# Руководство по установке и настройке ПО парковки

## 1. Установка зависимостей

```
sudo apt install curl git
curl -fsSL https://get.docker.com -o get-docker.sh
sudo chmod +x get-docker.sh
sudo ./get-docker.sh
sudo systemctl start docker
```

## 1.1. Установка драйверов для видеокарты NVIDIA

```
sudo apt install nvidia-smi

curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | sudo gpg
--dearmor -o /usr/share/keyrings/nvidia-container-toolkit-keyring.gpg \
  && curl -s -L https://nvidia.github.io/libnvidia-
container/stable/deb/nvidia-container-toolkit.list | \
    sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-
container-toolkit-keyring.gpg] https://#g' | \
    sudo tee /etc/apt/sources.list.d/nvidia-container-toolkit.list

sed -i -e '/experimental/ s/^#//g' /etc/apt/sources.list.d/nvidia-
container-toolkit.list

sudo apt-get update

export NVIDIA_CONTAINER_TOOLKIT_VERSION=1.17.8-1
  sudo apt-get install -y \
      nvidia-container-toolkit=${NVIDIA_CONTAINER_TOOLKIT_VERSION} \
      nvidia-container-toolkit-base=${NVIDIA_CONTAINER_TOOLKIT_VERSION} \
      libnvidia-container-tools=${NVIDIA_CONTAINER_TOOLKIT_VERSION} \
      libnvidia-container1=${NVIDIA_CONTAINER_TOOLKIT_VERSION}
```

Проверьте вызов команды "nvidia-smi" внутри тестового контейнера:

```
docker run --rm --gpus all nvidia/cuda:12.2.0-base-ubuntu20.04 nvidia-smi
```

Вывод должен быть похожим на следующее:

```
+-----------------------------------------------------------------------------
--------------+
| NVIDIA-SMI 535.247.01             Driver Version: 535.247.01   CUDA
Version: 12.2     |
|-----------------------------------------+----------------------+--------
--------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A |
Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-
Util  Compute M. |
|                                         |                      |
MIG M. |
|=========================================+======================+========
==============|
|   0  NVIDIA GeForce RTX 3050        Off | 00000000:01:00.0 Off |
N/A |
| 35%   48C    P2              N/A / 115W |   4444MiB /  8192MiB |      5%
Default |
|                                         |                      |
N/A |
+-----------------------------------------+----------------------+--------
--------------+
```

## 2. Запуск Backend части

Клонируем репозиторий и запускаем

```
cd parking-api-django
docker compose build
docker compose up -d
```

## 2.1. Создание superuser в Backend части

Перед созданием парковки необходимо создать пользователя, который будет являться главным администратором парковки. Для создания пользователя необходимо зайти в bash контейнера backend.

```
cd parking-api-django
docker compose exec backend sh
```

Далее вводим команду для создания суперпользователя

```
python manage.py createsuperuser
```

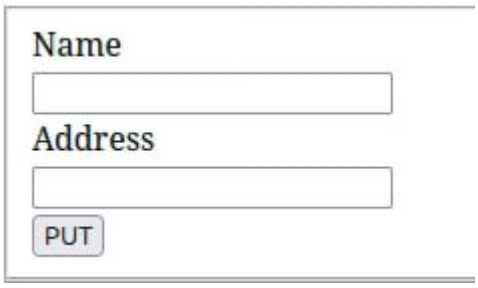
Будут выведены следующие выходные данные. Введите требуемое имя пользователя, электронную почту и пароль (этап с почтой можно пропустить):

```
Username (leave blank to use 'admin'): admin
Email address: admin@admin.com
Password: ********
Password (again): ********
Superuser created successfully.
```

Далее необходимо перейти в браузере по адресу Backend "localhost:8001/api/accounts/login" и ввести данные от созданного вами учётной записи


После чего вы получите 200 код, который говорит о успешной авторизации


Далее перейдите в /api/parking/ и создайте парковку с желаемым названием и адресом

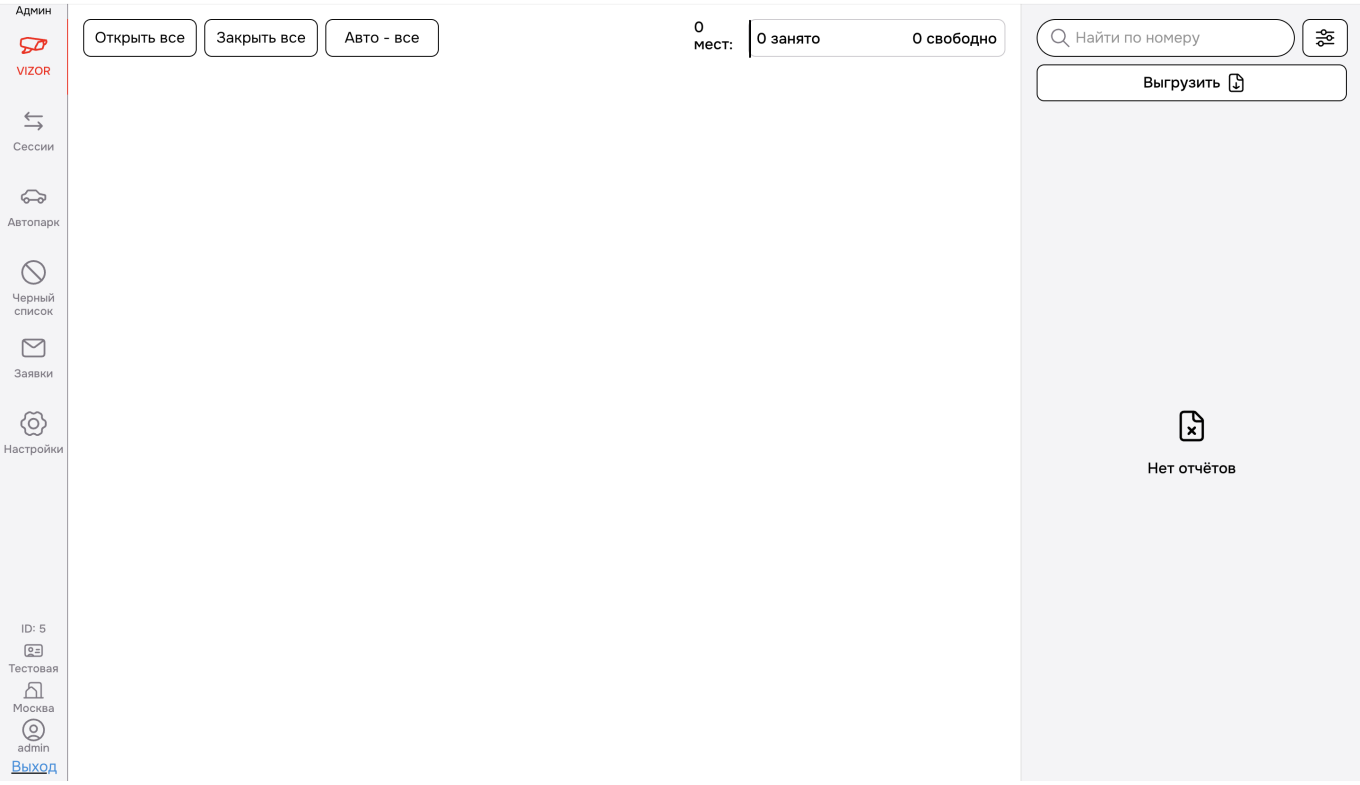



## 3. Запуск Frontend части

Клонируем репозиторий и запускаем

```
cd parking-ui-new
docker compose build
docker compose up -d
```

После чего перейдите на Frontend по адресу localhost:8080 и введите данные от созданной учётной записи




## 4. Конфигурация Nginx и certbot для домена

```
sudo apt install nginx
sudo apt install snapd -y
sudo snap install --classic certbot
sudo certbot --nginx -d ваш-домен.ru
```

Добавьте следующее для прокси в /etc/nginx/sites-enabled:

```
location /welcome {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ =404;
}

location /api/wsEvents {
    proxy_pass http://localhost:8002/api/wsEvents;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
}

location /api/wsLedMessages {
    proxy_pass http://localhost:8002/api/wsLedMessages;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
}


location / {
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_pass http://localhost:8080;
}

location /api {
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_pass http://localhost:8001/api;
}
```