**Employee**

Представленное решение реализует систему расчёта зарплаты сотрудников. У каждого сотрудника есть свой логин, пароль и его группа, к которой он относится. У каждой группы, кроме Employees есть свои подчинённые, которые также имеют соответствующие им группы. В системе есть супер пользователь, который может просматривать данные всех сотрудников, а также добавлять в базу данных новых сотрудников.  
  
При запуске программы необходимо ввести Логин и Пароль (Рис.1)

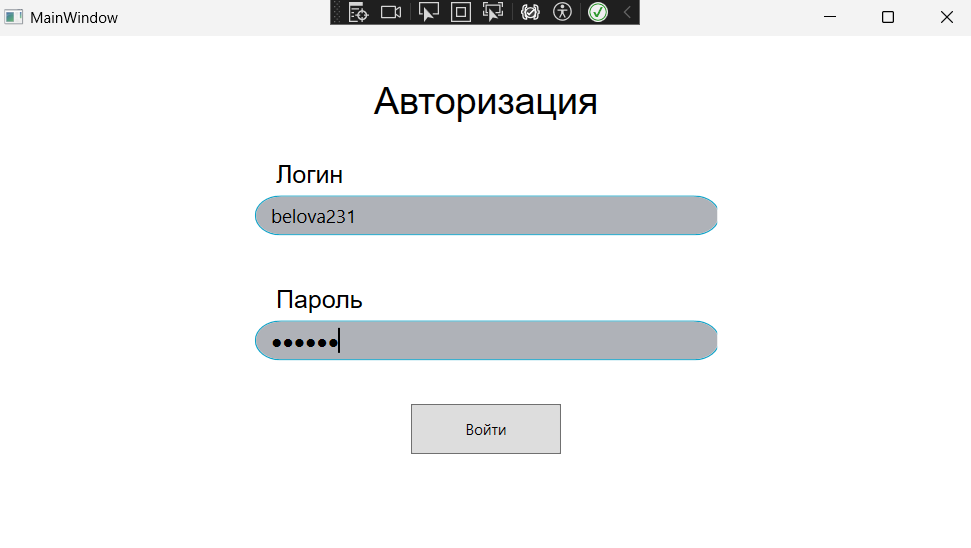


Рис.1 Окно авторизации

После ввода данных для авторизации открывается окно, где необходимо указать дату, на основе которой будет считаться конечная заработная плата, если поле для выбора даты оставить пустым, то система оповестит об этом. (Рис.2)

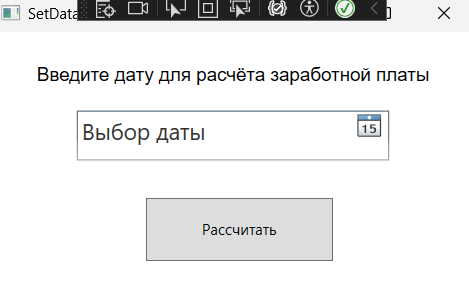


Рис.2. Окно с выбором даты

Кабинет пользователя представляет собой окно, где отображается ФИО пользователя, его конечная заработная плата и таблица с данными подчинённых (Рис.3).

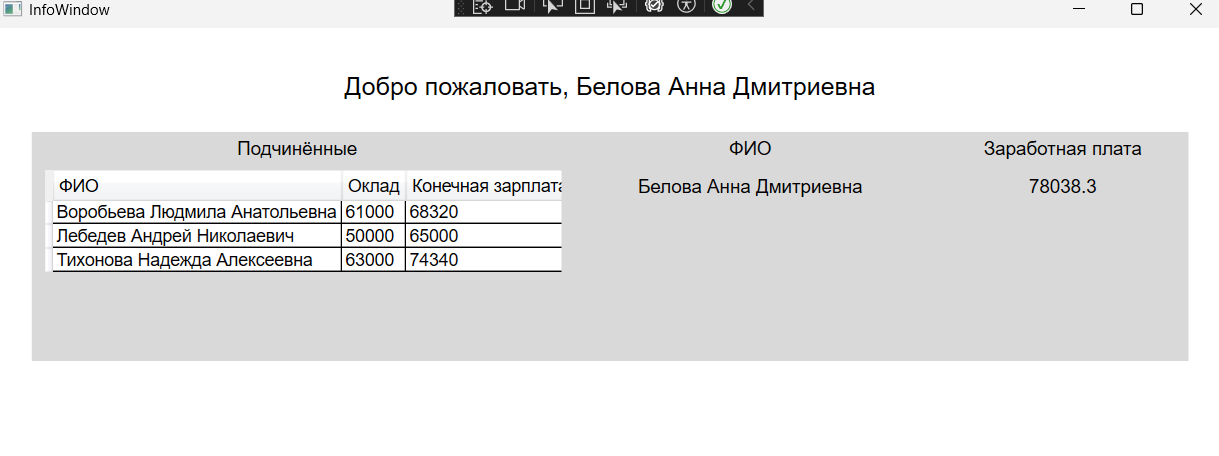


Рис. 3. кабинет пользователя

В кабинете Супер пользователя отображается информация о всех сотрудников, а также появляется кнопка для добавления пользователя (Рис.4)  
  


Рис.4 Кабинет Супер пользователя

Для добавления пользователя необходимо нажать на соответствующую кнопку: текущее окно закрывается, открывается окно для добавления пользователя (Рис.5). При указании должности система автоматически заносит данные о руководителях в соответствующий выпадающий список. После заполнения полей система проверяет на их содержания, при положительном результате пользователь будет зарегистрирован.

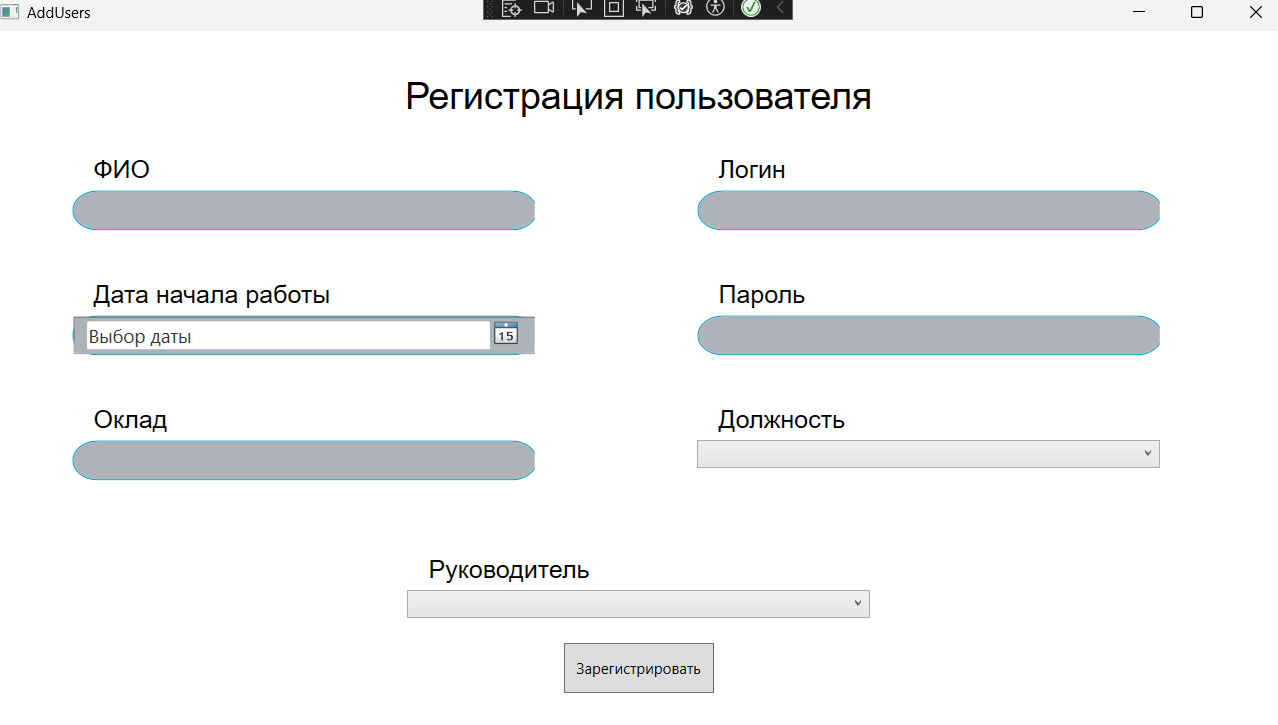


Рис. 5 Регистрация пользователя

Для решения этой задачи, первоначальные данные пользователей были внесены в файл Users.bd и реализованы структуры классов. В качестве создании интерфейсов был выбран WPF. При первом запуске программы база данных копируется в bin -> Debug для дальнейшей работы были созданы классы, каждый из них имеет определённую задачу, описание каждого класса представлено в архитектуре программы. Для взаимодействия с базой данных был использован Entity Framework 6, а также System.Data.SQLite — ADO.NET провайдер, Microsoft.Data.Sqlite — альтернативный провайдер

**Архитектура программы:**

**Интерфейс:**

* WPF (Windows Presentation Foundation) предоставляет средства разработки интерфейсов, включая язык разметки XAML, элементы управления, стили и др.
* DataGrid для отображения данных, которое позволяет сортировать таблицу
* Динамическое изменение UI под роли

**Бизнес-логика:**

* Полиморфные методы расчета зарплат
* Валидация бизнес-правил (даты, стаж)
* Иерархия подчинения классов

**Классы:**

* Класс **MainWindow** – класс представляет главное окно приложения WPF, наследуется от базового класса Window, его поля: newWind, super, db, login, password

Методы:

Next(object sender, RoutedEventArgs e) – Создаёт окно SetdateWindow, получает логин и пароль и проверяет его на пользователя, в случае Супер пользователя создаёт объект Login и пропускает окно с вводом даты

* Класс **SetdateWindow** – класс представляет окно для выбора даты, на основе которой будут дальнейшие расчёты зарплаты сотрудников. Поля класса: SelectedDate, main.

Методы:

NewInfo(object sender, RoutedEventArgs e) – Проверяет наличие даты в поле, затем создаёт объект Login, в который передаёт все необходимые параметры, затем закрывает окно.

* Класс **Info** – Представляет собой окно с информацией о пользователе.

Методы:

Adduser(object sender, RoutedEventArgs e) – Создаёт новое окно AddUser, открывает его и закрывает текущие

* Класс **ApplicationContext** – класс контекста базы данных в Entity Framework, наследуется от DbContext. Поля класса: Employees, JuniorSalesmans, SeniorManagers, JuniorManagers, JuniorSalesman, SeniorSalesman. Конструктор этого класса инициализирует подключения к базе данных, используя строку подключения из конфигурации.
* Класс **AddUser** – Окно WPF, предназначенное для добавления новых пользователей в систему. Наследуется от Window и содержит логику валидации и взаимодействия с интерфейсом, содержит поля:

Методы:

* Adduser(object sender, RoutedEventArgs e) – Закрывает текущее окно и открывает окно AddUser.
* ErrtextBox(Border Border) – Изменяет цвет заднего фона рамки.
* ExecuteQuery(string query, Dictionary<string, object> parameters) – Принимает запрос в виде строки и словарь. Выполняет запрос и добавляет параметры динамически из словаря, а также проверяет на правильность и наличие соответствующих данных в полях
* GetLastColumnName(string TableName) – принимает наименование таблицы и возвращает имя последнего столбца
* GetIdValue(string TableName) - принимает наименование таблицы и возвращает ID руководителя в зависимости от выбранного ФИО.
* AddUserInit(string TableName) - принимает наименование таблицы, формирует SQL запрос для добавления пользователя
* SetValue() – Определяет выбранную группу, на основе выбора загружается выпадающий список ФИО руководителей
* AddUserMain() – Основной метод для добавления пользователя, проверяет выбранную должность, загружает список руководителей, затем вызывает метод AddUserInit для сохранения данных
* ComboBoxDirector() – Заполняет выпадающий список данными из БД
* SetValueComboBoxDirector(object sender, SelectionChangedEventArgs e) – Отслеживание выбора в выпадающем списке, в зависимости от выбора группы, выпадающий список ФИО руководителей обновляется.
* ComboBoxDirector\_SelectionChanged(object sender, SelectionChangedEventArgs e) – повествует о выбранном руководителе
* Главный класс **User**, его поля: FIO, Login, Password, DateWork, Salary, статическое поле CurrentSelectedDate. От класса User наследуются классы, которые имеют свои поля:
* Employee: EmployeeID, JuniorManagerID.
* JuniorSalesman: JuniorSalesmanID,SeniorSalesmanID.
* SeniorManager: SeniorManagerID.
* JuniorManager: JuniorManagerID,SeniorManagerID.
* JuniorSalesman: JuniorSalesmanID, SeniorSalesmanID.
* SeniorSalesman: SeniorSalesmanID

Методы:

* SetCurrentSelectedDate() – Устанавливает выбранную дату
* LimitDate() – Проверяет корректность даты начала работы
* years() – Возвращает стаж работы в годах
* resSalary() – Рассчитывает конечную зарплату с учётом стажа и подчинённых
* SalaryrequestForJun() – Формирует SQL-запрос для расчёта зарплаты подчинённых
* FinalSalary() – Формирует запрос для расчёта заработной платы начальника
* FinalSalaryJun() – Вызывает базовый метод для расчёта зарплаты подчинённых
* JualyDay – Вспомогательный класс, который содержит преобразование даты
* Класс **SuperUser** – отдельный класс, который имеет свои поля: SuperUserID, FIO, Login, Password
* Класс **Login** – Класс описывает алгоритм авторизации пользователя, проверки вводимых данных, построения запросов в БД для вывода информации. Его методы:

1. Check\_input() – Проверяет длину логина/пароля, в зависимости от результата проверки вызывает соответствующие классы (MarkInvalid()/SetData)
2. MarkInvalid() – Отображает сообщение об ошибке и снова открывает то же окно
3. SetData() – Основной метод для аутентификации: Определяет группу пользователя, настраивает интерфейс, загружает данные, открывает окно, где располагаются данные
4. FinalSalaryData() – Рассчитывает итоговую заработную плату для пользователя с помощью вызова метода FinalSalary() определённого класса
5. SetDataTable(query) - Получает запрос и выполняет его, устанавливает данные в таблицу
6. queryRorSen() – Формирует SQL-запрос, который динамически изменяется в зависимости от пользователя. Выполнения запроса показывает подчинённых и их зарплаты
7. SetColumn() – Динамически определяет структуру таблиц БД путём возврата массива с именами таблиц
8. AdaptiveInterface() – Настаивает интерфейс, убирая ненужные для пользователя элементы управления интерфейса
9. GetTableName() – Возвращает имя таблицы по типу объекта
10. GetIDValue() – получает значения ID объекта
11. queryForSuperUser() – Формирует базовый SQL-запрос для супер пользователя

**Конфигурация:**

Файл App.config – файл предназначен для настройки работы с базой данных SQLite через Entity Framework 6 указывается:

* Настройка Entity Framework 6
* Указания версии .NET
* Строка подключение
* Регистрация провайдера SQLite

**База данных:**В качестве базы данных было использован SQLite. В базе данных есть несколько таблиц, связанных с пользователями разных ролей. (Рис. 6)

**Реализация БД:**

* Работа с базой данных SQLite через Entity Framework
* Классы-сущности с атрибутами [Table] для маппинга на таблицы БД
* Каждая хранится в отдельной таблице, но связана через Login и Password
* Прямые SQL-запросы для сложных вычислений

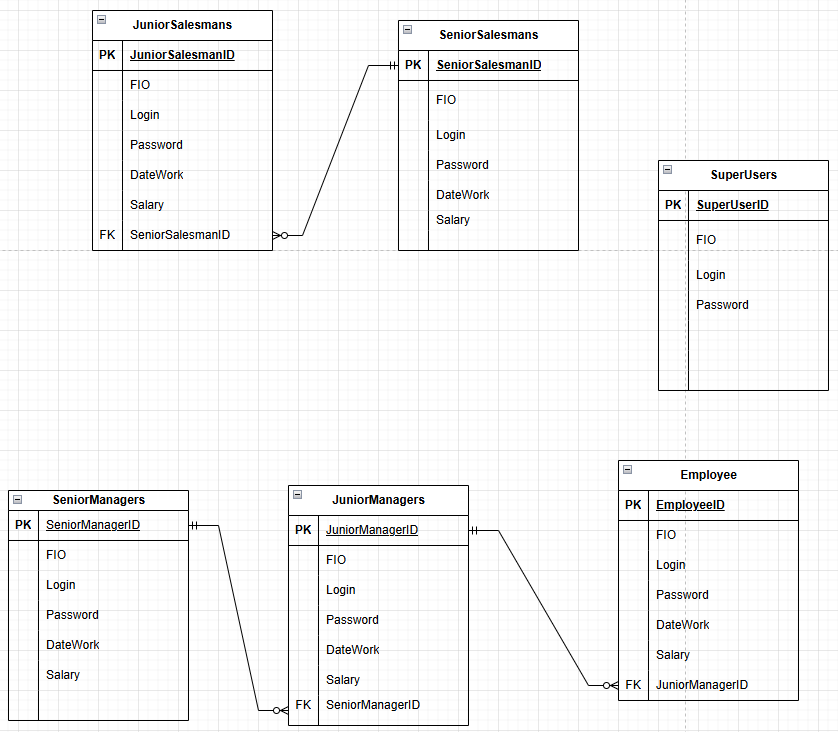


Рис.6 Физическая схема БД

**Плюсы и минусы архитектуры:**  
  
**Плюсы:**

* Класс Login – разделения действий на методы, реализация непосредственно в отдельном классе, динамический интерфейс, использование ORM – упрощённый доступ к БД. Простой способ поддержки разных типов пользователей, отображения дополнительных элементов для некоторых пользователей, обработка ошибок
* Класс User – использование ООП и его принципы, гибкость метода для расчёта заработной платы – каждая роль имеет свою логику расчёта. Использование маппинга, которые позволяет сопоставлять классы сущностей с таблицами. Динамический запрос в методе SalaryrequestForJun().
* Класс AddUser – Использование параметрических запросов, что защищает от SQL – инъекций. Легкая встраиваемость в БД. Динамически определяет структуру таблиц (GetLastcolumnName()). Валидация данных – проверка на пустые поля, неправильно заполненные поля, гибкость добавления пользователя, обновление интерфейса после успешного добавления (сброс цвета полей)
* БД - простота SQLite – не требует сервера, работает с файлом .db., Гибкость Entity Framework – ORM упрощает запросы, разделение ролей – каждая роль в своей таблице, что может упростить бизнес-логику, SQLite запускается локально.

**Минусы:**

* Класс Login - выполняет много функций, можно было разделить на несколько более специальные классы для улучшения поддержки кода. Жёсткая привязка к конкретным WPF элементам, сохранения полей в открытом виде, без хеширования.
* Класс User – Сложные запросы при расчёте заработной платы, а также строго указаны строки – проценты и т.п. Дублирование алгоритмов, строгая зависимость от базы данных.
* Класс AddUser – Реализация непосредственно в главном классе с окном и прочие недостатки, которые описаны в других классах
* БД – Пароли хранятся в открытом виде, могут быть утечки файлов, сложно модифицировать БД без использования специальных платформ. БД хранится файлом, и он должен быть всегда доступен

**Возможные улучшения:**

* Хеширование паролей.
* Использование паттерна MVVM.
* Добавить unit-тесты.
* Более грамотные разделения, реализация алгоритмов.
* Использование другой БД для расширения работы с ней.