Отчет по экзамену

Рычкин Валерий ИС2-20

Оглавление

[Жизненный цикл 2](#_Toc138804175)

[Функциональные и технические требования 3](#_Toc138804176)

[Диаграмма классов 6](#_Toc138804177)

[TO-BE 7](#_Toc138804178)

[База данных 9](#_Toc138804179)

[Интерфейс 14](#_Toc138804180)

[Инструменты разработки 16](#_Toc138804181)

[Политика безопасности и тестирование 18](#_Toc138804182)

# Жизненный цикл

Для данной работы я выбрал каскадную модель жизненного цикла ИС. Каскадная модель - модель процесса [разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F), в которой процесс разработки выглядит как поток, последовательно проходящий фазы анализа требований, проектирования, реализации, тестирования, интеграции и поддержки. Выбрана она была из-за того, что было точное ТЗ для разработки и надобности возвращаться к предыдущим пунктам разработки не было.

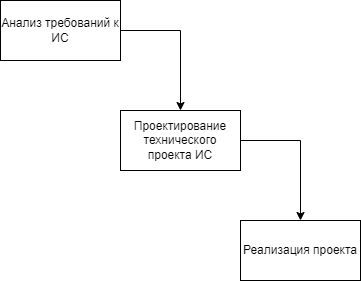


Рисунок 1 Жизненный цикл ИС

# Функциональные и технические требования

Функционально система должна уметь:

* Добавлять новые данные в базу данных,
* Редактировать данные из базы данных,
* Удалять данные из базы данных.

Требования к данным, ограничение по длине и по типу данных

| **№ п/п** | **Наименование** | **Тип данных** | **Длина** |
| --- | --- | --- | --- |
|  | Клиенты.Код | Число | 4 294 967 295 |
|  | Клиенты.Фамилия | Строка | 65 535 |
|  | Клиенты.Имя | Строка | 65 535 |
|  | Клиенты.Отчество | Строка | 65 535 |
|  | Клиенты.Пол | Строка | 65 535 |
|  | Клиенты.Дата рождения | Дата | 10 |
|  | Клиенты.Серия паспрота | Число | 4 294 967 295 |
|  | Клиенты.Номер паспорта | Число | 4 294 967 295 |
|  | Сотрудники.Код | Число | 4 294 967 295 |
|  | Сотрудники.Филиал | Число | 4 294 967 295 |
|  | Сотрудники.Фамилия | Строка | 65 535 |
|  | Сотрудники.Имя | Строка | 65 535 |
|  | Сотрудники.Отчество | Строка | 65 535 |
|  | Объекты страхования.Код | Число | 4 294 967 295 |
|  | Объекты страхования.Наименование | Строка | 65 535 |
|  | Договор.Код | Число | 4 294 967 295 |
|  | Договор.Дата заключения | Дата | 10 |
|  | Договор.Код клиента | Число | 4 294 967 295 |
|  | Договор.Код сотрудника | Число | 4 294 967 295 |
|  | Договор.Код объекта страхования | Число | 4 294 967 295 |
|  | Договор.Сумма выплаты | Число | 4 294 967 295 |
|  | Выплата страхования.Код | Число | 4 294 967 295 |
|  | Выплата страхования.Дата выплаты | Дата | 10 |
|  | Выплата страхования.Код договора | Число | 4 294 967 295 |
|  | Выплата страхования. Выплата страховки | Число | 4 294 967 295 |

Интерфейс системы должен соблюдать четкую иерархию. И важные моменты должны быть в быстром доступе.

У системы нет проблем с разрешениями экранов так как программа имеет минимальный размер окна

Для работы с системой на основе C# понадобится персональный компьютер, оснащенный следующими компонентами:

* Настольный ПК
  + Процессор: Intel Core i5-12400
  + Оперативная память: 16ГБ DDR4
  + Видеокарта: Intel UHD Graphics 730
  + Накопитель: 512 SSD
  + Корпус: DEXP Atlas
  + Блок питания Aerocool VX PLUS450
* Монитор DELL
* Компьютерная мышка Logitech
* Клавиатура Logitech

Данная конфигурация компьютера включает в себя достаточное количество оперативной памяти, обычно рекомендуется не менее 8 ГБ для удовлетворительной производительности системы C#, а также быстрый процессор с высоким тактовым числом, так как C# может быть требовательной к производительности при обработке большого количества данных. Важно также учитывать наличие достаточного объема свободного места на жестком диске для хранения баз данных PostgreSQL.

В данной системы всего один пользователь – рабочий персонал

Рабочий персонал имеет полный доступ к системе кроме редактирования исходного кода.

# Диаграмма классов

Для проектирования архитектуры информационной системы на языке C# спроектирована диаграмма классов

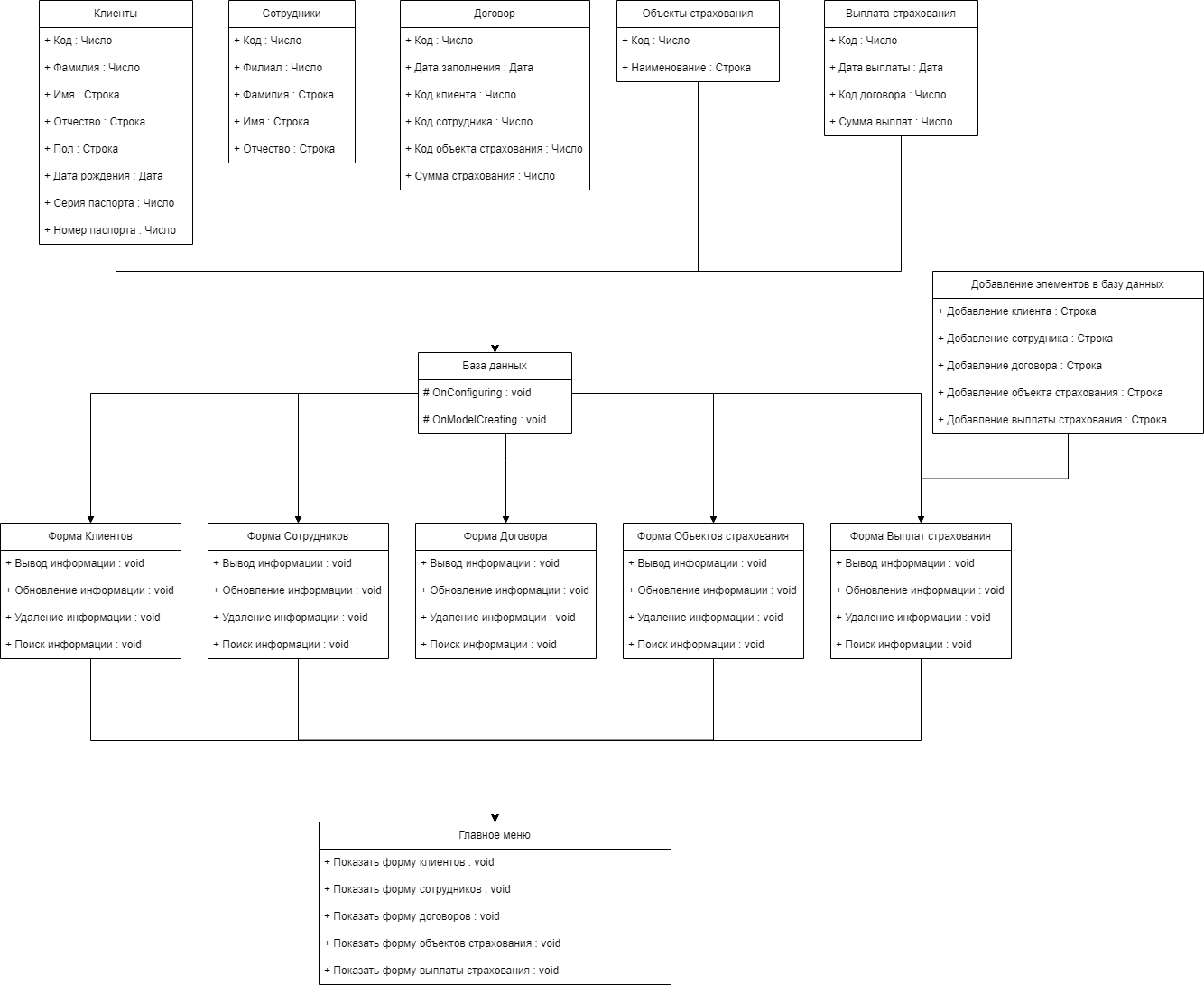


Рисунок 2 Диаграмма классов

В программе все состоит из классов, два основных класса — это «База данных» и «Главное Окно». «База данных» собирает основу базы данных из классов выше него, а «Главное Окно» позволяет открывать формы для редактирования, добавления, удаления объектов

# TO-BE

Бизнес-процесс — это серия взаимосвязанных действий, выполняемых в рамках организации, которые направлены на достижение определенной цели. Бизнес-процессы могут включать в себя любую деятельность, которая является частью бизнес-операций, включая производство товаров, оказание услуг, управление персоналом, финансовое планирование, маркетинг, продажи и т.д. Для демонстрирования часто используются модели AS-IS и TO-BE.

Моделирование TO-BE – это процесс разработки бизнес-процесса, которое должно быть оптимальным и эффективным. Оно выполняется путем анализа текущего процесса модели AS-IS. Определяются изменения, которые необходимо внести в текущий процесс, чтобы достичь более эффективного и оптимального состояния (*Рисунок 3*).

В бизнес-процессе участвуют следующие акторы: Клиент, Сотрудник, Работник за программой.

Клиент приходит с желанием получить страхование. Сотрудник обсуждает с клиентом все детали договора. После обсуждения персональные данные клиента передаются работнику за программой для внесения в базу данных. Для составления договора работник за программой запрашивает данные сотрудника и клиента. Бланк договора передается сотруднику на подписание с клиентом. После подписи договор передается работнику за программой на внесение в базу данных.

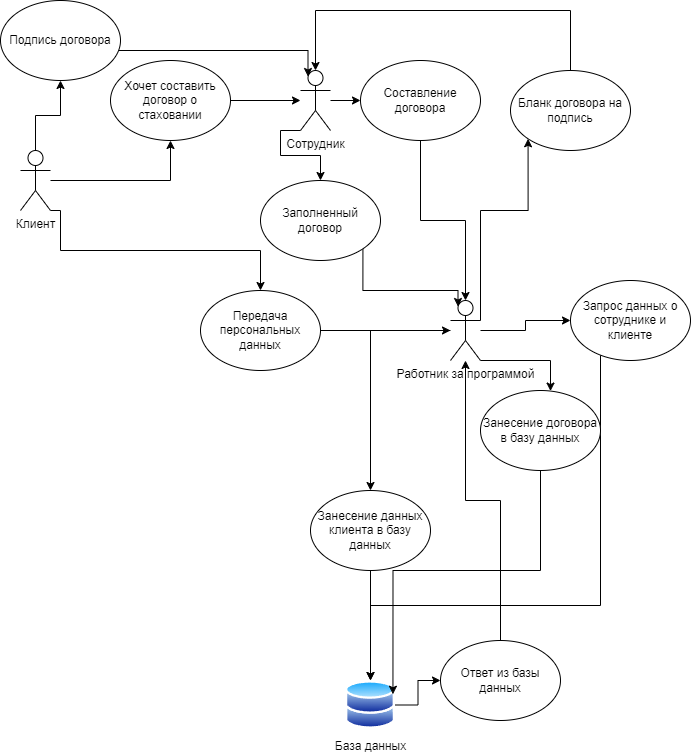


Рисунок 3 Модель TO-BE страховой компании

# База данных

Для реализации базы данных используется инфологическая модель.

Информационно-логическая модель данных является одним из этапов проектирования баз данных и служит основой для разработки физической модели данных, которая в свою очередь описывает конкретную реализацию базы данных на выбранной платформе или системе управления базами данных (СУБД). Инфологическая модель является абстракцией, которая предоставляет высокоуровневое представление данных и их отношений, и позволяет логически описать структуру и организацию данных в системе.

На модели представлены сущности, которые находятся в системе(Рисунок 4). Далее представлены все сущности и атрибуты, включенные в них:

* Клиенты
  + Код
  + Фамилия
  + Имя
  + Отчество
  + Пол
  + Дата рождения
  + Серия паспорта
  + Номер паспорта
* Сотрудники
  + Код
  + Филиал
  + Фамилия
  + Имя
  + Отчество
* Объекты страхования
  + Код
  + Наименование
* Договор
  + Код
  + Дата заключения
  + Код клиента
  + Код сотрудника
  + Код объекта страхования
  + Сумма выплаты
* Выплата страхования
  + Код
  + Дата выплаты
  + Код договора
  + Сумма выплаты страховки

У всех сущностей есть связь. Используется связь один ко многим. В базе данных связь один ко многим может быть представлена с помощью внешнего ключа, который указывает на первичный ключ связанной таблицы. Это позволяет связать несколько строк в одной таблице с одной строкой в другой таблице.

Сущность «Клиенты» связанна с «Договор» по Код клиента

Сущность «Сотрудники» связанна с «Договор» по Код сотрудника

Сущность «Объекты страхования» связанна с «Договор» по Код объекта страхования

Сущность «Договор» связанна с «Выплата страхования» по Код договора

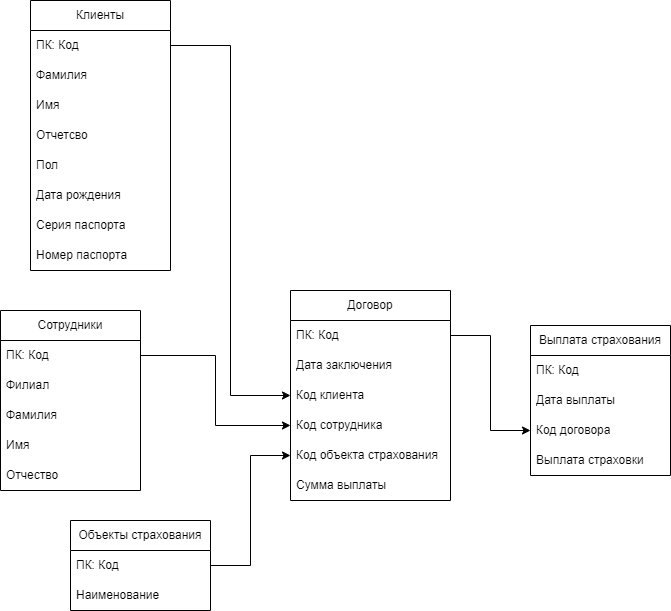


Рисунок 4 Инфологическая модель

Даталогическая модель (*Рисунок 5*) является одним из основных элементов проектирования баз данных, поскольку она помогает разработчикам понять структуру и содержание данных, которые будут храниться в базе данных. Эта модель служит основой для создания физической модели, которая определяет способ хранения данных на конкретной платформе базы данных.

Связи точно такие же, как и в Инфологической модели. Для представления типов данных используется (Таблица 1), где описаны все данные.

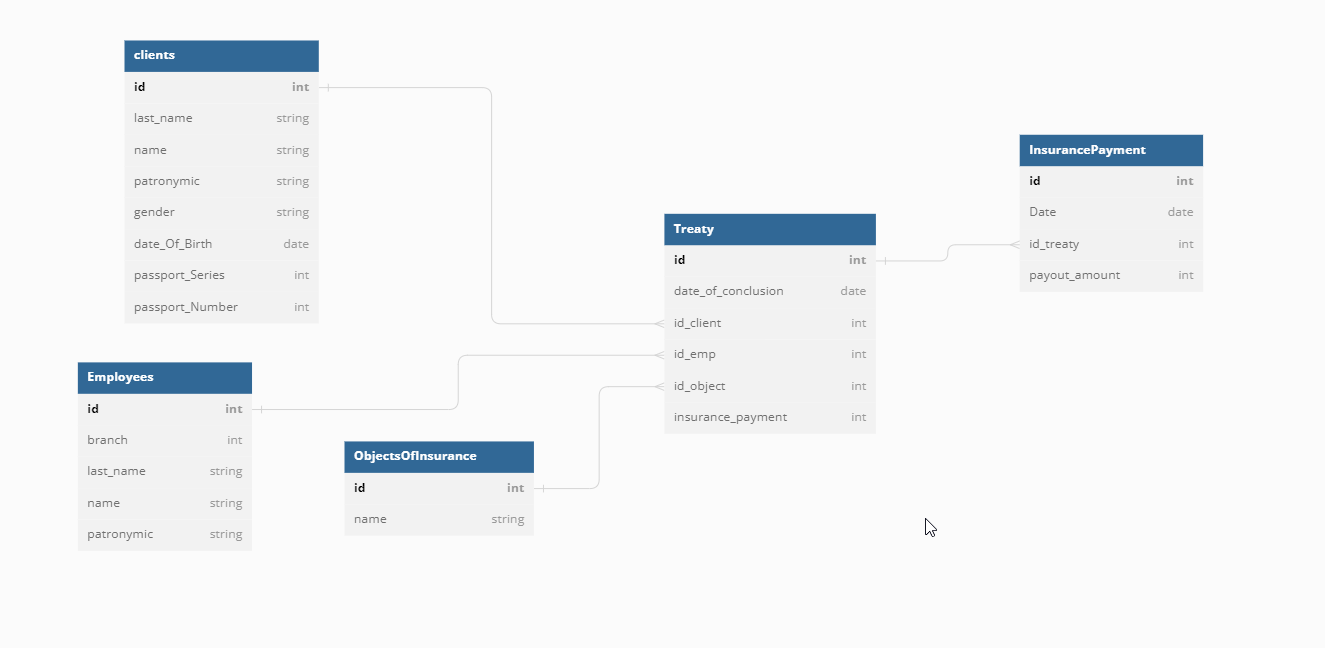


Рисунок 5 Даталогическая модель

Таблица 1 Данные для даталогической модели

| **Наименование** | **Описание** | **Тип данных** | **Ограничение** |
| --- | --- | --- | --- |
| Таблица «Employee» | | | |
| Id | Код сотрудника | Int | Primary Key |
| Branch | Филиал | Int |  |
| Last\_name | Фамилия | String |  |
| Name | Имя | String |  |
| Patronymic | Отчество | String |  |
| Таблица «Client» | | | |
| Id | Код клиента | Int | Primary Key |
| Last\_name | Фамилия | String |  |
| Name | Имя | String |  |
| Patronymic | Отчество | String |  |
| Gender | Пол | String |  |
| Date\_Of\_Birth | Дата рождения | Date |  |
| Passport\_Series | Серия паспорта | Int |  |
| Passport\_Number | Номер паспорта | Int |  |
| Таблица «ObjectsOfInsurance» | | | |
| Id | Код объекта | Int | Primary Key |
| Name | Наименование объекта | String |  |
| Таблица «Treaty» | | | |
| Id | Код договора | Int | Primary Key |
| Date\_Of\_Conclusion | Дата заключения | Date |  |
| Id\_Client | Код клиента | Int | Foreign Key (Client => Id) |
| Id\_Emp | Код сотрудника | Int | Foreign Key (Employee => Id) |
| Id\_Object | Код объекта страхования | Int | Foreign Key (ObjectsOfInsurance => Id) |
| Insurance\_Payment | Сумма выплаты | Int |  |
| Таблица «InsurancePayment» | | | |
| Id | Код выплаты | Int | Primary Key |
| Date | Дата выплаты | Date |  |
| Id\_Treaty | Код договора | Int | Foreign Key (Treaty => Id) |
| Payout\_Amount | Сумма выплаты | Int |  |

# Интерфейс

Проектирование графического интерфейса пользователя на языке C# сводится к написанию кода на языке верстки XAML. Необходимо создать удобный и интуитивно понятный интерфейс для пользователей приложения.

Несколько этапов проектирования графического интерфейса пользователя на языке C#

Красота интерфейса: Интерфейс должен быть дружелюбен к пользователю и не пугать его. Все кнопки, поля должны быть оформлены в едином стиле «Material Design». Так же немаловажным стоит отметить скругленность интерфейса. Пользователь не должен «Царапаться» об острые углы окна или кнопки.

Практичность: Этот этап требует сноровки, ведь разместить элементы так, чтобы ими было удобно и быстро пользоваться, требует знаний и фантазии. Нужно использовать по максимуму доступное место на форме программы. Но также с этим можно переборщить и перегрузить интерфейс.

Главное меню программы:

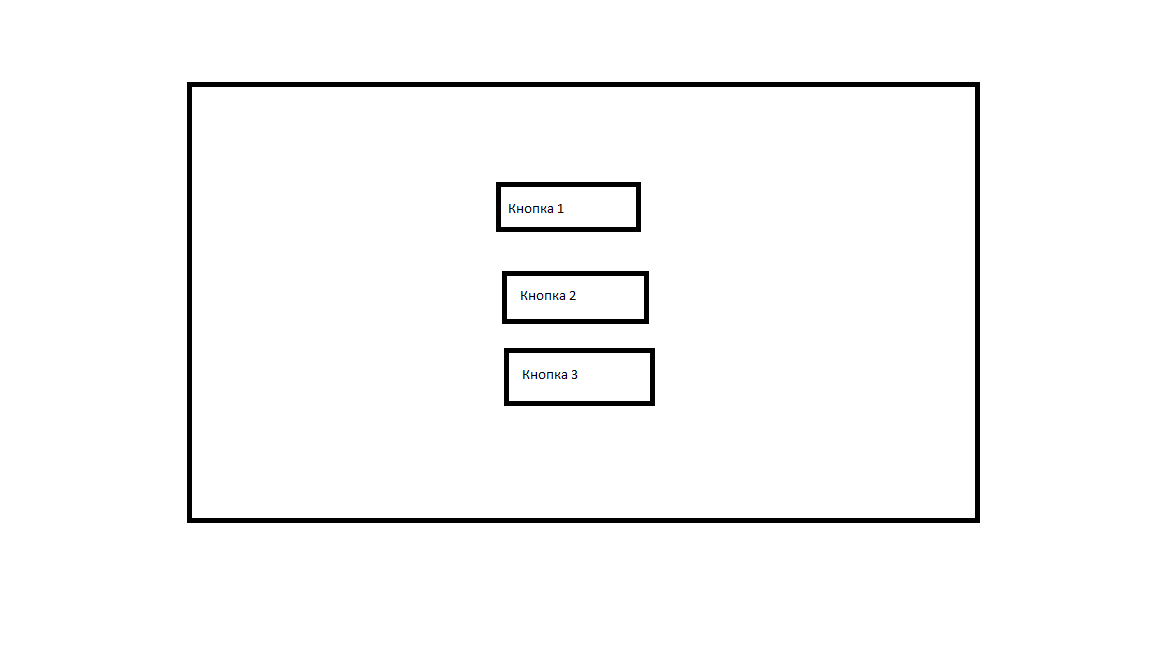


Рисунок 6 Главное меню программы

На главном меню присутствуют 5 кнопок навигации:

* Клиент – все действия с клиентами
* Сотрудник – все операции с сотрудниками
* Договор – все договоры
* Объекты страхования – вся информация об объектах страхования
* Выплата страхования – операция по выплате страхования

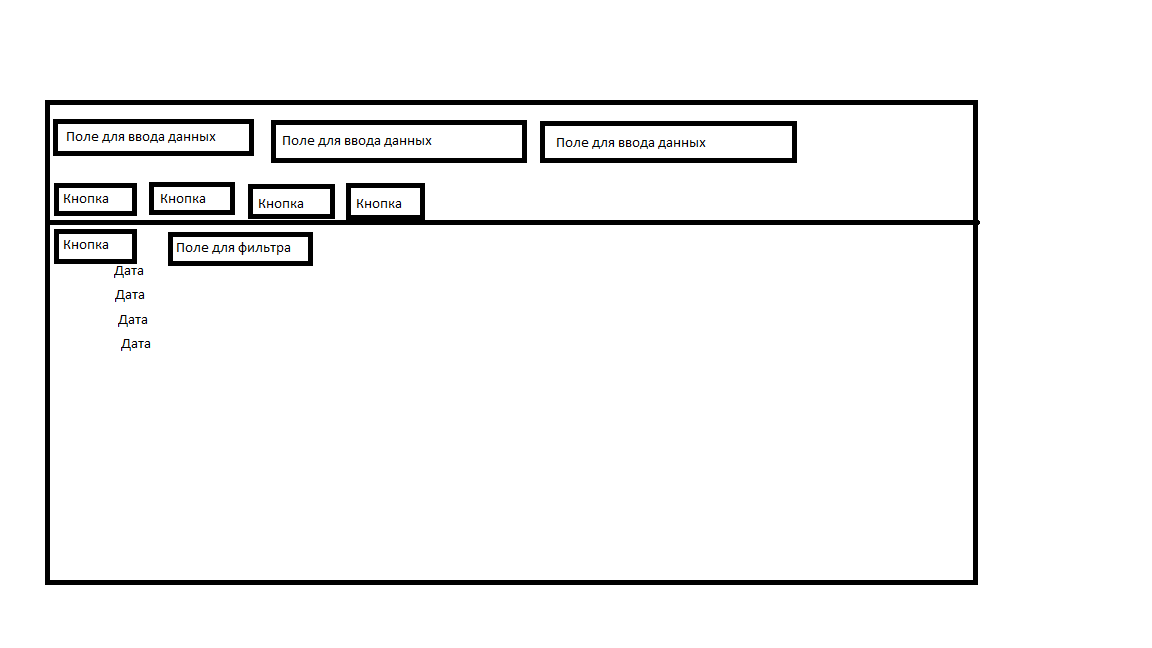


Рисунок 7. Форма взаимодействия с объектами

Вся форма поделена на две части: Часть ввода данных и просмотра этих данных.

Часть ввода данных – тут сосредоточенны все поля для ввода текста, а также кнопки для действий с этими данными.

Часть просмотра данных – тут всё внимание сосредоточенно на просмотр данных из базы данных.

# Инструменты разработки

Выбор C#, WPF и Entity Framework Core в качестве инструментов разработки информационной системы обоснован следующими причинами:

C#: C# (C-Sharp) является языком программирования, разработанным Microsoft, и широко используется для создания приложений на платформе .NET. Он обладает простым и понятным синтаксисом, хорошей производительностью и поддержкой объектно-ориентированного программирования. C# также предоставляет доступ к богатому экосистеме библиотек и фреймворков, что упрощает разработку сложных приложений.

WPF: Windows Presentation Foundation (WPF) — это технология разработки пользовательского интерфейса (UI) от Microsoft. Она предоставляет мощные средства для создания интерактивных и эстетически привлекательных графических пользовательских интерфейсов. WPF использует декларативный подход к разработке интерфейса с помощью языка разметки XAML (eXtensible Application Markup Language), что делает разработку более гибкой и легкой для поддержки сложных дизайнов.

Entity Framework Core: Entity Framework Core (EF Core) - это современный ORM (Object-Relational Mapping), предоставляемый Microsoft. Он облегчает взаимодействие с базами данных, позволяя разработчикам работать с данными в объектно-ориентированной парадигме без необходимости писать прямые запросы SQL. EF Core поддерживает различные провайдеры баз данных и предоставляет удобные инструменты для моделирования и миграции схемы базы данных.

Объединение этих инструментов позволяет создавать мощные информационные системы с удобным пользовательским интерфейсом, легким доступом к данным и удобным взаимодействием с базами данных. C# предоставляет языковую мощь и широкий выбор библиотек, WPF обеспечивает гибкую разработку интерфейса, а EF Core упрощает работу с данными и базами данных. Эти инструменты хорошо интегрируются между собой и поддерживаются Microsoft, что обеспечивает стабильность и надежность в разработке информационной системы.

# Политика безопасности и тестирование

Политика безопасности для информационной системы и базы данных страховой компании должна учитывать особенности данной отрасли и необходимость защиты конфиденциальных данных клиентов:

1. Аутентификация и авторизация:

- Все пользователи должны проходить процесс аутентификации, используя уникальные идентификаторы (логин и пароль, двухфакторную аутентификацию и т.д.).

- Реализовать строгую систему авторизации, которая определяет права доступа пользователей на основе их ролей и обязанностей в компании.

- Регулярно обновлять и пересматривать права доступа пользователей, исключая неактивных пользователей или тех, кому больше не требуется доступ.

2. Шифрование и защита данных:

- Применять сильное шифрование при передаче конфиденциальных данных через сеть, используя протоколы SSL/TLS или другие современные методы шифрования.

- Хранить конфиденциальные данные (личная информация клиентов, данные о страховых полисах) в базе данных, защищенной шифрованием на уровне хранения (полное или частичное шифрование данных).

- Защищать учетные данные и пароли пользователей с помощью хэширования или солей, чтобы предотвратить несанкционированный доступ к учетным записям.

3. Резервное копирование и восстановление:

- Регулярно создавать резервные копии базы данных и проверять их целостность.

- Хранить резервные копии в отдельном и безопасном месте, отделенном от основной базы данных.

- Тестировать процедуры восстановления, чтобы убедиться в возможности быстрого и точного восстановления данных в случае сбоев или чрезвычайных ситуаций.

4. Мониторинг и аудит:

- Установить систему мониторинга, которая будет отслеживать активности в информационной системе и базе данных, включая доступ, изменение данных и другие важные события.

- Регулярно анализировать журналы событий для обнаружения подозрительной активности, нарушений безопасности или попыток несанкционированного доступа.

-Проводить регулярные аудиты системы безопасности для оценки эффективности политик и процедур безопасности и выявления уязвимостей.

5. Физическая безопасность:

- Обеспечить физическую безопасность серверов и оборудования, содержащих базу данных, включая ограниченный доступ к помещениям, контроль доступа и видеонаблюдение.

- Ограничить физический доступ к серверам и оборудованию только авторизованным персоналом с соответствующими полномочиями.

6. Обучение и осведомленность:

- Проводить обучение персонала о политике безопасности, правилах использования системы и методах предотвращения угроз информационной безопасности.

- Повышать осведомленность сотрудников о социальной инженерии, фишинге и других методах атак, чтобы снизить риск несанкционированного доступа или утечки данных.

7. Обновление и мониторинг программного обеспечения:

- Регулярно обновлять и патчить программное обеспечение, используемое в информационной системе и базе данных, для исправления уязвимостей и поддержания безопасности.

- Установить систему мониторинга уязвимостей, чтобы обнаруживать и реагировать на новые угрозы и уязвимости программного обеспечения.

Это только общий набор мер безопасности, и реальная политика должна быть адаптирована к конкретным требованиям и угрозам страховой компании.

Тест кейс «Создание клиента»

|  |  |
| --- | --- |
| **Действие** | **Ожидаемый результат** |
| Нажать в меню на кнопку «Клиенты» | Открылась форма для работы с клиентами |
| Ввести корректно все данные клиента и нажать на кнопку «добавить» | Выведется сообщение о успешном добавлении клиента |

Тест кейс «Удаление сотрудника»

|  |  |
| --- | --- |
| **Действие** | **Ожидаемый результат** |
| Нажать в меню на кнопку «Сотрудники» | Открылась форма для работы с сотрудниками |
| Выбрать из списка сотрудника, которого нужно удалить | Данные этого сотрудника перенесутся в текстовые поля |
| Нажать на кнопку «Удалить» | Выведется сообщение об успешном удалении |

Тест кейс «Редактирование договора»

|  |  |
| --- | --- |
| **Действие** | **Ожидаемый результат** |
| Нажать в меню на кнопку «Договор» | Открылась форма для работы с договорами |
| Выбрать из списка документ, который нужно отредактировать | Данные этого договора перенесутся в текстовые поля |
| Корректно изменить данные в текстовых полях и нажать кнопку «Обновить» | Выведется сообщение о успешном обновлении данных |