UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și Tehnologia Informației
SPECIALIZAREA: Tehnologia Informației

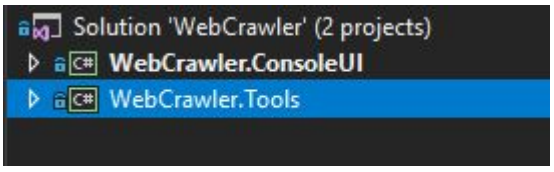
RIW Crawler

Disciplina: Regăsirea informațiilor pe web
Profesor coordonator: Alexandru Archip

Student:
Bălăucă Remus-Florin 1406A

1.Descrierea soluţiei

Proiectul a fost implementat în .NET Core 2.0, având posibilitatea de a rula aplicaţiile implementate şi în alte sisteme de operare în afara de Windows.



Proiectul **WebCrawler.ConsoleUI** este o aplicaţie de tip consolă ce foloseşte proiectul **WebCrawler.Tools** pentru partea de crawling.

1.1. Căutarea DNS-ului

Modulul de căutare a DNS-ului se află în clasa **DnsResolver**, în care se construieşte cererea în funcţie de host şi în care parsăm răspunsul pentru recordurile de tip **ANAME**.

```
public class DnsResolver
{

  public static DnsResponse GetDnsAddress(string host)
  {
    var requestSize = 12 + host.Length + 2 + 4; // header + url size +2 + qtype & qclass
    var requestBytes = new byte[requestSize];

    //ID - 2 bytes
    requestBytes[0] = 0x17;
    requestBytes[1] = 0x04;
    //flags
    requestBytes[2] = (byte)(QrFlag.Request | OperationCode.Query | AnswerFlag.NonAuthoritative | TruncationFlag.False | RecurtionDesired.True);
    requestBytes[3] = (byte)(RecurtionAvailable.False | Zero.Reserved | AuthenticationData.False | CheckingDisabled.False | ResponseCode.NoError);
    //total questions count
    requestBytes[4] = 0x00;
    requestBytes[5] = 0x01;
```

Construirea cererii

```
var type = (responseBytes[idx] << 8) | responseBytes[idx + 1];
if (type != 1)
{
  skip = true;
  //Console.WriteLine("Non ipv4 address");
}
idx += 4; //skip name, type and class 2+2+2


var ttl = (responseBytes[idx] << 24) | (responseBytes[idx + 1] << 16) | (responseBytes[idx + 2] << 8) | responseBytes[idx + 3];
var expireDate = DateTime.Now.AddSeconds(ttl);
idx += 4;

//since we extract IPV4 addresses, the length is 4 bytes
var rdLength = (responseBytes[idx] << 8) | responseBytes[idx + 1];
idx += 2;
if (!skip)
{
  var firstAname = new StringBuilder();
  for (int i = idx, end = idx + 4; i < end; i++)
  {
    var segment = (int)responseBytes[i];
    firstAname.Append($"{segment}.");
  }
  return new DnsResponse()
  {
    ExpireDate = expireDate,
    IpAddress = firstAname.ToString(0, firstAname.Length - 1) //remove last dot
  };
}
idx += rdLength;
```

Parsarea răspunsului.

1.2. Cozile de url-uri

Pentru partea de extragere a url-urilor, am folosit două cozi, definite in clasa **UrlFrontier**. O coadă pentru url-uri care aparţin aceluiaşi domeniu (cel curent, pe care îl procesează) şi o coadă pentru url-uri externe.

Atunci când coada cu url-uri pentru domeniul curent este goală, se extrage un alt domeniu din coada pentru url-uri externe. De asemenea, în cozile respective se adaugă adrese care nu conţin extensie sau daca extensia lor face parte din una cunoscută (pentru evitarea descărcării de fişiere în loc de conţinut html)

1.3. Cache DNS

Atunci când adăugăm un nou dns în cache, verificăm să nu depăşească limita definită de noi. În caz că limita este depăşită , ştergem dns-urile expirate, iar daca nu sunt foarte multe date invalide, mai ştergem câteva din primele dns-uri adăugate până ajungem la o zecime de dns-uri şterse. Datorită implementării cu 2 cozi, este improbabil să mai avem nevoie de primele dns-uri inserate.

```
public static void TrimDnsCache()
{
    var countBeforeTrim = Dns.Count;
    var tenth = countBeforeTrim / 10;

    //remove the ones with expired dates first
    foreach (var item in Dns.Where(x=>x.Value.ExpireDate < DateTime.Now).ToList())
    {
        Dns.Remove(item.Key);
    }
    if (Dns.Count != countBeforeTrim)
    {
        Console.WriteLine($"Removed {countBeforeTrim - Dns.Count} expired dns");
    }

    //if cache is still quite full, then trim a little more
    var rest = tenth - (countBeforeTrim - Dns.Count);
    if (rest < 0)
    {
        return;
    }
    foreach (var item in Dns.Take(rest).ToList())
    {
        Dns.Remove(item.Key);
    }
    Console.WriteLine("Trimming DNS to 9 tenths");
}
```

Ştergerea chache-ului DNS

1.4. Web Client

Conţinutul acestui modul se află în clasa **TcpCrawlingClient**. În acest modul extragem dns-ul din cache sau, dacă nu există, facem o cerere pentru a-l obţine. Construim headerele pentru cererea paginii şi extragem răspuns headerele acestuia şi conţinutul propriu zis.

```
var requestMessage = new StringBuilder();
requestMessage.Append($"GET {relativePath} HTTP/1.1\r\n");
requestMessage.Append($"Host:{uri.Authority}\r\n"); //riweb.tibeica.com
requestMessage.Append($"User-Agent:{RunSettings.UserAgent}\r\n");
requestMessage.Append("Connection:close\r\n");
```

Construirea cererii pentru o anumită pagină web

1.5. REP

Pentru respectarea acestui protocol, se efectuează o cerere pentru fişierul **robots.txt**, folosind modulul de **Web Client**. Răspunsul este parsat şi se extrag informaţiile de tip **Disallow** şi **Allow** pentru user agentul curent şi pentru wildcard (*). Înainte de a trimite o cerere, verificăm mai întâi dacă avem acces pe pagina respectivă. De asemenea, dacă am obţinut conţinutul paginii, căutăm tag-ul meta pentru robots şi luăm în considerare atributul rel=”nofollow” al ancorelor.

```
var robotsMeta = metaTags.FirstOrDefault(x => x.Attributes["name"]?.Value == "robots")?.Attributes["content"]?.Value;
if (robotsMeta != null)
{
  containsNoFollow = robotsMeta.Contains("nofollow");
  containsNoIndex = robotsMeta.Contains("noindex");
}


var anchors = doc.DocumentNode.Descendants("a");
if (!containsNoIndex)
{
  //we can save it on the disk so the indexer can do its job
  FileTreeWorker.CreateWebsiteTree(baseUrl, page.Html);
}
if (containsNoFollow)
{
  //do not extract the links
  return;
}

foreach (var a in anchors)
{
  var currentAnchor = a.Attributes["href"]?.Value;
  if (string.IsNullOrEmpty(currentAnchor)) continue;
  var rel = a.Attributes["rel"]?.Value;
  if (rel != null && rel == "nofollow")
  {
    continue;
  }
```

Robots Meta

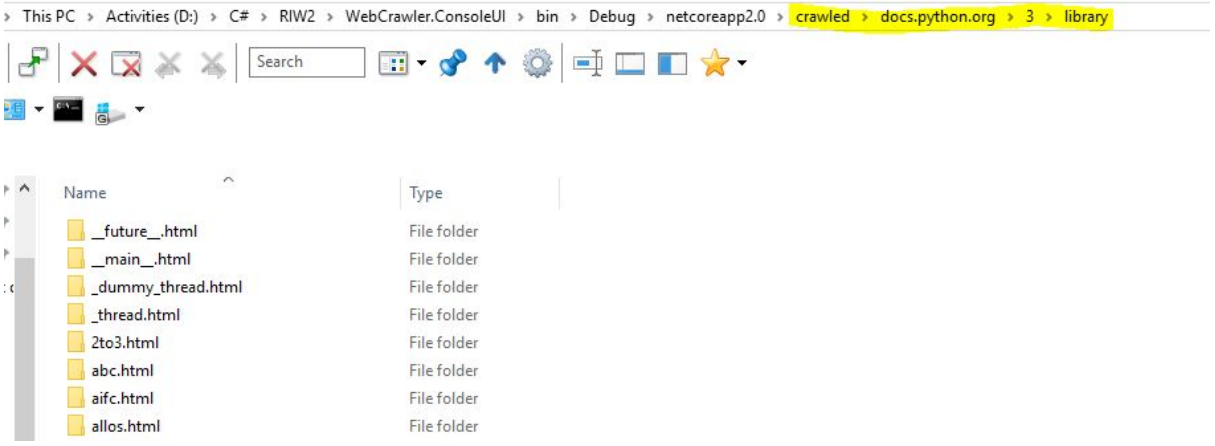
2. Rezultate

Adresele Url sunt procesate secvenţial, iar rezultatele depind şi de viteza de răspuns a paginilor. Pentru **riweb.tibeica.com**, primele 100 de pagini sunt procesate în ~27s.

```
http://riweb.tibeica.com/crawl/pyapi-mpfilt-meth.html
http://riweb.tibeica.com/crawl/pyapi-mpsrv-mem.html
http://riweb.tibeica.com/crawl/pyapi-mpsrv-meth.html
Downloaded 100 pages in 26.7427661 seconds.

http://riweb.tibeica.com/crawl/pyapi-mpconn-meth.html
http://riweb.tibeica.com/crawl/hand-pub-alg-trav.html
http://riweb.tibeica.com/crawl/hand-pub-alg-args.html
http://riweb.tibeica.com/crawl/hand-pub-alg-auth.html

--Got url from url queue: https://drive.google.com--

Processing redirect

--Got url from url queue: http://www.python.org--

https://www.python.org/psf-landing/
```

```
https://docs.python.org/3/library/marshal.html
https://docs.python.org/3/library/macpath.html
https://docs.python.org/3/library/email.contentmanager.html
https://docs.python.org/3/library/aifc.html
Trimming cached visited urls to 3 quarters
https://docs.python.org/3/library/filecmp.html
https://docs.python.org/3/library/operator.html
https://docs.python.org/3/library/xml.html
https://docs.python.org/3/howto/functional.html
https://docs.python.org/3/library/gettext.html
https://docs.python.org/3/library/poplib.html
https://docs.python.org/3/library/resource.html
Downloaded 100 pages in 29.4004441 seconds.
```

Paginile sunt salvate într-o structură aborescentă



3. Autoevaluare

Aplicație: 9p

Bonus: 2p