**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

Факультет прикладной математики и информатики

**Отчет по ППС**

Система АИС «Система начисления зарплаты»

студента 2 курса 13 группы

Никончика Даниила Викторовича

**Преподаватель**

Зенько Татьяна Алексеевна

Минск 2021

Оглавление

[Глава 1. Постановка задачи 4](#_Toc72939095)

[Глава 2. Определение требований 6](#_Toc72939096)

[Глава 3. Анализ вариантов использования 16](#_Toc72939097)

[Глава 4. Проектирование архитектуры системы 23](#_Toc72939098)

[Глава 5. Проектирование элементов системы 26](#_Toc72939099)

[Глава 6. Разработка приложения 29](#_Toc72939100)

[Глава 7. Результаты тестирования системы 37](#_Toc72939101)

# Глава 1. Постановка задачи

АИС «Система начисления зарплаты»

Перед информационной службой компании поставлена задача создания новой системы начисления зарплаты взамен морально устаревшей существующей системы. Новая система должна предоставлять служащим возможность записывать электронным способом информацию по учету рабочего времени и автоматически формировать чеки на оплату, учитывающие количество отработанных часов и общий объем продаж (для служащих, получающих комиссионное вознаграждение).

Новая система должна предоставлять служащим возможность ежедневно вводить информацию об отработанном времени, вводить заказы на поставку, изменять свои параметры (такие, как способ оплаты за работу), и формировать различные отчеты. Система должна работать на персональных компьютерах служащих всей компании. В целях обеспечения безопасности и аудита служащие должны иметь возможность доступа и редактирования только своих данных, доступ к чужим данным должен быть запрещен.

В системе должна храниться информация обо всех служащих компании в различных странах. Система должна обеспечивать правильную и своевременную оплату работы каждого служащего в соответствии с указанным им способом. Компания из соображений экономии расходов желает сохранить без изменений одну из существующих баз данных, которая содержит всю информацию относительно заказов, должностных месячных окладов и тарифов. Новая система может читать из нее данные, но не может обновлять их.

Некоторые служащие получают почасовую зарплату. Она начисляется на основе записей по учету рабочего времени, каждая из которых содержит дату и количество часов, отработанных в соответствии с конкретным тарифом. Если какой-либо служащий отработал в день более 8 часов, 12 сверхурочное время оплачивается с коэффициентом 1.5. Служащие-почасовики получают зарплату каждую пятницу.

Некоторые служащие получают фиксированный оклад. Однако система должна вести учет количества отработанных ими часов, чтобы снижать выплату на 5%, если служащий отработал менее 40 часов в неделю. Сверхурочные служащим с фиксированным окладом оплачиваются на тех же основаниях, что и почасовикам. Расчет и выплата служащим с окладом осуществляется в последний рабочий день месяца.

Некоторые из служащих с фиксированным окладом также получают комиссионное вознаграждение, учитывающее объем продаж. Помимо отработанного времени они указывают в системе заказы, по которым подсчитывается объем продаж за месяц. Процент комиссионного вознаграждения определяется индивидуально для каждого служащего и может составлять 10%, 15%, 25% или 35%.

Служащие ежегодно уходят в отпуск на 28 рабочих дней. Служащий может проинформировать систему об уходе в отпуск заблаговременно (срок, оставшийся до ухода в отпуск, не может быть меньше 60 дней). Отпускные выплачиваются служащему авансом и составляют среднее из выплат за 3 месяца, предшествующих отпуску.

Одной из наиболее часто используемых возможностей новой системы является формирование различных отчетов: запросить количество отработанных часов, суммарную зарплату, оставшееся время отпуска и т.д. Служащие могут выбирать способ оплаты за работу. Они могут получать свои чеки на оплату по почте, на счет в банке или на руки в офисе.

Администратор системы ведет информацию о служащих. В его обязанности входит ввод данных о новых служащих, удаление данных и изменение любой информации о служащем, такой, как имя, адрес и способ оплаты, а также формирование различных отчетов для руководства.

Приложение начисления зарплаты запускается автоматически каждую пятницу и в последний рабочий день месяца, рассчитывая в эти дни зарплату соответствующих служащих. Начисление зарплаты должно производиться автоматически, без ручного вмешательства.

В ходе выполнения этого варианта задания должна быть разработана схема базы данных по учету рабочего времени, отпускам и выплатам.

# Глава 2. Определение требований

**Назначения и цели создания**

Система создаётся с целью начисления зарплат, сверхурочных и отпускных выплат служащим компании. Система должна предоставлять весь необходимый функционал для учета данных служащих, заказов с тарифами и отработанных часов.

**Требования к системе**

Необходимо хранить информацию о служащих, количестве отработанных часов, времени отпусков и тарифах.

**Глоссарий**

|  |  |
| --- | --- |
| Фиксированный оклад | фиксированная сумма вознаграждения труда из расчета, что сотрудник работал весь календарный месяц (не брал больничный, отпуск) |
| Почасовая оплата труда | один из вариантов повременной системы оплаты труда, при которой заработная плата рассчитывается исходя из фактически отработанных работником часов. |
| Администратор | должностное лицо, управляющее в учреждении, коллективе, компании. |
| Сверхурочные часы | время, отработанное работником по распоряжению нанимателя сверх установленной для него нормы продолжительности рабочего времени |
| Отпускные выплаты | Средняя заработная плата , выплачиваемая за отпускное время |

**Дополнительная спецификация**

1. **Функциональные возможности**

Начисление зарплаты происходит автоматически в соответствии с количеством отработанных часов. Служащие с фиксированным окладом в конце месяца могут получить сверхурочные в размере 10%, 15%, 25% или 35%. Любой служащий способен самостоятельно поменять способ оплаты (чек на почту, начисление на счет в банке или выплата на руки в офисе) и время выхода в отпуск на 28 дней.

Администратор имеет возможность создавать отчеты различной тематики и выполнять CRUD данных служащих.

1. **Требования по реализации**

Система должна быть совместима с Windows.

1. **Надежность**

Система должна быть в работоспособном состоянии 24 часа в день 7 дней в неделю.

1. **Производительность**

Система должна поддерживать возможность одновременного доступа к ней всех служащих фирмы.

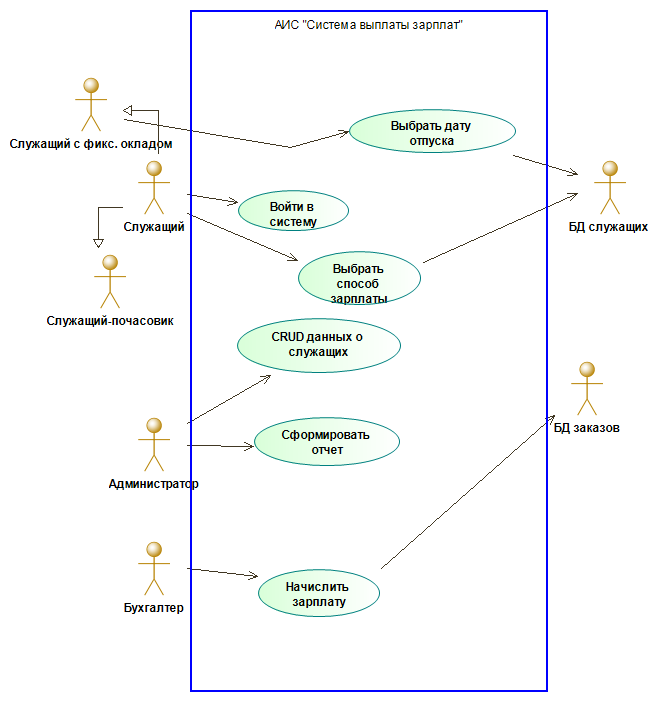
1. **Безопасность**

Система не должна позволять личные данные служащих всем, кроме администратора. Должна перепроверять правильность указанного числа отработанных часов и времени выхода в отпуск.

1. **Проектные ограничения**

Отсутствуют.

**Диаграмма вариантов использования**



**Список действующих лиц и их интересов:**

|  |  |
| --- | --- |
| Администратор | вносит изменения о служащих, создает отчеты |
| Служащий | выбирает способ оплаты и дату отпуска |
| БД служащих | Хранит данные о количестве отработанных часов каждым служащим, способе оплаты и дате отпуска |
| Бухгалтер | Начисляет зарплату служащим |

**Спецификации**

|  |  |
| --- | --- |
| Выбрать способ оплаты | описывает способ выбора оплаты служащими |
| Войти в систему | описывает вариант входа в систему служащим |
| Выбрать дату отпуска | описывает вариант выбора даты отпуска служащим |
| Сформировать отчет | описывает вариант формирования администратор различных отчетов |

**Вариант использования «Выбрать способ оплаты»:**

**Краткое описание**

Данный вариант использования описывает способ выбора оплаты служащими

**Основной поток событий**

1. Система отправляют пользователю список возможных способов оплаты.
2. Служащий отмечает наиболее подходящий ему способ.
3. Система сохраняет выбор пользователя в БД.
4. Система выводит сообщение об успешном сохранении способа оплаты.

**Альтернативные потоки**

2А Если пользователь не выберет ни одного способа и захочет уйти со страницы выбора, система автоматически выбирает значение по-умолчанию («получить на руки в бухгалтерии») и уведомляет пользователя.

**Предусловия**

Авторизоваться в системе.

**Постусловия**

Изменения сохраняются в БД.

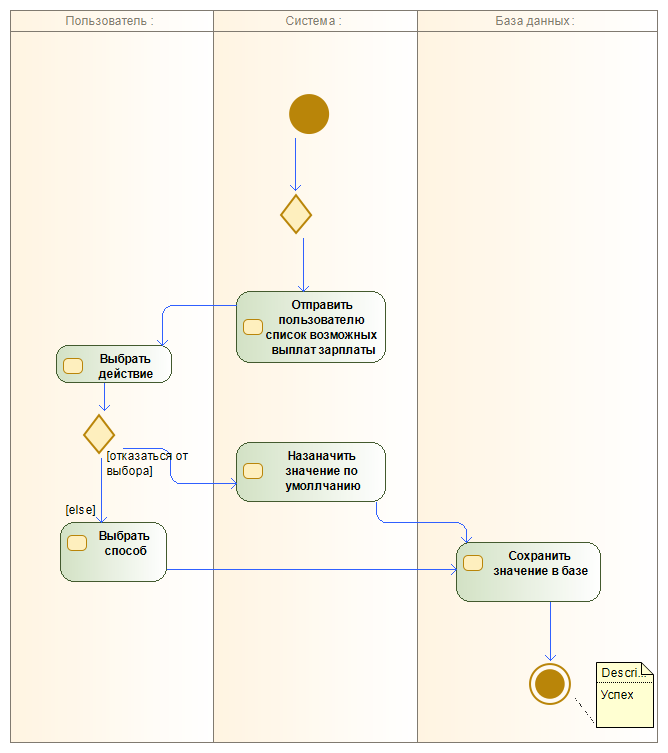


Рисунок 1 Диаграмма деятельности

**Вариант использования «Войти в систему»:**

**Краткое описание**

Данный вариант использования описывает вариант входа в систему служащим.

**Основной поток событий**

1. Система спрашивает зарегистрирован ли уже пользователь.
2. Система отправляет запрос пользователю на ввод логина
3. Пользователь вводит логин
4. Система проверяет, существует ли запись с таким логином в базе данных
5. Система отправляет запрос на ввод пароля
6. Пользователь вводит пароль
7. Система проверяет соответствие пароля
8. Система авторизует пользователя и разрешает пользоваться своими возможностями

**Альтернативные потоки**

1А Если пользователь не был зарегистрирован, он будет перенаправоен в окно «Зарегистрироваться в системе»

4А. Если системе не удалось найти переданный логин, то служащий переводится на экран ошибки.

(3,6)А Если пользователь не может вспомнить свой логин/ пароль, система перенаправляет его на экран «Восстановить логин/пароль»

(3,6)Б Если служащий ввел данные 3 раза неправильно, предложить ему функции «Восстановить логин/пароль»

7А. Если пароль не соответствует паролю, сохраненному в БД для такого логина, то служащий переводится на экран ошибки.

**Предусловия**

Отсутствуют.

**Постусловия**

Предложить пользователю сохранить пароль после входа в систему.

**Вариант использования «Выбрать дату отпуска»:**

**Краткое описание**

Данный вариант использования описывает вариант выбора даты отпуска служащим.

**Основной поток событий**

1. Система проверяет, выходил ли служащий в отпуск в текущем году
2. Система отправляет пользователю календарь, с возможными периодами времени выхода в отпуск
3. Пользователь выбирает дату
4. Система отправляет запрос администратору
5. Администратор проверяет , чтобы в отделе не осталось менее 25 % служащих.
6. Система уведомляет, что служащий может уйти в отпуск в текущий период.

**Альтернативные потоки**

1А В случае если служащий уже выходил в отпуск в текущем году, система блокирует запрос.

3А Пользователь может отказаться от выбора и уйти из текущего окна.

5А

1 Если слишком много служащих уходит в отпуск в этот период, администратор отправляет ответ системе об этом.

2 Система отправляет пользователю сообщение, чтобы последний обратился к главе своего отдела.

**Предусловия**

Авторизоваться в системе

**Постусловия**

Данные заносятся в БД.

Если пользователь отказался от отпуска, предложить ему отправиться в бухгалтерию за выплатой отпуска.

**Вариант использования «Сформировать отчет»:**

**Краткое описание**

Данный вариант использования описывает вариант формирования администратор различных отчетов.

**Основной поток событий**

1. Система проверяет, что именно администратор хочет выполнить отчет.
2. Система отправляет список возможных видов отчета
3. Администратор выбирает вид отчета из списка.
4. Система формирует отчет и возвращает его администратору

**Альтернативные потоки**

1А Запрос отменяется, если отчет просит сформировать не администратор

**Предусловия**

Авторизоваться в системе в качестве администратора.

**Постусловия**

Предложить выбрать еще один отчет.

**Вариант использования «Начислить зарплату»:**

**Краткое описание**

Данный вариант использования описывает вариант начисления зарплаты служащим.

**Основной поток событий**

1. Система проверяет служащий-почасовик или нет.
2. Система проверяет сколько часов отработал служащий за неделю
3. Высчитывается заработная плата согласно тарифам из БД заказов

**Альтернативные потоки**

2А.

1. Система проверяет, отработал ли служащий положенное количество часов.

2. В случае недоработки, положенная заработная плата снижается на 5%.

**Предусловия**

Авторизоваться в системе.

**Постусловия**

В базу вносится размер выданной зарплаты.

# Глава 3. Анализ вариантов использования

**Диаграмма последовательности:**

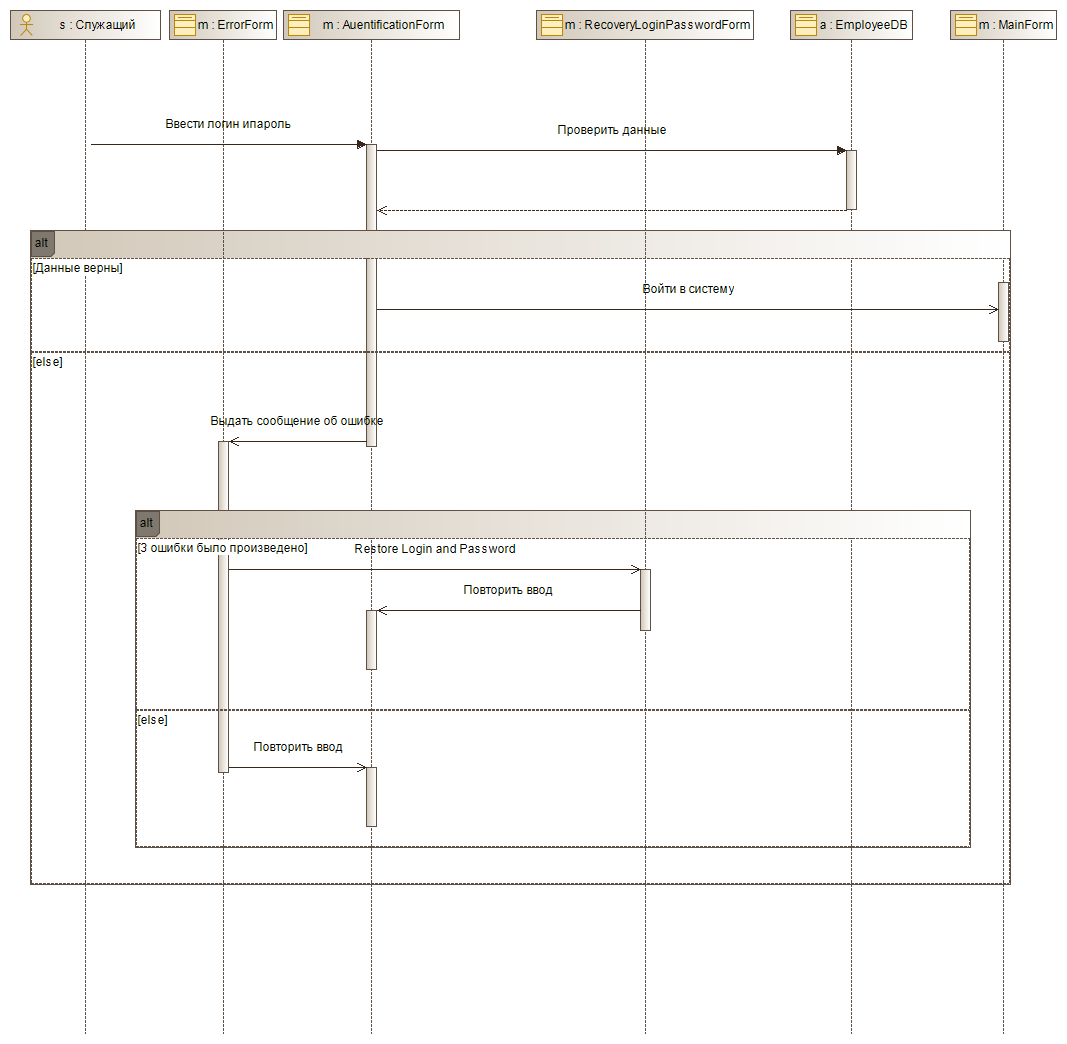


Рисунок 2 Диаграмма ауентификации

Краткое описание

Данный вариант использования описывает вариант входа в систему служащим.

Основной поток событий

1. Система спрашивает зарегистрирован ли уже пользователь.

2. Система отправляет запрос пользователю на ввод логина

3. Пользователь вводит логин

4. Система проверяет, существует ли запись с таким логином в базе данных

5. Система отправляет запрос на ввод пароля

6. Пользователь вводит пароль

7. Система проверяет соответствие пароля

8. Система авторизует пользователя и разрешает пользоваться своими возможностями

Альтернативные потоки

1А Если пользователь не был зарегистрирован, он будет перенаправоен в окно «Зарегистрироваться в системе»

4А. Если системе не удалось найти переданный логин, то служащий переводится на экран ошибки.

(3,6)А Если пользователь не может вспомнить свой логин/ пароль, система перенаправляет его на экран «Восстановить логин/пароль»

(3,6)Б Если служащий ввел данные 3 раза неправильно, предложить ему функции «Восстановить логин/пароль»

7А. Если пароль не соответствует паролю, сохраненному в БД для такого логина, то служащий переводится на экран ошибки.

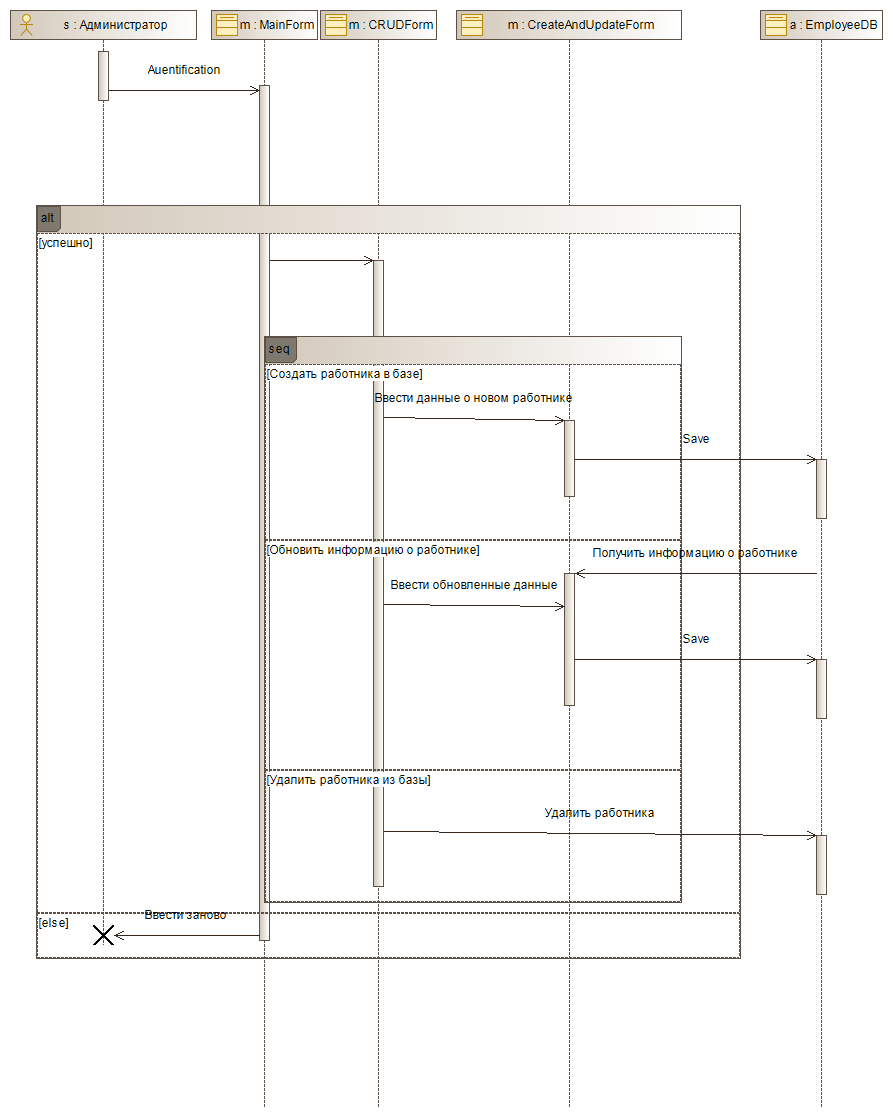


Рисунок CRUD диаграмма

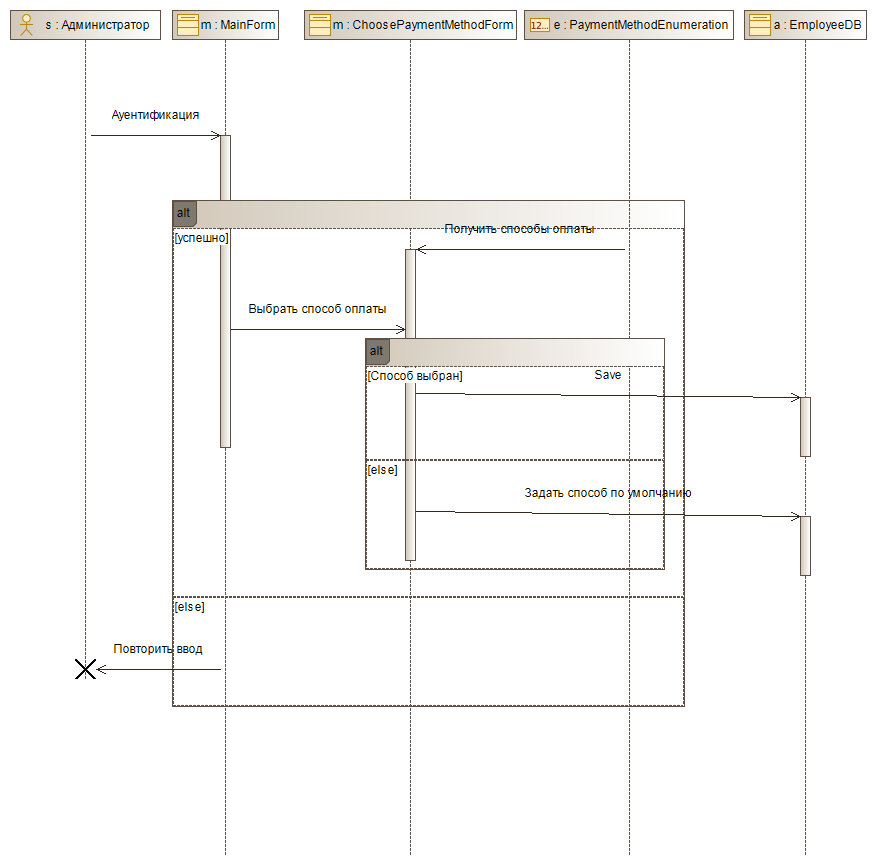


Рисунок 4 Диаграмма "Выбор способа оплаты"

Данный вариант использования описывает способ выбора оплаты служащими

Основной поток событий

1. Система отправляют пользователю список возможных способов оплаты.

2. Служащий отмечает наиболее подходящий ему способ.

3. Система сохраняет выбор пользователя в БД.

4. Система выводит сообщение об успешном сохранении способа оплаты.

Альтернативные потоки

2А Если пользователь не выберет ниодного способа и захочет уйти со страницы выбора, система автоматически выбирает значение по-умолчанию («получить на руки в бухгалтерии») и уведомляет пользователя.

**Class diagrams:**

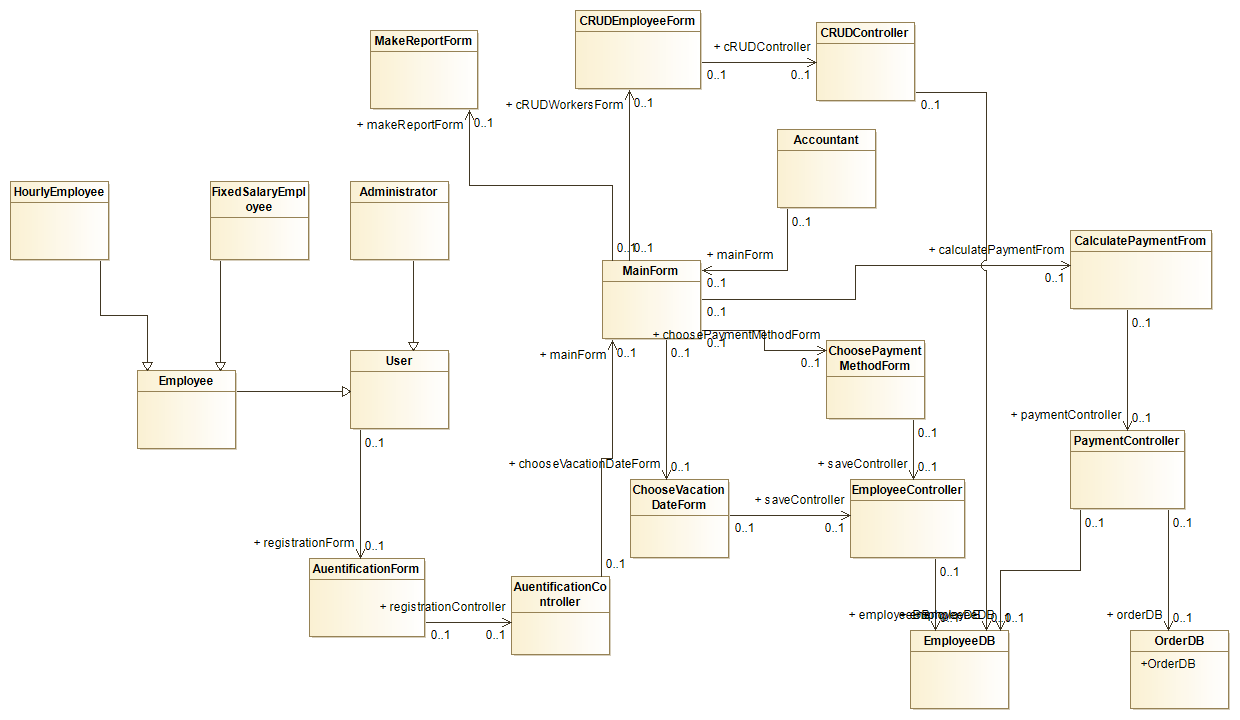


Рисунок Main class diagram

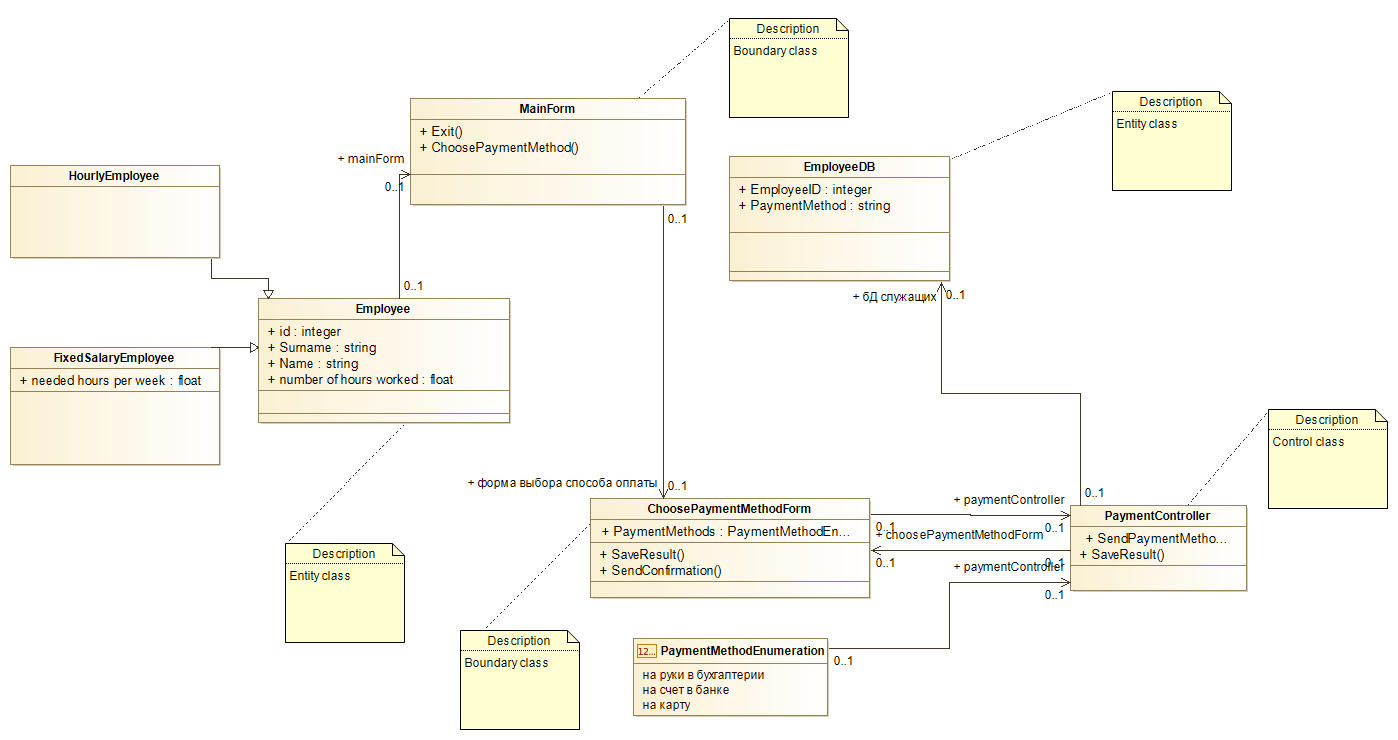


Рисунок Chhose paiment method class diagram

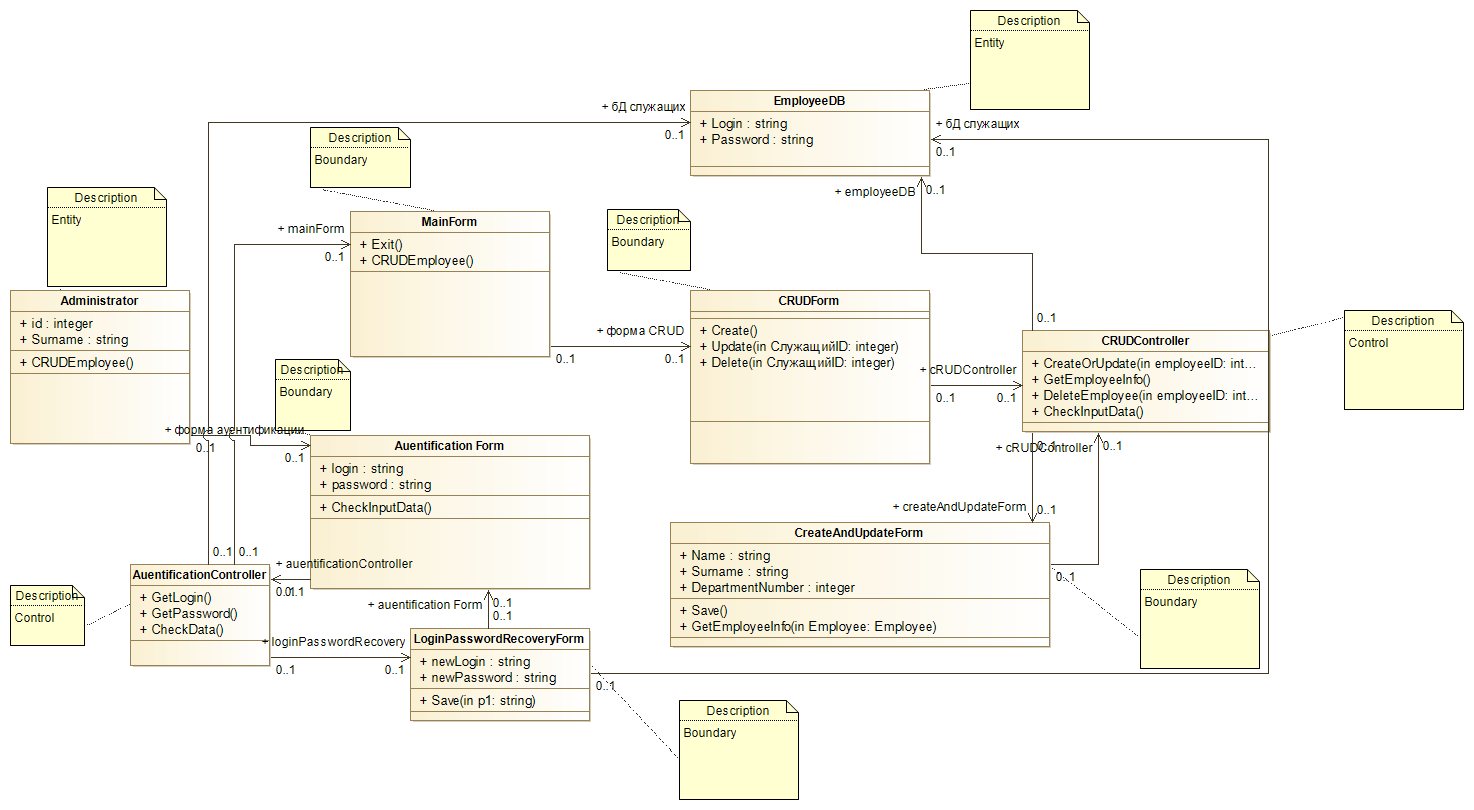


Рисунок CRUD class diagram

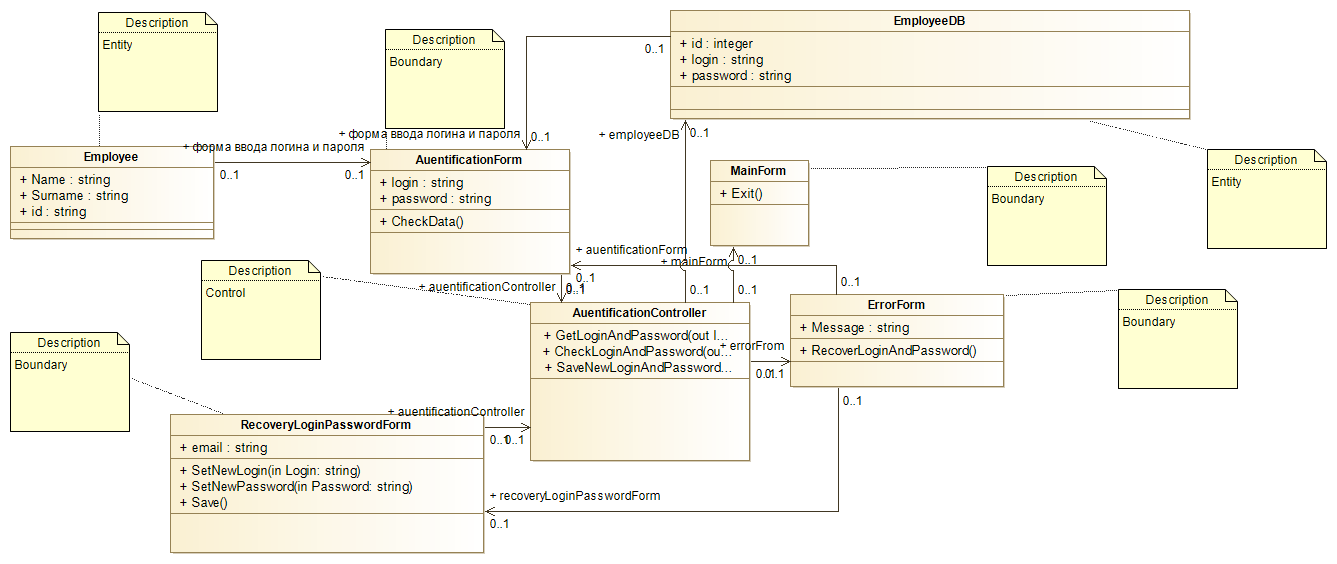
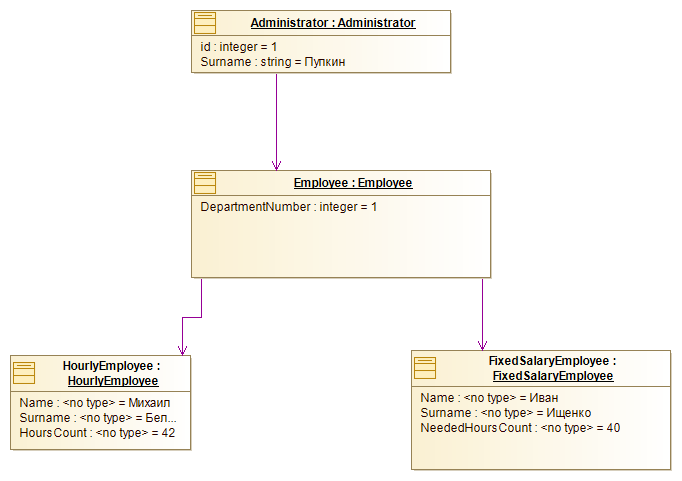


Рисунок Auentification class diagram

**Object diagram:**



# Глава 4. Проектирование архитектуры системы

**Диаграммы состояний:**

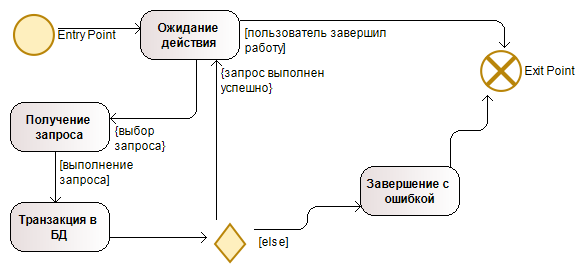


Рисунок 9 Работа с БД

Основной поток событий

* 1. Система ждет запроса от клиента
  2. Система обрабатывает выбранный запрос пользователем
  3. Система осуществляет транзакцию к БД
  4. Возвращается результат на экран пользователя
  5. Ожидание следующего действия от пользователя

Альтернативные потоки

2А) Пользователь вышел из системы

4А) Транзакция завершена с ошибкой и пользователь переводится на главный экран

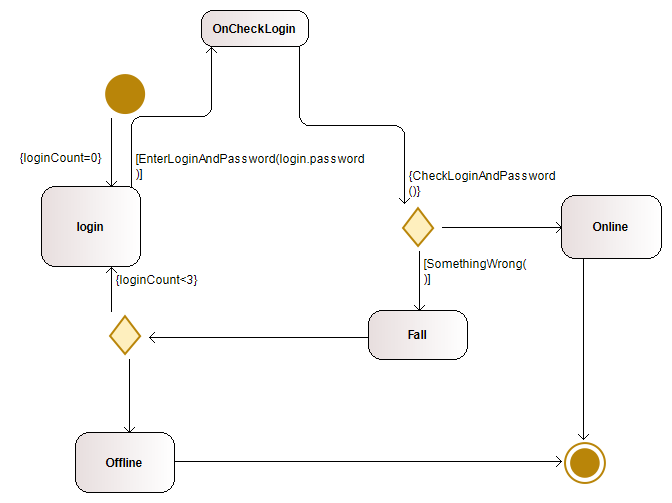


Рисунок 10 Ауентификация

Краткое описание

Данный вариант использования описывает вариант входа в систему служащим.

Основной поток событий

1. Система спрашивает зарегистрирован ли уже пользователь.

2. Система отправляет запрос пользователю на ввод логина

3. Пользователь вводит логин

4. Система проверяет, существует ли запись с таким логином в базе данных

5. Система отправляет запрос на ввод пароля

6. Пользователь вводит пароль

7. Система проверяет соответствие пароля

8. Система авторизует пользователя и разрешает пользоваться своими возможностями

Альтернативные потоки

1А Если пользователь не был зарегистрирован, он будет перенаправоен в окно «Зарегистрироваться в системе»

4А. Если системе не удалось найти переданный логин, то служащий переводится на экран ошибки.

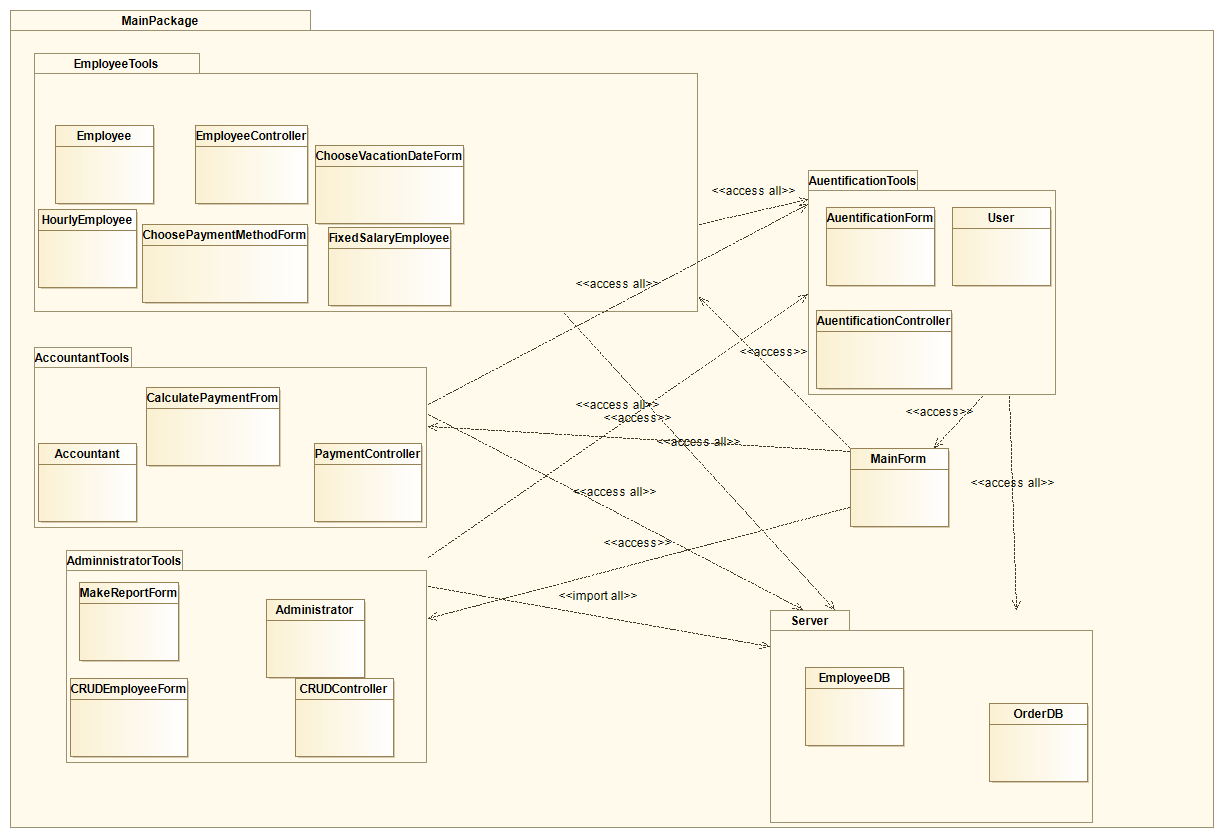
(3,6)А Если пользователь не может вспомнить свой логин/ пароль, система перенаправляет его на экран «Восстановить логин/пароль»

(3,6)Б Если служащий ввел данные 3 раза неправильно, предложить ему функции «Восстановить логин/пароль»

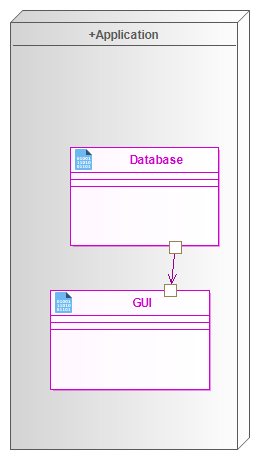
7А. Если пароль не соответствует паролю, сохраненному в БД для такого логина, то служащий переводится на экран ошибки.

# Глава 5. Проектирование элементов системы

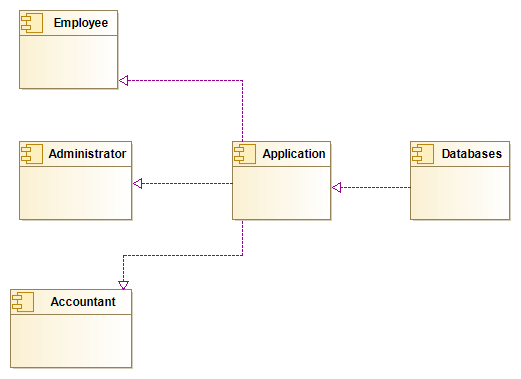
**Диаграмма пакетов**



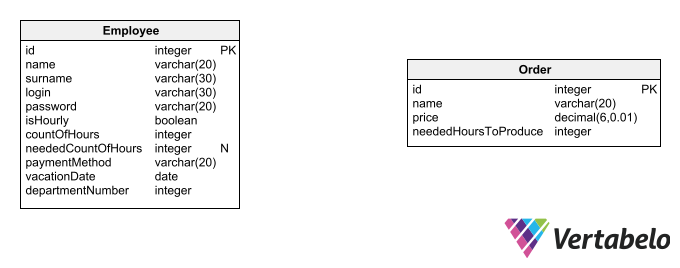
**Диаграмма развертывания**



**Диаграмма компонентов**

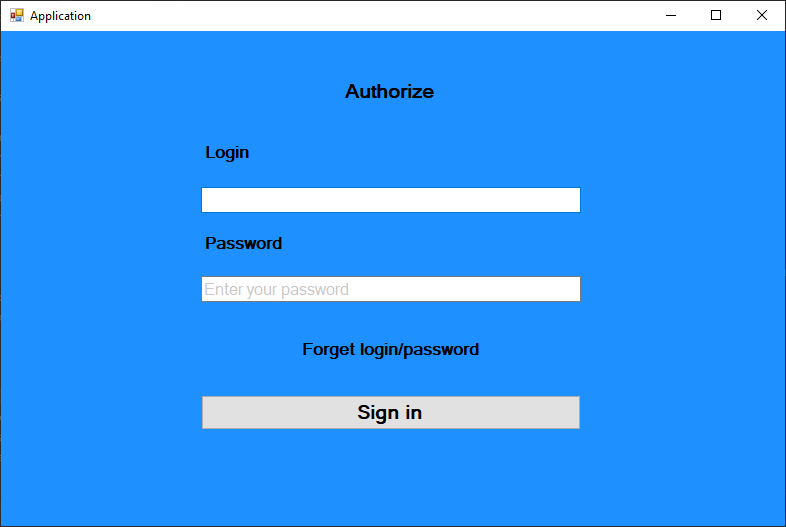


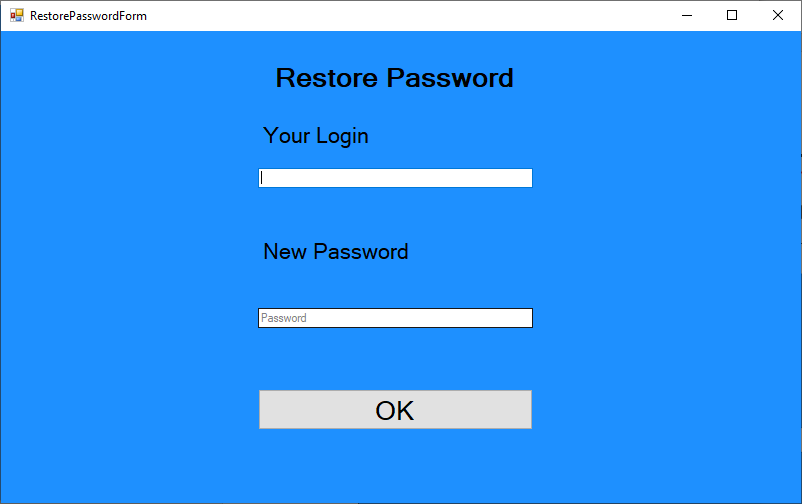
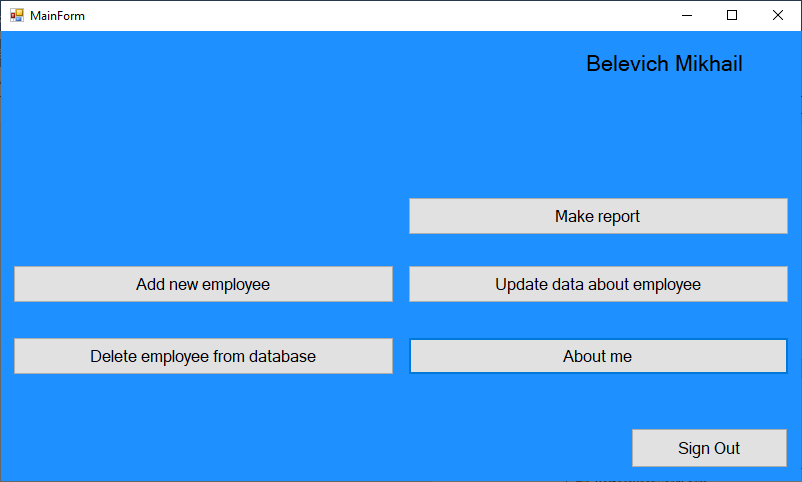
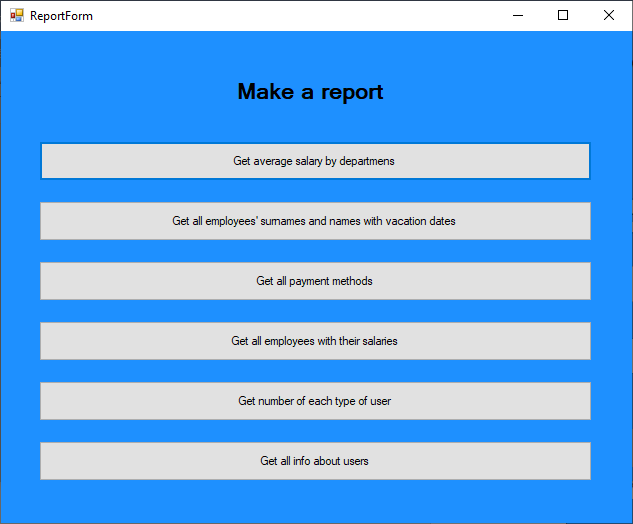
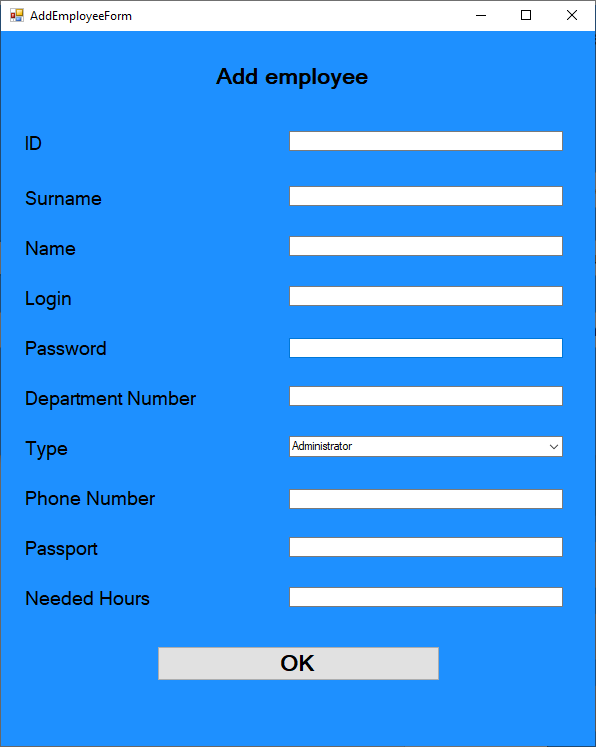
**Физическое представление базы данных**



# Глава 6. Разработка приложения

**Скриншоты приложения**



Пример сгенерированного кода:

import com.modeliosoft.modelio.javadesigner.annotations.objid;

@objid ("a5f220b9-9bb5-4b5b-94c6-91dcf76f1c26")

public class Log in to the system {

@objid ("51c9cbe7-2ed7-4c30-9253-4c3f761da630")

public String Login field;

@objid ("b9a7d3f2-82fc-48b8-88dd-9f8a8ae0295f")

public String Password field;

@objid ("5058712c-720a-42ef-a4ec-1cb4e62b9f4a")

public User user;

@objid ("d12a5e15-0f56-4016-98ff-60156a7d00e0")

public Type type;

@objid ("bd1650e1-f219-4ad9-a7b9-24d6ce60dd3c")

public Administrator staff of the educational department;

@objid ("7bf64c91-31d5-4a58-bb88-8374d2384ae0")

public System DB system DB;

@objid ("1e30aba0-9267-4c3c-8983-2db797f52eff")

public void Log in() {

}

@objid ("ea370c11-a985-4d78-9f66-ab28a436f35d")

public void Exception() {

}

}

Данные классы мне не пригодились, т.к. для приложения не понадобились такие сложные классы. Однако на основе этих классов я сделал облегченные версии на языке C#.

Код программы:

using System;  
using System.Collections.Generic;  
using System.Data.Common;  
using System.Data.SqlClient;  
using System.IO;  
using System.Linq;  
using System.Net;  
using System.Threading.Tasks;  
using System.Windows.Forms;  
using GUI.Forms;  
using GUI.Users;  
using System.Data.SQLite;  
  
namespace GUI.Controllers  
{  
 public class AuentificationController  
 {  
 private Form \_initForm;  
 private RestorePasswordForm \_restoreForm;  
 private int \_countOfErrors;  
 private SQLiteConnection sqlConn;  
  
 public void CheckLoginAndPassword(string login, string password)  
 {  
 User user = null;  
 var queryString = $"SELECT \* FROM Employees WHERE login = '{login}' AND password = '{password}'";  
 SQLiteCommand command = new SQLiteCommand(queryString, sqlConn);  
 var reader = command.ExecuteReader();  
  
   
 if (reader.Read())  
 {  
   
 var type = reader["userType"];  
 switch (type)  
 {  
 case 1:  
 {  
 user = new Administrator((int) reader["id"], (string) reader["name"],  
 (string) reader["surname"]);  
 break;  
 }  
 case 2:  
 {  
 user = new Accountant( (int) reader["id"], (string) reader["name"],  
 (string) reader["surname"]);  
 break;  
 }  
 case 3:  
 {  
 PaymentMethod method;  
 var date=reader["vacationDate"];  
 if (date == DBNull.Value)  
 date = null;  
 PaymentMethod.TryParse((string) reader["paymentMethod"], out method);  
 user = new FixedSalaryEmployee( (int) reader["id"], (string) reader["name"],  
 (string) reader["surname"], (double) reader["countOfHours"], method,  
 (DateTime?) date,   
 (double) reader["neededHours"]);  
 break;  
 }  
 case 4:  
 {  
 PaymentMethod method;  
 var date=reader["vacationDate"];  
 if (date == DBNull.Value)  
 date = null;  
 PaymentMethod.TryParse((string) reader["paymentMethod"], out method);  
 user = new HourlyEmployee((int) reader["id"], (string) reader["name"],  
 (string) reader["surname"], (double) reader["countOfHours"], method,  
 (DateTime?) date);  
 break;  
 }  
 }  
 MainForm \_mainFrom = new MainForm(user,sqlConn);  
 \_mainFrom.Show();  
 }  
 else  
 {  
 WrongLoginOrPassword();  
 }  
   
   
   
  
 *//\_initForm.Close();* }  
  
 public void InitRestorePasswordForm()  
 {  
 \_restoreForm = new RestorePasswordForm(this);  
 \_restoreForm.Show();  
 *// \_initForm.Close();* }  
  
 public void WrongLoginOrPassword()  
 {  
 MessageBox.Show("Wrong login or password");  
 \_countOfErrors++;  
 if (\_countOfErrors > 2)  
 {  
 InitRestorePasswordForm();  
 \_countOfErrors = 0;  
 }  
 else  
 {  
 \_initForm.Visible = true;  
 }  
 }  
  
 public void ChangePassword(string login, string password)  
 {  
 var queryString = $"UPDATE Employees SET password = '{password}' WHERE login = '{login}'";  
 SQLiteCommand command = new SQLiteCommand(queryString, sqlConn);  
 if (command.ExecuteNonQuery() == 0)  
 {  
 MessageBox.Show("Wrong login");  
 \_restoreForm.Visible = true;  
 }  
 else  
 {  
 \_restoreForm.Close();   
 }  
  
 }  
  
 public AuentificationController(Form initForm,SQLiteConnection sqLiteConnection)  
 {  
 \_initForm = initForm;  
 \_countOfErrors = 0;  
 sqlConn = sqLiteConnection;  
 }  
 }  
}

# Глава 7. Результаты тестирования системы

**Тесты**

**Unit Tests**

Тестируем различные метода программного кода разрабатываемой системы.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case №** | **Test Case Description** | **Test Data** | **Expected Result** |
| **1** | Тестирование метода execute (query: str) класса CRUDEmployeeController | SELECT \* FROM Employee | Возвращает список работников компании. |
| **2** | Тестирование метода execute (query: str) класса CRUDEmployeeController | SELECT \* FROM EmployeeTable | Вызывает исключение «Таблица отсутствует» |
| **3** | Тестирование метода addEmployee (query: str) класса CRUDEmployeeController | INSERT INTO Employee VALUES (“Belevich”, “Mikhail”, “mikabele”,  1923209,  24.06.2021,  “ON HANDS”,  3,  40,  33) | Всплывающее окно с подтверждением об успешном изменении базы данных |
| **4** | Тестирование метода OKButtonClicked () класса AuentificationForm | Введены верные логин и пароль (например “mikabele”, ”1923209”) | Включается главное меню |
| **5** | Тестирование метода OKButtonClicked () класса AuentificationForm | Введены неверные логин и пароль  (например “mikabele”, ”1900000”) | Выводится всплывающее окно с ошибкой о неверном вводе. При 3-х таких ошибках пользователь перенаправляется в окно смены пароля |
|  |  |  |  |

**Integration Tests**

Тестируем совместную работу компонент систему.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case №** | **Test Case Description** | **Test Data / Test Steps** | **Expected Result** |
| **1** | Тестирование подключения к базе данных | Выполнение запроса к базе данных при помощи методов классов Connector’ов | Запрос выполняется за приемлемое время либо «выбрасывается» исключение с продолжением работы приложения |
| **2** | Тестирование элементов  управления на формах | Путем нескольких нажатий на кнопки можно из любой формы перейти на любую другую | После нажатия на кнопку должен произойти переход на новую форму, на которой всегда есть кнопка «вернуться назад», либо необходимо будет произвести некоторые действия для возвращения |
|  |  |  |  |

**System Tests**

Тестируем систему на соответствие предъявленным требованиям.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case №** | **Test Case Description** | **Test Data / Test Steps** | **Expected Result** |
| **1** | Система должна иметь возможность добавлять сведения о новых сотрудниках в базу, изменять или удалять сведения об уже зарегистрированных сотрудниках | Зайти в систему от имени администратора, нажать на соответствующую кнопку в главном меню, ввести необходимые данные об служащем | Выводится всплывающее окно с подтверждением операции и изменением записей в базе данных либо окно с ошибкой |
| **2** | Система должна высчитывать зарплату каждого служащего согласно количеству отработанных часов и размеру произведенной продукции | Зайти в систему от имени бухгалтера, нажать кнопку «Вычислить зарплату»,  ввести необходимые данные о служащем | На экран выводится сообщение с заработной платой конкретного служащего |
| **3** | Система должна корректно проверять введенные пользователем логин и пароль и в зависимости от логина определять тип пользователя (администратор, служащий, бухгалтер) | Ввести по очереди логин и пароль администратор, служащего и бухгалтера | Доступ к главному меню с различными доступными возможностями (некоторые кнопки, доступные администратору, будут недоступны служащему) |

**Acceptance Tests**

Пользователи тестирует работу требуемых функций.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case №** | **Test Case Description** | **Test Data / Test Steps** | **Expected Result** |
| **1** | Администратор тестирует возможность добавления в базу данных новых сотрудников, изменения или удаления данных об уже зарегистрированных сотрудниках. | Нажать на соответствующую кнопку в главном меню, ввести необходимые данные об служащем | Выводится всплывающее окно с подтверждением операции и изменением записей в базе данных либо окно с ошибкой |
| **2** | Служащий тестирует возможность изменения даты отпуска | Нажать на кнопку “Изменить дату отпуска” в главном меню, на появившейся форме выбрать дату в соответствующем поле | Выводится всплывающее окно с подтверждением изменения строки в базе данных |

**Scenario Tests**

Тестирование различных сценариев, например, входы в систему и добавление информации о новом сотруднике.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case №** | **Test Case Description** | **Test Data / Test Steps** | **Expected Result** |
| **1** | Сценарий «Вход в систему» | Введены верные логин и пароль (например “mikabele”, ”1923209”) | Доступ к главному меню |
| **2** | Сценарий «Вход в систему» | Введены неверные логин и пароль (например “mikabele”, ”1900000”) | Выводится всплывающее окно с ошибкой о неверном вводе. При 3-х таких ошибках пользователь перенаправляется в окно смены пароля |
| **3** | Сценарий «Изменение информации о зарегистрированном сотруднике» | Введены верные данные, по которым можно найти информацию о сотруднике в базе данных | Появится форма изменения данных о сотруднике |
| **4** | Сценарий «Изменение информации о зарегистрированном сотруднике» | Введены неверные данные, по которым можно найти информацию о сотруднике в базе данных | Выводится всплывающее окно с ошибкой о том, что введены неверные данные, и пользователь перенаправляется в главное меню |