**Введение**

В рамках данного курсового проекта, поставлена задача разработать приложение для создания и редактирования семейных древ. Это приложение предназначено для всех, кто стремится раскрыть историю своего рода и узнать больше о своих предках.

Семейная история всегда волновала людей во все времена. Узнать о своих предках, исследовать корни своего рода и понять, откуда мы пришли, всегда вызывало интерес и желание узнать больше. В прошлом, люди создавали семейные деревья на бумаге, руководствуясь устными преданиями, старыми фотографиями и архивными документами. Они старательно восстанавливали паззл своей семейной истории и передавали эту информацию следующим поколениям.

С развитием технологий и появлением электронных средств хранения информации, возникла возможность создания и редактирования семейных древ в цифровом формате. Это дало людям новые возможности для более удобного и точного воссоздания своей генеалогии. Однако, найти подходящее приложение, которое бы сочетало в себе простоту использования, гибкость и функциональные возможности, оказалось сложной задачей.

Приложение обеспечивает удобный и интуитивно понятный интерфейс, позволяющий легко создавать, редактировать и визуализировать семейные древа. Пользователи могут добавлять информацию о своих родственниках, указывать их имена, даты рождения и смерти, фотографии и другие детали, которые помогут оживить их генеалогическую историю. Также пользователи могут добавлять информацию о памятных местах и событиях, объединивших семью, и, в конце концов, получить визуальное отображение генеалогического древа, применив предпочитаемые параметры для получения желаемого результата.

Далее приведем краткое описание разделов пояснительной записки.

Первый раздел «Анализ предметной области и формулировка требований к программе» посвящается исследованию предметной области, также в нем представлены инструменты разработки.

Во второй части «Проектирование» представлена диаграмма вариантов использования, диаграмма деятельности, диаграмма классов, а также описание тестов программного продукта.

Третий раздел «Построение программы» описывает выбор классов, определение атрибутов и методов класса.

Четвертый раздел «Тестирование». Он описывает полное и функциональное тестирование данного программного продукта. Смоделированы все действия пользователя при работе с программой.

Пятый раздел «Применение». Данный раздел содержит назначение и условия применения программы. Также раздел содержит выполнение программы, т.е. последовательность действий пользователя при работе с программой.

«Заключение» содержит краткую формулировку задачи, результаты проделанной работы, описание использованных методов и средств, описание степени автоматизации процессов на различных этапах разработки.

В «Списке использованных источников» приведен список используемых при разработке источников с информацией.

Приложение «А» представляет собой листинг программы с необходимыми комментариями.

Приложение «Б» содержит диаграмму деятельности.

Приложение «В» содержит диаграмму классов.

В графической части будет представлена диаграмма вариантов использования.

**1 Анализ предметной области и формулировка требований к программе**

**1.1 Постановка задачи**

На проектирование поставлена задача разработать программное обеспечение для создания и редактирования генеалогических деревьев.

Цель проекта – разработать программное обеспечение, обеспечивающее удобный и интуитивно понятный интерфейс, который позволит пользователям легко создавать, редактировать и визуализировать свои генеалогические древа. Программный продукт обеспечивает удобное и наглядное представление генеалогической информации, позволяя пользователям оживить свою генеалогическую историю и просматривать в электронном виде свои родственные связи.

В программе будут реализованы 4 основных пункта меню, при помощи которых пользователь будет перемещаться по нужным таблицам и изменять их, 6 основных функции со всеми таблицами. (добавление, удаление, просмотр, изменение, фильтрация), также пункт меню, позволяющий пользователю создавать отображение генеалогического древа, и пункт меню, позволяющий пользователю снова открывать и создавать базу древа, открывать примеры готовых баз данных, открыть справку.

Создаваемая программа будет рассчитана на пользователей, находящихся в поиске инструмента для исследования, организации и сохранения информации о своей семейной истории.

Пользователи смогут регулярно обновлять и редактировать информацию о членах семьи, добавлять фотографии и вести записи о событиях в семье, либо же использовать программу периодически в зависимости от доступности новой информации или особых событий в семье, таких как свадьбы, рождения или юбилеи.

За основу программы следует взять информацию из соответствующих источников (теоретические материалы по созданию настоящего генеалогического древа, существующие похожие программные средства, информацию о построении графов-деревьев) и максимально эффективно её объединить.

Исторически генеалогические деревья создавались на бумаге с использованием карандашей, ручек и линеек. Однако, процесс создания и обновления таких деревьев вручную может быть сложным, трудоемким и подверженным ошибкам. Создание генеалогических деревьев на бумаге имеет некоторые ограничения. Сложность масштабирования, ограниченное пространство для добавления новых членов семьи, трудности при внесении изменений и редактировании информации – все это может создавать проблемы и затруднения. Кроме того, бумажные генеалогические деревья могут быть подвержены физическому износу, повреждениям и утрате.

Однако, с развитием информационных технологий и программного обеспечения возникают новые возможности для создания генеалогических деревьев с использованием компьютерных приложений. Такие программы позволяют эффективно визуализировать и управлять генеалогическими данными, обеспечивая гибкость и удобство использования.

В контексте этой проблемы, разрабатываемое приложение становится важным решением.

Тем не менее, похожие программные продукты уже существуют. Их обзор поможет изучить конкурентные решения и определиться с особенностями и функциональностью будущего проекта. Вот несколько известных программ для генеалогического древа:

Ancestry: Ancestry является одной из самых популярных онлайн-платформ для создания генеалогических древ. Она предлагает богатый функционал, включая возможность создания и редактирования древа, добавление фотографий и документов, проведение исследований, поиск родственных связей и совместную работу с другими пользователями.

MyHeritage: MyHeritage также является популярной онлайн-платформой для генеалогического исследования. Она предлагает функции создания и редактирования генеалогического древа.

Gramps: Gramps – это бесплатное и открытое программное обеспечение для создания генеалогических древ. Оно предоставляет широкий набор функций, включая создание и редактирование древа, добавление фотографий и документов, отслеживание событий и связей между членами семьи;

Legacy Family Tree: Legacy Family Tree – это платная программа для создания генеалогических древ. Она предлагает широкие возможности для исследования, создания и редактирования древа, сохранения фотографий и документов.

Это лишь некоторые примеры существующих программ для генеалогического древа, и каждая из них имеет свои особенности и функциональность. Изучение этих программ поможет определиться с особенностями и требованиями проекта и создать уникальное и полезное решение, которое будет сочетать их основные функции, бесплатный доступ и удобный пользовательский интерфейс.

Разрабатываемый программный продукт предоставит возможность выполнить следующие действия:

* Создание базы данных генеалогического древа;
* Добавление, редактирование, удаление, фильтрация членов семьи;
* Добавление, редактирование, удаление мест;
* Добавление, редактирование, удаление событий;
* Создание, настройка и сохранение графического представления генеалогического древа;

В разрабатываемой программе будут использоваться следующие три вида данных:

1. Входные данные – это вводимые пользователем значения: пользователь будет вводить информацию о каждом члене семьи, новых местах и событиях, такую как имена, даты рождения, фотографии и другие атрибуты. Эти данные будут использоваться для создания и обновления генеалогического дерева;
2. Выходные данные – это результаты генеалогического дерева: после ввода данных пользователем, программа будет выводить генеалогическое дерево, которое визуально представит связи между членами семьи. Это дерево будет содержать информацию о каждом члене семьи, их родственные связи и другие дополнительные данные;
3. Условно-постоянная информация – это справочные материалы и примеры: пользователь будет иметь доступ к справочной информации, такой как руководства по пользованию программой, а также примеры готовых заполненных баз. Эти материалы помогут пользователям развивать свои знания и навыки в области генеалогического исследования.

**1.2 Инструменты разработки**

Для разработки данного проекта будет выбрана среда разработки Microsoft Visual Studio, так как Microsoft Visual Studio является наиболее распространенной интегрированной средой разработки (IDE) для разработки приложений на C#. Она обладает мощными инструментами разработки, включая визуальный дизайнер интерфейса, отладчик, автоматическую генерацию кода и многое другое.

Для создания графического интерфейса будет использован фреймворк Windows Presentation Foundation (WPF), его используют для создания пользовательских приложений в Windows с использованием C#. Он предоставляет множество возможностей для создания интерактивных и эстетически привлекательных пользовательских интерфейсов.

Для определения пользовательского интерфейса в WPF будет использоваться декларативный язык разметки XAML (Extensible Application Markup Language). Он позволяет разработчикам создавать графический интерфейс с помощью XML-подобного синтаксиса, что упрощает разработку и поддержку интерфейса. Также, для ускорения процесса разработки и увеличения его функциональности, будут использоваться библиотеки и компоненты: например, библиотеки для работы с базами данных, управления данными, создания графиков.

Для создания визуализации генеалогического древа будет использоваться библиотека GiGraph — для работы с графами в языке программирования C#. Она предоставляет набор инструментов и функций, которые позволяют создавать, визуализировать и манипулировать графами. GiGraph взаимодействует с Graphviz — программным инструментом для визуализации графов. Graphviz предоставляет набор инструментов и библиотек для создания визуализаций графов на основе их описания в формате DOT. GiGraph позволяет генерировать код DOT для графов и передавать его в Graphviz для создания визуализаций. Это обеспечивает мощные возможности визуализации и рендеринга графов.

В итоге, выбор Microsoft Visual Studio, WPF и GiGraph качестве инструментов разработки на C# позволит создать функциональное приложение для создания генеалогического древа с привлекательным и интуитивно понятным пользовательским интерфейсом.

При разработке данного программного продукта был использован компьютер со следующими характеристиками:

* Процессор: Intel Pentium Gold 7505 2.00 GHz;
* ОЗУ: 8Gb;
* Память: SSD 216Gb;
* Операционная система: Windows 10 Домашняя.

Также для разработки программы необходимы:

* Яндекс-браузер – нужен для нахождения информации и картинок;
* Word 2019 – нужен для написания пояснительной записки;
* СУБД MS Access 2019 – создание базы данных;
* Adobe Photoshop CS6 – нужен для создания и редактирования изображений;
* CorelDRAW 2020 – нужен для создания и редактирования векторных иллюстраций;
* Dr.Explain – нужен для создания справки приложения;
* Draw.io – нужен для создания диаграммы вариантов использования, диаграммы деятельности, диаграммы классов.

На этапе исследования предметной области был установленный целый ряд требований, которые предъявляют к разрабатываемой задаче. При проектировании форм приложения необходимо учесть удобство использования, интуитивность интерфейса и эстетическую привлекательность. Размещение элементов управления, их визуальное оформление и логическая организация должны быть продуманы таким образом, чтобы пользователь мог легко взаимодействовать с приложением и выполнять необходимые операции.

* Минимальные системные требования:
* Операционная система: Windows 7 или более поздняя версия;
* Процессор: Intel Core i3 или эквивалентный процессор;
* Оперативная память: минимум 4 ГБ RAM;
* Свободное место на жестком диске: минимум 1 ГБ;
* Разрешение экрана: минимум 1280x800 пикселей.

При разработке приложения будут использованы преимущественного зеленые, голубые и бирюзовые цвета и их оттенки для создания ассоциации с генеалогическими деревьями, выполненных на бумаге, которые часто делают в стиле настоящих. Основные разделы приложения должны быть доступны с первой страницы. Грамотный пользовательский интерфейс.

Следовательно, каждое окно должно иметь ясную визуальную иерархию своих элементов. Фрагменты текста должны располагаться на экране так, чтобы пользователя было просто и понятно принимать информацию.

Пользователь не должен испытывать какого-либо дискомфорта в плане восприятия информация, отображённой на экране. Объекты (рисунки и символы) не должны быть слишком мелкие.

Ошибки программы вследствие некорректного взаимодействия пользователя при работе с программой через графический интерфейс не должны влиять на работоспособность. Надёжность программы должна быть на высоком уровне, не допуская ситуаций программных ошибок и некорректной работы приложения.

**2 Проектирование**

**2.1 Диаграмма вариантов использования**

Диаграмма вариантов использования – диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью, так называемых вариантов использования.

Актером (actor) или действующим лицом называется любая сущность, взаимодействующая с системой извне (рисунок 1). Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик.

В разрабатываемом программном продукте роль актера выполняет пользователь.



Рисунок 1 – Графическое обозначение актера

Вариант использования является стандартным языком UML и применяется для спецификации общих особенностей системы и любой другой сущности. Отдельные варианты использования обозначаются на диаграмме эллипсом, в котором содержится его краткое название (рисунок 2).

****

Рисунок 2 – Графическое обозначение варианта использования

В данном проекте можно выделить следующие варианты использования:

* Открыть базу древа или создать новую;
* Открытие редактора древа и таблиц внутри него;
* Добавление, удаление, изменение, удаление, фильтрация таблиц;
* Настройка, просмотр и сохранение генеалогического древа;
* Открыть справку.

Отношение ассоциации является главным понятием языка UML и используется при построении всех графических моделей. Оно служит для обозначения роли актера в отдельном варианте использования. На диаграмме отношение ассоциации обозначается сплошной линией между актером и вариантом использования (рисунок 3).

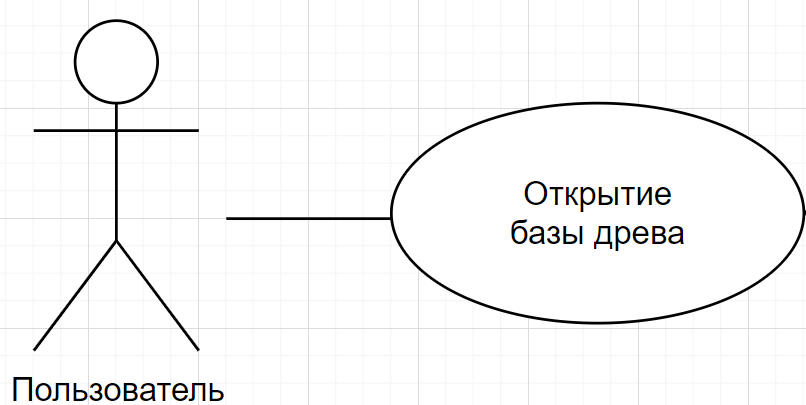


Рисунок 3 – Графическое обозначение отношения ассоциации

Для отображения взаимосвязи экземпляров отдельного варианта использования с более общим вариантом, используется отношение расширения, обозначаемое направленный пунктирной линией со стрелкой от исходного варианта. Данная линия помечается ключевым словом <<extend>> (рисунок 4).

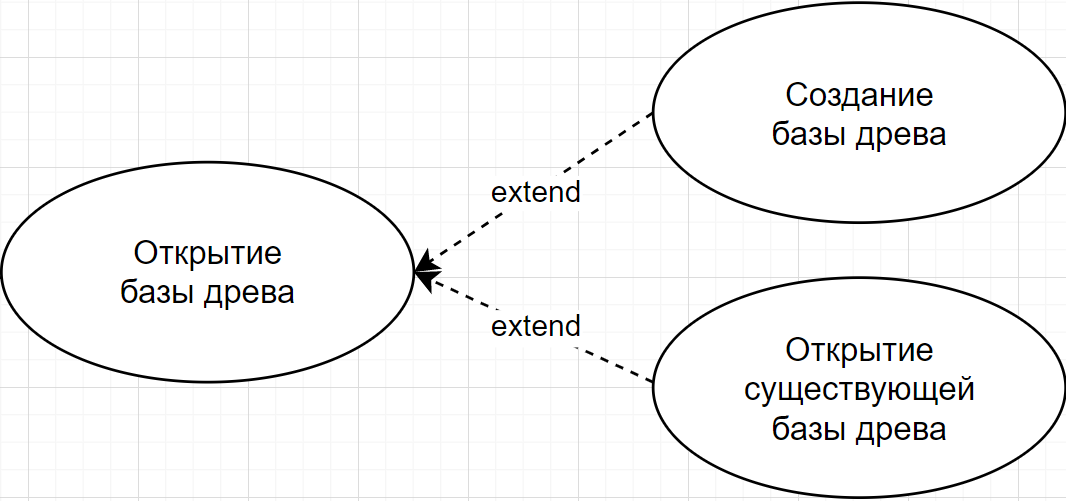


Рисунок 4 – Графическое обозначение отношения расширения

Отношение включения между двумя вариантами использования указывает, что некоторое заданное поведение для одного варианта использования включается в качестве составного компонента в последовательность поведения другого варианта использования. Данная линия помечается ключевым словом <<include>> (рисунок 5).

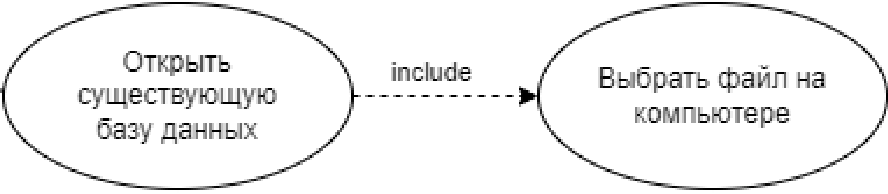


Рисунок 5 – Графическое обозначение отношения включения

Определяя для выбранного актера варианты использования и устанавливая отношения между вариантами использования, получим полную диаграмму вариантов, которая представлена в графической части.

**2.2 Диаграмма деятельности**

Диаграммы деятельности используется для моделирования вариантов использования, демонстрирует управляющие потоки от одних действий к другим.

Использование состояния деятельности и действия заключается в моделировании шага выполнения алгоритма или процедуры в рамках одного потока управления. Графически они изображаются прямоугольником со скругленными углами, а внутри записывается имя состояния деятельности (рисунок 6).

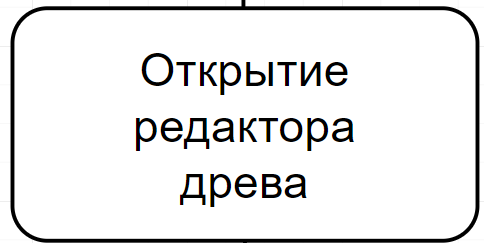


Рисунок 6 – Графическое обозначение состояния деятельности и действия

При построении диаграммы деятельности используют только не триггерные переходы, которые происходят сразу после завершения деятельности или выполнения соответствующего действия. На диаграмме такой переход изображается сплошной линией со стрелкой (рисунок 7). Если же таких переходов несколько, то для каждого из таких переходов должно быть явно записано собственное сторожевое условие в квадратных скобках (рисунок 8).

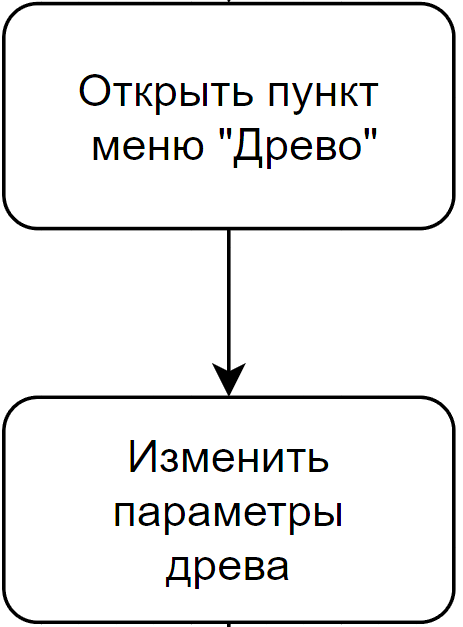


Рисунок 7 – Графическое обозначение переходов

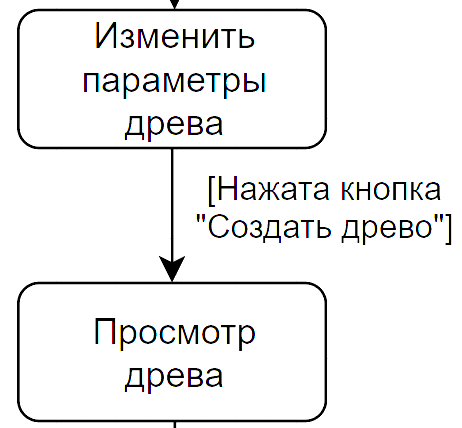


Рисунок 8 – Графическое обозначение сторожевого условия

Когда последовательно выполняемая деятельность должна разделиться на альтернативные ветви, используют ветвление. Графически ветвление обозначается символом решения, изображаемого в форме ромба, внутри которого нет текста (рисунок 9).

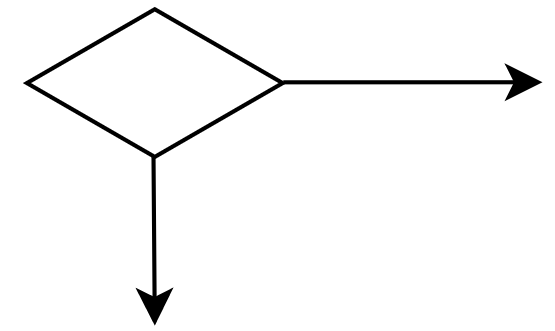


Рисунок 9 – Графическое обозначение решения

Для разделения и слияния параллельных вычислений или потоков управления используется прямая черточка (рисунок 10), которая шире линий простых переходов диаграммы деятельности.

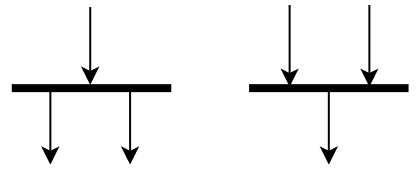


Рисунок 10 - Графическое обозначение разделения и слияния параллельных потоков управления

В итоге, получим полную диаграмму деятельности (Приложение Б).

**2.3 Описание тестов**

В ходе разработки программного продукта были составлены тесты, которые необходимо выполнить в дальнейшем. Они составлены таким образом, чтобы предусмотреть максимальное количество возможных действий. Тесты описаны в следующих таблице 1.

Таблица 1 – Тесты на использование

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Идентификатор | Приоритет | Модуль | Подмодуль | Заглавие (суть) теста-кейса | Ожидаемый результат |
| 1 | 2 | 3 | 4 | 5 | 6 |
| T1.1 | А | Приложение не запущено |  | Запуск приложения  1 Нажать на ярлык приложения | 1 Появится главная форма с кнопками «Создать древо» и «Открыть древо» |
| T1.2 | А | Приложение | Главная форма | Открытие базы данных  1 Нажатие на кнопку «Открыть древо»  2 Выбор файла на компьютере с базой семейного древа | 1 Открылся файловый диалог для открытия файлов, ожидающий выбора файла либо отмены действия  2 Открывается форма редактора древа с записями, ранее добавленными в него |
| T1.3 | А | Приложение | Главная форма | Открытие базы данных  1 Нажатие на кнопку «Создать древо»  2 Ввод названия и  выбор директории на компьютере | 1 Открылся файловый диалог для выбора папки, ожидающий выбор папки  2 Открывается пустая форма редактора древа (без записей) |
| T1.4 | А | Приложение | Главная форма | Открытие некорректной базы данных  1 Нажатие на кнопку «Открыть древо»  2 Выбор файла на компьютере с базой, созданной не в приложении | 1 Открылся файловый диалог для открытия файлов, ожидающий выбора файла либо отмены действия  2 Открывается форма с предупреждением о том, что база данных не является базой семейного древа |

Продолжение таблицы 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Идентификатор | Приоритет | Модуль | Подмодуль | Заглавие (суть) теста-кейса | Ожидаемый результат |
| 1 | 2 | 3 | 4 | 5 | 6 |
| T1.5 | А | Приложение | Главная форма | Открытие базы данных  1 Нажатие на кнопку «Открыть древо»  2 Ввод названия с неправильным расширением файла и  выбор директории файла на компьютере | 1 Открылся файловый диалог для создания файлов, ожидающий выбор места на диске и ввод названия файла  2 После нажатия на кнопку «ОК» программа сообщает, что расширение файла неверное. Файл не создается |
| T1.6 | А | Приложение | TreeEditor | Открытие формы редактора древа  1 Открытие или создание базы данных древа | 1 Текущей таблицей становится таблица с записями о персонах, содержащая информация о них |
| Т 1.7 | А | Приложение | TreeEditor | Просмотр таблицы «Персоны»  1 Нажатие на пункт меню «Персоны» | 1 Текущей таблицей становится обновленная таблица с записями о персонах, содержащая информация о них |
| T 1.8 | A | Приложение | TreeEditor | Просмотр таблицы «Места»  1 Нажатие пункт меню «Места» | 1 Текущей таблицей становится обновленная таблица с записями о местах, содержащая информация о них |
| T 1.9 | A | Приложение | TreeEditor | Просмотр таблицы «События»  1 Нажатие на кнопку «События» | 1 Текущей таблицей становится обновленная таблица с записями о событиях, содержащая информация о них |
| T 1.10 | A | Приложение | TreeEditor | Просмотр записи в таблице «Персоны»  1 Выделение одной из персон и нажатие на кнопку «Просмотр» | 1 Открывается форма с записью о выбранной персоне с информацией о родственных связях, персоне в целом и ссылки на места и упоминаемых людей. |

Продолжение таблицы 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Идентификатор | Приоритет | Модуль | Подмодуль | Заглавие (суть) теста-кейса | Ожидаемый результат |
| 1 | 2 | 3 | 4 | 5 | 6 |
| T1.11 | А | Приложение | TreeEditor | Просмотр записи в таблице «Места»  1 Выделение одного из мест нажатие на кнопку «Просмотр» | 1 Открывается форма с записью о выбранном месте с информацией о месте в целом и ссылкой на место в GoogleMaps. |
| T1.12 | А | Приложение | TreeEditor | Просмотр записи в таблице «События»  1 Выделение одного из событий и нажатие на кнопку «Просмотр» | 1 Открывается форма с записью о выбранном событии с информацией о событии в целом и ссылки на места и упоминаемых людей. |
| T 1.14 | A | Приложение | TreeEditor | Удаление записи в таблице «События»  1 Выделение одного из событий и нажатие на кнопку «Удаление»  2 Нажатие кнопки в появившейся форме «Да» | 1 Появляется форма, на которой нужно подтвердить удаление записи  2 Запись удаляется из таблицы, таблица обновляется |
| T 1.15 | A | Приложение | TreeEditor | Удаление записи в таблице «Персоны»  1 Выделение одного из событий и нажатие на кнопку «Удаление»  2 Нажатие кнопки в появившейся форме «Да» | 1 Появляется форма, на которой нужно подтвердить удаление записи  2 Запись удаляется из таблицы, таблица обновляется |

Продолжение таблицы 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Идентификатор | Приоритет | Модуль | Подмодуль | Заглавие (суть) теста-кейса | Ожидаемый результат |
| 1 | 2 | 3 | 4 | 5 | 6 |
| T1.16 | А | Приложение | TreeEditor | Удаление записи в таблице «Места»  1 Выделение одного из событий и нажатие на кнопку «Удаление»  2 Нажатие кнопки в появившейся форме «Да» | 1 Появляется форма, на которой нужно подтвердить удаление записи  2 Запись удаляется из таблицы, таблица обновляется |
| T1.17 | А | Приложение | TreeEditor | Просмотр, удаление, изменение при невыбранной записи  1 Нажатие на кнопку «Просмотр», «Удаление» или «Изменение» | 1 Открывается форма с предупреждением о том, что нужно выделить запись для просмотра, удаления или изменения |
| Т 1.18 | А | Приложение | TreeEditor | Просмотр, удаление, изменение при нескольких записях  1 Выделение нескольких записей в одной из таблиц и нажатие на кнопку «Просмотр», «Удаление» или «Изменение» | 1 Открывается форма с предупреждением о том, что нужно выделить лишь одну запись для просмотра, удаления или изменения |
| T 1.19 | A | Приложение | TreeEditor | Открытие формы для добавления персоны  1 Нажатие на «Добавление» в таблице «Персоны» | 1 Открывается форма с полями для добавления информации о персоне и фото |

Продолжение таблицы 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Идентификатор | Приоритет | Модуль | Подмодуль | Заглавие (суть) теста-кейса | Ожидаемый результат |
| 1 | 2 | 3 | 4 | 5 | 6 |
| T1.20 | А | Приложение | AddEdit-Person | Добавление новой персоны  1 Ввод по крайней мере в поле для имени и нажатие на «Сохранить» на форме добавления персоны | 1 Запись добавляется к таблице, таблица обновляется |
| T1.21 | А | Приложение | TreeEditor | Открытие формы для добавления места  1 Нажатие на «Добавление» в таблице «Места» | 1 Открывается форма с полями для добавления информации о месте и фото |
| Т 1.22 | А | Приложение | AddEdit-Place | Добавление нового места  1 Ввод по крайней мере в поле для названия и нажатие на «Сохранить» на форме добавления места | 1 Запись добавляется к таблице, таблица обновляется |
| T 1.23 | A | Приложение | TreeEditor | Открытие формы для добавления места  1 Нажатие на «Добавление» в таблице «События» | 1 Открывается форма с полями для добавления информации о событии и фото |
| T 1.24 | A | Приложение | AddEdit-Event | Добавление нового места  1 Ввод по крайней мере в поле для названия и нажатие на «Сохранить» на форме добавления места | 1 Запись добавляется к таблице, таблица обновляется |
| T 1.24 | A | Приложение | AddEdit-Person,  AddEdit-Place,  AddEdit-Event | Добавление или сохранение изменения полностью пустой записи  1 Нажатие на «Сохранить на форме» добавления или изменения места, персоны или события | 1 Открывается форма с предупреждением о том, что нужно вписать по крайней мере имя персоны или названия места или события |

Продолжение таблицы 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Идентификатор | Приоритет | Модуль | Подмодуль | Заглавие (суть)  теста-кейса | Ожидаемый результат |
| 1 | 2 | 3 | 4 | 5 | 6 |
| T1.25 | А | Приложение | TreeEditor | Открытие формы для изменения персоны  1 Выделение одной из персон и нажатие на кнопку «Изменение» | 1 Открывается форма с полями для изменения информации о персоне и фото, в которых есть текущая информация |
| T1.26 | А | Приложение | AddEdit-Person | Изменение персоны  1 Оставление не пустым по крайней мере поля для имени и нажатие на «Сохранить» на форме изменения места | 1 Запись меняется, таблица обновляется |
| Т 1.27 | А | Приложение | TreeEditor | Открытие формы для изменения места  1 Выделение одного из мест и нажатие на кнопку «Изменение» | 1 Открывается форма с полями для изменения информации о месте и фото, в которых есть текущая информация |
| T 1.28 | A | Приложение | AddEdit-Place | Изменение места  1 Оставление не пустым по крайней мере поля для названия и нажатие на «Сохранить» на форме изменения места | 1 Запись меняется, таблица обновляется |
| T 1.29 | A | Приложение | AddEdit-Event | Открытие формы для добавления события  1 Выделение одного из событий и нажатие на кнопку «Изменение» | 1 Открывается форма с полями для изменения информации о событии и фото, в которых есть текущая информация |

Продолжение таблицы 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Идентификатор | Приоритет | Модуль | Подмодуль | Заглавие (суть)  теста-кейса | Ожидаемый результат |
| 1 | 2 | 3 | 4 | 5 | 6 |
| T1.30 | А | Приложение | AddEditEvent | Изменение события  1 Оставление не пустым по крайней мере поля для названия и нажатие на «Сохранить» на форме изменения события | 1 Запись меняется, таблица обновляется |
| T1.31 | А | Приложение | TreeEditor | Открытие формы для настройки древа  1 Нажатие на пункт меню «Древо» | 1 Открывается форма с полями для изменения настроек будущего древа |
| Т 1.32 | А | Приложение | Tree | Применение настроек древа  1 Ввод или выбор необходимых настроек и нажатие на кнопку «Создать» | 1 Форма обновляется на изображение древа с примененными настройками |
| T 1.33 | A | Приложение | Tree | Сохранение древа  1 Нажатие на кнопку с элементами меню  2 Выбор опции «Сохранить»  3 Выбор директории и ввод имени файла, и нажатие ОК | 1 Возле кнопки всплывает меню с тремя опциями  2 Появляется диалог выбора имени и директории файла  3 Файл успешно сохраняется на компьютер |

При тестировании программного продукта, в первую очередь следует обратить внимание на корректную работу всех функций. Также следует проверить всю информацию на орфографию.

По подготовленным тестам будет осуществляться функциональное и полное тестирование программного продукта. Отчет о результатах тестирования будет представлен в 4 разделе пояснительной записки.

**3 Построение программы**

Диаграмма классов – структурная диаграмма, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов и взаимосвязей между ними. Она предназначена для представления внутренней структуры программы в виде классов и связей между ними.

У каждого класса должно быть только одно назначение и уникальное осмысленное имя, которое будет связано с этой целью.

Класс должен быть представлен в виде блока с двумя или тремя отделениями. Самый верхний из них содержит имя класса. Средний содержит атрибуты класса, если они имеются, а последний – методы класса, если они имеются. Атрибуты (поля и свойства) класса записываются следующим образом: сначала идет название атрибута, после него – его тип. Функции записываются похожим образом: сначала записывается название функции, в скобках – её параметры, потом ставится двоеточие и прописывается тип возвращаемого значения.

Членам класса (атрибутам и методам) назначена определенная видимость. В таблице 2 показано, как эту видимость можно представить на диаграмме классов.

Таблица 2 – Представление области видимости в диаграмме классов

|  |  |  |
| --- | --- | --- |
| Модификатор доступа | Обозначение в диаграмме классов | Область видимости |
| public | + | В любом месте программы и может быть вызван любым объектом внутри системы |
| private | - | Только определяющий класс |
| protected | # | Только определяющий класс или его подкласс (наследник) |
| package | ~ | Экземпляры других классов в пределах одного и того же пакета |

Для представления области видимости нужно поставить соответствующее модификатору доступа обозначение до атрибута или метода.

Пример обозначения класса представлен на рисунке 11. У этого класса есть только закрытые атрибуты (строковый, вещественный и целочисленный тип) и методы, не принимающие параметров и возвращающие вещественное число и список с целочисленными числами.

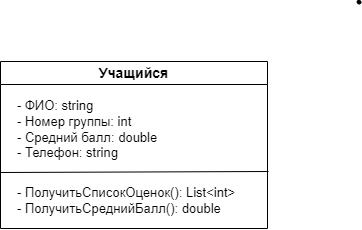


Рисунок 11 – Пример обозначения класса в диаграмме классов

Если значение атрибута установлено по умолчанию, после обозначения его типа нужно поставить «=» и написать это значение.

Если метод не принимает никаких параметров, круглые скобки после него надо оставить пустыми.

Также можно описать следующие отношения между классами:

* Ассоциация: это связь между двумя отдельными классами. Он объединяет две совершенно отдельные сущности. Существует четыре различных типа ассоциации: двунаправленная, однонаправленная, агрегация (включая композиционную агрегацию) и рефлексивная. Наиболее распространенными являются двунаправленные и однонаправленные ассоциации.
* Наследование: указывает на то, что дочерний элемент (подкласс) считается специализированной формой родительского элемента (суперкласса);
* Реализация/Имплементация: связь между двумя элементами модели, в которой один элемент модели реализует/выполняет поведение, заданное другим элементом модели;
* Зависимость часто представляется в виде частной формы использования, которое фиксирует отношение между клиентом, запрашивающим услугу, и сервером, предоставляющим эту услугу;
* Агрегация – особая форма ассоциации, которая представляет собой однонаправленную (то есть одностороннюю) связь между классами. Лучший способ понять эти отношения - назвать их отношениями “имеет” или “является частью”. Например, рассмотрим два класса: кошелек и деньги. В кошельке “есть” деньги. Но деньгам необязательно иметь кошелек, так что это однонаправленная связь;
* Композиция – это ограниченная форма агрегации, при которой две сущности (или, можно сказать, классы) сильно зависят друг от друга.

Для того, чтобы отобразить определенную связь между двумя классами, их достаточно связать стрелкой с соответствующим этой стрелке указателем (рисунок 12).

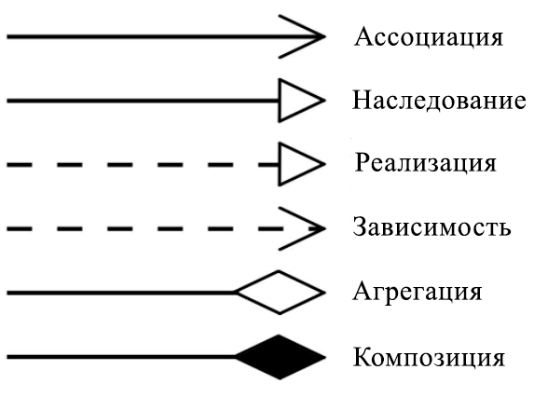


Рисунок 12 – Обозначения связей в диаграмме классов

В итоге, была получена следующая диаграмма классов (приложение В). Ее описание приведено в таблице 3.

Таблица 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Класс | Поля | Методы | Описание метода | Свойства | Значение |
| 1 | 2 | 3 | 4 | 5 | 6 |
| People | Relates | DeletePerson | Удаление персоны | BirthDate,  BirthPlaceId,  DeathDate,  DeathPlaceId,  CauseOfDeath,  FatherId,  FullName,  FirstName,  Gender, Age  IsAdult,  IsAlive,  IsSenior,  LastName,  MaidenName,  MiddleName,  MotherId,  Notes, Role,  Occupation,  PersonId,  Photo, Religion,  ResidenceId,  ZodiacSign | Класс для работы с таблицей «People» |
| GetPeople | Загрузка таблицы «People» из базы данных в список |
| ShowPerson | Просмотр персоны |
| EditPerson | Изменение персоны |
| AddPerson | Добавление персоны |
| ToString | Переопределение метода для приведения к строке для вывода людей и их роли в добавлении событий |
| Place | - | DeletePlace | Удаление места | Abbreviation  Address  Comment  Description  HistoricalName  Name  Picture  Type | Класс для работы с таблицей «Places» |
| GetPlaces | Загрузка таблицы «Events» базы данных в список |
| ShowPlace | Просмотр места |
| EditPlace | Изменение места |
| AddPlace | Добавление места |

Продолжение таблицы 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Класс | Поля | Методы | Описание метода | Свойства | Значение |
| 1 | 2 | 3 | 4 | 5 | 6 |
| Events | - | DeleteEvent | Удаление события по номеру из БД | Comment  DaysUntil-Anniversary  Description  EventDate  EventId  EventPlaceId  EventStatus  Formatted-EventDate  Picture  Type | Класс для работы с таблицей «Events» |
| GetEvents | Загрузка таблицы «Events» из базы данных в список |
| ShowEvent | Просмотр события |
| EditEvent | Изменение события в БД |
| AddEvent | Добавление события в БД |
| Partici-pants | - | - | - | EventId  PersonId  Role | Класс для работы с таблицей «Participants», напрямую никак не меняется и не просматривается пользователем, только при работе с «People» и «Events» |
| Tree-Editor | database  events  people  places  sort-Index | TreeEditor | Конструктор класса | - | Форма редактора базы данных древа |
| Add | Кнопка «Добавить» |
| Show | Кнопка «Показать» |
| Delete | Кнопка «Удалить» |
| CreateTree | Нажатие на пункт меню «Древо» |
| Switch-Events | Нажатие на пункт меню «События» |
| Switch-Places | Нажатие на пункт меню «Места» |
| Switch-People | Нажатие на пункт меню «Персоны» |
| Reset | Сброс данных в таблице |
| Sort | Сортировка |
| Activated | Активация формы |
| Closed | Закрытие формы |
| Loaded | Открытие формы |
| OpenDev | Открыть страницу разработчика |
| OpenHelp | Открыть справку |
| Open-Example | Открыть пример |

Продолжение таблицы 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Класс | Поля | Методы | Описание метода | Свойства | Значение |
| 1 | 2 | 3 | 4 | 5 | 6 |
| Add-Edit-Person | database  image  addOr-Edit  personId | Adding-Person | Конструктор класса | - | Форма для добавления либо редактирования персон |
| SavePerson | Сохранить персону |
| EditPerson | Редактировать персону |
| Change-Gender | Изменение базовой картинки в зависимости от пола |
| Close | Закрытие формы |
| LoadPhoto | Нажатие на кнопку «Открыть фото» |
| DeletePhoto | Нажатие на кнопку «Удалить фото» |
| Add-Edit-Place | database  image  addOr-Edit  placeId | Adding-Place | Конструктор класса | - | Форма для добавления либо редактирования мест |
| SavePlace | Сохранить место |
| EditPlace | Редактировать место |
| Close | Закрытие формы |
| LoadPhoto | Нажатие на кнопку «Открыть фото» |
| DeletePhoto | Нажатие на кнопку «Удалить фото» |
| Add-Edit-Event | database  image  addOr-Edit  eventId | Adding-Event | Конструктор класса | - | Форма для добавления либо редактирования cобытий |
| LoadPhoto | Нажатие на кнопку «Открыть фото» |
| DeletePhoto | Нажатие на кнопку «Удалить фото» |
| Delete-Participants | Удалить участника из списка участников |
| Add-Participants | Добавить участника в список участников |
| SaveEvent | Сохранить событие |
| EditEvent | Редактировать событие |
| Close | Закрытие формы |
| Select-Participant | Изменение ролей при выборе участника |

Продолжение таблицы 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Класс | Поля | Методы | Описание метода | Свойства | Значение |
| 1 | 2 | 3 | 4 | 5 | 6 |
| Tree | database | Tree | Конструктор формы | - | Форма для добавления либо редактирования персон |
| Save | Сохранение дерева |
| Increase | Увеличение счётчика |
| Decrease | Уменьшение счётчика |
| Choose-Color | Нажатие на цветной кружок для выбора цвета блока |
| Closed | Закрытие формы |
| GetColor | Получения цвета блока |
| GetShape | Получение названия формы блока |
| Show | database  image  id | Show | Конструктор формы | - | Форма для добавления либо редактирования мест |
| FindGeo | Нажатие по кнопке геолокации |
| ShowAgain | Переход по ссылкам |
| ShowEvent | Просмотр события |
| ShowPerson | Просмотр персоны |
| ShowPlace | Просмотр места |
| SizeChanged | Динамическое изменение размера формы в зависимости от наполнения |
| Main-  Window | - | MainWindow | Конструктор класса | - | Форма для добавления либо редактирования cобытий |
| ChooseExist-  FamilyTree | Выбор существующего файла базы данных |
| CreateNew-FamilyTree | Создание нового файла базы данных |
| Closed | Закрытие формы (закрывает полностью всё приложение) |
| Warning | - | Warning | Конструктор класса | - | Форма для уведомлений об ошибках |
| OKClick | Закрытие формы |
| Show-Warning | Открытие формы с параметрами заголовка и текста ошибки |

**4 Тестирование**

Тестирование является одним из самых важных этапов при создании программного продукта. Исходя из последующего внедрения программы в использование, необходимо осуществить тестирование по набору тестов, для того чтобы убедится в корректной работе ПП.

Опираясь на данный набор тестов, можно говорить о некоторой степени уверенности в правильности работы программного продукта.

Таким образом, основная часть ошибок и недоработок была выявлена и исправлена на этапе реализации проекта. После завершения этапа реализации было проведено тщательное тестирование.

Программный продукт был протестирован разработчиком – Кузьменок П.С. Расписание проведения и время, затраченное на тестирование, описано в таблице 4.

Таблица 4 - Расписание проведения тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| ФИО | Дата | Описание | Длительность, ч |
| Кузьменок П.С. | 25.06.2023 | Выполнение тест-кейсов | 1 |
| Кузьменок П.С. | 26.06.2023 | Анализ выполнения тест-кейсов | 1 |
| Кузьменок П.С. | 27.06.2023 | Повторное выполнение тест-кейсов | 1 |

Выявление по результатам ошибки тестирования приводятся в таблице 5.

Таблица 5 - Статистика по выявлению ошибок.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Статус | Количество ошибок | Важность | | | |
| Критическая | Высокая | Средняя | Минимальная |
| Исправлено | 11 | 4 | 0 | 4 | 4 |
| Проверено | 11 | 0 | 0 | 0 | 0 |
| Открыто заново | 0 | 0 | 0 | 0 | 0 |
| Найдено | 11 | 4 | 0 | 4 | 3 |
| Отклонено | 0 | 0 | 0 | 0 | 0 |

Таким образом, после проведения тестирования критических ошибок выявлено не было, однако были найдены ошибки, связанные с некорректным отображением некоторых элементов графического интерфейса.

Также были обнаружены и исправлены ошибки, связанные с отказом в доступе при создании файлов уже после установки проекта из установщика.

Опираясь на вышеперечисленное, можно сделать вывод о стабильности работы программного продукта и его основного функционала. Все ошибки, влияющие на работу приложения были устранены.

Таблица 6 – Статистика по тест-кейсам

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество | Пройдено | Не пройдено | Заблокировано | Не протестировано |
| 33 | 22 | 11 | 0 | 0 |

Рисунок 13 – Диаграмма тест-кейсов

**5 Применение**

**5.1 Назначение и условия применения программы**

Данное приложение представляет собой программное обеспечение для создания, редактирования и визуализации генеалогических деревьев.

Программа имеет такие функции как:

* Открытие файла базы данных;
* Создание нового проекта;
* Добавление записей в таблицы;
* Изменение записей в таблицах;
* Удаление записей в таблицах;
* Фильтрация записей в таблицах;
* Просмотр записей в таблицах;
* Настройка параметров древа;
* Создание визуального отображения древа;
* Сохранение визуального отображения древа;
* Просмотр справки.

Минимальными системными требованиями для нормального функционирования данного программного продукта являются:

* Операционная система: Windows 7 или более поздняя версия;
* Процессор: Intel Core i3 или эквивалентный процессор;
* Оперативная память: минимум 4 ГБ RAM;
* Свободное место на жестком диске: минимум 1 ГБ;
* Разрешение экрана: минимум 1280x800 пикселей.

**5.2 Инсталляция**

Для установки данного программного продукта необходимо вставить USB-флеш-накопитель в компьютер. Далее запустить файл «Setup.exe» (рисунок 14).



Рисунок 14 – Файл «Setup.exe»

Появится первоначальное окно установщика (рисунок 15). Нажмите «Далее».

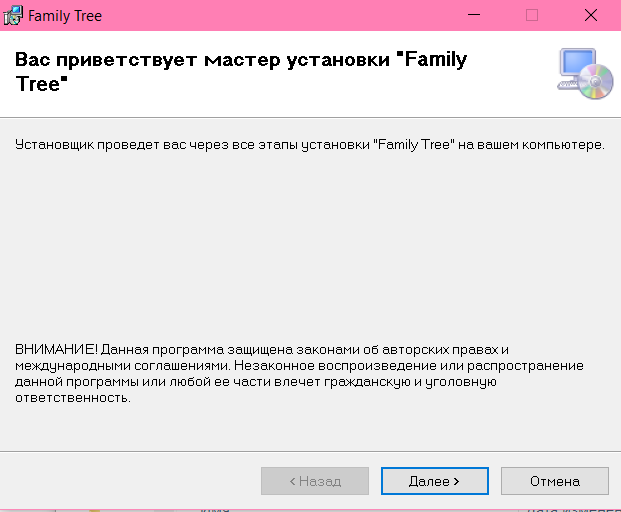


Рисунок 15 – Установка

В окне «Выбор папки установки» можно выбрать место для установки приложения (рисунок 16). Для выбора другого корневого каталога нажмите «Обзор…».

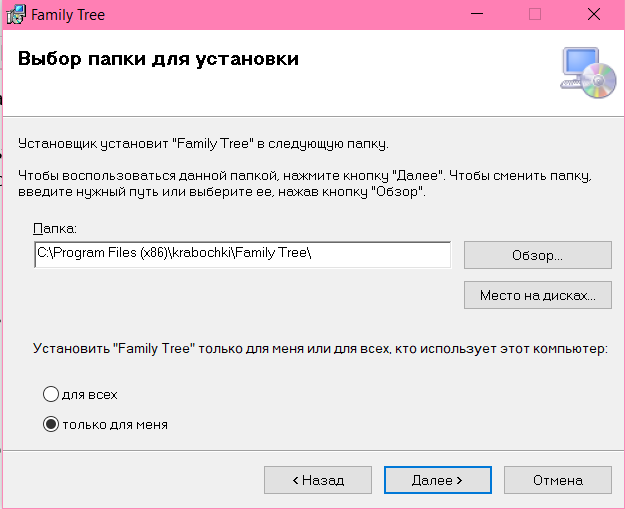


Рисунок 16 – Выбор папки установки

Далее появится окно, оповещающее о том, что все готово для начала установки (рисунок 17). Нажмите «Установить».

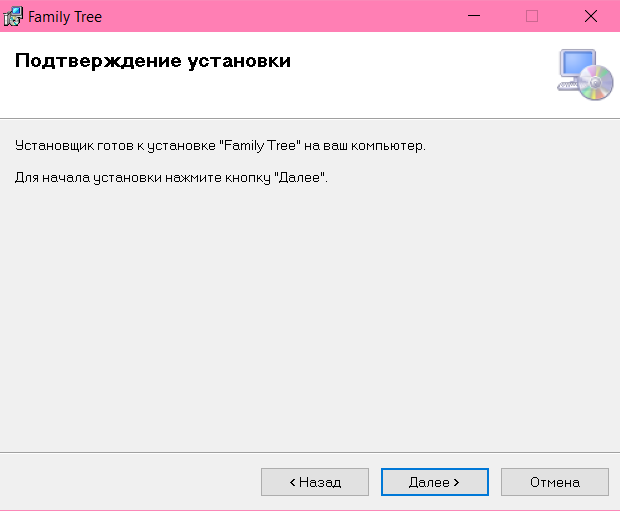


Рисунок 17 – Готовность к установке

Следующим шагом будет установка программы (рисунок 18) и оповещение об успешной установке (рисунок 19).

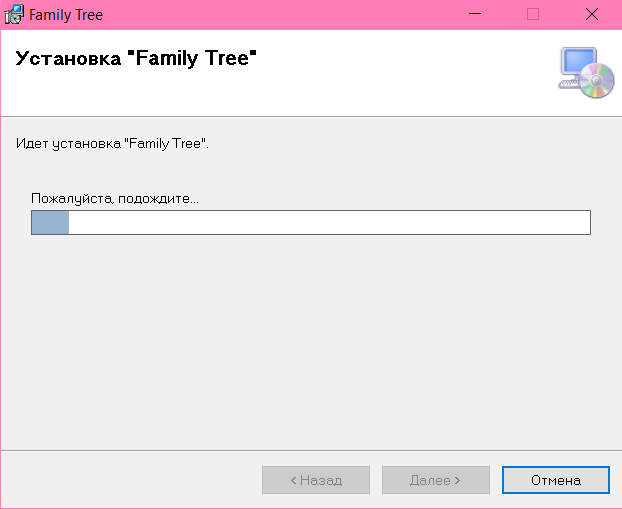


Рисунок 18 – Инсталляция

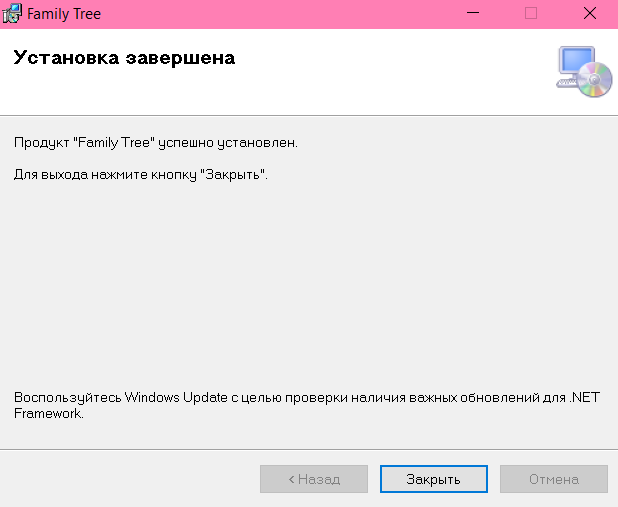


Рисунок 19 – Завершение установки

**5.3 Выполнение программы**

Для того чтобы запустить приложение запустите файл «Family Tree» (в виде ярлыка на рабочем столе) (рисунок 20).



Рисунок 20 – Файл «Family Tree»

После запуска программы появится окно главной формы (рисунок 21). На этой форме пользователь может открыть либо же создать генеалогическое древо:



Рисунок 21 – Главная форма

Пользователь может создать новую папку для проекта, в которой будет создан файл базы данных и папки с ресурсами. Это предоставляет возможность начать работу над новым генеалогическим древом.

Для начала, нужно нажать на кнопку "Создать древо" (рисунок 21):



Рисунок 22 - Кнопка «Создать древо»

После этого появится диалог для выбора или создания новой папки, в которой позже будет создан проект (рисунок 23):

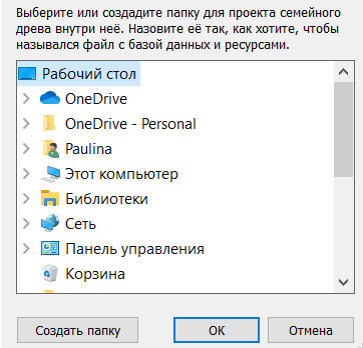


Рисунок 23 - Диалог для выбора папки проекта

Таким образом, пользователь может либо создать папку, либо выбрать существующую и нажать кнопку "ОК". Если пользовать нажмёт кнопку отмена, диалог выбора папки исчезнет, и у пользователя снова появится выбор между созданием и открытием проекта.

Если пользователь всё же нажал кнопку "ОК", в выбранной папке создается база данных с расширением .accdb и имеющая название такое же, как имя папки. Также, в выбранной папке создаётся папка, начинающаяся с названия выбранной папки и оканчивающаяся на "-Resources". Эта папка является папкой ресурсов проекта семейного древа, таких как изображения. В этой папке создаютя ещё три папки - People, Places и Events, куда в будущем, соответственно, будут копироваться все файлы изображений для записей данных таблиц. Например, если создать и/или выбрать папку с названием My Tree, после этого её состав пополнится следующими элементами (рисунок 24, 25):

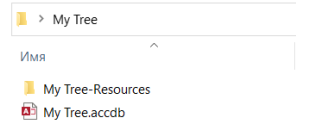


Рисунок 24 - Состав папки проекта

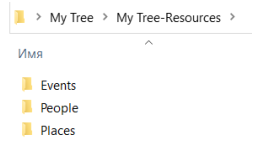


Рисунок 25 - Состав папки проекта с ресурсами

Если файл с базой данных с названием, как у папки, внутри уже существует, то создать новый проект в ней пока не получится: приложение предупредит о том, что папка уже содержит проект (рисунок 26).

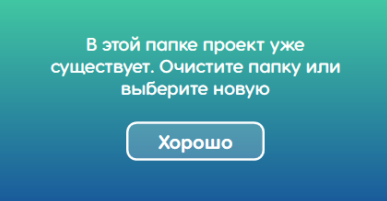


Рисунок 26 - Предупреждение о некорректности папки

Для открытия существующего файла базы данных древа, нужно нажать на кнопку "Открыть древо" на главной форме при открытии приложения (рисунок 27):



Рисунок 27 – Кнопка «Открыть древо»

После этого появится диалог для выбора базы данных древа, с которым позже можно будет продолжить работу (рисунок 28):

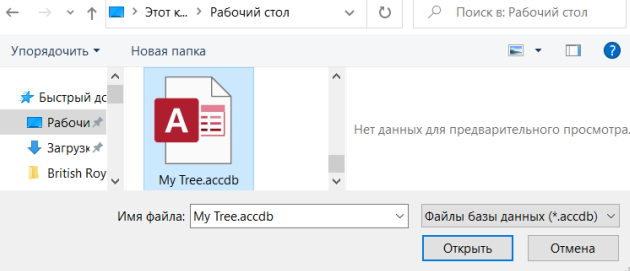


Рисунок 28 - Диалог для выбора базы данных древа

Если пользовать нажмёт кнопку отмена, диалог выбора файла исчезнет, и у пользователя снова появится выбор между созданием и открытием проекта.

В отличие от создания нового проекта, здесь пользователь должен выбрать именно файл с ограничением по расширению (только файлы базы данных .accdb), который ранее был создан при создании проекта. Если пользователь ничего не менял в папке, то приложение успешно обнаружит в дальнейшем папки с ресурсами и сможет их использовать. Однако, если по какой-то причине пользователь переместил файл, приложение найти потерянную папку уже не сможет. Тем не менее, для корректной работы оно создаст новые папки с ресурсами по тому же принципу, как при создании, в папке, в которой находится файл базы данных древа. После этого все изображения будут копироваться туда, а если эти папки были на месте, то в них.

Несмотря на ограничение по расширению, пользователи могут ошибиться с конкретным файлом базы данных. По этой причине, при открытии файла базы данных, не созданного ранее в этом приложении, пользователь встретит ошибку (рисунок 29):

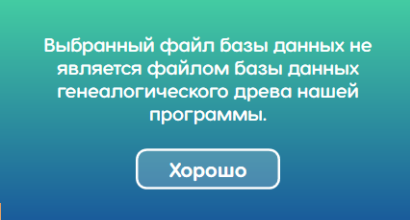


Рисунок 29 - Ошибка при открытии постороннего файла с базой данных

При открытии либо успешном создании нового проекта генеалогического древа, будет открыта форма редактора древа. Если проект был не создан, а открыт, и ранее заполнялся, то в таблице при открытии формы будет содержание таблицы Персоны (рисунок 30):

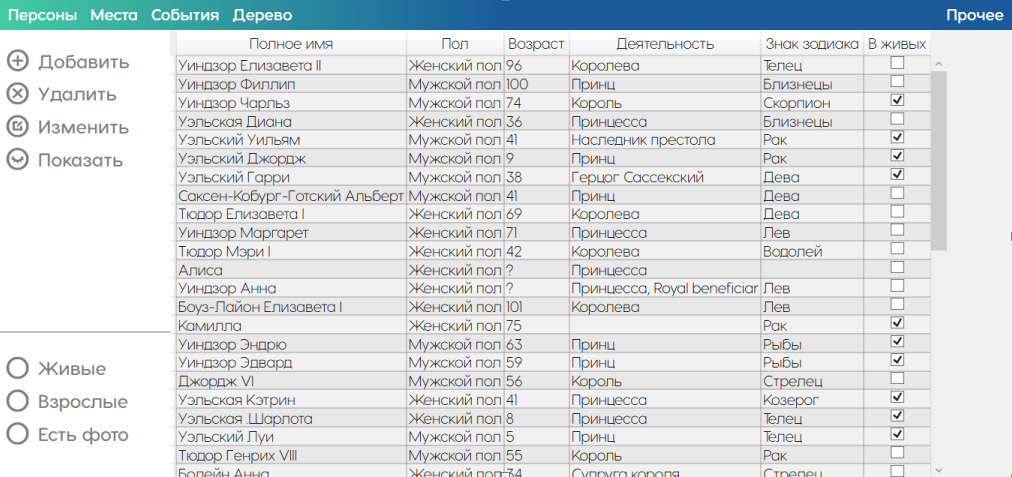


Рисунок 30 - Форма редактора древа

Каждый проект генеалогического древа имеет следующую структуру – таблица персон, таблица мест, таблица событий и таблица участников. Таблица персон нужна для работы с людьми, которые будут составлять основу будущего генеалогического древа. Таблица мест нужна для создания мест, которые будут так или иначе связаны с персонами и событиями – место рождения, место смерти, место проживания, место события. Таблица события нужна для фиксирования памятных событий в жизни семьи. Они могут как влиять на вид генеалогического древа (брак, развод), так и не влиять (путешествие, поступление, первый день в школе). Ключевыми в таблице События являются участники, это персоны, которые так или иначе связаны с событием. Таблица участники же напрямую никак не просматривается и не изменяется пользователем, на её записи можно повлиять только посредством редактирования, добавления или удаления персон и событий. Она нужна для связи события и его участников, а также для фиксирования роли персоны в том или ином событии.

Основные функции в программе выполняются посредством взаимодействия с меню на форме редактора древа (рисунок 31):



Рисунок 31 - Меню на форме редактора древа

С помощью кнопок Персоны, Места, События пользователь может перемещаться по одноименным таблицам и просматривать их содержание (рисунок 32, 33). Также, при нажатии на одну из этих кнопок, основные кнопки для работы с таблицами теперь будут выполнять функции, относящиеся именно к этой таблице. При переключении между таблицами они также обновляются - например, если пользователь добавил новую персону и переключился на события, там появится новое событие "Рождение", относящееся к добавленной персоне несмотря на то, что ранее его там не было.

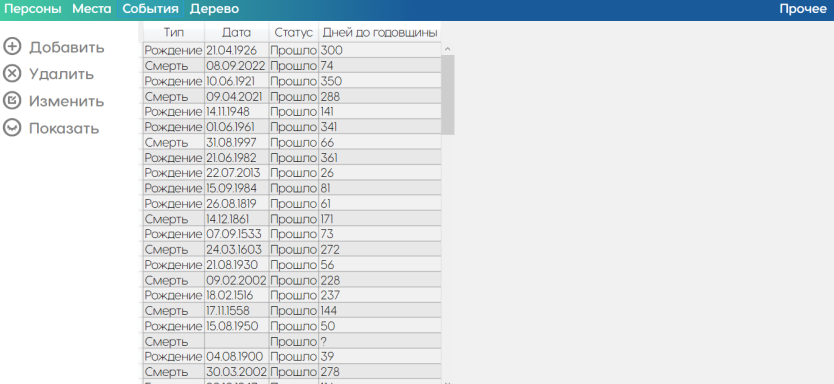


Рисунок 32 - Форма при нажатии на пункт меню "События"

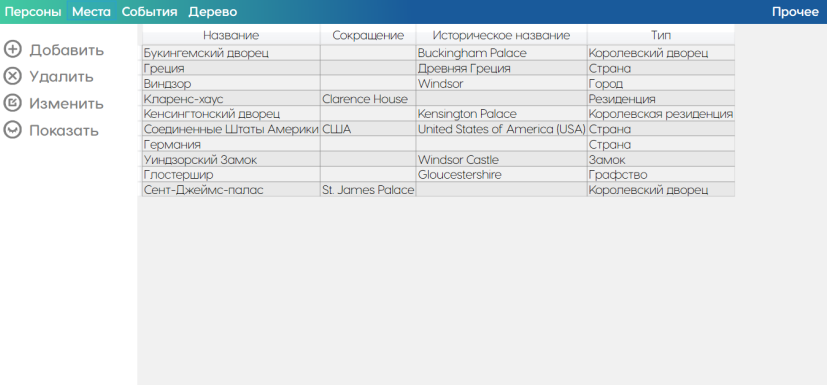


Рисунок 33 - Форма при нажатии на пункт меню "Места"

Все действия, возможные с тремя таблицами, в целом одинаковы и их смысл не меняется между таблицами. Всего для трех таблиц есть четыре действия – это добавление, удаление, изменение и удаление (рисунок 34).

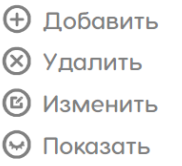


Рисунок 34 – Основные действия с таблицами

Для таблицы персон также отдельно есть возможность фильтрации (рисунок 35):

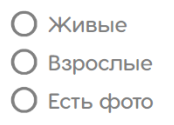


Рисунок 35 – Фильтры для таблицы "Персоны"

При попытке нажать на кнопку "Изменить", "Удалить" или "Показать" с не выбранной записью в таблице (рисунок 36) или выбранными более чем одной записями (рисунок 37), пользователь получит предупреждение:

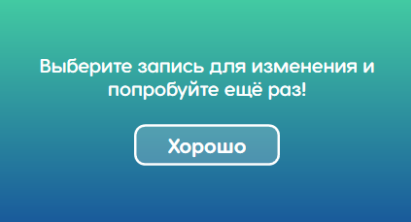


Рисунок 36 - Предупреждение об отсутствии объекта для изменения

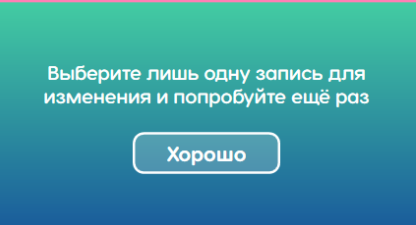


Рисунок 37 - Предупреждение о том, что записей много для изменения

При добавлении персоны (рисунок 38), места (рисунок 39), события (рисунок 40), появляется новая соответствующая форма для ввода информации о новой записи:

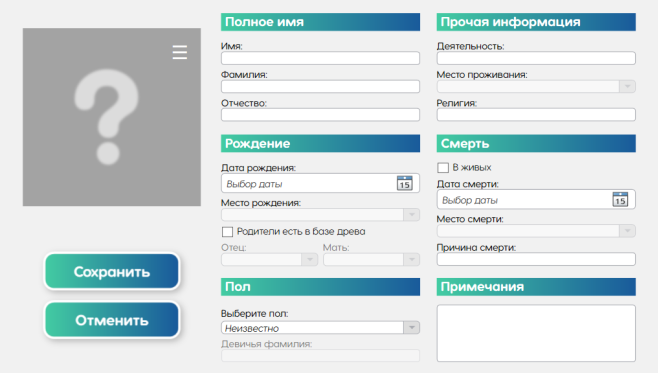


Рисунок 38 - Добавление новой персоны

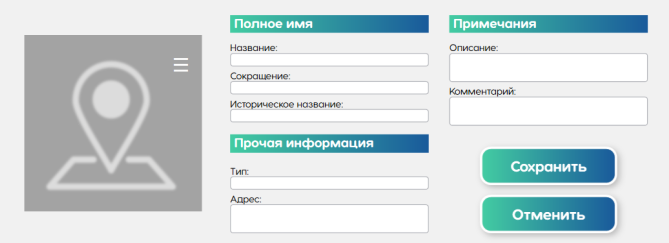


Рисунок 39 - Добавление нового места

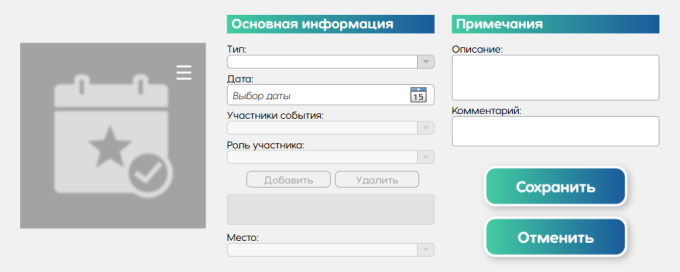


Рисунок 40 - Добавление нового события

При добавлении новой персоны некоторые поля сначала недоступны для изменения. Это поля для девичьей фамилии (активируется при выборе женского пола), поля для выбора матери отца (доступны при активном флажке в опции "Родители есть в базе древа"). Это справедливо для всех баз данных древ независимо от их заполненности.

Однако, если таблица персон пуста, поставить флажок в опции наличия родителей не получится, ведь их не может пока что быть. Дело в том, что и мать, и отец должны быть выбраны только из присутствующих в базе персон. Такие же ограничения накладываются на базы без мест - выбор места смерти, места рождения и места проживания невозможен.

Также при добавлении персоны и отметке её "В живых", поля для описания смерти станут недоступны – ведь персона жива (рисунок 41).

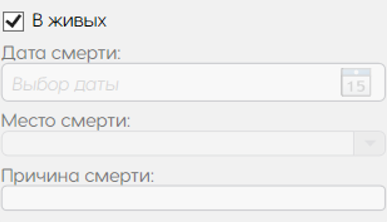


Рисунок 41 - Неактивность полей для ввода

Похожие ограничения присутствуют и при добавлении событий: изначально поля участников, ролей, кнопок для добавления и удаления недоступны. Участников можно выбрать только если люди в базе вообще есть, а также если уже выбран тип события. Если людей в таблице нет вообще, то событие не получится добавить - у каждого события должен быть по крайней мере один участник (рисунок 42).

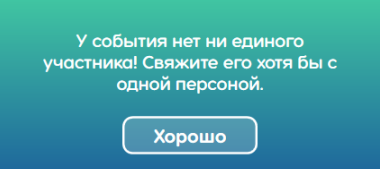


Рисунок 42 - Предупреждение об отсутствии участников

Если тип и участник выбран, пользователь может выбрать его роль и добавить его в список. Чтобы удалить его из списка, его нужно снова выбрать и нажать на кнопку "Удалить".

Содержание ролей генерируется при выборе типа и участника события. Для каждого типа свои участники, а также проверяется пол - например, персона с мужским полом не может выбрать роль "Бывшая жена" (рисунок 43).

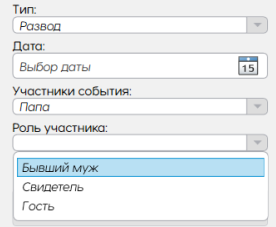


Рисунок 43 - Возможные участники для выбранного типа и персоны

Также при выборе типа события его описание вставляется автоматически (рисунок 44). При желании его можно поменять.

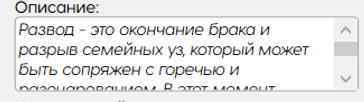


Рисунок 44 - Автоматическое описание

Если человек добавил участников, но решил изменить тип события, участники исчезнут из списка – ведь ранее выбранные роли для нового типа события уже недействительны. Для некоторых типов событий необходимо иметь как минимум 2 участника с конкретными ролями – это события, связанные с отношениями между людьми (рисунок 45).

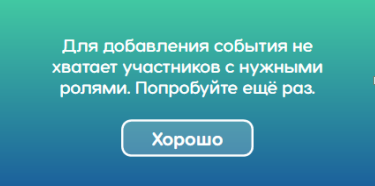


Рисунок 45 - Предупреждение о недостатке участников

Также для таких событий невозможно добавление 2 одинаковых главных ролей (например, две жены в типе "Брак").

Несмотря на вышеперечисленные ограничения, пользователь может продолжить вводить доступную информацию, а после сохранения события добавить недостающие места и персон и изменить нужный элемент благодаря наличию опции "Изменить".

Чтобы сохранить место, персону или событие, нужно нажать на кнопку Сохранить, соответственно для отмены сохранения на кнопку "Отменить". Если персона, место или событие не имеет имени, название или типа соответственно, появится подобное предупреждение (рисунок 46):

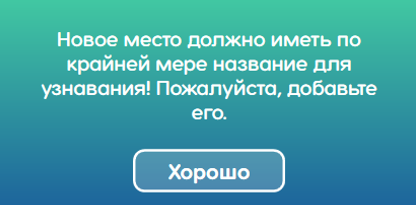


Рисунок 46 - Предупреждение об отсутствии имени

Если добавление персоны произошло успешно, то в таблице "События" появятся 1 или 2 события (в зависимости от того, жива персона или нет) –"Рождение" и "Смерть", где дата соответствующая и базовое описание, а в участниках эта персона. Позже эти события нельзя будет удалить или изменить в них привязанную информацию без изменения или удаления самой этой персоны.

Добавить или удалить фото записям можно нажав на кнопку в верхнем правом углу места для фотографии (рисунок 47):

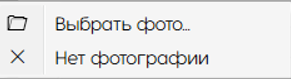


Рисунок 47 - Контекстное меню кнопки изображения

При нажатии на кнопку "Выбрать фото" открывается файловый диалог для выбора файла, при нажатии на кнопку "Нет фотографии" текущая фотография (если она есть) обнуляется. При открытии формы и обнулении фото у событий и мест вместо фото появляется базовое фото для всех событий и мест. У персон иначе – если пол неизвестен, то картинка для неизвестного пола, если пол известен – то базовая картинка для мужчины или женщины.

Для просмотра записи персоны, места или события нужно два раза нажать по ней либо нажать на кнопку просмотреть (рисунок 48):



Рисунок 48 - Просмотр персоны

Просмотр в целом одинаков для всех трех таблиц. Если персона, место или событие не имеют фото, оно просто не будет отображаться (рисунок 49):

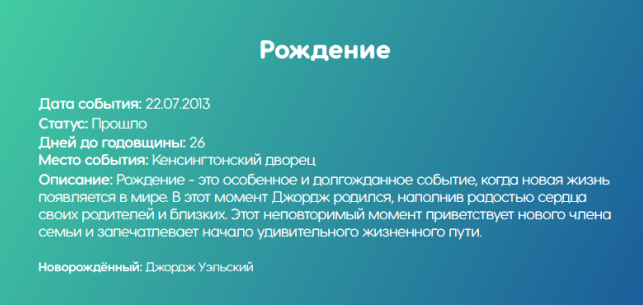


Рисунок 49 - Просмотр события без фото

При просмотре появляется автоматически вычисленная информация: возраст человека либо продолжительность жизни (если персона умерла), знак зодиака для персоны, дни до годовщины и статус для события. Также при просмотре персон или событий, у них появляются список родственников (при их наличии) и участников соответственно.

При просмотре персон или событий, при наличии ссылок на персон или места, по ним можно переходить. При этом будут открываться окно с просмотром соответствующего места или персоны (рисунок 51).

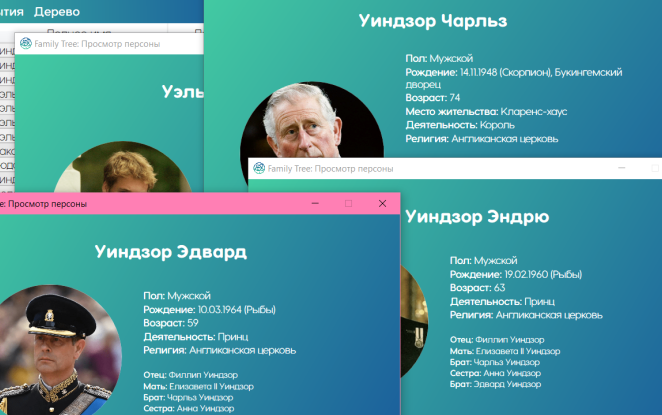


Рисунок 50 - Открытие ссылок

Основная форма просмотра открыта в режиме диалога (работать с формой сзади неё нельзя, пока она не будет закрыта), а последующие открытые ссылки открыты как обычные формы, чтобы можно было переключиться на формы просмотра сзади них.

При просмотре места, если оно имеет адрес, можно нажать на значок геолокации в верхнем правом углу (рисунок 51), и перейти на сайт в интернете (Google Maps) с картой данного места.



Рисунок 51 - Значок геолокации

Чтобы изменить запись, нужно выбрать нужную запись и нажать кнопку "Изменить". При изменении записи открывается та же форма, что и при создании для соответствующих таблиц. Отличие состоит в том, что в полях уже есть введенная информация о записи (рисунок 52).

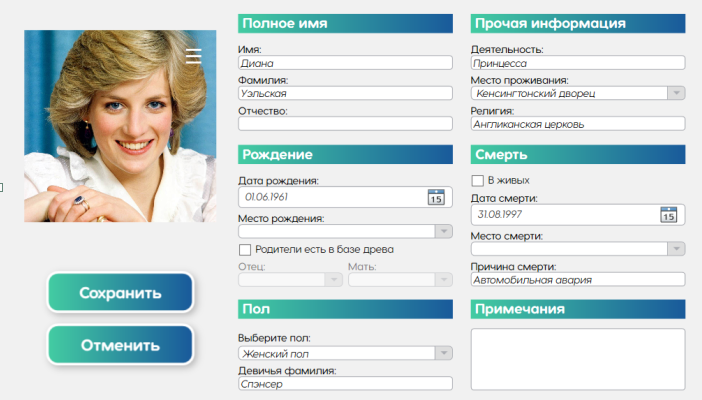


Рисунок 52 - Изменение персоны

Ограничения по видимым и доступным полям при добавлении записей действуют и при их изменении. Это же касается и предупреждения из-за отсутствия имени персоны, типа события или названия места.

При изменении имени, даты рождения и смерти, места рождения и смерти, изменения состояния флажка "В живых" или пола, происходят изменения с событиями соответствующей персоны.

Во-первых, соответствующие события рождения и смерти меняются (обновляются даты и места, описание). Во-вторых, если персона стала умершей, добавляется событие смерти. Если наоборот - оно удаляется из базы.

Также, при изменении пола меняются глаголы в описаниях событий рождения и смерти. Если персона была отцом или матерью у кого-либо из персон в базе, она перестает быть таковой. Если персона была участником в событиях по типу брак, свадьба, развод, эти события удаляются. Остальные события остаются нетронутыми.

При изменении событий рождения или смерти можно менять только список участников, комментарий или описание (рисунок 53). При этом из участников нельзя удалить или добавить умершего или родившегося, можно добавить лишь участников, не связанных с основной сутью события (нельзя добавить умершего, только, например, свидетеля).

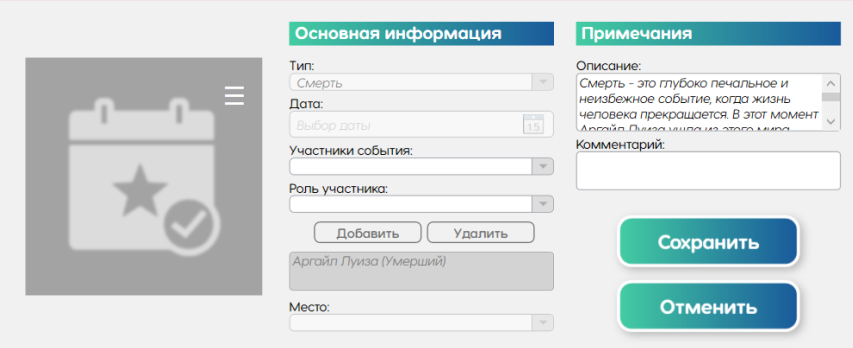


Рисунок 53 – Изменение события «Смерть»

Условия для базовых картинок такие же, как при добавлении записи. При изменении записи также можно удалить существующую картинку либо загрузить новую по кнопке в верхнем правом углу картинки.

При нажатии на кнопку "Сохранить" запись сохраняется с введенными изменениями, при нажатии на кнопку "Отменить" изменение записи отменяется, и форма закрывается.

Для удаления записи нужно выбрать нужную запись и нажать на кнопку удалить. При этом действии появится диалоговое окно подобного вида (рисунок 54):

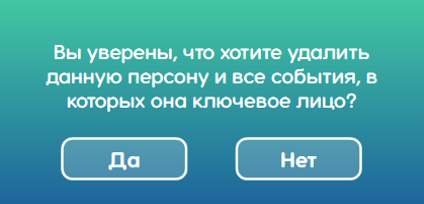


Рисунок 54 – Подтверждение удаления персоны

Соответственно, при нажатии на кнопку "Да", запись будет удалена, при нажатии на кнопку "Нет" удаление отменится. При удалении мест все ссылки на них при просмотре, изменении и добавлении мест и событий пропадут. При удалении персоны все ссылки на них при просмотре, изменении и добавлении персон и событий пропадут. Исчезнут все события, в которых персона была единственным участником, события смерти и рождения данной персоны (независимо от количества участников), события, где персона была ключевым участником (например жена в браке). Если персона была одним из участником события, но не являлась главным участником (например, гость), она исчезает из списка участников, но событие остается. Если персона была у кого-либо матерью или отцом, она перестает ими быть.

Для создания визуального отображения древа по нажатию на пункт меню «Древо» в таблице должно быть по крайней мере два человека – в противном случае создание древа не имело бы никакого смысла (рисунок 55).

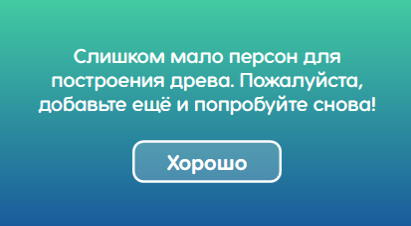


Рисунок 55 - Предупреждение о недостатке персон

Если персон достаточное количество, появится форма для подробной настройки внешнего вида будущего древа (рисунок 56).

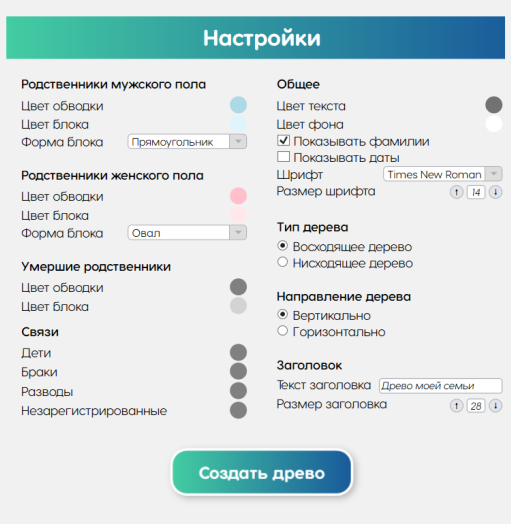


Рисунок 56 - Форма для настройки внешнего вида древа

Данная форма показывается в режиме диалога, это значит, что пока пользователь её не закроет, он не сможет продолжить работу с формой сзади. Это сделано для недопущения избыточного количества открытых форм.

Пользователь может настраивать шрифты, фон, вид блоков для женского, мужского пола и умерших персон, а также цвета для связей между персонами. Так как в будущем древе будет четыре типа связей – "Ребенок-Родитель", "Муж-Жена", "Бывший муж-Бывшая жена", "Отец ребенка-Мать ребенка" (для незарегистрированных отношений между родителями одной персоны), их можно обозначить по-разному.

Эти характеристики устанавливаются по разному: цвета можно выбрать при нажатии на соответствующий круг и выборе цвета из диалога выбора цветов (рисунок 57), при нажатии на список можно выбрать формы блоков и шрифт (рисунок 58), при нажатии на переключатели можно изменить направление и тип дерева (рисунок 59), при нажатии на флажки можно включить либо выключить отображение фамилий, и дат (рисунок 60). Текст заголовка нужно вводить в текстовое поле (рисунок 61), а размеры шрифтов нужно настраивать с помощью двух кнопок "больше" и "меньше" (рисунок 62)

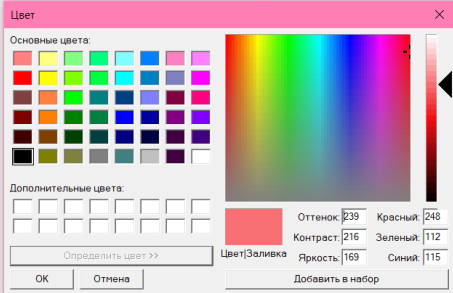


Рисунок 57 - Диалог выбора цветов



Рисунок 58 - Выпадающий список

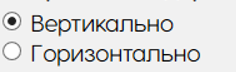


Рисунок 59 - Переключатели



Рисунок 60 - Флажки



Рисунок 61 - Текстовое поле



Рисунок 62 - Кнопки "Больше", "Меньше"

При нажатии на кнопку "Создать древо", откроется форма просмотра созданного древа (рисунок 63).



Рисунок 63 - Пример древа случайной семьи с начальными настройками

Построенные таким образом деревья могут рассказать, жива ли персона, к которой относится блок, а также какого она пола. Блоком в виде сердца отображается брак двух людей, блоком в виде разбитого сердца – развод, а блоком в виде паззла – незарегистрированный союз двух людей, у которых есть общие дети. Повлиять на возникновение таких блоков можно с помощью добавления соответствующих событий с нужными персонами.

При перемещении ползунка на элементе справа, изображение древа будет приближаться или отдаляться соответственно. При этом перемещение может осуществляться с полосами прокрутки рисунок 64).

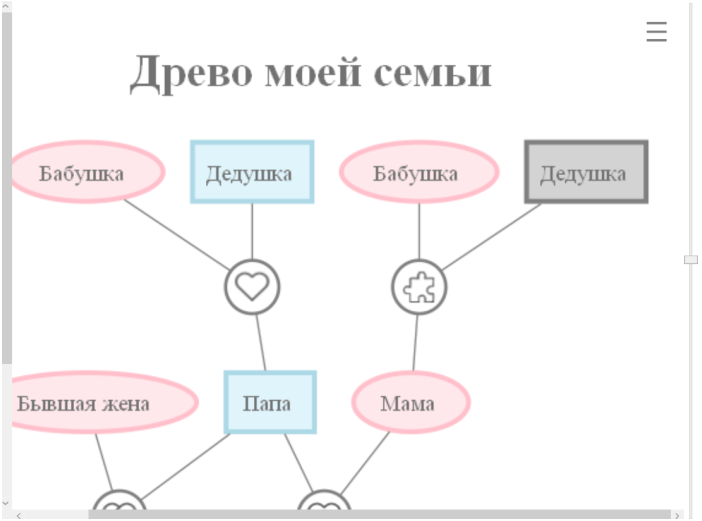


Рисунок 64 – Приближение древа

Также это окно можно развернуть в полный экран или просто менять его размер. При этом изображение тоже растянется по размерам окна.

При нажатии на кнопку в правом верхнем углу появляется контекстное меню (рисунок 65):

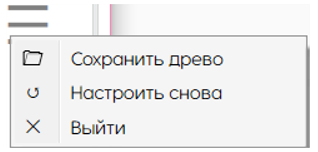


Рисунок 65 - Контекстное меню кнопки

С помощью элементов этого меню можно сохранить древо, настроить его снова (при этом текущая форма просмотра закроется), либо выйти из создания древа вообще – тогда закроется текущая форма просмотра и текущая форма настройки древа.

С помощью элемента контекстного меню правой кнопки в верхнем углу на форме «Просмотра дерева» «Сохранить древо» можно сохранить древо. Для этого нужно нажать на этот пункт и появится диалог выбора папки и названия файла (рисунок 66):

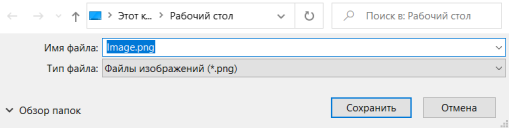


Рисунок 66 - Выбор папки для сохранения изображения древа

Нажатие на кнопку "Сохранить" позволит пользователю сохранить полученное древо в выбранной папке в файле изображения с расширением ".png".

При нажатии на пункт меню «Прочее» появится выпадающий список возле него, который является продолжением меню (рисунок 67):

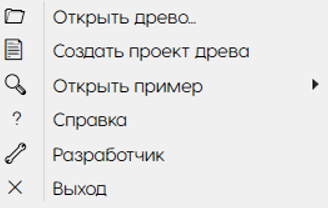


Рисунок 67 - Контекстное меню пункта меню "Прочее"

Все элементы контекстного меню имеют графическое отображение слева от текста для более легкого визуального восприятия.

Пункт меню «Открыть древо» по нажатию выполняет тот же функционал, что и начальная форма при открытии файла базы данных. При этом, если пользователь всё-таки успешно открыл файл, форма с текущей базой данных закрывается и открывается новая с новой базой.

Пункт меню "Создать проект древа" при нажатии делает то же, что и такая кнопка на главной форме. При отмене создания или открытия пользователь может продолжать работу, либо при успешном создании папки или выбора проекта форма закроется и откроется новая с новым проектом.

При нажатии на кнопку «Открыть пример» появится контекстное меню для данной кнопки с тремя пунктами (рисунок 68):

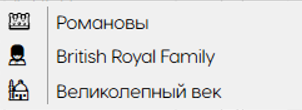


Рисунок 68 - Контекстное меню кнопки "Открыть пример"

При нажатии на какой-либо из пунктов меню, по сути выполнится функциональность кнопки "Открыть древо". Отличие заключается в том, что пользователю не нужно будет выбирать файл, а соответствующий проект откроется сразу и текущий сразу закроется. Открытые проекты представляют собой достаточно подробно заполненные файлы базы данных с множеством персон, мест и событий, а также соответствующими изображениями к ним. Соответственно, будут открыты проекты генеалогических деревьев семьи Романовых, британской королевской семьи либо же генеалогическое древо с ключевыми фигурами султанского османского рода времен Сулеймана Великолепного. Данные примеры нужны для наглядного представления о том, как выглядят заполненные базы данных, как работает их изменение и дополнение, и возможности заранее посмотреть, как будет примерно выглядеть генеалогическое древо без необходимости заполнения таблиц.

При нажатии на пункт меню Справка пользователь сможет просмотреть файл со справкой для приложения.

При нажатии на пункт меню Разработчик пользователь попадет на страницу разработчика в GitHub (крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки).

При нажатии на кнопку Выход все текущие окна программы закрываются, а сама программа заканчивает свою работу.

**Заключение**

Целью курсового проекта было создание приложения для редактирования генеалогических древ.

Приложение предоставляет пользователю возможность открывать существующие проекты, создавать новые папки для проектов и работать с таблицами персон, мест и событий. Пользователь имеет возможность добавлять, редактировать и удалять записи, а также настраивать и строить генеалогические древа.

С помощью данного приложения пользователи могут легко вести свои генеалогические исследования, отслеживать связи между родственниками, сохранять исторические данные и фотографии, а также создавать красочные и наглядные древа, которые помогут визуализировать родословную.

В поставленной задаче был реализован простой и понятный пользовательский интерфейс.

В ходе тестирования программного продукта никаких ошибок выявлено не было. При реализации программного продукта были выполнены все изначально поставленные задачи.

В дальнейшем приложение может быть модифицировано, так как на данный момент функционал недостаточной большой. Можно добавить еще несколько функций разной сложности. Это могло бы сделать программный продукт более интересным и масштабным. Несмотря на все это, данный программный продукт имеет законченный вид.

Исходя из этого, можно сделать вывод, что программа реализована успешно.

.

**Список использованных источников**

1. Официальная документация Microsoft для WPF [Электронный ресурс]. – Электронные данные. – Режим доступа: https://learn.microsoft.com/-ru-ru/dotnet/desktop/wpf/.
2. Metanit: WPF и C#. Полное руководство [Электронный ресурс]. – Электронные данные. – Режим доступа: https://metanit.com/sharp/wpf/.
3. Официальная документация библиотеки и инструментов для WPF на NuGet [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.nuget.org/
4. WPF Tutorial: Руководства и учебники по WPF [Электронный ресурс]. – Электронные данные. – Режим доступа: https://wpf-tutorial.com/.
5. Сообщество разработчиков WPF на Stack Overflow [Электронный ресурс]. – Электронные данные. – Режим доступа: https://stackoverflow.com/-questions/tagged/wpf.

**Приложение А**

**Листинг программы**

Класс Places.cs:

namespace Family\_Tree

{public class Places

{public int PlaceId { get; set; }

public string Name { get; set; }

public string Abbreviation { get; set; }

public string Historical\_Name { get; set; }

public string Type { get; set; }

public string Description { get; set; }

public string Address { get; set; }

public string Comment { get; set; }

public string Picture { get; set; }

public static List<Places> GetPlaces(string nameOfDatabaseFile)

{List<Places> places = new List<Places>();

string connect =$"Provider=Microsoft.ACE.OLEDB.12.0;Data Source={nameOfDatabaseFile};Jet OLEDB:Engine Type=5";

OleDbConnection dbconnection = new OleDbConnection(connect);

dbconnection.Open();

string query = "SELECT \* FROM Places"; //запрос

OleDbCommand dbCommand = new OleDbCommand(query, dbconnection);

OleDbDataReader dbDataReader = dbCommand.ExecuteReader();

while (dbDataReader.Read())

{Places place = new Places();

place.PlaceId = Convert.ToInt32(dbDataReader["PlaceId"]);

place.Name = dbDataReader["Name"] != DBNull.Value

? dbDataReader["Name"].ToString(): string.Empty;

place.Abbreviation = dbDataReader["Abbreviation"] != DBNull.Value

? dbDataReader["Abbreviation"].ToString(): string.Empty;

place.Historical\_Name = dbDataReader["Historical\_name"] != DBNull.Value

? dbDataReader["Historical\_name"].ToString(): string.Empty;

place.Type = dbDataReader["Type"] != DBNull.Value

? dbDataReader["Type"].ToString(): string.Empty;

place.Description = dbDataReader["Description"] != DBNull.Value

? dbDataReader["Description"].ToString(): string.Empty;

place.Address = dbDataReader["Address"] != DBNull.Value

? dbDataReader["Address"].ToString(): string.Empty;

place.Comment = dbDataReader["Comment"] != DBNull.Value

? dbDataReader["Comment"].ToString(): string.Empty;

place.Picture = dbDataReader["Picture"] != DBNull.Value

? dbDataReader["Picture"].ToString(): string.Empty;

places.Add(place);}

dbDataReader.Close();dbconnection.Close();return places;}

public static void DeletePlace(string nameOfDatabaseFile, Places selectedItem)

{string connect =$"Provider=Microsoft.ACE.OLEDB.12.0;Data Source={nameOfDatabaseFile};Jet OLEDB:Engine Type=5";

OleDbConnection dbconnection = new OleDbConnection(connect);

dbconnection.Open();

string query = $"DELETE \* FROM Places WHERE PlaceId={selectedItem.PlaceId}";

OleDbCommand dbCommand = new OleDbCommand(query, dbconnection);

dbCommand.ExecuteNonQuery();

query =$"UPDATE People SET ResidenceId = 0 WHERE ResidenceId={selectedItem.PlaceId}";

dbCommand = new OleDbCommand(query, dbconnection);

dbCommand.ExecuteNonQuery();

query =$"UPDATE People SET BirthPlaceId = 0 WHERE BirthPlaceId={selectedItem.PlaceId}";

dbCommand = new OleDbCommand(query, dbconnection);

dbCommand.ExecuteNonQuery();

query =$"UPDATE People SET DeathPlaceId = 0 WHERE DeathPlaceId={selectedItem.PlaceId}";

dbCommand = new OleDbCommand(query, dbconnection);

dbCommand.ExecuteNonQuery();

query =$"UPDATE Events SET EventLocationId = 0 WHERE EventLocationId={selectedItem.PlaceId}";

dbCommand = new OleDbCommand(query, dbconnection);

dbCommand.ExecuteNonQuery();

string folder = System.IO.Path.GetDirectoryName(nameOfDatabaseFile);;

string deleteFile = folder + $"\\{System.IO.Path.GetFileName(folder)}-Resources\\Places\\" + selectedItem.Picture;

if (System.IO.File.Exists(folder + $"\\{System.IO.Path.GetFileName(folder)}-Resources\\Places\\" + selectedItem.Picture))

{try{System.IO.File.Delete(deleteFile);}catch{}}}

public static void ShowPlace(string nameOfDatabaseFile, Places selectedItem)

{string selectedPlaceId = selectedItem.PlaceId.ToString();

Show show\_place =

new Show(nameOfDatabaseFile, selectedPlaceId, "Place");

show\_place.Show();}

public override string ToString()

{return Name;}}}

Класс People.cs:

namespace Family\_Tree

{public class People{

public int PersonId { get; set; }

public string FullName{get{

if (string.IsNullOrEmpty(MiddleName) && string.IsNullOrEmpty(LastName)){

return $"{FirstName}";}

else if (string.IsNullOrEmpty(MiddleName)){

return $"{LastName} {FirstName}";}

else if (string.IsNullOrEmpty(LastName))

{return $"{FirstName} {MiddleName}";}else{

return $"{LastName} {FirstName} {MiddleName}";}}}

public string Gender { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public string MiddleName { get; set; }

public string Age

{get{if (BirthDate == DateTime.MinValue){return "?";}

if(IsAlive == false && DeathDate == DateTime.MinValue)

{return "?";}else{if (IsAlive == false){

DateTime today = DateTime.Today;

int age = DeathDate.Year - BirthDate.Year;

if (today.Month < BirthDate.Month || (today.Month == BirthDate.Month && today.Day < BirthDate.Day)){age--;}

return age.ToString();}else{DateTime today = DateTime.Today;

int age = today.Year - BirthDate.Year;

if (today.Month < BirthDate.Month || (today.Month == BirthDate.Month && today.Day < BirthDate.Day))

{age--;}return age.ToString();}}}}

public bool IsAdult => BirthDate <= DateTime.Today.AddYears(-18);

public bool IsSenior => BirthDate <= DateTime.Today.AddYears(-65);

public string MaidenName { get; set; }

public string Occupation { get; set; }

public int ResidenceId { get; set; }

public string ZodiacSign{get{

int day = BirthDate.Day;

int month = BirthDate.Month;string sign = string.Empty;

DateTime emptyDate = new DateTime(1, 1, 1);

if (BirthDate == emptyDate){sign = "";return sign;}

if ((month == 3 && day >= 21) || (month == 4 && day <= 19)){sign = "Овен";}

else if ((month == 4 && day >= 20) || (month == 5 && day <= 20)){sign = "Телец";}

else if ((month == 5 && day >= 21) || (month == 6 && day <= 20)){sign = "Близнецы";}

else if ((month == 6 && day >= 21) || (month == 7 && day <= 22)){sign = "Рак";}

else if ((month == 7 && day >= 23) || (month == 8 && day <= 22)){sign = "Лев";}

else if ((month == 8 && day >= 23) || (month == 9 && day <= 22)){sign = "Дева";}

else if ((month == 9 && day >= 23) || (month == 10 && day <= 22)){sign = "Весы";}

else if ((month == 10 && day >= 23) || (month == 11 && day <= 21)){sign = "Скорпион";}

else if ((month == 11 && day >= 22) || (month == 12 && day <= 21)){sign = "Стрелец";}

else if ((month == 12 && day >= 22) || (month == 1 && day <= 19)){sign = "Козерог";}

else if ((month == 1 && day >= 20) || (month == 2 && day <= 18)){sign = "Водолей";}

else if ((month == 2 && day >= 19) || (month == 3 && day <= 20)){sign = "Рыбы";}

return sign;}}

public string Religion { get; set; }

public DateTime BirthDate { get; set; }

public int BirthPlaceId { get; set; }

public string Notes { get; set; }

public bool IsAlive { get; set; }

public DateTime DeathDate { get; set; }

public int DeathPlaceId { get; set; }

public string CauseOfDeath { get; set; }

public int FatherId { get; set; }

public int MotherId { get; set; }

public string Photo{get; set;}

public string Role { get; set; }

public List<Participants> relates = new List<Participants>();

public override string ToString()

{string showRole = "", showName = "";

if (!string.IsNullOrEmpty(LastName))

{showName = $"{FirstName} {LastName}";}

else{

showName = $"{FirstName}";}

if (!string.IsNullOrEmpty(Role))

{showRole = $" ({Role})";}

return $"{showName}{showRole}";}

public static List<People> GetPeople(string nameOfDatabaseFile)

{List<People> people = new List<People>();

string connect =

$"Provider=Microsoft.ACE.OLEDB.12.0;Data Source={nameOfDatabaseFile};Jet OLEDB:Engine Type=5";

OleDbConnection dbconnection = new OleDbConnection(connect);

dbconnection.Open();

string query = "SELECT \* FROM People"; // запрос

OleDbCommand dbCommand = new OleDbCommand(query, dbconnection);

OleDbDataReader dbDataReader = dbCommand.ExecuteReader();

while (dbDataReader.Read())

{People person = new People();

person.PersonId = Convert.ToInt32(dbDataReader["PersonId"]);

person.Gender = dbDataReader["Gender"] != DBNull.Value

? dbDataReader["Gender"].ToString(): null;

person.FirstName = dbDataReader["FirstName"] != DBNull.Value

? dbDataReader["FirstName"].ToString(): null;

person.LastName = dbDataReader["LastName"] != DBNull.Value

? dbDataReader["LastName"].ToString(): null;

person.MiddleName = dbDataReader["MiddleName"] != DBNull.Value

? dbDataReader["MiddleName"].ToString(): null;

person.MaidenName = dbDataReader["MaidenName"] != DBNull.Value

? dbDataReader["MaidenName"].ToString(): null;

person.Occupation = dbDataReader["Occupation"] != DBNull.Value

? dbDataReader["Occupation"].ToString(): null;

person.ResidenceId = dbDataReader["ResidenceId"] != DBNull.Value

? Convert.ToInt32(dbDataReader["ResidenceId"]): 0;

person.Religion = dbDataReader["Religion"] != DBNull.Value

? dbDataReader["Religion"].ToString(): null;

if (dbDataReader["BirthDate"] != DBNull.Value)

{person.BirthDate = Convert.ToDateTime(dbDataReader["BirthDate"]).Date;}

person.BirthPlaceId =

dbDataReader["BirthPlaceId"] != DBNull.Value

? Convert.ToInt32(dbDataReader["BirthPlaceId"]): 0;

person.Notes = dbDataReader["Notes"] != DBNull.Value

? dbDataReader["Notes"].ToString(): null;

person.IsAlive = dbDataReader["IsAlive"] != DBNull.Value

? Convert.ToBoolean(dbDataReader["IsAlive"]): false;

if (dbDataReader["DeathDate"] != DBNull.Value){

person.DeathDate = Convert.ToDateTime(dbDataReader["DeathDate"]).Date;}

person.DeathPlaceId =

(dbDataReader["DeathPlaceId"] != DBNull.Value)

? Convert.ToInt32(dbDataReader["DeathPlaceId"]): 0;

person.CauseOfDeath = dbDataReader["CauseOfDeath"] != DBNull.Value

? dbDataReader["CauseOfDeath"].ToString(): null;

person.FatherId = (dbDataReader["FatherId"] != DBNull.Value)

? Convert.ToInt32(dbDataReader["FatherId"]): 0;

person.MotherId = (dbDataReader["MotherId"] != DBNull.Value)

? Convert.ToInt32(dbDataReader["MotherId"]): 0;

person.Photo = dbDataReader["Photo"] != DBNull.Value

? dbDataReader["Photo"].ToString(): null;

people.Add(person);}

dbDataReader.Close();

dbconnection.Close();

return people;}

public static void DeletePerson(string nameOfDatabaseFile, People selectedItem ){

try{

string connect =

$"Provider=Microsoft.ACE.OLEDB.12.0;Data Source={nameOfDatabaseFile};Jet OLEDB:Engine Type=5";

OleDbConnection dbconnection = new OleDbConnection(connect);

dbconnection.Open();

string query =

$"DELETE FROM People WHERE PersonId={selectedItem.PersonId}";

OleDbCommand dbCommand = new OleDbCommand(query, dbconnection);

dbCommand.ExecuteNonQuery();

string WhoAmI =

(selectedItem.Gender == "Женский") ? "MotherId" : "FatherId";

if (WhoAmI == "MotherId"){

query =

$"UPDATE People SET MotherId = 0 WHERE {WhoAmI}={selectedItem.PersonId}";

dbCommand = new OleDbCommand(query, dbconnection);

dbCommand.ExecuteNonQuery();

}

else

{

query =

$"UPDATE People SET FatherId = 0 WHERE {WhoAmI}={selectedItem.PersonId}";

dbCommand = new OleDbCommand(query, dbconnection);

dbCommand.ExecuteNonQuery();}

query = $"DELETE FROM Events WHERE EventId IN (SELECT EventId FROM Participants WHERE EventId IN ( SELECT EventId FROM Participants WHERE PersonId = {selectedItem.PersonId} GROUP BY EventId HAVING COUNT(\*) = 1 ) AND PersonId = {selectedItem.PersonId} AND EventId IN ( SELECT EventId FROM Participants GROUP BY EventId HAVING COUNT(\*) = 1 ));";

dbCommand = new OleDbCommand(query, dbconnection);

dbCommand.ExecuteNonQuery();

query = $"DELETE FROM Events WHERE EventId IN (SELECT EventId FROM Participants WHERE EventId IN ( SELECT Participants.EventId FROM Participants WHERE Participants.EventId IN ( SELECT EventId FROM Participants GROUP BY EventId HAVING COUNT(PersonId) = {selectedItem.PersonId} )) AND PersonId={selectedItem.PersonId});";

dbCommand = new OleDbCommand(query, dbconnection);

dbCommand.ExecuteNonQuery();

query = $"DELETE FROM Events WHERE EventId IN (SELECT EventId FROM Participants WHERE PersonId = {selectedItem.PersonId} AND (Role = 'Новорождённый' OR Role = 'Умерший'));";

dbCommand = new OleDbCommand(query, dbconnection);

dbCommand.ExecuteNonQuery();

query = $"DELETE FROM Participants WHERE EventId IN (SELECT EventId FROM Participants WHERE PersonId = {selectedItem.PersonId} AND (Role = 'Новорождённый' OR Role = 'Умерший'));";

dbCommand = new OleDbCommand(query, dbconnection);

dbCommand.ExecuteNonQuery();

query =$"DELETE FROM Participants Where PersonId = {selectedItem.PersonId}";

dbCommand = new OleDbCommand(query, dbconnection);

dbCommand.ExecuteNonQuery();

dbconnection.Close();

string folder = System.IO.Path.GetDirectoryName(nameOfDatabaseFile);;

string deleteFile = folder + $"\\{System.IO.Path.GetFileName(folder)}-Resources\\People\\" + selectedItem.Photo;

if (System.IO.File.Exists(folder + $"\\{System.IO.Path.GetFileName(folder)}-Resources\\People\\" + selectedItem.Photo)){

try{System.IO.File.Delete(deleteFile);}catch{}}}

catch (Exception er)

{MessageBox.Show(er.Message);}}

public static void ShowPerson(string nameOfDatabaseFile, People selectedItem)

{string selectedPersonId = selectedItem.PersonId.ToString();

Show show\_person =

new Show(nameOfDatabaseFile, selectedPersonId, "Person");

show\_person.ShowDialog();}}}

Класс Participants.cs:

namespace Family\_Tree

{public class Participants{

        public int EventId { get; set; }

        public int PersonId { get; set; }

        public string Role { get; set; } }}

Класс Events.cs:

namespace Family\_Tree

{public class Events

{public int EventId { get; set; }

public string Type { get; set; }

public DateTime? EventDate { get; set; }

public string FormattedEventDate

{get{return EventDate.HasValue ? EventDate.Value.ToString("dd.MM.yyyy") : string.Empty;}}

public int EventLocationId { get; set; }

public string Description { get; set; }

public string Comment { get; set; }

public string EventStatus{

get{if ((EventDate < DateTime.Today) || (Type == "Рождение") || (Type=="Смерть"))

{return "Прошло";}

if (!EventDate.HasValue) // Проверяем, есть ли значение в EventDate

{return "Неизвестно"; //

}else{return "Предстоит";}}}

public string DaysUntilAnniversary

{get{DateTime today = DateTime.Today;

DateTime? eventDate = EventDate; // Присваиваем значение EventDate другой переменной типа DateTime?

if (!eventDate.HasValue) // Проверяем, есть ли значение в eventDate

{return "?"; // Возвращаем пустую строку, если eventDate не имеет значения}

DateTime nextAnniversary = new DateTime(today.Year, eventDate.Value.Month, eventDate.Value.Day);

if (nextAnniversary < today){

nextAnniversary = nextAnniversary.AddYears(1);}

TimeSpan timeUntilAnniversary = nextAnniversary - today;

int daysUntilAnniversary = timeUntilAnniversary.Days;

return $"{daysUntilAnniversary}";

}}public string Picture { get; set; }

public static List<Events> GetEvents(string nameOfDatabaseFile)

{List<Events> events = new List<Events>();

string connect =

$"Provider=Microsoft.ACE.OLEDB.12.0;Data Source={nameOfDatabaseFile};Jet OLEDB:Engine Type=5";

using (OleDbConnection dbConnection = new OleDbConnection(connect))

{dbConnection.Open();

string query = "SELECT \* FROM Events";

using (OleDbCommand dbCommand = new OleDbCommand(query, dbConnection))

{using (OleDbDataReader dbDataReader = dbCommand.ExecuteReader())

{while (dbDataReader.Read())

{Events evt = new Events();

evt.EventId = Convert.ToInt32(dbDataReader["EventId"]);

evt.Type = dbDataReader["Type"] != DBNull.Value

? dbDataReader["Type"].ToString(): null;

if (dbDataReader["EventDate"] != DBNull.Value)

{evt.EventDate =

Convert.ToDateTime(dbDataReader["EventDate"]).Date;}

evt.EventLocationId =(dbDataReader["EventLocationId"] != DBNull.Value)

? Convert.ToInt32(dbDataReader["EventLocationId"]): 0;

evt.Description = dbDataReader["Description"] != DBNull.Value

? dbDataReader["Description"].ToString(): null;

evt.Comment = dbDataReader["Comment"] != DBNull.Value

? dbDataReader["Comment"].ToString(): null;

evt.Picture = dbDataReader["Picture"] != DBNull.Value

? dbDataReader["Picture"].ToString(): null;events.Add(evt);}}}

dbConnection.Close();}return events;}public static void DeleteEvent(string nameOfDatabaseFile, Events selectedItem)

{string connect =

$"Provider=Microsoft.ACE.OLEDB.12.0;Data Source={nameOfDatabaseFile};Jet OLEDB:Engine Type=5";

OleDbConnection dbconnection = new OleDbConnection(connect);

dbconnection.Open();

string query = $"DELETE \* FROM Events WHERE EventId={selectedItem.EventId}";

OleDbCommand dbCommand = new OleDbCommand(query, dbconnection);

dbCommand.ExecuteNonQuery();

query =$"DELETE \* FROM Participants WHERE EventId={selectedItem.EventId}";

dbCommand = new OleDbCommand(query, dbconnection);

dbCommand.ExecuteNonQuery();

string folder = System.IO.Path.GetDirectoryName(nameOfDatabaseFile);;

string deleteFile = folder + $"\\{System.IO.Path.GetFileName(folder)}-Resources\\Events\\" + selectedItem.Picture;

if (System.IO.File.Exists(folder + $"\\{System.IO.Path.GetFileName(folder)}-Resources\\Events\\" + selectedItem.Picture))

{try{System.IO.File.Delete(deleteFile);}catch{}}}

public static void ShowEvent(string nameOfDatabaseFile, Events selectedItem){

string selectedEventId = selectedItem.EventId.ToString();

Show showEvent =

new Show(nameOfDatabaseFile, selectedEventId, "Event");

showEvent.Show();}}}

Класс Warning:

namespace Family\_Tree

{public partial class Warning : Window{

public Warning(){InitializeComponent();}

public static void WarningShow(string text,string title){

Warning warning = new Warning();

warning.warning\_textblock.Text = $"{text}";

warning.Title = $"{title}";

warning.ShowDialog();}

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

{Close();}

public void OKClick(object sender, RoutedEventArgs e)

{DialogResult = true;Close();}}}

Класс TreeEditor:

namespace Family\_Tree{

public partial class TreeEditor : Window{

private List<People> people = new List<People>();

private List<Places> places = new List<Places>();

private List<Events> events = new List<Events>();

private string databaseFilePath;

private readonly string AddOrEdit = "";

private readonly int editPersonId;

private void ResetDataGrid()

{foreach (DataGridColumn column in myDataGrid.Columns){

column.Header = "";column.Visibility = Visibility.Visible;}}

private void SwitchToPeople()

{myDataGrid.ItemsSource =null;myDataGrid.ItemsSource =people;}

private void SwitchToEvents()

{myDataGrid.ItemsSource =null;myDataGrid.ItemsSource =events;}

public TreeEditor(string filePath)

{InitializeComponent();try{databaseFilePath = filePath;

events = Events.GetEvents(databaseFilePath);

places = Places.GetPlaces(databaseFilePath);

people = People.GetPeople(databaseFilePath);}

catch{}}

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

{SwitchToPeople();}

private void Exit(object sender, RoutedEventArgs e)

{Application.Current.Shutdown();}

private void Sort(object sender, RoutedEventArgs e)

{people = People.GetPeople(databaseFilePath);

sortButton2.Tag = new BitmapImage(new Uri("Assets/Images/Icons/noicon.png", UriKind.Relative));

sortButton3.Tag = new BitmapImage(new Uri("Assets/Images/Icons/noicon.png", UriKind.Relative));

if (sortIndicator == 0){

sortButton1.Tag = new BitmapImage(new Uri("Assets/Images/Icons/icon.png", UriKind.Relative));

people.RemoveAll(p => !p.IsAlive);

SwitchToPeople();sortIndicator++;}

else if (sortIndicator == 2){

sortButton.Tag = new BitmapImage(new Uri("Assets/Images/Icons/noicon.png", UriKind.Relative));

SwitchToPeople();sortIndicator = 0;}

else if (sortIndicator1 == 1)

{sortButton.Tag = new BitmapImage(new Uri("Assets/Images/Icons/delete.png", UriKind.Relative));

people.RemoveAll(p => p.IsAlive);

SwitchToPeople();sortIndicator++;}}

int sortIndicator = 0;

private void Add(object sender, RoutedEventArgs e)

{try{if (myDataGrid.ItemsSource == people)

{AddEditPerson addingPerson = new AddEditPerson(databaseFilePath, "Add", 0);

addingPerson.ShowDialog();}

else if (myDataGrid.ItemsSource == places)

{AddEditPlace addingPlace = new AddEditPlace(databaseFilePath, "Add", 0);

addingPlace.ShowDialog();}

else if (myDataGrid.ItemsSource == events)

{AddEditEvent addingEvent = new AddEditEvent(databaseFilePath, "Add", 0);addingEvent.ShowDialog();}}catch { }}

private void Window\_Activated(object sender, EventArgs e)

{try{

if (myDataGrid.ItemsSource == people){people = People.GetPeople(databaseFilePath);

SwitchToPeople();}

else if (myDataGrid.ItemsSource == places){places = Places.GetPlaces(databaseFilePath);SwitchToPlaces();}

else if (myDataGrid.ItemsSource == events){events = Events.GetEvents(databaseFilePath);SwitchToEvents();}}

catch{}}

private void Click\_Show(object sender, RoutedEventArgs e){

ShowSmth();}

private void ShowSmth()

{if (myDataGrid.SelectedItem != null){if (myDataGrid.SelectedItems.Count == 1){

try{if (myDataGrid.ItemsSource == people){

People selectedPerson = myDataGrid.SelectedItem as People;

People.ShowPerson(databaseFilePath, selectedPerson);}

if (myDataGrid.ItemsSource == places){

Places selectedPlace = myDataGrid.SelectedItem as Places;

Places.ShowPlace(databaseFilePath, selectedPlace)}

if (myDataGrid.ItemsSource == events){

Events selectedEvent = myDataGrid.SelectedItem as Events;

Events.ShowEvent(databaseFilePath, selectedEvent);}}

catch{}}else{

Warning.WarningShow("Выберите лишь одну запись для просмотра и попробуйте ещё раз", "Выбрано более одной записи");}}else{

Warning.WarningShow("Выберите запись для просмотра и попробуйте ещё раз!", "Не выбрана ни одна запись");}}

private void Delete(object sender, RoutedEventArgs e)

{if (myDataGrid.SelectedItem != null)

{if (!(myDataGrid.SelectedItems.Count > 1)){try{

Warning warning = new Warning();

warning.NObutton.Visibility = Visibility.Visible;

warning.OKbutton.Content = "Да";

warning.OKbutton.Click -= warning.Button\_Click;

warning.OKbutton.Click += warning.YesButton\_Click;

warning.NObutton.Click += warning.NoButton\_Click;

warning.OKbutton.Width = 100;

warning.NObutton.Width = 100;

Grid.SetColumn(warning.NObutton, 1);

Grid.SetColumnSpan(warning.OKbutton, 1);

if (myDataGrid.ItemsSource == people){

People selectedItem = myDataGrid.SelectedItem as People;

warning.Title = "Подтвердите удаление персоны";

warning.warning\_textblock.Text = "Вы уверены, что хотите удалить данную персону и все события, в которых она ключевое лицо?";

warning.ShowDialog();

if ((bool)warning.DialogResult){

People.DeletePerson(databaseFilePath, selectedItem);

people = People.GetPeople(databaseFilePath);

SwitchToPeople();}return;}

if (myDataGrid.ItemsSource == places)

{Places selectedItem = myDataGrid.SelectedItem as Places;

warning.Title = "Подтвердите удаление места";

warning.warning\_textblock.Text = "Вы уверены, что хотите удалить данное место и ссылки на него во всех событиях и персонах?";

warning.ShowDialog();

if ((bool)warning.DialogResult){

Places.DeletePlace(databaseFilePath, selectedItem);

places = Places.GetPlaces(databaseFilePath);

SwitchToPlaces();}return;}

if (myDataGrid.ItemsSource == events){Events selectedItem = myDataGrid.SelectedItem as Events;if (selectedItem.Type == "Рождение" ||selectedItem.Type == "Смерть"){

Warning.WarningShow("Вы не можете удалить событие рождения или смерти, так как эти события являются базовыми. Удалите персону, с которой они связаны.", "Внимание");return;}

warning.Title = "Подтвердите удаление события";

warning.warning\_textblock.Text = "Вы уверены, что хотите удалить данное событие и все соответствующие связи?";

warning.ShowDialog();

if ((bool)warning.DialogResult)

{Events.DeleteEvent(databaseFilePath, selectedItem);

events = Events.GetEvents(databaseFilePath);

SwitchToEvents();}return;}}catch{}}else{

Warning.WarningShow("Выберите лишь одну запись для удаления и попробуйте ещё раз", "Выбрано более одной записи");

}}else{Warning.WarningShow("Выберите запись для удаления и попробуйте ещё раз!", "Не выбрана ни одна запись");}}

private void MenuCreateTree(object sender, RoutedEventArgs e)

{if (people.Count <= 1){

Warning.WarningShow("Слишком мало персон для построения древа. Пожалуйста, добавьте ещё и попробуйте снова!", "Построение древа не имеет смысла");}

else{

Tree tree = new Tree(databaseFilePath);tree.ShowDialog();return;}}

private void Window\_Closed(object sender, EventArgs e){

Application.Current.Shutdown();}

private void Edit(object sender, RoutedEventArgs e)

{if (myDataGrid.SelectedItem != null){

if (myDataGrid.SelectedItems.Count == 1){

try{if (myDataGrid.ItemsSource == people)

{People selectedPerson = myDataGrid.SelectedItem as People;

AddEditPerson addingPerson = new AddEditPerson(databaseFilePath, "Edit", selectedPerson.PersonId);

addingPerson.ShowDialog();}

else if (myDataGrid.ItemsSource == places){

Places selectedPlace = myDataGrid.SelectedItem as Places;

AddEditPlace addingPlace = new AddEditPlace(databaseFilePath, "Edit", selectedPlace.PlaceId);

addingPlace.ShowDialog();}

else if (myDataGrid.ItemsSource == events)

{Events selectedEvent = myDataGrid.SelectedItem as Events;

AddEditEvent addingEvent = new AddEditEvent(databaseFilePath, "Edit", selectedEvent.EventId);

addingEvent.ShowDialog();}}

catch { }}else{

Warning.WarningShow("Выберите лишь одну запись для изменения и попробуйте ещё раз", "Выбрано более одной записи");}}

else{

Warning.WarningShow("Выберите запись для изменения и попробуйте ещё раз!", "Не выбрана ни одна запись");}}

private void OpenDev(object sender, RoutedEventArgs e)

{string githubUrl = "https://github.com/krabochki";

Process.Start(githubUrl)?.Close();}

private void OpenHelp(object sender, RoutedEventArgs e)

{string gesturefile = Path.Combine(Environment.CurrentDirectory, @"Assets\Family Tree Help.chm");try{

Process.Start(gesturefile);}

catch (Exception ex)

{// Обработка ошибки при открытии файла справки}}}}

namespace Family\_Tree

{public partial class Tree : System.Windows.Window{

string databasefile;

public Tree(string databasefileinput){

InitializeComponent();

this.databasefile = databasefileinput;}

private DotNodeShape Get\_Node\_Shape(ComboBox picker)

{DotNodeShape shape = new DotNodeShape();

if (picker.Text == "Прямоугольник"){

shape = DotNodeShape.Box;}

else if (picker.Text == "Овал"){

shape = DotNodeShape.Oval;}

else if (picker.Text == "Овал"){

shape = DotNodeShape.Oval;}

else if (picker.Text == "Звезда"){

shape = DotNodeShape.Star;}

else if (picker.Text == "Треугольник"){

shape = DotNodeShape.Triangle;}

else if (picker.Text == "Дом")

{shape = DotNodeShape.House;}

else if (picker.Text == "Цилиндр"){

shape = DotNodeShape.Cylinder;}

else if (picker.Text == "Заметка"){

shape = DotNodeShape.Note;}

else if (picker.Text == "Компонент"){

shape = DotNodeShape.Component;}

else if (picker.Text == "Параллелограм"){

shape = DotNodeShape.Parallelogram;}

else if (picker.Text == "Папка"){

shape = DotNodeShape.Folder;}

else if (picker.Text == "Яйцо"){

shape = DotNodeShape.Egg;}return shape;}

private string Get\_Color(Ellipse ellipse){string hexString = "#000000";

if (ellipse.Fill is SolidColorBrush solidColorBrush){

System.Windows.Media.Color color = solidColorBrush.Color;

// Получение шестнадцатеричных значений каналов цвета

string redHex = color.R.ToString("X2");

string greenHex = color.G.ToString("X2");

string blueHex = color.B.ToString("X2");

// Формирование строки в формате HEX

hexString = "#" + redHex + greenHex + blueHex;}

return hexString;}

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

try

{string husbandOrWife = "";

string exhusbandOrexWife = "";

string momOrDad = "";

List<People> branches = new List<People>();

branches = People.GetPeople(databasefile);

foreach(People branch in branches) {

husbandOrWife = (branch.Gender == "Женский пол") ? "Муж" : "Жена";

string whoAmi1 = (branch.Gender == "Женский пол") ? "Жена" : "Муж";

string whoAmi2 =

(branch.Gender == "Женский пол") ? "Бывшая жена" : "Бывший муж";

exhusbandOrexWife =

(branch.Gender == "Женский пол") ? "Бывший муж" : "Бывшая жена";

momOrDad =

(branch.Gender == "Женский пол") ? "FatherId" : "MotherId";

List<People> husbands = new List<People>();

List<People> exes = new List<People>();

List<People> parents = new List<People>();

// Создание графa

var graph = new DotGraph();

graph.Edges.Directions =

GiGraph.Dot.Types.Edges.DotEdgeDirections.None;

graph.EdgeShape = DotEdgeShape.Line;

string rankdir = "";

if ((bool)ascending.IsChecked && (bool)vertical.IsChecked){rankdir = "TB";}

else if ((bool)descending.IsChecked && (bool)vertical.IsChecked){rankdir = "BT";}

else if ((bool)ascending.IsChecked && (bool)horizontal.IsChecked){rankdir = "LR";}

else if ((bool)descending.IsChecked && (bool)horizontal.IsChecked){rankdir = "RL";}

graph.Attributes.Collection.SetCustom("rankdir", rankdir);

graph.Attributes.Collection.SetCustom("bgcolor",

Get\_Color(backgroundcolor));

var relationshipNodes =

new List<string>(); // Словарь для хранения существующих узлов отношений

foreach (var branch in branches){

string birthdate = branch.BirthDate.ToShortDateString();

string deathdate = branch.DeathDate.ToShortDateString();

string date = "";

if ((bool)showyears.IsChecked){

if (birthdate != "01.01.0001"){

date += birthdate;

if (deathdate != "01.01.0001")

{date += " - " + birthdate;}

else if (!branch.IsAlive){date += " - ?";}}

else if (deathdate != "01.01.0001")

{date += "? - " + birthdate;}else{

date = "?";if (!branch.IsAlive){date = "???";}}}

graph.Nodes.Add(branch.PersonId.ToString(),

node => {

node.Font.Size = int.Parse(fontsizetextbox.Text);

node.Label = ((bool)showlastname.IsChecked?$"{branch.FirstName} {branch.LastName}":branch.FirstName);

if ((bool)showyears.IsChecked){

node.Label += $"\n{date}";}

node.Style.FillStyle = DotNodeFillStyle.Normal;

node.Size.Height = 0.5;

if (branch.Gender == "Женский пол"){

node.Shape = Get\_Node\_Shape(womanshape);

node.Attributes.Collection.SetCustom(

"fillcolor", Get\_Color(womanfillcolor));

node.Attributes.Collection.SetCustom(

"color", Get\_Color(womancolor));}

else{

node.Shape = Get\_Node\_Shape(manshape);

node.Attributes.Collection.SetCustom(

"fillcolor", Get\_Color(manfillcolor));

node.Attributes.Collection.SetCustom(

"color", Get\_Color(mancolor));}

string fontname = font.Text;

node.Attributes.Collection.SetCustom(

"fontcolor", Get\_Color(fontcolor));

node.Attributes.Collection.SetCustom(

"fontname", $"{fontname}");

node.BorderWidth = 3;

if (!branch.IsAlive){

node.Attributes.Collection.SetCustom(

"fillcolor", Get\_Color(deathfillcolor));

node.Attributes.Collection.SetCustom(

"color", Get\_Color(deathcolor));}});

if (branch.relates != null && branch.relates.Count > 0){

foreach (var relationship in branch.relates){

if (relationship.PersonId != 0){

string nodeId = "";nodeId =

$"Relationship\_{relationship.PersonId}\_{branch.PersonId}";

if (!relationshipNodes.Contains(nodeId)){

string creatingNodeId =

$"Relationship\_{branch.PersonId}\_{relationship.PersonId}";

// Создаем новый узел отношений и связываем его с человеком

graph.Nodes.Add(creatingNodeId, node => {

node.Label = "";

node.Style.FillStyle = DotNodeFillStyle.Normal;

node.Shape = DotNodeShape.Circle;

node.Size.Width = 0.2;

node.Size.Height = 0.2;

node.Attributes.Collection.SetCustom(

"image",

((relationship.Role == "Бывший")

? "Assets/Images/Icons/broken\_heart.png"

: ((relationship.Role == "Супруг")

? "Assets/Images/Icons/heart.png"

: (relationship.Role == "Родитель")? "Assets/Images/Icons/puzzle.png" : "")));

node.Attributes.Collection.SetCustom(

"color", ((relationship.Role == "Бывший")

? Get\_Color(connectdivorsion)

: ((relationship.Role == "Супруг")

? Get\_Color(connectmarrriage)

: Get\_Color(connectunknown))));

node.Attributes.Collection.SetCustom("style", "bold");

});

relationshipNodes.Add(creatingNodeId);

var edge = new DotEdge(relationship.PersonId.ToString(),

creatingNodeId.ToString());

edge.Attributes.Collection.SetCustom(

"color", ((relationship.Role == "Бывший")

? Get\_Color(connectdivorsion)

: ((relationship.Role == "Супруг")

? Get\_Color(connectmarrriage)

: Get\_Color(connectunknown))));

graph.Edges.Add(edge);

edge = new DotEdge(branch.PersonId.ToString(),

creatingNodeId.ToString());

edge.Attributes.Collection.SetCustom(

"color", ((relationship.Role == "Бывший")

? Get\_Color(connectdivorsion)

: ((relationship.Role == "Супруг")

? Get\_Color(connectmarrriage)

: Get\_Color(connectunknown))));

graph.Edges.Add(edge);}}}}

if (branch.MotherId != 0 && branch.FatherId != 0)

{if (branch.MotherId < branch.FatherId ){

var edge =

new DotEdge($"Relationship\_{branch.MotherId}\_{branch.FatherId}",

branch.PersonId.ToString());

edge.Attributes.Collection.SetCustom(

"color", Get\_Color(connectchildren));

graph.Edges.Add(edge);}else{

var edge =

new DotEdge($"Relationship\_{branch.FatherId}\_{branch.MotherId}",

branch.PersonId.ToString());

edge.Attributes.Collection.SetCustom(

"color", Get\_Color(connectchildren));

graph.Edges.Add(edge);}}

else{

if (branch.MotherId != 0){

var mom = branches.FirstOrDefault(b => b.PersonId == branch.MotherId);

if (mom != null){

var edge =new DotEdge(mom.PersonId.ToString(), branch.PersonId.ToString());

edge.Attributes.Collection.SetCustom(

"color", Get\_Color(connectchildren));

graph.Edges.Add(edge);}}

string folder = System.IO.Path.GetDirectoryName(databasefile);

graph.SaveToFile(folder + $"\\{System.IO.Path.GetFileName(folder)}-Resources\\graph.gv");

// Путь к утилите dot.exe

string dotPath = "Assets/Graphviz/bin/dot.exe";

// Путь к файлу GV

string inputPath = folder + $"\\{System.IO.Path.GetFileName(folder)}-Resources\\graph.gv";

// Путь к выходному PNG-изображению

string outputPath = folder + $"\\{System.IO.Path.GetFileName(folder)}-Resources\\graph.png";

Process process = new Process();

process.StartInfo.FileName = dotPath;

process.StartInfo.Arguments =

$"-Tpng \"{inputPath}\" -o \"{outputPath}\"";

process.StartInfo.UseShellExecute = false;

process.StartInfo.RedirectStandardOutput = true;

process.StartInfo.CreateNoWindow = true;

process.Start();

process.WaitForExit();

ShowTree showTree = new ShowTree(databasefile, this);

showTree.Background = backgroundcolor.Fill;

var bitmap = new BitmapImage();

var stream = System.IO.File.OpenRead(folder + $"\\{System.IO.Path.GetFileName(folder)}-Resources\\graph.png");

bitmap.BeginInit();

bitmap.CacheOption = BitmapCacheOption.OnLoad;

bitmap.StreamSource = stream;

bitmap.EndInit();

stream.Close();

stream.Dispose();

showTree.imageviewer.Source = bitmap;

showTree.Closed += Form\_Closed;

showTree.ShowDialog();}

catch (Exception er){

Warning.WarningShow("Непредвиденная ошибка. Попробуйте ещё раз.", "Ошибка");return;} }}

private void Form\_Closed(object sender, EventArgs e){try{

string folder = System.IO.Path.GetDirectoryName(databasefile);

string filePath = System.IO.Path.Combine(Environment.CurrentDirectory, folder + $"\\{System.IO.Path.GetFileName(folder)}-Resources\\graph.png");

System.IO.File.Delete(filePath);

filePath = System.IO.Path.Combine(Environment.CurrentDirectory, folder + $"\\{System.IO.Path.GetFileName(folder)}-Resources\\graph.gv");

System.IO.File.Delete(filePath);}catch{}}}}}

**Класс Show.cs:**

namespace Family\_Tree{

public partial class Show : Window

{string folder = "";

string ID;

string databaseFilePath;

string address;

bool doesPhoto;

string whattoshow;

public Show(string databasefilepath, string id, string whatToShow)

{InitializeComponent();

folder = System.IO.Path.GetDirectoryName(databasefilepath);

whattoshow = whatToShow;ID = id;

databaseFilePath = databasefilepath;

switch (whatToShow){

case "Person":

Show\_Person();

break;case "Event":Show\_Event();break;case "Place":Show\_Place();break;}}

private void Show\_New\_Show(string whattoshow, int id){try{if (whattoshow == "people"){

int selected\_person\_id = id;

Show show\_person =new Show(databaseFilePath, id.ToString(), "Person");

show\_person.Show();}

if (whattoshow == "places"){

int selected\_place\_id = id;

Show show\_place =

new Show(databaseFilePath, id.ToString(), "Place");

show\_place.Show();}

if (whattoshow == "events"){

int selected\_event\_id = id;

Show show\_event =

new Show(databaseFilePath, id.ToString(), "Event");

show\_event.Show();}}

catch{}}

private void Show\_Person()

{

geo\_button.Width = 0;

geo\_button.Height = 0;

People person = new People();

Title += ": Просмотр персоны";

List<People> husbands = new List<People>();

List<People> exes = new List<People>();

List<People> parentsOfChildren = new List<People>();

string mumName = "",

dadName = "",

birthPlaceName = "",

deathPlaceName = "",

residenceName = "";

List<People> siblings = new List<People>();

List<People> children = new List<People>();

string husbandOrWife = "";

string momOrDad = "";

string exhusbandOrexWife = "";

husbandOrWife = (person.Gender == "Женский пол") ? "Муж" : "Жена";

string whoAmi1 = (person.Gender == "Женский пол") ? "Жена" : "Муж";

string whoAmi2 = (person.Gender == "Женский пол") ? "Бывшая жена" : "Бывший муж";

husbandOrWife = (person.Gender == "Женский пол") ? "Муж" : "Жена";

exhusbandOrexWife = (person.Gender == "Женский пол") ? "Бывший муж" : "Бывшая жена";

momOrDad = (person.Gender == "Женский пол") ? "FatherId" : "MotherId";

fullname\_textblock.Text = person.FullName;

if (!person.IsAlive){

fullname\_textblock.Inlines.Add(new Run { Text = $" †", FontSize = 30, FontFamily = new FontFamily("Arial") });}

string gender = (person.Gender == "Женский пол") ? "Женский" : ((person.Gender == "Мужской пол") ? "Мужской" : "Неизвестно");

TextBlock textBlock = new TextBlock();

textBlock.Inlines.Add(new Run { Text = $"Пол: ", FontSize = 14, FontFamily = new FontFamily("Visby Round CF DemiBold") });

textBlock.Inlines.Add(new Run { Text = $"{gender}", FontSize = 14 });

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 2);

textblock.Inlines.Add(textBlock);

string birthdate = (person.BirthDate != DateTime.MinValue) ? person.BirthDate.ToShortDateString() : "";

string zodiac\_sign = (birthdate != "") ? (" (" + person.ZodiacSign + ")") : "";

string birth\_place = !string.IsNullOrEmpty(birthPlaceName) ? (((person.BirthDate == DateTime.MinValue) ? birthPlaceName : ", " + birthPlaceName)) : "";

if (!(birthdate == "" && birth\_place == "")){

textblock.Inlines.Add(new LineBreak()); // Добавление перехода на новую строку

textBlock = new TextBlock();

textBlock.Inlines.Add(new Run { Text = $"Рождение: ", FontSize = 14, FontFamily = new FontFamily("Visby Round CF DemiBold") });

textBlock.Inlines.Add(new Run { Text = $"{birthdate}{zodiac\_sign}", FontSize = 14 });

if (!string.IsNullOrEmpty(birth\_place)){

Hyperlink hyperlink = new Hyperlink(new Run(birth\_place));

hyperlink.FontSize = 14;

hyperlink.Click += (sender, e) => Show\_New\_Show("places", person.BirthPlaceId);

hyperlink.Foreground = Brushes.White;

hyperlink.Cursor = Cursors.Hand;

hyperlink.TextDecorations = new TextDecorationCollection();

textBlock.Inlines.Add(hyperlink);}

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 2);

textblock.Inlines.Add(textBlock);}

if (!person.IsAlive){

string deathdate = (person.DeathDate != DateTime.MinValue) ? person.DeathDate.ToShortDateString() : "";

string death\_place = !string.IsNullOrEmpty(deathPlaceName) ? (((person.DeathDate == DateTime.MinValue) ? deathPlaceName : ", " + deathPlaceName))"";

string death\_reason = !string.IsNullOrEmpty(person.CauseOfDeath) ? (((string.IsNullOrEmpty(deathPlaceName) && string.IsNullOrEmpty(deathdate)) ? person.CauseOfDeath : (" (" + person.CauseOfDeath.ToLower() + ")"))) : "";

if (!(deathdate == "" && death\_place == "" && death\_reason == ""))

{textBlock = new TextBlock();

textblock.Inlines.Add(new LineBreak()); // Добавление перехода на новую строку

textBlock.Inlines.Add(new Run { Text = $"Смерть: ", FontSize = 14, FontFamily = new FontFamily("Visby Round CF DemiBold") });

textBlock.Inlines.Add(new Run { Text = $"{deathdate}", FontSize = 14 });

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 2);

if (!string.IsNullOrEmpty(deathPlaceName))

{Hyperlink hyperlink = new Hyperlink(new Run(death\_place));

hyperlink.FontSize = 14;

hyperlink.Click += (sender, e) => Show\_New\_Show("places", person.DeathPlaceId);

hyperlink.Foreground = Brushes.White;

hyperlink.Cursor = Cursors.Hand;

hyperlink.TextDecorations = new TextDecorationCollection();

textBlock.Inlines.Add(hyperlink);}

textBlock.Inlines.Add(new Run { Text = $"{death\_reason}", FontSize = 14 });

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 2);

textblock.Inlines.Add(textBlock);}}

string age\_or\_live\_length = !person.IsAlive ? "Продолжительность жизни" : "Возраст";

string age = person.Age.ToString();

textBlock = new TextBlock();

textblock.Inlines.Add(new LineBreak()); // Добавление перехода на новую строку

textBlock.Inlines.Add(new Run { Text = $"{age\_or\_live\_length}: ", FontSize = 14, FontFamily = new FontFamily("Visby Round CF DemiBold") });

textBlock.Inlines.Add(new Run { Text = $"{age}", FontSize = 14 });

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 2);

textblock.Inlines.Add(textBlock);

textblock.Inlines.Add(new LineBreak()); // Добавление перехода на новую строку

string live\_place = residenceName;

if (!string.IsNullOrEmpty(live\_place)){

textBlock = new TextBlock();

textBlock.Inlines.Add(new Run { Text = "Место жительства: ", FontSize = 14, FontFamily = new FontFamily("Visby Round CF DemiBold") });

Hyperlink hyperlink = new Hyperlink(new Run(live\_place));

hyperlink.FontSize = 14;

hyperlink.Click += (sender, e) => Show\_New\_Show("places", person.ResidenceId);

hyperlink.Foreground = Brushes.White;

hyperlink.Cursor = Cursors.Hand;

hyperlink.TextDecorations = new TextDecorationCollection();

textBlock.Inlines.Add(hyperlink);

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 2);

textblock.Inlines.Add(textBlock);

textblock.Inlines.Add(new LineBreak()); // Добавление перехода на новую строку}

string occupation = person.Occupation;

if (!string.IsNullOrEmpty(occupation))

{textBlock = new TextBlock();

textBlock.Inlines.Add(new Run { Text = "Деятельность: ", FontSize = 14, FontFamily = new FontFamily("Visby Round CF DemiBold") });

textBlock.Inlines.Add(new Run { Text = $"{occupation}", FontSize = 14 });

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 2);

textblock.Inlines.Add(textBlock);

textblock.Inlines.Add(new LineBreak()); // Добавление перехода на новую строку}

string religion = person.Religion;

if (!string.IsNullOrEmpty(religion))

{textBlock = new TextBlock();

textBlock.Inlines.Add(new Run { Text = "Религия: ", FontSize = 14, FontFamily = new FontFamily("Visby Round CF DemiBold") });

textBlock.Inlines.Add(new Run { Text = $"{religion}", FontSize = 14 });

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 2);

textblock.Inlines.Add(textBlock);

textblock.Inlines.Add(new LineBreak());}

string note = person.Notes;

if (!string.IsNullOrEmpty(note))

{textBlock = new TextBlock();

textBlock.Inlines.Add(new Run { Text = $"{note}", FontSize = 16, FontStyle = FontStyles.Italic });

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 2);

textblock.Inlines.Add(textBlock);

textblock.Inlines.Add(new LineBreak());}

textblock.Inlines.Add(new LineBreak());

if (person.FatherId != 0)

{textBlock = new TextBlock();

textBlock.Inlines.Add(new Run { Text = "Отец: ", FontSize = 12, FontFamily = new FontFamily("Visby Round CF DemiBold") });

Hyperlink hyperlink = new Hyperlink(new Run(dadName));

hyperlink.FontSize = 12;

hyperlink.Click += (sender, e) => Show\_New\_Show("people", person.FatherId);

hyperlink.Foreground = Brushes.White;

hyperlink.Cursor = Cursors.Hand;

hyperlink.TextDecorations = new TextDecorationCollection();

textBlock.Inlines.Add(hyperlink);

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 1);

textblock.Inlines.Add(textBlock);

textblock.Inlines.Add(new LineBreak());}

if (person.MotherId != 0)

{textBlock = new TextBlock();

textBlock.Inlines.Add(new Run { Text = "Мать: ", FontSize = 12, FontFamily = new FontFamily("Visby Round CF DemiBold") });

Hyperlink hyperlink = new Hyperlink(new Run(mumName));

hyperlink.FontSize = 12;

hyperlink.Click += (sender, e) => Show\_New\_Show("people", person.MotherId);

hyperlink.Foreground = Brushes.White;

hyperlink.Cursor = Cursors.Hand;

hyperlink.TextDecorations = new TextDecorationCollection();

textBlock.Inlines.Add(hyperlink);

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 1);

textblock.Inlines.Add(textBlock);

textblock.Inlines.Add(new LineBreak());}

if (husbands.Count != 0){

foreach (var husband in husbands){

if (husband.PersonId != 0){

textBlock = new TextBlock();

textBlock.Inlines.Add(new Run { Text = $"{husbandOrWife}: ", FontSize = 12, FontFamily = new FontFamily("Visby Round CF DemiBold") });

Hyperlink hyperlink = new Hyperlink(new Run(husband.FirstName + " " + husband.LastName));

hyperlink.FontSize = 12;

hyperlink.Click += (sender, e) => Show\_New\_Show("people", husband.PersonId);

hyperlink.Foreground = Brushes.White;

hyperlink.Cursor = Cursors.Hand;

hyperlink.TextDecorations = new TextDecorationCollection();

textBlock.Inlines.Add(hyperlink);

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 1);

textblock.Inlines.Add(textBlock);

textblock.Inlines.Add(new LineBreak());}}}

if (exes.Count != 0){

foreach (var ex in exes){

if (ex.PersonId != 0){

textBlock = new TextBlock();

textBlock.Inlines.Add(new Run { Text = $"{exhusbandOrexWife}: ", FontSize = 12, FontFamily = new FontFamily("Visby Round CF DemiBold") });

Hyperlink hyperlink = new Hyperlink(new Run(ex.FirstName + " " + ex.LastName));

hyperlink.FontSize = 12;

hyperlink.Click += (sender, e) => Show\_New\_Show("people", ex.PersonId);

hyperlink.Foreground = Brushes.White;

hyperlink.Cursor = Cursors.Hand;

hyperlink.TextDecorations = new TextDecorationCollection();

textBlock.Inlines.Add(hyperlink);

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 1);

textblock.Inlines.Add(textBlock);

textblock.Inlines.Add(new LineBreak());}}}

momOrDad = (momOrDad == "FatherId") ?"Отец ребенка":"Мать ребенка";

if (parentsOfChildren.Count != 0){

foreach (var parent in parentsOfChildren){

if (parent.PersonId != 0){

textBlock = new TextBlock();

textBlock.Inlines.Add(new Run { Text = $"{momOrDad}: ", FontSize = 12, FontFamily = new FontFamily("Visby Round CF DemiBold") });

Hyperlink hyperlink = new Hyperlink(new Run(parent.FirstName + " " + parent.LastName));

hyperlink.FontSize = 12;

hyperlink.Click += (sender, e) => Show\_New\_Show("people", parent.PersonId);

hyperlink.Foreground = Brushes.White;

hyperlink.Cursor = Cursors.Hand;

hyperlink.TextDecorations = new TextDecorationCollection();

textBlock.Inlines.Add(hyperlink);

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 1);

textblock.Inlines.Add(textBlock);

textblock.Inlines.Add(new LineBreak());}}}

if (siblings.Count != 0){

foreach (People sibling in siblings){

string whoAreYou = (sibling.Gender == "Женский пол") ? "Сестра" : (sibling.Gender == "Мужской пол") ? "Брат" : "Неизвестно";

textBlock = new TextBlock();

textBlock.Inlines.Add(new Run { Text = $"{whoAreYou}: ", FontSize = 12, FontFamily = new FontFamily("Visby Round CF DemiBold") });

Hyperlink hyperlink = new Hyperlink(new Run($"{sibling.FirstName} {sibling.LastName}"));

hyperlink.FontSize = 12;

hyperlink.Click += (sender, e) => Show\_New\_Show("people", sibling.PersonId);

hyperlink.Foreground = Brushes.White;

hyperlink.Cursor = Cursors.Hand;

hyperlink.TextDecorations = new TextDecorationCollection();

textBlock.Inlines.Add(hyperlink);

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 1);

textblock.Inlines.Add(textBlock);

textblock.Inlines.Add(new LineBreak()); // Добавление перехода на новую строку}}

if (children.Count != 0){

foreach (People child in children){

string whoAreYou = (child.Gender == "Женский пол") ? "Дочь" : (child.Gender == "Мужской пол") ? "Сын" : "Неизвестно";

textBlock = new TextBlock();

textBlock.Inlines.Add(new Run { Text = $"{whoAreYou}: ", FontSize = 12, FontFamily = new FontFamily("Visby Round CF DemiBold") });

Hyperlink hyperlink = new Hyperlink(new Run($"{child.FirstName} {child.LastName}"));

hyperlink.FontSize = 12;

hyperlink.Click += (sender, e) => Show\_New\_Show("people", child.PersonId);

hyperlink.Foreground = Brushes.White;

hyperlink.Cursor = Cursors.Hand;

hyperlink.TextDecorations = new TextDecorationCollection();

textBlock.Inlines.Add(hyperlink);

textblock.Inlines.Add(textBlock);

textblock.Inlines.Add(new LineBreak()); // Добавление перехода на новую строку}}

doesPhoto = false;

BitmapImage bitmap;

try{BitmapImage photo = new BitmapImage();

var stream = System.IO.File.OpenRead(folder + $"\\{System.IO.Path.GetFileName(folder)}-Resources\\People\\" + person.Photo);

photo.BeginInit();

photo.CacheOption = BitmapCacheOption.OnLoad;

photo.StreamSource = stream;

photo.EndInit();

stream.Close();

stream.Dispose();

personPhoto\_image.Source = photo;

textblock.Margin = new Thickness(30, 20, 25, 0);

doesPhoto = true;}catch{

imageblock.Height = 0;

imageblock.Width = 0;

Grid.SetColumn(textblock, 0);

Grid.SetColumnSpan(textblock, 2);

doesPhoto = false;}}

private void Show\_Place(){

Places place = new Places();

Title += ": Просмотр места";

string abbreviation = place.Abbreviation;

fullname\_textblock.Text = place.Name;

TextBlock textBlock;

if (!string.IsNullOrEmpty(abbreviation))

{

textBlock = new TextBlock();

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 2);

textBlock.Inlines.Add(new Run { Text = "Аббревиатура: ", FontSize = 14, FontFamily = new FontFamily("Visby Round CF DemiBold") });

textBlock.Inlines.Add(new Run { Text = $"{abbreviation}", FontSize = 14 });

textblock.Inlines.Add(textBlock);

textblock.Inlines.Add(new LineBreak()); // Добавление перехода на новую строку}

string historical\_name = place.Historical\_Name;

if (!string.IsNullOrEmpty(historical\_name)){

textBlock = new TextBlock();

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 2);

textBlock.Inlines.Add(new Run { Text = "Историческое название: ", FontSize = 14, FontFamily = new FontFamily("Visby Round CF DemiBold") });

textBlock.Inlines.Add(new Run { Text = $"{historical\_name}", FontSize = 14 });

textblock.Inlines.Add(textBlock);

textblock.Inlines.Add(new LineBreak()); // Добавление перехода на новую строку}

string type = place.Type;

if (!string.IsNullOrEmpty(type)){

textBlock = new TextBlock();

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 2);

textBlock.Inlines.Add(new Run { Text = "Тип: ", FontSize = 14, FontFamily = new FontFamily("Visby Round CF DemiBold") });

textBlock.Inlines.Add(new Run { Text = $"{type}", FontSize = 14 });

textblock.Inlines.Add(textBlock);

textblock.Inlines.Add(new LineBreak()); // Добавление перехода на новую строку}

string description = place.Description;

if (!string.IsNullOrEmpty(description)){

textBlock = new TextBlock();

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 2);

textBlock.Inlines.Add(new Run { Text = "Описание: ", FontSize = 14, FontFamily = new FontFamily("Visby Round CF DemiBold") });

textBlock.Inlines.Add(new Run { Text = $"{description}", FontSize = 14 });

textblock.Inlines.Add(textBlock);

textblock.Inlines.Add(new LineBreak());}

string address = place.Address;

if (!string.IsNullOrEmpty(address)){

this.address = address;

textBlock = new TextBlock();

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 2);

textblock.VerticalAlignment = VerticalAlignment.Center;

textBlock.Inlines.Add(new Run { Text = "Адрес: ", FontSize = 14, FontFamily = new FontFamily("Visby Round CF DemiBold") });

textBlock.Inlines.Add(new Run { Text = $"{address}", FontSize = 14 });

textblock.Inlines.Add(textBlock);

textblock.Inlines.Add(new LineBreak()); // Добавление перехода на новую строку}

else{geo\_button.Height = 0;geo\_button.Width = 0;}

string comment = place.Comment;

if (!string.IsNullOrEmpty(comment))

{comment\_block.Text += "❝ " + comment + " ❞";comment\_block.SizeChanged += (sender, e) =>{

double height = comment\_block.ActualHeight;

// Используйте значение высоты здесь

Height += height + 27;

};}doesPhoto = false;BitmapImage bitmap;try{

BitmapImage photo = new BitmapImage();

var stream = System.IO.File.OpenRead(folder + $"\\{System.IO.Path.GetFileName(folder)}-Resources\\Places\\" + place.Picture);

photo.BeginInit();

photo.CacheOption = BitmapCacheOption.OnLoad;

photo.StreamSource = stream;

photo.EndInit();

stream.Close();

stream.Dispose();

personPhoto\_image.Source = photo;

doesPhoto = true;}catch{

personPhoto\_image.Height = 0;

personPhoto\_image.Width = 0;

Grid.SetColumn(textblock, 0);

Grid.SetColumnSpan(textblock, 2);

doesPhoto = false;}

double heightblock = textblock\_height;

if (doesPhoto == true){

heightblock = 300;

personPhoto\_image.ToolTip = new ToolTip(){

Style = (Style)FindResource("ToolTipStyle"),

Content = "Фото места " + place.Name};

textblock.Margin = new Thickness(30, 20, 25, 0);

Height += 12;}}

double textblock\_height = 0;

private void textblock\_SizeChanged(object sender, SizeChangedEventArgs e){

textblock\_height = textblock.ActualHeight;

if (doesPhoto && (textblock\_height < 200)){

if (whattoshow == "Person"){

textblock\_height = 215;}else{

textblock\_height = 200;}}

if (doesPhoto && (textblock\_height > 200)){

if (whattoshow == "Person"){

textblock\_height += 15;}

else{}}

if ((whattoshow == "Place") && doesPhoto){

Height += 20 + textblock\_height;}

else{

Height += textblock\_height;}

if ((whattoshow == "Event") && doesPhoto){

Height += 25 ;}}

private void Find\_Geo(object sender, RoutedEventArgs e)

{string encodedAddress = Uri.EscapeDataString(address);

string mapsUrl = $"https://www.google.com/maps/search/?api=1&query={encodedAddress}";

Process.Start(mapsUrl);}

private void Show\_Event(){

Events my\_event = new Events();

Places location = new Places();

Title += ": Просмотр события";

List<People> event\_participants = new List<People>();

List<Participants> participants = new List<Participants>();

fullname\_textblock.Text = my\_event.Type;

TextBlock textBlock;

if (my\_event.EventDate.HasValue)

{

textBlock = new TextBlock();

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 2);

textBlock.Inlines.Add(new Run { Text = "Дата события: ", FontSize = 14, FontFamily = new FontFamily("Visby Round CF DemiBold") });

textBlock.Inlines.Add(new Run { Text = $"{my\_event.FormattedEventDate}", FontSize = 14 });

textblock.Inlines.Add(textBlock);

textblock.Inlines.Add(new LineBreak()); // Добавление перехода на новую строку

textBlock = new TextBlock();

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 2);

textBlock.Inlines.Add(new Run { Text = "Статус: ", FontSize = 14, FontFamily = new FontFamily("Visby Round CF DemiBold") });

textBlock.Inlines.Add(new Run { Text = $"{my\_event.EventStatus}", FontSize = 14 });

textblock.Inlines.Add(textBlock);

textblock.Inlines.Add(new LineBreak()); // Добавление перехода на новую строку

textBlock = new TextBlock();

textBlock.Inlines.Add(new Run { Text = "Дней до годовщины: ", FontSize = 14, FontFamily = new FontFamily("Visby Round CF DemiBold") });

textBlock.Inlines.Add(new Run { Text = $"{my\_event.DaysUntilAnniversary}", FontSize = 14 });

textblock.Inlines.Add(textBlock);

textblock.Inlines.Add(new LineBreak()); // Добавление перехода на новую строку}

if (!string.IsNullOrEmpty(location.Name)){

textBlock = new TextBlock();

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 2);

textBlock.Inlines.Add(new Run { Text = "Место события: ", FontSize = 14, FontFamily = new FontFamily("Visby Round CF DemiBold") });

// Создание гиперссылки для названия локации

Hyperlink hyperlink = new Hyperlink(new Run(location.Name));

hyperlink.FontSize = 14;

hyperlink.Click += (sender, e) => Show\_New\_Show("places", location.PlaceId);

hyperlink.Foreground = Brushes.White;

hyperlink.Cursor = Cursors.Hand;

hyperlink.TextDecorations = new TextDecorationCollection();

textBlock.Inlines.Add(hyperlink);

textblock.Inlines.Add(textBlock);

textblock.Inlines.Add(new LineBreak()); // Добавление перехода на новую строку}

string description = my\_event.Description;

if (!string.IsNullOrEmpty(description)){

textBlock = new TextBlock();

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 2);

textBlock.Inlines.Add(new Run { Text = "Описание: ", FontSize = 14, FontFamily = new FontFamily("Visby Round CF DemiBold") });

textBlock.Inlines.Add(new Run { Text = $"{description}", FontSize = 14 });

textblock.Inlines.Add(textBlock);

textblock.Inlines.Add(new LineBreak());}

geo\_button.Height = 0;

geo\_button.Width = 0;

doesPhoto = false;

BitmapImage bitmap;try{

BitmapImage photo = new BitmapImage();

var stream = System.IO.File.OpenRead(folder + $"\\{System.IO.Path.GetFileName(folder)}-Resources\\Events\\" + my\_event.Picture);

photo.BeginInit();

photo.CacheOption = BitmapCacheOption.OnLoad;

photo.StreamSource = stream;

photo.EndInit();

stream.Close();

stream.Dispose();

personPhoto\_image.Source = photo;

textblock.Margin = new Thickness(30, 20, 25, 0);

doesPhoto = true;}

catch{

imageblock.Height = 0;

imageblock.Width = 0;

Grid.SetColumn(textblock, 0);

Grid.SetColumnSpan(textblock, 2);

doesPhoto = false;}

string comment = my\_event.Comment;

textblock.Inlines.Add(new LineBreak());

if (event\_participants.Count != 0){

foreach (People person in event\_participants){

int index = event\_participants.IndexOf(person);

string whoAreYou = participants[index].Role;

textBlock = new TextBlock();

textBlock.Inlines.Add(new Run { Text = $"{whoAreYou}: ", FontSize = 12, FontFamily = new FontFamily("Visby Round CF DemiBold") });

Hyperlink hyperlink = new Hyperlink(new Run($"{person.FirstName} {person.LastName}"));

hyperlink.FontSize = 12;

hyperlink.Click += (sender, e) => Show\_New\_Show("people", participants[index].PersonId);

hyperlink.Foreground = Brushes.White;

hyperlink.Cursor = Cursors.Hand;

hyperlink.TextDecorations = new TextDecorationCollection();

textBlock.Inlines.Add(hyperlink);

textBlock.TextAlignment = TextAlignment.Left;

textBlock.Margin = new Thickness(0, 0, 0, 1);

textblock.Inlines.Add(textBlock);

textblock.Inlines.Add(new LineBreak()); // Добавление перехода на новую строку}}

if (!string.IsNullOrEmpty(comment)){

comment\_block.Text += "❝ " + comment + " ❞";

comment\_block.SizeChanged += (sender, e) =>{

double height = comment\_block.ActualHeight;

// Используйте значение высоты здесь

Height += height + 20;}}}

Класс MainWindow.cs:

namespace Family\_Tree{

public partial class MainWindow : Window{

public MainWindow(){

InitializeComponent();}

public void CreateNewFamilyTree(object sender, RoutedEventArgs e){

bool x= createTree();

if(x) Hide();}

public static bool createTree(){

FolderBrowserDialog FBD = new FolderBrowserDialog();

FBD.Description = "Выберите или создайте папку для проекта семейного древа внутри неё. Назовите её так, как хотите, чтобы назывался файл с базой данных и ресурсами.";

DialogResult res = FBD.ShowDialog();

if (res == System.Windows.Forms.DialogResult.OK){try{

string dn = System.IO.Path.GetFileName(FBD.SelectedPath);

string selectedFilePath = FBD.SelectedPath + $"/{dn}.accdb";

Directory.CreateDirectory(FBD.SelectedPath + $"/{dn}-Resources");

Directory.CreateDirectory(FBD.SelectedPath + $"/{dn}-Resources/People");

Directory.CreateDirectory(FBD.SelectedPath + $"/{dn}-Resources/Places");

Directory.CreateDirectory(FBD.SelectedPath + $"/{dn}-Resources/Events");

// Проверка расширения файлаif (File.Exists(selectedFilePath)){

Warning.WarningShow("В этой папке проект уже существует. Очистите папку или выберите новую","Ошибка");return false;}

Catalog catalog = new Catalog();

// Создание строки соединения с базой данных

string connectionString =

$"Provider=Microsoft.ACE.OLEDB.12.0;Data Source={selectedFilePath};Jet OLEDB:Engine Type=5";

catalog.Create(connectionString);

using (OleDbConnection connection =

new OleDbConnection(connectionString)){

connection.Open();

// Создание таблицы "Персоны"

string createPeopleTableQuery =

"CREATE TABLE People ( PersonId INT PRIMARY KEY, Gender TEXT, FirstName TEXT, LastName TEXT, MiddleName TEXT, MaidenName TEXT, Occupation TEXT, ResidenceId INT, Religion TEXT, BirthDate DATE, BirthPlaceId INT, Notes TEXT, IsAlive YesNo, DeathDate DATE, DeathPlaceId INT, CauseOfDeath TEXT, FatherId INT, MotherId INT, Photo TEXT)";

using (OleDbCommand command =

new OleDbCommand(createPeopleTableQuery, connection)){

command.ExecuteNonQuery();}

// Создание таблицы "Места"

string createPlacesTableQuery =

"CREATE TABLE Places ( PlaceId INT PRIMARY KEY, Name TEXT, Abbreviation TEXT, Historical\_name TEXT, Type TEXT, Description TEXT, Address TEXT, Comment TEXT, Picture TEXT);"; using (OleDbCommand command =

new OleDbCommand(createPlacesTableQuery, connection)){command.ExecuteNonQuery();}

// Создание таблицы "События"

string createEventsTableQuery =

"CREATE TABLE Events ( EventId INT PRIMARY KEY, Type TEXT, EventDate DATE, EventLocationId INT, Description TEXT, Comment TEXT, Picture TEXT)";

using (OleDbCommand command =

new OleDbCommand(createEventsTableQuery, connection)){

command.ExecuteNonQuery();}

string createParticipantsTableQuery =

"CREATE TABLE Participants (EventId INT, PersonId INT, Role TEXT);";

using (OleDbCommand command = new OleDbCommand(

createParticipantsTableQuery, connection)){

command.ExecuteNonQuery();}

connection.Close();}

// Создание базы данных

// Закрытие соединения с базой данных

catalog.ActiveConnection = null;

TreeEditor treeEditor = new TreeEditor(selectedFilePath);

treeEditor.Show(); // или otherForm.ShowDialog();

return true;}catch{

Warning.WarningShow("Непредвиденная ошибка. Попробуйте ещё раз.","Ошибка");

return false;}}

else{

return false;}}

public static bool chooseTree(){

try{

var dialog = new Microsoft.Win32.OpenFileDialog{

DefaultExt = ".accdb", // Расширение файла по умолчанию

Filter ="Файлы базы данных (\*.accdb)|\*.accdb" // Фильтр для отображения

// только файлов с

// расширением .accdb};

bool? result = dialog.ShowDialog();

// Проверка результата диалога

if (result == true){

string selectedFilePath = dialog.FileName;

bool isValidFile = false;

using (var connection = new OleDbConnection(

$"Provider=Microsoft.ACE.OLEDB.12.0;Data Source={selectedFilePath};")){try{

connection.Open();

// Проверяем наличие таблиц "Events", "People", "Places" и

// "Participants"

using (var schema = connection.GetSchema("Tables")){

var tableNames =

schema.AsEnumerable()

.Select(row => row["TABLE\_NAME"].ToString())

.ToList();

if (tableNames.Contains("Events") &&

tableNames.Contains("People") &&

tableNames.Contains("Places") &&

tableNames.Contains("Participants")){

isValidFile = true;}}}

catch{}}

if (isValidFile){

string selectedFullFolder = System.IO.Path.GetDirectoryName(dialog.FileName);

string selectedFolderName = System.IO.Path.GetFileName(selectedFullFolder);

TreeEditor treeEditor = new TreeEditor(selectedFilePath);

treeEditor.Show(); }

else{

Warning.WarningShow(

"Выбранный файл базы данных не является файлом базы данных генеалогического древа нашей программы.",

"Ошибка");return false;}}

return (bool)result;}

catch{Warning.WarningShow(

"Непредвиденная ошибка. Проверьте корректность файла.", "Ошибка");

return false;}}

}

Класс AddEditPlace.cs:

namespace Family\_Tree{

public partial class AddEditPlace : Window{

private string imageFilePath;

private string databaseFilePath;

private readonly string AddOrEdit = "";

private readonly string folder = "";

private readonly int editPlaceId;

private readonly List<Places> places = new List<Places>();

public AddEditPlace(string filePath, string AddOrEdit, int editPlaceId){

InitializeComponent();

folder = System.IO.Path.GetDirectoryName(filePath);

this.AddOrEdit = AddOrEdit;

if (AddOrEdit == "Edit"){

this.editPlaceId = editPlaceId;

Title = "Family Tree: Изменение места";}

databaseFilePath = filePath;

places = Places.GetPlaces(databaseFilePath);

if (AddOrEdit == "Edit"){

Places place = places.Find(p => p.PlaceId == editPlaceId);

name\_textbox.Text = place.Name;

abbr\_textbox.Text = place.Abbreviation;

historicalname\_textbox.Text = place.Historical\_Name;

typeTextBox.Text = place.Type;

addressTextBox.Text = place.Address;

descrTextBox.Text = place.Description;

placeNotesTextBox.Text = place.Comment;

try{

BitmapImage photo = new BitmapImage();

var stream = System.IO.File.OpenRead(folder + $"\\{System.IO.Path.GetFileName(folder)}-Resources\\Places\\" + place.Picture);

photo.BeginInit();

photo.CacheOption = BitmapCacheOption.OnLoad;

photo.StreamSource = stream;

photo.EndInit();

stream.Close();

stream.Dispose();

place\_Photo.Source = photo;

imageFilePath = place.Picture;}catch{}}}

private void LoadPhoto(object sender, RoutedEventArgs e)

{OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter =

"Файлы изображений (\*.jpg, \*.png)|\*.jpg;\*.png|Все файлы (\*.\*)|\*.\*";

try{if (openFileDialog.ShowDialog() == true){

imageFilePath = openFileDialog.FileName;

BitmapImage photo = new BitmapImage();

var stream = System.IO.File.OpenRead(imageFilePath);

photo.BeginInit();

photo.CacheOption = BitmapCacheOption.OnLoad;

photo.StreamSource = stream;photo.EndInit();

stream.Close();stream.Dispose();

place\_Photo.Source = photo;}}catch{

Warning.WarningShow("Возникла непредвиденная ошибка загрузки фотографии. Пожалуйста, попробуйте ещё раз!", "Ошибка загрузки файла");}}

private void DeletePhoto(object sender, RoutedEventArgs e)

{imageFilePath = "Assets/Images/place.png";}

private void AddPerson(object sender, RoutedEventArgs e)

{try{if (string.IsNullOrEmpty(name\_textbox.Text)){

if (AddOrEdit == "Add"){Warning.WarningShow(

"Новое место должно иметь по крайней мере название для узнавания! Пожалуйста, добавьте его.","Обратите внимание");}else{

Warning.WarningShow(

"Оставьте изменяемому месту по крайней мере название для узнавания.",

"Обратите внимание");

}return;}

else

{

Places place = places.Find(p => p.PlaceId == editPlaceId);

string sourceFilePath = imageFilePath;

string destinationFilePath = folder + $"\\{System.IO.Path.GetFileName(folder)}-Resources\\Places\\" + System.IO.Path.GetFileName(sourceFilePath);;

if ((AddOrEdit == "Edit" && (place.Picture != imageFilePath)) || AddOrEdit == "Add"){

string query =

$"INSERT INTO Places VALUES({id},'{name}','{abbr}','{hist\_name}','{type}','{descr}','{address}','{comment}','{image}')";

OleDbCommand dbCommand = new OleDbCommand(query, dbconnection);

dbCommand.ExecuteNonQuery();

}

else

{

string query =

$"UPDATE Places SET Name ='{name}', Abbreviation = '{abbr}', Historical\_name = '{hist\_name}', Type = '{type}', Description = '{descr}', Address = '{address}', Comment = '{comment}', Picture = '{image}' WHERE PlaceId = {editPlaceId}";

OleDbCommand dbCommand0 = new OleDbCommand(query, dbconnection);

dbCommand0.ExecuteNonQuery();

place = places.Find(p => p.PlaceId == editPlaceId);

if ((AddOrEdit == "Edit" && (place.Picture != imageFilePath)) || AddOrEdit == "Add"){

string deleteFile = folder + $"\\{System.IO.Path.GetFileName(folder)}-Resources\\Places\\" + place.Picture}}}

dbconnection.Close(); Close();}}catch{}}}}

Класс AddEditPerson.cs:

namespace Family\_Tree{

public partial class AddEditPerson : Window{

private List<People> people = new List<People>();

private string imageFilePath;

private readonly string databaseFilePath;

private People selectedMother = new People();

private People selectedFather = new People();

private Places birthPlace = new Places();

private Places deathPlace = new Places();

private Places residencePlace = new Places();

private bool isInitialized = false;

private readonly string AddOrEdit = "";

private readonly int editPersonId;

private readonly string folder;

private readonly List<People> manList = new List<People>();

private readonly List<People> womanList = new List<People>();

public AddEditPerson(string filePath, string AddOrEdit, int editPersonId)

{InitializeComponent();

folder = System.IO.Path.GetDirectoryName(filePath);

databaseFilePath = filePath;

this.AddOrEdit = AddOrEdit;

if (AddOrEdit == "Edit"){

this.editPersonId = editPersonId;

Title = "Family Tree: Изменение персоны";}

List<Places> placeList = new List<Places>();

people = People.GetPeople(databaseFilePath);

if (placeList.Count != 1)

{

placeOfBirthComboBox.ItemsSource = null;

placeOfBirthComboBox.ItemsSource = placeList;

comboBoxPlaceDeath.ItemsSource = null;

comboBoxPlaceDeath.ItemsSource = placeList;

residenceComboBox.ItemsSource = null;

residenceComboBox.ItemsSource = placeList;}else{

comboBoxPlaceDeath.IsEnabled = false;

comboBoxPlaceDeath.Opacity = 0.5;

residenceComboBox.IsEnabled = false;

residenceComboBox.Opacity = 0.5;

placeOfBirthComboBox.IsEnabled = false;

placeOfBirthComboBox.Opacity = 0.5;}}}

dbConnection.Close();if (AddOrEdit == "Edit"){

People person = people.Find(p => p.PersonId == editPersonId);

genderComboBox.Text = person.Gender;

firstNameTextBox.Text = person.FirstName;

lastNameTextBox.Text = person.LastName;

middleNameTextBox.Text = person.MiddleName;

maidenNameTextBox.Text = person.MaidenName;

occupationTextBox.Text = person.Occupation;

residenceComboBox.SelectedItem =

placeList.Find(p => p.PlaceId == person.ResidenceId);

religionTextBox.Text = person.Religion;

DateTime emptyDate = new DateTime(1, 1, 1);

if (person.BirthDate != emptyDate){

dateOfBirthPicker.SelectedDate = person.BirthDate;}

placeOfBirthComboBox.SelectedItem =

placeList.Find(p => p.PlaceId == person.BirthPlaceId);

notesTextBox.Text = person.Notes;

aliveCheckBox.IsChecked = person.IsAlive;

if (person.DeathDate != emptyDate){

dateOfDeathPicker.SelectedDate = person.DeathDate;}

causedeathTextBox.Text = person.CauseOfDeath;

comboBoxPlaceDeath.SelectedItem =

placeList.Find(p => p.PlaceId == person.DeathPlaceId);try{

BitmapImage photo = new BitmapImage();

var stream = System.IO.File.OpenRead(folder + $"\\{System.IO.Path.GetFileName(folder)}-Resources\\People\\" + person.Photo);

photo.BeginInit();

photo.CacheOption = BitmapCacheOption.OnLoad;

photo.StreamSource = stream;

photo.EndInit();stream.Close();stream.Dispose();

personPhoto.Source = photo;

imageFilePath = person.Photo;}

catch{}

fatherComboBox.SelectedItem =

manList.Find(p => p.PersonId == person.FatherId);

motherComboBox.SelectedItem =

womanList.Find(p => p.PersonId == person.MotherId);

if(person.MotherId!=0 || person.FatherId != 0){

parentsCheckBox.IsChecked = true;}}}}

private void LoadPhoto(object sender, RoutedEventArgs e){

OpenFileDialog openFileDialog = new OpenFileDialog{

Filter ="Файлы изображений (\*.jpg, \*.png)|\*.jpg;\*.png|Все файлы (\*.\*)|\*.\*"};

try{if (openFileDialog.ShowDialog() == true){

imageFilePath = openFileDialog.FileName;

// Загрузка выбранного файла в Source картинки

BitmapImage photo = new BitmapImage();

var stream = System.IO.File.OpenRead(imageFilePath);

photo.BeginInit();

photo.CacheOption = BitmapCacheOption.OnLoad;

photo.StreamSource = stream;

query =

$"INSERT INTO People VALUES({person\_id},'{gender}','{firstname}','{lastname}','{middlename}','{maidenname}','{occupation}',{residenceID},'{religion}',{birthdate},{birthID},'{notes}',{isAlive},{deathdate},{deathID},'{deathcause}',{fatherID},{motherID},'{photo}')"; //запрос

OleDbCommand dbCommand = new OleDbCommand(query, dbconnection);

dbCommand.ExecuteNonQuery();

// Новое событие "Рождение"

maxIdQuery = "SELECT MAX(EventId) FROM Events";

dbCommand0 = new OleDbCommand(maxIdQuery, dbconnection);

result = dbCommand0.ExecuteScalar();

if (result != null && result != DBNull.Value)

{

lastPersonId = Convert.ToInt32(result);

}

lastPersonId++;

string howToName = (gender == "Женский пол") ? "родилась" : "родился";

string eventid = lastPersonId.ToString();

string eventtype = "Рождение";

string eventdate = birthdate;

string eventlocation = birthID;

string description =

$"Рождение - это особенное и долгожданное событие, когда новая жизнь появляется в мире. В этот момент {firstname} {howToName}, наполнив радостью сердца своих родителей и близких. Этот неповторимый момент приветствует нового члена семьи и запечатлевает начало удивительного жизненного пути.";

string comment = "";

query =

$"INSERT INTO Events VALUES({eventid},'{eventtype}',{eventdate},'{eventlocation}','{description}','{comment}','')"; //запрос

dbCommand = new OleDbCommand(query, dbconnection);

dbCommand.ExecuteNonQuery();

query =

$"INSERT INTO Participants VALUES({eventid},{person\_id},'Новорождённый')"; //запрос

dbCommand = new OleDbCommand(query, dbconnection);

dbCommand.ExecuteNonQuery();

// Новое событие "Смерть"

howToName = (gender == "Женский пол") ? "ушла" : "ушёл";

if (isAlive == "0")

{

eventid = (int.Parse(eventid) + 1).ToString();

eventtype = "Смерть";

eventdate = deathdate;

eventlocation = deathID;

description =

$"Смерть - это глубоко печальное и неизбежное событие, когда жизнь человека прекращается. В этот момент {firstname} {howToName} из этого мира, оставив тоску и горечь в сердцах своих близких. Этот непростой момент открывает новую главу, напоминая нам о хрупкости и ценности человеческой жизни.";

query =

$"INSERT INTO Events VALUES({eventid},'{eventtype}',{eventdate},'{eventlocation}','{description}','{comment}','')"; //запрос

dbCommand = new OleDbCommand(query, dbconnection);

dbCommand.ExecuteNonQuery();

query =

$"INSERT INTO Participants VALUES({eventid},{person\_id},'Умерший')"; //запрос

dbCommand = new OleDbCommand(query, dbconnection);

dbCommand.ExecuteNonQuery();

}

photo.EndInit();stream.Close();stream.Dispose();

personPhoto.Source = photo;}}

catch{Warning.WarningShow(

"Возникла непредвиденная ошибка загрузки фотографии. Пожалуйста, попробуйте ещё раз!",

"Ошибка загрузки файла");}}

private void DeletePhoto(object sender, RoutedEventArgs e)

{imageFilePath = "";

GenderChangePhoto();}

private void ChangeGender(object sender, SelectionChangedEventArgs e){

if (isInitialized){GenderChangePhoto();}

else{isInitialized = true;}}

photo = System.IO.Path.GetFileName(destinationFilePath);

}Close();}catch{}}}

Класс AddEditEvents.cs:

namespace Family\_Tree{

public partial class AddEditEvent : Window{

private string databaseFilePath;

private readonly string folder="";

List<People> selectedPeople = new List<People>();

List<Events> events = new List<Events>();

Places selectedPlace = new Places();

People selectedPerson = new People();

string imageFilePath;

private readonly string AddOrEdit = "";

private readonly int editEventId;

public AddEditEvent(string filePath, string AddOrEdit, int editEventId){

InitializeComponent();

databaseFilePath = filePath;

folder = System.IO.Path.GetDirectoryName(filePath);

this.AddOrEdit = AddOrEdit;

if (AddOrEdit == "Edit"){

this.editEventId = editEventId;

Title = "Family Tree: Изменение места";}

events = Events.GetEvents(databaseFilePath);

roleTextBox.IsEnabled = false;

participantsComboBox.IsEnabled = false;

buttonDeleteParticipant.IsEnabled = false;

buttonAddParticipant.IsEnabled = false;

roleTextBox.Opacity = 0.5;

participantsComboBox.Opacity = 0.5;

buttonAddParticipant.Opacity = 0.5;

buttonDeleteParticipant.Opacity = 0.5;

textBox.Opacity = 0.5;

string connect = $"Provider=Microsoft.ACE.OLEDB.12.0;Data Source={databaseFilePath};Jet OLEDB:Engine Type=5";

using (OleDbConnection dbConnection = new OleDbConnection(connect)){

dbConnection.Open();

List<Places> placeList = new List<Places>();

List<Participants> partList = new List<Participants>();

Events event\_ = events.Find(p => p.EventId == editEventId);

List<People> peopleList = new List<People>();

// Выполнение SQL-запроса на выборку данных

using (OleDbCommand command = new OleDbCommand()){

command.Connection = dbConnection;

// Запрос и чтение данных для списка людей

string sqlQuery = "SELECT PersonId, FirstName, LastName, Gender FROM People";

command.CommandText = sqlQuery;

using (OleDbDataReader reader = command.ExecuteReader()){

People emptyPerson = new People();

peopleList.Add(emptyPerson);

// Чтение результатов запроса и добавление их в список

while (reader.Read()){

int id = reader.GetInt32(0);

string firstName = reader.IsDBNull(1) ? string.Empty : reader.GetString(1);

string lastName = reader.IsDBNull(2) ? string.Empty : reader.GetString(2);

string gender = reader.IsDBNull(3) ? string.Empty : reader.GetString(3);

People person = new People{

PersonId = id,

FirstName = firstName,

Gender = gender,

LastName = lastName};

peopleList.Add(person);}

participantsComboBox.ItemsSource = peopleList;

participantsComboBox.DisplayMemberPath = "FullName";

if (peopleList.Count == 1){

participantsComboBox.IsEnabled = false;

participantsComboBox.Opacity = 0.5;

participantsComboBox.ItemsSource = null;}}

// Запрос и чтение данных для списка мест

sqlQuery = "SELECT PlaceId, Name, Abbreviation FROM Places";

command.CommandText = sqlQuery;

using (OleDbDataReader reader = command.ExecuteReader()){

Places emptyPlace = new Places();

placeList.Add(emptyPlace);

placelist = places.GetPlaces(databaseFilePath);

placesComboBox.ItemsSource = placeList;

if (placeList.Count == 1){

placesComboBox.IsEnabled = false;

placesComboBox.Opacity = 0.5;

placesComboBox.ItemsSource = null;}}

if (AddOrEdit == "Edit"){

sqlQuery = $"SELECT PersonId, Role FROM Participants WHERE EventId = {event\_.EventId}";

command.CommandText = sqlQuery;

using (OleDbDataReader reader2 = command.ExecuteReader()){

using (reader2){

while (reader2.Read()){

int personId = reader2.GetInt32(0);

string role = reader2.IsDBNull(1) ? string.Empty : reader2.GetString(1);

Participants part = new Participants{

PersonId = personId,

Role = role};

partList.Add(part);}

foreach (var part in partList){

foreach (var pipl in peopleList){

if (pipl.PersonId == part.PersonId){

People selected = new People(){

PersonId = part.PersonId,

FirstName = pipl.FirstName,

LastName = pipl.LastName,

Role = part.Role};

selectedPeople.Add(selected);}}}}}}}

dbConnection.Close();

if (AddOrEdit == "Edit"){

UpdateParticipantsTextBox();

typeComboBox.Text = event\_.Type;

roleTextBox.IsEnabled = true;

participantsComboBox.IsEnabled = true;

buttonDeleteParticipant.IsEnabled = true;

buttonAddParticipant.IsEnabled = true;

roleTextBox.Opacity = 1;

participantsComboBox.Opacity = 1;

buttonAddParticipant.Opacity = 1;

buttonDeleteParticipant.Opacity = 1;

textBox.Opacity = 1;

if (event\_.Type == "Рождение" || event\_.Type == "Смерть"){

typeComboBox.Opacity = 0.5;

typeComboBox.IsEnabled = false;

Warning warning = new Warning();

warning.Title = "Внимание!";

warning.Height += 50;

warning.Width += 50;

warning.warning\_textblock.Height += 30;

warning.warning\_textblock.Width += 35;

warning.OKbutton.Content = "Понятно";

datePickerOfEvent.Opacity = 0.5;

datePickerOfEvent.IsEnabled = false;

placesComboBox.Opacity = 0.5;

placesComboBox.IsEnabled = false;

if (event\_.Type == "Рождение"){

warning.warning\_textblock.Text = "Чтобы изменить дату и место события рождения, измените соответствующие поля у родившейся персоны. Удалить или поменять главного участника нельзя, так как это событие является обязательным для каждой персоны!";

People borningPerson = selectedPeople.Find(person => person.Role == "Новорождённый");

peopleList.RemoveAll(person => person.PersonId == borningPerson.PersonId);

participantsComboBox.ItemsSource = peopleList;}else{

warning.warning\_textblock.Text = "Чтобы изменить дату и место события смерти, измените соответствующие поля у умершей персоны. Удалить или поменять главного участника нельзя, так как это событие является обязательным для умерших персон!";

People deceasedPerson = selectedPeople.Find(person => person.Role == "Умерший");

peopleList.RemoveAll(person => person.PersonId == deceasedPerson.PersonId);

participantsComboBox.ItemsSource = peopleList;}

warning.ShowDialog();

UpdateRoles();}

DateTime emptyDate = new DateTime(1, 1, 1);

if (event\_.EventDate != emptyDate){

datePickerOfEvent.SelectedDate = event\_.EventDate;}

placesComboBox.SelectedItem =

placeList.Find(p => p.PlaceId == event\_.EventLocationId);

notesTextBox.Text = event\_.Comment;

descriptiontextbox.Text = event\_.Description;

if (event\_.Picture != ""){try{

imageFilePath = event\_.Picture;

BitmapImage photo = new BitmapImage();

var stream = System.IO.File.OpenRead(folder + $"\\{System.IO.Path.GetFileName(folder)}-Resources\\Events\\" + event\_.Picture);

photo.BeginInit();

photo.CacheOption = BitmapCacheOption.OnLoad;

photo.StreamSource = stream;

photo.EndInit();stream.Close();stream.Dispose();

eventPhoto.Source = photo;}

catch{}}}}}

private void DeleteParticipantFromList(object sender, RoutedEventArgs e)

{if (participantsComboBox.SelectedItem != null){

People deletablePerson = participantsComboBox.SelectedItem as People;

int personid = deletablePerson.PersonId;

foreach(var select in selectedPeople)

{if (select.PersonId == personid){

selectedPeople.Remove(select);}}

roleTextBox.SelectedItem = null;

participantsComboBox.SelectedItem = null;

UpdateParticipantsTextBox();}}

private void AddParticipantToList(object sender, RoutedEventArgs e){

string itsrole = roleTextBox.Text;

foreach (var selected in selectedPeople)

{if (((itsrole == "Муж" || itsrole == "Жена" || itsrole == "Невеста"

|| itsrole == "Жених" || itsrole == "Бывший муж" || itsrole == "Бывшая жена" || itsrole == "Возлюбленный" || itsrole == "Возлюбленная" || itsrole == "Бывший возлюбленный"

|| itsrole == "Бывшая возлюбленная") )&& itsrole == selected.Role){return;}}

People selectedPerson = participantsComboBox.SelectedItem as People;

selectedPerson.Role = roleTextBox.Text;

if (selectedPerson != null && !string.IsNullOrEmpty(selectedPerson.Role))

{if (!selectedPeople.Contains(selectedPerson)){

selectedPeople.Add(selectedPerson);}

participantsComboBox.SelectedItem = null;

roleTextBox.SelectedItem = null;

UpdateParticipantsTextBox();

roleTextBox.Text = "";}}

private void UpdateParticipantsTextBox()

{textBox.Text = string.Join(", ", selectedPeople);}

private void SaveEventToDatabase(object sender, RoutedEventArgs e){

bool first = false;bool second = false;

if (selectedPeople.Count == 0){Warning.WarningShow("У события нет ни единого участника! Свяжите его хотя бы с одной персоной.", "Не хватает участников");return;}

string eventtype = typeComboBox.Text;

string eventdate = "";

if (datePickerOfEvent.SelectedDate.HasValue){

eventdate = datePickerOfEvent.SelectedDate?.ToString();}

string eventlocation = selectedPlace.PlaceId.ToString();

string description = descriptiontextbox.Text;

string comment = notesTextBox.Text;

string image = imageFilePath;

string sourceFilePath = imageFilePath;

string destinationFilePath = folder + $"\\{System.IO.Path.GetFileName(folder)}-Resources\\Events\\" + System.IO.Path.GetFileName(sourceFilePath);;

Events event\_ = events.Find(p => p.EventId == editEventId);

if ( ( AddOrEdit=="Edit" && (event\_.Picture != imageFilePath)) || AddOrEdit =="Add"){

if (System.IO.File.Exists(destinationFilePath)){

string targetDirectory = System.IO.Path.GetDirectoryName(destinationFilePath);

string fileName = System.IO.Path.GetFileNameWithoutExtension(destinationFilePath);

string fileExtension = System.IO.Path.GetExtension(destinationFilePath);

// Генерация нового имени файла с добавлением числа

int count = 1;

string newFileName = $"{fileName} ({count}){fileExtension}";

string newFilePath = System.IO.Path.Combine(targetDirectory, newFileName);

while (System.IO.File.Exists(newFilePath)){

count++;newFileName = $"{fileName} ({count}){fileExtension}";

newFilePath = System.IO.Path.Combine(targetDirectory, newFileName);}

destinationFilePath = newFilePath;}try

{System.IO.File.Copy(sourceFilePath, destinationFilePath);}

catch{}

image = System.IO.Path.GetFileName(destinationFilePath);}

eventdate = string.IsNullOrEmpty(eventdate) ? "NULL" : $"'{eventdate}'";

string connect = $"Provider=Microsoft.ACE.OLEDB.12.0;Data Source={databaseFilePath};Jet OLEDB:Engine Type=5";

using (OleDbConnection dbConnection = new OleDbConnection(connect))

{try{dbConnection.Open();if (AddOrEdit == "Add"){

string maxIdQuery = "SELECT MAX(EventId) FROM Events";

int lastEventId = 0;string event\_id = lastEventId.ToString();

using (OleDbCommand maxIdCommand = new OleDbCommand(maxIdQuery, dbConnection)){

object result = maxIdCommand.ExecuteScalar();

if (result != null && result != DBNull.Value){

lastEventId = Convert.ToInt32(result);}}

lastEventId++;

event\_id = lastEventId.ToString();

string query = $"INSERT INTO Events VALUES({event\_id},'{eventtype}',{eventdate},'{eventlocation}','{description}','{comment}','{image}')";

using (OleDbCommand dbCommand = new OleDbCommand(query, dbConnection)){

dbCommand.ExecuteNonQuery();}

foreach (People person in selectedPeople){

string personID = person.PersonId.ToString();

string eventID = event\_id;

string role = person.Role.ToString();

query = $"INSERT INTO Participants VALUES({eventID},{personID},'{role}')";

using (OleDbCommand dbCommand = new OleDbCommand(query, dbConnection)){

dbCommand.ExecuteNonQuery();}}}

else{

string query = $"UPDATE Events SET EventDate ={eventdate},EventLocationId='{eventlocation}',Description='{description}',Comment='{comment}',Picture='{image}' WHERE EventId = {editEventId}";

using (OleDbCommand dbCommand = new OleDbCommand(query, dbConnection)){

dbCommand.ExecuteNonQuery();}

query = $"DELETE FROM Participants WHERE EventId = {editEventId}";

using (OleDbCommand dbCommand = new OleDbCommand(query, dbConnection)){

dbCommand.ExecuteNonQuery();}

foreach (People person in selectedPeople){

string personID = person.PersonId.ToString();

string eventID = editEventId.ToString();

string role = person.Role.ToString();

query = $"INSERT INTO Participants VALUES({eventID},{personID},'{role}')";

using (OleDbCommand dbCommand = new OleDbCommand(query, dbConnection)){

dbCommand.ExecuteNonQuery();}}

if ((AddOrEdit == "Edit" && (event\_.Picture != imageFilePath)) || AddOrEdit == "Add"){

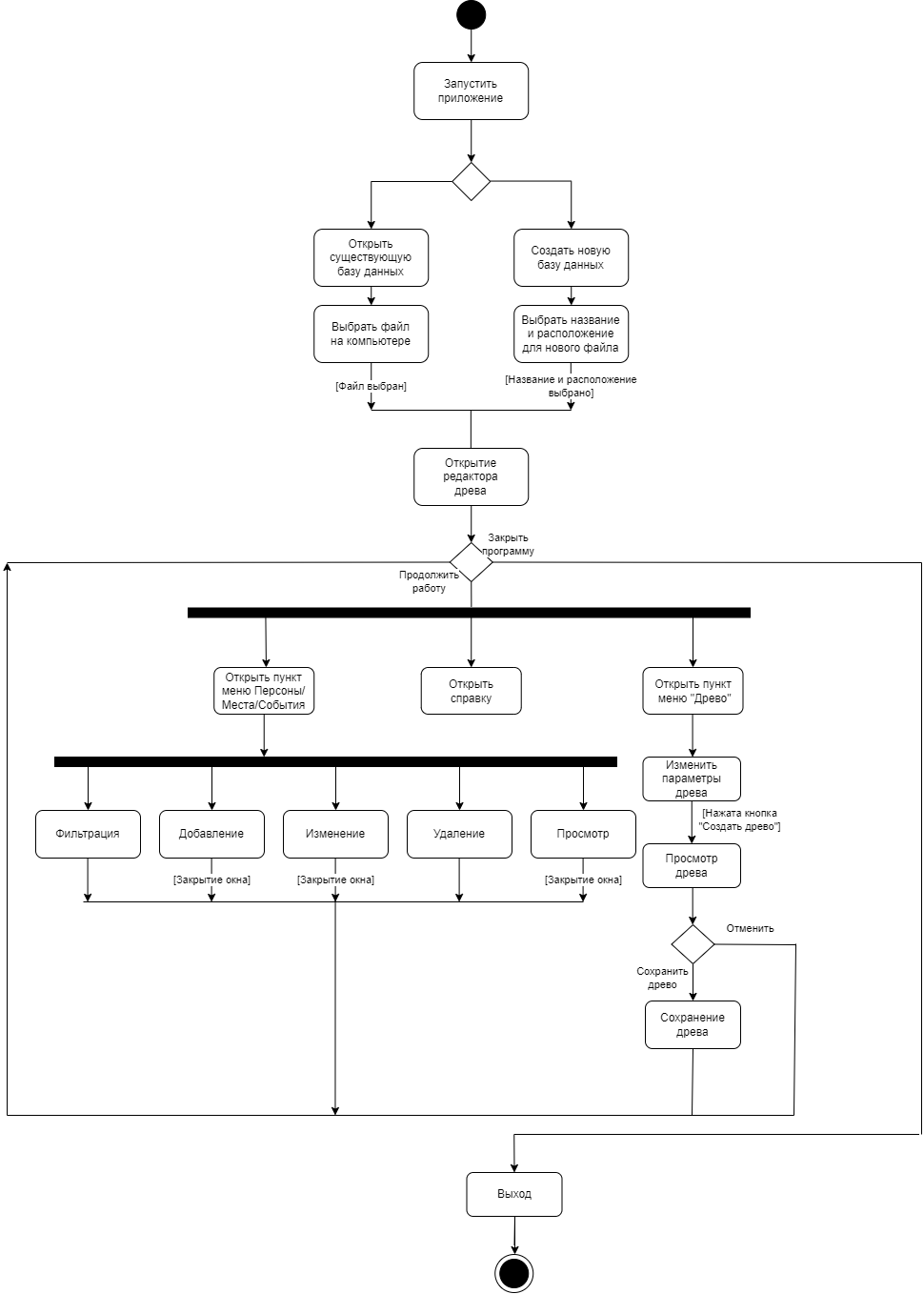
string deleteFile = folder + $"\\{System.IO.Path.GetFileName(folder)}-Resources\\Events\\" + event\_.Picture;

if (System.IO.File.Exists(folder + $"\\{System.IO.Path.GetFileName(folder)}-Resources\\Events\\" + event\_.Picture)){

try{System.IO.File.Delete(deleteFile);}catch{}}}}}catch(Exception er){}}Close();}}}}

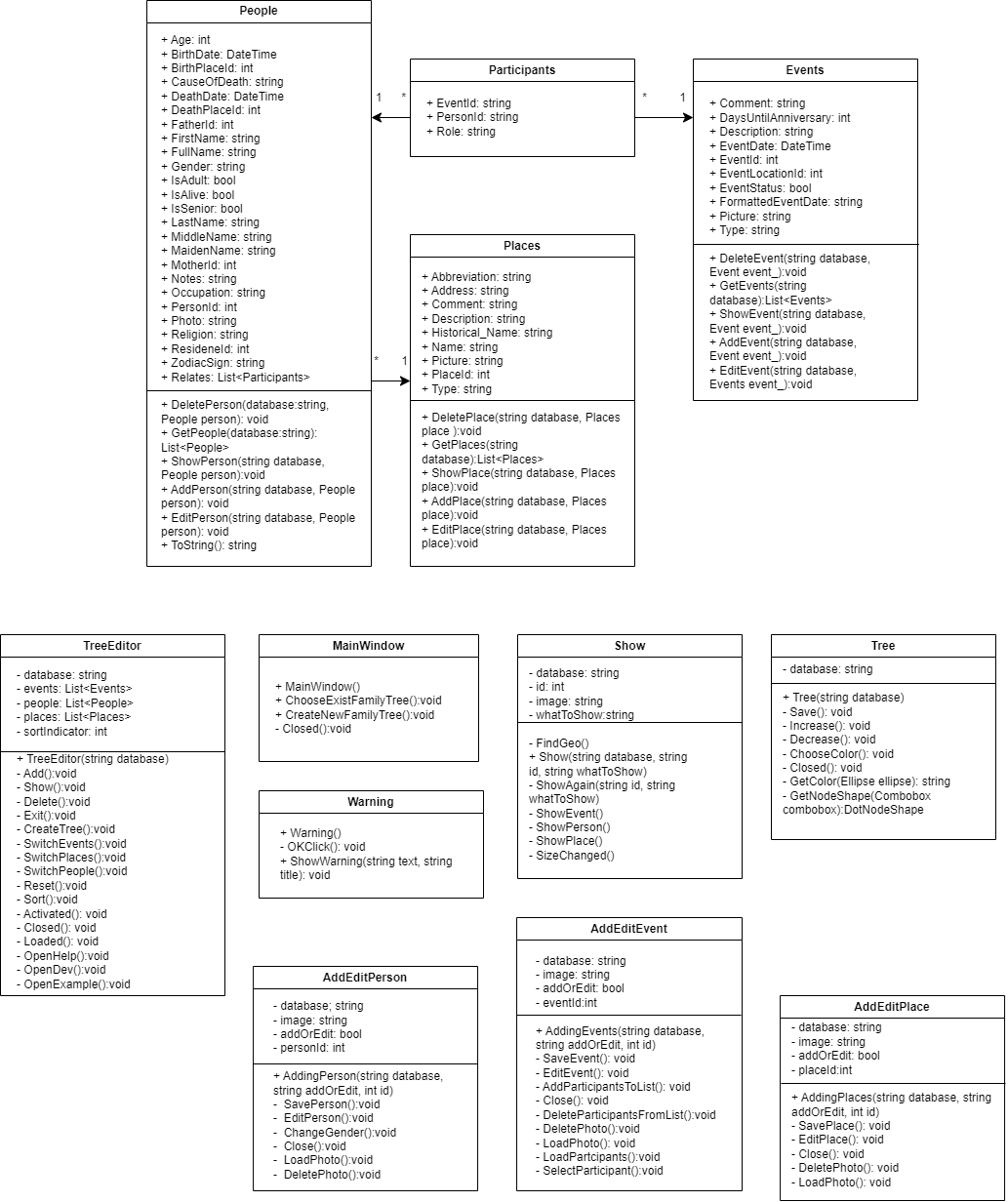
**Приложение Б**

**Диаграмма деятельности**

****

**Приложение В**

**Диаграмма классов**

****