Руководство по настройке системы голосования на Ubuntu с PostgreSQL

16 июля 2025

Введение

Данное руководство описывает настройку системы голосования на сервере Ubuntu с IP `10.130.130.15`, использующей PostgreSQL для хранения данных пользователей и результатов голосования (За, Против, Воздержался). Система включает сервер Node.js для обработки запросов и фронтенд для взаимодействия с пользователями. Особое внимание уделено безопасности: шифрование паролей, SSL, ограничение доступа и защита от повторного голосования.

Установка и настройка PostgreSQL

Установка PostgreSQL

Для установки PostgreSQL выполните следующие команды в терминале:

```
sudo apt update
sudo apt install postgresql postgresql-contrib
```

Для установки последней версии (например, PostgreSQL 16):

```
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $
(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc |
sudo apt-key add -
sudo apt update
sudo apt install postgresql-16 postgresql-client-16
```

Проверьте статус службы:

```
sudo systemctl status postgresql
```

Если служба не активна, запустите и включите автозапуск:

```
sudo systemctl start postgresql
sudo systemctl enable postgresql
```

Настройка пользователя PostgreSQL

Установите пароль для пользователя `postgres`:

```
sudo -u postgres psql
```

В psql-промпте выполните:

```
ALTER USER postgres WITH ENCRYPTED PASSWORD 'your_secure_password';
\q
```

Замените `your_secure_password` на надежный пароль.

Измените метод аутентификации на `scram-sha-256` в файле `/etc/postgresql/16/main/pg_hba.conf`:

```
sudo nano /etc/postgresql/16/main/pg_hba.conf
```

Найдите строку:

```
local   all   postgres   peer
```

Замените `peer` на `scram-sha-256`:

```
local   all   postgres   scram-sha-256
```

Перезапустите PostgreSQL:

```
sudo systemctl restart postgresql
```

Настройка подключений

Откройте файл `/etc/postgresql/16/main/postgresql.conf`:

```
sudo nano /etc/postgresql/16/main/postgresql.conf
```

Измените строку:

```
#listen_addresses = 'localhost'
```

на:

```
listen_addresses = '10.130.130.15'
```

Добавьте в `/etc/postgresql/16/main/pg_hba.conf` правило для SSL-соединений:

```
hostssl all all 10.130.130.0/24 scram-sha-256
```

Перезапустите PostgreSQL:

```
sudo systemctl restart postgresql
```

Настройка SSL

Установите Let's Encrypt для получения SSL-сертификата:

```
sudo apt install certbot
sudo certbot certonly --standalone -d your-domain.com
```

Скопируйте сертификаты в PostgreSQL:

```
sudo cp /etc/letsencrypt/live/your-domain.com/fullchain.pem
/etc/postgresql/16/main/server.crt
sudo cp /etc/letsencrypt/live/your-domain.com/privkey.pem
/etc/postgresql/16/main/server.key
sudo chown postgres:postgres /etc/postgresql/16/main/server.{crt,key}
sudo chmod 600 /etc/postgresql/16/main/server.{crt,key}
```

Перезапустите PostgreSQL:

```
sudo systemctl restart postgresql
```

Создание базы данных и таблиц

Создание базы данных

Подключитесь к PostgreSQL:

```
sudo -u postgres psql
```

Создайте базу данных:

```
CREATE DATABASE voting_system;
\q
```

Создание таблиц

Создайте таблицы `users` и `votes`:

```
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    password_hash VARCHAR(255) NOT NULL,
    email VARCHAR(100),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE votes (
    id SERIAL PRIMARY KEY,
    user_id INT NOT NULL,
    vote_option VARCHAR(10) CHECK (vote_option IN ('for', 'against',
'abstain')) NOT NULL,
    vote_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```

Сохраните этот код в файл `create_tables.sql` и выполните:

```
sudo -u postgres psql -d voting_system -f create_tables.sql
```

Добавление тестового пользователя

Сгенерируйте хэш пароля с помощью Node.js:

```
const bcrypt = require('bcrypt');
bcrypt.hash('yourpassword', 10).then(hash => console.log(hash));
```

Добавьте пользователя в таблицу `users`:

```
INSERT INTO users (username, password_hash)
VALUES ('testuser', '$2b$10$exampleHashedPassword');
```

Настройка сервера Node.js

Установка Node.js

Установите Node.js и npm:

```
sudo apt update
sudo apt install nodejs npm
```

Создайте проект:

```
mkdir voting-server
cd voting-server
npm init -y
npm install express pg bcrypt jsonwebtoken cors
```

Код сервера

Создайте файл `server.js`:

```
const express = require('express');
const { Pool } = require('pg');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const cors = require('cors');

const app = express();
app.use(express.json());
app.use(cors({ origin: 'http://10.130.130.15:8080' }));

const pool = new Pool({
  user: 'postgres',
  host: '10.130.130.15',
  database: 'voting_system',
  password: 'your_secure_password',
  port: 5432,
  ssl: { rejectUnauthorized: false }
});
```

```
const JWT_SECRET = 'your_jwt_secret_key';

app.post('/api/login', async (req, res) => {
  const { username, password } = req.body;
  try {
    const result = await pool.query('SELECT * FROM users WHERE
username = $1', [username]);
    if (result.rows.length === 0) {
      return res.status(401).json({ error: 'Неверный логин или пароль'
});
    }
    const user = result.rows[0];
    const isMatch = await bcrypt.compare(password,
user.password_hash);
    if (!isMatch) {
      return res.status(401).json({ error: 'Неверный логин или пароль'
});
    }
    const token = jwt.sign({ userId: user.id }, JWT_SECRET, {
expiresIn: '1h' });
    res.status(200).json({ message: 'Авторизация успешна', token,
userId: user.id });
  } catch (error) {
    console.error('Ошибка авторизации:', error);
    res.status(500).json({ error: 'Ошибка сервера' });
  }
});

app.post('/api/vote', async (req, res) => {
  const { token, voteOption } = req.body;
  try {
    const decoded = jwt.verify(token, JWT_SECRET);
    const userId = decoded.userId;
    const checkVote = await pool.query('SELECT * FROM votes WHERE
user_id = $1', [userId]);
    if (checkVote.rows.length > 0) {
      return res.status(400).json({ error: 'Вы уже проголосовали' });
    }
    await pool.query(
      'INSERT INTO votes (user_id, vote_option) VALUES ($1, $2)',
      [userId, voteOption]
    );
    res.status(200).json({ message: 'Голос учтен' });
  } catch (error) {
    console.error('Ошибка голосования:', error);
    res.status(500).json({ error: 'Ошибка сервера' });
  }
});
```

```
app.listen(3000, () => console.log('Сервер запущен на порту 3000'));
```

Запустите сервер:

```
node server.js
```

Замените `your_secure_password` и `your_jwt_secret_key` на свои значения.

Настройка фронтенда

HTML и JavaScript

Разместите следующий HTML-код в файле `voting.html` на веб-сервере:

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Голосование</title>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/gsap/3.11.5/gsap.min.js"><
/script>
  <link
href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.mi
n.css" rel="stylesheet">
</head>
<body class="bg-gray-100 font-sans">
  <section id="voting" class="container mx-auto p-4">
    <div class="bg-white rounded-lg shadow-lg p-6 mb-4">
      <h2 class="text-xl font-bold mb-2">Избрать председателем
собрания Пупкина Залупкина</h2>
      <button id="voteButton" class="bg-blue-500 text-white px-4 py-2
rounded-md">Проголосовать</button>
      <button id="voteButtonMobile" class="bg-blue-500 text-white px-4
py-2 rounded-md md:hidden mt-2">Проголосовать (мобильная
версия)</button>
    </div>
  </section>
  <div id="loginModal" class="fixed inset-0 hidden bg-black bg-
opacity-50 flex items-center justify-center z-50">
    <div class="bg-white rounded-lg p-6 w-full max-w-md">
      <h3 class="text-lg font-semibold mb-4">Авторизация</h3>
      <form id="loginForm" class="space-y-4">
        <div>
          <label for="username" class="block text-sm font-
medium">Логин</label>
          <input type="text" id="username" class="w-full border
```

```
rounded-md p-2" required>
        </div>
        <div>
          <label for="password" class="block text-sm font-
medium">Пароль</label>
          <input type="password" id="password" class="w-full border
rounded-md p-2" required>
        </div>
        <p id="errorMessage" class="text-red-500 hidden"></p>
        <div class="flex justify-end space-x-2">
          <button type="button" id="cancelButton" class="bg-gray-300
text-gray-700 px-4 py-2 rounded-md">Отмена</button>
          <button type="submit" class="bg-blue-500 text-white px-4 py-
2 rounded-md">Войти</button>
        </div>
      </form>
    </div>
  </div>
  <div id="voteModal" class="fixed inset-0 hidden bg-black bg-opacity-
50 flex items-center justify-center z-50">
    <div class="bg-white rounded-lg p-6 w-full max-w-md">
      <h3 class="text-lg font-semibold mb-4">Голосование за Пупкина
3.</h3>
      <div class="flex space-x-4">
        <button class="vote-option bg-green-500 text-white px-4 py-2
rounded-md" data-option="for">За</button>
        <button class="vote-option bg-red-500 text-white px-4 py-2
rounded-md" data-option="against">Против</button>
        <button class="vote-option bg-yellow-400 text-white px-4 py-2
rounded-md" data-option="abstain">Воздержался</button>
      </div>
      <p id="voteErrorMessage" class="text-red-500 hidden mt-4"></p>
      <button id="closeVoteModal" class="mt-4 bg-gray-300 text-gray-
700 px-4 py-2 rounded-md">Закрыть</button>
    </div>
  </div>
  <script>
    const voteButton = document.getElementById('voteButton');
    const voteButtonMobile =
document.getElementById('voteButtonMobile');
    const loginModal = document.getElementById('loginModal');
    const voteModal = document.getElementById('voteModal');
    const loginForm = document.getElementById('loginForm');
    const cancelButton = document.getElementById('cancelButton');
    const errorMessage = document.getElementById('errorMessage');
    const voteErrorMessage =
document.getElementById('voteErrorMessage');
    const closeVoteModal = document.getElementById('closeVoteModal');
    let authToken = null;
```

```
    function toggleModal(modal, show) {
      gsap.to(modal, {
        duration: 0.3,
        opacity: show ? 1 : 0,
        display: show ? 'flex' : 'none',
        ease: 'power2.out'
      });
    }

    voteButton.addEventListener('click', () => toggleModal(loginModal,
true));
    voteButtonMobile.addEventListener('click', () =>
toggleModal(loginModal, true));
    cancelButton.addEventListener('click', () =>
toggleModal(loginModal, false));
    closeVoteModal.addEventListener('click', () =>
toggleModal(voteModal, false));

    loginForm.addEventListener('submit', async (event) => {
      event.preventDefault();
      const username = document.getElementById('username').value;
      const password = document.getElementById('password').value;
      try {
        const response = await
fetch('http://10.130.130.15:3000/api/login', {
          method: 'POST',
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify({ username, password })
        });
        const data = await response.json();
        if (response.ok) {
          authToken = data.token;
          toggleModal(loginModal, false);
          toggleModal(voteModal, true);
        } else {
          errorMessage.classList.remove('hidden');
          errorMessage.textContent = data.error || 'Ошибка
авторизации';
        }
      } catch (error) {
        errorMessage.classList.remove('hidden');
        errorMessage.textContent = 'Ошибка соединения с сервером';
      }
    });

    document.querySelectorAll('.vote-option').forEach(button => {
      button.addEventListener('click', async () => {
        const voteOption = button.dataset.option;
        try {
          const response = await
```

```
fetch('http://10.130.130.15:3000/api/vote', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({ token: authToken, voteOption })
          });
          const data = await response.json();
          if (response.ok) {
            alert('Ваш голос учтен!');
            toggleModal(voteModal, false);
          } else {
            voteErrorMessage.classList.remove('hidden');
            voteErrorMessage.textContent = data.error || 'Ошибка при
голосовании';
          }
        } catch (error) {
          voteErrorMessage.classList.remove('hidden');
          voteErrorMessage.textContent = 'Ошибка соединения с
сервером';
        }
      });
    });
  </script>
</body>
</html>
```

Разместите файл на веб-сервере:

```
npm install -g http-server
http-server -p 8080
```

Обеспечение безопасности

Шифрование паролей

Используйте `bcrypt` для хэширования паролей:

```
const bcrypt = require('bcrypt');
bcrypt.hash('yourpassword', 10).then(hash => console.log(hash));
```

JWT-токены

После авторизации сервер выдает JWT-токен с временем действия 1 час. Замените `your_jwt_secret_key` на надежный ключ, сгенерированный, например:

```
openssl rand -base64 32
```

SSL/TLS

PostgreSQL настроен с SSL. Для фронтенда и сервера настройте HTTPS через Nginx:

```
sudo apt install nginx
sudo nano /etc/nginx/sites-available/voting
```

Пример конфигурации Nginx:

```
server {
    listen 443 ssl;
    server_name your-domain.com;
    ssl_certificate
/etc/letsencrypt/live/your-domain.com/fullchain.pem;
    ssl_certificate_key
/etc/letsencrypt/live/your-domain.com/privkey.pem;

    location / {
        proxy_pass http://10.130.130.15:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

Активируйте конфигурацию:

```
sudo ln -s /etc/nginx/sites-available/voting /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl restart nginx
```

Ограничение доступа

Ограничьте доступ к портам:

```
sudo ufw allow 5432/tcp comment 'PostgreSQL'
sudo ufw allow 3000/tcp comment 'Node.js server'
sudo ufw allow 80/tcp comment 'HTTP'
sudo ufw allow 443/tcp comment 'HTTPS'
sudo ufw enable
```

Роли и привилегии

Создайте пользователя для приложения:

```
CREATE ROLE app_user WITH LOGIN PASSWORD 'app_secure_password';
GRANT CONNECT ON DATABASE voting_system TO app_user;
GRANT SELECT, INSERT ON TABLE users, votes TO app_user;
```

Обновите `pool` в `server.js` для использования `app_user`.

Защита от повторного голосования

Сервер проверяет, голосовал ли пользователь, через запрос к таблице `votes`.

Тестирование системы

Проверка базы данных

Подключитесь к базе:

```
sudo -u postgres psql -d voting_system
```

Проверьте таблицы и данные:

```
\dt
SELECT * FROM users;
SELECT * FROM votes;
```

Проверка сервера

Тест авторизации:

```
curl -X POST http://10.130.130.15:3000/api/login -H "Content-Type:
application/json" -d
'{"username":"testuser","password":"yourpassword"}'
```

Тест голосования (замените `<token>` на полученный токен):

```
curl -X POST http://10.130.130.15:3000/api/vote -H "Content-Type:
application/json" -d '{"token":"<token>","voteOption":"for"}'
```

Проверка фронтенда

Откройте http://10.130.130.15:8080/voting.html, нажмите “Проголосовать”, введите логин/пароль, выберите вариант голоса и проверьте запись в таблице `votes`.

Дополнительные рекомендации

Резервное копирование

Создайте резервную копию:

```
sudo -u postgres pg_dump voting_system > backup.sql
```

Настройте регулярное копирование через `cron`.

Мониторинг

Установите `pgAdmin` для управления базой:

```
sudo apt install pgadmin4
```

Логирование

Добавьте логирование с помощью `winston` в `server.js`:

```
const winston = require('winston');
const logger = winston.createLogger({
  transports: [new winston.transports.File({ filename:
'server.log' })]
});
```

Заключение

Вы настроили безопасную систему голосования с PostgreSQL, Node.js и фронтендом. Для дальнейшей помощи обратитесь к документации PostgreSQL (https://www.postgresql.org) или Node.js (https://nodejs.org).