

Санкт-Петербургский политехнический университет Петра Великого
Институт Компьютерных Наук и Технологий
Кафедра «Высшая школа программной инженерии»

Курсовой проект

по дисциплине: «Микропроцессорные системы»

Выполнил
студент гр.33534/21

С.А. Фомин

Проверил
преподаватель

С.К. Крутлов

Санкт-Петербург
2018

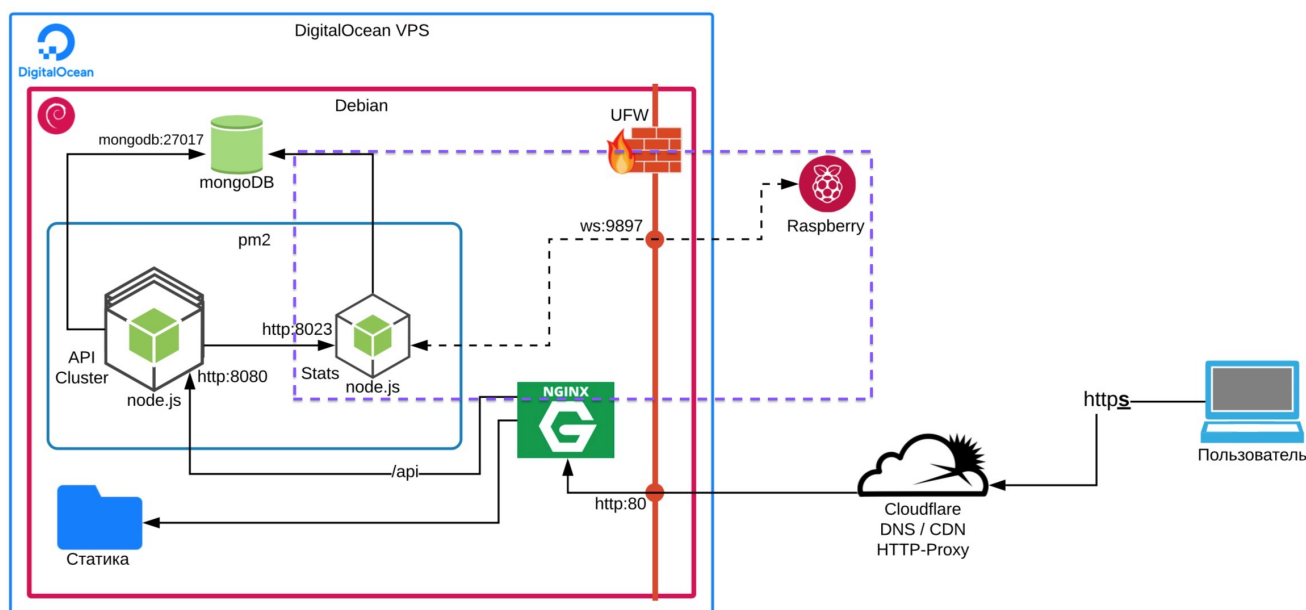


Рис. 1: Схема архитектуры крупным планом.

Постановка задачи

Разработать клиент-серверное приложение для вывода на внешний жидкокристаллический дисплей (с помощью Raspberry Pi) количества сокращенных ссылок из базы данных сервиса short.taxnuke.ru в режиме реального времени.

Ход работы

В качестве программной среды было принято решение использовать Node.js - кроссплатформенное решение, использующее в качестве JavaScript-движка V8, написанный на C. Для Node.js имеется множество готовых библиотек на JavaScript с биндингами к нативным модулям на C++ и C и с многопоточностью. Выбор обусловлен тем, что изначально сервис short.taxnuke.ru тоже написан Node.js и использование одной программной среды и языка позволяет минимизировать количество зависимостей и упростить интерфейс между компонентами. Также Node.js позволяет порождать дочерние процессы и передавать в них аргументы командной строки, управляющие последовательности и использовать потоки ввода/вывода.

На рисунке 1 изображена схема сервиса short.taxnuke.ru в целом, а также интегрированного в него клиент-серверного приложения для отображения статистики (в пунктирной фиолетовой рамке).

Результат работы

В результате выполнения курсового проекта было разработано серверное и клиентское приложения, работающие в паре и выполняющие поставленную задачу.

ПРИЛОЖЕНИЕ 1. Код сервера

```
const WebSocket = require('ws')
const hookPort = process.env.HOOK_PORT

const url = `mongodb://localhost:${process.env.MONGO_PORT}`

require('mongodb').MongoClient
  .connect(url)
  .then(mongoConnection => {
    let latestData = null

    const httpServer = require('express')()
    const websocketServer = new WebSocket.Server({
      port: process.env.WS_PORT
    })

    websocketServer.broadcast = function broadcast(data) {
      console.info(`broadcasting ${JSON.stringify(data)}...`)

      websocketServer.clients.forEach(function each(client) {
        if (client.readyState === WebSocket.OPEN) {
          client.send(data)
        }
      })
    }

    websocketServer.on('connection', function connection(ws) {
      ws.send('hi')

      setTimeout(() => {
        ws.send(latestData)
      }, 3000)
    })

    httpServer.use('*', (req, res) => {
      mongoConnection.db('url-shortener').collection('aliases')
        .countDocuments()
        .then(count => {
          res.end()

          const data = JSON.stringify({
            link_count: count
          })

          websocketServer.broadcast(data)

          latestData = data
        })
        .catch(console.error)
    })
  })
```

```
    })

    httpServer.listen(hookPort)
      .on('listening', () => {
        console.log('HTTP server listening on port ${hookPort}')
      })
  })
  .catch(console.error)
```

ПРИЛОЖЕНИЕ 2. Код клиента

```
const I2CLCDConnection = require('lcdi2c')
const WebSocket = require('ws')

/**
 * i2cdetect 1
 */
const I2C_ADDR = 0x3f
const lcdConnection = new I2CLCDConnection(1, I2C_ADDR, 16, 2)
lcdConnection.println('short.taxnuke.ru', 1)

const ws = new WebSocket(
  'ws://${process.env.WS_HOST}:${process.env.WS_PORT}'
)

ws.on('open', () => {
  lcdConnection.println('connected', 2)
  console.info('WebSocket connection established')
})

ws.on('close', function close() {
  console.warn('WebSocket connection closed')
  lcdConnection.println('conn closed', 2)
})

ws.on('message', data => {
  console.log(data)

  lcdConnection.clear()

  try {
    data = JSON.parse(data)
  } catch (e) {
    console.error(data)
  } finally {
    lcdConnection.println('short.taxnuke.ru', 1)
    lcdConnection.println(
      'links: ${data.link_count} || 'Error =( ',
      2
    )
  }
})
```

ПРИЛОЖЕНИЕ 3. Содержимое конфигурационного файла для раз- вертки

```
module.exports = {
  // Options reference: https://pm2.io/doc/en/runtime/reference/ecosystem-f
  apps: [
    {
      /**
       * SERVER
       */
      name: 'url-shortener-status-server',
      script: 'source/server.js',
      env: {
        watch: true,
        HOOK_PORT: 8813,
        WS_PORT: 9897,
        MONGO_PORT: 28017,
        NODE_ENV: 'development'
      },
      env_vps: {
        watch: false,
        HOOK_PORT: 8813,
        WS_PORT: 9897,
        MONGO_PORT: 27017,
        NODE_ENV: 'production'
      }
    },
    {
      /**
       * CLIENT
       */
      name: 'url-shortener-status-client',
      script: 'source/client.js',
      env: {
        watch: true,
        WS_PORT: 9897,
        WS_HOST: 'localhost',
        NODE_ENV: 'development'
      },
      env_raspberry: {
        watch: false,
        WS_PORT: 9897,
        WS_HOST: '138.68.183.160',
        NODE_ENV: 'production'
      }
    }
  ],
  deploy: {
    vps: {
```

```

    user: 'adminus',
    host: '138.68.183.160',
    ref: 'origin/master',
    repo: 'git@github.com:taxnuke/url-shortener-status.git',
    path: '/var/www/stat.short.taxnuke.ru',
    'post-deploy': 'npm i && pm2 reload ecosystem.config.js --env vps --o
  },
  raspberry: {
    user: 'pi',
    host: '192.168.1.70',
    ref: 'origin/master',
    repo: 'git@github.com:taxnuke/url-shortener-status.git',
    path: '/var/www/stat.short.taxnuke.ru',
    'post-deploy': 'npm i && pm2 reload ecosystem.config.js --env raspber
  }
}

```