**Минобрнауки России**

**Юго-Западный государственный университет**

**Кафедра программной инженерии**

**ОТЧЕТ О САМОСТОЯТЕЛЬНОЙ РАБОТЕ**

по дисциплине « Разработка интернет-приложений »

(название дисциплины)

на тему « Разработка приложения для сохранения данных из

(название темы)

сети Интернет в базу данных SQLite »

Направление подготовки (специальность) 09.04.04 Программная инженерия

(код, наименование)

Автор работы

(инициалы, фамилия) (подпись, дата)

Группа ПО-61мз

Руководитель работы Серебровский В.В.

(инициалы, фамилия) (подпись, дата)

Работа защищена

(дата)

Оценка

Курск 2018 г.

**Содержание**

[Текст программы 3](#__RefHeading__1317_316514094)

[Список использованных источников 4](#__RefHeading__1327_2125950363)

[Приложение А 5](#__RefHeading__815_1805353137)

[Приложение Б 16](#__RefHeading__1755_1745017547)

# Текст программы

Согласно ГОСТ 19.401-78 «Единая система программной документации (ЕСПД). Текст программы. Требования к содержанию и оформлению» текст программы оформляют следующим образом. п.2 Структуру и оформление документа устанавливают в соответствии с ГОСТ 19.105-78. Составление информационной части (аннотации и содержания) является необязательным. Для текста программы на исходном языке при наличии аннотации в нее включают краткое описание функций программы. п.3. Основная часть документа должна состоять из текстов одного или нескольких разделов, которым даны наименования. Допускается вводить наименование также и для совокупности разделов. п.4. … В символическую запись разделов рекомендуется включать комментарии,

которые могут отражать, например, функциональное назначение, структуру [1].

Текст программы файла scrptwitterandmedium.js на языке Javascript приведен в приложении А (Приложение А). Текст программы файла package.json в формате json приведен в приложении Б (Приложение Б).

При разработке программы использованы следующие стандарты: стандарт для языка JavaScript [2].

В текстах программ, представленных в приложениях А-Б включены комментарии, отражающие функциональное назначение и структуру программ.

# Список использованных источников

1. ГОСТ 19.401-78 «Единая система программной документации (ЕСПД). Текст программы. Требования к содержанию и оформлению» [Текст] : ГОСТ 19.401–78.- Введ.1980-01-01.

2. Standard ECMA-262 ECMAScript 2017 Language Specification 8th edition (June 2017) [Электронный ресурс] — 2017. // Режим доступа - http://www.ecma-international.org/publications/standards/Ecma-262.htm

# Приложение А

Текст программы файла scrptwitterandmedium.js на языке Javascript.

(код также доступен на github.com по ссылке <https://github.com/radioaktive/scrpmediumpagedb/blob/master/scrptwitterandmedium.js> )

// alg scraper of twitter and medium.com to cms pagekit db (sqlite)

const puppeteer = require('puppeteer');

const cheerio = require('cheerio');

const sqlite3 = require('sqlite3').verbose();

// alg input

const websites =[

// twitter Бутерина

{ startURL: "https://twitter.com/VitalikButerin",

// селектор ссылок на твиты

linkSelector: "#stream-items-id a.tweet-timestamp",

accountName: "Виталик Бутерин",

titleBeginWith: "Переводим твиттер Бутерина: ",

subjectSelector: ".permalink-tweet",

// селекторы кнопок для копирования кода для вставки твита

button1Selector: '.permalink-tweet span.Icon.Icon--caretDownLight.Icon--small',

button2Selector: '.permalink-tweet li.embed-link.js-actionEmbedTweet > button',

// селектор кода твита для вставки

embSelector: '#embed-tweet-dialog-dialog > div.modal-content > div.modal-body > div > form > div > textarea', // работает напрямую а не отсюда

//селектор текста твита

firstpSelector: '.permalink-tweet p.TweetTextSize--jumbo',

// селектор даты/времени

timeSelector: '.permalink-tweet span.metadata',

// селектор лайков

likeSelector: '.permalink-tweet a.request-favorited-popup'

},

// medium.com по поиску "криптовалют"

{ startURL: "https://medium.com/search?q=%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B2%D0%B0%D0%BB%D1%8E%D1%82",

// селектор ссылок на посты

linkSelector: ".postArticle-readMore a",

subjectSelector: "article",

// селектор текста

pSelector: "p, blockquote, figure",

// селектор родительского элемента текста

pparentSelector: '.sectionLayout--insetColumn, .sectionLayout--fullWidth',

// селектор начала текста

firstpSelector: '.sectionLayout--insetColumn p, .sectionLayout--insetColumn blockquote',

// селектор даты/времени

timeSelector: 'time',

// селектор лайков

likeSelector: '.js-actionMultirecommendCount'

}

];

// регулярка с ключевыми словами по которым проверяем соответствие

const regex = /(?:Bitcoin|биткоин|Buterin|Бутерин|криптовалюта|криптовалют|Cryptocurrency|crypto-currencies|cryptocurrencies|mining|майнинг|майнит|майнят|Ethereum|Scala|Solidity|Blockchain|блокчейн)/gi;

function writeDB(results) {

console.log("work with db starts");

console.dir(results);

var db = new sqlite3.Database('pagekit.db');

db.serialize(function(){

db.run('DROP TABLE IF EXISTS new');

// alg10 save content from array to db // сохраняем контент из массива в базу данных

db.run('CREATE TABLE new (id INTEGER, user\_id INTEGER UNSIGNED, slug VARCHAR (255), title VARCHAR (255), status SMALLINT, date DATETIME, modified DATETIME, content CLOB, excerpt CLOB, comment\_status BOOLEAN, comment\_count INTEGER, data CLOB, roles CLOB, link TEXT, time TEXT, keywords TEXT, likes INTEGER)');

var stmt = db.prepare('INSERT INTO new VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)');

for (var i = 0; i < results.length; i++) {

stmt.run(results[i]);

};

stmt.finalize();

// в родную базу pagekit в таблицу pk\_blog\_post нужно предварительно добавить 1 раз столбцы link, time, keywords, likes, state.

db.run('CREATE TABLE IF NOT EXISTS pk\_blog\_post (id INTEGER PRIMARY KEY AUTOINCREMENT, user\_id INTEGER UNSIGNED, slug VARCHAR (255), title VARCHAR (255), status SMALLINT, date DATETIME, modified DATETIME, content CLOB, excerpt CLOB, comment\_status BOOLEAN, comment\_count INTEGER, data CLOB, roles CLOB, link TEXT, time TEXT, keywords TEXT, likes INTEGER, state TEXT)');

db.run('UPDATE pk\_blog\_post set state = NULL');

// alg output

// alg11 check with db and save content only from new links // сверяем с базой и сохраняем в итоговую таблицу только контент из новых ссылок

db.run('INSERT INTO pk\_blog\_post SELECT id, user\_id, slug, title, status, date, modified, content, excerpt, comment\_status, comment\_count, data, roles, link, time, keywords, likes, "new" AS state FROM new ' +

'WHERE link IN (SELECT link FROM new EXCEPT SELECT link FROM pk\_blog\_post)');

db.run('DROP TABLE new');

db.close();

});

};

function hashCode(string) {

var hash = 0;

if (string.length === 0) return hash;

for (i = 0; i < string.length; i++) {

chr = string.charCodeAt(i);

hash = ((hash << 5) - hash) + chr;

hash |= 0; // Convert to 32bit integer

hash = Math.abs(hash);

}

return hash;

};

function translit(txt) {

var str = txt;

var space = '-';

var sluglink = '';

var transl = {

'а': 'a', 'б': 'b', 'в': 'v', 'г': 'g', 'д': 'd', 'е': 'e', 'ё': 'e', 'ж': 'zh',

'з': 'z', 'и': 'i', 'й': 'j', 'к': 'k', 'л': 'l', 'м': 'm', 'н': 'n',

'о': 'o', 'п': 'p', 'р': 'r','с': 's', 'т': 't', 'у': 'u', 'ф': 'f', 'х': 'h',

'ц': 'c', 'ч': 'ch', 'ш': 'sh', 'щ': 'sh','ъ': '',

'ы': 'y', 'ь': '', 'э': 'e', 'ю': 'yu', 'я': 'ya'

}

if (str != '')

str = str.toLowerCase();

for (var i = 0; i < str.length; i++){

if (/[а-яё]/.test(str.charAt(i))){ // заменяем символы на русском

sluglink += transl[str.charAt(i)];

} else if (/[a-z0-9]/.test(str.charAt(i))){ // символы на анг. оставляем как есть

sluglink += str.charAt(i);

} else {

if (sluglink.slice(-1) !== space) sluglink += space; // прочие символы заменяем на space

}

}

return sluglink

}

function getDate() {

var datenow = new Date();

var month = datenow.getMonth()+1;

var datecur = datenow.getDate()-1;// минусован 1 день, чтобы пост сразу отображался в pagekit

time\_sec=datenow.getSeconds();

time\_min=datenow.getMinutes();

time\_hours=datenow.getHours();

date=datenow.getFullYear() + "-" + month + "-" + datecur + " ";

date+=((time\_hours<10)?"0":"")+time\_hours;

date+=":";

date+=((time\_min<10)?"0":"")+time\_min;

date+=":";

date+=((time\_sec<10)?"0":"")+time\_sec;

return date;

}

const getText = async (link, website, browser) => {

console.log(`Now checking ${link}`);

try {

const page = await browser.newPage();

await page.waitFor(1500);

// alg6 open link to post from queue // открываем ссылку на пост в очереди

await page.goto(link);

await page.waitFor(3000);

let content = await page.content();

var $ = cheerio.load(content);

//var alltext = $("body").text();

//console.log(alltext);

var subject = $(website.subjectSelector).text();

//console.log(website.subjectSelector);

//console.log(subject);

var keywords = subject.match(regex);

// alg7 check keywords // проверяем на соответствие ключевым словам

if (keywords){

var res = [];

var elhtmls = [];

var title = '';

// alg8 find and parse content // находим и обрабатываем контент

if (website.button1Selector){

title += website.titleBeginWith;

if (link.startsWith(website.startURL)){

elhtmls.push('<p> ' + website.accountName + ' ' + $(website.timeSelector).first().text() + ' сделал такой твит: </p>');

} else {

elhtmls.push('<p> ' + website.accountName + ' ' + $(website.timeSelector).first().text() + ' сделал ретвит: </p>');

}

await page.click(website.button1Selector);

await page.waitFor(1500);

await page.click(website.button2Selector);

await page.waitFor(5000);

const emb = await page.evaluate(() => {

let direct = document.querySelector('#embed-tweet-dialog-dialog > div.modal-content > div.modal-body > div > form > div > textarea').value;

return direct;

});

elhtmls.push(emb);

elhtmls.push('<p>\_\_\_\_\_\_\_\_\_\_\_\_\_ <br> <a href="' + link + '">Источник</a> </p>');

var twitId = link.split('/').slice(-1);

elhtmls.push('<p>Пишите в комментариях свой вариант перевода начиная словами "это можно перевести как" и лучший вариант набравший больше всего лайков будет дополнен в пост. </p> <!-- Put this div tag to the place, where the Comments block will be --> <div id="vk\_comments\_' + twitId + '"></div> <script type="text/javascript"> VK.Widgets.Comments("vk\_comments\_' + twitId + '", {limit: 15, attach: "\*"}); </script>');

} else {

$(website.pSelector).parent(website.pparentSelector).each(function(i, element){

var elhtml = $(this).html();

var newelhtml1 = elhtml.replace(/<h1(.\*?)h1>/g, "");

var newelhtml2 = newelhtml1.replace(/<figure(.\*?)canvas>/g, "");

var newelhtml3 = newelhtml2.replace(/<noscript(.\*?)figure>/g, "");

var newelhtml4 = newelhtml3.replace(/data-src=/g, "src=");

elhtmls.push(newelhtml4);

});

elhtmls.push('<p>\_\_\_\_\_\_\_\_\_\_\_\_\_ <br> <a href="' + link + '">Источник</a> </p>');

}

if($(website.subjectSelector).find('h1').is('h1')) {

title = $('h1').text();

}

else {

title += $(website.firstpSelector).first().text().split(' ').slice(0, 3).join(' ');

};

//var title = $('h1').text();

//var slug = title.toLowerCase().replace(/[^\w\s]|\_/g, "").replace(/\s/ig, '-');

var slug = hashCode(link) + "-" + translit(title);

var date = getDate();

var likesSl = $(website.likeSelector).first().text();

//console.log(likesSl);

likesSl = likesSl.replace(/,/g, '');

likesSl = likesSl.replace(/\s+(?![^\d\s])/g, '');

var likes = parseInt(likesSl, 10);

if (isNaN(likes)) {

likes = 0;

}

var excerpt = likes + " лайков";

var nn = null;

// alg9 save content in an array and return it // сохраняем контент поста в массив и потом его возвращаем

res.push(

nn, // 1 id

"1", // 2 user\_id

slug, // 3 slug

title, //4 title

"3", // 5 status

date, // 6 date

date, // 7 modified

elhtmls.join(" <br>"), // 8 content

excerpt, // 9 excerpt

"1", // 10 comment\_status,

"0", // 11 comment\_count

'{"title":null,"markdown":false,"image":{"src":"","alt":""}}', // 12 data

nn, // 13 roles

link, // link

$(website.timeSelector).first().text(), // time

keywords.join(), // keywords

likes // likes

);

};

await page.close()

return res;

} catch (error) {

console.error(error);

throw error;

}

};

async function createGetLinks(website, browser) {

var GetLinksFunction;

if (website.startURL.startsWith('http')) {

GetLinksFunction = async function() {

const page = await browser.newPage();

await page.waitFor(2000);

// alg2 go to website // заходим на сайт

await page.goto(website.startURL);

await page.waitFor(10000);

await page.waitForSelector(website.linkSelector);

let content = await page.content();

var $ = cheerio.load(content);

//var alltext = $("body").text();

//console.log(alltext);

const websiteURL = website.startURL.split('/').slice(0, 3).join('/');

console.log(websiteURL);

var links = [];

//alg3 find links to posts // находим ссылки на посты

$(website.linkSelector).each(function(){

item = {}

item['link'] = $(this).attr('href');

item['link'] = item['link'].replace(/\?source.\*/, ''); //

if (item['link'].startsWith('/')){

item['link'] = websiteURL + item['link'];

};

links.push(item['link']) // массив с готовыми ссылками

});

await page.close();

//console.log(links);

//alg4 return array of links to posts // возвращаем массив ссылок на посты

return links;

};

} else {

GetLinksFunction = async function(){

console.log("website url is not correct");

};

}

return GetLinksFunction;

}

async function scrapeWebsite (website, browser) {

var getGetLinksFunction = await createGetLinks(website, browser);

await getGetLinksFunction().then(function (links) {

console.log(links);

//alg5 form queue of links to posts // формируем из ссылок на посты очередь

const series = links.reduce(async (queue, link) => {

const dataArray = await queue;

dataArray.push(await getText(link, website, browser));

return dataArray;

}, Promise.resolve([]));

series.then(results => {

console.log("in seriesthen");

console.dir(results);

writeDB(results);

})

})

};

/\*

async function scrapeWebsitesParallel(websites) {

const browser = await puppeteer.launch({headless: false});

websites.forEach(async website => {

await scrapeWebsite(website, browser);

});

};

\*/

async function scrapeWebsites(websites) {

const browser = await puppeteer.launch({headless: false});

// alg1 start scraping websites // запускаем сайты на скрапинг, в данном случае по идее сайты должны были запускаться последовательно, но работает параллельно

for (let i = 0; i < websites.length; i++) {

await scrapeWebsite(websites[i], browser);

};

/\*

try {

// alg end

browser.close(); // браузер закрывать вручную, иначе выдает ошибку

} catch (error) {

console.error(error);

throw error;

}

\*/

};

//scrapeWebsitesParallel(websites);

// alg0 start // запускаем

scrapeWebsites(websites);

# Приложение Б

Текст программы файла package.json в формате json.

{

"name": "nodejsscrpuppeteer",

"version": "0.0.0",

"private": true,

"scripts": {

"start": "node ./bin/www"

},

"dependencies": {

"@datafire/hacker\_news": "^2.0.2",

"body-parser": "~1.18.2",

"cheerio": "^1.0.0-rc.2",

"cookie-parser": "~1.4.3",

"d3-dsv": "^1.0.8",

"datafire": "^2.18.2",

"debug": "~2.6.9",

"express": "~4.15.5",

"morgan": "~1.9.0",

"puppeteer": "git+https://github.com/GoogleChrome/puppeteer.git",

"serve-favicon": "~2.4.5",

"sqlite3": "^3.1.13",

"twig": "~0.10.3"

}

}