МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Специализация 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет-изданий)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

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

Выполнил студент Сятковская Е. Д.

(Ф.И.О.)

Руководитель проекта ассистент Нистюк О. А.

(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: ассистент Нистюк О. А.

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

Нормоконтролер: ассистент Нистюк О. А.

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

Курсовой проект защищен с оценкой

Минск 2022

**Содержание**

[Введение 3](#_Toc122095178)

[1 Аналитический обзор аналогов 4](#_Toc122095179)

[1.1 Программное средство «РемОнлайн» 4](#_Toc122095180)

[1.2 Программное средство «RepairCenter» 4](#_Toc122095181)

[2 Разработка архитектуры проекта 6](#_Toc122095182)

[2.1 Обобщенная структура управления приложением 6](#_Toc122095183)

[2.2 Проектирование модели базы данных 7](#_Toc122095184)

[3 Использование Oracle 12c 11](#_Toc122095185)

[3.1 Создание таблиц 11](#_Toc122095186)

[3.2 Создание ролей для разграничения доступа 11](#_Toc122095187)

[3.3 Создание пакетов и процедур для работы с базой данных 12](#_Toc122095188)

[3.4 Описание процедур экспорта и импорта 14](#_Toc122095189)

[3.5 Тестирование производительности базы данных 15](#_Toc122095190)

[4 Описание технологий 18](#_Toc122095191)

[4.1 Основная технология резервного копирования и восстановления 18](#_Toc122095192)

[4.2 Технология шифрования 19](#_Toc122095193)

[5 Руководство пользователя 21](#_Toc122095194)

[5.1 Область клиента 21](#_Toc122095195)

[5.2 Область сотрудника 23](#_Toc122095196)

[5.3 Тестирование клиентской области 25](#_Toc122095197)

[5.4 Тестирование области работника 27](#_Toc122095198)

[Заключение 29](#_Toc122095199)

[Cписок используемых источников 30](#_Toc122095200)

[Приложение А 31](#_Toc122095201)

[Приложение Б 35](#_Toc122095202)

[Приложение В 39](#_Toc122095203)

# **Введение**

В данной записке приведено описание реляционной базы данных и desktop-приложения, разработанных в соответствии с заданием на курсовое проектирование по теме «База данных для службы по ремонту техники с использованием технологии резервного копирования и восстановления».

Цель разработки базы данных – спроектировать максимально удобную для встраивания в программное средство базу данных, внутри которой будет осуществлена вся работа с таблицами с помощью 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 клиент и работник. Это было сделано для того, чтобы пользователь не смог повредить базу данных.

В качестве интерфейса выбран API Windows Forms. Для работы с данным интерфейсом использовался объектно-ориентированный язык программирования C#.

Приложение будет обладать следующими функционалом:

* регистрация клиентов;
* авторизация клиентов и сотрудников;
* формирование заказа клиентом;
* просмотр заказов клиентов сотрудниками;
* изменение статуса заказа;
* заказ деталей сотрудником.

Для разработки и управления базой данных курсового проекта использовалась система управления реляционными базами данных Oracle 12с.

# **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 «РемОнлайн»

На рисунке 1.1 предствлено окно приложения «РемОнлайн»

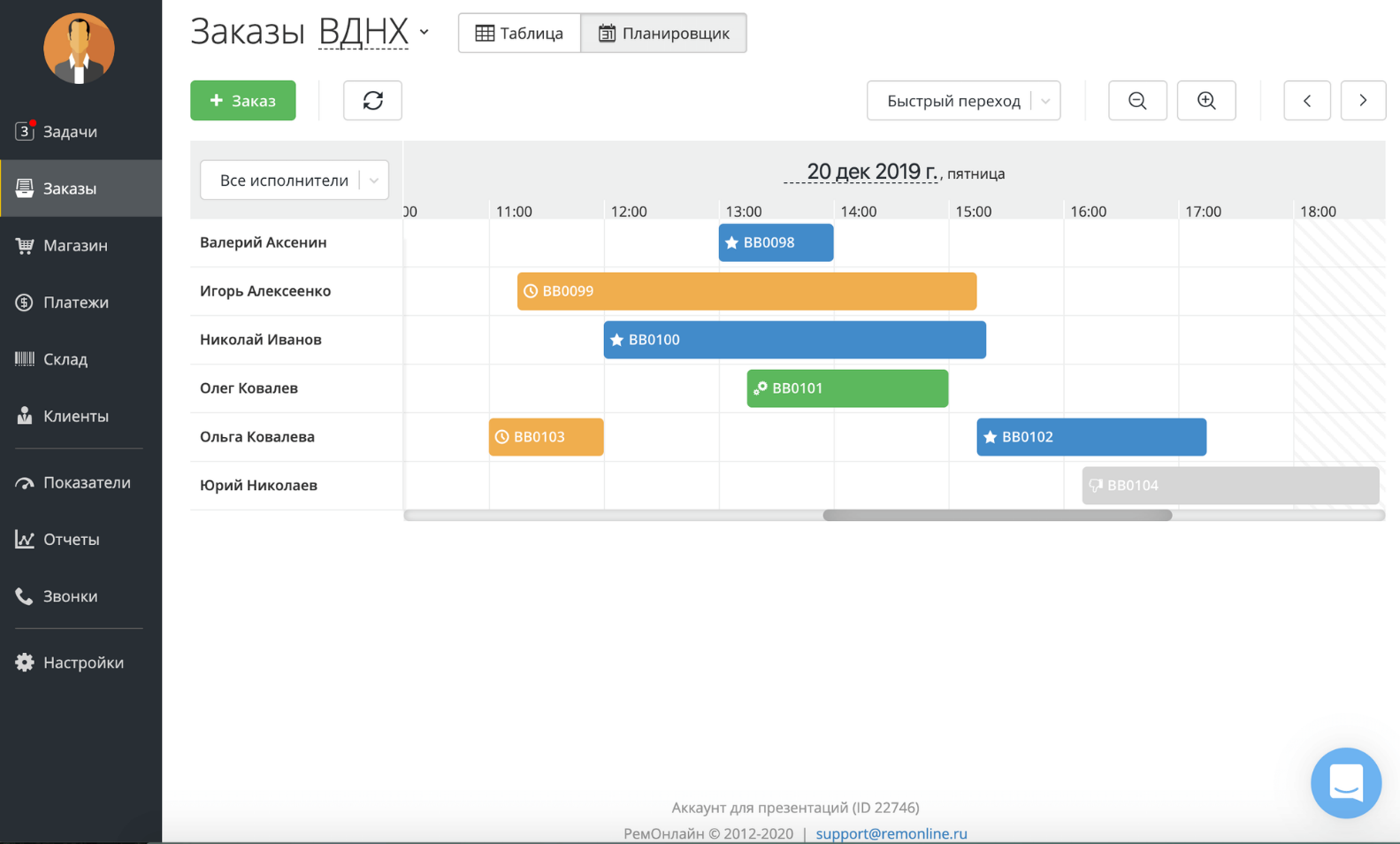


Рисунок 1.1 – Окно аналога приложения «РемОнлайн»

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

* использование списка для представления множества данных;
* возможность просматривать клиентов, заказы, склад;
* просмотр отчётов;
* возможность обновлять данные в списке;
* просмотр итого по выборке.

1.2 Программное средство «RepairCenter»

На рисунке 1.2 представлено главное окно приложения «RepairCenter».

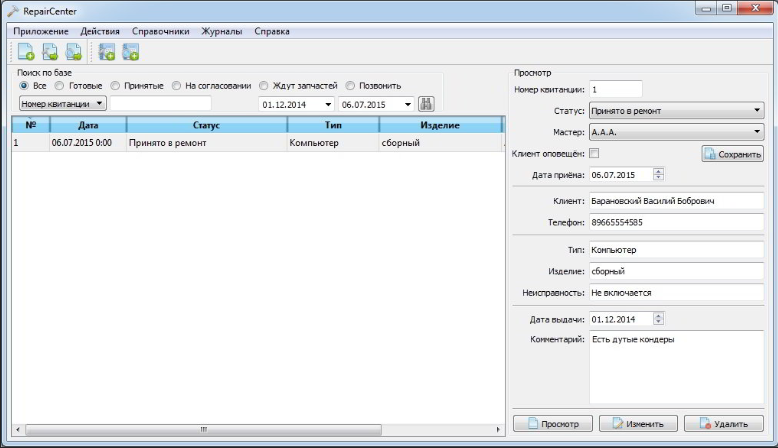


Рисунок 1.2 – Окно аналога приложения «RepairCenter»

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

* наличие формы приёмной квитанции с множеством различных полей для заполнения;
* возможность просмотра истории;
* возможность просмотра планируемых работ;
* возможность формирования квитанции.

# **2 Разработка архитектуры проекта**

## **2.1** **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.

Для наглядности демонстрации возможностей пользователей воспользуемся UML-диаграммами. UML (Unified Modeling Language – уникальный язык моделирования) – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем.

Диаграмма вариантов использования приложения для клиента представлена на рисунке 2.1.

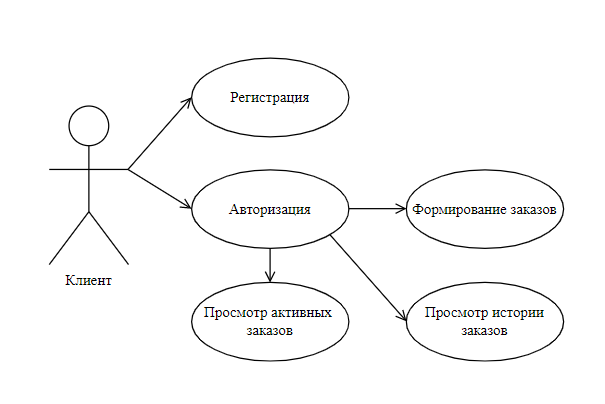


Рисунок 2.1 – Диаграмма использования приложения для клиента

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

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

Диаграмма вариантов использования приложения для работника представлена на рисунке 2.2.

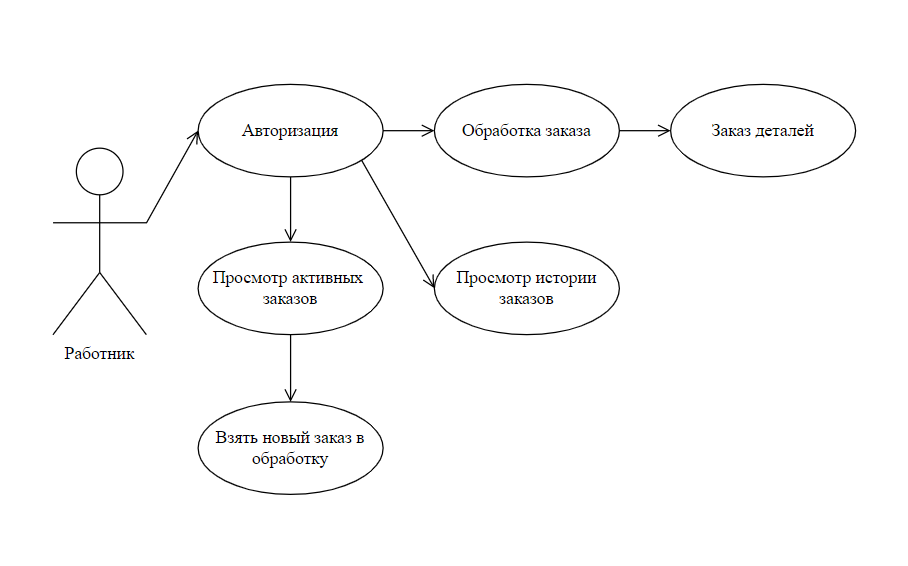


Рисунок 2.2 – Диаграмма использования приложения для работника

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 – SqlDeveloper или SqlPlus. Он ответственен за инициализацию, контроль и мониторинг информационной системы.

## **2.2 Проектирование 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 2.3).

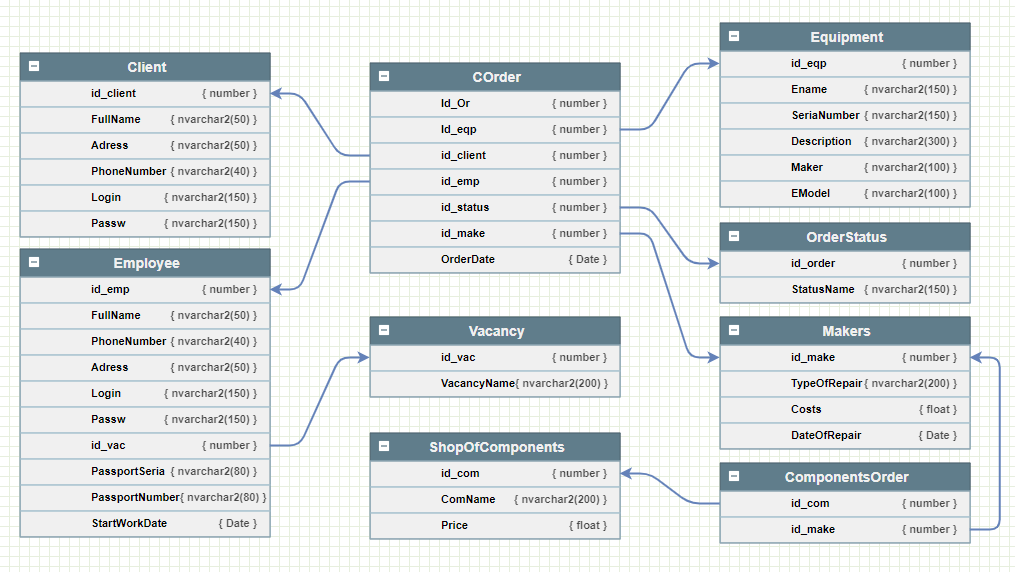


Рисунок 2.3 – Взаимосвязь таблиц данных

Таблица «Client» содержит информацию о клиентах. Описание ее полей приведено в таблице 2.1.

Таблица 2.1 – Структура таблицы «Client»

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

Таблица «Employee» содержит информацию о сотрудниках. Описание ее полей приведено в таблице 2.2.

Таблица 2.2 – Структура таблицы «Employee»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_Emp | number | Идентификатор сотрудника |
| FullName | nvarchar2 | ФИО |
| Id\_Vac | number | Идентификатор должности |
| PassportSeria | nvarchar2 | Серия паспорта |
| PassportNumber | nvarchar2 | Номер паспорта |

Продолжение таблицы 2.2

|  |  |  |
| --- | --- | --- |
| 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. Описание ее полей приведено в таблице 2.3.

Таблица 2.3 – Структура таблицы «OrderStatus»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_Order | number | Идентификатор заказа |
| StatusName | nvarchar2 | Имя статуса |

Таблица «Equipment» содержит информацию о сломанном оборудовании. Описание ее полей приведено в таблице 2.4.

Таблица 2.4 – Структура таблицы «Equipment»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_Eqp | number | Идентификатор оборудования |
| EName | nvarchar2 | Наименование оборудования |
| SeriaNumber | nvarchar2 | Серия оборудования |
| Description | nvarchar2 | Описание проблемы |
| Maker | nvarchar2 | Мастер |
| EModel | nvarchar2 | Модель |

Таблица «Makers» содержит информацию о проделанном ремонте. Описание ее полей приведено в таблице 2.5.

Таблица 2.5 – Структура таблицы «Makers»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_Make | number | Идентификатор ремонта |
| TypeOfRepair | nvarchar2 | Тип ремонта |
| Costs | number | Стоимость |
| DateOfRepair | date | Дата исполнения |

Таблица «ComponentsOrder» содержит информацию об использованных комплектующих мастером. Описание ее полей приведено в таблице 2.6.

Таблица 2.6 – Структура таблицы «ComponentsOrder»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_Com | number | Идентификатор компонента |
| Id\_Make | number | Идентификатор мастера |

Таблица «ShopOfComponents» содержит информацию о комплектующих. Описание ее полей приведено в таблице 2.7.

Таблица 2.7 – Структура таблицы «ShopOfComponents»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_Com | number | Идентификатор детали |
| ComName | nvarchar2 | Имя детали |
| Price | number | Цена |

Таблица «Vacancy» содержит информацию о должностях. Описание ее полей приведено в таблице 2.8.

Таблица 2.8 – Структура таблицы «Vacancy»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_Vac | number | Идентификатор должности |
| VacancyName | nvarchar2 | Имя должности |

Таблица «COrder» содержит основную информацию о заказах. Описание ее полей приведено в таблице 2.9.

Таблица 2.9 – Структура таблицы «COrder»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| Id\_Or | number | Идентификатор заказа |
| Id\_Eqp | number | Идентификатор оборудования |
| Id\_Client | number | Идентификатор клиента |
| Id\_Emp | number | Идентификатор работника |
| Id\_Status | number | Идентификатор статуса |
| Id\_Make | number | Идентификатор ремонта |
| OrderDate | date | Дата заказа |

Данный таблицы представляют логическую структуру базы данных.

# **3 Использование Oracle 12c**

## **3.1 Создание таблиц**

Для работы в базе данных мы должны создать таблицы и организовать между ними связи, которые будут представлять логическую структуру базы данных. 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 Tables.sql, для дальнейшей возможности запускать скрипт в целом файле в новой базе данных.

## **3.2 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 две роли им были выданы привилегии, также было создано два пользователя и назначены им роли (листинг 3.1).

create role RLClient;

grant create session to RLClient;

grant execute on cwClient to RLClient;

create role RLEmpl;

grant create session to RLEmpl;

grant execute on cwEmpl to RLEmpl;

create user C##Client identified by Password123;

create user C##Employee identified by Password123;

grant RLClient to C##Client;

grant RLEmpl to C##Employee;

Листинг 3.1 – Создание пользователей и выдача им привилегий

## **3.3 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:

* cwClient – пакет процедур для клиента;
* cwEmpl – пакет процедур для работника;
* cwAdmin – пакет процедур для администратора базы данных.

Заголовок пакета, содержащий процедуры для работы клиента представлен в листинге 3.2. Тело пакета расположено в приложении А.

create or replace package cwClient as

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

procedure getCountOfClientWithSameLogin(checklogin nvarchar2, results OUT number);

procedure getCountOfClientWithSamePhone(checkphone nvarchar2, results OUT number);

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

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

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

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

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

procedure getEpuipmentIdForOrder(results out number);

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

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

end cwClient;

Листинг 3.2 – Заголовок пакета cwClient

Данный пакет содержит процедуру addClient для регистрации клиентов, checkClientAccount отвечающую за авторизацию, makeOrder для оформления заказа, showCurrentClientOrders осуществляющая просмотр текущих заказов клиента, showHistoryClientOrders – просмотр выполненных заказов.

Заголовок пакета, содержащий процедуры для работы сотрудника представлен в листинге 3.3. Тело пакета расположено в приложении Б.

create or replace package cwEmpl as

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

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

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

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

procedure changeOrderStatus(idor number);

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

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

procedure getLastMakers(idmk out number);

procedure changeStatusAndMakers(ior number, idmk number);

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

procedure addComponentsOrder(idcomp number, idmaker number);

end cwEmpl;

Листинг 3.3 – Заголовок пакета cwEmpl

Данный пакет содержит процедуру checkEmpAccount отвечающая за авторизацию сотрудника в системе, getClientOrdersForEmployeeToDo для отображения новых заказов, getOrdersHistoryForEmployee выводящая список выполненных заказов.

Заголовок пакета, содержащий процедуры для работы администратора базы данных представлен в листинге 3.4.

create or replace package cwAdmin as

procedure showAllClients;

procedure addVacancy(vacName nvarchar2);

procedure showAllVacancy;

procedure addEmp(fullName nvarchar2, Id\_vac number, PSeria nvarchar2, PNumber nvarchar2, Adress nvarchar2,PhoneN nvarchar2, StartWorkDate date, Logi nvarchar2, Passw nvarchar2);

procedure showAllEmp;

procedure addStatus(stName nvarchar2);

procedure showAllOrderStatus;

procedure get (ConscriptsOut out sys\_refcursor);

procedure importXmlDataFromComponents;

procedure exportXmlToClients;

procedure addComponents(componname nvarchar2, compcost number);

procedure insert\_100k\_components;

end cwAdmin;

Листинг 3.4 – Заголовок пакета cwAdmin

Тело пакета администратора расположено в приложении В.

Данный пакет содержит процедуры addVacancy, addEmp, addStatus, addComponents для заполнения таблиц Vacancy, Employee, OrderStatus, ShopOfComponents соответственно. Процедуры showAllClients, showAllVacancy, showAllEmp, showAllOrderStatus для вывода информации из соответствующих таблиц. Процедуры exportXmlToClients, importXmlDataFromComponents предназначены для экспорта/ импорта информации.

## **3.4 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 база данных Oracle. Владельцем всех объектов directory в базе данных является пользователь sys, даже если объект directory создан другим пользователем. Имена объектов directory уникальны внутри всей базы данных. (листинг 3.5).

create or replace directory DIR as 'C:\XML';

Листинг 3.5 – Создание объекта directory

В листинге 3.6 представлена процедура импорта данных из файла components\_import.xml в базу данных.

procedure importXmlDataFromComponents

is

begin

insert into shopofcomponents (comname, price)

select ExtractValue(value(components), '//NAME') as comname,

ExtractValue(value(components), '//PRICE') as price

from table(XMLSequence(extract(xmltype(bfilename('DIR', 'components\_import.xml'),

nls\_charset\_id('UTF-8')),'/ROWSET/ROW'))) components;

exception

when others then

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

end importXmlDataFromComponents;

Листинг 3.6 – Процедура импорта данных

В листинге 3.7 представлена процедура экспорта данных из базы данных в файл client\_export.xml.

procedure exportXmlToClients

is

rc sys\_refcursor;

doc DBMS\_XMLDOM.DOMDocument;

begin

open rc for

select id\_client,fullname,adress,phonenumber,login,passw from client;

doc := DBMS\_XMLDOM.NewDOMDocument(xmltype(rc));

DBMS\_XMLDOM.WRITETOFILE(doc, 'DIR/client\_export.xml');

commit;

exception

when others then

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

end exportXmlToClients;

Листинг 3.7 – Процедура экспорта данных

Данные процедуры расположены в пакете администратора cwAdmin.

## **3.5 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 процедуры insert\_100k\_components, которая представлена в листинге 3.8, было добавлено 100000 строк.

procedure insert\_100k\_components

is

begin

for i in 1 .. 100000

loop

cwAdmin.addComponents(i, i);

end loop;

commit;

exception

when others then

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

end insert\_100k\_components;

Листинг 3.8 – Процедура заполнения таблицы

На рисунке 3.1 демонстрируется количество записей в заполненной процедурой таблице.

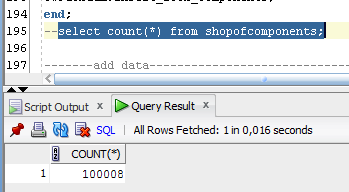


Рисунок 3.1 – Количество записей в таблице

Построим план запроса по всем элементам таблицы и посмотрим время его выполнения без индекса (рисунок 3.2).

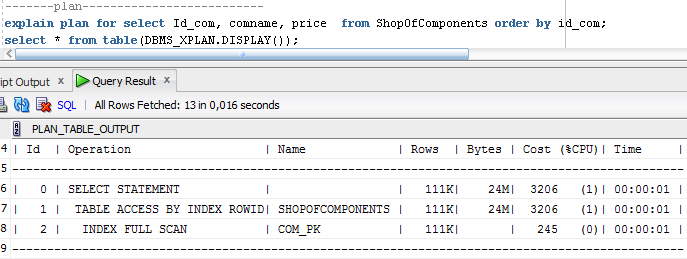


Рисунок 3.2 – План запроса без индекса

Для оптимизации работы с поисковыми запросами необходимо использовать индексы. Поэтому для этой таблицы был реализован индекс по наиболее используемой выборке (листинг 3.9).

create index shopofcomp\_idx on ShopOfComponents (id\_com, comname, price);

Листинг 3.9 – Индекс для таблицы ShopOfComponents

Построим план запроса по всем элементам таблицы и посмотрим время его выполнения вместе с индексом (рисунок 3.3).

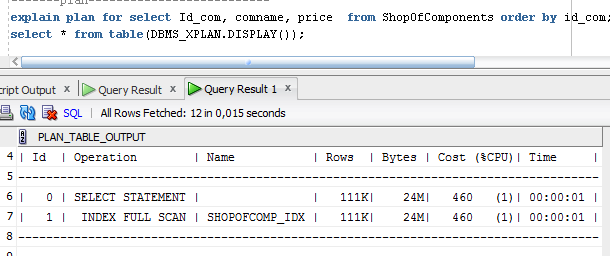


Рисунок 3.3 – План запроса с индексом

Исходя из приведенных выше результатов запросов, можно увидеть, что добавление индекса привело к уменьшению стоимости запроса примерно в 7 раз.

# **4 Описание технологий**

## **4.1 Основная технология 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 автоматизации процесса резервного копирования и восстановления были дополнительно разработаны файлы с расширением .rman и вызывающие их batch-скрипты.

В файле copy.rman (рисунок 4.1) содержатся команды, которые производят резервное копирование управляющего файла и табличных пространств базы данных.

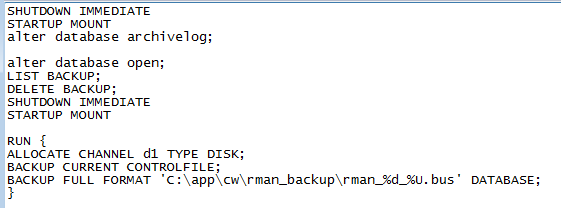


Рисунок 4.1 – Файл copy.rman

Команда List Backup предназначена для вывода списка всех имеющихся копий базы данных. Чтобы удалить старые копии используется команда delete backup. Сами функции копирования размещены в блоке run. Первым делом происходит выделение канала (соединение с сервером), затем происходит резервное копирование текущего управляющего файла, после происходит копирование файлов табличных пространств в указанном формате.

После можно удалить файлы и восстановить базу данных.

Для восстановления базы данных использую файл restore.rman (рисунок 4.2).

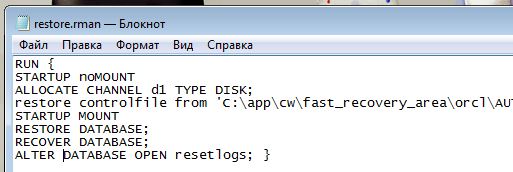


Рисунок 4.2 – Файл restore.rman

При помощи команды restore происходит восстановление файлов, командой recover восстанавливаем базу данных.

## **4.2 Технология шифрования**

В простейшем определении шифрование представляет собой «маскировку» данных, или их преобразование, при котором данные не могут использоваться посторонними. Для шифрования в PL/SQL используется встроенный пакет Oracle DBMS\_CRYPTO.

Была разработана функция encryption\_password. Ее реализацию можно увидеть в листинге 4.1. Эта функция позволяет зашифровать пароль пользователя. Она принимает один входной параметр: пароль пользователя, который следует зашифровать. В моем курсовом проекте для шифрования используется алгоритм AES с 128-разрядным ключом, метод сцепления CBC и механизм заполнения PKCS#5.

Для шифрования необходимо выбрать ключ. В приложении в качестве ключа используется строка «0710196810121972». Это значение относится к типу данных VARCHAR2. Чтобы использовать его в функции ENCRYPT, необходимо сначала преобразовать его к типу RAW. Для этого мы воспользуемся функцией STRING\_TO\_RAW встроенного пакета UTL\_I18N. Также вторым параметром указываем кодировку. Переменная l\_in\_val типа VARCHAR2 также должна быть преобразована к типу RAW. Переменную, которая будет передаваться в качестве возвращаемого значения функции, требуется преобразовать в шестнадцатеричный формат. Для этого используем функцию RAWTOHEX.

CREATE OR REPLACE FUNCTION encryption\_password(emp\_passw IN employee.passw%TYPE) RETURN employee.passw%TYPE

IS

l\_key VARCHAR2(2000) := '0710196810121972';

l\_in\_val VARCHAR2(2000) := emp\_passw;

l\_mod NUMBER := DBMS\_CRYPTO.encrypt\_aes128 + DBMS\_CRYPTO.chain\_cbc + DBMS\_CRYPTO.pad\_pkcs5;

l\_enc RAW(2000);

BEGIN

l\_enc := DBMS\_CRYPTO.encrypt(utl\_i18n.string\_to\_raw(l\_in\_val, 'AL32UTF8'), l\_mod, utl\_i18n.string\_to\_raw(l\_key, 'AL32UTF8'));

RETURN RAWTOHEX(l\_enc);

END encryption\_password;

Листинг 4.1 – Процедура encryption\_password

В обратных целях была разработана функция decryption\_password. Она в качестве параметра принимает зашифрованный пароль и должна в результате своей работе вернуть его в расшифрованном виде. Используя тот же самый ключ, передаем его в качестве одного из параметров в функцию DECRYPT, перед этим преобразовав его в RAW при помощи функции STRING\_TO\_RAW. Так же не забываем передать в функцию параметр typ и зашифрованный пароль типа RAW.

Результат функции преобразуем к строке при помощи функции RAW\_TO\_CHAR пакета UTL\_I18N. Реализацию этой функции можно увидеть в листинге 4.2.

CREATE OR REPLACE FUNCTION decryption\_password(emp\_passw IN employee.passw%TYPE) RETURN employee.passw%TYPE

IS

l\_key VARCHAR2(2000) := '0710196810121972';

l\_in\_val RAW(2000) := HEXTORAW(emp\_passw);

l\_mod NUMBER := DBMS\_CRYPTO.encrypt\_aes128 + DBMS\_CRYPTO.chain\_cbc + DBMS\_CRYPTO.pad\_pkcs5;

l\_dec RAW(2000);

BEGIN

l\_dec := DBMS\_CRYPTO.decrypt(l\_in\_val, l\_mod, utl\_i18n.string\_to\_raw(l\_key, 'AL32UTF8'));

RETURN utl\_i18n.raw\_to\_char(l\_dec);

END decryption\_password;

Листинг 4.2 – Процедура decryption\_password

В дальнейшем эти функции используются в таких процедурах как addEmp, checkEmpAccount, getCurrentEmplIdAndName, addClient, checkClientAccount, getClienIdAndName.

# **5 Руководство пользователя**

## **5.1 Область клиента**

Для начала работы в приложении пользователю требуется зарегистрироваться. Для этого ему требуется корректно заполнить все поля в форме регистрации (рисунок 5.1).

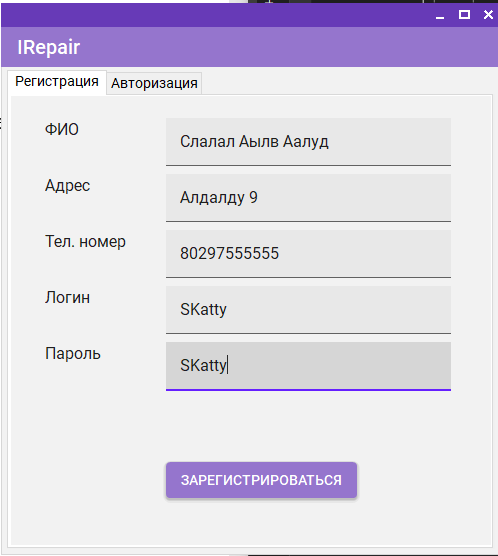


Рисунок 5.1 – Меню регистрации

Далее ему требуется авторизоваться. Для этого ему требуется прейти в меню авторизация и ввести свои логин и пароль (рисунок 5.2).

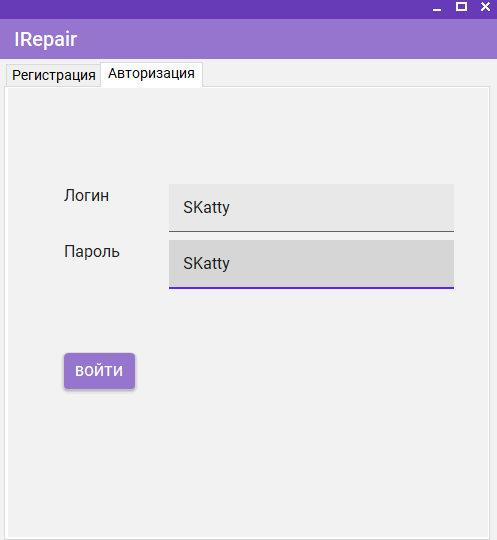


Рисунок 5.2 – Меню авторизации

После авторизации его встретит окно, в котором он сможет сформировать заказ. Для этого ему потребуется заполнить все поля и нажать на кнопку оформить заказ (рисунок 5.3).

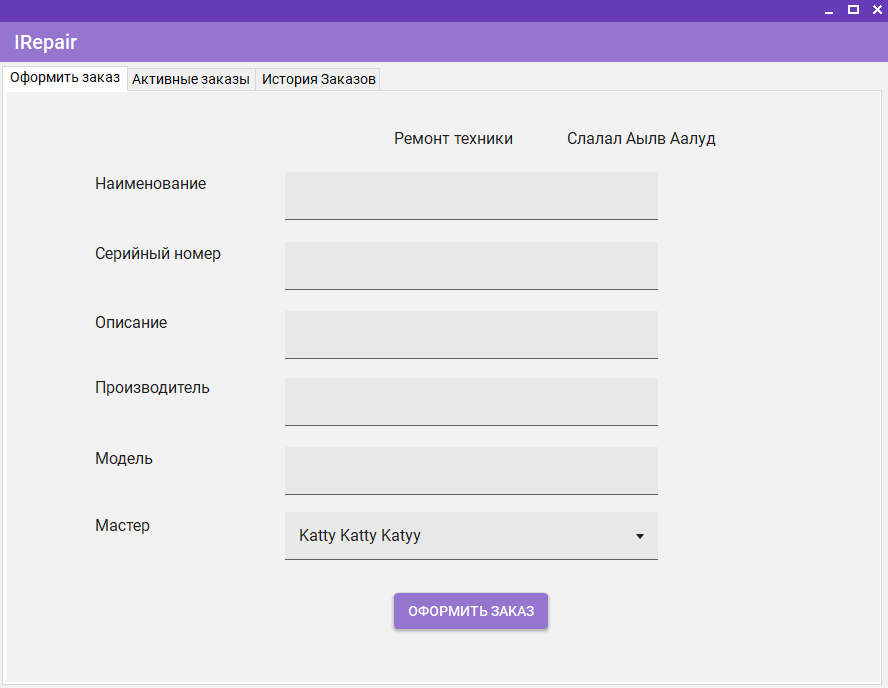


Рисунок 5.3 – Меню оформления заказов

После этого он сможет увидеть свой заказ в меню активных заказов (рисунок 5.4).

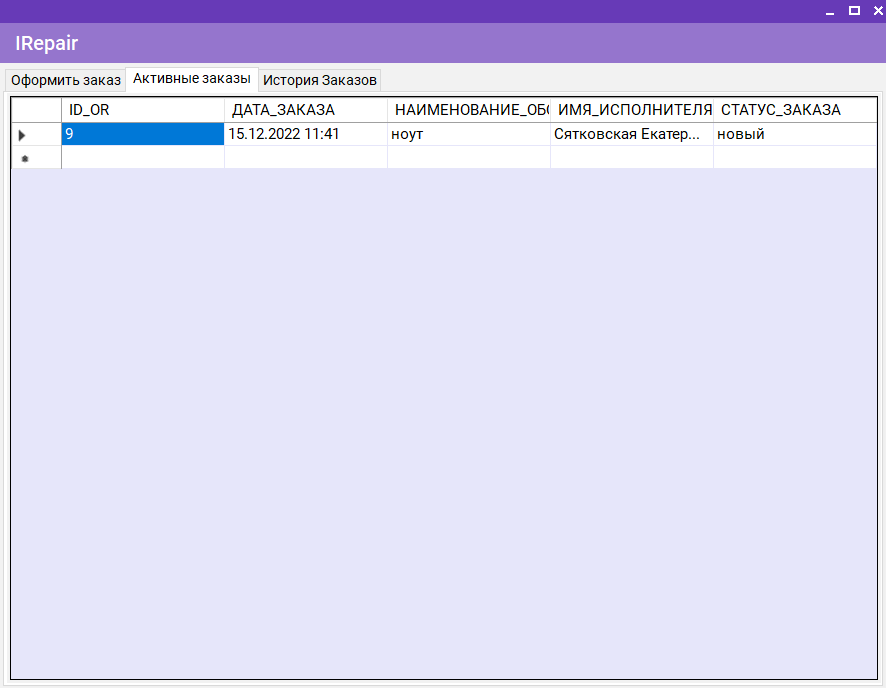


Рисунок 5.4 – Меню просмотра активных заказов

После того как работник обработает заказ клиента, клиент сможет увидеть свой заказ в истории заказов (рисунок 5.5).

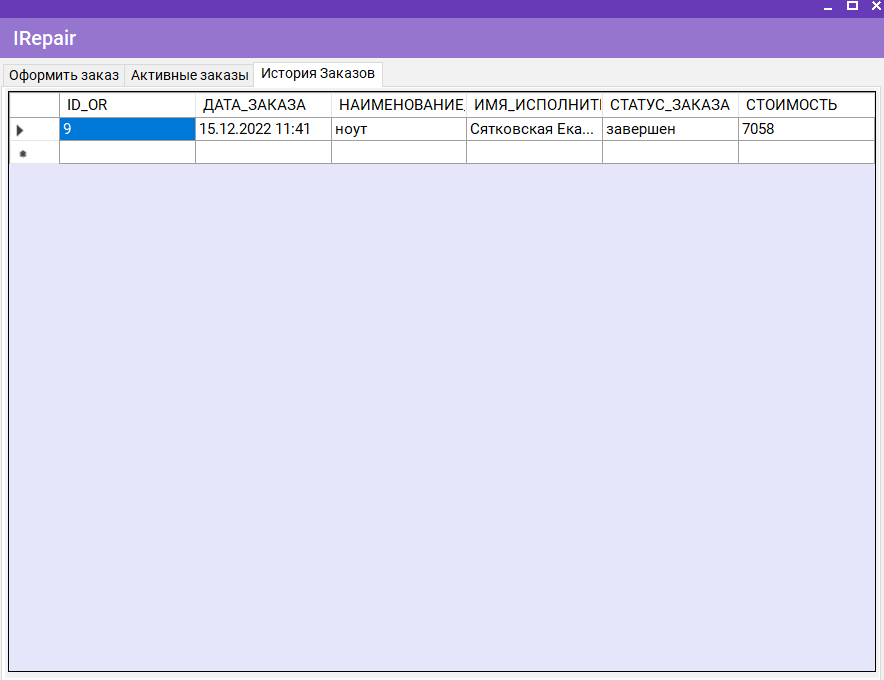


Рисунок 5.5 – Меню просмотра истории заказов

Окно истории заказов схоже с окном активных заказов. Различие заключается в статусе заказа и некоторой информацией, доступной лишь в истории заказов.

## **5.2 Область сотрудника**

Сотрудник имеет похожее окно авторизации, как и у клиента (рисунок 5.6).

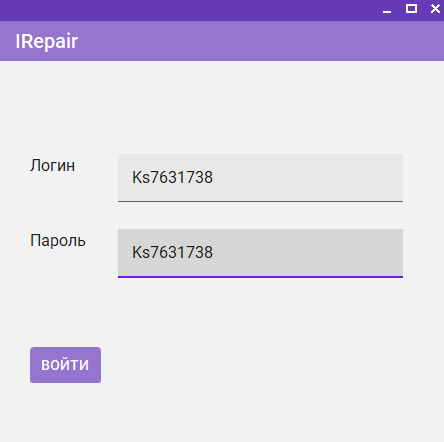


Рисунок 5.6 – Меню авторизации

После авторизации сотруднику будет доступно 3 окна, одно из них 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 5.7).

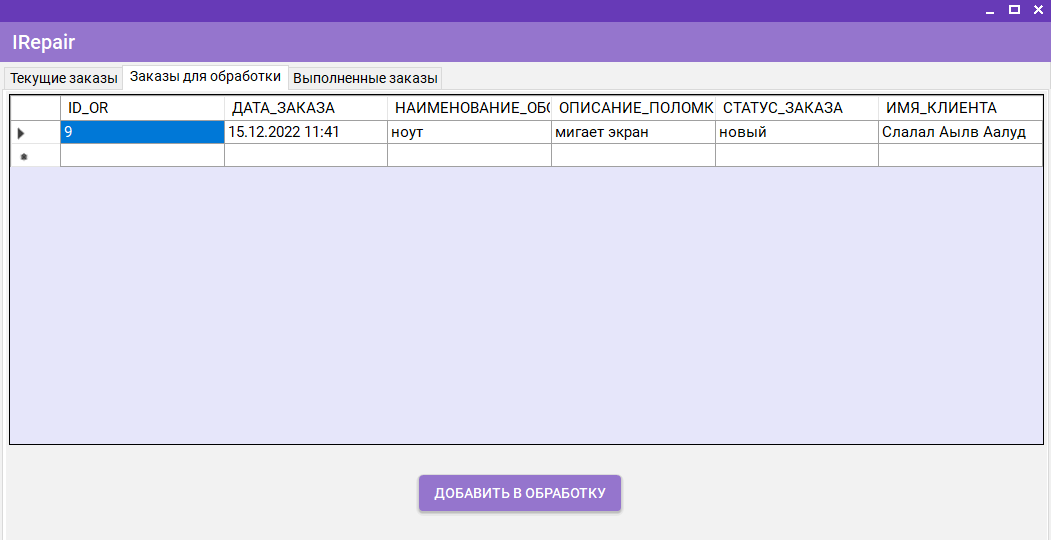


Рисунок 5.7 – Меню просмотра заказов для обработки

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 5.8).

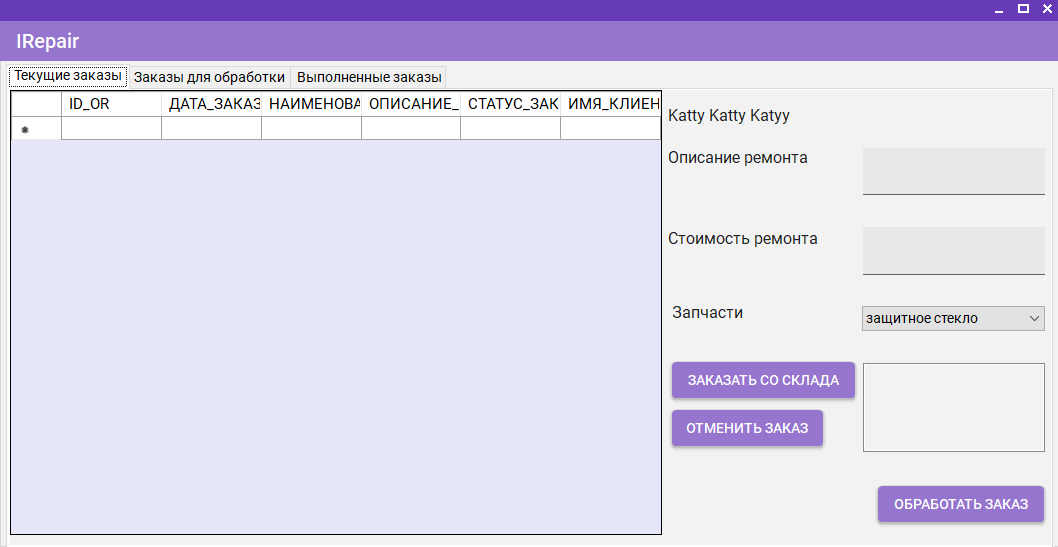


Рисунок 5.8 – Меню просмотра текущих заказов

После этого заказ поменяет статус и перейдёт в историю заказов клиента и работника (рисунок 5.9).

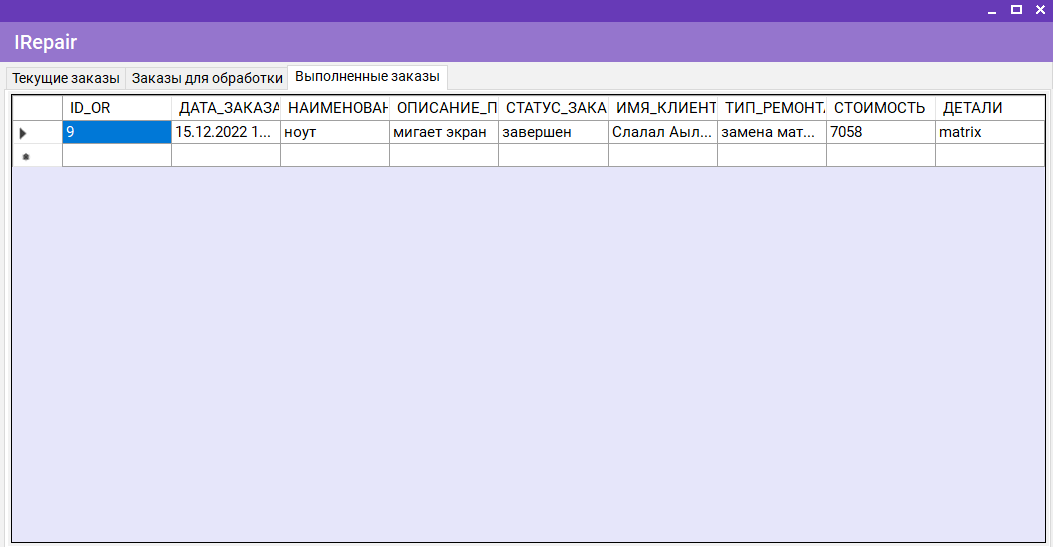


Рисунок 5.9 – Меню просмотра истории заказов

Также статус заказа меняется и при добавлении заказа в обработку.

## **5.3 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 5.10).

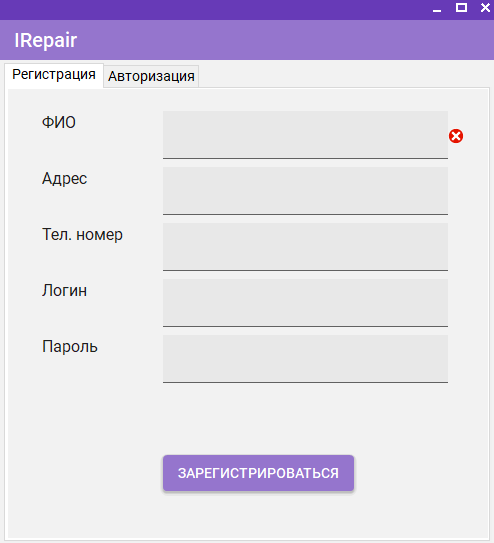


Рисунок 5.10 – Меню регистрации

При авторизации также происходит проверка полей на пустоту. При отсутствии зарегистрированного аккаунта у пользователя не получится авторизоваться (рисунок 5.11).

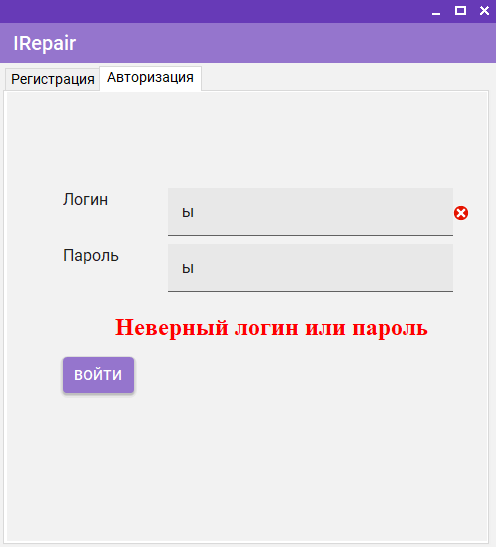


Рисунок 5.11 – Меню авторизации

После авторизации клиент может оформить заказ. При этом у него это получится в том случае, если он заполнит все поля (рисунок 5.12). В случае попытки осуществить заказ с незаполненными полями у него появятся предупреждения.

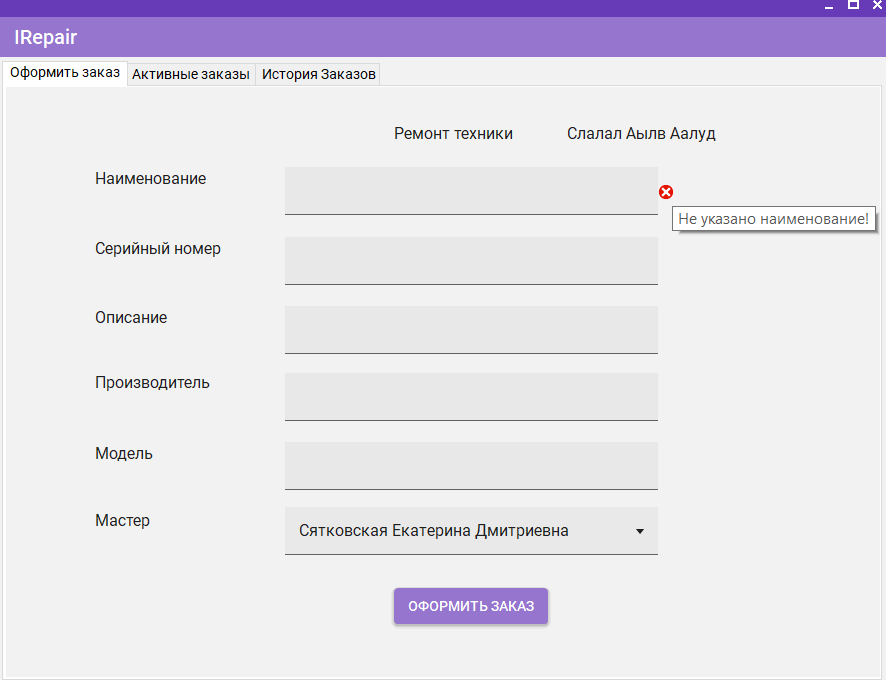


Рисунок 5.12 – Меню оформления заказов

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

## **5.4 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 базе данных (рисунок 5.13).

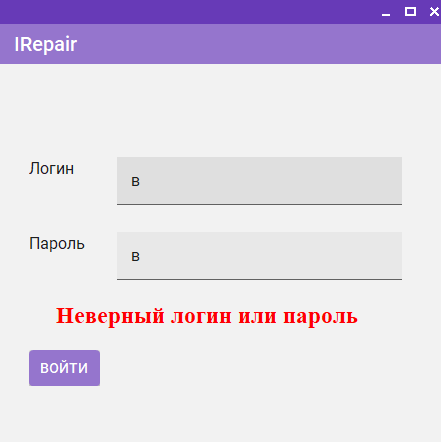


Рисунок 5.13 – Меню авторизации работника

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 5.14).

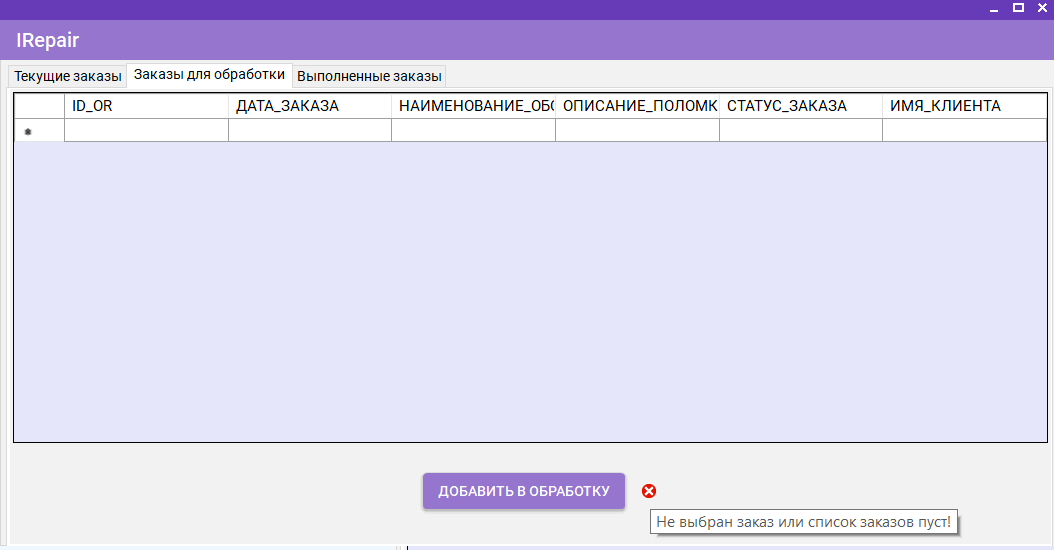


Рисунок 5.14 – Меню обработки заказов

В текущих заказах также стоит проверка на пустые поля и наличие заказов (рисунок 5.15).

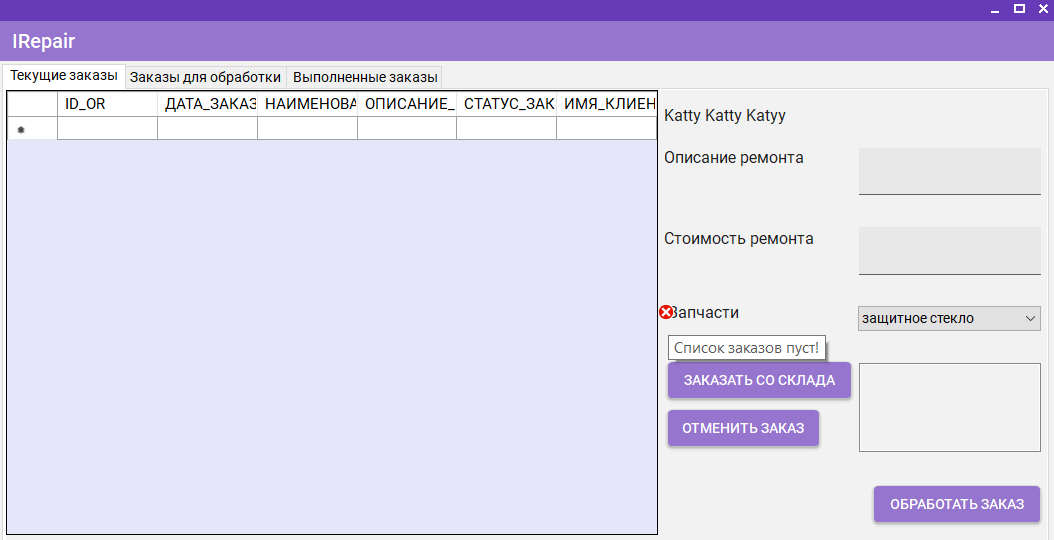


Рисунок 5.15 – Меню текущих заказов

Это были единственные места, где работник мог совершить ошибку.

# **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.

Приложение прошло тестирование при использовании в БД большого количество данных.

Реализация базы данных будет хорошим решением для любого сервиса данного направления, поскольку позволяет охватить необходимый минимум требований для администрирования и использования.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объёме.

# **Cписок используемых 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**

create or replace package body cwClient

as

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

as

sl number;

sp number;

begin

cwclient.getcountofclientwithsamelogin(Login,sl);

cwclient.getcountofclientwithsamephone(PhoneNumber,sp);

if (sl=0 and sp =0) then

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

commit;

else raise\_application\_error(-20000,'account already exist');

end if;

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 getCountOfClientWithSamePhone(checkphone nvarchar2, results out number)

as

begin

results:=0;

select count(\*) into results from Client where phonenumber=checkphone;

exception

when others then

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

end getCountOfClientWithSamephone;

procedure checkClientAccount(lg nvarchar2, ps nvarchar2, results out

number)

as

begin

results:=0;

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

select FullName into fio from Client where Login=lgin and decryption\_password(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;

end cwClient;

# **Приложение Б**

create or replace package body cwEmpl

as

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

as

begin

results:=0;

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

select FullName into fioe from Employee where Login=lgine and decryption\_password(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 order by id\_com;

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 = eplid

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 cwEmpl;

# **Приложение В**

create or replace package body cwAdmin

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

vac\_count number;

begin

select count(\*) into vac\_count from vacancy where vacname =vacancy.vacancyname;

if (vac\_count =0) then

insert into Vacancy(vacancyname) values(vacName);

commit;

else raise\_application\_error(-20000, 'vacancy exist');

end if;

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, PSeria nvarchar2, PNumber nvarchar2, Adress nvarchar2, PhoneN nvarchar2, StartWorkDate

date, Logi nvarchar2, Passw nvarchar2)

as

empl\_count number;

begin

select count(\*) into empl\_count from employee where PNumber = employee.passportnumber and Logi = employee.login and PhoneN= employee.phonenumber;

if (empl\_count =0) then

insert into Employee(fullname, id\_vac, passportseria, passportnumber, adress, phonenumber, startworkdate, login, passw) values(fullName, Id\_vac, PSeria, PNumber, Adress, PhoneN, StartWorkDate, Logi, encryption\_password(Passw));

commit;

else raise\_application\_error(-20000, 'account already exist');

end if;

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 importXmlDataFromComponents

is

begin

insert into shopofcomponents (comname, price)

select ExtractValue(value(components), '//NAME') as comname,

ExtractValue(value(components), '//PRICE') as price

from table(XMLSequence(extract(xmltype(bfilename('DIR', 'components\_import.xml'),

nls\_charset\_id('UTF-8')),'/ROWSET/ROW'))) components;

exception

when others then

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

end importXmlDataFromComponents;

procedure exportXmlToClients

is

rc sys\_refcursor;

doc DBMS\_XMLDOM.DOMDocument;

begin

open rc for

select id\_client,fullname,adress,phonenumber,login,passw from client;

doc := DBMS\_XMLDOM.NewDOMDocument(xmltype(rc));

DBMS\_XMLDOM.WRITETOFILE(doc, 'DIR/client\_export.xml');

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

comp number;

begin

select count(\*) into comp from shopofcomponents where componname= shopofcomponents.comname;

if ( comp =0) then

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

commit;

else raise\_application\_error(-20000, 'component already exsist');

end if;

exception

when others then

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

end addComponents;

procedure insert\_100k\_components

is

begin

for i in 1 .. 100000

loop

cwAdmin.addComponents(i, i);

end loop;

commit;

exception

when others then

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

end insert\_100k\_components;

end cwAdmin;