URL Shortener challenge

By Lorcan McAnallen

# Project overview

The goal of the project is to create a website that allows a user to enter a URL address and retrieve a shortened link that leads to the same address.

The website can be used by anybody without an account but their links are only valid for one year, this is to encourage the user to create an account if they wish to use the links long term. With an account the user can add permanent links and can view the links they have made.

The user accounts are created by a basic email and password. The email confirmation is currently disabled as its outside the scope of this challenge.

The temporary link and the 1-year expiration date is in place to prevent a situation down the line where there are millions of link’s in the data base, this could reduce speed in producing new links as either the website or data base has to ensure the link is unique. Also, it is an encouragement to user to create an account to use permanent links.

This project was created with the ASP.net core framework 3.1.

# Functional requirements

|  |  |  |
| --- | --- | --- |
| **FR** | **Name** | **Description** |
| FR 1 | Create temporary link. | A user with no account can go to the main page |
| FR 2 | Crete an account | The user should be able to create an account. |
| FR 3 | Login | The user should be able to login to a create account. |
| FR 4 | Crete a permanent link. | The user can create a permanent link. |
| FR 5 | Viewing/deleting links. | The logged in users can view/delete their links. |
| FR 6 | Searching for links. | The logged in user can search their links. |

# Framework

The ASP.net core framework was chosen for this project due to familiarity with developing .net core on local desktop applications. Also, the ASP.net core framework is open source and cross platform.

The ASP.net core framework also has a range of auto generated functionality that saved development time and provides strong security and adaptability.

An example would be the user accounts. The functionality for registering, logging in and managing the current user is handled by framework generated code.

# Database

The data base uses Entity Framework Core to manage the saving and loading of the users and short links objects.

The Entity Framework Core uses the object-relational mapping (ORM) tool which allows for handling database objects without writing SQL manual.

The objects relation between the links and their users is currently handled on the C# side but could be moved to the database for more rigid relations between uses and their multiple links. Currently if the link object stores a user ID then the expiration date is set to the furthest DateTime possible. If there is on user ID then the expiration date is the creation plus one year.

On the data base side the deletion of expired links is handled by a stored procedure. This procedure could be run by an SQL agent and set to run a daily basis. This would have to be set up on the server that the website is hosted on.

The database has multiple migration files to run for setting up the tables for the user and link objects. There is also a bak file with some default data for testing the site.

# Layout of code

The home controller that is generated as part of the ASP.net core generated code is still used as the home page. Instead of removing the pages and controller, the pages are adapted and there are calls to the short links controller from the home page.

The pages seen by the unregistered user are from the home controller, while the short link controller manages the pages the user sees when logged in. An example would be that the unregistered user would see the home index page, while the search function is limited to logged in users and is handled by the short links controller.

The links are sent through specific endpoint that is the host followed by “/l/” then the unique value for the short link.

The links passed through the endpoint and then the database is checked for a matching short link and its corresponding link entered by the user.

# Version Control

The project is backed up on GitHub using get. Committing was done using the git plugin in visual studio 2019.

Another option for source control I could have used would have been team foundation but it is a paid service and I have more experience with Git.

The database does not transfer across with the files in git but the migrations files allow for quick and easy local setup. Also the bak file is also transferred for restoring a database.