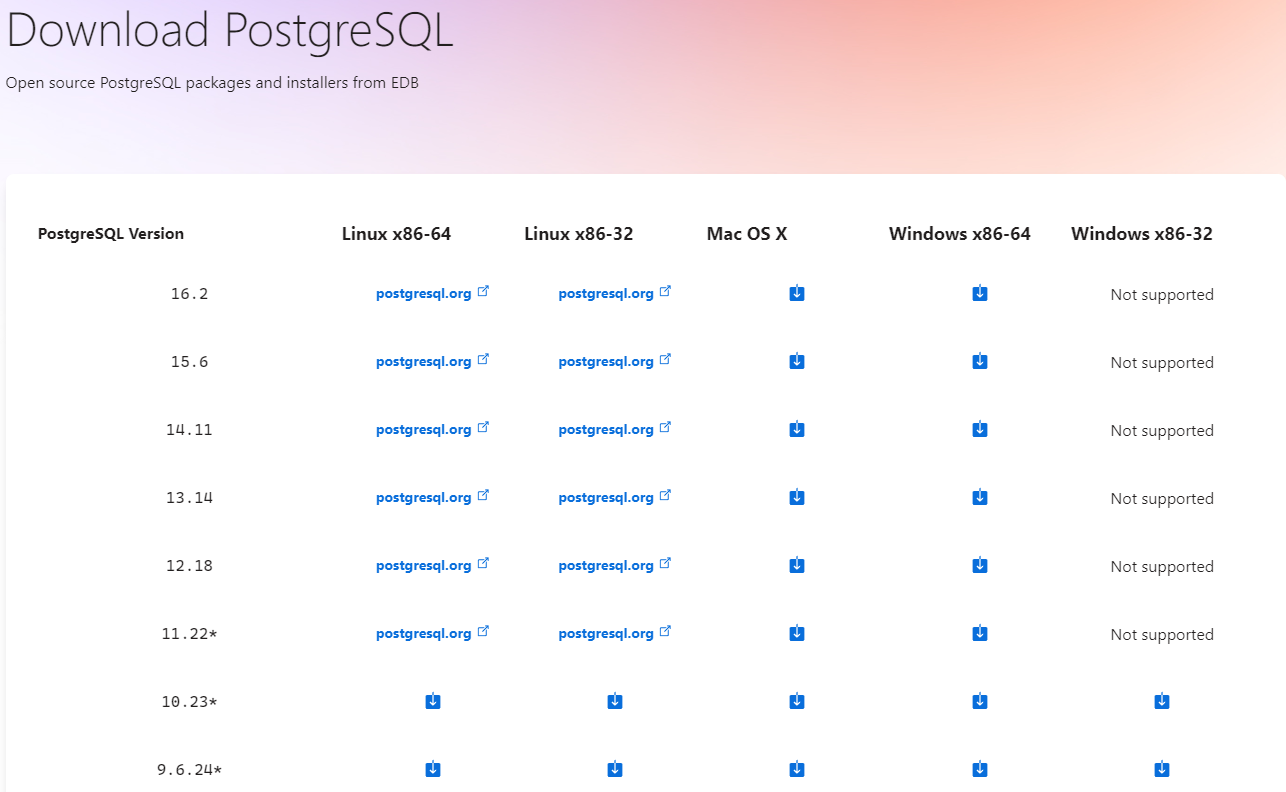
1. **Реализация задания «Модуль 2»**
   1. **Задача 1. Установка Postgre**

И так приступим к созданию подробного отчета с пошаговым описанием и описанием к скриншотам в текстовом формате подразумевает, что у нас уже есть готовые изображения для каждого шага.

Представим, как должны выглядеть шаги и описанием к скриншотам для отчета по установке Postgre и настройке подключения к ней из IntelliJ IDEA.

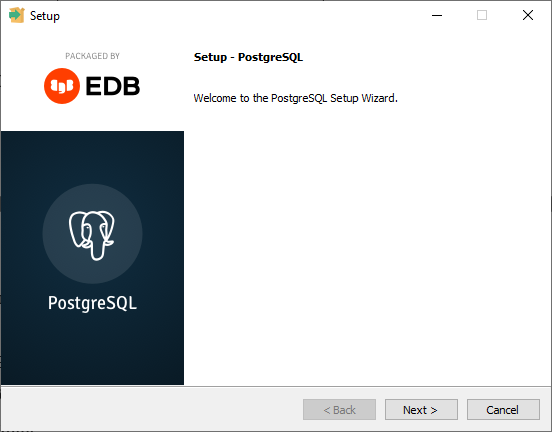
**Шаг 1: Загрузка Postgre**

****

**Скриншот 1:** Веб-страница с выбором операционной системы для загрузки Postgre.

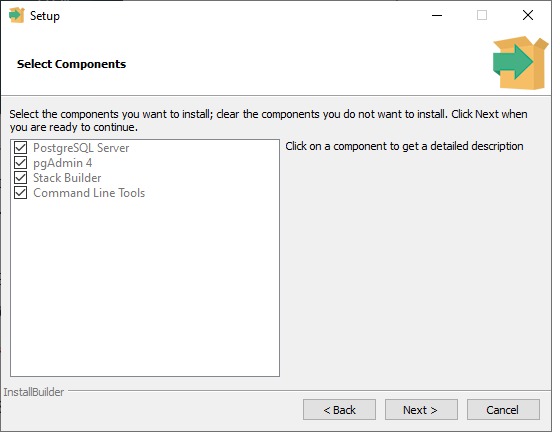
**Описание:** Страница загрузки Postgre, где я выбрал версию для Windows.

**Шаг 2: Установка Postgre**

****

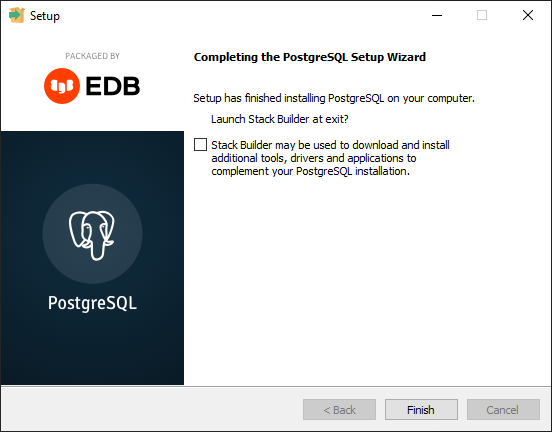
**Скриншот 2:** Окно установщика Postgre на начальном этапе.

**Описание:** Запуск установщика Postgre на локальном компьютере.



**Скриншот 3:** Выбор компонентов для установки.

**Описание:** Выбор компонентов Postgre для установки, включая сервер баз данных и pgAdmin.

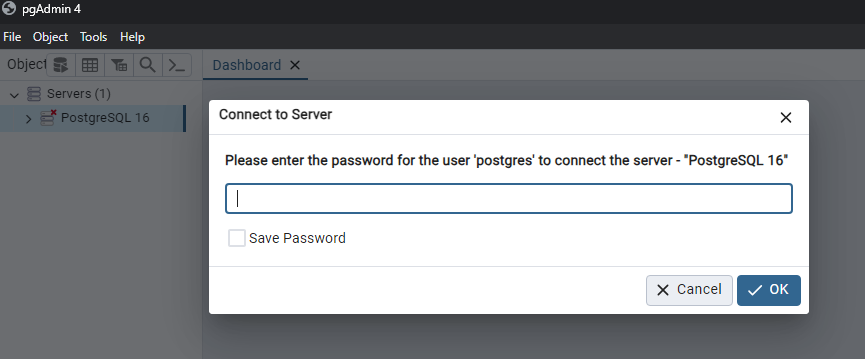


**Скриншот 4:** Завершение установки.

**Описание:** Успешное завершение установки Postgre.

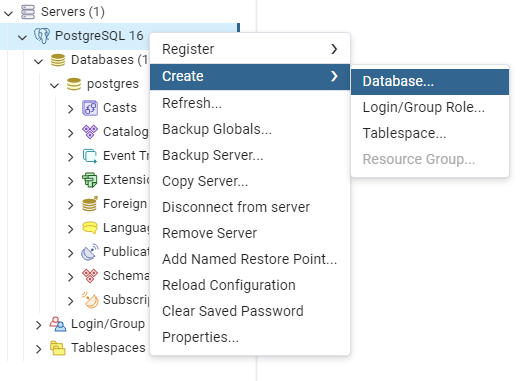
**Шаг 3: Настройка административной панели через pgAdmin**

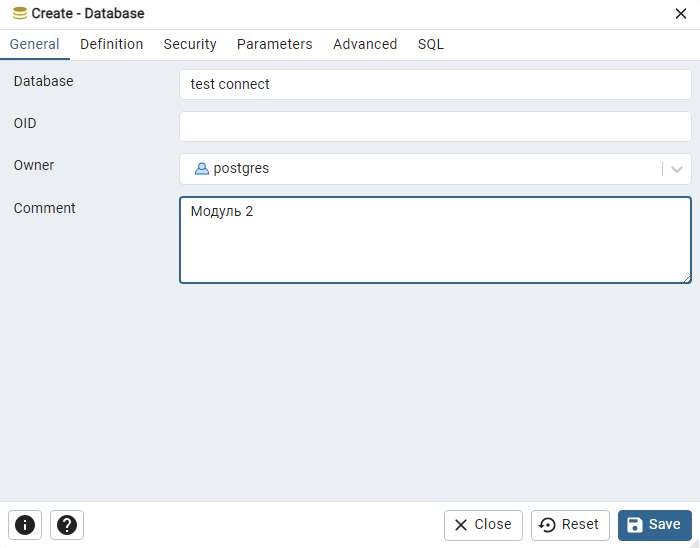




**Скриншот 5:** Окно входа в pgAdmin.

**Описание:** Вход в pgAdmin с использованием пароля администратора.

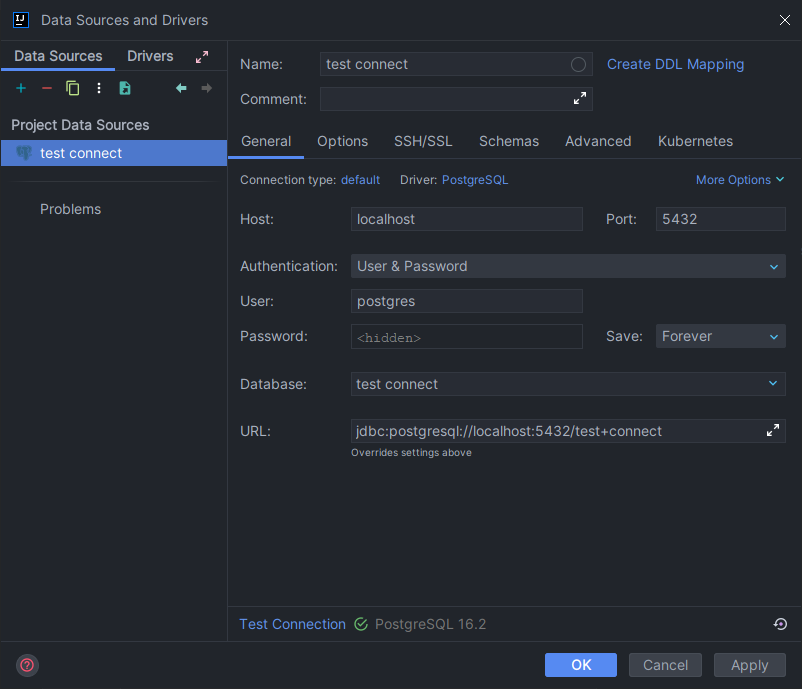
****

****

**Скриншот 6:** Создание новой базы данных через интерфейс pgAdmin.

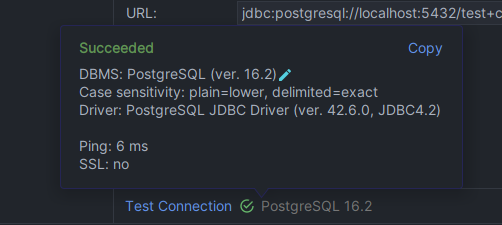
**Описание:** Использование pgAdmin для создания новой базы данных.

**Шаг 4: Проверка подключения к Postgre из IntelliJ IDEA**



**Скриншот 7:** Добавление нового источника данных в IntelliJ IDEA.

**Описание:** Конфигурация нового подключения к базе данных Postgre в IntelliJ IDEA.



**Скриншот 8:** Успешное подключение к базе данных.

**Описание:** Подтверждение успешного подключения к базе данных Postgre из IntelliJ IDEA.

* 1. **Задача 2. Создание схемы базы данных**

**Шаг 1: Создание schema.**

Созданный **schema.** файл будет содержать команды для создания трех таблиц: **Товар**, **Покупатель**, и **Заказ**, с учетом связей между ними. Ниже приведен пример содержания такого файла:

**-- Создание таблицы Товар**

CREATE TABLE IF NOT EXISTS Товар (

id SERIAL PRIMARY KEY,

описание TEXT NOT NULL,

стоимость NUMERIC NOT NULL,

количество INTEGER NOT NULL

);

**-- Создание таблицы Покупатель**

CREATE TABLE IF NOT EXISTS Покупатель (

id SERIAL PRIMARY KEY,

имя\_фамилия VARCHAR(255) NOT NULL

);

**-- Создание таблицы Заказ**

CREATE TABLE IF NOT EXISTS Заказ (

id SERIAL PRIMARY KEY,

id\_товара INTEGER NOT NULL,

id\_покупателя INTEGER NOT NULL,

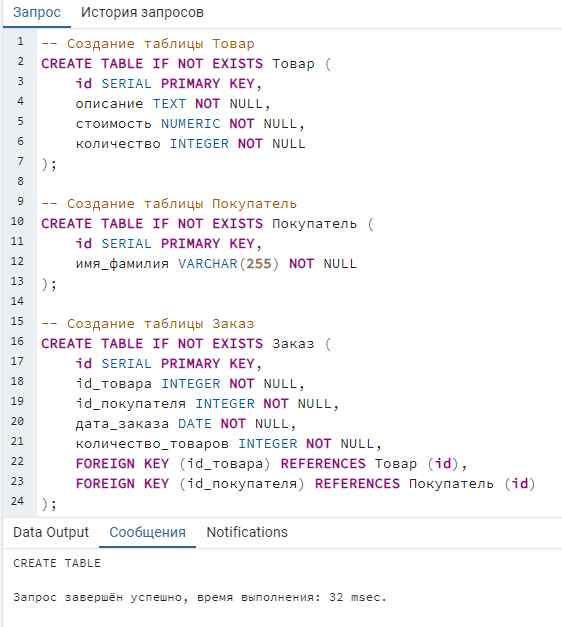
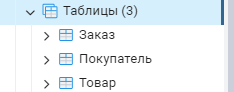
дата\_заказа DATE NOT NULL,

количество\_товаров INTEGER NOT NULL,

FOREIGN KEY (id\_товара) REFERENCES Товар (id),

FOREIGN KEY (id\_покупателя) REFERENCES Покупатель (id)

);

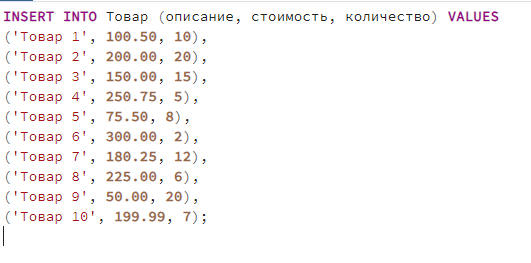


Результат создания таблиц

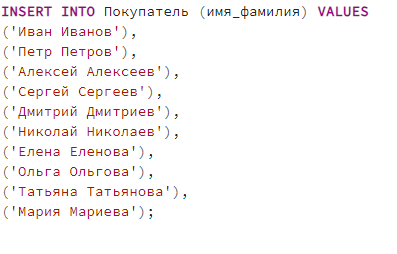
**Шаг 2: Заполнение таблиц данными**

Теперь добавим в **schema.** команды для заполнения таблиц данными.

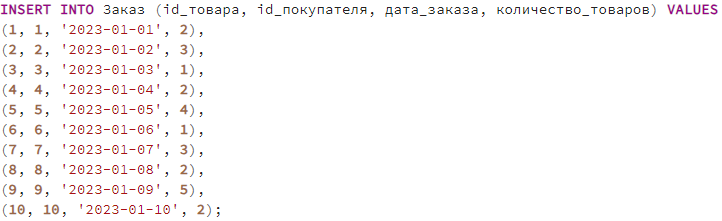
**--** Вставка данных в таблицу **Товар**



**--** Вставка данных в таблицу **Покупатель**

****

**--** Вставка данных в таблицу **Заказ**



Для вставки данных в таблицу **Заказ** мы убедимся, что **id\_товара** и **id\_покупателя** соответствуют существующим записям в таблицах **Товар** и **Покупатель**.

**Шаг 3: Написание test-queries.**

-- **Чтение данных**

SELECT \* FROM Товар;

SELECT \* FROM Покупатель;

SELECT \* FROM Заказ WHERE дата\_заказа BETWEEN '2023-01-01' AND '2023-01-31';

-- **Изменение данных**

UPDATE Товар SET количество = количество - 1 WHERE id = 1;

-- **Удаление данных**

DELETE FROM Товар WHERE id = 10;

**Запросы на чтение данных**

1. Выбрать все товары:

SELECT \* FROM Товар;

2. Выбрать имена всех покупателей:

SELECT имя\_фамилия FROM Покупатель;

3. Получить информацию о заказах, включая описание товара и имя покупателя:

SELECT Заказ.дата\_заказа, Товар.описание, Покупатель.имя\_фамилия, Заказ.количество\_товаров

FROM Заказ

JOIN Товар ON Заказ.id\_товара = Товар.id

JOIN Покупатель ON Заказ.id\_покупателя = Покупатель.id;

**Запросы на изменение данных**

4. Обновить стоимость товара:

UPDATE Товар SET стоимость = стоимость \* 1.1 WHERE id = 1;

5. Уменьшить количество товара на складе:

UPDATE Товар SET количество = количество - 1 WHERE id = 2;

6. Изменить имя покупателя:

UPDATE Покупатель SET имя\_фамилия = 'Иван Петров' WHERE id = 1;

**Запросы на удаление данных**

7. Удалить товар из таблицы Товар:

DELETE FROM Товар WHERE id = 10;

8. Удалить покупателя из таблицы Покупатель:

DELETE FROM Покупатель WHERE id = 10;

**Дополнительные запросы**

9. Выбрать все заказы, сделанные в январе 2023 года:

SELECT \* FROM Заказ WHERE дата\_заказа BETWEEN '2023-01-01' AND '2023-01-31';

10. Выбрать товары с количеством на складе меньше 5:

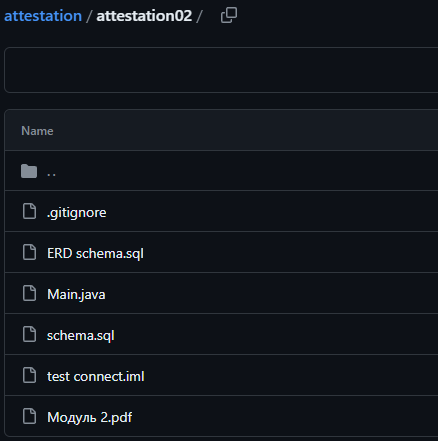
SELECT \* FROM Товар WHERE количество < 5;

**Шаг 4: Использование git**

**1. Создаем новую ветку:** `git checkout -b attestation/attestation02` для работы над заданием.

**2. Добавим файлов и коммит:** поместим `schema.sql` и `test-queries.sql` в мой Maven проект в `main -> resources`. Используем `git add .` для добавления изменений, а затем `git commit -m "Описание моего коммита"` для сохранения состояния работы.

**3.** **Публикация в репозиторий:** используем `git push origin attestation/attestation02` для отправки моей ветки в GitHub.

****

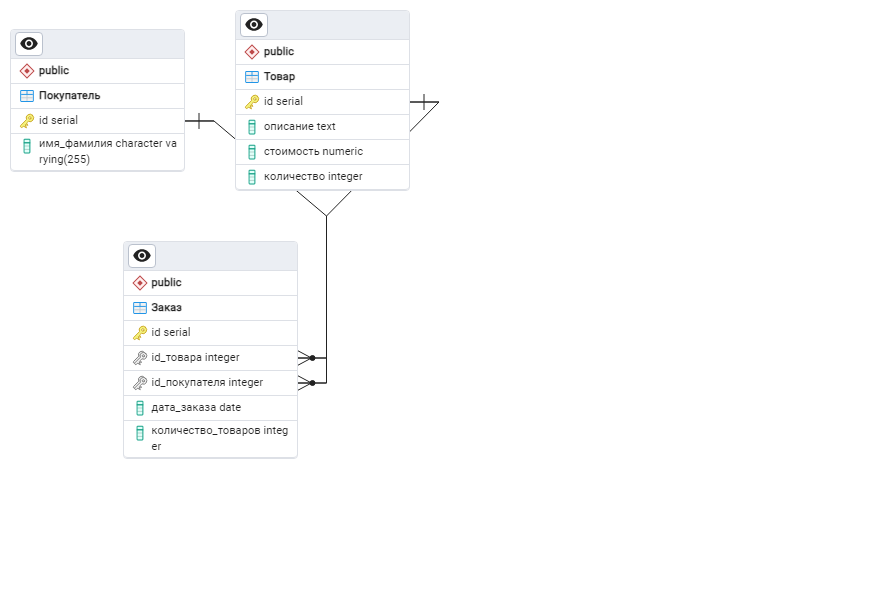


Схема БД из задания «Модуль 2»