Hogeschool van Amsterdam

PLAT4MATION

Third party tool integration in a service based cloud ecosystem

Author:
Dries Meerman
500685616

Supervisor:
John Somers

March 30, 2018

Abstract

Plat4mation does consultancy and custom app creation on the ServiceNow platform. The development tools on the platform are not ideal, so the developers prefer third party tools. These are not integrated with ServiceNow and come a disadvantage. Code that is created has to be manually transferred to the platform. This is a tedious manual task, which can be error prone.

The current development process at Plat4mation will be further explained, to demonstrate the problems at hand.

Deployment processes are analyzed, to understand what happens when code is ready for deployment. So it can be compared to the way ServiceNow does deployments to other instances of the platform.

Cloud computing and the ServiceNow platform will be researched. Giving context into the requirements that will have to be met, for synchronization tools.

The tools that are used during development of ServiceNow apps. Are explained to show why developers work locally.

The ServiceNow platform offers an API, that allows external tools to edit records on the platform. A tool that leverages this API, can be used to bridge the gap between the local development tools and the ServiceNow platform.

Finally tools that have been researched in the report, will be compared to give an overview of the options.

Based on all the knowledge gained during the creation of this report, recommendations will be given to Plat4mation.

Contents

Introduction

Plat4mation is both a consultancy and development company. They offer project and service implementations. It is an official training partner of ServiceNow and offers trainings.

One of the services Plat4mation provides is application development on the ServiceNow platform. Plat4mation wants to improve the development process, by integrating third-party development tools into the development workflow.

The developers at Plat4mation already use the Webstorm IDE, however integration with the ServiceNow platform is lacking. Code that is created has to be manually added to the platform. This is a tedious task that lowers developer productivity and happiness.

To achieve that goal the following question will be researched and answered. "*How can front-end build tools be leveraged in combination with the ServiceNow REST API to speed up the development process when developing service portal applications?*"

The question will be answered by first looking at the development processes, and how ours can be improved.

Then cloud computing models will be researched to give more context, into working on the ServiceNow cloud platform. ServiceNow will then be explained, what kind of platform it is. How development on the platform takes place.

Followed by an analysis of development tools, that are used when creating applications on ServiceNow's portal framework.

This leads into a section about solutions to the integration problem. That answers the question about how all previously researched tools. Can be integrated in a a workflow where the code needs saved on a closed of platform.

Finally all the options that have been researched have to be evaluated. Looking at the strengths and weaknesses of the solutions to problems. This will be the basis for the recommendations about improving the development process.

1 Context

This chapter will give some context on the graduation assignment by explaining more about Plat4mation and what kind of company it is. It will also give a brief summary about the ServiceNow platform so readers not familiar with the platform, can have a better understanding when it is referenced throughout this report.

1.1 Plat4mation

The company Plat4mation was founded in 2013 so it is still a relatively young company. Plat4mation is an international company with its headquarter in Utrecht, the Netherlands. There is a second office is located in Mechelen, Belgium.

Plat4mation's focus lies in implementing the ServiceNow platform at customers. But also offers application development focused on the ServiceNow platform. The Plat4mation has got a number of applications in the ServiceNow store [1] , nine at the time of writing. While the store contains a total of 310 applications [2], nearly 3% of the applications on the platforms store have been created by Plat4mation.

Core values

- Growing Together
- Customer Minded
- The Extra Mile, The Extra Smile
- Continuously Learning
- Fun and Energetic

Following these core values has helped Plat4mation to grow rapidly in past few years. Plat4mation BE moved into a new office this past November (van Oijen, 2017). A part of the growth of Plat4mation can be attributed to the fast growth [3] of the ServiceNow platform. The growth of platform leads

[1] See figure 15 in the appendix.
[2] Time of writing is November 2017
[3] "*ServiceNow's 2016 sales exploded 38% to $1.4 billion*" (Krause, 2017)

to a large market for professionals on the platform which Plat4mation can provide.

Plat4mation is managed by six managing directors: Elmer de Valk, Roel Schoenmakers, Bas Tax, Gerry Appeltants, Raymond Beijerling and Thijs Daemon see figure 17.

1.2 ServiceNow

ServiceNow is a cloud computing company founded in 2004 under the name GlideSoft. It offers the ServiceNow platform which allows businesses to streamline their IT service management [4] , IT security, customer service and human resources see figure 16. The platform allows businesses to use and create custom applications that can access the data on the platform. ServiceNow describes it as the following: *"Leverage the value of all your enterprisewide systems and data by integrating process workflows between people-to-people, people-to-system, or system-to-system interactions"* (ServiceNow, 2017a).

1.3 Assignment

The goal of my internship assignment is to improve the development workflow. Increase the amount of time the developers can spend developing, and decrease the amount of overhead that is created by the platform. Rather than copy-pasting code across different development tools, it should be automated.

1.3.1 Scope

The development process that is currently followed will be explained. This main focus of the assignment is to look at integration of third party development tools into the ServiceNow platform.

[4] ServiceNow offers applications on the platform that follow ITIL practices.

1.4 Motivation

The development process at Plat4mation is still young. It is in the process of maturing. The development process currently contains manual steps that could be automated. Deployment to development environments is done manually copy pasting code into the platform editor. This task is both repetitive and error prone. When an application is ready for publication to the ServiceNow store. The javascript files exposed to the client will be minified, this is currently a partially automated task. The developer's IDE generate minified javascript and the developer copy pastes it into the platform.

When all the code is written locally and not on the ServiceNow instances. GIT becomes more accessible as version control over ServiceNow's own file history. Which can be hard to navigate at times, and does not contain easy ways of tagging versions.

Automatic synchronization will also promote separation of concerns. When developers have to put in extra effort to split code of into multiple files when developing. They might not create new files because of the overhead that gets created. Even if the files locally contain code about different subjects that should be separated.

These tasks take time from the developers, without adding any value so the time spent on them should be minimized.

1.5 Research Question

How can front-end build tools be leveraged in combination with the ServiceNow REST API to speed up the development process when developing service portal applications?

1.5.1 Sub-Questions

How does the current development process look and which points can be improved?
To be able improve the development process, the weak points have to be located.

Which front-end build tools can be used in our development process?
This question will answer which what kind of tools could be used, and will look at the pros and the cons of a few specific tools.

How can the ServiceNow API be leveraged to integrate third-party tools into the development environment?
This question will be answered in the form of research synchronization tools.

Which problems can be expected when integrating third party development tools into the platform?
What challenges need to be overcome when integrating tools.

How does the development workflow look like, after the introduction of frontend build- and synchronization tooling?
This question is reflective and will look at the previous workflow and how the new flow is improved with the use of synchronization and build tooling.

2 Development process

This chapter will explain the process that developers at Plat4mation follow when working locally.

2.1 Local development

Because the development tools on the platform are not as feature rich, as competitors from the web-development world. Most of the developers at Plat4mation use the jetbrains IDE Webstorm. Some developers use alternatives such as Sublime, Atom or Notepad++.

2.2 Integration into the platform

ServiceNow does not support integration with the previously mentioned tools. Developers that want to test their code, have to manually put it on the development instance [5] .

The developer has to navigate to the file on ServiceNow. Then copy paste the code from their local code editor into the file on ServiceNow. Followed by saving the record that has just been pasted into the platform. Finally the developer can open the page on the platform that hosts the application.

If a developer is working in multiple files at once. Their workspace can get very crowded quickly.

It can also be error prone, if a developer makes a mistake copy pasting a file. The application will not function correctly.

2.3 Source control

At Plat4mation the source control GIT is used to track changes to code. It is not possible to create a connection to a git repository. For source control purposes on the ServiceNow platform.

The platform does offer GIT integration, but that is for other purposes. Those will be explained further in 5.7 Deployment process.

[5] What an instance is, will be explained in chapter 5 ServiceNow

3 Deployment process

This chapter will focus on the deployment process.

In an ideal development process the product is deployed bug free so the customer can use the product without any errors. The developers should spend their times developing and not fixing issues with the deployments.

The following development process differ in their degrees in automation but they all have a similar approach.

3.1 DTAP

DTAP is an acronym stands for Development, Test, Acceptance, Production it is a phased approach to development, testing and deployment processes. It also contains the management of associated hosting landscapes.

Each phase in this process usually has an dedicated environment. The development environment is often unstable and actively being changed. Testing it here would be futile because the tester cannot be sure about the version of the product that has been tested. Unit tests can still be used to test individual modules. And the environment might not be in close alignment with production deployments.

After the developers have bundled all new features in a new release, it can be deployed on a testing environment. Testing is usually done on a testing environment that contains specific test data. The new release is deployed and the testers can verify all the features. If the testers are satisfied by the release it can be deployed on a acceptance environment.

The acceptance environment is where the customer will test whether the release meets their expectations. Using realistic data including integration to external systems.

If the release is accepted it will be deployed to a production environment where all customers will have access to it.

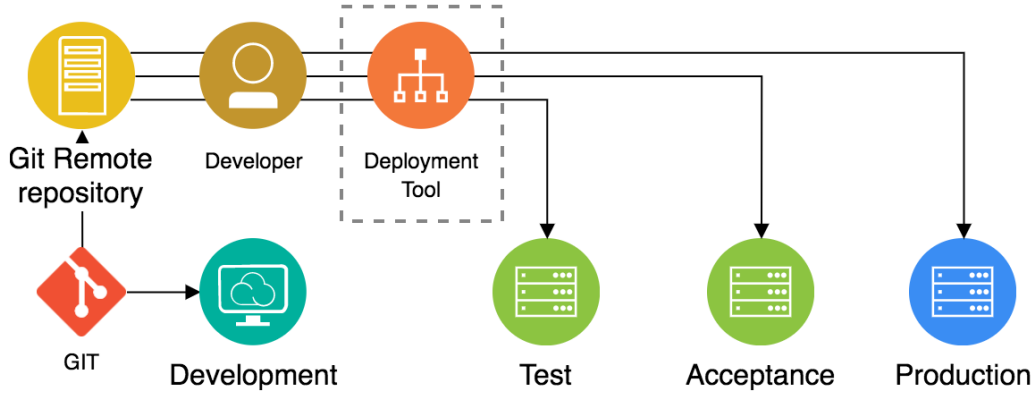

Figure 1: DTAP Cycle

3.2 Continuous integration

Continuous integration (CI) follows many of the same stages as the DTAP process. The primary goal of CI is reducing the time spent merging code and fixing merge related issues. This is achieved by merging smaller parts more often rather than, having to merge large commits less often. Once a merge request is created, a git hook starts the automated test process (Pittet, 2018). That's why unit testing and other types of automated testing are vital to continuous integration pipelines. If the merged build cannot be tested automatically, the process is still blocked by quality assurance. If all the tests pass, the build is merged into the main development branch. If a test fails the branch is rejected, and the developer that created the merge request is notified. This creates a fast feedback loop for catching bugs that have been created by new code.

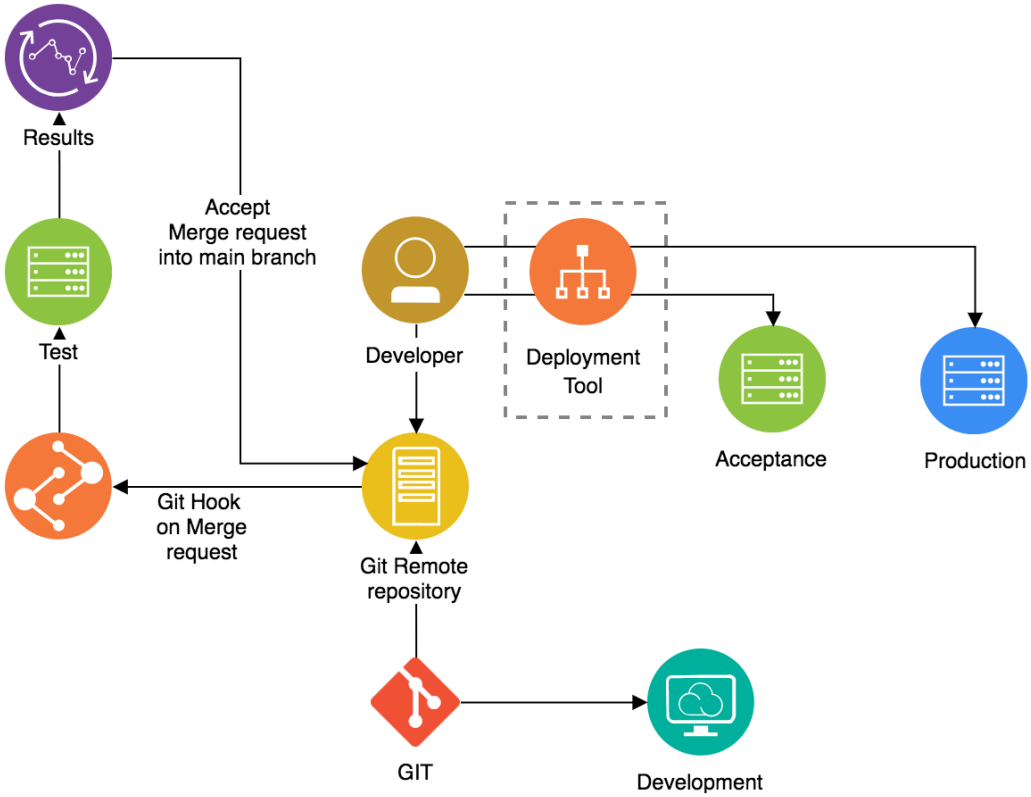

Figure 2: Continuous integration

3.3 Continuous delivery/deployment

Continuous delivery (CD) is the next step in automation, this automates the deployment process. The only action required to deploy a version of the software, is the press of a button. The CI process needs to be mature, to ensure that the deployment tool will only output trustworthy builds.

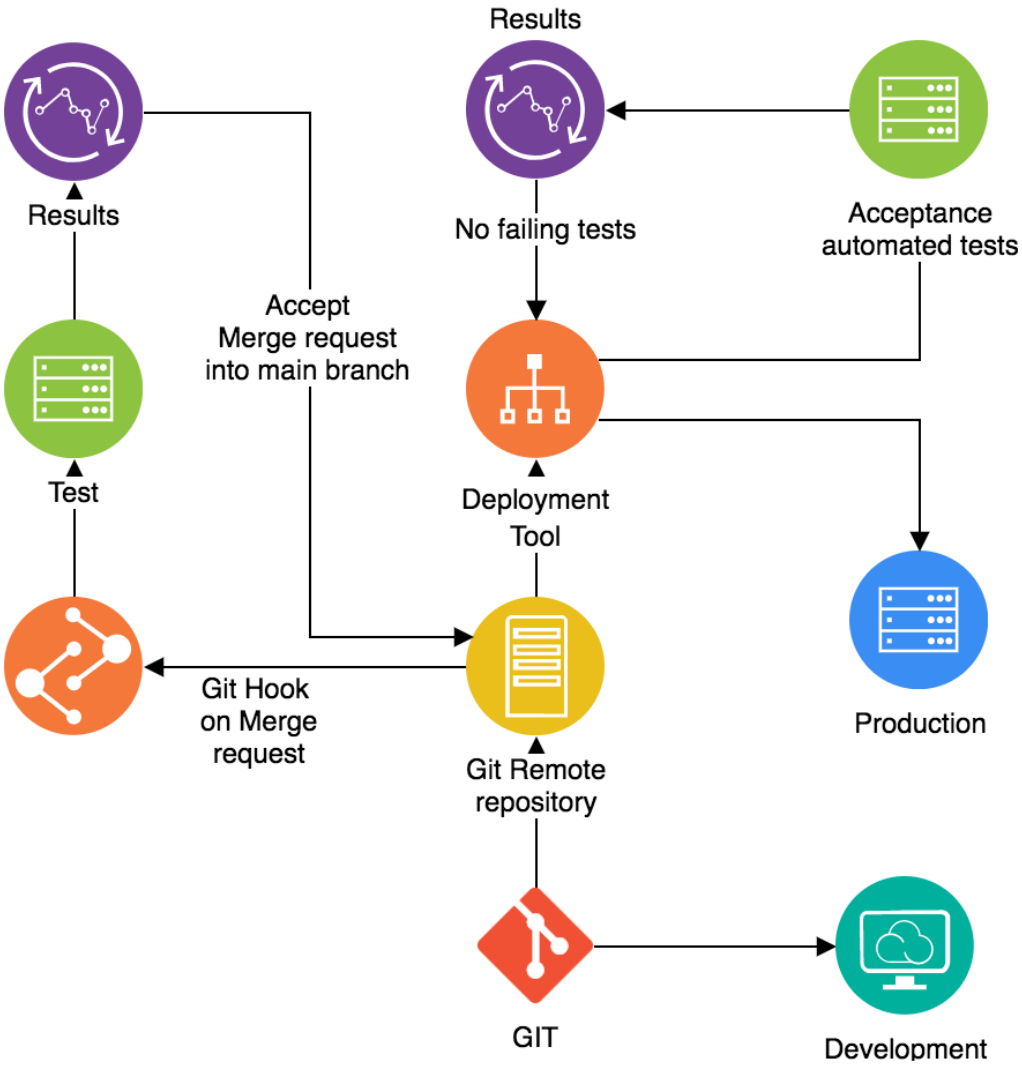

Figure 3: Continuous delivery

Continuous deployment is an even further step of automation. This automates the deployment step. Rather than a manual action of deploying with a tool. The system automatically deploys builds, if all the automated tests pass successfully.

3.4 Plat4mation's Process

The current process is similar to continues integration, but rather than GIT being a single source of truth. The ServiceNow development instance is the place where all code has to be integrated. From there the process is similar to continues delivery, since a developer or administrator can publish a new version at any moment. Production builds cannot be distributed as fast, because they have to go through the store validation process.

Plat4mation already uses GIT outside of ServiceNow, this is where integration between code from developers happens. The result that contains the merged code from all developers, has to be manually transferred to ServiceNow. The continues deployment/delivery process is handled by ServiceNow. Which is the platform that deploying and is the target of deployments.

The step that can be improved most is continues integration. After code is integrated, manually transferring code to the development instance should be automated. That would streamline the process further into less complicated steps, leading to fewer possibilities for error.

4 Cloud computing service models

Cloud computing services can be divided in three distinct categories

- Infrastructure as a service IaaS
- Platform as a service PaaS
- Software as a service SaaS

This list starts at the lower technical level where more knowledge is required but it gives the customer the largest amount of freedom. It ends at a service where the customer only has to use the product, but will be restricted by the limitations of that product.

4.1 Infrastructure as a service

IaaS is very similar to having traditional servers in a data center, you still have full access to the storage, operating system and the hardware (virtualised). The difference is the scalability, IaaS providers usually have a dashboard where more hardware can be provisioned on demand. In some cases it can be auto-scaled this keeps costs down when the demand on the infrastructure is low, but as soon as more capacity is needed it will scale up. This is a lot harder for IaaS than for the other models, because a virtual machine can't be magically scaled up. New virtual machines can be setup, but how are the configurations for these machines managed? Microsoft azure solved the problem by forcing customers to pre-create the virtual machines. The customer has to configure when more servers should turn on and how many and when it should scale back down again (Savill, 2014). The customer only has to pay for the active instances, so if the service on these servers has a varying demand auto-scale can help keeping the cost down. While still being able to serve the demand. The configuration of these machines can become very complicated for large systems, both platform as a Service and Software as a Service manage these configurations for the user. This results in less effort for the customer, but more restrictions in its capabilities.

4.2 Platform as a service

PaaS abstracts the hardware away, the customer doesn't have to maintain it's hardware virtual or otherwise. It can be used to develop and deploy complex

applications, the consumer doesn't have to handle the hardware and software infrastructure. The platform provides a hosting and development platform for customers. Different platforms support different technologies, so choosing a platform can limit the technology you can use. For example Heroku supports many languages some of these include: Java, PHP, Python, Go, Scala and Clojure. ServiceNow on the other side only supports development in Javascript.

Both platforms are very stable and the customers can depend on the up-time of these platforms. Developers might choose to use PaaS, when they only want to focus on their application. Rather than having to spend time on the infrastructure and configurations, of the servers that host their application.

4.3 Software as a service

When using SaaS the customer can request, procure, activate and use applications with lesser amounts of required configuration. SaaS in general uses monthly cost per user rather than a one time fee. A well known example of SaaS is Microsoft Office 365, it contains the well known suite of productivity apps: Word, Excel, PowerPoint and more. The old Office model was a license for each product with bulk licenses available for businesses. A consumer bought an office license it could be installed on a set amount of devices. These licenses would not have an expiration date. The office 365 license has to be payed on a regular basis there are monthly and yearly plans available. The license is for a user not for installs of the software. Office 365 works online in a web interface, it works on smart-phones, tables, laptops and desktops. The data created by users can be saved to a shared location on OneDrive, and can allow multiple people to work on the same documents at once from a device of their choosing. They don't have to host their own file server, Microsoft handles that and it is included in the service.

The downside of SaaS is being locked into that software. Let's take Word as an example: there are programs other than Word that can be used to open Word documents. For example open office, however they usually don't provide they same high quality service as the original product. A competitor for Office 365, G suite from Google contains the Docs application. It is possible to open Word documents but the styling of the documents will suffer. This makes it less likely for an existing customer of the Office 365 suite to switch to an alternative. Because the customer is locked into the office platform.

5 ServiceNow

This section will contain background information about ServiceNow to give some context into the platform Plat4mation does it's core business with.

5.1 Platform or Service

In cloud computing there are different models that are exploited, these are discussed in more detail in 4 Cloud computing service models. ServiceNow can both be used as platform or as a service this section will explain usages in both scenario's.

5.1.1 Usage as a service

ServiceNow can be configured as such that end users can use it as a service. Let look at incident management as an example. It includes the following features(ServiceNow, 2017c) [6].

- Log incidents in the instance or by sending email.
- Classify incidents by impact and urgency to prioritize work.
- Assign to appropriate groups for quick resolution.
- Escalate as necessary for further investigation.
- Resolve the incident and notify the user who logged it.
- Use reports to monitor, track, and analyze service levels and improvement.

To the end users it would just be an incident management application. A service that they use to manage incidents, they don't have to know anything about the platform.

5.1.2 Usage as a platform

Administrators on the platform can modify functionality, and even share those modifications through the store. The administrator can host other

[6] Incident management how ServiceNow supplies it with the Kingston release

software services alongside incident management, making it a platform that contains multiple services.

Applications can also be developed on the platform using ServiceNow front and back end scripting capabilities. Administrators of an ServiceNow instance can use these capabilities to modify existing plugins or applications.

5.2 ServiceNow architecture

A ServiceNow instance is a collections application nodes connected to the same database.

ServiceNow consists of multiple components, the application node layer consists of Java processes with embedded Tomcat. It contains it's own request processing threads and worker threads. These connect to the database(s) [7] of the instance, Mysql is the predominantly used database technology. However there are some ServiceNow instances where Oracle databases are used.

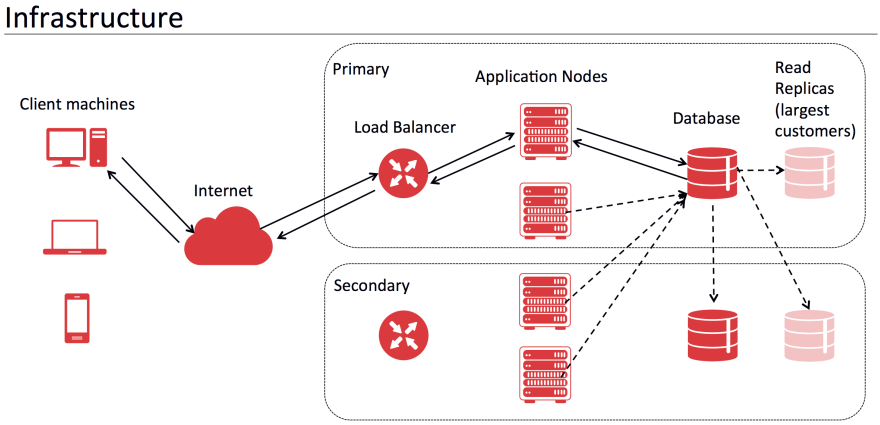

Figure 4: ServiceNow infrastructure

Figure 4 was taken from a presentation at ServiceNow's knowledge conference(Laethem & Bohy, 2016).

The diagram in figure4 shows a primary and secondary instance. The goal of these paired instances it to ensure high availability. In the event of a hardware or other problem that causes issues on the instance. The dns settings will be switched so users will be redirected to the secondary instance. The secondary

[7]Each instance only has a single database, but multiple duplicated db's can be used to enhance performance

instance is located in a paired data center. This way the customers data and uptime is secured. Even in large scale events such a fire in a data center. (Service-Now, 2017) The map in figure 5 shows the data center pairs.

Figure 5: ServiceNow Data centers

The resulting architecture, features high stability and can handle large volumes of users. Making it very suitable for enterprise environments.

5.3 External connections

System other than ServiceNow can connect to an instance through the REST[8] api. One of the API's that exists out of the box on an ServiceNow instance is the NOW table API . The Table API allows for CRUD access to the tables in ServiceNow.

[8]REST stands for Representational state transfer (ServiceNow, 2017a)

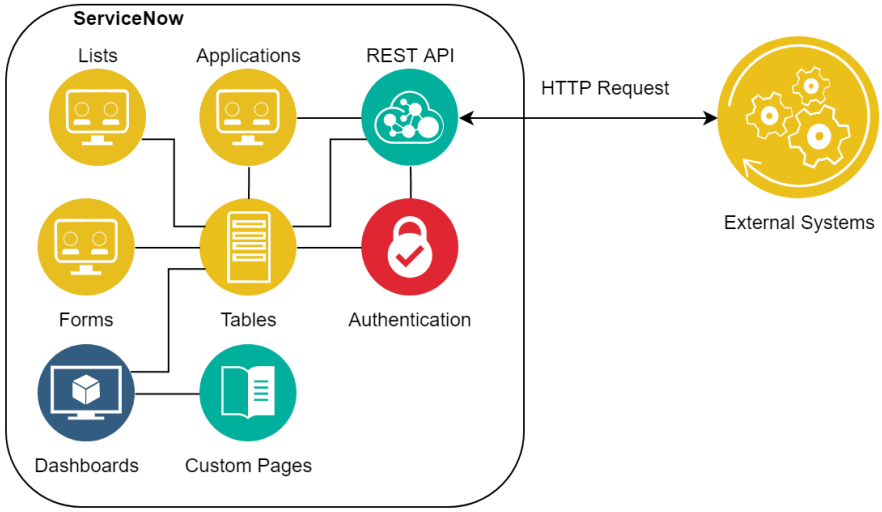

Figure 6: ServiceNow External Connections

Custom API endpoint can also be created, their behavior is defined by a javascript file.

However all the code written by the development team at Plat4mation. Is stored in tables it can be easily modified using the out of the box table api.

5.3.1 Development Tool integration

You can only install ServiceNow specific applications on the platform. Existing third party tools such as static code analyzers cannot be installed on an instance. All records containing code or markup files, can be retrieved and updated using the table API . So integration with third party tools is possible, through the use of the API .

This will be researched further in chapter 7 Third party tool integration solution

5.4 Service portal app architecture

This section is going to explain the architecture of the Service Portal, to give information about the components the developers use. The architecture of

the service portal framework is ingrained with AngularJS. For a detailed explanation about the Service portal components and how they interact with AngularJS see section Service portal in Appendix A about ServiceNow components. All code that can be executed has to be Javascript.

5.4.1 General

The data model for the application is defined by it's table structure. When a table is created a default form and list view are also created. These can be used to Create Read Update and Delete records. This functionality exists outside of the service portal framework.

Service portal is meant as a entry point to all the services on the Platform. A ServiceNow instance can contain multiple portals. Within each portal the administrators have control over styling of the portal's theme. A portal can contain multiple pages for example: a knowledge base, a 404 error page, a home page. The homepage usually servers as a dashboard for the end-users.

Most of the applications made at Plat4mation are full page applications. To achieve this we create a custom page portal with homepage for the application. That page contains a widget that contains the application.

The application can communicate with the platform in multiple ways. The server script of a widget, it has access to the server side libraries ServiceNow provides. It can be used to access data from the tables. Script includes are a layer of abstraction it are records filled with Javascript code that can be executed server side. How these components are connected to each other is shown in figure 6.

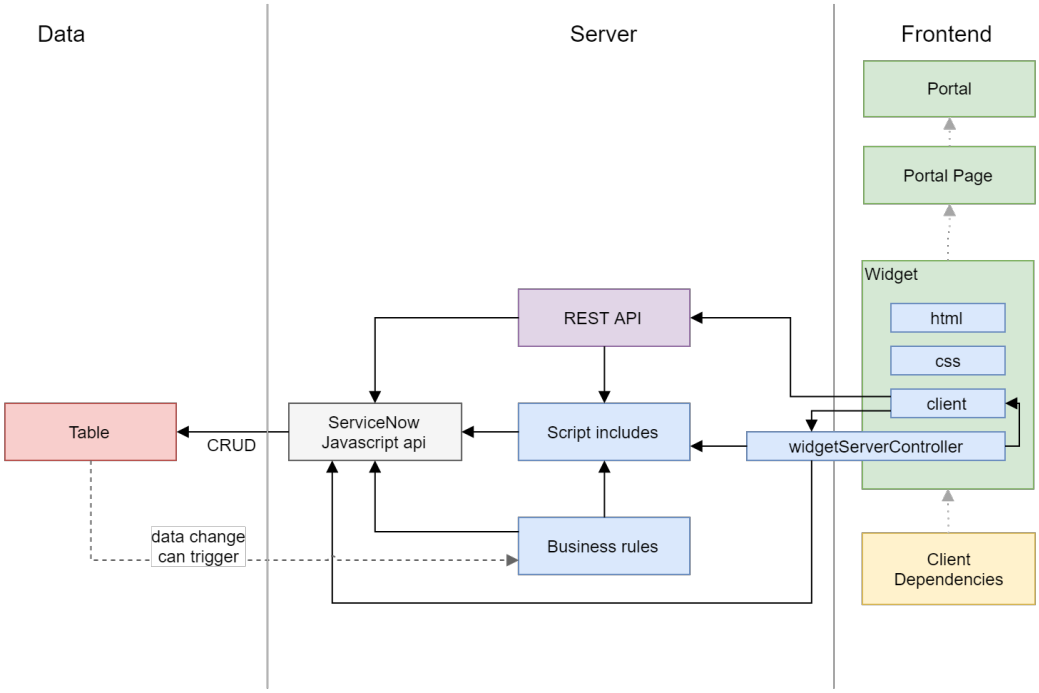

Figure 7: Service Portal app architecture

5.4.2 Widget server controller

In ServiceNow the widgetServerController is called a widget server script. The server script can set some initial data in the client script when the widget is loaded. Once the application is running the client can do data requests to the server script directly. In such cases the server script can a javascript object back to the client script. The client script can do http requests to the REST API's provided by the platform. ServiceNow provides some API's out of the box, such as a table API. It can be used to do CRUD operations on tables. Custom API's can also be created, they have access to the previously mentioned Javascript libraries. The API's can also access script includes to reduce code duplication.

These characteristics make the server script similar to a controller in a normal web application. The main difference being that, many widgets can be included on the same page. They can update and request data from their server scripts individually.

5.4.3 The client

is an AngularJS controller that handles the interactivity of the page. It is possible to use external Javascript libraries. Front-end javascript libraries such as momentJs, these are saved as external scripts and have to be linked to the widget as a dependency.

5.4.4 Business rules

are pieces of Javascript that are triggered by changes in the database. They are always connected to a specific table. The criteria for it's trigger can be simple rules. Examples are: on insert, on update, on delete, on query. Out of those the records that it applies to can be filtered. A filter could contain field: state is new. They can either execute simple or complex actions. Simple actions are configured with a condition builder. Where a simple change in data is specified such as: state is work in progress. It is also possible to use an advanced change, which can be scripted in JavaScript.

5.5 Development process

This section will give an overview on the technical side of the development process. During development a programmer starts writing code based on a user story. The code is written in a local IDE, most developers use Webstorm. They will then copy paste their code into the editor ServiceNow supplies to save their changes.

5.5.1 Local development

The developers use third party tools not supplied by the platform. When developing locally developers have a few advantages. They can files faster than on an online platform. Tools can be used to manipulate the files that are not available on the platform. Third party tools and their uses are explained further in 6 Development Tooling.

5.5.2 Version control on the platform

File version control is an important topic for developers. It is used to retrace in which version, a bug or feature has been introduced. And which developer

added that code.

The term file is a bit misleading since ServiceNow does not work in terms of files. Everything is saved as a record in the database. Scripts in custom applications are also saved as records. Script type records from multiple tables are also tracked in the sys_update_version table. This allows developers to return scripts to previous states.

However every time a record is saved it counts as a version. The application code cannot be run locally but only on the ServiceNow instance. Because of this a script might get saved many times during development.

This results in a version control history that is hard to navigate.

5.5.3 GIT version control

The development team at Plat4mation uses GIT to overcome the problems with ServiceNow's versioning. A folder structure that reflects the application structure on ServiceNow is used to store scripts locally. The developers usually use local code editors see section 5.3 Local development. Their local version of the code doesn't get automatically updated, when other developers work on the same code. GIT is used by the developers to ensure that developers can easily find the latest version of a file. It is also used to share code between developers working on the same project.

5.6 Development tools

This part of the chapter will discuss the development tools that ServiceNow offers.

5.6.1 Studio

ServiceNow offers the "studio" application that can be used by developers to create ServiceNow applications. It contains quick access to all the ServiceNow components that can be used when creating an application. The left side of the studio contains the "application explorer" where all the components that are part of the application can be found. The right side contains a tabbed overview of all opened components see figure 8.

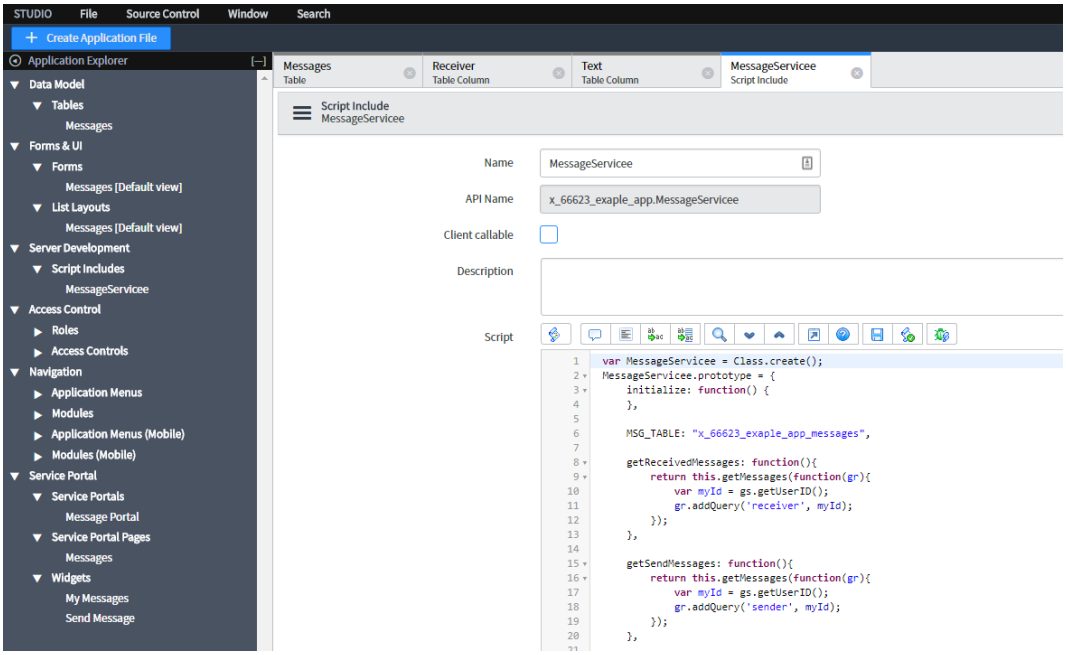

Figure 8: Studio

5.6.2 Portal Editor

The portal editor allows developers or admins to easily create portals for end users. It is a drag and drop system where containers can be put on the page and then filled with widgets. A portal could contain a container which holds a two column layout. Both of those columns can contain their own widgets see figure 18

5.6.3 Widget Editor

The widget editor can be used to edit the: html, client script, (s)css, server script, and angular providers; angular templates see figure 20. It comes with the disadvantage of saving all of them every time. So if someone updates an angular provider, and another developer saves the widget that they had already opened that widget in the widget editor. It will overwrite the angular provider if it is part of that widget.

5.6.4 Syntax highlighting

Syntax highlight is done using javascript, putting each "word" in it's own html element and coloring those elements. For larger script files this creates a lot of DOM elements [9] which hinders browser performance.

[9]DOM stands for Domain object model and refers to elements in a web page.

5.7 Deployment process

There are multiple ways of deploying applications on Service Now instances. This section gives information about integration with GIT, the internal application repository, the app-store and update-sets.

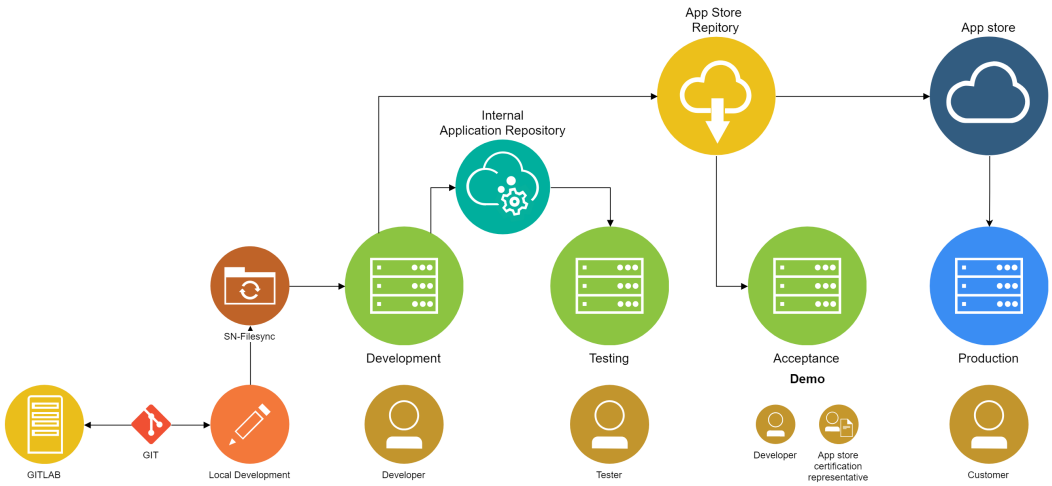

Figure 9: Development Lifecycle

5.7.1 Update-sets

An Service Now application can be exported to XML. Such an export contains all the records and meta data that are part of the app. These contain: script includes, widgets, table definitions and all other records the app needs to function.

Update sets can also be used to overwrite installed application files. This allows for hot-fixes to be installed without the time it takes to submit through the store.

5.7.2 Git integration

Service now offers integration with git to do version control. "*The source control integration allows application developers to integrate with a GIT source control repository to save and manage multiple versions of an application from a sub-production instance.*" (ServiceNow, 2017b)

This is source control between Service Now instances, and cannot be modified locally. The repository contains a set of xml files describing the records for an application.

This is not a very mature feature yet. There have been multiple occurrences at Plat4mation where this technology has failed. To resolve it we had to contact Service Now. They had to modify files that are not avaible even to administrators of an instance.

5.7.3 Internal Application repository

The platform offers a way of deploying scoped applications to instances using the application repository. This is limited to instances within the same company. Plat4mation uses this technique to deploy applications to testing instances.

5.7.4 Service Now store

Service Now offers an app-store that can be used to find and download applications. Plat4mation uses this method primarily for distributing applications to customers. Applications that are available on the store are verified by a technical team from Service Now.

6 Development Tooling

This Chapter will look into front end build tooling so an informed decision can be made about what tools would fit into our development workflow.

6.1 Webstorm

This is the main IDE used by developers at Plat4mation.

It contains integration with GIT . Webstorm is very extensible, the plugin repitory contains 1016 plugins at the time of writing(Jetbrains, 2018).

Smart code highlighting and warnings when code could be simplified. Overview of all project files, in a sidebar. Customizable shortcuts for actions such as: duplicate current line, move line, extend cursor to multiple lines.

6.2 Package managers

A package manager is a software tool or a combination of multiple software tools, that automates parts of the installation, configuration, upgrade and removal processes of software. At Plat4mation the NPM [10] package manager is used to download development tools. These tools include: task runners, Javascript transpilers and minification-software.

6.3 UglifyJS

UglifyJS is a tool that minifies javascript files. The minification process strips out whitespace, optimizes code to be as short as possible to make the resulting file as small as possible. The javascript code is often "mangled" as well to make the resulting code harder to read. Mangling often further reduces the file size, since variables are often renamed to single characters. The actual purpose of mangling is to protect the intellectual property of the code so it cannot be read and reused, without having access to the source.

[10] "npm is the package manager for JavaScript and the worlds largest software registry. Discover packages of reusable code and assemble them in powerful new ways."(npmjs.com, n.d.). Npm is a very popular package manager.

The following block of code shows the minified ouput for example code: see listing 13.

```
1 angular.module("app").directive("helloWorld",["$scope",
  ↪ "$http",function(t,e){return{restrict:"E",template:
  ↪ "<div><h1>{{init}}</h1></div>",controller:
  ↪ [function(t){t.init="Hello World"}]}}]);
```

Listing 1: Angular directive minified

The size reduction is dependent on multiple factors, strings don't get minified so string heavy code will be compressed less. In a more real world example a minified the code of three widgets that are part of the Agile4U suite.

Name	**.js**	**.min.js**	**Compression Ratio**
portfolioWall.client	43kb	19kb	56%
releaseView.client	43kb	15kb	66%
storyMap.client	51kb	22kb	57%

Table 1: Example file size changes

The files initially don't start off huge, but compression can still improve the load times for the end-user especially if they are on a slow internet connection.

6.4 BabelJS

BabelJS transpiles newer standards of Javascript to Javascript standards compatbile with browser you target. It allows developers to write in newer standers of Javascript such as ECMAScript 2015 and beyond, without breaking browser compatibility.

To demonstrate the usage of babel, arrow functions will be used as an example of a newer ECMAScript feature that is not supported in all browsers. Arrow functions are supported in: Chrome, FireFox, Edge and Safari; internet explorer does not support arrow functions (SphinxKnight et al., 2017). So using them will cause the code to fail on internet explorer. The developers can choose to not not support internet explorer or to change the code.

Alternatively BabelJS can be used to transpile the ES 2015 valid code to an output that is supported by all the browser the developers are trying to target.

The following snippet in listing 2 shows ES2015 valid code.

```
1 let list = [1,2,3].map(n => n + 1);
```

Listing 2: ES 2015 arrow function

Running it through babelJS would output the code in listing 3, it would execute correctly on internet explorer and other browsers.

```
1 var list = [1,2,3].map(function(n){
2         return n + 1;
3 });
```

Listing 3: Browser compatible version of the previous code listing

6.5 Task Runners

Task runners are used to minify and bundle Javascript files, style-sheets and other assets they compile assets that need it and copy the bundled ready to use files to a output directory. They can also be used to automatically run linters [11] and testing tools.

6.5.1 Gulp

Gulp is a javascript based task runner, where the "gulpfile" that holds the tasks is written in Javascript. Gulp itself has got a minimalistic API containing only 4 functions that can be called.

Table 2: Gulp API

Function	**Description**
gulp.task	Can be used to create gulp tasks, is passed a task name and a function it executes.
gulp.src	Creates a file(s) stream that can be piped to plugins or custom written functions.
gulp.dest	Writes the file stream that is passed to it to disk.
gulp.watch	Watches files for changes and activates on file change.

[11] Linters are tools that flag suspicious code and can automatically check whether the code follows a given code-style guide

Example usage

```
1  const gulp = require('gulp');
2  const uglify = require('gulp-uglify');
3  const babel = require('gulp-babel');
4  const rename = require('gulp-rename');
5
6  gulp.task('Minify js production', () => {
7
8      gulp.src(['src/script.js']).pipe(babel({
9          presets: ['env']
10     }))
11     .pipe(rename({ suffix: '.min' }))
12     .pipe(uglify())
13     .pipe(gulp.dest('./out/'))
14
15 });
```

This very basic gulpfile allows the developer to run the file through babel [12] then get minified and renamed and saved to an output directory.

The advantage of gulp lies in the fact that it's configuration is written as JavaScript, since all the developers already have to work with JavaScript daily it will be a low barrier to entry.

Once a gulp file is setup for a project all developers can use it. It could contain a task to minify all Javascript files. When a production ready version of a script, has to be created, the developer can minify all code with one task. Then the code in the output folder can be transferred to ServiceNow.

6.5.2 Grunt

Grunt is task runner with a large ecosystem of plugins. It allows developers to automate many tasks with small amounts of effort, thanks to the existing plugins. Configuring gulp is not like gulp, where tasks are written as code. They are configured through code, setting Javascript objects with the required settings.

[12] BabelJS is a transpiler for JavaScript it turns es6 into JavaScript that is compatible with current web browsers.

Here is an example of an simple grunt file to uglify an Javascript file.

```
module.exports = function(grunt) {
grunt.initConfig({
    babel: {
        dist: {
            files: [{
                "expand": true,
                "cwd": "src/js",
                "src": ["**/*.js"],
                "dest": "src/js-compiled/",
                "ext": "-compiled.js"
            }]
        }
    },
    uglify: {
        build: {
            src : 'src/js-compiled/**/*-compiled.js',
            dest : 'src/build/all.min.js'
        }
    }
});
    grunt.loadNpmTasks('grunt-babel');
    grunt.loadNpmTasks('grunt-contrib-uglify');
    grunt.registerTask("default", ["babel", "uglify"]);
};
```

Listing 4: Grunt config example

The registered default task contains babel and uglify, so when the grunt command is executed; the default task will run and will babel first and then uglify. It will run the Javascript files through babel and save the results in a js-compiled folder. Then the uglify task will run and minify the compiled Javascript into the build output folder. Grunt task can be executed from the command line by entering: “grunt [task]” or simply “grunt” for the default task.

Once files have been modified using grunt tasks, a release ready version has been created in an output folder. This can then be transferred to ServiceNow, so the next steps in the testing and acceptance process can be followed.

6.6 Combined

When the previously mentioned tools are used in combination with each other. It becomes easy to create a production build of the code. The developer can call a task on the task runner to:

First transpile code to a browser compatible version using babel.

Followed by a minification process obfuscating the code and creating better load times.

The developer can copy paste the onto the platform. Verify that the build has been properly “installed” on the development instance. It can now be deployed to a testing or acceptance instance to validate the build.

This allows the developers to write code in newer standards of Javascript. It protects the source code by obfuscating the code, and increases loading speed for the end user thanks to the minifcation process. Achieved by issuing a single command to a task runner.

7 Third party tool integration solution

The current environment of development tooling ServiceNow provides is lacking compared to other domains of web development. If the developers want to use existing tooling to help with their work, the code that has been created has to be manually synced to the server. Which is an error prone and repetitive task that is not good use of developer time.

7.1 Synchronization requirements

The synchronization should apply to the following types of files:

- HTML markup
- css/scss stylesheets
- javascript code

These are the types of files that have to be changed often by developers. These file types make up the core of our Service Portal applications. With these files synchronized the developers will be able to do all code writing locally. This would allow developers to keep using WebStorm [13] without having to copy-paste their code into ServiceNow. This requires the developer to open multiple script files, or to look up to correct file on ServiceNow every time.

7.2 Custom Tool

The purpose of the custom tool, is to upload locally developed scripts. To reduce the need of manually transferring code to a ServiceNow instance. It should start as a command line utility where the user calls the program, and specifies a file that needs to be uploaded.

It will do this by connecting with the ServiceNow REST API, it will use the HTTP PUT method. [14] It will use the table and id of the ServiceNow record that contains the code. To create an url query for the ServiceNow API . The

[13] The current most used idea for developers at Plat4mation

[14] According to RFC-2616 a PUT will create a new entity or update if an entity already exists (?, ?). ServiceNow's table API follows the standards for a RESTful API, and will adhere to this standard.

body of the PUT request will contain a JSON object. Where the keys will be the fields on the records that will be changed, and the content will be the new code.

To achieve this we locally need: a name to identify it with, the new code, the table in which it should be stored and the id of the record where it should be stored.

7.3 Architecture

The tool is written in Java because Webstorm plugins are written in Java. That would allow an integration with Webstorm, while using a large part of the same source code.

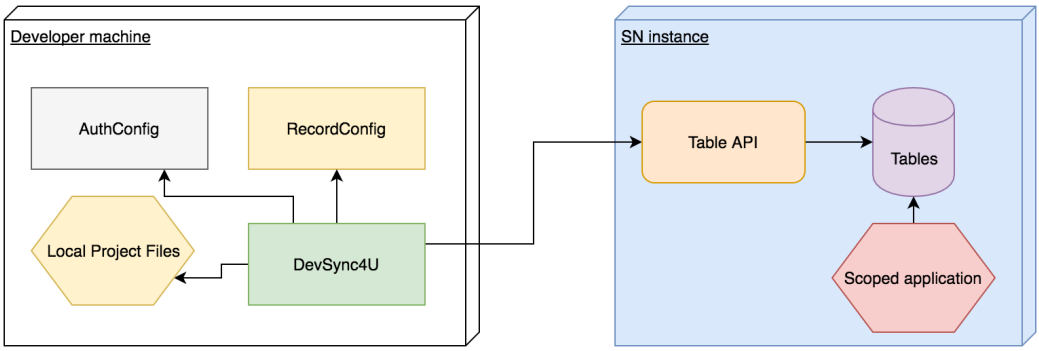

Figure 10: DevSync4U overview

The tool transfers local project files [15] to ServiceNow through the table API . To do this it needs Authentication information and information about the records and associated local files.

The application needs models for authentication configuration, record configuration, overall configuration combining the previous ones and for records. See figure 21 in the Appendix for the exact specification of the models.

A record in this application contains all the information needed for a request to the table API . The id of a record, the table it is located in and the data it is going to insert into ServiceNow.

[15] ServiceNow application project files.

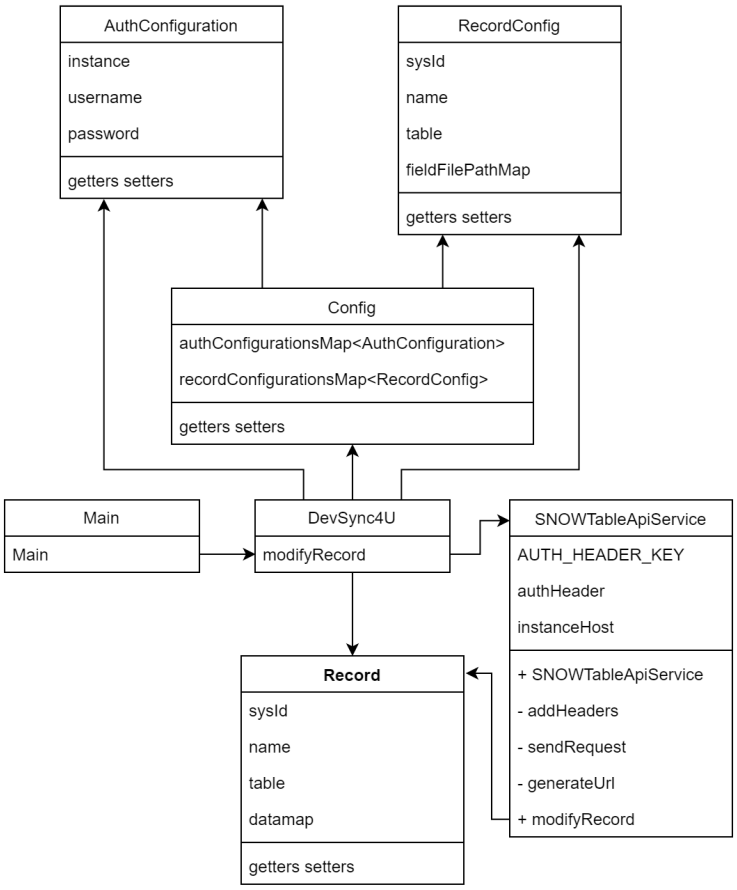

Figure 11: DevSync4U diagram

7.3.1 Applicationflow

The main function of the application, parses the command line arguments and loads the configuration files. For this example the arguments “devsync4u -u somerecord” where used. No custom configuration path was specified, so it wil look for configuration files in the current folder. The “-u” flag specifies that a record should be uploaded, somerecord is the name of the record that should be uploaded. The application will then look through its record configuration for an item with name “somerecord”. That contains fields and file paths, the tool will generate key value pairs based on the contents of the files[16]. The keys will be specified in the configuration file and refer to ServiceNow record fields.

[16]At those paths

The application now has all its needed configuration and created Record objects. It will create a SnowTableApiService object using the AuthConfiguration. The modifyRecord function can now be called with a Record. That function will create a web request, with authentication headers based on the authConfiguration object. The content of the request will be that of the generated key value pairs. This will be sent to the ServiceNow API, the API will update a specific record with the content of the request.

7.3.2 Configuration structure

The advantage of creating the tool is full control in it's data structure. The chosen structure empathizes configuration over code, to allow for flexibility with changes that might come to the platform. The configuration contains two components, the project specific config file that can be tracked in git. And the user configuration that contains their credentials to connect to the ServiceNow instance. This was split up so the developers only have to maintain a single configuration file. If multiple developers work on a the same application they get the updated configuration file when the pull the changes to the application from git.

Other tools used a single configuration file that also contained authentication information. In that situation the configuration file cannot be shared and has to be maintained in multiple places. One with authentication that all developers have access to and the file that is actually used by the tool.

The other part of the configuration contains records that can be uploaded see listing 15 for an example config file. The JSON[17] contains an object filled with objects with names for the tool as key and record configuration objects as values. The name is used by the Devsync4U tool to specify which record should be uploaded. The object itself contains a string's for the instance, sys_id and table. It also contains a key value pair tot specifies to which files which fields on the ServiceNow record are mapped to which local files.

This gives the developer full control to map any given record on ServiceNow to a field on a file, but that has the disadvantage that it can become a large amount of needed configuration for larger applications. The file however can

[17]JSON can contain key value pairs containing: nested objects, arrays, strings, numbers or booleans.

be tracked in git so it would only have to happen once for every record that needs to be pushed to ServiceNow.

7.3.3 Strengths

This configuration setup would allow any type of ServiceNow record to be mapped to a file. So it already supports new scripting types that are not yet available on ServiceNow. It splits up record and authentication configuration allowing record configuration to be shared in version control.

7.3.4 Weaknesses

Since this is a tool developed at Plat4mation it would have to be maintained by ourselves. If the ServiceNow API it uses changes, it would cost developer time to update it before it can be used again. The configuration is very extensive and has to be done for each file.

7.4 Atom servicenow-sync

Atom is a text editor created by Github that supports many programming languages. It also supports plugins in Javascript, which has created a large ecosystem of plugins for the tool. The Atom[18] plugin servicenow-sync syncs files to records on ServiceNow, however this tool can only be used in Atom.

It comes with a graphical user interface to create links between local files and ServiceNow records. The user can open the configure file UI by using a shortcut, or by using action search. Once it is open, the user can copy a ServiceNow record's url and enter it in the UI . This will detect the ServiceNow instance host and record sys_id and fill it in. The user than has to add a username and password, if they save this information. The record will be downloaded to the local machine. The file can be changed and when the upload command, is called it will upload the record to ServiceNow.

[18]Atom is an open-source code editor that can use Javascript based plugins.

7.4.1 Configuration

The configuration of this tool is based on a per file basis, so each file that would be synced to a field on a record would create a config file. This would double the amount of files we have in a project, and the configuration has to be redone by each developer since these configuration files also contain authentication configuration.

This tool created more work in configuration and did not fit well in our current development flow. That's To conclude This tool forces an new code editor and creates a lot of configuration files, that means it does not fit into our development flow nicely.

7.4.2 File structure

The tool lets the user decide its own file structure. It does require that configuration files are in the same folders, as their associated files. But it does not adhere to any stricter rules, so Plat4mation's existing project structure can be used.

7.5 sn-filesync

Sn-filesync is an open-source nodeJS based synchronization tool for synchronizing local files to ServiceNow.

In October of 2013 Fruition-partners created an synchronization tool between local files and the ServiceNow platform. This repository has not been maintained but has been forked by dynamicdan, the fork has gained more popularity because it is a more up to date version with more features. [19] The original fruition-partners filesync contains nine commits over the course of two days in 2013 (johncaruso, 2013). Dynamicdan's sn-filesync contains 187 commits over the course of 4 years (dynamicdan, 2017). An explanation on how to use this tool can be found in Appendix H Sn-filesync usage guide

[19]The original filesync tool can be found here `https://github.com/fruition-partners/filesync`.

7.5.1 Features

The tool can download files from the ServiceNow instance based on the configuration file. It can watch for changes in the files downloaded by the tool, so the synchronization process becomes as simple as saving the file after the developer has changed it. It is available on both Windows and MacOS however the developer does not use windows and does not actively test new features on that OS.

The tool saves file hashes so it can detect collisions and does not overwrite the server, it gives a warning. The developer can still choose to overwrite the record on the instance but will be informed that it will overwrite changes that where not created by the tool.

7.5.2 Configuration structure

This tool contains an extensive configuration file that allows for fitting the tool to Plat4mation's needs. The file structure that gets created by the tool, when downloading files of an instance is dedicated by configuration. An example of a complete configuration file for this tool can be found in Appendix I Example config file sn-filesync.

```
1  "search": {
2        "app": {
3             "query": "sys_scope=[APPLICATION SCOPE
              ↪  SYS\_ID]",
4             "records_per_search": 1000
5         }
6    }
```

Listing 5: Config search example.

The tool allows for custom queries so the developer can specify which files from ServiceNow should be downloaded. This allows the developer to download all the files in an application scope easily see listing 8.

For the files to be saved they need to be specified in the folder "folders" object in the configuration file see listing 9.

```
"folders": {
      "widgets": {
            "table": "sp_widget",
            "key": "name",
            "fields": {
                "client.js": "client_script",
                "demo.js": "demo_data",
                "server.js": "script",
                "html": "template",
                "style.scss": "css"
            },
            "subDirPattern": "name",
            "_custom": true
                }
        }
```

Listing 6: Widget file structure defined in the config file.

In the folders object in the config file the widgets object is declared, that means the tool will create a folder with the name "widgets". The table specified is "sp_widget" so it will only look at records in that table. It will then follow the sub directory pattern to create a folder for each record based on the record's name. Because that is the key that has been chosen, the fields object specifies which field will correspond with what kind of file extension. The widgets "client_scripts" will be saved as [widgetname].client.js

7.5.3 File structure

```
/Users/Dries/Projects/synchronizedProject/widgets/
|-- Scrumboard4U
|   |-- Scrumboard4U.client.js
|   |-- Scrumboard4U.html
|   |-- Scrumboard4U.server.js
|   `-- Scrumboard4U.style.scss
`-- Scrumboard4U\ release\ view
|   |-- Scrumboard4U\ release\ view.client.js
|   |-- Scrumboard4U\ release\ view.html
|   |-- Scrumboard4U\ release\ view.server.js
|   `-- Scrumboard4U\ release\ view.style.scss
```

Listing 7: Synchronized widget folder structure

7.5.4 Configuration sharing

The configuration that the file sync tool expects contains user credentials to authenticate to the Servcie Now REST API. That makes it unsuitable to track in git, which is something that could be very valuable since the configuration wouldn't have to be maintained multiple times.

That is why i created a script to solve that issue. The script uses an authentication configuration file, that can store the credentials for users on multiple ServiceNow instances. It uses an template configuration file, a configuration file that specifies the relative directory and a users private authentication file.

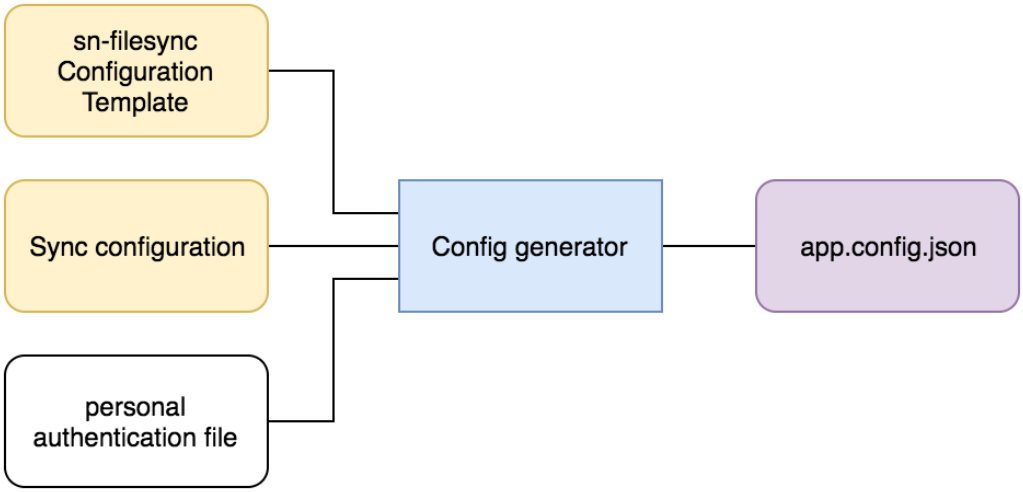

Figure 12: Configuration Generation

The yellow blocks in figure 10 can be tracked in git for each project. The sn-filesync configuration template contains all the data that will be synced from ServiceNow to the local environment. It will be the same for most projects with some, exceptions that can be easily managed since the file is tracked in git. The sync configuration file contains a relative path to the folder that will contain the synchronized files, and the host that it should sync to. The personal authentication file is not tracked in the project but can be saved as a hidden file in the user's home directory.

The filesync tool expects a "roots" object in the config file.

```
1  "roots": {
2   "/Users/Dries/Projects/synchronizedProject": {
3       "host": "dev41418.service-now.com",
4       "user": "dries.meerman@plat4mation.com",
5       "pass": "[REDACTED]",
6        "auth":  "ZHJpZXMubWVlcm1hbk.. =="
7   }
8  }
```

Listing 8: Roots Object in config file

The roots object contains a path or paths to folders, and those contain the

information that the tool uses to connect to the instance. A real config file wouldn't contain both "auth" and "user", "pass" since the tool replaces user and pass with an auth string every time it's used. the auth string contains a Base64 representation of the a string containing "user:pass".

When the config generator script is used, it uses the directory where it is executed as the parent for the relative path specified in the sync configuration. It uses the information in the personal authentication file and combines that with

7.6 Conclusion

The sn-filesync tool is the most feature complete tool that still offers a wide range of flexibility in its use. It is an open-source project so if anything can't be done with the tool we could fork the project, and add the functionality while still getting updates. The other tools are more verbose in their configuration which would create a higher amount of complexity. It's configuration is extensive without being complicated, making it easy to generate. That in combination with a script created to generate it's configuration leads to an ideal situation where the developer can easily use the synchronization tool, while being very powerful.

8 Evaluation and Comparison

8.1 Development Tooling

The available development tools will be compared in this section. The tools that will be compared are the ones most often encountered by developers. The script editor that is used script and markup records outside of the Service portal. The portal editor where widget related files are edited.

8.1.1 Matrix

The tools will be evaluated on the following metrics: hotkey support, code completion, code referencing, syntax validation and speed.

Table 3: Development tooling

	Hotkeys	Completion	Referencing	Validation	Speed	Total
Portal editor	1	1	0	1	0	3
Script editor	0	1	0	1	0	2
Webstorm	2	2	1	2	1	8

What is not shown in this comparison is deployment in ServiceNow. Saving a file in the portal or script editor deploys that code on that instance. Doing it in Webstorm on it's own doesn't do anything.

Looking at the totals Webstorm comes out as the decisive winner. The main reason is the web basis for the portal and script editor. Because they are web based some functionality is harder to achieve.

Looking at hotkeys webstorm and allows for remapping of hotkeys. The script editor doesn't have any hotkeys. Other than the browser supplied undo (ctrl + z). The portal editor overrides the browsers default behavior of (ctrl + s). Allowing developers to save the widget, instead of being prompted to save the web page.

In both the portal and script editors, there is a minimal implementation of code completion. The ServiceNow Javascript libraries functions are completed. But only the function names, there is no parameter hinting. The portal editor also offers completion of functions written in the same file.

Webstorm gets two points, because it's completion also shows parameters. It allows developers to add custom libraries for completion.

Both the portal and script editor don't have code referencing. While Webstorm allows control clicking functions that are being used. To open that function's definition, even if the definition is declared in a different file.

The editors offered by ServiceNow contain syntax validation. However it is not configurable, and limited to five colors. White background, green comments, blue variables and numbers. Javascript keywords are purple, strings are near black purple. Any other text is black.

Webstorm offers an extensive customizable colorscheme. Multiple themes come with the editor, and many others are freely available online. Webstorms syntax validation also offers code simplification.

As far as speed goes Webstorm has the advantage of being local. It does not have the delay of loading the web pages containing the editors. When editing larger files the portal and script editor become less responsive. The reason is because of the DOM manipulation that happens to create the syntax highlighting. Webstorm does not slow down even when editing larger files.

8.1.2 Managing the disadvantages

Since the portal and script editor are both maintained by ServiceNow. It's disadvantages cannot be changed. The only way to be unaffected by those disadvantages. Is by choosing not to use the tools.

The main disadvantage of Webstorm is no direct connection to ServiceNow. This means that code written in Webstorm still has to pass through the other editors to be saved. The way to manage this is by using a synchronization tool.

8.2 Task runners

Grunt and Gulp are two wel known task runners. Gulp and grunt are very similar, the main difference is the configuration ideology.

8.2.1 Matrix

The two tools will be compared on three points: speed, plugins and configurability.

Table 4: Task runners compared

	Speed	Plugins	Configuration	total
Grunt	0	1	0	1
Gulp	1	0	1	2

The statistics on speed and plugins where found in this article (Arsenault, 2017).

Looking at speed Gulp wins over grunt. However looking at plugins, Grunt has an advantage.

The configuration is harder because it is more ideological than objectively better. However gulp has an edge because it's configuration is in Javascript code rather than JSON . Since the code on the ServiceNow platform is Javascript, it is a small leap to use Javascript to configure the task runner.

8.2.2 Managing the disadvantages

Grunt is slower but it is not glacially slow. Either tool is still fast when completing it's tasks.

Looking at the types of plugins Plat4mation needs, Uglification, Babel [20] and copying files. Both tools have plugins to achieve these tasks.

For configuration using JSON rather than Javascript, there is a learning curve there. To overcome this it takes a time investment to learn it.

8.3 Third party integration

Multiple tools have been researched that can function as a bridge between local development and the development instance. This section will compare a custom created tool, a plugin for the atom code editor and sn-filesync.

[20] To transpile code to older web browsers.

8.3.1 Matrix

The synchronization tools will be compared on the following criteria. Configuration ability, how much effort is needed by developers to setup the tool for use. Environment restriction, are there requirements to use the tool. File watching, is it possible for the tool to detect the saving of files. Conflict resolution, does the tool notify the user when there are version mismatches. Can the tool download files from the ServiceNow instance.

Table 5: Synchronization

	Config	Environment	file watching	conflict	download	Total
Custom Tool	1	1	0	0	0	2
Atom plugin	0	0	1	1	1	3
Sn-filesync	2	1	1	1	1	6

The custom tool uses two configuration files. The first file contains authentication information. The second file specifies the links between local files and ServiceNow records. The developers still have to specify the sys_id and the relative local path for each file. Because this file doesn't contain authentication it can saved in the GIT repository.

Looking at configurability, the Atom plugin requires a configuration file for each that will be synchronized. Each of these configuration files can be created from a prompt within Atom. They contain the user's authentication credentials. For larger application this will lead to a very complex configuration.

The configuration of Sn-filesync is based on record type and folder structure. Rather than specifying which record is synchronized to which file. The structure contains all script includes in folder scriptincludes/[scriptincludename].js . This allows developers to use the same configuration file for multiple projects. As long as the project structure is the same.

8.3.2 Managing the disadvantages

The custom tool is created by developers at Plat4mation, so it can be expanded to contain the currently missing features.

The Atom plugin's configuration is very similar for all records. A custom script could be created to generate all the sync files. Based on a custom

configuration file that contains all record to file information. The tool is also open-source so it could be modified to work in our workflow.

8.4 New development flow

This section will explain the new development flow with inclusion of all the new tools.

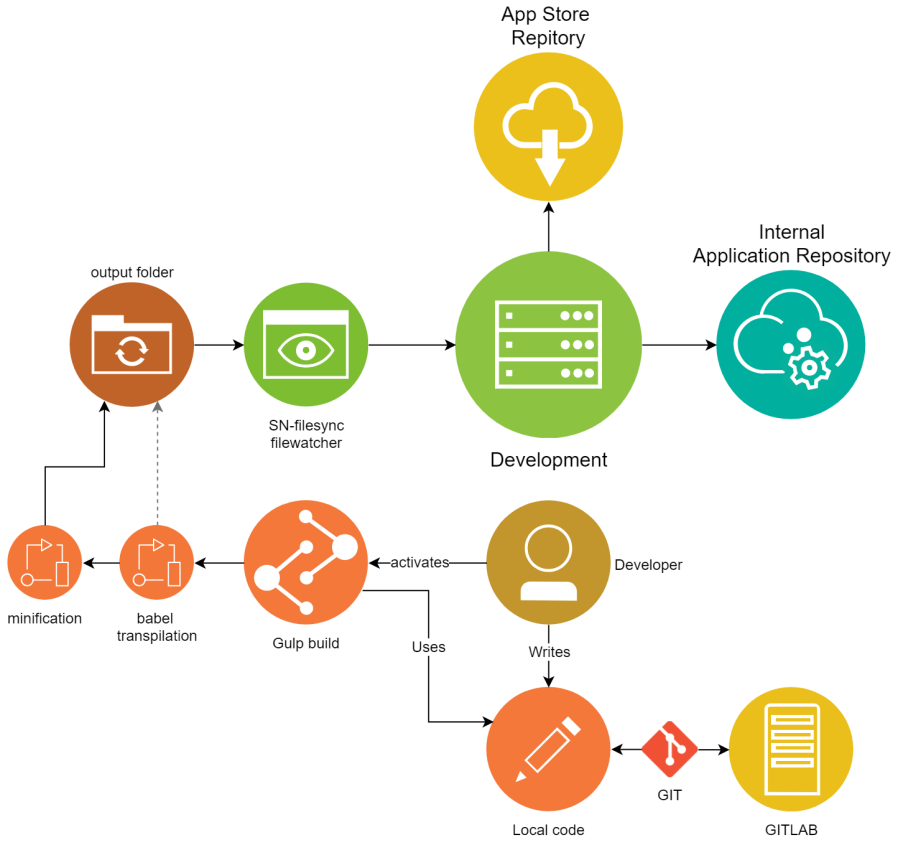

Figure 13: Development Lifecycle

In the new development flow the developer writes code locally and commits it to GIT. Other developers will do the same and they can merge code using GIT. Once the local code is ready to try, a partial gulp build can be started. It will transpile the code if needed to a browser compatible version of Javascript. The resulting code will be saved in an output folder which will be watched by the sn-filesync tool. If the tool detects changes in files it will upload them to

ServiceNow. If a file was unchanged it will not be uploaded. And if any files are in conflict because someone else updated them on ServiceNow, a warning will be given. If a complete build is done, the Javascript files will also be minified so the intelectual property will be protected.

Once the complete build is done, the developer can publish the version on the application repository through the development instance.

Conclusions

The goal of this report was to answer the following question: *How can front-end build tools be leveraged in combination with the Service Now REST API to speed up the development process when developing service portal applications?*
Task runners can be used to combine the results of multiple build tools. These results can be automatically transferred to ServiceNow, using a tool that connects to the REST API. This relieves developers of having to lookup files and copy paste contents. Removing a manual step, making developers more productive in the process.

Using third party development tools offer many advantages. The development tools offered by ServiceNow aren't as feature rich, as counter parts from the web development world. They can be faster and are more flexible, allowing the developer to change settings. Settings such as color scheming, shortcut configuration and plugins to integrate with other tools. Are available on third party tools, but not on the platform.

The platform has the advantage of instant integration. Once the code is created and saved in a editor on the platform, it is available on that environment. To bridge the gap between local editors, and the platform editors. A layer of synchronization had to be added.

Developing a proof of concept tool to upload files to the platform, gave a lot of insight. Into the requirements that the development team has for a synchronization tool. Easy configuration, that preferably can be shared between developers.

The tool that was chosen "sn-filesync" had some disadvantages. Configuration files could not be tracked in GIT, because it contained authentication data. Using knowledge from the POC that was created. I was able to create a configuration generator. That split up the configuration into two parts. Authentication configuration and project configuration. This allowed authentication to be tracked in GIT, without causing any security issues.

The sync tool solution gives the developers the best of both worlds. Free choice of tooling, without having to manually put code on the development environment.

Recommendations

The development team at Plat4mation is recommended to use a synchronization solution. Because it automates the tedious task of manually transferring code to the ServiceNow platform. Which is both error prone, and less productive leaving the developers less happy. Because can use tools that make them more productive, without having to manually copy paste any code to the platform.

The tool to do the synchronization should be a combination of sn-filesync and the custom config generator script. This leaves the developers with an up to date feature rich tool. Where it's weaknesses have been managed using config generator. Allowing the developers to share configuration, without exposing authentication data.

In combination with a task runner that starts other processes. The process of putting code on the platform becomes more structured. Code can be automatically minified before it arrives on the platform. Making it easier for Plat4mation to protect it's intellectual property. It makes more integration with GIT more accessible since there are fewer steps to take.

Overall, this gives more assurance about the integrity of the code that has been deployed to the platform. While taking up less effort in a shorter time span.

References

Arsenault, C. (2017, March). *Gulp vs grunt comparing both automation tools.* Retrieved 2018-03-05, from https://www.keycdn.com/blog/gulp-vs-grunt/

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). *Pattern-oriented software architecture, volume 1.* Wiley.

dynamicdan. (2017, May). *Dynamicdan / sn-filesync commit history.* Retrieved 2017-12-07, from https://github.com/dynamicdan/sn-filesync/commits/experimental

Jetbrains. (2018, March). *Jetbrains plugin repository.* Retrieved 2018-03-01, from https://plugins.jetbrains.com/webstorm

johncaruso. (2013, October). *Fruition partners / filesync commit history.* Retrieved 2017-12-07, from https://github.com/fruition-partners/filesync/commits/master

Krause, R. (2017, Februari). *Meet the 'fastest-growing' $1 billion enterprise software firm.* Retrieved 2017-12-02, from https://www.investors.com/research/the-new-america/servicenow-salesforce-com-push-saas-frontier-as-rivalry-grows/

Laethem, J., & Bohy, M. (2016, May). *Servicenow platform architecture.* Retrieved from https://community.servicenow.com/community?id=community_article&sys_id=ee7ca2e1dbd0dbc01dcaf3231f9619c7 (Knowledge 16 presentation)

npmjs.com. (n.d.). *npmjs.* Retrieved 2017-11-21, from http://npmjs.com

Pittet, S. (2018). *Continuous integration vs. continuous delivery vs. continuous deployment.* Retrieved 2018-01-26, from https://www.atlassian.com/continuous-delivery/ci-vs-ci-vs-cd

restfulapi.net. (2018). *Rest put vs post.* Retrieved 2018-03-22, from https://restfulapi.net/rest-put-vs-post/

Savill, J. (2014, Aug). *Azure auto-scale for iaas.* Retrieved from http://windowsitpro.com/azure/azure-auto-scale-iaasl

ServiceNow. (2017a). *Automate processes, reuse components and integrations, and delegate development.* Retrieved 2017-12-04, from https://www.servicenow.com/solutions/develop-new-cloud-business-apps.html

ServiceNow. (2017b, June). *Business rules.* Retrieved 2017-12-08, from https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/script/business-rules/concept/c_BusinessRules.html

ServiceNow. (2017c, November). *Business rules.* Retrieved 2018-02-18, from https://docs.servicenow.com/bundle/kingston-it-service-management/page/product/incident-management/concept/c_IncidentManagement.html

Service-Now. (2017, Oktober). *Delivering performance, scalability, and availability on the servicenow nonstop cloud.* Retrieved 2018-03-01, from https://www.servicenow.com/content/dam/servicenow/documents/whitepapers/wp-sn-advanced-high-availability-architecture.pdf (whitepaper: SN-EB-AHA-101717)

ServiceNow. (2017a). *Rest api reference.* Retrieved 2017-12-06, from https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions#Browser_compatibility

ServiceNow. (2017b, November). *Source control integration.* Retrieved 2018-02-20, from https://docs.servicenow.com/bundle/kingston-application-development/page/build/applications/concept/c_SourceControlIntegration.html

SphinxKnight, vidyasagarg1222, mariusschulz, CodeFry, all user, hkdnet, ... Jesse (2017). *Automate processes, reuse components and integrations, and delegate development.* Retrieved 2017-12-06, from https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions#Browser_compatibility

van Oijen, R. (2017, November). *Plat4mation groeit uit mechels coworkingkantoor.* Retrieved 2017-11-27, from https://www.linkedin.com/pulse/plat4mation-groeit-uit-mechels-coworkingkantoor-rik-van-oijen/

Glossary

Angular A frontend javascript framework that extends the capabilities of html. 58, 59

Apache A foundation that maintains and creates many opens source projects. https://www.apache.org/. 25

Base64 is a encoding scheme for binary data into 64 possible characters usually the lowercase and uppercase alphabet and numbers and a few more characters. 44

CD stands for continuous deployment. 10

CI stands for continuous integration. 9

CMS stands for content management system. A CMS supports the creation and modification of digital content.. 58

CRUD stands for Create Read Update Delete.. 20, 35, 57

Debian A distribution of Linux. 25, 28

dns stands domain name system. It is the system that translate domain names such as google.com into IP addresses.. 17

DOM stands document object model, it is the tree of all the elements on a webpage.. 24

ECMAScript ECMAScript (or ES) is a trademarked scripting-language specification standardized by Ecma International in ECMA-262 and ISO/IEC 16262. It was created to standardize JavaScript.. 33

ES Shorthand for ECMAScript.. 33

es6 Es6 stands for ecmascript 6 and is a standard of javascript not fully supported in most web browsers.. 30

git Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people. 27, 36, 42, 43

IDE stands for integrated development environment. Examples are: IntelliJ, Visual Studio. 4

ITIL formally an acronym for Information Technology Infrastructure Library, is a set of detailed practices for IT service management (ITSM) that focuses on aligning IT services with the needs of business.. 3

jelly is a tool for turnging XML into executable code.. 58

JSON stands for JavaScript Object Notation is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate.. 37

Linux is a generic term referring to the family of Unix-like computer operating systems that use the Linux kernel. 25, 28

minify Minification is the act of removing whitespace and doing other modifications to a file so that its functioanlity stays the same but the filesize becomse smaller.. 29, 59

nodeJS Javascript running from command line or as tools . 26, 27, 38

OS stands for Operating System.. 38

REST stands for Representational state transfer and defines a standard of predefined stateless operations.. 58

subversion subversion is a version control system for tracking changes in computer files and coordinating work on those files among multiple people. 27

9 Appendix

A: ServiceNow Components

This subsection in the appendix contains information about ServiceNow components that are and what they do in the platform.

Tables

Tables store data that the ServiceNow platform uses, existing tables can easily be modified or new ones can be created. The table configuration's are stored in the "sys_db_object" and "sys_dictionary" tables. The db_object contains the actual table, and the dictionary contains records for the table fields. To create a new table an administrator or developer only needs to create a record in the db_object table.

This creates a new table with the following default fields:

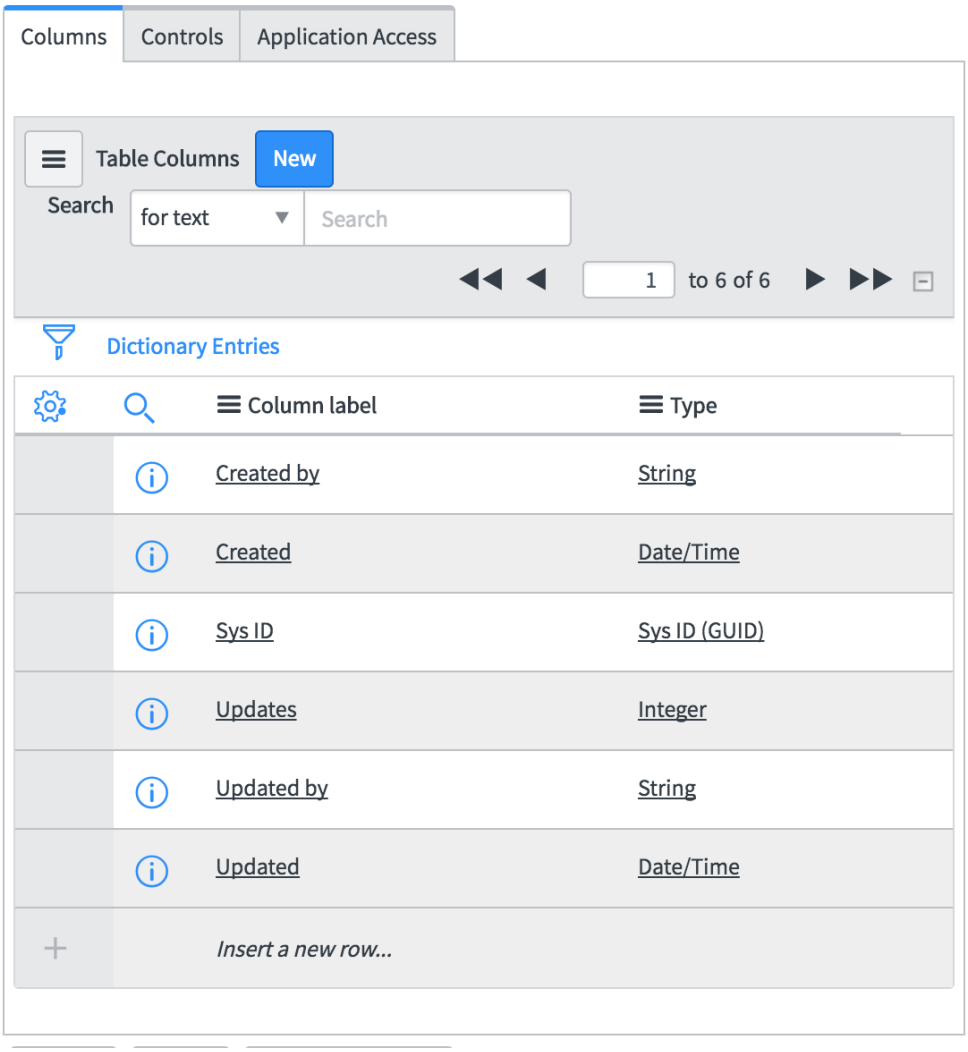

Figure 14: Table Fields

Querying

When looking at the list view of a table, the user can open the query builder to specify conditions the records must match to be shown. The query builder can access the fields of the current table, and can dotwalk [21]

[21] Dotwalking in ServiceNow refers to the act of referencing fields on other tables. A record could have a reference to a user. To get the user's email address dotwalking [22] can be applied "record.user.email' will contain the email address.'

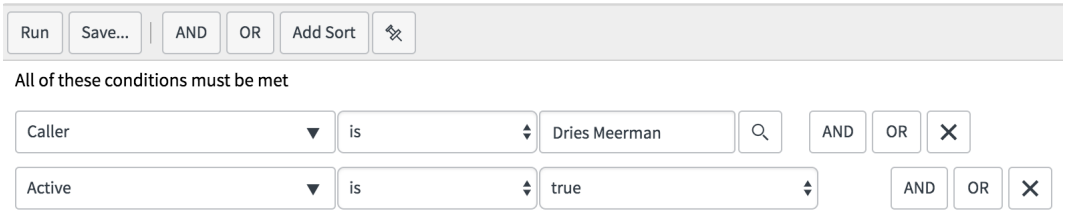

Figure 15: Query Builder

This query builder represents the following encoded query: “caller_id=javascript:gs.getUserID()^active=true”.

The encoded query can be copied from the query builder for reuse.

The server side GlideRecord API can be used in scripts to do CRUD operations on tables.

```
1   var gr = new GlideRecord('incident');
2   gr.addQuery('active', 'true');
3   //gr.addEncodedQuery('active=true'); encoded
    ↪  queries can also be used
4   gr.query();
5
6   var result = [];
7   while (gr.next()){
8   result.push(gr.getValue('number'));
9   };
10  return result;
```

Listing 9: Query example

General

Script Includes Script includes contain server side logic and can be called by other server side scripts and sometimes client side scripts. A script include record contains a flag that specifies whether it is client callable or not. At Plat4mation the development team uses script includes for server side utilities that are reused throughout our applications. Our applications often use Script includes to hold “data services” that contain the logic that interacts

with the database. Our applications follow the N-tier architecture also called multi-layer architecture [23], separating business logic from the application layer. Script includes often contain the business logic.

Scripted Rest API
ServiceNow allows developers to easily create new REST API endpoints to access server side data. The rest API's can process HTTP requests in the forms of: GET, PUT, POST, PATCH and DELETE requests. The endpoints contain server side scripts that the developer can use to send data back, or execute server side logic.

Business Rules
"A business rule is a server-side script that runs when a record is displayed, inserted, updated, or deleted, or when a table is queried." (ServiceNow, 2017b). Business rules can run before or after the action they are active on. Additional conditions can be added that specify when the business rule should run.

They can be used for simple operations such as setting indexes on records on creation. But since Business rules run server side they have got access to most of the ServiceNow API's and Script includes that exist on that instance. That means more complex operations such as calculations on data sets can be done, and those can be added to the record by the business rule.

UI Scripts
UI Scripts are records that are used to save reusable Javascript.

Style Sheets
Similar to UI Scripts style sheets store reusable css files.

Service Portal

Service Portal is a framework that allows ServiceNow administrators to create portals for end users. It is an alternative for the older CMS (Content management System) and it can interact with parts of the ServiceNow platform. The CMS offers more freedom of choice when choosing front end libraries to work with. In comparison to the Service Portal that is ingrained with angularJS. Service portal however adds new convenient ways to communicate data between the server and client. Where the CMS uses jelly and JSON API's to communicate data from the server to the client. Service Portal is

[23] Mutilayer architecture is described in the following book (Buschmann, Meunier, Rohnert, Sommerlad, & Stal, 1996)

highly ingrained with angular: each “portal widget” is essentially an Angular directive.

Service Portal is currently the new platform that many customers already use as the front-end for their end-users, that is why Plat4mation uses it as the primary technology for our applications.

The consists of multiple layers: the portal itself, portal pages and widgets.

Widgets

A widget consists of the following components: a html template, css, server script and a client script and a link object. In ServiceNow all widgets are saved in the “sp_widget” table. A widget can have dependencies, angular providers and angular ng-template which are saved in a many to many tables. These will be explained further later in this chapter.

A angular directive is normally defined in a javascript file and is attached to an Angular module. The first argument specifies its name, followed by an array or function. The Array would contain Strings of dependency names, followed by the function. If the array notation is not used but a function is used as argument, the variable names are used by angular dependency injection. When the code is minified and the parameters “$scope” and “$http” are minfied to letters such as “a” and “b”, angular wil try to inject the dependencies “a” and “b”. This will throw angular injection error, however when using the array notation the strings before the function are used instead solving this issue.

```
1  angular.module('app').directive('helloWorld', ['$scope',
   ↪  '$http', function($scope, $http){
2  return {
3  restrict: 'E',
4  template: '<div><h1>{{init}}</h1></div>',
5  controller: [function($scope){
6  $scope.init = "Hello World";
7  }],
8  }
9  }])
```

Listing 10: Angular directive Example

In the html angular does not follow pascal case but in the javascript it does. So the helloWorld directive becomes “hello-world”.

```
1  <!DOCTYPE html>
2  <html>
3  <body ng-app="app" ng-controller='controller'>
4  <hello-world></hello-world>
5  </body>
6  </html>
```

Listing 11: Angular directive Usage Example

The resulting output will show Hello World to the user.

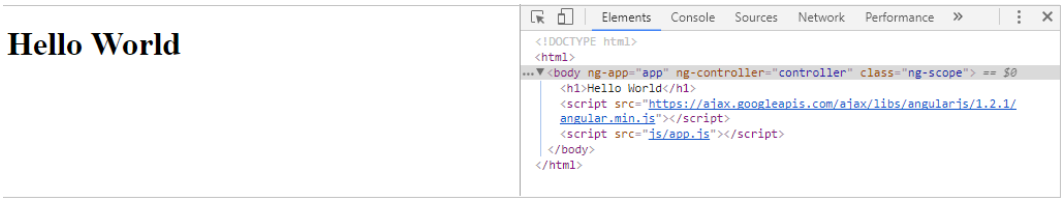

Figure 16: Portal Widget

A Service Portal widget is essentially an angular directive with a few twists.

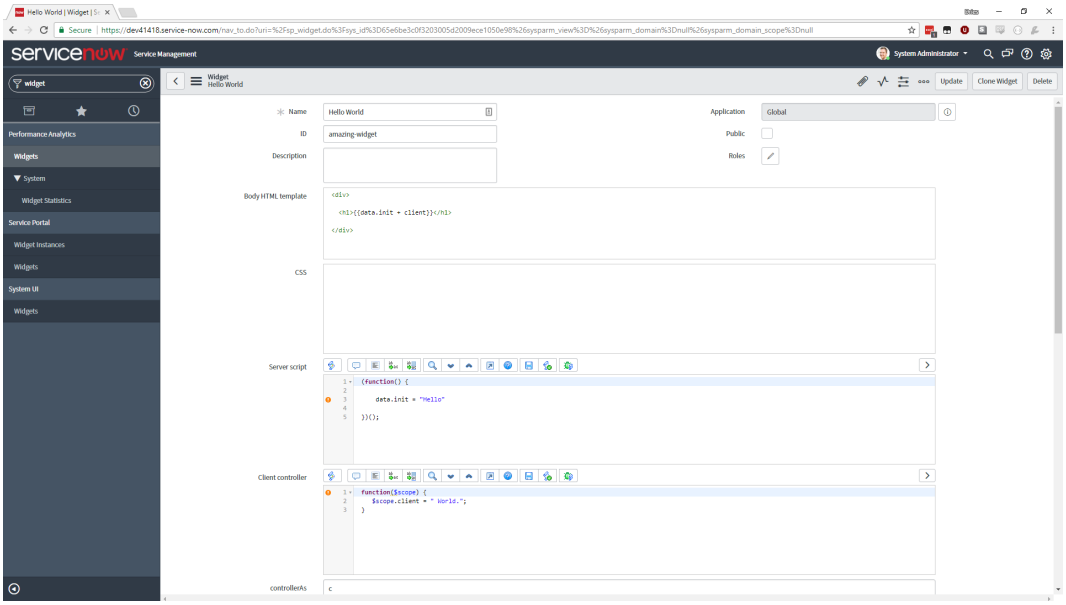

Figure 17: Portal Widget

The Widget record contains html, (s)css, link, client script and a server script. A widget cannot be called as a directive how a developer normally would in angular. A widget can be put into a "portal" or nested inside another widget.

An angular directive also does not contain a server side component or css. When a widget is loaded in Service Portal compiles the scss to css and loads it into the page. The server script is executed and can possibly set values in the controller's $scope. This can be accessed from the "data" field that is added to the $scope object.

The server script has got access to the ServiceNow server side API's and the "GlideSPScriptable" (used as $sp in the scripts). That gives the server script access to full CRUD access using the "GlideRecord" API. This makes it easier for developers to make the widget interactive using server side data.

Angular providers Angular providers contain angular directives, factories or services. ServiceNow adds all of a widgets providers to the same angular module, and injects them into the widget.

They are linked to widgets in the "m2m_sp_ng_pro_sp_widget" table.

Angular templates Angular can make use of templates, in the case of larger directives it is nice to be able to split it up into a javascript file and a html

template. Rather than creating a huge string of inline html that is used by a directive.

Dependencies Widgets in ServicePortal can define dependencies that need to be loaded for the widget to work. An example could be momentJS [24] The dependency itself is a many two many record, that references a widget dependency. The same dependency record can be used by multiple widgets. A widget dependency can contain css includes and UI scripts. If a widget is loaded the scripts and style sheets defined in its dependencies will be loaded into the page.

[24]Momentjs is a library that makes interacting with date and time objects easier.

B: Store apps

Figure 15 shows a screen shot of the applications by Plat4mation that are on the ServiceNow Store.

Retrieved on 08-03-2018.

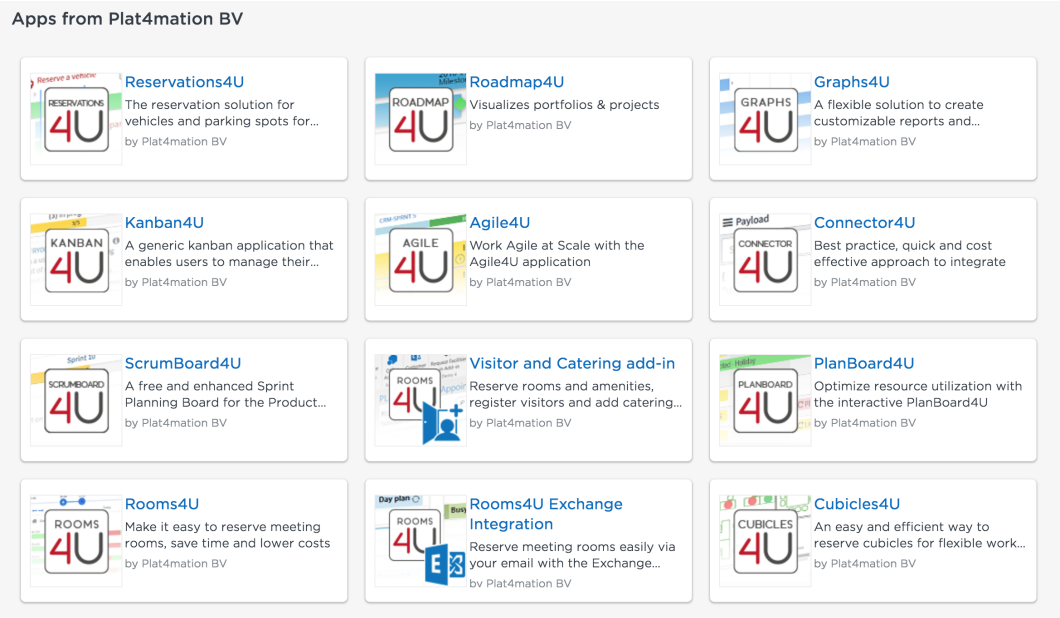

Figure 18: Store Apps

C: ServiceNow Platform

Figure 19: ServiceNow Platform

Retrieved from `https://www.servicenow.com/why-servicenow.html` on 02-12-2017.

D: Organogram

Figure 20: Organogram

E: Service portal development

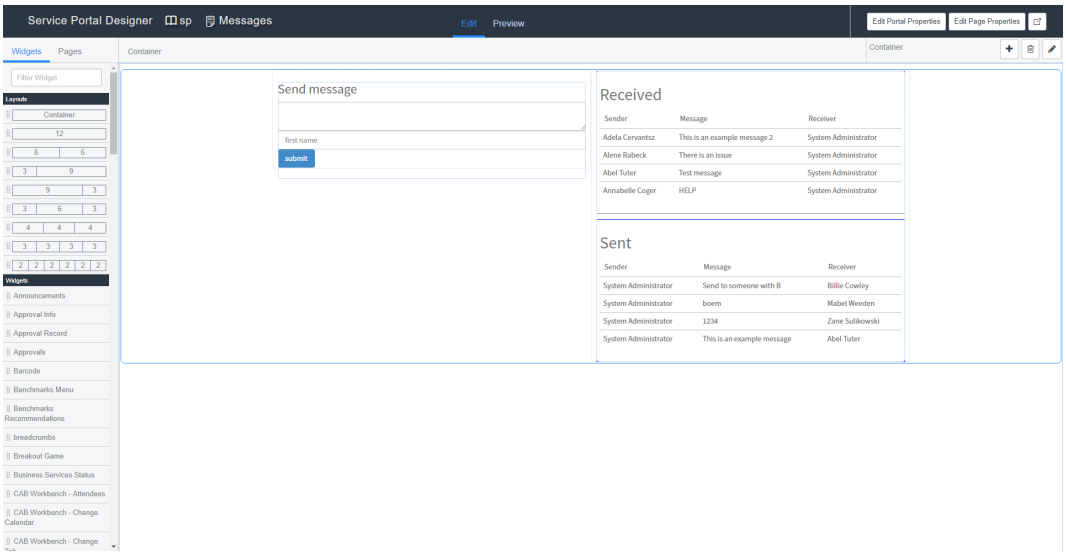

Figure 21: Portal Page Designer

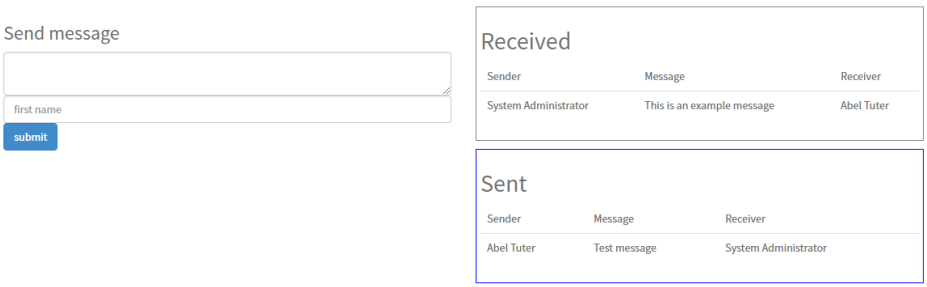

Figure 22: Message Portal

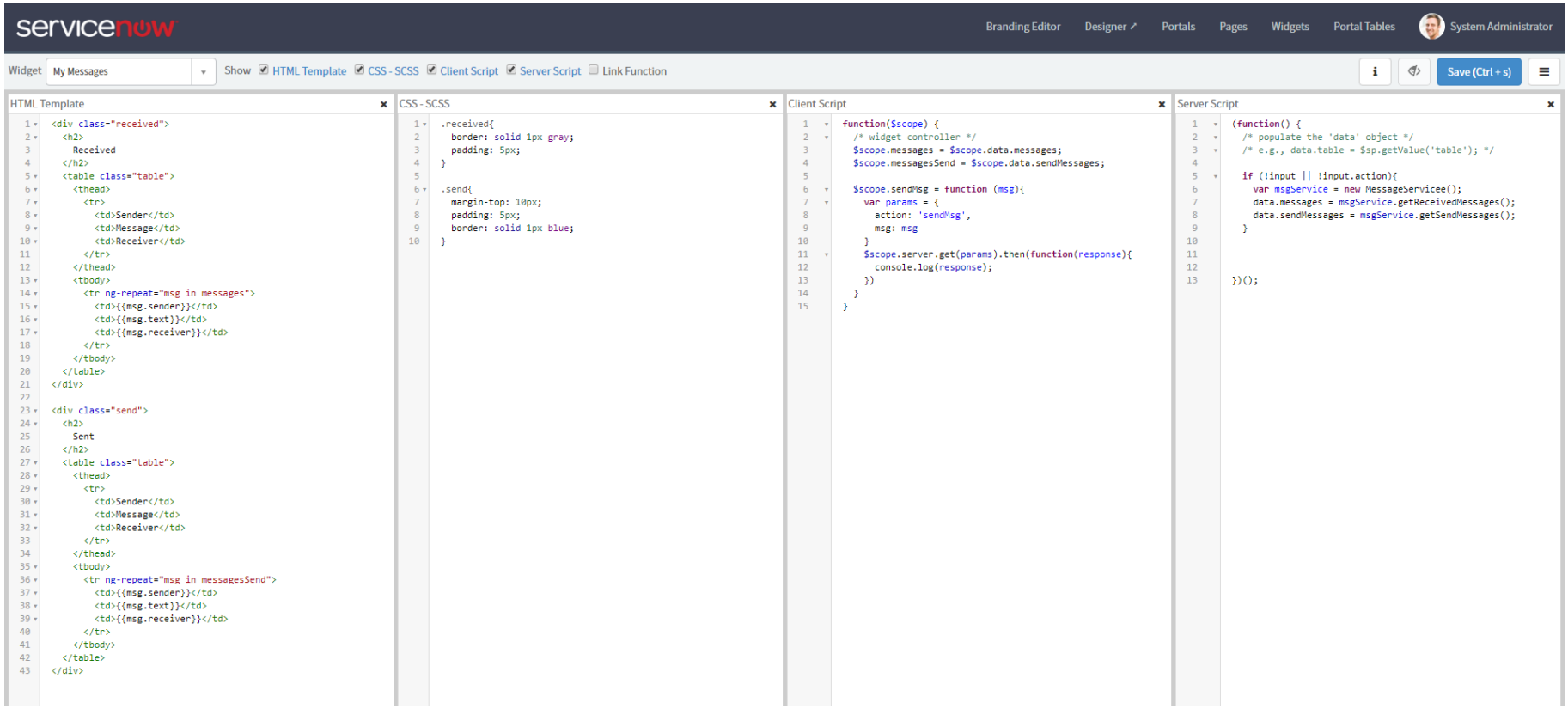

Figure 23: Widget Editor

F: Sync tool class diagram

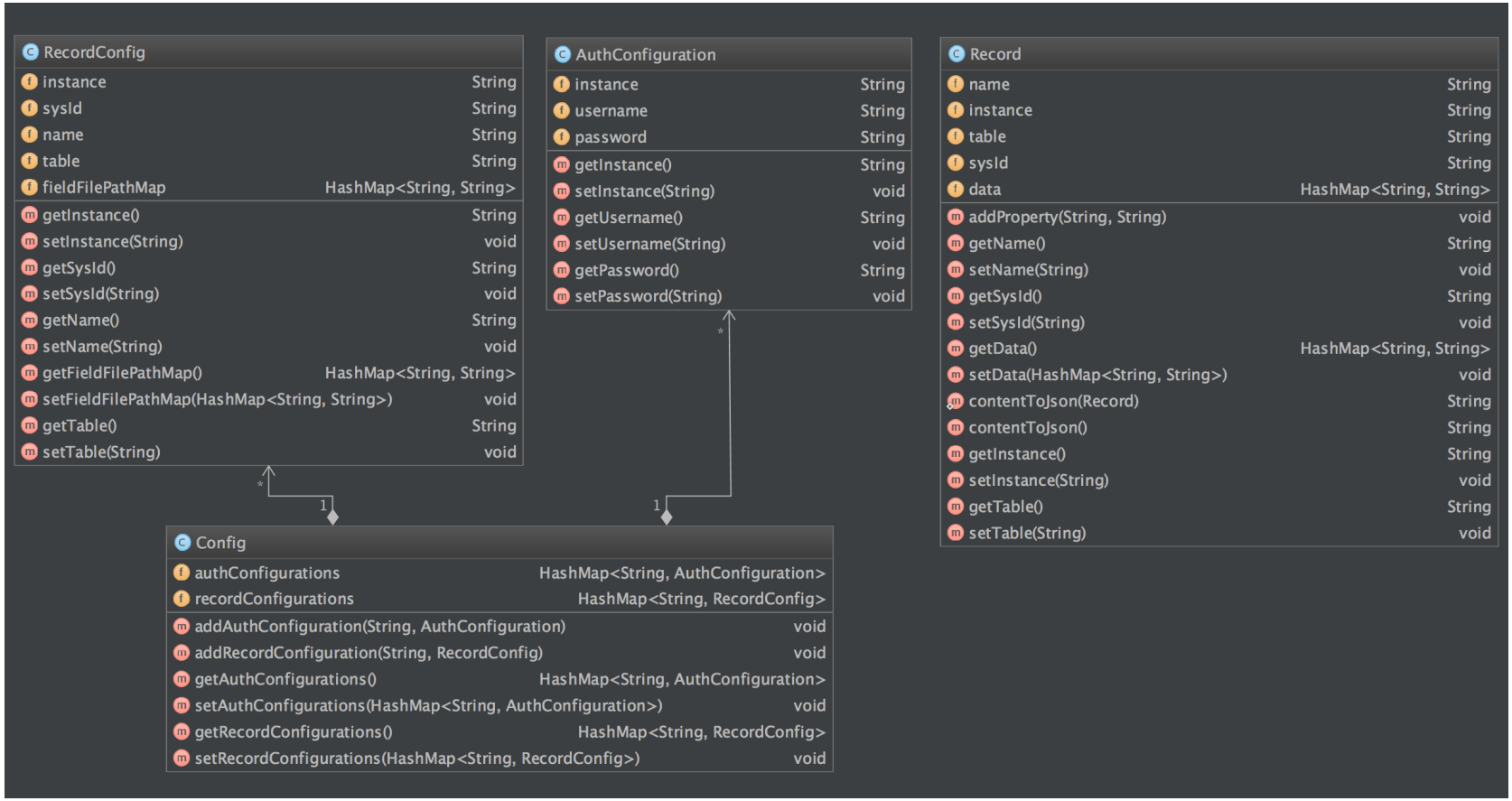

Figure 24: Custom Sync tool diagram

G: Example config file Custom Sync

```
{
  "dependency1": {
    "instance": "plat4mationdemo3.service-now.com",
    "sysId": "293882038j233w",
    "name": "dependency1",
    "fieldFilePathMap": {
      "script": "example_files/uiscripts/dependency1.js"
    }
  },
  "widget1": {
    "instance": "plat4mationdemo4.service-now.com",
    "sysId": "64c67aeedb1103004fe6fabdbf96195e",
    "table": "sp_widget",
    "name": "widget1",
    "fieldFilePathMap": {
      "css": "example_files/widget1/widget1.scss",
      "client_script":
      ↪  "example_files/widget1/widget1.client.js",
      "template": "example_files/widget1/widget1.html",
      "script": "example_files/widget1/widget1.server.js"
    }
  },
  "uiScript": {
    "instance": "dev32138.service-now.com",
    "sysId": "0de95cb8db3503004fe6fabdbf961933",
    "table": "sys_ui_script",
    "name": "uiScript",
    "fieldFilePathMap": {
      "script": "example_files/ui_script/script.js"
    }
  }
}
```

Listing 12: Devsync4u example config

H: SN filesync usage guide

The first step is to create a configuration file. Then the developer can execute the following command:

```
filesync --config app.config.json --search app --download
```

Listing 13: sn-filesync usage

The tool will start using the "app.config.json" configuration file, it will then look for files based on the search specified in the configuration file that is named "app" see listing 8. The argument "app" can be replaced with anything that exists in the configuration file." The search objects can contain encoded queries allowing developers to be very broad or specific in the files they want the tool to download see chapter [25]. The download argument tells the tool it should download the files it finds. It will save them in the folder specified in the roots object see listing 11.

The tool will start watching for changes in the downloaded files if either of the following commands are executed:

```
filesync --config app.config.json
```

Listing 14: Start watching for changes

The resync argument first downloads all the files in the synchronized folder again so the files are up to date.

```
filesync --config app.config.json --resync
```

Listing 15: Start watching for changes resync

To stop watching for changes the command can be canceled using ctrl + c in the command line window the command was started.

[25] See Appendix A ServiceNow components querying for more information on querying.

I: Example config file sn-filesync

```
{
    "roots": {
        "/Users/Dries/Projects/synchronizedProject": {
            "host": "plat4mationdemo4.service-now.com",
            "auth": "REDACTED"
        }
    },
    "search": {
        "mine": {
            "query":
             ↪  "active=true^sys_updated_by=javascript:
             ↪  gs.getUserName()^ORsys_created_by=javascript:
             ↪  gs.getUserName()^ORDERBYDESCsys_updated_on",
            "records_per_search": "100"
        },
        "customer-updated": {
            "query": "sys_customer_update=true",
            "records_per_search": "10000"
        },
        "app": {
            "query": "sys_scope=[APPLICATION SCOPE]",
            "records_per_search": 1000
        },
        "script-includes": {
            "table": "sys_script_include",
            "query": "sys_customer_update=true",
            "records_per_search": "100",
            "download": true
        }
    },
    "preLoad": true,
    "createAllFolders": false,
    "ensureUniqueNames": false,
    "ignoreDefaultFolders": false,
    "debug": false,
    "folders": {
        "business_rules": {
            "table": "sys_script",
```

```
            "key": "name",
            "fields": {
                "js": "script"
            },
            "subDirPattern": "collection/when",
            "_custom": true
        },
        "client_scripts": {
            "table": "sys_script_client",
            "key": "name",
            "fields": {
                "js": "script"
            },
            "subDirPattern": "table/type",
            "_custom": true
        },
        "email_scripts": {
            "table": "sys_script_email",
            "key": "name",
            "fields": {
                "js": "script"
            }
        },
        "inbound_actions": {
            "table": "sysevent_in_email_action",
            "key": "name",
            "fields": {
                "js": "script"
            }
        },
        "processors": {
            "table": "sys_processor",
            "key": "name",
            "fields": {
                "js": "script"
            }
        },
        "script_actions": {
            "table": "sysevent_script_action",
            "key": "name",
            "fields": {
```

```
                "js": "script"
            },
            "subDirPattern": "event_name"
        },
        "script_includes": {
            "table": "sys_script_include",
            "key": "name",
            "fields": {
                "js": "script"
            },
            "_custom": true
        },
        "style_sheets": {
            "table": "content_css",
            "key": "name",
            "fields": {
                "css": "style"
            },
            "_custom": true
        },
        "ui_actions": {
            "table": "sys_ui_action",
            "key": "name",
            "fields": {
                "js": "script"
            },
            "subDirPattern": "table/client_<client>",
            "_custom": true
        },
        "ui_macros": {
            "table": "sys_ui_macro",
            "key": "name",
            "fields": {
                "xhtml": "xml"
            },
            "subDirPattern": "category",
            "_custom": true
        },
        "ui_pages": {
            "table": "sys_ui_page",
            "key": "name",
```

```
            "fields": {
                "xhtml": "html",
                "client.js": "client_script",
                "server.js": "processing_script"
            },
            "subDirPattern": "category",
            "_custom": true
        },
        "ui_scripts": {
            "table": "sys_ui_script",
            "key": "name",
            "fields": {
                "js": "script"
            },
            "_custom": true
        },
        "dynamic_content_blocks": {
            "table": "content_block_programmatic",
            "key": "name",
            "fields": {
                "xhtml": "programmatic_content"
            },
            "subDirPattern": "category",
            "_custom": true
        },
        "email_templates": {
            "table": "sysevent_email_template",
            "key": "name",
            "fields": {
                "html": "message_html"
            },
            "subDirPattern": "collection"
        },
        "fix_scripts": {
            "table": "sys_script_fix",
            "key": "name",
            "fields": {
                "js": "script"
            }
        },
        "catalog_client_scripts": {
```

```
            "table": "catalog_script_client",
            "key": "name",
            "fields": {
                "js": "script"
            },
            "subDirPattern": "applies_to_<applies_to>/type"
        },
        "sys_web_service": {
            "table": "sys_web_service",
            "key": "name",
            "fields": {
                "js": "script"
            }
        },
        "web_service_resources": {
            "table": "sys_ws_operation",
            "key": "name",
            "fields": {
                "js": "operation_script"
            },
            "subDirPattern": "web_service_definition"
        },
        "widgets": {
            "table": "sp_widget",
            "key": "name",
            "fields": {
                "client.js": "client_script",
                "demo.js": "demo_data",
                "server.js": "script",
                "html": "template",
                "style.scss": "css"
            },
            "subDirPattern": "name",
            "_custom": true
        },
        "angular_providers": {
            "table": "sp_angular_provider",
            "key": "name",
            "fields": {
                "js": "script"
            },
```

```
            "_custom": true
        },
        "REST_API_resources": {
            "table": "sys_ws_operation",
            "key": "name",
            "fields": {
                "js": "operation_script"
            },
            "subDirPattern": "web_service_definition",
            "_custom": true
        },
        "scripted_REST_API": {
            "table": "sys_web_service",
            "key": "name",
            "fields": {
                "js": "script"
            },
            "_custom": true
        }
    }
}
```

Listing 16: sn-filesync example config