МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

ГЛАВНОЕ УПРАВЛЕНИЕ ПО ОБРАЗОВАНИЮ

МОГИЛЕВСКОГО ОБЛАСТНОГО ИСПОЛНИТЕЛЬНОГО КОМИТЕТА

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«МОГИЛЕВСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»

Специальность 2-40 01 01

Учебная группа ПО-455

Учебная дисциплина Конструирование программ и языки программирования

**КУРСОВОЙ ПРОЕКТ**

**РАЗРАБОТКА** **АВТОМАТИЗИРОВАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ**

**«АВТОБУСНЫЙ АВТОПАРК»**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КП ПО.00.00.000 ПЗ**

Разработал A.О.Савич

Руководитель проекта А.В.Карманов

2022

Содержание

Введение 4

1 Постановка задачи 5

1.1 Описание предметной области 6

1.2 Актуальность решаемой задачи 6

1.3 Характеристика решаемой задачи 6

2. Проектирование программного модуля 7

2.1 Разработка модели данных 7

2.2 Выбор программного обеспечения 16

2.3 Определение требований к техническим средствам 17

3. Реализация программного модуля 18

3.1 Определение формы представления входных и выходных данных 18

3.2 Тестирование программного модуля 24

3.3Защита информации 25

3.4Разработка справочной системы 25

4 Энерго- и ресурсосбережение 27

Заключение 28

Список используемых источников 29

Приложение А Текст программы 30

Приложение Б Входные и выходные формы

Приложение В Описание программы

**Введение**

За последние десятилетия, программное обеспечение претерпело огромные изменения, пройдя путь от программ, которые способны выполнять лишь простейшие логические и арифметические операции, до сложных систем управления организациями. На сегодняшний день управление организацией без компьютера просто немыслимо. Для принятия какого-либо решения в условиях неопределенности и риска нужно регулярно держать под контролем разные аспекты финансово-хозяйственной деятельности. На современном этапе развития информатизации и автоматизации характерно применение распределенной обработки информации. В качестве более эффективной и перспективной сферы применения концепции распределенной обработки информации выступает автоматизация планово-управленческих функций на основе персональных компьютеров, которые установлены непосредственно на рабочем месте того или иного специалиста. Системы, которые выполняют данные функции, наделены широким распространением, которые называются автоматизированные рабочие места (АРМ).

Программа автоматизации работы автопарка автобусов представляет собой программу управления базой данных. Реализация базы данных возможна несколькими способами:

- использование существующих систем управления базами данных (СУБД);

- создание собственной структурированной базы данных.

К преимуществу СУБД можно отнести централизованное хранение всей информации о базе данных в самой базе данных (БД). В системе изначально созданы и реализованы алгоритмы обработки массивов данных без внешнего вмешательства. Все операции по изменению, сохранению информации выполняются самой СУБД, следовательно, нет необходимости программисту самому реализовывать методы сохранения и открытия файла данных. К недостаткам СУБД можно отнести избыточность хранимой информации в файле БД, что существенно увеличивает размер файла БД. А также необходимость наличия СУБД на компьютере пользователя, что существенно усложняет поставку программ конечному пользователю.

Цель данного курсового проекта - создание программы для автоматизации работы автопарка автобусов, удовлетворяющий перечисленным ранее требованиям, а также реализованный с использованием соответствующих СУБД и программного обеспечения.

**1. Постановка задачи.**

**1.1 Описание предметной области.**

Услуги по перевозкам пассажиров в пределах какого-то региона должен обеспечивать городской автовокзал. Для этого организация должна иметь у себя в штате водителей, кассиров, системных администраторов (если необходимо), бухгалтеров, юристов, механиков, кондукторов, офис менеджера и т.д. Также имеется парк автобусов. Основные клиенты автовокзала - лица, проживающие в данном регионе.

Автопарк - Территория с комплексом зданий, сооружений и устройств для стоянки, технического обслуживания и текущего ремонта подвижного состава грузового или пассажирского автомобильного транспорта

Любая транспортная компания, которая дорожит своей репутацией на рынке автоперевозок, содержит большой автопарк, и старается регулярно обновлять модельный ряд автомобилей, которые в нем представлены. От качества автомобилей напрямую зависит качество предоставляемых клиенту услуг. Автомобиль, который имеет технические проблемы, может отказать в самый неожиданный момент, что неизбежно приведет к финансовым убыткам. Автопарк любой компании обычно состоит из той техники, которая больше всего востребована на рынке. Можно сказать, что есть несколько значений управления автопарком для бизнеса. Остановимся на них подробнее.

1) Отслеживание местоположения транспортных средств компании.

2) Управление маршрутами

3) Управление топливом

4) Управление водителем

Учитывая все вышесказанное можно заключить, что компания, инвестирующая в систему управления транспортом, может обладать следующими преимуществами:

1. Снижение затрат. Благодаря многим функциям, упомянутым выше, таким как планирование, выполнение перевозок, логистика, отслеживание перевозок, отчеты и анализ — все это может привести к сокращению затрат.

2. Улучшенное обслуживание клиентов и подотчетность. Выбор подходящего перевозчика, а также улучшенное управление логистикой и отчетами могут повысить качество обслуживания клиентов и помочь компании быть более ответственной.

3. Улучшенный финансовый менеджмент. Улучшенное планирование маршрутов, оптимизация загрузки, лучшие выбранные перевозчики могут обеспечить лучшее управление финансами, включая сокращение расходов, упомянутых выше.

4. Гибкость. Системы управления транспортом обеспечивают гибкость при изменении планов доставки.

5. Простая установка и использование веб-системы управления транспортировкой. Большинство транспортных систем имеют простую установку и должны быть удобными для пользователя.

В результате использования системы управления автопарком компания получает обработанную информацию, которая может использоваться для организационных решений по управлению автопарком и управлению ресурсами компании.

**1.2 Актуальность решаемой задачи.**

Эффективная работа каждого предприятия зависит от оперативности и рациональности выполняемых действий. В настоящее время для хозяйственного субъекта очень важно быть конкурентно способным, особенно в сфере предоставления услуг. От качества предоставляемых услуг напрямую зависит имидж предприятия среди клиентов, а также авторитет среди конкурирующих организаций на рынке. Одной из самых значимых сфер оказания услуг является сфера предоставления транспортных услуг. Ежедневно в мире перевозятся миллионы тонн грузов и сотни тысяч пассажиров.

Разрабатываемая программа позволит упростить работу сотрудников автопарка автобусов а также учет информации о продажах билетов, техническом обслуживании автобусов, сотрудниках автопарка автобусов, маршрутах автобусов.

**1.3 Характеристика решаемой задачи.**

В результате выполнения курсового проекта требуется разработать программу, позволяющую оперативно учитывать следующую информацию: маршруты автобусов; обеспечение учета информации о продажах билетов; состояние технического обслуживания автобусов; список сотрудников автопарка автобусов.

В программе также присутствует функционал создания и экспорта отчетов. В своей основе программа имеет базу данных Microsoft Sql Server.

Список функций:

1. Просмотр отчетов.
2. Экспорт простых и сложных отчетов.
3. Просмотр и занесение информации о продажах билетов.
4. Просмотр и занесение информации о техническом обслуживании автобусов.
5. Просмотр и занесение информации о сотрудниках автопарка автобусов.
6. Просмотр и занесение информации о маршрутах автобусов.

Значительным преимуществом автоматизации данных операций является скорость их выполнения, возможность в любой момент осуществить запрос к данным, находящимся в базе и получить по этому запросу исчерпывающую информацию. Программа позволяет отказаться от необходимости ведения бумажных архивов.

**2. Проектирование программного модуля.**

**2.1 Разработка модели данных.**

Диаграммы UML приведены в рисунках ниже.

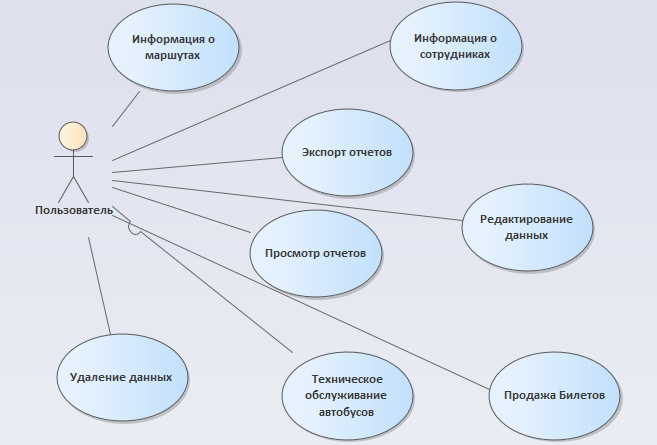


Рисунок 1. Диаграмма вариантов использования.

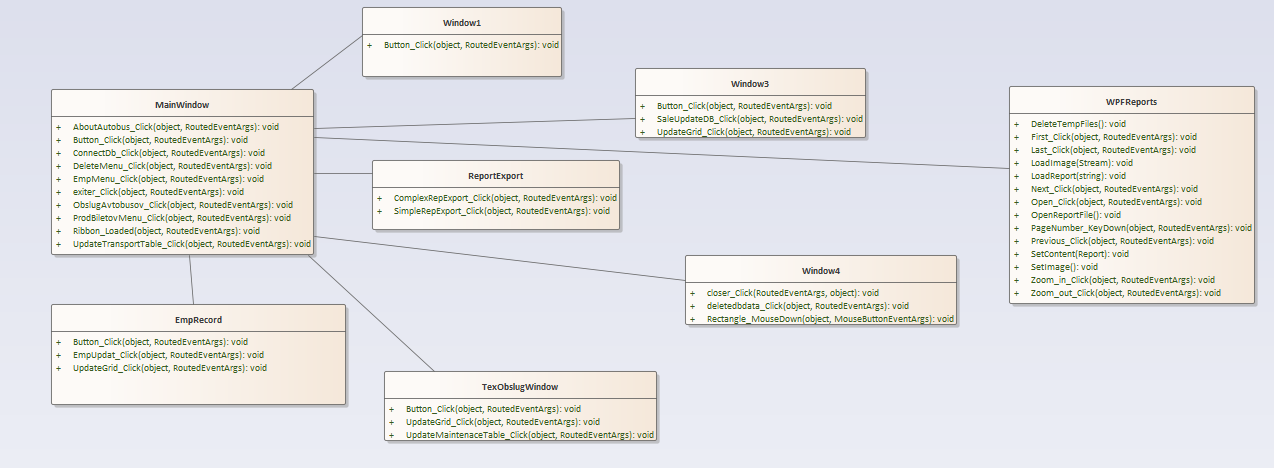


Рисунок 2. Диаграмма классов.

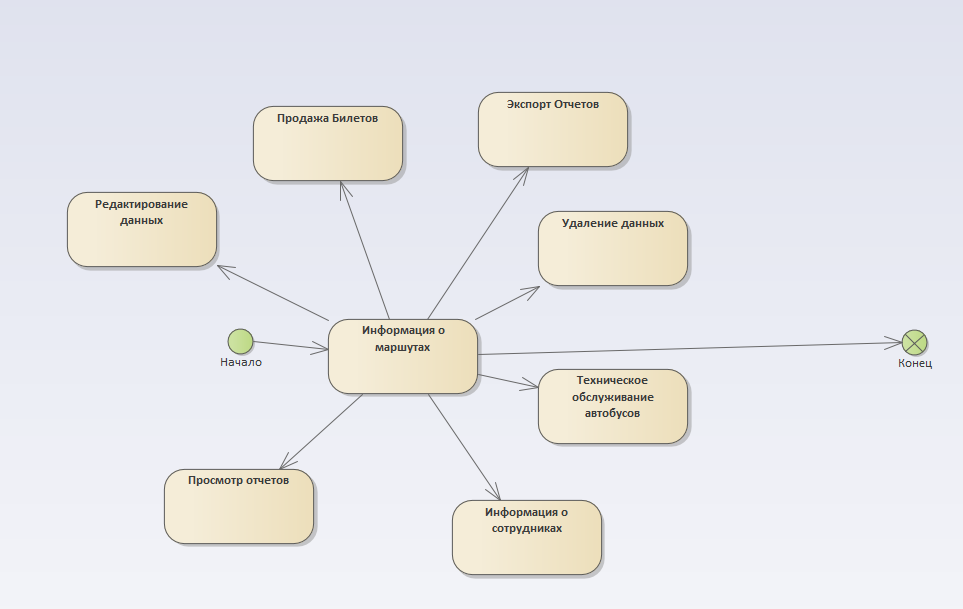


Рисунок 3. Диаграмма состояния.

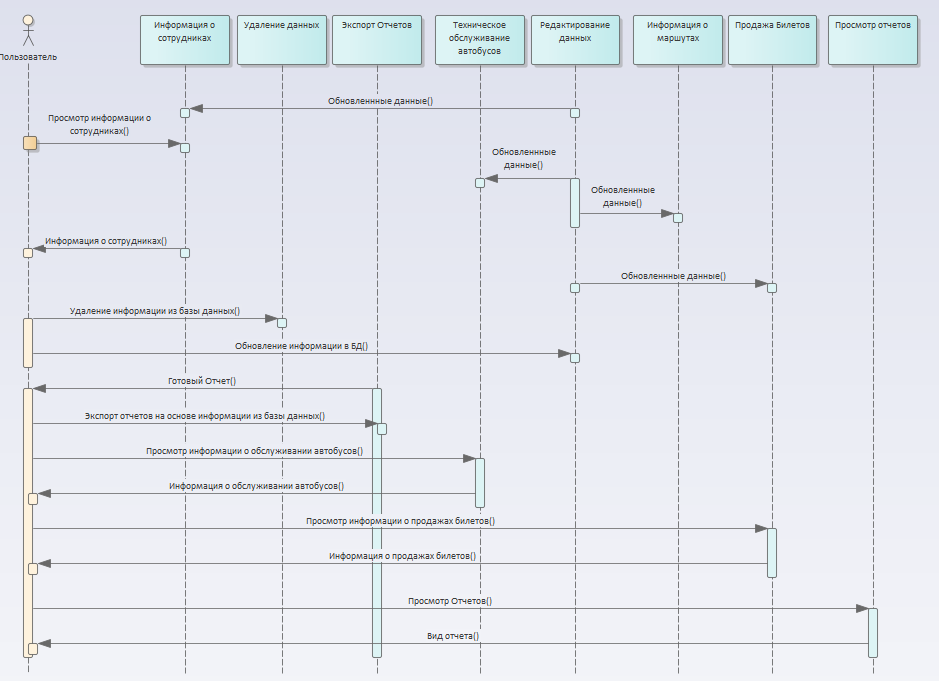


Рисунок 4. Диаграмма последовательности для всей программы.

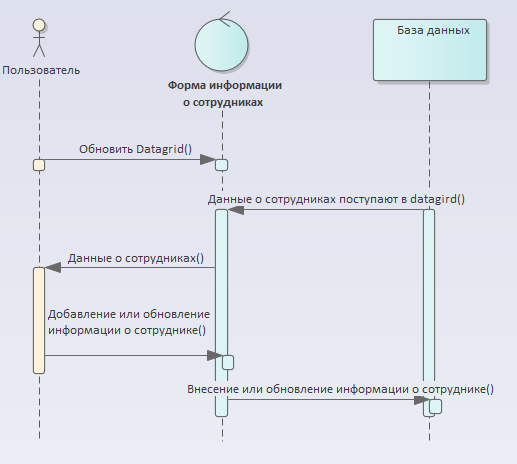


Рисунок 5. Диаграмма последовательности для формы информации о сотрудниках.

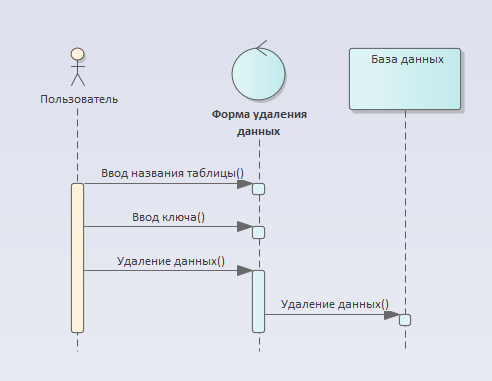


Рисунок 6. Диаграмма последовательности для формы удаления данных.

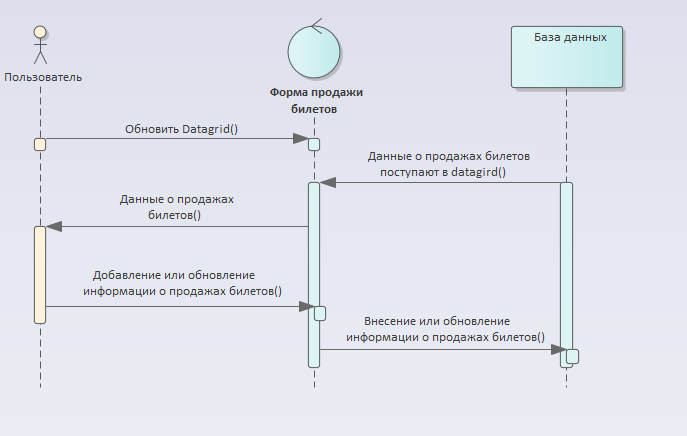


Рисунок 7. Диаграмма последовательности для формы продажи билетов.

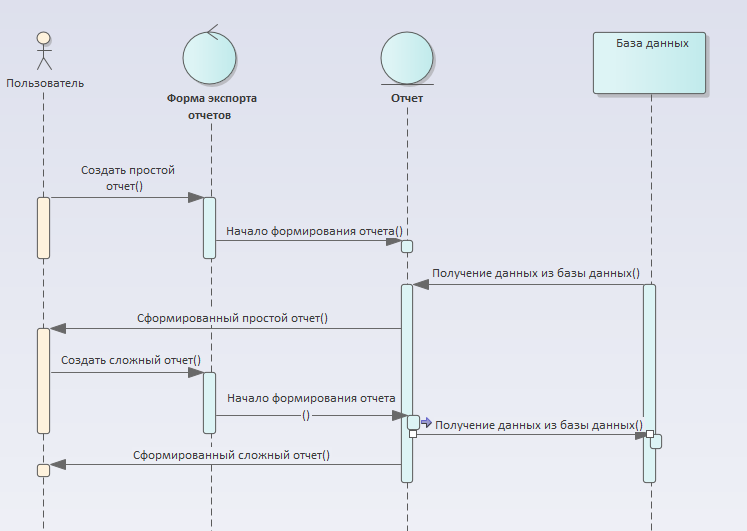


Рисунок 8. Диаграмма последовательности для формы экспорта отчетов.

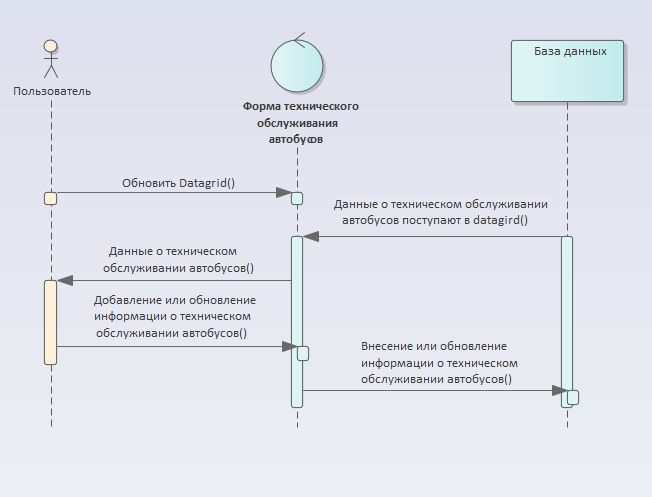


Рисунок 9. Диаграмма последовательности для формы технического обслуживания автобусов.

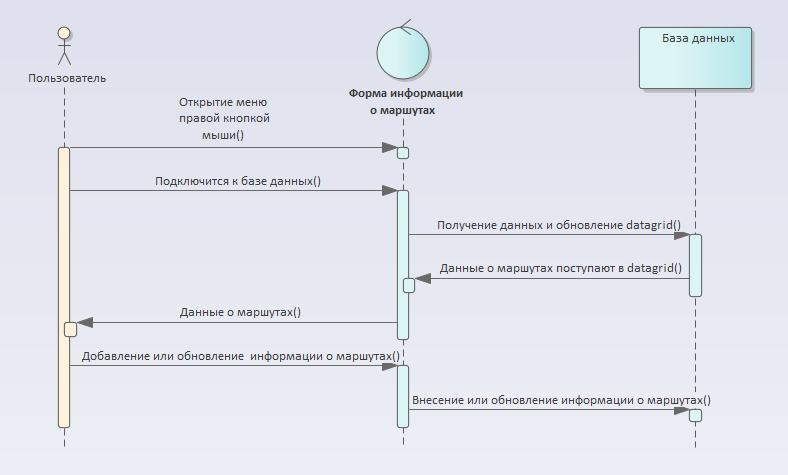


Рисунок 10. Диаграмма последовательности для формы информации о маршрутах.

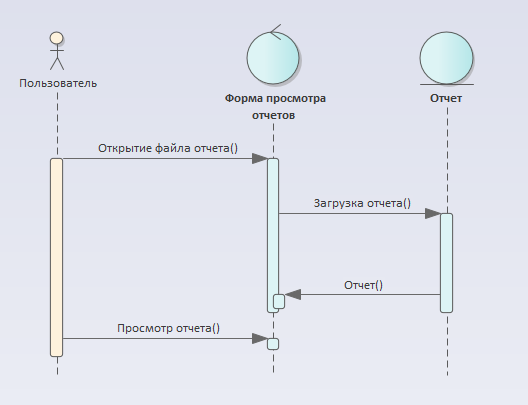


Рисунок 11. Диаграмма последовательности для формы просмотра отчетов.

Таблица 1. Описание процесса выполнения варианта использования.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Пользователь | Система | Экранная форма | Условие:  последующий шаг |
| 1. | Открывает программу | Программа загружает форму просмотра маршрутов | Информация о маршрутах. | Выбор одного из пунктов меню |
| 2. | Выбор пункта меню подключится к базе данных | Обновление DataGrid | Информация о маршрутах. | Выбор одного из пунктов меню или ввод информации в Textbox для добавления или обновления данных |
| 3. | Ввод информации и нажатие кнопки отправить данные в БД | Добавление введенной информации в таблицу Marshuti в базе данных после проверки на ошибки | Информация о маршрутах. | Выбор одного из пунктов меню |
| 4. | Ввод информации и нажатие кнопки обновить данные о маршрутах | Обновление данных в столбце с введенным ключом | Информация о маршрутах. | Выбор одного из пунктов меню |
| 5. | Выбор пункта меню продажа билетов | Открытие формы продажи билетов | Информация о маршрутах. | Выполнение ввода информации в форме продаже билетов и выбор одного из действий в форме или выбор одного из пунктов меню |
| 6. | Нажатие кнопки обновить DataGrid | Обновление DataGrid | Продажа билетов | Ввод информации в Textbox для добавления или обновления данных или выбор одного из пунктов меню в форме информации о маршрутах |
| 7. | Ввод информации и нажатие кнопки отправить данные в БД | Добавление введенной информации в таблицу Prodazhi в базе данных после проверки на ошибки | Продажа билетов | Выбор одного из пунктов меню в форме информации о маршрутах |
| 8. | Ввод информации и нажатие кнопки обновить данные о продажах | Обновление данных в столбце с введенным ключом в таблице Prodazhi в базе данных | Продажа билетов | Выбор одного из пунктов меню в форме информации о маршрутах |
| 9. | Выбор пункта меню удаление данных | Открытие формы удаления данных | Информация о маршрутах. | Выполнение ввода информации в форме удаления данных или выбор одного из пунктов меню |
| 10. | Ввод наименования таблицы и ключа столбца и нажатие кнопки удалить из таблицы | Удаление данных из введенной таблицы | Удаление данных | Выбор одного из пунктов меню в форме информации о маршрутах |
| 11. | Выбор пункта меню техническое обслуживание автобусов | Открытие формы технического обслуживания автобусов | Информация о маршрутах. | Выполнение ввода информации в форме технического обслуживания автобусов и выбор одного из действий в форме или выбор одного из пунктов меню |
| 12. | Нажатие кнопки обновить DataGrid | Обновление DataGrid | Техническое обслуживание автобусов | Ввод информации в Textbox для добавления или обновления данных или выбор одного из пунктов меню в форме информации о маршрутах |
| 13. | Ввод информации и нажатие кнопки отправить данные в БД | Добавление введенной информации в таблицу Maintenance в базе данных после проверки на ошибки | Техническое обслуживание автобусов | Выбор одного из пунктов меню в форме информации о маршрутах |
| 14. | Ввод информации и нажатие кнопки обновить данные о обслуживании автобусов | Обновление данных в столбце с введенным ключом в таблице Maintenance в базе данных | Техническое обслуживание автобусов | Выбор одного из пунктов меню в форме информации о маршрутах |
| 15. | Выбор пункта меню информация о сотрудниках | Открытие формы информации о сотрудниках | Информация о маршрутах. | Выполнение ввода информации в форме информации о сотрудниках и выбор одного из действий в форме или выбор одного из пунктов меню |
| 16. | Нажатие кнопки обновить DataGrid | Обновление DataGrid | Информация о сотрудниках | Ввод информации в Textbox для добавления или обновления данных или выбор одного из пунктов меню в форме информации о маршрутах |
| 17. | Ввод информации и нажатие кнопки отправить данные в БД | Добавление введенной информации в таблицу Employees в базе данных после проверки на ошибки | Информация о сотрудниках | Выбор одного из пунктов меню в форме информации о маршрутах |
| 18. | Ввод информации и нажатие кнопки обновить данные о обслуживании автобусов | Обновление данных в столбце с введенным ключом в таблице Employees в базе данных | Информация о сотрудниках | Выбор одного из пунктов меню в форме информации о маршрутах |
| 19. | Выбор пункта меню просмотр отчетов | Запрос пути к отдельно поставляемой программе просмотра отчетов и ее открытие | Информация о маршрутах. | Открытие и просмотр файла отчетов в программе просмотра отчетов либо выбор одного из пунктов меню в форме информации о маршрутах |
| 20. | Нажатие кнопки открытия файла | Открытие диалога выбора файла и загрузка файла отчета | Просмотр отчетов | Выбор одного из пунктов меню в форме информации о маршрутах |
| 21. | Нажатие кнопки First | Переход на первую страницу отчета | Просмотр отчетов | Выбор одного из пунктов меню в форме информации о маршрутах |
| 22. | Нажатие кнопки назад | Переход на предыдущую страницу отчета | Просмотр отчетов | Выбор одного из пунктов меню в форме информации о маршрутах |
| 23. | Нажатие кнопки вперед | Переход на следующую страницу отчета | Просмотр отчетов | Выбор одного из пунктов меню в форме информации о маршрутах |
| 24. | Нажатие кнопки Last | Переход на последнюю страницу отчета | Просмотр отчетов | Выбор одного из пунктов меню в форме информации о маршрутах |
| 25. | Нажатие кнопки + | Увеличение масштаба | Просмотр отчетов | Выбор одного из пунктов меню в форме информации о маршрутах |
| 26. | Нажатие кнопки - | Уменьшение масштаба | Просмотр отчетов | Выбор одного из пунктов меню в форме информации о маршрутах |
| 27. | Выбор пункта меню экспорт отчетов | Открытие формы экспорта отчетов | Информация о маршрутах. | Выбор одного из действий в форме экспорта отчетов или выбор одного из пунктов меню в форме информации о маршрутах |
| 28. | Нажатие кнопки экспортировать простой отчет | Подготовка и экспорт простого отчета в форматах .pdf и .frx | Экспорт отчетов | Выбор одного из действий в форме экспорта отчетов или выбор одного из пунктов меню в форме информации о маршрутах |
| 29. | Нажатие кнопки экспортировать сложный отчет | Подготовка и экспорт сложного отчета в форматах .pdf и .frx | Экспорт отчетов | Выбор одного из пунктов меню в форме информации о маршрутах |
| 30. | Выбор пункта меню о программе | Открытие формы о программе | Информация о маршрутах. | Закрытие программы |

**2.2 Выбор программного обеспечения.**

Интегрированная среда разработки (IDE) - это многофункциональная программа, которая поддерживает многие аспекты разработки программного обеспечения. Интегрированная среда разработки Visual Studio - это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Помимо стандартного редактора и отладчика, которые есть в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для улучшения процесса разработки. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

Система Microsoft Visual Studio является самой сложной интегрированной средой разработки (integrated development enviroment - IDE), доступной для программистов в настоящее время. Она является результатом долгой истории развития языков программирования и интерфейсов и вобрала в себя достижения многих сред разработки программного обеспечения.

**Microsoft SQL Server** — система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

**2.3. Определение требований к техническим средствам.**

Программа для автоматизации работы автопарка автобусов функционирует на компьютерах с определенным аппаратным и программным обеспечением.

Минимальные требования для корректной работы системы:

* центральный процессор - 2 ГГц
* оперативная память - 2 Гб.
* свободное место на HDD - не менее 5 Гб.
* мышь.
* клавиатура.
* монитор 14 и выше дюймов.
* операционная система Windows 7

**3. Реализация программного модуля.**

**3.1 Определение формы представления входных и выходных данных.**

**В программу из таблицы** ««**Employees**» базы данных Microsoft SQL server входят данные представленные в таблице 1.

Таблица 1. Данные таблицы **Employees.**

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Пояснение | Тип данных | Ключ поля |
| **Num** | Код сотрудника | **bigint** | Первичный |
| **Surname** | **Фамилия сотрудника** | **ntext** |  |
| **Name** | **Имя сотрудника** | **ntext** |  |
| **Patronym** | **Отчество сотрудника** | **ntext** |  |
| **Employed\_Since** | **Работает с** | **ntext** |  |
| **Job** | **Должность** | **ntext** |  |
| **Internship** | **Стажировка** | **ntext** |  |

Из таблицы «**Maintenance**» в программу входят данные представленные в таблице 2.

Таблица 2. Данные таблицы **Maintenance.**

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Пояснение | Тип данных | Ключ поля |
| **Num** | Код | **bigint** | Первичный |
| **Model\_Avtobusa** | **Модель автобуса** | **ntext** |  |
| **Data\_Poslednego\_Obsluzhivania** | **Дата последнего обслуживания** | **ntext** |  |
| **Ingener\_Obsluzhivania** | **Инженер, проводивший обслуживание** | **ntext** |  |
| **Problemi\_Avtobusa** | **Найденные проблемы автобуса** | **ntext** |  |
| **Data\_Sledueschego\_Obsluzivania** | **Дата следующего обслуживания** | **ntext** |  |
| **Goden\_K\_Doroge** | **Годность автобуса к дороге** | **ntext** |  |

Из таблицы «**Marshuti**» в программу входят данные представленные в таблице 3.

Таблица 3. Данные таблицы **Marshuti.**

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Пояснение | Тип данных | Ключ поля |
| **Nomer\_Marshuta** | Код маршрута | **bigint** | Первичный |
| **Nachalni\_Punkt** | **Начальный пункт** | **ntext** |  |
| **Konechni\_Punkt** | **Конечный пункт** | **ntext** |  |
| **Voditel** | **Водитель автобуса назначенного на маршрут** | **ntext** |  |
| **Model\_Avtobusa** | **Модель автобуса** | **ntext** |  |
| **Vremya\_Proezda** | **Время проезда маршрута** | **ntext** |  |

Из таблицы «Prodazhi» в программу входят данные представленные в таблице 4.

Таблица 4.Данные таблицы Prodazhi.

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Пояснение | Тип данных | Ключ поля |
| **Num** | Код | **bigint** | Первичный |
| **Sale\_Date** | **Дата продажи** | **ntext** |  |
| **Nomer\_Avtobusa** | **Номер автобуса** | **ntext** |  |
| **Punkt\_Posadki** | **Пункт посадки** | **ntext** |  |
| **Stoimost** | **Стоимость билета** | **ntext** |  |

Данные, которые отправляются в базу данных из программы представлены в таблице 5.

Таблица 5. Данные, которые отправляются в базу данных из программы.

|  |  |  |  |
| --- | --- | --- | --- |
| Выход | Имя Таблицы | Имя Поля Таблицы | Тип Поля |
| Выход в | **Employees** | **Num** | **bigint** |
| Выход в | **Employees** | **Surname** | **ntext** |
| Выход в | **Employees** | **Name** | **ntext** |
| Выход в | **Employees** | **Patronym** | **ntext** |
| Выход в | **Employees** | **Employed\_Since** | **ntext** |
| Выход в | **Employees** | **Job** | **ntext** |
| Выход в | **Employees** | **Internship** | **ntext** |
| Выход в | **Maintenance** | **Num** | **bigint** |
| Выход в | **Maintenance** | **Model\_Avtobusa** | **ntext** |
| Выход в | **Maintenance** | **Data\_Poslednego\_Obsluzhivania** | **ntext** |
| Выход в | **Maintenance** | **Ingener\_Obsluzhivania** | **ntext** |
| Выход в | **Maintenance** | **Problemi\_Avtobusa** | **ntext** |
| Выход в | **Maintenance** | **Data\_Sledueschego\_Obsluzivania** | **ntext** |
| Выход в | **Maintenance** | **Goden\_K\_Doroge** | **ntext** |
| Выход в | **Marshuti** | **Nomer\_Marshuta** | **bigint** |
| Выход в | **Marshuti** | **Nachalni\_Punkt** | **ntext** |
| Выход в | **Marshuti** | **Konechni\_Punkt** | **ntext** |
| Выход в | **Marshuti** | **Voditel** | **ntext** |
| Выход в | **Marshuti** | **Model\_Avtobusa** | **ntext** |
| Выход в | **Marshuti** | **Vremya\_Proezda** | **ntext** |
| Выход в | Prodazhi | **Num** | **bigint** |
| Выход в | Prodazhi | **Sale\_Date** | **ntext** |
| Выход в | Prodazhi | **Nomer\_Avtobusa** | **ntext** |
| Выход в | Prodazhi | **Punkt\_Posadki** | **ntext** |
| Выход в | Prodazhi | **Stoimost** | **ntext** |

При открытии программы вам предстает форма «Информация о маршрутах»

Форма «Информация о маршрутах» cлужит для просмотра информации о существующих маршрутах, а также для вызова остального функционала программы такого как: о программе, продажа билетов, подключение к базе данных, удаление данных, техническое обслуживание автобусов, информация о сотрудниках, просмотр отчетов, экспорт отчетов.

Форма «Информация о маршрутах» содержит следующие компоненты:

* Label -служит для показа названия полей базы данных;
* Textbox – служит для ввода данных добавляемых в базу данных или обновления существующих данных;
* Button – отправка новых данных в базу данных или обновление существующих;
* DataGrid – служит для показа информации находящийся в базе данных;
* TitleBarButton – служит для открытия меню по левому щелчку мыши.

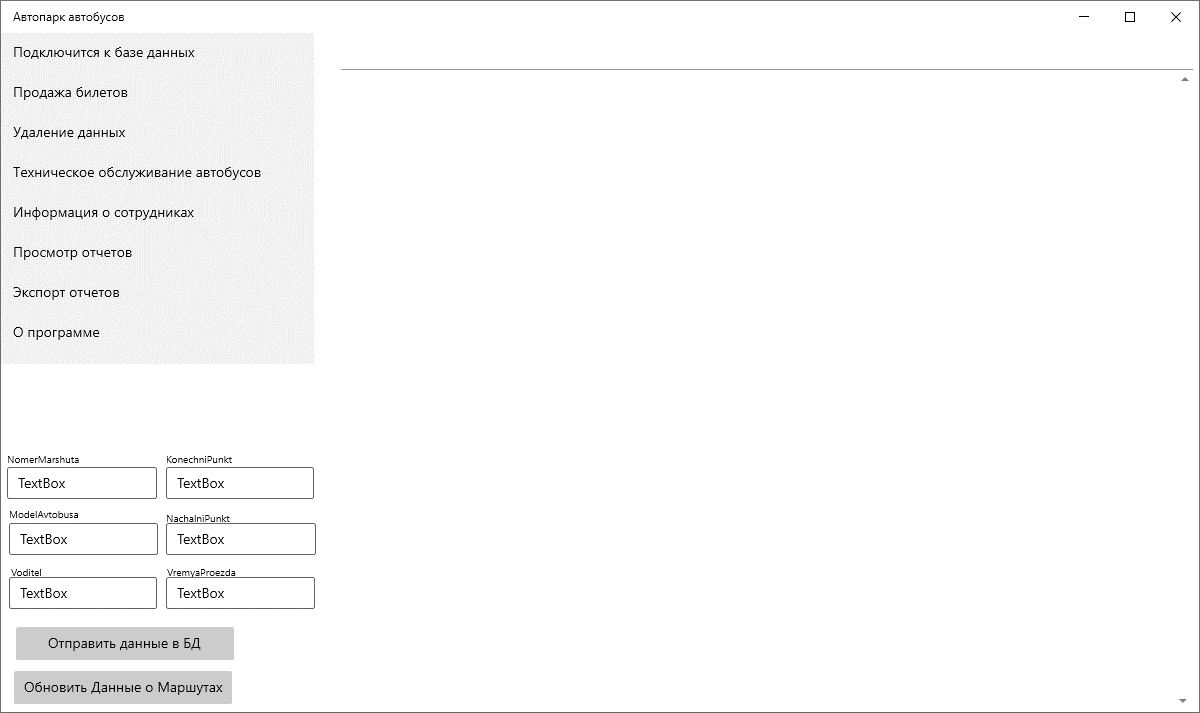


Рисунок 12. Форма «Информация о маршрутах»

Основное назначение формы «Продажа билетов» – просмотр данных о уже проданных билетах и занесение информации о новых проданных билетах

Форма «Продажа билетов» содержит следующие компоненты:

* Label -служит для показа названия полей базы данных;
* Textbox – служит для ввода данных добавляемых в базу данных или обновления существующих данных;
* Button – Обновление DataGrid или отправка новых данных в базу данных или обновление существующих;
* DataGrid – служит для показа информации находящийся в базе данных;

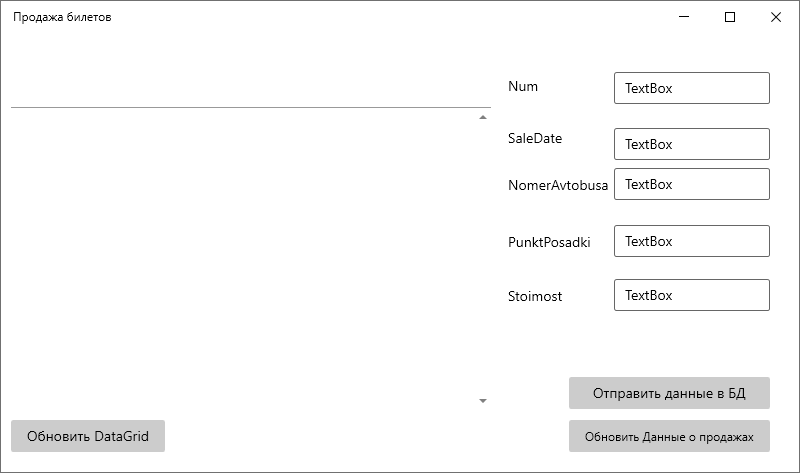


Рисунок 13. Форма «Продажа билетов».

Основное назначение формы «Удаление данных» - запрос на удаление определенной строки с введенным ключом из введенной таблицы

Форма «Удаление данных» содержит следующие компоненты:

* Label -служит для показа назначения TextBox;
* Textbox – служит для ввода таблицы и ключа для удаляемых данных;
* Button – Служит для отправки запроса на удаление данных;

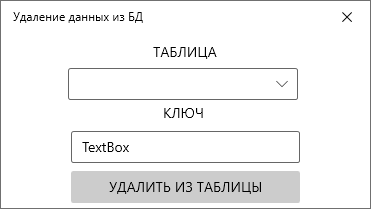


Рисунок 14. Форма «Удаление данных».

Основное назначение формы «Техническое обслуживание автобусов» - просмотр данных о состоянии технического обслуживания автобусов и занесение или обновление данных о техническом обслуживании автобусов

Форма «Техническое обслуживание автобусов» содержит следующие компоненты:

* Label -служит для показа названия полей базы данных;
* Textbox – служит для ввода данных добавляемых в базу данных или обновления существующих данных;
* Button – Обновление DataGrid или отправка новых данных в базу данных или обновление существующих;
* DataGrid – служит для показа информации находящийся в базе данных;

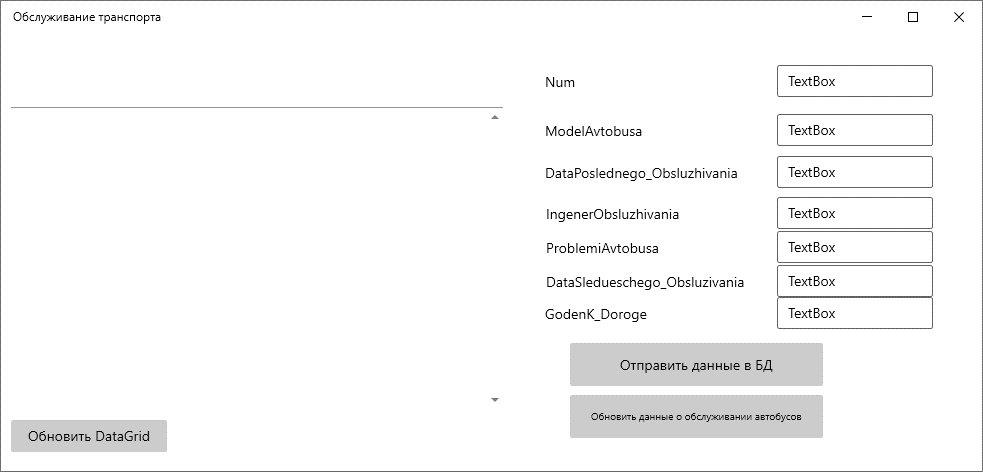


Рисунок 15. Форма «Техническое обслуживание автобусов».

Основное назначение формы «Информация о сотрудниках» - просмотр информации о сотрудниках автопарка автобусов и добавление или редактирование данных о сотрудниках.

Форма «Информация о сотрудниках» содержит следующие компоненты:

* Label -служит для показа названия полей базы данных;
* Textbox – служит для ввода данных добавляемых в базу данных или обновления существующих данных;
* Button – Обновление DataGrid или отправка новых данных в базу данных или обновление существующих;
* DataGrid – служит для показа информации находящийся в базе данных;

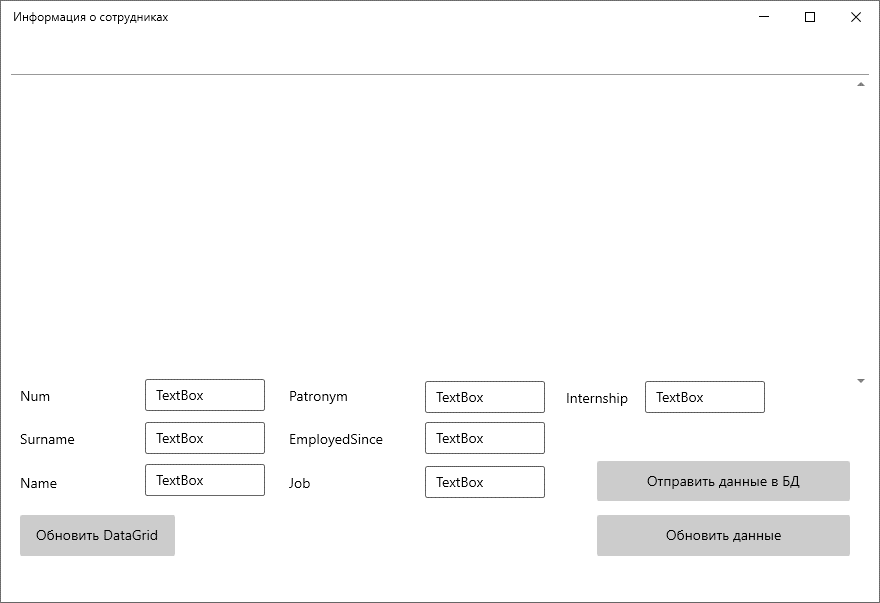


Рисунок 16. Форма «Информация о сотрудниках».

Основное назначение формы «Просмотр отчетов» - просмотр файлов отчетов в формате .frx.

Форма «Просмотр отчетов» содержит следующие компоненты:

* Button – служит для функционала открытия файлов, перехода на первую, следующую, предыдущую, последнюю страницы, а также для уменьшения и увеличения масштаба;
* ScrollViewer – служит для показа отчета.



Рисунок 17. Форма «Просмотр отчетов».

Основное назначение формы «Экспорт отчетов» - экспорта файлов сложного и простого отчета в формате .frx и формате .pdf.

Форма «Экспорт отчетов» содержит следующие компоненты:

Button – служит для запуска получения данных из базы данных и формирования файлов отчета (сложного или простого).

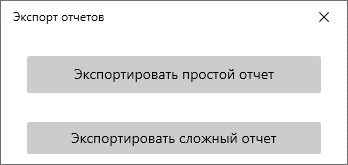


Рисунок 18. Форма «Экспорт отчетов».

**3.2 Тестирование программного модуля**

Тестирование проводилось с целью проверки корректности работы программы и её отладки.

Тестирование программы осуществлялось на персональном компьютере со следующими техническими характеристиками:

* процессор – AMD FX(tm)-6350 Six-Core Processor 3.90 GHz
* видеокарта – Nvidia Geforce GTX 750 TI
* оперативная память - 8,00 ГБ DDR3
* операционная система – Windows 10

Для обеспечения устойчивости программы и базы данных в каждой форме присутствует проверка на ошибки

Сообщение об ошибке имеет вид приведенный на рисунке 19

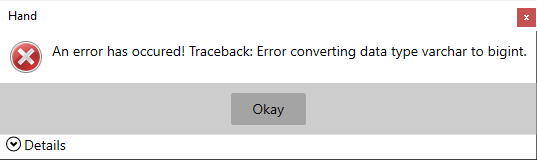


Рисунок 19 – Вид сообщения об ошибке

В результате тестирования установлено, что программа работает корректно, выполняет все заложенные в нее функции.

**3.3 Защита информации**

Защита информации - комплекс мероприятий, направленных на обеспечение важнейших аспектов информационной безопасности (целостности, доступности и, если нужно, конфиденциальности информации и ресурсов, используемых для ввода, хранения, обработки и передачи данных).

Система называется безопасной, если она, используя соответствующие аппаратные и программные средства, управляет доступом к информации так, что только должным образом авторизованные лица или же действующие от их имени процессы получают право читать, писать, создавать и удалять информацию.

Очевидно, что абсолютно безопасных систем нет, и здесь речь идет о надежной системе в смысле «система, которой можно доверять» (как можно доверять человеку). Система считается надежной, если она с использованием достаточных аппаратных и программных средств обеспечивает одновременную обработку информации разной степени секретности группой пользователей без нарушения прав доступа.

В программном модуле для автоматизации работы автопарка автобусов присутствует система входа для обеспечения различных прав доступа к базе данных

Вид формы приведен ниже на рисунке 20

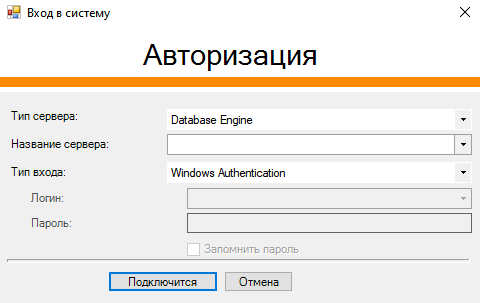


Рисунок 20 – Форма входа в систему

Вход в систему работает через логин и пароль от базы данных и позволяет при отсутствии у пользователя прав доступа к определенным таблицам выдать окно ошибки показанное на рисунке 19

Проверка целостности осуществляется с помощью встроенных методов базы данных

**3.4 Разработка справочной системы**

Разработанная справочная система для автоматизированной информационной системы «Автобусный автопарк» позволяет пользователю ознакомиться с краткой информацией о авторе программы и для чего она предназначена.

Вид справки «О программе» представлен на рисунке 21

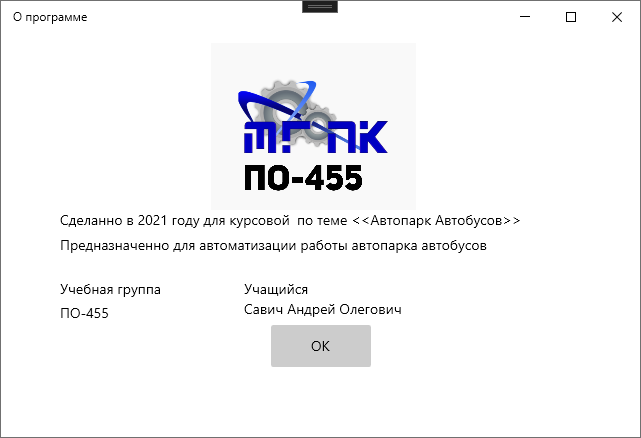


Рисунок 21 – Вид справки «О программе»

**4 Мероприятия по энерго- и ресурсосбережению**

В современном мире, в эпоху развивающейся промышленности необходимо принимать меры для сохранения окружающей нас среды. Поэтому в Республике Беларусь, как и в любой другой развитой стране, принята политика по ресурсо- и энергосбережению. Данная политика затрагивает все сферы общественной жизни, и конечно, не обошла и сферу образования. Стандарт управления энергопотреблением компьютеров, описывает три различных режима работы компьютера. Эти режимы отличаются потребляемой мощностью электроэнергии. Ждущий режим - позволяет экономить энергию, за счет отключения всех периферийных устройств, кроме ЦПУ, ОЗУ и материнской платы. Спящий режим - позволяет экономить электроэнергию на 100%, так как отключаются все периферийные устройства. Основной режим - при этом режиме экономии нет. Потребляемая мощность около 400Ватт.Для нахождения количества рабочих дней, в течение которых разрабатывался программный продукт, используем формулу:

n = Тпк /(8-tрп), (1)

где Тпк – время работы компьютера, ч;

Тпк =91,05 ч.

tрп – суммарное время регламентированных перерывов, в течение

рабочего дня, ч;

tрп =50 мин=5/6 часа.

дн



Для нахождения суммарной продолжительности регламентированных перерывов в течение всего времени разработки программного модуля:

Трп = n × tрп,(2)

где Трп – суммарная продолжительность регламентированных перерывов в течение всего времени разработки программного модуля;



Стоимость сэкономленной электроэнергии рассчитывается по формуле:

Сэн=Трп×(Wпк-Wсп)× Сэ, (3)

где Wпк – потребляемая мощность ПК, кВт;

Wсп – потребляемая мощность компьютера в ждущем режиме;

Сэ – стоимость 1 кВт электроэнергии, руб.

Wпк = 0,55 кВт;

Wсп = 0,35 кВт;

Сэ=0,31 руб/Кв;

Мероприятия по энерго и ресурсосбережению позволили сэкономить денежные средства в размере 62 копеек

### **Заключение**

В результате выполнения курсового проекта была разработана программа для автоматизации работы автопарка автобусов. Кроме этого были расширены знания из данной предметной области и укреплены знания языка программирования C# При реализации проекта были пройдены этапы описания и постановки задачи автоматизации работы автопарка автобусов, кодирования программы на алгоритмический язык и тестирования полученного приложения.

Программа позволяет:

* Просмотр отчетов.
* Экспорт простых и сложных отчетов.
* Просмотр и занесение информации о продажах билетов.
* Просмотр и занесение информации о техническом обслуживании автобусов.
* Просмотр и занесение информации о сотрудниках автопарка автобусов.
* Просмотр и занесение информации о маршрутах автобусов.

Программа обладает интуитивно-понятным интерфейсом, проста и удобна в использовании.

### **Список используемых источников**

1. ГОСТ ИСО/ МЭК 2382-99. Информационные технологии. Словарь. Ч. 1. Основные термины.
2. ГОСТ19.701-90 (ИСО 5807-85). Описание символов. Правила применения символов и выполнения схем.
3. СТУ 01-32-2019. Стандарт учреждения. Общие требования к оформлению текстовых документов».
4. Абрамян, М.А. Visual C# на примерах / М.А. Абрамян. - Москва: БХВ-Петербург, 2016.
5. Вагнер, Б. С# Эффективное программирование / Б. Вагнер. - Москва: ЛОРИ, 2017.
6. Дейтел, П. Как программировать на Visual C# 2012 / П. Дейтел. - Москва: Питер, 2016.
7. Зиборов, В.В. Visual C# 2012 на примерах / В.В. Зиборов. - Москва: БХВ-Петербург, 2015.
8. Ишкова, Э. А. Самоучитель С#. Начала программирования / Э.А. Ишкова.- Москва: Наука и техника, 2017.
9. Магда,Ю. С. NI Measurement Studio. Практика разработки систем измерения и управления на C# / Ю.С. Магда. - Москва: ДМК Пресс, 2017.
10. Прайс, Д.Visual C# 2.0. Полное руководство / Д.Прайс. - Москва: Век +, Корона-Век, Энтроп, 2015.
11. Рихтер,Д. CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C# / Д. Рихтер. - Москва: Питер, 2016.
12. Фленов, М.Е. Библия C# (+ CD-ROM) / М.Е. Фленов. - Москва: БХВ-Петербург, 2015.
13. Троелсен,Э.Язык программирования C#7 и платформы .NET и .NET Core/ Эндрю Троелсен,Филипп Джепикс – СПб.: ООО «Диалектика»,2018

.

# **Приложение А**

(обязательное)

**Текст программы**

using ModernWpf;

using MsgBoxEx;

using System;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Diagnostics;

using System.IO;

using System.Windows;

using System.Windows.Forms;

using SQLServerLoginTemplate;

using Application = System.Windows.Application;

namespace KursovayaAvtoparkAvtobusov

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public static string a;

public static string formations;

public MainWindow()

{

InitializeComponent();

var form1 = new FormConnectToSQLServer();

form1.ShowDialog();

a = form1.ConnectionString;

formations = a + ";Initial Catalog=KursovayaAvtoparkAvtobusov";

#if DEBUG

Console.WriteLine(a);

Console.WriteLine(formations);

#endif

}

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

{

try

{

SqlConnection cnn;

var conStr = MainWindow.formations;

cnn = new SqlConnection(conStr);

cnn.Open();

Console.Write("OPENING DB CONNECTION!!! \n");

Console.Write("[MENU] Connect db clicked - test \n");

var select = "SELECT \* FROM Marshuti";

var commandBuilder = new SqlCommand(select, cnn);

commandBuilder.ExecuteNonQuery();

var dataAdapter = new SqlDataAdapter(commandBuilder);

var ds = new DataTable("Avtobusx");

dataAdapter.Fill(ds);

dataGrid1.ItemsSource = ds.DefaultView;

Console.WriteLine("Connection established and the datagrid filled!");

cnn.Close();

}

catch (Exception ex)

{

MsgBoxExtendedFunctionality ext = new MsgBoxExtendedFunctionality()

{

DetailsText = ex.StackTrace

};

MessageBoxEx.ShowEx("An error has occured! Traceback: " + ex.Message, "Unexpected situation handling", MessageBoxButtonEx.OK, MessageBoxImage.Error, ext);

}

}

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

{

Console.WriteLine("[Menu] Opening about window!");

var wnxd = new Window2();

wnxd.Show();

}

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

{

var wndx = new Window3();

wndx.Show();

}

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

{

var wndxXx = new Window4();

wndxXx.Show();

}

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

{

var wndxXxx = new TexObslugWindow();

wndxXxx.Show();

}

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

{

var wndxXxxx = new EmpRecord();

wndxXxxx.Show();

}

private void Ribbon\_Loaded(object sender, RoutedEventArgs e)

{

if (ThemeManager.Current.ActualApplicationTheme == ApplicationTheme.Dark)

{

ThemeManager.Current.ApplicationTheme = ApplicationTheme.Dark;

ControlzEx.Theming.ThemeManager.Current.ChangeThemeBaseColor(Application.Current, "Dark");

}

else

{

ThemeManager.Current.ApplicationTheme = ApplicationTheme.Light;

ControlzEx.Theming.ThemeManager.Current.ChangeThemeBaseColor(Application.Current, "Light");

}

}

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

{

Application.Current.Shutdown();

}

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

{

try

{

SqlConnection cnn;

var conStr = MainWindow.formations;

cnn = new SqlConnection(conStr);

cnn.Open();

Console.Write("OPENING DB CONNECTION!!! \n");

var select =

"UPDATE Marshuti SET Nachalni\_Punkt=@a1,Konechni\_Punkt=@a2,Voditel=@a3,Model\_Avtobusa=@a4,Vremya\_Proezda=@a5 WHERE Nomer\_Marshuta=@a6";

var commandBuilder = new SqlCommand(select, cnn);

commandBuilder.Parameters.AddWithValue("a1", Editor2.Text);

commandBuilder.Parameters.AddWithValue("a2", Editor3.Text);

commandBuilder.Parameters.AddWithValue("a3", Editor4.Text);

commandBuilder.Parameters.AddWithValue("a4", Editor5.Text);

commandBuilder.Parameters.AddWithValue("a5", Editor6.Text);

commandBuilder.Parameters.AddWithValue("a6", int.Parse(Editor1.Text));

commandBuilder.ExecuteNonQuery();

Console.WriteLine("Connection established and the table updated!");

cnn.Close();

}

catch (Exception ex)

{

MsgBoxExtendedFunctionality ext = new MsgBoxExtendedFunctionality()

{

DetailsText = ex.StackTrace

};

MessageBoxEx.ShowEx("An error has occured! Traceback: " + ex.Message, "Unexpected situation handling", MessageBoxButtonEx.OK, MessageBoxImage.Error, ext);

}

}

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

{

try

{

var NomeMarshuta = Editor1.Text;

var NachalnPunkt = Editor2.Text;

var KonechnPunkt = Editor3.Text;

var Vodit = Editor4.Text;

var Modelvtobusa = Editor5.Text;

var VremProezda = Editor6.Text;

var query =

"INSERT INTO Marshuti(Nomer\_Marshuta, Nachalni\_Punkt, Konechni\_Punkt, Voditel,Model\_Avtobusa,Vremya\_Proezda) " +

"Values('" + NomeMarshuta + "', '" + NachalnPunkt + "', '" + KonechnPunkt + "', '" + Vodit + "','" +

Modelvtobusa + "', '" + VremProezda + "')";

SqlConnection cnn;

var conStr = MainWindow.formations;

cnn = new SqlConnection(conStr);

cnn.Open();

var command = new SqlCommand(query, cnn);

command.ExecuteNonQuery();

Console.WriteLine("Data inserted!\n Closing connection ");

cnn.Close();

Console.WriteLine("Connection has been closed , database ready for next operation!");

}

catch (Exception ex)

{

MsgBoxExtendedFunctionality ext = new MsgBoxExtendedFunctionality()

{

DetailsText = ex.StackTrace

};

MessageBoxEx.ShowEx("An error has occured! Traceback: " + ex.Message, "Unexpected situation handling", MessageBoxButtonEx.OK, MessageBoxImage.Error, ext);

}

}

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

{

var fbd = new FolderBrowserDialog();

string sSelectedPath = null;

fbd.Description = "Please locate your report viewer path";

if (fbd.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

sSelectedPath = fbd.SelectedPath;

}

else

{

return;

}

Process avReportLaunchProcess = new Process();

avReportLaunchProcess.StartInfo.FileName = Path.Combine(sSelectedPath, "WpfReporter.exe");

avReportLaunchProcess.Start();

}

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

{

var wndxXxxxX = new ReportExport();

wndxXxxxX.Show();

}

}

}

using FastReport;

using FastReport.Data;

using FastReport.Export.Image;

using FastReport.Export.PdfSimple;

using FastReport.Format;

using FastReport.Table;

using FastReport.Utils;

using MsgBoxEx;

using System;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.IO;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Windows;

namespace KursovayaAvtoparkAvtobusov

{

/// <summary>

/// Interaction logic for ReportExport.xaml

/// </summary>

public partial class ReportExport : Window

{

private static string outFolder = @"..\..\..\out\";

private static string inFolder = @"..\..\..\in\";

public ReportExport()

{

InitializeComponent();

}

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

{

RegisteredObjects.AddConnection(typeof(MsSqlDataConnection));

Report report = new Report();

MsSqlDataConnection connection = new MsSqlDataConnection();

connection.ConnectionString = ConfigurationManager.ConnectionStrings["KursovayaAvtoparkAvtobusov"].ToString();

connection.CreateAllTables();

report.Dictionary.Connections.Add(connection);

report.RegisterData(connection.DataSet, "mssql");

report.GetDataSource("Employees").Enabled = true;

report.GetDataSource("Maintenance").Enabled = true;

report.GetDataSource("Marshuti").Enabled = true;

report.GetDataSource("Prodazhi").Enabled = true;

ReportPage page = new ReportPage();

report.Pages.Add(page);

page.CreateUniqueName();

page.TopMargin = 10.0f;

page.LeftMargin = 10.0f;

page.RightMargin = 10.0f;

page.BottomMargin = 10.0f;

DataBand data = new DataBand();

data.CreateUniqueName();

page.AddChild(data);

data.DataSource = report.GetDataSource("Employees"); //adding a table to a bend

data.Height = Units.Centimeters \* 1.5f;

// Creating report title

page.ReportTitle = new ReportTitleBand();

page.ReportTitle.CreateUniqueName();

page.ReportTitle.Height = 4.0f \* Units.Centimeters;

TextObject titleText = new TextObject();

titleText.CreateUniqueName();

titleText.Left = 1.0f \* Units.Centimeters;

titleText.Top = 1.0f \* Units.Centimeters;

titleText.Width = 17.0f \* Units.Centimeters;

titleText.Height = 2.0f \* Units.Centimeters;

titleText.HorzAlign = HorzAlign.Center;

titleText.VertAlign = VertAlign.Center;

titleText.Font = new Font("Times New Roman", 32.0f);

titleText.TextColor = Color.Black;

titleText.FillColor = Color.Wheat;

titleText.Border.Color = Color.Black;

titleText.Border.Lines = BorderLines.All;

titleText.Border.Width = 4.0f;

titleText.Text = "Avtopark Report";

page.ReportTitle.Objects.Add(titleText);

// Header object

TextObject headerText = new TextObject();

headerText.CreateUniqueName();

headerText.Bounds = new RectangleF(0.0f, 2.7f \* Units.Centimeters,

19.0f \* Units.Centimeters, 0.8f \* Units.Centimeters);

headerText.HorzAlign = HorzAlign.Center;

headerText.VertAlign = VertAlign.Center;

headerText.Font = new Font("Times New Roman", 16.0f);

headerText.TextColor = Color.Black;

headerText.FillColor = Color.Wheat;

headerText.Border.Color = Color.Black;

headerText.Border.Lines = BorderLines.All;

headerText.Border.Width = 4.0f;

headerText.Text = "Employees" + " Table";

page.ReportTitle.Objects.Add(headerText);

// Employee Table Text bands

TextObject textx = new TextObject();

textx.Parent = data;

textx.CreateUniqueName();

textx.Bounds = new RectangleF(0 \* Units.Centimeters, 0, 1.0f \* Units.Centimeters, Units.Centimeters \* 0.5f);

textx.Font = new Font("Times New Roman", 5.0f);

textx.Text = "Num"; //table and its field

textx.Border.Lines = BorderLines.All;

TextObject textxx = new TextObject();

textxx.Parent = data;

textxx.CreateUniqueName();

textxx.Bounds = new RectangleF(1 \* Units.Centimeters, 0, Units.Centimeters \* 3, Units.Centimeters \* 0.5f);

textxx.Font = new Font("Times New Roman", 5.0f);

textxx.Text = "Surname"; //table and its field

textxx.Border.Lines = BorderLines.All;

TextObject textxxx = new TextObject();

textxxx.Parent = data;

textxxx.CreateUniqueName();

textxxx.Bounds = new RectangleF(4.0f \* Units.Centimeters, 0, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

textxxx.Font = new Font("Times New Roman", 5.0f);

textxxx.Text = "Name"; //table and its field

textxxx.Border.Lines = BorderLines.All;

TextObject textxxxx = new TextObject();

textxxxx.Parent = data;

textxxxx.CreateUniqueName();

textxxxx.Bounds = new RectangleF(7.4f \* Units.Centimeters, 0, Units.Centimeters \* 25, Units.Centimeters \* 0.5f);

textxxxx.Font = new Font("Times New Roman", 5.0f);

textxxxx.Text = "Patronym"; //table and its field

textxxxx.Border.Lines = BorderLines.All;

TextObject te = new TextObject();

te.Parent = data;

te.CreateUniqueName();

te.Bounds = new RectangleF(9.9f \* Units.Centimeters, 0, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

te.Font = new Font("Times New Roman", 5.0f);

te.Text = "Employed\_Since"; //table and its field

te.Border.Lines = BorderLines.All;

TextObject ter = new TextObject();

ter.Parent = data;

ter.CreateUniqueName();

ter.Bounds = new RectangleF(14 \* Units.Centimeters, 0, Units.Centimeters \* 30, Units.Centimeters \* 0.5f);

ter.Font = new Font("Times New Roman", 5.0f);

ter.Text = "Job"; //table and its field

ter.Border.Lines = BorderLines.All;

TextObject teXR = new TextObject();

teXR.Parent = data;

teXR.CreateUniqueName();

teXR.Bounds = new RectangleF(17.5f \* Units.Centimeters, 0, Units.Centimeters \* 25, Units.Centimeters \* 0.5f);

teXR.Font = new Font("Times New Roman", 5.0f);

teXR.Text = "Internship"; //table and its field

teXR.Border.Lines = BorderLines.All;

// Employee Table Fields

TextObject bandText = new TextObject();

bandText.CreateUniqueName();

bandText.HorzAlign = HorzAlign.Center;

bandText.Bounds = new RectangleF(0 \* Units.Centimeters, 25,

1.0f \* Units.Centimeters, 0.5f \* Units.Centimeters);

bandText.Font = new Font("Times New Roman", 5.0f);

bandText.Border.Lines = BorderLines.All;

bandText.Text = "[Employees.Num]";

data.AddChild(bandText);

TextObject text1 = new TextObject();

text1.Parent = data;

text1.CreateUniqueName();

text1.Bounds = new RectangleF(1 \* Units.Centimeters, 25, Units.Centimeters \* 3, Units.Centimeters \* 0.5f);

text1.Font = new Font("Times New Roman", 5.0f);

text1.Text = "[Employees.Surname]"; //table and its field

text1.Border.Lines = BorderLines.All;

TextObject text2 = new TextObject();

text2.Parent = data;

text2.CreateUniqueName();

text2.Bounds = new RectangleF(4.0f \* Units.Centimeters, 25, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

text2.Font = new Font("Times New Roman", 5.0f);

text2.Text = "[Employees.Name]"; //table and its field

text2.Border.Lines = BorderLines.All;

TextObject text3 = new TextObject();

text3.Parent = data;

text3.CreateUniqueName();

text3.Bounds = new RectangleF(7.4f \* Units.Centimeters, 25, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

text3.Font = new Font("Times New Roman", 5.0f);

text3.Text = "[Employees.Patronym]"; //table and its field

text3.Border.Lines = BorderLines.All;

TextObject text4 = new TextObject();

text4.Parent = data;

text4.CreateUniqueName();

text4.Bounds = new RectangleF(9.9f \* Units.Centimeters, 25, Units.Centimeters \* 30, Units.Centimeters \* 0.5f);

text4.Font = new Font("Times New Roman", 5.0f);

text4.Text = "[Employees.Employed\_Since]"; //table and its field

text4.Border.Lines = BorderLines.All;

TextObject text5 = new TextObject();

text5.Parent = data;

text5.CreateUniqueName();

text5.Bounds = new RectangleF(14 \* Units.Centimeters, 25, Units.Centimeters \* 5, Units.Centimeters \* 0.5f);

text5.Font = new Font("Times New Roman", 5.0f);

text5.Text = "[Employees.Job]"; //table and its field

text5.Border.Lines = BorderLines.All;

TextObject text6 = new TextObject();

text6.Parent = data;

text6.CreateUniqueName();

text6.Bounds = new RectangleF(17.5f \* Units.Centimeters, 25, Units.Centimeters \* 5, Units.Centimeters \* 0.5f);

text6.Font = new Font("Times New Roman", 5.0f);

text6.Text = "[Employees.Internship]"; //table and its field

text6.Border.Lines = BorderLines.All;

// Maintenance table things\

DataBand datax = new DataBand();

datax.CreateUniqueName();

page.AddChild(datax);

datax.DataSource = report.GetDataSource("Maintenance"); //adding a table to a bend

datax.Height = Units.Centimeters \* 5f;

//Maintenance Header object

TextObject headerSText = new TextObject();

headerSText.CreateUniqueName();

headerSText.Bounds = new RectangleF(0.0f, 7.1f \* Units.Centimeters,

19.0f \* Units.Centimeters, 0.8f \* Units.Centimeters);

headerSText.HorzAlign = HorzAlign.Center;

headerSText.VertAlign = VertAlign.Center;

headerSText.Font = new Font("Times New Roman", 16.0f);

headerSText.TextColor = Color.Black;

headerSText.FillColor = Color.Wheat;

headerSText.Border.Color = Color.Black;

headerSText.Border.Lines = BorderLines.All;

headerSText.Border.Width = 4.0f;

headerSText.Text = "Maintenance" + " Table";

page.ReportTitle.Objects.Add(headerSText);

//Maintenance Table Text bands

TextObject textxxrx = new TextObject();

textxxrx.Parent = datax;

textxxrx.CreateUniqueName();

textxxrx.Bounds = new RectangleF(0 \* Units.Centimeters, 39.9f, 0.7f \* Units.Centimeters, Units.Centimeters \* 0.5f);

textxxrx.Font = new Font("Times New Roman", 6.0f);

textxxrx.Text = "Num"; //table and its field

textxxrx.Border.Lines = BorderLines.All;

TextObject textxxxr = new TextObject();

textxxxr.Parent = datax;

textxxxr.CreateUniqueName();

textxxxr.Font = new Font("Times New Roman", 6.0f);

textxxxr.Bounds = new RectangleF(0.7f \* Units.Centimeters, 39.9f, Units.Centimeters \* 7, Units.Centimeters \* 0.5f);

textxxxr.Text = "Model\_Avtobusa"; //table and its field

textxxxr.Border.Lines = BorderLines.All;

TextObject textxxxxr = new TextObject();

textxxxxr.Parent = datax;

textxxxxr.CreateUniqueName();

textxxxxr.Bounds = new RectangleF(4.5f \* Units.Centimeters, 39.9f, Units.Centimeters \* 15, Units.Centimeters \* 0.5f);

textxxxxr.Font = new Font("Times New Roman", 5.0f);

textxxxxr.Text = "Data\_Poslednego\_Obsluzhivania"; //table and its field

textxxxxr.Border.Lines = BorderLines.All;

TextObject textxxxxx = new TextObject();

textxxxxx.Parent = datax;

textxxxxx.Font = new Font("Times New Roman", 5.0f);

textxxxxx.CreateUniqueName();

textxxxxx.Bounds = new RectangleF(7.7f \* Units.Centimeters, 39.9f, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

textxxxxx.Text = "Ingener\_Obsluzhivania"; //table and its field

textxxxxx.Border.Lines = BorderLines.All;

TextObject terzx = new TextObject();

terzx.Parent = datax;

terzx.CreateUniqueName();

terzx.Bounds = new RectangleF(10.4f \* Units.Centimeters, 39.9f, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

terzx.Font = new Font("Times New Roman", 5.0f);

terzx.Text = "Problemi\_Avtobusa"; //table and its field

terzx.Border.Lines = BorderLines.All;

TextObject terxr = new TextObject();

terxr.Parent = datax;

terxr.CreateUniqueName();

terxr.Bounds = new RectangleF(15 \* Units.Centimeters, 39.9f, Units.Centimeters \* 15, Units.Centimeters \* 0.5f);

terxr.Font = new Font("Times New Roman", 4.0f);

terxr.Text = "Data\_Sledueschego\_Obsluzivania"; //table and its field

terxr.Border.Lines = BorderLines.All;

TextObject teXRr = new TextObject();

teXRr.Parent = datax;

teXRr.CreateUniqueName();

teXRr.Bounds = new RectangleF(17.5f \* Units.Centimeters, 39.9f, Units.Centimeters \* 26, Units.Centimeters \* 0.5f);

teXRr.Font = new Font("Times New Roman", 6.0f);

teXRr.Text = "Goden\_K\_Doroge"; //table and its field

teXRr.Border.Lines = BorderLines.All;

//Maintenance table fields

TextObject textxxrxx = new TextObject();

textxxrxx.Parent = datax;

textxxrxx.CreateUniqueName();

textxxrxx.Bounds = new RectangleF(0 \* Units.Centimeters, 60.9f, 0.7f \* Units.Centimeters, Units.Centimeters \* 0.5f);

textxxrxx.Font = new Font("Times New Roman", 6.0f);

textxxrxx.Text = "[Maintenance.Num]"; //table and its field

textxxrxx.Border.Lines = BorderLines.All;

TextObject textxxxrx = new TextObject();

textxxxrx.Parent = datax;

textxxxrx.CreateUniqueName();

textxxxrx.Font = new Font("Times New Roman", 6.0f);

textxxxrx.Bounds = new RectangleF(0.7f \* Units.Centimeters, 60.9f, Units.Centimeters \* 7, Units.Centimeters \* 0.5f);

textxxxrx.Text = "[Maintenance.Model\_Avtobusa]"; //table and its field

textxxxrx.Border.Lines = BorderLines.All;

TextObject textxxxxrx = new TextObject();

textxxxxrx.Parent = datax;

textxxxxrx.CreateUniqueName();

textxxxxrx.Bounds = new RectangleF(4.5f \* Units.Centimeters, 60.9f, Units.Centimeters \* 15, Units.Centimeters \* 0.5f);

textxxxxrx.Font = new Font("Times New Roman", 5.0f);

textxxxxrx.Text = "[Maintenance.Data\_Poslednego\_Obsluzhivania]"; //table and its field

textxxxxrx.Border.Lines = BorderLines.All;

TextObject textxxxxxx = new TextObject();

textxxxxxx.Parent = datax;

textxxxxxx.Font = new Font("Times New Roman", 5.0f);

textxxxxxx.CreateUniqueName();

textxxxxxx.Bounds = new RectangleF(7.7f \* Units.Centimeters, 60.9f, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

textxxxxxx.Text = "[Maintenance.Ingener\_Obsluzhivania]"; //table and its field

textxxxxxx.Border.Lines = BorderLines.All;

TextObject terzxx = new TextObject();

terzxx.Parent = datax;

terzxx.CreateUniqueName();

terzxx.Bounds = new RectangleF(10.4f \* Units.Centimeters, 60.9f, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

terzxx.Font = new Font("Times New Roman", 5.0f);

terzxx.Text = "[Maintenance.Problemi\_Avtobusa]"; //table and its field

terzxx.Border.Lines = BorderLines.All;

TextObject terxrx = new TextObject();

terxrx.Parent = datax;

terxrx.CreateUniqueName();

terxrx.Bounds = new RectangleF(15 \* Units.Centimeters, 60.9f, Units.Centimeters \* 15, Units.Centimeters \* 0.5f);

terxrx.Font = new Font("Times New Roman", 6.0f);

terxrx.Text = "[Maintenance.Data\_Sledueschego\_Obsluzivania]"; //table and its field

terxrx.Border.Lines = BorderLines.All;

TextObject teXRrr = new TextObject();

teXRrr.Parent = datax;

teXRrr.CreateUniqueName();

teXRrr.Bounds = new RectangleF(17.5f \* Units.Centimeters, 60.9f, Units.Centimeters \* 26, Units.Centimeters \* 0.5f);

teXRrr.Font = new Font("Times New Roman", 6.0f);

teXRrr.Text = "[Maintenance.Goden\_K\_Doroge]"; //table and its field

teXRrr.Border.Lines = BorderLines.All;

// Prodazhi Header

TextObject headersSText = new TextObject();

headersSText.CreateUniqueName();

headersSText.Bounds = new RectangleF(0.0f, 11.1f \* Units.Centimeters,

19.0f \* Units.Centimeters, 0.8f \* Units.Centimeters);

headersSText.HorzAlign = HorzAlign.Center;

headersSText.VertAlign = VertAlign.Center;

headersSText.Font = new Font("Times New Roman", 16.0f);

headersSText.TextColor = Color.Black;

headersSText.FillColor = Color.Wheat;

headersSText.Border.Color = Color.Black;

headersSText.Border.Lines = BorderLines.All;

headersSText.Border.Width = 4.0f;

headersSText.Text = "Prodazhi" + " Table";

page.ReportTitle.Objects.Add(headersSText);

// Prodazhi Table Info

DataBand dataxr = new DataBand();

dataxr.CreateUniqueName();

page.AddChild(dataxr);

dataxr.DataSource = report.GetDataSource("Prodazhi"); //adding a table to a bend

dataxr.Height = Units.Centimeters \* 5f;

TextObject texrxx = new TextObject();

texrxx.Parent = dataxr;

texrxx.CreateUniqueName();

texrxx.Bounds = new RectangleF(0 \* Units.Centimeters, 43.99f, 0.7f \* Units.Centimeters, Units.Centimeters \* 0.5f);

texrxx.Font = new Font("Times New Roman", 6.0f);

texrxx.Text = "Num"; //table and its field

texrxx.Border.Lines = BorderLines.All;

TextObject texrxs = new TextObject();

texrxs.Parent = dataxr;

texrxs.CreateUniqueName();

texrxs.Font = new Font("Times New Roman", 6.0f);

texrxs.Bounds = new RectangleF(0.7f \* Units.Centimeters, 43.99f, Units.Centimeters \* 7, Units.Centimeters \* 0.5f);

texrxs.Text = "Sale\_Date"; //table and its field

texrxs.Border.Lines = BorderLines.All;

TextObject terscw = new TextObject();

terscw.Parent = dataxr;

terscw.CreateUniqueName();

terscw.Bounds = new RectangleF(4.5f \* Units.Centimeters, 43.99f, Units.Centimeters \* 15, Units.Centimeters \* 0.5f);

terscw.Font = new Font("Times New Roman", 5.0f);

terscw.Text = "Nomer\_Avtobusa"; //table and its field

terscw.Border.Lines = BorderLines.All;

TextObject terscwx = new TextObject();

terscwx.Parent = dataxr;

terscwx.Font = new Font("Times New Roman", 5.0f);

terscwx.CreateUniqueName();

terscwx.Bounds = new RectangleF(7.7f \* Units.Centimeters, 43.99f, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

terscwx.Text = "Punkt\_Posadki"; //table and its field

terscwx.Border.Lines = BorderLines.All;

TextObject terzrx = new TextObject();

terzrx.Parent = dataxr;

terzrx.CreateUniqueName();

terzrx.Bounds = new RectangleF(10.4f \* Units.Centimeters, 43.99f, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

terzrx.Font = new Font("Times New Roman", 5.0f);

terzrx.Text = "Stoimost"; //table and its field

terzrx.Border.Lines = BorderLines.All;

// Table Fields For Prodazhi

TextObject texrxxx = new TextObject();

texrxxx.Parent = dataxr;

texrxxx.CreateUniqueName();

texrxxx.Bounds = new RectangleF(0 \* Units.Centimeters, 53.99f, 0.7f \* Units.Centimeters, Units.Centimeters \* 0.5f);

texrxxx.Font = new Font("Times New Roman", 6.0f);

texrxxx.Text = "[Prodazhi.Num]"; //table and its field

texrxxx.Border.Lines = BorderLines.All;

TextObject texrxss = new TextObject();

texrxss.Parent = dataxr;

texrxss.CreateUniqueName();

texrxss.Font = new Font("Times New Roman", 6.0f);

texrxss.Bounds = new RectangleF(0.7f \* Units.Centimeters, 53.99f, Units.Centimeters \* 7, Units.Centimeters \* 0.5f);

texrxss.Text = "[Prodazhi.Sale\_Date]"; //table and its field

texrxss.Border.Lines = BorderLines.All;

TextObject terscwxxxr = new TextObject();

terscwxxxr.Parent = dataxr;

terscwxxxr.CreateUniqueName();

terscwxxxr.Bounds = new RectangleF(4.5f \* Units.Centimeters, 53.99f, Units.Centimeters \* 15, Units.Centimeters \* 0.5f);

terscwxxxr.Font = new Font("Times New Roman", 5.0f);

terscwxxxr.Text = "[Prodazhi.Nomer\_Avtobusa]"; //table and its field

terscwxxxr.Border.Lines = BorderLines.All;

TextObject terscwxxr = new TextObject();

terscwxxr.Parent = dataxr;

terscwxxr.Font = new Font("Times New Roman", 5.0f);

terscwxxr.CreateUniqueName();

terscwxxr.Bounds = new RectangleF(7.7f \* Units.Centimeters, 53.99f, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

terscwxxr.Text = "[Prodazhi.Punkt\_Posadki]"; //table and its field

terscwxxr.Border.Lines = BorderLines.All;

TextObject terzrxr = new TextObject();

terzrxr.Parent = dataxr;

terzrxr.CreateUniqueName();

terzrxr.Bounds = new RectangleF(10.4f \* Units.Centimeters, 53.99f, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

terzrxr.Font = new Font("Times New Roman", 5.0f);

terzrxr.Text = "[Prodazhi.Stoimost]"; //table and its field

terzrxr.Border.Lines = BorderLines.All;

// Marshuti table header

TextObject headersrSText = new TextObject();

headersrSText.CreateUniqueName();

headersrSText.Bounds = new RectangleF(0.0f, 15.1f \* Units.Centimeters,

19.0f \* Units.Centimeters, 0.8f \* Units.Centimeters);

headersrSText.HorzAlign = HorzAlign.Center;

headersrSText.VertAlign = VertAlign.Center;

headersrSText.Font = new Font("Times New Roman", 16.0f);

headersrSText.TextColor = Color.Black;

headersrSText.FillColor = Color.Wheat;

headersrSText.Border.Color = Color.Black;

headersrSText.Border.Lines = BorderLines.All;

headersrSText.Border.Width = 4.0f;

headersrSText.Text = "Marshuti" + " Table";

page.ReportTitle.Objects.Add(headersrSText);

// Marshuti fields

DataBand datarxr = new DataBand();

datarxr.CreateUniqueName();

page.AddChild(datarxr);

datarxr.DataSource = report.GetDataSource("Marshuti"); //adding a table to a bend

datarxr.Height = Units.Centimeters \* 2f;

TextObject txtef = new TextObject();

txtef.Parent = datarxr;

txtef.CreateUniqueName();

txtef.Bounds = new RectangleF(0 \* Units.Centimeters, 58.9f, 0.7f \* Units.Centimeters, Units.Centimeters \* 0.5f);

txtef.Font = new Font("Times New Roman", 6.0f);

txtef.Text = "Nomer\_Marshuta"; //table and its field

txtef.Border.Lines = BorderLines.All;

TextObject tfeqa = new TextObject();

tfeqa.Parent = datarxr;

tfeqa.CreateUniqueName();

tfeqa.Font = new Font("Times New Roman", 6.0f);

tfeqa.Bounds = new RectangleF(0.7f \* Units.Centimeters, 58.9f, Units.Centimeters \* 7, Units.Centimeters \* 0.5f);

tfeqa.Text = "Nachalni\_Punkt"; //table and its field

tfeqa.Border.Lines = BorderLines.All;

TextObject tcqqcq = new TextObject();

tcqqcq.Parent = datarxr;

tcqqcq.CreateUniqueName();

tcqqcq.Bounds = new RectangleF(4.5f \* Units.Centimeters, 58.9f, Units.Centimeters \* 15, Units.Centimeters \* 0.5f);

tcqqcq.Font = new Font("Times New Roman", 5.0f);

tcqqcq.Text = "Konechni\_Punkt"; //table and its field

tcqqcq.Border.Lines = BorderLines.All;

TextObject txtweq = new TextObject();

txtweq.Parent = datarxr;

txtweq.Font = new Font("Times New Roman", 5.0f);

txtweq.CreateUniqueName();

txtweq.Bounds = new RectangleF(7.7f \* Units.Centimeters, 58.9f, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

txtweq.Text = "Voditel"; //table and its field

txtweq.Border.Lines = BorderLines.All;

TextObject terrzx = new TextObject();

terrzx.Parent = datarxr;

terrzx.CreateUniqueName();

terrzx.Bounds = new RectangleF(10.4f \* Units.Centimeters, 58.9f, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

terrzx.Font = new Font("Times New Roman", 5.0f);

terrzx.Text = "Model\_Avtobusa"; //table and its field

terrzx.Border.Lines = BorderLines.All;

TextObject txerxr = new TextObject();

txerxr.Parent = datarxr;

txerxr.CreateUniqueName();

txerxr.Bounds = new RectangleF(15 \* Units.Centimeters, 58.9f, Units.Centimeters \* 15, Units.Centimeters \* 0.5f);

txerxr.Font = new Font("Times New Roman", 4.0f);

txerxr.Text = "Vremya\_Proezda"; //table and its field

txerxr.Border.Lines = BorderLines.All;

TextObject txteff = new TextObject();

txteff.Parent = datarxr;

txteff.CreateUniqueName();

txteff.Bounds = new RectangleF(0 \* Units.Centimeters, 82.9f, 0.7f \* Units.Centimeters, Units.Centimeters \* 0.5f);

txteff.Font = new Font("Times New Roman", 6.0f);

txteff.Text = "[Marshuti.Nomer\_Marshuta]"; //table and its field

txteff.Border.Lines = BorderLines.All;

TextObject tfeqafd = new TextObject();

tfeqafd.Parent = datarxr;

tfeqafd.CreateUniqueName();

tfeqafd.Font = new Font("Times New Roman", 6.0f);

tfeqafd.Bounds = new RectangleF(0.7f \* Units.Centimeters, 82.9f, Units.Centimeters \* 7, Units.Centimeters \* 0.5f);

tfeqafd.Text = "[Marshuti.Nachalni\_Punkt]"; //table and its field

tfeqafd.Border.Lines = BorderLines.All;

TextObject tcqqcqd = new TextObject();

tcqqcqd.Parent = datarxr;

tcqqcqd.CreateUniqueName();

tcqqcqd.Bounds = new RectangleF(4.5f \* Units.Centimeters, 82.9f, Units.Centimeters \* 15, Units.Centimeters \* 0.5f);

tcqqcqd.Font = new Font("Times New Roman", 5.0f);

tcqqcqd.Text = "[Marshuti.Konechni\_Punkt]"; //table and its field

tcqqcqd.Border.Lines = BorderLines.All;

TextObject txtweqq = new TextObject();

txtweqq.Parent = datarxr;

txtweqq.Font = new Font("Times New Roman", 5.0f);

txtweqq.CreateUniqueName();

txtweqq.Bounds = new RectangleF(7.7f \* Units.Centimeters, 82.9f, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

txtweqq.Text = "[Marshuti.Voditel]"; //table and its field

txtweqq.Border.Lines = BorderLines.All;

TextObject terrzxr = new TextObject();

terrzxr.Parent = datarxr;

terrzxr.CreateUniqueName();

terrzxr.Bounds = new RectangleF(10.4f \* Units.Centimeters, 82.9f, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

terrzxr.Font = new Font("Times New Roman", 5.0f);

terrzxr.Text = "[Marshuti.Model\_Avtobusa]"; //table and its field

terrzxr.Border.Lines = BorderLines.All;

TextObject txerxxr = new TextObject();

txerxxr.Parent = datarxr;

txerxxr.CreateUniqueName();

txerxxr.Bounds = new RectangleF(15 \* Units.Centimeters, 82.9f, Units.Centimeters \* 15, Units.Centimeters \* 0.5f);

txerxxr.Font = new Font("Times New Roman", 4.0f);

txerxxr.Text = "[Marshuti.Vremya\_Proezda]"; //table and its field

txerxxr.Border.Lines = BorderLines.All;

// Footer

page.PageFooter = new PageFooterBand();

page.PageFooter.CreateUniqueName();

page.PageFooter.Height = 0.5f \* Units.Centimeters;

TextObject footerText = new TextObject();

footerText.CreateUniqueName();

footerText.HorzAlign = HorzAlign.Right;

footerText.VertAlign = VertAlign.Center;

footerText.Bounds = new RectangleF(0.0f, 0.0f,

19.0f \* Units.Centimeters, 0.5f \* Units.Centimeters);

footerText.TextColor = Color.Black;

footerText.FillColor = Color.Wheat;

titleText.Border.Color = Color.Black;

titleText.Border.Lines = BorderLines.All;

titleText.Border.Width = 4.0f;

footerText.Text = "Page [Page]";

page.PageFooter.Objects.Add(footerText);

try

{

// Exporting the report

report.Prepare();

report.SavePrepared("Prepared Report.fpx");

PDFSimpleExport pdf = new PDFSimpleExport();

// Save the report

report.Export(pdf, "ExportedPDF.pdf");

}

catch (Exception ex)

{

MsgBoxExtendedFunctionality ext = new MsgBoxExtendedFunctionality()

{

DetailsText = ex.StackTrace

};

MessageBoxEx.ShowEx("An error has occured! Traceback: " + ex.Message, "Unexpected situation handling", MessageBoxButtonEx.OK, MessageBoxImage.Error, ext);

}

}

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

{

RegisteredObjects.AddConnection(typeof(MsSqlDataConnection));

Report report = new Report();

MsSqlDataConnection connection = new MsSqlDataConnection();

connection.ConnectionString = ConfigurationManager.ConnectionStrings["KursovayaAvtoparkAvtobusov"].ToString();

connection.CreateAllTables();

report.Dictionary.Connections.Add(connection);

report.RegisterData(connection.DataSet, "mssql");

report.GetDataSource("Employees").Enabled = true;

report.GetDataSource("Maintenance").Enabled = true;

report.GetDataSource("Marshuti").Enabled = true;

report.GetDataSource("Prodazhi").Enabled = true;

ReportPage page = new ReportPage();

report.Pages.Add(page);

page.CreateUniqueName();

page.TopMargin = 10.0f;

page.LeftMargin = 10.0f;

page.RightMargin = 10.0f;

page.BottomMargin = 10.0f;

DataBand data = new DataBand();

data.CreateUniqueName();

page.AddChild(data);

data.DataSource = report.GetDataSource("Employees"); //adding a table to a bend

data.Height = Units.Centimeters \* 1.5f;

// Creating report title

page.ReportTitle = new ReportTitleBand();

page.ReportTitle.CreateUniqueName();

page.ReportTitle.Height = 4.0f \* Units.Centimeters;

TextObject title2Text = new TextObject();

title2Text.CreateUniqueName();

title2Text.Left = 1.0f \* Units.Centimeters;

title2Text.Top = 1.0f \* Units.Centimeters;

title2Text.Width = 17.0f \* Units.Centimeters;

title2Text.Height = 2.0f \* Units.Centimeters;

title2Text.HorzAlign = HorzAlign.Center;

title2Text.VertAlign = VertAlign.Center;

title2Text.Font = new Font("Times New Roman", 32.0f);

title2Text.TextColor = Color.Black;

title2Text.FillColor = Color.Wheat;

title2Text.Border.Color = Color.Black;

title2Text.Border.Lines = BorderLines.All;

title2Text.Border.Width = 4.0f;

title2Text.Text = "Avtopark Report";

page.ReportTitle.Objects.Add(title2Text);

// Header object

TextObject headerText = new TextObject();

headerText.CreateUniqueName();

headerText.Bounds = new RectangleF(0.0f, 3.2f \* Units.Centimeters,

19.0f \* Units.Centimeters, 0.8f \* Units.Centimeters);

headerText.HorzAlign = HorzAlign.Left;

headerText.VertAlign = VertAlign.Center;

headerText.Font = new Font("Times New Roman", 10.0f);

headerText.Text = "Employees";

page.ReportTitle.Objects.Add(headerText);

// Employee Table Text bands

TextObject textx = new TextObject();

textx.Parent = data;

textx.CreateUniqueName();

textx.Bounds = new RectangleF(0 \* Units.Centimeters, 0, 1.0f \* Units.Centimeters, Units.Centimeters \* 0.5f);

textx.Font = new Font("Times New Roman", 5.0f);

textx.Text = "Num"; //table and its field

textx.Border.Lines = BorderLines.All;

TextObject textxx = new TextObject();

textxx.Parent = data;

textxx.CreateUniqueName();

textxx.Bounds = new RectangleF(1 \* Units.Centimeters, 0, Units.Centimeters \* 3, Units.Centimeters \* 0.5f);

textxx.Font = new Font("Times New Roman", 5.0f);

textxx.Text = "Surname"; //table and its field

textxx.Border.Lines = BorderLines.All;

TextObject textxxx = new TextObject();

textxxx.Parent = data;

textxxx.CreateUniqueName();

textxxx.Bounds = new RectangleF(4.0f \* Units.Centimeters, 0, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

textxxx.Font = new Font("Times New Roman", 5.0f);

textxxx.Text = "Name"; //table and its field

textxxx.Border.Lines = BorderLines.All;

TextObject textxxxx = new TextObject();

textxxxx.Parent = data;

textxxxx.CreateUniqueName();

textxxxx.Bounds = new RectangleF(7.4f \* Units.Centimeters, 0, Units.Centimeters \* 25, Units.Centimeters \* 0.5f);

textxxxx.Font = new Font("Times New Roman", 5.0f);

textxxxx.Text = "Patronym"; //table and its field

textxxxx.Border.Lines = BorderLines.All;

TextObject te = new TextObject();

te.Parent = data;

te.CreateUniqueName();

te.Bounds = new RectangleF(9.9f \* Units.Centimeters, 0, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

te.Font = new Font("Times New Roman", 5.0f);

te.Text = "Employed\_Since"; //table and its field

te.Border.Lines = BorderLines.All;

TextObject ter = new TextObject();

ter.Parent = data;

ter.CreateUniqueName();

ter.Bounds = new RectangleF(14 \* Units.Centimeters, 0, Units.Centimeters \* 30, Units.Centimeters \* 0.5f);

ter.Font = new Font("Times New Roman", 5.0f);

ter.Text = "Job"; //table and its field

ter.Border.Lines = BorderLines.All;

TextObject teXR = new TextObject();

teXR.Parent = data;

teXR.CreateUniqueName();

teXR.Bounds = new RectangleF(17.5f \* Units.Centimeters, 0, Units.Centimeters \* 25, Units.Centimeters \* 0.5f);

teXR.Font = new Font("Times New Roman", 5.0f);

teXR.Text = "Internship"; //table and its field

teXR.Border.Lines = BorderLines.All;

// Employee Table Fields

TextObject bandText = new TextObject();

bandText.CreateUniqueName();

bandText.HorzAlign = HorzAlign.Center;

bandText.Bounds = new RectangleF(0 \* Units.Centimeters, 25,

1.0f \* Units.Centimeters, 0.5f \* Units.Centimeters);

bandText.Font = new Font("Times New Roman", 5.0f);

bandText.Border.Lines = BorderLines.All;

bandText.Text = "[Employees.Num]";

data.AddChild(bandText);

TextObject text1 = new TextObject();

text1.Parent = data;

text1.CreateUniqueName();

text1.Bounds = new RectangleF(1 \* Units.Centimeters, 25, Units.Centimeters \* 3, Units.Centimeters \* 0.5f);

text1.Font = new Font("Times New Roman", 5.0f);

text1.Text = "[Employees.Surname]"; //table and its field

text1.Border.Lines = BorderLines.All;

TextObject text2 = new TextObject();

text2.Parent = data;

text2.CreateUniqueName();

text2.Bounds = new RectangleF(4.0f \* Units.Centimeters, 25, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

text2.Font = new Font("Times New Roman", 5.0f);

text2.Text = "[Employees.Name]"; //table and its field

text2.Border.Lines = BorderLines.All;

TextObject text3 = new TextObject();

text3.Parent = data;

text3.CreateUniqueName();

text3.Bounds = new RectangleF(7.4f \* Units.Centimeters, 25, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

text3.Font = new Font("Times New Roman", 5.0f);

text3.Text = "[Employees.Patronym]"; //table and its field

text3.Border.Lines = BorderLines.All;

TextObject text4 = new TextObject();

text4.Parent = data;

text4.CreateUniqueName();

text4.Bounds = new RectangleF(9.9f \* Units.Centimeters, 25, Units.Centimeters \* 30, Units.Centimeters \* 0.5f);

text4.Font = new Font("Times New Roman", 5.0f);

text4.Text = "[Employees.Employed\_Since]"; //table and its field

text4.Border.Lines = BorderLines.All;

TextObject text5 = new TextObject();

text5.Parent = data;

text5.CreateUniqueName();

text5.Bounds = new RectangleF(14 \* Units.Centimeters, 25, Units.Centimeters \* 5, Units.Centimeters \* 0.5f);

text5.Font = new Font("Times New Roman", 5.0f);

text5.Text = "[Employees.Job]"; //table and its field

text5.Border.Lines = BorderLines.All;

TextObject text6 = new TextObject();

text6.Parent = data;

text6.CreateUniqueName();

text6.Bounds = new RectangleF(17.5f \* Units.Centimeters, 25, Units.Centimeters \* 5, Units.Centimeters \* 0.5f);

text6.Font = new Font("Times New Roman", 5.0f);

text6.Text = "[Employees.Internship]"; //table and its field

text6.Border.Lines = BorderLines.All;

// page2

ReportPage page2 = new ReportPage();

report.Pages.Add(page2);

page2.CreateUniqueName();

page2.TopMargin = 10.0f;

page2.LeftMargin = 10.0f;

page2.RightMargin = 10.0f;

page2.BottomMargin = 10.0f;

page2.ReportTitle = new ReportTitleBand();

page2.ReportTitle.CreateUniqueName();

page2.ReportTitle.Height = 4.0f \* Units.Centimeters;

TextObject titleText = new TextObject();

titleText.CreateUniqueName();

titleText.Left = 1.0f \* Units.Centimeters;

titleText.Top = 1.0f \* Units.Centimeters;

titleText.Width = 17.0f \* Units.Centimeters;

titleText.Height = 2.0f \* Units.Centimeters;

titleText.HorzAlign = HorzAlign.Center;

titleText.VertAlign = VertAlign.Center;

titleText.Font = new Font("Times New Roman", 32.0f);

titleText.TextColor = Color.Black;

titleText.FillColor = Color.Wheat;

titleText.Border.Color = Color.Black;

titleText.Border.Lines = BorderLines.All;

titleText.Border.Width = 4.0f;

titleText.Text = "Avtopark Report";

page2.ReportTitle.Objects.Add(titleText);

// Maintenance table things\

DataBand datax = new DataBand();

datax.CreateUniqueName();

page2.AddChild(datax);

datax.DataSource = report.GetDataSource("Maintenance"); //adding a table to a bend

datax.Height = Units.Centimeters \* 5f;

//Maintenance Header object

TextObject headerSText = new TextObject();

headerSText.CreateUniqueName();

headerSText.Bounds = new RectangleF(0.0f, 3.2f \* Units.Centimeters,

19.0f \* Units.Centimeters, 0.8f \* Units.Centimeters);

headerSText.HorzAlign = HorzAlign.Left;

headerSText.VertAlign = VertAlign.Center;

headerSText.Font = new Font("Times New Roman", 10.0f);

headerSText.Border.Width = 4.0f;

headerSText.Text = "Maintenance";

page2.ReportTitle.Objects.Add(headerSText);

//Maintenance Table Text bands

TextObject textxxrx = new TextObject();

textxxrx.Parent = datax;

textxxrx.CreateUniqueName();

textxxrx.Bounds = new RectangleF(0 \* Units.Centimeters, 39.9f, 0.7f \* Units.Centimeters, Units.Centimeters \* 0.5f);

textxxrx.Font = new Font("Times New Roman", 6.0f);

textxxrx.Text = "Num"; //table and its field

textxxrx.Border.Lines = BorderLines.All;

TextObject textxxxr = new TextObject();

textxxxr.Parent = datax;

textxxxr.CreateUniqueName();

textxxxr.Font = new Font("Times New Roman", 6.0f);

textxxxr.Bounds = new RectangleF(0.7f \* Units.Centimeters, 39.9f, Units.Centimeters \* 7, Units.Centimeters \* 0.5f);

textxxxr.Text = "Model\_Avtobusa"; //table and its field

textxxxr.Border.Lines = BorderLines.All;

TextObject textxxxxr = new TextObject();

textxxxxr.Parent = datax;

textxxxxr.CreateUniqueName();

textxxxxr.Bounds = new RectangleF(4.5f \* Units.Centimeters, 39.9f, Units.Centimeters \* 15, Units.Centimeters \* 0.5f);

textxxxxr.Font = new Font("Times New Roman", 5.0f);

textxxxxr.Text = "Data\_Poslednego\_Obsluzhivania"; //table and its field

textxxxxr.Border.Lines = BorderLines.All;

TextObject textxxxxx = new TextObject();

textxxxxx.Parent = datax;

textxxxxx.Font = new Font("Times New Roman", 5.0f);

textxxxxx.CreateUniqueName();

textxxxxx.Bounds = new RectangleF(7.7f \* Units.Centimeters, 39.9f, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

textxxxxx.Text = "Ingener\_Obsluzhivania"; //table and its field

textxxxxx.Border.Lines = BorderLines.All;

TextObject terzx = new TextObject();

terzx.Parent = datax;

terzx.CreateUniqueName();

terzx.Bounds = new RectangleF(10.4f \* Units.Centimeters, 39.9f, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

terzx.Font = new Font("Times New Roman", 5.0f);

terzx.Text = "Problemi\_Avtobusa"; //table and its field

terzx.Border.Lines = BorderLines.All;

TextObject terxr = new TextObject();

terxr.Parent = datax;

terxr.CreateUniqueName();

terxr.Bounds = new RectangleF(15 \* Units.Centimeters, 39.9f, Units.Centimeters \* 15, Units.Centimeters \* 0.5f);

terxr.Font = new Font("Times New Roman", 4.0f);

terxr.Text = "Data\_Sledueschego\_Obsluzivania"; //table and its field

terxr.Border.Lines = BorderLines.All;

TextObject teXRr = new TextObject();

teXRr.Parent = datax;

teXRr.CreateUniqueName();

teXRr.Bounds = new RectangleF(17.5f \* Units.Centimeters, 39.9f, Units.Centimeters \* 26, Units.Centimeters \* 0.5f);

teXRr.Font = new Font("Times New Roman", 6.0f);

teXRr.Text = "Goden\_K\_Doroge"; //table and its field

teXRr.Border.Lines = BorderLines.All;

//Maintenance table fields

TextObject textxxrxx = new TextObject();

textxxrxx.Parent = datax;

textxxrxx.CreateUniqueName();

textxxrxx.Bounds = new RectangleF(0 \* Units.Centimeters, 60.9f, 0.7f \* Units.Centimeters, Units.Centimeters \* 0.5f);

textxxrxx.Font = new Font("Times New Roman", 6.0f);

textxxrxx.Text = "[Maintenance.Num]"; //table and its field

textxxrxx.Border.Lines = BorderLines.All;

TextObject textxxxrx = new TextObject();

textxxxrx.Parent = datax;

textxxxrx.CreateUniqueName();

textxxxrx.Font = new Font("Times New Roman", 6.0f);

textxxxrx.Bounds = new RectangleF(0.7f \* Units.Centimeters, 60.9f, Units.Centimeters \* 7, Units.Centimeters \* 0.5f);

textxxxrx.Text = "[Maintenance.Model\_Avtobusa]"; //table and its field

textxxxrx.Border.Lines = BorderLines.All;

TextObject textxxxxrx = new TextObject();

textxxxxrx.Parent = datax;

textxxxxrx.CreateUniqueName();

textxxxxrx.Bounds = new RectangleF(4.5f \* Units.Centimeters, 60.9f, Units.Centimeters \* 15, Units.Centimeters \* 0.5f);

textxxxxrx.Font = new Font("Times New Roman", 5.0f);

textxxxxrx.Text = "[Maintenance.Data\_Poslednego\_Obsluzhivania]"; //table and its field

textxxxxrx.Border.Lines = BorderLines.All;

TextObject textxxxxxx = new TextObject();

textxxxxxx.Parent = datax;

textxxxxxx.Font = new Font("Times New Roman", 5.0f);

textxxxxxx.CreateUniqueName();

textxxxxxx.Bounds = new RectangleF(7.7f \* Units.Centimeters, 60.9f, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

textxxxxxx.Text = "[Maintenance.Ingener\_Obsluzhivania]"; //table and its field

textxxxxxx.Border.Lines = BorderLines.All;

TextObject terzxx = new TextObject();

terzxx.Parent = datax;

terzxx.CreateUniqueName();

terzxx.Bounds = new RectangleF(10.4f \* Units.Centimeters, 60.9f, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

terzxx.Font = new Font("Times New Roman", 5.0f);

terzxx.Text = "[Maintenance.Problemi\_Avtobusa]"; //table and its field

terzxx.Border.Lines = BorderLines.All;

TextObject terxrx = new TextObject();

terxrx.Parent = datax;

terxrx.CreateUniqueName();

terxrx.Bounds = new RectangleF(15 \* Units.Centimeters, 60.9f, Units.Centimeters \* 15, Units.Centimeters \* 0.5f);

terxrx.Font = new Font("Times New Roman", 6.0f);

terxrx.Text = "[Maintenance.Data\_Sledueschego\_Obsluzivania]"; //table and its field

terxrx.Border.Lines = BorderLines.All;

TextObject teXRrr = new TextObject();

teXRrr.Parent = datax;

teXRrr.CreateUniqueName();

teXRrr.Bounds = new RectangleF(17.5f \* Units.Centimeters, 60.9f, Units.Centimeters \* 26, Units.Centimeters \* 0.5f);

teXRrr.Font = new Font("Times New Roman", 6.0f);

teXRrr.Text = "[Maintenance.Goden\_K\_Doroge]"; //table and its field

teXRrr.Border.Lines = BorderLines.All;

//page3

ReportPage page3 = new ReportPage();

report.Pages.Add(page3);

page3.CreateUniqueName();

page3.TopMargin = 10.0f;

page3.LeftMargin = 10.0f;

page3.RightMargin = 10.0f;

page3.BottomMargin = 10.0f;

page3.ReportTitle = new ReportTitleBand();

page3.ReportTitle.CreateUniqueName();

page3.ReportTitle.Height = 4.0f \* Units.Centimeters;

TextObject Title3 = new TextObject();

Title3.CreateUniqueName();

Title3.Left = 1.0f \* Units.Centimeters;

Title3.Top = 1.0f \* Units.Centimeters;

Title3.Width = 17.0f \* Units.Centimeters;

Title3.Height = 2.0f \* Units.Centimeters;

Title3.HorzAlign = HorzAlign.Center;

Title3.VertAlign = VertAlign.Center;

Title3.Font = new Font("Times New Roman", 32.0f);

Title3.TextColor = Color.Black;

Title3.FillColor = Color.Wheat;

Title3.Border.Color = Color.Black;

Title3.Border.Lines = BorderLines.All;

Title3.Border.Width = 4.0f;

Title3.Text = "Avtopark Report";

page3.ReportTitle.Objects.Add(Title3);

TextObject headersSText = new TextObject();

headersSText.CreateUniqueName();

headersSText.Bounds = new RectangleF(0.0f, 3.2f \* Units.Centimeters,

19.0f \* Units.Centimeters, 0.8f \* Units.Centimeters);

headersSText.HorzAlign = HorzAlign.Left;

headersSText.VertAlign = VertAlign.Center;

headersSText.Font = new Font("Times New Roman", 10.0f);

headersSText.Text = "Prodazhi";

page3.ReportTitle.Objects.Add(headersSText);

// Prodazhi Table Info

DataBand dataxr = new DataBand();

dataxr.CreateUniqueName();

page3.AddChild(dataxr);

dataxr.DataSource = report.GetDataSource("Prodazhi"); //adding a table to a bend

dataxr.Height = Units.Centimeters \* 5f;

TextObject texrxx = new TextObject();

texrxx.Parent = dataxr;

texrxx.CreateUniqueName();

texrxx.Bounds = new RectangleF(0 \* Units.Centimeters, 43.99f, 0.7f \* Units.Centimeters, Units.Centimeters \* 0.5f);

texrxx.Font = new Font("Times New Roman", 6.0f);

texrxx.Text = "Num"; //table and its field

texrxx.Border.Lines = BorderLines.All;

TextObject texrxs = new TextObject();

texrxs.Parent = dataxr;

texrxs.CreateUniqueName();

texrxs.Font = new Font("Times New Roman", 6.0f);

texrxs.Bounds = new RectangleF(0.7f \* Units.Centimeters, 43.99f, Units.Centimeters \* 7, Units.Centimeters \* 0.5f);

texrxs.Text = "Sale\_Date"; //table and its field

texrxs.Border.Lines = BorderLines.All;

TextObject terscw = new TextObject();

terscw.Parent = dataxr;

terscw.CreateUniqueName();

terscw.Bounds = new RectangleF(4.5f \* Units.Centimeters, 43.99f, Units.Centimeters \* 15, Units.Centimeters \* 0.5f);

terscw.Font = new Font("Times New Roman", 5.0f);

terscw.Text = "Nomer\_Avtobusa"; //table and its field

terscw.Border.Lines = BorderLines.All;

TextObject terscwx = new TextObject();

terscwx.Parent = dataxr;

terscwx.Font = new Font("Times New Roman", 5.0f);

terscwx.CreateUniqueName();

terscwx.Bounds = new RectangleF(7.7f \* Units.Centimeters, 43.99f, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

terscwx.Text = "Punkt\_Posadki"; //table and its field

terscwx.Border.Lines = BorderLines.All;

TextObject terzrx = new TextObject();

terzrx.Parent = dataxr;

terzrx.CreateUniqueName();

terzrx.Bounds = new RectangleF(10.4f \* Units.Centimeters, 43.99f, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

terzrx.Font = new Font("Times New Roman", 5.0f);

terzrx.Text = "Stoimost"; //table and its field

terzrx.Border.Lines = BorderLines.All;

// Table Fields For Prodazhi

TextObject texrxxx = new TextObject();

texrxxx.Parent = dataxr;

texrxxx.CreateUniqueName();

texrxxx.Bounds = new RectangleF(0 \* Units.Centimeters, 53.99f, 0.7f \* Units.Centimeters, Units.Centimeters \* 0.5f);

texrxxx.Font = new Font("Times New Roman", 6.0f);

texrxxx.Text = "[Prodazhi.Num]"; //table and its field

texrxxx.Border.Lines = BorderLines.All;

TextObject texrxss = new TextObject();

texrxss.Parent = dataxr;

texrxss.CreateUniqueName();

texrxss.Font = new Font("Times New Roman", 6.0f);

texrxss.Bounds = new RectangleF(0.7f \* Units.Centimeters, 53.99f, Units.Centimeters \* 7, Units.Centimeters \* 0.5f);

texrxss.Text = "[Prodazhi.Sale\_Date]"; //table and its field

texrxss.Border.Lines = BorderLines.All;

TextObject terscwxxxr = new TextObject();

terscwxxxr.Parent = dataxr;

terscwxxxr.CreateUniqueName();

terscwxxxr.Bounds = new RectangleF(4.5f \* Units.Centimeters, 53.99f, Units.Centimeters \* 15, Units.Centimeters \* 0.5f);

terscwxxxr.Font = new Font("Times New Roman", 5.0f);

terscwxxxr.Text = "[Prodazhi.Nomer\_Avtobusa]"; //table and its field

terscwxxxr.Border.Lines = BorderLines.All;

TextObject terscwxxr = new TextObject();

terscwxxr.Parent = dataxr;

terscwxxr.Font = new Font("Times New Roman", 5.0f);

terscwxxr.CreateUniqueName();

terscwxxr.Bounds = new RectangleF(7.7f \* Units.Centimeters, 53.99f, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

terscwxxr.Text = "[Prodazhi.Punkt\_Posadki]"; //table and its field

terscwxxr.Border.Lines = BorderLines.All;

TextObject terzrxr = new TextObject();

terzrxr.Parent = dataxr;

terzrxr.CreateUniqueName();

terzrxr.Bounds = new RectangleF(10.4f \* Units.Centimeters, 53.99f, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

terzrxr.Font = new Font("Times New Roman", 5.0f);

terzrxr.Text = "[Prodazhi.Stoimost]"; //table and its field

terzrxr.Border.Lines = BorderLines.All;

//page4

ReportPage page4 = new ReportPage();

report.Pages.Add(page4);

page4.CreateUniqueName();

page4.TopMargin = 10.0f;

page4.LeftMargin = 10.0f;

page4.RightMargin = 10.0f;

page4.BottomMargin = 10.0f;

page4.ReportTitle = new ReportTitleBand();

page4.ReportTitle.CreateUniqueName();

page4.ReportTitle.Height = 4.0f \* Units.Centimeters;

TextObject title4 = new TextObject();

title4.CreateUniqueName();

title4.Left = 1.0f \* Units.Centimeters;

title4.Top = 1.0f \* Units.Centimeters;

title4.Width = 17.0f \* Units.Centimeters;

title4.Height = 2.0f \* Units.Centimeters;

title4.HorzAlign = HorzAlign.Center;

title4.VertAlign = VertAlign.Center;

title4.Font = new Font("Times New Roman", 32.0f);

title4.TextColor = Color.Black;

title4.FillColor = Color.Wheat;

title4.Border.Color = Color.Black;

title4.Border.Lines = BorderLines.All;

title4.Border.Width = 4.0f;

title4.Text = "Avtopark Report";

page4.ReportTitle.Objects.Add(title4);

TextObject headtextx = new TextObject();

headtextx.CreateUniqueName();

headtextx.Bounds = new RectangleF(0.0f, 3.2f \* Units.Centimeters,

19.0f \* Units.Centimeters, 0.8f \* Units.Centimeters);

headtextx.HorzAlign = HorzAlign.Left;

headtextx.VertAlign = VertAlign.Center;

headtextx.Font = new Font("Times New Roman", 10.0f);

headtextx.Text = "Marshuti";

page4.ReportTitle.Objects.Add(headtextx);

// Marshuti fields

DataBand datarxr = new DataBand();

datarxr.CreateUniqueName();

page4.AddChild(datarxr);

datarxr.DataSource = report.GetDataSource("Marshuti"); //adding a table to a bend

datarxr.Height = Units.Centimeters \* 2f;

TextObject txtef = new TextObject();

txtef.Parent = datarxr;

txtef.CreateUniqueName();

txtef.Bounds = new RectangleF(0 \* Units.Centimeters, 58.9f, 0.7f \* Units.Centimeters, Units.Centimeters \* 0.5f);

txtef.Font = new Font("Times New Roman", 6.0f);

txtef.Text = "Nomer\_Marshuta"; //table and its field

txtef.Border.Lines = BorderLines.All;

TextObject tfeqa = new TextObject();

tfeqa.Parent = datarxr;

tfeqa.CreateUniqueName();

tfeqa.Font = new Font("Times New Roman", 6.0f);

tfeqa.Bounds = new RectangleF(0.7f \* Units.Centimeters, 58.9f, Units.Centimeters \* 7, Units.Centimeters \* 0.5f);

tfeqa.Text = "Nachalni\_Punkt"; //table and its field

tfeqa.Border.Lines = BorderLines.All;

TextObject tcqqcq = new TextObject();

tcqqcq.Parent = datarxr;

tcqqcq.CreateUniqueName();

tcqqcq.Bounds = new RectangleF(4.5f \* Units.Centimeters, 58.9f, Units.Centimeters \* 15, Units.Centimeters \* 0.5f);

tcqqcq.Font = new Font("Times New Roman", 5.0f);

tcqqcq.Text = "Konechni\_Punkt"; //table and its field

tcqqcq.Border.Lines = BorderLines.All;

TextObject txtweq = new TextObject();

txtweq.Parent = datarxr;

txtweq.Font = new Font("Times New Roman", 5.0f);

txtweq.CreateUniqueName();

txtweq.Bounds = new RectangleF(7.7f \* Units.Centimeters, 58.9f, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

txtweq.Text = "Voditel"; //table and its field

txtweq.Border.Lines = BorderLines.All;

TextObject terrzx = new TextObject();

terrzx.Parent = datarxr;

terrzx.CreateUniqueName();

terrzx.Bounds = new RectangleF(10.4f \* Units.Centimeters, 58.9f, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

terrzx.Font = new Font("Times New Roman", 5.0f);

terrzx.Text = "Model\_Avtobusa"; //table and its field

terrzx.Border.Lines = BorderLines.All;

TextObject txerxr = new TextObject();

txerxr.Parent = datarxr;

txerxr.CreateUniqueName();

txerxr.Bounds = new RectangleF(15 \* Units.Centimeters, 58.9f, Units.Centimeters \* 15, Units.Centimeters \* 0.5f);

txerxr.Font = new Font("Times New Roman", 4.0f);

txerxr.Text = "Vremya\_Proezda"; //table and its field

txerxr.Border.Lines = BorderLines.All;

TextObject txteff = new TextObject();

txteff.Parent = datarxr;

txteff.CreateUniqueName();

txteff.Bounds = new RectangleF(0 \* Units.Centimeters, 82.9f, 0.7f \* Units.Centimeters, Units.Centimeters \* 0.5f);

txteff.Font = new Font("Times New Roman", 6.0f);

txteff.Text = "[Marshuti.Nomer\_Marshuta]"; //table and its field

txteff.Border.Lines = BorderLines.All;

TextObject tfeqafd = new TextObject();

tfeqafd.Parent = datarxr;

tfeqafd.CreateUniqueName();

tfeqafd.Font = new Font("Times New Roman", 6.0f);

tfeqafd.Bounds = new RectangleF(0.7f \* Units.Centimeters, 82.9f, Units.Centimeters \* 7, Units.Centimeters \* 0.5f);

tfeqafd.Text = "[Marshuti.Nachalni\_Punkt]"; //table and its field

tfeqafd.Border.Lines = BorderLines.All;

TextObject tcqqcqd = new TextObject();

tcqqcqd.Parent = datarxr;

tcqqcqd.CreateUniqueName();

tcqqcqd.Bounds = new RectangleF(4.5f \* Units.Centimeters, 82.9f, Units.Centimeters \* 15, Units.Centimeters \* 0.5f);

tcqqcqd.Font = new Font("Times New Roman", 5.0f);

tcqqcqd.Text = "[Marshuti.Konechni\_Punkt]"; //table and its field

tcqqcqd.Border.Lines = BorderLines.All;

TextObject txtweqq = new TextObject();

txtweqq.Parent = datarxr;

txtweqq.Font = new Font("Times New Roman", 5.0f);

txtweqq.CreateUniqueName();

txtweqq.Bounds = new RectangleF(7.7f \* Units.Centimeters, 82.9f, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

txtweqq.Text = "[Marshuti.Voditel]"; //table and its field

txtweqq.Border.Lines = BorderLines.All;

TextObject terrzxr = new TextObject();

terrzxr.Parent = datarxr;

terrzxr.CreateUniqueName();

terrzxr.Bounds = new RectangleF(10.4f \* Units.Centimeters, 82.9f, Units.Centimeters \* 20, Units.Centimeters \* 0.5f);

terrzxr.Font = new Font("Times New Roman", 5.0f);

terrzxr.Text = "[Marshuti.Model\_Avtobusa]"; //table and its field

terrzxr.Border.Lines = BorderLines.All;

TextObject txerxxr = new TextObject();

txerxxr.Parent = datarxr;

txerxxr.CreateUniqueName();

txerxxr.Bounds = new RectangleF(15 \* Units.Centimeters, 82.9f, Units.Centimeters \* 15, Units.Centimeters \* 0.5f);

txerxxr.Font = new Font("Times New Roman", 4.0f);

txerxxr.Text = "[Marshuti.Vremya\_Proezda]"; //table and its field

txerxxr.Border.Lines = BorderLines.All;

// Footer

page.PageFooter = new PageFooterBand();

page.PageFooter.CreateUniqueName();

page.PageFooter.Height = 0.5f \* Units.Centimeters;

page2.PageFooter = new PageFooterBand();

page2.PageFooter.CreateUniqueName();

page2.PageFooter.Height = 0.5f \* Units.Centimeters;

page3.PageFooter = new PageFooterBand();

page3.PageFooter.CreateUniqueName();

page3.PageFooter.Height = 0.5f \* Units.Centimeters;

page4.PageFooter = new PageFooterBand();

page4.PageFooter.CreateUniqueName();

page4.PageFooter.Height = 0.5f \* Units.Centimeters;

TextObject footerText = new TextObject();

footerText.CreateUniqueName();

footerText.HorzAlign = HorzAlign.Right;

footerText.VertAlign = VertAlign.Center;

footerText.Bounds = new RectangleF(0.0f, 0.0f,

19.0f \* Units.Centimeters, 0.5f \* Units.Centimeters);

footerText.TextColor = Color.Black;

footerText.FillColor = Color.Wheat;

titleText.Border.Color = Color.Black;

titleText.Border.Lines = BorderLines.All;

titleText.Border.Width = 4.0f;

footerText.Text = "Page [Page]";

TextObject f2 = new TextObject();

f2.CreateUniqueName();

f2.HorzAlign = HorzAlign.Right;

f2.VertAlign = VertAlign.Center;

f2.Bounds = new RectangleF(0.0f, 0.0f,

19.0f \* Units.Centimeters, 0.5f \* Units.Centimeters);

f2.TextColor = Color.Black;

f2.FillColor = Color.Wheat;

titleText.Border.Color = Color.Black;

titleText.Border.Lines = BorderLines.All;

titleText.Border.Width = 4.0f;

f2.Text = "Page [Page]";

TextObject f3x = new TextObject();

f3x.CreateUniqueName();

f3x.HorzAlign = HorzAlign.Right;

f3x.VertAlign = VertAlign.Center;

f3x.Bounds = new RectangleF(0.0f, 0.0f,

19.0f \* Units.Centimeters, 0.5f \* Units.Centimeters);

f3x.TextColor = Color.Black;

f3x.FillColor = Color.Wheat;

titleText.Border.Color = Color.Black;

titleText.Border.Lines = BorderLines.All;

titleText.Border.Width = 4.0f;

f3x.Text = "Page [Page]";

TextObject f4x = new TextObject();

f4x.CreateUniqueName();

f4x.HorzAlign = HorzAlign.Right;

f4x.VertAlign = VertAlign.Center;

f4x.Bounds = new RectangleF(0.0f, 0.0f,

19.0f \* Units.Centimeters, 0.5f \* Units.Centimeters);

f4x.TextColor = Color.Black;

f4x.FillColor = Color.Wheat;

titleText.Border.Color = Color.Black;

titleText.Border.Lines = BorderLines.All;

titleText.Border.Width = 4.0f;

f4x.Text = "Page [Page]";

page.PageFooter.Objects.Add(footerText);

page2.PageFooter.Objects.Add(f2);

page3.PageFooter.Objects.Add(f3x);

page4.PageFooter.Objects.Add(f4x);

try

{

// Exporting the report

report.Prepare();

report.SavePrepared("PreparedComplexReport.fpx");

PDFSimpleExport pdf = new PDFSimpleExport();

// Save the report

report.Export(pdf, "ExportedComplexPDF.pdf");

}

catch (Exception ex)

{

MsgBoxExtendedFunctionality ext = new MsgBoxExtendedFunctionality()

{

DetailsText = ex.StackTrace

};

MessageBoxEx.ShowEx("An error has occured! Traceback: " + ex.Message, "Unexpected situation handling", MessageBoxButtonEx.OK, MessageBoxImage.Error, ext);

}

}

}

}

using MsgBoxEx;

using System;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Windows;

namespace KursovayaAvtoparkAvtobusov

{

/// <summary>

/// Interaction logic for EmpRecord.xaml

/// </summary>

public partial class EmpRecord : Window

{

public EmpRecord()

{

InitializeComponent();

}

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

{

try

{

SqlConnection cnn;

var conStr = MainWindow.formations;

cnn = new SqlConnection(conStr);

cnn.Open();

Console.Write("OPENING DB CONNECTION!!! \n");

Console.Write("[MENU] Connect db clicked - test \n");

var select = "SELECT \* FROM Employees";

var commandBuilder = new SqlCommand(select, cnn);

commandBuilder.ExecuteNonQuery();

var dataAdapter = new SqlDataAdapter(commandBuilder);

var ds = new DataTable("Avtobusx");

dataAdapter.Fill(ds);

dataGrid1.ItemsSource = ds.DefaultView;

Console.WriteLine("Connection established and the datagrid filled!");

cnn.Close();

}

catch (Exception ex)

{

MsgBoxExtendedFunctionality ext = new MsgBoxExtendedFunctionality()

{

DetailsText = ex.StackTrace

};

MessageBoxEx.ShowEx("An error has occured! Traceback: " + ex.Message, "Unexpected situation handling", MessageBoxButtonEx.OK, MessageBoxImage.Error, ext);

}

}

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

{

try

{

var Num = Exed1.Text;

var EmpName = Exed2.Text;

var EmpsName = Exed3.Text;

var Emppat = Exed4.Text;

var EmpES = Exed5.Text;

var Emptitle = Exed6.Text;

var Empint = Exed7.Text;

var query = "INSERT INTO Employees(Num, Surname, Name, Patronym,Employed\_Since,Job,Internship) " +

"Values('" + Num + "', '" + EmpName + "', '" + EmpsName + "', '" + Emppat + "','" + EmpES +

"','" + Emptitle + "','" + Empint + "')";

SqlConnection cnn;

var conStr = MainWindow.formations;

cnn = new SqlConnection(conStr);

cnn.Open();

var command = new SqlCommand(query, cnn);

command.ExecuteNonQuery();

Console.WriteLine("Data inserted!\n Closing connection ");

cnn.Close();

Console.WriteLine("Connection has been closed , database ready for next operation!");

}

catch (Exception ex)

{

MsgBoxExtendedFunctionality ext = new MsgBoxExtendedFunctionality()

{

DetailsText = ex.StackTrace

};

MessageBoxEx.ShowEx("An error has occured! Traceback: " + ex.Message, "Unexpected situation handling", MessageBoxButtonEx.OK, MessageBoxImage.Error, ext);

}

}

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

{

try

{

SqlConnection cnn;

var conStr = MainWindow.formations;

cnn = new SqlConnection(conStr);

cnn.Open();

Console.Write("OPENING DB CONNECTION!!! \n");

var select =

"UPDATE Employees SET Surname=@a1,Name=@a2,Patronym=@a3,Employed\_Since=@a4,Job=@a5,Internship=@a6 WHERE Num=@a7";

var commandBuilder = new SqlCommand(select, cnn);

commandBuilder.Parameters.AddWithValue("a1", Exed2.Text);

commandBuilder.Parameters.AddWithValue("a2", Exed3.Text);

commandBuilder.Parameters.AddWithValue("a3", Exed4.Text);

commandBuilder.Parameters.AddWithValue("a4", Exed5.Text);

commandBuilder.Parameters.AddWithValue("a5", Exed6.Text);

commandBuilder.Parameters.AddWithValue("a6", Exed7.Text);

commandBuilder.Parameters.AddWithValue("a7", int.Parse(Exed1.Text));

commandBuilder.ExecuteNonQuery();

Console.WriteLine("Connection established and the table updated!");

cnn.Close();

}

catch (Exception ex)

{

MsgBoxExtendedFunctionality ext = new MsgBoxExtendedFunctionality()

{

DetailsText = ex.StackTrace

};

MessageBoxEx.ShowEx("An error has occured! Traceback: " + ex.Message, "Unexpected situation handling", MessageBoxButtonEx.OK, MessageBoxImage.Error, ext);

}

}

}

}

using MsgBoxEx;

using System;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Windows;

namespace KursovayaAvtoparkAvtobusov

{

/// <summary>

/// Interaction logic for TexObslugWindow.xaml

/// </summary>

public partial class TexObslugWindow : Window

{

public TexObslugWindow()

{

InitializeComponent();

}

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

{

try

{

var Num = AvEditor1.Text;

var ModAvtobus = AvEditor2.Text;

var DatObslug = AvEditor3.Text;

var Engineer = AvEditor4.Text;

var ErAvto = AvEditor5.Text;

var DatSlegObslug = AvEditor6.Text;

var GodDorog = AvEditor7.Text;

var query =

"INSERT INTO Maintenance(Num, Model\_Avtobusa, Data\_Poslednego\_Obsluzhivania, Ingener\_Obsluzhivania,Problemi\_Avtobusa,Data\_Sledueschego\_Obsluzivania,Goden\_K\_Doroge) " +

"Values('" + Num + "', '" + ModAvtobus + "', '" + DatObslug + "', '" + Engineer + "','" + ErAvto +

"','" + DatSlegObslug + "','" + GodDorog + "')";

SqlConnection cnn;

var conStr = MainWindow.formations;

cnn = new SqlConnection(conStr);

cnn.Open();

var command = new SqlCommand(query, cnn);

command.ExecuteNonQuery();

Console.WriteLine("Data inserted!\n Closing connection ");

cnn.Close();

Console.WriteLine("Connection has been closed , database ready for next operation!");

}

catch (Exception ex)

{

MsgBoxExtendedFunctionality ext = new MsgBoxExtendedFunctionality()

{

DetailsText = ex.StackTrace

};

MessageBoxEx.ShowEx("An error has occured! Traceback: " + ex.Message, "Unexpected situation handling", MessageBoxButtonEx.OK, MessageBoxImage.Error, ext);

}

}

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

{

try

{

SqlConnection cnn;

var conStr = MainWindow.formations;

cnn = new SqlConnection(conStr);

cnn.Open();

Console.Write("OPENING DB CONNECTION!!! \n");

Console.Write("[MENU] Connect db clicked - test \n");

var select = "SELECT \* FROM Maintenance";

var commandBuilder = new SqlCommand(select, cnn);

commandBuilder.ExecuteNonQuery();

var dataAdapter = new SqlDataAdapter(commandBuilder);

var ds = new DataTable("Avtobusx");

dataAdapter.Fill(ds);

dataGrid1.ItemsSource = ds.DefaultView;

Console.WriteLine("Connection established and the datagrid filled!");

cnn.Close();

}

catch (Exception ex)

{

MsgBoxExtendedFunctionality ext = new MsgBoxExtendedFunctionality()

{

DetailsText = ex.StackTrace

};

MessageBoxEx.ShowEx("An error has occured! Traceback: " + ex.Message, "Unexpected situation handling", MessageBoxButtonEx.OK, MessageBoxImage.Error, ext);

}

}

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

{

try

{

SqlConnection cnn;

var conStrs = ConfigurationManager.ConnectionStrings["KursovayaAvtoparkAvtobusov"].ToString();

cnn = new SqlConnection(conStrs);

cnn.Open();

Console.Write("OPENING DB CONNECTION!!! \n");

var querier =

"UPDATE Maintenance SET Model\_Avtobusa=@a1,Data\_Poslednego\_Obsluzhivania=@a2,Ingener\_Obsluzhivania=@a3,Problemi\_Avtobusa=@a4,Data\_Sledueschego\_Obsluzivania=@a5,Goden\_K\_Doroge=@a6 WHERE Num=@a7";

var commandBuilders = new SqlCommand(querier, cnn);

commandBuilders.Parameters.AddWithValue("a1", AvEditor2.Text);

commandBuilders.Parameters.AddWithValue("a2", AvEditor3.Text);

commandBuilders.Parameters.AddWithValue("a3", AvEditor4.Text);

commandBuilders.Parameters.AddWithValue("a4", AvEditor5.Text);

commandBuilders.Parameters.AddWithValue("a5", AvEditor6.Text);

commandBuilders.Parameters.AddWithValue("a6", AvEditor7.Text);

commandBuilders.Parameters.AddWithValue("a7", int.Parse(AvEditor1.Text));

commandBuilders.ExecuteNonQuery();

Console.WriteLine("Connection established and the table updated!");

cnn.Close();

}

catch (Exception ex)

{

MsgBoxExtendedFunctionality ext = new MsgBoxExtendedFunctionality()

{

DetailsText = ex.StackTrace

};

MessageBoxEx.ShowEx("An error has occured! Traceback: " + ex.Message, "Unexpected situation handling", MessageBoxButtonEx.OK, MessageBoxImage.Error, ext);

}

}

}

}

using MsgBoxEx;

using System;

using System.Configuration;

using System.Data.SqlClient;

using System.Windows;

namespace KursovayaAvtoparkAvtobusov

{

/// <summary>

/// Interaction logic for Window1.xaml

/// </summary>

public partial class Window1 : Window

{

public Window1()

{

InitializeComponent();

}

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

{

try

{

var NomeMarshuta = Edit1.Text;

var NachalnPunkt = Edit2.Text;

var KonechnPunkt = Edit3.Text;

var Vodit = Edit4.Text;

var Modelvtobusa = Edit5.Text;

var VremProezda = Edit6.Text;

var query =

"INSERT INTO Marshuti(Nomer\_Marshuta, Nachalni\_Punkt, Konechni\_Punkt, Voditel,Model\_Avtobusa,Vremya\_Proezda) " +

"Values('" + NomeMarshuta + "', '" + NachalnPunkt + "', '" + KonechnPunkt + "', '" + Vodit + "','" +

Modelvtobusa + "', '" + VremProezda + "')";

SqlConnection cnn;

var conStr = MainWindow.formations;

cnn = new SqlConnection(conStr);

cnn.Open();

var command = new SqlCommand(query, cnn);

command.ExecuteNonQuery();

Console.WriteLine("Data inserted!\n Closing connection ");

cnn.Close();

Console.WriteLine("Connection has been closed , database ready for next operation!");

}

catch (Exception ex)

{

MsgBoxExtendedFunctionality ext = new MsgBoxExtendedFunctionality()

{

DetailsText = ex.StackTrace

};

MessageBoxEx.ShowEx("An error has occured! Traceback: " + ex.Message, "Unexpected situation handling", MessageBoxButtonEx.OK, MessageBoxImage.Error, ext);

}

}

}

}

using MsgBoxEx;

using System;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Windows;

namespace KursovayaAvtoparkAvtobusov

{

/// <summary>

/// Interaction logic for Window3.xaml

/// </summary>

public partial class Window3 : Window

{

public Window3()

{

InitializeComponent();

}

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

{

try

{

SqlConnection cnn;

var conStr = MainWindow.formations;

cnn = new SqlConnection(conStr);

cnn.Open();

Console.Write("OPENING DB CONNECTION!!! \n");

Console.Write("[MENU] Connect db clicked - test \n");

var select = "SELECT \* FROM Prodazhi";

var commandBuilder = new SqlCommand(select, cnn);

commandBuilder.ExecuteNonQuery();

var dataAdapter = new SqlDataAdapter(commandBuilder);

var ds = new DataTable("Avtobusx");

dataAdapter.Fill(ds);

dataGrid1.ItemsSource = ds.DefaultView;

Console.WriteLine("Connection established and the datagrid filled!");

cnn.Close();

}

catch (Exception ex)

{

MsgBoxExtendedFunctionality ext = new MsgBoxExtendedFunctionality()

{

DetailsText = ex.StackTrace

};

MessageBoxEx.ShowEx("An error has occured! Traceback: " + ex.Message, "Unexpected situation handling", MessageBoxButtonEx.OK, MessageBoxImage.Error, ext);

}

}

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

{

try

{

var Num = Edit1.Text;

var SaleDat = Edit2.Text;

var NomAvtobus = Edit3.Text;

var Punkt = Edit4.Text;

var Stoimost = Edit5.Text;

var query = "INSERT INTO Prodazhi(Num, Sale\_Date, Nomer\_Avtobusa, Punkt\_Posadki,Stoimost) " +

"Values('" + Num + "', '" + SaleDat + "', '" + NomAvtobus + "', '" + Punkt + "','" +

Stoimost + "')";

SqlConnection cnn;

var conStr = MainWindow.formations;

cnn = new SqlConnection(conStr);

cnn.Open();

var command = new SqlCommand(query, cnn);

command.ExecuteNonQuery();

Console.WriteLine("Data inserted!\n Closing connection ");

cnn.Close();

Console.WriteLine("Connection has been closed , database ready for next operation!");

}

catch (Exception ex)

{

MsgBoxExtendedFunctionality ext = new MsgBoxExtendedFunctionality()

{

DetailsText = ex.StackTrace

};

MessageBoxEx.ShowEx("An error has occured! Traceback: " + ex.Message, "Unexpected situation handling", MessageBoxButtonEx.OK, MessageBoxImage.Error, ext);

}

}

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

{

try

{

SqlConnection cnn;

var conStrs = ConfigurationManager.ConnectionStrings["KursovayaAvtoparkAvtobusov"].ToString();

cnn = new SqlConnection(conStrs);

cnn.Open();

Console.Write("OPENING DB CONNECTION!!! \n");

var querier =

"UPDATE Prodazhi SET Sale\_Date=@a1,Nomer\_Avtobusa=@a2,Punkt\_Posadki=@a3,Stoimost=@a4 WHERE Num=@a5";

var commandBuilder = new SqlCommand(querier, cnn);

commandBuilder.Parameters.AddWithValue("a1", Edit2.Text);

commandBuilder.Parameters.AddWithValue("a2", int.Parse(Edit3.Text));

commandBuilder.Parameters.AddWithValue("a3", Edit4.Text);

commandBuilder.Parameters.AddWithValue("a4", Edit5.Text);

commandBuilder.Parameters.AddWithValue("a5", int.Parse(Edit1.Text));

commandBuilder.ExecuteNonQuery();

Console.WriteLine("Connection established and the table updated!");

cnn.Close();

}

catch (Exception ex)

{

MsgBoxExtendedFunctionality ext = new MsgBoxExtendedFunctionality()

{

DetailsText = ex.StackTrace

};

MessageBoxEx.ShowEx("An error has occured! Traceback: " + ex.Message, "Unexpected situation handling", MessageBoxButtonEx.OK, MessageBoxImage.Error, ext);

}

}

}

}

using MsgBoxEx;

using PostSharp.Patterns.Diagnostics;

using System;

using System.Configuration;

using System.Data.SqlClient;

using System.Windows;

using System.Windows.Input;

namespace KursovayaAvtoparkAvtobusov

{

/// <summary>

/// Interaction logic for Window4.xaml

/// </summary>

public partial class Window4 : Window

{

public Window4()

{

InitializeComponent();

}

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

{

Close();

}

[Log("Audit")]

private void deletedbdata\_Click(object sender,

RoutedEventArgs e)

{

switch (TableName.Text)

{

case "Prodazhi":

{

SqlConnection cnn;

var conStr = MainWindow.formations;

cnn = new SqlConnection(conStr);

cnn.Open();

Console.Write("OPENING DB CONNECTION!!! \n");

var select = "DELETE FROM Prodazhi WHERE Num=@a2";

var commandBuilder = new SqlCommand(select, cnn);

commandBuilder.Parameters.AddWithValue("a2", keybox.Text);

try

{

commandBuilder.ExecuteNonQuery();

Console.WriteLine("Delete operation successful!");

}

catch (SqlException ex)

{

MessageBoxEx.ShowEx(ex.Message);

}

break;

}

case "Marshuti":

{

SqlConnection cnn;

var conStr = MainWindow.formations;

cnn = new SqlConnection(conStr);

cnn.Open();

Console.Write("OPENING DB CONNECTION!!! \n");

var select = "DELETE FROM Marshuti WHERE Nomer\_Marshuta=@a2";

var commandBuilder = new SqlCommand(select, cnn);

commandBuilder.Parameters.AddWithValue("a2", keybox.Text);

try

{

commandBuilder.ExecuteNonQuery();

Console.WriteLine("Delete operation successful!");

}

catch (SqlException ex)

{

MessageBoxEx.ShowEx(ex.Message);

}

break;

}

case "Maintenance":

{

SqlConnection cnn;

var conStr = MainWindow.formations;

cnn = new SqlConnection(conStr);

cnn.Open();

Console.Write("OPENING DB CONNECTION!!! \n");

var select = "DELETE FROM Maintenance WHERE Num=@a2";

var commandBuilder = new SqlCommand(select, cnn);

commandBuilder.Parameters.AddWithValue("a2", keybox.Text);

try

{

commandBuilder.ExecuteNonQuery();

Console.WriteLine("Delete operation successful!");

}

catch (SqlException ex)

{

MessageBoxEx.ShowEx(ex.Message);

}

break;

}

case "Employees":

{

SqlConnection cnn;

var conStr = MainWindow.formations;

cnn = new SqlConnection(conStr);

cnn.Open();

Console.Write("OPENING DB CONNECTION!!! \n");

var select = "DELETE FROM Employees WHERE Num=@a2";

var commandBuilder = new SqlCommand(select, cnn);

commandBuilder.Parameters.AddWithValue("a2", keybox.Text);

try

{

commandBuilder.ExecuteNonQuery();

Console.WriteLine("Delete operation successful!");

}

catch (SqlException ex)

{

MessageBoxEx.ShowEx(ex.Message);

}

break;

}

default:

MessageBoxEx.ShowEx(

"Введено неправильное имя таблицы! Пожалуйста введите одно из следующих - Prodazhi или Marshuti или Maintenance!");

break;

}

}

private void Rectangle\_MouseDown(object sender, MouseButtonEventArgs e)

{

if (e.ChangedButton == MouseButton.Left) DragMove();

}

}

}

using System.Windows;

namespace KursovayaAvtoparkAvtobusov

{

/// <summary>

/// Interaction logic for Window2.xaml

/// </summary>

public partial class Window2 : Window

{

public Window2()

{

InitializeComponent();

}

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

{

this.Close();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

namespace SQLServerLoginTemplate

{

// https://thomaslevesque.com/2013/05/21/an-easy-and-secure-way-to-store-a-password-using-data-protection-api/

public static class DataProtectionExtensions

{

public static string Protect(

this string clearText,

string optionalEntropy = null,

DataProtectionScope scope = DataProtectionScope.CurrentUser)

{

if (clearText == null)

throw new ArgumentNullException("clearText");

byte[] clearBytes = Encoding.UTF8.GetBytes(clearText);

byte[] entropyBytes = string.IsNullOrEmpty(optionalEntropy)

? null

: Encoding.UTF8.GetBytes(optionalEntropy);

byte[] encryptedBytes = ProtectedData.Protect(clearBytes, entropyBytes, scope);

return Convert.ToBase64String(encryptedBytes);

}

public static string Unprotect(

this string encryptedText,

string optionalEntropy = null,

DataProtectionScope scope = DataProtectionScope.CurrentUser)

{

if (encryptedText == null)

throw new ArgumentNullException("encryptedText");

byte[] encryptedBytes = Convert.FromBase64String(encryptedText);

byte[] entropyBytes = string.IsNullOrEmpty(optionalEntropy)

? null

: Encoding.UTF8.GetBytes(optionalEntropy);

byte[] clearBytes = new byte[0];

try

{

clearBytes = ProtectedData.Unprotect(encryptedBytes, entropyBytes, scope);

}

catch

{

//Do nothing - just return empty if unable to decrypt the original password

}

return Encoding.UTF8.GetString(clearBytes);

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace SQLServerLoginTemplate

{

public partial class FormConnectToSQLServer : Form

{

private readonly string \_WINDOWSAUTH = "Windows Authentication";

private readonly string \_SQLAUTH = "SQL Server Authentication";

private readonly string \_LOGINTYPEWINDOWS = "0";

private readonly string \_LOGINTYPESQL = "1";

private readonly string \_SERVERANDUSERSEPARATOR = "@&";

private readonly string \_USERANDPASSWORDSEPARATOR = "~~~";

public string ConnectionString = ""; //The code that calls this form should retrieve the connection string from this public property

public FormConnectToSQLServer()

{

InitializeComponent();

}

private void buttonCancel\_Click(object sender, EventArgs e)

{

this.DialogResult = DialogResult.Cancel;

this.Close();

}

private void FormConnectToSQLServer\_Load(object sender, EventArgs e)

{

GetDPI();

LoadSharedDefaultValues();

InitializePropertiesForFirstTimeUsers();

LoadLastServerSelectedByThisUser();

FormConnectToSQLServer\_Resize(sender, e);

}

bool \_highDPI = false;

private void GetDPI()

{

float dx, dy;

Graphics g = this.CreateGraphics();

try

{

dx = g.DpiX;

dy = g.DpiY;

if (dx > 96)

\_highDPI = true;

}

finally

{

g.Dispose();

}

}

public void LoadLastServerSelectedByThisUser()

{

if (Properties.Settings.Default.ServerNames != null)

{

var savedNames = Properties.Settings.Default.ServerNames;

foreach (var item in savedNames)

{

this.comboBoxServerName.Items.Add(item);

}

}

if (Properties.Settings.Default.LoginNames != null)

{

var savedNames = Properties.Settings.Default.LoginNames;

foreach (var item in savedNames)

{

this.comboBoxLogin.Items.Add(item);

}

}

//The changed event on this combobox will trigger other values to load

comboBoxServerName.Text = Properties.Settings.Default.LastServerName;

}

private void buttonConnect\_Click(object sender, EventArgs e)

{

EnableOrDisableAllFields(false);

if (TestUserCredentials())

{

Properties.Settings.Default.Save();

this.Close();

}

else

{

EnableOrDisableAllFields(true);

EnableOrDisableFields(comboBoxAuthentication.Text.Equals(\_SQLAUTH));

this.DialogResult = DialogResult.None;

}

}

private bool TestUserCredentials()

{

string selectedServer = comboBoxServerName.Text.Trim();

string authType = comboBoxAuthentication.Text.Trim();

string authNum = \_LOGINTYPEWINDOWS;

if (authType.Equals(\_SQLAUTH))

authNum = \_LOGINTYPESQL;

string selectedLogin = comboBoxLogin.Text.Trim();

string concat = selectedServer + \_SERVERANDUSERSEPARATOR + authNum + selectedLogin;

if (!Properties.Settings.Default.ServerNames.Contains(selectedServer))

{

Properties.Settings.Default.ServerNames.Add(selectedServer);

}

if (!string.IsNullOrWhiteSpace(selectedLogin))

{

if (!Properties.Settings.Default.LoginNames.Contains(selectedLogin))

{

Properties.Settings.Default.LoginNames.Add(selectedLogin);

}

}

Properties.Settings.Default.LastServerName = selectedServer;

Properties.Settings.Default.LastLogin = selectedLogin;

Properties.Settings.Default.RememberChecked = checkBoxRememberPassword.Checked;

if (checkBoxRememberPassword.Checked)

{

string encryptedPassword = textBoxPassword.Text.Trim().Protect(selectedServer);

concat += \_USERANDPASSWORDSEPARATOR + encryptedPassword;

}

bool found = false;

foreach (var item in Properties.Settings.Default.ConcatStuff)

{

if (item.StartsWith(selectedServer + \_SERVERANDUSERSEPARATOR))

{

Properties.Settings.Default.ConcatStuff.Remove(item);

Properties.Settings.Default.ConcatStuff.Add(concat);

found = true;

break;

}

}

if (found == false)

{

Properties.Settings.Default.ConcatStuff.Add(concat);

}

string login = "Integrated Security=true";

if (authType.Equals(\_SQLAUTH))

{

login = $"user id={selectedLogin};password={textBoxPassword.Text.Trim()}";

}

return AttemptDatabaseConnection("Data Source=" + selectedServer + ";" + login, selectedServer);

}

private bool AttemptDatabaseConnection(string connString, string selectedServer)

{

try

{

using (System.Data.SqlClient.SqlConnection conn = new System.Data.SqlClient.SqlConnection(connString))

{

conn.Open();

ConnectionString = connString; //The code that calls this form should retrieve the connection string from this public property

}

#if DEBUG

MessageBox.Show("Connection string " + connString + " result: true");

#endif

}

catch (Exception ex)

{

MessageBox.Show("Cannot connect to " + selectedServer + "." + Environment.NewLine + Environment.NewLine + "Additional information: " + Environment.NewLine + ex.Message, "Connect to Server", MessageBoxButtons.OK);

return false;

}

return true;

}

private void comboBoxServerName\_SelectedIndexChanged(object sender, EventArgs e)

{

string selectedServer = comboBoxServerName.Text.Trim();

foreach (var item in Properties.Settings.Default.ConcatStuff)

{

if (item.StartsWith(selectedServer + \_SERVERANDUSERSEPARATOR))

PopulateDefaultsForSelectedServer(item.Replace(selectedServer + \_SERVERANDUSERSEPARATOR, ""));

}

if (string.IsNullOrWhiteSpace(comboBoxAuthentication.Text))

{

comboBoxAuthentication.Text = comboBoxAuthentication.Items[0].ToString();

}

}

private void PopulateDefaultsForSelectedServer(string concat)

{

string lastAuth = concat.Substring(0, 1);

if (lastAuth == \_LOGINTYPEWINDOWS)

{

comboBoxAuthentication.Text = \_WINDOWSAUTH;

}

else

{

comboBoxAuthentication.Text = \_SQLAUTH;

concat = concat.Substring(1);

int indexOfPassword = concat.IndexOf(\_USERANDPASSWORDSEPARATOR);

if (indexOfPassword >= 0)

{

comboBoxLogin.Text = concat.Substring(0, indexOfPassword);

string lastPwd = concat.Substring(indexOfPassword + \_USERANDPASSWORDSEPARATOR.Length);

textBoxPassword.Text = lastPwd.Unprotect(comboBoxServerName.Text.Trim());

checkBoxRememberPassword.Checked = true;

}

else

{

comboBoxLogin.Text = concat;

textBoxPassword.Text = "";

checkBoxRememberPassword.Checked = false;

}

}

}

private void comboBoxAuthentication\_SelectedIndexChanged(object sender, EventArgs e)

{

EnableDisableBasedOnAuthMethod(comboBoxAuthentication.Text.Equals(\_SQLAUTH));

}

#region Helper Methods

private static void InitializePropertiesForFirstTimeUsers()

{

if (Properties.Settings.Default.ServerNames == null)

{

Properties.Settings.Default.ServerNames = new System.Collections.Specialized.StringCollection();

}

if (Properties.Settings.Default.LoginNames == null)

{

Properties.Settings.Default.LoginNames = new System.Collections.Specialized.StringCollection();

}

if (Properties.Settings.Default.ConcatStuff == null)

{

Properties.Settings.Default.ConcatStuff = new System.Collections.Specialized.StringCollection();

}

}

private void EnableOrDisableAllFields(bool enable)

{

EnableOrDisableFields(enable);

comboBoxAuthentication.Enabled = enable;

comboBoxServerName.Enabled = enable;

labelAuthentication.Enabled = enable;

labelServerName.Enabled = enable;

comboBoxServerType.Enabled = enable;

labelServerType.Enabled = enable;

buttonConnect.Enabled = enable;

buttonCancel.Enabled = enable; //The real SQL Server UI allows you to cancel

}

private void EnableDisableBasedOnAuthMethod(bool sqlAuth)

{

EnableOrDisableFields(sqlAuth);

if (sqlAuth == false)

{

comboBoxLogin.Text = "";

textBoxPassword.Text = "";

}

}

private void EnableOrDisableFields(bool enable)

{

comboBoxLogin.Enabled = enable;

textBoxPassword.Enabled = enable;

checkBoxRememberPassword.Enabled = enable;

labelLogin.Enabled = enable;

labelPassword.Enabled = enable;

}

private void LoadSharedDefaultValues()

{

buttonHelp.Visible = false;

buttonOptions.Visible = false;

comboBoxAuthentication.Items.Add(\_WINDOWSAUTH);

comboBoxAuthentication.Items.Add(\_SQLAUTH);

comboBoxAuthentication.Text = \_WINDOWSAUTH;

comboBoxAuthentication.DropDownStyle = ComboBoxStyle.DropDownList;

comboBoxServerType.Items.Add("Database Engine");

comboBoxServerType.Text = "Database Engine";

comboBoxServerType.DropDownStyle = ComboBoxStyle.DropDownList;

}

#endregion

private void FormConnectToSQLServer\_Resize(object sender, EventArgs e)

{

int rightoffset = 64;

if (!\_highDPI)

{

rightoffset = 28;

this.Height = 320;

}

comboBoxServerType.Width = this.Width - comboBoxServerType.Left - rightoffset;

comboBoxServerName.Width = this.Width - comboBoxServerName.Left - rightoffset;

comboBoxLogin.Width = this.Width - comboBoxLogin.Left - rightoffset;

comboBoxAuthentication.Width = this.Width - comboBoxAuthentication.Left - rightoffset;

textBoxPassword.Width = this.Width - textBoxPassword.Left - rightoffset;

}

}

}

namespace SQLServerLoginTemplate

{

partial class FormConnectToSQLServer

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.labelServerName = new System.Windows.Forms.Label();

this.labelAuthentication = new System.Windows.Forms.Label();

this.labelLogin = new System.Windows.Forms.Label();

this.labelPassword = new System.Windows.Forms.Label();

this.comboBoxServerName = new System.Windows.Forms.ComboBox();

this.comboBoxLogin = new System.Windows.Forms.ComboBox();

this.textBoxPassword = new System.Windows.Forms.TextBox();

this.buttonConnect = new System.Windows.Forms.Button();

this.buttonCancel = new System.Windows.Forms.Button();

this.buttonHelp = new System.Windows.Forms.Button();

this.buttonOptions = new System.Windows.Forms.Button();

this.checkBoxRememberPassword = new System.Windows.Forms.CheckBox();

this.label1 = new System.Windows.Forms.Label();

this.labelSQLServer = new System.Windows.Forms.Label();

this.comboBoxAuthentication = new System.Windows.Forms.ComboBox();

this.labelServerType = new System.Windows.Forms.Label();

this.comboBoxServerType = new System.Windows.Forms.ComboBox();

this.panel1 = new System.Windows.Forms.Panel();

this.panel2 = new System.Windows.Forms.Panel();

this.panel1.SuspendLayout();

this.SuspendLayout();

//

// labelServerName

//

this.labelServerName.AutoSize = true;

this.labelServerName.Location = new System.Drawing.Point(13, 108);

this.labelServerName.Margin = new System.Windows.Forms.Padding(1, 0, 1, 0);

this.labelServerName.Name = "labelServerName";

this.labelServerName.Size = new System.Drawing.Size(105, 13);

this.labelServerName.TabIndex = 0;

this.labelServerName.Text = "&Название сервера:";

//

// labelAuthentication

//

this.labelAuthentication.AutoSize = true;

this.labelAuthentication.Location = new System.Drawing.Point(13, 136);

this.labelAuthentication.Margin = new System.Windows.Forms.Padding(1, 0, 1, 0);

this.labelAuthentication.Name = "labelAuthentication";

this.labelAuthentication.Size = new System.Drawing.Size(61, 13);

this.labelAuthentication.TabIndex = 1;

this.labelAuthentication.Text = "&Тип входа:";

//

// labelLogin

//

this.labelLogin.AutoSize = true;

this.labelLogin.Location = new System.Drawing.Point(33, 162);

this.labelLogin.Margin = new System.Windows.Forms.Padding(1, 0, 1, 0);

this.labelLogin.Name = "labelLogin";

this.labelLogin.Size = new System.Drawing.Size(41, 13);

this.labelLogin.TabIndex = 2;

this.labelLogin.Text = "&Логин:";

//

// labelPassword

//

this.labelPassword.AutoSize = true;

this.labelPassword.Location = new System.Drawing.Point(33, 187);

this.labelPassword.Margin = new System.Windows.Forms.Padding(1, 0, 1, 0);

this.labelPassword.Name = "labelPassword";

this.labelPassword.Size = new System.Drawing.Size(48, 13);

this.labelPassword.TabIndex = 3;

this.labelPassword.Text = "&Пароль:";

//

// comboBoxServerName

//

this.comboBoxServerName.FormattingEnabled = true;

this.comboBoxServerName.Location = new System.Drawing.Point(171, 105);

this.comboBoxServerName.Margin = new System.Windows.Forms.Padding(1);

this.comboBoxServerName.Name = "comboBoxServerName";

this.comboBoxServerName.Size = new System.Drawing.Size(306, 21);

this.comboBoxServerName.TabIndex = 6;

this.comboBoxServerName.SelectedIndexChanged += new System.EventHandler(this.comboBoxServerName\_SelectedIndexChanged);

//

// comboBoxLogin

//

this.comboBoxLogin.FormattingEnabled = true;

this.comboBoxLogin.Location = new System.Drawing.Point(191, 159);

this.comboBoxLogin.Margin = new System.Windows.Forms.Padding(1);

this.comboBoxLogin.Name = "comboBoxLogin";

this.comboBoxLogin.Size = new System.Drawing.Size(287, 21);

this.comboBoxLogin.TabIndex = 8;

//

// textBoxPassword

//

this.textBoxPassword.Location = new System.Drawing.Point(191, 184);

this.textBoxPassword.Margin = new System.Windows.Forms.Padding(1);

this.textBoxPassword.Name = "textBoxPassword";

this.textBoxPassword.PasswordChar = '\*';

this.textBoxPassword.Size = new System.Drawing.Size(287, 20);

this.textBoxPassword.TabIndex = 9;

//

// buttonConnect

//

this.buttonConnect.DialogResult = System.Windows.Forms.DialogResult.OK;

this.buttonConnect.Location = new System.Drawing.Point(112, 242);

this.buttonConnect.Margin = new System.Windows.Forms.Padding(1);

this.buttonConnect.Name = "buttonConnect";

this.buttonConnect.Size = new System.Drawing.Size(110, 21);

this.buttonConnect.TabIndex = 10;

this.buttonConnect.Text = "&Подключится";

this.buttonConnect.UseVisualStyleBackColor = true;

this.buttonConnect.Click += new System.EventHandler(this.buttonConnect\_Click);

//

// buttonCancel

//

this.buttonCancel.DialogResult = System.Windows.Forms.DialogResult.Cancel;

this.buttonCancel.Location = new System.Drawing.Point(228, 242);

this.buttonCancel.Margin = new System.Windows.Forms.Padding(1);

this.buttonCancel.Name = "buttonCancel";

this.buttonCancel.Size = new System.Drawing.Size(69, 21);

this.buttonCancel.TabIndex = 11;

this.buttonCancel.Text = "Отмена";

this.buttonCancel.UseVisualStyleBackColor = true;

this.buttonCancel.Click += new System.EventHandler(this.buttonCancel\_Click);

//

// buttonHelp

//

this.buttonHelp.Location = new System.Drawing.Point(302, 241);

this.buttonHelp.Margin = new System.Windows.Forms.Padding(1);

this.buttonHelp.Name = "buttonHelp";

this.buttonHelp.Size = new System.Drawing.Size(81, 22);

this.buttonHelp.TabIndex = 12;

this.buttonHelp.Text = "&Помощь";

this.buttonHelp.UseVisualStyleBackColor = true;

//

// buttonOptions

//

this.buttonOptions.Location = new System.Drawing.Point(385, 242);

this.buttonOptions.Margin = new System.Windows.Forms.Padding(1);

this.buttonOptions.Name = "buttonOptions";

this.buttonOptions.Size = new System.Drawing.Size(91, 21);

this.buttonOptions.TabIndex = 13;

this.buttonOptions.Text = "&Настройки >>";

this.buttonOptions.UseVisualStyleBackColor = true;

//

// checkBoxRememberPassword

//

this.checkBoxRememberPassword.AutoSize = true;

this.checkBoxRememberPassword.Location = new System.Drawing.Point(191, 212);

this.checkBoxRememberPassword.Margin = new System.Windows.Forms.Padding(1);

this.checkBoxRememberPassword.Name = "checkBoxRememberPassword";

this.checkBoxRememberPassword.Size = new System.Drawing.Size(121, 17);

this.checkBoxRememberPassword.TabIndex = 14;

this.checkBoxRememberPassword.Text = "Запомнить пароль";

this.checkBoxRememberPassword.UseVisualStyleBackColor = true;

//

// label1

//

this.label1.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;

this.label1.Location = new System.Drawing.Point(11, 230);

this.label1.Margin = new System.Windows.Forms.Padding(1, 0, 1, 0);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(463, 4);

this.label1.TabIndex = 15;

//

// labelSQLServer

//

this.labelSQLServer.AutoSize = true;

this.labelSQLServer.Font = new System.Drawing.Font("Microsoft Sans Serif", 22F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.labelSQLServer.Location = new System.Drawing.Point(126, 9);

this.labelSQLServer.Margin = new System.Windows.Forms.Padding(1, 0, 1, 0);

this.labelSQLServer.Name = "labelSQLServer";

this.labelSQLServer.Size = new System.Drawing.Size(204, 36);

this.labelSQLServer.TabIndex = 16;

this.labelSQLServer.Text = "Авторизация";

//

// comboBoxAuthentication

//

this.comboBoxAuthentication.FormattingEnabled = true;

this.comboBoxAuthentication.Location = new System.Drawing.Point(171, 133);

this.comboBoxAuthentication.Margin = new System.Windows.Forms.Padding(1);

this.comboBoxAuthentication.Name = "comboBoxAuthentication";

this.comboBoxAuthentication.Size = new System.Drawing.Size(306, 21);

this.comboBoxAuthentication.TabIndex = 7;

this.comboBoxAuthentication.SelectedIndexChanged += new System.EventHandler(this.comboBoxAuthentication\_SelectedIndexChanged);

//

// labelServerType

//

this.labelServerType.AutoSize = true;

this.labelServerType.Location = new System.Drawing.Point(13, 80);

this.labelServerType.Margin = new System.Windows.Forms.Padding(1, 0, 1, 0);

this.labelServerType.Name = "labelServerType";

this.labelServerType.Size = new System.Drawing.Size(74, 13);

this.labelServerType.TabIndex = 17;

this.labelServerType.Text = "Тип сервера:";

//

// comboBoxServerType

//

this.comboBoxServerType.FormattingEnabled = true;

this.comboBoxServerType.Location = new System.Drawing.Point(171, 80);

this.comboBoxServerType.Margin = new System.Windows.Forms.Padding(1);

this.comboBoxServerType.Name = "comboBoxServerType";

this.comboBoxServerType.Size = new System.Drawing.Size(306, 21);

this.comboBoxServerType.TabIndex = 18;

//

// panel1

//

this.panel1.BackColor = System.Drawing.Color.White;

this.panel1.Controls.Add(this.panel2);

this.panel1.Controls.Add(this.labelSQLServer);

this.panel1.Location = new System.Drawing.Point(-2, 0);

this.panel1.Name = "panel1";

this.panel1.Size = new System.Drawing.Size(503, 63);

this.panel1.TabIndex = 19;

//

// panel2

//

this.panel2.BackColor = System.Drawing.Color.DarkOrange;

this.panel2.Location = new System.Drawing.Point(0, 54);

this.panel2.Name = "panel2";

this.panel2.Size = new System.Drawing.Size(500, 10);

this.panel2.TabIndex = 17;

//

// FormConnectToSQLServer

//

this.AcceptButton = this.buttonConnect;

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.CancelButton = this.buttonCancel;

this.ClientSize = new System.Drawing.Size(484, 282);

this.Controls.Add(this.panel1);

this.Controls.Add(this.comboBoxServerType);

this.Controls.Add(this.labelServerType);

this.Controls.Add(this.label1);

this.Controls.Add(this.checkBoxRememberPassword);

this.Controls.Add(this.buttonOptions);

this.Controls.Add(this.buttonHelp);

this.Controls.Add(this.buttonCancel);

this.Controls.Add(this.buttonConnect);

this.Controls.Add(this.textBoxPassword);

this.Controls.Add(this.comboBoxLogin);

this.Controls.Add(this.comboBoxAuthentication);

this.Controls.Add(this.comboBoxServerName);

this.Controls.Add(this.labelPassword);

this.Controls.Add(this.labelLogin);

this.Controls.Add(this.labelAuthentication);

this.Controls.Add(this.labelServerName);

this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedDialog;

this.Margin = new System.Windows.Forms.Padding(1);

this.MaximizeBox = false;

this.MinimizeBox = false;

this.Name = "FormConnectToSQLServer";

this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;

this.Text = "Вход в систему";

this.Load += new System.EventHandler(this.FormConnectToSQLServer\_Load);

this.Resize += new System.EventHandler(this.FormConnectToSQLServer\_Resize);

this.panel1.ResumeLayout(false);

this.panel1.PerformLayout();

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.Label labelServerName;

private System.Windows.Forms.Label labelAuthentication;

private System.Windows.Forms.Label labelLogin;

private System.Windows.Forms.Label labelPassword;

private System.Windows.Forms.ComboBox comboBoxServerName;

private System.Windows.Forms.ComboBox comboBoxLogin;

private System.Windows.Forms.TextBox textBoxPassword;

private System.Windows.Forms.Button buttonConnect;

private System.Windows.Forms.Button buttonCancel;

private System.Windows.Forms.Button buttonHelp;

private System.Windows.Forms.Button buttonOptions;

private System.Windows.Forms.CheckBox checkBoxRememberPassword;

private System.Windows.Forms.Label label1;

private System.Windows.Forms.Label labelSQLServer;

private System.Windows.Forms.ComboBox comboBoxAuthentication;

private System.Windows.Forms.Label labelServerType;

private System.Windows.Forms.ComboBox comboBoxServerType;

private System.Windows.Forms.Panel panel1;

private System.Windows.Forms.Panel panel2;

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace SQLServerLoginTemplate

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new FormConnectToSQLServer());

}

}

}

namespace SQLServerLoginTemplate.Properties {

// This class allows you to handle specific events on the settings class:

// The SettingChanging event is raised before a setting's value is changed.

// The PropertyChanged event is raised after a setting's value is changed.

// The SettingsLoaded event is raised after the setting values are loaded.

// The SettingsSaving event is raised before the setting values are saved.

internal sealed partial class Settings {

public Settings() {

// // To add event handlers for saving and changing settings, uncomment the lines below:

//

// this.SettingChanging += this.SettingChangingEventHandler;

//

// this.SettingsSaving += this.SettingsSavingEventHandler;

//

}

private void SettingChangingEventHandler(object sender, System.Configuration.SettingChangingEventArgs e) {

// Add code to handle the SettingChangingEvent event here.

}

private void SettingsSavingEventHandler(object sender, System.ComponentModel.CancelEventArgs e) {

// Add code to handle the SettingsSaving event here.

}

}

}

using System;

using System.ComponentModel;

using System.Drawing;

using System.Globalization;

using System.Media;

using System.Runtime.CompilerServices;

using System.Runtime.InteropServices;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Interop;

using System.Windows.Media;

using System.Windows.Media.Imaging;

namespace MsgBoxEx

{

// The MessageBoxEx class we getting kinda crowded, so I took advantage of the fact that it's a

// partial class, and segregated the static from the non-static. The code looks less chatotic

// and is easier to maintain as a result.

/// <summary>

/// Non-static interaction logic for MessageBoxEx.xaml

/// </summary>

public partial class MessageBoxEx : Window, INotifyPropertyChanged

{

#region INotifyPropertyChanged

private bool isModified = false;

public bool IsModified { get { return this.isModified; } set { if (value != this.isModified) { this.isModified = true; this.NotifyPropertyChanged(); } } }

/// <summary>

/// Occurs when a property value changes.

/// </summary>

public event PropertyChangedEventHandler PropertyChanged;

/// <summary>

/// Notifies that the property changed, and sets IsModified to true.

/// </summary>

/// <param name="propertyName">Name of the property.</param>

protected void NotifyPropertyChanged([CallerMemberName] String propertyName = "")

{

if (this.PropertyChanged != null)

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

if (propertyName != "IsModified")

{

this.IsModified = true;

}

}

}

#endregion INotifyPropertyChanged

// so we can run the browser when the optional URL is clicked

[DllImport("shell32.dll")]

public static extern IntPtr ShellExecute(IntPtr hwnd, string lpOperation

, string lpFile, string lpParameters

, string lpDirectory, int nShowCmd);

private const string \_DEFAULT\_CAPTION = "Application Message";

#region fields

private double screenHeight;

private string message;

private string messageTitle;

private MessageBoxButton? buttons;

private MessageBoxResult messageResult;

private MessageBoxButtonEx? buttonsEx;

private MessageBoxResultEx messageResultEx;

private ImageSource messageIcon;

private MessageBoxImage msgBoxImage;

private double buttonWidth = 0d;

private bool expanded = false;

private bool isDefaultOK;

private bool isDefaultCancel;

private bool isDefaultYes;

private bool isDefaultNo;

private bool isDefaultAbort;

private bool isDefaultRetry;

private bool isDefaultIgnore;

private bool usingExButtons = false;

#endregion fields

#region properties

/// <summary>

/// Get/set the screen's work area height

/// </summary>

public double ScreenHeight { get { return this.screenHeight; } set { if (value != this.screenHeight) { this.screenHeight = value; this.NotifyPropertyChanged(); } } }

/// <summary>

/// Get/set the message text

/// </summary>

public string Message { get { return this.message; } set { if (value != this.message) { this.message = value; this.NotifyPropertyChanged(); } } }

/// <summary>

/// Get/set the form caption

/// </summary>

public string MessageTitle { get { return this.messageTitle; } set { if (value != this.messageTitle) { this.messageTitle = value; this.NotifyPropertyChanged(); } } }

/// <summary>

/// Get/set the message box result (which button was pressed to dismiss the form)

/// </summary>

public MessageBoxResult MessageResult { get { return this.messageResult; } set { this.messageResult = value; } }

public MessageBoxResultEx MessageResultEx { get { return this.messageResultEx; } set { this.messageResultEx = value; } }

/// <summary>

/// Get/set the buttons ued in the form (and update visibility for them)

/// </summary>

public MessageBoxButton? Buttons

{

get { return this.buttons; }

set

{

if (value != this.buttons)

{

this.buttons = value;

this.NotifyPropertyChanged();

this.NotifyPropertyChanged("ShowOk");

this.NotifyPropertyChanged("ShowCancel");

this.NotifyPropertyChanged("ShowYes");

this.NotifyPropertyChanged("ShowNo");

}

}

}

public MessageBoxButtonEx? ButtonsEx

{

get { return this.buttonsEx; }

set

{

if (value != this.buttonsEx)

{

this.buttonsEx = value;

this.NotifyPropertyChanged();

this.NotifyPropertyChanged("ShowOk");

this.NotifyPropertyChanged("ShowCancel");

this.NotifyPropertyChanged("ShowYes");

this.NotifyPropertyChanged("ShowNo");

this.NotifyPropertyChanged("ShowAbort");

this.NotifyPropertyChanged("ShowRetry");

this.NotifyPropertyChanged("ShowIgnore");

}

}

}

/// <summary>

/// Get the visibility of the ok button

/// </summary>

public Visibility ShowOk

{

get

{

return (!this.usingExButtons && this.Buttons == MessageBoxButton.OK ||

!this.usingExButtons && this.Buttons == MessageBoxButton.OKCancel ||

this.usingExButtons && this.ButtonsEx == MessageBoxButtonEx.OK ||

this.usingExButtons && this.ButtonsEx == MessageBoxButtonEx.OKCancel) ? Visibility.Visible : Visibility.Collapsed;

}

}

/// <summary>

/// Get the visibility of the cancel button

/// </summary>

public Visibility ShowCancel

{

get

{

return (!this.usingExButtons && this.Buttons == MessageBoxButton.OKCancel ||

!this.usingExButtons && this.Buttons == MessageBoxButton.YesNoCancel ||

this.usingExButtons && this.ButtonsEx == MessageBoxButtonEx.OKCancel ||

this.usingExButtons && this.ButtonsEx == MessageBoxButtonEx.YesNoCancel ||

this.usingExButtons && this.ButtonsEx == MessageBoxButtonEx.RetryCancel) ? Visibility.Visible : Visibility.Collapsed;

}

}

/// <summary>

/// Get the visibility of the yes button

/// </summary>

public Visibility ShowYes

{

get

{

return (!this.usingExButtons && this.Buttons == MessageBoxButton.YesNo ||

!this.usingExButtons && this.Buttons == MessageBoxButton.YesNoCancel ||

this.usingExButtons && this.ButtonsEx == MessageBoxButtonEx.YesNo ||

this.usingExButtons && this.ButtonsEx == MessageBoxButtonEx.YesNoCancel) ? Visibility.Visible : Visibility.Collapsed;

}

}

/// <summary>

/// Get the visibility of the no button

/// </summary>

public Visibility ShowNo

{

get

{

return (!this.usingExButtons && this.Buttons == MessageBoxButton.YesNo ||

!this.usingExButtons && this.Buttons == MessageBoxButton.YesNoCancel ||

this.usingExButtons && this.ButtonsEx == MessageBoxButtonEx.YesNo ||

this.usingExButtons && this.ButtonsEx == MessageBoxButtonEx.YesNoCancel) ? Visibility.Visible : Visibility.Collapsed;

}

}

/// <summary>

/// Get the visibility of the retry button

/// </summary>

public Visibility ShowRetry

{

get

{

return (this.usingExButtons && this.ButtonsEx == MessageBoxButtonEx.AbortRetryIgnore ||

this.usingExButtons && this.ButtonsEx == MessageBoxButtonEx.RetryCancel) ? Visibility.Visible : Visibility.Collapsed;

}

}

/// <summary>

/// Get the visibility of the abort button

/// </summary>

public Visibility ShowAbort { get { return (this.usingExButtons && this.ButtonsEx == MessageBoxButtonEx.AbortRetryIgnore) ? Visibility.Visible : Visibility.Collapsed; } }

/// <summary>

/// Get the visibility of the ignore button

/// </summary>

public Visibility ShowIgnore { get { return (this.usingExButtons && this.ButtonsEx == MessageBoxButtonEx.AbortRetryIgnore) ? Visibility.Visible : Visibility.Collapsed; } }

/// <summary>

/// Get this visibility of the message box icon

/// </summary>

public Visibility ShowIcon { get { return (this.MessageIcon != null) ? Visibility.Visible : Visibility.Collapsed; } }

/// <summary>

/// Get/set the icon specified by the user

/// </summary>

public ImageSource MessageIcon { get { return this.messageIcon; } set { if (value != this.messageIcon) { this.messageIcon = value; this.NotifyPropertyChanged(); } } }

/// <summary>

/// Get/set the width of the largest button (so all buttons are the same width as the widest button)

/// </summary>

public double ButtonWidth { get { return this.buttonWidth; } set { if (value != this.buttonWidth) { this.buttonWidth = value; this.NotifyPropertyChanged(); } } }

/// <summary>

/// Get/set the flag inidcating whether the expander is expanded

/// </summary>

public bool Expanded { get { return this.expanded; } set { if (value != expanded) { this.expanded = value; this.NotifyPropertyChanged(); } } }

// default button flags

/// <summary>

/// Get/set the flag indicating whether OK is the default button

/// </summary>

public bool IsDefaultOK { get { return this.isDefaultOK; } set { if (value != this.isDefaultOK) { this.isDefaultOK = value; this.NotifyPropertyChanged(); } } }

/// <summary>

/// Get/set the flag indicating whether Cancel is the default button

/// </summary>

public bool IsDefaultCancel { get { return this.isDefaultCancel; } set { if (value != this.isDefaultCancel) { this.isDefaultCancel = value; this.NotifyPropertyChanged(); } } }

/// <summary>

/// Get/set the flag indicating whether Yes is the default button

/// </summary>

public bool IsDefaultYes { get { return this.isDefaultYes; } set { if (value != this.isDefaultYes) { this.isDefaultYes = value; this.NotifyPropertyChanged(); } } }

/// <summary>

/// Get/set the flag indicating whether No is the default button

/// </summary>

public bool IsDefaultNo { get { return this.isDefaultNo; } set { if (value != this.isDefaultNo) { this.isDefaultNo = value; this.NotifyPropertyChanged(); } } }

/// <summary>

/// Get/set the flag indicating whether Abort is the default button

/// </summary>

public bool IsDefaultAbort { get { return this.isDefaultAbort; } set { if (value != this.isDefaultAbort) { this.isDefaultAbort = value; this.NotifyPropertyChanged(); } } }

/// <summary>

/// Get/set the flag indicating whether Retry is the default button

/// </summary>

public bool IsDefaultRetry { get { return this.isDefaultRetry; } set { if (value != this.isDefaultRetry) { this.isDefaultRetry = value; this.NotifyPropertyChanged(); } } }

/// <summary>

/// Get/set the flag indicating whether Ignore is the default button

/// </summary>

public bool IsDefaultIgnore { get { return this.isDefaultIgnore; } set { if (value != this.isDefaultIgnore) { this.isDefaultIgnore = value; this.NotifyPropertyChanged(); } } }

#endregion properties

#region constructors

/// <summary>

/// Default constructor for VS designer

/// </summary>

private MessageBoxEx()

{

this.InitializeComponent();

this.DataContext = this;

this.LargestButtonWidth();

}

/// <summary>

/// Constructor for standard buttons

/// </summary>

/// <param name="msg"></param>

/// <param name="title"></param>

/// <param name="buttons">(Optinal) Message box button(s) to be displayed (default = OK)</param>

/// <param name="image">(Optional) Message box image to display (default = None)</param>

public MessageBoxEx(string msg, string title, MessageBoxButton buttons = MessageBoxButton.OK, MessageBoxImage image = MessageBoxImage.None)

{

this.InitializeComponent();

this.DataContext = this;

this.Init(msg, title, buttons, image);

}

/// <summary>

/// Constructor for extended buttons

/// </summary>

/// <param name="msg"></param>

/// <param name="title"></param>

/// <param name="buttons">(Optinal) Message box button(s) to be displayed (default = OK)</param>

/// <param name="image">(Optional) Message box image to display (default = None)</param>

public MessageBoxEx(string msg, string title, MessageBoxButtonEx buttons = MessageBoxButtonEx.OK, MessageBoxImage image = MessageBoxImage.None)

{

this.InitializeComponent();

this.DataContext = this;

this.Init(msg, title, buttons, image);

}

#endregion constructors

#region non-static methods

/// <summary>

/// Performs message box initialization when using standard message box buttons

/// </summary>

/// <param name="msg">The message to display</param>

/// <param name="title">Window title</param>

/// <param name="buttons">What buttons are to be displayed</param>

/// <param name="image">What message box icon image is to be displayed</param>

protected virtual void Init(string msg, string title, MessageBoxButton buttons, MessageBoxImage image)

{

this.InitTop(msg, title);

this.usingExButtons = false;

this.ButtonsEx = null;

this.Buttons = buttons;

this.SetButtonTemplates();

this.InitBottom(image);

this.FindDefaultButton(staticButtonDefault);

}

/// <summary>

/// Performs message box initialization when using extended message box buttons

/// </summary>

/// <param name="msg">The message to display</param>

/// <param name="title">Window title</param>

/// <param name="buttons">What buttons are to be displayed</param>

/// <param name="image">What message box icon image is to be displayed</param>

protected virtual void Init(string msg, string title, MessageBoxButtonEx buttons, MessageBoxImage image)

{

this.InitTop(msg, title);

this.usingExButtons = true;

this.Buttons = null;

this.ButtonsEx = buttons;

this.SetButtonTemplates();

this.InitBottom(image);

this.FindDefaultButtonEx(staticButtonDefault);

}

/// <summary>

/// Init the common stuff BEFORE buttons are processed

/// </summary>

/// <param name="msg"></param>

/// <param name="title"></param>

private void InitTop(string msg, string title)

{

// determine whether or not to show the details pane and checkbox

ShowDetailsBtn = (string.IsNullOrEmpty(DetailsText)) ? Visibility.Collapsed : Visibility.Visible;

ShowCheckBox = (CheckBoxData == null) ? Visibility.Collapsed : Visibility.Visible;

// Well, the binding for family/size don't appear to be working, so I have to set them

// manually. Weird...

this.FontFamily = MsgFontFamily;

this.FontSize = MsgFontSize;

this.LargestButtonWidth();

// determine the screen area height, and the height of the textblock

this.ScreenHeight = SystemParameters.WorkArea.Height - 150;

// configure the form based on specified criteria

this.Message = msg;

this.MessageTitle = (string.IsNullOrEmpty(title.Trim())) ? \_DEFAULT\_CAPTION : title;

// url (if specified)

if (Url != null)

{

this.tbUrl.Text = (string.IsNullOrEmpty(UrlDisplayName)) ? Url.ToString() : UrlDisplayName;

this.tbUrl.ToolTip = new ToolTip() { Content = Url.ToString() };

}

}

/// <summary>

/// Init common stuff AFTER buttons are processed

/// </summary>

/// <param name="image"></param>

private void InitBottom(MessageBoxImage image)

{

// set the form's colors (you can also set these colors in your program's startup code

// (either in app.xaml.cs or MainWindow.cs) before you use the MessageBox for the

// first time

MessageBackground = (MessageBackground == null) ? new SolidColorBrush(Colors.White) : MessageBackground;

MessageForeground = (MessageForeground == null) ? new SolidColorBrush(Colors.Black) : MessageForeground;

ButtonBackground = (ButtonBackground == null) ? new SolidColorBrush(ColorFromString("#cdcdcd")) : ButtonBackground;

this.MessageIcon = null;

this.msgBoxImage = image;

if (DelegateObj != null)

{

Style style = (Style)(this.FindResource("ImageOpacityChanger"));

if (style != null)

{

this.imgMsgBoxIcon.Style = style;

if (!string.IsNullOrEmpty(DelegateToolTip))

{

ToolTip tooltip = new ToolTip() { Content = DelegateToolTip };

// for some reason, Image elements can't do tooltips, so I assign the tootip

// to the parent grid. This seems to work fine.

this.imgGrid.ToolTip = tooltip;

}

}

}

// multiple images have the same ordinal value, and are indicated in the comments below.

// WTF Microsoft?

switch ((int)image)

{

case 16: // MessageBoxImage.Error, MessageBoxImage.Stop, MessageBox.Image.Hand

{

this.MessageIcon = this.GetIcon(SystemIcons.Error);

if (!isSilent) { SystemSounds.Hand.Play(); }

}

break;

case 64: // MessageBoxImage.Information, MessageBoxImage.Asterisk

{

this.MessageIcon = this.GetIcon(SystemIcons.Information);

if (!isSilent) { SystemSounds.Asterisk.Play(); }

}

break;

case 32: // MessageBoxImage.Question

{

this.MessageIcon = this.GetIcon(SystemIcons.Question);

if (!isSilent) { SystemSounds.Question.Play(); }

}

break;

case 48: // MessageBoxImage.Warning, MessageBoxImage.Exclamation

{

this.MessageIcon = this.GetIcon(SystemIcons.Warning);

if (!isSilent) { SystemSounds.Exclamation.Play(); }

}

break;

default:

this.MessageIcon = null;

break;

}

}

/// <summary>

/// Converts the specified icon into a WPF-comptible ImageSource object.

/// </summary>

/// <param name="icon"></param>

/// <returns>An ImageSource object that represents the specified icon.</returns>

public ImageSource GetIcon(System.Drawing.Icon icon)

{

var image = System.Windows.Interop.Imaging.CreateBitmapSourceFromHIcon(icon.Handle, Int32Rect.Empty, BitmapSizeOptions.FromEmptyOptions());

return image;

}

// The form is rendered and position BEFORE the SizeToContent property takes effect,

// so we have to take stepts to re-center it after the size changes. This code takes care

// of the re-positioning, and is called from the SizeChanged event handler.

/// <summary>

/// Center the form on the screen.

/// </summary>

protected virtual void CenterInScreen()

{

double width = this.ActualWidth;

double height = this.ActualHeight;

this.Left = (SystemParameters.WorkArea.Width - width) / 2 + SystemParameters.WorkArea.Left;

this.Top = (SystemParameters.WorkArea.Height - height) / 2 + SystemParameters.WorkArea.Top;

}

/// <summary>

/// Calculate the width of the largest button.

/// </summary>

protected void LargestButtonWidth()

{

// we base the width on the width of the content. This allows us to avoid the problems

// with button width/actualwidth properties, especially when a given button is

// Collapsed.

Typeface typeface = new Typeface(this.FontFamily, this.FontStyle, this.FontWeight, this.FontStretch);

StackPanel panel = (StackPanel)this.stackButtons.Child;

double width = 0;

string largestName = string.Empty;

foreach (Button button in panel.Children)

{

// Using the FormattedText object

// will strip whitespace before measuring the text, so we convert spaces to double

// hyphens to compensate (I like to pad button Content with a leading and trailing

// space) so that the button is wide enough to present a more padded appearance.

FormattedText formattedText = new FormattedText((button.Name == "btnDetails") ? "--Details--" : ((string)(button.Content)).Replace(" ", "--"),

CultureInfo.CurrentUICulture,

FlowDirection.LeftToRight,

typeface,

FontSize = this.FontSize,

System.Windows.Media.Brushes.Black,

VisualTreeHelper.GetDpi(this).PixelsPerDip);

if (width < formattedText.Width)

{

largestName = button.Name;

}

width = Math.Max(width, formattedText.Width);

}

this.ButtonWidth = Math.Ceiling(width/\*width + polyArrow.Width+polyArrow.Margin.Right+Margin.Left\*/);

}

/// <summary>

/// Sets the custom button template if necessary and possible

/// </summary>

private void SetButtonTemplates()

{

// set the button template (if specified)

if (!string.IsNullOrEmpty(ButtonTemplateName))

{

bool foundResource = true;

try

{

this.FindResource(ButtonTemplateName);

}

catch (Exception)

{

foundResource = false;

}

if (foundResource)

{

this.btnOK.SetResourceReference(Control.TemplateProperty, ButtonTemplateName);

this.btnYes.SetResourceReference(Control.TemplateProperty, ButtonTemplateName);

this.btnNo.SetResourceReference(Control.TemplateProperty, ButtonTemplateName);

this.btnCancel.SetResourceReference(Control.TemplateProperty, ButtonTemplateName);

this.btnAbort.SetResourceReference(Control.TemplateProperty, ButtonTemplateName);

this.btnRetry.SetResourceReference(Control.TemplateProperty, ButtonTemplateName);

this.btnIgnore.SetResourceReference(Control.TemplateProperty, ButtonTemplateName);

}

}

}

/// <summary>

/// Find the default button based on the extended buttons displayed, and the default button specified

/// </summary>

/// <param name="buttonDefault"></param>

private void FindDefaultButtonEx(MessageBoxButtonDefault buttonDefault)

{

// determine default button

this.IsDefaultOK = false;

this.IsDefaultCancel = false;

this.IsDefaultYes = false;

this.IsDefaultNo = false;

this.IsDefaultAbort = false;

this.IsDefaultRetry = false;

this.IsDefaultIgnore = false;

if (buttonDefault != MessageBoxButtonDefault.None)

{

switch (this.ButtonsEx)

{

case MessageBoxButtonEx.OK: this.IsDefaultOK = true; break;

case MessageBoxButtonEx.OKCancel:

{

switch (buttonDefault)

{

case MessageBoxButtonDefault.Button1:

case MessageBoxButtonDefault.OK:

case MessageBoxButtonDefault.MostPositive: this.IsDefaultOK = true; break;

case MessageBoxButtonDefault.Button2:

case MessageBoxButtonDefault.Cancel:

case MessageBoxButtonDefault.LeastPositive: this.IsDefaultCancel = true; break;

// windows.forms.messagebox default

case MessageBoxButtonDefault.Forms:

default: this.IsDefaultOK = true; break;

}

}

break;

case MessageBoxButtonEx.YesNoCancel:

{

switch (buttonDefault)

{

case MessageBoxButtonDefault.Button1:

case MessageBoxButtonDefault.Yes: break;

case MessageBoxButtonDefault.MostPositive: this.IsDefaultYes = true; break;

case MessageBoxButtonDefault.Button2:

case MessageBoxButtonDefault.No: this.IsDefaultNo = true; break;

case MessageBoxButtonDefault.Button3:

case MessageBoxButtonDefault.Cancel:

case MessageBoxButtonDefault.LeastPositive: this.IsDefaultCancel = true; break;

case MessageBoxButtonDefault.Forms:

default: this.IsDefaultYes = true; break;

}

}

break;

case MessageBoxButtonEx.YesNo:

{

switch (buttonDefault)

{

case MessageBoxButtonDefault.Button1:

case MessageBoxButtonDefault.Yes:

case MessageBoxButtonDefault.MostPositive: this.IsDefaultYes = true; break;

case MessageBoxButtonDefault.Button2:

case MessageBoxButtonDefault.No:

case MessageBoxButtonDefault.LeastPositive: this.IsDefaultNo = true; break;

case MessageBoxButtonDefault.Forms:

default: this.IsDefaultYes = true; break;

}

}

break;

case MessageBoxButtonEx.RetryCancel:

{

switch (buttonDefault)

{

case MessageBoxButtonDefault.Button1:

case MessageBoxButtonDefault.Retry:

case MessageBoxButtonDefault.MostPositive: this.IsDefaultRetry = true; break;

case MessageBoxButtonDefault.Button2:

case MessageBoxButtonDefault.Cancel:

case MessageBoxButtonDefault.LeastPositive: this.IsDefaultCancel = true; break;

case MessageBoxButtonDefault.Forms:

default: this.IsDefaultRetry = true; break;

}

}

break;

case MessageBoxButtonEx.AbortRetryIgnore:

{

switch (buttonDefault)

{

case MessageBoxButtonDefault.Button1:

case MessageBoxButtonDefault.Abort:

case MessageBoxButtonDefault.LeastPositive: this.IsDefaultAbort = true; break;

case MessageBoxButtonDefault.Button2:

case MessageBoxButtonDefault.Retry: this.IsDefaultRetry = true; break;

case MessageBoxButtonDefault.Button3:

case MessageBoxButtonDefault.Ignore:

case MessageBoxButtonDefault.MostPositive: this.IsDefaultIgnore = true; break;

case MessageBoxButtonDefault.Forms:

default: this.IsDefaultAbort = true; break;

}

}

break;

}

}

}

/// <summary>

/// Find the default button based on the standard buttons displayed, and the default button specified

/// </summary>

/// <param name="buttonDefault"></param>

private void FindDefaultButton(MessageBoxButtonDefault buttonDefault)

{

// determine default button

this.IsDefaultOK = false;

this.IsDefaultCancel = false;

this.IsDefaultYes = false;

this.IsDefaultNo = false;

this.IsDefaultAbort = false;

this.IsDefaultRetry = false;

this.IsDefaultIgnore = false;

if (buttonDefault != MessageBoxButtonDefault.None)

{

switch (this.Buttons)

{

case MessageBoxButton.OK: this.IsDefaultOK = true; break;

case MessageBoxButton.OKCancel:

{

switch (buttonDefault)

{

case MessageBoxButtonDefault.Button1:

case MessageBoxButtonDefault.OK:

case MessageBoxButtonDefault.MostPositive: this.IsDefaultOK = true; break;

case MessageBoxButtonDefault.Button2:

case MessageBoxButtonDefault.Cancel:

case MessageBoxButtonDefault.LeastPositive: this.IsDefaultCancel = true; break;

// windows.forms.messagebox default

case MessageBoxButtonDefault.Forms:

default: this.IsDefaultOK = true; break;

}

}

break;

case MessageBoxButton.YesNoCancel:

{

switch (buttonDefault)

{

case MessageBoxButtonDefault.Button1:

case MessageBoxButtonDefault.Yes: break;

case MessageBoxButtonDefault.MostPositive: this.IsDefaultYes = true; break;

case MessageBoxButtonDefault.Button2:

case MessageBoxButtonDefault.No: this.IsDefaultNo = true; break;

case MessageBoxButtonDefault.Button3:

case MessageBoxButtonDefault.Cancel:

case MessageBoxButtonDefault.LeastPositive: this.IsDefaultCancel = true; break;

case MessageBoxButtonDefault.Forms:

default: this.IsDefaultYes = true; break;

}

}

break;

case MessageBoxButton.YesNo:

{

switch (buttonDefault)

{

case MessageBoxButtonDefault.Button1:

case MessageBoxButtonDefault.Yes:

case MessageBoxButtonDefault.MostPositive: this.IsDefaultYes = true; break;

case MessageBoxButtonDefault.Button2:

case MessageBoxButtonDefault.No:

case MessageBoxButtonDefault.LeastPositive: this.IsDefaultNo = true; break;

case MessageBoxButtonDefault.Forms:

default: this.IsDefaultYes = true; break;

}

}

break;

}

}

}

#endregion non-static methods

////////////////////////////////////////////////////////////////////////////////////////////

// Form events

////////////////////////////////////////////////////////////////////////////////////////////

#region event handlers

#region buttons

/// <summary>

/// Handle the click event for the OK button

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

this.MessageResult = MessageBoxResult.OK;

this.MessageResultEx = MessageBoxResultEx.OK;

this.DialogResult = true;

}

/// <summary>

/// Handle the click event for the Yes button

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

this.MessageResult = MessageBoxResult.Yes;

this.MessageResultEx = MessageBoxResultEx.Yes;

this.DialogResult = true;

}

/// <summary>

/// Handle the click event for the No button

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

this.MessageResult = MessageBoxResult.No;

this.MessageResultEx = MessageBoxResultEx.No;

this.DialogResult = true;

}

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

{

this.MessageResult = MessageBoxResult.None;

this.MessageResultEx = MessageBoxResultEx.Abort;

this.DialogResult = true;

}

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

{

this.MessageResult = MessageBoxResult.None;

this.MessageResultEx = MessageBoxResultEx.Retry;

this.DialogResult = true;

}

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

{

this.MessageResult = MessageBoxResult.None;

this.MessageResultEx = MessageBoxResultEx.Ignore;

this.DialogResult = true;

}

/// <summary>

/// Handle the click event for the Cancel button

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

this.MessageResult = MessageBoxResult.Cancel;

this.MessageResultEx = MessageBoxResultEx.Cancel;

this.DialogResult = true;

}

#endregion buttons

/// <summary>

/// Handle the size changed event so we can re-center the form on the screen

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void NotifiableWindow\_SizeChanged(object sender, SizeChangedEventArgs e)

{

// we have to do this because the SizeToContent property is evaluated AFTER the window

// is positioned.

this.CenterInScreen();

}

/// <summary>

/// Handle the window loaded event

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

// if this in an error message box, this tooltip will be displayed. The intent is to set

// this value one time, and use it throughout the application session. However, you can

// certainly set it before displaying the messagebox to something that is contextually

// appropriate, but you'll have to clear it or reset it each time you use the MessageBox.

this.imgMsgBoxIcon.ToolTip = (this.msgBoxImage == MessageBoxImage.Error) ? MsgBoxIconToolTip : null;

}

/// <summary>

/// Handles the window closing event

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Window\_Closing(object sender, CancelEventArgs e)

{

// we always clear the details text and checkbox data.

DetailsText = null;

CheckBoxData = null;

// reset the default button to Forms.

staticButtonDefault = MessageBoxButtonDefault.Forms;

// if the user didn't click a button to close the form, we set the MessageResult to the

// most negative button value that was available.

if (this.MessageResult == MessageBoxResult.None)

{

if (usingExButtons)

{

switch (this.ButtonsEx)

{

case MessageBoxButtonEx.OK: this.MessageResultEx = MessageBoxResultEx.OK; break;

case MessageBoxButtonEx.YesNoCancel:

case MessageBoxButtonEx.OKCancel:

case MessageBoxButtonEx.RetryCancel:

case MessageBoxButtonEx.AbortRetryIgnore: this.MessageResultEx = MessageBoxResultEx.Cancel; break;

case MessageBoxButtonEx.YesNo: this.MessageResultEx = MessageBoxResultEx.No; break;

}

}

else

{

switch (this.Buttons)

{

case MessageBoxButton.OK: this.MessageResult = MessageBoxResult.OK; break;

case MessageBoxButton.YesNoCancel:

case MessageBoxButton.OKCancel: this.MessageResult = MessageBoxResult.Cancel; break;

case MessageBoxButton.YesNo: this.MessageResult = MessageBoxResult.No; break;

}

}

}

}

/// <summary>

/// Since an icon isn't a button, we have to look for the left-mouse-up event to know it's

/// been clicked

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void ImgMsgBoxIcon\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

// we only want to allow the click if this is an error message, and the delegate

// object has been specified.

if (DelegateObj != null && this.msgBoxImage == MessageBoxImage.Error && this.Buttons == MessageBoxButton.OK)

{

DelegateObj.PerformAction(this.Message);

//despite the result of the method, we close this message

if (ExitAfterErrorAction)

{

// make it like the user clicked the titlebar close button

this.MessageResult = MessageBoxResult.None;

this.DialogResult = true;

}

}

}

/// <summary>

/// Handle the left mouse up event for the url textblock

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void TbUrl\_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

ShellExecute(IntPtr.Zero, "open", Url.ToString(), "", "", 5);

}

// disables close button

[DllImport("user32.dll")]

private static extern IntPtr GetSystemMenu(IntPtr hWnd, bool bRevert);

[DllImport("user32.dll")]

private static extern bool EnableMenuItem(IntPtr hMenu, uint uIDEnableItem, uint uEnable);

private const uint MF\_BYCOMMAND = 0x00000000;

private const uint MF\_GRAYED = 0x00000001;

private const uint SC\_CLOSE = 0xF060;

private const int WM\_SHOWWINDOW = 0x00000018;

private void Window\_SourceInitialized(object sender, EventArgs e)

{

if (!enableCloseButton)

{

var hWnd = new WindowInteropHelper(this);

var sysMenu = GetSystemMenu(hWnd.Handle, false);

EnableMenuItem(sysMenu, SC\_CLOSE, MF\_BYCOMMAND | MF\_GRAYED);

}

}

#endregion event handlers

}

}

// if you insist on using the deprecated show overloads, uncomment the next line.

//#define \_\_OVERLOADED\_SHOW\_\_

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing.Text;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

namespace MsgBoxEx

{

/// <summary>

/// Dispays an configurable Message Box.

/// </summary>

public partial class MessageBoxEx : Window, INotifyPropertyChanged

{

#region static fields

private static double screenWidth = SystemParameters.WorkArea.Width - 100;

private static bool enableCloseButton = true;

private static ContentControl parentWindow;

private static string buttonTemplateName;

private static SolidColorBrush messageBackground;

private static SolidColorBrush messageForeground;

private static SolidColorBrush buttonBackground;

private static double maxFormWidth = screenWidth;

private static bool isSilent = false;

private static Visibility showDetailsBtn = Visibility.Collapsed;

private static string detailsText;

private static Visibility showCheckBox = Visibility.Collapsed;

private static MsgBoxExCheckBoxData checkBoxData = null;

private static System.Windows.Media.FontFamily msgFontFamily = new System.Windows.Media.FontFamily("Segoe UI");

private static double msgFontSize = 12;

private static Uri url = null;

private static Visibility showUrl = Visibility.Collapsed;

private static string urlDisplayName = null;

private static SolidColorBrush urlForeground = new SolidColorBrush(DefaultUrlForegroundColor);

private static string delegateToolTip;

private static List<string> installedFonts = new List<string>();

public static MessageBoxButtonDefault staticButtonDefault;

#endregion static fields

#region static properties

public static System.Windows.Media.Color DefaultUrlForegroundColor { get { return System.Windows.Media.Colors.Blue; } }

/// <summary>

/// Get/set the icon tooltip text

/// </summary>

private static string MsgBoxIconToolTip { get; set; }

/// <summary>

/// Get/set the external icon delegate object

/// </summary>

protected static MsgBoxExDelegate DelegateObj { get; set; }

/// <summary>

/// Get/set the flag that indicates whether the parent messagebox is closed after the

/// external action is finished.

/// </summary>

protected static bool ExitAfterErrorAction { get; set; }

/// <summary>

/// Get/set the parent content control

/// </summary>

public static ContentControl ParentWindow { get { return parentWindow; } set { parentWindow = value; } }

/// <summary>

/// Get/set the button template name (for styling buttons)

/// </summary>

public static string ButtonTemplateName { get { return buttonTemplateName; } set { buttonTemplateName = value; } }

/// <summary>

/// Get/set the brush for the message text background

/// </summary>

public static SolidColorBrush MessageBackground { get { return messageBackground; } set { messageBackground = value; } }

/// <summary>

/// Get/set the brush for the message text foreground

/// </summary>

public static SolidColorBrush MessageForeground { get { return messageForeground; } set { messageForeground = value; } }

/// <summary>

/// Get/set the brush for the button panel background

/// </summary>

public static SolidColorBrush ButtonBackground { get { return buttonBackground; } set { buttonBackground = value; } }

/// <summary>

/// Get/set max form width

/// </summary>

public static double MaxFormWidth { get { return maxFormWidth; } set { maxFormWidth = value; } }

/// <summary>

/// Get the visibility of the no button

/// </summary>

public static Visibility ShowDetailsBtn { get { return showDetailsBtn; } set { showDetailsBtn = value; } }

/// <summary>

/// Get/set details text

/// </summary>

public static string DetailsText { get { return detailsText; } set { detailsText = value; } }

/// <summary>

/// Get/set the visibility of the checkbox

/// </summary>

public static Visibility ShowCheckBox { get { return showCheckBox; } set { showCheckBox = value; } }

/// <summary>

/// Get/set the checkbox data object

/// </summary>

public static MsgBoxExCheckBoxData CheckBoxData { get { return checkBoxData; } set { checkBoxData = value; } }

/// <summary>

/// Get/set the font family

/// </summary>

public static System.Windows.Media.FontFamily MsgFontFamily { get { return msgFontFamily; } set { msgFontFamily = value; } }

/// <summary>

/// Get/set the font size

/// </summary>

public static double MsgFontSize { get { return msgFontSize; } set { msgFontSize = value; } }

/// <summary>

/// Get/set the uri object that represents the desired URL

/// </summary>

public static Uri Url { get { return url; } set { url = value; } }

/// <summary>

/// Get/set the visibility of the checkbox

/// </summary>

public static Visibility ShowUrl { get { return showUrl; } set { showUrl = value; } }

/// <summary>

/// Get/set the optional url display name

/// </summary>

public static string UrlDisplayName { get { return urlDisplayName; } set { urlDisplayName = value; } }

/// <summary>

/// Get/set the brush for the message text background

/// </summary>

public static SolidColorBrush UrlForeground { get { return urlForeground; } set { urlForeground = value; } }

/// <summary>

/// Get/set the delegate tooltip text

/// </summary>

public static string DelegateToolTip { get { return delegateToolTip; } set { delegateToolTip = value; } }

#endregion static properties

#region Show and ShowEx

/// <summary>

/// Does the work of actually opening the messagebox

/// </summary>

/// <param name="msg"></param>

/// <param name="title"></param>

/// <param name="buttons"></param>

/// <param name="image"></param>

/// <returns></returns>

private static MessageBoxResult OpenMessageBox(Window owner, string msg, string title, MessageBoxButton buttons, MessageBoxImage image)

{

if (owner == null)

{

owner = (Application.Current.MainWindow.Visibility == Visibility.Visible) ? Application.Current.MainWindow : null;

}

MessageBoxEx form = new MessageBoxEx(msg, title, buttons, image) { Owner = owner };

form.ShowDialog();

return form.MessageResult;

}

///<summary>

///Does the work of actually opening the messagebox with extended functionality

///</summary>

///<param name="msg"></param>

///<param name="title"></param>

///<param name="buttons"></param>

///<param name="image"></param>

///<returns></returns>

private static MessageBoxResultEx OpenMessageBox(Window owner, string msg, string title, MessageBoxButtonEx buttons, MessageBoxImage image)

{

if (owner == null)

{

owner = (Application.Current.MainWindow.Visibility == Visibility.Visible) ? Application.Current.MainWindow : null;

}

MessageBoxEx form = new MessageBoxEx(msg, title, buttons, image) { Owner = owner };

form.ShowDialog();

return form.MessageResultEx;

}

/// <summary>

/// Show the message box with the same characteristics as the standard WPF message box. The args <br/>

/// parameter can accept one or more of the following parameters. Sequence is not a factor, and <br/>

/// once a value that is NOT the default is encountered, evaluation stops for that object type. <br /><br/>

/// -- expected args: <see cref="string"/> (title), <see cref="Window"/> (owner), <see cref="MessageBoxButton"/>, <see cref="MessageBoxImage"/>, <see cref="MessageBoxButtonDefault"/><br /><br />

/// --------

/// </summary>

/// <param name="msg">The message to display</param>

/// <param name="args">An object array that contains 1 or more objects (see summary).</param>

/// <returns>Button that was clicked to dismiss the form.</returns>

public static MessageBoxResult Show(string msg, params object[] args)

{

string title = null;

Window owner = null;

MessageBoxButton buttons = MessageBoxButton.OK;

MessageBoxImage image = MessageBoxImage.None;

MessageBoxButtonDefault buttonDefault = MessageBoxButtonDefault.Forms;

foreach (object item in args)

{

if (item is string ttl && !string.IsNullOrEmpty(title)) { title = ttl; }

if (item is MessageBoxButton btn && buttons == MessageBoxButton.OK) { buttons = btn; }

if (item is MessageBoxImage img && image == MessageBoxImage.None) { image = img; }

if (item is MessageBoxButtonDefault def && buttonDefault == MessageBoxButtonDefault.Forms) { buttonDefault = def; }

if (item is Window wnd && owner == null) { owner = wnd; }

}

staticButtonDefault = buttonDefault;

title = (string.IsNullOrEmpty(title)) ? string.Empty : title.Trim();

if (string.IsNullOrEmpty(title))

{

if (image != MessageBoxImage.None)

{

title = image.ToString();

}

else

{

title = \_DEFAULT\_CAPTION;

}

}

return OpenMessageBox(owner, msg, title, buttons, image);

}

/// <summary>

/// Show the message box with extended message box functionality. The args parameter can accept <br />

/// one or more of the following parameters. Sequence is not a factor, and once a value that is NOT <br />

/// the default is encountered, evaluation stops for that object type. <br /><br/>

/// -- expected args: <see cref="string"/> (title), <see cref="Window"/> (owner), <see cref="MessageBoxButtonEx"/>, <see cref="MessageBoxImage"/>, <see cref="MessageBoxButtonDefault"/>, <see cref="MsgBoxExtendedFunctionality"/><br /><br />

/// --------

/// </summary>

/// <param name="msg">The message to display</param>

/// <param name="args">An object array that contains 1 or more objects (see summary).</param>

/// <returns>Button that was clicked to dismiss the form.</returns>

public static MessageBoxResultEx ShowEx(string msg, params object[] args)

{

string title = null;

Window owner = null;

MessageBoxImage image = MessageBoxImage.None;

MessageBoxButtonEx buttons = MessageBoxButtonEx.OK;

MessageBoxButtonDefault buttonDefault = MessageBoxButtonDefault.Forms;

MsgBoxExtendedFunctionality data = null;

foreach (object item in args)

{

if (item is string ttl && !string.IsNullOrEmpty(title)) { title = ttl; }

if (item is MessageBoxButtonEx btn && buttons == MessageBoxButtonEx.OK) { buttons = btn; }

if (item is MessageBoxImage img && image == MessageBoxImage.None) { image = img; }

if (item is MessageBoxButtonDefault def && buttonDefault == MessageBoxButtonDefault.Forms) { buttonDefault = def; }

if (item is Window wnd && owner == null) { owner = wnd; }

if (item is MsgBoxExtendedFunctionality mef && data == null) { data = mef; }

}

staticButtonDefault = buttonDefault;

if (data != null)

{

// details text ===================

DetailsText = data.DetailsText;

// checkbox =======================

ShowCheckBox = Visibility.Collapsed;

CheckBoxData = data.CheckBoxData;

// clickable icon =================

DelegateObj = data.MessageDelegate;

ExitAfterErrorAction = data.ExitAfterAction;

DelegateToolTip = data.DelegateToolTip;

// url ============================

// assume we're not showing a url

ShowUrl = Visibility.Collapsed;

Url = null;

UrlDisplayName = null;

// now, see if we want to

if (data.URL != null)

{

// if the url is ultimately null, no url will be displayed, and none of the following

// settings really mean anything

Url = data.URL.URL;

ShowUrl = (Url == null) ? Visibility.Collapsed : Visibility.Visible;

UrlDisplayName = (Url == null) ? null : data.URL.DisplayName;

// make sure we actually set a color. If the color was not included, use the message text color

UrlForeground = (data.URL.Foreground != null) ? new SolidColorBrush(data.URL.Foreground) : new SolidColorBrush(DefaultUrlForegroundColor);

}

}

title = (string.IsNullOrEmpty(title)) ? string.Empty : title.Trim();

if (string.IsNullOrEmpty(title.Trim()))

{

if (image != MessageBoxImage.None)

{

title = image.ToString();

}

else

{

title = \_DEFAULT\_CAPTION;

}

}

return OpenMessageBox(owner, msg, title, buttons, image);

}

#endregion Show and ShowEx

#if \_\_OVERLOADED\_SHOW\_\_

// I essentially abandoned this code because the overloads were becoming burdonsome

// and tedious If you reenable it, you are on your own.

#region static Show() methods

/////////////////////////////////////// without icons

/// <summary>

/// Show a messagebox, using default caption and just the OK button

/// </summary>

/// <param name="msg">The message to display</param>

/// <returns>The button that was clicked to dismiss the messagebox</returns>

public static MessageBoxResult Show(string msg)

{

return OpenMessageBox(null, msg, \_DEFAULT\_CAPTION, MessageBoxButton.OK, MessageBoxImage.None);

}

public static MessageBoxResult Show(Window owner, string msg)

{

return OpenMessageBox(owner, msg, \_DEFAULT\_CAPTION, MessageBoxButton.OK, MessageBoxImage.None);

}

/// <summary>

/// Show a messagebox with the specified caption, and just the OK button

/// </summary>

/// <param name="msg">The message to display</param>

/// <param name="title">The messagebox caption</param>

/// <returns>The button that was clicked to dismiss the messagebox</returns>

public static MessageBoxResult Show(string msg, string title)

{

title = (string.IsNullOrEmpty(title)) ? \_DEFAULT\_CAPTION : title;

return OpenMessageBox(null, msg, title, MessageBoxButton.OK, MessageBoxImage.None);

}

public static MessageBoxResult Show(Window owner, string msg, string title)

{

return OpenMessageBox(owner, msg, (string.IsNullOrEmpty(title)) ? \_DEFAULT\_CAPTION : title, MessageBoxButton.OK, MessageBoxImage.None);

}

/// <summary>

/// Show a messagebox with the default caption and the specified buttons

/// </summary>

/// <param name="msg">The message to display</param>

/// <param name="buttons">The buttons to display</param>

/// <returns>The button that was clicked to dismiss the messagebox</returns>

public static MessageBoxResult Show(string msg, MessageBoxButton buttons)

{

return OpenMessageBox(null, msg, \_DEFAULT\_CAPTION, buttons, MessageBoxImage.None);

}

public static MessageBoxResult Show(Window owner, string msg, MessageBoxButton buttons)

{

return OpenMessageBox(owner, msg, \_DEFAULT\_CAPTION, buttons, MessageBoxImage.None);

}

/// <summary>

/// Show a mesagebox with the specified caption and button(s)

/// </summary>

/// <param name="msg">The message to display</param>

/// <param name="title">The title for the message box</param>

/// <param name="parentWindow">The parent window that supplies the font family/size</param>

/// <param name="buttons">The buttons to display</param>

/// <returns>The button that was clicked to dismiss the messagebox</returns>

public static MessageBoxResult Show(string msg, string title, MessageBoxButton buttons)

{

return OpenMessageBox(null, msg, (string.IsNullOrEmpty(title)) ? \_DEFAULT\_CAPTION : title, buttons, MessageBoxImage.None);

}

public static MessageBoxResult Show(Window owner, string msg, string title, MessageBoxButton buttons)

{

return OpenMessageBox(null, msg, (string.IsNullOrEmpty(title)) ? \_DEFAULT\_CAPTION : title, buttons, MessageBoxImage.None);

}

/////////////////////////////////////// with icons

/// <summary>

/// Show a messagebox, using default caption, the OK button, and the specified icon

/// </summary>

/// <param name="msg">The message to display</param>

/// <param name="image">The message box icon to diplay</param>

/// <returns>The button that was clicked to dismiss the messagebox</returns>

public static MessageBoxResult Show(string msg, MessageBoxImage image)

{

return OpenMessageBox(null, msg, \_DEFAULT\_CAPTION, MessageBoxButton.OK, image);

}

/// <summary>

/// Show a messagebox with the specified caption, the OK button, and the specified icon

/// </summary>

/// <param name="msg">The message to display</param>

/// <param name="title">The messagebox caption</param>

/// <param name="image">The message box icon to diplay</param>

/// <returns>The button that was clicked to dismiss the messagebox</returns>

public static MessageBoxResult Show(string msg, string title, MessageBoxImage image)

{

title = (string.IsNullOrEmpty(title)) ? \_DEFAULT\_CAPTION : title;

return OpenMessageBox(null, msg, title, MessageBoxButton.OK, image);

}

/// <summary>

/// Show a messagebox with the default caption, the specified button(s), and the specified icon

/// </summary>

/// <param name="msg">The message to display</param>

/// <param name="buttons">The buttons to display</param>

/// <param name="image">The message box icon to diplay</param>

/// <returns>The button that was clicked to dismiss the messagebox</returns>

public static MessageBoxResult Show(string msg, MessageBoxButton buttons, MessageBoxImage image)

{

return OpenMessageBox(null, msg, \_DEFAULT\_CAPTION, buttons, image);

}

/// <summary>

/// Show a mesagebox with the specified caption, the specified button(s), and the specified icon

/// </summary>

/// <param name="msg">The message to display</param>

/// <param name="title">The title for the message box</param>

/// <param name="parentWindow">The parent window that supplies the font family/size</param>

/// <param name="buttons">The buttons to display</param>

/// <param name="image">The message box icon to diplay</param>

/// <returns>The button that was clicked to dismiss the messagebox</returns>

public static MessageBoxResult Show(string msg, string title, MessageBoxButton buttons, MessageBoxImage image)

{

return OpenMessageBox(null, msg, title, buttons, image);

}

#endregion static Show() methods

#region static ShowEx() methods with MessageBoxButtonsEx and returning MessageBoxResultEx

/// <summary>

/// Show a messagebox, using default caption and just the OK button

/// </summary>

/// <param name="msg">The message to display</param>

/// <returns>The button that was clicked to dismiss the messagebox</returns>

public static MessageBoxResultEx ShowEx(string msg)

{

return OpenMessageBox(msg, \_DEFAULT\_CAPTION, MessageBoxButtonEx.OK, MessageBoxImage.None);

}

/// <summary>

/// Show a messagebox with the specified caption, and just the OK button

/// </summary>

/// <param name="msg">The message to display</param>

/// <param name="title">The messagebox caption</param>

/// <returns>The button that was clicked to dismiss the messagebox</returns>

public static MessageBoxResultEx ShowEx(string msg, string title)

{

title = (string.IsNullOrEmpty(title)) ? \_DEFAULT\_CAPTION : title;

return OpenMessageBox(msg, title, MessageBoxButtonEx.OK, MessageBoxImage.None);

}

/// <summary>

/// Show a messagebox with the default caption and the specified buttons

/// </summary>

/// <param name="msg">The message to display</param>

/// <param name="buttons">The buttons to display</param>

/// <returns>The button that was clicked to dismiss the messagebox</returns>

public static MessageBoxResultEx ShowEx(string msg, MessageBoxButtonEx buttons)

{

return OpenMessageBox(msg, \_DEFAULT\_CAPTION, buttons, MessageBoxImage.None);

}

/// <summary>

/// Show a mesagebox with the specified caption and button(s)

/// </summary>

/// <param name="msg">The message to display</param>

/// <param name="title">The title for the message box</param>

/// <param name="parentWindow">The parent window that supplies the font family/size</param>

/// <param name="buttons">The buttons to display</param>

/// <returns>The button that was clicked to dismiss the messagebox</returns>

public static MessageBoxResultEx ShowEx(string msg, string title, MessageBoxButtonEx buttons)

{

title = (string.IsNullOrEmpty(title)) ? \_DEFAULT\_CAPTION : title;

return OpenMessageBox(msg, title, buttons, MessageBoxImage.None);

}

/////////////////////////////////////// with icons

/// <summary>

/// Show a messagebox, using default caption, the OK button, and the specified icon

/// </summary>

/// <param name="msg">The message to display</param>

/// <param name="image">The message box icon to diplay</param>

/// <returns>The button that was clicked to dismiss the messagebox</returns>

public static MessageBoxResultEx ShowEx(string msg, MessageBoxImage image)

{

return OpenMessageBox(msg, \_DEFAULT\_CAPTION, MessageBoxButtonEx.OK, image);

}

/// <summary>

/// Show a messagebox with the specified caption, the OK button, and the specified icon

/// </summary>

/// <param name="msg">The message to display</param>

/// <param name="title">The messagebox caption</param>

/// <param name="image">The message box icon to diplay</param>

/// <returns>The button that was clicked to dismiss the messagebox</returns>

public static MessageBoxResultEx ShowEx(string msg, string title, MessageBoxImage image)

{

title = (string.IsNullOrEmpty(title)) ? \_DEFAULT\_CAPTION : title;

return OpenMessageBox(msg, title, MessageBoxButtonEx.OK, image);

}

/// <summary>

/// Show a messagebox with the default caption, the specified button(s), and the specified icon

/// </summary>

/// <param name="msg">The message to display</param>

/// <param name="buttons">The buttons to display</param>

/// <param name="image">The message box icon to diplay</param>

/// <returns>The button that was clicked to dismiss the messagebox</returns>

public static MessageBoxResultEx ShowEx(string msg, MessageBoxButtonEx buttons, MessageBoxImage image)

{

return OpenMessageBox(msg, \_DEFAULT\_CAPTION, buttons, image);

}

/// <summary>

/// Show a mesagebox with the specified caption, the specified button(s), and the specified icon

/// </summary>

/// <param name="msg">The message to display</param>

/// <param name="title">The title for the message box</param>

/// <param name="parentWindow">The parent window that supplies the font family/size</param>

/// <param name="buttons">The buttons to display</param>

/// <param name="image">The message box icon to diplay</param>

/// <returns>The button that was clicked to dismiss the messagebox</returns>

public static MessageBoxResultEx ShowEx(string msg, string title, MessageBoxButtonEx buttons, MessageBoxImage image)

{

return OpenMessageBox(msg, title, buttons, image);

}

#endregion static Show() methods with MessageBoxButtonsEx and returning MessageBoxResultEx

#region show with extended functionality

/// <summary>

/// Show messagebox with one or more extended functionality items enabled

/// </summary>

/// <param name="ext">The extended functionality object</param>

/// <param name="message">The message to present</param>

/// <param name="title">The window caption</param>

/// <param name="buttons">The buttons to present</param>

/// <param name="image">The icon to present</param>

/// <returns>If the ext param is null, a normal messageboxex will be presented.</returns>

public static MessageBoxResult ShowExtBase(MsgBoxExtendedFunctionality ext, string message, string title, MessageBoxButton buttons, MessageBoxImage image)

{

// we ultimately want to always display a message box, so we try to gracefully degrade

// all of the "super" functionality.

if (ext != null)

{

// details text ===================

DetailsText = ext.DetailsText;

// checkbox =======================

ShowCheckBox = Visibility.Collapsed;

CheckBoxData = ext.CheckBoxData;

// clickable icon =================

DelegateObj = ext.MessageDelegate;

ExitAfterErrorAction = ext.ExitAfterAction;

DelegateToolTip = ext.DelegateToolTip;

// url ============================

// assume we're not showing a url

ShowUrl = Visibility.Collapsed;

Url = null;

UrlDisplayName = null;

// now, see if we want to

if (ext.URL != null)

{

// if the url is ultimately null, no url will be displayed, and none of the following

// settings really mean anything

Url = ext.URL.URL;

ShowUrl = (Url == null) ? Visibility.Collapsed : Visibility.Visible;

UrlDisplayName = (Url == null) ? null : ext.URL.DisplayName;

// make sure we actually set a color. If the color was not included, use the message text color

UrlForeground = (ext.URL.Foreground != null) ? new SolidColorBrush(ext.URL.Foreground) : new SolidColorBrush(DefaultUrlForegroundColor);

}

}

return OpenMessageBox(message, title, buttons, image);

}

/// <summary>

/// Show the MessageBoxEx with the specified extended functions

/// </summary>

/// <param name="ext">The extended functionality object</param>

/// <param name="message">The message text</param>

/// <returns>MessageBoxResult indicating the button used to disnmiss the window</returns>

public static MessageBoxResult ShowExt(MsgBoxExtendedFunctionality ext, string message)

{

return ShowExtBase(ext, message, \_DEFAULT\_CAPTION, MessageBoxButton.OK, MessageBoxImage.None);

}

/// <summary>

/// Show the MessageBoxEx with the specified extended functions

/// </summary>

/// <param name="ext">The extended functionality object</param>

/// <param name="message">The message text</param>

/// <param name="title">The window caption</param>

/// <returns>MessageBoxResult indicating the button used to disnmiss the window</returns>

public static MessageBoxResult ShowExt(MsgBoxExtendedFunctionality ext, string message, string title)

{

return ShowExtBase(ext, message, (string.IsNullOrEmpty(title))?\_DEFAULT\_CAPTION:title, MessageBoxButton.OK, MessageBoxImage.None);

}

/// <summary>

/// Show the MessageBoxEx with the specified extended functions

/// </summary>

/// <param name="ext">The extended functionality object</param>

/// <param name="message">The message text</param>

/// <param name="buttons">The message button(s)</param>

/// <returns>MessageBoxResult indicating the button used to disnmiss the window</returns>

public static MessageBoxResult ShowExt(MsgBoxExtendedFunctionality ext, string message, MessageBoxButton buttons)

{

return ShowExtBase(ext, message, \_DEFAULT\_CAPTION, buttons, MessageBoxImage.None);

}

/// <summary>

/// Show the MessageBoxEx with the specified extended functions

/// </summary>

/// <param name="ext">The extended functionality object</param>

/// <param name="message">The message text</param>

/// <param name="title">The window caption</param>

/// <param name="buttons">The message button(s)</param>

/// <returns>MessageBoxResult indicating the button used to disnmiss the window</returns>

public static MessageBoxResult ShowExt(MsgBoxExtendedFunctionality ext, string message, string title, MessageBoxButton buttons)

{

return ShowExtBase(ext, message, (string.IsNullOrEmpty(title))?\_DEFAULT\_CAPTION:title, buttons, MessageBoxImage.None);

}

/// <summary>

/// Show the MessageBoxEx with the specified extended functions

/// </summary>

/// <param name="ext">The extended functionality object</param>

/// <param name="message">The message text</param>

/// <param name="image">The message icon</param>

/// <returns>MessageBoxResult indicating the button used to disnmiss the window</returns>

public static MessageBoxResult ShowExt(MsgBoxExtendedFunctionality ext, string message, MessageBoxImage image)

{

return ShowExtBase(ext, message, \_DEFAULT\_CAPTION, MessageBoxButton.OK, image);

}

/// <summary>

/// Show the MessageBoxEx with the specified extended functions

/// </summary>

/// <param name="ext">The extended functionality object</param>

/// <param name="message">The message text</param>

/// <param name="title">The window caption</param>

/// <param name="image">The message icon</param>

/// <returns>MessageBoxResult indicating the button used to disnmiss the window</returns>

public static MessageBoxResult ShowExt(MsgBoxExtendedFunctionality ext, string message, string title, MessageBoxImage image)

{

return ShowExtBase(ext, message, (string.IsNullOrEmpty(title))?\_DEFAULT\_CAPTION:title, MessageBoxButton.OK, image);

}

/// <summary>

/// Show the MessageBoxEx with the specified extended functions

/// </summary>

/// <param name="ext">The extended functionality object</param>

/// <param name="message">The message text</param>

/// <param name="buttons">The message button(s)</param>

/// <param name="image">The message icon</param>

/// <returns>MessageBoxResult indicating the button used to disnmiss the window</returns>

public static MessageBoxResult ShowExt(MsgBoxExtendedFunctionality ext, string message, MessageBoxButton buttons, MessageBoxImage image)

{

return ShowExtBase(ext, message, \_DEFAULT\_CAPTION, buttons, image);

}

/// <summary>

/// Show the MessageBoxEx with the specified extended functions

/// </summary>

/// <param name="ext">The extended functionality object</param>

/// <param name="message">The message text</param>

/// <param name="title">The window caption</param>

/// <param name="buttons">The message button(s)</param>

/// <param name="image">The message icon</param>

/// <returns>MessageBoxResult indicating the button used to disnmiss the window</returns>

public static MessageBoxResult ShowExt(MsgBoxExtendedFunctionality ext, string message, string title, MessageBoxButton buttons, MessageBoxImage image)

{

return ShowExtBase(ext, message, (string.IsNullOrEmpty(title))?\_DEFAULT\_CAPTION:title, buttons, image);

}

#endregion show with extended functionality

/\*

#region ShowEx with extended functionality using MessageBoxButtonsEx and returning MessageBoxResultEx

/// <summary>

/// Show messagebox with one or more extended functionality items enabled

/// </summary>

/// <param name="ext">The extended functionality object</param>

/// <param name="message">The message to present</param>

/// <param name="title">The window caption</param>

/// <param name="buttons">The buttons to present</param>

/// <param name="image">The icon to present</param>

/// <returns>If the ext param is null, a normal messageboxex will be presented.</returns>

public static MessageBoxResultEx ShowExtBase(MsgBoxExtendedFunctionality ext, string message, string title, MessageBoxButtonEx buttons, MessageBoxImage image)

{

// we ultimately want to always display a message box, so we try to gracefully degrade

// all of the "super" functionality.

if (ext != null)

{

// details text ===================

DetailsText = ext.DetailsText;

// checkbox =======================

ShowCheckBox = Visibility.Collapsed;

CheckBoxData = ext.CheckBoxData;

// clickable icon =================

DelegateObj = ext.MessageDelegate;

ExitAfterErrorAction = ext.ExitAfterAction;

DelegateToolTip = ext.DelegateToolTip;

// url ============================

// assume we're not showing a url

ShowUrl = Visibility.Collapsed;

Url = null;

UrlDisplayName = null;

// now, see if we want to

if (ext.URL != null)

{

// if the url is ultimately null, no url will be displayed, and none of the following

// settings really mean anything

Url = ext.URL.URL;

ShowUrl = (Url == null) ? Visibility.Collapsed : Visibility.Visible;

UrlDisplayName = (Url == null) ? null : ext.URL.DisplayName;

// make sure we actually set a color. If the color was not included, use the message text color

UrlForeground = (ext.URL.Foreground != null) ? new SolidColorBrush(ext.URL.Foreground) : new SolidColorBrush(DefaultUrlForegroundColor);

}

staticButtonDefault = ext.ButtonDefault;

}

return OpenMessageBox(message, title, buttons, image);

}

/// <summary>

/// Show the MessageBoxEx with the specified extended functions

/// </summary>

/// <param name="ext">The extended functionality object</param>

/// <param name="message">The message text</param>

/// <returns>MessageBoxResult indicating the button used to disnmiss the window</returns>

public static MessageBoxResultEx ShowExtEx(MsgBoxExtendedFunctionality ext, string message)

{

return ShowExtBase(ext, message, \_DEFAULT\_CAPTION, MessageBoxButtonEx.OK, MessageBoxImage.None);

}

/// <summary>

/// Show the MessageBoxEx with the specified extended functions

/// </summary>

/// <param name="ext">The extended functionality object</param>

/// <param name="message">The message text</param>

/// <param name="title">The window caption</param>

/// <returns>MessageBoxResult indicating the button used to disnmiss the window</returns>

public static MessageBoxResultEx ShowExtEx(MsgBoxExtendedFunctionality ext, string message, string title)

{

return ShowExtBase(ext, message, (string.IsNullOrEmpty(title))?\_DEFAULT\_CAPTION:title, MessageBoxButtonEx.OK, MessageBoxImage.None);

}

/// <summary>

/// Show the MessageBoxEx with the specified extended functions

/// </summary>

/// <param name="ext">The extended functionality object</param>

/// <param name="message">The message text</param>

/// <param name="buttons">The message button(s)</param>

/// <returns>MessageBoxResult indicating the button used to disnmiss the window</returns>

public static MessageBoxResultEx ShowExtEx(MsgBoxExtendedFunctionality ext, string message, MessageBoxButtonEx buttons)

{

return ShowExtBase(ext, message, \_DEFAULT\_CAPTION, buttons, MessageBoxImage.None);

}

/// <summary>

/// Show the MessageBoxEx with the specified extended functions

/// </summary>

/// <param name="ext">The extended functionality object</param>

/// <param name="message">The message text</param>

/// <param name="title">The window caption</param>

/// <param name="buttons">The message button(s)</param>

/// <returns>MessageBoxResult indicating the button used to disnmiss the window</returns>

public static MessageBoxResultEx ShowExtEx(MsgBoxExtendedFunctionality ext, string message, string title, MessageBoxButtonEx buttons)

{

return ShowExtBase(ext, message, (string.IsNullOrEmpty(title))?\_DEFAULT\_CAPTION:title, buttons, MessageBoxImage.None);

}

/// <summary>

/// Show the MessageBoxEx with the specified extended functions

/// </summary>

/// <param name="ext">The extended functionality object</param>

/// <param name="message">The message text</param>

/// <param name="image">The message icon</param>

/// <returns>MessageBoxResult indicating the button used to disnmiss the window</returns>

public static MessageBoxResultEx ShowExtEx(MsgBoxExtendedFunctionality ext, string message, MessageBoxImage image)

{

return ShowExtBase(ext, message, \_DEFAULT\_CAPTION, MessageBoxButtonEx.OK, image);

}

/// <summary>

/// Show the MessageBoxEx with the specified extended functions

/// </summary>

/// <param name="ext">The extended functionality object</param>

/// <param name="message">The message text</param>

/// <param name="title">The window caption</param>

/// <param name="image">The message icon</param>

/// <returns>MessageBoxResult indicating the button used to disnmiss the window</returns>

public static MessageBoxResultEx ShowExtEx(MsgBoxExtendedFunctionality ext, string message, string title, MessageBoxImage image)

{

return ShowExtBase(ext, message, (string.IsNullOrEmpty(title))?\_DEFAULT\_CAPTION:title, MessageBoxButtonEx.OK, image);

}

/// <summary>

/// Show the MessageBoxEx with the specified extended functions

/// </summary>

/// <param name="ext">The extended functionality object</param>

/// <param name="message">The message text</param>

/// <param name="buttons">The message button(s)</param>

/// <param name="image">The message icon</param>

/// <returns>MessageBoxResult indicating the button used to disnmiss the window</returns>

public static MessageBoxResultEx ShowExtEx(MsgBoxExtendedFunctionality ext, string message, MessageBoxButtonEx buttons, MessageBoxImage image)

{

return ShowExtBase(ext, message, \_DEFAULT\_CAPTION, buttons, image);

}

/// <summary>

/// Show the MessageBoxEx with the specified extended functions

/// </summary>

/// <param name="ext">The extended functionality object</param>

/// <param name="message">The message text</param>

/// <param name="title">The window caption</param>

/// <param name="buttons">The message button(s)</param>

/// <param name="image">The message icon</param>

/// <returns>MessageBoxResult indicating the button used to disnmiss the window</returns>

public static MessageBoxResultEx ShowExtEx(MsgBoxExtendedFunctionality ext, string message, string title, MessageBoxButtonEx buttons, MessageBoxImage image)

{

return ShowExtBase(ext, message, (string.IsNullOrEmpty(title))?\_DEFAULT\_CAPTION:title, buttons, image);

}

#endregion show with extended functionality

\*/

#endif

#region static configuration methods

// colors

/// <summary>

/// Set the background color for the message area

/// </summary>

/// <param name="color">The color to set. Can be a name (White) or an octet string(#FFFFFF).</param>

public static void SetMessageBackground(System.Windows.Media.Color color)

{

try

{

MessageBackground = new SolidColorBrush(color);

}

catch (Exception ex)

{

MessageBoxEx.ShowEx(ex.Message, ex.ToString());

}

}

/// <summary>

/// Set the foreground color for the message area

/// </summary>

/// <param name="color">The color to set. Can be a name (White) or an octet string(#FFFFFF).</param>

public static void SetMessageForeground(System.Windows.Media.Color color)

{

try

{

MessageForeground = new SolidColorBrush(color);

}

catch (Exception ex)

{

MessageBoxEx.ShowEx(ex.Message, ex.ToString());

}

}

/// <summary>

/// Set the background color for the button panel area

/// </summary>

/// <param name="color">The color to set. Can be a name (White) or an octet string(#FFFFFF).</param>

public static void SetButtonBackground(System.Windows.Media.Color color)

{

try

{

ButtonBackground = new SolidColorBrush(color);

}

catch (Exception ex)

{

MessageBoxEx.ShowEx(ex.Message, ex.ToString());

}

}

/// <summary>

/// Create a WPF-compatible Color from an string (such as "White", "white", or "#FFFFFF").

/// </summary>

/// <param name="colorOctet"></param>

/// <returns>A Media.Color. If color is invalid, returns #000000.</returns>

public static System.Windows.Media.Color ColorFromString(string colorString)

{

System.Windows.Media.Color wpfColor = System.Windows.Media.Colors.Black;

try

{

wpfColor = (System.Windows.Media.Color)(System.Windows.Media.ColorConverter.ConvertFromString(colorString));

}

catch (Exception) { }

return wpfColor;

}

// font

/// <summary>

/// Set the font family and size from the application's main window

/// </summary>

public static void SetFont()

{

MsgFontFamily = Application.Current.MainWindow.FontFamily;

MsgFontSize = Application.Current.MainWindow.FontSize;

}

/// <summary>

/// Set the font family and size from the specified content control (usually the parent window)

/// </summary>

/// <param name="parent"></param>

public static void SetFont(ContentControl parent)

{

MsgFontFamily = parent.FontFamily;

MsgFontSize = parent.FontSize;

}

/// <summary>

/// Set the font family and size

/// </summary>

/// <param name="familyName">The name of the desired font family (will not be set if null/empty)</param>

/// <param name="size">Size of font (min size is 1)</param>

public static void SetFont(string familyName, double size)

{

if (!IsFontFamilyValid(familyName))

{

if (!string.IsNullOrEmpty(familyName))

{

MsgFontFamily = new System.Windows.Media.FontFamily(familyName);

}

}

MsgFontSize = Math.Max(1.0, size);

}

private static bool IsFontFamilyValid(string name)

{

if (installedFonts.Count == 0)

{

using (InstalledFontCollection fontsCollection = new InstalledFontCollection())

{

installedFonts = (from x in fontsCollection.Families select x.Name).ToList();

}

}

return installedFonts.Contains(name);

}

// mechanicals

/// <summary>

/// Set the custom button template \*NAME\*

/// </summary>

/// <param name="name"></param>

public static void SetButtonTemplateName(string name)

{

ButtonTemplateName = name;

}

/// <summary>

/// Sets the max form width to largest of 300 or the specified value

/// </summary>

/// <param name="value"></param>

public static void SetMaxFormWidth(double value)

{

MaxFormWidth = Math.Max(value, 300);

double minWidth = 300;

MaxFormWidth = Math.Max(minWidth, Math.Min(value, screenWidth));

}

/// <summary>

/// Resets the configuration items to default values

/// </summary>

public static void ResetToDefaults()

{

MsgFontSize = 12d;

MsgFontFamily = new System.Windows.Media.FontFamily("Segoe UI");

DelegateObj = null;

DetailsText = null;

MessageForeground = null;

MessageBackground = null;

ButtonBackground = null;

ParentWindow = null;

isSilent = false;

enableCloseButton = true;

ButtonTemplateName = null;

MsgBoxIconToolTip = null;

ShowCheckBox = Visibility.Collapsed;

CheckBoxData = null;

ExitAfterErrorAction = false;

MaxFormWidth = 800;

Url = null;

ShowUrl = Visibility.Collapsed;

UrlDisplayName = null;

UrlForeground = new SolidColorBrush(DefaultUrlForegroundColor);

staticButtonDefault = MessageBoxButtonDefault.Forms;

}

public static void EnableCloseButton(bool enable)

{

enableCloseButton = enable;

}

// message box icon

/// <summary>

/// Toggle system sounds associated with MessageBoxImage icons

/// </summary>

/// <param name="quiet"></param>

public static void SetAsSilent(bool quiet)

{

isSilent = quiet;

}

/// <summary>

/// Specify the button that will be displayed as the default button in the in the message

/// box. the message box will return to the default "Forms" setting when it is dismissed.

/// </summary>

/// <param name="buttonDefault"></param>

public static void SetDefaultButton(MessageBoxButtonDefault buttonDefault)

{

staticButtonDefault = buttonDefault;

}

#endregion static configuration methods

}

}

using System;

using System.ComponentModel;

using System.Runtime.CompilerServices;

namespace MsgBoxEx

{

// This class doesn't have to be inherted because its use is highly specific.

/// <summary>

/// Reresents the object that allows the checkbox state to be discoevered externally of the

/// messagebox.

/// </summary>

public class MsgBoxExCheckBoxData : INotifyPropertyChanged

{

#region INotifyPropertyChanged

private bool isModified = false;

public bool IsModified { get { return this.isModified; } set { if (value != this.isModified) { this.isModified = true; this.NotifyPropertyChanged(); } } }

/// <summary>

/// Occurs when a property value changes.

/// </summary>

public event PropertyChangedEventHandler PropertyChanged;

/// <summary>

/// Notifies that the property changed, and sets IsModified to true.

/// </summary>

/// <param name="propertyName">Name of the property.</param>

protected void NotifyPropertyChanged([CallerMemberName] String propertyName = "")

{

if (this.PropertyChanged != null)

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

if (propertyName != "IsModified")

{

this.IsModified = true;

}

}

}

#endregion INotifyPropertyChanged

private string checkBoxText;

private bool checkBoxIsChecked;

/// <summary>

/// Get/set the text content of the checkbox

/// </summary>

public string CheckBoxText { get { return this.checkBoxText; } set { if (value != this.checkBoxText ) { this.checkBoxText = value; this.NotifyPropertyChanged(); } } }

/// <summary>

/// Get/set the flag that indicates whether the checkbox is checked

/// </summary>

public bool CheckBoxIsChecked { get { return this.checkBoxIsChecked; } set { if (value != this.checkBoxIsChecked) { this.checkBoxIsChecked = value; this.NotifyPropertyChanged(); } } }

}

}

using System;

using System.Windows;

namespace MsgBoxEx

{

// This class MUST be inherited, and the PerformAction method MUST be overriden.

/// <summary>

/// Represents the object that allows a message box icon to execute code. This class must be

/// inherited.

/// </summary>

public abstract class MsgBoxExDelegate

{

/// <summary>

/// Get/set the message text from the calling message box

/// </summary>

public string Message { get; set; }

/// <summary>

/// Get/set the details text (if it was specified in the messagebox)

/// </summary>

public string Details { get; set; }

/// <summary>

/// Get/set the message datetime at which this object was created

/// </summary>

public DateTime MessageDate { get; set; }

/// <summary>

/// Performs the desired action, and returns the result. MUST BE OVERIDDEN IN INHERITING CLASS.

/// </summary>

/// <returns></returns>

public virtual MessageBoxResult PerformAction(string message, string details = null)

{

throw new NotImplementedException();

}

}

}

namespace MsgBoxEx

{

/// <summary>

/// Represents one of or more of the extended functionality items

/// </summary>

public class MsgBoxExtendedFunctionality

{

public MessageBoxButtonDefault ButtonDefault { get; set; }

/// <summary>

/// Get/set the details text to display

/// </summary>

public string DetailsText { get; set; }

/// <summary>

/// Get/set the checkbox data object

/// </summary>

public MsgBoxExCheckBoxData CheckBoxData { get; set; }

/// <summary>

/// Get/set the clickable icon delegate object

/// </summary>

public MsgBoxExDelegate MessageDelegate { get; set; }

/// <summary>

/// Get/set the flag indicating whether the messagebox is dismissed after the delegate

/// action has completed.

/// </summary>

public bool ExitAfterAction { get; set; }

public string DelegateToolTip { get; set; }

/// <summary>

/// Get/set the url

/// </summary>

public MsgBoxUrl URL { get; set; }

public MsgBoxExtendedFunctionality()

{

this.ButtonDefault = MessageBoxButtonDefault.Forms;

this.DetailsText = null;

this.CheckBoxData = null;

this.MessageDelegate = null;

this.URL = null;

this.DelegateToolTip = "Click this icon for additional info/actions.";

}

}

}

using System;

namespace MsgBoxEx

{

/// <summary>

/// Represents a class that simplifies internal moving around of URL-specific info

/// </summary>

public class MsgBoxUrl

{

/// <summary>

/// Get/set the web link. Any Uri type other than "http" is ignored. The URL is also used for the tooltip.

/// </summary>

public Uri URL { get; set; }

/// <summary>

/// Get/set the optional display name for the web link

/// </summary>

public string DisplayName { get; set; }

/// <summary>

/// Get/set the foreground color for the web link

/// </summary>

public System.Windows.Media.Color Foreground { get; set; }

public MsgBoxUrl()

{

// default color

this.Foreground = MessageBoxEx.DefaultUrlForegroundColor;

}

}

}

using System;

using System.Collections.Generic;

using System.IO;

using System.Net.Mime;

using Avalonia;

using Avalonia.Controls;

using Avalonia.Input;

using Avalonia.Interactivity;

using Avalonia.Markup.Xaml;

using FastReport;

using FastReport.Export.Image;

namespace WpfReporter

{

public class ReporterWindow : Window

{

public ReporterWindow()

{

InitializeComponent();

imHeight = img.Height;

imWidth = img.Width;

}

private Button OpenBtn;

private Image img;

private TextBox PageNumber;

private void InitializeComponent()

{

AvaloniaXamlLoader.Load(this);

OpenBtn = this.FindControl<Button>("Open");

OpenBtn.Click += OpenFile;

this.img = this.FindControl<Image>("img");

this.PageNumber = this.FindControl<TextBox>("PageNumber");

}

public List<Avalonia.Media.Imaging.Bitmap> pages = new List<Avalonia.Media.Imaging.Bitmap>();

public Report Report

{

get { return report; }

set

{

ex = new FastReport.Export.Image.ImageExport();

ex.HasMultipleFiles = true;

report = value;

SetContent(report);

SetImage();

}

}

public int CurrentPage

{

get { return currentPage; }

set

{

if (value >= 0 && value < pages.Count)

currentPage = value;

}

}

private void SetContent(Report report)

{

DeleteTempFiles();

ex.ImageFormat = ImageExportFormat.Png;

ex.Resolution = 96;

Random rnd = new Random();

ex.Export(report, Directory.GetCurrentDirectory() + "/test." + rnd.Next(100) + ".png");

foreach (string file in ex.GeneratedFiles)

{

//Avalonia.Media.Imaging.Bitmap image = new Avalonia.Media.Imaging.Bitmap();

//image.BeginInit();

//image.CacheOption = BitmapCacheOption.OnLoad;

//image.UriSource = new Uri(file);

//image.EndInit();

pages.Add(new Avalonia.Media.Imaging.Bitmap(file));

}

CurrentPage = 0;

}

public void DeleteTempFiles()

{

FileInfo[] path = new DirectoryInfo(Directory.GetCurrentDirectory()).GetFiles("\*test\*", SearchOption.AllDirectories);

pages.Clear();

foreach (FileInfo file in path)

{

File.Delete(file.FullName);

}

}

public void SetImage()

{

//if (CurrentPage >= 0 && CurrentPage < pages.Count())

//{

// im.Source = null;

img.Source = pages[CurrentPage];

//}

img.Height = imHeight;

img.Width = imWidth;

PageNumber.Text = (CurrentPage + 1).ToString();

}

public void FirstClick(object sender, Avalonia.Interactivity.RoutedEventArgs e)

{

CurrentPage = 0;

SetImage();

}

public void Prev\_Click(object sender, Avalonia.Interactivity.RoutedEventArgs e)

{

CurrentPage--;

SetImage();

}

public void Next\_Click(object sender, Avalonia.Interactivity.RoutedEventArgs e)

{

CurrentPage++;

SetImage();

}

public void Last\_Click(object sender, Avalonia.Interactivity.RoutedEventArgs e)

{

CurrentPage = pages.Count - 1;

SetImage();

}

public void PageNumber\_KeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter)

{

if (int.Parse(PageNumber.Text) > 0)

{

CurrentPage = int.Parse(PageNumber.Text) - 1;

SetImage();

}

}

}

private async void OpenFile(object sender, RoutedEventArgs e)

{

OpenFileDialog myDialog = new OpenFileDialog();

myDialog.Filters.Add(new FileDialogFilter() { Name = "Подготовленные отчёты(\*.FPX)", Extensions = new List<string> { "fpx" } });

myDialog.AllowMultiple = false;

var result = await myDialog.ShowAsync(this);

try

{

if (result.Length > 0 && result != null)

LoadReport(result[0]);

}

catch (Exception ex)

{

return;

}

}

void LoadReport(string report\_name)

{

Report rep = new Report();

rep.LoadPrepared(report\_name);

Report = rep;

}

public void Zoom\_in\_Click(object sender, RoutedEventArgs e)

{

if (ex != null)

{

ex.ImageFormat = ImageExportFormat.Png;

ex.Resolution += 25;

ex.PageNumbers = (CurrentPage + 1).ToString();

//ex.Zoom += 0.25f;

ex.Export(Report, Directory.GetCurrentDirectory() + "/testZoom.png");

//Report.Export(ex, stream);

if (CurrentPage >= 0 && CurrentPage < pages.Count)

img.Source = LoadImage(Directory.GetCurrentDirectory() + "/testZoom.png");

//new BitmapImage(new Uri(Directory.GetCurrentDirectory() + "/testZoom.png"));

PageNumber.Text = (CurrentPage + 1).ToString();

}

img.Width += 50;

img.Height += 50;

}

private static Avalonia.Media.Imaging.Bitmap LoadImage(string file)

{

return new Avalonia.Media.Imaging.Bitmap(file);

}

public void Zoom\_out\_Click(object sender, RoutedEventArgs e)

{

if (ex != null)

{

ex.ImageFormat = ImageExportFormat.Png;

ex.Resolution -= 25;

ex.PageNumbers = (CurrentPage + 1).ToString();

//ex.Zoom += 0.25f;

ex.Export(Report, Directory.GetCurrentDirectory() + "/testZoom.png");

//Report.Export(ex, stream);

if (CurrentPage >= 0 && CurrentPage < pages.Count)

img.Source = LoadImage(Directory.GetCurrentDirectory() + "/testZoom.png");

//new BitmapImage(new Uri(Directory.GetCurrentDirectory() + "/testZoom.png"));

PageNumber.Text = (CurrentPage + 1).ToString();

}

img.Width -= 50;

img.Height -= 50;

}

private Report report;

private FastReport.Export.Image.ImageExport ex;

private int currentPage = 0;

private double imHeight;

private double imWidth;

}

}

## Приложение Б

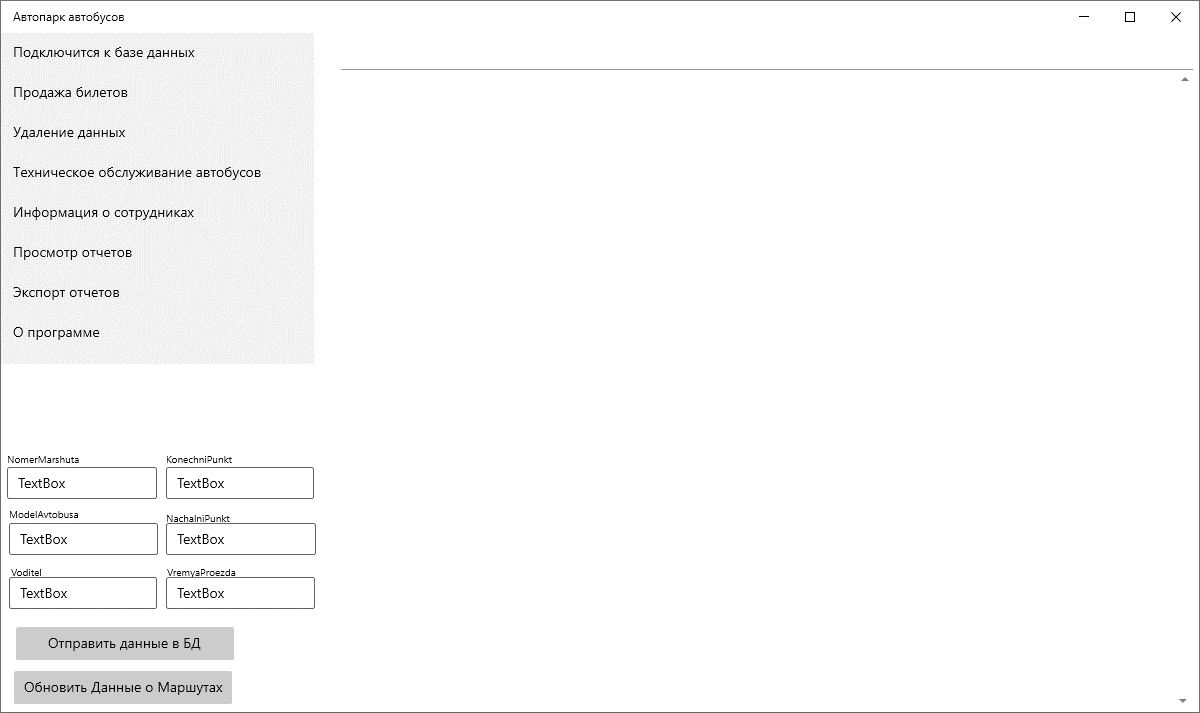


Рисунок 12. Форма «Информация о маршрутах»

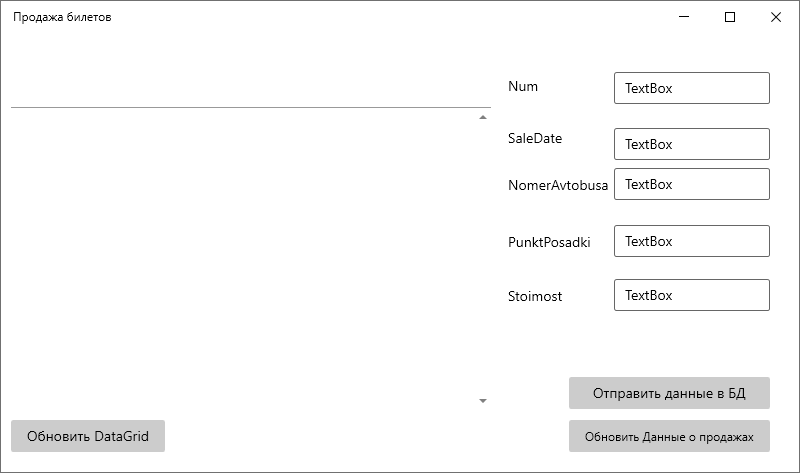


Рисунок 13. Форма «Продажа билетов».

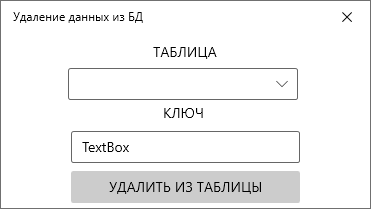


Рисунок 14. Форма «Удаление данных».

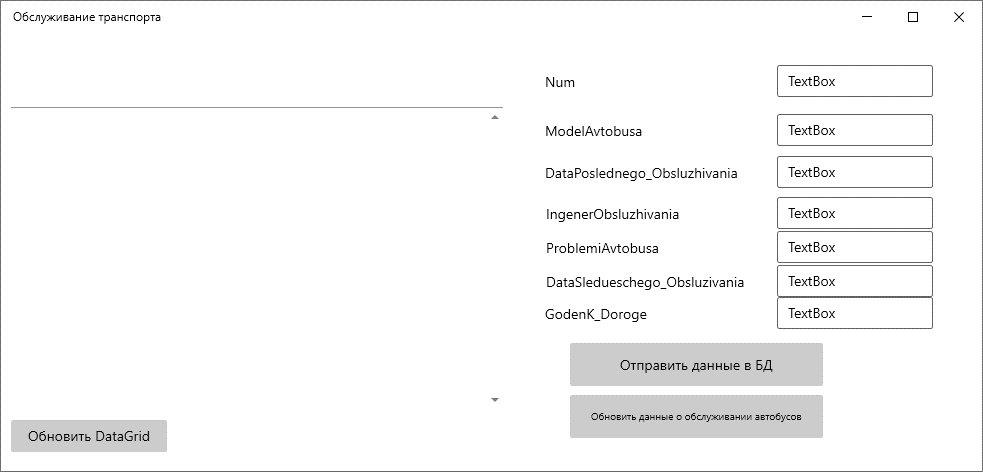


Рисунок 15. Форма «Техническое обслуживание автобусов».

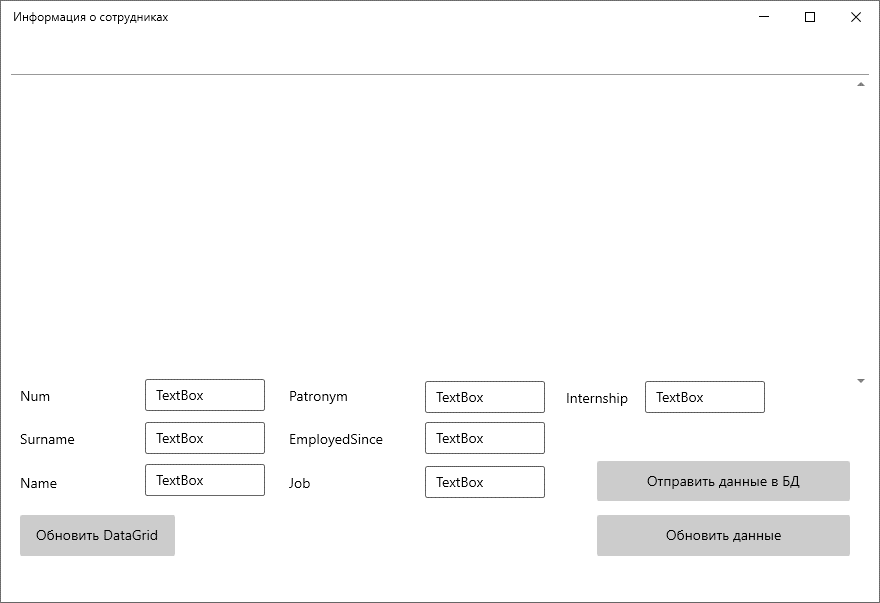


Рисунок 16. Форма «Информация о сотрудниках».



Рисунок 17. Форма «Просмотр отчетов».

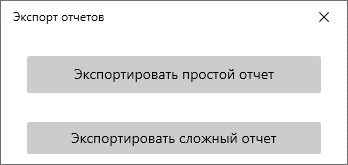


Рисунок 18. Форма «Экспорт отчетов».

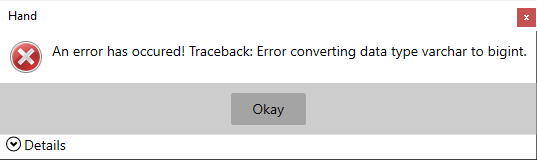


Рисунок 19 – Вид сообщения об ошибке

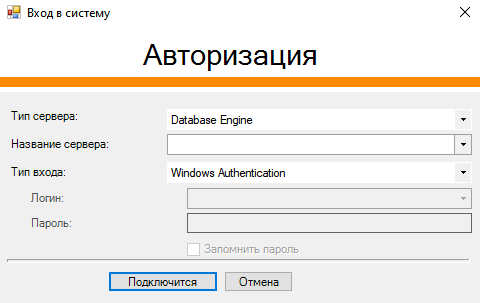


Рисунок 20 – Форма входа в систему

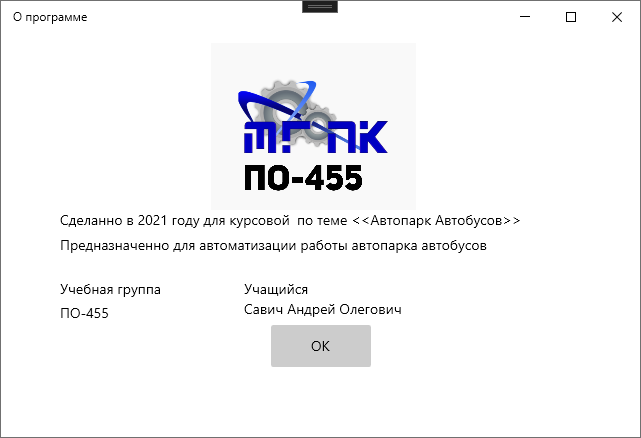


Рисунок 21 – Вид справки «О программе»

**ПРИЛОЖЕНИЕ В**

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

ГЛАВНОЕ УПРАВЛЕНИЕ ПО ОБРАЗОВАНИЮ

МОГИЛЕВСКОГО ОБЛАСТНОГО ИСПОЛНИТЕЛЬНОГО КОМИТЕТА

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«МОГИЛЕВСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»

# ОПИСАНИЕ ПРОГРАММЫ

2022

**Содержание**

1 Общие сведения 135

2 Функциональное назначение 135

3 Описание логической структуры 135

4 Используемые технические средства 140

5 Вызов и загрузка 141

**1 Общие сведения**

Наименование программы: РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ «АВТОБУСНЫЙ АВТОПАРК»

Автор программы:. Савич Андрей Олегович

Дата создания:Начало создания 13.12.2021 , Окончание работ 11.05.2022.

Версия 1.0.

Средой для функционирования программы является Microsoft Windows.

Программа написана на языке программирования С#.

**2 Функциональное назначение**

Разработка программы, позволит оперативно учитывать следующую информацию: маршруты автобусов; обеспечение учета информации о продажах билетов; состояние технического обслуживания автобусов; список сотрудников автопарка автобусов.

В программе также присутствует функционал создания и экспорта отчетов. В своей основе программа имеет базу данных Microsoft Sql Server.

Список функций:

1. Просмотр отчетов.
2. Экспорт простых и сложных отчетов.
3. Просмотр и занесение информации о продажах билетов.
4. Просмотр и занесение информации о техническом обслуживании автобусов.
5. Просмотр и занесение информации о сотрудниках автопарка автобусов.
6. Просмотр и занесение информации о маршрутах автобусов.

Значительным преимуществом автоматизации данных операций является скорость их выполнения, возможность в любой момент осуществить запрос к данным, находящимся в базе и получить по этому запросу исчерпывающую информацию. Программа позволяет отказаться от необходимости ведения бумажных архивов.

**3 Описание логической структуры**

Алгоритм программы приведен в таблице 1 ниже

Таблица 1. Описание процесса выполнения варианта использования.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Пользователь | Система | Экранная форма | Условие:  последующий шаг |
| 1. | Открывает программу | Программа загружает форму просмотра маршрутов | Информация о маршрутах. | Выбор одного из пунктов меню |
| 2. | Выбор пункта меню подключится к базе данных | Обновление DataGrid | Информация о маршрутах. | Выбор одного из пунктов меню или ввод информации в Textbox для добавления или обновления данных |
| 3. | Ввод информации и нажатие кнопки отправить данные в БД | Добавление введенной информации в таблицу Marshuti в базе данных после проверки на ошибки | Информация о маршрутах. | Выбор одного из пунктов меню |
| 4. | Ввод информации и нажатие кнопки обновить данные о маршрутах | Обновление данных в столбце с введенным ключом | Информация о маршрутах. | Выбор одного из пунктов меню |
| 5. | Выбор пункта меню продажа билетов | Открытие формы продажи билетов | Информация о маршрутах. | Выполнение ввода информации в форме продаже билетов и выбор одного из действий в форме или выбор одного из пунктов меню |
| 6. | Нажатие кнопки обновить DataGrid | Обновление DataGrid | Продажа билетов | Ввод информации в Textbox для добавления или обновления данных или выбор одного из пунктов меню в форме информации о маршрутах |
| 7. | Ввод информации и нажатие кнопки отправить данные в БД | Добавление введенной информации в таблицу Prodazhi в базе данных после проверки на ошибки | Продажа билетов | Выбор одного из пунктов меню в форме информации о маршрутах |
| 8. | Ввод информации и нажатие кнопки обновить данные о продажах | Обновление данных в столбце с введенным ключом в таблице Prodazhi в базе данных | Продажа билетов | Выбор одного из пунктов меню в форме информации о маршрутах |
| 9. | Выбор пункта меню удаление данных | Открытие формы удаления данных | Информация о маршрутах. | Выполнение ввода информации в форме удаления данных или выбор одного из пунктов меню |
| 10. | Ввод наименования таблицы и ключа столбца и нажатие кнопки удалить из таблицы | Удаление данных из введенной таблицы | Удаление данных | Выбор одного из пунктов меню в форме информации о маршрутах |
| 11. | Выбор пункта меню техническое обслуживание автобусов | Открытие формы технического обслуживания автобусов | Информация о маршрутах. | Выполнение ввода информации в форме технического обслуживания автобусов и выбор одного из действий в форме или выбор одного из пунктов меню |
| 12. | Нажатие кнопки обновить DataGrid | Обновление DataGrid | Техническое обслуживание автобусов | Ввод информации в Textbox для добавления или обновления данных или выбор одного из пунктов меню в форме информации о маршрутах |
| 13. | Ввод информации и нажатие кнопки отправить данные в БД | Добавление введенной информации в таблицу Maintenance в базе данных после проверки на ошибки | Техническое обслуживание автобусов | Выбор одного из пунктов меню в форме информации о маршрутах |
| 14. | Ввод информации и нажатие кнопки обновить данные о обслуживании автобусов | Обновление данных в столбце с введенным ключом в таблице Maintenance в базе данных | Техническое обслуживание автобусов | Выбор одного из пунктов меню в форме информации о маршрутах |
| 15. | Выбор пункта меню информация о сотрудниках | Открытие формы информации о сотрудниках | Информация о маршрутах. | Выполнение ввода информации в форме информации о сотрудниках и выбор одного из действий в форме или выбор одного из пунктов меню |
| 16. | Нажатие кнопки обновить DataGrid | Обновление DataGrid | Информация о сотрудниках | Ввод информации в Textbox для добавления или обновления данных или выбор одного из пунктов меню в форме информации о маршрутах |
| 17. | Ввод информации и нажатие кнопки отправить данные в БД | Добавление введенной информации в таблицу Employees в базе данных после проверки на ошибки | Информация о сотрудниках | Выбор одного из пунктов меню в форме информации о маршрутах |
| 18. | Ввод информации и нажатие кнопки обновить данные о обслуживании автобусов | Обновление данных в столбце с введенным ключом в таблице Employees в базе данных | Информация о сотрудниках | Выбор одного из пунктов меню в форме информации о маршрутах |
| 19. | Выбор пункта меню просмотр отчетов | Запрос пути к отдельно поставляемой программе просмотра отчетов и ее открытие | Информация о маршрутах. | Открытие и просмотр файла отчетов в программе просмотра отчетов либо выбор одного из пунктов меню в форме информации о маршрутах |
| 20. | Нажатие кнопки открытия файла | Открытие диалога выбора файла и загрузка файла отчета | Просмотр отчетов | Выбор одного из пунктов меню в форме информации о маршрутах |
| 21. | Нажатие кнопки First | Переход на первую страницу отчета | Просмотр отчетов | Выбор одного из пунктов меню в форме информации о маршрутах |
| 22. | Нажатие кнопки назад | Переход на предыдущую страницу отчета | Просмотр отчетов | Выбор одного из пунктов меню в форме информации о маршрутах |
| 23. | Нажатие кнопки вперед | Переход на следующую страницу отчета | Просмотр отчетов | Выбор одного из пунктов меню в форме информации о маршрутах |
| 24. | Нажатие кнопки Last | Переход на последнюю страницу отчета | Просмотр отчетов | Выбор одного из пунктов меню в форме информации о маршрутах |
| 25. | Нажатие кнопки + | Увеличение масштаба | Просмотр отчетов | Выбор одного из пунктов меню в форме информации о маршрутах |
| 26. | Нажатие кнопки - | Уменьшение масштаба | Просмотр отчетов | Выбор одного из пунктов меню в форме информации о маршрутах |
| 27. | Выбор пункта меню экспорт отчетов | Открытие формы экспорта отчетов | Информация о маршрутах. | Выбор одного из действий в форме экспорта отчетов или выбор одного из пунктов меню в форме информации о маршрутах |
| 28. | Нажатие кнопки экспортировать простой отчет | Подготовка и экспорт простого отчета в форматах .pdf и .frx | Экспорт отчетов | Выбор одного из действий в форме экспорта отчетов или выбор одного из пунктов меню в форме информации о маршрутах |
| 29. | Нажатие кнопки экспортировать сложный отчет | Подготовка и экспорт сложного отчета в форматах .pdf и .frx | Экспорт отчетов | Выбор одного из пунктов меню в форме информации о маршрутах |
| 30. | Выбор пункта меню о программе | Открытие формы о программе | Информация о маршрутах. | Закрытие программы |

**4 Используемые технические средства**

Программа для автоматизации работы автопарка автобусов функционирует на компьютерах с определенным аппаратным и программным обеспечением.

Минимальные требования для корректной работы системы:

* центральный процессор - 2 ГГц
* оперативная память - 2 Гб.
* свободное место на HDD - не менее 5 Гб.
* мышь.
* клавиатура.
* монитор 14 и выше дюймов.
* операционная система Windows 7
* 2 GB свободного пространства на жестком диске.

**5 Вызов и загрузка**

Для запуска приложения необходимо скопировать файлы программы на устройство