**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**РОССИЙСКАЯ ФЕДЕРАЦИЯ**

**Федеральное государственное автономное образовательное учреждение**

**высшего образования**

**«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК

КАФЕДРА ИНФОРМАЦИОННЫХ СИСТЕМ

КУРСОВАЯ РАБОТА

по дисциплине «Информатика и программирование»

«Разработка приложения для автоматизация учета нарушений правил дорожного движения и штрафов»

Исполнитель:

студент Мячиков Т. C.

группа 189-1

Руководитель:

канд. физ.- мат. наук,

доцент Моор П. К.

Тюмень - 2019

**1. Постановка задачи.**

Для выполнения работы были поставлены следующие задачи:

1. Разработать Windows Forms приложение на языке C#, реализующее основные функции СУБД для предметной области (добавление, удаление, редактирование данных, выборка). Сохранение файлов идет в формате XML.
2. Разработать БД и приложение Access по предметной области.
3. Реализовать учет в среде Excel.

Предметная область: нарушения правил дорожного движения и штрафы. Потребуются следующие таблицы для обработки и хранения данных:

1. Водители. Таблица будет содержать имя водителя, категорию его прав, дату рождения и уникальный ключ-идентификатор для каждого водителя.
2. Нарушения. Таблица будет содержать информацию о названии нарушения, наказания за него и ключ-идентификатор.
3. Штрафы. Таблица использует уникальные ключи из 2 предыдущих таблиц для хранения информации о том, кто совершил нарушение и какое нарушение это было. В дополнение к ним будет новое поле “дата”, содержащее дату нарушения.

Также в каждой таблице будет свойство “удалено”, если это свойство истинно, то данная запись отображаться не будет.

Для сохранения информации используются следующие модели данных:

Файл “водители” drivers.xml

Структура записи:

**Таблица 1 – файл drivers.xml**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Код водителя** | Фамилия | Категория | Дата рождения | Удалено |
| driverCode | name | category | bDay | removed |
| 1 | Иванов | B | 01/02/1995 | false |
| 2 | Петров | B | 02/03/1996 | false |

Файл “виды нарушений” violations.xml

Структура записи:

**Таблица 2 – файл violations.xml**

|  |  |  |  |
| --- | --- | --- | --- |
| Код нарушения | Название нарушения | Наказание | Удалено |
| violationCode | violation | punishment | removed |
| 1 | Проезд на красный свет | Штраф от 500 руб. | false |
| 2 | Превышение скорости | Штраф 1000 руб. | false |

Файл “штрафы” fines.xml

Структура записи:

**Таблица 3 – файл fines.xml**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Код штрафа** | Код водителя  вн.ключ | Код нарушения  вн. ключ | Дата нарушения | Удалено |
| fineCode | driverCode | violationCode | date | removed |
| 1 | 2 | 1 | 22/04/2019 | false |
| 2 | 1 | 1 | 21/04/2019 | false |

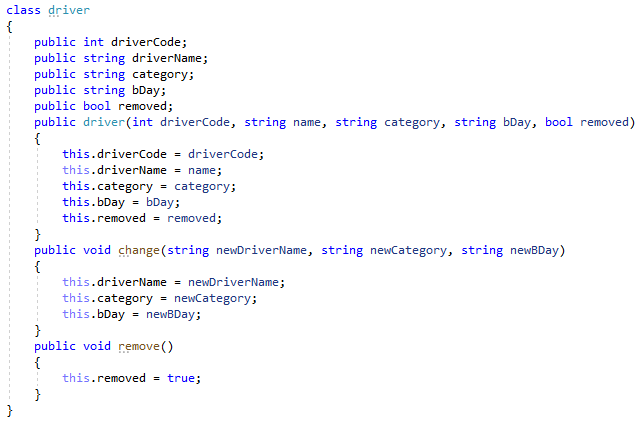
Хранение и обработка данных будет производится с помощью классов.

**2. Описание алгоритмов, классов и методов.**

**2.1 Описание классов.**

Для обработки таблиц были написаны 3 класса, каждый класс соответствует своей таблице.

**2.1.1 Класс driver.**

****

**Рис. 1 – Определение класса “driver”**

Класс содержит 5 свойств, 2 метода и конструктор.

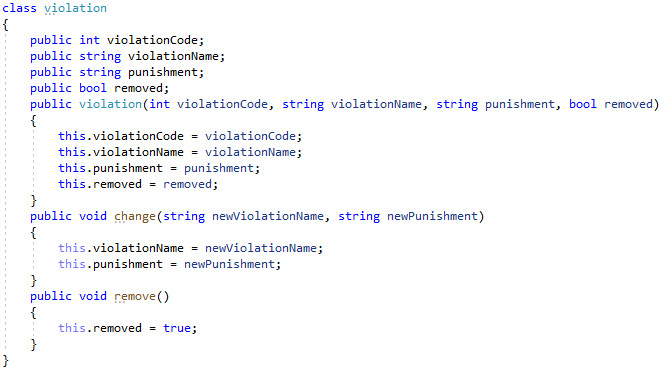
Свойства:

1. driverCode – ключ-идентификатор водителя;
2. driverName – имя водителя;
3. category – категория прав водителя;
4. bDay – дата рождения водителя;
5. removed – удалена ли запись, если значение ‘true’, то запись не отображается.

Методы:

1. change(newDriverName, newCategory, newBDay) – обновляет свойства объекта класса: newDriverName обновляет имя водителя, newCategory обновляет категорию, newBDay обновляет дату рождения.
2. remove() – удаляет запись путем изменения свойства removed на “true”.

**2.1.2 Класс violation.**



**Рис. 2 – Определение класса “violation”**

Класс содержит 4 свойства, 2 метода и конструктор.

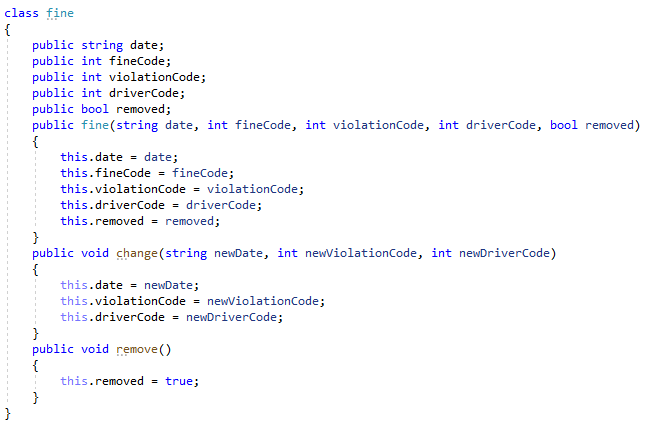
Свойства:

1. violationCode – ключ-идентификатор нарушения;
2. violationName – название нарушения;
3. punishment – наказание за нарушение;
4. removed – удалена ли запись.

Методы:

1. change(newViolationName, newPunishment) – меняет значения объекта класса: newViolationName – название нарушения, newPunishment – новое наказание.
2. remove() – удаляет запись путем изменения свойства removed на “true”.

**2.1.3 Класс fine.**



**Рис. 3 – Определение класса “fine”**

Класс содержит 5 свойств, 2 метода и конструктор.

Свойства:

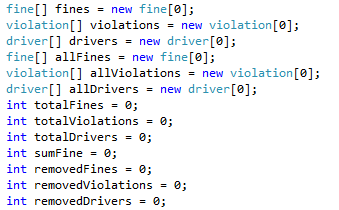
1. date – дата нарушения;
2. fineCode – код штрафа;
3. violationCode – код нарушения, берется из таблицы “нарушения”;
4. driverCode – код водителя, берется из таблицы “водители”;
5. removed – удален ли объект.

Методы:

1. change(newDate, newViolationCode, newDriverCode) – меняет значения
2. remove() – удаляет запись путем изменением свойства removed на “true”.

**2.2 Глобальные переменные.**

В проекте используются 12 глобальных переменных, которые можно разбить на 4 условные группы, по 3 в каждой.



**Рис. 4 – глобальные переменные.**

Массивы fines, violations и drivers содержат все объекты своих классов с свойством “removed” с значением “false”.

Массивы allFines, allViolations, allDrivers содержат в том числе и элементы, у которых значение свойства “removed” равно “true”, то есть в отличие от предыдущих массивов они также содержат объекты, помеченные как удаленные.

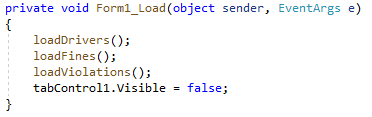
Переменные totalFines, totalViolations и totalDrivers хранят общее количество объектов для своих классов соответственно и используются при загрузке элементов и определяют ключ при добавлении элемента. Переменная sumFine хранит сумму штрафов в базе данных и используется при создании итоговой таблицы.

Переменные removedFines, removedViolations и removedDrivers хранят количество удаленных элементов во время нынешней сессии и используются при перерисовке таблицы после удаления элементов.

**2.3 Описание методов на уровне класса формы.**

**2.3.1 Методы загрузки и отображения элементов**

При загрузке формы вызываются методы loadDrivers(), loadFines() и loadViolations(), которые осуществляют загрузку объектов в каждую таблицу.

****

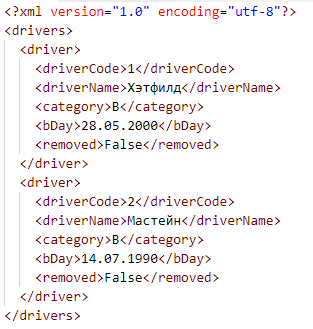
**Рис. 5 – определение метода “form1\_load”**

Также прячется контроль вкладок с добавлением и изменением элементов в таблице, так как при запуске ни одна из таблиц не открыта.

Логика работы методов загрузки:

1. Очищение массивов, содержащих, как только не удалённые, так и все элементы;
2. Создание объекта XMLDocument и загрузка файла.
3. Открывается первый цикл foreach, который обрабатывает каждый элемент в XML.
4. Открывается второй цикл foreach, определяющий свойство и присваивающее его значение переменным, к которым потом будет обращаться конструктор объекта.
5. Если значение объекта “removed” равно “false”, то идет увеличение размера массива объектов на 1 и вызывается конструктор объекта.
6. Независимо от значения свойства “removed” увеличивается размер массива, содержащего все элементы, и создается объект.

Если в таблице нет элементов, то появляется надпись “В таблице пусто”.



**Рис. 7 – файл “drivers.xml”**

При изменении выбранной таблицы в comboBox’е вызывается функция вывода элементов таблицы в dataGridView и в этой функции вызывается функция изменения вкладок добавления и изменения элементов, для каждой таблицы свой метод. Если выводится общая таблица, то вкладки добавления и изменения элементов не отображаются.

Все методы вывода элементов основных таблиц работают по одной схеме:

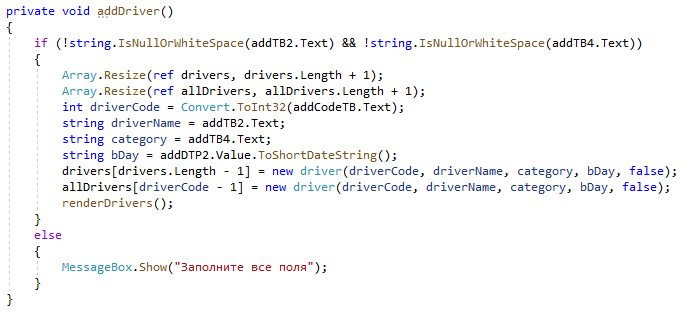
1. Задается нужное количество строк и столбцов;
2. Устанавливается ширина столбцов и их подпись;
3. В цикле перебираются все элементы массива fines, violations или drivers в зависимости от таблицы;
4. Вызывается метод отображения всех элементов формы (подписи, текстовые поля, comboBox’ы);
5. Ненужные для данной таблицы элементы скрываются, к примеру, если открыта таблица “штрафы”, то скрываются текстовые поля и на их месте появляются comboBox’ы, в которых пользователь выбирает водителя и нарушение, которое он совершил. Если открыта таблица ”нарушение”, то скрывается выбор даты.

**2.3.2 Метод добавления элементов в таблицу**

При нажатии на кнопку “добавить” на вкладке добавления элемента вызывается метод добавления элемента, для каждой таблицы свой метод.

Схема работы функций добавления элементов:

1. Идет проверка необходимых полей, где указаны данные, которые будут передаваться как свойства объектов.
2. Если поля непустые, то размеры массивов увеличиваются и вызывается конструктор объекта.
3. Вызывается метод render(), который перерисовывает таблицу с обновленными данными.



**Рис. 8 – определение метода addDriver()**

**2.3.3 Методы сохранения, изменения и удаления элементов**

При нажатии на кнопку “сохранить” вызывается метод сохранения элементов в зависимости от выбранной таблицы.

Логика работа методов сохранения:

1. Создается объект XMLDocument и загрузка файла.
2. Файл очищается.
3. В цикле foreach создаются переменные, содержащие свойства объектов, теги, и значения этих переменных присоединяются к тегам. После этого каждый тег присоединяется к объекту и каждый объект присоединяется к корневому элементу.
4. Сохранение XML-файла.

При нажатии на кнопку ”изменить” проверяются значения необходимых полей, и если они не пустые, то объект вызывает метод change(), куда передаются данные из полей и после это вызывается метод render() для обновления данных в таблице.

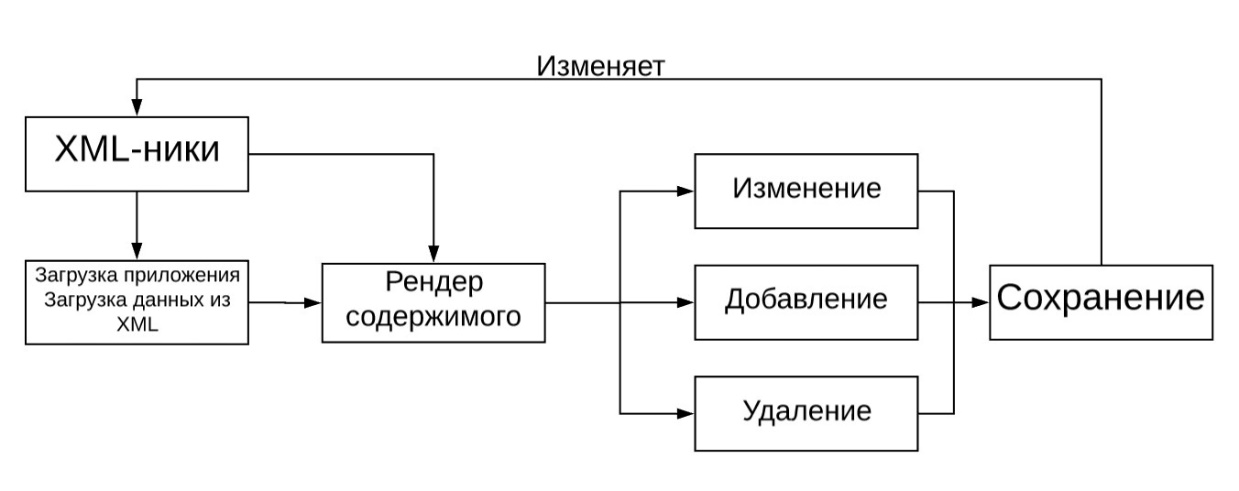
Нажатие на копку “удалить” вызывает метод delete() на экземпляр класса.

**2.3.4 Методы, используемые в выборке.**

В выборке есть выбор между двумя таблицами: водители и нарушения, в зависимости от выбранной таблицы заполняется comboBox с возможными значениями для данной таблицы. К примеру, если выбрана таблица “водители”, то в comboBox добавляются все водители в таблице. Если в выбранной таблице пусто, то отображается надпись “в таблице пусто”.

При нажатии на кнопку “показать" вызывается функция заполнения dataGridView выборки:

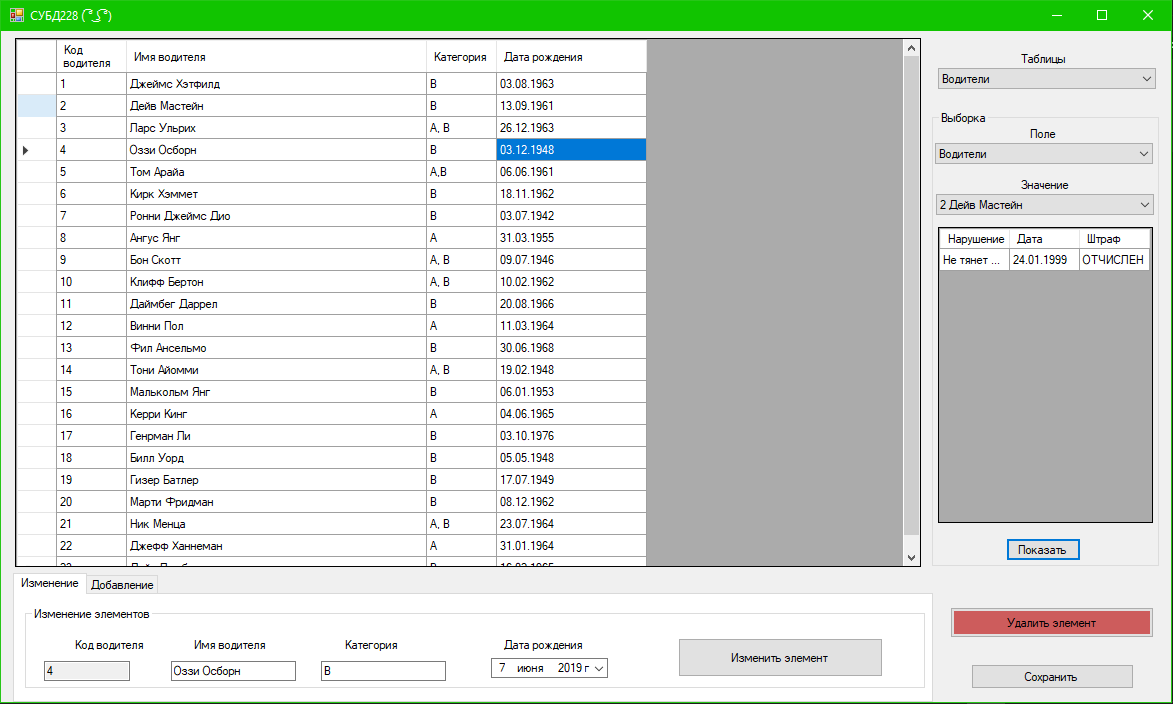
1. В цикле foreach в штрафах ищутся совпадения полей driverCode или violationCode (в зависимости от выбранной таблицы), на каждое совпадение увеличивается счетчик. По окончании цикла количество строк в таблице приравнивается счетчику.
2. В зависимости от выбранной таблицы устанавливается количество столбцов, их заголовки и их ширина.
3. В цикле перебираются все элементы массива и если свойство объекта совпадает со свойством, которое задал пользователь, то оно добавляется в выборку.



**Рис. 9 – логика работы программы**

**3. Описание пользовательского интерфейса.**

Пользовательский интерфейс программы состоит из одной формы, где собраны все необходимые элементы управления и чтения информации.

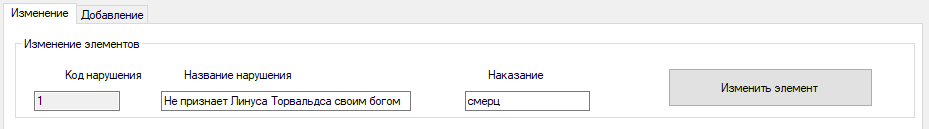


**Рис. 10 – интерфейс программы**

Большую часть интерфейса занимает dataGridView, которая содержит элементы выбранной таблицы. Если в таблице пусто, то отображается надпись “В таблице пусто”. Ширина столбцов и их заголовки задаются программно в зависимости от выбранной таблицы.

Справа расположены элементы выбора таблица, выборка, кнопка удаления элемента и кнопка сохранения таблицы.

Под таблицей расположен выбор вкладок изменения и добавления элемента. Если в таблице пусто, то вкладка “изменение” недоступна. Вкладки меняются в зависимости от выбранной таблицы. Если выбрана таблица “штрафы”, то textBox’ы меняются на comboBox’ы

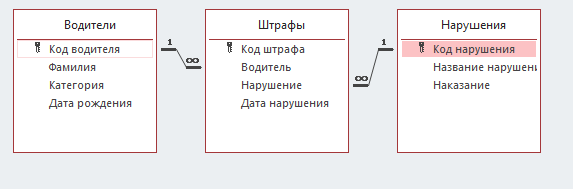


**Рис. 11 вкладка “Изменение”**

**4. Описание базы данных Access.**

**4.1 Таблицы.**

В базе данных используются 3 таблицы: водители, штрафы и нарушения.



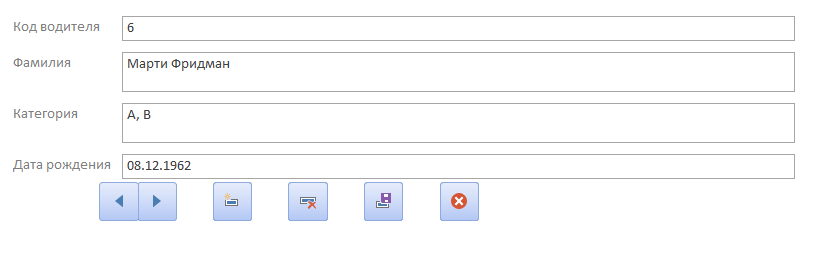
**Рис. 12 Схема данных**

Поля в каждой таблице Access идентичны полям в программе и XML-файлах.

**4.2 Формы.**

В базе данных используется 2 вида форм. Первая из них осуществляет навигацию по таблицам, удаление, добавление или изменение элементов.

Второй вид форм нужен для получения запросов с параметром. Параметр выбирается в выпадающем списке и по нажатии на кнопку выдается результат выборки.



**Рис. 13 Форма “водители”**

**4.3 Запросы.**

В базе данных есть 4 запроса, которые выдают все штрафы, полный запрос, а также 2 вспомогательные которые используются в формах с запросами. По полному запросу построен отчет.