**Государственное бюджетное профессиональное образовательное учреждение**

**«Нижегородский Губернский колледж»**

Методическая комиссия «Информатика и вычислительная техника»

Допущен к защите:

преподаватели

\_\_\_\_\_\_\_\_\_\_\_\_ Е.П. Голубева

\_\_\_\_\_\_\_\_\_\_\_\_ Л.В. Мухина

«18»\_\_апреля\_\_2025 г.

**ОТЧЕТ** **ПО**

**ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

ПМ.01 РАЗРАБОТКА МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ КОМПЬЮТЕРНЫХ СИСТЕМ

Руководитель: \_\_\_\_\_\_\_\_\_\_ Глодин М. Н. 18.04.2025 г.

Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Любивая А. А. 18.04.2025 г.

Специальность, группа: 09.02.07, 43П

Нижний Новгород

2025 г.

[Введение 3](#_Toc195794852)

[1. Анализ предприятия для разработка прикладного решения 5](#_Toc195794853)

[1.1. Описание деятельности предприятия 5](#_Toc195794854)

[1.2. Аппаратное и программное обеспечение предприятия 12](#_Toc195794855)

[1.3. Организация обмена данных между сотрудниками 15](#_Toc195794856)

[1.4. Использование антивирусного программного обеспечения и других средств защиты 17](#_Toc195794857)

[2. Проектирование информационной системы 19](#_Toc195794858)

[2.1. Составление плана этапов разработки прикладного решения для предприятия 19](#_Toc195794859)

[2.2. Создание макетов страниц разрабатываемого прикладного решения 22](#_Toc195794860)

[3. Разработка информационной системы для предприятия 28](#_Toc195794861)

[3.1. Разработка базы данных 28](#_Toc195794862)

[3.2. Разработка моделей таблиц 31](#_Toc195794863)

[3.3. Разработка функционала страниц 41](#_Toc195794864)

[3.4. Разработка конвертеров 51](#_Toc195794865)

[3.5. Разработка макетов страниц 53](#_Toc195794866)

[4. Тестирование информационной системы 81](#_Toc195794867)

[4.1. Составление чек-листов тестирования приложения 81](#_Toc195794868)

[4.2. Составление тест-кейсов 83](#_Toc195794869)

[Заключение 103](#_Toc195794870)

[Листинг 1 105](#_Toc195794871)

Введение

В современных условиях динамичного развития технологий и увеличения объемов данных, компьютерные системы становятся неотъемлемой частью функционирования предприятий различных сфер. Эффективное использование программного обеспечения (ПО) позволяет оптимизировать бизнес-процессы, улучшить взаимодействие между сотрудниками и повысить качество предоставляемых услуг. В связи с этим, разработка прикладных решений, способных удовлетворить специфические потребности предприятий, становится актуальной задачей.

Целью данной производственной практики является анализ предприятия с целью разработки прикладного решения, которое будет отвечать его требованиям и задачам. Практика включает в себя несколько ключевых этапов: анализ предметной области, проектирование информационной системы, разработка приложения, а также его тестирование.

Часть 1: Анализ предприятия

На первом этапе практики будет проведен анализ деятельности предприятия, что включает в себя изучение его функционала и специфики работы. Будет описано аппаратное и программное обеспечение, используемое в организации, а также организован обмен данными между сотрудниками, включая наличие локальной сети и корпоративной почты. Также будет рассмотрено использование средств защиты информации, таких как антивирусное программное обеспечение, пароли и электронно-цифровая подпись.

Часть 2: Проектирование информационной системы

На следующем этапе будет составлен план разработки прикладного решения, который включает в себя основные этапы и сроки выполнения работ. Будут созданы макеты страниц разрабатываемого приложения, что позволит визуализировать интерфейс и функциональность системы.

Часть 3: Разработка информационной системы

На этапе разработки будет создано прикладное решение, которое должно решать конкретную проблему предприятия. Приложение будет реализовано с графическим интерфейсом, соответствующим стилю и требованиям организации, и может быть как настольным, так и мобильным.

Часть 4: Тестирование информационной системы

Тестирование приложения будет включать в себя инспекцию кода для выявления потенциальных ошибок, составление чек-листа тестирования и создание тест-кейсов. Будет проведено тестирование согласно разработанным тест-кейсам, а также модульное тестирование для методов и свойств, принимающих или возвращающих значения. В процессе тестирования будут выявлены ошибки и предложены пути их решения. После доработки приложения будет проведена повторная инспекция кода и тестирование.

Таким образом, данная производственная практика направлена на получение практических навыков в разработке программного обеспечения, а также на понимание процессов, связанных с проектированием, реализацией и тестированием информационных систем. Результаты практики позволят более глубоко понять, как создаются эффективные и надежные программные решения для современных предприятий.

1. Анализ предприятия для разработка прикладного решения
   1. Описание деятельности предприятия

Государственное автономное учреждение Нижегородской области «Центр координации проектов цифровой экономики» (далее-Автономное учреждение), является некоммерческой организацией и действует в соответствии с настоящим Уставом, Гражданским кодексом РФ, Федеральным законом от 3 ноября 2006 г. № 174-ФЗ «Об автономных учреждениях» и иными нормативно-правовыми актами.

Автономное учреждение создано путем изменения типа государственного бюджетного учреждения Нижегородской области «Центр информационных технологий Нижегородской области» на основании распоряжения Правительства Нижегородской области от 6 сентября 2019 № 925-р.

Место нахождения и почтовый адрес Автономного учреждения: 603006, г. Нижний Новгород, ул. Ошарская, дом 14.

Официальное полное наименование: государственное автономное учреждение Нижегородской области «Центр координации проектов цифровой экономики».

Сокращенное наименование – ГАУ НО «ЦИТ».

Учредителем Автономного учреждения является Нижегородская область, функции и полномочия Учредителя осуществляет министерство информационных технологий и связи Нижегородской области (далее – Учредитель).

Имущество Автономного учреждения находится в государственной собственности.

Функции и полномочия собственника имущества Автономного учреждения от имени Нижегородской области осуществляет министерство имущественных и земельных отношений Нижегородской области.

Автономное учреждение с момента государственной регистрации является юридическим лицом в порядке, установленном законодательством Российской Федерации.

Автономное учреждение от своего имени может приобретать и осуществлять имущественные и личные неимущественные права, нести обязанности, быть истцом и ответчиком в суде.

Автономное учреждение имеет самостоятельный баланс и лицевые счета, открытые в министерстве финансов Нижегородской области, печать со своим наименованием, штамп, бланки и другие реквизиты.

Автономное учреждение отвечает по своим обязательствам закрепленным за ним имуществом, за исключением недвижимого имущества и особо ценного движимого имущества, закрепленного за ним учредителем или приобретенного Автономным учреждением за счет средств, выделенных ему учредителем на приобретение этого имущества.

Собственник имущества Автономного учреждения не несет ответственности по обязательствам Автономного учреждения.

Автономное учреждение не отвечает по обязательствам собственника имущества Автономного учреждения.

Целями деятельности Автономного учреждения являются:

- обеспечение функционирования межведомственных информационных систем и инфраструктуры электронного правительства в Нижегородской области;

- обеспечение функционирования и развития системы обеспечения вызова экстренных оперативных служб по единому номеру «112».

Предметом деятельности Автономного учреждения является:

- реализация единой технической политики в сфере цифровизации органов исполнительной власти, органов местного самоуправления, подведомственных организаций и приоритетных отраслей экономики Нижегородской области;

- обеспечение функционирования межведомственных информационных систем и инфраструктуры электронного правительства в Нижегородской области;

- обеспечение функционирования и развития системы обеспечения вызова экстренных оперативных служб по единому номеру «112».

Для достижения целей, указанных в пункте 2.1. настоящего Устава, Автономное учреждение осуществляет в установленном законодательством Российской Федерации порядке следующие виды деятельности:

- научно-исследовательская деятельность в сфере информационно-коммуникационных технологий;

- выполнение функций центра компетенций по цифровизации городского хозяйства;

- консультирование по аппаратным средствам вычислительной техники;

- разработка программного обеспечения и консультирование в этой области;

- деятельность по созданию и использованию баз данных и информационных ресурсов;

- техническое обслуживание и ремонт офисных машин и вычислительной техники;

- прочая деятельность, связанная с использованием вычислительной техники и информационных технологий;

- деятельность в области электросвязи и почтовой связи;

- организация эксплуатации системы обеспечения вызова экстренных оперативных служб по единому номеру «112» на территории Нижегородской области;

- прием по номеру «112» вызовов (сообщений о происшествиях), в том числе на иностранных языках;

- получение от оператора связи сведений о местонахождении лица, обратившегося по номеру «112», и (или) абонентского устройства, с которого был осуществлен вызов (сообщение о происшествии), а также иных данных, необходимых для обеспечения реагирования по вызову (сообщению о происшествии);

- анализ поступающей информации о происшествиях;

- направление информации о происшествиях, в том числе вызовов (сообщений о происшествиях), в дежурно-диспетчерские службы экстренных оперативных служб в соответствии с их компетенцией для организации экстренного реагирования;

- обеспечение дистанционной психологической поддержки лицу, обратившемуся по номеру «112»;

- автоматизированное восстановление соединения с пользовательским (оконечным) оборудованием лица, обратившегося по номеру «112», в случае внезапного прерывания соединения;

- регистрация всех входящих и исходящих вызовов (сообщений о происшествиях) по номеру «112»;

- ведение базы данных об основных характеристиках происшествий, о начале, завершении и об основных результатах экстренного реагирования на полученные вызовы (сообщения о происшествиях);

- организация и осуществление информационного обмена со всеми экстренными оперативными службами (пожарной охраны, полиции, скорой медицинской помощи, аварийной газовой службы, службы «Антитеррор» и организаций в области защиты населения и территории Нижегородской области от чрезвычайных ситуаций природного, техногенного характера в соответствии с их компетенцией), для обеспечения реагирования по вызову (сообщению о происшествии);

- осуществление работы по комплексной защите информации, информационной безопасности и защите персональных данных в информационных системах;

- организация учета, хранения, сбережения, эксплуатации и технического обслуживания оборудования средств связи и оповещения, средств вычислительной и оргтехники, находящихся на балансе Автономного учреждения;

- осуществление подготовки предложений в проекты нормативных правовых актов Нижегородской области по вопросам функционирования и развития системы-112;

- заключение договоров (соглашений) о взаимодействии с юридическими лицами по вопросам обмена информацией в рамках функционирования системы-112;

- осуществление функции государственного заказчика в соответствии с действующим законодательством о размещении заказов для государственных нужд;

- осуществление работы со сведениями, составляющими государственную тайну, и обеспечение сохранности сведений, составляющих государственную тайну.

Автономное учреждение не вправе осуществлять виды деятельности, не предусмотренные настоящим Уставом.

Автономное учреждение выполняет государственное задание, установленное Учредителем в соответствии с предусмотренной настоящим Уставом основной деятельностью.

Кроме государственного задания Автономное учреждение вправе выполнять работы, оказывать услуги, относящиеся к его основной деятельности, для граждан и юридических лиц за плату и на одинаковых при оказании однородных услуг условиях в порядке, установленном федеральными законами.

Финансовое обеспечение выполнения государственного задания осуществляется в виде субсидий из областного бюджета

Организационная структура предприятия

Отделы и Группы:

**Отдел информационной безопасности:** - Обеспечение безопасности информационных систем и данных. - Разработка и внедрение политики информационной безопасности. - Мониторинг и анализ угроз информационной безопасности. - Реагирование на инциденты информационной безопасности.

**2. Department of Implementation and Soprovoiding Software Products:** - Внедрение и сопровождение программных продуктов, использование в учреждениях. - Техническая поддержка пользователей. - Разработка и реализация документации по программным продуктам.

**3. Группа развития и сопровождения государственных услуг и межведомственного взаимодействия:** - Разработка и сопровождение систем, обеспечивающих предоставление государственных услуг (КП, РГУ, ГИС ГМП). - Мониторинг сбора отчётности. - Обеспечение межведомственного взаимодействия в сфере предоставления государственных услуг.

**4. Группа развития и сопровождения систем ОИВ НО:** - Разработка и сопровождение систем, Использование органов исполнительной власти Нижегородской области. - Обеспечение безопасности дорожного движения с использованием коммуникационных технологий.

**5. Управление по реализации технической политики в сфере цифровой экономики:** - Разработка и реализация технической политики в сфере цифровой экономики. - Координация проектов и инициатив в сфере цифровой экономики. - Взаимодействие с государственными органами и организациями.

**6. Отдел развития услуг цифровой экономики в сфере гос. управления:** - Разработка и внедрение сервисов цифровой экономики для государственного управления. - Оптимизация процессов государственного управления с помощью цифровых технологий.

**7. Отдел развития сервисов цифровой экономики в приоритетных отраслях экономики:** - Разработка и внедрение сервисов цифровой экономики для приоритетных отраслей экономики Нижегородской области. - Содействие развитию цифровой экономики в регионе.

**8. Управление правового и финансового обеспечения:** - Обеспечение правового сопровождения деятельности учреждения. - Финансовое планирование и управление. - Контроль за расходами.

**9. Планово-финансовый отдел:** - Разработка и исполнение бюджета. - Financial analysis and reporting.

**10. Организационно-правовой отдел:** - Обеспечение организационно-правового сопровождения деятельности Учреждения. - Ведение кадровой документации. - Обеспечение документооборота.

**11. Управление администрирования и видеонаблюдения:** - Управление информационными системами и оборудованием. - Обеспечение функционирования сети и инфраструктуры.

**12. Отдел видеонаблюдения:** - Обеспечение функционирования системы видеонаблюдения. - Мониторинг и запись видеоизображений.

**13. Отдел системного администрирования узла связи и корпоративной сети ОИВ:** - Обеспечение функционирования узла связи и корпоративной сети органов исполнительной власти Нижегородской области. - Техническая поддержка пользователей.

**14. Отдел обеспечения** функционирования системы экстренного вызова «112». - Прием и обработка вызовов. - Координация действий экстренных служб.

**15. Центр обработки вызовов СИСТЕМЫ - 112:** - Прием и обработка вызовов системы “112”. - Направление вызовов соответствующим экстренным службам.

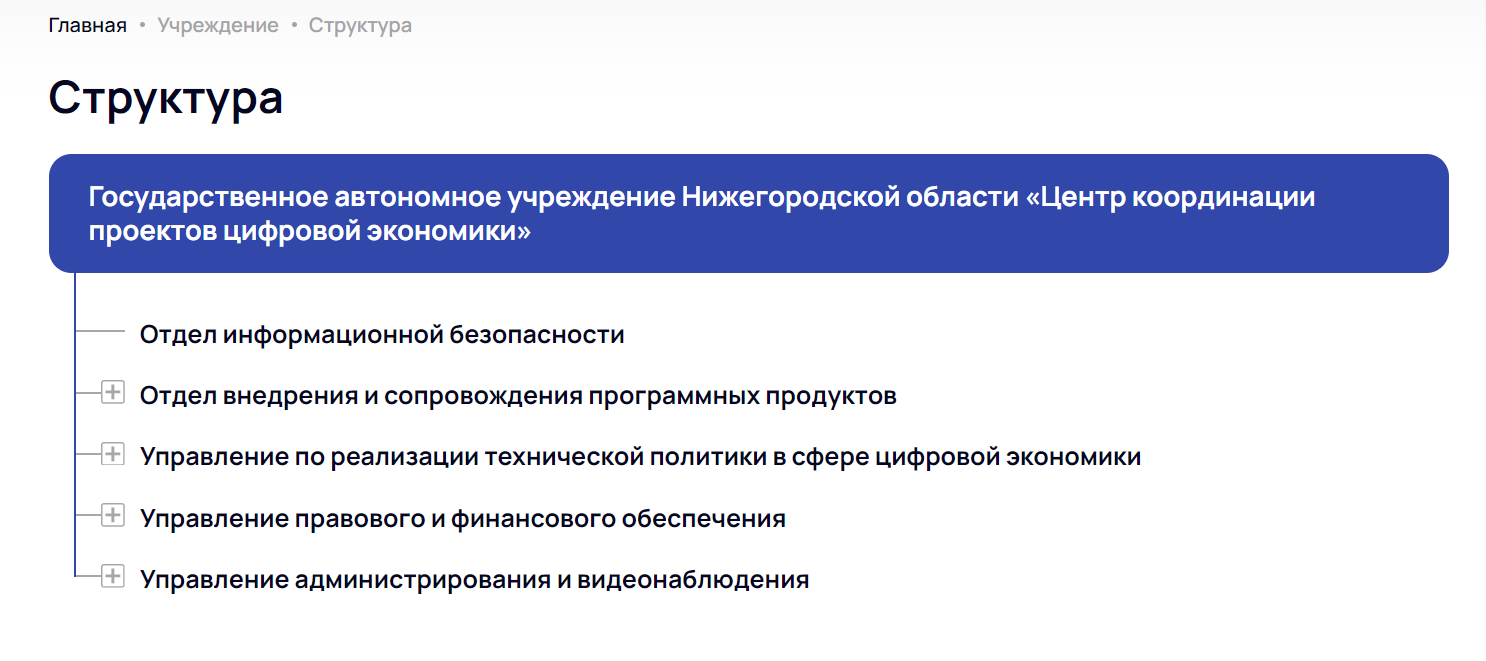


Рисунок 1 – Структура предприятия

* 1. Аппаратное и программное обеспечение предприятия

ГАУ НО ЦИТ использует современное оборудование от предварительных производителей, обеспечивающих высокую производительность и стабильную работу.

Серверы:

Типы серверов: Используются как физические серверы, так и виртуальные машины для оптимизации использования ресурсов.

Производительность: Серверы оснащены мощными процессорами, оперативной памятью Tom и быстрыми накопителями (SSD, NVMe) для обеспечения высокой скорости обработки данных.

Назначение: Серверы используются для хранения и обработки данных, в том числе для работы с базами данных, хранения веб-сайтов и сервисов, обеспечения доступа к информационным ресурсам и приложениям.

Безопасность: Серверы оборудованы системами безопасности, включая брандмауэры, антивирусное программное обеспечение и системы мониторинга для защиты данных от несанкционированного доступа и угроз.

Рабочие станции:

Типы: Используются стационарные компьютеры (ПК) и ноутбуки, оснащенные американскими процессорами, дополнительными устройствами оперативной памяти и жесткими дисками для эффективной работы с приложениями и данными.

Назначение: Рабочие наши сделки каждому сотруднику, чтобы обеспечить им удобное и эффективное рабочее место для выполнения своих задач.

Программное обеспечение: Рабочие станции оснащены необходимым программным обеспечением, включая операционную систему, пакет Microsoft Office, специализированные программы, а также антивирусное программное обеспечение.

Сеть:

Тип: Приведенная структурированная кабельная сеть с высокой пропускной способностью для обеспечения бесперебойного доступа к информационным ресурсам.

Безопасность: Сеть оснащена системами безопасности, включая брандмауэры, фильтры контента, системы аутентификации и авторизации.

Мониторинг: Используются системы мониторинга для контроля работы сети, выявляют аномалии и своевременность на проблемах.

Периферия:

Принтеры: Используются как черно-белые, так и цветные принтеры для печати документов, отчетов и других материалов.

Сканер: Используются сканеры для цифровизации документов, чертежей и других материалов.

Мультифункциональные устройства: Используются мультикитные устройства, входящие в состав функций принтера, сканера и копира.

Системы видеонаблюдения:

Типы: Используются цифровые системы видеонаблюдения с высоким разрешением и функциями записи и архивации.

Назначение: Системы видеонаблюдения обеспечивают круглосуточный мониторинг территории и помещений учреждений, чтобы повысить безопасность и предотвратить возникающие инциденты.

Technical providing:

Система электропитания: Обеспечивает бесперебойное питание всего оборудования, включая резервные источники питания (ИБП) для предотвращения сбоев в работе в случае отключения электроэнергии.

Система кондиционирования: Обеспечивает оранжевый температурный режим в серверных комнатах и ​​других помещениях с оборудованием для гарантии стабильной работы оборудования.

Система огнетушения: Обеспечивает пожарную безопасность в посторонних комнатах и других помещениях с оборудованием для предотвращения пожара и его последствий.

Описание инструментальных средств разработки и сопровождения программных средств

ГАУ НО ЦИТ использует широкий спектр программного обеспечения для эффективной работы сотрудников:

Операционные системы: В основном используется операционная система Windows.

Системы управления базами данных (СУБД): Для хранения и обработки данных используются современные СУБД, например, Oracle, PostgreSQL, MySQL.

Системы электронного документооборота (СЭД): Обеспечивают электронный обмен документами между сотрудниками и внешними организациями.

Программные продукты для разработки: Используются языки программирования (Java, Python, JavaScript, C#), фреймворки и инструменты для создания информационных систем и веб-приложений.

Специализированные программы: В зависимости от задач сотрудников используются специализированные программы для работы с графикой, мультимедиа, аналитикой данных, моделированием и т.д.

Программное обеспечение для обеспечения информационной безопасности: Обеспечивает защиту данных и информационных систем от угроз.

* 1. Организация обмена данных между сотрудниками

Для обеспечения эффективного обмена данными между сотрудниками нашей организации мы внедрили комплексную систему, включающую локальную сеть, корпоративную почту и сетевые папки. Это позволяет значительно упростить коммуникацию и повысить продуктивность работы.

1. Корпоративная почта

Используется корпоративный почтовый домен cit-nnov.ru, который предоставляет каждому сотруднику уникальный адрес электронной почты. Это обеспечивает:

Безопасность и конфиденциальность: Корпоративная почта защищена от внешних угроз, таких как спам и фишинг, что гарантирует безопасность деловой переписки.

Упрощенный обмен информацией: Сотрудники могут легко обмениваться документами, сообщениями и другой информацией, что способствует быстрому решению задач.

Календарь и планирование: Корпоративная почта включает функции календаря, что позволяет сотрудникам планировать встречи и мероприятия, а также отслеживать важные сроки.

2. Локальная сеть

Локальная сеть (LAN) является основой для внутренней коммуникации и обмена данными. Она обеспечивает:

Высокую скорость передачи данных: Сотрудники могут быстро обмениваться файлами и документами без задержек, что особенно важно для работы с объемными данными.

Доступ к общим ресурсам: Все сотрудники имеют доступ к сетевым папкам и общим ресурсам, что упрощает совместную работу над проектами и документами.

Централизованное управление: Локальная сеть позволяет IT-отделу централизованно управлять доступом к ресурсам, обеспечивая защиту данных и контроль за их использованием.

3. Сетевые папки

Сетевые папки играют важную роль в организации обмена данными. Они позволяют:

Хранение и совместное использование файлов: Сотрудники могут сохранять документы в общих папках, что делает их доступными для всех, кто работает над проектом.

Упрощение версионности: Все изменения в документах фиксируются, что позволяет отслеживать изменения и возвращаться к предыдущим версиям при необходимости.

Организация информации: Сетевые папки могут быть структурированы по проектам, отделам или темам, что упрощает поиск нужной информации.

Заключение

Организация обмена данными между сотрудниками через локальную сеть и корпоративную почту создает удобные условия для работы, способствует эффективному взаимодействию и повышает общую продуктивность. Наличие почтового домена cit-nnov.ru и хорошо структурированной локальной сети позволяет нашей организации легко справляться с любыми вызовами, связанными с обменом информации и совместной работой.

* 1. Использование антивирусного программного обеспечения и других средств защиты

В организации безопасность информационных систем является приоритетом. Для защиты данных и обеспечения надежной работы сетевой инфраструктуры мы используем антивирусное программное обеспечение и специализированные средства защиты информации. Рассмотрим подробнее, как именно это реализовано.

1. Антивирусное программное обеспечение

Одним из ключевых компонентов нашей системы безопасности является антивирусное программное обеспечение, в частности, Kaspersky. Этот продукт обеспечивает:

Защита от вирусов и вредоносного ПО: Kaspersky эффективно обнаруживает и блокирует вирусы, трояны, шпионские программы и другие угрозы, которые могут повредить данные или нарушить работу систем.

Регулярные обновления: Антивирусное ПО автоматически обновляется, что позволяет ему оставаться актуальным и эффективно противостоять новым угрозам, появляющимся на рынке.

Сканирование в реальном времени: Программа обеспечивает защиту в реальном времени, что позволяет обнаруживать и блокировать угрозы на этапе их появления, минимизируя риск заражения.

Безопасный интернет-серфинг: Kaspersky включает функции защиты при работе в интернете, такие как блокировка вредоносных сайтов и предупреждения о фишинговых атаках, что помогает сотрудникам безопасно пользоваться ресурсами сети.

2. Средства криптографической защиты информации (СКЗИ)

Кроме антивирусного программного обеспечения, в нашей организации используются средства криптографической защиты информации (СКЗИ), которые обеспечивают дополнительный уровень безопасности. В частности, мы применяем следующие решения:

Континент АП: Это система, обеспечивающая защиту конфиденциальной информации с помощью криптографических алгоритмов. Она позволяет шифровать данные и управлять доступом к ним, что критически важно для защиты чувствительной информации от несанкционированного доступа.

ViPNet: Эта система обеспечивает защищенные каналы связи между пользователями и серверами, что позволяет безопасно передавать данные по сети. ViPNet использует технологии виртуальных частных сетей (VPN) и шифрования, что делает коммуникацию между сотрудниками безопасной и защищенной от внешних угроз.

3. Комплексный подход к безопасности

Использование антивирусного программного обеспечения и средств криптографической защиты информации в нашей организации создает комплексный подход к безопасности. Это включает:

Многоуровневая защита: Комбинация антивирусного ПО и СКЗИ обеспечивает многоуровневую защиту, что минимизирует риски утечки данных и вредоносных атак.

Обучение сотрудников: Важно не только использовать технологии, но и обучать сотрудников основам кибербезопасности, чтобы они могли распознавать угрозы и правильно реагировать на них.

Регулярные проверки и обновления: Проводятся регулярные проверки систем безопасности и обновления программного обеспечения, что позволяет поддерживать высокий уровень защиты от новых угроз.

1. Проектирование информационной системы
   1. Составление плана этапов разработки прикладного решения для предприятия

Разработка прикладного решения для предприятия — это сложный процесс, который требует четкого планирования и организации. Ниже представлен план этапов, который поможет структурировать процесс разработки, используя класс Test1VM как основу для создания тестового приложения.

Этап 1: Анализ требований

Сбор требований:

Провести встречи с заинтересованными сторонами (менеджерами, пользователями, ИТ-отделом) для определения функциональных и нефункциональных требований.

Определить, какие функции должны быть реализованы (например, создание тестов, проверка ответов, управление пользователями).

Документация требований:

Оформить требования в виде документа, который будет служить основой для следующих этапов разработки.

Этап 2: Проектирование архитектуры

Определение архитектуры:

Выбрать архитектурный стиль

Определить основные компоненты приложения (ViewModels, Views, Models).

Проектирование базы данных:

Определить структуру базы данных для хранения вопросов, ответов и пользователей.

Создать ER-диаграмму и описать таблицы и их связи.

Создание прототипа интерфейса:

Разработать макеты пользовательского интерфейса, учитывая удобство использования и доступность.

Этап 3: Разработка

Настройка окружения разработки:

Установить необходимые инструменты и библиотеки (например, .NET, Entity Framework, MVVM библиотеки).

Реализация моделей:

Создать классы для работы с данными (например, Questions1, User , UsersTest).

Реализация ViewModels:

Реализовать классы ViewModel, такие как Test1VM, для управления логикой приложения и взаимодействия с пользовательским интерфейсом.

Создание представлений (Views):

Разработать пользовательский интерфейс, используя XAML или другой подходящий язык разметки.

Реализация команд и методов:

Написать команды для обработки действий пользователя (например, CheckAnswersCommand, Delete, Update).

Этап 4: Тестирование

Модульное тестирование:

Написать тесты для проверки отдельных компонентов (например, методов в Test1VM).

Интеграционное тестирование:

Проверить взаимодействие между компонентами (например, как ViewModel взаимодействует с Model и View).

Тестирование пользовательского интерфейса:

Провести тестирование интерфейса с реальными пользователями, чтобы получить обратную связь.

Этап 5: Внедрение

Подготовка к внедрению:

Подготовить документацию пользователя и техническую документацию для поддержки приложения.

Обучение пользователей:

Провести обучение для конечных пользователей, чтобы они могли эффективно использовать приложение.

Развертывание:

Установить приложение на сервер или предоставить его для скачивания пользователям.

Этап 6: Поддержка и сопровождение

Мониторинг работы приложения:

Отслеживать производительность и стабильность приложения.

Обработка отзывов и исправление ошибок:

Собирать отзывы от пользователей и вносить улучшения в приложение.

Обновление и расширение функционала:

Планировать и реализовывать новые функции на основе требований пользователей и изменений в бизнес-процессах.

* 1. Создание макетов страниц разрабатываемого прикладного решения

Необходимость создания макетов страниц в процессе разработки прикладного решения, включая его цели, преимущества и влияние на весь процесс разработки.

1. Визуализация интерфейса

Описание:  
Макеты страниц служат визуальным представлением того, как будет выглядеть конечный продукт. Это может быть как низкопрофильный (wireframe), так и высокопрофильный (high-fidelity) прототип.

Преимущества:

Ясность: Позволяет всем участникам проекта увидеть, как будет выглядеть интерфейс, что помогает избежать недопонимания.

Согласование: Можно быстро получить обратную связь от заинтересованных сторон и внести изменения до начала разработки.

Обсуждение: Упрощает процесс обсуждения дизайна и функциональности, так как все видят одно и то же.

2. Упрощение коммуникации

Описание:  
Макеты служат общим языком для всех участников проекта, включая разработчиков, дизайнеров, менеджеров и клиентов.

Преимущества:

Единое представление: Все участники имеют возможность видеть и обсуждать один и тот же макет, что снижает вероятность недопонимания.

Обсуждение изменений: Легче обсуждать изменения и получать согласие, когда есть визуальная основа.

Документация: Макеты могут служить частью документации проекта, фиксируя согласованные решения.

3. Тестирование пользовательского опыта (UX)

Описание:  
Макеты позволяют проводить тестирование пользовательского опыта на ранних этапах разработки.

Преимущества:

Обратная связь от пользователей: Можно провести тестирование с реальными пользователями, чтобы понять, насколько интерфейс удобен.

Идентификация проблем: Выявление проблем с навигацией, расположением элементов и общим восприятием интерфейса до начала разработки.

Итеративный процесс: Легче вносить изменения в макеты, чем в уже написанный код.

4. Экономия времени и ресурсов

Описание:  
Создание макетов позволяет избежать значительных затрат времени и ресурсов на доработку уже созданного кода.

Преимущества:

Снижение затрат: Исправление ошибок и недочетов на этапе макета обходится гораздо дешевле, чем на этапе разработки.

Упрощение процесса: Легче и быстрее вносить изменения в макет, чем в готовый продукт.

Планирование: Позволяет более точно планировать время и ресурсы на разработку.

5. Формирование требований

Описание:  
Макеты помогают в формировании и уточнении требований к функциональности приложения.

Преимущества:

Уточнение требований: Позволяют выявить и уточнить требования до начала разработки, что снижает риск изменений на более поздних этапах.

Согласование с клиентами: Можно легко показать клиенту, как будет выглядеть продукт, и получить его согласие на функциональность.

Документирование требований: Макеты могут служить основой для составления технического задания.

6. Поддержка дизайна

Описание:  
Создание макетов позволяет дизайнерам экспериментировать с различными элементами интерфейса, цветами, шрифтами и компоновкой.

Преимущества:

Креативность: Дизайнеры могут свободно экспериментировать с дизайном, не беспокоясь о технических ограничениях.

Стандартизация: Помогает установить стандарты и гайдлайны для всего приложения.

Обратная связь: Легче получать обратную связь от команды и пользователей на ранних этапах.

7. Упрощение разработки

Описание:  
Четкие макеты помогают разработчикам лучше понять, как реализовать функциональность.

Преимущества:

Снижение ошибок: Четкие визуальные инструкции помогают избежать недоразумений в процессе кодирования.

Планирование задач: Разработчики могут более точно оценивать время, необходимое для реализации каждой функции.

Командная работа: Упрощает взаимодействие между дизайнерами и разработчиками, так как у них есть общий ориентир.

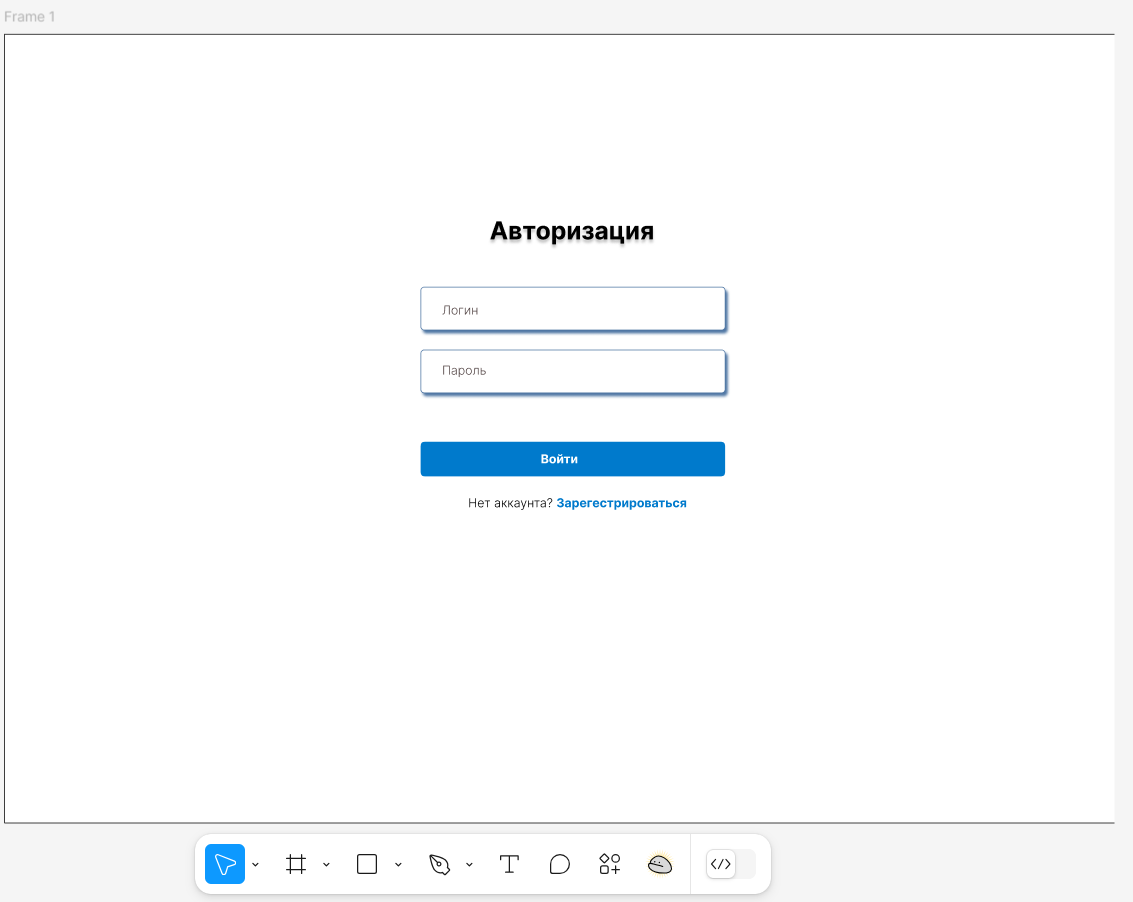


Рисунок 1 – страница авторизации

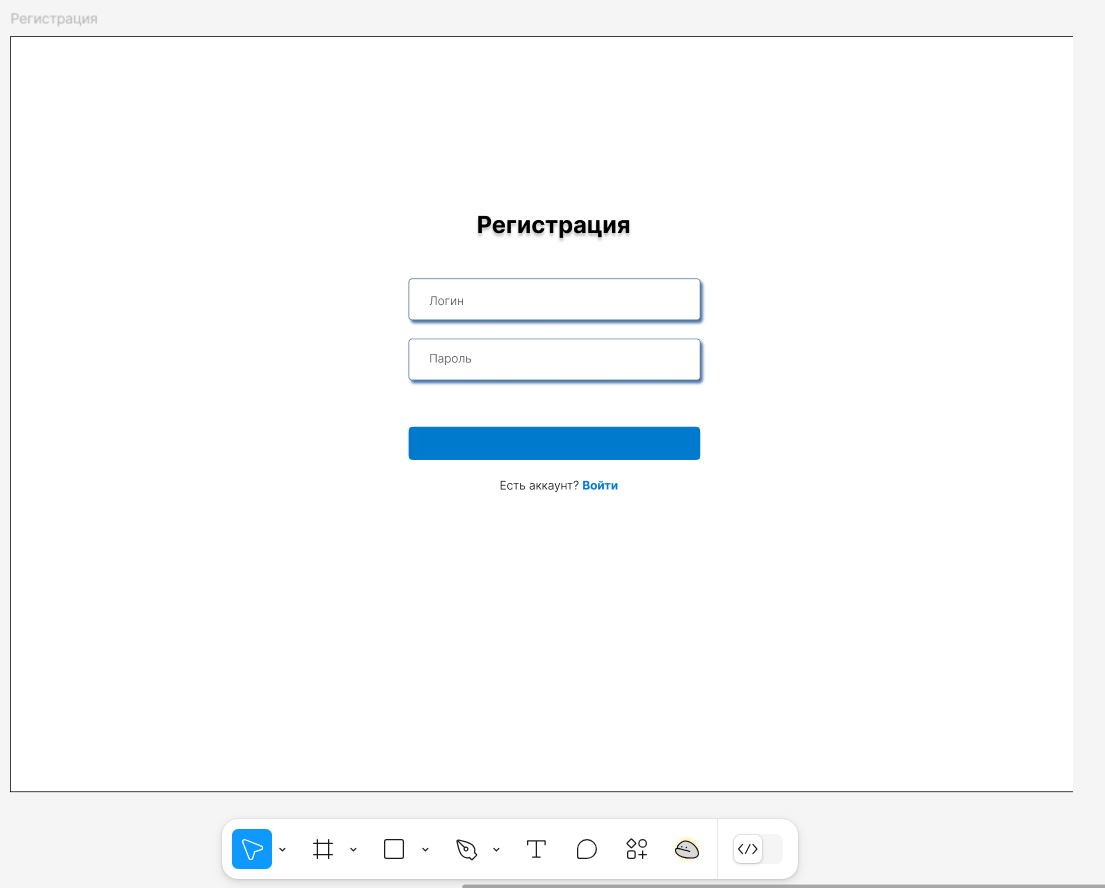


Рисунок 2 – страница регистрации

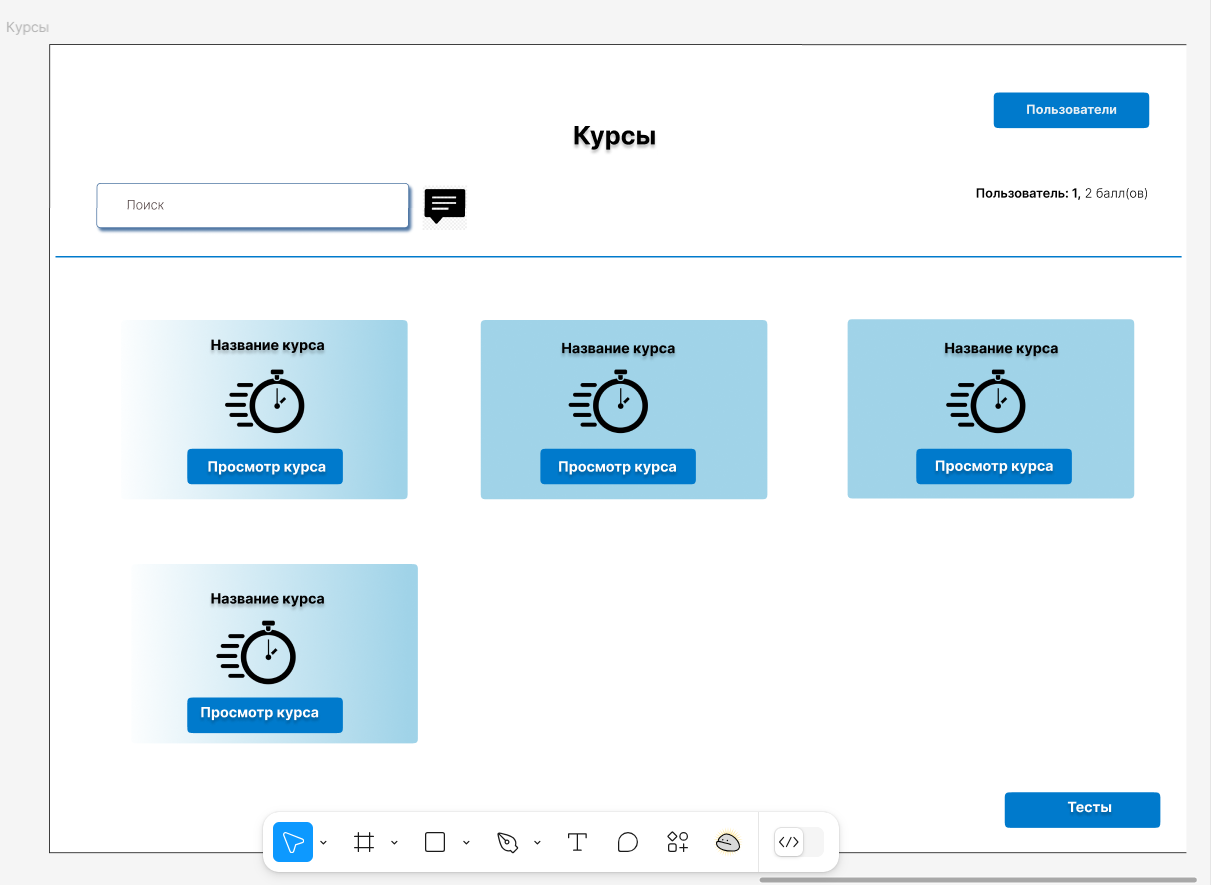


Рисунок 3 – страница Курсы

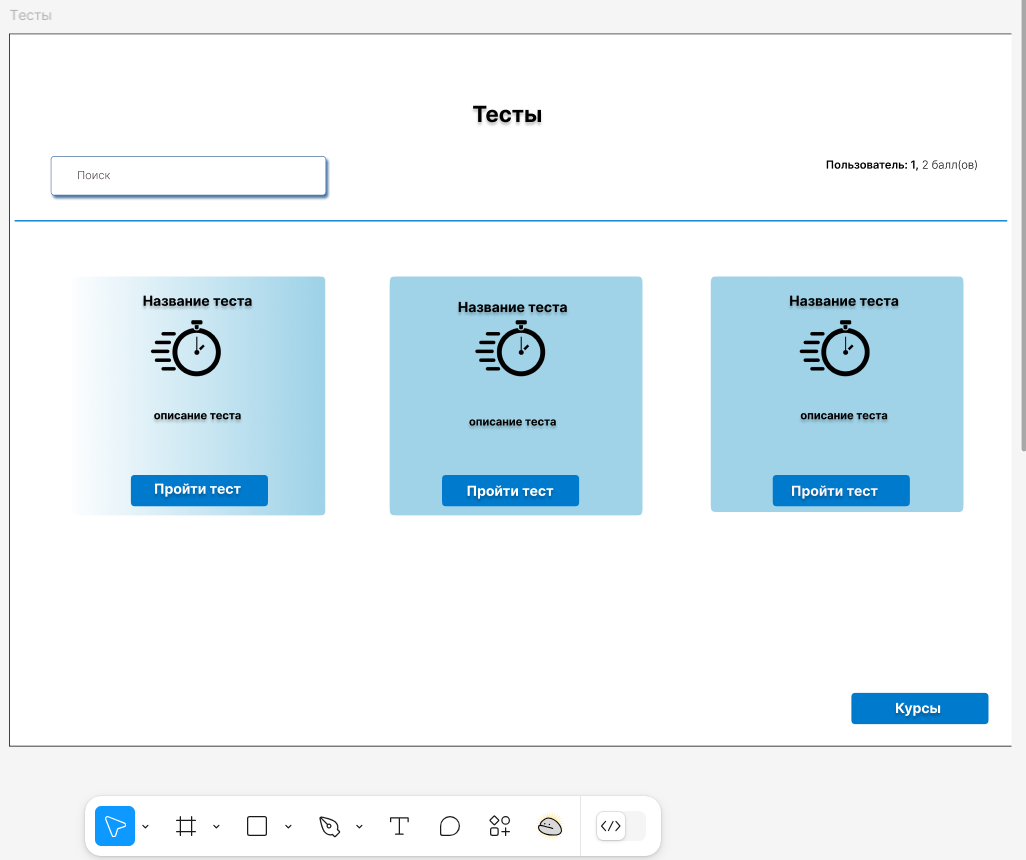


Рисунок 4 – страница Тесты

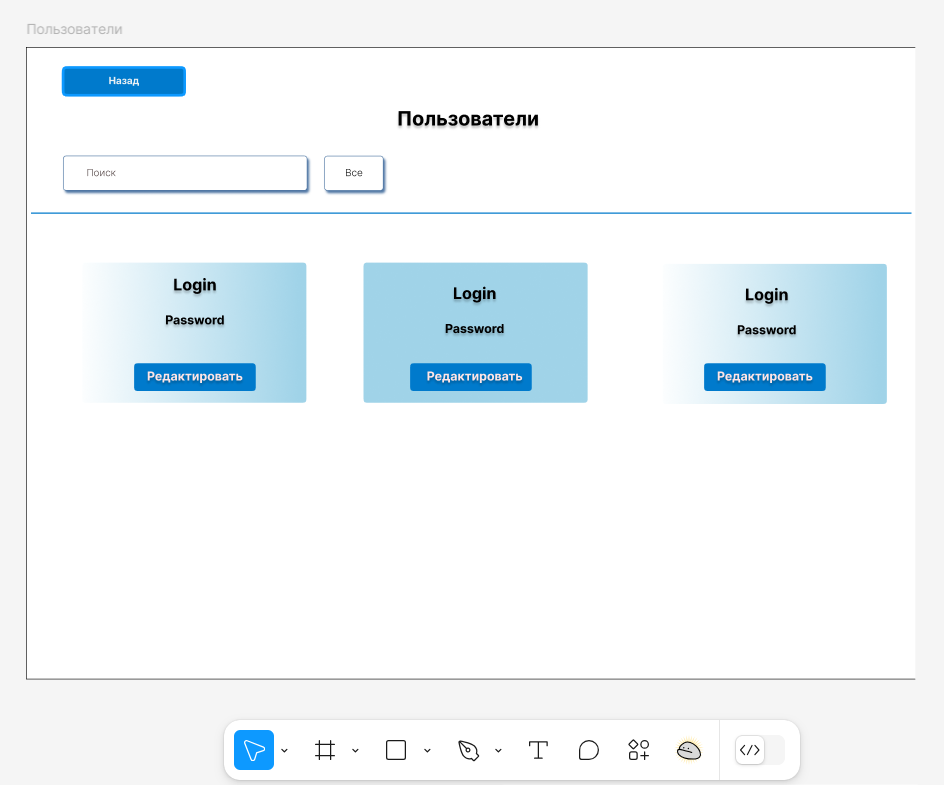


Рисунок 5 – страница Пользователи

1. Разработка информационной системы для предприятия
   1. Разработка базы данных

Созданная система представляет собой комплексное решение для управления процессами, связанными с документами, заданиями и тестированием в организации. Она предназначена для оптимизации работы с различными типами данных, повышения эффективности взаимодействия между пользователями и упрощения процессов обработки информации. (см. Листинг 1)

Основные компоненты системы

Роли пользователей:

Система поддерживает различные роли пользователей, что позволяет настраивать доступ к функционалу в зависимости от их обязанностей. Каждому пользователю назначается определенная роль, которая определяет его права и возможности в системе.

Пользователи:

Каждый пользователь имеет уникальный логин и пароль для входа в систему. Профили пользователей содержат информацию о назначенной роли, что позволяет контролировать доступ к различным функциям приложения.

Тестирование:

Система включает модуль тестирования, где пользователи могут проходить тесты, связанные с различными аспектами работы с документами и заданиями. Тесты содержат вопросы с несколькими вариантами ответов, что позволяет оценить уровень знаний пользователей.

Документы и задания:

Система позволяет создавать, редактировать и управлять документами и заданиями. Пользователи могут добавлять корреспондентов, назначать исполнителей и отслеживать историю операций с документами. Это обеспечивает прозрачность и контроль за выполнением задач.

Курсы и обучение:

Для повышения квалификации сотрудников предусмотрены курсы, которые содержат информацию о различных операциях с документами и заданиями. Каждый курс включает описание и визуальные материалы, что делает процесс обучения более доступным и понятным.

Вопросы и тесты:

В системе реализованы таблицы для хранения вопросов, связанных с тестами. Каждый вопрос имеет текст, правильный ответ и несколько вариантов ответов. Это позволяет создавать разнообразные тесты, которые могут использоваться для оценки знаний пользователей.

Функционал поиска:

Система предоставляет возможности для быстрого и полнотекстового поиска документов и заданий. Пользователи могут фильтровать результаты поиска по различным параметрам, что значительно упрощает процесс нахождения необходимой информации.

Загрузка и управление файлами:

В системе реализован функционал для загрузки файлов, что позволяет пользователям добавлять необходимые документы и скан-образы. При этом предусмотрены настройки для управления загруженными файлами и их дальнейшей обработкой.

Преимущества системы

Удобство использования: Интуитивно понятный интерфейс и четкая структура системы позволяют пользователям быстро ориентироваться в функционале и выполнять необходимые действия.

Гибкость и адаптивность: Возможность настройки ролей и прав доступа позволяет адаптировать систему под конкретные потребности организации.

Эффективность обучения: Наличие курсов и тестов способствует повышению квалификации сотрудников и улучшению их знаний о процессах, происходящих в организации.

Контроль и аналитика: Система обеспечивает возможность отслеживания выполнения заданий и тестов, что позволяет анализировать эффективность работы сотрудников и выявлять области для улучшения.

* 1. Разработка моделей таблиц

Модель Answer

Назначение классов

Организация вопросов и ответов: Каждый класс представляет отдельный вопрос, что позволяет организовать вопросы и связанные с ними ответы в логически разделенные единицы. Это особенно полезно при разработке приложений для тестирования, опросов или викторин, где необходимо четко структурировать информацию.

Упрощение доступа к ответам: Свойство Answers в каждом классе возвращает массив строк, содержащий возможные ответы на соответствующий вопрос. Это упрощает доступ к ответам, позволяя разработчикам и пользователям легко получать список ответов, не обращаясь к каждому отдельному свойству (например, Answer1, Answer2, Answer3).

Функционал

Хранение данных: Каждый класс может содержать дополнительные свойства, такие как текст вопроса, правильный ответ и другие метаданные, которые могут быть полезны для логики приложения. Например, можно добавить свойства для хранения текста вопроса или индикатора правильного ответа.

Группировка логики: Эти классы могут быть частью более крупной системы, которая управляет вопросами и ответами. Например, они могут быть использованы в приложении для викторины, где пользователи выбирают ответы на вопросы, и система проверяет правильность ответов.

Расширяемость: Структура классов позволяет легко добавлять новые вопросы, просто создавая новые классы (например, Questions8, Questions9 и т.д.) или расширяя существующие классы, если это необходимо. Это делает систему гибкой и адаптируемой к изменениям.

Поддержка различных типов вопросов: В будущем можно расширить функционал, добавив поддержку различных типов вопросов (например, множественный выбор, открытые вопросы и т.д.), просто добавив дополнительные методы или свойства в классы.

Интерфейс взаимодействия: Эти классы могут быть использованы для создания интерфейса взаимодействия с пользователем, где каждый вопрос и соответствующие ответы отображаются на экране, позволяя пользователю взаимодействовать с приложением.

Класс Course представляет собой модель данных, используемую для хранения информации о курсе в приложении. Давайте подробно рассмотрим каждое из свойств этого класса и его назначение. Свойства класса

Id:

Тип: int

Описание: Уникальный идентификатор курса. Это свойство обычно используется для различения различных курсов в базе данных. Оно должно быть уникальным для каждого курса.

Name:

Тип: string?

Описание: Название курса. Это свойство может содержать название курса, например, "Основы программирования". Использование string? указывает на то, что это свойство может быть null, что позволяет не указывать название курса в некоторых случаях.

Description:

Тип: string?

Описание: Описание курса. Это свойство может содержать более подробную информацию о содержании курса, его целях и темах, которые будут рассмотрены. Также может быть null, если описание не предоставлено.

Photo:

Тип: string?

Описание: Путь к изображению курса. Это свойство может использоваться для хранения URL или относительного пути к изображению, которое будет представлять курс в интерфейсе пользователя. Как и предыдущие свойства, оно может быть null.

Uk:

Тип: int?

Описание: Дополнительное поле, которое может использоваться для различных целей, например, для хранения идентификатора пользователя, который создал курс, или идентификатора категории, к которой относится курс. Это свойство также может быть null.

Класс Offer представляет собой модель данных, используемую для хранения информации о предложениях в приложении. Давайте подробно рассмотрим каждое из свойств этого класса и его назначение. Id:

Тип: int

Описание: Уникальный идентификатор предложения. Это свойство служит для однозначного различения различных предложений в базе данных. Обычно это свойство является первичным ключом, что обеспечивает уникальность каждого предложения.

Offers:

Тип: string?

Описание: Это свойство хранит текст или описание предложения. Оно может содержать информацию о скидках, акциях или специальных предложениях, доступных пользователям. Использование string? указывает на то, что это свойство может быть null, что позволяет не указывать текст предложения в некоторых случаях.

Класс Questions1 представляет собой модель данных, используемую для хранения информации о вопросах в приложении, например, в контексте викторин или тестов. Давайте рассмотрим каждое из свойств этого класса и его назначение. Затем мы создадим однотипные классы Questions2, Questions3, Questions4, Questions5, Questions6, и Questions7. Свойства класса

Id:

Тип: int

Описание: Уникальный идентификатор вопроса, который помогает различать вопросы в базе данных.

Questiontext:

Тип: string?

Описание: Хранит текст вопроса, который будет отображаться пользователю.

Correctanswerindex:

Тип: int

Описание: Индекс правильного ответа (0, 1 или 2 для Answer1, Answer2, Answer3). Это свойство позволяет определить, какой из предложенных ответов является правильным.

Answer1, Answer2, Answer3:

Тип: string?

Описание: Эти свойства хранят текстовые варианты ответов на вопрос. Они могут быть null, если варианты не указаны.

Selectedanswerindex:

Тип: int?

Описание: Индекс ответа, выбранного пользователем. Это свойство может быть null, если пользователь еще не выбрал ответ

Класс Role представляет собой модель данных, которая используется для хранения информации о ролях пользователей в системе. Эта модель может быть частью системы управления доступом, где каждая роль определяет набор прав и привилегий, которые могут быть назначены пользователям. Рассмотрим каждое из свойств класса более подробно.

Описание таблицы Role

Id:

Тип: Целое число (integer).

Описание: Это уникальный идентификатор роли. Он служит для однозначного различения различных ролей в базе данных. Обычно это поле является первичным ключом таблицы, что обеспечивает уникальность каждой записи.

Role1:

Тип: Строка (string).

Описание: Это свойство содержит название роли. Например, это может быть "Администратор", "Пользователь", "Модератор" и т. д. Название роли определяет, какие права и привилегии будут связаны с данной ролью. Это поле не может быть пустым, что гарантирует, что каждая роль будет иметь корректное название.

Users:

Тип: Коллекция пользователей (ICollection<User>).

Описание: Это свойство представляет собой коллекцию пользователей, которые связаны с данной ролью. Каждый пользователь может иметь одну или несколько ролей, и это свойство позволяет легко получить доступ к списку всех пользователей, которые имеют определенную роль. Это свойство инициализируется как новый список, что обеспечивает возможность добавления пользователей без необходимости предварительной инициализации коллекции.

Взаимосвязи и использование

Связь с пользователями: Таблица Role имеет связь с таблицей User , где каждый пользователь может быть связан с одной или несколькими ролями. Это позволяет реализовать гибкую систему управления доступом, где права пользователей могут быть легко изменены путем изменения их ролей.

Применение: Роли могут использоваться для определения уровней доступа к различным функциям системы, ограничения возможностей пользователей и управления правами на выполнение определенных действий. Например, администраторы могут иметь доступ к управлению пользователями и настройкам системы, в то время как обычные пользователи могут иметь доступ только к базовым функциям.

Класс Test представляет собой модель данных, используемую для хранения информации о тестах или экзаменах в системе. Эта модель может быть частью образовательного приложения или платформы для тестирования, где пользователи могут проходить различные тесты. Рассмотрим каждое из свойств класса более подробно.

Описание таблицы Test

Id:

Тип: Целое число (integer).

Описание: Это уникальный идентификатор теста. Он служит для однозначного различения различных тестов в базе данных и обычно является первичным ключом таблицы, что обеспечивает уникальность каждой записи.

Name:

Тип: Строка (string?).

Описание: Это свойство содержит название теста. Название теста должно быть информативным и отражать его содержание, чтобы пользователи могли легко понять, о чем тест. Это поле может быть пустым (null), если название теста не указано.

Description:

Тип: Строка (string?).

Описание: Это свойство предоставляет подробное описание теста. Описание может включать информацию о цели теста, его содержании, количестве вопросов и других важных аспектах. Это поле также может быть пустым (null), если описание не указано.

Photo:

Тип: Строка (string?).

Описание: Это свойство может использоваться для хранения ссылки на изображение, связанное с тестом. Например, это может быть логотип теста или изображение, которое иллюстрирует его тему. Это поле может быть пустым (null), если изображение не предоставлено.

Points:

Тип: Целое число (int?).

Описание: Это свойство указывает количество баллов, которые можно получить за успешное прохождение теста. Это значение может быть использовано для оценки результатов пользователей. Это поле может быть пустым (null), если количество баллов не указано.

UsersTests:

Тип: Коллекция пользователей, связанных с тестами (ICollection<UsersTest>).

Описание: Это свойство представляет собой коллекцию записей, связывающих пользователей с тестами. Каждая запись в этой коллекции может содержать информацию о том, какой пользователь проходил тест, его результаты и другие связанные данные. Это свойство инициализируется как новый список, что позволяет добавлять записи без необходимости предварительной инициализации коллекции.

Взаимосвязи и использование

Связь с пользователями: Таблица Test имеет связь с таблицей UsersTest, что позволяет отслеживать, какие пользователи проходили определенные тесты и какие результаты они получили. Это создает возможность для анализа успеваемости и управления процессом тестирования.

Применение: Модель Test может использоваться в образовательных приложениях для создания, редактирования и управления тестами. Информация о тестах может быть использована для формирования отчетов о результатах пользователей, анализа сложности тестов и оптимизации учебного процесса.

Класс User  представляет собой модель данных для хранения информации о пользователях в системе. Эта модель может быть частью приложения, связанного с тестированием или образовательной платформой. Давайте рассмотрим каждое из свойств класса более подробно.

Описание класса User

Id:

Тип: Целое число (int).

Описание: Уникальный идентификатор пользователя. Это поле служит первичным ключом и позволяет однозначно идентифицировать каждого пользователя в базе данных.

Login:

Тип: Строка (string).

Описание: Это свойство содержит логин пользователя. Логин должен быть уникальным и использоваться для аутентификации пользователя в системе. Поле инициализируется значением null!, что указывает на то, что оно не может быть null.

Password:

Тип: Строка (string).

Описание: Это свойство хранит пароль пользователя. Как и логин, пароль должен быть защищенным и не должен храниться в открытом виде. Поле также инициализируется значением null!, что указывает на то, что оно не может быть null.

Role:

Тип: Целое число (int?).

Описание: Это свойство определяет роль пользователя в системе. Роль может быть представлена в виде целого числа, которое соответствует определенному уровню доступа или типу пользователя (например, администратор, преподаватель, студент и т.д.). Это поле может быть пустым (null), если роль не указана.

RoleNavigation:

Тип: Объект класса Role?.

Описание: Это навигационное свойство, которое устанавливает связь между пользователем и его ролью. Оно позволяет получить доступ к дополнительной информации о роли пользователя, если такая информация хранится в отдельной таблице Role. Это свойство также может быть null, если роль не указана.

UsersTests:

Тип: Коллекция пользователей, связанных с тестами (ICollection<UsersTest>).

Описание: Это свойство представляет собой коллекцию записей, связывающих пользователей с тестами, которые они проходили. Каждая запись в этой коллекции может содержать информацию о том, какой тест прошел пользователь и его результаты. Коллекция инициализируется как новый список, что позволяет добавлять записи без необходимости предварительной инициализации.

Взаимосвязи и использование

Связь с ролями: Поле RoleNavigation устанавливает связь между пользователем и его ролью, что позволяет легко управлять правами доступа и функционалом для различных типов пользователей.

Связь с тестами: Коллекция UsersTests позволяет отслеживать, какие тесты прошел пользователь и какие результаты он получил. Это создает возможность для анализа успеваемости пользователей и управления процессом тестирования.

Класс UsersTest представляет собой модель данных, которая хранит информацию о тестах, прошедших пользователями. Давайте рассмотрим каждое из свойств этого класса более подробно.

Описание класса UsersTest

Id:

Тип: Целое число (int).

Описание: Уникальный идентификатор записи о тесте. Это поле служит первичным ключом и позволяет однозначно идентифицировать каждую запись в базе данных.

Users:

Тип: Целое число (int?).

Описание: Это свойство содержит идентификатор пользователя, который прошел тест. Оно может быть null, если запись не привязана к конкретному пользователю (например, в случае анонимного тестирования).

Tests:

Тип: Целое число (int?).

Описание: Это свойство содержит идентификатор теста, который был пройден. Оно также может быть null, если запись не привязана к конкретному тесту.

Points:

Тип: Целое число (int?).

Описание: Это свойство хранит количество баллов, набранных пользователем за прохождение теста. Оно может быть null, если результат теста еще не оценен или если тест не был завершен.

TestsNavigation:

Тип: Объект класса Test?.

Описание: Это навигационное свойство, которое устанавливает связь между записью о тесте и конкретным тестом. Оно позволяет получить доступ к дополнительной информации о тесте, если такая информация хранится в отдельной таблице Test. Это свойство может быть null, если тест не указан.

UsersNavigation:

Тип: Объект класса User ?.

Описание: Это навигационное свойство, которое устанавливает связь между записью о тесте и конкретным пользователем. Оно позволяет получить доступ к дополнительной информации о пользователе, если такая информация хранится в отдельной таблице User . Это свойство также может быть null, если пользователь не указан.

Взаимосвязи и использование

Связь с пользователями: Поле UsersNavigation позволяет легко получать информацию о пользователе, который прошел тест, что упрощает управление данными и анализ результатов.

Связь с тестами: Поле TestsNavigation позволяет получать информацию о тесте, который был пройден пользователем, что полезно для анализа и отчетности.

* 1. Разработка функционала страниц

Разработка моделей

Модель Answer

Назначение классов

Организация вопросов и ответов: Каждый класс представляет отдельный вопрос, что позволяет организовать вопросы и связанные с ними ответы в логически разделенные единицы. Это особенно полезно при разработке приложений для тестирования, опросов или викторин, где необходимо четко структурировать информацию.

Упрощение доступа к ответам: Свойство Answers в каждом классе возвращает массив строк, содержащий возможные ответы на соответствующий вопрос. Это упрощает доступ к ответам, позволяя разработчикам и пользователям легко получать список ответов, не обращаясь к каждому отдельному свойству (например, Answer1, Answer2, Answer3).

Функционал

Хранение данных: Каждый класс может содержать дополнительные свойства, такие как текст вопроса, правильный ответ и другие метаданные, которые могут быть полезны для логики приложения. Например, можно добавить свойства для хранения текста вопроса или индикатора правильного ответа.

Группировка логики: Эти классы могут быть частью более крупной системы, которая управляет вопросами и ответами. Например, они могут быть использованы в приложении для викторины, где пользователи выбирают ответы на вопросы, и система проверяет правильность ответов.

Расширяемость: Структура классов позволяет легко добавлять новые вопросы, просто создавая новые классы (например, Questions8, Questions9 и т.д.) или расширяя существующие классы, если это необходимо. Это делает систему гибкой и адаптируемой к изменениям.

Поддержка различных типов вопросов: В будущем можно расширить функционал, добавив поддержку различных типов вопросов (например, множественный выбор, открытые вопросы и т.д.), просто добавив дополнительные методы или свойства в классы.

Интерфейс взаимодействия: Эти классы могут быть использованы для создания интерфейса взаимодействия с пользователем, где каждый вопрос и соответствующие ответы отображаются на экране, позволяя пользователю взаимодействовать с приложением.

Класс Course представляет собой модель данных, используемую для хранения информации о курсе в приложении. Давайте подробно рассмотрим каждое из свойств этого класса и его назначение. Свойства класса

Id:

Тип: int

Описание: Уникальный идентификатор курса. Это свойство обычно используется для различения различных курсов в базе данных. Оно должно быть уникальным для каждого курса.

Name:

Тип: string?

Описание: Название курса. Это свойство может содержать название курса, например, "Основы программирования". Использование string? указывает на то, что это свойство может быть null, что позволяет не указывать название курса в некоторых случаях.

Description:

Тип: string?

Описание: Описание курса. Это свойство может содержать более подробную информацию о содержании курса, его целях и темах, которые будут рассмотрены. Также может быть null, если описание не предоставлено.

Photo:

Тип: string?

Описание: Путь к изображению курса. Это свойство может использоваться для хранения URL или относительного пути к изображению, которое будет представлять курс в интерфейсе пользователя. Как и предыдущие свойства, оно может быть null.

Uk:

Тип: int?

Описание: Дополнительное поле, которое может использоваться для различных целей, например, для хранения идентификатора пользователя, который создал курс, или идентификатора категории, к которой относится курс. Это свойство также может быть null.

Класс Offer представляет собой модель данных, используемую для хранения информации о предложениях в приложении. Давайте подробно рассмотрим каждое из свойств этого класса и его назначение. Id:

Тип: int

Описание: Уникальный идентификатор предложения. Это свойство служит для однозначного различения различных предложений в базе данных. Обычно это свойство является первичным ключом, что обеспечивает уникальность каждого предложения.

Offers:

Тип: string?

Описание: Это свойство хранит текст или описание предложения. Оно может содержать информацию о скидках, акциях или специальных предложениях, доступных пользователям. Использование string? указывает на то, что это свойство может быть null, что позволяет не указывать текст предложения в некоторых случаях.

Класс Questions1 представляет собой модель данных, используемую для хранения информации о вопросах в приложении, например, в контексте викторин или тестов. Давайте рассмотрим каждое из свойств этого класса и его назначение. Затем мы создадим однотипные классы Questions2, Questions3, Questions4, Questions5, Questions6, и Questions7. Свойства класса

Id:

Тип: int

Описание: Уникальный идентификатор вопроса, который помогает различать вопросы в базе данных.

Questiontext:

Тип: string?

Описание: Хранит текст вопроса, который будет отображаться пользователю.

Correctanswerindex:

Тип: int

Описание: Индекс правильного ответа (0, 1 или 2 для Answer1, Answer2, Answer3). Это свойство позволяет определить, какой из предложенных ответов является правильным.

Answer1, Answer2, Answer3:

Тип: string?

Описание: Эти свойства хранят текстовые варианты ответов на вопрос. Они могут быть null, если варианты не указаны.

Selectedanswerindex:

Тип: int?

Описание: Индекс ответа, выбранного пользователем. Это свойство может быть null, если пользователь еще не выбрал ответ

Класс Role представляет собой модель данных, которая используется для хранения информации о ролях пользователей в системе. Эта модель может быть частью системы управления доступом, где каждая роль определяет набор прав и привилегий, которые могут быть назначены пользователям. Рассмотрим каждое из свойств класса более подробно.

Описание таблицы Role

Id:

Тип: Целое число (integer).

Описание: Это уникальный идентификатор роли. Он служит для однозначного различения различных ролей в базе данных. Обычно это поле является первичным ключом таблицы, что обеспечивает уникальность каждой записи.

Role1:

Тип: Строка (string).

Описание: Это свойство содержит название роли. Например, это может быть "Администратор", "Пользователь", "Модератор" и т. д. Название роли определяет, какие права и привилегии будут связаны с данной ролью. Это поле не может быть пустым, что гарантирует, что каждая роль будет иметь корректное название.

Users:

Тип: Коллекция пользователей (ICollection<User>).

Описание: Это свойство представляет собой коллекцию пользователей, которые связаны с данной ролью. Каждый пользователь может иметь одну или несколько ролей, и это свойство позволяет легко получить доступ к списку всех пользователей, которые имеют определенную роль. Это свойство инициализируется как новый список, что обеспечивает возможность добавления пользователей без необходимости предварительной инициализации коллекции.

Взаимосвязи и использование

Связь с пользователями: Таблица Role имеет связь с таблицей User , где каждый пользователь может быть связан с одной или несколькими ролями. Это позволяет реализовать гибкую систему управления доступом, где права пользователей могут быть легко изменены путем изменения их ролей.

Применение: Роли могут использоваться для определения уровней доступа к различным функциям системы, ограничения возможностей пользователей и управления правами на выполнение определенных действий. Например, администраторы могут иметь доступ к управлению пользователями и настройкам системы, в то время как обычные пользователи могут иметь доступ только к базовым функциям.

Класс Test представляет собой модель данных, используемую для хранения информации о тестах или экзаменах в системе. Эта модель может быть частью образовательного приложения или платформы для тестирования, где пользователи могут проходить различные тесты. Рассмотрим каждое из свойств класса более подробно.

Описание таблицы Test

Id:

Тип: Целое число (integer).

Описание: Это уникальный идентификатор теста. Он служит для однозначного различения различных тестов в базе данных и обычно является первичным ключом таблицы, что обеспечивает уникальность каждой записи.

Name:

Тип: Строка (string?).

Описание: Это свойство содержит название теста. Название теста должно быть информативным и отражать его содержание, чтобы пользователи могли легко понять, о чем тест. Это поле может быть пустым (null), если название теста не указано.

Description:

Тип: Строка (string?).

Описание: Это свойство предоставляет подробное описание теста. Описание может включать информацию о цели теста, его содержании, количестве вопросов и других важных аспектах. Это поле также может быть пустым (null), если описание не указано.

Photo:

Тип: Строка (string?).

Описание: Это свойство может использоваться для хранения ссылки на изображение, связанное с тестом. Например, это может быть логотип теста или изображение, которое иллюстрирует его тему. Это поле может быть пустым (null), если изображение не предоставлено.

Points:

Тип: Целое число (int?).

Описание: Это свойство указывает количество баллов, которые можно получить за успешное прохождение теста. Это значение может быть использовано для оценки результатов пользователей. Это поле может быть пустым (null), если количество баллов не указано.

UsersTests:

Тип: Коллекция пользователей, связанных с тестами (ICollection<UsersTest>).

Описание: Это свойство представляет собой коллекцию записей, связывающих пользователей с тестами. Каждая запись в этой коллекции может содержать информацию о том, какой пользователь проходил тест, его результаты и другие связанные данные. Это свойство инициализируется как новый список, что позволяет добавлять записи без необходимости предварительной инициализации коллекции.

Взаимосвязи и использование

Связь с пользователями: Таблица Test имеет связь с таблицей UsersTest, что позволяет отслеживать, какие пользователи проходили определенные тесты и какие результаты они получили. Это создает возможность для анализа успеваемости и управления процессом тестирования.

Применение: Модель Test может использоваться в образовательных приложениях для создания, редактирования и управления тестами. Информация о тестах может быть использована для формирования отчетов о результатах пользователей, анализа сложности тестов и оптимизации учебного процесса.

Класс User  представляет собой модель данных для хранения информации о пользователях в системе. Эта модель может быть частью приложения, связанного с тестированием или образовательной платформой. Давайте рассмотрим каждое из свойств класса более подробно.

Описание класса User

Id:

Тип: Целое число (int).

Описание: Уникальный идентификатор пользователя. Это поле служит первичным ключом и позволяет однозначно идентифицировать каждого пользователя в базе данных.

Login:

Тип: Строка (string).

Описание: Это свойство содержит логин пользователя. Логин должен быть уникальным и использоваться для аутентификации пользователя в системе. Поле инициализируется значением null!, что указывает на то, что оно не может быть null.

Password:

Тип: Строка (string).

Описание: Это свойство хранит пароль пользователя. Как и логин, пароль должен быть защищенным и не должен храниться в открытом виде. Поле также инициализируется значением null!, что указывает на то, что оно не может быть null.

Role:

Тип: Целое число (int?).

Описание: Это свойство определяет роль пользователя в системе. Роль может быть представлена в виде целого числа, которое соответствует определенному уровню доступа или типу пользователя (например, администратор, преподаватель, студент и т.д.). Это поле может быть пустым (null), если роль не указана.

RoleNavigation:

Тип: Объект класса Role?.

Описание: Это навигационное свойство, которое устанавливает связь между пользователем и его ролью. Оно позволяет получить доступ к дополнительной информации о роли пользователя, если такая информация хранится в отдельной таблице Role. Это свойство также может быть null, если роль не указана.

UsersTests:

Тип: Коллекция пользователей, связанных с тестами (ICollection<UsersTest>).

Описание: Это свойство представляет собой коллекцию записей, связывающих пользователей с тестами, которые они проходили. Каждая запись в этой коллекции может содержать информацию о том, какой тест прошел пользователь и его результаты. Коллекция инициализируется как новый список, что позволяет добавлять записи без необходимости предварительной инициализации.

Взаимосвязи и использование

Связь с ролями: Поле RoleNavigation устанавливает связь между пользователем и его ролью, что позволяет легко управлять правами доступа и функционалом для различных типов пользователей.

Связь с тестами: Коллекция UsersTests позволяет отслеживать, какие тесты прошел пользователь и какие результаты он получил. Это создает возможность для анализа успеваемости пользователей и управления процессом тестирования.

Класс UsersTest представляет собой модель данных, которая хранит информацию о тестах, прошедших пользователями. Давайте рассмотрим каждое из свойств этого класса более подробно.

Описание класса UsersTest

Id:

Тип: Целое число (int).

Описание: Уникальный идентификатор записи о тесте. Это поле служит первичным ключом и позволяет однозначно идентифицировать каждую запись в базе данных.

Users:

Тип: Целое число (int?).

Описание: Это свойство содержит идентификатор пользователя, который прошел тест. Оно может быть null, если запись не привязана к конкретному пользователю (например, в случае анонимного тестирования).

Tests:

Тип: Целое число (int?).

Описание: Это свойство содержит идентификатор теста, который был пройден. Оно также может быть null, если запись не привязана к конкретному тесту.

Points:

Тип: Целое число (int?).

Описание: Это свойство хранит количество баллов, набранных пользователем за прохождение теста. Оно может быть null, если результат теста еще не оценен или если тест не был завершен.

TestsNavigation:

Тип: Объект класса Test?.

Описание: Это навигационное свойство, которое устанавливает связь между записью о тесте и конкретным тестом. Оно позволяет получить доступ к дополнительной информации о тесте, если такая информация хранится в отдельной таблице Test. Это свойство может быть null, если тест не указан.

UsersNavigation:

Тип: Объект класса User ?.

Описание: Это навигационное свойство, которое устанавливает связь между записью о тесте и конкретным пользователем. Оно позволяет получить доступ к дополнительной информации о пользователе, если такая информация хранится в отдельной таблице User . Это свойство также может быть null, если пользователь не указан.

Взаимосвязи и использование

Связь с пользователями: Поле UsersNavigation позволяет легко получать информацию о пользователе, который прошел тест, что упрощает управление данными и анализ результатов.

Связь с тестами: Поле TestsNavigation позволяет получать информацию о тесте, который был пройден пользователем, что полезно для анализа и отчетности.

* 1. Разработка конвертеров

Разработка конвертеров

Конвертер ImageConverter предназначен для преобразования значений, которые могут представлять имена изображений, в соответствующие объекты Bitmap. Это позволяет динамически загружать изображения в зависимости от данных, предоставляемых в интерфейсе, что может быть полезно для отображения различных изображений в зависимости от состояния приложения или данных.

Работа конвертера

Конвертер реализует два метода: Convert и ConvertBack. Рассмотрим каждый из них.

1. Метод Convert

Параметры:

value: Это значение, которое передается в конвертер. Обычно это может быть имя файла изображения или другой идентификатор.

targetType: Тип, в который нужно преобразовать значение (в данном случае, это должен быть тип, который может принимать объект Bitmap).

parameter: Дополнительный параметр, который может быть передан (в данном коде он не используется).

culture: Информация о культуре, которая может использоваться для локализации.

Логика работы:

Если value равно null, конвертер возвращает изображение по умолчанию (в данном случае, plug.png).

Если value не равно null, конвертер создает URI для изображения, основываясь на значении value, и загружает соответствующее изображение. Это достигается с помощью AssetLoader.Open, который открывает ресурс по указанному URI.

2. Метод ConvertBack

Этот метод предназначен для преобразования значения обратно в исходное состояние. Однако в данном случае он не реализован и выбрасывает NotImplementedException, что означает, что конвертер не поддерживает обратное преобразование. Это может быть приемлемо, если конвертер используется только для отображения изображений, и нет необходимости возвращать имя файла или другое исходное значение.

Использование конвертера

В этом примере ImageName — это свойство вашей модели, которое может содержать имя изображения. Конвертер ImageConverter преобразует это имя в объект Bitmap, который затем отображается в элементе Image.

* 1. Разработка макетов страниц

Подробное описание разработки страницы AddQ1 в приложении на AvaloniaUI

Страница AddQ1 представляет собой пользовательский контрол, предназначенный для добавления нового вопроса в систему. Она разработана с использованием паттерна MVVM (Model-View-ViewModel), что позволяет разделить логику приложения и пользовательский интерфейс. Ниже приведено подробное описание разработки этой страницы.

1. Структура страницы

Страница определена как пользовательский контрол, что позволяет использовать ее в других частях приложения. В ней заданы необходимые пространства имен, которые обеспечивают доступ к компонентам AvaloniaUI и привязке к ViewModel.

2. Визуальное оформление

На странице используется рамка для создания аккуратного оформления. Рамка имеет заданные параметры, такие как толщина, цвет, закругление углов и отступы, что придает элементам управления более привлекательный вид.

3. Верхняя панель с кнопкой "Назад"

В верхней части страницы размещена панель, содержащая кнопку "Назад". Эта кнопка связана с командой в ViewModel, позволяющей пользователю вернуться на предыдущую страницу. Она оформлена в соответствии с общей цветовой схемой приложения, что делает ее заметной и удобной для использования.

4. Основная часть с формой для ввода данных

Основная часть страницы содержит форму, в которой пользователи могут вводить данные для нового вопроса. В этой части предусмотрены следующие элементы:

Заголовок, который информирует пользователя о том, что он может создать новый вопрос.

Поле для ввода текста вопроса, связанное с моделью данных, что позволяет автоматически обновлять данные в ViewModel.

Выпадающий список для выбора индекса корректного ответа. Пользователь может выбрать один из трех вариантов, что упрощает процесс ввода данных и предотвращает ошибки.

5. Ввод ответов

Далее в форме предусмотрены текстовые поля для ввода трех возможных ответов на вопрос. Каждое поле связано с соответствующими свойствами в модели данных, что позволяет сохранять введенные ответы и использовать их в дальнейшем.

6. Кнопка "Сохранить"

Внизу страницы размещена кнопка "Сохранить", которая также связана с командой в ViewModel. Эта кнопка позволяет пользователю сохранить введенные данные о вопросе и ответах в системе. Она оформлена в соответствии с остальными элементами интерфейса, чтобы создать единый стиль.

Заключение

Страница AddQ1 разработана с акцентом на удобство использования и визуальную привлекательность. Использование MVVM позволяет эффективно управлять данными и логикой приложения, а четкая структура интерфейса способствует простоте и интуитивной понятности для пользователей.

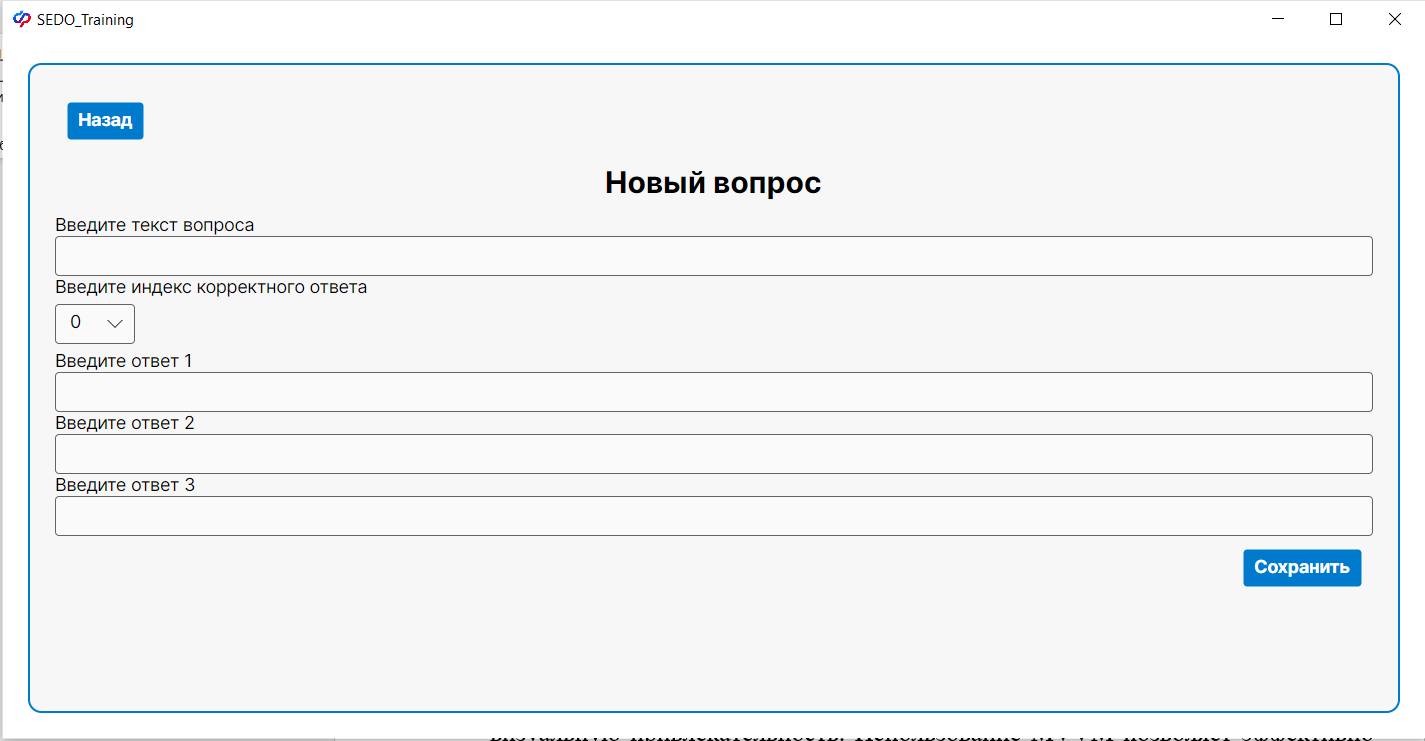


Рисунок 1 – страница добавления

Описание разработки страницы авторизации

Страница авторизации представляет собой пользовательский интерфейс, который позволяет пользователям вводить свои учетные данные для входа в систему. Основные элементы и их функции описаны ниже.

1. Общая структура

Страница оформлена в виде рамки, которая создает аккуратный и привлекательный вид. Рамка имеет заданные параметры, такие как толщина, цвет и закругление углов, что придает элементам управления более профессиональный вид.

2. Заголовок

В верхней части страницы расположен заголовок "Авторизация". Он выделяется за счет увеличенного размера шрифта и жирного начертания, что делает его заметным и информативным для пользователей.

3. Поля ввода

На странице предусмотрены два текстовых поля для ввода учетных данных:

Поле для логина: Пользователь может ввести свой логин. Поле имеет подсказку (watermark), которая исчезает при вводе текста, что помогает пользователю понять, что именно требуется ввести.

Поле для пароля: Пользователь вводит свой пароль в защищенном формате, где символы заменяются на звездочки. Также предусмотрена подсказка, которая помогает пользователю.

Оба поля имеют одинаковый стиль, включая высоту, ширину и параметры оформления, что создает единый визуальный стиль.

4. Кнопка "Войти"

Под полями ввода расположена кнопка "Войти", которая связана с командой в ViewModel. Эта кнопка позволяет пользователю отправить введенные данные для авторизации. Она выделена за счет цвета фона и жирного шрифта, что делает ее заметной и легкой для нажатия.

5. Ссылка на регистрацию

Ниже кнопки "Войти" размещен текстовый блок с вопросом "Нет аккаунта?" и ссылкой "Зарегистрироваться". Эта ссылка выделена цветом и жирным шрифтом, а также имеет курсор в виде руки при наведении, что указывает на интерактивность. При нажатии на ссылку происходит переход к процессу регистрации.

6. Компоновка элементов

Все элементы управления размещены в вертикальном и горизонтальном направлении с учетом удобства использования. Элементы выровнены по центру, что создает гармоничный и аккуратный интерфейс.

Заключение

Страница авторизации разработана с акцентом на удобство и простоту использования. Четкое оформление и логичная структура помогают пользователям легко вводить свои данные и переходить к следующему шагу, будь то вход в систему или регистрация нового аккаунта.

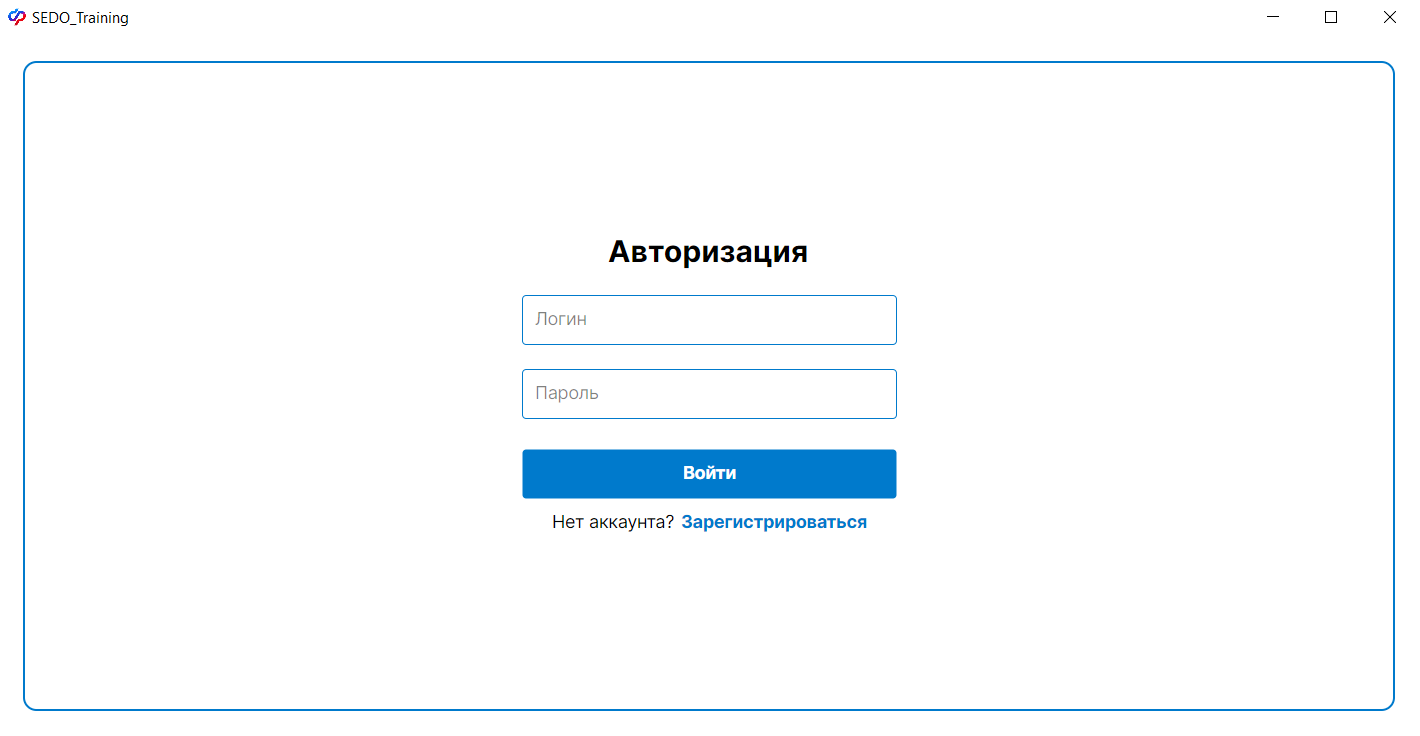


Рисунок 2 – страница авторизации

Описание страницы управления пользователями

Страница управления пользователями представляет собой интерфейс, который позволяет администраторам или пользователям с соответствующими правами просматривать, искать и редактировать информацию о пользователях в системе. Ниже приведено подробное описание основных элементов и функций этой страницы.

1. Общая структура

Страница оформлена в виде рамки, которая создает аккуратный и профессиональный вид. Рамка имеет заданные параметры, такие как толщина, цвет и закругление углов, что делает интерфейс более привлекательным.

2. Кнопка "Назад"

В верхней части страницы размещена кнопка "Назад", которая позволяет пользователю вернуться на предыдущую страницу. Эта кнопка оформлена в соответствии с общей цветовой схемой приложения и выделяется за счет жирного шрифта и яркого фона.

3. Заголовок страницы

Под кнопкой "Назад" расположен заголовок "Пользователи", который выделяется увеличенным размером шрифта и жирным начертанием. Это помогает пользователям быстро понять, на какой странице они находятся.

4. Поле поиска и фильтр по ролям

На странице предусмотрены два элемента для фильтрации пользователей:

Поле для поиска: Пользователь может ввести текст для поиска по списку пользователей. Поле имеет подсказку, которая исчезает при вводе, что помогает пользователю понять, что именно требуется ввести.

Комбо-бокс для фильтрации по ролям: Пользователь может выбрать роль из выпадающего списка, что позволяет отфильтровать пользователей по их ролям в системе. Это упрощает процесс поиска нужных пользователей.

5. Разделитель

Ниже полей поиска и фильтрации размещен горизонтальный разделитель, который визуально отделяет элементы управления от основного содержимого страницы, создавая более структурированный вид.

6. Список пользователей

Основная часть страницы содержит список пользователей, который представлен в виде прокручиваемого элемента. Каждый пользователь отображается в отдельном элементе, который включает:

Логин пользователя: Выделен жирным шрифтом и увеличенным размером, что делает его легко читаемым.

Роль пользователя: Отображается рядом с логином, также выделена для удобства восприятия.

Кнопка "Редактировать": Каждому пользователю соответствует кнопка, позволяющая перейти к редактированию его данных. Кнопка оформлена в соответствии с общим стилем страницы и содержит иконку для наглядности.

7. Компоновка элементов

Все элементы управления расположены в аккуратном порядке, что способствует удобству использования. Элементы выровнены по центру и организованы в вертикальном и горизонтальном направлениях, создавая гармоничный интерфейс.

Заключение

Страница управления пользователями разработана с акцентом на удобство и простоту использования. Четкое оформление и логическая структура помогают пользователям легко находить и редактировать нужную информацию о пользователях, что делает процесс управления более эффективным.

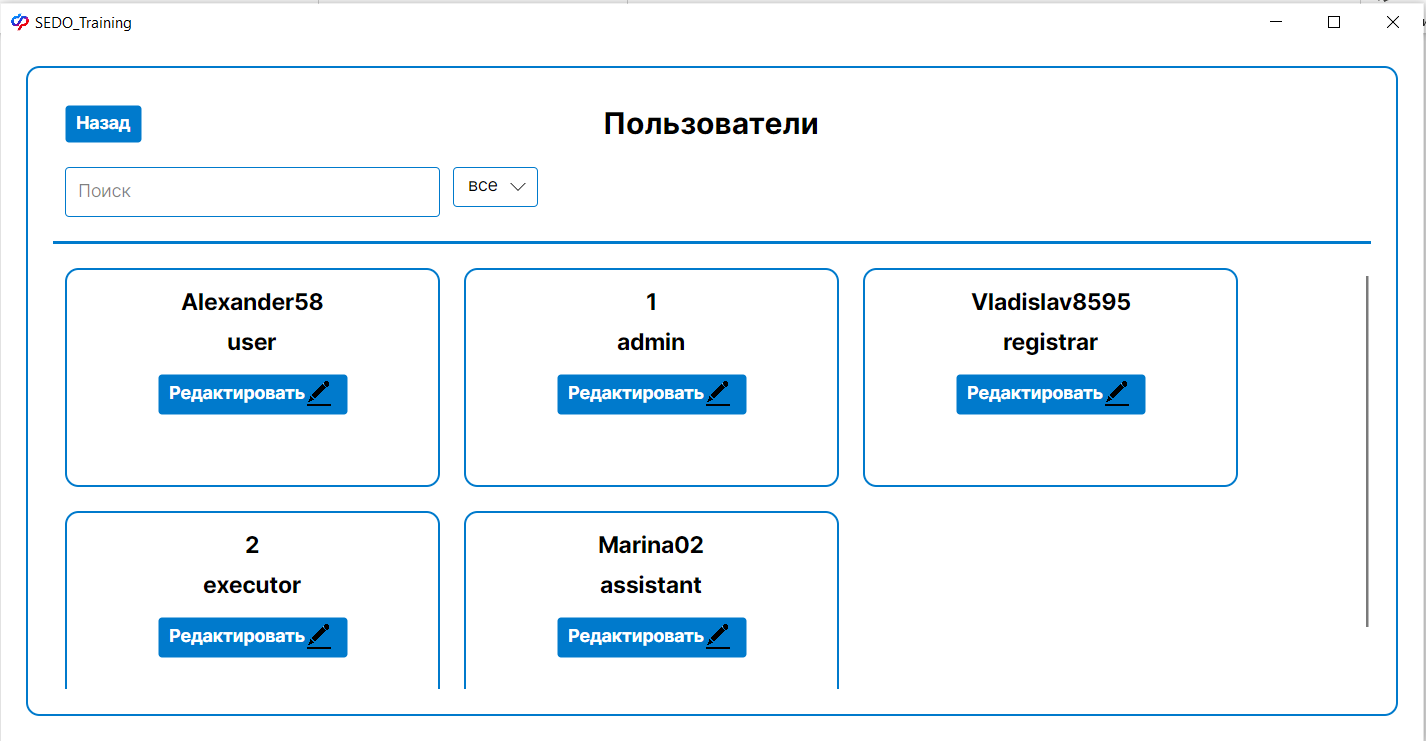


Рисунок 3 – страница пользователей

Описание страницы документации

Страница документации предназначена для управления документами и их атрибутами. Она обеспечивает удобный интерфейс для работы с различными полями документа, позволяя пользователям добавлять, редактировать и просматривать информацию о документах.

1. Общая структура

Страница оформлена в виде рамки с заданными параметрами, такими как толщина границы, цвет, закругление углов и фон. Это создает аккуратный и профессиональный вид.

2. Кнопки навигации

В верхней части страницы находятся две кнопки:

На главную: Позволяет пользователю вернуться на главную страницу.

Тест по курсу: Предоставляет доступ к тестам по курсу.

Обе кнопки выделены ярким фоном и жирным шрифтом, что делает их заметными.

3. Заголовок и описание полей документа

На странице есть заголовок "Поля документа", который выделен увеличенным размером шрифта и жирным начертанием. Под ним размещены различные текстовые блоки, объясняющие, как заполнять поля документа, включая:

Адресаты: Информация о том, как добавлять адресатов исходящего документа, включая юридических лиц.

Инструкции по заполнению: Пользователи получают рекомендации по вводу первых букв названия организации для выбора из списка или добавления новой записи.

4. Визуальные элементы

На странице представлены изображения, которые помогают визуализировать процесс работы с документами. Эти изображения оформлены в аккуратные рамки и располагаются в горизонтальных и вертикальных стеклянных панелях.

5. Поля для ввода

Страница содержит множество текстовых блоков, объясняющих различные поля документа, такие как:

Поле для ввода адреса электронной почты: Информация о том, как это поле отображается и какие требования к нему предъявляются.

Поле "Гриф": Указывает степень секретности документа и как его можно изменить.

Поле "Исполнитель": Указывает на сотрудника, подготавливающего документ, и дает инструкции по его назначению.

6. Дополнительные поля и функции

На странице также описаны дополнительные поля, которые могут отображаться в зависимости от настроек администратора. Например:

Поле "Содержание": Обязательное для регистрации документа, его можно заполнить вручную или выбрать из шаблона.

Поле "Срочность": Определяет категорию срочности документа.

7. Добавление новых корреспондентов

Пользователи могут добавлять новых корреспондентов (юридических и физических лиц), отсутствующих в справочнике, через соответствующие поля. Информация о процессе добавления и редактирования корреспондентов представлена в текстовых блоках и сопровождается иллюстрациями.

8. Номера документов и экземпляров

Страница также содержит разделы, посвященные номеру бланка и номеру документа, с объяснением, как их можно указать и изменять.

9. Подразделение регистратора и получатели

Информация о заполнении поля "Подразделение регистратора" и добавлении получателей для входящих документов также представлена на странице. Пользователи могут выбирать сотрудников из справочника или списка рассылки.

10. Заключение

Страница документации разработана с акцентом на удобство использования и доступность информации. Четкая структура, визуальные элементы и подробные инструкции помогают пользователям эффективно работать с документами и их атрибутами.

Общая структура

Border: Основной контейнер с рамкой, который задает стиль и отступы для всего содержимого.

Grid: Используется для организации элементов в строках и столбцах.

Верхняя панель

StackPanel (Горизонтальная панель): Содержит две кнопки:

Кнопка "На главную": Позволяет пользователю вернуться на главную страницу.

Кнопка "Тест по курсу": Открывает тест по курсу.

Основное содержимое

ScrollViewer: Позволяет прокручивать содержимое, если оно превышает размеры экрана.

StackPanel (Вертикальная панель): Содержит текстовые блоки и изображения, организованные в вертикальном порядке.

Текстовые блоки

Каждый текстовый блок содержит информацию о различных операциях с заданиями:

Создание задания: Описывает способы создания задания.

Открытие задания: Указывает, как открыть задание в интерфейсе.

Просмотр задания: Объясняет, как просмотреть информацию о задании.

Редактирование задания: Информирует о блокировках и доступных действиях при редактировании.

Управление заданием: Описывает смену состояния задания и как взять его в работу.

Исполнение задания: Информация о том, как исполнить задание и выполнить групповые операции.

Завершение исполнения: Объясняет, как завершить исполнение и вернуть задание на доработку.

Отклонение задания: Указывает, как отклонить задание и ввести причину.

Создание инициативного поручения: Описывает процесс создания поручения и его утверждения.

Согласование файлов: Объясняет, как согласовать файлы и что делать при замечаниях.

Визирование: Информация о визировании и действиях при положительном или отрицательном решении.

Утверждение: Описание процесса утверждения задания с подписями и возможными замечаниями.

Подписание документов: Указывает, как подписывать документы в системе.

Делегирование задания: Описание процесса делегирования и исполнения задания другими исполнителями.

Контроль исполнения поручений: Информация о контроле выполнения заданий.

Передача задания: Описание процесса передачи задания другому исполнителю.

Просмотр истории задания: Как просмотреть историю операций с заданием.

Просмотр отчетов: Инструкция по просмотру отчетов из задания.

Печать задания: Описание процесса печати задания.

Отправка по электронной почте: Информация о том, как отправить документ по электронной почте.

Групповые операции: Описание групповых операций с заданиями.

Изображения

Каждый раздел содержит изображения с подписями, которые иллюстрируют соответствующие действия или интерфейсные элементы, такие как кнопки и окна.

ScrollViewer: Позволяет пользователю прокручивать содержимое, если оно превышает размеры экрана. Это особенно полезно для длинных текстов и изображений.

StackPanel: Используется для вертикального расположения элементов, что упрощает восприятие информации.

Заголовки и текстовые блоки

Операции с резолюцией: Заголовок раздела, обозначающий, что далее будет информация о работе с резолюциями.

Создание резолюций из задания (групповая операция): Заголовок для описания процесса создания резолюций. Следующие блоки текста содержат подробные инструкции о том, как это сделать, включая важные моменты, такие как порядок добавления исполнителей и условия автозаполнения.

Отправить на утверждение (групповая операция): Информация о том, как отправить резолюции на утверждение, включая шаги, которые необходимо выполнить, и возможные ошибки, которые могут возникнуть.

Утвердить резолюцию (групповая операция): Описание процесса утверждения резолюций, включая инструкции по выбору заданий и возможные ошибки при выполнении операции.

Изображения

Изображения резолюций: Три изображения, которые, вероятно, иллюстрируют интерфейс или действия, связанные с созданием и утверждением резолюций. Каждое изображение заключено в рамку с заданными параметрами оформления.

Grid: Используется для организации элементов в строках и столбцах, где определены две строки: одна для кнопок и другая для содержимого.

StackPanel: Используется для вертикального или горизонтального расположения элементов, что упрощает восприятие информации.

ScrollViewer: Позволяет прокручивать содержимое, если оно превышает размеры экрана.

Заголовки и текстовые блоки

Поиск: Заголовок раздела, обозначающий, что далее будет информация о функциях поиска.

Поиск документов по штрихкоду: Описание процесса поиска документов с использованием штрих-кода, включая инструкции по использованию ручного сканера и сохранению документов.

Быстрый поиск: Объясняет, как выполнять поиск документов и заданий по введенным словам, и описывает поведение системы при превышении лимита найденных карточек.

Автоподбор: Описание функциональности автоподбора значений в полях карточек, включая условия для его использования.

Расширенный поиск: Описание возможностей расширенного поиска, включая варианты поиска документов, заданий и реестров.

Полнотекстовый поиск: Объясняет, как использовать полнотекстовый поиск для нахождения документов и заданий по введенным словам.

Изображения

Код содержит множество изображений, которые иллюстрируют различные аспекты поиска. Каждое изображение заключено в рамку с заданными параметрами оформления и сопровождается текстовым описанием.

Основные элементы

Border: Обрамляет содержимое, задавая стиль с границей, закругленными углами и фоновым цветом. Это создает четкую визуальную границу для содержимого.

Grid: Используется для организации элементов в строках и столбцах. Здесь определены две строки: первая для кнопок, а вторая для содержимого, связанного с отчетами.

StackPanel: Используется для вертикального или горизонтального расположения элементов, что упрощает восприятие информации.

ScrollViewer: Позволяет прокручивать содержимое, если оно превышает размеры экрана.

Заголовки и текстовые блоки

Отчёты: Заголовок раздела, обозначающий, что далее будет информация о доступных отчетах.

Просмотр списка доступных отчетов: Описание процесса просмотра отчетов с указанием действий, которые нужно выполнить для доступа к отчетам.

Карточка расширенного отчета Web-клиент: Инструкция по формированию выбранного отчета с пошаговыми действиями.

Изображения

В коде представлены изображения, которые иллюстрируют различные аспекты работы с отчетами:

Просмотр списка доступных отчетов: Изображение, показывающее интерфейс с отчетами.

Карточка расширенного отчета: Изображение, демонстрирующее карточку отчета.

Пример сформированного отчета: Изображение, показывающее, как выглядит сформированный отчет.

Выгрузка отчетов

Выгрузка отчета в Excel и Word: Инструкция по выгрузке отчетов с пошаговыми действиями и изображением кнопки для выгрузки.

Открытие карточки расширенного отчета

Открытие карточки расширенного отчета из браузера по ссылке: Информация о том, как сформировать отчет через браузер, с указанием структуры URL и описанием поведения системы в зависимости от настроек.

Общая структура интерфейса

Интерфейс предназначен для модуля, который позволяет пользователям добавлять скан-образы результатов потокового сканирования. Он включает в себя элементы навигации, текстовые инструкции и изображения, которые помогают пользователю понять, как использовать модуль.

Элементы интерфейса

Рамка (Border):

Интерфейс начинается с рамки, которая обрамляет все содержимое. Она имеет синюю рамку, светло-серый фон и закругленные углы. Это придает интерфейсу аккуратный и современный вид.

Сетка (Grid):

Внутри рамки используется сетка, разделенная на две строки. Первая строка автоматически подстраивается под содержимое, а вторая занимает оставшееся пространство. Это позволяет удобно размещать элементы интерфейса.

Кнопки навигации:

В первой строке размещены две кнопки. Одна из них ведет на главную страницу, а другая — на тест по курсу. Кнопки имеют яркий синий фон, белый текст и выделены жирным шрифтом, что делает их заметными и удобными для использования.

Прокручиваемая область (ScrollViewer):

Во второй строке находится область для прокрутки, которая позволяет пользователю видеть содержимое, если оно превышает доступное пространство. Это особенно полезно для длинных текстов или инструкций.

Текстовые инструкции:

Внутри прокручиваемой области размещены текстовые блоки с подробными инструкциями по добавлению скан-образов. Тексты оформлены с использованием различных размеров шрифта и жирного начертания для выделения важных пунктов. Инструкции объясняют, как проверить версию сервиса, открыть модуль, настроить загрузку файлов и завершить процесс.

Изображения:  
Также в интерфейсе представлены изображения, которые иллюстрируют различные этапы работы с модулем. Каждое изображение обрамлено рамкой и сопровождается текстовым описанием, что помогает пользователю визуально понять, как выглядит интерфейс и какие действия необходимо выполнить.

Общая структура интерфейса

Интерфейс оформлен в виде рамки, внутри которой размещены различные элементы, обеспечивающие пользователю удобство работы с функцией наложения штампов.

Элементы интерфейса

Рамка (Border):

Вся структура интерфейса обрамлена рамкой с синей границей и светло-серым фоном. Она имеет закругленные углы и внутренние отступы, что придает интерфейсу современный и аккуратный вид.

Сетка (Grid):

Внутри рамки используется сетка, состоящая из двух строк. Первая строка автоматически подстраивается под содержимое, а вторая занимает оставшееся пространство. Это позволяет удобно размещать элементы интерфейса.

Кнопки навигации:

В верхней части интерфейса находятся две кнопки:

Первая кнопка ведет на главную страницу приложения.

Вторая кнопка предназначена для перехода к тесту по курсу. Обе кнопки имеют яркий синий фон и белый текст, что делает их заметными и удобными для использования.

Прокручиваемая область (ScrollViewer):

Во второй строке размещена прокручиваемая область, которая позволяет пользователю просматривать длинные тексты и инструкции, если они превышают доступное пространство.

Текстовые инструкции:

Внутри прокручиваемой области размещены текстовые блоки с пошаговыми инструкциями по работе с мастером наложения штампов. Инструкции содержат подробные описания действий, которые необходимо выполнить, чтобы наложить штамп на документ. Тексты выделены различными размерами шрифта и жирным начертанием для акцента на ключевых моментах.

Изображения:  
В интерфейсе представлены изображения, которые иллюстрируют процесс работы с мастером наложения штампов. Каждое изображение обрамлено рамкой и сопровождается текстовым описанием, что помогает пользователю визуально понять, как выглядит интерфейс и какие действия необходимо выполнить.

Дополнительные инструкции:

Интерфейс включает разделы, описывающие дополнительные функции, такие как наложение нескольких штампов, автоматическое нанесение штампа на последнюю страницу документа и удаление наложенных штампов. Каждый из этих разделов содержит текстовые инструкции и изображения, что делает информацию более доступной и понятной.

Страница изменения пользователя

Общая структура интерфейса

Интерфейс оформлен в виде рамки, внутри которой размещены элементы для редактирования данных пользователя. Он включает в себя заголовок, поля ввода, выпадающие списки и кнопки для навигации и сохранения изменений.

Элементы интерфейса

Рамка (Border):

Вся структура обрамлена рамкой с синей границей и светло-серым фоном. У рамки закругленные углы и внутренние отступы, что придает интерфейсу современный вид.

Сетка (Grid):

Внутри рамки используется сетка, состоящая из трех строк:

Первая строка содержит кнопку для возврата на предыдущий экран.

Вторая строка занимает основное пространство для ввода данных.

Третья строка включает кнопку для сохранения изменений.

Кнопка "Назад":

В верхней части интерфейса расположена кнопка, позволяющая пользователю вернуться на предыдущий экран. Она выделена синим фоном и белым текстом, что делает её заметной.

Прокручиваемая область (ScrollViewer):

Во второй строке находится прокручиваемая область, которая позволяет пользователю просматривать и редактировать информацию о пользователе, если она превышает доступное пространство.

Заголовок:

Внутри прокручиваемой области размещен заголовок "Редактирование пользователя", который выделен крупным шрифтом и жирным начертанием.

Поля для ввода данных:

Login: Отображается текстовое поле с логином пользователя. Логин отображается в жирном шрифте.

Password: Аналогично, отображается текстовое поле с паролем пользователя, также выделенное жирным шрифтом.

Выбор роли: Выпадающий список (ComboBox) позволяет выбрать роль пользователя. Этот элемент связывается с источником данных, содержащим доступные роли, и отображает выбранную роль.

Список тестов пользователя:

Заголовок "Тесты текущего пользователя" выделен жирным шрифтом и крупным размером.

ItemsControl: Используется для отображения списка тестов текущего пользователя. Каждый тест представлен в виде рамки с информацией о названии теста и набранных баллах. Это позволяет пользователю видеть свои результаты в компактном и удобном формате.

Кнопка "Сохранить":В нижней части интерфейса находится кнопка для сохранения изменений. Она также выделена синим фоном и белым текстом, что делает её заметной и доступной для пользователя.

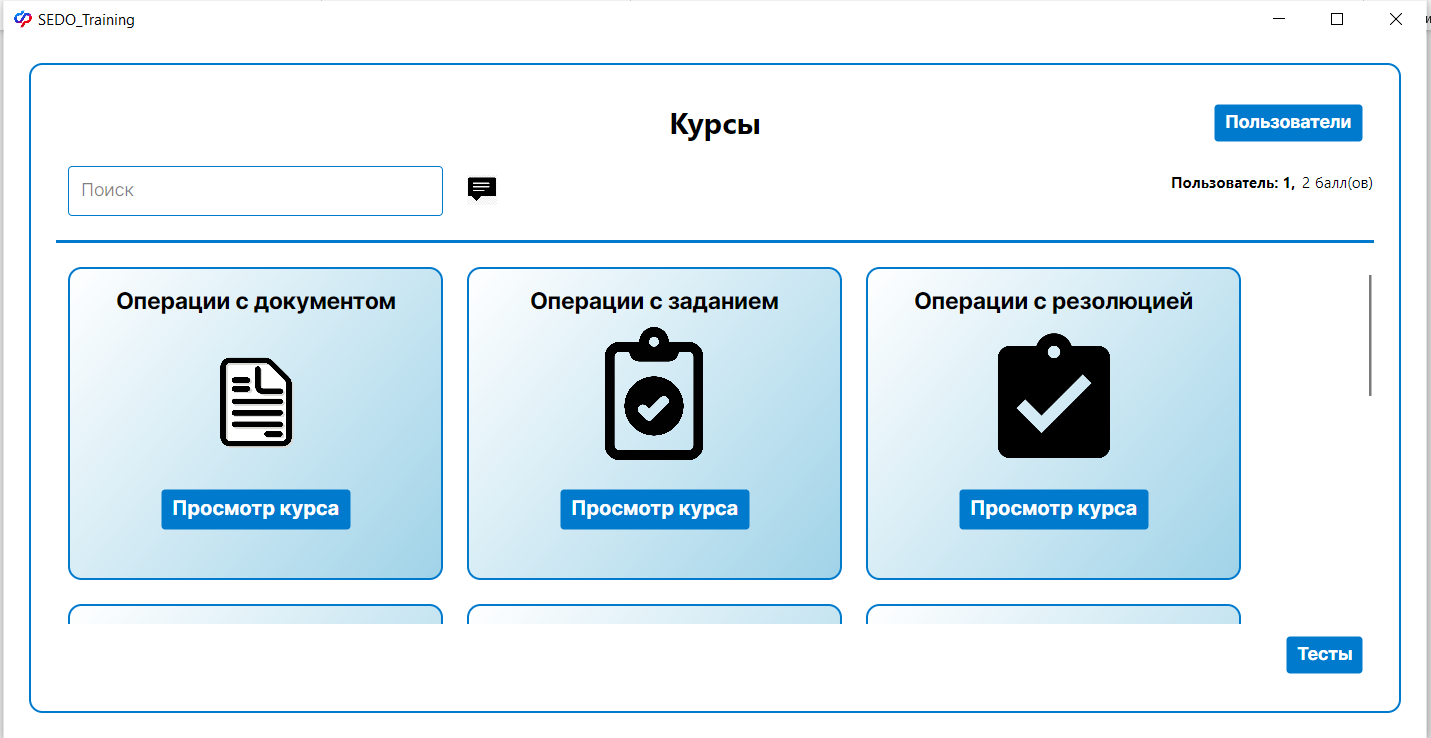


Рисунок 4 – страница курсов

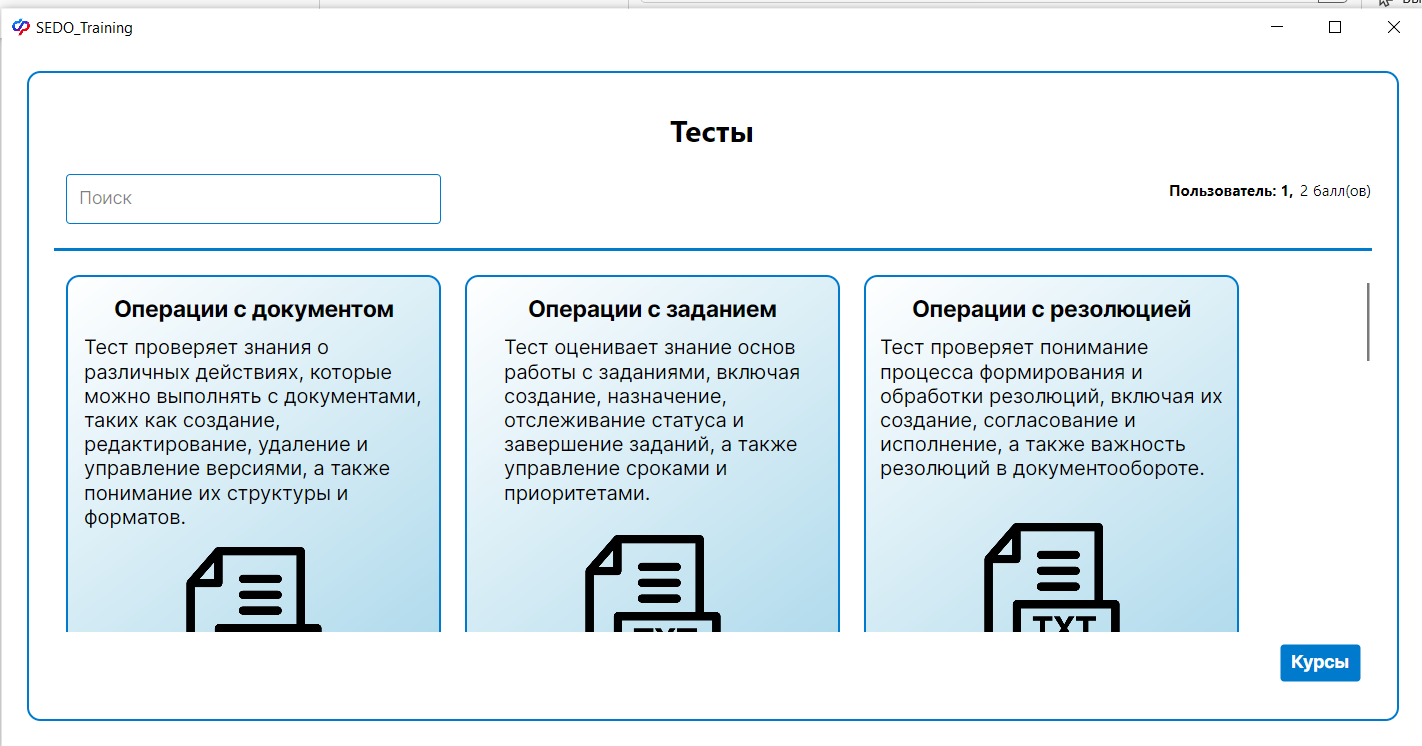


Рисунок 5 – страница тестов

Страница Input

Общая структура интерфейса

Интерфейс оформлен в виде рамки, внутри которой размещены элементы для ввода предложений. Он включает в себя заголовок, текстовое поле для ввода и кнопки для закрытия окна и отправки предложений.

Элементы интерфейса

Сетка (Grid):

Основной контейнер, содержащий все элементы управления. Он позволяет гибко организовать размещение элементов в вертикальном направлении.

Рамка (Border):

Вся структура обрамлена рамкой с синей границей и закругленными углами. Внутри рамки используется градиентный фон, который плавно переходит от белого к светло-голубому. Это создает приятный визуальный эффект.

Кнопка закрытия (Button):

В верхнем правом углу расположена кнопка с символом "✖", которая служит для закрытия окна. Она имеет прозрачный фон и границу, что делает её менее заметной, но при этом функциональной. Команда, связанная с этой кнопкой, возвращает пользователя в главное меню.

Заголовок:

Вторая строка содержит заголовок "Ваши предложения", который выделен крупным шрифтом и жирным начертанием. Это помогает привлечь внимание пользователя к основной теме окна.

Инструктивный текст:

Под заголовком размещен текст, объясняющий пользователю, что он может поделиться своими темами для создания новых курсов и тестов. Этот текст также выделен хорошим размером шрифта и обернут, чтобы быть более читаемым.

Поле для ввода предложений (TextBox):

В третьей строке находится текстовое поле, в которое пользователь может вводить свои предложения. Поле имеет белый фон и растягивается по ширине, что позволяет удобно вводить текст.

Кнопка "Отправить":

В последней строке размещена кнопка "Отправить", которая позволяет пользователю отправить свои предложения. Кнопка выделена синим фоном и белым текстом, что делает её заметной и доступной для пользователя. Команда, связанная с этой кнопкой, инициирует процесс отправки предложения.

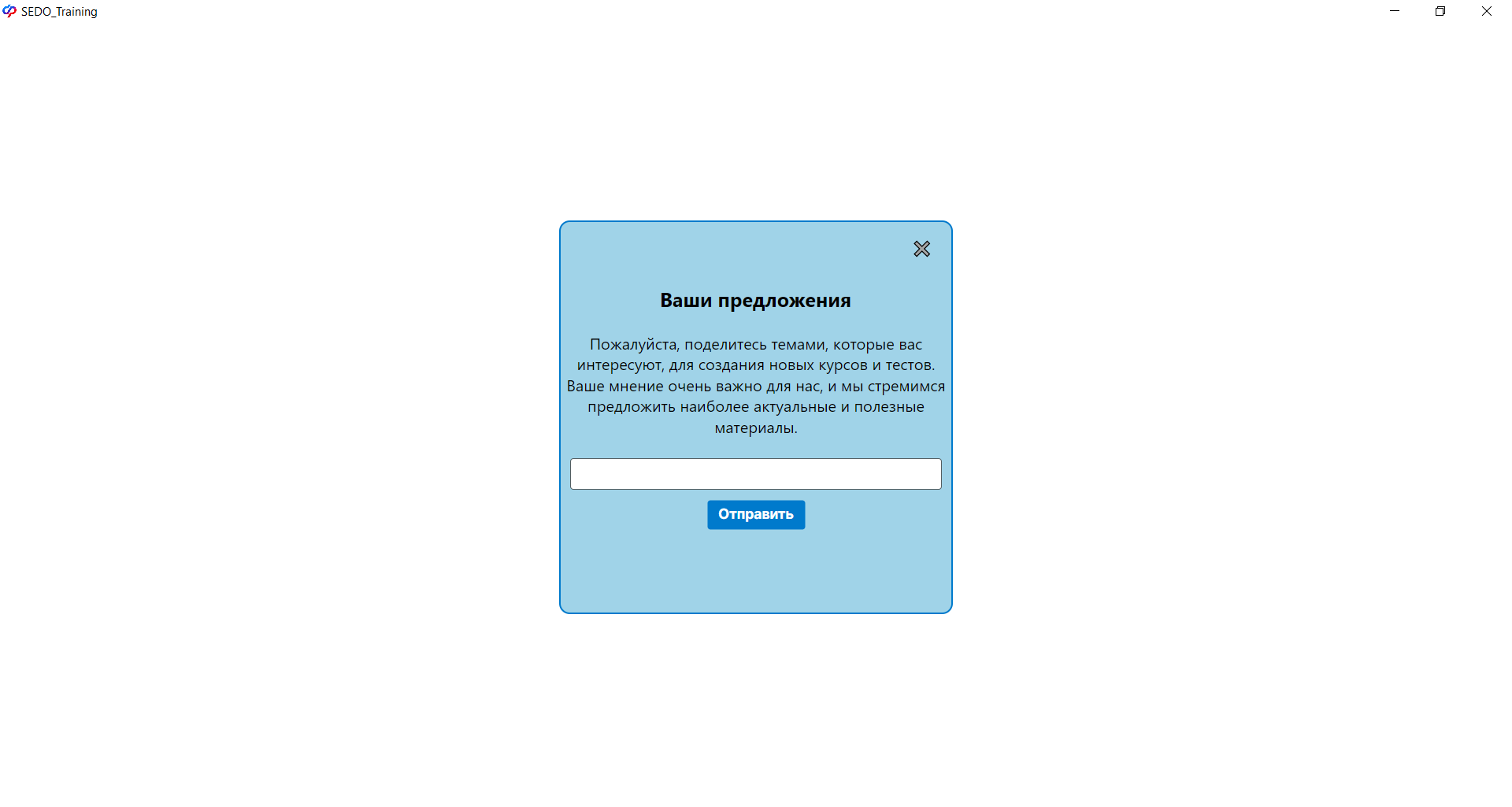


Рисунок 6 – страница ввода предложений

Страница меню

Общая структура интерфейса

Интерфейс организован в виде контейнера Border, который содержит Grid, позволяющий структурировать элементы в строках и столбцах. Это создает удобное и интуитивно понятное пространство для взаимодействия с пользователем.

Элементы интерфейса

Ресурсы (User Control.Resources):

Объявляется конвертер изображений (ImageConverter), который будет использоваться для преобразования данных изображения.

Рамка (Border):

Основной контейнер с синей границей, закругленными углами и внутренними отступами. Это создает аккуратный и современный вид.

Сетка (Grid):

Содержит строки и столбцы для организации элементов управления. Сетка имеет пять строк, что позволяет удобно размещать элементы.

Кнопка "Пользователи":

Кнопка, которая, при нажатии, выполняет команду ToClients. Она расположена в верхнем правом углу и имеет стиль, выделяющий её на фоне.

Заголовок "Курсы":

Текстовый блок, который служит заголовком для интерфейса. Он выделен большим шрифтом и жирным начертанием, что делает его заметным.

Информация о пользователе:

В первом StackPanel отображается информация о текущем пользователе и его баллах. Используются текстовые блоки для отображения имени пользователя и общего количества баллов.

Поле для поиска:

Второй StackPanel содержит текстовое поле для поиска курсов и кнопку с изображением (иконка комментария) для ввода. Поле имеет водяной знак "Поиск", что помогает пользователю понять его назначение.

Разделитель:

Горизонтальная линия, отделяющая элементы управления от списка курсов, создается с помощью второго Border.

Список курсов (ScrollViewer и ItemsControl):

ScrollViewer позволяет прокручивать список курсов. Внутри него используется ItemsControl, который отображает курсы с помощью DataTemplate.

Каждый курс представлен в виде рамки с градиентным фоном, содержащей название курса, изображение и кнопку "Просмотр курса".

Кнопка "Тесты":

Внизу интерфейса расположена кнопка для перехода к тестам, которая также имеет стиль, соответствующий другим кнопкам.

Страница регистрации

Общая структура интерфейса

Интерфейс организован в контейнере Border, который создает рамку для содержимого, обеспечивая визуальное отделение от фона. Внутри Border используется Grid для размещения различных элементов управления.

Элементы интерфейса

Рамка (Border):

Основной контейнер с синей границей, закругленными углами и внутренними отступами. Это создает аккуратный и современный вид для формы регистрации.

Сетка (Grid):

Сетка состоит из строк, каждая из которых содержит отдельный элемент управления. В данном случае используются пять строк, что позволяет удобно размещать заголовок и элементы ввода.

Заголовок "Регистрация":

Текстовый блок, который служит заголовком для формы регистрации. Он выделен большим шрифтом и жирным начертанием, что делает его заметным и привлекает внимание пользователя.

Текстовое поле для логина:

Первое текстовое поле с водяным знаком "Логин". Оно привязано к свойству Login в модели данных с использованием двусторонней привязки. Поле имеет стиль, соответствующий остальным элементам интерфейса.

Текстовое поле для пароля:

Второе текстовое поле для ввода пароля, с водяным знаком "Пароль". Оно также привязано к свойству Password и использует символ \* для сокрытия вводимого текста. Это обеспечивает безопасность при вводе пароля.

Кнопка "Регистрация":

Кнопка, которая выполняет команду RegisterCommand при нажатии. Она выделена синим фоном и белым текстом, что делает её заметной. Кнопка имеет фиксированную высоту и ширину, а также центрирует содержимое.

Информация о существующем аккаунте:

Внутри второго StackPanel находится текстовый блок, который предлагает пользователю перейти к форме входа, если у него уже есть аккаунт. Этот блок включает два текстовых элемента: один с обычным текстом "Есть аккаунт?" и второй с текстом "Войти", который выделен синим цветом и реагирует на нажатие (обработчик события PointerPressed).

Страница теста

Общая структура интерфейса

Интерфейс организован в Grid, который делится на две строки: первая для навигационных кнопок, а вторая для содержимого теста. Это позволяет удобно размещать элементы и делает интерфейс более структурированным.

Элементы интерфейса

Сетка (Grid):

Сетка состоит из двух строк: первая с высотой Auto для навигационных кнопок, вторая с высотой \*, что позволяет содержимому занимать оставшееся пространство.

Навигационная панель (StackPanel):

В первой строке находится StackPanel, который располагает кнопки горизонтально.

Кнопка "На главную": Команда ToMain возвращает пользователя на главную страницу.

Кнопка "Вернуться к курсу": Команда ToCourse1 возвращает к предыдущему курсу.

Кнопка "Добавить": Команда ToPageAddQ1 позволяет добавлять новые вопросы. Эта кнопка видима только при выполнении условия IsButtonVisible, что позволяет управлять её доступностью.

Прокручиваемая область (ScrollViewer):

Вторая строка содержит ScrollViewer, который позволяет прокручивать содержимое теста, если оно превышает доступное пространство.

Заголовок теста:

Текстовый блок с заголовком "Тест по теме «Поля документа»", выделенный крупным шрифтом и жирным начертанием. Это помогает пользователю понять, о чем тест.

Разделительная линия:

Горизонтальный Border, который служит визуальным разделителем между заголовком и вопросами теста.

Список вопросов (ItemsControl):

ItemsControl используется для отображения списка вопросов, привязанных к Questions1List в модели данных.

Шаблон элемента (DataTemplate): Каждый вопрос отображается в виде рамки с градиентным фоном, содержащей:

Текст вопроса: Отображается в текстовом блоке с возможностью переноса текста.

ComboBox для ответов: Позволяет пользователю выбрать ответ из списка, привязанного к свойству Answers.

Кнопка "Удалить": Удаляет вопрос, привязана к команде Delete, передающей идентификатор вопроса.

Кнопка "Редактировать": Позволяет редактировать вопрос, привязана к команде Update, также передающей идентификатор вопроса.

Кнопка "Проверить":

Внизу списка вопросов находится кнопка для проверки ответов, привязанная к команде CheckAnswersCommand. Она выделена синим фоном и белым текстом, что делает её заметной.

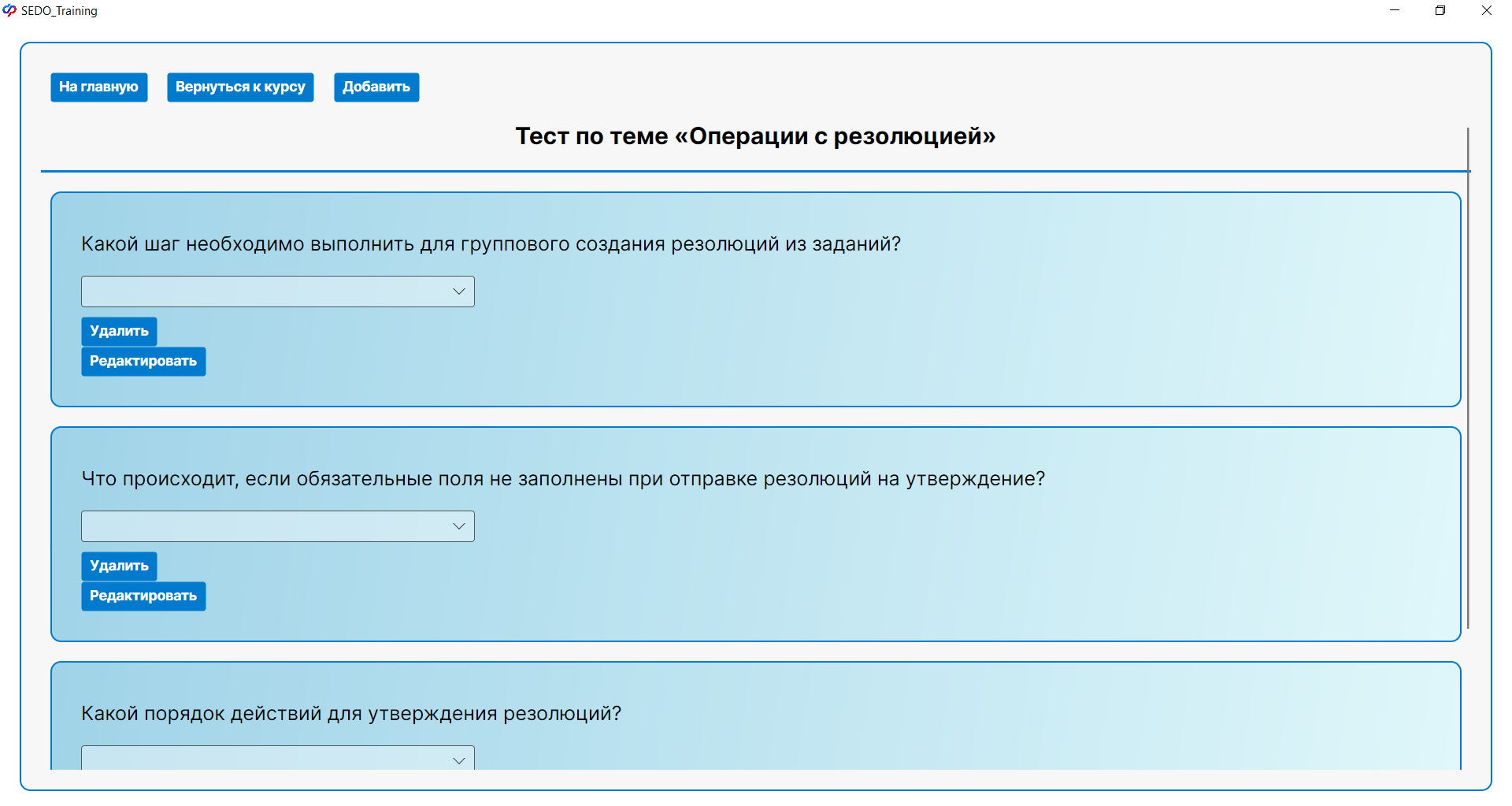


Рисунок 7 -страница тест1

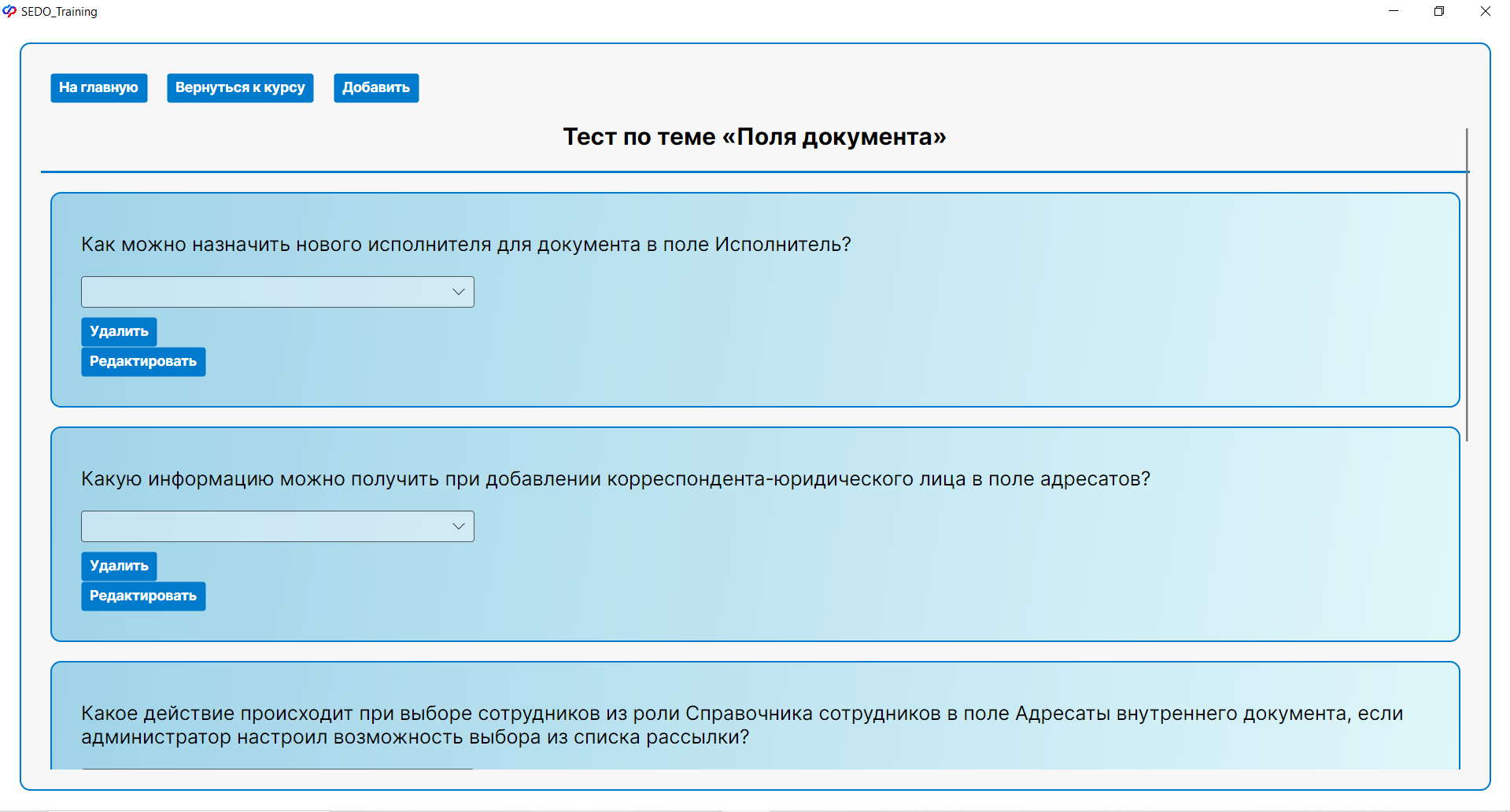


Рисунок 8 -страница тест2

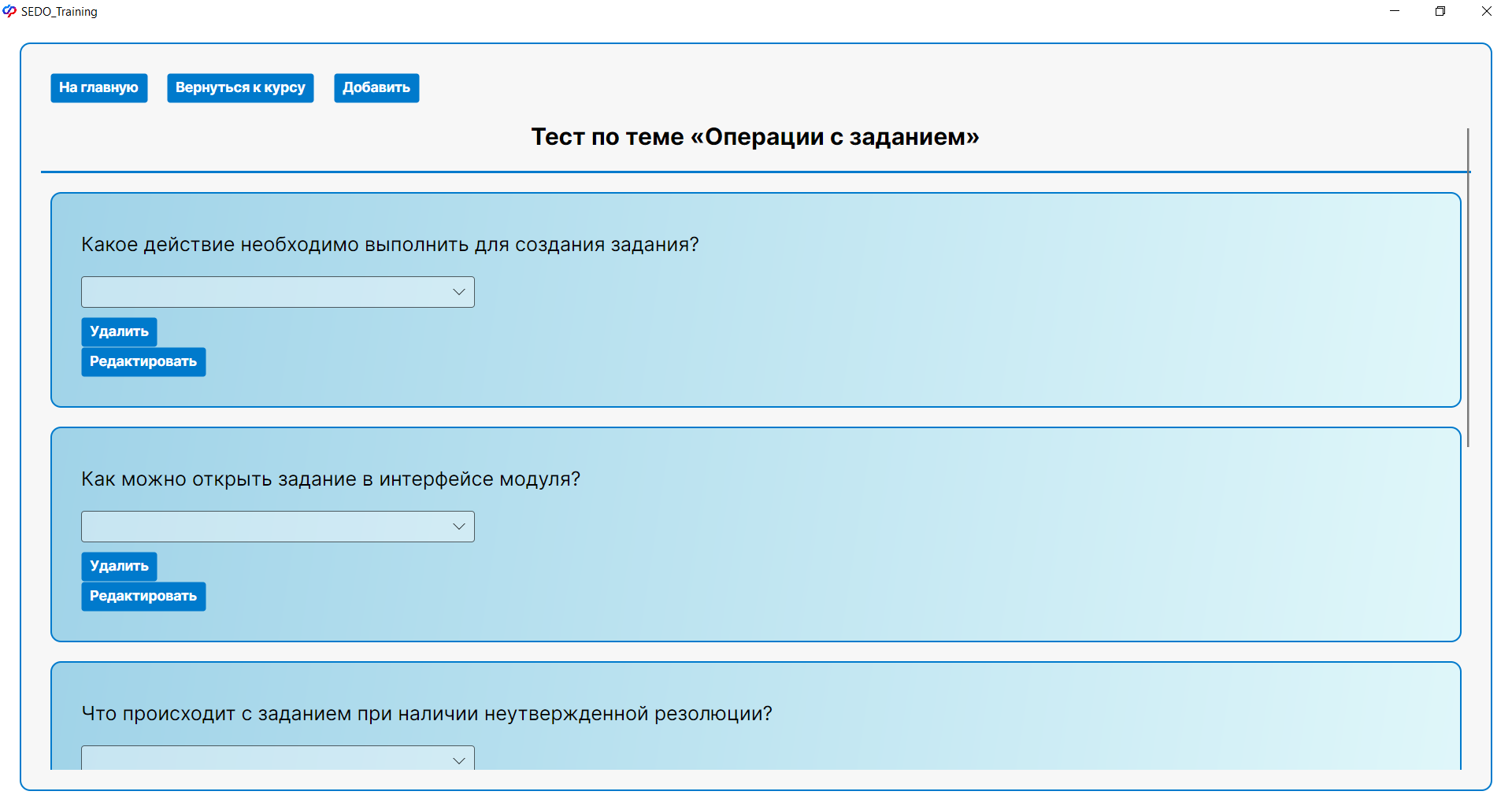


Рисунок 9 -страница тест3

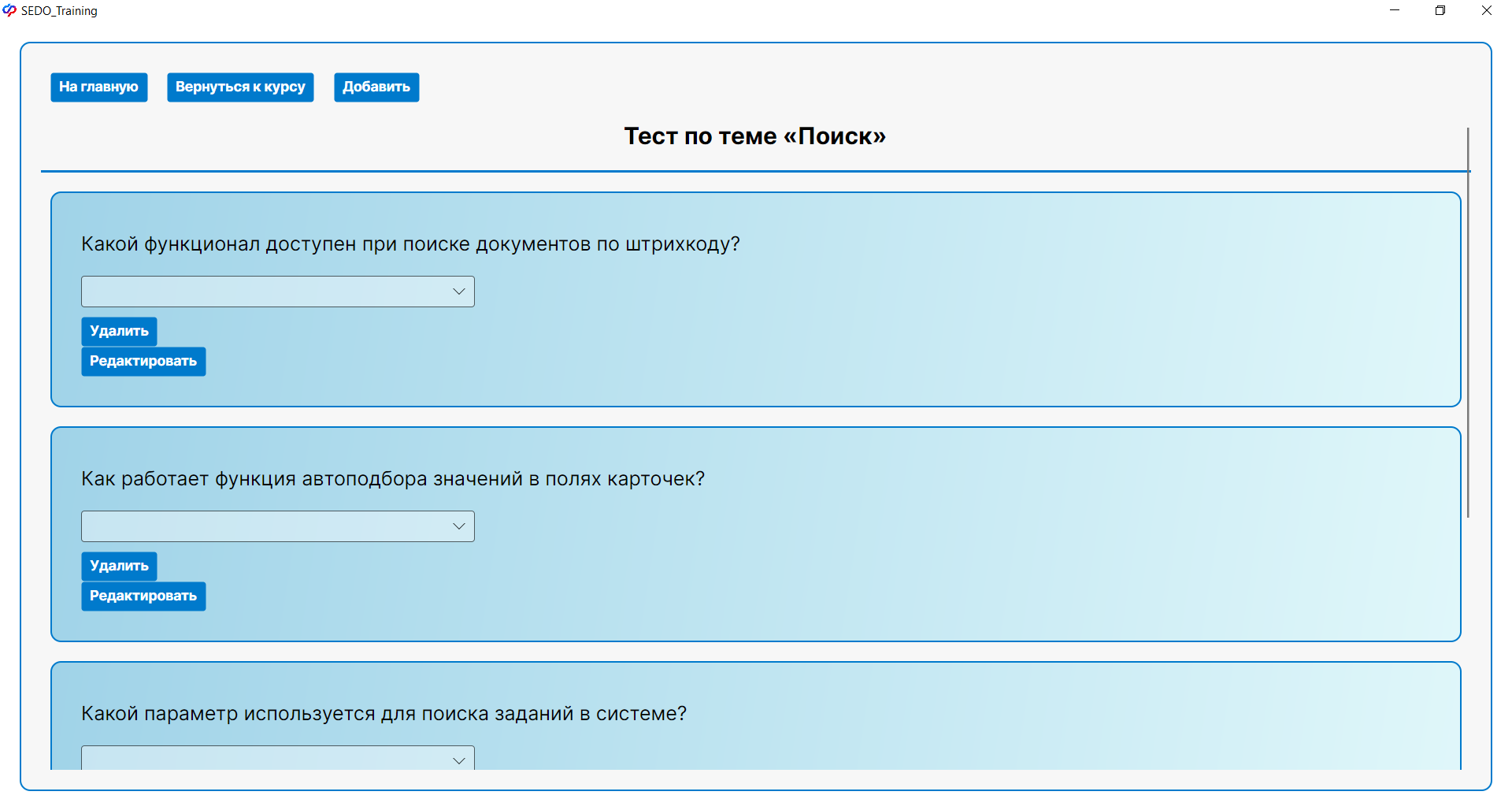


Рисунок 10 -страница тест4

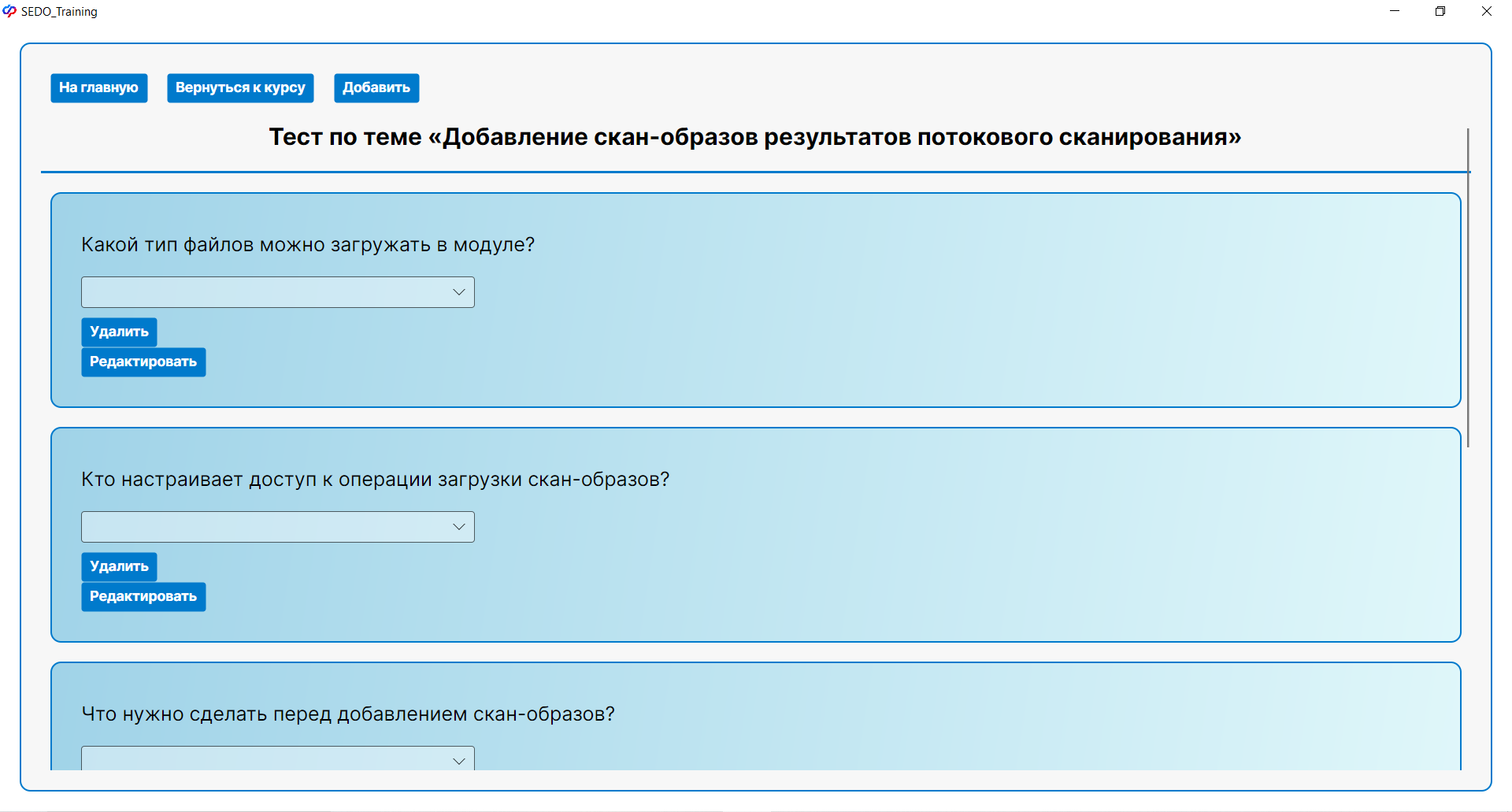


Рисунок 11 -страница тест5

Страница меню тестов

Общая структура интерфейса

Интерфейс организован в Border, который создает визуальное оформление с рамкой и закругленными углами. Внутри находится Grid, который делится на несколько строк и столбцов для упорядочивания элементов.

Элементы интерфейса

Ресурсы пользователя (User Control.Resources):

Определение конвертера изображений ImageConverter, который будет использоваться для преобразования источников изображений.

Коробка с рамкой (Border):

Создает визуальное разделение интерфейса с заданной толщиной рамки, цветом и закругленными углами.

Сетка (Grid):

Сетка разделена на строки и столбцы:

Строки:

Строка 0: Заголовок "Тесты".

Строка 1: Информация о пользователе и поле для поиска.

Строка 2: Разделительная линия.

Строка 3: Список тестов.

Строка 4: Кнопка для перехода к курсам.

Столбцы:

Первый столбец занимает оставшееся пространство.

Второй столбец используется для кнопки.

Кнопка "Пользователи":

Кнопка, которая связывается с командой ToClients. Она видима только при условии IsButtonVisible.

Заголовок "Тесты":

Текстовый блок с заголовком, выделенный крупным шрифтом и жирным начертанием, размещенный по центру.

Информация о пользователе:

StackPanel для отображения информации о текущем пользователе и его баллах:

Текстовый блок: Отображает имя пользователя.

Текстовый блок: Отображает общее количество баллов.

Текстовый блок: Статический текст "балл(ов)".

Поле для поиска (TextBox):

Поле для ввода текста с подсказкой "Поиск", связанное с Search в модели данных.

Разделительная линия:

Горизонтальный Border, который служит визуальным разделителем между информацией о пользователе и списком тестов.

Список тестов (ItemsControl):

ItemsControl, который отображает список тестов, привязанный к TestList в модели данных.

Шаблон элемента (DataTemplate): Каждый тест отображается в виде рамки с градиентным фоном, содержащей:

Название теста: Отображается в текстовом блоке с жирным шрифтом.

Описание теста: Отображается в текстовом блоке с обычным шрифтом.

Изображение теста: Отображается с помощью конвертера изображений.

Кнопка "Пройти тест": При нажатии запускает метод OnTestButtonClick.

Кнопка "Курсы":

Кнопка для перехода к курсам, связанная с командой ToMenu.

1. Тестирование информационной системы
   1. Составление чек-листов тестирования приложения

Чек-лист тестирования приложения

1. Общие проверки

* Приложение запускается без ошибок.
* Все необходимые зависимости и библиотеки установлены и работают корректно.
* Интерфейс приложения соответствует дизайну и требованиям.

2. Функциональное тестирование

* Инициализация
  + Проверить, что экземпляр TestsVM создается корректно.
  + Убедиться, что все необходимые свойства инициализируются.
* Пользователь
  + Проверить, что текущий пользователь устанавливается правильно.
  + Убедиться, что отображается логин текущего пользователя.
* Кнопки и действия
  + Проверить доступность кнопки в зависимости от роли пользователя.
  + Убедиться, что кнопка активируется/деактивируется при смене роли.
* Очки пользователя
  + Проверить правильность вычисления очков пользователя.
  + Убедиться, что TotalPoints обновляется корректно.
* Фильтрация тестов
  + Проверить, что тесты фильтруются по имени корректно.
  + Убедиться, что результаты фильтрации соответствуют ожиданиям.
* Загрузка тестов
  + Проверить, что тесты загружаются из базы данных.
  + Убедиться, что все тесты отображаются в списке.
* Навигация
  + Проверить переход к конкретному тесту.
  + Убедиться, что отображается сообщение о недоступности теста, если он отсутствует.
  + Проверить переход к странице клиентов.
  + Убедиться, что переход к главному меню работает корректно.

3. Тестирование пользовательского интерфейса

* Проверить, что все элементы интерфейса отображаются корректно на разных разрешениях экрана.
* Убедиться, что элементы управления (кнопки, текстовые поля) реагируют на действия пользователя.
* Проверить, что сообщения об ошибках и уведомления отображаются правильно.

4. Тестирование производительности

* Проверить время загрузки приложения.
* Убедиться, что приложения работает быстро при фильтрации и загрузке тестов.

5. Тестирование безопасности

* Проверить, что доступ к тестам ограничен в зависимости от роли пользователя.
* Убедиться, что данные пользователя защищены и не отображаются несанкционированным образом.

6. Тестирование на различных устройствах

* Проверить работу приложения на различных операционных системах (Windows, macOS и т.д.).
* Убедиться, что приложение корректно работает на разных версиях .NET.

7. Регрессионное тестирование

* Проверить, что новые изменения не сломали существующий функционал.
* Убедиться, что все ранее выполненные тесты проходят успешно.
  1. Составление тест-кейсов

Таблица 1. Тест-кейс для модели «Добавление вопроса»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тест-кейс ID | Описание теста | Входные данные | Ожидаемый результат | Примечание |
| TC1 | Проверка создания нового вопроса | null (для параметра user) | Объект NewQ должен быть инициализирован | Нет ошибок |
| TC2 | Проверка создания нового вопроса с заданным пользователем | User  (с Login = "testUser ") | Объект NewQ должен быть инициализирован, а CurrentUser  должен вернуть "testUser " | Нет ошибок |
| TC3 | Проверка добавления нового вопроса | NewQ с Id = 0, Correctanswerindex = 1, Questiontext = "Тестовый вопрос", Answer1 = "Ответ 1", Answer2 = "Ответ 2", Answer3 = "Ответ 3" | Новый вопрос должен быть добавлен в Questions1s | Нет ошибок |
| TC4 | Проверка обновления существующего вопроса | NewQ с Id = 1, Correctanswerindex = 2, Questiontext = "Обновленный вопрос", Answer1 = "Обновленный ответ 1", Answer2 = "Обновленный ответ 2", Answer3 = "Обновленный ответ 3" | Вопрос с Id = 1 должен быть обновлен в Questions1s | Нет ошибок |
| TC5 | Проверка сохранения вопроса с некорректным индексом ответа | NewQ с Correctanswerindex = -1 | Сообщение об ошибке "Индекс корректного ответа должен быть от 0 до 2." должно быть показано | Нет ошибок |
| TC6 | Проверка сохранения вопроса с корректным индексом ответа | NewQ с Correctanswerindex = 0 | Вопрос должен быть успешно добавлен или обновлен без ошибок | Нет ошибок |
| TC7 | Проверка навигации на страницу Test1 | Вызов метода ToLast() | Должен быть установлен PageContent на новый экземпляр Test1 | Нет ошибок |

Таблица 2. Тест-кейс для страницы «Авторизация»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тест-кейс ID | Описание теста | Входные данные | Ожидаемый результат | Примечание |
| TC1 | Проверка инициализации класса | null (при создании нового экземпляра) | User List должен быть инициализирован и не равен null | Нет ошибок |
| TC2 | Проверка отображения пароля | Установка свойства IsPasswordVisible в true | IsPasswordVisible должно быть true | Нет ошибок |
| TC3 | Проверка скрытия пароля | Установка свойства IsPasswordVisible в false | IsPasswordVisible должно быть false | Нет ошибок |
| TC4 | Проверка успешной авторизации | Login = "validUser ", Password = "validPassword" (существующий пользователь) | Должен быть создан экземпляр MenuVM, и PageContent должен измениться на Menu | Нет ошибок |
| TC5 | Проверка неуспешной авторизации | Login = "invalidUser ", Password = "invalidPassword" | Показать сообщение "Авторизация не успешна" | Нет ошибок |
| TC6 | Проверка перехода на страницу регистрации | Вызов метода ToRegistration() | PageContent должен измениться на Registration | Нет ошибок |
| TC7 | Проверка установки логина | Установка Login = "testLogin" | Значение свойства Login должно быть "testLogin" | Нет ошибок |
| TC8 | Проверка установки пароля | Установка Password = "testPassword" | Значение свойства Password должно быть "testPassword" | Нет ошибок |

Таблица 3. Тест-кейс для страницы «Клиенты»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Название теста | Предусловия | Действия | Ожидаемый результат | Примечание |
| TC01 | Проверка загрузки пользователей | Контекст базы данных содержит пользователей | Вызвать метод Load() | User List содержит загруженных пользователей | Нет ошибок |
| TC02 | Проверка расчета общей суммы очков | Установлен текущий пользователь с тестами | Вызвать метод CalculateTotalPoints() | TotalPoints равен сумме очков тестов текущего пользователя | Нет ошибок |
| TC03 | Проверка фильтрации пользователей по строке поиска | Контекст базы данных содержит пользователей | Установить \_search и вызвать AllFilters() | User List содержит только пользователей, соответствующих фильтру | Нет ошибок |
| TC04 | Проверка установки роли по умолчанию | Список ролей содержит несколько элементов | Проверить значение SelectedRoleFilter | SelectedRoleFilter равен первому элементу списка ролей | Нет ошибок |
| TC05 | Проверка фильтрации по выбранной роли | Контекст базы данных содержит пользователей | Установить SelectedRoleFilter и вызвать AllFilters() | User List содержит только пользователей с выбранной ролью | Нет ошибок |
| TC06 | Проверка обработки пустого списка пользователей | Контекст базы данных не содержит пользователей | Вызвать метод Load() | User List пустой | Нет ошибок |

Таблица 4. Тест-кейс страницы «Курсы»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Название теста | Предусловия | Действия | Ожидаемый результат | Примечание |
| TC01 | Проверка видимости кнопки для пользователя с ролью 2 | Текущий пользователь с ролью 2 | Вызвать конструктор Course1VM | IsButtonVisible равно true | Нет ошибок |
| TC02 | Проверка невидимости кнопки для пользователя с другой ролью | Текущий пользователь с ролью, отличной от 2 | Вызвать конструктор Course1VM | IsButtonVisible равно false | Нет ошибок |
| TC03 | Проверка навигации на Test1 | Текущий пользователь с ролью 2 | Вызвать метод ToTest1() | MainWindowViewModel.Instance.PageContent содержит Test1 | Нет ошибок |
| TC04 | Проверка навигации на Test2 | Текущий пользователь с ролью 2 | Вызвать метод ToTest2() | MainWindowViewModel.Instance.PageContent содержит Test2 | Нет ошибок |
| TC05 | Проверка навигации на Test3 | Текущий пользователь с ролью 2 | Вызвать метод ToTest3() | MainWindowViewModel.Instance.PageContent содержит Test\_3 | Нет ошибок |
| TC06 | Проверка навигации на Test4 | Текущий пользователь с ролью 2 | Вызвать метод ToTest4() | MainWindowViewModel.Instance.PageContent содержит Test\_4 | Нет ошибок |
| TC07 | Проверка навигации на Test5 | Текущий пользователь с ролью 2 | Вызвать метод ToTest5() | MainWindowViewModel.Instance.PageContent содержит Test\_5 | Нет ошибок |
| TC08 | Проверка навигации на Test6 | Текущий пользователь с ролью 2 | Вызвать метод ToTest6() | MainWindowViewModel.Instance.PageContent содержит Test6 | Нет ошибок |
| TC09 | Проверка навигации на Test7 | Текущий пользователь с ролью 2 | Вызвать метод ToTest7() | MainWindowViewModel.Instance.PageContent содержит Test\_7 | Нет ошибок |
| TC10 | Проверка навигации при отсутствии пользователя | Отсутствует текущий пользователь | Вызвать конструктор Course1VM | CurrentUser  равно null | Нет ошибок |

Таблица 5. Тест-кейс страницы «Изменение пользователя»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Название теста | Предусловия | Действия | Ожидаемый результат | Примечание |
| TC01 | Проверка загрузки пользователя по ID | Существующий пользователь с известным ID | Вызвать конструктор EditUser VM с ID | \_newU содержит загруженного пользователя | Нет ошибок |
| TC02 | Проверка загрузки пользователя, если ID не существует | Не существующий ID | Вызвать конструктор EditUser VM с несуществующим ID | \_newU — новый экземпляр User | Нет ошибок |
| TC03 | Проверка текущего пользователя | Установлен текущий пользователь | Вызвать CurrentUser | Возвращает логин текущего пользователя | Нет ошибок |
| TC04 | Проверка добавления изменений пользователя | Установлен текущий пользователь и NewU | Вызвать метод AddUs() и подтвердить действия | Изменения сохранены в базе данных | Нет ошибок |
| TC05 | Проверка изменения роли пользователя | Установлен текущий пользователь и NewU | Установить SelectedRole и вызвать AddUs() | Роль пользователя изменена на SelectedRole | Нет ошибок |
| TC06 | Проверка загрузки тестов текущего пользователя | Установлен текущий пользователь | Вызвать метод LoadCurrentUser Tests() | CurrentUser Tests содержит тесты текущего пользователя | Нет ошибок |
| TC07 | Проверка списка доступных тестов | Существующий пользователь | Вызвать Tests | Список тестов не содержит тестов, пройденных пользователем | Нет ошибок |
| TC08 | Проверка навигации обратно к списку клиентов | Установлен текущий пользователь | Вызвать метод ToLast() | MainWindowViewModel.Instance.PageContent содержит Clients | Нет ошибок |
| TC09 | Проверка инициализации ролей | Доступные роли в базе данных | Вызвать Roles | Список ролей загружен из базы данных | Нет ошибок |
| TC10 | Проверка корректности значения SelectedRole | Установлен текущий пользователь | Установить SelectedRole | SelectedRole равен роли пользователя из \_newU | Нет ошибок |

Таблица 6. Тест-кейс для страницы «Ввод предложений»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Название теста | Предусловия | Действия | Ожидаемый результат | Примечание |
| TC01 | Проверка инициализации InputVM | Создание экземпляра User | Создать экземпляр InputVM с пользователем | NewOf инициализирован как новый объект Offer | Нет ошибок |
| TC02 | Проверка текущего пользователя | Установлен текущий пользователь | Вызвать CurrentUser | Возвращает логин текущего пользователя | Нет ошибок |
| TC03 | Проверка добавления предложения | Установлено значение NewOf.Offers | Установить NewOf.Offers и вызвать AddOff() | Появляется сообщение об ошибке, если поле пустое | Нет ошибок |
| TC04 | Проверка подтверждения добавления предложения | Установлено значение NewOf.Offers | Установить NewOf.Offers, подтвердить добавление | NewOf добавляется в базу данных | Нет ошибок |
| TC05 | Проверка навигации в меню | Установлен текущий пользователь | Вызвать ToMenu() | MainWindowViewModel.Instance.PageContent содержит Menu | Нет ошибок |
| TC06 | Проверка обработки нажатия кнопки закрытия | Установлен текущий пользователь | Вызвать OnCloseButtonClick() | ToMenu() вызывается без ошибок | Нет ошибок |
| TC07 | Проверка, что AddOffCommand инициализирован | Создание экземпляра InputVM | Проверить AddOffCommand | AddOffCommand не равен null | Нет ошибок |
| TC08 | Проверка, что предложение не добавляется, если оно пустое | Установлено значение NewOf.Offers пустое | Вызвать AddOff() | Сообщение об ошибке отображается | Нет ошибок |
| TC09 | Проверка, что новое предложение добавляется в базу данных | Установлено значение NewOf.Offers не пустое | Установить NewOf.Offers, подтвердить добавление | NewOf должен быть в MainWindowViewModel.myConnection.Offers | Нет ошибок |

Таблица 7. Тест-кейс для страницы «Меню»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Название теста | Предусловия | Действия | Ожидаемый результат | Примечание |
| TC01 | Проверка инициализации MenuVM | Создание экземпляра User | Создать экземпляр MenuVM с пользователем | CurrentUser  инициализирован корректно | Нет ошибок |
| TC02 | Проверка получения логина текущего пользователя | Установлен текущий пользователь | Вызвать CurrentUser | Возвращает логин текущего пользователя | Нет ошибок |
| TC03 | Проверка доступности кнопки | Установлен текущий пользователь с ролью 2 | Вызвать ButtonEnabled | Возвращает true | Нет ошибок |
| TC04 | Проверка недоступности кнопки | Установлен текущий пользователь с ролью не 2 | Вызвать ButtonEnabled | Возвращает false | Нет ошибок |
| TC05 | Проверка инициализации списка курсов | Создание экземпляра MenuVM | Вызвать Load() | CourseList инициализирован и содержит курсы из базы данных | Нет ошибок |
| TC06 | Проверка фильтрации курсов по имени | Имеется список курсов | Установить Search и вызвать AllFilters() | CourseList должен содержать только курсы, соответствующие фильтру | Нет ошибок |
| TC07 | Проверка расчета общих баллов | Установлен текущий пользователь | Вызвать CalculateTotalPoints() | TotalPoints должен быть корректно рассчитан | Нет ошибок |
| TC08 | Проверка навигации к странице клиентов | Установлен текущий пользователь | Вызвать ToClients() | MainWindowViewModel.Instance.PageContent содержит Clients | Нет ошибок |
| TC09 | Проверка навигации к странице тестов | Установлен текущий пользователь | Вызвать ToTest() | MainWindowViewModel.Instance.PageContent содержит Tests | Нет ошибок |
| TC10 | Проверка навигации к странице ввода | Установлен текущий пользователь | Вызвать ToInput() | MainWindowViewModel.Instance.PageContent содержит Input | Нет ошибок |
| TC11 | Проверка навигации к курсу (существующий курс) | Установлен текущий пользователь | Вызвать NavigateToCourse(1) | MainWindowViewModel.Instance.PageContent содержит Course1 | Нет ошибок |
| TC12 | Проверка навигации к курсу (несуществующий курс) | Установлен текущий пользователь | Вызвать NavigateToCourse(999) | Появляется сообщение о том, что курс отсутствует | Нет ошибок |

Таблица 8. Тест-кейс для страницы «Регистрация»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Название теста | Предусловия | Действия | Ожидаемый результат | Примечание |
| TC01 | Проверка инициализации RegistrationVM | Нет | Создать экземпляр RegistrationVM | Экземпляр успешно создан | Нет ошибок |
| TC02 | Проверка установки логина | Создан экземпляр RegistrationVM | Установить Login | Login должен быть установлен корректно | Нет ошибок |
| TC03 | Проверка установки пароля | Создан экземпляр RegistrationVM | Установить Password | Password должен быть установлен корректно | Нет ошибок |
| TC04 | Проверка установки роли администратора | Создан экземпляр RegistrationVM | Установить IsAdmin | IsAdmin должен быть установлен корректно | Нет ошибок |
| TC05 | Проверка регистрации с пустыми полями | Создан экземпляр RegistrationVM | Установить Login и Password пустыми | Появляется сообщение об ошибке: "Все поля должны быть заполнены" | Нет ошибок |
| TC06 | Проверка регистрации с существующим логином | Создан экземпляр RegistrationVM и пользователь с логином "test" | Установить Login как "test", Password как "password" | Появляется сообщение об ошибке: "Пользователь с таким логином уже существует" | Нет ошибок |
| TC07 | Проверка успешной регистрации | Создан экземпляр RegistrationVM | Установить уникальный Login, Password, IsAdmin | Появляется сообщение: "Регистрация прошла успешно" | Пользователь добавлен в базу данных |
| TC08 | Проверка навигации к странице авторизации | Успешно зарегистрирован пользователь | Вызвать ToAuth() | MainWindowViewModel.Instance.PageContent содержит Authorization | Нет ошибок |
| TC09 | Проверка обработки исключений при регистрации | Создан экземпляр RegistrationVM | Искусственно вызвать исключение в context.SaveChangesAsync() | Появляется сообщение об ошибке: "Не удалось зарегистрировать пользователя: ..." | Нет ошибок |

Таблица 9. Тест-кейс для страницы «RelayCommand»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Название теста | Предусловия | Действия | Ожидаемый результат | Примечание |
| TC01 | Проверка инициализации RelayCommand | Нет | Создать экземпляр RelayCommand с действием | Экземпляр успешно создан | Нет ошибок |
| TC02 | Проверка выполнения команды | Создан экземпляр RelayCommand | Вызвать метод Execute | Метод Execute должен выполнить переданное действие | Нет ошибок |
| TC03 | Проверка условия выполнения команды | Создан экземпляр RelayCommand | Установить canExecute в true | CanExecute возвращает true | Нет ошибок |
| TC04 | Проверка условия выполнения команды (false) | Создан экземпляр RelayCommand | Установить canExecute в false | CanExecute возвращает false | Нет ошибок |
| TC05 | Проверка вызова события CanExecuteChanged | Создан экземпляр RelayCommand | Подписаться на событие CanExecuteChanged | Событие вызывается при вызове RaiseCanExecuteChanged | Проверка с помощью делегата |
| TC06 | Проверка обработки null в execute | Нет | Попытаться создать RelayCommand с null в execute | Генерируется ArgumentNullException | Нет ошибок |
| TC07 | Проверка работы с параметрами | Создан экземпляр RelayCommand | Вызвать Execute с параметром | Действие должно выполниться с переданным параметром | Проверка с помощью делегата |
| TC08 | Проверка работы с null в canExecute | Создан экземпляр RelayCommand | Установить canExecute в null | CanExecute возвращает true | Нет ошибок |

Таблица 10. Тест-кейс для страницы «Тест»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Название теста | Предусловия | Действия | Ожидаемый результат | Примечание |
| TC01 | Проверка инициализации Test1VM | Нет | Создать экземпляр Test1VM | Экземпляр успешно создан | Нет ошибок |
| TC02 | Проверка установки видимости кнопки | Создан экземпляр Test1VM | Установить \_currentUser .Role в 2 | IsButtonVisible должно быть true | Нет ошибок |
| TC03 | Проверка скрытия кнопки | Создан экземпляр Test1VM | Установить \_currentUser .Role в 1 | IsButtonVisible должно быть false | Нет ошибок |
| TC04 | Проверка удаления вопроса | Создан экземпляр Test1VM с вопросами | Вызвать Delete с существующим id вопроса | Вопрос удален из Questions1s, и сообщение подтверждения показано | Проверка с помощью делегата |
| TC05 | Проверка обновления вопроса | Создан экземпляр Test1VM | Вызвать Update с существующим id вопроса | Переход на страницу AddQ1 с правильным id | Нет ошибок |
| TC06 | Проверка перехода на страницу добавления вопроса | Создан экземпляр Test1VM | Вызвать ToPageAddQ1() | Переход на страницу AddQ1 | Нет ошибок |
| TC07 | Проверка проверки ответов | Создан экземпляр Test1VM с вопросами | Установить правильные и неправильные ответы | Сообщение с количеством правильных ответов показано | Проверка с помощью делегата |
| TC08 | Проверка добавления очков пользователю | Создан экземпляр Test1VM | Вызвать AddPointsToUser  с points | Очки добавлены пользователю, и запись в UsersTests создана или обновлена | Нет ошибок |
| TC09 | Проверка перехода на главную страницу | Создан экземпляр Test1VM | Вызвать ToMain() | Переход на страницу Tests | Нет ошибок |
| TC10 | Проверка перехода на курс 1 | Создан экземпляр Test1VM | Вызвать ToCourse1() | Переход на страницу Course1 | Нет ошибок |
| TC11 | Проверка получения текущего пользователя | Создан экземпляр Test1VM | Получить CurrentUser | Должен вернуть логин текущего пользователя | Нет ошибок |
| TC12 | Проверка обновления видимости кнопки после удаления | Создан экземпляр Test1VM с вопросами | Вызвать Delete с существующим id вопроса, затем UpdateButtonVisibility() | IsButtonVisible обновляется в зависимости от роли пользователя | Нет ошибок |

Таблица 11. Тест-кейс для страницы «Меню тестов»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Название теста | Предусловия | Действия | Ожидаемый результат | Примечание |
| TC01 | Проверка инициализации TestsVM | Нет | Создать экземпляр TestsVM | Экземпляр успешно создан | Нет ошибок |
| TC02 | Проверка установки текущего пользователя | Создан экземпляр TestsVM | Установить \_currentUser | CurrentUser  должен вернуть логин текущего пользователя | Нет ошибок |
| TC03 | Проверка доступности кнопки | Создан экземпляр TestsVM с ролью 2 | Проверить свойство ButtonEnabled | ButtonEnabled должно быть true | Нет ошибок |
| TC04 | Проверка недоступности кнопки | Создан экземпляр TestsVM с ролью 1 | Проверить свойство ButtonEnabled | ButtonEnabled должно быть false | Нет ошибок |
| TC05 | Проверка вычисления очков пользователя | Создан экземпляр TestsVM с существующим пользователем | Вызвать CalculateTotalPoints() | TotalPoints должен быть равен сумме очков пользователя | Проверка с помощью делегата |
| TC06 | Проверка фильтрации тестов по имени | Создан экземпляр TestsVM с тестами | Установить Search в "тест" | TestList должен содержать только тесты, содержащие "тест" | Нет ошибок |
| TC07 | Проверка загрузки тестов | Создан экземпляр TestsVM | Вызвать Load() | TestList должен содержать все тесты из базы данных | Нет ошибок |
| TC08 | Проверка навигации к тесту | Создан экземпляр TestsVM | Вызвать NavigateToTest(1) | Переход на страницу Test1 | Нет ошибок |
| TC09 | Проверка сообщения о недоступности теста | Создан экземпляр TestsVM | Вызвать NavigateToTest(99) | Показать сообщение "На данный момент тест отсутствует" | Проверка с помощью делегата |
| TC10 | Проверка перехода к странице клиентов | Создан экземпляр TestsVM | Вызвать ToClients() | Переход на страницу Clients | Нет ошибок |
| TC11 | Проверка перехода к главному меню | Создан экземпляр TestsVM | Вызвать ToMain() | Переход на страницу Menu | Нет ошибок |
| TC12 | Проверка обновления видимости кнопки | Создан экземпляр TestsVM | Изменить роль пользователя и вызвать UpdateButtonVisibility() | IsButtonVisible обновляется в зависимости от роли пользователя | Нет ошибок |

Заключение

В ходе производственной практики была проведена комплексная работа, направленная на изучение и применение методов разработки прикладных решений для компьютерных систем. Практика охватила все ключевые этапы разработки информационной системы: от анализа требований и проектирования до реализации и тестирования.

Основные достижения:

Анализ предприятия: Было проведено детальное исследование деятельности предприятия, включая описание его функционала, используемого аппаратного и программного обеспечения, а также организации обмена данными между сотрудниками. Это позволило четко определить потребности и требования к разрабатываемому приложению.

Проектирование системы: Составленный план разработки и созданные макеты страниц приложения обеспечили ясное представление о функциональности и интерфейсе системы, что значительно упростило процесс реализации.

Разработка прикладного решения: Успешная реализация приложения с графическим интерфейсом, соответствующим стилю предприятия, позволила создать инструмент, который эффективно решает конкретные проблемы, с которыми сталкивается организация.

Тестирование: Проведенные тесты, включая инспекцию кода и модульное тестирование, выявили потенциальные ошибки и недочеты, что позволило их устранить и улучшить качество конечного продукта. Составление чек-листов и тест-кейсов обеспечило системный подход к тестированию, что способствовало повышению надежности приложения.

Заключение

В результате практики были не только закреплены теоретические знания, полученные в процессе обучения, но и приобретены практические навыки работы с современными инструментами разработки, системами контроля версий и методами тестирования. Практика дала возможность глубже понять процессы, связанные с разработкой и тестированием программного обеспечения, а также научила работать в команде и соблюдать сроки выполнения задач. Эти навыки и знания будут полезны в дальнейшей профессиональной деятельности и помогут в успешной реализации будущих проектов в области информационных технологий.

Листинг 1

**CREATE** **TABLE** roles (

id **INT** **PRIMARY** **KEY**,

role **VARCHAR**(50) **NOT** **NULL**

);

**CREATE** **TABLE** users (

id **INT** **PRIMARY** **KEY**,

login **VARCHAR**(50) **NOT** **NULL** **UNIQUE**,

password **VARCHAR**(255) **NOT** **NULL**,

role **INT**,

**FOREIGN** **KEY** (role) **REFERENCES** roles(id) **ON** **DELETE** **SET** **NULL**

);

**CREATE** **TABLE** tests (

id **INT** **PRIMARY** **KEY**,

name **VARCHAR**(50),

description **VARCHAR**(100),

photo **VARCHAR**(255),

points **INT**

);

**ALTER** **TABLE** users **ALTER** **COLUMN** id **SET** **DEFAULT** nextval('users\_id\_seq');

**UPDATE** users **SET** id = nextval('users\_id\_seq') **WHERE** id **IS** **NULL**;

**CREATE** **TABLE** courses (

id **INT** **PRIMARY** **KEY**,

name **VARCHAR**(100),

description **VARCHAR**(100),

photo **VARCHAR**(255),

uk **INT**

);

**CREATE** **TABLE** test4 (

id **INT** **PRIMARY** **KEY**,

title **VARCHAR**(100),

description **VARCHAR**(100)

);

**CREATE** **TABLE** Questions1 (

id SERIAL **PRIMARY** **KEY**,

QuestionText **VARCHAR**(500),

CorrectAnswerIndex **INT** **NOT** **NULL**,

Answer1 **VARCHAR**(500),

Answer2 **VARCHAR**(500),

Answer3 **VARCHAR**(500),

SelectedAnswerIndex **INT**;

);

**CREATE** **TABLE** Questions3 (

id SERIAL **PRIMARY** **KEY**,

QuestionText **VARCHAR**(500),

CorrectAnswerIndex **INT** **NOT** **NULL**,

Answer1 **VARCHAR**(500),

Answer2 **VARCHAR**(500),

Answer3 **VARCHAR**(500),

SelectedAnswerIndex **INT**

);

**INSERT** **INTO** Questions3 (QuestionText, CorrectAnswerIndex, Answer1, Answer2, Answer3) **VALUES**

('Какой шаг необходимо выполнить для группового создания резолюций из заданий?', 1, 'Выбрать нужные задания и нажать кнопку "Создать резолюции".', 'Нажать кнопку "Отправить на утверждение".', 'Заполнить все поля в форме утверждения.'),

('Что происходит, если обязательные поля не заполнены при отправке резолюций на утверждение?', 2, 'Резолюции будут отправлены с пустыми полями.', 'Появится сообщение об ошибке, указывающее на незаполненные поля.', 'Система автоматически заполнит пустые поля.'),

('Какой порядок действий для утверждения резолюций?', 1, 'Установить флажки для заданий с неутвержденными резолюциями и нажать "Утвердить резолюцию".', 'Нажать кнопку "Создать резолюции" и заполнить форму.', 'Установить флажки для всех заданий и нажать "Отправить на утверждение".');

**INSERT** **INTO** Questions6 (QuestionText, CorrectAnswerIndex, Answer1, Answer2, Answer3, SelectedAnswerIndex) **VALUES**

('Какой тип файлов можно загружать в модуле?', 1, 'Файлы, содержащие текстовые документы', 'Файлы, содержащие штрихкод, распознанный при сканировании', 'Файлы, содержащие только изображения', **NULL**),

('Кто настраивает доступ к операции загрузки скан-образов?', 0, 'Администратор', 'Пользователи', 'Разработчики', **NULL**),

('Что нужно сделать перед добавлением скан-образов?', 1, 'Проверить наличие интернет-соединения', 'Убедиться, что установлена актуальная версия Digital\_Design\_Prioritet Web\_Service', 'Установить дополнительное программное обеспечение', **NULL**),

('Что происходит с успешно добавленными файлами, если выбрана соответствующая настройка?', 1, 'Они остаются в папке', 'Они могут быть удалены', 'Они копируются в другую папку', **NULL**),

('Что отображается после завершения загрузки?', 0, 'Таблица с результатами, включая названия файлов и статусы загрузки', 'Сообщение об ошибке', 'Статистика по всем загруженным файлам', **NULL**);

**CREATE** **TABLE** Questions6 (

id SERIAL **PRIMARY** **KEY**,

QuestionText **VARCHAR**(500),

CorrectAnswerIndex **INT** **NOT** **NULL**,

Answer1 **VARCHAR**(500),

Answer2 **VARCHAR**(500),

Answer3 **VARCHAR**(500),

SelectedAnswerIndex **INT**

);

**CREATE** **TABLE** Questions2 (

id SERIAL **PRIMARY** **KEY**,

QuestionText **VARCHAR**(500),

CorrectAnswerIndex **INT** **NOT** **NULL**,

Answer1 **VARCHAR**(500),

Answer2 **VARCHAR**(500),

Answer3 **VARCHAR**(500),

SelectedAnswerIndex **INT**

);

**INSERT** **INTO** Questions5 (QuestionText, CorrectAnswerIndex, Answer1, Answer2, Answer3, SelectedAnswerIndex) **VALUES**

('Как можно открыть карточку расширенного отчета из браузера?', 0, 'Ввести ссылку в адресное поле браузера: http://имя сервера:порт/cardExtReport/{id}', 'Нажать на кнопку "Показать отчет" в приложении.', 'Открыть отчет через меню "Файл".', **NULL**),

('Что происходит, если параметр "Запускать генерацию отчета при открытии карточки" выключен?', 1, 'Параметры отчета заполняются автоматически.', 'Появляется окно для ввода параметров.', 'Отчет формируется мгновенно без ввода параметров.', **NULL**),

('Какой формат файлов доступен для выгрузки отчета?', 1, 'PDF и TXT', 'Excel и Word', 'JPEG и PNG', **NULL**);

**INSERT** **INTO** Questions4 (QuestionText, CorrectAnswerIndex, Answer1, Answer2, Answer3, SelectedAnswerIndex) **VALUES**

('Какой функционал доступен при поиске документов по штрихкоду?', 1, 'Можно только просматривать документы.', 'Можно искать карточки документов с помощью ручного сканера.', 'Можно только удалять документы.', **NULL**),

('Как работает функция автоподбора значений в полях карточек?', 0, 'Автоподбор доступен только для текстовых полей.', 'Необходимо ввести несколько символов, после чего начнется поиск значений.', 'Автоподбор работает только для организаций.', **NULL**),

('Какой параметр используется для поиска заданий в системе?', 2, 'Только по дате регистрации.', 'По типу задания, исполнителям и сроку исполнения.', 'По номеру документа.', **NULL**),

('Как пользователи могут сохранять свои поисковые запросы?', 0, 'Запросы сохраняются автоматически без участия пользователя.', 'Пользователи могут сохранять запросы для дальнейшего использования через интерфейс.', 'Сохранение запросов доступно только администраторам.', **NULL**);

**INSERT** **INTO** Questions2 (QuestionText, CorrectAnswerIndex, Answer1, Answer2, Answer3, SelectedAnswerIndex) **VALUES**

('Какое действие необходимо выполнить для создания задания?', 1, 'Передать задание другому пользователю.', 'Заполнить форму создания проекта резолюции.', 'Открыть задание в режиме чтения.', **NULL**),

('Как можно открыть задание в интерфейсе модуля?', 1, 'Только по ссылке в адресной строке браузера.', 'По ссылке, из представления папки или через ссылки в блоках интерфейса.', 'Только через меню "Файл".', **NULL**),

('Что происходит с заданием при наличии неутвержденной резолюции?', 1, 'Задание становится доступным для редактирования.', 'Задание блокируется для других пользователей и доступно только в режиме чтения.', 'Задание автоматически закрывается.', **NULL**),

('Какое действие необходимо выполнить, чтобы взять задание в работу?', 1, 'Нажать кнопку "Передать задание".', 'Открыть задание и нажать кнопку "В работу".', 'Нажать кнопку "Отклонить".', **NULL**),

('Как можно просмотреть историю операций с заданием?', 0, 'Открыть задание и нажать на кнопку "История".', 'Открыть задание и выбрать "Файл" → "История".', 'История операций доступна только для администраторов.', **NULL**);

**CREATE** **TABLE** Questions4 (

id SERIAL **PRIMARY** **KEY**,

QuestionText **VARCHAR**(500),

CorrectAnswerIndex **INT** **NOT** **NULL**,

Answer1 **VARCHAR**(500),

Answer2 **VARCHAR**(500),

Answer3 **VARCHAR**(500),

SelectedAnswerIndex **INT**

);

**CREATE** **TABLE** Questions5 (

id SERIAL **PRIMARY** **KEY**,

QuestionText **VARCHAR**(500),

CorrectAnswerIndex **INT** **NOT** **NULL**,

Answer1 **VARCHAR**(500),

Answer2 **VARCHAR**(500),

Answer3 **VARCHAR**(500),

SelectedAnswerIndex **INT**

);

**CREATE** **TABLE** Questions7 (

id SERIAL **PRIMARY** **KEY**,

QuestionText **VARCHAR**(500),

CorrectAnswerIndex **INT** **NOT** **NULL**,

Answer1 **VARCHAR**(500),

Answer2 **VARCHAR**(500),

Answer3 **VARCHAR**(500),

SelectedAnswerIndex **INT**

);

**INSERT** **INTO** Questions1 (id, QuestionText, CorrectAnswerIndex, Answer1, Answer2, Answer3) **VALUES**

(1, 'Какую информацию можно получить при добавлении корреспондента-юридического лица в поле адресатов?', 1, 'Только полное наименование контрагента', 'Полное наименование, руководителя, регион, адрес, ИНН и ОГРН, а также дополнительные реквизиты, такие как номер экземпляра, вид доставки и электронная почта', 'Только номер экземпляра и вид доставки'),

(2, 'Какое действие происходит при выборе сотрудников из роли Справочника сотрудников в поле Адресаты внутреннего документа, если администратор настроил возможность выбора из списка рассылки?', 0, 'Появляется окно для выбора сотрудников из списка рассылки', 'Все сотрудники выбранной роли добавляются автоматически без подтверждения', 'Появляется окно для выбора сотрудников, но добавляются только неактивные сотрудники'),

(3, 'Как можно назначить нового исполнителя для документа в поле Исполнитель?', 0, 'Нажать на ФИО текущего исполнителя и выбрать нового из выпадающего списка', 'Ввести полное имя нового исполнителя вручную', 'Нажать на значок удаления рядом с текущим исполнителем'),

(4, 'Как можно добавить новых корреспондентов при регистрации документа?', 2, 'Ввести данные нового корреспондента вручную в поле для комментариев', 'Выбрать корреспондента из справочника и нажать кнопку Добавить', 'Нажать на кнопку +Создать нового корреспондента в конце списка найденных контрагентов'),

(5, 'Как указать номер бланка строгой отчетности при регистрации документа?', 0, 'Заполнить поле Адресаты или Корреспонденты, выбрать объект и указать номер бланка в открывшейся форме', 'Ввести номер бланка вручную в поле Номер документа', 'Нажать на кнопку Добавить рядом с полем Номер бланка');

**INSERT** **INTO** test4 (id, title, description) **VALUES**

(1, 'Быстрый поиск', 'Функция, которая позволяет пользователям находить документы и задания, вводя ключевые слова в поисковой строке на любой странице модуля. Для использования этой функции достаточно ввести текст и нажать Enter, после чего на экране отобразится список карточек, содержащих искомую информацию. В случае, если найденных карточек становится слишком много и превышает установленный лимит, пользователю будет предложено уточнить параметры поиска для более точных результатов'),

(2, 'Полнотекстовый поиск', 'Функция, позволяющая пользователям находить документы и задания в базе данных по введенным словам. Результаты отображаются в отдельном окне, где можно выбрать область поиска, включая реквизиты документа или вложенные файлы. Пользователи имеют возможность фильтровать результаты по типу документа, дате регистрации и другим атрибутам, а также сортировать карточки по различным критериям. Для удобства предварительного просмотра документов достаточно нажать на заголовок найденной карточки'),

(3, 'Входящие', 'Этот раздел предназначен для поиска документов, поступивших в организацию. Включает поля для корреспондента, получателя, даты регистрации и подписанта. Поддерживает автоподбор и спецсимвол «\*» для поиска как физических, так и юридических лиц. Также позволяет искать по номеру документа с условиями «Содержит» или «Начинается с», в зависимости от настроек администратора.'),

(4, 'Исходящие', 'Этот раздел предназначен для поиска документов, отправленных из организации, а также внутренних документов. Содержит поля для подписавшего, исполнителей, получателя и визирующего лица. Поддерживает автоподбор, возможность поиска по родительской организации и использование спецсимвола «\*». Также включает разделы для тематического тегирования, отслеживания операций с документами и использования свойств различных типов для более точного поиска'),

(5, 'Расширенный поиск', 'Этот функционал приложения предлагает три варианта поиска: документов, заданий и реестров приема-передачи. Результаты отображаются в виде строки с параметрами. Если результаты отсутствуют, пользователю предложат изменить параметры запроса. В случае превышения лимита карточек появится уведомление с просьбой уточнить запрос. Также предусмотрена возможность поиска по недоступным для выбора значениям и учет переводов сотрудников между подразделениями')

**INSERT** **INTO** courses (id, name, photo, uk) **VALUES**

(1, 'Операции с документом', "course1.png", 1),

(2, 'Операции с заданием', "course2.png", 2),

(3, 'Операции с резолюцией', "course3.png" 3),

(4, 'Поиск', "course4.png", 4),

(5, 'Отчёты', "course6.png", 5),

(6, 'Добавление скан-образов результатов потокового сканирования', "course7.png", 6);

**INSERT** **INTO** tests (id, name) **VALUES**

(1, 'Операции с документом'),

(2, 'Операции с заданием'),

(3, 'Операции с резолюцией'),

(4, 'Операции с реестром приема-передач'),

(5, 'Поиск'),

(6, 'Отчёты'),

(7, 'Добавление скан-образов результатов потокового сканирования');

**CREATE** **TABLE** users\_tests (

id SERIAL **PRIMARY** **KEY**,

users **INT**,

**FOREIGN** **KEY** (users) **REFERENCES** users(id) **ON** **DELETE** **SET** **null**,

tests **INT**,

**FOREIGN** **KEY** (tests) **REFERENCES** tests(id) **ON** **DELETE** **SET** **null**,

points **INT**

);

**CREATE** **TABLE** offers (

id SERIAL **PRIMARY** **KEY**,

offers TEXT

);