**ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ ГОРОДА МОСКВЫ**

**Государственное бюджетное профессиональное образовательное   
учреждение города Москвы**

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

**ДОПУЩЕН(А) К ЗАЩИТЕ**

И.о. заместителя директора

\_\_\_\_\_\_\_\_\_\_\_\_ Топильский А.В.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

**ДИПЛОМНЫЙ ПРОЕКТ**

**Тема: «Разработка информационного ресурса для обучения и дистанционного тестирования сотрудников НИИ неотложной детской хирургии и травматологии Департамента здравоохранения города Москвы»**

выполнен

обучающимся группы ИСП-411 Кузнецовым Матвеем Сергеевичем

(Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (дата)

по специальности

09.02.07 Информационные системы и программирование

(код и наименование специальности)

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_ Кузьмина О.Б. «\_\_\_\_» \_\_\_\_\_\_\_\_\_ 2022 г.

(подпись) (И.О. Фамилия)

Консультант по

экономической части \_\_\_\_\_\_\_\_\_\_\_\_\_ Носаченко Е.А. «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2022 г.

(подпись) (И.О. Фамилия)

**Согласовано**

Заведующий УО «Юниор»

\_\_\_\_\_\_\_\_\_\_\_ С.В. Лукьянов

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

2023 г.

**ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ ГОРОДА МОСКВЫ**

**Государственное бюджетное профессиональное образовательное   
учреждение города Москвы**

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

|  |  |
| --- | --- |
| РАССМОТРЕНО и ОДОБРЕНО  НА ЗАСЕДАНИИ СЦН ПО НАПРАВЛЕНИЮ «ПРИКЛАДНЫЕ ИНФОРМАЦИОННЫЕ  ТЕХНОЛОГИИ»  Протокол №\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Скиллсменеджер СЦН \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.Б. Кузьмина | УТВЕРЖДАЮ  Заведующий отделением  \_\_\_\_\_\_\_\_\_\_\_ С.В. Лукьянов  «\_\_\_» \_\_\_\_\_\_ 2023 г. |
|  |  |

**ЗАДАНИЕ**

НА ДИПЛОМНЫЙ ПРОЕКТ ОБУЧАЮЩЕМУСЯ (ЙСЯ)

Кузнецову Матвею Сергеевичу

(фамилия, имя, отчество)

|  |  |
| --- | --- |
| 1. Тема | Разработка информационного ресурса для обучения и дистанционного тестирования сотрудников НИИ неотложной детской хирургии и травматологии Департамента здравоохранения города Москвы |

Тема утверждена приказом по колледжу 28.11.2022 г. № 193-3/лу.

1. Срок сдачи обучающимся законченного дипломного проекта: \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.
2. Исходные данные к дипломному проекту: Техническое задание на разработку программного продукта, принципы стилистического оформления пользовательского интерфейса.
3. Перечень вопросов, подлежащих разработке в дипломном проекте:
   1. Провести анализ предметной области;
   2. Определить основные требования к функциональности приложения;
   3. Ознакомиться с аналогами разрабатываемого программного продукта;
   4. Выполнить проектирование интерфейса и дизайна приложения, путём создания прототипа;
   5. Разработать проектную документацию;
   6. Разработать базу данных на основе проектной документации и имеющихся данных;
   7. Реализовать поэтапную разработку настольного приложения;
   8. Выполнить тестирование и отладку приложения;
   9. Разместить исходный код настольного приложения в системе контроля версий;
   10. Произвести расчет себестоимости программного проекта;
   11. Произвести расчет стоимости программного продукта;
   12. Произвести оценку экономической эффективности внедрения проекта.
4. Перечень иллюстрированных материалов (графические материалы с точным указанием обязательных чертежей или раздаточных материалов, с указанием наименования таблиц, рисунков или схем).

Не предоставлено.

Консультанты:

|  |  |  |  |
| --- | --- | --- | --- |
| **Раздел работы содержание определяется СЦН по направлению** | **Консультант по разделу (ФИО)** | **Дата** | **Подпись** |
| Раздел: Экономическая часть | Носаченко Елена Александровна |  |  |

|  |  |
| --- | --- |
| Руководитель дипломного проекта | Задание принял (а) к исполнению обучающийся (щаяся) |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_/Кузьмина О.Б./  подпись Ф.И.О.  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_/Кузнецов М.С./  подпись Ф.И.О.  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г. |

Оглавление

[ВВЕДЕНИЕ 5](#_Toc137122371)

[ГЛАВА 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 7](#_Toc137122372)

[1.1. СОЗДАНИЕ ОБРАЗОВАТЕЛЬНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И ЕГО ИСТОРИЯ: ОТ МЕХАНИЧЕСКИХ МАШИН ДО ПЕРВЫХ КОМПЬЮТЕРОВ 7](#_Toc137122373)

[1.2. КОМПЬЮТЕРНЫЕ УЧЕБНЫЕ ПРОГРАММЫ 8](#_Toc137122374)

[1.3. ЯЗЫК ПРОГРАММИРОВАНИЯ C# 12](#_Toc137122375)

[1.4. ЯЗЫК ПРОГРАММИРОВАНИЯ C++ 13](#_Toc137122376)

[1.5. ЯЗЫК ПРОГРАММИРОВАНИЯ JAVA 14](#_Toc137122377)

[1.6. ПЛАТФОРМА WPF ДЛЯ РАЗРАБОТКИ НАСТОЛЬНЫХ ПРИЛОЖЕНИЙ 17](#_Toc137122378)

[1.7. ПЛАТФОРМА WINFORMS ДЛЯ РАЗРАБОТКИ НАСТОЛЬНЫХ ПРИЛОЖЕНИЙ 18](#_Toc137122379)

[1.8. СРЕДА РАЗРАБОТКИ VISUAL STUDIO 20](#_Toc137122380)

[1.9. СРЕДА РАЗРАБОТКИ JETBRAINS RIDER 22](#_Toc137122381)

[1.10. ОБОСНОВАНИЯ НЕОБХОДИМОСТИ РАЗРАБОТКИ ВНУТРЕННЕГО ПО ДЛЯ ТЕСТИРОВАНИЯ СОТРУДНИКОВ КОМПАНИИ 23](#_Toc137122382)

[1.11. ТРЕБОВАНИЯ К ПРОЕКТУ 24](#_Toc137122383)

[ГЛАВА 2. РАЗРАБОТКА ИНФОРМАЦИОННОГО РЕСУРСА ДЛЯ ОБУЧЕНИЯ И ДИСТАНЦИОННОГО ТЕСТИРОВАНИЯ СОТРУДНИКОВ. 29](#_Toc137122384)

[2.1 ПЛАН РАЗРАБОТКИ 29](#_Toc137122385)

[2.2. ПЛАНИРОВАНИЕ 30](#_Toc137122388)

[2.3. АНАЛИЗ 30](#_Toc137122389)

[2.4. ПРОЕКТИРОВАНИЕ 31](#_Toc137122390)

[2.5. РАЗРАБОТКА 35](#_Toc137122391)

[2.6. РАЗВЕРТЫВАНИЕ И ЗАПУСК 61](#_Toc137122392)

[ГЛАВА 3. ЭКОНОМИЧЕСКАЯ ЧАСТЬ 63](#_Toc137122393)

[ЗАКЛЮЧЕНИЕ 70](#_Toc137122394)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 71](#_Toc137122395)

# ВВЕДЕНИЕ

С давних времен человечество постоянно обучалось. Изучение нового является основным двигателем прогресса, поэтому способы получения знаний менялись и формировались на протяжении тысячелетий.

Обучение – это процесс приобретения нового понимания, знаний, поведения, навыков, ценностей, установок и предпочтений.

В процессе постоянного накопления знаний человеку стало сложно передавать их в устной форме, поэтому были созданы различные способы хранения информации. Основным из таких способов является письменность. Человечество записывало те знания, что необходимо было передать другим или же обратиться к ним впоследствии.

Совершенствование, структурирование и оптимизация технологий письменного хранения данных позволяло выводить самообучение на новый уровень. В следствии чего количество потребляемой информации серьезно увеличилось, поэтому необходимо было автоматизировать данный процесс. Автоматизация стала одним из основных терминов в сфере IT-технологий. Электронные системы обучения набирали популярность и увеличивали процесс изучения информации в несколько раз.

В наше время жизнь каждого человека строится на ежедневном поглощении информации из различных источников, поэтому актуальность автоматизированного обучения находится на очень высоком уровне. При приеме на работу очень неудобно лично обучать каждого сотрудника всем тонкостям работы в компании, поэтому куда эффективнее использовать готовые информационные ресурсы, после изучения которых сотрудник получает необходимые для продуктивной работы знания и навыки.

Целью дипломного проекта является создание электронной системы обучения с системой дистанционного тестирования сотрудников. Она позволит организовывать внедрение новых специалистов в работу компании на высоком уровне. Отличительная черта данной системы является её полная автономность.

Основные задачи дипломного проекта:

1. Провести анализ предметной области, изучить требования заказчика к продукту;
2. Ознакомиться с аналогами разрабатываемого программного продукта;
3. Выполнить проектирование интерфейса и дизайна приложения, путём создания прототипа;
4. Реализовать поэтапную разработку настольного приложения;
5. Выполнить тестирование и отладку приложения.

# ГЛАВА 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## СОЗДАНИЕ ОБРАЗОВАТЕЛЬНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И ЕГО ИСТОРИЯ: ОТ МЕХАНИЧЕСКИХ МАШИН ДО ПЕРВЫХ КОМПЬЮТЕРОВ

Сегодняшнее образовательное программное обеспечение (ПО) представляет собой комплекс приложений, которые предназначены для развития определенных навыков у студентов. Однако, история образования насчитывает более ста лет, и первые образовательные механические устройства появились в конце XIX века. Ранее, перед появлением механических устройств, главными источниками знаний были учителя и книги, но обучение было очень трудоемким и неэффективным.

Идея заменить преподавателей механическими учебными машинами появилась в связи с успехами промышленной революции, и была направлена на ускорение процесса обучения и повышение эффективности. Хотя идея заменить учителей механическими устройствами сегодня кажется наивной, она стала основой для развития современных технологий.

Первым образовательным механическим устройством для изучения грамматики стало устройство, разработанное Алкионом Скиннером и запатентованное им в 1866 году. В 1911 году Герберт Остин Эйкинс создал механическое устройство для обучения арифметике, чтению и правописанию. А в 1912 году Эдвард Ли Торндайк предложил новый подход к образованию, основанный на механической книге, где следующие разделы открываются только после того, как предыдущие были усвоены.

Сегодняшние технологии образования, такие как онлайн-курсы, электронные учебники, и программное обеспечение, являются продуктом продолжительного и эволюционного процесса, начавшегося с появления первых образовательных механических устройств в конце XIX века. Несмотря на прогресс, достигнутый в современных технологиях, эти первоначальные идеи остаются фундаментальными для развития современных методов обучения.

В 1924 году профессор Сидни Пресси из Огайо разработал систему обучения под названием Automatic Teacher, которая позволяла ученику выбирать правильный ответ из предложенных вариантов на барабане машины и отмечала количество верных ответов на счетчике, но не могла обучать, а только проверять знания путем прохождения тестов. Многие изобретатели начали создавать новые обучающие машины, объединяя опыт XIX века, идеи Торндайка и технологии нового времени. В 1954 году профессор Беррес Фредерик Скиннер сформулировал теорию программированного обучения, в которой главным компонентом обучающего устройства должна быть строгая программа с элементами для изучения и проверки материала, а сам процесс обучения является ступенчатым.

Скиннер представил «обучающую машину», которая печатала вопросы на бумажных карточках и демонстрировала их покадрово в специальном окошке, а ученик набирал ответ на клавиатуре устройства. Система Скиннера не только тестировала знания, но и обучала студентов, и была первым устройством, в котором удалось сочетать результаты теоретических изысканий в педагогической психологии с технологическими новшествами того времени. В 1960 году инженер Дональд Битцер создал компьютерную систему PLATO (Programmed Logic for Automated Teaching Operations), которая основывалась на теории программированного обучения и предоставляла студентам возможность изучения нескольких тематических курсов. В настоящее время система не используется, но компания PLATO Learning (ныне Edmentum) разрабатывает учебные курсы. В 60-е годы появилась критика новых технологий обучения, связанная с недостаточной методической и технической подготовкой учителей, дороговизной оборудования и возможной дегуманизацией образования. Однако дальнейшее развитие показало, что опасения были напрасными.

## КОМПЬЮТЕРНЫЕ УЧЕБНЫЕ ПРОГРАММЫ

КУП, также известные как компьютерные учебные программы, представляют собой программы, которые используются в учебном процессе и взаимодействуют с учащимся. Они включают в себя различные типы, такие как обучающие, тестирующие, тренажерные, моделирующие, учебные игры, прикладные и интегрированные программы. Обучающие программы, такие как автоматизированные учебные курсы (АУК), компьютерные учебники, программы искусственного интеллекта, компьютерные справочники и энциклопедии, могут использоваться для представления, проверки и оценки усвоения учебного материала.

АУК могут содержать вступительную часть, учебный материал и вопросы для проверки понимания. Они также могут предлагать предварительные тесты для определения уровня подготовки и дополнительные вопросы, помощь или правильные ответы в ответ на ответы учащихся. Обучение может осуществляться с помощью различных форматов, таких как текст, графика, мультимедиа и аудио- и видеоинформация.

Компьютерные учебники предназначены для самостоятельного изучения теоретического материала и могут быть построены на гипертекстовой основе. Они содержат структурированный учебный материал, представленный в виде интерактивных кадров с мультимедийными приложениями. Электронный учебник позволяет учащимся выбирать оптимальную траекторию изучения материала, регулировать темп и способ изложения в соответствии с их психофизиологическими особенностями восприятия.

Кроме того, в электронных учебниках может быть предусмотрена возможность протоколирования действий учащихся для анализа преподавателем. В настоящее время компьютерные учебники считаются наиболее развитым средством обучения.

КУП-ы, такие как компьютерные справочники и энциклопедии, предназначены исключительно для представления учебного материала и часто содержат большие объемы информации, поддерживаемые технологиями мультимедиа и гипертекста. Каждый КОП должен иметь справочную систему, которая может быть встроена в электронный учебник или являться самостоятельным элементом.

В образовательных целях используется и искусственный интеллект (ИИ), который включает два основных типа программ: программы, моделирующие учащегося, и экспертные системы. Разработчики обучающих программ ИИ считают важным, чтобы программа могла строить модель обучаемого, учитывая его способности и оптимальные режимы усвоения материала. После сравнения с образцовой моделью вырабатываются стратегии, методы и последовательность обучения.

Учебные программы искусственного интеллекта подразделяются на несколько типов, включая эвристические программы, реагирующие учебные среды, программы-инструкторы, программы-помощники и тренажерные программы. Эвристические программы используют методы обучения, которые дают общее направление решения задач. Реагирующая учебная среда позволяет исправлять ошибки и учить навыкам мышления, основанным на тьюторских правилах. Программа-инструктор оценивает действия обучаемого и моделирует его умения и навыки. Программа-помощник направляет процесс обучения, предлагая знания и помогая усвоить понятия. Тренажерные программы используются для практического закрепления учебного материала и тренировки операторов в различных технических системах. Компьютерные задачники могут также рассматриваться как тренажеры, которые позволяют отработать приемы решения типовых задач.

Компьютерное моделирование используется для имитации реальных объектов и процессов с целью изучения их свойств. Этот метод особенно полезен для моделирования сложных, опасных или дорогостоящих процессов, например, ядерных реакторов или превращения куколки в бабочку. Хотя компьютерные модели упрощают реальность, они позволяют обучаемым изучать объекты и процессы, управлять ими и понимать их особенности. Компьютерные модели не являются универсальными и предназначены для моделирования конкретных явлений. Они могут использоваться для демонстрации трудно воспроизводимых в учебной обстановке явлений и выяснения влияния параметров на изучаемые процессы и явления.

Учебные игры являются похожими на имитационные программы и помогают учащимся усваивать знания и навыки. Они могут быть разных типов, включая приключенческие, настольные, азартные, боевые и психомоторные игры. С развитием компьютеров в обучении, учебные заведения начали использовать прикладные программы общего назначения, называемые инструментальными средствами учащихся. Компьютеры могут использоваться для проведения расчетов, ввода данных, обучения основам управления роботами и аппаратами, хранения и поиска информации. Примерами прикладных программ, используемых в учебных заведениях, являются редакторы, базы данных и файловые системы, электронные динамические системы.

Редакторы используются для управления курсором на ранних стадиях обучения, редактирования документов, хранения данных, обучения последовательному изложению мыслей и составления различных текстовых документов. Базы данных и файловые системы используются для обучения пользованию готовыми файловыми системами, создания своих файлов, разработки, создания и использования файлов данных.

Интегрированные системы представляют собой программные комплексы, содержащие несколько типов компьютерных учебных программ, например, компьютерные учебники, экзаменаторы, тренажеры, программы моделирования и справочники.

Далее рассмотрим различные языки программирования, которые сегодня можно использовать для разработки для разработки компьютерных образовательный приложений.

## ЯЗЫК ПРОГРАММИРОВАНИЯ C#

C# (читается как «си шарп») – это Язык программирования, разработанный компанией Microsoft в 2000 году. Он основан на языке C++, но содержит множество усовершенствований и расширений.



Рисунок 1 – Логотип языка программирования C#

Преимущества языка C#:

Простота и удобство использования: C# имеет четкую и простую синтаксическую структуру, которая делает его более доступным для начинающих программистов. Большинство разработчиков на C# отмечают, что этот язык более удобен в использовании, чем C++.

Платформонезависимость: C# разрабатывался для использования на платформе .NET Framework, что позволяет создавать приложения, которые будут работать на любой платформе, поддерживающей эту технологию.

Безопасность: C# обладает встроенными средствами защиты от нежелательных действий, таких как переполнение буфера и другие уязвимости, что делает его более безопасным в использовании.

Обширная библиотека классов: .NET Framework включает обширную библиотеку классов, которая содержит множество готовых решений для программистов.

Недостатки языка C#:

Зависимость от платформы: Приложения, написанные на C#, требуют наличия установленного .NET Framework на компьютере пользователя. Это может создавать проблемы совместимости и использования на старых версиях операционных систем.

Производительность: Хотя C# проще и удобнее в использовании, чем C++, он может быть медленнее в работе в некоторых случаях, из-за того, что является языком со сборкой мусора и имеет другие особенности.

Ограничения: C# не является языком системного программирования и не подходит для написания низкоуровневых приложений.

Почему стоит выбрать C# для разработки настольных приложений?

Удобство использования: C# предоставляет более простой и понятный синтаксис, чем C++, что упрощает разработку приложений.

## ЯЗЫК ПРОГРАММИРОВАНИЯ C++

C++ – это компилируемый, статически типизированный язык программирования общего назначения. Он был разработан в 1980-х годах как расширение языка C и стал одним из наиболее популярных языков программирования в мире.

Основные плюсы языка C++:

Высокая производительность: C++ – это быстрый язык программирования, который позволяет разработчикам создавать высокопроизводительные приложения, использующие низкоуровневые механизмы работы с памятью.

Мощные возможности: C++ предоставляет разработчикам возможности для многопоточности, перегрузки операторов, наследования, шаблонов, исключений и других расширенных функций.

Платформенная независимость: C++ позволяет разработчикам писать код, который может быть скомпилирован на разных операционных системах, что делает его идеальным для разработки кроссплатформенных приложений.

Широкое распространение: C++ – один из наиболее популярных языков программирования, который используется во многих индустриях, включая игровую, финансовую, научную и промышленную сферы.

Несмотря на все преимущества, C++ также имеет свои минусы:

Сложность: C++ - это очень сложный язык программирования, который требует от разработчиков обширных знаний в области компьютерных наук, а также опыта работы с низкоуровневыми механизмами работы с памятью.

Уязвимость: C++ не является безопасным языком программирования, так как уязвимости могут возникать из-за ошибок в управлении памятью, например, из-за переполнения буфера.

Неудобство: C++ - это язык программирования, который требует от разработчиков большого количества усилий для создания приложения. Например, программисты должны написать значительный объем кода только для создания оконного интерфейса.

Сложность отладки: При использовании C++ могут возникать трудности в отладке кода из-за большого объема кода и использования сложных конструкций языка.



Рисунок 2 – Логотип языка программирования C++

## ЯЗЫК ПРОГРАММИРОВАНИЯ JAVA

Java – это объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems в 1995 году. Java был создан с целью упрощения разработки программного обеспечения, которое могло бы запускаться на различных платформах. Вот некоторые плюсы и минусы Java:

Плюсы Java:

Переносимость: Java код может выполняться на любой платформе, поддерживающей Java Virtual Machine (JVM). Это означает, что разработчики могут создавать приложения, которые могут быть запущены на Windows, Mac или Linux без необходимости переписывания кода.

Безопасность: Java был создан с учетом безопасности. Он имеет встроенные механизмы для предотвращения ошибок памяти и уязвимостей в программном обеспечении. Также Java имеет встроенный механизм обработки исключений, что повышает безопасность приложений.

Обширная стандартная библиотека: Java поставляется с обширной стандартной библиотекой, которая содержит множество функций и классов для решения различных задач. Это позволяет разработчикам быстро и легко создавать приложения, используя готовые компоненты.

Многопоточность: Java имеет встроенную поддержку многопоточности, что позволяет создавать приложения, которые могут выполнять несколько задач одновременно. Это особенно полезно для создания высокопроизводительных приложений и игр.

Широкое использование: Java широко используется в корпоративном секторе и веб-разработке. Многие известные платформы, такие как Apache Hadoop, Apache Tomcat, Eclipse, NetBeans и Android, разработаны на Java.

Минусы Java:

Низкая производительность. Java является языком программирования с высоким уровнем абстракции, что может привести к медленной работе приложения. Использование виртуальной машины и сборки мусора может снизить производительность и увеличить нагрузку на систему.

Высокое потребление памяти. Использование виртуальной машины и сборки мусора приводит к тому, что приложения на Java требуют больше оперативной памяти, чем некоторые другие языки программирования.

Неудобство при работе с низкоуровневыми задачами. Java не является идеальным выбором для разработки системного программного обеспечения, так как не обладает доступом к низкоуровневым системным функциям.

Сложность в проектировании интерфейсов. Java не обладает нативной поддержкой для разработки графического интерфейса пользователя. В то же время, многие среды разработки и библиотеки позволяют создавать интерфейсы, но это может потребовать дополнительного времени и усилий.

Большой размер приложений. Использование виртуальной машины и библиотек может привести к тому, что размер приложения на Java будет заметно больше, чем на некоторых других языках программирования.

Проблемы безопасности. Некоторые уязвимости безопасности могут возникнуть в связи с использованием Java, особенно если устройство имеет устаревшую версию Java Runtime Environment.

Ограниченная производительность при многопоточности. В некоторых случаях производительность приложения на Java может снизиться при работе с многопоточностью из-за использования мониторов, синхронизации и других механизмов синхронизации.



Рисунок 3 – Логотип языка программирования Java

## ПЛАТФОРМА WPF ДЛЯ РАЗРАБОТКИ НАСТОЛЬНЫХ ПРИЛОЖЕНИЙ

Существует множество платформ для разработки настольных приложений, и каждая из них имеет свои преимущества и недостатки. Одной из наиболее популярных и распространенных платформ для разработки настольных приложений является Windows Presentation Foundation (WPF).

WPF – это технология, которая была разработана компанией Microsoft и входит в состав .NET Framework. Она позволяет создавать приложения с привлекательным пользовательским интерфейсом, который поддерживает различные элементы управления, анимацию, графику и мультимедиа. Вот несколько преимуществ WPF:

Графический интерфейс. Один из основных преимуществ WPF – это его графический интерфейс, который позволяет создавать красивые и современные пользовательские интерфейсы. WPF предлагает множество элементов управления, которые легко настраиваются и поддерживают различные стили.

XAML. WPF использует язык разметки XAML (eXtensible Application Markup Language), который позволяет разработчикам создавать графические элементы и располагать их на форме без необходимости программирования на языке C#.

Анимация. WPF предоставляет мощный инструментарий для создания анимации и визуальных эффектов, которые могут быть использованы для улучшения пользовательского интерфейса.

Интеграция с другими технологиями Microsoft. WPF может быть интегрирован с другими технологиями Microsoft, такими как LINQ to SQL, Windows Communication Foundation (WCF) и Windows Workflow Foundation (WF).

Несмотря на все преимущества, у WPF есть и недостатки:

Сложность. WPF является сложной технологией, и ее изучение может занять много времени. Кроме того, при разработке на WPF может возникнуть необходимость в использовании других технологий Microsoft, что также может быть сложно для новичков.

Ограничения. WPF может быть ограничен в использовании на определенных платформах, таких как Linux и macOS.

Необходимость установки .NET Framework. Для запуска приложений, созданных на WPF, необходимо установить .NET Framework на компьютере пользователя.

В целом, WPF – это мощная и популярная платформа для разработки настольных приложений с красивым графическим интерфейсом.



Рисунок 4 – Логотип платформы WPF для разработки

## ПЛАТФОРМА WINFORMS ДЛЯ РАЗРАБОТКИ НАСТОЛЬНЫХ ПРИЛОЖЕНИЙ

WinForms – это технология для создания настольных приложений под операционную систему Windows. Она была введена Microsoft в 2002 году и стала популярной среди разработчиков благодаря своей простоте и удобству использования.

Плюсы WinForms:

Простота создания интерфейса пользователя – WinForms предоставляет разработчикам множество готовых элементов управления, таких как кнопки, поля ввода, таблицы и т.д., что значительно упрощает создание пользовательского интерфейса.

Поддержка событий – WinForms обеспечивает поддержку событий, что позволяет разработчикам реагировать на действия пользователя, такие как нажатие кнопки или ввод текста.

Высокая скорость работы – WinForms работает очень быстро благодаря использованию низкоуровневого кода и оптимизации производительности.

Удобство разработки – WinForms поддерживает языки программирования C# и VB.NET, которые позволяют разработчикам быстро создавать приложения.

Широкая поддержка - WinForms является широко используемой технологией, поэтому для нее доступно множество ресурсов и инструментов разработки.

Поддержка встроенных библиотек – WinForms обеспечивает поддержку встроенных библиотек, таких как ADO.NET и ASP.NET, что позволяет разработчикам легко создавать приложения, работающие с базами данных и сетью.

Минусы WinForms:

Ограниченный функционал – WinForms не предоставляет таких возможностей, как многопоточность или асинхронный код, что может быть необходимо для некоторых типов приложений.

Не поддерживает кроссплатформенность – WinForms разработана для работы только на операционной системе Windows, что может быть проблемой, если необходимо создавать приложения для других операционных систем.

Ограниченные возможности визуального дизайна – WinForms обеспечивает ограниченные возможности для визуального дизайна пользовательского интерфейса, что может быть проблемой для некоторых разработчиков.

Не поддерживает новые технологии – WinForms не поддерживает новые технологии, такие как HTML5 или CSS3, что может быть проблемой для разработчиков, которые хотят создавать современные веб-приложения.



Рисунок 5 – Логотип платформы WinForms для разработки

## СРЕДА РАЗРАБОТКИ VISUAL STUDIO

Visual Studio – это интегрированная среда разработки (IDE) от Microsoft, предназначенная для создания приложений для платформы Windows, веб-приложений, мобильных приложений и других приложений на основе .NET Framework. В этом ответе мы рассмотрим плюсы и минусы Visual Studio.

Плюсы:

Интуитивный интерфейс пользователя: Visual Studio имеет простой и интуитивно понятный интерфейс, который позволяет программистам легко находить и использовать различные функции.

Широкие возможности для разработки приложений: Visual Studio предлагает различные средства разработки, включая C++, C#, Visual Basic и другие. Кроме того, она также предлагает возможность разработки приложений для платформы Windows, веб-приложений, мобильных приложений и других приложений на основе .NET Framework.

Поддержка систем контроля версий: Visual Studio обеспечивает интеграцию с системами контроля версий, такими как Git, SVN и TFS. Это позволяет программистам управлять и контролировать версии своего кода, что важно для совместной работы над проектами.

Инструменты тестирования: Visual Studio предлагает широкий спектр инструментов тестирования, включая модульные тесты, тесты производительности и тесты нагрузки. Это позволяет разработчикам легко проверять и отлаживать свой код.

Расширяемость: Visual Studio имеет открытый API, который позволяет программистам создавать собственные расширения и добавлять их в среду разработки. Это дает возможность настраивать среду разработки под собственные нужды.

Минусы:

Тяжеловесность: Visual Studio требует значительных ресурсов компьютера для работы, что может замедлять процесс разработки. Большое количество функций и инструментов, которые предлагает Visual Studio, может быть избыточным для некоторых пользователей.

Стоимость: Visual Studio может быть довольно дорогой, особенно для небольших компаний и независимых разработчиков.

Сложность: Visual Studio может быть сложной для новичков, особенно для тех, кто только начинает изучать программирование. Некоторые функции и инструменты могут быть трудными для понимания и использования.

Недоступность для других операционных систем: Visual Studio доступна только для операционной системы Windows, что может быть проблемой.



Рисунок 6 – Логотип среды разработки Visual Studio

## СРЕДА РАЗРАБОТКИ JETBRAINS RIDER

JetBrains Rider - это среда разработки, созданная компанией JetBrains, которая изначально была ориентирована на разработку приложений для платформы .NET и языков программирования, таких как C#, VB.NET, F# и т.д. В последнее время среда также поддерживает разработку на других языках программирования, таких как JavaScript, TypeScript, HTML и CSS.

* Плюсы среды разработки JetBrains Rider:
* Одной из главных преимуществ Rider является её мощная интегрированная система управления версиями (VCS). Rider поддерживает множество систем управления версиями, таких как Git, Subversion и Mercurial, а также предоставляет удобный интерфейс для работы с ними.
* Rider предлагает продвинутые инструменты для отладки кода, включая поддержку отладки удаленного кода, использование точек останова и возможность пошагового выполнения кода.
* В Rider представлены удобные инструменты для автоматического тестирования кода, включая интеграцию с NUnit, xUnit и MSTest.
* Среда предлагает широкие возможности для рефакторинга кода, такие как переименование, выделение методов, изменение сигнатуры метода и многие другие.
* Rider обладает мощным механизмом анализа кода, который позволяет выявлять потенциальные ошибки и проблемы еще до компиляции кода.
* Один из ключевых плюсов Rider - это его быстродействие и отзывчивость. Разработчики могут эффективно работать с большими проектами без замедления работы среды.

Минусы среды разработки JetBrains Rider:

* Одним из минусов Rider является высокая стоимость, особенно для индивидуальных разработчиков или малых компаний.
* Еще одним недостатком Rider является его большой размер, который может затруднить установку и использование на маломощных компьютерах.
* Несмотря на широкие возможности для настройки среды, управление её конфигурацией может быть сложным и требовать времени для освоения.

В целом, среда разработки JetBrains Rider - это мощный и продвинутый инструмент для разработки приложений на платформе .NET и других языках программирования. Его многочисленные функции и инструменты позволяют разработчикам быстро и эффективно создавать программы.



Рисунок 7 – Логотип среды разработки Visual Studio

## ОБОСНОВАНИЯ НЕОБХОДИМОСТИ РАЗРАБОТКИ ВНУТРЕННЕГО ПО ДЛЯ ТЕСТИРОВАНИЯ СОТРУДНИКОВ КОМПАНИИ

Разработка нового программного обеспечения для тестирования сотрудников и прохождения курсов является необходимой задачей для многих компаний. Существует несколько причин, почему компании могут решить разработать свою собственную систему тестирования и обучения сотрудников.

* Нестандартные требования – каждая компания имеет свои уникальные требования к системе тестирования и обучения сотрудников, которые могут отличаться от стандартных решений, доступных на рынке. Например, компания может иметь специфические бизнес-процессы или требования к обучению сотрудников, которые не учитываются в стандартных системах.
* Поддержка бренда – компания может заинтересована в создании своего бренда и идентичности для обучения сотрудников. Разработка собственной системы тестирования и обучения может помочь компании усилить свой бренд и создать единый стиль для обучения сотрудников.
* Контроль данных – компания может хотеть иметь полный контроль над данными своих сотрудников. Разработка собственной системы тестирования и обучения позволяет компании контролировать доступ к данным и обеспечить их безопасность.
* Гибкость – разработка собственной системы тестирования и обучения позволяет компании создавать гибкие и настраиваемые решения, которые отвечают конкретным потребностям компании.
* Экономия – разработка собственной системы тестирования и обучения может оказаться более экономически выгодным решением, чем покупка готового решения. Кроме того, компания может сохранить деньги, не платя за лицензии и обновления сторонних продуктов.
* Интеграция – разработка собственной системы тестирования и обучения позволяет компании интегрировать ее с другими системами, используемыми в компании, например, системами управления персоналом, CRM-системами и т.д.
* Адаптация к изменениям – разработка собственной системы тестирования и обучения позволяет компании быстро реагировать на изменения внутри компании и изменения в бизнес-среде.

## ТРЕБОВАНИЯ К ПРОЕКТУ

Требования к надежности:

Надежность системы для тестирования сотрудников и прохождения курсов является одним из важнейших факторов, которые необходимо учитывать при ее разработке. Низкая надежность системы может привести к потере данных, сбоям в работе приложения или даже к разрушению всей системы. Поэтому необходимо учитывать следующие требования к надежности системы.

* Безопасность данных – система должна обеспечивать защиту данных, хранящихся в ней. Это включает в себя защиту от несанкционированного доступа, взломов и кражи данных. Для этого необходимо использовать механизмы шифрования, контроль доступа и многоуровневую аутентификацию.
* Резервное копирование данных – система должна регулярно создавать резервные копии данных, чтобы в случае сбоя или потери данных можно было бы их восстановить.
* Масштабируемость – система должна быть способна обрабатывать большое количество пользователей и запросов, особенно в периоды пиковой нагрузки. Для этого необходимо использовать масштабируемые архитектурные решения, такие как облачные сервисы.
* Отказоустойчивость – система должна быть способна обеспечивать работоспособность даже при сбоях в работе отдельных компонентов. Для этого необходимо использовать механизмы резервирования, балансировки нагрузки и автоматического восстановления после сбоев.
* Стабильность – система должна быть стабильной и надежной в работе, чтобы пользователи могли использовать ее без сбоев и ошибок. Для этого необходимо проводить тщательное тестирование системы перед ее запуском и регулярно проводить обновления и исправления ошибок.
* Доступность – система должна быть доступна для пользователей в любое время суток, поскольку многие пользователи могут использовать ее в свободное время или в ночное время. Для этого необходимо обеспечивать высокую доступность системы и минимальное время простоя в случае сбоев или обновлений.
* Удобство использования – система должна быть простой в использовании и интуитивно понятной для пользователей.

Требования к функциональным характеристикам:

Функциональные характеристики системы для тестирования сотрудников и прохождения курсов играют ключевую роль в обеспечении эффективности процесса обучения и оценки знаний сотрудников. Для того, чтобы система была полезной и удобной для пользователей, необходимо учитывать следующие требования к ее функциональным характеристикам.

* Управление курсами и контентом – система должна обеспечивать удобное управление курсами и контентом. Администраторы должны иметь возможность создавать и редактировать курсы, добавлять контент, обновлять их и определять доступ к ним.
* Разнообразие контента – система должна поддерживать разнообразный контент, включая видео, аудио, документы, изображения и интерактивные задания. Это позволит сотрудникам получать знания различными способами и в соответствии со своими потребностями.
* Оценка знаний – система должна иметь механизмы для оценки знаний сотрудников, такие как тесты и задания. Администраторы должны иметь возможность создавать и редактировать тесты, а также просматривать результаты прохождения тестов.
* Система обратной связи – система должна иметь механизмы для обеспечения обратной связи сотрудникам, чтобы они могли получать поддержку и помощь при необходимости. Это может быть реализовано через систему онлайн-чата или электронную почту.
* Мониторинг процесса обучения – система должна обеспечивать мониторинг процесса обучения, чтобы администраторы могли следить за прогрессом сотрудников и оценивать их успехи. Для этого необходимо иметь возможность просмотра статистики и отчетов о прохождении курсов и тестов.

Гибкость и настраиваемость – система должна быть гибкой и настраиваемой, чтобы администраторы могли адаптировать ее под свои потребности. Например, система должна позволять настраивать права доступа и уровень сложности заданий.

Требования к целостности данных:

Целостность данных является одним из самых важных аспектов для системы тестирования сотрудников и прохождения курсов. Целостность данных обеспечивает правильность и точность информации, которую используют сотрудники для обучения и оценки своих знаний. Вот несколько требований к целостности данных, которые должны быть учтены при разработке системы тестирования сотрудников и прохождения курсов.

* Надежность системы хранения данных – система должна иметь надежную систему хранения данных, которая обеспечивает сохранность информации и защиту от несанкционированного доступа. Для этого необходимо использовать современные методы шифрования, резервное копирование данных и автоматическую проверку целостности данных.
* Защита данных от несанкционированного доступа – система должна иметь механизмы для защиты данных от несанкционированного доступа. Это может быть достигнуто через использование паролей, аутентификации по IP-адресу, двухфакторной аутентификации и других методов.
* Ограничение прав доступа – система должна иметь возможность ограничивать права доступа к данным в соответствии с ролями пользователей. Например, администраторы должны иметь больше прав, чем обычные пользователи.
* Ограничение возможности изменения данных – система должна иметь механизмы для ограничения возможности изменения данных пользователем. Например, администраторы должны иметь возможность редактировать курсы и тесты, но не результаты тестирования.
* Мониторинг целостности данных – система должна иметь механизмы для мониторинга целостности данных. Например, система должна предупреждать об изменениях данных и отправлять уведомления администраторам.
* Резервное копирование данных – система должна иметь механизмы для резервного копирования данных, чтобы обеспечить сохранность информации в случае сбоя системы или несанкционированного доступа.
* Аудит доступа к данным – система должна иметь возможность аудита доступа к данным, чтобы администраторы могли просматривать журналы доступа и проверять, кто и когда получал доступ к данным.

# ГЛАВА 2. РАЗРАБОТКА ИНФОРМАЦИОННОГО РЕСУРСА ДЛЯ ОБУЧЕНИЯ И ДИСТАНЦИОННОГО ТЕСТИРОВАНИЯ СОТРУДНИКОВ.

## 2.1 ПЛАН РАЗРАБОТКИ

1) Планирование

* Определение целей и задач приложения
* Определение целевой аудитории
* Определение требований к функциональности и дизайну

2) Анализ

* Изучение существующих разработок и программных решений на рынке
* Анализ потребностей и ожиданий целевой аудитории
* Определение функциональных и технических требований

3) Проектирование

* Разработка структуры и функциональности приложения
* Разработка дизайна и интерфейса пользователя
* Определение технических решений и выбор используемых технологий

4) Разработка

* Написание кода и реализация функциональности

5) Развертывание и запуск

* Установка и настройка приложения
* Тестирование работоспособности и безопасности
* Запуск информационного ресурса

В процессе выполнения каждого этапа необходимо проводить регулярную проверку соответствия реализованных решений поставленным задачам и требованиям, а также корректировать план, если необходимо.

Важно также учитывать требования безопасности при разработке приложения, включая защиту персональных данных пользователей и предотвращение несанкционированного доступа к системе.



## ПЛАНИРОВАНИЕ

Основной задачей данного приложения является повышение эффективности ускоренного обучения и тестирования сотрудников.

Основные требования для разработки:

* Одним из главных требований являлась автономность продукта. Необходимо было создать программу, вероятность выхода из строя которой была сведена к минимуму;
* Возможность проходить учебные курсы;
* Возможность создавать учебные тесты;
* Возможность проходить учебные тесты;
* Функция перемешивания вопросов теста;
* Функция просмотра и сохранения результата

## АНАЛИЗ

Стадия анализа является одной из важнейших для успешной разработки приложения для обучения и дистанционного тестирования сотрудников. В рамках этой стадии были выполнены следующие шаги:

* Определение используемых средств для разработки. Для успешной разработки данного продукта было решено выбрать язык программирования C# и платформу WPF. Выбор обусловлен тем, что данные средства разработки обеспечат высокую скорость работы из-за простоты использования. Также наличие удобных для всех целей продукта библиотек, которые оптимизируют процесс написания кода и помогут создать качественный продукт.
* Анализ потребностей пользователей. Были определены группы пользователей, которые будут использовать приложение для обучения и дистанционного тестирования.
* Определение требований к приложению. На основе проведенного анализа были выбраны функциональные и нефункциональные требования к информационному ресурсу, а также определены требования к дизайну и интерфейсу пользователя.
* Составление технического задания. На основе определенных требований составлено техническое задание на разработку информационного ресурса для обучения и дистанционного тестирования сотрудников. В техническом задании четко определены требования к функциональности и дизайну информационного ресурса.

В результате выполнения стадии анализа было получено техническое задание на разработку информационного ресурса для обучения и дистанционного тестирования сотрудников, которое будет служить основой для следующих этапов разработки.

## ПРОЕКТИРОВАНИЕ

Стадия проектирования приложения для обучения и дистанционного тестирования сотрудников является важнейшей для разработки полноценного и функционального продукта. В рамках этой стадии необходимо выполнить следующие шаги:

Структура информационного ресурса:

1. Версия для модераторов
   1. Главное меню
   2. Выбор курсов
      1. Кнопка открытия курса
      2. Кнопки приближения и отдаления текста
   3. Выбор тестов
      1. Название теста
      2. Количество вопросов теста
      3. Кнопка начать тест
         1. Возможность выбора интервала вопросов
         2. Возможность перемешивать вопросы
         3. Кнопка включения режима проверки ответов
      4. Кнопка изменить тест
      5. Кнопка удалить тест
   4. Загрузка тестов
   5. Прохождение теста
      1. Кнопка для проверки правильности ответа на вопрос
      2. Кнопка для переключения на следующий вопрос
      3. Кнопка просмотра результатов теста
         1. Шкала с процентом правильных ответов
         2. Кнопка для просмотра верных ответов
         3. Кнопка для просмотра неверных ответов
         4. Кнопка для сохранения результатов тестирования
         5. Кнопка для выхода в главное меню
   6. Создание тестов
      1. Кнопка добавления новых вопросов
      2. Кнопка добавления правильных ответов на вопросы
      3. Кнопки удаления вопроса или ответа
      4. Кнопка сохранения теста
      5. Кнопка удаления теста
      6. Строка для ввода названия теста
   7. Кнопка выхода
   8. Кнопка возвращения на предыдущую страницу
2. Версия для пользователей
   1. Главное меню
   2. Выбор курсов
      1. Кнопка открытия курса
      2. Кнопки приближения и отдаления текста
   3. Выбор тестов
      1. Название теста
      2. Количество вопросов теста
      3. Кнопка начать тест
         1. Возможность выбора интервала вопросов
         2. Возможность перемешивать вопросы
         3. Кнопка включения режима проверки ответов
      4. Кнопка удалить тест
   4. Загрузка тестов
   5. Прохождение теста
      1. Кнопка для проверки правильности ответа на вопрос
      2. Кнопка для переключения на следующий вопрос
      3. Кнопка просмотра результатов теста
         1. Шкала с процентом правильных ответов
         2. Кнопка для просмотра верных ответов
         3. Кнопка для просмотра неверных ответов
         4. Кнопка для сохранения результатов тестирования
         5. Кнопка для выхода в главное меню
   6. Кнопка выхода
   7. Кнопка возвращения на предыдущую страницу

Разработка дизайна и интерфейса пользователя. Необходимо разработать дизайн и интерфейс пользователя, который будет максимально удобным для использования пользователем. Важно учитывать требования к дизайну и интерфейсу, определенные на стадии анализа, а также требования к удобству использования и эргономике.

Версия для пользователей.

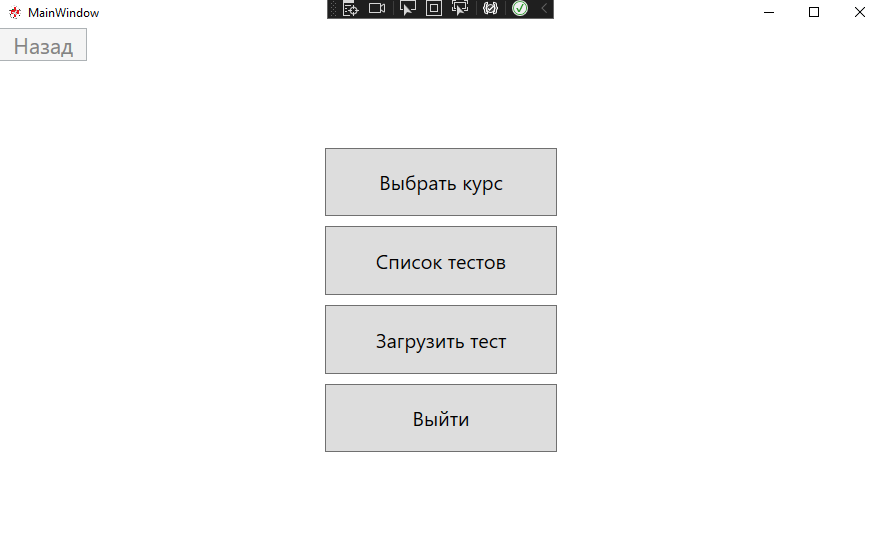


Рисунок 8 – Главное меню

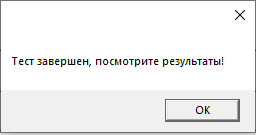


Рисунок 9 – Уведомление о завершении тестирования

Версия для модераторов:

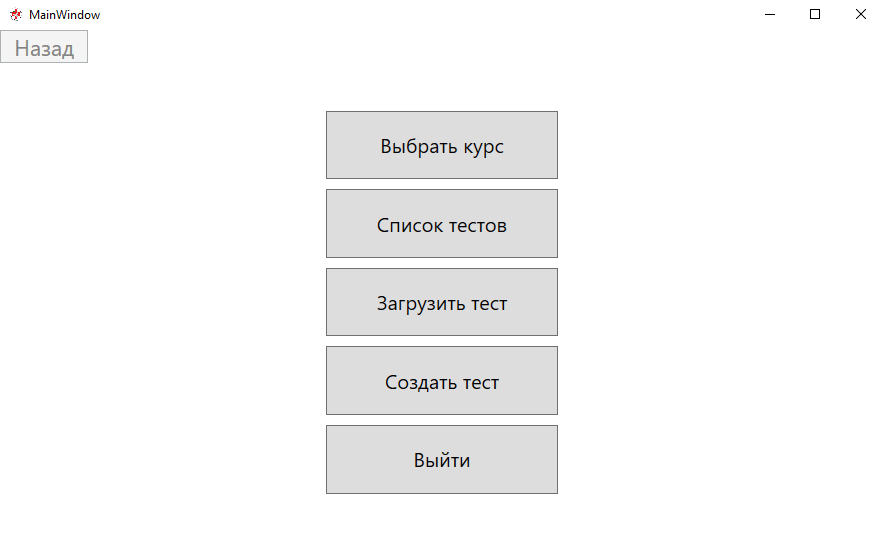


Рисунок 10 – Главное меню версии для модераторов

В результате выполнения стадии проектирования должен быть получен детальный проект информационного ресурса для обучения и дистанционного тестирования сотрудников, включающий структуру, дизайн и интерфейс пользователя, функциональность, систему тестирования и систему управления курсами. Этот проект будет служить основой для следующей стадии - разработки и реализации информационного ресурса.

## РАЗРАБОТКА

После полной разработки интерфейса программы и его утверждения у заказчика, началась разработка функциональной части проекта.

Структура проекта в основном поделена на 3 части:

* Страница
* Класс с интерфейсом страницы
* Класс с функционалом страницы

Данная структура позволяет писать чистый код, в котором легко разобраться стороннему специалисту.

В проекте активно использовался паттерн MVVM, для создания качественной файловой системы и системы взаимодействия бэкенда и фронтенда проекта.

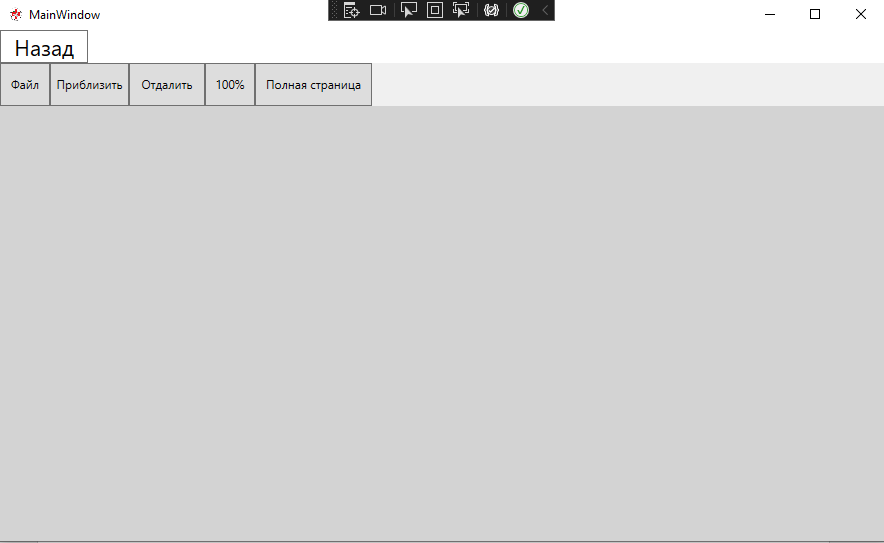


Рисунок 11 – Выбор курса

Страница выбора курса. Функция выбора необходимого для прохождения курса была реализована с помощью библиотеки MoonPdf. Код страницы просмотра курса выглядит следующим образом:

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

{

var dialog = new OpenFileDialog();

if (dialog.ShowDialog().GetValueOrDefault())

{

string filePath = dialog.FileName;

try

{

moonPdfPanel.OpenFile(filePath);

\_isLoaded = true;

}

catch (Exception)

{

\_isLoaded = false;

}

}

}

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

{

if (\_isLoaded)

{

moonPdfPanel.ZoomIn();

}

}

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

{

if (\_isLoaded)

{

moonPdfPanel.ZoomOut();

}

}

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

{

if (\_isLoaded)

{

moonPdfPanel.Zoom(1.0);

}

}

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

{

moonPdfPanel.ZoomToHeight();

}

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

{

moonPdfPanel.ViewType = MoonPdfLib.ViewType.Facing;

}

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

{

moonPdfPanel.ViewType = MoonPdfLib.ViewType.SinglePage;

}

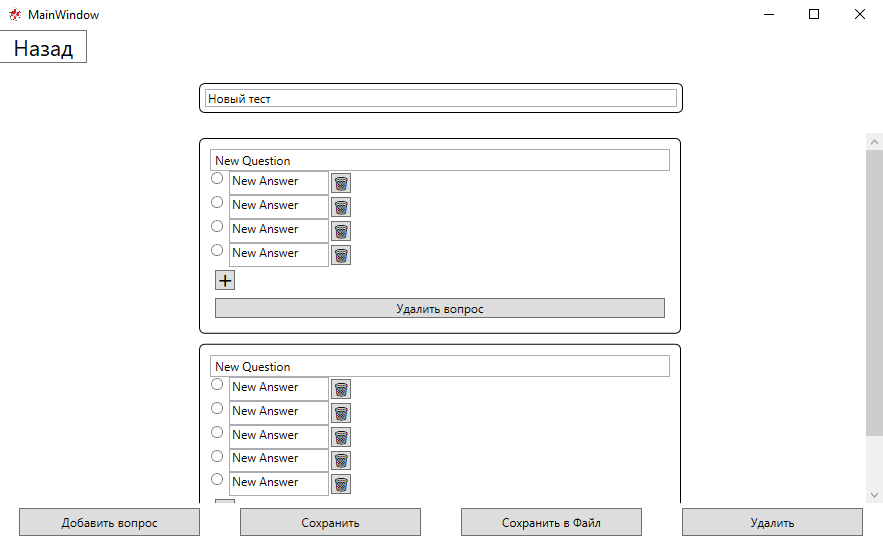


Рисунок 12 – Меню создания тестирования

Страница создания тестов. Во время создания теста модератору необходимо добавлять нужное для него количество вопросов и ответов, поэтому код для динамического добавления новых вопросов реализован следующим образом:

private readonly TestService \_testService;

private readonly TestState \_testState;

public Test Test { get; set; }

public CreateTestPageViewModel(TestService testService, TestState testState)

{

\_testService = testService;

\_testState = testState;

Test = \_testState.CurrentTest ?? new Test { Name = "Новый тест", Questions = new ObservableCollection<Question>() };

}

public ICommand SaveTestCommand => new DelegateCommand(() =>

{

\_testService.SaveTest(Test);

});

public ICommand DeleteTestCommand => new DelegateCommand(() =>

{

\_testService.DeleteTest(Test.Id);

});

public ICommand AddQuestionCommand => new DelegateCommand(() =>

{

var question = new Question

{

Name = "New Question"

};

Test.Questions.Add(question);

});

public ICommand DeleteQuestionCommand => new DelegateCommand<Guid>(id =>

{

var obj = Test.Questions.FirstOrDefault(x => x.Id == id);

Test.Questions.Remove(obj);

});

public ICommand DeleteAnswerCommand => new DelegateCommand<Guid>(id =>

{

foreach (var question in Test.Questions)

{

foreach (var answer in question.Answers)

{

if (answer.Id == id)

{

question.Answers.Remove(answer);

break;

}

}

}

});

public ICommand AddAnswerCommand => new DelegateCommand<Guid>((questionId) =>

{

var question = Test.Questions.FirstOrDefault(x => x.Id == questionId);

var answer = new Answer

{

Name = "New Answer"

};

question.Answers.Add(answer);

});

public ICommand SaveTestToFileCommand => new DelegateCommand(() =>

{

\_testService.SaveTestToFile(Test);

});

Данный код отвечает за внешний вид страницы создания тестов. Представленные интерфейсы ссылаются на отдельные классы, где находится нужная функция.

private readonly DialogService \_dialogService;

private readonly CustomFileService \_customFileService;

private readonly PageNavigationService \_navigationService;

private readonly Repository \_repository;

public TestService(DialogService dialogService, CustomFileService customFileService, PageNavigationService navigationService, Repository repository)

{

\_dialogService = dialogService;

\_customFileService = customFileService;

\_navigationService = navigationService;

\_repository = repository;

}

public Test GetTest(Guid testId)

{

return \_repository.FindAll<Test>().FirstOrDefault(x => x.Id == testId);

}

public void LoadTest()

{

if (\_dialogService.OpenFileDialog(out string filepath))

{

try

{

LoadTestFromFile(filepath);

\_dialogService.ShowMessage("Тест загружен успешно!");

}

catch(Exception e)

{

\_dialogService.ShowMessage(e.Message);

}

}

else

{

\_dialogService.ShowMessage("Неверный путь к файлу");

}

}

public void SaveTest(Test test)

{

test.CreationDate = DateTime.Now;

\_repository.Save(test);

\_dialogService.ShowMessage("Тест сохранен!");

\_navigationService.GoBack();

}

public void SaveTestToFile(Test test)

{

if (\_dialogService.SaveFileDialog(out string filepath))

{

try

{

\_customFileService.WriteTestToFile(filepath, test);

\_dialogService.ShowMessage("Тест сохранен успешно!");

}

catch (Exception e)

{

\_dialogService.ShowMessage(e.Message);

}

}

else

{

\_dialogService.ShowMessage("Неверный путь к файлу");

}

}

public void DeleteTest(Guid testId)

{

\_repository.Delete<Test>(testId);

\_dialogService.ShowMessage("Тест удален!");

\_navigationService.GoBack();

}

В одном классе находится множество функций. Например, на код выше в основном ссылаются интерфейсы для окна создания тестов. На код ниже ссылаются интерфейсы для окна загрузки тестов.

private void LoadTestFromFile(string filepath)

{

var test = \_customFileService.ReadTestFromFile(filepath);

if (\_repository.FindAll<Test>().Any(x => x.Id == test.Id))

{

test.Id = Guid.NewGuid();

test.CreationDate = DateTime.Now;

}

\_repository.Save(test);

}

Для работы с тестами было решено сохранять их в виде файла формата json. Когда модератор сохраняет тест, то он сериализуется в JSON файл и сохраняется в выбранном пути. Соответственно для открытия теста выбранный файл десериализуется и считывается.

public virtual Test ReadTestFromFile(string filepath)

{

var test = new Test();

if (File.Exists(filepath))

{

test = JsonConvert.DeserializeObject<Test>(File.ReadAllText(filepath));

}

return test;

}

public virtual Test WriteTestToFile(string filepath, Test test)

{

File.WriteAllText(filepath, JsonConvert.SerializeObject(test, Formatting.Indented));

return test;

}

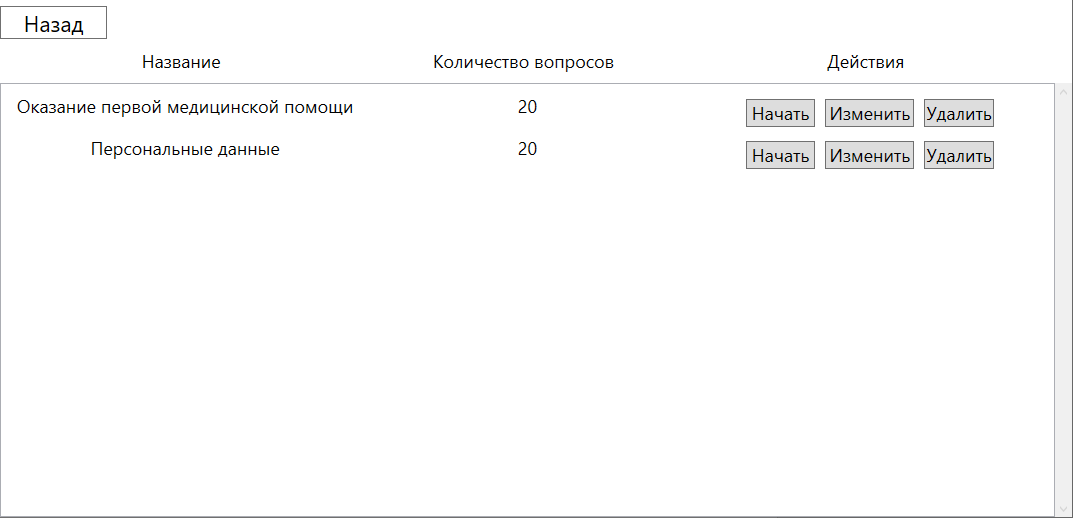


Рисунок 13 – Выбор теста

Окно выбора теста.

private readonly PageNavigationService \_navigation;

private readonly Repository \_repository;

private readonly TestState \_testState;

public ObservableCollection<Test> Tests { get; set; }

private Test selectedTest;

public Test SelectedTest

{

get => selectedTest;

set

{

selectedTest = value;

\_testState.CurrentTest = SelectedTest;

ShowTestDialog();

}

}

public TestListPageViewModel(PageNavigationService navigationService, Repository repository, TestState testState)

{

\_navigation = navigationService;

\_repository = repository;

\_testState = testState;

Tests = new ObservableCollection<Test>(\_repository.FindAll<Test>().OrderByDescending(x => x.CreationDate));

}

public ICommand OpenTestCommand => new DelegateCommand<Guid>((id) =>

{

\_testState.CurrentTest = Tests.FirstOrDefault(x => x.Id == id);

ShowTestDialog();

});

public ICommand EditTestCommand => new DelegateCommand<Guid>((id) =>

{

\_testState.CurrentTest = Tests.FirstOrDefault(x => x.Id == id);

\_navigation.Navigate(new CreateTestPage());

});

public ICommand DeleteTestCommand => new DelegateCommand<Guid>((id) =>

{

\_repository.Delete<Test>(id);

Tests.Remove(Tests.FirstOrDefault(x => x.Id == id));

});

private void ShowTestDialog()

{

Views.TestDialogWindow dialogWindow = new Views.TestDialogWindow();

if (dialogWindow.ShowDialog() == true)

{

\_navigation.Navigate(new TestPage());

}

}

Интерфейс данного окна позволяет отображать большое количество загруженных тестов. Интерфейсы функциональных кнопок так же ссылаются на классы, где прописана сама функция кнопки.

При попытке запуска тестирования нам выводится окно настройки параметров.

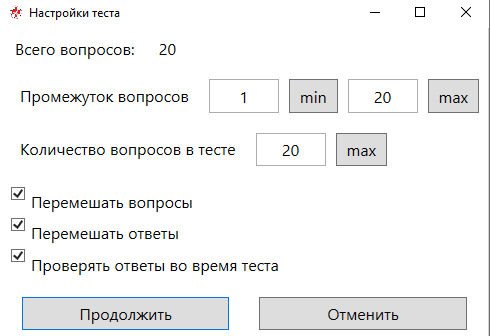
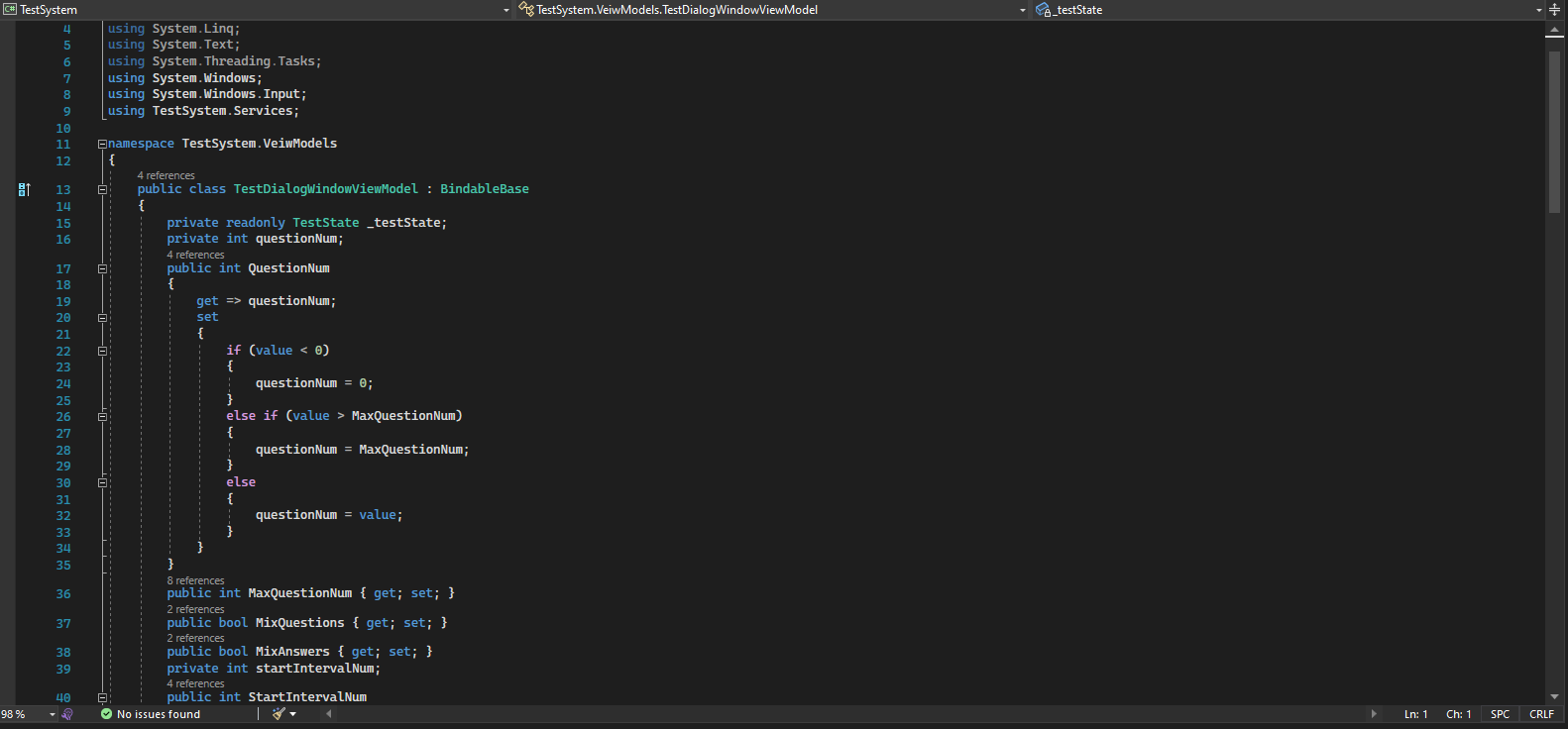
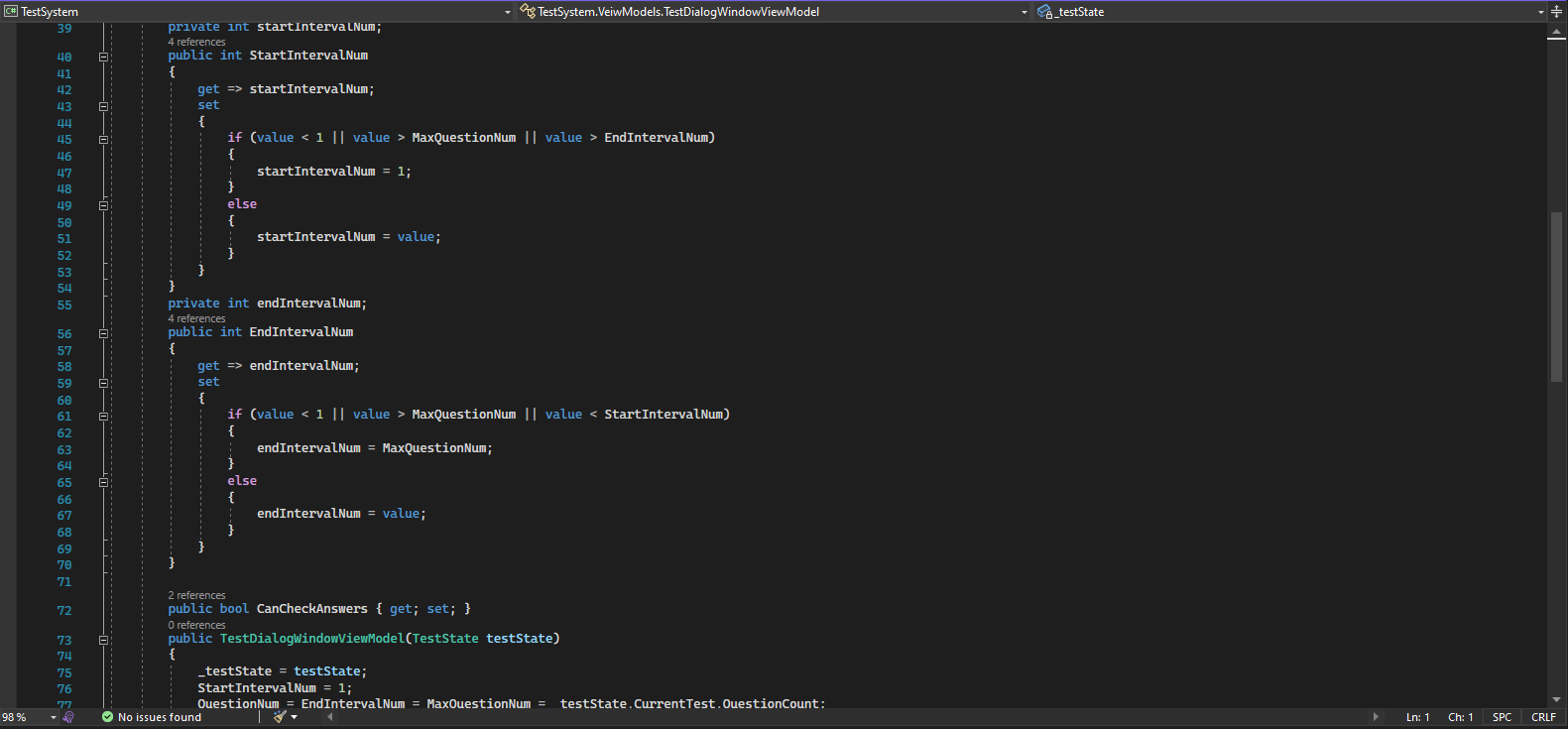
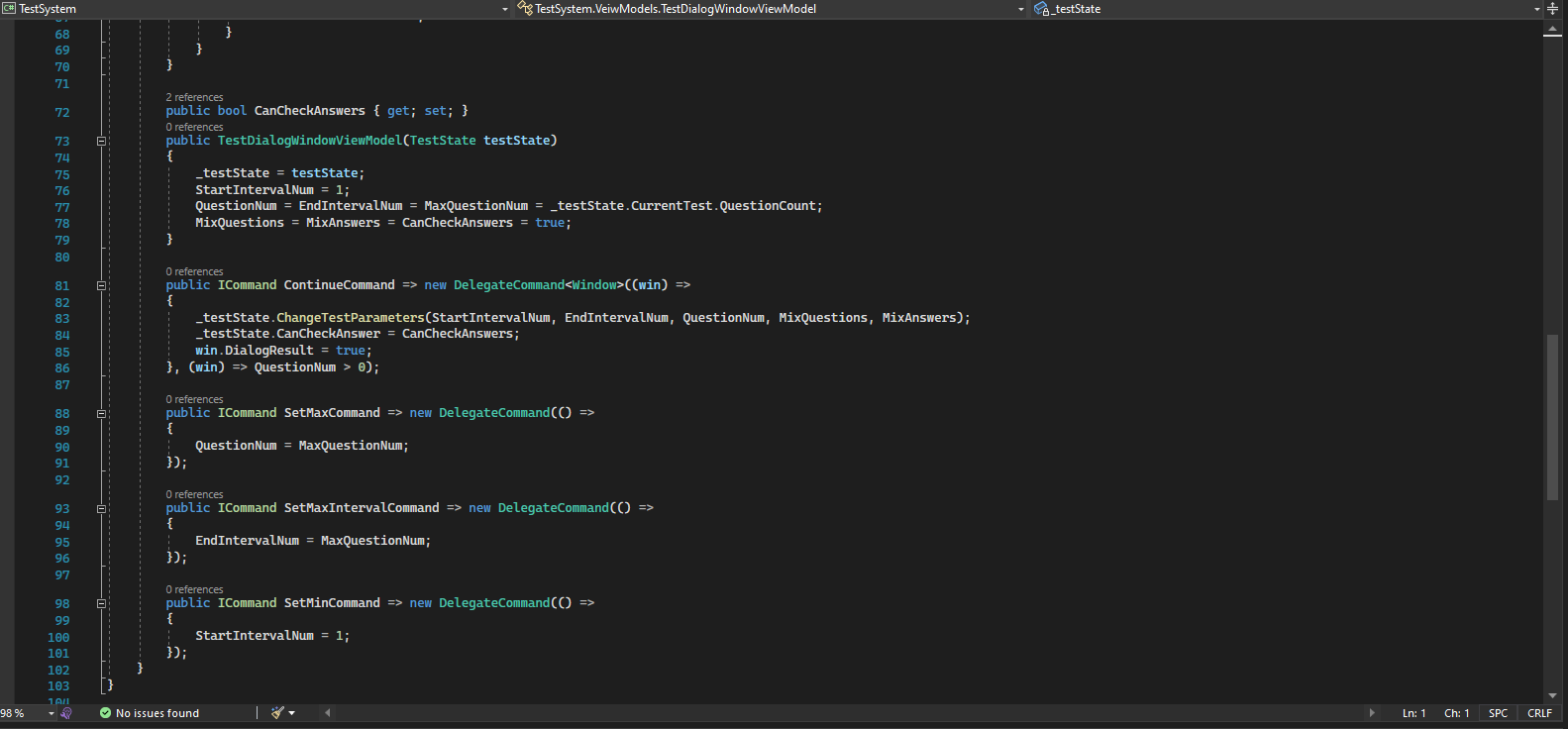


Рисунок 14 – Настройка теста

Код данного окна находится ниже:





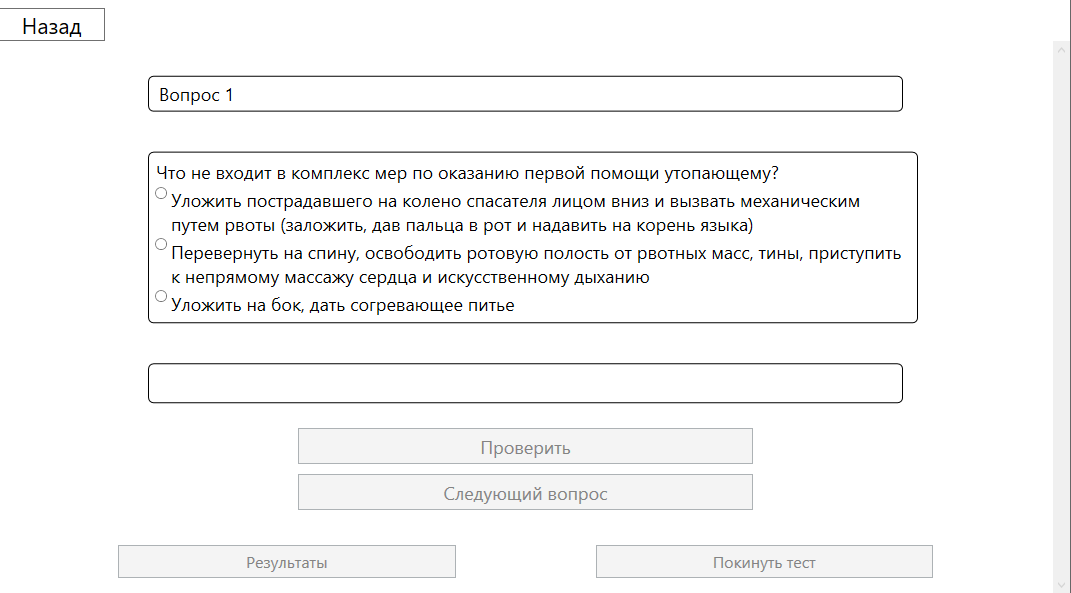


Рисунок 15 – Прохождение теста

После настройки теста открывается окно самого тестирования.

private readonly TestState \_testState;

private readonly PageNavigationService \_navigationService;

private readonly DialogService \_dialogService;

public Test Test { get; set; }

public int QuestionNumber => CurrentQuestionNumber + 1;

public int CurrentQuestionNumber { get; set; }

public Answer CorrectAnswer { get; set; }

public SolidColorBrush BorderColor { get; set; }

public bool IsEnabled { get; set; }

public bool CanCheckAnswer { get; set; }

public bool ShowPreviousButton { get; set; }

private List<Question> \_questions;

private bool \_testCompleted = false;

public Question CurrentQuestion

{

get

{

if (CurrentQuestionNumber >= \_questions.Count())

{

return \_questions.Last();

}

return \_questions[CurrentQuestionNumber];

}

}

public TestPageViewModel(TestState testState, PageNavigationService navigationService, DialogService dialogService)

{

\_testState = testState;

\_navigationService = navigationService;

\_dialogService = dialogService;

Test = \_testState.CurrentTest;

CurrentQuestionNumber = 0;

BorderColor = new SolidColorBrush(Colors.Black);

IsEnabled = true;

CanCheckAnswer = \_testState.CanCheckAnswer;

ShowPreviousButton = !\_testState.CanCheckAnswer;

\_questions = new List<Question>(Test.Questions.Select(x => CopyQuestion(x)));

}

public ICommand CheckAnswerCommand => new DelegateCommand(() =>

{

IsEnabled = false;

var question = Test.Questions.First(x => x.Id == CurrentQuestion.Id);

if (Check())

{

BorderColor = new SolidColorBrush(Colors.Green);

}

else

{

BorderColor = new SolidColorBrush(Colors.Red);

}

CorrectAnswer = question.GetRightAnswer();

}, () => CanCheckAnswer && IsEnabled && IsAnswerSelected());

public ICommand NextQuestionCommand => new DelegateCommand(() =>

{

IsEnabled = true;

BorderColor = new SolidColorBrush(Colors.Black);

CorrectAnswer = null;

if (CurrentQuestionNumber >= Test.QuestionCount - 1 && !\_testCompleted)

{

if (CanCheckAnswer)

{

IsEnabled = false;

}

\_testCompleted = true;

\_dialogService.ShowMessage("Тест завершен, посмотрите результаты!");

}

if (CurrentQuestionNumber < Test.QuestionCount - 1)

{

CurrentQuestionNumber++;

}

}, () => IsAnswerSelected() && !(\_testCompleted && CurrentQuestionNumber >= Test.QuestionCount - 1));

public ICommand PreviousQuestionCommand => new DelegateCommand(() =>

{

IsEnabled = true;

CorrectAnswer = null;

CurrentQuestionNumber--;

}, () => CurrentQuestionNumber > 0);

public ICommand ShowResultsCommand => new DelegateCommand(() =>

{

CheckResult();

\_navigationService.Navigate(new StatisticsPage());

}, () => \_testCompleted );

public ICommand LeaveTestCommand => new DelegateCommand(() =>

{

\_navigationService.GoBack();

}, () => \_testCompleted);

private Question CopyQuestion(Question q)

{

var res = new Question

{

Id = q.Id,

Name = q.Name,

Answers = new ObservableCollection<Answer>()

};

foreach (var item in q.Answers)

{

res.Answers.Add(new Answer

{

Id = item.Id,

Name = item.Name,

IsTrue = false,

});

}

return res;

}

private bool Check(Question q)

{

var question = Test.Questions.First(x => x.Id == q.Id);

var isCorrect = question.CheckAnswer(q);

return isCorrect;

}

private bool Check() => Check(CurrentQuestion);

private bool IsAnswerSelected()

{

foreach (var item in CurrentQuestion.Answers)

{

if (item.IsTrue == true)

{

return true;

}

}

return false;

}

private void CheckResult()

{

\_testState.RightAnswers.Clear();

\_testState.WrongAnswers.Clear();

foreach (var question in \_questions)

{

if (Check(question))

{

\_testState.RightAnswers.Add(question);

}

else

{

\_testState.WrongAnswers.Add(question);

}

}

}

Данный код реализовывает возможность ответа на вопрос, просмотра результата ответа, переключения на следующий вопрос и просмотра результатов тестирования.

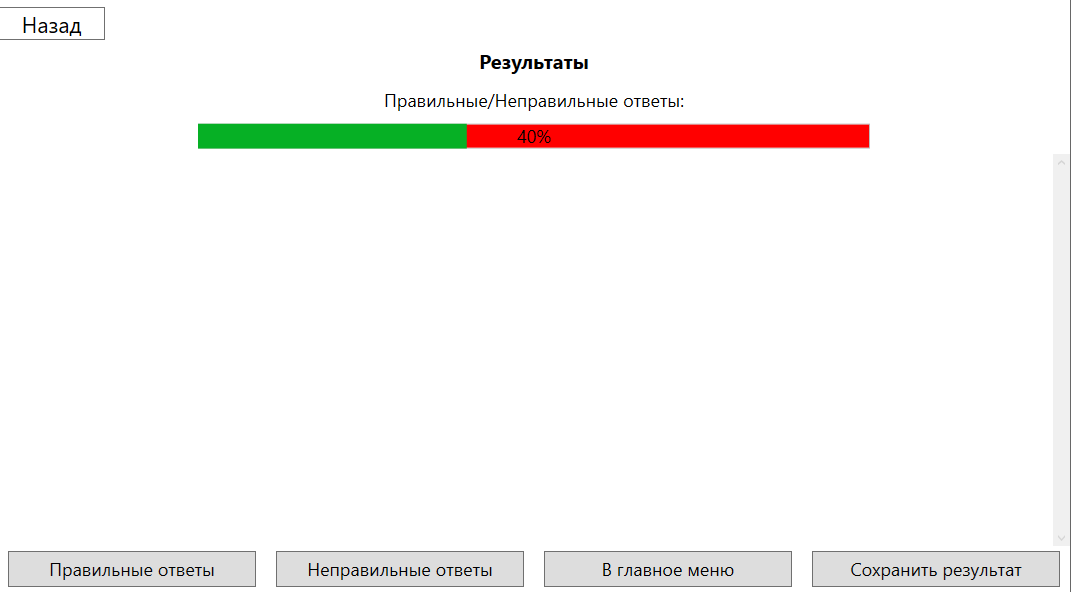


Рисунок 16 – Просмотр результатов теста

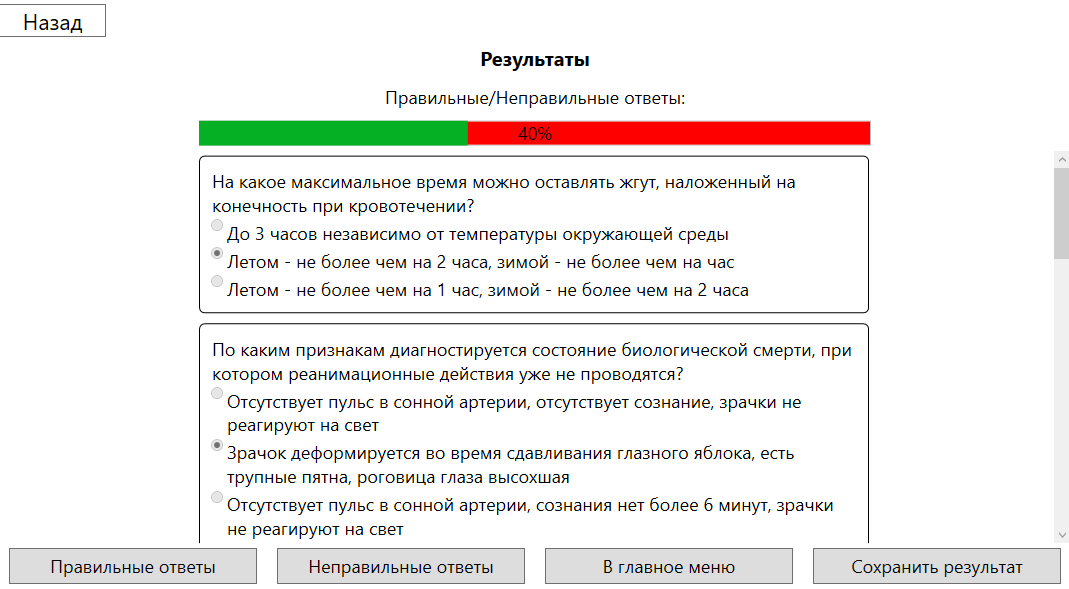


Рисунок 17 – Просмотр правильных ответов

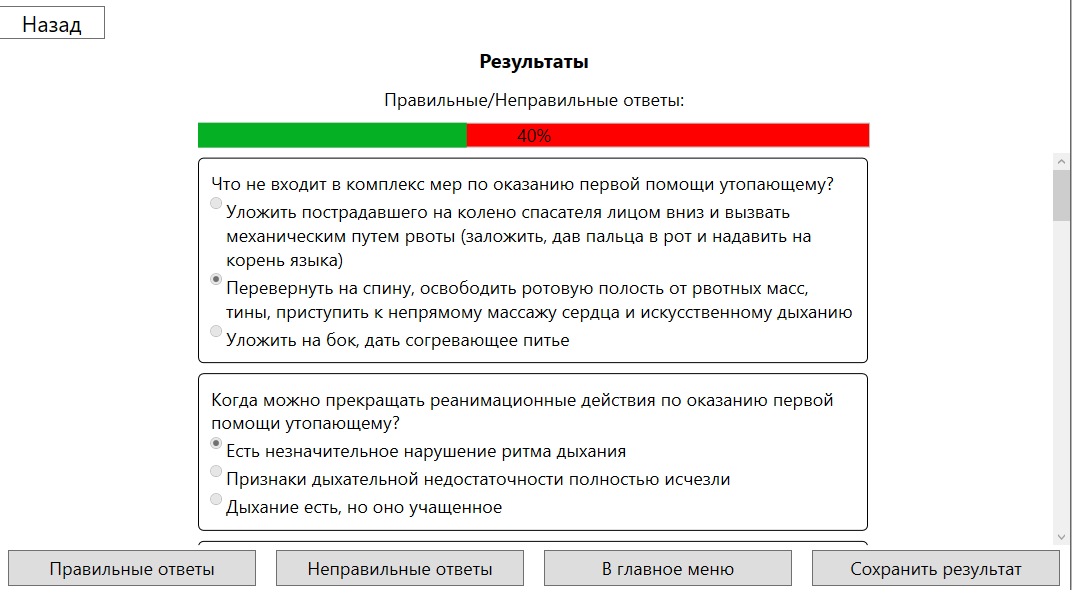


Рисунок 18 – Просмотр неправильных ответов

Внешний вид окна просмотра результатов тестирования.

private readonly TestState \_testState;

private readonly PageNavigationService \_navigationService;

public string Title { get; set; }

public int RightAnswersNum { get; set; }

public int WrongAnswersNum { get; set; }

public int Result { get; set; }

public ObservableCollection<Question> Questions { get; set; }

public StatisticsPageViewModel(TestState testState, PageNavigationService navigationService)

{

\_testState = testState;

\_navigationService = navigationService;

Questions = new ObservableCollection<Question>();

RightAnswersNum = \_testState.RightAnswers.Count;

WrongAnswersNum = \_testState.WrongAnswers.Count;

Result = \_testState.TestResult;

}

public ICommand ShowRightQuestions => new DelegateCommand(() =>

{

Title = "Правильные ответы";

Questions = \_testState.RightAnswers;

});

public ICommand ShowWrongQuestions => new DelegateCommand(() =>

{

Title = "Неправильные ответы";

Questions = \_testState.WrongAnswers;

});

public ICommand ToStartPage => new DelegateCommand(() =>

{

\_navigationService.ToStartPage();

});

Функционал кнопки просмотра результатов так же реализован в отдельном классе TestState.

private Test currentTest;

public Test CurrentTest

{

get => currentTest;

set

{

currentTest = value;

ClearState();

}

}

private int questionNum;

public int QuestionNum

{

get => questionNum;

set

{

questionNum = value > CurrentTest.QuestionCount ? CurrentTest.QuestionCount : value;

}

}

public ObservableCollection<Question> RightAnswers { get; set; }

public ObservableCollection<Question> WrongAnswers { get; set; }

public bool ShowRightAnswers { get; set; }

public int TestResult => (int)((double)RightAnswers.Count / CurrentTest.QuestionCount \* 100);

public bool CanCheckAnswer { get; set; }

private void ClearState()

{

RightAnswers = new ObservableCollection<Question>();

WrongAnswers = new ObservableCollection<Question>();

}

public void ChangeTestParameters(int start, int end, int qNum, bool mixQ, bool mixA)

{

SelectQuestionInterval(start, end);

if (mixQ)

{

CurrentTest.Questions.Shuffle();

}

if (mixA)

{

MixAnswers();

}

SelectQuestions(qNum);

}

private void SelectQuestionInterval(int start, int end)

{

CurrentTest.Questions = new ObservableCollection<Question>(CurrentTest.Questions.Skip(start - 1).Take(end - start + 1));

}

private void SelectQuestions(int num)

{

CurrentTest.Questions = new ObservableCollection<Question>(CurrentTest.Questions.Take(num));

}

private void MixAnswers()

{

foreach (var question in CurrentTest.Questions)

{

question.Answers.Shuffle();

}

}

Стоит отметить, что в данном классе в том числе есть функции для окна настройки тестирования.

Для комфортного взаимодействия с тестами (вопросами и ответами) созданы отдельные классы.

Класс Answer:

[JsonIgnore]

public Guid Id { get; set; }

public string Name { get; set; }

public bool IsTrue { get; set; }

public Answer()

{

Id = Guid.NewGuid();

}

public bool IsCorrectAnswer()

{

return !String.IsNullOrWhiteSpace(Name);

}

Класс Question:

[JsonIgnore]

public Guid Id { get; set; }

public string Name { get; set; }

public ObservableCollection<Answer> Answers { get; set; }

public Question()

{

Id = Guid.NewGuid();

Answers = new ObservableCollection<Answer>();

}

public bool CheckAnswer(Question question)

{

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

{

if (question.Answers[i].IsTrue != Answers[i].IsTrue)

{

return false;

}

}

return true;

}

public Answer GetRightAnswer()

{

Answer answer = null;

foreach (var item in Answers)

{

if (item.IsTrue)

{

answer = item;

break;

}

}

return answer ?? new Answer();

}

public bool IsCorrectQuestion()

{

if (String.IsNullOrWhiteSpace(Name))

{

return false;

}

if (Answers.Count == 0)

{

return false;

}

foreach (var item in Answers)

{

if (!item.IsCorrectAnswer())

{

return false;

}

if (item.IsTrue == true)

{

return true;

}

}

return false;

}

Класс Test:

[JsonIgnore]

public Guid Id { get; set; }

public string Name { get; set; }

[JsonIgnore]

public DateTime CreationDate { get; set; }

public ObservableCollection<Question> Questions { get; set; }

[JsonIgnore]

public int QuestionCount => Questions.Count;

public Test()

{

Id = Guid.NewGuid();

CreationDate = DateTime.Now;

Questions = new ObservableCollection<Question>();

}

public bool IsCorrectTest()

{

if (String.IsNullOrWhiteSpace(Name))

{

return false;

}

if (QuestionCount == 0)

{

return false;

}

foreach (var item in Questions)

{

if (!item.IsCorrectQuestion())

{

return false;

}

}

return true;

}

## РАЗВЕРТЫВАНИЕ И ЗАПУСК

После разработки проект был отправлен на утверждение заказчику. Заказчик утвердил программу и потребовал установить её на предоставленный компьютер.

Установка программы была проведена успешно. Все системы работали исправно. Однако возникла небольшая ошибка в распознавании картинок, поэтому было решено заменить некоторые изображения на текст.

# ГЛАВА 3. ЭКОНОМИЧЕСКАЯ ЧАСТЬ

**3.1. Цель и задачи экономического раздела**

В настоящем разделе выпускной квалификационной работы производится расчёт затрат на разработку информационного ресурса для обучения и дистанционного тестирования сотрудников.

Цели данного раздела:

* Расчёт себестоимости разработки информационного ресурса для обучения и дистанционного тестирования сотрудников
* Расчёт экономической эффективности информационного ресурса для обучения и дистанционного тестирования сотрудников

Для подсчета себестоимости проекта и экономической эффективности данного программного продукта, нужно знать следующие составляющие:

• расчет затрат на энергоресурсы;

• амортизационные отчисления;

• расчет фонда заработной платы;

• прочие (накладные) расходы

**3.2. Исходные данные для расчетов**

**Таблица 1 – Исходные данные**

|  |  |  |
| --- | --- | --- |
| **Показатели** | **Единицы измерения** | **Значение** |
| Страховые взносы (от заработной платы с учетом районного коэффициента), в том числе: | % | 30 |
| - отчисления в пенсионный фонд | % | 22 |
| - отчисления в фонд социального страхования | % | 2,9 |
| - отчисления в Федеральный и  Территориальный фонд обязательного медицинского страхования | % | 5,1 |
| Заработная плата специалиста (программиста .NET разработчика) | Руб./мес. | 120 400 |
| Фонд рабочего времени в 2023 году  при 40-часовой рабочей неделе\* | дней  часов | 247  1973 |
| Стоимость компьютера | Руб. | 50 000 |
| Срок полезного использования компьютера | лет | 5 |
| Амортизационные отчисления (исходя из срока полезного использования) | % | 20 |
| Потребляемая мощность компьютера (в час) | кВт | 0,45 |
| Тариф на электроэнергию | Руб./ кВт-час. | 5,92 |
| Затраты на текущий и профилактический ремонт компьютера, в том числе плата Интернет-провайдеру | Руб. | 2 000 |
| Затраты на расходные материалы, в том числе:  - оплата ПО необходимого для разработки (AdobeCreativeCloud (Учащимся и преподавателям), включающее AdobePhotoshop и AdobeIllustrator (ежемесячная оплата)) | Руб. | 10 000 |

*\* Всего в****2023 году****будет 365 дней, из которых 247****рабочих****и 118 выходных.****Рабочее****время****при 40****-****часовой неделе****составит 1 973 часов*

*\*\*Затраты=Стоимость OC\*Норму амортизации/12 месяцев*

**Таблица 2. Затраты на расходные материалы**

|  |  |  |
| --- | --- | --- |
| **Задача** | **Затраты** | **Ресурс** |
| оплата ПО необходимого для разработки (AdobeCreativeCloud), включающее AdobePhotoshop и AdobeIllustrator (ежемесячная оплата) | 10 000 |  |
| **Создание приложения** | 10 000 |  |

Для расчета стоимости проделанной работы потребуется узнать:

• Среднюю заработную плату программиста на предприятии;

• Затраты на оборудование;

• Затраты на электроэнергию.

Среднее значение заработной платы у .NET разработчика необходимо вычислить при помощи анализа заработной платы с нескольких предприятий по г. Москва (Московская область).

Например,

• Заработная плата в размере 250 000 рублей;

Источник:

https://podolsk.hh.ru/vacancy/78543270?from=vacancy\_search\_list&query=.net%20c%23

• Заработная плата в размере 150 000 рублей

Источник:

https://podolsk.hh.ru/vacancy/78134485?from=vacancy\_search\_list&query=.net%20c%23

• Заработная плата в размере 175 000 рублей

Источник:

https://podolsk.hh.ru/vacancy/78393533?from=vacancy\_search\_list&query=.net%20c%23

• Заработная плата в размере 195 000 рублей

Источник:

https://podolsk.hh.ru/vacancy/77600339?from=vacancy\_search\_list&query=.net%20c%23

• Заработная плата в размере 220 000 рублей

https://podolsk.hh.ru/vacancy/77423625?from=vacancy\_search\_list&query=.net%20c%23

Высчитываем среднее значение:

(250 000 + 150 000 + 175 000 + 195 000 + 220 000) / 5 = 198 000 рублей - средняя зарплата у .NET разработчика.

Вычисляем почасовую оплату. Для этого нужно разделить среднюю заработную плату за год на фонд рабочего времени в 2023 году:

198 000 рублей х 12 месяцев / 1973 часов = 1 204 рубля (среднегодовая почасовая оплата)

На создание предоставленной в выпускной квалификационной работе информационной системы в общем объеме ушло 100 часов.

1 204\*100 = 120 400 рублей оплата всех часов работы.

Общие затраты на оплаты труда составили:

Заработная плата + Страховые взносы во внебюджетные фонды (30 %)

120 400 + 120 400 \* 30% = 120 400 + 36 120 = 156 520 рублей

Рассчитаем примерные затраты на электроэнергию, узнав тариф на электроэнергию;

Тариф на 1 кВт/ч = 5,92 рублей.[[1]](#footnote-1)

Примерное использование кВт в час составляет 0,45. Следовательно, на электроэнергию мы тратим: 0,45 кВтч \* 5,92 руб. = 2,66 рубля (в час).

2,66 руб. \* 100 ч = 266 рубля – затраты на электроэнергию.

Таким образом, себестоимость разработки информационного ресурса для обучения и дистанционного тестирования сотрудников составила 168 786 рублей.

**Таблица 3 – Расчет себестоимости и цены программного продукта**

|  |  |  |
| --- | --- | --- |
| **Показатели** | **Условное обозначение** | **Значение** |
| 1. Прямые расходы, в том числе |  |  |
| 1.1 Затраты на оплату труда | ЗП | 120 400 |
| 1.2 Страховые взносы во внебюджетные фонды | СтрВзн | 36 120 |
| 1.3. Затраты на текущий и профилактический ремонт (Амортизационные отчисления *(50000\*20%/12)+ услуги Интернет провайдера)* | Зпр.р | 2000 |
| 1.5. Затраты на материалы, необходимые для эксплуатации | Зматер | 10 000 |
| 1.6. Затраты на электроэнергию | Зэл | 266 |
| 3. Итого затраты на разработку программного продукта (полная себестоимость) | ОбщЗ | 168 786 |
| 4. Плановая прибыль (20%) | П | 33 757 |
| 5. Цена РАЗРАБОТЧИКА | Ц | 202 543 |
| 6. НДС (20% от цены) | НДС | 40 509 |
| 7. Цена продажная в розницу с НДС | РознЦ | 243 052 |

**3.3. Экономический эффект и эффективность**

Стоимость программного продукта для конечного пользователя сопоставима с предложениями других поставщиков.

Ввиду проектно-исследовательского характера работы и отсутствия конкретных объектов внедрения приведены общие данные экономического эффекта внедрения программного продукта (информационного ресурса для обучения и дистанционного тестирования сотрудников НИИ неотложной детской хирургии и травматологии Департамента здравоохранения города Москвы).

Ожидаемый экономический эффект внедрения программного продукта (информационного ресурса для обучения и дистанционного тестирования сотрудников НИИ неотложной детской хирургии и травматологии Департамента здравоохранения города Москвы): Повышение эффективности обучения сотрудников компании.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта я смог решить все задачи, поставленные передо мной заказчиком и реализовать проект согласно требованиям. Основной целью этого проекта являлось создание электронной системы обучения с системой дистанционного тестирования сотрудников. Данная система позволяет сотрудникам компании максимально эффективно получать необходимую для работы информацию и демонстрировать качество усвоенной информации. Проект был подведен к готовому состоянию и протестирован согласно всем необходимым для этого требованиям. Так же данный продукт имеет высокую автономность и способен работать без вмешательства со стороны специалистов по обслуживанию данного проекта.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ ИНФОРМАЦИИ

**Книжные (Печатные) издания:**

1. Виталий Трунин: Путь программиста T-SQL. Самоучитель по языку Transact-SQL, 2020, 204 стр.
2. Фримен А.: Entity Framework Core 2 для ASP.NET Core MVC для профессионалов, 2019, 626 стр.
3. Казанский А. А*.*: Программирование на Visual C#: учебное пособие для среднего профессионального образования, 2021, 192 стр.
4. Джепикс Филипп, Троелсен Эндрю: Язык программирования C# 7 и платформы .NET и .NET Core, 2018, 1328 стр.
5. Скит Джон: C# для профессионалов. Тонкости программирования, 2019, 608 стр.
6. Чаллавала Шаббир: MySQL 8 для больших данных, 2018 226 стр.
7. Уорд Брайан: Инновации SQL Server 2019. Использование технологий больших данных и машинного обучения, 2020, 408 стр.

**Информационные ресурсы:**

1. https://metanit.com - Информационный ресурс «Метанит»
2. https://ru.wikipedia.org - Информационный ресурс «Википедия»
3. https://visualstudio.microsoft.com/ru/ - Интернет - магазин и разработок «Microsoft»
4. https://studopedia.ru/ - Учебные пособия и материалы для студентов «Студопедия»
5. https://www.jetbrains.com - Программные решения компании JetBrains
6. https://spravochnick.ru/programmirovanie/yazyki\_programmirovaniya/yazyk\_programmirovaniya\_basic/ - Информационный ресурс для программистов «Справочник»
7. <https://habr.com/ru/company/spbifmo/blog/476160/> - Рождение образовательного ПО и его история: от механических машин до первых компьютеров
8. <https://studfile.net/preview/8828003/page:10/> - Компьютерные учебные программы

1. [↑](#footnote-ref-1)