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

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОЙ РАБОТЕ НА ТЕМУ:**

«Реализация базы данных магазина настольных игр с применением технологии шифрования и маскирования в БД»

Выполнил студент Романчук А. М.

(Ф.И.О.)

Руководитель работы асс. Нистюк О. А.

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

И.о. зав. кафедрой ст. преп. Блинова Е. А.

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

Курсовая работа защищена с оценкой

Минск 2023

**Содержание**

[Введение 5](#_Toc154132813)

[1 Постановка задачи 6](#_Toc154132814)

[1.1 Обзор аналогичных решений 6](#_Toc154132815)

[1.2 Спецификация требований 7](#_Toc154132816)

[2 Проектирование базы данных 8](#_Toc154132817)

[3 Разработка объектов базы данных 11](#_Toc154132818)

[3.1 Разработка таблиц 11](#_Toc154132819)

[3.2 Разработка хранимых процедур и функций 11](#_Toc154132820)

[3.3 Разработка пользователей 12](#_Toc154132821)

[4 Описание процедур импорта и экспорта 13](#_Toc154132822)

[5 Тестирование производительности 15](#_Toc154132823)

[6 Описание технологии и её применения в базе данных 17](#_Toc154132824)

[7 Краткое описание приложения для демонстрации 20](#_Toc154132825)

[8 Руководство пользователя 21](#_Toc154132826)

[Заключение 26](#_Toc154132827)

[Список используемых источников 27](#_Toc154132828)

[Приложение А 28](#_Toc154132829)

[Приложение Б 29](#_Toc154132830)

[Приложение В 30](#_Toc154132831)

[Приложение Г 34](#_Toc154132832)

[Приложение Д 61](#_Toc154132833)

# **Введение**

Основой современных информационных технологий являются данные. Работа с ними может быть оптимизирована при помощи их структуризации. Для этой цели широко используются базы данных (БД) и системы управления базами данных (СУБД).

База данных – это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе [1].

Одним из актуальных примеров широкого использования баз данных может служить приложение интернет-магазина настольных игр, где необходимо хранить информацию о содержимом каталога, зарегистрированных пользователях, их корзинах, избранном, заказах.

Целью данной курсовой работы является разработка базы данных для магазина настольных игр с использованием технологии шифрования и маскирования. База данных реализуется в СУБД Oracle Database, используется интегрированная среда разработки Oracle SQL Developer.

С целью демонстрации работы БД разрабатывается веб-приложение для интернет-магазина настольных игр на JavaScript с использованием платформы Node.js и фреймворка Express. Express – это минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для веб-приложений [2]. Используется текстовый редактор Visual Studio Code.

Веб-приложение предоставляет разные возможности в зависимости от используемой роли. Гость может просматривать каталог товаров (настольных игр), выполнять поиск и фильтрацию по нему, аутентифицироваться в приложении. Пользователь (клиент) может также управлять личной корзиной, коллекцией избранных товаров, оформлять заказы, просматривать историю заказов, изменять личные данные аккаунта. Администратор может добавлять товары в каталог, изменять информацию о них, изменять статусы заказов клиентов, добавлять новых администраторов. Задачей приложения является обеспечение этих возможностей.

Для достижения цели курсовой работы необходимо выполнить следующие задачи:

* проанализировать существующие решения, разработать спецификацию требований к БД;
* выполнить проектирование модели БД;
* реализовать БД с применением технологии шифрования и маскирования;
* выполнить тестирование производительности;
* спроектировать и реализовать приложение для демонстрации;
* разработать руководство пользователя.

Выполнение каждой из этих задач описано в соответствующих разделах данной пояснительной записки.

# **1 Постановка задачи**

## **Обзор аналогичных решений**

В связи с расширением возможностей и удобства использования интернета, всё больше людей предпочитают заказывать настольные игры в онлайн-магазинах, используя их веб-сайты, мобильные и десктопные приложения. Для составления подробной спецификации требований к разрабатываемому приложению и определения структуры БД далее рассматриваются веб-сайты интернет-магазинов настольных игр Hobby Games (hobbygames.by) и Nastolki.by (nastolki.by).

Вид страницы сайта hobbygames.by при выборе настольной игры из каталога представлен на рисунке 1.1.

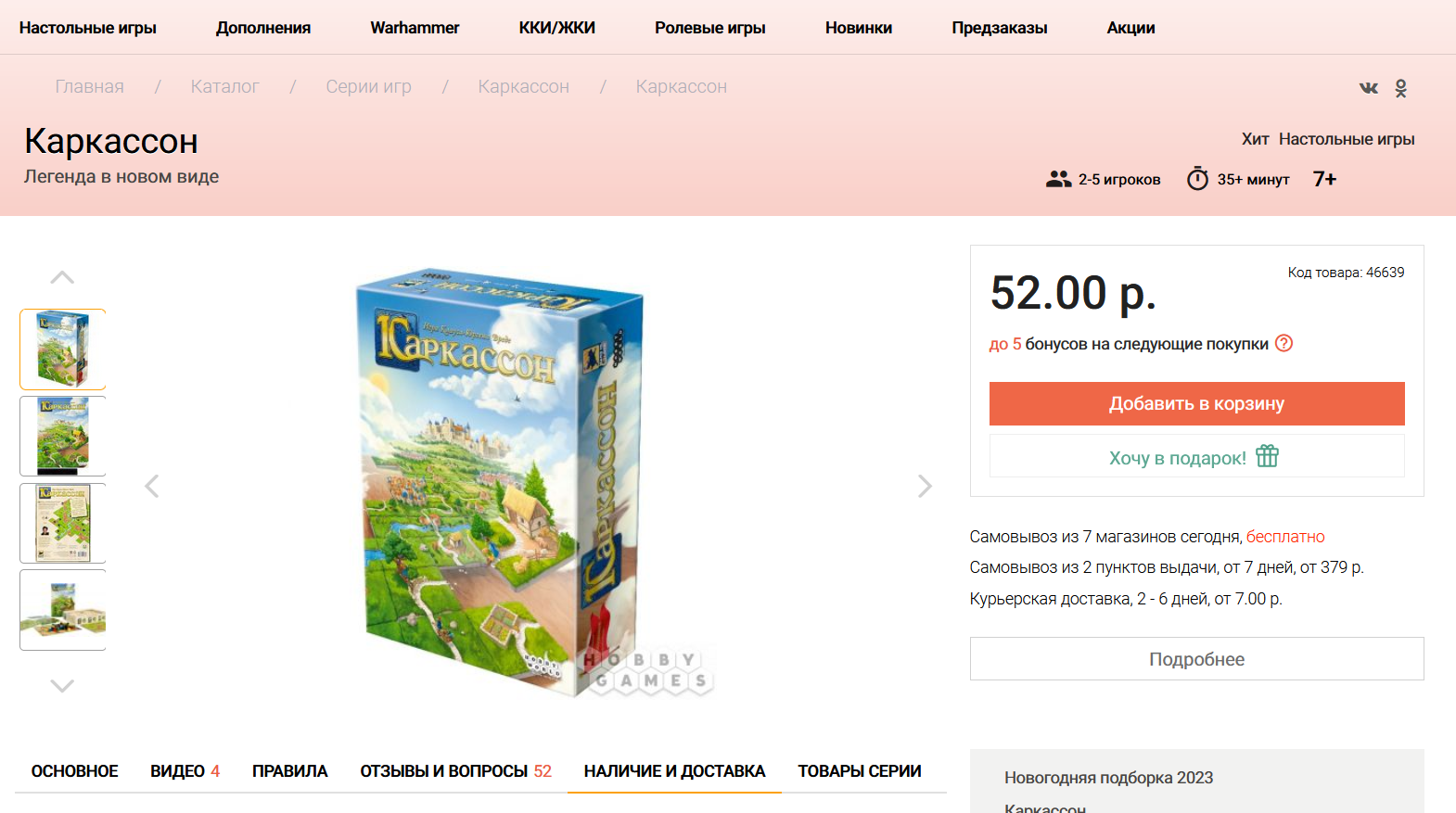


Рисунок 1.1 – Страница сайта hobbygames.by с информацией о настольной игре.

Сайт позволяет просматривать каталог товаров, выполнять по нему поиск, фильтровать его по категориям, диапазону цен, количеству игроков, производителю, а также предлагаются различные варианты сортировки, такие как по возрастанию или убыванию цены, названию, популярности и другие.

Для настольных игр указана следующая информация: название, описание, изображения, цена, средняя продолжительность игры, возрастное ограничение, категория, правила, видео-обзоры, отзывы, производитель, год выпуска и другое.

Для заказа необходимо сначала добавить товары в корзину. Заказ могут оформлять как авторизованные, так и неавторизованные пользователи. Для авторизованных пользователей есть система бонусов и заданий.

Полнота предоставляемой информации является большим преимуществом данного сайта. Недостатком же является отсутствие возможности добавлять товары в коллекцию избранного за неимением таковой.

Сайт nastolki.by позволяет просматривать каталог настольных игр, фильтровать по ключевым фразам, цене, количеству игроков, категориям, производителю. Есть возможность сортировки по цене и названию.

Для товаров указана цена, общая информация, включающая количество игроков, время партии, возрастное ограничение. Также для товаров указывается скидка. Вид сайта nastolki.by при переходе к конкретному товару представлен на рисунке 1.2. Кроме того, сайт позволяет управлять коллекцией избранного, корзиной; оформить заказ. Недостатком является нестабильная работа клиентской части сайта и меньший набор предоставляемой информации.



Рисунок 1.2 – Вид страницы сайта nastolki.by с информацией о товаре.

Таким образом, оба сайта предоставляют широкие возможности манипуляции с каталогом, управления корзиной, позволяют оформить заказ и изменить личную информацию пользователя. Подробная информация о товарах и широкие функциональные возможности могут выделить разрабатываемое приложение.

## **Спецификация требований**

База данных должна быть реализована в СУБД Oracle.

Доступ к данным должен осуществляться только через соответствующие процедуры. Должна быть обеспечена возможность импорта и экспорта данных из XML-файлов. Необходимо выполнить тестирование производительности базы данных на таблице, содержащей не менее 100 000 строк.

Необходимо применить технологии шифрования и маскирования в БД.

Для работы с базой данных должны поддерживаться следующие роли: администратор, пользователь (клиент), гость.

Всем ролям должен быть доступен просмотр каталога, поиск в нём, фильтрация и сортировка.

Клиент должен иметь возможность добавлять товары в корзину и исключать их из неё, добавлять товары в избранное и исключать их из него, оформлять заказ, просматривать историю своих заказов, изменять личную информацию аккаунта.

Администратор должен иметь возможность добавлять новые товары в каталог, изменять существующие, регулировать их доступность; изменять статусы заказов пользователей; добавлять новых администраторов.

Разрабатываемое в рамках курсовой работы приложение должно демонстрировать работоспособность БД, позволяя выполнять основные определённые выше операции.

2 Проектирование базы данных

База данных BGAMESSTORE включает в себя десять таблиц, хранящих информацию об аккаунтах, товарах, корзинах, заказах, избранном и иную. Записи во всех таблицах имеют собственные идентификаторы, что позволяет их легко связывать между собой посредством внешних ключей.

Таблица Roles содержит список возможных ролей пользователей. Описание её столбцов представлено в таблице 2.1.

Таблица 2.1 – Описание столбцов таблицы Roles

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| id | int | Идентификатор роли. |
| role\_name | varchar2(255) | Название роли. |

Таблица Accounts содержит информацию об аккаунтах пользователей. Описание её столбцов представлено в таблице 2.2.

Таблица 2.2 – Описание столбцов таблицы Accounts

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| id | int | Идентификатор аккаунта. |
| role\_id | int | Идентификатор роли пользователя. |
| login | varchar2(255) | Логин. |
| password | raw(2000) | Пароль. |
| is\_active | number(1) | Флаг активности аккаунта. |
| fullname | raw(2000) | Полное имя пользователя. |
| email | raw(2000) | Электронная почта пользователя. |

Таблица Clients содержит специфическую информацию о клиентах. Описание её столбцов представлено в таблице 2.3.

Таблица 2.3 – Описание столбцов таблицы Clients

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| id | int | Идентификатор клиента. |
| account\_id | int | Идентификатор аккаунта клиента. |
| address | raw(2000) | Адрес клиента. |

Таблица AccountsKeys содержит ключи безопасности пользователей, которые используются для шифрования и дешифрования их персональных данных (полного имени, электронной почты, адреса и др.). Описание её столбцов представлено в таблице 2.4.

Таблица 2.4 – Описание столбцов таблицы AccountsKeys

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| id | int | Идентификатор ключа. |
| account\_id | int | Идентификатор аккаунта. |
| key | raw(128) | Ключ безопасности. |

Таблица Items содержит информацию о товарах (настольных играх) в каталоге. Описание её столбцов представлено в таблице 2.5.

Таблица 2.5 – Описание столбцов таблицы Items

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| id | int | Идентификатор товара. |
| title | varchar2(255) | Название товара. |
| price | decimal(10, 2) | Цена товара. |
| count | int | Количество товара (штук). |
| image | varchar2(511) | Путь к изображению товара. |
| description | varchar2(2047) | Описание товара. |
| category | varchar2(255) | Категория товара. |
| publisher | varchar2(225) | Издатель товара. |
| year | number(4) | Год издания. |
| min\_players | int | Минимальное количество игроков. |
| max\_players | int | Максимальное количество игроков. |
| play\_time | int | Среднее время игры (минут). |
| player\_min\_age | int | Ограничение по возрасту. |
| is\_available | number(1) | Флаг доступности товара. |

Таблица Carts содержит информацию о корзинах клиентов. Описание её столбцов представлено в таблице 2.6.

Таблица 2.6 – Описание столбцов таблицы Carts

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| id | int | Идентификатор записи о товаре в корзине. |
| account\_id | int | Идентификатор аккаунта. |
| item\_id | int | Идентификатор товара. |
| count | int | Количество товара в корзине. |

Таблица OrderStatuses содержит список возможных статусов заказа. Описание её столбцов представлено в таблице 2.7.

Таблица 2.7 – Описание столбцов таблицы OrderStatuses

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| id | int | Идентификатор статуса. |
| os\_name | varchar2(255) | Название статуса. |

Таблица Orders содержит информацию о заказах клиентов. Описание её столбцов представлено в таблице 2.8.

Таблица 2.8 – Описание столбцов таблицы Orders

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| id | int | Идентификатор записи о заказе. |
| account\_id | int | Идентификатор клиента. |
| order\_number | varchar2(255) | Номер заказа. |
| order\_date | date | Дата оформления заказа (UTC). |
| status\_id | int | Идентификатор статуса заказа. |
| order\_comment | raw(2000) | Комментарий к заказу. |
| card\_number | raw(2000) | Номер карты. |
| client\_fullname | raw(2000) | Полное имя клиента. |
| client\_email | raw(2000) | Электронная почта клиента. |
| client\_address | raw(2000) | Адрес клиента (адрес доставки). |

Таблица OrderedItems содержит информацию о заказанных товарах. Описание её столбцов представлено в таблице 2.9.

Таблица 2.9 – Описание столбцов таблицы OrderedItems

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| id | int | Идентификатор записи о заказанном товаре. |
| order\_id | int | Идентификатор заказа. |
| item\_id | int | Идентификатор заказанного товара. |
| price | decimal | Цена товара на момент оформления заказа. |
| count | int | Количество экземпляров товара в заказе. |

Таблица Favourites содержит информацию об избранных товарах. Описание её столбцов представлено в таблице 2.10.

Таблица 2.10 – Описание столбцов таблицы Favourites

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| id | int | Идентификатор записи об избранном товаре. |
| account\_id | int | Идентификатор аккаунта. |
| item\_id | int | Идентификатор избранного товара. |

Таблицы Roles и OrderStatuses выполняют роль перечислений.

Модель базы данных BGAMESSTORE представлена в приложении А.

Для чёткого распределения привилегий между пользователями БД используются роли: гость, пользователь (клиент), администратор.

Диаграмма вариантов использования для ролей гостя, пользователя (клиента) и администратора представлена в приложении Б.

3 Разработка объектов базы данных

**3.1 Разработка таблиц**

После создания подключаемой базы данных было создано постоянное табличное пространство BGS\_TS. В нём от имени администратора PDB (пользователь БД SYSADMIN) создаются таблицы.

Все таблицы имеют столбец идентификатора записи (id) типа int, который автоматически инкрементируется при добавлении новых строк. Для этого для каждой таблицы создана своя последовательность (sequence). Столбец идентификатора выбран в качестве первичного ключа.

Для хранения целых чисел используется тип int, для хранения цен – decimal(10, 2), для флагов (доступности товара, активности аккаунта) – number(1). Для хранения строковых значений используется тип varchar2. Для хранения зашифрованных и хешированных значений используется тип raw(2000), для хранения ключа безопасности – raw(128). Пример создания таблицы Favourites представлен в листинге 3.1.

-- таблица с товарами, добавленными пользователями в избранное

create sequence favourites\_id\_seq;

create table Favourites (

id int default favourites\_id\_seq.nextval not null primary key,

account\_id int not null references Accounts(id) on delete cascade,

item\_id int not null references Items(id) on delete cascade

) tablespace BGS\_TS;

Листинг 3.1 – Создание таблицы Favourites

Когда все таблицы созданы, в таблицы Roles и OrderStatuses заносятся начальные значения (список ролей и список возможных статусов заказа).

Листинг создания таблиц представлен в приложении В.

3.2 Разработка хранимых процедур и функций

Хранимая процедура – это скомпилированный набор SQL-предложений, сохранённый в базе данных как именованный объект и выполняющийся как единый фрагмент кода [3].

В данной курсовой работе главную роль выполняют процедуры, а функции используются только как вспомогательные для процедур, например функции шифрования и дешифрования текста при помощи алгоритма AES-128.

Некоторые процедуры и функции используют специально созданные для них представления, в которых объединены несколько таблиц или проведена некоторая фильтрация. Листинги представлений следуют за созданием таблиц в приложении В.

Процедуры и функции могут выбрасывать специально определённые исключения, которые перехватываются приложением. Примером является процедура ValidateCart, код которой представлен в листинге 3.2.

-- удостовериться, что корзина товаров может быть заказана

create or replace procedure ValidateCart(

p\_account\_id in Carts.account\_id%type

) is

begin

if CartIsEmpty(p\_account\_id) then

raise\_application\_error(-20002, 'CART\_IS\_EMPTY\_EXCEPTION');

end if;

if CartContainsInvalidItems(p\_account\_id) then

raise\_application\_error(-20003, 'CART\_CONTAINS\_INVALID\_ITEMS\_EXCEPTION');

end if;

exception

when others then

raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);

end;

Листинг 3.2 – Создание процедуры ValidateCart

Листинги всех процедур и функций представлены в приложении Г, в том числе те, которые не используются в демонстрационном приложении.

3.3 Разработка пользователей

При создании PDB был указан пользователь SYSADMIN, от имени которого и были созданы объекты БД. Также были созданы роли администратора, пользователя, гостя: AdminRole, ClientRole, GuestRole. Им даётся право на выполнение определённых процедур и установление соединения. Код создания роли гостя представлен в листинге 3.3.

-- роль гостя

create role GuestRole;

grant create session to GuestRole;

--&

grant execute on GetCatalog to GuestRole;

grant execute on GetItemInfo to GuestRole;

--&

grant execute on SignUpAsClient to GuestRole;

grant execute on SignIn to GuestRole;

Листинг 3.3 – Создание роли гостя

Наконец, вызывается процедура CreateNewAdminAccount для создания начального аккаунта администратора с логином «admin» и паролем «admin».

Код создания ролей и пользователей представлен в приложении Д.

# **4 Описание процедур импорта и экспорта**

Для экспорта данных в XML-файл и импорта данных из XML-файла были разработаны процедуры ExportItemsToXML, ImportItemsFromXML. Первая процедура записывает элементы каталога в XML-файл, вторая процедура добавляет в каталог элементы из XML-файла. Среди трёх основных ролей только администратор может вызывать эти процедуры. Код процедуры экспорта представлен в листинге 4.1.

-- записать каталог в XML-файл

create or replace procedure ExportItemsToXML (

p\_filename in varchar2

) is

r\_xml\_data clob;

r\_file\_handle utl\_file.file\_type;

begin

select

xmlelement(

"items", xmlagg(xmlelement("item", xmlforest(

id as "id", title as "title", price as "price", count as "count", image as "image", description as "description",

category as "category", publisher as "publisher", year as "year", min\_players as "min\_players", max\_players as "max\_players",

play\_time as "play\_time", player\_min\_age as "player\_min\_age", is\_available as "is\_available"

)))).getClobVal()

into r\_xml\_data

from Items;

r\_file\_handle := utl\_file.fopen('BGAMESSTORE\_XML', p\_filename, 'w', 32767);

utl\_file.put(r\_file\_handle, r\_xml\_data);

utl\_file.fclose(r\_file\_handle);

exception

when others then

raise\_application\_error(-20005, 'EXPORT\_TO\_XML\_EXCEPTION: '||SQLERRM);

end;

Листинг 4.1 – Создание процедуры ExportItemsToXML

Здесь BGAMESSTORE\_XML – это заранее созданная папка, которая содержит (или будет содержать) используемый XML-файл. Файл открывается (или создаётся) при помощи функции utl\_file.fopen, читается при помощи utl.file.get\_line, закрывается на utl\_file.fclose().

Процедура импорта данных из XML-файла используется для добавления товаров в каталог. Код функции ImportItemsFromXML представлен в листинге 4.2.

|  |
| --- |
| -- добавить товары из XML-файла в каталог  create or replace procedure ImportItemsFromXML(  p\_filename in varchar2  ) is  r\_xml\_data clob;  r\_items\_xml xmltype;  r\_file\_handle utl\_file.file\_type;  buffer varchar2(32767);  begin  r\_file\_handle := utl\_file.fopen('BGAMESSTORE\_XML', p\_filename, 'r', 32767);  loop begin  utl\_file.get\_line(r\_file\_handle, buffer);  r\_xml\_data := r\_xml\_data || buffer;  exception when NO\_DATA\_FOUND then exit;  end;  end loop;  utl\_file.fclose(r\_file\_handle);  r\_items\_xml := xmltype(r\_xml\_data);  for item in (select \*  from xmltable('/items/item' passing r\_items\_xml columns  title varchar2(255) path 'title', price decimal(10, 2) path 'price',  count int path 'count', image varchar2(511) path 'image',  description varchar2(2047) path 'description', category varchar2(255) path 'category',  publisher varchar2(255) path 'publisher', year number(4) path 'year',  min\_players int path 'min\_players', max\_players int path 'max\_players',  play\_time int path 'play\_time', player\_min\_age int path 'player\_min\_age',  is\_available number(1) path 'is\_available')) loop  insert into Items (  title, price, count, image, description, category, publisher, year,  min\_players, max\_players, play\_time, player\_min\_age, is\_available) values (  item.title, item.price, item.count, item.image, item.description,  item.category, item.publisher, item.year, item.min\_players, item.max\_players,  item.play\_time, item.player\_min\_age, item.is\_available  );  end loop;  exception  when others then  raise\_application\_error(-20006, 'IMPORT\_FROM\_XML\_EXCEPTION: '||SQLERRM);end; |

Листинг 4.2 – Создание процедуры ImportItemsFromXML

Импорт данных из XML-файла может использоваться для восстановления данных каталога, если они были экспортированы в этот XML-файл и утеряны.

# **5 Тестирование производительности**

Одной из наиболее часто используемых таблиц является таблица Items, содержащая информацию о товарах. На ней и можно провести тестирование производительности.

Сначала в таблицу в цикле добавляются двести тысяч строк. Скрипт для этого с выводом нового количества строк в таблице представлен на рисунке 5.1.

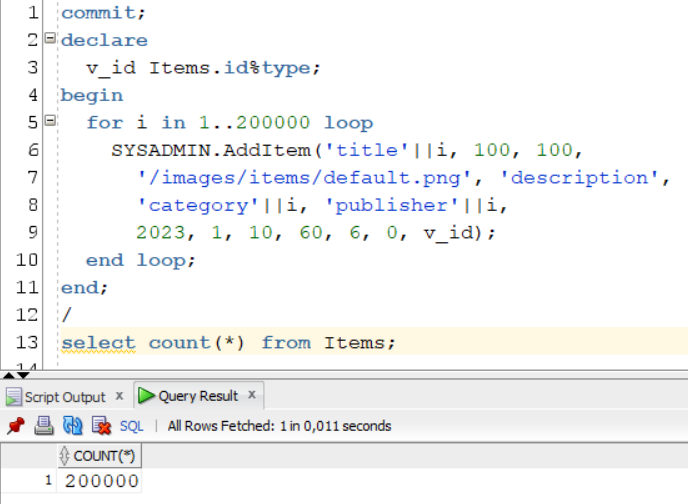


Рисунок 5.1 – Скрипт для добавления строк в тестируемую таблицу с выводом нового количества строк

Затем можно выполнить специально разработанный для тестирования select-запрос, схожий с тем, который используется в процедуре GetCatalog. Запрос и время его выполнения представлены на рисунке 5.2.

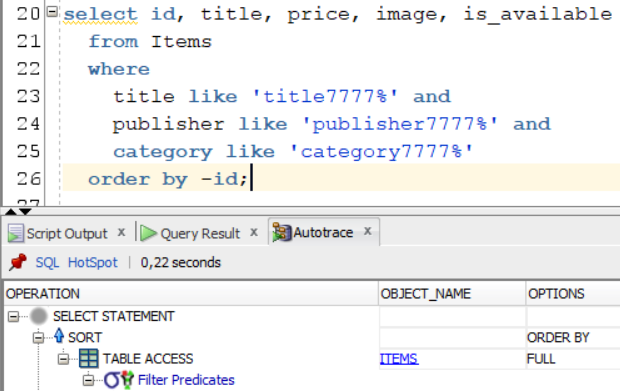


Рисунок 5.2 – Выполнение select-запроса без индекса

Согласно отчёту Autotrace, запрос выполнялся 0,22 секунды, а TABLE ACCESS FULL свидетельствует о чтении каждой строки таблицы для фильтрации. Так как фильтрация осуществляется по столбцам title, publisher и category, то можно добавить индекс на них и сравнить результаты выполнения запроса. Результат выполнения select-запроса с индексом представлен на рисунке 5.3.

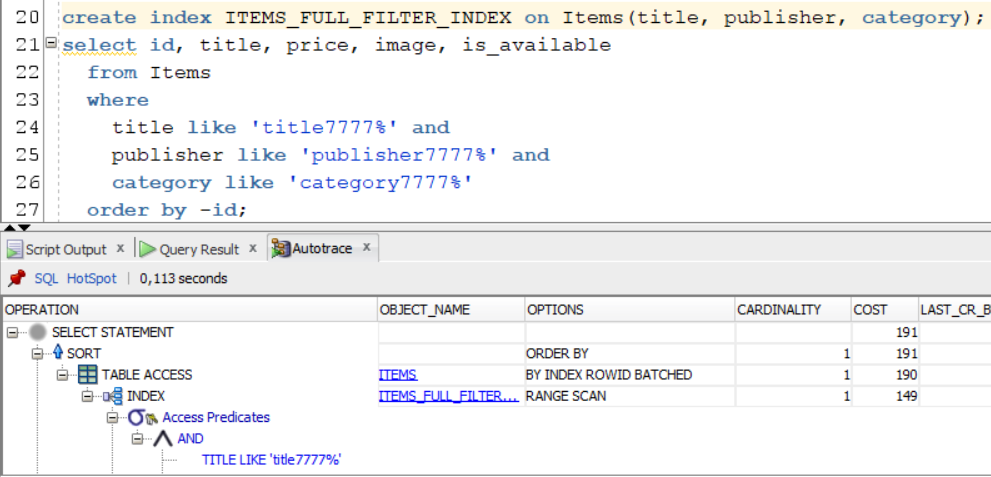


Рисунок 5.3 – Выполнение select-запроса с индексом

Время выполнения запроса сократилось примерно в два раза до 0,113 секунды, а отчёт Autotrace свидетельствует об использовании созданного индекса. Таким образом, индекс по столбцам фильтрации повышает производительность БД (по крайней мере при больших объёмах данных).

# **6 Описание технологии и её применения в базе данных**

**Шифрование – это технология кодирования и раскодирования данных.** Зашифрованные данные – это результат применения алгоритма для кодирования данных с целью сделать их недоступными для чтения. Данные могут быть раскодированы в исходную форму только путём применения специального ключа. Шифрование является важной частью обеспечения безопасности данных, поскольку оно защищает конфиденциальную информацию [4].

Маскирование данных – это методы модификации защищаемых данных, с помощью которых скрывается часть данных в ходе создания копии базы данных. При этом сохраняется общая информационная структура и остаётся доступной та часть данных, которая необходима для работы сотрудников или сторонних специалистов [5]. Нередко используется при тестировании БД.

В курсовой работе широко используется пакет DBMS\_CRYPTO, включающий в себя программы для зашифровывания, расшифровывания, генерации ключа и вычисления хеш-значений.

При добавлении нового аккаунта (в процедуре CreateAccount) для него псевдослучайным генерируется ключ безопасности, который хранится в таблице AccountsKeys. Этот ключ позволяет зашифровывать и расшифровывать конфиденциальные данные пользователей. Для генерации ключа используется функция DBMS\_CRYPTO.randombytes. Используемая схема шифрования показана на рисунке 6.1.

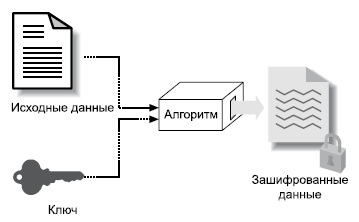


Рисунок 6.1 – Схема шифрования данных

Для шифрования данных (имени пользователя, адреса, номера карты и другой информации) используется алгоритм AES128. Функция, шифрующая текст со сгенерированным ключом по алгоритму AES128, представлена в листинге 6.1. Она использует DBMS\_CRYPTO.encrypt.

-- зашифровать строку ключом (AES128)

create or replace function EncryptAES128(

p\_text in varchar2,

p\_key in raw

) return raw is

begin

return DBMS\_CRYPTO.encrypt(

UTL\_I18N.string\_to\_raw(p\_text, 'AL32UTF8'),

DBMS\_CRYPTO.ENCRYPT\_AES128 + DBMS\_CRYPTO.CHAIN\_CBC + DBMS\_CRYPTO.PAD\_PKCS5,

p\_key

);

exception

when others then

raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);

end;

Листинг 6.1 – Функция EncryptAES128

Для дешифрования используется функция DecryptAES128, которая, в свою очередь, использует DBMS\_CRYPTO.decrypt. Она представлена в листинге 6.2.

-- дешифровать строку ключом (AES128)

create or replace function DecryptAES128(

p\_encrypted in raw,

p\_key in raw

) return varchar2 is

begin

return UTL\_I18N.raw\_to\_char(

DBMS\_CRYPTO.decrypt(

p\_encrypted,

DBMS\_CRYPTO.ENCRYPT\_AES128 + DBMS\_CRYPTO.CHAIN\_CBC + DBMS\_CRYPTO.PAD\_PKCS5,

p\_key

),

'AL32UTF8'

);

exception

when others then

raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);

end;

Листинг 6.2 – Функция DecryptAES128

Ещё одно применение DBMS\_CRYPTO – хэширование. В работе оно применяется для хэширования пароля по алгоритму MD5. При этом используется функция DBMS\_CRYPTO.hash. Код функции HashMD5 представлен в листинге 6.3.

-- хэшировать строку

create or replace function HashMD5(p\_text in varchar2)

return raw is

begin

return DBMS\_CRYPTO.hash(

UTL\_I18N.string\_to\_raw(p\_text, 'AL32UTF8'),

DBMS\_CRYPTO.HASH\_MD5

);

exception

when others then

raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);

end;

Листинг 6.3 – Функция HashMD5

Маскирование используется в функции GetCardNumber для скрытия номера карты пользователя после оформления заказа. Код функции GetCardNumber представлен в листинге 6.4.

-- получить номер карты (возможно, замаскированный)

create or replace function GetCardNumber(

p\_card\_number in raw,

p\_key in AccountsKeys.key%type

) return varchar2 is

r\_card\_number varchar2(255);

r\_is\_admin int;

begin

select case when exists(

select 1 from DBA\_ROLE\_PRIVS

where grantee = USER and granted\_role = 'ADMINROLE'

) then 1 else 0 end into r\_is\_admin from dual;

select DecryptAES128(p\_card\_number, p\_key) into r\_card\_number from dual;

if r\_is\_admin = 1 then

return regexp\_replace(r\_card\_number, '(.{4})', '\1 ');

else

return

regexp\_replace(lpad('\*', length(r\_card\_number) - 4, '\*'), '(.{4})', '\1 ') ||

substr(r\_card\_number, -4);

end if;

exception

when others then

raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);

end;

Листинг 6.4 – Функция GetCardNumber

В результате данного маскирования администратор видит номер карты без маски, а остальные пользователи видят лишь 4 последние цифры, при этом остальные цифры заменяются на символ «\*».

7 Краткое описание приложения для демонстрации

При запуске веб-сервера приложение становится доступно по адресу http://127.0.0.1:3000/. Происходит автоматическое перенаправление на страницу каталога товаров.

Изначальная роль пользователя – гость. Для гостя доступны две вкладки: каталога и аккаунта. На первой можно манипулировать каталогом (просматривать, фильтровать, сортировать), на второй – зарегистрироваться или авторизоваться как существующий пользователь. У разных пользователей не могут повторяться логины.

При входе в приложение как администратор на странице каталога появляется возможность экспорта и импорта данных из XML, а также редактирования товаров каталога. Появляются вкладки с возможностью добавить новый товар в каталог, вкладка управления заказами с возможностью их просмотра и изменения статуса. На вкладке аккаунта администратор может изменить данные аккаунта, добавить нового администратора, а также деактивировать свой аккаунт.

При входе в приложение как пользователь (клиент), на странице каталога появляются кнопки добавления либо исключения товара из корзины и из коллекции избранного, такие же кнопки есть и на собственной странице товара. На вкладке избранного можно переместить все избранные товары в корзину либо очистить избранное. На вкладке корзины можно перейти к оформлению заказа или очистить корзину. На вкладке заказов можно посмотреть историю заказов, заказанные товары. На вкладке аккаунта можно обновить данные аккаунта, клиентскую информацию (адрес) либо деактивировать аккаунт.

# **8 Руководство пользователя**

После запуска веб-сервера и открытия в браузере адреса http://127.0.0.1:3000/ пользователь попадает на главную страницу приложения – страницу каталога товаров, представленную на рисунке 8.1.

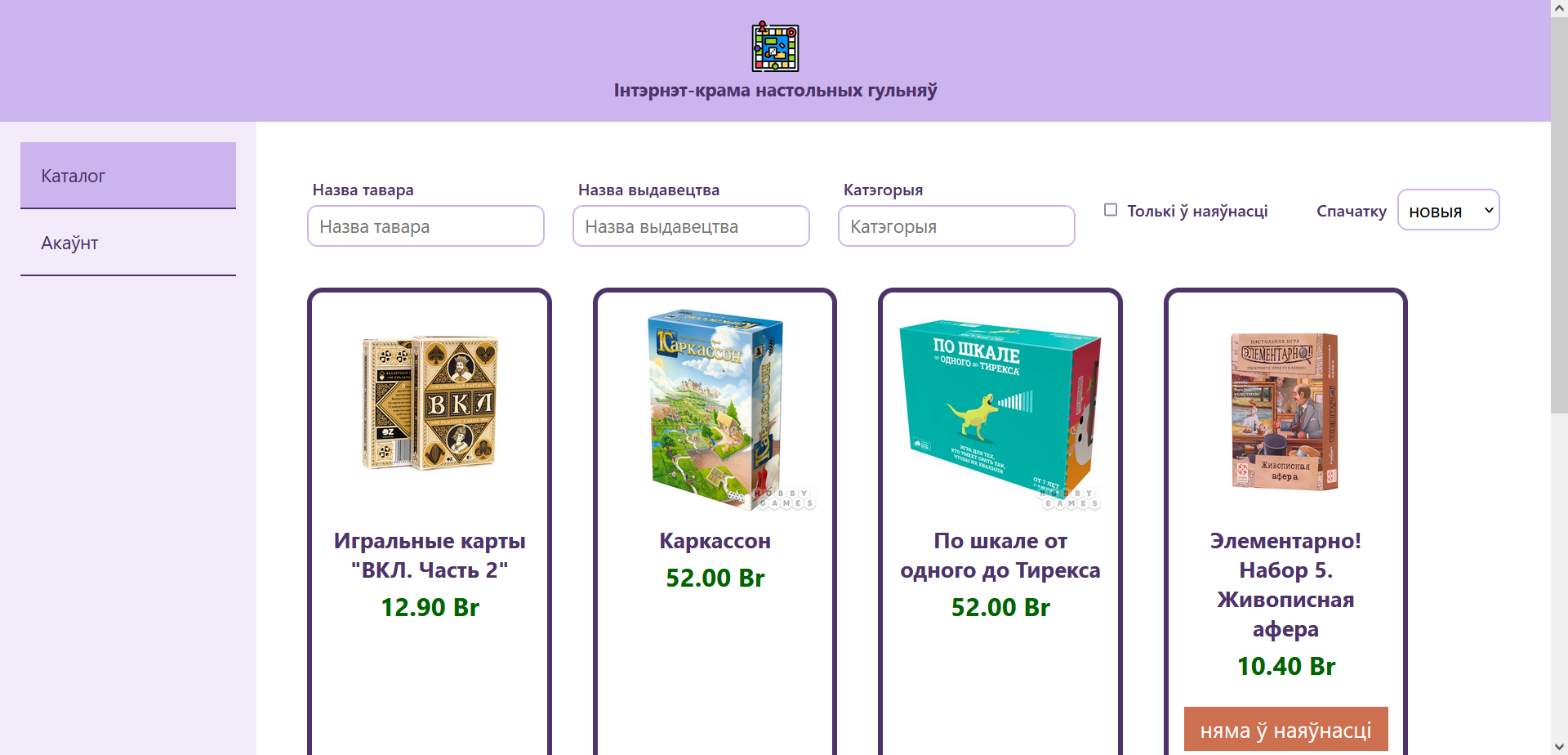


Рисунок 8.1 – Страница каталога для гостя

Каталог можно фильтровать по названию товара, названию издательства, категории, наличию товара на складе. Можно сортировать по дате добавления в каталог и по цене. По нажатию на блок товара происходит переход на страницу товара с подробной информацией о нём.

На вкладке аккаунта можно зарегистрироваться либо войти, если уже есть аккаунт. Страница аккаунта для гостя представлена на рисунке 8.2.

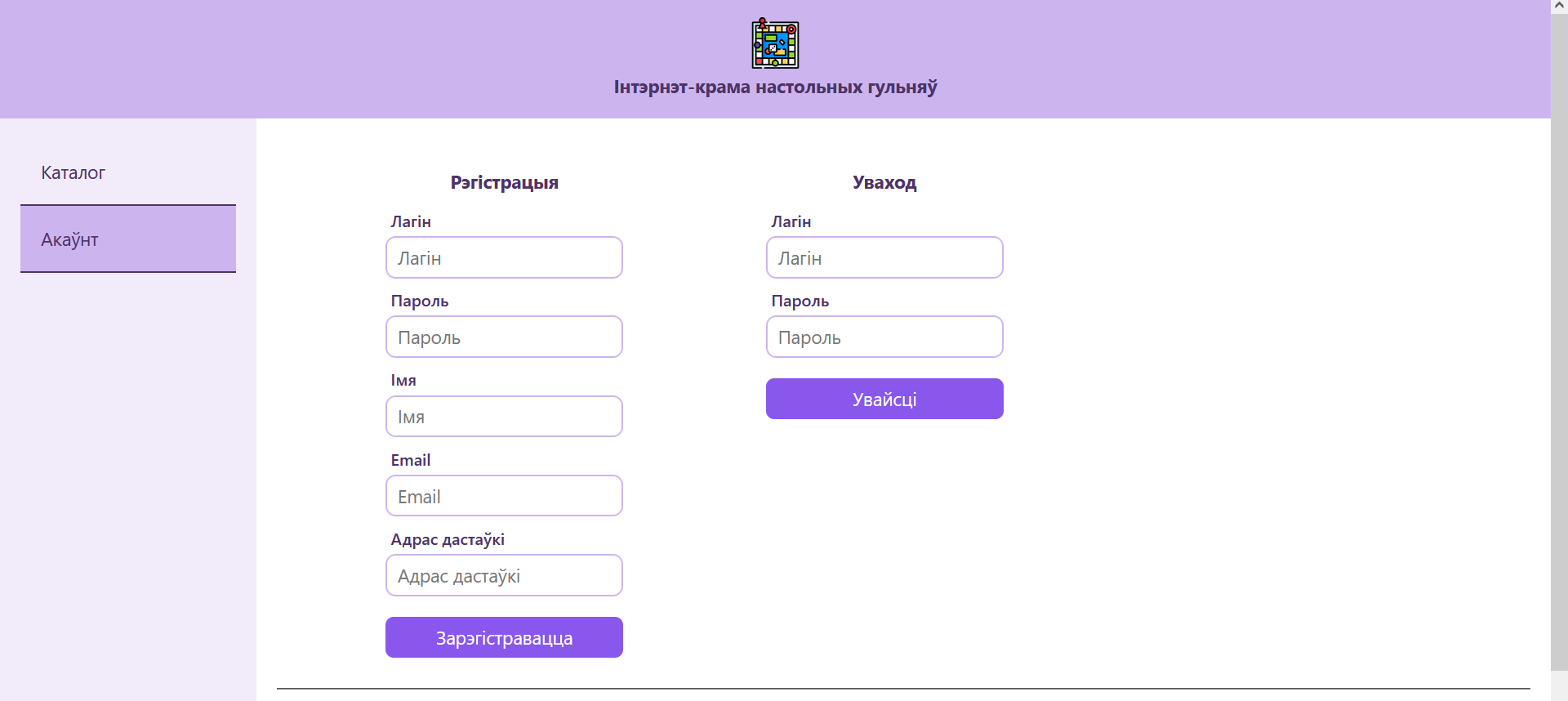


Рисунок 8.2 – Страница аккаунта для гостя

После входа в учётную запись администратора, на странице аккаунта можно изменить данные аккаунта и добавить нового администратора. Также можно выйти из аккаунта либо деактивировать его (фактически удалить). Вид страницы аккаунта для администратора представлен на рисунке 8.3.

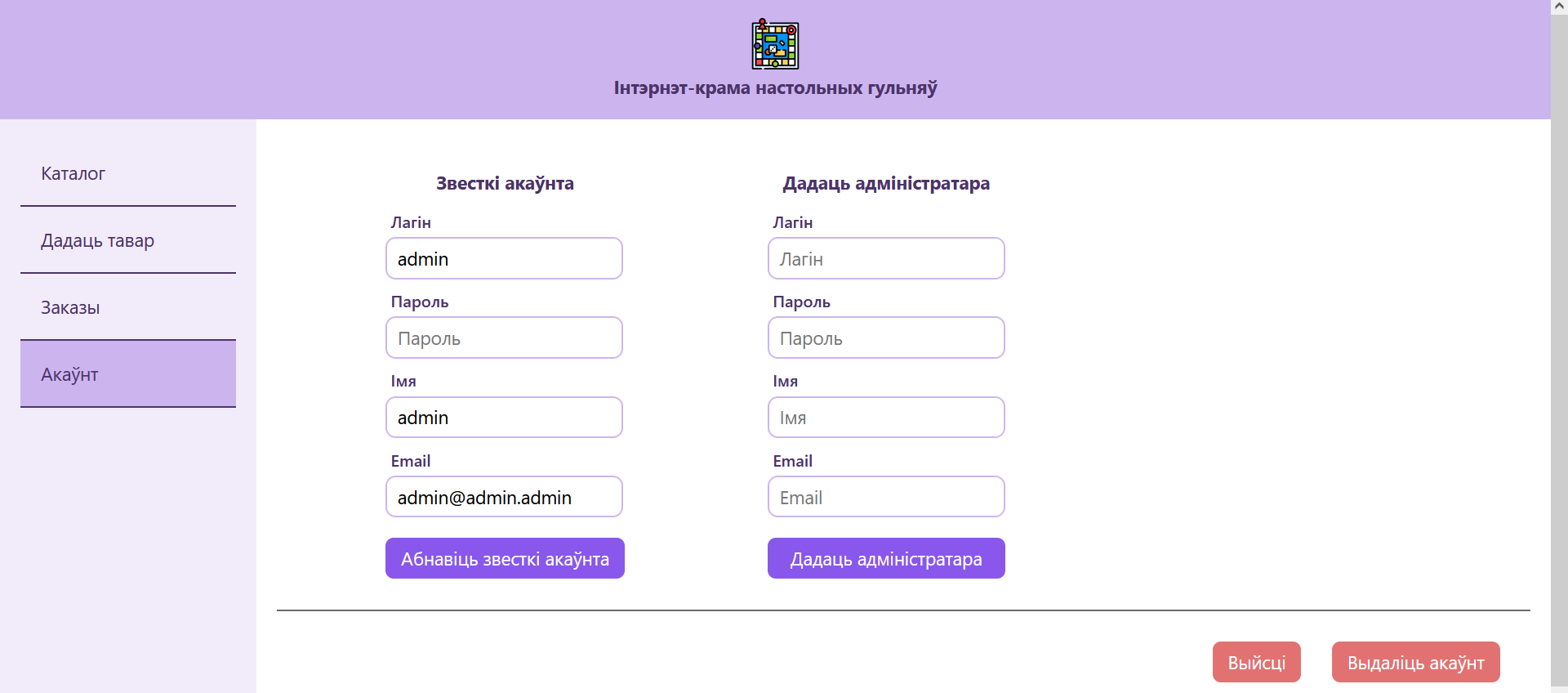


Рисунок 8.3 – Страница аккаунта для администратора

Для администратора изменяется и страница каталога: появляется возможность экспорта товаров в XML-файл и импорта товаров из файла в каталог. Также блоки товаров содержат кнопки для перехода к изменению информации о товаре.

На странице добавления товара в соответствующие поля заносится информация о товаре, можно выбрать путь к файлу с изображением. По нажатию на кнопку добавления товара появляется сообщение об успешном завершении операции, как показано на рисунке 8.4.

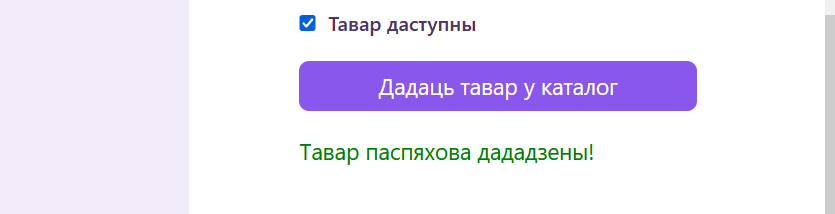


Рисунок 8.4 – Успешное добавление нового товара в каталог

На странице заказов администратор может просматривать информацию о заказах всех клиентов, в том числе список заказанных товаров, а также изменять статус заказа. Вид страницы заказов для администратора представлен на рисунке 8.5.

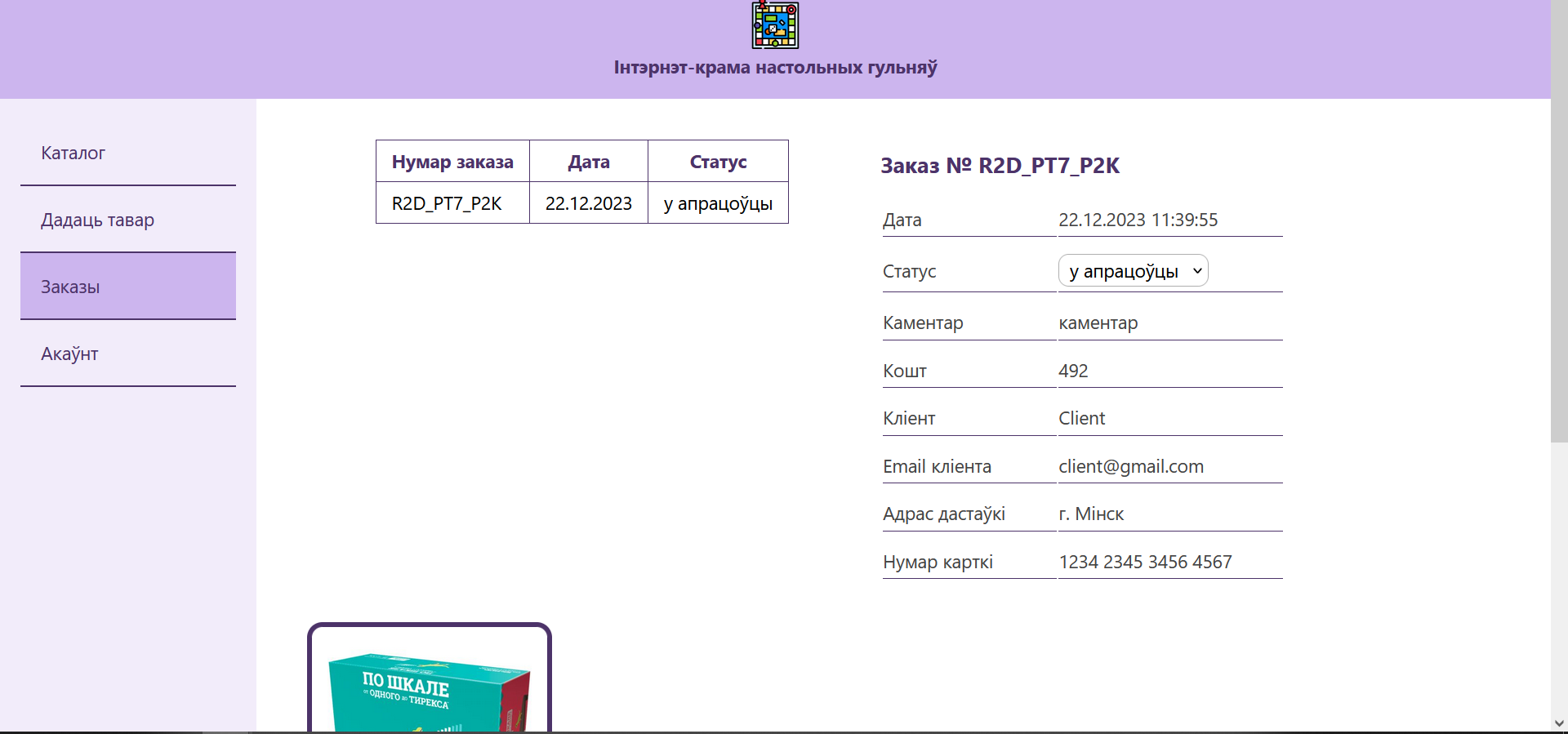


Рисунок 8.5 – Страница заказов для администратора

При входе в аккаунт пользователя (клиента) на странице аккаунта можно изменить данные аккаунта, изменить клиентскую информацию (адрес доставки), выйти из аккаунта и деактивировать его.

На странице каталога клиент может добавлять и исключать товары из корзины и из своей коллекции избранного при нажатии на соответствующие кнопки. Вид страницы каталога для клиента показан на рисунке 8.6.

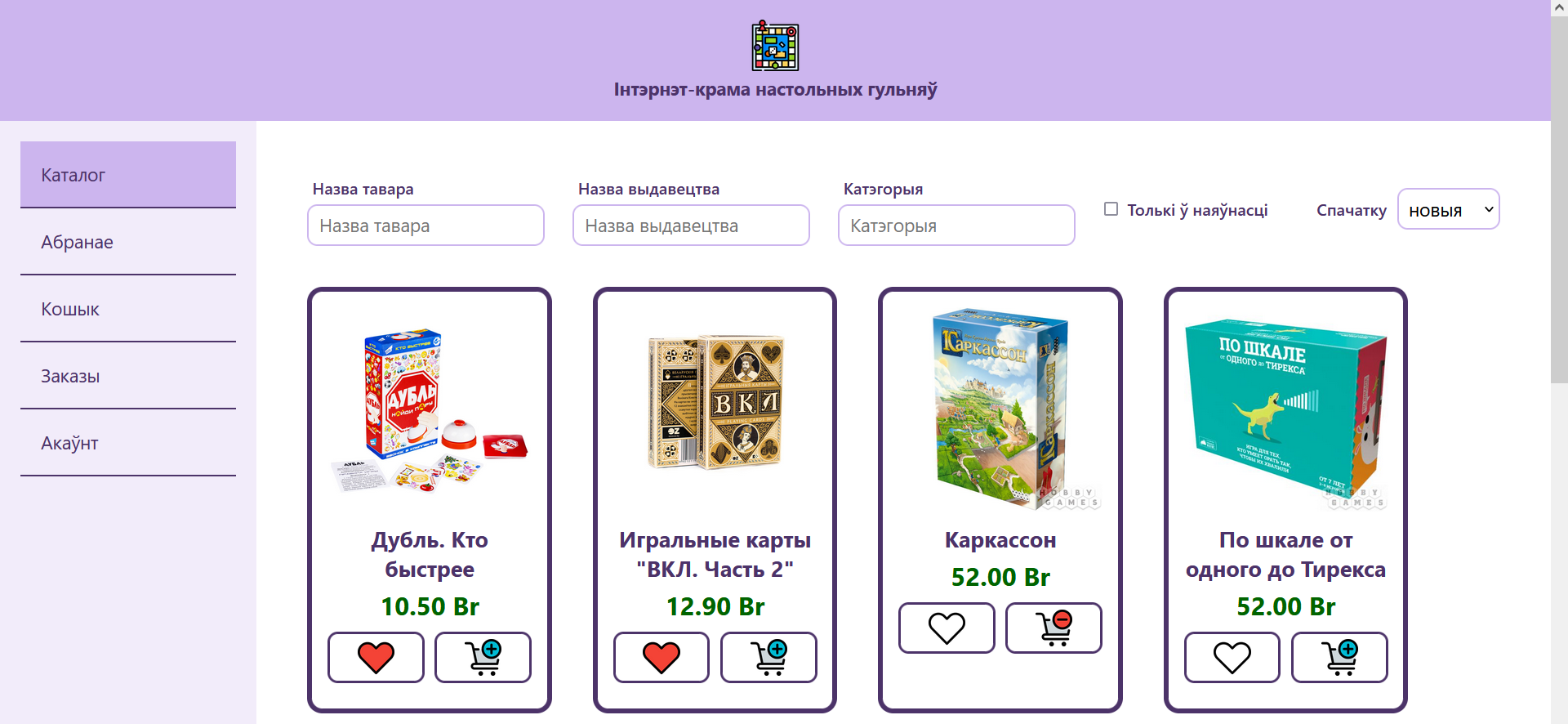


Рисунок 8.6 – Страница каталога для пользователя

На странице избранного пользователь может исключать товары из избранного, добавлять и исключать товары из корзины. Также можно очистить коллекцию избранного либо переместить товары из избранного в корзину. При этом товары, которых нет на складе или которые недоступны, не переносятся в корзину, а остаются в избранном. Страница избранного показана на рисунке 8.7.



Рисунок 8.7 – Страница избранного

На странице корзины пользователь может изменять количество товаров в корзине, исключать их из корзины, добавлять и исключать в избранное. Также отображается стоимость корзины. Можно очистить корзину или перейти к оформлению заказа. Страница корзины показана на рисунке 8.8.

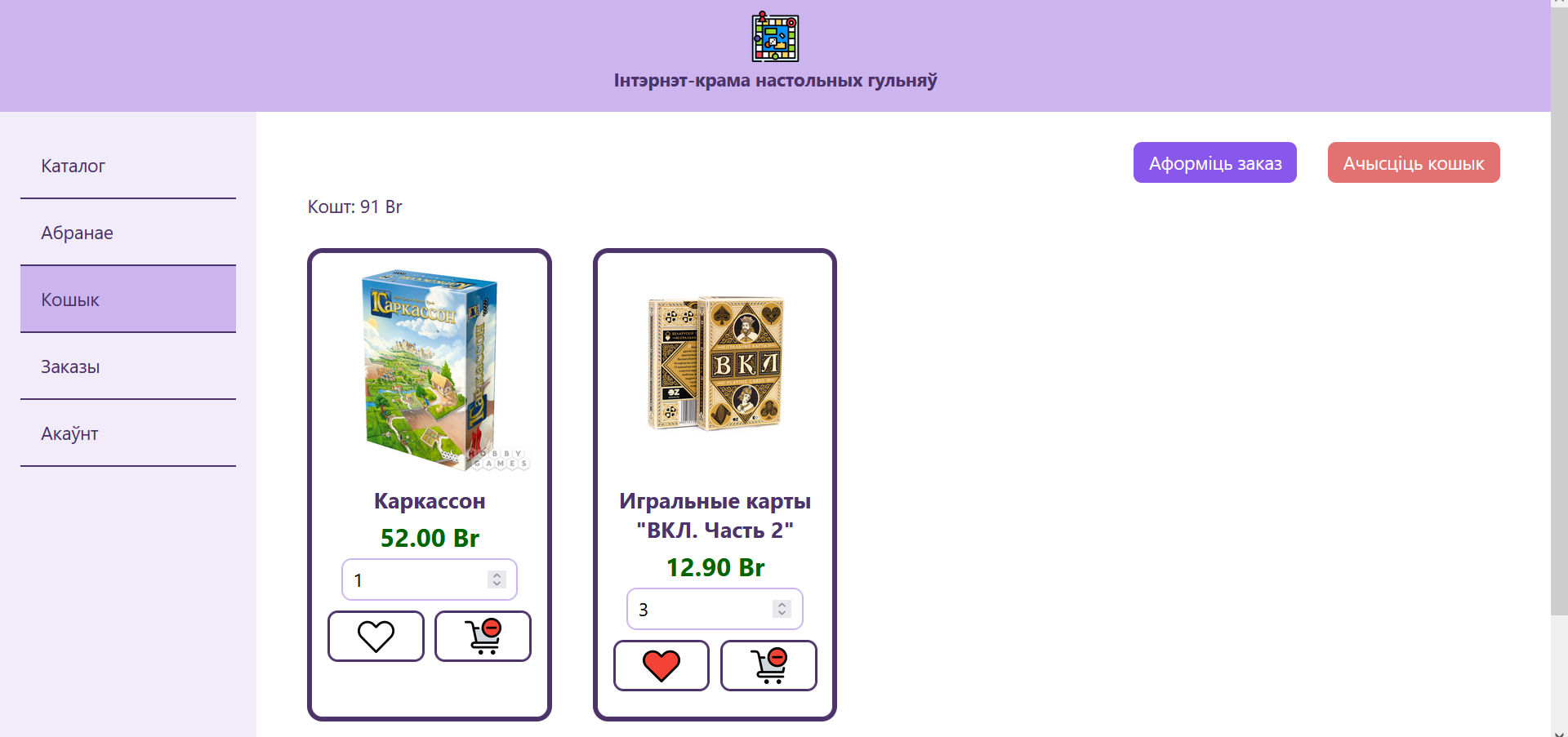


Рисунок 8.8 – Страница корзины

Страница заказов для пользователя такая же, как и для администратора, но в списке заказов отображаются заказы только этого пользователя и нет возможности изменить статус заказа, а только просмотреть.

При оформлении заказа необходимо указать данные карты, можно указать комментарий к заказу (необязательно). Если корзина не содержит недопустимых товаров, то по нажатию на соответствующую кнопку оформляется заказ. Страница оформления заказа показана на рисунке 8.9.

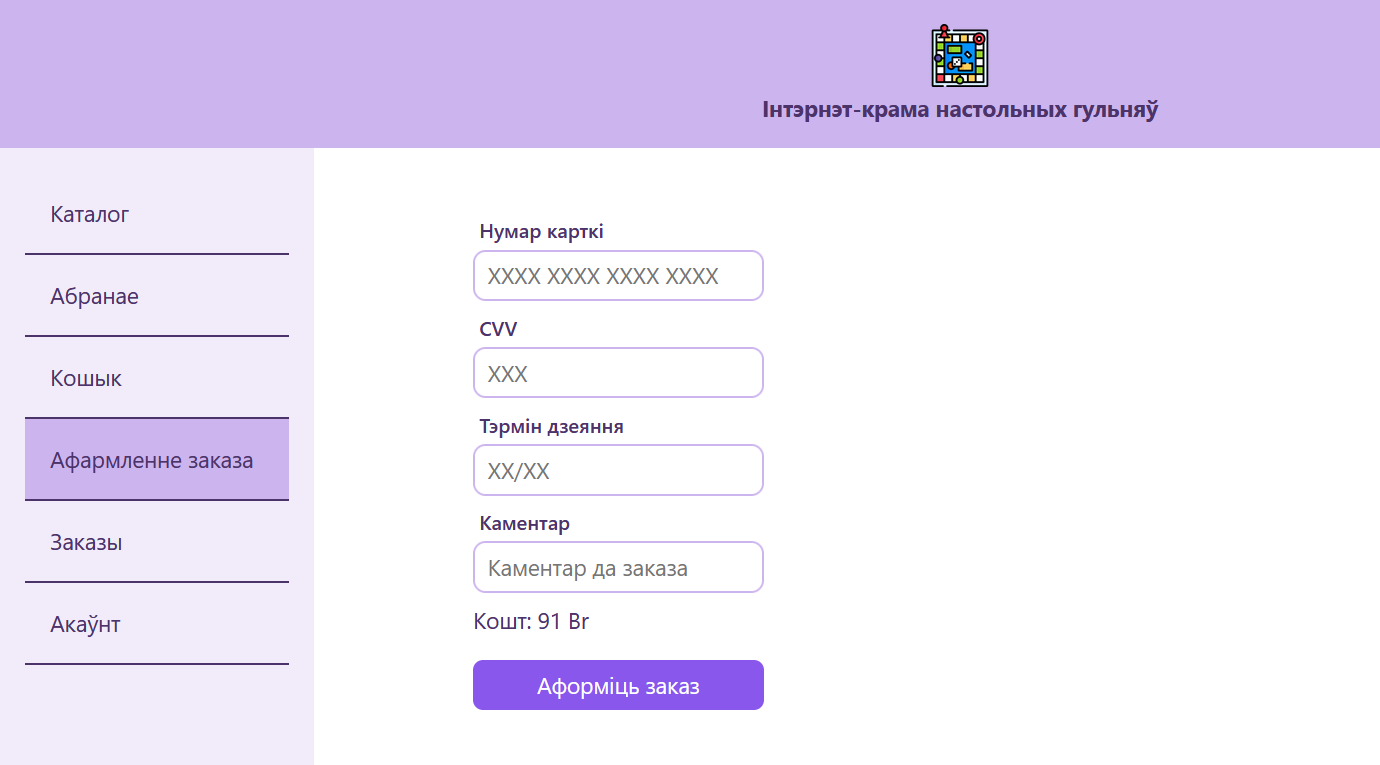


Рисунок 8.9 – Страница оформления заказа

Таким образом, приложение предоставляет различную функциональность для разных пользователей в зависимости от их роли. При этом она распределена по вкладкам для удобства обращения с приложением. Набор вкладок отличается для пользователей с разными ролями.

# **Заключение**

В результате выполнения курсовой работы при использовании СУБД Oracle была спроектирована и реализована база данных для интернет-магазина настольных игр, а также реализовано веб-приложение для демонстрации работоспособности БД.

Были выполнены все функциональные задачи, а именно:

* определены роли администратора, пользователя, гостя;
* реализовано управление каталогом настольных игр (просмотр, добавление товаров, изменение товаров);
* реализовано управление корзиной пользователя (добавление, исключение товаров);
* реализовано управление заказами пользователя (оформление, изменение статуса заказа).

Дополнительно было реализовано управление коллекцией избранных товаров (добавление, исключение товаров).

Были выполнены все поставленные требования:

* БД реализована в СУБД Oracle;
* доступ к данным осуществляется только через соответствующие процедуры;
* реализован импорт данных в XML-файл, экспорт данных из xml-файла;
* проведено тестирование производительности на таблице, содержащей не менее ста тысяч строк (двести тысяч строк), предприняты действия по повышению производительности (добавлен индекс);
* применена технология согласно выбранной теме (шифрование и маскирование данных);
* листинги содержат комментарии с описанием назначения каждой из процедур и функций.

Таким образом, все задачи курсовой работы были выполнены с учётом поставленных требований, её цель по разработке БД для интернет-магазина настольных игр была достигнута.

# **Список используемых источников**

1 Oracle. Что такое база данных? [Электронный ресурс]. – Режим доступа: https://www.oracle.com/cis/database/what-is-database/. – Дата доступа: 20.12.2023.

2 Express. Быстрый, гибкий и минималистичный веб-фреймворк для Node.js [Электронный ресурс]. – Режим доступа: https://expressjs.com/ru/. – Дата доступа: 20.12.2023.

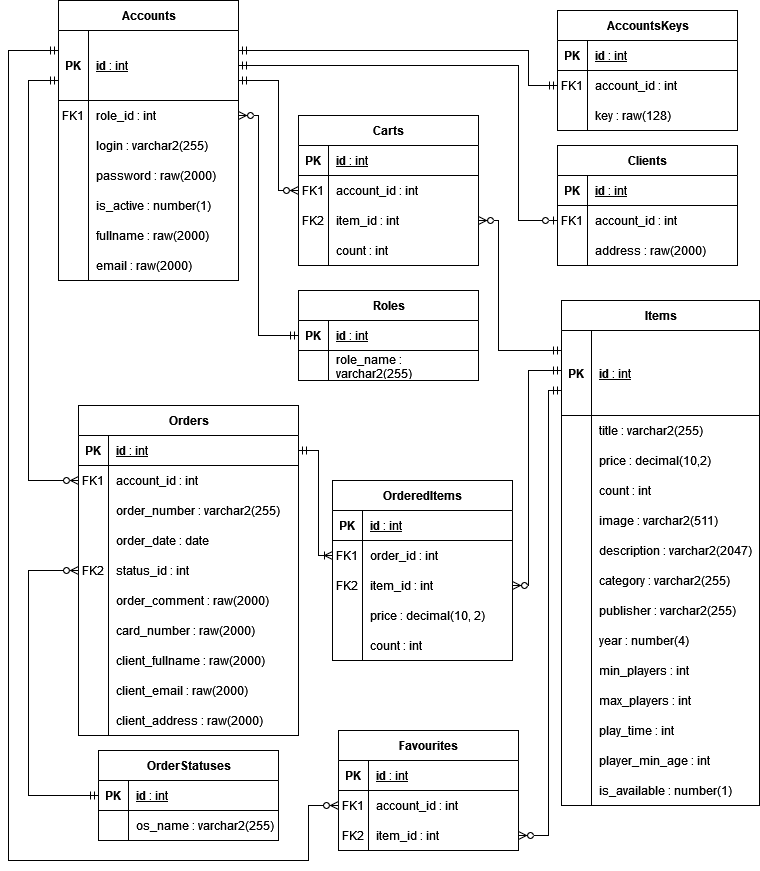
3 Концепция баз данных. Учебное пособие / Под ред. А. А. Шаховского. – М.: Издательство МГТУ им. Н. Э. Баумана, 2010. – 136 с. [Электронный ресурс]. – Режим доступа: https://helpiks.org/2-36749.html. – Дата доступа: 20.12.2023.

4 Касперский. Шифрование: что это такое и как оно работает [Электронный ресурс]. – Режим доступа: https://www.kaspersky.ru/resource-center/definitions/encryption. – Дата доступа: 20.12.2023.

5 Гардатех. GMA – система управления доступом [Электронный ресурс]. – Режим доступа: https://gardatech.ru/produkty/gma/. – Дата доступа: 20.12.2023.

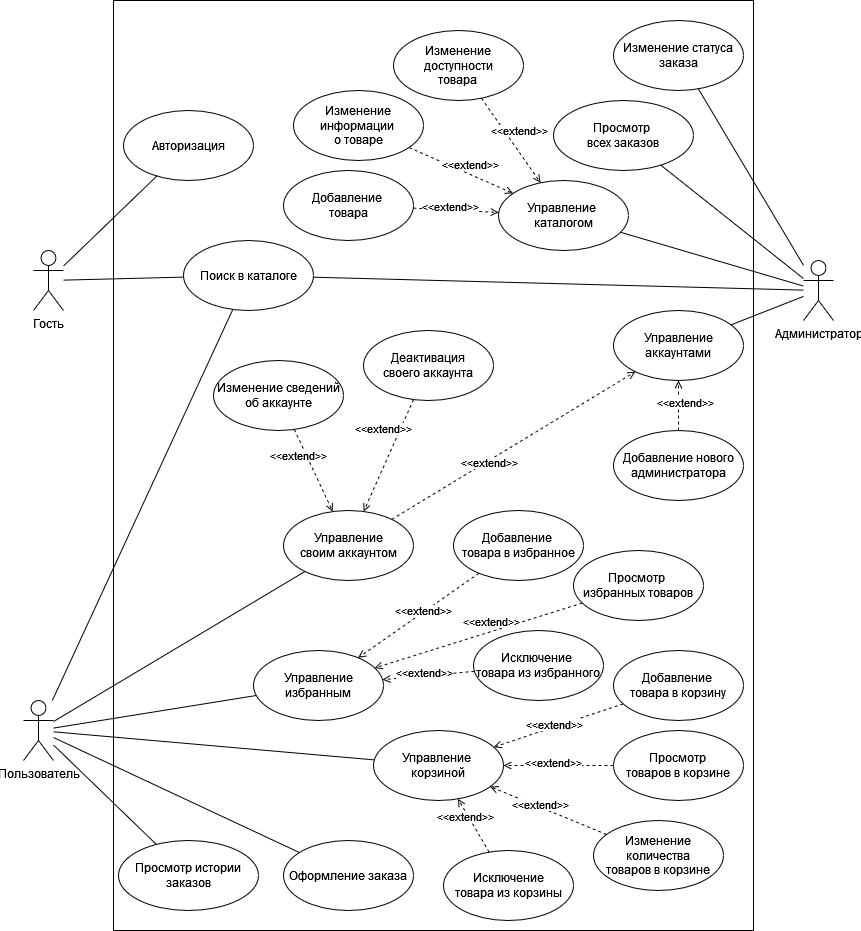
# **Приложение А**

Модель базы данных



# **Приложение Б**

Диаграмма вариантов использования



# **Приложение В**

Листинг создания таблиц, представлений и некоторых других объектов БД

|  |
| --- |
| --ORCLCDB, sys as SYSDBA  -- создать PDB  create pluggable database BGAMESSTORE  admin user SYSADMIN identified by SYSADMIN\_PASSWORD roles = (DBA)  file\_name\_convert = ('/opt/oracle/oradata/ORCLCDB/pdbseed/', '/opt/oracle/oradata/ORCLCDB/BGAMESSTORE/');  alter pluggable database BGAMESSTORE open;  --------------------------------------------------  -- позволить пользователю БД SYSADMIN вызывать методы DBMS\_CRYPTO  alter session set CONTAINER = BGAMESSTORE;  grant execute on sys.DBMS\_CRYPTO to SYSADMIN;  grant select any dictionary to SYSADMIN;  --------------------------------------------------  --BGAMESSTORE, SYSADMIN  -- создать табличное пространство БД  create tablespace BGS\_TS  datafile '/opt/oracle/oradata/ORCLCDB/BGAMESSTORE/BGS\_TS.dbf'  size 16M autoextend on next 16M;  alter user SYSADMIN default tablespace BGS\_TS quota unlimited on BGS\_TS;  -- таблица с полной информацией о товарах  create sequence items\_id\_seq;  create table Items(  id int default items\_id\_seq.nextval not null primary key,  title varchar2(255) not null,  price decimal(10, 2) check(price >= 0) not null,  count int check(count >= 0) not null,  image varchar2(511) not null,  description varchar2(2047) not null,  category varchar2(255) not null,  publisher varchar2(255) not null,  year number(4) check(year >= 0) not null,  min\_players int check(min\_players >= 0) not null,  max\_players int check(max\_players >= 0) not null,  play\_time int check(play\_time >= 0) not null,  player\_min\_age int check(player\_min\_age >= 0) not null,  is\_available number(1) default 1 check(is\_available in (0, 1)) not null,  constraint chk\_players\_range check (max\_players >= min\_players)  ) tablespace BGS\_TS;  -- таблица со списком ролей пользователей БД  create sequence roles\_id\_seq;  create table Roles(  id int default roles\_id\_seq.nextval not null primary key,  role\_name varchar2(255) not null  ) tablespace BGS\_TS;  -- таблица со сведениями об аккаунтах  create sequence accounts\_id\_seq;  create table Accounts (  id int default accounts\_id\_seq.nextval not null primary key,  role\_id int not null references Roles(id) on delete cascade,  login varchar2(255) not null,  password raw(2000) not null,  is\_active number(1) default 1 check(is\_active in (0, 1)) not null,  fullname raw(2000) not null,  email raw(2000) not null  ) tablespace BGS\_TS;  -- таблица с клиентской информацией аккаунтов  create sequence clients\_id\_seq;  create table Clients (  id int default accounts\_id\_seq.nextval not null primary key,  account\_id int not null references Accounts(id) on delete cascade unique,  address raw(2000) not null  ) tablespace BGS\_TS;  -- таблица с ключами шифрования аккаунтов  create sequence account\_keys\_id\_seq;  create table AccountsKeys (  id int default account\_keys\_id\_seq.nextval not null primary key,  account\_id int not null references Accounts(id) on delete cascade unique,  key raw(128) not null  ) tablespace BGS\_TS;  -- таблица с товарами, добавленными пользователями в корзину  create sequence carts\_id\_seq;  create table Carts (  id int default carts\_id\_seq.nextval not null primary key,  account\_id int not null references Accounts(id) on delete cascade,  item\_id int not null references Items(id) on delete cascade,  count int default 1 check(count > 0) not null  ) tablespace BGS\_TS;  -- таблица со списком возможных статусов заказа  create sequence order\_statuses\_id\_seq;  create table OrderStatuses (  id int default order\_statuses\_id\_seq.nextval not null primary key,  os\_name varchar2(255) not null  ) tablespace BGS\_TS;  -- таблица с информацией о заказах пользователей  create sequence orders\_id\_seq;  create table Orders (  id int default orders\_id\_seq.nextval not null primary key,  account\_id int not null references Accounts(id) on delete cascade,  order\_number varchar2(255) not null,  order\_date date not null,  status\_id int not null references OrderStatuses(id) on delete cascade,  order\_comment raw(2000),  card\_number raw(2000) not null,  client\_fullname raw(2000) not null,  client\_email raw(2000) not null,  client\_address raw(2000) not null  ) tablespace BGS\_TS;  -- таблица с информацией о заказанных товарах  create sequence ordered\_items\_id\_seq;  create table OrderedItems (  id int default ordered\_items\_id\_seq.nextval not null primary key,  order\_id int not null references Orders(id) on delete cascade,  item\_id int not null references Items(id) on delete cascade,  price decimal(10, 2) check(price >= 0) not null,  count int check(count > 0) not null  ) tablespace BGS\_TS;  -- таблица с товарами, добавленными пользователями в избранное  create sequence favourites\_id\_seq;  create table Favourites (  id int default favourites\_id\_seq.nextval not null primary key,  account\_id int not null references Accounts(id) on delete cascade,  item\_id int not null references Items(id) on delete cascade  ) tablespace BGS\_TS;  --------------------------------------------------  -- активные аккаунты  create or replace view V\_ActiveAccounts as  select \* from Accounts where is\_active = 1;    -- более подробная информация о заказе  create or replace view V\_Orders as  select  o.id, o.account\_id, o.order\_number, o.order\_date, os.os\_name, o.order\_comment,  o.client\_fullname, o.client\_email, o.client\_address, o.card\_number  from Orders o join OrderStatuses os on o.status\_id = os.id;    -- более подробная информация о заказанных товарах  create or replace view V\_OrderedItems as  select i.id, oi.order\_id, i.title, oi.price, oi.count, i.image  from OrderedItems oi join Items i on oi.item\_id = i.id;    -- более подробная информация о товарах в корзине  create or replace view V\_CartItems as  select i.id, c.account\_id, i.title, i.price, i.count, c.count as cart\_count, i.image, i.is\_available  from Carts c join Items i on c.item\_id = i.id;  -- более подробная информация о товарах в избранном  create or replace view V\_FavsItems as  select i.id, f.account\_id, i.title, i.price, i.count, i.image, i.is\_available  from Favourites f join Items i on f.item\_id = i.id;    ---- более подробная информация об аккаунте клиента  create or replace view V\_ClientAccountsInfo as  select a.id, a.fullname, a.email, c.address  from Accounts a join Clients c on c.account\_id = a.id;    -- аккаунты с ролями  create or replace view V\_AccountsWithRoles as  select a.id, a.login, r.role\_name  from V\_ActiveAccounts a join Roles r on a.role\_id = r.id;  --------------------------------------------------  -- заполнение таблиц Roles и OrderStatuses начальными значениями  insert into Roles(role\_name) values ('GUESTROLE'); -- гость  insert into Roles(role\_name) values ('CLIENTROLE'); -- пользователь (клиент)  insert into Roles(role\_name) values ('ADMINROLE'); -- администратор  insert into Roles(role\_name) values ('PDB\_DBS'); -- системный администратор  commit;  insert into OrderStatuses(os\_name) values ('PROCESSING'); -- в обработке  insert into OrderStatuses(os\_name) values ('SHIPPED'); -- отправлен  insert into OrderStatuses(os\_name) values ('DELIVERED'); -- доставлен  insert into OrderStatuses(os\_name) values ('CANCELLED'); -- отменён  insert into OrderStatuses(os\_name) values ('RETURNED'); -- возвращён  commit;  --------------------------------------------------  -- создать директорию для работы с XML-данными  --$ mkdir /opt/oracle/oradata/ORCLCDB/BGAMESSTORE/XML  create or replace directory BGAMESSTORE\_XML as '/opt/oracle/oradata/ORCLCDB/BGAMESSTORE/XML'; |

# **Приложение Г**

Листинг создания процедур и функций БД

|  |
| --- |
| -- хэшировать строку  create or replace function HashMD5(p\_text in varchar2)  return raw is  begin  return DBMS\_CRYPTO.hash(  UTL\_I18N.string\_to\_raw(p\_text, 'AL32UTF8'),  DBMS\_CRYPTO.HASH\_MD5  );  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- зашифровать строку ключом (AES128)  create or replace function EncryptAES128(  p\_text in varchar2,  p\_key in raw  ) return raw is  begin  return DBMS\_CRYPTO.encrypt(  UTL\_I18N.string\_to\_raw(p\_text, 'AL32UTF8'),  DBMS\_CRYPTO.ENCRYPT\_AES128 + DBMS\_CRYPTO.CHAIN\_CBC + DBMS\_CRYPTO.PAD\_PKCS5,  p\_key  );  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- дешифровать строку ключом (AES128)  create or replace function DecryptAES128(  p\_encrypted in raw,  p\_key in raw  ) return varchar2 is  begin  return UTL\_I18N.raw\_to\_char(  DBMS\_CRYPTO.decrypt(  p\_encrypted,  DBMS\_CRYPTO.ENCRYPT\_AES128 + DBMS\_CRYPTO.CHAIN\_CBC + DBMS\_CRYPTO.PAD\_PKCS5,  p\_key  ),  'AL32UTF8'  );  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- добавить аккаунт  create or replace procedure CreateAccount(  p\_role\_id in Accounts.role\_id%type,  p\_login in Accounts.login%type,  p\_password in varchar2,  p\_fullname in varchar2,  p\_email in varchar2,  p\_account\_id out Accounts.id%type  ) is  r\_key AccountsKeys.key%type;  e\_duplicate\_login exception;  begin  r\_key := DBMS\_CRYPTO.randombytes(16);  insert into Accounts(role\_id, login, password, fullname, email)  values (  p\_role\_id,  p\_login,  HashMD5(p\_password),  EncryptAES128(p\_fullname, r\_key),  EncryptAES128(p\_email, r\_key)  ) returning id into p\_account\_id;  insert into AccountsKeys(account\_id, key)  values (p\_account\_id, r\_key);  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- деактивировать аккаунт  create or replace procedure DeactivateAccount(  p\_account\_id in Accounts.id%type  ) is  begin  update Accounts set is\_active = 0 where id = p\_account\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- сгенерировать номер заказа в формате XXX\_XXX\_XXX  create or replace function GenerateOrderNumber  return varchar2 is  begin  return  DBMS\_RANDOM.string('X', 3) || '\_' ||  DBMS\_RANDOM.string('X', 3) || '\_' ||  DBMS\_RANDOM.string('X', 3);  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- существует ли аккаунт с заданным логином (и паролем)  create or replace function AccountExists(  p\_login in Accounts.login%type,  p\_password in varchar2 default null  ) return boolean is  r\_exists number(1);  r\_password raw(2000) default null;  begin  if p\_password is not null then  r\_password := HashMD5(p\_password);  end if;  select case when exists(  select 1 from V\_ActiveAccounts  where login = p\_login and  (r\_password is null or password = r\_password)  ) then 1 else 0 end  into r\_exists from dual;  return r\_exists != 0;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- сделать товар недоступным  create or replace procedure MakeItemUnavailable(  p\_item\_id in Items.id%type  ) is  begin  delete from Carts where item\_id = p\_item\_id;  update Items set is\_available = 0 where id = p\_item\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- пуста ли корзина пользователя  create or replace function CartIsEmpty(  p\_account\_id Accounts.id%type  ) return boolean is  r\_count int;  begin  select count(\*)  into r\_count  from Carts  where account\_id = p\_account\_id;  return r\_count = 0;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- может ли товар быть заказан  create or replace function ItemCanBeOrdered(  p\_id in Items.id%type  ) return int is  r\_can\_be\_ordered int;  begin  select case when is\_available = 1 and count > 0 then 1 else 0 end  into r\_can\_be\_ordered  from Items where id = p\_id;  return r\_can\_be\_ordered;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- содержит ли корзина товары, которые нельзя заказать  create or replace function CartContainsInvalidItems(  p\_account\_id in Accounts.id%type  ) return boolean is  r\_contains\_invalid\_items int;  begin  select case when exists(  select 1 from V\_CartItems  where account\_id = p\_account\_id and ItemCanBeOrdered(id) = 0  ) then 1 else 0 end  into r\_contains\_invalid\_items from dual;  return r\_contains\_invalid\_items != 0;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- получить стоимость заказа  create or replace function GetOrderCost(  p\_order\_id in Accounts.id%type  ) return decimal is  r\_order\_cost decimal default 0;  begin  select sum(price \* count)  into r\_order\_cost  from OrderedItems  where order\_id = p\_order\_id;  return r\_order\_cost;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- есть ли товар в корзине пользователя  create or replace function ItemIsInCart(  p\_account\_id in Accounts.id%type,  p\_item\_id in Items.id%type  ) return boolean is  r\_is\_in\_cart int;  begin  select case when exists(  select 1 from Carts  where account\_id = p\_account\_id and item\_id = p\_item\_id  ) then 1 else 0 end  into r\_is\_in\_cart from dual;  return r\_is\_in\_cart != 0;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- есть ли товар в избранном пользователя  create or replace function ItemIsInFavs(  p\_account\_id in Accounts.id%type,  p\_item\_id in Items.id%type  ) return boolean is  r\_is\_in\_favs int;  begin  select case when exists(  select 1 from Favourites  where account\_id = p\_account\_id and item\_id = p\_item\_id  ) then 1 else 0 end  into r\_is\_in\_favs from dual;  return r\_is\_in\_favs != 0;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- получить номер карты (возможно, замаскированный)  create or replace function GetCardNumber(  p\_card\_number in raw,  p\_key in AccountsKeys.key%type  ) return varchar2 is  r\_card\_number varchar2(255);  r\_is\_admin int;  begin  select case when exists(  select 1 from DBA\_ROLE\_PRIVS  where grantee = USER and granted\_role = 'ADMINROLE'  ) then 1 else 0 end into r\_is\_admin from dual;  select DecryptAES128(p\_card\_number, p\_key) into r\_card\_number from dual;  if r\_is\_admin = 1 then  return regexp\_replace(r\_card\_number, '(.{4})', '\1 ');  else  return  regexp\_replace(lpad('\*', length(r\_card\_number) - 4, '\*'), '(.{4})', '\1 ') ||  substr(r\_card\_number, -4);  end if;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- получить товары каталога  create or replace procedure GetCatalog(  p\_cur out sys\_refcursor,  p\_f\_title in Items.title%type default null,  p\_f\_publisher in Items.publisher%type default null,  p\_f\_in\_stock in int default 0,  p\_f\_category in Items.category%type default null,  p\_ordering in varchar2 default 'last\_added',  p\_account\_id in Accounts.id%type default null  ) is  f\_title Items.title%type := case when p\_f\_title is null then null else trim(p\_f\_title) end;  f\_publisher Items.publisher%type := case when p\_f\_publisher is null then null else trim(p\_f\_publisher) end;  f\_category Items.category%type := case when p\_f\_category is null then null else trim(p\_f\_category) end;  begin  open p\_cur for  select  id, title, price, image,  (case when count > 0 then 1 else 0 end) as is\_in\_stock,  is\_available,  (case when p\_account\_id is not null and exists (select 1 from Favourites f where f.account\_id = p\_account\_id and f.item\_id = i.id) then 1 else 0 end) as is\_in\_favs,  (case when p\_account\_id is not null and exists (select 1 from Carts c where c.account\_id = p\_account\_id and c.item\_id = i.id) then 1 else 0 end) as is\_in\_cart  from Items i  where  (f\_title is null or title like f\_title||'%') and  (f\_publisher is null or publisher like f\_publisher||'%') and  (p\_f\_in\_stock = 0 or (is\_available = 1 and count > 0)) and  (f\_category is null or category like f\_category||'%')  order by case  when p\_ordering = 'price\_asc' then price  when p\_ordering = 'price\_desc' then -price  else -id  end;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- получить полную информацию о товаре  create or replace procedure GetItemInfo(  p\_item\_id in Items.id%type,  p\_account\_id in Accounts.id%type default null,  p\_title out Items.title%type,  p\_price out Items.price%type,  p\_count out Items.count%type,  p\_image out Items.image%type,  p\_description out Items.description%type,  p\_category out Items.category%type,  p\_publisher out Items.publisher%type,  p\_year out Items.year%type,  p\_min\_players out Items.min\_players%type,  p\_max\_players out Items.max\_players%type,  p\_play\_time out Items.play\_time%type,  p\_player\_min\_age out Items.player\_min\_age%type,  p\_is\_available out number,  p\_is\_in\_favs out number,  p\_is\_in\_cart out number  ) is  begin  select  title, price, count, image,  description, category,  publisher, year,  min\_players, max\_players,  play\_time, player\_min\_age,  is\_available,  (case when p\_account\_id is not null and exists (select 1 from Favourites f where f.account\_id = p\_account\_id and f.item\_id = i.id) then 1 else 0 end) as is\_in\_favs,  (case when p\_account\_id is not null and exists (select 1 from Carts c where c.account\_id = p\_account\_id and c.item\_id = i.id) then 1 else 0 end) as is\_in\_cart  into  p\_title, p\_price, p\_count, p\_image,  p\_description, p\_category,  p\_publisher, p\_year,  p\_min\_players, p\_max\_players,  p\_play\_time, p\_player\_min\_age,  p\_is\_available, p\_is\_in\_favs, p\_is\_in\_cart  from Items i  where id = p\_item\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- получить полную информацию о заказе  create or replace procedure GetOrderInfo(  p\_order\_id in Orders.id%type,  p\_order\_number out Orders.order\_number%type,  p\_order\_date out Orders.order\_date%type,  p\_os\_name out OrderStatuses.os\_name%type,  p\_order\_comment out varchar2,  p\_order\_cost out decimal,  p\_client\_fullname out varchar2,  p\_client\_email out varchar2,  p\_client\_address out varchar2,  p\_card\_number out varchar2  ) is  r\_key AccountsKeys.key%type;  begin  select ak.key into r\_key  from Orders o join AccountsKeys ak on o.account\_id = ak.account\_id  where o.id = p\_order\_id;  select  order\_number, order\_date, os\_name,  DecryptAES128(order\_comment, r\_key),  GetOrderCost(id),  DecryptAES128(client\_fullname, r\_key),  DecryptAES128(client\_email, r\_key),  DecryptAES128(client\_address, r\_key),  GetCardNumber(card\_number, r\_key)  into  p\_order\_number, p\_order\_date, p\_os\_name, p\_order\_comment, p\_order\_cost,  p\_client\_fullname, p\_client\_email, p\_client\_address, p\_card\_number  from V\_Orders  where id = p\_order\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- получить список товаров в заказе  create or replace procedure GetOrderItems(  p\_order\_id in Orders.id%type,  p\_cur out sys\_refcursor  ) is  begin  open p\_cur for  select  id, title, price, count, image  from V\_OrderedItems  where order\_id = p\_order\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- получить сведения аккаунта  create or replace procedure GetAccountInfo(  p\_id in Accounts.id%type,  p\_login out Accounts.login%type,  p\_fullname out varchar2,  p\_email out varchar2  ) is  r\_key AccountsKeys.key%type;  begin  select key into r\_key from AccountsKeys where account\_id = p\_id;  select login, DecryptAES128(fullname, r\_key), DecryptAES128(email, r\_key)  into p\_login, p\_fullname, p\_email  from Accounts  where id = p\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- обновить сведения аккаунта  create or replace procedure UpdateAccountInfo(  p\_id in Accounts.id%type,  p\_login in varchar2,  p\_password in varchar2,  p\_fullname in varchar2,  p\_email in varchar2  ) is  r\_old\_login Accounts.login%type;  LOGIN\_ALREADY\_EXISTS\_EXCEPTION exception;  pragma exception\_init(LOGIN\_ALREADY\_EXISTS\_EXCEPTION, -20001);  r\_key AccountsKeys.key%type;  begin  if p\_login is not null then  select login into r\_old\_login from Accounts where id = p\_id;  if p\_login != r\_old\_login and AccountExists(p\_login) then  raise\_application\_error(-20001, 'LOGIN\_ALREADY\_EXISTS\_EXCEPTION');  end if;  end if;  select key into r\_key from AccountsKeys where account\_id = p\_id;  update Accounts  set  login = case when p\_login is null then login else p\_login end,  password = case when p\_password is null then password else HashMD5(p\_password) end,  fullname = case when p\_fullname is null then fullname else EncryptAES128(p\_fullname, r\_key) end,  email = case when p\_email is null then email else EncryptAES128(p\_email, r\_key) end  where id = p\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- деактивировать аккаунт клиента  create or replace procedure DeactivateClientAccount(  p\_account\_id in Accounts.id%type  ) is  begin  delete from Favourites where account\_id = p\_account\_id;  delete from Carts where account\_id = p\_account\_id;  DeactivateAccount(p\_account\_id);  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- добавить нового администратора  create or replace procedure CreateNewAdminAccount(  p\_login in Accounts.login%type,  p\_password in varchar2,  p\_fullname in varchar2,  p\_email in varchar2,  p\_account\_id out Accounts.id%type  ) is  r\_role\_id Roles.id%type;  begin  if AccountExists(p\_login) then  raise\_application\_error(-20001, 'LOGIN\_ALREADY\_EXISTS\_EXCEPTION');  end if;  select id into r\_role\_id from Roles where role\_name = 'ADMINROLE';  CreateAccount(  r\_role\_id, p\_login, p\_password, p\_fullname, p\_email,  p\_account\_id  );  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- деактивировать аккаунт администратора  create or replace procedure DeactivateAdminAccount(  p\_account\_id in Accounts.id%type  ) is  begin  DeactivateAccount(p\_account\_id);  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- добавить товар в каталог  create or replace procedure AddItem(  p\_title in Items.title%type,  p\_price in Items.price%type,  p\_count in Items.count%type,  p\_image in Items.image%type,  p\_description in Items.description%type,  p\_category in Items.category%type,  p\_publisher in Items.publisher%type,  p\_year in Items.year%type,  p\_min\_players in Items.min\_players%type,  p\_max\_players in Items.max\_players%type,  p\_play\_time in Items.play\_time%type,  p\_player\_min\_age in Items.player\_min\_age%type,  p\_is\_available in Items.is\_available%type,  p\_id out Items.id%type  ) is  begin  insert into Items(  title, price, count,  image, description, category,  publisher, year,  min\_players, max\_players,  play\_time, player\_min\_age, is\_available  ) values (  p\_title, p\_price, p\_count,  p\_image, p\_description, p\_category,  p\_publisher, p\_year,  p\_min\_players, p\_max\_players,  p\_play\_time, p\_player\_min\_age, p\_is\_available  ) returning id into p\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- обновить информацию о товаре  create or replace procedure UpdateItem(  p\_id in Items.id%type,  p\_title in Items.title%type,  p\_price in Items.price%type,  p\_count in Items.count%type,  p\_image in Items.image%type,  p\_description in Items.description%type,  p\_category in Items.category%type,  p\_publisher in Items.publisher%type,  p\_year in Items.year%type,  p\_min\_players in Items.min\_players%type,  p\_max\_players in Items.max\_players%type,  p\_play\_time in Items.play\_time%type,  p\_player\_min\_age in Items.player\_min\_age%type,  p\_is\_available in Items.is\_available%type  ) is  begin  update Items  set  title = case when p\_title is null then title else p\_title end,  price = case when p\_price is null then price else p\_price end,  count = case when p\_count is null then count else p\_count end,  image = case when p\_image is null then image else p\_image end,  description = case when p\_description is null then description else p\_description end,  category = case when p\_category is null then category else p\_category end,  publisher = case when p\_publisher is null then publisher else p\_publisher end,  year = case when p\_year is null then year else p\_year end,  min\_players = case when p\_min\_players is null then min\_players else p\_min\_players end,  max\_players = case when p\_max\_players is null then max\_players else p\_max\_players end,  play\_time = case when p\_play\_time is null then play\_time else p\_play\_time end,  player\_min\_age = case when p\_player\_min\_age is null then player\_min\_age else p\_player\_min\_age end,  is\_available = case when p\_is\_available is null then is\_available else p\_is\_available end  where id = p\_id;  if p\_is\_available = 0 then  MakeItemUnavailable(p\_id);  end if;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- задать статус заказа  create or replace procedure SetOrderStatus(  p\_order\_id in Orders.id%type,  p\_new\_os\_name in OrderStatuses.os\_name%type  ) is  r\_os\_id OrderStatuses.id%type;  begin  select id  into r\_os\_id  from OrderStatuses  where os\_name = p\_new\_os\_name;  update Orders  set status\_id = r\_os\_id  where id = p\_order\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- получить список возможных статусов заказа  create or replace procedure GetOrderStatuses (  p\_ref\_cursor out sys\_refcursor  ) as  begin  open p\_ref\_cursor for  select id, os\_name from OrderStatuses;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- получить список всех заказов  create or replace procedure GetAllOrdersList(  p\_cur out sys\_refcursor  ) is  begin  open p\_cur for  select  id, order\_number, order\_date, os\_name  from V\_Orders  order by order\_date desc;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- записать каталог в XML-файл  create or replace procedure ExportItemsToXML (  p\_filename in varchar2  ) is  r\_xml\_data clob;  r\_file\_handle utl\_file.file\_type;  begin  select  xmlelement("items", xmlagg(xmlelement("item", xmlforest(  id as "id", title as "title", price as "price", count as "count", image as "image", description as "description",  category as "category", publisher as "publisher", year as "year", min\_players as "min\_players", max\_players as "max\_players",  play\_time as "play\_time", player\_min\_age as "player\_min\_age", is\_available as "is\_available"  )))).getClobVal()  into r\_xml\_data  from Items;    r\_file\_handle := utl\_file.fopen('BGAMESSTORE\_XML', p\_filename, 'w', 32767);  utl\_file.put(r\_file\_handle, r\_xml\_data);  utl\_file.fclose(r\_file\_handle);  exception  when others then  raise\_application\_error(-20005, 'EXPORT\_TO\_XML\_EXCEPTION: '||SQLERRM);  end;  /  -- добавить товары из XML-файла в каталог  create or replace procedure ImportItemsFromXML(  p\_filename in varchar2  ) is  r\_xml\_data clob;  r\_items\_xml xmltype;  r\_file\_handle utl\_file.file\_type;  buffer varchar2(32767);  begin  r\_file\_handle := utl\_file.fopen('BGAMESSTORE\_XML', p\_filename, 'r', 32767);  loop  begin  utl\_file.get\_line(r\_file\_handle, buffer);  r\_xml\_data := r\_xml\_data || buffer;  exception  when NO\_DATA\_FOUND then  exit;  end;  end loop;  utl\_file.fclose(r\_file\_handle);  r\_items\_xml := xmltype(r\_xml\_data);  for item in (select \*  from xmltable('/items/item' passing r\_items\_xml columns  title varchar2(255) path 'title', price decimal(10, 2) path 'price',  count int path 'count', image varchar2(511) path 'image',  description varchar2(2047) path 'description', category varchar2(255) path 'category',  publisher varchar2(255) path 'publisher', year number(4) path 'year',  min\_players int path 'min\_players', max\_players int path 'max\_players',  play\_time int path 'play\_time', player\_min\_age int path 'player\_min\_age',  is\_available number(1) path 'is\_available')) loop  insert into Items (  title, price, count, image, description, category, publisher, year,  min\_players, max\_players, play\_time, player\_min\_age, is\_available  ) values (  item.title, item.price, item.count, item.image, item.description,  item.category, item.publisher, item.year, item.min\_players, item.max\_players,  item.play\_time, item.player\_min\_age, item.is\_available  );  end loop;  exception  when others then  raise\_application\_error(-20006, 'IMPORT\_FROM\_XML\_EXCEPTION: '||SQLERRM);  end;  /  -- получить список товаров в корзине пользователя  create or replace procedure GetCartItems(  p\_account\_id in Accounts.id%type,  p\_cur out sys\_refcursor  ) is  begin  open p\_cur for  select  id, title, price, cart\_count, image,  (case when count > 0 then 1 else 0 end) as is\_in\_stock,  is\_available,  (case when exists (select 1 from Favourites f where f.account\_id = p\_account\_id and f.item\_id = f.id) then 1 else 0 end) as is\_in\_favs  from V\_CartItems  where account\_id = p\_account\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- получить список товаров в избранном пользователя  create or replace procedure GetFavsItems(  p\_account\_id in Accounts.id%type,  p\_cur out sys\_refcursor  ) is  begin  open p\_cur for  select  id, title, price, image,  (case when count > 0 then 1 else 0 end) as is\_in\_stock,  is\_available,  (case when exists (select 1 from Carts c where c.account\_id = p\_account\_id and c.item\_id = f.id) then 1 else 0 end) as is\_in\_cart  from V\_FavsItems f  where account\_id = p\_account\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- обновить количество товара в корзине пользователя  create or replace procedure UpdateCartItemCount(  p\_account\_id in Accounts.id%type,  p\_item\_id in Items.id%type,  p\_count in out Carts.count%type  ) is  r\_is\_available Items.is\_available%type;  r\_item\_count Items.count%type;  r\_count int;  begin  select is\_available, count  into r\_is\_available, r\_item\_count  from Items  where id = p\_item\_id;  if r\_is\_available = 0 then  delete  from Carts  where account\_id = p\_account\_id and item\_id = p\_item\_id;  p\_count := 0;  return;  end if;    if p\_count > 0 then  if r\_item\_count = 0 then  p\_count := 0;  return;  end if;    select count(\*)  into r\_count  from Carts  where account\_id = p\_account\_id and item\_id = p\_item\_id;  if r\_count = 0 then  insert into Carts(account\_id, item\_id, count)  values (p\_account\_id, p\_item\_id, p\_count);  else  update Carts  set count = p\_count  where account\_id = p\_account\_id and item\_id = p\_item\_id;  end if;  else  delete  from Carts  where account\_id = p\_account\_id and item\_id = p\_item\_id;  p\_count := 0;  end if;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- добавить/исключить товар из избранного пользователя  create or replace procedure ToggleItemInFavs(  p\_account\_id in Accounts.id%type,  p\_item\_id in Items.id%type,  p\_is\_in\_favs out boolean  ) is  r\_is\_in\_favs boolean;  begin  r\_is\_in\_favs := ItemIsInFavs(p\_account\_id, p\_item\_id);  if r\_is\_in\_favs then  delete from Favourites  where account\_id = p\_account\_id and item\_id = p\_item\_id;  else  insert into Favourites(account\_id, item\_id)  values (p\_account\_id, p\_item\_id);  end if;  p\_is\_in\_favs := not r\_is\_in\_favs;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- добавить товар в избранное  create or replace function AddToFavs(  p\_account\_id in Favourites.account\_id%type,  p\_item\_id in Favourites.item\_id%type  ) return number as  r\_count number;  begin  select count(\*) into r\_count from Favourites where account\_id = p\_account\_id and item\_id = p\_item\_id;  if r\_count = 0 then  insert into Favourites(account\_id, item\_id) values (p\_account\_id, p\_item\_id);  return 1;  else  return 0;  end if;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- исключить товар из избранного  create or replace procedure RemoveFromFavs(  p\_account\_id in Favourites.account\_id%type,  p\_item\_id in Favourites.item\_id%type  ) as  begin  delete from Favourites where account\_id = p\_account\_id and item\_id = p\_item\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- добавить/исключить товар из корзины пользователя  create or replace procedure ToggleItemInCart(  p\_account\_id in Accounts.id%type,  p\_item\_id in Items.id%type,  p\_is\_in\_cart out boolean  ) is  r\_count int;  begin  r\_count := case when ItemIsInCart(p\_account\_id, p\_item\_id)  then 0 else 1 end;  UpdateCartItemCount(p\_account\_id, p\_item\_id, r\_count);  p\_is\_in\_cart := case when r\_count = 0 then false else true end;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- добавить товар в корзину пользователя  create or replace procedure AddToCart (  p\_account\_id in Carts.account\_id%type,  p\_item\_id in Carts.item\_id%type,  p\_count in Carts.count%type default 1,  p\_rows\_added out number  ) as  begin  select count(\*) into p\_rows\_added from Carts  where account\_id = p\_account\_id and item\_id = p\_item\_id;  if p\_rows\_added = 0 then  insert into Carts (account\_id, item\_id, count)  values (p\_account\_id, p\_item\_id, p\_count);  p\_rows\_added := 1;  else  p\_rows\_added := 0;  end if;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- исключить товар из корзины пользователя  create or replace procedure RemoveFromCart (  p\_account\_id in Carts.account\_id%type,  p\_item\_id in Carts.item\_id%type  ) as  begin  delete from Carts  where account\_id = p\_account\_id and item\_id = p\_item\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- удостовериться, что корзина товаров может быть заказана  create or replace procedure ValidateCart(  p\_account\_id in Carts.account\_id%type  ) is  begin  if CartIsEmpty(p\_account\_id) then  raise\_application\_error(-20002, 'CART\_IS\_EMPTY\_EXCEPTION');  end if;  if CartContainsInvalidItems(p\_account\_id) then  raise\_application\_error(-20003, 'CART\_CONTAINS\_INVALID\_ITEMS\_EXCEPTION');  end if;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- оформить заказ  create or replace procedure MakeOrder(  p\_account\_id in Orders.account\_id%type,  p\_order\_comment in varchar2,  p\_card\_number in varchar2,  p\_order\_id out Orders.id%type  ) is  r\_order\_date Orders.order\_date%type;  r\_os\_id Orders.status\_id%type;  r\_client\_fullname raw(2000);  r\_client\_email raw(2000);  r\_client\_address raw(2000);  r\_key AccountsKeys.key%type;  begin  ValidateCart(p\_account\_id);    select fullname, email, address  into r\_client\_fullname, r\_client\_email, r\_client\_address  from V\_ClientAccountsInfo  where id = p\_account\_id;  select sys\_extract\_utc(SYSTIMESTAMP) into r\_order\_date from dual;  select id into r\_os\_id from OrderStatuses where os\_name = 'PROCESSING';  select key into r\_key from AccountsKeys where account\_id = p\_account\_id;  insert into Orders(  account\_id, order\_number,  order\_date, status\_id,  client\_fullname, client\_email, client\_address,  order\_comment, card\_number  ) values (  p\_account\_id, GenerateOrderNumber(),  r\_order\_date, r\_os\_id,  r\_client\_fullname, r\_client\_email, r\_client\_address,  EncryptAES128(p\_order\_comment, r\_key),  EncryptAES128(p\_card\_number, r\_key)  ) returning id into p\_order\_id;  for cart\_record in (  select id, account\_id, price, cart\_count  from V\_CartItems  where account\_id = p\_account\_id  ) loop  insert into OrderedItems(order\_id, item\_id, price, count)  values (  p\_order\_id,  cart\_record.id,  cart\_record.price,  cart\_record.cart\_count  );  end loop;  delete from Carts where account\_id = p\_account\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- получить список заказов пользователя  create or replace procedure GetClientOrdersList(  p\_account\_id in Accounts.id%type,  p\_cur out sys\_refcursor  ) is  begin  open p\_cur for  select  id, order\_number, order\_date, os\_name  from V\_Orders  where account\_id = p\_account\_id  order by order\_date desc;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- получить клиентские сведения аккаунта пользователя  create or replace procedure GetClientAccountInfo(  p\_id in Accounts.id%type,  p\_address out varchar2  ) is  r\_key AccountsKeys.key%type;  begin  select key into r\_key from AccountsKeys where account\_id = p\_id;  select DecryptAES128(address, r\_key)  into p\_address  from V\_ClientAccountsInfo  where id = p\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- обновить клиентские сведения аккаунта пользователя  create or replace procedure UpdateClientAccountInfo(  p\_id in Accounts.id%type,  p\_address varchar2  ) is  r\_key AccountsKeys.key%type;  begin  select key into r\_key from AccountsKeys where account\_id = p\_id;  update Clients  set address = case when p\_address is null then address else EncryptAES128(p\_address, r\_key) end  where account\_id = p\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- получить стоимость корзины пользователя  create or replace procedure GetCartCost(  p\_account\_id in Accounts.id%type,  p\_cart\_cost out decimal  ) is  begin  select coalesce(sum(price \* cart\_count), 0)  into p\_cart\_cost  from V\_CartItems  where account\_id = p\_account\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- переместить избранные товары в корзину  create or replace procedure MoveFavsToCart(  p\_account\_id in accounts.id%type  ) is  r\_count int;  begin  for fav in (select id, is\_available, count from V\_FavsItems where account\_id = p\_account\_id) loop  if fav.is\_available = 1 and fav.count > 0 then  select count(\*) into r\_count from Carts  where account\_id = p\_account\_id and item\_id = fav.id;  if r\_count = 0 then  insert into Carts (account\_id, item\_id, count)  values (p\_account\_id, fav.id, 1);  end if;  delete from Favourites  where item\_id = fav.id and account\_id = p\_account\_id;  end if;  end loop;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- очистить избранное пользователя  create or replace procedure ClearFavs(  p\_account\_id in accounts.id%type  ) is  begin  delete from Favourites where account\_id = p\_account\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- очистить корзину пользователя  create or replace procedure ClearCart(  p\_account\_id in accounts.id%type  ) is  begin  delete from Carts where account\_id = p\_account\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- зарегистрироваться как клиент  create or replace procedure SignUpAsClient(  p\_login in Accounts.login%type,  p\_password in varchar2,  p\_fullname in varchar2,  p\_email in varchar2,  p\_address in varchar2,  p\_account\_id out Accounts.id%type  ) is  r\_exists boolean;  r\_role\_id Roles.id%type;  r\_key AccountsKeys.key%type;  begin  if AccountExists(p\_login) then  raise\_application\_error(-20001, 'LOGIN\_ALREADY\_EXISTS\_EXCEPTION');  end if;    select id into r\_role\_id from Roles where role\_name = 'CLIENTROLE';  CreateAccount(r\_role\_id, p\_login, p\_password, p\_fullname, p\_email, p\_account\_id);  select key into r\_key from AccountsKeys where account\_id = p\_account\_id;  insert into Clients(account\_id, address)  values (p\_account\_id, EncryptAES128(p\_address, r\_key));  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- войти  create or replace procedure SignIn(  p\_login in Accounts.login%type,  p\_password in varchar2,  p\_account\_id out Accounts.id%type,  p\_role\_name out Roles.role\_name%type  ) is  begin  if not AccountExists(p\_login, p\_password) then  raise\_application\_error(-20004, 'ACCOUNT\_NOT\_FOUND\_EXCEPTION');  end if;    select id, role\_name  into p\_account\_id, p\_role\_name  from V\_AccountsWithRoles  where login = p\_login;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- ! - получить сведения всех аккаунтов  create or replace procedure sys\_Get\_Accounts (  p\_cur out sys\_refcursor  ) is  begin  open p\_cur for  select a.id, a.role\_id, a.login, a.password, a.is\_active,  DecryptAES128(fullname, ak.key), DecryptAES128(email, ak.key)  from Accounts a join AccountsKeys ak on a.id = ak.account\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- ! - удалить аккаунт  create or replace procedure sys\_Delete\_Account (  p\_id in accounts.id%type  ) is  begin  delete from Accounts where id = p\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- ! - получить всю клиентскую информацию аккаунтов  create or replace procedure sys\_Get\_Clients (  p\_cur out sys\_refcursor  ) is  begin  open p\_cur for  select c.id, c.account\_id, DecryptAES128(address, ak.key)  from Clients c join AccountsKeys ak on c.account\_id = ak.account\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- ! - получить содержимое всех корзин пользователей  create or replace procedure sys\_Get\_Carts (  p\_cur out sys\_refcursor  ) is  begin  open p\_cur for  select id, account\_id, item\_id, count  from Carts;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- добавить новый возможный статус заказа  create or replace procedure sys\_Add\_OrderStatus(  p\_os\_name in OrderStatuses.os\_name%type  ) is  begin  insert into OrderStatuses(os\_name) values (p\_os\_name);  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- ! - удалить возможный статус заказа  create or replace procedure sys\_Delete\_OrderStatus(  p\_id in OrderStatuses.id%type  ) is  begin  delete from OrderStatuses where id = p\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- ! - изменить возможный статус заказа  create or replace procedure sys\_Update\_OrderStatus(  p\_id in OrderStatuses.id%type,  p\_os\_name in OrderStatuses.os\_name%type  ) as  begin  update OrderStatuses set os\_name = p\_os\_name where id = p\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- ! - изменить данные заказа  create or replace procedure sys\_UpdateOrder(  p\_id in orders.id%type,  p\_account\_id in orders.account\_id%type,  p\_order\_number in orders.order\_number%type,  p\_order\_date in orders.order\_date%type,  p\_status\_id in orders.status\_id%type,  p\_order\_comment in varchar2,  p\_card\_number in varchar2,  p\_client\_fullname in varchar2,  p\_client\_email in varchar2,  p\_client\_address in varchar2  ) as  r\_order Orders%rowtype;  r\_key AccountsKeys.key%type;  begin  select \* into r\_order  from Orders where id = p\_id;  select key into r\_key from AccountsKeys where account\_id = r\_order.account\_id;  update Orders  set account\_id = coalesce(p\_account\_id, r\_order.account\_id),  order\_number = coalesce(p\_order\_number, r\_order.order\_number),  order\_date = coalesce(p\_order\_date, r\_order.order\_date),  status\_id = coalesce(p\_status\_id, r\_order.status\_id),  order\_comment = case when p\_order\_comment is null then r\_order.order\_comment else EncryptAES128(p\_order\_comment, r\_key) end,  card\_number = case when p\_card\_number is null then r\_order.card\_number else EncryptAES128(p\_card\_number, r\_key) end,  client\_fullname = case when p\_client\_fullname is null then r\_order.client\_fullname else EncryptAES128(p\_client\_fullname, r\_key) end,  client\_email = case when p\_client\_email is null then r\_order.client\_email else EncryptAES128(p\_client\_email, r\_key) end,  client\_address = case when p\_client\_address is null then r\_order.client\_address else EncryptAES128(p\_client\_address, r\_key) end  where id = p\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- ! - удалить заказ из истории заказов  create or replace procedure sys\_DeleteOrder(  p\_id in Orders.id%type  ) is  begin  delete from Orders where id = p\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- ! - получить информацию о всех заказах  create or replace procedure sys\_GetOrders(  p\_cur out sys\_refcursor  ) is  begin  open p\_cur for  select o.id, o.account\_id, o.order\_number, o.order\_date, o.status\_id,  DecryptAES128(o.order\_comment, key) as order\_comment,  DecryptAES128(o.card\_number, key) as card\_number,  DecryptAES128(o.client\_fullname, key) as client\_fullname,  DecryptAES128(o.client\_email, key) as client\_email,  DecryptAES128(o.client\_address, key) as client\_address  from Orders o join AccountsKeys ak on o.account\_id = ak.account\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- ! - добавить новый заказанный товар  create or replace procedure sys\_Add\_OrderedItem(  p\_order\_id in OrderedItems.order\_id%type,  p\_item\_id in OrderedItems.item\_id%type,  p\_price in OrderedItems.price%type,  p\_count in OrderedItems.count%type  ) is  begin  insert into OrderedItems(order\_id, item\_id, price, count)  values (p\_order\_id, p\_item\_id, p\_price, p\_count);  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- ! - удаление заказанного товара  create or replace procedure sys\_Delete\_OrderedItem(  p\_id in OrderedItems.id%type  ) is  begin  delete from OrderedItems where id = p\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- ! - изменить информацию о заказанном товаре  create or replace procedure sys\_Update\_OrderedItem(  p\_id in OrderedItems.id%type,  p\_price in OrderedItems.price%type,  p\_count in OrderedItems.count%type  ) is  begin  update OrderedItems  set  price = coalesce(p\_price, price),  count = coalesce(p\_count, count)  where id = p\_id;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- получить список всех заказанных товаров  create or replace procedure sys\_Get\_OrderedItems(  p\_cur out sys\_refcursor  ) is  begin  open p\_cur for  select id, order\_id, item\_id, price, count  from OrderedItems;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  /  -- ! - получить список всех избранных товаров пользователей  create or replace procedure sys\_Get\_Favs(  p\_cur out sys\_refcursor  ) is  begin  open p\_cur for select id, account\_id, item\_id from Favourites;  exception  when others then  raise\_application\_error(-20099, 'OTHER\_EXCEPTION: '||SQLERRM);  end;  / |

# **Приложение Д**

Листинг создания ролей и пользователей БД

|  |
| --- |
| -- роль администратора  create role AdminRole;  grant create session to AdminRole;  --&  grant execute on GetCatalog to AdminRole;  grant execute on GetItemInfo to AdminRole;  grant execute on AddItem to AdminRole;  grant execute on UpdateItem to AdminRole;  --&  grant read, write on directory BGAMESSTORE\_XML to AdminRole;  grant execute on ExportItemsToXML to AdminRole;  grant execute on ImportItemsFromXML to AdminRole;  --&  grant execute on GetAllOrdersList to AdminRole;  grant execute on GetOrderInfo to AdminRole;  grant execute on GetOrderItems to AdminRole;  grant execute on SetOrderStatus to AdminRole;  grant execute on GetOrderStatuses to AdminRole;  --&  grant execute on GetAccountInfo to AdminRole;  grant execute on UpdateAccountInfo to AdminRole;  grant execute on DeactivateAdminAccount to AdminRole;  grant execute on CreateNewAdminAccount to AdminRole;  --& // not used in app  grant execute on DeactivateClientAccount to AdminRole;  --------------------------------------------------  -- роль пользователя (клиента)  create role ClientRole;  grant create session to ClientRole;  --&  grant execute on GetCatalog to ClientRole;  grant execute on GetItemInfo to ClientRole;  --&  grant execute on GetFavsItems to ClientRole;  grant execute on ToggleItemInFavs to ClientRole;  grant execute on MoveFavsToCart to ClientRole;  grant execute on ClearFavs to ClientRole;  --&  grant execute on GetCartItems to ClientRole;  grant execute on GetCartCost to ClientRole;  grant execute on ToggleItemInCart to ClientRole;  grant execute on UpdateCartItemCount to ClientRole;  grant execute on ClearCart to ClientRole;  grant execute on ValidateCart to ClientRole;  --&  grant execute on ValidateCart to ClientRole;  grant execute on MakeOrder to ClientRole;  grant execute on GetClientOrdersList to ClientRole;  grant execute on GetOrderInfo to ClientRole;  grant execute on GetOrderItems to ClientRole;  --&  grant execute on GetAccountInfo to ClientRole;  grant execute on GetClientAccountInfo to ClientRole;  grant execute on UpdateAccountInfo to ClientRole;  grant execute on UpdateClientAccountInfo to ClientRole;  grant execute on DeactivateClientAccount to ClientRole;  --& // not used in app  grant execute on AddToCart to ClientRole;  grant execute on RemoveFromCart to ClientRole;  --------------------------------------------------  -- роль гостя  create role GuestRole;  grant create session to GuestRole;  --&  grant execute on GetCatalog to GuestRole;  grant execute on GetItemInfo to GuestRole;  --&  grant execute on SignUpAsClient to GuestRole;  grant execute on SignIn to GuestRole;  --------------------------------------------------  -- профиль и пользователи БД  create profile UserProfile limit  password\_life\_time unlimited  idle\_time 180;    create user ADMIN identified by "ADMIN\_PASSWORD"  profile UserProfile  account unlock;  grant AdminRole to ADMIN;  create user CLIENT identified by "CLIENT\_PASSWORD"  profile UserProfile  account unlock;  grant ClientRole to CLIENT;  create user GUEST identified by "GUEST\_PASSWORD"  profile UserProfile  account unlock;  grant GuestRole to GUEST;  --------------------------------------------------  -- администратор по умолчанию  declare id Accounts.id%type;  begin  CreateNewAdminAccount('admin', 'admin', '?', '?', id);  commit;  end;  -------------------------------------------------- |