Computer Linguistics

Course Work

"Link Retrieval" – A web scraping project

Author: Dimitar Milkov

Course: Informatics

Faculty Number: **1701261066**

E-mail: stu1701261066@uni-plovdiv.bg & dimitar.milkov@hotmail.com

Contents

Introduction

Greetings to you, readers! In this document, you will find all the necessary information regarding this course work – what its purpose is, what are the main objectives, how was the implementation achieved and so on.

The project is a course work assigned to me for the subject “Computer Linguistics”, specialty “Informatics” in the Plovdiv University “Paisii Hilendarski”. The discipline is led by Associate Professor Georgi Pashev for fourth-year students.

When I first started developing this project for the course work, I was eager to create a robust console application which fulfilled the requirements for this project, so I began the implementation of the web scraping part, but one evening while I was reading some articles regarding the new trends in web development, “know-hows” for system designers and software architects, “Top 10 tips for Angular developers”, I thought to myself “Why don’t I develop a full-scale web application?”.

So I began drawing some schematics on pieces of paper. At one point, my brain was so over-stimulated by random different ideas on how to develop the project that I threw all the pieces of paper, rolled up my sleeves and directly went to work. After I created the project structure for the back-end, I decided to call it a day in order to rethink my strategy, as mindlessly writing code gets you nowhere. After a day or two, the whole plan hatched in my brain and I finally realized what I have to do.

I hope you find the following pages useful and easy to read. Do keep in mind that this project is not officially finished and I have future plans for further implementing even more features and refining the data structure. Due to the lack of contributors with QA experience, there may be bugs which will be fixed later on.

Overview

The purpose of the software developed for this course work is to search a given web page for URLs via a regular expression at a specified depth level, all of which are provided by the user via a simple and robust user interface in a client application. Users can also browse all of the previously executed searches via a separate view. Upon clicking on a specific search, a detailed view is shown, in which the user can see all found URLs on that address. All queries are sent and executed via an API which is implemented with .NET 9 and Microsoft SQL using Entity Framework Core.

The used technologies are:

1. Front-End
   - 1.1. Angular 19
   - 1.2. Angular Material
   - 1.3. Angular CLI
   - 1.4. Node.js
   - 1.5. npm (Node Package Manager)
2. Back-End
   - 2.1. .NET 9
   - 2.2. MSSQL (Microsoft SQL)
   - 2.3. Entity Framework Core

Objectives

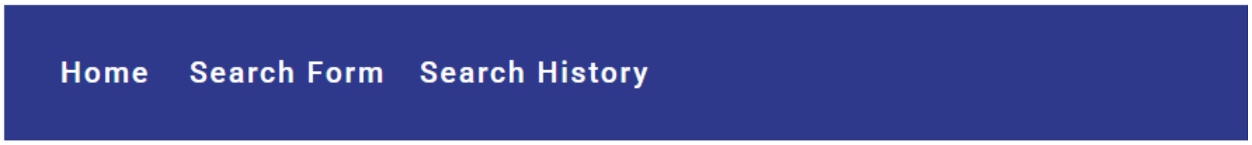
1. Develop an application that accepts a URL to any web page as input.
2. Load and parse the HTML source code of the specified page.
3. Search for text patterns using a user-defined regular expression.
4. Identify additional web page URLs through regular expressions and push them onto a stack for later processing.
5. Recursively process each discovered page in the same manner.
6. Store all matched text results in a database.
7. Maintain a database record of all visited and processed URLs to avoid duplication.

Implementation

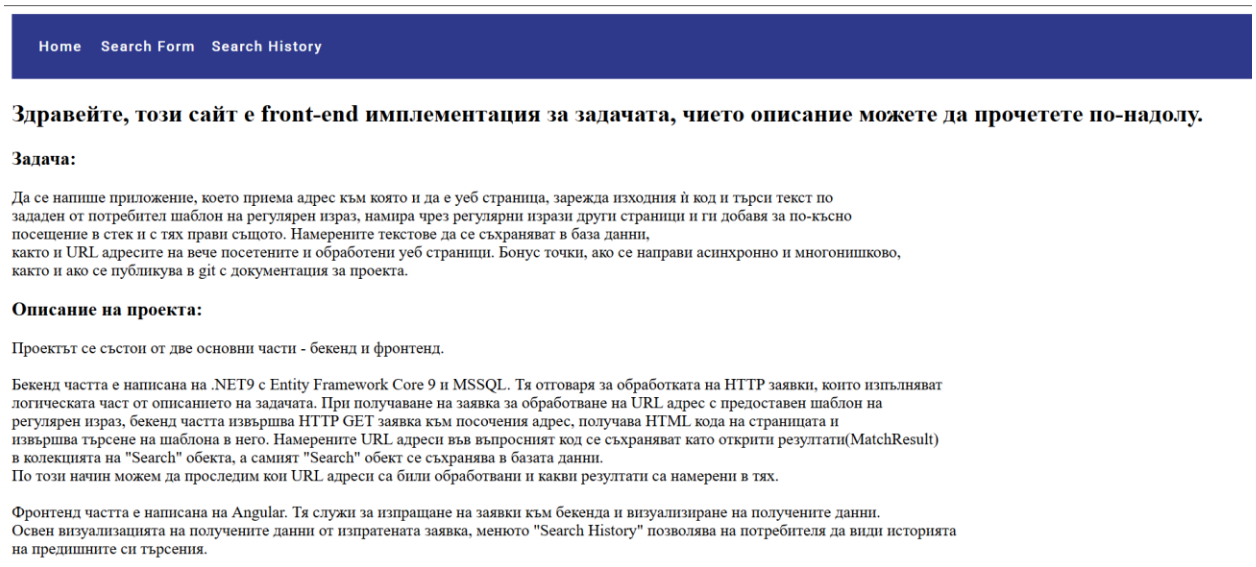
(this section with be split into two parts – Front-End and Back-End in order to make the reading smoother)

Front-End

The Front-End is created using Angular framework, Angular CLI, node.js and node package manager. Using separate components for custom elements in the views, the client application consists of a landing page("Home" page), "Search Form" in which the main functionality for web scraping is located and a "Search History" where all previous queries are listed using a table structure. All of these views are accessible through the navigation bar on the topmost part throughout the whole website.



(fig. 1 – Navigation bar)



Здравейте, този сайт е front-end имплементация за задачата, чието описание можете да прочетете по-надолу.

Задача:

Да се напише приложение, което приема адрес към която и да е уеб страница, зарежда изходния ѝ код и търси текст по
зададен от потребител шаблон на регулярен израз, намира чрез регулярни изрази други страници и ги добавя за по-късно
посещение в стек и с тях прави същото. Намерените текстове да се съхраняват в база данни,
както и URL адресите на вече посетените и обработени уеб страници. Бонус точки, ако се направи асинхронно и многонишково,
както и ако се публикува в git с документация за проекта.

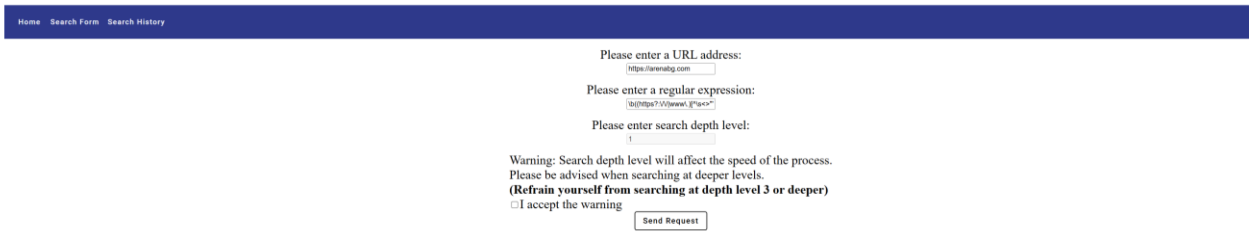
Описание на проекта:

Проектът се състои от две основни части - бекенд и фронтенд.

Бекенд частта е написана на .NET9 с Entity Framework Core 9 и MSSQL. Тя отговаря за обработката на HTTP заявки, които изпълняват
логическата част от описанието на задачата. При получаване на заявка за обработване на URL адрес с предоставен шаблон на
регулярен израз, бекенд частта извършва HTTP GET заявка към посочения адрес, получава HTML кода на страницата и
извършва търсене на шаблона в него. Намерените URL адреси във въпросният код се съхраняват като открити резултати(MatchResult)
в колекцията на "Search" обекта, а самият "Search" обект се съхранява в базата данни.
По този начин можем да проследим кои URL адреси са били обработвани и какви резултати са намерени в тях.

Фронтенд частта е написана на Angular. Тя служи за изпращане на заявки към бекенда и визуализиране на получените данни.
Освен визуализацията на получените данни от изпратената заявка, менюто "Search History" позволява на потребителя да види историята
на предишните си търсения.

(fig. 2 – Home Page)



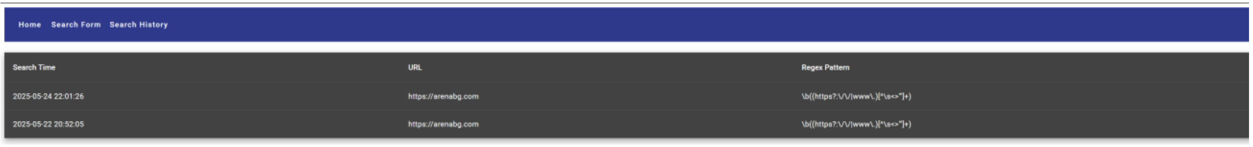
Please enter a URL address:
https://arenabg.com

Please enter a regular expression:
\b((https?:\/\/|www\.)[^\s<>"

Please enter search depth level:
1

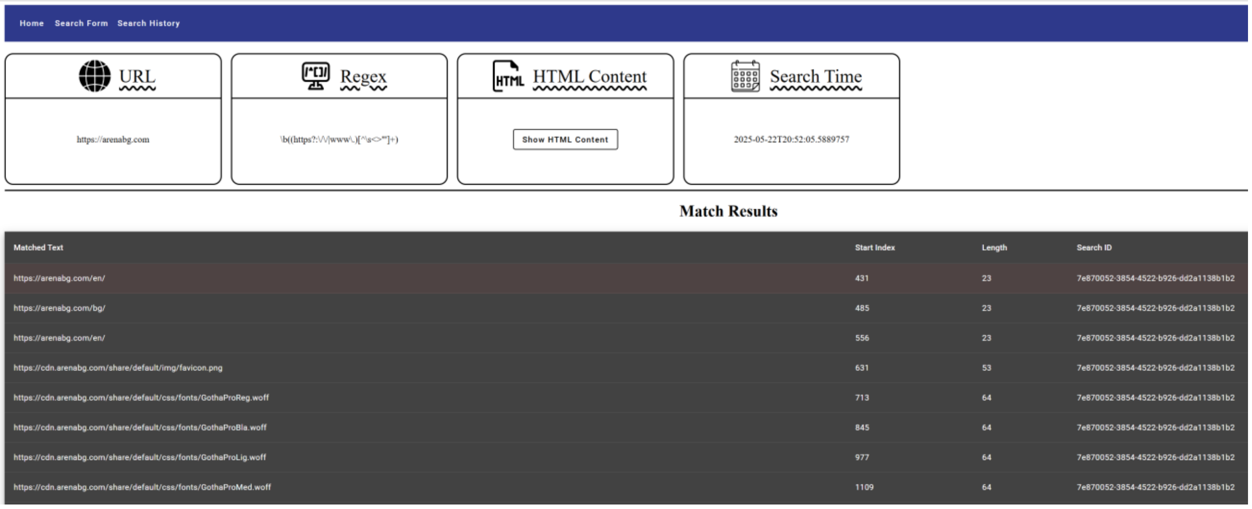
Warning: Search depth level will affect the speed of the process.
Please be advised when searching at deeper levels.
(Refrain yourself from searching at depth level 3 or deeper)
☐ I accept the warning

(fig. 3 – Search Form)



Search Time	URL	Regex Pattern
2025-05-24 22:01:26	https://arenabg.com	\b((https?:\/\/|www\.)[^\s<>"]+)
2025-05-22 20:52:05	https://arenabg.com	\b((https?:\/\/|www\.)[^\s<>"]+)

(fig. 4 – Search History)



https://arenabg.com

\b((https?:\/\/|www\.)[^\s<>"]+)

2025-05-22T20:52:05.5889757

Match Results

Matched Text	Start Index	Length	Search ID
https://arenabg.com/en/	431	23	7e870052-3854-4522-b926-dd2a1138b1b2
https://arenabg.com/bg/	485	23	7e870052-3854-4522-b926-dd2a1138b1b2
https://arenabg.com/en/	556	23	7e870052-3854-4522-b926-dd2a1138b1b2
https://cdn.arenabg.com/share/default/img/favicon.png	631	53	7e870052-3854-4522-b926-dd2a1138b1b2
https://cdn.arenabg.com/share/default/css/fonts/GothaProReg.woff	713	64	7e870052-3854-4522-b926-dd2a1138b1b2
https://cdn.arenabg.com/share/default/css/fonts/GothaProBla.woff	845	64	7e870052-3854-4522-b926-dd2a1138b1b2
https://cdn.arenabg.com/share/default/css/fonts/GothaProLig.woff	977	64	7e870052-3854-4522-b926-dd2a1138b1b2
https://cdn.arenabg.com/share/default/css/fonts/GothaProMed.woff	1109	64	7e870052-3854-4522-b926-dd2a1138b1b2

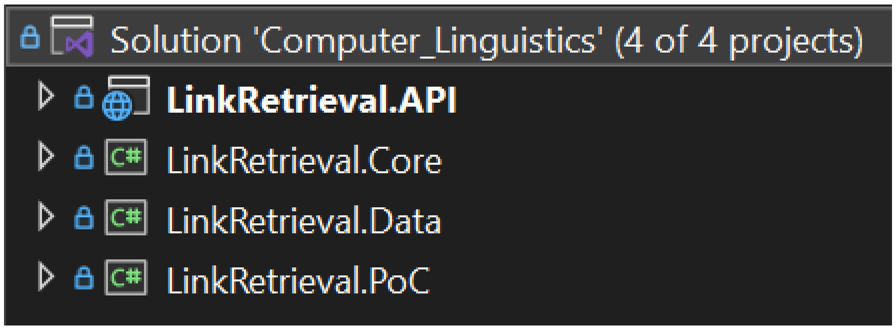
(fig. 5 – Detailed view for a search result)

Back-End

The Back-End is implemented using the following stack - .NET 9, Microsoft SQL and Entity Framework Core. The IDE used for this is Visual Studio 2022. The solution is separated in several projects – console project for PoC (Proof of Concept), Web API where all of the configurations and controller(s) are located, Core class library for service(s) and Data class library for models and Database Connector.

Here is a detailed explanation on what the projects consist of:

1. LinkRetrieval.API – startup project which consists of configuration files, a controller in which the implemented action methods for HTTP calls are located, “Program.cs” and “Startup.cs” where all services and properties are initialized and configured.
2. LinkRetrieval.Core – a class library project which contains the service interfaces and their implementations.
3. LinkRetrieval.Data – a class library project which contains the database connector and the data structure models.
4. LinkRetrieval.PoC – a console application in which proof of concept implementations are being made and tested.


Solution 'Computer_Linguistics' (4 of 4 projects)
- LinkRetrieval.API
- LinkRetrieval.Core
- LinkRetrieval.Data
- LinkRetrieval.PoC

(fig. 6 – project structure in the solution)

Testing

At this point in time, no test-specific projects have been made; future plans for this project include unit tests, etc. The only tests which were conducted during the development were manual tests during the development of the web application and manual tests during the initial Proof of Concept.

Conclusion and Future Development

The development of this project and the course work itself were a terrific joyride for me. As a software engineer who mainly works with Angular and AngularJS on Front-End tasks at work, I was able to further improve my skills with the Back-End stack used for this project. Each mistake and lack of understanding at certain points in the development lead me to read the technical documentation for the frameworks and gain the necessary knowledge to move on.

This product has a huge potential for improvements and learning material for those who are interested in implementing similar (if not same) functionalities or just looking for a project from which to read and learn. A little insight on what my future plans for this project are – probably improving the design and adding animations, as the current one is rather “plain”. Regarding functional improvements – adding GraphQL, user structure and authentication are the three things which I have in mind. The current state of the web application doesn’t separate the search results based on users who have executed them, so anyone can view what the previous searchers were.

As a person with over 10 years of coding experience with different technologies behind their back, I want to tell you the following thing – software engineering is not only science, but art as well, art with which you can spring your ideas and visions into reality and create all sorts of product(s) and the beauty of it is the process of “trial-error”, which I’ve learned to embrace throughout the years. What’s scarier than failing to achieve the requirements and submit your tasks on time is losing motivation and interest.

Literature Used

1. Microsoft Docs – .NET
2. Microsoft Docs – C#
3. Angular Docs
4. Mozilla Development Network Web Docs(MDN)