# **4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ**

Листинг программы представлен в Приложении А.

## **Создание базы данных PostgreSQL**

Первоначально необходимо создать новую базу данных для разработанного приложения. Ее можно создать, используя стандартные средства, предоставляемые Postgres, такими как pgAdmin (версии 4). Это стандартная утилита, предоставляющая графический интерфейс для упрощенного управления базами данных. Для создания новой базы данных необходимо включить утилиту.

Если запуск pgAdmin будет первым, то приложение выведет на экран окно «Set Master Password», в котором мы должны задать «мастер-пароль». Это нужно для дополнительного шифрования паролей. После установки мастер-пароля все существующие сохраненные пароли будут повторно зашифрованы с использованием мастер-пароля.

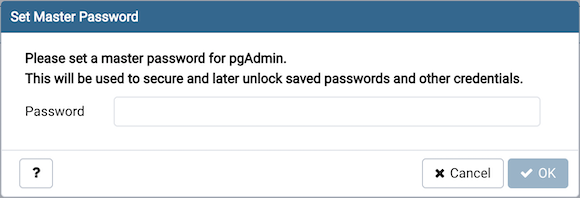


Рисунок 4.1 - Окно «Set Master Password» в pgAdmin 4

Если же вход в приложение не первый, то оно предложит ввести «мастер-пароль», выведя на экран окно «Unlock Saved Passwords».

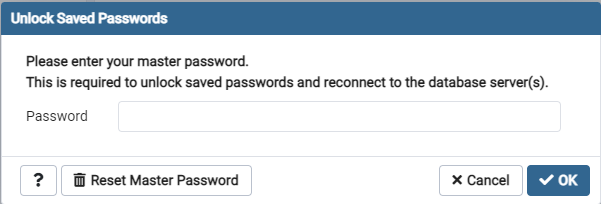


Рисунок 4.2 - Окно «Unlock Saved Passwords» в pgAdmin 4

После авторизации в приложении можно создавать новую базу данных. Для этого необходимо выполнить действия, пошагово продемонстрированные на рисунке 4.3:

Шаг 1. Правой кнопкой мыши нажать около слова «Databases» (1).

Шаг 2. В появившемся меню навести на кнопку «Create» (2).

Шаг 3. В новом меню выбрать пункт «Database…» (3).

Шаг 4. Далее, в новом окне (4), необходимо указать имя базы данных, которое будет использоваться в дальнейшем (5).

Шаг 5. Последний же шаг, подтверждение созхдания новой базы данных.

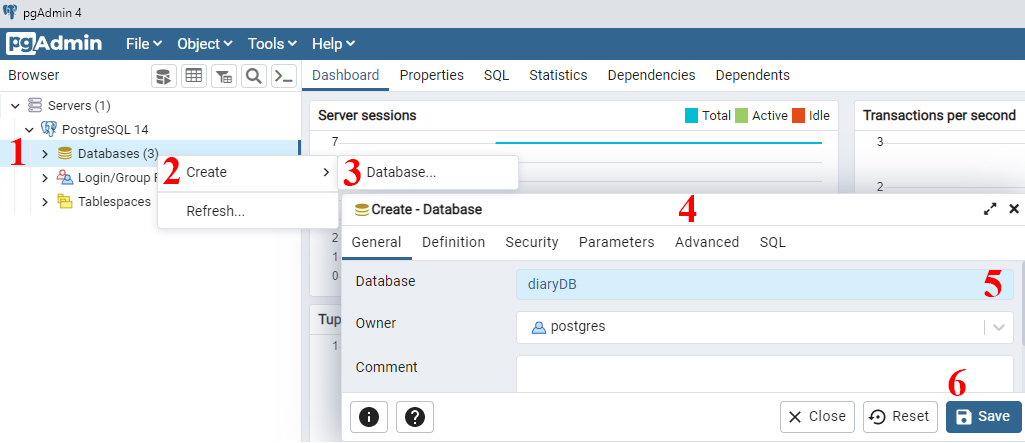


Рисунок 4.3 - Создание новой базы данных посредством pgAdmin 4

## **Создание пользователей базы данных PostgreSQL**

Следующий шаг в разработке приложения – создание пользователей с разными правами. Для этого был написан скрипт на языке SQL с использованием приложения DataGrip. Данная среда разработки необходима для написания SQL-запросов и работы с базами данных.

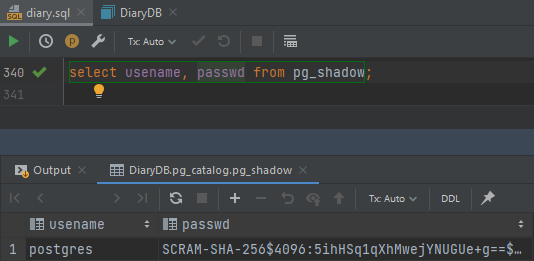


Рисунок 4.4 - Первоначальные пользователи базы данных PostgreSQL

Для просмотра существующих ролей и пользователей можно использовать запрос к таблице «pg\_shadow», которая содержит информацию об имени роли и пароле пользователя, действительности пароля пользователя, лимите подключения пользователя и роли, автоматически наследующей привилегии ролей, членом которых он является, подробную информацию о пользователях и управлении привилегиями. Для демонстрации создаваемых автоматически пользователей используем запрос, который отобразит имя пользователя и пароль для него:

select usename, passwd from pg\_shadow;

Для разделения ролей будет создана новая роль auth\_user. Ее суть – предоставления права для чтения данных из таблиц, без возможности их модернизации. Пароль задается сугубо для примера «pgsecret», что не является надежным, однако хорошим для демонстрации написания запроса.

CREATE USER auth\_user WITH ENCRYPTED PASSWORD 'pgsecret';

GRANT USAGE ON SCHEMA public to auth\_user;

ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT SELECT ON TABLES TO auth\_user;

GRANT CONNECT ON DATABASE "DiaryDB" to auth\_user;

ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT ALL ON TABLES TO auth\_user;

GRANT USAGE ON SCHEMA public to auth\_user;

GRANT SELECT ON ALL SEQUENCES IN SCHEMA public TO auth\_user;

GRANT SELECT ON ALL TABLES IN SCHEMA public TO auth\_user;

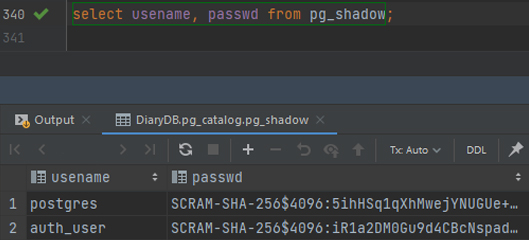


Рисунок 4.5 - Все пользователи базы данных PostgreSQL

После выполнения данного запроса можно проверить его выполнение уже описанным ранее запросом (рисунок 4.5).

## **Создание резервных копий базы данных PostgreSQL**

При написании данного дипломного проекта рассматривается автономное создание резервной копии базы данных. Для этого необходимо написать скрипт с расширением «.cmd». Шаги по написанию данного скрипта:

Шаг 1. Установим кодовую страницу, соответствующую Windows кодировке.

Шаг 2. Создадим путь до корневой папки PostgreSQL.

Шаг 3. Укажем адрес сервера.

Шаг 4. Укажем порт сервера.

Шаг 5. Установим имя пользователя, имеющего права на данную базу данных.

Шаг 6. Укажем пароль пользователя, указанного на шагу 5.

Шаг 7. Создадим переменную, указывающую на текущее время.

Шаг 8. Укажем имя базы данных, для которой создается бэкап.

Шаг 9. Выполнение бэкапа с указанием конечного места копирования.

Шаг 10. Также для уменьшения объема памяти, занимаемого приложением, удаляются старые копии, старше указанного срока (на примере указано 29 дней).

CHCP 1251

SET PGBIN=C:\Program Files\PostgreSQL\14\bin

SET PGHOST=localhost

SET PGPORT=3306

SET PGUSER= DBadmin

SET PGPASSWORD=diary

SET DATETIME=%DATE:~6,4%-%DATE:~3,2%-%DATE:~0,2%

SET PGDATABASE=diaryDB

CALL "%PGBIN%\pg\_dump.exe" --format=custom --verbose --file=C:\14\back\%PGDATABASE%\_%DATETIME%.dump

forfiles /p C:\14\ /m \*.dump /s /d -29 /c "cmd /c del @path /q"

Следующим будет добавление данного скрипта в планировщик заданий. Сделать это не является проблемным. Необходимо зайти в «планировщик заданий», нажать правой кнопкой мыши в районе существующих задач и выбрать пункт «создание простой задачи». Далее необходимо указать название задачи, частоту запуска и путь к созданному ранее скрипту. После чего активировать данный скрипт. В результате мы получим автономно выполняющиеся задачи по созданию резервной копии, созданной ранее базы данных, необходимой для повышенной безопасности, в случае потери данных, или «неудачи» по внесению обновлению базы данных, что может привести к утрате актуальных данных.

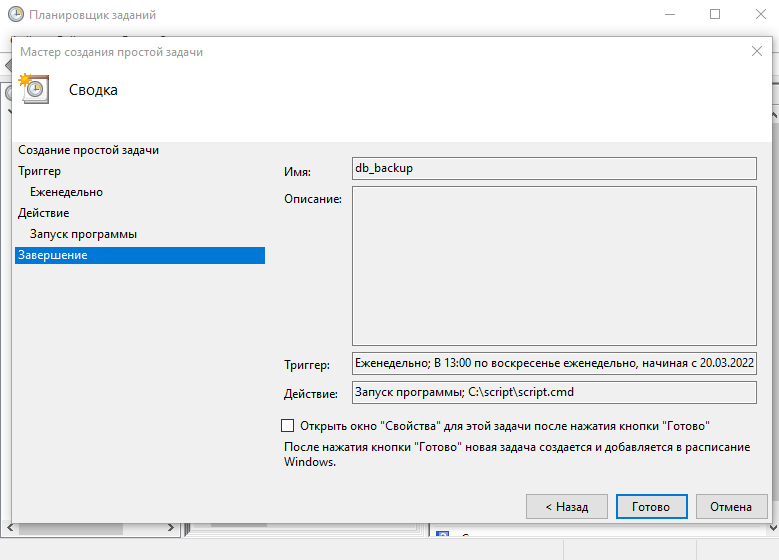


Рисунок 4.6 - Создание задачи по резервному копированию базы данных

## **Создание таблиц и связей базы данных PostgreSQL с использованием**

Созданная заранее ER-модель преобразуется в реляционную модель согласно шагам, описываемым в методическом пособии к лабораторным работам [13]. В результате выполнения всех шагов, можно получить файл, с расширением «.dll». Выполнив преобразования (не все типы данных, существующие в приложении Oracle SQL Developer Data Modeler, поддерживаются другими приложениями, например, Data Grip) получаем следующий скрипт, по созданию базы данных. Ниже представлены несколько таблиц и связь между ними, чтобы не загромождать описываемый раздел. Полный скрипт будет описан в приложении.

CREATE TABLE ids (

student\_id INTEGER NOT NULL,

teacher\_id INTEGER NOT NULL

);

ALTER TABLE ids ADD CONSTRAINT ids\_pk PRIMARY KEY ( student\_id, id );

CREATE TABLE teacher (

full\_name\_t VARCHAR,

gender VARCHAR,

id INTEGER NOT NULL,

rank VARCHAR,

phone\_number VARCHAR,

subject\_id INTEGER NOT NULL,

birth\_date\_t DATE

);

ALTER TABLE teacher ADD CONSTRAINT teacher\_pk PRIMARY KEY ( id );

CREATE TABLE student (

full\_name\_st VARCHAR,

gender VARCHAR,

id INTEGER NOT NULL,

studying\_group\_id INTEGER NOT NULL,

birth\_date\_st DATE,

stud\_parent\_id INTEGER NOT NULL

);

ALTER TABLE student ADD CONSTRAINT student\_pk PRIMARY KEY ( id );

ALTER TABLE ids

ADD CONSTRAINT ids\_student\_fk FOREIGN KEY ( student\_id )

REFERENCES student ( id );

ALTER TABLE ids

ADD CONSTRAINT ids\_teacher\_fk FOREIGN KEY ( teacher\_id )

REFERENCES teacher ( id );

С помощью запроса, описанного ниже, можно увидеть, что таблицы действительно создались:

select \* from information\_schema.tables where table\_schema='public';

Результат выполнения запроса предоставлен на рисунке 4.7.

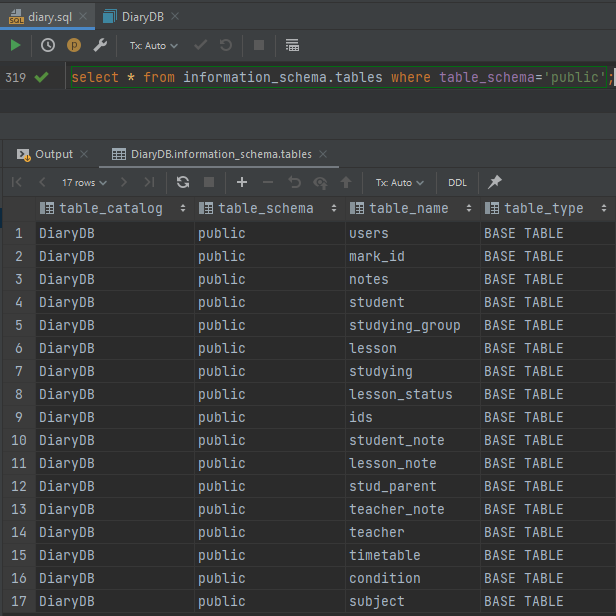


Рисунок 4.7 - Результат проверки создания таблиц в базе данных

## **Регистрация приложения** **в API Яндекс ID**

Регистрация нового приложения является простой операцией, не требующей особых усилий. Необходимо перейти на сайт данного API, расположенного по адресу <https://oauth.yandex.ru/>. После чего сайт потребует пользователя авторизоваться. Стоит отметить, что аккаунт, с которого будет произведена авторизация в дальнейшем будет считаться «администратором». Поэтому заранее рекомендуется защитить авторизуемый аккаунт сильным паролем. После авторизации необходимо нажать на кнопку «зарегистрировать новое приложения». Следующий экран предложит «добавить» приложение, указав его название, описание, логотип, ссылку на сайт. При выборе веб-сервисов указывается его callback URI (адрес, на который пользователь возвращается после того, как он разрешил или отказал приложению в доступе (соответствует redirect\_uri протокола OAuth)). Также можно добавить тот факт, что «Яндекс» предоставляет собственный адрес <https://oauth.yandex.ru/verification_code>, что значительно упрощает разработку.

«Яндекс» бесплатно предоставляет доступ ко множеству своих сервисов (например, «Яндекс.Дисплей», что позволяет размещать рекламные баннеры в приложении, «Яндекс ID», используемая в данном дипломном проекте для доступа к логину, имени и фамилии, полу пользователя, а также доступа к адресу электронной почты), что позволяет рассчитывать на масштабирование в дальнейшем с использованием данного сервиса.

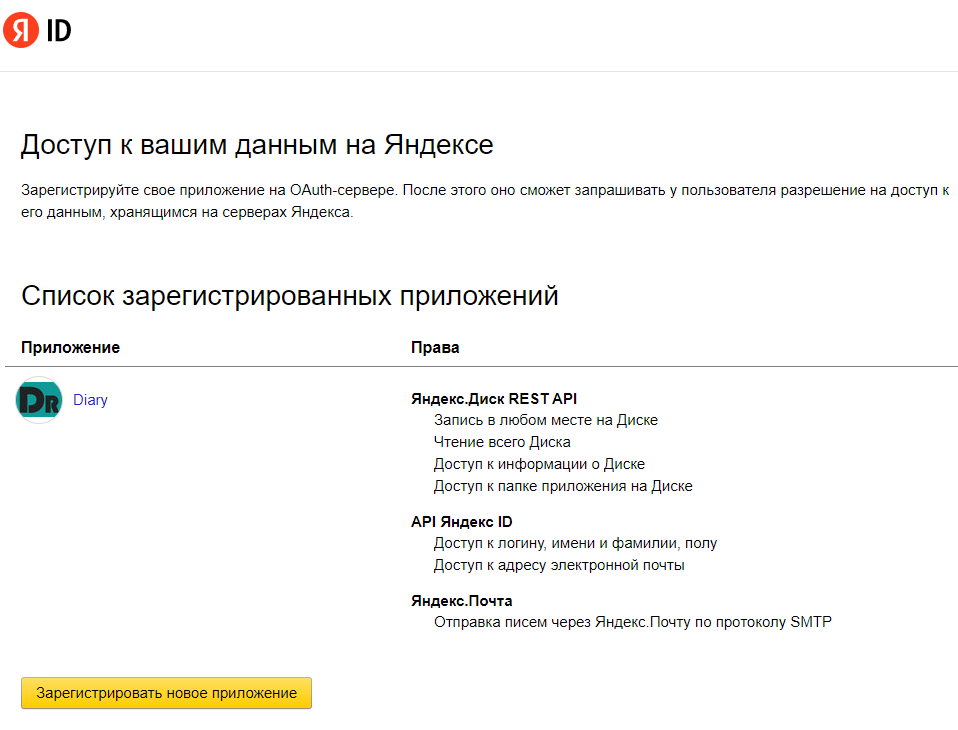


Рисунок 4.8 – Зарегистрированные приложения в API Яндекс ID

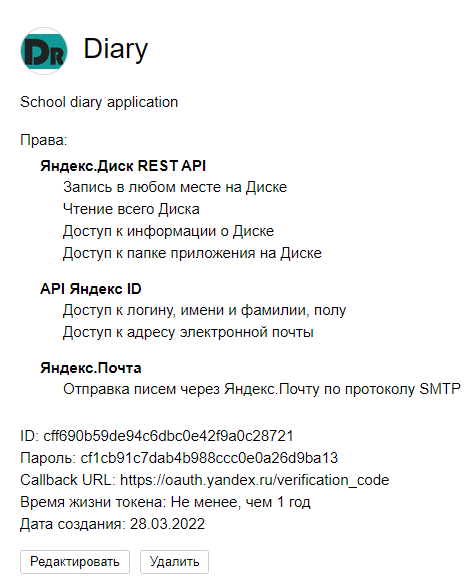


Рисунок 4.9 – Приложение в API Яндекс ID

После успешного создания приложения пользователь увидит нечто похожее на то, что изображено на рисунке 4.9. Самый интересующие здесь момент – это ID (идентификационный номер приложения).

Позже, данный идентификационный номер будет использоваться при авторизации пользователей, при составлении ссылки, по которой производится перенаправление:

https://oauth.yandex.ru/authorize?response\_type=token&client\_id=cff690b59de94c6dbc0e42f9a0c28721

Как можно заметить, данный ID указывается после знака «=».

## **Класс**