The Impact of the Evolution of Operating Systems on Older Web Applications

António Godinho[1(✉)], José Rosado[2,3], Filipe Sá[2], and Filipe Cardoso[3,4]

[1] Coimbra Business School, Polytechnic Institute of Coimbra, Quinta Agrícola - Bencanta, 3045-231 Coimbra, Portugal
agodinho@iscac.pt

[2] Coimbra Institute of Engineering, Polytechnic Institute of Coimbra, Rua Pedro Nunes - Quinta da Nora, 3030-199 Coimbra, Portugal
{jfr,filipe.sa}@isec.pt

[3] INESC Coimbra—Instituto de Engenharia de Sistemas e Computadores de Coimbra, Rua Sílvio Lima, Pólo II, 3030-790 Coimbra, Portugal

[4] Escola Superior de Gestão e Tecnologia, Politécnico de Santarém, Complexo Andaluz, Apartado 295, 2001-904 Santarém, Portugal
filipe.cardoso@esg.ipsantarem.pt

Abstract. At the beginning of 2020, the major browser-developing companies announced that newer software versions no more extended support for older TLS, 1.0 and 1.1. A warning message was displayed in older versions; the user could override it and enter the website. After implementing the deprecation of TLS 1.0 and TLS 1.1, the users can no longer enter those websites. It's becoming more unusual for websites to exist for over ten years and keep active, but there are legacy web platforms where the cost of updating an older platform may need to be revised.

The Microsoft .NET Framework has been used for almost twenty years and is supported by Microsoft Windows operating systems. In the last years, with the development of .NET Core and the release of .NET 5, Microsoft no longer develops ASP.NET Web Forms Framework. It's expected that existing web platforms will not run on newer operating systems from Microsoft and should be replaced and removed from active systems.

Keywords: ASP.NET Web Forms · Visual Basic .NET · Deprecation of TLS 1.0 and TLS 1.1 · Windows Server 2003 · IIS 6.0 · Operating System

1 Introduction

Over time, software applications and the underlying technologies or programming languages they depend on naturally undergo a process of aging, eventually becoming outdated [1]. This progression can result in a deterioration of the performance of these applications and, in certain instances, vulnerabilities [2,3].

P. J. Coelho et al. (Eds.): GOODTECHS 2023, LNICST 556, pp. 17–29, 2024.
https://doi.org/10.1007/978-3-031-52524-7_2

Additionally, these applications can be tethered to particular operating systems or technology versions, making upgrades infeasible [4].

Concurrently, as this technological aging occurs, development teams consistently confront difficulties associated with the acquisition and departure of team members [5,6]. The dynamics within these teams often revolve around the specialization of individuals in diverse technologies, a necessity driven by the rapid evolution of the technological landscape [7]. This aspect also needs to be revised to upgrade the existing systems.

Because of the economic consequences, numerous institutions and companies across various sectors offer outdated applications that operate on obsolete operating systems and technologies. It is essential to examine and assess this impact to determine whether the cost of replacing these legacy applications outweighs the benefits, especially when compared to creating new, supported applications through rewriting and redesigning [8]. All resources implicated in the process must be taken into account prior to undertaking such a venture. Nevertheless, this remains a crucial aspect within software development and system administration tasks, as the connection between outdated applications and the seriousness of vulnerabilities and glitches is significant [9,10].

Internet Information Services (IIS) 6.0 and Windows Server 2003 support ended in July 2015 [11]. Without updates, newer and modern encryption protocols used by Hyper Text Transfer Protocol Secure (HTTPS) cannot be used.

Browser developers like Google, Mozilla, Microsoft, and Apple made similar announcements that they were deprecating TLS 1.0 and TLS 1.1 around the spring of 2020 [12–15]. In April 2020, Google announced the release of Google Chrome 81, when the drop would happen, and almost by the same time, Mozilla released Firefox 74. Due to the COVID-19 pandemic, these changes were delayed for an undetermined time. Dropping the support during the pandemic could cause issues with critical government or healthcare sites that use somewhat outdated encryption protocols [16], and even Microsoft postponed the deprecation of TLS 1.0/1.1 for Microsoft 365/Office 365 for the same reasons.

In the middle of July, those changes were finally implemented, and website access was blocked automatically after the browsers updated to the latest stable version.

Almost in the same period, Microsoft announced the release of .NET Core 5 and, most importantly, the next release of their .NET family, .NET 5. With this new version, Microsoft will stop the development of the ASP.NET Web Forms Framework, making this technology obsolete [17]. Microsoft will continue to support .NET Framework (which ASP.Net WebForms is part of) for some time since much of its functionality is based on the core .Net Framework, even on modern Operating Systems like Windows 11 and Windows Server 2022 [18].

Nevertheless, the ASP .NET Web Forms Framework is a successful technology for developing web platforms. The framework has a set of built.in controls that provide visual components or functionalities to web applications.

After the time limit, a working platform was available, with all the main features working. The document is organized as follows: After this Introduction,

the case study is presented in Sect. 2. The process for rebuilding the application is presented in Sect. 3. The results are shown in Sect. 4. For last, Sect. 5 gives the Conclusions.

2 Case Study - A Web-Based File Manager

The study was conducted on a higher education institution with a website that provided a file manager for all the institution's services. This application allowed all the regular file manager operations: upload, rename, delete, and, most importantly, provide the public URL to access the uploaded documents. This website was developed on ASP.NET Web Forms, using Visual Basic .NET 2.0 for the code behind it. The website serves as an internal tool for more than 3,500 users and is accessible to all visitors who download files from the site.

The website can be separated into three different components:

1. The login web page allows user authentication on Windows Active Directory.
2. The web page provides file management.
3. The file and directories repository, where the users upload the documents, and the folder organization.

2.1 Application End of Life

The institution workers have their systems configured to allow the browsers to be automatically upgraded. As refereed in Sect. 1, when browsers were updated in July, the users could no longer access the file manager, which posed a problem since all public documents, from forms for teachers and students to institution communications, weren't accessible. It was a problem that had to be fixed as soon as possible.

2.2 Upgrade Application or Develop a New Website

The original server was running Windows Server 2003 operating system, with IIS 6.0, and the website was over ten years old. The website could be moved to a new server with a recent version of IIS. Still, if the application was just copied to a newer server (running Windows Server 2019 or 2022), this would require setting up an Application Pool on .NET v2.0 Classic on IIS for the website to run. An Application Pool on .NET v2.0 Classic processes the requests in the app pool by using separate processing pipelines for IIS and ISAPI [19]. IIS7 and IIS8 were re-architected with the superior and faster Integrated Mode pipeline but retained the "Classic" mode for compatibility [19] (Fig. 1).

While the Application Pool on .NET v4.5 may run as an integrated pipeline, IIS and ASP.NET will take advantage of the improved features of IIS 7.0 using only one process. It will mean that the web platform will still be old and won't take advantage of all the potential of the newer Operating System and IIS. It wouldn't be more than a band-aid because some features were requested, and some minor bug fixes were required. Setting the application pool as v2.0 classic would still be the most straightforward task, as shown in Fig. 2.

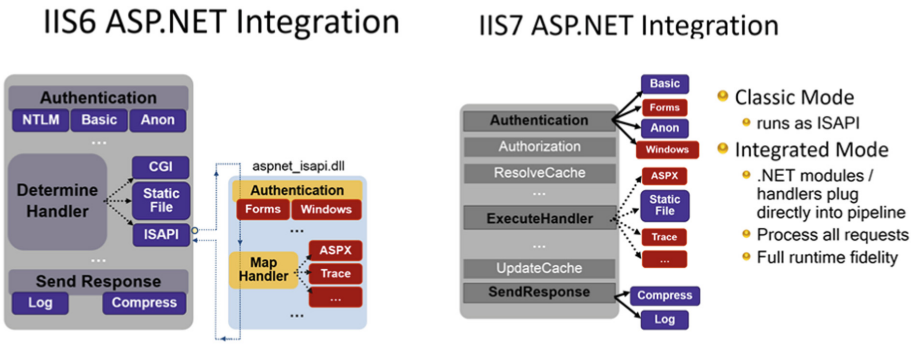

Fig. 1. Application Pools on IIS

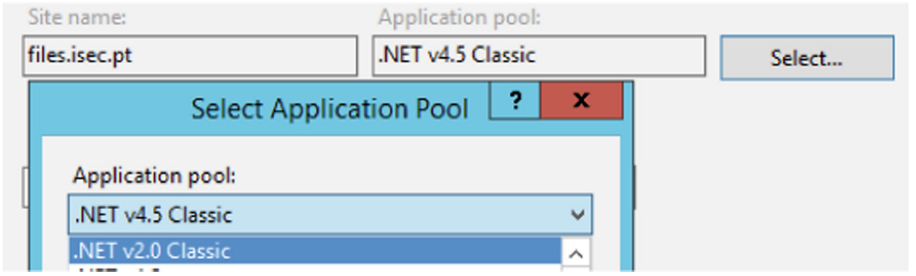

Fig. 2. Setting the Application Pool on IIS

2.3 VB vs C#

The update of the existing web platform had a significant problem. While Visual Basic .NET has not been deprecated, and there are still developers and applications that use it. Microsoft has continued to update and enhance Visual Basic .NET alongside C# in various versions of the .NET platform. However, C# has received more attention and new features in recent years, and it has become the primary language of choice for many developers working with the .NET framework. Microsoft has introduced many cutting-edge features and enhancements in C# to keep it competitive in the modern software development landscape. Microsoft has not abandoned Visual Basic .NET, but C# has become the dominant language in the .NET ecosystem, with more emphasis on its development and features.

Also, Visual Basic may not support future features of .NET Core that require language changes. Due to differences in the platform, there will be some differences between Visual Basic on .NET Framework and .NET Core [20]. Also, the webpage's code was complex due to limitations of the ASP.NET Web Forms platform version used to develop the website. One widespread mistake by web developers in the early 2000s was using inline styles. Inline styles don't separate content from design, with many locations to check when changing an element

property. It also added complexity to the task of modernizing the platform to a new and more appealing design [21]. One crucial point to consider is that Microsoft no longer develops ASP.NET Web Forms and may be obsolete at any time soon. Other technology should be used if a new application is created from scratch.

2.4 IT Team Coding Expertise

The IT service staff at the institution had gradually lost their expertise in developing applications using Visual Basic (VB.NET). Over the past decade, as team members left and newcomers joined all development work shifted to C#. Consequently, Visual Basic .NET became an unfamiliar technology. Introducing a new coding language to the team was a challenging decision. It involved dealing with a learning curve, where team members required time to become proficient in the new language. Additionally, there were considerations regarding adopting specific development tools, libraries, and frameworks. This transition demanded resources and attention away from ongoing projects, and committing to a language fading into obsolescence seemed impractical.

Considering all these factors, the goal was to update the existing code from VB.NET to C# to a newer ASP.NET Web Forms running .NET Framework 4.8. The objective was to have a working website with only one developer involved within three days. A new platform should be planned and developed from scratch if this objective isn't achieved within the time limit.

3 Rebuild the Application

The existing web application is made of only two web pages. The first is the login page for user authentication on the Windows Active Directory, and the other web page is the file manager per se, with all the file and directory management features.

3.1 Tools

Developing reactive web pages or Responsive Web Design (RWD) from source is a complex process. Several frameworks provide generic functionality with already written modules and tailored components created traditionally. Web developers use front-end frameworks for implementing Cascading Style Sheets to facilitate the development of RWD [22]. Bootstrap is the most popular HTML, CSS, and JavaScript framework for developing a responsive and mobile-friendly website. It is free to download and use. It is a front-end framework for easier and faster web development [23]. Bootstrap [24] uses a grid system that allows a fluid grid system that appropriately scales up to 12 columns as the device or viewport size increases [23], allowing the implementation of the RWD. Bootstrap framework was chosen as the base for CSS formatting and visual elements to speed up the development process for this web platform. The JavaScript Framework JQuery and fontawsome icons were also added to provide visual details and user interaction.

3.2 Day One - Move from Visual Basic to C#

The first step was creating an Empty ASP.NET Web Forms solution, choosing C# for its code. The second step was the initial configuration of the application. This kind of application configuration is made by editing the web.config file, which defines the behavior of ASP.NET applications [25]. ASP.NET applications come with a default Web.config file that can be edited with the working IDE, in this case, and for this type of technology, it was Visual Studio.

Generally, Web.config files contain comments that make editing the file self-explanatory [25]. For this web application, and since the user authentication is made using Windows Active Directory, it was required to define a Connection String with the IP address and port of the LDAP server, with the credentials of a user with AD browsing access. This Connection String was then used by a platform's component, the Membership Provider, responsible for user management on the web application. Inside Authentication Mode Forms, this component allows the framework ASP .NET to authenticate users on the Windows Active Directory without needing extra coding.

3.3 Login Page

The first web page developed was the login page. It is a straightforward standard web page with an ASP .NET Web Forms Login Control. This framework component provides user interface elements for logging in to a website [26]. The control generates two input text boxes for the username and password, a "Remember me" option checkbox, and a submit button. This web page uses other controls from the framework, validating both text boxes to make both fields (username and password) required. Using the configuration defined on the web.config, this page is set as the default login web page. When an unauthenticated user tries to access a private area, it will be redirected to this web page.

3.4 File Manager Page

The second web page is the website's core, with all the directory and file management functionalities. With the slim timeline, rebuilding the whole application from scratch wasn't the objective, and the existing code would be used as a starting point. Again, to speed up the development process, the existing visual basic code was converted to C# with the help of an online tool from a company called Progress [27]. This company has a set of long-time successful tools for ASP.NET and provides this free service on its website. The output generated from the converter was incomplete and had many problems, but it served its purpose, Fig. 3. It was challenging because we started with a base code made by another developer. The next step was to analyze the generated code by the converter to understand all the features contained.

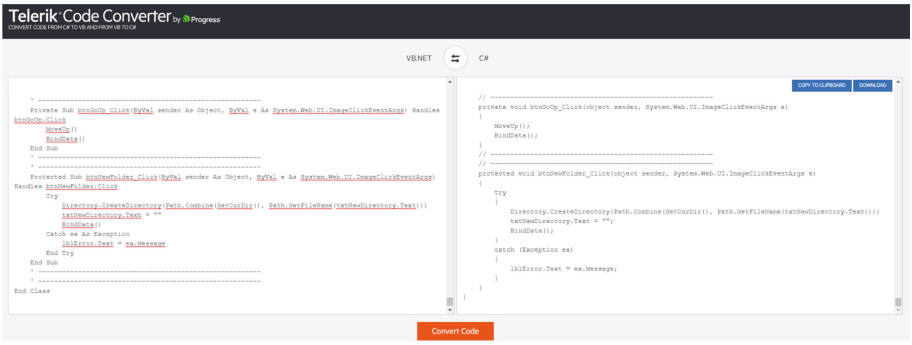

Fig. 3. Telerik code converter from visual basic to C#

Something stood out right from the beginning. The authorization options related to access levels were hard coded on the code behind the webpage. It used switch case statements with usernames or Services/Departments. It shouldn't be the way to restrict user access to specific areas, so the code blocks that provided this feature were removed entirely. To provide user authorization at a later stage of development, the framework's Authorization Manager Role Provider configuration using SQL Server was added to the web.config file.

The work developed on the first day allowed domain users to log in, access files, and do directory listing. Browsing through the directories tree failed to move up or down the hierarchy, as shown in Fig. 4.

3.5 Day Two - File/directory Operations and Design

With the login page already working, the focus was the file upload functionalities. The second page is the core of the application. The page is constituted of two sections side by side.

Left Section
Block with the information and file operation, composed of two different blocks. The one on the top has the following elements:

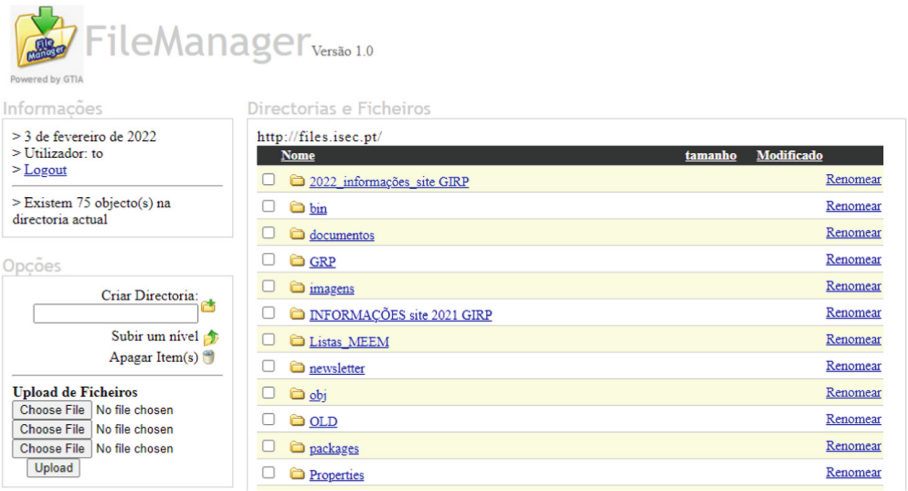

Fig. 4. Running website on a new server

– Information about the logged-in user.
– Logout button.
– A summary of the number of objects in the current directory.

The block below was the file and directory options, with multiple controls and components:

– A input textbox and button to create new directories.
– A button allows browsing up one level on the directories tree.
– Button to delete selected files and/or directories.
– Two file upload components and respective buttons permit the user to upload up to two files simultaneously to the current directory.

Right Section, the Directory Contents
This section is composed of a table with all directory files and sub-folders. They are marked with icons identifying folders and files using the font awesome icon pack. There are also columns with file size, creation/change date, and buttons to rename and obtain the public link. A checkbox on each line allows selecting a row to apply a single action to multiple rows. The page table and how the elements are disposed of can be seen in Fig. 5.

Developing the Second Page
File and directory are the main features of a file manager. At this point, only delete and file upload was working. The folder navigation needed to be addressed. When a user doubles and clicks a folder, the method that processed the click event fails, and the table with the current folder content will become empty. The

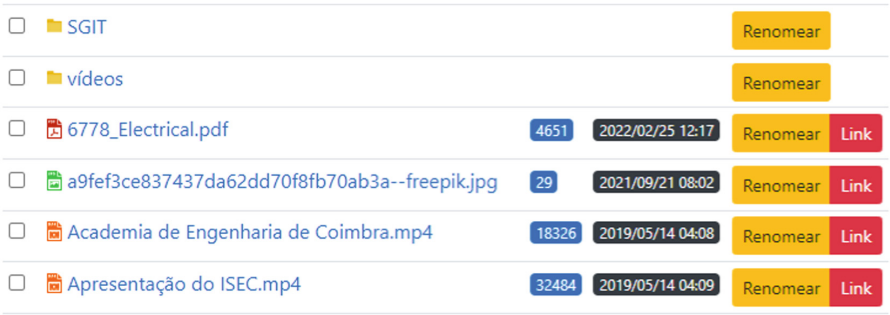

Fig. 5. Table with directory contents

original version used ASP.NET Web Forms Repeater Control to render the table with the folder contents. This control was changed to a GridView Control, and all the methods that list the files and directories were redone.

```
protected DataTable GetFiles()
{
  String dirMae = GetCurDir();
  DirectoryInfo dirInfo = new DirectoryInfo(dirMae);
  FileInfo[] info = dirInfo.GetFiles();
  DirectoryInfo[] dirs = dirInfo.GetDirectories();
  DataTable dt = CreateDataSource();
  DataRow dr;
  ...
  curr = GetCurDir();
  foreach (var dir in dirs)
  {
    aux = curr + @"\" + dir.Name;
    aux = aux.Replace(@"\\", @"\").ToUpper();
    location = InvisibleDirs.IndexOf(aux);
    if (location < 0)
    {
        dr = dt.NewRow();
        dr["filename"] = dir.Name;
            ...
        dr["type"] = "0";
        dt.Rows.Add(dr);
    }
  }
  string auxExt;
  foreach (var file in info)
  {
    dr = dt.NewRow();
    dr["filename"] = file.Name;
    ...
    dr["type"] = "1";
```

```
32      dt.Rows.Add(dr);
33    }
34    dt.AcceptChanges();
35    return dt;
36  }
```

Listing 1.1. Method to get dir contents, directories and files

The source data that feeds the GridView is now an instance of a DataTable object that is built in code behind, shown on Code 1.1.

There were several bugs across all the main features. The main task during the day was to correct all non-working features, with several exceptions sent to the web server while running the web platform.

4 Results

After the time limit, a working platform was available, with all the main features working. Some lingering problems will require significant code changes to fix. For example, the application used a session var that stores the current directory. This var was changed when a user entered a sub-folder by clicking on one of the folders on the table, as shown in Fig. 5. The session var could also be changed by pressing the "up one level" button, which allows changing to the parent directory. The problem with using a session var was that if the user used the back button from the browser instead of using the navigation buttons, the web page stopped working because the directory shown on the web page did not match the one on the session var. The multiple file upload failed when the total size exceeded the maximum defined on IIS. ASP.NET doesn't allow complete customization of its components. It led to some violations of the CSS framework Bootstrap, which resulted in some elements not being fully responsive. The results were satisfactory due to urgency, but several elements must be remade.

5 Conclusions

Upgrading older ASP.NET Web Forms applications from visual basic to C# for small websites/applications is possible, and it can be sustainable. But there are different aspects to be considered when trying to do this kind of update/upgrade. Other persons, over time, usually develop old applications with extended longevity; another issue may be existing documentation of these types of old web platforms. There are a couple of issues that will increase the understanding of code. Still, if the update/upgrade is decided, there are valuable tools that help convert the code from one language to the other to speed up the development. Still, this conversion results in extremely buggy code, requiring a deep analysis of the generated code, and on some occasions, it may take the same amount of time as developing a web page from scratch.

Over the years, web design has evolved, providing users with more friendly and appealing User Interfaces (UI). This approach can also make the UI modernization process more difficult because it is more connected to the old layout being used.

On more extensive applications, converting old code from a different language and using that generated code as a basis for an upgraded version of a web platform may be even more confusing than help full, and this approach should be disarted.

There is also the question of the longevity of the ASP.NET Web Forms platform. A newer technology should be used if the application has to be made from scratch. Choosing from the Microsoft ecosystem, more recent frameworks like .NET Core 3.1 and .NET 6/7 are considered LTS, but when this work was written, the last version was .NET 6 and should be chosen. There are also other options with different technologies. On the Microsoft ecosystem, there is Blazor, and on JavaScript full-stack Frameworks, there are several options like React, Angular, Vue, or Node.

The choice to upgrade or develop a new application has to be made case by case, considering the technologies where the IT team's skills are vital. Also, the time and effort required to upgrade an existing application following the path chosen in this work are to consider the complexity of the current web platform and probably the knowledge of the team of the application code and functionalities. The sum of all these factors has to be compared to the cost of developing a new web application.

Acknowledgements. This work is funded by FCT/MEC through national funds and co-funded by FEDER—PT2020 partnership agreement under the project **UIDB/50008/2020**. This work is partially funded by National Funds through the FCT - Foundation for Science and Technology, I.P., within the scope of the projects UIDB/00308/2020, UIDB/05583/2020 and MANaGER (POCI-01-0145-FEDER-028040). Furthermore, we would like to thank the Polytechnics of Coimbra and Santarém for their support.

References

1. Fürnweger, A., Auer, M., Biffl, S.: Software evolution of legacy systems. In: ICEIS 2016, p. 413 (2016)
2. Zerouali, A., Mens, T., Robles, G., Gonzalez-Barahona, J.M.: On the relation between outdated docker containers, severity vulnerabilities, and bugs. In: 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER), Hangzhou, China, pp. 491–501 (2019). https://doi.org/10.1109/SANER.2019.8668013
3. Narayana Samy, G., Ahmad, R., Ismail, Z.: Security threats categories in healthcare information systems. Health Inform. J. **16**(3), 201–209 (2010). https://doi.org/10.1177/1460458210377468
4. Zerouali, A., Cosentino, V., Mens, T., Robles, G., Gonzalez-Barahona, J.M.: On the impact of outdated and vulnerable Javascript packages in docker images. In: 2019 IEEE 26th International Conference on Software Analysis, Evolution and

Reengineering (SANER), Hangzhou, China, pp. 619–623 (2019). https://doi.org/10.1109/SANER.2019.8667984
5. Savor, T., et al.: Continuous deployment at Facebook and OANDA. In: Proceedings of the 38th International Conference on Software Engineering Companion (2016)
6. Goodman, E., Loh, L.: Organizational change: a critical challenge for team effectiveness. Bus. Inf. Rev. **28**(4), 242–250 (2011)
7. Acar, Y., Stransky, C., Wermke, D., Weir, C., Mazurek, M.L., Fahl, S.: Developers need support, too: a survey of security advice for software developers. In: 2017 IEEE Cybersecurity Development (SecDev), Cambridge, MA, USA, pp. 22–26 (2017). https://doi.org/10.1109/SecDev.2017.17
8. Christensen, C.M.: The innovator's Dilemma: When New Technologies Cause Great Firms to Fail. Harvard Business Review Press (2013)
9. Hossain, M.M., Fotouhi, M., Hasan, R.: Towards an analysis of security issues, challenges, and open problems in the internet of things. In: 2015 IEEE world Congress on Services. IEEE (2015)
10. Habibzadeh, H., et al.: A survey on cybersecurity, data privacy, and policy issues in cyber-physical system deployments in smart cities. Sustain. Cities Soc. **50**, 101660 (2019)
11. Microsoft. Internet information services (IIS) - microsoft lifecycle. Microsoft Lifecycle—Microsoft Docs. https://docs.microsoft.com/en-us/lifecycle/products/internet-information-services-iis. Accessed 15 Nov 2021
12. Benjamin, D.: Modernizing transport security. Google Online Security Blog (2018). https://security.googleblog.com/2018/10/modernizing-transport-security.html. Accessed 18 Nov 2021
13. Thomson, M.: Removing old versions of TLS. Mozilla Security Blog (2018). https://blog.mozilla.org/security/2018/10/15/removing-old-versions-of-tls/. Accessed 18 Nov 2021
14. Pflug, K.: Modernizing TLS connections in Microsoft edge and internet explorer 11. Microsoft Edge Blog (2020). https://blogs.windows.com/msedgedev/2018/10/15/modernizing-tls-edge-ie11/. Accessed 18 Nov 2021
15. Wood, C.: Deprecation of legacy TLS 1.0 and 1.1 versions. WebKit (2018). https://webkit.org/blog/8462/deprecation-of-legacy-tls-1-0-and-1-1-versions/. Accessed 18 Nov 2021
16. Laflamme, R.: Chrome 81 features and release date. Insightportal (2020). https://www.insightportal.io/news/all-news/chrome-81-beta-features-and-release-date. Accessed 18 Nov 2021
17. [MSFT], R., Rich Lander [MSFT] Program Manager, 6, A., Asthana, A., 6, S., [MSFT], S., Cheong00. Introducing.NET 5 (2021). https://devblogs.microsoft.com/dotnet/introducing-net-5/. Accessed 28 May 2022
18. Microsoft. Microsoft.NET Framework - Microsoft Lifecycle (n.d.). https://docs.microsoft.com/en-gb/lifecycle/products/microsoft-net-framework. Accessed 28 May 2022
19. Hanselman, S.: Moving old apps from IIS6 to IIS8 and why Classic Mode exists (2013). https://www.hanselman.com/blog/moving-old-apps-from-iis6-to-iis8-and-why-classic-mode-exists. Accessed 15 Nov 2022
20. .NET Team. Visual basic support planned for.NET 5.0 (2020). https://devblogs.microsoft.com/vbteam/visual-basic-support-planned-for-net-5-0/. Accessed 15 Nov 2022
21. Kyrnin, J.: Avoid inline styles for CSS design. ThoughtCo (2020). https://www.thoughtco.com/avoid-inline-styles-for-css-3466846. Accessed 24 Nov 2021

22. Shenoy, A., Prabhu, A.: CSS Framework Alternatives: Explore Five Lightweight Alternatives to Bootstrap and Foundation with Project Examples. Apress (2018)
23. Gaikwad, S.S., Adkar, P.: A review paper on bootstrap framework. IRE J. **2**(10), 349–351 (2019)
24. Mark Otto, J.: Bootstrap (2022). https://getbootstrap.com/. Accessed 17 Nov 2022
25. HaiyingYu. Edit configuration of an ASP.NET application - ASP.NET. Edit configuration of an ASP.NET application - ASP.NET—Microsoft Docs (n.d.). https://docs.microsoft.com/pt-PT/troubleshoot/developer/webapps/aspnet/development/edit-web-config. Accessed 4 Mar 2022
26. Anderson, R.: Login class (system.web.ui.webcontrols). (System.Web.UI.WebControls)—Microsoft Docs (n.d.). https://docs.microsoft.com/en-us/dotnet/api/system.web.ui.webcontrols.login?view=netframework-4.8. Accessed 7 Mar 2022
27. Code converter C# to VB and VB TO C#. Telerik. (n.d.). https://converter.telerik.com/. Accessed 29 Nov 2021