МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1-40 05 01 Информационные системы и технологии

Направление специальности 1-40 05 01 Информационные системы

и технологии

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«База данных службы по ремонту вычислительной техники с использованием технологии резервного копирования и восстановления»

Выполнил студент Цветков Н.С.

(Ф.И.О.)

Руководитель проекта доцент Е.А Блинова

(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: доцент Е.А Блинова.

(учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: доцент Е.А Блинова.

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2021

**Содержание**

[Введение 3](#_Toc71576021)

[1. Аналитический обзор аналогов 4](#_Toc71576022)

[1.1 Программное средство «система автоматизации сервисного центра» 4](#_Toc71576023)

[1.2 Программное средство «Сервисный Центр» 5](#_Toc71576024)

[2. Анализ требований к программному средству и разработка функциональных требований 6](#_Toc71576025)

[3. Разработка архитектуры проекта 7](#_Toc71576026)

[3.1 Обобщенная структура управления приложением 7](#_Toc71576027)

[3.2 Диаграммы UML, взаимосвязь всех компонентов 8](#_Toc71576028)

[4. Разработка модели базы данных 9](#_Toc71576029)

[5. Использование Oracle 12с 12](#_Toc71576030)

[5.1 Создание таблиц 12](#_Toc71576031)

[5.2 Создание ролей для разграничения доступа 12](#_Toc71576032)

[5.3 Создание пакетов и процедур для работы с базой данных 12](#_Toc71576033)

[5.4 Описание процедур экспорта и импорта 14](#_Toc71576034)

[5.5 Тестирование производительности базы данных 15](#_Toc71576035)

[7. Тестирование, проверка работоспособности и анализ полученных результатов 18](#_Toc71576036)

[7.1 Тестирование клиентской области 18](#_Toc71576037)

[7.2 Тестирование области работника 19](#_Toc71576038)

[8.1 Область клиента 21](#_Toc71576039)

[8.2 Область сотрудника 22](#_Toc71576040)

[Заключение 24](#_Toc71576041)

[Cписок источников 25](#_Toc71576042)

[Приложение А 26](#_Toc71576043)

[Приложение Б 27](#_Toc71576044)

[Приложение В 30](#_Toc71576045)

[Приложение Г 35](#_Toc71576046)

[Приложение Г 39](#_Toc71576047)

# **Введение**

Данный курсовой проект посвящён разработке базы данных для службы по ремонту вычислительной техники.

Цель разработки базы данных – спроектировать максимально удобную для встраивания в программное средство базу данных, внутри которой будет осуществлена вся работа с таблицами с помощью пакетов и хранимых процедур.

Данная база данных должна находится на удалённом сервере и быстро разворачиваться на любом другом сервере, посредством последовательного выполнения скриптов.

В базе данных должны присутствовать разграничения для различных пользователей, таких как администратор, клиент и работник. Это было сделано для того, чтобы пользователь не смог повредить базу данных.

В качестве интерфейса был выбран API Windows Forms (WinForms).

Для работы с данным интерфейсом использовался объектно-ориентированный язык программирования C#.

При разработке программного средства были использованы:

* Orcacle 12c;
* Visual Studio 2020;
* Oracle.ManagedDataAccess.Core 3.21.1.

Для нормального функционирования приложения:

* ОС: Windows 7 и выше;
* процессор: Dual Core 3.0 Ghz;
* оперативная память: 4Gb;
* место на диске: 200MB;

видеокарта: 256mb Video Memory.

# **1. Аналитический обзор аналогов**

В наши дни множество аналогов программных средств можно найти на различных веб-ресурсах, связанных с данной тематикой. Перед тем как приступить к разработке требовалось провести анализ существующих аналогов.

1.1 Программное средство «система автоматизации сервисного центра»

На рисунке 1.1 предствлено главное окно приложения «системы автоматизации сервисного центра»

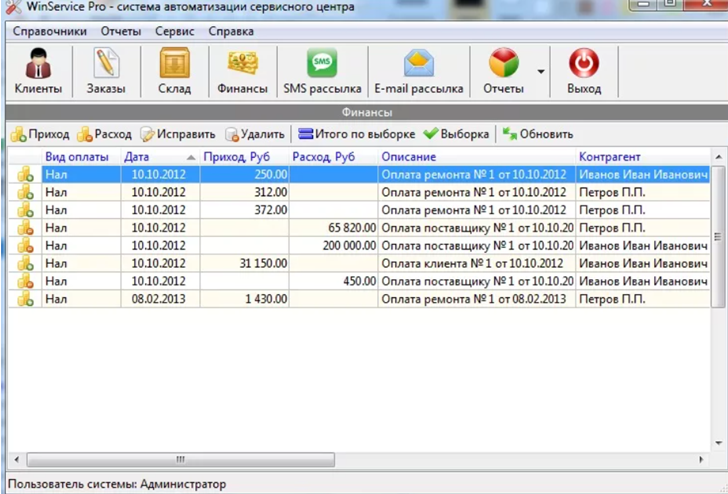


Рисунок 1.1 – Окно аналога приложения «системы автоматизации сервисного центра»

Ознакомившись с программных средством, был сделан вывод об основных функциональных особенностях:

* использование списка для представления множества данных;
* возможность просматривать клиентов, заказы, склад;
* возможность осуществлять sms и email рассылку;
* просмотр отчётов;
* просмотр финансов;
* возможность обновлять данные в списке;
* просмотр итого по выборке;
* возможность удалять и редактировать записи.

1.2 Программное средство «Сервисный Центр»

На рисунке 1.2 представлено главное окно приложения «системы автоматизации сервисного центра».

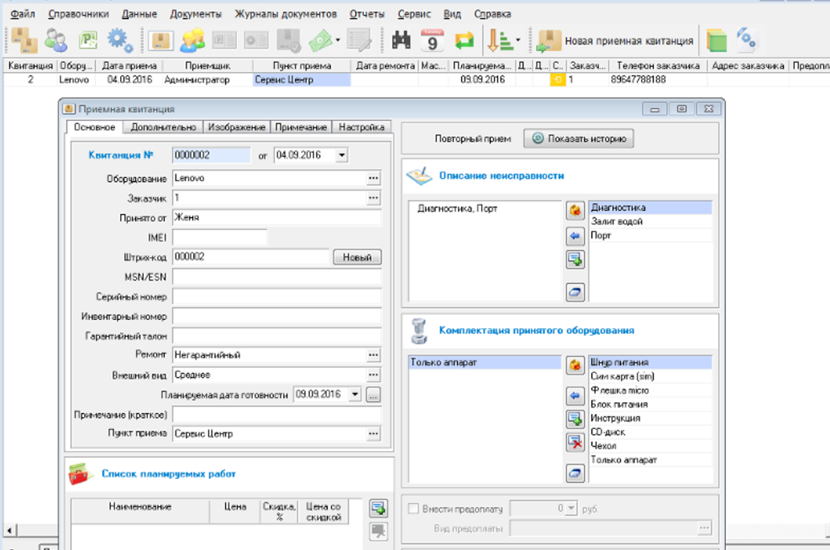


Рисунок 1.2 – Окно аналога приложения «ironbook.ru»

Ознакомившись с программных средством, был сделан вывод об основных функциональных особенностях:

наличие формы приёмной квитанции с множеством различных полей для заполнения;

* возможность прикреплять фотографию;
* возможность просмотра истории;
* возможность просмотра планируемых работ;
* возможность формирования квитанции.

# **2. Анализ требований к программному средству и разработка функциональных требований**

Проанализировав соответствующие аналоги по теме, были выдвинуты следующие требования:

разрабатываемая база данных должна иметь авторизацию для работников и клиентов;

* формирование заказа клиентом;
* просмотр заказов клиентов сотрудником;
* формирование заказа комплектующих сотрудником;
* изменение статуса заказа.

Форма реализации продукта продукта – десктоп приложение.

Десктоп приложение должно быть реализовано с использованием языка С#. Технология для реализации – Windows Forms. База данных приложения должно взаимодействовать с Oracle Database 12с.

В программном средстве требуется организовать возможность регистрироваться новым клиентам, авторизоваться. Предоставить возможность оформлять заказ, выбирать исполняющего мастера. Для работника предоставить возможность менять статус заказа путём взятия его в обработку. Также необходимо предоставить возможность заказа деталей, если это необходимо.

# **Разработка архитектуры проекта**

## **Обобщенная структура управления приложением**

В приложении имеется разделение пользователей на клиентов и работников. Роль администратора в приложении не предусмотрена, она возлагается на администратора базы данных. Программное средство разделено на два приложения. Одно предусмотрено для клиентов, а второе для работников.

Диаграмма вариантов использования приложения для клиента представлена на рисунке 3.1.

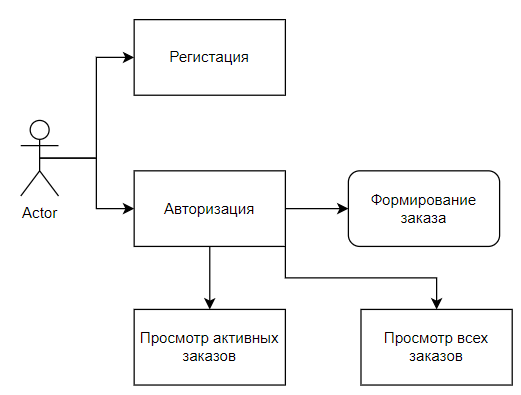


Рисунок 3.1 – Диаграмма использования пс для клиента

Диаграмма вариантов использования приложения для работника представлена на рисунке 3.2.

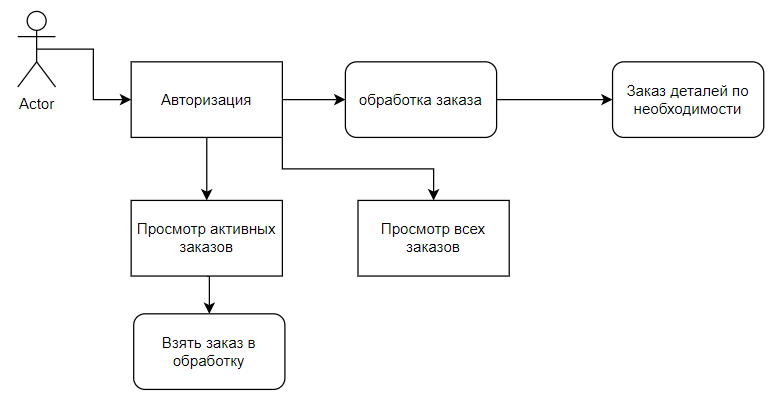


Рисунок 3.2 – Диаграмма использования пс для работника

Клиентское приложение – сразу после открытия предоставляет возможность авторизоваться либо же зарегистрироваться. В клиентском приложении предоставляется возможность работы с:

* формой формирования заказа;
* списком активных заказов;
* списком всех обработанных заказов.

Приложение работника – сразу после открытия предоставляет возможность авторизоваться. В данном приложении предоставляется возможность работы с:

* списком активных заказов;
* добавлять заказ из списка активных заказов в список обработки;
* списком всех выполненных заказов.

Область работы администратора – Sql Developer или SqlPlus.

## **Диаграммы UML, взаимосвязь всех компонентов**

Диаграмма UML – это графическое представление выбора элементов, изображаемое чаще всего в виде графа с вершинами (сущностями) и рёбрами (отношениями). Диаграммы рисуют для визуализации. Основная цель диаграмм – визуализация разрабатываемой системы с разных точек зрения.

На рисунке 3.2 представлена диаграмма клиентского приложения.

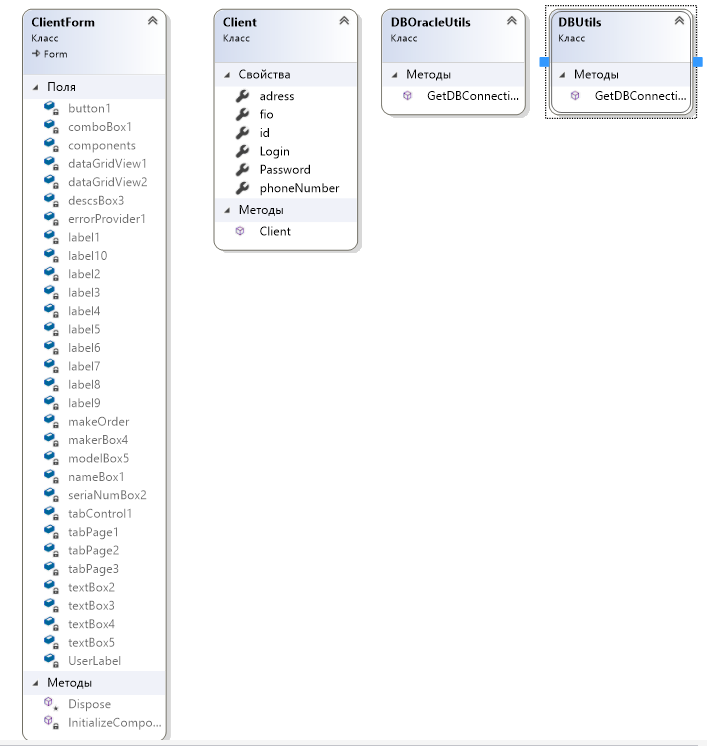


Рисунок 3.2 – Классы для работы клиентского приложения

Все основные методы для работы с БД находятся в ClientForm.cs. Класс Client определяет структуру полей клиента в БД. Классы DBOracleUtils и DBUtils необходимы для подключения к бд и работы с хранимыми процедурами.

На рисунке 3.3 представлена диаграмма классов приложения для работника.

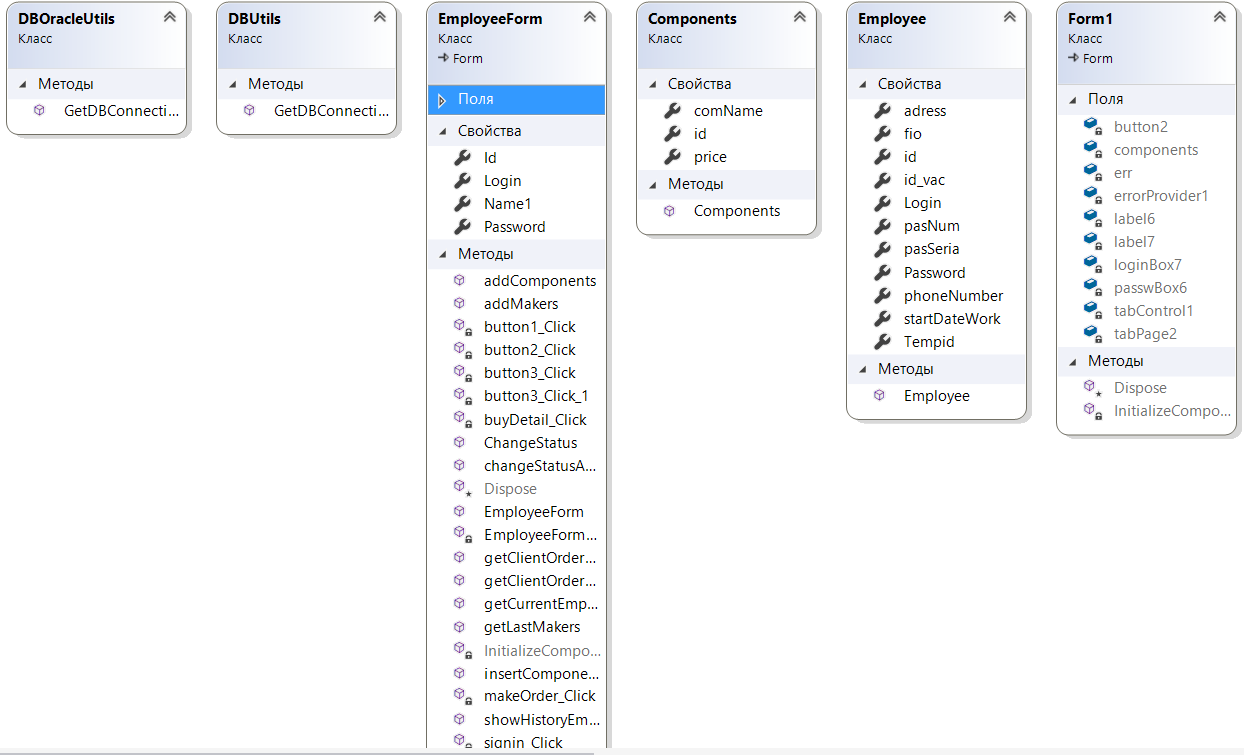


Рисунок 3.3 – Классы для работы приложения для работника

Структура классов приложения для работника и клиента схожи. Основные различия заключаются, что вызов хранимых процедур происходит через разные пакеты. То есть для клиента и работника были созданы отдельные пакеты и пользователи БД.

# **Разработка модели базы данных**

Первым этапом выполнения курсовой работы будет создание логически взаимосвязанных таблиц. Чтобы составить визуальную взаимосвязанную структуры нашей базы данных, нам необходимо продумать, какая информация будет храниться в этих таблицах, после этого создать связи с помощью первичных и внешних ключей.

База данных состоит из 9 таблиц, перечисленных ниже (рисунок 4.1).

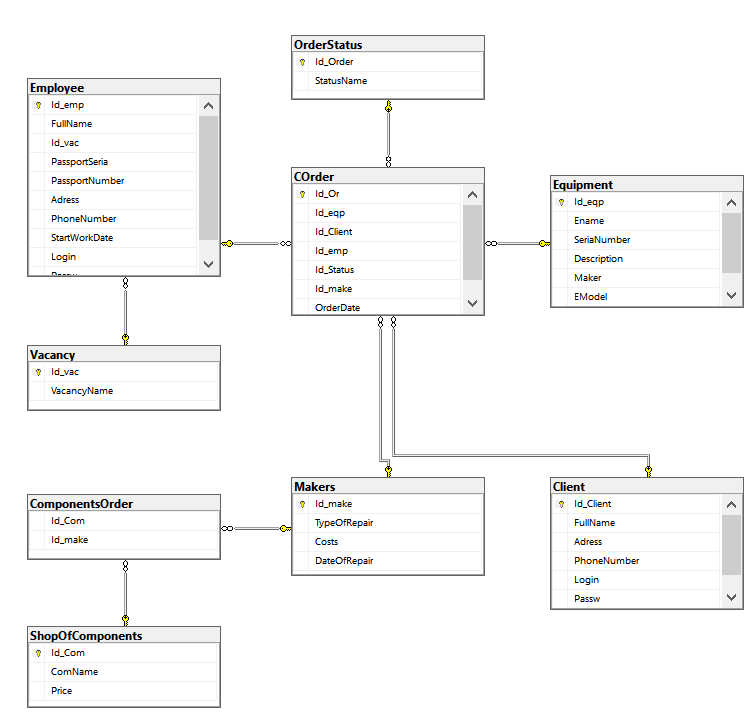


Рисунок 4.1 – Взаимосвязь таблиц данных

Далее приведена их структура и описание.

Таблица 4.1 – Структура таблицы «Client»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_Client | NUMBER | Id клиента |
| FullName | nvarchar2 | ФИО |
| Adress | nvarchar2 | Адрес |
| PhoneNumber | nvarchar2 | Номер телефона |
| Login | nvarchar2 | Логин |
| Passw | nvarchar2 | Пароль |

Таблица «Employee» содержит идентификатор, ФИО, идентификатор вакансии, номер паспорта, серия паспорта, адрес, номер телефона, дата приёма на работу, логин, пароль. Описание ее полей приведено в таблице 4.2.

Таблица 4.2 – Структура таблицы «Employee»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_emp | NUMBER | Id сотрудника |
| FullName | nvarchar2 | Название фирмы-производителя |
| Id\_vac | nvarchar2 | Id вакансии |
| PassportSeria | nvarchar2 | Серия паспорта |
| PassportNumber | nvarchar2 | Номер паспорта |
| Adress | nvarchar2 | Адрес |
| PhoneNumber | nvarchar2 | Номер телефона |
| StartWorkDate | Date | Дата начала работы |
| Login | nvarchar2 | Логин |
| Passw | nvarchar2 | Пароль |

Таблица «OrderStatus» содержит информацию о статусе заказа. Структура представлена в таблице 4.3.

Таблица 4.3 – Структура таблицы «OrderStatus»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_Order | NUMBER | Идентификатор материнской платы |
| StatusName | nvarchar2 | Имя статуса |

Таблица «Equipment» содержит описание оборудования. Структура представлена в таблице 4.4.

Таблица 4.4 – Структура таблицы «Equipment»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_eqp | NUMBER | Id оборудования |
| Ename | nvarchar2 | Наименование оборудования |
| SeriaNumber | nvarchar2 | Серия оборудования |
| Description | nvarchar2 | Описание проблемы |
| Maker | nvarchar2 | Исполнитель |
| EModel | nvarchar2 | Модель |

Таблица «Makers» содержит информацию об исполнителе и типе ремонта. Структура представлена в таблице 4.5.

Таблица 4.5 – Структура таблицы «Makers»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_make | NUMBER | Id исполнения |
| TypeOfRepair | nvarchar2 | Тип ремонта |
| Costs | nvarchar2 | Стоимость |
| DateOfRepair | Date | Дата исполнения |

Таблица «ComponentsOrder» содержит информацию об идентификаторах компонентов и испольнителе. Структура представлена в таблице 4.6.

Таблица 4.6 – Структура таблицы «ComponentsOrder»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_Com | number | Id компонента |
| Id\_make | number | Id исполнителя |

Таблица «ShopOfComponents» содержит информацию о компонентах. Структура представлена в таблице 4.7.

Таблица 4.7 – Структура таблицы «ShopOfComponents»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_Com | NUMBER | Id компонента |
| ComName | nvarchar2 | Имя компонента |
| Price | NUMBER | Цена |

Таблица «Vacancy» содержит информацию о вакансиях. Структура представлена в таблице 4.8.

Таблица 4.8 – Структура таблицы «Vacancy»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_vac | number | Id вакансии |
| VacancyName | nvarchar2 | Имя вакансии |

Таблица «COrder» содержит основную информацию о заказах клиентов. Структура представлена в таблице 4.9.

Таблица 4.9 – Структура таблицы «COrder»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_Or | number | Id заказа |
| Id\_eqp | number | Id оборудования |
| Id\_Client | number | Id клиента |
| Status | number | Статуса |
| Id\_emp | number | Id работника |
| Id\_Status | number | Id статуса |
| Id\_make | number | Id исполнения |

# **Использование Oracle** **12с**

## **Создание таблиц**

Для работы в базе данных мы должны создать таблицы и организовать между ними связи, которые будут представлять логическую структуру БД. В нашу базу будут включены такие таблицы:

* Client;
* Vacancy;
* Employee;
* OrderStatus;
* Equipment;
* Makers;
* ComponentsOrder;
* ShopOfComponents;
* COrder.

Создание этих таблиц проходит исполнение команд блока PL/SQL.

Все таблицы создаются в отдельном файле Create\_Tables.sql, для дальнейшей возможности запускать скрипт в целом файле в новой базе данных.

## **Создание ролей для разграничения доступа**

Для правильной работы десктоп приложения с базой данных необходимо разграничение ролей. Для выполнения данной задачи в СУБД Oracle предусмотрены встроенные функции, такие как пользователи, роли и привилегии.

В данном случае были созданы два пользователя и им было выдано право на выполнение процедур из пакета (рисунок 5.2).

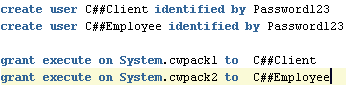


Рисунок 5.2 – Создание пользователей и выдача им прав

## **Создание пакетов и процедур для работы с базой данных**

После добавления ролей и создания всех необходимых таблиц приступаем к созданию процедур, позволяющих работать с данными.

Пакет Oracle PL/SQL – это объект схемы, который группирует логически связанные типы, элементы и подпрограммы. Пакеты обычно состоят из двух частей: спецификации и тела, хотя иногда тело не нужно.

Хранимая процедура (stored procedure) – это программа, которая выполняет некоторые действия с информацией в базе данных и при этом сама хранится в базе данных.

Для логического разделения процедур были созданы три пакета:

* cwpack1 – пакет процедур для клиента;
* cwpack2 – пакет процедур для работника;
* cwPackadm – пакет процедур для администратора БД.

Заголовок пакета, содержащий процедуры для работы клиента представлен на рисунке 5.3.1.

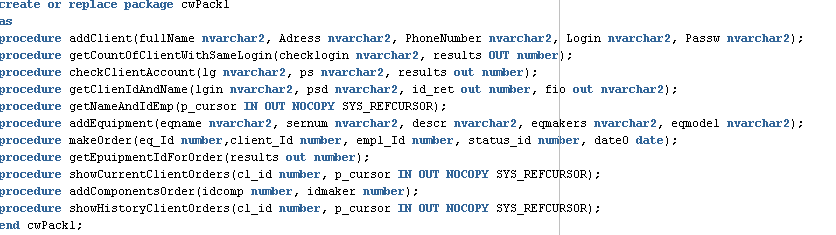


Рисунок 5.3.1 – Заголовок пакета cwpack1

Заголовок пакета, содержащий процедуры для работы сотрудника представлен на рисунке 5.3.2.

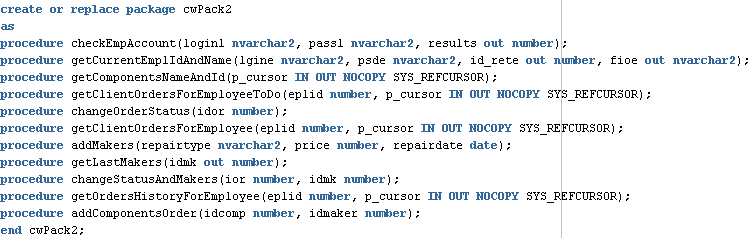


Рисунок 5.3.2 – Заголовок пакета cwpack2

Заголовок пакета, содержащий процедуры для работы администратора БД представлен на рисунке 5.3.3.

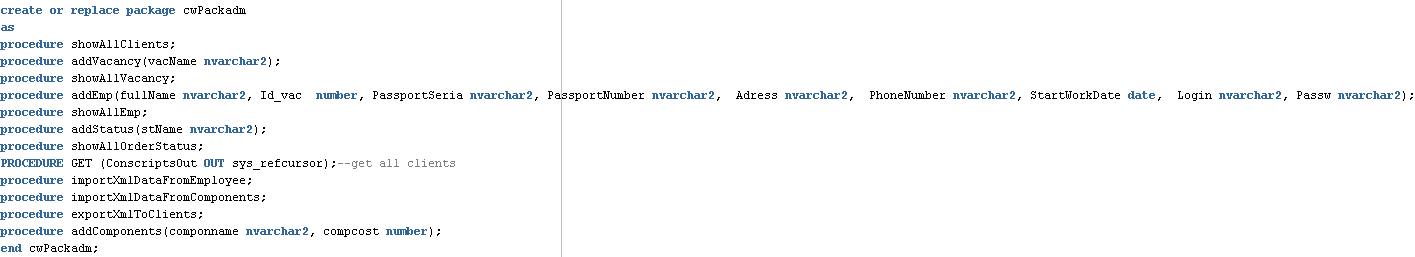


Рисунок 5.3.3 – Заголовок пакета cwpack2

## **Описание процедур экспорта и импорта**

Для комфортного администрирования базы данных в ней необходимо реализовать возможность экспорта и импорта данных в удобный для администратора формат. В нашем случае это формат xml;

Для доступа к папке из СУБД Oracle нужно должен быть создан Directory.

Объект Directory является логической ссылкой в базе данных на каталог файловой системы сервера, где установлена БД Oracle. Владельцем всех объектов Directory в базе данных является пользователь SYS, даже если объект Directory создан другим пользователем. Имена объектов Directory уникальны внутри всей БД. (рисунок 5.4.1).



Рисунок 5.4.1 – Создание объекта Directory

На рисунке 5.4.2 представлена процедура импорта данных из БД в файл IMPORTEMP.xml.

из БД

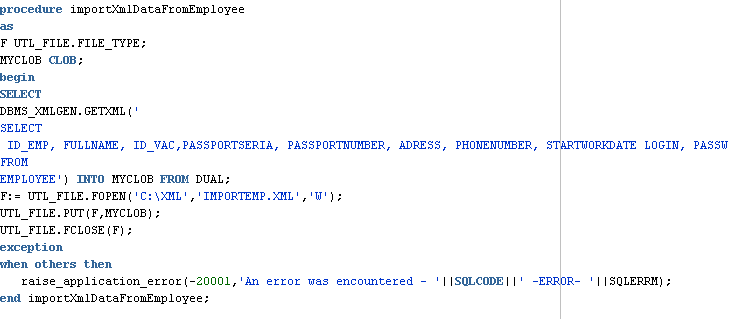


Рисунок 5.4.2 – Процедура импорта данных

На рисунке 5.4.3 представлена процедура экспорта данных из файла CLIENTS1.Xml в БД.

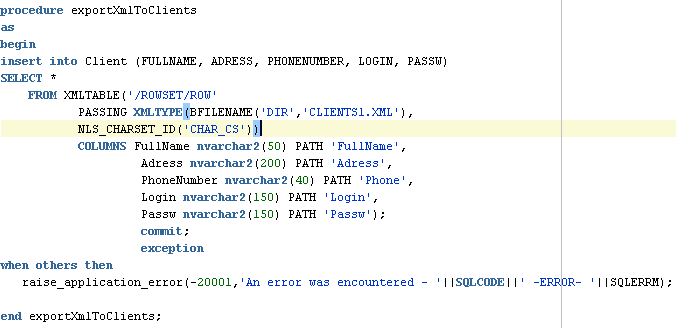


Рисунок 5.4.3 – Процедура экспорта данных

## **Тестирование производительности базы данных**

Для проверки производительности базы данных необходимо заполнить ее большим количеством различных данных и узнать время выполнения одного запроса. Для этого с помощью функции экспорта было добавлено 100000 строк (рисунок 5.5.1).

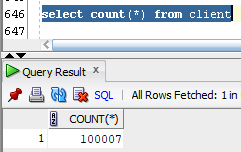


Рисунок 5.5.1 – Количество записей в таблице

Построим план запроса по всем элементам таблицы и посмотрим время его выполнения (рисунок 5.5.2).

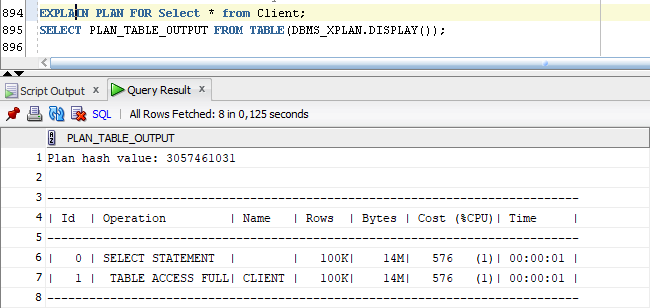


Рисунок 5.5.2 – План запроса

Как можно увидеть из рисунка запрос на 100000 строк выполнился очень быстро.

1. **Описание** **технологии резервного копирования и восстановления**

Традиционный пользовательский метод резервного копирования состоит в применении команд операционной системы для копирования необходимых файлов в другое место и/или на ленточное устройство. В случае применения утилиты RMAN резервное копирование файлов базы данных Oracle выполняться внутри базы данных посредством самого сервера баз данных. RMAN умеет делать резервные копии и копии образов файлов данных, управляющих файлов, архивных журналов повторного выполнения, файлов SPFILE и фрагментов резервных копий RMAN. Поэтому компания Oracle рекомендует применять для резервного копирования баз данных именно интерфейс RMAN.

Утилита RMAN упрощает процедуры резервного копирования, позволяя использовать мощные и в то же время легкие в написании сценарии резервного копирования и восстановления. Вдобавок RMAN обладает функциям наподобие обнаружения повреждений внутри блоков данных и способностью выполнять резервное копирование только изменившихся блоков в базе данных. Сценарии RMAN можно сохранять в базе данных и использовать прямо оттуда, что избавляет от необходимости писать сценарии уровня операционной системы. RMAN автоматически обеспечивает создание резервных копий всех файлов базы данных, что исключает вероятность допущения человеческой ошибки, которая присутствует в методах резервного копирования, основанных на применении команд операционной системы.

Для подключения к Rman используется строка $rman target system/system\_password (Рисунок 6.1.1).

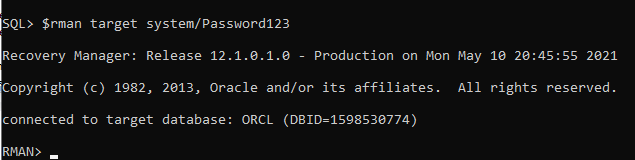


Рисунок 6.1.1 – Подключение к Rman

Для копирования используется строка BACKUP FULL DATABASE TAG "FULL\_DATABASE\_BACKUP1" PLUS ARCHIVELOG TAG "FULL\_ARCHIVELOGS\_BACKUP" (рисунок 6.1.2);

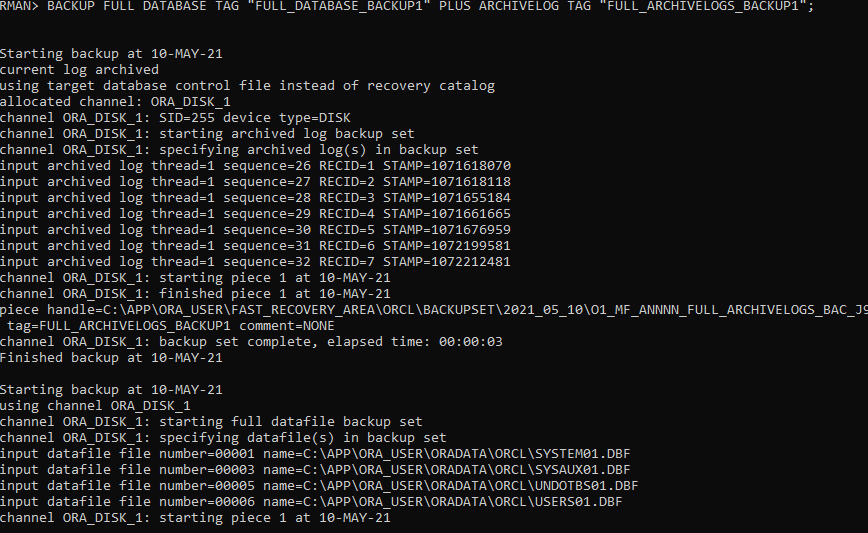


Рисунок 6.1.2 – Резервное копирование

Для полного восстановления используется команда RESTORE DATABASE PREVIEW SUMMARY;

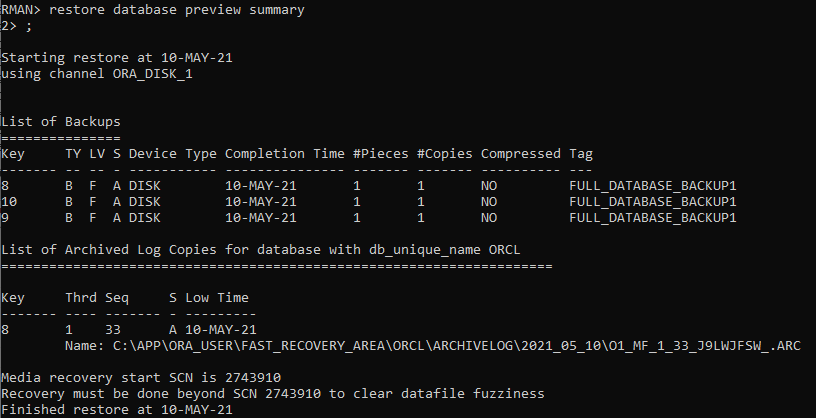


Рисунок 6.1.3 – Восстановление

# **Тестирование, проверка работоспособности и анализ полученных результатов**

## **Тестирование клиентской области**

Приложение проектировалось и разрабатывалось в направлении минимизации человеческих ошибок. В приложении присутствует минимальное количество мест, где клиент может вызвать ошибку программы. Все варианты использования приложения продуманы и не допускают возможности сделать некорректное действие.

При регистрации пользователя нельзя оставлять поля незаполненными. Если попробовать зарегистрироваться, не заполнив поля, появятся предупреждения, то есть у пользователя не получится зарегистрироваться пока он не заполнит все поля (рисунок 7.1.1).

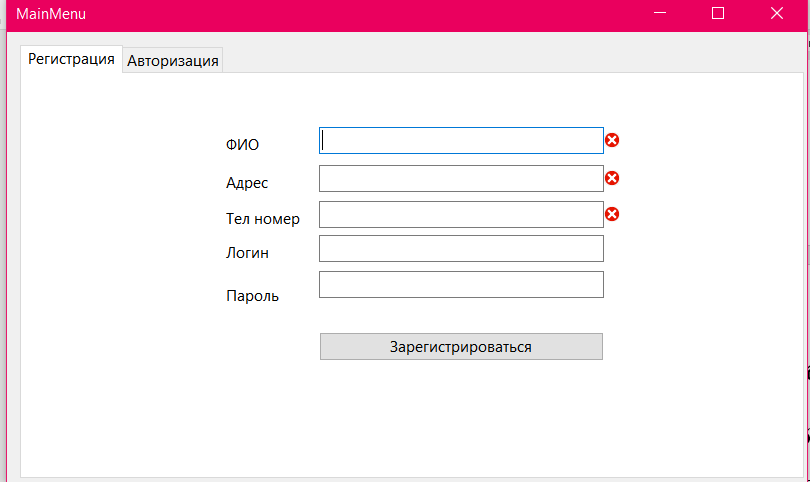


Рисунок 7.1.1 – Меню регистрации

При авторизации также происходит проверка полей на пустоту. При отсутсвии зарегистрированного аккаунта у пользователя не получится авторизоваться (рисунок 7.1.2).

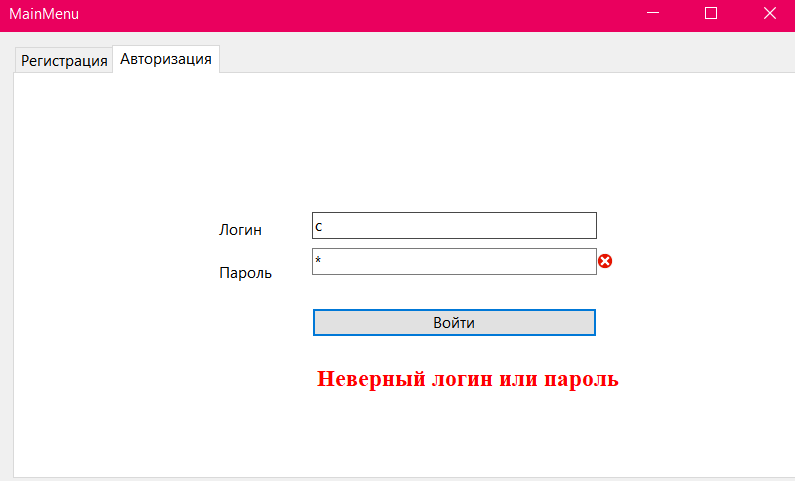


Рисунок 7.1.2 – Меню авторизации

После авторизации клиент может оформить заказ. При этом у него это получится в том случае, если он заполнит все поля (рисунок 7.1.3).

В случае попытки осуществить заказ с незаполненными полями у него появятся предупреждения.

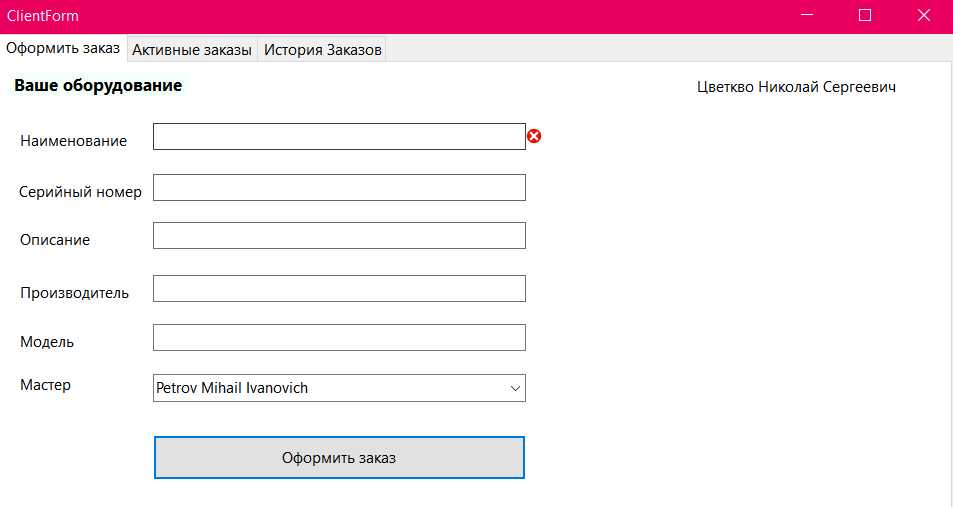


Рисунок 7.1.3 – Меню оформления заказов

В клиентском приложении это были все места, где он мог бы допустить ошибку.

## **Тестирование области работника**

У работника значительно больше мест, где можно было бы допустить ошибки.

При авторизации также происходит проверка полей на пустоту, а также проверка на наличие аккаунта в БД (рисунок 7.2.1).

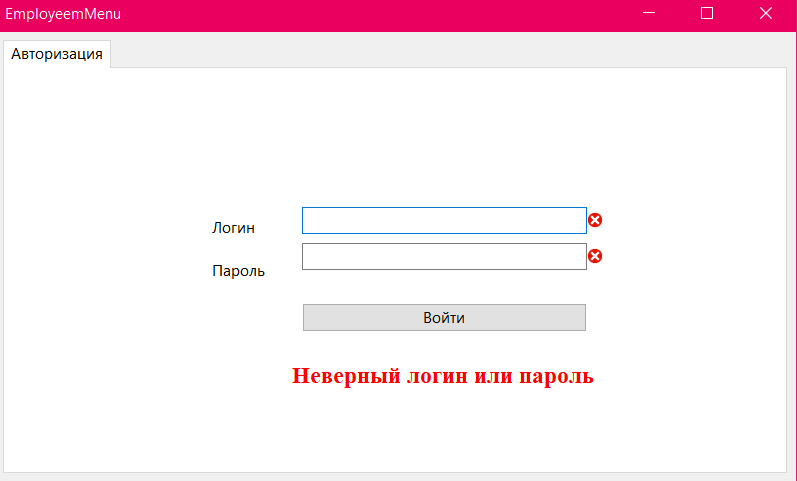


Рисунок 7.2.1 – Меню авторизации работника

После авторизации сотрудник попадает в окно с выбором заказа для обработки. Для этого ему требуется нажать на требуемый заказ.

Если заказов не будет, и работник нажмёт на кнопку обработки, то он получит предупреждение, что заказов нет (рисунок 7.2.2).

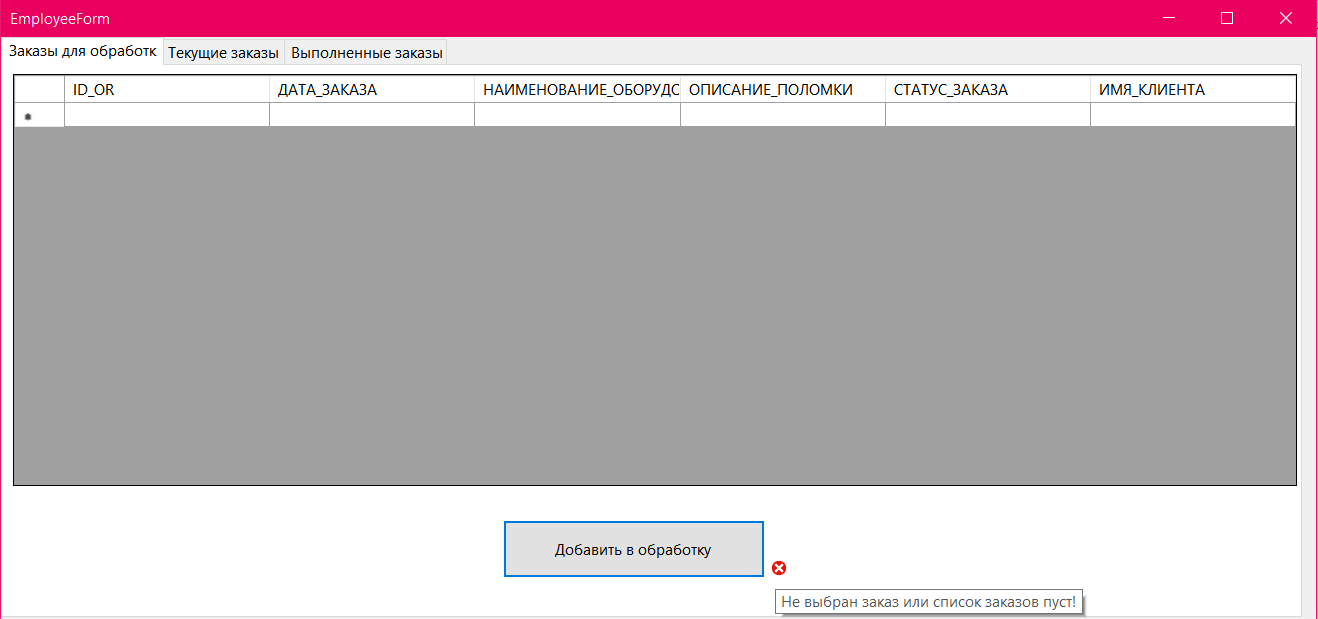


Рисунок 7.2.2 – Меню обработки заказов

В текущих заказах также стоит проверка на пустые поля и наличие заказов (рисунок 7.2.3).

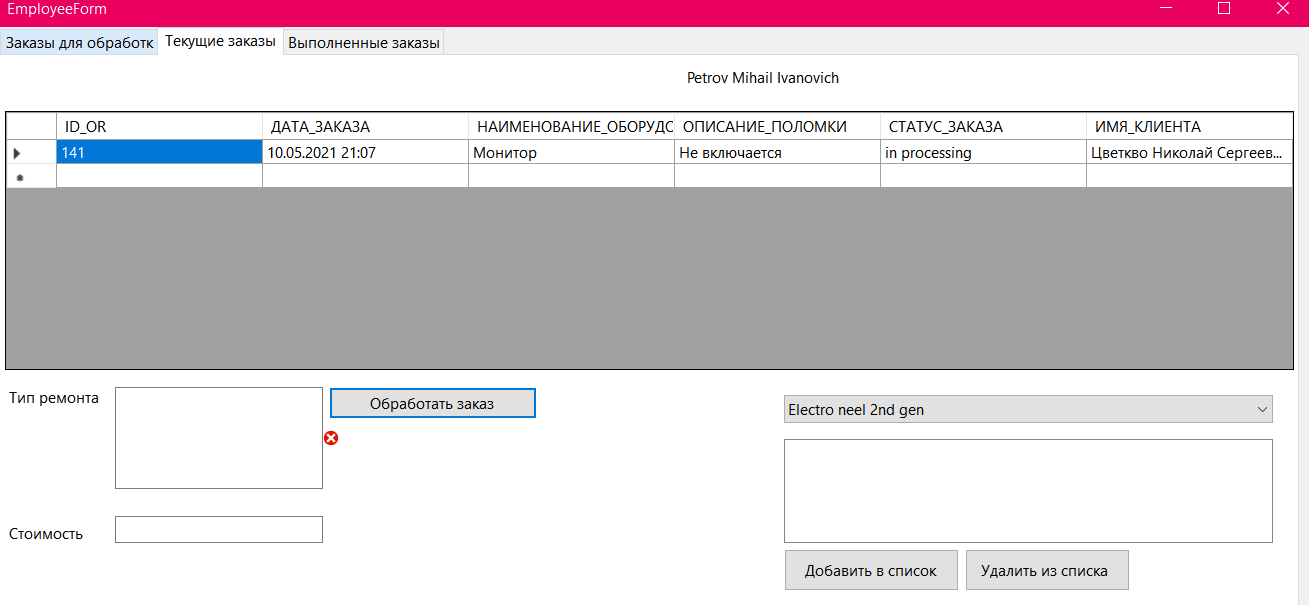


Рисунок 7.2.3 – Меню текущих заказов

Это были единственные места, где работник мог совершить ошибку.

1. **Руководство по использованию программного средства**
   1. Область клиента

Для начала пользователю требуется завести свой аккаунт. Для этого ему требуется корректно заполнить все поля в форме (рисунок 8.1.1).

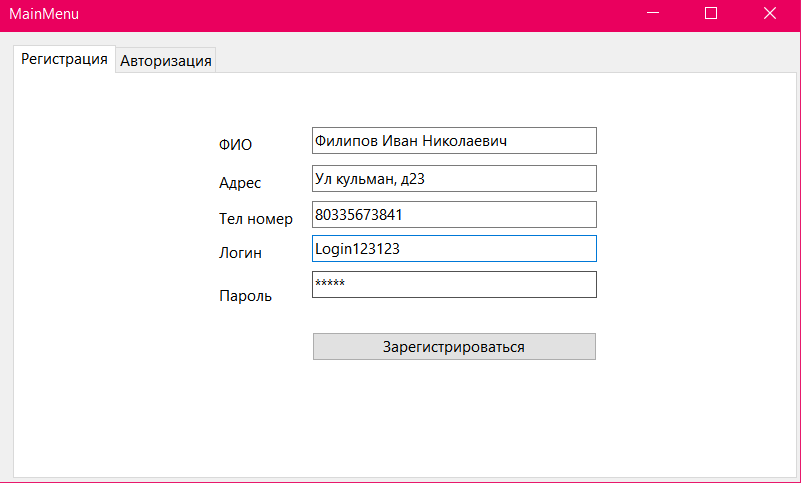


Рисунок 8.1.1 – Меню регистрации

Далее ему требуется авторизоваться. Для этого ему требуется прейти в меню авторизация и ввести свои логин и пароль. После авторизации его встретит окно, в котором он сможет сформировать заказ (рисунок 8.1.2).

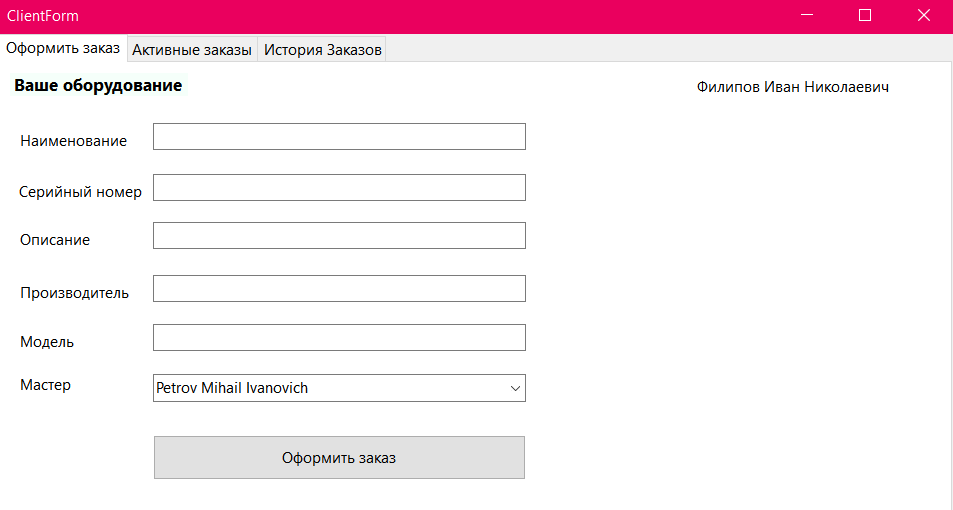


Рисунок 8.1.2 – Меню оформления заказов

После этого клиенту требуется сформировать заказ. Для этого ему потребуется заполнить все поля и нажать на кнопку оформить заказ.

После этого он сможет увидеть свой заказ в меню активных заказов (рисунок 8.1.3).

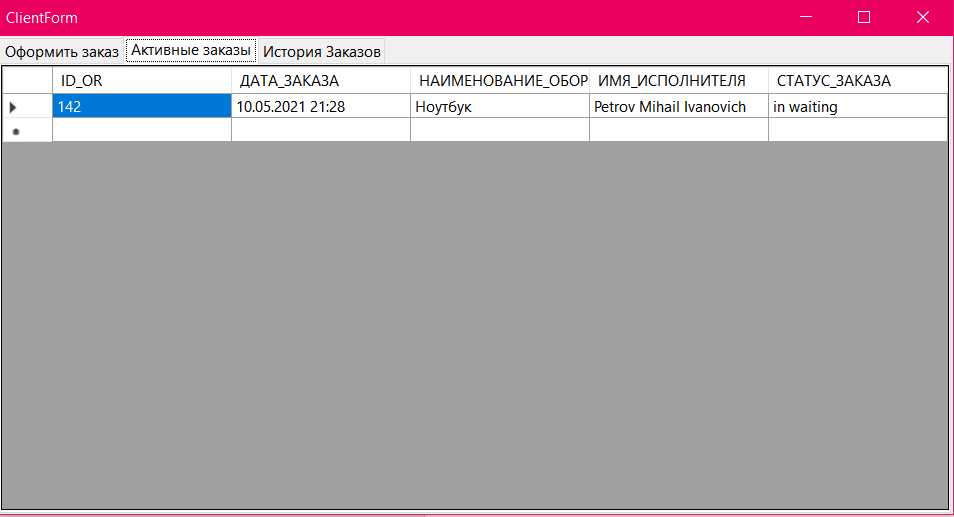


Рисунок 8.1.2 – Меню просмотра активных заказов

После того как работник обработает заказ клиента, клиент сможет увидеть свой заказ в истории заказов. Окно истории заказов схоже с окном активных заказов. Различие заключается в статусе заказа и некоторой информацией, доступной лишь в истории заказов.

* 1. Область сотрудника

Сотрудник имеет похожее окно авторизации, как и у клиента. После авторизации сотруднику будет доступно окно с просмотром активных заказов. Он может выбрать любой на обработку. Для этого ему потребуется нажать на один заказ и таблицы и добавить его в обработку (рисунок 8.2.1).

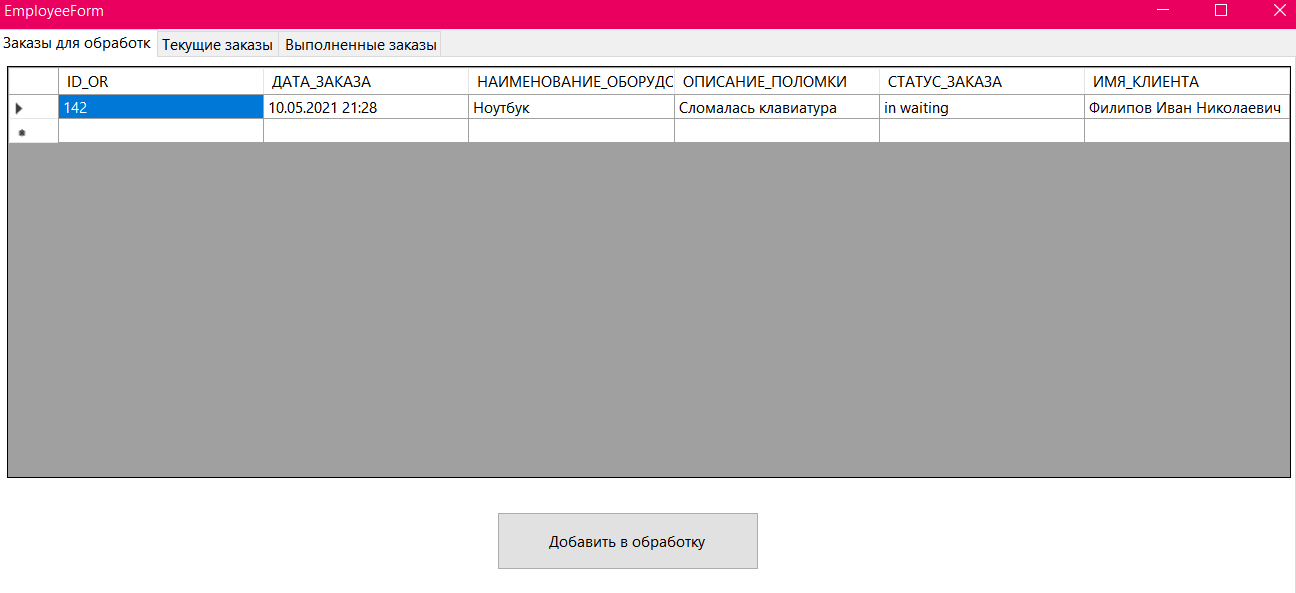


Рисунок 8.2.1 – Меню просмотра заказов для обработки

После того как заказ попал в обработку работник должен его обработать.

Для этого ему требуется перейти в меню текущих заказов, выбрать заказ, заполнить необходимые поля, добавить детали, если они необходимы и выполнить заказ (рисунок 8.2.2).

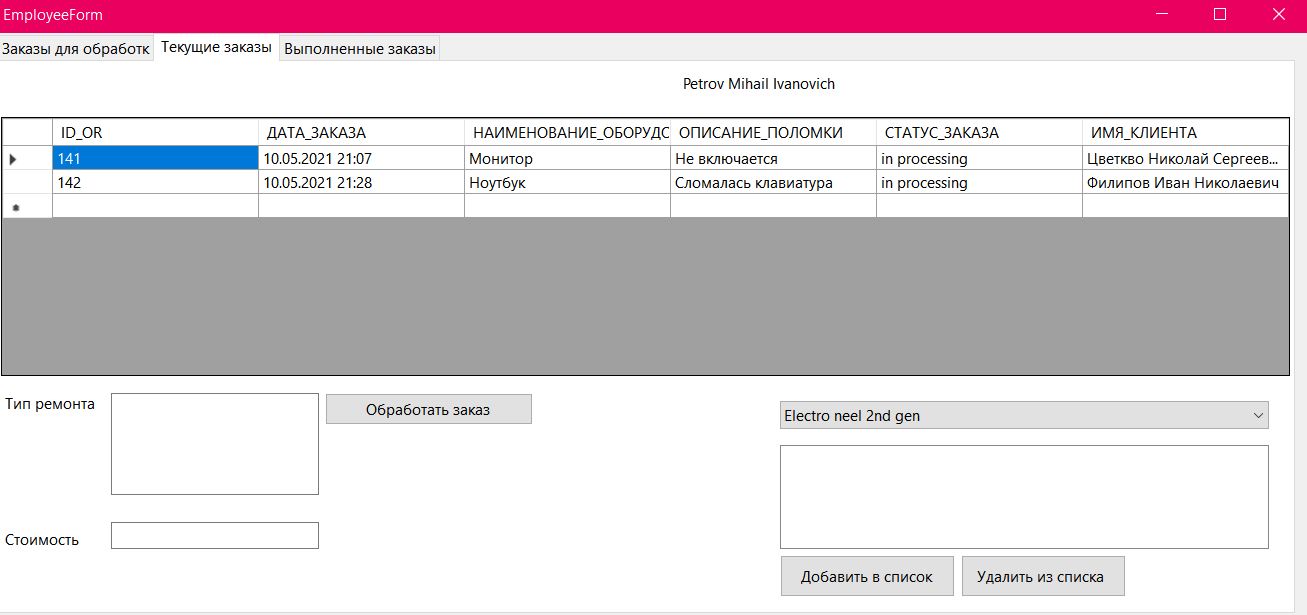


Рисунок 8.2.2 – Меню просмотра текущих заказов

После этого заказ поменяет статус и перейдёт в историю заказов клиента и работника (рисунок 8.2.3).

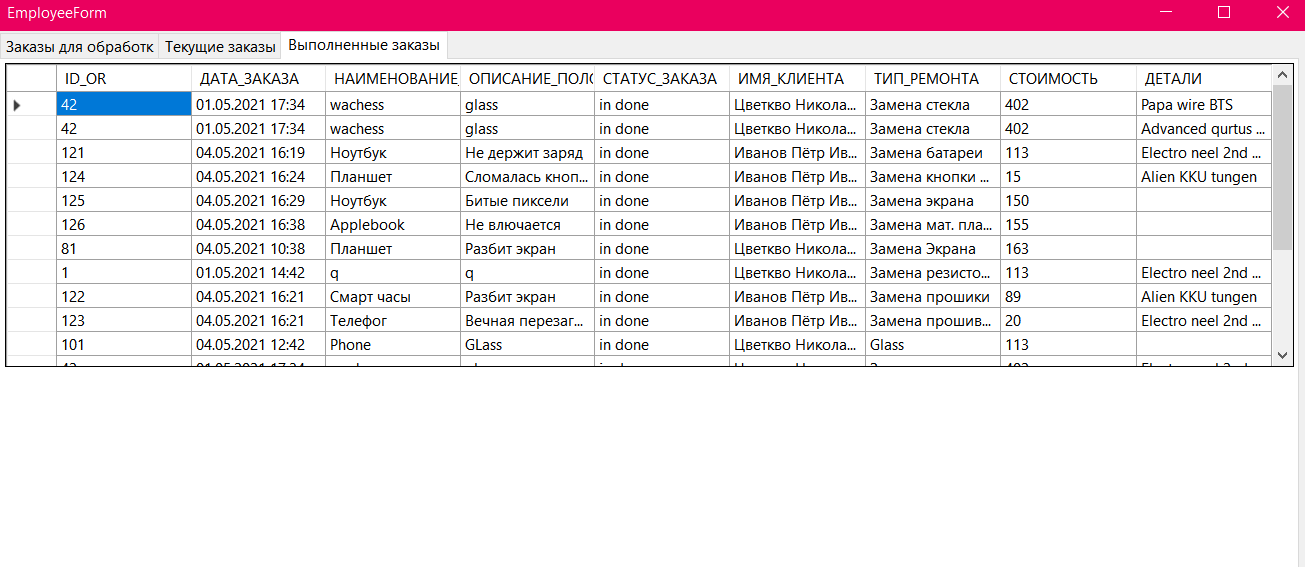


Рисунок 8.2.3 – Меню просмотра истории заказов

Также статус заказа меняется и при добавлении заказа в обработку.

# **Заключение**

Данный курсовой проект был направлен на развитие навыков администрирования базы данных, а также на улучшение понимания взаимодействия программного средства с БД Oracle 12C на удалённом сервере.

В ходе выполнения работы были достигнуты следующие результаты:

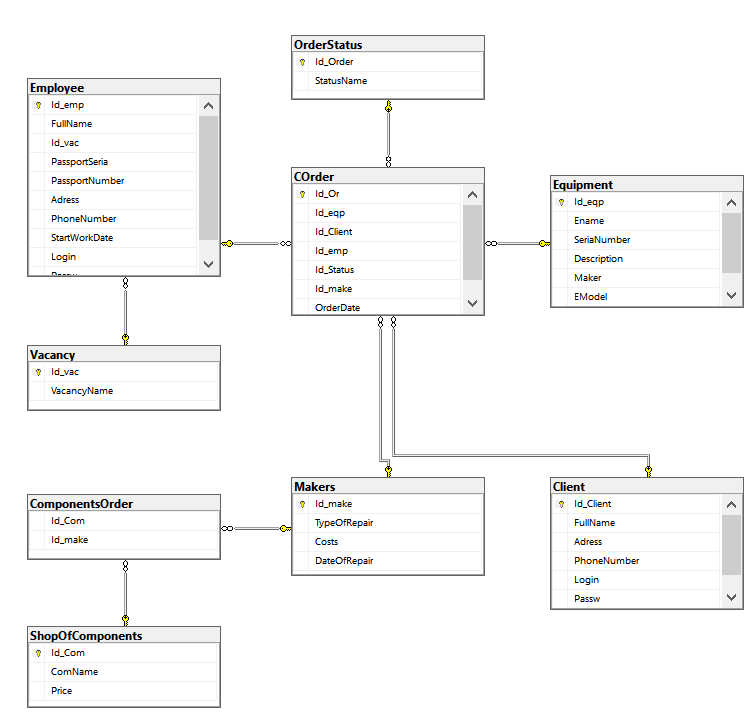
* исследованы обучающие приложения-аналоги;
* сформулирована модель данных и приведено её графическое представление;
* подготовлена база данных Oracle с использованием технологии резервного копирования и восстановления;
* подготовлены процедуры для осуществления импорта и экспорта данных с форматом xml;
* были созданы пользователи с различными привилегиями для работы с базой данных;
* были созданы скрипты для быстрого развертывания начальных таблиц компонентов и начальных таблиц пользователей базы данных с дальнейшим их заполнением;
* разработано приложение для взаимодействия с базой данных.

# **Cписок источников**

1. Язык C# и .NET Framework [Электронный ресурс] – http://professorweb.ru/my/csharp/charp\_theory/level1/infonet.php.
2. Руководство по ADO.NET и работе с базами данных [Электронный ресурс] – https://metanit.com/sharp/adonet/.
3. Документация Oracle [Электронный ресурс] / Foundation, Inc. https://docs.oracle.com/cd/B28359\_01/server.111/b31222/toc.htm
4. Официальный сайт Oracle [Электронный ресурс] / Foundation, Inc. https://www.oracle.com/database/database-vault/index.html
5. Продукты Oracle [Электронный ресурс] [Электронный ресурс] / Foundation, Inc. http://www.interface.ru/home.asp?artId=24678
6. Документы Oracle [Электронный ресурс] / Foundation, Inc. http://www.oracle.com/technetwork/database/security/database-vault-ds-12c-1898877.pdf

## **Приложение А**

**Логическая схема базы данных**



## **Приложение Б**

**Создание таблиц**

create table Client

(

Id\_Client NUMBER GENERATED ALWAYS AS IDENTITY,

CONSTRAINT Client\_pk PRIMARY KEY (Id\_Client),

FullName nvarchar2(50) not null,

Adress nvarchar2(200) not null,

PhoneNumber nvarchar2(40) not null,

Login nvarchar2(150 ) not null UNIQUE,

Passw nvarchar2(150) not null

);

create table Vacancy

(

Id\_vac NUMBER GENERATED ALWAYS AS IDENTITY,

VacancyName nvarchar2(200) not null,

CONSTRAINT Vac\_pk PRIMARY KEY (Id\_vac)

);

create table Employee

(

Id\_emp NUMBER GENERATED ALWAYS AS IDENTITY,

CONSTRAINT Emp\_pk PRIMARY KEY (Id\_emp),

FullName nvarchar2(50) not null,

Id\_vac number not null,

CONSTRAINT fk\_Id\_vac FOREIGN KEY (Id\_vac) REFERENCES Vacancy(Id\_vac),

PassportSeria nvarchar2(60) not null,

PassportNumber nvarchar2(60) not null,

Adress nvarchar2(200) not null,

PhoneNumber nvarchar2(40) not null,

StartWorkDate DATE,

Login nvarchar2(150) not null UNIQUE,

Passw nvarchar2(150) not null

);

create table OrderStatus

(

Id\_Order NUMBER GENERATED ALWAYS AS IDENTITY,

CONSTRAINT IdOrder\_pk PRIMARY KEY (Id\_Order),

StatusName nvarchar2(150) not null

);

create table Equipment

(

Id\_eqp NUMBER GENERATED ALWAYS AS IDENTITY,

CONSTRAINT Eqp\_pk PRIMARY KEY (Id\_eqp),

Ename nvarchar2(150) not null,

SeriaNumber nvarchar2(150) not null,

Description nvarchar2(300) not null,

Maker nvarchar2(100) not null,

EModel nvarchar2(100) not null

);

create table Makers

(

Id\_make NUMBER GENERATED ALWAYS AS IDENTITY,

CONSTRAINT Make\_pk PRIMARY KEY (Id\_make),

TypeOfRepair nvarchar2(200) not null,

Costs number not null,

DateOfRepair date not null

);

create table ComponentsOrder

(

Id\_Com NUMBER not null,

CONSTRAINT fk\_Id\_Com FOREIGN KEY (Id\_Com) REFERENCES ShopOfComponents(Id\_Com),

Id\_make number not null,

CONSTRAINT fk\_Id\_make FOREIGN KEY (Id\_make) REFERENCES Makers(Id\_make)

);

create table ShopOfComponents

(

Id\_Com NUMBER GENERATED ALWAYS AS IDENTITY,

CONSTRAINT Com\_pk PRIMARY KEY (Id\_Com),

ComName nvarchar2(200) not null,

Price number not null

);

create table COrder

(

Id\_Or NUMBER GENERATED ALWAYS AS IDENTITY,

CONSTRAINT Or\_pk PRIMARY KEY (Id\_Or),

Id\_eqp number not null,

CONSTRAINT fk\_Id\_Eq FOREIGN KEY (Id\_eqp) REFERENCES Equipment(Id\_eqp),

Id\_Client number not null,

CONSTRAINT fk\_Id\_Cl FOREIGN KEY (Id\_Client) REFERENCES Client(Id\_Client),

Id\_emp number not null,

CONSTRAINT fk\_Id\_Emp FOREIGN KEY (Id\_emp) REFERENCES Employee(Id\_emp),

Id\_Status number not null,

CONSTRAINT fk\_Id\_OrStatus FOREIGN KEY (Id\_Status) REFERENCES OrderStatus(Id\_Order),

Id\_make number default null,

CONSTRAINT fk\_Id\_mk FOREIGN KEY (Id\_make) REFERENCES Makers(Id\_make),

OrderDate date not null

);

Приложение В

**Тело пакета с процедурами для работы клиента**

create or replace package body cwPack1

as

procedure addClient(fullName nvarchar2, Adress nvarchar2, PhoneNumber nvarchar2, Login nvarchar2, Passw nvarchar2)

as

begin

insert into Client(fullname, adress,phonenumber, login, passw) values(fullName, Adress, PhoneNumber, Login, Passw);

commit;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end addClient ;

procedure getCountOfClientWithSameLogin(checklogin nvarchar2, results OUT number)

as

begin

results:=0;

select count(\*) into results from Client where Login=checklogin;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end getCountOfClientWithSameLogin;

procedure checkClientAccount(lg nvarchar2, ps nvarchar2, results out number)

as

begin

results:=0;

select Count(\*) into results from client where Login=lg and Passw=ps;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end checkClientAccount;

procedure getClienIdAndName(lgin nvarchar2, psd nvarchar2, id\_ret out number, fio out nvarchar2)

as

begin

select Id\_Client into id\_ret from Client where Login=lgin and Passw=psd;

select FullName into fio from Client where Login=lgin and Passw=psd;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end getClienIdAndName;

procedure getNameAndIdEmp(p\_cursor IN OUT NOCOPY SYS\_REFCURSOR)

as

begin

OPEN p\_cursor FOR

select Id\_emp, fullName from employee;

end getNameAndIdEmp;

procedure addEquipment(eqname nvarchar2, sernum nvarchar2, descr nvarchar2, eqmakers nvarchar2, eqmodel nvarchar2)

as

begin

insert into equipment(Ename, SeriaNumber, Description, Maker, EModel) values(eqname,sernum, descr, eqmakers, eqmodel);

commit;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end addEquipment;

procedure getEpuipmentIdForOrder(results out number)

as

begin

results:=0;

select max(id\_eqp) into results from equipment;

end getEpuipmentIdForOrder;

procedure makeOrder(eq\_Id number,client\_Id number, empl\_Id number, status\_id number, dateO date)

as

begin

insert into corder(Id\_eqp, Id\_Client, Id\_emp, Id\_Status, OrderDate) values(eq\_Id ,client\_Id , empl\_Id , status\_id , dateO );

commit;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end makeOrder;

procedure showCurrentClientOrders(cl\_id number, p\_cursor IN OUT NOCOPY SYS\_REFCURSOR)

as

begin

OPEN p\_cursor FOR

select distinct Corder.id\_or, corder.orderdate as Дата\_Заказа, equipment.ename as Наименование\_оборудования, Employee.FULLNAME as Имя\_исполнителя, OrderStatus.statusname as Статус\_заказа

from Corder

inner join Client on Corder.Id\_client = cl\_id

inner join equipment on corder.id\_eqp = equipment.Id\_eqp

inner join Employee on Corder.Id\_emp = Employee.Id\_Emp

inner join OrderStatus on Corder.Id\_status = OrderStatus.Id\_order

where Corder.Id\_status=1 or Corder.Id\_status=2;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end showCurrentClientOrders;

procedure showHistoryClientOrders(cl\_id number, p\_cursor IN OUT NOCOPY SYS\_REFCURSOR)

as

begin

OPEN p\_cursor FOR

select distinct Corder.id\_or, corder.orderdate as Дата\_Заказа, equipment.ename as Наименование\_оборудования, Employee.FULLNAME as Имя\_исполнителя, OrderStatus.statusname as Статус\_заказа, makers.COSTS as Стоимость

from Corder

inner join Client on Corder.Id\_client = cl\_id

inner join equipment on corder.id\_eqp = equipment.Id\_eqp

inner join Employee on Corder.Id\_emp = Employee.Id\_Emp

inner join OrderStatus on Corder.Id\_status = OrderStatus.Id\_order

inner join Makers on COrder.ID\_MAKE = makers.ID\_MAKE

where Corder.Id\_status=3;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end showHistoryClientOrders;

procedure addComponentsOrder(idcomp number, idmaker number)

as

begin

insert into ComponentsOrder(Id\_Com, Id\_make) values(idcomp, idmaker);

commit;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end addComponentsOrder;

end cwPack1;

Приложение Г

**Тело пакета с процедурами для работы сотрудника**

create or replace package body cwPack2

as

procedure checkEmpAccount(login1 nvarchar2, pass1 nvarchar2, results out number)

as

begin

results:=0;

select count(\*) into results from employee where (Login=login1 and Passw=pass1);

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end checkEmpAccount;

procedure getCurrentEmplIdAndName(lgine nvarchar2, psde nvarchar2, id\_rete out number, fioe out nvarchar2)

as

begin

select Id\_Emp into id\_rete from Employee where Login=lgine and Passw=psde;

select FullName into fioe from Employee where Login=lgine and Passw=psde;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end getCurrentEmplIdAndName;

procedure getComponentsNameAndId(p\_cursor IN OUT NOCOPY SYS\_REFCURSOR)

as

begin

OPEN p\_cursor FOR

select Id\_com, comname, price from ShopOfComponents;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end getComponentsNameAndId;

procedure getClientOrdersForEmployeeToDo(eplid number, p\_cursor IN OUT NOCOPY SYS\_REFCURSOR)

as

begin

OPEN p\_cursor FOR

select distinct Corder.id\_or, corder.orderdate as Дата\_Заказа, equipment.ename as Наименование\_оборудования, equipment.Description as Описание\_поломки, OrderStatus.statusname as Статус\_заказа, Client.Fullname as Имя\_Клиента

from Corder

inner join Client on Corder.Id\_client = Client.Id\_client

inner join equipment on corder.id\_eqp = equipment.Id\_eqp

inner join Employee on Corder.Id\_emp = 3

inner join OrderStatus on Corder.Id\_Status= OrderStatus.ID\_order

where Corder.Id\_Status=1;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end getClientOrdersForEmployeeToDo;

procedure changeOrderStatus(idor number)

as

begin

update COrder set Id\_Status=2 where Id\_Or=idor;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end changeOrderStatus;

procedure getClientOrdersForEmployee(eplid number, p\_cursor IN OUT NOCOPY SYS\_REFCURSOR)

as

begin

OPEN p\_cursor FOR

select distinct Corder.id\_or, corder.orderdate as Дата\_Заказа, equipment.ename as Наименование\_оборудования, equipment.Description as Описание\_поломки, OrderStatus.statusname as Статус\_заказа, Client.Fullname as Имя\_Клиента

from Corder

inner join Client on Corder.Id\_client = Client.Id\_client

inner join equipment on corder.id\_eqp = equipment.Id\_eqp

inner join Employee on Corder.Id\_emp = eplid

inner join OrderStatus on Corder.Id\_Status= OrderStatus.ID\_order

where Corder.Id\_Status=2;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end getClientOrdersForEmployee;

procedure addMakers(repairtype nvarchar2, price number, repairdate date)

as

begin

insert into Makers(TypeOfRepair, Costs, DateOfRepair) values(repairtype,price, repairdate);

commit;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end addMakers;

procedure getLastMakers(idmk out number)

as

begin

idmk:=0;

select max(id\_make) into idmk from Makers;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end getLastMakers;

procedure changeStatusAndMakers(ior number, idmk number)

as

begin

update COrder set ID\_MAKE=idmk, Id\_Status=3 where Id\_Or=ior;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end changeStatusAndMakers;

procedure getOrdersHistoryForEmployee(eplid number, p\_cursor IN OUT NOCOPY SYS\_REFCURSOR)

as

begin

OPEN p\_cursor FOR

select distinct Corder.id\_or, corder.orderdate as Дата\_Заказа, equipment.ename as Наименование\_оборудования, equipment.Description as Описание\_поломки, OrderStatus.statusname as Статус\_заказа, Client.Fullname as Имя\_Клиента, makers.typeofrepair as Тип\_Ремонта, makers.COSTS as Стоимость, ShopOfComponents.comname as Детали

from Corder

inner join Client on Corder.Id\_client = Client.Id\_client

inner join equipment on corder.id\_eqp = equipment.Id\_eqp

inner join Employee on Corder.Id\_emp = eplid

inner join OrderStatus on Corder.Id\_Status= OrderStatus.ID\_order

inner join Makers on COrder.ID\_MAKE = makers.ID\_MAKE

left join componentsorder on COrder.ID\_MAKE = componentsorder.ID\_MAKE

left join ShopOfComponents on ShopOfComponents.ID\_Com = componentsorder.ID\_Com

where Corder.Id\_Status=3;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end getOrdersHistoryForEmployee;

procedure addComponentsOrder(idcomp number, idmaker number)

as

begin

insert into ComponentsOrder(Id\_Com, Id\_make) values(idcomp, idmaker);

commit;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end addComponentsOrder;

end cwPack2;

Приложение Г

**Тело пакета с процедурами для работы администратора БД**

create or replace package body cwPackadm

as

procedure showAllClients

as

begin

for client in (select id\_client, fullname, adress,phonenumber from Client)

loop

dbms\_output.put\_line(client.id\_client|| ' ' ||client. fullname|| ' ' ||client.adress|| ' ' ||client.phonenumber);

end loop;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end showAllClients ;

procedure addVacancy(vacName nvarchar2)

as

begin

insert into Vacancy(vacancyname) values(vacName);

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end addVacancy;

procedure showAllVacancy

as

begin

for vac in (select \* from Vacancy)

loop

dbms\_output.put\_line(vac.Id\_Vac|| ' ' ||vac.VacancyName);

end loop;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end showAllVacancy;

procedure addEmp(fullName nvarchar2, Id\_vac number, PassportSeria nvarchar2, PassportNumber nvarchar2, Adress nvarchar2, PhoneNumber nvarchar2, StartWorkDate date, Login nvarchar2, Passw nvarchar2)

as

begin

insert into Employee(fullname, id\_vac, passportseria,passportnumber,adress,phonenumber,startworkdate, login, passw) values(fullName, Id\_vac, PassportSeria, PassportNumber, Adress, PhoneNumber, StartWorkDate, Login, Passw);

commit;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end addEmp;

procedure showAllEmp

as

begin

for emp in (select fullname, id\_vac, passportseria,passportnumber,adress,phonenumber,startworkdate from Employee)

loop

dbms\_output.put\_line(emp.fullname|| ' ' ||emp.passportseria|| ' ' ||emp.passportnumber|| ' ' ||emp.adress|| ' ' ||emp.phonenumber|| ' ' ||to\_char(emp.startworkdate,'DD-MM-YYYY'));

end loop;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end showAllEmp;

procedure addStatus(stName nvarchar2)

as

begin

insert into OrderStatus(StatusName) values(stName);

commit;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end addStatus;

procedure showAllOrderStatus

as

begin

for status in (select StatusName from OrderStatus)

loop

dbms\_output.put\_line(status.statusName);

end loop;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end showAllOrderStatus;

PROCEDURE GET (ConscriptsOut OUT sys\_refcursor)

as

BEGIN

OPEN ConscriptsOut FOR

SELECT id\_client, fullname, adress,phonenumber

FROM Client;

end GET;

procedure importXmlDataFromEmployee

as

F UTL\_FILE.FILE\_TYPE;

MYCLOB CLOB;

begin

SELECT

DBMS\_XMLGEN.GETXML('

SELECT

ID\_EMP, FULLNAME, ID\_VAC,PASSPORTSERIA, PASSPORTNUMBER, ADRESS, PHONENUMBER, STARTWORKDATE LOGIN, PASSW

FROM

EMPLOYEE') INTO MYCLOB FROM DUAL;

F:= UTL\_FILE.FOPEN('C:\XML','IMPORTEMP.XML','W');

UTL\_FILE.PUT(F,MYCLOB);

UTL\_FILE.FCLOSE(F);

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end importXmlDataFromEmployee;

procedure importXmlDataFromComponents

as

F UTL\_FILE.FILE\_TYPE;

MYCLOB CLOB;

begin

SELECT

DBMS\_XMLGEN.GETXML('

SELECT

ID\_COM, COMNAME, PRICE

FROM

ShopOfComponents') INTO MYCLOB FROM DUAL;

F:= UTL\_FILE.FOPEN('C:\XML','IMPORTShopOfComponents.XML','W');

UTL\_FILE.PUT(F,MYCLOB);

UTL\_FILE.FCLOSE(F);

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end importXmlDataFromComponents;

procedure exportXmlToClients

as

begin

insert into Client (FULLNAME, ADRESS, PHONENUMBER, LOGIN, PASSW)

SELECT \*

FROM XMLTABLE('/ROWSET/ROW'

PASSING XMLTYPE(BFILENAME('DIR','CLIENTS1.XML'),

NLS\_CHARSET\_ID('CHAR\_CS'))

COLUMNS FullName nvarchar2(50) PATH 'FullName',

Adress nvarchar2(200) PATH 'Adress',

PhoneNumber nvarchar2(40) PATH 'Phone',

Login nvarchar2(150) PATH 'Login',

Passw nvarchar2(150) PATH 'Passw');

commit;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end exportXmlToClients;

procedure addComponents(componname nvarchar2, compcost number)

as

begin

insert into ShopOfComponents (ComName, Price) values(componname, compcost);

commit;

exception

when others then

raise\_application\_error(-20001,'An error was encountered - '||SQLCODE||' -ERROR- '||SQLERRM);

end addComponents;

end cwPackadm;