МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные систем и технологии

Специальность 1-40 01 01 «Программное обеспечение информационных технологий»

Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Разработка базы данных для магазина средств связи с использованием мультимедийных типов данных

Выполнил студент Петух Никита Сергеевич

(Ф.И.О.)

Руководитель проекта преп.-стаж. 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: преп.-стаж. Нистюк О.А.

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

Нормоконтролер: преп.-стаж. Нистюк О.А.

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

Курсовой проект защищен с оценкой

Минск 2019

Оглавление

[Введение 3](#_Toc27550809)

[1 Аналитический обзор литературы 4](#_Toc27550810)

[2 Постановка задачи 5](#_Toc27550811)

[3 Разработка модели базы данных 6](#_Toc27550812)

[4 Разработка объектов базы данных 7](#_Toc27550813)

[4.1 Таблицы 7](#_Toc27550814)

[4.2 Пользователи 9](#_Toc27550815)

[4.3 Процедуры 9](#_Toc27550816)

[4.4 Последовательности 11](#_Toc27550817)

[4.5 Триггеры 12](#_Toc27550818)

[5 Описание процедур импорта и экспорта данных 13](#_Toc27550819)

[5.1 Описание процедуры импорта 13](#_Toc27550820)

[5.2 Описание процедуры экспорта 13](#_Toc27550821)

[6 Тестирование производительности 15](#_Toc27550822)

[7 Описание используемой технологии 16](#_Toc27550823)

[8 Руководство пользователя 18](#_Toc27550824)

[Заключение 23](#_Toc27550825)

[Список используемых источников 24](#_Toc27550826)

[Приложение А 25](#_Toc27550827)

[Приложение Б 28](#_Toc27550828)

[Приложение В 32](#_Toc27550829)

[Приложение Г 43](#_Toc27550830)

# Введение

Программное средство «4You» – это приложение позволяющее приобретать мобильные устройства, просматривать их подробную информацию, выбирать наиболее подходящее устройство, путем сравнения товаров, оставлять комментарии с рейтингом об устройстве, а также, добавлять устройства в избранные товары. На сегодняшний день большинство людей все чаще и чаще приобретают товары в онлайн-магазинах, не выходя из дома. Основное отличие Интернет-магазина от традиционного в типе торговой площадки. Если обычному магазину нужен торговый зал, витрины, ценники, а также продавцы, кассиры и опытные консультанты, то у его онлайн-магазинов вся инфраструктура реализована программно. Функции витрины и торгового зала выполняют страницы с иллюстрированным каталогом товаров, а консультантов подсказки, инструкции и описания. Все остальное как в обычном магазине. Даже в интерфейсе Интернет-магазина сохраняются привычные элементы, например виртуальная тележка (корзина), куда мы по пути к кассе складываем выбранные товары. Ассортимент онлайн-магазинов во много раз шире, т.к. вас не ограничивает рамки одного помещения, а также вы в самый кротчайший срок подберете себе товар, который вы искали.

Целью данного курсового проекта является создание базы данных для приложения «4You». Конечным пользователем программы может быть любой человек. Интерфейс программы должен быть понятен. Основная функция программы должна заключается в том, чтобы пользователь мог легко просматривать, находить и сохранять необходимые материалы.

Для разработки базы данных использовалась СУБД Oracle 12c, а в качестве интерфейса прикладного программирования был выбран API-интерфейс — Windows Presentation Foundation (WPF), предназначенный для создания настольных программ с графически насыщенным пользовательским интерфейсом. Для работы с WPF использовался объектно-ориентированный язык программирования — С#. Связь между базой данных и приложением на С# осуществляется через Entity Framework. Эта предоставляет собой технологию работы с данными, которая основана на платформе .NET Framework.

# Аналитический обзор литературы

В качестве основы для разработки собственного программного средства рассмотрим уже существующие.

Первым рассмотрим один из самых популярных онлайн-магазинов «Onliner.by» В Беларуси. Данный сайт позволяет:

* просматривать каталог товаров различных товаров;
* просматривать подробную информацию о товаре;
* заходить на сайт в качестве зарегистрированного пользователя;
* оставлять комментарии и рейтинг о товаре и покупатели;
* просматривать информацию о продавцах;
* осуществлять поиск товара на сайте;
* добавлять товар в корзину;
* сравнивать товары.

На рисунке 1.1 представлен интерфейс данного сайта.

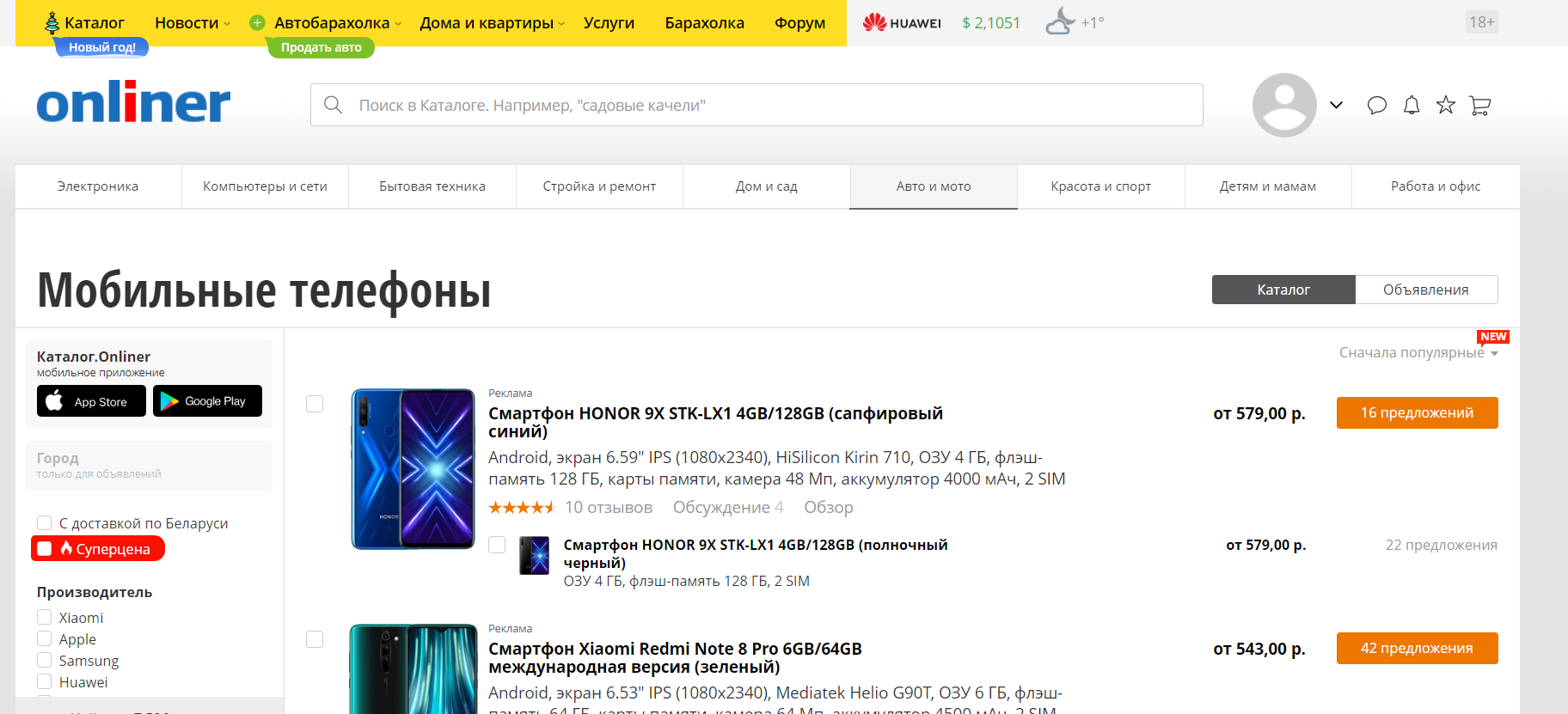


Рисунок 1.1 – Интерфейс веб-сайта «Onliner.by»

Из рисунка 1.1 видно, что интерфейс данного сервиса прост и интуитивно понятен пользователю, что является большим преимуществом, для такого рода приложений. В верхней части сайта расположены основные средства взаимодействия, а именно: строка поиска, кнопки для входа и регистрации пользователей, корзина. По центру располагается каталог товаров, слева фильтры для более быстрого выбора товара.

# Постановка задачи

В соответствии с требованиями к курсовому проекту необходимо:

* Спроектировать базу данных в соответствии с выбранной предметной областью, а именно базу данных для магазина мобильных устройств;
* Создать объекты базы данных (таблицы, представления, индексы, пользователи и пр.) в количестве, необходимом для покрытия нужд предметной области;
* Обеспечить необходимый уровень безопасности и изолированности пользователей;
* Создать процедуры для экспорта и импорта данных в XML;
* Применить технологию базы данных согласно выбранной теме: подробно описать применяемые системные пакеты, утилиты или технологии; показать применение указанной технологии в базе данных.
* Создать приложение, позволяющее протестировать все функции базы данных и её объектов.

1. **Разработка модели базы данных**

Для базы данных было разработано 12 таблиц. Структура связей базы данных представлена на рисунке 3.1. База данных была разработана в СУБД Oracle 12c.

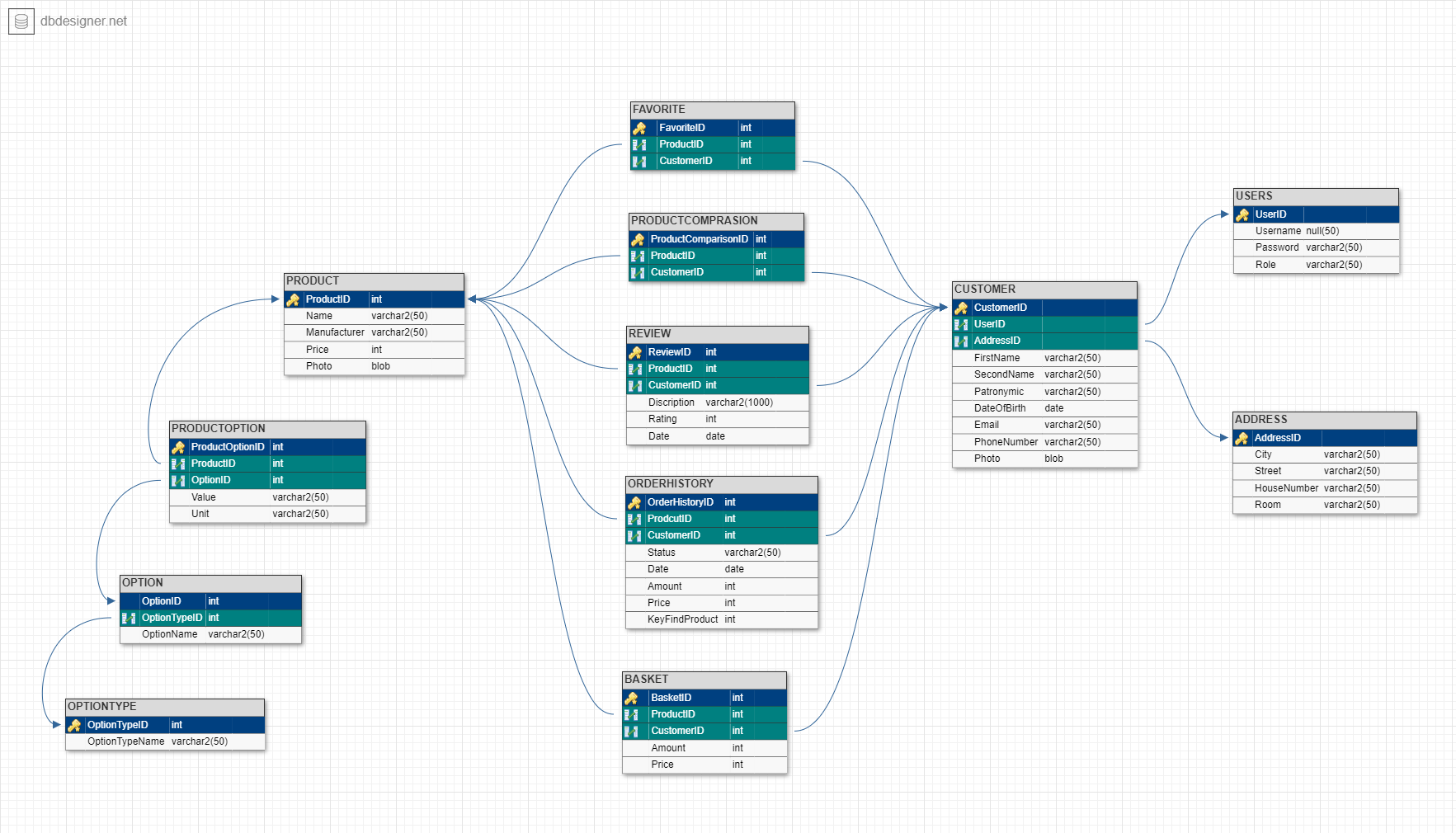


Рисунок 3.1 – Диаграмма модели базы данных

Подробнее, структура базы данных и её объекты, будут рассмотрены в следующей главе.

# Разработка объектов базы данных

База данных данного курсового проекта содержит следующие объекты: таблицы, хранимые процедуры, пользователи, представления, последовательности триггеры. Их подробное описание представлено ниже.

## Таблицы

Разработанная база данных содержит 12 таблиц, описание которых представлено ниже.

Таблица USERS предназначена для хранения информации о пользователях, и имеет следующие поля:

* UserID – идентификатор пользователя;
* Username – имя пользователя;
* Password – пароль пользователя;
* Role – роль пользователя (Admin, User).

Таблица CUSTOMER предназначена для хранения информации о личной информации клиента, и имеет следующие поля:

* CustomerID – идентификатор клиента;
* UserID – идентификатор пользователя, находящийся в таблице USERS;
* AddressID – идентификатор адреса клиента, находящийся в таблице ADDRESS;
* FirstName – имя клиента;
* LastName – фамилия клиента;
* Patronymic – отчество клиента;
* DateOfBirth – дата рождения клиента;
* Email – электронная почта клиента;
* PhoneNumber – номер телефона клиента;
* Photo – фотография клиента.

Таблица ADDRESS предназначена для хранения информации о адресе клиента, и имеет следующие поля:

* AddressID – идентификатор адреса клиента;
* City – город проживания клиента;
* Street – улица клиента;
* HouseNumber – номер дома клиента;
* Room – квартира клиента.

Таблица PRODUCT предназначена для хранения информации о устройствах связи, и имеет следующие поля:

* ProductID – идентификатор товара;
* Name – наименование товара;
* Manufacturer – бренд товара;
* Price – цена товара;
* Photo – фотография товара.

Таблица PRODUCTOPTION предназначена для хранения информации о характеристики устройствах связи, и имеет следующие поля:

* ProductOptionID – идентификатор характеристики товара;
* ProductID – идентификатор товара, находящийся в таблице PRODUCT;
* OptionID – параметры товара, находящийся в таблице OPTION;
* Value – значение характеристики товара;
* Unit – единица измерения;

Таблица OPTION предназначена для хранения информации о параметрах характеристики устройствах связи, и имеет следующие поля:

* OptionID – идентификатор параметра;
* OptionTypeID – идентификатор типа параметра, находящийся в таблице OPTIONTYPE;
* OptionName – наименование параметра.

Таблица OPTIONTYPE предназначена для хранения информации о характеристики устройствах связи, и имеет следующие поля:

* OptionTypeID – идентификатор типа параметра;
* OptionTypeName – наименование типа параметра.

Таблица BASKET предназначена для хранения информации о товарах добавленные в корзину клиентом, и имеет следующие поля:

* BasketID – идентификатор корзины;
* ProductID – идентификатор товара, находящийся в таблице PRODUCT;
* CustomerID – идентификатор клиента, находящийся в таблице CUSTOMER;
* Amount – количество товара в корзине;
* Price – общая цена корзины.

Таблица ORDERHISTORY предназначена для хранения информации о всех покупках клиентов, и имеет следующие поля:

* OrderHistoryID – идентификатор истории клиентов;
* ProductID – идентификатор товара, находящийся в таблице PRODUCT;
* CustomerID – идентификатор клиента, находящийся в таблице CUSTOMER;
* Status – статус товара (в корзине, оплачен);
* Date – дата покупки товара;
* Amount – количество купленного товара клиентом;
* Price – общая цена купленных товаров клиентом;
* KeyFindProduct – идентификатор корзины.

Таблица FAVORITE предназначена для хранения информации о избранных товаров клиента, и имеет следующие поля:

* FavoriteID – идентификатор избранных товаров клиента;
* ProductID – идентификатор товара, находящийся в таблице PRODUCT;
* CustomerID – идентификатор клиента, находящийся в таблице CUSTOMER.

Таблица REVIEW предназначена для хранения информации о отзыве клиента на товар, и имеет следующие поля:

* ReviewID – идентификатор отзыва клиента;
* ProductID – идентификатор товара, находящийся в таблице PRODUCT;
* CustomerID – идентификатор клиента, находящийся в таблице CUSTOMER;
* Description – описание отзыва;
* Rating – рейтинг товара;
* Date – дата отзыва.

Таблица PRODUCTCOMPRASION предназначена для хранения информации о сравнении товаров и их характеристик, который выбрал клиент, и имеет следующие поля:

* ProductComprasionID – идентификатор избранных товаров клиента;
* ProductID – идентификатор товара, находящийся в таблице PRODUCT;
* CustomerID – идентификатор клиента, находящийся в таблице CUSTOMER.

Скрипты создания таблиц приведены в приложении А.

## Пользователи

Пользователь базы данных – это физическое или юридическое лицо, которое имеет доступ к базе данных и пользуется услугами информационной системы, для получения информации. При проектировании базы данных, понадобилось создать 2 пользователей: APPUSER, APPADMIN, APPCLIENT, APPANONIM. У каждого пользователя есть свои права, предназначенные для выполнения соответствующих ему требований.

Пользователь APPUSER наделен правами администратора. Он является владельцем всех объектов базы данных и имеет доступ ко всем процедурам, включая процедуры экспорта и импорта.

Пользователь APPADMIN имеет только доступ к процедурам входа, добавления, изменения, удаления и отображения данных.

Пользователь APPCLIENT имеет право просматривать информацию о себе и своих сохраненных материалах, а также добавлять материалы в избранные.

Пользователь APPANONIM имеет право просматривать информацию товарах.

Скрипты создания пользователей приведены в приложении Б.

## Процедуры

Использование хранимых процедур в курсовом проекте предназначено для ограничения доступа пользователей к таблицам базы данных, оставив пользователям только разрешения на выполнение некоторых процедур, обеспечивающих косвенный и строго регламентированный доступ к данным.

Пользователь APPADMIN имеет доступ к следующим процедурам:

* ADMINADDCUSTOMER (p\_username IN, p\_password IN, p\_role IN, p\_city IN, p\_street IN, p\_housenumber IN, p\_room IN, p\_firstname IN, p\_secondname IN, p\_patronymic IN, p\_dateofbirth IN, p\_email IN, p\_phonenumber IN) – процедура, предназначенная для добавления строки в таблицу CUSTOMER;
* ADMINADDOPTION (p\_productid IN, p\_optionid IN, p\_value IN, p\_unit IN) – процедура, предназначенная для добавления строки в таблицу PRODUCTOPTION;
* ADMINADDPRODUCT (p\_name IN, p\_manuf IN, p\_price IN) – процедура, предназначенная для добавления строки в таблицу PRODUCT;
* ADMINDELETECUSTOMER (p\_userid IN, p\_addressid IN, p\_customerid IN) – процедура, предназначенная для удаления строки в таблицу CUSTOMER;
* ADMINDELETEOPTION (p\_productoptionid IN) – процедура, предназначенная для добавления строки в таблицу PRODUCTOPTION;
* ADMINDELETEPRODUCT (p\_productid IN) – процедура, предназначенная для добавления строки в таблицу PRODUCT;
* ADMINUPDATECUSTOMER (p\_username IN, p\_password IN, p\_role IN, p\_city IN, p\_street IN, p\_housenumber IN, p\_room IN, p\_firstname IN, p\_secondname IN, p\_patronymic IN, p\_dateofbirth IN, p\_email IN, p\_phonenumber IN) – процедура, предназначенная для обновления строки в таблицу CUSTOMER;
* ADMINUPDATEOPTION (p\_productid IN, p\_optionid IN, p\_value IN, p\_unit IN) – процедура, предназначенная для обновления строки в таблицу PRODUCTOPTION;
* ADMINUPDATEPRODUCT (p\_productid IN) – процедура, предназначенная для обновления строки в таблицу PRODUCT;
* LOGIN (p\_username IN, p\_password IN, p\_userid OUT, p\_addressid OUT, p\_customerid OUT, p\_role OUT) – процедура, предназначенная для входа пользователя в приложение;
* REGISTEADMIN (p\_username IN, p\_password IN) – процедура, предназначенная для регистрации пользователя ADMIN в приложение;
* REGISTERUSER (p\_username IN, p\_password IN) – процедура, предназначенная для регистрации пользователя CLIENT в приложение;
* TABLE\_TO\_XML\_FILE (table\_name IN) – процедура, предназначенная для экспорта данных;
* SHOWCATALOG (p\_productid OUT, p\_name OUT, p\_manufacturer OUT, p\_price OUT) - процедура, предназначенная для просмотра каталога товаров.

Пользователь APPCLIENT имеет доступ к следующим процедурам:

* BASKETDELETEONE (p\_basketid IN, p\_customerid IN) – процедура, предназначенная для удаления строки в таблице BASKET;
* LOGIN (p\_username IN, p\_password IN, p\_userid OUT, p\_addressid OUT, p\_customerid OUT, p\_role OUT) – процедура, предназначенная для входа пользователя в приложение;
* PERSONAINFOUPDATE (p\_userid IN, p\_username IN, p\_password IN, p\_addressid IN, p\_city IN, p\_street IN, p\_housenumber IN, p\_room IN, p\_firstname IN, p\_secondname IN, p\_patronymic IN, p\_dateofbirth IN, p\_email IN, p\_phonenumber IN) – процедура, предназначенная на изменения персональных данных в таблице CUSTOMER и ADDRESS;
* PERSONALINFOSELECT (p\_userid IN, p\_username OUT, p\_password OUT, p\_addressid OUT, p\_city OUT, p\_street OUT, p\_housenumber OUT, p\_room OUT, p\_firstname IN, p\_secondname OUT, p\_patronymic OUT, p\_dateofbirth OUT, p\_email OUT, p\_phonenumber OUT) – процедура, предназначенная на просмотра персональных данных в таблице CUSTOMER и ADDRESS;
* REGISTERUSER (p\_username IN, p\_password IN) – процедура, предназначенная для регистрации пользователя CLIENT в приложение;;
* SHOWBASKET (p\_customerid IN) - процедура, предназначенная для просмотра корзины;
* SHOWCATALOG (p\_productid OUT, p\_name OUT, p\_manufacturer OUT, p\_price OUT) - процедура, предназначенная для просмотра каталога товаров;
* SHOWFAVORITE (p\_customerid IN) - процедура, предназначенная для просмотра избранных товаров;
* SHOWHISTORY (p\_customerid IN) - процедура, предназначенная для просмотра истории покупок;
* SHOWOPTIONPRODUCT (p\_productid IN) - процедура, предназначенная для просмотра характеристик товара;
* SHOWREVIEW (p\_productid IN) - процедура, предназначенная для просмотра отзывов и рейтинга о товаре.

Пользователь APPCLIENT имеет доступ к следующим процедурам:

* + SHOWCATALOG (p\_productid OUT, p\_name OUT, p\_manufacturer OUT, p\_price OUT) - процедура, предназначенная для просмотра каталога товаров;
  + SHOWOPTIONPRODUCT (p\_productid IN) - процедура, предназначенная для просмотра характеристик товара;
  + SHOWREVIEW (p\_productid IN) - процедура, предназначенная для просмотра отзывов и рейтинга о товаре.

Скрипты создания пользователей приведены в приложении В.

## Последовательности

**Последовательность** – это объект базы данных, который генерирует целые числа в соответствии с правилами, установленными во время его создания. Для последовательности можно указывать как положительные, так и отрицательные целые числа. В системах баз данных последовательности применяют для самых разных целей, но в основном для автоматической генерации первичных ключей. Тем не менее к первичному ключу таблицы последовательность никак не привязана, так что в некотором смысле она является еще и объектом коллективного пользования. Если первичный ключ нужен лишь для обеспечения уникальности, а не для того, чтобы нести определенный смысл, последовательность является отличным средством.

Для курсовой работы были созданы последовательности для автоматической генерации первичных ключей:

* SQ\_ADDRESS – автоматическая генерация первичного ключа для таблицы ADDRESS;
* SQ\_BASKET – автоматическая генерация первичного ключа для таблицы BASKET;
* SQ\_CUSTOMER – автоматическая генерация первичного ключа для таблицы CUSTOMER;
* SQ\_FAVORITE – автоматическая генерация первичного ключа для таблицы FAVORITE;
* SQ\_Options – автоматическая генерация первичного ключа для таблицы OPTION;
* SQ\_OptionTypes – автоматическая генерация первичного ключа для таблицы OPTIONTYPE;
* SQ\_ORDERHISTORY – автоматическая генерация первичного ключа для таблицы ORDERHISTORY;
* SQ\_PRODUCTCOMPARISON – автоматическая генерация первичного ключа для таблицы PRODUCTCOMPARISON;
* SQ\_ProductOptions – автоматическая генерация первичного ключа для таблицы PRODUCTOPTION;
* SQ\_Products – автоматическая генерация первичного ключа для таблицы PRODUCT;
* SQ\_USERS – автоматическая генерация первичного ключа для таблицы USERS.

Скрипты создания последовательностей приведены в приложении Г.

## Триггеры

Триггеры представляют специальный тип хранимой процедуры, которая вызывается автоматически при выполнении определенного действия

над таблицей или представлением, в частности, при добавлении, изменении или удалении данных, то есть при выполнении команд INSERT, UPDATE, DELETE.

В курсовой работе был созданы триггер BEFORE типа INSER. До вставки строки в таблицу, выполняется триггер для получения следующего значения последовательности:

* TR\_ADDRESS – триггер для последовательности SQ\_ ADDRESS;
* TR\_BASKET – триггер для последовательности SQ\_ BASKET;
* TR\_CUSTOMER – триггер для последовательности SQ\_ CUSTOMER;
* TR\_FAVORITE – триггер для последовательности SQ\_ FAVORITE;
* TR\_Options – триггер для последовательности SQ\_ Options;
* TR\_OptionTypes – триггер для последовательности SQ\_ OptionTypes;
* TR\_ORDERHISTORY – триггер для последовательности SQ\_ ORDERHISTORY;
* TR\_PRODUCTCOMPARISON – триггер для последовательности SQ\_ PRODUCTCOMPARISON;
* TR\_ProductOptions – триггер для последовательности SQ\_ ProductOptions;
* TR\_Products – триггер для последовательности SQ\_Products;
* TR\_USERS – триггер для последовательности SQ\_USERS;

Скрипты создания триггеров приведены в приложении Г.

1. **Описание процедур импорта и экспорта данных**

Функции экспорта и импорта данных могут использоваться в качестве способа аварийного восстановления базы данных и/или переноса данных между базами данных, в случае если встроенные средства СУБД по каким-либо причинам невозможно использовать.

## Описание процедуры импорта

В рамках курсовой работы были написаны процедуры для импорта данных из таблиц базы данных из XML-файлов. При этом текст XML-файла разбирается на отдельные элементы, соответствующие атрибутам целевых таблиц, и вставляется функцией Insert.

Процедура импорта показана на рисунке 5.1.

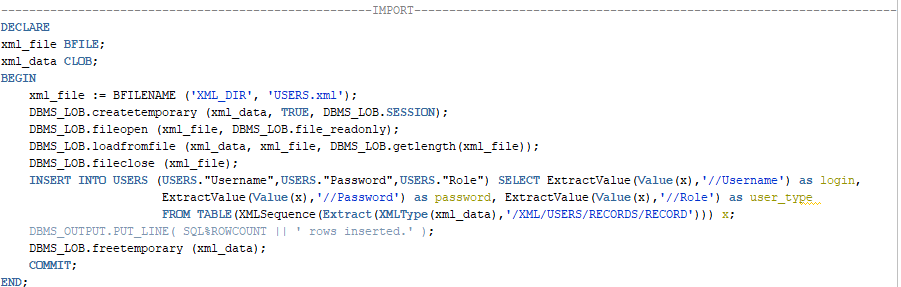


Рисунок 5.1 – Процедура импорта данных

## Описание процедуры экспорта

В рамках курсовой работы была написана процедура для экспорта данных таблиц базы данных в XML-файл. При вызове процедуры TABLE\_TO\_XML\_FILE, параметры которой, название таблицы, которую хотим выгрузить. Вначале создаем новый контекст для выражения SQL c помощью функции NEWCONTEXT, которая находится в пакеты XML\_GEN (используется для преобразования результатов SQL запроса в XML формат), затем устанавливает имя корневого элемента документа c помощью процедуры SETROWSETTAG и устанавливает имя элемента, разделяющего все строки SETROWTAG. После сериализуем SQL-запрос в XML при помощи XMLSERIALIZE. Затем записываем в файл все элементы через цикл с помощью пакета UTL\_FILE (позволяет производить чтения и записи текстовых файлов).

Результат экспорта представлен на рисунке 5.2.



Рисунок 5.1 – Результат экспорта данных

1. **Тестирование производительности**

Для тестирования производительности в таблицу PRODUCT были добавлены 100000 строк. Скрипт для вставки строк приведен на рисунке 6.1.

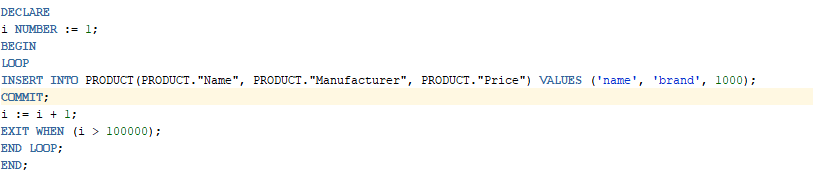


Рисунок 6.1 – Скрипт для вставки строк в таблицу PRODUCT

Для тестирования производительности базы данных выполним различные запросы к таблице и посмотрим план их выполнения.

Выполним простой Select-запрос к таблице. При этом время выполнения запроса составляет 3.591 секунд. План запроса приведен на рисунке 5.2.

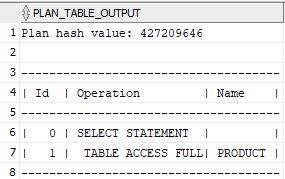


Рисунок 6.2 План запроса

Как мы видим, запрос выполняется за 2 шага, индексы не задействуются.

Выполним Select-запрос к таблице с секцией Where. При этом время выполнения запроса составляет 2.999 секунда. Как мы видим, запрос выполняется за 3 шага, задействуется индекс. План запроса приведен на рисунке 6.3.

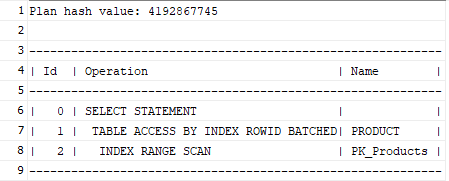


Рисунок 6.3 План запроса

1. **Описание используемой технологии**

Oracle и PL/SQL поддерживают несколько разновидностей типов данных, предназначенных специально для работы с большими объектами (LOB, Large OBjects). Такие объекты позволяют хранить огромные (от 8 до 128 терабайт) объемы двоичных (например, графических, аудио, видео) или текстовых данных.

До выхода Oracle9i Release2 в объектах LOB можно было хранить до 4 Гбайт данных. Начиная с Oracle10g, ограничение было повышено до величины от 8 до 128 терабайт (конкретное значение зависит от размера блока вашей базы данных).

В PL/SQL можно объявлять большие объекты четырех типов:

* BFILE — двоичный файл. Переменная этого типа содержит локатор файла, указывающий на файл операционной системы вне базы данных. Oracle интерпретирует содержимое файла как двоичные данные.
* BLOB — большой двоичный объект. Переменная этого типа содержит локатор LOB, указывающий на большой двоичный объект, хранящийся в базе данных.
* CLOB — большой символьный объект. Переменная этого типа содержит локатор LOB, указывающий на хранящийся в базе данных большой блок текстовых данных в наборе символов базы данных.
* NCLOB — большой символьный объект с поддержкой символов национальных языков (NLS). Переменная этого типа содержит локатор LOB, указывающий на хранящийся в базе данных большой блок текстовых данных с национальным набором символов.

Большие объекты можно разделить на две категории: внутренние и внешние. Внутренние большие объекты (типы BLOB, CLOB и NCLOB) хранятся в базе данных и могут участвовать в транзакциях на сервере базы данных. Внешние большие объекты (тип BFILE) представляют двоичные данные, хранящиеся в файлах операционной системы вне таблиц базы данных. Они не могут участвовать в транзакциях, то есть вносимые в них изменения нельзя сохранить или отменить в зависимости от результата транзакции. Целостность данных обеспечивается только на уровне файловой системы. Кроме того, повторное чтение из BFILE может приводить к разным результатам — в отличие от внутренних больших объектов, соответствующих модели логической целостности чтения.

В данном курсовом проекта мультимедийные типы данных использовались в таблице CUSTOMER (фотографии клиентов), PRODUCT (фотографии товаров). Для хранения изображения использовался тип BLOB. Установить фото можно как в самом Oracle Developer, так и в нашем приложении.

Чтобы установить фото в Oracle Developer, нужно вначале открыть таблицу в которой находится тип BLOB, щелкнуть 2 раза по полю, откроется окно представленное на рисунке 6.1. Здесь можно как просмотреть фото в виде массива байт, если выберем «Text», а также просмотреть фото, если выберем «Image», можем сохранить фото, если нажмем на «Download», можем загрузить фото, если нажмем «Load», можем поставить полю значение NULL, если нажмем «Set NULL», а также просмотреть информацию о фото.

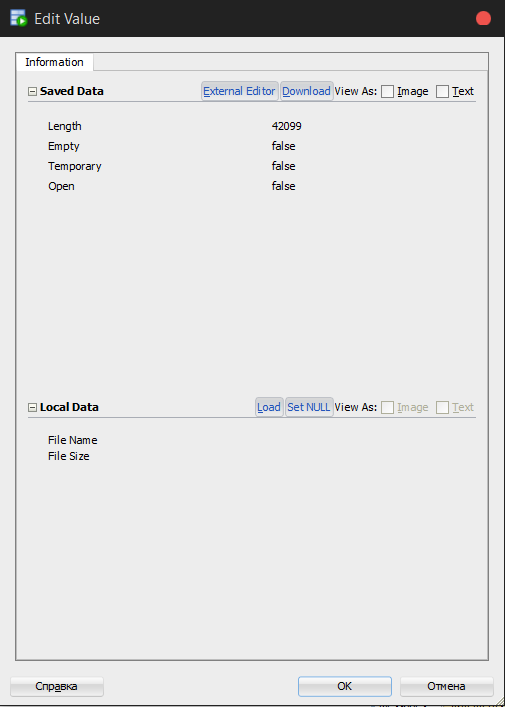


Рисунок 6.1 – окно для работы с BLOB

В приложении пользователь выбирает файл, который хочет загрузить, преобразовывает его в массив байт, который после добавляется в базу данных. Функция, загружающая файл из файловой системы и преобразующая его в пригодный для вставки в базу данных формат изображена на рисунке 6.2.

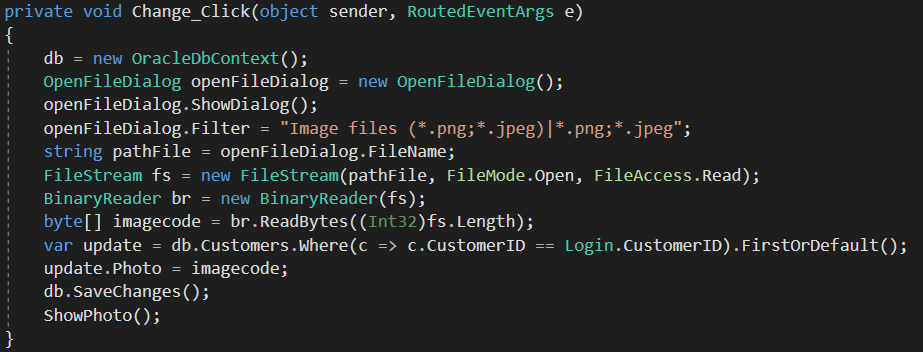


Рисунок 6.2 – функция, которая добавляет фото в БД

# Руководство пользователя

При запуске приложения пользователь является анонимным, и может только просматривать каталог товаров, характеристики и комментарии других пользователей. Интерфейс приложения показан на рисунке 8.1 – 8.2.

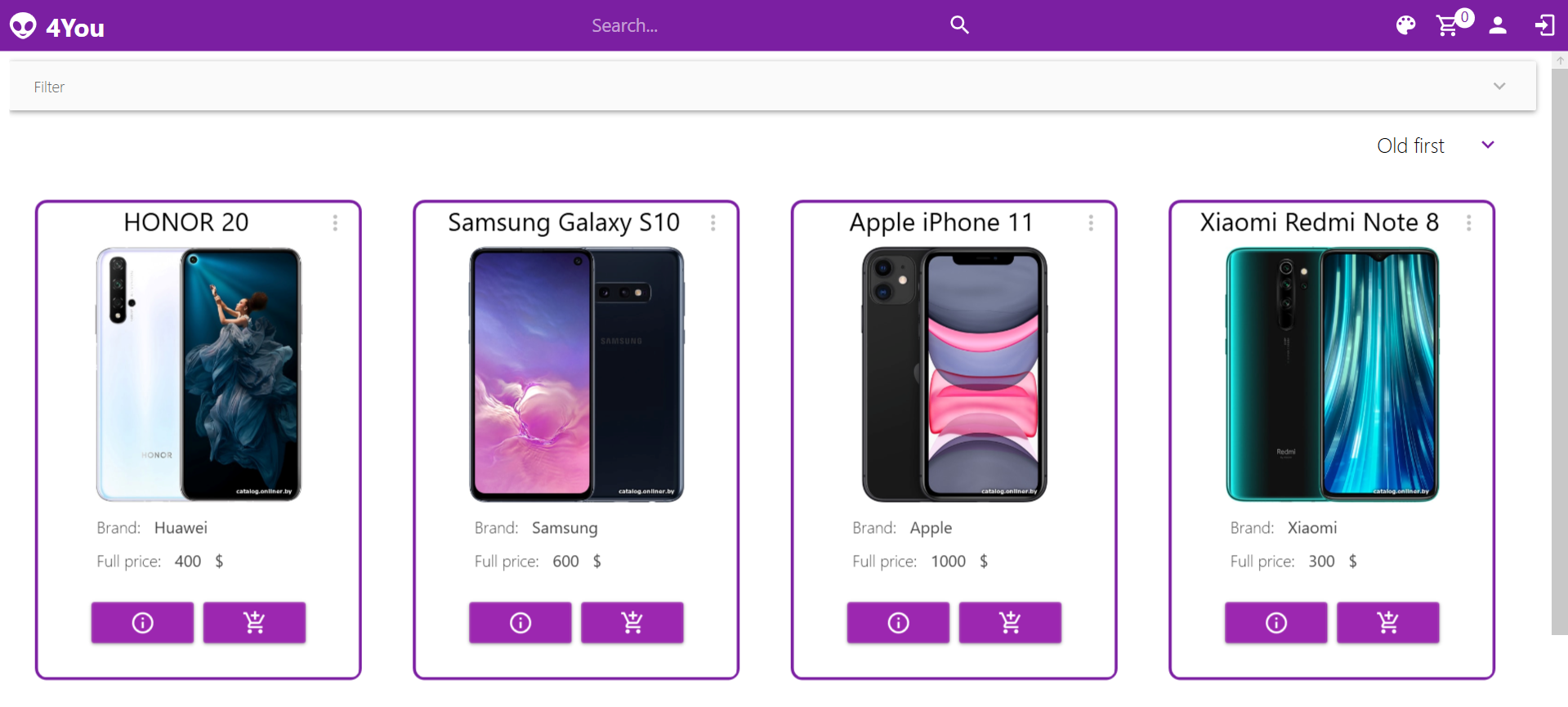


Рисунок 8.1 – Интерфейс приложения для анонимного пользователя

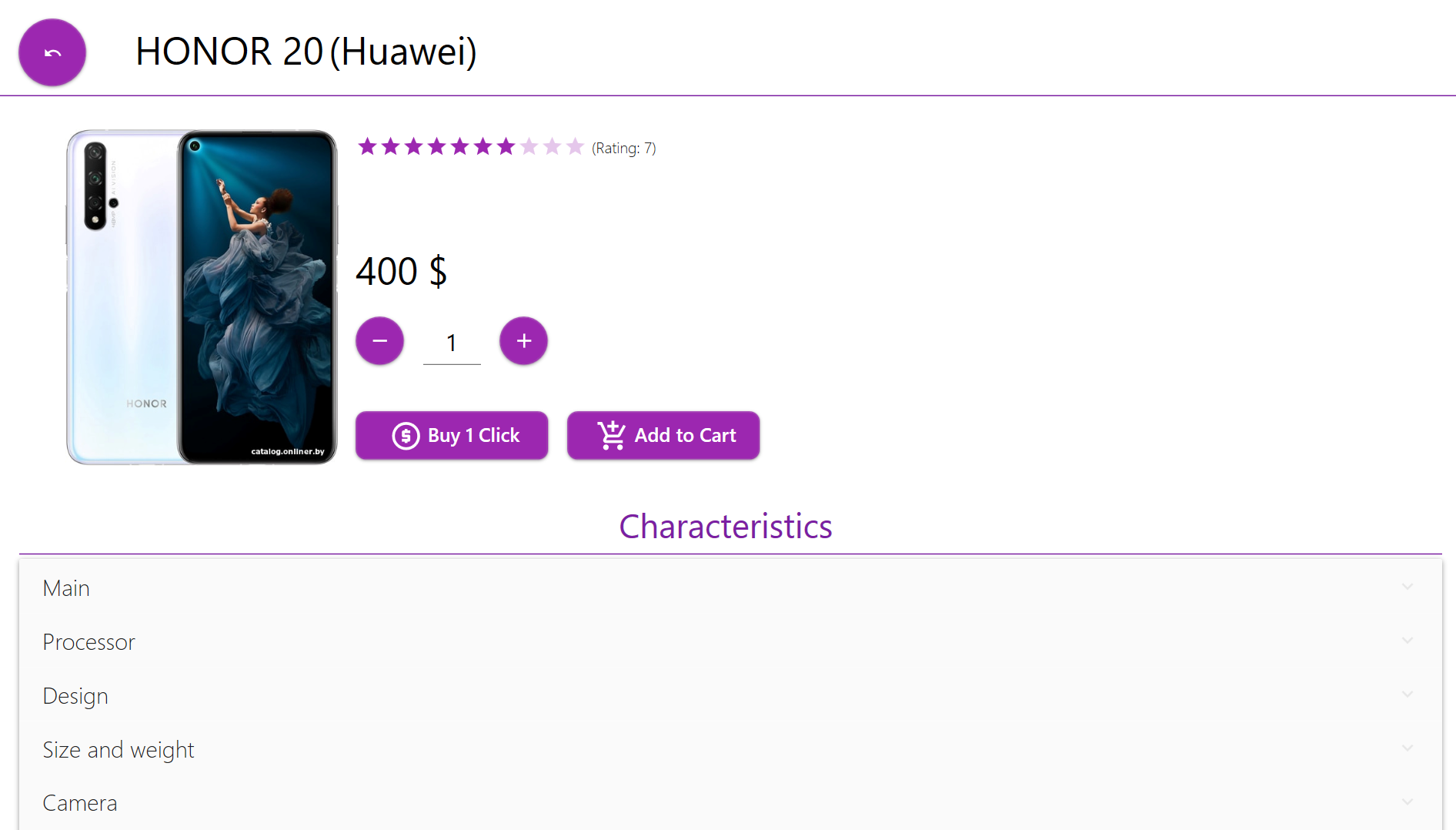


Рисунок 8.2 – Интерфейс приложения для анонимного пользователя

Для того чтобы войти в приложение в качестве зарегистрированного пользователя или администратора, анонимному пользователю необходимо нажать на соответствующий пункт меню, которое отображается, при нажатии на кнопку в правом верхнем углу. На рисунках 8.3 – 8.4 показаны окна для входа и регистрации.

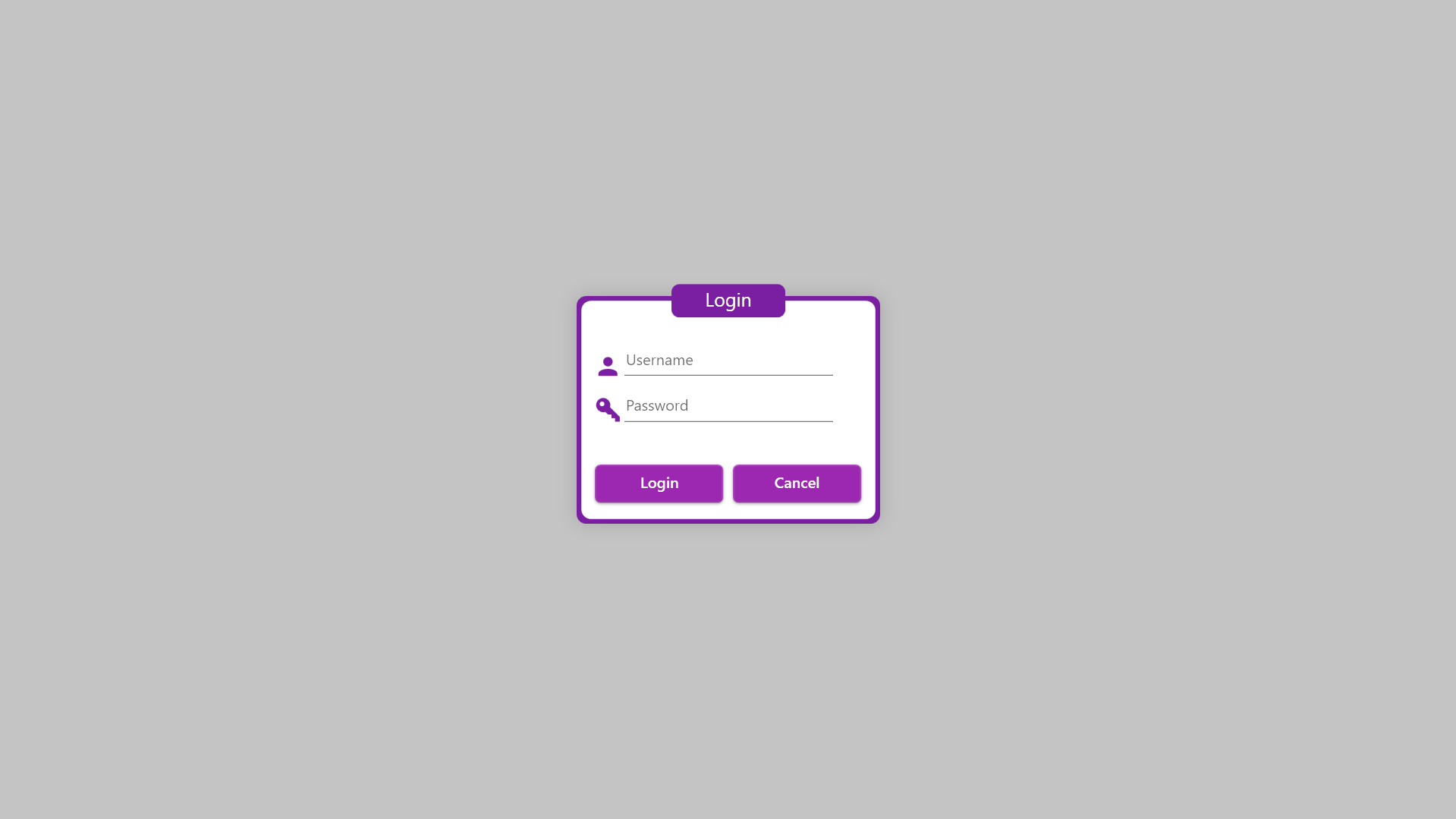


Рисунок 8.3 – Окно входа

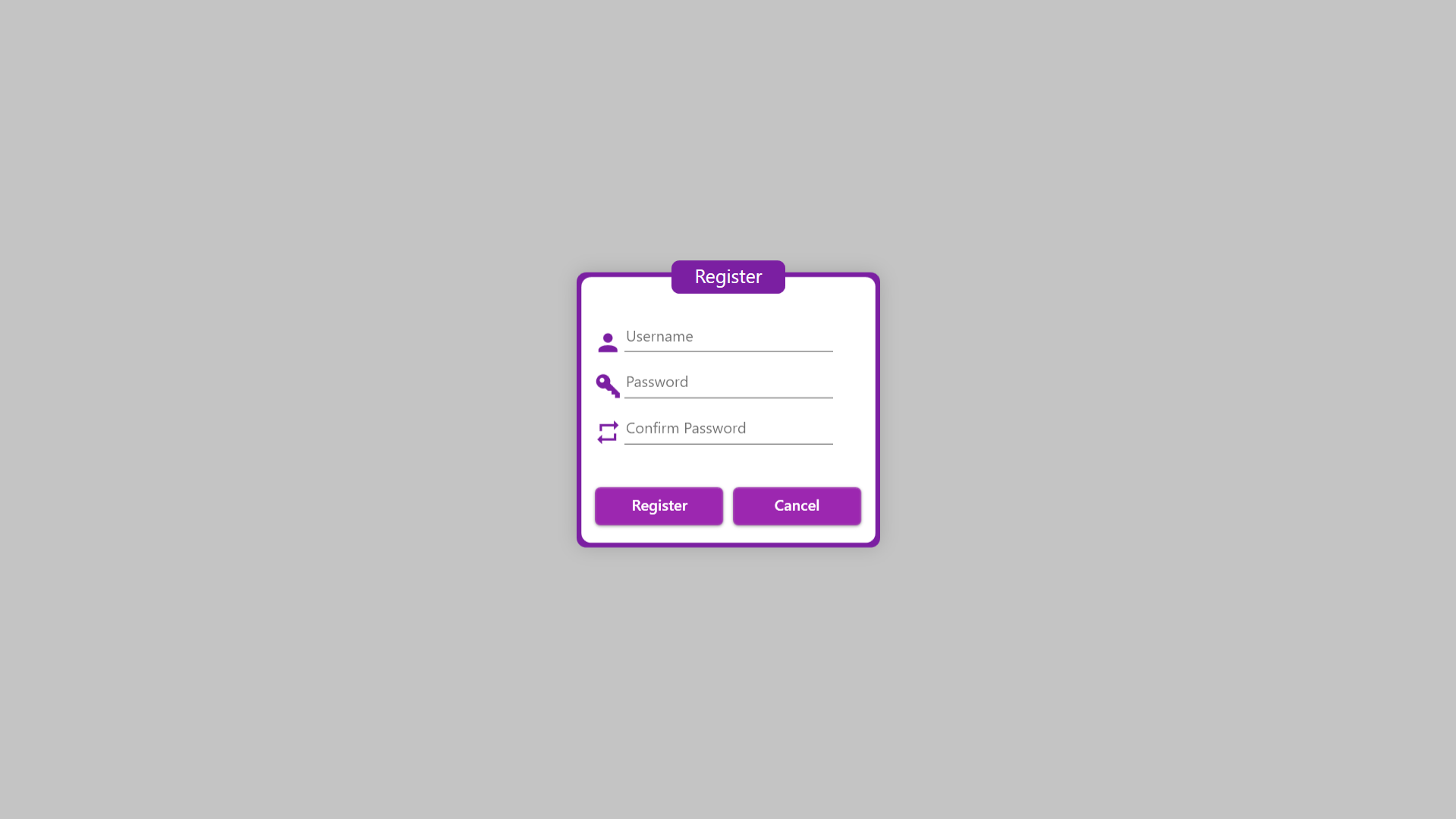


Рисунок 8.4 – Окно регистрации

Войдем в приложение, как зарегистрированный пользователь и рассмотрим интерфейс и возможности зарегистрированного пользователя. Интерфейс для зарегистрированного пользователя, представлен на рисунках 8.5 – 8.7.

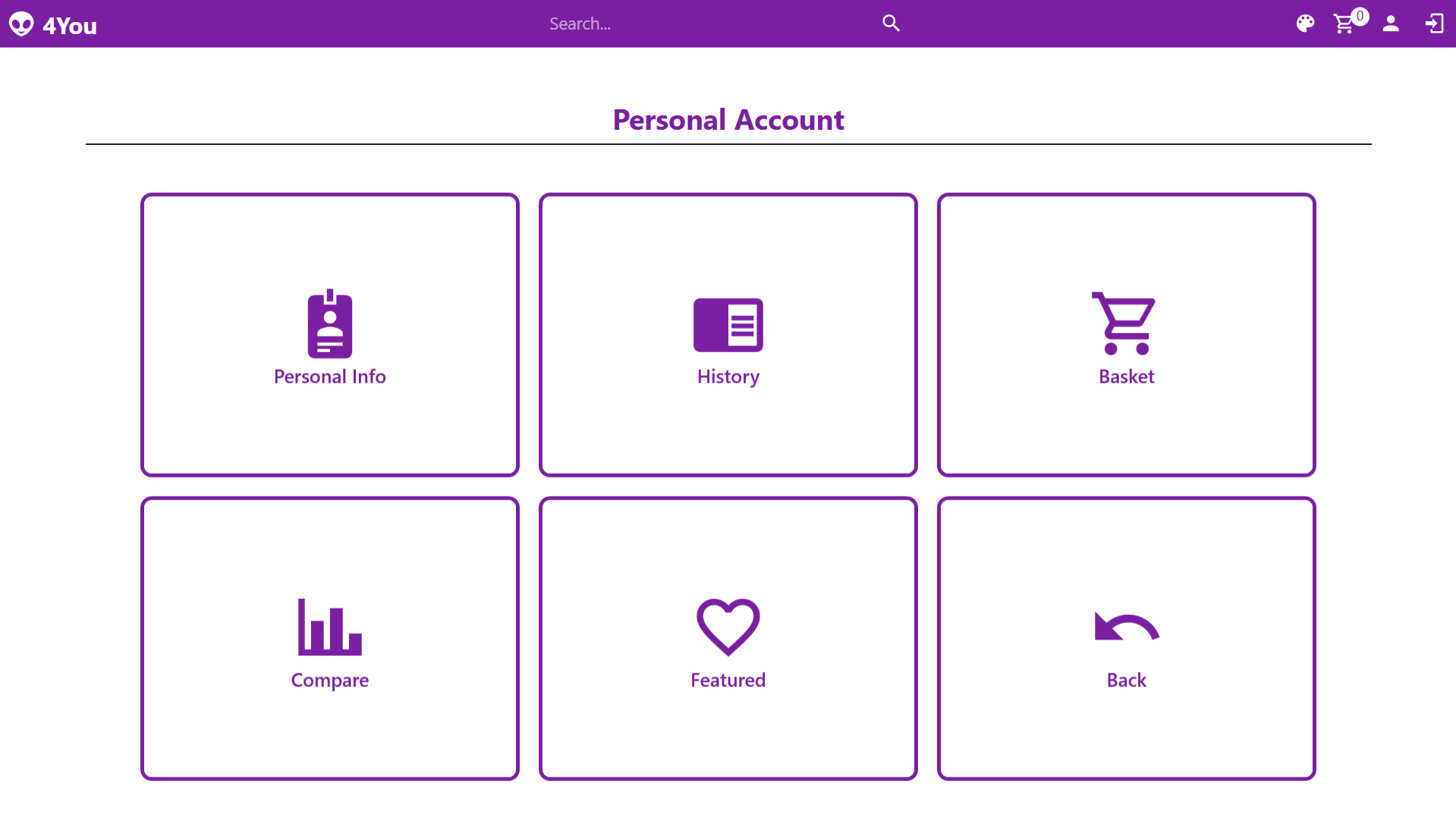


Рисунок 8.5 – Интерфейс приложения, для зарегистрированного пользователя

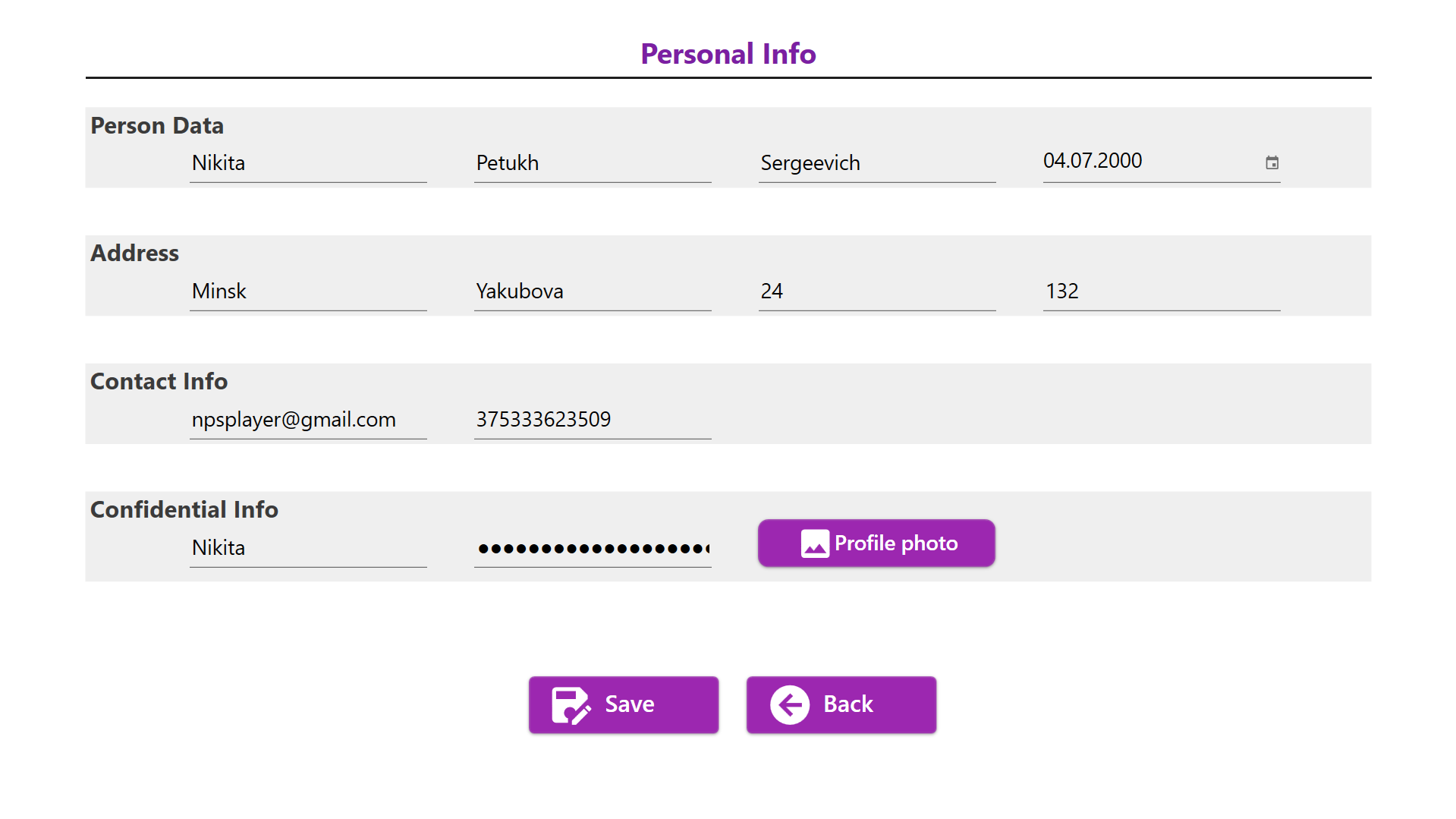


Рисунок 8.6 – Интерфейс приложения, для зарегистрированного пользователя

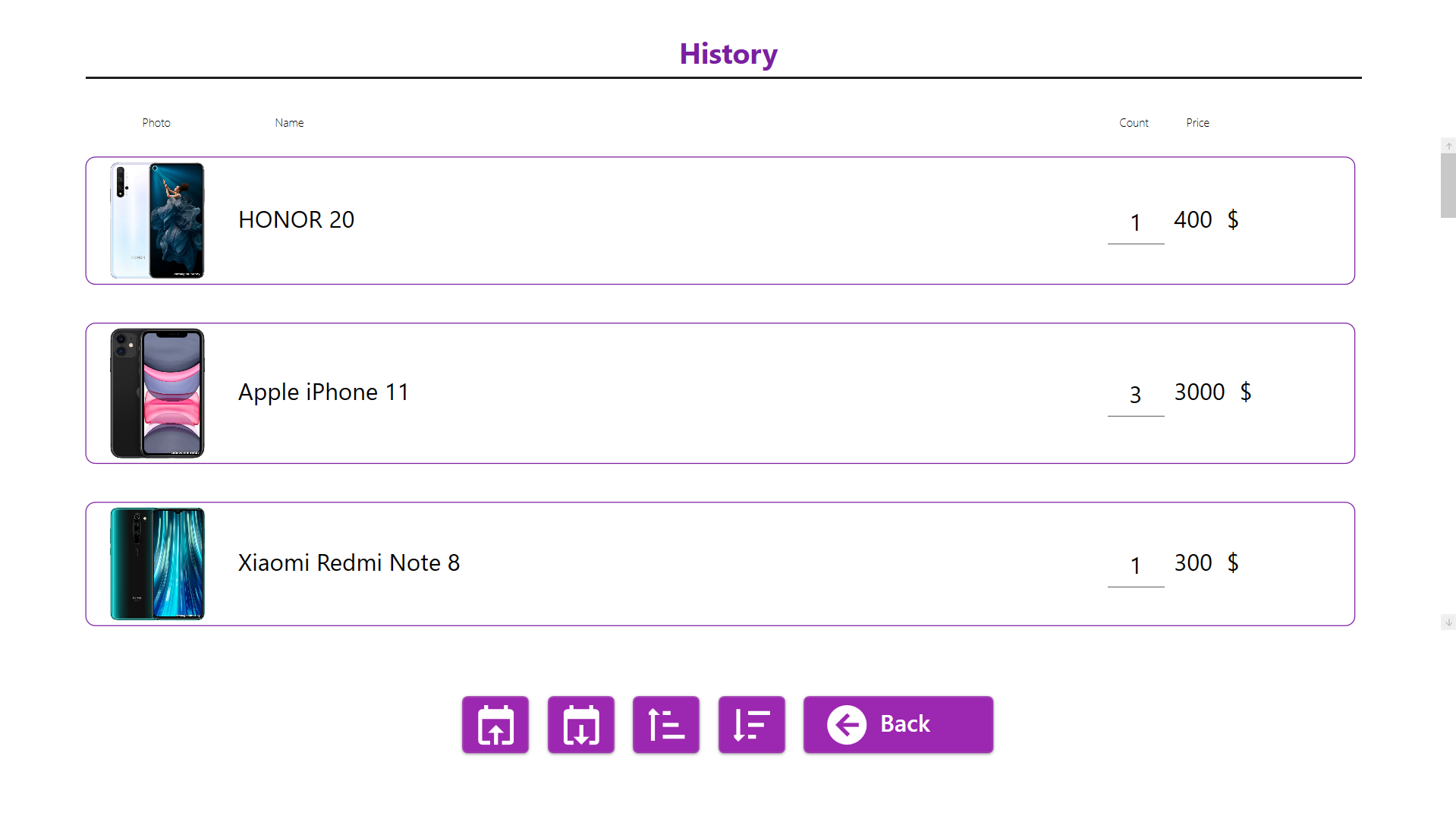


Рисунок 8.7 – Интерфейс приложения, для зарегистрированного пользователя

Так же важной функцией зарегистрированного пользователя, является возможность оставлять отзывы о товаре, интерфейс представлен на рисунке 8.5.

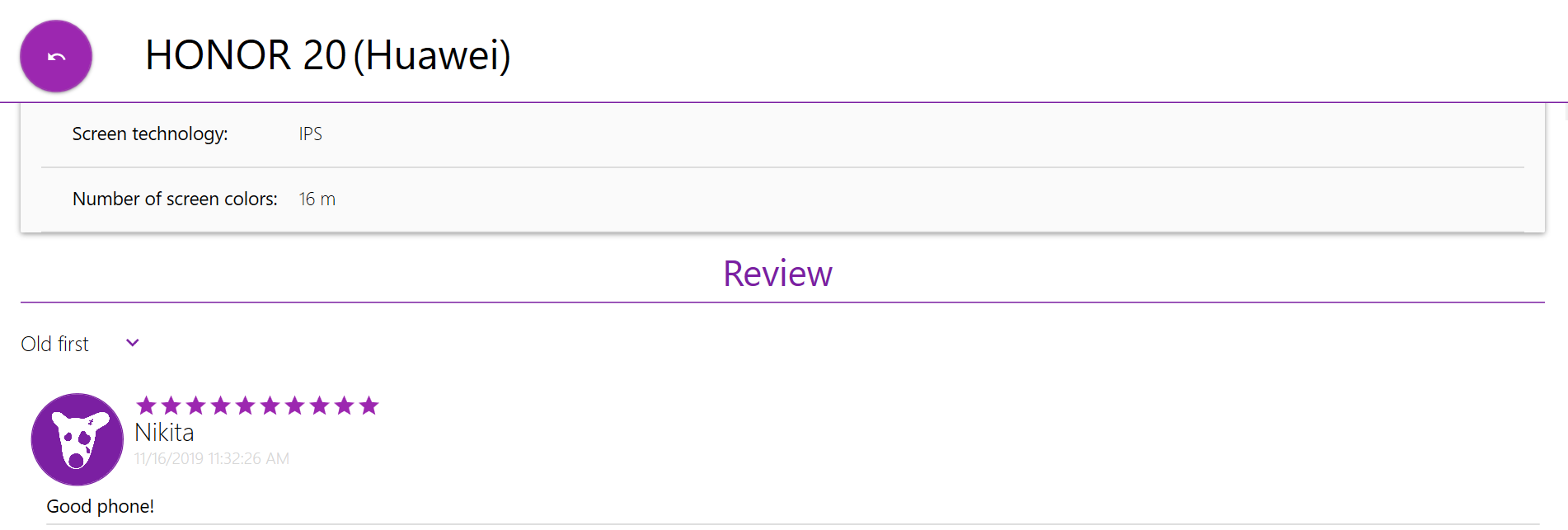
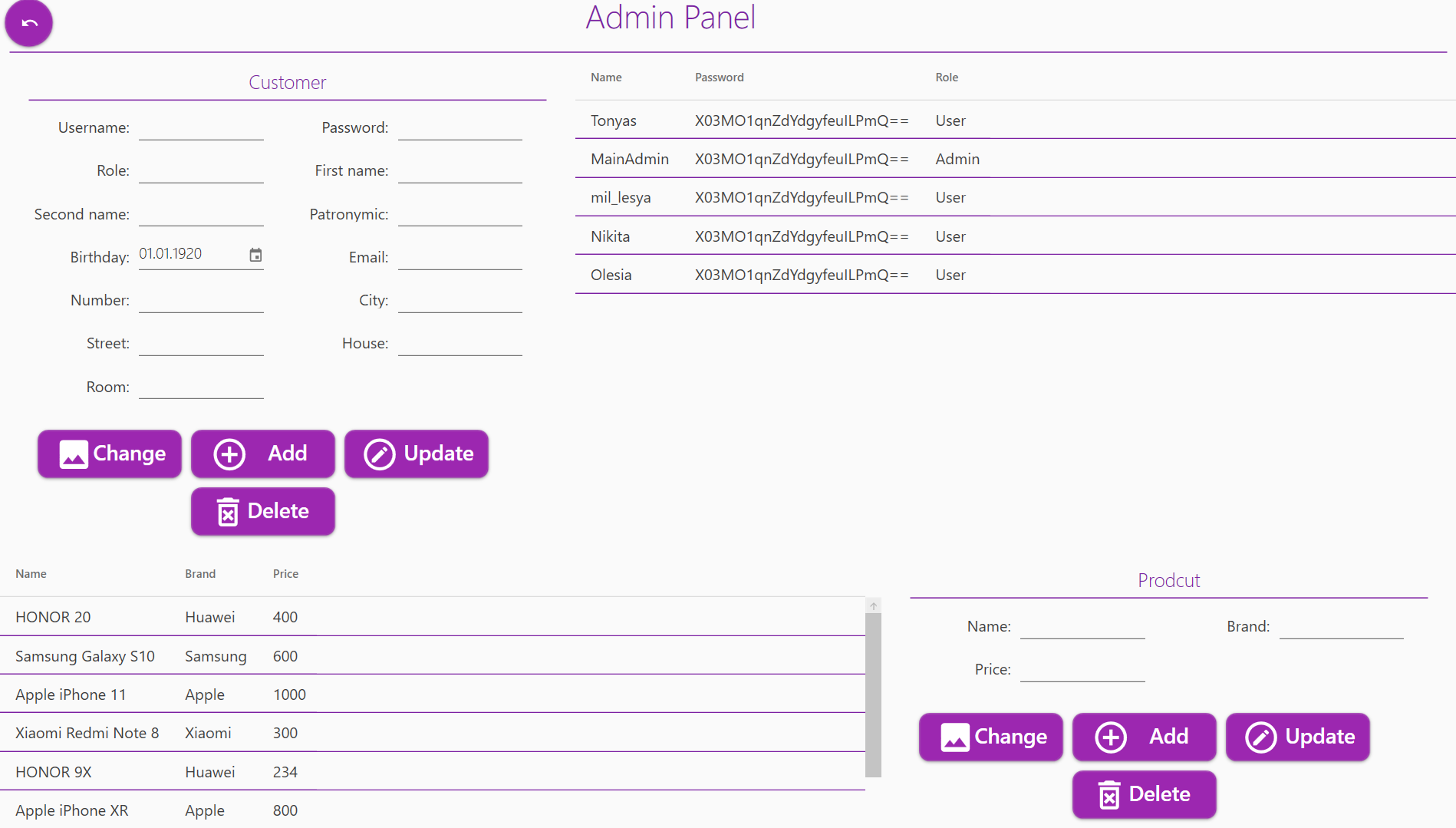


Рисунок 8.8 – Интерфейс отзывов о товаре

Выйдем из аккаунта и зайдем в приложение в качестве администратора и рассмотрим его возможности.

Рассмотрим возможности администратора на примере статей. Интерфейс администратора показан на рисунке 8.8.

  
Рисунок 8.8 – Интерфейс администратор

# Заключение

В результате выполнения данного курсового проекта была создана база данных для магазина мобильных устройств, содержащая:

* 12 таблиц;
* 24 процедуры;
* 4 пользователя базы данных.

В программном средстве были реализованы функции, описанные далее.

Анонимный пользователь может:

* выполнять авторизацию;
* просматривать каталог товаров;
* просматривать характеристики товара;
* просматривать отзывы других пользователей о товаре.

Администратор может:

− добавлять товары;

− редактировать товары;

− удалять товары;

* просматривать товары.

Зарегистрированный пользователь может:

* просматривать товары;
* добавлять товары в свои избранные;
* удалять товары из избранного;
* добавлять в корзину;
* удалять из корзины;
* добавлять в сравнения;
* удалять из сравнения;
* просматривать историю покупок;
* добавлять конфиденциальную информацию в профиле пользователя;
* удалять конфиденциальную информацию в профиле пользователя;
* изменять конфиденциальную информацию в профиле пользователя;
* изменять изображение в профиле пользователя;
* осуществлять поиск.

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

# Список используемых источников

1. Блинова Е.А. Курс лекций по базам данных / Е.А. Блинова
2. Пацей, Н.В. Технология разработки программного обеспечения / Н.В. Пацей – Минск: БГТУ, 2016. – 129 с.
3. ProfessorWeb .NET & Web Programming [Электронный ресурс] – Режим доступа: https://professorweb.ru.
4. Хабр [Электронный ресурс] – Режим доступа: <https://habr.com>
5. Oracle Help Center [Электронный ресурс] – Режим доступа <https://docs.oracle.com/>

# Приложение А

CREATE TABLE "USERS" (

"UserID" NUMBER(10,0) NOT NULL,

"Username" VARCHAR2(50) NOT NULL,

"Password" VARCHAR2(50) NOT NULL,

"Role" VARCHAR2(50) NOT NULL,

constraint USERS\_PK PRIMARY KEY ("UserID"));

CREATE TABLE "CUSTOMER" (

"CustomerID" NUMBER(10,0) NOT NULL,

"UserID" NUMBER(10,0) NOT NULL,

"AddressID" NUMBER(10,0) NOT NULL,

"FirstName" VARCHAR2(50) NOT NULL,

"SecondName" VARCHAR2(50) NOT NULL,

"Patronymic" VARCHAR2(50) NOT NULL,

"DateOfBirth" DATE NOT NULL,

"Email" VARCHAR2(50) NOT NULL,

"PhoneNumber" VARCHAR2(50) NOT NULL,

"Photo" BLOB NOT NULL,

constraint CUSTOMER\_PK PRIMARY KEY ("CustomerID"));

CREATE TABLE "ADDRESS" (

"AddressID" NUMBER(10,0) NOT NULL,

"City" VARCHAR2(50) NOT NULL,

"Street" VARCHAR2(50) NOT NULL,

"HouseNumber" VARCHAR2(50) NOT NULL,

"Room" VARCHAR2(50) NOT NULL,

constraint ADDRESS\_PK PRIMARY KEY ("AddressID"));

CREATE TABLE "PRODUCT" (

"ProductID" NUMBER(10,0) NOT NULL,

"Name" VARCHAR2(50) NOT NULL,

"Manufacturer" VARCHAR2(50) NOT NULL,

"Price" INT NOT NULL,

"Photo" BLOB NOT NULL,

constraint PRODUCT\_PK PRIMARY KEY ("ProductID"));

CREATE TABLE "OPTIONTYPE" (

"OptionTypeID" NUMBER(10,0) NOT NULL,

"OptionTypeName" VARCHAR2(50) NOT NULL,

constraint OPTIONTYPE\_PK PRIMARY KEY ("OptionTypeID"));

CREATE TABLE "OPTION" (

"OptionID" NUMBER(10,0) NOT NULL,

"OptionTypeID" NUMBER(10,0) NOT NULL,

"OptionName" VARCHAR2(50) NOT NULL);

CREATE TABLE "PRODUCTOPTION" (

"ProductOptionID" NUMBER(10,0) NOT NULL,

"ProductID" NUMBER(10,0) NOT NULL,

"OptionID" NUMBER(10,0) NOT NULL,

"Value" VARCHAR2(50) NOT NULL,

"Unit" VARCHAR2(50) NOT NULL,

constraint PRODUCTOPTION\_PK PRIMARY KEY ("ProductOptionID"));

CREATE TABLE "BASKET" (

"BasketID" NUMBER(10,0) NOT NULL,

"ProductID" NUMBER(10,0) NOT NULL,

"CustomerID" NUMBER(10,0) NOT NULL,

"Amount" INT NOT NULL,

"Price" INT NOT NULL,

constraint BASKET\_PK PRIMARY KEY ("BasketID"));

CREATE TABLE "FAVORITE" (

"FavoriteID" NUMBER(10,0) NOT NULL,

"ProductID" NUMBER(10,0) NOT NULL,

"CustomerID" NUMBER(10,0) NOT NULL,

constraint FAVORITE\_PK PRIMARY KEY ("FavoriteID"));

CREATE TABLE "ORDERHISTORY" (

"OrderHistoryID" NUMBER(10,0) NOT NULL,

"ProdcutID" NUMBER(10,0) NOT NULL,

"CustomerID" NUMBER(10,0) NOT NULL,

"Status" VARCHAR2(50) NOT NULL,

"Date" DATE NOT NULL,

"Amount" INT NOT NULL,

"Price" INT NOT NULL,

"KeyFindProduct" INT NOT NULL,

constraint ORDERHISTORY\_PK PRIMARY KEY ("OrderHistoryID"));

CREATE TABLE "REVIEW" (

"ReviewID" NUMBER(10,0) NOT NULL,

"ProductID" NUMBER(10,0) NOT NULL,

"CustomerID" NUMBER(10,0) NOT NULL,

"Discription" VARCHAR2(1000) NOT NULL,

"Rating" INT NOT NULL,

"Date" DATE NOT NULL,

constraint REVIEW\_PK PRIMARY KEY ("ReviewID"));

CREATE TABLE "PRODUCTCOMPRASION" (

"ProductComparisonID" NUMBER(10,0) NOT NULL,

"ProductID" NUMBER(10,0) NOT NULL,

"CustomerID" NUMBER(10,0) NOT NULL,

constraint PRODUCTCOMPRASION\_PK PRIMARY KEY ("ProductComparisonID"));

ALTER TABLE "CUSTOMER" ADD CONSTRAINT "CUSTOMER\_fk0" FOREIGN KEY ("UserID") REFERENCES "USERS"("UserID");

ALTER TABLE "CUSTOMER" ADD CONSTRAINT "CUSTOMER\_fk1" FOREIGN KEY ("AddressID") REFERENCES "ADDRESS"("AddressID");

ALTER TABLE "OPTION" ADD CONSTRAINT "OPTION\_fk0" FOREIGN KEY ("OptionTypeID") REFERENCES "OPTIONTYPE"("OptionTypeID");

ALTER TABLE "PRODUCTOPTION" ADD CONSTRAINT "PRODUCTOPTION\_fk0" FOREIGN KEY ("ProductID") REFERENCES "PRODUCT"("ProductID");

ALTER TABLE "PRODUCTOPTION" ADD CONSTRAINT "PRODUCTOPTION\_fk1" FOREIGN KEY ("OptionID") REFERENCES "OPTION"("OptionID");

ALTER TABLE "BASKET" ADD CONSTRAINT "BASKET\_fk0" FOREIGN KEY ("ProductID") REFERENCES "PRODUCT"("ProductID");

ALTER TABLE "BASKET" ADD CONSTRAINT "BASKET\_fk1" FOREIGN KEY ("CustomerID") REFERENCES "CUSTOMER"("CustomerID");

ALTER TABLE "FAVORITE" ADD CONSTRAINT "FAVORITE\_fk0" FOREIGN KEY ("ProductID") REFERENCES "PRODUCT"("ProductID");

ALTER TABLE "FAVORITE" ADD CONSTRAINT "FAVORITE\_fk1" FOREIGN KEY ("CustomerID") REFERENCES "CUSTOMER"("CustomerID");

ALTER TABLE "ORDERHISTORY" ADD CONSTRAINT "ORDERHISTORY\_fk0" FOREIGN KEY ("ProdcutID") REFERENCES "PRODUCT"("ProductID");

ALTER TABLE "ORDERHISTORY" ADD CONSTRAINT "ORDERHISTORY\_fk1" FOREIGN KEY ("CustomerID") REFERENCES "CUSTOMER"("CustomerID");

ALTER TABLE "REVIEW" ADD CONSTRAINT "REVIEW\_fk0" FOREIGN KEY ("ProductID") REFERENCES "PRODUCT"("ProductID");

ALTER TABLE "REVIEW" ADD CONSTRAINT "REVIEW\_fk1" FOREIGN KEY ("CustomerID") REFERENCES "CUSTOMER"("CustomerID");

ALTER TABLE "PRODUCTCOMPRASION" ADD CONSTRAINT "PRODUCTCOMPRASION\_fk0" FOREIGN KEY ("ProductID") REFERENCES "PRODUCT"("ProductID");

ALTER TABLE "PRODUCTCOMPRASION" ADD CONSTRAINT "PRODUCTCOMPRASION\_fk1" FOREIGN KEY ("CustomerID") REFERENCES "CUSTOMER"("CustomerID");

# Приложение Б

ALTER SESSION SET "\_ORACLE\_SCRIPT"=true;

DROP TABLESPACE TS\_APPUSER;

CREATE TABLESPACE TS\_APPUSER

DATAFILE 'TS\_APPUSER.dat'

SIZE 10M

AUTOEXTEND ON;

DROP TABLESPACE TS\_APPUSER\_TEMP;

CREATE TEMPORARY TABLESPACE TS\_APPUSER\_TEMP

TEMPFILE 'TS\_APPUSER\_TEMP.dat'

SIZE 5M

AUTOEXTEND ON;

CREATE ROLE RL\_APPUSER;

GRANT ALL PRIVILEGES TO RL\_APPUSER;

CREATE PROFILE PF\_APPUSER LIMIT

PASSWORD\_LIFE\_TIME 180

SESSIONS\_PER\_USER 3

FAILED\_LOGIN\_ATTEMPTS 7

PASSWORD\_LOCK\_TIME 1

PASSWoRD\_REUSE\_TIME 10

PASSWORD\_GRACE\_TIME DEFAULT

CONNECT\_TIME 180

IDLE\_TIME 30;

DROP USER APPUSER CASCADE;

CREATE USER APPUSER IDENTIFIED BY Pa$$w0rd

DEFAULT TABLESPACE TS\_APPUSER QUOTA UNLIMITED ON TS\_APPUSER

TEMPORARY TABLESPACE TS\_APPUSER\_TEMP

PROFILE PF\_APPUSER;

GRANT RL\_APPUSER TO APPUSER;

DROP TABLESPACE TS\_APPADMIN;

CREATE TABLESPACE TS\_APPADMIN

DATAFILE 'TS\_APPADMIN.dat'

SIZE 10M

AUTOEXTEND ON;

DROP TABLESPACE TS\_APPADMIN\_TEMP;

CREATE TEMPORARY TABLESPACE TS\_APPADMIN\_TEMP

TEMPFILE 'TS\_APPADMIN\_TEMP.dat'

SIZE 5M

AUTOEXTEND ON;

CREATE ROLE RL\_APPADMIN;

GRANT CREATE SESSION TO RL\_APPADMIN;

GRANT EXECUTE ON ADMINADDCUSTOMER TO RL\_APPADMIN;

GRANT EXECUTE ON ADMINADDOPTION TO RL\_APPADMIN;

GRANT EXECUTE ON ADMINADDPRODUCT TO RL\_APPADMIN;

GRANT EXECUTE ON ADMINDELETECUSTOMER TO RL\_APPADMIN;

GRANT EXECUTE ON ADMINDELETEOPTION TO RL\_APPADMIN;

GRANT EXECUTE ON ADMINDELETEPRODUCT TO RL\_APPADMIN;

GRANT EXECUTE ON ADMINUPDATECUSTOMER TO RL\_APPADMIN;

GRANT EXECUTE ON ADMINUPDATEOPTION TO RL\_APPADMIN;

GRANT EXECUTE ON ADMINUPDATEPRODUCT TO RL\_APPADMIN;

GRANT EXECUTE ON LOGIN TO RL\_APPADMIN;

GRANT EXECUTE ON REGISTEADMIN TO RL\_APPADMIN;

GRANT EXECUTE ON REGISTERUSER TO RL\_APPADMIN;

GRANT EXECUTE ON TABLE\_TO\_XML\_FILE TO RL\_APPADMIN;

GRANT EXECUTE ON SHOWCATALOG TO RL\_APPADMIN;

CREATE PROFILE PF\_APPADMIN LIMIT

PASSWORD\_LIFE\_TIME 180

SESSIONS\_PER\_USER 3

FAILED\_LOGIN\_ATTEMPTS 7

PASSWORD\_LOCK\_TIME 1

PASSWoRD\_REUSE\_TIME 10

PASSWORD\_GRACE\_TIME DEFAULT

CONNECT\_TIME 180

IDLE\_TIME 30;

DROP USER APPAPPADMIN CASCADE;

CREATE USER APPADMIN IDENTIFIED BY Pa$$w0rd

DEFAULT TABLESPACE TS\_APPADMIN QUOTA UNLIMITED ON TS\_APPADMIN

TEMPORARY TABLESPACE TS\_APPADMIN\_TEMP

PROFILE PF\_APPADMIN;

GRANT RL\_APPADMIN TO APPADMIN;

DROP TABLESPACE TS\_APPCLIENT;

CREATE TABLESPACE TS\_APPCLIENT

DATAFILE 'TS\_APPCLIENT.dat'

SIZE 10M

AUTOEXTEND ON;

DROP TABLESPACE TS\_APPCLIENT\_TEMP;

CREATE TEMPORARY TABLESPACE TS\_APPCLIENT\_TEMP

TEMPFILE 'TS\_APPCLIENT\_TEMP.dat'

SIZE 5M

AUTOEXTEND ON;

CREATE ROLE RL\_APPCLIENT;

GRANT CREATE SESSION TO RL\_APPCLIENT;

GRANT EXECUTE ON BASKETDELETEONE TO RL\_APPCLIENT;

GRANT EXECUTE ON LOGIN TO RL\_APPCLIENT;

GRANT EXECUTE ON PERSONAINFOUPDATE TO RL\_APPCLIENT;

GRANT EXECUTE ON PERSONALINFOSELECT TO RL\_APPCLIENT;

GRANT EXECUTE ON REGISTERUSER TO RL\_APPCLIENT;

GRANT EXECUTE ON SHOWBASKET TO RL\_APPCLIENT;

GRANT EXECUTE ON SHOWCATALOG TO RL\_APPCLIENT;

GRANT EXECUTE ON SHOWFAVORITE TO RL\_APPCLIENT;

GRANT EXECUTE ON SHOWHISTORY TO RL\_APPCLIENT;

GRANT EXECUTE ON SHOWOPTIONPRODUCT TO RL\_APPCLIENT;

GRANT EXECUTE ON SHOWREVIEW TO RL\_APPCLIENT;

CREATE PROFILE PF\_APPCLIENT LIMIT

PASSWORD\_LIFE\_TIME 180

SESSIONS\_PER\_USER 3

FAILED\_LOGIN\_ATTEMPTS 7

PASSWORD\_LOCK\_TIME 1

PASSWoRD\_REUSE\_TIME 10

PASSWORD\_GRACE\_TIME DEFAULT

CONNECT\_TIME 180

IDLE\_TIME 30;

DROP USER APPCLIENT CASCADE;

CREATE USER APPCLIENT IDENTIFIED BY Pa$$w0rd

DEFAULT TABLESPACE TS\_APPCLIENT QUOTA UNLIMITED ON TS\_APPCLIENT

TEMPORARY TABLESPACE TS\_APPCLIENT\_TEMP

PROFILE PF\_APPCLIENT;

GRANT RL\_APPCLIENT TO APPCLIENT;

DROP TABLESPACE TS\_APPANONIM;

CREATE TABLESPACE TS\_APPANONIM

DATAFILE 'TS\_APPANONIM.dat'

SIZE 10M

AUTOEXTEND ON;

DROP TABLESPACE TS\_APPANONIM\_TEMP;

CREATE TEMPORARY TABLESPACE TS\_APPANONIM\_TEMP

TEMPFILE 'TS\_APPANONIM\_TEMP.dat'

SIZE 5M

AUTOEXTEND ON;

CREATE ROLE RL\_APPANONIM;

GRANT CREATE SESSION TO RL\_APPANONIM;

GRANT EXECUTE ON SHOWCATALOG TO RL\_APPANONIM;

GRANT EXECUTE ON SHOWOPTIONPRODUCT TO RL\_APPANONIM;

GRANT EXECUTE ON SHOWREVIEW TO RL\_APPANONIM;

CREATE PROFILE PF\_APPANONIM LIMIT

PASSWORD\_LIFE\_TIME 180

SESSIONS\_PER\_USER 3

FAILED\_LOGIN\_ATTEMPTS 7

PASSWORD\_LOCK\_TIME 1

PASSWoRD\_REUSE\_TIME 10

PASSWORD\_GRACE\_TIME DEFAULT

CONNECT\_TIME 180

IDLE\_TIME 30;

DROP USER APPANONIM CASCADE;

CREATE USER APPANONIM IDENTIFIED BY Pa$$w0rd

DEFAULT TABLESPACE TS\_APPANONIM QUOTA UNLIMITED ON TS\_APPANONIM

TEMPORARY TABLESPACE TS\_APPANONIM\_TEMP

PROFILE PF\_APPANONIM;

GRANT RL\_APPANONIM TO APPANONIM;

# Приложение В

DROP PROCEDURE REGISTERUSER;

CREATE OR REPLACE PROCEDURE REGISTERUSER (p\_username IN USERS."Username"%TYPE,

p\_password IN USERS."Password"%TYPE

) IS

BEGIN

INSERT INTO USERS("Username", "Password", "Role")

VALUES (p\_username, p\_password, 'User');

INSERT INTO ADDRESS("AddressID")

VALUES (SQ\_ADDRESS.NEXTVAL);

INSERT INTO CUSTOMER("CustomerID","AddressID","UserID", "DateOfBirth")

VALUES (SQ\_CUSTOMER.nextval,SQ\_ADDRESS.CURRVAL, SQ\_USERS.CURRVAL, '01.01.1920');

COMMIT;

END;

DROP PROCEDURE REGISTEADMIN;

CREATE OR REPLACE PROCEDURE REGISTEADMIN (p\_username IN USERS."Username"%TYPE,

p\_password IN USERS."Password"%TYPE

) IS

BEGIN

INSERT INTO USERS("Username", "Password", "Role")

VALUES (p\_username, p\_password, 'Admin');

INSERT INTO ADDRESS("AddressID")

VALUES (SQ\_ADDRESS.NEXTVAL);

INSERT INTO CUSTOMER("CustomerID","AddressID","UserID", "DateOfBirth")

VALUES (SQ\_CUSTOMER.nextval,SQ\_ADDRESS.CURRVAL, SQ\_USERS.CURRVAL, '01.01.1920');

COMMIT;

END;

DROP PROCEDURE ADMINADDCUSTOMER;

CREATE OR REPLACE PROCEDURE ADMINADDCUSTOMER (p\_username IN USERS."Username"%TYPE,

p\_password IN USERS."Password"%TYPE,

p\_role IN USERS."Role"%TYPE,

p\_city IN ADDRESS."City"%TYPE,

p\_street IN ADDRESS."Street"%TYPE,

p\_housenumber IN ADDRESS."HouseNumber"%TYPE,

p\_room IN ADDRESS."Room"%TYPE,

p\_firstname IN CUSTOMER."FirstName"%TYPE,

p\_patronymic IN CUSTOMER."Patronymic"%TYPE,

p\_dateofbirth IN CUSTOMER."DateOfBirth"%TYPE,

p\_email IN CUSTOMER."Email"%TYPE,

p\_phonenumber IN CUSTOMER."PhoneNumber"%TYPE

) IS BEGIN

INSERT INTO USERS("Username", "Password", "Role")

VALUES (p\_username, p\_password, p\_role);

INSERT INTO ADDRESS("AddressID", "City", "Street", "HouseNumber", "Room")

VALUES (SQ\_ADDRESS.NEXTVAL, p\_city, p\_street, p\_housenumber, p\_room);

INSERT INTO CUSTOMER("CustomerID","AddressID","UserID", "DateOfBirth", "FirstName", "SecondName",

"Patronymic", "Email", "PhoneNumber")

VALUES (SQ\_CUSTOMER.nextval,SQ\_ADDRESS.CURRVAL, SQ\_USERS.CURRVAL, p\_dateofbirth, p\_firstname, p\_secondname,

p\_patronymic, p\_email, p\_phonenumber); COMMIT;

END;

DROP PROCEDURE ADMINUPDATECUSTOMER;

CREATE OR REPLACE PROCEDURE ADMINUPDATECUSTOMER(p\_userid IN USERS."UserID"%TYPE,

p\_username IN USERS."Username"%TYPE,

p\_password IN USERS."Password"%TYPE,

p\_role IN USERS."Role"%TYPE,

p\_addressid IN ADDRESS."AddressID"%TYPE,

p\_city IN ADDRESS."City"%TYPE,

p\_street IN ADDRESS."Street"%TYPE,

p\_housenumber IN ADDRESS."HouseNumber"%TYPE,

p\_room IN ADDRESS."Room"%TYPE,

p\_firstname IN CUSTOMER."FirstName"%TYPE,

p\_secondname IN CUSTOMER."SecondName"%TYPE,

p\_patronymic IN CUSTOMER."Patronymic"%TYPE,

p\_dateofbirth IN CUSTOMER."DateOfBirth"%TYPE,

p\_email IN CUSTOMER."Email"%TYPE,

p\_phonenumber IN CUSTOMER."PhoneNumber"%TYPE) IS

BEGIN

UPDATE USERS SET USERS."Username" = p\_username, USERS."Password" = p\_password, USERS."Role" = p\_role

WHERE USERS."UserID" = p\_userid;

UPDATE ADDRESS SET ADDRESS."City" = p\_city, ADDRESS."Street" = p\_street, ADDRESS."HouseNumber" = p\_housenumber,

ADDRESS."Room" = p\_room

WHERE ADDRESS."AddressID" = p\_addressid;

UPDATE CUSTOMER SET CUSTOMER."FirstName" = p\_firstname, CUSTOMER."SecondName" = p\_secondname, CUSTOMER."Patronymic" = p\_patronymic,

CUSTOMER."DateOfBirth" = p\_dateofbirth, CUSTOMER."Email" = p\_email, CUSTOMER."PhoneNumber" = p\_phonenumber

WHERE CUSTOMER."UserID" = p\_userid AND CUSTOMER."AddressID" = p\_addressid;

COMMIT;

END;

DROP PROCEDURE ADMINDELETECUSTOMER;

CREATE OR REPLACE PROCEDURE ADMINDELETECUSTOMER (p\_userid IN USERS."UserID"%TYPE,

p\_addressid IN ADDRESS."AddressID"%TYPE,

p\_customerid IN CUSTOMER."CustomerID"%TYPE

) IS BEGIN

DELETE FROM CUSTOMER WHERE CUSTOMER."CustomerID" = p\_customerid;

DELETE FROM USERS WHERE USERS."UserID" = p\_userid;

DELETE FROM ADDRESS WHERE ADDRESS."AddressID" = p\_addressid;

COMMIT;

END;

DROP PROCEDURE ADMINADDPRODUCT;

CREATE OR REPLACE PROCEDURE ADMINADDPRODUCT(p\_name IN PRODUCT."Name"%TYPE,

p\_manuf IN PRODUCT."Manufacturer"%TYPE,

p\_price IN PRODUCT."Price"%TYPE) IS BEGIN

INSERT INTO PRODUCT("Name", "Manufacturer", "Price") VALUES (p\_name, p\_manuf, p\_price);

COMMIT;

END;

DROP PROCEDURE ADMINUPDATEPRODUCT;

CREATE OR REPLACE PROCEDURE ADMINUPDATEPRODUCT(p\_productid IN PRODUCT."ProductID"%TYPE,

p\_name IN PRODUCT."Name"%TYPE,

p\_manuf IN PRODUCT."Manufacturer"%TYPE,

p\_price IN PRODUCT."Price"%TYPE) IS

BEGIN

UPDATE PRODUCT SET PRODUCT."Name" = p\_name, PRODUCT."Manufacturer" = p\_manuf, PRODUCT."Price" = p\_price

WHERE PRODUCT."ProductID" = p\_productid;

COMMIT;

END;

DROP PROCEDURE ADMINDELETEPRODUCT;

CREATE OR REPLACE PROCEDURE ADMINDELETEPRODUCT(p\_productid IN PRODUCT."ProductID"%TYPE) IS

BEGIN

DELETE FROM PRODUCT WHERE PRODUCT."ProductID" = p\_productid;

COMMIT; END;

DROP PROCEDURE ADMINADDOPTION;

CREATE OR REPLACE PROCEDURE ADMINADDOPTION(p\_productid IN PRODUCTOPTION."ProductID"%TYPE,

p\_optionid IN PRODUCTOPTION."OptionID"%TYPE,

p\_value IN PRODUCTOPTION."Value"%TYPE,

p\_unit IN PRODUCTOPTION."Unit"%TYPE) IS

BEGIN

INSERT INTO PRODUCTOPTION("ProductID", "OptionID", "Value", "Unit") VALUES (p\_productid, p\_optionid, p\_value, p\_unit);

COMMIT; END;

DROP PROCEDURE ADMINUPDATEOPTION;

CREATE OR REPLACE PROCEDURE ADMINUPDATEOPTION(p\_productoptionid IN PRODUCTOPTION."ProductOptionID"%TYPE,

p\_productid IN PRODUCTOPTION."ProductID"%TYPE,

p\_optionid IN PRODUCTOPTION."OptionID"%TYPE,

p\_value IN PRODUCTOPTION."Value"%TYPE,

p\_unit IN PRODUCTOPTION."Unit"%TYPE) IS BEGIN

UPDATE PRODUCTOPTION SET PRODUCTOPTION."ProductID" = p\_productid, PRODUCTOPTION."OptionID" = p\_optionid,

PRODUCTOPTION."Value" = p\_value, PRODUCTOPTION."Unit" = p\_unit

WHERE PRODUCTOPTION."ProductOptionID" = p\_productoptionid;

COMMIT;

END;

DROP PROCEDURE ADMINDELETEOPTION;

CREATE OR REPLACE PROCEDURE ADMINDELETEOPTION(p\_productoptionid IN PRODUCTOPTION."ProductOptionID"%TYPE) IS BEGIN

DELETE FROM PRODUCTOPTION WHERE PRODUCTOPTION."ProductOptionID" = p\_productoptionid;

COMMIT;

END;

DROP PROCEDURE LOGIN;

CREATE OR REPLACE PROCEDURE LOGIN (p\_username IN USERS."Username"%TYPE,

p\_password IN USERS."Password"%TYPE,

p\_userid OUT USERS."UserID"%TYPE,

p\_addressid OUT ADDRESS."AddressID"%TYPE,

p\_customerid OUT CUSTOMER."CustomerID"%TYPE,

p\_role OUT USERS."Role"%TYPE)

IS

BEGIN

SELECT USERS."UserID", ADDRESS."AddressID", CUSTOMER."CustomerID", USERS."Role" INTO p\_userid, p\_addressid,

p\_customerid, p\_role FROM CUSTOMER

INNER JOIN ADDRESS ON CUSTOMER."AddressID" = ADDRESS."AddressID"

INNER JOIN USERS ON CUSTOMER."UserID" = USERS."UserID"

WHERE USERS."Username" = p\_username AND USERS."Password" = p\_password; END LOGIN;

DROP PROCEDURE PERSONALINFOSELECT;

CREATE OR REPLACE PROCEDURE PERSONALINFOSELECT (p\_userid IN USERS."UserID"%TYPE,

p\_username OUT USERS."Username"%TYPE,

p\_password OUT USERS."Password"%TYPE,

p\_city OUT ADDRESS."City"%TYPE,

p\_street OUT ADDRESS."Street"%TYPE,

p\_housenumber OUT ADDRESS."HouseNumber"%TYPE,

p\_room OUT ADDRESS."Room"%TYPE,

p\_firstname OUT CUSTOMER."FirstName"%TYPE,

p\_secondname OUT CUSTOMER."SecondName"%TYPE,

p\_patronymic OUT CUSTOMER."Patronymic"%TYPE,

p\_dateofbirth OUT CUSTOMER."DateOfBirth"%TYPE,

p\_email OUT CUSTOMER."Email"%TYPE,

p\_phonenumber OUT CUSTOMER."PhoneNumber"%TYPE

) IS BEGIN

SELECT USERS."Username", USERS."Password", ADDRESS."City", ADDRESS."Street", ADDRESS."HouseNumber", ADDRESS."Room",

CUSTOMER."FirstName", CUSTOMER."SecondName", CUSTOMER."Patronymic", CUSTOMER."DateOfBirth", CUSTOMER."Email",

CUSTOMER."PhoneNumber"

INTO p\_username, p\_password, p\_city, p\_street, p\_housenumber, p\_room, p\_firstname, p\_secondname, p\_patronymic, p\_dateofbirth,

p\_email, p\_phonenumber FROM CUSTOMER

INNER JOIN ADDRESS ON CUSTOMER."AddressID" = ADDRESS."AddressID"

INNER JOIN USERS ON CUSTOMER."UserID" = USERS."UserID"

WHERE USERS."UserID" = p\_userid;

END PERSONALINFOSELECT;

DROP PROCEDURE PERSONAINFOUPDATE;

CREATE OR REPLACE PROCEDURE PERSONAINFOUPDATE(p\_userid IN USERS."UserID"%TYPE,

p\_username IN USERS."Username"%TYPE,

p\_password IN USERS."Password"%TYPE,

p\_addressid IN ADDRESS."AddressID"%TYPE,

p\_city IN ADDRESS."City"%TYPE,

p\_street IN ADDRESS."Street"%TYPE,

p\_housenumber IN ADDRESS."HouseNumber"%TYPE,

p\_room IN ADDRESS."Room"%TYPE,

p\_firstname IN CUSTOMER."FirstName"%TYPE,

p\_secondname IN CUSTOMER."SecondName"%TYPE,

p\_patronymic IN CUSTOMER."Patronymic"%TYPE,

p\_dateofbirth IN CUSTOMER."DateOfBirth"%TYPE,

p\_email IN CUSTOMER."Email"%TYPE,

p\_phonenumber IN CUSTOMER."PhoneNumber"%TYPE

) IS BEGIN

UPDATE USERS SET USERS."Username" = p\_username, USERS."Password" = p\_password

WHERE USERS."UserID" = p\_userid;

UPDATE ADDRESS SET ADDRESS."City" = p\_city, ADDRESS."Street" = p\_street, ADDRESS."HouseNumber" = p\_housenumber,

ADDRESS."Room" = p\_room

WHERE ADDRESS."AddressID" = p\_addressid;

UPDATE CUSTOMER SET CUSTOMER."FirstName" = p\_firstname, CUSTOMER."SecondName" = p\_secondname, CUSTOMER."Patronymic" = p\_patronymic,

CUSTOMER."DateOfBirth" = p\_dateofbirth, CUSTOMER."Email" = p\_email, CUSTOMER."PhoneNumber" = p\_phonenumber

WHERE CUSTOMER."UserID" = p\_userid AND CUSTOMER."AddressID" = p\_addressid;

COMMIT;

END;

DROP PROCEDURE SHOWCATALOG;

CREATE OR REPLACE PROCEDURE SHOWCATALOG(p\_productid OUT "PRODUCT"."ProductID"%TYPE, p\_name OUT "PRODUCT"."Name"%TYPE,

p\_manufacturer OUT "PRODUCT"."Manufacturer"%TYPE, p\_price OUT "PRODUCT"."Price"%TYPE) IS

BEGIN

FOR rec IN (

SELECT PRODUCT."ProductID", PRODUCT."Name", PRODUCT."Manufacturer", PRODUCT."Price"

INTO p\_productid, p\_name, p\_manufacturer, p\_price

FROM PRODUCT)

LOOP

DBMS\_OUTPUT.enable;

DBMS\_OUTPUT.put\_line( rec."ProductID" || ' ' || rec."Name" || ' ' || rec."Manufacturer" || ' ' || rec."Price"); END LOOP;

END SHOWCATALOG;

DROP PROCEDURE BASKETDELETEONE;

CREATE OR REPLACE PROCEDURE BASKETDELETEONE(p\_basketid IN BASKET."BasketID"%TYPE,

p\_customerid IN BASKET."CustomerID"%TYPE) IS

BEGIN

DELETE FROM BASKET WHERE BASKET."BasketID" = p\_basketid AND BASKET."CustomerID" = p\_customerid;

UPDATE ORDERHISTORY SET ORDERHISTORY."Status" = 'Delete from basket' WHERE ORDERHISTORY."CustomerID" = p\_customerid AND

ORDERHISTORY."KeyFindProduct" = p\_basketid;

END;

CREATE OR REPLACE PROCEDURE TOTALPRICEBASKET(p\_customer IN "BASKET"."CustomerID"%TYPE,

p\_total OUT INTEGER) IS

BEGIN

SELECT SUM((BASKET."Price"))

INTO p\_total

FROM BASKET

WHERE BASKET."CustomerID" = p\_customer;

END TOTALPRICEBASKET;

CREATE OR REPLACE PROCEDURE RATINGPRODUCT(p\_product IN "REVIEW"."ProductID"%TYPE,

p\_rating OUT INT) IS

BEGIN

SELECT CEIL(AVG(REVIEW."Rating")) INTO p\_rating FROM REVIEW WHERE REVIEW."ProductID" = p\_product;

END;

CREATE OR REPLACE PROCEDURE SHOWREVIEW(p\_productid IN PRODUCT."ProductID"%TYPE) AS

BEGIN

FOR rec IN (

SELECT PRODUCT."Name", CUSTOMER."FirstName", REVIEW."Description", REVIEW."Rating", REVIEW."Date"

FROM REVIEW INNER JOIN PRODUCT ON REVIEW."ProductID" = PRODUCT."ProductID"

INNER JOIN CUSTOMER ON REVIEW."CustomerID" = CUSTOMER."CustomerID"

WHERE REVIEW."ProductID" = p\_productid) LOOP

DBMS\_OUTPUT.enable;

DBMS\_OUTPUT.put\_line( rec."Name" || ' ' || rec."FirstName" || ' ' || rec."Description" || ' ' ||

rec."Rating" || ' ' || rec."Date");

END LOOP;

END SHOWREVIEW;

SHOWBASKET(p\_customerid IN CUSTOMER."CustomerID"%TYPE) AS

BEGIN

FOR rec IN (

SELECT PRODUCT."Name", CUSTOMER."FirstName", BASKET."Amount", BASKET."Price"

FROM BASKET INNER JOIN PRODUCT ON BASKET."ProductID" = PRODUCT."ProductID"

INNER JOIN CUSTOMER ON BASKET."CustomerID" = CUSTOMER."CustomerID"

WHERE BASKET."CustomerID" = p\_customerid)

LOOP

DBMS\_OUTPUT.enable;

DBMS\_OUTPUT.put\_line( rec."Name" || ' ' || rec."FirstName" || ' ' || rec."Amount" || ' ' ||

rec."Price");

END LOOP;

END SHOWBASKET;

CREATE OR REPLACE PROCEDURE SHOWFAVORITE(p\_customerid IN CUSTOMER."CustomerID"%TYPE) AS

BEGIN

FOR rec IN (

SELECT PRODUCT."Name", CUSTOMER."FirstName"

FROM FAVORITE INNER JOIN PRODUCT ON FAVORITE."ProductID" = PRODUCT."ProductID"

INNER JOIN CUSTOMER ON FAVORITE."CustomerID" = CUSTOMER."CustomerID"

WHERE FAVORITE."CustomerID" = p\_customerid) LOOP

DBMS\_OUTPUT.enable;

DBMS\_OUTPUT.put\_line( rec."Name" || ' ' || rec."FirstName");

END LOOP;

END SHOWFAVORITE;

CREATE OR REPLACE PROCEDURE SHOWHISTORY(p\_customerid IN CUSTOMER."CustomerID"%TYPE) AS

BEGIN

FOR rec IN (

SELECT PRODUCT."Name", CUSTOMER."FirstName", ORDERHISTORY."Status", ORDERHISTORY."Date", ORDERHISTORY."Amount", ORDERHISTORY."Price"

FROM ORDERHISTORY INNER JOIN PRODUCT ON ORDERHISTORY."ProductID" = PRODUCT."ProductID"

INNER JOIN CUSTOMER ON ORDERHISTORY."CustomerID" = CUSTOMER."CustomerID"

WHERE ORDERHISTORY."CustomerID" = p\_customerid)

LOOP

DBMS\_OUTPUT.enable;

DBMS\_OUTPUT.put\_line( rec."Name" || ' ' || rec."FirstName" || ' ' || rec."Status" || ' ' || rec."Date" || ' ' || rec."Amount" || ' ' || rec."Price");

END LOOP;

END SHOWHISTORY;

CREATE OR REPLACE PROCEDURE SHOWOPTIONPRODUCT(p\_productid IN PRODUCT."ProductID"%TYPE) AS

BEGIN FOR rec IN (

SELECT APPUSER."PRODUCT"."Name", APPUSER."OPTIONTYPE"."OptionTypeName", APPUSER."OPTION"."OptionName",

APPUSER."PRODUCTOPTION"."Value", APPUSER."PRODUCTOPTION"."Unit"

FROM APPUSER."PRODUCTOPTION" INNER JOIN APPUSER."PRODUCT" ON APPUSER."PRODUCTOPTION"."ProductID" = APPUSER."PRODUCT"."ProductID"

INNER JOIN APPUSER."OPTION" ON APPUSER."PRODUCTOPTION"."OptionID" = APPUSER."OPTION"."OptionID"

INNER JOIN APPUSER."OPTIONTYPE" ON APPUSER."OPTION"."OptionTypeID" = APPUSER."OPTIONTYPE"."OptionTypeID"

WHERE APPUSER."PRODUCTOPTION"."ProductID" = p\_productid)

LOOP

DBMS\_OUTPUT.enable;

DBMS\_OUTPUT.put\_line( rec."Name" || ' ' || rec."OptionTypeName" || ' ' || rec."OptionName" || ' ' || rec."Value" || ' ' ||

rec."Unit");

END LOOP;

END SHOWOPTIONPRODUCT;

CREATE OR REPLACE DIRECTORY XML\_DIR as 'D:\CP\XML';

CREATE OR REPLACE PROCEDURE TABLE\_TO\_XML\_FILE(table\_name IN VARCHAR2) AS

ctx DBMS\_XMLGEN.CTXHANDLE;

clb CLOB;

file UTL\_FILE.FILE\_TYPE;

buffer VARCHAR2(32767);

position PLS\_INTEGER := 1;

chars PLS\_INTEGER := 32767; BEGIN

ctx := DBMS\_XMLGEN.NEWCONTEXT('SELECT \* FROM "' || table\_name || '"');

DBMS\_XMLGEN.SETROWSETTAG(ctx, 'RECORDS');

DBMS\_XMLGEN.SETROWTAG(ctx, 'RECORD');

SELECT XMLSERIALIZE(DOCUMENT XMLELEMENT("XML", XMLELEMENT(EVALNAME(table\_name), DBMS\_XMLGEN.GETXMLTYPE(ctx))) INDENT SIZE = 2)

INTO clb FROM dual;

DBMS\_XMLGEN.CLOSECONTEXT(ctx);

file := UTL\_FILE.FOPEN('XML\_DIR', table\_name || '.xml', 'w', 32767);

WHILE position < DBMS\_LOB.GETLENGTH(clb)

LOOP DBMS\_LOB.READ(clb, chars, position, buffer);

UTL\_FILE.PUT(file, buffer);

UTL\_FILE.FFLUSH(file);

position := position + chars;

END LOOP;

UTL\_FILE.FCLOSE(file);

COMMIT;

END TABLE\_TO\_XML\_FILE;

DECLARE

xml\_file BFILE;

xml\_data CLOB;

BEGIN

xml\_file := BFILENAME ('XML\_DIR', 'USERS.xml');

DBMS\_LOB.createtemporary (xml\_data, TRUE, DBMS\_LOB.SESSION);

DBMS\_LOB.fileopen (xml\_file, DBMS\_LOB.file\_readonly);

DBMS\_LOB.loadfromfile (xml\_data, xml\_file, DBMS\_LOB.getlength(xml\_file));

DBMS\_LOB.fileclose (xml\_file);

INSERT INTO USERS (USERS."Username",USERS."Password",USERS."Role") SELECT ExtractValue(Value(x),'//Username') as login,

ExtractValue(Value(x),'//Password') as password, ExtractValue(Value(x),'//Role') as user\_type

FROM TABLE(XMLSequence(Extract(XMLType(xml\_data),'/XML/USERS/RECORDS/RECORD'))) x;

DBMS\_OUTPUT.PUT\_LINE( SQL%ROWCOUNT || ' rows inserted.' );

DBMS\_LOB.freetemporary (xml\_data);

COMMIT; END;

SELECT COUNT("Name") FROM PRODUCT;

DECLARE

i NUMBER := 1;

BEGIN

LOOP

INSERT INTO PRODUCT(PRODUCT."Name", PRODUCT."Manufacturer", PRODUCT."Price") VALUES ('name', 'brand', 1000);

COMMIT;

i := i + 1;

EXIT WHEN (i > 100000);

END LOOP;

END;

# Приложение Г

CREATE sequence "USERS\_USERID\_SEQ";

CREATE sequence "CUSTOMER\_CUSTOMERID\_SEQ";

CREATE sequence "ADDRESS\_ADDRESSID\_SEQ";

CREATE sequence "PRODUCT\_PRODUCTID\_SEQ";

CREATE sequence "OPTIONTYPE\_OPTIONTYPEID\_SEQ";

CREATE sequence "PRODUCTOPTION\_PRODUCTOPTIONID\_SEQ";

CREATE sequence "BASKET\_BASKETID\_SEQ";

CREATE sequence "FAVORITE\_FAVORITEID\_SEQ";

CREATE sequence "ORDERHISTORY\_ORDERHISTORYID\_SEQ";

CREATE sequence "REVIEW\_REVIEWID\_SEQ";

CREATE sequence "PRODUCTCOMPRASION\_PRODUCTCOMPARISONID\_SEQ";

CREATE trigger "BI\_USERS\_USERID"

before insert on "USERS"

for each row

begin

select "USERS\_USERID\_SEQ".nextval into :NEW."UserID" from dual;

end;

CREATE trigger "BI\_CUSTOMER\_CUSTOMERID"

before insert on "CUSTOMER"

for each row

begin

select "CUSTOMER\_CUSTOMERID\_SEQ".nextval into :NEW."CustomerID" from dual;

end;

CREATE trigger "BI\_ADDRESS\_ADDRESSID"

before insert on "ADDRESS"

for each row

begin

select "ADDRESS\_ADDRESSID\_SEQ".nextval into :NEW."AddressID" from dual;

end;

CREATE trigger "BI\_PRODUCT\_PRODUCTID"

before insert on "PRODUCT"

for each row

begin

select "PRODUCT\_PRODUCTID\_SEQ".nextval into :NEW."ProductID" from dual;

end;

CREATE trigger "BI\_OPTIONTYPE\_OPTIONTYPEID"

before insert on "OPTIONTYPE"

for each row

begin

select "OPTIONTYPE\_OPTIONTYPEID\_SEQ".nextval into :NEW."OptionTypeID" from dual;

end;

CREATE trigger "BI\_PRODUCTOPTION\_PRODUCTOPTIONID"

before insert on "PRODUCTOPTION"

for each row

begin

select "PRODUCTOPTION\_PRODUCTOPTIONID\_SEQ".nextval into :NEW."ProductOptionID" from dual;

end;

CREATE trigger "BI\_BASKET\_BASKETID"

before insert on "BASKET"

for each row

begin

select "BASKET\_BASKETID\_SEQ".nextval into :NEW."BasketID" from dual;

end;

CREATE trigger "BI\_FAVORITE\_FAVORITEID"

before insert on "FAVORITE"

for each row

begin

select "FAVORITE\_FAVORITEID\_SEQ".nextval into :NEW."FavoriteID" from dual;

end;

CREATE trigger "BI\_ORDERHISTORY\_ORDERHISTORYID"

before insert on "ORDERHISTORY"

for each row

begin

select "ORDERHISTORY\_ORDERHISTORYID\_SEQ".nextval into :NEW."OrderHistoryID" from dual;

end;

CREATE trigger "BI\_REVIEW\_REVIEWID"

before insert on "REVIEW"

for each row

begin

select "REVIEW\_REVIEWID\_SEQ".nextval into :NEW."ReviewID" from dual;

end;

CREATE trigger "BI\_PRODUCTCOMPRASION\_PRODUCTCOMPARISONID"

before insert on "PRODUCTCOMPRASION"

for each row

begin

select "PRODUCTCOMPRASION\_PRODUCTCOMPARISONID\_SEQ".nextval into :NEW."ProductComparisonID" from dual;

end;