МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1–40 01 01 «Программное обеспечение информационных

технологий»

Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

База данных для программного средства «Музей» с технологией «Резервное копирование и восстановление»

Выполнил студент Коржова Валерия Сергеевна

(Ф.И.О.)

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

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

Нормоконтролер: пр.-стаж. Нистюк О.А.

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

Курсовой проект защищен с оценкой

Минск 2022

Содержание

[Введение 3](#_Toc122276099)

[1 Аналитический обзор литературы 4](#_Toc122276100)

[1.1 Аналитический обзор аналогов 4](#_Toc122276101)

[1.2 Вывод обзора аналогов 6](#_Toc122276102)

[2 Разработка модели базы данных 7](#_Toc122276103)

[3 Разработка необходимых объектов 8](#_Toc122276104)

[3.1 Таблицы 8](#_Toc122276105)

[3.2 Представления 10](#_Toc122276106)

[3.3 Хранимые процедуры 11](#_Toc122276107)

[3.4 Функции 15](#_Toc122276108)

[4 Описание используемых технологий 16](#_Toc122276109)

[4.1 Резервное копирование и восстановление 16](#_Toc122276110)

[4.2 Шифрование и дешифрование данных 17](#_Toc122276111)

[4.3 Мультимедийный тип данных 19](#_Toc122276112)

[5 Тестирование производительности 20](#_Toc122276113)

[6 Руководство пользователя 22](#_Toc122276114)

[Заключение 30](#_Toc122276115)

[Список литературы 31](#_Toc122276116)

[Приложение А 32](#_Toc122276117)

[Приложение Б 33](#_Toc122276118)

# Введение

Вокруг нас всегда много разных данных, например: телефонные номера, дела на день, записи на бумажках, стикерах и в блокнотах, опубликованные мысли разных людей, фотографии в смартфоне, и всё остальное, что можно прочитать, увидеть или услышать. Данные всегда играли важную роль в жизни людей и в разные времена их хранили по-разному. В наше время всё больше информации хранится в цифровом виде и для хранения такой информации отлично подходят различные базы данных. Их основными задачами являются: сохранение наших данных по запросу, изменение наших данных по запросу, поиск этих данные по запросу, защита информации от несанкционированного доступа, поддержание структуры данных для быстрого доступа к информации, масштабирование, возможность восстановления данных в случае удаления базы данных. Из этого можно сделать вывод, что использование базы данных упрощает управление и хранение данных. Одним из примеров успешного внедрения баз данных является использование их различными культурными учреждениями, например, музеями.

Сейчас есть множество удобных приложений, позволяющих без особых усилий посмотреть экспонаты, которые представлены в том или ином музее, узнать его историю, сохранить себе на устройство его изображение.

Целью моего курсового проекта является создание базы данных для музея.

Основными задачами курсовой работы являются:

* провести аналитический обзор литературы;
* спроектировать базу данных;
* реализовать функциональность базы данных;
* провести тестирование используемой технологии в базе данных;
* разработать приложение для работы с базой данных;
* написать руководство пользователя.

База данных — это организованная структура, предназначенная для хранения информации, систематизированная таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины.

Реляционная база данных — база данных, основанная на реляционной модели данных.

В качестве СУБД для базы данных была выбрана «Oracle 12c», в связи с ее простотой, производительностью и надежностью.

1. **Аналитический обзор литературы**

В соответствии с заданием курсового проекта следует создать базу данных для музея. Для того чтобы сформировать окончательные требования к проектируемому программному средству сначала рассмотрим аналоги из той же области, а так же технологии, которые используются в данном проекте.

* 1. **Аналитический обзор аналогов**

Постепенно с развитием программного обеспечения ЭВМ появились идеи создания управляющих систем, которые позволяли бы накапливать, хранить и обновлять взаимосвязанные данные по целому комплексу решаемых задач. Эти идеи нашли свое воплощение в системах управления базами данных (СУБД). СУБД взаимодействуют не с локальными, а взаимосвязанными по информации массивами, называемыми базами данных. С появлением персональных компьютеров СУБД становятся наиболее популярным средством обработки табличной информации.

Можно рассмотреть структуру организации работы базы данных музеев.

Рассмотрим в качестве примера сайта художественного музея «Национальный художественный музей Республики Беларусь», изображенного на рисунке 1.1.

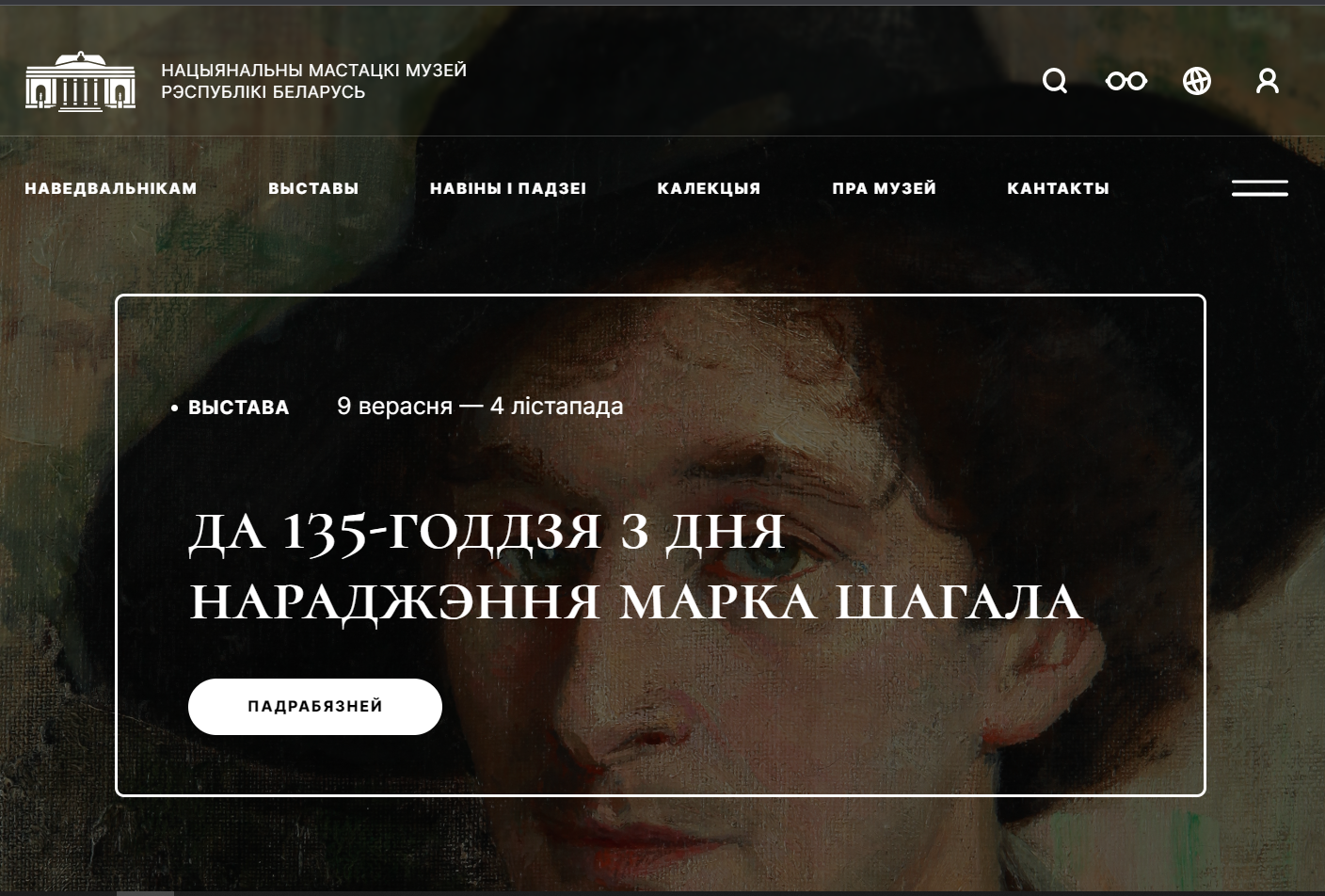


Рисунок 1.1 – Сайт музея «Национальный художественный музей Республики Беларусь»

Раньше, для того, чтобы посетить какой-либо музей, нужно было купить билеты на выставку, приехать в локацию, пройти вдоль залов с выставками. Сейчас же можно посетить данные культурные заведения не выходя из дома, в любое время дня. Все это стало возможно благодаря базам данных с разработанным для них программным обеспечением, позволяя хранить все экспонаты в одном месте.

Тут видно, что на данном сайте есть возможность просмотра экспонатов. Кликнув на раздел «Коллекция», пользователь может ознакомиться с ассортиментом картин. Их можно отсортировать по периоду написания, по автору и по филиалу, где можно увидеть их вживую. Эта функциональность продемонстрирована на рисунке 1.2.

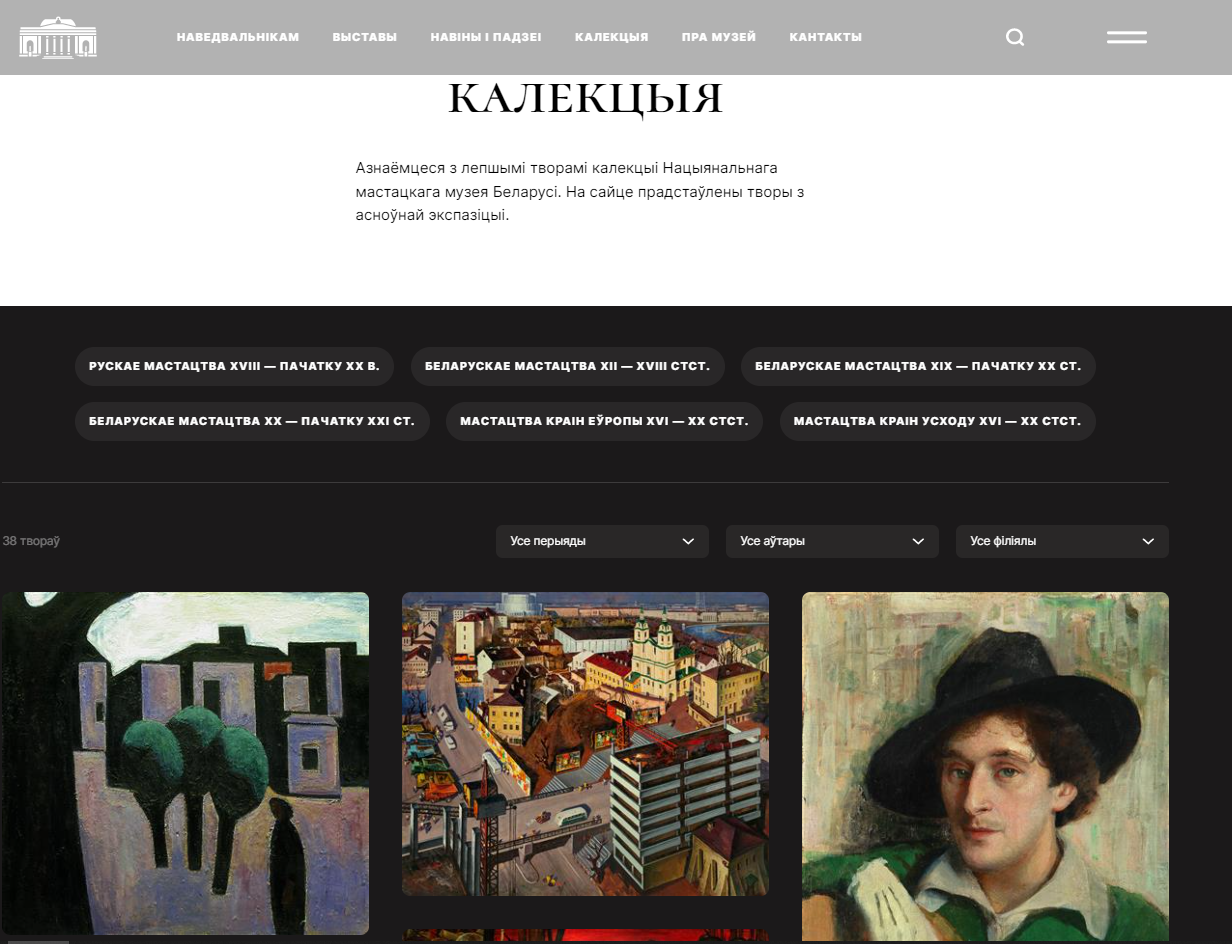


Рисунок 1.2 – Информация на вкладке «Коллекция»

Также на данном сайте осуществлена возможность добавления информации о пользователе, такой как: имя пользователя, его электронная почта, номер мобильного телефона и опция смены пароля, для осуществления которой нужно ввести старый пароль, новый пароль и подтвердить новый пароль. Эта функциональность изображена на рисунке 1.3.

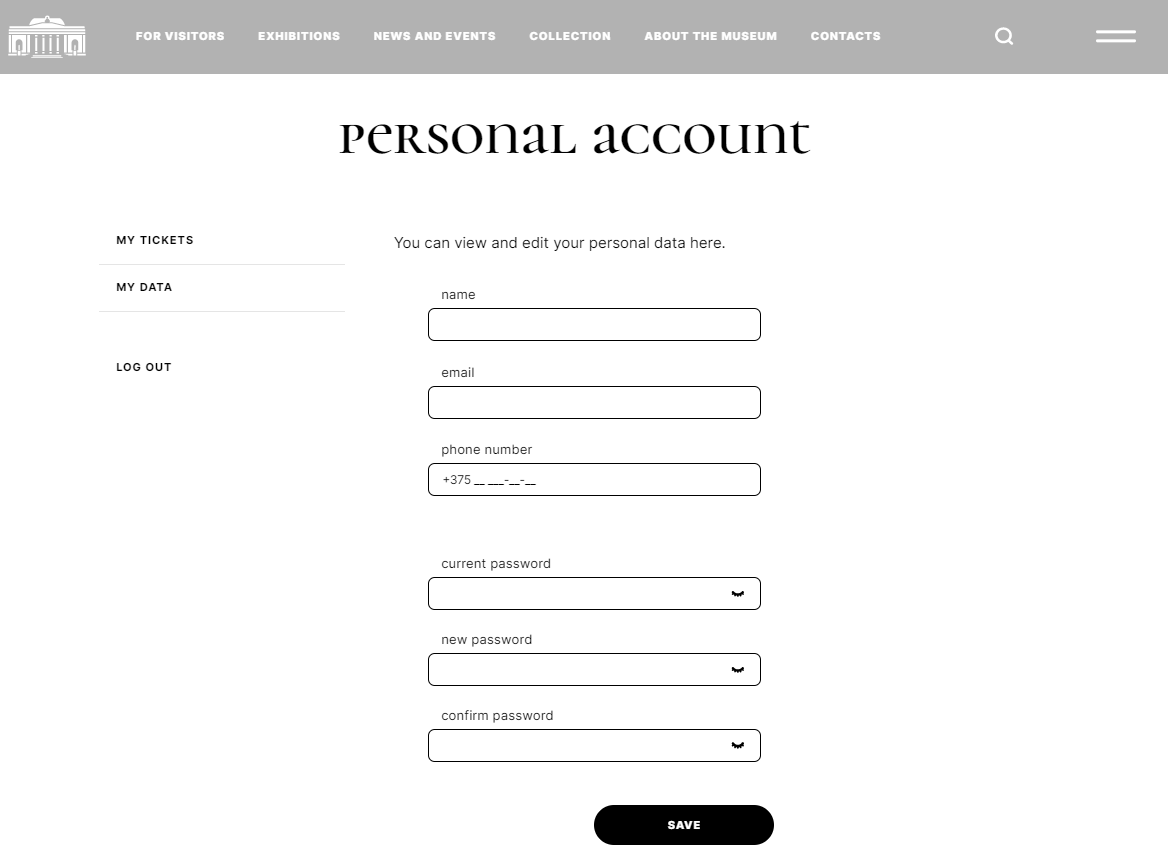


Рисунок 1.3 – Возможности изменения информации о пользователе

Недостатком данного ресурса является отсутствие способа добавления понравившихся картин в свою коллекцию.

* 1. **Вывод обзора аналогов**

Ознакомившись с представленными выше аналогом, была определена необходимая структура проектируемой базы данных. Рассмотрев функциональные особенности аналога, был сформирован общий функционал необходимый для разработки процедур.

1. **Разработка модели базы данных**

При разработке приложения для курсового проекта была использована база данных Oracle 12c.

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

Для реализации базы данных для музея было разработано 7 таблиц. Диаграмма базы данных со структурой связей представлена в приложении А.

Создание перечисленных таблиц с установлением внешних связей описано в главе 3.

1. **Разработка необходимых объектов**

Для создания необходимых объектов в данном курсовом проекте предварительно была создана с помощью Database Configuration Assistant подключаемая база данных с именем «museum», в нем табличное пространство «museum», схему пользователя «lera», профиль безопасности, роль и администратора базы данных.

При разработке курсового проекта понадобились следующие объекты:

* Таблицы;
* Представления;
* Хранимые процедуры;
* Функции;
  1. **Таблицы**

Таблицы являются неотъемлемой частью любой реляционной базы данных. Непосредственно именно в них хранятся все необходимые для работы данные. Ниже мы рассмотрим их подробнее.

Таблица userroles состоит из двух столбцов (таблица 3.1):

Таблица 3.1 – Столбцы таблицы userroles

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | идентификатор роли, первичный ключ | number |
| name | название роли | nvarchar2 |

Таблица users представляет список пользователей состоит из столбцов (таблица 3.2):

Таблица 3.2 – Столбцы таблицы users

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| email | электронная почта пользователя, первичный ключ | nvarchar2 |
| username | имя пользователя | nvarchar2 |
| password | пароль пользователя | nvarchar2 |
| role\_id | идентификатор роли, внешний ключ к таблице userroles | number |

Таблица authors представляет список авторов картин, таблица состоит из столбцов (таблица 3.3):

Таблица 3.3 – Столбцы таблицы authors

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | идентификатор автора картины | number |

Продолжение таблицы 3.3

|  |  |  |
| --- | --- | --- |
| Name | имя автора | nvarchar2 |
| info | описание автора | nvarchar2 |

Таблица categories представляет список категорий картин, состоит из столбцов (таблица 3.4):

Таблица 3.4 – Столбцы таблицы categories

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | идентификатор категории, первичный ключ | number |
| name | название категории | nvarchar2 |
| info | информация о категории | nvarchar2 |

Таблица pictures представляет список картин, состоит из столбцов (таблица 3.5):

Таблица 3.5 – Столбцы таблицы pictures

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | идентификатор картины, первичный ключ | number |
| name | название картины | nvarchar2 |
| author\_id | идентификатор автора, внешний ключ к таблице authors | number |
| category\_id | идентификатор категории, внешний ключ к таблице categories | number |
| year | год создания картины | number |
| info | описание картины | nvarchar2 |
| picture | двоичное представление фотографии картины | blob |

Таблица collections представляет список коллекций картин, который пользователь добавил, состоит из столбцов (таблица 3.6):

Таблица 3.6 – Столбцы таблицы collections

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | идентификатор коллекции, первичный ключ | number |
| name | название коллекции | nvarchar2 |
| email | электронная почта пользователя, является внешним ключом к таблице users | nvarchar2 |

Таблица collection\_pictures представляет список картин, которые добавлены в коллекции пользователей, состоит из столбцов (таблица 3.7):

Таблица 3.7 – Столбцы таблицы collection\_pictures

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| id | идентификатор картины в коллекции, первичный ключ | number |
| collection\_id | идентификатор коллекции, внешний ключ к таблице collections | number |
| picture\_id | идентификатор картины, внешний ключ к таблице pictures | number |

Код создания каждой из таблиц можно увидеть в Приложении Б.

* 1. **Представления**

Представления являются необязательной частью реляционной базы данных, однако они позволяют быстрее обращаться к информации. Далее изложены описание созданных представлений, код их создания представлен в Приложении В.

Представление user\_userrole\_view состоит из столбцов (таблица 3.8):

Таблица 3.8 – Столбцы представления user\_userrole\_view

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| email | электронная почта пользователя | nvarchar2 |
| username | имя пользователя | nvarchar2 |
| name | название роли | nvarchar2 |

Представление full\_user\_userrole\_view состоит из столбцов (таблица 3.9):

Таблица 3.9 – Столбцы представления full\_user\_userrole\_view

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| email | электронная почта пользователя | nvarchar2 |
| username | имя пользователя | nvarchar2 |
| password | пароль пользователя | nvarchar2 |
| name | название роли | nvarchar2 |

Представление picture\_view состоит из столбцов (таблица 3.10):

Таблица 3.10 – Столбцы представления picture\_view

|  |  |  |
| --- | --- | --- |
| Наименование | Описание | Тип |
| name | имя автора | nvarchar2 |
| info | описание автора | nvarchar2 |

Продолжение таблицы 3.10

|  |  |  |
| --- | --- | --- |
| author\_name | имя автора | nvarchar2 |
| categories\_name | название категории | nvarchar2 |
| id | идентификатор картины | number |
| year | год написания картины | number |
| picture | двоичное представление фотографии картины | blob |

* 1. **Хранимые процедуры**

Использование хранимых процедур позволяет ограничить либо вообще исключить непосредственный доступ пользователей к таблицам базы данных, оставив пользователям только разрешения на выполнение хранимых процедур, обеспечивающих косвенный и строго регламентированный доступ к данным.

Всего было разработано 18 процедур для работы с данными. Их можно разделить на 5 категорий: добавление данных, удаление данных, их изменение, вывод данных и работа с xml. Краткое описание процедур описано далее.

Пример создания хранимой процедуры add\_picture, которая добавляет в базу данных новую картину, представлен на листинге 3.11.

|  |
| --- |
| create or replace procedure add\_picture  (i\_name in pictures.name%type,  i\_author\_name in authors.name%type,  i\_category\_name in categories.name%type,  i\_year in pictures.year%type,  i\_info in pictures.info%type,  i\_picture in pictures.picture%type) is  picture\_count number;  author\_id authors.id%type;  category\_id categories.id%type;  begin  select authors.id into author\_id from authors where upper(i\_author\_name) = upper(authors.name);  select categories.id into category\_id from categories where upper(i\_category\_name) = upper(categories.name);  select count(\*) into picture\_count from pictures where upper(pictures.name) = upper(i\_name) and pictures.author\_id = author\_id;  if (picture\_count = 0) then insert into pictures(name, author\_id, category\_id, year, info, PICTURE) values (i\_name, author\_id, category\_id, i\_year, i\_info, i\_picture); commit;  else raise\_application\_error(-20004, 'picture already exists'); end if;  exception  when no\_data\_found  then raise\_application\_error(-20020, 'This collection or author doesnt exists');  end add\_picture; |

Листинг 3.11 – Описание процедуры add\_picture

Процедура принимает шесть параметров – название картины (i\_name), имя автора (i\_author\_name), название категории (i\_category\_name), год написания картины (i\_year), информация о картине (i\_info) и изображение (i\_picture). Процедура находит идентификаторы автора и категории с помощью точной выборки из двух таблиц authors и categories и добавляет данные в таблицу pictures.

Пример создания хранимой процедуры delete\_picture, которая удаляет картину из базы данных, представлен на листинге 3.12.

|  |
| --- |
| create or replace  procedure delete\_picture  (i\_id in pictures.id%type)  is  picture\_count number;  begin  select count(\*) into picture\_count from pictures where PICTURES.ID = i\_id;  if (picture\_count = 1) then  delete pictures where PICTURES.ID = i\_id;  else  raise\_application\_error(-20009, 'cannot delete, picture doesnt exist');  end if;  end delete\_picture; |

Листинг 3.12 – Описание процедуры delete\_picture

Представленная выше процедура принимает один параметр – идентификатор картины, осуществляет точную выборку из таблицы pictures, и, если значение будет найдено, удаляет строку из таблицы.

Пример создания хранимой процедуры update\_user, которая изменяет данные пользователя, представлен на листинге 3.13.

|  |
| --- |
| create or replace  procedure update\_user  (i\_email in users.email%type,  i\_username in users.username%type,  i\_password in users.password%type) is  user\_count number; begin  select count(\*) into user\_count from users where upper(i\_email) = upper(users.EMAIL);  if (user\_count = 1) then update users set users.USERNAME = i\_username, users.PASSWORD = encrypt\_password(i\_password) where upper(i\_email) = upper(users.email);  commit;  else  raise\_application\_error(-20006, 'cannot update user because it doesnt exist');  end if;  end update\_user; |

Листинг 3.13 – Описание процедуры update\_user

Представленная выше процедура принимает три параметра – электронная почта пользователя, его имя, пароль и осуществляет точную выборку из таблицы users, и, если значение будет найдено, обновляет данные в таблице.

Пример создания хранимой процедуры full\_list\_users\_collections, которая выводит коллекции пользователя с картинами, представлен на листинге 3.14.

|  |
| --- |
| create or replace  procedure full\_list\_users\_collections  (i\_email in users.email%type,  o\_collection\_cursor out sys\_refcursor)  is  begin  open o\_collection\_cursor for  select COLLECTIONS.ID,  COLLECTIONS.NAME,  COLLECTIONS.EMAIL,  PICTURE\_VIEW.ID as Pictures\_id,  PICTURE\_VIEW.NAME as Picture\_name,  PICTURE\_VIEW.AUTHOR\_NAME,  PICTURE\_VIEW.CATEGORIES\_NAME,  PICTURE\_VIEW.YEAR,  PICTURE\_VIEW.INFO,  PICTURE\_VIEW.PICTURE from COLLECTIONS  full join collection\_pictures on  collections.id = COLLECTION\_ID  join picture\_view on  COLLECTION\_PICTURES.PICTURE\_ID = PICTURE\_VIEW.ID  where upper(i\_email) = upper(collections.email)  order by COLLECTION\_ID;  end full\_list\_users\_collections; |

Листинг 3.14 – Описание процедуры full\_list\_users\_collections

Представленная выше процедура принимает два параметра – входной параметр электронная почта пользователя и выходной параметр типа курсор, который выполняет выборку из нескольких таблиц.

Для осуществления импорта xml предварительно нужно создать директорию. Код создания директории представлен на листинге 3.15.

|  |
| --- |
| CREATE OR REPLACE  DIRECTORY MYDIR  AS 'C:\XMLFILE'; |

Листинг 3.15 – Создание директории MYDIR

Пример создания хранимой процедуры import\_xml, которая выполняет импорт xml, представлен на листинге 3.16.

|  |
| --- |
| create or replace  procedure import\_xml  as  rc sys\_refcursor;  doc DBMS\_XMLDOM.DOMDocument;  begin  OPEN rc FOR SELECT \* FROM users;  doc := DBMS\_XMLDOM.NewDOMDocument(XMLTYPE(rc));  DBMS\_XMLDOM.WRITETOFILE(doc, 'MYDIR/userexport.xml');  commit;  exception  when others then  raise\_application\_error(-20017,'error during xml import');  end import\_xml; |

Листинг 3.16 – Описание процедуры import\_xml

Пример создания хранимой процедуры export\_xml, которая выполняет экспорт xml, представлен на листинге 3.17.

|  |
| --- |
| create or replace  procedure export\_xml  as  begin  insert into USERS (EMAIL, USERNAME, PASSWORD, ROLE\_ID)  SELECT \*  FROM XMLTABLE('/ROWSET/ROW'  PASSING XMLTYPE(BFILENAME('MYDIR','userexport.xml'),  NLS\_CHARSET\_ID('CHAR\_CS'))  COLUMNS EMAIL nvarchar2(100) PATH 'EMAIL',  USERNAME nvarchar2(100) PATH 'USERNAME',  PASSWORD nvarchar2(255) PATH 'PASSWORD',  ROLE\_ID number(10) PATH 'ROLE\_ID');  commit;  exception  when others then  raise\_application\_error(-20018,'error during xml export');  end export\_xml; |

Листинг 3.17 – Описание процедуры export\_xml

Далее представлен список всех процедур и их описание:

– register\_user – добавление нового пользователя в базу данных;

– log\_in\_user – авторизация пользователя;

– delete\_user – удаление пользователя;

– update\_user – обновление данных о пользователе;

– add\_author – добавление автора;

– add\_category – добавление категории;

– add\_picture – добавление картины;

– delete\_category – удаление категории;

– delete\_author – удаление автора;

– delete\_picture – удаление картины;

– add\_collection – добавление коллекции;

– delete\_collection – удаление коллекции;

– full\_list\_users\_collections – получение списка коллекций юзера с картинами;

– list\_users\_collections – получение списка коллекций пользователя;

– add\_picture\_to\_collection – добавление картины в коллекцию пользователя;

– delete\_picture\_from\_collection – удаление картины из коллекции;

– import\_xml – импорт в данных из xml в базу данных;

– export\_xml – экспорт данных из базы данных в xml;

* 1. **Функции**

Функции базы данных oracle выполняют те же самые задачи, что и процедуры. Главным их различием является то, что функции могут возвращать значения. Это может быть полезно в некоторых ситуациях. В рамках данного курсового проекта было разработано три функции: шифрование пароля пользователя, его дешифрование и поиск картин по названию. Листинги первой и второй функции представлены в главе 4, а поиск картин по названию представлен на листинге 3.18.

|  |
| --- |
| create or replace  function search\_picture  (i\_name in PICTURES.name%type)  return sys\_refcursor  is  o\_picture\_cursor sys\_refcursor;  begin  open o\_picture\_cursor for  select \* from PICTURE\_VIEW  where upper(NAME) like upper('%' || i\_name|| '%');  return o\_picture\_cursor;  end; |

Листинг 3.18 – Описание функции search\_picture

Функция принимает один параметр – название картины (i\_name) и возвращает переменную типа курсор, в которой хранятся значения из представления picture\_view.

1. **Описание используемых технологий**

В данном разделе рассматривается применяемые в курсовом проекте технологии, а именно резервное копирование и восстановление, шифрование и дешифрование пароля пользователя и технология хранения информации представленной в мультимедийном виде, а именно изображения.

* 1. **Резервное копирование и восстановление**

Резервная копия – это набор всех файлов, которые необходимы для восстановления корректной работы базы данных при частичном удалении файлов или нарушении структуры данных.

Существует два способа резервного копирования: холодное резервное копирование и горячее резервное копирование. Холодным резервным копированием называют копирование файлов, в то время, когда база данных отключена, то есть является недоступной для пользователей. Горячее резервное копирование – это создание копии файлов, когда база данных включена и доступна пользователям. Был выбран второй вариант реализации.

Также существует способ реализации с помощью средств операционной системы, копируя каждый отдельный файл данных, или при помощи встроенного в установочный пакет oracle компонента Recovery Manager (rman). С целью демонстрации удобства и быстроты разработки был выбран второй вариант реализации.

Включение режима резервного копирования снижает производительность базы данных: во-первых, увеличивается объём журналов, а во-вторых, если вдруг в режиме резервного копирования произойдёт сбой, восстановление будет более продолжительным, т. к. заголовки файлов данных не обновляются.

Для того, чтобы создать резервную копию необходимо подготовить файл со специфичным для данного компонента расширением – .rman, в котором описан код создания полной копии базы данных с тэгом «fullbackup», листинг 4.1.

|  |
| --- |
| run  {  BACKUP DATABASE  TAG 'fullbackup';  } |

Листинг 4.1 – Содержимое файла full\_backup\_database.rman

Для того, чтобы резервная копия осуществлялась на регулярной основе, был написан batch скрипт, который запускает файл .rman ежедневно, как рекомендовано самой компанией oracle, листинг 4.2

|  |
| --- |
| @echo off  echo backup database in progress  echo -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-  RMAN target sys/Vv1542139 @X:\full\_backup\_database.rman; |

Листинг 4.2 – Содержимое файла full\_backup\_database.bat

Для осуществления восстановления файлов базы данных был разработан файл restore\_database.rman, содержимое которого показано на листинге 4.3.

|  |
| --- |
| startup mount;  run  {  restore database;  recover database;  }  sql 'alter database open'; |

Листинг 4.3 – Содержимое файла restore\_database.rman

Чтобы разрешить запустить данный файл как последовательный набор команд, был реализован batch скрипт restore\_database.bat, листинг 4.4.

|  |
| --- |
| @echo off  echo oraсle restore in progress...  RMAN target sys/Vv1542139  @X:\restore\_database.rman;  PAUSE; |

Листинг 4.4 – Содержимое файла restore\_database.bat

* 1. **Шифрование и дешифрование 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 путем применения специальный ключа, котоый генерируется при операции шифрования данных. Шифрование является важной частью обеспечения безопасности данных, потому что предотвращают доступ к данным нежелательным лицам. Oracle создал специальный пакет – DBMS\_CRYPTO, который предоставляет интерфейс для шифрования и расшифрования хранимых данных и может использоваться в сочетании с программами PL/SQL, выполняющими сетевые коммуникации. Он обеспечивает поддержку нескольких стандартных алгоритмов шифрования и хэширования, включая алгоритм шифрования Advanced Encryption Standard (AES). Для того, чтобы использовать данный пакет, нужно от имени главного администратора дать роли или пользователю соответствующую привелегию, листинг 4.5.

|  |
| --- |
| grant EXECUTE ON sys.dbms\_crypto  to admin\_role; |

Листинг 4.5 – Привилегия на пакет sys.dbms\_crypto

Функцию шифрования пароля пользователя можно увидеть в листинге 4.6.

|  |
| --- |
| create or replace  FUNCTION encrypt\_password  (i\_password in users.password%type)  RETURN users.password%type  IS  my\_key VARCHAR2(2000) := '2022202220222022';  my\_val VARCHAR2(2000) := i\_password;  my\_mod NUMBER := sys.DBMS\_CRYPTO.encrypt\_aes128  + sys.DBMS\_CRYPTO.chain\_cbc  + sys.DBMS\_CRYPTO.pad\_pkcs5;  encrypted\_password RAW(2000);  BEGIN  encrypted\_password := sys.DBMS\_CRYPTO.encrypt(utl\_i18n.string\_to\_raw(my\_val, 'AL32UTF8'), my\_mod,  utl\_i18n.string\_to\_raw(my\_key, 'AL32UTF8'));  RETURN RAWTOHEX(encrypted\_password);  END encrypt\_password; |

Листинг 4.6 – Функция encrypt\_password

Функцию дешифрования пароля пользователя можно увидеть в листинге 4.7.

|  |
| --- |
| create or replace  FUNCTION decrypt\_password  (i\_password in users.password%type)  return users.password%type  IS  my\_key VARCHAR2(2000) := '2022202220222022';  my\_val RAW(2000) := HEXTORAW(i\_password);  my\_mod NUMBER := DBMS\_CRYPTO.encrypt\_aes128  + DBMS\_CRYPTO.chain\_cbc  + DBMS\_CRYPTO.pad\_pkcs5;  decrypted\_password RAW(2000);  BEGIN  decrypted\_password :=  DBMS\_CRYPTO.decrypt(my\_val,  my\_mod,  utl\_i18n.string\_to\_raw(my\_key, 'AL32UTF8'));  RETURN utl\_i18n.raw\_to\_char(decrypted\_password);  END decrypt\_password; |

Листинг 4.7 – Функция decrypt\_password

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

Результат функции преобразуем к строке при помощи функции RAW\_TO\_CHAR пакета UTL\_I18N.

* 1. **Мультимедийный тип данных**

Мультимедийный тип данных окружает человека в повседневной жизни: при использовании интернета, при прослушивании музыки, при просмотре фильма в кинотеатре.

Для работы с данными большого объема СУБД Oracle предоставляет типы данных BLOB, CLOB, NCLOB и BFILE. Здесь LOB означает Large OBject, или большой объект, и далее по тексту термины LOB и "большой объект" взаимозаменяемы. По сути, большой объект – это абстрактный тип для манипуляции данными большого объема внутри БД, а типы BLOB, CLOB, NCLOB и BFILE - его конкретные реализации.

В данном курсовом проекте технология мультимедийных типов данных отражена в необходимости хранить изображения, без которых все приложение потеряло бы смысл. Речь идет о таблице pictures, а конкретно об его столбце picture. Для хранения в поле изображений в СУБД применяется тип BLOB. Тип BLOB представляет из себя массив двоичных данных, поэтому процесс сводился к разбиению мультимедийных файлов на массив битов при загрузке их в базу данных, и к сборке изображения из массива битов на обратном пути. Эти преобразования представлены в листинге 4.8.

|  |
| --- |
| public void addPicture(String name, String authorName, String categoryName, Integer year, String info, String picturePath) {  try {  CallableStatement cs = connection.prepareCall("{call add\_picture(?, ?, ?, ?, ?, ?)}");  cs.setString(1, name);  cs.setString(2, authorName);  cs.setString(3, categoryName);  cs.setInt(4, year);  cs.setString(5, info);  FileInputStream fileInputStream = new FileInputStream(picturePath);  cs.setBinaryStream(6, fileInputStream, fileInputStream.available());  cs.executeQuery();  cs.close();  } catch (SQLException e) {  logger.error(e.getMessage());  } catch (FileNotFoundException e) {  e.printStackTrace();  } catch (IOException e) {  e.printStackTrace();  }  } |

Листинг 4.8 – Загрузка изображения в базу данных

Таким образом, после превращения изображения в массив битов его запросто можно хранить в базе данных.

1. **Тестирование производительности**

Тестирование производительности проводилось на таблице Authors, которая заполнена на 100000 строк. Анонимный блок по заполнению строк представлен в листинге 5.1.

|  |
| --- |
| declare  begin  for a in 1 .. 100000  loop  add\_author('name' || a, 'info' || a);  end loop;  end; |

Листинг 5.1 – Заполнение таблицы Authors данными

Далее осуществляется выборка автора с заданным сочетанием букв в имени и описании, листинг 5.2.

|  |
| --- |
| explain plan for SELECT name FROM authors WHERE name like '%name1%' and info like '%a%';  select \* from table(DBMS\_XPLAN.DISPLAY()); |

Листинг 5.2 – Выборка автора с заданными критериями

Стоимость данной операции представлена на рисунке 5.1.

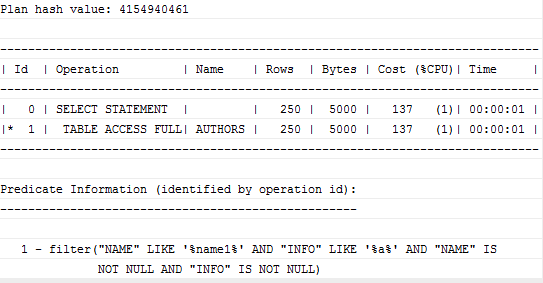


Рисунок 5.1 – Стоимость запроса

Теперь, чтобы повысить производительность запроса, был создан индекс, который представлен в листинге 5.3.

|  |
| --- |
| create index authors\_i on authors(name, info); |

Листинг 5.3 – Создание индекса authors\_i

После создание индекса снова была снова посчитана стоимость запроса. Данная операция представлена на рисунке 5.2.

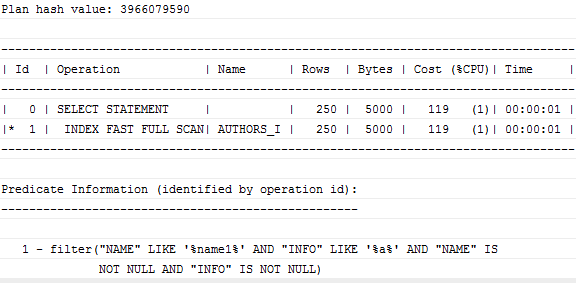


Рисунок 5.2 – Стоимость запроса после создания индекса

Как видно из рисунка 5.2, стоимость запроса уменьшилась. Таким образом, с помощью индексов мы можем увеличить производительность наших процедур, что позволит нам быстрее получать необходимую информацию.

1. **Руководство пользователя**

Приложение представляет собой web-сайт, с которым взаимодействуют пользователи. При запуске приложения открывается главная страница, рисунок 6.1.



Рисунок 6.1 – Главная страница

Для того, чтобы войти или зарегестрироваться на сайте необходимо нажать кнопку «Log in / Registration». После этого откроется страница входа в приложение, рисунок 6.2.

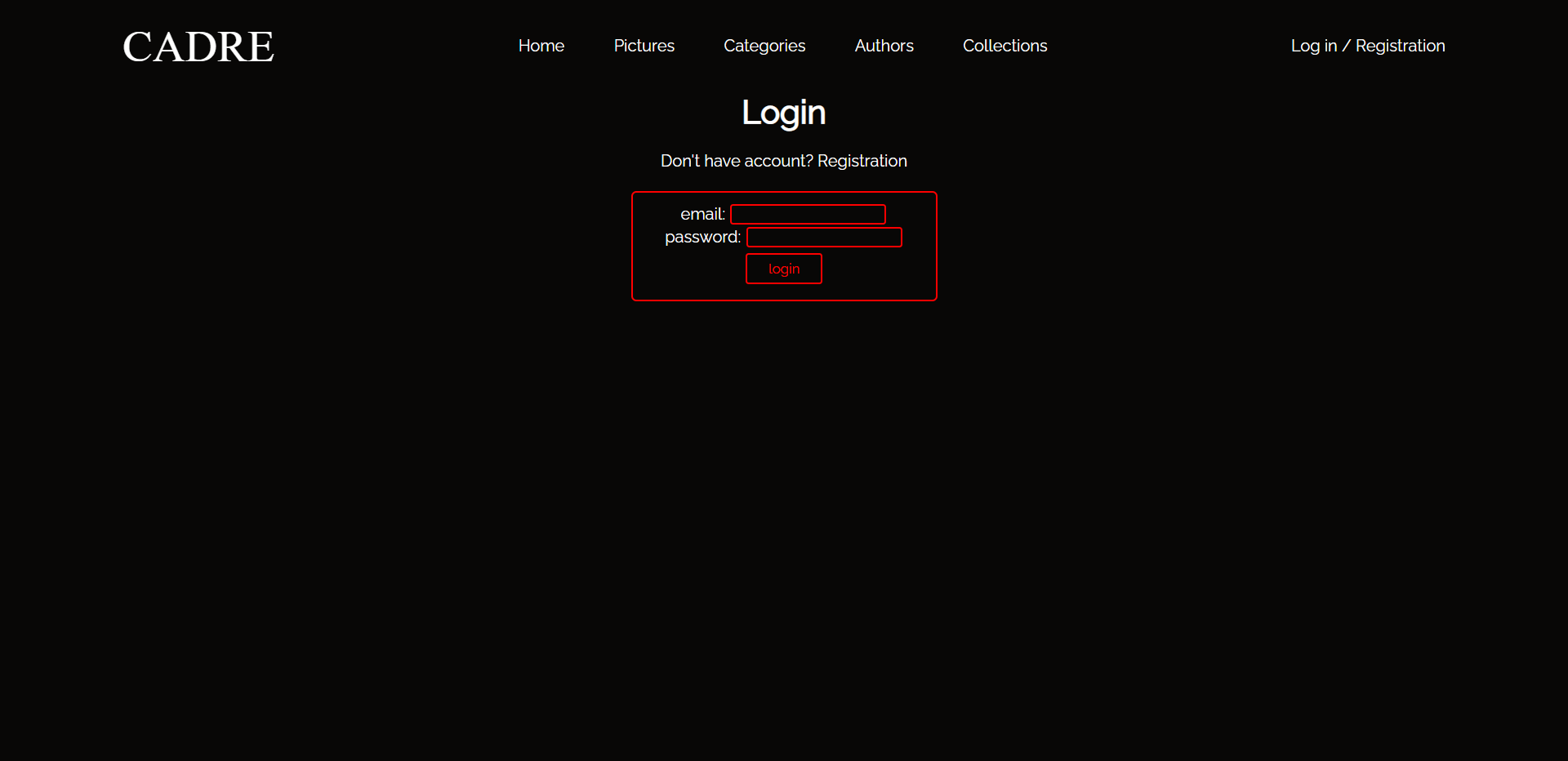


Рисунок 6.2 – Страница входа

Для перехода к регистрации пользователя необходимо нажать кнопку «Don’t have account? Registration». Откроется страница, представленная на рисунке 6.3.

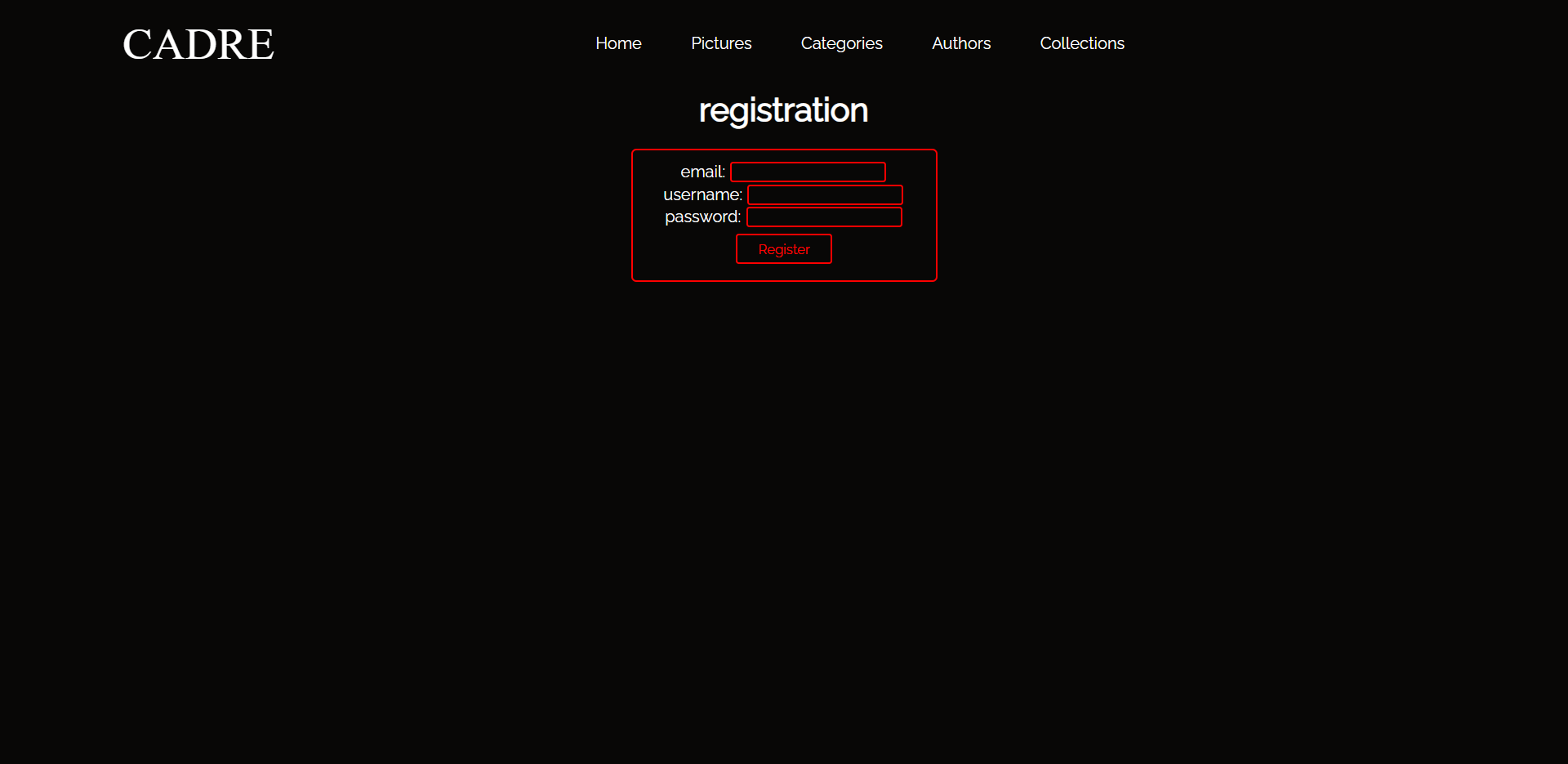


Рисунок 6.3 – Страница регистрации

Введем в поля значение электронной почты, пользователь с которой уже существует, рисунок 6.4.

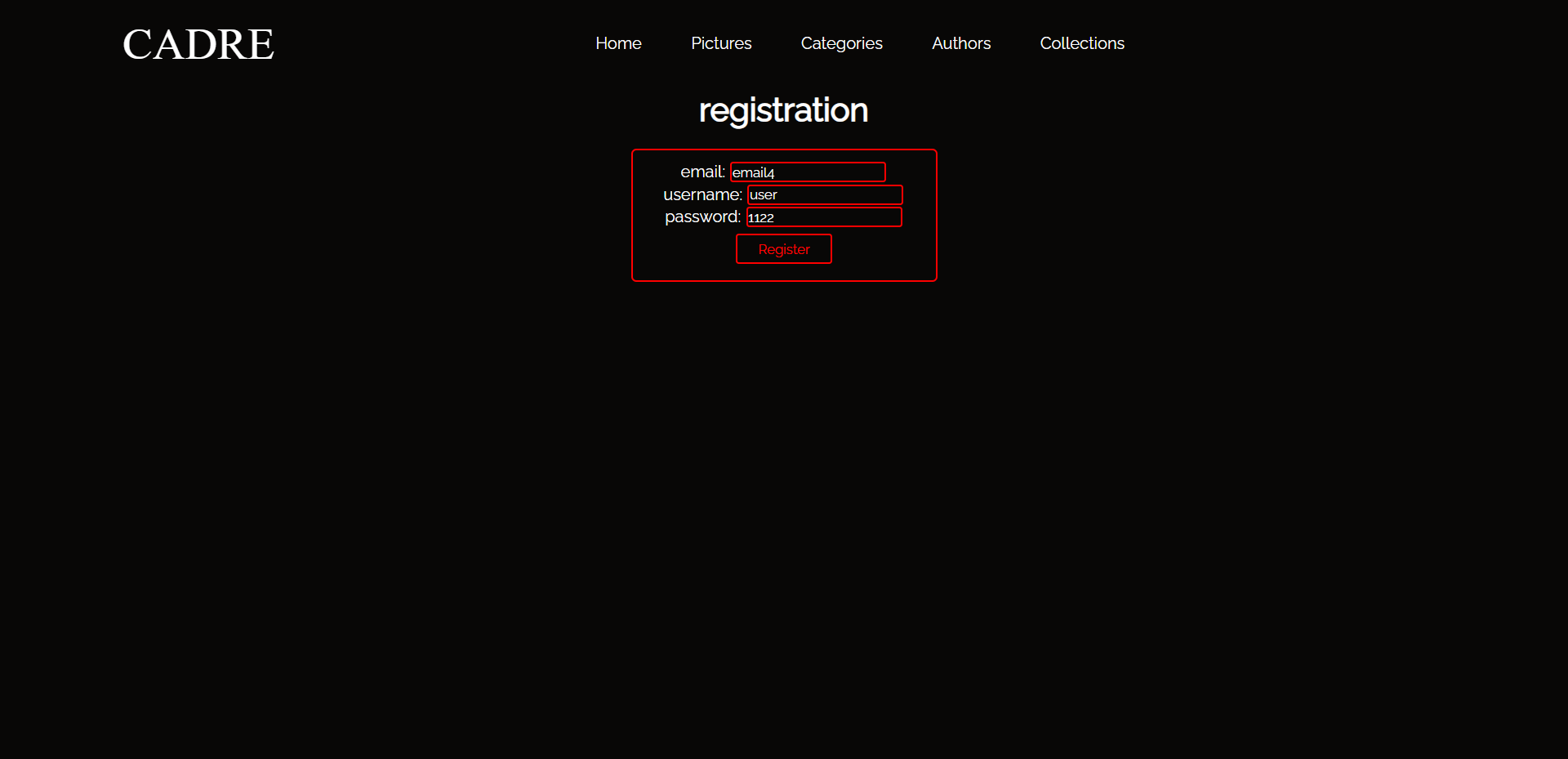


Рисунок 6.4 – Страница регистрации с введенными значениями

При этом в консоль выводится соответствующая ошибка, рисунок 6.5.

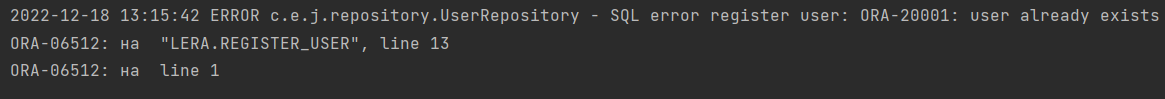


Рисунок 6.5 – Сообщение об ошибке

Вернемся на страницу входа и введем электронную почту и пароль пользователя, которого не существует, рисунок 6.6.

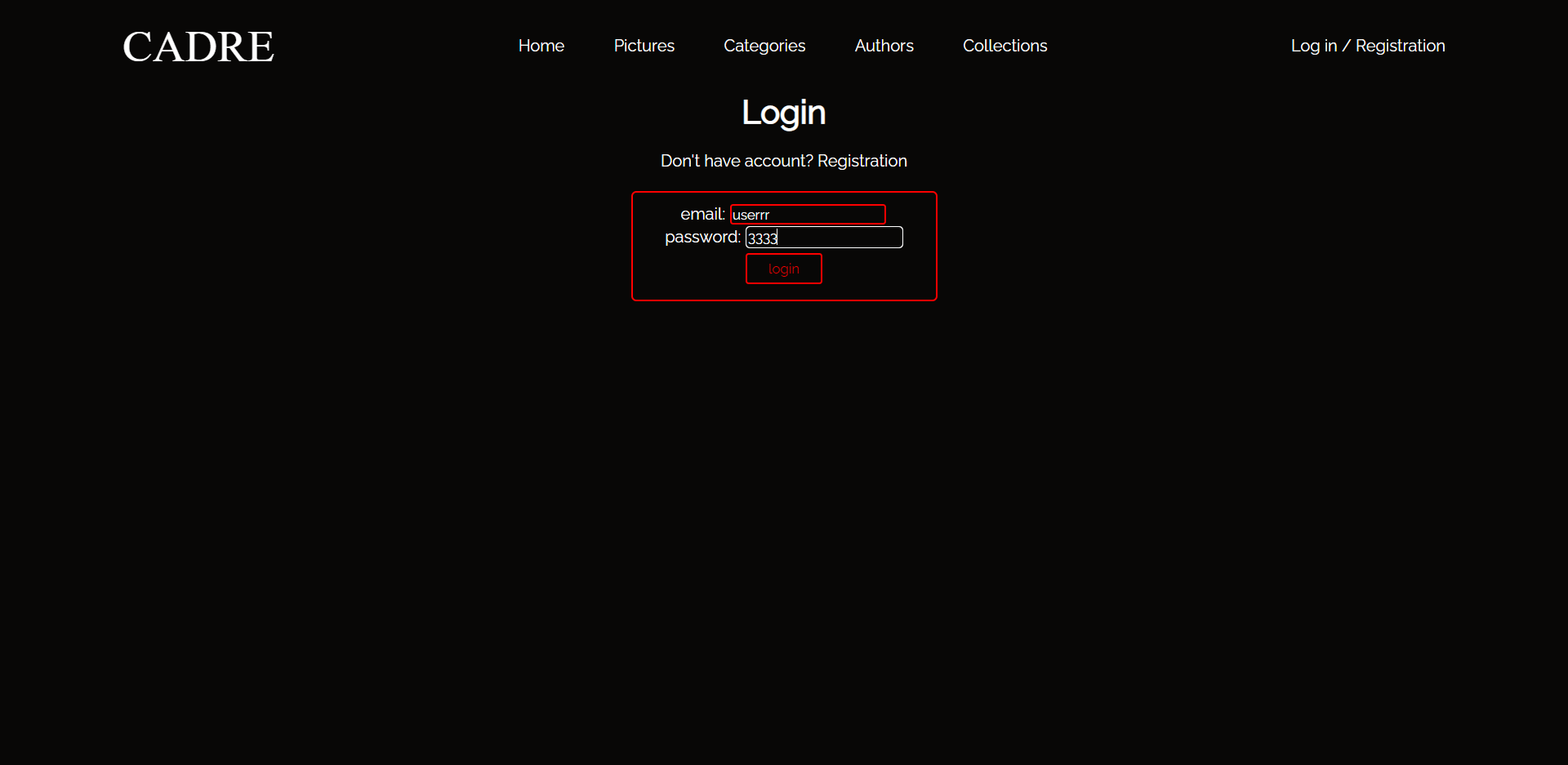


Рисунок 6.6 – Страница входа

При этом в консоль выводится соответствующая ошибка, рисунок 6.7.

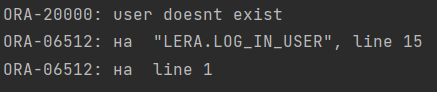


Рисунок 6.7 – Сообщение об ошибке

Выполним вход в приложение за администратора. Основное отличие аккаунта администратора от аккаунта пользователя состоит в появление у администратора дополнительной страницы «Admin», рисунок 6.8.

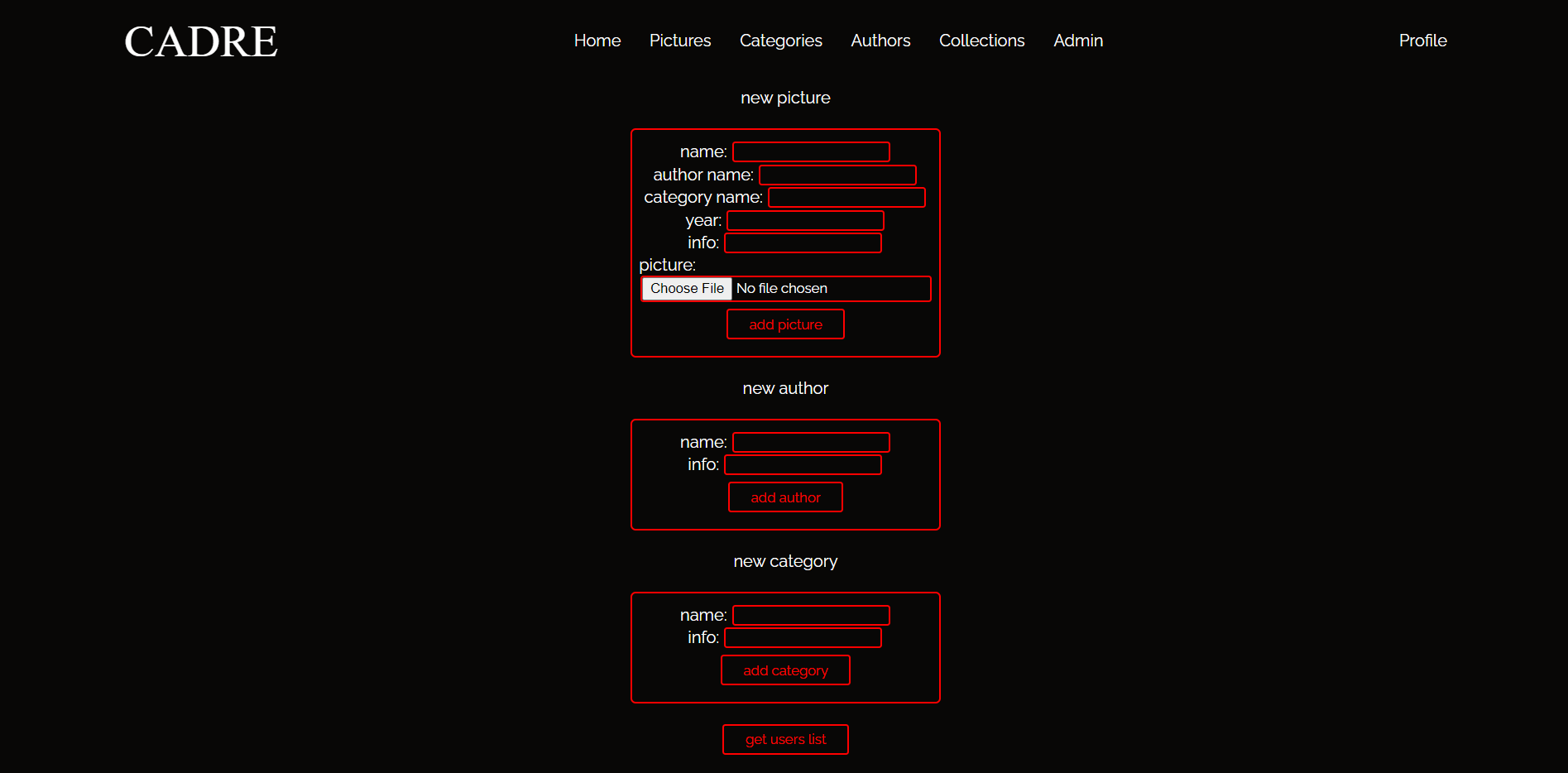


Рисунок 6.8 – Страница «Admin»

При нажатии на кнопку «get users list» выводится список с информацией о пользователях, рисунок 6.9.



Рисунок 6.9 – Информация о пользователях

При нажатии на кнопку «delete» напротив соответствующего пользователя он удаляется из базы данных. Продемонстрируем это на примере пользователя с электронной почтой «email23», рисунок 6.10.

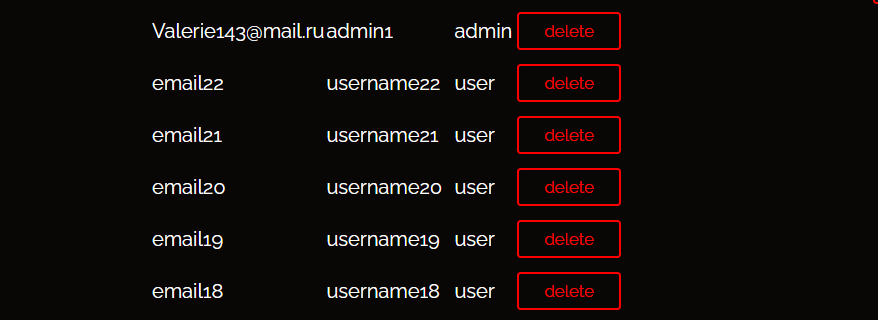


Рисунок 6.10 – Пользователь с данными «email23» отсутствует

Добавим новую картину. Для этого заполним форму «new picture» всеми необходимыми данными и наждем кнопку «add picture», рисунок 6.11.



Рисунок 6.11 – Добавление картины

Если нажать еще раз на кнопку, появится сообщение об ошибке, потому что картину с таким же названием и именем автора уже существует, рисунок 6.12.

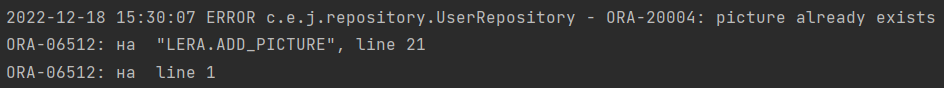


Рисунок 6.12 – Сообщение об ошибке

Убедимся, что картина добавлена, рисунок 6.13.

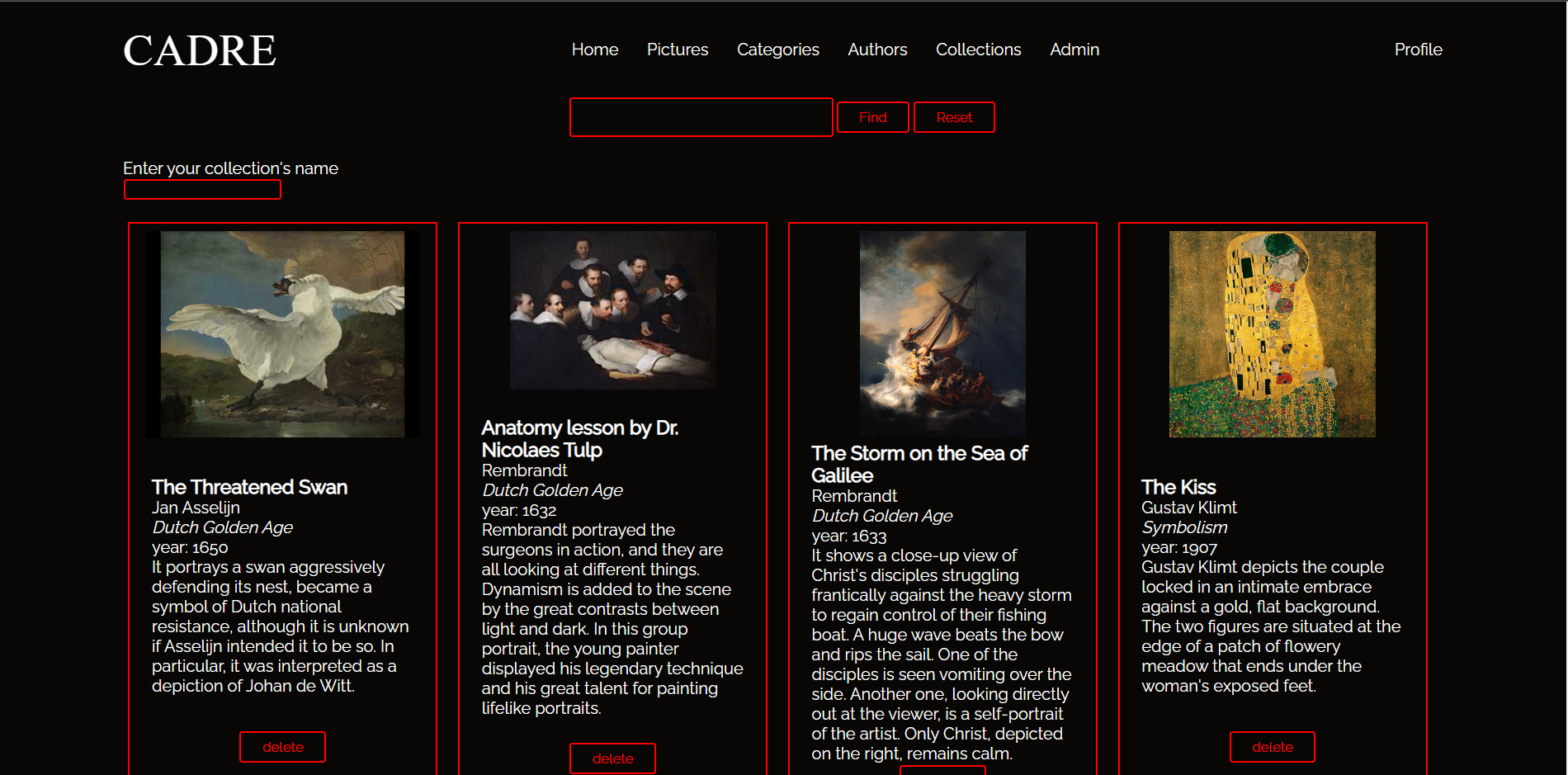


Рисунок 6.13 – Страница списка картин

Так как мы зашли за администратора, нам доступна кнопка «delete», нажав на которую можно удалить картину. Удалим картину с лебедем. Результат выполнения представлен на рисунке 6.14.

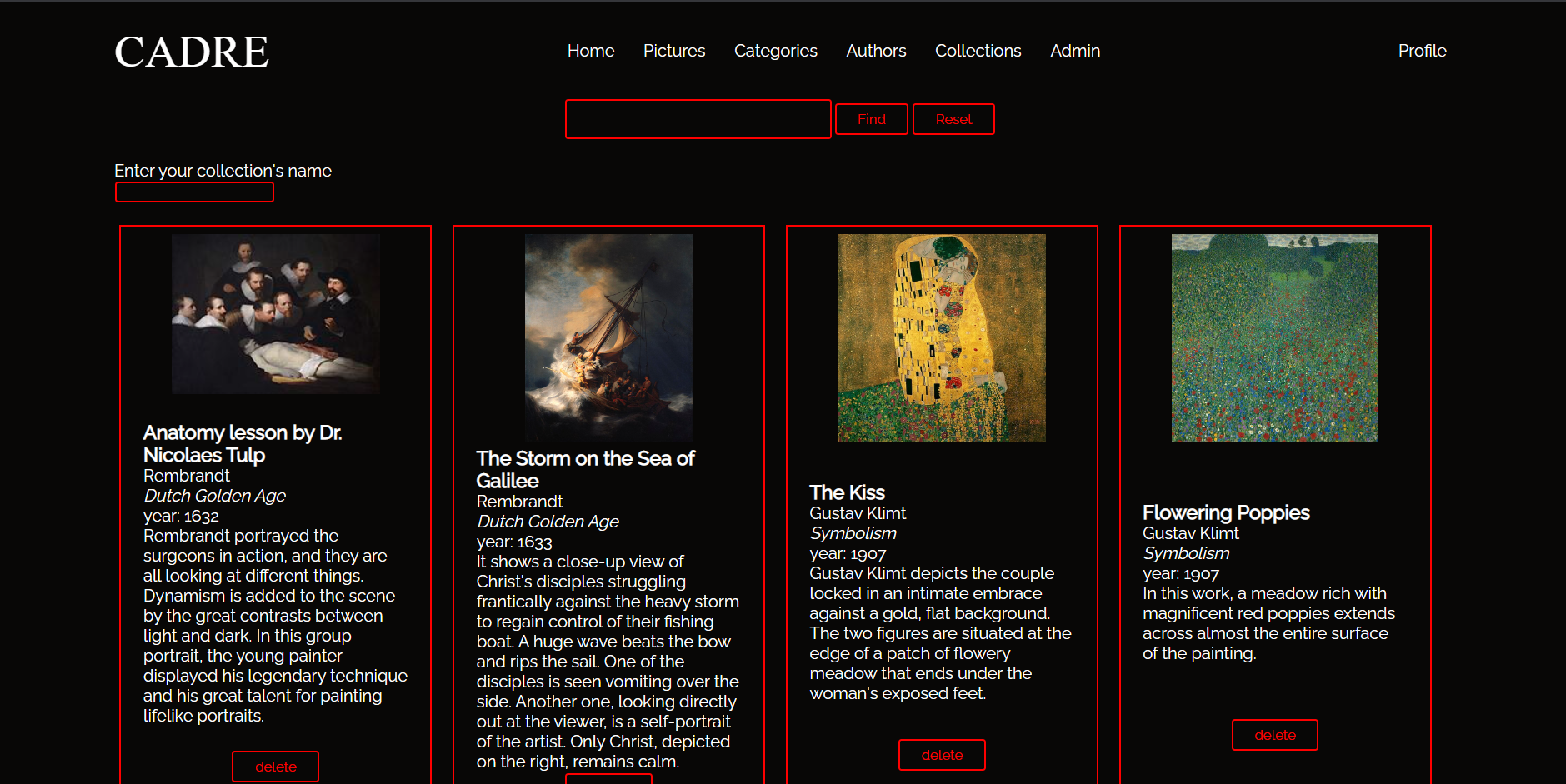


Рисунок 6.14 – Страница списка картин

Также всем пользователям доступен поиск картины по названию. Найдем элементы, в названии которых есть сочетание букв «ki», рисунок 6.15.

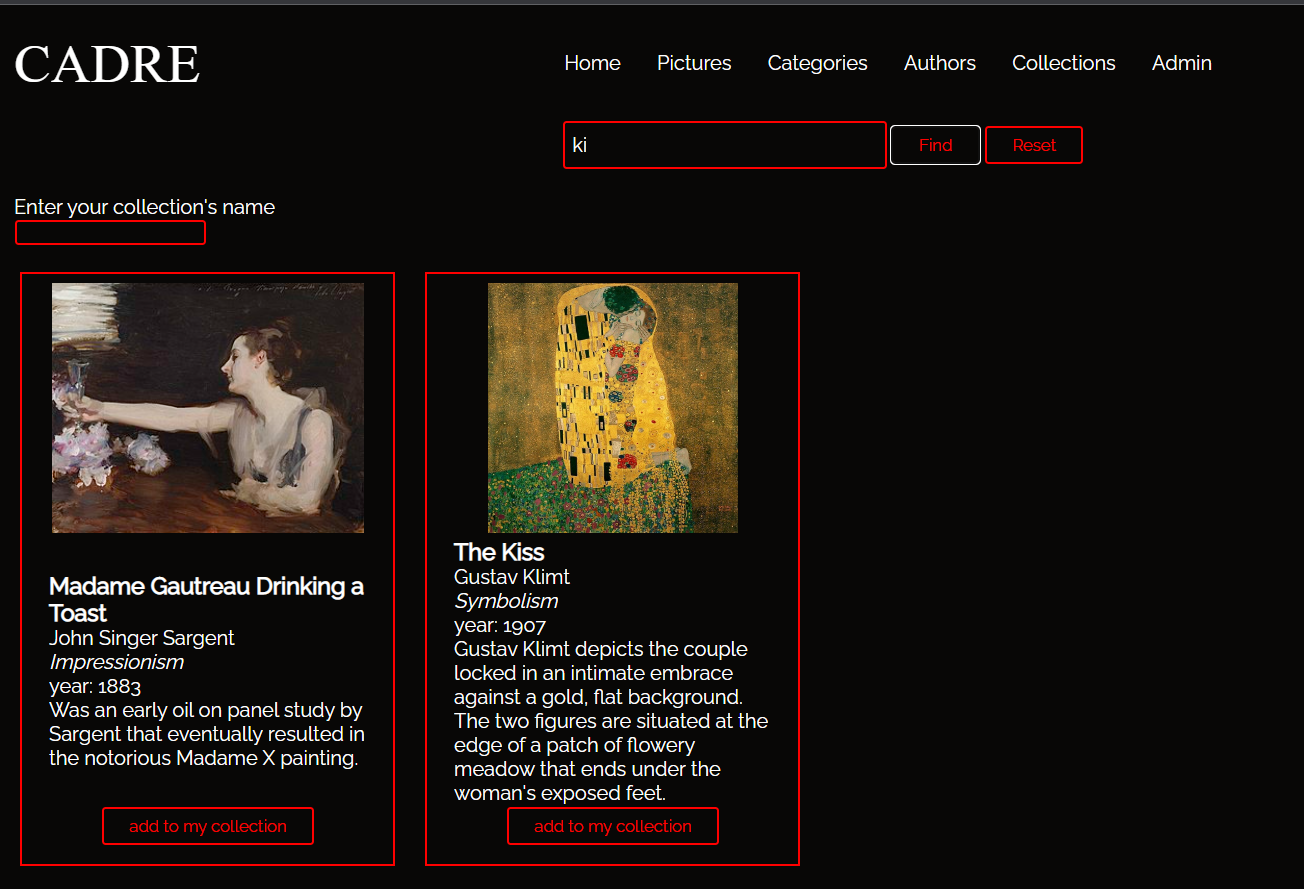


Рисунок 6.15 – Поиск картин

В приложении предусмотрена возможность создавать свои коллекции картин, они выступают в роли списков, в которые пользователь может объединять полюбившиеся картины. Для того, чтобы продемонстрировать эту возможность создадим коллекцию с названием «one» и коллекцию с названием «two», рисунок 6.16.

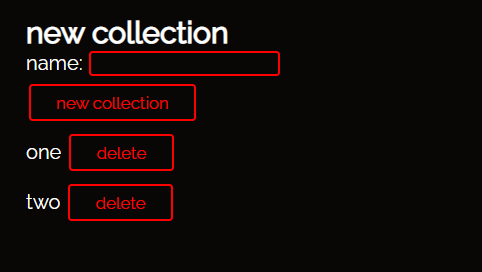


Рисунок 6.15 – Названия коллекций

Теперь добавим пару картин в созданные коллекции. Для этого необходимо ввести в поле «Enter your collection's name» название соответствующей коллекции и нажать кнопку «add to my collection» в карточке с понравившейся картиной. Предоставляется возможность добавить одну и ту же картину в несколько коллекций, что и продемонстрировано на рисунке 6.16.

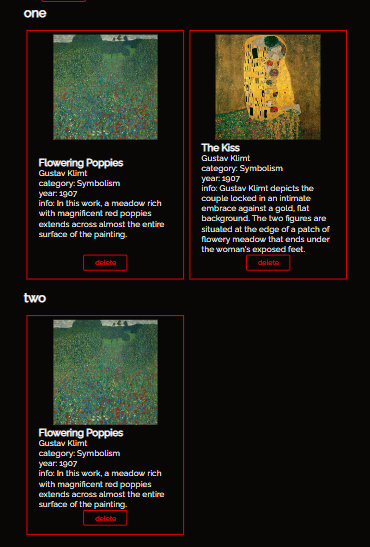


Рисунок 6.16 – Коллекции

Также пользователь может удалить картину из коллекции, нажав в соответствующей карточке кнопку «delete», рисунок 6.17.

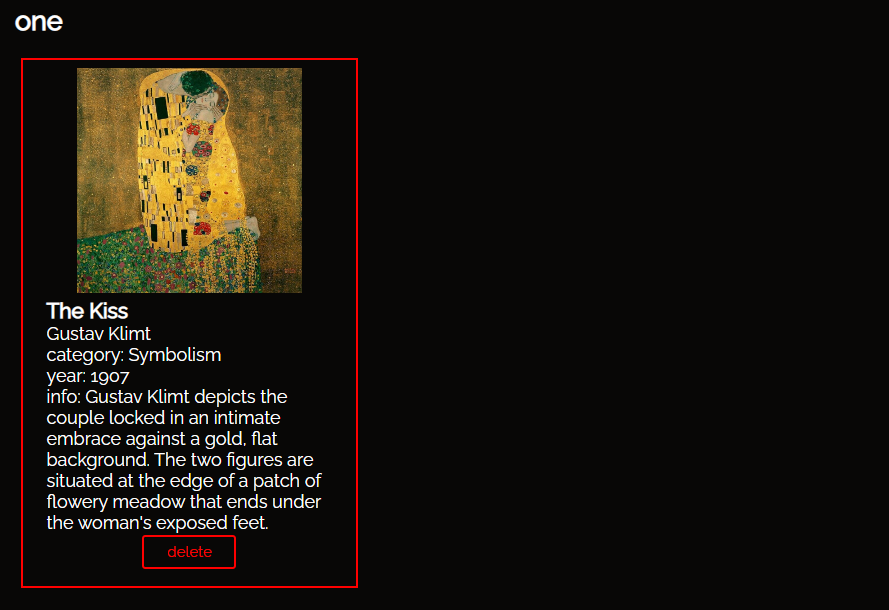


Рисунок 6.17 – Удаление картины

Перейдем на вкладку «Profile». Здесь пользователь может просмотреть свои данные и изменить, если он захочет, свое имя и пароль. Для этого необходимо ввести данные и нажать кнопку «update me», рисунок 6.18.

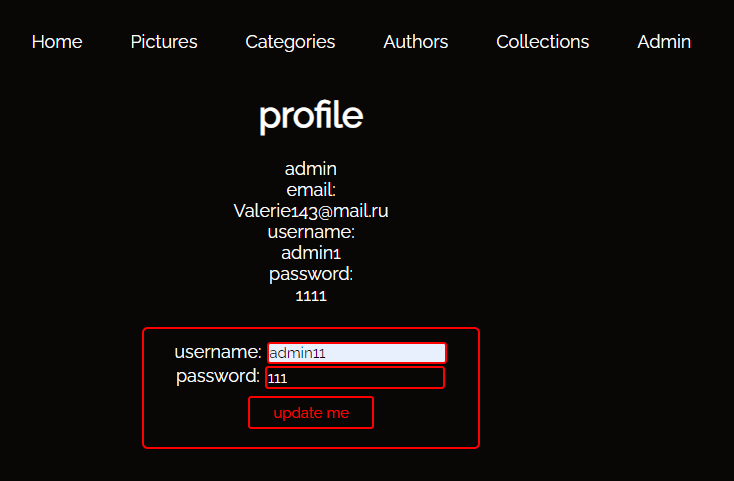


Рисунок 6.18 – Вкладка профиль

Данные пользователя изменились, рисунок 6.19.

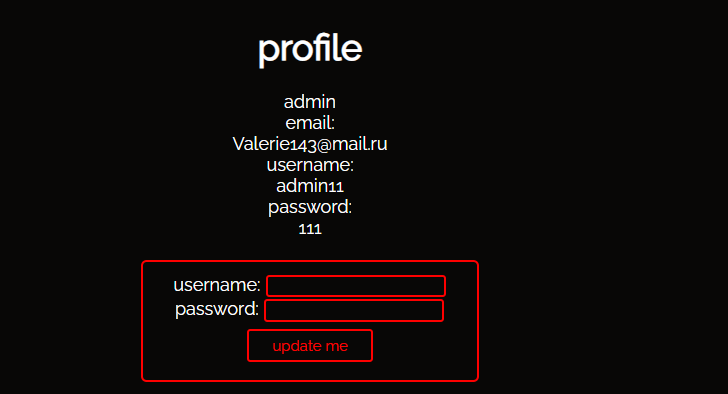


Рисунок 6.19 – Новые данные

Таким образом, все заявленные в листе заданий функции приложения были реализованы.

**Заключение**

Результатом выполнения курсового проекта стало .Web-приложение «Музей». В ходе курсовой работы была разработана база данных в Oracle 12c на данную тему. Приложение является средством взаимодействия клиента с базой данных.

Были реализованы основные требования, указанные в листе задания вместе со следующими пунктами:

регистрация и авторизация пользователей с учётом выданных им ролей;

пароли при регистрации шифруются;

мультимедийные типы данных, корректно записывающиеся в базу данных и затем воспроизводящиеся;

производится резервное копирование и восстановление данных;

поиск картин по названию;

возможность каждому пользователю сохранять понравившиеся картины в свои коллекции;

добавление авторов администратором;

добавление категорий администратором;

добавление картин администратором;

администратор имеет доступ к информации о пользователях;

все пользователи могут изменять свое имя и пароль;

администратор может удалять аккаунты;

импорт и экспорт данных таблиц в XML формат;

заполнение таблицы на 100 000 записей (таблица авторов).

Протестировав все компоненты приложения, можно прийти к заключению, что все основные требования выполнены и приложение работает исправно.

**Список литературы**

1. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com – Дата доступа: 15.10.2022.

2. Oracle-dba.ru [Электронный ресурс] / Режим доступа: https://oracle-dba.ru – Дата доступа: 27.10.2022.

3. Шифрование в PL/SQL: уроки программирования [Электронный ресурс] / Режим доступа: <https://oracle-patches.com/db/sql/3939-shifrovanie-v-pl-sql-uroki-programmirovaniya> – Дата доступа: 16.10.2022.

4. DBMS\_XMLDOM [Электронный ресурс] / Режим доступа: <https://docs.oracle.com/cd/B1930601/appdev.102/b14258/dxmldom.htm#i1076719> – Дата доступа: 20.10.2022.

5. Developing and Using Stored Procedures [Электронный ресурс] / Режим доступа: <https://docs.oracle.com/cd/B2835901/appdev.111/b28843/tdddgprocedures.htm> – Дата доступа: 26.10.2022.

6. Stackoverflow.com [Электронный ресурс] / Режим доступа: https://stackoverflow.com – Дата доступа: 08.12.2022

# Приложение А



# Приложение Б

|  |
| --- |
| create table userroles (  id number(10) GENERATED AS IDENTITY  (START WITH 1 INCREMENT BY 1 NOCYCLE ORDER),  name varchar2(50),  CONSTRAINT roles\_pk PRIMARY KEY (id)  );  create table users (  email varchar2(100),  username varchar2(100) not null,  password varchar2(255) not null,  role\_id number(10),  CONSTRAINT users\_pk PRIMARY KEY (email),  constraint role\_fk foreign key (role\_id)  references userroles(id)  );  create table authors (  id number(10) GENERATED AS IDENTITY  (START WITH 1 INCREMENT BY 1 NOCYCLE ORDER),  name varchar2(100),  info varchar2(2000),  CONSTRAINT authors\_pk PRIMARY KEY (id)  );  create table categories (  id number(10) GENERATED ALWAYS AS IDENTITY  (START WITH 1 INCREMENT BY 1 NOCYCLE ORDER),  name varchar2(100),  info varchar2(2000),  CONSTRAINT categories\_pk PRIMARY KEY (id)  );  create table pictures (  id number(10) GENERATED AS IDENTITY  (START WITH 1 INCREMENT BY 1 NOCYCLE ORDER),  name varchar2(255),  author\_id number(10),  category\_id number(10),  year number(10),  info varchar2(3000),  picture blob,  CONSTRAINT pictures\_pk PRIMARY KEY (id),  constraint author\_fk foreign key (author\_id)  references authors(id),  constraint category\_fk foreign key (category\_id)  references categories(id)  );  create table collections (  id number(10) GENERATED AS IDENTITY  (START WITH 1 INCREMENT BY 1 NOCYCLE ORDER),  name varchar2(255),  email varchar2(255),  CONSTRAINT collections\_pk PRIMARY KEY (id),  constraint users\_fk foreign key (email)  references users(email) on delete cascade  );  create table collection\_pictures (  id number(10) GENERATED AS IDENTITY  (START WITH 1 INCREMENT BY 1 NOCYCLE ORDER),  collection\_id number(10),  picture\_id number(10),  CONSTRAINT collection\_pictures\_pk PRIMARY KEY (id),  constraint collection\_fk foreign key (collection\_id)  references collections(id) on delete cascade,  constraint picture\_fk foreign key (picture\_id)  references pictures(id) on delete cascade  ); |

# Приложение В

|  |
| --- |
| create view user\_userrole\_view as  select users.email,  users.username,  userroles.name  from users left join userroles on  USERS.ROLE\_ID = USERROLES.ID;  create view full\_user\_userrole\_view as  select users.email,  users.username,  decrypt\_password(users.password) as password,  userroles.name  from users left join userroles on  USERS.ROLE\_ID = USERROLES.ID;  create view picture\_view as  select PICTURES.NAME,  PICTURES.INFO,  AUTHORS.NAME as author\_name,  CATEGORIES.NAME as categories\_name,  PICTURES.ID,  PICTURES.YEAR,  PICTURES.PICTURE from PICTURES  left join authors on  PICTURES.AUTHOR\_ID = AUTHORS.ID  left join categories on  PICTURES.CATEGORY\_ID = CATEGORIES.ID |