sheet.Cells[14, 6] = "Выдано наличными";
            sheet.Cells[14, 8] = itogZarplata;

            sheet.Cells.WrapText = true;
            ex.Visible = true;

        }
        /// <summary>
        /// сопоставление ФИО и ID сотрудника
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void CB_SotrudID_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            CB_FIO.SelectedIndex = CB_SotrudID.SelectedIndex;
        }
        /// <summary>
        /// подгрузка информации о сотруднике
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void CB_FIO_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            BTN_Export.IsEnabled = false;
            BTN_Raschet.IsEnabled = false;
            CB_Date.Items.Clear();
            CB_SotrudID.SelectedIndex = CB_FIO.SelectedIndex;
```

```
        DB.LoadDataComboBox(CB_Date, "SELECT [Data] FROM [BD_Zarplata].[bd_zarplta].[zp] wHERE [Sotrudnik_idSotrudnik]= " +
CB_SotrudID.SelectedItem, 0);
        L_IdDoljnost.Content = DB.queryScalar("SELECT [idDoljnost],[full_name] FROM[BD_Zarplata].[bd_zarplta].[sotrudnik] WHERE
[idSotrudnik]=" + CB_SotrudID.SelectedItem);
        L_DoljnostTitle.Content = DB.queryScalar("SELECT [title] FROM[BD_Zarplata].[bd_zarplta].[doljnost] WHERE [idDoljnost]=" +
L_IdDoljnost.Content);
        L_Oklad.Content = DB.queryScalar("SELECT   [Oklad]  FROM [BD_Zarplata].[bd_zarplta].[doljnost] where [idDoljnost] =" +
L_IdDoljnost.Content);
        TRavmat_stavka = Convert.ToDouble(DB.queryScalar("SELECT   [Travmat]  FROM [BD_Zarplata].[bd_zarplta].[doljnost] where
[idDoljnost] =" + L_IdDoljnost.Content));
      }
      /// <summary>
      /// Изменение даты на календаре
      /// </summary>
      /// <param name="sender"></param>
      /// <param name="e"></param>
      private void CB_Date_SelectionChanged(object sender, SelectionChangedEventArgs e)
      {
        calendar.DisplayDate = Convert.ToDateTime(CB_Date.SelectedItem);
        calendar.SelectedDate = Convert.ToDateTime(CB_Date.SelectedItem);
        BTN_Raschet.IsEnabled = true;
      }
   }
}
```

Page_SotrudnikMainInfo.xaml.cs

```
using BDZarplata.Classes;
using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;

namespace BDZarplata.Pages
{
```

```
    /// <summary>
    /// Логика взаимодействия для Page_SotrudnikMainInfo.xaml
    /// </summary>
    public partial class Page_SotrudnikMainInfo : Page
    {
        public Page_SotrudnikMainInfo()
        {
            InitializeComponent();
            Classes.DB.loadDataGrid(DG_Sotrud_Anketa, "SELECT [idSotrudnik], full_name , t2.title, family_status, num_zd_kids,num_invalid_kids,
opeka, SpecStatus, Staj FROM[bd_zarplta].[sotrudnik] t1 LEFT JOIN bd_zarplta.doljnost t2 ON t1.idDoljnost = t2.idDoljnost");
            Classes.DB.LoadDataListBox(LB_Sotrud_FIO, "SELECT  [idSotrudnik],[full_name]  FROM [BD_Zarplata].[bd_zarplta].[sotrudnik]", 1);
            Classes.DB.LoadDataListBox(LB_Sotrud_id, "SELECT [idSotrudnik]  ,[full_name] FROM [BD_Zarplata].[bd_zarplta].[sotrudnik]", 0);
            DB.LoadDataComboBox(CB_StatusDay, "SELECT DISTINCT [StatusDay] FROM[BD_Zarplata].[bd_zarplta].[graphik_rabot]", 0);
            DB.LoadDataComboBox(CB_StatusSotrud, "SELECT DISTINCT [StatusSotrud] FROM[BD_Zarplata].[bd_zarplta].[graphik_rabot]", 0);
        }
        /// <summary>
        /// Переход в режим редактирования
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void Btn_Redactir_Click(object sender, RoutedEventArgs e)
        {
            string sotrudid = Classes.DG.GetSelectCell(DG_Sotrud_Anketa, 0);
            switch (TabC_Main.SelectedIndex)
            {
                case 0:
                    Manager.MainFrame.Navigate(new Page_AddRedAnketa(sotrudid));
                    break;
            }
        }
        /// <summary>
        /// сохранение измененных полей таблицы БД
        /// </summary>
        /// <param name="sender"></param>
```

```
        /// <param name="e"></param>
        private void Btn_Save_Click(object sender, RoutedEventArgs e)
        {

        }
        /// <summary>
        /// Сопоставление ФИО и ID сотрудника
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void LB_Sotrud_FIO_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            LB_Sotrud_id.SelectedIndex = LB_Sotrud_FIO.SelectedIndex;
        }
        /// <summary>
        /// Подгрузка данных о Расписании выбранного сотрудника
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void LB_Sotrud_id_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            Classes.DB.loadDataGrid(DG_Raspisnie, $"SELECT FORMAT([DATE],'d') as 'Date' ,[StatusSotrud] ,[StatusDay] FROM
[BD_Zarplata].[bd_zarplta].[graphik_rabot] where [Sotrudnik_idSotrudnik] = {LB_Sotrud_id.SelectedItem}");
        }


        /// <summary>
        /// Переход в режим редактирования
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void DG_Sotrud_Anketa_MouseDoubleClick(object sender, MouseButtonEventArgs e)
        {
            Btn_Redactir_Click(sender, e);
```

```
}
/// <summary>
/// Переход в режим добавления нового сотрудника
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void Btn_Add_Click(object sender, RoutedEventArgs e)
{
   Manager.MainFrame.Navigate(new Page_AddRedAnketa());
}
/// <summary>
/// Подгрузка данных о выбранном дне
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void DG_Raspisnie_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
   try
   {
      calendar_raspisan.DisplayDate = Convert.ToDateTime(DG.GetSelectCell(DG_Raspisnie, 0));
      calendar_raspisan.SelectedDate = Convert.ToDateTime(DG.GetSelectCell(DG_Raspisnie, 0));
      CB_StatusDay.SelectedItem = DG.GetSelectCell(DG_Raspisnie, 2);
      CB_StatusSotrud.SelectedItem = DG.GetSelectCell(DG_Raspisnie, 1);
      BTN_RedRaspisan.IsEnabled = true;
   }
   catch
   {
      calendar_raspisan.DisplayDate = DateTime.Now;
      calendar_raspisan.SelectedDate = DateTime.Now;
      CB_StatusDay.SelectedIndex = 1;
      CB_StatusSotrud.SelectedIndex = 1;
      BTN_RedRaspisan.IsEnabled = false;
   }
}
```

```
        /// <summary>
        /// сохранение измененных полей таблицы БД
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void BTN_RedRaspisan_Click(object sender, RoutedEventArgs e)
        {
            Procedure.UpdateTable
                (
                "[BD_Zarplata].[bd_zarplta].[graphik_rabot]"
                , new List<string>() { "[StatusSotrud]", "[StatusDay]" }
                , new List<string>() { "'" + CB_StatusSotrud.SelectedItem + "'", "'" + CB_StatusDay.SelectedItem + "'" }
                , $"where[Sotrudnik_idSotrudnik] = {LB_Sotrud_id.SelectedItem} AND[DATE] = '{calendar_raspisan.SelectedDate}'"
                );
            Classes.DB.loadDataGrid(DG_Raspisnie, $"SELECT FORMAT([DATE],'d') ,[StatusSotrud] ,[StatusDay] FROM
[BD_Zarplata].[bd_zarplta].[graphik_rabot] where [Sotrudnik_idSotrudnik] = {LB_Sotrud_id.SelectedItem}");
            MessageBox.Show("Успешно Обновленно!");
        }

        private void TabI_MainData_GotFocus(object sender, RoutedEventArgs e)
        {
            Btn_Add.Visibility = Visibility.Visible;
        }

        private void TabItem_GotFocus(object sender, RoutedEventArgs e)
        {
            Btn_Add.Visibility = Visibility.Hidden;
        }
    }
}
```

СПИСОК ИСТОЧНИКОВ

1. Федеральный закон от 29.12.2006 N 255-ФЗ (ред. от 29.12.2020) «Об обязательном социальном страховании на случай временной нетрудоспособности и в связи с материнством» Статья 14.
2. Налоговый Кодекс Российской Федерации Статья 218. Стандартные налоговые вычеты
3. "Трудовой кодекс Российской Федерации" статья 136
4. Основы проектирования баз данных: учебник для студ. Учреждений сред. Проф образования Г.Н. Федорова.-М.: Издательский центр «Академия», 2017. – 224 с.
5. Г.Н. Федорова. Разработка и администрирование и защита баз данных: учебник для студ. учреждений сред. проф. образования. –М.: Издательский центр «Академия», 2018.- 288с
6. 1С-Старт [Электронный ресурс]; Предельная величина базы для начисления страховых взносов: новые лимиты в 2019 и 2020 году; авт. 1С-Старт;2021; – Режим доступа: https://www.regberry.ru/ , свободный. Загл. с экрана – Яз. рус.
7. Ip-on-line.ru [Электронный ресурс]; Порядок начисления и выплаты зарплаты – Режим доступа https://ip-on-line.ru/kadry/poryadok-nachisleniya-i-vyplaty-zarplaty.html, свободный. Загл. с экрана – Яз. рус.
8. subsidii.net [Электронный ресурс]: Оплата больничного листа в 2021 году: сроки выплаты и размер процентов. – Режим доступа: https://subsidii.net/ , свободный. Загл. с экрана – Яз. рус.
9. Ассистентус [Электронный ресурс]; Расчетный листок по заработной плате – Режим доступа https://assistentus.ru/forma/raschetnyj-listok/, свободный. Загл. с экрана – Яз. рус.
10. Богданова А.Л. Базы данных. Теория и практика применения (2-е издание) [Электронный ресурс]: учебное пособие/ Богданова А.Л., Дмитриев Г.П.,

Медников А.В.— Электрон. текстовые данные.— Химки: Российская международная академия туризма, 2016.— 128 с.— Режим доступа: http://www.iprbookshop.ru/47625.— ЭБС «IPRbooks», по паролю

11. ЗАРПЛАТА/Практический журнал для бухгалтеров для расчета зарплаты [Электронный ресурс]: Выплата больничного: порядок и новые сроки в 2021 году; – Режим доступа:https://www.zarplata-online.ru/ , свободный. Загл. с экрана – Яз. рус.

12. КонсультантПлюс / надежная правовая поддержка [Электронный ресурс] ; – Режим доступа: http://www.consultant.ru, свободный. Загл. с экрана – Яз. рус.

13. Контур [Электронный ресурс]: Журнал / МРОТ — 2021: изменения; авт. Марина Крицкая Режим доступа: https://kontur.ru/ , свободный. Загл. с экрана – Яз. рус.

14. Контур.Школа [Электронный ресурс] : Расчет и оплата больничного листа в 2021 году; авт. Бусыгина Ю. О. . – Режим доступа: https://school.kontur.ru/, свободный. Загл. с экрана – Яз. рус.

15. Култыгин О.П. Администрирование баз данных. СУБД MS SQL Server [Электронный ресурс]: учебное пособие/ Култыгин О.П.— Электрон. текстовые данные.— М.: Московский финансово-промышленный университет «Синергия», 2016.— 232 с.— Режим доступа: http://www.iprbookshop.ru/17009.— ЭБС «IPRbooks», по паролю

16. НАЛОГ-НАЛОГ.ру [Электронный ресурс]; Облагается ли больничный лист (больничный) НДФЛ?;авт. Степанова Наталья.- – Режим доступа: https://nalog-nalog.ru/ , свободный. Загл. с экрана – Яз. рус.

17. Основы современных баз данных [Электронный ресурс]: методическая разработка к выполнению лабораторных работ (№1-3)/ — Электрон. текстовые данные.— Липецк: Липецкий государственный технический университет, ЭБС АСВ, 2016.— 37 с.— Режим доступа: http://www.iprbookshop.ru/22906.— ЭБС «IPRbooks», по паролю

18. Туманов В.Е. Основы проектирования реляционных баз данных [Электронный ресурс]/ Туманов В.Е.— Электрон. текстовые данные.— М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.— 502 с.— Режим доступа: http://www.iprbookshop.ru/22431.— ЭБС «IPRbooks», по паролю

19. Упрощенка [Электронный ресурс]: Оплата и расчет больничного листа в 2021 году: изменения и новые правила;2021; – Режим доступа: https://www.26-2.ru/ , свободный. Загл. с экрана – Яз. рус.

20. Швецов В.И. Базы данных [Электронный ресурс]/ Швецов В.И.— Электрон. текстовые данные.— М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.— 218 с.— Режим доступа: http://www.iprbookshop.ru/16688.— ЭБС «IPRbooks», по паролю

21. https://www.klerk.ru/buh/articles/506743/