

титул

Содержание

Введение	3
Глава 1 Теоретические и технические основы	5
1.1 Характеристика информационной системы на предприятии	5
1.2 Анализ предметной области	6
1.3 Постановка задачи	7
1.4 Выбор аппаратных и программных средств	8
1.4.1 C# .NET	8
1.4.2 MariaDB	9
1.4.3 WPF .Net Core	10
Глава 2 Проектирование программного продукта	12
2.1 Разработка структуры и состава ИС	12
2.2 Проектирование базы данных и обработки данных	16
2.3 Проектирование классов системы	20
Глава 3 Описание процесса реализации	22
3.1 Размещение базы данных	22
3.2 Подход к реализации интерфейса	25
3.3 Реализация интерфейса приложения	27
3.4 Вывод в EXCEL	38
3.5 Тестирование программного продукта	38
3.6 Итоговая структура проекта	40
Заключение	43
Список используемой литературы	45
Приложение	47
Приложение А Функция заполнения списка заявок	47
Приложение Б Функция заполнения списка товаров	48
Приложение В Функция заполнения списка добавленных услуг	49
Приложение Г Функция заполнения формы заявки	50

Введение

На текущий день рынок наполнен различными малыми предприятиями, каждое из которых старается укрепить экономическое состояние своего бизнеса с применением всевозможных тактик. В ход идут самые разнообразные способы: от налоговой оптимизации и грамотной политики учёта ресурсов до широких рекламных акций и привлечения экспертов по развитию предприятия.

Динамика развития малого бизнеса зависит от внутренней организации и готовности к увеличению объемов производства также, как и внешний облик предприятия, объемы спроса и предложения на рынке. Часто именно недостаток структурированной информационной системы является причиной неэффективного производства и в дальнейшем экономического застоя малых фирм.

Примером стартапа выступает некий индивидуальный предприниматель, занимающийся сбором клиентских заказов и изготовлением продукции на дому. Продуктом производства являются различные декоративные изделия, изготовленные на лазерном станке для резки. Заказчик имеет возможность выбора габаритов, вариантов последующей обработки и окраски изделия, а также материалов, из которых будут изготовлены элементы или изделие целиком.

Принять заказ, изготовить продукт и передать заказчику, - казалось бы, простая схема, однако, уже с небольшим увеличением объема заявок, предприятие начинает буквально задыхаться без надлежащего менеджмента рабочих процессов, коими в данном случае являются: мониторинг процессов изготовления продукции по каждой заявке и удобный учёт необходимых для работы объектов производства, а также возможность цифровизации этих процессов.

Первоочередная необходимость для такого предприятия - встраивание в работу часто не большой, но эффективной информационной системы, которая позволила бы как хранить данные, так и автоматизировать процесс обработки и получения необходимой информации и в целом оптимизировать процесс обслуживания клиентов, сделав его наглядным и структурированным.

Целями создания такой информационной системы являются организация эффективного процесса обработки заявок с применением компьютерных технологий, мониторинг по всем этапам исполнения заявок, а также автоматизация процессов учёта и хранения клиентской информации в базе данных.

Для реализации целей разработки информационной системы необходимо выполнение следующих основных задач:

- спроектировать и реализовать хранение и обработку данных средствами системы управления реляционными базами данных;
- реализовать возможность просмотра, редактирования клиентский данных, а также внутренних данных предприятия;
- разработать наглядный и оптимальный интерфейс, поддерживающий одновременную работу с несколькими формами;
- организовать необходимое тестирование прототипа программного продукта.

В работе выделяются теоретическая и практическая части. В теоретической части рассматривается и анализируется предметная область, определяются цели и задачи, а также описываются инструменты, используемые при разработке системы.

Практическая часть разделена на 2 главы:

- в главе “проектирование” рассматривается проектирование потоков данных, базы данных и классов системы;
- в главе “реализация” рассматриваются вопросы реализации программного продукта, практического выполнения задач для достижения поставленных целей.

Глава 1 Теоретические и технические основы

1.1 Характеристика информационной системы на предприятии

Система, в общем смысле слова, является такой совокупностью ресурсов, объектов или элементов совершенно произвольной природы, которые являются взаимосвязанными и организуют некую целостность, выполняют определённые задачи. Системы различаются между собой не только по составу, но и по направленности функционала.

Добавление к понятию «система» слова «информационная» отражает цель ее создания и функционирования.

Информация — это сведения об явлениях, объектах, событиях окружающего мира, уменьшающие неопределенность знаний о каждом из них. Эти знания отражают некую объективную, либо же субъективную действительность в сознании человека. Настоящая и полезная информация должна быть полной, непротиворечивой, достоверной, своевременной и здоровой.

Информационная система представляет собой такую совокупность техник и средств для хранения, обработки и передачи информации с использованием информационных технологий, направленную на достижение поставленных целей, в частности, на автоматизацию некоторых производственных процессов.

На текущий день информационные системы используются в качестве основного посредника и технического средства для обработки информации с использованием компьютера. Также существует следующая необходимость: техническое воплощение информационной системы само по себе ничего не значит без учёта роли человека в процессах работы информационной системы, ведь именно для него предназначена хранимая и производимая информация и естественно без человека невозможно ее получение и представление.

Также необходимо понимать разницу между компьютером и информационной системой. Компьютер оснащен специализированными программными средствами, которые являются технической базой и средством выполнения необходимых задач для информационных систем. Сама информационная система в свою очередь невозможна без обслуживающего персонала, взаимодействующего с компьютерами и телекоммуникациями.

В нормативно-правовом смысле *информационная система* определяется как «организационно упорядоченная совокупность документов (массив документов) и

информационных технологий, в том числе и с использованием средств вычислительной техники и связи, реализующих информационные процессы».

1.2 Анализ предметной области

Наименование предметной области:

Управление производственной деятельностью малого предприятия, учет клиентских заявок, мониторинг процессов исполнения заявок.

Общее описание:

Существует малое предприятие, занимающееся изготовлением и сбытом изделий, изготовленных из дерева и металла посредством лазерной резки, что позволяет получить высокую точность, скорость и дешевизну изготовления изделий, хотя и накладывает некоторые ограничения.

Клиент имеет возможность ознакомиться с предлагаемой предприятием продукцией на веб-сайте и в социальной сети компании. Клиенту представлены изображения, чертежи макетов, описание, возможные материалы для производства, а также цена производства выбранного продукта.

Для оформления заказа клиент может связаться с сотрудником по работе с клиентами посредством социальных сетей, либо по контактному телефону компании. Клиент выбирает изделие, материал и дополнительную обработку изделия, а также определяет количество каждого изделия для изготовления.

Предприятие проверяет возможность изготовления выбранных клиентом изделий, подготавливает макеты изделий и в дальнейшем занимается производством продукции по заказу клиента.

В процессе производства предприятие отслеживает изготовление каждого пункта в заказе и перепроверяет каждое действие для предотвращения ошибок и неточностей в размерах и прочих параметрах произведенных изделий. Также предприятие отслеживает качество и количество производственных материалов на складе.

По завершению производства по всем пунктам заказа, производятся конечные проверки и согласования с клиентом.

Для клиента распечатывается краткий отчет, составленный в таблицах Excel, содержащий в себе краткую информацию по заказу.

Завершающий этап обработки заявки - передача клиенту изготовленной продукции. С клиентом согласовываются время и способ доставки.

По итогу передачи продукции клиенту, в журнал учета вносится отметка о завершении заказа и выставляется дата закрытия заявки.

1.3 Постановка задачи

Цель разработки информационной системы:

Организация и автоматизация эффективного и структурированного процесса обработки заявок. Мониторинг процессов изготовления и исполнения заявок. Мониторинг процессов производства по каждому пункту в заявке. Автоматизация учета и структурированного хранения клиентской информации в базе данных.

Перечень процессов, описывающих деятельность предприятия:

Заказчик связывается с сотрудником по работе с клиентами по установленным каналам связи. Заказчиком выставляются необходимые требования по изготовлению изделий и их обработке (из предложенных вариантов на сайте компании, исключение: собственный чертеж). Сотрудник предприятия записывает данные заказчика в журнал учета (полное имя, контактные данные) и оформляет заявку.

В заявку клиента входят:

- полное имя заказчика;
- дата оформления заявки, сроки исполнения (если потребуется);
- заказанные клиентом изделия;
- дополнительные пожелания по обработке изделий (необязательная опция);
- количество изделий.

Изделия обладают следующими свойствами:

- наименование изделия;
- тип изделия;
- доступный материал для изготовления;
- краткое описание;
- цена без учёта материала и дополнительных работ.

Добавленные же в заказ изделия обладают следующим списком дополнительных свойств:

- количество на отдельный заказ;
- выбранный клиентом материал;
- общая цена по количеству и материалам;
- дополнительная обработка изделия.

В процессе анализа предметной области и подготовки к разработке системы, были выявлены следующие сущности:

- Заказчик;
- Сотрудник по работе с клиентами;
- Мастер;
- Заказ;
- Изделие;
- Вариант обработки изделия;
- Материал.

Основные функции информационной системы предприятия, определенные по итогу анализа предприятия и определения требований:

- эффективное и наглядное представление рабочей информации;
- запуск в работу новых клиентских заявок;
- контроль за выполнением активных заявок;
- предварительный расчет фактической стоимости заказа клиента по каждому пункту;
- вывод и печать отчета по заявке клиента в Microsoft Excel.

Перечень задач, выполнение которых позволит добиться поставленных целей:

- спроектировать и реализовать хранение и обработку данных средствами системы управления реляционными базами данных;
- реализовать возможность просмотра, редактирования клиентских данных, а также внутренних данных предприятия;
- разработать наглядный и оптимальный интерфейс, поддерживающий одновременную работу с несколькими формами;
- организовать необходимое тестирование прототипа программного продукта.

1.4 Выбор аппаратных и программных средств

1.4.1 C# .NET

Для реализации логики приложения был выбран язык программирования высокого уровня C# (Си-шарп) на программной платформе .NET Core.

Язык C# является объектно-ориентированным, а следовательно, предлагает следующие принципы и возможности:

- принципы наследования классов, вместо композиции;

- инкапсуляция кода;
- полиморфизм;
- программирование под интерфейсы;
- принцип делегирования в другие классы;
- принцип и вероятная необходимость использования абстракции;
- модульность.

Объектно-ориентированный подход позволяет писать гибкий и связный код с низким зацеплением.

Низкое зацепление означает, что функциональные блоки кода будут мало связаны друг с другом, что обеспечивает больше возможности для безопасного редактирования отдельных блоков кода.

Также для реализации необходимого функционала и оптимизации процесса разработки были использованы некоторые библиотеки для платформы C# .NET Core.

При помощи средства управления пакетами NuGet в проект были загружены необходимые библиотеки:

- библиотека “MySQL.Data” - набор классов и методов для взаимосвязи между приложением на базе .NET и базами данных MySQL и MariaDB;
- библиотека “PropertyChanged.Fody” - ответвление от большого пакета методов Fody, позволяющее отслеживать изменение значений свойств классов и оповещать об этом привязанное к свойствам представление, изменяя параметры элементов языка разметки XAML;
- библиотека “Material Design by James Willock”

1.4.2 MariaDB

Для организации и хранения данных в информационной базе было выбрано ответвление от системы управления базами данных “MySQL”, под названием “MariaDB”.

MariaDB представляет собой СУБД под лицензией “GNU General Public Licence”, дающей право пользователю на свободное копирование, модификацию и распространение программного обеспечения.

Являясь программным продуктом от разработчиков MySQL, MariaDB поддерживает высокий уровень совместимости с MySQL, а также обеспечивает полное соответствие командам и API платформы MySQL.

Данная СУБД позволяет эффективно выполнить все поставленные задачи проекта и обладает следующими преимуществами перед СУБД реляционного типа:

- релизы обновлений программы и безопасности синхронизированы, что позволяет избегать и быстро исправлять ошибки, а также все исправления задокументированы в удобном для пользователя варианте;
- в целом, MariaDB развивается быстрее и имеет больше возможностей, чем, например, конкурент MySQL. Предоставляет больше возможностей оптимизации и работы с памятью, а также имеет более высокую производительность;
- СУБД MariaDB поддерживает серьезно больше движков хранения данных и все они включены в официальный релиз программы и надежно интегрированы, в отличие от, например, MySQL и Microsoft SQL, где большинство таких движков доступны лишь в качестве плагинов, что, конечно, отрицательно сказывается на производительности;
- косвенно упомянутое ранее, но не менее важное преимущество - MariaDB разрабатывается полностью открыто, все решения по разработке и новые идеи возможно легко обсуждать в системе обмена сообщениями об ошибках. Разработчики активно сотрудничают с пользователями, проявляющими интерес к улучшению платформы и предлагающими новые идеи и правки системы.

Так как MariaDB не имеет собственного клиента (приложения), было принято решение использовать в качестве клиента для разработки и администрирования баз данных “HeidiSQL”.

1.4.3 WPF .Net Core

Для визуализации необходимых функций была выбрана система для построения пользовательских приложений Windows Presentation Foundation (WPF), базирующаяся на технологии .NET (Microsoft).

Главная особенность WPF - весь интерфейс описывается на языке XAML, подмножестве XML, в основе которого лежат технологии HTML и CSS. Весь внешний вид элементов окна можно описывать отдельными стилями и легко изменять простой заменой файла со стилями.

Также несомненным плюсом, который в современном мире стал негласным стандартным, является использование технологии DirectX, предоставляющей серьезный прирост производительности при помощи аппаратного ускорения.

Элементы интерфейса в WPF легко подстраиваются под разные разрешения, а механизм привязки данных позволяет ускорить процесс разработки и сделать код более читаемым, как, например, при использовании шаблона проектирования приложений Model-View-ViewModel (в дальнейшем - MVVM), задействованного в данном проекте и описанного далее.

Глава 2 Проектирование программного продукта

2.1 Разработка структуры и состава ИС

Модели и схемы необходимой информационной системы разработаны в программе “AllFusion Process Modeler r7”, позволяющей создавать и редактировать различные нотации, такие как, IDEF0, DFD и IDEF3.

Графическая нотация IDEF0 представляет собой методологию функционального моделирования, необходимую для формализации и описания бизнес-процессов, а также процессов в информационных системах. На рисунке 1 представлено общее описание функционального блока контекстной диаграммы.

Выделяются следующие виды стрелок: "Вход", "Выход", "Механизм", "Управление". Входы преобразуются процессом, для создания того, что по итогу окажется на выходе. Управления определяют условия, необходимые процессу, чтобы произвести правильный выход. Выходы - это данные или материальные объекты, произведенные процессом и используемые в дальнейшем цикле функционирования. Механизмы идентифицируют средства, поддерживающие выполнение процесса. Блок IDEF0 показывает преобразование входа в выход с помощью механизмов с учетом управляющих воздействий.

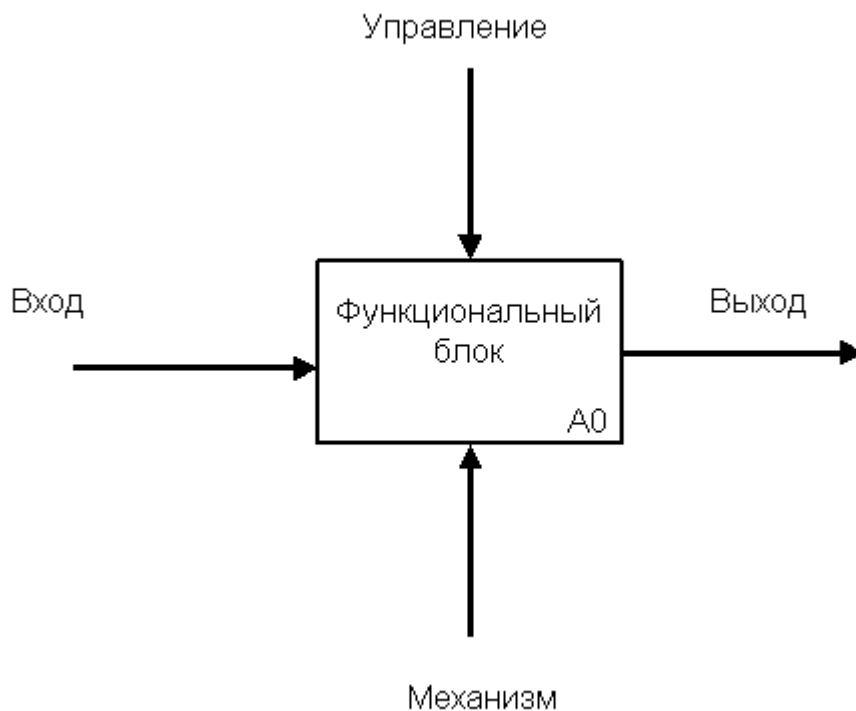


Рисунок 2.1.1 – Общий вид функционального блока IDEF0

На схеме 1 представлена базовая IDEF0-диаграмма “Описание процесса обработки заказа”, раскрытая в последующих схемах. На схеме представлен основной блок “Обработка заказа”, а также все необходимые для работы системы рабочие потоки.

Входами блока являются:

- основные данные клиента;
- исходная база данных;
- склад материалов;
- данные по заказу клиента.

Выходами блока являются:

- измененная база данных;
- отчет по заказу;
- получение клиентом заказа;

Механизмами для работы блока являются сотрудник по работе с клиентами и мастера.

Управляющими потоками блока являются каталог компании, чертежи и список расценок компании.

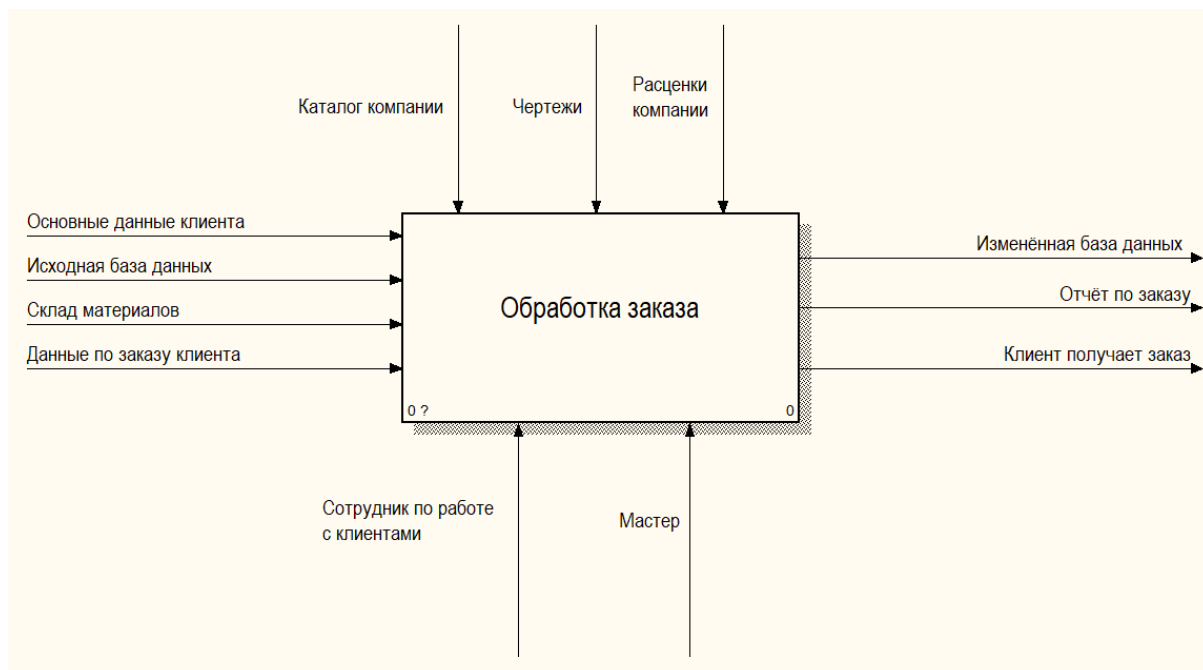


Рисунок 2.1.2 – Базовая IDEF0-диаграмма “Описания процесса обработки заказа”

На рисунке 2.1.3 представлена раскрытая диаграмма IDEF0, ранее изображенная на рисунке 2.1.2 и, соответственно, имеющая одинаковые рабочие потоки.

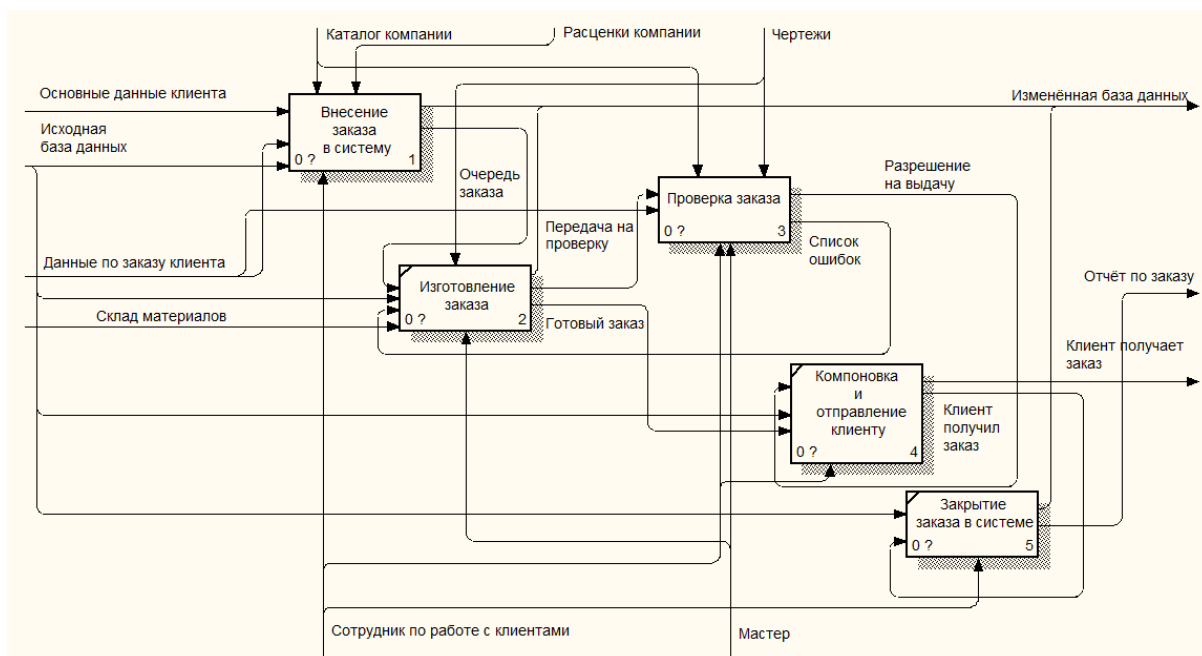


Рисунок 2.1.3 – Подробная IDEF0-диаграмма “Описания процесса обработки заказа”

На рисунке 2.1.3 изображены 5 последовательных блоков нотации:

- внесение заказа в систему: в данном блоке происходит внесение сотрудником по работе с клиентом заказа в систему (оформление заявки), передача очереди заказа на блок изготовления и изменение базы данных соответственно внесенной заявке;
- изготовление заказа: в данном блоке происходит итеративное выполнение процесса изготовления продукции по заявкам в системе. По завершении итерации происходит передача заказа в блок проверки заказа. Если итерации закончились и отсутствуют ошибки по результатам проверки заказа - изготовленная продукция передается на компоновку и отправление клиенту;
- проверка заказа: в данном блоке происходит проверка полученной продукции на каждой итерации изготовления по заявке клиента. При несоответствии полученной продукции требованиям клиента - продукт передается на доработку со списком ошибок. На последней итерации, если не найдены ошибки, передается разрешение на выдачу заказа клиенту;
- компоновка и отправление клиенту: в данном блоке происходит компоновка заказа и передача его клиенту на заранее оговоренных условиях. По завершении данного процесса информация об этом передается в блок закрытия заявки в системе;
- закрытие заказа в системе: в данном блоке происходит окончательное закрытие заявки, присвоение даты и статуса закрытия с последующим внесением данных в базу данных и печать отчета по заказу, при желании.

На рисунке 2.1.4 изображена DFD-диаграмма, раскрывающая блок “Внесение заказа в систему”, изображенный ранее на рисунке 2.1.3 и подробно описывающая процессы внесения заказа в базу данных.

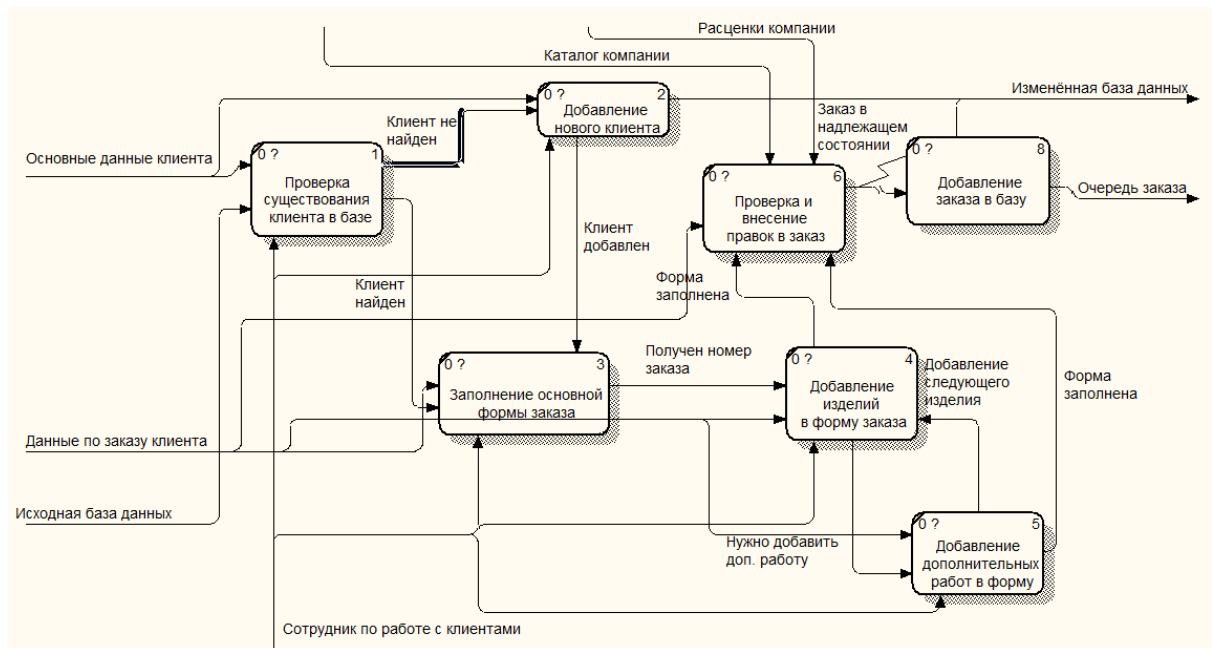


Рисунок 2.1.4 – DFD-диаграмма “Описание процесса внесения заказа в базу”

На рисунке 2.1.5 изображена DFD-диаграмма, раскрывающая блок “Проверка заказа”, изображенный ранее на рисунке 2.1.3, и подробно описывающая процессы проверки заказа на каждой итерации производства.

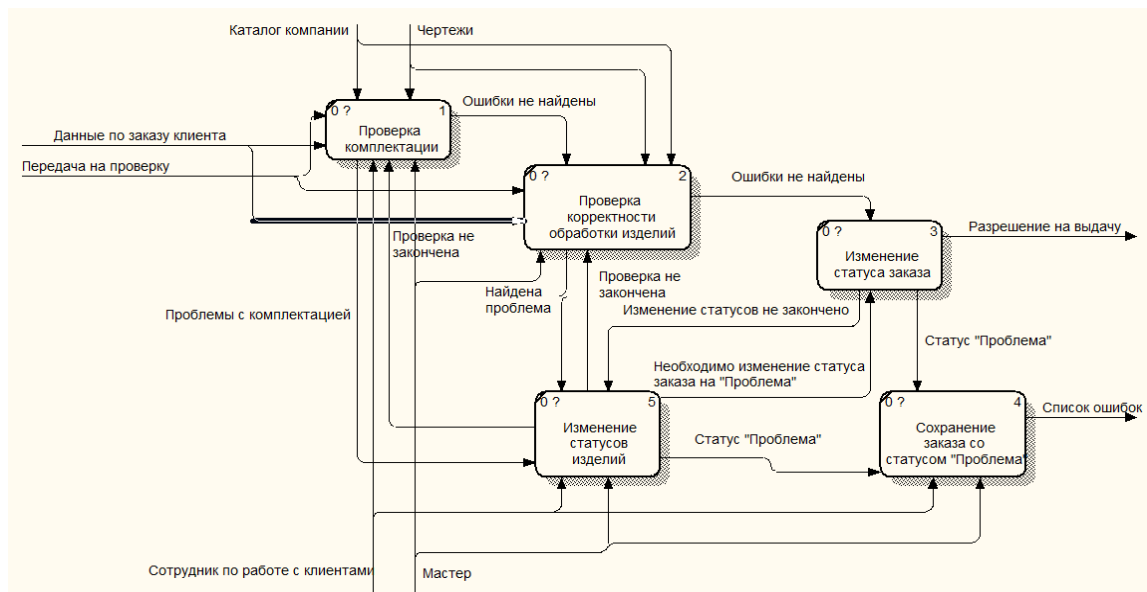


Рисунок 2.1.5 – DFD-диаграмма “Процесс проверки изделий на производстве”

2.2 Проектирование базы данных и обработки данных

Выявленные требования для базы данных в информационной системе:

- хранение клиентской информации, включающей в себя полное имя клиента, список контактов клиента, предоставленные клиентом индивидуальные чертежи или эскизы;
- хранение информации по остаткам материалов для изготовления на складе;

По итогам анализа выявленных требований спроектированы следующие объекты базы данных, перечисленные в таблице 1, таблице 2, таблице 3, а также составлена логическая модель базы данных (Рисунок 2.2.1).

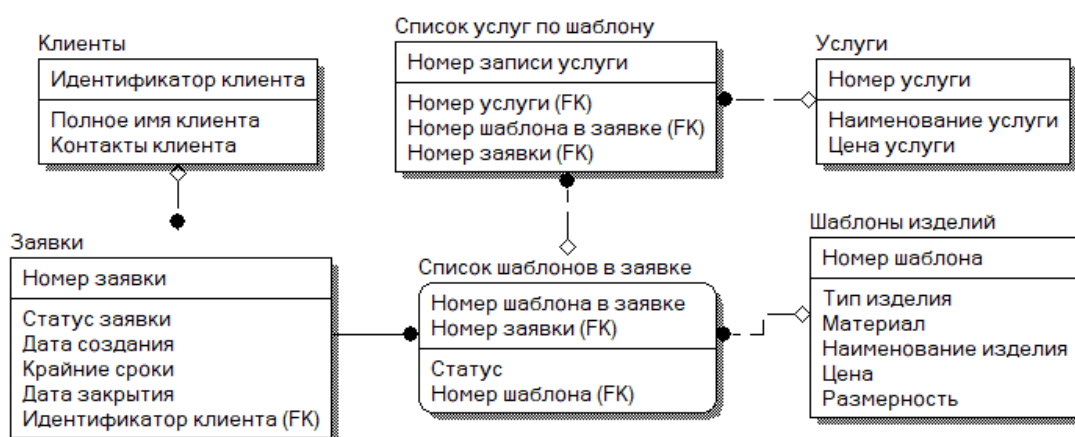


Рисунок 2.2.1 – Логическая модель базы данных

Во-первых, в Таблице 1 перечислены таблицы БД со своими атрибутами и описанием.

Таблица 1 – Таблицы базы данных

Наименование объекта БД	Параметры объекта		
	Поля объекта	Описание	Связи
Clients	id FIO	Информация о клиентах	-
Clientsowndrawing	id DrawingName description sizing	Предложенные клиентами чертежи	-

Продолжение Таблицы 1.

Contacts	id phone inst email clientcamefrom ClientID	Контакты клиентов	FK ClientID (id from Client)
Orders	id ClientID OrderStatus created deadline closed	Заявки клиентов	FK ClientID (id from Client), FK OrderStatus (id from Statustable)
OrderItems	id ItemID OrderID OwnDrawing description quantity	Добавленные в заявку изделия к изготовлению	FK ItemID (id from Items), FK OrderID(id from Orders), FK OwnDrawind(id from clientsowndrawings)
Items	id ItemTypeID ItemName MaterialID price sizing	Изделия для добавления в заявку	FK ItemTypeID (id from ItemTypes), FK MaterialID(id from Materials)
ItemTypes	id description	Типы изделий	-
Materials	id description price	Материалы для изготовления изделий	-
Additional	id description price	Дополнительные работы по изделию	-
Additionalonitem	id OrderItemsID AddID	Связь добавленного изделия и дополнительной работы над ним	FK OrderItemsID (id from OrderItems), FK AddID (id from Additional)
Statustable	id description	Таблица с возможными статусами	-

Продолжение Таблицы 1.

Delivering	id OrderID address dateofdelivery	Информация по доставке заказа	FK OrderID (id from Orders)
------------	--	-------------------------------	-----------------------------

В таблице 2 перечислены представления, используемые для гибкого доступа к данным информационной базы из непосредственно приложения.

Таблица 2 – Представления базы данных

Наименование объекта БД	Параметры объекта		
	Поля объекта	Описание	Используемые таблицы
view_added_additional	-id; -description; -price; -orderid;	Представление добавленных к изделиям дополнительных услуг	-additional; -additionalitem
view_itemsad	-id; -description;	Представление наименований типов изделий	-itemtypes; -items;
view_items_full	-id; -itemname; -description(type); -description(mat); -price; -sizing;	Представление полных параметров шаблонов изделий	-view_itemsad; -items; -materials;
view_materials	-id; -description; -price;	Представление материалов	-materials;
view_types	-id; -description;	Представление типов изделий	-types;
view_works	-id; -description; -price;	Представление дополнительных услуг	-additional;

Продолжение Таблицы 2.

view_orders	-id; -FIO; -description; -created; -deadline; -closed;	Представление списка заявок	-statustable; -orders; -clients;
-------------	---	--------------------------------	--

В таблице 3 перечислены процедуры информационной базы, используемые для доступа к данным, а также для внесения данных в базу.

Таблица 3 – Процедуры базы данных

Наименование объекта БД	Параметры объекта		
	Параметры	Описание	Выходные значения
proc_add_orderitem	-id изделия -id заявки	Добавление в базу данных изделия к заявке	В базу добавлено заказанное изделие
proc_add_orderitem _additional	-id добавленного изделия -id услуги	Добавление в базу данных услуги на изделие	В базу добавлена заказанная услуга
proc_client_by_fio	-ФИО клиента	Поиск клиента в базе данных по имени клиента	Возвращает идентификатор клиента и его заявок
proc_order_items	-id заявки	Вывод списка заказанных изделий в заявке	Возвращает таблицу добавленных изделий
proc_remove_orderitem	-id добавленного изделия	Удаление изделия из заявки	Из базы удалено изделие в заявке
proc_remove_orderitem _additional	-id добавленной услуги	Удаление услуги	Из базы удалена услуга

2.3 Проектирование классов системы

Для связи базы данных и приложения необходимы классы, с помощью которых можно управлять потоками данных в системе. Классы должны полностью отражать типы данных в базе данных.

По итогу проектирования были разработаны следующие классы:

- Additional: описывает данные о дополнительных услугах;
- AdditionalOnItem: описывает данные о дополнительных услугах, добавленных к товару;
- Type: описывает данные о типах изделий;
- Material: описывает данные о материалах;
- Item: описывает свойства шаблонов изделий;
- OrderItem: описывает свойства добавленного в заказ товара;
- OrderItemListView: описывает список добавленных в заказ товаров;
- Contacts: описывает контакты клиента;
- Client: описывает информацию о клиенте;
- Order: описывает все необходимые поля заявки;
- OrderView: описывает функции по редактированию заявки;
- ServiceSQL: описывает функции для доступа к базе данных.

На рисунке 2.3.1 представлены основные классы информационной системы, описывающие потоки данных. Реализованные в приложении с использованием данной диаграммы классы будут полностью отражать необходимые свойства объектов и необходимый функционал для работы с данными.

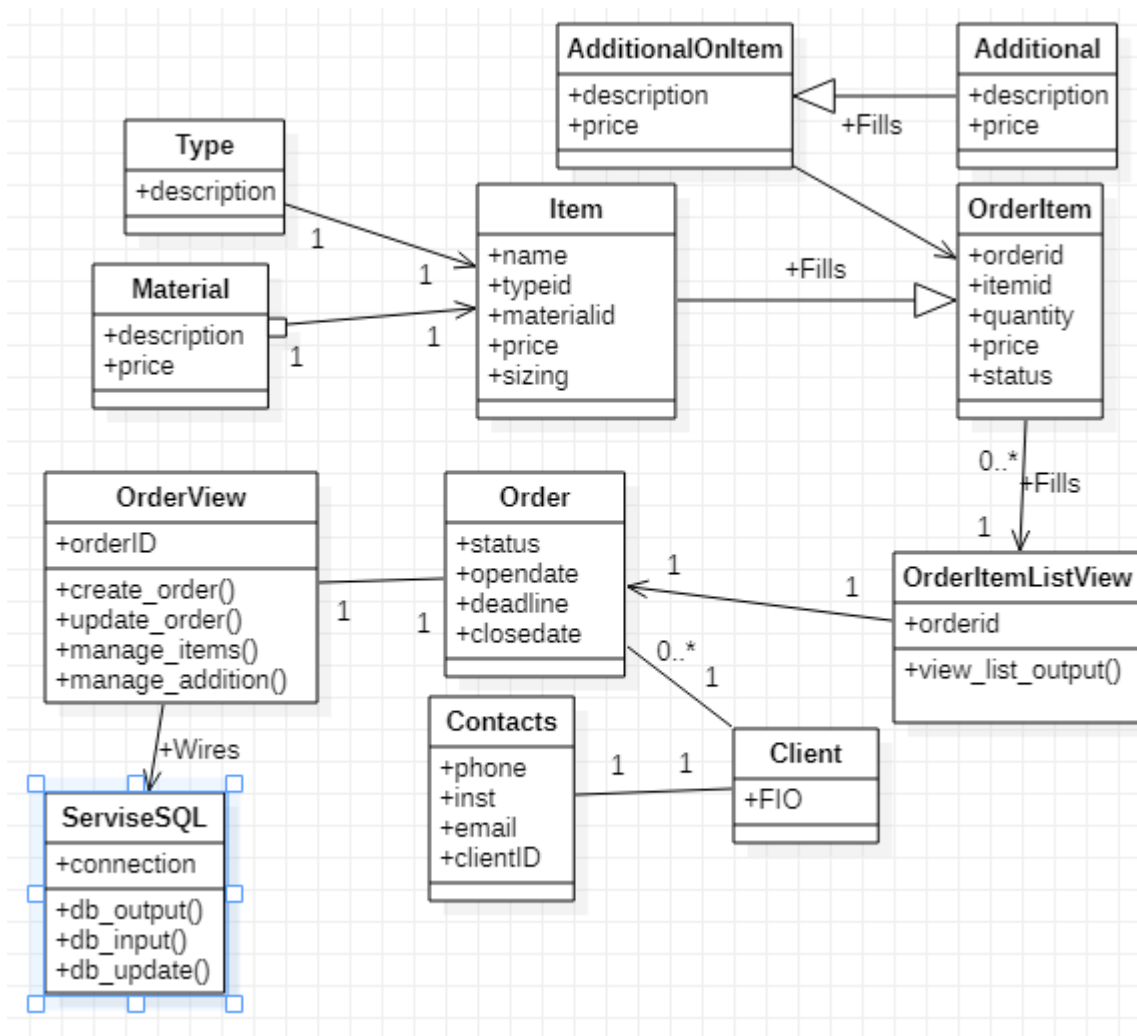


Рисунок 2.3.1 – Диаграмма классов

Работа с классами в приложении организована посредством паттерна MVVM. Классы типа ViewModel необходимы для непосредственной привязки данных к элементам интерфейса.

Класс “ServiseSQL” - специальный, используется для реализации связи между приложением и базой данных и всего функционала для вывода и изменения данных.

Глава 3 Описание процесса реализации

3.1 Размещение базы данных

С использованием СУБД MariaDB и клиента HeidiSQL была реализована необходимая база данных, физическая модель которой изображена на рисунке 3.1.1.

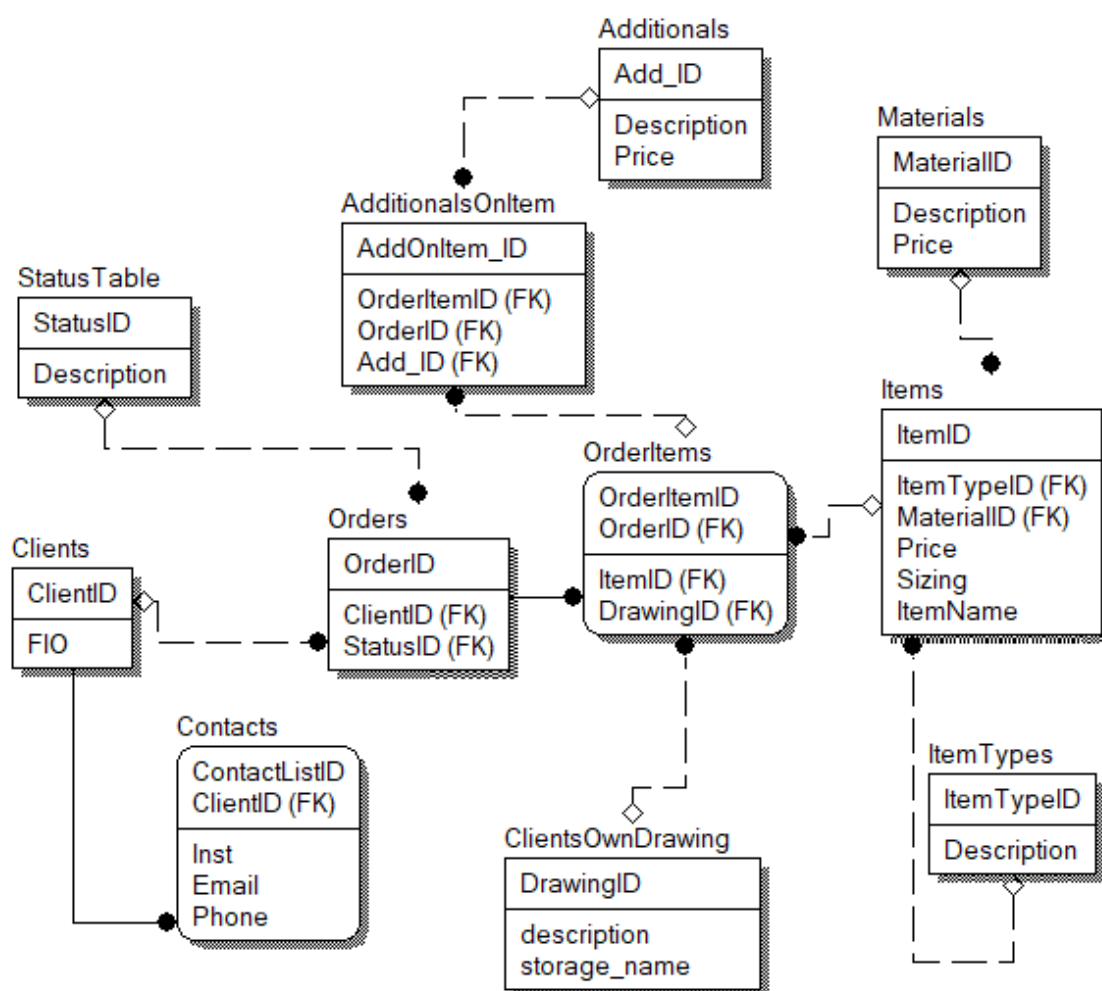


Рисунок 3.1.1 – Физическая модель базы данных

На рисунке 3.2.2 изображено дерево таблиц, полученное в результате реализации базы данных посредством клиента HeidiSQL.

accounts	0	16,0 KiB
additionalonit...	12	48,0 KiB
additional	8	16,0 KiB
clients	14	16,0 KiB
clientsowndra...	0	16,0 KiB
contacts	14	32,0 KiB
delivering	0	32,0 KiB
items	14	48,0 KiB
itemtypes	16	16,0 KiB
materials	7	16,0 KiB
OnInsertQuant...		
orderitems	41	64,0 KiB
orders	8	48,0 KiB

Рисунок 3.2.2 – Таблицы в базе данных

Примером создания таблицы может послужить следующий код в листинге 3.1.1, используемый для создания таблицы “orders” и установления внешних связей.

Листинг 3.1.1 - Создание таблицы средствами MariaDB

```
CREATE TABLE `orders` (
  `id` SMALLINT(5) UNSIGNED NOT NULL AUTO_INCREMENT,
  `ClientID` SMALLINT(5) UNSIGNED NULL DEFAULT NULL,
  `OrderStatus` SMALLINT(5) UNSIGNED NULL DEFAULT NULL,
  `created` DATE NULL DEFAULT NULL,
  `deadline` DATE NULL DEFAULT NULL,
  `closed` DATE NULL DEFAULT NULL,
  PRIMARY KEY (`id`) USING BTREE,
  INDEX `fk_order_client` (`ClientID`) USING BTREE,
  INDEX `fk_order_status` (`OrderStatus`) USING BTREE,
  CONSTRAINT `fk_order_client` FOREIGN KEY (`ClientID`) REFERENCES
`azarov1`.`clients` (`id`) ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT `fk_order_status` FOREIGN KEY (`OrderStatus`) REFERENCES
`azarov1`.`statustable` (`id`) ON UPDATE CASCADE ON DELETE CASCADE
)
COLLATE='utf32_unicode_ci'
ENGINE=InnoDB
AUTO_INCREMENT=25;
```

В результате использования приведенного выше программного кода, в базе данных будет создана таблица, физический вид которой, предоставляемый клиентом HeidiSQL, изображён на рисунке 3.2.3.

#	Имя	Тип данных	Длина/Знач...	Беззна...	Разрешить N...	Zerofill	По умолчанию
1	id	SMALLINT	5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREME..
2	ClientID	SMALLINT	5	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
3	OrderStatus	SMALLINT	5	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
4	created	DATE		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
5	deadline	DATE		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
6	closed	DATE		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL

Рисунок 3.2.3 – Таблица “orders”

Для доступа к данным используются представления. Примером представления является представление “view_orders”, реализующее отображение основных данных по заявкам. В листинге 3.1.2 представлен код создания представления.

Листинг 3.1.2 - Создание представления средствами MariaDB

```
SELECT a.id, b.FIO, s.description, a.created, a.deadline, a.closed
FROM azarov1.statustable s RIGHT JOIN azarov1.orders a
ON a.OrderStatus = s.id INNER JOIN azarov1.clients b ON a.ClientID = b.id
```

На рисунке 3.2.4 представлены данные все основные данные по заявке, полученные при помощи представления.

azarov1.view_orders					
id	FIO	description	created	deadline	closed
1	Зябкин И.Н.	5. Закрыт	2021-01-23	2021-04-14	2021-04-13
13	Игорь @igorpetrov	5. Закрыт	2021-05-22	(NULL)	2021-05-11
18	Михаил Федосов	1. Новый	2021-05-21	2021-05-30	(NULL)
19	Егоров Д.Г.	2. Обработан	2021-05-23	2021-06-22	(NULL)
20	Любимка П.А.	3. Готов	2021-05-24	(NULL)	(NULL)
21	Букин Г.К.	4. Проблема	2021-04-12	2021-05-22	(NULL)
22	Мордор С.А.	3. Готов	2021-05-25	2021-06-04	(NULL)
23	Иванов И.И.	1. Новый	2021-05-23	(NULL)	(NULL)
25	Панихидов А.А.	5. Закрыт	2021-04-30	2021-05-03	2021-05-30

Рисунок 3.2.4 – Данные из представления “view_orders”

Для редактирования данных используются хранимые процедуры. Примером процедуры добавления данных в базу является процедура, добавляющая товар к заявке, код которой представлен в листинге 3.1.3.

Листинг 3.1.3 – Процедура добавления товара в заявку

```
BEGIN
    INSERT INTO azarov1.orderitems(itemID, orderID)
    VALUES (itempar,orderpar);
END
```


3.2 Подход к реализации интерфейса

Model View ViewModel (MVVM) - это архитектурный шаблон, который представлен Microsoft и используется в разработке программного обеспечения, специализируется на шаблонном проектировании моделей и представлений. Он основан на шаблоне Model-view-controller (MVC) и ориентирован на современные платформы разработки пользовательского интерфейса (WPF и Silverlight), где разработчик UX имеет различные требования, чем более "традиционный" разработчик.

MVVM - это способ создания клиентских приложений, которые используют основные функции платформы WPF, позволяют производить простое модульное тестирование функциональности приложения и помогают разработчикам и дизайнерам работать с меньшими техническими проблемами.

Представление: представление (View) определяется в XAML и не должно иметь никакой жестко привязанной к нему логики. Оно подключается к модели представления только с помощью привязки данных.

Модель: модель отвечает за отображение данных таким образом, чтобы их можно было легко использовать в WPF. `INotifyPropertyChanged` и/или `INotifyCollectionChanged` должны применяться соответствующим образом для реализации автоматического оповещения приложения об изменении параметров и свойств классов.

VIEW MODEL: ViewModel - это модель для представления в приложении, или можно сказать, что это абстракция представления. Оно раскрывает данные, связанные с представлением, и обычно описывает поведение для представлений с помощью команд.

На рисунке 3.2.1 представлена структура паттерна.

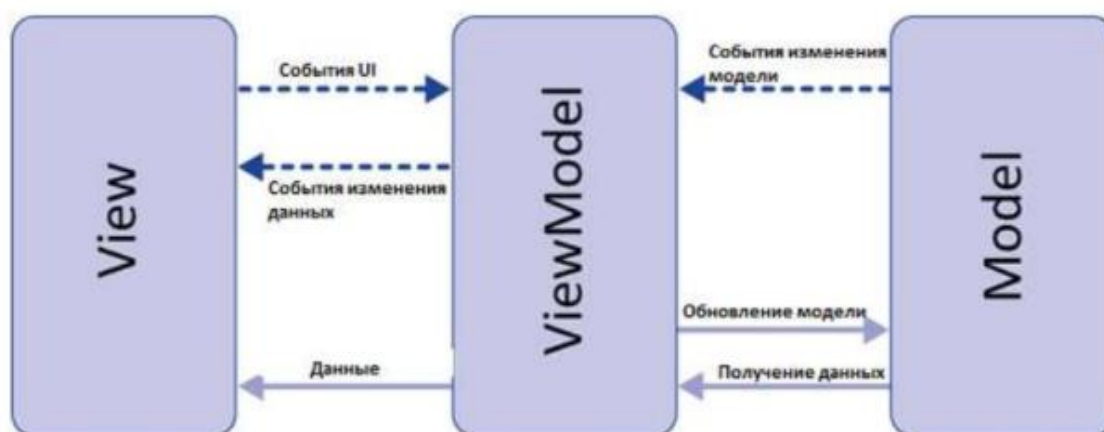


Рисунок 3.2.1 – Структура паттерна MVVM

Применение паттерна MVVM на примере функции заполнения таблицы заявок разрабатываемой системы:

- Создать представление (окно MainWindow.xaml);
- Создать ViewModel для представления с необходимым функционалом (класс MainClass);
- Создать модель - класс, описывающий структуру данных (класс Orders)
- Связать DataContext представления с объектом класса ViewModel.
- Осуществить привязку данных (Binding) таблицы (DataGrid) в xaml-файле.

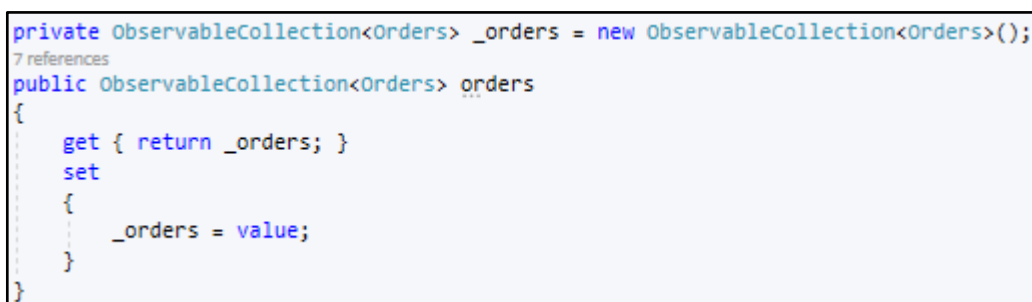
Для начала, необходимо описать данные, получаемые из базы данных. В данном случае это табличные данные, все поля которой необходимо описать в классе Orders (см. Рисунок 3.2.2). Также для отслеживания изменения данных по ходу заполнения необходимо подключить библиотеку INotifyPropertyChanged и интерфейс PropertyChanged для неё.



```
18 references
public class Orders : INotifyPropertyChanged
{
    //interface for fody inotify
    public event PropertyChangedEventHandler PropertyChanged = (sender, e) => { };
    private int _id;
    private string _FIO;
    private string _status;
    private string _created;
    private string _deadline;
    private string _closed;
```

Рисунок 3.2.2 – Класс, описывающий таблицу заявок

Привязку данных DataGrid необходимо совершать по отношению к списку, в данном случае - к списку объектов класса ObservableCollection. На рисунке продемонстрирована инициализация пустого списка. Доступ к списку необходимо обеспечить через свойство такого же типа с применением конструкции “get; set;” (см. Рисунок 3.2.3). Заполнение списка происходит через функцию заполнения (см. Приложение А), хранимую в отдельном классе, описывающем весь функционал для работы с базой данных.



```
private ObservableCollection<Orders> _orders = new ObservableCollection<Orders>();
7 references
public ObservableCollection<Orders> orders
{
    get { return _orders; }
    set
    {
        _orders = value;
    }
}
```

Рисунок 3.2.3 – Инициализация списка заявок для привязки данных

Далее необходимо изменить DataContext управляющего класса файла MainWindow.xaml на новый объект ViewModel класса, в данном случае - MainClass. Это необходимо сделать для того, чтобы названия свойств (контекст), с помощью которых будет выполнена привязка, интерфейс искал в свойствах объекта класса MainClass (см. Рисунок 3.2.4).

```
0 references
public MainWindow()
{
    InitializeComponent();
    this.DataContext = new MainClass();
}
```

Рисунок 3.2.4 – Назначение “DataContext” на объект ViewModel

Далее необходимо прописать непосредственно привязку в XAML-файла к элементу DataGrid при помощи свойства ItemsSource и функции Binding (см. Рисунок 3.2.5).

```
<!--Main DataGrid with Orders-->
<DataGrid Background="#OldLace" AutoGenerateColumns="False"
    ItemsSource="{Binding Path=orders}"
```

Рисунок 3.2.5 – Привязка “DataGrid” к списку объектов

Также необходимо установить свойство “AutoGenerateColumns” в состояние “False”, для того, чтобы отключить автоматическую генерацию столбцов с названиями свойств класса “Orders”. Вместо автоматической генерации, необходимо вручную прописать каждую колонну и привязать их к названиям свойств класса, а также задать заголовочные свойства “Header” в удобный читаемый вид (см. Рисунок 3.2.6).

```
<DataGrid.Columns>
    <DataGridTextColumn Header="ID" Binding="{Binding id}"/>
    <DataGridTextColumn Header="Полное имя" Binding="{Binding FIO}" Width="250"/>
    <DataGridTextColumn Header="Статус заявки" Binding="{Binding status}"/>
    <DataGridTextColumn Header="Дата создания" Binding="{Binding created}"/>
    <DataGridTextColumn Header="Сроки" Binding="{Binding deadline}" Width="130"/>
    <DataGridTextColumn Header="Дата закрытия" Binding="{Binding closed}"/>
</DataGrid.Columns>
```

Рисунок 3.2.6 – Привязка столбцов таблицы к свойствам объектов списка

3.3 Реализация интерфейса приложения

Главное окно приложения (см. Рисунок 3.3.1) располагается на основной форме проекта и предоставляет следующий функционал:

- просмотр существующих в базе заявок;
- просмотр шаблонов изделий, добавленных в конкретную заявку;
- редактирование статусов заявок;
- редактирование статусов изготовления по отдельным пунктам в заявке;

- поиск по заявкам;
- переход к формам редактирования сервисной информации.

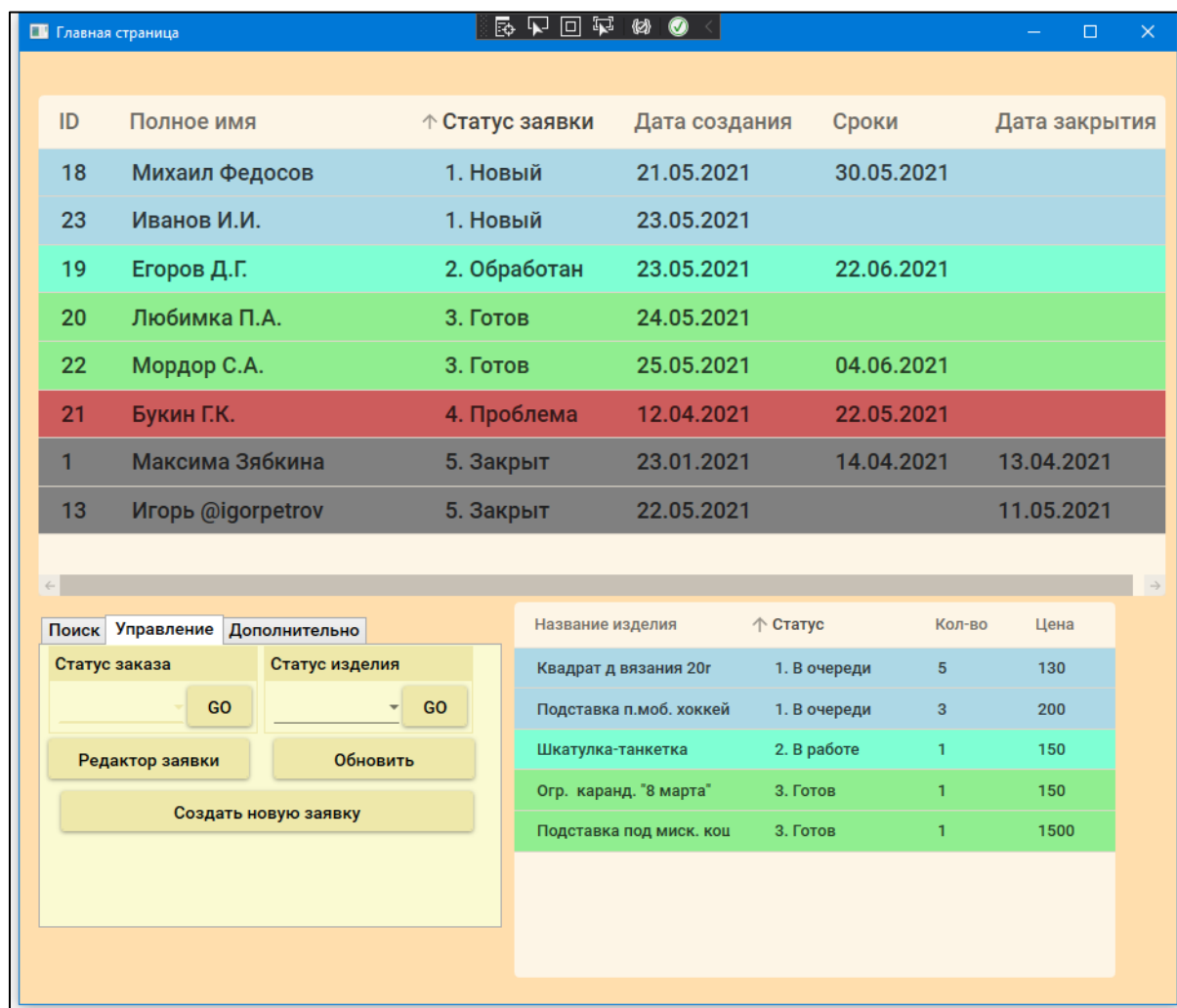


Рисунок 3.3.1 – Главное окно приложения

Основное пространство главного окна (см. Рисунок 3.3.1) занимает таблица заявок, внесенных в базу, функция заполнения в Приложении А.

Таблица имеет следующие отображаемые столбцы: ID (номер заявки), полное имя клиента, статус заявки, дата создания заявки, заданные клиентом сроки выполнения работы, дата закрытия заявки.

Возможна сортировка по алфавиту, либо по датам. Для этого необходимо нажать на заголовок столбца, по которому следует произвести сортировку.

Каждая заявка в таблице окрашивается собственным цветом соответственно статусу заявке, для удобства восприятия пользователем. Для каждого статуса выбраны следующие цвета:

- статус 1.Новый - код цвета #FFADD8E6;
- статус 2.Обработан - код цвета #FF7FFFD4;

- статус 3.Готов - код цвета #FF90EE90
- статус 4.Проблема - код цвета #FFCD5C5C;
- статус 5.Закрыт - код цвета #FF808080;
- без статуса - код цвета #FFD3D3D3;

Каждый цвет был выбран для каждого статуса исходя из естественного эмоционального восприятия цвета человеком.

Для изменения текущего статуса заявки используется выпадающий список (см. Рисунок 3.3.2) с возможностью выбора каждого статуса.

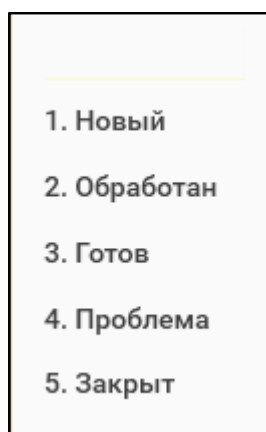


Рисунок 3.3.2 – Выпадающий список статусов заявок

После выбора нового статуса необходимо нажать на кнопку справа от списка для передачи команды системе и для внесения изменений в базу данных. Перед изменением статуса необходимо одиночным нажатием левой кнопки мыши выделить необходимую заявку в таблице заявок. Кнопка подтверждения и выпадающий список находятся во вкладке “Управление” на панели вкладок в левом нижнем углу окна (см. Рисунок 3.3.1). При нажатии кнопки подтверждения, пользователю будет выведено диалоговое окно, требующее необходимость повторного подтверждения действия.

Также при выборе заявки в таблице заявок, в таблице добавленных изделий, расположенной в правом нижнем углу окна, будут отображены добавленные в заявку товары к изготовлению (см. Рисунок 3.3.3), код функции в Приложении Б.

Название изделия	↑ Статус	Кол-во	Цена
Квадрат д вязания 20г	1. В очереди	5	130
Подставка п.моб. хоккей	1. В очереди	3	200
Шкатулка-танкетка	2. В работе	1	150
Огр. каранда. "8 марта"	3. Готов	1	150
Подставка под миск. кош	4. Проблема	1	1500

Рисунок 3.3.3 – Таблица добавленных в заявку товаров

Каждый пункт в таблице товаров имеет следующие поля: название изделия, статус изделия, количество изделий в пункте, цена.

Каждый пункт в таблице окрашивается в цвет, выбранный соответственно статусу:

- статус 1.В очереди - код цвета #FFADD8E6;
- статус 2.В работе - код цвета #FF7FFFD4;
- статус 3.Готов - код цвета #FF90EE90
- статус 4.Проблема - код цвета #FFCD5C5C;
- без статуса - код цвета #FFD3D3D3;

Кнопка “Редактор заявки” открывает для пользователя окно редактирования выбранной им заявки. Если же заявка не была выбрана, пользователю будет показано окно об ошибке и просьба выбрать заявку перед нажатием на кнопку (см. Рисунок 3.3.4).

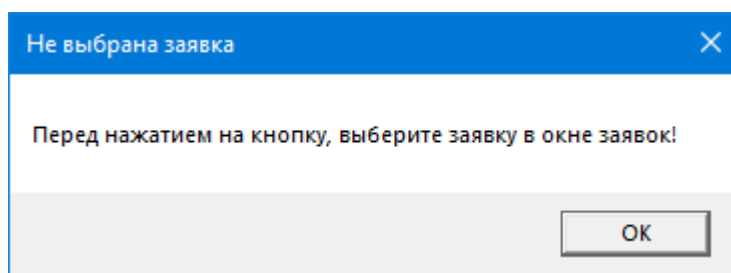


Рисунок 3.3.4 – Окно об ошибке “Не выбрана заявка”

Кнопка “Обновить” выполняет команду генерации отображаемых данных посредством повторного обращения к базе данным и переписывания значений свойств элементов интерфейса в программе. Вследствии этого происходит визуальное обновление отображаемых на интерфейсе данных.

Кнопка “Создать новую заявку” выполняет открытие окна редактирования заявки с пустыми и доступными для заполнения полями. Выбор заявки в таблице не требуется. При нажатии кнопки, пользователю будет выведено диалоговое окно, требующее необходимость повторного подтверждения действия.

Также на панели вкладок на вкладке “Поиск” возможен поиск заявок по имени клиента и по номеру конкретной заявки (см. Рисунок 3.3.5). При вводе номера заявки и нажатия на кнопку подтверждения, в таблице отобразится только заявка со введенным номером, если таковая имеется в базе.

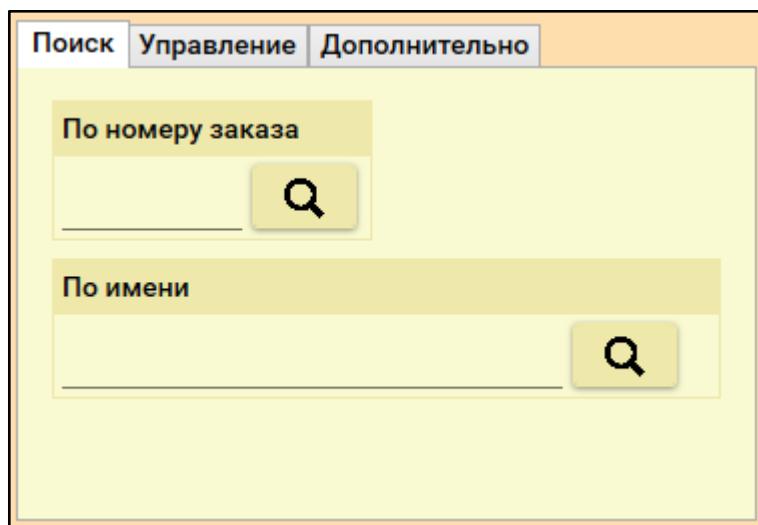


Рисунок 3.3.5 – Элементы управления поиском по заявкам

При поиске по имени отобразятся все заявки, в которых было найдено совпадение либо по меньшей части, либо по полному имени клиента.

На вкладке “Дополнительно” (см. Рисунок 3.3.6) размещены кнопки для перехода в редактор сервисной информации и редактор шаблонов изделий.

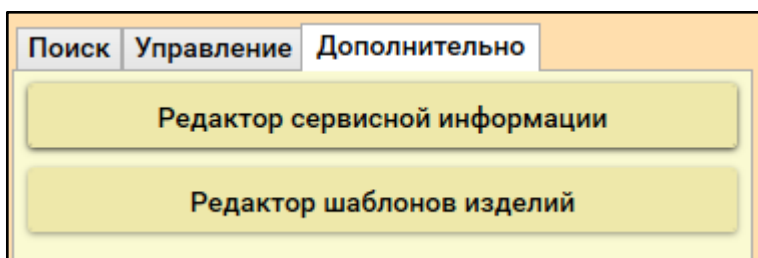


Рисунок 3.3.6 – Элементы управления дополнительным функционалом

Кнопка “Редактор сервисной информации” открывает окно редактирования сервисной информации.

Кнопка “Редактор шаблонов изделий” открывает окно редактирования шаблонов изделий.

При выборе заявки в таблице заявок и последующем нажатии на кнопку “Редактор заявки”, система открывает окно “Редактор заявки #[номер заявки]” (см. Рисунок 3.3.7), в котором доступно полное редактирование информации в заявке: изменение имени клиента, изменение статуса заявки, изменение даты, открытие редактора компонентов заявки. Код наполнения формы заявки находится в Приложении Г.

Редатор заявки #19.

Полное имя: Егоров Д.Г.

Статус заявки: 2. Обработан

Создан: 2021-05-23

Ограничение: 2021-06-22

Закрыт

Сегодня

Закреть

Сохранить изменения

Обновить всё

Создание новой заявки

Создать пустой

Добавить новый

ID	Наименование	Статус
64	Шкатулка-танкетка	2. В работе
65	Квадрат д вязания 20г	1. В очереди
66	Огр. каранда. "8 марта"	3. Готов

Редактор компонентов

ID	Наименование	Цена
10	Краск бол.	200
11	Лак обычн. бол.	150
12	Полир. мелкая	120

Рисунок 3.3.7 – Окно редактора заявки

Форма редактора поделена на 2 части:

- левая половина окна отведена под редактирование имени клиента, статуса заявки, даты создания заявки, ограничения по срокам, даты закрытия заявки. Также в этой же половине расположены следующие элементы управления: выпадающий список, кнопка “Сегодня”, кнопка “Закреть”, кнопка “Сохранить изменения”, кнопка “Обновить всё” и форма с кнопками для создания новой заявки;
- правая половина окна отведена на просмотр и редактирование компонентов заявки: верхняя таблица для отображения добавленных в заявку товаров, нижняя таблица для отображения добавленных услуг, а также кнопка “Редактор компонентов” и выпадающий список с кнопкой изменения статуса товара.

Кнопка “Сегодня” вносит в поле “Создан” текущую дату, также возможно внесение даты вручную в указанном формате. При нажатии кнопки подтверждения, пользователю будет выведено диалоговое окно, требующее необходимость повторного подтверждения действия.

Кнопка “Закреть” вносит в поле “Закреть” текущую дату, также возможно внесение даты вручную в указанном формате. При нажатии, сразу вносит изменения в базу данных и изменяет статус заявки на “Закрета”. При нажатии кнопки подтверждения, пользователю будет выведено диалоговое окно, требующее необходимость повторного подтверждения действия.

Кнопка “Сохранить изменения” позволяет внести в базу все произведенные изменения значений полей заявки, проверяет корректность введенных данных, а также обновляет всю форму заявки соответственно внесенным данным. При нажатии кнопки подтверждения, пользователю будет выведено диалоговое окно, требующее необходимость повторного подтверждения действия.

Кнопка “Обновить всё” вызывает функцию, обращающуюся к базе данных и получающую актуальные данные, по которым происходит обновление интерфейса.

В блоке “Создание новой заявки” представлены 2 кнопки:

- кнопка “Создать пустой” полностью очищает форму редактора и предоставляет возможность создания новой заявки;
- кнопка “Добавить новый” срабатывает только после нажатия на кнопку “Создать пустой” и заполнения хотя-бы одного поля в форме редактирования заявки. Присваивает заявке номер и вносит данные в базу. При нажатии кнопки подтверждения, пользователю будет выведено диалоговое окно, требующее необходимость повторного подтверждения действия.

При выборе элемента таблицы, отображающей товары, происходит заполнение таблицы, отображающей дополнительные услуги по отдельному товару.

Таблица дополнительных услуг отображает дополнительные услуги по обработке конкретного изделия и содержит следующие поля: идентификатор записи, наименование услуги и цена услуги. Код заполнения таблицы дополнительных услуг находится в Приложении В.

Выпадающий список в правой части окна позволяет выбрать новый статус товара и по нажатию на кнопку применить его к выбранному пункту в таблице товаров. При нажатии кнопки подтверждения, пользователю будет выведено диалоговое окно, требующее необходимость повторного подтверждения действия.

Кнопка “Редактор компонентов” открывает редактор товаров и услуг заявки (см. Рисунок 3.3.8), позволяющий добавить в заявку товары и дополнительные услуги, заказанные клиентом. При добавлении каждого нового пункта в заявку, пользователю будет выведено диалоговое окно, требующее необходимость повторного подтверждения действия.

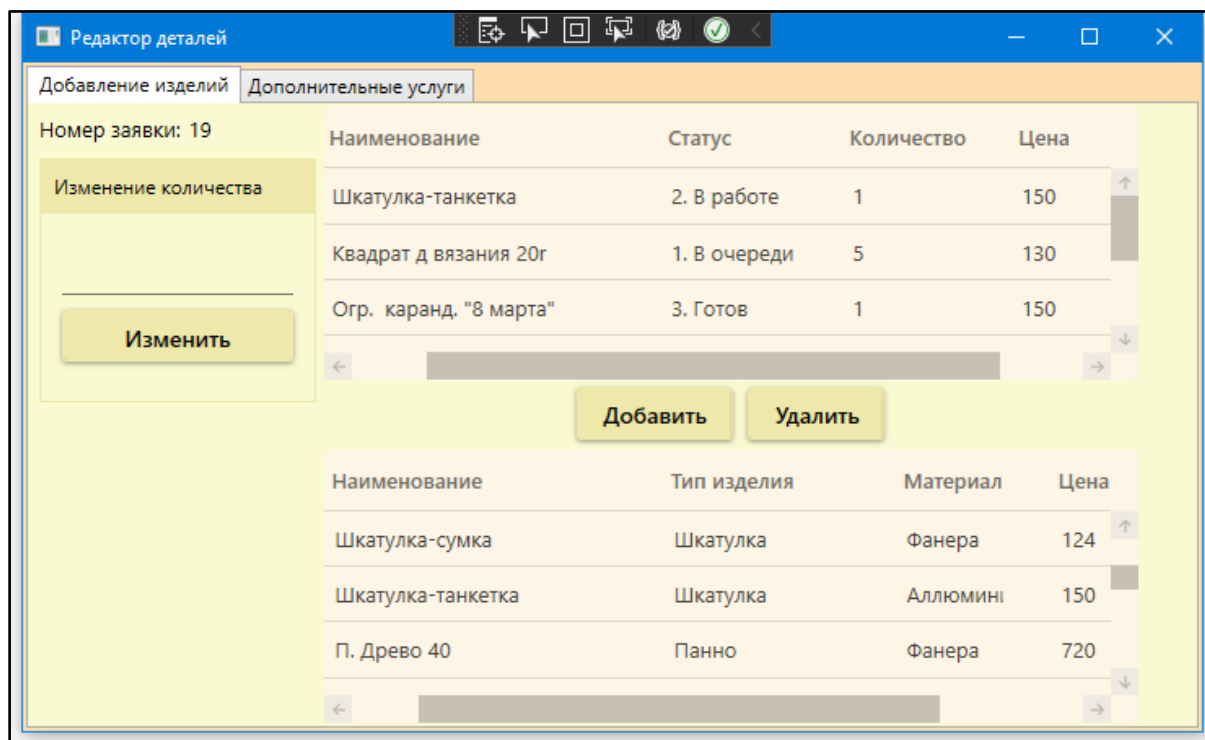


Рисунок 3.3.8 – Редактор деталей заявки

Всё пространство редактора компонентов занимает панель вкладок, где можно переключаться между вкладкой “Добавление изделий” и вкладкой “Дополнительные услуги”. На рисунке представлена первая вкладка.

На вкладке “Добавление изделий” реализована процедура добавления выбранного клиентом товара в заявку, удаление ненужных товаров из заявки, а также изменение количества товаров в одном пункте.

В первой сверху таблице представлены уже добавленные в заявку товары со следующими полями: идентификатор, наименование товара, статус товара, количество, цена товара, идентификатор заявки, идентификатор шаблона изделия.

Во второй таблице представлены шаблоны изделий, которые можно добавить в список товаров, таблица содержит следующие поля: идентификатор шаблона, наименование шаблона, материал изделия, тип изделия, цена, общая размерность.

Для добавления и удаления товаров используются соответственно кнопки “Добавить” и “Удалить”. Для добавления необходимо выделить изделие во второй таблице, а для удаления - выделить товар в первой таблице и нажать нужную кнопку. При нажатии кнопки подтверждения, пользователю будет выведено диалоговое окно, требующее необходимость повторного подтверждения действия.

В блоке “Изменение количества” возможно изменение количества в пункте в первой таблице. Необходимо выбрать пункт, ввести количество и нажать на кнопку “Изменить”.

При нажатии кнопки подтверждения, пользователю будет выведено диалоговое окно, требующее необходимость повторного подтверждения действия. Новые сведения запишутся в базу данных и окно будет обновлено в соответствии с новыми данными.

Также отображается номер заявки, с которой происходит работа в текущем окне.

На рисунке 3.3.9 изображена вкладка “Дополнительные услуги”. По аналогии с предыдущей вкладкой, содержит две таблицы и кнопки “Удалить” и “Добавить”. При нажатии кнопки подтверждения, пользователю будет выведено диалоговое окно, требующее необходимость повторного подтверждения действия.

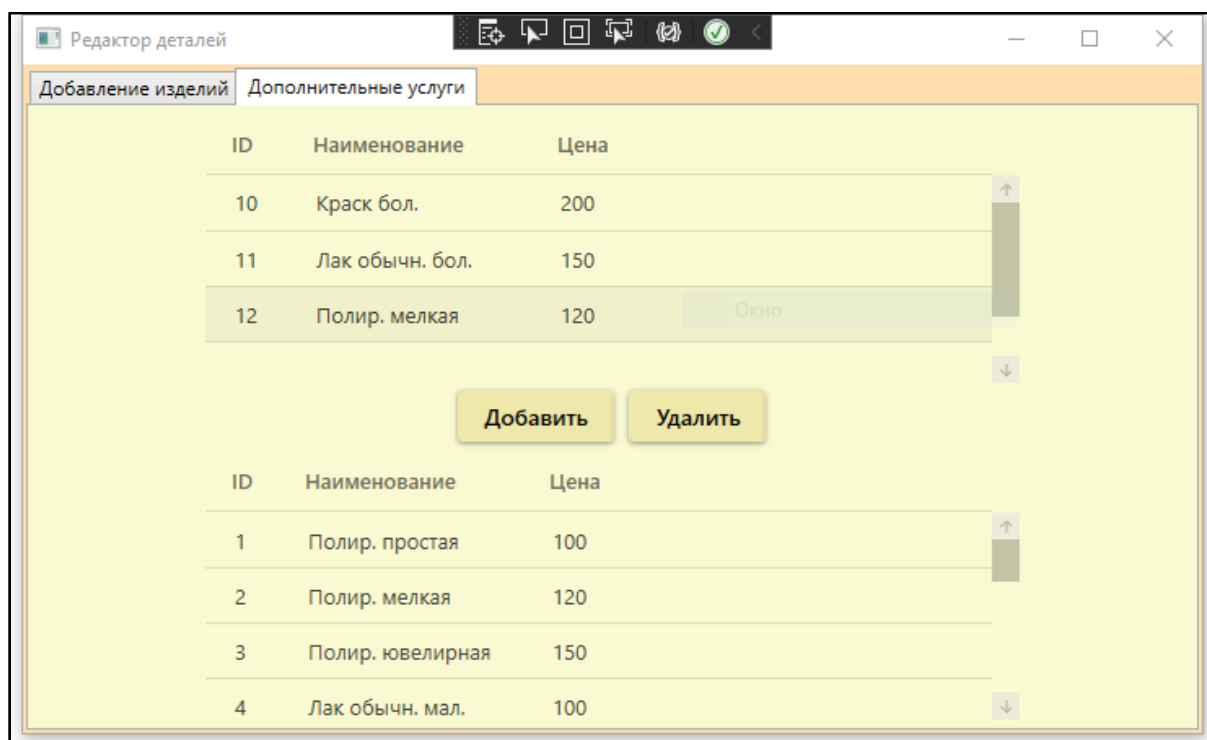


Рисунок 3.3.9 – Добавление услуг в заявку

Первая таблица сверху отображает добавленные к выбранному изделию дополнительные опции по обработке и имеет следующие поля: идентификатор, наименование и цена.

Вторая таблица отображает возможные для добавления дополнительные услуги и имеет следующие поля: идентификатор, наименование и цена.

Две кнопки “Добавить” и “Удалить” работают по аналогии с предыдущей вкладкой.

На рисунке 3.3.10 изображен редактор сервисной информации, вызываемый при помощи кнопки “Редактор сервисной информации” на главной форме приложения.

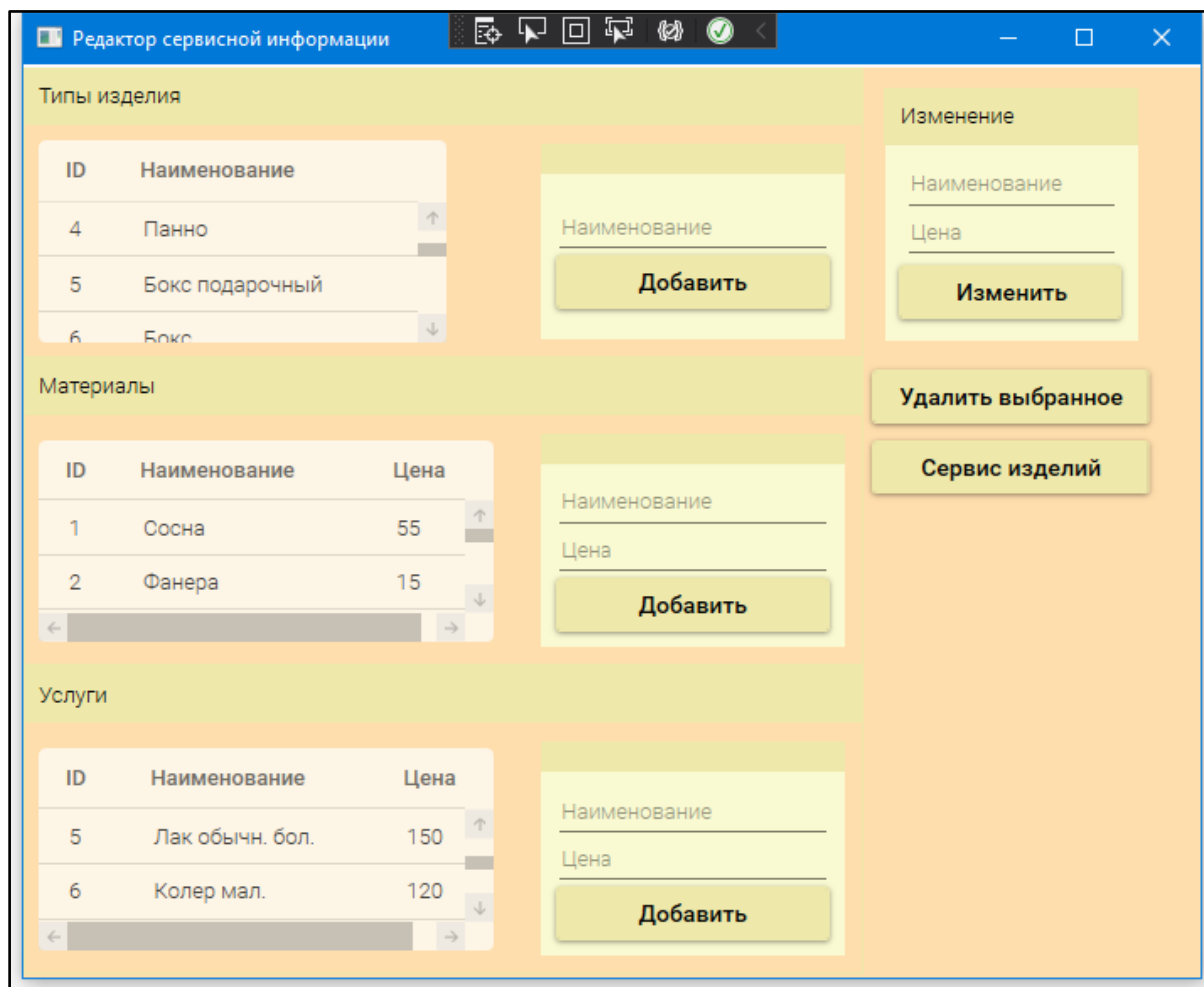


Рисунок 3.3.10 – Редактор сервисной информации

Окно имеет 4 поля редактирования сервисных данных, а также 2 отдельные кнопки.

Кнопка “Удалить выбранное” производит удаление выбранного пункта из таблицы, выделение пункта в которой было крайним, по сравнению с остальными таблицами. При нажатии кнопки подтверждения, пользователю будет выведено диалоговое окно, требующее необходимость повторного подтверждения действия.

Кнопка “Сервис изделий” открывает окно редактирования шаблонов изделий, изображенное на рисунке 3.3.11.

В поле “Изменение” можно изменить данные последнего выбранного объекта в базе данных. При нажатии кнопки подтверждения, пользователю будет выведено диалоговое окно, требующее необходимость повторного подтверждения действия.

Также в каждом табличном блоке есть возможность добавления нового объекта в базу данных. При нажатии кнопки подтверждения, пользователю будет выведено диалоговое окно, требующее необходимость повторного подтверждения действия.

Редактор шаблонов изделий

Полное наименование _____

Тип изделия _____

Материал _____

Чистая цена _____

Размерность _____

Обновить

Изменить

Чистая форма

Добавить новый

- номер выбранного изделия

ID	Наименование	Тип изделия	Материал	Цена
7	Бокс бут. мягк.	Бокс подарочный	Фанера	60
14	Купюрница #4 С д.р. Робот	Бокс подарочный	Фанера 6	35
9	Круг для вязания 15г	Д.вяз.круг.15	Фанера 6	80
10	Круг для вязания 20г	Д.вяз.круг.20	Фанера 6	10

Рисунок 3.3.11 – Редактор шаблонов изделий

Редактор шаблонов изделий позволяет произвести следующие действия с базой данных:

- редактировать любое из полей шаблона: наименование шаблона, тип шаблона - выпадающий список справа от поля, материал изделия - выпадающий список справа от поля, цену изделия и основную размерность;
- при нажатии на кнопку “Изменить” - сохранить изменения полей шаблона базе данных;
- при нажатии на кнопку “Обновить” - обновить отображаемые данные посредством функции сравнения задействованных свойств, отвечающих за отраженные в окне данные, и текущих данных в базе данных;
- при нажатии на кнопку “Чистая форма” - очистить поля формы для добавления нового шаблона изделия в базу;
- при нажатии на кнопку “Добавить новый” - добавить новый шаблон после очистки формы и заполнения новых данных.
- просмотр и выбор для редактирования уже существующих шаблонов изделий из базы данных.

3.4 Вывод в EXCEL

Для вывода отчета в EXCEL использована библиотека EPPlus .NET. В отчет включаются номер заявки, товары в заявке, количество и цена каждого товара. Итоговые суммы просчитываются уже в EXCEL-файле.

На рисунке 3.4.1 представлен пример вывода отчета в EXCEL и направление на его на печать.

LazerPan	ТЕЛЕФОН КОМПАНИИ		8-666-123-45-67	
	ЗАКАЗ №15			
№	наименование	кол-во	цена	сумма
1	Медаль "Лыжник"	250	25,00 Р	6 250,00 Р
2	Панно "Древо"	2	250,00 Р	500,00 Р
3	Подставка д. кош. Мис. 4.	1	1 200,00 Р	1 200,00 Р
4	Подставка п. моб. "Хок."	2	150,00 Р	300,00 Р
ИТОГОВАЯ ЦЕНА:				8 250,00 Р

Рисунок 3.4.1 – Вывод отчета по заявке в EXCEL

3.5 Тестирование программного продукта

Для тестирования полученного продукта были разработаны пара тестовых сценариев: тестирование сервисных функций и тестирование полного цикла обработки заявки.

В первом тестовом сценарии необходимо было добавить следующие данные в систему:

- внести новый тип изделия - резной бокс;
- внести новый материал - “фанера 4”;
- редактировать цену материала фанера 4 - приравнять к 25;
- добавить дополнительную услугу - “прожигание дополнительных символов”;
- редактировать услугу “прожигание дополнительных символов” - приравнять цену к 50;
- проверить на дублирование - добавить ещё услугу “прожигание дополнительных символов”;
- изменить тип “Без типа” на символ нижнего подчеркивания;
- изменить тип “<символ нижнего подчеркивания>” на тип “Без типа”;

- добавить новое изделие: резная чайная шкатулка, тип – шкатулка, материал - фанера 4, цена - 250, размерность - 10x20x30;
- изменить цену и материал прежде добавленного изделия на: фанера 6, цена - 400.

Первый сценарий был успешно пройден, ошибки не выявлены, система работала в штатном режиме. По итогу в систему был занесён новый шаблон изделия, получивший порядковый номер 15 (см. Рисунок 3.5.1).

Полное наименование	Резная чайная шкатулка	
Тип изделия	Шкатулка	Шкатулка ▼
Материал	Фанера 4	Фанера 4 ▼
Чистая цена	250	
Размерность	10x20x30	
15- номер выбранного изделия		

Рисунок 3.5.1 - Полученный шаблон в тестовом сценарии 1

Во втором тестовом сценарии необходимо было выполнить следующие действия для прохода по полному циклу обработки заказа:

- добавить новую заявку, имя клиента - Панихидов А.А., статус – Новый;
- внести сегодняшнюю дату создания заявки;
- добавить в заявку произвольные товары, в количестве не менее 3 и не более 10;
- на каждый товар добавить дополнительные услуги в количестве не менее 2 и не более 6 услуг;
- внести сроки выполнения заявки на завтрашний день;
- перевести системное время на 1 день вперед, проверить на наличие ошибок автоматическое изменение статуса изделия при запаздывании выполнения заявки на “Проблема”;
- изменить сроки ещё на 2 дня вперёд;
- изменить статус заявки на “Обработан”;
- изменить статусы изделий на “В очереди”;
- последовательно изменять статусы товаров с “В очереди” на “В работе”;
- последовательно изменять статусы товаров с “В работе” на “Готов”;
- проверить работоспособность статуса товара “Проблема”;

- изменить статус заявки на “Готов”;
- закрыть заявку;
- проверить корректность заполнения полей закрытой заявки.

По результатам тестового сценария была успешно внесена в систему и обработана тестовая заявка с порядковым номером 25, изображенная на рисунке 3.5.2. Выявлены некоторые неточности работы системы.

ID	Наименование	Статус
73	Круг для вязания 20г	3. Готов
74	П. Дерево 40	3. Готов
75	Подставка под миск. кош. #4	3. Готов

ID	Наименование	Цена
16	Полир. мелкая	120
17	Лак обычн. мал.	100

Рисунок 3.5.2 – Результат второго тестового сценария

По результатам тестовых сценариев, функционал системы был полностью проверен, ошибки и неточности в работе системы устранены в полном объеме.

Продукт тестировался на портативном компьютере с ACPI на базе x64 со следующими основными системными характеристиками:

- центральный процессор Intel Core i5-3330 4-core 3.00-3.20 GHz;
- видеокарта NVIDIA GeForce GTX 1050 2Gb;
- ОЗУ DDR3 16Gb.

3.6 Итоговая структура проекта

Итоговая структура проекта в Visual Studio 2019 изображена на рисунке 3.6.1. Было получено 6 файлов .cs, 5 окон в формате XAML, сервисный класс для управления базой данных, иконки поиска для кнопок поиска.

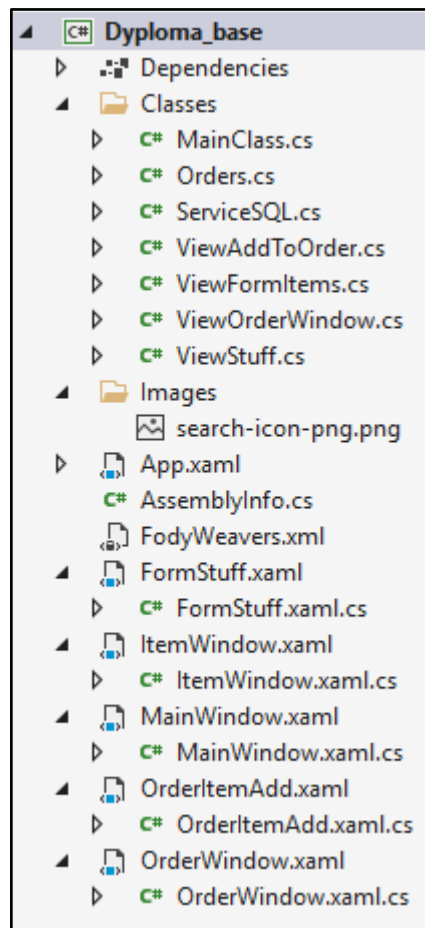


Рисунок 3.6.1 – Итоговая структура проекта

Полученные файлы окон проекта на языке разметки XAML, у каждого файла есть управляющий класс с одинаковым названием и расширением “.cs”:

- MainWindow.xaml - главное окно, обеспечивает доступ и связь с остальными окнами;
- OrderWindow.xaml - окно редактора отдельной заявки;
- OrderItemAdd.xaml - окно редактора деталей заявки;
- FormStuff.xaml - окно редактора сервисной информации;
- ItemWindow.xaml - окно редактора шаблонов изделий.

Также были созданы различные файлы “.cs”, полностью описывающие логику приложения в отрыве от реализации интерфейса:

- MainClass.cs: реализована ViewModel MainClass для представления MainWindow, реализован класс DelegateCommand для использования библиотеки ICommand;
- Orders.cs: реализована Model Orders для основных данных заявки, реализована модель OrderItems для отображения данных по товарам в заявке, реализована модель AdditionalOnItem для отображения данных по услугам в заявке;

- ViewOrderWindow.cs: реализована ViewModel ViewOrderWindow для привязки данных в представлении OrderWindow.xaml;
- ViewAddToOrder.cs: реализована ViewModel ViewAddToOrder для привязки данных в представлении OrderItemAdd.xaml;
- ViewStuff.cs: в файле описана ViewModel ViewStuff для привязки данных в представлении FormStuff.xaml, также реализованы классы-модели Types, Materials, Additions, Items;
- ViewFormItem.cs: реализована ViewModel ViewFormItem для привязки данных в представлении FormItem.xaml;
- ServiseSQL.cs: реализован класс с методами для обработки данных, получаемых из базы данных.

Количество строк кода по завершению работы над прототипом: *~3500 строк*.

Итоговый вес проекта: 15,3 МБ (16 052 492 байт).

Заключение

В результате выпускной квалификационной бакалаврской работы была спроектирована и разработана графическая система учёта и мониторинга рабочих процессов на предприятии на платформе “WPF”.

Цели работы были достигнуты благодаря успешному выполнению следующих поставленных задач:

- реализованы хранение и обработка данных средствами системы управления реляционными базами данных;
- реализована возможность просмотра, редактирования клиентский данных, а также внутренних данных предприятия;
- разработан наглядный и оптимальный интерфейс, поддерживающий одновременную работу с несколькими формами;
- организовано необходимое тестирование прототипа программного продукта.

В ходе разработки системы, основной упор был сделан на полноценное овладение технологией WPF и системой управления базами данных MariaDB. По итогу, данный проект позволил досконально изучить возможности WPF и получить необходимые навыки для эффективной работы с технологией, а применение специализированного архитектурного паттерна - познакомить поближе с необходимыми ограничениями в рамках разработки серьезного проекта.

Наличие реального товара, пригодного к реализации, и объективных требований, упростило понимание предметной области и ускорило постановку задач и проектирование структуры данных, а также позволило глубже понять необходимость и реальную пригодность систем такого типа в условиях рынка.

Использование паттерна MVVM позволило серьезно структурировать процесс разработки и упростило поддержку и дальнейшее улучшение системы. При желании, возможно полное изменение и адаптирование под современные стандарты и требования интерфейса системы в краткие сроки. Не будет необходимости переписывания кода, поэтому изменением интерфейса может заняться дизайнер, не имеющий профессионального опыта в программировании логики и потоков данных. Также, если в дальнейшем поддержкой и улучшением проекта будет заниматься разработчик, ранее не знакомый с системой, структура проекта позволит ему в кратчайшие сроки адаптироваться к системе.

Также использование паттерна MVVM впоследствии дало возможность удобного и эффективного использования модульного и функционального тестирования, при необходимости. Это возможно благодаря тому, что сама структура паттерна является модульной и функции имеют встроенные механизмы защиты и возвращают конкретные значения, а также, в большинстве своём, выполняют всего одно действие и не связаны друг с другом излишней логикой. Модели данных вообще не имеют методов, только свойства.

Несмотря на то, что в качестве хранилища данных используется актуальная, открытая и активно поддерживаемая разработчиками СУБД MariaDB, итоговая система построена таким образом, что, при желании, возможен быстрый и дешёвый переход на популярные сегодня NoSQL системы управления базами данных. Переход, например, на JSON-ориентированную систему практически не потребует серьёзного изменения кода проекта, так как существующие в проекте модели паттерна MVVM практически готовы к использованию для описания структуры данных JSON-файлов.

В целом, система получилась простой и надёжной. Технические ошибки могут появиться только при обновлении программного обеспечения на новые версии, а ошибки пользователя сведены к минимуму посредством диалоговых окон, внутренних проверок и возможности гибкого редактирования уже внесённых изменений.

По итогу выполнения в полном объеме всех поставленных задач и использования современных средств проектирования, основная цель работы была успешно достигнута.

Список используемой литературы

1. ГОСТ 7.32 — 2001 Отчет о научно-исследовательской работе. Структура и правила оформления.
2. Гвоздева, Т.В. Проектирование информационных систем. Стандартизация: Учебное пособие / Т.В. Гвоздева, Б.А. Баллод. - СПб.: Лань, 2019. - 252 с.
3. Белов, В.В. Проектирование информационных систем: Учебник / В.В. Белов. - М.: Академия, 2018. - 144 с.
4. Мартишин, С.А. Проектирование и реализация баз данных в СУБД MySQL с использованием MySQL Workbench: Методы и средства проектирования информационных систем и технологии / С.А. Мартишин, В.Л. Симонов, М.В. Храпченко. - М.: Форум, 2018. - 61 с.
5. Подбельский В. Язык декларативного программирования XAML М.: Издательство «ДМК Пресс», 2018. 336 с.
6. Прайс, М. С# 7 и .NET Core. Кросс-платформенная разработка для профессионалов. - Пер. с англ. / С. Черников - Издательский дом «Питер», 2018. - 640 с.
7. Мак-Дональд WPF: Windows Presentation Foundation в .NET 4.5 с примерами на С# 5.0 для профессионалов. - 4-е изд.: Изд-во «Вильямс», 2017. - 1024 с.
8. Джозеф А. С# 7.0. Справочник. Полное описание языка, 7-е издание
9. Холл Г.М. Адаптивный код: гибкое кодирование с помощью паттернов проектирования и принципов SOLID. - 2-е изд.: Изд-во «Диалектика», 2018. -448 стр.: с ил.
10. Дудин М. Н., Лясников Н. В., Сенин А. С. Менеджмент малого бизнеса (Управление малым предприятием): учебное пособие. М.: Издательство «Элит», 2016. 397 с.
11. Эспозито Д. Разработка современных веб-приложений: анализ предметных областей и технологий.-Изд-во «Вильямс», 2017. - 464 стр., с ил.
12. Матюшин М.Г. Справочник по цвету: закономерность изменчивости цветовых сочетаний М.: Издатель Д. Аронов, 2007. 76 с.
13. Закон РФ «Об участии в информационном обмене» от 04.07.1996, № 85-ФЗ
14. Хабр. Старт работы с Excel на С#
<https://habr.com/ru/post/525492/>
15. Хабр. MVVM: полное понимание (+WPF) Часть 1.
<https://habr.com/ru/post/338518/>

16. Хабр. MVVM: полное понимание (+WPF) Часть 2.
<https://habr.com/ru/post/338518/>
17. Хабр. Как программисту написать диплом. Полное руководство.
<https://habr.com/ru/post/491164/>
18. c-sharpcornered. Data Binding in WPF DataGrid Control Using SQL Server Database.
<https://www.c-sharpcorner.com/UploadFile/deepak.sharma00/data-binding-in-wpf-datagrid-control-using-sql-server-databa/>

Приложение

Приложение А Функция заполнения списка заявок

```
public ObservableCollection<Orders> orders_Fill()
{
    string sql = "SELECT * FROM azarov1.view_orders";
    DataTable dt = new DataTable();
    using (MySQLConnection connection = new
MySQLConnection(MyConnectionString))
    {try
        {connection.Open();
            using (MySQLCommand cmdSel = new MySQLCommand(sql,
connection))
            {
                MySQLDataAdapter da = new MySQLDataAdapter(cmdSel);
                da.Fill(dt);
            }
            foreach (DataRow row in dt.Rows)
            {
                var game = new Orders
                {
                    id = Convert.ToInt32(row["id"]),
                    FIO = row["FIO"].ToString(),
                    status = row["description"].ToString(),
                    created = row["created"].ToString(),
                    deadline = row["deadline"].ToString(),
                    closed = row["closed"].ToString(),
                };
                _orders_Fill.Add(game);
            }
            //обрезаем время у datetime для отображения только даты
            foreach (Orders ord in _orders_Fill)
            {
                ord.created =
String.Concat(ord.created.Reverse().Skip(8).Reverse());
                ord.closed =
String.Concat(ord.closed.Reverse().Skip(8).Reverse());
                ord.deadline =
String.Concat(ord.deadline.Reverse().Skip(8).Reverse());
            }
            return _orders_Fill;
        }
        catch (Exception ex)
        {
            //MessageBox.Show(ex.Message);return _orders_Fill;
        }
        finally
        {
            connection.Close();
        }
    }
}
```

Приложение Б Функция заполнения списка товаров

```
private ObservableCollection<OrderItems> _orderitemsfill = new
ObservableCollection<OrderItems>();
public ObservableCollection<OrderItems> orderitemsfill(string
orderid)
{
    if (orderid != null)
    {
        string sql = "call proc_order_items(" + orderid + ")";
        DataTable dt = new DataTable();
        using (MySQLConnection connection = new
MySQLConnection(MyConString))
        {
            try
            {
                connection.Open();
                using (MySQLCommand cmdSel = new MySQLCommand(sql,
connection))
                {
                    MySQLDataAdapter da = new
MySQLDataAdapter(cmdSel);
                    da.Fill(dt);
                }
                foreach (DataRow row in dt.Rows)
                {
                    var orderitem1 = new OrderItems
                    {
                        id = Convert.ToInt32(row["id"]),
                        orderid = Convert.ToInt32(row["orderid"]),
                        quantity = Convert.ToInt32(row["quantity"]),
                        itemname = row["itemname"].ToString(),
                        price = Convert.ToInt32(row["price"]),
                        statusitem = row["statusitem"].ToString(),
                    };
                    _orderitemsfill.Add(orderitem1);
                }
                return _orderitemsfill;
            }
            catch (Exception ex)
            {
                return _orderitemsfill;
            }
            finally
            {
                connection.Close();
            }
        }
    }
    else return null;
}
```


Приложение В Функция заполнения списка добавленных услуг

```
private ObservableCollection<AdditionalsonItem> _additionalsonfill = new
ObservableCollection<AdditionalsonItem>();
    public ObservableCollection<AdditionalsonItem> additionalsonfill(string
orderitem)
    {

        if (Convert.ToInt32(orderitem) != 0)
        {
            using (MySqlConnection connection = new
MySqlConnection(MyConString))
            {
                try
                {
                    string msg_in = "select * from
view_added_additionalson where orderitemsid =" + orderitem + ";";
                    connection.Open();
                    MySqlDataAdapter SDA2 = new MySqlDataAdapter(msg_in,
connection);

                    DataTable DATA2 = new DataTable();
                    SDA2.Fill(DATA2);
                    foreach (DataRow row in DATA2.Rows)
                    {
                        var add = new AdditionalsonItem
                        {
                            id = Convert.ToInt32(row["id"]),
                            description = row["description"].ToString(),
                            price = Convert.ToInt32(row["price"]),
                        };
                        _additionalsonfill.Add(add);
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
                finally
                {
                    connection.Close();
                }
            }
        }

        return _additionalsonfill;
    }
```

Приложение Г Функция заполнения формы заявки

```
public void LoadOrder(ref string orderid, ref string fio, ref string
status, ref string opendate, ref string deadline, ref string closedate)
{
    using (MySQLConnection conn = new MySQLConnection(MyConString))
    { try
        {
            string name = null;
            conn.Open();
            // запрос
            string sql = "SELECT a.fio FROM azarov1.Clients a inner
join azarov1.orders b on b.clientid = a.id where b.id =" + orderid + ";";
            // объект для выполнения SQL-запроса
            MySqlCommand command = new MySqlCommand("SELECT b.fio
FROM azarov1.orders a left join azarov1.clients b on a.clientid = b.id where
a.id = " + orderid + ";", conn);
            // выполняем запрос и получаем ответ
            name = command.ExecuteScalar().ToString();
            fio = name.ToString();
            try
            {
                sql = "SELECT a.created FROM azarov1.orders a where
a.id =" + orderid + ";";
                // объект для выполнения SQL-запроса
                command = new MySqlCommand(sql, conn);
                // выполняем запрос и получаем ответ
                string pars1 = command.ExecuteScalar().ToString();
                if (pars1 != null)
                {
                    DateTime datestring = DateTime.Parse(pars1);
                    //datestring =
String.Concat(datestring.Reverse().Skip(8).Reverse());
                    opendate = datestring.ToString("yyyy-MM-dd");
                }
            }
            catch {}
            try
            {
                sql = "SELECT a.deadline FROM azarov1.orders a where
a.id =" + orderid + ";";
                // объект для выполнения SQL-запроса
                command = new MySqlCommand(sql, conn);
                // выполняем запрос и получаем ответ
                string pars2 = command.ExecuteScalar().ToString();
                if (pars2 != null && pars2 != "")
                {
                    DateTime date2 = DateTime.Parse(pars2);
                    deadline = date2.ToString("yyyy-MM-dd");
                }
            }
            catch{}
            try
            {
```

```

        sql = "SELECT a.closed FROM azarov1.orders a where
a.id =" + orderid + ";;";
        // объект для выполнения SQL-запроса
        command = new MySqlCommand(sql, conn);
        // выполняем запрос и получаем ответ
        string pars3 = command.ExecuteScalar().ToString();
        if (pars3 != null && pars3 != "")
        {
            DateTime date3 = DateTime.Parse(pars3);
            closedate = date3.ToString("yyyy-MM-dd");
        }
    }
    catch{ }
    try
    {
        sql = "select a.description from azarov1.statustable
a right join azarov1.orders b on b.orderstatus = a.id where b.id = " +
orderid + ";;";

        command = new MySqlCommand(sql, conn);
        status = command.ExecuteScalar().ToString();
    }
    catch{}
}
catch (Exception ex)
{
    // MessageBox.Show(ex.Message);
}
finally
{
    conn.Close();
    // orderitemsfill();
}
}
}

```

