МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Дальневосточный федеральный университет»**

**ШКОЛА ЕСТЕСТВЕННЫХ НАУК**

**Кафедра информационных систем управления**

Константинов Остап Владимирович

Разработка базы данных «ГИБДД»

**Курсовой проект**

|  |
| --- |
| Студент гр. Б8319  О.В. Константинов \_\_\_\_\_\_\_\_\_\_\_  подпись |
| Руководитель доцент  Л.В. Красюк \_\_\_\_\_\_\_\_\_\_\_  подпись |
| Регистрационный № \_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись И.О. Фамилия  « \_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г. | Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись И.О. Фамилия  « \_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г. |

г. Владивосток

2018

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Дальневосточный федеральный университет»**

|  |
| --- |
| **ШКОЛА ЕСТЕСТВЕННЫХ НАУК** |

**Кафедра информационных систем управления**

**Задание**

на курсовой проект

|  |
| --- |
| студенту Константинову О. В. группы Б8319 |

(фамилия, имя, отчество)

на тему Разработка базы данных «ГИБДД»

|  |
| --- |
|  |
|  |

Вопросы, подлежащие разработке (исследованию):

|  |
| --- |
| разработка базы данных; |
| анализ предметной области и ее реализация инструментальными средствами; |
| основы проектирование таблиц, отношение между таблицами; |
| методы проектирования и реализации запросов; |
| реализация интерфейса пользователя. |
|  |
|  |
|  |

Основные источники информации и прочее, используемые для разработки темы

|  |
| --- |
| Базы данных: теория и практика: Учебник для вузов/ Б. Я. Советов, |
| В. В. Цехановский, В. Д. Чертовской. – М. : Высшая школа, 2005. – 462 с. |
| Диго, С. М. Базы данных: проектирование и использование: Учебник/ С. М. Диго. |
| – М. : Финансы и статистика, 2005. |

Срок представления работы «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г.

Дата выдачи задания «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г.

Оглавление

[Введение 3](#_Toc518006158)

[2 Описание предметной области 5](#_Toc518006159)

[3 Глоссарий основных терминов 6](#_Toc518006160)

[4 Анализ предметной области 7](#_Toc518006161)

[4.1 Структура подразделений ГИБДД и их подчиненность 7](#_Toc518006162)

[4.2 Бизнес-модель ГИБДД 8](#_Toc518006163)

[4.3 Потоки данных ГИБДД 10](#_Toc518006164)

[5 Постановка задачи 12](#_Toc518006165)

[6 Обоснование решаемой задачи 13](#_Toc518006166)

[7 Назначение базы данных 15](#_Toc518006167)

[7.1 Описание выходной информации 15](#_Toc518006168)

[7.2 Описание входной информации 15](#_Toc518006169)

[8 Проектирование базы данных 17](#_Toc518006170)

[8.1 Концептуальное проектирование 17](#_Toc518006171)

[8.2 Логическое проектирование 21](#_Toc518006172)

[8.3 Физическое проектирование 22](#_Toc518006173)

[9 Клиентское приложение 26](#_Toc518006174)

[Заключение 31](#_Toc518006175)

[Список литературы 32](#_Toc518006176)

[Приложение А 33](#_Toc518006177)

[A.1 Обобщенное описание работы программы 33](#_Toc518006178)

[A.2 Заполнение справочников 34](#_Toc518006179)

[A.3 Регистрация ДТП 36](#_Toc518006180)

[A.4 Печать бланков постановлений 37](#_Toc518006181)

[A.5 Статистика по сотрудникам и виновникам 39](#_Toc518006182)

[Приложение Б 40](#_Toc518006183)

[B.1 Модуль метаданных таблиц 40](#_Toc518006185)

[B.2 Главная форма 44](#_Toc518006186)

[B.3 Унифицированная форма справочников 45](#_Toc518006187)

[B.4 Унифицированная форма редактирования записей 53](#_Toc518006188)

[B.5 Форма регистрации ДТП 57](#_Toc518006189)

[B.6 Форма печати бланков постановлений 66](#_Toc518006190)

# Введение

На сегодняшний день применение программных средств получило весьма большое значение для многих организаций, которые для упрощения своей работы применяют компьютерные технологии.

Любая организация имеет в процессе своего функционирования большое количество информации, требующей организации и учёта. Не исключением является и деятельность Государственной инспекции безопасности дорожного движения.

Основу дорожного инспектирования ГИБДД составляет регистрация административных правонарушений. Основная цель протоколирования – выявление и регистрация правонарушения. Анализ составленных протоколов предупреждает значительную часть правонарушений и несчастных случаев, связанных с ними, так как указывает инспекторам на какую именно область правонарушений необходимо обратить особое внимание.

Объектом анализ данного курсового проекта является деятельность ГИБДД.

В качестве предмета исследования выступает деятельность сотрудников дорожного движения.

Задачей выполнения курсового проекта является разработка проекта программного средства и реализация БД для автоматизации работы ГИБДД. А так же, приобретение практических навыков обследования предметной области, концептуального, логического и физического проектирования базы данных.

Цель курсового проекта – закрепление знаний, полученных при изучении дисциплины, а также получение практических навыков проектирования и реализации БД с использованием современных технологий и инструментальных средств.

# Описание предметной области

ГИБДД занимается учетом и анализом ДТП (дорожно-транспортное происшествие). При регистрации ДТП фиксируется: дата, тип происшествия (наезд на пешехода, наезд на ограждение либо столб, лобовое столкновение, наезд на впереди стоящий транспорт, боковое столкновение на перекрестке и т.п.), место происшествия, марки пострадавших автомобилей, государственный номер, тип машины (легковая, грузовая, специальная), краткое содержание, число пострадавших, сумма ущерба, причина, дорожные условия и т.п. Анализ накопленной по ДТП статистике поможет правильно расставить запрещающие и предупреждающие знаки на улицах города, а также спланировать местонахождение постов патрульных.

Дорожно-патрульная служба – крупнейшая и наиболее многофункциональная служба в составе Государственной инспекции безопасности дорожного движения МВД России. ДПС – это специфическое направление обеспечения безопасности дорожного движения путем реализации комплекса правовых, технических и естественнонаучных знаний.

Основными задачами дорожно-патрульной службы являются:

* сохранение жизни, здоровья и имущества участников дорожного движения, защита их законных прав и интересов, а также интересов общества и государства;
* обеспечение безопасного и бесперебойного движения транспортных средств;
* предупреждение и пресечение преступлений и административных правонарушений в области дорожного движения.

# Глоссарий основных терминов

*ГИБДД* — самостоятельное структурное подразделение центрального аппарата Министерства внутренних дел Российской Федерации.

*ДТП* — событие, возникшее в процессе движения по дороге транспортного средства и с его участием, при котором погибли или пострадали люди или повреждены транспортные средства, сооружения, грузы, либо причинён иной материальный ущерб.

*ДПС* — структурное подразделение Министерства внутренних дел России, а также министерств внутренних дел стран СНГ.

*КоАП РФ* — кодифицированный нормативный акт, регулирующий общественные отношения по привлечению к административной ответственности.

*ПТС* — документ, содержащий сведения об основных технических характеристиках транспортного средства, идентификационные данные основных агрегатов, сведения о собственнике, постановке на учёт и снятии с учёта.

*ПДД* — свод правил, регулирующих обязанности участников дорожного движения, а также технические требования, предъявляемые к транспортным средствам, для обеспечения безопасности дорожного движения.

*Постановление* — один из видов подзаконных нормативно-правовых актов, принимаемых органом исполнительной власти.

*Свидетельство о регистрации транспортного средства* — официальный документ, подтверждающий постановку автомобиля на регистрационный учет и принадлежность автомобиля конкретному владельцу.

# Анализ предметной области

Анализ предметной области позволяет выделить ее сущности, определить первоначальные требования к функциональности и определить границы проекта. Проведение анализа предметной области в интересах последующего проектирования базы данных является задачей, формирующей единый взгляд на сведения, которые в предметной области обрабатываются, учитывая не только их структуры, но и правила хранения и обработки, что отражается в выделяемых функциях и задачах.

## Структура подразделений ГИБДД и их подчиненность

В составе ГИБДД дорожно-патрульная служба формируется преимущественно в виде строевых подразделений. Термин «строевые» означает, что их структура, внутренняя организация и система подчиненности близки к воинским формированиям.

В зависимости от количества зарегистрированных ТС, протяженности автомобильных дорог, уровня ДТП, а также наличия необходимых ресурсов в пределах административно-территориальных единиц могут комплектоваться следующие подразделения ДПС:

* группа;
* отдельный взвод;
* моторизованный взвод (группа);
* отдельная рота;
* отдельный батальон;
* полк.

В соответствие с типовыми штатами строевых подразделений ДПС ГИБДД делится на группы, отдельные и моторизованные взводы численностью до 50 человек обычно создаются при районных и городских ОВД. Более крупные подразделения – отдельные роты, батальоны и полки ДПС – формируются в республиках, краях, областях и крупных городах. Как правило, они подчиняются по вертикали – вышестоящим органам ГИБДД (отделам или управлениям ГИБДД УВД МВД субъектов Российской Федерации) и по горизонтали – территориальным органам внутренних дел (УВД ГОРОВД, Рисунок 1).



Рисунок 1 – Схема подчиненности ДПС

## Бизнес-модель ГИБДД

Разработка моделей в стандарте IDEF0 позволяет наглядно и эффективно отобразить весь механизм деятельности предприятия в нужном разрезе.

Описание системы с помощью IDEF0 называется функциональной моделью. Функциональная модель предназначена для описания существующих бизнес-процессов, в котором используются как естественный, так и графический языки.

Методология IDEF0 предписывает построение иерархической системы диаграмм – единичных описаний фрагментов системы. Сначала проводится описание системы в целом и ее взаимодействия с окружающим миром (контекстная диаграмма), после чего проводится функциональная декомпозиция – система разбивается на подсистемы и каждая подсистема описывается отдельно (диаграммы декомпозиции). Затем каждая подсистема разбивается на более мелкие и так далее до достижения нужной степени подробности.

Разработанная бизнес-модель демонстрируется на диаграммах:

* контекстная диаграмма (Рисунок 2);
* диаграмма подсистем (Рисунок 3).

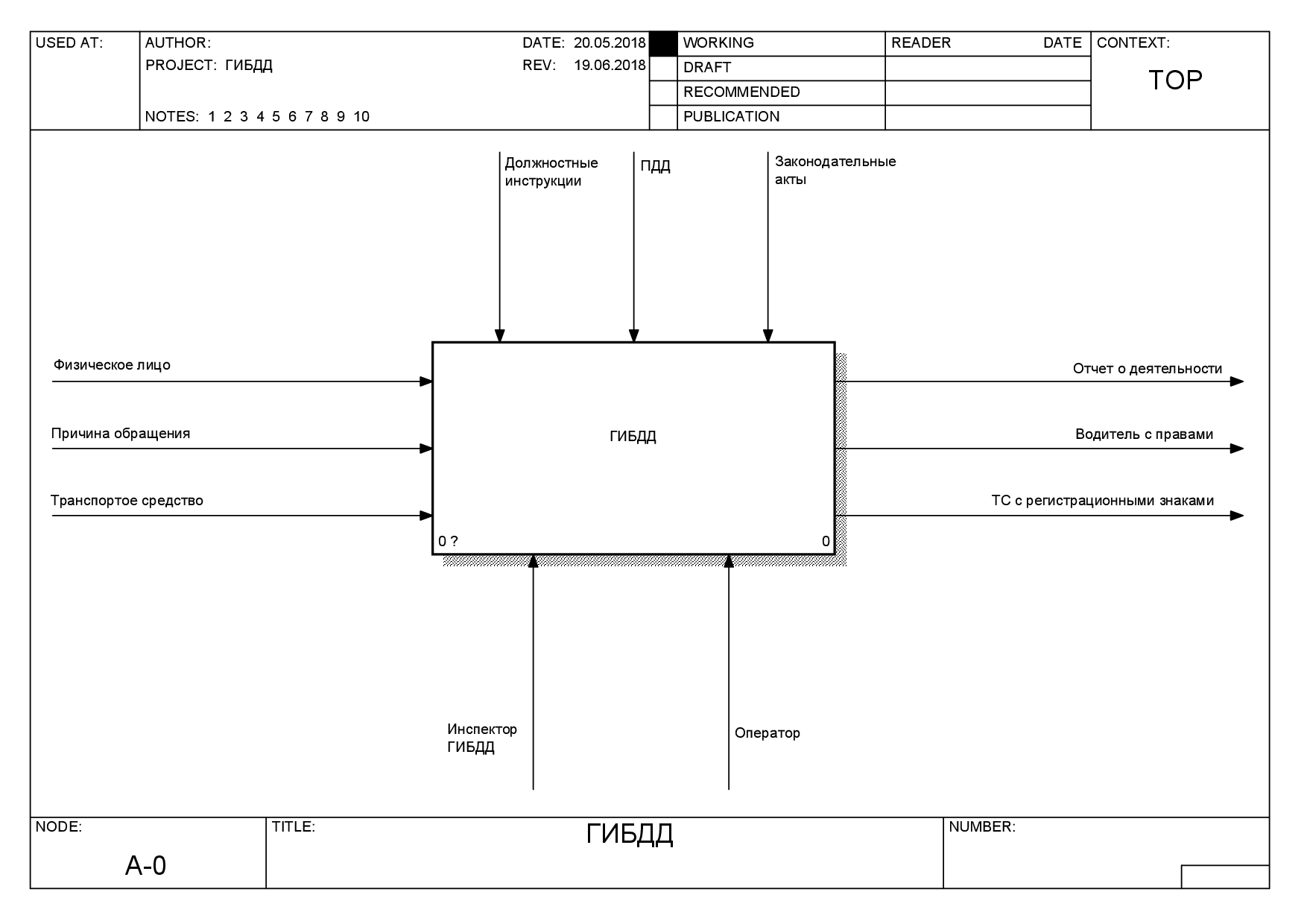


Рисунок 2 – Контекстная диаграмма процессов управления дорожным движением

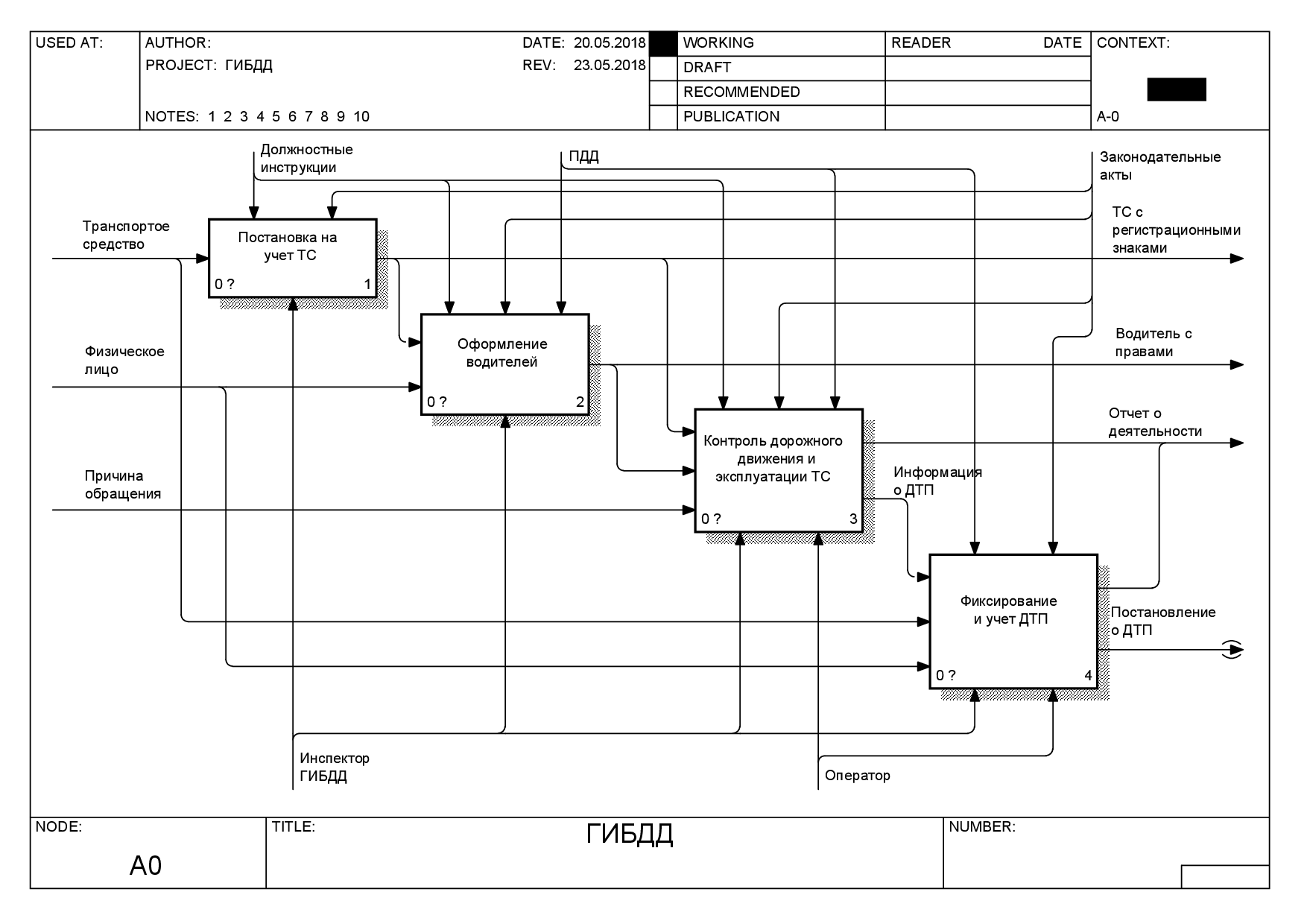


Рисунок 3 – Диаграмма подсистем управления дорожным движением

Модели были спроектированы при помощи средств визуального моделирования бизнес-процессов BPwin.

## Потоки данных ГИБДД

Диаграммы потоков данных (DFD) представляют собой иерархию функциональных процессов, связанных потоками данных. Цель такого представления – продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между этими процессами.

Основными компонентами, которыми проводится описание диаграмм потоков данных, являются: внешние сущности, системы и подсистемы, процессы, накопители данных, потоки данных.

Контекстная диаграмма (Рисунок 4), и диаграмма подсистем ПС (Рисунок 5) помогают проектировщику лучше выбрать процесс для автоматизации.

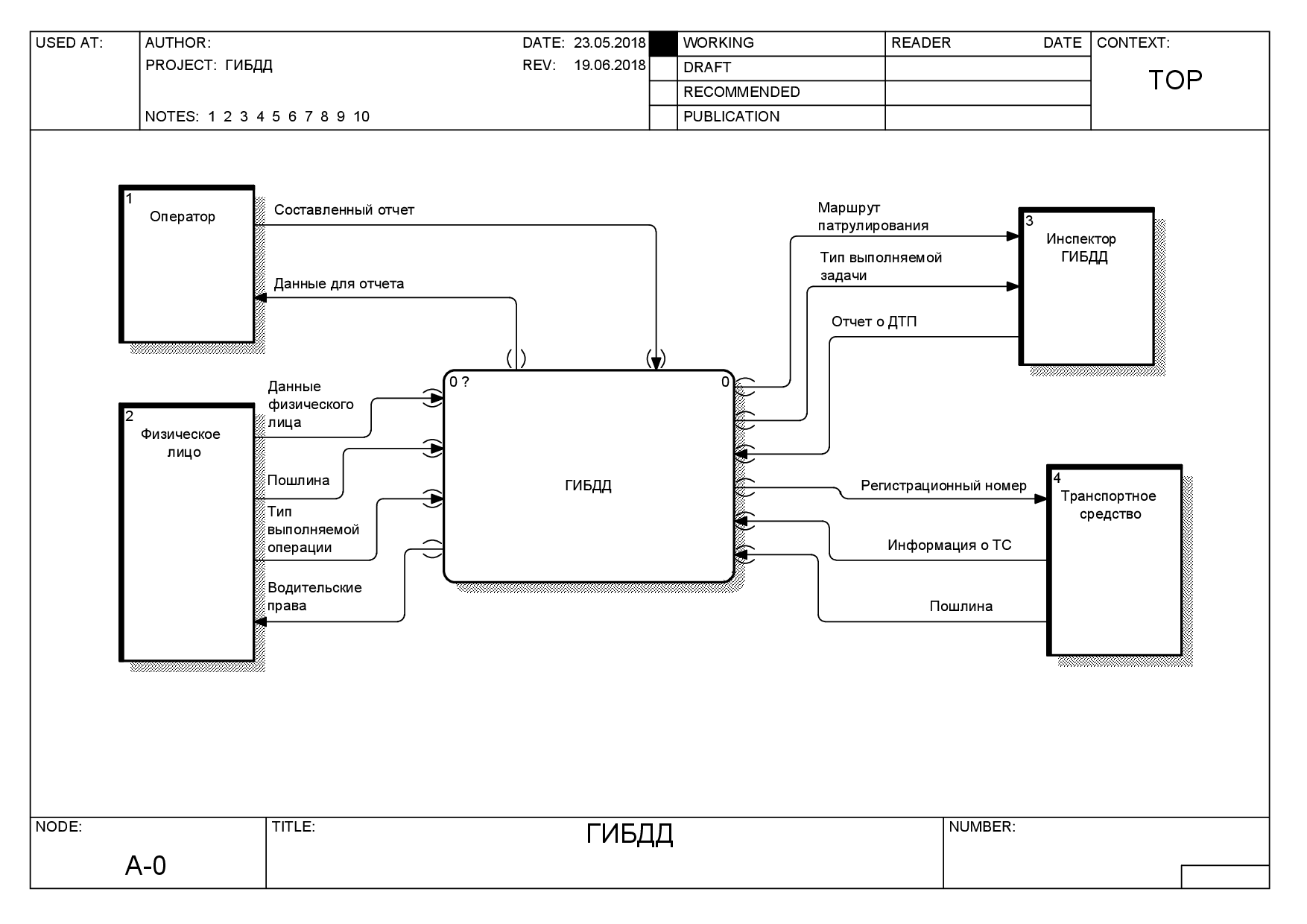


Рисунок 4 – Контекстная диаграмма потоков управления дорожным движением

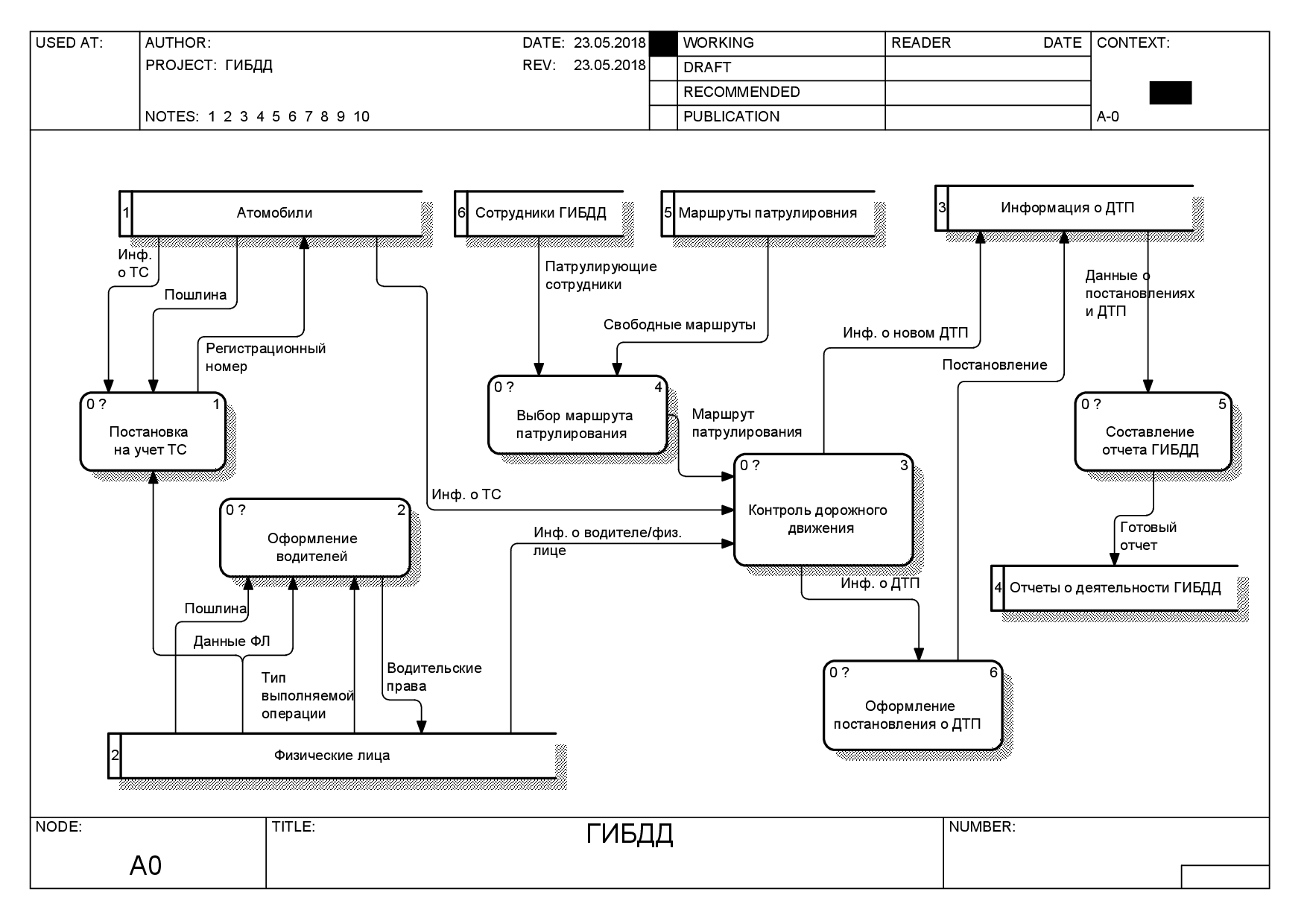


Рисунок 5 – Диаграмма потоков информации управления дорожным движением

Путем анализа DFD диаграммы было выделено несколько хранилищ, которые представлены в существующей модели в виде бумажных документов:

* автомобили;
* физические лица;
* сотрудники ГИБДД;
* маршруты патрулирования;
* информация о ДТП;
* отчеты о деятельности ГИБДД.

# Постановка задачи

В результате проведенного анализа было выявлен неоптимальный бизнес-процесс на уровне модели «Фиксирование и учет ДТП». Формирование постановления о ДТП происходит вручную, что занимает много времени. Данный процесс поддается автоматизации.

Целью разработки программного средства выбрана автоматизация оформления постановления о ДТП.

Необходимо хранить все сведения о ДТП (место и время происшествия, пункты нарушения), сведения об участниках (ФИО, паспортные данные, год и место рождения, статус в ДТП), сведения об участвующих ТС (ПТС, VIN, цвет, марка, тип, гос. номер, владелец), также следует хранить информацию о составителе документа (ФИО, должность, звание).

Все эти сведения будут фигурировать в итоговом документе – постановлении о ДТП.

# Обоснование решаемой задачи

На основании данных протоколирования инспектора заполняют постановление об административном правонарушении, в которых отражается информация о:

* дата и место составления документа;
* должность, фамилия и инициалы инспектора ГИБДД;
* данные о владельце транспортного средства;
* данные о свидетелях и потерпевших, в частности их ФИО и адреса;
* показания свидетелей;
* указание нормативно-правового акта, который был нарушен, место и время совершения правонарушения, а также подробное описание ситуации и роли в ней участников нарушения;
* указание статьи, которая предусматривает ответственность за правонарушение;
* иные сведения, которые могут пригодиться для разрешения дела;
* форма административного наказания.

При этом следует принять во внимание, что участниками могут быть как автомобили, так и пешеходы и оба могут фигурировать в постановлении о ДТП в качестве виновников, участников или потерпевших. Вместе с гос. номеров автомобиля требуется хранить информацию о водителе, находящемся за рулем во время ДТП и информацию о владельце автотранспортного средства. Информация о водителе должна опирать на номер паспорта, т. к. у водителя на момент нарушения может не быть водительских прав.

Каждое нарушение ПДД может быть классифицировано по нескольким пунктам КоАП РФ и иметь разные поправочные коэффициента, в зависимости от тяжести правонарушения.

Составление данной отчётности хотя бы за месяц требует пересмотра большого количества документации, что занимает немало времени. Следить за состоянием постановлений по картотеке крайне трудоёмко: данная работа требует больших временных затрат и имеется вероятность совершения большого количества ошибок. Автоматизация данного процесса позволит не только увеличить точность и скорость получения информации, но и как следствие этого позволит инспекторам проверить больше водителей на дороге, а значит увеличить вероятность предотвращения несчастных случаев на дорогах с участием водителей и пешеходов.

# Назначение базы данных

Перед тем, как перейти к реализации БД, требуется определиться с количеством и типом хранимой информации. Для этого нужно выделить входную и выходную информацию.

## Описание выходной информации

Результатом разработки программного средства должно быть постановление о ДТП. Постановление должно представлять собой документ, отражающий следующую информацию:

* дата и место составления документа;
* должность, фамилия и инициалы инспектора ГИБДД;
* данные о владельце транспортного средства;
* данные о свидетелях и потерпевших, в частности их ФИО и адреса;
* показания свидетелей;
* указание нормативно-правового акта, который был нарушен, место и время совершения правонарушения, а также подробное описание ситуации и роли в ней участников нарушения.
* указание статьи, которая предусматривает ответственность за правонарушение;
* иные сведения, которые могут пригодиться для разрешения дела;
* форма административного наказания.

## Описание входной информации

Для осуществления программным средством возложенной на нее задачи, необходимо ввести следующие данные:

Информация о физических лицах, фигурирующих в участниках ДТП и владельцах авто (ФИО, № паспорта, Адрес проживания, Дата рождения).

Информация об актуальных водительских правах (№ паспорта водителя, № водительских прав).

Информация об автомобилях (ПТС, Модель, Цвет, VIN, № кузова, № двигателя, Год выпуска). Если автомобиль имеет регистрационный знак, то требуется хранить и его.

Информацию о сотрудниках ГИБДД (ФИО, Должность, Звание).

Информацию о типах возможных правонарушений (Название нарушения, Название, Размер штрафа, КоАП).

Описанной выше информации будет достаточно для составления постановления и формирования отчетов о деятельности ГИБДД.

# Проектирование базы данных

В рамках проектной деятельности требуется построить концептуальную, логическую и физическую модели объекта автоматизации.

## Концептуальное проектирование

На этом этапе создаются подробные модели пользовательских представлений данных предметной области. Затем они интегрируются в концептуальную модель, в которой фиксируют все элементы корпоративных данных, подлежащих загрузке в базу данных. Эту модель называют концептуальной схемой базы данных.

Средством моделирования предметной области на этапе концептуального проектирования является модель «сущность-связь». Часто ее называют ER-моделью. В ней моделирование структуры данных предметной области базируется на использовании графических средств – ER-диаграмм. В наглядном виде они представляют связи между сущностями.

Основными понятиями ER-диаграммы являются сущность, атрибут, связь.

Сущность представляет собой объект, информация о котором хранится в базе данных. Сущность имеет экземпляры, отличающиеся друг от друга значениями атрибутов и допускающие однозначную идентификацию.

Атрибут – это свойство сущности. Атрибут, который уникальным образом идентифицирует экземпляры сущности, называется ключом. Может быть составной ключ, представляющий комбинацию нескольких атрибутов.

Связь представляет взаимодействие между сущностями. Она характеризуется мощностью, которая показывает, сколько сущностей участвует в связи.

Создадим ER-модель предметной области ГИБДД (Рисунок 6). Каждая из сущностей приведенной ER-модели может быть описана своим набором атрибутов. Первичные ключи для сущностей выделим оранжевой заливкой, а внешние – серой.



Рисунок 6 – ER-диаграмма предметной области

ER-модель в совокупности с наборами атрибутов сущностей может служить примером концептуальной модели предметной области или концептуальной схемы базы данных.

Рассмотрим проектирование базы данных ГИБДД. В ней могут быть определены следующие сущности:

* цвет – перечисление всех возможных цветов автомобилей;
* модель – информация о модели и категории автомобиля;
* машина – описание конкретного автомобиля;
* физическое лицо – информацию о конкретном человеке;
* водитель – граждане с водительским удостоверением;
* свидетельство о регистрации ТС – ассоциирует авт. с гос. номером;
* сотрудник – все сотрудники ГИБДД;
* нарушение – возможные типы нарушений ПДД;
* тип участника – определение статуса участника при ДТП;
* дтп – информация о постановлении при ДТП;
* участники автомобили – ассоциирует автомобиль с постановлением;
* участники пешеходы – ассоциирует физ. лицо с постановлением;
* штрафы – ассоциирует нарушение с постановлением.

Набор атрибутов для каждой сущности проще всего представить в табличном виде (Таблица 1). Название таблиц выделены нижн. подчеркиванием, **первичные ключи** – жирным, *внешние ключи* – курсивом.

Таблица 1 – Наборы атрибутов сущностей предметной области

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Цвет | | **№ цвета** | | Название | | |  | | --- | | Модель | | **№ модели** | | Категория | | Наименование | | |  | | --- | | Машина | | **ПТС** | | *№ модели* | | *№ цвета* | | VIN | | № кузова | | № двигателя | | Год выпуска | | |  | | --- | | Физическое лицо | | **Паспорт** | | ФИО | | Адр. проживания | | Дата рождения | |
| |  | | --- | | Водитель | | *Паспорт* | | Водительское удостоверение | | | |  | | --- | | Свидетельство о регистрации ТС | | **Госномер** | | *Паспорт* | | *ПТС* | | |
| |  | | --- | | Нарушение | | **№ нарушения** | | Название | | Размер штрафа | | КоАП | | |  | | --- | | Тип участника | | **№ типа** | | Статус | | |  | | --- | | Штрафы | | *№нарушения* | | *Постановление* | | Надбавка | | |  | | --- | | Сотрудник | | **№ сотрудника** | | ФИО | | Должность | | Звание | |
| |  | | --- | | ДТП | | **Постановление** | | *№ сотрудника* | | Место происшествия | | Описание | | Дата | | | |  | | --- | | Участники автомобили | | *Паспорт* | | *Постановление* | | *Госномер* | | *№ типа* | | |
| |  | | --- | | Участники пешеходы | | *Паспорт* | | *Постановление* | | *№ типа* | | |  | |

Для уточнения связей между объектами ER-диаграммы, построим таблицу связей между сущностями (Таблица 2).

Таблица 2 – Связи между ассоциативными сущностями в ER-диаграмме

|  |  |  |
| --- | --- | --- |
| **Объект** | **Связь** | **Объект** |
| Свид. о рег. ТС | один к одному | Машина |
| один ко многим | Физическое лицо |
| Участники автомобили | один ко многим | Тип участника |
| один ко многим | Физическое лицо |
| один ко многим | ДТП |
| один ко многим | Свид. о рег. ТС |
| Участники пешеходы | один ко многим | Тип участника |
| один ко многим | Физическое лицо |
| один ко многим | ДТП |
| Штрафы | один ко многим | Нарушение |
| один ко многим | ДТП |

Данная модель на языке инфологического моделирования будет иметь следующий вид:

Модель (№ модели, Категория, Наименование) {Машина}

Цвета (№ цвета, Название) {Машина}

Машина (VIN, ПТС, № кузов, № цвета, № двигателя, Год выпуска, № модели)

Физическое лицо (ФИО, Паспорт, Адрес проживания, Дата рождения)

Водитель (Водительское удостоверение, Паспорт) {Физическое лицо}

Свидетельство о регистрации ТС [Физическое лицо 1, Машина \*] (Паспорт, Госномер, Дата регистрации, ПТС)

Сотрудник (№ сотрудника, ФИО, Должность, Звание) {ДТП}

ДТП (Дата, № сотрудника, Место происшествия, Постановление, Описание)

Нарушение (№ нарушения, Название, КоАП, Размер штрафа) {Штрафы}

Штрафы [Нарушение \*, ДТП \*] (№ нарушения, Постановление, Надбавка)

Тип участника (№ типа, Статус) {Участники пешеходы, Участники автомобили}

Участники пешеходы [Физическое лицо \*, ДТП \*, Тип участника \*] (Паспорт, Постановление, № типа)

Участники автомобили [Свидетельство о регистрации ТС \*, Физическое лицо \*, ДТП \*, Тип участника \*] (Госномер, Паспорт, Постановление, № типа)

На основе созданной инфологической модели, представленной в форме ER-диаграммы и её описания на языке инфологического моделирования, создаётся логическая модель проектируемой базы данных.

## Логическое проектирование

Концептуальные модели позволяют более точно представить предметную область, чем реляционные и другие более ранние модели. Но в настоящее время существует немного систем управления базами данных, поддерживающих эти модели. На практике наиболее распространены системы, реализующие реляционную модель (Рисунок 7). Поэтому необходим метод перевода концептуальной модели в реляционную. Такой метод основывается на формировании набора предварительных таблиц из ER-диаграмм.

Для каждой сущности создается таблица. Причем каждому атрибуту сущности соответствует столбец таблицы.

Правила генерации таблиц из ER-диаграмм опираются на два основных фактора – тип связи и класс принадлежности сущности.

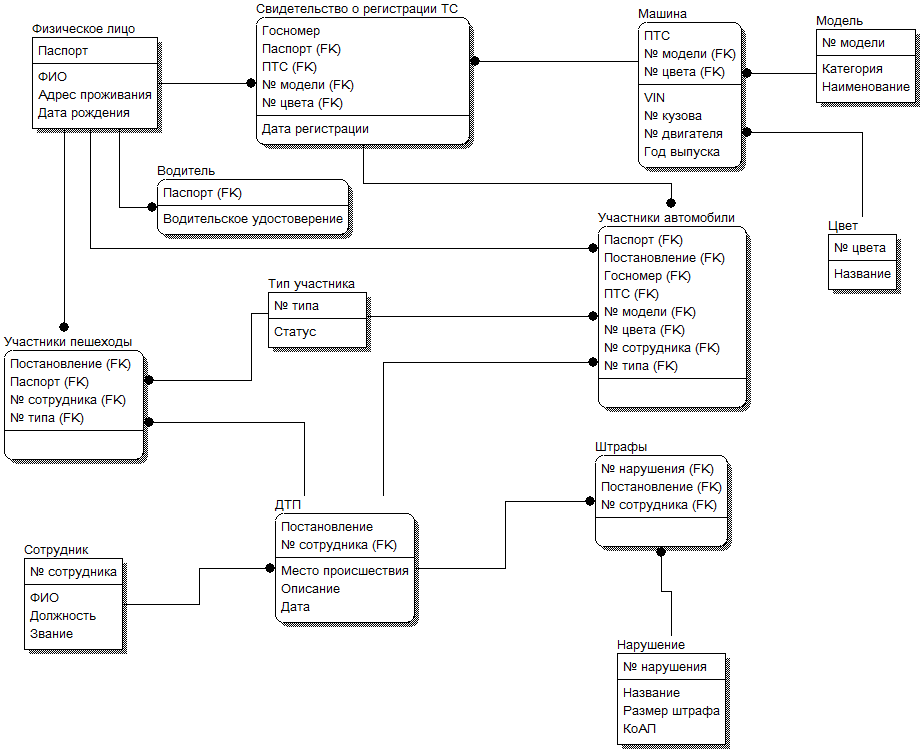


Рисунок 7 – Логическая модель предметной области

Т. к. в ER-диаграмме отношения многие ко многим уже были развернуты в отдельные сущности – ассоциации, то логическая модель была построена на основе ER-диаграммы практически без изменений.

## Физическое проектирование

Цель этапа физического проектирования – описание конкретной реализации базы данных, размещаемой во внешней памяти компьютера. Это описание структуры хранения данных и эффективных методов доступа к данным базы. При логическом проектировании отвечают на вопрос – что надо сделать, а при физическом – выбирается способ, как это сделать.

Для проектирования таблиц базы данных будем использовать компактную встраиваемую СУБД SQLite (Рисунок 8).

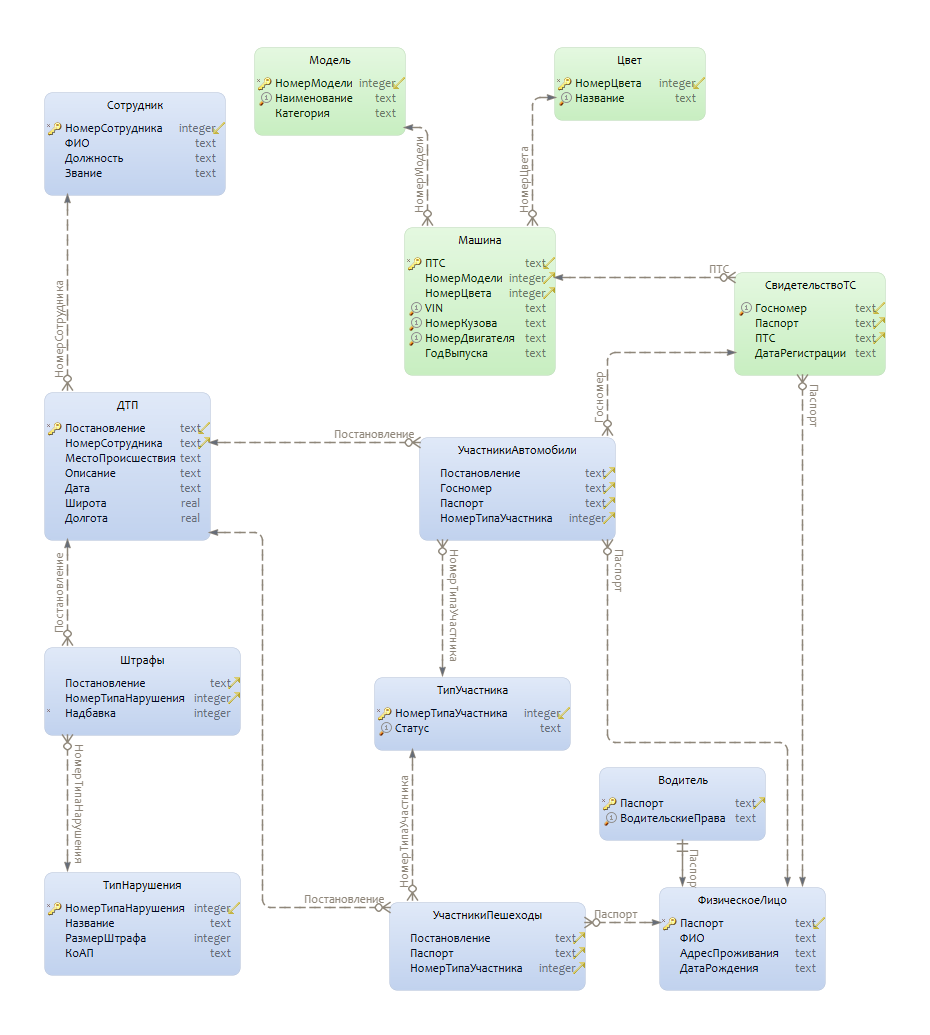


Рисунок 8 – Физическая модель предметной области в SQLite

*Код генерации DDL:*

--

-- Использованная кодировка текста: UTF-8

--

PRAGMA foreign\_keys = off;

BEGIN TRANSACTION;

-- Таблица: Водитель

CREATE TABLE Водитель (Паспорт VARCHAR PRIMARY KEY REFERENCES ФизическоеЛицо (Паспорт) ON DELETE CASCADE ON UPDATE CASCADE NOT NULL, ВодительскиеПрава VARCHAR UNIQUE);

-- Таблица: ДТП

CREATE TABLE ДТП (Постановление VARCHAR PRIMARY KEY NOT NULL, НомерСотрудника VARCHAR REFERENCES Сотрудник (НомерСотрудника) ON DELETE CASCADE ON UPDATE CASCADE, МестоПроисшествия VARCHAR, Описание VARCHAR, Дата VARCHAR DEFAULT (strftime('%d.%m.%Y', 'now')), Широта REAL, Долгота REAL);

-- Таблица: Машина

CREATE TABLE Машина (ПТС VARCHAR PRIMARY KEY NOT NULL, НомерМодели INTEGER REFERENCES Модель (НомерМодели) ON DELETE CASCADE ON UPDATE CASCADE, НомерЦвета INTEGER REFERENCES Цвет (НомерЦвета) ON DELETE CASCADE ON UPDATE CASCADE, VIN VARCHAR UNIQUE, НомерКузова VARCHAR UNIQUE, НомерДвигателя VARCHAR UNIQUE, ГодВыпуска VARCHAR);

-- Таблица: Модель

CREATE TABLE Модель (НомерМодели INTEGER NOT NULL PRIMARY KEY, Наименование VARCHAR UNIQUE, Категория VARCHAR);

-- Таблица: СвидетельствоТС

CREATE TABLE СвидетельствоТС (Госномер VARCHAR PRIMARY KEY, Паспорт VARCHAR REFERENCES ФизическоеЛицо (Паспорт) ON DELETE CASCADE ON UPDATE CASCADE, ПТС VARCHAR REFERENCES Машина (ПТС) ON DELETE CASCADE ON UPDATE CASCADE, ДатаРегистрации VARCHAR);

-- Таблица: Сотрудник

CREATE TABLE Сотрудник (НомерСотрудника INTEGER PRIMARY KEY NOT NULL, ФИО VARCHAR, Должность VARCHAR, Звание VARCHAR);

-- Таблица: ТипНарушения

CREATE TABLE ТипНарушения (НомерТипаНарушения INTEGER NOT NULL PRIMARY KEY, Название VARCHAR, РазмерШтрафа INTEGER, КоАП VARCHAR);

-- Таблица: ТипУчастника

CREATE TABLE ТипУчастника (НомерТипаУчастника INTEGER PRIMARY KEY NOT NULL, Статус VARCHAR UNIQUE);

-- Таблица: УчастникиАвтомобили

CREATE TABLE УчастникиАвтомобили (Постановление VARCHAR REFERENCES ДТП (Постановление) ON DELETE CASCADE ON UPDATE CASCADE, Госномер VARCHAR REFERENCES СвидетельствоТС (Госномер) ON DELETE CASCADE ON UPDATE CASCADE, Паспорт VARCHAR REFERENCES ФизическоеЛицо (Паспорт) ON DELETE CASCADE ON UPDATE CASCADE, НомерТипаУчастника INTEGER REFERENCES ТипУчастника (НомерТипаУчастника) ON DELETE CASCADE ON UPDATE CASCADE);

-- Таблица: УчастникиПешеходы

CREATE TABLE УчастникиПешеходы (Постановление VARCHAR REFERENCES ДТП (Постановление) ON DELETE CASCADE ON UPDATE CASCADE, Паспорт VARCHAR REFERENCES ФизическоеЛицо (Паспорт) ON DELETE CASCADE ON UPDATE CASCADE, НомерТипаУчастника INTEGER REFERENCES ТипУчастника (НомерТипаУчастника) ON DELETE CASCADE ON UPDATE CASCADE);

-- Таблица: ФизическоеЛицо

CREATE TABLE ФизическоеЛицо (Паспорт VARCHAR PRIMARY KEY NOT NULL, ФИО VARCHAR, АдресПроживания VARCHAR, ДатаРождения VARCHAR);

-- Таблица: Цвет

CREATE TABLE Цвет (НомерЦвета INTEGER NOT NULL PRIMARY KEY, Название VARCHAR UNIQUE);

-- Таблица: Штрафы

CREATE TABLE Штрафы (Постановление VARCHAR REFERENCES ДТП (Постановление) ON DELETE CASCADE ON UPDATE CASCADE, НомерТипаНарушения INTEGER REFERENCES ТипНарушения (НомерТипаНарушения) ON DELETE CASCADE ON UPDATE CASCADE, Надбавка INTEGER NOT NULL DEFAULT (0));

COMMIT TRANSACTION;

PRAGMA foreign\_keys = on;

# Клиентское приложение

Клиентское приложение для БД ГИБДД реализовано на Lazarus версии 1.8.4 с использованием сервисов Google Maps и дополнительных компонентов: fpCEF (для отображения карт) и LazReport (для визуализации формы постановления). Сама БД была реализована в СУБД SQLite при помощи SQLiteStudio версии 3.1.1. Для решения задач управления постановлениями о ДТП, создано несколько форм с удобным интерфейсом, позволяющие решать типовые задачи. Он состоит из таблиц, для вывода данных из БД, элементов управления данными и навигации по таблице. Каждая форма имеет собственный функциональный набор.

Навигация по существующим формам осуществляется через меню основного окна программы (Рисунок 9).

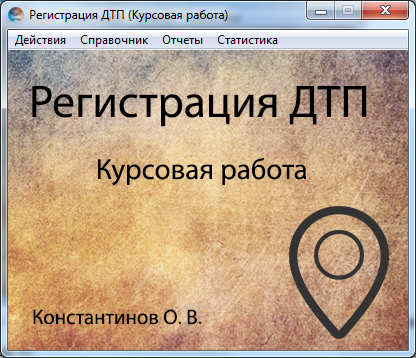


Рисунок 9 – Главное окно программы

В качестве выходной информации выступает постановление о ДТП (Рисунок 10), данные в котором формируются в СУБД при помощи представления «ОформленныеПостановления», а отображаются в форме клиентского приложения при помощи компонента LazReport.

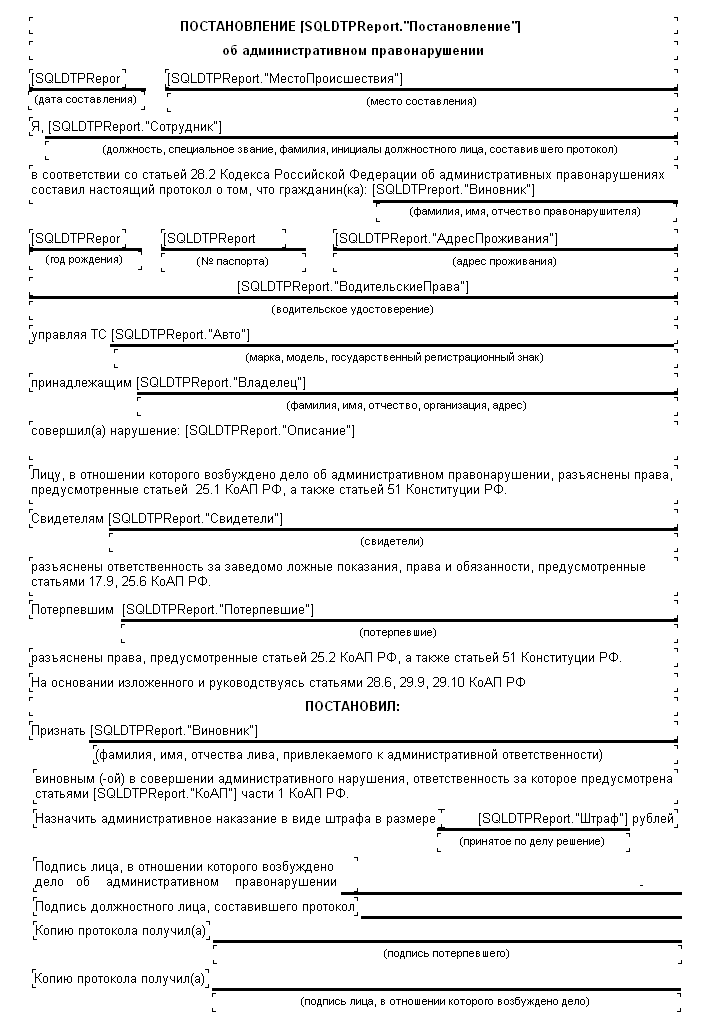


Рисунок 10 – Форма постановления о ДТП

*Код представления «АвтоПоГосНомеру»:*

SELECT стс.Госномер,

мд.Наименование || ', цвет: ' || ц.Название || ', госномер: ' || стс.Госномер || ', год выпуска: ' || м.ГодВыпуска Авто,

фл.ФИО || ', дата рождения: ' || фл.ДатаРождения || ', паспорт: ' || фл.Паспорт Владелец

FROM СвидетельствоТС стс

INNER JOIN

Машина м ON м.ПТС = стс.ПТС

INNER JOIN

Модель мд ON мд.НомерМодели = м.НомерМодели

INNER JOIN

Цвет ц ON ц.НомерЦвета = м.НомерЦвета

INNER JOIN

ФизическоеЛицо фл ON фл.Паспорт = стс.Паспорт

*Код представления «УчастникиПоПостановлению»:*

SELECT уч.Постановление,

фл.ФИО,

уч.Авто,

уч.Владелец,

фл.Паспорт,

фл.ДатаРождения,

фл.АдресПроживания,

в.ВодительскиеПрава,

ту.Статус

FROM (

SELECT Постановление,

Паспорт,

Авто,

Владелец,

НомерТипаУчастника

FROM УчастникиАвтомобили уа

INNER JOIN

АвтоПоГосНомеру оапгн ON уа.Госномер = оапгн.Госномер

UNION ALL

SELECT Постановление,

Паспорт,

NULL,

NULL,

НомерТипаУчастника

FROM УчастникиПешеходы

)

уч

INNER JOIN

ФизическоеЛицо фл ON фл.Паспорт = уч.Паспорт

INNER JOIN

ТипУчастника ту ON ту.НомерТипаУчастника = уч.НомерТипаУчастника

LEFT JOIN

Водитель в ON в.Паспорт = фл.Паспорт

*Код представления «ШтрафПоПостановлению»:*

SELECT д.Постановление,

SUM(тн.РазмерШтрафа + ш.Надбавка) Штраф,

GROUP\_CONCAT(тн.КоАП, ', ') КоАП

FROM ДТП д

LEFT JOIN

Штрафы ш ON ш.Постановление = д.Постановление

LEFT JOIN

ТипНарушения тн ON тн.НомерТипаНарушения = ш.НомерТипаНарушения

GROUP BY д.Постановление

Все вышеописанные представления служат для формирования составного представления по совершенным ДТП.

*Код представления «ОформленныеПостановления»:*

SELECT с.ФИО || ', ' || с.Должность || ', ' || с.Звание Сотрудник,

д.Постановление,

д.МестоПроисшествия,

д.Описание,

д.Дата,

упп.ФИО Виновник,

упп.Паспорт,

упп.ДатаРождения,

упп.АдресПроживания,

упп.Авто,

упп.Владелец,

упп.ВодительскиеПрава,

шпп.Штраф,

шпп.КоАП,

(

SELECT GROUP\_CONCAT(ФИО || ', п.: ' || Паспорт, '; ')

FROM УчастникиПоПостановлению

WHERE Статус = 'свидетель' AND

Постановление = д.Постановление

GROUP BY Статус

LIMIT 2

)

Свидетели,

(

SELECT GROUP\_CONCAT(ФИО || ', п.: ' || Паспорт, '; ')

FROM УчастникиПоПостановлению

WHERE Статус = 'потерпевший' AND

Постановление = д.Постановление

GROUP BY Статус

LIMIT 2

)

Потерпевшие

FROM ДТП д

INNER JOIN

Сотрудник с ON с.НомерСотрудника = д.НомерСотрудника

INNER JOIN

(

SELECT \*

FROM УчастникиПоПостановлению

WHERE Статус = 'виновник'

)

упп ON упп.Постановление = д.Постановление

INNER JOIN

ШтрафПоПостановлению шпп ON шпп.Постановление = д.Постановление

Результатом исполнения представления «ОформленныеПостановления» будет список ДТП со следующими полями:

* Сотрудник.
* Постановление.
* МестоПроисшествия.
* Описание.
* Дата.
* ФИО Виновник.
* Паспорт.
* ДатаРождения.
* АдресПроживания.
* Авто.
* Владелец.
* ВодительскиеПрава.
* Штраф.
* КоАП.

# Заключение

В результате проведенных работ по курсовому проекту, была спроектировано и реализовано программное средство и база дынных. Внедрение данного программного средства позволит более оперативно составлять постановления сотрудниками ГИБДД, снизить случайные ошибки связанных с ручным заполнением форм протоколов.

При проектировании были созданы модели IDEF0, DFD, ER-диаграмма, логическая и физическая модели БД программного средства.

Для реализации курсового проекта использовались средства проектирования BPWin и ERWin в качестве среды построения моделей, IDE Lazarus для реализации оболочки программного средства, SQLiteStudio для создания БД в СУБД SQLite.

В результате проведенной работы была достигнутая поставленная цель: разработка проекта программного средства для ГИБДД.

# Список литературы

1. Астахова, И.Ф. СУБД: язык SQL в примерах и задачах : учебное пособие / И.Ф. Астахова, В.М. Мельников, А.П. Толстобров, В.В. Фертиков. — Москва : Физматлит, 2009. — 168 с
2. Гудов, А.М. Базы данных и системы управления базами данных. Программирование на языке PL/SQL : учебное пособие / А.М. Гудов, С.Ю. Завозкин, Т.С. Рейн. — Кемерово : КемГУ, 2010. — 133 с
3. Зудилова, Т.В. Создание запросов в Microsoft SQL Server 2008 : учебно-методическое пособие / Т.В. Зудилова, Г.Ю. Шмелева. — Санкт-Петербург : НИУ ИТМО, 2013. — 149 с
4. Кара-Ушанов, В.Ю. SQL — язык реляционных баз данных : учебное пособие / В.Ю. Кара-Ушанов. Екатеринбург : УрФУ, 2016. — 156 с.
5. Красюк Л.В. Методические указания по выполнению курсовой работы по дисциплине «Базы данных» / Л.В. Красюк. – Владивосток: ДВФУ, 2016. – 20 с.
6. Кудрявцев, К.Я. Создание баз данных: учебное пособие / К.Я. Кудрявцев. — Москва : НИЯУ МИФИ, 2010. — 155 с.
7. Марасанов, А.М. Распределенные базы и хранилища данных: учебное пособие / А.М. Марасанов, Н.П. Аносова, О.О. Бородин, Е.С. Гаврилов. — Москва : , 2016. — 254 с.

Приложение А

**(обязательное)**

**Руководство пользователя**

* 1. Обобщенное описание работы программы

Программа состоит из двух частей: исполняемый файл и база данных. База данных хранится в формате \*.sqlite. Работа с базой данных осуществляется посредством исполняемого файла. При запуске программы (Рисунок 11) идет создание пунктов справочника и демонстрация основного меню программы.

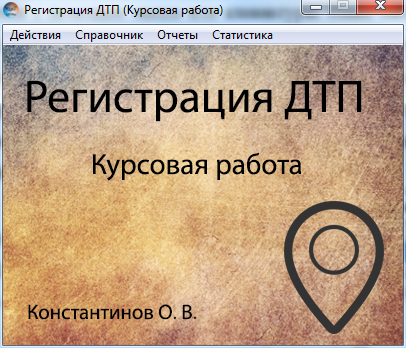


Рисунок 11 – Основное окно программы

При нажатии на интересующий пункт, открывается окно с таблицей, где каждая строка является записью из БД. Навигация в окне справочника осуществляется с помощью мыши или клавиатуры. Ниже таблицы расположены пункты для добавления, удаления, редактирования, поиска и сортировки данных таблицы. При нажатии на кнопки редактирования и вставки записи, открывается окно, в котором можно осуществить желаемые действия. При нажатии на кнопку удаления, окно не появляется, а пользователя просят повторно подтвердить желаемое действие.

После заполнения справочников полезной информацией, пользователь выбирает пункт работы с происшествиями, через который возможно добавление нового ДТП, редактирование или удаление существующих ДТП. Также через данное окно осуществляется работы с картами происшествий, привязывать к ДТП участников (виновников, свидетелей, потерпевших) и назначать штрафы за конкретные пункты нарушения ПДД.

При возникновении необходимости печати постановления пользователь переходит к окну отчетов. В данном окне пользователь может просмотреть и выбрать заполненный бланк постановления. Список постановлений формируется из списка ДТП при установленном виновнике случившегося.

* 1. Заполнение справочников

Вся информация, хранимая в БД представима в виде справочников, каждый справочник представляет собой окно, которое можно разделить на три части: просмотр, действие, представление.

При открытии каждого справочника (Рисунок 12) происходит формирование шапки таблицы и заполнение значений таблицы актуальными данными из БД.

Сама таблица не предоставляет возможности для редактирования или удаления данных, для этого есть три кнопки выбора действий, находящиеся под таблицей.

Кнопки выбора действий дают возможность вставки нового значения, изменение и удаления существующих записей.

Также, пользователь может отфильтровать и/или отсортировать список по выбранным полям через панель представления. Действия по применению/сбросу фильтра и сортировки осуществляют две последние кнопки.

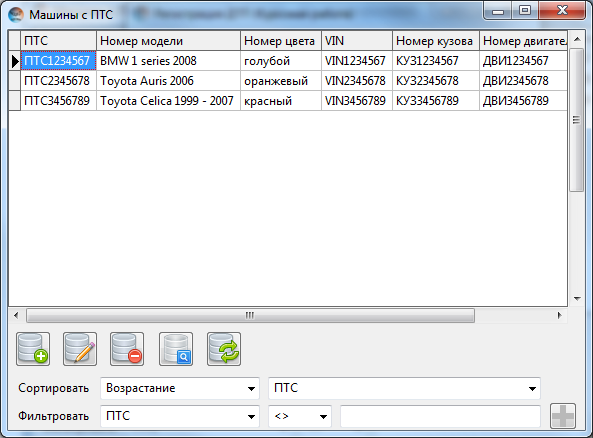


Рисунок 12 – Справочник «Машины с ПТС»

При выборе пунктов добавления или изменения записей, генерируется форма редактирования записи (Рисунок 13). Форма будет заполнена, если выбран пункт изменения записи, или пустой – в противном случае.

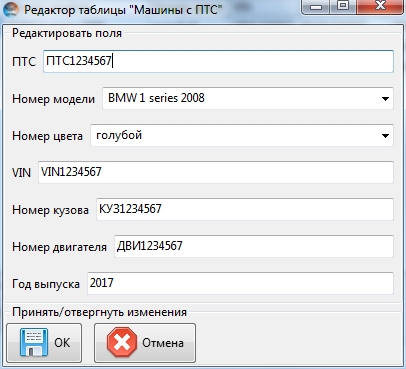


Рисунок 13 – Редактор справочника «Машины с ПТС»

Для сохранения изменений в окне редактора справочника нужно нажать на кнопку «ОК», для отмены изменений – на «Отмена».

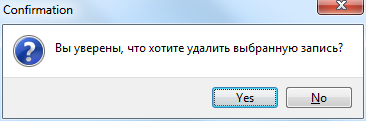


Рисунок 14 – Предупреждение об удалении записи из справочника

При выборе пункта удаления пользователю требуется подтвердить удаление записи из справочника (Рисунок 14), нажав «Yes».

Всего в программе существуют две генерируемые формы, для вывода каждого из тринадцати справочников: «Модели автомобилей», «Цвета автомобилей», «Машины ПТС», «Физические лица», «Сотрудники», «Водители», «Свидетельство о регистрации ТС», «Типы нарушений», «Типы участников», «Дорожно-транспортные происшествия», «Участвующие в ДТП пешеходы», «Участвующие в ДТП автомобили», «Штрафы за ДТП». Окна генерируются в момент работы приложения, что предоставляет возможность для динамического расширения функционала программы в соответствии с требованиями.

* 1. Регистрация ДТП

После заполнения справочников, пользователь переходит к процедуре регистрации ДТП (Рисунок 15). Окно можно условно разделить на две части: карту и информацию о ДТП. В левом верхнем углу окна расположены кнопки навигации по записям: переход к предыдущей записи, переход к следующей записи, добавление новой записи, удаление записи, обновление записей.

При удалении записи, требуется подтверждение от пользователя. Осуществление других действий не требует подтверждения. При добавлении новой записи, можно воспользоваться поиском места происшествия на карте посредством нажатия кнопки «Найти». Указание координат места происшествия является опциональной возможностью. При завершении всех действий, требуется применить изменения, нажав на кнопку «Применить».

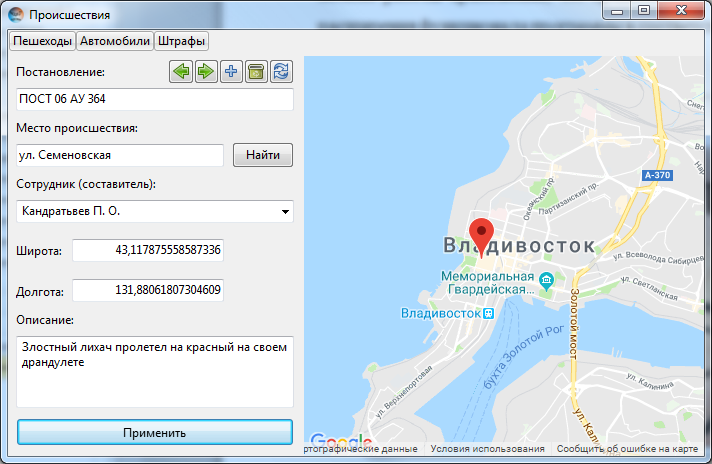


Рисунок 15 – Окно редактирования происшествий

При попытке выхода из окна редактирование происшествий без применения изменений происходит предупреждение пользователя о необходимости сохранения результатов работы (Рисунок 16), при выборе пункта «Yes», изменения будут отвергнуты.

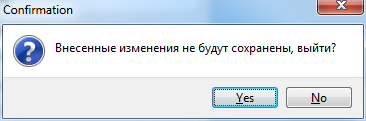


Рисунок 16 – Предупреждение об потери внесенных изменений

* 1. Печать бланков постановлений

После занесения информации о ДТП, пользователь может просмотреть и распечатать бланк постановления о ДТП посредством выбора меню «Отчеты». Окно формы разделено на две части: предпросмотр текущего выбранного бланка постановления и списка всех зарегистрированных ДТП (с участниками).

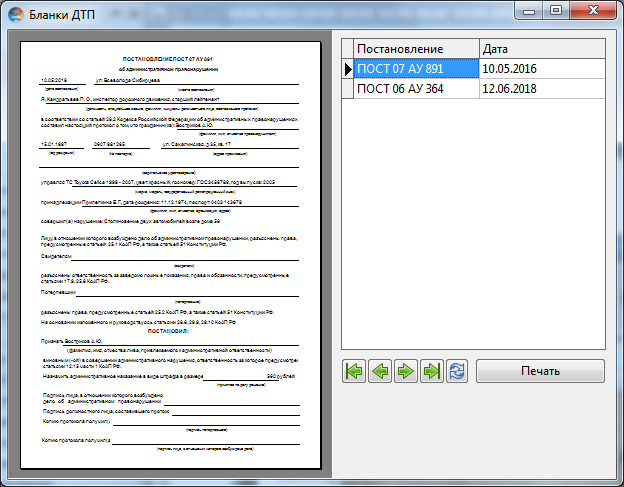


Рисунок 17 – Окно просмотра постановлений

Выбрав бланк для печати, есть возможность его распечатать посредством нажатия кнопки «Печать», также присутствует возможность выбора принтера печати и число копий (Рисунок 18).

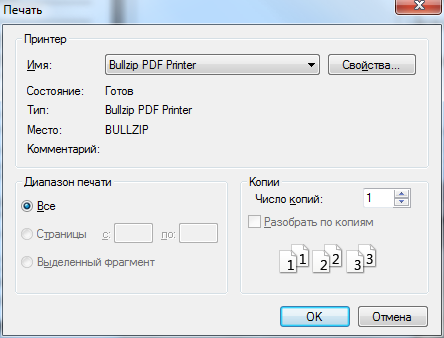


Рисунок 18 – Окно выбора принтера для печати

* 1. Статистика по сотрудникам и виновникам

В процессе работы с программой бывает необходимость узнать статистику по сотрудникам и участникам ДТП (Рисунок 19). Такая статистика может пригодится при указании штрафов ДТП за очередное правонарушение.

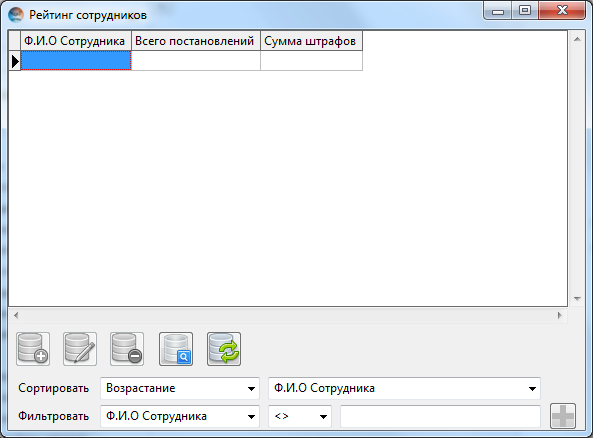


Рисунок 19 – Окно просмотра статистики работы

В отличии от формы справочника, здесь нет возможности изменения или удаления значений, но есть возможность сортировки и фильтрации.

Приложение Б

**(справочное)**

**Код реализации программного средства**

1. 1. Модуль метаданных таблиц

Данный модуль предназначен для хранения информации о таблицах и их полей в виде глобального объекта Mdata класса DBMetadata. Класс DBMetadata состоит из динамического массива записей TTable и методов манипулирования с данными. Инициализация этого класса происходит внутри модуля при его инициализации.

*Код Metadata.pas:*

unit Metadata;

{$mode objfpc}{$H+}

interface

type

TColumns = record

DataField: string;

CaptionField: string;

ListField: string;

KeyTable: string;

KeyField: string;

Primary: boolean;

end;

TTable = record

NameTable: string;

CaptionTable: string;

ReadOnly: boolean;

Columns: array of TColumns;

end;

DBMetadata = class

public

Tables: array of TTable;

private

procedure AddTable(aNameTable, aCaptionTable: string);

procedure AddStatics(aNameTable, aCaptionTable: string);

procedure AddColumn(aDataField, aCaptionField: string);

procedure AddKey(aDataField, aCaptionField, aListField, aKeyTable,

aKeyField: string);

end;

var

Mdata: DBMetadata;

implementation

procedure DBMetadata.AddTable(aNameTable, aCaptionTable: string);

begin

SetLength(Tables, Length(Tables) + 1);

with Tables[High(Tables)] do

begin

NameTable := aNameTable;

CaptionTable := aCaptionTable;

end;

end;

procedure DBMetadata.AddStatics(aNameTable, aCaptionTable: string);

begin

SetLength(Tables, Length(Tables) + 1);

with Tables[High(Tables)] do

begin

NameTable := aNameTable;

CaptionTable := aCaptionTable;

ReadOnly := True;

end;

end;

procedure DBMetadata.AddColumn(aDataField, aCaptionField: string);

begin

with Tables[High(Tables)] do

begin

SetLength(Columns, Length(Columns) + 1);

with Columns[High(Columns)] do

begin

DataField := aDataField;

CaptionField := aCaptionField;

Primary := False;

end;

end;

end;

procedure DBMetadata.AddKey(aDataField, aCaptionField, aListField,

aKeyTable, aKeyField: string);

begin

with Tables[High(Tables)] do

begin

SetLength(Columns, Length(Columns) + 1);

with Columns[High(Columns)] do

begin

DataField := aDataField;

CaptionField := aCaptionField;

ListField := aListField;

KeyTable := aKeyTable;

KeyField := aKeyField;

Primary := True;

end;

end;

end;

initialization

Mdata := DBMetadata.Create;

with Mdata do

begin

AddTable('Модель', 'Модели автомобилей');

AddColumn('Категория', 'Категория');

AddColumn('Наименование', 'Наименование');

AddTable('Цвет', 'Цвета автомобилей');

AddColumn('Название', 'Название');

AddTable('Машина', 'Машины с ПТС');

AddColumn('ПТС', 'ПТС');

AddKey('НомерМодели', 'Номер модели',

'Наименование', 'Модель', 'НомерМодели');

AddKey('НомерЦвета', 'Номер цвета', 'Название',

'Цвет', 'НомерЦвета');

AddColumn('VIN', 'VIN');

AddColumn('НомерКузова', 'Номер кузова');

AddColumn('НомерДвигателя', 'Номер двигателя');

AddColumn('ГодВыпуска', 'Год выпуска');

AddTable('ФизическоеЛицо', 'Физические лица');

AddColumn('Паспорт', 'Паспорт');

AddColumn('ФИО', 'Ф.И.О.');

AddColumn('АдресПроживания', 'Адрес проживания');

AddColumn('ДатаРождения', 'Дата рожения');

AddTable('Сотрудник', 'Сотрудники');

AddColumn('ФИО', 'Ф.И.О.');

AddColumn('Должность', 'Должность');

AddColumn('Звание', 'Звание');

AddTable('Водитель', 'Водители');

AddKey('Паспорт', 'Ф.И.О', 'ФИО', 'ФизическоеЛицо',

'Паспорт');

AddColumn('ВодительскиеПрава',

'ВодительскиеПрава');

AddTable('СвидетельствоТС',

'Свидетельство о регистрации ТС');

AddColumn('Госномер', 'Госномер');

AddKey('Паспорт', 'Ф.И.О', 'ФИО', 'ФизическоеЛицо',

'Паспорт');

AddKey('ПТС', 'VIN', 'VIN', 'Машина', 'ПТС');

AddColumn('ДатаРегистрации', 'Дата регистрации');

AddTable('ТипНарушения', 'Типы нарушений');

AddColumn('Название', 'Название');

AddColumn('РазмерШтрафа', 'Размер штрафа');

AddTable('ТипУчастника', 'Типы участников');

AddColumn('Статус', 'Статус');

AddTable('ДТП', 'Дорожно-транспортные происшествия');

AddColumn('Постановление', 'Постановление');

AddKey('НомерСотрудника', 'Ф.И.О Сотрудника',

'ФИО', 'Сотрудник', 'НомерСотрудника');

AddColumn('МестоПроисшествия',

'Место происшествия');

AddColumn('Описание', 'Описание');

AddColumn('Дата', 'Дата');

AddTable('УчастникиПешеходы',

'Участвующие в ДТП пешеходы');

AddKey('Постановление', 'Постановление',

'Постановление',

'ДТП', 'Постановление');

AddKey('Паспорт', 'Ф.И.О', 'ФИО', 'ФизическоеЛицо',

'Паспорт');

AddKey('НомерТипаУчастника', 'Статус',

'Статус', 'ТипУчастника',

'НомерТипаУчастника');

AddTable('УчастникиАвтомобили',

'Участвующие в ДТП автомобили');

AddKey('Постановление', 'Постановление',

'Постановление',

'ДТП', 'Постановление');

AddKey('Госномер', 'Госномер', 'Госномер',

'СвидетельствоТС', 'Госномер');

AddKey('Паспорт', 'Ф.И.О', 'ФИО', 'ФизическоеЛицо',

'Паспорт');

AddKey('НомерТипаУчастника', 'Статус',

'Статус', 'ТипУчастника',

'НомерТипаУчастника');

AddTable('Штрафы', 'Штрафы за ДТП');

AddKey('Постановление', 'Постановление',

'Постановление',

'ДТП', 'Постановление');

AddKey('НомерТипаНарушения', 'Тип нарушения',

'Название', 'ТипНарушения',

'НомерТипаНарушения');

AddColumn('Надбавка', 'Надбавка');

AddStatics('РейтингСотрудников',

'Рейтинг сотрудников');

AddColumn('Сотрудник', 'Ф.И.О Сотрудника');

AddColumn('ВсегоПостановлений',

'Всего постановлений');

AddColumn('СуммаШтрафов', 'Сумма штрафов');

AddStatics('РейтингВиновников',

'Рейтинг виновников');

AddColumn('Виновник', 'Ф.И.О Виновника');

AddColumn('Паспорт', 'Паспорт');

AddColumn('ВодительскиеПрава',

'Водительские права');

AddColumn('КоличествоДТП', 'Всего совершено ДТП');

AddColumn('СуммаШтрафов', 'Сумма штрафов');

end;

end.

* 1. Главная форма

Основной функцией этого модуля является создание пунктов меню справочника и статистики. Пункты меню генерируются из модуля Metadata.pas при событии TMainForm.FormCreate.

*Код Main.pas:*

unit Main;

{$mode objfpc}{$H+}

interface

uses

Classes, Forms, Menus, ExtCtrls, Metadata, Catalog, AccidentReg, Report;

type

TDBMenuItem = class(TMenuItem)

private

fModalWin: TDirectory;

public

procedure MenuClick(Sender: TObject);

constructor CreateMenu(TheOwner: TComponent);

end;

{ TMainForm }

TMainForm = class(TForm)

published

BackGround: TImage;

MainMenu: TMainMenu;

CatalogItem: TMenuItem;

ActionItem: TMenuItem;

StatisticsItem: TMenuItem;

ReportItem: TMenuItem;

FilledAccident: TMenuItem;

NewAccident: TMenuItem;

procedure FilledAccidentClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure NewAccidentClick(Sender: TObject);

end;

var

MainForm: TMainForm;

implementation

{$R \*.lfm}

procedure TDBMenuItem.MenuClick(Sender: TObject);

begin

if not Checked then

begin

fModalWin := TDirectory.CreateCatalog(Self);

Checked := True;

end;

fModalWin.ShowOnTop;

end;

constructor TDBMenuItem.CreateMenu(TheOwner: TComponent);

begin

inherited Create(TheOwner);

Tag := TheOwner.Tag;

Caption := Mdata.Tables[Tag].CaptionTable;

OnClick := @MenuClick;

end;

procedure TMainForm.FormCreate(Sender: TObject);

var

i: integer;

begin

for i := 0 to High(Mdata.Tables) do

begin

Tag := i;

if Mdata.Tables[i].ReadOnly then

StatisticsItem.Add(TDBMenuItem.CreateMenu(Self))

else

CatalogItem.Add(TDBMenuItem.CreateMenu(Self));

end;

end;

procedure TMainForm.FilledAccidentClick(Sender: TObject);

begin

ReportForm.Show;

end;

procedure TMainForm.NewAccidentClick(Sender: TObject);

begin

AccidentForm.Show;

end;

end.

* 1. Унифицированная форма справочников

В данных модулях происходит формирование запроса к базе данных на выборку элементов (в виде SELECT запроса). Также в этом модуле происходит обработка событий нажатия на кнопки: добавить, изменить, удалить, окна справочника. Визуально, форма содержит фреймы, реализующие фильтрацию: MiniFilterFrm, FilterFrm.

*Код Catalog.pas:*

unit Catalog;

{$mode objfpc}{$H+}

interface

uses

Classes, DB, sqldb, Forms, Controls, Dialogs, ExtCtrls, DBGrids, StdCtrls,

Buttons, Metadata, Grids, FileCtrl, DBCtrls, Editor, SysUtils, FilterFrm,

MiniFilterFrm, Menus;

type

TDirectory = class(TForm)

private

fTable: TTable;

fQuary: string;

published

AddFilter: TSpeedButton;

MiniFilterFrame1: TMiniFilterFrame;

FilterLabel: TLabel;

FilterPanel: TPanel;

OrderField: TFilterComboBox;

OrderWay: TFilterComboBox;

OrderLabel: TLabel;

DataSource: TDataSource;

DBGrid: TDBGrid;

GridPanel: TPanel;

ControlPanel: TPanel;

AddBtn: TSpeedButton;

EditBtn: TSpeedButton;

DelBtn: TSpeedButton;

SerchBtn: TSpeedButton;

RefreshBtn: TSpeedButton;

SQLQuery: TSQLQuery;

procedure DBGridDrawColumnCell(Sender: TObject; const Rect: TRect;

DataCol: integer; Column: TColumn; State: TGridDrawState);

procedure FormClose(Sender: TObject; var CloseAction: TCloseAction);

procedure AddBtnClick(Sender: TObject);

procedure DelBtnClick(Sender: TObject);

procedure EditBtnClick(Sender: TObject);

procedure RefreshBtnClick(Sender: TObject);

procedure SerchBtnClick(Sender: TObject);

procedure AddFilterClick(Sender: TObject);

constructor CreateCatalog(TheOwner: TComponent);

constructor CreateCatalogByName(TheOwner: TComponent; TableName: string);

end;

implementation

{$R \*.lfm}

function GenSelectSQL(aTable: TTable): string;

var

i: integer;

begin

Result := 'SELECT ';

with aTable do

for i := 0 to High(Columns) do

begin

if i > 0 then

Result += ', ';

with Columns[i] do

begin

Result += NameTable + '.' + DataField;

if Primary then

Result += ', ' + KeyTable + '.' + ListField;

end;

end;

end;

function GenJoinSQL(aTable: TTable): string;

var

i: integer;

begin

with aTable do

begin

Result := NameTable;

for i := 0 to High(Columns) do

with Columns[i] do

if Primary then

Result += ' INNER JOIN ' + KeyTable + ' ON ' + NameTable +

'.' + DataField + ' = ' + KeyTable + '.' + KeyField;

end;

end;

function GetOrderFields(aTable: TTable): string;

const

CQuery = '%s|ORDER BY %s.%s|';

var

i: integer;

begin

Result := '';

with aTable do

for i := 0 to High(Columns) do

with Columns[i] do

if Primary then

Result += Format(CQuery, [CaptionField, KeyTable, ListField])

else

Result += Format(CQuery, [CaptionField, NameTable, DataField]);

Result := LeftStr(Result, Length(Result) - 1);

end;

function GenFromSQL(aTable: TTable): string;

begin

Result := 'FROM ' + GenJoinSQL(aTable);

end;

constructor TDirectory.CreateCatalogByName(TheOwner: TComponent; TableName: string);

var

i, j: integer;

begin

for i := 0 to High(Mdata.Tables) do

if Mdata.Tables[i].NameTable = TableName then

begin

j := TheOwner.Tag;

try

TheOwner.Tag := i;

CreateCatalog(TheOwner);

finally

TheOwner.Tag := j;

end;

end;

end;

constructor TDirectory.CreateCatalog(TheOwner: TComponent);

var

i: integer;

begin

Tag := TheOwner.Tag;

inherited Create(TheOwner);

fTable := Mdata.Tables[Tag];

if fTable.ReadOnly then

begin

AddBtn.Enabled := False;

EditBtn.Enabled := False;

DelBtn.Enabled := False;

end;

with fTable do

begin

Caption := CaptionTable;

// Подготовка таблицы (установка имен, ширины столбцов)

for i := 0 to High(Columns) do

with DBGrid.Columns.Add, Columns[i] do

begin

if Primary then

FieldName := ListField

else

FieldName := DataField;

Title.Caption := CaptionField;

Width := 10 + Canvas.TextWidth(Title.Caption);

end;

end;

OrderField.Filter := GetOrderFields(fTable);

SQLQuery.SQL.Append(GenSelectSQL(fTable));

SQLQuery.SQL.Append(GenFromSQL(fTable));

fQuary := SQLQuery.SQL.Text;

// Текст запроса

//ShowMessage(fQuary);

SQLQuery.Open;

end;

procedure TDirectory.RefreshBtnClick(Sender: TObject);

begin

SQLQuery.SQL.Text := fQuary;

SQLQuery.Refresh;

end;

procedure TDirectory.SerchBtnClick(Sender: TObject);

var

i: integer;

begin

SQLQuery.SQL.Text := fQuary;

SQLQuery.SQL.Append('WHERE ' + MiniFilterFrame1.CompileFilter);

for i := 0 to FilterPanel.ComponentCount - 1 do

if FilterPanel.Components[i] is TFilterFrame then

SQLQuery.SQL.Append((FilterPanel.Components[i] as TFilterFrame).CompileFilter);

SQLQuery.SQL.Append(OrderField.Mask + OrderWay.Mask);

// Текст запроса

//ShowMessage(SQLQuery.SQL.Text);

SQLQuery.Refresh;

end;

procedure TDirectory.AddFilterClick(Sender: TObject);

begin

TFilterFrame.Create(FilterPanel);

end;

procedure TDirectory.FormClose(Sender: TObject; var CloseAction: TCloseAction);

begin

if (Owner is TMenuItem) then

TMenuItem(Owner).Checked := False;

CloseAction := caFree;

end;

procedure TDirectory.DBGridDrawColumnCell(Sender: TObject; const Rect: TRect;

DataCol: integer; Column: TColumn; State: TGridDrawState);

var

MaxWidth: integer = 10;

begin

MaxWidth += Canvas.TextWidth(Column.Field.Text);

if MaxWidth > Column.Width then

Column.Width := MaxWidth;

end;

procedure TDirectory.DelBtnClick(Sender: TObject);

var

selected: integer;

begin

selected := MessageDlg(

'Вы уверены, что хотите удалить выбранную запись?',

mtConfirmation, [mbYes, mbNo], 0);

if selected = mrYes then

TCellEditor.CreateAndDel(Self, fTable, SQLQuery).Free;

end;

procedure TDirectory.EditBtnClick(Sender: TObject);

begin

TCellEditor.CreateAndEdit(Self, fTable, SQLQuery).ShowModal;

end;

procedure TDirectory.AddBtnClick(Sender: TObject);

begin

TCellEditor.CreateAndAdd(Self, fTable, SQLQuery).ShowModal;

end;

end.

*Код NameFrm:*

unit NameFrm;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, FileUtil, Forms, Controls;

type

TNamedFrame = class(TFrame)

published

constructor Create(TheOwner: TWinControl);

end;

implementation

constructor TNamedFrame.Create(TheOwner: TWinControl);

begin

inherited Create(TheOwner);

Parent := TheOwner;

Name := '\_' + IntToStr(cardinal(Self));

end;

end.

*Код BoxFrm:*

unit BoxFrm;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, DB, sqldb, FileUtil, Forms, Controls, StdCtrls, DBCtrls,

NameFrm, Metadata;

type

TBoxFrame = class(TNamedFrame)

published

ListSource: TDataSource;

DBLComboBox: TDBLookupComboBox;

FieldName: TLabel;

SQLQuery: TSQLQuery;

constructor Create(TheOwner: TWinControl; aColumn: TColumns);

end;

implementation

{$R \*.lfm}

constructor TBoxFrame.Create(TheOwner: TWinControl; aColumn: TColumns);

const

CQuary = 'SELECT %s, %s FROM %s';

begin

inherited Create(TheOwner);

with aColumn do

begin

FieldName.Caption := CaptionField;

DBLComboBox.DataField := DataField;

DBLComboBox.KeyField := KeyField;

DBLComboBox.ListField := ListField;

SQLQuery.SQL.Append(Format(CQuary, [KeyField, ListField, KeyTable]));

end;

SQLQuery.Open;

end;

end.

*Код MiniFilterFrm:*

unit MiniFilterFrm;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, FileUtil, ShortPathEdit, Forms, Controls, StdCtrls,

FileCtrl, Metadata;

type

TMiniFilterFrame = class(TFrame)

published

FilterEdit: TEdit;

FilterWay: TFilterComboBox;

FilterField: TFilterComboBox;

constructor Create(TheOwner: TComponent); override;

function CompileFilter: string;

end;

implementation

{$R \*.lfm}

function GetFilterFields(aTable: TTable): string;

const

CQuery = '%s| %s.%s |';

var

i: integer;

begin

Result := '';

with aTable do

for i := 0 to High(Columns) do

with Columns[i] do

if Primary then

Result += Format(CQuery, [CaptionField, KeyTable, ListField])

else

Result += Format(CQuery, [CaptionField, NameTable, DataField]);

Result := LeftStr(Result, Length(Result) - 1);

end;

constructor TMiniFilterFrame.Create(TheOwner: TComponent);

begin

inherited Create(TheOwner);

FilterField.Filter := GetFilterFields(Mdata.Tables[TheOwner.Tag]);

end;

function TMiniFilterFrame.CompileFilter: string;

begin

Result := FilterField.Mask + FilterWay.Mask + '''' + FilterEdit.Text + '''';

end;

end.

*Код FilterFrm:*

unit FilterFrm;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, FileUtil, Forms, Controls, FileCtrl, Buttons, StdCtrls,

MiniFilterFrm, NameFrm;

type

TFilterFrame = class(TNamedFrame)

published

FilterVar: TFilterComboBox;

MiniFilterFrame1: TMiniFilterFrame;

RemoveFilter: TSpeedButton;

function CompileFilter: string;

procedure RemoveFilterClick(Sender: TObject);

end;

implementation

{$R \*.lfm}

function TFilterFrame.CompileFilter: string;

begin

Result := FilterVar.Mask + MiniFilterFrame1.CompileFilter;

end;

procedure TFilterFrame.RemoveFilterClick(Sender: TObject);

begin

Free;

end;

end.

* 1. Унифицированная форма редактирования записей

Задачей этого модуля является runtime-генерация формы редактирования записей справочника, класс TCellEditor. Данный класс имеет 3 конструктора, по одному на каждую операцию: удаление, добавление, редактирование. Также, в этом модуле формируется запрос на редактирование определенной записи в БД. Фреймы EditFrm и BoxFrm – предназначены в качестве полей редактирования записей в БД.

*Код Editor.pas:*

unit Editor;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, DB, sqldb, Forms, Dialogs, StdCtrls, Buttons, Metadata,

EditFrm, BoxFrm;

type

{ TCellEditor }

TCellEditor = class(TForm)

SQLTransaction: TSQLTransaction;

private

fSQLQuery: TSQLQuery;

published

SaveBtn: TBitBtn;

CancelBtn: TBitBtn;

DataSource: TDataSource;

EditPanel: TGroupBox;

ControlPanel: TGroupBox;

Save: TSpeedButton;

Cancel: TSpeedButton;

SQLQuery: TSQLQuery;

procedure CancelBtnClick(Sender: TObject);

constructor CreateOnly(TheOwner: TComponent; aTable: TTable;

aSQLQuery: TSQLQuery);

constructor CreateAndAdd(TheOwner: TComponent; aTable: TTable;

aSQLQuery: TSQLQuery);

constructor CreateAndEdit(TheOwner: TComponent; aTable: TTable;

aSQLQuery: TSQLQuery);

constructor CreateAndDel(TheOwner: TComponent; aTable: TTable;

aSQLQuery: TSQLQuery);

procedure FormClose(Sender: TObject; var CloseAction: TCloseAction);

procedure SaveBtnClick(Sender: TObject);

end;

implementation

{$R \*.lfm}

function GenSelectSQL(aTable: TTable): string;

var

i: integer;

begin

Result := 'SELECT ';

with aTable do

for i := 0 to High(Columns) do

begin

if i > 0 then

Result += ', ';

Result += Columns[i].DataField;

end;

end;

function GenWhereSQL(aTable: TTable; aSQLQuery: TSQLQuery): string;

const

CQuery = ' %s = ''%s''';

CNullQuery = ' %s IS NULL';

var

i: integer;

fl: TField;

begin

Result := 'WHERE';

with aTable do

for i := 0 to High(Columns) do

begin

if i > 0 then

Result += ' AND';

with Columns[i] do

begin

fl := aSQLQuery.FieldByName(DataField);

if fl.IsNull then

Result += Format(CNullQuery, [DataField])

else

Result += Format(CQuery, [DataField, fl.Text]);

end;

end;

end;

function GenFromSQL(aTable: TTable): string;

begin

Result := 'FROM ' + aTable.NameTable;

end;

constructor TCellEditor.CreateOnly(TheOwner: TComponent; aTable: TTable;

aSQLQuery: TSQLQuery);

var

i: integer;

begin

inherited Create(TheOwner);

fSQLQuery := aSQLQuery;

with aTable do

begin

Caption := 'Редактор таблицы "' + CaptionTable + '"';

for i := 0 to High(Columns) do

if Columns[i].Primary then

TBoxFrame.Create(EditPanel, Columns[i])

else

TEditFrame.Create(EditPanel, Columns[i]);

end;

SQLQuery.SQL.Append(GenSelectSQL(aTable));

SQLQuery.SQL.Append(GenFromSQL(aTable));

SQLQuery.SQL.Append(GenWhereSQL(aTable, aSQLQuery));

// Текст запроса

//Dialogs.ShowMessage(SQLQuery.SQL.Text);

SQLQuery.Open;

end;

constructor TCellEditor.CreateAndAdd(TheOwner: TComponent; aTable: TTable;

aSQLQuery: TSQLQuery);

begin

CreateOnly(TheOwner, aTable, aSQLQuery);

SQLQuery.Insert;

end;

constructor TCellEditor.CreateAndEdit(TheOwner: TComponent; aTable: TTable;

aSQLQuery: TSQLQuery);

begin

CreateOnly(TheOwner, aTable, aSQLQuery);

end;

constructor TCellEditor.CreateAndDel(TheOwner: TComponent; aTable: TTable;

aSQLQuery: TSQLQuery);

begin

CreateOnly(TheOwner, aTable, aSQLQuery);

try

SQLQuery.Delete;

except

// TODO: Обработать все исключения

on e: EDatabaseError do

MessageDlg('Не удалось удалить выбранную запись',

mtError, [mbOK], 0);

end;

SaveBtnClick(Self);

end;

procedure TCellEditor.FormClose(Sender: TObject; var CloseAction: TCloseAction);

begin

CloseAction := caFree;

end;

procedure TCellEditor.SaveBtnClick(Sender: TObject);

begin

try

SQLQuery.ApplyUpdates;

// Если через Commit, то закрывается TSQLQuery

SQLQuery.SQLTransaction.CommitRetaining;

fSQLQuery.Refresh;

Close;

except

// TODO: Обработать все исключения

on e: EDatabaseError do

MessageDlg('Произошла ошибка при применении изменений', mtError, [mbOK], 0);

end;

end;

procedure TCellEditor.CancelBtnClick(Sender: TObject);

begin

Close;

end;

end.

*Код EditFrm:*

unit EditFrm;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, FileUtil, Forms, Controls, StdCtrls, DBCtrls, NameFrm,

Metadata;

type

TEditFrame = class(TNamedFrame)

published

FieldEditor: TDBEdit;

FieldName: TLabel;

constructor Create(TheOwner: TWinControl; aColumn: TColumns);

end;

implementation

{$R \*.lfm}

constructor TEditFrame.Create(TheOwner: TWinControl; aColumn: TColumns);

begin

inherited Create(TheOwner);

FieldName.Caption := aColumn.CaptionField;

FieldEditor.DataField := aColumn.DataField;

end;

end.

*Код BoxFrm:*

unit BoxFrm;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, DB, sqldb, FileUtil, Forms, Controls, StdCtrls, DBCtrls,

NameFrm, Metadata;

type

TBoxFrame = class(TNamedFrame)

published

ListSource: TDataSource;

DBLComboBox: TDBLookupComboBox;

FieldName: TLabel;

SQLQuery: TSQLQuery;

constructor Create(TheOwner: TWinControl; aColumn: TColumns);

end;

implementation

{$R \*.lfm}

constructor TBoxFrame.Create(TheOwner: TWinControl; aColumn: TColumns);

const

CQuary = 'SELECT %s, %s FROM %s';

begin

inherited Create(TheOwner);

with aColumn do

begin

FieldName.Caption := CaptionField;

DBLComboBox.DataField := DataField;

DBLComboBox.KeyField := KeyField;

DBLComboBox.ListField := ListField;

SQLQuery.SQL.Append(Format(CQuary, [KeyField, ListField, KeyTable]));

end;

SQLQuery.Open;

end;

end.

* 1. Форма регистрации ДТП

Данная форма состоит из нескольких частей: полей с информацией о ДТП, панели навигации по существующим происшествиям и карты с маркерами о месте происшествия. Для вывода карты используется компонент fpCEF, который предоставляет web-интерфейс и js-скрипт из файла maps\_template.html, который осуществляет взаимодействие с сервисами Google Maps.

*Код AccidentReg.pas:*

unit AccidentReg;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, DB, BufDataset, sqldb, SdfData, memds, FileUtil, Forms,

Controls, Graphics, Dialogs, DBCtrls, DBGrids, ExtCtrls, PairSplitter,

StdCtrls, Buttons, FileCtrl, ComCtrls, cef3types, cef3lib, cef3intf, cef3lcl,

Catalog, cef3gui, Fpjson, jsonparser;

type

{ TAccidentForm }

TAccidentForm = class(TForm)

FindAddrButton: TButton;

CommitChanges: TBitBtn;

DTPCustomerDataSource: TDataSource;

CustomerLookupCB: TDBLookupComboBox;

CustomerLabel: TLabel;

NumberEdit: TDBEdit;

PlaceEdit: TDBEdit;

LatEdit: TDBEdit;

LngEdit: TDBEdit;

DBNavigator: TDBNavigator;

DTPCustomerSQLQuery: TSQLQuery;

DescriptionMemo: TDBMemo;

DTPDataSource: TDataSource;

GoogleMap: TChromium;

DTPPanel: TPanel;

DTPSQLQuery: TSQLQuery;

DescriptionLabel: TLabel;

PlaceLabel: TLabel;

NumberLabel: TLabel;

LatLabel: TLabel;

LngLabel: TLabel;

DTPToolBar: TToolBar;

CustomersToolButton: TToolButton;

AutoToolButton: TToolButton;

SanctionsToolButton: TToolButton;

procedure CommitChangesClick(Sender: TObject);

procedure DBNavigatorBeforeAction(Sender: TObject; Button: TDBNavButtonType);

procedure DTPDataSourceDataChange(Sender: TObject; Field: TField);

procedure DTPDataSourceStateChange(Sender: TObject);

procedure DTPDataSourceUpdateData(Sender: TObject);

procedure DTPSQLQueryAfterDelete(DataSet: TDataSet);

procedure DTPSQLQueryAfterRefresh(DataSet: TDataSet);

procedure FindAddrButtonClick(Sender: TObject);

procedure FormCloseQuery(Sender: TObject; var CanClose: boolean);

procedure FormCreate(Sender: TObject);

procedure FormShow(Sender: TObject);

procedure GoogleMapBeforePopup(Sender: TObject; const browser: ICefBrowser;

const frame: ICefFrame; const targetUrl, targetFrameName: ustring;

targetDisposition: TCefWindowOpenDisposition; userGesture: boolean;

var popupFeatures: TCefPopupFeatures; var windowInfo: TCefWindowInfo;

var client: ICefClient; var settings: TCefBrowserSettings;

var noJavascriptAccess: boolean; out Result: boolean);

procedure GoogleMapConsoleMessage(Sender: TObject;

const Browser: ICefBrowser; const message, Source: ustring;

line: integer; out Result: boolean);

procedure CustomersToolButtonClick(Sender: TObject);

procedure AutoToolButtonClick(Sender: TObject);

procedure GoogleMapLoadEnd(Sender: TObject; const Browser: ICefBrowser;

const Frame: ICefFrame; httpStatusCode: integer);

procedure SanctionsToolButtonClick(Sender: TObject);

private

procedure MoveGoogleMark();

public

end;

var

AccidentForm: TAccidentForm;

implementation

{$R \*.lfm}

{ TAccidentForm }

procedure TAccidentForm.MoveGoogleMark();

const

Coord: string = 'marker.setPosition({lat: %g, lng: %g});';

var

ds: char;

c: string;

flat, flng: TField;

begin

flat := DTPSQLQuery.FieldByName('Широта');

flng := DTPSQLQuery.FieldByName('Долгота');

if (flat.IsNull or flng.IsNull) then

exit;

ds := DecimalSeparator;

DecimalSeparator := '.';

try

c := Format(Coord, [flat.AsFloat, flng.AsFloat]);

GoogleMap.Browser.MainFrame.ExecuteJavaScript(c, 'about:blank', 0);

finally

DecimalSeparator := ds;

end;

end;

procedure TAccidentForm.FormCreate(Sender: TObject);

begin

{$IFDEF DARWIN}

// Uncomment for a single process application

//CefSingleProcess := True;

{$ELSE}

{$INFO subprocess is set here, comment out to use the main program as subprocess}

CefBrowserSubprocessPath := '.' + PathDelim + 'subprocess'

{$IFDEF WINDOWS}

+ '.exe'

{$ENDIF}

;

{$ENDIF}

GoogleMap.DefaultUrl := GetCurrentDir + PathDelim + 'maps\_tamplate.html';

DTPCustomerSQLQuery.Open;

DTPSQLQuery.Open;

end;

procedure TAccidentForm.DTPDataSourceDataChange(Sender: TObject; Field: TField);

begin

MoveGoogleMark;

end;

procedure TAccidentForm.DTPDataSourceStateChange(Sender: TObject);

begin

CommitChanges.Enabled := (DTPSQLQuery.ChangeCount > 0) or

(DTPSQLQuery.State in [dsEdit, dsInsert]);

end;

procedure TAccidentForm.DTPDataSourceUpdateData(Sender: TObject);

var

i: integer;

begin

with DTPDataSource.DataSet do

for i := 0 to FieldCount - 1 do

if Fields[i].Required then

Fields[i].Required := False;

end;

procedure TAccidentForm.DTPSQLQueryAfterDelete(DataSet: TDataSet);

begin

CommitChanges.Enabled := True;

end;

procedure TAccidentForm.DTPSQLQueryAfterRefresh(DataSet: TDataSet);

begin

CommitChanges.Enabled := False;

end;

procedure TAccidentForm.FindAddrButtonClick(Sender: TObject);

const

Address: string = 'codeAddress(''%s'', %s);';

var

c, f: string;

begin

if (LatEdit.Field.IsNull or LngEdit.Field.IsNull) then

f := 'true'

else

f := 'false';

c := Format(Address, [PlaceEdit.Field.AsString, f]);

GoogleMap.Browser.MainFrame.ExecuteJavaScript(c, 'about:blank', 0);

end;

procedure TAccidentForm.FormCloseQuery(Sender: TObject; var CanClose: boolean);

const

q = 'Внесенные изменения не будут сохранены, выйти?';

begin

if (DTPSQLQuery.State = dsBrowse) and (DTPSQLQuery.ChangeCount = 0) then

CanClose := True

else

if MessageDlg(q, mtConfirmation, [mbYes, mbNo], 0) = mrYes then

begin

DTPSQLQuery.CancelUpdates;

CanClose := True;

end

else

CanClose := False;

end;

procedure TAccidentForm.CommitChangesClick(Sender: TObject);

begin

try

DTPSQLQuery.ApplyUpdates;

DTPSQLQuery.SQLTransaction.CommitRetaining;

CommitChanges.Enabled := False;

except

// TODO: Обработать все исключения

on e: EDatabaseError do

MessageDlg('Произошла ошибка при применении изменений', mtError, [mbOK], 0);

end;

end;

procedure TAccidentForm.DBNavigatorBeforeAction(Sender: TObject;

Button: TDBNavButtonType);

begin

case Button of

nbRefresh:

DTPSQLQuery.CancelUpdates;

nbDelete: if MessageDlg(

'Вы дейтсвительно хотите удалить ДТП?',

mtConfirmation, [mbYes, mbNo], 0) = mrNo then

Abort;

end;

end;

procedure TAccidentForm.FormShow(Sender: TObject);

begin

DTPCustomerSQLQuery.Refresh;

DTPSQLQuery.Refresh;

end;

procedure TAccidentForm.GoogleMapBeforePopup(Sender: TObject;

const browser: ICefBrowser; const frame: ICefFrame;

const targetUrl, targetFrameName: ustring;

targetDisposition: TCefWindowOpenDisposition; userGesture: boolean;

var popupFeatures: TCefPopupFeatures; var windowInfo: TCefWindowInfo;

var client: ICefClient; var settings: TCefBrowserSettings;

var noJavascriptAccess: boolean; out Result: boolean);

begin

Result := True;

end;

procedure TAccidentForm.GoogleMapConsoleMessage(Sender: TObject;

const Browser: ICefBrowser; const message, Source: ustring;

line: integer; out Result: boolean);

var

J: TJSONData;

lat, lng: double;

begin

try

J := GetJSON(UTF8Encode(message));

try

lat := J.FindPath('lat').AsFloat;

lng := J.FindPath('lng').AsFloat;

finally

J.Free;

end;

if not (DTPSQLQuery.State in [dsEdit, dsInsert]) then

DTPSQLQuery.Edit;

LatEdit.Field.Value := lat;

LngEdit.Field.Value := lng;

except

// TODO: Обработать исключение

on e: EJSONParser do

begin

DTPDataSource.OnDataChange := nil;

FindAddrButton.Enabled := False;

GoogleMap.Load(GetCurrentDir + PathDelim + 'error\_page.html');

end;

end;

Result := False;

end;

procedure TAccidentForm.CustomersToolButtonClick(Sender: TObject);

var

cat: TDirectory;

begin

cat := TDirectory.CreateCatalogByName(Self, 'УчастникиПешеходы');

cat.Position := poOwnerFormCenter;

cat.ShowModal;

end;

procedure TAccidentForm.AutoToolButtonClick(Sender: TObject);

var

cat: TDirectory;

begin

cat := TDirectory.CreateCatalogByName(Self,

'УчастникиАвтомобили');

cat.Position := poOwnerFormCenter;

cat.ShowModal;

end;

procedure TAccidentForm.GoogleMapLoadEnd(Sender: TObject;

const Browser: ICefBrowser; const Frame: ICefFrame; httpStatusCode: integer);

begin

if ExtractFileName(Browser.MainFrame.Url) <> 'maps\_tamplate.html' then

exit;

DTPDataSource.OnDataChange := @AccidentForm.DTPDataSourceDataChange;

FindAddrButton.Enabled := True;

MoveGoogleMark;

end;

procedure TAccidentForm.SanctionsToolButtonClick(Sender: TObject);

var

cat: TDirectory;

begin

cat := TDirectory.CreateCatalogByName(Self, 'Штрафы');

cat.Position := poOwnerFormCenter;

cat.ShowModal;

end;

end.

*Код maps\_template.html:*

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Google Maps Marker</title>

<style>

/\* Always set the map height explicitly to define the size of the div

\* element that contains the map. \*/

#map {

height: 100%;

}

/\* Optional: Makes the sample page fill the window. \*/

html, body {

height: 100%;

margin: 0;

padding: 0;

}

</style>

</head>

<body>

<div id="map"></div>

<script>

*var* map;

*var* marker;

*var* geocoder;

*var* city = 'Владивосток';

*function* initMap() {

geocoder = *new* google.maps.Geocoder;

geocoder.geocode({'address': city}, *function*(results, status) {

*if* (status == 'OK') {

*var* myLatLng = results[0].geometry.location;

*var* mapOptions = {

zoom: 13,

center: myLatLng,

disableDefaultUI: *true*

}

map = *new* google.maps.Map(document.getElementById('map'), mapOptions);

marker = *new* google.maps.Marker({

map: map,

position: myLatLng,

draggable: *true*,

animation: google.maps.Animation.DROP,

title: 'Место ДТП'

});

marker.addListener('click', toggleBounce);

marker.addListener('dragend', consoleLog);

} *else* {

*throw* *new* Error('Geocode was not successful for the following reason: ' + status);

}

});

}

*function* codeAddress(address, moveMarker) {

geocoder.geocode( { 'address': city + ', ' + address}, *function*(results, status) {

*if* (status == 'OK') {

*var* myLatLng = results[0].geometry.location;

map.setCenter(myLatLng);

*if* (moveMarker === *true*)

marker.setPosition(myLatLng);

} *else* {

*throw* *new* Error('Geocode was not successful for the following reason: ' + status);

}

});

}

*function* toggleBounce() {

*if* (marker.getAnimation() !== *null*) {

marker.setAnimation(*null*);

} *else* {

marker.setAnimation(google.maps.Animation.BOUNCE);

}

}

*function* consoleLog() {

*var* str = JSON.stringify(marker.getPosition());

console.log(str);

}

</script>

<script async defer

src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBLabKpo3SkqOSRWzDLuWFS-NmaAM3df7A&language=ru&callback=initMap">

</script>

</body>

</html>

*Код error\_page.html:*

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Some Error Happened</title>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

.unselectable {

-webkit-user-select: none;

-moz-user-select: none;

-ms-user-select: none;

user-select: none;

}

\* {

line-height: 1.2;

margin: 0;

}

html {

color: #888;

display: table;

font-family: sans-serif;

height: 100%;

text-align: center;

width: 100%;

}

body {

display: table-cell;

vertical-align: middle;

margin: 2em auto;

}

h1 {

color: #555;

font-size: 2em;

font-weight: 400;

}

p {

margin: 0 auto;

width: 280px;

}

@media only screen and (max-width: 280px) {

body, p {

width: 95%;

}

h1 {

font-size: 1.5em;

margin: 0 0 0.3em;

}

}

</style>

</head>

<body class="unselectable">

<h1>Произошла ошибка</h1>

<p>Извините, не удалось отобразить карту ДТП.</p>

</body>

</html>

<!-- IE needs 512+ bytes: https://blogs.msdn.microsoft.com/ieinternals/2010/08/18/friendly-http-error-pages/ -->

*Код программы subrocess.lpr (для плавного отображения карты):*

Program subprocess;

{$mode objfpc}{$H+}

Uses

{$IFDEF UNIX}

cthreads,

{$ENDIF}

cef3types, cef3api;

Var

Args : TCefMainArgs;

begin

CefLoadLibrary;

{$IFDEF WINDOWS}

Args.instance := HINSTANCE();

Halt(cef\_execute\_process(@Args, nil, nil));

{$ELSE}

Args.argc := argc;

Args.argv := argv;

Halt(cef\_execute\_process(@Args, nil, nil));

{$ENDIF}

end.

* 1. Форма печати бланков постановлений

Код модуля, отвечающий за отображения и печать бланков постановлений достаточно компактный, в виду того, что основной задачей данного модуля является отображение отчета из шаблона report.lrf (в отчете не приводится т. к. содержит 3011 строк) при помощи компонента LazReport.

*Код модуля Report.pas:*

unit Report;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, DB, sqldb, FileUtil, PrintersDlgs, LR\_Class, LR\_DBSet,

LR\_View, LR\_PGrid, Forms, Controls, Graphics, Dialogs,

StdCtrls, DBGrids, ExtCtrls, DBCtrls;

type

{ TReportForm }

TReportForm = class(TForm)

PrintButton: TButton;

DBReportNavigator: TDBNavigator;

DBReportGrid: TDBGrid;

DTPReportDS: TDataSource;

frDTPDataSetReport: TfrDBDataSet;

frDTPReport: TfrReport;

frDTPPreview: TfrPreview;

ReportPrintDialog: TPrintDialog;

ReportPanel: TPanel;

SQLDTPReport: TSQLQuery;

procedure PrintButtonClick(Sender: TObject);

procedure DTPReportDSDataChange(Sender: TObject; Field: TField);

procedure FormCreate(Sender: TObject);

procedure FormShow(Sender: TObject);

private

public

end;

var

ReportForm: TReportForm;

implementation

{$R \*.lfm}

{ TReportForm }

procedure TReportForm.DTPReportDSDataChange(Sender: TObject; Field: TField);

begin

frDTPReport.ShowReport;

end;

procedure TReportForm.PrintButtonClick(Sender: TObject);

begin

if frDTPReport.PrepareReport and ReportPrintDialog.Execute then

frDTPReport.PrintPreparedReport('', 1);

end;

procedure TReportForm.FormCreate(Sender: TObject);

begin

SQLDTPReport.Open;

frDTPReport.LoadFromFile('report.lrf');

end;

procedure TReportForm.FormShow(Sender: TObject);

begin

SQLDTPReport.Refresh;

end;

end.