Developer guide

# Contents

[Contents 2](#_Toc54263236)

[Document Information 3](#_Toc54263237)

[Change History 3](#_Toc54263238)

[Distribution 3](#_Toc54263239)

[Acronyms 3](#_Toc54263240)

[Introduction 4](#_Toc54263241)

[Starting off 5](#_Toc54263242)

[Coding in the Open 7](#_Toc54263243)

[GDS Design system 8](#_Toc54263244)

[Developers Charter 9](#_Toc54263245)

[Accessibility 10](#_Toc54263246)

[Application Security 11](#_Toc54263247)

[Branching Strategy 12](#_Toc54263248)

[Deployment Process 13](#_Toc54263249)

[Design Patterns 14](#_Toc54263250)

# Document Information

|  |  |  |  |
| --- | --- | --- | --- |
| Title | Developer guide | **Version** | 0.1 |
| Author | Michael Stevenson | **Date** |  |
| Reviewed By |  | **Review Date** |  |
| RM Reference |  | **Status** |  |

# Change History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ver. No. | Ver. Date | Revised By | Description | Reviewer | Status |
| 0.1 | 21/10/2020 | Michael Stevenson | Initial Document |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# Distribution

|  |  |  |
| --- | --- | --- |
| Name | Position | Contact Details |
|  |  |  |
|  |  |  |
|  |  |  |

# Acronyms

|  |  |
| --- | --- |
| DOF | Department of Finance |
| DSS | Digital Shared Services |
| EDD | Enterprise Digital Development |
| ESS | Enterprise Shared Services |
| NICS | Northern Ireland Civil Service |

# Introduction

This document will serve as a starting guide to developers working in the **new world** of development in **Enterprise Digital Development**. This will be a guide whilst comprehensive but will not be exhaustive. Below are the current technology stacks for the **old world** (how we used to develop) and the **new world** (the future state were we are going.

**Old World**

C#, Sql server

DotNet Core

MVC

IT Assist on premises web servers

GitLab

Jenkins

Sonar Qube

**New World**

C#, MySql

DotNet Core

MVC

Gov.UK PaaS – Hosted applications in the Gov.UK cloud backed by cloud Foundry.

GitHub

Circle CI

Sonar Cloud

WhiteSource Bolt

Git Guardian

Cloud Foundry

Amazon AWS

# Starting off

Starting you will have to install the below on your nigov machine and alter some settings to get around the corporate proxy.

**Accounts**

You will need set up these accounts:

* PaaS Account
* GitHub account
* AWS account
* Circle CI account (you can do this with your GitHub Account)
* MSDN license to download Visual Studio

**Software to install**

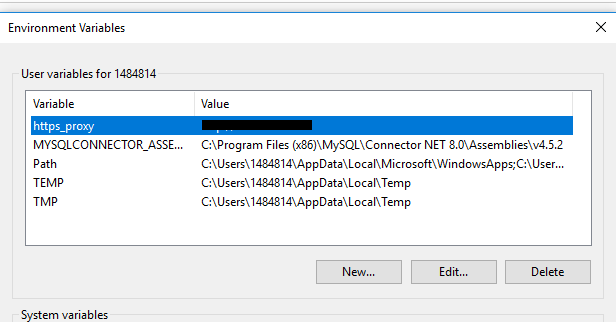
* Cloud Foundry CF CLI – <https://github.com/cloudfoundry/cli#downloads>
* Visual Studio
* Sonar Lint Visual Studio plugin -  <https://www.sonarlint.org/visualstudio/>
* Visual Studio Code - <https://code.visualstudio.com/download>
* MySQL workbench and MySQL - <https://dev.mysql.com/downloads/windows/installer/8.0.html>
* Retire.js Chrome plugin (this checks for vulnerable libraries) - <https://chrome.google.com/webstore/detail/retirejs/moibopkbhjceeedibkbkbchbjnkadmom?hl=en>
* Axe accessibility Chrome plugin - <https://chrome.google.com/webstore/detail/axe-web-accessibility-tes/lhdoppojpmngadmnindnejefpokejbdd>
* Chrome Lighthouse Auditing tools - this is built into Chrome

**How to install if software does not install normally**

Move install files to Program files on C: drive and run as administrator or as admin.

**Proxy settings**

You may need to update your environment variables on your pc to allow CF CLI and GitHub to work. Set a new variable called https\_proxy and set it to the NICS proxy.



# Coding in the Open

For most of our project’s we will be [coding in the open](https://gds.blog.gov.uk/2012/10/12/coding-in-the-open/). What this means in practice is that at the start of a project we will set our repository as **private** i.e. not accessible to the internet and outside world. Then when ready the project repository will be set to **public** and open sourced to the public. The code is funded by the taxpayer and is public opened so it is only right that it is made available to the general populace.

**Important:**

With this in mind we have to be careful not to merge **user secrets** into the project repos.

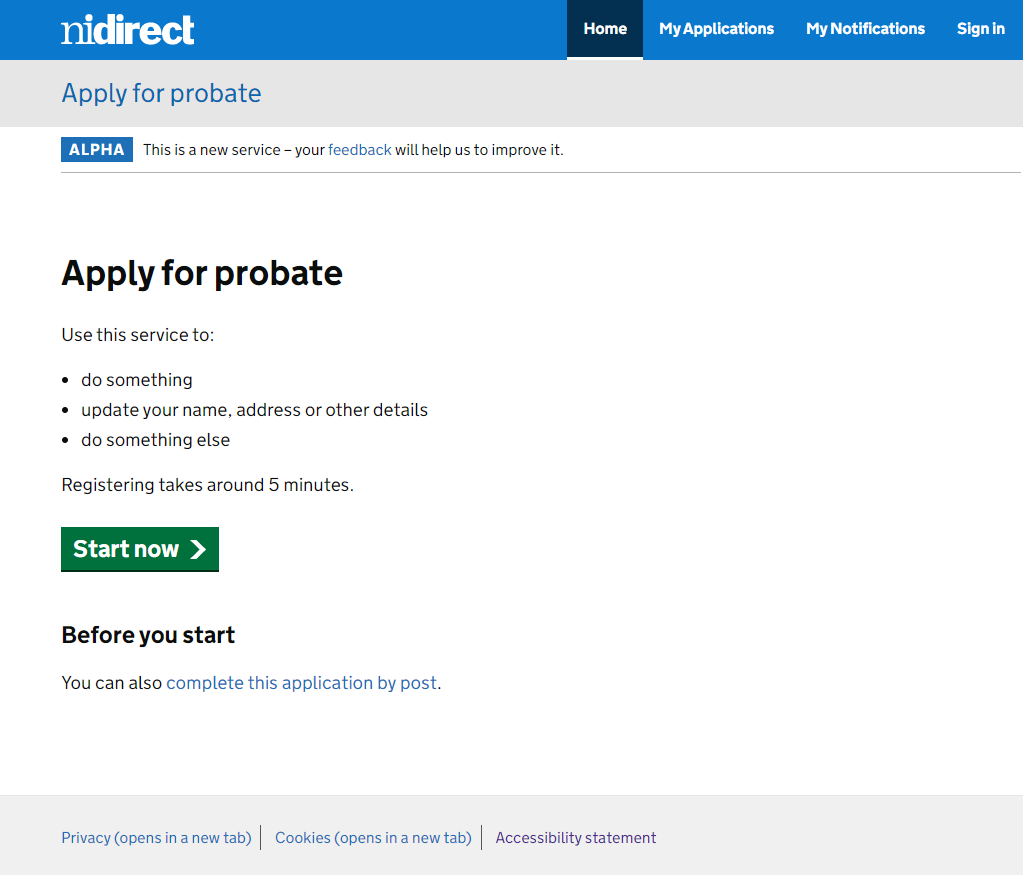
User Secrets are database connections, api keys, AWS keys, PaaS Credentials and so on.

In order to work locally with secrets on our local machine in Visual Studio we need to use the secrets manager in order to do this. The below link provides the info on how to do this (it is easy enough). Git Guardian will pick up any policy breaks before the code is merged into main – more on this later.

**User Secrets Manager** - <https://docs.microsoft.com/en-us/aspnet/core/security/app-secrets?view=aspnetcore-3.1&tabs=windows>

# GDS Design system

For the UI look and feel we are using the [GDS design system](https://design-system.service.gov.uk/). We will be using the NI Direct style for the header and footer but the main body design will use the GDS Design system. This comes as a nuget / npm package for our templates.



# Developers Charter

This is just an idea to help us self-regulate our software quality and the products we deploy. We should endeavour to hone our craft and set the bar high in terms of software quality, consistency, accessibility, security and availability. The charter asks everyone to be:

* I shall do no harm intentionally
* Treat all artefacts as sacred
* Do not release vulnerable code
* Make accessibility a key feature so to make Northern Ireland service available to all of the society
* Be mindful of security
* Be open to change and new technologies
* Share everything with everyone
* Be an ambassador for our craft
* Perpetuate my craft
* Be Humble
* Continue to enjoy learning

# Accessibility

We will make our applications WCAG 2.1 AA compliant in line with legislation. We will use the below tools when completing each story to make sure our work is accessible. Our public facing sites will have to be bullet-proof as we want all of the society to be able to use these systems easily. We have to use a variety of tools as current tools such as Wave only pick up around 30-40% of all issues.

**Sonar Cloud**

This will pick up html accessibility issues in our CI CD pipeline.

**Chrome plugins**

**Retire.js** Chrome plugin (this checks for vulnerable libraries) - <https://chrome.google.com/webstore/detail/retirejs/moibopkbhjceeedibkbkbchbjnkadmom?hl=en>

**Wave** - <https://chrome.google.com/webstore/detail/wave-evaluation-tool/jbbplnpkjmmeebjpijfedlgcdilocofh>

**Axe** accessibility Chrome plugin - <https://chrome.google.com/webstore/detail/axe-web-accessibility-tes/lhdoppojpmngadmnindnejefpokejbdd>

Chrome **Lighthouse** Auditing tools - this is built into Chrome

**Aria checker** – <https://chrome.google.com/webstore/detail/aria-validator/oigghlanfjgnkcndchmnlnmaojahnjoc>

**AA accessibility checker**

<https://achecker.ca/checker/index.php>

Use this site to copy html source to check for accessibility errors.

# Application Security

We will make our applications as secure as possible using best practices. Sonar cloud will pick up some security issues before it enters the main branch. We will also use retire.js and White Source Bolt to discover vulnerabilities in third party software that we may use.

**OWASP** etc – these flaws our picked up by Sonar Cloud in the CI / CD pipeline.

**Git Guardian**

Git Guardian is used to make sure that we do not merge secrets into the public repo in GitHub. This is scanning all the time and is included in our CI/CD pipeline so that secrets are not merged in. This gives us a level of assurance of the security of our applications.



**Penetration testing** – TBC

**Burpsuite / other security tools** - TBC

# Branching Strategy

We will use the [Microsoft branching strategy](https://docs.microsoft.com/en-us/azure/devops/repos/git/git-branching-guidance?view=azure-devops). This means that we will have only one branch for a project which is called “main”. Instead of forking the branch just clone the project on your local machine and create branches which you will push up and create a pull request.

Forking means you have to keep 2 repos up to date which is a bit time consuming.

**Creating a pull request** - <https://docs.github.com/en/free-pro-team@latest/github/collaborating-with-issues-and-pull-requests/creating-a-pull-request>

**Whenever a pull request is made assign it to a team member to review and merge in.**

# Deployment Process

We will use Circle CI to deploy our applications. In GitHub we have a workflow that works through a CI CD pipeline and processes some checks before merging to main.

**Preview branch**

The CI CD pipeline creates a preview version of the site whenever a merge is requested this allows the requestor the ability to check the changes.

**Merge accepted**

Once the merge is accepted the below checks are made to see if it can be merged into main and start the deployment process.

**Merge initiated** 🡪 Git Guardian Check 🡪 Whitesource Bolt check 🡪 Sonar Cloud Check 🡪 Circle CI Build 🡪 Preview Deployed 🡪 **Merge Accepted** 🡪 **Deploy application**

# Design Patterns

**Design patterns to use** – TBC

The general idea is that design patterns we use should be maintainable and suit the problem we are trying to solve in the first place. We should use the principle of SOLID.

**Useful resources**

Pattern in C# - <https://refactoring.guru/>

SOLID - [medium.com](https://medium.com/@mirzafarrukh13/solid-design-principles-c-de157c500425)