Министерство образования и науки Республики Башкортостан

Государственное автономное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и вычислительной техники

|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ  Заместитель директора  по учебной работе  З.З. Курмашева  « » 2025г. |

ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ И РАЗРАБОТКА ПРИЛОЖЕНИЯ

ДЛЯ ВЕДЕНИЯ КАТАЛОГОВ ПРОИЗВЕДЕНИЙ ИСКУССТВ

Пояснительная записка к курсовому проекту

МДК.11.01 Технология разработки и защиты баз данных

|  |  |
| --- | --- |
|  | Руководитель проекта  Р.Ф.Каримова  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025 г. |
|  | Студент гр. 22П-1  Я.А. Петрова  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025 г. |

2025

Министерство образования и науки Республики Башкортостан

Государственное автономное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и вычислительной техники

|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ  Заместитель директора  по учебной работе  З.З. Курмашева  « » 2025 г. |

ЗАДАНИЕ

на курсовой проект студенту дневного отделения, группы 22П-1, специальности 09.02.07 «Информационные системы и программирование».

Фамилия, имя, отчество: Петрова Яна Андреевна

Тема курсового проекта: «Проектирование базы данных и разработка приложения для ведения справочника произведений искусств».

Текст задания:

при выполнении курсового проекта должны быть решены следующие задачи:

1. спроектирована база данных;
2. разработана и реализована структура программы;
3. реализованы функции поиска, сортировки и фильтрации данных по разным критериям, формирования аннотации произведения искусства.

В результате выполнения курсового проекта должны быть представлены:

1. пояснительная записка, состоящая из следующих разделов:

Введение

1 Постановка задачи

2 Экспериментальный раздел

Заключение

Приложения

Список сокращений

Список использованных источников

1. электронный носитель, содержащий разработанный программный продукт;
2. презентация курсового проекта в электронном виде.

Список рекомендуемых источников:

1. **Бенжамин, Л.** Искусство управления данными: как строить эффективные базы данных. — М.: Альпина Паблишер, 2021. — 200 с.
2. Википедия [Электронный ресурс] // Свободная энциклопедия. – Режим доступа: <http://ru.wikipedia.org/wiki/>, свободный
3. **Федорова, Г.Н.** Основы проектирования баз данных: учебник для студентов учреждений среднего профессионального образования. — М.: Издательский центр «Академия», 2021. — 256 с.
4. Федорова, Г.Н. Разработка и администрирование и защита баз данных: учебник для студ. учреждений сред. проф. образования. –М.: Издательский центр «Академия», 2018.- 288с.
5. **Червоненко, А.Ю.** Проектирование и разработка баз данных: теоретические и практические аспекты. — СПб.: Питер, 2019. — 312 с.

Задание к выполнению получил «29» января 2025 г.

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Петрова Яна Андреевна

Срок окончания «25» июня 2025 г.

Руководитель курсового проекта Р.Ф. Каримова

Задание рассмотрено на заседании цикловой комиссии информатики

протокол № 4 от «11» января 2025 г.

Председатель цикловой комиссии информатики О.В.Фатхулова

Министерство образования и науки Республики Башкортостан

Государственное автономное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и вычислительной техники

ЗАКЛЮЧЕНИЕ

на курсовой проект

Студент Петрова Яна Андреевна

Группа 22П-1

Специальность 09.02.07 Информационные системы и программирование

Тема Проектирование базы данных и разработка приложения для ведения справочника произведений искусств

Объем курсового проекта:

количество листов пояснительной записки

количество листов графической части

Заключение о степени соответствия заданию на курсовое проектирование

Характеристика качеств, проявленных студентом при работе над проектом: самостоятельность, дисциплинированность, умение планировать работу и пользоваться литературным материалом и т.д.

Положительные стороны курсового проекта

Недостатки курсового проекта

Характеристика общетехнической и специальной подготовки студента

Заключение и предлагаемая оценка за курсовой проект

Руководитель курсового проекта \_\_\_\_\_\_\_\_\_Р.Ф. Каримова

«\_\_\_\_» 2025 г.

Подпись

АННОТАЦИЯ

Пояснительная записка к курсовому проекту содержит постановку и программу решения задачи «Проектирование базы данных и разработка приложения для ведения справочника произведений искусств».

Программа ArtGallery.exe написана на языке C# в среде программирования Visual Studio 2022 с использованием системы управления базой данных MySQL, предназначена для работы в операционной системе MS Windows 10 и выше, отлажена на данных контрольного примера.

СОДЕРЖАНИЕ

|  |  |
| --- | --- |
|  | лист |
| Введение | 4 |
| 1 Постановка задачи | 6 |
| 1.1 Описание предметной области | 6 |
| 1.2 Описание входной информации | 9 |
| 1.3 Описание выходной информации | 9 |
| 1.4 Концептуальное моделирование | 11 |
| 1.5 Логическое моделирование | 12 |
| 1.6 Описание структуры базы данных | 14 |
| 1.7 Контрольный пример | 16 |
| 1.8 Общие требования к программному продукту | 16 |
| 2 Экспериментальный раздел | 18 |
| 2.1 Описание программы | 18 |
| 2.2 Протокол тестирования программного продукта | 29 |
| 2.3 Руководство пользователя | 34 |
| 2.4 Меры по обеспечению защиты информации | 45 |
| Заключение | 46 |
| Приложения | 47 |
| Список сокращений | 103 |
| Список использованных источников | 104 |

ВВЕДЕНИЕ

Произведения искусства - это результат творческой деятельности человека, который выражает его эмоции, идеи, мировоззрение или эстетическое восприятие. Искусство охватывает множество форм, таких как живопись, скульптура, музыка, литература, театр, кино и многие другие.

Создание каталога произведений искусства становится важным шагом в условиях современного общества, где информация о культурном наследии и художественных ценностях становится ключевым ресурсом. Наличие структурированной информации способствует обучению и популяризации искусства, позволяя использовать каталог в учебных заведениях и культурных центрах. Систематизация данных о произведениях искусства делает их более доступными для широкой аудитории, включая исследователей, студентов, художников и любителей искусства.

Актуальность разработки данной информационной системы обусловлена растущим интересом к искусству и необходимостью упрощения доступа к информации о произведениях. В условиях современного общества, где информация становится ключевым ресурсом, создание такой системы представляется не только полезным, но и крайне необходимым шагом для поддержки культурного развития и сохранения художественного наследия.

Основной целью данного проекта является ознакомление пользователей с произведениями искусства и создание информационной системы, которая позволит хранить и управлять данными о произведениях искусства, будет обеспечивать удобный доступ к информации для пользователей.

Альтернативные программы, которые знакомят пользователей с произведениями искусства: The Metropolitan Museum of Art и Artstor.

The Metropolitan Museum of Art - это музей, расположенный в Нью-Йорке, чья коллекция охватывает более 5000 лет истории искусства со всего мира. Музей также имеет свою онлайн-платформу, которая хранит информацию о происхождениях, истории и технике создания произведении искусства, их изображения высокого качества и виртуальные туры. Изображения можно скачивать в высоком разрешении. Также есть кураторские статьи и эссе. Из недостатков, только то, что на сайте будет информация по имеющийся в музее коллекции и сложная навигация сайта.

Artstor - это цифровая библиотека, предоставляющая доступ к огромной коллекции изображений высокого качества, охватывающей различные дисциплины, включая искусство, архитектуру, археологию, антропологию и многое другое. Изображения в Artstor поступают из надежных источников, таких как музеи, архивы и университеты. Это гарантирует точность и достоверность информации. Artstor предоставляет пользователям инструменты для сравнения изображений, создания презентаций и организации коллекций. Из недостатков же это платная подписка, ограничение на использование изображений, сложный интерфейс и больше предназначен не для широкой публики, а для учащихся в образовательных учреждениях.

Мой проект превосходит Artstor тем, что он является полностью бесплатным, предоставляя доступ к информации о произведениях искусства без необходимости оплачивать подписку. Кроме того, в базе данных моего проекта будут представлены произведения искусства со всего мира, что делает его универсальной и обширной платформой для изучения искусства. Пользователи смогут легко находить и просматривать информацию по различным критериям, а удобный интерфейс обеспечит комфортное взаимодействие с системой.

1 Постановка задачи

* 1. Описание предметной области

Требуется разработать информационную систему для хранения и управления информацией о произведениях искусства. Она будет обеспечивать удобный доступ к данным, позволяя пользователям искать и извлекать информацию по различным критериям, таким как автор, год создания, техника исполнения и т.д. База данных должна поддерживать функции поиска, фильтрации и сортировки, позволяя пользователям находить нужные произведения и получать полную информацию о них.

Каждое произведение уникально, но они могут иметь схожие черты и характеристики.

Чтобы создать произведение искусства, нужно выбрать его технику исполнения:

* ID\_perfomance\_technique;
* Вид искусства (скульптура, живопись, графика);
* Техника исполнения.

Они могут быть написаны простым карандашом или же сделаны из прочного мрамора. Если это скульптура, то техникой исполнения не может быть карандаш, также как картина не может быть вылита медью.

Основная информация, касающаяся жанра картины, будет прописана в признаке жанр:

* ID\_genre;
* Разновидность жанра;
* Направление;
* Жанр.

Это может быть пейзаж моря, портретная скульптура или карикатура, изображённая в стиле барокко.

Чтобы хранить произведения искусства в целостности и сохранности, есть определённые места их хранения. Их основные характеристики:

* ID\_storage\_location;
* Страна;
* Тип места хранения;
* Название;
* Доступность (публична, сокрыта или является частной коллекцией).

Музей, галерея, частная коллекция или же архивы государства, к которым нужен особый доступ. Данная характеристика может иметь значение: «Неизвестно».

Чтобы создать произведение искусства, нужен его создатель, или же автор:

* ID\_author;
* Идентификатор автора;
* ФИО;
* Дата рождения;
* Дата смерти;
* Страна происхождения;
* Нация;
* Пол;
* Интересные факты.

Место хранения, жанр, автор и техника исполнения. Совокупность этих качеств даёт нам общее представление о произведениях искусства. Конкретный экземпляр же это совокупность следующих качеств:

* ID\_Art;
* Название;
* Год создания;
* Страна создания;
* ID\_author;
* ID\_storage\_location;
* ID\_genre;
* ID\_perfomance\_technique;
* Аннотация.

Чтобы добавлять и редактировать работы, нужно войти в систему с помощью администратора:

* ID\_admin;
* Login;
* Password.

С данной информационной системой должны работать следующие роли:

* Пользователь;
* Администратор.

Пользователь сможет заходить в систему только как гость, просматривать каталог произведений, сортировать их и использовать выборку по разным критериям, таким как жанр, место хранения, автор и другое.

Вход администратора будет осуществляться через логин и пароль.

Пользователю будет дана возможность поиска произведения искусства по разным критериям:

* ФИО автора;
* Название картины;
* Техника исполнения.

Фильтровать данные можно по данным критериям:

* Год создания;
* Страна создания;
* Тип места хранения;
* Название места хранения;
* Вид искусства;
* Жанр;
* Посещаемость.

Также ему будет дана возможность просматривать полную информацию о произведениях искусства, полученную с помощью поиска. Данная информация может быть отправлена на почту пользователя в виде файла, который может быть скачен.

Пользователь сможет сортировать данные по двум критериям:

* Год создания;
* Посещаемость.

Посещаемость будет вычисляться по количеству посещений данной работы.

Администратор сможет:

* Добавлять новые произведения искусства, авторов, техники исполнения, жанры и места хранения.
* Удалять некорректные данные о произведениях искусства, технике исполнения, жанре, месте хранения и об авторов.
* Редактировать данные о произведениях, авторов, технике исполнения, жанрах и местах хранения.

1.2 Описание входной информации

Входной информацией для выполнения задачи являются справочники: произведения искусства, администраторы, поиск и каталог произведений.

1.3 Описание выходной информации

Выходной информацией является результат поиска, а также каталог произведений искусств и файл, содержащий результат поиска, выданный в виде pdf-файла.

Описание выходных документов представлено в таблице 1.3.1.

Таблица 1.3.1 – Описание выходных документов

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Наименование документа (шифр) | Периодичность выдачи документа | Кол-во экз. | Куда передаются | Поля сортировки | Поля группировки | Итоги |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Каталог произведений искусств | По мере необходимости | 1 | Клиенту  Администратору | - | - | - |
| Pdf-файл с результатом поиска | По мере необходимости | 1 | Клиенту Администратору | - | - | Файл |

Шаблоны выходных документов представлены на рисунках 1.3.1 – 1.3.2.



Рисунок 1.3.1 – Шаблон выходного документа «Каталог произведений искусств»

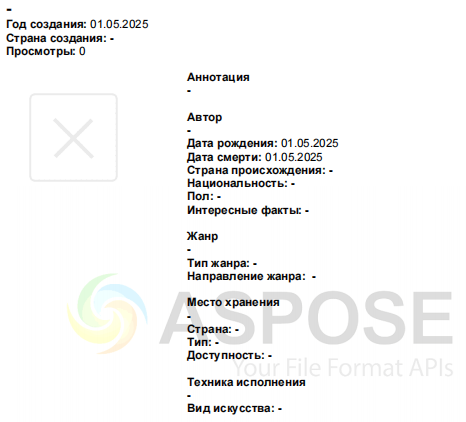


Рисунок 1.3.2 – Шаблон выходного документа «Pdf-файл с результатом поиска»

1.4. Концептуальное моделирование

Концептуальная модель базы данных - это некая наглядная диаграмма, нарисованная в принятых обозначениях и подробно показывающая связь между объектами и их характеристиками. Создаётся концептуальная модель для дальнейшего проектирования базы данных и перевод её, например, в реляционную базу данных. На концептуальной модели в визуально удобном виде прописываются связи между объектами данных и их характеристиками.

В концептуальной модели есть принятые обозначения элементов. Сущность или объект обозначать прямоугольником, отношения обозначать ромбом, атрибуты объектов, обозначаются овалом. Если сущность связана с отношением, то их связь обозначается прямой линией со стрелкой. Необязательная связь обозначается пунктирной линией. Мощная связь обозначается двойной линией.

Концептуальная модель базы данных представлена в рисунке 1.4.1

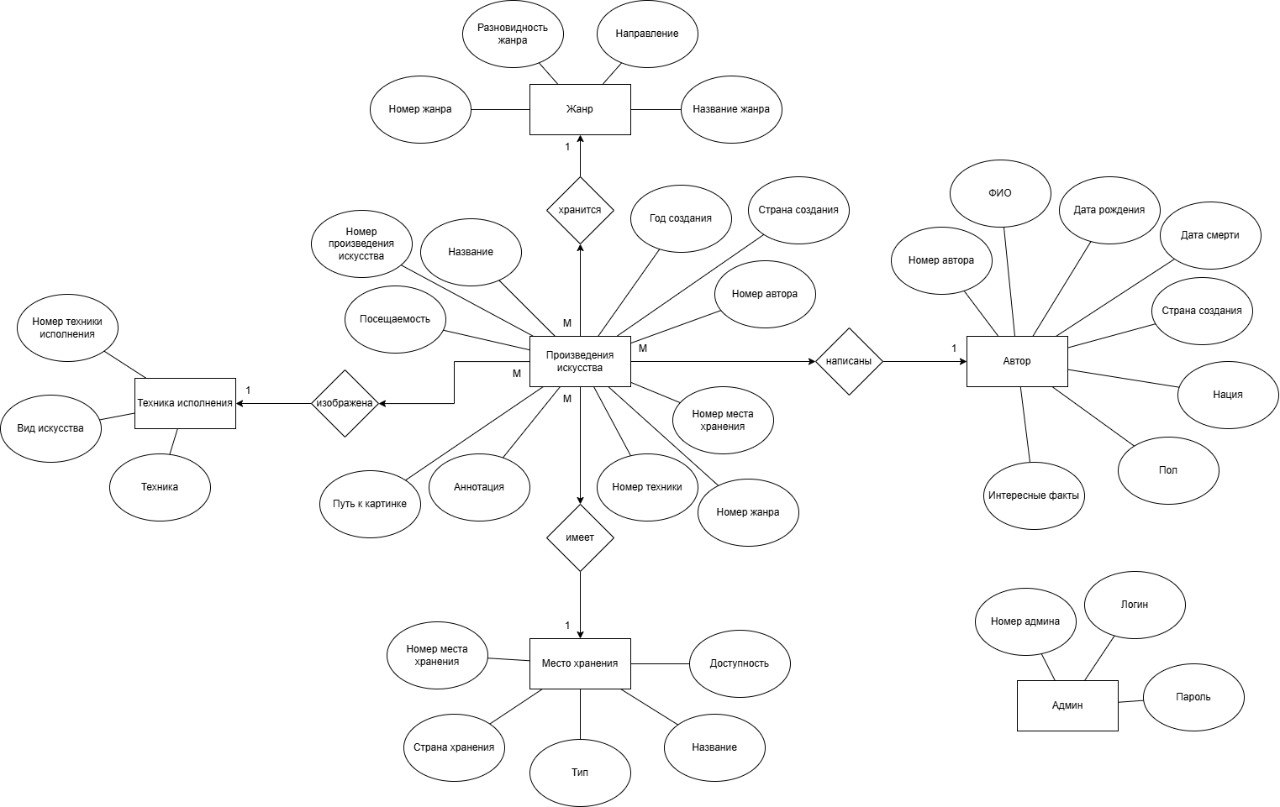


Рисунок 1.4.1 – Концептуальная модель БД

1.5. Логическое моделирование

При логическом моделировании происходит окончательное определение структуры данных, определяются ограничения, накладываемые на эти данные, целью которых является обеспечить целостность данных. Наиболее распространённой моделью данных является реляционная модель. В этой модели данных каждая сущность представляется в виде таблицы.

Логическое моделирование заключается в переходе от концептуальной модели к взаимосвязанным таблицам. Этот переход состоит из следующих шагов:

1. Преобразование сущностей:

- каждая простая сущность становится таблицей.

- каждый атрибут становится столбцом таблицы.

- уникальный идентификатор сущности становится ключом таблицы.

2. Преобразование связи:

- сущности, связанные обязательной связью один к одному можно объединить в одну таблицу.

- связи типа один к одному возможные и связи типа один ко многим реализуются путем переноса ключевых атрибутов таблиц, соответствующих сущностей, стоящих со стороны один в таблице соответствующих сущностей, стоящих со стороны многие.

- связи типа многие ко многим реализуются при помощи промежуточных таблиц, содержащих ключевые атрибуты связываемых таблиц в качестве внешних ключей.

Схема данных – это структура базы данных, описанная на формальном языке, поддерживаемом СУБД (системой управления базы данных). В реляционных базах данных схема определяет таблицы, поля в каждой таблице и ограничения целостности, такие как первичный и внешний ключи.

Схема данных представлена на рисунке 1.5.1.

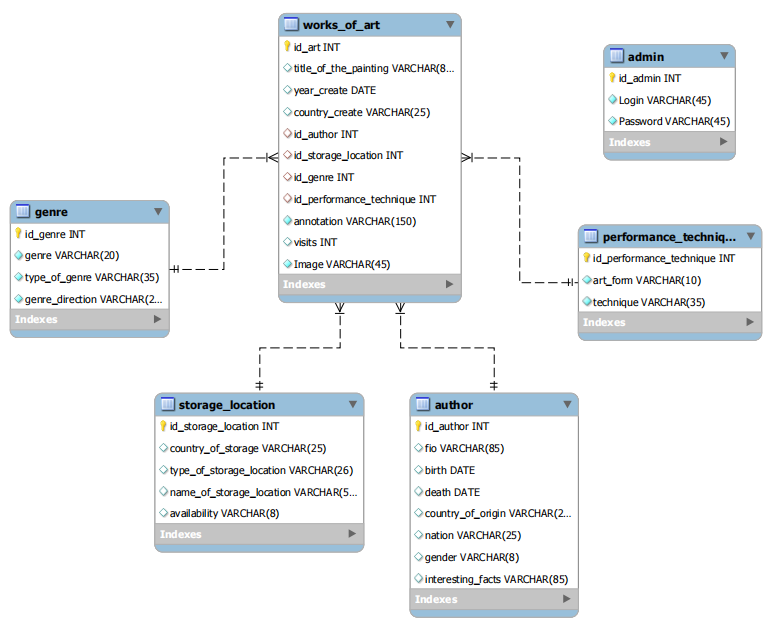


Рисунок 1.5.1 – Схема БД

1.6. Описание структуры базы данных

Описание структуры базы данных представлено в таблицах 1.6.1 – 1.6.7.

Таблица 1.6.1- Performance\_technique(Список техник исполнений)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, длина | Примечания |
| Номер техники | id\_performance\_technique | INT | первичный ключ |
| Вид искусства | art\_form | VARCHAR(10) | обязательное поле |
| Техника исполнения | technique | VARCHAR(35) | обязательное поле |

Таблица 1.6.2 - Author(Список авторов)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, длина | Примечания |
| 1 | 2 | 3 | 4 |
| Номер автора | id\_author | INT | первичный ключ |

Продолжение таблицы 1.6.2

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| ФИО автора | fio | VARCHAR(85) | необязательное поле |
| Дата рождения | birth | DATE | необязательное поле |
| Дата смерти | death | DATE | необязательное поле |
| Страна, в которой родился | country\_of\_origin | VARCHAR(25) | необязательное поле |
| Нация | nation | VARCHAR(25) | необязательное поле |
| Пол | gender | VARCHAR(8) | необязательное поле |
| Интересные факты | interesting\_facts | VARCHAR(180) | необязательное поле |

Таблица 1.6.3 - Storage\_location(Список мест хранения)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, длина | Примечания |
| Номер места хранения | id\_storage\_location | INT | первичный ключ |
| Страна, где хранится | country\_of\_storage | VARCHAR(25) | необязательное поле |
| Тип места хранения | type\_of\_storage\_location | VARCHAR(26) | необязательное поле |
| Название места хранения | name\_of\_storage\_location | VARCHAR(55) | необязательное поле |
| Доступность | availability | VARCHAR(20) | необязательное поле |

Таблица 1.6.4 - Genre (Список жанров)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, длина | Примечания |
| Номер жанра | id\_genre | INT | первичный ключ |
| Тип жанра | type\_of\_genre | VARCHAR(35) | необязательное поле |
| Жанр | genre | VARCHAR(65) | необязательное поле |
| Направление жанра | genre\_direction | VARCHAR(25) | необязательное поле |

Таблица 1.6.5 - Works\_of\_art(Список произведений искусств)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, длина | Примечания |
| 1 | 2 | 3 | 4 |
| Номер произведения искусства | id\_art | INT | первичный ключ |
| Название произведения искусства | title\_of\_the\_painting | VARCHAR(85) | необязательное поле |
| Год создания | year\_create | DATE | необязательное поле |
| Страна, в которой написана картина | country\_create | VARCHAR(25) | необязательное поле |

Продолжение таблицы 1.6.5

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Номер автора | id\_author | INT | вторичный ключ |
| Номер места хранения | id\_storage\_location | INT | вторичный ключ |
| Номер жанра | id\_genre | INT | вторичный ключ |
| Номер техники исполнения | id\_performance\_technique | INT | вторичный ключ |
| Аннотация | annotation | VARCHAR(350) | обязательное поле |
| Количество посещений | visits | VARCHAR(45) | по умолчанию 0 |
| Путь к картинке(короткий) | image | VARCHAR(45) | необязательное поле |

Таблица 1.6.6 – Admin (Список администраторов)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, длина | Примечания |
| Номер админа | id\_admin | INT | первичный ключ |
| Логин | login | VARCHAR(45) | обязательное поле |
| Пароль | password | VARCHAR(45) | обязательное поле |

1.7. Контрольный пример

Контрольный пример является ручным подсчётом задачи. По составленной программе обрабатываются исходные данные контрольного примера. Полученные результаты сравниваются с известными результатами контрольного примера. При несовпадении результатов производится поиск, исправление ошибок, и снова производится выполнение программы.

Входные данные для контрольных примеров показаны в приложении А

Выходные данные для контрольных примеров показаны в приложении Б.

1.8 Общие требования к программному продукту

Пользователи должны иметь базовые навыки пользования персональным компьютером.

Минимальные требования к техническому обеспечению программного продукта следующие:

* ОС:  Windows 7 Service Pack1/ Windows 7 64Bit Service Pack1/ Windows 8.1 64Bit / Windows 10 64Bit / Mac OS;
* процессор: 2.40 Ггц (четырехъядерный) / AMD Phenom 9850 (четырехъядерный) @ 2.5 Ггц;
* оперативная память: 512МБ (Win 7/Win8 и выше);
* видеокарта: NVIDIA 9800 GT c 512 МБ видеопамяти/ AMD HD 4870 с 1 Гб видеопамяти (DX 9, 10, 10.1);
* жесткий диск: 10 гигабайт свободного места;
* Microsoft DirectX версия 9.0c;

Функциональные возможности программного продукта:

* приложение должно формировать и отображать выходные данные пользователю;
* в приложении должен быть обеспечен просмотр таблиц (справочников) базы данных с возможность добавления, редактирования, удаления данных.

Требования к надежности:

* приложение должно обрабатывать ошибочные действия пользователя и сообщать ему об этом;
* приложение должно обеспечивать контроль входной и выходной информации.

Требования к информационной и программной совместимости: обеспечить работу приложения с таблицами СУБД MySQL.

1. Экспериментальный раздел

2.1 Описание программы

Разработанное приложение «**ArtGallery.exe»** представляет собой информационную систему для учета и каталогизации произведений искусства, реализованную на языке C# в среде программирования Visual Studio 2022 с использованием СУБД MySQL. Программа имеет модульную схему.

Модульная схема представлена на рисунке 2.1.1.

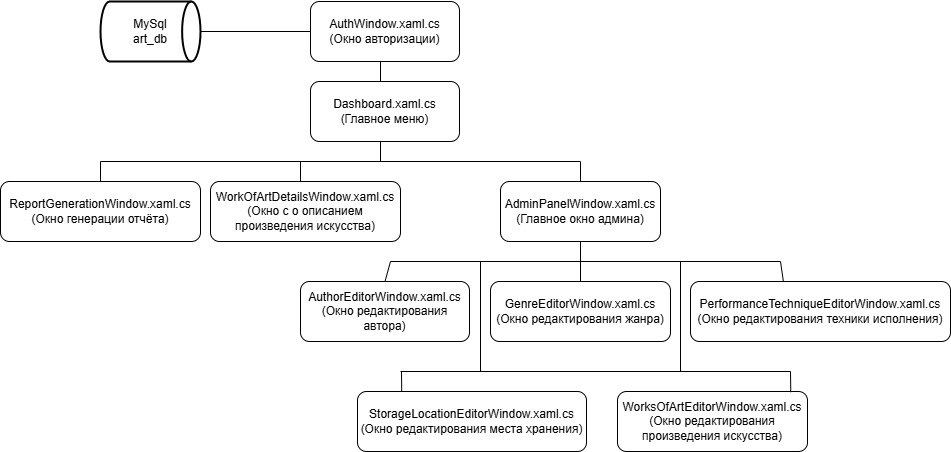


Рисунок 2.1. – Модульная схема

В таблице 2.1.1 описывается каждый модуль перечисляются все процедуры, входящие в модуль, описываются все действия, выполняемые в каждой процедуре модуля.

Таблица 2.1.1 – Описание процедур

|  |  |
| --- | --- |
| Процедуры | Описание процедуры |
| 1 | 2 |
| AuthWindow.xaml.cs «Окно авторизации» | |
| private void OnWindowKeyUp(object sender, System.Windows.Input.KeyEventArgs e) | При нажатии на кнопку «Enter» передает данные и вызывает функцию OnAuthButtonClick() |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| private void OnAuthButtonClick(object sender, RoutedEventArgs e) | При нажатии на кнопку «Войти» происходит проверка логина и пароля на соответствие и существование в базе данных, открывает страницу «Заглавная – Art Gallery» |
| private void OnContinueAsGuestButtonClick(object sender, RoutedEventArgs e) | При нажатии на кнопку «Гостевой режим» открывает страницу «Заглавная – Art Gallery» |
| DashboardWindow.xaml.cs «Окно каталога произведений искусств» | |
| private void OnWindowLoaded(object sender, RoutedEventArgs e) | Инициализирует элементы управления выбором, вызывает InitializeSelectionControls() иOnSelectionPropertiesChange() |
| private void InitializeSelectionControls() | Инициализирует значения и обработчики событий для элементов управления выбором |
| private void InitializeSelectionControlsValues() | Инициализирует значения элементов управления фильтрацией и сортировкой |
| private void InitializeSelectionControlsEventListeners() | Инициализирует обработчики событий для элементов управления фильтрацией и сортировкой |
| private void InitializeFilteringSelectionControlsValues() | Инициализирует значения элементов управления фильтрацией |
| private void InitializeFilteringSelectionControlsEventListeners() | Инициализирует обработчики событий для элементов управления фильтрацией |
| private void InitializeSortingSelectionControlsValues() | Инициализирует значения элементов управления сортировкой |
| private void InitializeSortingSelectionControlsEventListeners() | Инициализирует обработчики событий для элементов управления сортировкой |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| private void OnSelectionPropertiesChange(object? sender, dynamic? e) | Вызывается при изменении свойств выбора, вызывает CompleteSelection() и UpdateListView() |
| private List<WorksOfArt> CompleteSelection() | Выполняет полную выборку данных с учетом поиска, фильтрации и сортировки |
| private List<WorksOfArt> CompleteSearch(List<WorksOfArt> selectedRecords) | Выполняет поиск по данным |
| private List<WorksOfArt> CompleteFiltering(List<WorksOfArt> selectedRecords) | Выполняет фильтрацию данных |
| private List<WorksOfArt> CompleteSorting(List<WorksOfArt> selectedRecords) | Выполняет сортировку данных |
| private void UpdateListView(List<WorksOfArt> selectedRecords) | Обновляет источник данных для ListView |
| private void OnResetButtonClick(object sender, RoutedEventArgs e) | Обрабатывает нажатие кнопки "Сбросить", сбрасывает значения элементов управления фильтрацией и сортировкой |
| private void ResetFilteringSelectionControlsSelectedValues() | Сбрасывает значения элементов управления фильтрацией |
| private void ResetSortingSelectionControlsSelectedValues() | Сбрасывает значения элементов управления сортировкой |
| private void OnRefreshButtonClick(object sender, RoutedEventArgs e) | Обрабатывает нажатие кнопки "Обновить", обновляет источник данных и вызывает InitializeSelectionControlsValues() и OnSelectionPropertiesChange() |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| private void OnWorksOfArtListViewMouseDoubleClick(object sender, System.Windows.Input.MouseButtonEventArgs e) | Обрабатывает двойной клик по элементу ListView, открывает окно с подробностями выбранного произведения искусства |
| private void OnWorksOfArtListViewContextMenuItemEditButtonClick(object sender, RoutedEventArgs e) | Обрабатывает нажатие на кнопку "Редактировать" в контекстном меню ListView, открывает окно редактирования выбранного произведения искусства |
| private async void OnWorksOfArtListViewContextMenuItemDeleteButtonClickAsync(object sender, RoutedEventArgs e) | Обрабатывает нажатие на кнопку "Удалить" в контекстном меню ListView, удаляет выбранное произведение искусства |
| private void OnOpenAdminPanelButtonClick(object sender, RoutedEventArgs e) | Обрабатывает нажатие на кнопку "Открыть панель администратора", открывает окно панели администратора |
| private void OnSaveCurrentDataButtonClick(object sender, RoutedEventArgs e) | Обрабатывает нажатие на кнопку "Сохранить текущие данные", открывает окно генерации отчета |
| private void OnExitButtonClick(object sender, RoutedEventArgs e) | Обрабатывает нажатие на кнопку "Выход", открывает окно авторизации |
| ReportGenerationWindow.xaml.cs «Окно генерации отчёта» | |
| private void OnWindowLoaded(object sender, RoutedEventArgs e) | Инициализирует значение поля OutputPathTextBox по умолчанию, добавляет обработчики событий изменения текста для OutputPathTextBox и EmailTextBox |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| private void OnOutputPathTextBoxTextChanged(object sender, System.Windows.Controls.TextChangedEventArgs e) | Обновляет текст блока IsPathValidTextBlock в зависимости от корректности введенного пути |
| private void OnEmailTextBoxTextChanged(object sender, System.Windows.Controls.TextChangedEventArgs e) | Обновляет текст блока IsEmailValidTextBlock в зависимости от корректности введенного email-адреса |
| private void OnBrowseButtonClick(object sender, RoutedEventArgs e) | Открывает диалоговое окно выбора файла и обновляет значение OutputPathTextBox |
| private async void OnGenerateReportButtonClickAsync(object sender, RoutedEventArgs e) | Вызывает BeginDocumentGenerationAsync() для генерации отчета, если введенные параметры корректны |
| private async Task BeginDocumentGenerationAsync(string outputPath, bool showNotifications = true) | Генерирует отчет в указанном пути, отображая соответствующие уведомления |
| private async void OnSendMailMessageButtonClickAsync(object sender, RoutedEventArgs e) | Вызывает BeginPreparingTempFileForMessageSeindingAsync() и BeginMailMessageSendingAsync() для отправки отчета по email, если введенные параметры корректны |
| private async Task<string> BeginPreparingTempFileForMessageSeindingAsync(string recipientEmail) | Генерирует временный файл с отчетом для отправки по email |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| private async Task BeginMailMessageSendingAsync(string tempFileName, bool showNotifications = true) | Отправляет email с прикрепленным отчетом, отображая соответствующие уведомления |
| private bool IsTargetPathCorrect() | Проверяет корректность введенного пути для сохранения отчета |
| private bool IsTargetEmailValid() | Проверяет корректность введенного email-адреса |
| private MailMessage GenerateMessage() | Генерирует объект MailMessage с необходимыми параметрами для отправки email |
| private static SmtpClient GenerateSmtpClient() | Генерирует объект SmtpClient с необходимыми параметрами для отправки email |
| WorkOfArtDetailsWindow.xaml.cs «Окно с детальным описанием произведения искусства» | |
| private void OnWindowLoaded(object sender, RoutedEventArgs e) | Устанавливает заголовок окна и вызывает UpdateWorkOfArtVisits() |
| private void UpdateWorkOfArtVisits() | Обновляет количество просмотров произведения искусства в базе данных |
| private void OnCloseThisWindowClick(object sender, RoutedEventArgs e) | Обрабатывает нажатие на кнопку закрытия окна, закрывает окно |
| AdminPanelWindow.xaml.cs «Основное окно админа» | |
| private void OnWindowLoaded(object sender, RoutedEventArgs e) | Инициализирует обработчики событий и привязки списков |
| private void InitializeEventHandlers() | Инициализирует обработчики событий изменения текста в полях поиска |
| private void InitializeListViewsBindings() | Инициализирует привязки списков к моделям данных |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| private void UpdateStorageLocationsListView(List<StorageLocation> locations) | Обновляет список хранилищ в списке просмотра |
| private void UpdatePerformanceTechniquesListView(List<PerformanceTechnique> techniques) | Обновляет список техник исполнения в списке просмотра |
| private void UpdateGenresListView(List<Genre> genres) | Обновляет список жанров в списке просмотра |
| private void UpdateAuthorsListView(List<Author> authors) | Обновляет список авторов в списке просмотра |
| private void UpdateWorksOfArtListView(List<WorksOfArt> worksOfArt) | Обновляет список произведений искусства в списке просмотра |
| private void OnSearchElementTextChanged(object sender, TextChangedEventArgs e) | Обрабатывает изменение текста в полях поиска, вызывая соответствующие методы поиска |
| private List<StorageLocation> SearchForStorageLocation(string search) | Выполняет поиск хранилищ по введенному тексту |
| private List<PerformanceTechnique> SearchForPerformanceTechnique(string search) | Выполняет поиск техник исполнения по введенному тексту |
| private List<Genre> SearchForGenre(string search) | Выполняет поиск жанров по введенному тексту |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| private List<Author> SearchForAuthor(string search) | Выполняет поиск авторов по введенному тексту |
| private List<WorksOfArt> SearchForWorkOfArt(string search) | Выполняет поиск произведений искусства по введенному тексту |
| private void OnCreateRecordButtonClick(object sender, RoutedEventArgs e) | Обрабатывает нажатие на кнопку создания новой записи, вызывая соответствующий редактор |
| private static bool CreateNewRecord(string tag) | Создает новую запись в зависимости от типа, отображая соответствующее окно редактора |
| private void OnEditRecordButtonClick(object sender, RoutedEventArgs e) | Обрабатывает нажатие на кнопку редактирования записи, вызывая соответствующий редактор |
| private bool EditRecord(string tag) | Редактирует запись в зависимости от типа, отображая соответствующее окно редактора |
| private void OnDeleteRecordButtonClick(object sender, RoutedEventArgs e) | Обрабатывает нажатие на кнопку удаления записи, вызывая соответствующий метод удаления |
| private bool DeleteRecord(string tag) | Удаляет запись в зависимости от типа, сохраняя изменения в базе данных |
| private void UpdateTargetListViewByItsTag(string tag) | Обновляет соответствующий список просмотра после изменения данных |
| AuthorEditorWindow.xaml.cs «Окно настройки автора» | |
| private async void OnSaveButtonClickAsync(object sender, RoutedEventArgs e) | Обрабатывает нажатие на кнопку сохранения, валидирует модель и сохраняет изменения в базе данных |
| private void OnExitButtonClick(object sender, RoutedEventArgs e) | Обрабатывает нажатие на кнопку выхода, закрывает окно |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| private static List<string> GetModelErrors(Author model) | Проверяет модель данных Author на наличие ошибок и возвращает список ошибок |
| GenreEditorWindow.xaml.cs «Окно настройки жанров» | |
| private async void OnSaveButtonClickAsync(object sender, RoutedEventArgs e) | Обрабатывает нажатие на кнопку сохранения, валидирует модель и сохраняет изменения в базе данных |
| private void OnExitButtonClick(object sender, RoutedEventArgs e) | Обрабатывает нажатие на кнопку выхода, закрывает окно |
| private static List<string> GetModelErrors(Genre model) | Проверяет модель данных Genre на наличие ошибок и возвращает список ошибок |
| PerformanceTechniqueEditorWindow.xaml.cs «Окно настройки техники исполнения» | |
| private async void OnSaveButtonClickAsync(object sender, RoutedEventArgs e) | Обрабатывает нажатие на кнопку сохранения, валидирует модель и сохраняет изменения в базе данных |
| private void OnExitButtonClick(object sender, RoutedEventArgs e) | Обрабатывает нажатие на кнопку выхода, закрывает окно |
| private static List<string> GetModelErrors(PerformanceTechnique model) | Проверяет модель данных PerformanceTechnique на наличие ошибок и возвращает список ошибок |
| StorageLocationEditorWindow.xaml.cs «Окно настройки места хранения» | |
| private async void OnSaveButtonClickAsync(object sender, RoutedEventArgs e) | Обрабатывает нажатие на кнопку сохранения, валидирует модель и сохраняет изменения в базе данных |
| private void OnExitButtonClick1(object sender, RoutedEventArgs e) | Обрабатывает нажатие на кнопку выхода, закрывает окно |
| private static List<string> GetModelErrors(StorageLocation model) | Проверяет модель данных StorageLocation на наличие ошибок и возвращает список ошибок |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| WorksOfArtEditorWindow.xaml.cs «Окно настройки произведения искусства» | |
| private void OnWindowLoaded(object sender, RoutedEventArgs e) | Обрабатывает событие загрузки окна, устанавливает значение даты создания произведения искусства |
| private async void OnSaveButtonClickAsync(object sender, RoutedEventArgs e) | Обрабатывает нажатие на кнопку сохранения, валидирует модель и сохраняет изменения в базе данных |
| private void OnExitButtonClick(object sender, RoutedEventArgs e) | Обрабатывает нажатие на кнопку выхода, закрывает окно |
| private static List<string> GetModelErrors(WorksOfArt model) | Проверяет модель данных WorksOfArt на наличие ошибок и возвращает список ошибок |
| DashboardWindowModel.cs «Логика отображения данных для пользователя» | |
| public void UpdateSourceList() | Обновляет списки сущностей (техники исполнения, места хранения, жанры, произведения искусства) из БД |
| public Visibility AdminControlsVisibilityBinding(Admin admin) | Возвращает видимость элементов управления администратора в зависимости от наличия объекта Admin |
| AdminPanelWindowModel.cs «Логика отображения данных для админа» | |
| public void UpdateLists() | Вызывает методы для обновления списков сущностей (произведения искусства, авторы, жанры, техники исполнения, места хранения) |
| public void UpdateWorkOfArts() | Обновляет список произведений искусства из базы данных |
| public void UpdateAuthors() | Обновляет список авторов из базы данных |
| public void UpdateGenres() | Обновляет список жанров из базы данных |
| public void UpdatePerformanceTechniques() | Обновляет список техник исполнения из базы данных |
| public void UpdateStorageLocations() | Обновляет список мест хранения из базы данных |
| WorksOfArtEditorWindowModel.cs «Логика редактирования произведения искусства» | |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| 1 | 2 |
| private void UpdateLists() | Обновляет списки сущностей (места хранения, техники исполнения, жанры, авторы) из базы данных |
| ReportController.cs «Логика генерации отчёта» | |
| public async Task GenerateReportAsync() | Асинхронный метод, который генерирует отчет в формате PDF на основе списка произведений искусства. Создает новый документ Document и DocumentBuilder для построения отчета, итерируется по списку произведений искусства и добавляет информацию о каждом произведении в документ, включая заголовок, ключ-значение строки с информацией, таблицу с изображением и дополнительной информацией, и сохраняет документ в указанном пути в формате PDF |
| private static void AddTitle(DocumentBuilder builder, string titleText) | Добавляет заголовок с указанным текстом в документ |
| private static void AddHeading(DocumentBuilder builder, string headingText) | Добавляет заголовок раздела с указанным текстом в документ |
| private static void AddKeyValueLine(DocumentBuilder builder, string key, string value) | Добавляет строку с ключом и значением в документ |
| private static void AddContentToRightCell(DocumentBuilder builder, WorksOfArt workOfArt) | Заполняет правую ячейку таблицы дополнительной информацией о произведении искусства |
| private static async Task InsertImageAsync(DocumentBuilder builder, string imageUrl, string fallbackImagePath) | Асинхронно вставляет изображение произведения искусства в документ. Если основное изображение недоступно, используется заменитель |

* 1. Протокол тестирования программного продукта

В ходе тестирования программного продукта на корректных и некорректных данных не было обнаружено ошибок, которые влияли бы на работу самого программного продукта и всей системы.

Данный программный продукт удовлетворяет всем предъявленным требованиям, имеет комфортный интерфейс и интуитивно понятный функционал, исключает появления системных ошибок.

В таблицах 2.2.1 – 2.2.5 представлены протоколы тестирования.

Таблица 2.2.1 – Тестирование авторизации пользователя с некорректными данными

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 2 |
| Дата теста | 24.06.2025 |
| Приоритет тестирования (*Низкий/Средний/Высокий*) | Высокий |
| Заголовок/название теста | Тестирование авторизации пользователя с некорректными данными |
| Резюме испытания | Отображение корректного отображения ошибки при попытке авторизации с несовпадающими паролями |
| Этапы теста | 1 ввод логина;  2 ввод пароля;  3 нажатие на кнопку «Вход». |
| Тестовые данные | 1 пароль «Yana»;  2 логин «qwkdy12». |
| Ожидаемый результат | Отображение сообщения «Учётная запись не найдена» |
| Фактический результат | Сообщение отобразилось, авторизация не произошла |

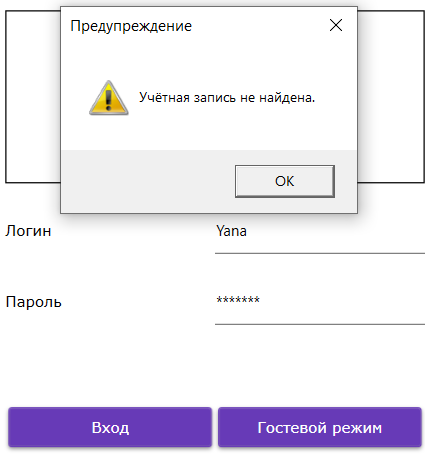


Рисунок 2.2.1 – Результат тестирования авторизации с некорректными данными

Таблица 2.2.2 – Тестирование авторизации пользователя с корректными данными

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 2 |
| Дата теста | 24.06.2025 |
| Приоритет тестирования (*Низкий/Средний/Высокий*) | Высокий |
| Заголовок/название теста | Тестирование авторизации пользователя с корректными данными |
| Резюме испытания | Проверка успешной авторизации при вводе корректных данных |
| Этапы теста | 1 ввод логина;  2 ввод пароля;  3 нажатие на кнопку «Вход». |

Продолжение таблицы 2.2.2

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 2 |
| Тестовые данные | 1 пароль «yanina»;  2 логин «qwU72\*». |
| Ожидаемый результат | Отображение сообщения «Успешный вход (yanina)» |
| Фактический результат | Сообщение отобразилось, авторизация произошла |

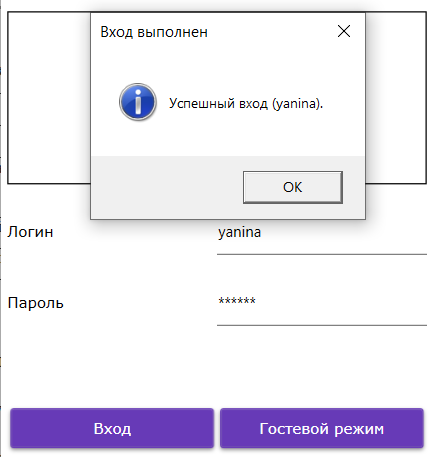


Рисунок 2.2.2 – Результат тестирования авторизации пользователя с корректными данными

В протоколе тестирования отражаются:

− тестирование на корректных данных;

− тестирование на некорректных данных;

− тестировании продукта на данных контрольного примера.

Тестирование на генерацию и сохранение отчёта при корректных данных, ожидаемое сообщение «Отчёт успешно сгенерирован по пути: C:\Users\Яна\Desktop\Report13.pdf » (рисунок 2.2.3).

Таблица 2.2.3 – Тестирование сохранения и генерации отчёта с корректными данными

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 2 |
| Дата теста | 24.06.2025 |
| Приоритет тестирования (*Низкий/Средний/Высокий*) | Высокий |
| Заголовок/название теста | Тестирование генерации и сохранения отчёта с корректными данными |
| Резюме испытания | Проверка успешной генерации и сохранении отчёта при вводе корректных данных |
| Этапы теста | 1 ввод пути;  2 нажатие на кнопку «Сгенерировать отчёт». |
| Тестовые данные | 1 Путь «C:\Users\Яна\Desktop\Report13.pdf». |
| Ожидаемый результат | Отображение сообщения «Отчёт успешно сгенерирован по пути: C:\Users\Яна\Desktop\Report13.pdf» |
| Фактический результат | Сообщение отобразилось, сохранение и генерация отчёта прошли успешно |

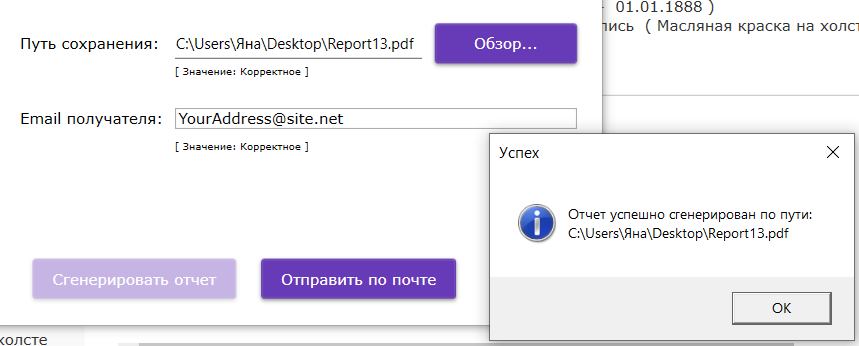


Рисунок 2.2.3 – Сообщение об успешном сохранении и генерации отчёта

Тестирование на генерацию и сохранение отчёта при некорректных данных, ожидаемое сообщение «Введенные параметры некорректны» (рисунок 2.2.4).

Таблица 2.2.4 – Тестирование сохранения и генерации отчёта с корректными данными

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 2 |
| Дата теста | 24.06.2025 |
| Приоритет тестирования (*Низкий/Средний/Высокий*) | Высокий |
| Заголовок/название теста | Тестирование генерации и сохранения отчёта с некорректными данными |
| Резюме испытания | Отображение корректного отображения ошибки при попытке сохранения с некорректными данными |
| Этапы теста | 1 ввод пути;  2 нажатие на кнопку «Сгенерировать отчёт». |
| Тестовые данные | 1 Путь «C:\Users\Яна\Desktop\Report13.pdf». |
| Ожидаемый результат | Отображение сообщения «Введенные параметры некорректны» |
| Фактический результат | Сообщение отобразилось, сохранение и генерация отчёта не произошли |

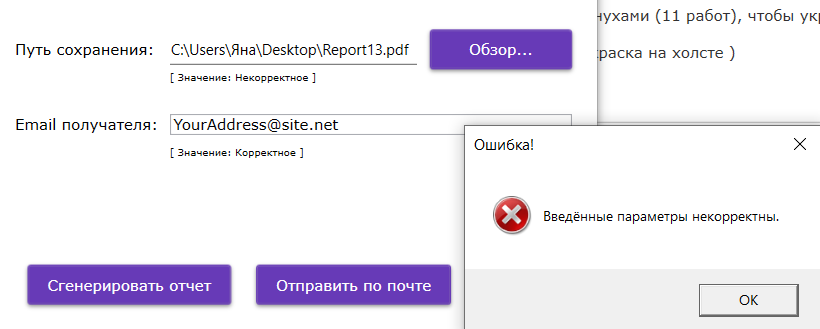


Рисунок 2.2.4 – Результат попытки сохранения и генерации отчёта при некорректных данных

Таблица 2.2.5 – Тестирование на удаление картины

|  |  |
| --- | --- |
| Поле | Описание |
| 1 | 2 |
| Дата теста | 24.06.2025 |
| Приоритет тестирования (*Низкий/Средний/Высокий*) | Высокий |
| Заголовок/название теста | Тестирование на удаление картины |
| Резюме испытания | Отображение сообщения, подтверждающее то, что мы уверены в своём выборе удаления картины |
| Этапы теста | 1 выбор картины;  2 нажатие на кнопку «Удалить».  3 нажать на кнопку «Да» |
| Тестовые данные | 1 Картина из списка |
| Ожидаемый результат | Отображение сообщения «Удаление необратимо. Вы уверены?» |
| Фактический результат | Сообщение отобразилось, удаление произошло успешно |

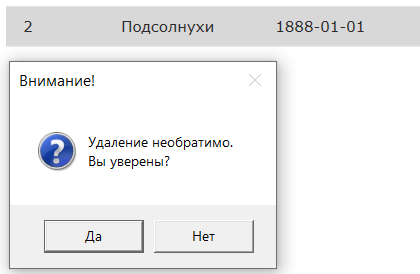


Рисунок 2.2.5 – Тестирование на удаление картины

* 1. Руководство пользователя

Назначение системы

Программа «ArtGallery.exe» представляет собой специализированное программное обеспечение, разработанное для комплексного учета и систематизации информации о произведениях искусства. Система предназначена для музеев, галерей, частных коллекционеров и образовательных учреждений, позволяя эффективно организовывать каталоги художественных произведений.

Основная цель приложения - создание единого цифрового пространства для хранения и обработки данных о картинах, скульптурах и других видах искусства с возможностью быстрого поиска и анализа информации.

Условия применения системы

Программное обеспечение разрабатывается для персональной вычислительной техники со следующими характеристиками:

* Microsoft Windows 7 / 8 / 10;
* процессор 1 ГГц;
* 128 МБ ОЗУ;
* 55 МБ свободного пространства на диске;
* разрешение экрана монитора не менее 1920 × 1080;
* дополнительное оборудование.

Программа «ArtGallery» предназначена для пользователей, имеющих как минимум первоначальные навыки работы с графической операционной системой, на которой будет запускаться данная программа.

Подготовка системы к работе

При запуске программы появляется страница входа. На рисунке 2.3.1 изображено окно авторизации.



Рисунок 2.3.1 – Окно «Аутентификация – Art Gallery»

При нажатии на кнопку «Гостевой режим» открывается главное меню, на котором расположен вывод произведений искусств и их основной информации. Также имеются кнопки «Сохранить», при нажатии которой открывается окно для успешного сохранения или отправки по почте отчёта по результату поиска произведений искусств, и «Выйти», которая отправляет нас на предыдущее окно «Аутентификация». На рисунке 2.3.2 изображено главное окно.

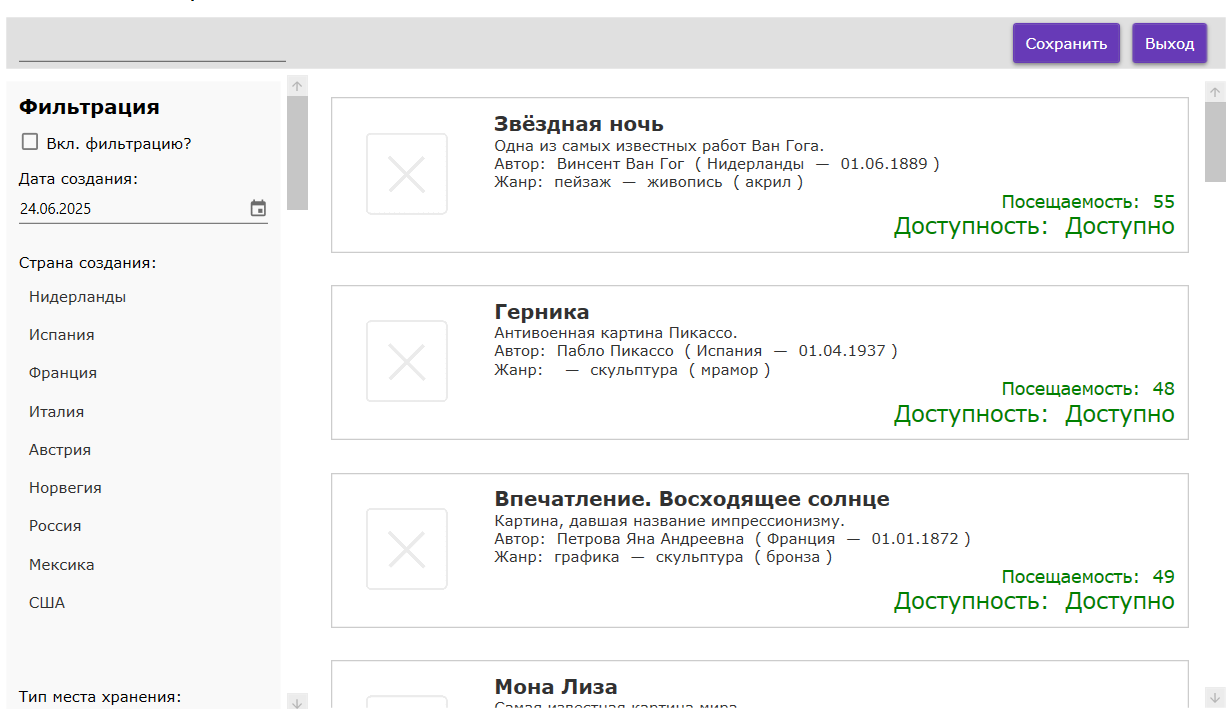


Рисунок 2.3.2 – Окно «Заглавная – Art Gallery»

При нажатии на кнопку «Обзор...» открывается проводник, в котором мы можем выбрать место для сохранения отчёта, путь которого автоматически появляется в нашем поле «Путь сохранения».

При нажатии на кнопку «Сгенерировать отчёт» у нас автоматически генерируется и сохраняется отчёт по указанному пути.

При верно введенной почте и нажатии на кнопку «Отправить по почте» наш отчёт будет отправлен в виде pdf-файла. На рисунке 2.3.3 изображено окно генерации отчёта.

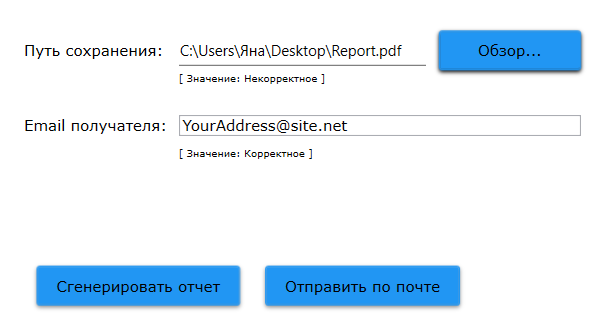


Рисунок 2.3.3 – Окно «Генерация отчёта – Art Gallery»

Найти произведение искусства можно, вписав его автора, название или технику исполнения в поисковике. На рисунке 2.3.4 изображен результат поиска.

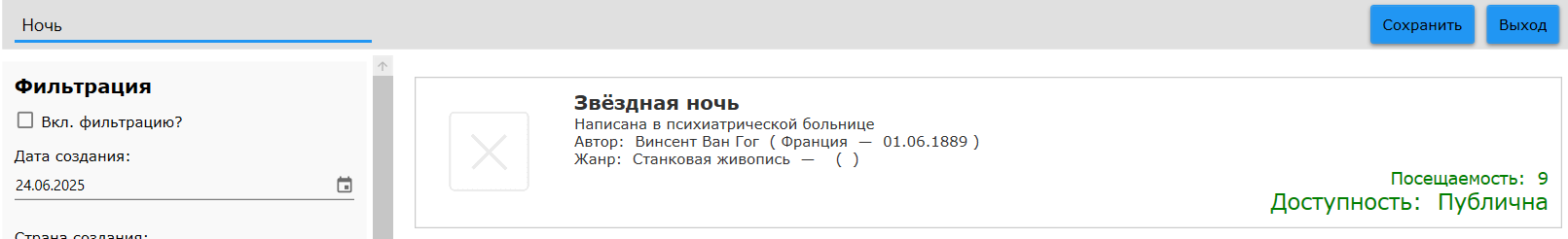


Рисунок 2.3.4 – Результат поиска «Ночь»

При отмеченной кнопке «Включить фильтрацию?» будет происходить выборка данных по выбранным критериям. На рисунке 2.3.5 изображен результат фильтрации.

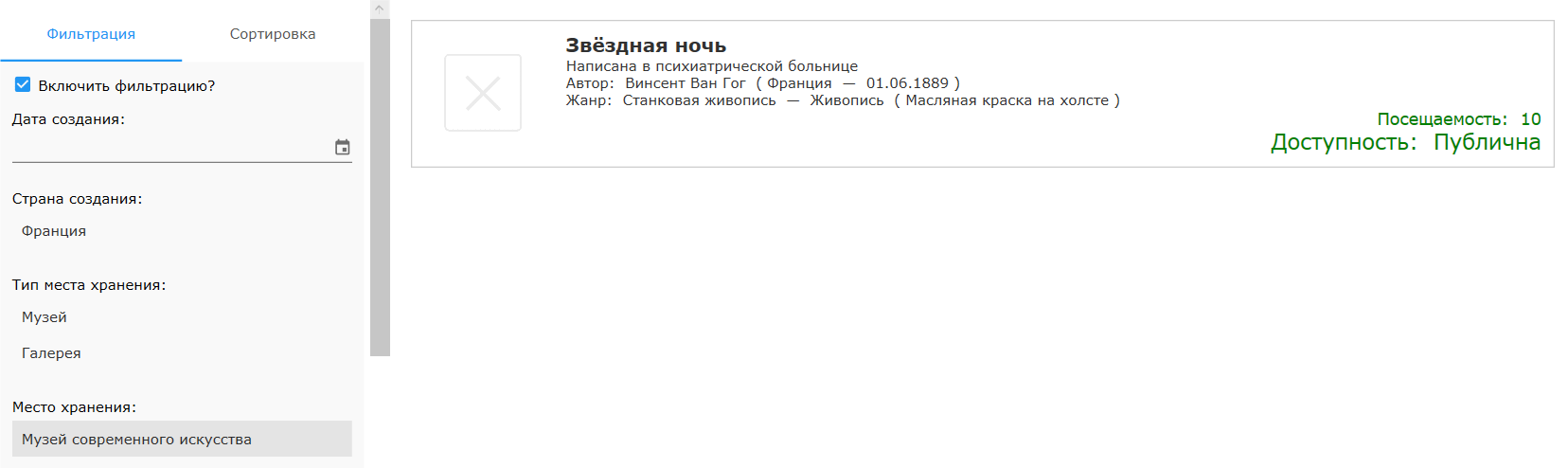


Рисунок 2.3.5 – Результат фильтрации

При отмеченной кнопке «Включить сортировку?» будет происходить выборка данных по выбранным критериям. На рисунке 2.3.6 изображен результат сортировки.

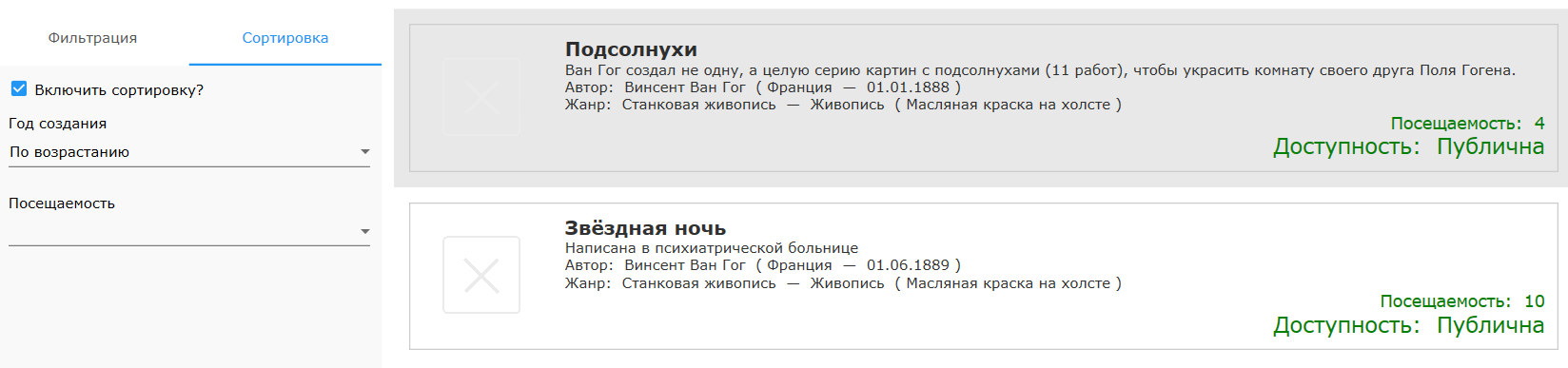


Рисунок 2.3.6 – Результат сортировки

При нажатии кнопки «Сбросить» происходит полный сброс всех параметров фильтрации и сортировки к начальным значениям. При нажатии кнопки «Обновить» происходит полная перезагрузка данных из базы и сброс фильтров и сортировки. Результаты нажатия этих кнопок представлены на рисунках 2.3.7-2.3.8.

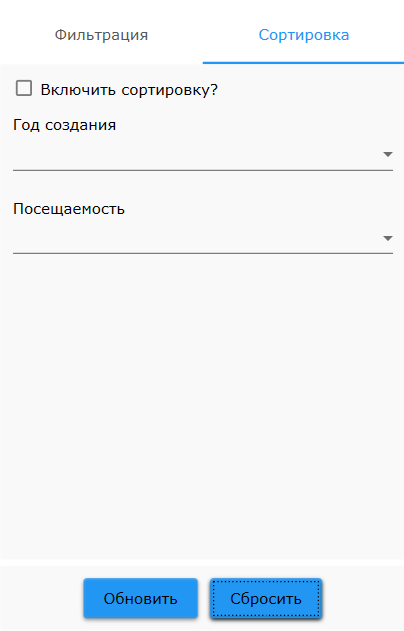


Рисунок 2.3.7 – Результат нажатия кнопки «Сбросить»

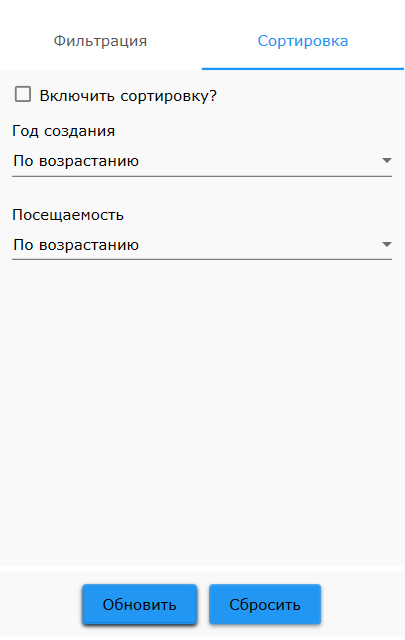


Рисунок 2.3.8 – Результат нажатия кнопки «Обновить»

При двойном нажатии на произведение искусства открывается окно с основной информацией о нем. Данное окно показано на рисунке 2.3.9.

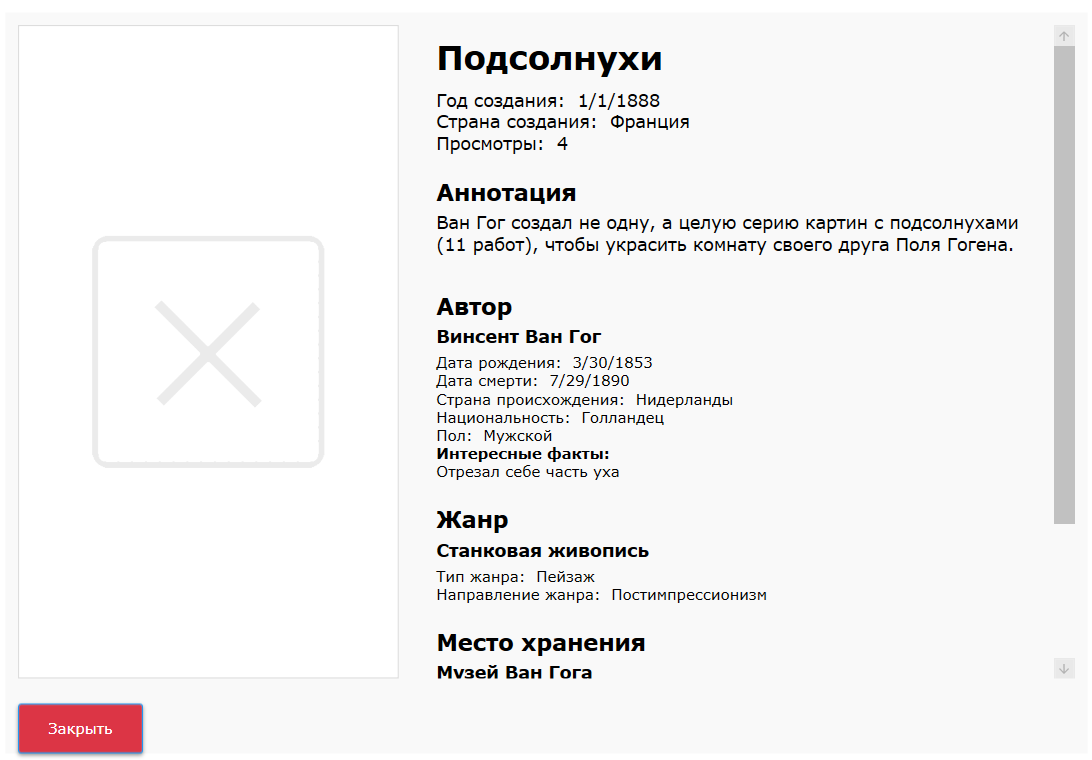


Рисунок 2.3.9 –Окно с основной информацией о картине

При нажатии кнопки «Выход» мы возвращаемся на окно аутентификации.

Далее необходимо ввести корректные данные для поля «Логин» и «Пароль». После успешной авторизации выводится сообщение «Успешный вход (Логин)». На рисунке 2.3.10 изображен вход в программу с данными менеджера.

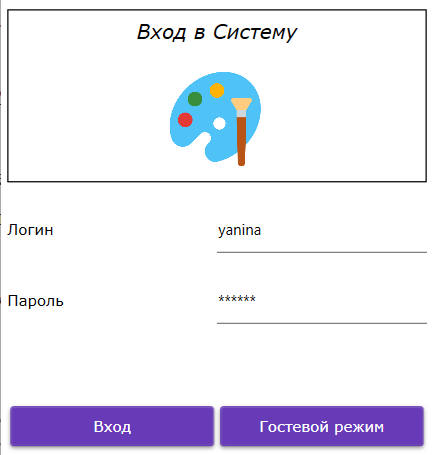


Рисунок 2.3.10 – Ввод корректных данных для авторизации

После успешной авторизации нам открывается окно администратор, которое показано на рисунке 2.3.11.

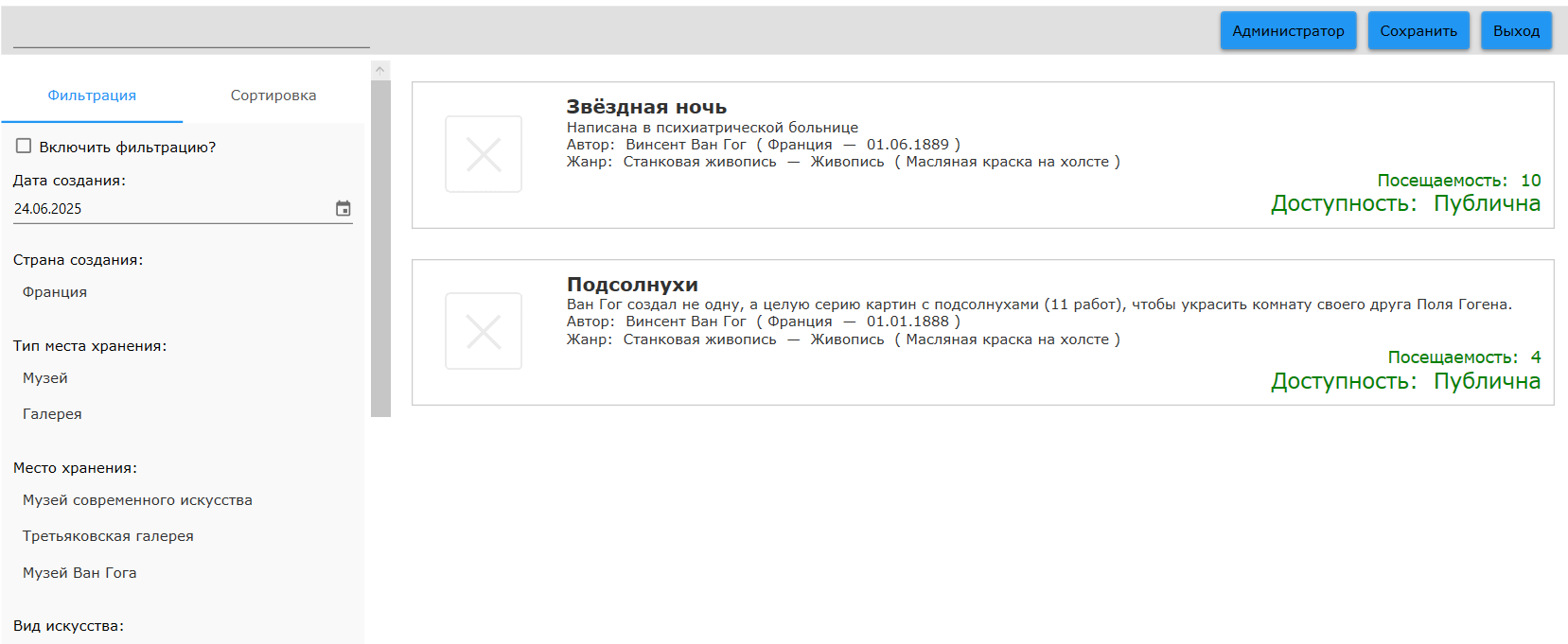


Рисунок 2.3.11 – Окно администратора

При нажатии на правую кнопку мыши нам открывается контекстное меню, где мы можем отредактировать или же удалить произведение искусства. Контекстное меню показано на рисунке 2.3.12.

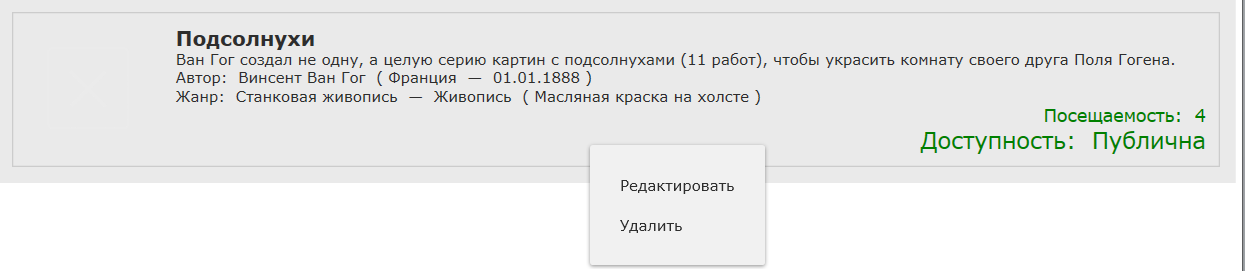


Рисунок 2.3.12 – Контекстное меню произведения искусства

При нажатии на кнопку «Удалить» происходит удаление картины из БД. При нажатии на кнопку «Редактировать» открывается окно, с помощью которого мы можем изменить основные данные картины. Данное окно показано на рисунке 2.3.13.

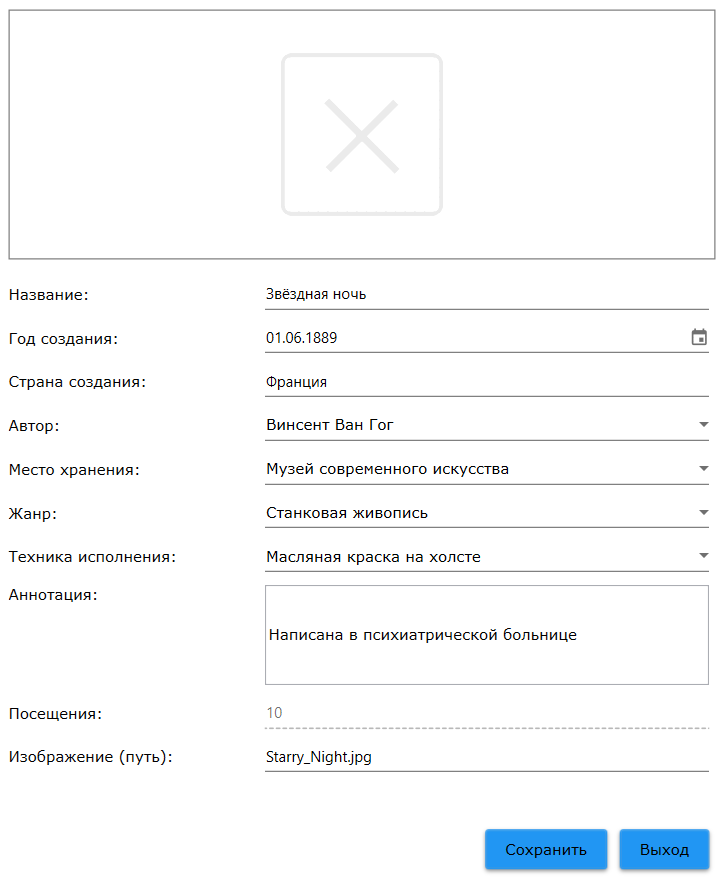


Рисунок 2.3.13 – Меню редактирования картины

При нажатии кнопки сохранить измененные данные сохраняются в БД. При нажатии кнопки «Выход» данное окно закрывается.

При нажатии на кнопку «Администратор» открывается окно, содержащее данные о всех справочниках с их возможностью редактирования, удаления и создания. Для более удобного поиска данных предусмотрен поисковик. Данное окно изображено на рисунке 2.3.14.

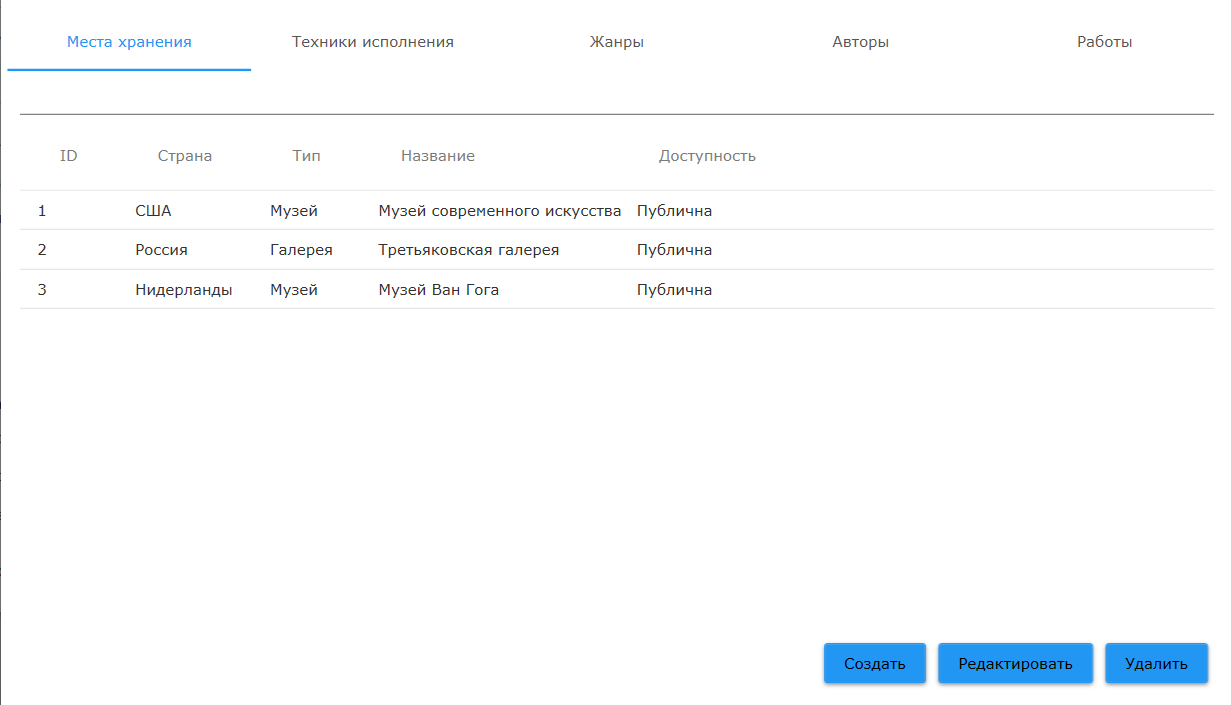


Рисунок 2.3.15 – Окно с данными справочников

При нажатии на кнопку «Создать» открывается окно для создания и сохранения в БД новых данных о месте хранения (рисунок 2.3.16).

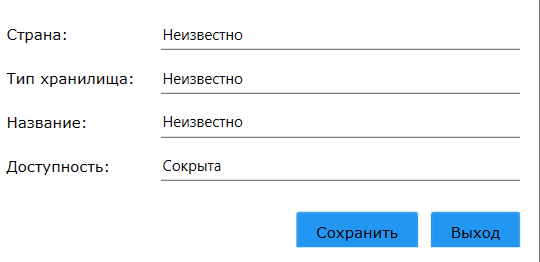


Рисунок 2.3.16 – Окно создания и редактирования места хранения

При выборе конкретного места хранения можно отредактировать или же удалить его. При его редактировании открывается тоже окно, но уже с заполненными данными. Для удаления нужно лишь подтвердить своё решение (рисунок 2.3.17).

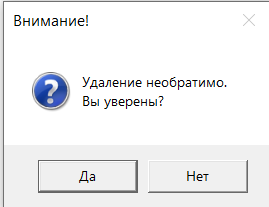


Рисунок 2.3.17 – Подтверждение удаления

Для других справочников процесс удаления и редактирования идентичен этому. Единственное изменение это сами окна редактирования, которые показаны на рисунках 2.3.18 – 2.3.22.

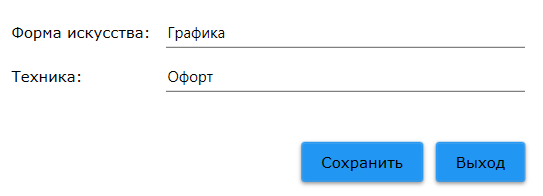


Рисунок 2.3.18 – Окно создания и редактирования техники исполнения

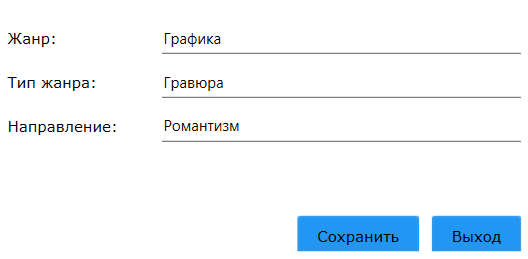


Рисунок 2.3.19 – Окно создания и редактирования жанров

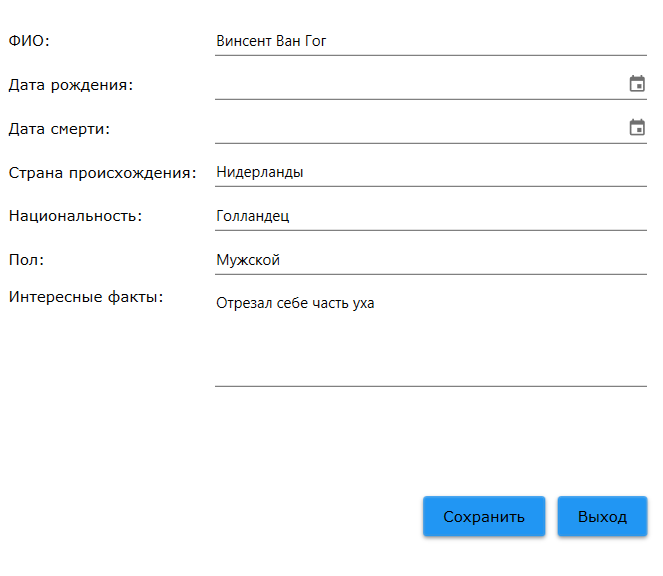


Рисунок 2.3.19 – Окно создания и редактирования автора

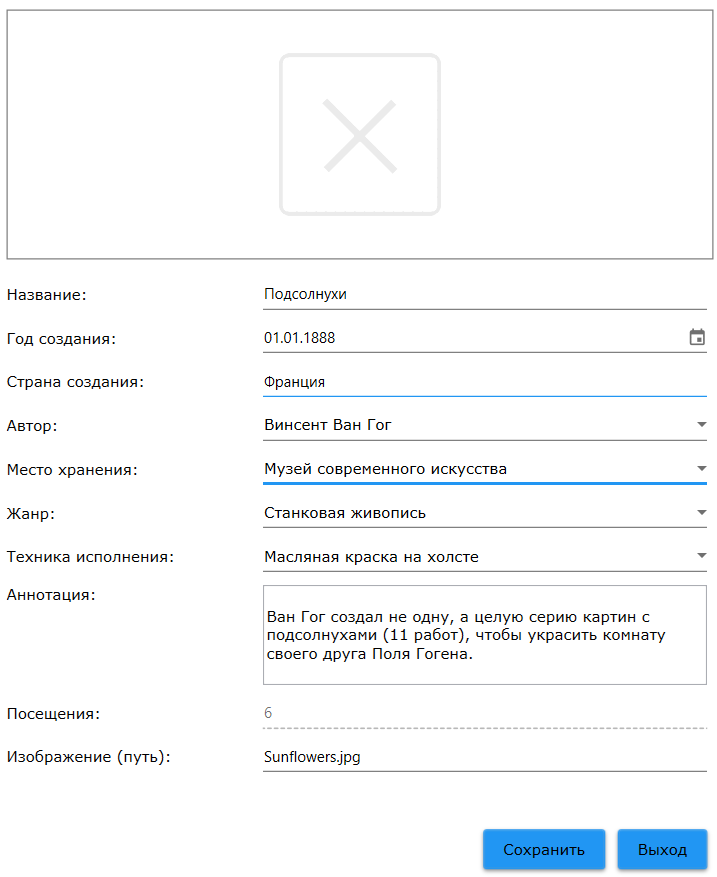


Рисунок 2.3.19 – Окно создания и редактирования произведения искусства

Чтобы выйти из системы пользователь должен нажать на кнопку «Выйти» в правом верхнем углу экрана, после чего пользователя пренаправит на страницу входа.

* 1. Меры по обеспечению защиты информации

Программный продукт включает несколько ключевых мер безопасности, направленных на защиту данных и предотвращение несанкционированного доступа.

Во-первых, авторизация и аутентификация обеспечивают доступ к административным функциям только после ввода корректных логина и пароля. Это гарантирует, что только уполномоченные пользователи могут выполнять важные операции.

Во-вторых, контроль целостности данных осуществляется через проверку вводимых данных на корректность, включая форматы дат, обязательные поля и ограничение длины строк. Это предотвращает ввод некорректной информации и ошибки в системе.

Кроме того, ограничение прав пользователей в гостевом режиме позволяет только просматривать данные без возможности их изменения. Это защищает информацию от несанкционированных действий.

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсового проекта были разработаны структура и алгоритм работы WPF-приложения «ArtGallery».

При этом были изучены особенности реализации компонентов WPF для построения клиентских приложений с визуально привлекательными возможностями взаимодействия с пользователем.

Основным достижением проекта стало создание полнофункциональной системы учета произведений искусства, включающей модуль каталогизации, гибкую систему поиска и фильтрации, генерацию отчетов в PDF-формате и разграничение прав доступа через систему авторизации. Приложение прошло всестороннее тестирование, в ходе которого подтвердилась корректность работы всех модулей, соответствие выходных данных контрольным примерам, устойчивость к некорректным входным данным и оптимальная производительность системы.

WPF-приложение написано на языке C# в среде разработки Visual Studio 2022 с использованием языка разметки XAML и системы управления базой данных MS SQL Server 2022.

Были проведены опытная эксплуатация и отладочное тестирование WPF приложения. По результатам отладочного тестирования были устранены некоторые недостатки, в частности были обнаружены и исправлены неточности в реализации алгоритма: усовершенствован контроль на входные данные и отформатирован вывод документов. После этого было написано руководство пользователя.

С помощью приложения на основании данных контрольного примера были получены результаты, которые полностью совпадают с выходной информацией контрольного примера.

Входные данные контрольного примера

Таблица А.1 – Справочник авторов

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Номер автора | ФИО | Дата рождения | Дата смерти | Страна | Нация | Пол | Интересные факты |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | Винсент Ван Гог | 1853-03-30 | 1890-07-29 | Нидерланды | Голландец | Мужской | Создал более 2,100 произведений, но продал только одну картину при жизни. |
| 2 | Леонардо да Винчи | 1452-04-15 | 1519-05-02 | Италия | Итальянец | Мужской | Был вегетарианцем и покупал птиц в клетках, чтобы выпускать их на свободу. |
| 3 | Пабло Пикассо | 1881-10-25 | 1973-04-08 | Испания | Испанец | Мужской | Его полное имя состоит из 23 слов. |
| 4 | Фрида Кало | 1907-07-06 | 1954-07-13 | Мексика | Мексиканка | Женский | После аварии перенесла 32 операции и рисовала, лежа в постели. |
| 5 | Микеланджело Буонарроти | 1475-03-06 | 1564-02-18 | Италия | Итальянец | Мужской | Спал в одежде и сапогах, работая над фресками Сикстинской капеллы. |
| 6 | Клод Моне | 1840-11-14 | 1926-12-05 | Франция | Француз | Мужской | После операции по удалению катаракты видел ультрафиолет и писал кувшинки в новых оттенках. |
| 7 | Сальвадор Дали | 1904-05-11 | 1989-01-23 | Испания | Испанец | Мужской | Арендовал броненосца и выгуливал его на золотом поводке в Париже. |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 8 | Энди Уорхол | 1928-08-06 | 1987-02-22 | США | Американец | Мужской | Коллекционировал банки с супом Campbell’s и хотел, чтобы их похоронили с ним. |
| 9 | Рембрандт ван Рейн | 1606-07-15 | 1669-10-04 | Нидерланды | Голландец | Мужской | Написал более 80 автопортретов, фиксируя старение. |
| 10 | Густав Климт | 1862-07-14 | 1918-02-06 | Австро-Венгрия | Австриец | Мужской | Ходил в халате без белья и работал в саду среди кошек. |
| 11 | Анри Матисс | 1869-12-31 | 1954-11-03 | Франция | Француз | Мужской | В старости создавал шедевры из вырезанной бумаги. |
| 12 | Иван Айвазовский | 1817-07-29 | 1900-05-02 | Россия | Русский | Мужской | Написал около 6,000 картин, работая по памяти. |
| 13 | Казимир Малевич | 1879-02-23 | 1935-05-15 | Россия | Русский | Мужской | В 2012 году кратер на Меркурии назвали в его честь. |
| 14 | Марк Шагал | 1887-07-07 | 1985-03-28 | Россия/Франция | Еврей | Мужской | Прожил 97 лет, заявив: «Искусство сильнее смерти». |
| 15 | Диего Ривера | 1886-12-08 | 1957-11-24 | Мексика | Мексиканец | Мужской | Был исключен из Академии за протесты, но стал главным художником Мексики. |

Таблица А.2 – Справочник жанров

|  |  |  |  |
| --- | --- | --- | --- |
| Номер жанра | Тип жанра | Жанр | Направление жанра |
| 1 | Станковая живопись | Пейзаж | Постимпрессионизм |
| 2 | Монументальная живопись | Фреска | Ренессанс |
| 3 | Графика | Гравюра | Романтизм |
| 4 | Станковая живопись | Портрет | Реализм |
| 5 | Станковая живопись | Натюрморт | Импрессионизм |
| 6 | Станковая живопись | Исторический | Академизм |
| 7 | Скульптура | Статуя | Классицизм |
| 8 | Станковая живопись | Абстрактный | Абстракционизм |
| 9 | Графика | Литография | Экспрессионизм |
| 10 | Монументальная живопись | Мозаика | Византийский |
| 11 | Станковая живопись | Жанровая сцена | Реализм |
| 12 | Графика | Линогравюра | Экспрессионизм |
| 13 | Станковая живопись | Религиозная | Возрождение |
| 14 | Скульптура | Рельеф | Барокко |
| 15 | Станковая живопись | Символический | Символизм |

Таблица А.3 – Справочник техник исполнения

|  |  |  |
| --- | --- | --- |
| Номер техники исполнения | Вид искусства | Техника |
| 1 | Живопись | Масляная краска на холсте |
| 2 | Скульптура | Литьё из бронзы |
| 3 | Графика | Офорт |
| 4 | Живопись | Акварель |
| 5 | Скульптура | Резьба по камню |
| 6 | Графика | Линогравюра |
| 7 | Живопись | Темпера |
| 8 | Скульптура | Ковка |
| 9 | Графика | Литография |
| 10 | Живопись | Фреска |
| 11 | Скульптура | Керамика |
| 12 | Графика | Гравюра на дереве |
| 13 | Живопись | Масляная пастель |
| 14 | Скульптура | Мрамор |
| 15 | Графика | Монотипия |

Таблица А.4 – Справочник мест хранения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер места хранения | Страна | Тип | Название | Доступность |
| 1 | США | Музей | Музей современного искусства | Публична |
| 2 | Россия | Галерея | Третьяковская галерея | Публична |
| 3 | Нидерланды | Музей | Музей Ван Гога | Публична |
| 4 | Франция | Музей | Лувр | Публична |
| 5 | Испания | Музей | Прадо | Публична |
| 6 | Италия | Музей | Уффици | Публична |
| 7 | Великобритания | Галерея | Национальная галерея | Публична |
| 8 | Германия | Музей | Старая пинакотека | Публична |
| 9 | Австрия | Музей | Музей истории искусств | Публична |
| 10 | Франция | Музей | Центр Помпиду | Публична |
| 11 | Мексика | Музей | Музей Фриды Кало | Публична |
| 12 | Нидерланды | Музей | Рейксмузеум | Публична |
| 13 | Россия | Музей | Эрмитаж | Публична |
| 14 | Франция | Музей | Орсе | Публична |
| 15 | Италия | Музей | Сикстинская капелла | Публична |

Таблица А.5 – Справочник админов

|  |  |  |
| --- | --- | --- |
| Номер админа | Логин | Пароль |
| 1 | admin | 1sk2) |
| 2 | yanina | qwU72\* |

Таблица А.6 – Справочник произведений искусства

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер произведения искусства | Название | Год создания | Страна создания | № автора | № места хранения | № жанра | № техники исполнения | Аннотация | Посещаемость | Путь к картине (короткий) |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 1 | Мона Лиза | 1503-01-01 | Италия | 2 | 4 | 4 | 1 | Загадочная улыбка Джоконды - результат применения техники "сфумато", где Леонардо наносил до 40 слоев полупрозрачной краски. В 1911 году картину похитили из Лувра, и 2 года она висела в квартире вора-итальянца, который считал, что она должна вернуться на родину | 0 | Mona\_Lisa.jpg |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 2 | Звёздная ночь | 1889-06-01 | Франция | 1 | 3 | 1 | 1 | Написана в психиатрической лечебнице Сен-Реми. Врачи запрещали Ван Гогу рисовать на улице, поэтому он воссоздавал вид из окна по памяти. Интересно, что астрономы подтвердили точное положение звезд на тот день (июнь 1889 года) | 0 | Starry\_Night.jpg |
| 3 | Герника | 1937-05-01 | Испания | 3 | 5 | 4 | 1 | Создана за 35 дней как реакция на бомбардировку баскского города. На черно-белой палитре Пикассо сэкономил $200 (около $4,000 сегодня), сказав: "Цвета были бы слишком красивы для этой трагедии" | 0 | Guernica.jpg |
| 4 | Автопортрет с отрезанным ухом | 1889-09-01 | Франция | 1 | 3 | 4 | 1 | Ван Гог отрезал мочку уха и отнес ее в подарок проститутке в борделе. На картине он изобразил себя с перевязанным правым ухом, хотя на самом деле повредил левое - это отражение в зеркале | 0 | SelfPortrait.jpg |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 5 | Танец | 1909-10-01 | Франция | 3 | 1 | 5 | 1 | Ярко-красные фигуры на сине-зеленом фоне вызвали скандал в 1910 году. Критики называли работу "варварской", но сегодня она - символ фовизма. Матисс создал 3 версии, последнюю - по заказу русского коллекционера Щукина | 0 | Dance.jpg |
| 6 | Сотворение Адама | 1511-01-01 | Италия | 5 | 15 | 13 | 10 | На фреске в Сикстинской капелле Бог изображен внутри анатомически точного человеческого мозга. Искусствовед Фрэнк Мешбергер обнаружил, что контуры соответствуют разрезу мозга, включая мозжечок и зрительные нервы | 0 | CreationOfAdam.jpg |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 7 | Водяные лилии | 1919-01-01 | Франция | 6 | 14 | 1 | 1 | Цикл из 250 картин Моне писал в собственном саду в Живерни, который специально спроектировал с японским мостиком. В старости, теряя зрение из-за катаракты, он видел лилии как голубые пятна - это отразилось на поздних работах | 0 | WaterLilies.jpg |
| 8 | Постоянство памяти | 1931-01-01 | Испания | 7 | 5 | 15 | 1 | "Текучие часы" появились под впечатлением от плавленого сыра камамбер. Картину размером 24×33 см Дали написал за 2 часа, пока Гала ходила в кино. | 0 | PersistenceOfMemory.jpg |
| 9 | Крик | 1893-01-01 | Норвегия | 10 | 7 | 4 | 9 | Существует 4 версии картины. На одной из них карандашом написано: "Это мог написать только сумасшедший!" (это сделал сам Мунк после нервного срыва). В 2012 году одна версия была продана за $120 млн | 0 | TheScream.jpg |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 10 | Поцелуй | 1907-01-01 | Австрия | 11 | 9 | 4 | 1 | Золотые пластины на картине - настоящее сусальное золото. Климт унаследовал любовь к золоту от отца-ювелира. Моделями стали сам художник и его возлюбленная Эмилия Флёге, хотя их отношения оставались платоническими | 0 | TheKissKlimt.jpg |
| 11 | Девушка с жемчужной сережкой | 1665-01-01 | Нидерланды | 10 | 12 | 4 | 1 | Жемчужина размером с голубиное яйцо - оптическая иллюзия, она не могла существовать в природе | 0 | GirlWithAPearlEarring.jpg |
| 12 | Фрида и Диего Ривера | 1931-01-01 | Мексика | 12 | 11 | 4 | 1 | На двойном портрете Фрида изобразила мужа как великого художника, а себя - как его "маленькую жену". В реальности она была выше Риверы на 12 см, но специально уменьшила свой рост на картине | 0 | FridaAndDiegoRivera.jpg |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 13 | Подсолнухи | 1888-01-01 | Франция | 1 | 3 | 1 | 1 | Ван Гог создал 11 версий подсолнухов. Желтый цвет стал возможен благодаря новому пигменту "хромово-желтый", который позже потемнел из-за химической реакции. Один из вариантов был куплен за $39.9 млн в 1987 году | 0 | Sunflowers.jpg |
| 14 | Ночные ястребы | 1942-01-01 | США | 8 | 1 | 11 | 1 | Прообразом бара стал реальный ресторан в Гринвич-Виллидж. Хоппер отрицал, что изобразил одиночество, говоря: "Я просто нарисовал хорошо освещенное место, где можно купить гамбургер в 3 часа ночи" | 0 | NightHawks.jpg |
| 15 | Поцелуй | 1908-01-01 | Австрия | 11 | 9 | 4 | 1 | Бранкузи вырезал ее из цельного камня, оставив лишь схематичные очертания лиц, сливающихся в единую форму | 0 | TheKissBrankuzi.jpg |

Таблица Б.1 – Результат поиска по запросу «Звёздная ночь»

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Результат поиска по запросу «Звёздная ночь» | | | | | | | |
| Название картины: | Звёздная ночь | | | | | | |
| Аннотация: | Написана в психиатрической больнице | | | | | | |
| Автор: | Винсент Ван Гог | Страна автора: | Нидерланды | Дата рождения | 1853-03-30 | Посещаемость | 5 |
| Жанр: | **Пейзаж** | Тип жанра: | Станковая живопись | Техника исполнения | Масляная краска на холсте | Доступность | Публична |

Таблица Б.2 - Результат поиска по запросу «Звёздная ночь» в виде pdf-файла

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Результат поиска по запросу «Звёздная ночь» в виде pdf-файла | | | | | | | |
| Название: | Звёздная ночь | Автор: | Винсент Ван Гог | Пол: | Мужской | Место хранения: | Музей современного искусства |
| Год создания: | 01.06.1889 | Дата рождения: | 30.03.1853 | Интересные факты: | Отрезал себе часть уха | Жанр: | Пейзаж |
| Страна создания: | Нидерланды | Дата смерти: | 29.07.1890 | Страна хранения: | США | Тип жанра: | Масляная краска на холсте |
| Просмотры: | 5 | Страна происхождения: | Нидерланды | Тип: | Музей | Направление жанра: | **Постимпрессионизм** |
|  |  | Национальность: | Нидерландец | Доступность: | Публична | Аннотация: | Написана в психиатрической больнице |

Код программы

Форма DashboardWindow.Xaml.cs (Главное окно)

using System.Windows;

using Serilog;

using ArtGallery.Desktop.Models.Entities;

using ArtGallery.Desktop.Models.Views.Windows.Client;

using ArtGallery.Desktop.Views.Windows.Admin;

using ArtGallery.Desktop.Views.Windows.Admin.Editors;

using ArtGallery.Desktop.Views.Windows.Auth;

namespace ArtGallery.Desktop.Views.Windows.Client

{

/// <summary>

/// Interaction logic for DashboardWindow.xaml.

/// </summary>

public partial class DashboardWindow : Window

{

#region Region: Fields and Properties.

public DashboardWindowModel Model { get; private set; }

#endregion

#region Region: Initializers.

public DashboardWindow(Models.Entities.Admin? admin)

{

Model = new DashboardWindowModel(admin);

InitializeComponent();

DataContext = Model;

}

private void OnWindowLoaded(object sender, RoutedEventArgs e)

{

InitializeSelectionControls();

OnSelectionPropertiesChange(sender, e);

}

private void InitializeSelectionControls()

{

InitializeSelectionControlsValues();

InitializeSelectionControlsEventListeners();

}

private void InitializeSelectionControlsValues()

{

InitializeFilteringSelectionControlsValues();

InitializeSortingSelectionControlsValues();

}

private void InitializeSelectionControlsEventListeners()

{

SearchTextBox.TextChanged += OnSelectionPropertiesChange;

InitializeFilteringSelectionControlsEventListeners();

InitializeSortingSelectionControlsEventListeners();

}

private void InitializeFilteringSelectionControlsValues()

{

FilteringSwitch.IsChecked = false;

DateOfCreationPicker.SelectedDate = DateTime.Now;

CountryOfCreationPicker.ItemsSource = Model.WorksOfArt.Select(art => art.CountryCreate)

.Distinct().ToList();

TypeOfStorageLocationPicker.ItemsSource = Model.StorageLocations.DistinctBy(type => type.TypeOfStorageLocation)

.ToList();

StorageLocationPicker.ItemsSource = Model.StorageLocations.ToList();

PerformanceTechniquePicker.ItemsSource = Model.PerformanceTechniques.ToList();

GenrePicker.ItemsSource = Model.Genres.ToList();

}

private void InitializeFilteringSelectionControlsEventListeners()

{

FilteringSwitch.Checked += OnSelectionPropertiesChange;

FilteringSwitch.Unchecked += OnSelectionPropertiesChange;

DateOfCreationPicker.SelectedDateChanged += OnSelectionPropertiesChange;

CountryOfCreationPicker.SelectionChanged += OnSelectionPropertiesChange;

TypeOfStorageLocationPicker.SelectionChanged += OnSelectionPropertiesChange;

StorageLocationPicker.SelectionChanged += OnSelectionPropertiesChange;

PerformanceTechniquePicker.SelectionChanged += OnSelectionPropertiesChange;

GenrePicker.SelectionChanged += OnSelectionPropertiesChange;

MinVisitsPicker.TextChanged += OnSelectionPropertiesChange;

}

private void InitializeSortingSelectionControlsValues()

{

SortingSwitch.IsChecked = false;

SortByYearOfCreationSelector.SelectedIndex = 0;

SortByVisitsSelector.SelectedIndex = 0;

}

private void InitializeSortingSelectionControlsEventListeners()

{

SortingSwitch.Checked += OnSelectionPropertiesChange;

SortingSwitch.Unchecked += OnSelectionPropertiesChange;

SortByYearOfCreationSelector.SelectionChanged += OnSelectionPropertiesChange;

SortByVisitsSelector.SelectionChanged += OnSelectionPropertiesChange;

}

#endregion

#region Region: Content-Related Functions.

#region Sub-Region: Selection Handlers.

private void OnSelectionPropertiesChange(object? sender, dynamic? e)

{

List<WorksOfArt> selectedRecords = CompleteSelection();

UpdateListView(selectedRecords);

}

private List<WorksOfArt> CompleteSelection()

{

var selectedRecords = Model.WorksOfArt.ToList();

selectedRecords = CompleteSearch(selectedRecords);

selectedRecords = CompleteFiltering(selectedRecords);

selectedRecords = CompleteSorting(selectedRecords);

return selectedRecords;

}

private List<WorksOfArt> CompleteSearch(List<WorksOfArt> selectedRecords)

{

if (SearchTextBox.Text is string searchRequest && !string.IsNullOrWhiteSpace(searchRequest))

{

selectedRecords = selectedRecords.Where(work =>

{

var isMatchFioSearch = work.Author?.Fio?.Contains(searchRequest, StringComparison.CurrentCultureIgnoreCase) ?? false;

var isMatchTitleSearch = work.TitleOfThePainting?.Contains(searchRequest, StringComparison.CurrentCultureIgnoreCase) ?? false;

var isMatchTechniqueSearch = work.PerformanceTechnique?.Technique.Contains(searchRequest, StringComparison.CurrentCultureIgnoreCase) ?? false;

return isMatchFioSearch || isMatchTitleSearch || isMatchTechniqueSearch;

}).ToList();

}

return selectedRecords;

}

private List<WorksOfArt> CompleteFiltering(List<WorksOfArt> selectedRecords)

{

if (FilteringSwitch.IsChecked != true)

return selectedRecords;

if (DateOfCreationPicker.SelectedDate is DateTime chosenDateTime)

{

var chosenDate = DateOnly.FromDateTime(chosenDateTime);

selectedRecords = selectedRecords.Where(work => work.YearCreate.Equals(chosenDate))

.ToList();

}

if (CountryOfCreationPicker.SelectedItem is string country)

{

selectedRecords = selectedRecords.Where(work => work.CountryCreate == country)

.ToList();

}

if (TypeOfStorageLocationPicker.SelectedItem is StorageLocation locationType)

{

selectedRecords = selectedRecords.Where(work =>

{

var storageLocationType = work.StorageLocation?.TypeOfStorageLocation ?? string.Empty;

var isStorageMatchesRequest = storageLocationType.Equals(locationType.TypeOfStorageLocation, StringComparison.CurrentCultureIgnoreCase);

return isStorageMatchesRequest;

}).ToList();

}

if (StorageLocationPicker.SelectedItem is StorageLocation locationName)

{

selectedRecords = selectedRecords.Where(work =>

{

var storageLocationName = work.StorageLocation?.NameOfStorageLocation ?? string.Empty;

var isStorageMatchesRequest = storageLocationName == locationName.NameOfStorageLocation;

return isStorageMatchesRequest;

}).ToList();

}

if (PerformanceTechniquePicker.SelectedItem is PerformanceTechnique technique)

{

selectedRecords = selectedRecords.Where(work =>

{

var techniqueName = work.PerformanceTechnique?.Technique ?? string.Empty;

var isTechniqueMatchesRequest = techniqueName == technique.Technique;

return isTechniqueMatchesRequest;

}).ToList();

}

if (GenrePicker.SelectedItem is Genre genre)

{

selectedRecords = selectedRecords.Where(work =>

{

var genreName = work.Genre?.GenreName ?? string.Empty;

var isGenreMatchesRequest = genreName == genre.GenreName;

return isGenreMatchesRequest;

}).ToList();

}

if (MinVisitsPicker.Text is string minVisitsText && int.TryParse(minVisitsText, out int minVisits))

{

selectedRecords = selectedRecords.Where(work => work.Visits >= minVisits)

.ToList();

}

return selectedRecords;

}

private List<WorksOfArt> CompleteSorting(List<WorksOfArt> selectedRecords)

{

if (SortingSwitch.IsChecked != true)

return selectedRecords;

IEnumerable<WorksOfArt> query = selectedRecords;

bool isPrimarySortApplied = false;

switch (SortByYearOfCreationSelector.SelectedIndex)

{

case 0:

query = query.OrderBy(work => work.YearCreate);

isPrimarySortApplied = true;

break;

case 1:

query = query.OrderByDescending(work => work.YearCreate);

isPrimarySortApplied = true;

break;

}

switch (SortByVisitsSelector.SelectedIndex)

{

case 0:

if (isPrimarySortApplied)

{

query = ((IOrderedEnumerable<WorksOfArt>)query).ThenBy(work => work.Visits);

}

else

{

query = query.OrderBy(work => work.Visits);

isPrimarySortApplied = true;

}

break;

case 1:

if (isPrimarySortApplied)

{

query = ((IOrderedEnumerable<WorksOfArt>)query).ThenByDescending(work => work.Visits);

}

else

{

query = query.OrderByDescending(work => work.Visits);

isPrimarySortApplied = true;

}

break;

}

if (isPrimarySortApplied)

return query.ToList();

else

return selectedRecords;

}

private void UpdateListView(List<WorksOfArt> selectedRecords)

{

WorksOfArtListView.ItemsSource = null;

WorksOfArtListView.ItemsSource = selectedRecords;

}

#endregion -- Sub-Region: Selection Handlers.

#region Sub-Region: Selection Controls Handlers.

private void OnResetButtonClick(object sender, RoutedEventArgs e)

{

SearchTextBox.Text = string.Empty;

ResetFilteringSelectionControlsSelectedValues();

ResetSortingSelectionControlsSelectedValues();

}

private void ResetFilteringSelectionControlsSelectedValues()

{

FilteringSwitch.IsChecked = false;

DateOfCreationPicker.SelectedDate = DateTime.Now;

CountryOfCreationPicker.SelectedIndex = -1;

TypeOfStorageLocationPicker.SelectedIndex = -1;

StorageLocationPicker.SelectedIndex = -1;

PerformanceTechniquePicker.SelectedIndex = -1;

GenrePicker.SelectedIndex = -1;

MinVisitsPicker.Text = string.Empty;

}

private void ResetSortingSelectionControlsSelectedValues()

{

SortByYearOfCreationSelector.SelectedIndex = -1;

SortByVisitsSelector.SelectedIndex = -1;

}

private void OnRefreshButtonClick(object sender, RoutedEventArgs e)

{

Model.UpdateSourceList();

InitializeSelectionControlsValues();

OnSelectionPropertiesChange(sender, e);

}

#endregion -- Sub-Region: Selection Controls Handlers.

#region Sub-Region: List Items Handlers.

private void OnWorksOfArtListViewMouseDoubleClick(object sender, System.Windows.Input.MouseButtonEventArgs e)

{

if (WorksOfArtListView.SelectedItem is WorksOfArt selectedWork)

{

WorkOfArtDetailsWindow newWindow = new(selectedWork);

newWindow.Show();

OnSelectionPropertiesChange(sender, e);

}

}

private void OnWorksOfArtListViewContextMenuItemEditButtonClick(object sender, RoutedEventArgs e)

{

if (WorksOfArtListView.SelectedItem is WorksOfArt work)

{

WorksOfArtEditorWindow newWindow = new(work);

var result = newWindow.ShowDialog();

if (result == true)

OnRefreshButtonClick(sender, e);

}

}

private async void OnWorksOfArtListViewContextMenuItemDeleteButtonClickAsync(object sender, RoutedEventArgs e)

{

if (WorksOfArtListView.SelectedItem is WorksOfArt work)

{

var confirmationResult = MessageBox.Show("Удаление необратимо.\nВы уверены?", "Внимание!", MessageBoxButton.YesNo, MessageBoxImage.Question);

if (confirmationResult == MessageBoxResult.Yes)

{

try

{

Log.Information("Attempt to delete 'WorksOfArt' record: {Record}.", work.TitleOfThePainting);

ArtGalleryDBContext.Instance.WorksOfArts.Remove(work);

await ArtGalleryDBContext.Instance.SaveChangesAsync();

OnRefreshButtonClick(sender, e);

Log.Information("Deletion complete.");

MessageBox.Show("Запись успешно удалена.", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

}

catch (Exception ex)

{

Log.Error(ex, "Attempt to delete 'WorksOfArt' record was faulted. Info: {Ex}.", ex.Message);

MessageBox.Show($"В процессе удаления записи произошла ошибка.\n\nДетали:\n{ex.Message}.", "Ошибка!",

MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

}

#endregion -- Sub-Region: List Items Handlers.

#endregion - Region: Content-Related Functions.

#region Region: Navigation Handlers.

private void OnOpenAdminPanelButtonClick(object sender, RoutedEventArgs e)

{

var newWindow = new AdminPanelWindow();

newWindow.ShowDialog();

}

private void OnSaveCurrentDataButtonClick(object sender, RoutedEventArgs e)

{

if (WorksOfArtListView.ItemsSource is List<WorksOfArt> selectedWorks)

{

if (!selectedWorks.Any())

{

MessageBox.Show("Результат выборки пуст. Создание отчёта невозможно.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

ReportGenerationWindow newDialogWindow = new(selectedWorks);

newDialogWindow.ShowDialog();

}

}

private void OnExitButtonClick(object sender, RoutedEventArgs e)

{

AuthWindow newWindow = new();

newWindow.Show();

Close();

}

#endregion

}

}

Форма AuthWindow.xaml.cs (Авторизации)

using System.Windows;

using Serilog;

using ArtGallery.Desktop.Models.Entities;

using ArtGallery.Desktop.Views.Windows.Client;

namespace ArtGallery.Desktop.Views.Windows.Auth

{

/// <summary>

/// Interaction logic for AuthWindow.xaml.

/// </summary>

public partial class AuthWindow : Window

{

public AuthWindow()

{

InitializeComponent();

Log.Information("AuthWindow is initialized.");

ArtGalleryDBContext.Instance.Admins.Any();

}

private void OnWindowKeyUp(object sender, System.Windows.Input.KeyEventArgs e)

{

if (e.Key == System.Windows.Input.Key.Enter)

OnAuthButtonClick(sender, e);

}

private void OnAuthButtonClick(object sender, RoutedEventArgs e)

{

var userLogin = userLoginInput.Text;

var userPassword = userPasswordInput.Password;

if (ArtGalleryDBContext.Instance.Admins.FirstOrDefault(admin => admin.Login == userLogin &&

admin.Password == userPassword) is Models.Entities.Admin admin)

{

Log.Information("Successfull auth as {Login}.", admin.Login);

MessageBox.Show($"Успешный вход ({admin.Login}).", "Вход выполнен", MessageBoxButton.OK, MessageBoxImage.Information);

DashboardWindow newWindow = new(admin);

newWindow.Show();

Close();

}

else

{

Log.Information("Auth failed for credentials: {Login}.", userLogin);

MessageBox.Show($"Учётная запись не найдена.", "Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

private void OnContinueAsGuestButtonClick(object sender, RoutedEventArgs e)

{

DashboardWindow newWindow = new(null);

newWindow.Show();

Close();

}

}

}

ReportGenerationWindow.xaml.cs (Сохранение отчёта)

using System.IO;

using System.Net;

using System.Net.Mail;

using System.Windows;

using System.Windows.Input;

using ArtGallery.Desktop.Controllers.Docs;

using ArtGallery.Desktop.Models.Entities;

using Microsoft.Extensions.Configuration;

using Microsoft.Win32;

using Serilog;

namespace ArtGallery.Desktop.Views.Windows.Client

{

/// <summary>

/// Interaction logic for ReportGenerationWindow.xaml.

/// </summary>

public partial class ReportGenerationWindow : Window

{

#region Region: Fields and Properties.

private const string \_correctValueMessage = "[ Значение: Корректное ]";

private const string \_incorrentValueMessage = "[ Значение: Некорректное ]";

private readonly List<WorksOfArt> \_selectedWorks;

#endregion

#region Region: Initializers.

public ReportGenerationWindow(List<WorksOfArt> selectedWorks)

{

\_selectedWorks = selectedWorks;

InitializeComponent();

}

private void OnWindowLoaded(object sender, RoutedEventArgs e)

{

string defaultPath = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Desktop),

"Report.pdf");

OutputPathTextBox.TextChanged += OnOutputPathTextBoxTextChanged;

OutputPathTextBox.Text = defaultPath;

EmailTextBox.TextChanged += OnEmailTextBoxTextChanged;

EmailTextBox.Text = "YourAddress@site.net";

}

#endregion

#region Region: Event Handlers (Settings).

private void OnOutputPathTextBoxTextChanged(object sender, System.Windows.Controls.TextChangedEventArgs e)

{

var isTargetPathCorrect = IsTargetPathCorrect();

IsPathValidTextBlock.Text = isTargetPathCorrect ? \_correctValueMessage : \_incorrentValueMessage;

}

private void OnEmailTextBoxTextChanged(object sender, System.Windows.Controls.TextChangedEventArgs e)

{

var isTargetMailCorrect = IsTargetEmailValid();

IsEmailValidTextBlock.Text = isTargetMailCorrect ? \_correctValueMessage : \_incorrentValueMessage;

}

private void OnBrowseButtonClick(object sender, RoutedEventArgs e)

{

SaveFileDialog saveFileDialog = new()

{

Filter = "PDF files (\*.pdf)|\*.pdf|All files (\*.\*)|\*.\*",

FileName = Path.GetFileName(OutputPathTextBox.Text),

InitialDirectory = Path.GetDirectoryName(OutputPathTextBox.Text)

};

var result = saveFileDialog.ShowDialog();

if (result == true)

OutputPathTextBox.Text = saveFileDialog.FileName;

}

#endregion

#region Region: Event Handlers (Generation).

private async void OnGenerateReportButtonClickAsync(object sender, RoutedEventArgs e)

{

string outputPath = OutputPathTextBox.Text;

if (!IsTargetPathCorrect())

{

Log.Warning("Attempt to generate report with invalid data.");

MessageBox.Show("Введённые параметры некорректны.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

await BeginDocumentGenerationAsync(outputPath);

}

private async Task BeginDocumentGenerationAsync(string outputPath, bool showNotifications = true)

{

GenerateReportButton.IsEnabled = false;

Cursor = Cursors.Wait;

try

{

Log.Information("Report generation with {Count} of records has started.", \_selectedWorks.Count);

string? directory = Path.GetDirectoryName(outputPath);

if (!string.IsNullOrEmpty(directory) && !Directory.Exists(directory))

Directory.CreateDirectory(directory);

ReportController reportController = new(outputPath, \_selectedWorks);

await reportController.GenerateReportAsync();

Log.Information("Report successfully generated at: {OutputPath}", outputPath);

if (showNotifications)

MessageBox.Show($"Отчет успешно сгенерирован по пути:\n{outputPath}", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

}

catch (Exception ex)

{

Log.Error(ex, "Error occurred during the report generation. Details: {Exception}.", ex.Message);

if (showNotifications)

{

MessageBox.Show($"Произошла ошибка во время создания отчёта.\nИнформация:\n{ex.Message}.", "Ошибка!",

MessageBoxButton.OK, MessageBoxImage.Error);

}

}

finally

{

GenerateReportButton.IsEnabled = true;

Cursor = null;

}

}

private async void OnSendMailMessageButtonClickAsync(object sender, RoutedEventArgs e)

{

string recipientEmail = EmailTextBox.Text;

if (!IsTargetEmailValid())

{

Log.Warning("Attempt to send report with invalid email address: {Email}", recipientEmail);

MessageBox.Show("Пожалуйста, введите корректный E-Mail адрес получателя.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

string fileName = await BeginPreparingTempFileForMessageSeindingAsync(recipientEmail);

await BeginMailMessageSendingAsync(fileName);

}

private async Task<string> BeginPreparingTempFileForMessageSeindingAsync(string recipientEmail)

{

Log.Information("Attempting to generate temporary report for email to: {Email}", recipientEmail);

var fileName = Path.GetRandomFileName() + ".pdf";

await BeginDocumentGenerationAsync(fileName, false);

Log.Information("Temporary report generated at: {TempPath} for email.", fileName);

return fileName;

}

private async Task BeginMailMessageSendingAsync(string tempFileName, bool showNotifications = true)

{

GenerateReportButton.IsEnabled = false;

SendMailMessageButton.IsEnabled = false;

Cursor = Cursors.Wait;

try

{

using (var message = GenerateMessage())

{

Log.Information("Mail message has been generated.");

message.Attachments.Add(new(tempFileName));

using (var client = GenerateSmtpClient())

{

Log.Information("SMTP client has been generated.");

try

{

await client.SendMailAsync(message);

Log.Information("Message has been sent to the recipient.");

if (showNotifications)

{

MessageBox.Show("Письмо успешно отправлено на целевой адрес.", "Информация",

MessageBoxButton.OK, MessageBoxImage.Information);

}

}

catch (Exception ex)

{

Log.Error(ex, "<Sending> Error occurred during the message sending. Details: {Error}.", ex.Message);

if (showNotifications)

{

MessageBox.Show($"В процессе отправки письма произошла ошибка.\nДетали:\n{ex.Message}.", "Ошибка!",

MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

}

}

catch (Exception ex)

{

Log.Error(ex, "<General> Error occurred during the message initialization. Details: {Error}.", ex.Message);

if (showNotifications)

{

MessageBox.Show($"В процессе инициализации письма произошла ошибка.\nДетали:\n{ex.Message}.", "Ошибка!",

MessageBoxButton.OK, MessageBoxImage.Error);

}

}

finally

{

File.Delete(tempFileName);

Log.Information("File ({FileName}) deleted succesfully.", tempFileName);

GenerateReportButton.IsEnabled = true;

SendMailMessageButton.IsEnabled = true;

Cursor = null;

}

}

#endregion

#region Region: Standalone Functions.

private bool IsTargetPathCorrect()

{

var path = OutputPathTextBox.Text;

if (string.IsNullOrWhiteSpace(path))

return false;

if (!Path.Exists(Path.GetDirectoryName(path)))

return false;

if (Path.Exists(path))

return false;

return true;

}

private bool IsTargetEmailValid()

{

var email = EmailTextBox.Text;

if (string.IsNullOrWhiteSpace(email))

return false;

if (!email.Contains('@'))

return false;

return true;

}

private MailMessage GenerateMessage()

{

var configurationBuilder = new ConfigurationBuilder().AddUserSecrets<App>();

var secretProvider = configurationBuilder.Build();

var originMailAddress = secretProvider["SMTP.Google.Login"];

var recipientAddress = EmailTextBox.Text;

var result = new MailMessage();

result.From = new MailAddress(originMailAddress, "Art Gallery");

result.To.Add(recipientAddress);

result.Subject = $"[Art Gallery] Отчёт ({DateTime.Now:yyyy-MM-dd})";

result.Body = """

Это автоматическое письмо, сгенерированное приложением 'Art Gallery'.

В приложениях к письму представлен отчёт по выбранным данным.

""";

result.BodyEncoding = System.Text.Encoding.UTF8;

return result;

}

private static SmtpClient GenerateSmtpClient()

{

var configurationBuilder = new ConfigurationBuilder().AddUserSecrets<App>();

var secretProvider = configurationBuilder.Build();

var originMailAddress = secretProvider["SMTP.Google.Login"];

var originMailPassword = secretProvider["SMTP.Google.Password"];

var smtpServer = "smtp.gmail.com";

int smtpPort = 587;

var result = new SmtpClient(smtpServer, smtpPort);

result.Credentials = new NetworkCredential(originMailAddress, originMailPassword);

result.EnableSsl = true;

result.Timeout = 20 \* 1000;

return result;

}

#endregion

}

}

WorkOfArtDetailsWindow.xaml.cs (Основная информация о произведениях искусства)

using System.Windows;

using ArtGallery.Desktop.Models.Entities;

namespace ArtGallery.Desktop.Views.Windows.Client

{

/// <summary>

/// Interaction logic for WorkOfArtDetailsWindow.xaml.

/// </summary>

public partial class WorkOfArtDetailsWindow : Window

{

public WorksOfArt Model { get; private set; }

public WorkOfArtDetailsWindow(WorksOfArt model)

{

Model = model;

InitializeComponent();

DataContext = Model;

}

private void OnWindowLoaded(object sender, RoutedEventArgs e)

{

Title = $"{Model.TitleOfThePainting} — {Title}";

UpdateWorkOfArtVisits();

}

private void UpdateWorkOfArtVisits()

{

if (Model.Visits.HasValue)

Model.Visits++;

else

Model.Visits = 1;

ArtGalleryDBContext.Instance.SaveChanges();

}

private void OnCloseThisWindowClick(object sender, RoutedEventArgs e) =>

Close();

}

}

AdminPanelWindow.xaml.cs (Окно администратора)

using System.Windows;

using System.Windows.Controls;

using Serilog;

using ArtGallery.Desktop.Models.Views.Windows.Admin;

using ArtGallery.Desktop.Models.Entities;

using ArtGallery.Desktop.Views.Windows.Admin.Editors;

namespace ArtGallery.Desktop.Views.Windows.Admin

{

/// <summary>

/// Interaction logic for AdminPanelWindow.xaml.

/// </summary>

public partial class AdminPanelWindow : Window

{

#region Region: Fields and Properties.

public AdminPanelWindowModel Model { get; private set; }

#endregion

#region Region: Initializers.

public AdminPanelWindow()

{

Model = new();

InitializeComponent();

DataContext = Model;

}

private void OnWindowLoaded(object sender, RoutedEventArgs e)

{

InitializeEventHandlers();

InitializeListViewsBindings();

}

private void InitializeEventHandlers()

{

SearchStorageLocationBox.Text = string.Empty;

SearchStorageLocationBox.TextChanged += OnSearchElementTextChanged;

SearchPerformanceTechniqueBox.Text = string.Empty;

SearchPerformanceTechniqueBox.TextChanged += OnSearchElementTextChanged;

SearchGenreBox.Text = string.Empty;

SearchGenreBox.TextChanged += OnSearchElementTextChanged;

SearchAuthorBox.Text = string.Empty;

SearchAuthorBox.TextChanged += OnSearchElementTextChanged;

SearchWorkOfArtBox.Text = string.Empty;

SearchWorkOfArtBox.TextChanged += OnSearchElementTextChanged;

}

private void InitializeListViewsBindings()

{

UpdateStorageLocationsListView(Model.StorageLocations);

UpdatePerformanceTechniquesListView(Model.PerformanceTechniques);

UpdateGenresListView(Model.Genres);

UpdateAuthorsListView(Model.Author);

UpdateWorksOfArtListView(Model.WorksOfArt);

}

private void UpdateStorageLocationsListView(List<StorageLocation> locations)

{

StorageLocationListView.ItemsSource = null;

StorageLocationListView.ItemsSource = locations;

}

private void UpdatePerformanceTechniquesListView(List<PerformanceTechnique> techniques)

{

PerformanceTechniqueListView.ItemsSource = null;

PerformanceTechniqueListView.ItemsSource = techniques;

}

private void UpdateGenresListView(List<Genre> genres)

{

GenreListView.ItemsSource = null;

GenreListView.ItemsSource = genres;

}

private void UpdateAuthorsListView(List<Author> authors)

{

AuthorListView.ItemsSource = null;

AuthorListView.ItemsSource = authors;

}

private void UpdateWorksOfArtListView(List<WorksOfArt> worksOfArt)

{

WorkOfArtListView.ItemsSource = null;

WorkOfArtListView.ItemsSource= worksOfArt;

}

#endregion

#region Region: Content Functions.

private void OnSearchElementTextChanged(object sender, TextChangedEventArgs e)

{

if (sender is TextBox selectedBox && selectedBox.Tag is string tag &&

selectedBox.Text is string search)

{

switch (tag)

{

case "SL":

var selectedLocations = SearchForStorageLocation(search);

UpdateStorageLocationsListView(selectedLocations);

break;

case "PT":

var selectedTechniques = SearchForPerformanceTechnique(search);

UpdatePerformanceTechniquesListView(selectedTechniques);

break;

case "G":

var selectedGenres = SearchForGenre(search);

UpdateGenresListView(selectedGenres);

break;

case "A":

var selectedAuthors = SearchForAuthor(search);

UpdateAuthorsListView(selectedAuthors);

break;

case "WoA":

var selectedWorks = SearchForWorkOfArt(search);

UpdateWorksOfArtListView(selectedWorks);

break;

default:

Log.Warning("Untagged search box was clicked.");

MessageBox.Show("Поисковая строка не распознана.", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

break;

}

}

}

private List<StorageLocation> SearchForStorageLocation(string search)

{

var selectedRecords = Model.StorageLocations.Where(location => {

var isLocationNameMatches = location.NameOfStorageLocation?.Contains(search, StringComparison.CurrentCultureIgnoreCase) ?? false;

var isLocationTypeMatches = location.TypeOfStorageLocation?.Contains(search, StringComparison.CurrentCultureIgnoreCase) ?? false;

return isLocationNameMatches || isLocationTypeMatches;

});

return selectedRecords.ToList();

}

private List<PerformanceTechnique> SearchForPerformanceTechnique(string search)

{

var selectedRecords = Model.PerformanceTechniques.Where(technique => {

var isTechniqueNameMatches = technique.Technique.Contains(search, StringComparison.CurrentCultureIgnoreCase);

var isTechniqueArtFormMatches = technique.ArtForm.Contains(search, StringComparison.CurrentCultureIgnoreCase);

return isTechniqueNameMatches || isTechniqueArtFormMatches;

});

return selectedRecords.ToList();

}

private List<Genre> SearchForGenre(string search)

{

var selectedRecords = Model.Genres.Where(genre => {

var isGenreNameMatches = genre.GenreName.Contains(search, StringComparison.CurrentCultureIgnoreCase);

var isGenreTypeMatches = genre.TypeOfGenre.Contains(search, StringComparison.CurrentCultureIgnoreCase);

var isGenreDirectionMatches = genre.GenreDirection.Contains(search, StringComparison.CurrentCultureIgnoreCase);

return isGenreNameMatches || isGenreTypeMatches || isGenreDirectionMatches;

});

return selectedRecords.ToList();

}

private List<Author> SearchForAuthor(string search)

{

var selectedRecords = Model.Author.Where(author => {

var isAuthorNameMatches = author.Fio?.Contains(search, StringComparison.CurrentCultureIgnoreCase) ?? false;

var isAuthorNationMatches = author.Nation?.Contains(search, StringComparison.CurrentCultureIgnoreCase) ?? false;

var isAuthorGenderMatches = author.Gender?.Contains(search, StringComparison.CurrentCultureIgnoreCase) ?? false;

return isAuthorNameMatches || isAuthorNationMatches || isAuthorGenderMatches;

});

return selectedRecords.ToList();

}

private List<WorksOfArt> SearchForWorkOfArt(string search)

{

var selectedRecords = Model.WorksOfArt.Where(work =>

{

var isWorkTitleMatches = work.TitleOfThePainting?.Contains(search, StringComparison.CurrentCultureIgnoreCase) ?? false;

var isWorkAuthorFioMatches = work.Author?.Fio?.Contains(search, StringComparison.CurrentCultureIgnoreCase) ?? false;

return isWorkTitleMatches || isWorkAuthorFioMatches;

});

return selectedRecords.ToList();

}

private void OnCreateRecordButtonClick(object sender, RoutedEventArgs e)

{

if (sender is Button clickedButton && clickedButton.Tag is string tag)

{

bool result = CreateNewRecord(tag);

if (result)

UpdateTargetListViewByItsTag(tag);

}

}

private static bool CreateNewRecord(string tag)

{

Window? newWindow = null;

switch (tag)

{

case "SL":

newWindow = new StorageLocationEditorWindow(null);

break;

case "PT":

newWindow = new PerformanceTechniqueEditorWindow(null);

break;

case "G":

newWindow = new GenreEditorWindow(null);

break;

case "A":

newWindow = new AuthorEditorWindow(null);

break;

case "WoA":

newWindow = new WorksOfArtEditorWindow(null);

break;

default:

Log.Warning("Unresolved tag was encountered during new record creation. Tag: {Tag}.", tag);

MessageBox.Show("Произошла внутренняя ошибка распознавания типов объектов.", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

break;

}

var result = newWindow?.ShowDialog() ?? false;

if (result)

{

Log.Information("New record ({Tag}) was added.", tag);

MessageBox.Show("Новая запись успешно добавлена.", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

}

return result;

}

private void OnEditRecordButtonClick(object sender, RoutedEventArgs e)

{

if (sender is Button clickedButton && clickedButton.Tag is string tag)

{

bool result = EditRecord(tag);

if (result)

UpdateTargetListViewByItsTag(tag);

}

}

private bool EditRecord(string tag)

{

Window? newWindow = null;

switch (tag)

{

case "SL":

if (StorageLocationListView.SelectedItem is StorageLocation location)

newWindow = new StorageLocationEditorWindow(location);

else

MessageBox.Show("Для редактирования выберите элемент из списка.", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

break;

case "PT":

if (PerformanceTechniqueListView.SelectedItem is PerformanceTechnique technique)

newWindow = new PerformanceTechniqueEditorWindow(technique);

else

MessageBox.Show("Для редактирования выберите элемент из списка.", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

break;

case "G":

if (GenreListView.SelectedItem is Genre genre)

newWindow = new GenreEditorWindow(genre);

else

MessageBox.Show("Для редактирования выберите элемент из списка.", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

break;

case "A":

if (AuthorListView.SelectedItem is Author author)

newWindow = new AuthorEditorWindow(author);

else

MessageBox.Show("Для редактирования выберите элемент из списка.", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

break;

case "WoA":

if (WorkOfArtListView.SelectedItem is WorksOfArt work)

newWindow = new WorksOfArtEditorWindow(work);

else

MessageBox.Show("Для редактирования выберите элемент из списка.", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

break;

default:

Log.Warning("Unresolved tag was encountered during record edition. Tag: {Tag}.", tag);

MessageBox.Show("Произошла внутренняя ошибка распознавания типов объектов.", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

break;

}

var result = newWindow?.ShowDialog() ?? false;

if (result)

{

Log.Information("Record ({Tag}) was altered.", tag);

MessageBox.Show("Запись успешно изменена.", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

}

return result;

}

private void OnDeleteRecordButtonClick(object sender, RoutedEventArgs e)

{

if (sender is Button clickedButton && clickedButton.Tag is string tag)

{

bool result = DeleteRecord(tag);

if (result)

UpdateTargetListViewByItsTag(tag);

}

}

private bool DeleteRecord(string tag)

{

var confirmation = MessageBox.Show("Удаление необратимо.\nВы уверены?", "Внимание!", MessageBoxButton.YesNo, MessageBoxImage.Question);

if (confirmation != MessageBoxResult.Yes)

return false;

switch (tag)

{

case "SL":

if (StorageLocationListView.SelectedItem is StorageLocation location && location.WorksOfArts.Count > 0)

ArtGalleryDBContext.Instance.StorageLocations.Remove(location);

else

MessageBox.Show("Для удаления выберите элемент из списка.", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

break;

case "PT":

if (PerformanceTechniqueListView.SelectedItem is PerformanceTechnique technique && technique.WorksOfArts.Count > 0)

ArtGalleryDBContext.Instance.PerformanceTechniques.Remove(technique);

else

MessageBox.Show("Для удаления выберите элемент из списка.", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

break;

case "G":

if (GenreListView.SelectedItem is Genre genre && genre.WorksOfArts.Count > 0)

ArtGalleryDBContext.Instance.Genres.Remove(genre);

else

MessageBox.Show("Для удаления выберите элемент из списка.", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

break;

case "A":

if (AuthorListView.SelectedItem is Author author && author.WorksOfArts.Count > 0)

ArtGalleryDBContext.Instance.Authors.Remove(author);

else

MessageBox.Show("Для удаления выберите элемент из списка.", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

break;

case "WoA":

if (WorkOfArtListView.SelectedItem is WorksOfArt work)

ArtGalleryDBContext.Instance.WorksOfArts.Remove(work);

else

MessageBox.Show("Для удаления выберите элемент из списка.", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

break;

default:

Log.Warning("Unresolved tag was encountered during record edition. Tag: {Tag}.", tag);

MessageBox.Show("Произошла внутренняя ошибка распознавания типов объектов.", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

break;

}

var result = false;

try

{

ArtGalleryDBContext.Instance.SaveChanges();

result = true;

}

catch (Exception ex)

{

Log.Error(ex, "Error occurred during the element deletion. Info: {Err}.", ex.Message);

MessageBox.Show("Произошла ошибка при удалении элемента.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

return result;

}

private void UpdateTargetListViewByItsTag(string tag)

{

switch (tag)

{

case "SL":

Model.UpdateStorageLocations();

UpdateStorageLocationsListView(Model.StorageLocations);

break;

case "PT":

Model.UpdatePerformanceTechniques();

UpdatePerformanceTechniquesListView(Model.PerformanceTechniques);

break;

case "G":

Model.UpdateGenres();

UpdateGenresListView(Model.Genres);

break;

case "A":

Model.UpdateAuthors();

UpdateAuthorsListView(Model.Author);

break;

case "WoA":

Model.UpdateWorkOfArts();

UpdateWorksOfArtListView(Model.WorksOfArt);

break;

default:

Log.Warning("Untagged control element was sent to 'UpdateTargetListViewByItsTarget'.");

MessageBox.Show("Нажатая кнопка не распознана.", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

break;

}

}

#endregion

}

}

WorksOfArtEditorWindowModel.xaml.cs (Окно редактирования произведений искусств)

using System.Windows;

using Serilog;

using ArtGallery.Desktop.Models.Entities;

using ArtGallery.Desktop.Models.Views.Windows.Admin.Editors;

namespace ArtGallery.Desktop.Views.Windows.Admin.Editors

{

public partial class WorksOfArtEditorWindow : Window

{

#region Region: Fields and Properties.

public WorksOfArtEditorWindowModel Model { get; private set; }

public bool IsInCreationMode { get; init; }

#endregion

#region Region: Initializers.

public WorksOfArtEditorWindow(WorksOfArt? model)

{

IsInCreationMode = model == null;

Model = new(model);

InitializeComponent();

DataContext = Model;

}

private void OnWindowLoaded(object sender, RoutedEventArgs e)

{

if (Model.EditingRecord.YearCreate is DateOnly date)

{

var finalDate = new DateTime(date, TimeOnly.MinValue);

CreateYearPicker.SelectedDate = finalDate;

}

}

#endregion

#region Region: Event Handlers.

private async void OnSaveButtonClickAsync(object sender, RoutedEventArgs e)

{

if (CreateYearPicker.SelectedDate is DateTime dt)

Model.EditingRecord.YearCreate = DateOnly.FromDateTime(dt);

var errors = GetModelErrors(Model.EditingRecord);

if (errors.Count > 0)

{

Log.Warning("Attempt to save invalid WorkOfArt model. Error count: {Count}", errors.Count);

MessageBox.Show($"Некорректные данные автора:\n\n{string.Join("\n", errors)}",

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

return;

}

try

{

if (IsInCreationMode)

ArtGalleryDBContext.Instance.WorksOfArts.Add(Model.EditingRecord);

await ArtGalleryDBContext.Instance.SaveChangesAsync();

DialogResult = true;

if (IsInCreationMode)

Close();

}

catch (Exception ex)

{

Log.Error(ex, "Exception was encountered during new WorkOfArt saving. Message: {Message}.", ex.Message);

MessageBox.Show($"Произошла ошибка при сохранении новой записи.\nИнформация:\n{ex.Message}.", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void OnExitButtonClick(object sender, RoutedEventArgs e)

{

DialogResult = false;

Close();

}

#endregion

#region Region: Validation.

private static List<string> GetModelErrors(WorksOfArt model)

{

var errors = new List<string>();

if (string.IsNullOrWhiteSpace(model.Annotation))

errors.Add("Аннотация не может быть пустой.");

if (model.Annotation.Length >= 150)

errors.Add("Аннотация слишком длинная.");

if (string.IsNullOrWhiteSpace(model.Image))

errors.Add("Необходимо добавить данные об изображении");

if (model.Image.Length >= 256)

errors.Add("Путь к изображению слишком длинный.");

return errors;

}

#endregion

}

}

StorageLocationEditorWindow.xaml.cs (Окно редактирования места хранения)

using System.Windows;

using Serilog;

using ArtGallery.Desktop.Models.Entities;

namespace ArtGallery.Desktop.Views.Windows.Admin.Editors

{

/// <summary>

/// Interaction logic for StorageLocationEditorWindow.xaml.

/// </summary>

public partial class StorageLocationEditorWindow : Window

{

#region Region: Properties.

public bool IsInCreationMode { get; init; }

public StorageLocation Model { get; private set; }

#endregion

#region Region: Initializers.

public StorageLocationEditorWindow(StorageLocation? model)

{

IsInCreationMode = model == null;

Model = model ?? new();

InitializeComponent();

DataContext = Model;

}

#endregion

#region Region: Event Handlers.

private async void OnSaveButtonClickAsync(object sender, RoutedEventArgs e)

{

var errors = GetModelErrors(Model);

if (errors.Count != 0)

{

Log.Warning("Attempt to create errored model. Amount of errors: {Amount}.", errors.Count);

MessageBox.Show($"Данные записи некорректны.\n\nОбнаруженные ошибки:\n{string.Join(',', errors)}.", "Внимание",

MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

try

{

if (IsInCreationMode)

ArtGalleryDBContext.Instance.StorageLocations.Add(Model);

await ArtGalleryDBContext.Instance.SaveChangesAsync();

DialogResult = true;

if (IsInCreationMode)

Close();

}

catch (Exception ex)

{

Log.Error(ex, "Exception was encountered during new storage location saving. Message: {Message}.", ex.Message);

MessageBox.Show($"Произошла ошибка при сохранении новой записи.\nИнформация:\n{ex.Message}.", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void OnExitButtonClick1(object sender, RoutedEventArgs e)

{

DialogResult = false;

Close();

}

#endregion

#region Region: Validation.

private static List<string> GetModelErrors(StorageLocation model)

{

var errorsList = new List<string>();

if (string.IsNullOrWhiteSpace(model.CountryOfStorage))

errorsList.Add("Страна не указана.");

if (string.IsNullOrWhiteSpace(model.TypeOfStorageLocation))

errorsList.Add("Тип места хранения не указан.");

if (string.IsNullOrWhiteSpace(model.NameOfStorageLocation))

errorsList.Add("Название места хранения не указано.");

if (string.IsNullOrWhiteSpace(model.Availability))

errorsList.Add("Доступность не указана.");

return errorsList;

}

#endregion

}

}

PerformanceTechniqueEditorWindow.xaml.cs (Окно редактирования техники исполнения)

using System.Windows;

using Serilog;

using ArtGallery.Desktop.Models.Entities;

namespace ArtGallery.Desktop.Views.Windows.Admin.Editors

{

public partial class PerformanceTechniqueEditorWindow : Window

{

#region Region: Fields and Properties.

public bool IsInCreationMode { get; init; }

public PerformanceTechnique Model { get; init; }

#endregion

#region Region: Initializers.

public PerformanceTechniqueEditorWindow(PerformanceTechnique? model)

{

IsInCreationMode = model == null;

Model = model ?? new();

InitializeComponent();

DataContext = Model;

}

#endregion

#region Region: Event Handlers.

private async void OnSaveButtonClickAsync(object sender, RoutedEventArgs e)

{

var errors = GetModelErrors(Model);

if (errors.Count != 0)

{

Log.Warning("Attempt to create errored PerformanceTechnique. Errors count: {Count}", errors.Count);

MessageBox.Show($"Данные техники исполнения некорректны.\n\nОшибки:\n{string.Join(", ", errors)}.",

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

return;

}

try

{

if (IsInCreationMode)

ArtGalleryDBContext.Instance.PerformanceTechniques.Add(Model);

await ArtGalleryDBContext.Instance.SaveChangesAsync();

DialogResult = true;

if (IsInCreationMode)

Close();

}

catch (Exception ex)

{

Log.Error(ex, "Exception was encountered during new technique saving. Message: {Message}.", ex.Message);

MessageBox.Show($"Произошла ошибка при сохранении новой записи.\nИнформация:\n{ex.Message}.", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void OnExitButtonClick(object sender, RoutedEventArgs e)

{

DialogResult = false;

Close();

}

#endregion

#region Region: Validation.

private static List<string> GetModelErrors(PerformanceTechnique model)

{

var errors = new List<string>();

if (string.IsNullOrWhiteSpace(model.ArtForm))

errors.Add("Форма искусства не указана");

if (string.IsNullOrWhiteSpace(model.Technique))

errors.Add("Техника не указана");

return errors;

}

#endregion

}

}

GenreEditorWindow.xaml.cs(Окно редактирования жанра)

using System.Windows;

using Serilog;

using ArtGallery.Desktop.Models.Entities;

namespace ArtGallery.Desktop.Views.Windows.Admin.Editors

{

public partial class GenreEditorWindow : Window

{

#region Region: Fields and Proprties.

public bool IsInCreationMode { get; init; }

public Genre Model { get; private set; }

#endregion

#region Region: Initializers.

public GenreEditorWindow(Genre? model)

{

IsInCreationMode = model == null;

Model = model ?? new();

InitializeComponent();

DataContext = Model;

}

#endregion

#region Region: Event Handlers.

private async void OnSaveButtonClickAsync(object sender, RoutedEventArgs e)

{

var errors = GetModelErrors(Model);

if (errors.Count != 0)

{

Log.Warning("Attempt to create errored GenreName. Errors count: {Count}", errors.Count);

MessageBox.Show($"Данные жанра некорректны.\n\nОшибки:\n{string.Join(", ", errors)}.",

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

return;

}

try

{

if (IsInCreationMode)

ArtGalleryDBContext.Instance.Genres.Add(Model);

await ArtGalleryDBContext.Instance.SaveChangesAsync();

DialogResult = true;

if (IsInCreationMode)

Close();

}

catch (Exception ex)

{

Log.Error(ex, "Exception was encountered during new genre saving. Message: {Message}.", ex.Message);

MessageBox.Show($"Произошла ошибка при сохранении новой записи.\nИнформация:\n{ex.Message}.", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void OnExitButtonClick(object sender, RoutedEventArgs e)

{

DialogResult = false;

Close();

}

#endregion

#region Region: Validation.

private static List<string> GetModelErrors(Genre model)

{

var errors = new List<string>();

if (string.IsNullOrWhiteSpace(model.GenreName))

errors.Add("Жанр не указан.");

if (string.IsNullOrWhiteSpace(model.TypeOfGenre))

errors.Add("Тип жанра не указан.");

if (string.IsNullOrWhiteSpace(model.GenreDirection))

errors.Add("Направление жанра не указано.");

return errors;

}

#endregion

}

}

AuthorEditorWindow.xaml.cs (Окно редактирования автора)

using System.Windows;

using Serilog;

using ArtGallery.Desktop.Models.Entities;

namespace ArtGallery.Desktop.Views.Windows.Admin.Editors

{

public partial class AuthorEditorWindow : Window

{

#region Region: Fields and Properties.

public bool IsInCreationMode { get; init; }

public Author Model { get; private set; }

#endregion

#region Region: Initializers.

public AuthorEditorWindow(Author? model)

{

IsInCreationMode = model == null;

Model = model ?? new();

InitializeComponent();

DataContext = Model;

}

#endregion

#region Region: Event Handlers.

private async void OnSaveButtonClickAsync(object sender, RoutedEventArgs e)

{

if (BirthDatePicker.SelectedDate is DateTime birthDate)

Model.Birth = DateOnly.FromDateTime(birthDate);

if (DeathDatePicker.SelectedDate is DateTime deathDate)

Model.Death = DateOnly.FromDateTime(deathDate);

var errors = GetModelErrors(Model);

if (errors.Count > 0)

{

Log.Warning("Attempt to save invalid Author model. Error count: {Count}", errors.Count);

MessageBox.Show($"Некорректные данные автора:\n\n{string.Join("\n", errors)}",

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

return;

}

try

{

if (IsInCreationMode)

ArtGalleryDBContext.Instance.Authors.Add(Model);

await ArtGalleryDBContext.Instance.SaveChangesAsync();

DialogResult = true;

if (IsInCreationMode)

Close();

}

catch (Exception ex)

{

Log.Error(ex, "Exception was encountered during new author saving. Message: {Message}.", ex.Message);

MessageBox.Show($"Произошла ошибка при сохранении новой записи.\nИнформация:\n{ex.Message}.", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void OnExitButtonClick(object sender, RoutedEventArgs e)

{

DialogResult = false;

Close();

}

#endregion

#region Region: Validation.

private static List<string> GetModelErrors(Author model)

{

var errors = new List<string>();

if (string.IsNullOrWhiteSpace(model.Fio))

errors.Add("ФИО не указано.");

if (model.Fio?.Length >= 85)

errors.Add("ФИО слишком длинное");

if (model.Birth is DateOnly birth && model.Death is DateOnly death && death < birth)

errors.Add("Дата смерти не может быть раньше даты рождения.");

if (model.CountryOfOrigin?.Length >= 25)

errors.Add("Название страны происхождения слишком длинная.");

if (model.Nation?.Length >= 25)

errors.Add("Название национальности слишком длинное.");

if (model.Gender?.Length >= 8)

errors.Add("Название пола слишком длинное.");

if (model.InterestingFacts?.Length >= 85)

errors.Add("Строка с фактами слишком длинная.");

return errors;

}

#endregion

}

}

DashboardWindowModel.cs (Класс главного окна)

using ArtGallery.Desktop.Models.Entities;

using System.Windows;

namespace ArtGallery.Desktop.Models.Views.Windows.Client

{

public class DashboardWindowModel

{

#region Region: Properties (Main).

public Entities.Admin? Admin { get; private set; }

public List<PerformanceTechnique> PerformanceTechniques { get; private set; }

public List<StorageLocation> StorageLocations { get; private set; }

public List<Genre> Genres { get; private set; }

public List<WorksOfArt> WorksOfArt { get; private set; }

#endregion

#region Region: Properties (Virtual).

public Visibility AdminControlsVisibilityBinding

{

get

{

return Admin != null ? Visibility.Visible :

Visibility.Hidden;

}

}

#endregion

public DashboardWindowModel(Entities.Admin? admin)

{

Admin = admin;

UpdateSourceList();

}

public void UpdateSourceList()

{

PerformanceTechniques = ArtGalleryDBContext.Instance.PerformanceTechniques.ToList();

StorageLocations = ArtGalleryDBContext.Instance.StorageLocations.ToList();

Genres = ArtGalleryDBContext.Instance.Genres.ToList();

WorksOfArt = ArtGalleryDBContext.Instance.WorksOfArts.ToList();

}

}

}

AdminPanelWindowModel.cs (Класс админа)

using ArtGallery.Desktop.Models.Entities;

namespace ArtGallery.Desktop.Models.Views.Windows.Admin

{

public class AdminPanelWindowModel

{

#region Region: Fields and Properties.

public List<StorageLocation> StorageLocations { get; set; }

public List<PerformanceTechnique> PerformanceTechniques { get; set; }

public List<Genre> Genres { get; set; }

public List<Author> Author { get; set; }

public List<WorksOfArt> WorksOfArt { get; set; }

#endregion

public AdminPanelWindowModel()

{

UpdateLists();

}

public void UpdateLists()

{

UpdateStorageLocations();

UpdatePerformanceTechniques();

UpdateGenres();

UpdateAuthors();

UpdateWorkOfArts();

}

public void UpdateWorkOfArts() =>

WorksOfArt = ArtGalleryDBContext.Instance.WorksOfArts.ToList();

public void UpdateAuthors() =>

Author = ArtGalleryDBContext.Instance.Authors.ToList();

public void UpdateGenres() =>

Genres = ArtGalleryDBContext.Instance.Genres.ToList();

public void UpdatePerformanceTechniques() =>

PerformanceTechniques = ArtGalleryDBContext.Instance.PerformanceTechniques.ToList();

public void UpdateStorageLocations() =>

StorageLocations = ArtGalleryDBContext.Instance.StorageLocations.ToList();

}

}

WorksOfArtEditorWindowModel.cs (Класс редактирования произведения искусства)

using ArtGallery.Desktop.Models.Entities;

namespace ArtGallery.Desktop.Models.Views.Windows.Admin.Editors

{

public class WorksOfArtEditorWindowModel

{

#region Region: Fields and Properties.

public WorksOfArt EditingRecord { get; private set; }

public List<StorageLocation> StorageLocations { get; private set; }

public List<PerformanceTechnique> PerformanceTechniques { get; private set; }

public List<Genre> Genres { get; private set; }

public List<Author> Authors { get; private set; }

#endregion

#region Region: Initializers.

public WorksOfArtEditorWindowModel(WorksOfArt? record)

{

EditingRecord = record ?? new();

UpdateLists();

}

private void UpdateLists()

{

StorageLocations = ArtGalleryDBContext.Instance.StorageLocations.ToList();

PerformanceTechniques = ArtGalleryDBContext.Instance.PerformanceTechniques.ToList();

Genres = ArtGalleryDBContext.Instance.Genres.ToList();

Authors = ArtGalleryDBContext.Instance.Authors.ToList();

}

#endregion

}

}

ReportController.cs (Генерация отчёта)

using System.IO;

using System.Net.Http;

using System.Drawing;

using Aspose.Words;

using Aspose.Words.Tables;

using Serilog;

using ArtGallery.Desktop.Models.Entities;

namespace ArtGallery.Desktop.Controllers.Docs

{

public class ReportController

{

#region Region: Fields and Properties.

private readonly string \_outputPath;

private readonly IEnumerable<WorksOfArt> \_worksOfArt;

private static readonly HttpClient \_httpClient = new();

private const string FallbackImagePath = "./Assets/Images/Records/!NotFound.png";

#endregion

#region Region: Initializers.

public ReportController(string outputPath, IEnumerable<WorksOfArt> worksOfArt)

{

\_outputPath = outputPath;

\_worksOfArt = worksOfArt;

}

#endregion

#region Region: Main Functions.

public async Task GenerateReportAsync()

{

Document doc = new();

DocumentBuilder builder = new(doc);

builder.Font.Name = "Arial";

bool isFirstPage = true;

foreach (var workOfArt in \_worksOfArt)

{

if (!isFirstPage)

{

builder.InsertBreak(BreakType.PageBreak);

}

AddTitle(builder, workOfArt.TitleOfThePainting ?? "[ Название недоступно ]");

AddKeyValueLine(builder, "Год создания", workOfArt.YearCreate?.ToString() ?? "[ Год неизвестен ]");

AddKeyValueLine(builder, "Страна создания", workOfArt.CountryCreate ?? " [ Страна неизвестна ] ");

AddKeyValueLine(builder, "Просмотры", workOfArt.Visits.ToString() ?? "[ Просмотры недоступны ]");

builder.Writeln();

Table table = builder.StartTable();

builder.InsertCell();

table.FirstRow.FirstCell.CellFormat.VerticalAlignment = CellVerticalAlignment.Top;

table.FirstRow.FirstCell.CellFormat.PreferredWidth = PreferredWidth.FromPercent(35);

await InsertImageAsync(builder, workOfArt.Image, FallbackImagePath);

builder.InsertCell();

table.FirstRow.LastCell.CellFormat.PreferredWidth = PreferredWidth.FromPercent(65);

table.SetBorders(LineStyle.None, 0, Color.Empty);

AddContentToRightCell(builder, workOfArt);

builder.EndRow();

builder.EndTable();

isFirstPage = false;

}

doc.Save(\_outputPath, SaveFormat.Pdf);

}

private static void AddTitle(DocumentBuilder builder, string titleText)

{

builder.Font.Size = 16;

builder.Font.Bold = true;

builder.Writeln(titleText);

builder.Font.Size = 11;

builder.Font.Bold = false;

}

private static void AddHeading(DocumentBuilder builder, string headingText)

{

builder.Font.Bold = true;

builder.Writeln(headingText);

builder.Font.Bold = false;

}

private static void AddKeyValueLine(DocumentBuilder builder, string key, string value)

{

builder.Font.Bold = true;

builder.Write(key);

builder.Write(": ");

builder.Font.Bold = false;

builder.Writeln(value);

}

private static void AddContentToRightCell(DocumentBuilder builder, WorksOfArt workOfArt)

{

AddHeading(builder, "Аннотация");

builder.Writeln(workOfArt.Annotation ?? "[ Аннотация отсутствует ]");

builder.Writeln();

if (workOfArt.Author != null)

{

AddHeading(builder, "Автор");

builder.Writeln(workOfArt.Author.Fio ?? "[ Имя автора неизвестно ]");

AddKeyValueLine(builder, "Дата рождения", workOfArt.Author.Birth?.ToString("d") ?? "[ Дата рождения неизвестна ]");

AddKeyValueLine(builder, "Дата смерти", workOfArt.Author.Death?.ToString("d") ?? "[ Дата смерти неизвестна ]");

AddKeyValueLine(builder, "Страна происхождения", workOfArt.Author.CountryOfOrigin ?? "[ Страна происхождения неизвестна ]");

AddKeyValueLine(builder, "Национальность", workOfArt.Author.Nation ?? "[ Национальность ]");

AddKeyValueLine(builder, "Пол", workOfArt.Author.Gender ?? "[ Пол неизвестен ]");

builder.Font.Bold = true;

builder.Write("Интересные факты");

builder.Write(": ");

builder.Font.Bold = false;

builder.Writeln(workOfArt.Author.InterestingFacts ?? "[ Факты отсутствуют ]");

builder.Writeln();

}

else

{

AddHeading(builder, "Автор");

builder.Writeln("[ Информация об авторе недоступна ]");

builder.Writeln();

}

if (workOfArt.Genre != null)

{

AddHeading(builder, "Жанр");

builder.Writeln(workOfArt.Genre.GenreName ?? "[ Название жанра неизвестно ]");

AddKeyValueLine(builder, "Тип жанра", workOfArt.Genre.TypeOfGenre ?? "[ Тип жанра неизвестен ]");

AddKeyValueLine(builder, "Направление жанрa", workOfArt.Genre.GenreDirection ?? "[ Направление жанра неизвестно ]");

builder.Writeln();

}

else

{

AddHeading(builder, "Жанр");

builder.Writeln("[ Информация о жанре недоступна ]");

builder.Writeln();

}

if (workOfArt.StorageLocation != null)

{

AddHeading(builder, "Место хранения");

builder.Writeln(workOfArt.StorageLocation.NameOfStorageLocation ?? "[ Название места хранения неизвестно ]");

AddKeyValueLine(builder, "Страна", workOfArt.StorageLocation.CountryOfStorage ?? "[ Страна хранения неизвестна ]");

AddKeyValueLine(builder, "Тип", workOfArt.StorageLocation.TypeOfStorageLocation ?? "[ Тип места хранения неизвестен ]");

AddKeyValueLine(builder, "Доступность", workOfArt.StorageLocation.Availability ?? "[ Доступность неизвестна ]");

builder.Writeln();

}

else

{

AddHeading(builder, "Место хранения");

builder.Writeln("[ Информация о месте хранения недоступна ]");

builder.Writeln();

}

if (workOfArt.PerformanceTechnique != null)

{

AddHeading(builder, "Техника исполнения");

builder.Writeln(workOfArt.PerformanceTechnique.Technique ?? "[ Техника исполнения неизвестна ]");

AddKeyValueLine(builder, "Вид искусства", workOfArt.PerformanceTechnique.ArtForm ?? "[ Вид искусства неизвестен ]");

}

else

{

AddHeading(builder, "Техника исполнения");

builder.Writeln("[ Информация о технике исполнения недоступна ]");

}

}

private static async Task InsertImageAsync(DocumentBuilder builder, string imageUrl, string fallbackImagePath)

{

bool isImageInserted = false;

byte[]? imageBytes = null;

if (!string.IsNullOrWhiteSpace(imageUrl))

{

try

{

HttpResponseMessage response = await \_httpClient.GetAsync(imageUrl);

if (response.IsSuccessStatusCode)

imageBytes = await response.Content.ReadAsByteArrayAsync();

else

Log.Warning("Image download failed for URL: {Warning}. Status: {Status}.", imageUrl, response.StatusCode);

}

catch (Exception ex)

{

Log.Error(ex, "Exception during image download from URL: {Error}.", imageUrl);

}

}

if (imageBytes == null && !string.IsNullOrWhiteSpace(fallbackImagePath))

{

try

{

string fullFallbackPath = Path.GetFullPath(fallbackImagePath);

if (File.Exists(fullFallbackPath))

imageBytes = await File.ReadAllBytesAsync(fullFallbackPath);

else

Log.Warning("Fallback image file not found: {FullFallbackPath}", fullFallbackPath);

}

catch (Exception ex)

{

Log.Error(ex, "Exception during loading fallback image from path: {FallbackImagePath}.", fallbackImagePath);

}

}

if (imageBytes != null)

{

try

{

using (MemoryStream ms = new(imageBytes))

{

var preferredImageSizeInPixels = 128;

builder.InsertImage(ms, preferredImageSizeInPixels, preferredImageSizeInPixels);

isImageInserted = true;

}

}

catch (Exception ex)

{

Log.Error(ex, "Exception during inserting image into document.");

isImageInserted = false;

}

}

if (!isImageInserted)

{

builder.Font.Italic = true;

builder.Writeln("[ Изображение недоступно ]");

builder.Font.Italic = false;

}

}

#endregion

}

}

СПИСОК СОКРАЩЕНИЙ

БД – База данных

РФ – Российская Федерация

UML – Unified Modeling Language диаграмма

XML – eXtensible Markup Language

СУБД – Система управления базы данных

ИС – Информационная система

ПО – Программное обеспечение

ОС – Операционная система

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Российская Федерация. Приказ Минобрнауки России. Об утверждении федерального государственного образовательного стандарта среднего профессионального образования по специальности 09.02.07 информационные системы и программирование: Приказ от 9 декабря 2016 г. № 1547: ред. от 17.12.2020: Утвержден приказом Министерства образования и науки Российской Федерации – Текст: электронный. – URL: https://fgos.ru/fgos/fgos-09-02-07-informacionnye-sistemy-i-programmirovanie-1547/ (дата обращения: 13.02.2024). – Режим доступа: свободный;
2. ГОСТ Р 59792 – 2021. Информационные технологии. Комплекс стандартов на автоматизированные системы. Виды испытаний автоматизированных систем: Национальный стандарт Российской федерации;
3. ГОСТ 19.001 – 77. Общее положение. Единая система программной документации: дата введения 1980 – 01–01. – Москва: Стандартинформ, 2010 – https://files.stroyinf.ru/Index/25/25115.htm (дата обращения: 09.02. 2024). – Текст: электронный;
4. ГОСТ 19.104 – 78. Единая система программной продукции. Основные надписи. – дата введения: 1980 – 01 – 01. – Москва: Стандартинформ, 2010 – https://files.stroyinf.ru/Index/32/32404.htm (дата обращения: 09.02. 2024). – Текст: электронный;
5. ГОСТ 19.701 – 90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения: национальный стандарт Российской Федерации: дата введения 1992-01-01 - https://internet-law.ru/gosts/gost/28346/ (дата обращения: 23.01.2024). – Текст: электронный;
6. ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила
7. выполнения. – дата введения: 1992 - 01 - 01. – Москва: Стандартинформ, 2010 – https://files.stroyinf.ru/Index/28/28346.htm (дата обращения: 09.02. 2024). – Текст: электронный;
8. ГОСТ 34.602-2020. Информационные технологии. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы: национальный стандарт Российской Федерации: дата введения 2022-01-01 [https://docs.cntd.ru/document/1200181804?ysclid=ld8kr8 gl1r496583163(дата](https://docs.cntd.ru/document/1200181804?ysclid=ld8kr8%20gl1r496583163(дата) обращения: 24.01.2024). – Текст: электронный;
9. Абрамян, Михаил. Visual C# на примерах / Михаил Абрамян. – М.: БХВ-Петербург, 2020. – 231 с. – ISBN не указан. – Текст: непосредственный;
10. Албахари, Джозеф; Албахари, Бен. C# 3.0. Справочник / Джозеф Албахари, Бен Албахари. – М.: БХВ-Петербург, 2020. – 944 с. – ISBN не указан. – Текст: непосредственный;
11. Биллиг, В. А. Основы программирования на C# / В. А. Биллиг. – М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2022. – 488 с. – ISBN не указан. – Текст: непосредственный;
12. Волков, Андрей Александрович. Язык программирования C# 8.0 и платформа .NET Core 3.0. Санкт-Петербург: БХВ-Петербург, 2020. – 928 с. – ISBN 978-5-4461-1346-4 – Текст: непосредственный;
13. Дейт К.Дж. SQL и реляционная теория. Как грамотно писать код на SQL / К.Дж. Дейт. – М.: Символ–плюс, 2021. - 753 с;
14. Дунаев, В.В. Базы данных. Язык SQL для студента / В.В. Дунаев. — М.: БХВ-Петербург, 2022. – 288 с. – ISBN не указан. – Текст: непосредственный;
15. Климов, А. C#. Советы программистам / А. Климов. – М.: БХВ-Петербург, 2018. – 544 с. – ISBN не указан. – Текст: непосредственный;
16. Культин, Н. С# в задачах и примерах / Н. Культин. – М.: БХВ-Петербург, 2020. – 1293 с. – ISBN не указан. – Текст: непосредственный;
17. Стиллмен, Эндрю. Изучаем C# / Эндрю Стиллмен. – М.: Питер, 2020. – 356 с. – ISBN не указан. – Текст: непосредственный;
18. Хартли, Г., Шарп, Дж. C# для профессионалов / Генри Хартли, Джон Шарп. – М.: ДМК Пресс, 2021. – 640 с. – ISBN 978-5-97060-928-2. – Текст: непосредственный;
19. Хлебостроев, В. Г. C# в примерах и задачах: Учебное пособие / В. Г. Хлебостроев. – Москва: Лань, 2020. – 272 с. – ISBN 978-5-907114-69-2. – Текст: непосредственный;
20. Трофимов, В. В. Алгоритмизация и программирование : учебник для вузов / В. В. Трофимов, Т. А. Павловская ; под редакцией В. В. Трофимова. – Москва: Издательство Юрайт, 2023. – 137 с. – (Высшее образование). – ISBN 978-5-534-07834-3. – Текст: электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/513269 (дата обращения: 19.02.2024);
21. Язык запросов SQL. – Текст: электронный // структурированный язык манипулирования данными SQL: [сайт]. – URL: [https://sql-language.ru](https://sql-language.ru/) (дата обращения: 13.02.2024)., Режим доступа: свободный. Текст: электронный.