2022

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc137064178)

[1 Исследовательский раздел 4](#_Toc137064179)

[2 Конструкторский раздел 8](#_Toc137064180)

[2.1 Проектирование информационной модели данных 8](#_Toc137064181)

[2.2 Проектирование серверной части приложения 9](#_Toc137064182)

[2.3 Проектирование клиентской части приложения 14](#_Toc137064185)

[2.4 Обеспечение коллективного доступа. Защита информации 25](#_Toc137064190)

[3 Технологическая часть 28](#_Toc137064191)

[3.1 Тестирование и отладка приложения 28](#_Toc137064192)

[3.2 Инструкция администратора базы данных 31](#_Toc137064193)

[3.3 Инструкция по эксплуатации приложения 41](#_Toc137064194)

[4 Технико-экономический раздел 55](#_Toc137064195)

[5 Раздел охраны труда 66](#_Toc137064196)

[Заключение 68](#_Toc137064197)

[Список используемых источников 69](#_Toc137064198)

[Приложение А 70](#_Toc137064199)

Введение

В рамках данного дипломного проекта планируется разработка приложения, где будет реализован процесс заказа товаров, обработки заявок, добавления новых товаров, а также составления плана закупок. Пользователь сможет выбирать любой доступный товар, его количество и добавлять его в список заявок. Администратор сможет добавлять и удалать товары из списка, а также принимать заявки пользователей или отправлять их на доработку.

Осоновной бизнес-процесс заключается в формировании печатного плана-графика закупок. Заявки пользователей будут просматриваться администратором и после подтверждения будут добавлены в этот план. Также будет представлена возможность создать структуру организации на основе сотрудников и отделов.

Данное приложение имеет потенциал к использованию в любых организациях, которым нужно подовать заявки на гос-закупки. Это приложение будет иметь простой и понятный интерфейс, а также довольно простой процесс формирования заявок и итогого плана закупок. Приложение потребует минимальных временных затрат при использовании, а также обеспечит защиту работы каждого пользователя при помощи процессов авторизации и регистрации.

Удобнее всего это будет реализовать путем использования клиент-серверной архитектуры. Все данные будут храниться в базе данных на сервере, а клиент будет взаимодействовать с клиентской частью приложения.

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 архитектуры была выбрана платформа WPF (Windows Presentetion Foundation).

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-подобная модель компоновки;**
* **богатая модель рисования;**
* **развитая текстовая модель;**
* **анимация;**
* **поддержка аудио и видео;**
* **стили и шаблоны;**
* **команды;**
* **декларативный пользовательский интерфейс;**
* **приложения на основе страниц.**

При выборе сред разработки были рассмотрены Visual Studio и Project Rider. Visual Studio – это удобная интегрированная среда разработки (IDE) от Microsoft, позволяющая быстро и эффективно создавать, и разрабатывать проект, выбрав для этого все необходимое. Среда использует платформы разработки программного обеспечения Microsoft: Windows API, Windows Forms, Windows Presentation Foundation, Windows Store и Microsoft Silverlight. 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].

Project Rider – это среда от JetBrains для работы с платформой .NET. Она обладает поддержкой полного цикла. Фирменная черта продуктов JetBrains, воплощенная и в Project Rider. С Project Rider появиться возможность организовать весь цикл создания программного обеспечения: от идеи до поддержки. Функциональность Project Rider 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 Project Rider работает с Windows, Linux и MacOS. Из минусов можно выделить её молодость. Часть функциональности еще в разработке, не все стартовые ошибки исправлены. Так же можно отметить её стоимость. Самая дешевая версия Project Rider 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 Visual Studio, так как он обладает всем необходимым функционалом для реализации проекта, а также она является бесплатной и дольше находится на рынке труда.

В интернете существует аналог разрабатываемого приложения [4], у которого есть ряд ключевых отличий. У аналога нет возможности составить структуру предприятия. Можно отметить хороший функционал у аналога, например, в отличии от разрабатываемого приложения у аналога есть возможность отправить созданный план закупок на официальный сайт.

Приложение должно предоставить возможность пользователю выбрать неоюходимой товар и оформить заявку за заказ эотого товара. Пользователю необходимо зарегистрироваться в приложении при помощи специального кода, полученного у администратора, после чего у него откроется доступ к окнам выбора товаров и собственных заявок. Данные будут автоматически браться из базы данных. В приложении будет присутствовать отдельная панель даминистратора, которая будет доступна для пользователей с определенным уровнем доступа. В панели администратора, соответсвующий пользователь сможет редактировать базу данных.

Основными процессами будут выступать процессы сбора информации в итоговый план закупок и формирования структуры организации. Эти процессы будут необходимы для генерации интерфейса, и хранения определенного формата данных для дальнейшей работы с ними.

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.

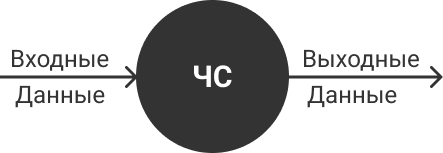


Рисунок 2.1 – Модель «Черная сфера»

Представим наше приложение в виде черной сферы. В приложении будут присутствовать такие входные данные, как пользователь и товары. На выходе будет поступать план закупок. Пользователем будет выступать человек, планирующий заказать товар в свой отдел, либо администратор, который будет редактировать список товаров и заявки пользователей. Товар – список позиций доступный к заказу. Заявки – список заказанных товаров, с которым будет работать администратор. Планом закупок будет результат, полученный после взаимодействия пользователя с товарами, после подтверждения администратором. Черная сфера с перечисленными входными и выходными параметрами представлена на рисунке 2.2.

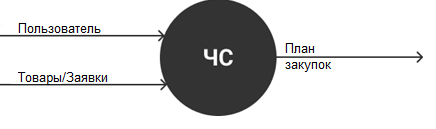


Рисунок 2.2 – Черная сфера с параметрами

В приложении существует 4 основных процесса отвечающие за генерации списков, каждый из которых хранит в себе объекты, представляющие из себя сущности с определенными свойствами необходимыми для их визуального отображения. Сами списки являются источниками данных с удобным для манипулирования форматом. Большая часть данных для заполнения подобных листов поступает из базы данных и форматируется под интерфейс.

2.2 Проектирование серверной части приложения

2.2.1 Разработка схемы базы данных

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

На первом этапе нормализации можно представить модель как связь между пользователем и товарами или заявками. Первый этап нормализации представлен на рисунке 2.3.

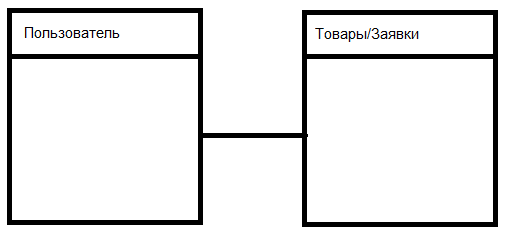


Рисунок 2.3 – Первый этап нормализации

Во втором этапе нормализации разобьём сущность Товары/Заявки на 3 сущности: Товары, Заявки и Статусы заявок. Сущность Статус заявок будет содержать в себе информацию о статусе заявки пользователя. Второй этап нормализации на рисунке 2.4.

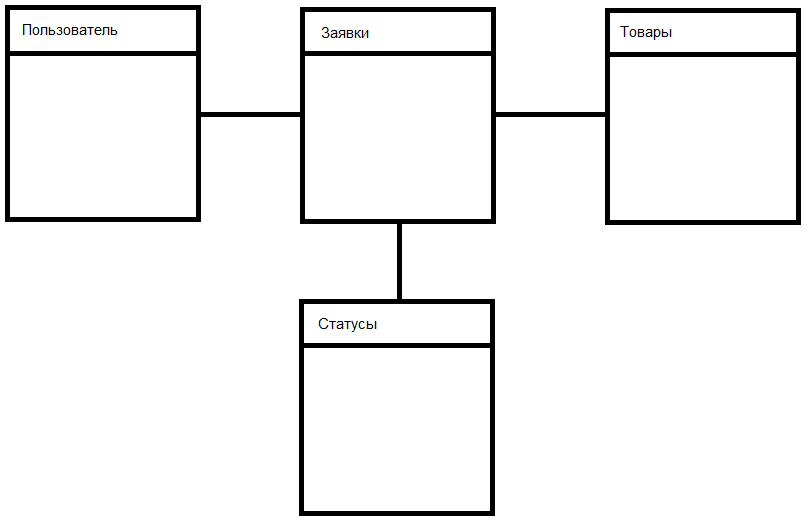


Рисунок 2.4 – Второй этап нормализации

В третьем заключительном этапе нормализации получим полную концептуальную схему, вынеся из пользователя в отдельную сущность личные данные сотрудников. Нстроим связь пользователей с заявками через отделы, а также добавим структуру организации. Третий этап нормализации представлен на рисунке 2.5.

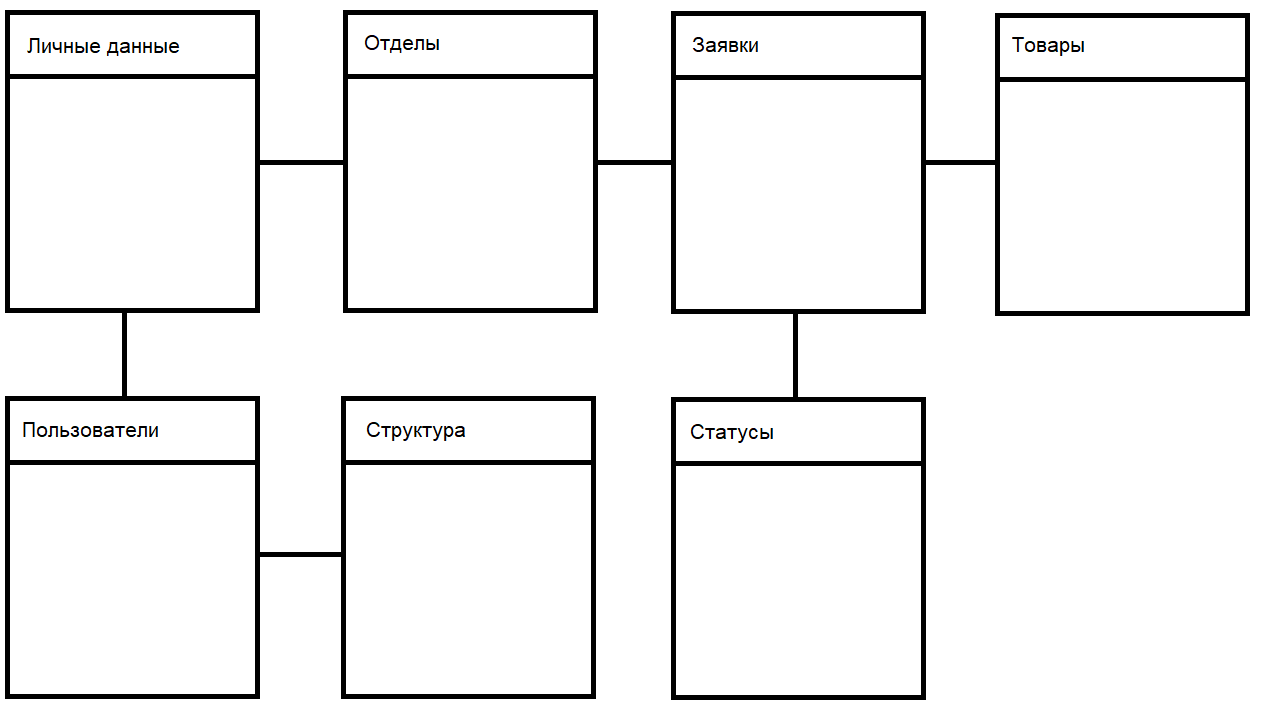


Рисунок 2.5 – Третий этап нормализации

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

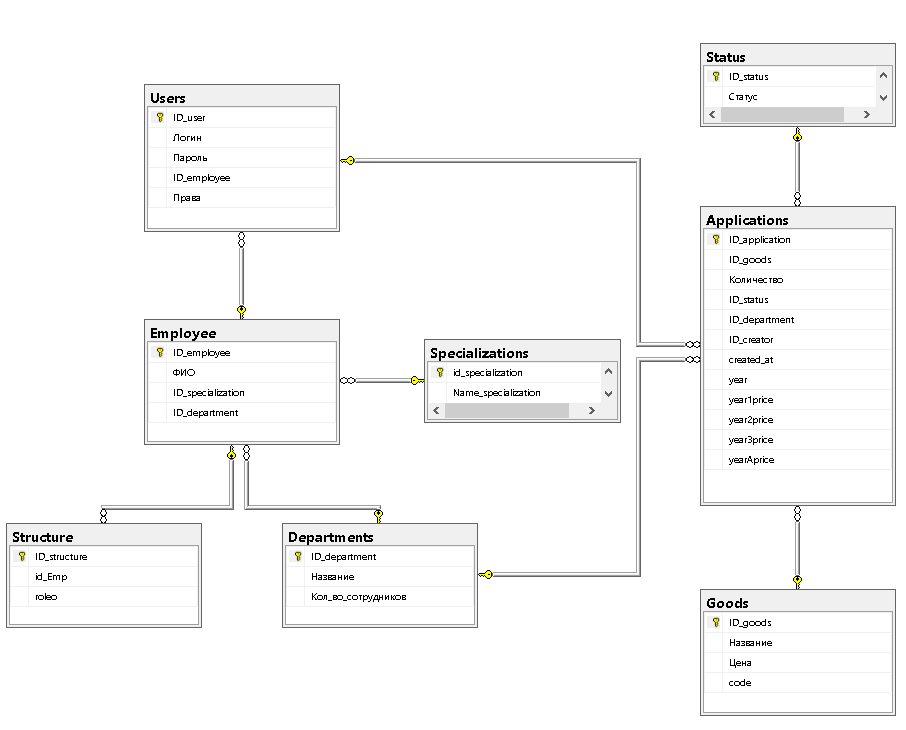


Рисунок 2.6 – Логическая модель данных

Получим физическую модель данных, включающую ассоциативные таблицы, которые иллюстрируют отношения между сущностями, а также первичные и внешние ключи для связи данных. Физическая модель данных представлена на рисунке 2.7.

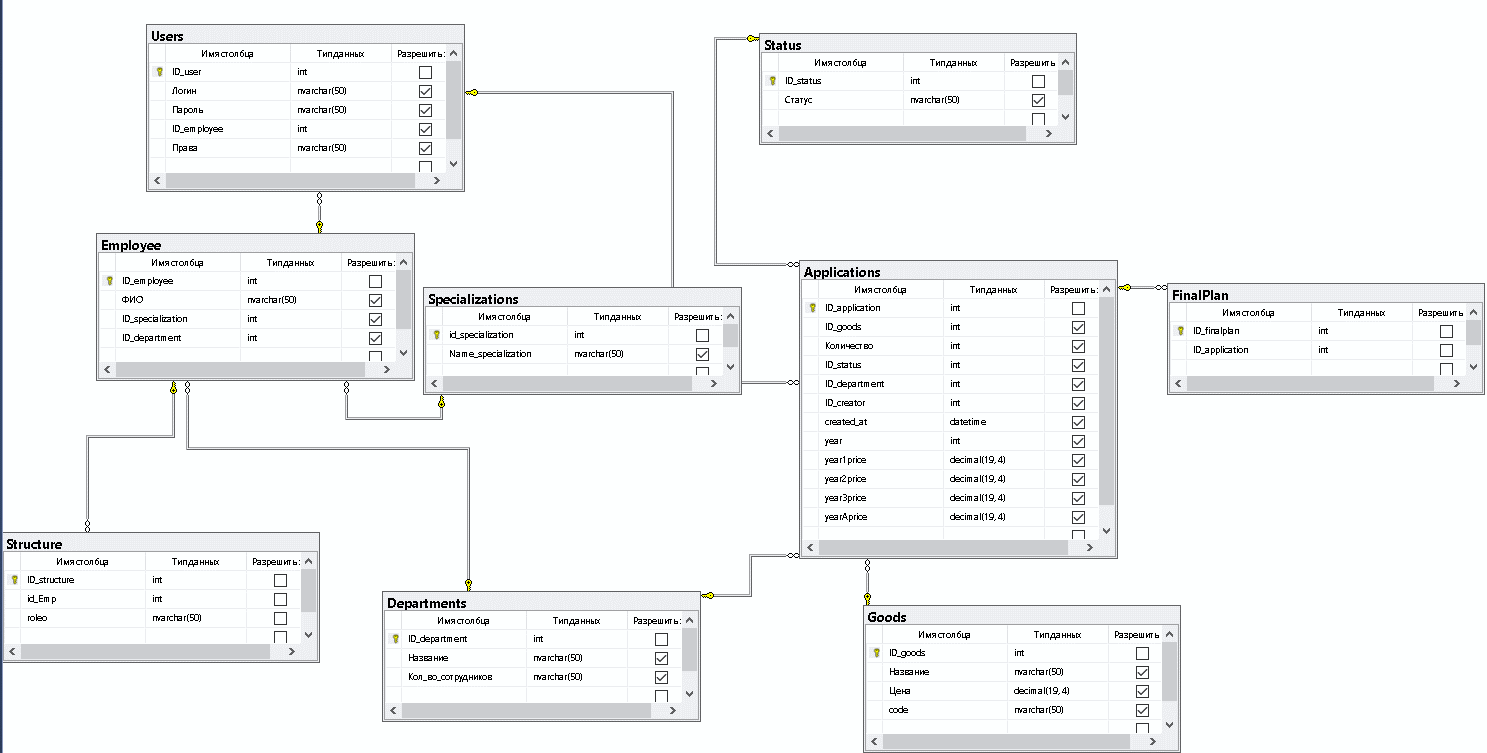


Рисунок 2.7 – Физическая модель данных

2.2.2 Разработка 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 |
| Users | Содержит данные для авторизации пользователя приложения, а также его роль доступа | int, nvarchar(50) | - | Employee |
| Employee | Содержит личную информацию о пользователях приложения | int, nvarchar(50) | Users | Departments |

Окончание таблицы 2.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Departments | Содержит информацию об отделах | int, nvarchar(50) | Employee, Applications | - |
| Applications | Содержит информацию о заявках | int | - | Departments, Goods, Status |
| Goods | Содержит информацию о товарах | int, nvarchar(50), money | Applications | - |
| Status | Содержит информацию о статусах заявок | int, nvarchar(50) | Applications | - |
| Structure | Содержит информацию о структуре | int, nvarchar(50) | - | Users |

2.3 Проектирование клиентской части приложения

2.3.1 Разработка 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].

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 2.7.

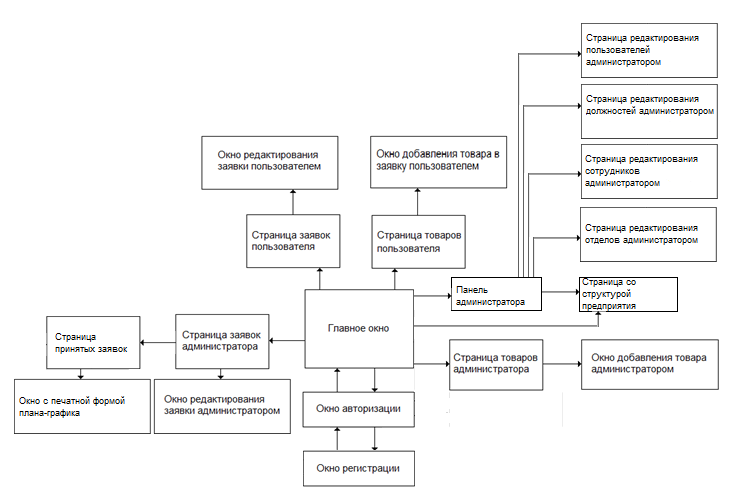


Рисунок 2.7 – Модульная схема клиентской части приложения

В составе модульной схемы присутствуют следующие элементы:

* Главное окно, для выбора списка товаров или заявок;
* Страница товаров администратора, для добавления и удаления;
* Страница товаров пользователя, для добавления в заявку;
* Страница заявок администратора, для редактирования статуса заявок пользователей;
* Страница принятых заявок, для подготвки к печати;
* Окно с печатной формой плана-графика, для просмотра итогового внешнего вида;
* Страница заявок пользователя, для редактирования собственных заявок;
* Панель администратора, для редактирования списков;
* Страница редактирования отделов, для редактирования списка отделов;
* Страница редактирования должностей, для редактирования списка должностей;
* Страница редактирования сотрудников, для редактирования списка сотрудников;
* Страница редактирования пользователей, для редактирования списка пользователей;
* Страница со структурой предприятия;
* Окно авторизации, для авторизации пользователей;
* Окно регистрации, для регистрации пользователя;
* Окно добавления товара администратором, для добавления нового возможного к заказу товара;
* Окно добавления товара в заявку пользователем, для изменения количетсва выбранного товара пользователем;
* Окно редактирования заявки адмнистратором, для изменения статуса заявки пользователя;
* Окно редактирования заявки пользователем, для изменения параметров заявки пользователя.

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 (Graphical User Interface или GUI).** Самый популярный тип UI. Представляет собой окошко с различными элементами управления. Пользователи взаимодействуют с ними с помощью клавиатуры, мыши и голосовых команд: жмут на кнопки, тыкают мышкой, смахивают пальцем.

**XAML (Extensible Application Markup Language — расширяемый язык разметки приложений)** 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.

Главное окно, содержащее в себе страницу с ссылками для перехода в страницы справочников, отвечающее за отображение интерфейса для взаимодействия пользователя с выбором справочника, а также две навигационные сслыки на другие окна. Ссылки меняются в зависимости от уровня доступа пользователя и его идентификации.

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

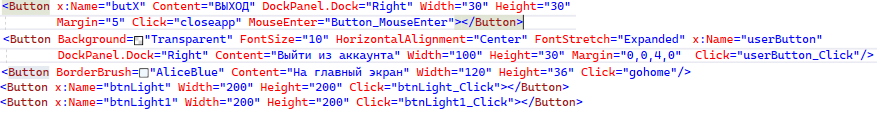


Рисунок 2.8 – Разметка ссылок

Пример разметки окна представлен на рисунке 2.9.



Рисунок 2.9 – Пример разметки окна

Окно авторизации содержит в себе 2 текстовых поля (логин, пароль) для ввода данных и 3 кнопки. Кнопка входа, для подтверждения введенных данных и входа в аккаунт, кнопки перехода на главное окно и окно регистрации.

Разметка текстовых полей представлена на рисунке 2.10.



Рисунок 2.10 – Пример разметки листа

Окно регистрации содержит 5 текстовых поля (имени, пароля, уникального номера, показа и ввода капчи), а также 4 кнопки. Кнопка, отвечающая за скрытие и показа пароля, кнопка подтверждения введенных, кнопка возвращение на окно авторизации и кнопка отображения капчи.

Разметка капчи представлена на рисунке 2.11.

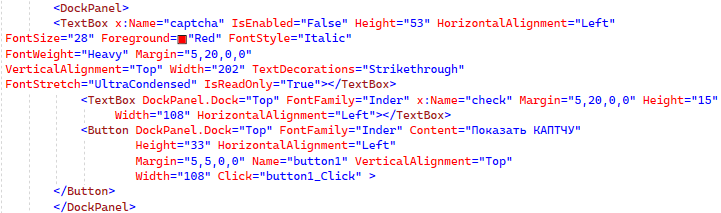


Рисунок 2.11 – Разметка капчи

Разметка DataGrid для товаров, содержащая таблицу с списком товаром, содержащую информацию о названии товара и его цене. В зависимости от того, кто вошел в приложение отображается разный набор управляющих кнопок. У пользователя есть возможность только добавить выбранный товар в заявку. У администратора есть возможность удалять товары из списка, а также добавлять новые позиции. Разметка DataGrid для товаров, одинакова для пользователя и администратора, поэтому только один пример разметки DataGrid для товаров представлен на рисунке 2.12.

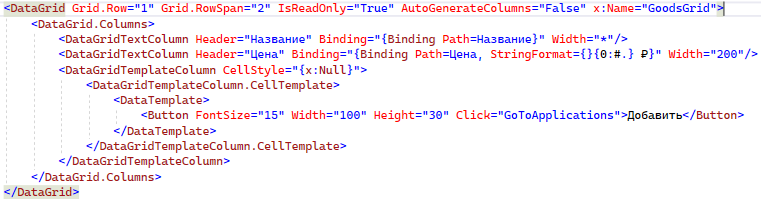


Рисунок 2.12 – Разметка DataGrid товаров

Окно добавления нового товара состоит из двух текстовых полей для ввода данных о товаре (наименовение и цена) и четырех кнопок, две из которых увеличивают и уменьшают цену товара на определенное количество, а также кнопок для подтверждения добавления товара и отмены.

Разметка добавления товара представлена на рисунке 2.13.



Рисунок 2.13– Разметка добавления товара

Окно добавления товара в заявку пользователем состоит из трех текстовых полей, два из которых заблокированы и служат только для вывода информации и одного для указания количества, а также двух кнопок, отвечающих за добавление товара и отмену операции. Окно редактирования заявки пользователем схожа с окном добавления.

Разметка добавления товара в заявку пользователем представлена на рисунке 2.14



Рисунок 2.14 – Разметка добавления товара в заявку пользователем

Окно заявок содержит в себе таблицу DataGrid, которая показывет информацию о названии товара, его количества, статуса заявки и отдела, из которого пришла заявка. В зависмости от того, кто вошел в приложение отображается разная информация. У пользователя есть возможность количество товара в его заявке, а также удалить саму заявку. Администратор может редактировать статус заявки пользователя. Разметка DataGrid для заявок, одинакова для пользователя и администратора, поэтому только один пример разметки DataGrid для заявок представлен на рисунке 2.15.

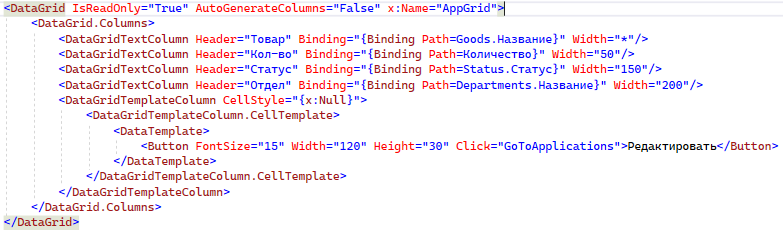


Рисунок 2.15 – Разметка DataGrid для заявок

Окно редактирования заявки состоит из чертырех текстовых полей, которые служат для вывода информации о названии товара, количества, статуса заявки и отдела заявителя. При работе пользователя с окном ему доступно только поле количество для редактирования, администратор может изменять статус заявки пользователя. Окно также содержить четыре кнопки, две из которых изменяют количество товара при работе пользователя, а также две кнопки для подтверждения изменений или отмены их.

Разметка окна редактирования заявки представлена на рисунке 2.16



Рисунок 2.16 – Разметка окна редактирования заявки

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.

Возьмем для примера страницу с заявками, где участвует DataGrid. DataGrid страницы с заявками представлен на рисунке 2.17.

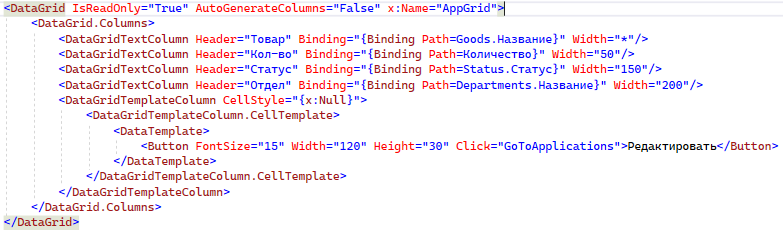


Рисунок 2.17 – DataGrid страницы с заявками

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 функциональной задачей приложения является создание плана закупок на основе рассмотренных заявок. BPMN диаграмма основного бизнесс процесса представлена на рисунке 2.19.

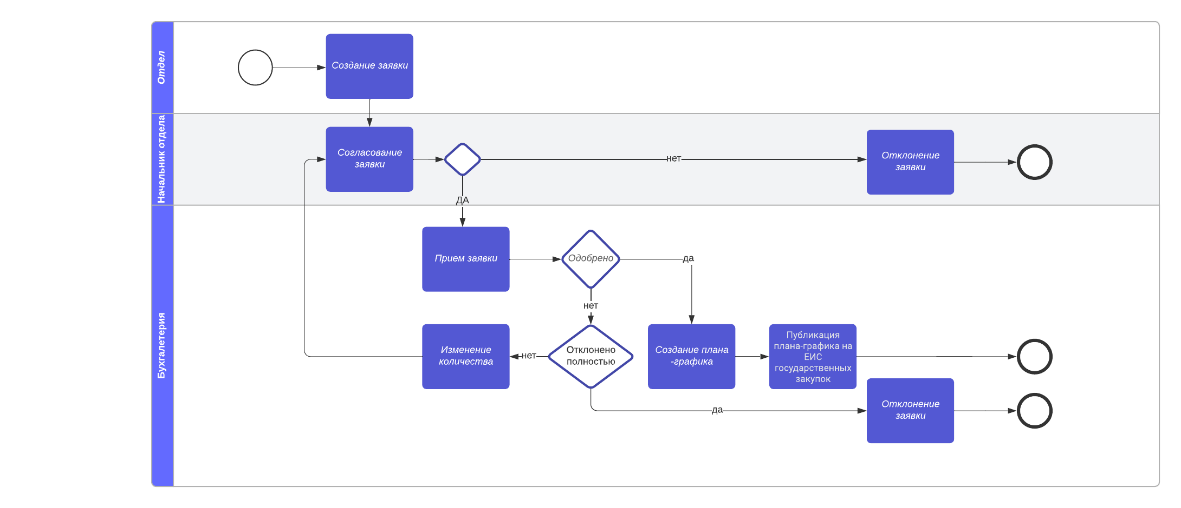


Рисунок 2.19 – BPMN – диаграмма основного бизнесс процесса

Метод AddAPP предназначен для добавления новых записей в базу данных о новой заявке на товар. В качестве входных данных поступает количество товара. У метода не выходных данных. Блок-схема метода AddAPP представлена на рисунке 2.20.

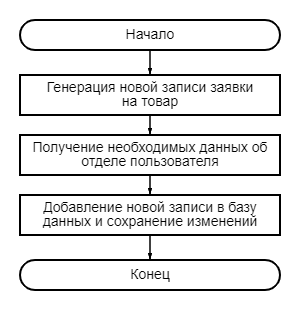
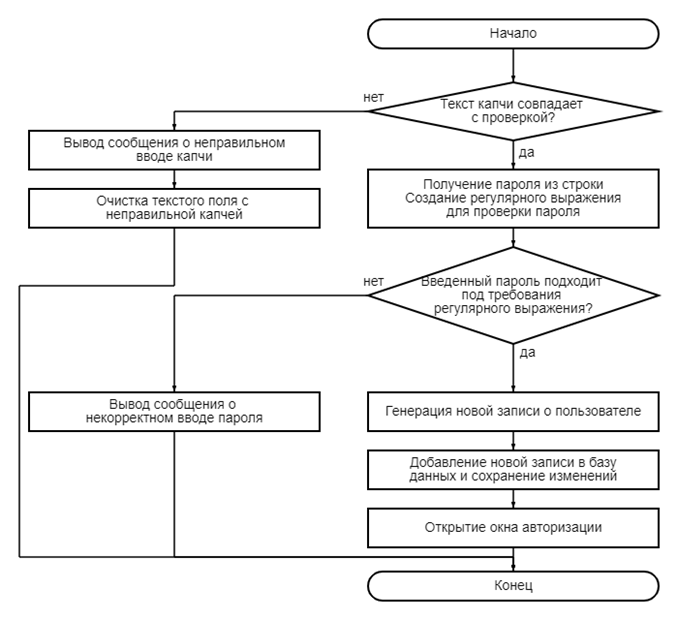


Рисунок 2.20 – Блок-схема метода AddAPP

Метод DoReg предназначен для регистрации нового пользователя в базе данных. Метод проверяет полученные данные на соответствие нормативам и требует от пользователя правильного прохождения капчи. Блок-схема метода DoReg представлена на рисунке 2.21.

 Рисунок 2.21 – Блок-схема метода DoReg

На основании разработанных блок-схем был написан программный код, приведенный в Приложении А к пояснительной записке.

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.

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. Также у пользователя есть возможность вернуться на главный экран. Окно авторизации представлено на рисунке 2.22.

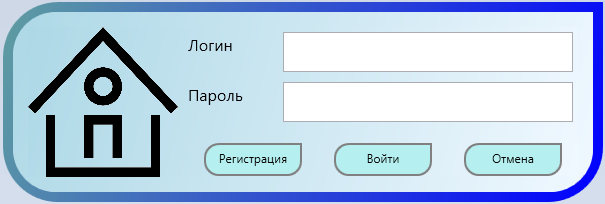


Рисунок 2.22 – Окно авторизации

Для регистрации пользователю необходимо ввести имя и уникальный номер, придумать пароль, а также пройти капчу. Все поля обладают своей валидацией и в случае некорректного ввода данных, появится окно с предупреждением. При желании пользователь может вернуться на главный экран или окно авторизации. Окно регистрации представлено на рисунке 2.23.

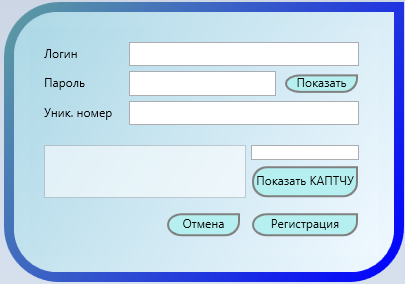


Рисунок 2.23 – Окно регистрации

3 Технологическая часть

3.1 Тестирование и отладка приложения

Отладка — этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки, информация из работы. 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.

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. Тест кейс для функций представлен в таблице 3.1.

Таблица 3.1 – Тест-кейс для методов

|  |  |  |
| --- | --- | --- |
| Имя метода | Управляющее воздействие | Результат воздействия |
| AuthorizProc | Нажать кнопку войти в окне авторизации | Отображение окна предупреждения если логин или пароль были введены неправильно, иначе происходит авторизация |
| DoReg | Вызывается при нажатии кнопки «Регистрация» в окне регистрации | Отображение окна предупреждения если пароль не соответсвует требованиям и/или капча введена некорректно. Если все данные корректны происходит создание нового пользователя в системе |
| CreateKAPCHA | Вызывается при нажатии кнопки «Показать КАПЧУ» в окне регистрации | Отображение проверочной капчи в специальном окне |

Продолжение таблицы 3.1

|  |  |  |
| --- | --- | --- |
| Имя метода | Управляющее воздействие | Результат воздействия |
| navigation | Вызывается при сборке класса MainWindow | Вывод сообщения об ошибке в случае неудачного подключения к базе данных |
| GetName | Вызывается при сборке класса MainWindow | Вывод имени текущего пользователя и его роли в специальное окно |
| GoHome | Вызывается при нажатии кнопки «На главный экран» | Открытие главного экрана приложения |
| sizeCombo\_Selection-  Changed | Вызывается при изменении параметров комбинированного списка «Масштаб» | Изменение размеров приложения |
| init | Вызывается при сборке класса «MainFrame» | Изменение надписей под кнопками перехода в справочники в зависимости от роли текущего пользователя |
| redraw | Вызывается при изменении размеров текущего окна | Изменение размеров страницы адаптивно к размеру основного окна, а также размеров кнопок и шрифта |
| DeleteGood | Вызывается при нажатии кнопки «Удалить» напротив товара в справочнике товаров администратора | Удаление объекта из списка. Если объект учавствует в заявке будет получено предупреждение без удаления |
| RedactUserApp | Вызывается при нажатии кнопки «Редактировать» напротив заявки в справочнике заявок пользователя | Открытие нового окна для редактирования. Если статус заявки будет «Приянто» - вывод предупреждение без открытия окна |
| delete | Вызывается при нажатии кнопки «Удалить» напротив заявки в справочнике заявок пользователя | Удаление заявки из справочника. Если статус заявки будет «Приянто» - вывод предупреждение без открытия окна. Если автором заявки будет не авторизованный на данный момент пользователь, то удаления не произвойдет и откроется окно с предупреждением |

Окончание таблицы 3.1

|  |  |  |
| --- | --- | --- |
| Имя метода | Управляющее воздействие | Результат воздействия |
| initAppRed | Вызывается при открытии окна редактирования заявки | Заполнение текстовых блоков соответсвующей информацией из связанных таблиц |
| AppRedactionClick | Вызывается при нажатии кнопки «Подтвердить» в окне редактирования заявки | Изменение данных о количестве товара в справочники или вывод сообщения об ошибке если количетсов было введено не цифрами |
| initCastApp | Вызывается при сборке окна создания заявки | Вывод всех данных по текущей заявке и отделу текущего пользователя |
| AddApp | Вызывается при нажатии кнопки «Подтвердить» в окне создания новой заявки | Добавление новой записи в список заявок пользователя, установка статуса «Не рассмотрено» |
| decrease | Вызывается при нажатии на кнопку «∧» | Уменьшение числового значения соседнего текстого поля |
| increase | Вызывается при нажатии на кнопку «∨» | Увеличение числового значения соседнего текстого поля |
| UpdateGrid | Вызывается при добавлении и удалении записей | Обновление таблицы новыми данными |
| UpdatePrintGrid | Вызывается при сборке класса PrintPreview | Заполнение печатной формы плана-графика информацией из базы данных |
| loadStructure | Вызывается при сборке класса Structure | Открытие окна структуры предприятия. Если структура заполнена, будет возможность удалить любого сотрудника из нее. Если структура не заполнена, будет возможность назначить сотрудника на должность |
| DlgLoad | Вызывается при помощи кнопок добавления, копирования и изменения записей на странице, и кнопок добавления и отмены в окне редактирования записи | Отображение и скрытие окна редактирования записей |

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. В СНГ MSSQL чаще всего применяется для собственных разработок прикладного ПО и для 1С.

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

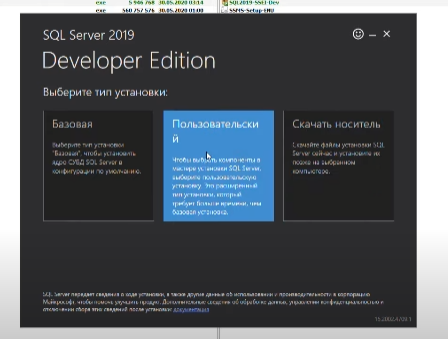


Рисунок 3.1 – «Пользовательский» тип установки

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.

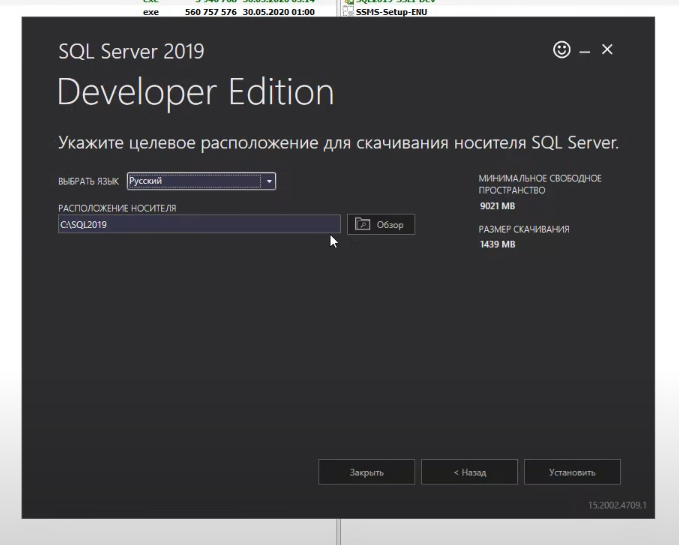


Рисунок 3.2 – Изображение окна

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

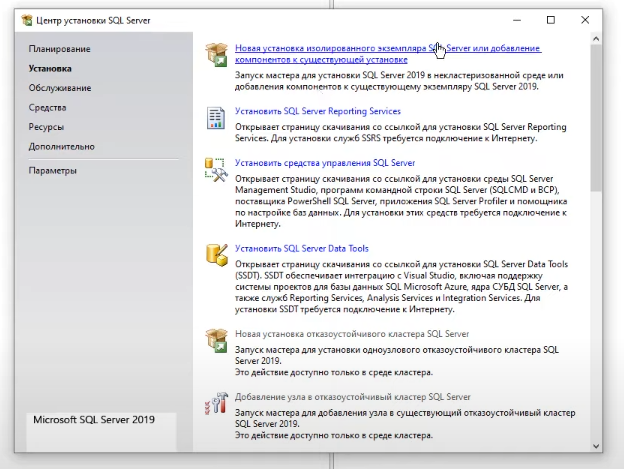


Рисунок 3.3 – Новая установке изолированного экземпляра SQL Server

После установки произойдет обновление продукта. Рисунок 3.4.

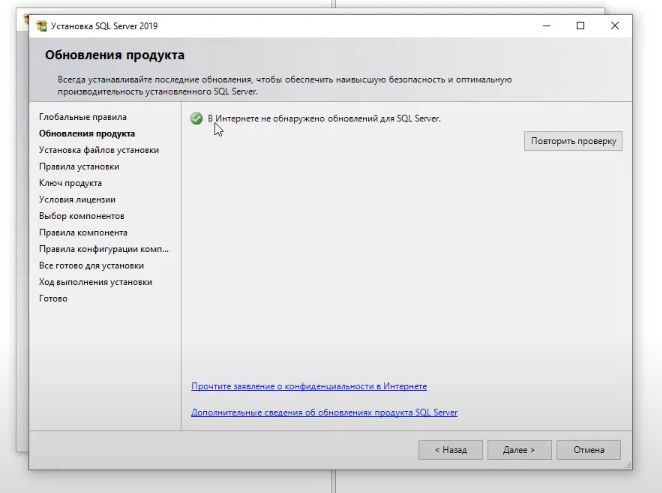


Рисунок 3.4 – Обновление продукта

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

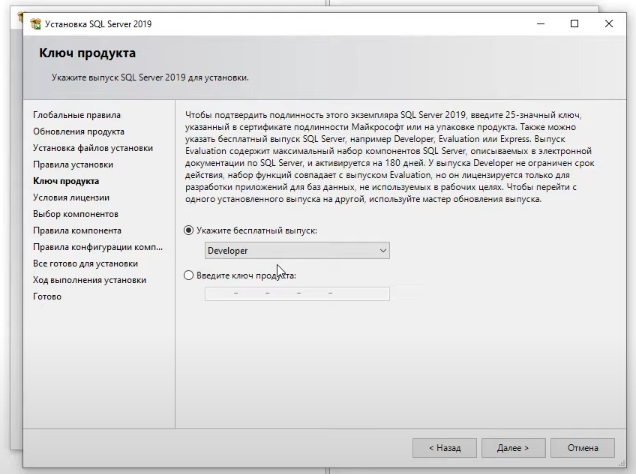


Рисунок 3.5 – Версия «Developer»

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

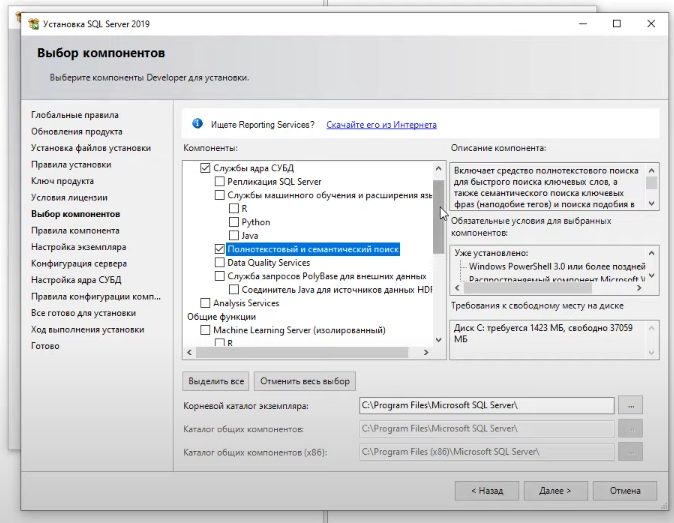


Рисунок 3.6 – Выбор компонентов

В разделе правила компонента всё оставить по умолчанию и перейти 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 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.

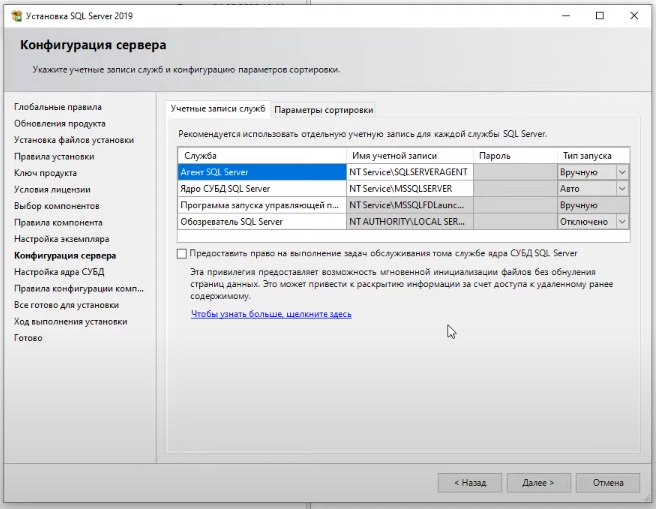


Рисунок 3.7 – Конфигурация сервера

В разделе «Настройка ядра СУБД» 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 предложат создать учетную запись SQL Server. Оставить режим аутентификации Windows и выбрать пользователя как показано на рисунке 3.8.

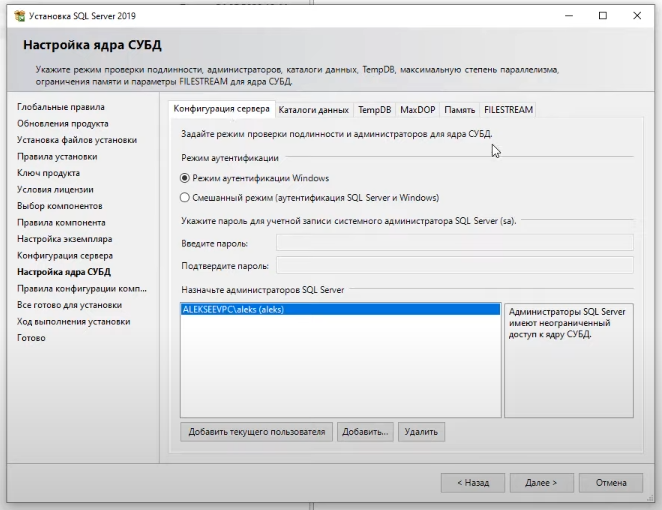


Рисунок 3.8 – Настройка ядра СУБД

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

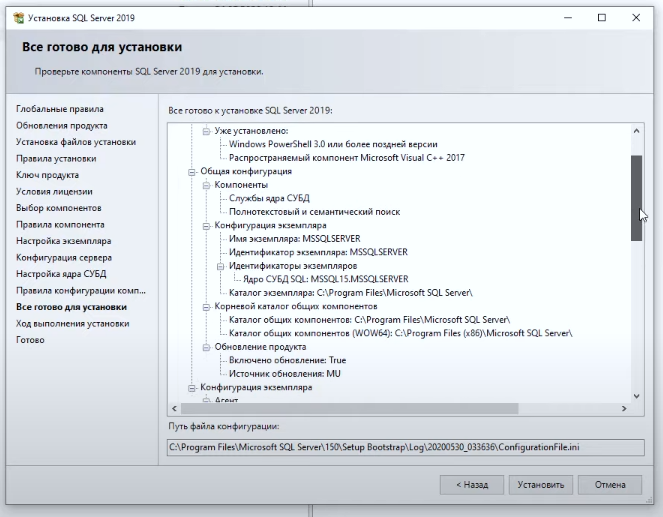


Рисунок 3.9 – Все готово для установки

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

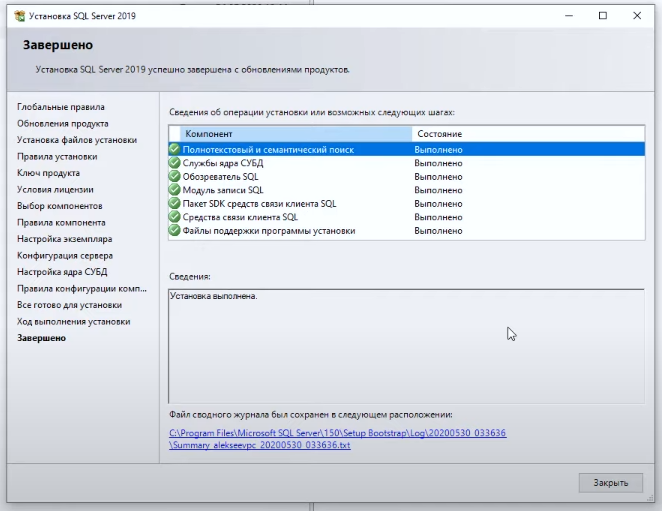


Рисунок 3.10 – Финальный экран

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

Необходимо снова зайти в центр установки SQL Server и наживать "Установить средства управления SQL Server", как показано на рисунке 3.11.

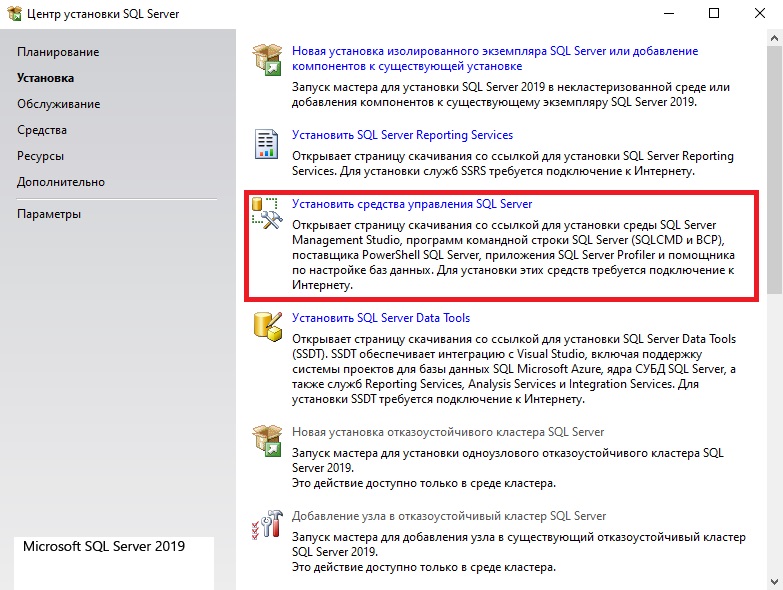


Рисунок 3.11 – Установка SQL Server Management Studio

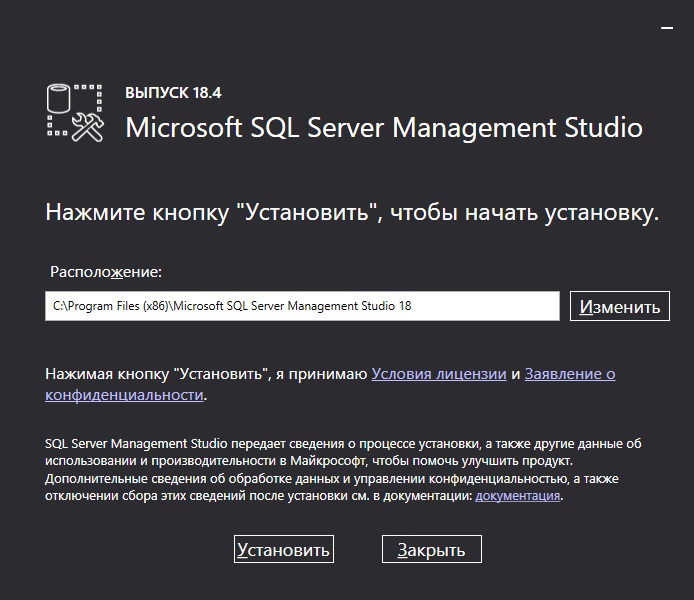
При нажатии откроется [сайт Microsoft](https://docs.microsoft.com/ru-ru/sql/ssms/download-sql-server-management-studio-ssms?redirectedfrom=MSDN&view=sql-server-ver15) и нужно будет скачать SSMS. Надо нажать кнопку "Установить", как показано на рисунке 3.12.

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;
* Employee – Сотрудники;
* Departments – Отделы;
* Applications – Заявки;
* Goods – Товары;
* Status – Статусы заявок;
* Structure – Структура предприятия.

3.3 Инструкция по эксплуатации приложения

Приложение предназначено для удобного заказа товаров внутри фирмы. Пользователь может установить приложение, после чего пройти регистрацию и в любой момент заказать товар необходимый его отделу. После этого ему останется дождаться одобрения заявки администратором.

Чтобы начать работать с приложением, необходимо его запустить, а после этого авторизоваться или зарегистрироваться для получения доступа к его функциям. Для добавления новых товаров, редактирования статусов заявок, составления структуры организации или подготовки печатной формы плана закупок, пользователь должен войти со статусом администратора. Окно с авторизованным пользователем с правами администратора представлено на рисунке 3.14.

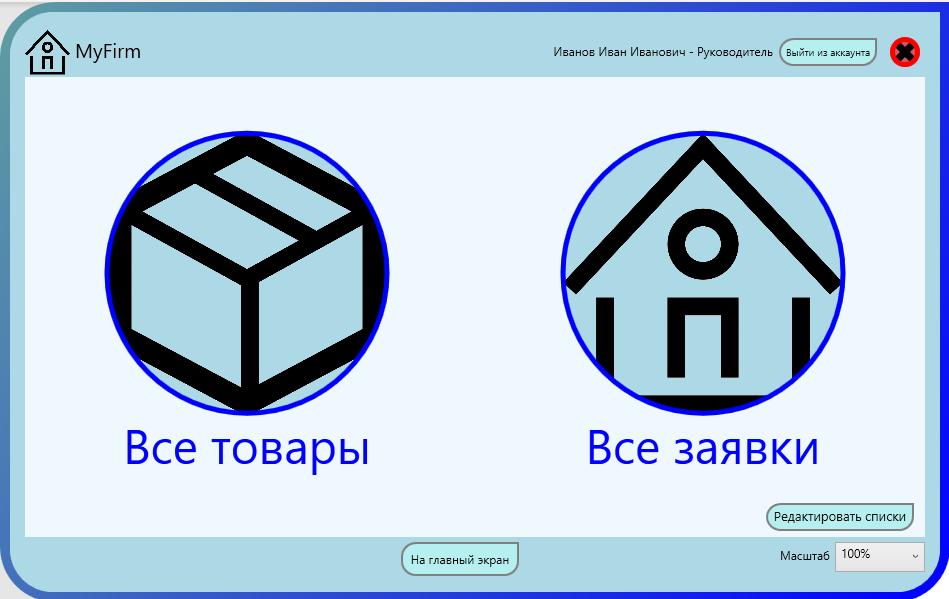


Рисунок 3.14 – Окно с авторизованным пользователем с правами администратора

При нажатии на кнопку «Все товары», пользователь попадет на окно справочника товаров. Окно справочника товаров администратора представлено на рисунке 3.15.

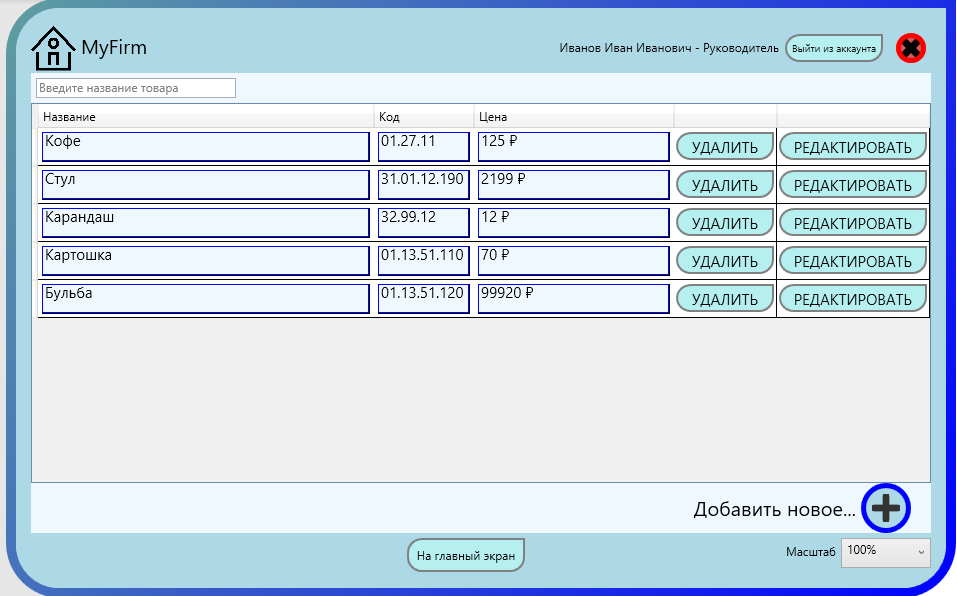


Рисунок 3.15 – Окно справочника товаров администратора

При нажатии на кнопку «+» откроется новое окно, которое позволит администратору добавить новую запись в базу данных. При нажатии кнопки откроется окно как на рисунке 3.16.

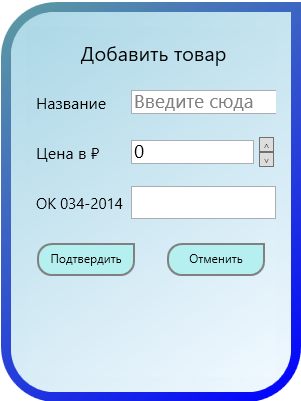


Рисунок 3.16 – Окно добавления нового товара

При нажатии на кнопку «Все заявки», администратор попадет на окно справочника всех заявок. Окно со справочником всех заявок представлено в окне 3.17.

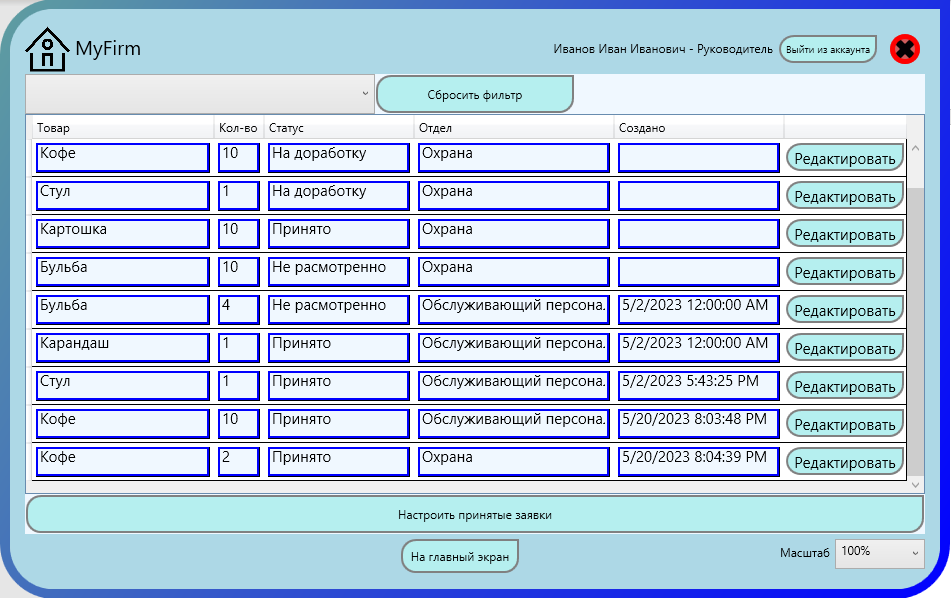


Рисунок 3.17 – Окно со справочником заявок

При нажатии на кнопку «Редактировать» откроется новое окно, которое позволит администратору редактировать заявку пользователя. Окно редактирования заявки представлено на рисунке 3.18.

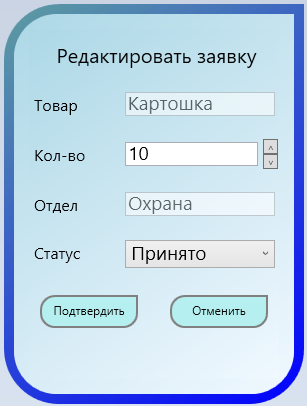


Рисунок 3.18 – Окно редактирования заявки пользователя

При нажатии на кнопку «Настроить принятые заявки», откроется окно с принятыми заявками. Окно настройки принятых заявок расположено на рисунке 3.19.

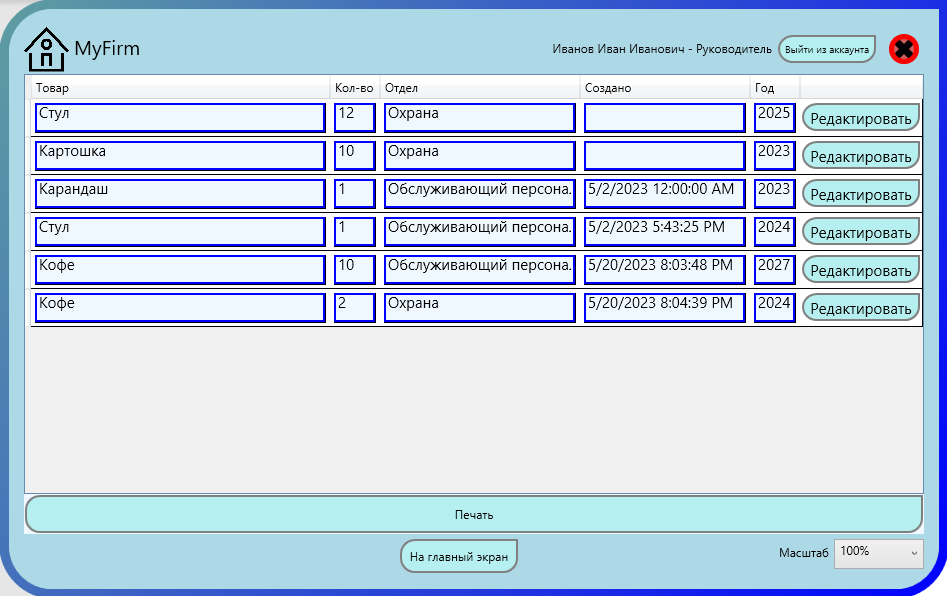


Рисунок 3.19– Окно настройки приянтых заявок

При нажатии на кнопку «Печать» откроется окно, в которм будет представлена заполненная печатная форма плана-графика. Окно с заполненной печатной формой представлено на рисунке 3.20.

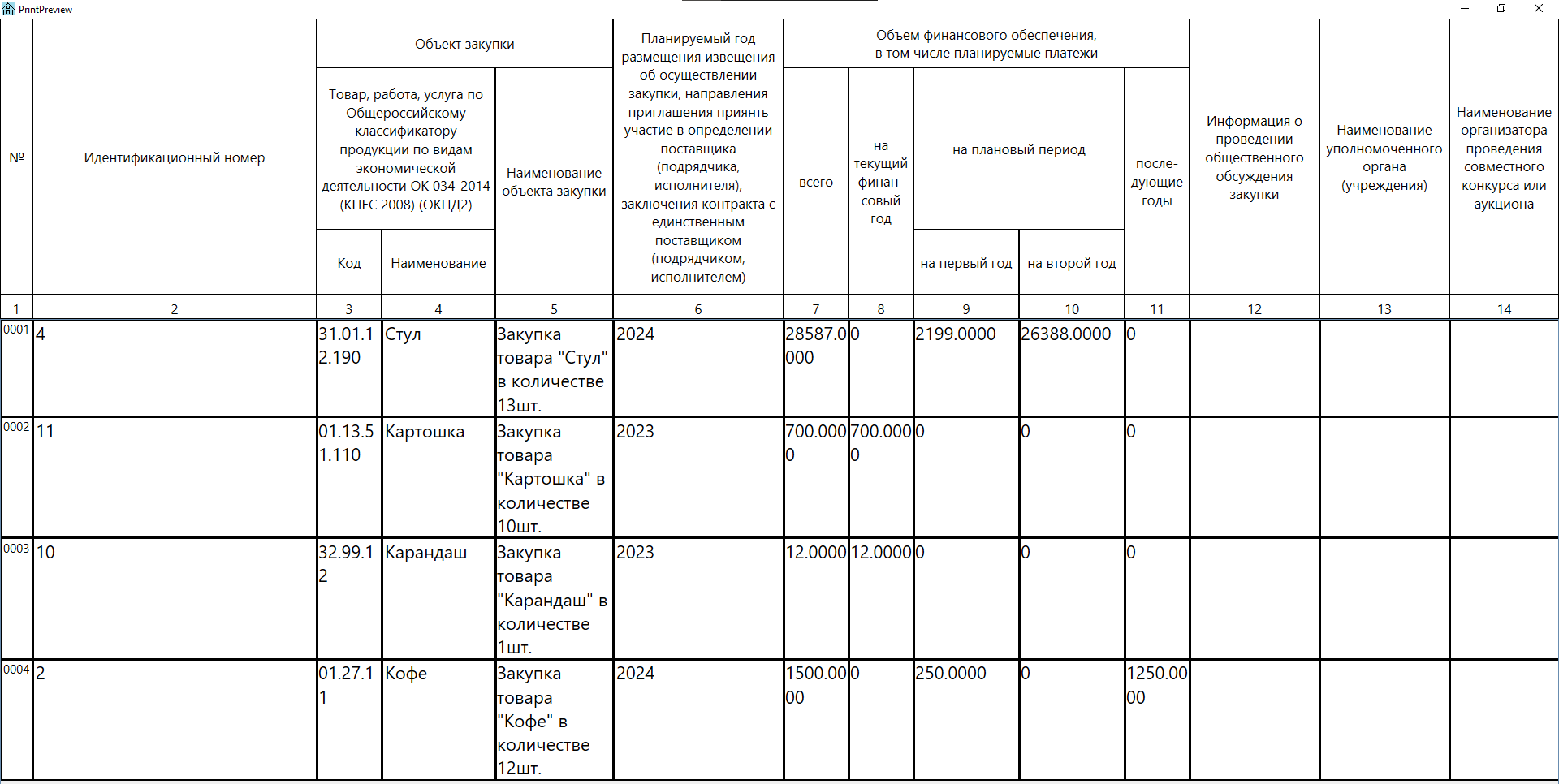


Рисунок 3.20 – Окно с заполненной печатной формой

При нажатии на кнопку «Редактировать списки», которая расположена на главной странице, администратор попадет на окно для редактирования всех справочников, которые задействованы в работе приложения. В этом окне администратор сможет создать или отредактировать структуру организации. Панель администратора представлена на рисунке 3.21.

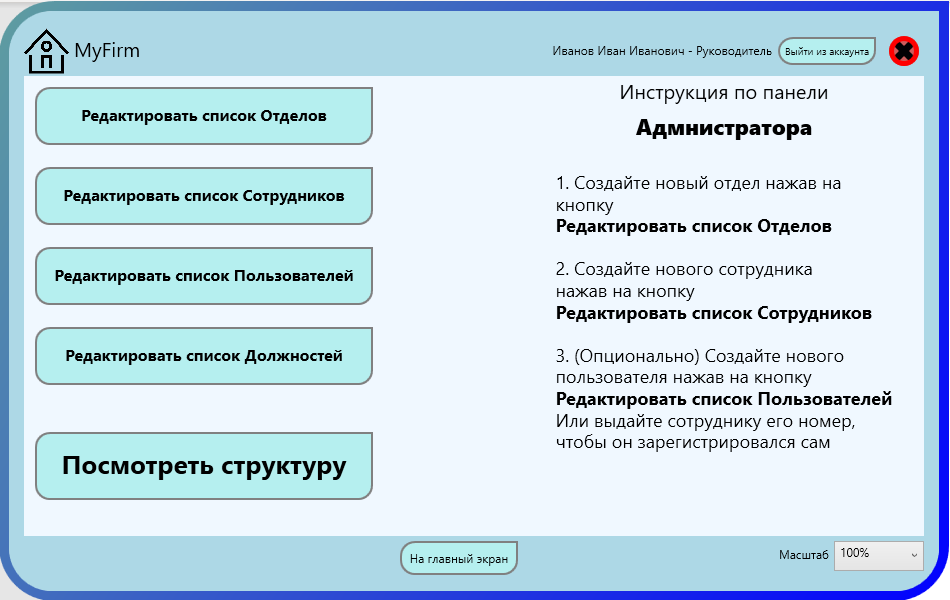


Рисунок 3.21 – Панель администратора

Окно редактирования списков выглядят схожим образом и отличаются только самими списками. Окно редактирования списка представлено на рисунке 3.22.

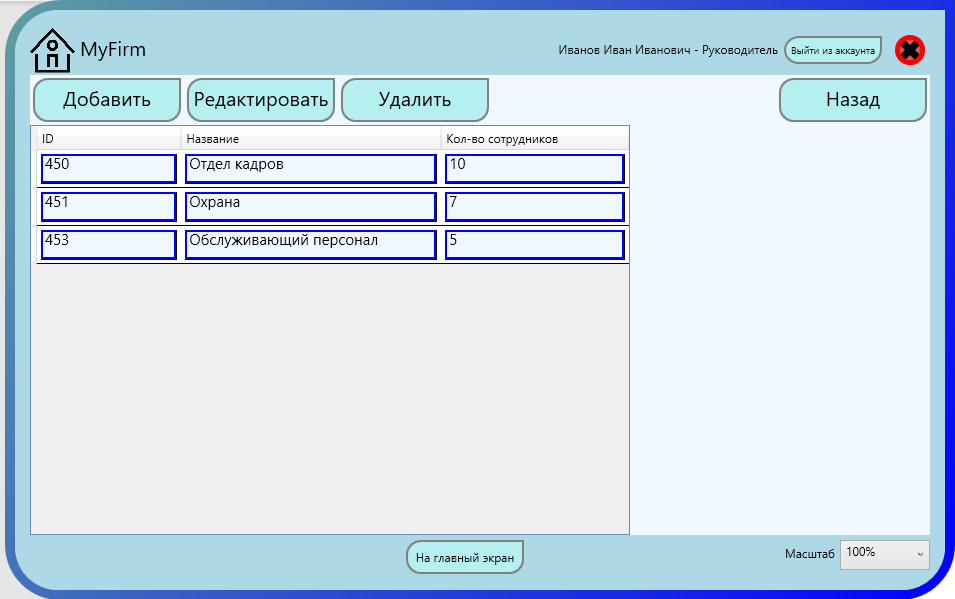


Рисунок 3.22 – Окно редактирования списка

Механизмы добавления, редактирования или удаления идентичны для каждого списка и различаются только полями для заполнения. Окно добавления новой записи представлено на рисунке 3.23

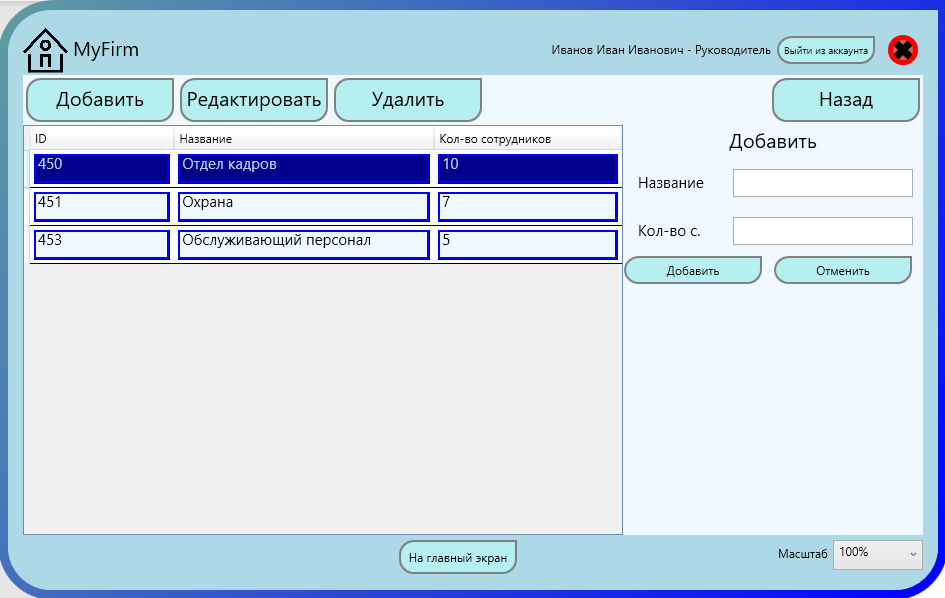


Рисунок 3.23 – Окно добавления новой записи

Окно редактирования записи в списке представлено на рисунке 3.24.

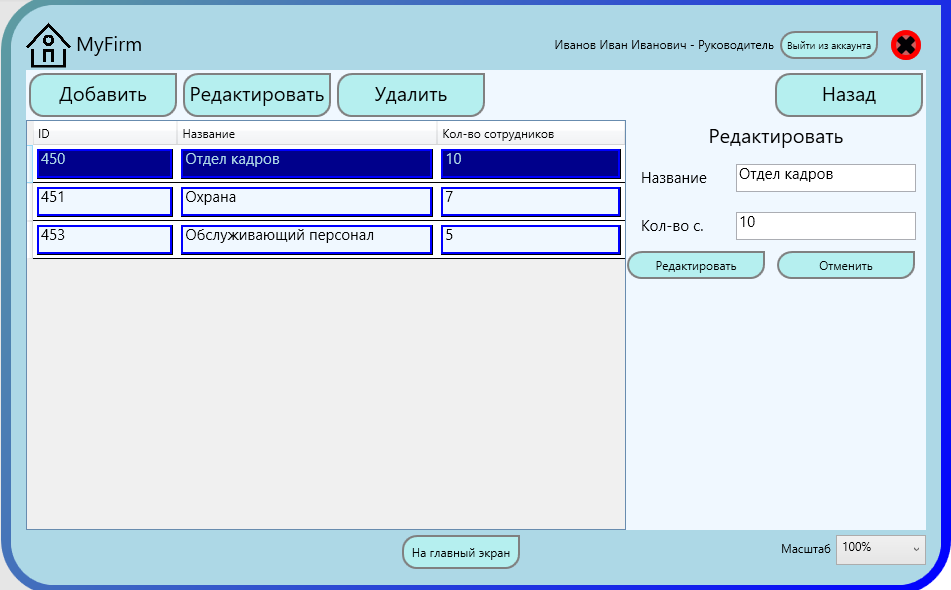


Рисунок 3.24 – Окно редактирования записи

При нажатии на кнопку «Посмотреть структуру» открывается окно, на котором представлена структура организации в незаполненном виде. Окно с незаполненной структурой предоставлено на рисунке 3.25.

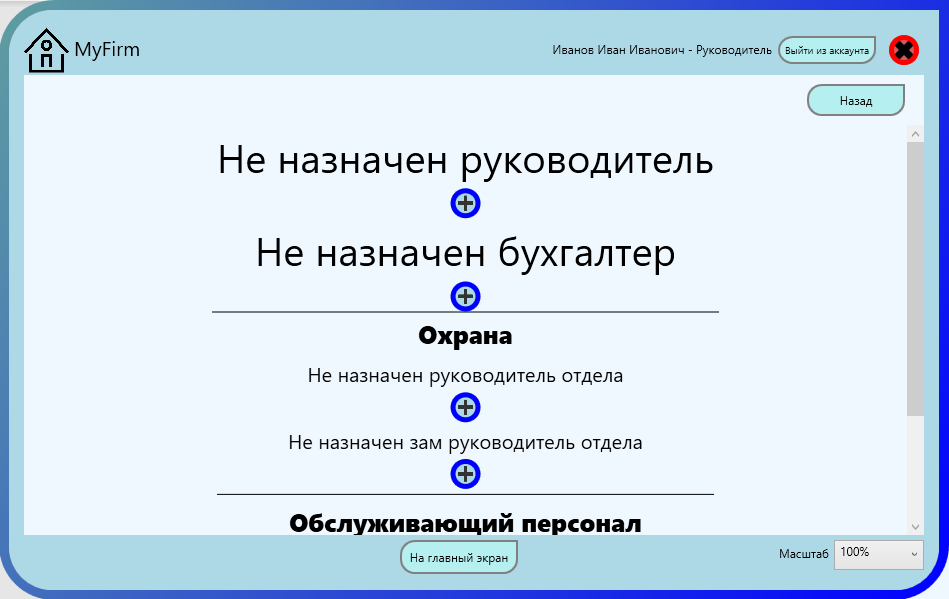


Рисунок 3.25 – Окно с незаполненной структурой

Администратор может заполнить структуру, для этого необходимо нажать на кнопку «+», после чего выбрать сотрудника на должность. Список сотрудников был создан вручную в приложении. Окно выбора сотрудника на должность предоставлено на рисунке 3.26.

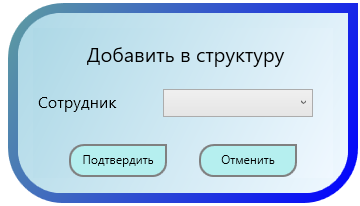


Рисунок 3.26 – Окно с выбором сотрудника на должность

После назначения, в структуре будет отображаться ФИО сотрудника и его должность. Также будет возможность удалить сотрудника с должности, нажатием на кнопку «х». Окно структуры с добавленным сотрудником представлено на рисунке 3.27.

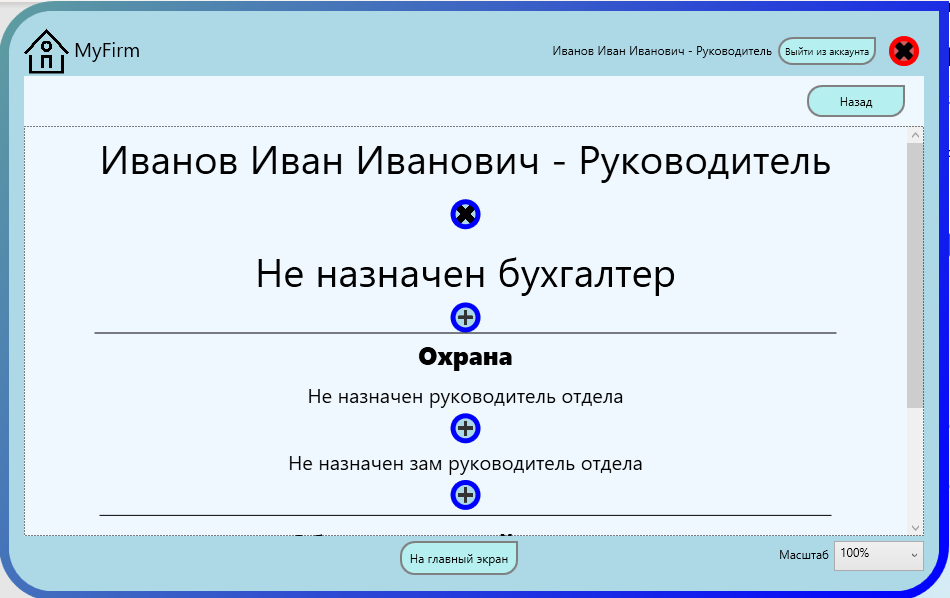


Рисунок 3.27 – Окно структуры с добавленным сотрудником

Для создания заявки или редактирования своих собственных заявок пользователь должен войти со статусом обычного пользователя. Окно с авторизованным пользователем с правами обычного пользователя представлено на рисунке 3.28.

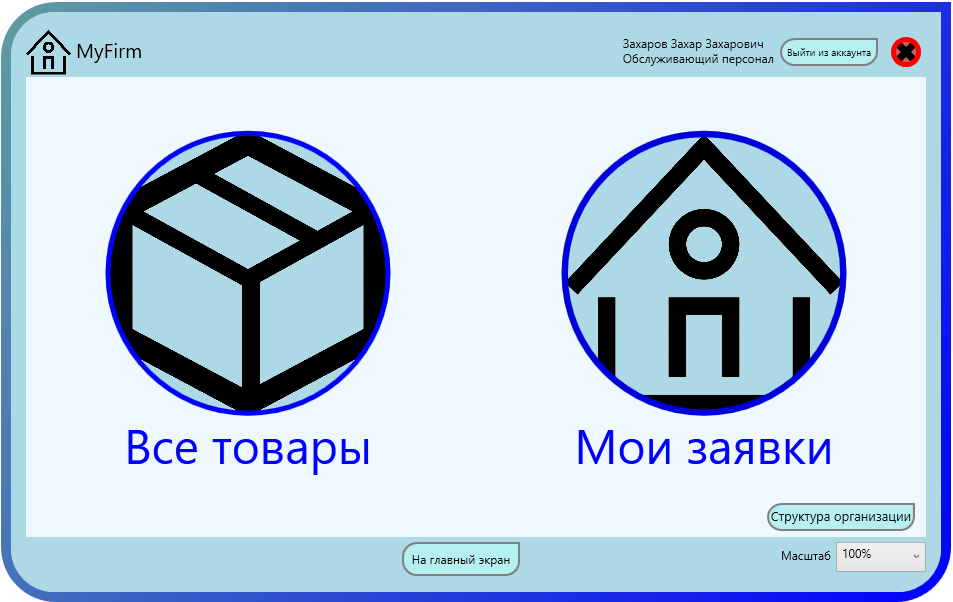


Рисунок 3.28 – Окно с авторизованным пользователем с правами администратора

Окно товаров, доступных пользователю представлено на рисунке 3.29.

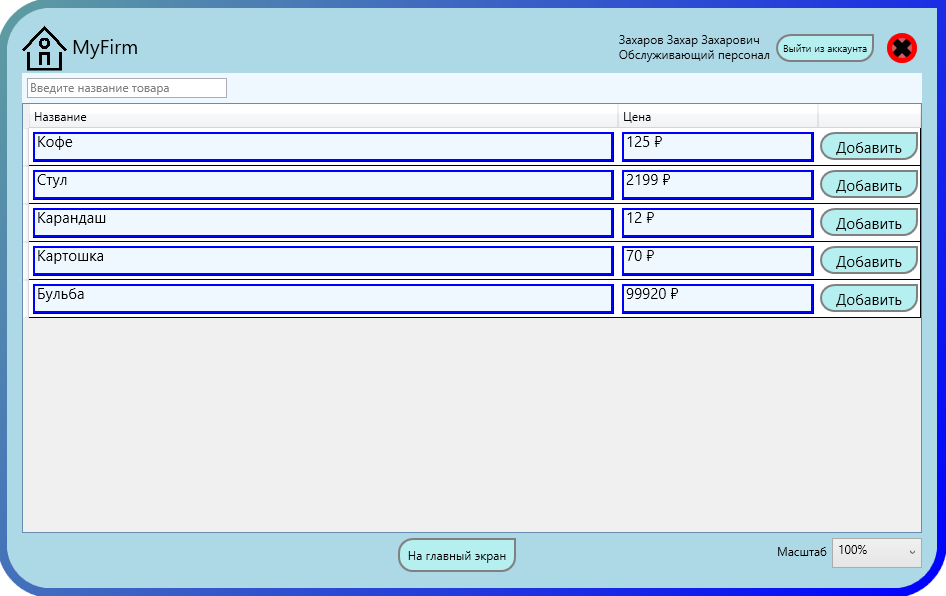


Рисунок 3.29 – Окно товаров пользователя

Окно добавления новой заявки пользователем представлено на рисунке 3.30

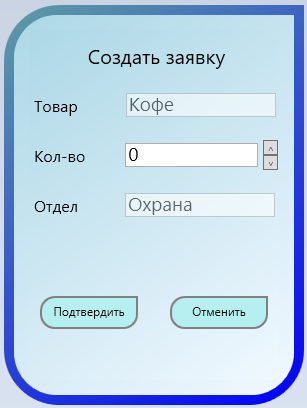


Рисунок 3.30 – Добавление новой заявки пользователем

При нажатии на кнопку «Мои заявки», пользователь попадет на окно справочника собственных заявок. Окно со справочником заявок пользователя представлено на рисунке 3.31.

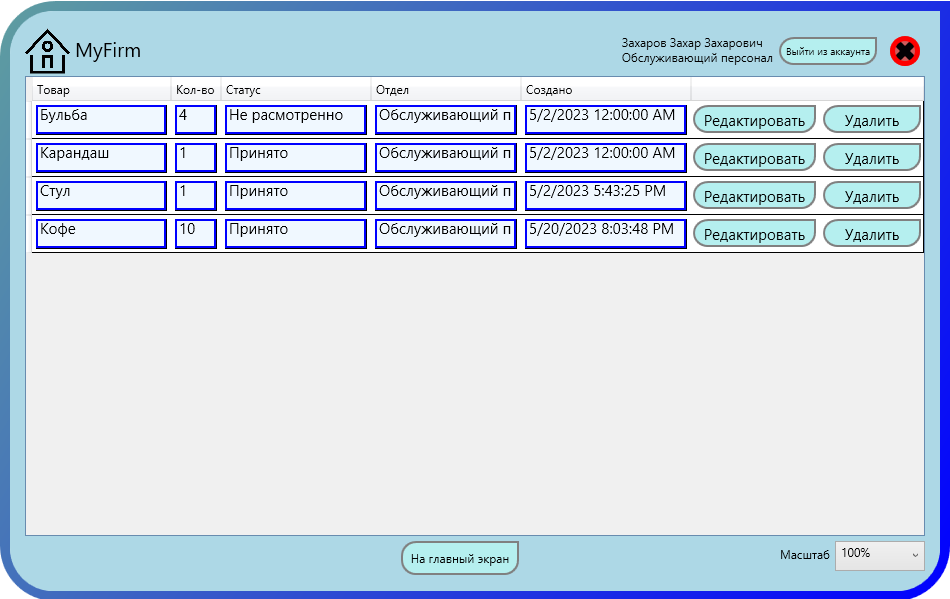


Рисунок 3.31 – Окно со справочником заявок пользователя

При нажатии на кнопку «Редактировать» откроется новое окно, которое позволит пользователю редактировать свою заявку. Окно редактирования пользователем собственной заявки представлено на рисунке 3.32.

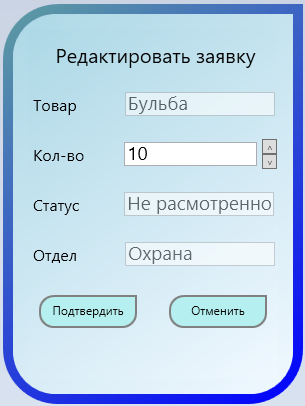


Рисунок 3.32 – Окно редактирования пользователем своей заявки

Пользователь может изменить масштаб приложения выбрав нужное значение в выпадающем списке.

Пример открытия выпадающего списка представлен на рисунке 3.33.

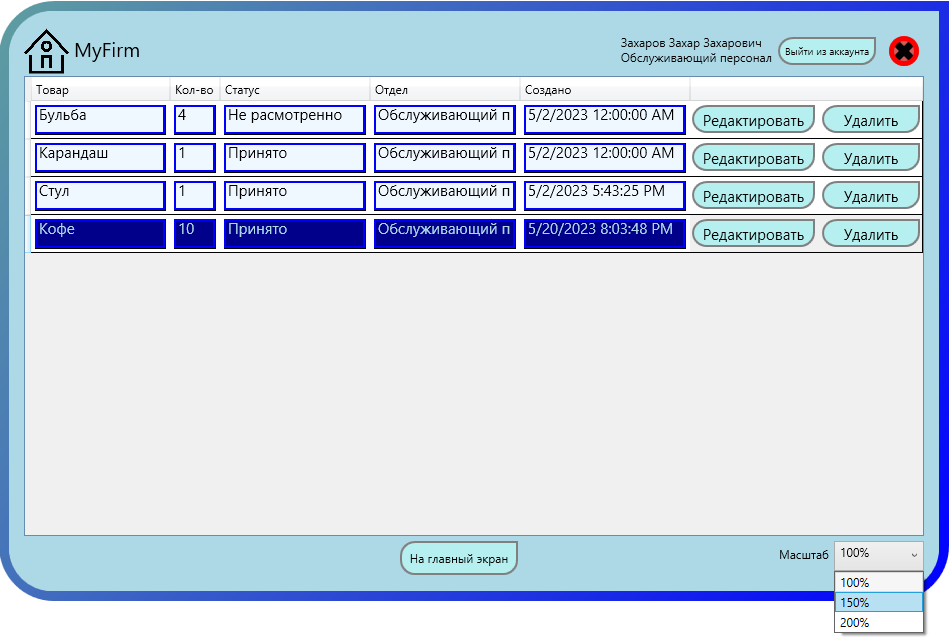


Рисунок 3.33 – Выпадающий список с выбором масштаба

Пример окна приложения с измененным масштабом представлен на рисунке 3.34.

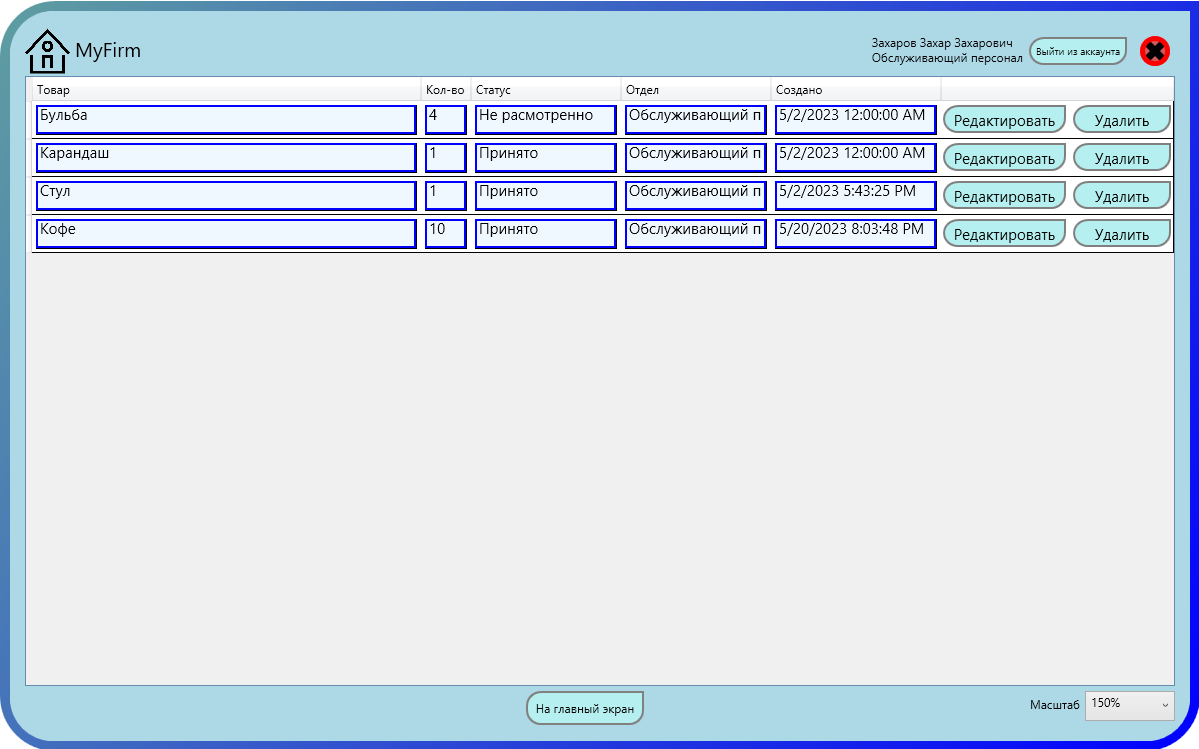


Рисунок 3.34 – Окно с измененным масштабом

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 1 Гб | 999,00 | 1 | 999,00 |
| Итого |  |  | 999,00 |

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 4.2:

, (4.2)

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. | 2,00 | 100 | 200,00 |
| Ватман формата А0, шт. | 165,00 | 3 | 495,00 |
| Картридж для принтера, шт. | 1490,00 | 0,2 | 289,00 |
| Итого |  |  | 984,00 |

Подстановкой указанных выше значений в формулу 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 4,55 руб.

Совокупная потребленная мощность *W* рассчитывается по формуле 4.5

, (4.5)

где Wi – мощность, потрецбляемая 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 | Продолжи-тельность эксплуатации в смену, час. | Количество смен в цикле производства (разработки), шт. | Потребленная мощность, КВт×ч |
| Компьютер | 0,33 | 5 | 45 | 74,25 |
| Принтер | 0,34 | 0.5 | 2 | 0,34 |
| Итого |  |  |  | 74,59 |

Подстановкой значений из таблицы 4.3 в формулу 4.5 получено значение потребленной мощности:



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 – 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 [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 4.6:

, (4.6)

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 | 5 740,00 | 5 | 1148,00 | 45 | 141,53 |
| Офисное кресло | 5 390,00 | 4 | 1347,5 | 45 | 166,13 |
| Компьютер | 61590,00 | 7 | 8798,57 | 45 | 1084,75 |
| Принтер | 2690,00 | 5 | 538,00 | 2 | 2,94 |

Окончание таблицы 4.4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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. |
| Windows 10 Professional | 10 360,00 | 10 | 1036,00 | 45 | 87,99 |
| Подписка Visual Studio Professional | - | - | 54 000,00 | 45 | 4586,30 |
| Microsoft Office professional plus 2019 | 6 700,00 | 5 | 1340,00 | 7 | 25,69 |
| Итого |  |  |  |  | 6095,33 |

Подстановкой значений из таблицы 4.4 в формулу 4.6 получено:



Следующим этапом сметного расчета является затраты на оплату труда.

Фонд оплаты труда ЗП, руб., включаемый в затраты по проекту, можно рассчитать по формуле 4.7:

, (4.7)

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:

, (4.8)

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 4.9:

, (4.9)

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 первой категории | 65 000,00 | 5 | 15931,37 |
| Техник-программист | 27 500,00 | 31 | 41789,21 |
| Дизайнер интерфейса | 60 000,00 | 4 | 11764,70 |
| Итого |  |  | 69485,28 |

Подстановкой указанных выше значений в формулу 4.9 получено:



Подстановкой указанных выше значений в формулу 4.8 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 4.10:

, (4.10)

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 4.10 получено:



Подстановкой указанных выше значений в формулу 4.7 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 4.11:

, (4.11)

где ЗП – фонд оплаты труда по проекту, рассчитанный ранее (руб.), составляет 76 433,808 руб.;

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

Ставка страховых взносов ССВ на 2023 год составляет 30,0%.

Подстановкой указанных выше значений в формулу 4.11 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 4.12:

, (4.12)

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

СНР – размер накладных расходов, %.

Размер накладных расходов СНР составляет 10%.

Подстановкой указанных выше значений в формулу 4.12 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 4.13.

*Сб* = *М* + ВМ + ЭР + АМ + *ЗП* + *СВ* +*НР,* (4.13)

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 4.13 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 4.14:

, (4.14)

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 4.14 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 4.15:

, (4.15)

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 4.15 получено:



Конечная стоимость разработки *С*, руб. рассчитывается как сумма себестоимости, прибыли и НДС, для расчёта можно воспользоваться формулой 4.16:

, (4.16)

где Сб – себестоимость разработки, руб.

П – прибыль. руб.;

НДС – сумма налога на добавленную стоимость, руб.;

Подстановкой указанных выше значений в формулу 4.16 получено:



Исходя из вышеперечисленных расчетов следует, что программный продукт является конкурентоспособным, его стоимость составляет 180 063,058.

Данный продукт является востребованным для предприятия, так как это программное решение повысит эффективность работы сотрудников на предприятии, что приведёт к экономии средств и времени. Данная стоимость является небольшой с учётом того, что отсутствуют ограничения на количество машин для установки, оплата совершается единоразово, а дальнейшие улучшения 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 пользователю зарегистрироваться и ознакомться с списком различных товаров. Пользователь может добавить нужный ему товар в заявку. Для пользователей с правами администратора есть доступ к редактированию списка товаров, статусов заявок пользователей, составлению структуры предприятия и формированию печатной формы плана-графика.

Главным достоинством можно выделить простой и прятный глазу интерфейс и простоту в использовании приложения. Все действия выполняются на интуитивно понятном уровне.

Приложение можно использовать для любой фирмы, который требуется приложение составления плана закупок.

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 окна MainWindow

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;

using WorkPlan.Pages;

namespace WorkPlan

{

public partial class MainWindow : Window

{

private Base.wpEntities DataBase;

public Base.Users User { get; set; }

public Base.Employee Employee { get; set; }

int roleNum = 0;

public MainWindow(int Role)

{

InitializeComponent();

roleNum = Role;

navigation(Role);

sizeCombo.Items.Add("100%");

sizeCombo.Items.Add("150%");

sizeCombo.Items.Add("200%");

sizeCombo.SelectedIndex = 0;

}

public void getName(int Role)

{

userButton.Content = "Выйти из аккаунта";

Base.Users User = DataBase.Users.SingleOrDefault(U => U.ID\_user == Role);

int emcode = Convert.ToInt32(User.ID\_employee);

Base.Employee Employee = DataBase.Employee.SingleOrDefault(U => U.ID\_employee == emcode);

Base.Departments dep = DataBase.Departments.SingleOrDefault(u => u.ID\_department == Employee.ID\_department);

if (User.Права == "ADMIN")

{

text.Text = Employee.ФИО + " - " + User.Права;

}

else

{

text.Text = Employee.ФИО + "\n" + dep.Название;

}

}

public void navigation(int role)

{

try

{

DataBase = new Base.wpEntities();

}

catch

{

MessageBox.Show("Не удалось подключиться к базе данных. Проверьте настройки подключения приложения.",

"Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

Close();

}

RootFrame.Navigate(new Pages.MainFrame(role, this.Width,this.Height,this));

getName(role);

}

private void OnKeyDownHandler(object sender, KeyEventArgs e)

{

if (e.Key == Key.Escape)

{ navigation(roleNum); }

}

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

{

SiGlogWind window = new SiGlogWind();

window.Show();

Close();

}

public double widthNow()

{

return this.Width;

}

private void gohome(object sender, RoutedEventArgs e)

{

navigation(roleNum);

}

private void closeapp(object sender, RoutedEventArgs e)

{

if (System.Windows.MessageBox.Show("Выйти из приложения?", "Выход", MessageBoxButton.OKCancel, MessageBoxImage.Warning) == MessageBoxResult.OK)

{

Close();

}

}

private void closeAppText(object sender, MouseButtonEventArgs e)

{

if (System.Windows.MessageBox.Show("Выйти из приложения?", "Выход", MessageBoxButton.OKCancel, MessageBoxImage.Stop) == MessageBoxResult.OK)

{

Close();

}

}

private void sizeCombo\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (sizeCombo.SelectedItem.ToString() == "100%")

{

this.Width = 700;

this.Height = 500;

}

if (sizeCombo.SelectedItem.ToString() == "150%")

{

this.Width = 1050;

this.Height = 750;

}

if (sizeCombo.SelectedItem.ToString() == "200%")

{

this.Width = 1400;

this.Height = 1000;

}

}

private void Window\_MouseDown(object sender, MouseButtonEventArgs e)

{

this.DragMove();

}

}

}

Программный код окна SiGlogWind

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.IO;

using System.Data.SqlClient;

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.Drawing.Imaging;

namespace WorkPlan

{

public partial class SiGlogWind : Window

{

private Base.Entities DataBase;

public SiGlogWind()

{

InitializeComponent();

try

{

DataBase = new Base.Entities();

//GetBase64ImageFromDb();

//PutImageBase64InDb(@"D:\321.jpg");

}

catch

{

MessageBox.Show("Не удалось подключиться к базе данных. Проверьте настройки подключения приложения.",

"Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

Close();

}

}

private void OnKeyDownHandler(object sender, KeyEventArgs e)

{

if (e.Key == Key.Return)

{

AuthorizProc();

}

if (e.Key == Key.Escape)

{ Close(); }

}

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

{

AuthorizProc();

}

private void AuthorizProc()

{

Base.Users User = DataBase.Users.SingleOrDefault(U => U.Логин == LoginText.Text && U.Пароль.Equals(Pass wordText.Password, StringComparison.CurrentCulture));

if (User != null)

{

SourceCore.UserID = User.ID\_user;

MainWindow window = new MainWindow(User.ID\_user);

window.Show();

Close();

}

else

{

MessageBox.Show("Неверно указан логин и/или пароль!", "Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

};

}

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

{

if (MessageBox.Show("Вы действительно хотите выйти из программы?", "Внимание", MessageBoxButton.OKCancel, MessageBoxImage.Warning) == MessageBoxResult.OK)

{

Close();

}

}

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

{

}

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

{

}

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

{

}

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

{

SiGlogREGWind window = new SiGlogREGWind(DataBase);

Close();

window.ShowDialog();

}

}

}

Программный код окна SiGlogREGWind

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices;

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.Shapes;

using System.Text.RegularExpressions;

namespace WorkPlan

{

public partial class SiGlogREGWind : Window

{

private Base.Entities DataBase;

public Border myBorder;

public SiGlogREGWind(Base.Entities DataBase)

{

InitializeComponent();

this.DataBase = DataBase;

}

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

{

SiGlogWind window = new SiGlogWind();

Close();

window.ShowDialog();

}

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

{

// Переброска необходимой информации во временные буферы

String Password = PasswordPasswordBox.Password;

Visibility Visibility = PasswordPasswordBox.Visibility;

double Width = PasswordPasswordBox.ActualWidth;

// Изменение подписи на кнопке

PasswordButton.Content = Visibility == Visibility.Visible ? "Скрыть" : "Показать";

// Переброска информации из TextBox'а в PasswordBox

PasswordPasswordBox.Password = PasswordTextBox.Text;

PasswordPasswordBox.Visibility = PasswordTextBox.Visibility;

PasswordPasswordBox.Width = PasswordTextBox.Width;

// Возврат информации из временных буферов в TextBox

PasswordTextBox.Text = Password;

PasswordTextBox.Visibility = Visibility;

PasswordTextBox.Width = Width;

}

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

{

check.Text = " " + check.Text;

if (captcha.Text == check.Text)

{

try

{

// Создание и инициализация нового пользователя системы

string passwordstring = PasswordPasswordBox.Password != "" ? PasswordPasswordBox.Password : PasswordTextBox.Text; ;

Regex validate = new Regex("^(?=.\*?[A-Z])(?=.\*?[a-z])(?=.\*?[0-9]).{8,}$");

if (validate.IsMatch(passwordstring))

{

Base.Users User = new Base.Users();

Base.Users UserFind = DataBase.Users.SingleOrDefault(U => U.Логин == LoginTextBox.Text);

if (UserFind == null)

{

User.Логин = LoginTextBox.Text;

User.Пароль = passwordstring;

User.Права = "WORKER";

int result = Int32.Parse(IDTextBox.Text);

User.ID\_employee = result;

// Добавление его в базу данных

DataBase.Users.Add(User);

// Сохранение изменений

DataBase.SaveChanges();

SiGlogWind window = new SiGlogWind();

Close();

window.ShowDialog();

}

else

{

MessageBox.Show("Данный логин уже занят!", "Пердупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

LoginTextBox.Text = "";

}

} else

{

MessageBox.Show("Пожалуйста укажите пароль, используя не менее 1 заглавной и 1 строчной буквы латинского алфавита, не менее 1 цифры и общей длинной не менее 8 символов!", "Пердупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

catch {

MessageBox.Show("Неправильные данные","Пердупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

else {

MessageBox.Show("НИПРАВЕЛЬНА!",

"Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

check.Text = "";

}

}

private void button1\_Click(object sender, RoutedEventArgs e)

{

String allowchar = " ";

allowchar = "A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z";

allowchar += "a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,y,z";

allowchar += "1,2,3,4,5,6,7,8,9,0";

//разделитель

char[] a = { ',' };

//расщепление массива по разделителю

String[] ar = allowchar.Split(a);

String pwd = " ";

string temp = " ";

Random r = new Random();

for (int i = 0; i < 5; i++)

{

temp = ar[(r.Next(0, ar.Length))];

pwd += temp;

}

captcha.Text = pwd;

}

}

Программный код окна MainFrame

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 WorkPlan.Pages

{

/// <summary>

/// Interaction logic for MainFrame.xaml

/// </summary>

public partial class MainFrame : Page

{

private Base.Entities DataBase;

int role = 0;

private Window mainWindow;

private double \_width;

private string userRole;

public MainFrame(int roleN, double width, double height, Window MainWindow)

{

InitializeComponent();

role = roleN;

mainWindow = MainWindow;

\_width = width;

DataBase = new Base.Entities();

init(role);

}

public void init(int role)

{

Base.Users UserFind = DataBase.Users.SingleOrDefault(U => U.ID\_user == role);

userRole = UserFind.Права;

if (userRole=="ADMIN" || userRole=="MAINBUH")

{

goods.Content = "Все товары";

apps.Content = "Все заявки";

admPanel.Content = "Редактировать списки";

admPanel.Click += admPanel\_Click;

if (userRole == "MAINBUH")

{

admPanel.Content = "Структура организации";

admPanel.Click += admPanel\_Click1;

}

}

else

{

goods.Content = "Все товары";

apps.Content = "Мои заявки";

admPanel.Content = "Структура организации";

admPanel.Click += admPanel\_Click1;

}

}

public void redraw(double width)

{

double widthneed = ((MainWindow)Window.GetWindow(this)).widthNow();

btnLight.Width = widthneed \* 0.3;

btnLight.Height = widthneed \* 0.3;

btnLight1.Width = widthneed \* 0.3;

btnLight1.Height = widthneed \* 0.3;

leftPanel.Margin = new Thickness(0,0,btnLight.Width\*0.3,0);

rigthPanel.Margin = new Thickness(btnLight1.Width\*0.3,0,0,0);

goods.FontSize = Math.Round(widthneed \* 0.05);

apps.FontSize = Math.Round(widthneed \* 0.05);

}

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

{

if (userRole=="ADMIN")

{

Goods goods = new Goods();

NavigationService.Navigate(goods);

}

else

{

userGoods uGoods = new userGoods(role);

NavigationService.Navigate(uGoods);

}

}

private void btnLight1\_Click(object sender, RoutedEventArgs e)

{

if (userRole == "ADMIN" || userRole == "MAINBUH")

{

Applications apps = new Applications();

NavigationService.Navigate(apps);

}

else

{

userApps uApps = new userApps(role);

NavigationService.Navigate(uApps);

}

}

private void Grid\_SizeChanged(object sender, SizeChangedEventArgs e)

{

redraw(\_width);

}

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

{

AdminPanel adminPanel= new AdminPanel();

NavigationService.Navigate(adminPanel);

}

private void admPanel\_Click1(object sender, RoutedEventArgs e)

{

Structure strPanel = new Structure();

NavigationService.Navigate(strPanel);

}

}

}

Программный код окна Goods

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;

using System.Windows.Forms;

using System.Collections.ObjectModel;

using TextBox = System.Windows.Controls.TextBox;

using System.Data.Entity;

namespace WorkPlan.Pages

{

public partial class Goods : Page

{

Base.wpEntities DataBase;

private Base.Goods SelectedApp;

public Goods()

{

InitializeComponent();

DataContext = this;

UpdateGrid(null);

DataBase = new Base.wpEntities();

}

public void UpdateGrid(Base.Goods Good)

{

if ((Good == null) && (GoodsGrid.ItemsSource != null))

{

Good = (Base.Goods)GoodsGrid.SelectedItem;

}

ObservableCollection<Base.Goods> Goods =

new ObservableCollection<Base.Goods>(SourceCore.MyBase.Goods.ToList());

GoodsGrid.ItemsSource = Goods;

GoodsGrid.SelectedItem = Good;

}

private void DeleteGood(object sender, RoutedEventArgs e)

{

try

{

SelectedApp = (Base.Goods)GoodsGrid.SelectedItem;

Base.Applications checkApps = DataBase.Applications.SingleOrDefault(q => q.ID\_goods == SelectedApp.ID\_goods);

if (System.Windows.MessageBox.Show("Удалить запись?", "Внимание", MessageBoxButton.OKCancel, MessageBoxImage.Warning) == MessageBoxResult.OK)

{

if (checkApps == null)

{

SourceCore.MyBase.Goods.Remove((Base.Goods)GoodsGrid.SelectedItem);

SourceCore.MyBase.SaveChanges();

UpdateGrid(null);

}

else

{

System.Windows.MessageBox.Show("Вы не можете удалить товар, который в настоящий момент учавствует в заявке пользователя!");

}

}

}

catch { System.Windows.MessageBox.Show("Вы не можете удалить товар, который в настоящий момент учавствует в заявке пользователя!"); }

}

private void OnKeyDownHandler(object sender, System.Windows.Input.KeyEventArgs e)

{

if (e.Key == Key.Escape)

{ NavigationService.GoBack(); }

}

private void addNewGood(object sender, RoutedEventArgs e)

{

AddGood newWindow = new AddGood(this);

newWindow.ShowDialog();

}

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

{

var textbox = sender as TextBox;

if (textbox.Text != "Введите название товара")

{

GoodsGrid.ItemsSource = SourceCore.MyBase.Goods.Where(Q => Q.Название.Contains(textbox.Text)).ToList();

}

}

private void filterText\_GotFocus(object sender, RoutedEventArgs e)

{

if (filterText.Text == "Введите название товара")

{

filterText.Text = "";

filterText.Foreground = new SolidColorBrush(Colors.Black);

}

}

private void filterText\_LostFocus(object sender, RoutedEventArgs e)

{

if (filterText.Text == "")

{

filterText.Text = "Введите название товара";

filterText.Foreground = new SolidColorBrush(Colors.Gray);

}

}

}

}

Программный код страницы Applications

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

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 WorkPlan.Pages

{

/// <summary>

/// Interaction logic for Applications.xaml

/// </summary>

public partial class Applications : Page

{

private bool filterMain;

public Base.Applications SelectedApp;

public Applications()

{

InitializeComponent();

DataContext = this;

//GoodsGrid.ItemsSource = SourceCore.MyBase.Goods.ToList();

UpdateGrid(null,false);

}

public void UpdateGrid(Base.Applications application, bool filter)

{

if ((application == null) && (AppGrid.ItemsSource != null))

{

application = (Base.Applications)AppGrid.SelectedItem;

}

comboFilter.ItemsSource = SourceCore.MyBase.Departments.ToList();

if(filter)

{

Base.Departments selected = comboFilter.SelectedItem as Base.Departments;

ObservableCollection<Base.Applications> applications =

new ObservableCollection<Base.Applications>(SourceCore.MyBase.Applications.Where(Q=>Q.ID\_department==selected.ID\_department).ToList());

AppGrid.ItemsSource = applications;

AppGrid.SelectedItem = application;

}

else

{

ObservableCollection<Base.Applications> applications =

new ObservableCollection<Base.Applications>(SourceCore.MyBase.Applications.ToList());

AppGrid.ItemsSource = applications;

AppGrid.SelectedItem = application;

filterMain = true;

}

}

private void GoToApplications(object sender, RoutedEventArgs e)

{

SelectedApp = (Base.Applications)AppGrid.SelectedItem;

appRedaction newWindow = new appRedaction(SelectedApp,this);

newWindow.ShowDialog();

}

private void OnKeyDownHandler(object sender, System.Windows.Input.KeyEventArgs e)

{

//????????????????????????HELP

if (e.Key == Key.Escape)

{ NavigationService.GoBack(); }

}

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

{

NavigationService.Navigate(new Pages.FinalPlan());

}

private void comboFilter\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (filterMain)

{

UpdateGrid(null, true);

}

}

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

{

filterMain = false;

comboFilter.SelectedItem = null;

UpdateGrid(null, false);

}

}

}

Программный код страницы userApps

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

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 WorkPlan.Pages

{

/// <summary>

/// Interaction logic for userApps.xaml

/// </summary>

public partial class userApps : Page

{

private int roleN = 0;

private Base.Applications SelectedApp;

public Base.Entities DataBase;

public string status;

public userApps(int roleID)

{

InitializeComponent();

DataContext = this;

DataBase = new Base.Entities();

roleN = roleID;

//GoodsGrid.ItemsSource = SourceCore.MyBase.Goods.ToList();

UpdateGrid(null);

AppGrid.UpdateLayout();

}

public void UpdateGrid(Base.Applications application)

{

if ((application == null) && (AppGrid.ItemsSource != null))

{

application = (Base.Applications)AppGrid.SelectedItem;

}

Base.Users setUsers = DataBase.Users.SingleOrDefault(U => U.ID\_user == roleN);

Base.Employee setEmp = DataBase.Employee.SingleOrDefault(U => U.ID\_employee == setUsers.ID\_employee);

ObservableCollection<Base.Applications> applications =

new ObservableCollection<Base.Applications>(SourceCore.MyBase.Applications.Where(U => U.ID\_department == setEmp.ID\_department).ToList());

AppGrid.ItemsSource = applications;

AppGrid.SelectedItem = application;

}

private void OnKeyDownHandler(object sender, KeyEventArgs e)

{

}

private void GoToApplications(object sender, RoutedEventArgs e)

{

SelectedApp = (Base.Applications)AppGrid.SelectedItem;

Base.Status setStatus = DataBase.Status.SingleOrDefault(U => U.ID\_status == SelectedApp.ID\_status);

if (SelectedApp.ID\_creator != roleN)

{

MessageBox.Show("Вы не можете редактировать чужую заявку!");

}

else

{

if (setStatus.Статус != "Принято" || setStatus.Статус!="Закрыто")

{

castRedact newWindow = new castRedact(SelectedApp, this);

newWindow.ShowDialog();

}

else

{

MessageBox.Show("Вы не можете редактировать эту заявку!");

}

}

}

private void delete(object sender, RoutedEventArgs e)

{

SelectedApp = (Base.Applications)AppGrid.SelectedItem;

Base.Status setStatus = DataBase.Status.SingleOrDefault(U => U.ID\_status == SelectedApp.ID\_status);

if (SelectedApp.ID\_creator != roleN)

{

MessageBox.Show("Вы не можете удалить чужую заявку!");

}

else

{

if (setStatus.Статус != "Принято" || setStatus.Статус != "Закрыто")

{

if (MessageBox.Show("Удалить запись?", "Внимание", MessageBoxButton.OKCancel, MessageBoxImage.Warning) == MessageBoxResult.OK)

{

SourceCore.MyBase.Applications.Remove((Base.Applications)AppGrid.SelectedItem);

SourceCore.MyBase.SaveChanges();

UpdateGrid(null);

}

}

else

{

MessageBox.Show("Вы не можете удалить эту заявку!");

}

}

}

}

}

Программный код страницы userGoods

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

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 WorkPlan.Pages

{

public partial class userGoods : Page

{

public Base.Goods SelectedGood;

private int roleN;

public userGoods(int roleID)

{

InitializeComponent();

DataContext = this;

roleN = roleID;

UpdateGrid(null);

}

public void UpdateGrid(Base.Goods Good)

{

if ((Good == null) && (GoodsGrid.ItemsSource != null))

{

Good = (Base.Goods)GoodsGrid.SelectedItem;

}

ObservableCollection<Base.Goods> Goods =

new ObservableCollection<Base.Goods>(SourceCore.MyBase.Goods.ToList());

GoodsGrid.ItemsSource = Goods;

GoodsGrid.SelectedItem = Good;

}

private void GoToApplications(object sender, RoutedEventArgs e)

{

SelectedGood = (Base.Goods)GoodsGrid.SelectedItem;

castApp newWindow =new castApp(SelectedGood,this,roleN);

newWindow.ShowDialog();

}

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

{

var textbox = sender as TextBox;

if (textbox.Text != "Введите название товара")

{

GoodsGrid.ItemsSource = SourceCore.MyBase.Goods.Where(Q => Q.Название.Contains(textbox.Text)).ToList();

}

}

private void filterText\_GotFocus(object sender, RoutedEventArgs e)

{

if (filterText.Text == "Введите название товара")

{

filterText.Text = "";

filterText.Foreground = new SolidColorBrush(Colors.Black);

}

}

private void filterText\_LostFocus(object sender, RoutedEventArgs e)

{

if (filterText.Text == "")

{

filterText.Text = "Введите название товара";

filterText.Foreground = new SolidColorBrush(Colors.Gray);

}

}

}

}

Программный код страницы AddGood

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.Shapes;

namespace WorkPlan

{

public partial class AddGood : Window

{

private Pages.Goods goodsPage;

public AddGood(Pages.Goods GoodsPage)

{

InitializeComponent();

DataContext = this;

goodsPage = GoodsPage;

}

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

{

try

{

var NewGood = new Base.Goods();

NewGood.Название = nametext.Text;

NewGood.Цена = Convert.ToDecimal(amounttext.Text);

NewGood.code = codetext.Text;

SourceCore.MyBase.Goods.Add(NewGood);

SourceCore.MyBase.SaveChanges();

goodsPage.UpdateGrid(null);

goodsPage.GoodsGrid.SelectedItem = NewGood;

goodsPage.GoodsGrid.UpdateLayout();

goodsPage.GoodsGrid.ScrollIntoView(goodsPage.GoodsGrid.SelectedItem);

Close();

}

catch

{

MessageBox.Show("Пожалуйста, введите название, используя любые символы, и укажите цену, используя только цифры");

}

}

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

{

Close();

}

private void increase(object sender, RoutedEventArgs e)

{

Int32 num = Convert.ToInt32(amounttext.Text);

amounttext.Text = ((num + 10).ToString());

}

private void decrease(object sender, RoutedEventArgs e)

{

if (Convert.ToInt32(amounttext.Text) >= 10)

{

Int32 num = Convert.ToInt32(amounttext.Text);

amounttext.Text = ((num - 10).ToString());

}

}

private void nametext\_GotFocus(object sender, RoutedEventArgs e)

{

if (nametext.Text.Equals("Введите сюда"))

{

nametext.Text = "";

nametext.Foreground = new SolidColorBrush(Colors.Black);

}

}

private void nametext\_LostFocus(object sender, RoutedEventArgs e)

{

if (nametext.Text.Equals(""))

{

nametext.Text = "Введите сюда";

nametext.Foreground = new SolidColorBrush(Colors.Gray);

}

}

}

}

Программный код страницы appRedaction

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.Shapes;

namespace WorkPlan

{

/// <summary>

/// Interaction logic for appRedaction.xaml

/// </summary>

public partial class appRedaction : Window

{

private Pages.Applications appPage;

private Base.Applications SelectedApp;

private Base.Entities DataBase;

public appRedaction(Base.Applications selectedItem, Pages.Applications AppPage)

{

InitializeComponent();

DataContext = this;

SelectedApp = selectedItem;

DataBase = new Base.Entities();

appPage = AppPage;

init();

}

private void init()

{

Base.Goods setGood = DataBase.Goods.SingleOrDefault(U => U.ID\_goods == SelectedApp.ID\_goods);

Base.Status setStatus = DataBase.Status.SingleOrDefault(U => U.ID\_status == SelectedApp.ID\_status);

Base.Departments setDep = DataBase.Departments.SingleOrDefault(U => U.ID\_department == SelectedApp.ID\_department);

nametext.Text = setGood.Название;

deptext.Text = setDep.Название;

amounttext.Text = SelectedApp.Количество.ToString();

combo.ItemsSource = SourceCore.MyBase.Status.ToList();

combo.Text = setStatus.Статус;

}

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

{

try

{

var EditApp = new Base.Applications();

EditApp = SourceCore.MyBase.Applications.First(p => p.ID\_application == SelectedApp.ID\_application);

EditApp.Количество = Convert.ToInt32(amounttext.Text);

EditApp.Status = (Base.Status)combo.SelectedItem;

SourceCore.MyBase.SaveChanges();

appPage.UpdateGrid(null, false) ;

appPage.AppGrid.SelectedItem = EditApp;

appPage.AppGrid.UpdateLayout();

appPage.AppGrid.ScrollIntoView(appPage.AppGrid.SelectedItem);

Close();

}

catch

{

MessageBox.Show("Пожалуйста, укажите количество, используя цифры");

}

}

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

{

Close();

}

private void increase(object sender, RoutedEventArgs e)

{

Int32 num = Convert.ToInt32(amounttext.Text);

amounttext.Text = ((num + 10).ToString());

}

private void decrease(object sender, RoutedEventArgs e)

{

if (Convert.ToInt32(amounttext.Text) >= 10)

{

Int32 num = Convert.ToInt32(amounttext.Text);

amounttext.Text = ((num - 10).ToString());

}

}

}

}

Программный код страницы castApp

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.Shapes;

namespace WorkPlan

{

/// <summary>

/// Interaction logic for castApp.xaml

/// </summary>

public partial class castApp : Window

{

private Pages.userGoods goodsPage;

private Base.Goods selectedGood;

private Base.Entities DataBase;

private int roleN;

public castApp(Base.Goods SelectedGood, Pages.userGoods GoodsPage, int roleID)

{

InitializeComponent();

DataContext = this;

selectedGood = SelectedGood;

goodsPage = GoodsPage;

roleN = roleID;

DataBase = new Base.Entities();

init();

}

private void init()

{

Base.Users setUsers = DataBase.Users.SingleOrDefault(U => U.ID\_user == roleN);

Base.Employee setEmp = DataBase.Employee.SingleOrDefault(U => U.ID\_employee == setUsers.ID\_employee);

Base.Departments setDep = DataBase.Departments.SingleOrDefault(U => U.ID\_department == setEmp.ID\_department);

nametext.Text = selectedGood.Название;

amounttext.Text = "0";

deptext.Text = setDep.Название;

}

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

{

var newApp = new Base.Applications();

Base.Users setUsers = DataBase.Users.SingleOrDefault(U => U.ID\_user == roleN);

Base.Employee setEmp = DataBase.Employee.SingleOrDefault(U => U.ID\_employee == setUsers.ID\_employee);

Base.Status status = DataBase.Status.SingleOrDefault(U=>U.Статус=="Не расмотренно");

newApp.ID\_goods = selectedGood.ID\_goods;

newApp.ID\_department = setEmp.ID\_department;

newApp.ID\_status = status.ID\_status;

newApp.ID\_creator = roleN;

newApp.created\_at= DateTime.Now;

newApp.Количество = Convert.ToInt32(amounttext.Text);

SourceCore.MyBase.Applications.Add(newApp);

SourceCore.MyBase.SaveChanges();

Close();

}

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

{

Close();

}

private void decrease(object sender, RoutedEventArgs e)

{

if (Convert.ToInt32(amounttext.Text) >= 1)

{

Int32 num = Convert.ToInt32(amounttext.Text);

amounttext.Text = ((num - 1).ToString());

}

}

private void increase(object sender, RoutedEventArgs e)

{

Int32 num = Convert.ToInt32(amounttext.Text);

amounttext.Text = ((num + 1).ToString());

}

}

}

Программный код страницы castRedact

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.Shapes;

namespace WorkPlan

{

/// <summary>

/// Interaction logic for castRedact.xaml

/// </summary>

public partial class castRedact : Window

{

private Pages.userApps appPage;

private Base.Applications SelectedApp;

private Base.Entities DataBase;

public castRedact(Base.Applications selectedItem, Pages.userApps AppPage)

{

InitializeComponent();

DataContext = this;

SelectedApp = selectedItem;

DataBase = new Base.Entities();

appPage = AppPage;

init();

}

private void init()

{

Base.Goods setGood = DataBase.Goods.SingleOrDefault(U => U.ID\_goods == SelectedApp.ID\_goods);

Base.Status setStatus = DataBase.Status.SingleOrDefault(U => U.ID\_status == SelectedApp.ID\_status);

Base.Departments setDep = DataBase.Departments.SingleOrDefault(U => U.ID\_department == SelectedApp.ID\_department);

nametext.Text = setGood.Название;

statustext.Text = setStatus.Статус;

deptext.Text = setDep.Название;

amounttext.Text = SelectedApp.Количество.ToString();

}

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

{

try

{

var EditApp = new Base.Applications();

EditApp = SourceCore.MyBase.Applications.First(P => P.ID\_application == SelectedApp.ID\_application);

EditApp.Количество = Convert.ToInt32(amounttext.Text);

SourceCore.MyBase.SaveChanges();

appPage.UpdateGrid(null);

appPage.AppGrid.SelectedItem = EditApp;

appPage.AppGrid.UpdateLayout();

appPage.AppGrid.ScrollIntoView(appPage.AppGrid.SelectedItem);

Close();

}

catch

{

MessageBox.Show("Пожалуйста, укажите количество, используя цифры");

}

}

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

{

Close();

}

private void increase(object sender, RoutedEventArgs e)

{

Int32 num = Convert.ToInt32(amounttext.Text);

amounttext.Text = ((num + 1).ToString());

}

private void decrease(object sender, RoutedEventArgs e)

{

if (Convert.ToInt32(amounttext.Text) >= 1)

{

Int32 num = Convert.ToInt32(amounttext.Text);

amounttext.Text = ((num - 1).ToString());

}

}

}

}

Программый код окна LoadingInit

using System;

using System.Collections.Generic;

using System.Linq;

using System.Reflection;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Timers;

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 WorkPlan

{

/// <summary>

/// Логика взаимодействия для LoadingInit.xaml

/// </summary>

public partial class LoadingInit : Window

{

private Base.Entities DataBase;

public LoadingInit()

{

InitializeComponent();

try

{

DataBase = new Base.Entities();

checkIfAdminPresent();

}

catch

{

MessageBox.Show("Не удалось подключиться к базе данных. Проверьте настройки подключения приложения.",

"Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

Close();

}

}

public void checkIfAdminPresent()

{

Base.Users User = DataBase.Users.SingleOrDefault(U=> U.Права == "ADMIN");

if (User != null)

{

SiGlogWind window = new SiGlogWind();

window.Show();

Close();

}

else

{

CreateAdmin window = new CreateAdmin(DataBase);

window.Show();

Close();

}

}

}

}

Программный код окна CreateAdmin

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices;

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.Shapes;

using System.Text.RegularExpressions;

namespace WorkPlan

{

/// <summary>

/// Interaction logic for SiGlogREGWind.xaml

/// </summary>

public partial class CreateAdmin : Window

{

private Base.Entities DataBase;

public Border myBorder;

public CreateAdmin(Base.Entities DataBase)

{

InitializeComponent();

this.DataBase = DataBase;

}

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

{

Close();

}

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

{

try

{

// Создание и инициализация нового пользователя системы

string passwordstring = PasswordPasswordBox.Text;

Base.Users User = new Base.Users();

Base.Users UserFind = DataBase.Users.SingleOrDefault(U => U.Логин == LoginTextBox.Text);

if (UserFind == null)

{

User.Логин = LoginTextBox.Text;

User.Пароль = passwordstring;

User.Права = "ADMIN";

// Добавление его в базу данных

DataBase.Users.Add(User);

// Сохранение изменений

DataBase.SaveChanges();

SiGlogWind window = new SiGlogWind();

Close();

window.ShowDialog();

}

else

{

MessageBox.Show("Данный логин уже занят!", "Пердупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

LoginTextBox.Text = "";

}

}

catch {

MessageBox.Show("Неправильные данные","Пердупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

private void button1\_Click(object sender, RoutedEventArgs e)

{

String allowchar = " ";

allowchar = "A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z";

allowchar += "a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,y,z";

allowchar += "1,2,3,4,5,6,7,8,9,0";

//разделитель

char[] a = { ',' };

//расщепление массива по разделителю

String[] ar = allowchar.Split(a);

String pwd = " ";

string temp = " ";

Random r = new Random();

for (int i = 0; i < 5; i++)

{

temp = ar[(r.Next(0, ar.Length))];

pwd += temp;

}

}

}

}

Программный код окна addToStructure

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.Shapes;

using WorkPlan.Base;

namespace WorkPlan

{

/// <summary>

/// Логика взаимодействия для addToStructure.xaml

/// </summary>

public partial class addToStructure : Window

{

public Pages.Structure strParent;

public string RoleName;

public int depNum;

public string depFull;

public addToStructure(string rolename,int depnum, Pages.Structure parent)

{

InitializeComponent();

RoleName = rolename;

depNum = depnum;

strParent = parent;

emp.ItemsSource = SourceCore.MyBase.Employee.ToList();

depFull = rolename + depNum;

}

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

{

var CurrentEmp = emp.SelectedItem as Employee;

switch(RoleName)

{

case "mainman":

var NewEmp = new Base.Structure();

NewEmp.id\_Emp = CurrentEmp.ID\_employee;

NewEmp.roleo = RoleName;

SourceCore.MyBase.Structure.Add(NewEmp);

SourceCore.MyBase.SaveChanges();

break;

case "buh":

var NewEmpBuh = new Base.Structure();

NewEmpBuh.id\_Emp = CurrentEmp.ID\_employee;

NewEmpBuh.roleo = RoleName;

SourceCore.MyBase.Structure.Add(NewEmpBuh);

SourceCore.MyBase.SaveChanges();

break;

case "mainDep\_":

var newDepMain = new Base.Structure();

newDepMain.id\_Emp = CurrentEmp.ID\_employee;

newDepMain.roleo = depFull;

SourceCore.MyBase.Structure.Add(newDepMain);

SourceCore.MyBase.SaveChanges();

break;

case "zamDep\_":

var newDepZam = new Base.Structure();

newDepZam.id\_Emp = CurrentEmp.ID\_employee;

newDepZam.roleo = depFull;

SourceCore.MyBase.Structure.Add(newDepZam);

SourceCore.MyBase.SaveChanges();

break;

}

strParent.loadStructure();

Close();

}

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

{

Close();

}

}

}

Программный код окна PrintPreview

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Data;

using System.Linq;

using System.Reflection;

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.Forms;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace WorkPlan

{

public class GrouppedItem

{

public int appId { get; set; }

public string Name { get; set; }

public decimal Sum { get; set; }

public int year { get; set; }

public int Id { get; set; }

public string IdString { get; set; }

public decimal sum1 { get; set; }

public decimal sum2 { get; set; }

public decimal sum3 { get; set; }

public decimal sumA { get; set; }

public int idNum { get; set; }

public string code { get; set; }

public string description { get; set; }

}

public partial class PrintPreview : Window

{

public Base.Entities DataBase;

public PrintPreview()

{

InitializeComponent();

DataContext = this;

DataBase = new Base.Entities();

UpdatePrintGrid(null);

}

public void UpdatePrintGrid(Base.Applications application)

{

if ((application == null) && (printGrid.ItemsSource != null))

{

application = (Base.Applications)printGrid.SelectedItem;

}

Base.Status setstatus = DataBase.Status.SingleOrDefault(Q => Q.Статус == "Принято");

ObservableCollection<Base.Applications> applications =

new ObservableCollection<Base.Applications>(SourceCore.MyBase.Applications.Where(Q => Q.ID\_status == setstatus.ID\_status).ToList());

var groupedData = applications

.GroupBy(item => item.ID\_goods)

.Select((group, index )=>

{

var totalValue = group.Sum(item => item.TotalPrice);

var yearS = group.Min(item=>item.year);

var y1p = group.Sum(item=>item.year1price);

var y2p = group.Sum(item => item.year2price);

var y3p = group.Sum (item=>item.year3price);

var yAp = group.Sum(item => item.yearAprice);

var amount = group.Sum(item=>item.Количество);

return new GrouppedItem

{

idNum = SourceCore.MyBase.Goods.Single(Q=>Q.ID\_goods == group.Key).ID\_goods,

Id = (index + 1),

Name = SourceCore.MyBase.Goods.Single(Q=>Q.ID\_goods==group.Key).Название,

year = Convert.ToInt32(yearS),

Sum = totalValue,

sum1 = (decimal)y1p,

sum2 = (decimal)y2p,

sum3 = (decimal)y3p,

sumA = (decimal)yAp,

code = SourceCore.MyBase.Goods.Single(Q=>Q.ID\_goods==group.Key).code,

description = "Закупка товара \"" + SourceCore.MyBase.Goods.Single(Q=>Q.ID\_goods==group.Key).Название+ "\" в количестве "+amount+"шт.",

};

})

.Select(item =>

{

item.IdString = item.Id.ToString("D4");

return item;

});

var groupedCollection = new ObservableCollection<GrouppedItem>(groupedData);

printGrid.ItemsSource = groupedCollection;

printGrid.SelectedItem = application;

}

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

{

this.Close();

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

System.Windows.Controls.PrintDialog p =new System.Windows.Controls.PrintDialog ();

if(p.ShowDialog() == true)

{

p.PrintVisual(this,"Printing");

}

}

}

}

Программный код страницы AdminPanel

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 WorkPlan.Pages

{

/// <summary>

/// Логика взаимодействия для AdminPanel.xaml

/// </summary>

public partial class AdminPanel : Page

{

public AdminPanel()

{

InitializeComponent();

adminLabel.Content = "Адмнистратора\n ";

fstBlock.Text = "1. Создайте новый отдел нажав на \nкнопку";

fstLabel.Text = "Редактировать список Отделов\n ";

scnBlock.Text = "2. Создайте нового сотрудника\nнажав на кнопку";

scnLabel.Text = "Редактировать список Сотрудников\n ";

thrBlock.Text = "3. (Опционально) Создайте нового\nпользователя нажав на кнопку";

thrLabel.Text = "Редактировать список Пользователей";

fouBlock.Text = "Или выдайте сотруднику его номер,\nчтобы он зарегистрировался сам";

}

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

{

NavigationService.Navigate(new Pages.redUsersList());

}

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

{

NavigationService.Navigate(new Pages.redEmpList());

}

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

{

NavigationService.Navigate(new Pages.redDepsList());

}

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

{

NavigationService.Navigate(new Pages.redSpecList());

}

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

{

NavigationService.Navigate(new Pages.Structure());

}

}

}

Программный код страницы FinalPlan

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

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;

using System.Drawing.Printing;

namespace WorkPlan.Pages

{

/// <summary>

/// Логика взаимодействия для FinalPlan.xaml

/// </summary>

public partial class FinalPlan : Page

{

private Base.Applications selectedApp;

public Base.Entities DataBase;

public FinalPlan()

{

InitializeComponent();

DataContext = this;

DataBase = new Base.Entities();

UpdateGrid(null);

int year = System.DateTime.Now.Year;

for(int i = 0; i<=4; i++)

{

yearCombo.Items.Add(year + i);

}

yearCombo.SelectedIndex = 0;

redFinal.Width = 0;

}

public void UpdateGrid(Base.Applications application)

{

if ((application == null) && (AppGrid.ItemsSource != null))

{

application = (Base.Applications)AppGrid.SelectedItem;

}

Base.Status setstatus = DataBase.Status.SingleOrDefault(Q=>Q.Статус=="Принято");

ObservableCollection<Base.Applications> applications =

new ObservableCollection<Base.Applications>(SourceCore.MyBase.Applications.Where(Q=>Q.ID\_status==setstatus.ID\_status).ToList());

AppGrid.ItemsSource = applications;

AppGrid.SelectedItem = application;

}

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

{

foreach(Base.Applications apps in DataBase.Applications)

{

Base.Applications app = new Base.Applications();

app = SourceCore.MyBase.Applications.Single(Q => Q.ID\_application == apps.ID\_application);

if (apps.year == DateTime.Now.Year)

{

app.year1price = app.TotalPrice;

app.year2price = 0;

app.year3price = 0;

app.yearAprice = 0;

}

if (apps.year == DateTime.Now.Year+1)

{

app.year1price = 0;

app.year2price = app.TotalPrice;

app.year3price = 0;

app.yearAprice = 0;

}

if (apps.year == DateTime.Now.Year+2)

{

app.year1price = 0;

app.year2price = 0;

app.year3price = app.TotalPrice;

app.yearAprice = 0;

}

if (apps.year >= DateTime.Now.Year+3)

{

app.year1price = 0;

app.year2price = 0;

app.year3price = 0;

app.yearAprice = app.TotalPrice;

}

SourceCore.MyBase.SaveChanges();

}

PrintPreview pp = new PrintPreview();

pp.Show();

}

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

{

selectedApp = (Base.Applications)AppGrid.SelectedItem;

redFinal.Width = 300;

}

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

{

try

{

int yearSelect = (int)yearCombo.SelectedItem;

Base.Applications redApp = SourceCore.MyBase.Applications.Single(Q=>Q.ID\_application== selectedApp.ID\_application);

redApp.year = yearSelect;

if(yearSelect==DateTime.Now.Year)

{

redApp.year1price = redApp.TotalPrice;

redApp.year2price = 0;

redApp.year3price = 0;

redApp.yearAprice = 0;

}

if(yearSelect==DateTime.Now.Year+1)

{

redApp.year1price = 0;

redApp.year2price= redApp.TotalPrice;

redApp.year3price = 0;

redApp.yearAprice = 0;

}

if(yearSelect==DateTime.Now.Year+2)

{

redApp.year1price = 0;

redApp.year2price = 0;

redApp.year3price= redApp.TotalPrice;

redApp.yearAprice = 0;

}

if (yearSelect >= DateTime.Now.Year + 3)

{

redApp.year1price = 0;

redApp.year2price = 0;

redApp.year3price = 0;

redApp.yearAprice = redApp.TotalPrice;

}

SourceCore.MyBase.SaveChanges();

UpdateGrid(null);

}

catch

{

MessageBox.Show("ERROR 4040404040404040404040404");

}

redFinal.Width = 0;

}

}

}

Программный код страницы redDepsList

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

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 WorkPlan.Pages

{

/// <summary>

/// Логика взаимодействия для redDepsList.xaml

/// </summary>

public partial class redDepsList : Page

{

public Base.Departments SelectedDep;

private Base.Entities DataBase;

private bool addOrRed = false;

public redDepsList()

{

InitializeComponent();

DataContext = this;

addPanel.Visibility = Visibility.Hidden;

DataBase = new Base.Entities();

UpdateGrid(null);

}

public void UpdateGrid(Base.Departments department)

{

if ((department == null) && (AppGrid.ItemsSource != null))

{

department = (Base.Departments)AppGrid.SelectedItem;

}

ObservableCollection<Base.Departments> departments =

new ObservableCollection<Base.Departments>(SourceCore.MyBase.Departments.ToList());

AppGrid.ItemsSource = departments;

AppGrid.SelectedItem = department;

}

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

{

NavigationService.GoBack();

}

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

{

if(addOrRed)

{

try

{

var newDep = new Base.Departments();

newDep.Название = name.Text;

newDep.Кол\_во\_сотрудников = count.Text;

SourceCore.MyBase.Departments.Add(newDep);

SourceCore.MyBase.SaveChanges();

UpdateGrid(newDep);

addPanel.Visibility = Visibility.Collapsed;

}

catch

{

MessageBox.Show("Перепроверьте данные");

}

}

else

{

try

{

var editDep = new Base.Departments();

editDep = SourceCore.MyBase.Departments.First(p => p.ID\_department == SelectedDep.ID\_department);

editDep.Название = name.Text;

editDep.Кол\_во\_сотрудников = count.Text;

SourceCore.MyBase.SaveChanges();

UpdateGrid(editDep);

AppGrid.SelectedItem= editDep;

addPanel.Visibility = Visibility.Collapsed;

}

catch

{

MessageBox.Show("Перепроверьте данные");

}

}

}

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

{

addPanel.Visibility = Visibility.Hidden;

}

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

{

try

{

SelectedDep = (Base.Departments)AppGrid.SelectedItem;

if (MessageBox.Show("Удалить отдел?", "Внимание", MessageBoxButton.OKCancel, MessageBoxImage.Warning) == MessageBoxResult.OK)

{

SourceCore.MyBase.Departments.Remove((Base.Departments)AppGrid.SelectedItem);

SourceCore.MyBase.SaveChanges();

UpdateGrid(null);

}

}

catch

{

MessageBox.Show("Не выбран отдел");

}

}

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

{

SelectedDep = (Base.Departments)AppGrid.SelectedItem;

if (SelectedDep != null)

{

addBtn.Content = "Редактировать";

mainLabel.Content = "Редактировать";

addOrRed = false;

name.Text = SelectedDep.Название;

count.Text = SelectedDep.Кол\_во\_сотрудников;

addPanel.Visibility = Visibility.Visible;

}

else

{

MessageBox.Show("Не выбран отдел");

}

}

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

{

addBtn.Content = "Добавить";

mainLabel.Content = "Добавить";

addOrRed = true;

name.Text = "";

count.Text = "";

addPanel.Visibility = Visibility.Visible;

}

}

}

Программный код страницы redEmpList

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

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 WorkPlan.Base;

namespace WorkPlan.Pages

{

public partial class redEmpList : Page

{

public Base.Employee SelectedEmp;

public Base.Departments SelectedDep;

private Base.Entities DataBase;

private bool addOrRed = false;

public List<Departments> Dep { get; set; }

public List<Specializations> Spec { get; set; }

public redEmpList()

{

InitializeComponent();

DataContext = this;

UpdateGrid(null);

DataBase = new Base.Entities();

addPanel.Visibility = Visibility.Hidden;

}

public void UpdateGrid(Base.Employee emp)

{

if ((emp == null) && (AppGrid.ItemsSource != null))

{

emp = (Base.Employee)AppGrid.SelectedItem;

}

ObservableCollection<Base.Employee> emps =

new ObservableCollection<Base.Employee>(SourceCore.MyBase.Employee.ToList());

AppGrid.ItemsSource = emps;

AppGrid.SelectedItem = emp;

}

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

{

addBtn.Content = "Добавить";

mainLabel.Content = "Добавить";

addOrRed = true;

fio.Text = "";

spec.Text = "";

department.Text = "";

ObservableCollection<Base.Departments> dep =

new ObservableCollection<Departments>(SourceCore.MyBase.Departments.ToList());

Dep = dep.ToList();

department.ItemsSource = Dep;

ObservableCollection<Base.Specializations> specl =

new ObservableCollection<Specializations>(SourceCore.MyBase.Specializations.ToList());

Spec = specl.ToList();

spec.ItemsSource = Spec;

addPanel.Visibility= Visibility.Visible;

}

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

{

SelectedEmp = (Base.Employee)AppGrid.SelectedItem;

if (SelectedEmp != null)

{

addBtn.Content = "Редактировать";

mainLabel.Content = "Редактировать";

addOrRed = false;

fio.Text = SelectedEmp.ФИО;

ObservableCollection<Base.Specializations> specl =

new ObservableCollection<Specializations>(SourceCore.MyBase.Specializations.ToList());

Spec = specl.ToList();

spec.ItemsSource = Spec;

ObservableCollection<Base.Departments> dep =

new ObservableCollection<Departments>(SourceCore.MyBase.Departments.ToList());

Dep = dep.ToList();

department.ItemsSource = Dep;

addPanel.Visibility = Visibility.Visible;

}

else

{

MessageBox.Show("Не выбран сотрудник");

}

}

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

{

try

{

SelectedEmp = (Base.Employee)AppGrid.SelectedItem;

if (MessageBox.Show("Удалить сотрудника?", "Внимание", MessageBoxButton.OKCancel, MessageBoxImage.Warning) == MessageBoxResult.OK)

{

SourceCore.MyBase.Employee.Remove((Base.Employee)AppGrid.SelectedItem);

SourceCore.MyBase.SaveChanges();

UpdateGrid(null);

}

}

catch

{

MessageBox.Show("Не выбран сотрудник");

}

}

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

{

NavigationService.GoBack();

}

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

{

if(addOrRed)

{

try

{

var specl = spec.SelectedItem as Specializations;

var dept = department.SelectedItem as Departments;

Base.Specializations setSpec = DataBase.Specializations.SingleOrDefault(M => M.id\_specialization == specl.id\_specialization) ;

Base.Departments setDep = DataBase.Departments.SingleOrDefault(U => U.ID\_department == dept.ID\_department);

var newEmp = new Base.Employee();

newEmp.ФИО = fio.Text;

newEmp.ID\_specialization = specl.id\_specialization;

newEmp.ID\_department = dept.ID\_department;

SourceCore.MyBase.Employee.Add(newEmp);

SourceCore.MyBase.SaveChanges();

UpdateGrid(newEmp);

addPanel.Visibility= Visibility.Collapsed;

}

catch

{

MessageBox.Show("Перепроверьте данные");

}

}

else

{

var specl = spec.SelectedItem as Specializations;

var depts = department.SelectedItem as Departments;

if (depts == null || specl == null)

{

MessageBox.Show("Перепроверьте данные");

}

else

{

try

{

var editEmp = new Base.Employee();

editEmp = SourceCore.MyBase.Employee.First(P => P.ID\_employee == SelectedEmp.ID\_employee);

editEmp.ФИО = fio.Text;

editEmp.ID\_specialization= specl.id\_specialization;

editEmp.ID\_department = depts.ID\_department;

SourceCore.MyBase.SaveChanges();

UpdateGrid(editEmp);

AppGrid.SelectedItem= editEmp;

addPanel.Visibility= Visibility.Collapsed;

}

catch

{

MessageBox.Show("Перепроверьте данные");

}

}

}

}

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

{

addPanel.Visibility = Visibility.Hidden;

}

}

}

Программный код страницы redSpecList

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

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 WorkPlan.Pages

{

public partial class redSpecList : Page

{

public Base.Specializations SelectedSpec;

private Base.Entities DataBase;

private bool addOrRed = false;

public redSpecList()

{

InitializeComponent();

DataContext= this;

addPanel.Visibility = Visibility.Hidden;

DataBase = new Base.Entities();

UpdateGrid(null);

}

public void UpdateGrid(Base.Specializations specialization)

{

if ((specialization == null) && (AppGrid.ItemsSource != null))

{

specialization = (Base.Specializations)AppGrid.SelectedItem;

}

ObservableCollection<Base.Specializations> specializations =

new ObservableCollection<Base.Specializations>(SourceCore.MyBase.Specializations.ToList());

AppGrid.ItemsSource = specializations;

AppGrid.SelectedItem = specialization;

}

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

{

NavigationService.GoBack();

}

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

{

addPanel.Visibility = Visibility.Hidden;

}

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

{

try

{

SelectedSpec = (Base.Specializations)AppGrid.SelectedItem;

if (MessageBox.Show("Удалить должность?", "Внимание", MessageBoxButton.OKCancel, MessageBoxImage.Warning) == MessageBoxResult.OK)

{

SourceCore.MyBase.Specializations.Remove((Base.Specializations)AppGrid.SelectedItem);

SourceCore.MyBase.SaveChanges();

UpdateGrid(null);

}

}

catch

{

MessageBox.Show("Не выбрана должность");

}

}

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

{

SelectedSpec = (Base.Specializations)AppGrid.SelectedItem;

if (SelectedSpec != null)

{

addBtn.Content = "Редактировать";

mainLabel.Content = "Редактировать";

addOrRed = false;

name.Text = SelectedSpec.Name\_specialization;

addPanel.Visibility = Visibility.Visible;

}

else

{

MessageBox.Show("Не выбрана должность");

}

}

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

{

addBtn.Content = "Добавить";

mainLabel.Content = "Добавить";

addOrRed = true;

name.Text = "";

addPanel.Visibility = Visibility.Visible;

}

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

{

if (addOrRed)

{

try

{

var newSpec = new Base.Specializations();

newSpec.Name\_specialization= name.Text;

SourceCore.MyBase.Specializations.Add(newSpec);

SourceCore.MyBase.SaveChanges();

UpdateGrid(newSpec);

addPanel.Visibility = Visibility.Collapsed;

}

catch

{

MessageBox.Show("Перепроверьте данные");

}

}

else

{

try

{

var editSpec = new Base.Specializations();

editSpec = SourceCore.MyBase.Specializations.First(p => p.id\_specialization == SelectedSpec.id\_specialization);

editSpec.Name\_specialization = name.Text;

SourceCore.MyBase.SaveChanges();

UpdateGrid(editSpec);

AppGrid.SelectedItem = editSpec;

addPanel.Visibility = Visibility.Collapsed;

}

catch

{

MessageBox.Show("Перепроверьте данные");

}

}

}

}

}

Программный код страницы redUsersList

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

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;

using WorkPlan.Base;

namespace WorkPlan.Pages

{

public partial class redUsersList : Page

{

public Base.Users SelectedUser;

public Base.Employee SelectedEmp;

private Base.Entities DataBase;

private bool addOrRed = false;

public List<Employee> Emp { get; set; }

public redUsersList()

{

InitializeComponent();

DataContext = this;

UpdateGrid(null);

DataBase = new Base.Entities();

addPanel.Visibility = Visibility.Hidden;

}

public void UpdateGrid(Base.Users user)

{

if((user == null)&&(AppGrid.ItemsSource != null))

{

user = (Base.Users)AppGrid.SelectedItem;

}

ObservableCollection<Base.Users> users =

new ObservableCollection<Base.Users>(SourceCore.MyBase.Users.ToList());

AppGrid.ItemsSource = users;

AppGrid.SelectedItem = user;

}

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

{

addPanel.Visibility=Visibility.Hidden;

}

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

{

addBtn.Content = "Добавить";

mainLabel.Content = "Добавить";

addOrRed = true;

login.Text = "";

pass.Text = "";

rights.Text = "";

employee.Text="";

addPanel.Visibility = Visibility.Visible;

ObservableCollection<Base.Employee> emp =

new ObservableCollection<Base.Employee>(SourceCore.MyBase.Employee.ToList());

Emp = emp.ToList();

employee.ItemsSource = Emp;

}

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

{

SelectedUser = (Base.Users)AppGrid.SelectedItem;

if (SelectedUser != null)

{

addBtn.Content = "Редактировать";

mainLabel.Content = "Редактировать";

addPanel.Visibility = Visibility.Visible;

addOrRed = false;

login.Text = SelectedUser.Логин;

pass.Text = SelectedUser.Пароль;

rights.Text = SelectedUser.Права;

ObservableCollection<Base.Employee> emp =

new ObservableCollection<Base.Employee>(SourceCore.MyBase.Employee.ToList());

Emp = emp.ToList();

Base.Employee setEmp = DataBase.Employee.SingleOrDefault(U => U.ID\_employee == SelectedUser.ID\_employee);

employee.ItemsSource = Emp;

employee.SelectedItem = setEmp;

}

else

{

MessageBox.Show("Не выбран пользователь");

}

}

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

{

try

{

SelectedUser = (Base.Users)AppGrid.SelectedItem;

if (SelectedUser.Права != "ADMIN")

{

if (MessageBox.Show("Удалить пользователя?", "Внимание", MessageBoxButton.OKCancel, MessageBoxImage.Warning) == MessageBoxResult.OK)

{

SourceCore.MyBase.Users.Remove((Base.Users)AppGrid.SelectedItem);

SourceCore.MyBase.SaveChanges();

UpdateGrid(null);

}

}

else

{

if (MessageBox.Show("Удалить Администратора?", "Внимание", MessageBoxButton.OKCancel, MessageBoxImage.Warning) == MessageBoxResult.OK)

{

SourceCore.MyBase.Users.Remove((Base.Users)AppGrid.SelectedItem);

SourceCore.MyBase.SaveChanges();

UpdateGrid(null);

}

}

}

catch

{

MessageBox.Show("Не выбран пользователь");

}

}

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

{

if (addOrRed)

{

try

{

var empl = employee.SelectedItem as Employee;

Base.Employee setEmp = DataBase.Employee.SingleOrDefault(U => U.ID\_employee == empl.ID\_employee);

var newUser = new Base.Users();

newUser.Логин = login.Text;

newUser.Пароль = pass.Text;

newUser.Права = rights.Text;

newUser.ID\_employee = empl.ID\_employee;

SourceCore.MyBase.Users.Add(newUser);

SourceCore.MyBase.SaveChanges();

UpdateGrid(newUser);

addPanel.Visibility = Visibility.Collapsed;

}

catch

{

MessageBox.Show("Перепроверьте данные");

}

}

else

{

var empl = employee.SelectedItem as Employee;

if (empl == null)

{

MessageBox.Show("Перепроверьте данные");

}

else

{

try

{

var EditUser = new Base.Users();

EditUser = SourceCore.MyBase.Users.First(p => p.ID\_user == SelectedUser.ID\_user);

EditUser.Логин = login.Text;

EditUser.Пароль = pass.Text;

EditUser.Права = rights.Text;

EditUser.ID\_employee = empl.ID\_employee;

SourceCore.MyBase.SaveChanges();

UpdateGrid(EditUser);

AppGrid.SelectedItem = EditUser;

addPanel.Visibility = Visibility.Collapsed;

}

catch

{

MessageBox.Show("Перепроверьте данные");

}

}

}

}

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

{

NavigationService.GoBack();

}

}

}

Программный код страницы Structure

using System;

using System.Activities.Expressions;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Drawing.Imaging;

using System.Linq;

using System.Runtime.Remoting.Messaging;

using System.Security.Cryptography.X509Certificates;

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 WorkPlan.Base;

namespace WorkPlan.Pages

{

public partial class Structure : Page

{

public List<Employee> Emp { get; set; }

private Base.Entities DataBase;

private bool loadStrc = true;

private int depNum = 0;

private Base.Users User;

public Structure()

{

InitializeComponent();

DataBase = new Base.Entities();

User = SourceCore.MyBase.Users.Single(Q=>Q.ID\_user==SourceCore.UserID);

loadStructure();

backButton.Background = new ImageBrush(new BitmapImage(

new Uri("xmark.png", UriKind.Relative)));

}

public void loadStructure()

{

List<int> depIds = new List<int>();

stackTest.Children.Clear();

int str = DataBase.Structure.Count();

Base.Structure strucMain = DataBase.Structure.SingleOrDefault(U => U.roleo == "mainman");

if (strucMain == null)

{

var lostlabel = new Label

{

Content = "Не назначен руководитель",

FontSize = 40,

HorizontalAlignment = HorizontalAlignment.Center,

};

stackTest.Children.Add(lostlabel);

var btnMainAdd = new Button

{

Content = "Добавить руководителя",

Width = 30,

Height = 30,

};

Style style = this.FindResource("addStyle") as Style;

btnMainAdd.Style = style;

btnMainAdd.Click += addMain;

stackTest.Children.Add(btnMainAdd);

}

else

{

Base.Employee sEmpTest = DataBase.Employee.First(U => U.ID\_employee == strucMain.id\_Emp);

Base.Specializations specTest = DataBase.Specializations.First(U => U.id\_specialization == sEmpTest.ID\_specialization);

var label = new Label

{

Content = sEmpTest.ФИО + " - " + specTest.Name\_specialization,

FontSize = 40,

HorizontalAlignment = HorizontalAlignment.Center,

};

var btn = new Button

{

Content = "Удалить",

VerticalAlignment = VerticalAlignment.Center,

Width = 30,

Height = 30,

};

Style style = this.FindResource("testStyle") as Style;

btn.Style = style;

btn.Click += delMain;

stackTest.Children.Add(label);

stackTest.Children.Add(btn);

}

Base.Structure strucBuh = DataBase.Structure.SingleOrDefault(U => U.roleo == "buh");

if (strucBuh == null)

{

var lostlabel = new Label

{

Content = "Не назначен бухгалтер",

FontSize = 40,

HorizontalAlignment = HorizontalAlignment.Center,

};

var btnMainAdd = new Button

{

Content = "Добавить Бухгалтера",

Width = 30,

Height = 30,

};

Style style = this.FindResource("addStyle") as Style;

btnMainAdd.Style = style;

btnMainAdd.Click += addBuh;

stackTest.Children.Add(lostlabel);

stackTest.Children.Add(btnMainAdd);

}

else

{

Base.Employee sEm1pTest = DataBase.Employee.First(U => U.ID\_employee == strucBuh.id\_Emp);

Base.Specializations specTest1 = DataBase.Specializations.First(U => U.id\_specialization == sEm1pTest.ID\_specialization);

var label1 = new Label

{

Content = sEm1pTest.ФИО + " - " + specTest1.Name\_specialization,

FontSize = 40,

HorizontalAlignment = HorizontalAlignment.Center,

};

var btn = new Button

{

Content = "Удалить",

Width = 30,

Height = 30,

VerticalAlignment = VerticalAlignment.Center,

};

Style style = this.FindResource("testStyle") as Style;

btn.Style = style;

btn.Click += delBuh;

stackTest.Children.Add(label1);

stackTest.Children.Add(btn);

}

var rect = new Rectangle

{

Height = 1,

};

rect.Fill = new SolidColorBrush(System.Windows.Media.Colors.Black);

stackTest.Children.Add(rect);

var stackHor = new StackPanel

{

Orientation = Orientation.Vertical

};

stackTest.Children.Add(stackHor);

foreach (Base.Departments department in DataBase.Departments)

{

if (department.Название != "Отдел кадров")

{

depIds.Add(department.ID\_department);

var stackPanelAdd = new StackPanel

{

};

var namelabel = new Label

{

Content = department.Название,

FontSize = 25,

FontWeight = FontWeights.Heavy,

HorizontalAlignment = HorizontalAlignment.Center,

};

stackPanelAdd.Children.Add(namelabel);

Base.Structure strMain = DataBase.Structure.SingleOrDefault(U => U.roleo == "mainDep\_" + department.ID\_department.ToString());

if (strMain != null)

{

Base.Employee empMain = DataBase.Employee.SingleOrDefault(U => U.ID\_employee == strMain.id\_Emp);

var labeldepMain = new Label

{

Content = empMain.ФИО + " - Начальник",

FontSize = 20,

HorizontalAlignment = HorizontalAlignment.Center,

};

var btn = new Button

{

Style = null,

Content = "Удалить начальника отдела "+department.Название,

VerticalAlignment = VerticalAlignment.Center,

Width=30,

Height=30,

FontSize=1,

};

Style style = this.FindResource("testStyle") as Style;

btn.Style= style;

btn.Background = new ImageBrush(new BitmapImage(

new Uri("xmark.png", UriKind.Relative)));

btn.Click += delDepMain;

stackPanelAdd.Children.Add(labeldepMain);

stackPanelAdd.Children.Add(btn);

}

else

{

var lostlabel = new Label

{

Content = "Не назначен руководитель отдела",

FontSize = 20,

HorizontalAlignment = HorizontalAlignment.Center,

};

var btnMainAdd = new Button

{

Content = "Добавить руководителя в отдел "+department.Название,

Width = 30,

Height = 30,

};

Style style = this.FindResource("addStyle") as Style;

btnMainAdd.Style= style;

if (depNum == 0)

{

depNum = department.ID\_department;

}

btnMainAdd.Name = "";

btnMainAdd.Click += addDepMain;

stackPanelAdd.Children.Add(lostlabel);

stackPanelAdd.Children.Add(btnMainAdd);

}

Base.Structure strZam = DataBase.Structure.SingleOrDefault(U => U.roleo == "zamDep\_" + department.ID\_department.ToString());

if (strZam != null)

{

Base.Employee empZam = DataBase.Employee.SingleOrDefault(U => U.ID\_employee == strZam.id\_Emp);

var labelDepZam = new Label

{

Content = empZam.ФИО + " - Зам. начальника",

FontSize = 20,

HorizontalAlignment = HorizontalAlignment.Center,

};

var btn = new Button

{

Content = "Удалить зама начальника отдела "+department.Название,

VerticalAlignment = VerticalAlignment.Center,

Width = 30,

Height = 30,

};

Style style = this.FindResource("testStyle") as Style;

btn.Style = style;

btn.Click += delDepZam;

stackPanelAdd.Children.Add(labelDepZam);

stackPanelAdd.Children.Add(btn);

}

else

{

var lostlabel = new Label

{

Content = "Не назначен зам руководитель отдела",

FontSize = 20,

HorizontalAlignment = HorizontalAlignment.Center,

};

var btnMainAdd = new Button

{

Content = "Добавить зама руководителя отдела "+department.Название,

Width = 30,

Height = 30,

};

Style style = this.FindResource("addStyle") as Style;

btnMainAdd.Style = style;

if (depNum == 0)

{

depNum = department.ID\_department;

}

btnMainAdd.Click += addDepZam;

stackPanelAdd.Children.Add(lostlabel);

stackPanelAdd.Children.Add(btnMainAdd);

}

var rect1 = new Rectangle

{

Height = 1,

};

rect1.Margin = new Thickness(5);

rect1.Fill = new SolidColorBrush(System.Windows.Media.Colors.Black);

stackPanelAdd.Children.Add(rect1);

stackHor.Children.Add(stackPanelAdd);

}

}

}

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

{

throw new NotImplementedException();

}

public void createNewStructure()

{

}

private void delDepZam(object sender, RoutedEventArgs e)

{

try

{

Button button = sender as Button;

string content = button.Content.ToString();

content = content.Substring(31);

Base.Departments dep = DataBase.Departments.Single(U => U.Название == content);

DataBase.Configuration.ValidateOnSaveEnabled = false;

Base.Structure str = DataBase.Structure.Single(U => U.roleo == "zamDep\_" + dep.ID\_department);

DataBase.Entry(str).State = System.Data.Entity.EntityState.Deleted;

DataBase.SaveChanges();

loadStructure();

}

finally

{

DataBase.Configuration.ValidateOnSaveEnabled = true;

}

}

private void delDepMain(object sender, RoutedEventArgs e)

{

try

{

Button button = sender as Button;

string content = button.Content.ToString();

content = content.Substring(26);

Base.Departments dep = DataBase.Departments.Single(U => U.Название == content);

DataBase.Configuration.ValidateOnSaveEnabled = false;

Base.Structure str = DataBase.Structure.Single(U => U.roleo == "mainDep\_" + dep.ID\_department);

DataBase.Entry(str).State = System.Data.Entity.EntityState.Deleted;

DataBase.SaveChanges();

loadStructure();

}

finally

{

DataBase.Configuration.ValidateOnSaveEnabled = true;

}

depNum = 0;

}

private void delBuh(object sender, RoutedEventArgs e)

{

try

{

DataBase.Configuration.ValidateOnSaveEnabled = false;

Base.Structure str = DataBase.Structure.Single(U => U.roleo == "buh");

DataBase.Entry(str).State = System.Data.Entity.EntityState.Deleted;

DataBase.SaveChanges();

loadStructure();

}

finally

{

DataBase.Configuration.ValidateOnSaveEnabled = true;

}

depNum = 0;

}

private void delMain(object sender, RoutedEventArgs e)

{

try

{

DataBase.Configuration.ValidateOnSaveEnabled = false;

Base.Structure str = DataBase.Structure.Single(U => U.roleo == "mainman");

DataBase.Entry(str).State = System.Data.Entity.EntityState.Deleted;

DataBase.SaveChanges();

loadStructure();

}

finally

{

DataBase.Configuration.ValidateOnSaveEnabled = true;

}

}

private void addBuh(object sender, RoutedEventArgs e)

{

addToStructure window = new addToStructure("buh", 0, this);

window.ShowDialog();

}

private void addMain(object sender, RoutedEventArgs e)

{

addToStructure window = new addToStructure("mainman", 0, this);

window.ShowDialog();

}

private void addDepMain(object sender, RoutedEventArgs e)

{

Button button = sender as Button;

string content = button.Content.ToString();

content = content.Substring(30);

Base.Departments depint = DataBase.Departments.Single(U => U.Название == content);

addToStructure window = new addToStructure("mainDep\_", depint.ID\_department, this);

depNum = 0;

window.ShowDialog();

}

private void addDepZam(object sender, RoutedEventArgs e)

{

Button button = sender as Button;

string content = button.Content.ToString();

content= content.Substring(34);

Base.Departments depInt = DataBase.Departments.Single(U => U.Название == content);

addToStructure window = new addToStructure("zamDep\_", depInt.ID\_department, this);

depNum = 0;

window.ShowDialog();

}

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

{

NavigationService.GoBack();

}

private void mouseEnter(object sender, MouseEventArgs e)

{

Button button = sender as Button;

button.Foreground = new ImageBrush(new BitmapImage(

new Uri("xmark.png", UriKind.Relative)));

}

private void mouseLeave(object sender, MouseEventArgs e)

{

Button button = sender as Button;

button.Foreground = new ImageBrush(new BitmapImage(

new Uri("xmark.png", UriKind.Relative)));

}

}

}

Программный код класса SourceCore

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using WorkPlan.Base;

namespace WorkPlan

{

class SourceCore

{

public static Entities MyBase = new Entities();

public static int UserID = new int();

}

}