Частное учреждение образования

«Колледж бизнеса и права»

Разработка программного средства для учета киберинцидентов

Пояснительная записка

к курсовому проекту по учебному предмету «Конструирование

программ и языки программирования»

КП Т.219015.401

Руководитель проекта ( Е.В. Багласова )

Обучающаяся ( А.В.Писарик )

2024

**СОДЕРЖАНИЕ**

*Изм.*

*ЛЛист*

*№ докум.*

*Подпись*

*ДДата*

*Лист*

У 3

*КП Т.219015.401 ПЗ*

*Разраб.*

Писарик А.В.

*Провер.*

Багласова Е.В.

*Т. Контр.*

*Н. Контр.*

*Утверд.*

*Разработка программного средства для учета киберинцидентов*

*Лит.*

*Листов*

57

*КБП*

[Введение 4](#_Toc170320159)

[1 Описание задачи 6](#_Toc170320160)

[1.1 Анализ предметной области 6](#_Toc170320161)

[1.2 Постановка задачи 7](#_Toc170320162)

[2 Проектирование системы 9](#_Toc170320163)

[2.1 Требования к приложению 9](#_Toc170320164)

[2.2 Проектирование модели 9](#_Toc170320165)

[2.3 Концептуальный прототип 11](#_Toc170320166)

[3 Описание реализации программного средства 15](#_Toc170320167)

[3.1 Инструменты разработки и применяемые технологии 15](#_Toc170320168)

[3.2 Организация данных 16](#_Toc170320169)

[3.3 Функции: логическая и физическая организация 18](#_Toc170320170)

[3.4 Входные и выходные данные 19](#_Toc170320171)

[3.5 Функциональное тестирование 20](#_Toc170320172)

[3.6 Описание справочной системы 24](#_Toc170320173)

[4 Применение 25](#_Toc170320174)

[4.1 Назначение программного средства 25](#_Toc170320175)

[4.2 Условие применения 25](#_Toc170320176)

[Заключение 27](#_Toc170320177)

[Список использованных источников 28](#_Toc170320178)

[ПРИЛОЖЕНИЕ А Текст программы 29](#_Toc170320179)

[ПРИЛОЖЕНИЕ Б Результат функционального тестирования 49](#_Toc170320180)

# Введение

Киберинцидент – событие, которое фактически или потенциально угрожает конфиденциальности, целостности, подлинности, доступности и сохранности информации, а также представляет собой нарушение (угрозу нарушения) политик безопасности.

Последние время организации подвергаются беспрецедентным по своему размаху и интенсивности компьютерным атакам. Киберпреступность наносит большой материальный и репутационный ущерб государственным и коммерческим организациям вне зависимости от их величины и сферы.

Компьютерные криминалисты отмечают, что одним из главных ресурсов при проведении расследования и реагирования на киберинциденты является время. В связи с этим для компьютерных криминалистов вырастает роль эффективного анализа событий, произошедших в инфраструктуре организации.

При ведении расследования киберинцидента нужно уделять внимание организации учета инцидентов для дальнейшего анализа состояния информационной безопасности в организации.

В настоящее время в распоряжении организаций не имеется программных средств, позволяющих вести сбор, анализ, хранение информации о произошедших киберинцидентов в организации. В данный момент учет киберинцидентов производится по средству заполнения информации в таблицу Excel, что очень трудоёмко и не исключает большого количества ошибок из-за человеческого фактора.

Учитывая вышесказанное, можно сделать вывод, что автоматизация учета киберинцидентов для организаций и, в частности, для сотрудников информационной безопасности является актуальной проблемой и требует поиска путей решения. Выходом можем стать разработка автоматизированной системы, которая позволит вести учет киберинцидентов по всем подразделениям системы.

Актуальность работы: данное исследование направлено на решение одной из важнейших проблем сотрудников по защите информации – учет киберинцидентов, который является залогом точного и своевременного учета всех событий информационной безопасности.

Целью курсового проекта является проектирование и разработка программного средства, предназначенного для учета киберинцидентов в организациях.

Пояснительная записка к проекту состоит из четырех разделов, которые содержат решение поставленных задач, а также необходимую информацию по использованию программного средства «IB.exe».

В первом разделе «Описание задачи» описывается предметная область и круг задач, которые должны быть решены. Приводится характеристика периодичности использования программного средства. Описываются существующие аналоги.

Во втором разделе «Проектирование системы» перечисляются требования к приложению. Проводится моделирование всего программного средства и структуры базы данных. Предоставляется модель внешнего пользовательского интерфейса.

В третьем разделе «Описание реализации программного средства» описываются инструменты разработки, а также используемые технологии. Описывается инфраструктура базы данных, логическая и физическая организация данных реализованных функций. Описывается испытание разработанного программного средства с описанием организации входных и выходных данных. Так же описывается раздел справочной системы.

В четвертом разделе «Применение» предназначен для описания сведений о назначении программного средства и области его применения.

В заключении будет проанализировано созданное программное средство, определена степень соответствия поставленной задачи и выполненной работы. И описаны причины несоответствия, если таковые имеются.

Приложение А будет содержать текст программы.

Приложение Б будет содержать результаты функционального тестирования.

В графической части будут представлены диаграммы вариантов использования, классов и деятельности.

# 1 Описание задачи

## 1.1 Анализ предметной области

Темой проекта является разработка программного средства для учета киберинцидентов. Предметной областью является запись информации о киберинцидентах.

В настоящее время большинство специалистов по информационной безопасности ведет учет событий информационной безопасности (далее- ИБ) в неавтоматизированном формате - файлы Excel.

Для успешной работы необходимо постоянно владеть информацией о виде и количестве инцидентов, произошедших в инфраструктуре организации. С этим и помогают справляться системы учета инцидентов.

В предметной области учета киберинцидентов можно выделить несколько ключевых бизнес-процессов:

* идентификация инцидента: получение сообщений о киберинцидентах от пользователей, систем мониторинга или других источников; регистрация инцидента в системе с указанием первичных данных (дата, время, описание).
* классификация инцидента: определение типа инцидента (например, вирусная атака, утечка данных, DDoS-атака и т.д.); присвоение инциденту уникального идентификатора и типа отчета на основе классификации; назначение и расследование: назначение инцидента соответствующему подразделению для расследования; сбор и анализ данных по инциденту, выявление причин и последствий; запись предпринятых мер и решений по инциденту;
* отчетность и анализ: создание отчетов по инцидентам для внутреннего анализа и для представления внешним аудиториям (например, регуляторам); анализ данных для выявления тенденций и потенциальных угроз;
* закрытие инцидента: закрытие инцидента после завершения расследования и реализации всех необходимых мер; архивирование данных по инциденту для последующего анализа и отчетности.

Для эффективного управления и анализа киберинцидентов используются следующие алгоритмы получения промежуточных и итоговых показателей:

1. алгоритм классификации инцидентов:
   1. на основе типа отчета и данных об инциденте определить категорию и степень серьезности инцидента.
2. алгоритм расчета времени реагирования:
   1. вычислить время с момента идентификации инцидента до его закрытия.
   2. рассчитать среднее время реагирования для разных типов инцидентов.
3. алгоритм выявления тенденций:
   1. анализ данных по инцидентам за определенные периоды для выявления увеличения или уменьшения числа инцидентов.
   2. определение наиболее уязвимых подразделений и регионов.
4. алгоритм оценки эффективности мер:
   1. анализ предпринятых мер для каждого инцидента и оценка их эффективности.
   2. выявление мер, которые наиболее часто приводят к успешному разрешению инцидентов.

Потенциальная аудитория для программного средства учета киберинцидентов включает:

* ИТ-специалисты и специалисты по кибербезопасности: специалисты, ответственные за мониторинг, расследование и реагирование на киберинциденты.
* менеджмент: руководители, которым необходима информация для принятия стратегических решений по улучшению безопасности.
* регуляторы и аудиторы: внешние аудиторы и регуляторы, требующие отчетов о киберинцидентах и мерах, предпринятых для их устранения.
* аналитики: специалисты, занимающиеся анализом данных для выявления угроз и разработки превентивных мер.
* конечные пользователи: сотрудники организации, которые могут стать источниками информации о киберинцидентах и нуждаются в информации о том, как действовать в случае инцидента.

Эти сущности и их атрибуты позволяют отслеживать и управлять инцидентами, их типами, подразделениями, которые регистрируют инциденты, и регионами, к которым эти подразделения принадлежат, обеспечивая целостность и удобство доступа к информации.

## 1.2 Постановка задачи

Планируется разработать программное средство учета киберинцидентов для подразделения. В разрабатываемой программе должны быть реализованы следующие функции:

* добавление, удаление и редактирование события информационной безопасности;
* обновление списка инцидентов;
* просмотр событий информационной безопасности;
* добавление даты и времени события;
* выбор подразделения исходя из принадлежности определенному территориальному подразделению;
* заполнение данных по событию;
* описание принятых мер.

Программное средство должно обладать простым, удобным и легко осваиваемым интерфейсом. Также должна быть разработана база данных (далее – БД), в которой предусмотрено хранение информации об инцидентах, дате происшествия, принадлежность к определенному подразделению, описание события, краткие сведения по инциденту, описание принятых мер. База данных должна обеспечивать надежное хранение и доступ к информации.

Приложение не должно быть перегружено излишним функционалом.

Программа должна быть написана с использованием языка программирования С# с использованием Entity Framework. Графический интерфейс реализуется с использованием технологии Windows Presentation Foundation Microsoft.NET Framework (WPF). В качестве СУБД будет использоваться SQLite.

# 2 Проектирование системы

## 2.1 Требования к приложению

Приложение должно предоставлять пользовательский интерфейс, который легко понимается и прост в использовании. Он должен быть интуитивным, обеспечивая пользователям комфортное взаимодействие с функционалом приложения. Поддержка многопользовательского доступа должна быть предусмотрена, позволяя настраивать уровни привилегий в соответствии с ролями пользователей.

Одним из главных вещей в разработке является компьютер. Написание проекта будет вестись на ноутбуке со следующими характеристиками:

* процессор 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz 2.80

GHz, 2020 года выпуска, техпроцесс 10 нанометров, частота 2.80 ГГц и 4,70 ГГц в турбо режиме, 4 ядра, 8 потоков, 64-х битный набор команд;

* оперативная память 16 ГБ DDR4 3200 Гц;
* клавиатура и тачпад, встроенные в ноутбук.

## 2.2 Проектирование модели

На основе предъявленных требований были разработаны диаграммы вариантов использования, классов и деятельности Диаграмма вариантов использований иллюстрирует функциональные требования, диаграмма классов представляет структуру данных и связи между основными сущностями.

Функциональные требования хорошо иллюстрирует диаграмма вариантов использования. В качестве актера выступает офицер информационной безопасности. Актер взаимодействует с системой через четыре основных варианта использования:

* вывод данных: актер может запросить данные о событиях информационной безопасности для анализа и отчетности. Связь типа «include» указывает, что вывод данных является неотъемлемой частью других действий, таких как редактирование или удаление событий;
* добавление: актер может добавить новое событие ИБ в систему; связь типа «include» с выводом данных указывает, что после добавления данных может потребоваться их вывод для проверки или анализа.
* удаление: актер может удалить событие ИБ из системы; связь типа «include» с выводом данных указывает, что перед удалением данных необходимо их вывести для подтверждения правильности выбора.
* редактирование: актер может редактировать существующие события ИБ в системе; связь типа «include» с выводом данных указывает, что перед редактированием данных необходимо их вывести для проверки и внесения изменений.

Диаграмма вариантов использования представлена в графической части на листе 2.

Диаграмма классов представляет структуру данных и связи между основными сущностями, которые используются в системе учета киберинцидентов.

Основные классы и их атрибуты:

Incident (Инцидент)

* 1. атрибуты:
     1. Date: дата происшествия.
     2. DateCreateModify: дата создания или последнего изменения.
     3. Description: описание инцидента.
     4. Division: подразделение, зарегистрировавшее инцидент.
     5. DivisionId: идентификатор подразделения.
     6. IncidentData: дополнительные данные об инциденте.
     7. MeasuresTaken: предпринятые меры.
     8. Time: время происшествия.
     9. TypeReport: тип отчета об инциденте.
     10. TypeReportId: идентификатор типа отчета.

Division (Подразделение)

* 1. атрибуты:
     1. Id: уникальный идентификатор подразделения.
     2. Name: название подразделения.
     3. RegionId: идентификатор региона.

Region (Регион)

* 1. атрибуты:
     1. Id: уникальный идентификатор региона.
     2. Name: название региона.

TypeReport (Тип отчета)

* 1. атрибуты:
     1. Id: уникальный идентификатор типа отчета.
     2. Name: название типа отчета.
     3. Sort: порядок сортировки.

EFDBContext (Контекст базы данных)

* 1. атрибуты:
     1. Incidents: список инцидентов.
     2. Divisions: список подразделений.
     3. Regions: список регионов.
     4. TypeReports: список типов отчетов.

Эти классы и их атрибуты обеспечивают структурированное хранение и доступ к информации о киберинцидентах, подразделениях, регионах и типах отчетов.

Диаграмма классов представлена в графической части на листе 1.

На основе представленных диаграмм была разработана база данных. В представленной схеме базы данных есть несколько сущностей, каждая из которых имеет свои атрибуты.

Сущность TypeReports включает такие атрибуты, как уникальный идентификатор типа отчета (Id), название типа отчета (Name) и порядок сортировки типов отчетов (Sort).

Сущность Incidents содержит уникальный идентификатор инцидента (Id), идентификатор подразделения (DivisionId), идентификатор типа отчета (TypeReportId), описание инцидента (Description), дату происшествия (Date), время происшествия (Time), дополнительные данные об инциденте (IncidentData), предпринятые меры (MeasuresTaken), дату создания или последнего изменения записи (DateCreateModify) и дату фиксации инцидента в системе (IncidentDate).

Сущность Division состоит из уникального идентификатора подразделения (ID), идентификатора региона (RegionId) и названия подразделения (Name).

Сущность Regions включает уникальный идентификатор региона (ID) и название региона (Name).

Сущность sqlite\_sequence состоит из уникального идентификатора (ID), имени таблицы, для которой отслеживается последовательность (name), и текущего значения последовательности (seq).

Эти сущности и их атрибуты позволяют отслеживать и управлять инцидентами, их типами, подразделениями, которые регистрируют инциденты, и регионами, к которым эти подразделения принадлежат, обеспечивая целостность и удобство доступа к информации.

## 2.3 Концептуальный прототип

Исходя из диаграммы вариантов использования и этапа анализа требований был спроектирован простой и интуитивно понятный графический пользовательский интерфейс. Макеты разрабатываемых окон и описание полей приведены на рисунках 1-3.

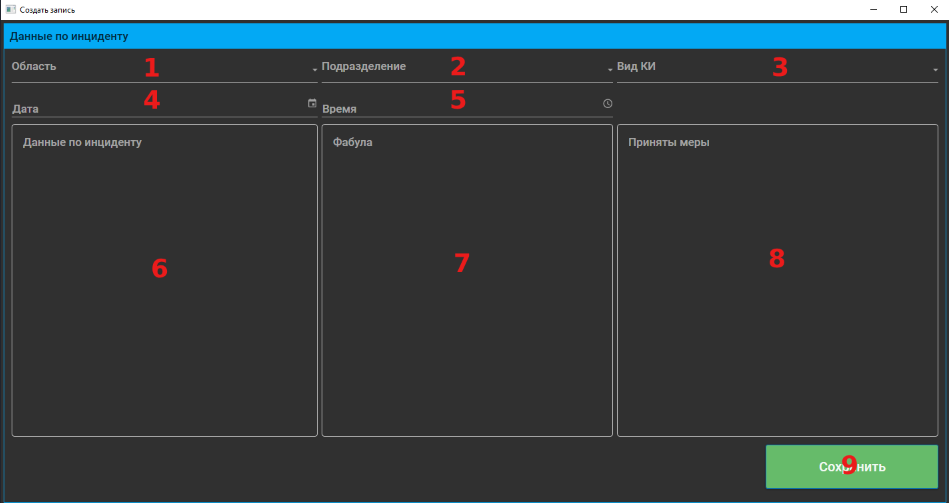


Рисунок 1 – Интерфейс окна добавления инцидента

На рисунке 1 представлен интерфейс окна добавления инциндента. Интерфейс выполнен в минималистическом дизайне с удобным и понятным расположением объектов.

* раскрывающийся список для выбора области: выбор любого из предоставленных областей: (combobox);
* раскрывающийся список для выбора подразделения: выбор любого из предоставленных подразделений в зависимости от выбранной области: (combobox);
* раскрывающийся список для выбора вида инцидента: выбор любого из предоставленных областей: (combobox);
* элемент «Календарь» для выбора даты события ИБ: выбор даты, когда случился инцидент: (DatePicker);
* элемент «Часы» для выбора времени события ИБ: выбор даты, когда случился инцидент: (DatePicker);
* поле для ввода информации по инциденту: ввод данных о произошедшем инцинденте: (текстовое поле);
* поле для ввода краткой информации (фабула) по инциденту: ввод кратких данных о произошедшем инцинденте: (текстовое поле);
* поле для ввода информации по принятым мерам по инциденту: ввод данных о принятых мерах по произошедшему инцинденте: (текстовое поле);
* кнопка для сохранения введенной информации: сохранение введенных данных в таблицу учёта.

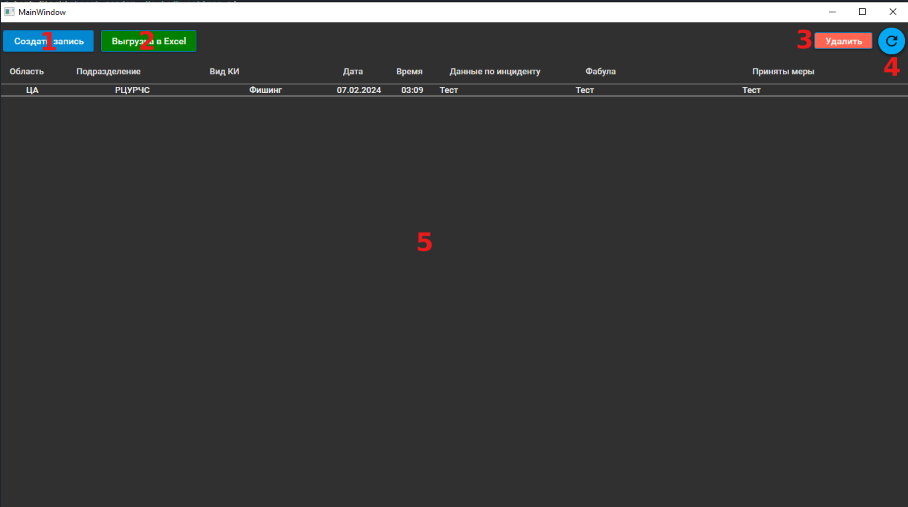


Рисунок 2 – Интерфейс окна вывода списка инцидента

* кнопка для создания записи нового инцидента: открытие окна для создания записи о новом инцинденте;
* кнопка для выгрузки списка всех инцидентов в Excel: создаётся файл для выгрузки; все данные с таблицы выгружаются в созданный файл;
* кнопка для удаления выделенной записи: выбранное поле будет удалено;
* кнопка для обновления списка записей: обновляется таблица записей;
* поле для вывода списка всех инцидентов: отображение всех зафиксированных инциндентов.

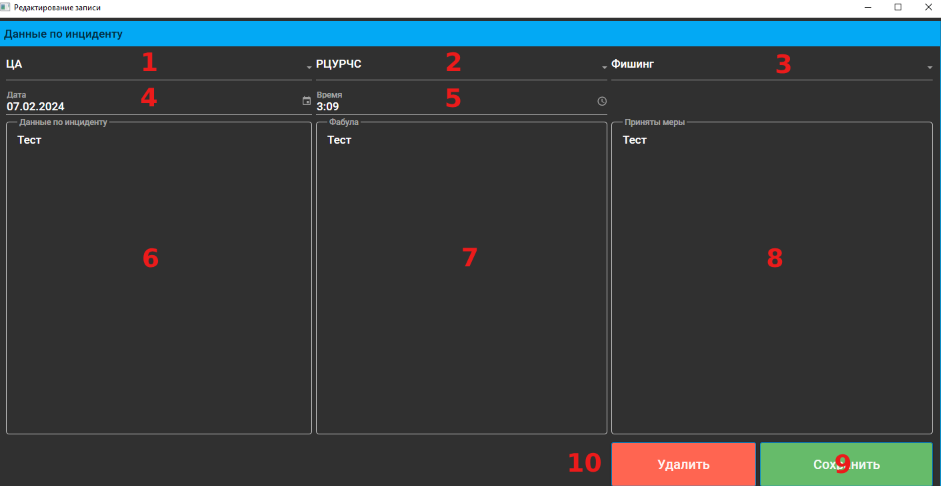


Рисунок 3 – Интерфейс окна редактирования инцидента

* раскрывающийся список для редактирования выбора области: выбор любого из предоставленных областей: (combobox);
* раскрывающийся список для редактирования выбора подразделения: выбор любого из предоставленных подразделений в зависимостей от выбранной области: (combobox);
* раскрывающийся список для редактирования выбора вида инцидента: выбор любого из предоставленных областей: (combobox);
* элемент «Календарь» для редактирования выбора даты события ИБ: выбор даты, когда случился инцидент: (DatePicker);
* элемент «Часы» для редактирования выбора времени события ИБ: выбор даты, когда случился инцидент: (DatePicker);
* поле для редактирования ввода информации по инциденту: ввод данных о произошедшем инцинденте: (текстовое поле);
* поле для редактирования ввода краткой информации (фабула) по инциденту: ввод кратких данных о произошедшем инцинденте: (текстовое поле);
* поле для редактирования ввода информации по принятым мерам по инциденту: ввод данных о принятых мерах по произошедшему инцинденте: (текстовое поле);
* кнопка для сохранения отредактированной информации: сохранение введенных данных в таблицу учёта;
* кнопка удаления редактирования: удаление отредактированных данных.

# 3 Описание реализации программного средства

## 3.1 Инструменты разработки и применяемые технологии

В настоящее время на рынке программного обеспечения (далее – ПО) существует множество программных продуктов для автоматизации работы сотрудников информационной безопасности.

Они отличаются друг от друга набором функций, стабильностью, удобством работы, стоимостью.

Приступая к анализу основных из них, следует отметить, что на рынке отсутствуют решения, предназначенные исключительно для учета киберинцидентов.

Для анализа выбрана SIEM– система (Security Information and Event Management). Примеры программных окон представлены на рисунке 4.

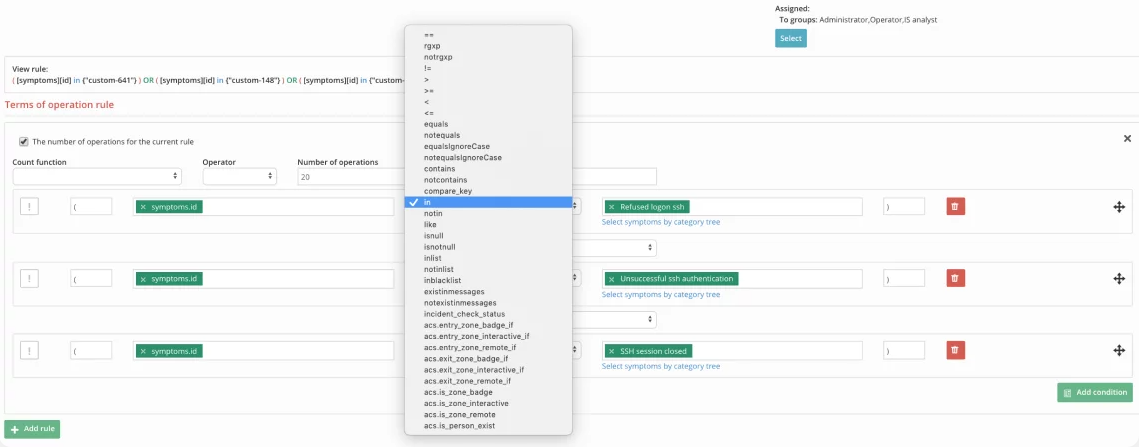


Рисунок 4 – Интерфейс программы RUSIEM

SIEM обеспечивает анализ в реальном времени событий безопасности, исходящих от сетевых устройств и приложений, и позволяет реагировать на них до наступления существенного ущерба. Основными задачами систем данного класса являются сбор, обработка и анализ событий безопасности, поступающих в систему из множества источников, также обнаружение в режиме реального времени атак и нарушений критериев и политик безопасности. На основе получаемых событий ИБ осуществляется оперативная оценка защищенности информационных, телекоммуникационных и других критически важных ресурсов, анализ и управление рисками безопасности. Во многие современные SIEM системы встроены механизмы принятия решений и инструменты расследования инцидентов. Также неотъемлемой частью SIEM-системы является наличие средств для формирования отчетных документов. На рисунках 1-2 представлен пример окна программы.

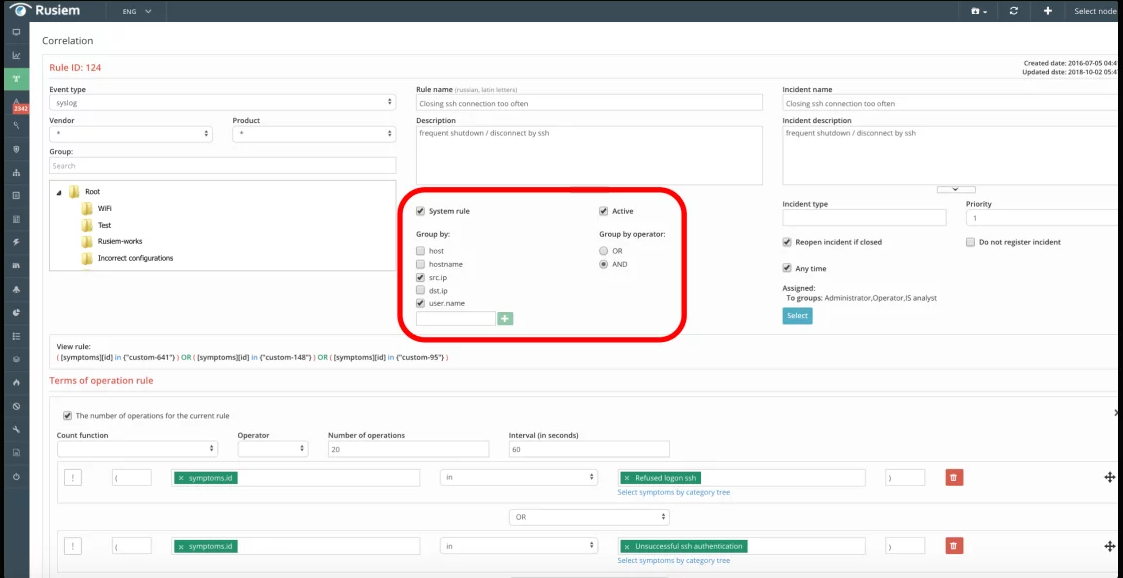


Рисунок 5 – Интерфейс программы RUSIEM

Эта программа обладают существенными недостатками для маленьких организаций (ограниченность в финансовых ресурсах): достаточно высокая стоимость, необходимость обучения, платное сопровождение, и приобретение в пакете излишнего функционала.

Таким образом, принято решение в ходе данной работы реализовать собственное программное средство по учету киберинцидентов, лишенная указанных недостатков.

## 3.2 Организация данных

Схема БД должна включать набор всех схем ее таблиц, т.е. описание всех колонок этих таблиц (их типов, допустимых значений, связей между таблицами, типа внешних ключей и т.д.), без учета конкретных данных.

При составлении схемы базы данных разрабатываемой программы создано необходимое количество таблиц для хранения нужной информации – Division, Incident, Region, TypeReport.

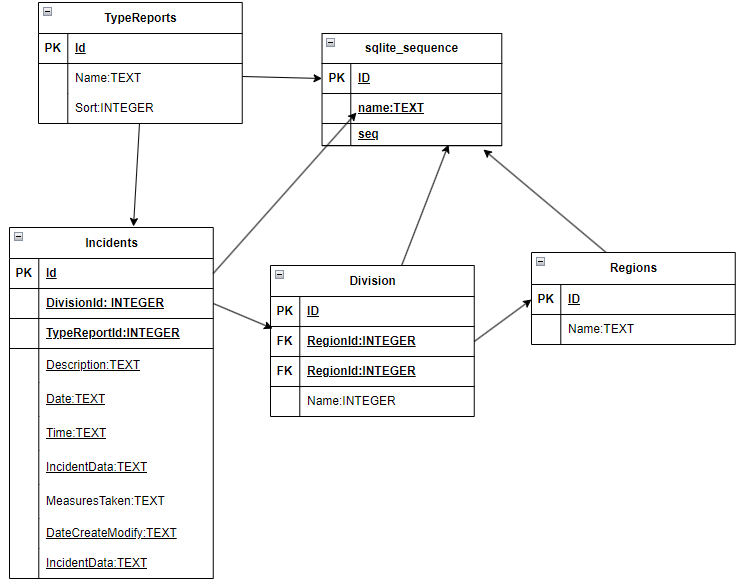


Рисунок 6 – Схема базы данных

Данные таблицы соответствуют одноименным сущностям. Значения атрибутов сущностей представлены в следующих таблицах. Сущность «Инцидент» предназначена для хранения информации о событиях безопасности в организации. Описание атрибутов представлено в таблице 1.

Таблица 1 – Структура таблицы «Инцидент»

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Тип данных | Размер поля, байт | Описание |
| Id (РК) | Integer | 6 | Код инцидента |
| DivisionId (FK) | Integer | 15 | Код подразделение |
| TypeReportId (FK) | Text | 20 | Код вида инцидента |
| Description | Text | 20 | Фабула |
| Date | Text | 10 | Дата |
| Time(FK) | Text | 10 | Время |
| MeasuresToken | Text | 20 | Принятые меры |

Сущность «Подразделение» предназначена для хранения информации

о подразделениях в разрезе территориальной расположенности. Описание атрибутов представлено в таблице 2.

Таблица 2 – Структур таблицы «Подразделение»

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Тип данных | Размер поля, байт | Описание |
| Id (РК) | Integer | 25 | Код подразделения |
| Name | Integer | 50 | Подразделение |
| RegionId(FK) | Integer | 50 | Код области |

Сущность «Область» предназначена для хранения информации о областях. Описание атрибутов представлено в таблице 3.

Таблица 3 – Структур таблицы «Область»

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Тип данных | Размер поля,байт | Описание |
| Id (РК) | Integer | 25 | Номер области |
| Name | Text | 50 | Название области |

Сущность «Вид инцидента» предназначена для хранения информации о видах инцидента. Описание атрибутов представлено в таблице 4.

Таблица 4 – Структур таблицы «Вид инцидента»

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Тип данных | Размер поля, байт | Описание |
| Id (РК) | Integer | 25 | Код вида инцидента |
| Name | Text | 50 | Наименование вида |

## 3.3 Функции: логическая и физическая организация

Основываясь на диаграмме вариантов использования, можно выделить все основные функции программы, это такие как: добавление новых событий информационной безопасности в систему, редактирование данных о существующих событиях информационной безопасности, удаление событий информационной безопасности из системы, получение и отображение данных о событиях информационной безопасности. Эти функции охватывают основные действия, которые может выполнять офицер ИБ с событиями информационной безопасности в системе.

Код выгрузки данных в Excel:

private void Button\_Click(object sender, RoutedEventArgs e)//выгрузка в эксель

{

if (MessageBox.Show("Выгрузить документ? Он будет сохранен на диск D!", "Внимание!", MessageBoxButton.YesNo) == MessageBoxResult.Yes)

{

List<Incident> items = \_context.Incidents

.Include(s => s.Division.Region)

.Include(s => s.TypeReport)

.ToList();

byte[] res = ExcelIncident.Generate(items);

FileStream aFile = new FileStream(PathFile, FileMode.Create);

aFile.Seek(0, SeekOrigin.Begin);

aFile.Write(res, 0, res.Length);

aFile.Close();

}

return;

}

Код функции обновления страницы:

private async void LoadData() //функция обновления

{

try

{

DgIncidents.ItemsSource = await \_context.Incidents.AsNoTracking().Include(x => x.TypeReport).Include(x => x.Division.Region).ToListAsync();

if (ApplicationGlobal.Regions == null)

ApplicationGlobal.Regions = await \_context.Regions.Include(x => x.Divisions).AsNoTracking().ToListAsync();

if (ApplicationGlobal.TypeReports == null)

ApplicationGlobal.TypeReports = await \_context.TypeReports.AsNoTracking().ToListAsync();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void btnRefresh\_Click(object sender, RoutedEventArgs e)//обновление

{

LoadData();

}

Полный код программы представлен в приложении А.

## 3.4 Входные и выходные данные

Проект разработка программного средства для учета киберинцидентов разрабатывается как инструментальное приложение для управления инцидентами информационной безопасности. Его характер направлен на обеспечение эффективного учета и анализа инцидентов. Организация проекта следует принципам объектно-ориентированного программирования (ООП), обеспечивая четкую структуру и управление данными.

Предварительная подготовка

Предварительная подготовка включает анализ требований пользователя и определение основных функциональных возможностей. Также проведено исследование методов управления инцидентами и требований к информационной безопасности для обеспечения правильной реализации соответствующих функций.

Формат, описание и способ кодирования:

* формат кода: используется стандартный формат кодирования C# в соответствии с принятыми стандартами стиля кода. Это включает в себя четкое и понятное форматирование, использование пробелов, правильное именование переменных и методов.
* описание кода: каждый класс, метод и переменная должны быть адекватно и понятно описаны комментариями. Описание должно включать в себя информацию о назначении элемента, его параметрах, возвращаемых значениях, а также особенностях реализации.
* способ кодирования: применение ООП для разделения функциональности на логические классы и методы; использование современных и эффективных практик разработки, таких как отлаживаемость кода, обработка ошибок и использование событий для обратной связи с пользователем; поддержка многопоточности для обеспечения отзывчивости приложения и эффективной передачи данных.
* формат данных: данные, вводимые пользователем и обрабатываемые приложением, должны соответствовать стандартам информационной безопасности; формат данных определяется спецификацией функциональных требований, обеспечивая корректное взаимодействие между различными компонентами приложения.
* шифрование данных: в случае использования SSL/TLS, данные передаются в зашифрованном виде, обеспечивая конфиденциальность информации пользователя.
* тестирование: применение тестирования на всех уровнях разработки, включая модульное тестирование для отдельных компонентов и функций, а также интеграционное и системное тестирование для проверки работы всего приложения.
* документация: поддержание подробной документации, включающей в себя описание классов и методов, комментарии к коду, а также инструкции по использованию и настройке приложения.

## 3.5 Функциональное тестирование

Для оценки и повышения качества полученного программного продукта было проведено функциональное тестирование на соответствие требованиям.

Функциональное тестирование (Functional Testing) – тестирование, основанное на сравнительном анализе спецификации и функциональности компонента или системы.

В качестве рабочей тестовой документации использовались Test Cases. Test Cases представляет собой перечень всех модулей и функций приложения, а также конкретные проверки для них. Каждая проверка содержит набор входных значений, предусловий, шагов выполнения и ожидаемых результатов. Всегда приводятся ожидаемые результаты для каждого шага проверки.

Тестирование проводилось на уровне Minimal Acceptance Test (MAT). Данный уровень тестирования предполагает выполнение тестов на корректных данных, сценариях.

Было проведено тестирование следующих модулей:

* тестирование запуска приложения;
* тестирование модуля меню;
* тестирование модуля удаления данных;
* тестирование модуля добавления данных;
* тестирование модуля обновления данных.

Выполненный набор тест-кейсов представлен в таблице 5.1. Результаты функционального тестирования представлены в приложении Б.

Таблица 5.1 – Функциональное тестирование

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Модуль | | Описание | Ожидаемый результат | Фактический результат | |
| 1 | Приложение | | **Запуск приложения:**  Запустить IB.exe | Отображение окна запущенного  приложения. | Выполнено  успешно | |
| 2 | Главное меню | | **Добавление записи:**  Нажать на кнопку «Создать запись». | Открытие окна для создания новой записи. | Выполнено  успешно | |
| 3 | Главное меню | | **Удаление записи:**   1. Выбираем в списке инцидентов любую запись для удаления. 2. Нажимаем кнопку «Удалить». 3. Появляется окно-подтверждение. 4. Нажимаем кнопку «Да». Нажимаем кнопку обновить. | 1. Запись выбирается. 2. После нажатия на кнопку появляется окно-подтверждение. 3. Запись удалена | Выполнено  Успешно (результат функционального тестирования предоставлен на рисунке Б.1-Б.3) | |
| 4 | Главное меню | | **Выгрузка в Excel:**  Нажимаем на кнопку «Выгрузить в Excel». | Данные успешно выгружены в таблицу. | Выполнено  успешно | |
| 5 | Главное меню | | **Отчёт по кол-ву КИ по областям:**  1. Выбрать пункт «Области по кол-ву КИ»  2.Отображение отчёта | Данные отображаются корректно | Выполнено успешно (результат функционального тестирования предоставлен на рисунке Б.10-Б.11) | |
|  |  | |  |  |  | |
| Приложение таблицы 5.1 | | | | | | |
| № | | Модуль | Описание | Ожидаемый результат | Фактический результат |
| 6 | Окно создания новой записи инцидента | | **Создание новой записи:**   1. Нажимаем кнопку «Создание записи». 2. В окне создания записи выбираем из списков данные, дату и время, заполняем все поля. 3. Нажимаем кнопку «Сохранить». 4. Закрываем окно создания записи. 5. Нажимаем кнопку «Обновить». | 1. См. 2, п. 1. 2. Поля заполняются исправно. 3. Данные сохраняются. 4. После обновления страницы данные отображаются на главной странице. | Выполнено  успешно (результат функционального тестирования предоставлен на рисунке Б.4-Б.6) | |
| 7 | Главное меню | | **Кнопка обновления:**  Нажать на кнопку обновления страницы. | Страница удачно обновилась и верно отображает текущую информацию. | Выполнено  успешно |
| 8 | Окно редактирования инцидента | | **Редактирование инциндента:**   1. В главном окне в списке инцидентов двойным щелчком выбираем инцидент для редактирования. 2. Изменяем нужные позиции. 3. Нажимаем кнопку «Сохранить». 4. Закрываем окно «Редактирования». 5. Нажимаем кнопку «Обновить». | 1. Окно для редактирования записи открывается. 2. Нужные позиции отредактированы. 3. Данные сохранены. 4. После обновления отредактированные данные отображаются корректно. | Выполнено  Успешно (результат функционального тестирования предоставлен на рисунке Б.7-Б.9) |
|  |  | |  |  |  |

Также были разработаны и проведены несколько unit-тестов. Unit-тесты позволяют быстро и автоматически протестировать отдельные участки кода независимо от остальной части программы. Для имитации и эмуляции объектов и их функциональности использовалась библиотека Moq.

Все выявленные при проведении тестирования ошибки были устранены. По результатам тестирования можно сделать выводы:

* об отсутствии критических ошибок в приложении;
* о соответствии разработанного приложения предъявляемым в задании требованиям.

## 3.6 Описание справочной системы

Справочная система в приложении отсутствует. Перед использованием рекомендуется прочитать инструкцию по использованию, представленная в «Инструкция.docx».

# 4 Применение

## 4.1 Назначение программного средства

Программа разработка программного средства для учета киберинцидентов представляет собой эффективный инструмент, разработанный для облегчения управления инцидентами информационной безопасности. Её назначение заключается в создании удобной и надежной среды для регистрации, анализа и отслеживания инцидентов, связанных с информационной безопасностью.

Основной функционал программы охватывает разнообразные задачи. Возможность регистрации и управления инцидентами, предоставление аналитической информации, а также информационная поддержка делают "Приложение учета киберинцидентов" всесторонним инструментом для обеспечения информационной безопасности. Пользователь может легко добавлять, редактировать и удалять записи о киберинцидентах, поддерживая порядок и организованность в учёте событий ИБ.

Информационная поддержка включает в себя получение полезных данных о киберинцидентах, таких как их описание, дата и время происшествия, затронутые системы и последствия. Это обеспечивает пользователю всю необходимую информацию для принятия взвешенных решений при управлении и анализе инцидентов.

Безопасность данных - один из приоритетов "Приложения учета киберинцидентов". Программа предоставляет возможность использования шифрованного соединения через SSL/TLS, а также предусматривает аутентификацию пользователя для обеспечения конфиденциальности и целостности данных.

Визуальная часть программы разработана с учетом удобства использования. Интуитивный пользовательский интерфейс позволяет легко вводить данные об инцидентах, отслеживать результаты анализа и воспользоваться всеми функциональными возможностями «Приложения учета киберинцидентов».

Все компоненты программы написаны с использованием четких стандартов кодирования и снабжены подробными комментариями. Это обеспечивает не только эффективность, но и легкость поддержки и дальнейшего развития проекта.

## 4.2 Условие применения

Для корректной работы программы нет необходимости в сторонних программных продуктах или средствах. Программа является полностью автономной и может переноситься без особого труда на разные компьютеры. Единственным условием, которое может помочь для более комфортного просмотра экспортированных данных является наличие на компьютере текстового редактора.

Также для работы программы у компьютера должны следующие минимальные аппаратные характеристики:

* процессор AMD A12-9720P, 2017 года выпуска, техпроцесс 28 нанометров, частота 2,6 ГГц и 3,6 ГГц в турбо режиме, 4 ядра, 4 потока, 64-х битный набор команд;
* оперативная память 4 ГБ DDR4 1866 Гц;
* операционная система MS Windows 10, 64-х разрядная;
* клавиатура;
* мышь.

Данные аппаратные характеристики основаны примере компьютера, на котором проводилось тестирование приложения.

# Заключение

В рамках курсового проектирования проекте было создано программное средство, которое служит для автоматизации учета инцидентов информационной безопасности. Цель разработки: минимизация временных затрат и повышение точности в учете киберинцидентов.

Программа предоставляет пользователю (офицеру безопасности) возможность добавлять новые инциденты, удалять, редактировать существующие, выгружать весь список инцидентов в Excel. Разработанное программное средство позволяет офицеру безопасности вести учет всех событий информационной безопасности в организации.

Приложение реализовано посредством языка программирования С# (среда Visual Studio 2022 Community) с использованием встроенной БД SQLite.

В результате выполнения данной работы разработано программное средство, отвечающее поставленным задачам. Несмотря на выполнение поставленных задач, в настоящее время намечены работы, которые помогут расширить функциональные возможности программного средства.

# Список использованных источников

1. Багласова, Т.Г. Методические указания по оформлению курсовых проектов, дипломных проектов и отчетов для учащихся специальности 2-40 01 01 «Программное обеспечение информационных технологий» // Т.Г. Багласова, К.О. Якимович и др. – Минск : ЧУО «КБП» , 2022 . – 41 с.
2. RuSIEM [Электронный ресурс]. – 2024. – Режим доступа: https://rusiem.com/ru/products/rusiem
3. Habr [Электронный ресурс]. - 2024. - Режим доступа: https://habr.com/ru/articles/647259/.
4. Справочник по C# (электронный ресурс). – Электронные данные. – Режим доступа: - https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/index
5. Positive technologies [Электронный ресурс]. - Режим доступа: https://www.ptsecurity.com/ru-ru/.
6. Общие требования к тестовым документам: ГОСТ 2.105-95. – Введ. 01.01.1996. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 1995 . – 84 с
7. Объектно-ориентированный анализ и проектирование с примерами приложений / Гради Буч [и др.]. – 3-е изд. – М.: ООО «И.Д. Вильямс», 2008 . – 720 с.
8. Описание программы. Требования к содержанию, оформлению и контролю качества: ГОСТ 19.402-2000. – Введ. 01.09.2001. – Минск: Межгос. совет по стандартизации, метрологии и сертификации, 2000 . – 14 с.
9. Программа и методика испытаний. Требования к содержанию, оформлению и контролю качества: ГОСТ 19.301-2000. – Введ. 01.09.2001. – Минск: Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 14 с.
10. Текст программы. Требования к содержанию, оформлению и контролю качества: ГОСТ 19.401-2000. – Введ. 01.09.2001. – Минск: Межгос. совет по стандартизации, метрологии и сертификации, 2000 . – 16 с.
11. Draw.io – бесплатное средство для создания блок-схем,

инфографики, прототипов [Электронный ресурс]. – EL-BLOG.RU, 2024 – Режим доступа: https://el-blog.ru/draw-io/. – Дата доступа: 15.01.2024.

1. Visual Studio 2022 [Электронный ресурс]. – Microsoft, 2024. – Режим

доступа: https://visualstudio.microsoft.com/ru/vs. – Дата доступа 15.04.2024.

# ПРИЛОЖЕНИЕ А

**(обязательное)  
Текст программы**

// Подразделения

namespace IB.DataService.Models

{

public class Division

{

public int Id { get; set; }

public string Name { get; set; }

public Region Region { get; set; }

public int RegionId { get; set; }

public override string ToString()

{

return Name;

}

}

}

//Инциденты

using System;

namespace IB.DataService.Models

{

public class Incident

{

public int Id { get; set; }

public Division Division { get; set; }

public int DivisionId { get; set; }

public string IncidentData { get; set; } // данные по инциденту

public string MeasuresTaken { get; set; } // приняты меры

public string Description { get; set; } // фабула

public DateTime Date { get; set; }

public DateTime Time { get; set; }

public TypeReport TypeReport { get; set; } // вид КИ

public int TypeReportId { get; set; }

public DateTime DateCreateModify { get; set; }

}

}

// Область

using System.Collections.Generic;

namespace IB.DataService.Models

{

public class Region

{

public int Id { get; set; }

public string Name { get; set; }

public List<Division> Divisions { get; set; }

public Region()

{

Divisions = new List<Division>();

}

public override string ToString()

{

return Name;

}

}

}

//Информация по инциденту (контекст)

namespace IB.DataService.Models

{

public class TypeReport

{

public int Id { get; set; }

public string Name { get; set; }

public override string ToString()

{

return Name;

}

}

}

// Вырузка в Excel

using IB.DataService.Models;

using OfficeOpenXml;

using System.Collections.Generic;

using System.IO;

namespace IB.Excel

{

public static class ExcelIncident

{

public static byte[] Generate(List<Incident> incidents)

{

byte[] result;

ExcelPackage.LicenseContext = LicenseContext.NonCommercial;

using (ExcelPackage excelPackage = new ExcelPackage(new FileInfo("Incidents.xlsx")))

{

int row, col;

row = col = 1;

ExcelWorksheet PPS = excelPackage.Workbook.Worksheets.Add("Учет");

PPS.Cells[row, col].Value = "Область";

ExcelSupport.AddStyle(PPS.Cells[row, col], verticalWithHorizontalAligment: true, bold: true);

++col;

PPS.Cells[row, col].Value = "Подразделение";

ExcelSupport.AddStyle(PPS.Cells[row, col], verticalWithHorizontalAligment: true, bold: true);

++col;

PPS.Cells[row, col].Value = "Вид КИ";

ExcelSupport.AddStyle(PPS.Cells[row, col], verticalWithHorizontalAligment: true, bold: true);

++col;

PPS.Cells[row, col].Value = "Дата";

ExcelSupport.AddStyle(PPS.Cells[row, col], verticalWithHorizontalAligment: true, bold: true);

++col;

PPS.Cells[row, col].Value = "Время";

ExcelSupport.AddStyle(PPS.Cells[row, col], verticalWithHorizontalAligment: true, bold: true);

++col;

PPS.Cells[row, col].Value = "Данные по КИ";

ExcelSupport.AddStyle(PPS.Cells[row, col], verticalWithHorizontalAligment: true, bold: true);

++col;

PPS.Cells[row, col].Value = "Фабула";

ExcelSupport.AddStyle(PPS.Cells[row, col], verticalWithHorizontalAligment: true, bold: true);

++col;

PPS.Cells[row, col].Value = "Приняты меры";

ExcelSupport.AddStyle(PPS.Cells[row, col], verticalWithHorizontalAligment: true, bold: true);

++col;

PPS.Column(1).Width = 30;

PPS.Column(2).Width = 14;

PPS.Column(3).Width = 14;

PPS.Column(4).Width = 18;

PPS.Column(5).Width = 12;

PPS.Column(6).Width = 14;

PPS.Column(7).Width = 15;

PPS.Column(8).Width = 15;

PPS.Column(1).Style.WrapText = true;

PPS.Column(2).Style.WrapText = true;

PPS.Column(3).Style.WrapText = true;

PPS.Column(4).Style.WrapText = true;

PPS.Column(5).Style.WrapText = true;

PPS.Column(6).Style.WrapText = true;

PPS.Column(7).Style.WrapText = true;

PPS.Column(8).Style.WrapText = true;

SetData(PPS, incidents);

result = excelPackage.GetAsByteArray();

}

return result;

}

private static void SetData(ExcelWorksheet excelWorksheet, List<Incident> incidents)

{

int row, col;

row = 2;

foreach (Incident i in incidents)

{

col = 1;

excelWorksheet.Cells[row, col].Value = i.Division.Region.Name;

ExcelSupport.AddStyle(excelWorksheet.Cells[row, col], verticalWithHorizontalAligment: true);

++col;

excelWorksheet.Cells[row, col].Value = i.Division.Name;

ExcelSupport.AddStyle(excelWorksheet.Cells[row, col], verticalWithHorizontalAligment: true);

++col;

excelWorksheet.Cells[row, col].Value = i.TypeReport.Name;

ExcelSupport.AddStyle(excelWorksheet.Cells[row, col], verticalWithHorizontalAligment: true);

++col;

excelWorksheet.Cells[row, col].Value = i.Date;

ExcelSupport.AddStyle(excelWorksheet.Cells[row, col], verticalWithHorizontalAligment: true);

++col;

excelWorksheet.Cells[row, col].Value = i.Time;

ExcelSupport.AddStyle(excelWorksheet.Cells[row, col], verticalWithHorizontalAligment: true);

++col;

excelWorksheet.Cells[row, col].Value = i.IncidentData;

ExcelSupport.AddStyle(excelWorksheet.Cells[row, col], verticalWithHorizontalAligment: true);

++col;

excelWorksheet.Cells[row, col].Value = i.Description;

ExcelSupport.AddStyle(excelWorksheet.Cells[row, col], verticalWithHorizontalAligment: true);

++col;

excelWorksheet.Cells[row, col].Value = i.MeasuresTaken;

ExcelSupport.AddStyle(excelWorksheet.Cells[row, col], verticalWithHorizontalAligment: true);

++row;

}

}

}

}

using OfficeOpenXml.Style;

using OfficeOpenXml;

namespace IB.Excel

{

public static class ExcelSupport

{

public static void AddStyle(

ExcelRange excelRange,

int size = 10,

bool bold = false,

bool rotate = false,

bool verticalWithHorizontalAligment = false,

bool italic = false,

bool textWrap = false)

{

excelRange.Style.Font.Bold = bold;

excelRange.Style.Font.Size = size;

excelRange.Style.Font.Italic = italic;

if (textWrap)

excelRange.Style.WrapText = true;

if (rotate)

excelRange.Style.TextRotation = 90;

if (verticalWithHorizontalAligment)

{

excelRange.Style.VerticalAlignment = ExcelVerticalAlignment.Center;

excelRange.Style.HorizontalAlignment = ExcelHorizontalAlignment.Center;

}

}

public static void AddBorder(

ExcelRange excelRange,

bool setLeft = false,

bool setRight = false,

bool setTop = false,

bool setBottom = false,

bool doubleBorder = false)

{

ExcelBorderStyle excelBorderStyle = doubleBorder ?

ExcelBorderStyle.Thick : ExcelBorderStyle.Thin;

if (setLeft)

excelRange.Style.Border.Left.Style = excelBorderStyle;

if (setRight)

excelRange.Style.Border.Right.Style = excelBorderStyle;

if (setTop)

excelRange.Style.Border.Top.Style = excelBorderStyle;

if (setBottom)

excelRange.Style.Border.Bottom.Style = excelBorderStyle;

}

}

}

using System.Collections.ObjectModel;

using System.Windows.Input;

using TermPaper.Commands;

using TermPaper.Domain.Entities;

using TermPaper.UseCases;

using TermPaper.UseCases.Interfaces;

namespace TermPaper.ViewModels

{

public class MainWindowViewModel : BaseViewModel

{

IGenericRepository<Warehouse> repository;

private IReportPrinter printer;

public MainWindowViewModel(IGenericRepository<Warehouse> repository, IReportPrinter printer)

{

this.repository = repository;

this.printer = printer;

Warehouses = new ObservableCollection<Warehouse>(repository.GetAll());

}

private ObservableCollection<Warehouse> warehouses;

public ObservableCollection<Warehouse> Warehouses

{

get => warehouses;

private set

{

Set(ref warehouses, value);

}

}

private Warehouse selectedItem;//храним выбранный пользователем склад, тк весь объект хранится

public Warehouse SelectedItem {

get

{

return selectedItem;

}

set

{

Set(ref selectedItem, value);

}

}

#region Добавить новый или редактировать склад

private ICommand \_addOrEditWarehouseCommand;

public ICommand AddOrEditWarehouseCommand => \_addOrEditWarehouseCommand ?? new RelayCommand(OnAddOrEdit);

private void OnAddOrEdit(object p)

{

if (SelectedItem.Number == 0)

{

repository.Create(SelectedItem);

}

else

{

repository.Update(SelectedItem);

}

OnRefresh(null);

}

#endregion

#region Удаление ПХ

private RelayCommand \_deleteCommand;

public ICommand DeleteCommand => \_deleteCommand ?? new RelayCommand(OnDeleteWarehouseEx, (obj) => SelectedItem != null);

private void OnDeleteWarehouseEx(object p)

{

repository.Delete(SelectedItem);

OnRefresh(null);

}

#endregion

// команда для обновления данных

public override ICommand RefreshCommand => refreshCommand ??= new RelayCommand(OnRefresh);

private ICommand refreshCommand;

private void OnRefresh(object param)

{

repository.Cancel(SelectedItem); //при редактировании склада, если польз отказался, для отмены изменений в окне

Warehouses = new ObservableCollection<Warehouse>(repository.GetAll());

SelectedItem = null;

}

#region Печать отчета по остаткам

private ICommand \_printReportCommand;

public ICommand PrintReportCommand => \_printReportCommand ?? new RelayCommand(OnPrintReport, (p) =>

{

return SelectedItem != null;

});

private void OnPrintReport(object p)

{

printer.PrintReport(SelectedItem);

}

#endregion

}

}

using IB.DataService;

using IB.DataService.Models;

using IB.DialogsView;

using IB.Excel;

using IB.Models;

using OfficeOpenXml;

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices.ComTypes;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

namespace IB

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

private readonly EFDBContext \_context;

private readonly string PathFile = @"D:\Incidents.xlsx";

public MainWindow()

{

InitializeComponent();

\_context = new EFDBContext();

LoadData();

}

private async void LoadData()

{

try

{

DgIncidents.ItemsSource = await \_context.Incidents.AsNoTracking().Include(x => x.TypeReport).Include(x => x.Division.Region).ToListAsync();

if (ApplicationGlobal.Regions == null)

ApplicationGlobal.Regions = await \_context.Regions.Include(x => x.Divisions).AsNoTracking().ToListAsync();

if (ApplicationGlobal.TypeReports == null)

ApplicationGlobal.TypeReports = await \_context.TypeReports.AsNoTracking().ToListAsync();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private async void test\_Click(object sender, RoutedEventArgs e)

{

using (EFDBContext context = new EFDBContext())

{

var regions = await context.Regions.Include(x => x.Divisions).ToListAsync();

}

}

private void DgIncident\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

}

private void btnCreate\_Click(object sender, RoutedEventArgs e)

{

IncidentWindow incident = new IncidentWindow(new Incident(), "Создать запись");

incident.ShowDialog();

}

private void btnRefresh\_Click(object sender, RoutedEventArgs e)

{

LoadData();

}

private void DgIncidents\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

Incident incident = (Incident)DgIncidents.SelectedItem;

if (incident != null)

{

IncidentWindow incidentWindow = new IncidentWindow(incident, "Редактирование записи");

incidentWindow.ShowDialog();

}

else MessageBox.Show("Ошибка");

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

if (MessageBox.Show("Выгрузить документ? Он будет сохранен на диск D!", "Внимание!", MessageBoxButton.YesNo) == MessageBoxResult.Yes)

{

List<Incident> items = \_context.Incidents

.Include(s => s.Division.Region)

.Include(s => s.TypeReport)

.ToList();

byte[] res = ExcelIncident.Generate(items);

FileStream aFile = new FileStream(PathFile, FileMode.Create);

aFile.Seek(0, SeekOrigin.Begin);

aFile.Write(res, 0, res.Length);

aFile.Close();

}

return;

}

private void btnRemove\_Click(object sender, RoutedEventArgs e)

{

if (MessageBox.Show("Вы действительно хотите удалить запись?", "Внимание", MessageBoxButton.YesNo) == MessageBoxResult.Yes)

{

Incident incident = \_context.Incidents.FirstOrDefault(s => s.Id == ((Incident)DgIncidents.SelectedItem).Id);

if (incident == null)

{

MessageBox.Show("Данной записи не существует", "Внимание", MessageBoxButton.OK);

return;

}

\_context.Entry(incident).State = EntityState.Deleted;

\_context.SaveChanges();

}

return;

}

}

}

<Window x:Class="IB.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:IB"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

TextElement.FontWeight="Medium"

TextElement.FontSize="14"

FontFamily="{materialDesign:MaterialDesignFont}"

Background="{DynamicResource MaterialDesignPaper}"

mc:Ignorable="d"

WindowState="Maximized"

Title="MainWindow" Height="768" Width="1380" MinHeight="768" MinWidth="1380">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="150"/>

<ColumnDefinition Width="150"></ColumnDefinition>

<ColumnDefinition Width="434\*"></ColumnDefinition>

<ColumnDefinition Width="50"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="56"></RowDefinition>

<RowDefinition/>

</Grid.RowDefinitions>

<Button HorizontalAlignment="Center" VerticalAlignment="Center" Grid.Column="1" Grid.Row="0" Style="{StaticResource MaterialDesignRaisedDarkButton}" Background="Green" Content="Выгрузка в Excel" Click="Button\_Click" />

<Button HorizontalAlignment="Center" VerticalAlignment="Center" Grid.Column="0" Grid.Row="0" Style="{StaticResource MaterialDesignRaisedDarkButton}" Content="Создать запись" x:Name="btnCreate" Click="btnCreate\_Click"

IsEnabled="{Binding DataContext.ControlsEnabled, RelativeSource={RelativeSource FindAncestor, AncestorType=Window}}" />

<Button HorizontalAlignment="Right" VerticalAlignment="Center" Grid.Column="2" x:Name="btnRemove" Style="{StaticResource MaterialDesignRaisedDarkButton}" Click="btnRemove\_Click" Background="#ff6551" Margin="3,9" Height="Auto">Удалить</Button>

<Button HorizontalAlignment="Center" VerticalAlignment="Center" x:Name="btnRefresh" Click="btnRefresh\_Click" Grid.Column="3" Grid.Row="0" Style="{StaticResource MaterialDesignFloatingActionMiniSecondaryDarkButton}"

ToolTip="MaterialDesignFloatingActionMiniSecondaryDarkButton" IsEnabled="{Binding DataContext.ControlsEnabled, RelativeSource={RelativeSource FindAncestor, AncestorType=Window}}">

<materialDesign:PackIcon Kind="Refresh" Height="25" Width="30" />

</Button>

<DataGrid Grid.Row="1" Grid.ColumnSpan="4" SelectionMode="Single" CanUserSortColumns="False" CanUserResizeColumns="True" IsReadOnly="True" CanUserAddRows="False" SelectionChanged="DgIncident\_SelectionChanged" AutoGenerateColumns="False" x:Name="DgIncidents" Background="{x:Null}" MouseDoubleClick="DgIncidents\_MouseDoubleClick">

<DataGrid.CellStyle>

<Style TargetType="{x:Type DataGridCell}">

<Style.Triggers>

<Trigger Property="IsSelected" Value="True">

<Setter Property="Background" Value="{x:Static SystemColors.HighlightBrush}"></Setter>

<Setter Property="Foreground" Value="{x:Static SystemColors.HighlightTextBrush}"></Setter>

</Trigger>

</Style.Triggers>

</Style>

</DataGrid.CellStyle>

<DataGrid.Columns>

<DataGridTextColumn Header="Область" Binding="{Binding Division.Region.Name}" Width="100"

EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnEditingStyle}" >

<DataGridTextColumn.ElementStyle>

<Style>

<Setter Property="TextBlock.VerticalAlignment" Value="Top" />

<Setter Property="TextBlock.HorizontalAlignment" Value="Center" />

</Style>

</DataGridTextColumn.ElementStyle>

</DataGridTextColumn>

<DataGridTextColumn Header="Подразделение" Binding="{Binding Division.Name}" Width="200"

EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnEditingStyle}">

<DataGridTextColumn.ElementStyle>

<Style>

<Setter Property="TextBlock.VerticalAlignment" Value="Top" />

<Setter Property="TextBlock.HorizontalAlignment" Value="Center" />

</Style>

</DataGridTextColumn.ElementStyle>

</DataGridTextColumn>

<DataGridTextColumn Header="Вид КИ" Binding="{Binding TypeReport}" Width="200"

EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnEditingStyle}">

<DataGridTextColumn.ElementStyle>

<Style>

<Setter Property="TextBlock.VerticalAlignment" Value="Top" />

<Setter Property="TextBlock.HorizontalAlignment" Value="Center" />

</Style>

</DataGridTextColumn.ElementStyle>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Date,StringFormat=\{0:dd.MM.yyy\}}" Header="Дата" Width="80"

EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnEditingStyle}">

<DataGridTextColumn.ElementStyle>

<Style>

<Setter Property="TextBlock.VerticalAlignment" Value="Top" />

<Setter Property="TextBlock.HorizontalAlignment" Value="Center" />

</Style>

</DataGridTextColumn.ElementStyle>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Time,StringFormat=\{0:HH:mm\}}" Header="Время" Width="80"

EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnEditingStyle}">

<DataGridTextColumn.ElementStyle>

<Style>

<Setter Property="TextBlock.VerticalAlignment" Value="Top" />

<Setter Property="TextBlock.HorizontalAlignment" Value="Center" />

</Style>

</DataGridTextColumn.ElementStyle>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding IncidentData}" Header="Данные по инциденту" Width="250\*"

EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnEditingStyle}">

<DataGridTextColumn.ElementStyle>

<Style>

<Setter Property="TextBlock.TextWrapping" Value="Wrap" />

</Style>

</DataGridTextColumn.ElementStyle>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding Description}" Header="Фабула" Width="250"

EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnEditingStyle}">

<DataGridTextColumn.ElementStyle>

<Style>

<Setter Property="TextBlock.TextWrapping" Value="Wrap" />

</Style>

</DataGridTextColumn.ElementStyle>

</DataGridTextColumn>

<DataGridTextColumn Binding="{Binding MeasuresTaken}" Header="Приняты меры" Width="250"

EditingElementStyle="{StaticResource MaterialDesignDataGridTextColumnEditingStyle}">

<DataGridTextColumn.ElementStyle>

<Style>

<Setter Property="TextBlock.TextWrapping" Value="Wrap" />

</Style>

</DataGridTextColumn.ElementStyle>

</DataGridTextColumn>

</DataGrid.Columns>

</DataGrid>

</Grid>

</Window>

using IB.DataService;

using IB.DataService.Models;

using IB.Models;

using System;

using System.Data.Entity;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

namespace IB.DialogsView

{

/// <summary>

/// Логика взаимодействия для IncidentWindow.xaml

/// </summary>

public partial class IncidentWindow : Window

{

private Incident \_incident;

public IncidentWindow(Incident incident, string title)

{

InitializeComponent();

\_incident = incident;

LoadData();

this.Title = title;

}

private void LoadData()

{

cmbRegion.ItemsSource = ApplicationGlobal.Regions.ToList();

cmbType.ItemsSource = ApplicationGlobal.TypeReports.ToList();

if (\_incident.Id != 0)

{

cmbRegion.SelectedValue = \_incident.Division.RegionId;

cmbDivision.SelectedValue = \_incident.DivisionId;

cmbType.SelectedValue = \_incident.TypeReportId;

txtIncidentData.Text = \_incident.IncidentData;

txtDescription.Text = \_incident.Description;

txtMeasuresTaken.Text = \_incident.MeasuresTaken;

pickDate.SelectedDate = \_incident.Date;

pickTime.SelectedTime = \_incident.Time;

btnRemove.Visibility = Visibility.Visible;

}

else btnRemove.Visibility = Visibility.Collapsed;

}

private void PresetTimePicker\_SelectedTimeChanged(object sender, RoutedPropertyChangedEventArgs<DateTime?> e)

{

}

private void cmbRegion\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

Region region = (Region)cmbRegion.SelectedItem;

if (region != null)

cmbDivision.ItemsSource = region.Divisions.ToList();

}

private async void btnSave\_Click(object sender, RoutedEventArgs e)

{

var date = pickDate.SelectedDate;

Division division = (Division)cmbDivision.SelectedItem;

TypeReport type = (TypeReport)cmbType.SelectedItem;

if (division != null && type != null && pickDate.SelectedDate != null && pickTime.SelectedTime != null

&& !string.IsNullOrEmpty(txtIncidentData.Text)

&& !string.IsNullOrEmpty(txtDescription.Text)

&& !string.IsNullOrEmpty(pickTime.Text)

&& !string.IsNullOrEmpty(txtMeasuresTaken.Text))

{

using (EFDBContext context = new EFDBContext())

{

if (\_incident.Id == 0)

{

context.Incidents.Add(

new Incident

{

DivisionId = division.Id,

Date = (DateTime)pickDate.SelectedDate,

Time = (DateTime)pickTime.SelectedTime,

Description = txtDescription.Text,

IncidentData = txtIncidentData.Text,

MeasuresTaken = txtMeasuresTaken.Text,

TypeReportId = type.Id,

DateCreateModify = DateTime.Now

});

}

else

{

var modelFromdb = await context.Incidents.FirstOrDefaultAsync(x => x.Id == \_incident.Id);

if (modelFromdb != null)

{

modelFromdb.Date = (DateTime)pickDate.SelectedDate;

modelFromdb.Time = (DateTime)pickTime.SelectedTime;

modelFromdb.Description = txtDescription.Text;

modelFromdb.IncidentData = txtIncidentData.Text;

modelFromdb.MeasuresTaken = txtMeasuresTaken.Text;

modelFromdb.TypeReportId = type.Id;

modelFromdb.DateCreateModify = DateTime.Now;

context.Entry(modelFromdb).State = EntityState.Modified;

}

}

await context.SaveChangesAsync();

}

}

else MessageBox.Show("Ошибка при заполнении формы");

}

private void btnRemove\_Click(object sender, RoutedEventArgs e)

{

this.DialogResult = true;

}

}

}

<Window x:Class="IB.DialogsView.IncidentWindow"

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:IB.DialogsView"

mc:Ignorable="d"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

TextElement.FontWeight="Medium"

TextElement.FontSize="17"

Foreground="White"

FontFamily="{materialDesign:MaterialDesignFont}"

Background="{DynamicResource MaterialDesignPaper}"

Title="Данные инцидента" MinHeight="768" MinWidth="1380"

WindowState="Maximized">

<GroupBox Header="Данные по инциденту" Style="{DynamicResource MaterialDesignGroupBox}" UseLayoutRounding="True" Margin="5">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="61\*"></ColumnDefinition>

<ColumnDefinition Width="58\*"></ColumnDefinition>

<ColumnDefinition Width="64\*"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="45"></RowDefinition>

<RowDefinition Height="56"></RowDefinition>

<RowDefinition Height="285\*"></RowDefinition>

<RowDefinition Height="50\*"></RowDefinition>

<RowDefinition Height="2\*" />

</Grid.RowDefinitions>

<ComboBox SelectedValuePath="Id" SelectionChanged="cmbRegion\_SelectionChanged" x:Name="cmbRegion" Grid.Row="0" Grid.Column="0" materialDesign:HintAssist.Hint="Область" Margin="3"/>

<ComboBox SelectedValuePath="Id" x:Name="cmbDivision" Grid.Row="0" Grid.Column="1" materialDesign:HintAssist.Hint="Подразделение" Margin="3"/>

<ComboBox SelectedValuePath="Id" x:Name="cmbType" Grid.Row="0" Grid.Column="2" materialDesign:HintAssist.Hint="Вид КИ" Margin="3"/>

<DatePicker x:Name="pickDate" Grid.Column="0" Grid.Row="1" materialDesign:HintAssist.Hint="Дата" Style="{StaticResource MaterialDesignFloatingHintDatePicker}" Margin="3,7" Height="42"/>

<materialDesign:TimePicker x:Name="pickTime" Margin="3,7" Grid.Row="1" Grid.Column="1" Style="{StaticResource MaterialDesignFloatingHintTimePicker}" Is24Hours="True" materialDesign:HintAssist.Hint="Время" Height="42" />

<TextBox x:Name="txtIncidentData" Grid.Row="2" Grid.Column="0" Margin="3" Style="{StaticResource MaterialDesignOutlinedTextBox}"

TextWrapping="Wrap" materialDesign:TextFieldAssist.CharacterCounterVisibility="{Binding Path=IsChecked, ElementName=MaterialDesignOutlinedPasswordFieldTextCountComboBox, Converter={StaticResource BooleanToVisibilityConverter}}"

VerticalScrollBarVisibility="Auto" materialDesign:HintAssist.Hint="Данные по инциденту" />

<TextBox x:Name="txtDescription" Grid.Row="2" Grid.Column="1" Margin="3" Style="{StaticResource MaterialDesignOutlinedTextBox}"

TextWrapping="Wrap" materialDesign:TextFieldAssist.CharacterCounterVisibility="{Binding Path=IsChecked, ElementName=MaterialDesignOutlinedPasswordFieldTextCountComboBox, Converter={StaticResource BooleanToVisibilityConverter}}"

VerticalScrollBarVisibility="Auto" materialDesign:HintAssist.Hint="Фабула" />

<TextBox x:Name="txtMeasuresTaken" Grid.Row="2" Grid.Column="2" Margin="3" Style="{StaticResource MaterialDesignOutlinedTextBox}"

TextWrapping="Wrap" materialDesign:TextFieldAssist.CharacterCounterVisibility="{Binding Path=IsChecked, ElementName=MaterialDesignOutlinedPasswordFieldTextCountComboBox, Converter={StaticResource BooleanToVisibilityConverter}}"

VerticalScrollBarVisibility="Auto" materialDesign:HintAssist.Hint="Приняты меры" />

<Grid Grid.Row="3" Grid.Column="2">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="117\*"></ColumnDefinition>

<ColumnDefinition Width="139\*"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Button Grid.Column="0" FontSize="20" x:Name="btnRemove" Click="btnRemove\_Click" Background="#ff6551" Style="{StaticResource MaterialDesignRaisedDarkButton}" Margin="3,9" Height="Auto">Удалить</Button>

<Button Grid.Column="1" FontSize="20" x:Name="btnSave" Click="btnSave\_Click" Background="#66bb6a" Style="{StaticResource MaterialDesignRaisedDarkButton}" Margin="3,9" Height="Auto">Сохранить</Button>

</Grid>

</Grid>

</GroupBox>

</Window>

# ПРИЛОЖЕНИЕ Б

# (справочное)

**Результаты функционального тестирования**

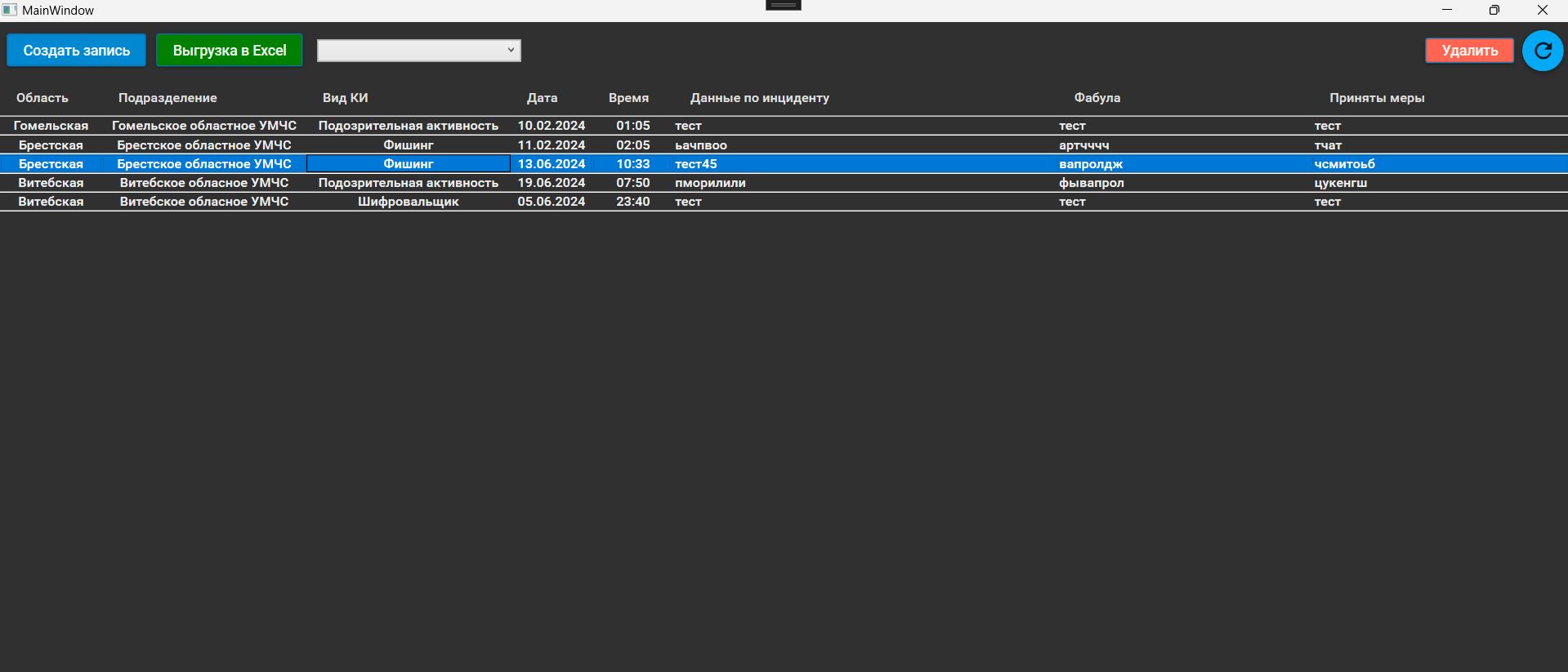


Рисунок Б.1 – Выбор удаляемого объекта

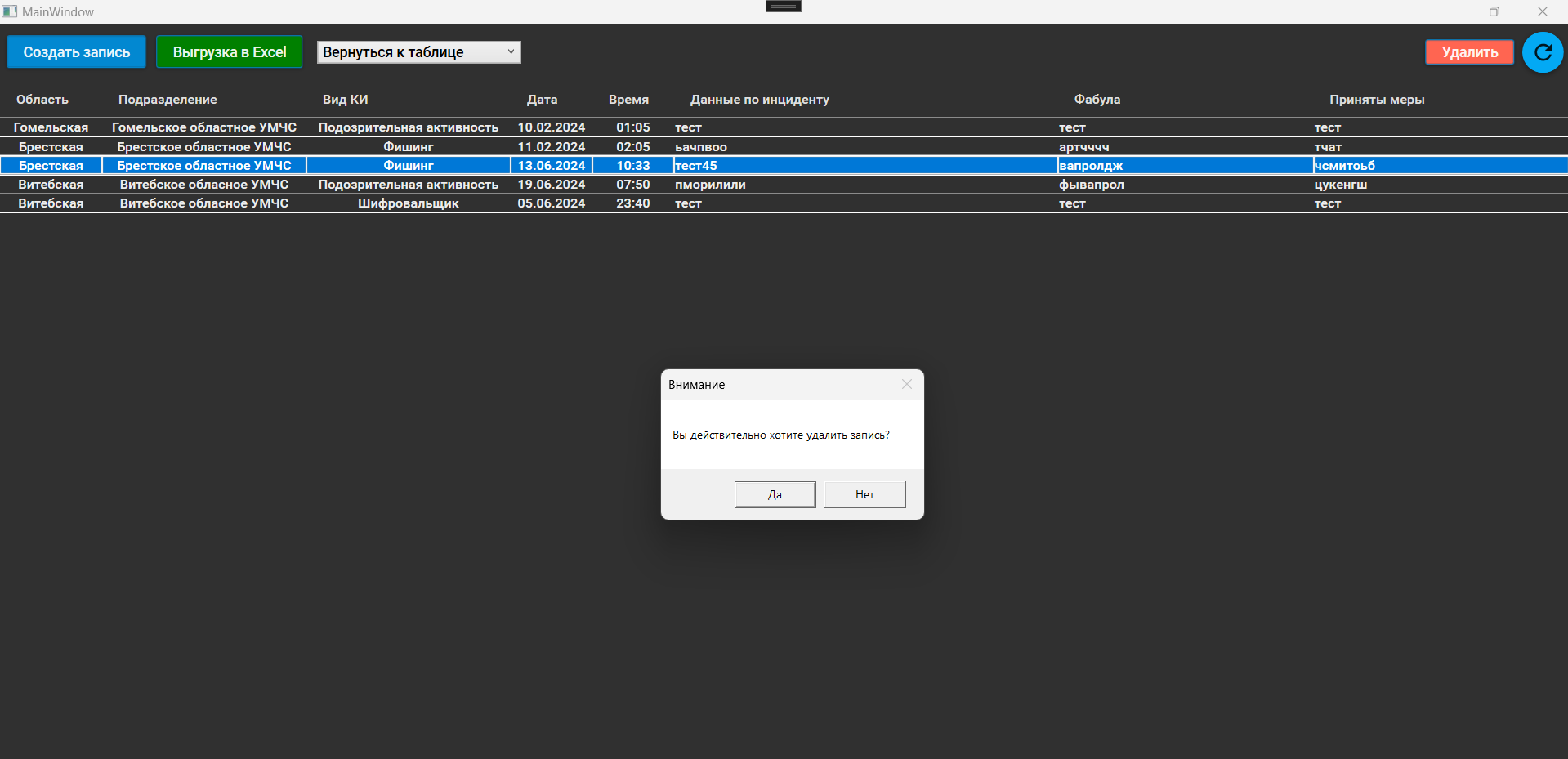


Рисунок Б.2 – Соглашение на удаление объекта

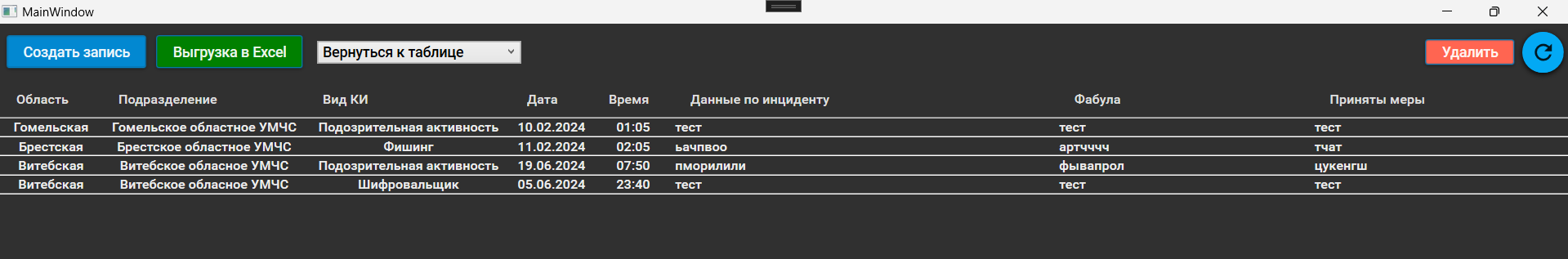


Рисунок Б.3 – Отображение списка инциндентов после удаления

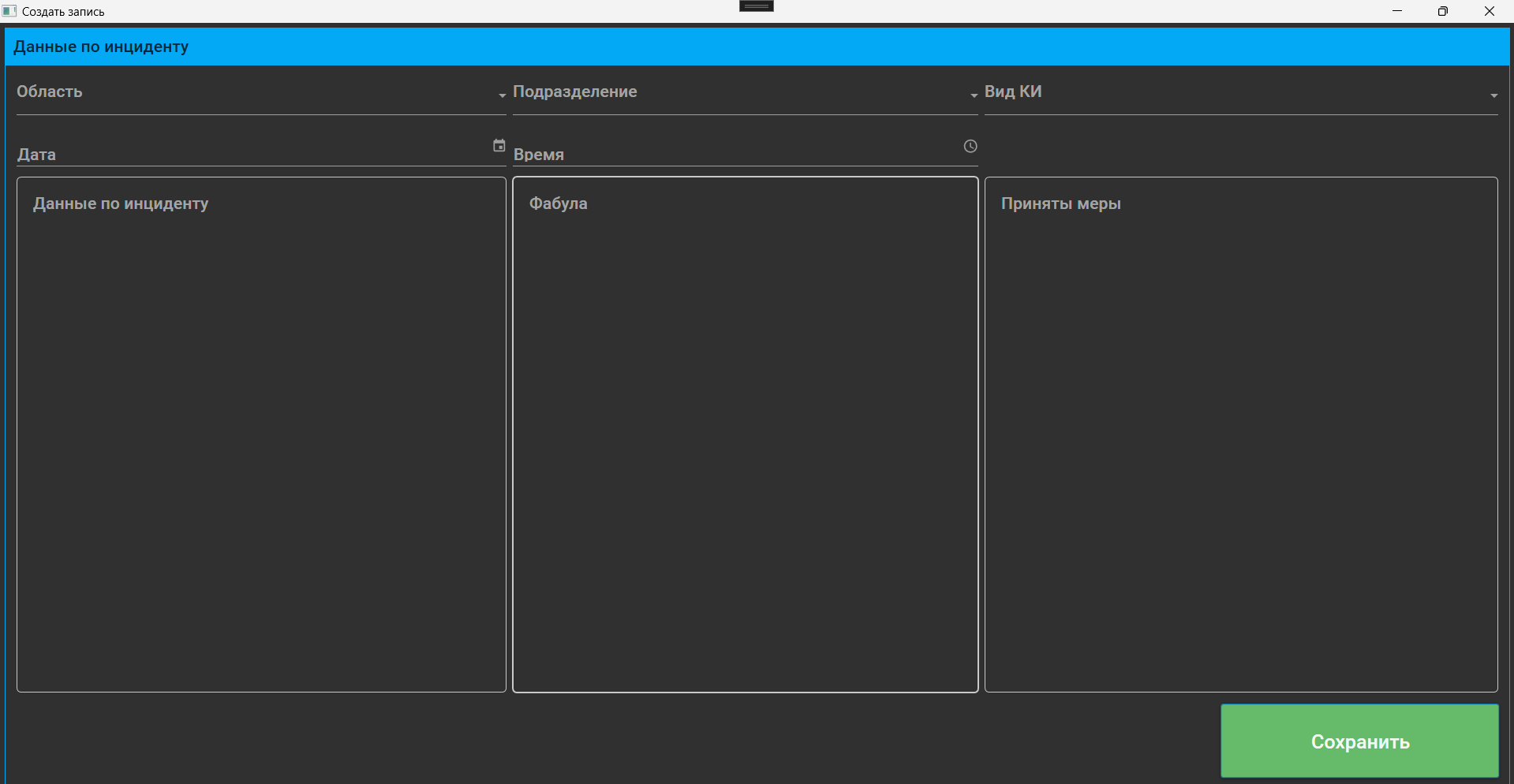


Рисунок Б.4 – Открытие окна добавления новой записи

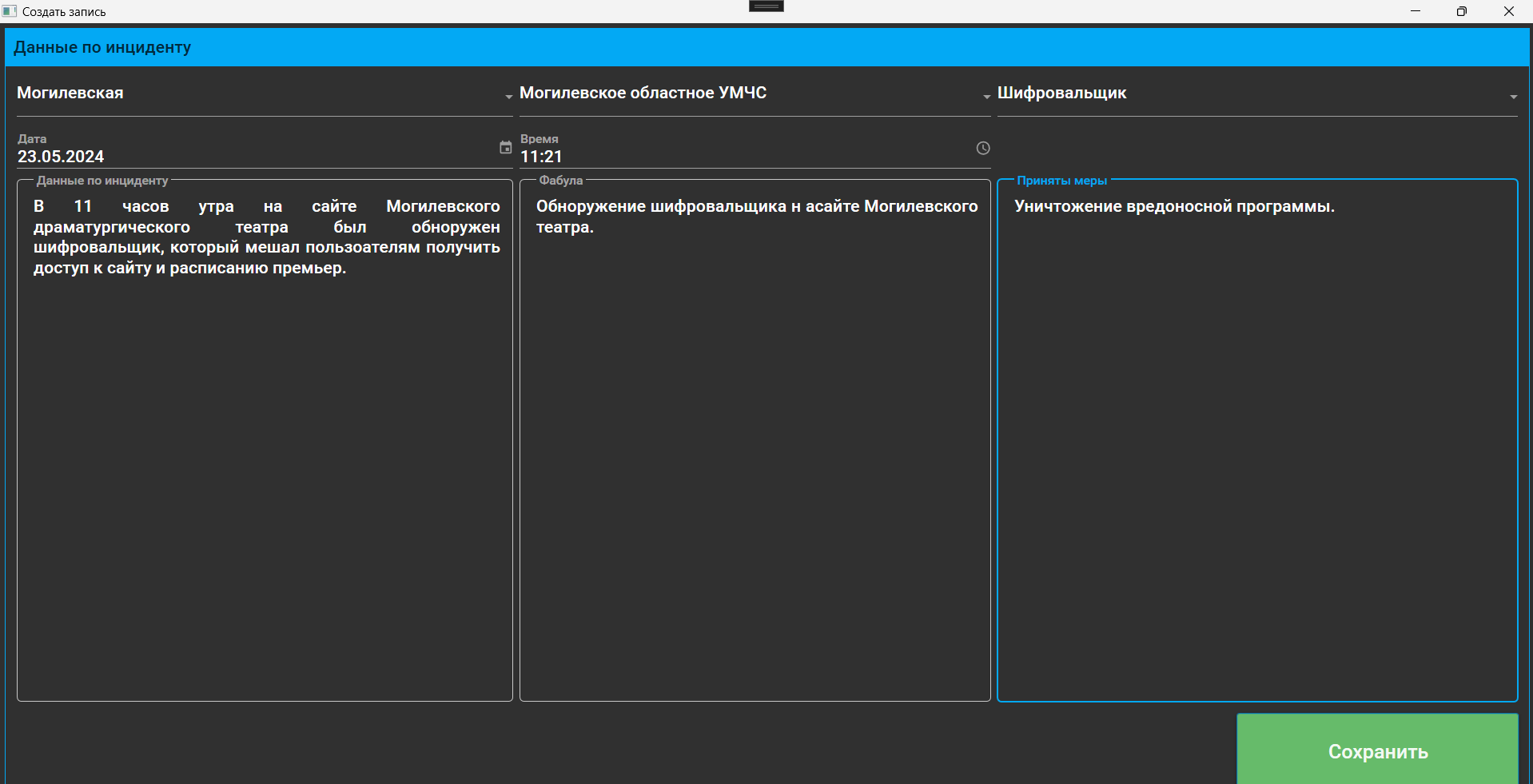


Рисунок Б.9 – Заполнение полей для создания новой записи

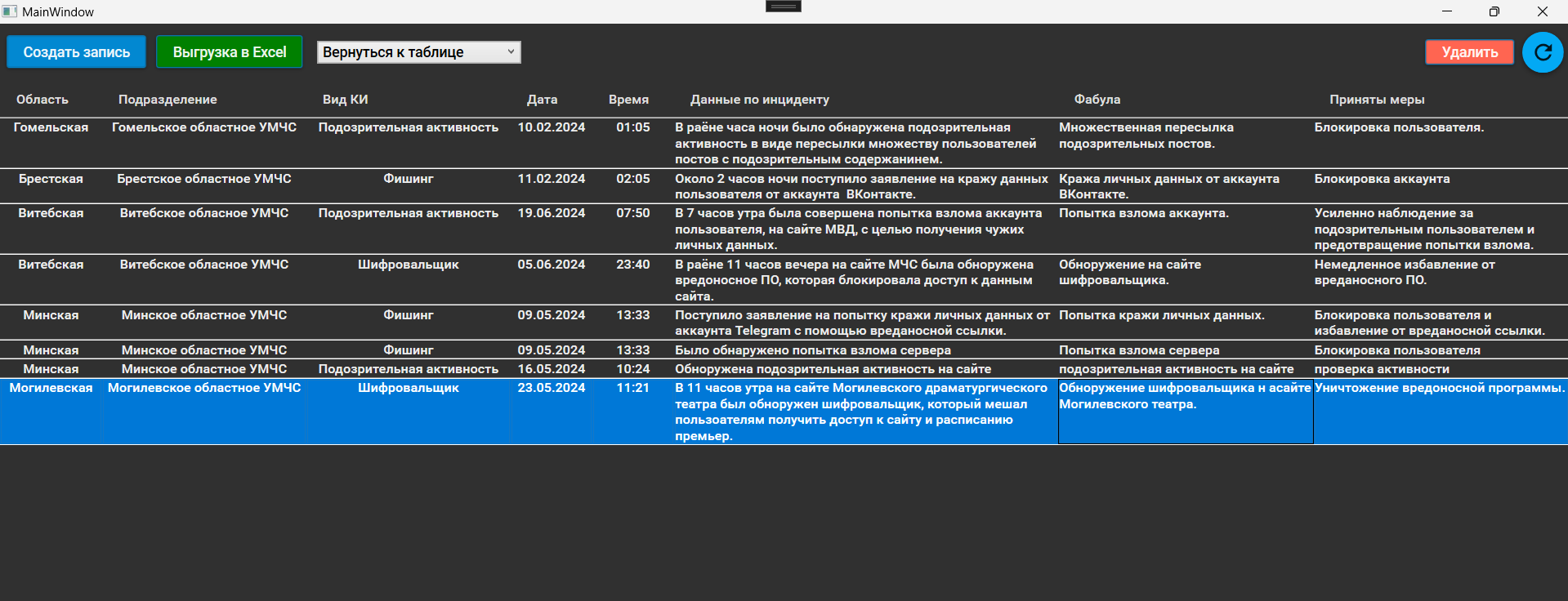


Рисунок Б.10 – Сохранение записи

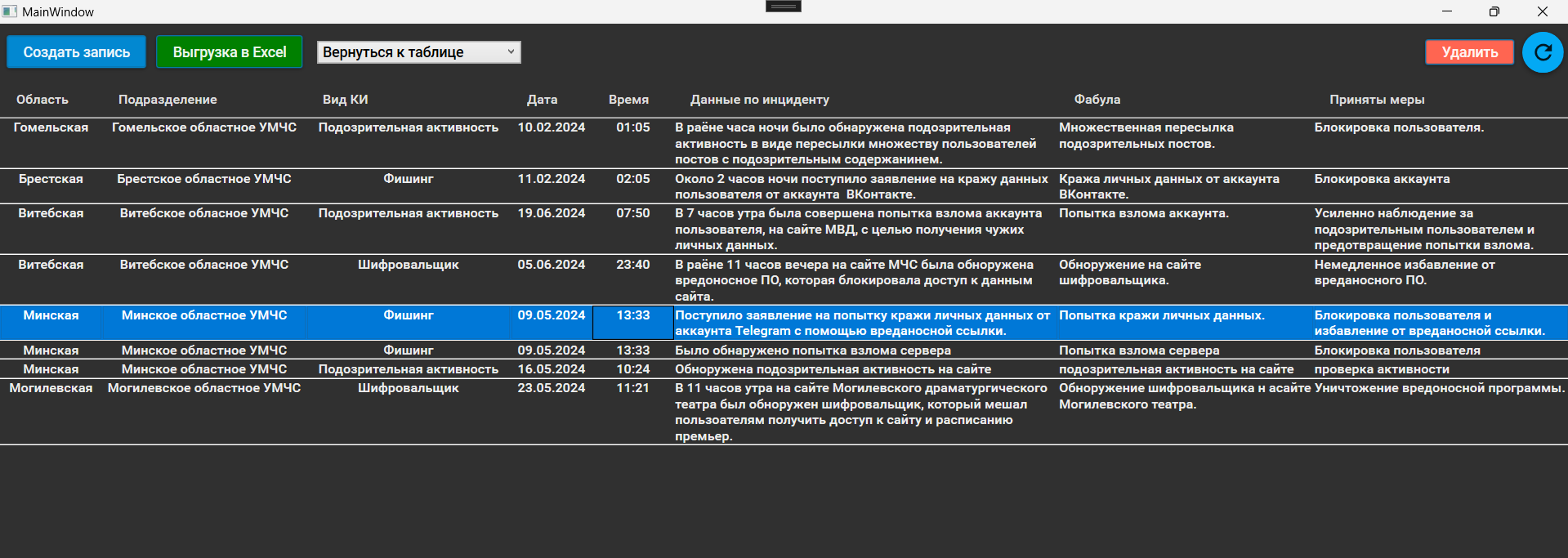


Рисунок Б.11 – Выбор редактируемой записи

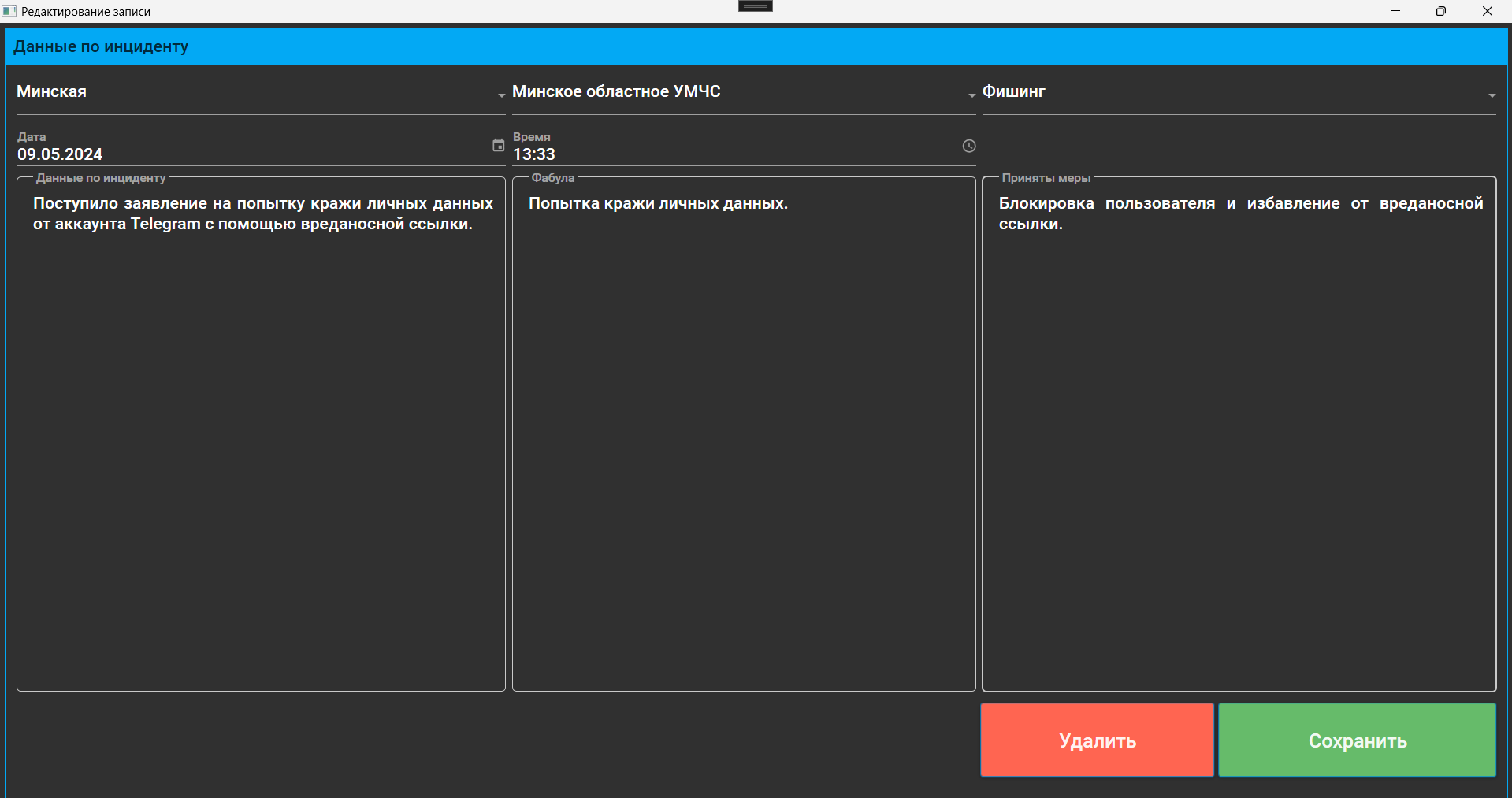


Рисунок Б.12 – Двойным кликом на запись открытие окна редактирования

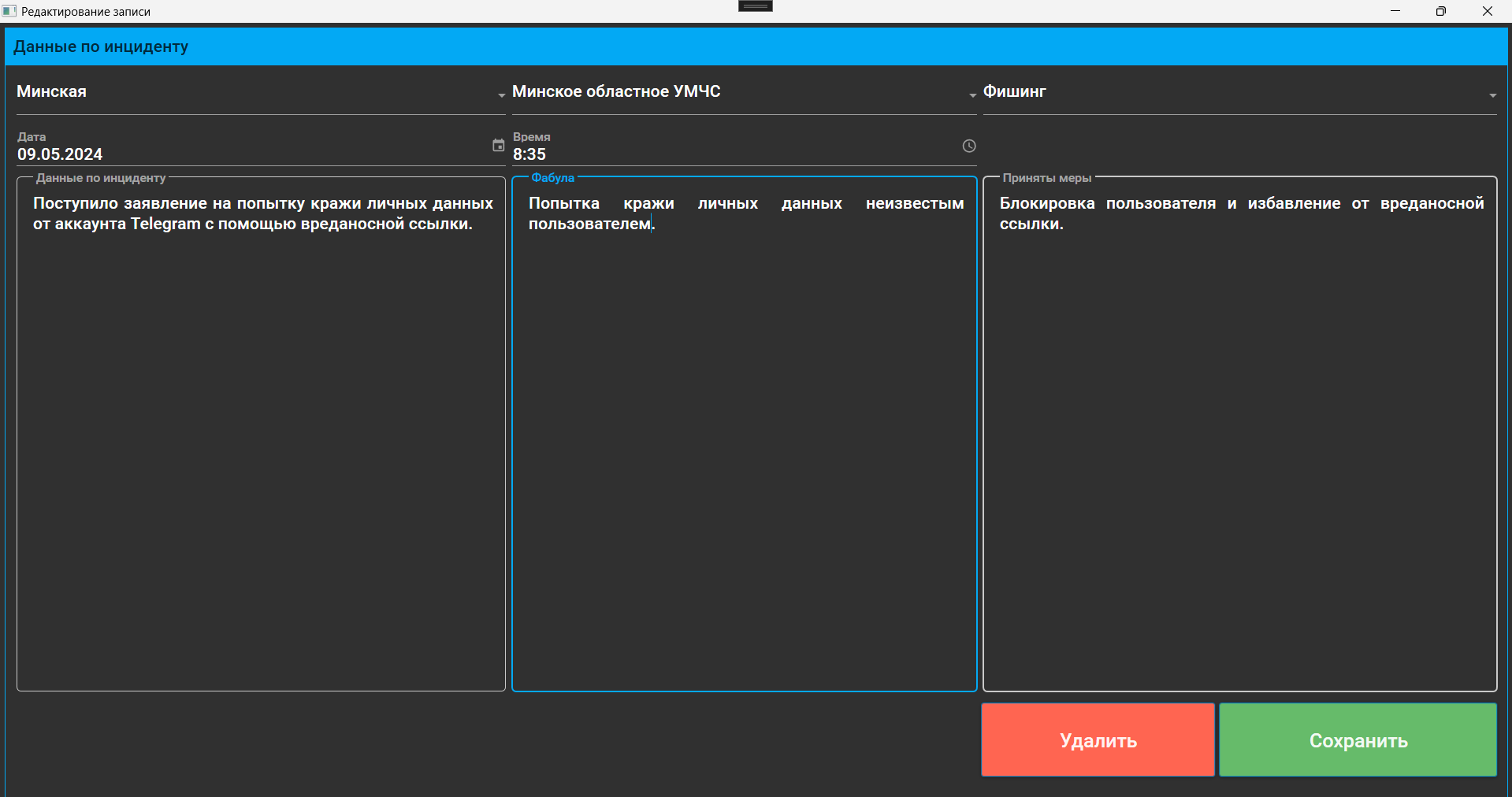


Рисунок Б.13 – Изменение и сохранение данных

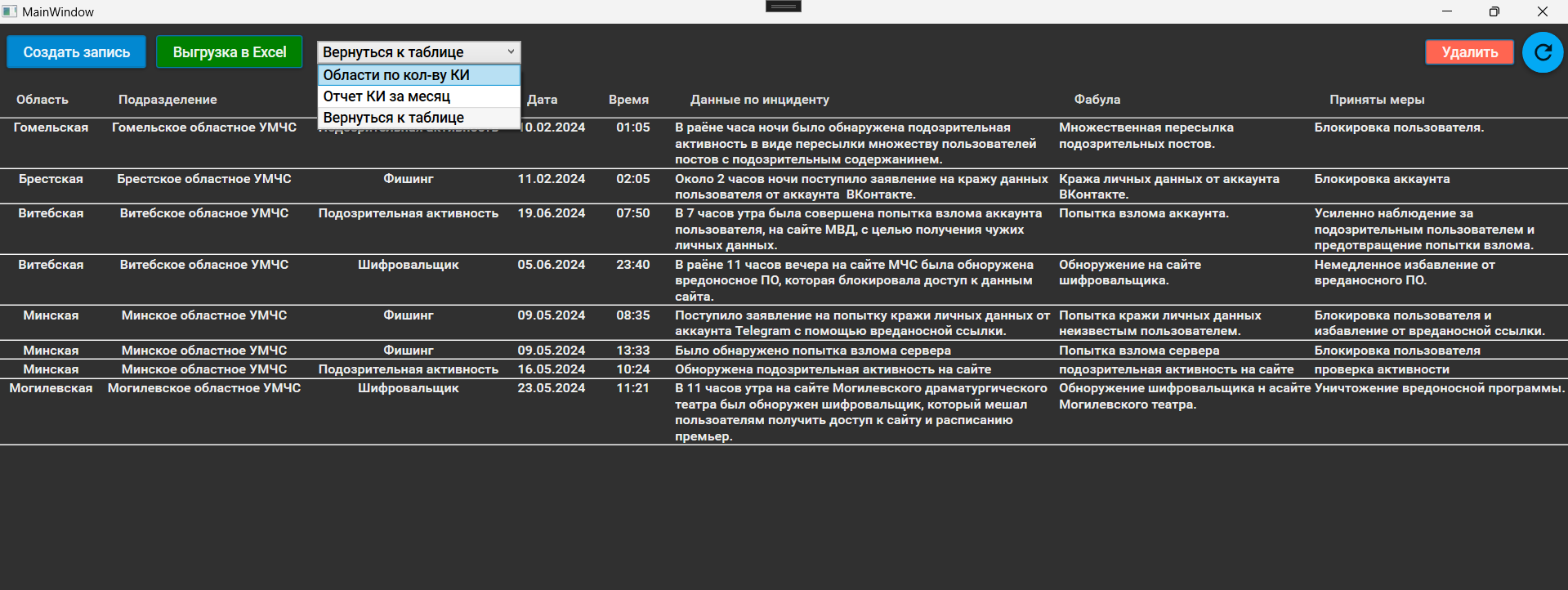


Рисунок Б.14 – Выбор пункта «Области по кол-ву КИ»

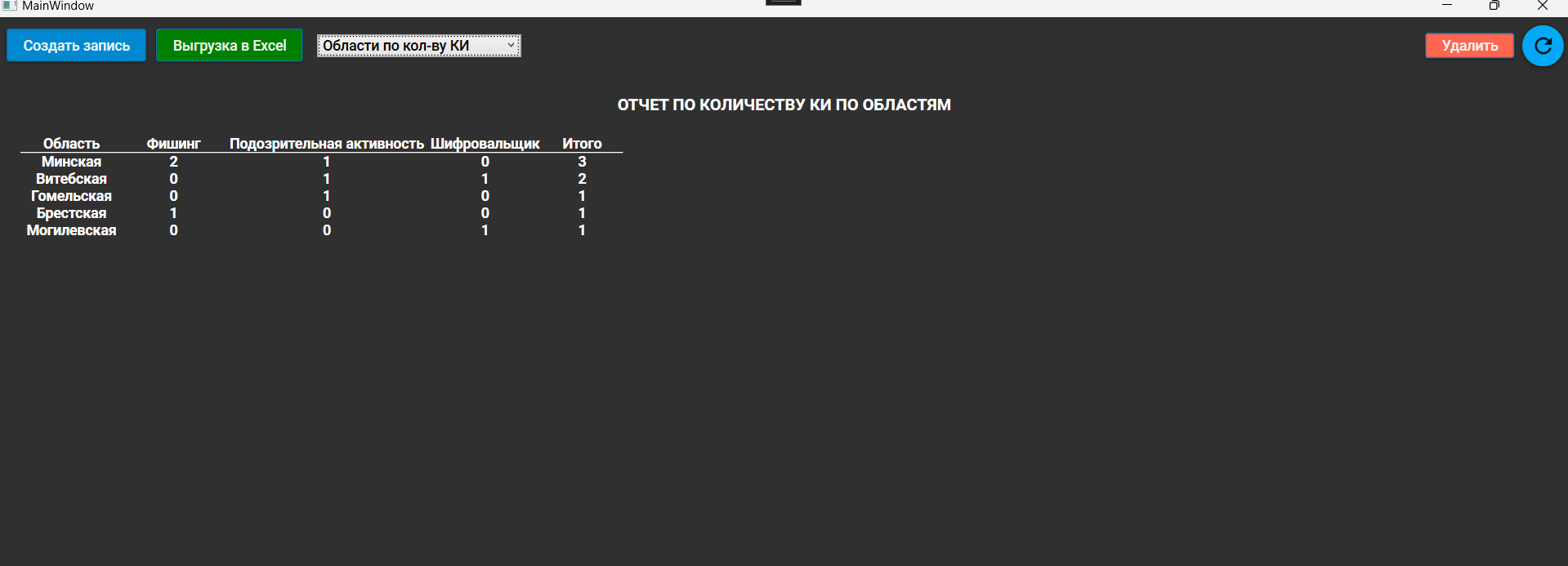


Рисунок Б.15 – Отображение отчета количества К

КП Т.219015.401 ГЧ

КБиП

У

Т. Контр.

Разраб. Разраб.

Писарик А.В.

Провер. Провер.

Багласова Е.В.

Реценз.

Утверд.

Н. Контр.

Разработка программного средства для учета киберинцидентов

Диаграмма классов

№ докум.

Подпись

Дата

Изм.

Лист

Масса

Лит.

Масштаб

Лист 2

Листов 2

Инв.№подл.

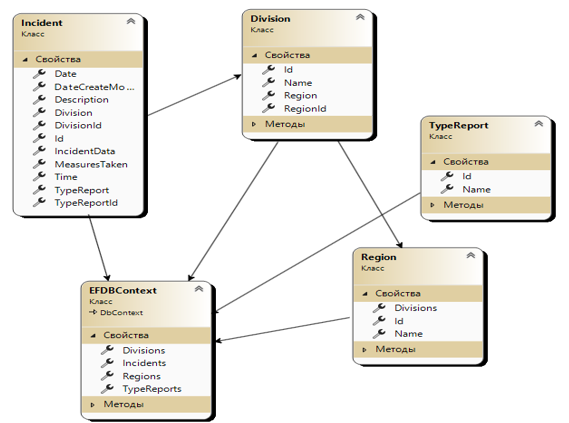
Подп. и дата

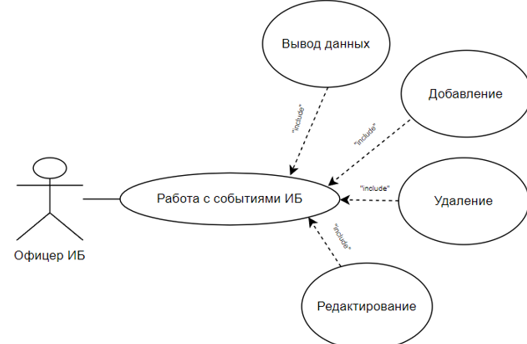
Взам.инв.№

Инв.№дубл.

Подп. и дата

КП Т.219015.401 ГЧ



****

КП Т.219015.401 ГЧ

КБиП

У

Т. Контр.

Разраб. Разраб.

Писарик А.В.

Провер. Провер.

Багласова Е.В.

Реценз.

Утверд.

Н. Контр.

Разработка программного средства для учета киберинцидентов

Диаграмма вариантов

использования

№ докум.

Подпись

Дата

Изм.

Лист

Масса

Лит.

Масштаб

Лист 2

Листов 2

Инв.№подл.

Подп. и дата

Взам.инв.№

Инв.№дубл.

Подп. и дата

КП Т.219015.401 ГЧ

**Удостоверяющий лист**

электронного документа – курсовой проект

Тема «Разработка программного средства для учета киберинцидентов».

Обозначение \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ КП Т.219015.401\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Разработчик \_Писарик А.В.\_\_\_ Руководитель БагласоваЕ.В.\_\_

(Ф.И.О.) (Ф.И.О.)

Подписи лиц, ответственных за разработку электронного документа

|  |  |  |
| --- | --- | --- |
| Состав электронного документа | Разработчик | Руководитель |
| Пояснительная записка (на бумажном носителе формата А4), Пояснительная записка КИ.docх |  |  |
| Папка с проектом «C# курсовая3» |  |  |
| Тип носителя: оптический диск |  |  |

**Этикетка**

для курсовых проектов

**Курсовой проект**

Тема «Разработка программного средства для учета киберинцидентов»

КП Т.219015.401

Разработан А.В.Писарик

Утвержден

Руководитель: Е.В.Багласова

Технические средства:

Программные средства:

**Состав документа:**

Пояснительная записка – Пояснительная записка КИ.docх

Папка с проектом – C# курсовая3