Системные требования сервера(ов)

Системные требования:

- Операционная система: Linux с поддержкой русского языка и версией ядра не ниже 4.15. Предпочтительная версия – Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-128-generic x86_64).
- CPU – 4 потока или более.
- ОЗУ – 8 Гб или более.
- HDD – 40 Гб или более.

Установка и настройка базы данных

Примечание: перед распаковкой архива с базой данных убедиться, что на сервере установлено *postgresql 10* и на нем создан пользователь *mobnius* с паролем *mobnius-0*

```
sudo apt update
sudo apt install postgresql postgresql-contrib
sudo apt install postgresql-plv8
sudo timedatectl set-timezone Europe/Moscow
sudo apt install nodejs npm
```

Восстанавливаем базу данных из архива *db.bak*

```
sudo su - postgres
psql -c 'create database "cic-release-db";'
psql -c 'ALTER DATABASE "cic-release-db" OWNER TO mobnius;'
psql -c 'ALTER DATABASE "cic-release-db" SET timezone TO "Europe/Moscow";'
psql -U postgres cic-release-db < db.bak
```

Установка и настройка web-сервера

Устанавливаем дополнительное ПО:

```
sudo apt install nodejs npm
sudo apt install nginx
```

В каталоге /var/www создаем bash-скрипт, который устанавливает/обновляет серверное приложение:

```
sudo chown -R www-data:www-data /var/www
sudo chmod -R 775 /var/www
cd /var/www
touch release-deploy.sh # где release - код проекта
sudo chmod -R 775 /var/www/release-deploy.sh
```

Далее вставляем в скрипт *release-deploy.sh* следующий текст:

```
#!/bin/bash

#переходим в корневой каталог
```

```
cd /var/www

DIR1="release-node"
FILE1="deploy.zip"

echo "clear data..."

if [ -f "$FILE1" ]; then
   if [ -d "$DIR1" ]; then
   find $DIR1 -delete
   fi

   unzip $FILE1 -d $DIR1
   cd $DIR1
   echo "change setting to production"
   touch production.js
   echo " var fs = require('fs');
   var join = require('path').join;
   var file = join(__dirname, 'configs', 'app.json'); var str = fs.readFileSync(file).toString(); var data =
/\"reference\":\s\"\w*\",/gm.exec(str); if (data.length > 0) { str = str.replace(data[0], '\"reference\":
\"production\",'); } fs.writeFileSync(file, str);" >> production.js
   node production.js
   rm -f production.js
   cd ..
   rm -f $FILE1
   sudo chown -R www-data:www-data $DIR1
   sudo chmod -R 775 $DIR1
   echo "finish deploy"
fi

sudo systemctl restart $DIR1
echo "finish"
```

Сохраняем информацию (выполнять не нужно) и переходим к настройке службы.

Настройка службы

Переходим в каталог /lib/systemd/system, предназначенный для хранения собственных нужд, выполнив команду:

```
cd /lib/systemd/system
```

Создаем службу для серверного приложения:

```
touch release-node.service
```

Далее добавляем следующий код:

```
[Unit]

Description=Описание нашей службы

After=network-online.target
```

[Service]

User=www-data

Restart=on-failure

 # тут указываем путь к серверному приложению

WorkingDirectory=/var/www/release-node

ExecStart=/usr/bin/node /var/www/release-node/bin/www port=3000 debug virtual_dir_path="/release" connection_string="host:127.0.0.1;port:5432;user:user-name;password:mobnius-0;database:cic-release-db"

[Install]

WantedBy=multi-user.target

Примечание: порт приложения может быть любым, но принято для node начинать с 3000. Параметр debug нужен в момент тестирования, на release можно убрать.

Сохраняем содержимое файла и выполняем следующие команды:

sudo systemctl daemon-reload // необходимо выполнять при внесении изменений в службы
sudo systemctl enable release-node.service // добавление в автозапуск

Далее переходим в каталог на сервере /var/www и выполняем команду:

cd /var/www
./release-deploy.sh

Настройка Nginx

Далее настраиваем Nginx на стандартном порту 80:

cd /etc/nginx/sites-enabled

Открываем файл default и добавляем location:

```
location /release {
   proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
   proxy_set_header Host $host;
   proxy_pass http://localhost:3000;

   # enable WebSockets
   proxy_http_version 1.1;
   proxy_set_header Upgrade $http_upgrade;
   proxy_set_header Connection "upgrade";
}
```

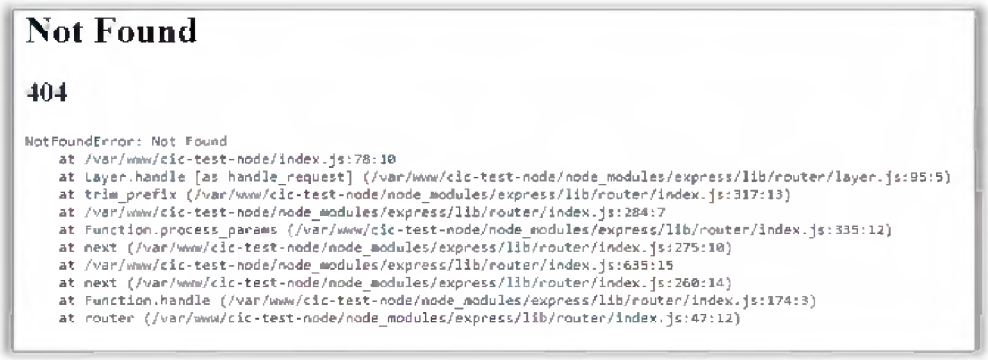
Далее перезапускаем Nginx, выполнив команду:

sudo systemctl restart nginx

Для проверки работоспособности сервиса выполняем в браузере следующий запрос:

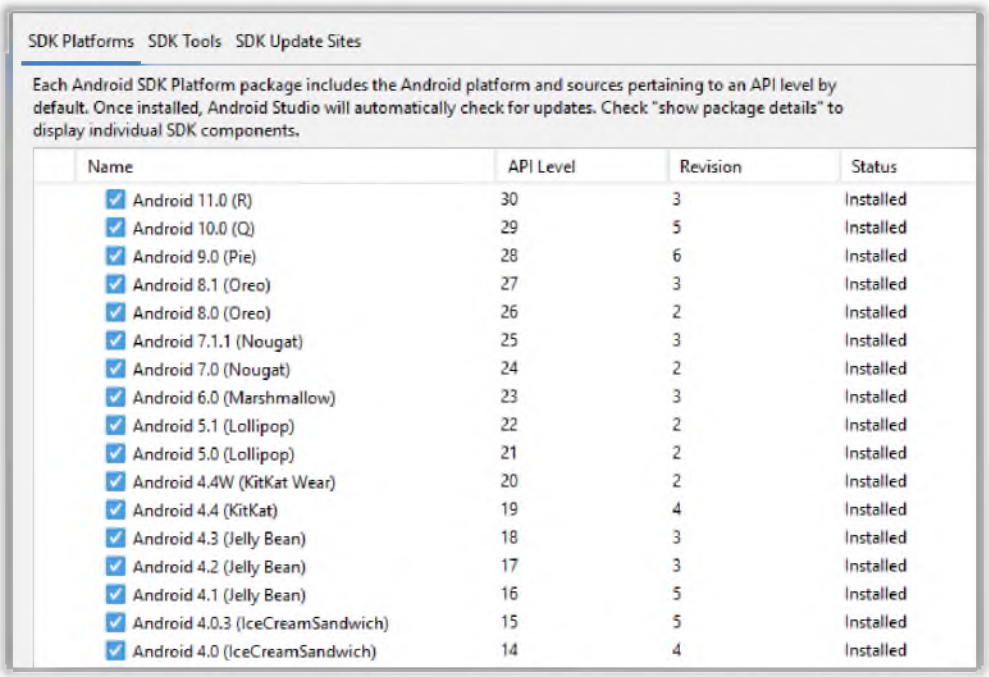
http://[IP or domain]/release

Если настройка выполнена верна, отобразиться данная информация:



Сборка APK

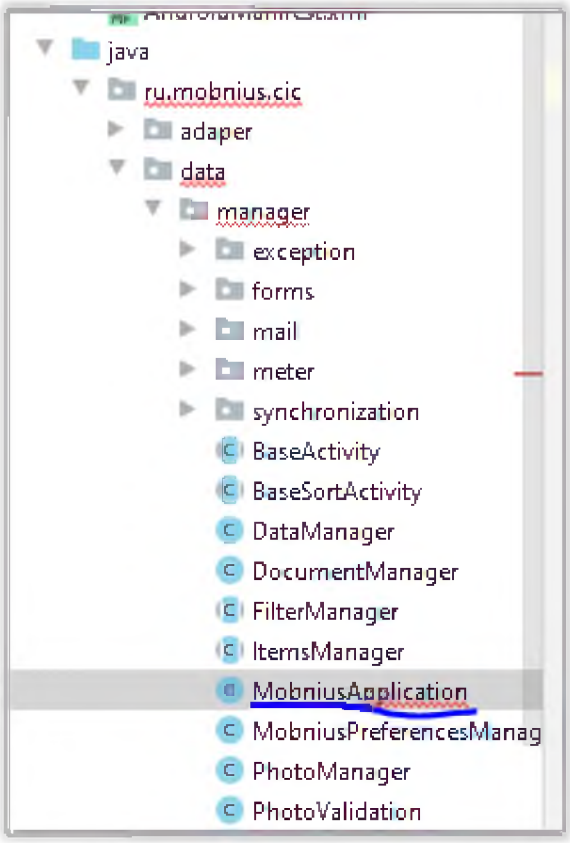
Устанавливаем Android Studio и ставим все API с Level 14



Далее открываем проект, который храниться в архиве .zip.

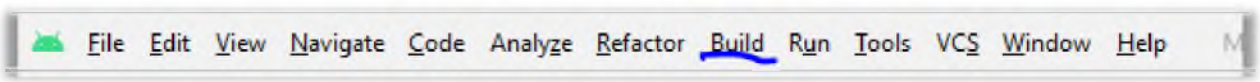
Открываем файл /res/xml/network_security_config.xml и редактируем IP на требуемый (адрес сервера, на котором расположен nodejs - сервис).

Далее открываем файл MobniusApplication.java



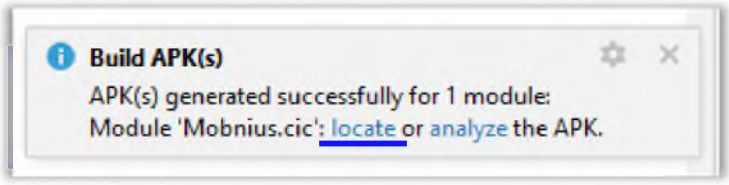
И заменяет IP адрес на указанный выше.

И запускаем build



«Build -> Bundle(s)/APK(s) -> Build APK(s)»

Дожидаемся завершения и устанавливаем APK на телефон Android.



Инструкция по разворачиванию APM

Доменное имя: https://example.com
[example] - это наименование конфига в APM, желательно по доменному имени для ясности.
http://cic.it-serv.ru/ - cic.it-serv.ru

/[example-app-dir] – это виртуальная директория приложения.
http://cic.it-serv.ru/arm/dev/#/main - /arm/dev

/[example-rpc-dir] – это виртуальная директория бэкенда-rpc.
http://cic.it-serv.ru/dev/rpc - /dev/rpc

[tile-url] – это url тайлов OSM (openstreetmap), например
https://tile.openstreetmap.org/8/153/74.png - tile.openstreetmap.org

1. Склонировать проект
- git clone [url проекта]

2. Установить пакеты
- выполнить команду npm install

3. Добавить в src/config отдельный конфиг
- создать файл src/config/[example].json

Содержимое файла:

```
"BASE_URL": "https://[example]/[example-rpc-dir]",
"WS_URL": "https://[example]",
"WS_PATH": "/[example-rpc-dir]/socket.io",
"map": {
  "center": [
    51.730846, 36.193015
  ],
  "zoom": 13,
  "maxZoom": 19,
  "tileUrl": "https://[tile-url]/osm/{z}/{x}/{y}.png",
  "routeUrl": "https://[route-url]",
  "animate": true,
  "doubleClickZoom": true,
  "scrollWheelZoom": true,
  "className": "osm",
  "bounds": [
    [
```

```
        56.268142388481465,
        47.866744995117195
      ],
      [
        56.00298848125946,
        46.59645080566407
      ]
    ],
    "zoomControl": false,
    "style": {
      "line": {
      "strokeWidth": 2,
      "color": "var(--blue)"
      }
    }
}
```

4. Добавить в файл .env-cmdrc

```
"REACT_APP_REFERENCE": "[example]",
"PUBLIC_URL": "/[example-app-dir]"
"REACT_APP_APK_VERSION_URL": "https://[example]/[example-rpc-dir]/upload/version/0",
"REACT_APP_APK_DOWNLOAD_URL": "https://[example]/[example-rpc-dir]/upload/version-file?from=release"
```

5. Добавить в package.json раздел scripts конфиг

например "build_[example]": "env-cmd -e [example] react-scripts build",

6. Собрать приложение npm run build_[example]

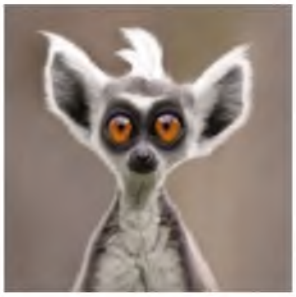
7. Скопировать на сервер linux папку build в /var/www/[example]

8. Настроить nginx

sudo nano /etc/nginx/sites-enabled/default

```
server {
    ...
    location /[example] {
      alias /var/www/[example];
    }
    ...
}
```

9. Нужно проверить доступ к бэкенду https://[example]/[example-rpc-dir]



Not Found

404

```
NotFoundError: Not Found
    at /var/www/cic-dev-node/index.j
    at Layer.handle [as handle_reque
    at trim_prefix (/var/www/cic-dev
    at /var/www/cic-dev-node/node_mo
    at Function.process_params (/var
    at next (/var/www/cic-dev-node/n
    at /var/www/cic-dev-node/node_mo
    at next (/var/www/cic-dev-node/n
    at Function.handle (/var/www/cic
    at router (/var/www/cic-dev-node
```

Результат доступен по адресу https://[example]/[example-app-dir]