Содержание

[Реферат 5](#_Toc470072311)

[Введение 6](#_Toc470072312)

[1 Постановка задачи 7](#_Toc470072313)

[2 Проектирование 8](#_Toc470072314)

[2.1 Выбор архитектуры 8](#_Toc470072315)

[2.2 Выбор способа хранения информации 9](#_Toc470072316)

[2.3 Диаграмма вариантов использования 11](#_Toc470072317)

[2.3 Диаграмма последовательности 16](#_Toc470072318)

[2.4 Проектирование графического интерфейса 17](#_Toc470072319)

[3 Разработка 19](#_Toc470072320)

[3.1 Разработка диаграммы классов 19](#_Toc470072321)

[3.2 Диаграмма классов с отображением на архитектуру 21](#_Toc470072322)

[3.3 Описание сущностей 21](#_Toc470072323)

[3.4 Схемы алгоритмов для ключевых методов 25](#_Toc470072324)

[4 Верификация и тестирование программы 26](#_Toc470072325)

[4.1 Верификация программы 26](#_Toc470072326)

[4.2 Тестирование программы 26](#_Toc470072327)

[Выводы 29](#_Toc470072328)

[Перечень источников 30](#_Toc470072329)

[Приложение А. Техническое задание 31](#_Toc470072330)

[Приложение Б. Исходные тексты программы 36](#_Toc470072331)

[Приложение В. Руководство оператора 57](#_Toc470072332)

[Приложение Г. Презентация 61](#_Toc470072333)

Реферат

Пояснительная записка на курсовой проект: 67 с., 36 ил., 29 табл., 4 приложения.

Данная работа посвящена хранению паролей от учетных записей одного пользователя, результатом которой является приложение «Менеджер паролей».

Целью данной работы является программа, которая должна позволять создавать новые и открывать существующие базы данных паролей. База данных должна храниться в зашифрованном файле. В каждой базе данных должны быть записи. Каждая запись должна обладать встроенным базовым набором полей. В программе должна быть предусмотрена возможность добавления записей, а также изменения или удаления уже существующих.

Программу следует использовать на персональном компьютере (ПК). Выбор файла базы данных происходит через диалоговое окно для выбора пути к файлу с установленным фильтром для \*.savepass файлов. Приложение было реализовано с помощью языка программирования C# на платформе .NET Framework с использованием Windows Forms.

МЕНЕДЖЕР ПАРОЛЕЙ; УЧЕТНАЯ ЗАПИСЬ; ПАРОЛЬ; ВЕБ-СТРАНИЦА; ШИФРОВАНИЕ.

Введение

Сегодня приходится запоминать множество паролей. Вам потребуется пароль для электронной почты и локальных сетей, для домашней страницы и доступа по FTP, пароли в интернете (например, аккаунты на форумах или веб-сайтах) и т.д. Этот список бесконечен. Кроме того, необходимо использовать разные пароли для каждой учетной записи. Ведь если вы используете только один пароль и везде его применяете, то злоумышленник, зная только один пароль, сможет получить доступ ко всем Вашим аккаунтам.

1 Постановка задачи

Задачей данной курсовой работы является реализация так называемого «менеджера паролей».

Менеджер паролей – это программа, которая поможет вам управлять вашими паролями и сохранит их в безопасности. Вы сможете поместить все ваши пароли в одну базу данных, которая надежно закрыта единственным мастер-паролем. Поэтому, необходимо запомнить, только один ключ, чтобы управлять всей базой данных. База данных, находится в зашифрованном виде с применением одного из самых лучших и безопасных, из известных в настоящее время, алгоритмов шифрования, AES.

Разработанная программа должна позволять:

1. Действия с базой данных:

* создать базу данных;
* открыть базу данных;
* сохранить базу данных;
* задать имя базы данных;
* задать пароль базы данных.

1. Действия с записями базы данных:

* создать запись;
* редактировать запись;
* удалить запись;
* копировать поля записи в буфер обмена;
* перейти на веб-страницу, которая указана в записи.

Каждая запись должна иметь следующие поля:

* имя записи;
* имя пользователя;
* пароль;
* электронная почта, привязанная к аккаунту;
* веб-страница;
* комментарии.

2 Проектирование

2.1 Выбор архитектуры

Архитектура – совокупность важнейших решений об организации программной системы.

Архитектура включает в себя:

1. выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов;
2. соединение выбранных элементов структуры и поведения во всё более крупные системы;
3. архитектурный стиль, который направляет всю организацию – все элементы, их интерфейсы, их сотрудничество и их соединение.

Существует три типа системной архитектуры:

1. одноуровневая (монолитная);
2. двухуровневая;
3. многоуровневая (распределённая).

В одноуровневой системной архитектуре все действия реализации графического интерфейса и решения задачи выполняются на одном уровне системы.

В двухуровневой архитектуре компоненты, отвечающие за графический интерфейс и прикладную логику, размещаются на клиентской машине и обращаются к общему источнику данных по сети. В таких приложениях пользовательский интерфейс и прикладная логика образуют один уровень на клиентском компьютере, а сервисы данных предоставляются отдельными приложениями (на серверах). Этот сценарий часто встречается в клиент-серверных приложениях.

В многоуровневой архитектуре презентационный сервис (первый уровень), прикладная логика (второй уровень) и сервис данных (третий уровень) отделены друг от друга. В многоуровневом приложении клиент предоставляет только пользовательский интерфейс. Прикладная логика реализуется промежуточным уровнем, который размещается между пользовательским интерфейсом и системой хранения данных.

Разрабатываемая программа должна реализовывать определенный алгоритм – бизнес логику, взаимодействовать с пользователем через интерфейс, и доступ к базам данных реализуется файловой системой операционной системы. Поэтому для решения поставленной задачи была выбрана двухуровневая системная архитектура.

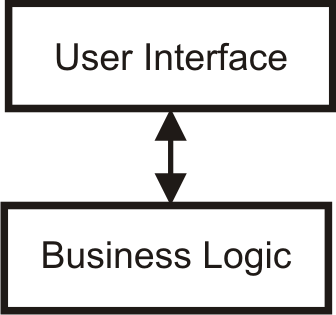


Рисунок 2.1 – Двухуровневая архитектура

2.2 Выбор способа хранения информации

Одной из задач при разработке программы является сохранение результата в зашифрованный файл. Поэтому выбрано такое решение: сериализация выходных данных в JSON-текст с последующим щифрованием с помощью криптостойкого алгоритма AES (Rijndael).

**JSON** (англ. **JavaScript Object Notation**) – текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми. Формат JSON был разработан Дугласом Крокфордом.

JSON-текст представляет собой одну из двух структур:

* набор пар ключ: значение. В различных языках это реализовано как объект, запись, структура, словарь, хэш-таблица, список с ключом или ассоциативный массив. Ключом может быть только строка, значением – любая форма;
* упорядоченный набор значений. Во многих языках это реализовано как массив, вектор, список или последовательность.

В качестве значений в JSON могут быть использованы:

* объект – это неупорядоченное множество пар ключ:значение, заключённое в фигурные скобки «{}». Пары ключ-значение отделяются друг от друга запятыми;
* массив (одномерный) – это упорядоченное множество значений, заключенное в квадратные скобки «[]». Значения разделяются запятыми;
* число;
* литералы true, false и null;
* строка – это упорядоченное множество из нуля или более символов юникода, заключенное в двойные кавычки.

Пример JSON-текста:

{

"firstName": "Иван",

"lastName": "Иванов",

"address": {

"streetAddress": "Московское ш., 101, кв.101",

"city": "Ленинград",

"postalCode": 101101

},

"phoneNumbers": [

"812 123-1234",

"916 123-4567"

]

}

**Advanced Encryption Standard (AES)**, также известный как **Rijndael** – симметричный алгоритм блочного шифрования, принятый в качестве стандарта шифрования правительством США по результатам конкурса AES. Этот алгоритм хорошо проанализирован и сейчас широко используется. Национальный институт стандартов и технологий США опубликовал спецификацию AES 26 ноября 2001 года после пятилетнего периода, в ходе которого были созданы и оценены 15 кандидатур. 26 мая 2002 года AES был объявлен стандартом шифрования. По состоянию на 2009 год AES является одним из самых распространённых алгоритмов симметричного шифрования.

2.3 Диаграмма вариантов использования

Диаграммы вариантов использования (Use Case) показывают взаимодействия между вариантами использования и действующими лицами, отражая функциональные требования к системе с точки зрения пользователя.

Цель построения – документирование функциональных требований в общем виде (требование – простота). Вариант использования – это последовательность действий (транзакций), выполняемых системой в ответ на событие, инициируемое некоторым внешним объектом (действующим лицом). Вариант использования описывает типичное взаимодействие между пользователем и системой и отражает представление о поведении системы с точки зрения пользователя.

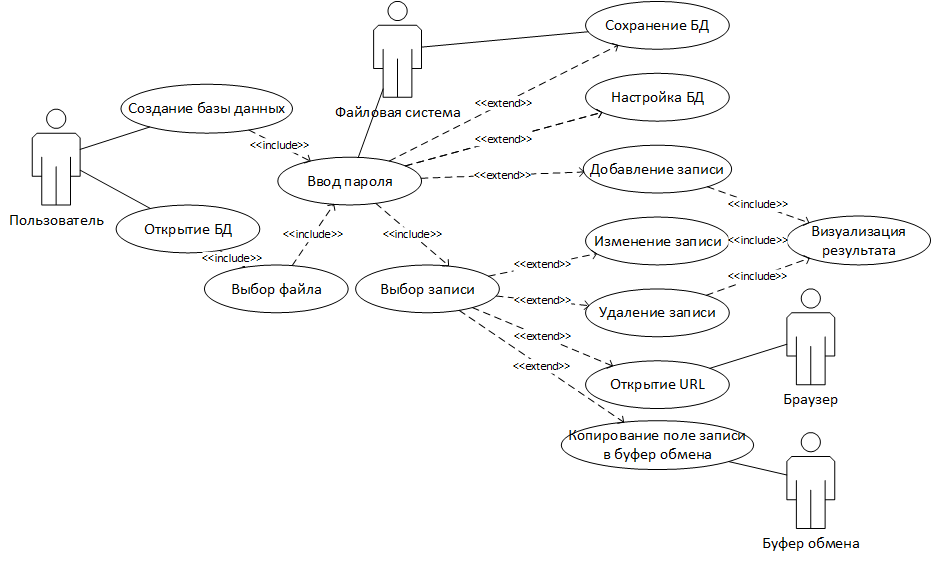


Рисунок 2.2 – Диаграмма вариантов использования

Исходя из анализа построенной диаграммы прецедентов, а также функций, которые должна выполнять программа, были предусмотрены следующие варианты использования.

Таблица 2.1 – Прецедент «Создание базы данных»

|  |  |
| --- | --- |
| ID | 1 |
| Название | Создание базы данных |
| Участники | Пользователь |
| Краткое описание | Создание пустой базы данных в программе |
| Входные условия | Нет |
| Выходные условия | Выбран пункт создание файла |
| Предусловия | Нет |
| Порядок действия | Пользователь нажимает на кнопку создания файла |
| Приоритет | Высокий |

Таблица 2.2 – Прецедент «Открытие БД»

|  |  |
| --- | --- |
| ID | 2 |
| Название | Открытие БД |
| Участники | Пользователь |
| Краткое описание | Открытие базы данных в программе |
| Входные условия | Нет |
| Выходные условия | Выбран пункт открытия файла |
| Предусловия | Нет |
| Порядок действия | Пользователь нажимает на кнопку открытия файла |
| Приоритет | Высокий |

Таблица 2.3 – Прецедент «Выбор файла»

|  |  |
| --- | --- |
| ID | 3 |
| Название | Выбор файла |
| Участники | Пользователь |
| Краткое описание | Открытие базы данных в программе |
| Входные условия | Нет |
| Выходные условия | Файл выбран |
| Предусловия | Выбран пункт открытия файла |
| Порядок действия | Пользователь выбирает путь к файлу с помощью диалога открытия файла |
| Приоритет | Высокий |

Таблица 2.4 – Прецедент «Ввод пароля»

|  |  |
| --- | --- |
| ID | 4 |
| Название | Ввод пароля |
| Участники | Пользователь |
| Краткое описание | Ввод пароля для базы данных |
| Входные условия | Нет |
| Выходные условия | Введен пароль от базы данных |
| Предусловия | Файл выбран / Выбран пункт создание файла |
| Порядок действия | Пользователь вводит пароль в соответствующую форму |
| Приоритет | Высокий |

Таблица 2.5 – Прецедент «Сохранение БД»

|  |  |
| --- | --- |
| ID | 5 |
| Название | Сохранение БД |
| Участники | Пользователь |
| Краткое описание | Сохранение базы данных в файл |
| Входные условия | Файл создан или открыт |
| Выходные условия | Нет |
| Предусловия | Нет |
| Порядок действия | Пользователь нажимает на кнопку сохранения файла |
| Приоритет | Высокий |

Таблица 2.6 – Прецедент «Настройка БД»

|  |  |
| --- | --- |
| ID | 6 |
| Название | Настройка БД |
| Участники | Пользователь |
| Краткое описание | Изменение имени базы данных и пароля |
| Входные условия | Файл создан или открыт |
| Выходные условия | Нет |
| Предусловия | Нет |
| Порядок действия | Пользователь нажимает на кнопку настройки БД |
| Приоритет | Высокий |

Таблица 2.7 – Прецедент «Добавление записи»

|  |  |
| --- | --- |
| ID | 7 |
| Название | Добавление записи |
| Участники | Пользователь |
| Краткое описание | Добавление записи в базу данных |
| Входные условия | Файл создан или открыт |
| Выходные условия | Запись добавлена |
| Предусловия | Нет |
| Порядок действия | Пользователь нажимает на кнопку добавления записи |
| Приоритет | Высокий |

Таблица 2.8 – Прецедент «Изменение записи»

|  |  |
| --- | --- |
| ID | 8 |
| Название | Изменение записи |
| Участники | Пользователь |
| Краткое описание | Изменение записи в базе данных |
| Входные условия | Запись выбрана |
| Выходные условия | Визуализировать результат |
| Предусловия | Нет |
| Порядок действия | Пользователь нажимает на кнопку изменения записи |
| Приоритет | Высокий |

Таблица 2.9 – Прецедент «Удаление записи»

|  |  |
| --- | --- |
| ID | 9 |
| Название | Удаление записи |
| Участники | Пользователь |
| Краткое описание | Удаление записи из базы данных |
| Входные условия | Запись выбрана |
| Выходные условия | Визуализировать результат |
| Предусловия | Нет |
| Порядок действия | Пользователь нажимает на кнопку удаления записи |
| Приоритет | Высокий |

Таблица 2.10 – Прецедент «Выбор записи»

|  |  |
| --- | --- |
| ID | 10 |
| Название | Выбор записи |
| Участники | Пользователь |
| Краткое описание | Выбор записи из списка |
| Входные условия | Файл создан или открыт, хотя бы одна запись существует |
| Выходные условия | Визуализировать результат |
| Предусловия | Нет |
| Порядок действия | Пользователь выбирает запись из списка |
| Приоритет | Высокий |

Таблица 2.11 – Прецедент «Открытие URL»

|  |  |
| --- | --- |
| ID | 11 |
| Название | Открытие URL |
| Участники | Пользователь |
| Краткое описание | Открытие URL-адреса, указанного в записи |
| Входные условия | Запись выбрана |
| Выходные условия | Открывается браузер с указанным URL |
| Предусловия | Нет |
| Порядок действия | Пользователь нажимает на кнопку открытия URL |
| Приоритет | Высокий |

Таблица 2.12 – Прецедент «Копирование поля записи в буфер обмена»

|  |  |
| --- | --- |
| ID | 12 |
| Название | Копирование поле в буфер обмена |
| Участники | Пользователь |
| Краткое описание | Копирование поле записи в буфер обмена |
| Входные условия | Запись выбрана |
| Выходные условия | Копируется поле записи в буфер обмена |
| Предусловия | Нет |
| Порядок действия | Пользователь нажимает на кнопку копирования поля записи в буфер обмена |
| Приоритет | Высокий |

Таблица 2.13 – Прецедент «Визуализация результата»

|  |  |
| --- | --- |
| ID | 13 |
| Название | Визуализация результата |
| Участники | Пользователь |
| Краткое описание | Обновляется информация о записи |
| Входные условия | Произошли изменения в списке записей |
| Выходные условия | Нет |
| Предусловия | Нет |
| Порядок действия | Нет |
| Приоритет | Высокий |

2.3 Диаграмма последовательности

Для того, чтобы показать последовательность выполнение задачи приложения, была составлена диаграмма последовательности, которая представлена ниже.

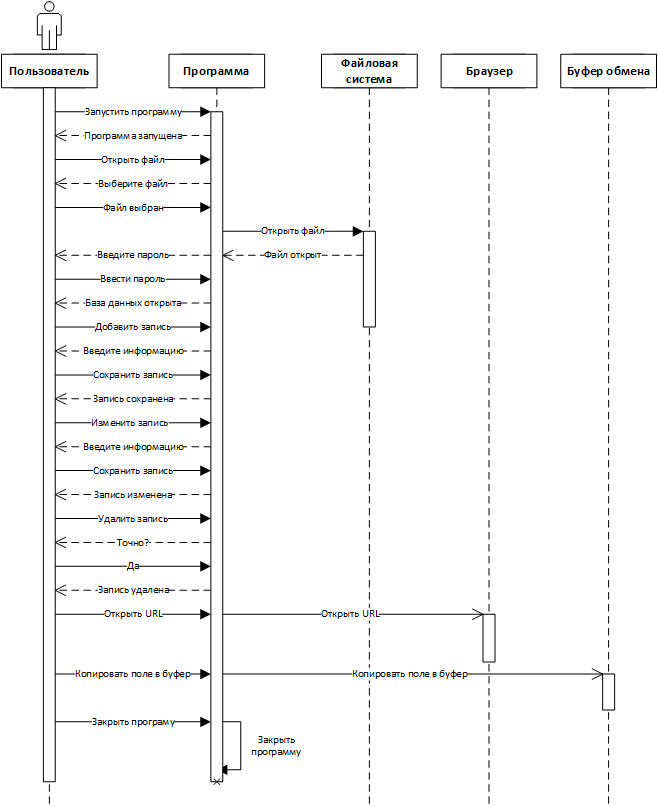


Рис. 2.3. Диаграмма последовательности

2.4 Проектирование графического интерфейса

Исходя из анализа поставленной задачи, выяснилось, что графический интерфейс разрабатываемого программного продукта должен содержать: меню, панель инструментов, область для вывода списка записей, а также  
информацию о текущей записи.

Ввод входных данных выполняется с помощью дополнительных диалоговых окон (окно ввода пароля, окно настройки базы данных).

Диаграмма навигации между окнами представлена на рисунке ниже.

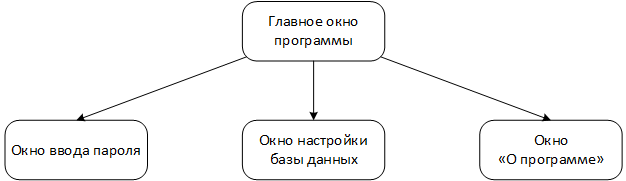


Рис. 2.4. Диаграмма навигации между окнами

Уточненный графический интерфейс пользователя разрабатываемого  
программного продукта изображен на рисунке 2.5.

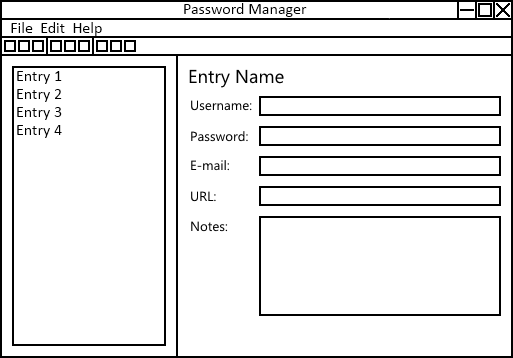


Рис. 2.5. Главное окно программы

Главное меню должно включать такие пункты как:

1. Файл:
   1. Создать;
   2. Открыть;
   3. Закрыть;
   4. Сохранить;
   5. Сохранить как;
   6. Настройки базы данных;
   7. Выйти.
2. Правка:
   1. Добавить запись;
   2. Редактировать запись;
   3. Удалить запись.
3. Помощь:
   1. О программе.

3 Разработка

3.1 Разработка диаграммы классов

Диаграмма классов (class diagram) служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений. На данной диаграмме не указывается информация о временных аспектах функционирования системы. С этой точки зрения диаграмма классов является дальнейшим развитием концептуальной модели проектируемой системы.

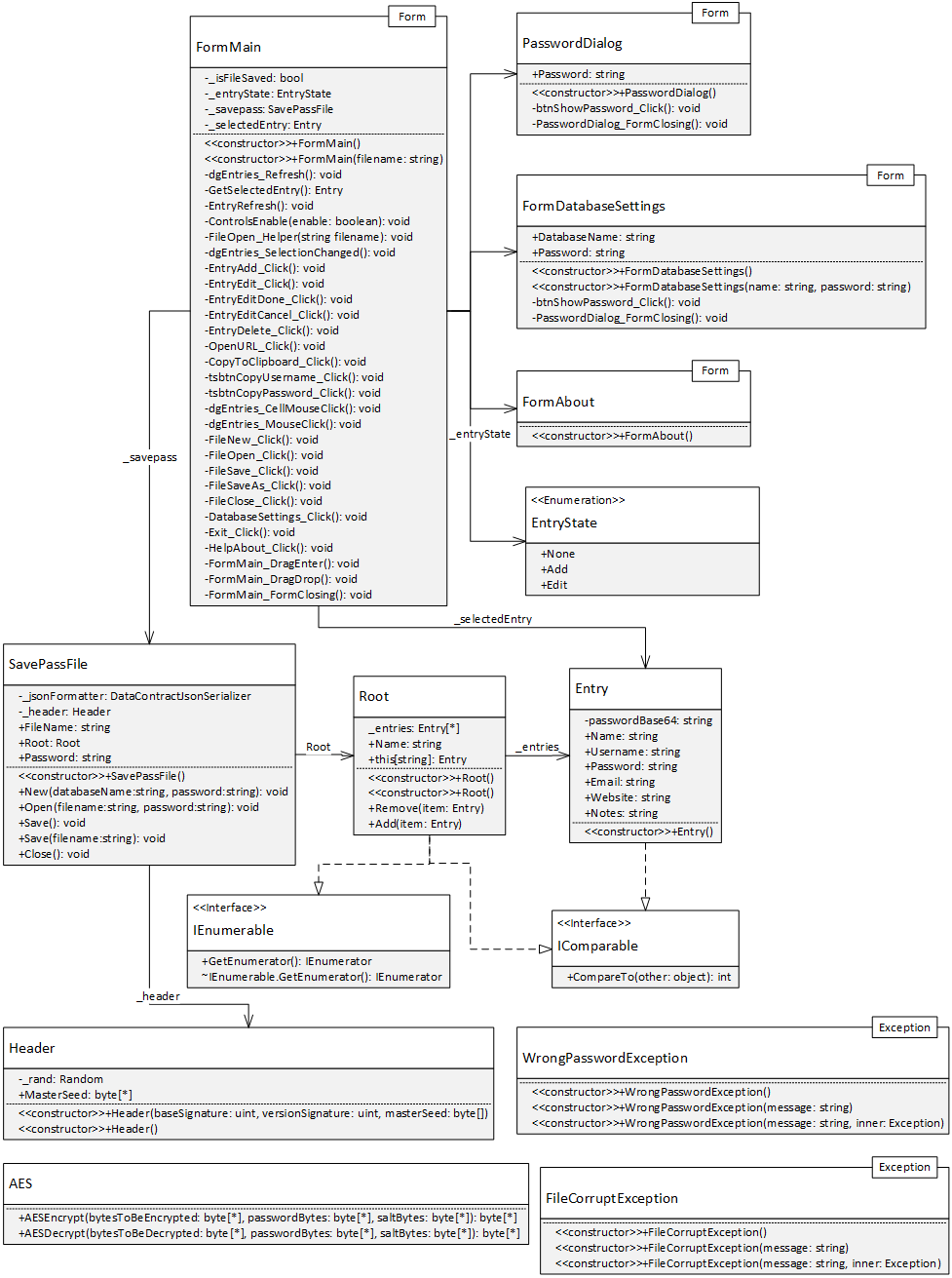


Рисунок 3.1 – Диаграмма классов

3.2 Диаграмма классов с отображением на архитектуру

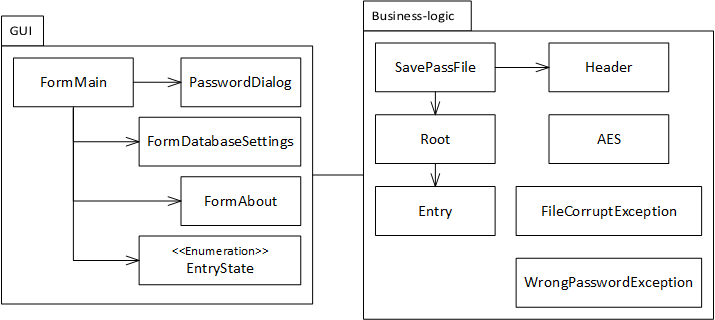


Рисунок 3.2 – Диаграмма классов с отображением на архитектуру

3.3 Описание сущностей

Таблица 3.1 – Основные сущности

|  |  |
| --- | --- |
| **Сущность** | **Описание** |
| SavePassFile | Работа с файловой системой |
| Entry | Запись |
| Root | Список записей |
| Header | Заголовок файла |
| AES | Методы для шифрования данных |
| FormMain | Главное окно программы |
| PasswordDialog | Окно запроса пароля. Выполняется при открытии файла |
| FormDatabaseSettings | Окно настройки базы данных (имя БД, пароль) |
| FornAbout | Окно «О программе» |

Таблица 3.2 – Класс SavePassFile

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| \_jsonFofmatter | JSON-сериализатор |
| \_header | заголовок. Первые 16 байт файла |
| FileName | имя открытого файла |
| Root | объект класса Root |
| Password | пароль базы данных |
| **Метод** | **Описание** |
| SavePassFile | Конструктор класса SavePassFile |
| New | Создаёт базы данных с заданным именем и паролем |
| Open | Открывает существующий файл SavePass |
| Save | Сохраняет открытый файл |
| Close | Закрывает открытый файл |

Таблица 3.3 – Класс Entry

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| \_passwordBase64 | Пароль в кодировке base64 |
| Name | Имя записи |
| Username | Имя пользователя |
| Password | Пароль |
| Email | Электронная почта |
| Website | Адрес веб-сайта |
| Notes | Комментарии к записи |
| **Метод** | **Описание** |
| Entry | Инициализирует пустую запись с заданным именем |
| CompareTo | Реализация интерфейса IComparable |

Таблица 3.4 – Класс Root

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| \_entries | Список полей |
| Name | Имя базы данных |
| this[] | Индексатор класса Entry |
| **Метод** | **Описание** |
| Root | Инициализирует класс Root |
| Add | Добавляет запись в базу данных |
| Remove | Удаляет запись с базы данных |
| CompareTo | Реализация интерфейса IComparable |
| GetEnumerator | Реализация интерфейса IEnumerable |

Таблица 3.5 – Класс Header

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| BaseSignature | Сигнатура программы |
| VersionSignature | Сигнатура версии |
| MasterSeed | Соль |

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| Header | Инициализирует класс Header |

Таблица 3.6 – Класс Header

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| AESEncrypt | Возвращает данные в зашифрованном виде |
| AESDecrypt | Возвращает данные в разшифрованном виде |

Таблица 3.7 – Класс FormMain

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| \_isFileSaved | Показывает, сохранены ли изменения в файле. |
| \_entryState | Показывает текущий режим отображения: просмотр, создание или редактирование записи |
| \_savepass | Объект класса SavePassFile |
| \_selectedEntry | Текущая выделенная запись в DataGridView |
| **Метод** | **Описание** |
| FormMain | Инициализирует главную форму |
| dgEntries\_Refresh | Обновляет список записей в DataGridView |
| GetSelectedEntry | Возвращает текущую выделенную запись в DataGridView |
| EntryRefresh | Обновить информацию в форме о текущей записи |
| ControlsEnable | Активирует/деактивирует некоторые элементы управления на форме |
| FileOpen\_Helper | Метод-помощник для открытия файла |
| **Обработчик события** | **Описание** |
| dgEntries\_SelectionChanged | Возникает при изменении выделения в DataGridView |
| EntryAdd\_Click | Перейти в режим создания записи |
| EntryEdit\_Click | Перейти в режим редактирования записи |
| EntryEditDone\_Click | Успешно завершить создания/редактирования записи |
| EntryEditCalcel\_Click | Отменить создание/редактирование записи |
| EntryDelete\_Click | Удалить записи |
| OpenURL\_Click | Открыть URL-адрес текущей записи |
| CopyToClipboard\_Click | Копировать поле текущей записи в буфер обмена |
| tsbtn CopyUsername\_Click | Копировать имя пользователя записи в буфер обмена |
| tsbtn CopyPassword\_Click | Копировать пароль записи в буфер обмена |
| dgEntries\_CellMouseClick | Открыть контекстное меню при клике на элемент DataGridView |
| dgEntries\_MouseClick | Открыть контекстное меню при клике на пустое место в DataGridView |
| FileNew\_Click | Создать базу данных |
| FileOpen\_Click | Открыть базу данных из файла |
| FileSave\_Click | Сохранить базу данных в файл |
| FileSaveAs\_Click | Сохранить базу данных в файл с другим путём |
| FileClose\_Click | Закрыть базу данных |
| DatabaseSettings\_Click | Открыть окно настроек базы данных |
| Exit\_Click | Закрыть программу |
| HelpAbout\_Click | Открыть окно «О программе» |
| FormMain\_DragEnter | Разрешить перетаскивание файла на окно |
| FormMain\_DragDrop | Открыть файл, перетащенный на окно |
| FormMain\_FormClosing | Закрыть программу |

Таблица 3.8 – Класс PasswordDialog

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| Password | Пароль для возврата |
| **Обработчик события** | **Описание** |
| PasswordDialog | Инициализирует форму |
| btnShowPassword\_Click | Показать/скрыть пароль |
| PasswordDialog\_FormClosing | Закрыть форму |

Таблица 3.9 – Класс FormDatabaseSetting

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| DatabaseName | Имя базы данных |
| Password | Пароль для возврата |
| **Обработчик события** | **Описание** |
| FormDatabaseSetting | Инициализирует форму |
| btnShowPassword\_Click | Показать/скрыть пароль |
| FormClosing | Закрыть форму |

3.4 Схемы алгоритмов для ключевых методов

Схемы алгоритмов основных методов («открыть файл», «добавить запись», «создать БД», «сохранить файл») указаны на рисунке 3.3.

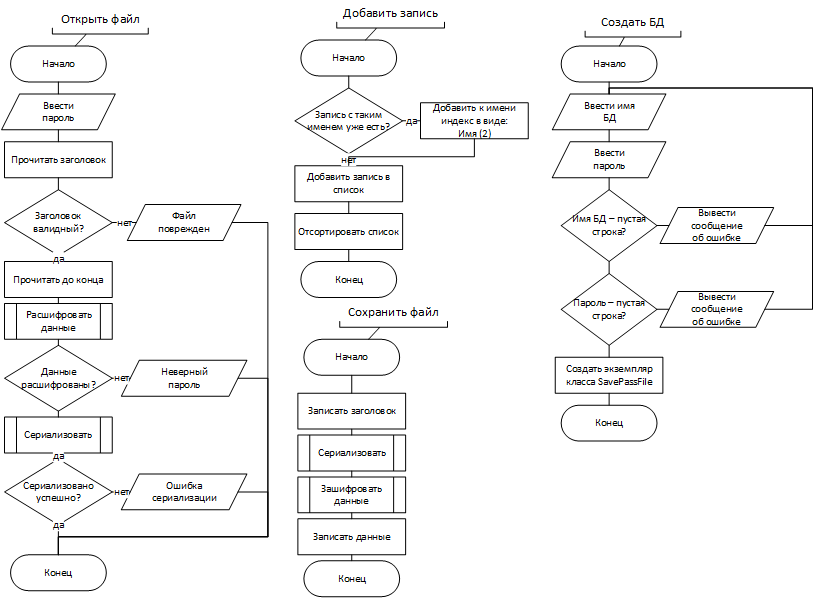


Рисунок 3.3 – Схемы алгоритмов для методов «открыть файл», «добавить запись», «создать БД», «сохранить файл».

4 Верификация и тестирование программы

4.1 Верификация программы

Верификация – проверка, способ подтверждения каких-либо теоретических положений, алгоритмов, программ и процедур путем их сопоставления с опытными (эталонными или эмпирическими) данными, алгоритмами и программами.

В частности, верификация проверяет соответствие между нормами стандартов, описанием требований (техническим заданием) к ПО, проектными решениями, исходным кодом, пользовательской документацией и функционированием самого ПО.

Верификацию будем выполнять при помощи опросного листа, путём анализа из проектной документации (ТЗ, ПЗ), а также кодов программы.

Результаты верификации программы приведены в таблице 4.1.

Таблица 4.1 – Верификация и её результаты

|  |  |  |
| --- | --- | --- |
| Требование в ТЗ | Метод | Где реализован |
| создание файла | New | SavePassFile |
| открытие файла | Open | SavePassFile |
| сохранение файла | Save | SavePassFile |
| добавление записи | Add | Root |
| редактирование записи | EntryEdit\_Click | FormMain |
| удаление записи | Remove | Root |
| копирование поля записи в буфер обмена | CopyToClipboard\_Click | FormMain |
| открытие URL в браузере | OpenURL\_Click | FormMain |

4.2 Тестирование программы

Таблица 4.1 – Неправильность пароля

|  |  |
| --- | --- |
| Действие | Открытие файла, ввод неправильного пароля |
| Ожидаемый результат | Сообщение об неправильности пароля |
| Полученный результат |  |

Таблица 4.1 – Поврежденность файла

|  |  |
| --- | --- |
| Действие | Открыть поврежденный файл |
| Ожидаемый результат | Сообщение об поврежденности файла |
| Полученный результат |  |

Таблица 4.2 – Успешное открытие файла

|  |  |
| --- | --- |
| Действие | Открытие файла, ввод правильного пароля |
| Ожидаемый результат | Окно программы в открытой базой данных |
| Полученный результат |  |

Таблица 4.3 – Успешное создание файла

|  |  |
| --- | --- |
| Действие | Создание новой БД и сохранение в файл |
| Ожидаемый результат | Созданный файл |
| Полученный результат |  |

Таблица 4.4 – Проверка корректности сохранения

|  |  |
| --- | --- |
| Действие | Создать файл, добавить запись, сохранить БД в файл, закрыть программу, открыть этот же файл. |
| Ожидаемый результат | Файл успешно открылся |
| Полученный результат |  |

Выводы

В результате проделанной работы (разработки и отладки программы, тестирования и анализа результатов) было создано программное обеспечение, позволяющее поместить все ваши пароли в одну базу данных, которая надежно закрыта единственным мастер-паролем.

Для сериализации данных использовался JSON-сериализатор. Для шифрования файла использовался криптостойкий алгоритм шифрования AES (Rijndael).

Во время тестирования не было обнаружено каких-то серьезных недостатков в работе программы. Программа готова к использованию.

Возможные пути развития программы:

* расширение интерфейса;
* возможность добавлять свои поля;
* поддержка группы паролей;
* корзина удаленных записей;
* поиск;
* встроенный генератор случайных паролей;
* мультиязычность;
* экспорт данных в TXT, HTML, XML, CSV файлы;
* импорт из файлов других форматов;
* добавление новых алгоритмов шифрования (например, Twofish);
* защита паролей In-Memory.

Перечень источников

1. Троелсен Эндрю: Язык программирования C# 5.0 и платформа .NET, 6-е изд.: Пер. с англ. – М.: ООО «И.Д Вильямс». 2013. – 1312 с.
2. Порядок оформления учебных и научно-исследовательских документов / В.Н. Павленко, А.С. Набатов, И.М. Тараненко. – Учеб. пособие. – Харьков: Нац. аэрокосм. ун-т «Харьк. авиац. ин-т», 2007. – 65 с.
3. Программное обеспечение для компьютерных систем и сетей. Разработка технического задания / В.И. Дужий, А.В. Волковой, А.А. Волковая, А.А. Галькевич, А.С. Годунов. – Харьков: Нац. аэрокосм. ун-т «Харьк. авиац. ин-т», 2007. – 109 с.
4. Официальный сайт professorweb.ru. [Электронный ресурс]. – Режим доступа <http://professorweb.ru/> (дата обращения: 13.11.2016).
5. Официальный сайт codeproject.com. [Электронный ресурс]. – Режим доступа <http://codeproject.com> (дата обращения: 13.11.2016).

Приложение А. Техническое задание

1 Введение

1.1 Наименование программы

Наименование – «Менеджер паролей» (далее по тексту – Программа).

1.2. Краткая характеристика области применения

Данная программа предназначена для личного использования. Цель программы заключается в создании базы данных паролей.

2 Основания для разработки

2.1 Основания для разработки

Основанием для разработки Программы является задание на курсовой проект кафедры "Компьютерные системы и сети" (503) Национального аэрокосмического университета им. Н.Е. Жуковского "ХАИ".

2.2 Наименование и условное обозначение разработки

Наименование темы разработки – «Менеджер паролей».

Условное обозначение темы разработки (шифр темы) - "SavePass".

3 Назначение разработки

3.1 Функциональное назначение программы

Цель программы заключается в создании базы данных паролей для множества аккаунтов пользователя(ей). Она должна управление записями паролей (создание, изменение, удаление).

3.2 Эксплуатационное назначение программы

Программа работает в ручном режиме, начинает работать сразу после ее включения и до момента завершения работы программы.

Программа может функционировать в офисе и домашних условиях работы, т.е. в условиях работы настольного ПК.

Программа может работать в необслуживаемом режиме. Квалификация персонала – рядовой пользователь ПК.

Программа устанавливается путем копирования ее файлов на нужный ПК.

Отказ программы не влечет за собой критических последствий.

4 Требования к программе или программному изделию

4.1 Требования к функциональным характеристикам

4.1.1 Требования к составу выполняемых функций

В рамках задания программа должна выполнять следующие функции:

* создание базы данных;
* редактирование базы данных;
* сохранение базы данных на жесткий диск;
* создание записи;
* редактирование записи;
* удаление записи.

4.1.2 Требования к организации входных данных

Входные данные должны вводиться с клавиатуры при помощи GUI.

4.1.3 Требования к организации выходным данных

Выходными данными является файл с расширением «savepass».

4.1.4 Требования к временным характеристикам

Требования к временным характеристикам не предъявляются.

4.2 Требования к надежности

Данный программный продукт является прикладной программой, не работающей в системе реального времени или в области критических технологий, на выполнение запускается однократно и функционирует только в необходимый момент времени. Поэтому специальные требования к надежности программы не предъявляются.

4.3. Условия эксплуатации

4.3.1 Требования к видам обслуживания

После установки на компьютер программа не требует проведения каких-либо видов обслуживания.

4.3.2 Требования к численности и квалификации персонала

Для пользования программой хватает одной штатной единиц – конечный пользователь программы (оператор).

Программа устанавливается путем копирования на магнитный носитель. Для установки программы и ее эксплуатации необходим пользователь с обычными правами доступа, и обладающий навыками работы на компьютере.

Конечный пользователь программы (оператор) должен понимать, каким образом необходимо работать с GUI программы, т.е. должен иметь представление о назначении программы.

Персонал должен быть аттестован на II квалификационную группу по электробезопасности (для работы с конторским оборудованием).

4.4 Требования к составу и параметрам технических средств

Для исполнения программы необходим универсальный компьютер любого типа, для которого получен двоичный код, представленный в виде исполнимого файла (.exe-файла), способного выполняться установленным в ПК процессором.

В состав технических средств должен входить:

В состав технических средств должен входить персональный компьютер (ПЭВМ), включающий в себя:

1. процессор с тактовой частотой, не менее 1.5 ГГц;
2. материнскую плату с FSB, не менее 133 МГц;
3. оперативную память объемом не менее 512 Мб;
4. свободного дискового пространства для хранения программы, не менее 20 Мб;
5. дисплей и видеоадаптер такого типа, которые смогут отображать графическую информацию, являющуюся результатом выполнения программы;
6. клавиатуру;
7. мышь;
8. если предполагается работа со сменными дисками, необходимо наличие USB-адаптера.

4.4 Требования к информационной и программной совместимости

4.4.1 Требования к информационным структурам и методам решения

Формат текста вводимых и выводимых в программе: ".savepass".

4.4.2 Требования к исходным кодам и языкам программирования

Для разработки функциональной части программы в качестве языка программирования нужно использовать C#. Для разработки клиентской части следует использовать технологию Windows Forms.

Для разработки функциональной и клиентской части программы следует использовать интегрированную среду разработки Microsoft Visual Studio 2015.

Исходные тексты программ должны содержать достаточное количество комментариев, позволяющих выполнять их верификацию и валидацию, а также контролировать стиль программирования программиста.

Дополнительные настройки компиляторов не требуются.

4.4.3 Требования к программным средствам, используемым программой

Программа должна быть реализована в виде приложения, исполняемого под управлением ОС Windows.

Также на ПК должна быть установлена программная платформа .NET Framework версии 4.0 или новее.

Для функционирования программы на компьютере должно быть установлено следующие ПО.

4.4.4 Требования к защите информации и программ

Программа должна шифровать выходной файл базы данных криптостойким алгоритмом Rijndael.

4.6 Специальные требования

Программа должна быть реализована с использованием Windows Forms.

Программа должна осуществлять взаимодействие с пользователем при помощи GUI.

5 Требования к программной документации

5.1 Предварительный состав программной документации

В результате разработки программы должна быть представлена программная документация:

1. техническое задание;
2. текст программы;
3. инструкция пользователя;
4. схемы алгоритмов;
5. результаты тестирования и верификации.

Кроме программного обеспечения на диске обязательно должна находиться пояснительная записка в электронном виде, содержащая весь комплект документации, предусмотренный в пункте 5 настоящего технического задания.

6 Стадии и этапы разработки

6.1 Стадии разработки

Разработка должна быть проведена в три стадии:

1. разработка технического задания;
2. рабочее проектирование;
3. внедрение.

6.2 Этапы разработки

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены следующие этапы работ:

1. разработка программы;
2. разработка программной документации;
3. испытания программы.

На стадии внедрения должен быть выполнен этап разработки - подготовка и передача программы.

6.3 Содержание работ по этапам

На этапе разработки технического задания должны быть выполнены такие работы:

1. постановка задачи;
2. определение и уточнение требований к техническим средствам;
3. определение требований к программе;
4. определение стадий, этапов и сроков разработки программы и документации на неё;
5. выбор языков программирования;
6. согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с ГОСТ 19.101-77 с требованием п. 5.1 настоящего технического задания.

На этапе испытаний программы должны быть выполнены следующие виды работ:

* разработка, согласование и утверждение программы и методики испытаний;
* проведение приемосдаточных испытаний;
* корректировка программы и программной документации по результатам испытаний.

На этапе внедрения программы должна быть выполнена работа по подготовке и передаче программы и программной документации.

Приложение Б. Исходные тексты программы

**Текст файла «Program.cs»**

using System;

using System.Windows.Forms;

using SavePass.Forms;

namespace SavePass

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main(string[] args)

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

switch (args.Length)

{

case 1:

Application.Run(new FormMain(args[0]));

break;

default:

Application.Run(new FormMain());

break;

}

}

}

}

**Текст файла «AES.cs»**

using System.IO;

using System.Security.Cryptography;

namespace SavePass.Crypto

{

class AES

{

public static byte[] AESEncrypt(byte[] bytesToBeEncrypted, byte[] passwordBytes, byte[] saltBytes)

{

byte[] encryptedBytes = null;

using (MemoryStream ms = new MemoryStream())

{

using (RijndaelManaged AES = new RijndaelManaged())

{

AES.KeySize = 256;

AES.BlockSize = 128;

var key = new Rfc2898DeriveBytes(passwordBytes, saltBytes, 1000);

AES.Key = key.GetBytes(AES.KeySize / 8);

AES.IV = key.GetBytes(AES.BlockSize / 8);

AES.Mode = CipherMode.CBC;

using (var cs = new CryptoStream(ms, AES.CreateEncryptor(), CryptoStreamMode.Write))

{

cs.Write(bytesToBeEncrypted, 0, bytesToBeEncrypted.Length);

cs.Close();

}

encryptedBytes = ms.ToArray();

}

}

return encryptedBytes;

}

public static byte[] AESDecrypt(byte[] bytesToBeDecrypted, byte[] passwordBytes, byte[] saltBytes)

{

byte[] decryptedBytes = null;

using (MemoryStream ms = new MemoryStream())

{

using (RijndaelManaged AES = new RijndaelManaged())

{

AES.KeySize = 256;

AES.BlockSize = 128;

var key = new Rfc2898DeriveBytes(passwordBytes, saltBytes, 1000);

AES.Key = key.GetBytes(AES.KeySize / 8);

AES.IV = key.GetBytes(AES.BlockSize / 8);

AES.Mode = CipherMode.CBC;

using (var cs = new CryptoStream(ms, AES.CreateDecryptor(), CryptoStreamMode.Write))

{

cs.Write(bytesToBeDecrypted, 0, bytesToBeDecrypted.Length);

cs.Close();

}

decryptedBytes = ms.ToArray();

}

}

return decryptedBytes;

}

}

}

**Текст файла «FileCorruptException.cs»**

using System;

namespace SavePass.Exceptions

{

public class FileCorruptException : Exception

{

public FileCorruptException() { }

public FileCorruptException(string message) : base(message) { }

public FileCorruptException(string message, Exception inner) : base(message, inner) { }

}

}

**Текст файла «WrongPasswordException.cs»**

using System;

namespace SavePass.Exceptions

{

public class WrongPasswordException : Exception

{

public WrongPasswordException() { }

public WrongPasswordException(string message) : base(message) { }

public WrongPasswordException(string message, Exception inner) : base(message, inner) { }

}

}

**Текст файла «Entry.cs»**

using System;

using System.Runtime.Serialization;

namespace SavePass.Nodes

{

/// <summary>

/// Класс записи паролей

/// </summary>

[DataContract]

public class Entry : IComparable<Entry>

{

/// <summary>

/// Пароль в кодировке base64

/// </summary>

[DataMember(Name = "Password")]

private string \_passwordBase64;

/// <summary>

/// Имя записи

/// </summary>

[DataMember]

public string Name { get; set; }

/// <summary>

/// Имя пользователя

/// </summary>

[DataMember]

public string Username { get; set; }

/// <summary>

/// Пароль

/// </summary>

public string Password

{

get

{

return System.Text.Encoding.UTF8.GetString(Convert.FromBase64String(\_passwordBase64));

}

set

{

\_passwordBase64 = Convert.ToBase64String(System.Text.Encoding.UTF8.GetBytes(value));

}

}

/// <summary>

/// Электронная почта

/// </summary>

[DataMember]

public string Email { get; set; }

/// <summary>

/// Адрес веб-сайта

/// </summary>

[DataMember]

public string Website { get; set; }

/// <summary>

/// Комментарии к записи

/// </summary>

[DataMember]

public string Notes { get; set; }

/// <summary>

/// Инициализирует пустую запись с заданным именем

/// </summary>

/// <param name="entryName">имя записи</param>

public Entry(string entryName)

{

Name = entryName;

Username = "";

\_passwordBase64 = "";

Email = "";

Website = "";

Notes = "";

}

/// <summary>

/// Сравнивает данный экземпляр с объектом Entry и показывает, расположен ли данный экземпляр перед, до или на той же позициив порядке сортировки, что и заданный объект Entry

/// </summary>

/// <param name="other">объект, имеющий значение Entry</param>

/// <returns>32-битовое целое число со знаком, которое показывает, расположен ли данный экземпляр перед, после или на той же позиции в порядке сортировки, что и параметр</returns>

int IComparable<Entry>.CompareTo(Entry other)

{

return Name.CompareTo(other.Name);

}

}

}

**Текст файла «Root.cs»**

using System;

using System.Collections;

using System.Collections.Generic;

using System.Runtime.Serialization;

namespace SavePass.Nodes

{

/// <summary>

/// Коллекция класса Entry

/// </summary>

[DataContract]

public class Root : IComparable<Root>, IEnumerable<Entry>

{

/// <summary>

/// Список полей

/// </summary>

[DataMember(Name = "Entries")]

private List<Entry> \_entries;

/// <summary>

/// Имя базы данных

/// </summary>

[DataMember]

public string Name { get; set; }

/// <summary>

/// Индексатор класса Entry

/// </summary>

/// <param name="key">Имя записи</param>

/// <returns></returns>

public Entry this[string key]

{

get

{

foreach (var item in \_entries)

{

if (key == item.Name)

{

return item;

}

}

return null;

}

}

/// <summary>

/// Инициализирует класс Root с заданным именем

/// </summary>

/// <param name="name">имя базы данных</param>

public Root(string name)

{

Name = name;

\_entries = new List<Entry>();

}

/// <summary>

/// Инициализирует класс Root с заданным именем и заданной коллекцией класса Entry

/// </summary>

/// <param name="name">имя базы данных</param>

/// <param name="collection">коллекция класса Entry</param>

public Root(string name, IEnumerable<Entry> collection) : this(name)

{

\_entries.AddRange(collection);

}

/// <summary>

/// Добавляет запись в базу данных

/// </summary>

/// <param name="item">класс Entry</param>

public void Add(Entry item)

{

string name = item.Name;

for (int i = 2; this[item.Name] != null; i++)

{

item.Name = string.Format("{0} ({1})", name, i);

}

\_entries.Add(item);

\_entries.Sort();

}

/// <summary>

/// Удаляет запись с базы данных

/// </summary>

/// <param name="item">класс Entry</param>

public bool Remove(Entry item)

{

return \_entries.Remove(item);

}

/// <summary>

/// Сравнивает данный экземпляр с объектом Root и показывает, расположен ли данный экземпляр перед, до или на той же позициив порядке сортировки, что и заданный объект Root

/// </summary>

/// <param name="other">объект, имеющий значение Root</param>

/// <returns>32-битовое целое число со знаком, которое показывает, расположен ли данный экземпляр перед, после или на той же позиции в порядке сортировки, что и параметр</returns>

public int CompareTo(Root other)

{

return Name.CompareTo(other.Name);

}

/// <summary>

/// Возвращает перечислитель, осуществляющий перебор элементов списка \_entries

/// </summary>

public IEnumerator<Entry> GetEnumerator()

{

return \_entries.GetEnumerator();

}

/// <summary>

/// Возвращает перечислитель, осуществляющий перебор элементов списка \_entries

/// </summary>

IEnumerator IEnumerable.GetEnumerator()

{

return this.GetEnumerator();

}

}

}

**Текст файла «Header.cs»**

using System;

using SavePass.Exceptions;

namespace SavePass.Nodes

{

/// <summary>

/// Заголовок файла

/// </summary>

public class Header

{

private static Random \_rand = new Random();

/// <summary>

/// Сигнатура программы

/// </summary>

public const uint BaseSignature = 0x1C8D7D4A;

/// <summary>

/// Сигнатура версии

/// </summary>

public const uint VersionSignature = 0x00010000;

/// <summary>

/// Соль

/// </summary>

public readonly byte[] MasterSeed;

/// <summary>

/// Инициализирует класс Header

/// </summary>

/// <param name="baseSignature">cигнатура программы</param>

/// <param name="versionSignature">cигнатура версии</param>

/// <param name="masterSeed">соль</param>

public Header(uint baseSignature, uint versionSignature, byte[] masterSeed)

{

if (baseSignature != BaseSignature || versionSignature != VersionSignature)

{

throw new FileCorruptException();

}

MasterSeed = masterSeed;

}

/// <summary>

/// Инициализирует класс Header

/// </summary>

public Header()

{

MasterSeed = new byte[8];

\_rand.NextBytes(MasterSeed);

}

}

}

**Текст файла «SavePassFile.cs»**

using System;

using System.IO;

using System.Runtime.Serialization.Json;

using System.Text;

using SavePass.Crypto;

using SavePass.Exceptions;

namespace SavePass.Nodes

{

public class SavePassFile

{

/// <summary>

/// JSON-сериализатор

/// </summary>

private DataContractJsonSerializer \_jsonFormatter = new DataContractJsonSerializer(typeof(Root));

/// <summary>

/// Заголовок файла

/// </summary>

private Header \_header;

/// <summary>

/// Имя файла

/// </summary>

public string FileName { get; private set; }

/// <summary>

/// Пароль базы данных

/// </summary>

public string Password { get; set; }

/// <summary>

/// База данных

/// </summary>

public Root Root { get; private set; }

/// <summary>

/// Инициализирует экземпляр класса SavePassFile

/// </summary>

public SavePassFile()

{

}

/// <summary>

/// Создаёт базы данных с заданным именем и паролем

/// </summary>

/// <param name="databaseName">имя базы данных</param>

/// <param name="password">пароль базы данных</param>

public void New(string databaseName, string password)

{

Close();

Password = password;

\_header = new Header();

Root = new Root(databaseName);

}

/// <summary>

/// Открывает существующий файл SavePass

/// </summary>

/// <param name="filename">имя файла</param>

/// <param name="password">пароль</param>

public void Open(string filename, string password)

{

FileName = filename;

Password = password;

using (Stream stream = new FileStream(filename, FileMode.Open, FileAccess.ReadWrite))

{

using (BinaryReader binrd = new BinaryReader(stream))

{

uint baseSignature = binrd.ReadUInt32();

uint versionSignature = binrd.ReadUInt32();

byte[] masterSeed = binrd.ReadBytes(8);

\_header = new Header(baseSignature, versionSignature, masterSeed);

byte[] jsonEncrypted = binrd.ReadBytes((int)(stream.Length - stream.Position));

byte[] jsonDecrypted;

try

{

jsonDecrypted = AES.AESDecrypt(jsonEncrypted, Encoding.Unicode.GetBytes(Password), masterSeed);

}

catch (System.Security.Cryptography.CryptographicException)

{

throw new WrongPasswordException();

}

using (MemoryStream ms = new MemoryStream(jsonDecrypted))

{

try

{

Root = (Root)\_jsonFormatter.ReadObject(ms);

}

catch (Exception)

{

throw new FileCorruptException();

}

}

}

}

}

/// <summary>

/// Сохраняет открытый файл

/// </summary>

public void Save()

{

using (Stream stream = new FileStream(FileName, FileMode.Create, FileAccess.Write))

{

using (BinaryWriter binwr = new BinaryWriter(stream))

{

binwr.Write(Header.BaseSignature);

binwr.Write(Header.VersionSignature);

binwr.Write(\_header.MasterSeed);

byte[] jsonDecrypted, jsonEncrypted;

using (MemoryStream ms = new MemoryStream())

{

\_jsonFormatter.WriteObject(ms, Root);

ms.Position = 0;

jsonDecrypted = new byte[ms.Length];

ms.Read(jsonDecrypted, 0, (int)ms.Length);

}

jsonEncrypted = AES.AESEncrypt(jsonDecrypted, Encoding.Unicode.GetBytes(Password), \_header.MasterSeed);

binwr.Write(AES.AESEncrypt(jsonDecrypted, Encoding.Unicode.GetBytes(Password), \_header.MasterSeed));

}

}

}

/// <summary>

/// Сохраняет открытый файл с указанных путём файла

/// </summary>

public void Save(string filename)

{

FileName = filename;

Save();

}

/// <summary>

/// Закрывает открытый файл

/// </summary>

public void Close()

{

FileName = null;

Root = null;

}

}

}

**Текст файла «FormMain.cs»**

using System;

using System.Diagnostics;

using System.Windows.Forms;

using System.Text.RegularExpressions;

using SavePass.Nodes;

using SavePass.Exceptions;

using System.IO;

namespace SavePass.Forms

{

enum EntryState { None, Add, Edit }

public partial class FormMain : Form

{

private bool \_isFileSaved = true;

private EntryState \_entryState = EntryState.None;

private SavePassFile \_savepass = new SavePassFile();

private Entry \_selectedEntry;

public FormMain()

{

InitializeComponent();

}

public FormMain(string filename) : this()

{

FileOpen\_Helper(filename);

}

#region Helpers

private void dgEntries\_Refresh()

{

dgEntries.Rows.Clear();

foreach (var item in \_savepass.Root)

{

int index = dgEntries.Rows.Add(item.Name);

if (\_selectedEntry != null && item.Name == \_selectedEntry.Name) dgEntries.Rows[index].Selected = true;

}

}

private Entry GetSelectedEntry()

{

return \_savepass.Root[dgEntries.SelectedCells[0].Value.ToString()];

}

private void EntryRefresh()

{

Entry tmp = GetSelectedEntry();

if (tmp == null) return;

btnEntryName.Text = tbxEntryName.Text = tmp.Name;

btnUsername.Text = tbxUsername.Text = tmp.Username;

btnPassword.Text = Regex.Replace(tmp.Password, ".", "●");

tbxPassword.Text = tmp.Password;

btnEmail.Text = tbxEmail.Text = tmp.Email;

btnWebsite.Text = tbxWebsite.Text = tmp.Website;

btnNotes.Text = tbxNotes.Text = tmp.Notes;

}

private void ControlsEnable(bool enable)

{

panEntriesList.Enabled = enable;

miFileClose.Enabled = enable;

miFileSave.Enabled = enable;

miFileSaveAs.Enabled = enable;

miDatabaseSettings.Enabled = enable;

miEntryAdd.Enabled = enable;

tsbtnFileSave.Enabled = enable;

tsbtnEntryAdd.Enabled = enable;

tsbtnCopyUsername.Enabled = enable;

tsbtnCopyPassword.Enabled = enable;

tsbtnOpenURL.Enabled = enable;

//tsbtnEntryEdit.Enabled = enable; // fix

//tsbtnEntryDelete.Enabled = enable;

//miEntryEdit.Enabled = enable;

//miEntryDelete.Enabled = enable;

}

#endregion Helpers

#region Events

private void dgEntries\_SelectionChanged(object sender, EventArgs e)

{

if (dgEntries.SelectedCells.Count == 0 || dgEntries.SelectedCells[0].Value == null)

{

panEntryInfo.Enabled = false;

tsbtnEntryEdit.Enabled = false;

tsbtnEntryDelete.Enabled = false;

miEntryEdit.Enabled = false;

miEntryDelete.Enabled = false;

btnEntryName.Text = "";

btnUsername.Text = "";

btnPassword.Text = "";

btnEmail.Text = "";

btnWebsite.Text = "";

btnNotes.Text = "";

}

else

{

panEntryInfo.Enabled = true;

tsbtnEntryEdit.Enabled = true;

tsbtnEntryDelete.Enabled = true;

miEntryEdit.Enabled = true;

miEntryDelete.Enabled = true;

EntryRefresh();

}

}

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

{

\_selectedEntry = new Entry("New entry");

\_savepass.Root.Add(\_selectedEntry);

dgEntries\_Refresh();

EntryEdit\_Click(sender, e);

\_entryState = EntryState.Add;

}

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

{

\_selectedEntry = GetSelectedEntry();

EntryRefresh();

panEntryEdit.Visible = true;

panEntryInfo.Visible = false;

panEntriesList.Enabled = false;

menuStrip1.Enabled = false;

toolStrip1.Enabled = false;

\_entryState = EntryState.Edit;

tbxEntryName.Focus();

}

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

{

if (\_savepass.Root[tbxEntryName.Text] != null && \_selectedEntry.Name != tbxEntryName.Text)

{

MessageBox.Show("Entry with this name is already exists", Application.ProductName);

return;

}

\_selectedEntry.Name = tbxEntryName.Text;

\_selectedEntry.Username = tbxUsername.Text;

\_selectedEntry.Password = tbxPassword.Text;

\_selectedEntry.Email = tbxEmail.Text;

\_selectedEntry.Website = tbxWebsite.Text;

\_selectedEntry.Notes = tbxNotes.Text;

EntryRefresh();

panEntryInfo.Visible = true;

panEntryEdit.Visible = false;

panEntriesList.Enabled = true;

menuStrip1.Enabled = true;

toolStrip1.Enabled = true;

\_entryState = EntryState.None;

\_isFileSaved = false;

this.Text = string.Format("{0} \* Password Manager", \_savepass.Root.Name);

dgEntries\_Refresh();

}

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

{

if (\_entryState == EntryState.Add)

{

\_savepass.Root.Remove(\_selectedEntry);

}

panEntryInfo.Visible = true;

panEntryEdit.Visible = false;

panEntriesList.Enabled = true;

menuStrip1.Enabled = true;

toolStrip1.Enabled = true;

\_entryState = EntryState.None;

dgEntries\_Refresh();

}

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

{

if (MessageBox.Show("Do you want to delete entry?", Application.ProductName, MessageBoxButtons.YesNo, MessageBoxIcon.Question) != DialogResult.Yes) return;

\_savepass.Root.Remove(GetSelectedEntry());

dgEntries\_Refresh();

\_isFileSaved = false;

this.Text = string.Format("{0} \* Password Manager", \_savepass.Root.Name);

}

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

{

try

{

string website = GetSelectedEntry().Website;

if (string.IsNullOrEmpty(website)) return;

if (!website.StartsWith(@"http://") && !website.StartsWith(@"https://"))

{

website = "http://" + website;

}

Process.Start(website);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

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

{

Button control = (Button)(((ContextMenuStrip)(((ToolStripMenuItem)sender).Owner)).SourceControl);

Entry tmp = GetSelectedEntry();

string strToCopied = string.Empty;

if (control == btnEntryName)

{

strToCopied = tmp.Name;

}

else if (control == btnUsername)

{

strToCopied = tmp.Username;

}

else if (control == btnPassword)

{

strToCopied = tmp.Password;

}

else if (control == btnEmail)

{

strToCopied = tmp.Email;

}

else if (control == btnWebsite)

{

strToCopied = tmp.Website;

}

else if (control == btnNotes)

{

strToCopied = tmp.Notes;

}

if (string.IsNullOrEmpty(strToCopied)) return;

Clipboard.SetText(strToCopied);

}

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

{

Entry tmp = GetSelectedEntry();

if (string.IsNullOrEmpty(tmp.Username)) return;

Clipboard.SetText(tmp.Username);

}

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

{

Entry tmp = GetSelectedEntry();

if (string.IsNullOrEmpty(tmp.Password)) return;

Clipboard.SetText(tmp.Password);

}

private void dgEntries\_CellMouseClick(object sender, DataGridViewCellMouseEventArgs e)

{

if (e.Button == MouseButtons.Right)

{

dgEntries[e.ColumnIndex, e.RowIndex].Selected = true;

cmiEntryEdit.Enabled = true;

cmiEntryDelete.Enabled = true;

cmEntryList.Show(MousePosition);

}

}

private void dgEntries\_MouseClick(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Right)

{

cmiEntryEdit.Enabled = false;

cmiEntryDelete.Enabled = false;

cmEntryList.Show(MousePosition);

}

}

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

{

//if (saveFileDialog1.ShowDialog() != DialogResult.OK) return;

FormDatabaseSettings dbsett = new FormDatabaseSettings();

if (dbsett.ShowDialog() != DialogResult.OK) return;

\_savepass.New(dbsett.DatabaseName, dbsett.Password);

\_isFileSaved = false;

this.Text = string.Format("{0} \* Password Manager", \_savepass.Root.Name);

ControlsEnable(true);

miEntryEdit.Enabled = false;

miEntryDelete.Enabled = false;

tsbtnEntryEdit.Enabled = false;

tsbtnEntryDelete.Enabled = false;

dgEntries\_Refresh();

}

private void FileOpen\_Helper(string filename)

{

PasswordDialog passwdDialog = new PasswordDialog();

if (passwdDialog.ShowDialog() != DialogResult.OK) return;

try

{

\_savepass.Open(filename, passwdDialog.Password);

\_isFileSaved = true;

this.Text = string.Format("{0} - Password Manager", \_savepass.Root.Name);

ControlsEnable(true);

dgEntries\_Refresh();

}

catch (FileCorruptException)

{

MessageBox.Show("File is corrupt", Application.ProductName, MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (WrongPasswordException)

{

MessageBox.Show("Password is wrong", Application.ProductName, MessageBoxButtons.OK, MessageBoxIcon.Information);

FileOpen\_Helper(filename);

}

}

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

{

openFileDialog1.FileName = "";

if (openFileDialog1.ShowDialog() != DialogResult.OK) return;

FileOpen\_Helper(openFileDialog1.FileName);

}

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

{

if (string.IsNullOrEmpty(\_savepass.FileName))

{

saveFileDialog1.FileName = \_savepass.Root.Name;

FileSaveAs\_Click(sender, e);

}

else

{

\_savepass.Save();

\_isFileSaved = true;

this.Text = string.Format("{0} - Password Manager", \_savepass.Root.Name);

}

}

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

{

if (!string.IsNullOrEmpty(\_savepass.FileName)) saveFileDialog1.FileName = new FileInfo(\_savepass.FileName).Name;

if (saveFileDialog1.ShowDialog() != DialogResult.OK) return;

\_savepass.Save(saveFileDialog1.FileName);

\_isFileSaved = true;

this.Text = string.Format("{0} - Password Manager", \_savepass.Root.Name);

}

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

{

if (!\_isFileSaved)

{

var question = MessageBox.Show("Do you want to save changes?", Application.ProductName, MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question);

if (question == DialogResult.Cancel) return;

if (question == DialogResult.Yes)

{

if (string.IsNullOrEmpty(\_savepass.FileName))

{

if (saveFileDialog1.ShowDialog() != DialogResult.OK) return;

\_savepass.Save(saveFileDialog1.FileName);

}

else

{

\_savepass.Save();

}

}

}

\_isFileSaved = true;

this.Text = string.Format("Password Manager");

ControlsEnable(false);

dgEntries.Rows.Clear();

btnEntryName.Text = "Title";

btnUsername.Text = "";

btnPassword.Text = "";

btnEmail.Text = "";

btnWebsite.Text = "";

btnNotes.Text = "";

\_savepass.Close();

}

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

{

FormDatabaseSettings dbsetts = new FormDatabaseSettings(\_savepass.Root.Name, \_savepass.Password);

if (dbsetts.ShowDialog() != DialogResult.OK) return;

\_savepass.Root.Name = dbsetts.DatabaseName;

\_savepass.Password = dbsetts.Password;

\_isFileSaved = false;

this.Text = string.Format("{0} \* Password Manager", \_savepass.Root.Name);

}

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

{

Close();

}

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

{

new FormAbout().ShowDialog();

}

private void FormMain\_DragEnter(object sender, DragEventArgs e)

{

if (e.Data.GetDataPresent(DataFormats.FileDrop))

{

e.Effect = DragDropEffects.Copy;

}

}

private void FormMain\_DragDrop(object sender, DragEventArgs e)

{

string[] dragDropFiles = (string[])e.Data.GetData(DataFormats.FileDrop);

if (dragDropFiles.Length != 1) return;

FileOpen\_Helper(dragDropFiles[0]);

}

private void FormMain\_FormClosing(object sender, FormClosingEventArgs e)

{

if (!\_isFileSaved)

{

var question = MessageBox.Show("Do you want to save changes?", Application.ProductName, MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question);

if (question == DialogResult.Cancel)

{

e.Cancel = true;

return;

}

if (question == DialogResult.Yes)

{

if (string.IsNullOrEmpty(\_savepass.FileName))

{

if (saveFileDialog1.ShowDialog() != DialogResult.OK)

{

e.Cancel = true;

return;

}

\_savepass.Save(saveFileDialog1.FileName);

}

else

{

\_savepass.Save();

}

}

}

}

#endregion Events

}

}

**Текст файла «FormDatabaseSettings.cs»**

using System;

using System.Windows.Forms;

namespace SavePass.Forms

{

public partial class FormDatabaseSettings : Form

{

public string DatabaseName { get; set; }

public string Password { get; set; }

public FormDatabaseSettings()

{

InitializeComponent();

}

public FormDatabaseSettings(string databaseName, string password) : this()

{

txtDatabaseName.Text = databaseName;

txtPassword.Text = txtPasswordRepeat.Text = password;

}

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

{

if (txtPassword.UseSystemPasswordChar)

{

txtPassword.UseSystemPasswordChar = false;

txtPasswordRepeat.Enabled = false;

}

else

{

txtPassword.UseSystemPasswordChar = true;

txtPasswordRepeat.Enabled = true;

}

}

private void FormDatabaseSettings\_FormClosing(object sender, FormClosingEventArgs e)

{

if (DialogResult == DialogResult.OK)

{

if (txtDatabaseName.Text == "")

{

MessageBox.Show("Database name field cannot be blank", Application.ProductName);

e.Cancel = true;

return;

}

if (txtPassword.UseSystemPasswordChar && txtPassword.Text != txtPasswordRepeat.Text)

{

MessageBox.Show("Passwords do not match", Application.ProductName);

e.Cancel = true;

return;

}

if (txtPassword.Text == "")

{

if (MessageBox.Show("You have selected to use an empty password\n\nAre you sure you want to use an empty master password?",

Application.ProductName,

MessageBoxButtons.YesNo,

MessageBoxIcon.Question,

MessageBoxDefaultButton.Button2) == DialogResult.No)

{

e.Cancel = true;

return;

}

}

DatabaseName = txtDatabaseName.Text;

Password = txtPassword.Text;

}

}

}

}

**Текст файла «PasswordDialog.cs»**

using System;

using System.Windows.Forms;

namespace SavePass.Forms

{

public partial class PasswordDialog : Form

{

public string Password { get; set; }

public PasswordDialog()

{

InitializeComponent();

}

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

{

txtPassword.UseSystemPasswordChar = !txtPassword.UseSystemPasswordChar;

}

private void PasswordDialog\_FormClosing(object sender, FormClosingEventArgs e)

{

if (DialogResult == DialogResult.OK)

{

Password = txtPassword.Text;

}

}

}

}

**Текст файла «FormAbout.cs»**

using System;

using System.Reflection;

using System.Windows.Forms;

namespace SavePass.Forms

{

partial class FormAbout : Form

{

public FormAbout()

{

InitializeComponent();

this.Text = String.Format("About {0}", AssemblyTitle);

this.labelProductName.Text = AssemblyProduct;

this.labelVersion.Text = String.Format("Version {0}", AssemblyVersion);

this.labelCopyright.Text = AssemblyCopyright;

this.labelCompanyName.Text = AssemblyCompany;

}

#region Методы доступа к атрибутам сборки

public string AssemblyTitle

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyTitleAttribute), false);

if (attributes.Length > 0)

{

AssemblyTitleAttribute titleAttribute = (AssemblyTitleAttribute)attributes[0];

if (titleAttribute.Title != "")

{

return titleAttribute.Title;

}

}

return System.IO.Path.GetFileNameWithoutExtension(Assembly.GetExecutingAssembly().CodeBase);

}

}

public string AssemblyVersion

{

get

{

return Assembly.GetExecutingAssembly().GetName().Version.ToString();

}

}

public string AssemblyDescription

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyDescriptionAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyDescriptionAttribute)attributes[0]).Description;

}

}

public string AssemblyProduct

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyProductAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyProductAttribute)attributes[0]).Product;

}

}

public string AssemblyCopyright

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyCopyrightAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyCopyrightAttribute)attributes[0]).Copyright;

}

}

public string AssemblyCompany

{

get

{

object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyCompanyAttribute), false);

if (attributes.Length == 0)

{

return "";

}

return ((AssemblyCompanyAttribute)attributes[0]).Company;

}

}

#endregion

}

}

Приложение В. Руководство оператора

**Назначение программы**

Назначение программы «Менеджер паролей» состоит в сохранении всех Ваших паролей в одну базу данных, которая надежно закрыта единственным мастер-паролем. Поэтому, необходимо запомнить, только один мастер-пароль, чтобы управлять всей базой данных. База данных, находится в зашифрованном виде с применением одного из самых лучших и безопасных, из известных в настоящее время, алгоритмов шифрования, AES.

**Системные требования**

В состав технических средств должен входить персональный компьютер (ПЭВМ), включающий в себя:

1. процессор с тактовой частотой, не менее 1.5 ГГц;
2. материнскую плату с FSB, не менее 133 МГц;
3. оперативную память объемом не менее 512 Мб;
4. свободного дискового пространства для хранения программы, не менее 20 Мб;
5. дисплей и видеоадаптер такого типа, которые смогут отображать графическую информацию, являющуюся результатом выполнения программы;
6. клавиатуру;
7. мышь;

**Эксплуатация программы**

Запустите приложение «SavePass.exe».

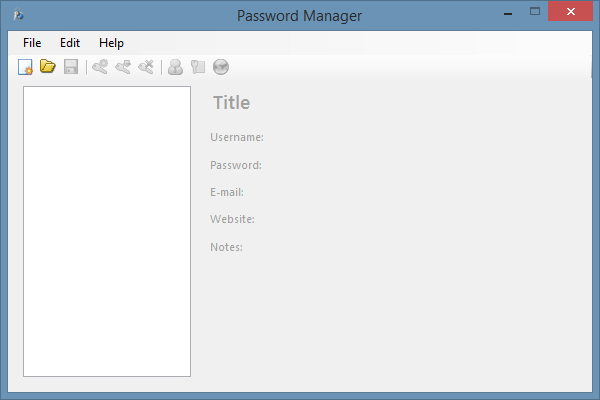


Рисунок В.1 – Внешний вид главного окна

Чтобы создать базу данных, воспользуйтесь одним из способов:

* нажатие соответствующего пункта меню;
* нажатие соответствующего пункта на ToolBar;
* с помощью горячей клавиши (Ctrl+N).

Чтобы открыть базу данных, воспользуйтесь одним из способов:

* нажатие соответствующего пункта меню;
* нажатие соответствующего пункта на ToolBar;
* с помощью горячей клавиши (Ctrl+O);
* с помощью Drag & Drop файла на главное окно программы.

Чтобы закрыть базу данных, воспользуйтесь одним из способов:

* нажатие соответствующего пункта меню;
* с помощью горячей клавиши (Ctrl+W).

Чтобы сохранить базу данных, воспользуйтесь одним из способов:

* нажатие соответствующего пункта меню;
* нажатие соответствующего пункта на ToolBar;
* с помощью горячей клавиши (Ctrl+S).

Чтобы сохранить базу данных по другому пути, воспользуйтесь одним из способов:

* нажатие соответствующего пункта меню;
* с помощью горячей клавиши (Ctrl+Shift+S).

Чтобы открыть окно настройки базы данных, воспользуйтесь одним из способов:

* нажатие соответствующего пункта меню;
* с помощью горячей клавиши (Ctrl+P).

Чтобы добавить запись в базу данных, воспользуйтесь одним из способов:

* нажатие соответствующего пункта меню;
* нажатие соответствующего пункта на ToolBar;
* с помощью контекстного меню на списке.

Чтобы изменить запись, воспользуйтесь одним из способов:

* нажатие соответствующего пункта меню;
* нажатие соответствующего пункта на ToolBar;
* с помощью контекстного меню на списке.

Чтобы удалить запись, воспользуйтесь одним из способов:

* нажатие соответствующего пункта меню;
* нажатие соответствующего пункта на ToolBar;
* с помощью контекстного меню на списке.

Чтобы вызвать окно «О программе», воспользуйтесь одним из способов:

* нажатие соответствующего пункта меню;
* с помощью горячей клавиши (F1).

При создании базы данных вы должны указать имя базы данных и пароль:

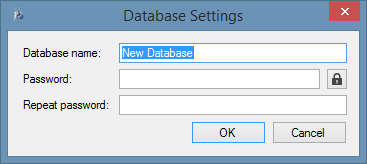


Рисунок В.2 – Окно настройки базы данных

Вы можете показать/скрыть пароль, нажав на кнопку замка.

Поле пароля можно оставить пустым, однако вы получите предупреждение о небезопасности данного действия:

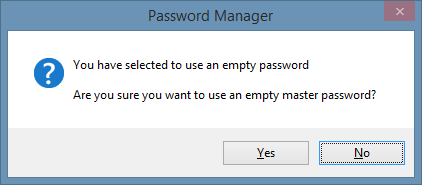


Рисунок В.3 – Предупреждение, если пустой пароль

При открытии файла вы получите диалоговое окно с запросом пароля:

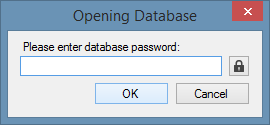


Рисунок В.3 – Диалоговое окно с запросом пароля

В режиме создания или редактирования записи главное окно имеет такой вид.

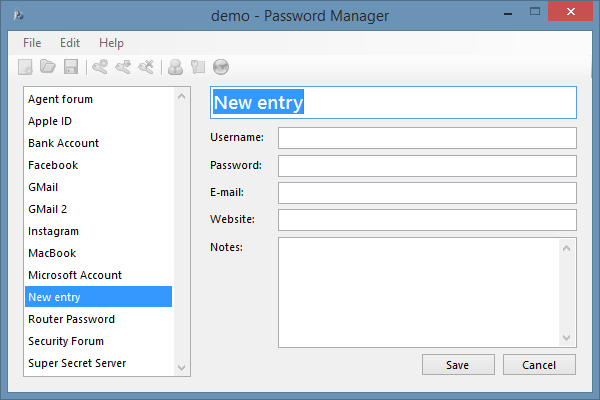


Рисунок В.4 – Режим редактирования записи

Нажмите «Save», чтобы сохранить результат.

Нажмите «Cancel», чтобы отменить создание/редактирование записи.

Приложение Г. Презентация

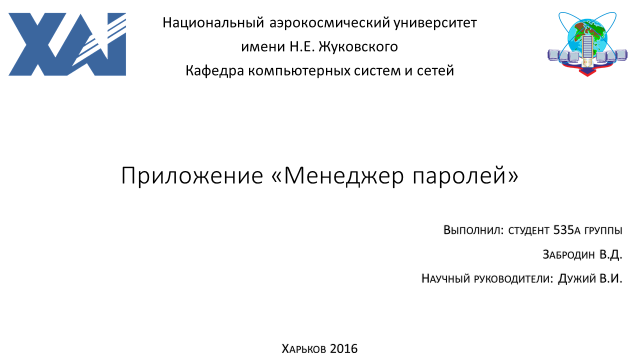


Рисунок Г.1 – Слайд 1

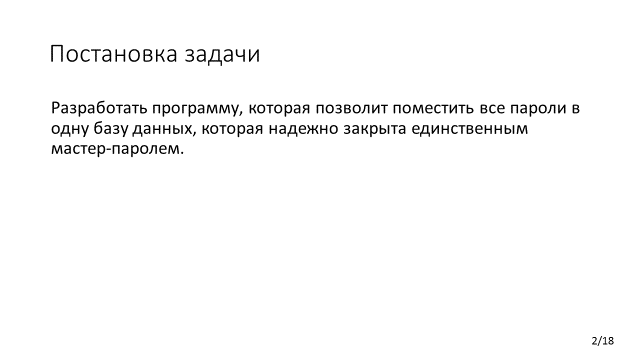


Рисунок Г.2 – Слайд 2

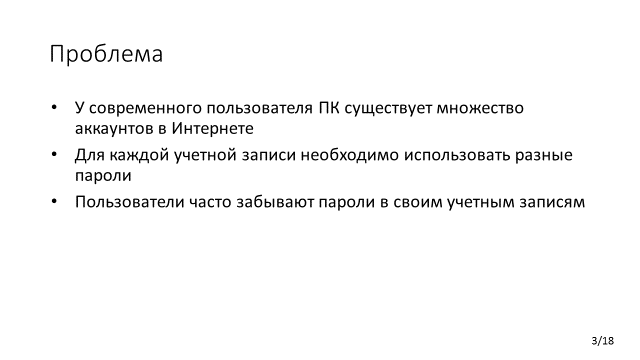


Рисунок Г.3 – Слайд 3

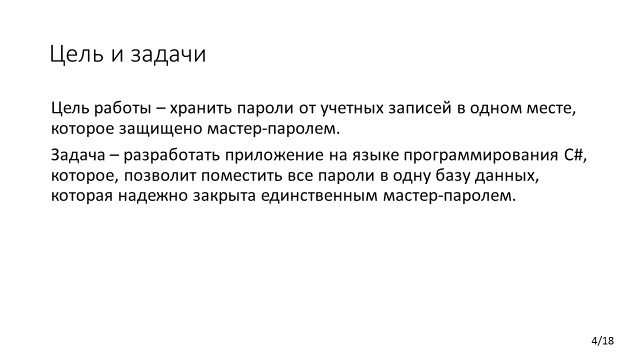


Рисунок Г.4 – Слайд 4

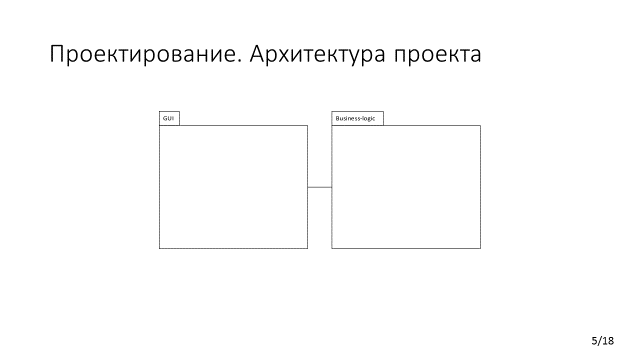


Рисунок Г.5 – Слайд 5



Рисунок Г.6 – Слайд 6

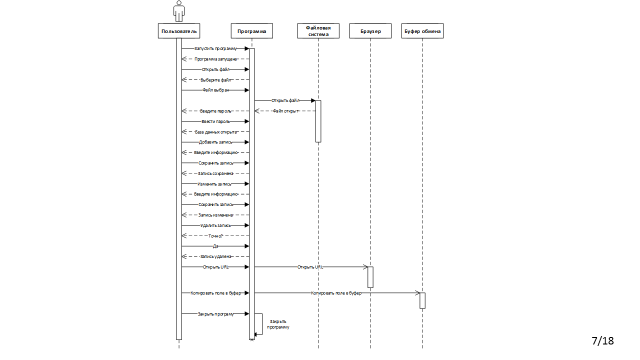


Рисунок Г.7 – Слайд 7

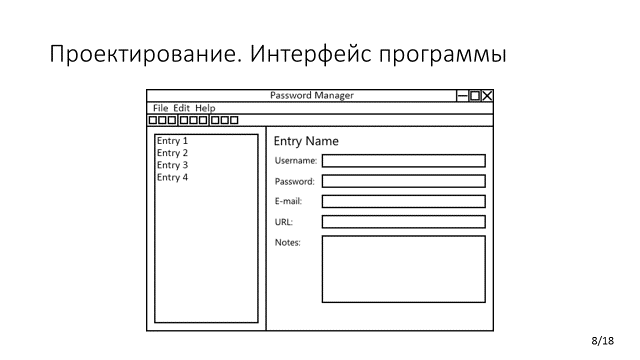


Рисунок Г.8 – Слайд 8

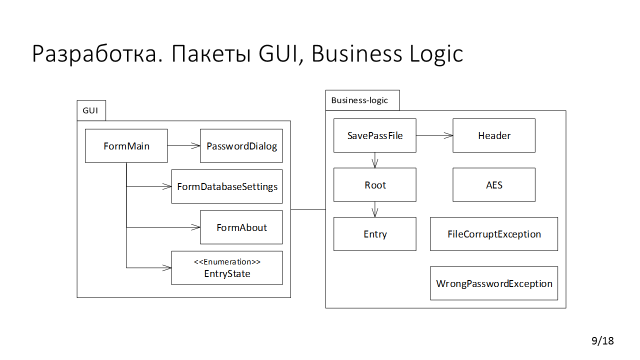


Рисунок Г.9 – Слайд 9

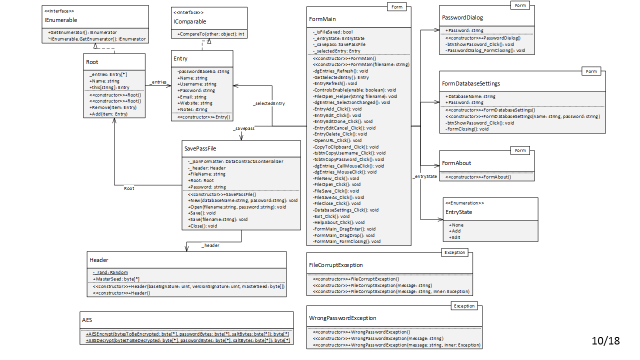


Рисунок Г.10 – Слайд 10

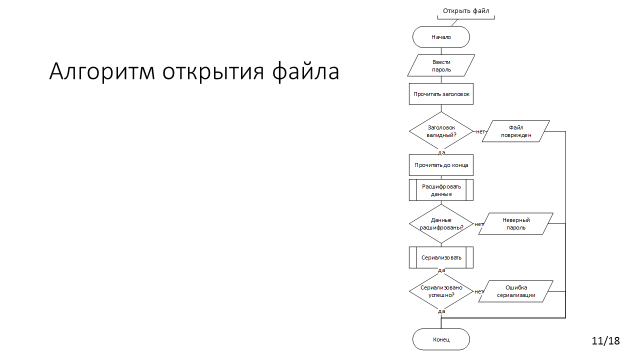


Рисунок Г.11 – Слайд 11

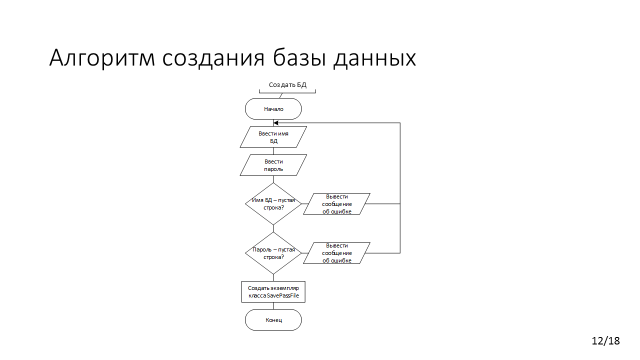


Рисунок Г.12 – Слайд 12



Рисунок Г.13 – Слайд 13



Рисунок Г.14 – Слайд 14



Рисунок Г.15 – Слайд 15

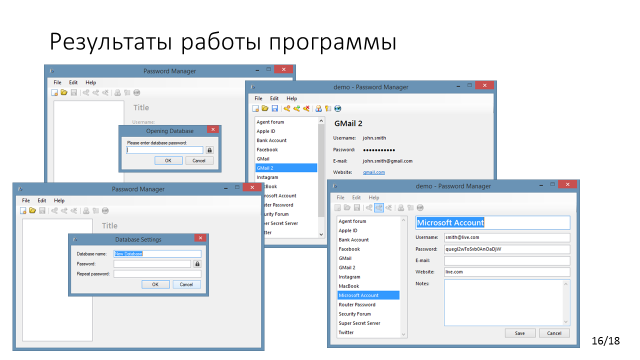


Рисунок Г.16 – Слайд 16

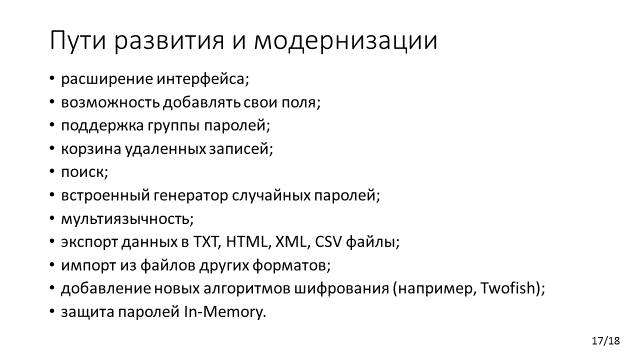


Рисунок Г.17 – Слайд 17

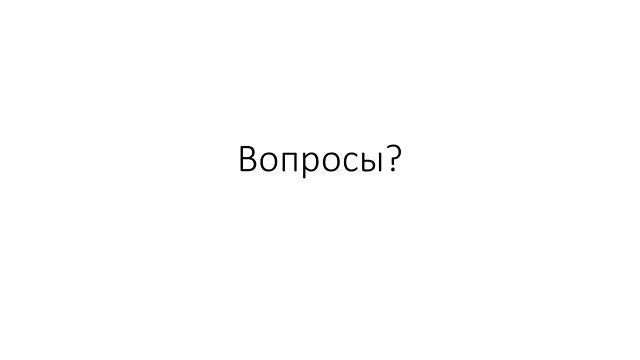


Рисунок Г.18 – Слайд 18