**ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ ЯРОСЛАВСКОЙ ОБЛАСТИ**

государственное профессиональное образовательное учреждение   
Ярославской области

Рыбинский полиграфический колледж

КУРСОВОЙ ПРОЕКТ

|  |  |
| --- | --- |
| Разработка клиент-серверного приложения «Система тестирования знаний: составление и прохождение тестов» | |
|  | |
| по дисциплине | Разработка, администрирование и защита баз данных |
|  | |

Пояснительная записка

|  |
| --- |
| КП.090207.11.000000.00 ПЗ |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент группы | 4-ИС-3 |  |  |  | И. А. Малышкин |
|  | *(Код учебной группы)* |  | *(Подпись, дата)* |  | *(И.О.Фамилия)* |
| Руководитель | преподаватель |  |  |  | Е. А. Лобанова |
|  | *(Должность, звание)* |  | *(Подпись, дата)* |  | *( И.О.Фамилия)* |
| Нормоконтроль | преподаватель |  |  |  | Е. А. Лобанова |
|  | *(Должность, звание)* |  | *(Подпись, дата)* |  | *(И.О.Фамилия)* |

г. Рыбинск

2024

содержание

[Введение 3](#_Toc167175076)

[1 Исследовательский раздел 6](#_Toc167175077)

[2 Конструкторский раздел 10](#_Toc167175078)

[2.1 Проектирование информационной модели данных 10](#_Toc167175079)

[2.2 Проектирование серверной части приложения 11](#_Toc167175080)

[2.2.1 Разработка схемы базы данных 11](#_Toc167175081)

[2.2.2 Разработка сущностей базы данных 13](#_Toc167175082)

[2.3 Проектирование клиентской части приложения 15](#_Toc167175083)

[2.3.1 Разработка модулей схемы 15](#_Toc167175084)

[2.3.2 Разработка пользовательского интерфейса 17](#_Toc167175085)

[2.3.3 Организация доступа к объектам базы данных 22](#_Toc167175086)

[2.3.4 Разработка блох-схем алгоритмов процедур и функций 23](#_Toc167175087)

[2.4 Обеспечение коллективного доступа. Защита информации 26](#_Toc167175088)

[3 Технологическая часть 29](#_Toc167175089)

[3.1 Тестирование и отладка приложения 29](#_Toc167175090)

[3.2 Инструкция администратора базы данных 31](#_Toc167175091)

[3.3 Инструкция по эксплуатации приложения 41](#_Toc167175092)

[4 Технико-экономический раздел 47](#_Toc167175093)

[5 Раздел охраны труда 59](#_Toc167175094)

[Заключение 61](#_Toc167175095)

[Список используемых источников 62](#_Toc167175096)

[Приложение А 63](#_Toc167175097)

Введение

В наше время вычислительная техника является неотъемлемой частью нашей жизни и широко применяется в различных отраслях. Она играет особенно важную роль в научных исследованиях, где позволяет изучать и анализировать разнообразные аспекты окружающего мира, микроорганизмы, историю, космос и многое другое. Вычислительная техника также используется для обработки, хранения и анализа огромного объема данных, а также для создания цифровых моделей объектов реального мира.

Ключевым компонентом вычислительной техники является компьютер, для управления которым необходимо программное обеспечение. Языки программирования были созданы для упрощения процесса написания программного кода. С самых ранних низкоуровневых языков программирования, разработанных специально для конкретных компьютеров, до появления языков высокого уровня, которые стали более удобными и абстрактными, прошло много времени. Первым “настоящим” языком программирования стал Plankalkül [2] (Расчет плана) разработанный человеком по имени Конрад Цузе где-то между 1944 и 1945 годами.

Середина 1950-х годов стала переломным периодом в развитии языков программирования, когда появились первые языки высокого уровня, не привязанные к конкретным типам компьютеров. Первые алгоритмические языки эпохи ЭВМ Фортран и Ангол. Они были ориентированы в основном на системные расчеты. Причем Фортран был первым языком, для которого были созданы компиляторы для многих ЭВМ. Компилятор переводит программу с языка высокого уровня на язык ЭВМ. Алгол был разработан в 1958 году в международном комитете под руководством Питера Наура [3]. Алгоритмические языки предложили новый подход, позволяющий повторно использовать код и имеющий более "человеческий" синтаксис. Это существенно упростило процесс написания программ и сократило объем кода.

Однако с развитием программ стало очевидно, что структура программы имеет большое значение, особенно при работе с длинными и сложными программами. В результате появилась методология структурного программирования, которая ввела понятие процедур и функций, улучшая читаемость и организацию кода.

Однако для решения ещё более сложных задач требовался новый подход, который бы позволил более эффективно организовывать программы и разбивать их на составные части. Таким подходом стало объектно-ориентированное программирование (ООП), которое позволяет описывать программу в терминах объектов и классов, формируя иерархию наследования и взаимодействуя между собой с помощью сообщений. Первым языком программирования, в котором были предложены основные понятия, впоследствии сложившиеся в парадигму, была Simula-1, но термин «объектная ориентированность» не использовался в контексте использования этого языка. В момент его появления в 1967 году в нём были предложены революционные идеи: объекты, классы, виртуальные методы и др., однако это всё не было воспринято современниками как нечто грандиозное. Фактически, Симула была «Алголом с классами», упрощающим выражение в процедурном программировании многих сложных концепций [7].

Объектно-ориентированное программирование предлагает ряд принципов, таких как инкапсуляция, наследование и полиморфизм, которые способствуют более гибкой и модульной разработке программного кода. Этот подход не только повышает эффективность работы программистов, но и позволяет лучше отражать реальный мир в структуре программы.

В рамках данного курсового проекта будет разработан wpf проект, представляющий собой клиент-серверное приложение "Система тестирования знаний". Цель курсового проекта состоит в разработке и обосновании системы тестирования знаний, основанной на языке программирования C# и WPF, с учетом современных требований образовательного и корпоративного обучения. В данной работе рассматриваются ключевые аспекты, такие как дистанционное обучение, персонализированное обучение, стандартизация, а также эффективность пользовательского интерфейса, объединенные с целью создания многофункциональной и востребованной системы тестирования знаний.

1 Исследовательский раздел

Представьте себе, что вы создаете систему тестирования знаний, где студенты могут проверять свои знания, а преподаватели могут оценивать и анализировать результаты. Ваша система будет клиент-серверным приложением, где клиентская часть представляет собой интерфейс пользователя, а серверная - базу данных и бизнес-логику.

Принцип разделения: Студенты (клиенты) взаимодействуют с приложением через удобный интерфейс, вводят ответы на вопросы, а приложение проверяет правильность ответов и передает эти данные на сервер для сохранения и обработки. Преподаватели могут проверить результаты, создавать тесты и следить за активностью студентов.

**Принцип безопасности:** Для обеспечения конфиденциальности и целостности данных, сервер должен обеспечивать аутентификацию пользователей и использовать шифрование данных при их передаче.

**Принцип надежности:** Серверная часть приложения должна быть устойчивой к сбоям, чтобы предотвращать потерю данных студентов и сохранять их результаты тестирования.

**Принцип гибкости:** Программа тестирования должна быть легко настраиваемой и готовой к обновлениям. Это может включать в себя возможность добавления новых типов вопросов или адаптацию системы под различные учебные программы.

Одним из основных преимуществ выбора WPF для написания клиентской части данного клиент-серверного приложения является его мощная система разработки графических интерфейсов. Выбор WPF для клиентской части приложения обеспечивает баланс между эффективностью разработки и созданием современного, привлекательного интерфейса для пользователей.

Visual Studio (VS) — интегрированная среда разработки от Microsoft. Она поддерживает множество технологий и языков программирования, включая разработку приложений на платформе .NET.

**Основные характеристики:**

**Поддержка языков программирования**: Позволяет разрабатывать приложения на C#, VB.NET, F# и других языках.

**Дизайнер интерфейса:** Визуальные инструменты для разработки интерфейса приложений, включая дизайнер XAML для WPF.

**Отладка и профилирование:** Интегрированные средства отладки и профилирования приложений для быстрой и эффективной разработки.

**Богатый редактор кода:** Мощный редактор кода с подсветкой синтаксиса, автодополнением и другими инструментами.

**Интеграция с платформой .NET:** Поддержка всех основных версий .NET Framework и .NET Core.

**Расширяемость:** Возможность установки плагинов и расширений для индивидуализации среды разработки.

Причины выбора:

**Полная интеграция с технологиями Microsoft:** Visual Studio предоставляет лучшую интеграцию с платформой .NET и другими технологиями Microsoft, что облегчает разработку для этой экосистемы.

**Широкие возможности для разработки WPF-приложений:** Встроенные инструменты для работы с WPF и XAML обеспечивают комфортную разработку приложений с богатым графическим интерфейсом.

**Большое сообщество и поддержка:** Visual Studio имеет обширное сообщество разработчиков, множество ресурсов и обширную базу знаний, что облегчает решение проблем и поиск поддержки.

**Сравнение с JetBrains Rider:**

**Языки программирования:**

Visual Studio: Поддерживает широкий спектр языков, включая C#, VB.NET, F#, Python и другие.

Rider: Ориентирована на использование с языками, работающими на платформе .NET, такими как C#, VB.NET, F#.

**Поддержка платформ:**

Visual Studio: Имеет мощную поддержку .NET и других технологий Microsoft.

Rider: Ориентирована на кроссплатформенную разработку и может быть использована для проектов на Linux и macOS.

**Производительность:**

Visual Studio: Некоторые разработчики могут считать Visual Studio более ресурсоемкой, особенно для больших проектов.

Rider: Известна своей хорошей производительностью и быстрым запуском.

**Стоимость:**

Visual Studio: Имеет несколько версий, включая бесплатную Community Edition и платные Professional и Enterprise.

Rider: Обычно требует лицензии, но может предоставляться бесплатно для определенных типов проектов.

Выбор между Visual Studio и Rider зависит от ваших предпочтений, опыта и конкретных требований проекта. Обе среды предоставляют мощные инструменты для разработки клиент-серверных приложений с использованием WPF.

В интернете существует большое количество систем тестирования занний. Одним из таких примеров можно ознакомиться на сайте «moretestov» [4]. Данная версия системы – это пример классического сайта для прохождения и составления тестов. В этой версии можно создавать, удалять, изменять и проходить тесты. Управление осуществляется с помощью манипулятора типа «мышь». Недостатками данной версии является обязательное подключение к интернету и то, что она не является desktop-приложением.

Цель данного курсового проекта – создание desktop-приложения «Система тестирования знаний» лишённой таких недостатков.

В данном приложении планируется реализовать приятный для глаз интерфейс с возможностью кастомизации. Возможность авторизации и регистрации в приложении, профиль пользователя с возможностью его изменения, двухфакторная аутентификация с помощью электронной почты. А также функционал для создания, редактирования, удаления и прохождения тестов.

2 Конструкторский раздел

2.1 Проектирование информационной модели данных

Для описания метода проектирования приложения будет применяться последовательная декомпозиция, начиная с концепции "черной сферы". Черная сфера представляет собой систему, внутреннее устройство которой не имеет значения. В эту систему вводятся входные данные, а на выходе получаются выходные данные. Черная сфера представлена на рисунке 2.1.



Рисунок 2.1 – Черная сфера

Представим наше приложение в виде черной сферы. В приложении будут присутствовать такие входные данные, как пользователь и тесты. На выходе из приложения будут результаты тестов. Пользователем будет выступать человек, планирующий пройти или с ставить свой тест. Приложение – это то, что позволит пользователю взаимодействовать с тестами. Результат, полученный после взаимодействия пользователя с приложением это итоги теста. Черная сфера с

перечисленными входными и выходными параметрами представлена на рисунке 2.2.

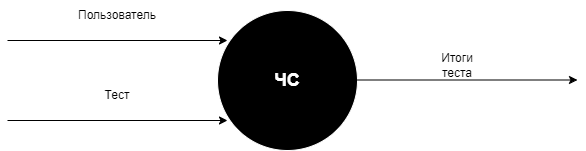


Рисунок 2.2 – Черная сфера с параметрами

2.2 Проектирование серверной части приложения

2.2.1 Разработка схемы базы данных

Создание эффективной и надежной базы данных – ключевой этап в разработке информационных систем, обеспечивающий эффективное хранение, управление и извлечение данных. База данных является основой для функционирования многих приложений, от корпоративных систем до веб-приложений, поэтому разработка ее схемы требует особого внимания и профессионализма.

Этот процесс включает в себя не только определение сущностей и их атрибутов, но также учет требований к производительности, безопасности и масштабируемости системы. Введение в разработку схемы базы данных включает в себя тщательное планирование, анализ бизнес-процессов и учет особенностей предметной области.

На первом этапе нормализации можно представить модель как связь между «Пользователем» и «Тестом ». Первый этап нормализации представлен на рисунке 2.3.



Рисунок 2.3 – Первый этап нормализации

Во втором заключительном этапе нормализации разобьём эту схему на полную диаграмму базы данных. Тесты разделятся на «вопросы» и «результаты», а от вопросов будет отходить ещё одна сущность «ответы» Второй этап нормализации представлен на рисунке 2.4.

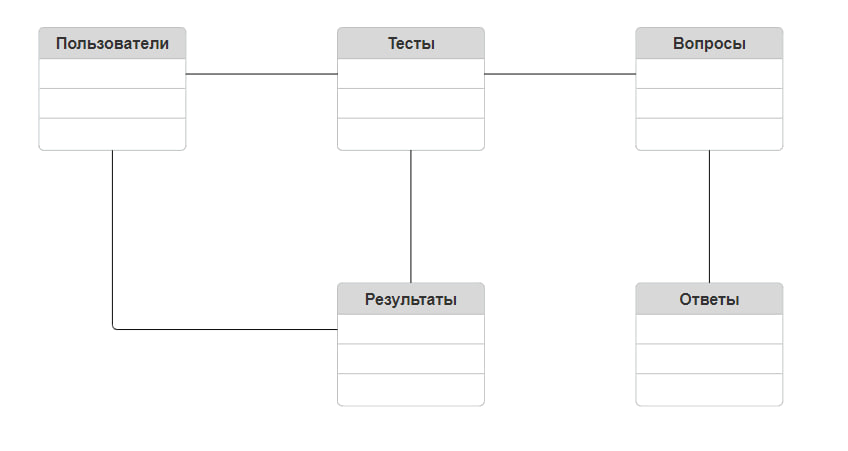


Рисунок 2.4 – Второй этап нормализации

Получим логическую модель данных с содержанием всех сущностей, связей и атрибутов данных. Логическая модель данных представлена на рисунке 2.5.

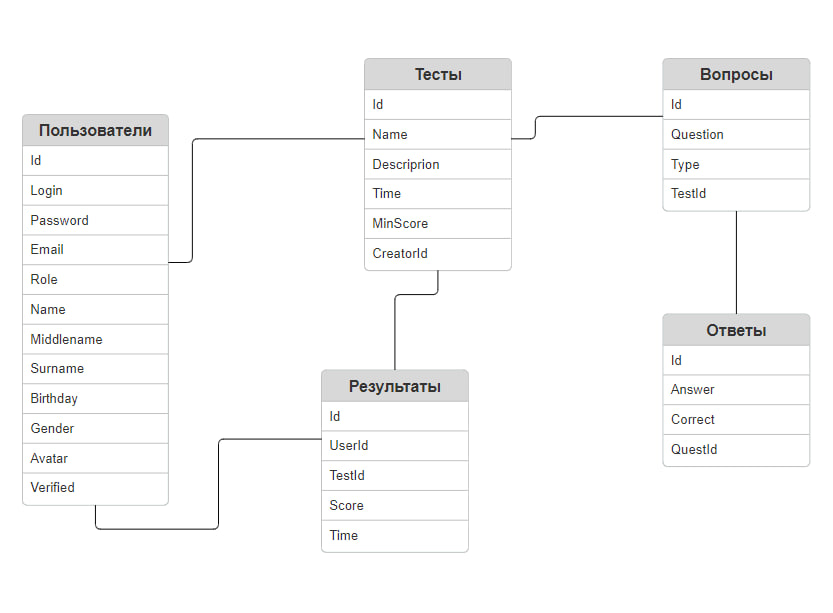


Рисунок 2.5 – Логическая модель данных

Получим физическую модель данных, включающая ассоциативные таблицы, которые иллюстрируют отношения между сущностями, а также первичные и внешние ключи для связи данных. Физическая модель данных представлена на рисунке 2.6.

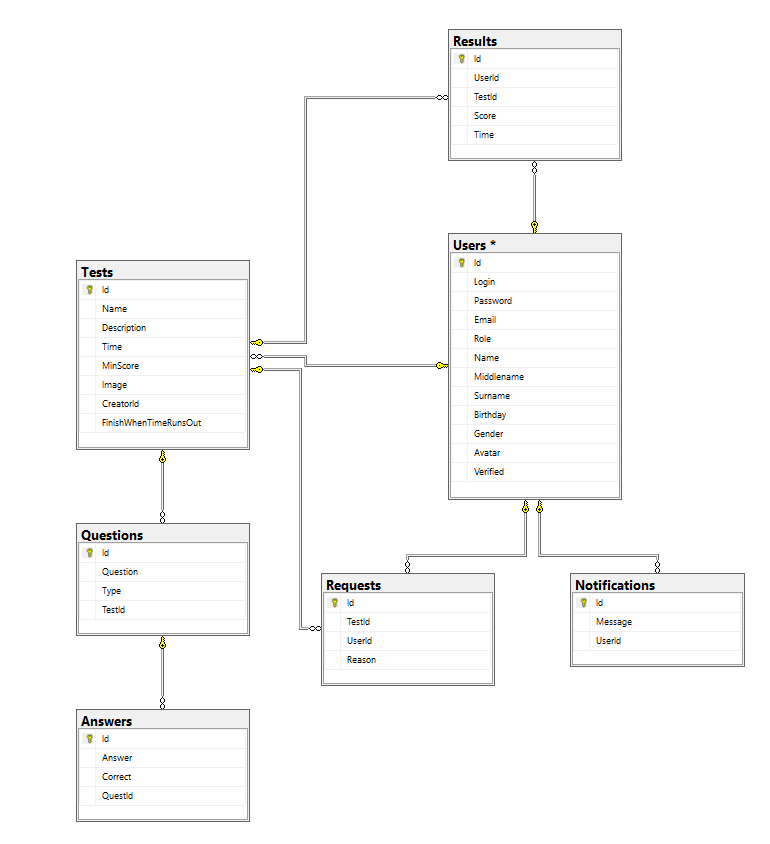


Рисунок 2.6 – Физическая модель данных

2.2.2 Разработка сущностей базы данных

В разделе "Разработка схемы базы данных" была создана структура базы данных, из которой вытекает необходимость наличия определенных сущностей для полноценного функционирования приложения. Все эти сущности удобно представлены в табличной форме для легкости восприятия. Таблица 2.1 содержит представление сущностей, входящих в схему базы данных.

Таблица 2.1 – Сущности схемы базы данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя сущности | Назначение сущности | Типы данных | Перечисление наименований сущностей, которые подчиняются текущей сущности | Перечисление наименований сущностей, которым подчиняется текущая сущность |
| Tests | Содержит данные о тестах | int, nvarchar(50), nvarchar(200),time(7), int, int | Questions, Results | Users |
| Questions | Содержит вопросы тестов | int, nvarchar(150), nvarchar(150), int | Answers | Tests |
| Answers | Содержит информацию о ответах на вопросы | Int, nvarchar(500), bit, int | - | Questions |
| Results | Содержит информацию о результатах тестов | Int, int,int,float,time(7) | - | Users, Tests |
| Users | Содержит информацию о всех пользователях | int, nvarchar(30), nvarchar(50), nvarchar(150), nvarchar(150), nvarchar(150), nvarchar(150), nvarchar(150), date, bit, varbinary(max), bit | Results, Tests, Notifications | - |

Окончание таблицы 2.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя сущности | Назначение сущности | Типы данных | Перечисление наименований сущностей, которые подчиняются текущей сущности | Перечисление наименований сущностей, которым подчиняется текущая сущность |
| Notifications | Содержит данные об уведомлениях пользователей | int, nvarchar(500) | - | Users |
| Requests | Содержит информацию о запросах на пере прохождение тестов | int, int, int, nvarchar(500) | Results | Users, Tests |

2.3 Проектирование клиентской части приложения

2.3.1 Разработка модулей схемы

WPF предоставляет обширный набор возможностей для разработки приложений, включающий в себя язык разметки XAML, элементы управления, механизмы привязки к данным, систему компоновки, инструменты для работы с дву- и трехмерной графикой, анимацию, стили, шаблоны, обработку документов, мультимедийные ресурсы, текстовые и типографические функции. Поскольку WPF является частью платформы .NET, вы можете создавать приложения, которые взаимодействуют с другими элементами API .NET [5].

Представление клиентской части приложения в виде модульной схемы позволяет наглядно отобразить взаимосвязь между окнами, классами и страницами при организации клиентской части приложения. Схема модульной структуры клиентской части приложения представлена на рисунке 2.7.

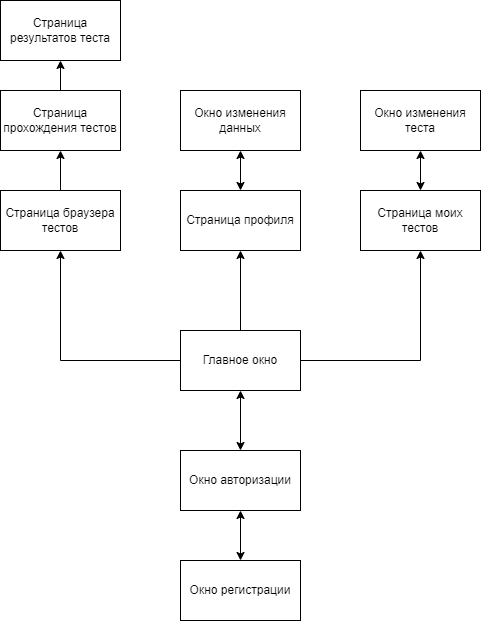


Рисунок 2.7 – Модульная схема клиентской части приложения

* Окно регистрации, для регистрации пользователей;
* Окно авторизации, для авторизации пользователей;
* Главное окно, для взаимодействия пользователя с основными окнами программы;
* Страница моих тестов, для создания и изменения собственных тестов;
* Окно изменения теста, для редактирования состава теста;
* Страница профиля, для просмотра своего профиля;
* Окно изменения данных, для редактирования данных пользователя;
* Страница браузера тестов, для поиска тестов;
* Страница прохождения теста, для прохождения теста;
* Страница результатов теста, для просмотра итогов теста;

2.3.2 Разработка пользовательского интерфейса

Интерфейс пользователя (User Interface, UI) представляет собой точку взаимодействия между человеком и компьютерной программой или устройством. Этот интерфейс включает в себя все элементы и методы, которые пользователь использует для взаимодействия с программой, веб-сайтом, приложением или устройством. UI охватывает различные аспекты, такие как графические элементы, элементы управления, ввод данных, анимации, макеты и другие аспекты, которые делают использование программы или устройства удобным и эффективным для пользователя.

Графический интерфейс пользователя (Graphical User Interface, GUI) представляет собой форму UI, которая в основном использует графические элементы, такие как окна, кнопки, иконки и другие визуальные элементы, чтобы обеспечить пользователю более интуитивное и привычное взаимодействие с программой или устройством.

XAML — это декларативный язык разметки. При применении к модели программирования .NET XAML упрощает создание пользовательского интерфейса для приложения .NET. Можно создать видимые элементы пользовательского интерфейса в декларативной XAML-разметке, а затем отделить определение пользовательского интерфейса от логики времени выполнения, используя файлы кода программной части, присоединенные к разметке с помощью определений разделяемых классов.[1]

Состав блоков модульной схемы:

- Главное окно содержит в себе меню с кнопками для перехода по 3 различным страницам и зону для вывода информации

Разметка ссылок для перехода на другие окна представлена на рисунке 2.8.

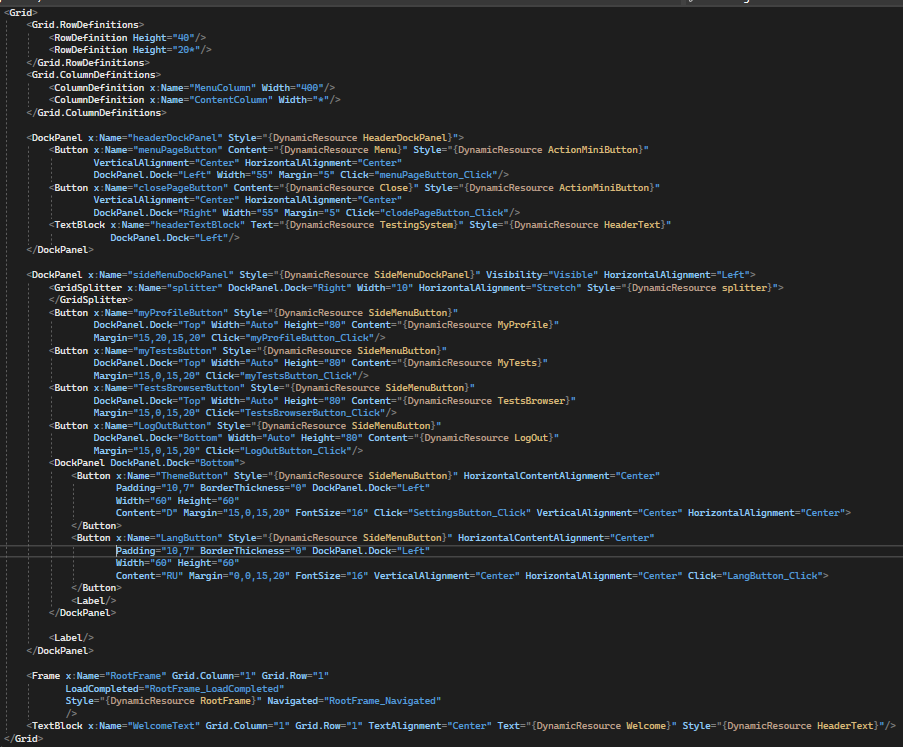


Рисунок 2.8 – Разметка ссылок

- Окно авторизации содержит в себе 2 поля логин и пароль для ввода данных и 4 кнопки. Кнопка для скрытия пароля, кнопка регистрации входа и отмены.

Разметка текстовых полей представлены на рисунке 2.9

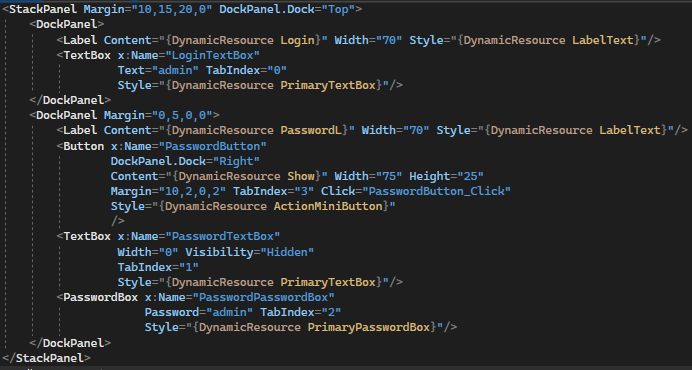


Рисунок 2.9 – Пример разметки листа

- Окно регистрации 7 текстовых полей для логина, пароля, почты, личной информации и капчи ещё есть 1 поле с выбором даты рождения, а также 4 кнопки. Капча состоит из 2 текстовых полей с капчой и её вводом, а также кнопки подтверждения введенной капчи.

Разметка капчи представлена на рисунке 2.10.



Рисунок 2.10 – Разметка капчи

- Разметка Frame и приветствующего текста представлены на рисунке 2.11

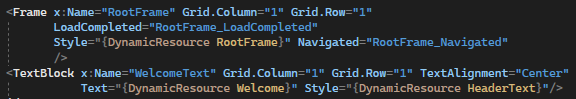


Рисунок 2.11 – Разметка Frame

- Страница профиля состоит из текстовых блоков с информацией о пользователе кнопок для её редактирования и блока с аватаром пользователя.

Разметка блока с аватаром пользователя представлена на рисунке 2.12.

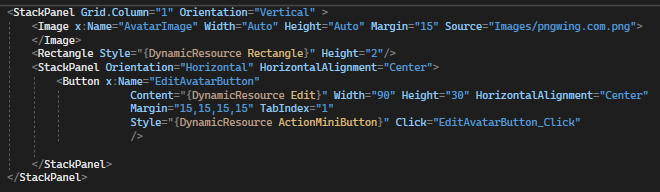


Рисунок 2.12 – Разметка аватара

- Страница моих тестов состоит из верхнего меню с элементами управления тестами и строкой поиска, а также из списка тестов в основной части экрана реализованных с помощью ListView.

Разметка списка тестов представлена на рисунке 2.13.

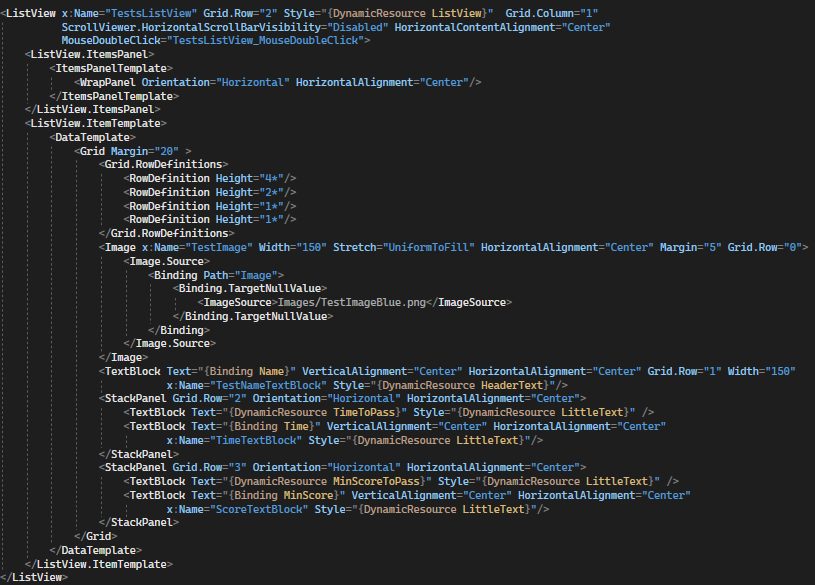


Рисунок 2.13 – Разметка списка тестов

- Страница прохождения теста состоит из заголовка сверху с названием теста, полем с вопросами и ответами по центру экрана и секции снизу в которой показывается оставшееся время и кнопка завершения теста.

Разметка ListBox с вопросами и ответами представлена на рисунке 2.14.

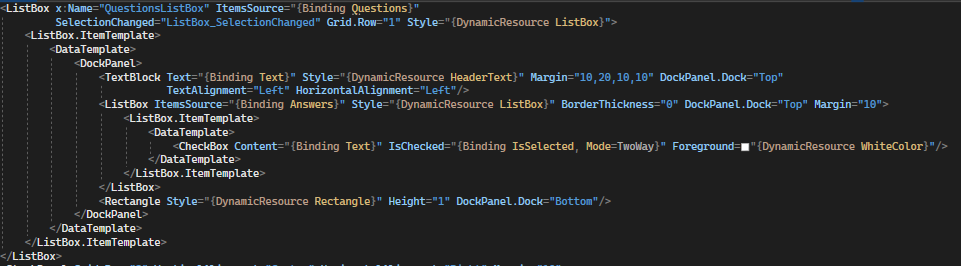


Рисунок 2.14 – Разметка списка вопросов и ответов

- Страница результатов теста состоит из заголовка с названием теста, результатом и кнопки закрытия. Центральной области в которой отображаются все ответы и меню снизу с более развёрнутой информацией.

Список с ответами представлен на рисунке 2.15.

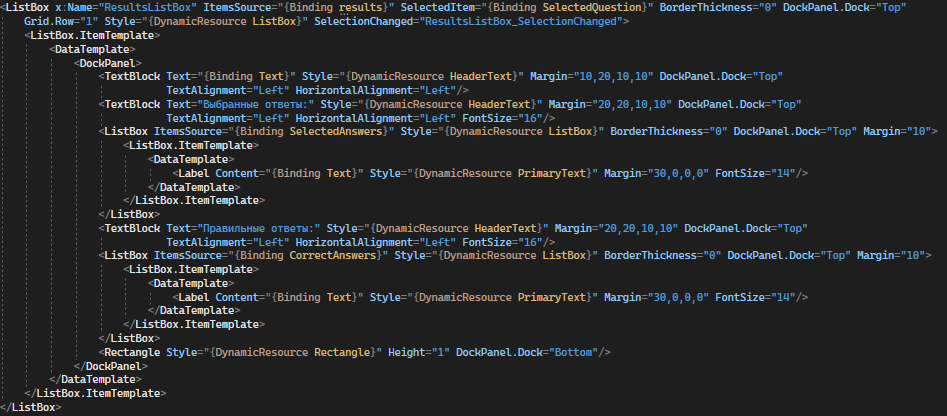


Рисунок 2.15 – Лист ответов на вопросы

- Страница редактирования вопросов состоит из двух DataGrid в которых выводится список вопросов и список ответов привязанных к выбранному вопросу, а также из верхнего меню с элементами управления списками.

Разметка DataGrid справочника ответы представлена на рисунке 2.16.

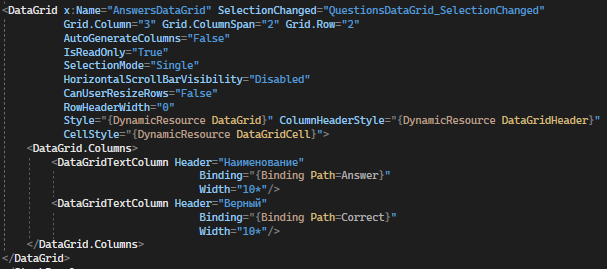


Рисунок 2.16 – Разметка DataGrid справочника ответы

2.3.3 Организация доступа к объектам базы данных

В WPF привязка (binding) является мощным инструментом программирования, без которого не обходится ни одно серьезное приложение. Привязка подразумевает взаимодействие двух объектов: источника и приемника. Объект-приемник создает привязку к определенному свойству объекта-источника. В случае модификации объекта-источника, объект-приемник также будет модифицирован. [6]

Для примера возьмём страницу редактирование вопросов где участвует DataGrid.

DataGrid страницы с вопросами представлен на рисунке 2.17

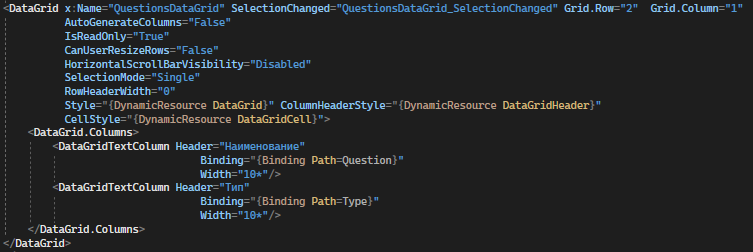


Рисунок 2.17 – Разметка DataGrid страницы с вопросами

У Binding в свойство Path мы записываем свойство объекта источника. Привязка объекта с данными к DataGrid представлена на рисунке 2.18.



Рисунок 2.18 – Привязка объекта с данными к DataGrid

Беря данные из базы данных, мы привязываем их с DataGrid. Сам DataGrid верстается под конкретно содержимое.

2.3.4 Разработка блох-схем алгоритмов процедур и функций

Основной задачей приложения является составление и прохождение тестов. Для составления теста нужно заполнить вопросы и ответы в тесте, а также указать время и минимальное количество балов. Для прохождения теста нужно выбрать нужный вам тест в браузере и ответить на все вопросы, после этого вы получите итоги тестирования.

Метод CalculateScore() предназначен для подсчета результата теста с учетом штрафов по времени. В качестве входных параметров выступает список результатов. Выходные же параметры состоят из количества полученых пользователем балов и количества ответов. Блок-схема метода CalculateScore() представлена на рисунке 2.19.



Рисунок 2.19 – Метод CalculateScore()

Метод SendMessage() отправляет письмо подтверждения на email пользователя. В качестве входных параметров он принимает адрес эл. Почты, а в выходных параметрах выдаёт результат отправки сообщения. Блок-схема метода SendMessage() представлена на рисунке 2.20.

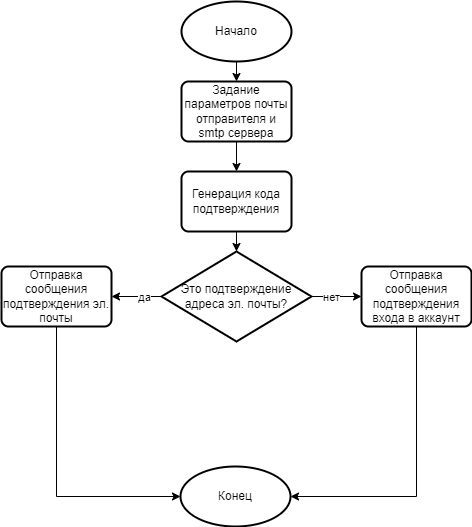


Рисунок 2.20 – Метод SendMessage()

Метод GenerateCode() генерирует код для отправки на эл. Почту. Входных параметров нет. Возвращает сгенерированный код. Блок-схема метода GenerateCode() представлена на рисунке 2.21.



Рисунок 2.21 – Метод GenerateCode()

2.4 Обеспечение коллективного доступа. Защита информации

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

Ролевой метод управления доступом контролирует доступ пользователей к информации на основе типов их активностей в системе. Применение данного метода подразумевает определение ролей в системе. Понятие роль можно определить, как совокупность действий и обязанностей, связанных с определенным видом деятельности. Таким образом, вместо того чтобы указывать все типы доступа для каждого пользователя к каждому объекту, достаточно указать тип доступа к объектам для роли. А пользователям, в свою очередь, указать их роли. Пользователь, «выполняющий» роль, имеет доступ, определенный для роли [7].

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

Для авторизации пользователю необходимо ввести логин и пароль. В случае если пользователь не зарегистрирован, он сможет перейти на страницу регистрации из окна авторизации. Окно авторизации представлено на рисунке 2.22.

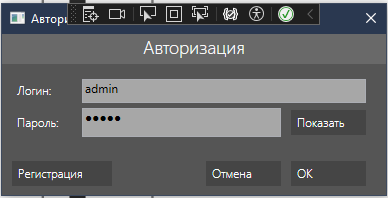


Рисунок 2.22 – Окно авторизации

Для регистрации пользователю необходимо ввести логин, пароль, почту, ФИО, пол, день рождения, повторить показанную капчу. Все поля обладают своей валидацией и в случае некорректного ввода данных, появится окно с уведомлением о некоректном вводе данных. При желании пользователь может вернуться на окно авторизации. Окно регистрации представлено на рисунке 2.23.

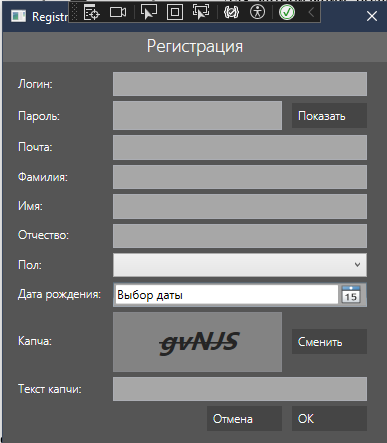


Рисунок 2.23 – Окно регистрации

После корректного заполнения всех полей и нажатия на кнопку «Зарегистрироваться», вас перенесёт в главное меню приложения.

3 Технологическая часть

3.1 Тестирование и отладка приложения

В процессе тестирования будет использоваться подход с использованием тест-кейсов, поскольку они позволяют проверить продукт без необходимости изучать всю документацию. Создание удобного и легко поддерживаемого тест-кейса, который будет написан один раз, сэкономит значительное количество времени и усилий у тестировщиков. Тест-кейс представляет собой документ, описывающий набор шагов, определенные условия и параметры, необходимые для проверки функциональности или ее части. Когда тест-кейсы передаются тестировщику, он должен последовательно пройти через каждый пункт и выполнить описанные действия, чтобы достичь определенных результатов.[5] Тест-кейс для свойств представлен в таблице 3.1.

Таблица 3.1. Тест-кейс свойств класса

|  |  |  |
| --- | --- | --- |
| Имя метода | Управляющее воздействие | Результат воздействия |
| InitializeComponent | Вызывается при создании объекта страницы | Загрузка откомпилированной страницы компонента |
| GenerateCapcha | Вызывается при создании объекта страницы регистрации | Заполнение текстового поля случайными символами длиной от 5 до 10 символов |
| Вызывается при нажатии на кнопку «Новая капча» в окне регистрации | ~ |
| CalculateScore | Вызывается при открытии страницы результатов | Возвращает значение балов за тест |
| GenerateCode | Вызывается при отправке сообщения на почту | Возвращает семизначный код |
| SendMessage | Вызывается при входе с подтверждённого аккаунта | Отправляет сообщение на почту и в качестве ответа возвращает результат отправки |
| Вызывается при подтверждении аккаунта |

Окончание таблицы 3.1

|  |  |  |
| --- | --- | --- |
| Имя метода | Управляющее воздействие | Результат воздействия |
| ParseImageToByte | Вызывается при смене аватара | Возврат массива байтов |
| ParseByteToImage | Вызывается при загрузке профиля | Возвращает приобразованную картинку |
| UpdateTests | Вызывается при изменении поля поиска | Обновляет список тестов |
| SetLang | Вызывается при нажатии на кнопку смены языка | Меняет язык приложения |
| SetTheme | Вызывается при нажатии на кнопку смены темы | Меняет тему приложения |
| CheckInfo | Вызывается при нажатии на ОК | Проверяет введённую информацию |
| StartButton\_Click | Вызывается при нажатии на кнопку | Начинает тестирование |
| ChangeTimer | Вызывается при начале теста | Уменьшение таймера на секунду |
| Button\_Click | Вызывается при нажатии на кнопку завершения теста | Передает информацию в окно результатов |
| DBSaveException | Вызывается при ошибке сохранения | Выводит сообщение об ошибке и откатывает изменения в БД |
| SettingsButton\_Click | Вызывается при нажатии на кнопку настроек | Меняет видимость кнопок настроек |
| initInfo | Вызывается при открытии страницы профиля | Инициализирует информацию профиля |

3.2 Инструкция администратора базы данных

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

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

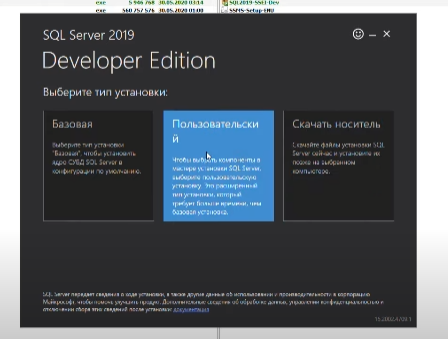


Рисунок 3.1 – «Пользовательский» тип установки

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

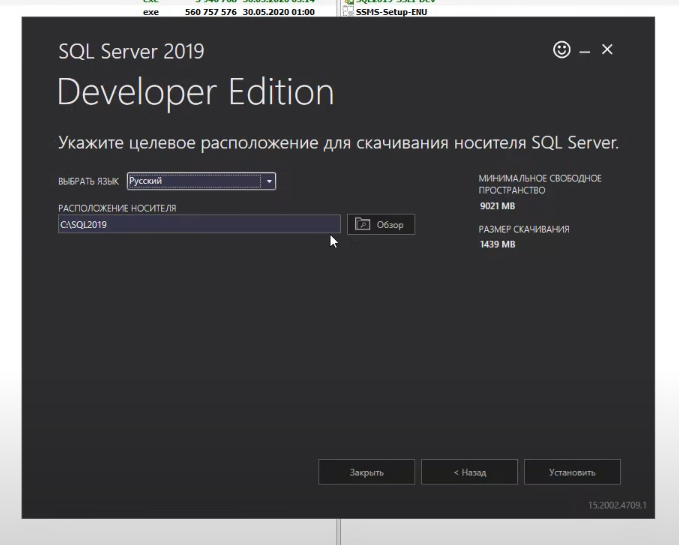


Рисунок 3.2 – Изображение окна

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

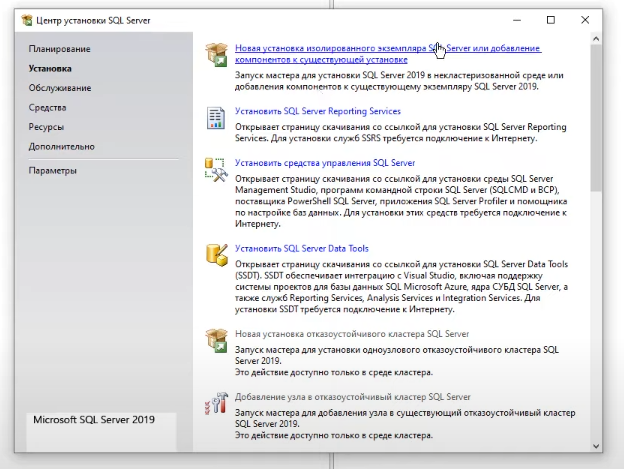


Рисунок 3.3 – Новая установке изолированного экземпляра SQL Server

После установки произойдет обновление продукта. Рисунок 3.4.

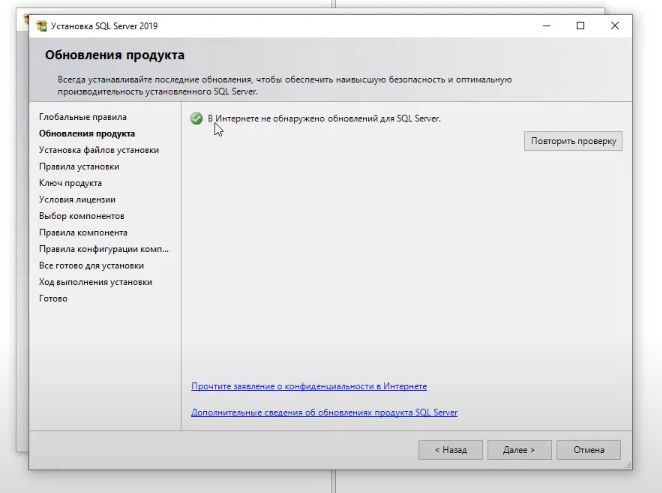


Рисунок 3.4 – Обновление продукта

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

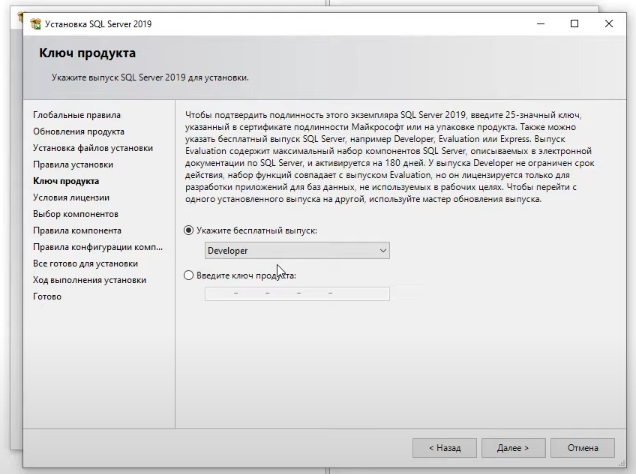


Рисунок 3.5 – Версия «Developer»

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

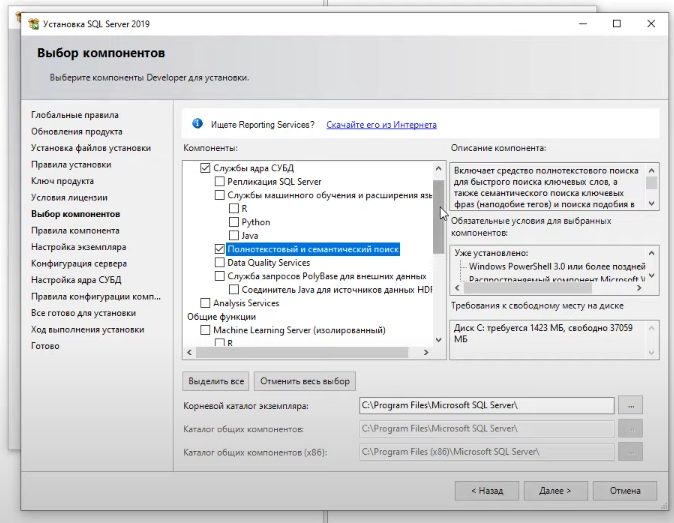


Рисунок 3.6 – Выбор компонентов

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

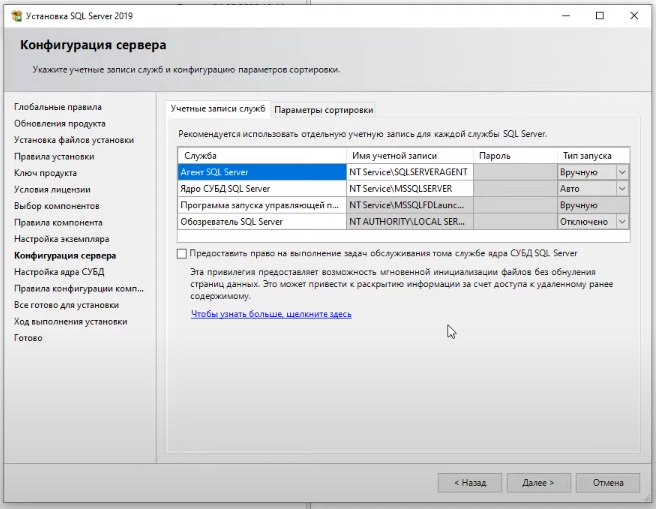


Рисунок 3.7 – Конфигурация сервера

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

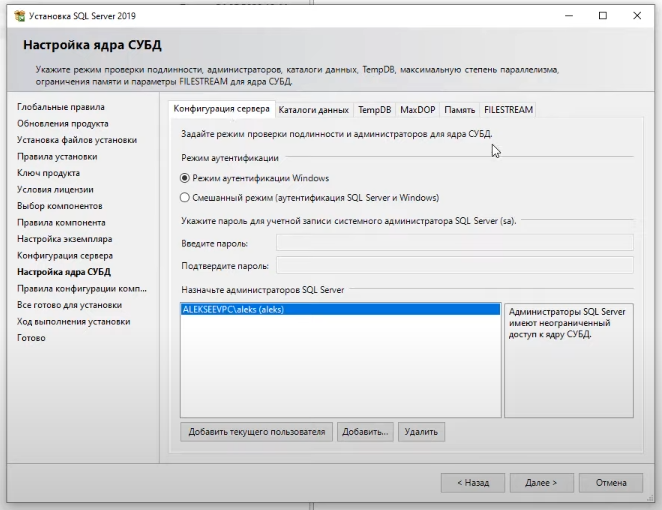


Рисунок 3.8 – Настройка ядра СУБД

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

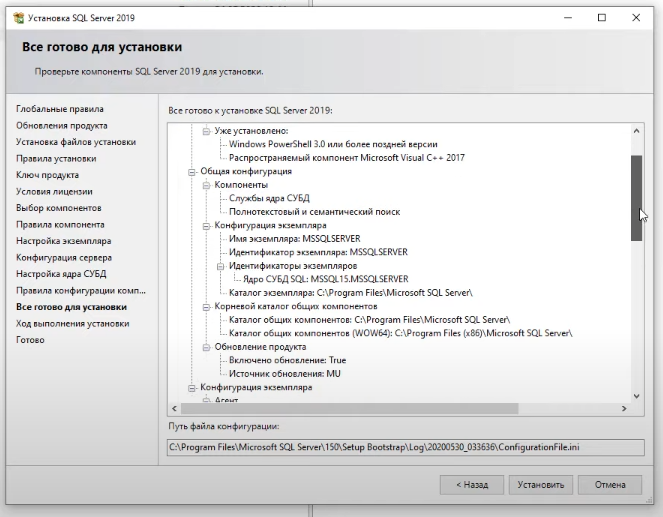


Рисунок 3.9 – Все готово для установки

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

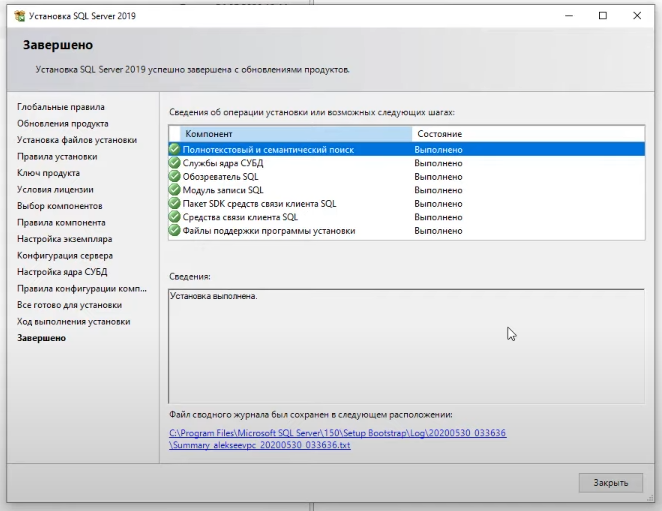


Рисунок 3.10 – Финальный экран

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

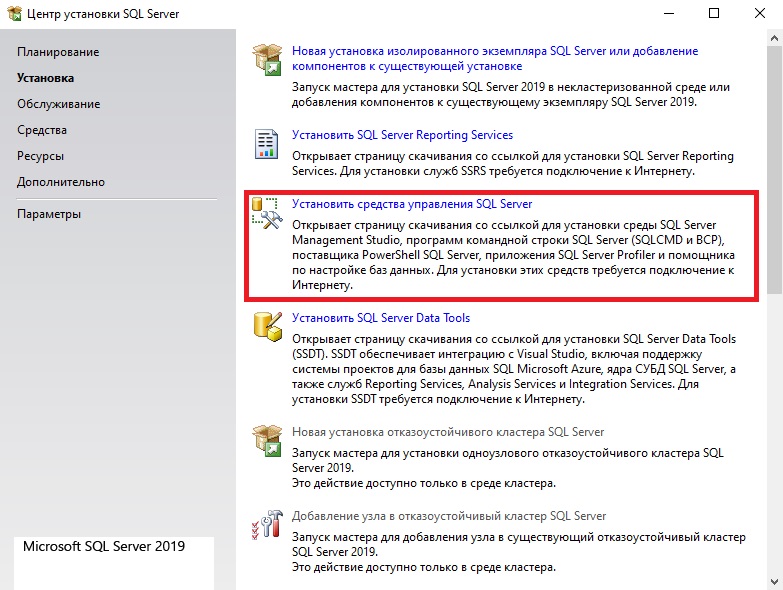
Заходим снова в центр установки SQL Server и наживаем "Установить средства управления SQL Server", как показано на рисунке 3.11.

Рисунок 3.11 – Установка SQL Server Management Studio

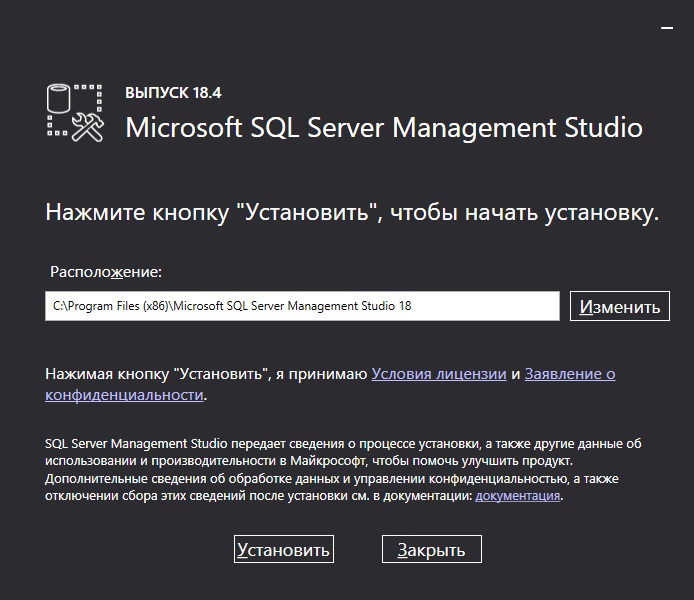
При нажатии у нас откроется [сайт Microsoft](https://docs.microsoft.com/ru-ru/sql/ssms/download-sql-server-management-studio-ssms?redirectedfrom=MSDN&view=sql-server-ver15) и нам нужно будет скачать SSMS. Нажимаем "Установить", как показано на рисунке 3.12.

Рисунок 3.12 – Установка SQL Server Management Studio

Реляционная схема базы данных представлена на рисунке 3.13.

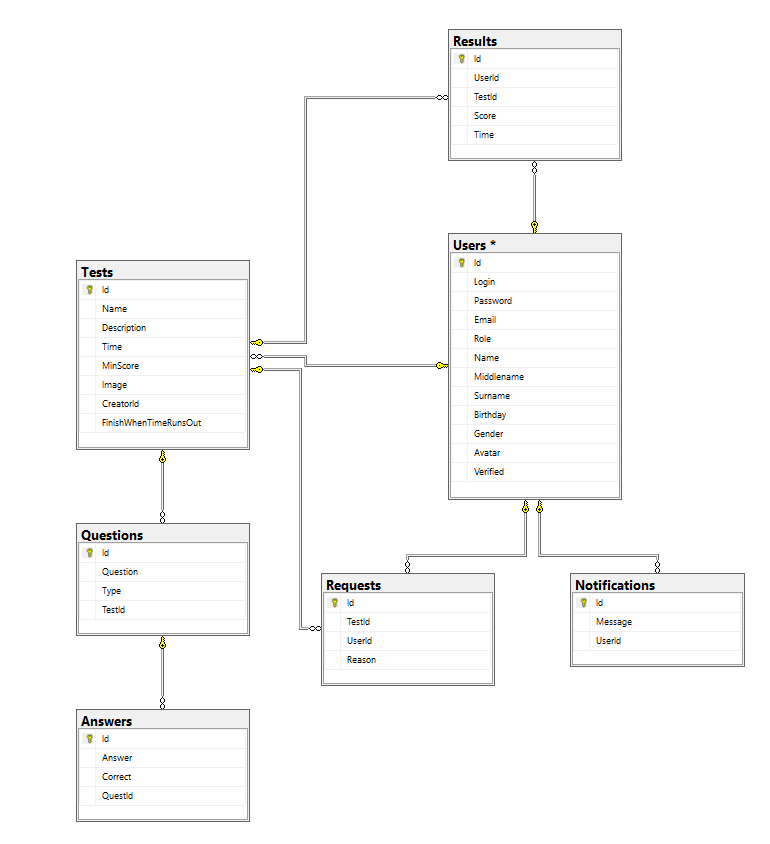


Рисунок 3.13 – Реляционная схема базы данных

Схема базы данных состоит из шести, описывающих сущности, отношений:

* + Users– Пользователи;
  + Tests – Тесты;
  + Questions – Вопросы;
  + Answers – Ответы;
  + Results – Результаты;
  + Requests – Запросы;
  + Notifications – Уведомления;

3.3 Инструкция по эксплуатации приложения

Приложение предназначено для удобного создания, редактирования и прохождения тестов других пользователей. Для доступа к приложению пользователь должен обязательно авторизоваться либо зарегистрироваться. Окно приложения с авторизованным пользователем представлено на рисунке 3.14



Рисунок 3.14 – Главное меню приложения

При нажатии на кнопку «Меню», пользователь может открывать и закрывать боковое меню. При нажатии на кнопку «Выйти из аккаунта» пользователя вернёт на окно авторизации. Левая кнопка над кнопкой «Выйти из аккаунта» меняет тему приложения, а правая язык.

Если нажать на кнопку «Мой профиль», то откроется страница с профилем текущего пользователя. Страница профиля представлена на рисунке 3.15

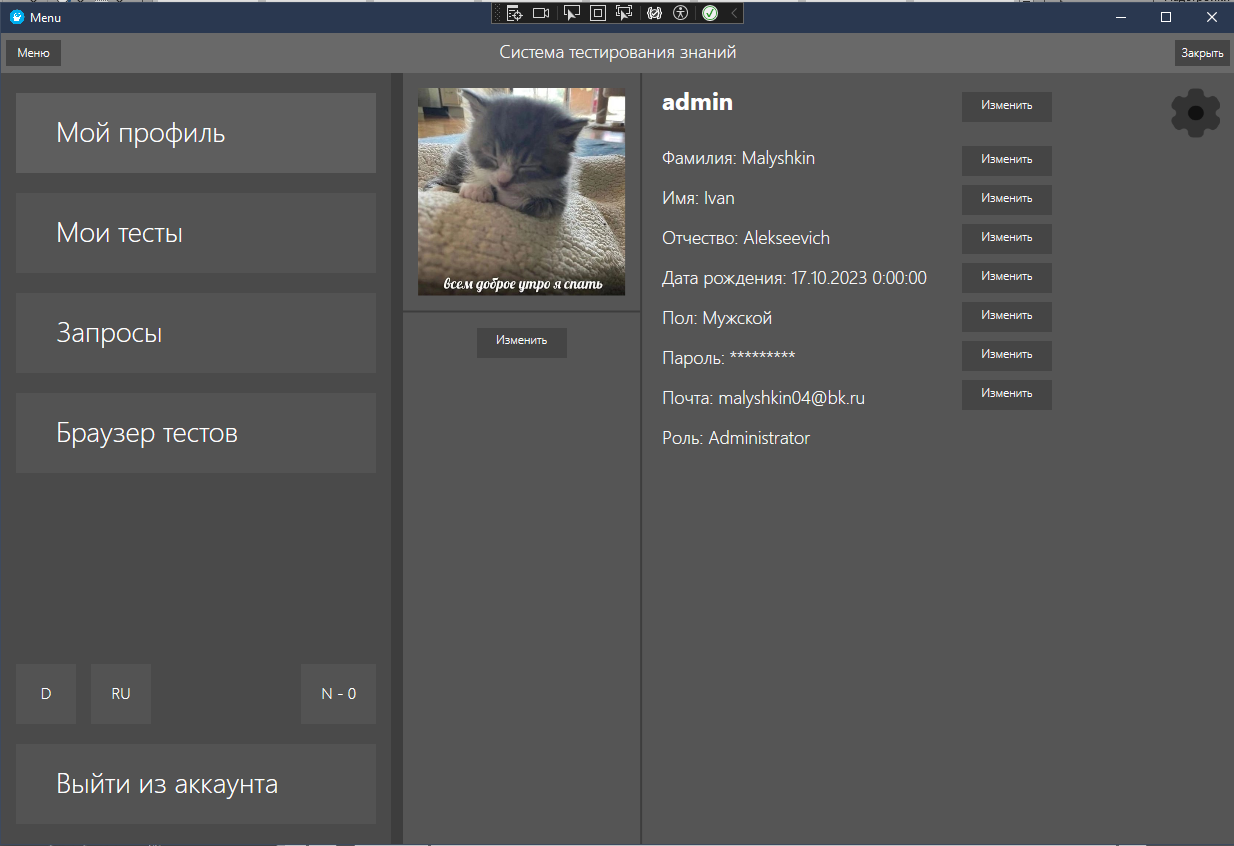


Рисунок 3.15 – Профиль пользователя

При нажатии на кнопку «Изменить» откроется меню редактирования параметра напротив которого находилась кнопка. С помощью «Шестерёнки» в правом верхнем углу можно скрыть кнопки изменения. Кнопка закрыть справа сверху позволяет закрыть текущее окно.

При нажатии на кнопку «Мои тесты» откроется страница с тестами пользователя. Окно представлено на рисунке 3.16

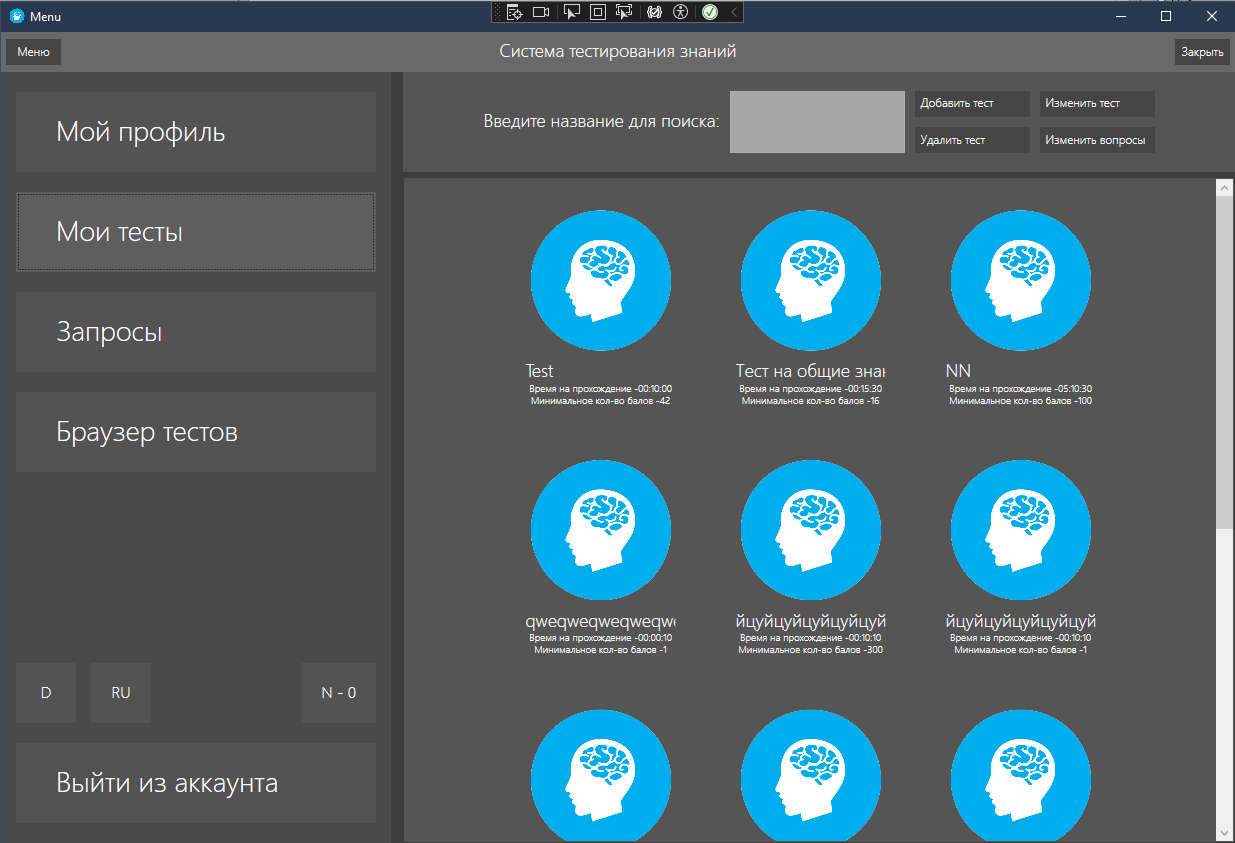


Рисунок 3.16 – Тесты пользователя

С помощью текстового окна сверху можно искать тесты. Также в верхнем меню есть кнопки для воздействия на тесты, чтобы выполнить действие которое написано на кнопке нужно сначала выделить тест.

При нажатии на кнопку «Браузер тестов» откроется соответствующая страница. Страница представлена на рисунке 3.17

Рисунок 3.17 – Браузер тестов

Сверху находится строка поиска тестов и галочка позволяющая увидеть уже пройденные тесты.

При двойном нажатии на тест откроется страница теста. Страница теста изображена на рисунке 3.18

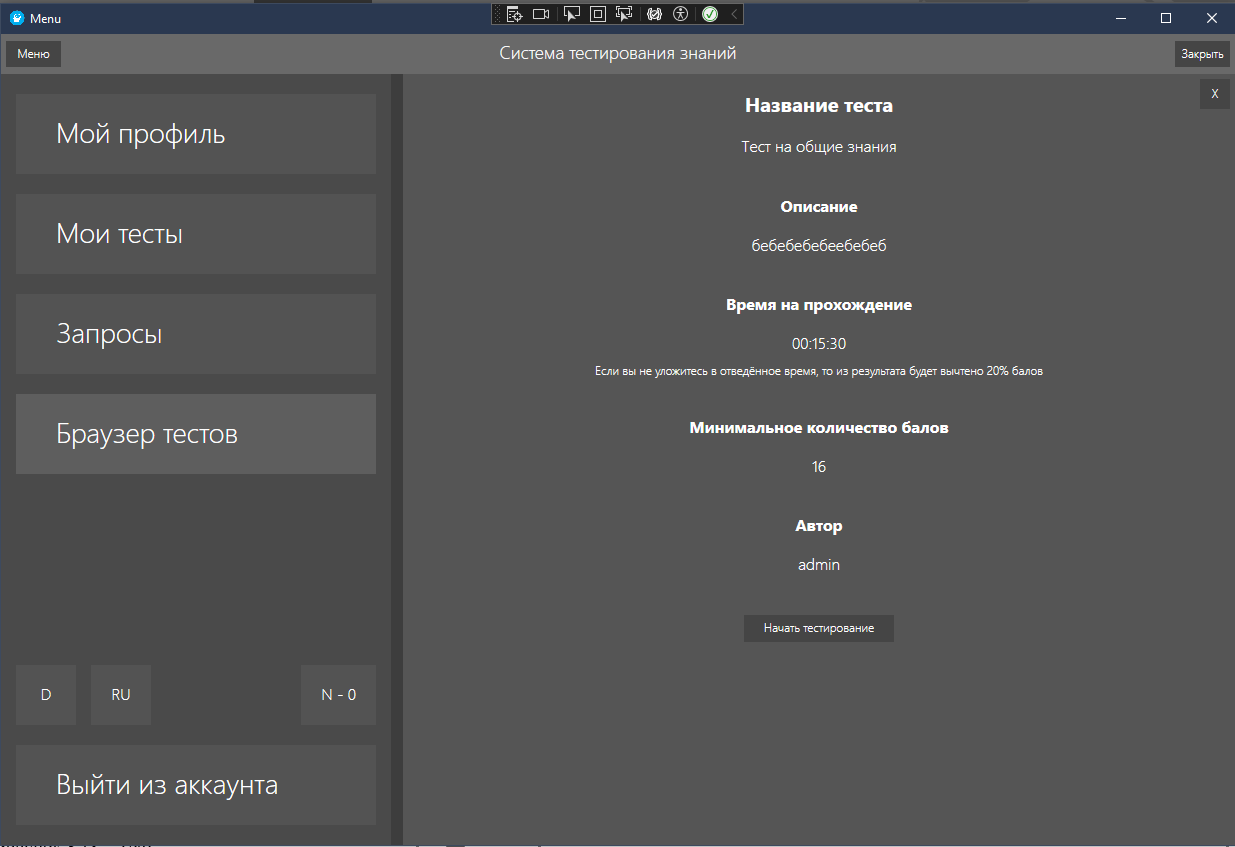


Рисунок 3.18 – Тест

На данной странице указана вся информация о тесте и кнопка его начала.

Если тест уже был пройден, то вместо кнопки начала будет кнопка отправки запроса на пере прохождение теста, а информация о тесте заменится на результаты предыдущего прохождения теста.

При нажатии на неё начнётся тестирование. Окно тестирования представлено на рисунке 3.19

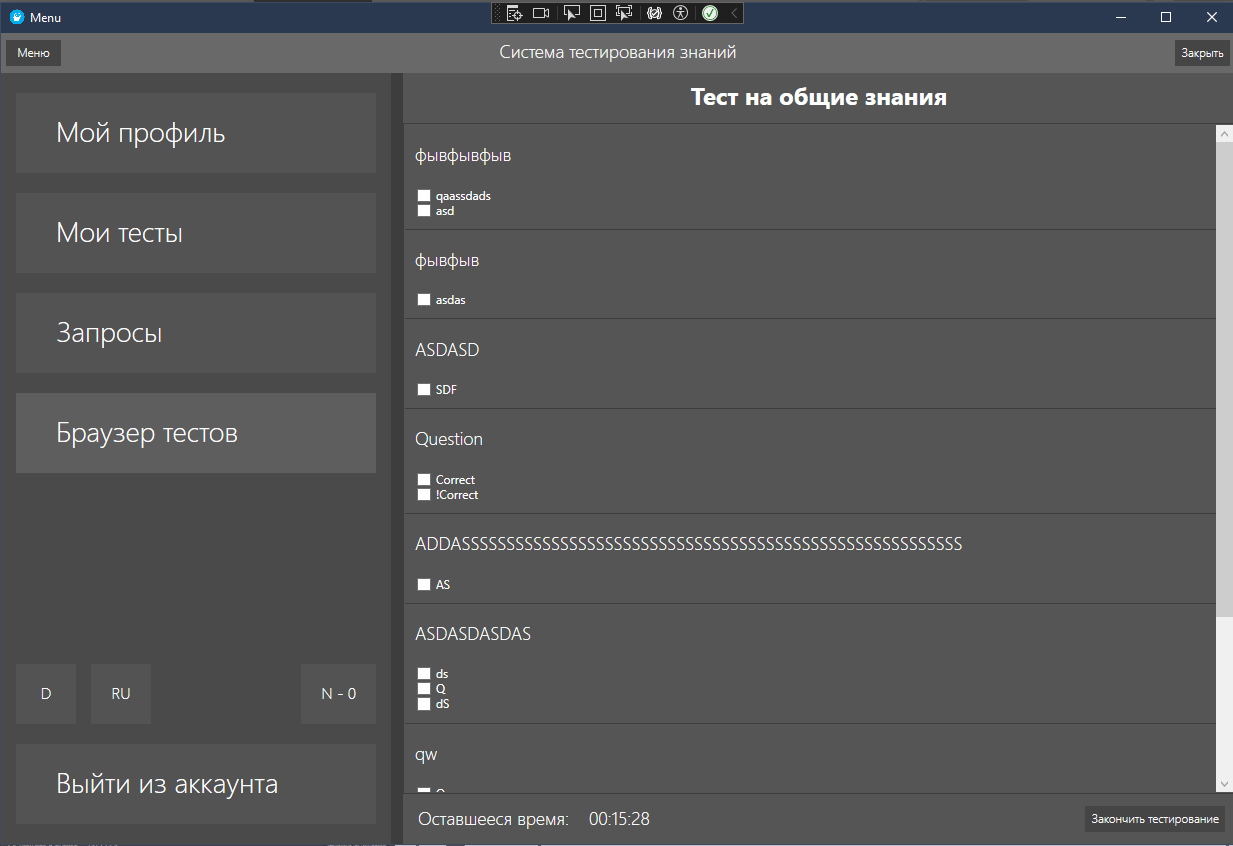


Рисунок 3.19 – Прохождение теста

На этой странице выведены все вопросы и ответы на них. Для выбора ответа на вопрос нужно нажать на поле рядом с ответом. Снизу пишется оставшееся время. Чтобы закончить тестирование нужно нажать кнопку «Закончить тестирование», для того чтобы она нажалась нужно обязательно ответить на все вопросы. При завершении теста выводится страница с результатами тестирования представленная на рисунке 3.20

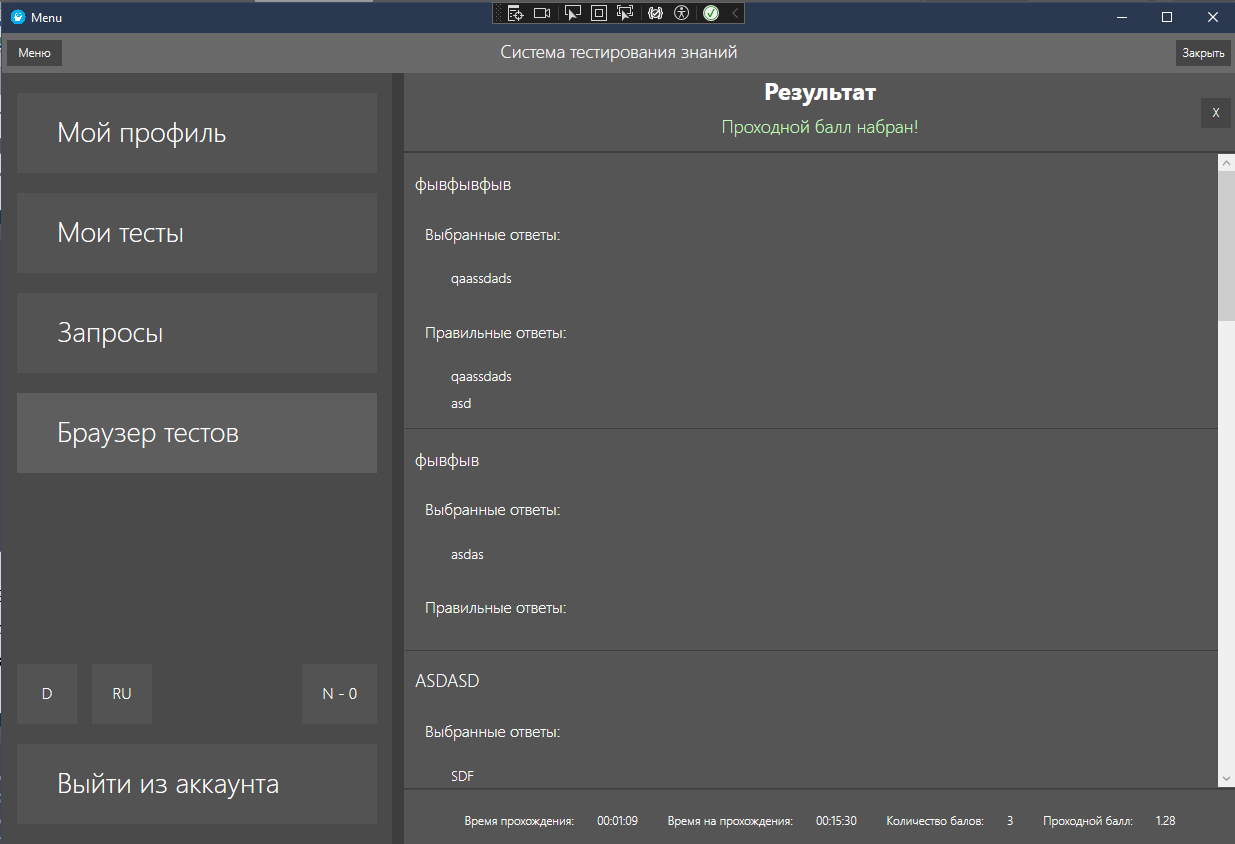


Рисунок 3.20 – Результаты теста

На данной странице находятся результаты тестирования, чтобы их закрыть нужно нажать на крестик справа сверху.

При нажатии на кнопку «Запросы» откроется страница с запросами на пере прохождение ваших тестов. Окно представлено на рисунке 3.21

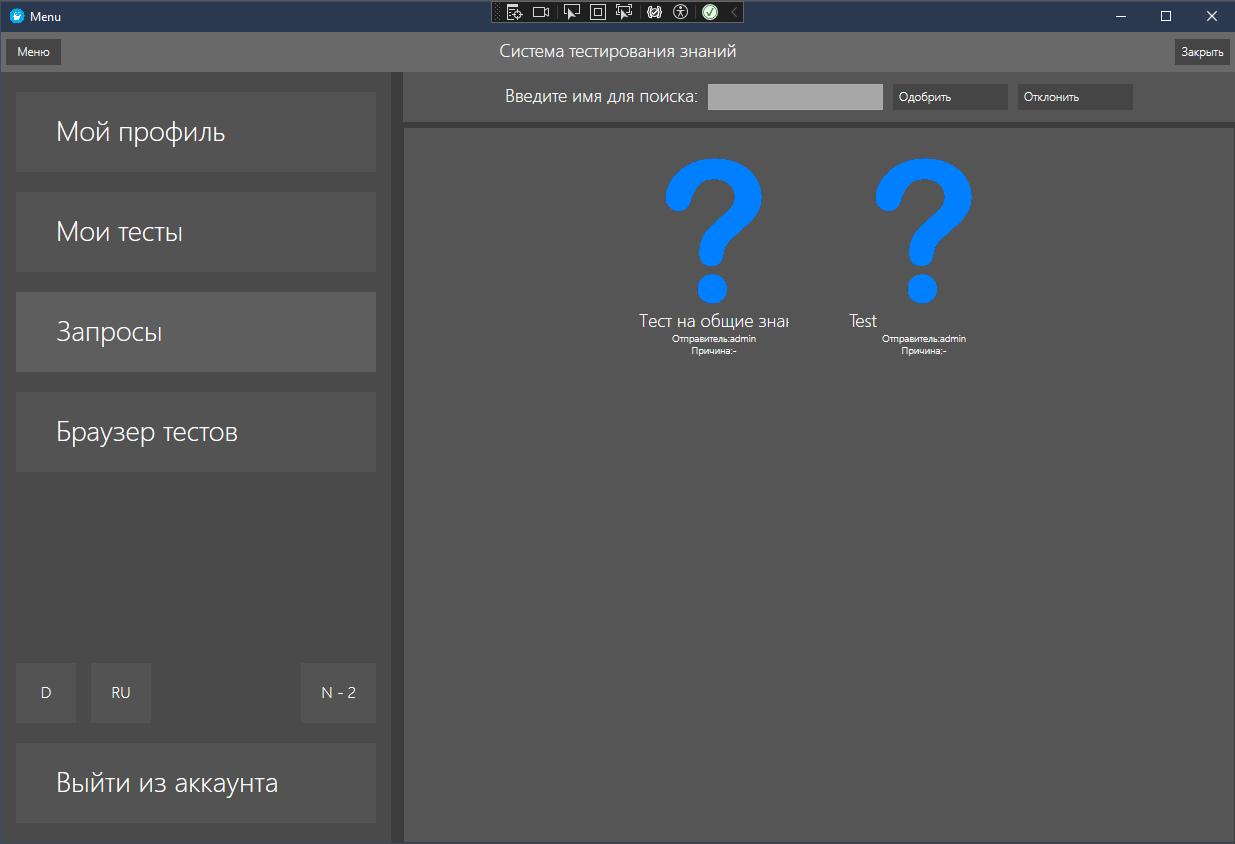


Рисунок 3.21 – Страница запросов на пере прохождение

На данной странице находятся запросы на пере прохождение ваших тестов чтобы одобрить или отклонить запрос нужно его выбрать ЛКМ и нажать сверху соответствующую кнопку.

При нажатии на кнопку «N - X» откроется страница с вашими уведомлениями. Окно представлено на рисунке 3.22

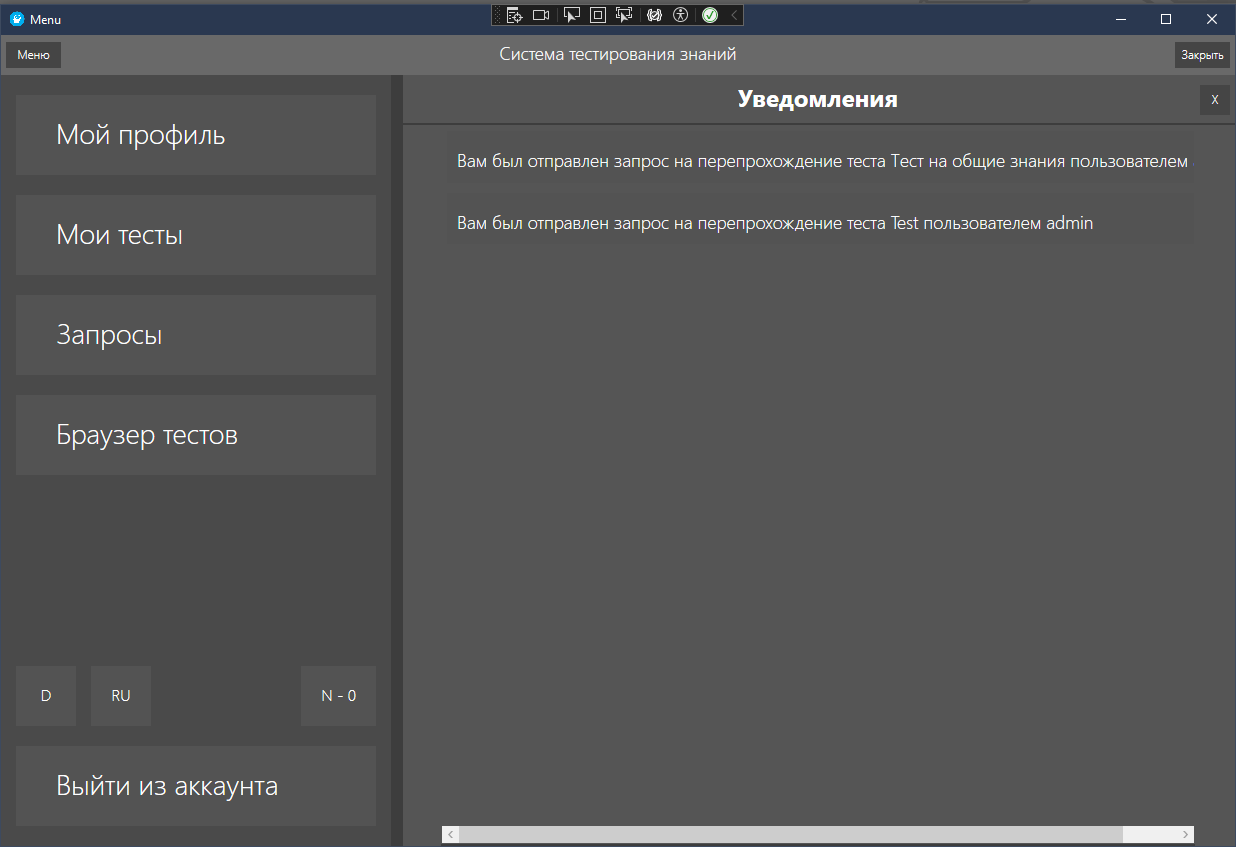


Рисунок 3.22 – Страница уведомлений

На данной странице находятся ваши уведомления (в основном о запросах которые вам отправили и о статусах запросов которые отправили вы). При открытии данной страницы уведомления автоматически удаляются.

4 Технико-экономический раздел

Основой расчета затрат на любой производственный процесс обычно является смета затрат. Смета затрат представляет собой сводный план всех расходов предприятия или организации на рассматриваемый период деятельности. Она определяет общую сумму издержек производства по видам используемых ресурсов, стадиям производственной деятельности, уровням управления предприятием и другим направлениям расходов. В смету включаются затраты основного и вспомогательного производства, связанные с изготовлением и продажей рассматриваемого продукта, а также на содержание административно-управленческого персонала, выполнение различных работ и услуг, в том числе и не входящих в основную производственную деятельность предприятия или организации.

Первым компонентом, входящим в сметный расчет, является материалы. В состав этих затрат принято включать стоимость материалов, которые будут проданы заказчику вместе с программным продуктом.

Совокупные затраты на материалы М*,* руб. рассчитываются по формуле 4.1:

, (4.1)

где *Мi* – стоимость, затраченная на каждый материал, руб.

Все материалы, используемые для изготовления проекта, их количество и стоимость включены, сведены в таблицу 4.1.

Таблица 4.1 –Материалы

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Цена за единицу, руб., коп. | Количество,  ед. изм. | Стоимость,  руб., коп. |
| USB-накопитель 4 Гб | 299,00 | 1 | 299,00 |
| Итого |  |  | 299,00 |

Стоимость отдельного материала М*i,* руб. рассчитывается по формуле 4.2:

, (4.2)

где Ni – стоимость отдельного материала за штуку, руб.;

Ki – количество отдельного материала, руб.

Подстановкой указанных выше значений в формулу 4.1 получено:

Вторым компонентом, включаемым в сметный расчет, являются затраты на вспомогательные материалы. В состав этих затрат принято включать стоимость расходуемых за период работ покупных инструментов и малоценного хозяйственного инвентаря.

Совокупные затраты на вспомогательные материалы и малоценный инвентарь ВМ*,* руб. рассчитываются по формуле 4.3:

, (4.3)

где *ВМi* – стоимость, затраченная на каждый вспомогательный материал, руб.

Все вспомогательные материалы, используемые в рамках работы непосредственно над проектом, их количество и стоимость включены в таблицу 4.2.

Таблица 4.2 – Вспомогательные материалы

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Цена за единицу, руб., коп. | Количество,  ед. изм. | Стоимость,  руб., коп. |
| Бумажный лист формата А4. | 0,8 | 100 | 80,00 |
| Ватман формата А0, шт. | 160,00 | 2 | 320,00 |
| Картридж для принтера, шт. | 1490,00 | 0,4 | 596,00 |
| Итого |  |  | 996,00 |

Стоимость отдельного вспомогательного материала *ВМi,* руб. рассчитывается по формуле 4.4:

, (4.4)

где Ni – стоимость отдельного вспомогательного материала за штуку, руб.;

Ki – количество отдельного вспомогательного материала, руб.

Подстановкой указанных выше значений в формулу 4.3 получено:

Третьим компонентом затрат на разработку являются затраты на энергетические ресурсы ЭР, руб, состоящие из затрат на электроэнергию и затрат на топливо, и рассчитываемые по формуле 4.5:

, (4.5)

где Э – затраты на электроэнергию, руб.;

Т – затраты на топливо, руб.

Так как разработка программного продукта не требует затрат на топливо, то стоимость топлива принимается равным нулю.

Затраты на электроэнергию вычисляются по формуле 4.6:

, (4.6)

где W – совокупная потребленная мощность, КВт×ч;

ЦЭ – стоимость одного КВт×ч электроэнергии, руб.; составляет 3,99руб.

Совокупная потребленная мощность *W* рассчитывается по формуле 4.8:

, (4.7)

где Wi – мощность, потребляемая отдельным устройством, КВт.

В таблице 4.3 представлены все возможные устройства, используемые в разработке продукта, потребляющие электроэнергию.

Таблица 4.3 – Используемые устройства

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование | Потребляемая мощность, КВт | Продолжительность эксплуатации в смену, час. | Количество смен в цикле производства (разработки), шт. | Потребленная мощность, КВт×ч |
| Компьютер | 0,5 | 8 | 20 | 80 |
| Принтер | 0,34 | 0,5 | 20 | 3,4 |
| Итого |  |  |  | 83,4 |

Потребленная мощность *Wi*, кВт×ч для каждого из устройств рассчитывается по формуле 4.8:

, (4.8)

где Pi – мощность устройства, КВт;

ti – продолжительность эксплуатации устройства за одну рабочую смену. час.;

КСi – количество смен, в течение которых использовалось устройство, шт.

Подстановкой значений из таблицы 4.3 в формулу 4.7 получено значение потребленной мощности:

Подстановкой указанных выше значений в формулу 4.6, получаем затраты на электроэнергию:

Подстановкой указанных выше значений в формулу 4.5, получаем затраты на энергетические ресурсы:

Четвертым этапом сметного расчета является расчет амортизации использованного в проекте оборудования.

Амортизация – это процесс постепенного переноса стоимости средств, по мере их износа, на стоимость продукции, которая производится. Амортизация распространяется на основные средства, основные фонды и нематериальные активы. Она исчисляется линейным, нелинейным методом, способом уменьшения остаточной стоимости, способом списания стоимости по сумме лет использования.

Амортизационные отчисления – отчисления части стоимости основных средств для возмещения их износа. Амортизационные отчисления включаются в издержки производства и производятся коммерческими организациями на основе установленных норм и балансовой стоимости основных средств.

Общий объем амортизационных отчислений *АМ*, руб., включаемый в расчет по проекту для всех устройств можно вычислить по формуле 4.9:

, (4.9)

где *Аi* – амортизационное отчисление для отдельного основного средства, руб.

Для разработки программного продукта потребовалось оборудование и нематериальные активы, перечисленные в таблице 4.4.

Таблица 4.4 – Информация по расчету амортизационных отчислений

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование основного средства | Изначальная стоимость, руб., коп. | Срок службы, лет | Сумма ежегодного амортиза-ционного отчисления,  руб., коп. | Период использования в проекте, кален. дн. | Сумма амортиза-ционного отчисления, руб., коп. |
| Офисный стол | 5 399,00 | 4 | 1 349,75 | 28 | 103,54 |
| Офисное кресло | 3 169,00 | 5 | 633,80 | 28 | 48,62 |
| Компьютер | 80 000,00 | 5 | 18 000,00 | 28 | 1 380,82 |
| Принтер | 6 200,00 | 5 | 1 240,00 | 28 | 95,12 |
| Windows 10 Pro | 9 800,00 | 2 | 4 900,00 | 28 | 375,89 |
| Microsoft Office professional 2021 | 59 590,00 | 6 | 9 931,67 | 28 | 761,88 |
| Итого |  |  |  |  | 2 765,87 |

Амортизационное отчисление *Аi,* руб., включаемое в расходы по проекту, для каждого из основных средств можно вычислить по формуле 4.10:

, (4.10)

где *АОi*– сумма планируемого амортизационного отчисления для основного средства за год, руб.;

*КДi*– количество календарных дней, в течение которых использовалось основное средство, дн.;

*365* – это количество календарных дней в году, дн.

Для всех основных средств и нематериальных активов рассчитывается сумма годового амортизационного отчисления *АОi*, руб. по формуле 4.11, которое каждый год будет составлять одну и ту же величину для одного и того же устройства.

, (4.11)

где *НСi* – начальная стоимость основного средства, руб.;

*Срi* – срок службы основного средства, лет.

Далее для каждого основного средства рассчитывается период его использования в проекте в календарных днях *КДi*, формула 4.12:

, (4.12)

где *КСi* – это количество рабочих дней (фактически, при односменном режиме работы – количество рабочих смен);

*5* – это количество рабочих дней на наделе;

*7* – это количество календарных дней в неделе.

Подстановкой значений из таблицы 4.4 в формулу 4.9 получено:

Следующим этапом сметного расчета является затраты на оплату труда.

Фонд оплаты труда ЗП, руб., включаемый в затраты по проекту, можно рассчитать по формуле 4.13:

, (4.13)

где ОЗП – основная заработная плата всех работников, участвовавших в проекте, руб.;

ДЗП – дополнительная заработная плата всех работников, участвовавших в проекте, руб.;

Основная заработная плата ОЗП, руб., включаемая в затраты проекта, будет складываться из двух частей, для расчёта можно воспользоваться формулой 4.14:

, (4.14)

где ОкЗП – сумма заработных плат всех рабочих, участвовавших в проекте, оплачиваемых по окладной системе оплаты труда, руб.;

СдЗП – сумма заработных плат всех рабочих, участвовавших в проекте, оплачиваемых по сдельной системе оплаты труда;

Заработная плата всех рабочих ОкЗП, руб., участвовавших в проекте, оплачиваемых по окладной системе оплаты труда, вычисляется по формуле 4.15:

, (4.15)

где ОкЗПi – заработная плата отдельного рабочего, участвовавшего в проекте, оплачиваемого по окладной системе оплаты труда, руб.

В таблице 4.5 сведены данные по работникам, принимавшим участие в проекте, оплачиваемым по окладной системе оплаты труда.

Таблица 4.5 – Заработная плата

|  |  |  |  |
| --- | --- | --- | --- |
| Должность, профессия  работника | Месячный оклад, руб., коп. | Отработано смен, шт. | Заработная плата, руб., коп. |
| Инженер-программист | 40 000,00 | 20 | 39 215,69 |
| Итого |  |  | 39 215,69 |

В рамках проекта заработная плата ОкЗПi, руб., заработанная каждым из работников, оплачиваемых по окладной системе оплаты труда, рассчитывается по формуле 4.16:

, (4.16)

где Оi – оклад работника, руб.,

КСi – количество смен (рабочих дней), которое он работал по проекту, шт., составляет 20 дней;

20,4 – это среднемесячное количество смен (рабочих дней), шт.

Подстановкой указанных выше значений в формулу 4.15 получено:

Заработная плата всех рабочих, оплачиваемых по сдельной системе оплаты труда, , руб., рассчитывается по формуле 4.17:

, (4.17)

где СдЗПi – сумма заработных плат всех рабочих, участвовавших в проекте, оплачиваемых по сдельной системе оплаты труда, руб.

Так как в разработке проекта не участвовали рабочие, оплачиваемые по сдельной системе оплаты труда, сдельная заработная плата равна нулю.

Подстановкой указанных выше значений в формулу 4.14 получено:

Предполагается, что в оплату труда по проекту должна быть заложена в том числе, и дополнительная заработная плата ДЗП, которую необходимо будет выплачивать тем же самым работникам, например, в связи с очередным отпуском, то ее расчет будем осуществлять исходя из того, что ее размер должен составлять какую-то разумную обоснованную долю от основной заработной платы.

Известно, что большинство работников находятся в очередных плановых отпусках в среднем около одного месяца в году, то логично предположить, что доля дополнительной заработной платы ДЗП будет около 1/12 от основной заработной платы, что составляет приблизительно 8-9%. Логично допустить, что кроме отпусков работникам в соответствии с действующим законодательством предстоит делать еще какие-либо выплаты в виде дополнительной заработной платы. По этой причине полученную долю можно несколько увеличить и выбирать ее в размере 9-12%. Таким образом, сумму затрат на дополнительную заработную плату ДЗП, руб., можно рассчитать по формуле 4.18:

, (4.18)

где ОЗП – основная заработная плата всех работников, участвовавших в проекте, руб.;

КДЗП – коэффициент, %.

В рамках проекта КДЗП выбран равным 10%.

Подстановкой указанных выше значений в формулу 4.18 получено:

Подстановкой указанных выше значений в формулу 4.13 получено:

Шестым компонентом, включаемым в сметный расчет, являются страховые взносы.

Социальное страхование – это система социальной защиты, задача которой – обеспечивать реализацию конституционного права экономически активных граждан на материальное обеспечение в старости, в случае болезни, полной или частичной утраты трудоспособности, потери кормильца, безработицы.

Уплата страховых взносов является прямой обязанностью организации-работодателя, при этом часть выплат носит персональный характер. Так, например, выплат на лицевой счет каждого из работников в пенсионном фонде зависит от размера его персональной заработной платы.

Общий размер страховых взносов СВ, руб., необходимых для перечисления во все страховые фонды и включаемый в состав затрат при производстве, можно вычислить как некоторый процент от фонда заработной платы, для расчёта можно воспользоваться формулой 4.19:

, (4.19)

где ЗП – фонд оплаты труда по проекту, рассчитанный ранее (руб.), составляет 38 823,53 руб.;

ССВ – ставка (размер налога) страховых взносов, %.

Ставка страховых взносов ССВ на 2022 год составляет 30,2%.

Подстановкой указанных выше значений в формулу 4.19 получено:

Следующим этапом сметного расчета является накладные расходы.

В зависимости от размеров предприятия или организации накладные расходы НР, руб., могут варьироваться в достаточно широком диапазоне. Поэтому в обычной практике в сметных расчетах их принято отражать в процентном отношении к заработной плате основных работников, для расчёта можно воспользоваться формулой 4.20:

, (4.20)

где ЗП – фонд оплаты труда по проекту, руб.;

СНР – размер накладных расходов, %.

Размер накладных расходов СНР составляет 10%.

Подстановкой указанных выше значений в формулу 4.20 получено:

Восьмым этапом сметного расчета является расчет себестоимости.

Себестоимость – это текущие затраты на производство товара или услуги, запуск их в обращение и реализацию. Расчет и анализ себестоимости продукции является важнейшей задачей любого предприятия и входит в систему управленческого учета, так как именно себестоимость лежит в основе большинства управленческих решений.

Себестоимость Сб, руб. программного продукта определяется по формуле 4.21.

Сб = М + ВМ + ЭР + АМ + ЗП + СВ +НР, (4.21)

где *М* – затраты на материалы руб.;

*ВМ* – затраты на вспомогательные материалы, руб.;

*ЭР* – энергетические затраты, руб.;

*АМ* – затраты на амортизацию, руб.;

*ЗП* – затраты на заработную плату, руб.;

*СВ* – страховые взносы, уплаченные работодателем с фонда оплаты труда, руб.;

*НР* – накладные расходы, возникающие как непредусмотренные всеми предшествующими статьями, руб.;

Подстановкой указанных выше значений в формулу 4.21 получено:

После расчета себестоимости (всех возможных затрат на разработку и производство), к ней добавляется необходимый объем прибыли, он и будет являться девятым продуктом сметного расчета.

Объем прибыли обычно определяется как процент от рассчитанной себестоимости. Объем прибыли П, руб. можно рассчитать по формуле 4.22:

, (4.22)

где Сб – себестоимость производства или разработки, руб.;

СП – процент прибыли, на которую рассчитывает организатор коммерческого предприятия, %.

Процент прибыли СП, на которую рассчитывает организатор коммерческого предприятия составляет 30%.

Подстановкой указанных выше значений в формулу 4.22 получено:

Следующим этапом сметного расчета является налог на добавленную стоимость.

Налог на добавленную стоимость (НДС) – косвенный налог, форма изъятия в бюджет государства части стоимости товара, работы или услуги, которая создаётся на всех стадиях процесса производства товаров, работ и услуг и вносится в бюджет по мере реализации.

Для расчета налога на добавленную стоимость НДС, руб. можно воспользоваться формулой 4.23:

, (4.23)

где Сб – себестоимость разработки, руб.;

П – прибыль, руб.;

СНДС – ставка налога на добавленную стоимость, %.

Ставка налога СНДС на добавленную стоимость составляет 20%.

Подстановкой указанных выше значений в формулу 4.23 получено:

Конечная стоимость разработки *С*, руб. рассчитывается как сумма себестоимости, прибыли и НДС, для расчёта можно воспользоваться формулой 4.24:

, (4.24)

где Сб – себестоимость разработки, руб.

П – прибыль. руб.;

НДС – сумма налога на добавленную стоимость, руб.;

Подстановкой указанных выше значений в формулу 4.24 получено:

Исходя из вышеперечисленных расчетов следует, что программный продукт является конкурентоспособным, его стоимость составляет 101 200,44 руб.

Данный продукт является востребованным для пользователей, так как функциональность приложения несколько больше, чем у аналогов.

5 Раздел охраны труда

Охрана труда - система законодательных актов, социально-экономических, организационных, технических, гигиенических и лечебно-профилактических мероприятий и средств, обеспечивающих безопасность, сохранение здоровья и работоспособности человека в процессе труда. Научно-технический прогресс внес серьезные изменения в условия производственной деятельности работников умственного труда. Их труд стал более интенсивным, напряженным, требующим значительных затрат умственной, эмоциональной и физической энергии. Это потребовало комплексного решения проблем эргономики, гигиены и организации труда, регламентации режимов труда и отдыха.

Рабочее место - это часть пространства, в котором инженер осуществляет трудовую деятельность, и проводит большую часть рабочего времени. Рабочее место, хорошо приспособленное к трудовой деятельности инженера, правильно и целесообразно организованное, в отношении пространства, формы, размера обеспечивает ему удобное положение при работе и высокую производительность труда при наименьшем физическом и психическом напряжении.

При правильной организации рабочего места производительность труда инженера возрастает с 8 до 20 процентов.

Согласно ГОСТ 12.2.032-78 конструкция рабочего места и взаимное расположение всех его элементов должно соответствовать антропометрическим, физическим и психологическим требованиям. Большое значение имеет также характер работы [8].

Для рабочего места с компьютером, оснащенным жидкокристаллическим или плазменным экраном, рекомендуется обеспечить площадь не менее 4,5 кв. м. Расстояние между столами с мониторами (от задней части одного монитора до экрана другого) должно быть не менее 2 метров. Монитор следует размещать на расстоянии от 50 до 70 см от глаз программиста. Характеристики рабочего стола для сотрудника включают возможность регулировки высоты рабочей поверхности или точно установленную высоту в 72,5 см, ширину стола варьирующуюся от 80, 100, 120 до 140 см, а также глубину рабочей поверхности в 80 или 100 см. Пространство под столешницей (для ног) должно иметь высоту и ширину не менее 50 см, глубина на уровне колен должна быть не менее 45 см, а на уровне вытянутых ног - не менее 65 см. Надлежащее освещение рабочего места является важным аспектом трудовой деятельности, влияющим на работоспособность. Отсутствие должного освещения может вызывать напряжение зрения, снижение внимания и преждевременную усталость. Слишком яркое освещение также нежелательно, поскольку может вызывать ослепление, раздражение и дискомфорт в глазах. При искусственном освещении обычно используются два типа ламп: накаливания и люминесцентные.

Известно, что шум ухудшает условия труда и оказывает вредное воздействие на организм человека. Согласно ГОСТ 12.1.003-88 «Шум для помещений расчетчиков и программистов, уровни шума не должны превышать соответственно: 71, 61, 54, 49, 45, 42, 40, 38 дБ», информация из работы [9].

В процессе эксплуатации компьютерной техники выделяется значительное количество тепла, что может представлять опасность возникновения пожара. Приборы, используемые для обслуживания, устройства электропитания и системы кондиционирования воздуха также могут стать источниками возгорания. Особенно опасны электроизоляционные материалы, применяемые для защиты от механических повреждений электронных компонентов. В связи с этим, зоны, где присутствует компьютерная техника, относятся к категории пожароопасных "В".

В случае пожара людям необходимо немедленно покинуть помещение. В помещениях с компьютерной техникой не рекомендуется использовать воду или пену, поскольку они могут нанести ущерб или полностью вывести из строя дорогостоящее электронное оборудование. Для тушения пожаров рекомендуется применять углекислотные и порошковые огнетушители, которые обладают высокой скоростью тушения, продолжительным действием, а также могут быть безопасно использованы для потушения пожаров в электроустановках. Эти огнетушители являются эффективным средством борьбы с огнем. Воду допускается использовать только во вспомогательных помещениях, где отсутствует электронное оборудование или информация из основной рабочей среды.

Заключение

По итогу работы было разработано клиент-серверное приложение «Система тестирования знаний». Приложение является адаптивным и представляет из себя окно с меню внутри которого можно менять вкладки внутри окна такие как профиль, мои тесты(редактирование и создание тестов), запросы, уведомления и браузер тестов в котором можно проходить свои и чужие тесты.

Главными достоинствами разработанного приложения можно считать уникальность приложения, систему прохождения и составления тестов, красивый и удобный пользовательский интерфейс с выбором нескольких стилей и возможность кастомизировать свой профиль.

У приложения есть перспективы развития, а именно, внедрение различных анимаций, добавление стилей приложения и звуков, а также потом можно будет добавить различные типы вопросов.

Разработанное приложение можно использовать для прохождения и составления тестов. Так же приложение можно использовать в учебных целях. Приложение может проиллюстрировать различные методы рисования, объектно-ориентированный подход. В приложение используются различного рода алгоритмы, на которые можно опираться при разработке других клиент-серверных приложений.

Список используемых источников

1. <https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/xaml/?view=netdesktop-7.0>
2. <https://itanddigital.ru/historycoding>
3. [https://studopedia.ru/19\_420644\_istoriya-programmirovaniya-i-algoritmicheskih-yazikov.html#:~:text=Первые%20алгоритмические%20языки%20эпохи%20ЭВМ,комитете%20под%20руководством%20Питера%20Наура](https://studopedia.ru/19_420644_istoriya-programmirovaniya-i-algoritmicheskih-yazikov.html#:~:text=Первые алгоритмические языки эпохи ЭВМ,комитете под руководством Питера Наура)
4. https://moretestov.com
5. https://www.qa-book.cloud/testovaya-dokumentaciya/test-keis
6. <https://metanit.com/sharp/wpf/11.php>
7. <https://www.garshin.ru/it/computer-languages/object-pioners.html>
8. <https://otherreferats.allbest.ru/life/00130223_0.html>
9. https://дцо.рф/opisanie-rabochego-mesta-programmista-na-predpriyatii/

Приложение А

Программный код окна MainWindow

using QQCourse.Pages;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Animation;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace QQCourse

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

private List<Page> ActivePages;

private int CurrentPageIndex;

private bool isOpen = true;

private int curTheme;

private int curLang;

protected System.Timers.Timer \_Timer;

protected System.Timers.Timer \_Timer1;

public MainWindow()

{

InitializeComponent();

curTheme = Properties.Settings.Default.Theme;

curLang = Properties.Settings.Default.Lang;

ActivePages = new List<Page>();

CurrentPageIndex = -1;

Core.AppMainWindow = this;

sideMenuDockPanel.Width = 400;

\_Timer = new System.Timers.Timer(500);

\_Timer.Elapsed += OnTimer;

\_Timer.AutoReset = false;

\_Timer.Enabled = false;

\_Timer1 = new System.Timers.Timer(500);

\_Timer1.Elapsed += OnTimer1;

\_Timer1.AutoReset = false;

\_Timer1.Enabled = false;

SetTheme(curTheme);

SetLang(curLang);

if (Core.CurrentUser.Birthday.Value.Date==DateTime.Now.Date)

{

WelcomeText.Text = "С днём рождения!";

}

else

{

WelcomeText.Text = "Добро пожаловать!";

}

}

private int GetCurrentPageIndexByType(Type PageType)

{

int Index = ActivePages.Count - 1;

while (Index > 0 && ActivePages[Index].GetType() != PageType)

{

Index--;

}

return Index;

}

public void ShowPage(Type PageType)

{

Page Page;

if (PageType != null)

{

CurrentPageIndex = GetCurrentPageIndexByType(PageType);

if (CurrentPageIndex < 0)

{

Page = (Page)Activator.CreateInstance(PageType);

ActivePages.Add(Page);

CurrentPageIndex = ActivePages.Count - 1;

}

else

{

Page = ActivePages[CurrentPageIndex];

}

RootFrame.Navigate(Page);

WelcomeText.Visibility = Visibility.Hidden;

}

}

public void CloseAllPage()

{

ActivePages.RemoveAll(p => true);

CurrentPageIndex = -1;

RootFrame.Navigate(null);

if (Core.CurrentUser.Birthday.Value.Date == DateTime.Now.Date)

{

WelcomeText.Text = "С днём рождения!";

}

else

{

WelcomeText.Text = "Добро пожаловать!";

}

}

private void RootFrame\_LoadCompleted(object sender, NavigationEventArgs e)

{

}

private void SetTheme(int cur)

{

//if (cur == null) cur = 0;

switch (cur)

{

case 0:

var uri = new Uri("Themes/DarkTheme.xaml", UriKind.Relative);

ResourceDictionary resourceDictionary = Application.LoadComponent(uri) as ResourceDictionary;

Application.Current.Resources.Clear();

Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);

ThemeButton.Content = "D";

break;

case 1:

uri = new Uri("Themes/LightTheme.xaml", UriKind.Relative);

resourceDictionary = Application.LoadComponent(uri) as ResourceDictionary;

Application.Current.Resources.Clear();

Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);

ThemeButton.Content = "L";

break;

}

Properties.Settings.Default.Theme = cur;

Properties.Settings.Default.Save();

}

private void SetLang(int cur)

{

switch (cur)

{

case 0:

var uri = new Uri("Langs/RussianLanguage.xaml", UriKind.Relative);

ResourceDictionary resourceDictionary = Application.LoadComponent(uri) as ResourceDictionary;

Application.Current.Resources.Clear();

Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);

LangButton.Content = "RU";

break;

case 1:

uri = new Uri("Langs/EnglishLanguage.xaml", UriKind.Relative);

resourceDictionary = Application.LoadComponent(uri) as ResourceDictionary;

Application.Current.Resources.Clear();

Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);

LangButton.Content = "EN";

break;

}

Properties.Settings.Default.Lang = cur;

Properties.Settings.Default.Save();

RootFrame.UpdateLayout();

}

private void SettingsButton\_Click(object sender, RoutedEventArgs e) {

curTheme++;

if (curTheme > 1) curTheme = 0;

SetTheme(curTheme);

Properties.Settings.Default.Theme = curTheme;

Properties.Settings.Default.Save();

}

private void ChangeButtonsStyles(Button button)

{

myProfileButton.Style = (Style)FindResource("SideMenuButton");

myTestsButton.Style = (Style)FindResource("SideMenuButton");

TestsBrowserButton.Style = (Style)FindResource("SideMenuButton");

if(button!= null) {

button.Style = (Style)FindResource("SelectedSideMenuButton");

}

}

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

{

CloseAllPage();

ShowPage(typeof(Pages.ProfilePage));

ChangeButtonsStyles(myProfileButton);

}

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

{

CloseAllPage();

ShowPage(typeof(Pages.MyTestsPage));

ChangeButtonsStyles(myTestsButton);

}

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

{

CloseAllPage();

ShowPage(typeof(Pages.TestsBrowserPage));

ChangeButtonsStyles(TestsBrowserButton);

}

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

{

AuthWindow a = new AuthWindow();

Core.CurrentUser = null;

a.Show();

Close();

}

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

{

CloseAllPage();

WelcomeText.Visibility = Visibility.Visible;

ChangeButtonsStyles(null);

}

protected virtual void OnTimer(Object source, System.Timers.ElapsedEventArgs e)

{

Dispatcher.Invoke((Action)(() =>

{

Grid.SetColumn(RootFrame, 0);

Grid.SetColumnSpan(RootFrame, 3);

Grid.SetColumn(WelcomeText, 0);

Grid.SetColumnSpan(WelcomeText, 3);

Grid.SetColumn(sideMenuDockPanel, 0);

}));

}

protected virtual void OnTimer1(Object source, System.Timers.ElapsedEventArgs e)

{

Dispatcher.Invoke((Action)(() =>

{

menuPageButton.IsEnabled = true;

}));

}

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

{

if (isOpen)

{

beginSideMenuAnimation(400, 0);

\_Timer.Start();

menuPageButton.IsEnabled = false;

\_Timer1.Start();

}

else

{

beginSideMenuAnimation(0, 400);

Grid.SetColumn(RootFrame, 2);

Grid.SetColumn(WelcomeText, 2);

menuPageButton.IsEnabled = false;

\_Timer1.Start();

//Grid.SetColumnSpan(RootFrame, 0);

}

isOpen = !isOpen;

}

private void beginSideMenuAnimation(int from,int to)

{

DoubleAnimation widthAnimation = new DoubleAnimation(from, to, new Duration(TimeSpan.FromSeconds(0.5)));

widthAnimation.EasingFunction = new CubicEase { EasingMode = EasingMode.EaseIn };

sideMenuDockPanel.BeginAnimation(WidthProperty, widthAnimation);

}

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

{

curLang++;

if (curLang > 1) curLang = 0;

SetLang(curLang);

}

private void RootFrame\_Navigated(object sender, NavigationEventArgs e)

{

}

}

}

Программный код окна RegistrationWindow

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace QQCourse

{

/// <summary>

/// Interaction logic for RegistrationWindow.xaml

/// </summary>

public partial class RegistrationWindow : Window

{

private string Captcha;

private int CaptchaLength = 5;

public RegistrationWindow()

{

InitializeComponent();

CreateCaptcha();

}

private void CreateCaptcha()

{

Captcha = "";

Random Gen = new Random();

string Alphabet = "1234567890QWERTYUIOPASDFGHJKLZXCVBNMqwertyuiopasdfghjklzxcvbnm";

for (int i = 0; i < CaptchaLength; i++)

{

Captcha += Alphabet[Gen.Next(Alphabet.Length)];

}

CaptchaImage.Text = Captcha;

}

private void DBSaveException()

{

MessageBox.Show(FindResource("SaveDBException").ToString(), FindResource("Error").ToString(), MessageBoxButton.OK, MessageBoxImage.Error);

Core.CancelChanges(Core.Database.Users);

}

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

{

CreateCaptcha();

}

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

{

}

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

{

AuthWindow registrationWindow = new AuthWindow();

registrationWindow.Show();

this.Close();

}

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

{

if (CheckInfo() && SaveInfo())

{

MainWindow mainWindow = new MainWindow();

Core.CurrentUser = Core.Database.Users

.SingleOrDefault(U => U.Login == LoginTextBox.Text && U.Password == (PasswordPasswordBox.IsVisible ? PasswordPasswordBox.Password : PasswordTextBox.Text));

mainWindow.Show();

this.Close();

}

}

private bool CheckInfo()

{

if (CaptchaTextBox.Text != Captcha)

{

MessageBox.Show("Капча введена неверно!", FindResource("Notification").ToString(), MessageBoxButton.OK, MessageBoxImage.Information);

CaptchaTextBox.Focus();

CreateCaptcha();

return false;

}

if (LoginTextBox.Text == String.Empty)

{

MessageBox.Show("Поле логин не заполнено!", FindResource("Notification").ToString(), MessageBoxButton.OK, MessageBoxImage.Information);

LoginTextBox.Focus();

return false;

}

if (LoginTextBox.Text == String.Empty)

{

MessageBox.Show("Поле логин не заполнено!", FindResource("Notification").ToString(), MessageBoxButton.OK, MessageBoxImage.Information);

LoginTextBox.Focus();

return false;

}

string password = PasswordPasswordBox.IsVisible ? PasswordPasswordBox.Password : PasswordTextBox.Text;

Regex regex1 = new Regex("^(?=.\*?[A-Z])(?=.\*?[a-z])(?=.\*?[0-9])(?=.\*?[#?!@$%^&\*-]).{8,}$");

if (password == String.Empty)

{

MessageBox.Show("Поле пароль не заполнено!", FindResource("Notification").ToString(), MessageBoxButton.OK, MessageBoxImage.Information);

PasswordTextBox.Focus();

PasswordPasswordBox.Focus();

return false;

}

if (!regex1.IsMatch(password))

{

MessageBox.Show("Пароль не соответствует паттерну! (Минимум 8 символов один из них должен быть заглавным, один прописным, один цифрой и один спец. символ)", FindResource("Notification").ToString(), MessageBoxButton.OK, MessageBoxImage.Information);

PasswordTextBox.Focus();

PasswordPasswordBox.Focus();

return false;

}

Regex regex = new Regex(@"^([\w\.\-]+)@([\w\-]+)((\.(\w){2,3})+)$");

if(EmailTextBox.Text == String.Empty)

{

MessageBox.Show("Поле почта не заполнено!", FindResource("Notification").ToString(), MessageBoxButton.OK, MessageBoxImage.Information);

EmailTextBox.Focus();

return false;

}

if (!regex.IsMatch(EmailTextBox.Text))

{

MessageBox.Show(FindResource("InvalidEmailPattern").ToString(), FindResource("Notification").ToString(), MessageBoxButton.OK, MessageBoxImage.Information);

EmailTextBox.Focus();

return false;

}

if (SurnameTextBox.Text == String.Empty)

{

MessageBox.Show("Поле фамилия не заполнено!", FindResource("Notification").ToString(), MessageBoxButton.OK, MessageBoxImage.Information);

SurnameTextBox.Focus();

return false;

}

if (NameTextBox.Text == String.Empty)

{

MessageBox.Show(FindResource("NameFieldEmpty").ToString(), FindResource("Notification").ToString(), MessageBoxButton.OK, MessageBoxImage.Information);

NameTextBox.Focus();

return false;

}

if (MiddlenameTextBox.Text == String.Empty)

{

MessageBox.Show("Поле отчество не заполнено!", FindResource("Notification").ToString(), MessageBoxButton.OK, MessageBoxImage.Information);

MiddlenameTextBox.Focus();

return false;

}

if (BirthdayDatePicker.Text == String.Empty)

{

MessageBox.Show("Поле дата рождения не заполнено!", FindResource("Notification").ToString(), MessageBoxButton.OK, MessageBoxImage.Information);

BirthdayDatePicker.Focus();

return false;

}

if (GenderComboBox.Text == String.Empty)

{

MessageBox.Show("Поле пол не заполнено!", FindResource("Notification").ToString(), MessageBoxButton.OK, MessageBoxImage.Information);

GenderComboBox.Focus();

return false;

}

try

{

if (Core.Database.Users.FirstOrDefault(u => u.Login == LoginTextBox.Text) != null)

{

MessageBox.Show(FindResource("LoginAlreadyBusy").ToString(), FindResource("Notification").ToString(), MessageBoxButton.OK, MessageBoxImage.Information); ;

LoginTextBox.Focus();

return false;

}

if (Core.Database.Users.FirstOrDefault(u => u.Email == EmailTextBox.Text) != null)

{

MessageBox.Show(FindResource("EmailAlreadyBusy").ToString(), FindResource("Notification").ToString(), MessageBoxButton.OK, MessageBoxImage.Information);

EmailTextBox.Focus();

return false;

}

}

catch

{

MessageBox.Show(FindResource("ConnectDBException").ToString(), FindResource("Error").ToString(),

MessageBoxButton.OK, MessageBoxImage.Error, MessageBoxResult.OK);

}

return true;

}

private bool SaveInfo()

{

try

{

Data.Users user = new Data.Users();

user.Id = Core.VOID;

user.Login = LoginTextBox.Text;

user.Name = NameTextBox.Text;

user.Surname = SurnameTextBox.Text;

user.Middlename = MiddlenameTextBox.Text;

user.Role = "User";

user.Email = EmailTextBox.Text;

user.Password = PasswordPasswordBox.IsVisible ? PasswordPasswordBox.Password : PasswordTextBox.Text;

user.Birthday= DateTime.Parse(BirthdayDatePicker.Text);

user.Gender = GenderComboBox.Text == FindResource("Male").ToString() ? true : false;

user.Verified = false;

Core.Database.Users.Add(user);

Core.Database.SaveChanges();

}catch

{

DBSaveException();

return false;

}

return true;

}

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

{

if (PasswordPasswordBox.Visibility == Visibility.Visible)

{

PasswordTextBox.Text = PasswordPasswordBox.Password;

PasswordTextBox.Width = PasswordPasswordBox.ActualWidth;

PasswordPasswordBox.Visibility = Visibility.Hidden;

PasswordTextBox.Visibility = Visibility.Visible;

}

else

{

PasswordPasswordBox.Password = PasswordTextBox.Text;

PasswordTextBox.Width = 0;

PasswordPasswordBox.Visibility = Visibility.Visible;

PasswordTextBox.Visibility = Visibility.Hidden;

}

}

private void CaptchaImage\_TextChanged(object sender, TextChangedEventArgs e)

{

}

}

}

Программный код окна AuthWindow

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Web.UI.WebControls;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using System.Windows.Threading;

namespace QQCourse

{

/// <summary>

/// Interaction logic for AuthWindow.xaml

/// </summary>

public partial class AuthWindow : Window

{

public AuthWindow()

{

InitializeComponent();

SetLang(Properties.Settings.Default.Lang);

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

//BlockingTime = StandartBlockingTime.Subtract(DateTime.Now - Properties.Settings.Default.BlockingStartTime);

}

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

{

if (PasswordPasswordBox.Visibility == Visibility.Visible)

{

PasswordTextBox.Text = PasswordPasswordBox.Password;

PasswordTextBox.Width = PasswordPasswordBox.ActualWidth;

PasswordPasswordBox.Visibility = Visibility.Hidden;

PasswordTextBox.Visibility = Visibility.Visible;

}

else

{

PasswordPasswordBox.Password = PasswordTextBox.Text;

PasswordTextBox.Width = 0;

PasswordPasswordBox.Visibility = Visibility.Visible;

PasswordTextBox.Visibility = Visibility.Hidden;

}

}

private void SetLang(int? cur)

{

if (cur == null) cur = 0;

switch (cur)

{

case 0:

var uri = new Uri("Langs/RussianLanguage.xaml", UriKind.Relative);

ResourceDictionary resourceDictionary = Application.LoadComponent(uri) as ResourceDictionary;

Application.Current.Resources.Clear();

Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);

break;

case 1:

uri = new Uri("Langs/EnglishLanguage.xaml", UriKind.Relative);

resourceDictionary = Application.LoadComponent(uri) as ResourceDictionary;

Application.Current.Resources.Clear();

Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);

break;

}

}

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

{

string Login = LoginTextBox.Text;

string Password = PasswordPasswordBox.Visibility == Visibility.Visible ?

PasswordPasswordBox.Password : PasswordTextBox.Text;

try

{

Core.CurrentUser = Core.Database.Users

.SingleOrDefault(U => U.Login == Login && U.Password == Password);

if (Core.CurrentUser != null)

{

if (Core.CurrentUser.Verified!=true) {

MainWindow Window = new MainWindow();

Window.Show();

Close();

}

else

{

EmailMessageSender emailMessageSender = new EmailMessageSender(true);

emailMessageSender.SendMessage(Core.CurrentUser.Email);

PrimaryDialog primaryDialog = new PrimaryDialog(FindResource("Confirmation").ToString(), FindResource("InputEmailCode").ToString()+" "+Core.CurrentUser.Email.ToString(),PrimaryDialog.InputType.Text);

primaryDialog.ShowDialog();

if (primaryDialog.DialogResponse!=null) {

if (primaryDialog.DialogResponse == Core.VereficationCode)

{

MainWindow Window = new MainWindow();

Window.ShowDialog();

Close();

}

else

{

MessageBox.Show(FindResource("InvalidCode").ToString(), FindResource("Error").ToString(),

MessageBoxButton.OK, MessageBoxImage.Error, MessageBoxResult.OK);

}

}

}

}

else

{

MessageBox.Show(FindResource("InvalidLoginOrPassword").ToString(), FindResource("Information").ToString(),

MessageBoxButton.OK, MessageBoxImage.Information, MessageBoxResult.OK);

/\*Attempts--;

if (Attempts == 0)

{

//Close();

Properties.Settings.Default.BlockingStartTime = DateTime.Now;

Properties.Settings.Default.Save();

BlockingTime = StandartBlockingTime;

Attempts = MaxAttempts;

}\*/

}

}

catch

{

MessageBox.Show(FindResource("ConnectDBException").ToString(), FindResource("Error").ToString(),

MessageBoxButton.OK, MessageBoxImage.Error, MessageBoxResult.OK);

}

}

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

{

this.Close();

}

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

{

RegistrationWindow registrationWindow = new RegistrationWindow();

registrationWindow.Show();

this.Close();

}

}

}

Программный код страницы ProfilePage

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Drawing.Imaging;

using Microsoft.Win32;

using System.IO;

namespace QQCourse.Pages

{

/// <summary>

/// Interaction logic for ProfilePage.xaml

/// </summary>

public partial class ProfilePage : Page

{

public ProfilePage()

{

InitializeComponent();

initInfo();

}

private bool SettingsVisible=true;

private ImageParserBase64 ImageParser = new ImageParserBase64();

public void initInfo()

{

if (Core.CurrentUser.Avatar!=null)

{

AvatarImage.Source = ImageParser.ParseByteToImage(Core.CurrentUser.Avatar);

}

else

{

AvatarImage.UpdateDefaultStyle();

}

NicknameLabel.Content = Core.CurrentUser.Login;

LastnameLabel.Content = FindResource("Lastname").ToString()+" "+ Core.CurrentUser.Surname;

NameLabel.Content = FindResource("Name").ToString() + " " + Core.CurrentUser.Name;

PatronimicLabel.Content = FindResource("Middlename").ToString() + " " + Core.CurrentUser.Middlename;

BirthdayLabel.Content = FindResource("Birthday").ToString() + " " + Core.CurrentUser.Birthday;

RoleLabel.Content = FindResource("Role").ToString() + " " + Core.CurrentUser.Role;

GenderLabel.Content = FindResource("Gender").ToString() + " " + (Core.CurrentUser.Gender == true ? FindResource("Male").ToString() : FindResource("Female").ToString());

EmailLabel.Content = FindResource("Email").ToString() + " " + Core.CurrentUser.Email;

if (Core.CurrentUser.Verified == false|| Core.CurrentUser.Verified==null)

{

ConfirmEmailButton.Visibility = Visibility.Visible;

}

else

{

ConfirmEmailButton.Visibility = Visibility.Hidden;

}

}

private void DBSaveException()

{

MessageBox.Show(FindResource("SaveDBException").ToString(), FindResource("Error").ToString(), MessageBoxButton.OK, MessageBoxImage.Error);

Core.CancelChanges(Core.Database.Users);

}

private void NicknameLabel\_TargetUpdated(object sender, DataTransferEventArgs e)

{

}

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

{

PrimaryDialog primaryDialog = new PrimaryDialog(FindResource("ChLogin").ToString(), FindResource("ChLoginMessage").ToString(), PrimaryDialog.InputType.Text);

primaryDialog.Height = 200;

primaryDialog.ShowDialog();

if (primaryDialog.DialogResponse != null)

{

try

{

Core.CurrentUser.Login = primaryDialog.DialogResponse;

Core.Database.SaveChanges();

}

catch

{

DBSaveException();

}

}

initInfo();

}

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

{

PrimaryDialog primaryDialog = new PrimaryDialog(FindResource("ChLastname").ToString(), FindResource("ChLastnameMessage").ToString(), PrimaryDialog.InputType.Text);

primaryDialog.Height = 200;

primaryDialog.ShowDialog();

if (primaryDialog.DialogResponse != null)

{

try

{

Core.CurrentUser.Surname = primaryDialog.DialogResponse;

Core.Database.SaveChanges();

}

catch

{

DBSaveException();

}

}

initInfo();

}

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

{

PrimaryDialog primaryDialog = new PrimaryDialog(FindResource("ChName").ToString(), FindResource("ChNameMessage").ToString(), PrimaryDialog.InputType.Text);

primaryDialog.Height = 200;

primaryDialog.ShowDialog();

if (primaryDialog.DialogResponse != null)

{

try

{

Core.CurrentUser.Name = primaryDialog.DialogResponse;

Core.Database.SaveChanges();

}

catch

{

DBSaveException();

}

}

initInfo();

}

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

{

PrimaryDialog primaryDialog = new PrimaryDialog(FindResource("ChMiddlename").ToString(), FindResource("ChMiddlenameMessage").ToString(), PrimaryDialog.InputType.Text);

primaryDialog.Height = 200;

primaryDialog.ShowDialog();

if (primaryDialog.DialogResponse != null)

{

try

{

Core.CurrentUser.Middlename = primaryDialog.DialogResponse;

Core.Database.SaveChanges();

}

catch

{

DBSaveException();

}

}

initInfo();

}

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

{

PrimaryDialog primaryDialog = new PrimaryDialog(FindResource("ChBirthday").ToString(), FindResource("ChBirthdayMessage").ToString(), PrimaryDialog.InputType.Date,230);

primaryDialog.Height = 230;

primaryDialog.ShowDialog();

if (primaryDialog.DialogResponse != null)

{

try

{

Core.CurrentUser.Birthday = DateTime.Parse(primaryDialog.DialogResponse);

Core.Database.SaveChanges();

}

catch

{

DBSaveException();

}

}

initInfo();

}

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

{

PrimaryDialog primaryDialog = new PrimaryDialog(FindResource("ChGender").ToString(), FindResource("ChGenderMessage").ToString(), PrimaryDialog.InputType.Gender);

primaryDialog.Height = 200;

primaryDialog.ShowDialog();

if (primaryDialog.DialogResponse != null)

{

try

{

Core.CurrentUser.Gender = primaryDialog.DialogResponse== FindResource("Male").ToString() ? true:false;

Core.Database.SaveChanges();

}

catch

{

DBSaveException();

}

}

initInfo();

}

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

{

PrimaryDialog primaryDialog = new PrimaryDialog(FindResource("Confirmation").ToString(), FindResource("ConfirmPasswordMessage").ToString(), PrimaryDialog.InputType.Text);

primaryDialog.Height = 200;

primaryDialog.ShowDialog();

if (primaryDialog.DialogResponse != null)

{

if (primaryDialog.DialogResponse == Core.CurrentUser.Password)

{

PrimaryDialog d = new PrimaryDialog(FindResource("ChPassword").ToString(), FindResource("ChPasswordMessage").ToString(), PrimaryDialog.InputType.Text);

d.Height = 200;

d.ShowDialog();

if (d.DialogResponse != null)

{

try

{

Core.CurrentUser.Password = d.DialogResponse;

Core.Database.SaveChanges();

}

catch

{

DBSaveException();

}

}

}

else

{

MessageBox.Show(FindResource("InvalidPassword").ToString(), FindResource("Error").ToString(), MessageBoxButton.OK, MessageBoxImage.Error);

}

initInfo();

}

}

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

{

PrimaryDialog primaryDialog = new PrimaryDialog(FindResource("ChEmail").ToString(), FindResource("ChEmailMessage").ToString(), PrimaryDialog.InputType.Text);

primaryDialog.Height = 200;

primaryDialog.ShowDialog();

if (primaryDialog.DialogResponse != null)

{

Regex regex = new Regex(@"^([\w\.\-]+)@([\w\-]+)((\.(\w){2,3})+)$");

if (regex.IsMatch(primaryDialog.DialogResponse)) {

if (Core.Database.Users.Where(u=>u.Email==primaryDialog.DialogResponse).Count()<=0) {

try

{

Core.CurrentUser.Email = primaryDialog.DialogResponse;

Core.CurrentUser.Verified = false;

Core.Database.SaveChanges();

}

catch

{

DBSaveException();

}

}

else

{

MessageBox.Show(FindResource("EmailAlreadyBusy").ToString(), FindResource("Error").ToString(), MessageBoxButton.OK, MessageBoxImage.Error);

}

}

else

{

MessageBox.Show(FindResource("InvalidEmailPattern").ToString(), FindResource("Error").ToString(), MessageBoxButton.OK, MessageBoxImage.Error);

}

}

initInfo() ;

}

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

{

EmailMessageSender emailMessageSender = new EmailMessageSender(false);

if (emailMessageSender.SendMessage(Core.CurrentUser.Email))

{

PrimaryDialog primaryDialog = new PrimaryDialog(FindResource("EmailVerefication").ToString(), FindResource("InputEmailCode").ToString()+" "+Core.CurrentUser.Email, PrimaryDialog.InputType.Text);

primaryDialog.Height = 230;

primaryDialog.ShowDialog();

if (primaryDialog.DialogResponse != null)

{

if (primaryDialog.DialogResponse == Core.VereficationCode)

{

try

{

Core.CurrentUser.Verified = true;

Core.Database.SaveChanges();

MessageBox.Show(FindResource("SuccefullyVereficated").ToString(), FindResource("Notification").ToString(), MessageBoxButton.OK, MessageBoxImage.Information);

}

catch

{

DBSaveException();

}

}

else

{

MessageBox.Show(FindResource("InvalidCode").ToString(), FindResource("Error").ToString(), MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

else

{

MessageBox.Show(FindResource("FailedSendCode").ToString(), FindResource("Error").ToString(), MessageBoxButton.OK, MessageBoxImage.Error);

}

initInfo();

}

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

{

Visibility visibility = SettingsVisible?Visibility.Hidden:Visibility.Visible;

SettingsVisible = !SettingsVisible;

if (Core.CurrentUser.Verified == false || Core.CurrentUser.Verified == null)

{

ConfirmEmailButton.Visibility = visibility;

}

EditBirthdayButton.Visibility = visibility;

EditEmailButton.Visibility = visibility;

EditGenderButton.Visibility = visibility;

EditLastnameButton.Visibility = visibility;

EditLoginButton.Visibility = visibility;

EditMiddlenameButton.Visibility = visibility;

EditNameButton.Visibility = visibility;

EditPasswordButton.Visibility = visibility;

EditAvatarButton.Visibility = visibility;

//DeleteAvatarButton.Visibility = visibility;

}

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

{

var dialog = new Microsoft.Win32.OpenFileDialog();

dialog.FileName = "Image";

dialog.Filter = "Image files (\*.BMP, \*.JPG, \*.GIF, \*.TIF, \*.PNG, \*.ICO, \*.EMF, \*.WMF)|\*.bmp;\*.jpg;\*.gif; \*.tif; \*.png; \*.ico; \*.emf; \*.wmf";

bool? result = dialog.ShowDialog();

if (result == true)

{

try

{

string filename = dialog.FileName;

BitmapImage image = new BitmapImage(new Uri(filename));

BitmapSource bitmapSource = image;

Core.CurrentUser.Avatar = ImageParser.ParseImageToByte(bitmapSource);

Core.Database.SaveChanges();

}

catch

{

MessageBox.Show(FindResource("SaveDBExceptionImage").ToString(), FindResource("Error").ToString(), MessageBoxButton.OK, MessageBoxImage.Error);

try

{

Core.CancelChanges(Core.Database.Users);

}

catch { }

}

}

initInfo();

}

}

}

Программный код страницы MyTestsPage

using QQCourse.Windows;

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace QQCourse.Pages

{

/// <summary>

/// Логика взаимодействия для MyTestsPage.xaml

/// </summary>

public partial class MyTestsPage : Page

{

private List<Page> ActivePages;

private int CurrentPageIndex;

public MyTestsPage()

{

InitializeComponent();

UpdateTests();

CurrentPageIndex = -1;

ActivePages = new List<Page>();

}

public void UpdateTests()

{

if (SearchTextBox.Text.Length < 3)

{

var Tests = Core.Database.Tests.Where(T => T.CreatorId == Core.CurrentUser.Id);

TestsListView.ItemsSource = Tests.ToList();

}

else

{

var Tests = Core.Database.Tests.Where(T => T.CreatorId == Core.CurrentUser.Id && T.Name.Contains(SearchTextBox.Text));

TestsListView.ItemsSource = Tests.ToList();

}

}

private void DBSaveException()

{

MessageBox.Show(FindResource("SaveDBException").ToString(), FindResource("Error").ToString(), MessageBoxButton.OK, MessageBoxImage.Error);

Core.CancelChanges(Core.Database.Tests);

}

private void SearchTextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

UpdateTests();

}

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

{

EditTestDialog edt = new EditTestDialog(null);

edt.ShowDialog();

UpdateTests();

}

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

{

if (TestsListView.SelectedIndex >= 0)

{

if (MessageBox.Show(FindResource("DeleteConfirmation").ToString(), FindResource("Confirmation").ToString(), MessageBoxButton.YesNo, MessageBoxImage.Question, MessageBoxResult.No) == MessageBoxResult.Yes)

{

try

{

Data.Tests selected = (TestsListView.SelectedItem as Data.Tests);

List<Data.Questions> questions = Core.Database.Questions.Where(x => x.TestId == selected.Id).ToList();

List<Data.Results> results = Core.Database.Results.Where(x => x.TestId == selected.Id).ToList();

foreach (var a in results)

{

Core.Database.Results.Remove(a);

}

foreach (var q in questions)

{

List<Data.Answers> ans = Core.Database.Answers.Where(x => x.QuestId == q.Id).ToList();

foreach (var a in ans)

{

Core.Database.Answers.Remove(a);

}

Core.Database.Questions.Remove(q);

}

Core.Database.Tests.Remove(selected);

Core.Database.SaveChanges();

}

catch

{

MessageBox.Show(FindResource("DeleteDBException").ToString(), FindResource("Error").ToString(), MessageBoxButton.OK, MessageBoxImage.Error);

Core.CancelChanges(Core.Database.Tests);

Core.CancelChanges(Core.Database.Questions);

Core.CancelChanges(Core.Database.Answers);

Core.CancelChanges(Core.Database.Results);

}

UpdateTests();

}

}

}

public void CloseAllPage()

{

ActivePages.RemoveAll(p => true);

CurrentPageIndex = -1;

QeustionsFrame.Navigate(null);

}

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

{

if (TestsListView.SelectedIndex >= 0)

{

EditTestDialog edt = new EditTestDialog(TestsListView.SelectedItem as Data.Tests);

edt.ShowDialog();

UpdateTests();

}

}

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

{

if (TestsListView.SelectedItem!=null)

{

QeustionsFrame.Navigate(new EditQuestionsPage(this,(TestsListView.SelectedItem as Data.Tests).Id));

}

}

private void QeustionsFrame\_Navigated(object sender, NavigationEventArgs e)

{

}

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

{

if (TestsListView.SelectedIndex >= 0)

{

EditTestDialog edt = new EditTestDialog(TestsListView.SelectedItem as Data.Tests);

edt.ShowDialog();

UpdateTests();

}

}

}

}

Программный код страницы TestsBrowserPage

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace QQCourse.Pages

{

/// <summary>

/// Логика взаимодействия для TestsBrowserPage.xaml

/// </summary>

public partial class TestsBrowserPage : Page

{

public TestsBrowserPage()

{

InitializeComponent();

UpdateTests();

}

private void UpdateTests()

{

if (SearchTextBox.Text.Length < 3)

{

if (PassedCheckBox.IsChecked.Value)

{

var Tests = Core.Database.Tests.Where(t=>Core.Database.Results.FirstOrDefault(r=>r.UserId==Core.CurrentUser.Id).TestId==t.Id);

TestsListView.ItemsSource = Tests.ToList();

}

else

{

var Tests = Core.Database.Tests;

TestsListView.ItemsSource = Tests.ToList();

}

}

else

{

if (PassedCheckBox.IsChecked.Value)

{

var Tests = Core.Database.Tests.Where(t => Core.Database.Results.FirstOrDefault(r => r.UserId == Core.CurrentUser.Id).TestId == t.Id&&t.Name.Contains(SearchTextBox.Text));

TestsListView.ItemsSource = Tests.ToList();

}

else

{

var Tests = Core.Database.Tests.Where(T => T.Name.Contains(SearchTextBox.Text));

TestsListView.ItemsSource = Tests.ToList();

}

}

}

public void CloseAllPage()

{

TestFrame.Navigate(null);

}

private void SearchTextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

UpdateTests();

}

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

{

if (TestsListView.SelectedIndex >= 0)

{

TestPage tp = new TestPage(this,TestsListView.SelectedItem as Data.Tests);

TestFrame.Navigate(tp);

}

}

private void TestFrame\_Navigated(object sender, NavigationEventArgs e)

{

}

private void PassedCheckBox\_Checked(object sender, RoutedEventArgs e)

{

UpdateTests();

}

}

}

Программный код страницы ResultsPage

**using System;**

**using System.Collections.Generic;**

**using System.Drawing;**

**using System.Linq;**

**using System.Text;**

**using System.Threading.Tasks;**

**using System.Windows;**

**using System.Windows.Controls;**

**using System.Windows.Data;**

**using System.Windows.Documents;**

**using System.Windows.Input;**

**using System.Windows.Media;**

**using System.Windows.Media.Imaging;**

**using System.Windows.Navigation;**

**using System.Windows.Shapes;**

**using QQCourse.Data;**

**namespace QQCourse.Pages**

**{**

**/// <summary>**

**/// Interaction logic for ResultsPage.xaml**

**/// </summary>**

**public partial class ResultsPage : Page**

**{**

**private Results res;**

**TimeSpan timeToPass;**

**TimeSpan time;**

**public ResultsPage(List<QuestResult> results, TimeSpan time,TimeSpan timeToPass, Tests test)**

**{**

**InitializeComponent();**

**this.timeToPass = timeToPass;**

**this.time = time;**

**var (score, maxScore) = CalculateScore(results);**

**ResultsListBox.ItemsSource = results;**

**TimeLabel.Content = time.ToString("hh':'mm':'ss");**

**TimeToPassLabel.Content = timeToPass.ToString("hh':'mm':'ss");**

**ScoreLabel.Content = score;**

**if (maxScore != 0)**

**{**

**if (score \* 1.0 / maxScore \* 100.0 >= maxScore / 100.0 \* test.MinScore)**

**{**

**ResLabel.Content = "Проходной балл набран!";**

**}**

**else**

**{**

**ResLabel.Content = "Проходной балл не набран!";**

**ResLabel.Foreground = new System.Windows.Media.SolidColorBrush(System.Windows.Media.Color.FromRgb(255, 141, 123));**

**}**

**}**

**MinScoreLabel.Content = maxScore / 100.0 \* test.MinScore;**

**Results results1 = new Results();**

**results1.Id = Core.VOID;**

**results1.Score = score;**

**results1.TestId = test.Id;**

**results1.Time = time;**

**results1.UserId = Core.CurrentUser.Id;**

**res = results1;**

**SaveInfo();**

**}**

**private (double,int) CalculateScore(List<QuestResult> results)**

**{**

**int correct = 0;**

**int count = 0;**

**foreach(var r in results)**

**{**

**if (r.IsCorrect) correct++;**

**count++;**

**}**

**double score = correct;**

**if (timeToPass < time)**

**{**

**score = correct - correct / 100.0 \* 20.0;**

**}**

**return (score, count);**

**}**

**private bool SaveInfo()**

**{**

**try**

**{**

**if (res.Id == Core.VOID)**

**{**

**Core.Database.Results.Add(res);**

**}**

**Core.Database.SaveChanges();**

**}**

**catch**

**{**

**DBSaveException();**

**return false;**

**}**

**return true;**

**}**

**private void DBSaveException()**

**{**

**MessageBox.Show(FindResource("SaveDBException").ToString(), FindResource("Error").ToString(), MessageBoxButton.OK, MessageBoxImage.Error);**

**Core.CancelChanges(Core.Database.Results);**

**}**

**private void CloseButton\_Click(object sender, RoutedEventArgs e)**

**{**

**Core.AppMainWindow.CloseAllPage();**

**}**

**private void ResultsListBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)**

**{**

**}**

**}**

**}**

Программный код класса EmailMessageSender

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Mail;

using System.Text;

using System.Threading.Tasks;

namespace QQCourse

{

public class EmailMessageSender

{

private string smtpServer = "smtp.mail.ru";

private int smtpPort = 587;

private string smtpUsername = "testpassingsystem@mail.ru";

private string smtpPassword = "ER1zGG1jB7ytRd4BL9Ff";

private bool IsAuthentification;

public EmailMessageSender(bool IsAuthentification) {

this.IsAuthentification = IsAuthentification;

}

public bool SendMessage(string recipient)

{

using(SmtpClient smtpClient = new SmtpClient(smtpServer, smtpPort))

{

smtpClient.Credentials=new NetworkCredential(smtpUsername, smtpPassword);

smtpClient.EnableSsl = true;

using (MailMessage mailMessage = new MailMessage())

{

mailMessage.From=new MailAddress(smtpUsername);

mailMessage.To.Add(recipient);

Core.VereficationCode = GenerateCode();

if (!IsAuthentification) {

mailMessage.Subject = "Подтверждение адреса электронной почты в системе тестирования знаний.";

mailMessage.Body = $"Подтверждение адреса электронной почты в системе тестирования знаний.\n\r\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\rКод подтверждения: {Core.VereficationCode}\n\r\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\rЭто письмо создано автоматически, отвечать на него не нужно.\n\rЕсли вы считаете что это сообщение пришло вам по ошибке, то просто удалите его.";

}

else

{

mailMessage.Subject = "Подтверждение входа в систему тестирования знаний.";

mailMessage.Body = $"Подтверждение входа в систему тестирования знаний.\n\r\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\rКод подтверждения: {Core.VereficationCode}\n\r\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\rЭто письмо создано автоматически, отвечать на него не нужно.\n\rЕсли это не вы пытаетесь войти в аккаунт, то немедленно поменяйте данные своей учётной записи.";

}

try

{

smtpClient.Send(mailMessage);

//smtpClient.SendMailAsync(mailMessage);

return true;

}catch (Exception ex)

{

Console.WriteLine(ex.ToString());

return false;

}

}

}

}

private string GenerateCode()

{

string Digits = "1234567890";

string Alphabet = "QWERTYUIOPASDFGHJKLZXCVBNM";

string code = "";

Random generator = new Random();

code += Alphabet[generator.Next(0, Alphabet.Length)]+"-";

for(int i = 0; i<6; i++)

{

code += Digits[generator.Next(0, Digits.Length)];

}

return code;

}

}

}