Разработка Веб-Фронтенда

На нашем будущем сервере будет постоянно запущен веб-сервер Nginx. Он предоставит пользователю возможность управлять системой через веб-приложение в окне браузера по временному порту 7000. Если пользователь захочет использовать собственный веб-сервер, в приложении ему будет доступен веб-сервер Apache2 с поддержкой PHP. Службы OpenSSH и Samba, автозапуск которых настраивается для создания веб-приложения, также будут по умолчанию отключены с возможностью их постоянного включения через панель управления.

Настройка Nginx на обработку PHP

Файл nginx.conf

Мы перепишем конфигурационный файл nginx.conf, добавив правильный блок location ~ \.php$, работающий через UNIX-сокет.

#bash

```
cat << 'EOF' | sudo tee /etc/nginx/nginx.conf > /dev/null
worker_processes  1;

events {
    worker_connections  1024;
}

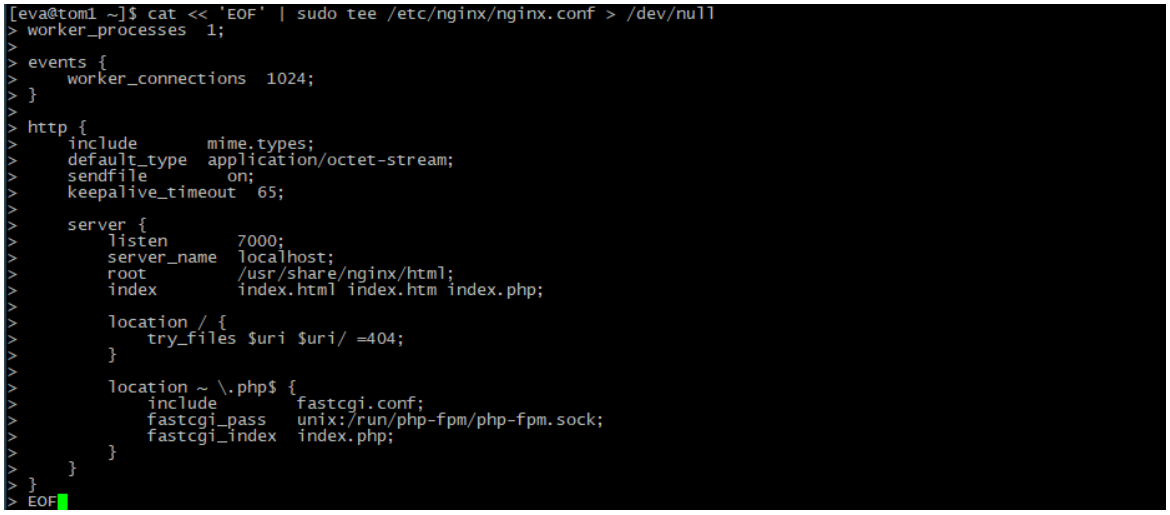
http {
    include       mime.types;
    default_type  application/octet-stream;
    sendfile        on;
    keepalive_timeout  65;

    server {
        listen       7000;
        server_name  localhost;
        root         /usr/share/nginx/html;
        index        index.html index.htm index.php;

        location / {
            try_files $uri $uri/ =404;
        }

        location ~ \.php$ {
            include        fastcgi.conf;
            fastcgi_pass   unix:/run/php-fpm/php-fpm.sock;
            fastcgi_index  index.php;
        }
    }
```

```
}
EOF
```

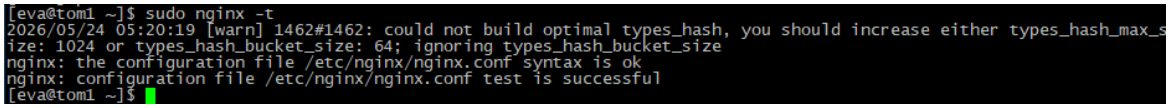


Тестирование синтаксиса nginx.conf

Нам нужно протестировать синтаксис Nginx, чтобы убедиться, что все скобки закрыты и сокет прописан без ошибок.

#bash

```
sudo nginx -t
```



(Тест пройден успешно (syntax is ok, test is successful). Предупреждение о types_hash — это стандартный информационный ворнинг, на работу он не влияет.)

Проверка содержимого файла nginx.conf

#bash

```
cat -n /etc/nginx/nginx.conf
```

```
[eva@tom1 ~]$ cat -n /etc/nginx/nginx.conf
     1  worker_processes  1;
     2
     3  events {
     4      worker_connections  1024;
     5  }
     6
     7  http {
     8      include       mime.types;
     9      default_type  application/octet-stream;
    10      sendfile        on;
    11      keepalive_timeout  65;
    12
    13      server {
    14          listen       7000;
    15          server_name  localhost;
    16          root         /usr/share/nginx/html;
    17          index        index.html index.htm index.php;
    18
    19          location / {
    20              try_files $uri $uri/ =404;
    21          }
    22
    23          location ~ \.php$ {
    24              include        fastcgi.conf;
    25              fastcgi_pass   unix:/run/php-fpm/php-fpm.sock;
    26              fastcgi_index  index.php;
    27          }
    28      }
    29  }
[eva@tom1 ~]$
```

Перезапуск веб-сервера

#bash

```
sudo systemctl restart nginx
```

```
[eva@tom1 ~]$ sudo systemctl restart nginx
[eva@tom1 ~]$
```

Проверка статуса службы

#bash

```
sudo systemctl status nginx
```

```
[eva@tom1 ~]$ sudo systemctl status nginx
● nginx.service - nginx web server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: disabled)
     Active: active (running) since Sun 2026-05-24 05:31:24 UTC; 2min 13s ago
 Invocation: c70deed23faa4bcca7e4807d5183a936
    Process: 1520 ExecStart=/usr/bin/nginx (code=exited, status=0/SUCCESS)
   Main PID: 1521 (nginx)
      Tasks: 2 (limit: 11644)
     Memory: 4.8M (peak: 5.3M)
        CPU: 16ms
     CGroup: /system.slice/nginx.service
             ├─1521 "nginx: master process /usr/bin/nginx"
             └─1522 "nginx: worker process"

May 24 05:31:24 tom1 systemd[1]: Stopping nginx web server...
May 24 05:31:24 tom1 systemd[1]: nginx.service: Deactivated successfully.
May 24 05:31:24 tom1 systemd[1]: Stopped nginx web server.
May 24 05:31:24 tom1 systemd[1]: Starting nginx web server...
May 24 05:31:24 tom1 nginx[1520]: 2026/05/24 05:31:24 [warn] 1520#1520: could not build optimal types_hash, you shou
d increase either types_hash_max_size: 1024 or types_hash_bucket_size: 64; ignoring types_hash_bucket_size
May 24 05:31:24 tom1 systemd[1]: Started nginx web server.
[eva@tom1 ~]$
```

Проверим директорию

Чтобы не запутаться в структуре бэкенда и фронтенда, давайте сначала проверим, что сейчас вообще находится внутри корневой папки Nginx на tom_1.

#bash

```
ls -la /usr/share/nginx/html/
```



Проверим запуск и работу веб-сервера на порту 7000 в браузере на странице http://192.168.1.72:7000/



Проверяем автозапуск службы php-fpm

Убедимся, что служба PHP-FPM активирована в автозапуске, чтобы при старте флешки в ОЗУ она запустилась сама вместе с Nginx.

#bash

```
systemctl is-enabled php-fpm
```



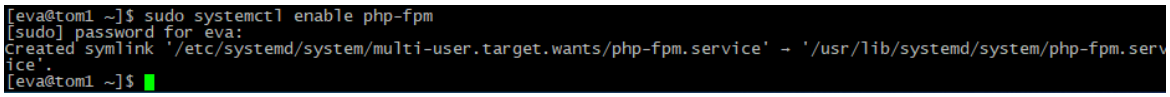
(Примечание: статус disabled - выключена)

Включаем службу php-fpm в автозапуск

Включим службу PHP-FPM в автозапуске, чтобы при старте флешки в ОЗУ она запустилась сама вместе с Nginx.

#bash

```
sudo systemctl enable php-fpm
```



(Примечание: созданы системные симлинки)

Проверяем статус автозапуска службы php-fpm

Убедимся, что служба PHP-FPM активирована в автозапуске, чтобы при старте флешки в ОЗУ она запустилась сама вместе с Nginx. Убедимся, что система теперь рапортует правильный статус.

#bash

```
systemctl is-enabled php-fpm
```



(Примечание: статус enabled - включена)

Снятие системной изоляции с PHP

В Arch Linux служба `php-fpm` по умолчанию заперта (ProtectSystem=full). Из-за этого PHP видит системные файлы пользователей как Read-Only. Снимем это ограничение.

Откройте переопределение настроек службы:

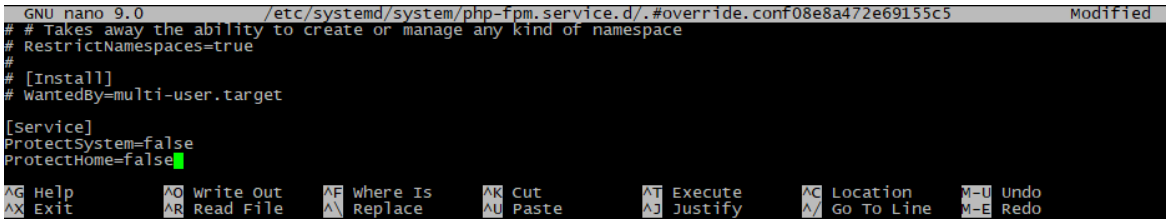
#bash

```
sudo systemctl edit php-fpm
```

Вставте свои строки строго между первой и второй строками комментариев (обычно там есть явная подсказка ### Lines below this comment will be discarded или аналогичная).:

ini

```
[Service]
ProtectSystem=false
ProtectHome=false
```



(Ctrl+O для сохранения, затем Enter и Ctrl+X для выхода)



(Вывод означает правильно отредактированную конфигурацию, и systemd успешно создал конфигурационный файл (drop-in файл) с вашими настройками по пути /etc/systemd/system/php-fpm.service.d/override.conf)

Перезапустите службу:

```
sudo systemctl restart php-fpm
```



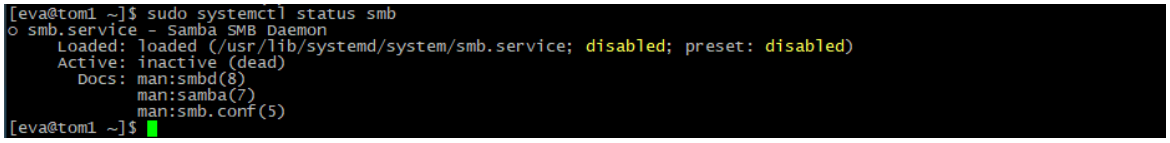
Настройка Samba-сервера

Проверяем состояние Samba-сервера на tom_1

Нам нужно узнать, запущен ли демон Samba (smb), чтобы понять, сможем ли мы сразу подключить сетевой диск в Windows.
Выполните в терминале команду:

#bash

```
sudo systemctl status smb
```



(Этот вывод означает, что служба установлена в системе, но её автозапуск отключен и в данный момент служба не запущена (остановлена))

Создание конфигурационного файла smb.conf

Создадим конфигурационный файл в редакторе nano.
Выполните в терминале команду:

#bash

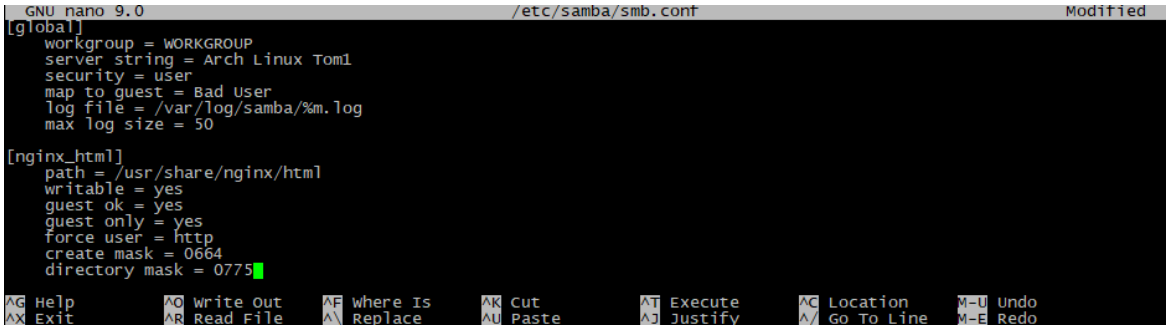
```
sudo nano /etc/samba/smb.conf
```

Файл пустой и готов к заполнению. Вставьте в него следующий минимальный рабочий конфиг, чтобы открыть доступ к директории Nginx (/usr/share/nginx/html) с автоматическим наследованием безопасных прав доступа (775 для папок и 664 для файлов):

ini

```
[global]
    workgroup = WORKGROUP
    server string = Arch Linux Tom1
    security = user
    map to guest = Bad User
    log file = /var/log/samba/%m.log
    max log size = 50

[nginx_html]
    path = /usr/share/nginx/html
    writable = yes
    guest ok = yes
    guest only = yes
    force user = http
    create mask = 0664
    directory mask = 0775
```



Файл изменен. Нажмите последовательно:CTRL + O, затем клавишу Enter (для записи файла).CTRL + X (для выхода из редактора nano)

Проверка синтаксиса

выполните встроенную команду Samba для проверки синтаксиса файла конфигурации:

#bash

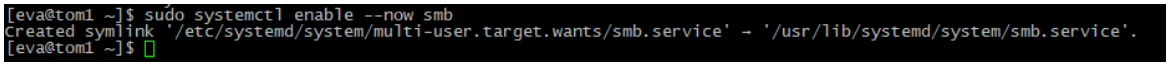
```
testparm -s
```



Тест синтаксиса пройден успешно (Loaded services file OK). Ошибок в файле smb.conf нет.
Сетевая папка nginx_html определена верно.
Следующий шаг — запуск и добавление службы Samba в автозагрузку.

#bash

```
sudo systemctl enable --now smb
```

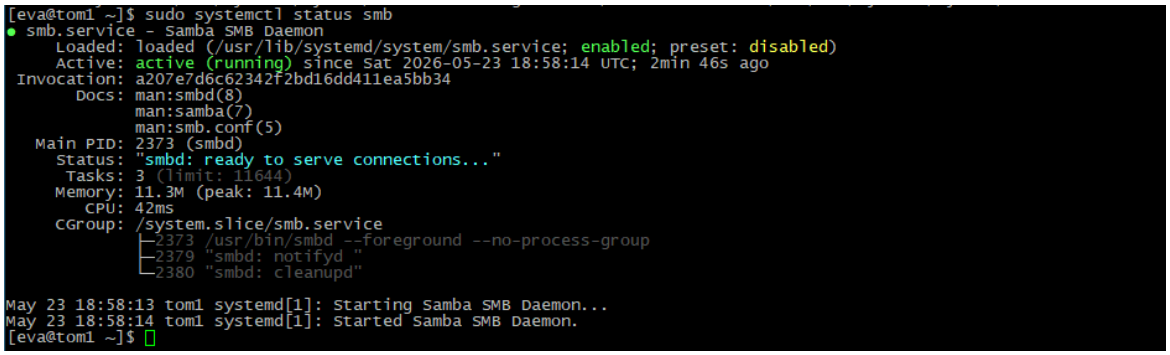


(Симлинк успешно создан, служба добавлена в автозапуск.)
Следующий шаг — обязательная проверка статуса запущенного демона Samba.

Проверка статуса

#bash

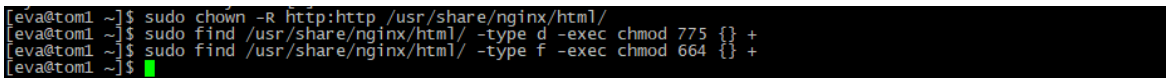
```
sudo systemctl status smb
```



(Служба smb работает в режиме active (running) и полностью готова принимать сетевые подключения.)
Службы настроены. Передаем владение корневым каталогом сайта встроенному веб-пользователю http. Задаем права 775 для всех папок (чтобы Samba и Nginx могли создавать файлы) и 664 для файлов (только чтение и запись, без флагов исполнения):.
Выполните в терминале команду:

#bash

```
sudo chown -R http:http /usr/share/nginx/html/
sudo find /usr/share/nginx/html/ -type d -exec chmod 775 {} +
sudo find /usr/share/nginx/html/ -type f -exec chmod 664 {} +
```



(Права 777/664 назначены успешно.)
Следующий обязательный шаг по нашему плану — проверка того, как система применила эти права к содержимому каталога.

#bash

```
ls -la /usr/share/nginx/html
```



(Проверка прав прошла успешно. Строки drwxrwxr-x для текущей папки (.) и -rw-rw-r– для файлов index.html и 50x.html подтверждают, что доступ полностью открыт на чтение, запись для пользователей системы.)

Подключение сетевой папки в Windows

Получим точный текущий IP-адрес сервера tom_1

Выполните в терминале команду:

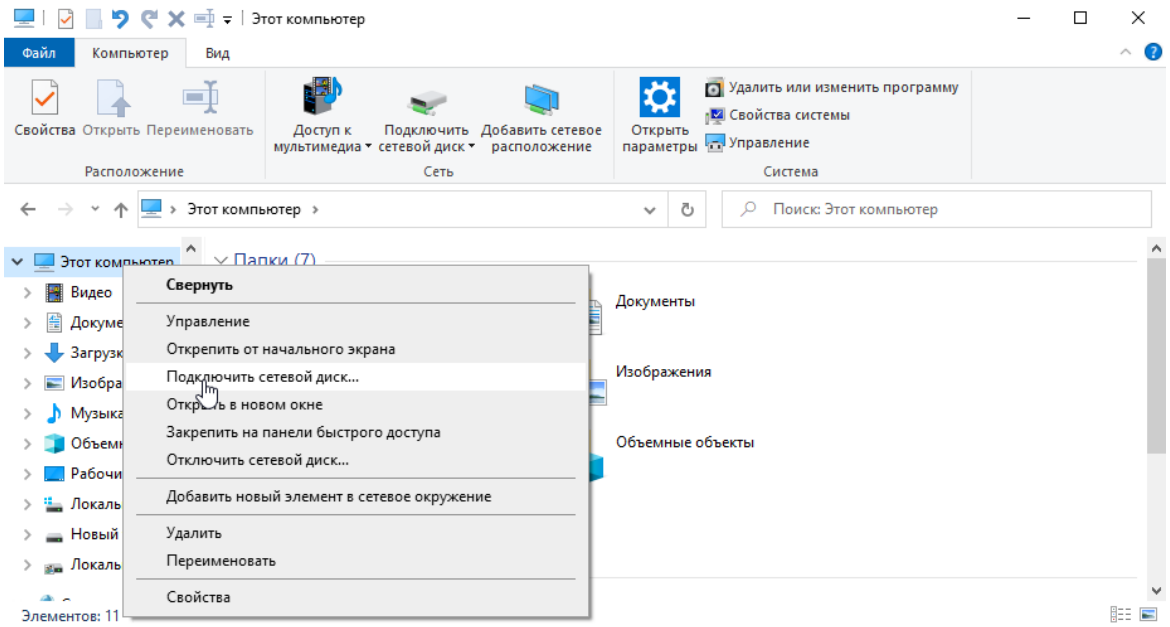
#bash

```
ip -br address show scope global | awk '{print $3}' | cut -d/ -f1
```

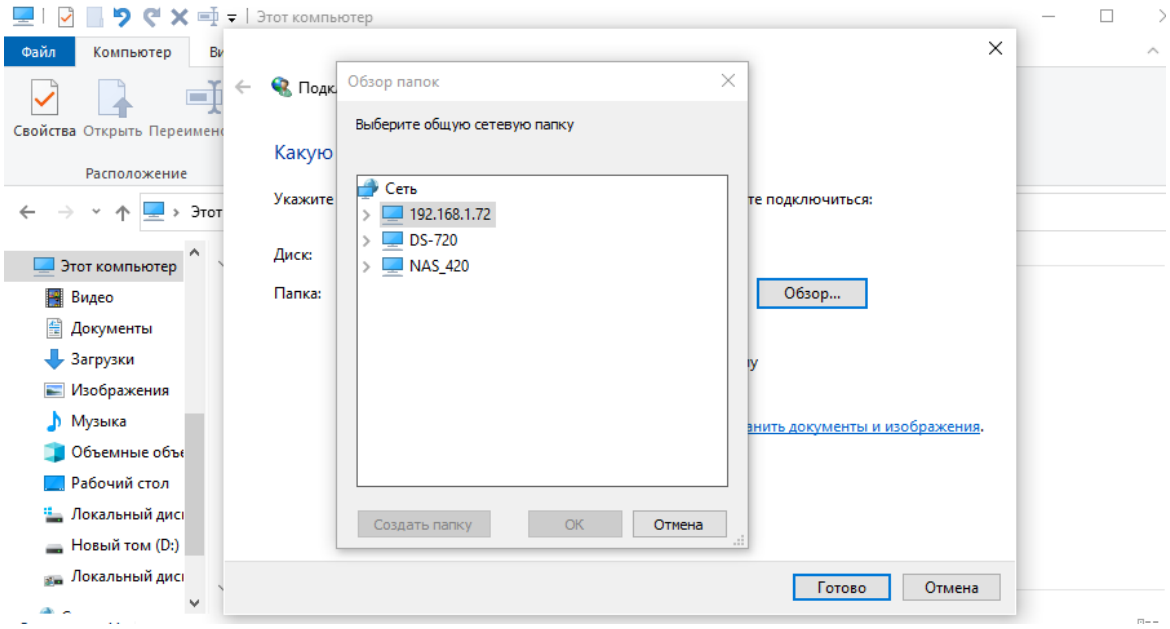
```
[eva@tom1 ~]$ ip -br address show scope global | awk '{print $3}' | cut -d/ -f1
192.168.1.72
[eva@tom1 ~]$
```

Теперь папка готова к подключению в качестве сетевого диска в среде Windows, чтобы вы могли открыть её через Notepad++.

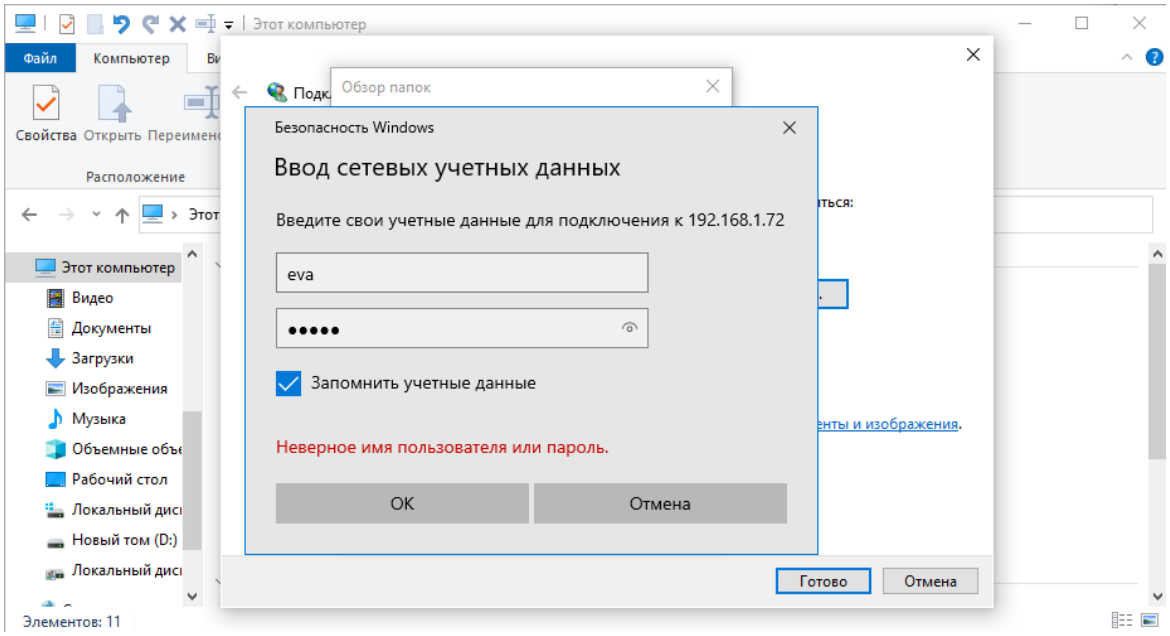
1. Откройте Проводник (Этот компьютер) на вашей Windows-машине.
2. В верхнем меню нажмите кнопку «Подключить сетевой диск» (или нажмите правой кнопкой мыши по «Этот компьютер» → «Подключить сетевой диск»).



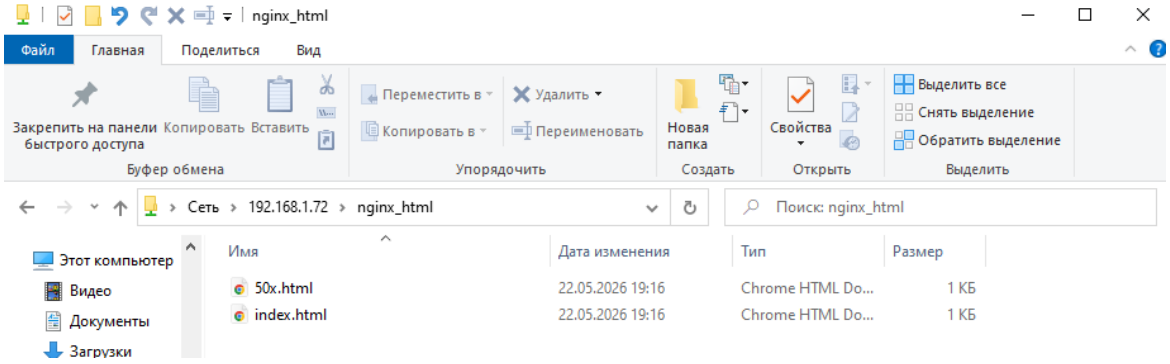
1. В поле «Папка» введите сетевой путь, используя IP-адрес вашего сервера tom_1 (из логов PuTTY: 192.168.1.72 или через кнопку обзор)



Введите сетевые учетные данные



Зайдите через проводник



(В корне /usr/share/nginx/html/ только файлы 50x.html и index.html.)
Разворачиваем структуру каталогов для нашего веб-интерфейса. Создадим стандартные папки для стилей, скриптов и серверной логики.

Системный пользователь http

Пользователь http в Arch Linux — это встроенный системный пользователь, от имени которого по умолчанию работают веб-серверы (например, Apache или Nginx) и сопутствующие им службы.
Он создается автоматически при установке этих программ для изоляции процессов и обеспечения безопасности.

Назначение

- Безопасность: Веб-серверы не должны работать под правами суперпользователя (root). Если злоумышленник найдет уязвимость в вашем сайте, он получит доступ только к файлам с правами пользователя http, что убережет остальную систему от взлома.
- Права на файлы: Этот пользователь владеет файлами и папками, к которым сервер имеет доступ (обычно они расположены в директории /srv/http/).
- Группа http: Для удобства существует одноименная группа http, в которую могут входить другие пользователи, чтобы иметь возможность редактировать файлы сайта без изменения прав доступа к ним через sudo

Применение

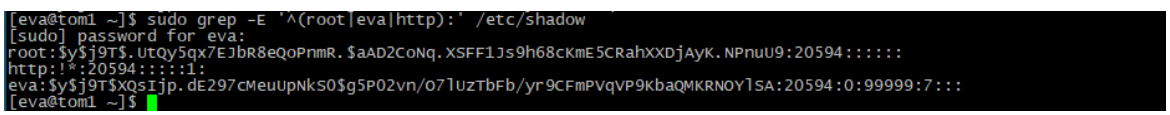
- Размещение сайтов: При настройке Nginx, Apache или PHP-FPM часто требуется указывать, что процесс должен запускаться от имени http.
- Настройка разрешений (Permissions): Если сайт выдает ошибку доступа (например, 403 Forbidden), обычно это означает, что у пользователя http нет прав на чтение нужных файлов или папок.
- Безопасность каталогов: Если скриптам на сайте нужно загружать файлы на сервер, папке загрузки необходимо выдать права (владельца) для пользователя http

Вывод строк пользователей системы

Выводим строки трех нами известных пользователей из базы данных.

#bash

```
sudo grep -E '^(root|eva|http):' /etc/shadow
```



root:yj9T$...:20594:::::

- Второе поле — длинный хэш y.... Это зашифрованный пароль суперпользователя.Число 20594 — дата последнего изменения пароля (в днях от 1970 года).В конце строки — пустые поля. Это значит, что для root нет никаких ограничений.

eva:yj9T$...:20594:0:99999:7:::

- Тоже видим хэш личного пароля.
- Параметры 0:99999:7 означают стандартные правила пользователя: пароль можно менять сразу (0), он действует 99999 дней, а за 7 дней до истечения система начнет предупреждать. Восьмое поле пустое — аккаунт не блокируется.

http:!*:20594::::::1:

- Второе поле содержит !*. Это маркер того, что пароль заблокирован (вход по паролю невозможен, учетка техническая).Внимание в самый конец строки: ::::::1:
- На предпоследней позиции (8-е поле) стоит цифра 1. В системе Linux это означает, что учетная запись принудительно заблокирована подсистемой безопасности PAM через 1 день после начала эпохи Unix (то есть 2 января 1970 года).

Изменяем параметры пользователя http

Уберем эту единицу из конца строки, чтобы сделать учетную запись бессрочной.

#bash

```
sudo chage -E -1 http
```



Проверка изменений в файле /etc/shadow

#bash

```
sudo grep '^http:' /etc/shadow
```



(Строка завершается чистыми двоеточиями (::::::), что означает: блокировка PAM полностью снята, аккаунт http стал бессрочным)

Настройка беспарольного доступа в sudoers

Чтобы PHP-скрипты могли вызывать утилиты управления аккаунтами без ввода пароля и без наличия текстового экрана (TTY), настроим правила безопасности. Команды будут пробрасываться во внешнюю систему через утилиту `systemd-run` для обхода ограничений безопасности PHP.

Откройте конфигурационный файл строго через visudo:

#bash

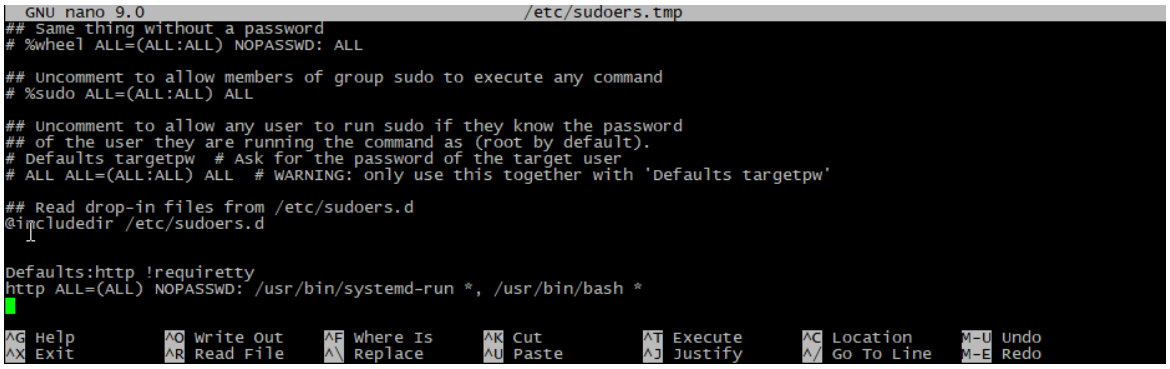
```
sudo EDITOR=nano visudo
```



В самый конец файла добавьте следующие строки:

text

```
Defaults:http !requiretty
http ALL=(ALL) NOPASSWD: /usr/bin/systemd-run *, /usr/bin/bash *
```



Обновление контекста PHP-FPM

Так как PHP кэширует права сессий для вызова exec(), обязательно примените изменения через перезапуск служб, поочередно введя 3 команды:

#bash

```
sudo systemctl daemon-reload
sudo systemctl restart php-fpm
sudo systemctl restart nginx
```

```
[eva@tom1 ~]$ sudo systemctl restart nginx^C
[eva@tom1 ~]$ sudo systemctl daemon-reload
[eva@tom1 ~]$ sudo systemctl restart php-fpm
[eva@tom1 ~]$ sudo systemctl restart nginx
[eva@tom1 ~]$
```

Схема веб-панели управления в основной системе (tom_1)

Папка веб-сервера nginx_html находится по пути /usr/share/nginx/html/ и имеет следующую структуру файлов бэкенда (PHP) и фронтенда (JS/CSS):

```
/usr/share/nginx/html/                  # Корневая директория веб-сервера Nginx
├── index.html                          # Главный интерфейс панели (вкладки, таблицы, модальные окна)
├── css/
│   └── style.css                       # Стили оформления интерфейса панели управления
├── js/
│   └── app.js                          # Клиентская логика (асинхронные Fetch-запросы к API, фильтры)
└── api/
    ├── users.php                       # Серверный обработчик для системных пользователей (/etc/passwd)
    └── groups.php                      # Серверный обработчик для системных групп (/etc/group)
```

Создание директорий

Временно изменим права для работы в консоли

#bash

```
sudo chown -R http:http /usr/share/nginx/html/
sudo find /usr/share/nginx/html/ -type d -exec chmod 775 {} +
```

```
[eva@tom1 ~]$ sudo chown -R http:http /usr/share/nginx/html/
[eva@tom1 ~]$ sudo find /usr/share/nginx/html/ -type d -exec chmod 775 {} +
[eva@tom1 ~]$
```

Выполните в терминале PuTTY одну команду:

#bash

```
mkdir -p /usr/share/nginx/html/{css,js,api,assets}
```

```
[eva@tom1 ~]$ mkdir -p /usr/share/nginx/html/{css,js,api,assets}
[eva@tom1 ~]$
```

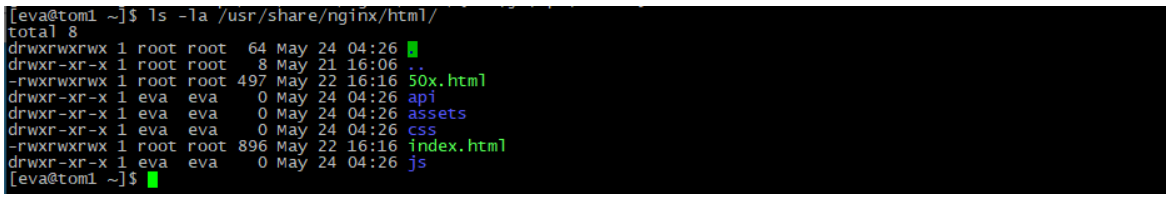
Папки созданы. Теперь обязательный шаг контроля: проверяем, какие права доступа и

владельцы назначены для новых директорий, чтобы Windows-пользователь Samba и веб-сервер Nginx могли с ними работать.

Контроль папок

#bash

```
ls -la /usr/share/nginx/html/
```



(Папки создались под пользователем eva, но у них стоят ограниченные права drwxr-xr-x. Из-за этого Windows через Samba не сможет создавать или изменять файлы внутри этих подкаталогов.)

Права доступа к файлам

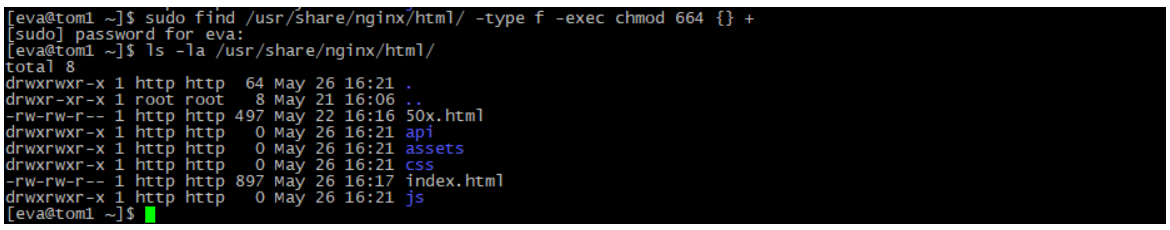
#bash

```
sudo find /usr/share/nginx/html/ -type f -exec chmod 664 {} +
```

Проверка назначения прав пользователя

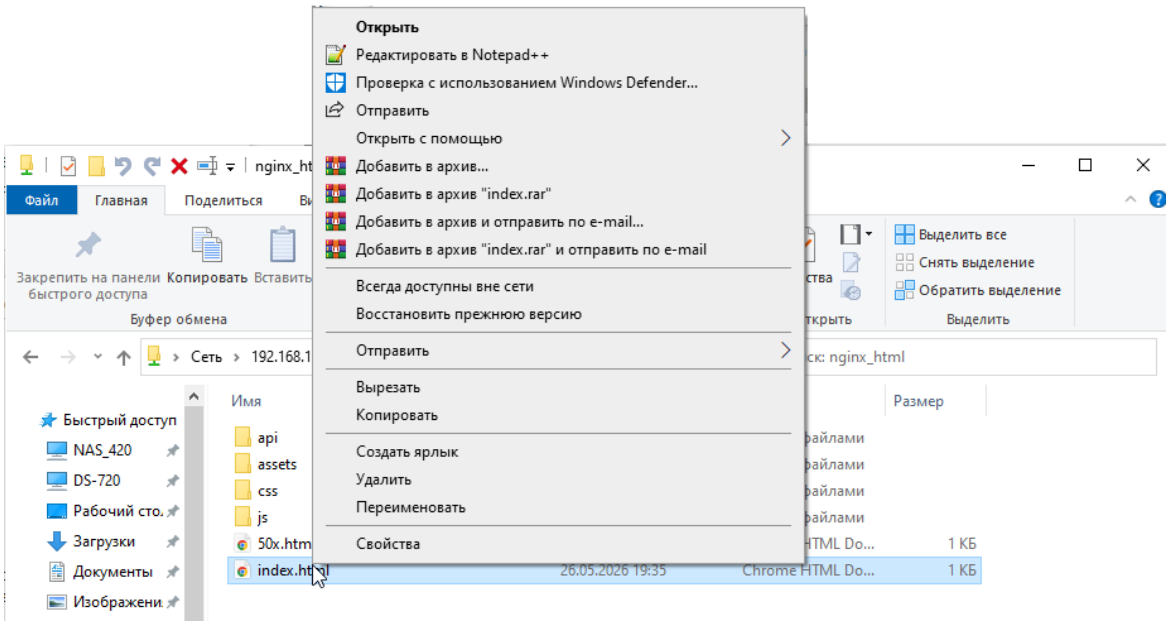
#bash

```
ls -la /usr/share/nginx/html/
```



(Права drwxrwxr-x (775) успешно применились ко всем новым директориям (api, assets, css, js) - подсвечены синим, и -rw-rw-r– к файлам они подсвечены белым. Теперь пользователи eva и системный пользователь http, и Samba имеют полный доступ.)

Прверим создание папок в Проводнике виндовс и откроем его в редакторе notepad++



Главный интерфейс панели (index.html)

Сейчас мы с вами создадим тестовое приложение для нашего сервера, которое подтвердить правильность наших действий по настройке беспарольного доступа в sudoers и отключение системной изоляции PHP-FPM, а так же настройки сервера и прав на папки и файлы.
Сейчас мы не будем разбирать html, php и javascript тестового приложения, т.к. наша главная задача собрать iso - образ и при установке с флешки на сервер, убедиться в том, что все настройки сохранились и приложение взаимодействует с сервером, а написание всего web-приложения нас ждет позже, после тестирования iso-образа.
Отредактируйте файл index.html в редакторе. Целиком замените дефолтный код файла на приведенный ниже. Он формирует окно панели управления, вкладки переключения, таблицы и скрытые модальные формы:

index.html

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Control Panel</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <div class="window">
        <header class="window-header">
            <span class="title">Control Panel</span>
            <div class="window-controls">
```

```
                <button class="win-btn">?</button>
                <button class="win-btn">–</button>
                <button class="win-btn">□</button>
                <button class="win-btn close">×</button>
            </div>
        </header>

        <div class="window-body">
            <aside class="sidebar">
                <div class="search-box">
                    <input type="text" placeholder="🔍 Search">
                </div>
                <nav class="menu">
                    <div class="menu-group">^ File Sharing</div>
                    <a href="#" class="menu-item">📁 Shared Folder</a>
                    <a href="#" class="menu-item">⇆ File Services</a>
                    <a href="#" class="menu-item active">👤 User &
Group</a>
                    <a href="#" class="menu-item">🌐 Domain/LDAP</a>
                    <div class="menu-group">^ Connectivity</div>
                    <a href="#" class="menu-item">🌍 External Access</a>
                    <a href="#" class="menu-item">🖧 Network</a>
                    <a href="#" class="menu-item">🛡🔒 Security</a>
                    <a href="#" class="menu-item">🖥 Terminal & SNMP</a>
                </nav>
            </aside>

            <main class="main-content">
                <div class="tabs">
                    <button class="tab active" id="tab-
user">User</button>
                  <button class="tab" id="tab-group">Group</button>
                    <button class="tab">Advanced</button>
                </div>

                <div class="toolbar">
                    <div class="actions">
                        <button class="btn primary" id="btn-
create">Create</button>
                        <button class="btn" id="btn-edit"
disabled>Edit</button>
                        <button class="btn" id="btn-delete"
disabled>Delete</button>
                        <button class="btn">Export ▾</button>
                        <button class="btn">Delegate ▾</button>
                    </div>
                    <div class="filter">
                        <input type="text" id="table-filter"
placeholder="∇ Filter">
                    </div>
                </div>
```

```
                <div class="table-container">
                    <table id="users-table">
                        <thead>
                            <tr>
                                <th>Name ▲</th>
                                <th>Email</th>
                                <th>Description</th>
                                <th>2FA Status</th>
                                <th>Status</th>
                            </tr>
                        </thead>
                        <tbody>
             <!-- Данные загружаются через JS -->
                        </tbody>
                    </table>
                </div>

                <footer class="table-footer">
                    <span id="items-count">0 items</span>
                    <button id="refresh-btn" class="btn">↻</button>
                </footer>
            </main>
        </div>
    </div>

  <!-- Модальное окно Создания / Редактирования -->
    <div class="modal" id="user-modal">
        <div class="modal-content">
            <h3 id="modal-title">Create User</h3>
            <form id="user-form">
                <input type="hidden" id="form-action" value="create">
                <input type="hidden" id="old-username">

                <div class="form-group">
                    <label for="username">Имя пользователя:</label>
                    <input type="text" id="username" required
pattern="^[a-z_][a-z0-9_-]*$" title="Маленькие латинские буквы и цифры">
                </div>
                <div class="form-group">
                    <label for="description">Описание (GECOS):</label>
                    <input type="text" id="description">
                </div>
                <div class="form-group" id="password-group">
                    <label for="password">Пароль:</label>
                    <input type="password" id="password">
                </div>
                <div class="form-buttons">
                    <button type="button" class="btn" id="btn-modal-
cancel">Cancel</button>
                    <button type="submit" class="btn
```

```
primary">Save</button>
                </div>
            </form>
        </div>
    </div>

 <!-- Модальное окно Групп -->
  <div class="modal" id="group-modal">
      <div class="modal-content">
          <h3>Create Group</h3>
          <form id="group-form">
              <div class="form-group">
                  <label for="group-name">Имя группы:</label>
                  <input type="text" id="group-name" required
pattern="^[a-z_][a-z0-9_-]*$">
              </div>
              <div class="form-buttons">
                  <button type="button" class="btn" id="btn-group-
cancel">Cancel</button>
                  <button type="submit" class="btn
primary">Save</button>
              </div>
          </form>
      </div>
  </div>


    <script src="js/app.js"></script>
</body>
</html>
```

Стили оформления интерфейса (css/style.css)

Создайте файл css/style.css в подпапке css/. Код задает внешний вид окна приложения, таблиц данных, кнопок управления и всплывающих модальных окон:

style.css

```
*{
    box-sizing: border-box;
    margin: 0;
    padding: 0;
    font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto,
sans-serif;
}

body {
    background-color: #f0f2f5;
```

```
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}

.window {
    width: 1000px;
    height: 550px;
    background: #fff;
    border-radius: 6px;
    box-shadow: 0 5px 25px rgba(0,0,0,0.1);
    display: flex;
    flex-direction: column;
    overflow: hidden;
    border: 1px solid #dcdcdc;
}

.window-header {
    background: #fff;
    padding: 12px 15px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    border-bottom: 1px solid #e2e8f0;
}

.window-header .title {
    font-size: 14px;
    color: #2d3748;
    font-weight: 500;
}

.window-controls .win-btn {
    border: none;
    background: none;
    padding: 4px 8px;
    cursor: pointer;
    color: #718096;
}

.window-body {
    display: flex;
    flex: 1;
    overflow: hidden;
}

/* Навигация */
.sidebar {
    width: 230px;
    background: #f7fafc;
```

```
    border-right: 1px solid #e2e8f0;
    padding: 12px;
}

.search-box input {
    width: 100%;
    padding: 6px 10px;
    border: 1px solid #cbd5e0;
    border-radius: 4px;
    margin-bottom: 15px;
}

.menu-group {
    font-size: 11px;
    text-transform: uppercase;
    color: #a0aec0;
    margin: 12px 0 6px 6px;
    font-weight: 600;
}

.menu-item {
    display: block;
    padding: 8px 12px;
    color: #4a5568;
    text-decoration: none;
    font-size: 13px;
    border-radius: 4px;
}

.menu-item.active {
    background: #ebf8ff;
    color: #2b6cb0;
    font-weight: 600;
}

/* Основная рабочая область */
.main-content {
    flex: 1;
    display: flex;
    flex-direction: column;
    padding: 0 20px;
}

.tabs {
    display: flex;
    border-bottom: 1px solid #e2e8f0;
    margin-top: 10px;
}

.tab {
    padding: 10px 20px;
```

```
    border: none;
    background: none;
    cursor: pointer;
    font-size: 14px;
    color: #718096;
}

.tab.active {
    color: #3182ce;
    border-bottom: 2px solid #3182ce;
    font-weight: 600;
}

.toolbar {
    display: flex;
    justify-content: space-between;
    margin: 15px 0;
}

.btn {
    padding: 6px 14px;
    border: 1px solid #cbd5e0;
    background: #fff;
    border-radius: 4px;
    cursor: pointer;
    font-size: 13px;
    color: #4a5568;
}

.btn:disabled {
    background: #f7fafc;
    color: #a0aec0;
    cursor: not-allowed;
    border-color: #e2e8f0;
}

.btn:hover:not(:disabled) {
    background: #f7fafc;
}

.btn.primary {
    background: #3182ce;
    color: #fff;
    border-color: #3182ce;
}

.btn.primary:hover {
    background: #2b6cb0;
}

.filter input {
```

```
    padding: 6px 10px;
    border: 1px solid #cbd5e0;
    border-radius: 4px;
    font-size: 13px;
}

/* Таблица */
.table-container {
    flex: 1;
    overflow-y: auto;
    border: 1px solid #e2e8f0;
    border-radius: 4px;
}

table {
    width: 100%;
    border-collapse: collapse;
    font-size: 13px;
}

th, td {
    padding: 10px 12px;
    text-align: left;
    border-bottom: 1px solid #edf2f7;
    user-select: none;
}

th {
    background: #f7fafc;
    color: #4a5568;
    font-weight: 600;
    position: sticky;
    top: 0;
}

tbody tr {
    cursor: pointer;
}

tbody tr:hover {
    background: #f7fafc;
}

tr.selected-user {
    background: #e8f0fe !important;
}

.status-deactivated { color: #e53e3e; }
.status-normal { color: #38a169; }

.table-footer {
```

```
    display: flex;
    justify-content: flex-end;
    align-items: center;
    padding: 12px 0;
    gap: 15px;
    font-size: 13px;
    color: #718096;
}

/* Модальные окна */
.modal {
    display: none;
    position: fixed;
    top: 0; left: 0; width: 100%; height: 100%;
    background: rgba(0,0,0,0.4);
    justify-content: center;
    align-items: center;
    z-index: 1000;
}

.modal.open {
    display: flex;
}

.modal-content {
    background: #fff;
    padding: 20px;
    border-radius: 6px;
    width: 400px;
    box-shadow: 0 4px 15px rgba(0,0,0,0.2);
}

.modal-content h3 {
    margin-bottom: 15px;
    color: #2d3748;
}

.form-group {
    margin-bottom: 12px;
}

.form-group label {
    display: block;
    font-size: 12px;
    color: #4a5568;
    margin-bottom: 4px;
}

.form-group input {
    width: 100%;
    padding: 8px;
```

```
    border: 1px solid #cbd5e0;
    border-radius: 4px;
}

.form-buttons {
    display: flex;
    justify-content: flex-end;
    gap: 10px;
    margin-top: 18px;
}
```

Шаг 3.3. Серверный обработчик пользователей (api/users.php)

Создайте файл `api/users.php` в подпапке `api/`. Скрипт обрабатывает GET-запросы для вывода учетных записей (исключая технического nobody) и POST-запросы для выполнения атомарных операций через `systemd-run` в обход изоляции:

users.php

```
<?php
header('Content-Type: application/json; charset=utf-8');

// Обработка получения списка (GET)
if ($_SERVER['REQUEST_METHOD'] === 'GET') {
    $usersList = [];

    if (is_readable('/etc/passwd')) {
        $lines = file('/etc/passwd', FILE_IGNORE_NEW_LINES |
FILE_SKIP_EMPTY_LINES);
        foreach ($lines as $line) {
            $parts = explode(':', $line);
            if (count($parts) >= 5) {
                $username = $parts[0];
                $uid = (int)$parts[2];
                $description = $parts[4];

                // Выводим root (UID 0), системных и обычных пользователей
(UID >= 1000)
                if (($uid === 0 || $uid >= 1000) && $username !==
'nobody' || $username === 'guest' || $username === 'admin') {


                    // Сопоставление статуса активности учетной записи
                    $status = 'Normal';
                    if ($username === 'guest') {
                        $status = 'Deactivated';
                    }
```

```
                    $usersList[] = [
                        'name' => $username,
                        'email' => '',
                        'desc' => $description,
                        'tfa' => 'Disabled',
                        'status' => $status
                    ];
                }
            }
        }
    }
    echo json_encode($usersList, JSON_UNESCAPED_UNICODE);
    exit;
}

// Обработка действий создания/изменения/удаления (POST)
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $input = json_decode(file_get_contents('php://input'), true);

    if (!isset($input['action'])) {
        echo json_encode(['success' => false, 'error' => 'Отсутствует
действие']);
        exit;
    }

    $action = $input['action'];
    $username = preg_replace('/[^a-z0-9_-]/', '', $input['username'] ??
'');
    $description = escapeshellarg($input['description'] ?? '');
    $password = $input['password'] ?? '';

    if (empty($username)) {
        echo json_encode(['success' => false, 'error' => 'Некорректное
имя пользователя']);
        exit;
    }

    switch ($action) {
        case 'create':
            if (empty($username)) {
                echo json_encode(['success' => false, 'error' =>
'Некорректное имя пользователя']);
                exit;
            }

            // Генерируем хэш пароля
            $salt = '$1$' . substr(md5(uniqid(rand(), true)), 0, 8) .
'$';
            $hashed_password = crypt($password, $salt);
            $uid_gid = rand(1100, 1900);
```

```
            $passwd_line =
"{$username}:x:{$uid_gid}:{$uid_gid}:{$description}:/home/{$username}:/
bin/bash";
            $shadow_line =
"{$username}:{$hashed_password}:19500:0:99999:7:::";
            $group_line = "{$username}:x:{$uid_gid}:";

            // Собираем все команды в одну строку для запуска через bash
            $system_cmd = "echo '{$passwd_line}' >> /etc/passwd && " .
                          "echo '{$shadow_line}' >> /etc/shadow && " .
                          "echo '{$group_line}' >> /etc/group && " .
                          "mkdir -p /home/{$username} && " .
                          "chown -R {$uid_gid}:{$uid_gid}
/home/{$username}";

            // Вызываем встроенную службу systemd-run. Флаг -G заставляет её
отработать от root наружу.
            $cmd = "sudo /usr/bin/systemd-run -G /usr/bin/bash -c " .
escapeshellarg($system_cmd) . " 2>&1";

            exec($cmd, $output, $return_var);

            if ($return_var !== 0) {
                $err = implode(' ', $output);
                echo json_encode(['success' => false, 'error' =>
"Ошибка запуска: {$err} (Код: {$return_var})"]);
                exit;
            }

            echo json_encode(['success' => true]);
            break;


        case 'update':
            // Читаем параметры из JSON-запроса от браузера
            $old_username = preg_replace('/[^a-z0-9_-]/', '',
$input['old_username'] ?? '');
            $new_description = $input['description'] ?? '';
            $new_password = $input['password'] ?? '';

            if (empty($old_username)) {
                echo json_encode(['success' => false, 'error' => 'Не
указан пользователь для редактирования']);
                exit;
            }

            // Экранируем описание, чтобы оно не сломало синтаксис команды sed
            $clean_desc = str_replace('/', '\/', $new_description);

            // 1. Команда для обновления описания (GECOS) в /etc/passwd
            $update_cmd = "sed -i -E
```

```
's/^({$old_username}:[^:]*:[^:]*:[^:]*):[^:]*(:.*)/\\1:{$clean_desc}\\2
/' /etc/passwd";

            // 2. Если в форму ввели новый пароль — добавляем команду обновления
хэша в /etc/shadow
            if (!empty($new_password)) {
                $salt = '$1$' . substr(md5(uniqid(rand(), true)), 0, 8)
. '$';
                $hashed_password = crypt($new_password, $salt);
                $clean_hash = str_replace('/', '\/', $hashed_password);

                $update_cmd .= " && sed -i -E
's/^({$old_username}:)[^:]*(:.*)/\\1{$clean_hash}\\2/' /etc/shadow";
            }

            // Вызываем итоговую команду через systemd-run наружу от root
            $cmd = "sudo /usr/bin/systemd-run -G /usr/bin/bash -c " .
escapeshellarg($update_cmd) . " 2>&1";

            exec($cmd, $output, $return_var);

            if ($return_var === 0) {
                echo json_encode(['success' => true]);
            } else {
                $err = implode(' ', $output);
                echo json_encode(['success' => false, 'error' =>
"Ошибка обновления: {$err}"]);
            }
            break;



        case 'delete':
            if ($username === 'root') {
                echo json_encode(['success' => false, 'error' =>
'Удаление root запрещено']);
                exit;
            }

            // Формируем команды для полной очистки записей из passwd,
shadow, group и удаления папки
            $delete_cmd = "sed -i '/^{$username}:/d' /etc/passwd && " .
                          "sed -i '/^{$username}:/d' /etc/shadow && " .
                          "sed -i '/^{$username}:/d' /etc/group && " .
                          "rm -rf /home/{$username}";

            // Вызываем команду через systemd-run наружу от имени root
            $cmd = "sudo /usr/bin/systemd-run -G /usr/bin/bash -c " .
escapeshellarg($delete_cmd) . " 2>&1";

            exec($cmd, $output, $return_var);
```

```
            if ($return_var === 0) {
                echo json_encode(['success' => true]);
            } else {
                $err = implode(' ', $output);
                echo json_encode(['success' => false, 'error' =>
"Ошибка удаления: {$err}"]);
            }
            break;


        default:
            echo json_encode(['success' => false, 'error' => 'Неизвестная
операция']);
            break;
    }
    exit;
}
```

Шаг 3.4. Серверный обработчик групп (api/groups.php)

Создайте файл `api/groups.php` в подпапке `api/`. Скрипт парсит системный файл `/etc/group`, выстраивает связи участников и нативно создаёт/удаляет группы в ОС через `systemd-run`:

groups.php

```
<?php
header('Content-Type: application/json; charset=utf-8');

$input = json_decode(file_get_contents('php://input'), true);
$action = $input['action'] ?? $_SERVER['REQUEST_METHOD'];

// --- ОБРАБОТКА ПОЛУЧЕНИЯ СПИСКА ГРУПП (GET) ---
if ($action === 'GET') {
    $groupsList = [];

    if (is_readable('/etc/group')) {
        $lines = file('/etc/group', FILE_IGNORE_NEW_LINES |
FILE_SKIP_EMPTY_LINES);
        foreach ($lines as $line) {
            // Формат /etc/group: имя_группы:пароль:GID:список_пользователей
            $parts = explode(':', $line);
            if (count($parts) >= 3) {
                $group_name = $parts[0];
                $gid = (int)$parts[2];
                $users = $parts[3] ?? ''; // Пользователи через запятую
```

```
                // Фильтруем системные группы, оставляем root (GID 0), wheel
и кастомные (GID >= 1000)
                if ($gid === 0 || $gid === 998 || $gid >= 1000) {
                    // Исключаем технического nobody
                    if ($group_name !== 'nobody') {
                        $groupsList[] = [
                            'name' => $group_name,
                            'gid' => $gid,
                            'users' => empty($users) ? '—' :
str_replace(',', ', ', $users),
                            'status' => 'Normal'
                        ];
                    }
                }
            }
        }
    }
    echo json_encode($groupsList, JSON_UNESCAPED_UNICODE);
    exit;
}

// --- ОБРАБОТКА ИЗМЕНЕНИЙ (POST) ---
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $group_name = preg_replace('/[^a-z0-9_-]/', '',
$input['group_name'] ?? '');

    if (empty($group_name)) {
        echo json_encode(['success' => false, 'error' => 'Некорректное
имя группы']);
        exit;
    }

    switch ($input['action']) {
        case 'create':
            // Создание группы через systemd-run наружу от root
            $cmd = "sudo /usr/bin/systemd-run -G /usr/bin/bash -c " .
escapeshellarg("groupadd {$group_name}") . " 2>&1";
            exec($cmd, $output, $return_var);

            if ($return_var === 0) {
                echo json_encode(['success' => true]);
            } else {
                $err = implode(' ', $output);
                echo json_encode(['success' => false, 'error' =>
"Ошибка создания группы: {$err}"]);
            }
            break;

        case 'delete':
            if ($group_name === 'root' || $group_name === 'wheel') {
                echo json_encode(['success' => false, 'error' =>
```

```
'Удаление системных групп запрещено']);
                exit;
            }
            // Удаление группы
            $cmd = "sudo /usr/bin/systemd-run -G /usr/bin/bash -c " .
escapeshellarg("groupdel {$group_name}") . " 2>&1";
            exec($cmd, $output, $return_var);

            if ($return_var === 0) {
                echo json_encode(['success' => true]);
            } else {
                $err = implode(' ', $output);
                echo json_encode(['success' => false, 'error' =>
"Ошибка удаления группы: {$err}"]);
            }
            break;

        default:
            echo json_encode(['success' => false, 'error' => 'Неизвестная
операция']);
            break;
    }
    exit;
}
```

Клиентские скрипты логики (js/app.js)

Создайте файл `js/app.js` в подпапке `js/`. Скрипт управляет асинхронным обновлением таблиц (fetch), переключением контекста вкладок, фильтрацией на лету и валидацией полей ввода:

app.js

```
document.addEventListener('DOMContentLoaded', () => {
    const tableBody = document.querySelector('#users-table tbody');
    const itemsCount = document.getElementById('items-count');
    const refreshBtn = document.getElementById('refresh-btn');
    const filterInput = document.getElementById('table-filter');

    // Кнопки управления
    const btnCreate = document.getElementById('btn-create');
    const btnEdit = document.getElementById('btn-edit');
    const btnDelete = document.getElementById('btn-delete');

    // Элементы модального окна
    const userModal = document.getElementById('user-modal');
    const userForm = document.getElementById('user-form');
    const modalTitle = document.getElementById('modal-title');
```

```
    const btnModalCancel = document.getElementById('btn-modal-cancel');

    let selectedUsername = null;
    let selectedUserRow = null;

    // Загрузка данных
    async function loadUsers() {
        try {
            const response = await fetch('api/users.php');
            if (!response.ok) throw new Error('Ошибка сервера');
            const data = await response.json();
            renderTable(data);
            resetSelection();
        } catch (error) {
            alert('Не удалось обновить список пользователей');
        }
    }

    function renderTable(users) {
        tableBody.innerHTML = '';
        users.forEach(user => {
            const tr = document.createElement('tr');
            tr.dataset.username = user.name;
            tr.dataset.desc = user.desc;

            const statusClass = user.status === 'Normal' ? 'status-
normal' : 'status-deactivated';

            tr.innerHTML = `
                <td><b>${escapeHtml(user.name)}</b></td>
                <td>${escapeHtml(user.email)}</td>
                <td>${escapeHtml(user.desc)}</td>
                <td>${escapeHtml(user.tfa)}</td>
                <td
class="${statusClass}">${escapeHtml(user.status)}</td>
            `;

            // Логика выбора строки кликом
            tr.addEventListener('click', () => {
                if (selectedUserRow)
selectedUserRow.classList.remove('selected-user');

                if (selectedUsername === user.name) {
                    resetSelection();
                } else {
                    selectedUsername = user.name;
                    selectedUserRow = tr;
                    tr.classList.add('selected-user');
                    btnEdit.disabled = false;
                    // Запрещаем удалять root-пользователя напрямую из UI
ради безопасности
```

```
                    btnDelete.disabled = (user.name === 'root');
                }
            });

            tableBody.appendChild(tr);
        });
        itemsCount.textContent = `${users.length} items`;
    }

    function resetSelection() {
        selectedUsername = null;
        selectedUserRow = null;
        btnEdit.disabled = true;
        btnDelete.disabled = true;
    }

    // Фильтрация таблицы
    filterInput.addEventListener('input', (e) => {
        const value = e.target.value.toLowerCase();
        Array.from(tableBody.querySelectorAll('tr')).forEach(tr => {
            const match = tr.textContent.toLowerCase().includes(value);
            tr.style.display = match ? '' : 'none';
        });
    });

    // Открытие модального окна создания
    btnCreate.addEventListener('click', () => {
        userForm.reset();
        document.getElementById('form-action').value = 'create';
        document.getElementById('username').disabled = false;
        document.getElementById('password-group').style.display =
'block';
        modalTitle.textContent = 'Create User';
        userModal.classList.add('open');
    });

    // Открытие модального окна редактирования
    btnEdit.addEventListener('click', () => {
        if (!selectedUsername) return;
        userForm.reset();
        document.getElementById('form-action').value = 'update';
        document.getElementById('old-username').value =
selectedUsername;

        const usernameInput = document.getElementById('username');
        usernameInput.value = selectedUsername;
        usernameInput.disabled = true; // Имя пользователя в Linux менять
через useradd напрямую нельзя

        document.getElementById('description').value =
selectedUserRow.dataset.desc || '';
```

```
        document.getElementById('password-group').style.display =
'block'; // Пароль заполнять по желанию

        modalTitle.textContent = 'Edit User';
        userModal.classList.add('open');
    });

    // Обработка кнопки удаления
    btnDelete.addEventListener('click', async () => {
        if (!selectedUsername) return;
        if (confirm(`Вы уверены, что хотите удалить пользователя
${selectedUsername} вместе с домашней директорией?`)) {
            await sendAction({ action: 'delete', username:
selectedUsername });
        }
    });

    // Закрытие модального окна
    btnModalCancel.addEventListener('click', () =>
userModal.classList.remove('open'));

    // Отправка формы (Создание / Изменение)
    userForm.addEventListener('submit', async (e) => {
        e.preventDefault();
        const action = document.getElementById('form-action').value;

        const payload = {
            action: action,
            username: document.getElementById('username').value,
            description: document.getElementById('description').value,
            password: document.getElementById('password').value,
            old_username: document.getElementById('old-username').value
        };

        await sendAction(payload);
        userModal.classList.remove('open');
    });

    async function sendAction(data) {
        try {
            const response = await fetch('api/users.php', {
                method: 'POST',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify(data)
            });
            const result = await response.json();
            if (result.success) {
                loadUsers();
            } else {
                alert('Ошибка:' + result.error);
            }
```

```
        } catch (error) {
            alert('Ошибка сети при отправке запроса');
        }
    }

    function escapeHtml(text) {
        if (!text) return '';
        return text.toString().replace(/&/g, "&amp;").replace(/</g,
"&lt;").replace(/>/g, "&gt;");
    }

    refreshBtn.addEventListener('click', loadUsers);
    loadUsers();

    // --- ЛОГИКА ВКЛАДКИ GROUP ---
    const tabUser = document.getElementById('tab-user');
    const tabGroup = document.getElementById('tab-group');
    const tableHeader = document.querySelector('#users-table thead
tr');

    const groupModal = document.getElementById('group-modal');
    const groupForm = document.getElementById('group-form');
    const btnGroupCancel = document.getElementById('btn-group-cancel');

    let currentTab = 'user'; // Храним активную вкладку ('user' или
'group')
    let selectedGroupName = null;

    // Переключение на вкладку User
    tabUser.addEventListener('click', () => {
        tabGroup.classList.remove('active');
        tabUser.classList.add('active');
        currentTab = 'user';
        tableHeader.innerHTML = `
            <th>Name ▲</th>
            <th>Email</th>
            <th>Description</th>
            <th>2FA Status</th>
            <th>Status</th>
        `;
        resetSelection();
        loadUsers(); // Вызываем старую функцию загрузки пользователей
    });

    // Переключение на вкладку Group
    tabGroup.addEventListener('click', () => {
        tabUser.classList.remove('active');
        tabGroup.classList.add('active');
        currentTab = 'group';
        tableHeader.innerHTML = `
            <th>Group Name ▲</th>
```

```
            <th>GID</th>
            <th>Members (Users)</th>
            <th>Status</th>
        `;
        resetSelection();
        loadGroups();
    });

    // Загрузка групп с бэкенда
    async function loadGroups() {
        try {
            const response = await fetch('api/groups.php');
            const groups = await response.json();
            renderGroupsTable(groups);
        } catch (error) {
            alert('Ошибка загрузки групп');
        }
    }

    function renderGroupsTable(groups) {
        tableBody.innerHTML = '';
        groups.forEach(group => {
            const tr = document.createElement('tr');
            tr.innerHTML = `
                <td><b>${escapeHtml(group.name)}</b></td>
                <td>${group.gid}</td>
                <td>${escapeHtml(group.users)}</td>
                <td class="status-normal">${group.status}</td>
            `;

            tr.addEventListener('click', () => {
                if (selectedUserRow)
selectedUserRow.classList.remove('selected-user');

                if (selectedGroupName === group.name) {
                    selectedGroupName = null;
                    selectedUserRow = null;
                    btnDelete.disabled = true;
                } else {
                    selectedGroupName = group.name;
                    selectedUserRow = tr;
                    tr.classList.add('selected-user');
                    btnEdit.disabled = true; // Для групп редактирование
отключим
                    btnDelete.disabled = (group.name === 'root' ||
group.name === 'wheel');
                }
            });
            tableBody.appendChild(tr);
        });
        itemsCount.textContent = `${groups.length} items`;
```

```
    }

    // Модифицируем общие кнопки под контекст активной вкладки
    btnCreate.addEventListener('click', (e) => {
        if (currentTab === 'group') {
            e.stopPropagation(); // Останавливаем открытие модалки
пользователей
            groupForm.reset();
            groupModal.classList.add('open');
        }
    });

    btnDelete.addEventListener('click', async () => {
        if (currentTab === 'group' && selectedGroupName) {
            if (confirm(`Удалить группу ${selectedGroupName}?`)) {
                await sendGroupAction({ action: 'delete', group_name:
selectedGroupName });
            }
        }
    });

    btnGroupCancel.addEventListener('click', () =>
groupModal.classList.remove('open'));

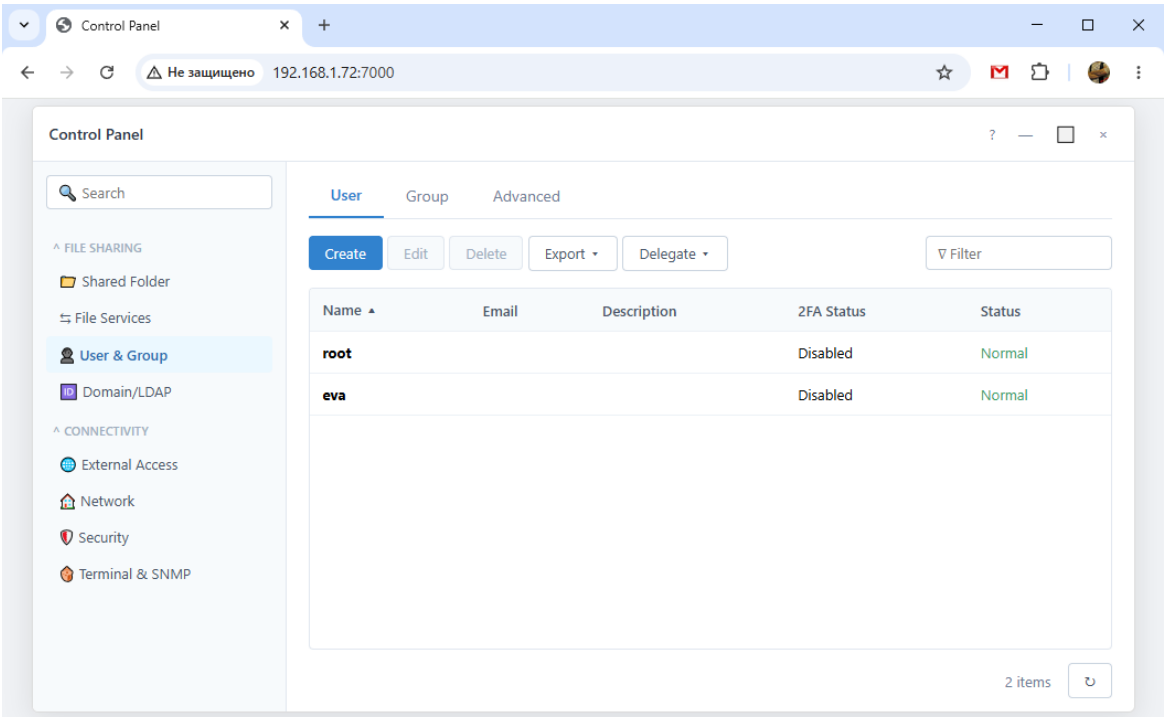
    groupForm.addEventListener('submit', async (e) => {
        e.preventDefault();
        await sendGroupAction({
            action: 'create',
            group_name: document.getElementById('group-name').value
        });
        groupModal.classList.remove('open');
    });

    async function sendGroupAction(data) {
        try {
            const response = await fetch('api/groups.php', {
                method: 'POST',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify(data)
            });
            const result = await response.json();
            if (result.success) {
                loadGroups();
            } else {
                alert('Ошибка:' + result.error);
            }
        } catch (error) {
            alert('Ошибка сети при обработке группы');
        }
    }
```

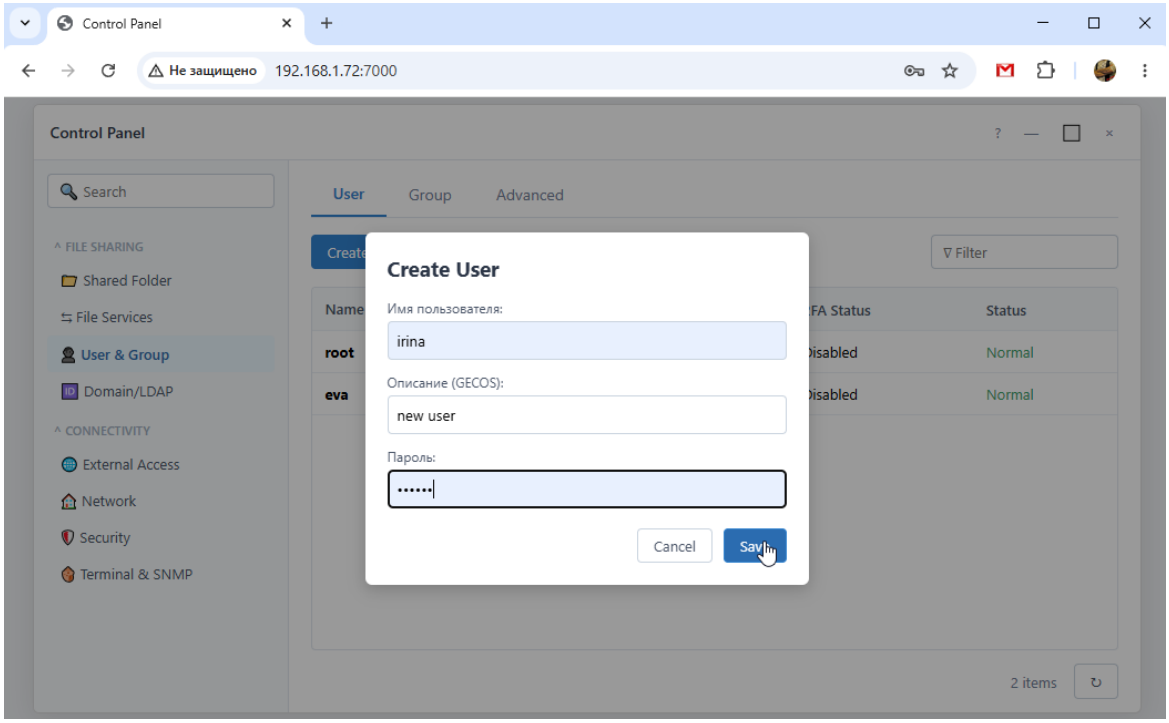
```
    // Модифицируем круглую кнопку обновления (↻)
    refreshBtn.addEventListener('click', () => {
        if (currentTab === 'group') loadGroups();
    });


});
```

Сохраним файлы и проверим в окне браузера, перейдя по ссылке http://192.168.1.72:7000/



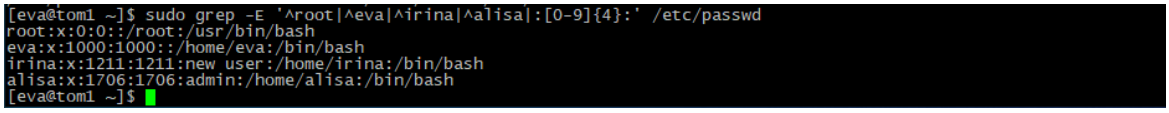
мы видим вывод в таблице наших пользователей root и eva, проверим работу web-страницы на предмет добавления пользователя irina с описанием new user и alisa / admin



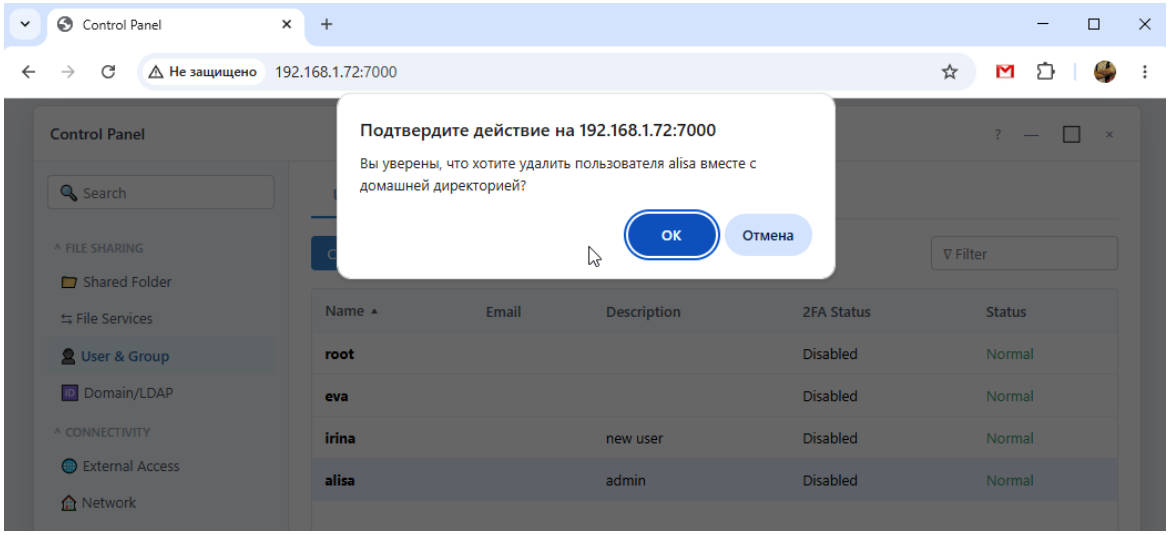
Проверим новых пользователей в консоли

#bash

```
sudo grep -E '^root|^eva|^irina|^alisa|:[0-9]{4}:' /etc/passwd
```



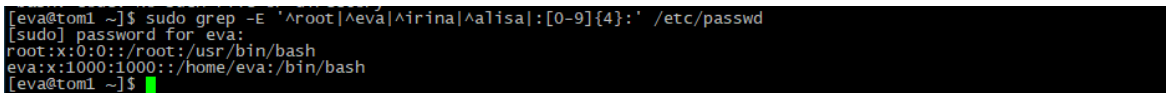
Внимание! Пользователей root и eva не удаляем, иначе консоль отключиться и мы с вами больше не попадем в управление сервером, а без пользователей с правами суперпользователей система нас просто не пустит.
Удалим строго только новых пользователей в веб-приложении.



И снова проверим консоль

#bash

```
sudo grep -E '^root|^eva|^irina|^alisa|:[0-9]{4}:' /etc/passwd
```



!!!!!!!!!!! Разработка Веб-Фронтенда !!!!!!!!!!!

!!!!!! Убрать перед сборкой iso !!!!!!!! Проверить версии linux Драйверы сетевой карты реального сервера Изменить временно на 192.168.1.150 Стереть sfid

запись исо Изменить постоянно на 192.168.1.72 Вернуть sfid

!!!!!!!!!!! переходим к iso !!!!!!!!!!!

From:
https://wwoss.ru/ - **worldwide open-source software**

Permanent link:
https://wwoss.ru/doku.php?id=tmp_26.05.26_frontend

Last update: **2026/05/26 22:10**