**СОДЕРЖАНИЕ**

[Введение 3](#_Toc137585879)

[1 Исследовательский раздел 4](#_Toc137585880)

[2 Конструкторский раздел 8](#_Toc137585881)

[2.1 Проектирование информационной модели данных 8](#_Toc137585882)

[2.2 Проектирование серверной части приложения 9](#_Toc137585883)

[2.2.1Разработка схемы базы данных 9](#_Toc137585884)

[2.2.2Разработка сущностей базы данных 13](#_Toc137585885)

[2.3Проектирование клиентской части приложения 15](#_Toc137585886)

[2.3.1Разработка модулей схемы 15](#_Toc137585887)

[2.3.2 Разработка пользовательского интерфейса 17](#_Toc137585888)

[2.3.3 Организация доступа к объектам базы данных 21](#_Toc137585889)

[2.3.4 Разработка блох-схем алгоритмов процедур и функций 22](#_Toc137585890)

[2.4 Обеспечение коллективного доступа. Защита информации 24](#_Toc137585891)

[3 Технологическая часть 26](#_Toc137585892)

[3.1 Тестирование и отладка приложения 26](#_Toc137585893)

[3.2 Инструкция администратора базы данных 28](#_Toc137585894)

[3.3 Инструкция по эксплуатации приложения 38](#_Toc137585895)

[4 Технико-экономический раздел 42](#_Toc137585896)

[5 Раздел охраны труда 54](#_Toc137585897)

[Заключение 57](#_Toc137585898)

[Список используемых источников 58](#_Toc137585899)

[Приложение А 59](#_Toc137585900)

Введение

В современном мире разработка клиент-серверных приложений становится все более актуальной и востребованной задачей. В рамках данного дипломного проекта было решено разработать такое приложение для ведения региональных проектов субъектом РФ.

Приложение по ведению региональных проектов субъектом РФ может быть полезным для государственных органов и управления, ответственных за реализацию проектов на территории региона. Оно может использоваться для мониторинга и управления региональными проектами, включая отслеживание прогресса выполнения задач и контроль за бюджетом.

Для реализации данного проекта были использованы современные технологии, такие как WPF и C#, а также базы данных. Это позволило создать удобный и функциональный инструмент для управления проектами, что может быть полезно как для государственных органов, так и для частных компаний. В данной работе будет представлено описание проекта, его архитектуры, особенностей реализации и результатов тестирования.

1 Исследовательский раздел

Процесс разработки программного обеспечения – набор правил, согласно которым построена разработка программного обеспечения. Приложение можно назвать клиент серверным если оно включает в себя клиент-серверную архитектуру. Разработку клиент-серверного приложения необходимо начинать с выбора архитектуры клиент-сервера.

Для разработки клиент-серверных систем имеется два подхода. Первый подход построение систем на основе двухзвенной архитектуры. Состоит из клиентской и серверной части. Как правило, серверная часть представляет собой сервер БД, на котором расположены общие данные. А клиентская часть представляет приложение, которое связывается с сервером БД, осуществляет к нему запросы и получает ответы. Такие системы используются в локальных сетях, т.к. нет затруднений с установкой клиентской части. Также системы с такой архитектурой более безопасны, т.к. могут использовать собственные протоколы передачи данных, не известные злоумышленникам. Поэтому многие крупные компании, которые располагаются не в едином месте и для соединения подразделений используют глобальную сеть Интернет, выбирают именно такую архитектуру построения клиент-серверных систем.

При разработке информационных систем, рассчитанных на широкую аудиторию, возникают проблемы с использованием двухзвенной архитектуры. Во-первых, пользователю необходимо иметь в наличии клиентскую часть, а, во-вторых, у неопытного пользователя, могут возникнуть проблемы с конфигурированием такой системы. Поэтому в последнее время, более часто разрабатывают приложения на базе трехзвенной архитектуры.

Второй подход построение систем на основе трехзвенной архитектуры. Серверная часть в этой архитектуре представляет собой сервер приложений и сервер БД. А в качестве клиента выступает web-браузер. Такая система очень проста для пользователя. Ему необходимо знать только адрес сервера приложения и наличие web-браузера на рабочем компьютере. Все данные представляются в виде html-разметки, с использование графики (jpeg, gif, flash) и JavaScript. Передача запросов от клиента к серверу приложений происходит по средствам CGI-интерфейса. Сервер приложений общается с сервером БД, используя другой интерфейс, зависящий от того, на основе каких средств строится конкретная информационная система. Недостатками такой архитектуры является использование общеизвестных протоколов и интерфейсов передачи данных. Злоумышленник может осуществить взлом системы, если она будет недостаточно хорошо проверять поступившие запросы от клиента.

При разработке клиент-серверных приложений необходимо учитывать:

* на каких пользователей будет рассчитана данная информационная система;
* какие требования предъявляются к безопасности.

Если информационная система должна быть общедоступной и рассчитана на широкую аудиторию, то необходимо использовать трехзвенную архитектуру.

Если информационная система используется внутри предприятия, доступ имеют к ней ограниченные пользователи и требуется создать максимально безопасную и защищенную систему, то следует отдать предпочтение двухзвенной архитектуре [1].

В рамках курсового проекта был выбран первый способ для разработки клиент-серверной системы на основе двухзвенной архитектуры.

Для реализации двухзвенной архитектурыбыла выбрана платформа WPF (WindowsPresentetionFoundation)

При выборе WPF можно выделить такие преимущества как аппаратное ускорение через DirectX, что сильно влияет на производительность. Также можно отметить веб-подобную модель компоновки. Вместо того чтобы фиксировать элементы управления на месте с определенными координатами, WPF поддерживает гибкий поток, размещающий элементы управления на основе их содержимого. В результате получается пользовательский интерфейс, который может быть адаптирован для отображения высоко динамичного содержимого или к разным языкам.

Преимуществами WPF являются:

* **веб-подобнаямоделькомпоновки;**
* **богатаямодельрисования;**
* **развитаятекстоваямодель;**
* **анимация;**
* **поддержкааудио и видео;**
* **стили и шаблоны;**
* **команды;**
* **декларативныйпользовательскийинтерфейс;**
* **приложениянаосновестраниц.**

При выборе сред разработки были рассмотреныVisualStudio и ProjectRider.VisualStudio – это удобная интегрированная среда разработки (IDE) от Microsoft, позволяющая быстро и эффективно создавать, и разрабатывать проект, выбрав для этого все необходимое. Средаиспользует платформы разработки программного обеспечения Microsoft:WindowsAPI, WindowsForms, WindowsPresentationFoundation, WindowsStoreиMicrosoftSilverlight. Так же она принимает плагины, которые расширяют функциональные возможности практически на каждом уровне, включая добавление поддержки систем управления исходным кодом (таких как Subversion) и обеспечивает стандартный для Windows вид окон приложения. Единственным минусом можно считать сложность освоения данной среды разработки из-за её большого количества различных функций, спрятанных в подразделах меню. Информация из работы [2].

ProjectRider – это среда от JetBrains для работы с платформой .NET. Она обладает поддержкой полного цикла. Фирменная черта продуктов JetBrains, воплощенная и в ProjectRider. С ProjectRiderпоявиться возможность организовать весь цикл создания программного обеспечения: от идеи до поддержки. Функциональность ProjectRider позволяет подключить MSBuild и XBuild, работать с CLI-проектами и организовать отладку приложений .NET and Mono. Множество опций для быстрого создания кода улучшает производительность. Кроссплатформенность ProjectRider работает с Windows, Linux и MacOS. Из минусов можно выделить её молодость. Часть функциональности еще в разработке, не все стартовые ошибкиисправлены. Так же можно отметить её стоимость. Самая дешевая версия ProjectRider обойдется в 139 долларов за первый год использования. Но есть триал-версия и специальные предложения для студентов и непрофильных организаций. Информация из работы [3].

Из этих двух сред разработки был выбран VisualStudio, так как он обладает всем необходимым функционалом для реализации проекта, а также она является бесплатной и дольше находится на рынке труда.

Приложение предоставляет возможность пользователю выбрать интересующий его региональный проект, ознакомиться с подробной информацией о нём, его целями и задачами. Данные будут загружаться по желанию пользователя, с API в базу данных, по нажатию определённой кнопки.

Основными процессами выступают загрузка данных, с API в базу данных, и отображение их на интерфейсе.

2 Конструкторский раздел

2.1 Проектирование информационной модели данных

Черная сфера представляет собой систему, внутреннее устройство которой не важно. В эту систему подаются входные данные, а на выходе из системы поступают выходные данные. Черная сфера представлена на рисунке 2.1.



Рисунок 2.1 – Модель «Черная сфера»

Представим наше приложение в виде черной сферы. В приложении будут присутствовать такие выходные данные, как пользователь. На выходе из приложения будут поступать региональные проекты. Пользователем будет выступать человек, совершающий какие-либо действия с приложением. Региональными проектами будут результаты, полученные после манипуляций пользователя. Черная сфера с перечисленными входными и выходными параметрами представлена на рисунке 2.2.

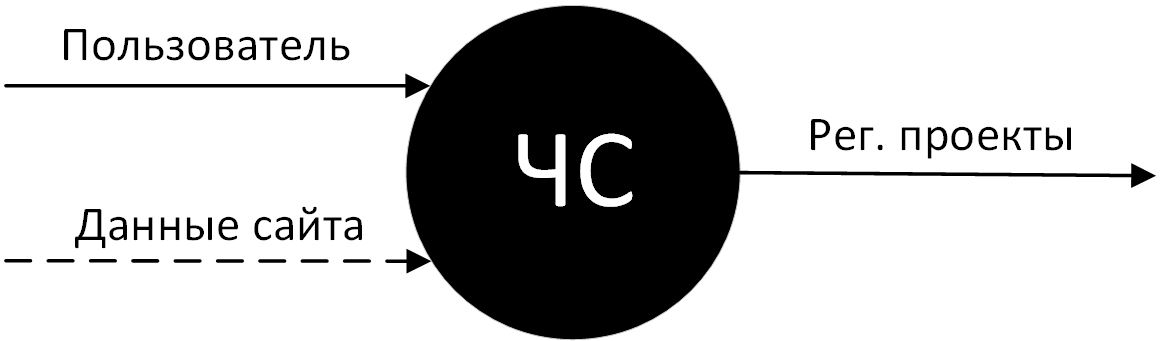


Рисунок 2.2 – Черная сфера с параметрами

В приложении существует два основных процесса отвечающие за выставление расписания для загрузки данных и рабтуAPI с базой данных. Данные об региональных проектах поступают из базы и форматируются на интерфейс.

2.2 Проектирование серверной части приложения

2.2.1Разработка схемы базы данных

Для выявления всех возможных сущностей будущей базы и получения концептуальной модели данных будет проведено несколько серийнормализации.

На первом этапе нормализации можно представить модель как связь между пользователем и региональными проектами. Первый этап нормализациипредставлен на рисунке 2.3.

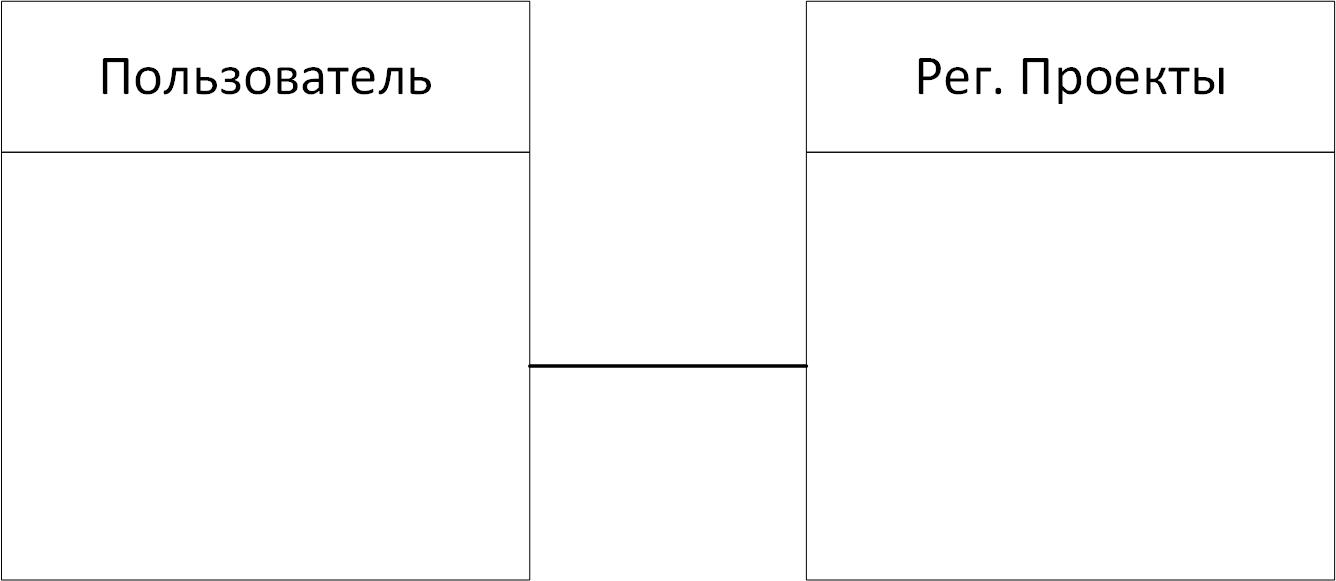


Рисунок 2.3– Первый этап нормализации

Во втором этапе нормализации разобьём сущность теста на 3 сущности: Информация об региональных проектах, Цели региональных проектов и Задачи региональных проектов. Второй этап нормализации на рисунке 2.4.

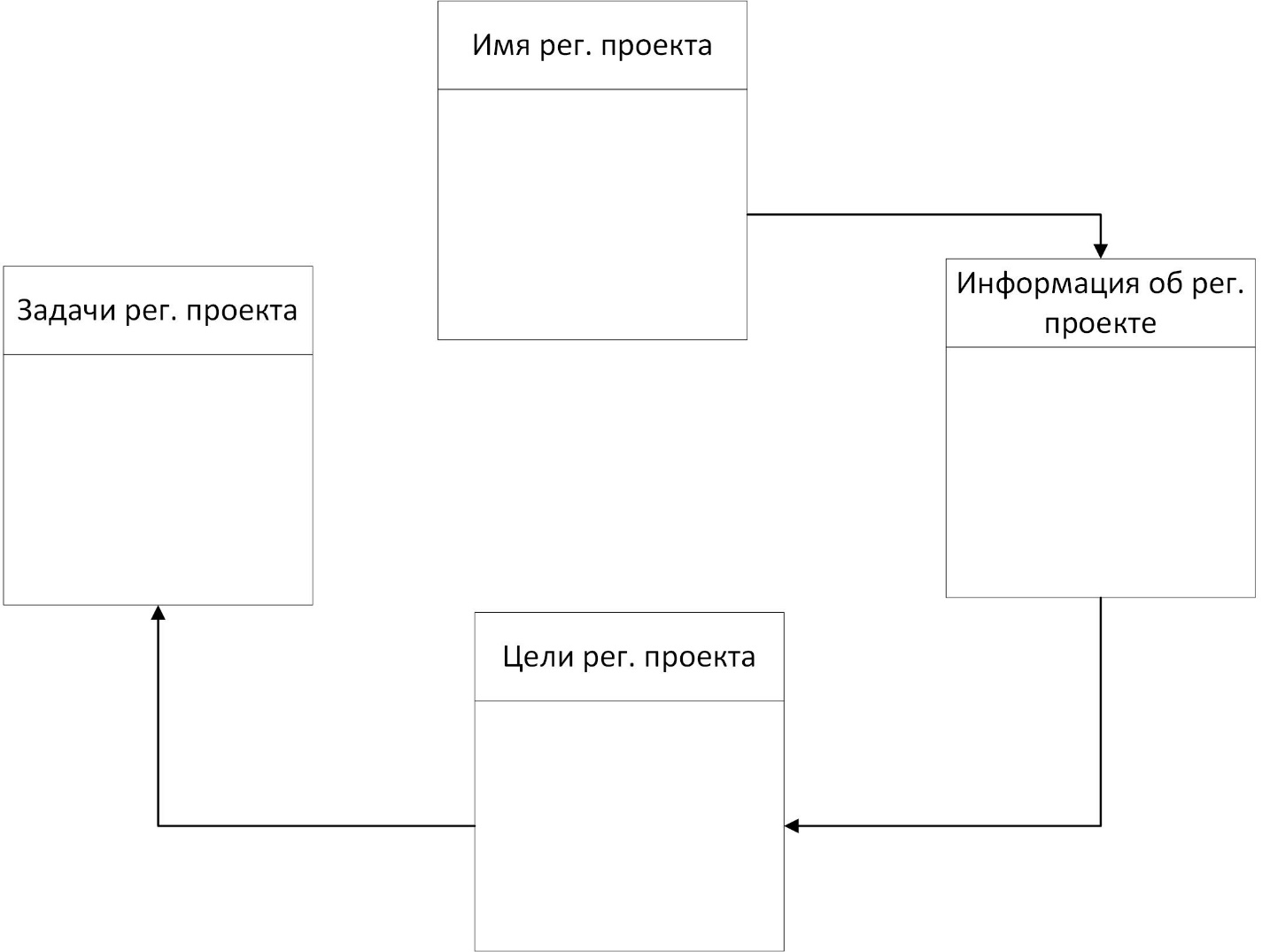


Рисунок 2.4 – Второй этап нормализации

В третьем заключительном этапе нормализации получим полную концептуальную схему добавив Задачи региональных проектов и Результаты региональных проектов. Третий этап нормализации представлена на рисунке 2.5.

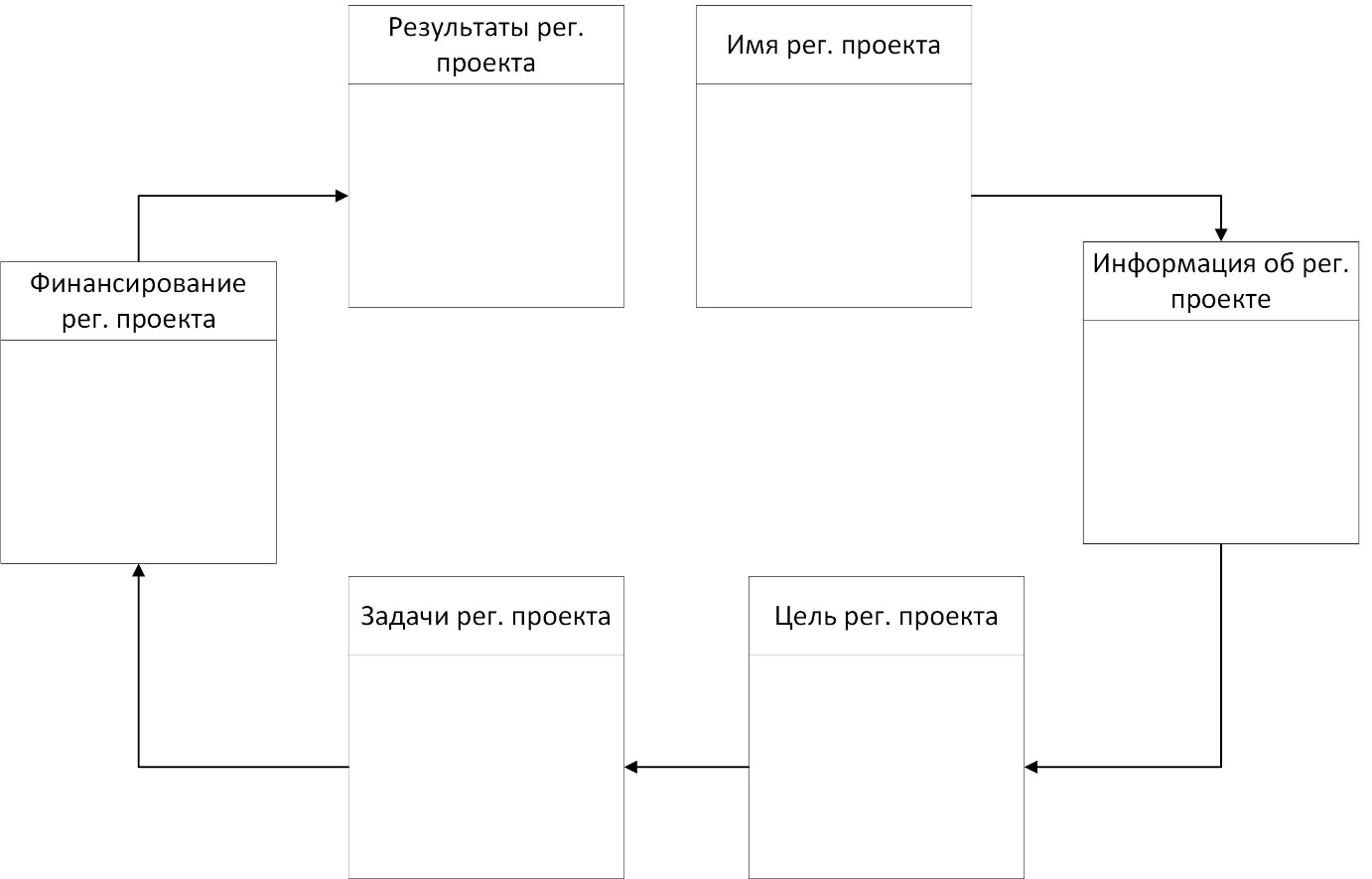


Рисунок 2.5– Третий этап нормализации

Получим логическую модель данных с содержанием всех сущностей, связей и атрибутов данных. Логическая модель данных представлена на рисунке 2.6.

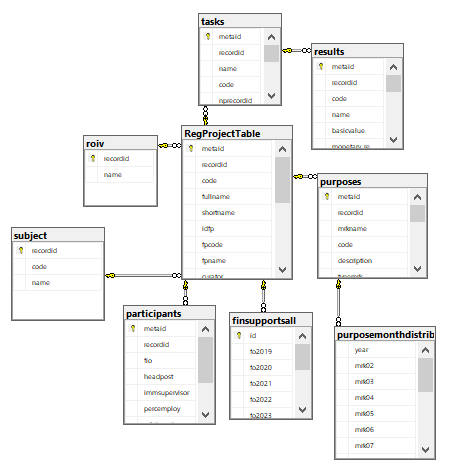


Рисунок 2.6– Логическая модель данных

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 2.7.

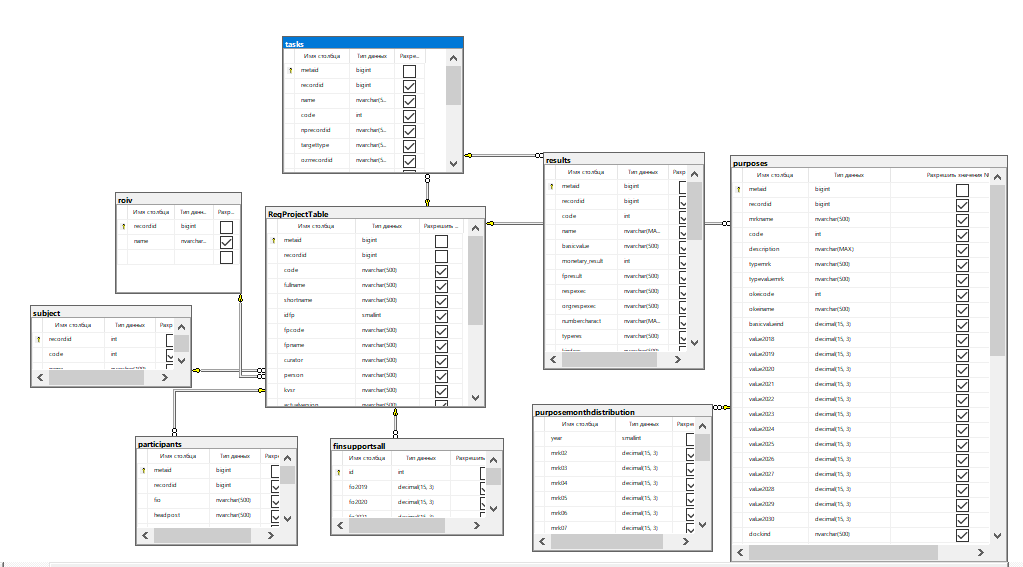


Рисунок 2.7– Физическая модель данных

2.2.2Разработка сущностей базы данных

В результатах раздела «Разработка схемы базы данных» получена схема базы данных, из которой следует необходимость присутствия определенных сущностей необходимых для полноценной работы приложения. Для удобства все сущности сведены в табличном виде. Сущности схемы базы данных представлены в таблице 2.1.

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 |
| RegProject | Содержит информацию об региональных проектах РФ | int, nvarchar(100), smallint | roiv, subject | tasks, purposes, results, finsupportsall, purposemonthdistribution |
| roiv | Содержит информацию об министерствах РФ | int, nvarchar(100) | - | RegProject |
| subject | Содержит информацию об субъектах РФ | binary(100), nvarchar(100) | - | RegProject |
| tasks | Содержит информацию об задачах региональных проектов РФ | int, nvarchar(100), binary(100), | result | RegProject |
| purposes | Содержит информацию об целях региональных проектов РФ | int, nvarchar(100), binary(100), decimal(15, 3), smallint | purposemonthdistribution | RegProject |
| results | Содержит информацию об результатах региональных проектах РФ | int, nvarchar(100), binary(100), smallint | - | tasks |
| purposemonthdistribution | Содержит информацию о годах | smallint, decimal(15, 3), nvarchar(100), binary(100), int | - | purposes |

2.3Проектирование клиентской части приложения

2.3.1Разработка модулей схемы

WPF предоставляет комплексный набор функций разработки приложений, которые включают в себя язык XAML, элементы управления, привязку к данным, макет, двумерную и трехмерную графику, анимацию, стили, шаблоны, документы, мультимедиа, текст и типографические функции. WPF является частью .NET, поэтому вы можете создавать приложения, включающие другие элементы .NET API. [5].

JSON API Данные – это формат обмена данными, который используется в API для передачи информации между клиентом и сервером. Он основан на языке JSON (JavaScript Object Notation) и предоставляет стандартизированный формат для описания ресурсов, их атрибутов и связей. JSON API Данные определяет также набор правил для запросов и ответов, что облегчает работу с API и упрощает интеграцию с другими системами.

Представим клиентскую часть приложения в виде модульной схемы показывающая связь между окнами, классами и страницами при организации клиентской части приложения. Модульная схема клиентской части приложения представлена на рисунке 2.8.

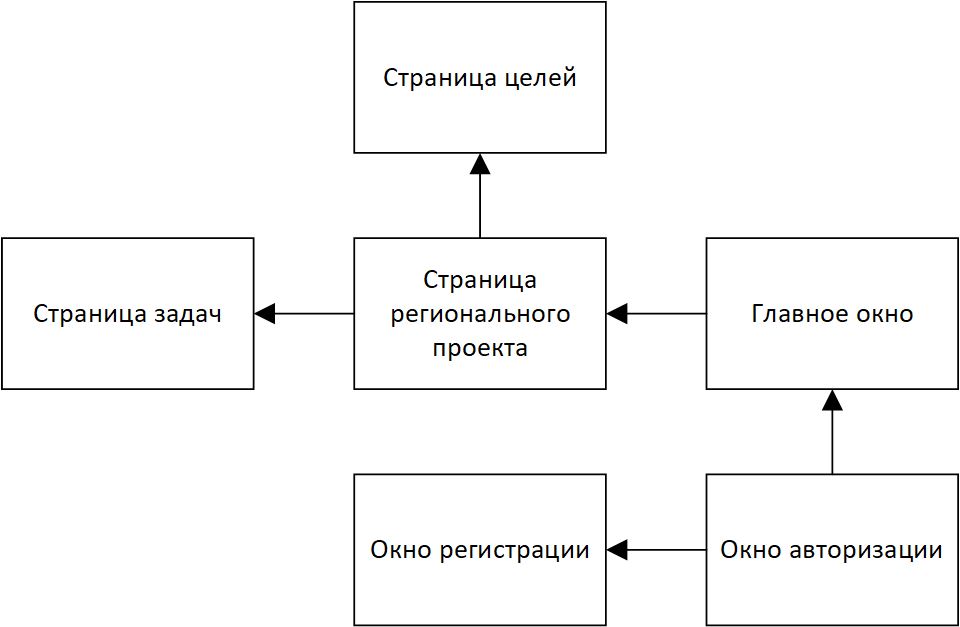


Рисунок 2.8– Модульная схема клиентской части приложения

В составе модульной схемы присутствует окно авторизации пользователя, главное окно, для просмотра и выбора региональных проектов и страница расписания, для регулировки закачеваемых данных.

Часть получаемых данных представлена в виде API в формате Json, приходящего с открытого источника после запроса пользователя. Результат ответа в Json формате представлена на рисунке 2.9

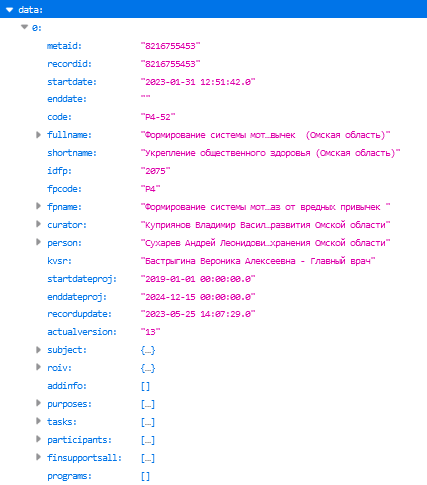


Рисунок 2.9 – Результат ответа в Json формате

В составе ответа присутствуют данные по всем региональным проектам, их цели и задачи.

2.3.2 Разработка пользовательского интерфейса

Пользовательский интерфейс – это совокупность информационной модели проблемной области, средств и способов взаимодействия пользователя с информационной моделью, а также компонентов, обеспечивающих формирование информационной модели в процессе работы программной системы.

**Графический пользовательский интерфейс – самый** популярный тип пользовательского интерфейса. Он представляет собой окно с различными элементами управления. Пользователи взаимодействуют с элементами с помощью клавиатуры, компьютерной мыши и голосовых команд.

**XAML**представляет собой язык разметки, используемый для создания экземпляров объектов .NET. Хотя язык XAML — это технология, которая может быть применима ко многим различным предметным областям, его главное назначение — конструирование пользовательских интерфейсов WPF. Другими словами, документы XAML определяют расположение панелей, кнопок и прочих элементов управления, составляющих окна в приложении WPF.

На этом этапе рассмотрим каждый блок модульной схемы поподробнее. При запуске приложения открывается окно авторизации. Данное окно содержит в себе название приложения, два текстовых поля для ввода логина и пароля, а также кнопка для входа в программу, кнопка регистрации пользователя и кнопка выхода из программы. Разметка страницы авторизации представлен на рисунке 2.10.



Рисунок 2.10 – Разметка страницы авторизации

Страница регистрации содержит название приложения. Имеются два текстовых поля для ввода логина, пароля, а также кнопка для создания аккаунта и ссылка для возвращения. Разметка страницы регистрации представлена на рисунке 2.11.



Рисунок 2.11 – Разметка страницы регистрации

При авторизации под клиентской учетной записью происходит переход на главную страницу, которая пользователя плитками с информацией по региональным проектам, кнопка закачки данных и кнопки для перехода на страницу вперёд и назад. При нажати на имя регионального проекта произойдёт переход на окно целей и задач данного проекта. Разметка главной страницы представлена на рисунке 2.12.



Рисунок 2.12 – Разметка главной страницы

При нажатии на кнопку целей произойдёт переход на одноименную страницу с более подробной информацией. Разметка страницы целей представлена на рисунке 2.13.



Рисунок 2.13 – Разметка страницы целей и задач

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]

Возьмем для примера страницу с вывдом региональных проектов на главной странице, где участвует ItemsControl. ItemsControl страницы с региональными проектами представлен на рисунке 2.14.



Рисунок 2.14 – ItemsControl страницы региональных проектов

В Binding мы записываем свойство объекта источника. Привязка объекта с данными к ItemsControl представлена на рисунке 2.15.

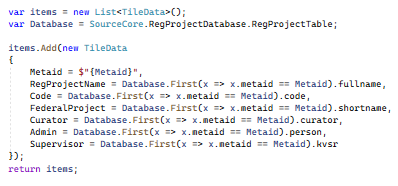


Рисунок 2.15 – Привязка объекта с данными к ItemsControl

Беря данные из базы данных, мы связываем их с ItemsControl. Сам ItemsControl верстается под конкретное содержимое.

2.3.4 Разработка блох-схем алгоритмов процедур и функций

Основными процессом приложения является получение и загрузка данных.

Метод UrlParse отвечает за получение и загрузку данных. Блок-схема метода UrlParse представлена на рисунке 2.16.

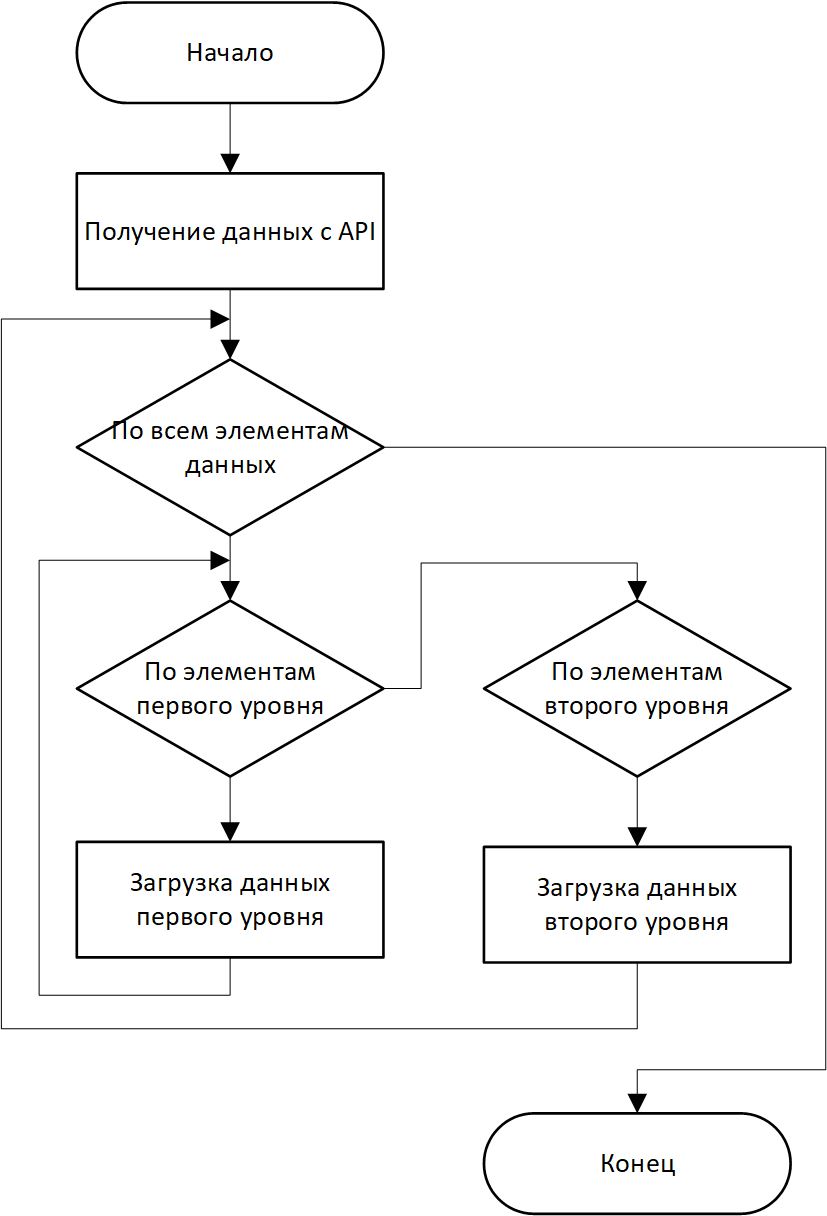


Рисунок 2.16 – Блок-схема метода UrlParse

Метод RegUsers отвечает за получение и загрузку данных. Блок-схема метода RegUsers представлена на рисунке 2.17.

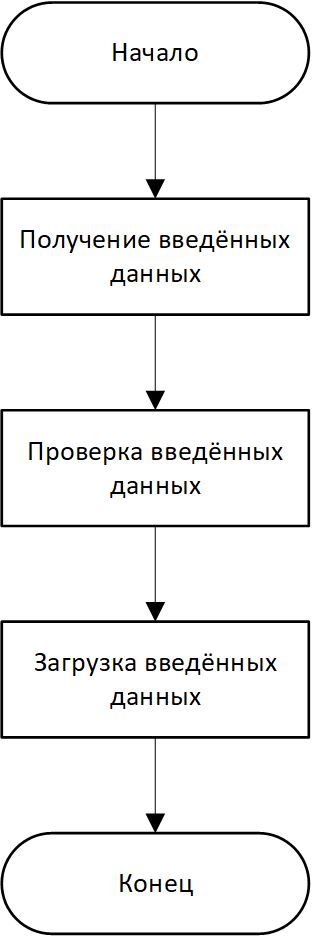


Рисунок 2.17 – Блок-схема метода RegUsers

2.4 Обеспечение коллективного доступа. Защита информации

Основная идея ролевой модели контроля за доступом (Role-BasedAccessControl — 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 доступны две роли: роль обычного пользователя и администратора. Роль пользователя имеет возможность просмотра всей доступной информации об региональных проектах и добавления собственных проектов. Роль администратора имеет возможность обновлять данные и менять уже существующие.

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

Для регистрации пользователю требуется ввести его логин и пароль. Все поля обладают своей валидацией и в случае некорректного ввода данных выводит сообщение с ошибкой. При желании пользователь может вернуться на страницу авторизации.

3 Технологическая часть

3.1 Тестирование и отладка приложения

Отладка — этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки, информация из работы [8]. В связи с тем, что почти невозможно составить реальную программу без ошибок, и почти невозможно для достаточно сложной программы быстро найти и устранить все имеющиеся в ней ошибки. Разумно уже при разработке программы на этапах алгоритмизации и программирования готовиться к обнаружению ошибок на стадии отладки принимать профилактические меры по их предупреждению, информация из работы [9].

Тестирование будет происходить через тест кейсы. Тест кейс –это артефакт, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части. При передаче тестировщику тест-кейсов, он должен пройтись по всем его пунктам и выполнить описанные действия, которые должны привести к определенным результатам. Информация из работы [10]. Тест кейс для функций представлен в таблице 3.1.

Таблица 3.1 – Тест-кейс для методов

|  |  |  |
| --- | --- | --- |
| Имя метода | Управляющее воздействие | Результат воздействия |
| AuthButton\_Click | Вызывается при нажатии кнопки «Войти» в окне авторизации | Отображение диалога ошибки если данные были введены некорректно |
| RegButton\_Click | Вызывается при нажатии кнопки «Регистрировать» в окне регистрации | Отображение диалога ошибки если данные были введены некорректно |
| PrevButton\_Click | Вызывается при нажатии кнопки «Назад» на главной странице | Листает на страницу назад |

Окончание таблицы 3.1

|  |  |  |
| --- | --- | --- |
| Имя метода | Управляющее воздействие | Результат воздействия |
| NextButton\_Click | Вызывается при нажатии кнопки «Вперёдт» на главной странице | Листает на страницу вперёд |
| TextBlock\_PreviewMouseDown | Вызывается при нажатии на название регионального проекта на главной странице | Переносит на страницу целей и задачей регионального проекта |
| UploadNow\_Click | Вызывается при нажатии кнопки «Закачать данные» на главной странице | Закачивает данные региональных проектов |

3.2 Инструкция администратора базы данных

Перед началом работы с приложением необходимо установить и настроить SQLServer 2019. MSSQLServer - это лидирующая РСУБД (Реляционная система управления базами данных) а также главный конкурент OracleDatabase в корпоративном сегменте. В СНГ MSSQL чаще всего применяется для собственных разработок прикладного ПО и для 1С.

Для установки надо перейти на официальный сайт Microsoft и скачать бесплатную версию SQL Server 2019 для тестирования и разработки (Developer). Далее запустить установщик и выбрать тип установки «Пользовательский». Как на рисунке 3.1

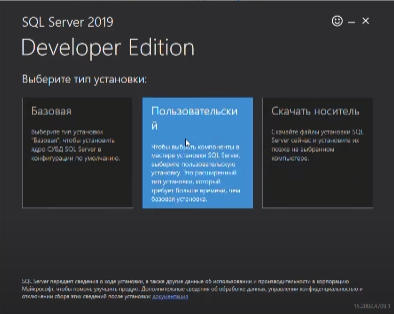


Рисунок 3.1 – «Пользовательский» тип установки

После выбора типа установки следует перейти на 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.

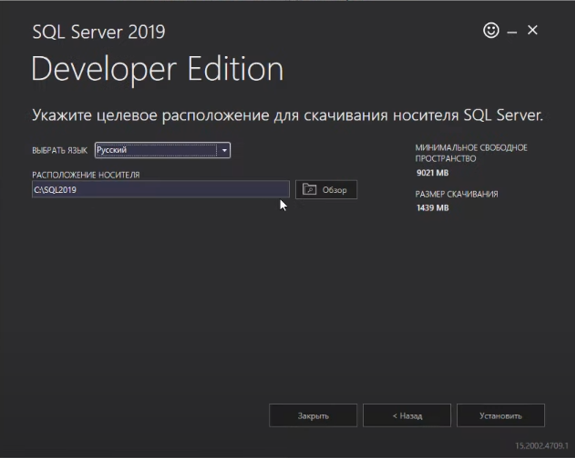


Рисунок 3.2 – Изображение окна

После установки откроется центр установки SQLserver, где стоит перейти в раздел установки «Новая установке изолированного экземпляра SQLServer или добавление компонентов к существующей установке», как показано на рисунке 3.3.

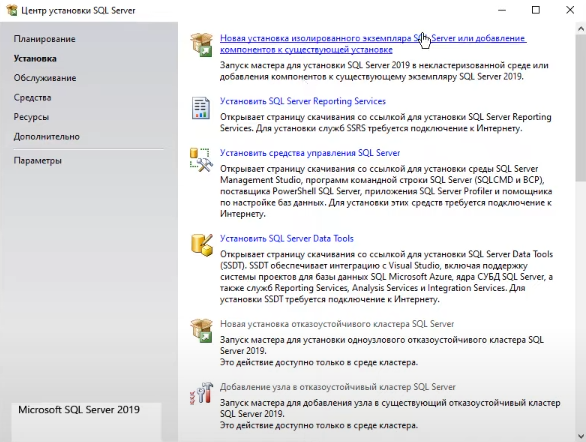


Рисунок 3.3 – Новая установке изолированного экземпляра SQLServer

После установки произойдет обновление продукта. Рисунок 3.4.

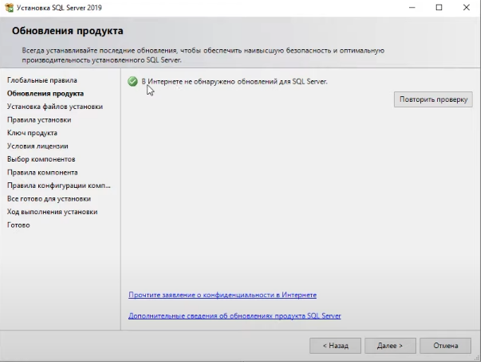


Рисунок 3.4 – Обновление продукта

После обновления можно пропустить все пункты до ключа продукта и выбирать версию «Developer» как на рисунке 3.5.

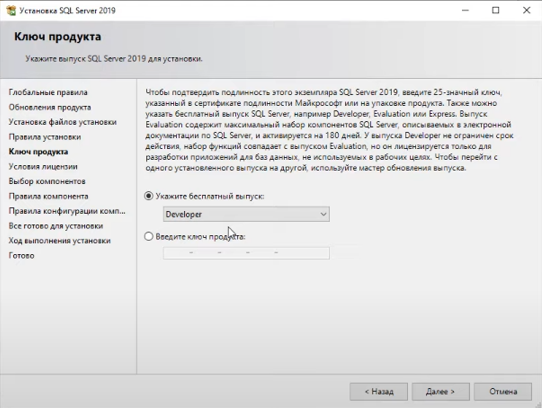


Рисунок 3.5 – Версия «Developer»

В пункте «Условия лицензии» нужно приянть условия и перейти в раздел «Выбор компонентов», где следует установить базовый набор компонентов: «Служба ядра СУБД» и «Полнотекстовой и семантический поиск». Как представлено на рисунке 3.6.

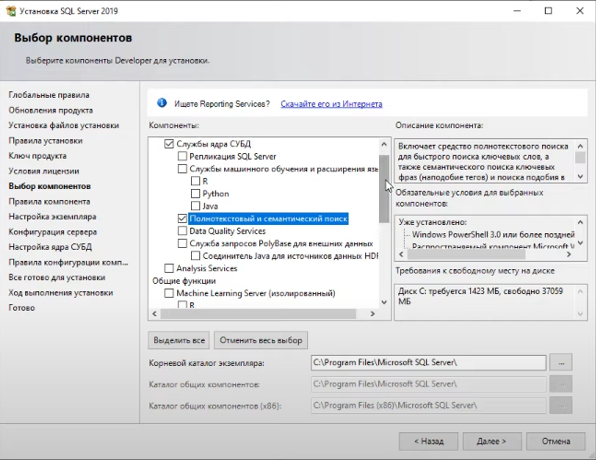


Рисунок 3.6 – Выбор компонентов

В разделе правила компонента можно всё оставить по умолчанию и перейти в раздел «Конфигурация сервера» где нужно настроить работу служб SQLServer. Далее надо задать тип запуска службы. Надо поставить ее на автозапуск, вручную, или отключить. Так же можно зайти в меню "Параметры сортировки" — это настройки таблицы кодировок. Ещё позволенно выполнить сортировку, как учитывать верхний и нижний регистр, как реагировать на символы, и т.п. Настройки можно оставить по умолчанию как на рисунке 3.7.

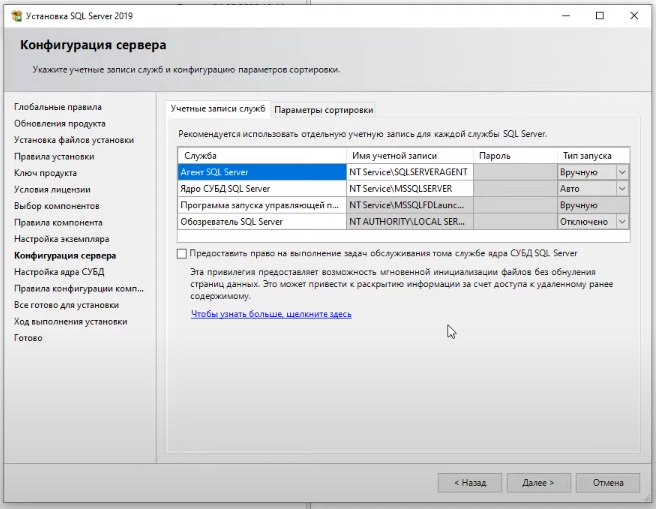


Рисунок 3.7 – Конфигурация сервера

В разделе «Настройка ядра СУБД» надо выбрать режим входа под учетными записями Windows, либо смешанный режим, т.е. возможность входа под учетной записью Windows и под учетной записью SQLServer. Можно оставить режим аутентификации Windows и выбираем пользователякак показано на рисунке 3.8.

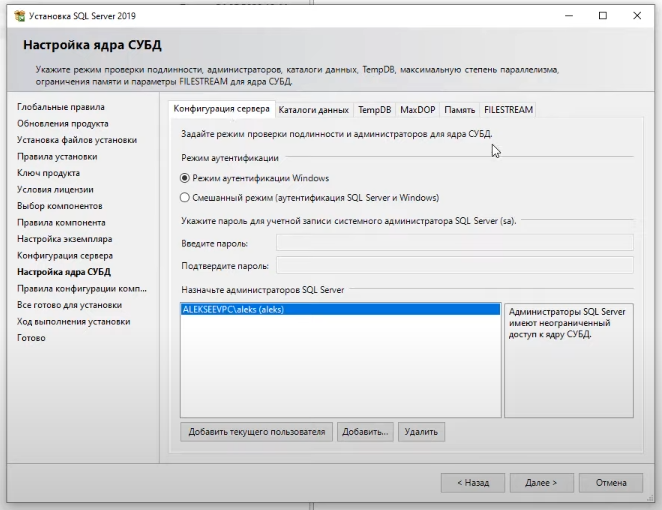


Рисунок 3.8 – Настройка ядра СУБД

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

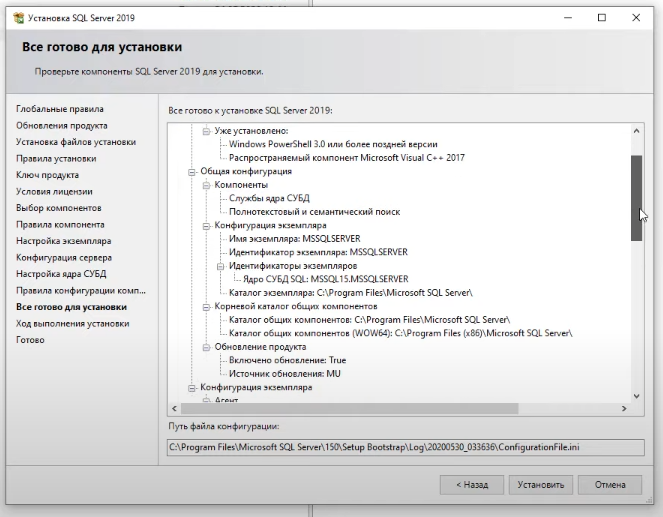


Рисунок 3.9 – Все готово для установки

Следует завершить установку и закрыть окно. Финальный экран представлен на рисунке 3.10.

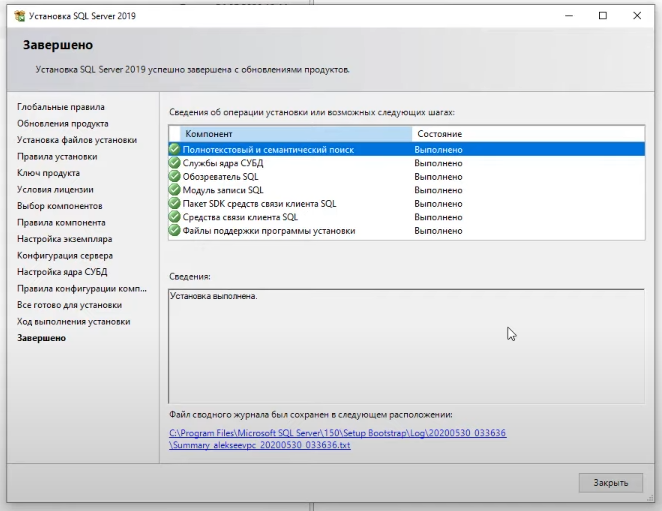
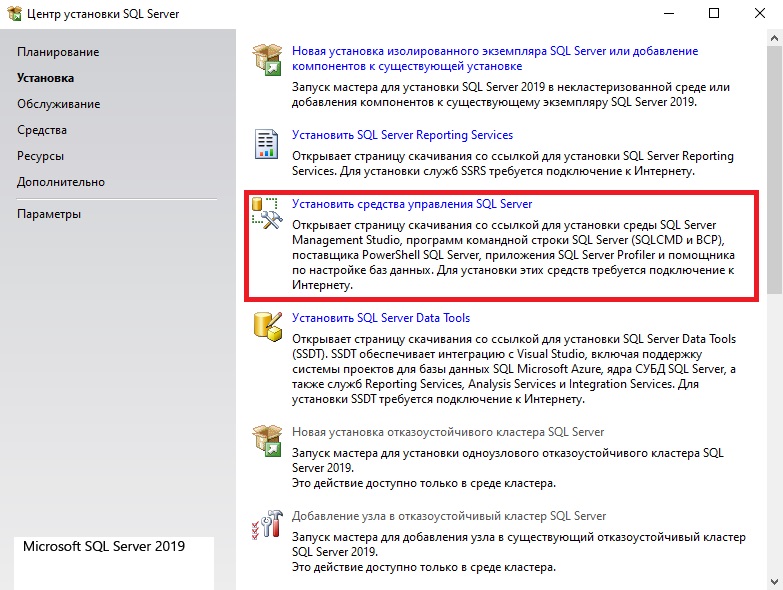


Рисунок 3.10 – Финальный экран

После того, как установка SQLServer 2019 завершена, нужно установить приложение, с помощью которого будем подключаться к серверу баз данных. Это приложение SQLServerManagementStudio (SSMS).

Следует снова перейти в центр установки SQLServer и нажать "Установить средства управления SQLServer", как показано на рисунке 3.11.

Рисунок 3.11 – Установка SQL Server Management Studio

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

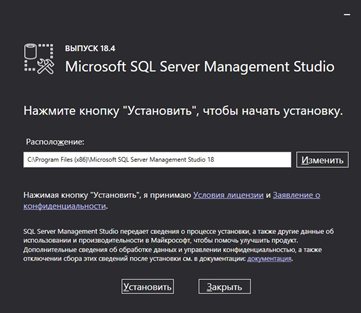


Рисунок 3.12 – УстановкаSQLServerManagementStudio

Реляционная схема базы данных представлена на рисунке 3.13.

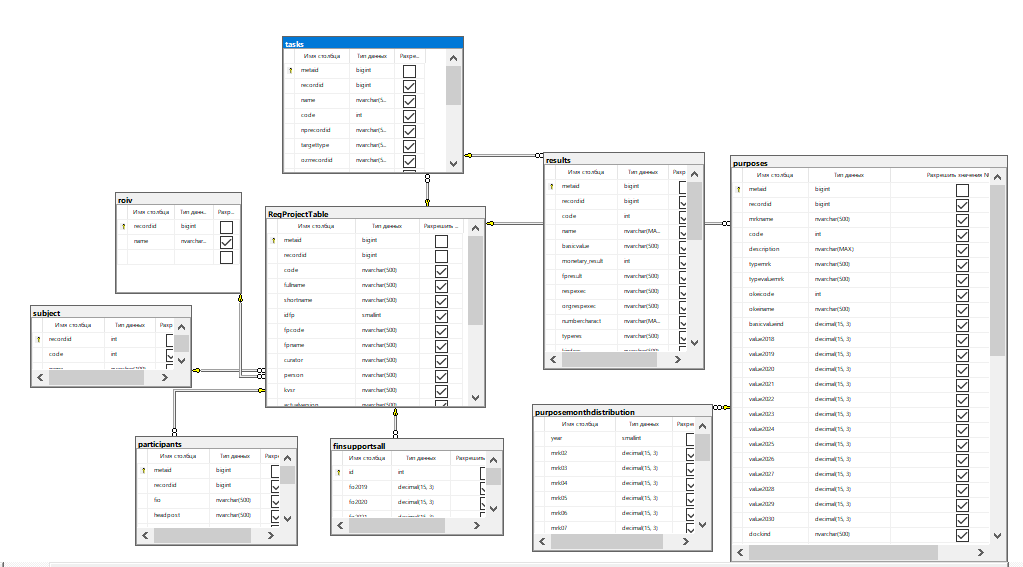


Рисунок 3.13– Реляционная схема базы данных

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

* finsupportsall – Финансирование;
* participants – Участники;
* purposemonthdistribution – Месячная цель регионального проекта;
* purposes – Цели регионального проекта;
* RegProjectTable – Региональные проекты;
* results – Результаты;
* roiv – Министерства;
* subject – Объекты;
* tasks – Задачи регионального проекта.

3.3 Инструкция по эксплуатации приложения

Приложение предназначено для просмотра всей информации существующих региональных проектов. Пользователь может установить приложения, после чего получить логин и пароль для входа и произвести его.

Чтобы начать работать с приложение достаточно его запустить. После запуска, приложение готово к работе. Окно с авторизованным пользователемпредставлено на рисунке 3.14.

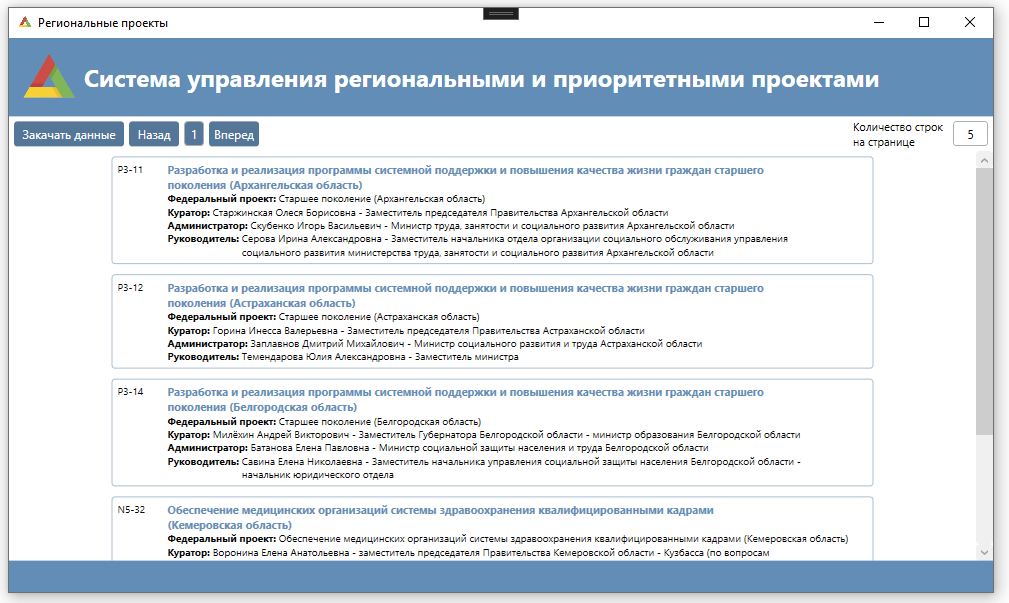


Рисунок 3.14– Окно с авторизованным пользователем

Чтобы просмотреть цели и задачи регионального проекта достаточно нажать на название интересующего проекта. Страница целей и задачей региональных проектов представлена на рисунке 3.15.

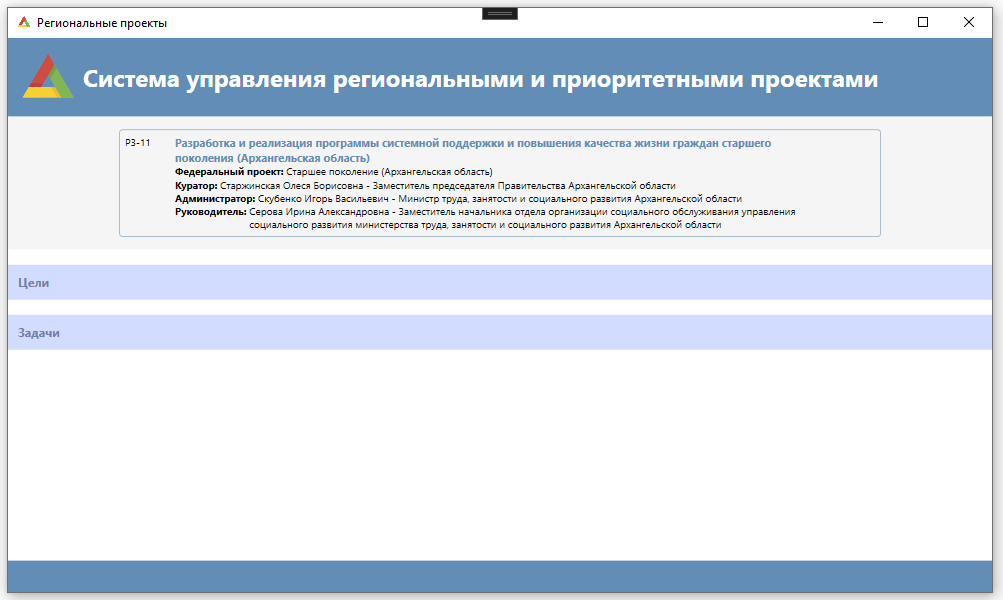


Рисунок 3.15 - Страница целей и задачей региональных проектов

Чтобы посмотреть конерктно цели достаточно просто нажать на надпись «Цели». Страница целей регионального проекта представлена нарисунке 3.16.

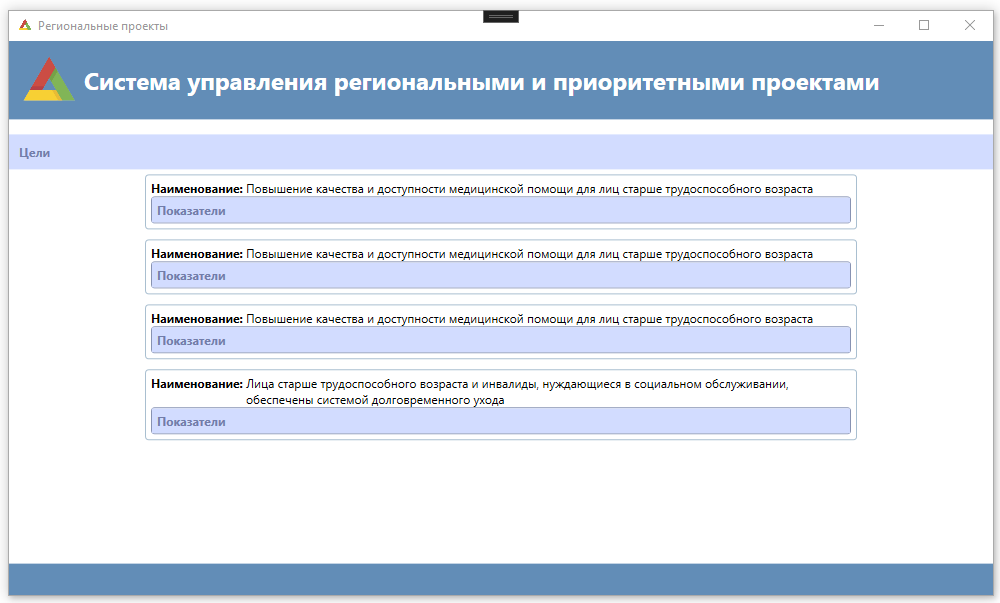


Рисунок 3.16 – Страница целей регионального проекта

Для просмотра всей информации, кроме названия, целей нужно нажать на панель «Показатели» после чего откроется окно со всей информацией. Окно всей информации цели представлена нарисунке 3.17.

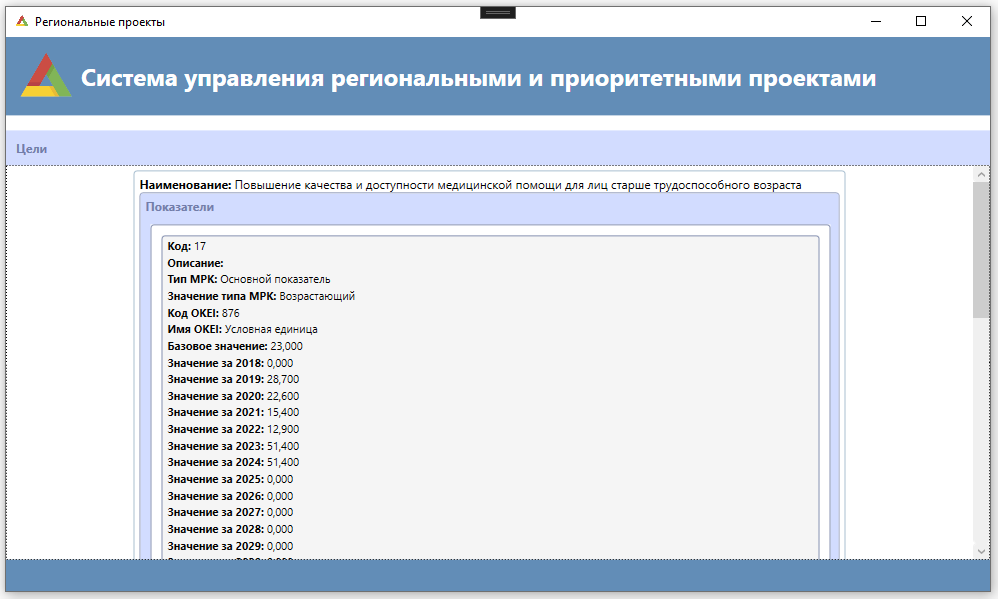


Рисунок 3.17 – Страница прохождения теста

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. Поскольку разработанный программный продукт будет поставляться в электронном виде, то никаких материалов при производстве не было затрачено, и список остается пустым.

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 4.1:

, (4.1)

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 4.1.

Таблица 4.1 – Вспомогательные материалы

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Цена за единицу, руб., коп. | Количество,  ед. изм. | Стоимость,  руб., коп. |
| Бумажный лист формата А4. | 1,50 | 100 | 150,00 |
| Ватман формата А0, шт. | 130,00 | 3 | 390,00 |
| Картридж для принтера, шт. | 1390,00 | 0,3 | 417,00 |
| Итого |  |  | 957,00 |

Стоимость отдельного вспомогательного материала*ВМi,* руб. рассчитывается по формуле 4.2:

, (4.2)

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

Ki – количество отдельного вспомогательного материала, руб.

Подстановкой указанных выше значений в формулу 4.1 и 4.2 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 4.3:

, (4.3)

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 4.4:

, (4.4)

где W – совокупная потребленная мощность, КВт×ч;

ЦЭ– стоимость одного КВт×ч электроэнергии, руб.; составляет 4,26 руб.

Совокупная потребленная мощность *W* рассчитывается по формуле 4.5:

, (4.5)

где Wi – мощность, потребляемая отдельным устройством, КВт.

В таблице 4.2 представлены все возможные устройства, используемые в разработке продукта, потребляющие электроэнергию.

Таблица 4.2 – Используемые устройства

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование | Потребляемая мощность, КВт | Продолжительность эксплуатации в смену, час. | Количество смен в цикле производства (разработки), шт. | Потребленная мощность, КВт×ч |
| Ноутбук | 0,6 | 6 | 14 | 50,40 |
| Принтер | 0,34 | 0,5 | 7 | 1,19 |
| Итого |  |  |  | 51,59 |

Потребленная мощность *Wi*, кВт×ч для каждого из устройств рассчитывается по формуле 4.6:

, (4.6)

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 4.2 в формулу 4.5 получено значение потребленной мощности:

*кВт×ч*

Подстановкой указанных выше значений в формулу 4.4, получаем затраты на электроэнергию:

Подстановкой указанных выше значений в формулу 4.3, получаем затраты на энергетические ресурсы:

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

Амортизация (от англ. "depreciation", лат. "amortizatio" — "погашение") — это перенос основных средств в процессе производства на стоимость продукции по мере их износа (материального и морального). Другими словами, это списание с баланса денежных средств по мере устаревания объектов [9].

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 4.7:

, (4.7)

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

Для разработки программного продукта потребовалось оборудование и нематериальные активы, перечисленные в таблице 4.3.

Таблица 4.3 – Информация по расчету амортизационных отчислений

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование основного средства | Изначальная стоимость, руб., коп. | Срок службы, лет | Сумма ежегодного амортизационного отчисления,  руб., коп. | Период использования в проекте, кален. дн. | Сумма амортиза-ционного отчисления, руб., коп. |
| Офисный стол | 8 317,00 | 5 | 1 663,40 | 14 | 63,80 |
| Офисное кресло | 5 680,00 | 5 | 1 136,00 | 14 | 43,57 |
| Компьютер | 36 258,00 | 5 | 7 251,60 | 14 | 278,14 |
| Принтер | 10 590,00 | 5 | 2 118,00 | 7 | 40,62 |
| Visual Studio | - | - | - | 14 | - |

Окончание таблицы 4.3.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование основного средства | Изначальная стоимость, руб., коп. | Срок службы, лет | Сумма ежегодного амортизационного отчисления,  руб., коп. | Период использования в проекте, кален. дн. | Сумма амортиза-ционного отчисления, руб., коп. |
| Microsoft Office professionalplus 2019 | 3990,00 | 5 | 798,00 | 14 | 30,61 |
| Итого |  |  |  |  | 456,74 |

Амортизационное отчисление *Аi,* руб., включаемое в расходы по проекту, для каждого из основных средств можно вычислить по формуле 4.8:

, (4.8)

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 4.9, которое каждый год будет составлять одну и ту же величину для одного и того же устройства.

, (4.9)

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

*Срi* – срок службы основного средства, лет.

Подстановкой значений из таблицы 4.3 в формулу 4.9 получено:

Подстановкой значений из таблицы 4.3 в формулу 4.8 получено:

Подстановкой значений из таблицы 4.3 в формулу 4.7 получено:

Далее для каждого основного средства рассчитывается период его использования в проекте в календарных днях *КДi*, формула 4.10:

, (4.10)

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 указанных выше значений в формулу 4.10 получено:

Следующим этапом сметного расчета является затраты на оплату труда.

Фонд оплаты труда ЗП, руб., включаемый в затраты по проекту, можно рассчитать по формуле 4.11:

, (4.11)

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 4.12:

, (4.12)

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 рабочихОкЗП, руб., участвовавших в проекте, оплачиваемых по окладной системе оплаты труда, вычисляется по формуле 4.13:

, (4.13)

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

В таблице 4.4 сведены данные по работникам, принимавшим участие в проекте, оплачиваемым по окладной системе оплаты труда.

Таблица 4.4 – Заработная плата

|  |  |  |  |
| --- | --- | --- | --- |
| Должность, профессия работника | Месячный оклад, руб., коп. | Отработано смен, шт. | Заработная плата, руб., коп. |
| Инженер-программист | 35500,00 | 14 | 24 362,75 |
| Итого |  |  | 24 362,75 |

В рамках проекта заработная платаОкЗПi, руб., заработанная каждым из работников, оплачиваемых по окладной системе оплаты труда, рассчитывается по формуле 4.14:

, (4.14)

где Оi– оклад работника, руб.,

КСi– количество смен (рабочих дней), которое он работал по проекту, шт., составляет 14 дней;

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

Подстановкой указанных выше значений в формулу 4.13 и 4.14 получено:

Заработная плата всех рабочих, оплачиваемых по сдельной системе оплаты труда, , руб., рассчитывается по формуле 4.15:

(4.15)

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 платаравна нулю.

Подстановкой указанных выше значений в формулу 4.12 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 10-13%. Таким образом, сумму затрат на дополнительную заработную плату ДЗП, руб., можно рассчитать по формуле 4.16:

, (4.16)

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

КДЗП – коэффициент, %.

В рамках проекта КДЗП выбран равным 11%.

Подстановкой указанных выше значений в формулу 4.16 получено:

Подстановкой указанных выше значений в формулу 4.11 получено:

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

Социальное страхование — это система, по которой государство защищает людей, временно оставшихся без дохода по каким-то причинам — например, из-за болезни. Фонд социального страхования — ФСС — существует, чтобы государство могло контролировать расходы на социальное страхование граждан, по[12].

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 4.17:

, (4.17)

где ЗП – фонд оплаты труда по проекту, рассчитанный ранее (руб.), составляет 27 042,65 руб.;

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

Ставка страховых взносов ССВна 2023 год составляет 30%.

Подстановкой указанных выше значений в формулу 4.17 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 4.18:

, (4.18)

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

СНР– размер накладных расходов, %.

Размер накладных расходов СНРсоставляет 10%.

Подстановкой указанных выше значений в формулу 4.18 получено:

Восьмым этапом сметного расчета является расчетсебестоимости.

Себестоимость — это совокупность всех затрат на производство и реализацию товаров. [12].

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 4.19.

(4.19)

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 4.19 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 4.20:

, (4.20)

где Сб – себестоимость производства или разработки, руб.;

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

Рассчитываемый процент прибыли от производства программного продукта СП составляет 25%.

Подстановкой указанных выше значений в формулу 4.20 получено:

Следующим этапом сметного расчета является налог на добавленную стоимость.

Налог на добавленную стоимость (НДС) — косвенный налог, способ изъять в бюджет часть стоимости товара, работы или услуги. НДС присутствует на всех стадиях процесса производства товаров, работ и услуг и передается в бюджет после реализации, по [10].

Для расчета налога на добавленную стоимость НДС, руб. можно воспользоваться формулой 4.21:

, (4.21)

где Сб – себестоимость разработки, руб.;

П – прибыль, руб.;

СНДС – ставка налога на добавленную стоимость, %.

Ставка налога СНДС на добавленную стоимость в 2021 годусоставляет 20%.

Подстановкой указанных выше значений в формулу 4.21 получено:

Конечная стоимость разработки *С*, руб. рассчитывается как сумма себестоимости, прибыли и НДС, для расчёта можно воспользоваться формулой 4.22:

, (4.22)

где Сб– себестоимость разработки, руб.

П – прибыль. руб.;

НДС – сумма налога на добавленную стоимость, руб.;

Подстановкой указанных выше значений в формулу 4.22 получено:

Исходя из вышеперечисленных расчетов следует, что стоимость разработанного приложения составляет 59 387,65 руб.

Разработанный программный продукт является востребованным для учебного заведения, так как это программное решение повысит удобство просмотра расписания учебных занятий среди студентов, их родителей и преподавателей. Данная стоимость является небольшой с учётом того, что оплата совершается единоразово, а дальнейшие улучшения программного продукта потребуют минимальных затрат.

5 Раздел охраны труда

Охрана труда – это целая система законодательных и нормативно-правовых актов, технических, гигиенических, лечебно-профилактических мероприятий и средств, которые обеспечивают безопасность, сохранение здоровья и работоспособности человека в процессе труда. В наши дни труд стал более интенсивным и требует огромных затрат умственной, эмоциональной и физической нагрузок.

На рабочем месте программист осуществляет трудовую деятельность и проводит большую часть рабочего времени. Правильная организация рабочего места программиста повышает производительность труда от 8 до 20%. Следуя рекомендациям ГОСТ 12.2.032-78, необходимо организовать рабочее место таким образом, чтобы взаимное расположение всех его элементов соответствовало физическим и психологическим требованиям. Главные элементы рабочего места программиста – это письменный стол и кресло. Рабочее место организуется в соответствии с ГОСТ 12.2.032-78, информация из работы [11].

Площадь рабочего места с компьютером с жидкокристаллическим или плазменным экраном должна быть не менее 4,5 кв. м, а расстояние между столами с мониторами (от тыла одного монитора до экрана другого) не менее 2 м. Монитор должен располагаться на расстоянии 50-70 см от глаз программиста. Параметры рабочего стола сотрудника: возможность регулировки высоты рабочего стола, или точная высота — 72,5 см, ширина — 80, 100, 120 или 140 см, глубина рабочего стола 80 или 100 см, высота и ширина пространства под столешницей (для ног) – не менее 50 см, глубина на уровне колен не менее 45 см, а на уровне вытянутых ног не менее 65 см.

Правильное освещение рабочего места – это очень важный момент в трудовой деятельности человека, влияющий на эффективность труда, при этом такой момент предупреждает травматизм и профессиональные заболевания. При недостаточном освещении приходится напрягать зрение, при этом ослабляется внимание и это приводит к наступлению преждевременной утомленности. Слишком яркое освещение тоже плохо, так как оно вызывает ослепление, раздражение и резь в глазах. При искусственном освещении, источниками света служат два вида ламп: лампы 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 [12].

При работе компьютерной техники выделяется много тепла, что может привести к пожароопасной ситуации. Источниками зажигания так же могут служить приборы, применяемые для технического обслуживания, устройства электропитания, кондиционеры воздуха. Серьёзную опасность представляют различные электроизоляционные материалы, используемые для защиты от механических воздействий отдельных радиодеталей. В связи с этим, участки, на которых используется компьютерная техника, по пожарной опасности относятся к категории пожароопасных “В”.При пожаре люди должны покинуть помещение в течение минимального времени. В помещениях с компьютерной техникой, недопустимо применение воды и пены ввиду опасности повреждения или полного выхода из строя дорогостоящего электронного оборудования. Для тушения пожаров необходимо применять углекислотные и порошковые огнетушители, которые обладают высокой скоростью тушения, большим временем действия, возможностью тушения электроустановок, высокой эффективностью борьбы с огнем. Воду разрешено применять только во вспомогательных помещениях, информация из работы [13].

Заключение

По итогу работы было разработано приложение, позволяющее пользовталею просматривать сводку региональных проектов их цели и задачи.

Главным достоинством можно выделить простоту в использовании приложения. Все дейсвтия выполняются на интуитовно понятном уровне.

Список используемых источников

1. <https://kazedu.com/referat/133091/5>
2. <https://ruprogi.ru/software/visual-studio>
3. <https://gb.ru/posts/c_sharp_ides>
4. <https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/overview/?view=netdesktop-6.0>
5. <https://metanit.com/sharp/wpf/11.php>
6. <https://steptosleep.ru/ролевая-модель/#:~:text=Основная%20идея%20ролевой%20модели%20контроля,типов%20их%20активностей%20в%20системе>
7. <https://studopedia.ru/22_29871_neobhodimost-otladki-programmnogo-produkta.html>
8. <https://infopedia.su/4x1ec5.html>
9. <https://sergeygavaga.gitbooks.io/kurs-lektsii-testirovanie-programnogo-obespecheni/content/lektsiya-4-ch3.html>
10. <https://www.retail.ru/rbc/pressreleases/tsentr-povysheniya-kvalifikatsii-lider-organizatsiya-rabochego-mesta-ofisnogo-rabotnika/>
11. <https://xn--d1aux.xn--p1ai/opisanie-rabochego-mesta-programmista-na-predpriyatii/>
12. <https://studopedia.ru/8_107307_osveshchenie-pomeshcheniy-vichislitelnih-tsentrov.html>

Приложение А

Программный код класса MainPage.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

using System.Net;

using Newtonsoft.Json;

using static DesktopApp.JsonClass;

using DesktopApp.Base;

using System.Data.Entity.Validation;

using System.Globalization;

using System.Data.Entity;

using DesktopApp.Windows;

using DesktopApp.Classes;

namespace DesktopApp.Pages

{

public class ViewModel

{

public List<TileData> Items { get; set; }

}

/// <summary>

/// Логика взаимодействия для MainPage.xaml

/// </summary>

public partial class MainPage : Page

{

private DatabaseHelper databaseHelper;

public MainPage(List<UserLogin> user)

{

InitializeComponent();

databaseHelper = new DatabaseHelper();

\_user = user;

items = databaseHelper.GetItems();

SetPage(currentPage);

CheckUserRole();

}

private List<UserLogin> \_user;

private List<TileData> items;

private int currentPage = 0;

private int itemsPerPage = 5;

private void CheckUserRole()

{

if (\_user == null) return;

foreach (var rpt in \_user)

{

RegUsers.Visibility = Visibility.Collapsed;

if (rpt.Role == 1)

{

RegUsers.Visibility = Visibility.Visible;

}

}

}

//-------------------пагинация-----------------

private void SetPage(int page)

{

currentPage = page;

PageNumber.Text = $"{currentPage + 1}";

List<TileData> tileDatas = items.Select(x => x).Skip(currentPage \* itemsPerPage).Take(itemsPerPage).ToList();

DataContext = new ViewModel { Items = tileDatas };

}

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

{

if (currentPage > 0)

{

SetPage(currentPage - 1);

}

}

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

{

if (currentPage < (items.Count / itemsPerPage) - 1)

{

SetPage(currentPage + 1);

}

}

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

{

itemsPerPage = Int32.Parse(tbItemsPerPage.Text);

}

//-------------------навигация-----------------

private void TextBlock\_PreviewMouseDown(object sender, MouseButtonEventArgs e)

{

var regProjectItem = ((FrameworkElement)sender).DataContext as TileData; //коллекция элементов

if (regProjectItem != null)

{

this.NavigationService.Navigate(new MoreInfoPage(regProjectItem.Metaid, \_user));

//MainPageFrame.Navigate(new MoreInfoPage(regProjectItem.Metaid));

}

}

//-------------------выделение текста-----------------

private void tbRegProject\_MouseEnter(object sender, MouseEventArgs e)

{

var focusTextBlock = (TextBlock)sender;

focusTextBlock.Foreground = new SolidColorBrush(Color.FromRgb(77, 112, 145));

focusTextBlock.TextDecorations = TextDecorations.Underline;

}

private void tbRegProject\_MouseLeave(object sender, MouseEventArgs e)

{

var focusTextBlock = (TextBlock)sender;

focusTextBlock.Foreground = new SolidColorBrush(Color.FromRgb(98, 141, 183));

focusTextBlock.TextDecorations = null;

}

//-------------------основной процесс-----------------

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

{

UrlParse();

}

private void UrlParse()

{

//3 переменных для ссылки

string url = "https://budget.gov.ru/epbs/registry/7710168360-REGIONALPROJECT/data";

Root jsonObject = JsonConvert.DeserializeObject<Root>(new WebClient().DownloadString(url));

//---1---

RoivUpload(jsonObject);

SubjectUpload(jsonObject);

//---2---

RegProjectUpload(jsonObject);

//---3---

PurposeUpload(jsonObject);

FinsupportsallUpload(jsonObject);

ParticipantUpload(jsonObject);

TaskUpload(jsonObject);

//---4---

PurposemonthdistributionUpload(jsonObject);

ResultUpload(jsonObject);

}

private void RoivUpload(Root root)

{

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

{

var roiv = root.data[i].roiv;

var data = new roiv

{

recordid = Int64.Parse(roiv.recordid),

name = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(roiv.name))

};

SourceCore.RegProjectDatabase.roiv.Add(data);

}

try

{

try

{

SourceCore.RegProjectDatabase.SaveChanges();

}

catch { }

}

catch (DbEntityValidationException ex)

{

foreach (DbEntityValidationResult validationError in ex.EntityValidationErrors)

{

MessageBox.Show("Object: " + validationError.Entry.Entity.ToString());

MessageBox.Show("");

foreach (DbValidationError err in validationError.ValidationErrors)

{

MessageBox.Show(err.ErrorMessage + "");

}

}

}

}

private void SubjectUpload(Root root)

{

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

{

var subject = root.data[i].subject;

var data = new subject

{

recordid = Int32.Parse(subject.recordid),

code = Int32.Parse(subject.code),

name = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(subject.name))

};

SourceCore.RegProjectDatabase.subject.Add(data);

}

try

{

try

{

SourceCore.RegProjectDatabase.SaveChanges();

}

catch { }

}

catch (DbEntityValidationException ex)

{

foreach (DbEntityValidationResult validationError in ex.EntityValidationErrors)

{

MessageBox.Show("Object: " + validationError.Entry.Entity.ToString());

MessageBox.Show("");

foreach (DbValidationError err in validationError.ValidationErrors)

{

MessageBox.Show(err.ErrorMessage + "");

}

}

}

}

private void RegProjectUpload(Root root)

{

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

{

var regProject = root.data[i];

var data = new RegProjectTable

{

metaid = Int64.Parse(regProject.metaid),

recordid = Int64.Parse(regProject.recordid),

code = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(regProject.code)),

fullname = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(regProject.fullname)),

shortname = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(regProject.shortname)),

idfp = short.Parse(root.data[i].idfp),

fpcode = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(regProject.fpcode)),

fpname = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(regProject.fpname)),

curator = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(regProject.curator)),

person = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(regProject.person)),

kvsr = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(regProject.kvsr)),

actualversion = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(regProject.actualversion)),

subject = Int32.Parse(regProject.subject.recordid),

roiv = Int64.Parse(regProject.roiv.recordid)

};

SourceCore.RegProjectDatabase.RegProjectTable.Add(data);

}

try

{

try

{

SourceCore.RegProjectDatabase.SaveChanges();

}

catch { }

}

catch (DbEntityValidationException ex)

{

foreach (DbEntityValidationResult validationError in ex.EntityValidationErrors)

{

MessageBox.Show("Object: " + validationError.Entry.Entity.ToString());

MessageBox.Show("");

foreach (DbValidationError err in validationError.ValidationErrors)

{

MessageBox.Show(err.ErrorMessage + "");

}

}

}

}

private decimal DecimalPoint(string num)

{

return decimal.Parse(num, CultureInfo.InvariantCulture);

}

private void PurposeUpload(Root root)

{

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

{

var dp = root.data[i];

for (int j = 0; j < dp.purposes.Count; j++)

{

var purpose = dp.purposes[j];

var data = new purposes

{

metaid = Int64.Parse(purpose.metaid),

recordid = Int64.Parse(purpose.recordid),

mrkname = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(purpose.mrkname)),

code = Int32.Parse(purpose.code),

description = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(purpose.description)),

typemrk = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(purpose.typemrk)),

typevaluemrk = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(purpose.typevaluemrk)),

okeicode = Int32.Parse(purpose.okeicode),

okeiname = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(purpose.okeiname)),

basicvalueind = DecimalPoint(purpose.basicvalueind),

value2018 = DecimalPoint(purpose.value2018),

value2019 = DecimalPoint(purpose.value2019),

value2020 = DecimalPoint(purpose.value2020),

value2021 = DecimalPoint(purpose.value2021),

value2022 = DecimalPoint(purpose.value2022),

value2023 = DecimalPoint(purpose.value2023),

value2024 = DecimalPoint(purpose.value2024),

value2025 = DecimalPoint(purpose.value2025),

value2026 = DecimalPoint(purpose.value2026),

value2027 = DecimalPoint(purpose.value2027),

value2028 = DecimalPoint(purpose.value2028),

value2029 = DecimalPoint(purpose.value2029),

value2030 = DecimalPoint(purpose.value2030),

dockind = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(purpose.dockind)),

approgv = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(purpose.approgv)),

approvtdatenpa = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(purpose.approvtdatenpa)),

numbernpa = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(purpose.numbernpa)),

namenpa = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(purpose.namenpa)),

sourcedata = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(purpose.sourcedata)),

fpmrkid = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(purpose.fpmrkid)),

fpmrkname = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(purpose.fpmrkname)),

tasksrecordid = Int64.Parse(purpose.tasksrecordid),

taskcode = Int32.Parse(purpose.taskcode),

taskname = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(purpose.taskname)),

taskrespexec = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(purpose.taskrespexec)),

targettype = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(purpose.targettype)),

ozrname = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(purpose.ozrname)),

ozrnum = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(purpose.ozrnum)),

ozrbeneficiar = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(purpose.ozrbeneficiar)),

ozrokeicode = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(purpose.ozrokeicode)),

ozrokeiname = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(purpose.ozrokeiname)),

ozrtype = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(purpose.ozrtype)),

yearend = Int32.Parse(purpose.yearend),

establishact = Int32.Parse(purpose.establishact),

lvlmrk = Int32.Parse(purpose.lvlmrk),

metaid\_rp = Int64.Parse(dp.metaid)

};

SourceCore.RegProjectDatabase.purposes.Add(data);

}

}

try

{

try

{

SourceCore.RegProjectDatabase.SaveChanges();

}

catch { }

}

catch (DbEntityValidationException ex)

{

foreach (DbEntityValidationResult validationError in ex.EntityValidationErrors)

{

MessageBox.Show("Object: " + validationError.Entry.Entity.ToString());

MessageBox.Show("");

foreach (DbValidationError err in validationError.ValidationErrors)

{

MessageBox.Show(err.ErrorMessage + "");

}

}

}

}

private void PurposemonthdistributionUpload(Root root)

{

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

{

var dp = root.data[i];

for (int j = 0; j < dp.purposes.Count; j++)

{

var purpose = dp.purposes[j];

for (int k = 0; k < purpose.purposemonthdistribution.Count; k++)

{

var prb = purpose.purposemonthdistribution[k];

var data = new purposemonthdistribution

{

year = short.Parse(prb.year),

mrk02 = DecimalPoint(prb.mrk02),

mrk03 = DecimalPoint(prb.mrk03),

mrk04 = DecimalPoint(prb.mrk04),

mrk05 = DecimalPoint(prb.mrk05),

mrk06 = DecimalPoint(prb.mrk06),

mrk07 = DecimalPoint(prb.mrk07),

mrk08 = DecimalPoint(prb.mrk08),

mrk09 = DecimalPoint(prb.mrk09),

mrk10 = DecimalPoint(prb.mrk10),

mrk11 = DecimalPoint(prb.mrk11),

mrk12 = DecimalPoint(prb.mrk12),

mrkendyaer = DecimalPoint(prb.mrkendyaer),

reasonsmo = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(prb.reasonsmo)),

recordid = Int64.Parse(prb.recordid),

metaid\_purposes = Int64.Parse(purpose.metaid)

};

SourceCore.RegProjectDatabase.purposemonthdistribution.Add(data);

}

}

}

try

{

try

{

SourceCore.RegProjectDatabase.SaveChanges();

}

catch { }

}

catch (DbEntityValidationException ex)

{

foreach (DbEntityValidationResult validationError in ex.EntityValidationErrors)

{

MessageBox.Show("Object: " + validationError.Entry.Entity.ToString());

MessageBox.Show("");

foreach (DbValidationError err in validationError.ValidationErrors)

{

MessageBox.Show(err.ErrorMessage + "");

}

}

}

}

private void FinsupportsallUpload(Root root)

{

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

{

var dp = root.data[i];

for (int j = 0; j < dp.finsupportsall.Count; j++)

{

var fs = dp.finsupportsall[j];

var data = new finsupportsall

{

fo2019 = DecimalPoint(fs.fo2019),

fo2020 = DecimalPoint(fs.fo2020),

fo2021 = DecimalPoint(fs.fo2021),

fo2022 = DecimalPoint(fs.fo2022),

fo2023 = DecimalPoint(fs.fo2023),

fo2024 = DecimalPoint(fs.fo2024),

finsource = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(fs.finsource)),

fo\_total = DecimalPoint(fs.fo\_total),

finsourcecode = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(fs.finsourcecode)),

metaid\_rp = Int64.Parse(dp.metaid)

};

SourceCore.RegProjectDatabase.finsupportsall.Add(data);

}

}

try

{

try

{

SourceCore.RegProjectDatabase.SaveChanges();

}

catch { }

}

catch (DbEntityValidationException ex)

{

foreach (DbEntityValidationResult validationError in ex.EntityValidationErrors)

{

MessageBox.Show("Object: " + validationError.Entry.Entity.ToString());

MessageBox.Show("");

foreach (DbValidationError err in validationError.ValidationErrors)

{

MessageBox.Show(err.ErrorMessage + "");

}

}

}

}

private void ParticipantUpload(Root root)

{

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

{

var dp = root.data[i];

for (int j = 0; j < dp.participants.Count; j++)

{

var participant = dp.participants[j];

var data = new participants

{

metaid = Int64.Parse(participant.metaid),

recordid = Int64.Parse(participant.recordid),

fio = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(participant.fio)),

headpost = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(participant.headpost)),

immsupervisor = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(participant.immsupervisor)),

percemploy = Int32.Parse(participant.percemploy),

roleinproj = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(participant.roleinproj)),

codereestr = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(participant.codereestr)),

metaid\_rp = Int64.Parse(dp.metaid)

};

SourceCore.RegProjectDatabase.participants.Add(data);

}

}

try

{

try

{

SourceCore.RegProjectDatabase.SaveChanges();

}

catch { }

}

catch (DbEntityValidationException ex)

{

foreach (DbEntityValidationResult validationError in ex.EntityValidationErrors)

{

MessageBox.Show("Object: " + validationError.Entry.Entity.ToString());

MessageBox.Show("");

foreach (DbValidationError err in validationError.ValidationErrors)

{

MessageBox.Show(err.ErrorMessage + "");

}

}

}

}

private void TaskUpload(Root root)

{

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

{

var dp = root.data[i];

for (int j = 0; j < dp.tasks.Count; j++)

{

var task = dp.tasks[j];

var data = new tasks

{

metaid = Int64.Parse(task.metaid),

recordid = Int64.Parse(task.recordid),

name = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(task.name)),

code = Int32.Parse(task.code),

nprecordid = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(task.nprecordid)),

targettype = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(task.targettype)),

ozrrecordid = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(task.ozrrecordid)),

ozrname = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(task.ozrname)),

ozrnum = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(task.ozrnum)),

ozrbeneficiar = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(task.ozrbeneficiar)),

ozrokeicode = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(task.ozrokeicode)),

ozrokeiname = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(task.ozrokeiname)),

ozrtype = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(task.ozrtype)),

metaid\_rp = Int64.Parse(dp.metaid)

};

SourceCore.RegProjectDatabase.tasks.Add(data);

}

}

try

{

try

{

SourceCore.RegProjectDatabase.SaveChanges();

}

catch { }

}

catch (DbEntityValidationException ex)

{

foreach (DbEntityValidationResult validationError in ex.EntityValidationErrors)

{

MessageBox.Show("Object: " + validationError.Entry.Entity.ToString());

MessageBox.Show("");

foreach (DbValidationError err in validationError.ValidationErrors)

{

MessageBox.Show(err.ErrorMessage + "");

}

}

}

}

private void ResultUpload(Root root)

{

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

{

var dp = root.data[i];

for (int j = 0; j < dp.tasks.Count; j++)

{

var task = dp.tasks[j];

for (int k = 0; k < task.results.Count; k++)

{

var result = task.results[k];

var data = new results

{

metaid = Int64.Parse(result.metaid),

recordid = Int64.Parse(result.recordid),

code = Int32.Parse(result.code),

name = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(result.name)),

basicvalue = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(result.basicvalue)),

monetary\_result = Int32.Parse(result.monetary\_result),

fpresult = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(result.fpresult)),

respexec = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(result.respexec)),

orgrespexec = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(result.orgrespexec)),

numbercharact = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(result.numbercharact)),

typeres = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(result.typeres)),

kindres = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(result.kindres)),

realizemo = Int32.Parse(result.realizemo),

notsubjfund = Int32.Parse(result.notsubjfund),

subjfund = Int32.Parse(result.subjfund),

iskey = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(result.iskey)),

okeicode = short.Parse(result.okeicode),

okeiname = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(result.okeiname)),

sourcedata = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(result.sourcedata)),

cumulative = Int32.Parse(result.cumulative),

cbaccumulationtype = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(result.cbaccumulationtype)),

typevaluemrk = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(result.typevaluemrk)),

costwaycode = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(result.costwaycode)),

directionexpenses = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(result.directionexpenses)),

direxpcoderesf = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(result.direxpcoderesf)),

direxpnameresf = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(result.direxpnameresf)),

executpost = Encoding.UTF8.GetString(Encoding.GetEncoding("Windows-1251").GetBytes(result.executpost)),

metaid\_task = Int64.Parse(task.metaid)

};

SourceCore.RegProjectDatabase.results.Add(data);

}

}

}

try

{

try

{

SourceCore.RegProjectDatabase.SaveChanges();

}

catch { }

}

catch (DbEntityValidationException ex)

{

foreach (DbEntityValidationResult validationError in ex.EntityValidationErrors)

{

MessageBox.Show("Object: " + validationError.Entry.Entity.ToString());

MessageBox.Show("");

foreach (DbValidationError err in validationError.ValidationErrors)

{

MessageBox.Show(err.ErrorMessage + "");

}

}

}

}

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

{

RegistrationWindow registrationWindow = new RegistrationWindow();

registrationWindow.Show();

}

}

}