**ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ ГОРОДА МОСКВЫ**

**Государственное бюджетное профессиональное образовательное учреждение города Москвы**

**«Колледж автоматизации и информационных технологий № 20»**

**ДОПУЩЕН К ЗАЩИТЕ**

И.о. заместителя директора

/ И.О. Фамилия

« » 2025 г.

**ДИПЛОМНЫЙ ПРОЕКТ**

**Тема:** **«Разработка информационной системы по автоматизации процесса бронирования и продаж для ООО «Глобал Текс»**

|  |  |
| --- | --- |
| обучающегося группы ИСП431 | Андреевой Софьи Леонидовны |
|  | (Ф.И.О) |

|  |  |  |  |
| --- | --- | --- | --- |
| ⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀ |  |  | ⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀ |
| (подпись) |  |  | (дата) |

по специальности

09.02.07 Информационные системы и программирование

(код и наименование специальности)

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель проекта | ⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀ | А.В. Лисавина | «⠀⠀⠀» ⠀⠀⠀⠀⠀⠀⠀⠀ 2025 г. |
|  | (подпись) | (И.О. Фамилия) |  |
| Консультант по экономической части | ⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀ | И.О. Фамилия | «⠀⠀⠀» ⠀⠀⠀⠀⠀⠀⠀⠀ 2025 г. |
|  | (подпись) | (И.О. Фамилия) |  |

**Согласовано**

Заведующий отделением

И.О. Фамилия

« » 2025 г.

2025 г.

**ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ ГОРОДА МОСКВЫ**

**Государственное бюджетное профессиональное образовательное   
учреждение города Москвы**

**«Колледж автоматизации и информационных технологий № 20»**

|  |  |
| --- | --- |
| РАССМОТРЕНО и ОДОБРЕНО  НА ЗАСЕДАНИИ ППО «ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ  Протокол №\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Председатель\_\_\_\_\_\_ О.Б. Кузьмина | УТВЕРЖДАЮ  Заведующий отделением  \_\_\_\_\_\_\_\_\_\_\_\_\_ И.О. Фамилия  «\_\_\_» \_\_\_\_\_\_\_\_ 20\_\_ г. |

**ЗАДАНИЕ**

НА ДИПЛОМНЫЙ ПРОЕКТ ОБУЧАЮЩЕМУСЯ (ЙСЯ)

Андреевой Софье Леонидовне

(фамилия, имя, отчество)

|  |  |
| --- | --- |
| 1. Тема: | Разработка информационной системы по автоматизации процесса бронирования и продаж для ООО «Глобал Текс» |

1. Тема утверждена приказом по колледжу от 31.01.2025 г. № 15-10/уч.
2. Срок сдачи обучающимся законченного дипломного проекта:«\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2025 г.
3. Исходные данные к дипломному проекту:

Данные для импорта в базу данных, описание функциональных требований к системе, изображения картин.

1. Перечень вопросов, подлежащих разработке в дипломном проекте:

Провести анализ существующих решений, сформировать требования к системе, осуществить выбор технологий для разработки, спроектировать архитектуру системы, разработать базу данных и пользовательский интерфейс, реализовать функциональные модули системы, провести тестирование и отладку, провести оценку экономической эффективности.

1. Перечень иллюстрированных материалов (графические материалы с точным указанием обязательных чертежей или раздаточных материалов, с указанием наименования таблиц, рисунков или схем):

Диаграмма компонентов, ER-диаграмма.

Консультанты:

|  |  |  |  |
| --- | --- | --- | --- |
| **Раздел работы содержание определяется ППО по направлению** | **Консультант по разделу (ФИО)** | **Дата** | **Подпись** |
| Раздел: Экономическая часть | Иванов Иван Иванович |  |  |

|  |  |
| --- | --- |
| Руководитель дипломного проекта | Задание принял (а) к исполнению обучающийся (щаяся) |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/  подпись Ф.И.О.  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ г. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/  подпись Ф.И.О.  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ г. |

СОДЕРЖАНИЕ

[СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ 6](#_Toc197512329)

[ВВЕДЕНИЕ 7](#_Toc197512330)

[ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И СУЩЕСТВУЮЩИХ РЕШЕНИЙ 9](#_Toc197512331)

[1.1 Описание предметной области 9](#_Toc197512332)

[1.2 Анализ бизнес-процессов художественной галереи 10](#_Toc197512333)

[1.3 Целевые пользователи и их потребности 11](#_Toc197512334)

[1.4 Анализ аналогов и существующих решений 13](#_Toc197512335)

[1.5 Требования к функционалу приложения 20](#_Toc197512336)

[1.6 Технологический стек 21](#_Toc197512337)

[1.6.1 Выбор системы управления базой данных 21](#_Toc197512338)

[1.6.2 Выбор языка программирования 28](#_Toc197512339)

[1.6.3 Выбор среды разработки 32](#_Toc197512340)

[1.7 Особенности конфиденциальности и безопасности данных 34](#_Toc197512341)

[1.8 Проблемы текущих подходов к управлению коллекцией 35](#_Toc197512342)

[ГЛАВА 2 Структура и функциональность приложения GalleryApp 36](#_Toc197512343)

[2.1 Общая структура приложения 36](#_Toc197512344)

[2.2 Модель данных 38](#_Toc197512345)

[2.3 Описание основных страниц интерфейса 41](#_Toc197512346)

[2.4 Логическая структура страниц 47](#_Toc197512347)

[2.5 Реализация логики взаимодействия с базой данных 54](#_Toc197512348)

[2.6 Генерация документов Word 56](#_Toc197512349)

[2.7 Архитектура хранения и визуализации изображений в приложении 62](#_Toc197512350)

[ГЛАВА 3 Итоги разработки программного обеспечения 66](#_Toc197512351)

[3.1 Общая характеристика разработанного программного продукта 66](#_Toc197512352)

[3.2 Результаты тестирования и проверка работоспособности 69](#_Toc197512353)

[3.3 Возможности масштабирования и дальнейшего развития 71](#_Toc197512354)

## **СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ**

ПО – программное обеспечение.

ИС – информационная система.

## **ВВЕДЕНИЕ**

В настоящее время технологии быстро развиваются, и многие компании чувствуют необходимость выйти на цифровой рынок, создав сайты или приложения. Все вышеперечисленное необходимо для того, чтобы наладить новые, более персонализированные отношения с каждым пользователем. Особенно это значимо для галереи, ведь восприятие искусства каждым человеком уникально.

Многие люди чувствуют, что сфера искусства для них закрыта и непонятна, некоторые считают галереи и музеи бессмысленными или слишком сложными для понимания. Именно поэтому первоочередная задача каждой художественной институции – сделать искусство простым, доступным и привлекательным для каждого посетителя. Хорошая галерея предлагает качественные предметы с проверенным провенансом и позволяет сформировать близкую связь между искусством и посетителем, благодаря шансу приобрести предмет искусства. Кроме того, для художников, галерея – это возможность реализовать свое творчество, найти своего покупателя.

Приложение для галереи сможет дать посетителю представление о пространстве и заинтересовать его, а автоматизация производственного процесса поможет ускорить и упростить работу сотрудникам. Помимо этого, уменьшение рутинного труда позволит работникам заняться творческими задачами, уменьшит стресс и соответственно уменьшит вероятность совершения ошибок.

Объектом исследования выступает процесс взаимодействия клиента с художественной галереей.

Предметом исследования являются разработка информационной системы для автоматизации производственных процессов художественной галереи.

Целью проекта является разработка информационной системы по автоматизации процесса бронирования и продажи картин художественной галереи.

Сформулированы следующие задачи:

1. Провести анализ существующих решений;
2. Сформировать требования к системе;
3. Осуществить выбор технологий для разработки;
4. Спроектировать архитектуру системы;
5. Разработать базу данных и пользовательский интерфейс;
6. Реализовать функциональные модули системы;
7. Провести тестирование и отладку;
8. Провести оценку экономической эффективности.

Дипломный проект состоит из введения, …………………..

Объем пояснительной записки составляет 0 страниц, объем списка использованных источников – 0 источников, количество приложений – 0.

**ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И СУЩЕСТВУЮЩИХ РЕШЕНИЙ**

* 1. ****Описание предметной области****

**Художественная галерея, как и многие другие институции, подразумевает значительную работу с документами. Искусство, являясь признанной ценностью в обществе, требует фиксации каждого перемещения произведения в документах. При работе с искусством, особенно перед его выставлением или продажей, необходимо установить: наиболее точный провенанс предмета, его подлинность, дату создания, историю перемещений, подтвердить возраст использованных материалов и провести атрибуцию. Все эти детали помогают подтвердить подлинность произведения. Также важен процесс детального осмотра предмета для фиксации состояния и возможных повреждений, а также определения условий хранения и необходимости реставрации.**

**При продаже галерее важно составить грамотный договор с покупателем, подробно оговорив права и обязанности сторон. Например, покупателю следует объяснить, что договор купли-продажи предоставляет право на хранение, но не позволяет извлекать прибыль от воспроизведения произведения, например, на открытках или другой продукции. Закон об исключительном праве на произведение предполагает, что автор или правообладатель контролирует использование своего произведения и вправе потребовать компенсацию за его неправомерное использование. Возможны различные варианты договоров: лицензионный договор с частичным использованием прав, договор об отчуждении с передачей всех прав новому владельцу или свободное использование произведения после истечения срока охраны исключительного права (70 лет со дня смерти автора).**

**Все эти аспекты требуют тщательной фиксации в документах, что делает автоматизацию и снижение человеческого фактора важной задачей при разработке программного обеспечения.**

**Кроме того, художественная галерея должна быть ориентирована на зрителя.**

* 1. ****Анализ бизнес-процессов художественной галереи****

Основными и неотъемлемыми функциями художественной галереи можно назвать управление коллекцией произведений искусства, организацию и проведение выставок, продажу и бронирование произведений, а также обеспечение удобного взаимодействия между художником и покупателем. Эти процессы являются ключевыми для успешной работы галереи, так как обеспечивают не только сохранность произведений искусства, но и их доступность для широкой аудитории.

Управление коллекцией произведений искусства включает в себя подробный учет данных о каждом произведении, определение особенностей их индивидуального хранения, хранение параметров температурного режима и уровня влажности для разных видов произведений искусства. Это необходимо для сохранения экспонатов в идеальном состоянии. Также важным аспектом является выбор помещений с оптимальными показателями для хранения произведений в перерывах между выставками. Постоянный мониторинг безопасности и состояния каждого объекта – еще одна важная часть этого процесса. Для удобства управления коллекцией необходима подробная база данных, которая должна обеспечивать быстрый доступ ко всей информации об экспонатах. В эту базу данных включается описание выясненного провенанса экспоната, его текущего состояния, возможных повреждений, оптимальных условий хранения, используемых материалов, размеров и веса, наименования, имени автора, а также информации о страховании. Кроме того, управление коллекцией подразумевает организацию логистики для перемещения экспонатов между экспозициями и их подготовку к передаче в случае продажи.

Организация и проведение выставок – это еще один важный аспект работы художественной галереи. В этот процесс входит создание плана по подготовке и реализации выставки, выбор тематики и концепции, исходя из ресурсов галереи и интересов аудитории. Особое внимание уделяется отбору произведений искусства и переговорам с их авторами или правообладателями, рассматривая галерею как посредника между художником и покупателем. Подготовка включает распределение произведений по залам, разработку логики и маршрута для посетителей, а также создание материалов, таких как таблички с описаниями, каталоги, мультимедийный контент и адаптация информации для различных языков.

Продажа и бронирование произведений искусства – еще один важный процесс. Для удобства покупателей менеджеры галереи должны предоставить каталог экспонатов с возможностью поиска по различным критериям.

Автоматизация рутинных процессов существенно повысит эффективность работы галереи. К таким процессам относятся учет и инвентаризация экспонатов, управление выставками, ведение клиентской базы, финансовая отчетность и подготовка юридической документации. Без автоматизации возрастает вероятность ошибок, увеличивается время на выполнение задач, а работа между отделами становится менее скоординированной. Внедрение информационных технологий позволит не только ускорить выполнение задач, но и сократить затраты, минимизировать ошибки и улучшить взаимодействие между всеми участниками процесса.

Таким образом, автоматизация рутинных процессов и внедрение современных технологий – это важный шаг для развития галереи, который позволит обеспечить высокое качество услуг и увеличить её конкурентоспособность.

* 1. ****Целевые пользователи и их потребности****

**Целевыми пользователями художественной галереи можно считать партнеров, инвесторов, коллекционеров, галеристов и художников. Каждый из них имеет свои уникальные интересы, предпочтения и ожидания. Галерея, стремящаяся к успешной деятельности, должна находить баланс между этими интересами, предлагая решения, которые удовлетворяют запросы всех сторон.**

**Галерея, чтобы оставаться востребованной и успешной, должна быть не просто местом для продажи предметов искусства, но также центром культурной жизни, площадкой для экспериментов и уникальных идей. Интересная и хорошо организованная галерея всегда чем-то выделяется среди других. Она не должна напоминать обыкновенный магазин, только с предметами искусства. Напротив, галерея обязана стать пространством, которое объединяет искусство и концепцию, раскрывая произведения через идеи и философские осмысления. Именно оригинальное представление работ художников и их нестандартная подача имеют особую ценность для галеристов.**

**Галеристы, как профессионалы, стремятся не только продвигать произведения искусства, но и формировать культурное наследие. Их интересует, чтобы предметы искусства, представленные в галерее, были тщательно подобраны, объединив их в общую идею, которая бы раскрывала смысл и потенциал работ. Такой подход способствует созданию уникальной атмосферы, привлекающей внимание зрителей и покупателей.**

**Для партнеров, инвесторов и коллекционеров важнейшее значение имеют провенанс произведений искусства, их состояние и репутация галереи. Провенанс позволяет подтвердить подлинность произведения и его историю, что создает ощущение уверенности и доверия у заинтересованных сторон. Надежная репутация галереи становится основой для привлечения новых клиентов, а также способствует укреплению уже существующих связей. Организация закрытых мероприятий, которые дают возможность личного общения с художниками, погружения в мир их творчества и идей, также играет важную роль. Подобные события помогают партнерам и коллекционерам чувствовать себя вовлеченными в искусство, а также укрепляют их эмоциональную связь с произведениями и галереей в целом.**

**Художники, как одни из ключевых участников галерейного процесса, также имеют свои ожидания. Для них крайне важно признание, поэтому участие в выставках и мероприятиях галереи должно быть ярким и запоминающимся. Громкие проекты и смелые эксперименты способны привлечь внимание к художнику и его творчеству, что в дальнейшем может привести к сотрудничеству с крупными институциями, спонсорами, коллекционерами, СМИ и другими галереями. Художники видят в галерее не только способ продвижения своих работ, но и возможность заявить о себе, стать частью культурного сообщества и добиться новых профессиональных высот.**

**Таким образом, успешная художественная галерея должна стремиться к построению многоуровневой экосистемы, в которой учитываются интересы всех участников. Это предполагает создание доверительных отношений с коллекционерами и инвесторами, укрепление репутации галереи через профессиональный подход к провенансу и состоянию произведений, а также предоставление художникам возможности для роста и самореализации. Баланс интересов, ориентация на уникальность и инновации, а также создание ярких событий позволят галерее не только выделяться среди конкурентов, но и сохранять свою актуальность и значимость в долгосрочной перспективе.**

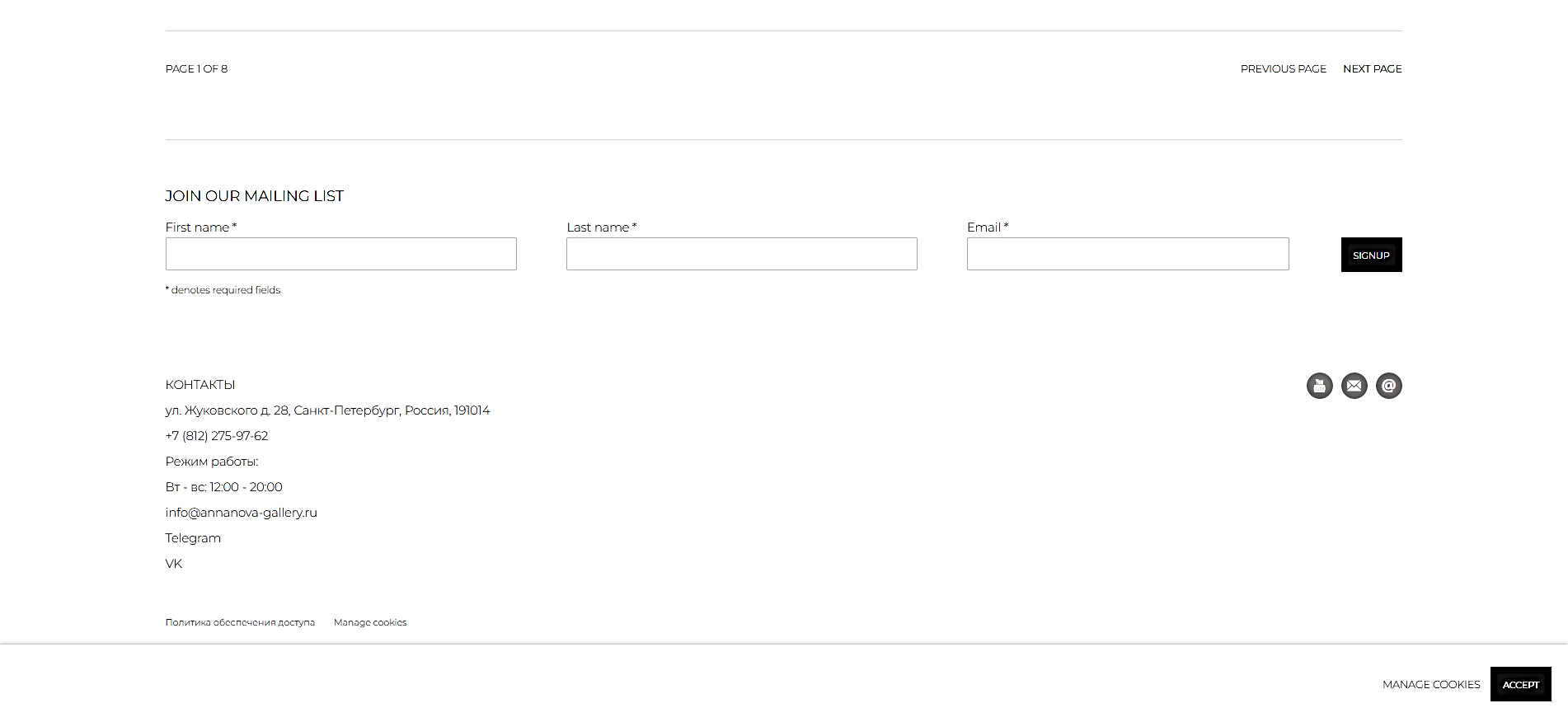
* 1. ****Анализ аналогов и существующих решений****

**Для создания эффективного и удобного программного обеспечения важно опираться на существующий пользовательский опыт и избегать повторения ошибок. Далее рассмотрены аналоги различных типов, которые могут быть полезны для анализа и сравнения. Особое внимание уделено решениям галерей и платформ, поддерживающих коллекционеров и художников, чтобы выявить их преимущества и недостатки.**

**Так как разрабатываемое программное обеспечение (ПО) будет использоваться для управления галереей, имеет смысл изучить, как другие галереи представляют себя в цифровом пространстве. Ниже приведены примеры нескольких известных галерей.**

***Галерея Anna Nova****[[1]](#footnote-1)*

**Галерея Anna Nova одна из ведущих институций в области современного искусства, основанная в 2005 году в Санкт-Петербурге. Сотрудничает с российскими и зарубежными художниками и кураторами, создавая смелые и новаторские проекты, обращаясь к основным тенденциям актуального искусства.** [[2]](#footnote-2)

****

**Рисунок 1.1 – Интерфейс «AnnaNova»**

**Преимущества галереи Anna Nova:**

1. Минималистичный дизайн сайта, который создает благоприятное впечатление;
2. Продуманная сортировка предметов искусства по категориям, таким как автор, размер, цена и другим. Это упрощает навигацию для посетителей;
3. Возможность записи на мероприятия или выставки с помощью сайта. Это повышает удобство для пользователей.

**Недостатки галереи:**

1. Отсутствие мобильного приложения, что ограничивает удобство использования для пользователей мобильных устройств;
2. Мало информации о выставках и художниках на других языках, что может затруднить привлечение иностранной аудитории;
3. Не реализован функционал персонализации, например, рекомендации на основе интересов пользователя.

*Artsy[[3]](#footnote-3)*

Artsy для коллекционирования произведений искусства. Как ведущая торговая площадка для произведений искусства от начинающих и признанных мировых художников, мы упростили для новых и опытных коллекционеров поиск, покупку и продажу произведений искусства и многое другое. Все, **что вам когда-либо понадобится для коллекционирования произведений искусства, вы найдете** на Artsy.

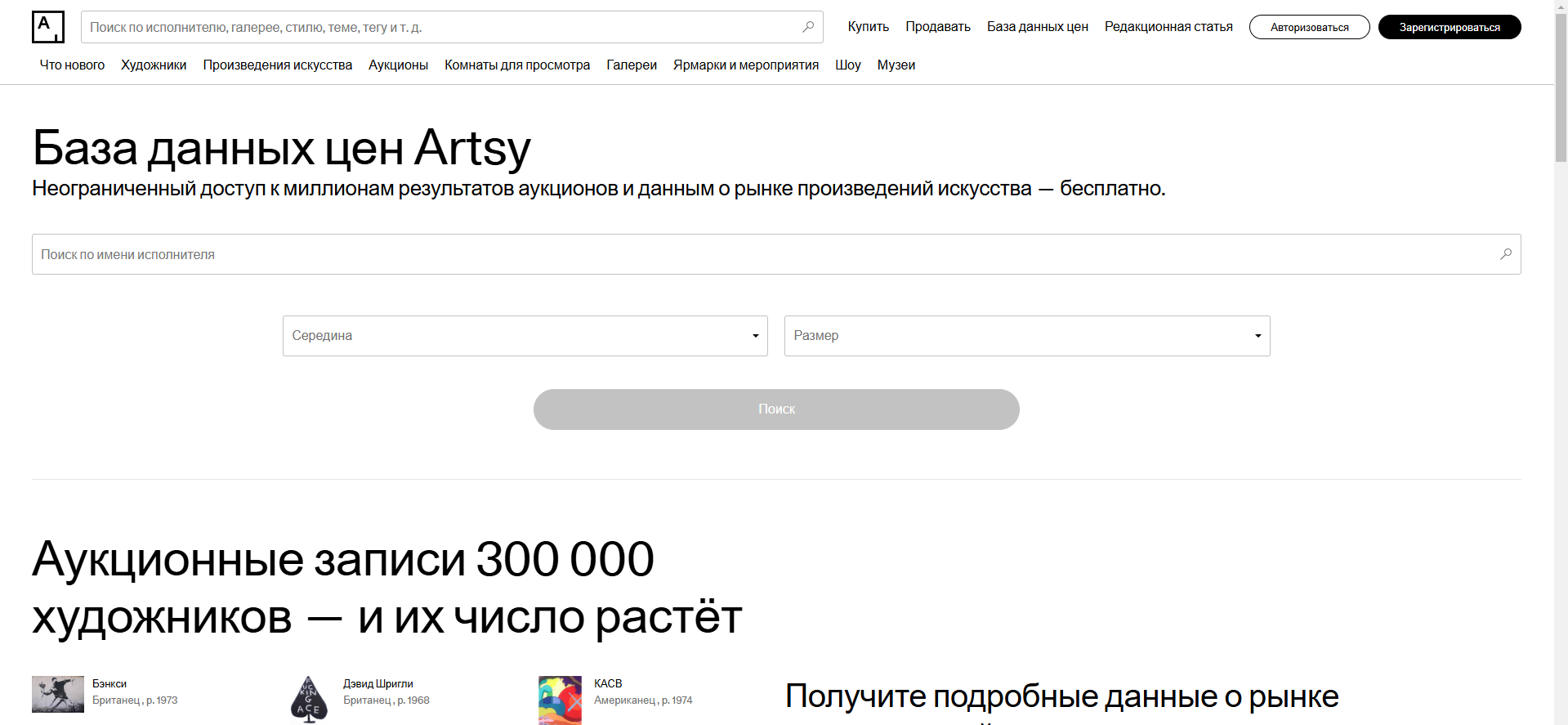
Преимущества Artsy:

Artsy может предотавить доступ к произведениям всех художников. И неизвестных, и всемирно признанных. Это является привлекательной чертой для всех коллекционером, не зависимо от их бюджета.

1. Покупка и продажа произведений искусства на этой платформе очень понятна и интуитивно понятна, и очень удобна. Это помогает всем пользователям быстрее привыкнуть к сайту;
2. У каждого объекта есть высококачественные снимки и детальной и очень информативное описание, это помогает пользователям взвешивать и обдумывать свою покупку в наилучших условиях;
3. На платформе есть множество статей, интервью с художниками и экспертами, что помогает пользователям узнавать что-то новое и углублять свои знания о современном искусстве.

**Недостатки Artsy:**

1. Комиссии за покупку и продажу могут быть высокими, что снижает привлекательность покупки и выгоду для некоторых коллекционеров;
2. Из-за отсутствия возможности лично увидеть и оценить произведение, покупка может быть сопряжена с рисками для коллекционеров. На платформе представлено множество произведений, и не всегда легко выделиться или найти нужное среди множества вариантов;
3. В случае возникновения проблем с произведением искусства могут быть трудности с возвратом или компенсацией, так как сделка проходит через онлайн-платформу.



**Рисунок 1.2 – Интерфейс «Artsy»**

*КАМИС[[4]](#footnote-4)*

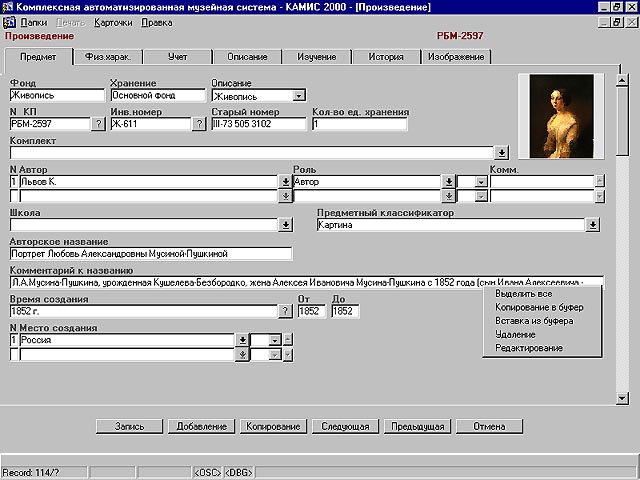
КАМИС – самая мощная современная музейная информационная система, обеспечивающая решение широкого круга музейных задач. С помощью КАМИС в музеях создаются интегрированные базы данных музейных коллекций, включающие различные среды (тексты, изображения, аудио, видео, анимацию), объединенные гипермедийными ссылками. Она реализована на современной СУБД Oracle. КАМИС структурирует информацию, позволяет осуществлять быстрый поиск и выборку данных по всем атрибутам и их сочетаниям, вести учетные операции и подготавливать различные виды списков, каталогов, документов, а также электронные интерактивные публикации, сводные базы данных с доступом из сети Интернет.

Преимущества КАМИС:

1. Позволяет создавать интегрированные базы данных, содержащие тексты, изображения, аудио, видео и анимацию, что дает музеям возможность эффективно управлять коллекциями;
2. Возможность объединения различных сред с помощью гипермедийных ссылок для создания интерактивных публикаций и улучшения пользовательского опыта;
3. Быстрый и точный поиск по атрибутам и их сочетаниям помогает эффективно находить нужные данные.

Недостатки КАМИС:

1. Из-за высокой функциональности и технологической сложности система может требовать значительных временных и финансовых затрат на внедрение;
2. Использование Oracle[[5]](#footnote-5) может увеличивать стоимость лицензирования и обслуживания;
3. Зависимость от разработчика системы для получения своевременных обновлений и устранения ошибок.

****

**Рисунок 1.3 – Интерфейс «КАМИС»**

**Для изучения существующих решений, более ориентированных на базы данных экспонатов, поддержку провенанса и художников, далее будут сравниваться не галереи, а платформы-посредники между коллекционерами и художниками.**

**Таблица 1 – Сравнение аналогов**

|  |  |  |  |
| --- | --- | --- | --- |
| **Показатель** | **AnnaNova** | **Artsy** | **КАМИС** |
| **Назначение** | **Система для управления коллекциями произведений искусства, выставками и архивами определенной галереи.** | **Веб-платформа для взаимодействия коллекционеров, галерей и художников, позволяющая искать и продавать работы.** | **Комплексное решение для автоматизации музейной деятельности, включающее учет экспонатов, реставрационные работы и выставки.** |
| **Основные функции** | **Ведение базы данных экспонатов, организация выставок, отслеживание перемещений и состояния произведений.** | Поиск и каталогизация произведений, организация продаж, интеграция с социальными сетями и онлайн-выставки. | Учет и систематизация экспонатов, контроль провенанса, ведение документации, подготовка отчетности и управление реставрацией. |
| **Целевая аудитория** | Галереи и музеи, нуждающиеся в систематизированном управлении коллекциями и организации выставок. | **Коллекционеры, галеристы, художники и любители искусства, ориентированные на глобальное взаимодействие.** | Музеи и крупные культурные учреждения, которым требуется глубокая автоматизация внутренних процессов. |
| **Географический охват** | Преимущественно локальный или региональный (российский рынок). | Международный, с глобальной базой пользователей и клиентов. | Национальный (акцент на российские музеи) с возможностью адаптации под локальные требования. |
| **Технологическая база** | Специализированное ПО, ориентированное на задачи учета и организации выставочной деятельности. | Веб-ориентированная платформа с динамической базой данных, поддерживающая мультимедийный контент и интерактивность. | Интегрированная система, построенная на корпоративных СУБД (Oracle), с акцентом на надежность и масштабируемость. |
| **Пользовательский интерфейс** | Интуитивный и минималистичный дизайн, упрощающий навигацию и работу с базой данных экспонатов. | Современный, адаптивный интерфейс, оптимизированный под различные устройства и обеспечивающий высокую юзабилити. | Классический и функциональный интерфейс, ориентированный на профессиональное использование в музейной сфере. |
| **Масштабируемость** | Подходит для небольших и средних галерей; расширяемость может быть ограничена при резком росте коллекции. | Высокая масштабируемость, позволяющая обслуживать как небольшие коллекции, так и огромные базы данных произведений искусства. | Высокая масштабируемость, позволяющая эффективно управлять большими музейными коллекциями и адаптироваться под рост учреждения. |
| **Обновляемость** | Регулярное обновление контента, но с меньшим числом функциональных новшеств по сравнению с веб-платформами. | Постоянное обновление функционала и расширение возможностей за счет интеграции с новыми сервисами и технологиями. | Систематическое обновление с учетом специфики музейной деятельности, ориентированное на долгосрочную эксплуатацию и стабильность. |
| **Стоимость внедрения** | Относительно доступное решение для галерей с небольшим бюджетом, легкость начальной интеграции. | Возможны значительные затраты в зависимости от объема функционала и масштабов работы платформы. | Высокая первоначальная стоимость внедрения, компенсируемая экономией за счет автоматизации и оптимизации музейных процессов. |
| **Поддержка и обучение** | Предоставляется базовая техническая поддержка, обучение персонала осуществляется за счет внутренних ресурсов галереи. | Широкий спектр онлайн-материалов, обучающих программ и поддержка сообщества пользователей. | Требуется специализированное обучение для сотрудников музея, поддержка осуществляется разработчиками с учетом специфики учреждения. |

* 1. ****Требования к функционалу приложения****

**Для успешной работы приложения и его интеграции в повседневные процессы галереи оно должно быть удобным и интуитивно понятным как для персонала галереи, так и для внешних пользователей, взаимодействующих с ним. Ролей всего три, это администратор, работник галереи и посетитель. Администратор может заводить новые товары и новых пользователей, а также изменять существующие. Работники могут писать комментарии к отдельным экспонатам, которые потом попадают к администратору на рассмотрение. Клиенты могут только просматривать приложение.**

**Одной из ключевых функций приложения должна стать реализация подробной и организованной базы данных, содержащей информацию обо всех экспонатах галереи. Эта база данных обязана предоставлять возможность полного отслеживания провенанса (истории владения и передвижения произведения искусства). Такая функция крайне важна для поддержания «чистой» репутации галереи и обеспечения доверия со стороны коллекционеров, партнеров и других заинтересованных сторон.**

**База данных должна быть удобной для восприятия, с простым интерфейсом, позволяющим легко находить и обрабатывать нужную информацию. Для повышения эффективности работы приложение должно включать инструменты автоматического формирования отчетов. Эти отчеты должны генерироваться в различных форматах, включая возможность экспорта данных в файлы Excel, что упростит дальнейшую обработку информации или передачу ее другим пользователям.**

**Кроме того, приложение должно поддерживать сортировку экспонатов по различным критериям, таким как автор, дата создания, цена, выставки и другое. Это поможет как сотрудникам галереи, так и коллекционерам быстро находить нужные данные, минимизируя временные затраты.**

* 1. ****Технологический стек****
     1. ****Выбор системы управления базой данных****

**Существует несколько систем управления баз данных, они подразделяются на: Реляционные СУБД,** Документные **СУБД, Ключ-значение СУБД, Графовые СУБД, Колонко-ориентированные СУБД, Объектно-ориентированные СУБД.**

**Реляционные СУБД – это СУБД, где данные хранятся, как таблицы и их структура заранее определена. К таким относятся: Microsoft SQL Server, MySQL, PostgreSQL, Oracle Database.**

Документные СУБД – это СУБД, основанные на хранении баз данных, как документов. Примером такой СУБД может быть MongoDB. Ключ-Значения СУБД – СУБД с простой структурой, где первичные ключи ассоциируются со значениями, например Amazon DynamoDB.

Графовые СУБД – это такие СУБД, которые обычно используются в социальных сетях для данных с множественными взаимосвязями, самая знаменитая Amazon Neptune.

Объектно-ориентированные СУБД поддерживают объектные структуры данных.

Колонко-ориентированные СУБД оптимизированы для аналитических запросов.

Для создания приложения для галереи больше всего подходят, либо реляционные СУБД, либо Документные СУБД.

**Таблица 2** – **Сравнение реляционных СУБД и документных СУБД**

|  |  |  |
| --- | --- | --- |
| **Характеристика** | **Реляционные СУБД** | **Документные СУБД** |
| **Хранение данных** | Таблицы с заранее заданной схемой. | Документы (обычно JSON). |
| **Гибкость структуры** | Строгая, заранее заданная схема. | Высокая, структура документов может различаться |
| **Сложность запросов** | Используют SQL. | Менее сложный, поддерживает NoSQL |
| **Примеры** | Microsoft SQL Server, MySQL. | MongoDB, Couchbase |
| **Применение** | Финансы, ERP, CRM. | Big Data, приложения с быстрым изменением структуры данных |
| **Масштабируемость** | Вертикальная | Горизонтальная |
| **Производительность** | Высокая для транзакций | Высокая для больших объемов неструктурированных данных |

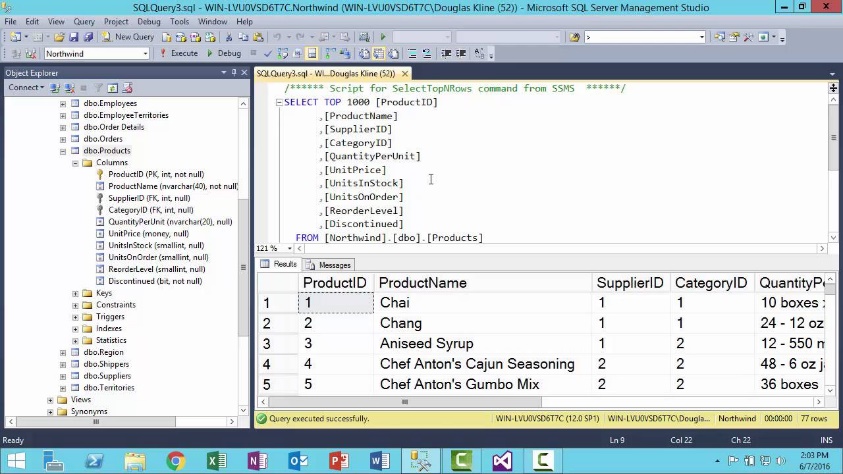
**Было отобрано несколько СУБД для выбора из них самой подходящей для этого проекта.**

**Таблица 3** – **Сравнение СУБД**

|  |  |  |  |
| --- | --- | --- | --- |
| **База данных** | **Тип** | **Преимущества** | **Недостатки** |
| **Microsoft SQL Server** | Реляционная | Высокая производительность, интеграция с продуктами Microsoft, продвинутые функции безопасности | Высокая стоимость лицензий, завязка на экосистему Microsoft |
| **MySQL** | Реляционная | Бесплатная (Community Edition), простота использования, широкая поддержка сообществом | Ограниченная функциональность в бесплатной версии, менее эффективна для масштабируемых систем |
| **PostgreSQL** | Реляционная | Открытый исходный код, высокая совместимость с другими системами, поддержка сложных запросов | Более сложное администрирование и настройка по сравнению с другими СУБД |
| **Oracle Database** | Реляционная | Высокая производительность, поддержка крупных корпоративных проектов | Высокая стоимость лицензий, сложность освоения и администрирования |
| **MongoDB** | Документная | Гибкость при работе с неструктурированными данными, масштабируемость | Ограничения в сложных запросах, необходимость адаптации для работы с реляционными системами |
| **SQLite** | Реляционная | Простота и портативность, встроенная в приложения | Ограничения в масштабируемости, не подходит для многопользовательских системы |

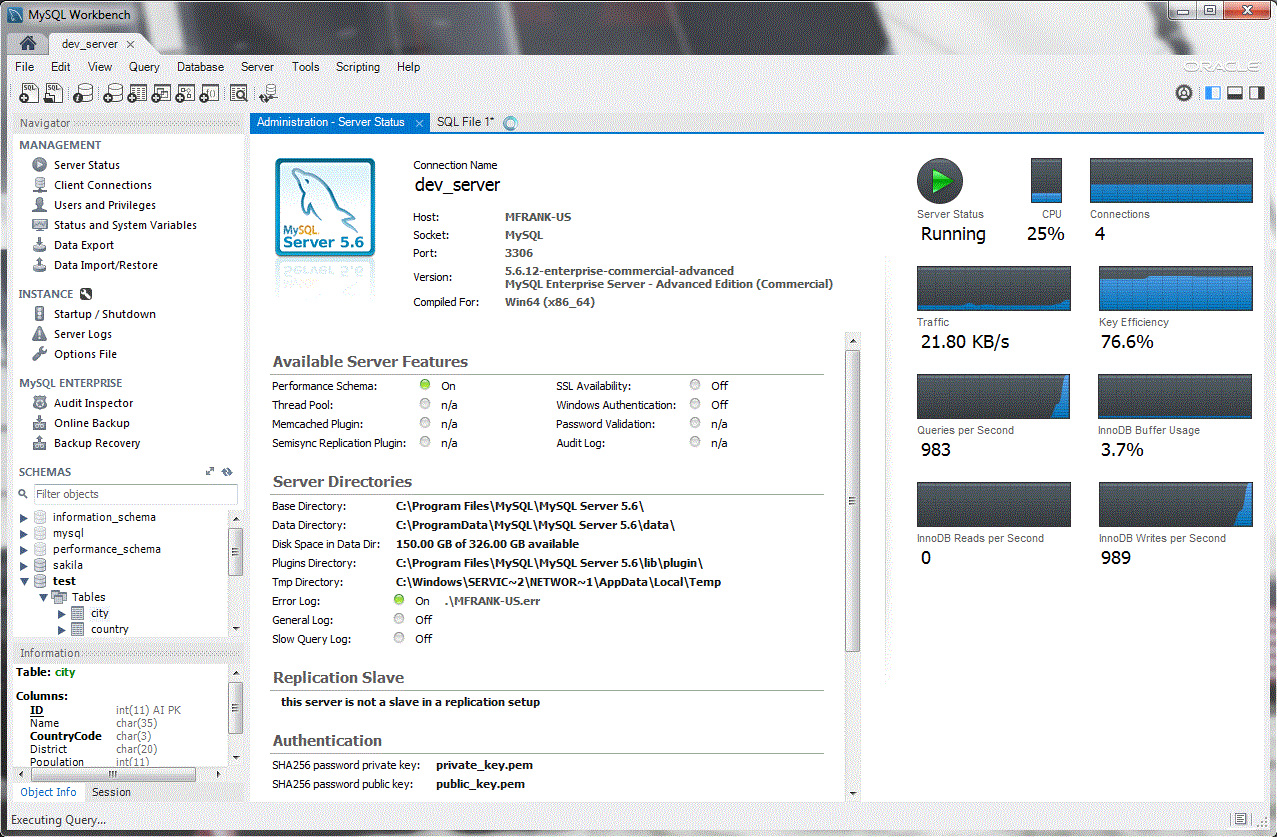
***Microsoft SQL Server***

**Microsoft SQL Server – это мощная реляционная система управления базами данных, обеспечивающая высокую скорость работы и расширенные возможности защиты. Она легко интегрируется с продуктами Microsoft, что делает её удобной для корпоративного использования. Бесплатная версия (SQL Server Express) позволяет разрабатывать приложения без финансовых затрат на лицензии.**



**Рисунок 1.4 – Интерфейс Microsoft SQL Server**

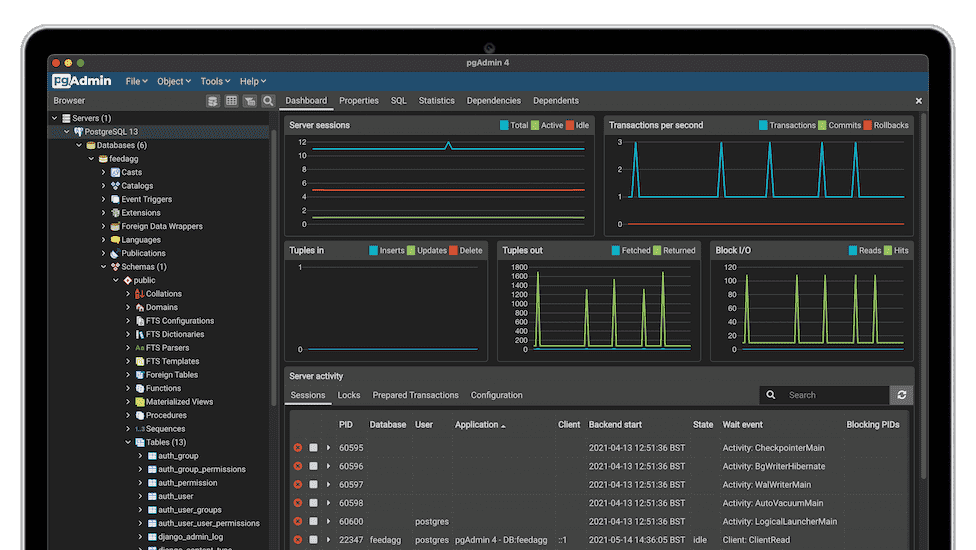
***MySQL***

**MySQL – популярная реляционная СУБД с открытым кодом, известная своей лёгкостью в освоении и высокой скоростью обработки данных. Её поддерживает большое сообщество разработчиков, и она широко применяется в веб-разработке, особенно совместно с PHP. Однако бесплатный вариант имеет определённые ограничения по сравнению с коммерческими версиями.** 

**Рисунок 1.5 – Интерфейс MySQL**

***PostgreSQL***

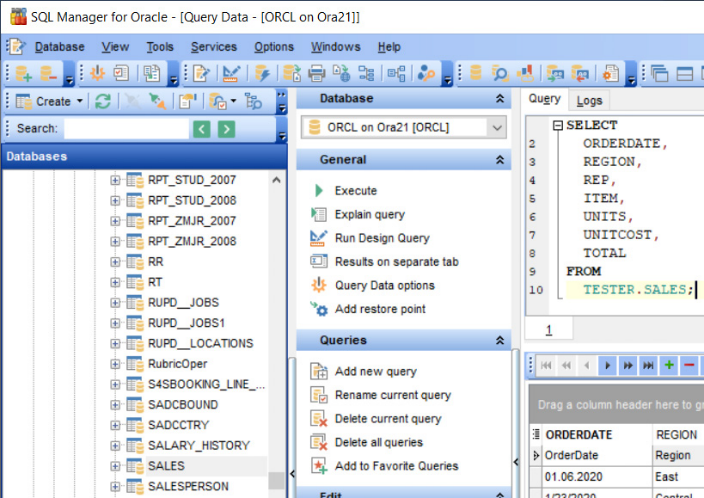
**PostgreSQL – это мощная СУБД с открытым исходным кодом, поддерживающая сложные SQL-запросы и возможность расширения функционала с помощью дополнительных модулей. PostgreSQL отличается высокой стабильностью и надёжностью, но требует более детальной настройки и управления. PostgreSQL часто применяется в аналитических системах и крупных корпоративных проектах.**



**Рисунок 1.5 – Интерфейс PostgreSQL**

***Oracle Database***

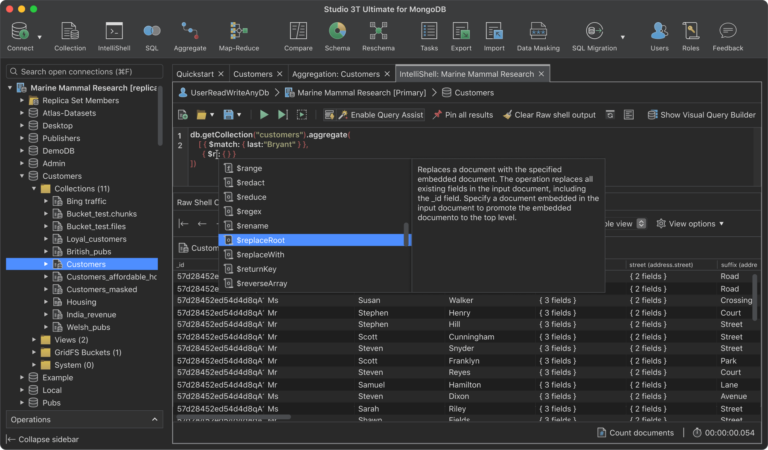
**Oracle Database – это высокопроизводительная реляционная база данных, предназначенная для работы с крупными объёмами информации в корпоративных средах. Она поддерживает масштабирование, сложные транзакционные операции и отличается высокой надёжностью. Однако её использование требует значительных затрат на лицензирование и хорошей подготовки специалистов.**



**Рисунок 1.6 – Интерфейс Oracle Database**

***MongoDB***

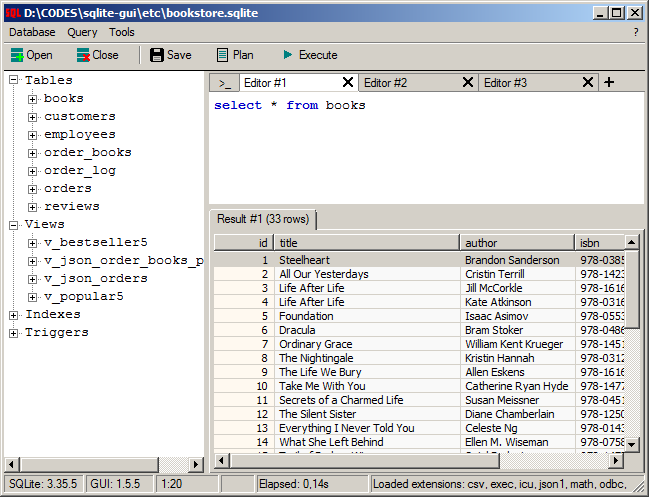
**MongoDB – это система управления базами данных, ориентированная на работу с документами, где информация хранится в формате JSON-подобных объектов. PostgreSQL хорошо подходит для веб-приложений, требующих гибкой структуры данных и лёгкого масштабирования. Однако обработка сложных запросов в этой системе менее эффективна, чем в классических реляционных системах.**



**Рисунок 1.7 – Интерфейс MongoDB**

***SQLite***

**SQLite – это встраиваемая реляционная база данных, не требующая отдельного сервера и идеально подходящая для мобильных и настольных приложений. Её основными преимуществами являются простота в использовании и отсутствие необходимости в сложной конфигурации. Однако из-за ограниченной производительности она не подходит для многопользовательских проектов с высокой нагрузкой.**



**Рисунок 1.8 – Интерфейс SQLite**

**Была выбрана Microsoft SQL Server. В ней оптимально сочетание цены и функционала, версия Microsoft SQL Server бесплатна, что помогает избежать трат на лицензию, при этом она предоставляет все базовые необходимые функции. Проект создается на базе WPF и Microsoft SQL Server идеально с ней интегрируется, это упрощает разработку и поддержку приложения. Для небольшой галереи не требуется сложной и дорогой СУБД. В Microsoft SQL Server отсутствует сложная система и она предоставляет удобный инструмент для управления базой данных без лишних затрат и долгого обучения. Также, когда потребуется расширить функционал или изменить масштаб проекта, можно просто перейти на более мощную версию Microsoft SQL Server, например SQL Server,** **сохранив при этом все данные и настройки.** **Microsoft SQL Server может обеспечить высокую производительность, особенно при работе с транзакциями и она подходит для надежного хранения данных, а это необходимо для управления коллекцией произведений искусств.**

**Галерея** – **это небольшой проект с относительно простыми требованиями к базе данных. Использование Microsoft SQL Server позволит сосредоточиться на разработке приложения, не беспокоясь о сложностях настройки и администрирования базы данных.**

**Самый ближайший аналог разрабатываемого ПО** – это КАМИС. Она работает на базе **Oracle Database и это делает её мощной и гибкой для управления большими и сложными музейными коллекциями.**

**Преимущества Oracle Database:**

1. **Она обеспечивает высокий уровень устойчивости к сбоям;**
2. **Также она подходит как для локальных решений, так и для облачных систем;**
3. **Располагает широкими возможности шифрования данных и защиты от угроз.**

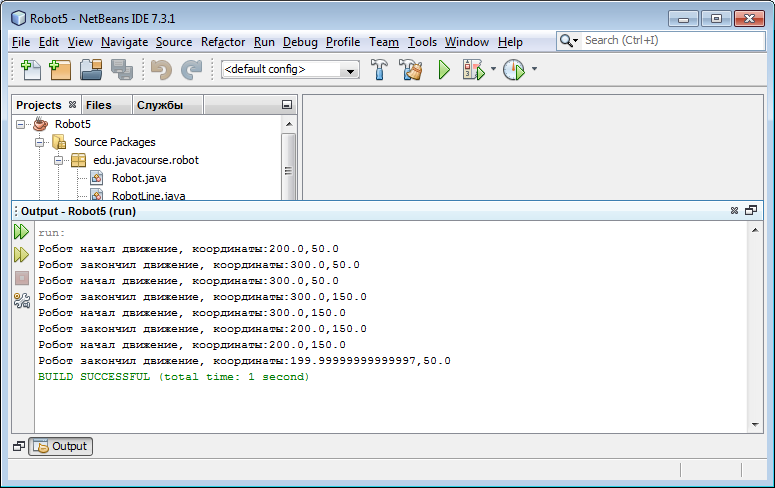
**Недостатки Oracle Database:**

* 1. **Высокая цена лицензий и поддержки делает её дорогим решением, что не подходит для галереи;**
  2. **Требует высокой квалификации администраторов для настройки и управления, что усложняет процесс обучения сотрудников и замедляет обновления;**
  3. **Меньше гибкости при интеграции с другими платформами по сравнению с открытыми решениями, такими как PostgreSQL и Microsoft SQL Server.**
     1. ****Выбор языка программирования****

**При создании приложения для внутренней работы галереи с удобным доступом к базе данных важно выбрать подходящий язык программирования.**

***Java***

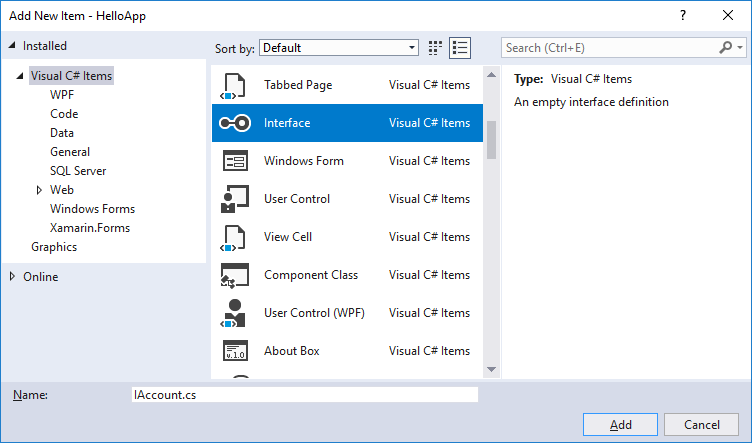
**Язык Java помогает обеспечить «платформенную независимость», так как программы, написанные на нем, работают в виртуальной машине (JVM**[[6]](#footnote-6)**) и это обеспечивает возможность запуска на различных ОС. У Java образовалось довольно широкое сообщество разработчиков и большое количеством библиотек для облегчения создания корпоративных приложений. Но при этом, для создания нативных**[[7]](#footnote-7) **настольных приложений под Windows его возможности ограничены, а производительность может уступать компилируемым языкам.**



**Рисунок 1.9 – Синтаксис языка программирования Java**

***C#***

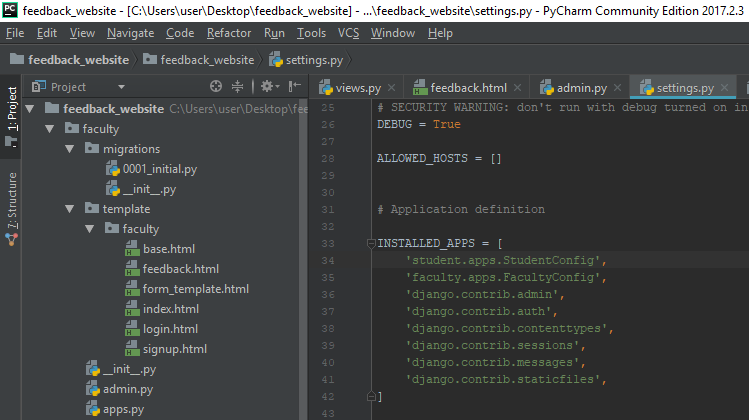
**C# – это язык высокого уровня, обладающий мощной интеграцией с платформой .NET. Его преимущества заключаются в автоматическом управлении памятью, строгой типизации и возможности создания масштабируемых и надежных приложений. Интегрированная среда разработки Visual Studio предоставляет обширный набор инструментов для отладки, рефакторинга**[[8]](#footnote-8) **и визуального проектирования, что значительно ускоряет процесс создания и тестирования программного продукта.**

****

**Рисунок 1.****10 – Синтаксис языка программирования C#**

***Python***

**Python известен своей простотой и лаконичностью, этот язык часто используется для недолгой в разработке реализации прототипов и создания функционала, который не требует слишком высоких вычислительных ресурсов. Его огромная экосистема модулей и библиотек позволяет решать разнообразные задачи. Но несмотря на это, язык не всегда способен обеспечить достаточную скорость работы и качественную поддержку сложных пользовательских интерфейсов, что особенно видно в приложениях, ориентированных на обработку больших массивов данных и требующих взаимодействия с ОС на глубоком уровне Windows.**



**Рисунок 1.11 – Синтаксис языка программирования Python**

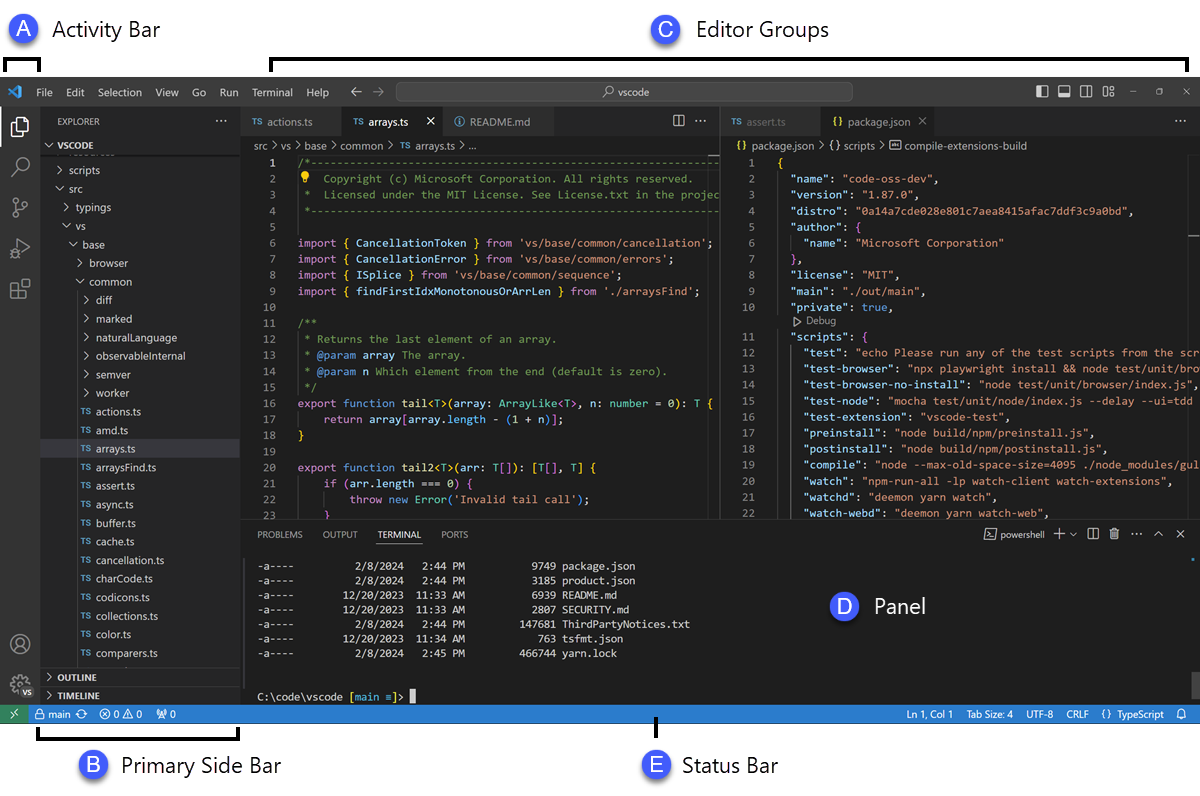
**Таблица 5 – Сравнение языков программирования**

|  |  |  |  |
| --- | --- | --- | --- |
| **Показатель** | **Java** | **C#** | **Python** |
| **Производительность** | **Хорошая производительность, но уступает компилируемым языкам в ресурсоемких приложениях.** | **Отличная производительность для бизнес-приложений, оптимизированных под Windows.** | **Производительность ниже, особенно в вычислительно интенсивных сценариях.** |
| **Управление памятью** | **Автоматическое благодаря сборщику мусора, но может приводить к непредсказуемым задержкам.** | **Автоматическое управление памятью снижает вероятность ошибок, связанных с ресурсами.** | **Автоматизированное, однако накладные расходы могут сказываться на скорости исполнения.** |
| **Сложность синтаксиса** | **Объектно-ориентированный, синтаксис понятен, однако может быть громоздким в крупных проектах.** | **Современный и лаконичный синтаксис, упрощающий разработку, и снижает порог вхождения.** | **Очень простой и удобный для быстрого прототипирования, но недостаточно строгий для больших систем.** |
| **Поддержка UI** | **Не ориентирован непосредственно на нативный Windows UI, требует дополнительных библиотек.** | WPF предоставляет мощный и гибкий набор инструментов для создания современных UI под Windows. | **Возможна реализация через сторонние библиотеки, но функциональность ограничена.** |
| **Инструменты разработки** | Поддерживается различными средами (Eclipse, NetBeans), однако интеграция с Windows не на высоте. | Глубокая интеграция с Visual Studio, что обеспечивает богатый функционал для отладки и дизайна. | Широкий выбор редакторов и IDE, но отсутствует глубокая интеграция с платформой Windows. |
| **Кроссплатформенность** | **Благодаря JVM работает на разных платформах, но нативный вид на Windows может быть утрачено.** | Основная направленность – Windows, что является плюсом для приложений с WPF и узконаправленным функционалом. | Универсальный язык с кроссплатформенной поддержкой, но не всегда соответствует требованиям настольных приложений. |
| **Сообщество и экосистема** | **Огромное сообщество, множество готовых решений и библиотек, но сложность настройки может быть помехой.** | Активное сообщество .NET, отличная поддержка от Microsoft, широкая экосистема компонентов и интеграция с корпоративными решениями. | Большое сообщество, отличная документация, однако недостаточная строгость для крупных корпоративных решений. |

**Из-за оптимального сочетания производительности C#, удобства разработки и поддержки современных технологий, этот язык с использованием WPF в Visual Studio подходит больше всего для создания надежного и масштабируемого настольного приложения с внедрением базы данных.**

* + 1. ****Выбор среды разработки****

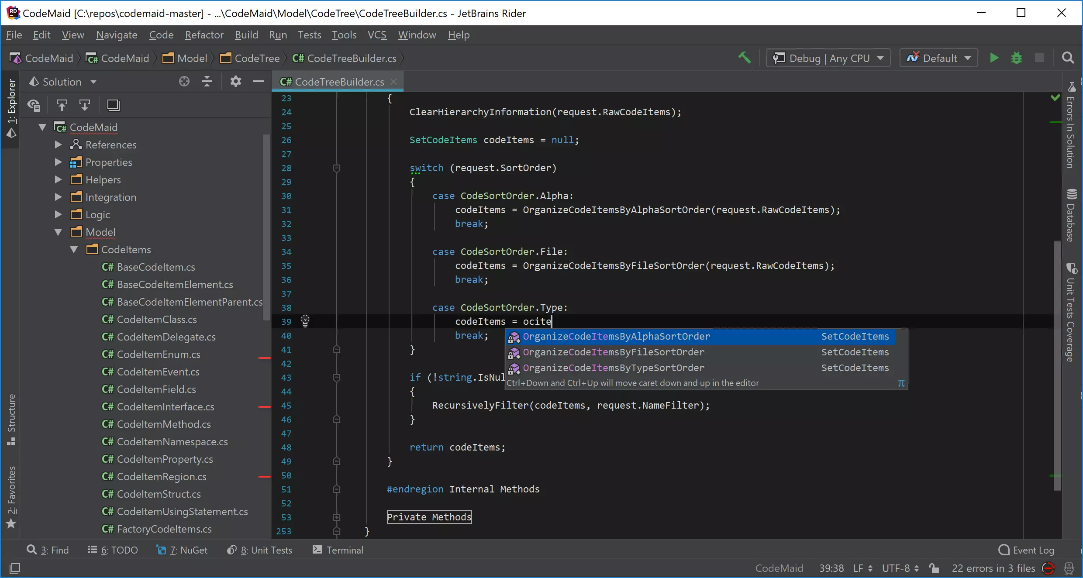
***Visual Studio***

**Visual Studio – это мощная интегрированная среда разработки (IDE), разработанная Microsoft, поддерживающая множество языков программирования, включая C#, C++ и Python. Она предоставляет широкий спектр инструментов для отладки, рефакторинга и автоматизированного тестирования, что делает её удобной для профессиональных разработчиков. Благодаря интеграции с Azure и GitHub, она отлично подходит для создания корпоративных и облачных приложений.**

**Рисунок 1.12 – Интерфейс Visual Studio**

***JetBrains Rider***

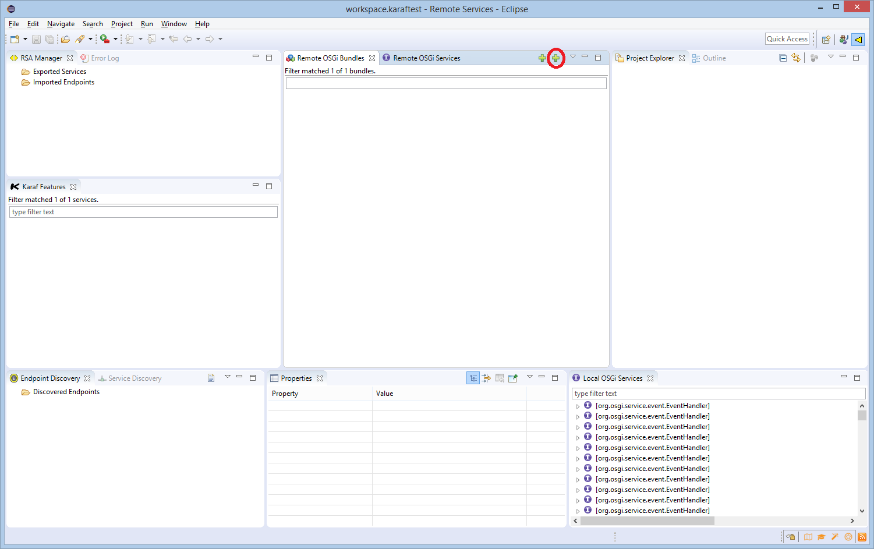
**JetBrains Rider – это современная среда разработки, ориентированная на работу с языками платформы .NET, включая C# и F#. Она отличается высокой скоростью работы и интеллектуальной системой подсказок, а также глубокой интеграцией с инструментами JetBrains, такими как ReSharper. Rider поддерживает кроссплатформенную разработку и является хорошим выбором для тех, кто использует альтернативные операционные системы, помимо Windows.**



**Рисунок 1.13 – Интерфейс JetBrains Rider**

***Eclipse***

**Eclipse – это бесплатная и кроссплатформенная среда разработки с открытым исходным кодом, поддерживающая Java, C++ и другие языки. Она обладает гибкостью за счёт системы плагинов, позволяя адаптировать функционал под конкретные задачи. Однако её производительность может зависеть от конфигурации системы, а интерфейс требует некоторого привыкания.**



**Рисунок 1.14 – Синтаксис языка программирования Eclipse**

**Была выбрана Visual Studio, так как она предоставляет наиболее удобные инструменты для работы с C# и интеграции с Microsoft SQL Server. Её мощные средства отладки, встроенная поддержка WPF и возможность расширения делают её оптимальным выбором для данного проекта.**

* 1. ****Особенности конфиденциальности и безопасности данных****

Приложение должно обеспечивать высокий уровень конфиденциальности и безопасности данных. Это особенно важно в контексте работы с произведениями искусства, стоимость которых может быть чрезвычайно высокой, а история владения часто требует защиты от несанкционированного доступа.

В приложении должна быть реализована система пользователей с подтвержденной регистрацией. Разделение пользователей на категории, такие как, сотрудники галереи, партнеры, коллекционеры, позволит регулировать доступ к различным функциям и данным. Так же будет использоваться шифрование данных. Все данные, включая провенанс, транзакции и личные данные пользователей, должны быть зашифрованы как в процессе передачи, так и на сервере. Должно регулярно проводится резервное копирование. Лучше всего, если резервные копии базы данных будут создаваться автоматически для предотвращения потери информации.

Также, так как это приложение нужно для более удобного взаимодействиям с экспонатами требуется подробная фильтрация. Какого размера предмет, какого веса, должна быть возможность отбирать экспонаты по авторам и жанрам. Все это поможет быстрее ориентироваться в коллекции и уменьшит время и нагрузку для поисков затерявшихся экспонатов и быструю отправку покупателям. Также для удобства работы в галерее, необходима минимальное взаимодействие сотрудников между собой. В приложении есть всего три роли: клиент, менеджер и администратор. Последний способен модерировать и заносить предметы в базу данных, а также сотрудников. Менеджеры имеют доступ ко всей базе данных с возможностями поиска по ней и оставления комментариев для администратора в случае нахождения ошибки в информации. И последняя роль клиента нужна на всякий случай, если понадобиться предоставить доступ к коллекции постороннему только с возможностью просмотра.

* 1. ****Проблемы текущих подходов к управлению коллекцией****

**На данный момент многие галереи продолжают использовать ручное введение данных о коллекции, что связано с рядом проблем. В первую очередь, человеческий фактор способствует возникновению ошибок в записях, что может привести к потере данных об экспонатах или их провенансе. Это не только замедляет работу, но и ставит под сомнение достоверность предоставляемой информации.**

**Процессы формирования отчетов, проверки провенанса и учета требуют значительных временных затрат. И это время увеличивается при отсутствии инструментов для быстрого поиска и анализа информации и самый главный негативный фактор – отсутствие централизованной базы данных. Также при ручном введении может получиться так, что данные могут храниться в разрозненных файлах или системах, что затрудняет доступ к информации. Невозможно быстро и четко подтвердить историю экспоната. Из-за чего могут возникнуть сомнения в достоверности информации могут повлиять на репутацию галереи.**

**Для устранения перечисленных недостатков необходимо внедрить автоматизированную систему, которая будет поддерживать единый стандарт работы с коллекцией. Такая система повысит точность учета, ускорит работу сотрудников и улучшит доверие со стороны партнеров и коллекционеров.**

1. **Структура и функциональность приложения GalleryApp**
   1. ****Общая структура приложения****

Приложение было создано на базе WPF-приложения в среде разработки Visual Studio, оно использует C#, ADO.NET, XAML, MS SQL Server и Word Interop. Основные задачи этого приложения – это упрощенная работа с товарами галереи картин, обработка разных заказов и в общем возможность вести учет товаров. Приложение специально использует самые эффективные технологии для комфортного управления данными, для создания удобного интерфейса, который будет подстраиваться под нужды всех предусмотренных ролей пользователей, а это администраторы, менеджеры и клиенты, у которых прав намного меньше, чем у остальных ролей.

В основном архитектура этого приложения построена на WPF, это помогает разработчикам обеспечить прекрасную гибкость и сделать интерфейс более отзывчивым. Было решено использовать подход, при котором абсолютно вся логика управления пользовательским интерфейсом и всякие взаимодействия с ним реализована просто в коде. В данном случае не нужно использование сложных архитектурных паттернов, именно потому было выбрано такое решение. Чтобы упростить разработку и снизить общую сложность. К тому же этот подход позволит в будущем, если, например возникнет необходимость в расширении функционала, можно будет перейти к использованию MVVM, а это поможет улучшить структуру и упростит тестируемость приложения.

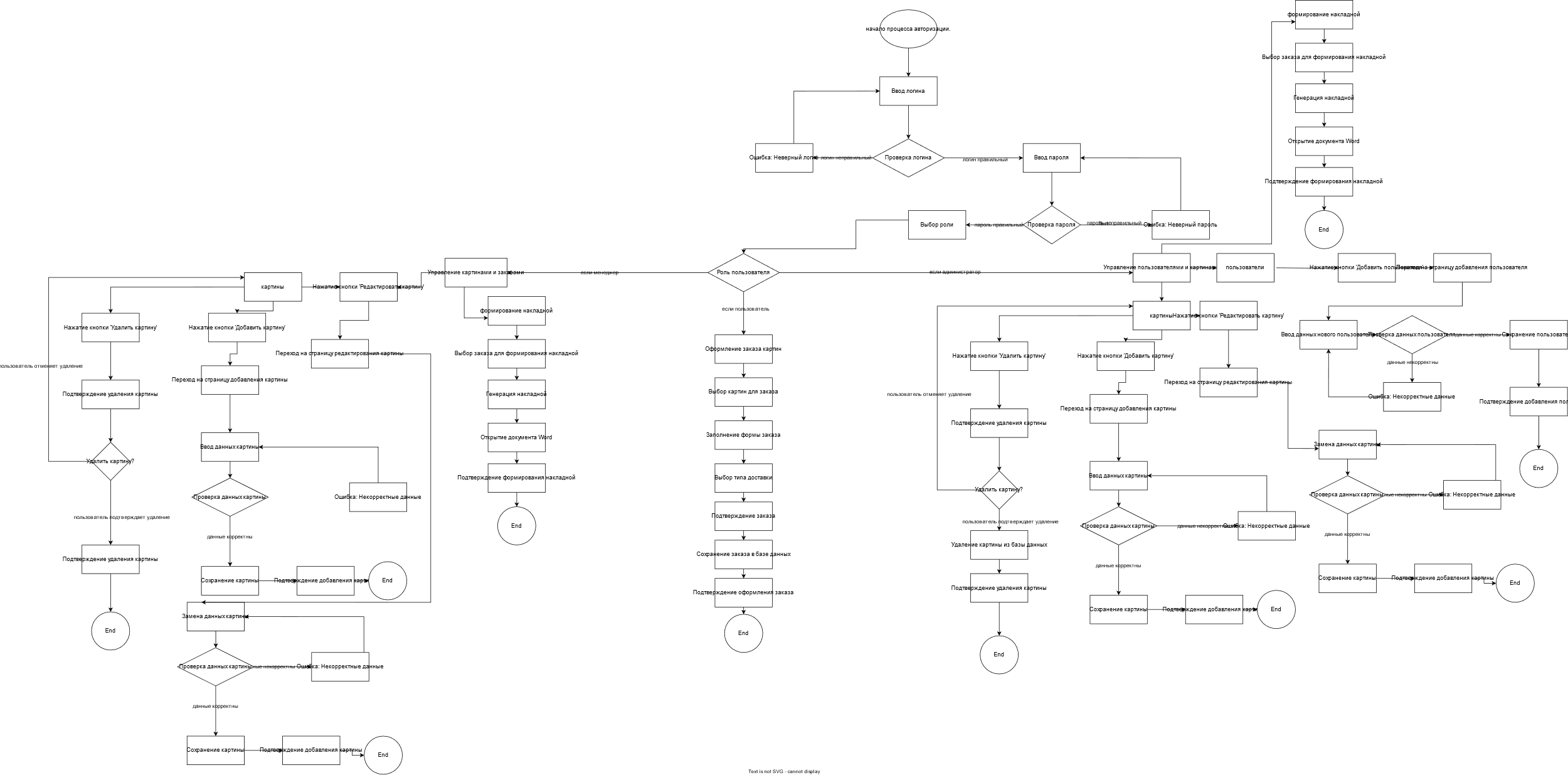
Вся работа приложения, если смотреть на неё с технической стороны, сопряжена с базой данных, названной gallerydatabase. В ней хранятся данные о пользователях, картинах, выставках, заказов – всё данные собрано в одном месте. Приложение постоянно подключается к базе данных и когда пользователь входит или выходит, и при обновлении списка товаров, так же в самом коде реализованы возможности добавления информации прямо в базу данных, а также удаление из нее.

Для связи с базой выбрана технология ADO.NET. Эта технология не является новейшей, но работает стабильно и быстро. С её помощью без лишних сложностей можно подключаться к базе данных, выполнять запросы и сохранять изменения. Кроме неё в проекте применяется Entity Framework. Благодаря этому таблицы в базе данных, можно в коде реализовать, как обычные классы, а строки, как объекты. Это ускоряет и упрощает разработку, не нужно писать SQL-запросы вручную – всё можно делать, как в обычном C#-коде.

В интерфейсе, который реализован на XAML, тоже существует четкая и удобная структура. Там есть кнопки, поля для текста, выпадающие списки – всё, что нужно пользователю для удобства взаимодействия с системой. Все эти элементы соединены с данными с помощью data binding – это механизм, который следит за тем, чтобы при изменении данных в базе интерфейс тоже сразу обновлялся. Например, если выбрать картину из списка, то всё, что к ней относится: автор, цена, стиль – сразу появится на экране.

Для лучшего понимания устройства и работы программы ниже представлена диаграмма активности, она наглядно показывает, как пользователь себя может взаимодействовать с системой, в зависимости от его роли. То есть: куда он заходит, что делает и как на его действия реагирует программа. Она наглядно показывает устройство программы, что упрощает понимание программы другим разработчикам и людям, связанным с разработкой или последующим улучшением программы. Благодаря ей проще понять, что требует доработки, оптимизации, изменения или абсолютно не нужно, сразу видна логика программы.

Например, для обычного пользователя есть один путь взаимодействия с программой, он имеет доступ к просмотру товаров и к корзине. Также он может добавлять и удалять товары из корзины. Еще он может подтверждать заказ и тем самым добавлять данные в базу данных.   
Менеджер тоже может просматривать и фильтровать товары по размеру, выставкам и названию. Но у него нет доступа к корзине. Также он может формировать накладные к заказам, добавлять и редактировать товары.

Администратор может делать все тоже, что и менеджер, но, кроме этого, он способен добавлять новых пользователей и раздавать им разные роли.

**Рисунок 1.15 – Диаграмма активностей**

* 1. ****Модель данных****

Хранение данных в приложении построено на базе, которая является его центральной опорой. Именно в ней содержится вся значимая информация: о пользователях, картинах, выставках, заказах и многом другом. Эта база разбита на несколько таблиц – и каждая из них отвечает за свою область. Users, Art, Order, WorkerInfo, Exibition, TypeSize, ShippingType, Role, Position – именно эти таблицы задают структуру системы, её логику и логику взаимодействия компонентов.

Всё начинается с таблицы Users. Она – ключевая и является связующей таблицей для всех остальных. В ней находятся личные данные пользователей: имя, фамилия, контактная информация, а также данные, необходимые для входа в систему. Через связи с другими таблицами, например с WorkerInfo, становится возможным определить, кто пользователь – администратор он, менеджер или просто гость.

Связь с WorkerInfo даёт системе понять, какое место в структуре занимает пользователь. Именно оттуда система узнаёт, какую роль и должность он имеет. Эти данные – не просто служебные поля, они напрямую влияют на поведение и взаимодействие интерфейса по отношению к пользователю. Администратор может многое: добавлять других пользователей, редактировать картины, работать с заказами. Обычный пользователь – в основном просматривает и оформляет.

Вся информация о картинах находится в таблице Art. Это название, автор, цена, изображение. Для связи картин и выставок существует таблица Exibition. Между ними выстроена чёткая связь: одна выставка может содержать много произведений, а каждое произведение может быть связано с конкретной выставкой. Это логично и удобно.

В таблице о выставках содержатся записи об описании и информации о том, какие произведения в неё входят. Взаимосвязь "один ко многим" между Exibition и Art позволяет легко получить нужную информацию – например, всё, что представлено на конкретной выставке.

Не менее важны и заказы. Таблица Order – для этого. Она хранит всё о том кто оформил, что заказал, какие картины выбрал, какой способ доставки предпочёл, оставил ли комментарий к доставке. Это позволяет организовать работу магазина: понимать спрос, отслеживать заказы, готовить доставку. IdUser в этой таблице – это внешний ключ, связующий заказ с Id из таблицы User.

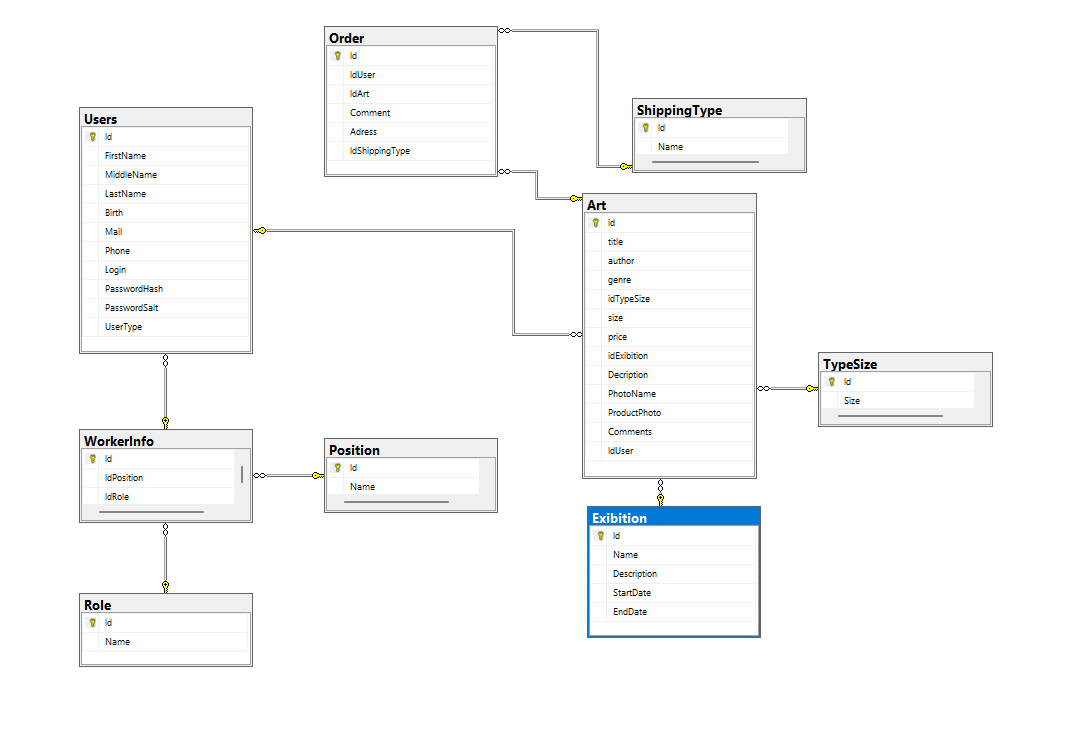
Размер картин подлежит фильтрации. Для уменьшения занимаемого пространства создана таблица TypeSize. В ней содержатся категории: маленькие, средние и огромные. Такая классификация упрощает фильтрацию и поиск. Кто-то ищет большую картину в офис, кто-то – компактную для дома. Всё это учитывается. И не забыта доставка – таблица ShippingType содержит сведения о способах, которыми картины можно получить: самовывоз, доставка курьером и т. д.

Таблицы Role и Position нужны для разграничения прав. Администратор, менеджер, пользователь – каждый со своими возможностями. Position дополняет Role, уточняя, что именно может человек. В итоге система гибко настраивается: права доступа, интерфейс, поведение приложения – всё зависит от этих параметров. Что позволяет в будущем доработать систему разграничения прав без потери данных о пользователях.

Связи между таблицами реализованы через внешние ключи – FK. Это обеспечивает стабильность. Например, если Art ссылается на Exibition, то запись не появится без указания, на какой выставке она будет. Такие ограничения делают структуру базы более четкой, а данные – целостными и полными.

Типы данных в базы данных выбраны для уменьшения занимаемого пространства и удобству работы с каждым типом данных. Для строк – varchar. Для чисел – int. Для цены – decimal. Каждое поле подобрано с учётом смысла: чтобы данные хранились правильно и легко обрабатывались. Это мелочи, которые на деле оказываются важнейшими деталями.

Обработка всех этих данных происходит через ADO.NET. Вставка, обновление, удаление, выборка с фильтрами – всё через этот механизм. Он даёт стабильность, надёжность и хорошую производительность. Особенно при работе с большими объёмами.

Так создаётся удобная, масштабируемая система: из таблиц, ролей, связей, данных и интерфейсов. Количество элементов было максимально сокращено для ускорения работы программы и удобства использования. Все эти элементы составляют основу модели данных приложения. Связи между таблицами, выбранные типы данных и структура базы данных позволяют эффективно организовать работу с информацией и поддерживать целостность данных.  
** Рисунок 1.16 – Диаграмма связей между таблицами**

* 1. ****Описание основных страниц интерфейса****

Интерфейс приложения построен так, чтобы чётко отражать разграничение прав между разными категориями пользователей. В основе лежит ролевая модель: три уровня – Администратор, Менеджер и Пользователь. Каждая роль – это набор полномочий, строго определённый и зафиксированный в системе. Они не пересекаются по критически важным функциям и не допускают двусмысленности.

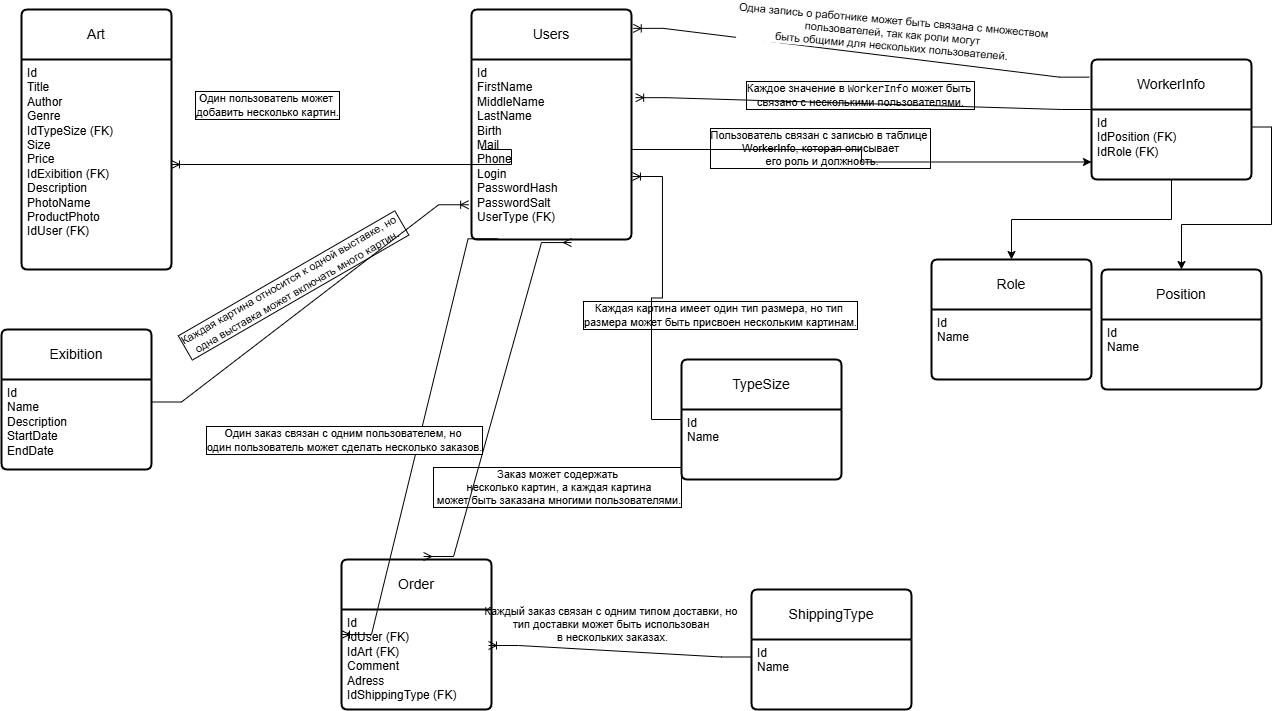
Начнём с самого верха. Администратор – это вершина иерархии. Ему доступны почти все ключевые действия. Он добавляет новые картины, корректирует их свойства – будь то описание, стоимость или визуальное представление. Он же отвечает за ввод новых пользователей в систему, может присваивать им роли и тем самым управлять их правами. Но при всей этой власти у него есть намеренные ограничения. Администратор, например, не имеет возможности редактировать уже созданные учётные записи и не видит список пользователей. Это не упущение – это предусмотрено мерами безопасности. Таким образом минимизируются риски ошибок и злоупотреблений при доступе к данным других пользователей.

Менеджер – роль промежуточная. Он действует в системе почти наравне с администратором, но без прав на управление пользователями. Он может всё, что касается картин: добавлять, менять, обновлять. Также ему доступны функции анализа – формирование отчётов, просмотр заказов, отслеживание динамики. Но пользователей он не видит, не добавляет и не управляет ими. Такой подход разделяет ответственность.

А вот пользователь – это самая простая и ограниченная роль. Ему доступно только то, что связано с просмотром произведений искусства, выбором нужных, добавлением в корзину и завершением покупки. Никаких действий вне своей зоны: он не может изменить картину, не имеет доступа к чужим заказам и, конечно, не видит других пользователей. Это разграничение прав делает систему устойчивой и безопасной.

Авторизация осуществляется через логин и пароль, так как программа для внутреннего использования, то регистрация специально не предусмотрена, пользователей добавляет администратор. После успешной проверки данных система определяет роль входящего и показывает интерфейс в соответствии с его полномочиями. Если это администратор – открываются все доступные инструменты: и управление галереей, и контроль за пользователями. Менеджеру – интерфейс с картинами и заказами. А пользователю – корзина и просмотр.

Такой механизм распределения доступа построен на чёткой логике и защищён от вторжений. Каждая роль видит только то, что ей положено. Это обеспечивает не просто удобство, но и защищённость: никто не сможет выполнить действие, на которое у него нет прав. Гибкость в интерфейсе при этом сохраняется – он адаптируется под роль, делая работу в системе простой и интуитивной.

Именно благодаря этому принципу – авторизации и разделения ролей – система остаётся стабильной, надёжной и управляемой. Это не просто интерфейс, это продуманная модель взаимодействия с учётом рисков и задач.

**Рисунок 1.17 – ER диаграмма**

**Хэширование паролей и безопасность**

Хранить пользовательские пароли просто так, в открытом виде, – недопустимо. Это не просто плохая практика, а реальный риск. Поэтому в системе используется метод с добавлением соли и последующим хэшированием. Этот метод криптографически стойкий, с индивидуальной солью, которая генерируется на каждого нового пользователя отдельно.

Когда пользователь регистрируется или создаётся вручную, его пароль не записывается в базу в прямом виде. Вместо этого из него получают байтовую последовательность, а потом – добавляют соль. Соль, что важно, – это случайный набор байтов, сгенерированный через специальный механизм из пространства System.Security.Cryptography. Важно отметить: каждый раз создается новая соль. Даже если два пользователя выберут одинаковый пароль, итоговый хэш у них всё равно будет разный. Потому что соль – уникальна.

Процесс генерации соли – несложный, но надёжный. Используется генератор криптографически стойких случайных чисел.

public static byte[] GenerateSalt(int size = 16)

{

var rng = RandomNumberGenerator.Create();

var saltBytes = new byte[size];

rng.GetBytes(saltBytes);

return saltBytes;

}

Массив байтов создаётся, заполняется случайными значениями и возвращается. В этот момент в базе ещё ничего не сохраняется – просто генерируется материал для будущего хэша.

К байтам пароля добавляется соль. Вместе они составляют новую последовательность, которую уже можно обрабатывать хэш-функцией. Здесь используется SHA-256. Он надёжен, широко применяется, и он неизменяем. То есть даже если злоумышленник получит хэш, восстановить оригинальный пароль из него он не сможет.

Хэширование выглядит так:

public static byte[] HashPassword(byte[] passwordBytes, byte[] salt)

{

using (var sha256 = SHA256.Create())

{

var combined = passwordBytes.Concat(salt).ToArray();

return sha256.ComputeHash(combined);

}

}

Пароль записывается вместе с солью в одну последовательность, и уже затем в SHA-256. В итоге в базе данных уже получется массив байтов, который и будет храниться, как зашифрованный пароль.

Когда пользователь создаётся, система генерирует соль, получает хэш, и только потом сохраняет эти данные.

byte[] salt = GenerateSalt();

byte[] passwordBytes = Encoding.UTF8.GetBytes(password);

byte[] hashBytes = HashPassword(passwordBytes, salt);

А дальше – создание объекта Users. В него записываются логин, хэш пароля, соль и другие поля, такие как имя, почта, телефон и даты рождения и типа пользователся.

var newUser = new Users

{

Login = login,

PasswordHash = hashBytes,

PasswordSalt = salt,

FirstName = "First",

MiddleName = "Middle",

LastName = "Last",

Mail = "example@mail.com",

Phone = "123456789",

Birth = DateTime.Now,

UserType = newWorkerInfo.Id

};

Перед сохранением есть проверка на повторный логин. Если такого же логина не существует программа успешно добавляет новые данные в базу данных и пользователь успешно создан.

if (context.Users.Any(u => u.Login == LoginTextBox.Text))

{

MessageBox.Show("Пользователь с таким логином уже существует!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

Создаётся объект WorkerInfo, в котором указываются должность и роль. Он сохраняется в базу, и уже после этого создаётся пользователь, которому присваивается только что добавленный UserType.

var workerInfo = new WorkerInfo

{

IdPosition = (int?)PositionComboBox.SelectedValue,

IdRole = (int?)RoleComboBox.SelectedValue

};

context.WorkerInfo.Add(workerInfo);

context.SaveChanges();

Так реализовано уже конечное добавление пользователя, уже с конкретной должностью и ролью, и, конечно, с хэшем и солью:

var user = new Users

{

FirstName = FirstNameTextBox.Text,

MiddleName = MiddleNameTextBox.Text,

LastName = LastNameTextBox.Text,

Mail = MailTextBox.Text,

Phone = PhoneTextBox.Text,

Birth = BirthDatePicker.SelectedDate,

Login = LoginTextBox.Text,

PasswordHash = hashBytes,

PasswordSalt = salt,

UserType = workerInfo.Id

};

context.Users.Add(user);

context.SaveChanges();

Пользователь добавлен, его данные сохранены, а пароль – надёжно зашифрован. Даже если база попадёт в чужие руки, никакой расшифровки не будет.

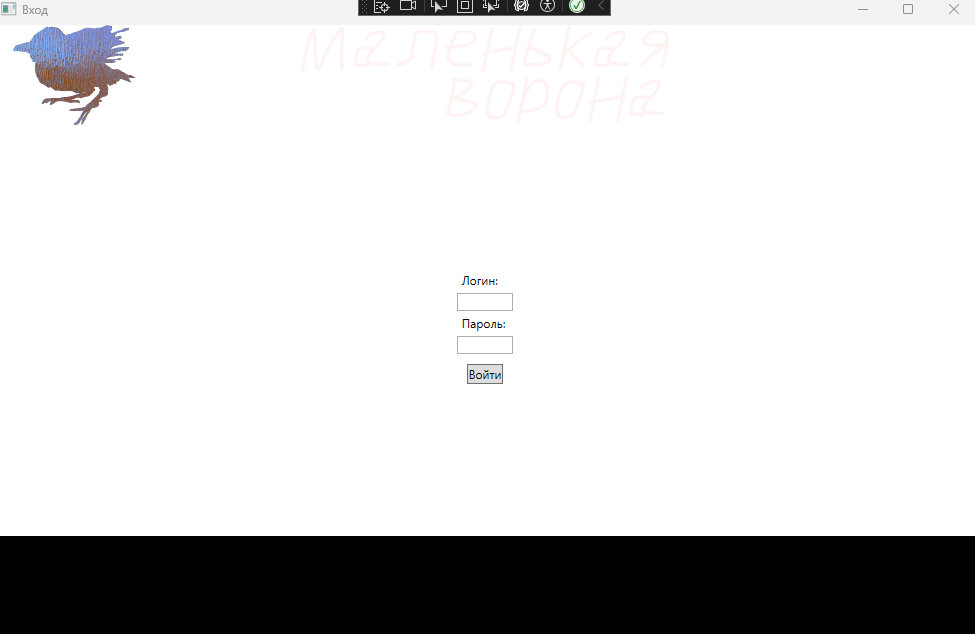
Защита здесь строится на простом, но надёжном принципе – данные, однажды зашифрованные, уже не подлежат дешифровке. Потому что даже если кто-то попробует подобрать пароль по хэшу – ему придётся перебирать миллионы вариантов.

Кроме того, соль делает каждый хэш уникальным. И это исключает возможность сравнения хэшей между пользователями. Один пароль – один результат, только для одного человека.

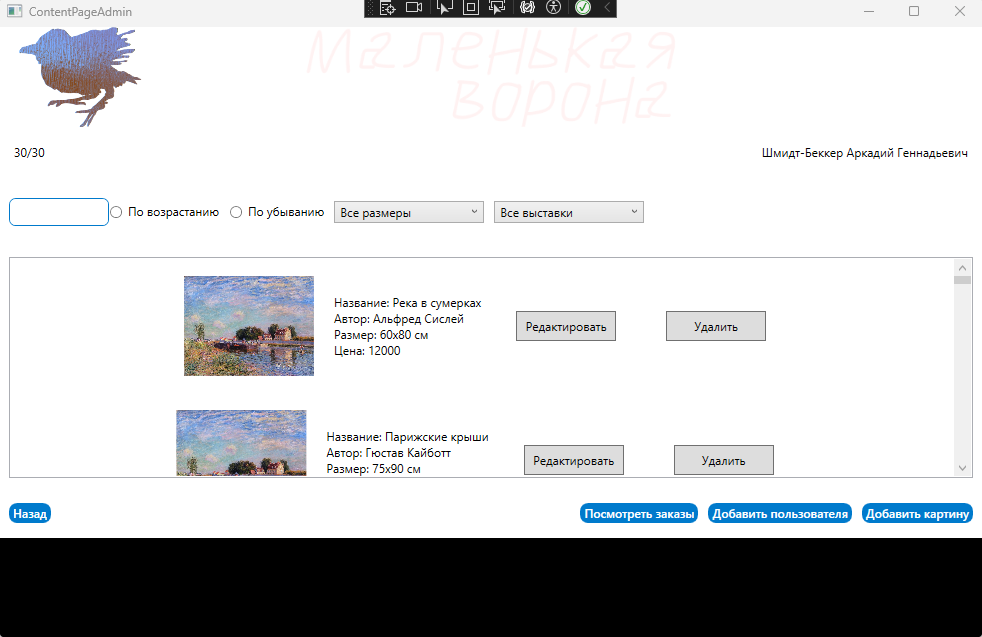
* 1. ****Логическая структура страниц****

Во взаимодействии пользователя с системой важнейшую роль играют страницы, каждая из которых имеет своё назначение. Интерфейс построен так, чтобы доступ к функциям соответствовал ролям. Приложение организовано несложно, но последовательно: всё начинается с логина и заканчивается действиями, зависящими от того, кто вошёл.

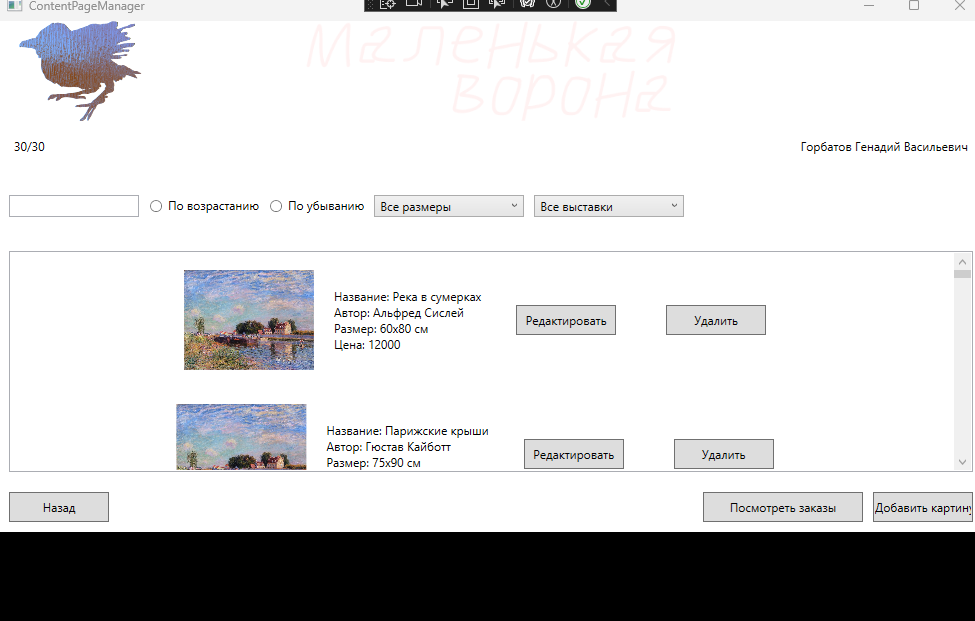
LoginPage – это единственная общая страница для всех пользователей. Там происходит идентификация и дальнейшее перенаправление на страницу, советующую роли пользователя.



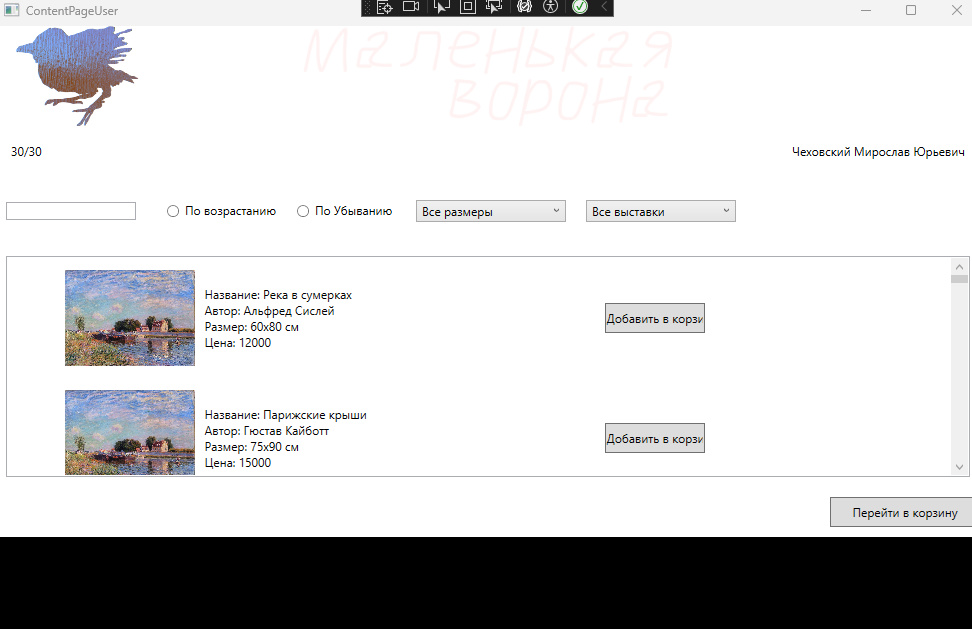
**Рисунок 1.18 – Страница входа**

К примеру, администратор после входа оказывается на ContentPageAdmin. Это его панель управления, откуда он имеет доступ ко всем функциям. Также при вхождении в аккаунт справа отображается полное имя вошедшего пользователя, для полной уверенности пользователя. 

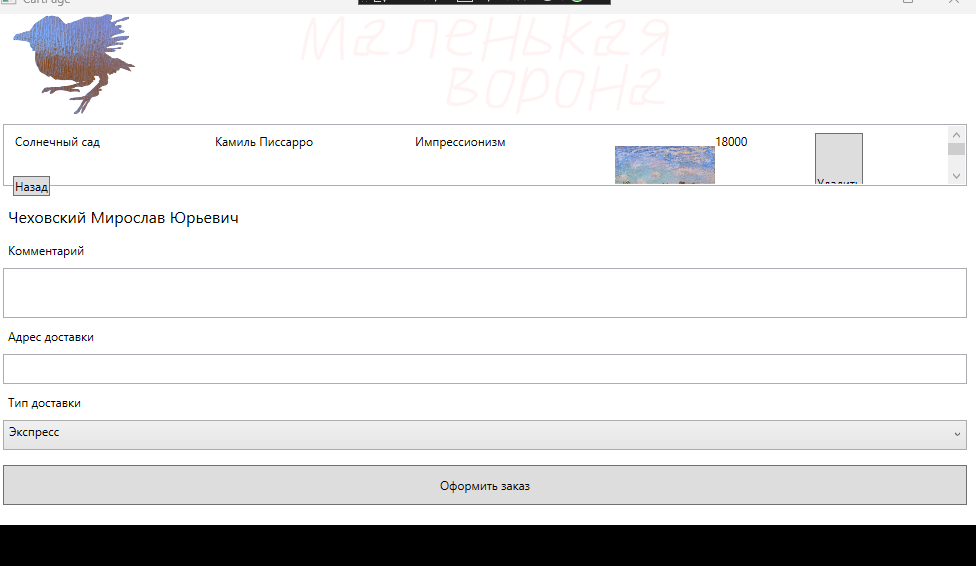
**Рисунок 1.19 – Страница** ContentPageAdmin

Менеджер, в отличие от администратора, получает доступ к ContentPageManager. Он может работать с заказами, может редактировать картины, но не может управлять учетными записями, в отличие от администратора. Это ограничение логичное: нет нужды давать права выше необходимых. Главное, что он может обслуживать клиентов и следить за актуальностью представленных произведений искусства. 

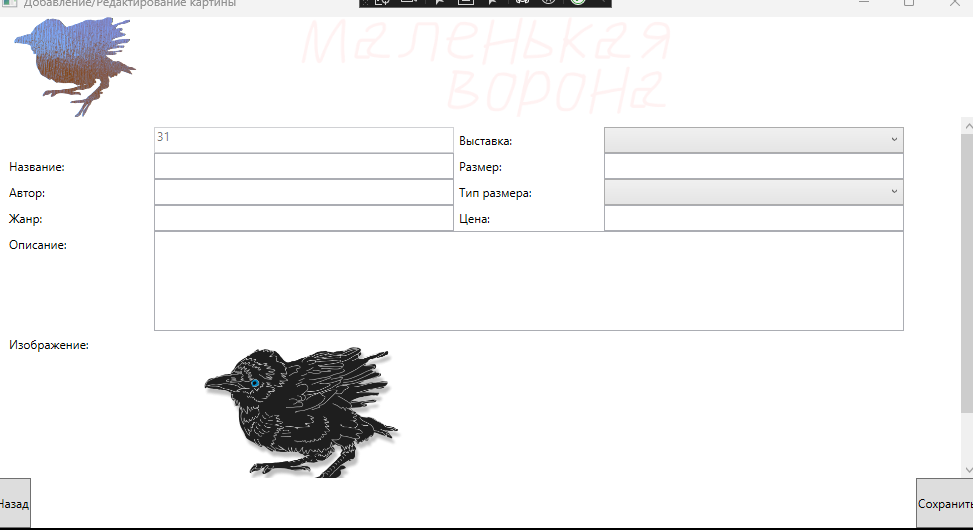
**Рисунок 1.20 – Страница** ContentPageManager

Что касается обычных пользователей – для них своя страница, ContentPageUser. Здесь они могут просматривать доступные картины, добавлять их в корзину, делать заказы. 

**Рисунок 1.21 – Страница** ContentPageUser

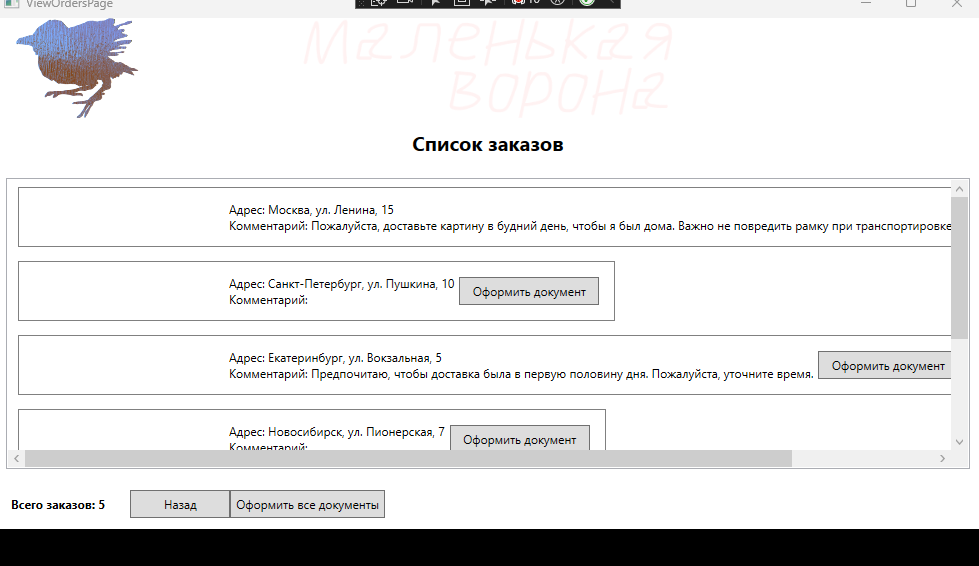
Корзина реализована через CartPage. Это важная часть пользовательского опыта. Там можно не только увидеть отобранные картины, но и удалить и завершить оформление заказа. 

**Рисунок 1.22 – Страница** CartPage

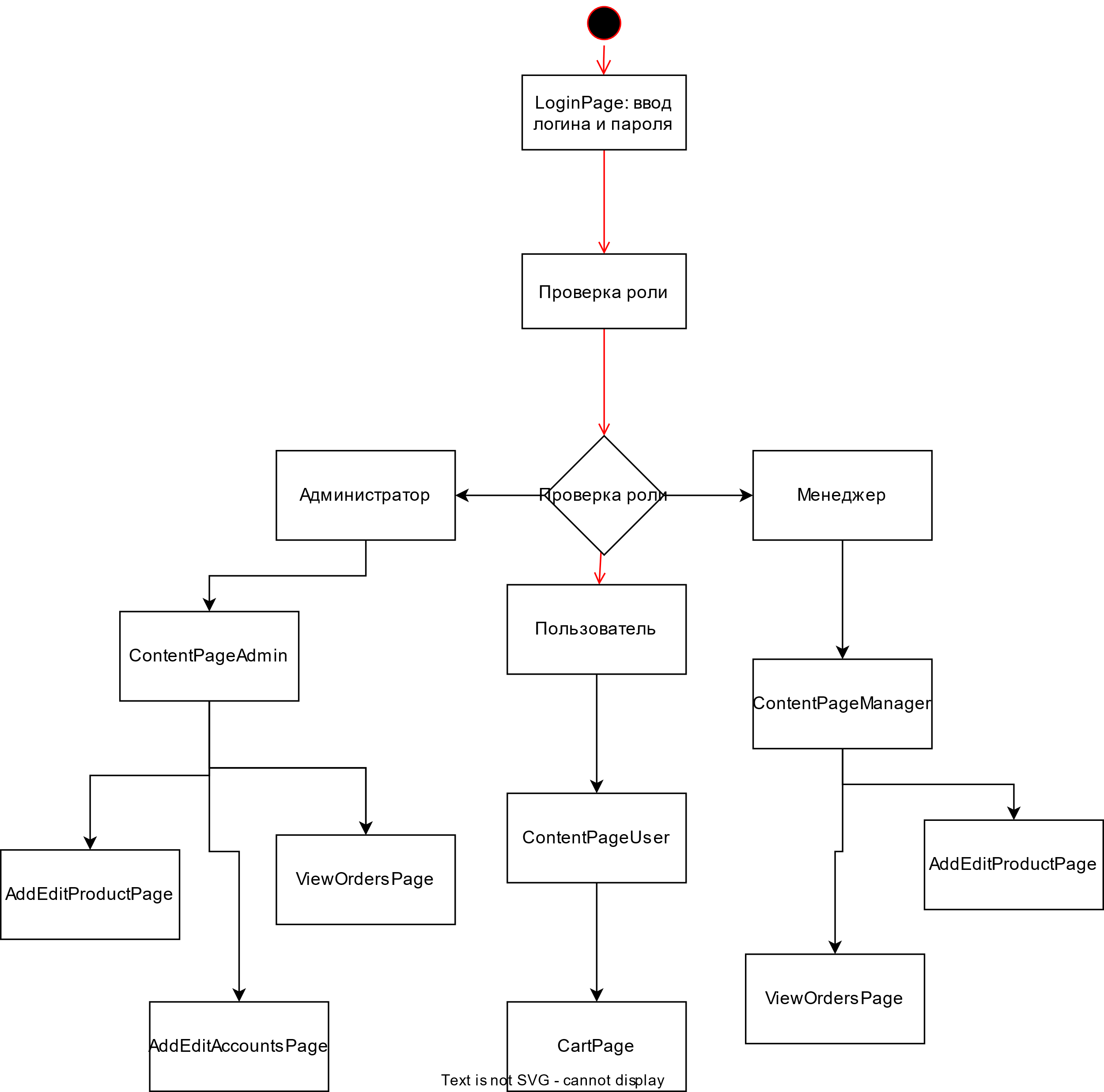
Работа с произведениями происходит через AddEditProductPage. Эта страница – инструмент менеджеров и администраторов. Здесь реализваны возможности загрузки изображения, изменение описания, указания авторов и направления.

**Рисунок 1.23 – Страница** AddEditProductPage

На странице ViewOrdersPage администраторы и менеджеры могут отслеживать кто что заказал, куда доставить. Подробная информация по каждому заказу позволяет держать процесс под контролем. Это особенно важно, когда объём заказов растёт.



**Рисунок 1.24 – Страница** ViewOrdersPage

Архитектура приложения построена на разграничении прав. Это делает систему безопасной, понятной и удобной для изменений. Диаграмма навигации помогает понять как реализованы проверки и перенаправлени системой пользователей на соответствующие страницы.

**Рисунок 1.25 –** Диаграмма навигации

* 1. ****Реализация логики взаимодействия с базой данных****

В основе всей системы лежит Entity Framework, работающий совместно с ADO.NET. Именно такое сочетание в проекте используется для связи между приложением и базой данных. Благодаря этому подходу таблицы базы превращаются в классы, строки таблиц становятся объектами с полями, а управление данными осуществляется через контекст – gallerydatabaseEntities. ADO.NET предоставляет набор классов, через которые происходит взаимодействие с источниками данных, в данном случае через SQL Server. С помощью SqlConnection, а это один из основых компоненотов ADO.NET, можно открывать соединение с базой данных. SQLCommand позволяет легко выполнять SQL-запросы. А SQLAdapter и DataSet позволяют использовать binding для работы с данными, который используется чрезвычайно часто в коде. Эта программа практически построена на упрощенном взаимодействии пользователя и базы данных и Entity Framework позволяет работать с объектами C# без необходимости писать SQL-запросы вручную. Преимущества Entity Framework реализованы методом GetContext(). Он возвращает один и тот же экземпляр контекста, не создавая новые. Потому что множественные контексты – это лишние ресурсы и, к тому же, возможные потери данных. Это исключает дублирование, сокращает расходы памяти, и повышает стабильность.

Вся логика приложения завязана на контексте. Если добавить новую картину – сначала создаётся объект Art. Далее в него вносятся данные: автор, цена, год создания, размер и прочее. После чего объект добавляется в контекст через Add(), а изменения сохраняются в базу вызовом SaveChanges().

Когда записей много вручную сортировать и обрабатывать записи становится сложно, поэтому были добавлены фильтры по выставкам и размерам, сортировки по цене и поиск по названиям и авторам. Все это должно облегчить работу и взаимодействие пользователей с данными. Всё это делается через LINQ – это быстрый и эффективный способ обращения к данным с помощью языка SQL, но в проекте, написанном на C#. После использования фильтров список товаров автоматически каждый раз обновляется без нужды дополнительного вмешательства пользователя.

В итоге получается довольно устойчивая архитектура с возможностями дальнейшей модернизации и изменениями. Разработана защита через шифрование, гибкость с помощью LINQ и контроль через единый контекст. Если потребуется добавить новую таблицу или изменить логику – сделать это будет несложно, благодаря удобной структуре приложения.

* 1. ****Генерация документов Word****

Внутри WPF-приложения, разработанного для ускорения и упрощения работы сотрудников художественной галереи, реализована возможность автоматического создания товарных накладных в формате Microsoft Word. Это действительно важная и влияющая на процесс работы функция, так как она позволяет избавить пользователей от ручного оформления документации, ускоряет процесс документооборота и минимизирует количество ошибок, которые связаны с человеческим фактором.

Формирование документов происходит в определённой последовательности, начиная с извлечения необходимых данных из базы данных и заканчивая сохранением итогового документа в формате .docx. Для логического разделения файлов программы и улучшения общей структуры, логика генерации накладной была вынесена в отдельный статический класс WordDocumentGenerator, основная функция которого, отвечающая за процесс, называется CreateInvoice(int orderId).

Сперва метод CreateInvoice обращается к контексту базы данных, который получается через Data.gallerydatabaseEntities.GetContext(). После по переданному идентификатору заказа orderId, из таблицы Order извлекается конкретный заказ:.

var order = context.Order.FirstOrDefault(o => o.Id == orderId);

Если такой заказ не найден, пользователю выводится сообщение об ошибке:

MessageBox.Show("Заказ не найден!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

После этого осуществляется извлечение связанных сущностей – клиент, информация о котором содержится в таблице Users, произведения искусства соответственно в таблице Art, а информации о способе доставки в ShippingType. Для этого удобнее всего использовать первичные и внешние ключи:

var user = context.Users.FirstOrDefault(u => u.Id == order.IdUser);

var art = context.Art.FirstOrDefault(a => a.id == order.IdArt);

var shippingType = context.ShippingType.FirstOrDefault(s => s.Id == order.IdShippingType);

Если какие-нибудь из этих объектов отсутствуют, формирование документа прекращается, поскольку эти данные являются обязательными для корректного заполнения накладной. Далее создаётся объект приложения Word с помощью:

var wordApp = new Word.Application();

wordApp.Visible = false;

Здесь используется пространство имён Microsoft.Office.Interop.Word. Оно позволяет из C# управлять Word через COM-интерфейсы. Создание новой пустой страницы Word:

var document = wordApp.Documents.Add();

А здесь настраиваются отступы на странице через свойства PageSetup и единицы измерения CentimetersToPoints:

document.PageSetup.TopMargin = wordApp.CentimetersToPoints(2);

Шрифт документа задаётся глобально, потому что другие шрифты в накладной не используются:

document.Content.Font.Name = "Times New Roman";

document.Content.Font.Size = 12;

Первый абзац формируется вручную через Word.Paragraph:

Word.Paragraph para = document.Content.Paragraphs.Add();

string formattedDate = DateTime.Now.ToString("d MMMM yyyy г.", new CultureInfo("ru-RU"));

para.Range.Text = $"Товарная накладная № {order.Id} от {formattedDate}";

Абзац выравнивается по центру, устанавливается жирный шрифт и увеличенный размер

para.Range.Font.Size = 16;

para.Range.Font.Bold = 1;

para.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter; После добавляется информация о поставщике:

para = document.Content.Paragraphs.Add();

para.Range.Text = "Поставщик:\nООО «Глобал Текст», ИНН 7701234567...";

Для добавления данных о покупателе сначала формируется ФИО клиента через соединение вызванных полей из базы данных: LastName, FirstName и MiddleName из таблицы Users:

string clientName = $"{user.LastName} {user.FirstName} {user.MiddleName}";

Эта информация выводится с форматированием в отдельный абзац:

para = document.Content.Paragraphs.Add();

para.Range.Text = $"Покупатель:\n{clientName}, Адрес: {order.Adress}\nТелефон: {user.Phone}";

Одна из основных частей накладной – таблица с описанием проданного произведения. Она создаётся на основе метода Tables.Add, где указывается диапазон вставки и количество строк/столбцов:

Word.Table table = document.Tables.Add(para.Range, 2, 5);

Таблица содержит заголовки и данные о товаре. Каждая ячейка заполняется напрямую через table.Cell(row, column).Range.Text:

table.Cell(1, 2).Range.Text = "Наименование произведения";

table.Cell(2, 2).Range.Text = art.title;

Цены форматируются с двумя знаками после запятой:

table.Cell(2, 4).Range.Text = art.price.ToString("N2");

Далее добавляются дополнительные поля: комментарий к заказу, тип доставки и адрес. Всё это вставляется в текст документа с соответствующими отступами и оформлением

para.Range.Text = $"Комментарий к заказу:\n{order.Comment}";

para.Range.Text = $"Тип доставки:\n{shippingType.Name}";

para.Range.Text = $"Адрес доставки:\n{order.Adress}";

В конце документа добавляется блок для подписей сторон

para.Range.Text = "\n\nПодписи сторон:\n\n" +

"От поставщика: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Иванов И.И./\n\n" +

$"От покупателя: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /{user.LastName} {user.FirstName}/";

Итоговый путь для сохранения файла формируется автоматически. Сначала создаётся папка Накладные на рабочем столе пользователя string directoryPath: = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Desktop), "Накладные");

Файл получает уникальное имя, включающее имя клиента и текущую метку времени, чтобы избежать появления дубликатов и возможной потери информации при замене одним файлом другой:

string filePath = Path.Combine(directoryPath, $"Накладная\_{clientName}\_{DateTime.Now:yyyyMMddHHmmss}.docx");

document.SaveAs2(filePath);

После сохранения документ закрывается, а приложение Word завершается:

document.Close();

wordApp.Quit();

Пользователь получает уведомление об успешной генерации:

MessageBox.Show($"Накладная успешно создана:\n{filePath}", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

Разработанный класс WordDocumentGenerator.cs для автоматической генерации накладных на основе библиотеки Microsoft.Office.Interop.Word обеспечил уменьшение затрат и нагрузки на документооборот, повысил производительность сотрудников галереи и минимизировал количество ошибок при оформлении документации, связанных с человеческим фактором. Этот класс наглядно показывает, как с помощью программного управления программы Microsoft Word можно добиться высокого уровня автоматизации процессов, которые раньше выполнялись вручную.

Вся программная логика формирования документа прописана в статическом методе CreateInvoice(int orderId) класса WordDocumentGenerator. Этот метод выступает связывает базу данных, которая содержит всю необходимую информацию по заказу, и приложение Microsoft Word, которое используется как движок для визуального представления этих данных в виде официально отформатированного документа. Программа взаимодействует с таблицами Order, Users, Art, ShippingType через ORM-средство[[9]](#footnote-9) Entity Framework, чтобы извлечь всю нужную информацию. Благодаря этому достигается модульность и изоляция ответственности: один класс занимается бизнес-логикой, а другой – формированием интерфейса документа.

Создание шаблона Word-документа позволяет гибко управлять содержимым и структурой документа накладной без необходимости поддерживать шаблоны и это упрощает дальнейшее сопровождение кода. Вместо того чтобы загружать уже готовый файл с закладками, метод CreateInvoice динамически создаёт новый документ с нуля. Этот документ создаётся программно: устанавливаются поля страницы, задаётся стиль шрифта, выравнивание абзацев и вставляются параграфы с созданным на основе данных из базы текстом, например, номер заказа, дата, контактные данные клиента и поставщика.

Одной из самых важных частей документа является таблица с данными о произведении искусства. Таблица создаётся с помошью Word.Tables.Add, а ячейки заполняются напрямую, через обращения table.Cell(row, column).Range.Text. Такой подход позволяет точно управлять расположением данных и добиться визуального порядка. Также к форматированию относится – выравниванию текста по центру, включению рамок и регулировке отступов, что позволяет применить к накладной более официальное форматирование.

После основного контента добавляются такие важные элементы, как комментарий клиента, тип и адрес доставки, а также блок с подписями обеих сторон.

Последний этап – это сохранение уже сформированного файла на рабочий стол пользователя в также созданную программно папку «Накладные». Имя файла формируется по уникальному шаблону с использованием ФИО клиента и текущей метки времени (DateTime.Now:yyyyMMddHHmmss), это позволяет исключать возможные конфликты имён и перезаписи файлов. Сохранение производится через метод SaveAs2, а после этого Word-приложение корректно закрывается, и освобождаются все задействованные ранее ресурсы.

В рамках WPF-интерфейса предусмотрены два способа запуска этого механизма: генерация накладной для одного заказа по нажатию кнопки, связанной с конкретным заказом, и генерация сразу всех накладных – массовая обработка заказов. Это реализовано в классе ViewOrdersPage, где данные, отображаемые в DataGrid, позволяют пользователю визуально оценить список заказов, а затем выполнить требуемое действие. Это действительно ускоряет работу администраторов и менеджеров галереи, особенно в условиях высокой нагрузки.

Это решение показывает, как с помощью C#, WPF, Entity Framework и Word Interop можно построить надёжную, масштабируемую и легко поддерживаемую систему документооборота. Этот подход уменьшает влияние человеческого фактора и снижает необходимость ручных правок, он также экономит время, если нужно создать массу накладных, унифицирует оформление всех документов, снижает технические требования к сотрудникам, добавляют возможность для расширения и добавления новых элементов в документ без редактирования шаблона и в конце концов создание документов – это рутина, которую немногие желают.

Всё это делает модуль генерации накладных одним из главных компонентов приложения. Это решение способно адаптироваться под новые требования бизнеса, что действительно важно в условиях постоянно меняющихся задач и объёмов работы.

* 1. ****Архитектура хранения и визуализации изображений в приложении****

В программе цифровой галереи важно создать метод загрузки, хранения и отображения изображений, так как визуальные материалы представляют собой центральный элемент системы и играют ключевую роль как в восприятии интерфейса пользователем, так и в заполнении самой базы данных. Изображения, сопровождающие информацию о произведениях искусства, были реализованы не в виде двоичных данных, сохраняемых непосредственно в структуре базы, а с помощью альтернативного подхода, при котором изображения размещаются во внешней файловой системе – в пределах папки проекта – и привязываются к записям базы данных путём сохранения относительного пути к файлу. Такой метод организации позволяет существенно оптимизировать работу приложения, так как при большом объеме визуального контента база данных не перегружается и сохраняет высокую скорость обработки запросов. Более того, подобная структура позволяет избежать проблем с миграцией базы, а резервное копирование контента упрощается, поскольку визуальная часть может обрабатываться отдельно от данных. Это также даёт гибкость при переносе проекта на сервер или в облачную инфраструктуру.

Реализация отображения изображений начинается уже на уровне XAML-разметки главного окна приложения, где элементы управления <Image> используют относительные пути к изображениям, чтобы задать оформление. В частности, логотипы приложения и надпись бренда отображаются с помощью следующих строк:

<Image Source="resources\secondLogo.png" HorizontalAlignment="left" Margin="20,0,0,0"/>

Эти элементы находятся в первой строке сетки (Grid.Row="0"), что позволяет визуально выделить верхнюю панель окна. Размещение изображений внутри папки Resources даёт возможность организовать структуру проекта централизованно, не разбрасывая файлы по различным директориям, что особенно важно при масштабировании системы.

Однако особенно интересным и важным этапом стало добавление изображений пользователями – точнее, администраторами и менеджерами – при загрузке новых картин в систему. Пользовательский выбор изображения осуществляется через стандартный диалог выбора файла, что реализовано с помощью класса OpenFileDialog из пространства имён Microsoft.Win32. После того как пользователь выбирает файл, происходит копирование изображения во внутреннюю папку проекта, где оно будет храниться и откуда будет подгружаться. Фрагмента кода, реализующего эту логику, выглядит следующим образом:

OpenFileDialog openFileDialog = new OpenFileDialog();

if (openFileDialog.ShowDialog() == true)

{

string fileName = System.IO.Path.GetFileName(openFileDialog.FileName);

string destPath = System.IO.Path.Combine(Environment.CurrentDirectory, "Images", fileName);

File.Copy(openFileDialog.FileName, destPath, true);

art.ImagePath = $"Images/{fileName}";

}

Сперва пользователь выбирает файл, затем извлекается только имя файла без полного пути, после чего формируется конечный путь в папке проекта. Если в указанной папке уже существует файл с таким именем, он будет перезаписан – этот параметр был определен передачей параметра true в метод File.Copy. Как только изображение размещено в папке Images, в свойство ImagePath объекта art сохраняется путь, который и будет записан в базу данных при сохранении произведения искусства. Таким образом, в таблице Art каждое изображение привязано к соответствующей записи через строковое значение, а не через бинарный массив, что делает базу гораздо более лёгкой и управляемой.

Отображение изображения в пользовательском интерфейсе при просмотре информации о картине осуществляется через элемент управления Image, куда подгружается источник изображения по ранее сохранённому пути. Для этого используется следующий код:

img.Source = new BitmapImage(new Uri(art.ImagePath, UriKind.RelativeOrAbsolute));

Таким образом, строка пути, сохранённая в базе, используется для динамической подгрузки картинки. Поддерживаются как относительные пути, так и абсолютные, что делает возможным адаптацию под различные условия запуска приложения, включая работу с внешними файлами или сетевыми ресурсами. Такой подход является достаточно гибким, а при необходимости позволяет модернизировать архитектуру, заменив локальные пути на URL-ссылки к изображениям, размещённым на удалённых серверах или в облачном хранилище.

Кроме основной функции загрузки и отображения, заложен и потенциал к дальнейшему расширению. Например, может быть реализована предварительная валидация изображений – как по формату, так и по размеру. Приложение можно дополнить логикой, которая будет, например, запрещать загрузку изображений определённых форматов (например, .bmp) или ограничивать размер файлов, чтобы не перегружать систему. Более того, возможно автоматическое переименование загружаемых файлов с целью исключения коллизий и сохранения уникальности, а также масштабирование изображений до нужного размера сразу после загрузки, чтобы унифицировать их отображение в интерфейсе.

Таким образом, реализованная система загрузки и хранения изображений представляет собой надёжное и эффективное решение, которое учитывает как ограничения среды, так и потенциальные перспективы развития. Гибкость архитектуры позволяет использовать локальные изображения, не привязываясь жёстко к типу хранения, а сохранение относительных путей в базе даёт возможность легко модифицировать логику хранения в будущем, не затрагивая структуру базы данных. Всё это в комплексе делает реализацию устойчивой, масштабируемой и удобной в сопровождении, что особенно важно при долгосрочной эксплуатации приложения в реальных условиях.

* 1. ****Архитектура хранения и визуализации изображений в приложении****

**В процессе разработки приложения для цифровой галереи особое внимание было уделено вопросам хранения, загрузки и отображения изображений, так как визуальные материалы являются неотъемлемой частью приложения. Визуальные компоненты, такие как изображения картин, играют ключевую роль в восприятии пользовательского интерфейса, создавая не только визуальную привлекательность, но и влияя на функциональность и удобство работы с приложением. Без качественной организации этого процесса система не смогла бы эффективно управлять большим объёмом графического контента, что существенно бы ухудшило её производительность.**

**Для повышения производительности и упрощения структуры хранения данных было принято решение не хранить изображения в базе данных в виде двоичных объектов. Такой подход значительно бы увеличил нагрузку на базу данных, замедляя её работу, особенно при большом объёме данных. Вместо этого был использован более гибкий и эффективный метод, при котором изображения сохраняются не в самой базе данных, а в структуре файловой системы, находящейся вне базы данных. Конкретно, все изображения размещаются в пределах проекта в отдельной папке, а в базе данных сохраняется только путь, указывающий на место нахождения изображения в файловой системе. Это решение позволяет избежать перегрузки базы данных, что существенно повышает её производительность, так как система теперь работает только с текстовыми строками (путями), а не с большими бинарными файлами.**

**При таком подходе приложение имеет несколько ключевых преимуществ. Во-первых, разгрузка базы данных способствует более быстрой обработке запросов и улучшению общей производительности приложения. Во-вторых, при большом объёме изображений и данных система работает гораздо быстрее, а также уменьшается вероятность возникновения ошибок, связанных с перегрузкой памяти базы данных. Третий важный момент заключается в том, что такой способ организации данных упрощает процесс миграции базы данных и резервного копирования. Теперь резервное копирование можно производить отдельно для текстовых данных и для изображений, что упрощает процесс бэкапа и восстановления данных в случае непредвиденных ситуаций.**

**Также, этот подход обладает гибкостью при переносе проекта в облачную инфраструктуру или на сервер. Относительные пути к файлам можно легко адаптировать под различные условия работы приложения, включая возможность работы с удалёнными файлами, например, в облачных хранилищах. Для этого в будущем достаточно будет просто заменить относительные пути на URL-ссылки на изображения, размещённые в интернете. Это решение не требует значительных изменений в структуре базы данных и приложения, что делает систему легко масштабируемой и гибкой при адаптации к новым условиям.**

**Важной частью реализации этой архитектуры является отображение изображений в пользовательском интерфейсе приложения. Для этого в XAML-разметке используется элемент управления <Image>, который принимает на вход путь к изображению. Путь может быть как относительным, так и абсолютным, что позволяет адаптировать приложение под различные варианты хранения изображений, как в локальной файловой системе, так и в облачных хранилищах или других внешних источниках. Например, логотипы приложения и изображения картин отображаются с помощью указания относительного пути к файлу. Это даёт гибкость и позволяет менять местоположение файлов без необходимости редактировать саму структуру приложения.**

**К тому же, важно отметить, что процесс загрузки изображений осуществляется с использованием стандартного диалога выбора файла, который предоставляет пользователю возможность выбрать картину для добавления в систему. Этот процесс был реализован с помощью класса OpenFileDialog из пространства имен Microsoft.Win32. После того как пользователь выбирает нужное изображение, оно копируется в специальную папку проекта, что позволяет централизованно хранить все изображения, не разбрасывая их по различным директориям и упрощая организацию файловой структуры приложения.**

**Далее, чтобы избежать ошибок, связанных с именами файлов, и обеспечить уникальность каждого изображения, была реализована система копирования, которая автоматически сохраняет изображения в папке проекта, присваивая им уникальные имена. В случае, если файл с таким именем уже существует, он будет перезаписан. Это решение также упрощает работу с изображениями и предотвращает появление конфликтов при добавлении новых картин в систему.**

**Но на этом возможности системы не ограничиваются. В приложении предусмотрена возможность дополнительной валидации изображений, которая проверяет формат и размер загружаемых файлов. Это позволяет ограничить загрузку изображений определённых форматов, например, запрещая загрузку файлов с расширением .bmp, или установить ограничения по размеру изображения, чтобы избежать перегрузки системы и сохранить её производительность. Также, в будущем можно будет добавить возможность автоматического переименования загружаемых файлов, что исключит вероятность появления коллизий и обеспечит уникальность имен.**

**Кроме того, система может быть расширена для автоматического масштабирования изображений до нужных размеров сразу после их загрузки. Это обеспечит единообразие отображения изображений в интерфейсе приложения и улучшит общую производительность системы. Например, изображения, загруженные пользователями, можно будет автоматически привести к единому размеру, что упростит работу с ними на разных устройствах и в разных разрешениях экрана.**

**В итоге, архитектура хранения и визуализации изображений в приложении не только эффективно решает текущие задачи, но и предоставляет возможности для дальнейшего расширения функционала. Система гибкая и легко адаптируемая, что делает её подходящей для разных типов проектов и условий эксплуатации. Важно отметить, что все изменения, связанные с хранением изображений, не требуют значительных изменений в структуре базы данных, что упрощает процесс модификации и масштабирования приложения в будущем.**

* 1. ****Логирование и мониторинг: обеспечение стабильности работы****

**Для обеспечения стабильной и предсказуемой работы программного обеспечения необходимо реализовать комплексную систему логирования и мониторинга. Эти механизмы играют ключевую роль в поддержании надёжности, анализе возникающих сбоев и предотвращении возможных отказов системы. В процессе разработки приложения GalleryApp особое внимание было уделено внедрению таких решений, способствующих постоянному контролю за внутренними процессами.**

**Система логирования предназначена для фиксации информации о событиях, происходящих в процессе функционирования приложения. К таким событиям относятся действия пользователя, обращения к базе данных, результаты выполнения функций, а также ошибки и исключения, возникающие при нарушении стандартного хода выполнения. Внедрение логирования позволяет впоследствии анализировать поведение системы и оперативно реагировать на непредвиденные ситуации.**

**В качестве инструмента логирования в проекте была выбрана библиотека NLog – одно из наиболее широко применяемых решений в .NET-среде. Данный инструмент обеспечивает поддержку различных уровней логирования (Trace, Debug, Info, Warn, Error, Fatal), а также предоставляет гибкую настройку формата сообщений, местоположения файлов логов и стратегий архивирования. Такой подход позволяет адаптировать поведение системы логирования к требованиям как этапа разработки, так и эксплуатации.**

**Формат логов включает в себя временную метку события, уровень важности, источник вызова и сообщение. Данная структура обеспечивает прозрачность и простоту анализа при поиске причин сбоев или некорректной работы приложения. Файлы журналов сохраняются в отдельной директории проекта, что гарантирует независимость хранения и доступность для последующей обработки.**

**Отдельное внимание было уделено организации мониторинга – непрерывному отслеживанию состояния системы и её ключевых характеристик. Мониторинг позволяет своевременно фиксировать аномалии, такие как чрезмерное потребление ресурсов, продолжительное выполнение операций или снижение отзывчивости интерфейса. Несмотря на то, что в текущей версии приложения реализован базовый локальный мониторинг, архитектура проекта предусматривает возможность масштабирования данной подсистемы с последующей интеграцией с внешними платформами визуализации и анализа (например, Prometheus или Grafana).**

**Наличие системы логирования также решает задачи аудита и контроля действий пользователей. Например, при выполнении административных операций, таких как добавление произведений искусства, изменение данных или формирование накладных, сведения об этих действиях фиксируются в логах. Это обеспечивает как прозрачность всех изменений, так и защиту от возможных злоупотреблений.**

**Важно отметить, что эффективное логирование предполагает соблюдение баланса между избыточной детализацией и достаточной полнотой информации. Слишком обширные журналы могут затруднить анализ, в то время как недостаточно полные – не позволят выявить источник ошибки. В ходе проектирования были определены ключевые точки логирования, охватывающие критические участки бизнес-логики и взаимодействия с базой данных, что позволило сформировать целостную систему наблюдения за жизненным циклом приложения.**

**В заключение следует подчеркнуть, что логирование и мониторинг являются неотъемлемыми компонентами современного программного обеспечения, особенно в условиях повышенных требований к надёжности и отказоустойчивости. Реализация этих механизмов в приложении GalleryApp обеспечивает как удобство сопровождения, так и возможность масштабируемого развития системы в будущем.**

* 1. ****Пользовательский опыт: анализ и улучшение взаимодействия****

**В современном программном обеспечении пользовательский опыт (User Experience, UX) приобретает особую значимость и выходит на первый план при разработке интерфейсов и логики взаимодействия. Удобство использования, интуитивная понятность элементов управления, логичность навигации, а также визуальная согласованность интерфейса напрямую влияют на восприятие системы конечными пользователями и степень их удовлетворённости. В данной подглаве рассматриваются ключевые аспекты пользовательского опыта в рамках приложения GalleryApp, выявляются существующие проблемы и предлагаются пути их решения.**

**Понимание особенностей целевой аудитории – основа для формирования эффективного UX. В процессе разработки были учтены характеристики предполагаемых пользователей, к числу которых относятся администраторы галереи, менеджеры по работе с выставками, а также обычные посетители, осуществляющие просмотр и оформление заказов. Для каждой из этих категорий разрабатывался собственный интерфейс с набором функций, адаптированных под их задачи. Такой подход позволил минимизировать перегрузку пользовательского интерфейса и исключить ненужные действия, не имеющие отношения к роли конкретного пользователя.**

**Особое внимание в GalleryApp было уделено согласованности элементов интерфейса. Применение единого визуального стиля во всех разделах, использование цветовой схемы, соответствующей тематике галереи, а также аккуратное размещение элементов управления обеспечивают пользователю комфортное визуальное восприятие и предсказуемость взаимодействий. Интерфейс был реализован с использованием технологий WPF, что позволило достичь гибкости в верстке, а также применять современные подходы к построению визуальных компонентов.**

**Одним из важных параметров, влияющих на пользовательский опыт, является скорость отклика интерфейса. Проведённый анализ показал, что при работе с большими объёмами данных (например, списками картин или заказов) может наблюдаться некоторая задержка в обновлении информации. В рамках оптимизации были реализованы механизмы ленивой загрузки, асинхронного получения данных и кэширования, что позволило значительно повысить отзывчивость интерфейса и сократить время ожидания.**

**Также важным элементом UX является ясность и предсказуемость сценариев взаимодействия. Приложение GalleryApp строится по принципу минимального числа действий до достижения цели: оформление заказа, добавление новой картины или редактирование информации осуществляется в несколько логичных шагов без избыточной навигации. Все формы ввода снабжены поясняющими подписями, обязательные поля выделены, а сообщения об ошибках формулируются понятно и информативно, что значительно снижает вероятность пользовательской ошибки.**

**С целью оценки степени удовлетворённости пользователей в рамках тестирования был проведён сбор отзывов среди представителей каждой роли. Полученные данные позволили выделить несколько направлений для дальнейшего улучшения. Среди них – адаптация интерфейса под более высокие разрешения экрана, реализация тёмной темы, а также внедрение поисковых фильтров и сортировок в наиболее нагруженных разделах, таких как списки заказов и произведений искусства.**

**В качестве направления развития пользовательского опыта рассматривается также внедрение элементарных анимаций и микровзаимодействий, улучшающих восприятие переходов и нажатий. Такие детали, хотя и не являются критичными с точки зрения функциональности, существенно повышают субъективное ощущение качества и современности интерфейса.**

**В долгосрочной перспективе возможно привлечение более формализованных методов анализа UX, включая A/B-тестирование различных версий компонентов, сбор и анализ метрик поведения пользователей (например, продолжительности сессий, частоты возвратов к разделам) и построение тепловых карт активности. Эти меры позволят не только выявить узкие места в логике взаимодействия, но и определить наиболее эффективные пути улучшения.**

**В заключение необходимо отметить, что пользовательский опыт является неотъемлемой составляющей конкурентоспособности программного продукта. В условиях растущих требований к удобству и качеству интерфейсов, систематический анализ UX и его постоянное совершенствование представляют собой важную задачу, напрямую влияющую на успех внедрения и распространения приложения. Реализация указанных улучшений в GalleryApp подтверждает ориентацию проекта на конечного пользователя и стремление обеспечить комфортную и продуктивную работу с системой.**

1. **Итоги разработки программного обеспечения**
   1. ****Общая характеристика разработанного программного продукта****

Разработанный программный продукт представляет собой настольное приложение, созданное с использованием технологий Windows Presentation Foundation (WPF) и языка программирования C# в среде разработки Visual Studio. Его основное предназначение заключается в автоматизации работы галереи, а именно – в упрощении процессов учёта художественных произведений, формирования заказов, управления пользовательскими правами и генерации документов. По сути, это комплексное решение, в котором учтены как базовые административные задачи, так и потребности конечных пользователей. Программа реализует ряд функций, важных для повседневной деятельности сотрудников учреждения, обслуживающего художественные экспозиции.

С архитектурной точки зрения приложение построено по принципу модульности. В его основе лежит чёткое разделение логики, визуального интерфейса и доступа к данным. Это позволяет обеспечить удобство поддержки, масштабируемость и адаптацию системы в будущем. Основные логические блоки сгруппированы в отдельные классы, такие как Manager, UserSecurityManager и другие, каждый из которых отвечает за строго определённый набор задач. Например, класс Manager управляет навигацией между страницами, хранит информацию о текущем пользователе, реализует добавление и удаление произведений искусства в виртуальную корзину, а также взаимодействует с сущностями базы данных. Такой подход способствует повышению читаемости и повторному использованию кода, а также снижает риск логических ошибок.

Хранилищем информации служит база данных gallerydatabaseEntities, связанная с приложением посредством технологии Entity Framework. Это даёт возможность работать с данными в объектно-ориентированной манере, при этом сохраняя доступ к полноценному SQL-функционалу. База данных включает ключевые таблицы, такие как Users, Art, Order, ShippingType, WorkerInfo, Exibition, Role, Position и другие. Каждая таблица отражает реальный элемент предметной области, а связи между ними обеспечивают целостность и логическую связанность данных.

Визуальный интерфейс программы разработан с учётом принципов удобства, лаконичности и визуальной логики. Главное окно приложения содержит стандартную сетку с тремя рядами: верхняя часть оформлена логотипом и декоративными элементами, центральная – предназначена для динамической загрузки содержимого (страниц), нижняя служит декоративной границей. Навигация осуществляется через элемент Frame, которому из кода присваиваются соответствующие страницы. Благодаря этому реализована последовательная, понятная и интуитивно доступная структура взаимодействия с приложением.

Форма авторизации является первым экраном, который видит пользователь. Она запрашивает логин и пароль, при этом процесс аутентификации не просто сверяет строки – используется криптографическая проверка с помощью HMACSHA512. При входе данные пароля хэшируются с использованием соли, хранящейся в базе, и только после этого производится сравнение с хранимым хэшем. Это повышает безопасность и исключает возможность простого перебора паролей или их хранения в открытом виде.

Функциональные возможности программы распределены по ролям пользователей: администратор, менеджер и обычный пользователь. Каждая роль открывает доступ к своему набору действий. Например, администратор может добавлять новых сотрудников, управлять картинами, а также формировать заказы. Менеджеру доступны функции добавления новых произведений искусства, редактирования их характеристик, просмотра списка заказов. Пользователь, в свою очередь, может просматривать доступные произведения, добавлять их в корзину и оставлять комментарии к заказу. Такой подход реализован с помощью фильтрации доступа на уровне интерфейса, а также через проверку ролей в коде.

Особое внимание уделено механизму формирования заказов. При выборе пользователем произведения искусства оно добавляется в список CartItems, а в базу данных добавляется запись в таблицу Order. При этом используются стандартные значения, такие как адрес по умолчанию и тип доставки, которые могут быть изменены в последующем. При необходимости пользователь может удалить элемент из корзины, и система корректно обработает это как на уровне интерфейса, так и в базе данных.

Что касается добавления изображений, каждое произведение может сопровождаться графическим файлом, путь к которому хранится в базе. Загрузка изображений реализована с помощью элемента Image в XAML и классов BitmapImage и Uri в C#, что позволяет динамически подгружать визуальное представление каждой картины. Это даёт пользователю наглядное понимание содержимого и облегчает выбор при заказе. Хранение изображений осуществляется в папке ресурсов проекта или на локальном пути, указанном при добавлении.

Разработка велась с учётом принципов расширяемости. В коде можно найти заделы под будущие улучшения: например, в классе Manager предусмотрена возможность расширения логики авторизации, а также механизмы формирования отчётов на основе шаблонов Word, которые в перспективе могут быть связаны с заказами или экспозициями.

Итоговый программный продукт представляет собой стабильную, функциональную и логически выстроенную систему. Он не только справляется с поставленными задачами, но и обладает потенциалом для масштабирования. Благодаря использованию современных средств разработки, модульной архитектуре и вниманию к деталям, приложение способно удовлетворить потребности как небольших галерей, так и более масштабных учреждений с расширенной структурой.

* 1. Результаты тестирования и проверка работоспособности

Тестирование разработанного программного продукта проводилось последовательно, поэтапно и, что важно, вручную – с привлечением различных сценариев использования, характерных для реальных условий работы галереи. Проверка осуществлялась сразу после завершения реализации ключевых функций, а также после добавления новых компонентов, таких как страницы редактирования аккаунтов, формы оформления заказов и механизм генерации документов. Такой подход позволил своевременно выявлять отклонения в логике работы, непредусмотренные ошибки и некорректное поведение интерфейса, ещё до того, как они могли бы повлиять на пользовательский опыт.

Во время тестирования особое внимание уделялось проверке корректности авторизации. Как известно, безопасность – это не просто опция, а основа доверия к системе. Поэтому хэширование паролей с использованием соли (в частности, HMACSHA512) тестировалось как на этапе регистрации, так и при входе в систему. Были проведены ручные попытки входа с неправильными логинами и паролями, тестировались сценарии с пустыми полями, а также проверялась работа при использовании одинаковых паролей с разной солью. Всё это позволило убедиться, что авторизация работает надёжно, не давая лишнего пространства для манипуляций или уязвимостей.

Одним из ключевых направлений проверки была навигация между страницами приложения. Система, как известно, состоит из множества отдельных окон – от страницы входа до страниц добавления и редактирования данных. Навигационная логика, реализованная через Frame, где страницы присваиваются в Manager.MainFrame.Navigate(), протестирована была в разных условиях – при авторизации под разными ролями, при возвратах назад, при перезапусках приложения. Не было выявлено сбоев в навигации, структура переходов осталась устойчивой даже при быстром переключении между действиями, что важно для комфортной работы пользователей.

Не обошлось, конечно, без мелких недочётов. Например, на раннем этапе была обнаружена ошибка, связанная с отображением изображений при просмотре карточек произведений искусства. При отсутствии корректного пути в базе программа пыталась загрузить несуществующий файл, что вызывало исключение. Проблема была устранена путём добавления проверки на существование файла и использования стандартного изображения-заглушки при его отсутствии. Этот простой, но важный шаг повысил устойчивость системы и исключил критические сбои при отображении данных.

Проверке подверглись и функции, связанные с базой данных. Оформление заказов, сохранение данных в таблице Order, удаление произведений искусства, изменение информации о пользователях – все эти действия тестировались в разных комбинациях, чтобы исключить вероятность потери данных или логических конфликтов. Например, при удалении записи о произведении из базы проверялась реакция интерфейса и то, как изменяется состояние связанной корзины пользователя. В результате была добавлена логика, предотвращающая удаление, если произведение уже включено в заказ. Такое поведение защищает целостность информации и исключает «висячие» ссылки.

Отдельно можно упомянуть тестирование ролей. Каждая роль – будь то администратор, менеджер или обычный пользователь – имеет строго ограниченный круг допустимых действий. Вручную проверялись все сценарии: от попытки обычного пользователя попасть на страницу добавления произведений до входа администратора и редактирования всех данных. Ошибки прав доступа не были зафиксированы, что подтверждает корректность реализации разграничения доступа на основе роли.

Говоря о стабильности, стоит признать: система в целом работает уверенно. Во время тестирования не наблюдалось ни зависаний, ни критических сбоев. Даже при множественных открытых страницах, при параллельной работе с базой и при загрузке изображений приложение сохраняло отзывчивость. Это означает, что реализованная архитектура справляется с нагрузкой, характерной для предполагаемого сценария эксплуатации.

Немаловажным аспектом также стало тестирование на различных устройствах. Программа запускалась как на стандартных настольных компьютерах, так и на ноутбуках с разным разрешением экрана. Интерфейс вёл себя предсказуемо, элементы не выходили за границы окна, и пользовательский опыт оставался стабильным. Это говорит о корректной работе XAML-разметки и использовании элементов, которые адаптируются под различные параметры экрана.

Таким образом, по результатам тестирования можно уверенно заявить: программный продукт работает стабильно, функционально полноценно и соответствует всем заявленным требованиям. Обнаруженные в процессе проверки недочёты оперативно устранялись, в большинстве случаев – с доработкой логики или улучшением интерфейса. Тестирование показало, что система готова к реальному использованию в условиях художественной галереи и может быть внедрена без существенных ограничений или доработок.

* 1. ****Возможности масштабирования и дальнейшего развития****

Возможности дальнейшего развития и масштабирования разработанного программного продукта открывают перед нами огромный потенциал для его совершенствования, улучшения функционала и удовлетворения растущих требований пользователей. Важным направлением является возможность масштабирования, которая может быть реализована через внедрение различных инновационных подходов. На текущий момент система работает стабильно и эффективно, однако внедрение новых технологий и архитектурных решений сделает её ещё более гибкой и масштабируемой.

Одним из очевидных шагов является переход на клиент-серверную архитектуру. Это решение обеспечит разделение обработки данных и интерфейса, что значительно улучшит производительность системы, а также повысит её отказоустойчивость. В текущей реализации все процессы выполняются непосредственно на клиенте, что ограничивает возможности масштабирования, особенно в условиях увеличения числа пользователей. С переносом базы данных и обработки запросов на сервер нагрузка на клиентские устройства будет значительно снижена. Такое разделение обеспечит быстрый доступ к данным и их эффективную обработку, что крайне важно, если приложение будет работать с большим объёмом данных, например, изображениями картин, документами и другими мультимедийными файлами.

Кроме того, использование серверной архитектуры повысит отказоустойчивость всей системы, что немаловажно при увеличении потока пользователей. Центральное хранилище данных на сервере обеспечит не только большую безопасность, но и удобство администрирования. Сервер также будет использовать логи для мониторинга и анализа работы системы, что позволит оперативно реагировать на возникающие проблемы и повышать эффективность работы всей платформы.

Облачные технологии – ещё одно важное направление для масштабирования. На данный момент все файлы (включая изображения картин) хранятся локально, что ограничивает объём доступного пространства на сервере. Однако по мере роста коллекции произведений искусства и данных, таких как фотографии и видео, потребуется значительное расширение возможностей хранилища. Использование облачных сервисов, например, Amazon S3 или Google Cloud Storage, решит эту проблему, обеспечив гибкость и масштабируемость. Вдобавок, облачные решения обеспечат надёжность хранения данных, так как информация будет защищена с помощью технологий резервного копирования и распределения между несколькими географически удалёнными центрами обработки данных.

Внедрение облачных технологий позволит существенно расширить функционал системы, добавив гибкости и устранение ограничений, которые могут возникать при локальном хранении данных. С помощью таких сервисов можно легко масштабировать объём хранимых данных, не увеличивая инфраструктурные затраты. Эти решения также позволят улучшить доступность данных и безопасность, а значит, повысится стабильность и надежность всей системы.

Автоматизация создания отчётов – это ещё одна ключевая возможность для улучшения функциональности программы. В текущем виде отчёты о заказах, произведениях искусства и пользователях генерируются вручную, что требует значительных временных затрат. Внедрение автоматической генерации отчётов значительно упростит этот процесс. Генерация отчётов в таких форматах, как Excel и PDF, а также их отправка на электронную почту значительно сэкономит время и повысит эффективность работы сотрудников галереи. Помимо этого, отчёты могут включать не только статистику, но и аналитические данные, что окажет помощь в принятии более обоснованных решений.

Возможность добавления различных фильтров в отчёты, включая отчёты по продажам, анализ покупок, тренды интересов, а также отчёты по определённым периодам, будет способствовать более точному анализу данных. Это упростит задачу менеджеров и администраторов, повысив точность их решений и оперативность работы. Вдобавок, отчёты смогут содержать более развернутую информацию о популярных произведениях искусства, таких как самые востребованные картины, информация о трендах в предпочтениях покупателей и другие показатели, которые окажутся полезными для дальнейшего развития галереи.

Введение системы авторизации через внешние сервисы – это ещё один важный шаг в развитии программы. В настоящее время пользователи проходят авторизацию с помощью внутренних логинов и паролей. Однако внедрение таких возможностей, как вход через Google, Facebook или другие внешние сервисы, упростит процесс регистрации и входа. Это не только избавит пользователей от необходимости запоминать пароли, но и повысит уровень безопасности. Внешние сервисы используют продвинутые методы защиты, такие как двухфакторная аутентификация, что дополнительно повысит защиту данных.

Кроме того, внедрение системы персонализированных рекомендаций значительно улучшит пользовательский опыт. Такие рекомендации могут быть основаны на анализе поведения пользователя, его предыдущих заказов и предпочтений. Система может использовать алгоритмы машинного обучения для анализа этих данных и предоставления рекомендаций на основе интересов пользователя. Это улучшит взаимодействие с системой и поможет увеличить продажи, так как пользователи будут видеть те произведения искусства, которые могут их заинтересовать.

Интеграция с финансовыми системами и системой учёта продаж также имеет большое значение. Это позволит автоматизировать процесс отслеживания финансовых потоков, учитывать все поступления и расходы, а также производить расчёты прибыли в реальном времени. Это также обеспечит менеджеров оперативной информацией о ключевых финансовых показателях. Система будет автоматически отслеживать все финансовые операции и предоставлять отчёты о прибыли и расходах, что ускорит принятие управленческих решений.

Масштабирование системы и внедрение новых технологий – это не просто улучшение текущей работы. Это возможность подготовить платформу к будущим изменениям и требованиям. Все эти улучшения создадут гибкую и мощную систему, способную справляться с растущими нагрузками и быстро адаптироваться к изменениям на рынке. Технологии машинного обучения, облачные решения, автоматизация процессов и улучшенная безопасность – все эти меры помогут системе работать более эффективно и устойчиво в долгосрочной перспективе.

**Список использованных источников**

1. **Сделки на арт-рынке: необходимые шаги при покупке произведения искусства [Электронный ресурс] – URL: https://www.gorodissky.ru/publications/articles/sdelki-na-art-rynke-neobkhodimye-shagi-pri-pokupke-proizvedeniya-iskusstva/ Режим доступа: свободный (дата обращения: 06.02.2025);**
2. **Исключительное право на произведение [Электронный ресурс] – URL: https://amigdala.pro/blog/intellectual-property/isklyuchitelnoe-pravo-na-proizvedenie/ Режим доступа: свободный (дата обращения: 04.02.2025).**
3. **Разница между реляционными и нереляционными базами данных. [Электронный ресурс] – URL: https://habr.com/ru/companies/sberbank/articles/672022/ Режим доступа: свободный (дата обращения: 06.02.2025).**
4. **Entity Framework | Microsoft Learn [Электронный ресурс] – URL: https://learn.microsoft.com/ru-ru/aspnet/entity-framework Режим доступа: свободный (дата обращения: 06.03.2025)**
5. **ГОСТ 19.201-78. Единая система программной документации. Требования к программам. [Электронный ресурс] – URL: https://docs.cntd.ru/document/1200000791 Режим доступа: свободный (дата обращения: 08.03.2025).**
6. **Entity Framework Core in Action / Джон Смит. – Manning Publications, 2018.**
7. **Соль (криптография) – Википедия [Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/Соль\_(криптография) Режим доступа: свободный (дата обращения: 07.03.2025).**
8. **C# 9 и .NET 5. Современная кроссплатформенная разработка / Марк Дж. Прайс. – Питер, 2021.**
9. **Обзор - ADO.NET | Microsoft Learn [Электронный ресурс] – URL: https://learn.microsoft.com/ru-ru/dotnet/framework/data/adonet/ado-net-overview Режим доступа: свободный (дата обращения: 21.04.2025).**
10. **ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы. [Электронный ресурс] – URL: https://docs.cntd.ru/document/1200000792 Режим доступа: свободный (дата обращения: 18.03.2025).**
11. **Pro WPF 4.5 in VB: Windows Presentation Foundation in .NET 4.5 / Мэтью Макдональд. – Apress, 2013.**
12. **Что такое хеш с солью? И чем он отличается от хеша без соли? [Электронный ресурс] – URL: https://otvet.mail.ru/question/229919218 Режим доступа: свободный (дата обращения: 13.04.2025).**
13. **Введение в ADO.NET [Электронный ресурс] – URL: https://metanit.com/sharp/adonet/1.1.php Режим доступа: свободный (дата обращения: 08.04.2025).**
14. **Cryptography and Network Security: Principles and Practice / Уильям Столлингс. – Pearson, 2017.**
15. **Что такое Windows Presentation Foundation - WPF .NET [Электронный ресурс] – URL: https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/overview/?view=netdesktop-9.0 Режим доступа: свободный (дата обращения: 16.04.2025).**
16. **Безопасное хранение соли [Электронный ресурс] – URL: https://ru.stackoverflow.com/questions/1186646/ Режим доступа: свободный (дата обращения: 16.04.2025).**
17. **Учебник по языку C# 13 и платформе .NET 9 [Электронный ресурс] – URL: https://metanit.com/sharp/tutorial/ Режим доступа: свободный (дата обращения: 16.04.2025).**
18. **ГОСТ 19.101-77. Единая система программной документации. Виды программ и программных документов. [Электронный ресурс] – URL: https://docs.cntd.ru/document/1200000790 Режим доступа: свободный (дата обращения: 20.04.2025).**
19. **Entity Framework Core 2 для ASP.NET Core MVC для профессионалов / Валерий Смирнов. – ДМК Пресс, 2018.**
20. **Пароли, шифрование, хеширование и СОЛЬ [Электронный ресурс] – URL: https://www.youtube.com/watch?v=VbB99SOR8OE Режим доступа: свободный (дата обращения: 20.04.2025).**
21. **Работа с базами данных в C# и .NET [Электронный ресурс] – URL: https://metanit.com/sharp/ado.php Режим доступа: свободный (дата обращения: 20.04.2025).**
22. **Соль (криптография) [Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/Соль\_(криптография) Режим доступа: свободный (дата обращения: 22.04.2025).**
23. **C# и WPF | Привязка [Электронный ресурс] – URL: https://metanit.com/sharp/wpf/11.php Режим доступа: свободный (дата обращения: 22.04.2025).**
24. **Что такое Entity Framework и как его использовать [Электронный ресурс] – URL: https://skillbox.ru/media/code/entity\_framework/ Режим доступа: свободный (дата обращения: 23.04.2025).**
25. **Как правильно хранить пароли своих пользователей [Электронный ресурс] – URL: https://www.kaspersky.ru/blog/how-to-store-passwords/36161/ Режим доступа: свободный (дата обращения: 23.04.2025).**
26. **Руководство по C# – управляемый язык .NET [Электронный ресурс] – URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/ Режим доступа: свободный (дата обращения: 23.04.2025).**
27. **WPF и C# | Полное руководство [Электронный ресурс] – URL: https://metanit.com/sharp/wpf/ Режим доступа: свободный (дата обращения: 23.04.2025).**

1. Главная страница сайта галереи AnnaNova [Электронный ресурс] – URL: <https://www.annanova-gallery.ru/> Режим доступа: свободный (дата обращения: 06.02.2025). [↑](#footnote-ref-1)
2. Страница сайта галереи AnnaNova с описанием [Электронный ресурс] – URL: <https://www.annanova-gallery.ru/about/> Режим доступа: свободный (дата обращения: 06.02.2025). [↑](#footnote-ref-2)
3. Сайт торговой площадки Artsy [Электронный ресурс] – URL: <https://www.artsy.net/> Режим доступа: свободный (дата обращения: 06.02.2025). [↑](#footnote-ref-3)
4. Официальный сайт ИС КАМИС [Электронный ресурс] – URL: <https://www.kamis.ru/> Режим доступа: свободный (дата обращения: 06.02.2025). [↑](#footnote-ref-4)
5. Страница информации об ИС КАМИС [Электронный ресурс] – URL: <http://artinfo.ru/eva/eva2000m/2000/rus/altsoft/product.htm> Режим доступа: свободный (дата обращения: 06.02.2025). [↑](#footnote-ref-5)
6. JVM (Java Virtual Machine) – это платформенно-независимый язык программирования. [Электронный ресурс] – URL: <https://www.ibm.com/docs/en/b2b-integrator/6.1.1?topic=management-java-virtual-machine> Режим доступа: свободный (дата обращения: 06.02.2025). [↑](#footnote-ref-6)
7. Нативное приложения – приложение, создаваемое под конкретную платформу. [Электронный ресурс] – URL: <https://appmaster.io/ru/blog/otlichie-nativnyh-mobilnyh-prilozhenij-ot-vseh-ostalnyh> Режим доступа: свободный (дата обращения: 06.02.2025). [↑](#footnote-ref-7)
8. Рефакторинг – это переработка исходного кода программы, чтобы он стал более простым и понятным. [Электронный ресурс] – URL: https://skillbox.ru/media/code/chto\_takoe\_refaktoring\_koda\_i\_zachem\_on\_nuzhen/ Режим доступа: свободный (дата обращения: 21.02.2025). [↑](#footnote-ref-8)
9. Официальный сайт flexberry [Электронный ресурс] – URL https://flexberry.github.io/ru/gbt\_orm.html Режим доступа: свободный (дата обращения: 04.01.2025). [↑](#footnote-ref-9)