Спецификация проекта

Для товаров, заказов, корзины:

Post - create

Put - update

Get - read

Delete - delete

Четыре основные таблицы:

1 - tovars(товары),

2 - orders(заказы),

3 - cart(корзина),

4 - customers(пользователи)

Таблица товаров содержит следующие атрибуты:

1 id идентификатор

2 name название товара

3 price цена товара

4 amount количество товара

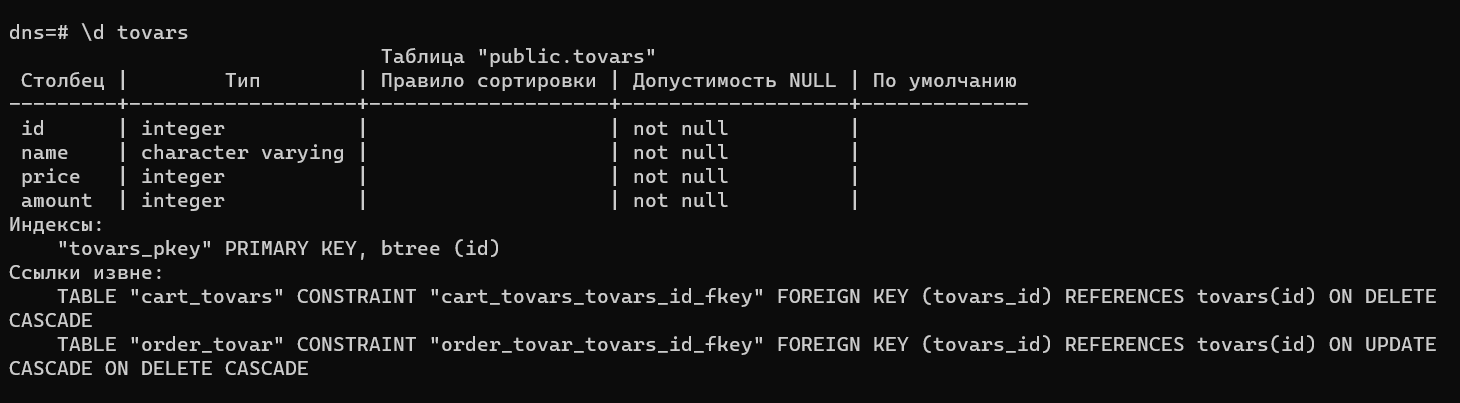


Таблица заказов содержит:

1 id идентификатор

2 spisok список товаров в заказе

3 sum\_price итоговая сумма заказа, которая состоит из суммы товаров в заказе

4 orders\_date дата оформления заказа

5 customerid покупатель

6 status статус заказа

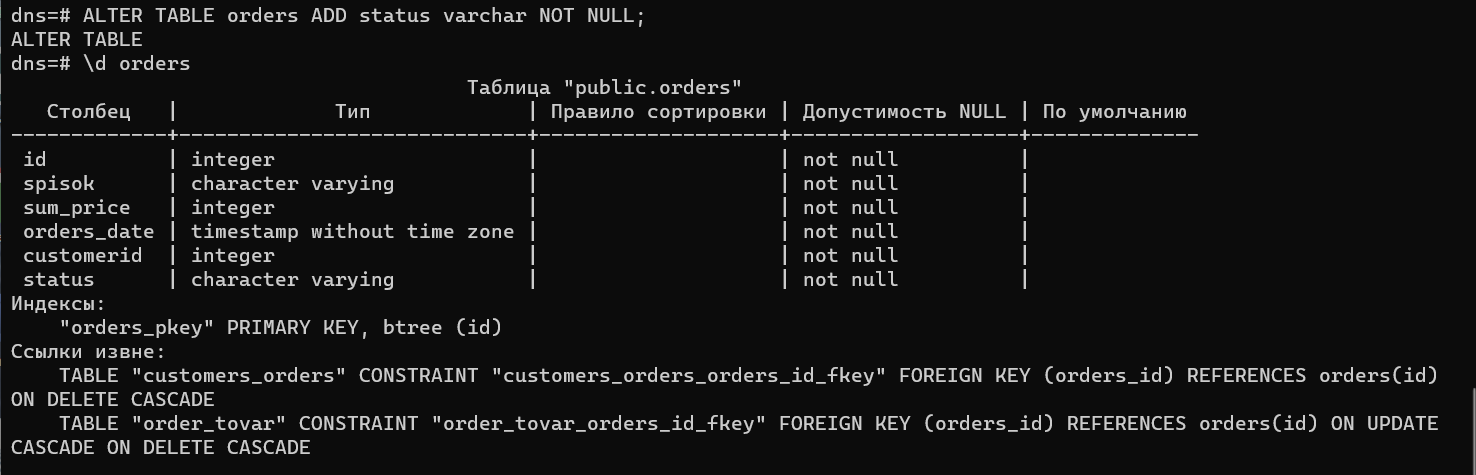


Таблица корзины содержит 👍

1 id идентификатор

2 list список заказов

3 sumprice итоговая цена заказов

4 customerid идентификатор покупателя

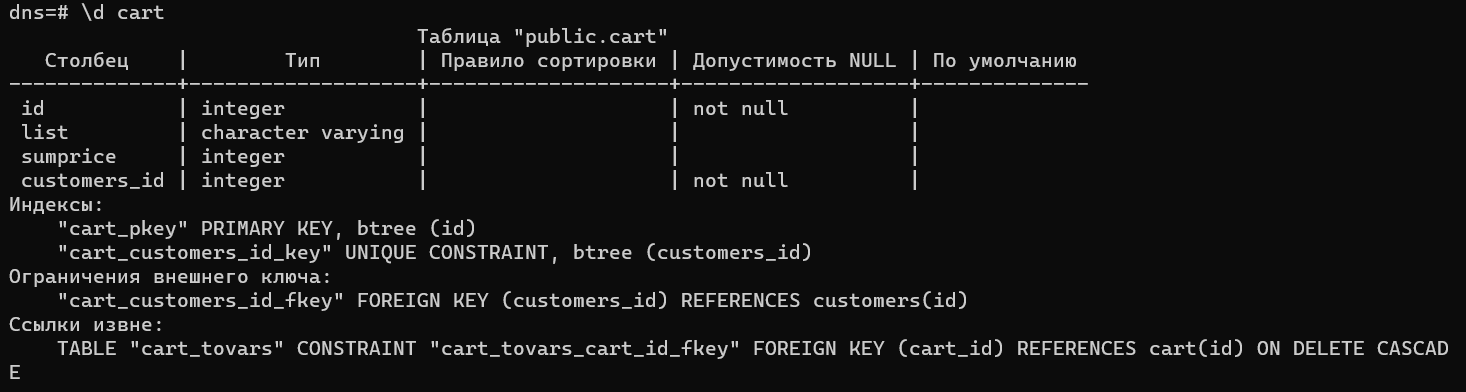


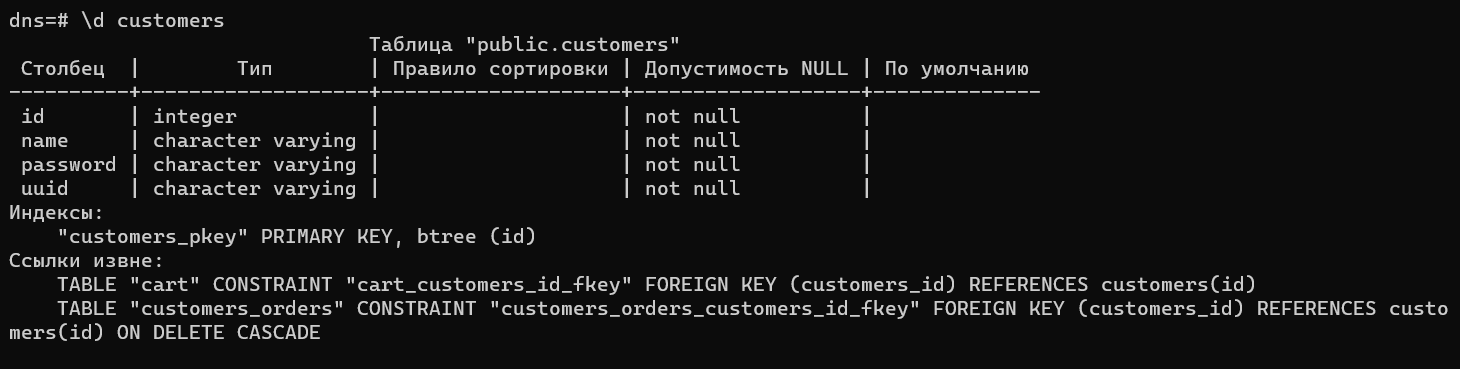
Таблица пользователей содержит 🙂

1 id идентификатор в бд

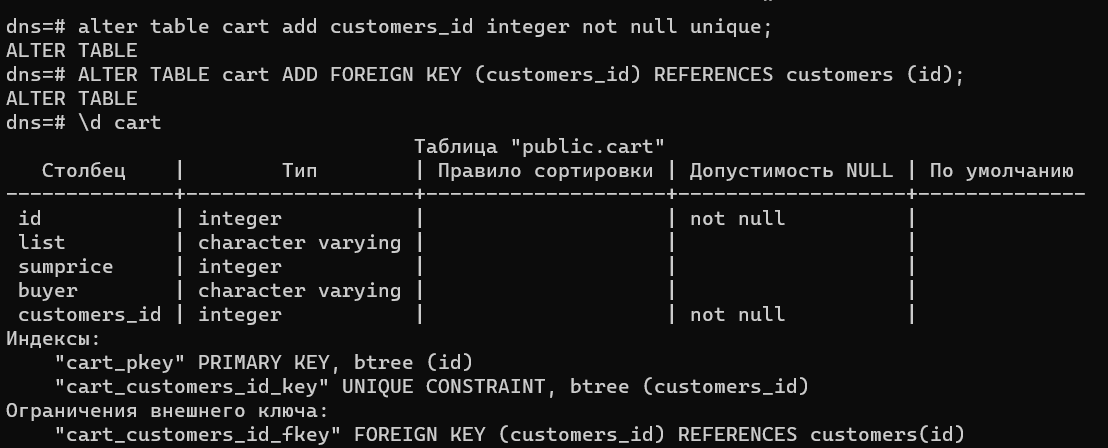
2 name имя пользователя

3 password пароль пользователя

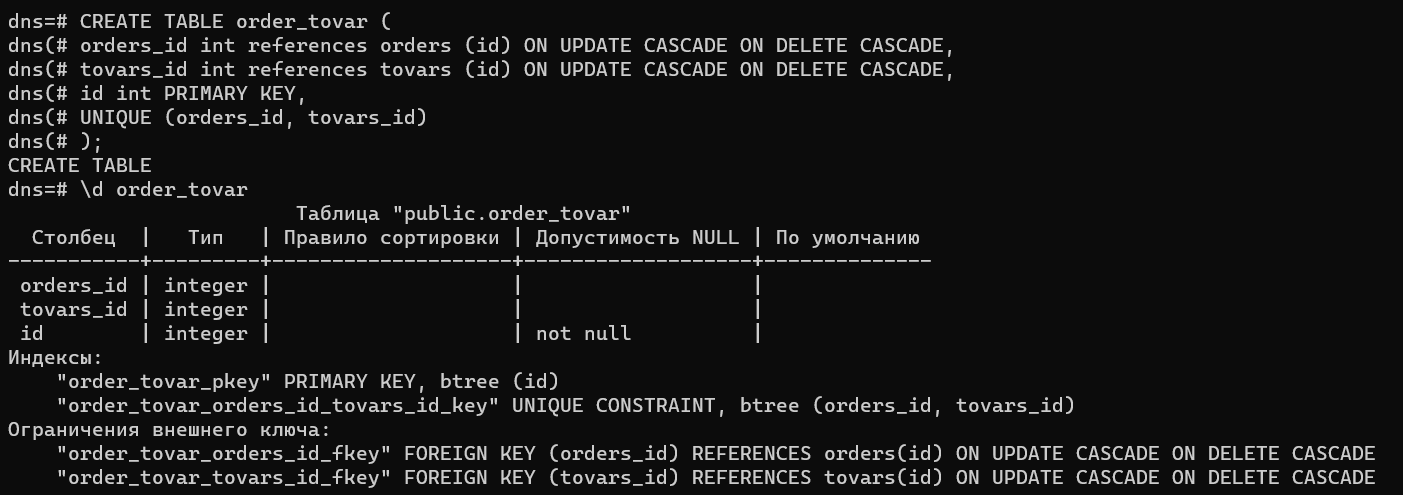
4 uuid идентификатор на сервере



Создание внешнего ключа таблицы козина(cart) на таблицу покупатель(customers):



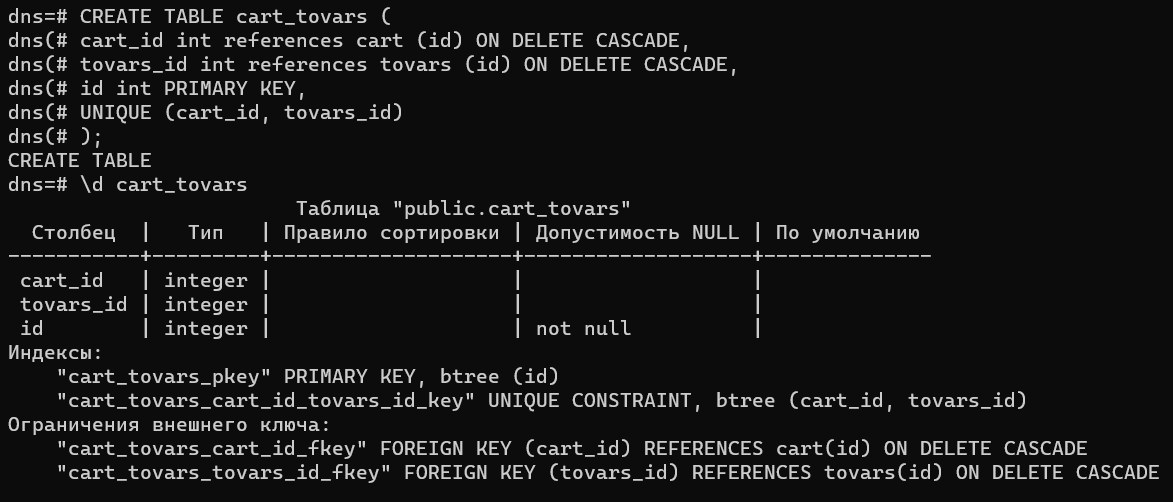
Создание связи многие ко многим между таблицами заказов и товаров через третью таблицу order\_tovar 👏



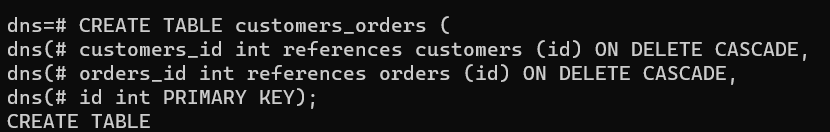
Ограничения UNIQUE я удалил:

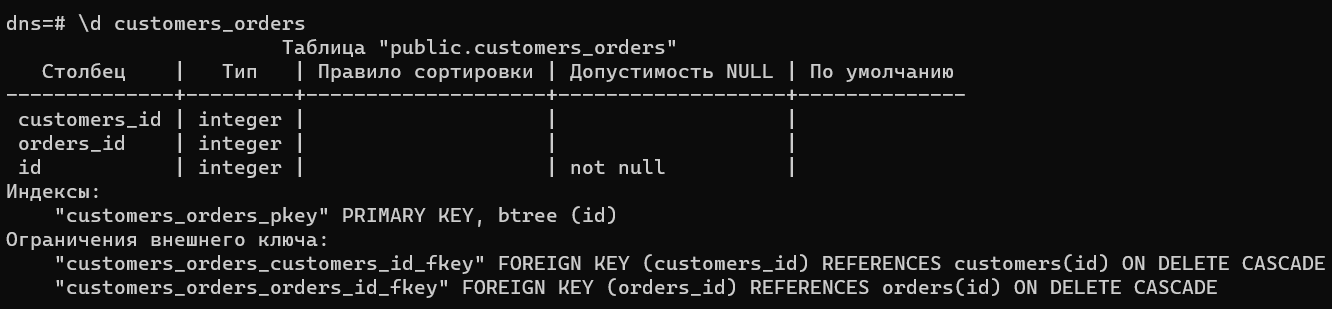


Создание связи многие ко многим между таблицами товары и корзина:



Создание связи многие ко многим между таблицами заказы и покупатели:





Настройка последовательности serial:

Получается у нас есть следующие связи

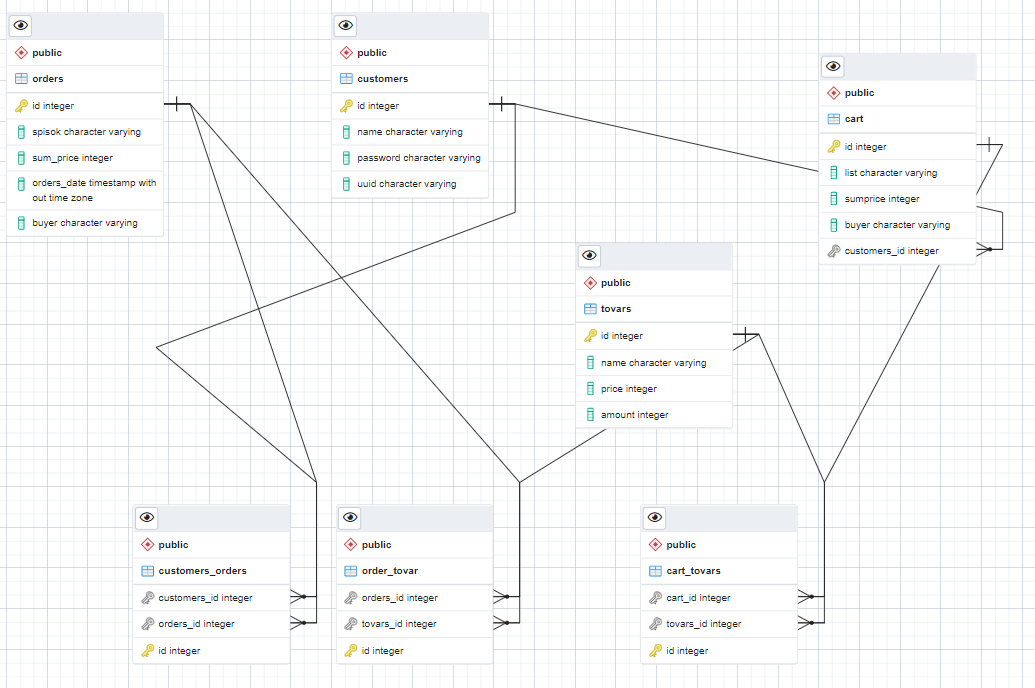
корзина —> покупатель (один ко многим)

заказы —> покупатели (многие ко многим)

заказы —> товары (многие ко многим)

корзины —> товары (многие ко многим)

ERD диаграмма базы данных:



**Веб сервер**

Спецификация

Примеры API-методов для магазина техники с таблицами products, orders, customers, cart:

* Получение списка товаров - GET /catalog  
  Возвращает список всех товаров в базе данных.

{

"id": 1,

"name": "Smartphone Honor9X",

"price": 15000,

"amount": 27,

"createdAt": "2023-03-11T17:47:20.920Z",

"updatedAt": "2023-03-11T17:47:20.920Z"

},

* Получение информации о конкретном товаре - GET /tovars/{id}/{name}  
  Возвращает информацию о товаре с указанным идентификатором.

{

"id": 2,

"name": "processor-intel-core13-12100F",

"price": 9000,

"amount": 43,

"createdAt": "2023-03-11T17:55:41.864Z",

"updatedAt": "2023-03-11T17:55:41.864Z"

}

* Добавление нового товара - POST /tovars  
  Добавляет новый товар в базу данных.

{

"name": "processor-intel-core13-12100F",

"price": 9000,

"amount": 43

}

* Изменение информации о товаре - PUT /tovars/{id}  
  Изменяет информацию о товаре с указанным идентификатором.

{

"name": "processor-intel-core13-12100F",

"price": 9000,

"amount": 42

}

* Удаление товара - DELETE /tovars/{id}  
  Удаляет товар с указанным идентификатором из базы данных. Выводит сообщение:

{

"message": "Tovar delete!"

}

* Добавление товара в корзину - POST /cart  
  Добавляет товар в корзину пользователя.

{

"customerUuid": "c1cba4e5-d9db-4ca8-af60-8d672dd56871",

"sumprice": 15000,

"tovarid": 1

}

* Получение списка товаров в корзине - GET /cart  
  Возвращает список товаров, добавленных в корзину пользователя.

{

"id": 1,

"list": "Smartphone Honor9X",

"sumprice": 15000,

"customerId": 1,

"createdAt": "2023-03-11T17:47:42.445Z",

"updatedAt": "2023-03-11T17:47:42.445Z"

},

* Удаление товара из корзины - DELETE /cart/{id}  
  Удаляет товар с указанным идентификатором из корзины пользователя.

{

"message": "Tovar is deleted!"

}

* Создание заказа - POST /orders  
  Создает новый заказ на основе товаров в корзине пользователя.

{

"cartId": 2,

"status": "In assembly",

"orders\_date": "2023-03-08"

}

* Получение информации о конкретном заказе - GET /orders/{id}  
  Возвращает информацию о заказе с указанным идентификатором.

{

"spisok": "Smartphone Honor9X",

"sum\_price": 15000,

"orders\_date": "2023-03-08T00:00:00.000Z",

"cartId": 1,

"status": "completed",

"createdAt": "2023-03-11T17:52:02.650Z",

"updatedAt": "2023-03-12T09:43:22.763Z",

"customer": {

"name": "DJon Dou",

"password": "84673328rufr",

"uuid": "cf6c8a03-a325-42e0-bd30-42ae2a4e9f48",

"createdAt": "2023-03-11T17:47:11.780Z",

"updatedAt": "2023-03-11T17:47:11.780Z"

}

}

* Получение списка заказов - GET /orders  
  Возвращает список всех заказов в базе данных.

{

"spisok": "Smartphone Honor9X",

"sum\_price": 15000,

"orders\_date": "2023-03-08T00:00:00.000Z",

"cartId": 1,

"status": "In assembly",

"createdAt": "2023-03-11T17:53:42.398Z",

"updatedAt": "2023-03-11T17:53:42.398Z"

},

* Обновление статуса заказа - PUT /orders/{id}/status  
  Изменяет статус заказа с указанным идентификатором. Тело запроса:

{

"status": "completed"

}

* Регистрация нового пользователя - POST /users  
  Регистрирует нового пользователя в базе данных. Тело запроса:

{

"name": "DungensOfDragons",

"password": "5665t5g45bg12gqw"

}

Ответ сервера:

{

"uuid": "c1cba4e5-d9db-4ca8-af60-8d672dd56871",

"name": "DungensOfDragons",

"password": "5665t5g45bg12gqw",

"updatedAt": "2023-03-11T17:57:08.730Z",

"createdAt": "2023-03-11T17:57:08.730Z"

}

* ~~Авторизация пользователя - POST /users/login  
  Авторизует пользователя на основе его логина и пароля.~~
* ~~Получение информации о текущем пользователе - GET /users/me  
  Возвращает информацию о текущем авторизованном пользователе.~~

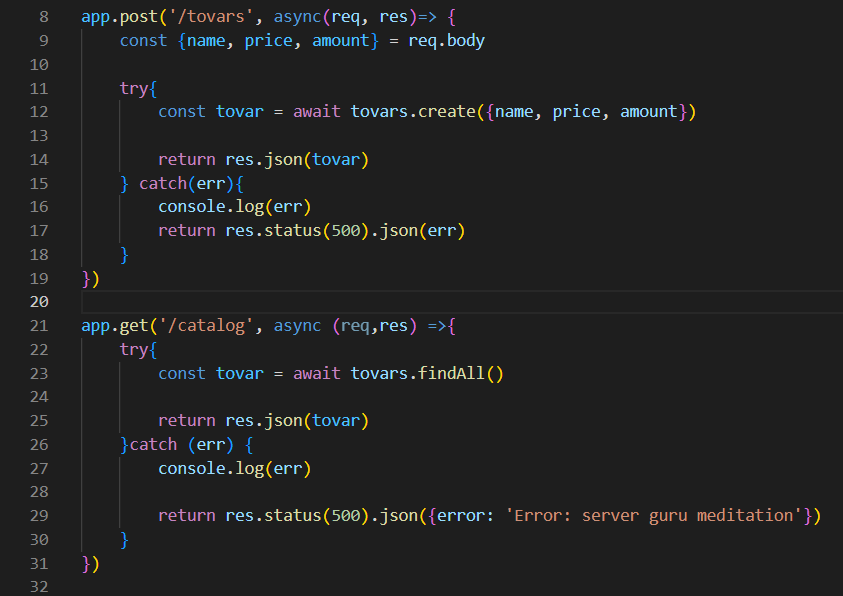
Каждый метод с телом возвращает и принимает значения в формате JSON. Для доступа к API требуется наличие токена авторизации uuid

Что уже реализовано (артем на жс)

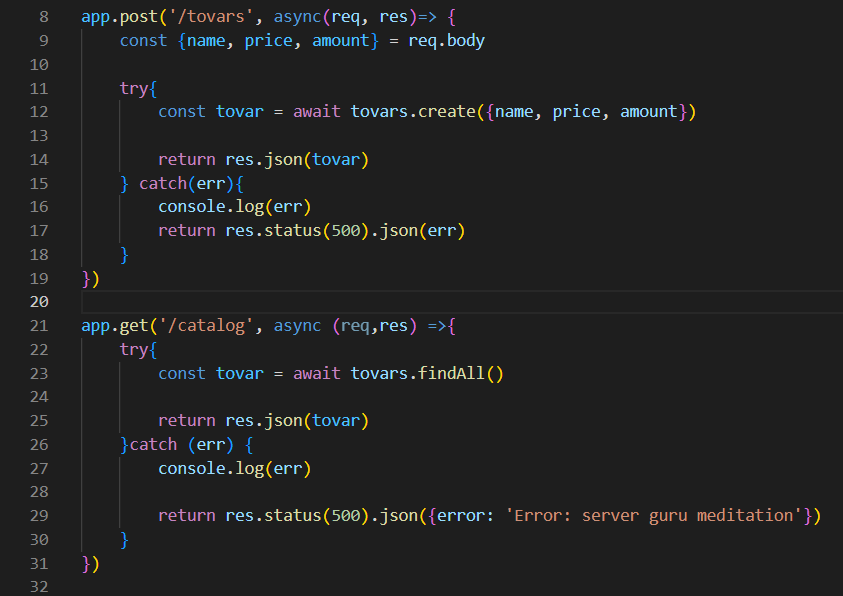
* ~~Получение списка товаров - GET /catalog  
  Возвращает список всех товаров в базе данных.~~
* ~~Получение информации о конкретном товаре - GET /tovars/{id}/{name}  
  Возвращает информацию о товаре с указанным идентификатором.~~
* ~~Добавление нового товара - POST /tovars  
  Добавляет новый товар в базу данных.~~
* ~~Изменение информации о товаре - PUT /tovars/{id}  
  Изменяет информацию о товаре с указанным идентификатором.~~
* ~~Удаление товара - DELETE /tovars/{id}  
  Удаляет товар с указанным идентификатором из базы данных.~~
* ~~Добавление товара в корзину - POST /cart  
  Добавляет товар в корзину пользователя(в json передается идентификатор товара).~~
* ~~Получение списка товаров в корзине - GET /cart/(нужен ли здесь индентификатор пользователя?)  
  Возвращает список товаров, добавленных в корзину пользователя.~~
* ~~Удаление товара из корзины - DELETE /cart/{id}  
  Удаляет товар с указанным идентификатором из корзины пользователя. Пока делает просто удаление корзины по идентификатору~~
* ~~Создание заказа - POST /orders  
  Создает новый заказ на основе товаров в корзине пользователя.~~
* ~~Получение информации о конкретном заказе - GET /orders/{id}  
  Возвращает информацию о заказе с указанным идентификатором.~~
* ~~Получение списка заказов - GET /orders  
  Возвращает список всех заказов в базе данных.~~
* ~~Обновление статуса заказа - PUT /orders/{id}/status  
  Изменяет статус заказа с указанным идентификатором.~~
* ~~Регистрация нового пользователя - POST /customers  
  Регистрирует нового пользователя в базе данных.~~
* Авторизация пользователя - POST /customers/name  
  Авторизует пользователя на основе его логина (name) и пароля (password).
* Получение информации о текущем пользователе - GET /users/me  
  Возвращает информацию о текущем авторизованном пользователе.

**Код запросов реализованный на js.**

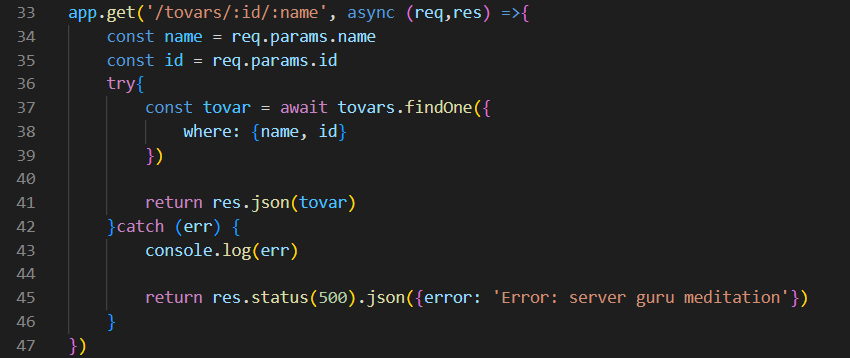
Создание товаров:



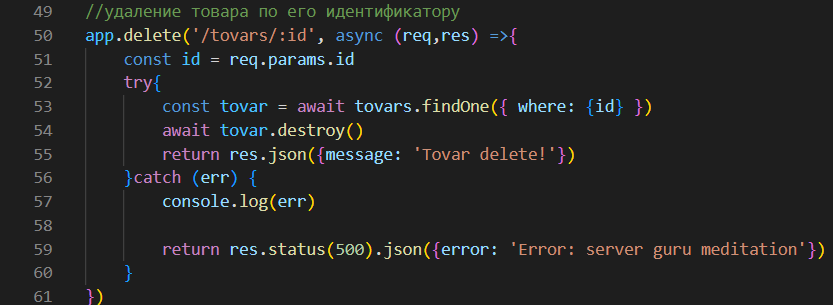
Список товаров:



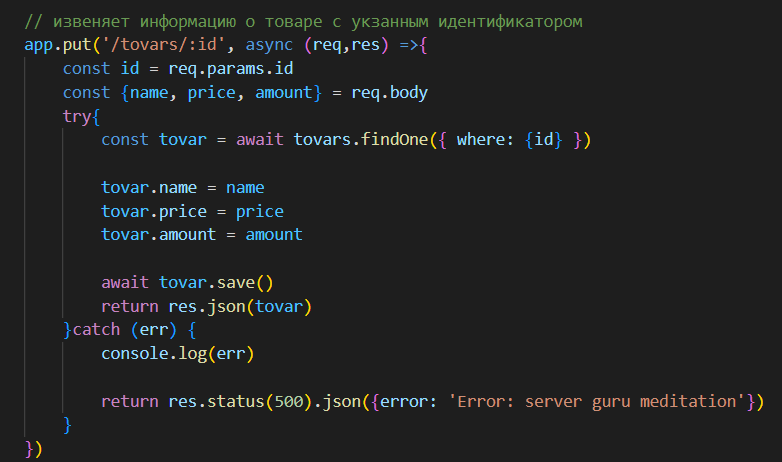
просмотр конкретного товара:



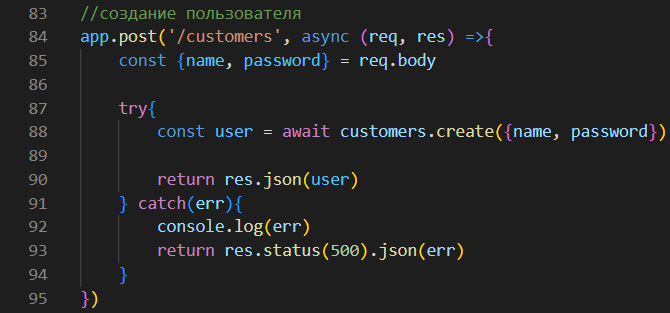
удаление товара по идентификаттору

****

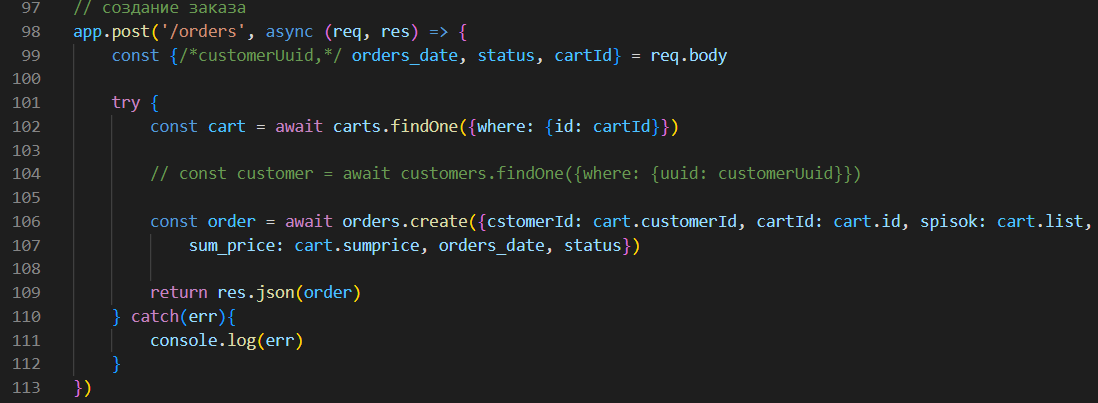
изменение информации о товаре

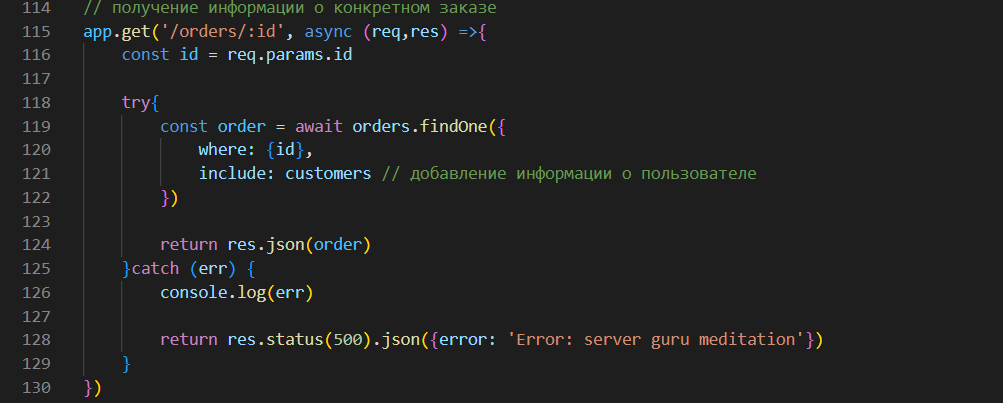


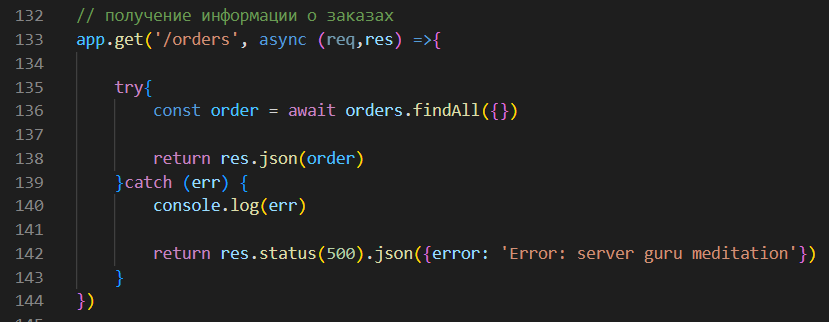
регистрация пользователя

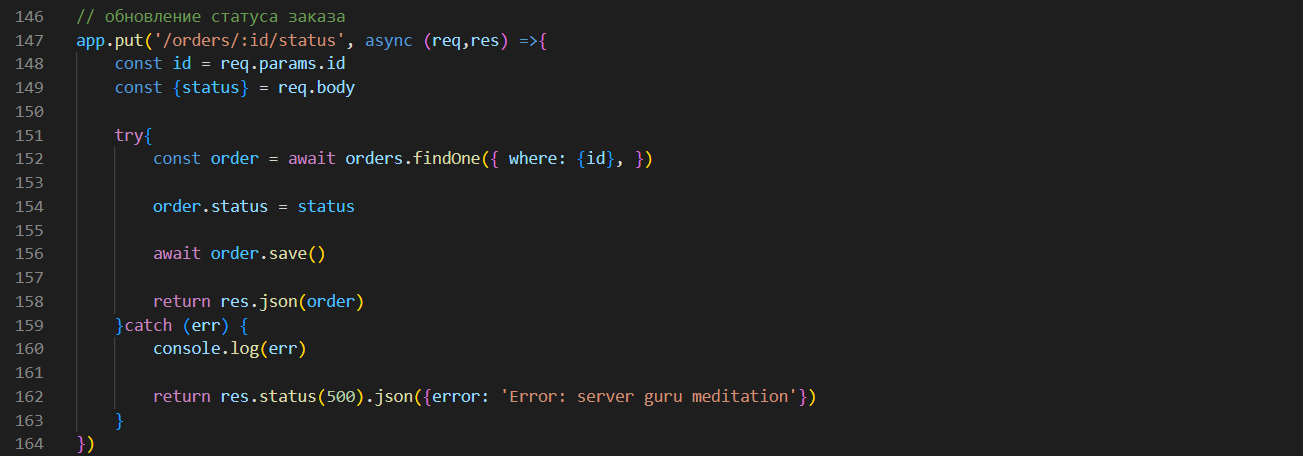


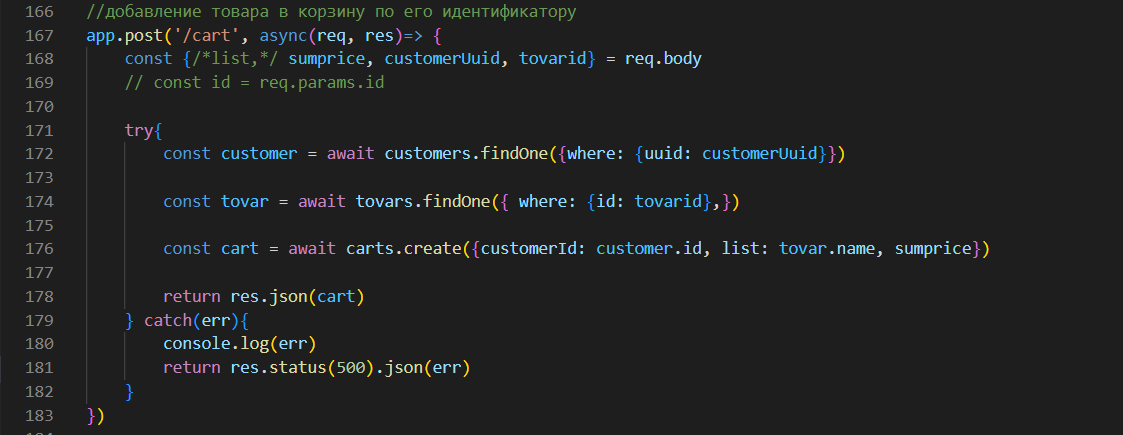
создание заказа

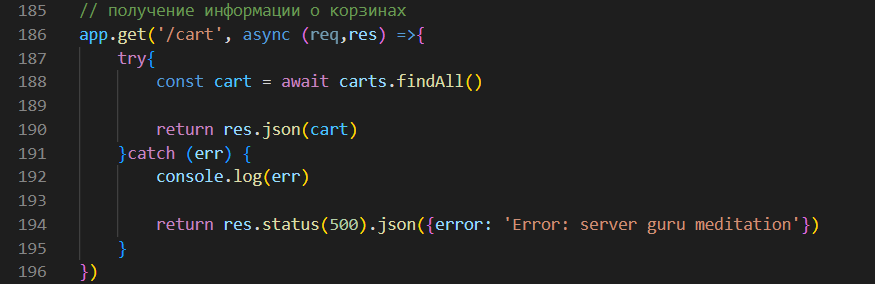


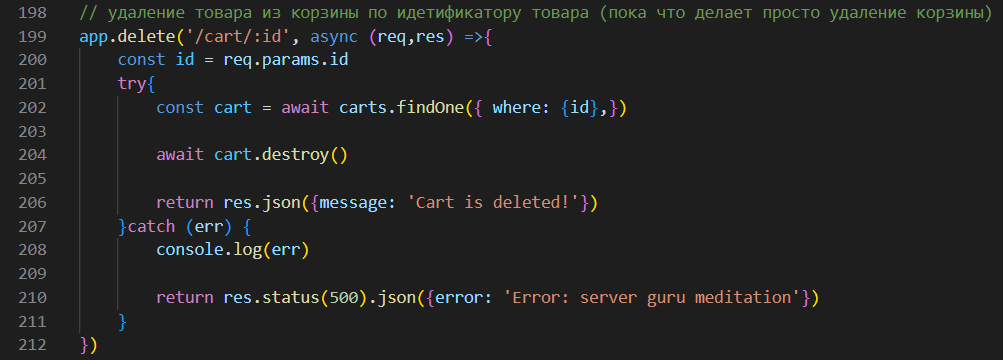










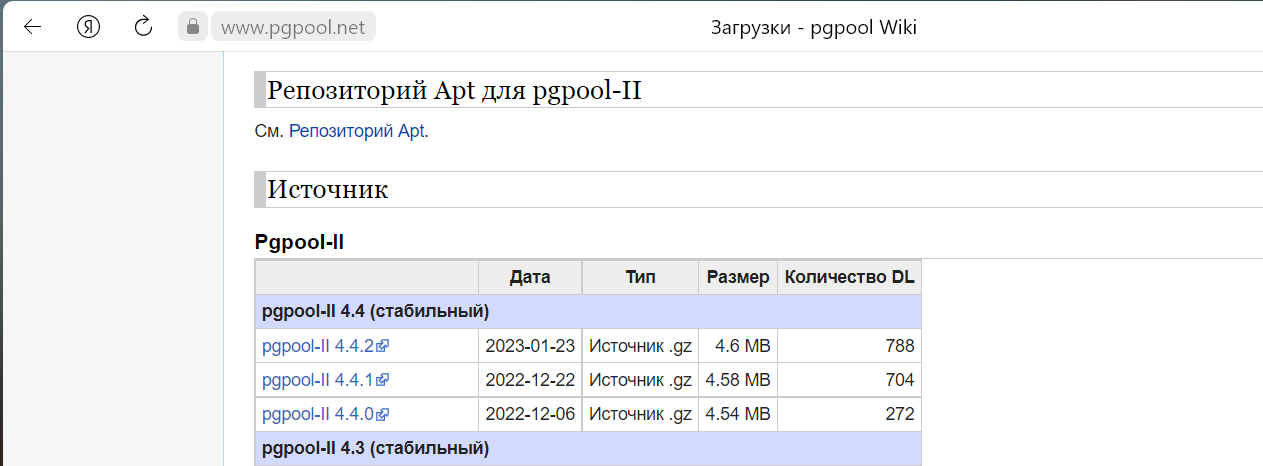


**Мультимастер-репликация БД**

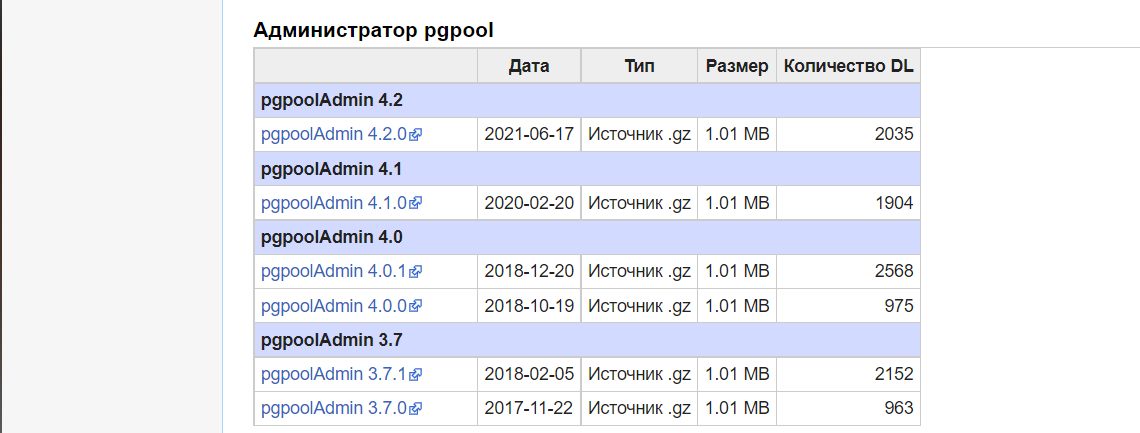
**с использованием pgpool.**

полезные ресурсы: <https://www.youtube.com/watch?v=qpxKlH7DBjU&list=PLBrWqg4Ny6vVwwrxjgEtJgdreMVbWkBz0&index=1>

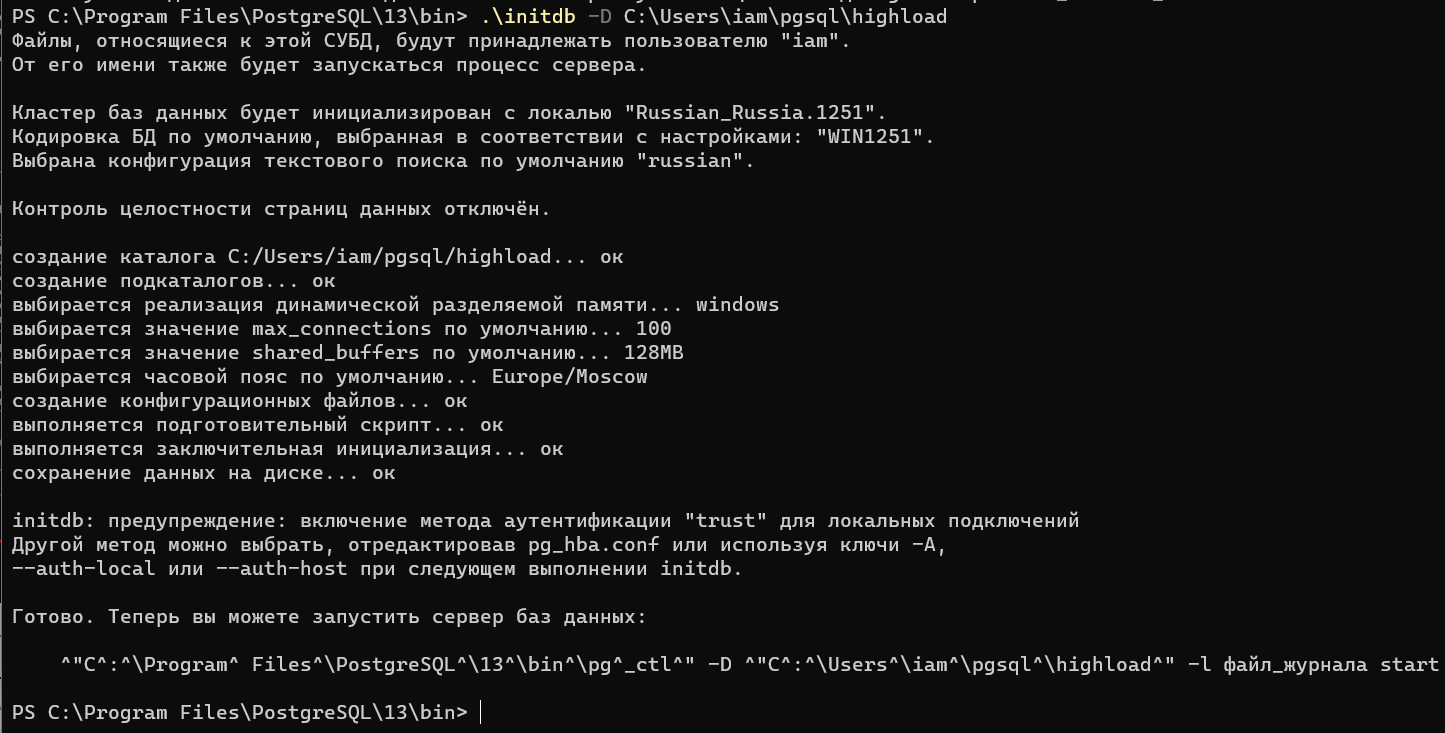
скачивание pgpool: <https://www.pgpool.net/mediawiki/index.php/Downloads>



скачивание pgpoolAdmin: <https://www.pgpool.net/mediawiki/index.php/Downloads>



Создание кластера базы данных:

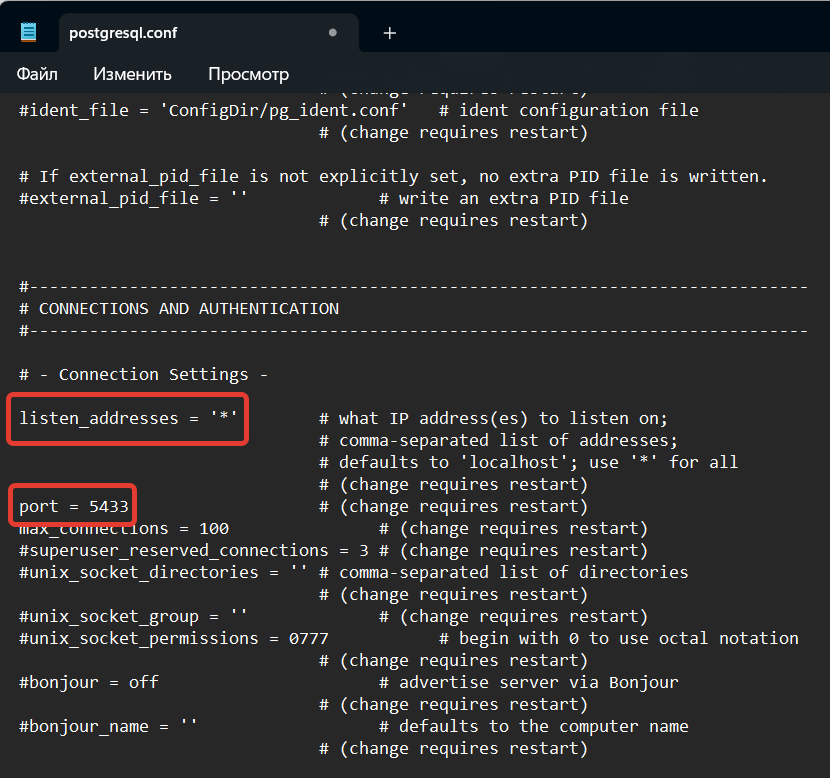


Изменяем файл конфигурации кластера postgresql:

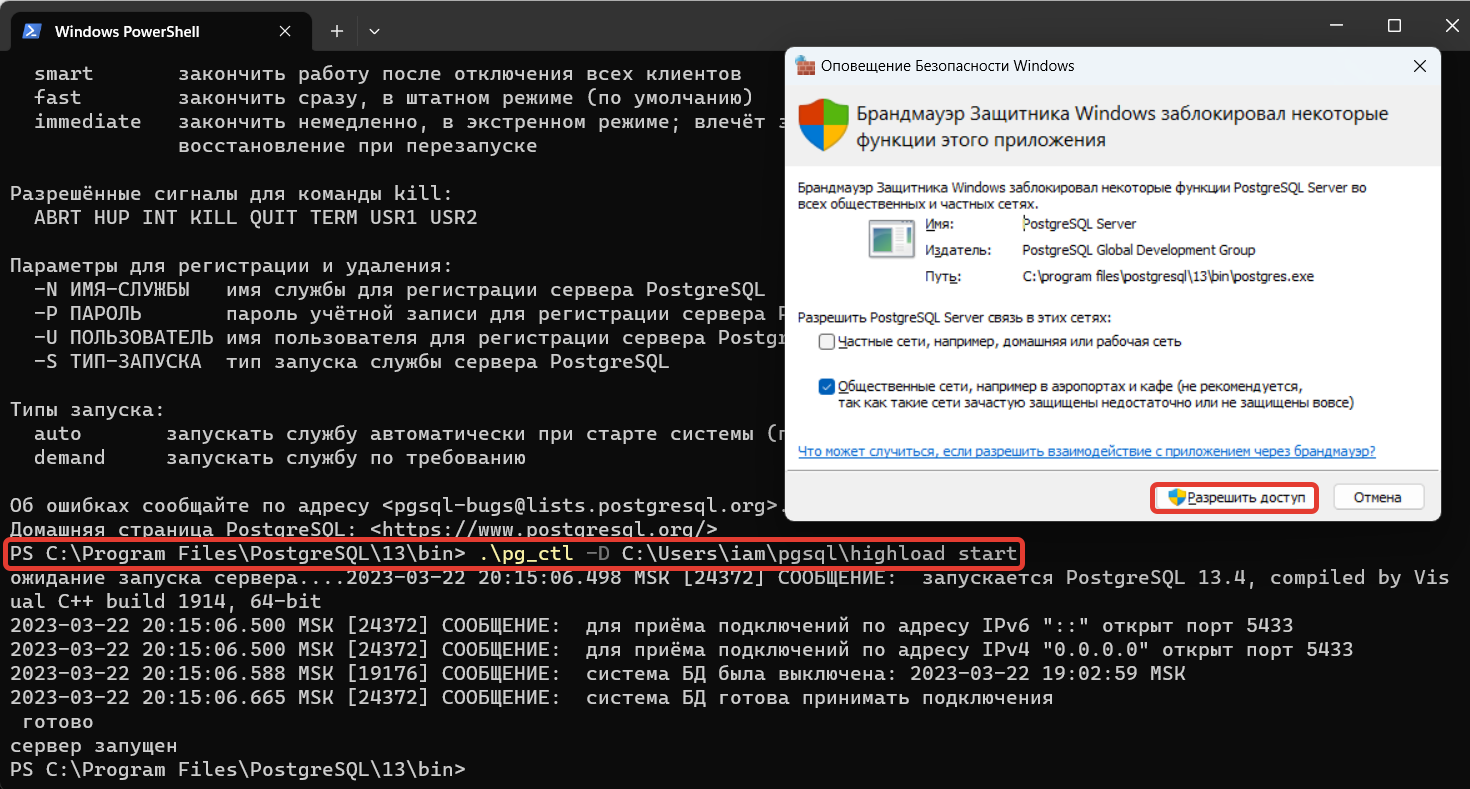
Где рефакторим строки:

* listen\_addresses изменяя значения с localhost на \* тем самым разрешая ему прослушивать все хосты
* port изменяя значение на 5433 тем самым избегая запуска стандартного сервера и кластера на одном порту.

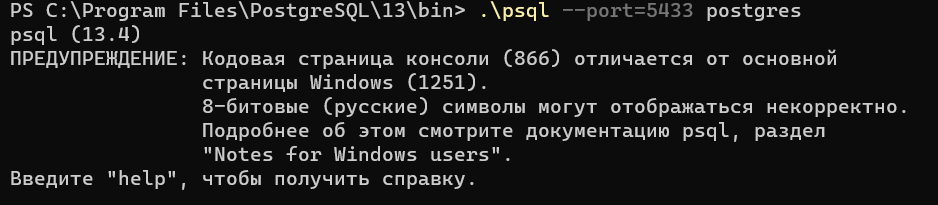
На линуксе это проще сделать через nano.

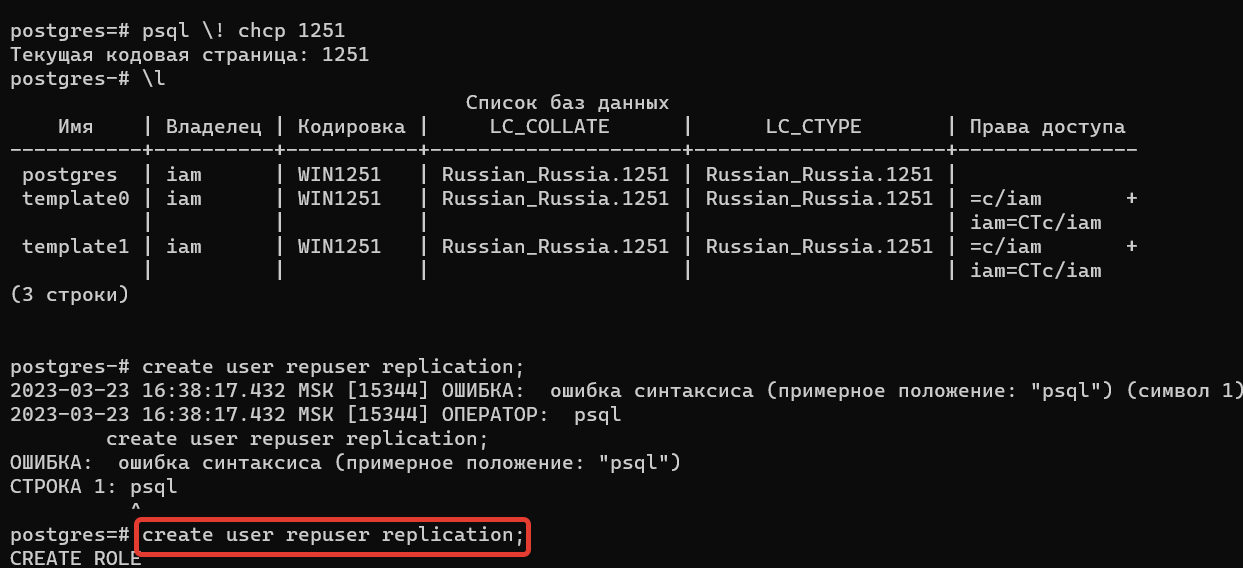


Запускаем класстер и разрешаем доступ:

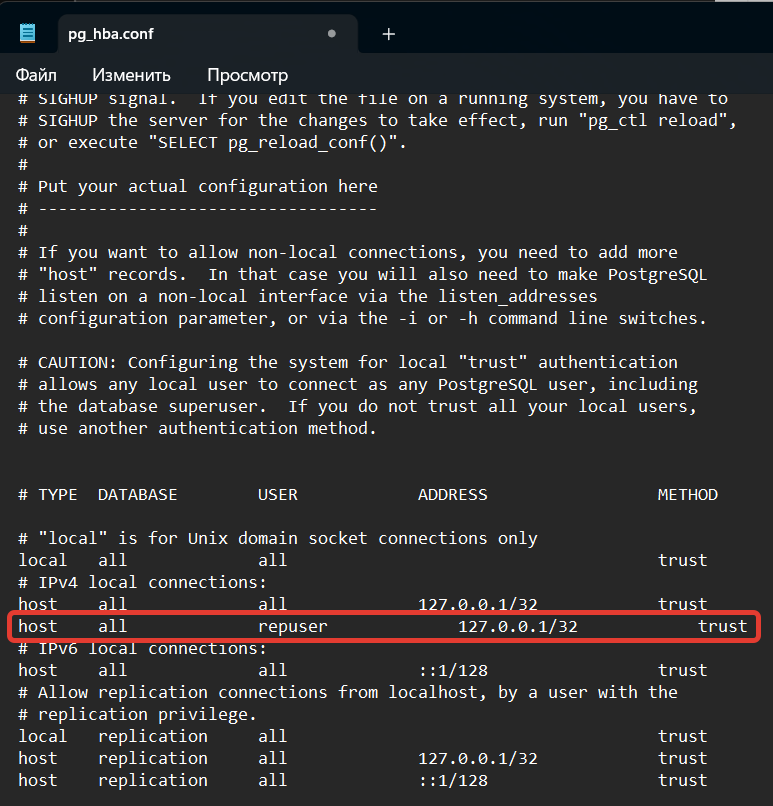


подключение к базе данных на класстере с портом 5433:



Создание роли для репликации, это не обязательно но является хорошей практикой 🙂

изменение конфигурации файла, добавление роли



перезапуск сервера:

