СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 6](#_Toc100854967)

[1. АНАЛИТИЧЕСКАЯ ЧАСТЬ 7](#_Toc100854968)

[1.1. Обоснование актуальности задачи 7](#_Toc100854969)

[1.1.1. Актуальность темы 7](#_Toc100854970)

[1.1.2. Анализ существующих разработок по данной теме 7](#_Toc100854971)

[1.2. Обоснование выбора и описание методологии разработки программы решения задачи 9](#_Toc100854972)

[1.3. Обоснование выбора и описание инструментальных средств разработки программы решения задачи 11](#_Toc100854973)

[1.4. Содержательная постановка задачи 13](#_Toc100854974)

[1.4.1. Описательная модель задачи 13](#_Toc100854975)

[1.4.2. Математическая модель задачи 13](#_Toc100854976)

[1.4.3. Описание входной информации 13](#_Toc100854977)

[1.4.4. Описание выходной информации 13](#_Toc100854978)

[1.4.5. Описание бизнес-процессов предприятия 14](#_Toc100854979)

[1.4.6. Описание потоков данных предприятия 14](#_Toc100854980)

[1.4.7. Диаграмма вариантов использования 14](#_Toc100854981)

[1.4.8. Диаграмма последовательности 14](#_Toc100854982)

[1.4.9. Диаграмма взаимосвязей модулей 14](#_Toc100854983)

[2. ПРОЕКТНАЯ ЧАСТЬ 15](#_Toc100854984)

[2.1. Разработка структуры данных программы 15](#_Toc100854985)

[2.2. Логическая модель программы 22](#_Toc100854986)

[2.3. Разработка пользовательского интерфейса программы 22](#_Toc100854987)

[2.3.1. Разработка требований к интерфейсу 22](#_Toc100854988)

[2.3.2. Проектирование и описание элементов управления 22](#_Toc100854989)

[2.4. Листинг программы 32](#_Toc100854990)

[2.5. Разработка тестовых данных 52](#_Toc100854991)

[2.6. Результат работы программы 62](#_Toc100854992)

[2.7. Руководство по использованию программы 65](#_Toc100854993)

[2.7.1. Руководство системного администратора 65](#_Toc100854994)

[2.7.2. Руководство программиста 65](#_Toc100854995)

[2.7.3. Руководство оператора 66](#_Toc100854996)

[2.8. Разработка политики безопасности 70](#_Toc100854997)

[2.9. Эксплуатация и развитие программы 71](#_Toc100854998)

[ЗАКЛЮЧЕНИЕ 72](#_Toc100854999)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 73](#_Toc100855000)

[ПРИЛОЖЕНИЕ 1 – Бизнес-процесс IDEF0 75](#_Toc100855001)

[ОКОНЧАНИЕ ПРИЛОЖЕНИЯ 2 76](#_Toc100855002)

[ПРИЛОЖЕНИЕ 2 – Потоки данных DFD 77](#_Toc100855003)

[ОКОНЧАНИЕ ПРИЛОЖЕНИЯ 3 78](#_Toc100855004)

[ПРИЛОЖЕНИЕ 3 – Диаграмма вариантов использования 79](#_Toc100855006)

[ПРИЛОЖЕНИЕ 4 – Диаграмма последовательности 80](#_Toc100855007)

[ПРИЛОЖЕНИЕ 5 – Диаграмма классов 81](#_Toc100855008)

[ПРИЛОЖЕНИЕ 6 – Диаграмма деятельности 82](#_Toc100855010)

[ПРИЛОЖЕНИЕ 7 – Диаграмма развёртывания 83](#_Toc100855011)

[ПРИЛОЖЕНИЕ 8 – Логическая модель 84](#_Toc100855012)

[ПРИЛОЖЕНИЕ 9 – Физическая модель 85](#_Toc100855014)

# ВВЕДЕНИЕ

Для учета работы конструкторского бюро для производства изделий на предприятии используется устаревшее ПО. Для повышения эффективности работы предприятия необходимо разработать программу, которая автоматизирует и улучшает данный процесс.

Требуется вести учет состояний позиций заказов. Записью и формированием данных документов занимается отдел учёта процесса изготовления производственных заказов. В отдел поступают данные об изделиях и сотрудники в позициях заказа учитывают когда (Дата) оканчивается изготовление позиции в заказе, которые фиксируются не только в программе, но и в базе данных.

Для эффективности работы необходимо ведение базы данных, содержащей информацию о заказе, данные из номенклатуры, данные в позиции заказа заносятся инженером-технологом для учета готовности заказов.

Целью данной работы является разработка программного обеспечения, решающего следующие задачи:

* работа с состоянием готовности позиций заказов;
* просмотр клиентов и их заказов;
* работа с состоянием готовности заказов;
* формирование отчётности по состоянию выполнения заказа;
* формирование статистики выполнения заказов.

Объектом разработки является отчёт, который хранит в себе все данные по заказу и процессу его выполнения.

Предметом разработки является отдел инженеров как система контроля готовности заказов.

# 1. АНАЛИТИЧЕСКАЯ ЧАСТЬ

1.1. Обоснование актуальности задачи

1.1.1. Актуальность темы

В наши дни приложения для предприятий весьма актуальны. Автоматизация учёта заказов позволит ускорить рабочий процесс конструкторского бюро и принесёт немалый рост эффективности менеджмента заказов. В данной задаче автоматизируется учёт готовности заказов и их позиций. Такой учет необходим для быстрой работы предприятия по отслеживанию заказов готовых к выдаче клиенту и оперативного взаимодействия разных корпусов предприятия.

Автоматизировав процесс, предприятие с легкостью сможет узнать необходимую информацию. Быстрый поиск готовых заказов позволит своевременно передавать их клиенту.

1.1.2. Анализ существующих разработок по данной теме

«Класс365» - это облачный сервис для автоматизации бизнеса. Сервис позволяет легко автоматизировать бизнес-процессы розничной и оптовой торговли, сферы услуг, а также организовать эффективную деятельность отдела продаж и склада. Также Класс365 поддерживает интеграцию со многими популярными платформами интернет-магазинов. «Supasoft CRM» - универсальная программа для учета нормативно-справочной информации предприятия. Сетевая версия позволит нескольким пользователям работать в единой базе с разными правами доступа. Автоматическое заполнение и печать документов.

«РемОнлайн» - это программа для разных типов бизнеса услуг: мастерских, автосервисов, ателье, ремонта электроники, типографий, химчисток, проката и аренды, ремонта и заправки картриджеи, управления автомобильной СТО. Доступен поиск заказов по различным параметрам: номер, наименованию, имя или телефон клиента. Складской учёт всего объёма товаров и запчастей мастерской: применение, списание, перенос запчастей между складами мастерскими, у каждого товара доступны категория и артикул. Модуль магазина для регистрации розничных продаж. После завершения сделки есть опция печати «Товарного чека» для отчётности.

«Бизнес.ру» - Отслеживание запасов, оперативная регистрация новых заказов и перемещение заказов на обработку; Укрепление отношений с клиентами (менеджер при общении с клиентом уже знает всю информацию о них, заказе, доставке, оплате и т.д). Упорядоченный процесс управления заказами, возможность быстро найти нужный заказ. Уменьшение времени на создание отчетов по заказам;

Плюсы и минусы этих систем представлены в таблице 1.

Таблица 1 – плюсы и минусы систем автоматизации процесса

|  |  |  |
| --- | --- | --- |
| Наименование | Плюсы | Минусы |
| «Класс365» | Удобная визуализация состояний заказов | Множество модулей программы поставляется в отдельном, платном пакете |
| Возможность отправки данных в ФМС | Нет бесплатного тарифа |
| Облачная система | Нет многофакторной авторизация |
| «РемОнлайн» | Достаточная проста в эксплуатации | Медленное создание отчёта |
| Широкий функционал | Не работает в оффлайн режиме |
| Разбиение заказов по категориям | Отсутствует возможность сохранения фильтров |
| «Бизнес.ру» | Автоматизированное выставление счета-фактуры по условиям поставки | Средняя оценка пользователей |
| Удобная интеграция | Отсутствует система лояльности |
| Возможность интеграции программы учета с CMS | Платный тариф |

Вывод: проанализировав данные представленные в таблице, необходимо разработать программный продукт, который будет повторять плюсы представленных программ. Приложение должно иметь широкий функционал, должен быть понятный обычному пользователю интерфейс. Необходимо решить такие минусы, как нагрузку на работу компьютера и произведение вычислений и обращений к БД с устройства клиента.

1.2. Обоснование выбора и описание методологии разработки программы решения задачи

Каскадная модель (англ. waterfall model, иногда переводят как модель «Водопад») — модель процесса разработки программного обеспечения, в ней каждый этап разработки, соответствующий стадии жизненного цикла ПО, продолжает предыдущий. То есть для того, чтобы перейти на новый этап, мы полностью должны завершить текущий.

Следуя каскадной модели, разработчик переходит от одной стадии к другой строго последовательно. Сначала полностью завершается этап «определение требований», в результате чего получается список требований к ПО. После того как требования полностью определены, происходит переход к проектированию, в ходе которого создаются документы, подробно описывающие для программистов способ и план реализации указанных требований. После того, как проектирование полностью выполнено, программистами выполняется реализация полученного проекта. На следующей стадии процесса происходит интеграция отдельных компонентов, разрабатываемых различными командами программистов. После того, как реализация и интеграция завершены, производится тестирование и отладка продукта; на этой стадии устраняются все недочёты, появившиеся на предыдущих стадиях разработки. После этого программный продукт внедряется и обеспечивается его поддержка — внесение новой функциональности и устранение ошибок.

Каскадная модель проста и понятна, но не так практична как раньше. В условиях динамично изменяющихся требований, строго структурированный процесс может из преимущества превратиться в помеху на пути успешного завершения разработки системы. Поэтому сегодня водопадная модель применяется преимущественно крупными компаниями для больших и сложных проектов, которые предполагают всеобъемлющий контроль рисков.

V-образная модель применима к системам, которым особенно важно бесперебойное функционирование. Особенностью модели можно считать то, что она направлена на тщательную проверку и тестирование продукта, находящегося уже на первоначальных стадиях проектирования. Стадия тестирования проводится одновременно с соответствующей стадией разработки.

В инкрементной модели полные требования к системе делятся на различные сборки. Терминология часто используется для описания поэтапной сборки ПО. Цикл разделен на более мелкие легко создаваемые модули. Каждый модуль проходит через фазы определения требований, проектирования, кодирования, внедрения и тестирования.

В RAD-модели компоненты или функции разрабатываются несколькими высококвалифицированными командами параллельно, будто несколько мини-проектов. Временные рамки одного цикла жестко ограничены. Созданные модули затем интегрируются в один рабочий прототип.

Итерационная модель жизненного цикла не требует для начала полной спецификации требований. Вместо этого, создание начинается с реализации части функционала, становящейся базой для определения дальнейших требований. Этот процесс повторяется. Версия может быть неидеальна, главное, чтобы она работала.

«Спиральная модель» похожа на инкрементную, но с акцентом на скорость разработки продукта – в ней главная задача как можно быстрее показать пользователям системы работоспособный продукт, тем самым активизируя процесс уточнения и дополнения требований.

Из вышеперечисленного следует что в данной приложении используется каскадная модель.

1.3. Обоснование выбора и описание инструментальных средств разработки программы решения задачи

Microsoft Visual Studio 2019 — это набор инструментов для создания программного обеспечения: от планирования до разработки пользовательского интерфейса, написания кода, тестирования, отладки, анализа качества кода и производительности, развертывания в средах клиентов и сбора данных телеметрии по использованию. Эти инструменты предназначены для максимально эффективной совместной работы; все они доступны в интегрированной среде разработки (IDE) Visual Studio. В качестве языка программирования клиентской части данного проекта был выбран язык C#.

Visual Studio Code (VS Code) — редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, IntelliSense и средства для рефакторинга. Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации. Распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом, но готовые сборки распространяются под проприетарной лицензией.

После сравнения в качестве среды программирование был выбран Microsoft Visual Studio 2019.

C# является языком программирования, который разработан для создания множества приложений, работающих в среде .NET Framework. Язык C# прост, безопасен и объектно-ориентирован. Благодаря множеству нововведений C# обеспечивает возможность быстрой разработки приложений, но при этом сохраняет выразительность и элегантность, присущую С-подобным языков.

C# упрощает разработку компонентов программного обеспечения благодаря нескольким инновационным конструкциям языка, в число которых входят следующие:

* инкапсулированные сигнатуры методов, называемые делегатами, которые поддерживают типобезопасные уведомления о событиях;
* свойства, выступающие в роли методов доступа для закрытых переменных-членов;
* атрибуты с декларативными метаданными о типах во время выполнения;
* встроенные комментарии XML-документации;
* LINQ, предлагающий встроенные возможности запросов в различных источниках данных.

Java — строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Разработка ведётся сообществом, организованным через Java Community Process; язык и основные реализующие его технологии распространяются по лицензии GPL. Права на торговую марку принадлежат корпорации Oracle.

Python — высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным в том плане, что всё является объектами.

После сравнения в качестве языка программирования клиентской части данного проекта был выбран язык C#.

Проанализировав данные, приведённые ниже, в качестве СУБД был выбран Microsoft SQL Server 2019.

Решение Microsoft SQL Server 2019 представляет собой высокопроизводительную платформу обработки и анализа данных для бизнеса любого размера, которая отвечает всем современным требованиям по работе с данными любых типов и поддержке удобной и быстрой разработки приложений. Продукт реализует концепцию комплексной обработки данных и позволяет преобразовывать ценную информацию в знания.

MySQL является решением для малых и средних приложений. Входит в состав серверов WAMP, AppServ, LAMP и в портативные сборки серверов Денвер, XAMPP, VertrigoServ. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

1.4. Содержательная постановка задачи

1.4.1. Описательная модель задачи

Необходимо разработать приложение для учёта деятельности предприятия АО «РАТЕП». Приложение автоматизирует деятельность конструкторского бюро, то есть позволяет управлять состоянием готовности заказов и их позиций, просматривать клиентов и их заказы, формировать отчётность состояния процесса выполнения заказа. Отчётность делается как на бумажном носителе, а также и в электронном виде.

### 1.4.2. Математическая модель задачи

В данной задаче математическая формула отсутствует.

### 1.4.3. Описание входной информации

Таблица 2 - Входная информация

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Идентификатор | Тип данных | Источник данных |
| Код изделия | Num\_product | Числовой | «Договор поставки изделий» |
| Название изделия | Name\_product | Текстовый | «Договор поставки изделий» |

1.4.4. Описание выходной информации

Выходная информация представлена на таблице 3.

Таблица 3 – Выходная информация

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Идентификатор | Тип данных | Приемник данных |
| Название изделия | Name\_product | Текстовый | «Договор поставки изделий» |
| Количество | Quantity | Текстовый | «Отчет по изготовлению изделия» |

1.4.5. Описание бизнес-процессов предприятия

Бизнес-процесс IDEF0 с детализацией представлен в приложении 2.

1.4.6. Описание потоков данных предприятия

Потоки данных DFD с детализацией представлены в приложении 3.

1.4.7. Диаграмма вариантов использования

Диаграмма вариантов использования представлена в приложении 4.

1.4.8. Диаграмма последовательности

Диаграмма последовательности представлена в приложении 5.

1.4.9. Диаграмма взаимосвязей модулей

Диаграмма взаимосвязей модулей представлена в приложении 6.

2. ПРОЕКТНАЯ ЧАСТЬ

2.1. Разработка структуры данных программы

Описания структуры таблиц представлены в таблицах 4-13.

Таблица 4 – описание таблицы «Материал»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| № материала | NumMaterial | Числовой | Первичный ключ  Обязательное поле  Условие назначения- не может быть отрицательным | № материала |
| Наименование | NameMaterial | Текстовый | Размер – 40 символов  Обязательное поле  Неиндексируемое | Наименование материала |
| Стоимость | PriceMaterial | Денежный | Условие назначения- не может быть отрицательным | Стоимость материала |
| № ед. измерения | CodeUnit | Числовой | Внешний ключ  Обязательное поле  Условие назначения- не может быть отрицательным | № ед. измерения материала |

Таблица 5 – описание таблицы «Единица измерения»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| № ед. измерения | CodeUnit | Числовой | Первичный ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | № ед. измерения материала |
| Значение | NameUnit | Текстовый | Размер – 10 символов  Обязательное поле  Неиндексируемое  Не может быть пустым | Наименование материала |

Таблица 6 – описание таблицы «Материальная карта»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| № изделия | NumProduct | Числовой | Внешний ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | № изделия |
| № материала | NumMaterial | Числовой | Внешний ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | № материала |
| Количество | Quantity | Числовой | Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным.  Не может быть пустым | Количество материала |

Таблица 7 – описание таблицы «Сборочная единица»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| № изделия | PAUID | Числовой | Первичный ключ  Обязательное поле  Условие назначения- не может быть отрицательным | № изделия |
| Номер цеха | ManufactoryID | Числовой | Внешний ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | Номер цеха на котором производится деталь |
| Наименование | NameProduct | Текстовый | Размер – 50 символов  Обязательное поле  Неиндексируемое  Не может быть пустым | Наименование изделия |

Таблица 8 – описание таблицы «Структура»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| № материала | NumProductWhere | Числовой | Составной ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | № изделия |

Продолжение таблицы 8 – описание таблицы «Структура»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| № материала | NumProductWhat | Числовой | Составной ключ  Обязательное поле  Размер поля - длинное целое | № изделия |
| Количество | QuantityProduct | Числовой | Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным.  Не может быть пустым | Количество используемых изделий |

Таблица 9 – описание таблицы «Карта трудоемкости»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| № изделия | NumProduct | Числовой | Внешний ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | № изделия |
| Код должности | CodePost | Числовой | Внешний ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | Код должности |
| № операции | NumOperation | Числовой | Внешний ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | № операции |
| Этапы разработки | OrderOperation | Числовой | Условие назначения- не может быть отрицательным.  Не может быть пустым | Этап разработки по порядку |
| Затраченное время | TimeSpent | Время | Не может быть пустым | Затраченное время на данный этап |

Таблица 10 – описание таблицы «Операция»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| № операции | NumOperation | Числовой | Первичный ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | № операции |
| Наименование | NameOperation | Текстовый | Размер – 30 символов  Обязательное поле  Неиндексируемое  Не может быть пустым | Наименование операции |

Таблица 11 – описание таблицы «Должность»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| Код должности | CodePost | Числовой | Первичный ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | Код должности |
| Наименование | NamePost | Текстовый | Размер – 30 символов  Обязательное поле  Неиндексируемое  Не может быть пустым | Наименование должности |
| Зарплата за час | SalaryPerHours | Числовой | Размер поля - длинное целое  Условие назначения- не может быть отрицательным | Зарплата за час работы |

Таблица 12 – описание таблицы «Сотрудник»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| Код сотрудника | EmployeeID | Числовой | Первичный ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | Код сотрудника |

Продолжение таблицы 12 – описание таблицы «Сотрудник»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| Код должности | CodePost | Числовой | Внешний ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | Код должности |
| Номер цеха | ManufactoryID | Числовой | Внешний ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | Номер цеха на котором производится деталь |

Таблица 13 – описание таблицы «Пользователь»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| Номер аккаунта | AccountID | Числовой | Первичный ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | Код аккаунта |
| Логин | Login | Строковый | Размер – 30 символов | Логин пользователя |
| Пароль | Password | Строковый | Размер – 1024 символов | Пароль пользователя |
| ФИО | FullName | Строковый | Размер – 300 символов | ФИО пользователя |

Таблица 14 – описание таблицы «Клиент»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| Номер клиента | ClientID | Числовой | Первичный ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | Код клиента |

Продолжение таблицы 14 – описание таблицы «Клиент»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| Номер аккаунта | AccountID | Числовой | Внешний ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | Код аккаунта |

Таблица 15 – описание таблицы «Контракт»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| Номер контракта | ContractID | Числовой | Первичный ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | Код контракта |
| Номер клиента | ClientID | Числовой | Внешний ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | Код клиента |
| Номер сотрудника | EmployeeID | Числовой | Внешний ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | Код сотрудника |
| Дата регистрации | RegDate | Дата | Обязательное поле | Дата регистрации заказа |

Таблица 16 – описание таблицы «Заказ»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| Номер заказа | OrderID | Числовой | Первичный ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | Код заказа |
| Номер контракта | ContractID | Числовой | Внешний ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | Код контракта |

Продолжение таблицы 16 – описание таблицы «Заказ»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| Номер клиента | ClientID | Числовой | Внешний ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | Код клиента |
| Номер сотрудника | EmployeeID | Числовой | Внешний ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | Код сотрудника |
| Дата регистрации | RegDate | Дата | Обязательное поле | Дата регистрации заказа |
| Дата завершения | CompletionDate | Дата | Необязательное поле | Дата завершения заказа |

Таблица 17 – описание таблицы «Позиции заказа»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название поля | Идентификатор | Тип данных | Условия и ограничения | Описание |
| Номер позиции заказа | PosID | Числовой | Первичный ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | Код позиции заказа |
| Номер заказа | OrderID | Числовой | Внешний ключ  Обязательное поле  Размер поля - длинное целое  Условие назначения- не может быть отрицательным | Код заказа |
| № изделия | PAUID | Числовой | Внешний ключ  Обязательное поле  Условие назначения- не может быть отрицательным | № изделия |
| Количество | Ammount | Числовой | Обязательное поле  Условие назначения- не может быть отрицательным | Количество изделий в позиции заказа |
| Дата завершения | CompletionDate | Дата | Необязательное поле | Дата завершения заказа |

2.2. Логическая модель программы

Логическая модель программы представлена в приложении 7.

Физическая модель представлена в приложении 8.

2.3. Разработка пользовательского интерфейса программы

2.3.1. Разработка требований к интерфейсу

Основные требования к пользовательскому интерфейсу:

* понятность и логичность;
* обеспечение высокой скорости работы пользователя;
* обеспечение защиты от ошибок

2.3.2. Проектирование и описание элементов управления

Макет страницы «Главное меню» представлен на рисунке 1.

Интерфейс страницы «Главное меню» представлен на рисунке 2.



Рисунок 1– Макет страницы «Главное меню»

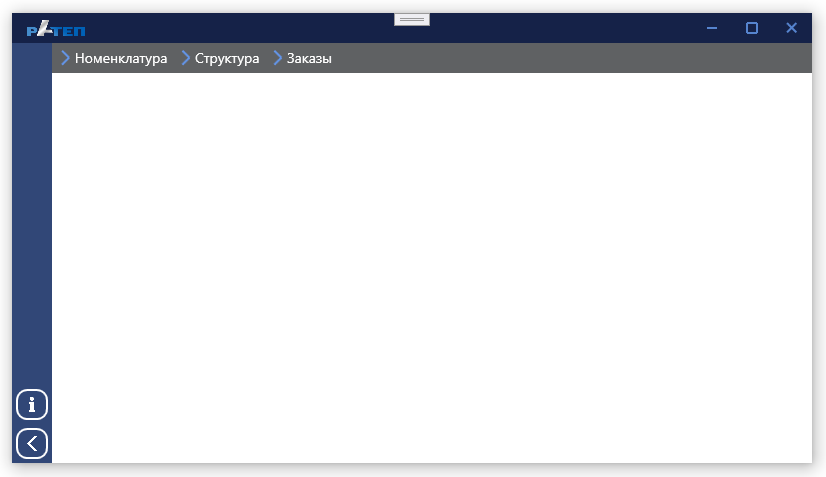


Рисунок 2 – Интерфейс страницы «Главное меню»

Таблица 18 – Компоненты и свойства страницы «Главное меню»

|  |  |  |
| --- | --- | --- |
| Компонент | Наименование компонента | Свойства компонента |
| Grid | - | - |
| Frame | MainFrame | NavigationUIVisibility="Hidden" Source="../Pages/Autorization.xaml" |
| Grid | - | - |
| Border | - | Background="#314777" Grid.RowSpan="2" |
| StackPanel | - | Orientation="Vertical" VerticalAlignment="Bottom" |
| Button | - | Width="40"  Height="40"  Style="{DynamicResource MainButton}" HorizontalAlignment="Left"  Margin="5, 0, 0, 0" |
| Image | - | Source=”../../src/Images/Info.png" |
| Button | - | Width="40"  Height="40"  Style="{DynamicResource MainButton}" HorizontalAlignment="Left"  Margin="5, 0, 0, 0" |
| Image | - | Source=”../../src/Images/Back.png" |
| Border | - | Grid.Row="0" |

Продолжение таблицы 18 – Компоненты и свойства страницы «Главное меню»

|  |  |  |
| --- | --- | --- |
| StackPanel | - | Orientation="Horizontal" |
| Button | - | Style="{DynamicResource TabButton}"  Content="Номенклатура" |
| Button | - | Style="{DynamicResource TabButton}"  Content="Структура" |
| Button | - | Style="{DynamicResource TabButton}"  Content="Заказы" |
| Frame | DataFrame | Grid.Row="1"  Grid.Column="1" NavigationUIVisibility="Hidden" |

Макет страницы «Авторизация» представлен на рисунке 3.

Интерфейс страницы «Авторизация» представлен на рисунке 4.

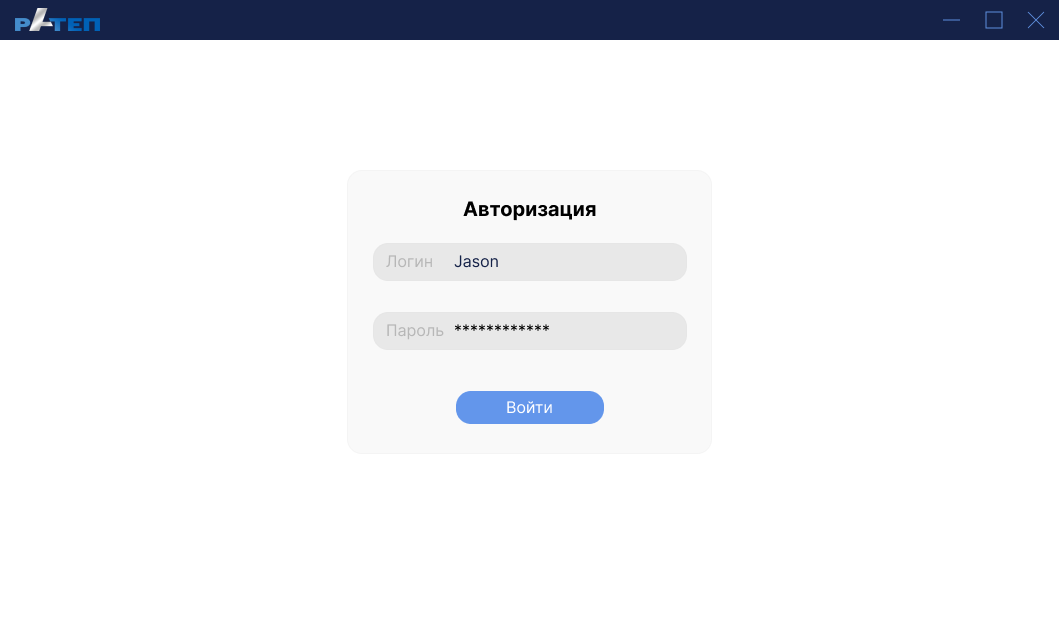


Рисунок 3– Макет страницы «Авторизация»

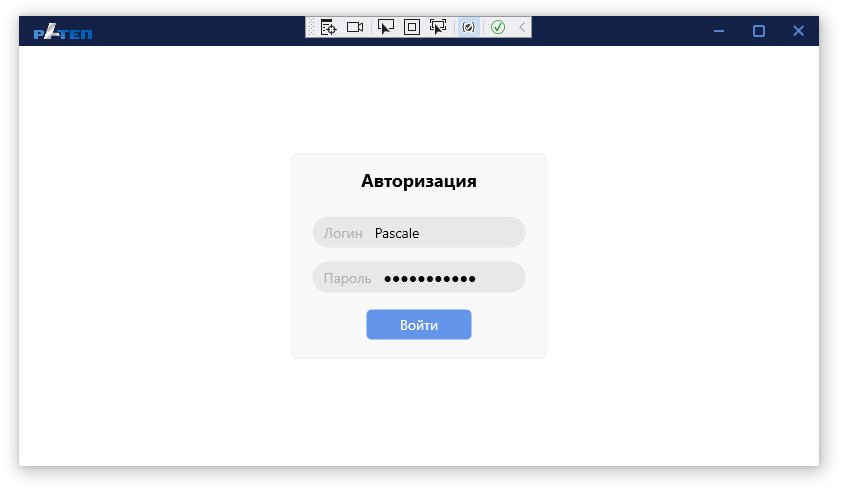


Рисунок 4 – Интерфейс страницы «Авторизация»

Таблица 19 – Компоненты и свойства страницы «Авторизация»

|  |  |  |
| --- | --- | --- |
| Компонент | Наименование компонента | Свойства компонента |
| Grid | - | - |
| Border | - | HorizontalAlignment="Center" VerticalAlignment="Center"  Width="255" Height="205"  BorderThickness="1"  CornerRadius="5"  BorderBrush="#E8E8E8"  Background="#F9F9F9" |
| Grid | - |  |
| Label | - | Style="{DynamicResource HeaderText}"  Grid.Row="0"  VerticalAlignment="Center" HorizontalAlignment="Center" |
| Border | - | Background="#E8E8E8"  Margin="20, 0, 20, 0"  BorderThickness="1" BorderBrush="#E0E0E0"  CornerRadius="15"  Grid.Row="1" |

Продолжение таблицы 19 – Компоненты и свойства страницы «Авторизации»

|  |  |  |
| --- | --- | --- |
| Компонент | Наименование компонента | Свойства компонента |
| StackPanel | - | Orientation="Horizontal" |
| Label | - | VerticalAlignment="Center" Foreground="#B0B0B0"  Margin="5, 0, 0, 0" |
| TextBox | LoginTxtBx | Background="Transparent"  Width="225"  VerticalAlignment="Center" BorderThickness="0" Margin="5" |
| Label | - | Style="{DynamicResource HeaderText}"  Grid.Row="0"  VerticalAlignment="Center" HorizontalAlignment="Center" |
| Border | - | Background="#E8E8E8"  Margin="20, 0, 20, 0"  BorderThickness="1" BorderBrush="#E0E0E0"  CornerRadius="15"  Grid.Row="1"  HorizontalAlignment="Center" VerticalAlignment="Center" |
| StackPanel | - | Orientation="Horizontal" |
| Label | - | VerticalAlignment="Center" Foreground="#B0B0B0"  Margin="5, 0, 0, 0" |
| TextBox | PasswordTxtBx | Background="Transparent"  Width="225"  VerticalAlignment="Center" BorderThickness="0" Margin="5" |

Макет страницы «Заказы» представлен на рисунке 5.

Интерфейс страницы «Заказы» представлен на рисунке 6.

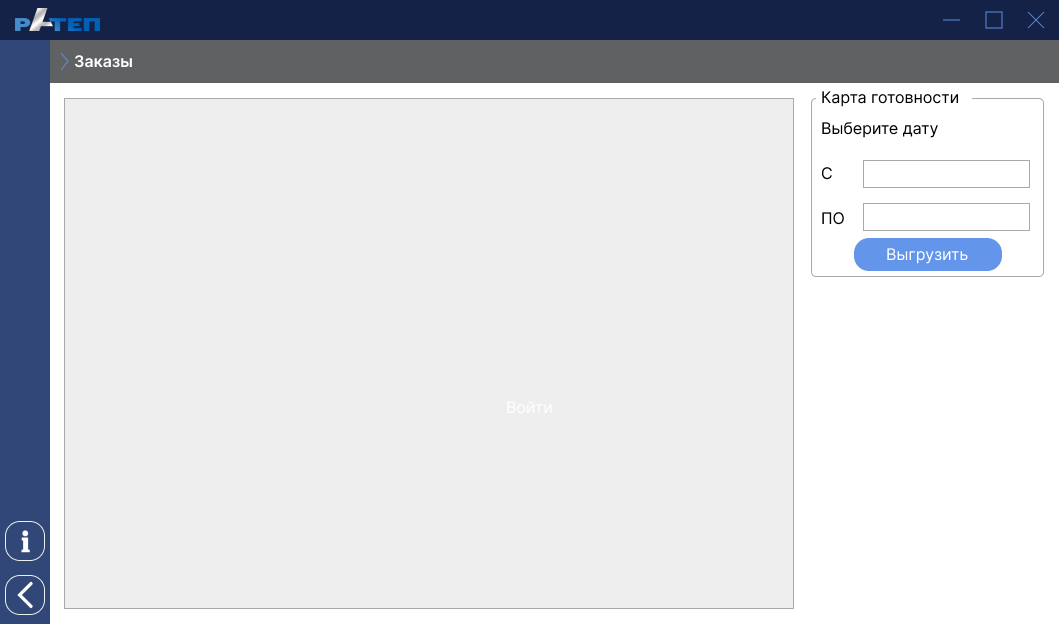


Рисунок 5– Макет страницы «Заказы»

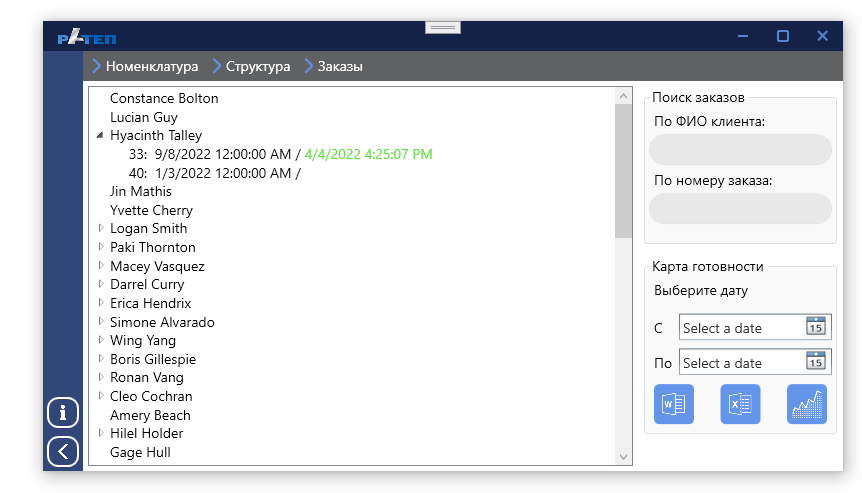


Рисунок 6 – Интерфейс страницы «Заказы»

Таблица 20 – Компоненты и свойства страницы «Заказы»

|  |  |  |
| --- | --- | --- |
| Компонент | Наименование компонента | Свойства компонента |
| Grid | - | - |
| TreeView | OrderData | Margin="5" Grid.Row="1" |
| HierarchicalDataTemplate | - | ItemsSource="{Binding Path=Nodes}" |
| StackPanel | - | Orientation="Horizontal" |
| TextBlock | - | Text="{Binding Description}" |
| TextBlock | - | Text="{Binding RegDate} |

Продолжение таблицы 20 – Компоненты и свойства страницы «Заказы»

|  |  |  |
| --- | --- | --- |
| TextBlock | - | Text="{Binding EndDate}"  Foreground="#FF5FE03D" |
| StackPanel | - | Grid.Column="1"  Orientation="Vertical" |
| GroupBox | - | Height="auto" Margin="5"  Header="Поиск заказов"  Background="#F9F9F9" |
| Grid | - | - |
| StackPanel | - | Margin="0, 0, 0, 15" |
| StackPanel | - | - |
| Label | - | Content="По ФИО клиента:" |
| Border | - | Background="#E8E8E8" BorderThickness="1" BorderBrush="#E0E0E0" CornerRadius="15" |
| Grid | - | Width="auto" |
| TextBox | ClientFullNameSearch | Background="Transparent"  Width="auto"  VerticalAlignment="Center" BorderThickness="0" Margin="5" |
| StackPanel | - | - |
| Label | - | Content="По номеру заказа:" |
| Border | - | Background="#E8E8E8" BorderThickness="1" BorderBrush="#E0E0E0" CornerRadius="15" |
| Grid | - | Width="auto" |
| TextBox | OrderNumSearch | Background="Transparent"  Width="auto"  VerticalAlignment="Center" BorderThickness="0" Margin="5" |
| GroupBox | - | Header="Карта готовности"  Background="#F9F9F9" |

Продолжение таблицы 20 – Компоненты и свойства страницы «Заказы»

|  |  |  |
| --- | --- | --- |
| Grid | - | - |
| Label | - | Grid.ColumnSpan="2" |
| Label | - | Grid.Row="1" VerticalAlignment="Center" |
| DatePicker | FromDate | Grid.Row="1"  Grid.Column="1" VerticalAlignment="Center" |
| Label | - | Grid.Row="1" VerticalAlignment="Center" |
| DatePicker | ToDate | Grid.Row="1"  Grid.Column="1" VerticalAlignment="Center" |
| Grid | - | Grid.Row="3"  Grid.ColumnSpan="2" |
| Button | - | Width="40"  Height="40"  Style="{DynamicResource MainButton}" HorizontalAlignment="Left"  Margin="5, 0, 0, 0" |
| Image | - | Source="../../src/Images/Word.png" |
| Button | - | Width="40"  Height="40"  Style="{DynamicResource MainButton}" HorizontalAlignment="Left"  Margin="5, 0, 0, 0" |
| Image | - | Source="../../src/Images/Excel.png" |
| Button | - | Width="40"  Height="40"  Style="{DynamicResource MainButton}" HorizontalAlignment="Left"  Margin="5, 0, 0, 0" |
| Image | - | Source="../../src/Images/Chart.png" |

Макет страницы «Позиции заказа» представлен на рисунке 7.

Интерфейс страницы «Позиции заказа» представлен на рисунке 8.

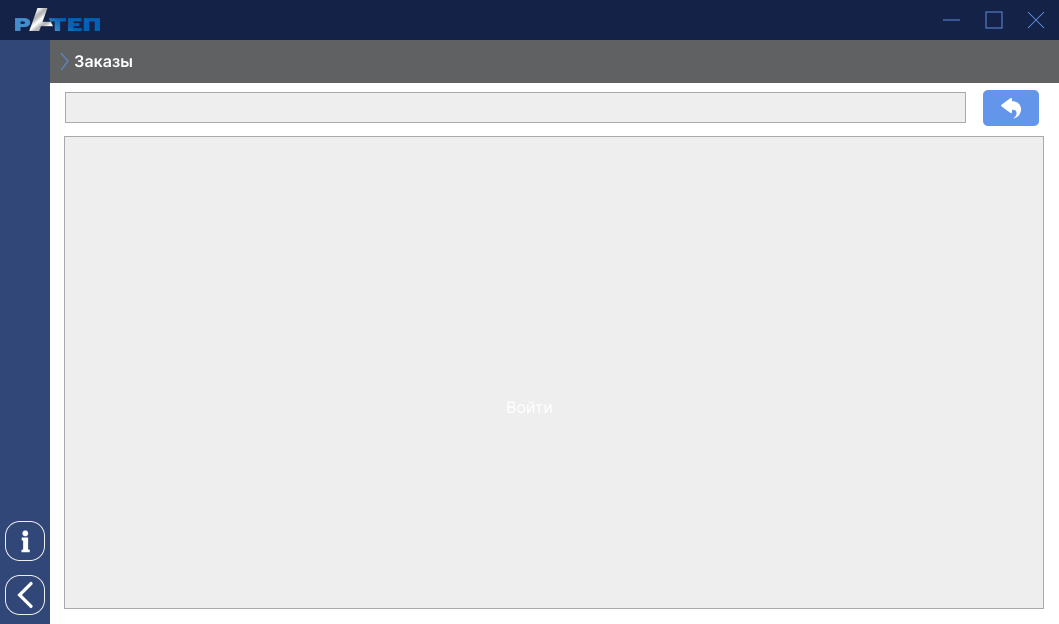


Рисунок 7– Макет страницы «Позиции заказа»

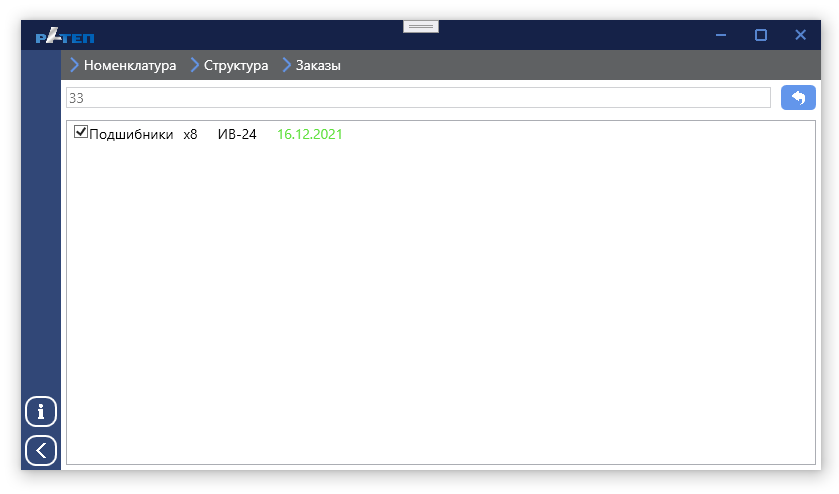


Рисунок 8 – Интерфейс страницы «Позиции заказа»

Таблица 21 – Компоненты и свойства страницы «Позиции заказа»

|  |  |  |
| --- | --- | --- |
| Компонент | Наименование компонента | Свойства компонента |
| Grid | - | - |
| TextBox | SelectedTxtBx | IsEnabled="False"  Margin="5"  VerticalAlignment="Center" |
| Button | - | Style="{DynamicResource MainButton}" HorizontalAlignment="Left" |

Продолжение таблицы 21 – Компоненты и свойства страницы «Позиции заказа»

|  |  |  |
| --- | --- | --- |
| Image | - | Source=”../../src/Images/back2.png" |
| TreeView | OrderDetailData | Margin="5" Grid.Row="1" |
| HierarchicalDataTemplate | - | ItemsSource="{Binding Path=Nodes}" |
| StackPanel | - | Orientation="Horizontal" HorizontalAlignment="Left" |
| TextBlock | - | Text="{Binding posID}"  Visibility="Collapsed" |
| CheckBox | - | IsChecked="{Binding Checked}" |
| TextBlock | - | Text="{Binding Name}" |
| TextBlock | - | Text="x" Margin="10, 0, 0, 0" |
| TextBlock | - | Text="{Binding Quantity}" |
| TextBlock | - | Text="{Binding Manufactory}"  Margin="20, 0, 0, 0" |
| TextBlock | - | Text="{Binding EndDate}"  Margin="20, 0, 0, 0"  Foreground="#FF5FE03D" |

Макет страницы «Статистика» представлен на рисунке 9.

Интерфейс страницы «Статистика» представлен на рисунке 10.

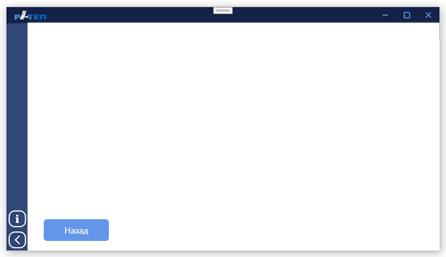


Рисунок 9– Макет страницы «Статистика»



Рисунок 10 – Интерфейс страницы «Статистика»

Таблица 22 – Компоненты и свойства страницы «Статистика»

|  |  |  |
| --- | --- | --- |
| Компонент | Наименование компонента | Свойства компонента |
| Grid | - | Margin="15" |
| WpfPlot | ChartPlace | - |
| Button | - | Width="105"  Height="35"  Grid.Row="1"  Style="{DynamicResource MainButton}"  HorizontalAlignment="Left" |

2.4. Листинг программы

Логика класса «ApiControl.cs» представлена в листингах 1 – 2.

using System;

using System.Collections.Generic;

using System.Net;

using System.Net.Http;

using System.Text;

using System.Threading.Tasks;

namespace Ratep.Class

Листинг 1 - логика класса «ApiControl.cs»

{

public class ApiControl

{

public static async Task<string> GetRequest(string method)

{

HttpClient client = new HttpClient();

var response = await client.GetAsync(App.URL + method);

var responseString = await response.Content.ReadAsStringAsync();

return responseString;

}

public static async Task<string> POSTRequest(string args, Dictionary<string, string> body = null)

{

HttpClient client = new HttpClient();

var content = new FormUrlEncodedContent(body == null ? new Dictionary<string, string>() : body);

var response = await client.PostAsync(App.URL + args, content);

var responseString = await response.Content.ReadAsStringAsync();

return responseString;

}

}

}

Листинг 2 - логика класса «ApiControl.cs»

Логика класса «ApiError.cs» представлена в листингах 3 – 4.

using System.Windows;

namespace Ratep.Class

{

public enum ApiErrors

{

ErrorNotFound = 0,

DBNotFound = 404,

UserNotFound = 405,

ApiConnectionFailed = 406,

PermissionDenied = 504,

ErrorData = 505

}

public class ApiError

{

public static ApiErrors GetErrorByCode(int code)

{

switch (code)

{

case (int)ApiErrors.DBNotFound:

return ApiErrors.DBNotFound;

case (int)ApiErrors.UserNotFound:

return ApiErrors.UserNotFound;

case (int)ApiErrors.PermissionDenied:

return ApiErrors.PermissionDenied;

case (int)ApiErrors.ErrorData:

return ApiErrors.ErrorData;

}

return ApiErrors.ErrorNotFound;

}

public static void ShowError(ApiErrors error)

{

Листинг 3 - логика класса «ApiError.cs»

switch (error)

{

case ApiErrors.ErrorNotFound:

MessageBox.Show("Произошла непредвиденная ошибка!", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

break;

case ApiErrors.DBNotFound:

MessageBox.Show("Не удалось установить соединение с базой данных. Пожалуйста, повторите попытку позже.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

break;

case ApiErrors.UserNotFound:

MessageBox.Show("Указаный пользователь не найден.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

break;

case ApiErrors.PermissionDenied:

MessageBox.Show("Недостаточно прав, доступ запрещён!", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

break;

case ApiErrors.ApiConnectionFailed:

MessageBox.Show("Не удалось установить соединение с сервером, пожалуйста повторите попытку позже.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

break;

case ApiErrors.ErrorData:

MessageBox.Show("Данные введены неверно!", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

break;

}

}

}

}

Листинг 4 - логика класса «ApiError.cs»

Логика класса «Cryptograph.cs» представлена в листинге 5.

using System;

using System.Security.Cryptography;

using System.Text;

namespace Ratep.Class

{

public class Cryptograph

{

public static string EncryptStringToSHA256(string value)

{

StringBuilder Sb = new StringBuilder();

using (SHA256 hash = SHA256Managed.Create())

{

Encoding enc = Encoding.UTF8;

Byte[] result = hash.ComputeHash(enc.GetBytes(value));

foreach (Byte b in result)

Sb.Append(b.ToString("x2"));

}

return Sb.ToString();

}

}

}

Листинг 5 - логика класса «Cryptograph.cs»

Логика страницы «Autorization.xaml.cs» представлена в листингах 6-7.

using Ratep.Models;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using Ratep.Class;

using System.Collections.Generic;

using System.IO;

using System;

using Ratep.Views.Pages;

using System.Configuration;

using Ratep.Class;

using System.Threading.Tasks;

namespace Ratep.Pages

{

/// <summary>

/// Логика взаимодействия для Autorization.xaml

/// </summary>

public partial class Autorization : Page

{

public Autorization()

{

InitializeComponent();

}

public async void Button\_Click(object sender, RoutedEventArgs e)

{

string response = await Authorizate(LoginTxtBx.Text, PasswordTxtBx.Password);

if (response.Length == 3)

{

ApiError.ShowError(ApiError.GetErrorByCode(Convert.ToInt32(response)));

return;

}

if (response != null)

{

App.Token = response;

NavigationService.Navigate(new Interface());

}

}

public static async Task<string> Authorizate(string Login, string Password)

{

string request;

string CryptedLogin = Cryptograph.EncryptStringToSHA256(Login);

string CryptedPassword = Cryptograph.EncryptStringToSHA256(Password);

try

{

request = await ApiControl.POSTRequest($"Users?CryptedLogin={CryptedLogin}&CryptedPassword={CryptedPassword}");

}

catch (Exception ex)

{

Листинг 6 - логика страницы «Autorization.xaml.cs»

return ((int)ApiErrors.ApiConnectionFailed).ToString();

}

return request;

}

private void DatabaseClick(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(new DatabaseConnectionSettings());

}

}

}

Листинг 7 - логика страницы «Autorization.xaml.cs»

Логика страницы «Interface.xaml.cs» представлена в листингах 8 - 9.

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media.Animation;

namespace Ratep.Pages

{

/// <summary>

/// Логика взаимодействия для Interface.xaml

/// </summary>

public partial class Interface : Page

{

static public Nomenclature NomenclaturePage = new Nomenclature();

static public StructurePage StructurePage = new StructurePage();

static public OrderPage orderPage = new OrderPage();

static public Frame frame;

public Interface()

{

InitializeComponent();

frame = DataFrame;

}

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

{

if (StructurePage.MiscSeance != null) { StructurePage.MiscSeance = null; }

DataFrame.Navigate(NomenclaturePage);

NomenclaturePage.UpdateData();

}

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

{

DataFrame.Navigate(StructurePage);

}

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

{

new Info().ShowDialog();

}

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

{

Листинг 8- логика страницы «Interface.xaml.cs»

NavigationService.Navigate(new Autorization());

}

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

{

DataFrame.Navigate(orderPage);

}

}

}

Листинг 9- логика страницы «Interface.xaml.cs»

Логика страницы «OrderPage.xaml.cs» представлена в листингах 10 - 15.

using Microsoft.Win32;

using Ratep;

using Ratep.Models.ApiModels;

using Ratep.Views.Pages;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.IO;

using System.Linq;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using Word = Microsoft.Office.Interop.Word;

using Excel = Microsoft.Office.Interop.Excel;

using Ratep.Class;

using Newtonsoft.Json;

namespace Ratep.Pages

{

public partial class OrderPage : Page

{

static public TreeView orderData { get; set; }

public static ObservableCollection<Client> SavedData { get; set; }

public OrderPage()

{

InitializeComponent();

orderData = OrderData;

UpdateTreeView();

}

public static async void UpdateTreeView()

{

string response;

try

{

response = await ApiControl.GetRequest($"OrderPositions/GetClientsOrderList?Token={App.Token}");

}

catch (Exception ex)

{

ApiError.ShowError(ApiErrors.ApiConnectionFailed);

return;

}

SavedData = JsonConvert.DeserializeObject<ObservableCollection<Client>>(response);

orderData.ItemsSource = SavedData;

Листинг 10 - логика страницы « OrderPage.xaml.cs»

}

private void OrderData\_PreviewMouseDoubleClick(object sender, System.Windows.Input.MouseButtonEventArgs e)

{

Contract contractData = (Contract)OrderData.SelectedItem;

if (contractData == null ) return;

NavigationService.Navigate(new OrderDetail(contractData.Order));

}

public static List<OrderPosition> filterCompletionOrderPositionByDate(List<OrderPosition> Data, DateTime fromDate, DateTime toDate)

{

List<OrderPosition> result = new List<OrderPosition>();

foreach (OrderPosition item in Data)

{

if (item.CompletionDate > fromDate && item.CompletionDate < toDate)

{

result.Add(item);

}

}

return result;

}

public static async Task<int> PrintExcel(Order orderDB, string savePath = null)

{

if (savePath != null) { return 1; }

if (orderDB == null) { return -1; }

SaveFileDialog savedialog = new SaveFileDialog();

savedialog.Title = "Сохранить файл как...";

savedialog.OverwritePrompt = true;

savedialog.CheckPathExists = true;

savedialog.Filter = "Excel документ (\*.xls)|\*.xlsx";

if (savedialog.ShowDialog() == true)

{

await Task.Run(() =>

{

List<OrderPosition> orderPos = orderDB.OrderPositions.ToList();

try

{

Excel.Application excel = new Excel.Application();

Excel.Workbook wb = excel.Workbooks.Add(System.Reflection.Missing.Value);

Excel.Worksheet ws = (Excel.Worksheet)wb.Sheets[1];

((Excel.Range)ws.Cells[1, 1]).Value2 = "№Детали";

((Excel.Range)ws.Cells[1, 2]).Value2 = "Наименование";

((Excel.Range)ws.Cells[1, 3]).Value2 = "Кол-во";

((Excel.Range)ws.Cells[1, 4]).Value2 = "Цех";

((Excel.Range)ws.Cells[1, 5]).Value2 = "Дата завершения";

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

{

((Excel.Range)ws.Cells[i + 2, 1]).Value2 = Convert.ToString(orderPos[i].Pau.Pauid);

Листинг 11- логика страницы «OrderPage.xaml.cs»

((Excel.Range)ws.Cells[i + 2, 2]).Value2 = orderPos[i].Pau.NameProduct;

((Excel.Range)ws.Cells[i + 2, 3]).Value2 = Convert.ToString(orderPos[i].Ammount);

((Excel.Range)ws.Cells[i + 2, 4]).Value2 = orderPos[i].Pau.Manufactory.Name;

((Excel.Range)ws.Cells[i + 2, 5]).Value2 = orderPos[i].CompletionDate.ToString();

}

wb.SaveAs(savedialog.FileName);

wb = null;

excel.Quit();

MessageBox.Show("Файл успешно сохранён");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

});

}

return 1;

}

public static async Task<int> UnloadFileData(DateTime? FromDate, DateTime? ToDate, Contract orderDB, string savePath = null)

{

if (orderDB == null) { return -1; }

if (savePath != null) { return 1; }

SaveFileDialog savedialog = new SaveFileDialog();

savedialog.Title = "Сохранить файл как...";

savedialog.OverwritePrompt = true;

savedialog.CheckPathExists = true;

savedialog.Filter = "Word документ (\*.doc)|\*.doc";

Word.Document doc = null;

Word.Application app = new Word.Application();

if (savedialog.ShowDialog() == true)

{

await Task.Run(() =>

{

DateTime? fromDate = null;

DateTime? toDate = null;

if (orderDB == null) return;

List<OrderPosition> orderPositions = orderDB.Order.OrderPositions.ToList();

List<OrderPosition> completedOrderPositions = orderDB.Order.OrderPositions.Where(c => c.CompletionDate != null).ToList();

Client client = SavedData.FirstOrDefault(c => c.ClientId == orderDB.ClientId);

if (FromDate != null && ToDate != null &&

FromDate < ToDate)

{

fromDate = FromDate;

toDate = ToDate;

completedOrderPositions = filterCompletionOrderPositionByDate(completedOrderPositions, Convert.ToDateTime(FromDate), Convert.ToDateTime(ToDate));

}

Try

Листинг 12 - логика страницы «OrderPage.xaml.cs»

{

string source = $@"{Directory.GetCurrentDirectory()}\orderPattern.docx";

doc = app.Documents.Open(source);

doc.Activate();

Word.Bookmarks bookmarks = doc.Bookmarks;

Word.Table Structure = bookmarks["OrderCard"].Range.Tables.Add(bookmarks["OrderCard"].Range, completedOrderPositions.Count + 1, 5);

Structure.Borders.InsideLineStyle = Word.WdLineStyle.wdLineStyleSingle;

Structure.Borders.OutsideLineStyle = Word.WdLineStyle.wdLineStyleSingle;

bookmarks["OrderCardNum"].Range.Text = Convert.ToString($"{orderDB.Order.OrderId}{completedOrderPositions.Count}");

bookmarks["DocumentDate"].Range.Text = DateTime.Now.ToString("dd-MM-yyyy");

bookmarks["OrderNum"].Range.Text = Convert.ToString(orderDB.Order.OrderId);

bookmarks["EmployeeAccept"].Range.Text = client.Account.FullName;

bookmarks["CompletedPosAmmount"].Range.Text = Convert.ToString(completedOrderPositions.Count);

bookmarks["PosAmmount"].Range.Text = Convert.ToString(orderPositions.Count);

bookmarks["Date"].Range.Text = fromDate == null ? "" : $"{Convert.ToDateTime(fromDate).ToString("dd-MM-yyyy")}/{Convert.ToDateTime(toDate).ToString("dd-MM-yyyy")}";

Structure.Cell(1, 1).Range.Text = "№Детали";

Structure.Cell(1, 2).Range.Text = "Наименование";

Structure.Cell(1, 3).Range.Text = "Кол-во";

Structure.Cell(1, 4).Range.Text = "Цех";

Structure.Cell(1, 5).Range.Text = "Дата завершения";

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

{

Structure.Cell(i + 2, 1).Range.Text = Convert.ToString(completedOrderPositions[i].Pau.Pauid);

Structure.Cell(i + 2, 2).Range.Text = completedOrderPositions[i].Pau.NameProduct;

Structure.Cell(i + 2, 3).Range.Text = Convert.ToString(completedOrderPositions[i].Ammount);

Structure.Cell(i + 2, 4).Range.Text = completedOrderPositions[i].Pau.Manufactory.Name;

Structure.Cell(i + 2, 5).Range.Text = completedOrderPositions[i].CompletionDate.ToString();

}

doc.SaveAs2(savedialog.FileName);

doc.Close(Word.WdSaveOptions.wdDoNotSaveChanges);

doc = null;

app.Quit(Word.WdSaveOptions.wdDoNotSaveChanges);

MessageBox.Show("Файл успешно сохранён");

}

catch (Exception ex)

{

Листинг 13 - логика страницы «OrderPage.xaml.cs»

MessageBox.Show(ex.Message);

}

});

}

return 1;

}

private async void UnloadOrderStatus\_Click(object sender, RoutedEventArgs e)

{

if (orderData.SelectedItem == null) { MessageBox.Show("Выберите заказ."); return; }

try

{

Contract contractData = (Contract)OrderData.SelectedItem;

await UnloadFileData(FromDate.SelectedDate, ToDate.SelectedDate, contractData);

} catch (Exception ex)

{

MessageBox.Show("Выберите заказ."); return;

}

}

public static ObservableCollection<Client> SearchByFIO(ObservableCollection<Client> data, string searched)

{

return new ObservableCollection<Client>(data.Where(c => c.Account.FullName.ToLower().Contains(searched.ToLower())));

}

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

{

ObservableCollection<Client> searchedData = SearchByFIO(SavedData, ClientFullNameSearch.Text);

if (searchedData.Count == 0)

{

MessageBox.Show("Клиенты не найдены!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Asterisk);

}

orderData.ItemsSource = searchedData;

}

public static ObservableCollection<Client> SearchByNum(ObservableCollection<Client> data, string OrderID)

{

ObservableCollection<Client> searchedData = new ObservableCollection<Client>();

foreach (Client item in data)

{

if (item.Contracts.Where(c => c.Order.OrderId.ToString().Contains(OrderID)).Any())

{

searchedData.Add(item);

}

}

return searchedData;

Листинг 14 - логика страницы «OrderPage.xaml.cs»

}

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

{

ObservableCollection<Client> searchedData = SearchByNum(SavedData, OrderNumSearch.Text);

if (searchedData.Count == 0)

{

MessageBox.Show("Заказы не найдены!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Asterisk);

}

orderData.ItemsSource = searchedData;

}

private void ChartUnloadClick(object sender, RoutedEventArgs e)

{

if (FromDate.SelectedDate != null && ToDate.SelectedDate != null)

{

if (FromDate.SelectedDate < ToDate.SelectedDate)

{

List<OrderPosition> orders = new List<OrderPosition>();

foreach (Client client in SavedData)

{

foreach (Contract contract in client.Contracts)

{

foreach (OrderPosition order in contract.Order.OrderPositions)

{

if (order != null)

{

orders.Add(order);

}

}

}

}

NavigationService.Navigate(new ChartBuilder(orders,

Convert.ToDateTime(FromDate.SelectedDate), Convert.ToDateTime(ToDate.SelectedDate)));

}

}

}

private async void ExcelUnloadClick(object sender, RoutedEventArgs e)

{

if (orderData.SelectedItem == null) { MessageBox.Show("Выберите заказ."); return; }

try

{

Contract contractData = (Contract)OrderData.SelectedItem;

await PrintExcel(contractData.Order);

} catch (Exception ex)

{

MessageBox.Show("Выберите заказ."); return;

}

}

}

Листинг 15 - логика страницы «OrderPage.xaml.cs»

Логика страницы «OrderDetail.xaml.cs» представлена в листингах 16 - 18.

using Newtonsoft.Json;

using Ratep.Class;

using Ratep.Models.ApiModels;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Threading.Tasks;

using System.Web.Script.Serialization;

using System.Windows;

using System.Windows.Controls;

namespace Ratep.Pages

{

/// <summary>

/// Логика взаимодействия для OrderDetail.xaml

/// </summary>

public partial class OrderDetail : Page

{

public TextBox selectedOrderDescription { get; set; }

public ListView orderDetailData { get; set; }

static public Order \_selectedOrder { get; set; }

static public bool isChecked { get; set; }

public ObservableCollection<OrderPosition> OrderPositions { get; set; }

public OrderDetail(Order order)

{

InitializeComponent();

selectedOrderDescription = SelectedTxtBx;

orderDetailData = OrderDetailData;

\_selectedOrder = order;

LoadOrderData(order);

}

public async void LoadOrderData(Order order)

{

selectedOrderDescription.Text = order.OrderId.ToString();

OrderPositions = new ObservableCollection<OrderPosition>(order.OrderPositions);

orderDetailData.ItemsSource = OrderPositions;

}

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

{

NavigationService.GoBack();

}

public async void CloseOrder(object sender)

{

if (isChecked == true) { isChecked = false; return; }

MessageBoxResult boxResult = MessageBox.Show("Вы уверены, что хотите закрыть товар?", "Уведомление!", MessageBoxButton.YesNo);

if (boxResult == MessageBoxResult.No) { isChecked = true; ((CheckBox)sender).IsChecked = true; return; }

StackPanel currentItem = (StackPanel)((CheckBox)sender).Parent;

Листинг 16 - логика страницы «OrderDetail.xaml.cs»

int posID = Convert.ToInt32(((TextBlock)currentItem.Children[0]).Text);

if (posID == -1) { isChecked = true; ((CheckBox)sender).IsChecked = true; return; }

else

{

string response;

try

{

response = await CloseOrderApiRequest(posID, App.Token);

}

catch (Exception ex)

{

ApiError.ShowError(ApiErrors.ApiConnectionFailed);

return;

}

if (Convert.ToInt32(response) != 1)

{

isChecked = true; ((CheckBox)sender).IsChecked = false;

ApiError.ShowError(ApiError.GetErrorByCode(Convert.ToInt32(response)));

return;

}

else

{

((TextBlock)currentItem.Children[6]).Text = DateTime.Now.ToString("dd.MM.yyyy");

MessageBox.Show("Позиция в заказе успешно закрыта!", "Уведомление!", MessageBoxButton.OK, MessageBoxImage.Asterisk);

}

OrderPage.UpdateTreeView();

}

}

public static async Task<string> CloseOrderApiRequest(int posID, string userToken)

{

return await ApiControl.POSTRequest($"OrderPositions/CloseOrderPosition?OrderPositionID={posID}&Token={userToken}", new Dictionary<string, string>());

}

public static async Task<string> OpenOrderApiRequest(int posID, string userToken)

{

return await ApiControl.POSTRequest($"OrderPositions/OpenOrderPosition?OrderPositionID={posID}&Token={userToken}", new Dictionary<string, string>());

}

public async void OpenOrder(object sender)

{

if (isChecked == true) { isChecked = false; return; }

MessageBoxResult boxResult = MessageBox.Show("Вы уверены, что хотите открыть товар?", "Уведомление!", MessageBoxButton.YesNo);

if (boxResult == MessageBoxResult.No) { isChecked = true; ((CheckBox)sender).IsChecked = true; return; }

StackPanel currentItem = (StackPanel)((CheckBox)sender).Parent;

int posID = Convert.ToInt32(((TextBlock)currentItem.Children[0]).Text);

if (posID == -1) { isChecked = true; ((CheckBox)sender).IsChecked = true; return; }

Листинг 17 - логика страницы «OrderDetail.xaml.cs»

else

{

string response;

try

{

response = await OpenOrderApiRequest(posID, App.Token);

} catch (Exception ex)

{

ApiError.ShowError(ApiErrors.ApiConnectionFailed);

return;

}

if (Convert.ToInt32(response) != 1)

{

isChecked = true; ((CheckBox)sender).IsChecked = true;

ApiError.ShowError(ApiError.GetErrorByCode(Convert.ToInt32(response)));

return;

}

else

{

((TextBlock)currentItem.Children[6]).Text = "";

MessageBox.Show("Позиция в заказе успешно открыта!", "Уведомление!", MessageBoxButton.OK, MessageBoxImage.Asterisk);

}

OrderPage.UpdateTreeView();

}

}

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

{

CloseOrder(sender);

}

private void CheckBox\_Unchecked(object sender, RoutedEventArgs e)

{

OpenOrder(sender);

}

}

}

Листинг 18 - логика страницы «OrderDetail.xaml.cs»

Логика страницы «DatabaseConnectionSettings.xaml.cs» представлена в листингах 19 – 20.

using Ratep.Class;

using System;

using System.Collections.Generic;

using System.Configuration;

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;

Листинг 19 - логика страницы «DatabaseConnectionSettings.xaml.cs»

namespace Ratep.Views.Pages

{

/// <summary>

/// Логика взаимодействия для DatabaseConnectionSettings.xaml

/// </summary>

public partial class DatabaseConnectionSettings : Page

{

public DatabaseConnectionSettings()

{

InitializeComponent();

}

private void ConnectButtonClick(object sender, RoutedEventArgs e)

{

ConnectChange(ServerName.Text, DBName.Text);

MessageBox.Show("Соединение успешно изменено!");

}

public static async Task<int> ConnectChange(string ServerName, string DBName)

{

string response = await ApiControl.POSTRequest($"DBSettings/ChangeConnectString?ServerName={ServerName}&DBName={DBName}");

return 1;

}

private void BackButtonClick(object sender, RoutedEventArgs e)

{

NavigationService.GoBack();

}

}

}

Листинг 20 - логика страницы «DatabaseConnectionSettings.xaml.cs»

Далее представлены листинги классов Web.API.

Логика класса «Program.cs» представлена в листингах 21 – 22.

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddJwtBearer(options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

ValidIssuer = builder.Configuration["Jwt:Issuer"],

ValidAudience = builder.Configuration["Jwt:Audience"],

IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Configuration

["Jwt:Key"]))

};

Листинг 21 - логика класса «Program.cs»

});

builder.Services.AddMvc();

builder.Services.AddControllers();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

Листинг 22 – логика класса «Program.cs»

Логика класса «DBSettings.cs» представлена в листинге 33.

#nullable disable

using Microsoft.AspNetCore.Mvc;

using RatepAPI.Models;

namespace RatepAPI.Controllers

{

[ApiController]

[Route("[controller]")]

public class DBSettings : ControllerBase

{

[HttpPost("ChangeConnectString")]

public ActionResult<string> ChangeConnectString(string ServerName, string DBName)

{

VeloRaContext.ConnectString = $"Server={ServerName};Database={DBName};Trusted\_Connection=True;";

UsersController.DBContext = new VeloRaContext();

return Ok();

}

}

}

Листинг 23 - логика класса «DBSettings.cs»

Логика класса «OrderPositionsController.cs» представлена в листингах 24 – 27.

#nullable disable

using Microsoft.AspNetCore.Mvc;

using Newtonsoft.Json;

using RatepAPI.Models;

namespace RatepAPI.Controllers

{

[ApiController]

Листинг 24 - логика класса «OrderPositionsController.cs»

[Route("[controller]")]

public class OrderPositionsController : ControllerBase

{

[HttpPost ("OpenOrderPosition")]

public ActionResult<string> OpenOrderPosition(int OrderPositionID, string Token)

{

User user;

Employee employee;

OrderPosition orderPosition;

Order order;

PartAssemblyUnit pau;

try

{

user = UsersController.DBContext.Users.FirstOrDefault(c => c.Token == Token);

employee = UsersController.DBContext.Employees.FirstOrDefault(c => c.AccountId == user.AccountId);

orderPosition = UsersController.DBContext.OrderPositions.FirstOrDefault(c => c.PosId == OrderPositionID);

order = UsersController.DBContext.Orders.FirstOrDefault(c => c.OrderPositions.Contains(orderPosition));

pau = UsersController.DBContext.PartAssemblyUnits.FirstOrDefault(c => c.OrderPositions.Where(c => c.PosId == OrderPositionID).Any());

}

catch (Exception ex)

{

return NotFound(505);

}

if (orderPosition == null || user == null)

return NotFound(505);

if (employee.ManufactoryId != pau.ManufactoryId)

{

return BadRequest(504);

}

try

{

orderPosition.CompletionDate = null;

order.CompletionDate = null;

UsersController.DBContext.SaveChanges();

} catch

{

return NotFound(404);

}

return "1";

}

[HttpPost("CloseOrderPosition")]

public ActionResult<string> CloseOrderPosition(int OrderPositionID, string Token)

{

User user;

Employee employee;

OrderPosition orderPosition;

Order order;

PartAssemblyUnit pau;

Try

Листинг 25 - логика класса «OrderPositionsController.cs»

{

user = UsersController.DBContext.Users.FirstOrDefault(c => c.Token == Token);

employee = UsersController.DBContext.Employees.FirstOrDefault(c => c.AccountId == user.AccountId);

orderPosition = UsersController.DBContext.OrderPositions.FirstOrDefault(c => c.PosId == OrderPositionID);

order = UsersController.DBContext.Orders.FirstOrDefault(c => c.OrderPositions.Contains(orderPosition));

pau = UsersController.DBContext.PartAssemblyUnits.FirstOrDefault(c => c.OrderPositions.Where(c => c.PosId == OrderPositionID).Any());

}

catch (Exception ex)

{

return NotFound(505);

}

if (orderPosition == null || user == null)

return NotFound(505);

if (employee.ManufactoryId != pau.ManufactoryId)

{

return BadRequest(504);

}

try

{

orderPosition.CompletionDate = DateTime.Now;

if (UsersController.DBContext.OrderPositions.Where(c => c.OrderId == order.OrderId).Select(c => c.CompletionDate == null).Any())

order.CompletionDate = DateTime.Now;

UsersController.DBContext.SaveChanges();

}

catch

{

return NotFound(404);

}

return Ok(1);

}

[HttpGet("GetClientsOrderList")]

public ActionResult<IEnumerable<Client>> GetClientsOrderList(string Token)

{

try

{

List<Client> clientsOrder = UsersController.DBContext.Clients.ToList();

foreach (Client client in clientsOrder)

{

client.Account = UsersController.DBContext.Users.FirstOrDefault(c => c.AccountId == client.AccountId);

client.Account.Clients = null;

client.Contracts = UsersController.DBContext.Contracts.Where(c => c.ClientId == client.ClientId).ToList();

foreach (Contract contract in client.Contracts)

{

contract.Orders = UsersController.DBContext.Orders.Where(c => c.ContractId == contract.ContractId).ToList();

contract.Client = null;

Листинг 26 - логика класса «OrderPositionsController.cs»

contract.Orders.First().Contract = null;

contract.Orders.First().OrderPositions = UsersController.DBContext.OrderPositions.Where(c => c.OrderId == contract.Orders.First().OrderId).ToList();

foreach (OrderPosition position in contract.Orders.First().OrderPositions)

{

position.Pau = UsersController.DBContext.PartAssemblyUnits.FirstOrDefault(c => c.Pauid == position.Pauid);

position.Pau.Manufactory = UsersController.DBContext.Manufactories.FirstOrDefault(c => c.ManufactoryId == position.Pau.ManufactoryId);

position.Pau.Manufactory.PartAssemblyUnits = null;

position.Pau.Manufactory.Employees = null;

position.Order = null;

position.Pau.OrderPositions = null;

}

}

}

UsersController.DBContext = new VeloRaContext();

return clientsOrder;

}

catch (Exception ex)

{

return NotFound(404);

}

}

}

}

Листинг 27 - логика класса «OrderPositionsController.cs»

Логика класса «UsersController.cs» представлена в листингах 28 – 29.

#nullable disable

using Microsoft.AspNetCore.Mvc;

using RatepAPI.Models;

using RatepAPI.Class;

using Microsoft.AspNetCore.Authorization;

using Microsoft.IdentityModel.Tokens;

using System.Text;

using System.Security.Claims;

using System.IdentityModel.Tokens.Jwt;

namespace RatepAPI.Controllers

{

[ApiController]

[Route("[controller]")]

public class UsersController : ControllerBase

{

public static VeloRaContext DBContext;

private IConfiguration \_config;

public UsersController(IConfiguration config)

{

\_config = config;

DBContext = new VeloRaContext();

}

Листинг 28 - логика класса «UsersController.cs»

[AllowAnonymous]

[HttpPost]

public IActionResult Login(string CryptedLogin, string CryptedPassword)

{

User user;

try

{

user = DBContext.Users.ToList().FirstOrDefault(c => Security.StringToSHA256(c.Login) == CryptedLogin

&& Security.StringToSHA256(c.Password) == CryptedPassword);

} catch (Exception ex)

{

return NotFound(404);

}

if (user != null)

{

string token = GenerateToken(user);

if (user.Token != token)

{

try

{

user.Token = token;

DBContext.SaveChanges();

}

catch (Exception ex)

{

return NotFound(404);

}

}

return Ok(token);

}

return NotFound(405);

}

private string GenerateToken(User user)

{

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(\_config["Jwt:Key"]));

var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);

var claims = new[]

{

new Claim(ClaimTypes.NameIdentifier, user.FullName.Split(' ')[0]),

new Claim(ClaimTypes.Surname, user.FullName.Split(' ')[1]),

new Claim(ClaimTypes.Role, user.Role)

};

var token = new JwtSecurityToken(\_config["Jwt:Issuer"],

\_config["Jwt:Audience"],

claims,

expires: DateTime.UtcNow.AddMinutes(15),

signingCredentials: credentials);

return new JwtSecurityTokenHandler().WriteToken(token);

}

}

}

Листинг 29 - логика класса «UsersController.cs»

2.5. Разработка тестовых данных

Разработка тестов представлена в таблицах 23 – 42.

Таблица 23 – Тест №1

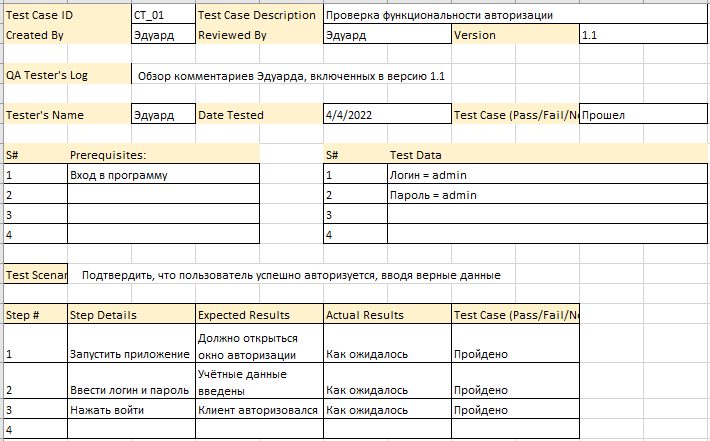


Таблица 24 – Тест №2



Таблица 25 – Тест №3



Таблица 26 – Тест №4



Таблица 27 – Тест №5



Таблица 28 – Тест №6



Таблица 29 – Тест №7



Таблица 30 – Тест №8



Таблица 31 – Тест №9



Таблица 32 – Тест №10



Таблица 33 – Тест №11



Таблица 34 – Тест №12



Таблица 35 – Тест №13



Таблица 36 – Тест №14



Таблица 37 – Тест №15



Таблица 38 – Тест №16



Таблица 39 – Тест №17



Таблица 40 – Тест №18



Таблица 41 – Тест №19



Таблица 42 – Тест №20



Результат тестов приведён на рисунке 11.

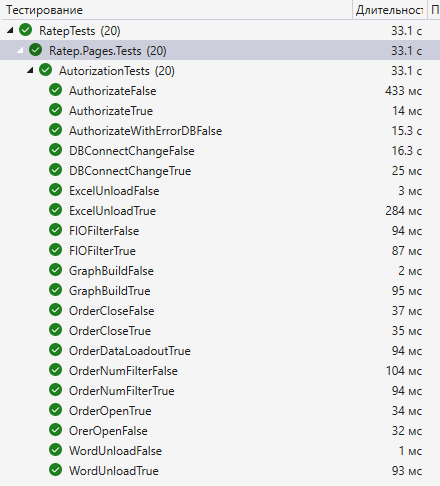


Рисунок 11

2.6. Результат работы программы

Результат работы программы представлен на рисунках 12 - 16.

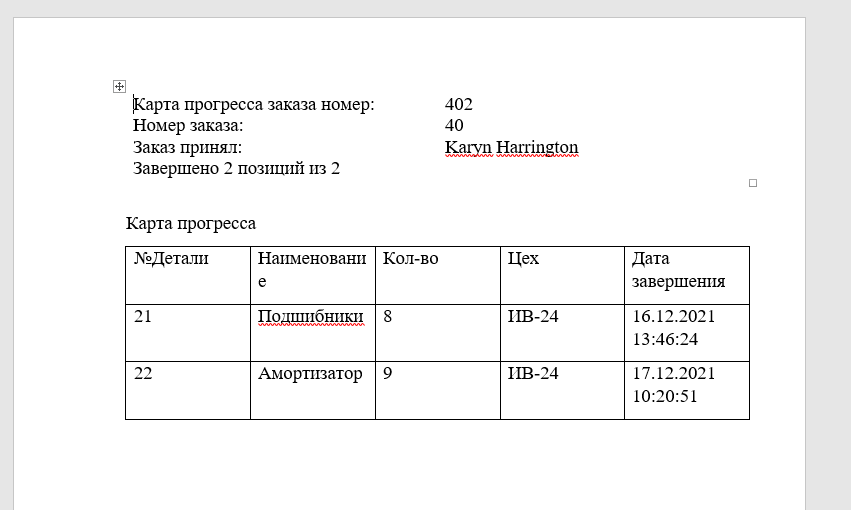


Рисунок 12 – Результат работы программы «Формирование отчётности»

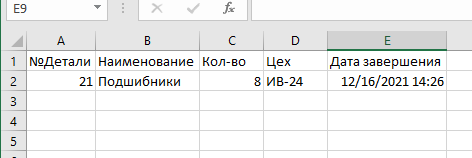


Рисунок 13 – Результат работы программы «Формирование отчётности»



Рисунок 14 – Результат работы программы «Статистика за период»

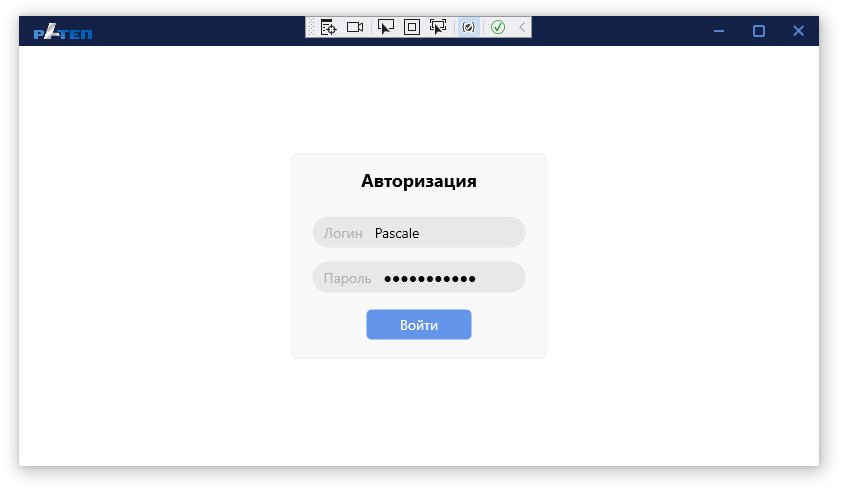


Рисунок 15 – Результат работы программы «Авторизация»

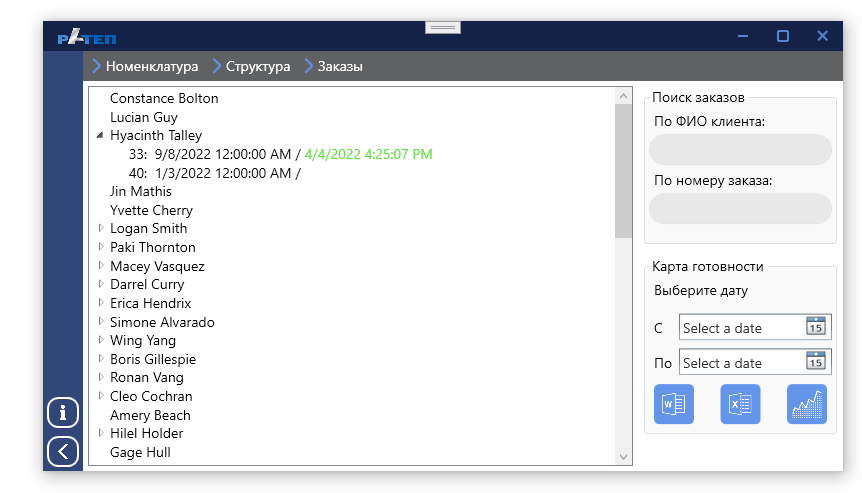


Рисунок 16 – Результат работы программы «Главное меню»

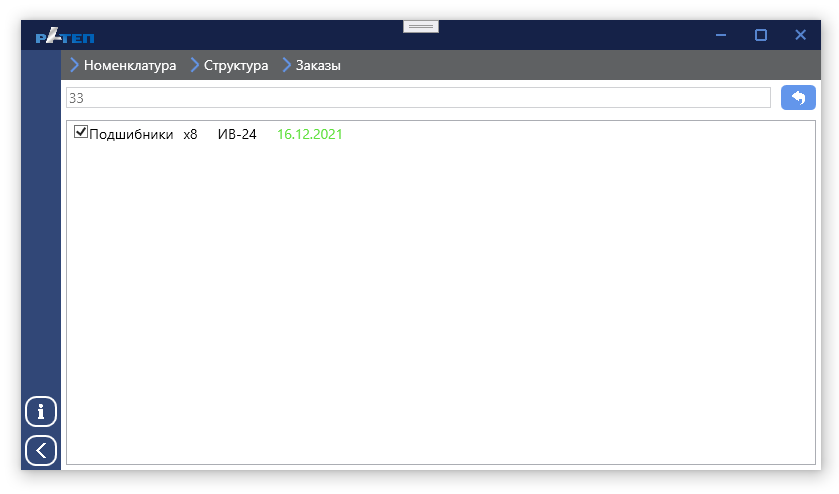


Рисунок 17 – Результат работы программы «Инфо заказа»

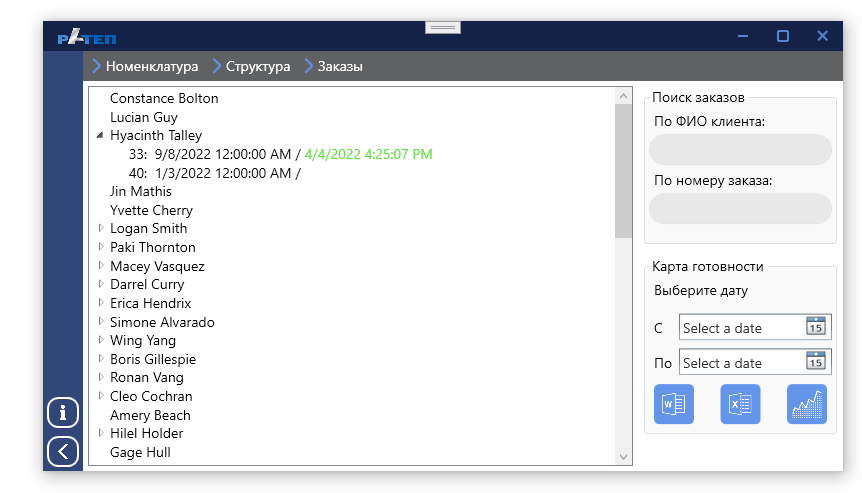


Рисунок 18 – Результат работы программы «Клиенты»

2.7. Руководство по использованию программы

2.7.1. Руководство системного администратора

Для работы с базами данных необходимо установить приложение установить программу для управления базами данных на сервере. Поэтому надо установить SQL Server Management Studio 19. Все необходимые материалы для установки можно найти по адресу <https://docs.microsoft.com/ru-ru/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15>.

Присоединение базы данных:

1. В SQL Server Management Studio обозревателе объектов Компонент SQL Server Database Engine подключитесь к экземпляру компонента и разверните его представление в SSMS.
2. Щелкните правой кнопкой мыши узел «**Базы данных»** и выберите команду «**Присоединить»**.

Чтобы указать присоединяемую базу данных, в диалоговом окне «**Присоединение баз данных»** нажмите кнопку «**Добавить»**, в диалоговом окне «**Расположение файлов базы данных»** выберите диск, на котором находится база данных, и разверните дерево каталогов, чтобы найти и выбрать MDF-файл.

Таблица 43 – Сообщение системному программисту

|  |  |
| --- | --- |
| Сообщение | Описание действий при получении сообщения |
| Ссылка на объект не указывает на экземпляр объекта. | Убедитесь, что объекты, участвовавшие в работе алгоритма, не равны null. |
| Не удалось найти строку подключения с именем "Ratep" в файле конфигурации приложения. | Убедитесь, что файл с конфигурацией приложения в целостности. |

2.7.2. Руководство программиста

Данная программа предназначена для автоматизации процесса регистрации заявок клиента на заселение в гостиницу.

Программа предусматривает авторизацию для одной роли «Ресепшионист»

Для корректной работы данной программы требуется следующее аппаратное и программное обеспечение:

* процессор AMD A8-5500 или совместимый с ним;
* 4 Гб оперативной памяти;
* видеоадаптер, поддерживаемый Windows;
* операционная система: Windows 7, Windows 8, Windows 10;
* Microsoft Word 2007 или выше.

Настройка программы.

Для начала установки программы вам необходимо установить Visual Studio 2019. Скачать установочные файлы можно на официальном сайте Microsoft: https://visualstudio.microsoft.com/ru/downloads/. Далее нужно запустить установочный файл Setup. Следуйте инструкции мастера установки. Выберите нужный путь расположения файлов на вашем компьютере, либо установите в уже заданную директорию по умолчанию.

Важно: не забудьте куда вы установили программу!

После успешной установки, в заданной директории при установке, вы найдете файлы приложения и саму программу. Файлы приложения можно оставить там же или перенести их в стандартную директорию файлов Visual Studio 2019.

Внимание: обычно все файлы Visual Studio 2019 находятся в директории: C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\Common7\IDE\

Запускаем VisualStudio 2019. Для начала необходимо найти программу «Hotel». Для этого при открытии Visual Studio 2019 написать в поиске «Hotel» или найти при нажатии «Открыть проект или решение» программу, которую вы расположили.

Таблица 44 – Сообщения программисту

|  |  |
| --- | --- |
| Сообщение | Описание действий при получении сообщения |
| Попытка обращения к элементу массива или коллекции с индексом, который находится вне границ | Убедитесь, что указанный индекс существует в коллекции. |
| Ошибках операций арифметического приведения или преобразования | Убедитесь, что приведение объекта к данному типу возможно. |

2.7.3. Руководство оператора

Для начала вам нужно запустить программу.

После запуска программы вам откроется окно авторизации. Здесь необходимо ввести логин и пароль после нажать на кнопку “Войти”. Интерфейс данной формы представлен на рисунке 19.

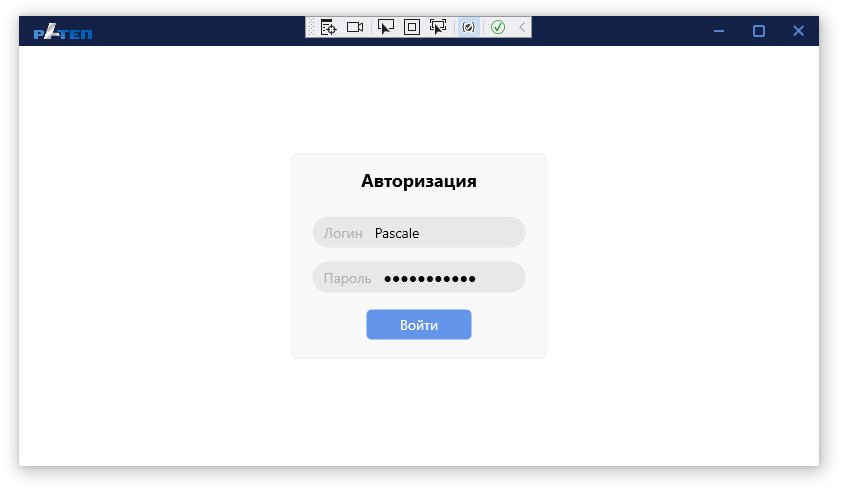


Рис. 19 – Авторизация

Далее вы попадете в главное меню, с которым вы и будете работать.

Кнопка «Заказы» позволяет открыть интерфейс, в которым мы можем просмотреть заказы на рисунке 20. Кнопка “Выгрузить” позволяет получить данные по готовности заказов за указанный период.

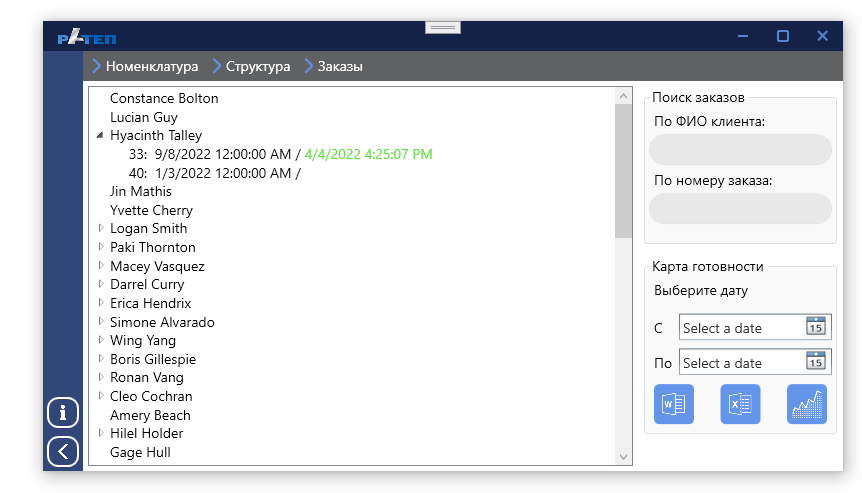


Рис. 20 – Главное меню

Нажав два раза на левую кнопку мыши появится полная информация о заказе где можно проставить готовность позиций заказов. Интерфейс данный формы представлен на рисунке 21.

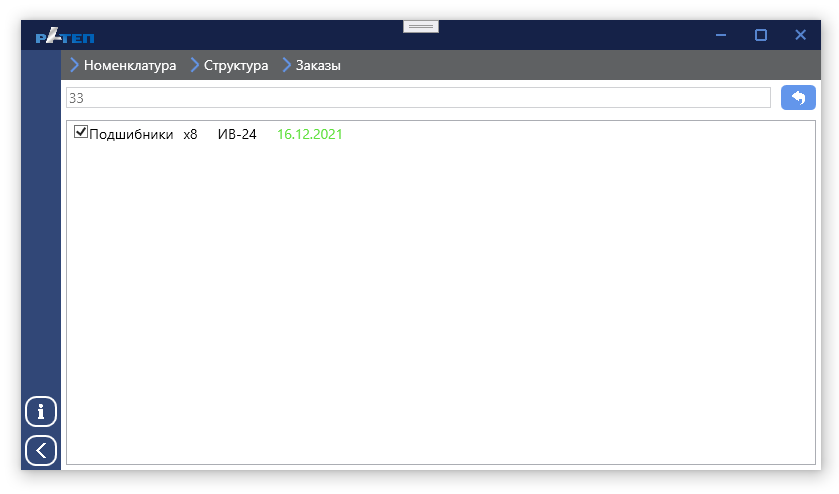


Рис. 21 – Инфо заказ

После проставления галочек на всех позициях заказов и нажав кнопку «Назад» можно увидеть дату завершения заказа. Подробнее на рисунке 22.

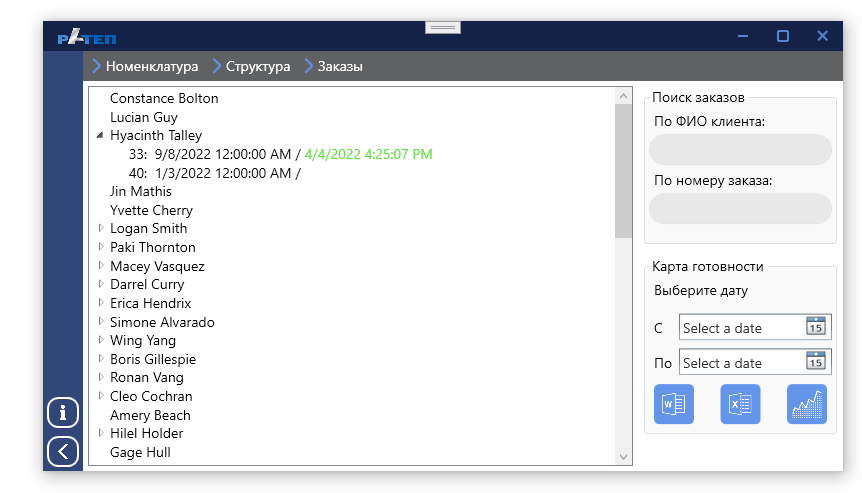


Рис. 22 – Клиенты

Заполнив данными поля «Выбор даты» выберите нужный заказ и нажмите кнопку «Выгрузить в Word». В результате вы увидите уведомление об успешном создании файла. Файл отчётности за указанную дату представлен на рисунке 23. По нажатию на кнопку «Выгрузить в Excel» вы увидите отчётность, представленную на рисунке 24. По нажатию на кнопку «Построить график статистики» вы увидите статистике, представленную на рисунке 25.

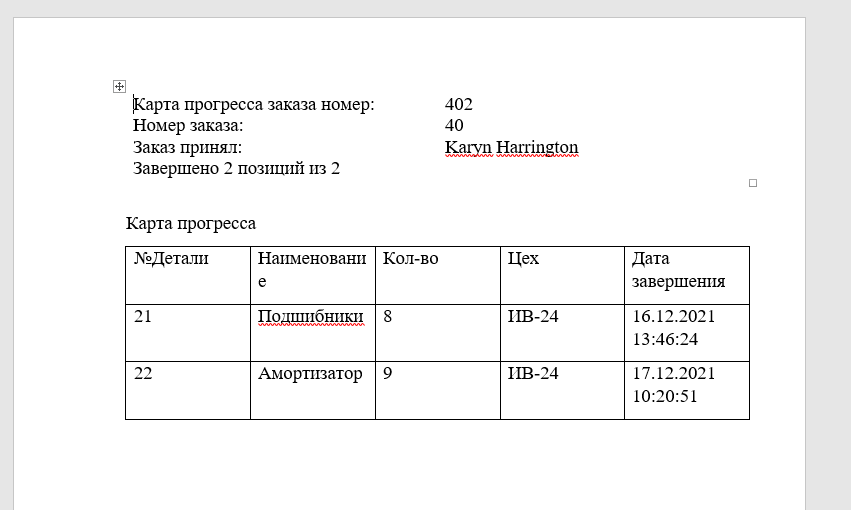


Рис. 23– Отчётный файл

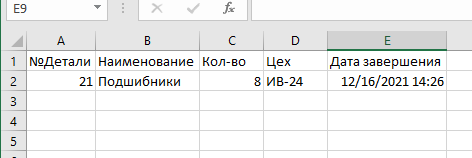


Рис. 24– Отчётный файл



Рис. 25– Отчётная форма

Таблица 45 – Сообщения оператору

|  |  |
| --- | --- |
| Сообщение | Описание действий при получении сообщения |
| Не удалось подключиться к базе данных | Убедитесь, что запущена служба SQL Server. Убедитесь, что SQL Server настроен правильно |
| Данные введены неверно | Убедитесь, что заполнены все поля верными данными |
| Не удалось удалить запись | Убедитесь, что выбранный клиент не имеет привязанных договоров. |
| Не удалось создать файл Word | Убедитесь, что недавно созданный файл Word закрыт, а также в целостности данные Word. |

2.8. Разработка политики безопасности

Основной целью, на достижение которой направлены все положения настоящей Политики, является защита информационных ресурсов от возможного нанесения им материального, физического, морального или иного ущерба, посредством случайного или преднамеренного воздействия на информацию, её носители, процессы обработки и передачи, а также минимизация рисков ИБ.

В данной программе было при отправке запроса Web.API данные пользователя подвергаются шифровке с помощью метода шифрования, представленного на листинге 30.

public static string EncryptStringToSHA256(string value)

{

StringBuilder Sb = new StringBuilder();

using (SHA256 hash = SHA256Managed.Create())

{

Encoding enc = Encoding.UTF8;

Byte[] result = hash.ComputeHash(enc.GetBytes(value));

foreach (Byte b in result)

Sb.Append(b.ToString("x2"));

}

return Sb.ToString();

}

Листинг 30 – метод шифрования данных

Также для обеспечения последующего безопасного взаимодействия между клиентом и сервером генерируется JWT-токен с помощью метода, представленного на листингах 31 - 32.

private string GenerateToken(User user)

{

var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(\_config["Jwt:Key"]));

var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);

var claims = new[]

{

new Claim(ClaimTypes.NameIdentifier, user.FullName.Split(' ')[0]),

Листинг 31 – метод генерации JWT-токена

new Claim(ClaimTypes.Surname, user.FullName.Split(' ')[1]),

new Claim(ClaimTypes.Role, user.Role)

};

var token = new JwtSecurityToken(\_config["Jwt:Issuer"],

\_config["Jwt:Audience"],

claims,

expires: DateTime.UtcNow.AddMinutes(15),

signingCredentials: credentials);

return new JwtSecurityTokenHandler().WriteToken(token);

}

Листинг 32 – метод генерации JWT-токена

2.9. Эксплуатация и развитие программы

Данное приложение разрабатывалось для предприятия АО «РАТЕП» для автоматизации процессов производства.

Ранее ведение состояний закзазов осуществлялось конструкторским бюро на бумажных носителях вручную, без поддержки ПО. Для подтверждения изготовления позиции заказов инженеру было необходимо явиться в конструкторское бюро и заполнить бланк. С помощью этого приложения инженер сможет отмечать готовность дистанционно, что упростит процесс и работу данных сотрудников.

Таким образом можно сделать вывод, что данное ПО будет иметь дальнейшее сопровождение и в случае успеха сможет стать программой массового использования малых торговых организаций.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данной практической работы было разработано приложение «Ratep».

Первым этапом было рассмотрение и анализ программ «Класс365», «РемОнлайн», «Бизнес.ру», сравнение функциональных особенностей приложений, их архитектуры и технологии, при помощи которых написаны данные приложения.

Для организации хранения данных приложения были рассмотрены основные системы управления базами данных и основываясь на используемых технологиях для написания приложения и интеграции с уже имеющимися сервисами было принято использовать Microsoft SQL Server.

В качестве среды разработки были рассмотрены Microsoft Visual Studio 2019 и Visual Studio Code. Выбор был отдан Microsoft Visual Studio 2019, т.к. это решение является готовой IDE, направленной на разработку приложений на платформе WPF.

Для организации доступа к данным была сгенерирована модель данных с использованием технологии ADO.NET Entity Framework, которое при помощи технологии LINQ, а также Entity SQL организует доступ данным из БД и позволяет легко сопровождать написанный код.

Клиентское приложение написано с использованием языка программирования C#. Приложение используется одним пользователем, согласно роли – инженер, которому доступен следующий функционал:

* работа с состоянием готовности позиций заказов;
* просмотр клиентов и их заказов;
* работа с состоянием готовности заказов;
* формирование отчётности по состоянию выполнения заказа;
* формирование статистики выполнения заказов.

Благодаря использованию различных инструментальных средств и пользовательских элементов управления, был создан привлекательный, удобный и интуитивно понятный пользовательский интерфейс.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Professor Web – Пользовательские элементы WPF [Электронный ресурс] (Дата обращения 10.04.2022) – <https://professorweb.ru/my/WPF/Template/level18/18_1.php>;
2. Фленов М. – Transact-SQL [Электронный ресурс] (Дата обращения 10.04.2022) – <https://www.flenov.info/books/read/transact-sql>;
3. Ускоряемся в EF Core [Электронный ресурс] (Дата обращения 10.04.2022): <https://habr.com/ru/post/487734/;>
4. ГОСТ 19.001-77 ЕСПД. Общие положения;
5. Entity Framework Tutorial [Электронный ресурс] (Дата обращения 10.04.2022): <https://www.entityframeworktutorial.net/>;
6. Подход Database first [Электронный ресурс] (Дата обращения 10.04.2022) – <https://metanit.com/sharp/entityframework/2.4.php>;
7. Язык программирования C# и платформа .NET [Электронный ресурс] (Дата обращения 10.04.2022) – <https://metanit.com/sharp/>;
8. Техническая документация Microsoft [Электронный ресурс] (Дата обращения 10.04.2022) – <https://docs.microsoft.com/ru-ru/>;
9. Документация по C#. Начало работы, руководства, справочные материалы [Электронный ресурс]. Дата обращения: 10.04.2022. URL: <https://docs.microsoft.com/ru-ru/dotnet/csharp/>;
10. Руководство Microsoft по Microsoft.Office.Interop.Excel [Электронный ресурс] (Дата обращения: 10.04.2022) <URL:https://docs.microsoft.com/dotnet/api/microsoft.office.interop.excel>
11. Казакова И.А. Основы языка Transact SQL: учеб. пособие / И.А. Казакова. – Пенза: Изд-во ПГУ, 2010. – 164 с. (Дата обращения 10.04.2022)
12. QnA-система StackOverFlow [Электронный ресурс] (Дата обращения: 10.04.2022) URL: <https://ru.stackoverflow.com/>
13. Официальная документация Microsoft по языку C# [Электронный ресурс] (Дата обращения: 10.04.2022) URL: <https://docs.microsoft.com/>
14. Официальная документация Microsoft по языку T-SQL [Электронный ресурс] (Дата обращения 10.04.2022) URL: https://docs.microsoft.com/ru-ru/sql/t-sql/language-reference
15. Команды Git [Электронный ресурс] (Дата обращения 10.04.2022) URL: https://gist.github.com/stanruss/498e5f5549ecc4a44d540f7214b92063

ПРИЛОЖЕНИЕ 1 – Бизнес-процесс IDEF0

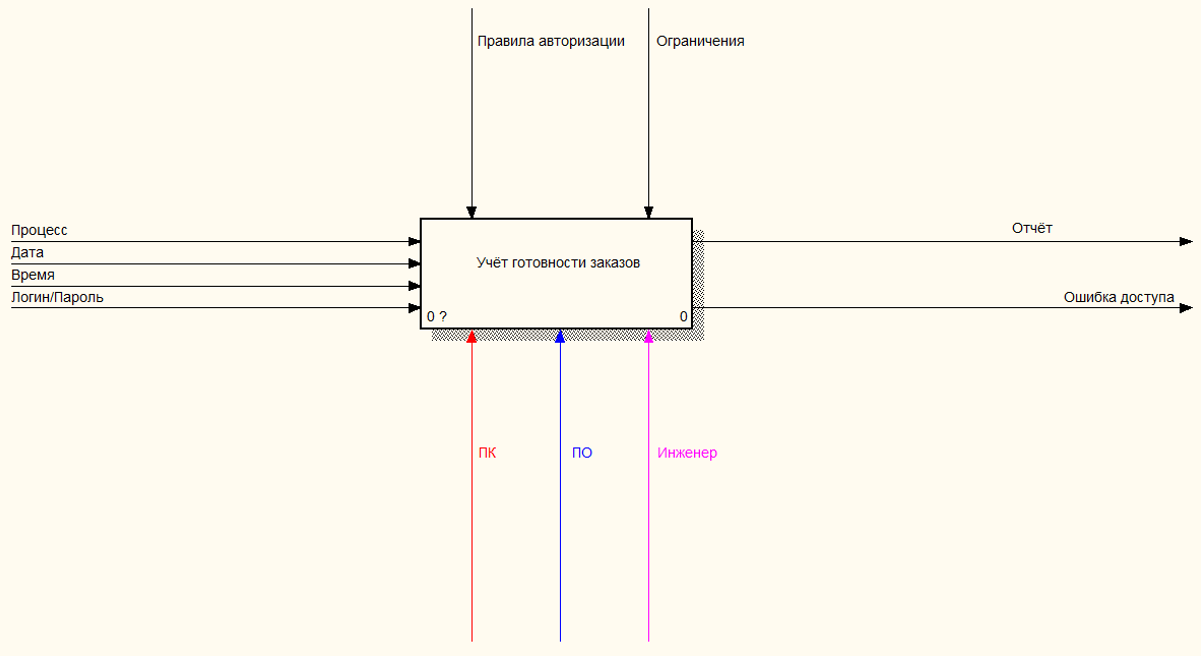


Рисунок 26 - бизнес-процесс IDEF0

ОКОНЧАНИЕ ПРИЛОЖЕНИЯ 2

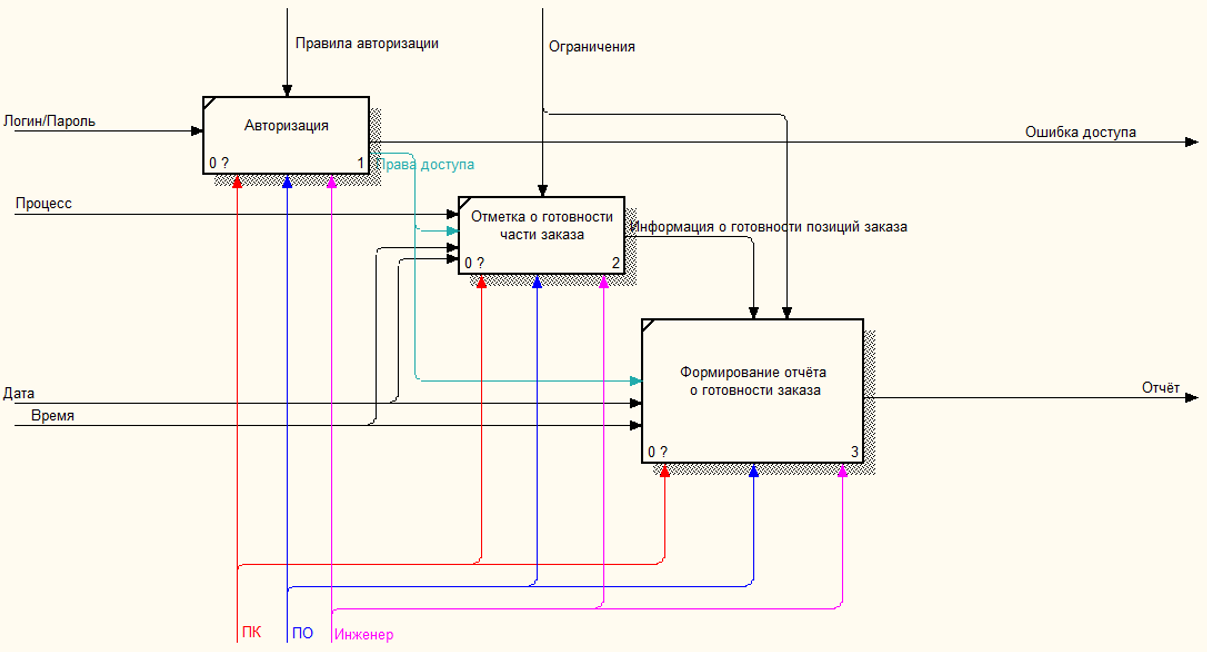


Рисунок 27 - бизнес-процесс IDEF0

ПРИЛОЖЕНИЕ 2 – Потоки данных DFD

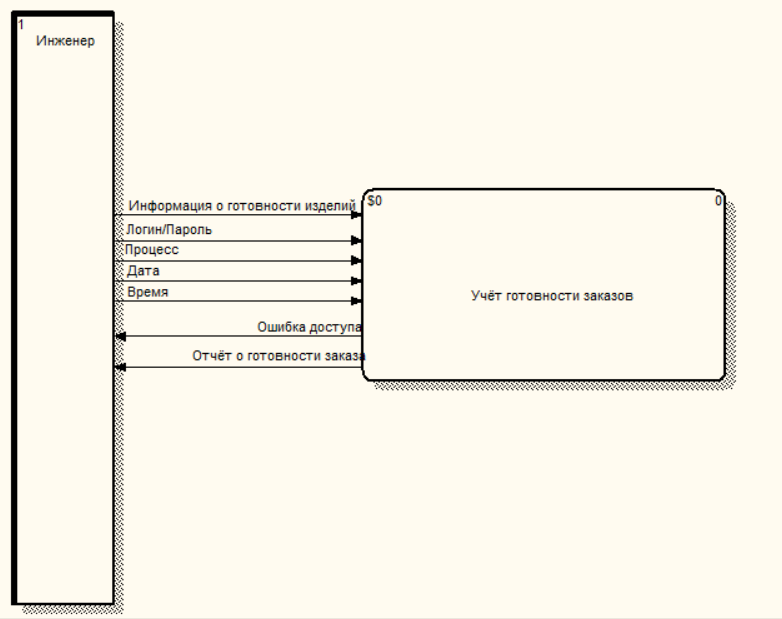


Рисунок 28 - потоки данных DFD

ОКОНЧАНИЕ ПРИЛОЖЕНИЯ 3

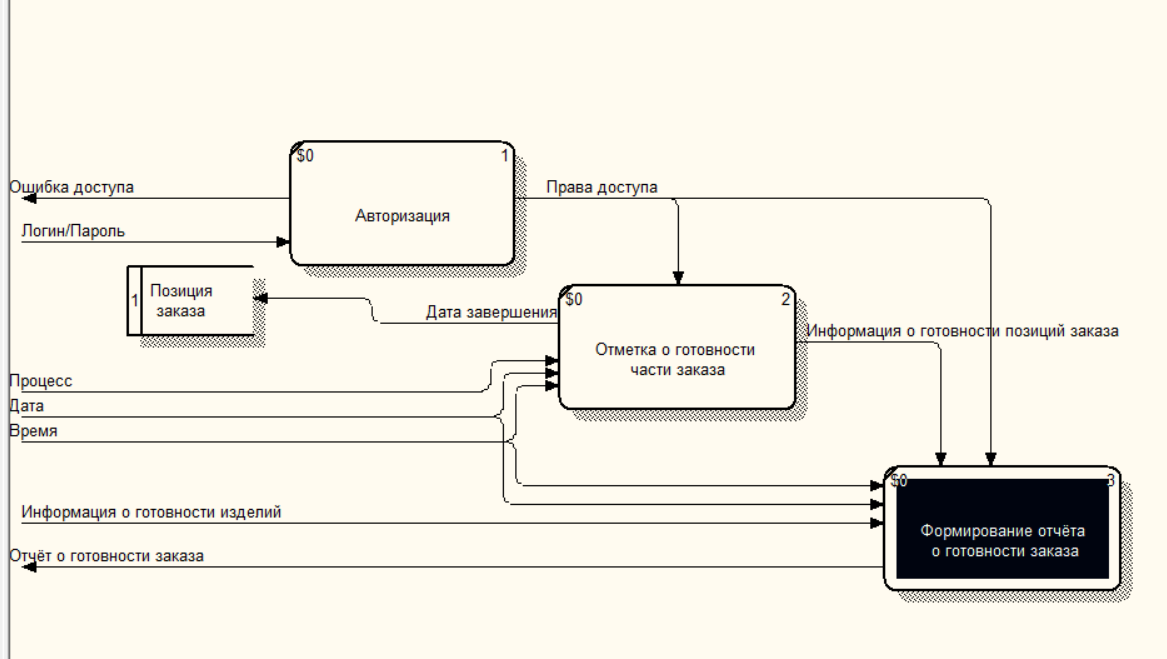


Рисунок 29 - потоки данных DFD

ПРИЛОЖЕНИЕ 3 – Диаграмма вариантов использования

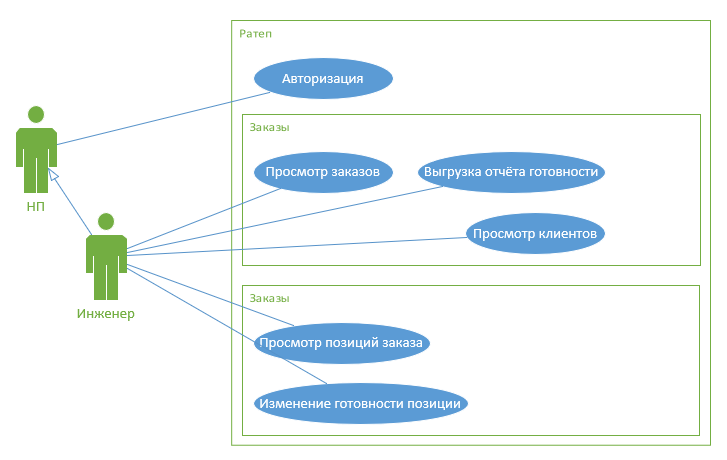


Рисунок 30 - диаграмма вариантов использования

ПРИЛОЖЕНИЕ 4 – Диаграмма последовательности

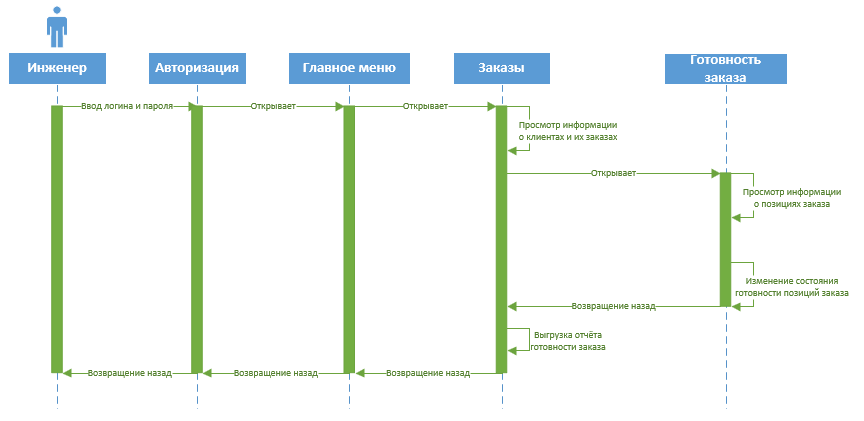


Рисунок 31 - диаграмма последовательности

ПРИЛОЖЕНИЕ 5 – Диаграмма классов

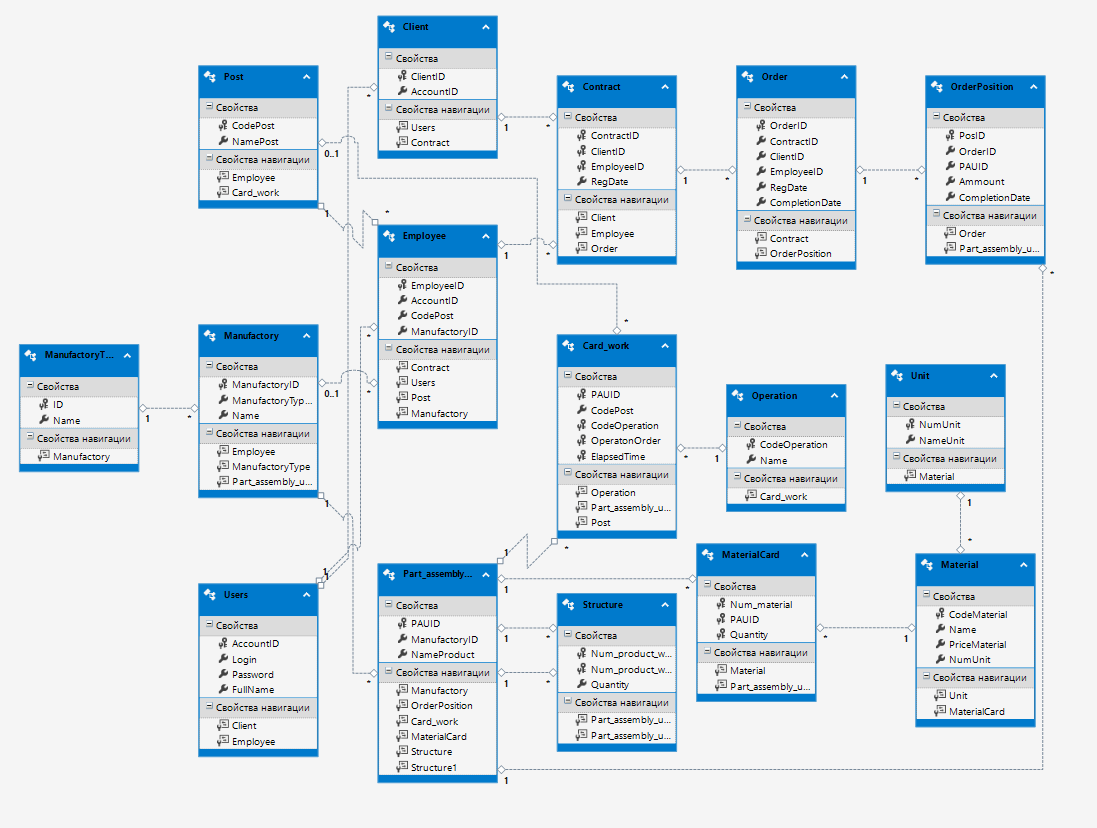


Рисунок 32 - диаграмма классов

ПРИЛОЖЕНИЕ 6 – Диаграмма деятельности

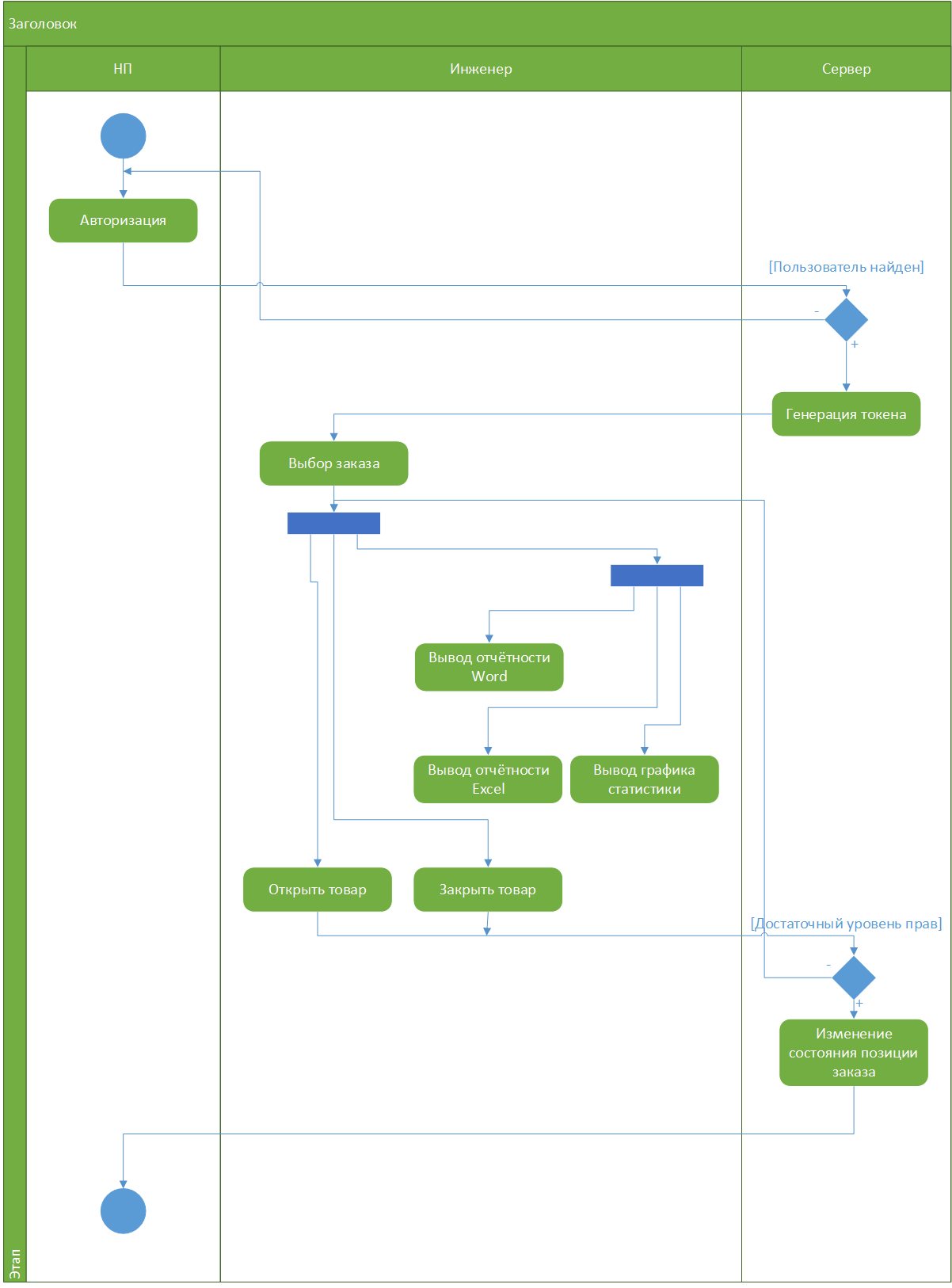


Рисунок 33 - диаграмма деятельности

ПРИЛОЖЕНИЕ 7 – Диаграмма развёртывания

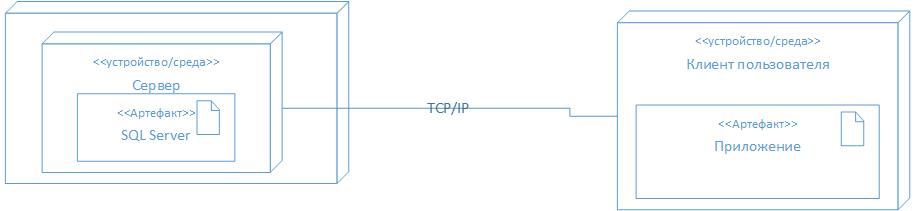


Рисунок 34 - диаграмма развёртывания

ПРИЛОЖЕНИЕ 8 – Логическая модель

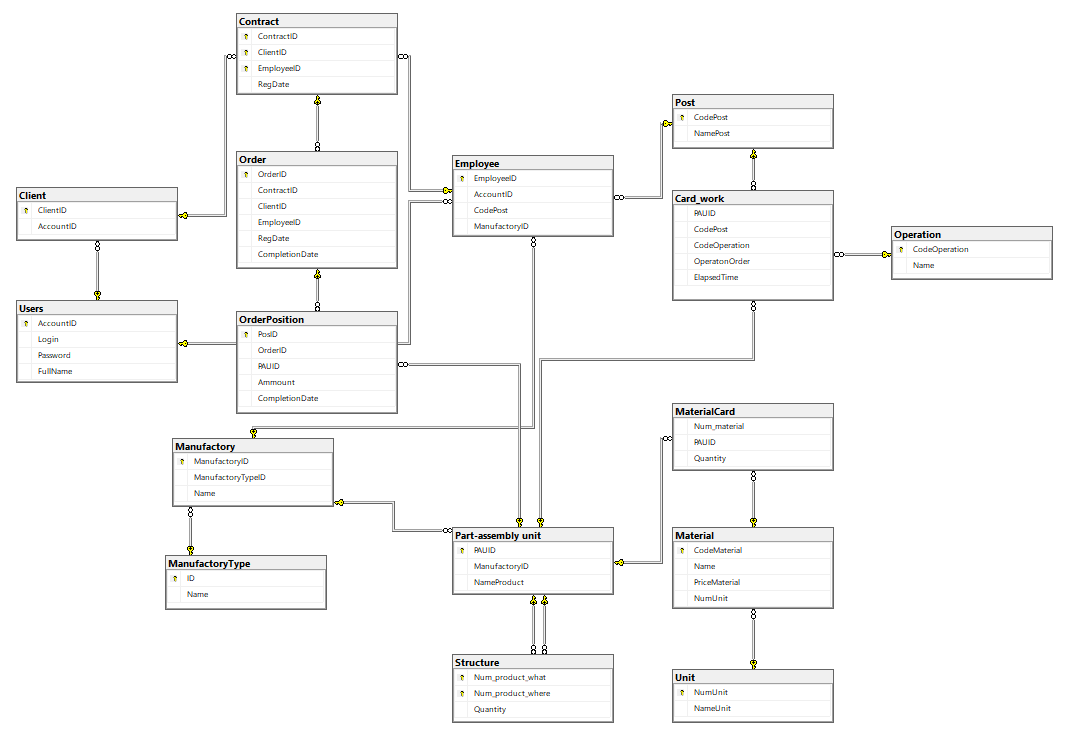


Рисунок 35 - логическая модель

ПРИЛОЖЕНИЕ 9 – Физическая модель

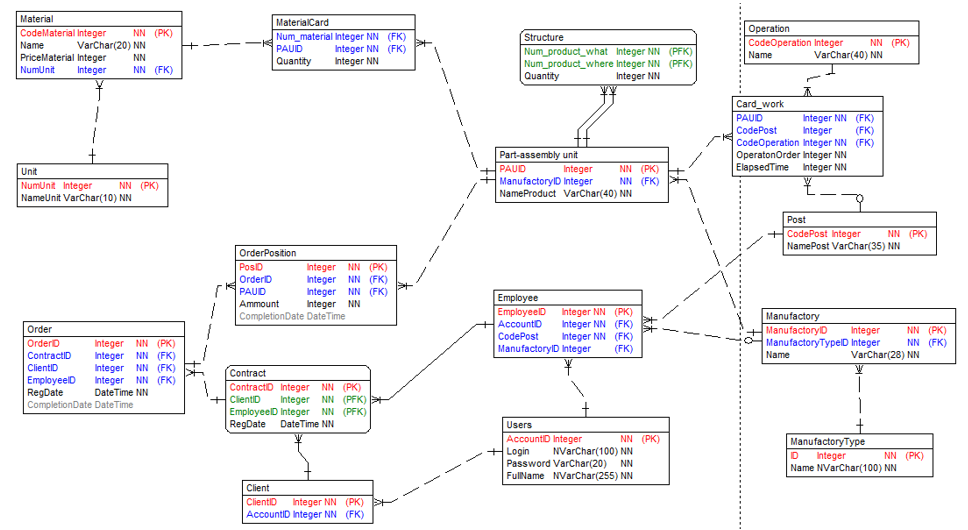


Рисунок 36 - физическая модель