**ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ ЯРОСЛАВСКОЙ ОБЛАСТИ**

государственное профессиональное образовательное учреждение   
Ярославской области

Рыбинский полиграфический колледж

**КУРСОВОЙ ПРОЕКТ**

|  |  |
| --- | --- |
| Разработка клиент-серверного приложения «ведение архивного фонда» | |
|  | |
| по дисциплине | Технология разработки и защиты баз данных |
|  | |

Пояснительная записка

|  |
| --- |
| КП.0902.09.000000.00 ПЗ |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент группы | 4-ИС-2 |  |  |  | Т. Х. Бабаев |
|  | *(Код учебной группы)* |  | *(Подпись, дата)* |  | *(И.О.Фамилия)* |
| Руководитель | преподаватель |  |  |  | Е. А. Лобанова |
|  | *(Должность, звание)* |  | *(Подпись, дата)* |  | *( И.О.Фамилия)* |
| Нормоконтроль | преподаватель |  |  |  | Е. А. Лобанова |
|  | *(Должность, звание)* |  | *(Подпись, дата)* |  | *(И.О.Фамилия)* |

г. Рыбинск

2022

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc122904910)

[1 Исследовательский раздел 4](#_Toc122904911)

[2 Конструкторский раздел 8](#_Toc122904912)

[2.1 Проектирование информационной модели данных 8](#_Toc122904913)

[2.2 Проектирование серверной части приложения 9](#_Toc122904914)

[2.2.1 Разработка схемы базы данных 9](#_Toc122904915)

[2.2.2 Разработка сущностей базы данных 11](#_Toc122904916)

[2.3 Проектирование клиентской части приложения 12](#_Toc122904917)

[2.3.1 Разработка модулей схемы 12](#_Toc122904918)

[2.3.2 Разработка пользовательского интерфейса 13](#_Toc122904919)

[2.3.3 Организация доступа к объектам базы данных 18](#_Toc122904920)

[2.3.4 Разработка блох-схем алгоритмов процедур и функций 19](#_Toc122904921)

[2.4 Обеспечение коллективного доступа. Защита информации 20](#_Toc122904922)

[3 Технологическая часть 22](#_Toc122904923)

[3.1 Тестирование и отладка приложения 22](#_Toc122904924)

[3.2 Инструкция администратора базы данных 23](#_Toc122904925)

[3.3 Инструкция по эксплуатации приложения 33](#_Toc122904926)

[4 Раздел охраны труда 37](#_Toc122904927)

[Заключение 39](#_Toc122904928)

[Список используемых источников 40](#_Toc122904929)

[Приложение А 41](#_Toc122904930)

Введение

В рамках данного курсового проекта планируется разработка приложения, где будет реализовано просмотр и хранение объектов разных клиентов архивного фонда. Пользователь сможет просматривать записи архивного фонда, которые хранят информацию о хранимых объектах. Данное приложение должно позволить реализовать удобное взаимодействие пользователя с хранимыми объектами в архивном фонде, удобнее всего реализовать это путем использования клиент-серверной архитектуры. Все данные будут храниться в базе данных на сервере, а клиент будет взаимодействовать с клиентской частью приложения.

Приложение позволит достичь автоматизации процессов ведения и хранения записей архивного фонда.

1 Исследовательский раздел

Процесс разработки программного обеспечения – набор правил, согласно которым построена разработка программного обеспечения. Приложение можно назвать клиент серверным если оно включает в себя клиент-серверную архитектуру. Разработку клиент-серверного приложения необходимо начинать с выбора архитектуры клиент-сервера.

Для разработки клиент-серверных систем имеется два подхода. Первый подход построение систем на основе двухзвенной архитектуры. Состоит из клиентской и серверной части. Как правило, серверная часть представляет собой сервер БД, на котором расположены общие данные. А клиентская часть представляет приложение, которое связывается с сервером базы данных, осуществляет к нему запросы и получает ответы. Такие системы используются в локальных сетях, т.к. нет затруднений с установкой клиентской части. Также системы с такой архитектурой более безопасны, т.к. могут использовать собственные протоколы передачи данных, не известные злоумышленникам. Поэтому многие крупные компании, которые располагаются не в едином месте и для соединения подразделений используют глобальную сеть Интернет, выбирают именно такую архитектуру построения клиент-серверных систем.

При разработке информационных систем, рассчитанных на широкую аудиторию, возникают проблемы с использованием двухзвенной архитектуры. Во-первых, пользователю необходимо иметь в наличии клиентскую часть, а, во-вторых, у неопытного пользователя, могут возникнуть проблемы с конфигурированием такой системы. Поэтому в последнее время, более часто разрабатывают приложения на базе трехзвенной архитектуры.

Второй подход построение систем на основе трехзвенной архитектуры. Серверная часть в этой архитектуре представляет собой сервер приложений и сервер базы данных. А в качестве клиента выступает веб-браузер. Такая система очень проста для пользователя. Ему необходимо знать только адрес сервера приложения и наличие веб-браузера на рабочем компьютере. Все данные представляются в виде HTML-разметки, с использование графики и JavaScript. Передача запросов от клиента к серверу приложений происходит по средствам CGI-интерфейса. Сервер приложений общается с сервером базы данных, используя другой интерфейс, зависящий от того, на основе каких средств строится конкретная информационная система. Недостатками такой архитектуры является использование общеизвестных протоколов и интерфейсов передачи данных. Злоумышленник может осуществить взлом системы, если она будет недостаточно хорошо проверять поступившие запросы от клиента.

При разработке клиент-серверных приложений необходимо учитывать на каких пользователей будет рассчитана данная информационная система и какие требования предъявляются к безопасности.

Если информационная система должна быть общедоступной и рассчитана на широкую аудиторию, то необходимо использовать трехзвенную архитектуру.

Если информационная система используется внутри предприятия, доступ имеют к ней ограниченные пользователи и требуется создать максимально безопасную и защищенную систему, то следует отдать предпочтение двухзвенной архитектуре [1].

В рамках курсового проекта был выбран первый способ для разработки клиент-серверной системы на основе двухзвенной архитектуры. Для реализации двухзвенной архитектуры была выбрана платформа WPF (Windows Presentetion Foundation).

При выборе WPF можно выделить такие преимущества как аппаратное ускорение через DirectX, что сильно влияет на производительность. Также можно отметить веб-подобную модель компоновки. Вместо того чтобы фиксировать элементы управления на месте с определенными координатами, WPF поддерживает гибкий поток, размещающий элементы управления на основе их содержимого. В результате получается пользовательский интерфейс, который может быть адаптирован для отображения высоко динамичного содержимого или к разным языкам. Преимуществами WPF являются такие особенности как, **веб-подобная модель компоновки, богатая модель рисования, развитая текстовая модель, стили и шаблоны, команды, декларативный пользовательский интерфейс, приложения на основе страниц.**

При выборе сред разработки были рассмотрены Visual Studio и Project Rider. Visual Studio – это удобная интегрированная среда разработки (IDE) от Microsoft, позволяющая быстро и эффективно создавать, и разрабатывать проект, выбрав для этого все необходимое. Среда использует платформы разработки программного обеспечения Microsoft: Windows API, Windows Forms, Windows Presentation Foundation, Windows Store и Microsoft Silverlight. Так же она принимает плагины, которые расширяют функциональные возможности практически на каждом уровне, включая добавление поддержки систем управления исходным кодом (таких как Subversion) и обеспечивает стандартный для Windows вид окон приложения. Единственным минусом можно считать сложность освоения данной среды разработки из-за её большого количества различных функций, спрятанных в подразделах меню. Информация из работы [2].

Project Rider – это среда от JetBrains для работы с платформой .NET. Она обладает поддержкой полного цикла. Фирменная черта продуктов JetBrains, воплощенная и в Project Rider. С Project Rider появиться возможность организовать весь цикл создания программного обеспечения: от идеи до поддержки. Функциональность Project Rider позволяет подключить MSBuild и XBuild, работать с CLI-проектами и организовать отладку приложений .NET and Mono. Множество опций для быстрого создания кода улучшает производительность. Кроссплатформенность Project Rider работает с Windows, Linux и MacOS. Из минусов можно выделить её молодость. Часть функциональности еще в разработке, не все стартовые ошибки исправлены. Так же можно отметить её стоимость. Самая дешевая версия Project Rider обойдется в 139 долларов за первый год использования. Но есть триал-версия и специальные предложения для студентов и непрофильных организаций. Информация из работы [3].

Из этих двух сред разработки был выбран Visual Studio, так как он обладает всем необходимым функционалом для реализации проекта, а также она является бесплатной и дольше находится на рынке труда.

2 Конструкторский раздел

2.1 Проектирование информационной модели данных

Для начала проектирования разрабатываемого клиент-серверного приложения следует представить его в виде модели «Черная сфера». Черная сфера представляет собой систему, внутреннее устройство которой неизвестно. В эту систему подаются входные данные, а на выходе из системы поступают выходные данные. Черная сфера представлена на рисунке 2.1.

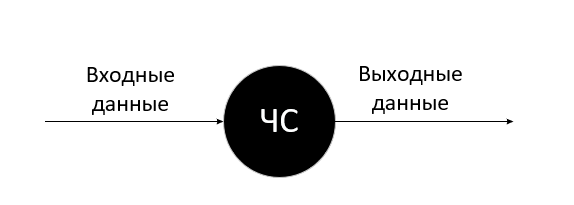


Рисунок 2.1 – Модель «Черная сфера»

Представим наше приложение в виде черной сферы. В приложении будут присутствовать такие выходные данные, как клиент и архивный фонд. На выходе из приложения будет объект. Клиентом будет выступать человек, планирующий сдать на хранение или забрать объект. Архивный фонд – это то, что позволит пользователю не думать о потере важной информации, ведь она будет безаопасно хранится в архивном фонде. Объект будет результатом, полученным после взаимодействия клиента с архивным фондом. Черная сфера с перечисленными входными и выходными параметрами представлена на рисунке 2.2.

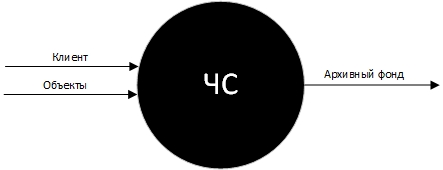


Рисунок 2.2 – Черная сфера с параметрами

В приложении существует один основной процесс смены содержимого окна. Изначально имеется три заготовленных вида содержимого. При определенных обстоятельствах, например, при нажатии пользователем на кнопку, ширина активного содержимого окна устанавливается в ноль, а ширина требуемого содержимого расширяется на всё окно. Большая часть данных для заполнения содержимого подгружается из базы данных и форматируется под интерфейс.

2.2 Проектирование серверной части приложения

2.2.1 Разработка схемы базы данных

Для выявления всех возможных сущностей будущей базы и получения концептуальной модели данных будет проведено несколько серий нормализации.

На первом этапе нормализации можно представить модель как связь между клиентом и архивным фондом. Первый этап нормализации представлен на рисунке 2.3.

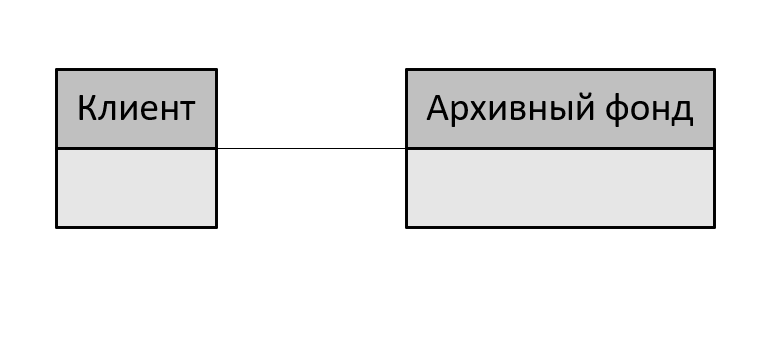


Рисунок 2.3 – Первый этап нормализации

Во втором этапе нормализации разобьём сущность архивного фонда на 2 сущности: объекты, и периоды хранения. Сущность объекты будет содержать в себе информацию об клиенте и о периодах хранения. Второй этап нормализации на рисунке 2.4.

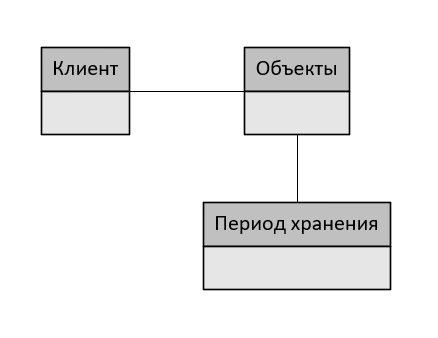


Рисунок 2.4 – Второй этап нормализации

В третьем заключительном этапе нормализации получим полную концептуальную схему. К сущности объекты добавилась сущность статус и секция хранения. Третий этап нормализации представлена на рисунке 2.5.

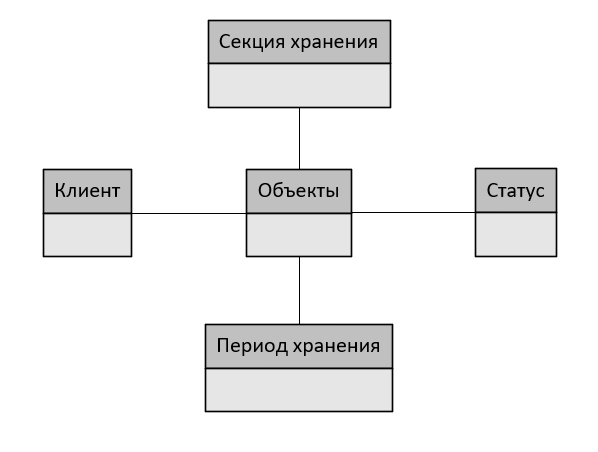


Рисунок 2.5 – Третий этап нормализации

Получим физическую модель данных, включающая ассоциативные таблицы, которые иллюстрируют отношения между сущностями, а также первичные и внешние ключи для связи данных. Физическая модель данных представлена на рисунке 2.7.



Рисунок 2.6 – Физическая модель данных

2.2.2 Разработка сущностей базы данных

В результатах раздела «Разработка схемы базы данных» получена схема базы данных, из которой следует необходимость присутствия определенных сущностей необходимых для полноценной работы приложения. Для удобства все сущности сведены в табличном виде. Сущности схемы базы данных представлены в таблице 2.1.

Таблица 2.1 – Сущности схемы базы данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя сущности | Назначение сущности | Типы данных | Перечисление наименований сущностей, которые подчиняются текущей сущности | Перечисление наименований сущностей, которым подчиняется текущая сущность |
| Client | Содержит данные об пользователе приложения | nvarchar(50), bit | Data | Отсутсвуют |
| Status | Содержит информацию о статусах объекта | Nvarchar(50) | Data | Отсутсвуют |
| StoragePeriod | Содержит информацию о периодах хранения | Nvarchar(50) | Data | Отсутсвуют |
| StorageSection | Содержит информацию о секциях хранения | Nvarchar(50) | Data | Отсутсвуют |
| Data | Содержит информацию об объектах хранения | Nvarchar(50), int, date | Отсутсвуют | Client, Status, StoragePeriod, StorageSection |

2.3 Проектирование клиентской части приложения

2.3.1 Разработка модулей схемы

WPF предоставляет комплексный набор функций разработки приложений, которые включают в себя язык XAML, элементы управления, привязку к данным, макет, двумерную и трехмерную графику, анимацию, стили, шаблоны, документы, мультимедиа, текст и типографические функции. WPF является частью .NET, поэтому вы можете создавать приложения, включающие другие элементы .NET API [4].

Представим клиентскую часть приложения в виде модульной схемы показывающая связь между окнами, классами и страницами при организации клиентской части приложения. Модульная схема клиентской части приложения представлена на рисунке 2.7.

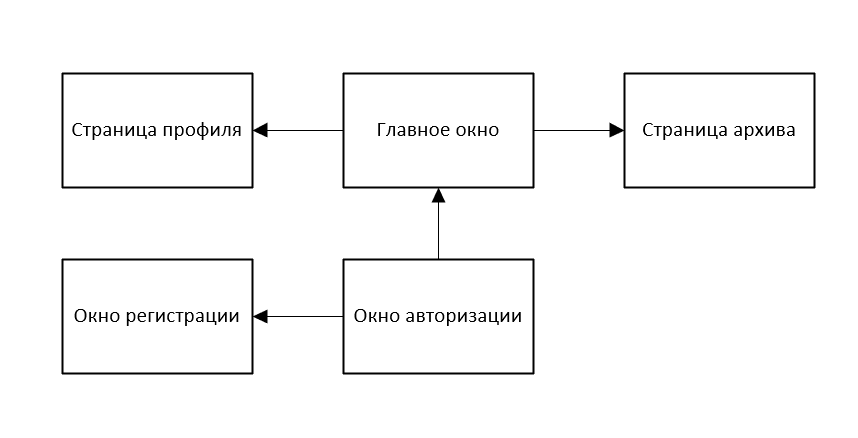


Рисунок 2.7 – Модульная схема клиентской части приложения

В составе модульной схемы присутствует окно авторизации и регистрации пользователя, главное окно, для просмотра всего архива, страница архива, для взаимодействия с таблицей с хранимыми объектами, страница профиля, для взаимодействия с личным кабинетом.

2.3.2 Разработка пользовательского интерфейса

Пользовательский интерфейс – это совокупность информационной модели проблемной области, средств и способов взаимодействия пользователя с информационной моделью, а также компонентов, обеспечивающих формирование информационной модели в процессе работы программной системы.

**Графический пользовательский интерфейс – самый** популярный тип пользовательского интерфейса. Он представляет собой окно с различными элементами управления. Пользователи взаимодействуют с элементами с помощью клавиатуры, компьютерной мыши и голосовых команд.

**XAML** представляет собой язык разметки, используемый для создания экземпляров объектов .NET. Хотя язык XAML — это технология, которая может быть применима ко многим различным предметным областям, его главное назначение — конструирование пользовательских интерфейсов WPF. Другими словами, документы XAML определяют расположение панелей, кнопок и прочих элементов управления, составляющих окна в приложении WPF.

На этом этапе рассмотрим каждый блок модульной схемы поподробнее. При запуске приложения открывается окно авторизации. Данное окно содержит в себе логотип и название приложения, имеются два текстовых поля для ввода логина и пароля, а также кнопка для входа в программу, кнопка регистрации пользователя и кнопка выхода из программы. Внешний вид страницы авторизации представлен на рисунке 2.8.

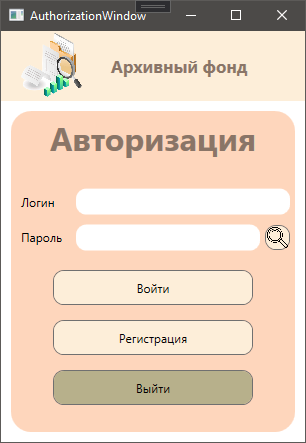


Рисунок 2.8 – Внешний вид окна авторизации

Окно регистрации содержит в себе логотип и название приложения. Имеются пять текстовых поля для ввода имени, номера телефона, логина, пароля и подтверждение пароля, а также две кнопки для создания аккаунта и кнопка отмены. Внешний вид страницы регистрации представлен на рисунке 2.9.

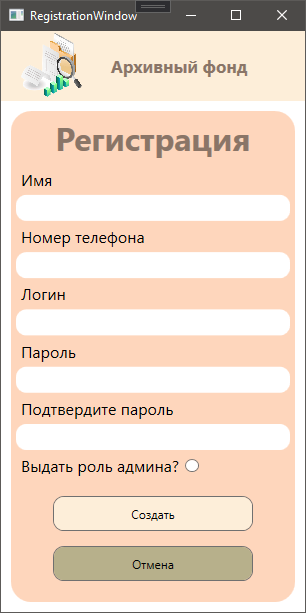


Рисунок 2.9 – Внешний вид окна регистрации

При авторизации под клиентской учетной записью в главном окне присутствует три кнопки управления и область для отображения страницы. При нажатии на кнопку архив открывается страница со справочником. На странице присутствуют две кнопки управления и таблица с записями архивного фонда. При двойном нажатии правой кнопкой компьютерной мыши по записи таблицы, справа от таблицы отобразится информация о записи. Первая кнопка над справочником отвечает за открытие панели с фильтром по разным группам в виде древовидной системы. При нажатии по элементу, в справочнике отобразится информация о нем. Вторая кнопка отвечает за фильтрацию таблицы по записям клиента. При нажатии по кнопке отобразятся только клиентские записи. Внешний вид страницы регистрации представлен на рисунке 2.10.

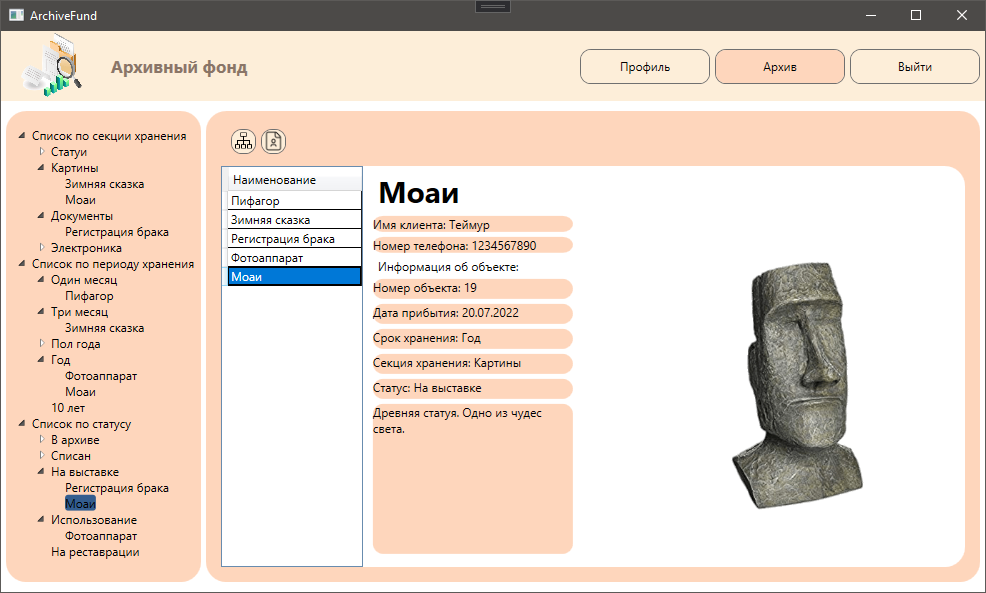


Рисунок 2.10 – Внешний вид страницы справочника

При нажатии на кнопку профиля откроется страница личного кабинета. В личном кабинете отображается вся информация о клиенте. Информацию можно отредактировать и нажать на кнопку изменить. При нажатии на фотографию откроется окно с выбором фотографии. После выбора фотографии, она отобразится в личном кабинете. Внешний вид страницы регистрации представлен на рисунке 2.11.

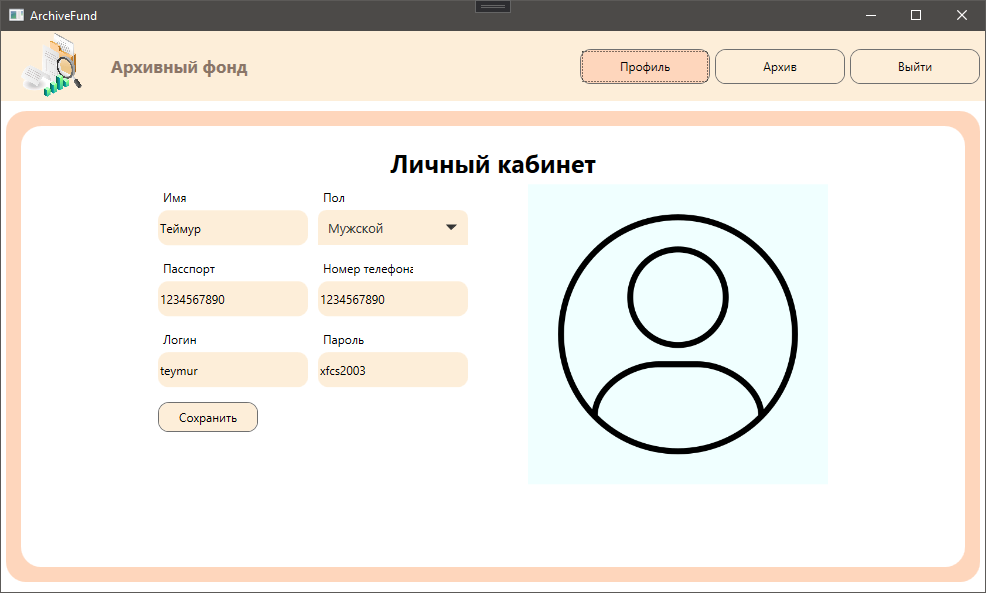


Рисунок 2.11 – Внешний вид страницы личного кабинета

При авторизации под администраторской учетной записью в главном окне присутствует кнопка выхода из учетной записи и страница со справочником. На странице справочника присутствует пять кнопок управления и таблица с записями архивного фонда. Первая кнопка отвечает за открытие панели с фильтром в виде древовидной системой. Следующие кнопки являются административные. В их состав входят кнопка добавления новой, копирование, редактирования и удаление записи. При нажатии на кнопку удаления записи, выйдет сплывающее окно с подтверждением удаления. Внешний вид страницы справочника представлен на рисунке 2.12.

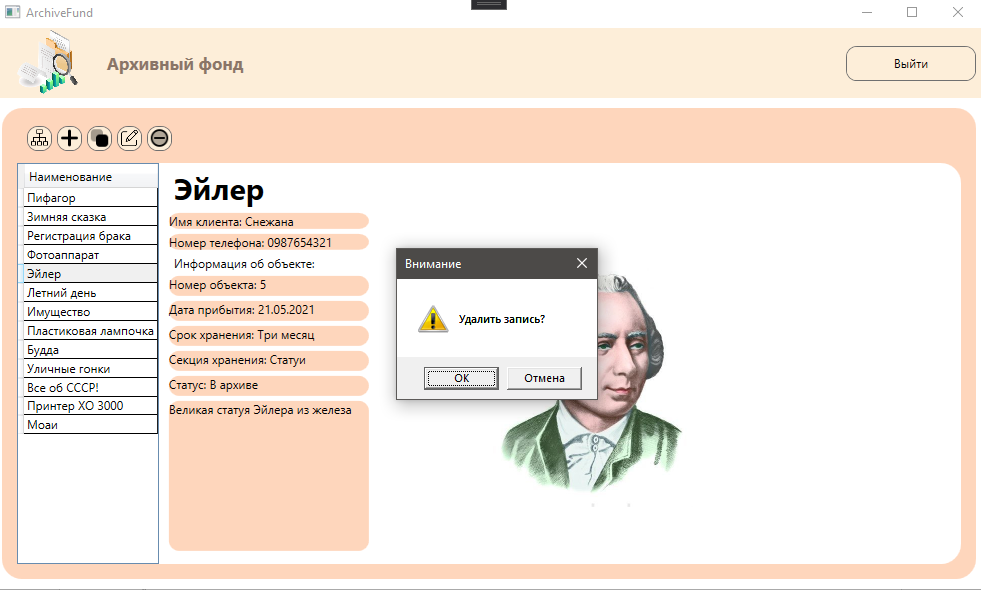


Рисунок 2.12 – Внешний вид страницы справочника

При нажатии кнопки на кнопку добавления новой записи в справочник откроется панель создания записи. При нажатии на кнопку копирования и удаления произойдет тоже самое. Внешний вид страницы справочника представлен на рисунке 2.13.

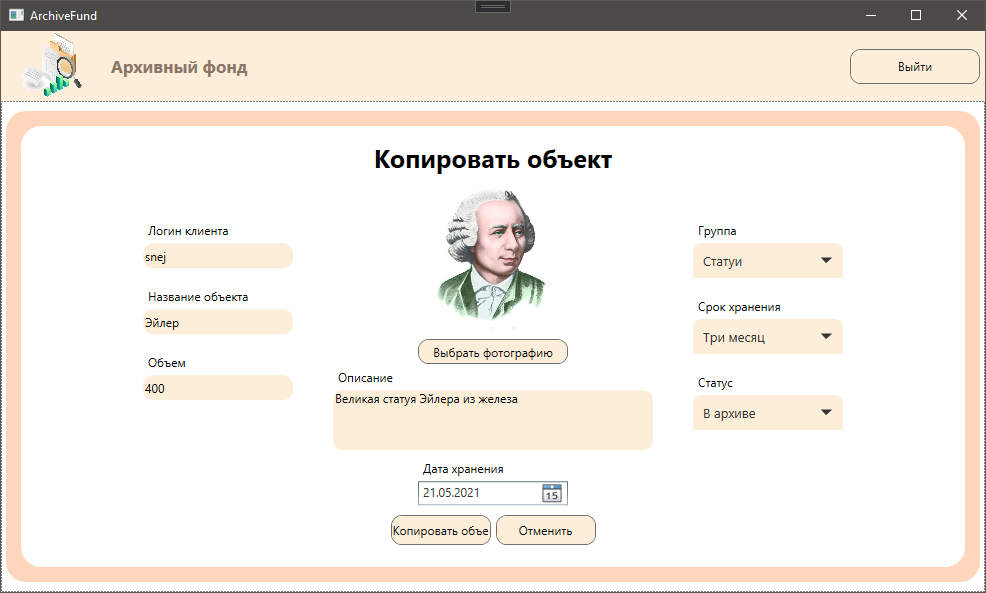


Рисунок 2.13 Внешний вид панели редактирования на странице справочника

2.3.3 Организация доступа к объектам базы данных

В WPF привязка (binding) является мощным инструментом программирования, без которого не обходится ни одно серьезное приложение.

Привязка подразумевает взаимодействие двух объектов: источника и приемника. Объект-приемник создает привязку к определенному свойству объекта-источника. В случае модификации объекта-источника, объект-приемник также будет модифицирован. [6]

Возьмем для примера страницу справочника, где участвует DataGrid. DataGrid страницы справочника представлен на рисунке 2.14.

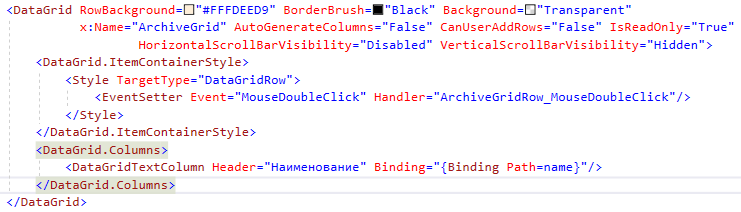


Рисунок 2.14 – DataGrid страницы справочника

У Binding в свойство Path мы записываем свойство объекта источника. Привязка объекта с данными к DataGrid представлена на рисунке 2.15.



Рисунок 2.15 – Привязка объекта с данными к DataGrid

Берем данные из базы данных, затем привязываем их с DataGrid. Сам DataGrid верстается под конкретно содержимое.

2.3.4 Разработка блох-схем алгоритмов процедур и функций

Основными функциональными задачами программного обеспечения являются просмотр и добавление записей в архивном фонде. В связи с этим рассмотрим блок-схемы основных алгоритмов программного обеспечения, а именно, загрузка самих записей в древовидном виде, чтобы иметь возможность просматривать записи в разных группах.

Метод TreeViewLoader предназначен для просмотра записей архивного фонда в виде древовидной структуры. У метода нет ни входных, ни выходных данных, так как данные получаются в методе и в нём же передаются в элемент отображения. Блок-схема метода TreeViewLoader представлена на рисунке 2.16.

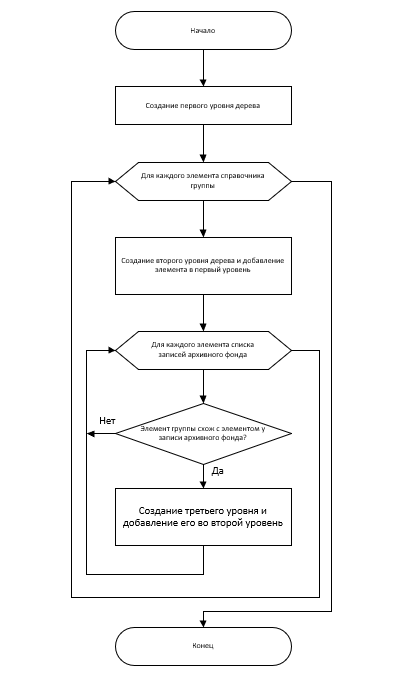


Рисунок 2.16 – Блок-схема метода TreeViewLoader

2.4 Обеспечение коллективного доступа. Защита информации

Основная идея ролевой модели контроля за доступом (Role-Based Access Control — RBAC) основана на максимальном приближении логики работы системы к реальному разделению функций персонала в организации.

Ролевой метод управления доступом контролирует доступ пользователей к информации на основе типов их активностей в системе. Применение данного метода подразумевает определение ролей в системе. Понятие роль можно определить, как совокупность действий и обязанностей, связанных с определенным видом деятельности. Таким образом, вместо того, чтобы указывать все типы доступа для каждого пользователя к каждому объекту, достаточно указать тип доступа к объектам для роли. А пользователям, в свою очередь, указать их роли. Пользователь, «выполняющий» роль, имеет доступ, определенный для роли [6].

В системе доступно две роли, администратор и обычный пользователь. У администратора в отличии от обычного пользователя есть одно отличие – это наличие доступа к панели администратора, где можно редактировать таблицы из базы данных.

Для авторизации пользователю необходимо ввести логин и пароль. В случае если пользователь не зарегистрирован, он сможет перейти на страницу регистрации из окна авторизации.

Для регистрации пользователю требуется ввести его имя и фамилию, почту, а также пароль и повторно подтвердить пароль. Все поля обладают своей валидацией и в случае некорректного ввода данных, поле ввода приобретет красный цвет обводки. При желании пользователь может вернуться на страницу авторизации.

3 Технологическая часть

3.1 Тестирование и отладка приложения

Отладка — этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки, информация из работы [7]. В связи с тем, что почти невозможно составить реальную программу без ошибок, и почти невозможно для достаточно сложной программы быстро найти и устранить все имеющиеся в ней ошибки. Разумно уже при разработке программы на этапах алгоритмизации и программирования готовиться к обнаружению ошибок на стадии отладки принимать профилактические меры по их предупреждению, информация из работы [8].

Тестирование будет происходить при помощи составления тест кейсов. Тест кейс – это артефакт, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части. При передаче тестировщику тест-кейсов, он должен пройтись по всем его пунктам и выполнить описанные действия, которые должны привести к определенным результатам. информация из работы [9]. Тест кейс для функций представлен в таблице 3.1.

Таблица 3.1 – Тест-кейс для методов

|  |  |  |
| --- | --- | --- |
| Имя метода | Управляющее воздействие | Результат воздействия |
| AuthorizationCommit | Вызывается при нажатии кнопки «Войти» в окне авторизации | Отображение диалога ошибки если данные были введены некорректно |
| IsLoginValidation | Вызывается при нажатии кнопки «Регистрация» в окне регистрации | Отображение диалога ошибки если логин с таким названием уже существует |
| IsRegisterValidation | Вызывается при нажатии кнопки «Регистрация» в окне регистрации | Отображение диалога ошибки если данные были введены некорректно |

Окончание таблицы 3.1

|  |  |  |
| --- | --- | --- |
| Имя метода | Управляющее воздействие | Результат воздействия |
| PrintSelectedItem | Вызывается при двойном нажатии по записи | Заполнение графического элемента данными |
| TreeViewLoader | Вызывается при нажатии на кнопку «Показать древовидную структуру» в окне справочника | Заполнение графического элемента данными |
| GetClient | Вызывается при нажатии на кнопку «Добавить объект» на панели редактирования записи в окне справочника | Отображение диалога ошибки если логина с таким названием не существует |
| ConvertToBase64 | Вызывается при вставке в графический объект фотографию в главном окне на странице личного кабинета | Заполнение элемента данными |
| ConvertToImage | Вызывается при вставке в графический объект фотографию в главном окне на странице личного кабинета | Заполнение графического элемента данными |
| GetData | Вызывается при нажатии на элемент в древовидной структуре в главном окне на странице справочника | Возвращение значения |

3.2 Инструкция администратора базы данных

Перед началом работы с приложением необходимо установить и настроить SQL Server 2019. MS SQL Server это лидирующая РСУБД (Реляционная система управления базами данных) а также главный конкурент Oracle Database в корпоративном сегменте. В СНГ MSSQL чаще всего применяется для собственных разработок прикладного ПО и для 1С.

Для установки переходим на официальный сайт Microsoft и скачиваем бесплатную версию SQL Server 2019 для тестирования и разработки (Developer). Далее запускаем установщик и выбираем тип установки «Пользовательский». Как на рисунке 3.1

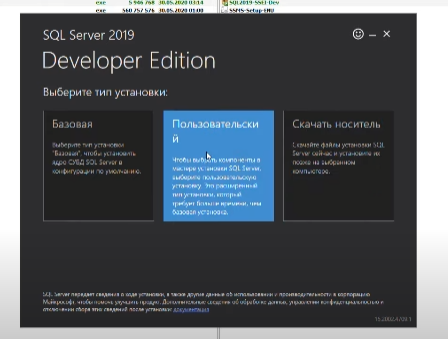


Рисунок 3.1 – «Пользовательский» тип установки

После выбора типа установки открывается следующее окно где предлагается выбрать язык и место расположения носителя, можно выбрать стандартные настройки и нажать на кнопку «Установить». После чего начнется процесс загрузки. Изображение окна представлено на рисунке 3.2.

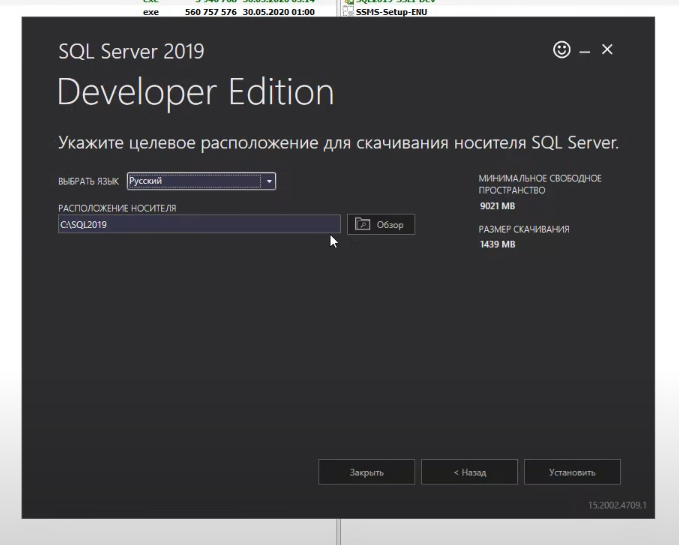


Рисунок 3.2 – Изображение окна

После установки откроется центр установки SQL server, где мы переходим в раздел установки «Новая установке изолированного экземпляра SQL Server или добавление компонентов к существующей установке», как показано на рисунке 3.3.

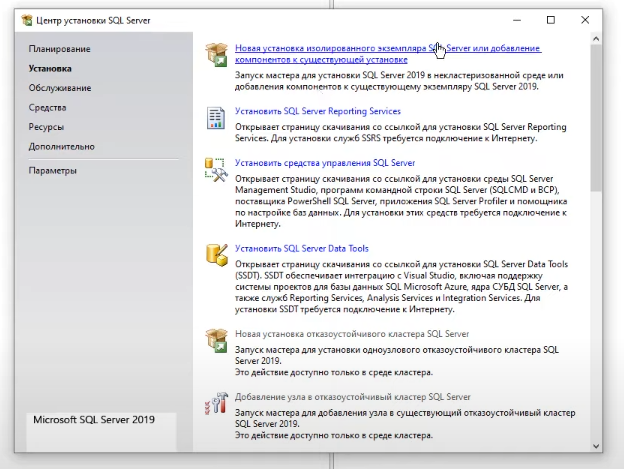


Рисунок 3.3 – Новая установке изолированного экземпляра SQL Server

После установки произойдет обновление продукта. Рисунок 3.4.

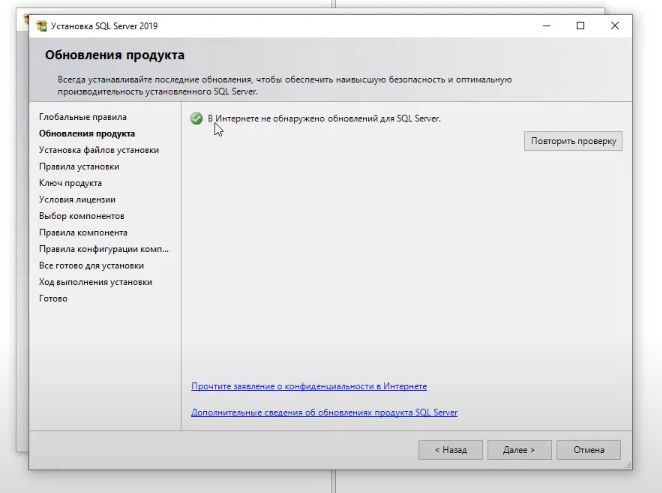


Рисунок 3.4 – Обновление продукта

После обновления пропускаем все пункты до ключа продукта и выбираем версию «Developer» как на рисунке 3.5.

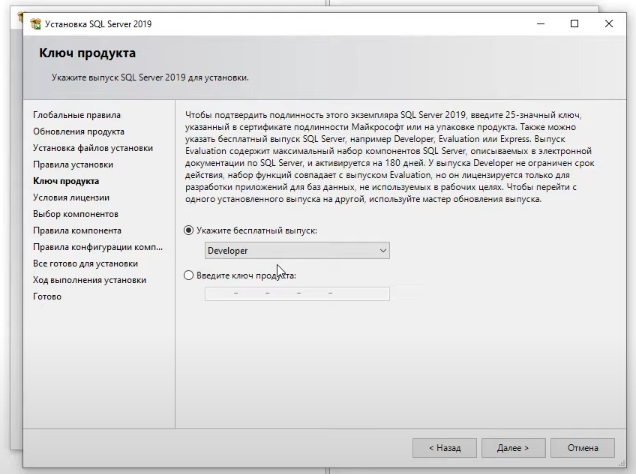


Рисунок 3.5 – Версия «Developer»

В пункте «Условия лицензии» принимаем условия и переходим в раздел «Выбор компонентов», где установим базовый набор компонентов: «Служба ядра СУБД» и «Полнотекстовой и семантический поиск». Как представлено на рисунке 3.6.

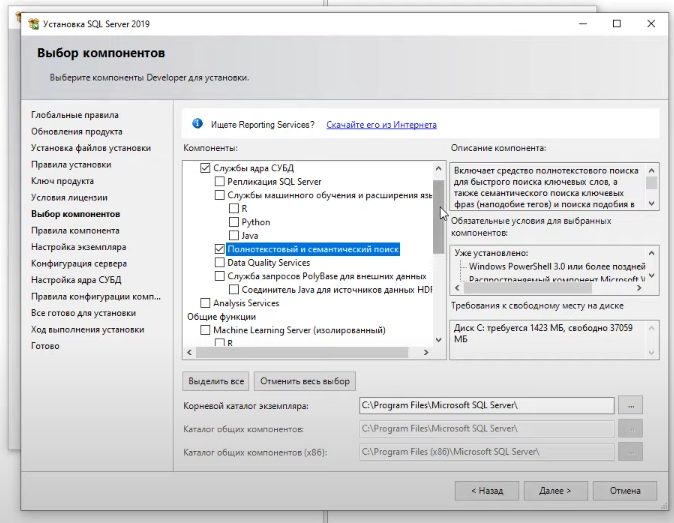


Рисунок 3.6 – Выбор компонентов

В разделе правила компонента всё оставляем по умолчанию и переходим в раздел «Конфигурация сервера» где можно настроить работу служб SQL Server. Задать тип запуска какой-либо службы. Поставить ее на автозапуск, вручную, или отключить. Так же можем зайти в меню "Параметры сортировки" — это настройки таблицы кодировок. Выполнять сортировку, как учитывать верхний и нижний регистр, как реагировать на символы, и т.п. Настройки можно оставить по умолчанию как на рисунке 3.7.

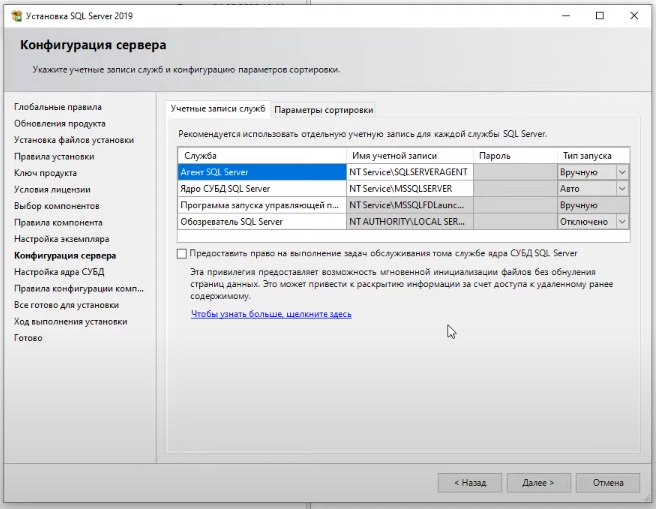


Рисунок 3.7 – Конфигурация сервера

В разделе «Настройка ядра СУБД» нам предлагают выбор режим входа под учетными записями Windows, либо смешанный режим, т.е. возможность входа под учетной записью Windows и под учетной записью SQL Server, если выбрать смешанную, то вам предложат создать учетную запись SQL Server. Оставляем режим аутентификации Windows и выбираем пользователя как показано на рисунке 3.8.

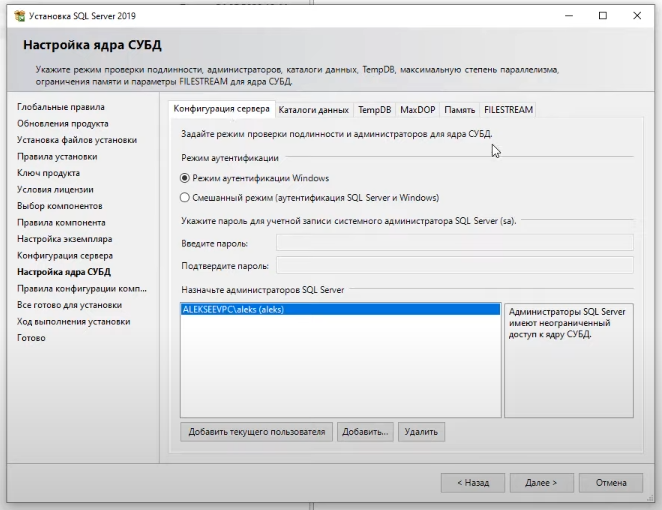


Рисунок 3.8 – Настройка ядра СУБД

В разделе «Все готово для установки» можно сверится с выбранными настройками» и начать установку. Окно с разделом представлено на рисунке 3.9.

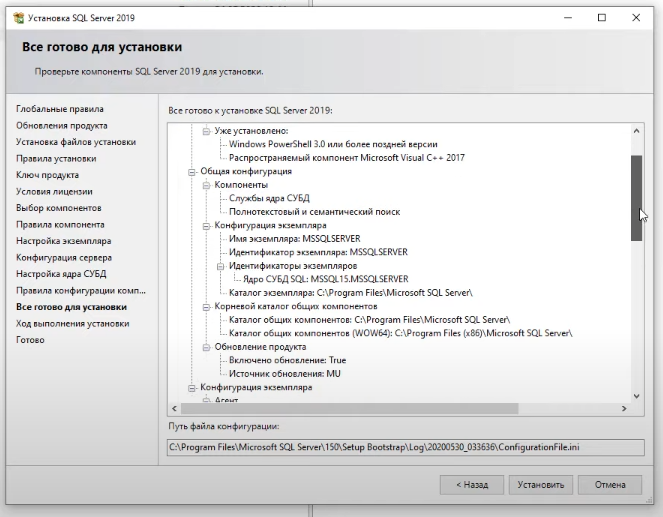


Рисунок 3.9 – Все готово для установки

Установка завершена и можно закрыть окно. Финальный экран представлен на рисунке 3.10.

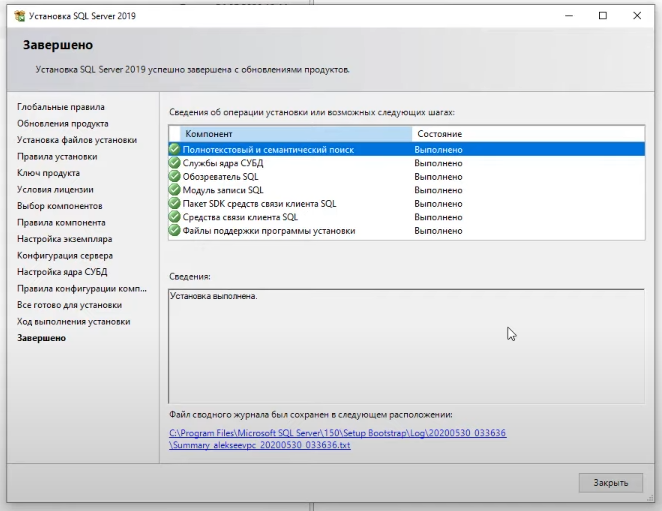


Рисунок 3.10 – Финальный экран

Если все прошло успешно, закрываем окно. После того, как установка SQL Server 2019 завершена, нам нужно установить приложение, с помощью которого мы будем подключаться к серверу баз данных. Это приложение SQL Server Management Studio (SSMS).

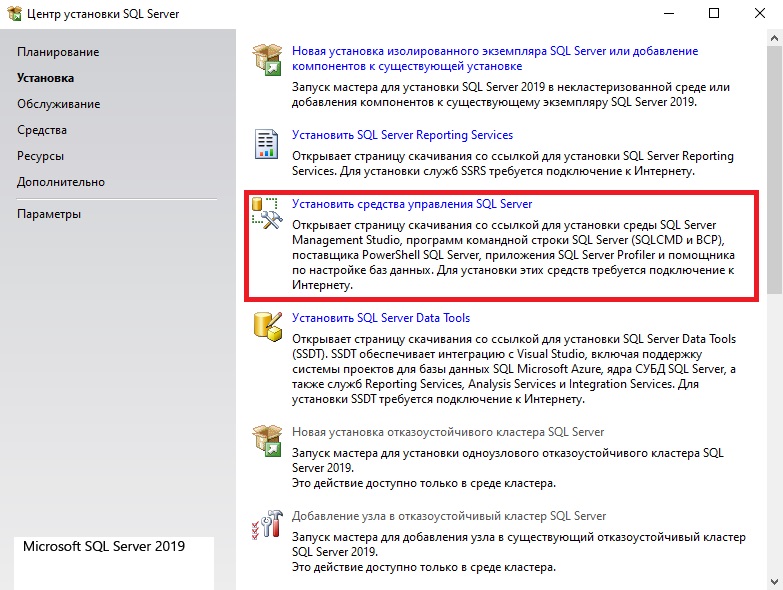
Заходим снова в центр установки SQL Server и наживаем "Установить средства управления SQL Server", как показано на рисунке 3.11.

Рисунок 3.11 – Установка SQL Server Management Studio

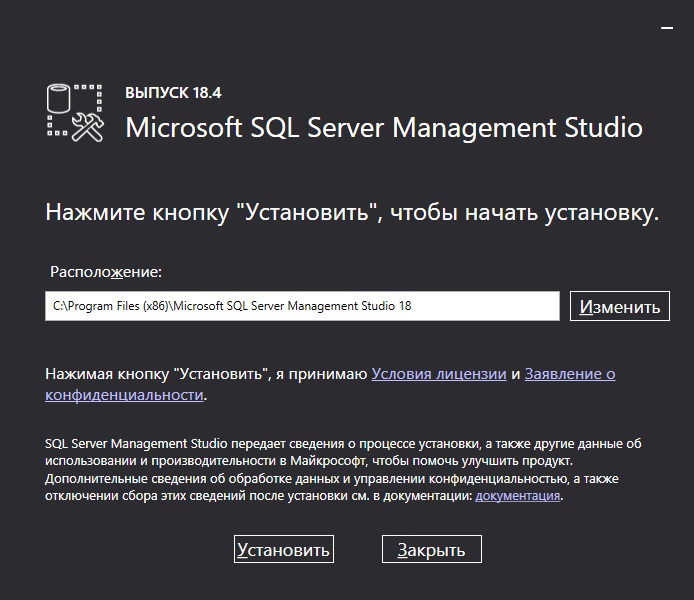
При нажатии у нас откроется [сайт Microsoft](https://docs.microsoft.com/ru-ru/sql/ssms/download-sql-server-management-studio-ssms?redirectedfrom=MSDN&view=sql-server-ver15) и нам нужно будет скачать SSMS. Нажимаем "Установить", как показано на рисунке 3.12.

Рисунок 3.12 – Установка SQL Server Management Studio

3.3 Инструкция по эксплуатации приложения

Приложение предназначено для удобного ведения архивного фонда, просмотра записей и их редактирование. Пользователь может установить приложения, после чего пройти регистрацию и в любой момент воспользоваться им просматривая все или свои записи в архивном фонде. Клиент сможет увидеть свои записи после того, как администратор добавит их в справочник.

Чтобы начать работать с приложение достаточно его запустить. После запуска, приложение готово к работе. Для доступа к справочникам должен войти пользователь со статусом администратора. Тогда ему будет доступна панель, позволяющая работать со справочником. Окно с авторизованным пользователем с правами администратора представлено на рисунке 3.13.

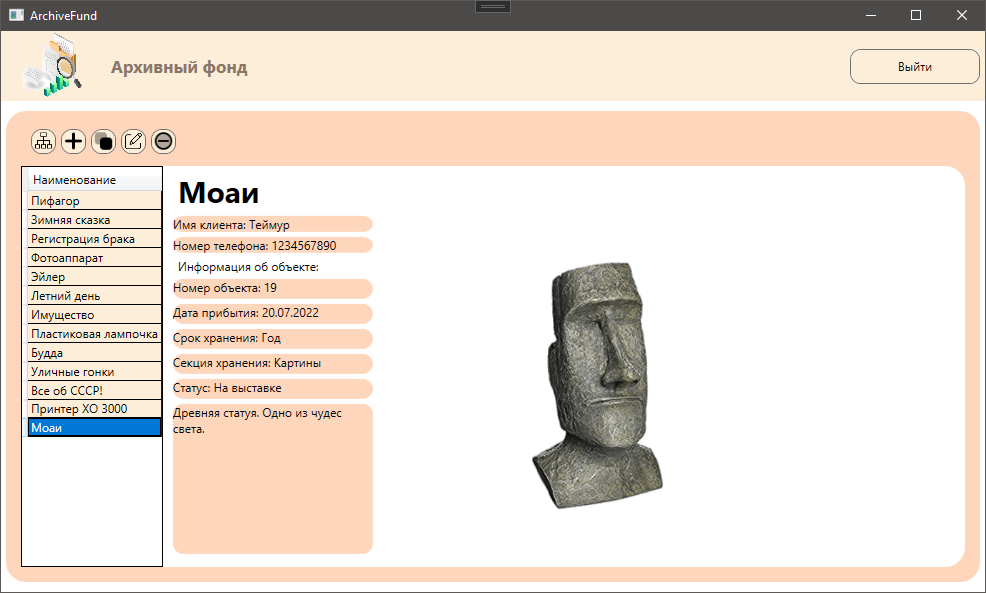


Рисунок 3.13 – Окно с авторизованным пользователем с правами администратора

Чтобы добавить в справочник материалы, которые принес клиент, нужно на панели инструментов нажать на кнопку добавления новой записи. Вести имя клиента и информацию об объекте хранения, а также выбрать срок хранения и статус объекта. Панель добавления новой записи представлено на рисунке 3.14.

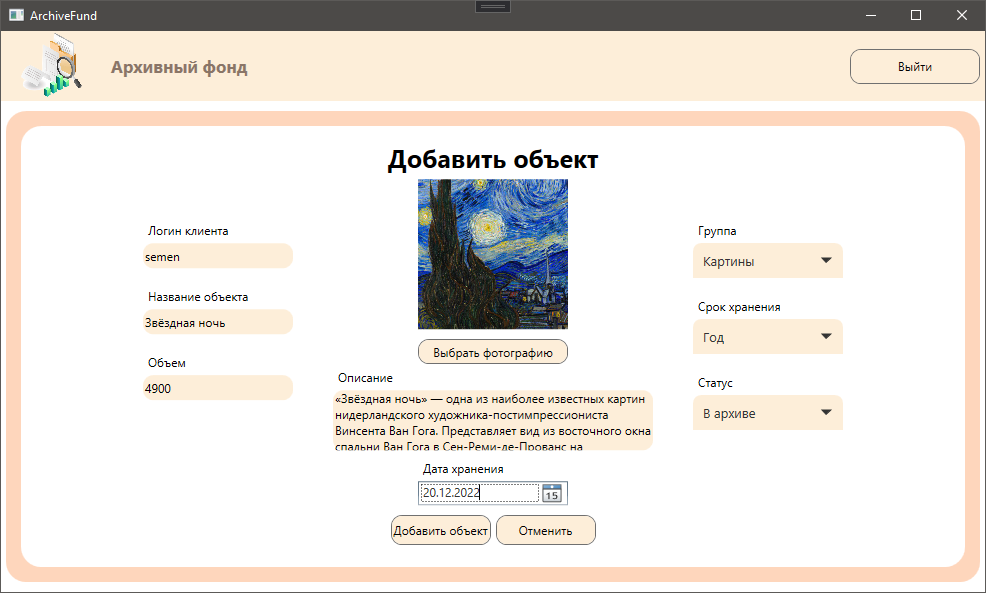


Рисунок 3.14 - Панель добавления новой записи

После добавления записи в справочник, она отобразится в справочнике и при двойном клике по левой кнопке мыши, отобразится вся информация о данном объекте. Справочник с новой записью представлен на рисунке 3.15.

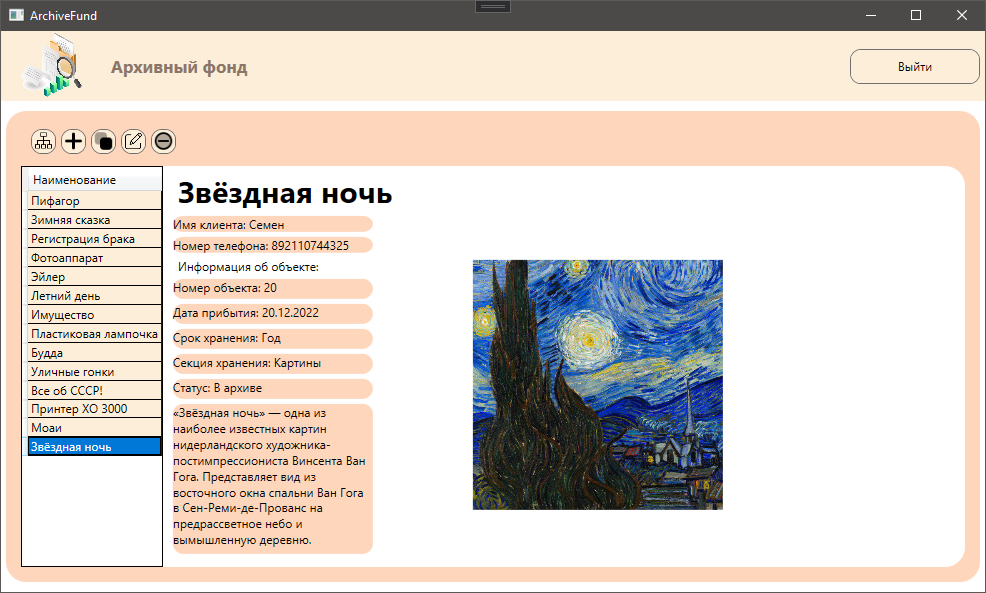


Рисунок 3.15 – Справочник с новой записью

После того как администратор добавил материалы об объекте клиента в справочник. Клиент заходит в учетную запись под своим логином и паролем. При нажатии на кнопку показа личных записей на панели инструментов, клиент сможет увидеть свою запись. Окно с авторизованным пользователем с правами клиента представлено на рисунке 3.16.

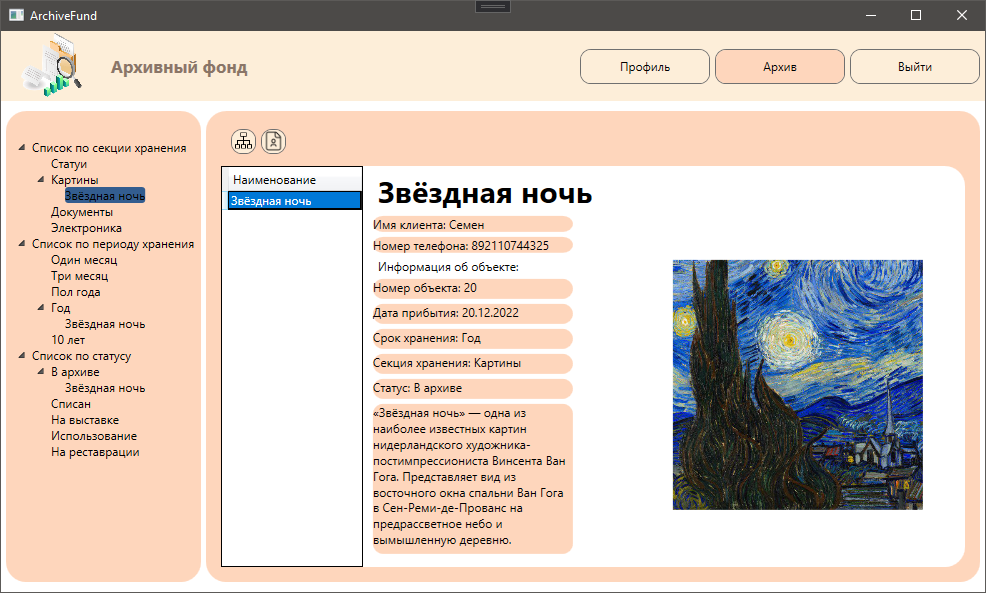


Рисунок 3.16 - Окно с авторизованным пользователем с правами клиента

Также клиент может настроить информацию об учетной записи в личном кабинете. При нажатии на кнопку профиль откроется страница с личным кабинетом, где можно отредактировать данные. Страница с личным кабинетом представлена на рисунке 3.17.

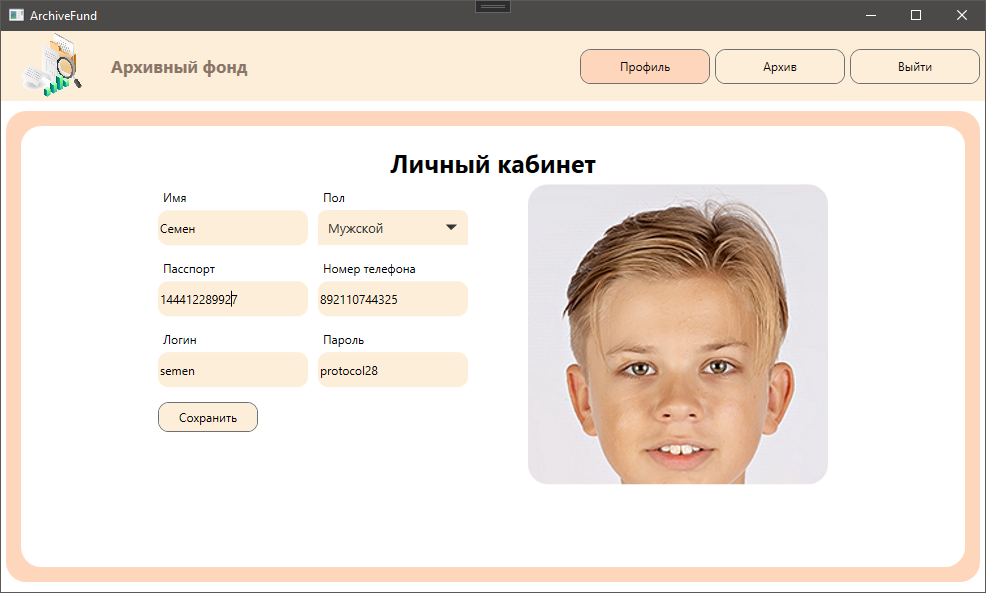


Рисунок 3.17 – Страница личного кабинета

4 Раздел охраны труда

Охрана труда – это целая система законодательных и нормативно-правовых актов, технических, гигиенических, лечебно-профилактических мероприятий и средств, которые обеспечивают безопасность, сохранение здоровья и работоспособности человека в процессе труда. В наши дни труд стал более интенсивным и требует огромных затрат умственной, эмоциональной и физической нагрузок.

На рабочем месте программист осуществляет трудовую деятельность и проводит большую часть рабочего времени. Правильная организация рабочего места программиста повышает производительность труда от 8 до 20%. Следуя рекомендациям ГОСТ 12.2.032-78, необходимо организовать рабочее место таким образом, чтобы взаимное расположение всех его элементов соответствовало физическим и психологическим требованиям. Главные элементы рабочего места программиста – это письменный стол и кресло. Рабочее место организуется в соответствии с ГОСТ 12.2.032-78, информация из работы [10].

Площадь рабочего места с компьютером с жидкокристаллическим или плазменным экраном должна быть не менее 4,5 кв. м, а расстояние между столами с мониторами (от тыла одного монитора до экрана другого) не менее 2 м. Монитор должен располагаться на расстоянии 50-70 см от глаз программиста. Параметры рабочего стола сотрудника: возможность регулировки высоты рабочего стола, или точная высота — 72,5 см, ширина — 80, 100, 120 или 140 см, глубина рабочего стола 80 или 100 см, высота и ширина пространства под столешницей (для ног) – не менее 50 см, глубина на уровне колен не менее 45 см, а на уровне вытянутых ног не менее 65 см.

Правильное освещение рабочего места – это очень важный момент в трудовой деятельности человека, влияющий на эффективность труда, при этом такой момент предупреждает травматизм и профессиональные заболевания. При недостаточном освещении приходится напрягать зрение, при этом ослабляется внимание и это приводит к наступлению преждевременной утомленности. Слишком яркое освещение тоже плохо, так как оно вызывает ослепление, раздражение и резь в глазах. При искусственном освещении, источниками света служат два вида ламп: лампы накаливания и люминесцентные.

Известно, что шум ухудшает условия труда и оказывает вредное воздействие на организм человека. Согласно ГОСТ 12.1.003-88 «Шум для помещений расчетчиков и программистов, уровни шума не должны превышать соответственно: 71, 61, 54, 49, 45, 42, 40, 38 дБ», информация из работы [11].

При работе компьютерной техники выделяется много тепла, что может привести к пожароопасной ситуации. Источниками зажигания так же могут служить приборы, применяемые для технического обслуживания, устройства электропитания, кондиционеры воздуха. Серьёзную опасность представляют различные электроизоляционные материалы, используемые для защиты от механических воздействий отдельных радиодеталей. В связи с этим, участки, на которых используется компьютерная техника, по пожарной опасности относятся к категории пожароопасных “В”. При пожаре люди должны покинуть помещение в течение минимального времени. В помещениях с компьютерной техникой, недопустимо применение воды и пены ввиду опасности повреждения или полного выхода из строя дорогостоящего электронного оборудования. Для тушения пожаров необходимо применять углекислотные и порошковые огнетушители, которые обладают высокой скоростью тушения, большим временем действия, возможностью тушения электроустановок, высокой эффективностью борьбы с огнем. Воду разрешено применять только во вспомогательных помещениях, из работы [12].

Заключение

В результате работы над курсовым проектом было разработано приложение, позволяющее пользователю ознакомиться с хранением объектов в архивном фонде. Для возможности использования программного обеспечения, пользователь обязан зарегистрироваться в приложении, либо авторизоваться, если он ранее уже был зарегистрирован. Для пользователей с правами администратора есть доступ к редактированию информации о записях справочника.

Главным достоинством можно выделить приятный интерфейс и простоту в использовании приложения. Приложение удобно для пользователя и интуитивно-понятно.

Разработанное приложение имеет курс доработки, улучшив оптимизацию интерфейса, добавления новых функций с пользователями и хранимыми объектами в архивном фонде. Основное направление доработки приложения состоит в разработке улучшенного взаимодействия пользователя записями архивного фонда, их облегченный показ и редактирование.

Список используемых источников

1. <https://kazedu.com/referat/133091/5>
2. <https://ruprogi.ru/software/visual-studio>
3. <https://gb.ru/posts/c_sharp_ides>
4. <https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/overview/?view=netdesktop-6.0>
5. <https://metanit.com/sharp/wpf/11.php>
6. <https://steptosleep.ru/ролевая-модель/#:~:text=Основная%20идея%20ролевой%20модели%20контроля,типов%20их%20активностей%20в%20системе>
7. <https://studopedia.ru/22_29871_neobhodimost-otladki-programmnogo-produkta.html>
8. <https://infopedia.su/4x1ec5.html>
9. <https://sergeygavaga.gitbooks.io/kurs-lektsii-testirovanie-programnogo-obespecheni/content/lektsiya-4-ch3.html>
10. <https://www.retail.ru/rbc/pressreleases/tsentr-povysheniya-kvalifikatsii-lider-organizatsiya-rabochego-mesta-ofisnogo-rabotnika/>
11. <https://xn--d1aux.xn--p1ai/opisanie-rabochego-mesta-programmista-na-predpriyatii/>
12. <https://studopedia.ru/8_107307_osveshchenie-pomeshcheniy-vichislitelnih-tsentrov.html>

Приложение А

Программный код окна MainWindow.xaml.cs

using System.Windows;

using System.Windows.Media;

namespace Archive

{

public partial class MainWindow : Window

{

public Base.Client User;

public MainWindow(Base.Client User)

{

InitializeComponent();

this.User = User;

if(User.roleClient == true)

{

ProfileButton.Visibility = Visibility.Collapsed;

ArchiveDataButton.Visibility = Visibility.Collapsed;

}

BrushConverter bc = new BrushConverter();

ArchiveDataButton.Background = (Brush)bc.ConvertFrom("#FED6BC");

ArchiveFrame.Navigate(new Pages.ArchivePage(User));

}

//Открытие страницы со справочником

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

{

ArchiveFrame.Navigate(new Pages.ArchivePage(User));

BrushConverter bc1 = new BrushConverter();

ProfileButton.Background = (Brush)bc1.ConvertFrom("#FDEED9");

BrushConverter bc2 = new BrushConverter();

ArchiveDataButton.Background = (Brush)bc2.ConvertFrom("#FED6BC");

}

//выход из учетной записи

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

{

new AuthorizationWindow().Show();

Close();

}

//Открытие страницы с личным кабинетом

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

{

BrushConverter bc1 = new BrushConverter();

ArchiveDataButton.Background = (Brush)bc1.ConvertFrom("#FDEED9");

BrushConverter bc2 = new BrushConverter();

ProfileButton.Background = (Brush)bc2.ConvertFrom("#FED6BC");

ArchiveFrame.Navigate(new Pages.ProfilePage(User));

}

}

}

Программный код окна AuthorizationWindow.xaml.cs

using System;

using System.Linq;

using System.Windows;

using System.Windows.Input;

namespace Archive

{

/// <summary>

/// Interaction logic for AuthorizationWindow.xaml

/// </summary>

public partial class AuthorizationWindow : Window

{

public Base.Client User;

private Base.ArciveFundEntities3 DataBase;

public AuthorizationWindow()

{

InitializeComponent();

try

{

DataBase = new Base.ArciveFundEntities3();

}

catch

{

MessageBox.Show("Не удалось подключиться к базе данных. Проверьте настройки подключения приложения.",

"Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

Close();

}

}

//кнопка регистрации

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

{

new RegistrationWindow(DataBase).Show();

Close();

}

//вход в программу

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

{

User = DataBase.Client.SingleOrDefault(U => U.loginClient == LoginText.Text && U.passwordClient == PasswordPasswordBox.Password);

if (User != null)

{

new MainWindow(User).Show();

Close();

}

else

{

MessageBox.Show("Неверно указан логин и/или пароль!", "Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

//кнопка выхода из программы

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

{

if (MessageBox.Show("Вы действительно хотите выйти из программы?", "Внимание", MessageBoxButton.OKCancel, MessageBoxImage.Warning) == MessageBoxResult.OK)

{

Close();

}

}

//показ пароля

private void PasswordButton\_MouseEnter(object sender, MouseEventArgs e)

{

String Password = PasswordPasswordBox.Password;

Visibility Visibility = PasswordPasswordBox.Visibility;

double Width = PasswordPasswordBox.ActualWidth;

PasswordPasswordBox.Password = PasswordTextBox.Text;

PasswordPasswordBox.Visibility = PasswordTextBox.Visibility;

PasswordPasswordBox.Width = PasswordTextBox.Width;

PasswordTextBox.Text = Password;

PasswordTextBox.Visibility = Visibility;

PasswordTextBox.Width = Width;

}

}

}

Программный код окна RegistrationWindow.xaml.cs

using System.Collections.ObjectModel;

using System.Text;

using System.Windows;

namespace Archive

{

public partial class RegistrationWindow : Window

{

private Base.ArciveFundEntities3 Database;

public ObservableCollection<Base.Client> Clients;

public RegistrationWindow(Base.ArciveFundEntities3 Database)

{

InitializeComponent();

this.Database = Database;

Clients = new ObservableCollection<Base.Client>(SourceCore.entities.Client);

}

private bool IsLoginValidation()

{

for (int i = 0; i < Clients.Count; i++)

{

if (Clients[i].loginClient.Equals(LoginTextBox.Text))

{

MessageBox.Show("Имя пользователя занято!");

return true;

}

}

return false;

}

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

{

StringBuilder errors = new StringBuilder();

if (string.IsNullOrEmpty(NameTextBox.Text)) errors.AppendLine("Укажите имя");

if (string.IsNullOrEmpty(numberPhoneTextBox.Text)) errors.AppendLine("Укажите номер");

if (string.IsNullOrEmpty(LoginTextBox.Text)) errors.AppendLine("Укажите логин");

if (string.IsNullOrEmpty(PasswordTextBox.Text)) errors.AppendLine("Укажите пароль");

if (string.IsNullOrEmpty(PasswordTextRepeat.Text)) errors.AppendLine("Подтвердите пароль");

if(errors.Length > 0)

{

MessageBox.Show(errors.ToString());

return;

}

Base.Client User = new Base.Client();

User.nameClient = NameTextBox.Text;

User.numberPhoneClient = numberPhoneTextBox.Text;

User.loginClient = LoginTextBox.Text;

User.passwordClient = PasswordTextBox.Text;

User.roleClient = IdRole.IsChecked.Value;

if(Clients.Count != 0) if (IsLoginValidation()) return;

if(PasswordTextRepeat.Text == PasswordTextBox.Text)

{

Database.Client.Add(User);

Database.SaveChanges();

new AuthorizationWindow().Show();

Close();

}

else MessageBox.Show("Пароли не совпадают!", "Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

}

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

{

new AuthorizationWindow().Show();

Close();

}

}

}

Программный код страницы ArchivePage.xaml.cs

using Microsoft.Win32;

using System;

using System.Collections.ObjectModel;

using System.IO;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media.Imaging;

namespace Archive.Pages

{

public partial class ArchivePage : Page

{

bool isClientData = false;

bool isTree = false;

int DlgMode;

public Base.Client User;

public Base.Data SelectedData;

public ObservableCollection<Base.Data> Data;

public ObservableCollection<Base.Client> Clients;

private ImageConverter converter = new ImageConverter();

private ConverterValue converterValue = new ConverterValue();

private string \_lastBase64;

public ArchivePage(Base.Client User)

{

InitializeComponent();

Clients = new ObservableCollection<Base.Client>(SourceCore.entities.Client);

this.User = User;

if(User.roleClient == true)

{

ClientData.Visibility = Visibility.Collapsed;

}

else

{

AddGroup.Visibility = Visibility.Collapsed;

Add.Visibility = Visibility.Collapsed;

Copy.Visibility = Visibility.Collapsed;

Edit.Visibility = Visibility.Collapsed;

Delete.Visibility = Visibility.Collapsed;

}

UpdateGrid();

TreeViewLoader();

DlgLoad(0, "");

PrintSelectedItem();

DataContext = this;

}

//Генерация дерева

private void TreeViewLoader()

{

treeView1.Items.Clear();

//storageSection

TreeViewItem StorageSectionItem = new TreeViewItem();

StorageSectionItem.Header = "Список по секции хранения";

treeView1.Items.Add(StorageSectionItem);

var storageSectionList = SourceCore.entities.StorageSection.ToList();

for (int i = 0; i < storageSectionList.Count; i++)

{

TreeViewItem ChildItem = new TreeViewItem();

ChildItem.Header = storageSectionList[i].name;

StorageSectionItem.Items.Add(ChildItem);

var listData = Data.ToList();

if (isClientData) listData = Data.Where(item => item.clientId == User.clientId).ToList();

else listData = listData = Data.ToList();

for (int j = 0; j < listData.Count; j++)

{

if(listData[j].stoSecId == storageSectionList[i].id)

{

TreeViewItem SubChildItem = new TreeViewItem();

SubChildItem.Style = (Style)FindResource("TreeViewItem");

SubChildItem.Header = listData[j].name;

SubChildItem.Selected += SubChildItem\_Selected;

ChildItem.Items.Add(SubChildItem);

TreeViewItem idItem = new TreeViewItem();

idItem.Header = listData[j].dataId;

idItem.Visibility = Visibility.Collapsed;

SubChildItem.Items.Add(idItem);

}

}

}

//stoPeriod

TreeViewItem StoPeriodItem = new TreeViewItem();

StoPeriodItem.Header = "Список по периоду хранения";

treeView1.Items.Add(StoPeriodItem);

var storagePeriodList = SourceCore.entities.StoragePeriod.ToList();

for (int i = 0; i < storagePeriodList.Count; i++)

{

TreeViewItem ChildItem = new TreeViewItem();

ChildItem.Header = storagePeriodList[i].name;

StoPeriodItem.Items.Add(ChildItem);

var listData = Data.ToList();

if (isClientData) listData = Data.Where(item => item.clientId == User.clientId).ToList();

else listData = listData = Data.ToList();

for (int j = 0; j < listData.Count; j++)

{

if (listData[j].stoPerId == storagePeriodList[i].id)

{

TreeViewItem SubChildItem = new TreeViewItem();

SubChildItem.Style = (Style)FindResource("TreeViewItem");

SubChildItem.Header = listData[j].name;

SubChildItem.Selected += SubChildItem\_Selected;

ChildItem.Items.Add(SubChildItem);

TreeViewItem idItem = new TreeViewItem();

idItem.Header = listData[j].dataId;

idItem.Visibility = Visibility.Collapsed;

SubChildItem.Items.Add(idItem);

}

}

}

//status

TreeViewItem StatusItem = new TreeViewItem();

StatusItem.Header = "Список по статусу";

treeView1.Items.Add(StatusItem);

var statusList = SourceCore.entities.Status.ToList();

for (int i = 0; i < statusList.Count; i++)

{

TreeViewItem ChildItem = new TreeViewItem();

ChildItem.Header = statusList[i].name;

StatusItem.Items.Add(ChildItem);

var listData = Data.ToList();

if (isClientData) listData = Data.Where(item => item.clientId == User.clientId).ToList();

else listData = listData = Data.ToList();

for (int j = 0; j < listData.Count; j++)

{

if (listData[j].statusId == statusList[i].id)

{

TreeViewItem SubChildItem = new TreeViewItem();

SubChildItem.Style = (Style)FindResource("TreeViewItem");

SubChildItem.Header = listData[j].name;

SubChildItem.Selected += SubChildItem\_Selected;

ChildItem.Items.Add(SubChildItem);

TreeViewItem idItem = new TreeViewItem();

idItem.Header = listData[j].dataId;

idItem.Visibility = Visibility.Collapsed;

SubChildItem.Items.Add(idItem);

}

}

}

}

//Вывод информации об объекте

private void PrintSelectedItem()

{

SelectedData = (Base.Data)ArchiveGrid.SelectedItem;

//data

dataName.Content = SelectedData.name;

dataId.Text = "Номер объекта: " + SelectedData.dataId;

dateStart.Text = "Дата прибытия: " + ((DateTime)SelectedData.dateStart).ToString("dd.MM.yyyy");

description.Text = SelectedData.description;

statusTextBlock.Text = "Статус: " + SelectedData.Status.name;

stoPerTextBlock.Text = "Срок хранения: " + SelectedData.StoragePeriod.name;

stoSecTextBlock.Text = "Секция хранения: " + SelectedData.StorageSection.name;

if (SelectedData.photo == null) dataImage.Source = new BitmapImage(new Uri("/Images/nullphoto.png", UriKind.RelativeOrAbsolute));

else dataImage.Source = converter.ConvertToImage(SelectedData.photo);

//client

if (SelectedData.Client != null)

{

nameClient.Text = "Имя клиента: " + SelectedData.Client.nameClient;

numberPhoneClient.Text = "Номер телефона: " + SelectedData.Client.numberPhoneClient;

}

}

public Base.Data GetData(int id)

{

Base.Data selectedData = null;

for (int i = 0; i < Data.Count; i++)

{

if(Data[i].dataId == id)

{

selectedData = Data[i];

}

}

return selectedData;

}

private void SubChildItem\_Selected(object sender, RoutedEventArgs e)

{

TreeViewItem treeViewItem = (TreeViewItem)sender;

TreeViewItem item = (TreeViewItem)treeViewItem.ItemContainerGenerator.Items.ElementAt(0);

UpdateGrid(GetData((int)item.Header));

PrintSelectedItem();

}

private void ArchiveGridRow\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

PrintSelectedItem();

}

private int GetClient()

{

for (int i = 0; i < Clients.Count; i++)

{

if (clientLogin.Text.Equals(Clients[i].loginClient) && Clients[i].roleClient == false)

{

return Clients[i].clientId;

}

}

MessageBox.Show("Клиента под таким логином не существует или данная операция с этим логином невозможно!");

return -1;

}

//кнопка принятия изменений об объекте

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

{

StringBuilder errors = new StringBuilder();

if (string.IsNullOrEmpty(clientLogin.Text))

errors.AppendLine("Укажите логин клиента");

if (string.IsNullOrEmpty(objectName.Text))

errors.AppendLine("Укажите название объекта");

if (string.IsNullOrEmpty(objectDescription.Text))

errors.AppendLine("Укажите описание объекта");

if (string.IsNullOrEmpty(objectVolume.Text))

errors.AppendLine("Укажите объем объекта");

if (objectstoSecId.SelectedIndex == -1)

errors.AppendLine("Укажите группу");

if (objectstoPerId.SelectedIndex == -1)

errors.AppendLine("Укажите срок хранения");

if (objectStatusId.SelectedIndex == -1)

errors.AppendLine("Укажите статус");

if (errors.Length > 0)

{

MessageBox.Show(errors.ToString());

return;

}

if (GetClient() == -1)

return;

if (DlgMode == 0)

{

var data = new Base.Data();

data.name = objectName.Text;

data.clientId = GetClient();

data.description = objectDescription.Text;

data.dateStart = DataPicker.SelectedDate;

data.volume = int.Parse(objectVolume.Text);

data.photo = \_lastBase64;

data.stoSecId = objectstoSecId.SelectedIndex + 1;

data.stoPerId = objectstoPerId.SelectedIndex + 1;

data.statusId = objectStatusId.SelectedIndex + 1;

SourceCore.entities.Data.Add(data);

SelectedData = data;

}

else

{

var EditData = new Base.Data();

EditData = SourceCore.entities.Data.First(i => i.dataId == SelectedData.dataId);

EditData.name = objectName.Text;

EditData.clientId = GetClient();

EditData.description = objectDescription.Text;

EditData.dateStart = DataPicker.SelectedDate;

EditData.volume = int.Parse(objectVolume.Text);

EditData.photo = \_lastBase64;

EditData.stoSecId = objectstoSecId.SelectedIndex + 1;

EditData.stoPerId = objectstoPerId.SelectedIndex + 1;

EditData.statusId = objectStatusId.SelectedIndex + 1;

}

try

{

SourceCore.entities.SaveChanges();

DlgLoad(0, "");

UpdateGrid(SelectedData);

PrintSelectedItem();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

//кнопка отмены

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

{

DlgLoad(0, "");

UpdateGrid(SelectedData);

}

//кнопка добавления объекта

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

{

DlgLoad(1, "Добавить объект");

DataContext = null;

DlgMode = 0;

}

//кнопка копирования объекта

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

{

if (ArchiveGrid.SelectedItem != null)

{

DlgLoad(1, "Копировать объект");

SelectedData = (Base.Data)ArchiveGrid.SelectedItem;

clientLogin.Text = SelectedData.Client.loginClient;

objectName.Text = SelectedData.name;

objectDescription.Text = SelectedData.description;

DataPicker.Text = SelectedData.dateStart.ToString();

objectVolume.Text = SelectedData.volume.ToString();

if (SelectedData.photo == null) imageData.Source = new BitmapImage(new Uri("/Images/nullphoto.png", UriKind.RelativeOrAbsolute));

else imageData.Source = converter.ConvertToImage(SelectedData.photo);

objectstoSecId.SelectedIndex = (int)SelectedData.stoSecId-1;

objectstoPerId.SelectedIndex = (int)SelectedData.stoPerId-1;

objectStatusId.SelectedIndex = (int)SelectedData.statusId-1;

DlgMode = 0;

}

else

{

MessageBox.Show("Не выбрано ниодной строки!", "Сообщение", MessageBoxButton.OK);

}

}

//кнопка редактирования объекта

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

{

if (ArchiveGrid.SelectedItem != null)

{

DlgLoad(1, "Изменить объект");

SelectedData = (Base.Data)ArchiveGrid.SelectedItem;

clientLogin.Text = SelectedData.Client.loginClient;

objectName.Text = SelectedData.name;

objectDescription.Text = SelectedData.description;

DataPicker.Text = SelectedData.dateStart.ToString();

objectVolume.Text = SelectedData.volume.ToString();

if (SelectedData.photo == null) imageData.Source = new BitmapImage(new Uri("/Images/nullphoto.png", UriKind.RelativeOrAbsolute));

else imageData.Source = converter.ConvertToImage(SelectedData.photo);

objectstoSecId.SelectedIndex = (int)SelectedData.stoSecId-1;

objectstoPerId.SelectedIndex = (int)SelectedData.stoPerId-1;

objectStatusId.SelectedIndex = (int)SelectedData.statusId-1;

}

else

{

MessageBox.Show("Не выбрано ниодной строки!", "Сообщение", MessageBoxButton.OK);

}

}

public void DlgLoad(int mode, string DlgModeContent)

{

switch (mode)

{

case 0:

ListColumnChange.Width = new GridLength(100, GridUnitType.Star);

BookColumnChange.Width = new GridLength(0);

TreeColumnChange.Width = new GridLength(0);

ArchiveGrid.IsHitTestVisible = true;

Add.IsEnabled = true;

Copy.IsEnabled = true;

Edit.IsEnabled = true;

Delete.IsEnabled = true;

DlgMode = -1;

break;

case 1:

ListColumnChange.Width = new GridLength(0);

BookColumnChange.Width = new GridLength(100, GridUnitType.Star);

TreeColumnChange.Width = new GridLength(0);

ArchiveGrid.IsHitTestVisible = false;

actionName.Content = DlgModeContent;

AddCommit.Content = DlgModeContent;

Add.IsEnabled = false;

Copy.IsEnabled = false;

Edit.IsEnabled = false;

Delete.IsEnabled = false;

break;

case 2:

BookColumnChange.Width = new GridLength(0);

TreeColumnChange.Width = new GridLength(30, GridUnitType.Star);

ArchiveGrid.IsHitTestVisible = true;

actionName.Content = DlgModeContent;

AddCommit.Content = DlgModeContent;

Add.IsEnabled = true;

Copy.IsEnabled = true;

Edit.IsEnabled = true;

Delete.IsEnabled = true;

break;

default:

ListColumnChange.Width = new GridLength(100, GridUnitType.Star);

BookColumnChange.Width = new GridLength(0);

TreeColumnChange.Width = new GridLength(0);

ArchiveGrid.IsHitTestVisible = true;

Add.IsEnabled = true;

Copy.IsEnabled = true;

Edit.IsEnabled = true;

Delete.IsEnabled = true;

DlgMode = -1;

break;

}

}

//вывод информации в справочник

public void UpdateGrid(Base.Data data = null)

{

if ((data == null) && (ArchiveGrid.ItemsSource != null))

{

data = (Base.Data)ArchiveGrid.SelectedItem;

}

Data = new ObservableCollection<Base.Data>(SourceCore.entities.Data);

if (!isClientData) ArchiveGrid.ItemsSource = Data;

else ArchiveGrid.ItemsSource = Data.Where(i => i.clientId == User.clientId);

ArchiveGrid.SelectedItem = data ?? Data[0];

objectStatusId.ItemsSource = SourceCore.entities.Status.ToList();

objectstoSecId.ItemsSource = SourceCore.entities.StorageSection.ToList();

objectstoPerId.ItemsSource = SourceCore.entities.StoragePeriod.ToList();

}

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

{

if (isTree == false)

{

FormStackPanel.Visibility = Visibility.Collapsed;

treeView1.Visibility = Visibility.Visible;

isTree = true;

DlgLoad(2, "");

}

else

{

FormStackPanel.Visibility = Visibility.Visible;

treeView1.Visibility = Visibility.Collapsed;

isTree = false;

DlgLoad(0, "");

}

}

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

{

if (isClientData == false)

{

isClientData = true;

UpdateGrid(SelectedData);

TreeViewLoader();

}

else

{

isClientData = false;

UpdateGrid(SelectedData);

TreeViewLoader();

}

}

//кнопка удаления объекта

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

{

if (MessageBox.Show("Удалить запись?", "Внимание", MessageBoxButton.OKCancel, MessageBoxImage.Warning) == MessageBoxResult.OK)

{

try

{

Base.Data DeletingData = (Base.Data)ArchiveGrid.SelectedItem;

if (ArchiveGrid.SelectedIndex < ArchiveGrid.Items.Count - 1)

{

ArchiveGrid.SelectedIndex++;

}

else

{

if (ArchiveGrid.SelectedIndex > 0)

{

ArchiveGrid.SelectedIndex--;

}

}

Base.Data SelectingData = (Base.Data)ArchiveGrid.SelectedItem;

SourceCore.entities.Data.Remove(DeletingData);

SourceCore.entities.SaveChanges();

UpdateGrid(SelectingData);

}

catch

{

MessageBox.Show("Невозможно удалить запись, так как она используется в других справочниках базы данных.", "Предупреждение",

MessageBoxButton.OK, MessageBoxImage.Warning, MessageBoxResult.None);

}

}

}

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

{

FormStackPanel.Visibility = Visibility.Visible;

treeView1.Visibility = Visibility.Collapsed;

}

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

{

if (GroupNameTextBox.Text.Equals("")) return;

switch (SelectGroupComboBox.SelectedIndex)

{

case 0:

var StoragePeriod = new Base.StoragePeriod();

StoragePeriod.name = GroupNameTextBox.Text;

SourceCore.entities.StoragePeriod.Add(StoragePeriod);

break;

case 1:

var Status = new Base.Status();

Status.name = GroupNameTextBox.Text;

SourceCore.entities.Status.Add(Status);

break;

case 2:

var StorageSection = new Base.StorageSection();

StorageSection.name = GroupNameTextBox.Text;

SourceCore.entities.StorageSection.Add(StorageSection);

break;

}

try

{

SourceCore.entities.SaveChanges();

FormStackPanel.Visibility = Visibility.Collapsed;

treeView1.Visibility = Visibility.Visible;

TreeViewLoader();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

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

{

FormStackPanel.Visibility = Visibility.Collapsed;

treeView1.Visibility = Visibility.Visible;

}

//кнопка добавления фотографии

private void photoPath\_PreviewMouseDown(object sender, MouseButtonEventArgs e)

{

OpenFileDialog ofd = new OpenFileDialog();

ofd.Filter = "Фотографии (\*.png)|\*.png";

ofd.Title = "Выбор фотографии";

ofd.InitialDirectory = @"D:\course2\ArchiveFund\Archive\Images";

if (ofd.ShowDialog() == true)

{

if (!File.Exists(ofd.FileName))

{

MessageBox.Show("Файл не существует!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

\_lastBase64 = converter.ConvertToBase64(ofd.FileName);

imageData.Source = converter.ConvertToImage(\_lastBase64);

}

}

}

}

Программный код страницы ProfilePage.xaml.cs

using Microsoft.Win32;

using System;

using System.IO;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

namespace Archive.Pages

{

/// <summary>

/// Interaction logic for ProfilePage.xaml

/// </summary>

public partial class ProfilePage : Page

{

private ImageConverter converter = new ImageConverter();

private string \_lastBase64;

public Base.Client User;

public ProfilePage(Base.Client User)

{

InitializeComponent();

this.User = User;

Update();

}

private Base.Client getClient()

{

var clientList = SourceCore.entities.Client.ToList();

for (int i = 0; i < clientList.Count; i++)

{

if (clientList[i].clientId == User.clientId)

{

return clientList[i];

}

}

MessageBox.Show("Клиента под таким логином не существует или данная операция с этим логином невозможно!");

return null;

}

//вывод информации о клиенте

private void Update()

{

Base.Client client = getClient();

clientName.Text = client.nameClient;

if(client.sexClientId == true)

{

clientSexCombobox.SelectedIndex = 0;

}

else

{

clientSexCombobox.SelectedIndex = 1;

}

if (client.passportClient == null) clientPassport.Text = "Укажите пасспорт";

else clientPassport.Text = client.passportClient;

clientNumberPhone.Text = client.numberPhoneClient;

clientLogin.Text = client.loginClient;

clientPassword.Text = client.passwordClient;

if(client.imageClient != null) clientImage.Source = converter.ConvertToImage(client.imageClient);

}

//проверка логинов

private bool ClientLoginValidation()

{

var clientList = SourceCore.entities.Client.ToList();

for (int i = 0; i < clientList.Count; i++)

{

if (clientLogin.Text.Equals(clientList[i].loginClient))

{

MessageBox.Show("Логин занят!");

return true;

}

}

return false;

}

//кнопка сохранения информации о клиенте

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

{

StringBuilder errors = new StringBuilder();

if (string.IsNullOrEmpty(clientName.Text))

errors.AppendLine("Укажите имя");

if (clientSexCombobox.SelectedIndex == -1)

errors.AppendLine("Укажите пол");

if (string.IsNullOrEmpty(clientPassport.Text))

errors.AppendLine("Укажите пасспорт");

if (string.IsNullOrEmpty(clientNumberPhone.Text))

errors.AppendLine("Укажите номер телефона");

if (string.IsNullOrEmpty(clientLogin.Text))

errors.AppendLine("Укажите логин");

if (string.IsNullOrEmpty(clientPassword.Text))

errors.AppendLine("Укажите пароль");

if (errors.Length > 0)

{

MessageBox.Show(errors.ToString());

return;

}

if(clientLogin.Text != User.loginClient)

{

if (ClientLoginValidation()) return;

}

var EditClient = new Base.Client();

EditClient = SourceCore.entities.Client.First(i => i.clientId == User.clientId);

EditClient.nameClient = clientName.Text;

if (clientSexCombobox.SelectedIndex == 0) EditClient.sexClientId = true;

else EditClient.sexClientId = false;

EditClient.passportClient = clientPassport.Text;

EditClient.numberPhoneClient = clientNumberPhone.Text;

EditClient.loginClient = clientLogin.Text;

EditClient.passwordClient = clientPassword.Text;

if(\_lastBase64 != null) EditClient.imageClient = \_lastBase64;

try

{

SourceCore.entities.SaveChanges();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

//изменение фотографии клиента

private void clientImage\_PreviewMouseDown(object sender, MouseButtonEventArgs e)

{

OpenFileDialog ofd = new OpenFileDialog();

ofd.Filter = "Фотографии (\*.png)|\*.png";

ofd.Title = "Выбор фотографии";

ofd.InitialDirectory = @"D:\course2\ArchiveFund\Archive\Images";

if (ofd.ShowDialog() == true)

{

if (!File.Exists(ofd.FileName))

{

MessageBox.Show("Файл не существует!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

\_lastBase64 = converter.ConvertToBase64(ofd.FileName);

clientImage.Source = converter.ConvertToImage(\_lastBase64);

}

}

}

}

Программный код класса SourceCore.cs

using Archive.Base;

namespace Archive

{

public class SourceCore

{

public static ArciveFundEntities3 entities = new ArciveFundEntities3();

}

}

Программный код класса ImageConverter.cs

using System;

using System.Globalization;

using System.IO;

using System.Windows.Data;

using System.Windows.Media.Imaging;

namespace Archive

{

public class ImageConverter : IValueConverter

{

// base64 -> BitmapImage

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

if (value == null) return null;

string base64 = value as string;

byte[] data = System.Convert.FromBase64String(base64);

BitmapImage bi = new BitmapImage();

bi.BeginInit();

bi.StreamSource = new MemoryStream(data);

bi.EndInit();

return bi;

}

// BitmapImage -> base64

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

if (value == null) return null;

string path = value as string;

if (!File.Exists(path)) return null;

byte[] imageArray = File.ReadAllBytes(path);

string base64 = System.Convert.ToBase64String(imageArray);

return base64;

}

public BitmapImage ConvertToImage(string base64) => Convert(base64, null, null, null) as BitmapImage;

public string ConvertToBase64(string path) => ConvertBack(path, null, null, null) as string;

}

}