INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Internal Framework Refactor/Improvement

BRUNO MIGUEL GONÇALVES DOS SANTOS
Outubro de 2018

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Internal Framework Refactor/Improvement

Bruno Miguel Gonçalves dos Santos

Dissertation to obtain the Master Degree in Computer Engineering, Area of Expertise in Software Engineering.

Advisor: Paulo Alexandre Maio

Porto, October 2018

Acknowledgements

I would like to express my deepest appreciation to all those who stood by my side and, directly or indirectly, made the conclusion of this report possible.

Also, I would like to express my gratitude to my parents, who made huge sacrifices to get me where I am today, and to them, I owe the world

To my girlfriend, who didn't let me throw in the towel and was one of my major motivations during this graduation.

I am highly indebted to DEI-ISEP, who gave me the opportunity to develop such a challenging project. I am also truly honoured to be part of this institution and would like to thank all its professionals, who gave me so much and allowed me to grow not only professionally but also as an individual.

To Blip, that made me feel at home since the first day and supported this project in every possible aspect and, more particular, to the V8 team and all its member that supported me with their patience, knowledge and advice.

To V8 Team, the team that made all this possible. The team whose elements were inspirations, drivers and whose knowledge helped me to overcome many challenges.

Last, but not least, I would like to thank my supervisors and mentors, Professor Paulo Maio, Ricardo Canastro and Rui Lima, who guided me through this journey and whose advice led to this amazing experience.

To all of you, my deepest gratitude. This is the work of all of us.

Abstract

The FCN (Fusion CoNsole) is a framework created with the purpose of providing developers with the rapid development of internal management applications that bundles solutions for common requisites in medium to large companies, reducing the setup boilerplate for the teams. The developed applications are available under a single hostname to provide its users with a single point of access.

However, the current version of the FCN has several problems and bottlenecks that resulted from the lack of opportunity to solve them and due to the technology stack used becoming outdated. Such problems sometimes lead to time expensive processes and dependencies that, otherwise, could be avoided. From these needs, IAP – Internal Applications Platform – is born. IAP goals are to be a new version of the FCN where the raised problems are solved.

The purpose of this thesis is to raise, understand, and solve the current problems of the framework. Such was achieved through a deep analysis of the current architecture, technologies that could be used to solve them or enhance the current version and to study and design solutions that allow to overcome them.

It is hoped that the presented solutions not only solve the raised problems but improve the overall usability for developers and applications' users.

In the end, the presented solution not only solves the raised problems as it improves IAP's usability for software developers and users of the applications built on top of it.

Keywords: Framework, Management, Platform, Improvements

Resumo

O FCN (Fusion CoNsole) é uma framework criada com o propósito de fornecer aos desenvolvedores de software um rápido desenvolvimento de aplicações de gestão interna que contém soluções para requisitos comuns em médias e grandes empresas, reduzindo todo o trabalho inicial de configuração de uma nova aplicação. As aplicações desenvolvidas com o FCN estão disponíveis através de um único *hostname* de forma a garantir um ponto de acesso único.

Apesar de poder ser visto como apenas uma coleção de componentes e ferramentas para desenvolvimento *frontend*, o FCN tenta, não só, promover o padrão da utilização e criação de componentes *web* reutilizáveis, bem como tenta facilitar o desenvolvimento de tais componentes utilizando um único tema definido pela empresa. O FCN é, ainda, implantado sobre uma plataforma que apresenta uma arquitetura de micro-serviços desenvolvida internamente. Tal permite quer a adição quer a utilização de serviços disponíveis para compor aplicações que tenham como objetivo vir a ser usadas por toda a empresa.

Uma aplicação desenvolvida com o FCN, terá uma aparência e experiência de utilização semelhante a todas as aplicações desenvolvidas dentro da empresa, o que torna a experiência mais suave e ameniza a mudança de contexto entre as aplicações.

Contudo, a versão atual do FCN apresenta vários problemas e limitações que resultam da falta de oportunidade para a resolução dos mesmos e devido à utilização de tecnologias desatualizadas. Problemas esses que levam, muitas vezes, a processos mais lentos e a dependências indesejadas que, caso contrário, poderiam ser evitadas. Nasce desta necessidade o IAP – Internal Applications Platform. O IAP tem como objetivo ser uma nova versão do FCN em que os problemas identificados estão resolvidos e/ou minimizados.

Esta tese tem como objetivo o levantamento, a compreensão e a resolução dos problemas da atual framework. Tal foi alcançado através de uma profunda análise da arquitetura atual, das tecnologias que podiam ser utilizadas para ajudar a resolver e melhorar tais problemas e do estudo e design de soluções que possibilitem a sua resolução.

No final, a solução apresentada neste documento não só resolve os problemas levantados como também melhora a usabilidade para os desenvolvedores de software e utilizadores das respetivas aplicações.

Palavras-Chave: Framework, Gestão, Plataforma, Melhorias

Table of Contents

Index of Figures

Index of Tables

Index of Codes

Acronyms and Symbols

Acronyms List

AD	*Active Directory*
AHP	*Analytic Hierarchy Process*
AIS	*Admin Identity Service*
ATZ	*Authorization Service*
AWS	*Amazon Web Services*
BMU	*Booking Mapping UI*
CDN	*Content Delivery Network*
CI	*Consistency Index*
CMS	*Content Management System*
CPU	*Central Processing Unit*
CR	*Consistency Ratio*
CSS	*Cascading Style Sheets*
DOM	*Document Object Model*
DRY	*Don't Repeat Yourself*
FCN	*Fusion CoNsole*
FMU	*Feature Management UI*
GCP	*Google Cloud Platform*
HTML	*Hypertext Markup Language*
IaaS	*Infrastructure as a Service*
IAP	*Internal Applications Platform*
IDE	*Integrated Development Environment*
ISS	*Identity Single Sign-On*
JS	*JavaScript*

JSON	*JavaScript Object Notation*
MVC	*Model – View – Controller*
NCD	*New Concept Development*
PaaS	*Platform as a Service*
PoC	*Proof of Concept*
RAM	*Random-Access Memory*
RI	*Random Index*
SaaS	*Software as a Service*
SASS	*Syntactically Awesome Style Sheets*
SLA	*Service-Level Agreement*
SSO	*Single Sign-On*
UI	*User Interface*
UML	*Unified Modelling Language*
UTC	*Universal Time Coordinated*
UX	*User Experience*
W3C	*World Wide Web Consortium*

1 Introduction

This chapter aims to allow a better understanding of the framework that is being refactored and improved. It starts with the contextualization of the project, where it is presented the purpose of such a framework.

A list of the problems of the current framework is further presented, exposing where the need for this project came from.

Then, it is presented the goals that this project aims to reach as well as the expected outcomes defined by the company, Paddy Power Betfair, in which the project runs. This chapter ends with a description of the structure of this report, so the reader can easily read through it.

1.1 Context

The FCN – Fusion CoNsole – was born in 2014 with the purpose of creating a developing framework where any internal management application, from the Paddy Power Betfair company, could integrate, leveraging its modules for look and feel, authentication, logging and notification. The Fusion CoNsole framework aims to be a unified way to develop and aggregate all the internal management applications into one single website, providing a single point of access. The applications that are integrated into the FCN support many core business processes of the company such as, for instance, content management. To better understand the kind of applications that are under the FCN scope, some of them are briefly described.

AmTote Race Off is an application that lets its users select horse races from the United States and Australia and be notified when one of the selected races starts. Dial-a-Bet is used by the company's customer services to place the bets that the users request by the phone. Gandalf, the company's Content Management System (CMS), is used to control and provide content to the Paddy Power websites. Competition Mapping, Feeds Entity Manager, Selection Mapping and others are tools that the company's traders use to map feeds, sports, competitions,

events, markets and others between each other's and different services. In this context, a sport contains several competitions, each competition contains a series of events that contain a series of markets, which are where the users bet on. Feeds are responsible to provide information for each of these entities.

From a developer point of view, the FCN framework was created with the purpose of being time-saving for developers, to provide common functionalities that can be used across all the applications and to ease the developer work in any application. By having all the applications following the same code standards and structure, any developer is able to work in any of the applications without a painful learning curve.

From a business point of view, such framework provides less context switching for the end users, since all the management applications are under the same website and with a consistent look-and-feel which greatly improves the user experience.

However, a lot of problems were not approached in this first version of the framework and a lot were raised due to the technological stack becoming outdated. Next section provides a deeper look into the problems of the current framework version.

1.2 Problem

In this section, the problems of this framework are enumerated. For each one of them, there is a brief explanation of its cause. These problems were identified based on (i) a survey sent to the developing teams working with FCN; (ii) existing FCN-based applications were studied; as well as (iii) the current technological stack was analysed.

i) The teams have no independent release process

First, it is important to clarify and distinguish the concepts of deployment and release. As used in this work, a release of an application consists of the installation and configuration of the new application version in the infrastructure. Instead, a deployment consists in making such version available to the public [1].

At the time of writing this document, every time an application has a new change that needs to be deployed, all the applications that are aggregated in the Fusion Console Framework need to be released. This is a painful process since a release now takes an enormous amount of time. This impacts the time to market and urgent fixes which can degrade the end user experience. This is the major priority of the company.

ii) Complex theme scheme (look and feel)

The themes that the framework offers are organized in a complex and unclear way. The themes live in a central place where the CSS variables are defined and that every component of the framework depends on. Moreover, the final CSS bundle is generated inside these

themes, which means that the themes have a dependency on every component from the framework to fetch the CSS from each one of them and add it to the bundle.

iii) Technological Dependencies

The Fusion Console Framework has around three years and in this time frame, a lot of alternatives have arisen and a lot of these technologies have become outdated. There needs to be a study to understand if there are better alternatives to some technologies, or if there is the need to use some of them at all.

The current framework is built on top of the client side framework *AngularJS* 1.5. This forces every application inside the Fusion Console Framework to use that as well. Not only it is an outdated framework but it castrates the technological independence that the teams are supposed to have.

Another problem that leads to having forced technological dependencies is that there is no version management or maintenance of some applications. Such forbids the upgrade of many technologies because there are dependencies that could break those applications.

iv) Documentation

One problem raised by the developers that use this framework is the documentation. Not only it is scarce but it misses some important details that can difficult the developer work. This can lead to cases where a developer is forced to implement something that already exists or where it spends an unnecessary amount of time searching for implementation details.

The current documentation consists of outdated and incomplete guides on how to use and integrate the different components from the framework and has almost no diagrams that represent the whole system and/or some critical fragments. Such bottleneck in documentation can make the boilerplate and adaptation to the project a painful process.

v) Authentication and Authorisation

Once a user successfully authenticates into the Fusion Console Network website, it has instant access to every application. If there is the need to restrict accesses to an application, it must be that application to implement an authorisation system. This is one of the problems resulting from the lack of an authentication system. Such can lead to unauthorised access to critical company information, violating the company privacy rules.

The company has developed an authentication and authorisation service, Admin Identity Service (AIS), that is only being used to achieve user authentication.

vi) All the Applications Run on the Same Machine

Currently, all the applications of the Fusion Console Framework run on the same machine, which can lead to resource starvation, affecting the availability of others applications.

vii) Logging

The framework lacks the implementation of a logging system. Such can difficult the detection of problems, which can impact the resolution in time of problems.

viii) Lack of Conventions Between Projects

After a brief analysis of some projects that are inside the Fusion Console Framework, it is possible to identify a lack of development conventions between them. Some projects have well-defined conventions while others do not have any. Even between those that have them, they are not identical. This may be caused by the lack of communication and discussion between the users and developers of the framework to reach a consensus in which are the best conventions and standards to be followed.

For instances, the *linting* rules between projects, and even between components of the same project, tend to be different. The project structure, like folders and file naming, for projects with similar purposes or approaches, tend to be another divergent point.

ix) Lack of User Interface Consistency

In the development of this thesis, a new problem gained emphasis. The lack of user interface consistency was something that was well identified but was not a priority. However, since there was a rebranding in the middle of the process, this problem came to light once again.

1.3 Goals

The main goal of this work is to develop/prototype a new version of the FCN framework. This new version is named IAP, Internal Applications Platform. The objectives are to address the problems stated in the previous section, but following the company's technical roadmap, where the major problems are prioritized according to the company's goals.

The goals can be divided into two groups: (i) the strategical work and (ii) the tactical issues. The strategical work is the goals that the company has defined as a priority (independent releases, separate clusters per application, authentication and authorisation). After the strategical work is completed, the tactical issues will be addressed. Inside the tactical issues, are the issues defined in the previous point, such as the refactor of the complex theme scheme, documentation, conventions and the remaining ones.

Besides addressing all the problems raised previously, there are more goals that are desired to be achieved such as allowing that existing apps developed based on the current framework can coexist with the apps that will be developed based on the new version, to turn the website into a responsive progressive web app and to have functional and unit tests coverage.

Such goals require a deep analysis of the current framework to understand how it can be improved. There is also the need to research and design different solutions for the several raised problems. New technologies must be considered.

1.4 Expected Outcomes

The company has established the technical roadmap depicted in Figure 1, which represents a high-level planning of the work to be done. The following content of this section is based on the company's planning made by Jorge Silva, one of the Paddy Power Betfair software architects.

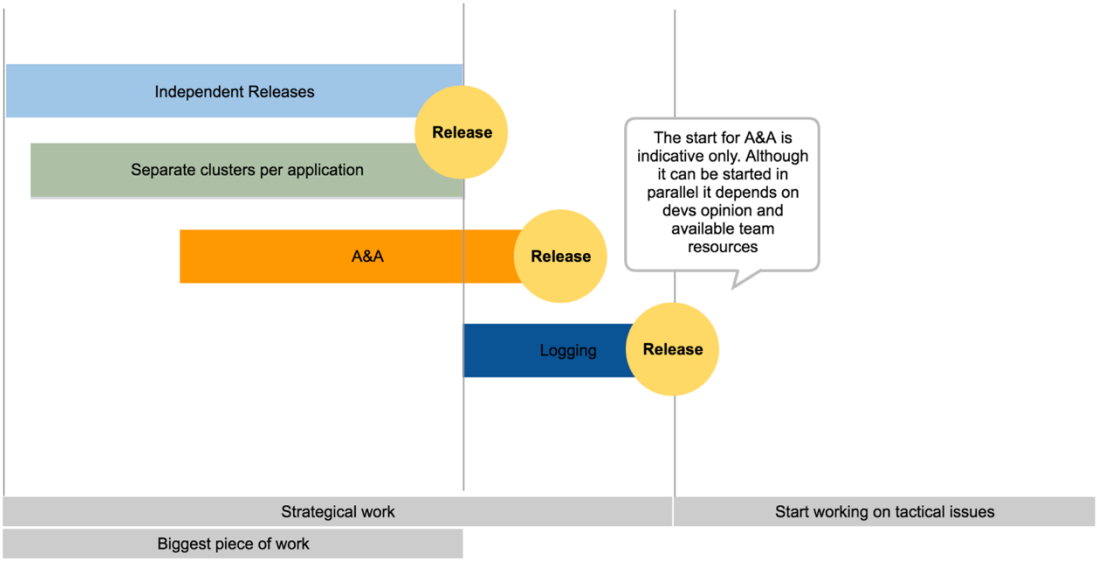


Figure 1 – Technical Roadmap (Jorge Silva, Blip Software Architect)

The proposed technical roadmap has the goal of tackling the pressing issues first to make room to work on the minor ones with proper bandwidth. The issues are ordered by importance.

Independent releases and separate clusters per application can - and should - be tackled in parallel as they are tightly correlated. From a delivery perspective, this is a big piece of work and imply some changes to all existing applications.

Authentication, Authorisation and Logging are disjoint and can also be tackled in parallel. These are smaller pieces of work, very contained in scope.

Ideally, Logging is done after Independent releases and separate clusters per application to skip the resource starvation issue.

1.5 Document Organization

This section exposes the structure of this document and the topics that are addressed.

Chapter 1 contains the contextualization of the project, a list of the problems of the current framework, the expected goals and the expected outcomes defined by the company.

In Chapter 2 it is described the state of the art of the framework that is being refactored. There, a series of problems and bottlenecks are raised and the reasons behind them are explained.

Chapter 3 contains a state of the art applications where applications with similar goals are explored. Technologies that can be used to develop IAP are also analysed.

Chapter 4 describes the business where this project is inserted into, by exposing the main concepts, actors, processes, and restrictions. A list of functional requirements and non-functional requirements is also raised.

In Chapter 5, a detailed value proposition is presented so that it is understood what is the value that is being delivered to the clients

Chapter 6 carries the analysis and design of the proposed solutions to solve each one of the raised problems.

In Chapter 7 it is described how the technical solution developed in Chapter 6 was implemented.

Chapter 8 describes the experiments carried out to evaluate some particular goals and requirements. For each, a set of metrics, methodologies, results and the respective analysis are presented.

Chapter 9 sums up the developed worked in this project and does a retrospective about the goals and requirements that were achieved.

Finally, the Annexes section, containing supplementary information to this dissertation.

2 Current State of the FCN

To understand how the raised problems appeared and how can they be tackled, first, it is important to study the state of the art of the current solution. For that, and to keep the analysis focused on the raised problems, such studied is divided by problem. For each problem, the current architecture and/or proof of its existence are presented.

2.1 Independent Releases

One of the most painful processes in FCN is the releases ones. Every time a new application needs to be released, all the other versions need to be released as well. To better understand what a release implies and how the FCN serves the several applications, Figure 2 represents the core of the framework.

Enumerating each component of the Figure 2, (i) the Application A represents an application developed using FCN; (ii) Hogan.JS is a compiler for the templating language Mustache [2]; (iii) the Template Strand and (iv) the FCN Website, that is the single point of access to the applications. Figure 3 represents how the components interact amongst themselves.

To clarify what a strand is, a strand is an aggregation function that is run by Fabric [3]. Fabric is a nodejs middleware, developed and used internally by Paddy Power Betfair, that was created with the purpose of running small IO (Input/Output) bound data collation functions. The main goal of this middleware is to allow the creation of aggregation functions in an easier and faster way.

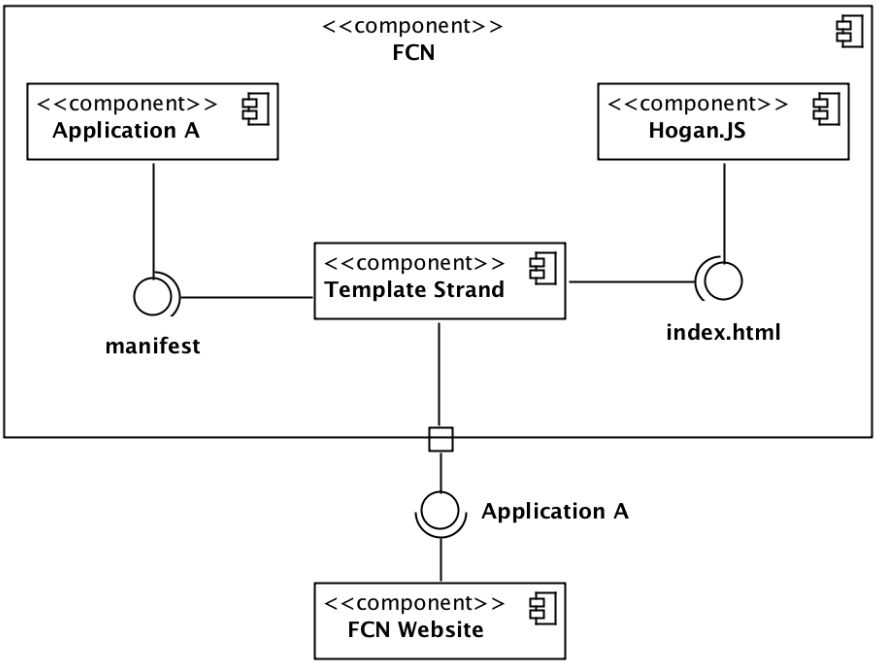


Figure 2 – FCN Releases UML diagram

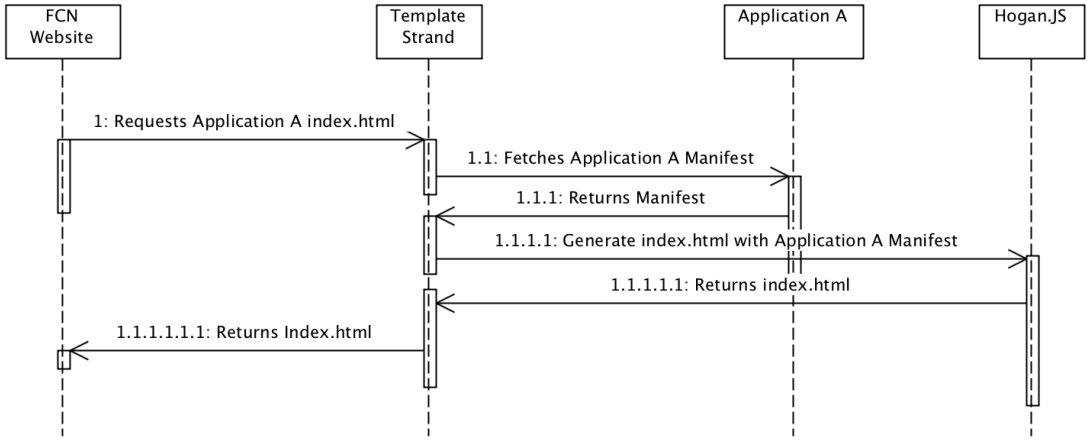


Figure 3 – FCN Releases UML Sequence Diagram

The problem with this approach is that all the applications live inside FCN. Rephrasing, if a new version of an application needs to be released, the FCN must be released as well and, with that, all the applications that are part of the FCN.

To top all this, there is the dependencies' version problem: all applications are tied to fabric and applications tend to get (team) orphaned once they're finished, and, when a major version is released, the whole platform cannot upgrade unless changes are made across all the applications to guarantee backward compatibility, and to top this, if another application depends on the latest fabric version, the upgrade is mandatory and the cost is as great as the

number of applications with fabric dependent micro-services. The version problem spreads to the FCN functional core, which is dangerous to change due to the same set of problems.

2.2 Applications Running on the Same Machine

Currently, all the applications are deployed into the same set of machines. The rationale behind this decision was that if applications are a collection of static files processed in the clients', they can all live under the same set of machines. However, that is not entirely true.

Applications come with a set of micro-services attached in order to support its needs. Joining all the applications and respective micro-services into the same set of machines can cause resource starvation which can lead to the unavailability of all the services since they are sharing the same physical resources. Moreover, since the deployment of applications to the different environments is sequential (due to the FCN having to deploy all the applications, as stated in subsection 2.1), if one of the applications' deployment fails, the entire process breaks.

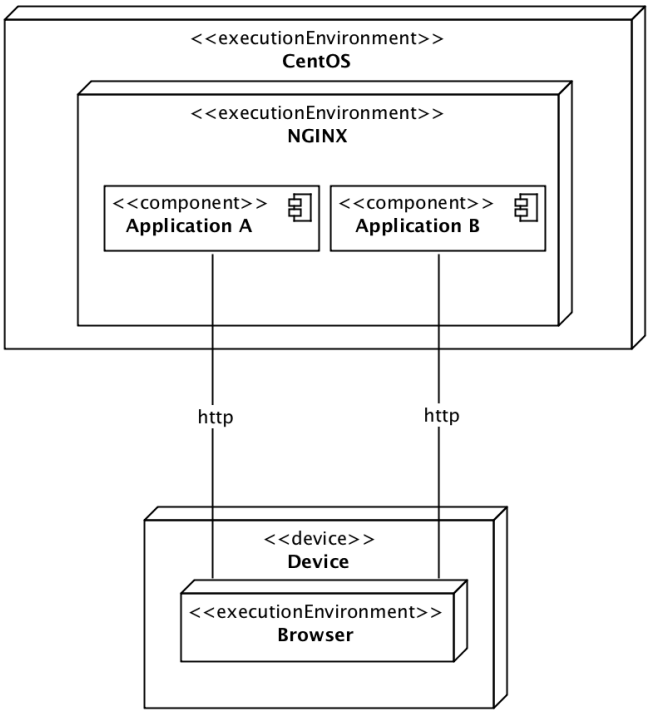


Figure 4 – FCN State of the Art UML Deployment Diagram

2.3 Complex Theme Scheme

As stated in section 1.2, the problem (ii), the theme scheme of the FCN is complex and unclear. Figure 5 represents how are the themes managed in the FCN framework.

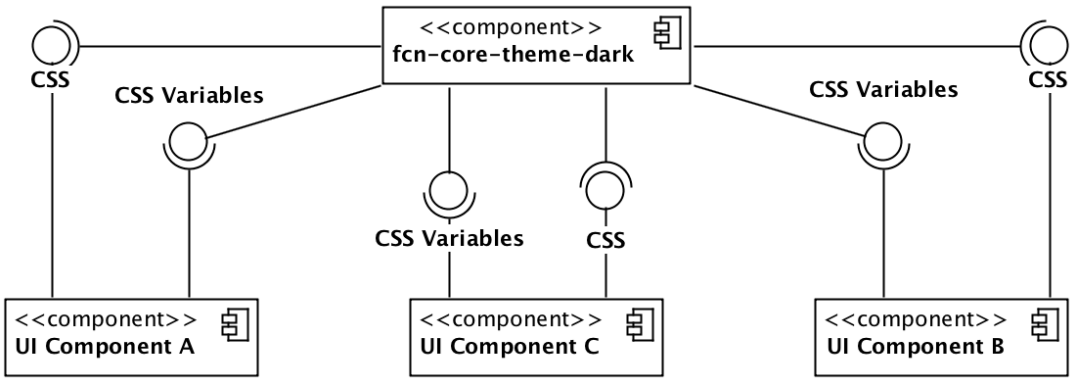


Figure 5 – Current architecture of the theme scheme

The themes live in the *fcn-core-theme-dark* component where the CSS variables are defined and that every component of the framework depends on (Component A and B serve as an example). The final CSS bundle is generated inside the *fcn-core-theme-dark* component, which means that the themes have a dependency on every component from the framework to fetch the CSS from each one of them and add it to the bundle.

Such an approach leads to circular dependency and a symbolic link nightmare which makes the development and maintenance of the themes a painful process. It is mandatory to study an approach that not only removes the circular dependency but allows the creation of new themes to be an easy process.

To demonstrate the complexity of such scheme, Code 1 represents some dependencies of the *fcn-core-theme-dark* components.

```
// FCN CORE BUTTONS
@import 'overrides/fcn-core-buttons/fcn-button-properties';
@import 'overrides/fcn-core-buttons/fcn-checkbox-properties';
@import 'overrides/fcn-core-buttons/fcn-dropdown-button-properties';
@import 'overrides/fcn-core-buttons/fcn-toggle-properties';
@import 'overrides/fcn-core-buttons/fcn-counter-properties';
@import 'fcn-core-buttons';

// FCN CORE INPUTS
@import 'overrides/fcn-core-inputs/fcn-input-box-properties';
@import 'overrides/fcn-core-inputs/fcn-input-checkbox-properties';
@import 'overrides/fcn-core-inputs/fcn-dropdown-list-properties';
@import 'overrides/fcn-core-inputs/fcn-input-datetime-properties';
@import 'overrides/fcn-core-inputs/fcn-input-radio-properties';
@import 'overrides/fcn-core-inputs/fcn-textarea-properties';
@import 'fcn-core-inputs';
```

Code 1 – Source code of the *fcn-core-theme-dark* with some of its dependencies

It is possible to observe that the *fcn-core-theme-dark* has dependencies over the *fcn-core-buttons* and the *fcn-core-inputs*. When looking at the source code of both components (Code 2 and Code 3), it is possible to see that they both depend on the *fcn-core-theme-dark*.

```
var paths = [
 path.join('node_modules', 'fcn-core-vendor', 'dist', 'styles'),
 path.join('node_modules', 'fcn-core-styles', 'assets, 'styles'),
 path.join('node_modules', 'fcn-core-theme-dark', 'theme', 'overrides, fcn-
core-buttons'),
 path.join('node_modules', 'fcn-core-theme-dark', 'theme'),
 path.join('assets', 'styles')
];
```

Code 2 – Source code of the *fcn-core-buttons* with some of its dependencies

```
var paths = [
 path.join('node_modules', 'fcn-core-vendor', 'dist', 'styles'),
 path.join('node_modules', 'fcn-core-styles', 'assets, 'styles'),
 path.join('node_modules', 'fcn-core-theme-dark', 'theme', 'overrides, fcn-
core-inputs'),
 path.join('node_modules', 'fcn-core-theme-dark', 'theme'),
 path.join('assets', 'styles')
];
```

Code 3 – Source code of the *fcn-core-inputs* with some of its dependencies

The scenario depicted throughout these samples (Code 1, Code 2 and Code 3) is presented in every visual component of the FCN.

2.4 Technological Dependencies

There are several crucial technologies used in FCN for developing applications:

- **Application**
  - Angular 1.5 – Client Side Framework
  - SASS - Syntactically Awesome Style Sheets
  - Jade (currently known as Pug) – Template Engine
  - Gulp – Toolkit for tasks automation
  - Karma – Test Runner for Javascript
  - Mocha – Javascript Test Framework
  - Sinon – "Standalone test spies, stubs and mocks for JavaScript" [4]
- **Automation**
  - Jenkins – Automation server for Continuous Integration and Continuous Delivery

- Go Pipelines - Automation server for Continuous Integration and Continuous Delivery
- GitLab – Git repository

- **Persistence**
  - Artifactory - Universal artefact repository manager

- **Infrastructure**
  - OpenStack – Platform used for cloud computing
  - NGINX – Web Server
  - CentOS – Linux Distribution

One problem with the technological stack is that FCN forces all applications to use a specific version of Angular and its dependencies. This happens because the template that generates the index.html of the applications uses Angular to load them. The code present in Code 4, which is a sample of the code present in the template file, shows how applications are loaded.

```
<script>
angular
.module("fcn", [
        "ui.router",
        "fcn.core.templates",
        "fcn.core.config",
        "fcn.core.services",
        "fcn.core.storage",
        "fcn.core.modals",
        "{{ module }}"
    ])
     .constant('fcn.apps', {{{ apps }}})
     .constant('fcn.registry.active', "{{{ active }}}")
     .constant('fcn.registry', {{{ fcn }}})
</script>
<script src="/core/scripts/app.@@BUILD.js"></script>
{{{ scripts }}}
```

Code 4 – *index.mustache* template sample with the loading of the applications

Angular is used to load several modules that the apps depend on (*fcn.core.templates*, *fcn.core.config*, amongst others) as well as the apps themselves. There is no way to avoid building an FCN application without Angular, which constrains the teams' decisions regarding which framework/technologies to use.

2.5 Documentation

Documentation is a major source of information for developers to easily understand and use the components provided by the FCN. However, the lack and the outdated documentation not only leads to a more time-consuming boilerplate as it leads to the wrong usage of such components. The Figure 6 and Figure 7 shows an example of a component, *fcn-core-buttons* (it provides different types of buttons), that the last update, at the time of this writing, was made in the 23rd of June of 2017 but last change made to the documentation dates to the 24th of August of 2016.

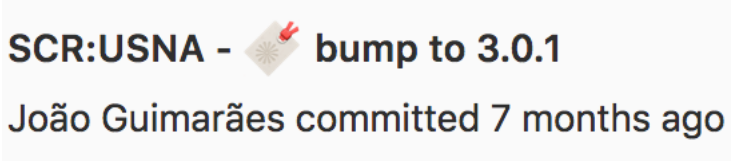


Figure 6 – Last commit made to the *fcn-core-button*

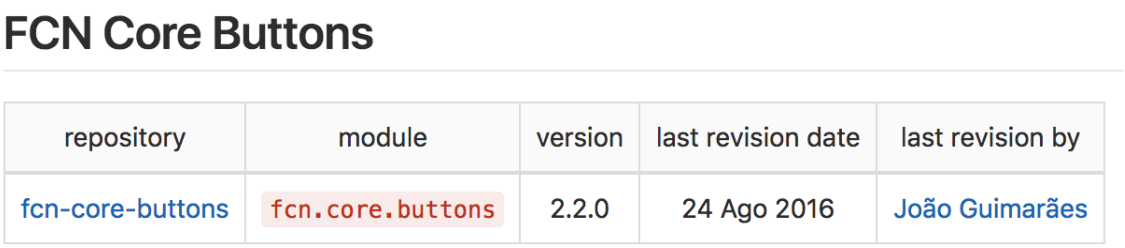


Figure 7 – Last change made to the *fcn-core-buttons* documentation

This kind of discrepancy can lead to changes that are not reflected in the documentation and when a developer tries to follow it in order to use a component, it may face some difficulties.

Another problem is the lack of documentation. One example can be found in the *fcn-core-theme-dark*. There is no documentation explaining how the themes are managed across components and applications, which leads to blind development and extensive hours of research.

2.6 Authentication and Authorisation

Currently, a user that can authenticate into the FCN is able to access every application that the FCN provides unless the application provides an authorisation mechanism. Figure 8 represents an initial draft from 2015 of the authorisation and authentication where authorisation mechanisms were contemplated through an authorisation service, ATZ (Authorization Service).

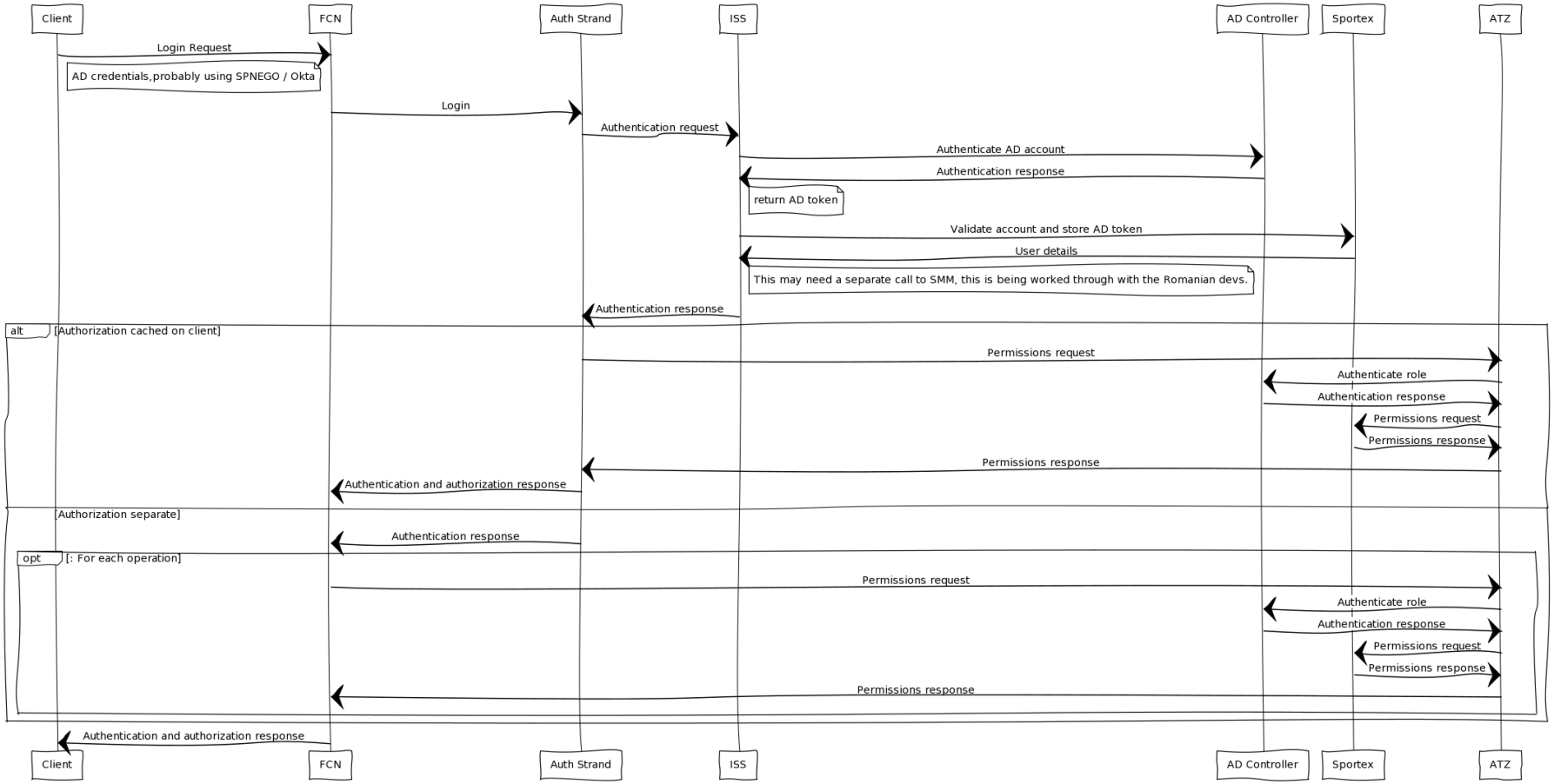


Figure 8 – Initial draft for authentication and authorisation

The Fusion Console calls an authentication strand run by fabric that performs an authentication request to the ISS (Identity Single Sign-On). The ISS, then, authenticates the account using the AD (Active Directory) Controller that returns the AD Token. Once the ISS retrieved the token, it would send it to Sportex (where the users' details are stored) in order to validate the account, store that same token and return the details from that user. The authentication process is fully implemented and functional.

However, and although there were plans to use the ATZ service to perform the authorisation of the users inside the FCN, such was never implemented, which leads to the problem of having no control over who accesses what applications. This can violate privacy rules, exposing sensitive data to personnel who are not authorised to do so.

2.7 Logging

There is no logging whatsoever implemented in the FCN platform. Currently, the services provided by the FCN, being frontend or backend services, do not have any logging mechanism the allows them to be stored and to keep track of the system behaviour. In case of failure or hard to track errors, logging can provide a precious footprint to what may have caused the problem.

There were plans to develop a strand that could be consumed by the applications under FCN to log the frontend errors that occurred when users interacted with the application. Since errors that occur in the browsers are hard to track, this strand would provide persistence for such errors, making it easier to track and debug them. Nevertheless, such strand was never implemented.

2.8 Conventions Between Projects

FCN had as a goal the definition of a series of conventions that should be adopted across projects to reduce the boilerplate for the developers when working with several applications. However, few conventions were agreed upon and even those that were, are not being implemented across projects.

One example is the existence of an FCN component that provides several formatting rules for the code of the applications. This component, *eslint-config-fcn,* can be extended by the applications in order to keep the code formatting according to the established rules. Yet, very few projects adopt such rules.

There is also the lack of naming conventions and folders structure. This can difficult the navigation inside a project when a developer is used to a certain organization.

This is a controversial point. Different technologies, different projects and different teams can have different needs regarding such aspects. If, on one hand, the goal is to give teams more

freedom of choice, forcing conventions can undermine such goal. There is the need to understand what conventions could be used across projects without constraining the teams' decisions.

2.9 Lack of User Interface Consistency

One of the goals of FCN and IAP consists in reducing the context switching of the website users. To achieve that, it is important that the user interface keeps being consistent across applications.

However, FCN fails completely regarding this matter. Figure 10, Figure 9, Figure 11 and Figure 12 represent four different applications available in FCN. It is clear that there is very little consistency between them. Even the header, that is supposed to be equal in every application, is different in one of them. It is also possible to verify that there are indeed some components that are being shared (the buttons for instance in Figure 10 and Figure 12) but even that is only a glimpse of what was desired.

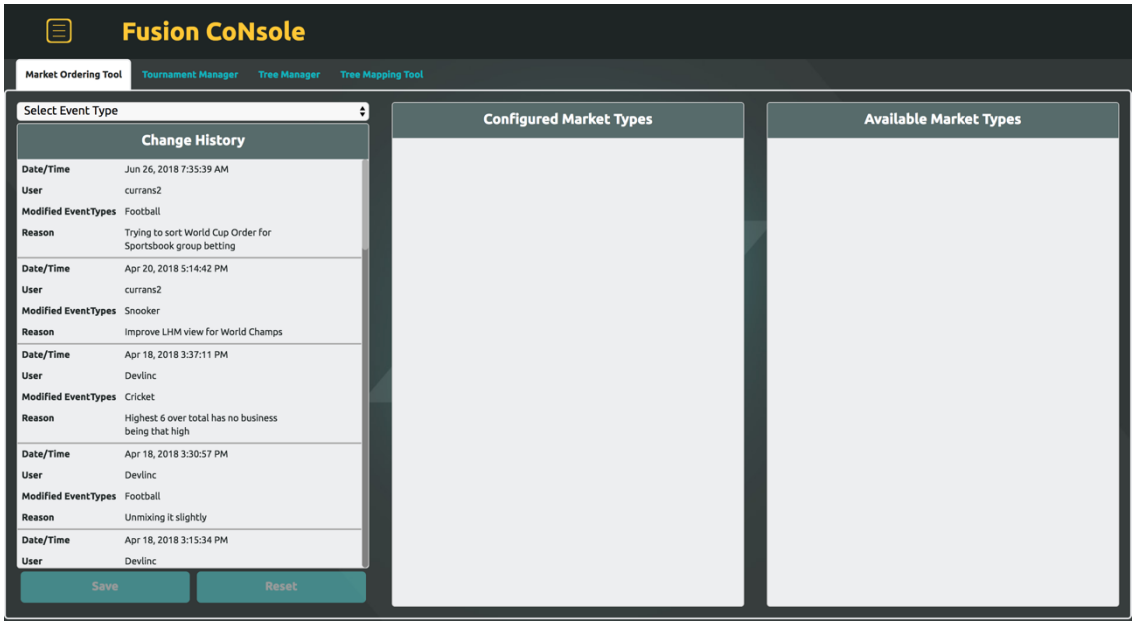


Figure 9 – Catalogue Manager Application

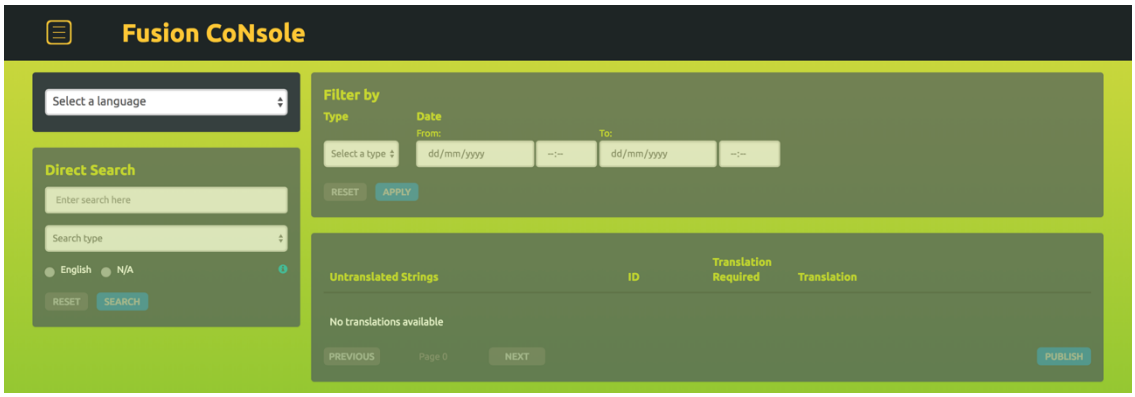


Figure 10 – Localisation Tool Application

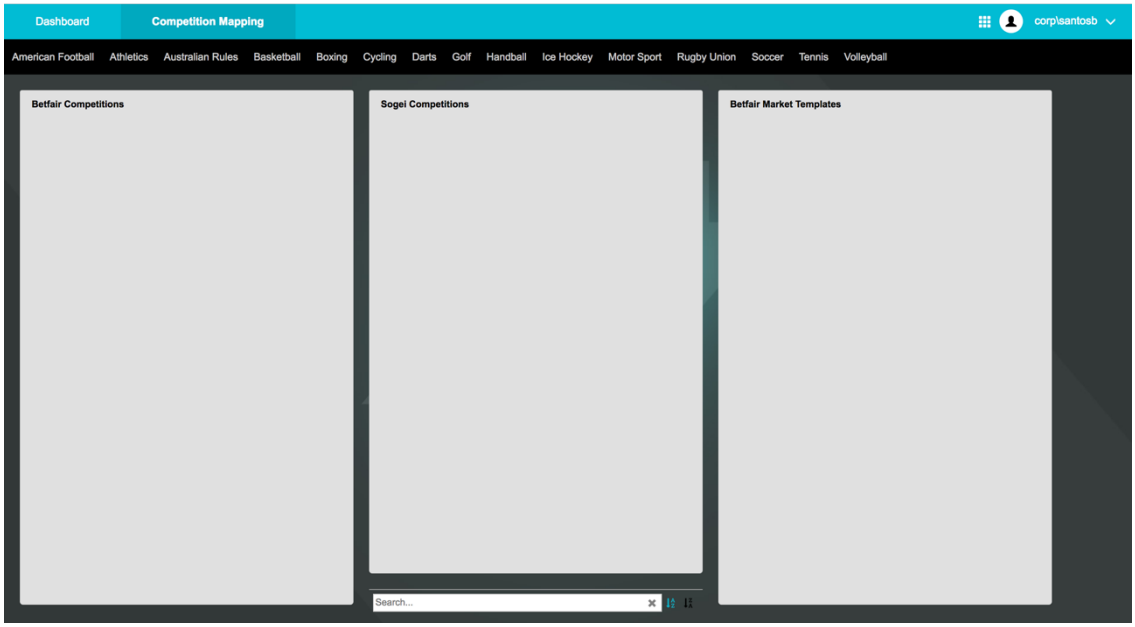


Figure 11 -Competition Mapping Application

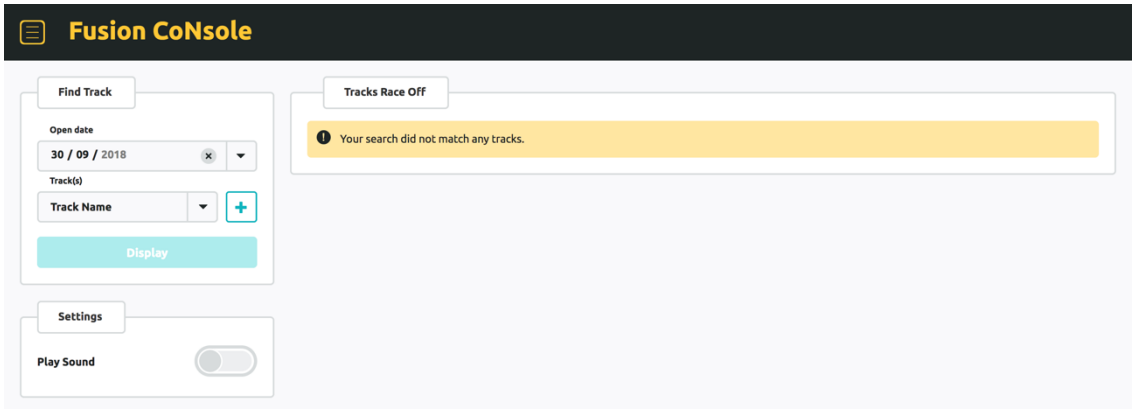


Figure 12 – AmTote Race Off Application

This lack of consistency completely defeats the reduction of context switching. However, since this kind of work is the responsibility of the UI (User Interface) and UX (User Experience) teams. Nevertheless, in subchapter 3.1, there is a study of the UI and UX of existent solutions. This study was shared with the UI and UX teams so they could create a new design for IAP.

3 State of the Art

This chapter begins with the analysis of some existent solutions by understanding what services they offer, how can an application leverage on such services and how is the user experience while navigation through such services. These solutions not only are an inspiration to the current FCN version but they also are an inspiration to IAP.

Two of the problems of the current framework is the outdated technologies dependencies and the complex theme scheme to maintain the look and feel between components and applications. Two major distinct approaches can be followed: the use of a client-side framework that would be enforced upon all applications or the use of web components that would give the teams more freedom when choosing the technologies for their applications. Nevertheless, there is the need to understand what technologies could be used in each case.

After an individual analysis of each client-side framework, a comparison between them using (i) market acceptance and support; (ii) components complexity; (iii) templating; (iv) state management and data binding; (v) learning curve; and (vi) size and performance.

The same is done for the web components libraries, where they are compared based on (i) browser support; (ii) development environment; (iii) performance; (iv) and integration with client-side frameworks.

3.1 Existent Solutions

There are three major solutions in the market that have the same purpose as the FCN but, while they are made to support any kind of business in several fronts, the FCN is built to support the management operations inside the company's business. They also serve their applications through a single website, keeping the look and feel between them. These solutions are an inspiration for the current version and to the new one.

This analysis plays an important role in the all project since FCN was strongly based on these solutions and bringing the new solution closer to these ones helps to improve the engagement with users and developers. Understanding what kind of services are offered by these solutions and how can they be adapted to fit the IAP purposes is crucial.

Moreover, understanding how all these services are organized, provided to the users and the look and feel of the overall platform, can help to mitigate one of the major problems of the FCN, the complex theme scheme and consequent look and feel disruption. The three presented solutions are market leaders and understanding the reasons why can greatly improve the overall Paddy Power Betfair adoption of the IAP.

3.1.1 Amazon Web Services

Amazon Web Services, or AWS, is a business unit of Amazon that "provides a highly reliable, scalable, low-cost infrastructure platform in the cloud that powers hundreds of thousands of businesses" [5]. Amazon Web Services (AWS) uses the power of cloud computing to provide an abstraction that developers and IT departments can leverage on what really matters, develop software, and avoid tasks like maintenance and capacity planning. [6]

AWS offers a wide range of services that include Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [7]. Their offer crosscuts all the layers and offers several alternatives to each one of them, as seen in Figure 13.

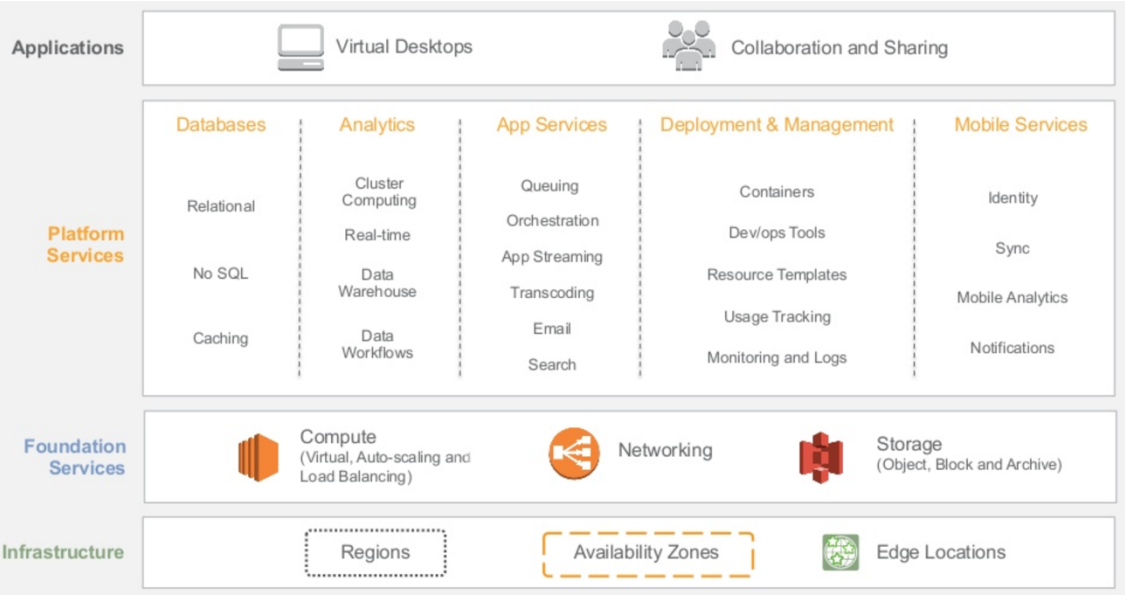


Figure 13 – AWS Cloud Computing [8]

Figure 14 helps to represent the different application layers that AWS offers and what services offer in each one of them:

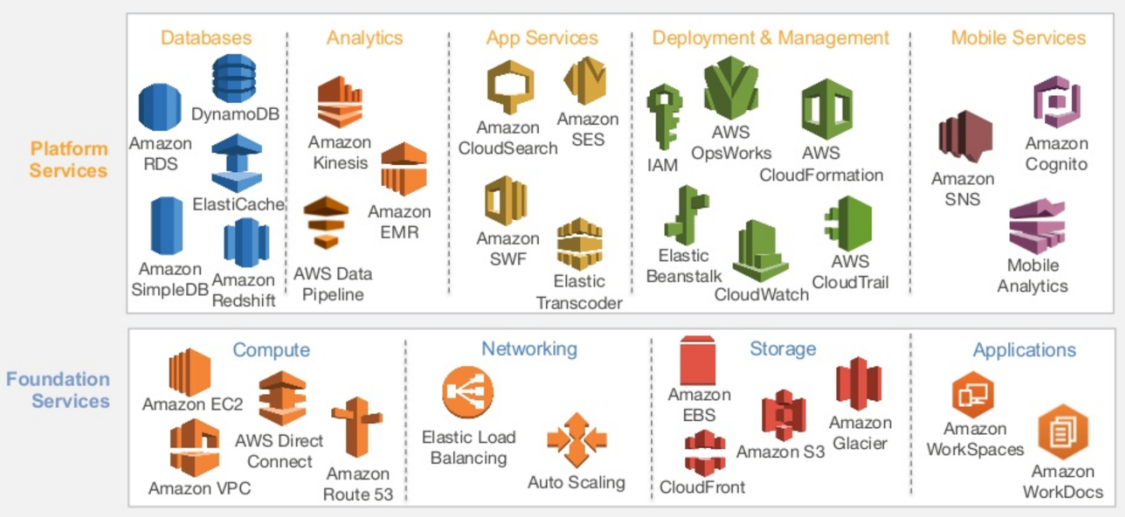


Figure 14 – AWS Services [8]

From an IaaS level, AWS offers storage services (Amazon S3 and EBS), compute services (Amazon EC2 and VPC), database services (Amazon RDS and Simple DB) amongst others. In this scope, Amazon provides “access to networking features, computers (virtual or on dedicated hardware), and data storage space” [6].

Moving now to the PaaS level, it removes the need to handle the underlying infrastructure by providing services that manage operations like “resource procurement, capacity planning, software maintenance, patching, or any of the other undifferentiated heavy lifting involved in running applications” [6]. Good examples of such services are the ones found in the Analytics and Deployment & Management groups in Figure 14 – AWS Services .

Last, AWS also offers SaaS. Such services are complete products that abstract the user of the two previous layers (Infrastructure and Platform) and only required that one knows what to do with them. Amazon WorkSpaces and WorkDocs are two examples.

Another important aspect to consider and that has a particular impact on this project, is the organisation of the website where the applications are served and the look and feel between them.

The initial dashboard lists all the available services and has a header with that in the left side allows to explore the services and join them into groups and in the right side has some account management, location and support options.

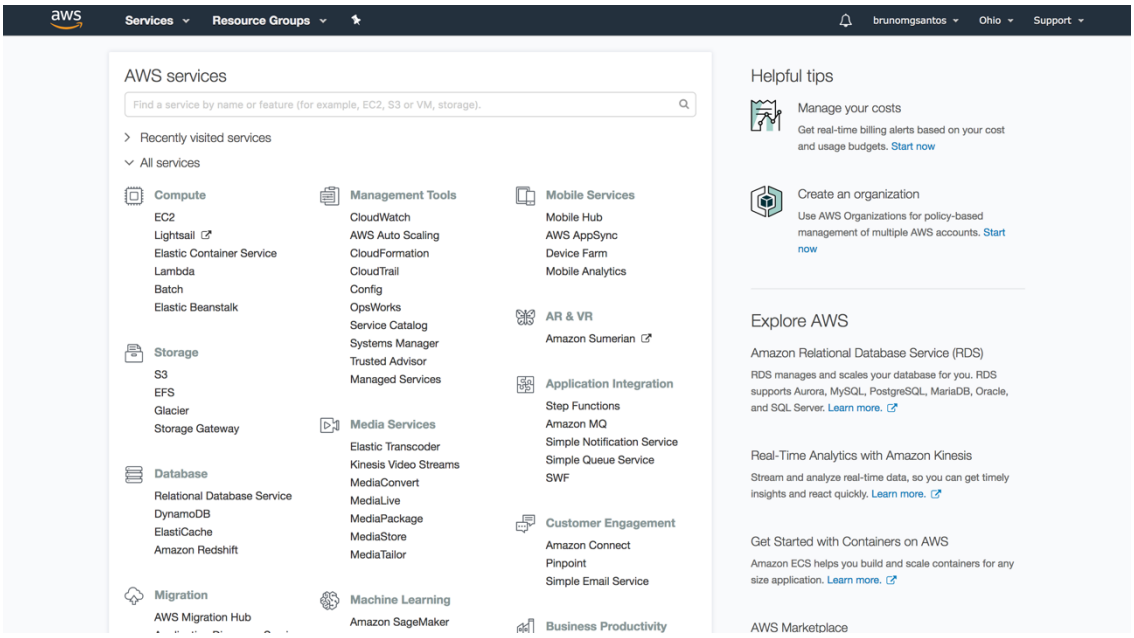


Figure 15 – Initial AWS Dashboard

When navigating to another service a full page reload happens to load the new application. The Figures Figure 16 – EC2 Dashboard, Figure 17 – Amazon ES3 Dashboard and Figure 18 – Redshift Dashboard show different applications. After them, a brief analysis is going to be made about the look and feel of all the AWS environment.

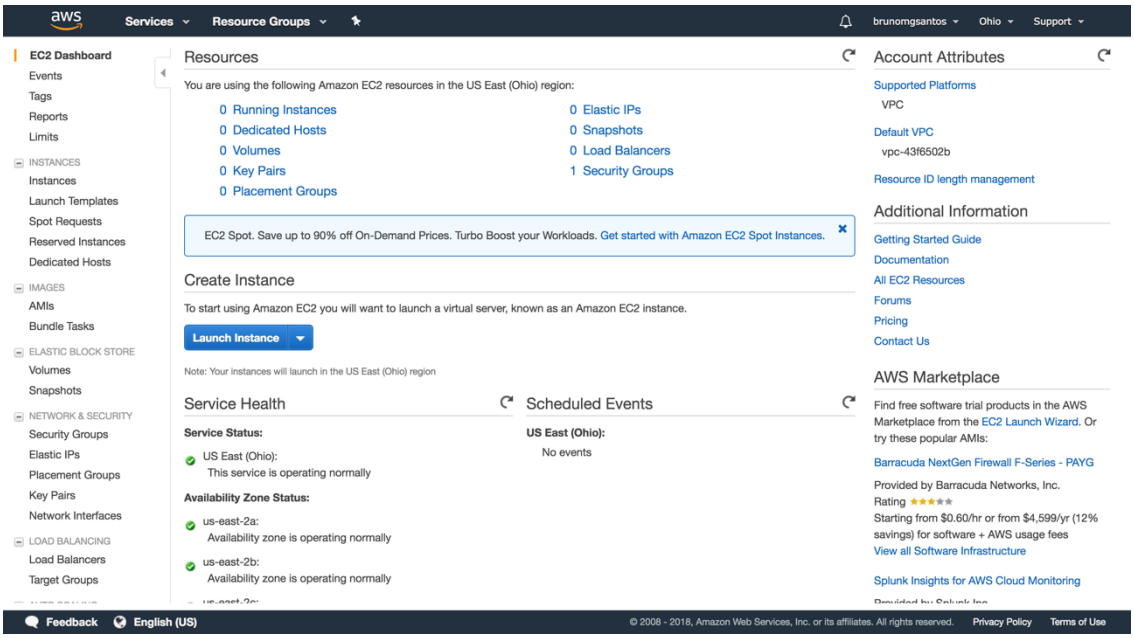


Figure 16 – EC2 Dashboard

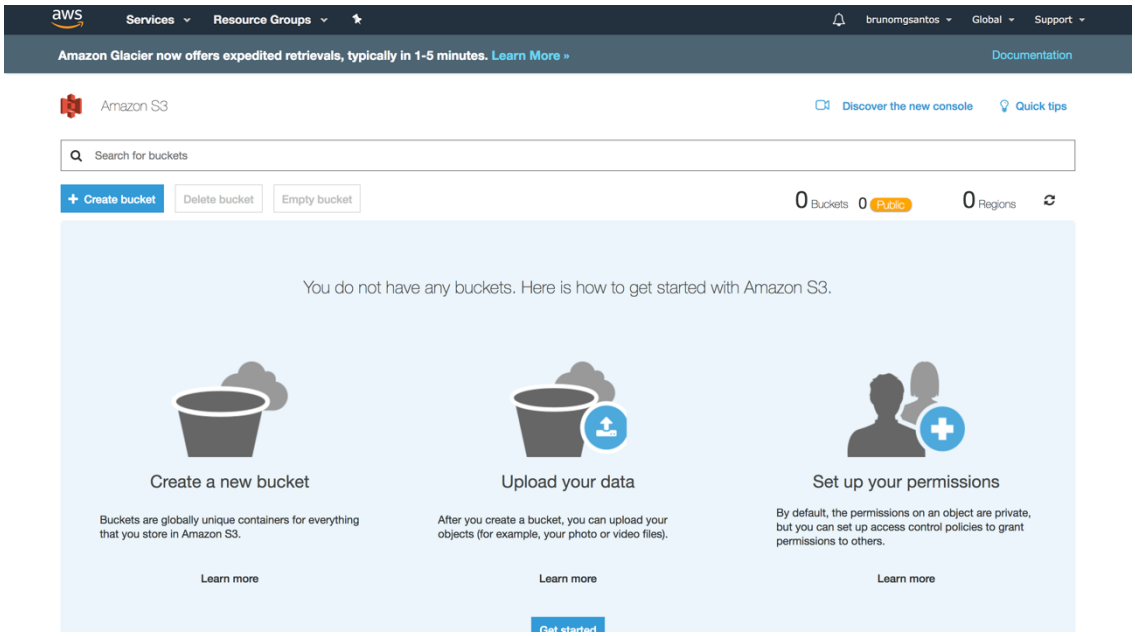


Figure 17 – Amazon ES3 Dashboard

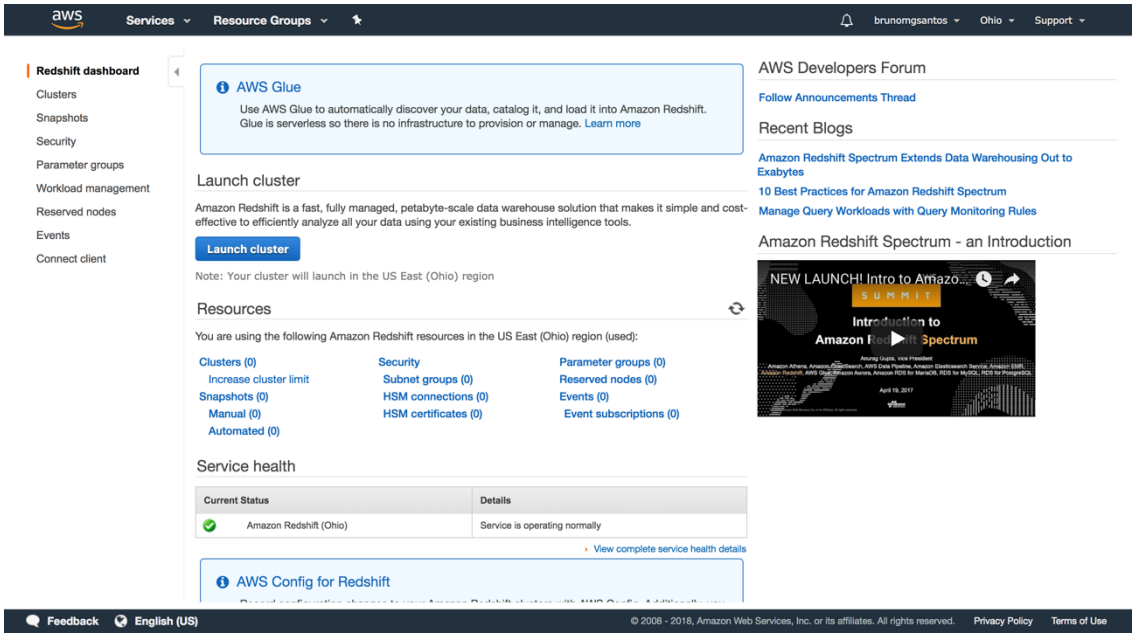


Figure 18 – Redshift Dashboard

After studying the journey between these applications, it is clear that the look and feel are consistent which makes the context switching almost null. By reducing the context switching and providing users with such a smooth journey, they increase productivity because their clients can be focused on the tasks they must complete and know what to expect from the applications.

Getting into a more detailed overview, it is possible to verify that the dashboard header is exactly the same between the applications. The colour scheme, font, pages organisation and input components (such as the buttons) do not change much or do not change at all which leads to a perception that is all one big application.

In the end, FCN and IAP are strongly based on the AWS solution, although its services have only internal purposes and are not as extensive as the Amazon ones.

3.1.2 Google Cloud Platform

Google Cloud Platform (GCP) frees the users "from the overhead of managing infrastructure, provisioning servers and configuring networks. To let innovators, innovate and let coders, well, just code" [9]. As the AWS, the main goal of GCP is to provide developers with tools that allow them to focus on developing software by simplifying infrastructural and platform management tasks.

As the AWS, it offers a wide range of services that include IaaS, PaaS and SaaS. Their offer crosscuts all the layers and offers several alternatives to each one of them, as seen in Figure 19.

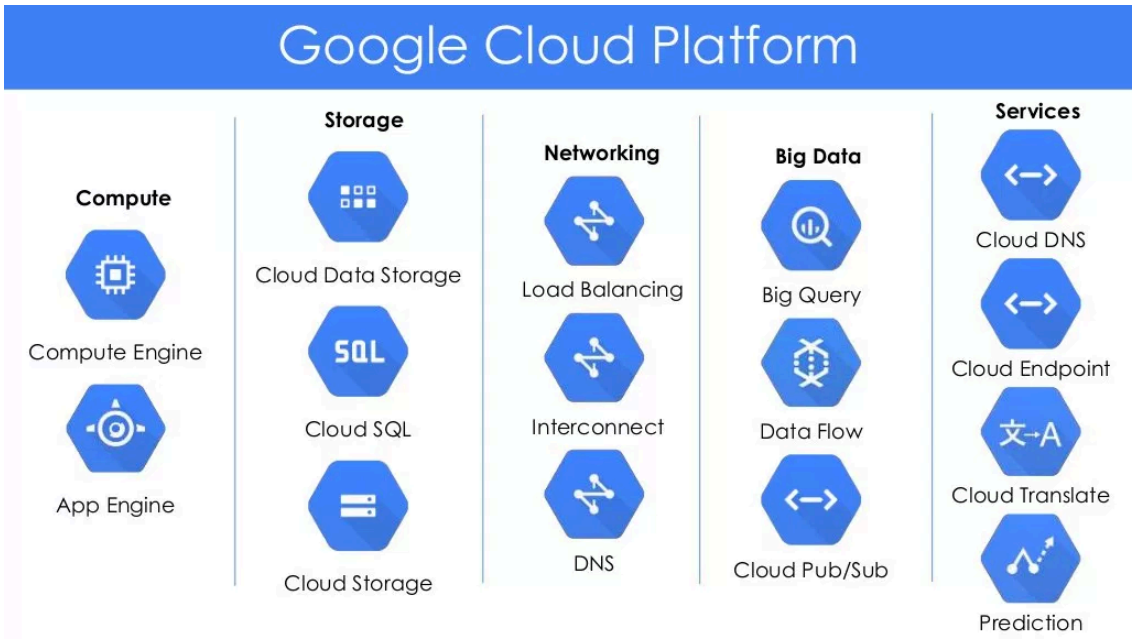


Figure 19 – Google Cloud Platform Services [10]

From an IaaS level, GCP offers storage services (Cloud Data Storage and Cloud SQL), compute services (Compute Engine), networking (Load Balancing), amongst others.

Moving now to the PaaS level, it is possible to leverage on services such as the App Engine. Although App Engine is part of the Compute offer, it is a PaaS for building web applications and mobile backends.

Last, GCP also offers SaaS. Such services are complete products that abstract the user of the two previous layers (Infrastructure and Platform) and only required that one knows what to do with them. Management Services such as Stackdriver (monitoring, logging and diagnostics) or Trace (performance and bottleneck analysis) are good examples.

To understand how an application could leverage on such services, Figure 20 represents a web application built on Google App Engine and its interactions with the GCP services.

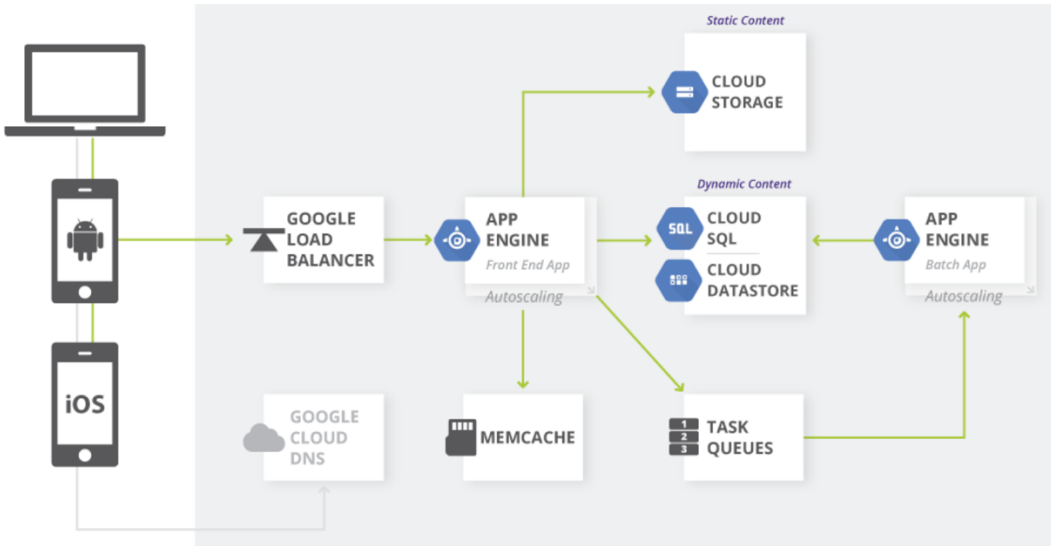


Figure 20 – Web Application on Google App Engine Architecture [11]

Developers can build a Web Application ready for desktop and mobile by using the Google App Engine, that simplifies the development and deployment process. The App Engine comes with integrated features such as in-memory cache, task queues, different data storages and it uses the Google Load Balancer, which abstracts another task from the developer [11].

Moving now to the organisation of the hosting website and the look and feel between applications, the same approach used the AWS study is used.

The initial dashboard contains a left menu that lists some of the available services as well as options such as the billing, how to get started and to direct the user to a page where all the services and APIs can be found. It has a header that allows to expand and collapse the left menu, search and account management options. The centre of the page contains an overview of the usage and state of the projects and services used.

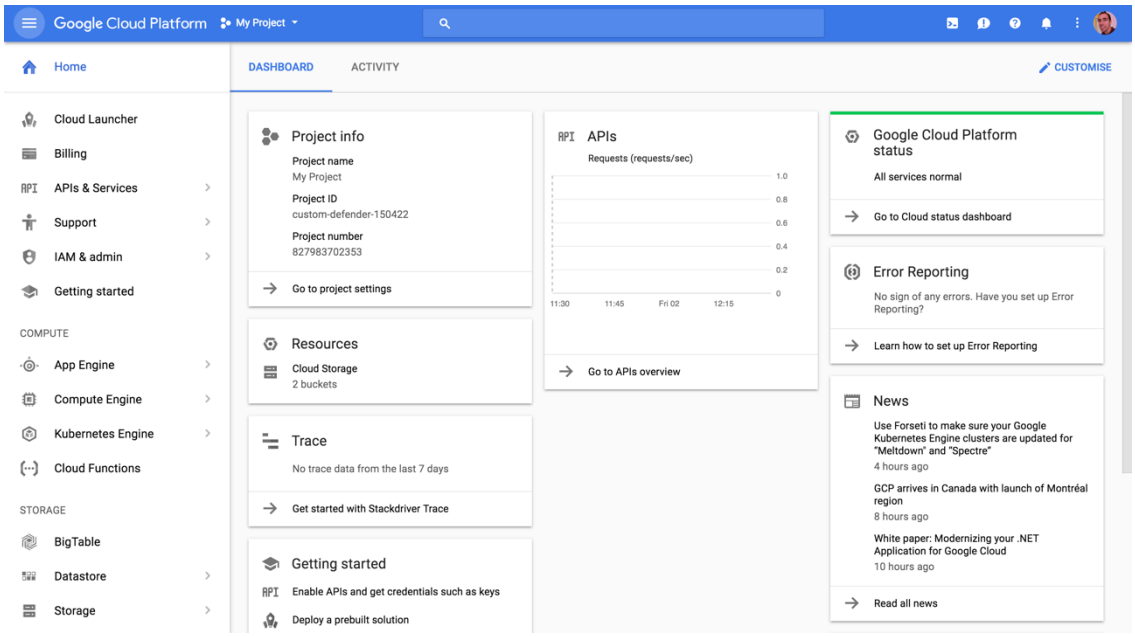


Figure 21 – GCP Initial Dashboard

When navigating to another service there is a partial reload of the page. The header is not reloaded and the left menu collapses. The new application or service is loaded into the centre component of the screen, which leads to an extremely smooth journey since the user never leaves the page.

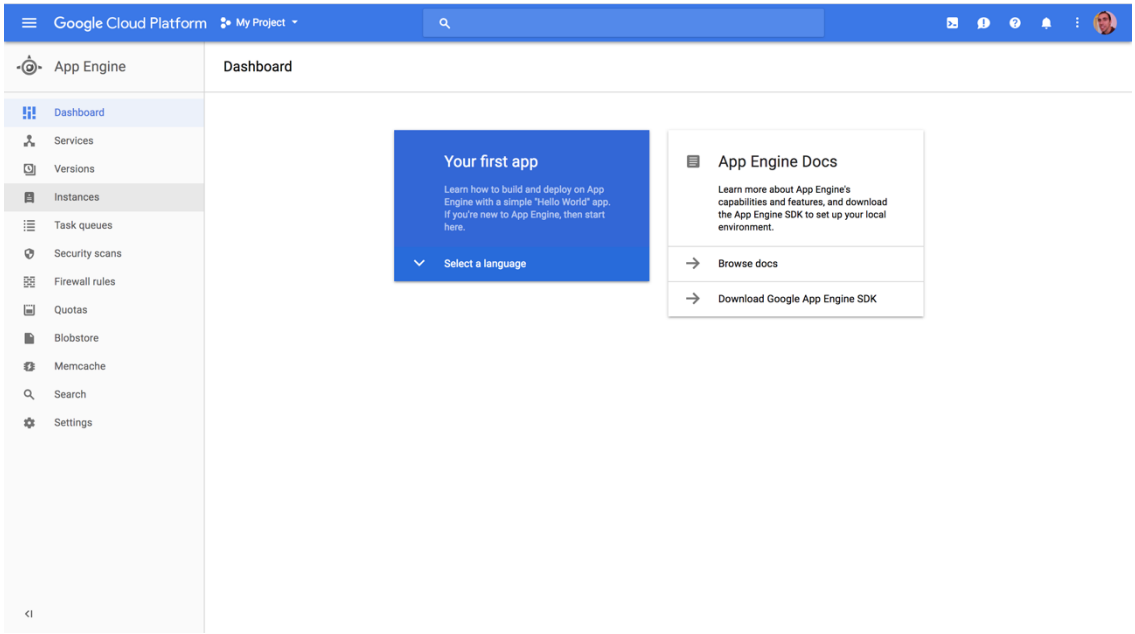


Figure 22 – App Engine Dashboard

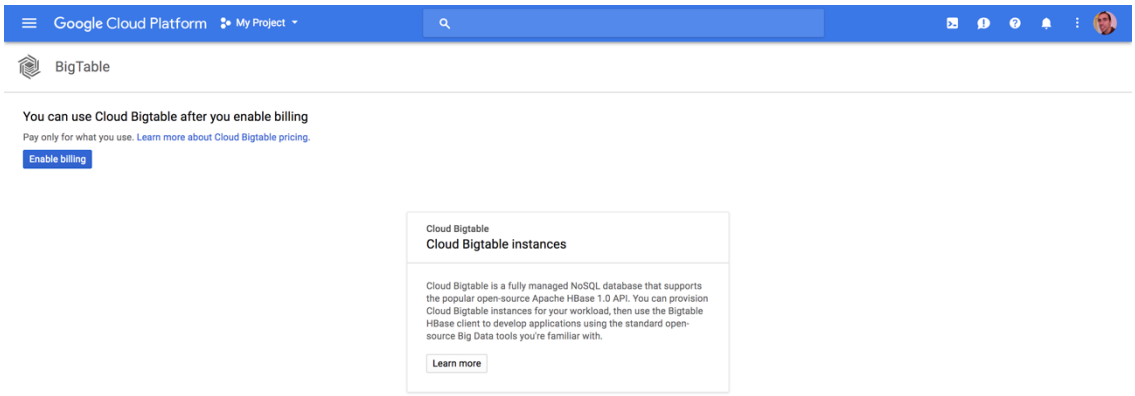


Figure 23 – BigTable Dashboard

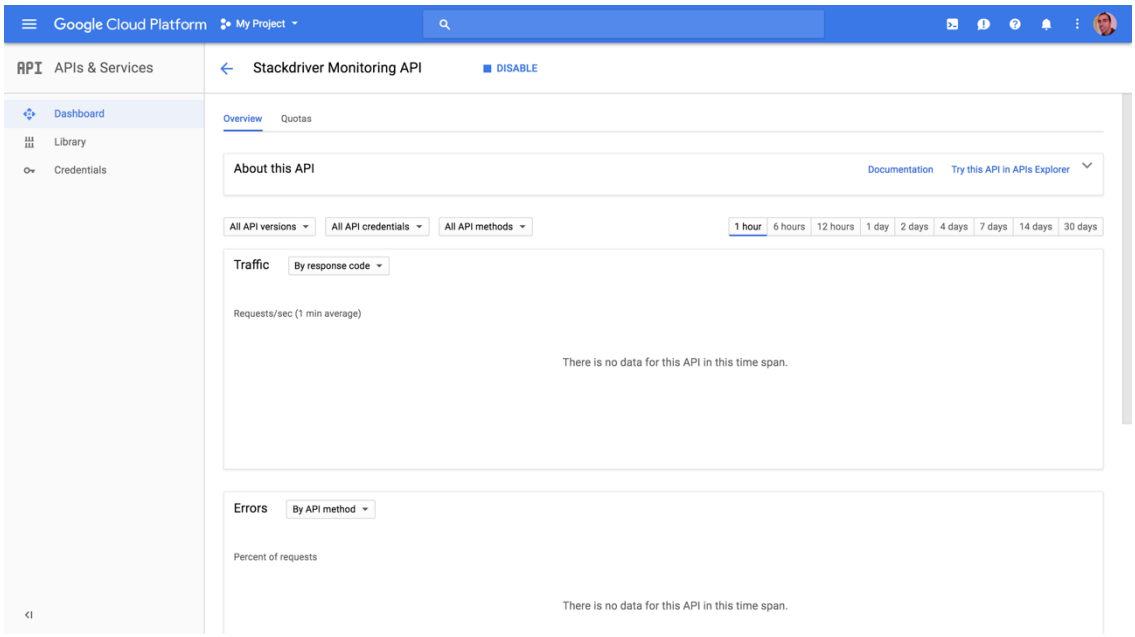


Figure 24 – Stackdriver Monitoring API Dashboard

After analysing the journey between these services, it is possible to affirm that the look and feel of the application greatly improve the user experience, since there is no full-page refresh during the navigation. Not only it reduces the context switching but it also improves the user perceived performance of the applications.

Getting into a more detailed overview, it is possible to verify that the dashboard header and left menu are exactly the same between the applications. The colour scheme, font, pages organisation and input components do not change at all which leads to an increased user experience.

3.1.3 Microsoft Azure

Microsoft Azure "is a comprehensive set of cloud services that developers and IT professionals use to build, deploy and manage applications" through their global network of data centres. Like the two previous solutions, it offers a wide range of services that include IaaS, PaaS and SaaS that crosscuts all the layers and offers several alternatives to each one of them, as seen in Figure 25.

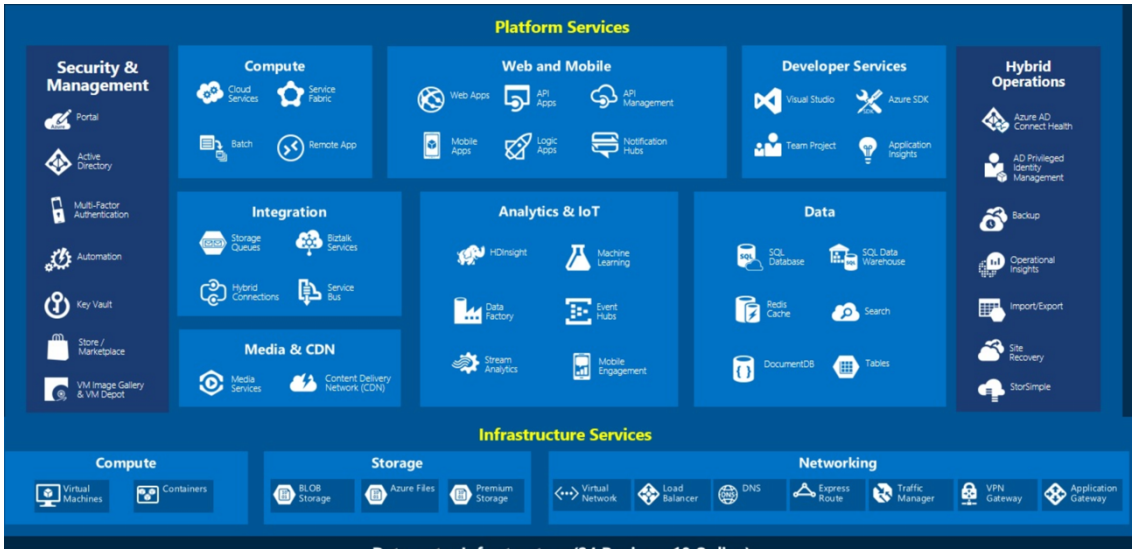


Figure 25 – Microsoft Azure Services [12]

From an IaaS level, Azure offers storage services (BLOB Storage and Azure Files), networking (Virtual Network and Load Balancer), amongst others.

Moving now to the PaaS level, it is possible to use services such as Cloud Services. Cloud Services abstracts the operating system details, letting developers configure what operating system to use and how many virtual machines instances are used to run the application.

Last, Azure also offers SaaS. Management Services such as Store and Marketplace (plugins and add-ons) or the Key Vault (key storage) and Developers Services such as the Visual Studio IDE.

To understand how an application could leverage on such services, Figure 26 represents the architecture of a social mobile and web application with authentication.

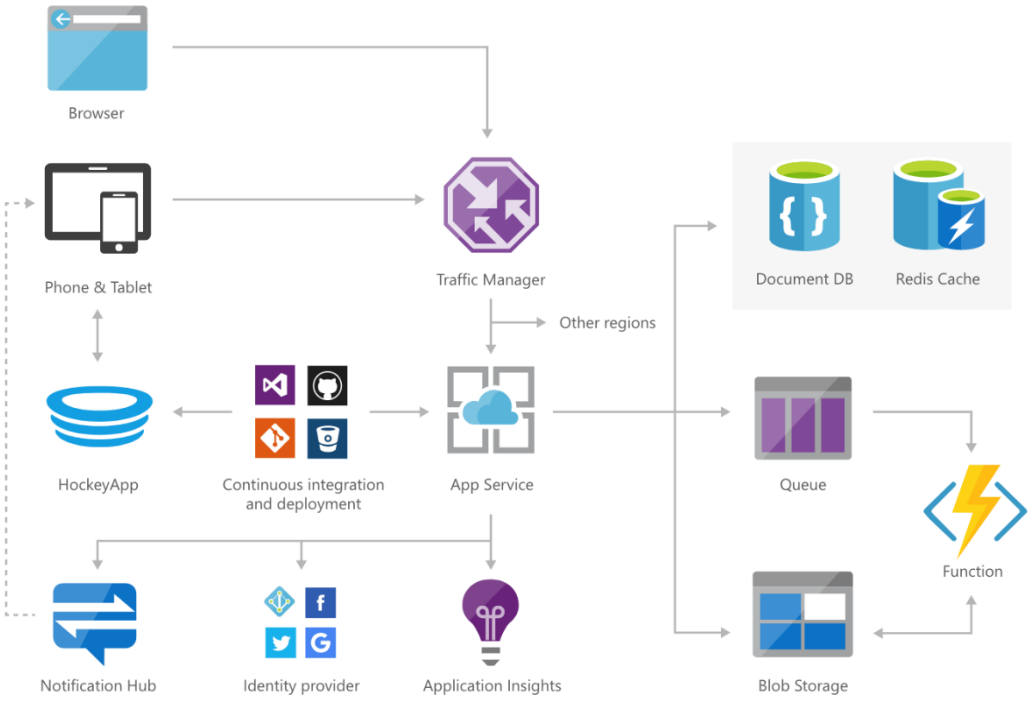


Figure 26 – Architecture for a social mobile and web app with authentication [13]

App Service works as the GCP App Engine and through its integrations with continuous integration and deployment services, it deploys the application into Hockey App that distributes the application into mobile devices. It leverages on Traffic Manager to handle the load over the application. Regarding data storages and operations between them, it uses Azure Cosmos DB, Redis Cache and Functions. Finally, it uses Application Insights to monitor the application and Notification Hub to send a push notification to the mobile devices [13].

Moving now to the organisation of the hosting website and the look and feel between applications, the same approach used in the two previous solutions.

The initial dashboard contains a left menu that lists some of the available services but, unfortunately, there are only icons and no name or description, which makes it harder to find the wanted service when ones are not yet familiarized. It has a header that allows to expand and collapse the left menu (showing the services names), search and account management options. The centre of the page contains a customisable overview of the usage of the resources and tutorials amongst others.

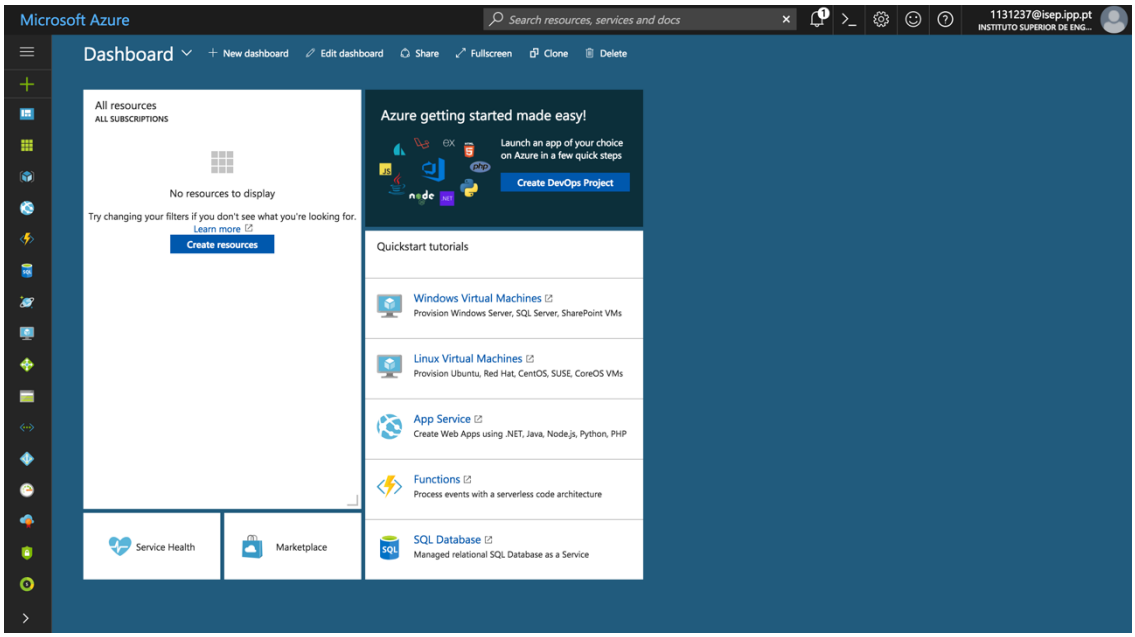


Figure 27 – Azure Initial Dashboard

The navigation between the different services follows that exact same behaviour as the one experienced in the Google Cloud Platform.

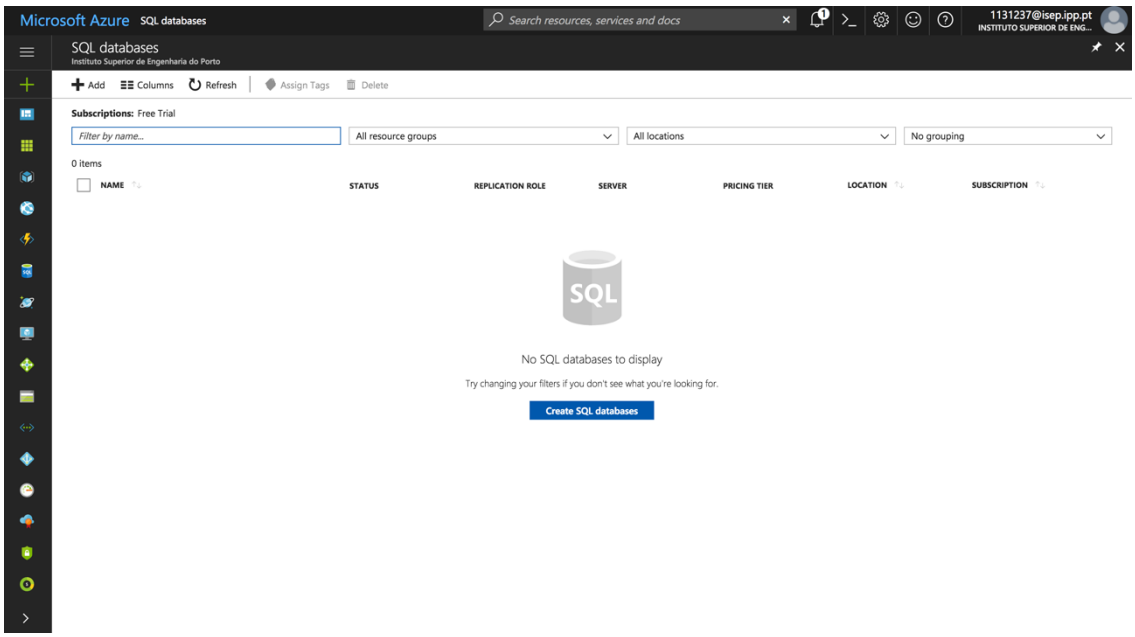


Figure 28 – Azure SQL Databases Dashboard

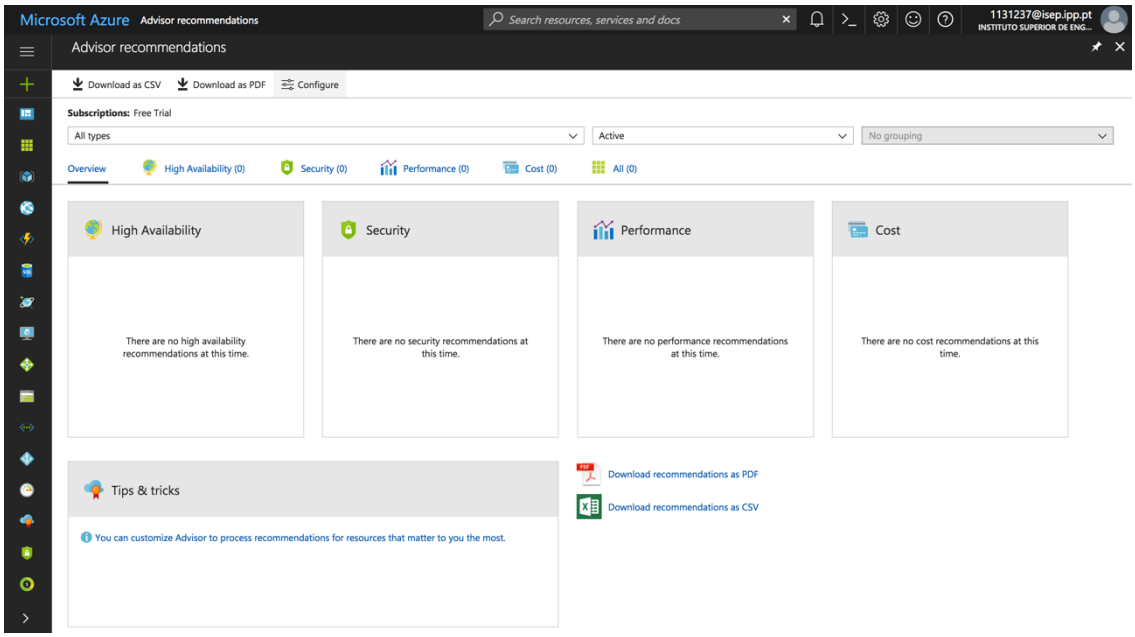


Figure 29 – Azure Advisor Dashboard

Regarding the user experience and look and feel between the different services, there is no difference from the GCP. There is also no full-page refresh during the navigation and the new application or service is loaded into the centre component of the screen. The header and left menu are the same between the navigation and colour scheme, font, page organisation and different user interface components are the same between applications/services.

3.2 Client Side Frameworks

Nowadays, there are over twenty client-side frameworks [14]. In order to choose the best frameworks that could fit this project purposes, there is the need to understand which of these frameworks are worth studying. For that, it was taken into account popularity, support, growing and team/company's comfort. The chosen ones were React, Angular and Vue, from a list that also included Backbone and Ember [15].

Regarding popularity, these three frameworks are the more popular at the time of this writing, alongside Backbone and Ember. React and Angular are the clear winners with Vue, Ember and Backbone being close to each other [15].

Looking at the support and who is behind the frameworks, Angular is developed by Google and React is developed by Facebook. Backbone, Ember and Vue don't have a technological giant supporting. However, Vue is developed by a former Google Engineer that, previously, worked in the Angular team [15].

Analysing the growth of each one of them now, Backbone is being in a great decline in the last five years. Angular growth has stagnated. Ember growth declined in the year 2016 but last year started growing back up. Both React and Vue are the fastest growing, with Vue a little ahead of React [15].

Finally, looking at the team and the company's comfort with these technologies, Angular and React are already used within the company. That means that the boilerplate of developing applications with these technologies would be lower since experience, knowledge and support for such applications already exist inside the company.

After this analysis, both Angular and React were clear choices of applications worth studying, However, Vue's fastest growth is something to consider and, for that reason, is also being studied.

3.2.1 React

React is "a JavaScript library for building user interfaces" in a declarative and component-based way. By designing simple views for each state of an application, React is capable of "efficiently update and render just the right components" when data changes. When building components, React encourages the creation of reusable ones that are able to manage their own state. Since the "component logic is written in JavaScript instead of templates", one can easily pass rich data through the application and keep the state out of the DOM [16].

Since React only handles the user interfaces of an application, it can be considered the View in an MVC (Model – View - Controller) architecture.

One of the innovative features of react is the DOM abstraction created, that offers a "simpler programming model and better performance". React it is also capable of performing server-side rendering and use native features of the mobile devices with React Native. React Native is a framework that allows developers to build native mobile applications using React [17].

React implements a one-way data flow and can be complemented with Flux [18]. One-way data flow implementations mean that the model is the single source of truth of an application. For instances, when a user interacts with an application, messages are triggered to signal the model. The model is the one responsible to access and change the application's state. Data always flow in a single direction, making it easier to reason about the application. As oppose, two-way data binding dynamically bounds the UI to the mode data, meaning that when the UI changes, the model is updated and when the model changes, the UI is updated. Flux is a pattern for managing data flow and assuring that data flows in one single direction [19].

Since React only handles user interfaces, the architecture of a React application using the Flux pattern would have the architecture portrayed in Figure 30. It is important to notice that an informal notation is used. An explanation about the same can be found after the Figure.

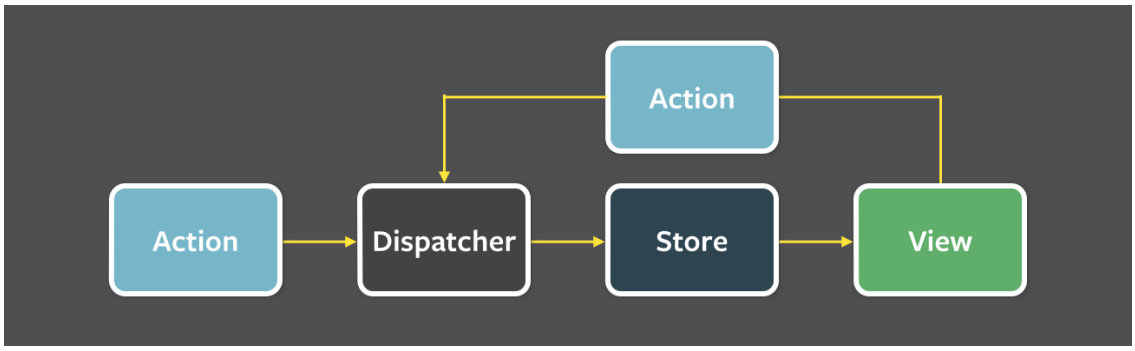


Figure 30 – Structure and Data Flow of a React application using Flux [20]

The architecture present in Figure 30 shows four distinct elements: (i) actions, that capture the several ways that an application can be interacted with; (ii) a dispatcher, responsible for receiving the actions and dispatching them to the stores that have that dispatcher registered; (iii) stores, that contain the application data and register the dispatchers to receive the actions; (iv) and views, that corresponds to React , that are responsible for handling user interactions and dispatch the according actions [19].

According to [18], the advantages and limitations of this framework are:

Advantages

- Uses virtual DOM which is a JavaScript object. This improves the application's performance since JavaScript virtual DOM is faster than the regular DOM.
- Can be used on client and server side as well as with other frameworks.
- Component and data patterns improve readability, which helps to maintain larger apps.
- Reusable Components.

Limitations

- Covers only the view layer of the app, hence there is still the need to choose other technologies to get a complete tooling set for development.
- Uses inline templating and JSX [18], which mixes templating and scripts in the same file, which can be harder for a developer to get used to.

3.2.2 Angular

Angular is a client-side framework for constructing client applications in HTML and either JavaScript or TypeScript, a language that transpiles to JavaScript [21]. Since the project is currently using the Angular 1.5, the state of the art analysis is focused on the versions that

came afterwards. Angular 2 was the next big release from the Angular team and, since then Angular 4 and 5 were released, but only added improvements to the Angular 2 version.

The applications are built by combining HTML templates with “Angularized markup”, creating component classes to handle those templates, creating services that are responsible for the application logic and aggregate those components and services in modules. Figure 31 represents the Angular architecture using an informal language. An explanation of the same can be found after the Figure.

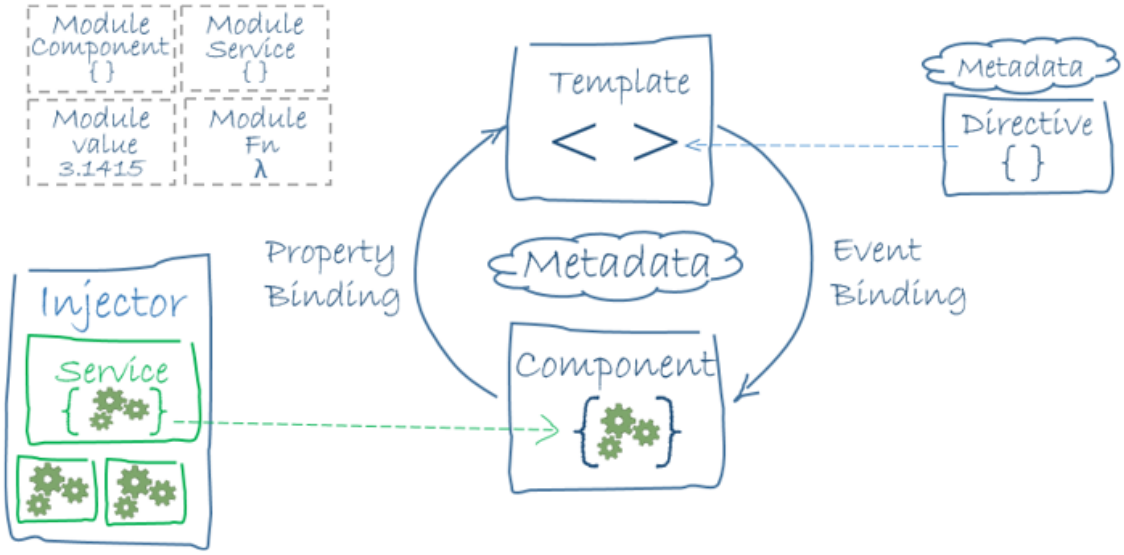


Figure 31 – Angular Architecture [21]

After analysing Figure 31 and according to Angular Documentation [21], it is possible to identify eight main building blocks:

- Modules – Angular apps are modular and Angular has its own modularity system called NgModules.
- Components – Controls a patch of a screen called view;
- Templates – Form of HTML that lets angular know how to render the component;
- Metadata – Tells Angular how to process a class;
- Data Binding – Mechanism for coordinating parts of a template with parts of a component;
- Directives - Angular templates are dynamic. When Angular renders them, it transforms the DOM according to the instructions given by directives.
- Services - Service is a broad category encompassing any value, function, or feature that your application needs.
- Dependency Injection - Dependency injection is a way to supply a new instance of a class with the fully-formed dependencies it requires;

Advantages

- Two-way data binding.
- Directives.
- Dependency Injection.
- Component-Based Architecture.
- Testability.

Limitations

- Performance
- Steep Learning Curve
- Divided and churning community.
- Verbose and Complex

3.2.3 Vue

Vue is a progressive framework, meaning that can be adopted incrementally, for building user interfaces. Vue has a core library that is focused only on the view layer and can be enhanced with several tools and libraries according to the developers' needs [22].

Vue assumes that its syntax is like the Angular one while it also shares many functionalities with React [23]. Having that in mind, this state of the art analysis is only focused on the unique Vue features.

One of Vue's most unique features is the reactive system. When a new object is added to the state view iterates over it and converts every property into getters and setters. From this moment on, Vue's reactive system keeps track of any change in the application state and will automatically re-render the DOM when such happens [24] [25]. Figure 32 helps to visualize how the reactive system works using an informal notation to describe it.

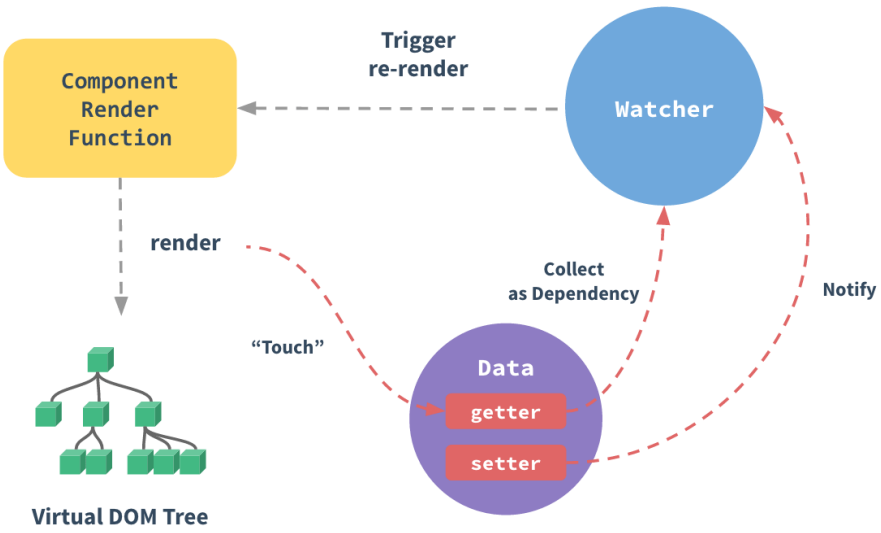


Figure 32 – Vue Reactivity Workflow [24]

Vue also offers data-binding, directives and components.

Due to the similarities between the two previous frameworks, the advantages and limitations are focused on particular aspects.

Advantages

- Small Size
- Ease of understanding and development
- Simple integration
- Comprehensive Documentation

Limitations

- Small developer community (compared to the previous ones)
- The creator is a Chinese American that defends the Chinese development community leading to a vast number of plugins that are documented only in Chinese.

3.2.4 Comparison

Now that the basis of each framework was presented, it is time to compare them. For that, it is adopted the metrics used by Jens Neuhaus in [26].

(i) Market Acceptance and Support

One important metric is to understand the market acceptance and support of each framework. The best source of information is the GitHub page of each application.

In the previous six months, it is clear that React is the market leader by a fair margin. Vue and Angular share closely the same number of downloads.

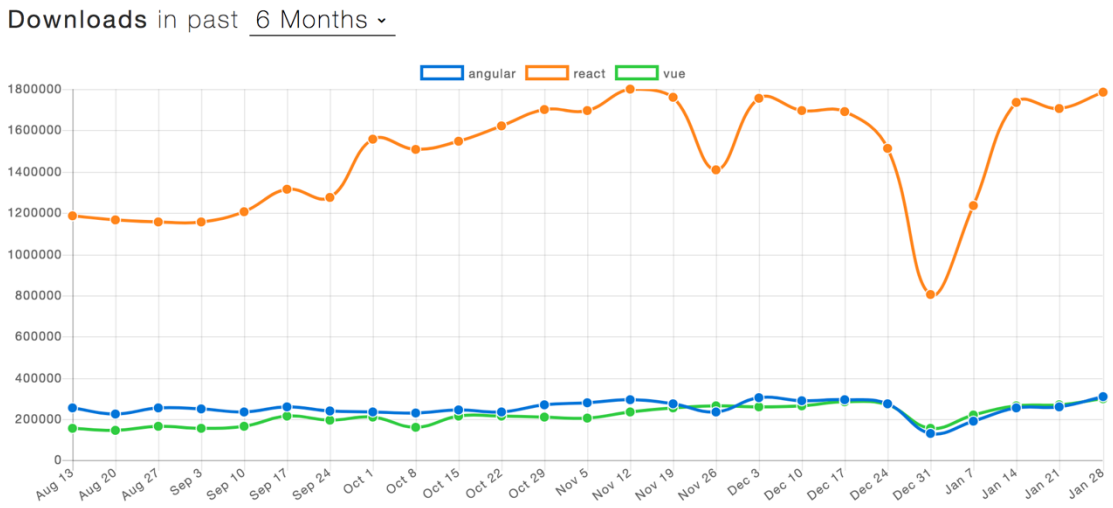


Figure 33 – Framework downloads in the previous six months [27]

Although trends should not and cannot be a decisive factor when choosing a framework, it can help to understand the reach of each one of them and to understand how vast the community is. A vast community can mean a bigger support for the developers.

(ii) Components

The three frameworks are component-based. However, due to their simplicity, Vue and React are particularly good for building dumb components: "small, stateless functions that receive an input and return elements as output" [26].

(iii) Templates

Angular templates consist of HTML enhanced with Angular specific directives, which forces the developer to learn Angular-Syntax.

React uses JSX, an optional pre-processor for HTML-like syntax which will be compiled in JavaScript later. JSX is a big advantage for development, because everything is in one place, and code completion and compile-time checks work better. When one makes a typo in JSX, React does not compile, and it prints out the line number where the typo is present [26].

Vue joins everything in a single file. Each file represents a component and has templates, scripts and styles in ordered sections. Vue has also libraries that allows the usage of JSX.

(iv) State Management and Data Binding

These three frameworks handle data differently. React uses the Flux pattern to handle the data flow. One popular implementation of the Flux pattern is the Redux library. Redux is governed by three principles: one application should have only one source of truth, the state of an application should be read-only and changes should be made with pure functions. Rephrasing, "the status of the complete application is stored in an object tree within a single store" [26].

Vue has another implementation of the Flux pattern, the Vuex library. It is like Redux, however, it mutates the state as opposed to the immutable state in Redux.

Although Angular does not have one particular implementation of Flow or other state management library, it can use Redux for that purpose. Other libraries with that purpose are being built, but none has yet proved to be an alternative.

Regarding data binding, both Angular and Vue offer one-way and two-way data binding. React only offers one-way data binding.

Two-way data binding changes the model state when a UI element is updated as opposite to one-way binding, where the model is updated first and only then the UI elements are rendered.

(v) Learning Curve

Although Angular has an extensive documentation, it has a really steep learning curve. Angular has a big eco-system and a lot of features and tools built in so, there is a lot to learn before one can work efficiently with it.

React difficulties come with the decisions that need to be made regarding the third-party libraries. React applications are built with a lot of third-party libraries that compose the React core library. The developer must have the time and knowledge to find the library the best fits his interests.

On the other hand, Vue is considered the easier of the three to learn. "Companies switch to Vue because it seems to be much easier for junior developers" [26].

(vi) Size and Performance

Regarding the size of each one of these frameworks, the gzipped file size of Angular has about 143kb (kilobyte), the React one has about 43kb and the Vue one has about 43kb.

Regarding performance, there are several performance benchmarks available. The one presented here was run on a MacBook Pro 15 (2,5 GHz i7, 16 GB RAM, OSX 10.12.5, Chrome 58.0.3029.110 (64-bit)). The results show the duration, in milliseconds of several operations

plus the respective standard deviation. Both React and Vue use virtual DOM which is supposed to improve performance so it is expected that they show better results.

Name	angular-v4.1.2-keyed	react-v15.5.4-redux-v3.6.0	vue-v2.3.3-keyed
create rows Duration for creating 1000 rows after the page loaded.	193.1 ± 7.9 (1.2)	212.2 ± 14.2 (1.3)	166.7 ± 8.6 (1.0)
replace all rows Duration for updating all 1000 rows of the table (with 5 warmup iterations).	197.4 ± 5.3 (1.2)	206.7 ± 7.3 (1.2)	168.5 ± 5.0 (1.0)
partial update Time to update the text of every 10th row (with 5 warmup iterations).	13.0 ± 4.5 (1.0)	18.0 ± 1.6 (1.1)	17.3 ± 2.9 (1.1)
select row Duration to highlight a row in response to a click on the row. (with 5 warmup iterations).	3.4 ± 2.3 (1.0)	8.7 ± 2.9 (1.0)	9.3 ± 1.7 (1.0)
swap rows Time to swap 2 rows on a 1K table. (with 5 warmup iterations).	13.4 ± 1.0 (1.0)	17.1 ± 1.3 (1.1)	18.3 ± 1.5 (1.1)
remove row Duration to remove a row. (with 5 warmup iterations).	46.1 ± 3.2 (1.0)	52.4 ± 1.7 (1.1)	52.6 ± 2.7 (1.1)
create many rows Duration to create 10,000 rows	1946.0 ± 41.8 (1.2)	1931.7 ± 35.6 (1.2)	1587.5 ± 33.9 (1.0)
append rows to large table Duration for adding 1000 rows on a table of 10,000 rows.	324.6 ± 10.1 (1.0)	366.4 ± 10.9 (1.1)	399.5 ± 11.0 (1.2)
clear rows Duration to clear the table filled with 10.000 rows.	379.9 ± 11.3 (1.5)	410.9 ± 9.8 (1.6)	254.5 ± 5.0 (1.0)
startup time Time for loading, parsing and starting up	84.3 ± 2.6 (1.5)	93.8 ± 6.9 (1.7)	56.6 ± 2.5 (1.0)

Figure 34 – Duration of several operations in the different frameworks [28]

It is possible to see that Vue is the one with the better performance of the three. However, all the three are close to each other.

Another crucial aspect is the memory allocation of each framework. Figure 35, shows the memory allocation in MegaBytes and the respective standard deviation of two operations:

Name	angular-v4.1.2-keyed	react-v15.5.4-redux-v3.6.0	vue-v2.3.3-keyed
ready memory Memory usage after page load.	4.8 ± 0.0 (1.3)	4.9 ± 0.1 (1.3)	3.8 ± 0.0 (1.0)
run memory Memory usage after adding 1000 rows.	10.9 ± 0.1 (1.4)	10.8 ± 0.1 (1.4)	7.5 ± 0.1 (1.0)

Figure 35 – Memory Allocation of several operations in the different frameworks [28]

Once again, Vue is the clear winner. Angular and React are close to one another.

Overall, Vue is the framework with lower memory allocation, with faster performance and with the lower file size. However, the values from Angular and React show close results with Vue, meaning that size and performance, alone, cannot be a decisive factor.

(vii) Conclusion

Taking into consideration every aspect used to compare the different frameworks, Vue is the framework that best fits the needs of this project. However, due to the popularity of React, its community, and the fact that a technological giant is behind it, React is also a good choice to meet these project goals.

3.3 Web Component Libraries

In order to provide teams with agnostic components that could be used with any framework, one of the solutions is to build web components.

According to [29], web components can be defined as being a set of APIs available in the browser to create reusable, custom encapsulated HTML tags that developers can use in their web pages and web apps. Components built with Web Component standards will work across browsers who already support those standards and can work with any JavaScript framework or library. They are built based on the current web standards and have four main specifications:

- Custom Elements – used to build “fully-featured DOM elements” [30];
- Shadow DOM – used to encapsulate style and markup in web components;
- HTML Imports – used to include and reuse HTML documents in other HTML documents;
- HTML Template – used to “declare fragments of markup that go unused at page load, but can be instantiated later on at runtime” [29];

The code present in the Code 5 sample represents the creation of a custom element, without detailing the element content. Inside the component, Shadow DOM can be used to encapsulate markup and style.

```
class AppDrawer extends HTMLElement {...}
window.customElements.define('app-drawer', AppDrawer);
```

Code 5 – Creation of custom element [31].

One alternative to create elements without custom elements is to create HTML templates and importing them using HTML imports.

After creating an element (through custom elements or HTML templates) importing the script or the HTML template, the created element can be used as any other HTML element, as seen in the Sample Code 6.

```
<app-drawer></app-drawer>
```

Code 6 – Usage of a custom element [31].

One problem with web components and its specifications is the support across the browsers. Since it is a relatively new technology the support is still very limited. Figure 36, Figure 37, Figure 38 and Figure 39 show the support for each specification across different browsers. Each column represents a different browser and number inside a cell represents its version. When a cell colour is red, it means that that specific version of the browser does not support such technology. Light green means it's partially supported and dark green means it's fully supported by that particular version of that browser. The centre row represents the current browser version, while the ones above represent the previous versions and the rows bellow represent the versions that are being developed and in the beta phase.

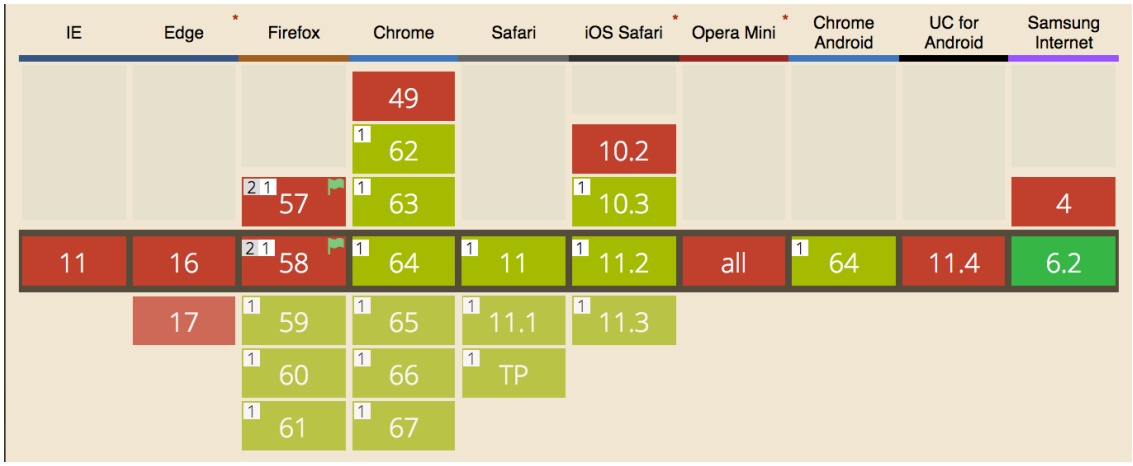


Figure 36 – Custom Elements Support [32]

Regarding the support of custom elements, there is still a long road to be covered. It is currently fully supported only by Samsung Internet 6.2 and partially supported by Firefox, Chrome, Safari, iOS Safari and Chrome Android.

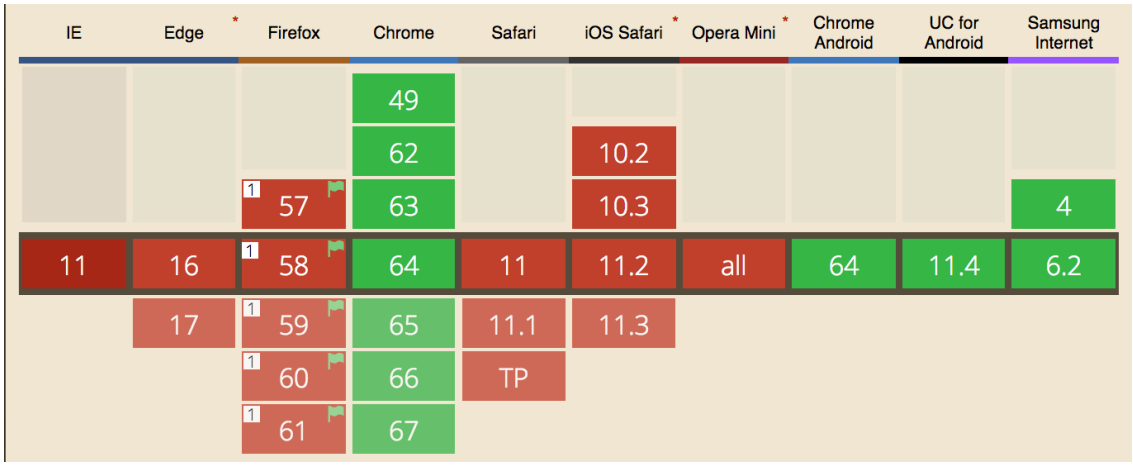


Figure 37 – HTML Imports Support [33]

HTML Imports are supported by few browsers, however, those that do, fully support it. Amongst them are Chrome, Chrome Android UC for Android and Samsung Internet.

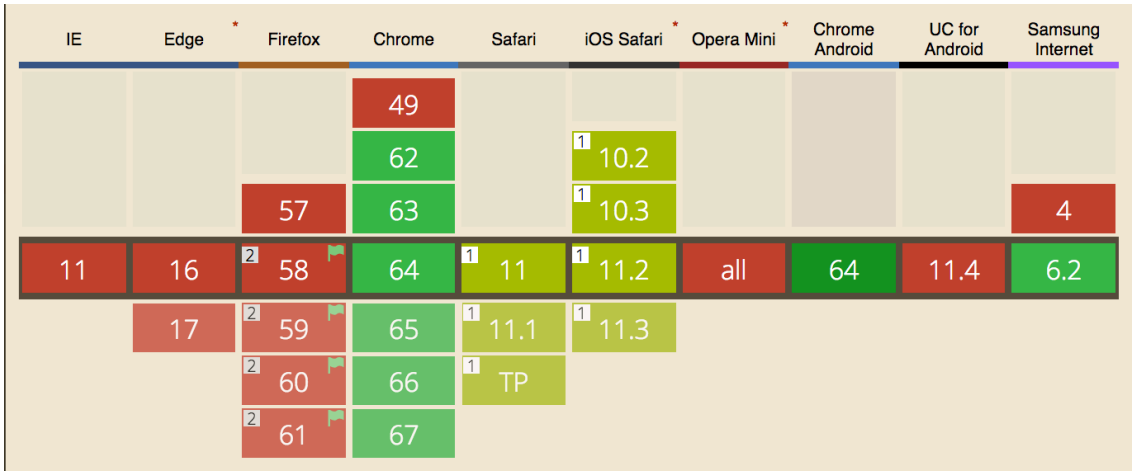


Figure 38 – Shadow DOM Support [34]

Shadow DOM is Fully supported only by Chrome, Chrome Android and version 6.2 of Samsung Internet. Nonetheless, Safari and iOS Safari partially support it.

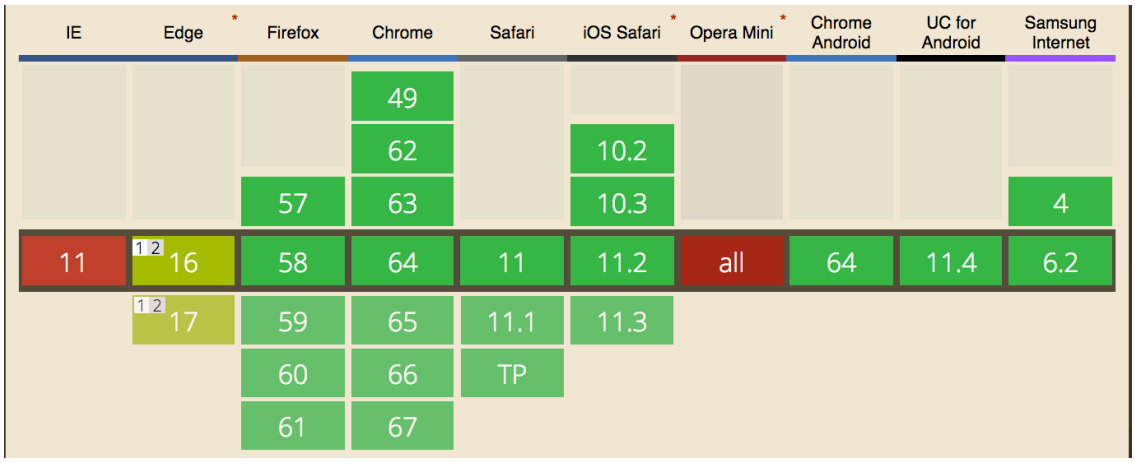


Figure 39 – HTML Templates Support [35]

HTML Templates is the specification that has wider support. It is fully supported by Firefox, Chrome, Safari, iOS Safari, Chrome Android, UC for Android and Samsung Internet. Edge has partial support.

Only Chrome and Samsung Internet fully support the four specifications (except for the custom elements that, at the moment, are only partially supported.)

Since FCN and IAP only obligation is to work in the Chrome most recent versions, support does not seem to be a problem.

Even though Web Components can be built without the help of any tool or library, that would increase the effort and difficulty to do so. However, there are already some technologies that are a great help in creating web components.

Now that a brief introduction to what web components are, it is easier to understand and analyse the tools that can help developers build them.

Since Web Components is a recent technology and specifications such as Shadow DOM and HTML Imports are still working drafts by the W3C (World Wide Web Consortium) [36], there is not a vast offer of libraries or other technologies to help developers build Web Components. The technologies found were (i) Bosonic; (ii) Polymer; (iii) Skate; (iv) X-Tag; (v) Slim; (vi) and Stencil [37].

The most important deciding factor to choose what libraries should be studied is the capacity of easily integrating the developed web components into other frameworks, in other words, create cross-framework components. From the previous six enumerated libraries and the time of this writing, only two of them provide such capability: (ii) Skate [38]; and (vi) Stencil [39]. However, the next version of (ii) Polymer will also increment such capabilities into the library and, because of that, should be taken into account [40].

3.3.1 Polymer

Polymer [41] is a JavaScript library that assists developers to create custom reusable HTML elements by providing a set of features on over plain web components. Such features encompass both one-way and two-way data binding, computed properties, conditional and repeat templates and gesture events.

As stated previously, one problem with web components it's the browser support for it. Polymer allows developers to use all the web components features by providing a set of polyfills, a piece of code (or plugin) that provides the technology the developer expects the browser to provide natively [42]. This improves the range of browsers that can run the developed web components, meaning a better market reach for the applications that are built using Polymer. Polymer also offers one-way and two-way binding to connect" data from a custom element to a property or attribute of an element in its local DOM" [43].

The Polymer latest version (2.0) appears to have some problems and difficulties to be used with other frameworks [44]. That represents a problem since the goal of producing agnostic components in the IAP framework is to ease the developers' job and not constraining their decisions. However, there are some libraries, such as angular-polymer [45] or react-polymer [46] that provide support for Polymer elements in Angular and React respectively.

The version 3.0 of Polymer is planned to be launched at the end of this year first quarter and it brings features that allow an easy integration of the generated web components with other frameworks: "JavaScript Module bundling. Polymer Bundler will produce either HTML imports based bundles or JS modules based bundles" [40]. By creating HTML imports or JS modules base bundles, the web components generated can be imported to other projects with different technologies (Angular, React, Vue, etc) without a direct dependency of the Polymer library itself, reducing the boilerplate of the integration.

3.3.2 Skate

Skate [38] is a library that helps developers to build components that are compatible with other frameworks. Not only it provides "clean property/attribute semantics" as it "adds several lifecycle callbacks for responding to prop updates, rendering and updating, as well as a way to manage internal component state".

Skate has the ability to use plugins that hook into renderers of other frameworks/view libraries, such as React, Preact, amongst others. This allows catching certain lifecycle events unique to each framework architecture.

Skate only uses two of the web component standards, Custom Elements and Shadow DOM, and it allows developers to not use Shadow DOM although they recommend against doing so.

3.3.3 Stencil

Stencil [39] is a compiler that produces Web Components. It combines the best concepts and features of some of the most popular client-side frameworks, such as Angular, React and Vue, into a build-time tool [47].

Stencil features some concepts that were already approached in this state of the art analysis:

- Virtual DOM
- Async Rendering
- Reactive data-binding
- TypeScript
- JSX
- Server-Side Rendering
- Pre-Rendering
- Object-as-properties

The web components generated by the Stencil's compiler can work out of the box with many popular frameworks and have the advantage of having key capabilities on top of them, such as server-side rendering, pre-rendering and object-as-properties (a technique that allows component's properties to be Objects instead of Strings) [39].

Stencil also improves the simplicity of writing web components by providing a new set of APIs like JSX, Virtual DOM and Async Rendering.

All these features are added to the web components development without ever breaking the compatibility with the web components standards. However, it does not use HTML Imports [48].

In the end, the Stencil's compiler creates plain JavaScript, without any dependencies and it even includes polyfills for the browsers that are missing specific features [48].

3.3.4 Comparison

The three libraries provide a lot of tools to help developers build standardized web components. However, there are some differentiators factors between them: browser support, the development environment (how difficult it is for a developer to work with) and integration of the generated web components with client-side frameworks and other libraries.

(i) Browser Support

Regarding the browser support, there is no doubt that both Polymer and Stencil are the ones with the wider range. Due to the integrated polyfills that they both offer, it is easier for the

developers to build cross-browser applications without having the concern of building polyfills by hand.

Skate depends on the browsers support for each one of the four standards of the web components. Such bottleneck can be overcome with hand-made polyfills, but that would increase the pain of developing web components.

(ii) Development Environment

Studying the development environment of each one, the three provide good tools, APIs and documentation that developers could leverage for a quick learning process.

However, it is important to notice that Stencil is the one that offers the most features besides the web component standards. Such features can help create to improve not only the developers' productivity but it can also help to create more performant components due to the offered techniques.

(iii) Performance

Regarding performance, there were no benchmark tests found like the ones from the client-side frameworks. Nevertheless, Stencil is the only of the three libraries that offer specific techniques to improve the performance of the components (async rendering, pre-rendering and virtual DOM).

Another metric that can be used, is the bundle size of each library. All the values presented assume that the bundle has been compressed (gzipped). Polymer latest version (2.0) can go from 10kb to 49kb, depending on the number of polyfills needed [49]. Stencil's bundle has 6kb [50] and Skate has around 4kb [51].

(iv) Integration

Regarding the integration, both Stencil and Skate work with other frameworks out of the box, making the integration process a simple task.

However, the current version of Polymer seems to present some challenges and difficulties to developers, making integration one major problem [44]. There are some libraries that allow the usage of Polymer elements in some client-side frameworks, but that adds up another dependency, increasing complexity and maintainability.

(v) Conclusion

After comparing Skate, Polymer and Stencil with the previous metrics, Stencil is the one the best fits the project needs. Despite the fact that the three in their essence provide an easier development experience when developing web components, Stencil's wide range of features, seen in sub-section 3.3.3, and easy integration with other frameworks give more flexibility and freedom. However, Polymer 3.0 (version yet to be released), has in its roadmap the resolution

of the integration problems stated in sub-section 3.3.1, which can also put Polymer in a position where it fits the project needs.

4 Business Analysis

A business analysis is a vital study made to understand the current state of the business and with that identify needs and possible solutions. In this chapter, the core business concepts of the FCN and IAP are enumerated as well as the processes and actors that take part in it. The chapter ends with a brief consideration of the business restrictions.

4.1 Stakeholders

There are three major actors in the business: (i) the framework maintainers, (ii) the developers that use the framework to build the applications and (iii) the users of these applications. Besides these actors, there are a couple more of stakeholders that are also part of the decision making process: (i) software architects; (ii) product owners; (iii) and the higher levels of the management layer.

The software architects contribute to the solution and have the final call in approving or rejecting the design and architecture of the proposed solution. They are stakeholders because IAP can contribute and be part of their solutions for other projects.

The product owners are the ones that define the requirements. They are the ones that decide what features and functionalities must IAP provide, not only for the users of IAP but also for the teams that built their applications upon IAP.

The management layer is the one who decides if the project can be put in practice and, in the end, gives the go or no-go to the project financing.

Several processes are used to orchestrate the dynamics between these actors. On the developers and maintainers teams, continuous integration and continuous delivery processes are used to ensure a fast time to market and the quality of the delivered software. These teams also use the SCRUM agile methodology. One crucial aspect is that the sprints of the framework maintainers and the ones from the developers who use it must be synchronized. For instance, if there is a need for a new feature in a framework component raised by the

applications development teams, the sprints must be synchronized to assure that there are no dependency conflicts. How hard it may be to achieve, IAP is supposed to make the development process of the teams more agile. By adding a management bottleneck, such is undermined.

4.2 Requirements

In this section, the requirements for this project are raised. To do so, this section is divided into two sub-sections. The first one describes the functionalities and behaviours that the system should offer (functional requirements) while the second one focuses on the qualitative and operational aspects of the system (non-functional requirements). The raised requirements are a junction between the requirements for the FCN and IAP versions.

4.2.1 Functional Requirements

This subsection is dedicated to understanding the functionalities and behaviours that the IAP provides to its users. These requirements are focused on the features that the platform and website where the applications are aggregated presents. IAP does not offer many features. What IAP offers are a series of components and services that applications can leverage on to provide diverse features to its users. The features are present in Figure 40.

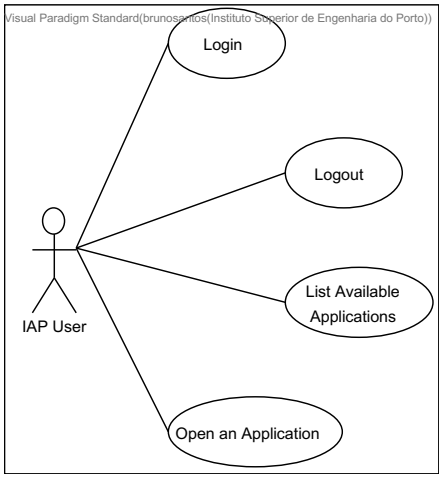


Figure 40 – IAP Use Cases (UCs)

Table 1 exposes the use cases raised in Figure 40 with a short explanation, using a user story like notation: “As a <type of user>, I want <some goal> so that <some reason>”.

Table 1 – IAP Use Cases Descriptions

UC Code	Name	Short Explanation
UC001	Login	As an IAP User, I want to login into the platform so that I have access to the applications.
UC002	Logout	As an IAP User, I want to log out of the platform so that I can end my session.
UC003	List Available Applications	As an IAP User, I want to be able to see a list of all the available applications so that I know what applications can I have access to.
UC004	Open an Application	As an IAP User, I want to be able to open an application so that I can use the features that it provides.

Since the IAP focus is in providing components and features for applications to be developed on, what it lacks in functional requirements (that are mainly on the applications side) it has in non-functional requirements, as seen in section 4.2.2.

4.2.2 Non-Functional Requirements

To document the non-functional requirements, it was used the FURPS+, which is a technique used to validate and classify the non-functional requirements after understanding the client needs. This section is more dedicated to the developers of the IAP and the applications built upon the framework.

4.2.2.1 Functionality

Functionality covers aspects such as the feature, capabilities and security of the system. These requirements are of particular importance for developers and many of the goals require them to be completed.

- NFR1 – Data Validation, both on client and server side.
- NFR2 – Accounts must be locked after five unsuccessful login attempts (when the username is correct but password incorrect) for critical and regulated systems and ten login attempts otherwise.
- NFR3 – Transmitted and stored data must follow the Paddy Power Betfair's Cryptography Policy.
- NFR4 – All active sessions must be terminated for intentionally locked accounts.
- NFR5 – Relationships between user accounts must be identified and tracked.
- NFR6 – Verification that a unique persistent cookie value is issued to each new web user.
- NFR7 – The applications developed under the IAP must be deployed independently from each other.
- NFR8 – Only users with the right permissions should be able to access the applications. Authorisation in each application is mandatory.
- NFR9 – Logging mechanisms must be provided.

- NFR10 – Themes must be provided.
- NFR11- IAP must be framework agnostic.

4.2.2.2 Usability

Usability approaches areas such as human factors, help, and documentation.

- NFR12 – A document must exist that describes the log files format.
- NFR13 – The applications under the IAP scope must have documentation with guides on how to release and deploy applications.
- NFR14 – Architectural documentation should be provided.
- NFR15 – Documentation for each component and service provided must exist.
- NFR16 – The look and feel of the applications must be easy to handle.
- NFR17 – The look and feel must be consistent across applications. Guidelines should be provided.

4.2.2.3 Reliability

Reliability covers aspects such as the frequency of failure, recoverability, and predictability of the system.

- NFR18 – In the event of failure of a called service or component, the system must be able to recover without manual intervention.
- NFR19 – The applications must remain available if a single point fails or if non-mandatory dependencies fail.

4.2.2.4 Performance

Performance covers properties as response times, throughput, accuracy, availability and resources usage.

- NFR20 – The application must present an availability percentage of 99.9%. Such a value is a company goal for every service and application.
- NFR21 – The solution must be highly scalable. The application performance must not be affected by increased usage.
- NFR22 – It must not consume more resources than FCN. Resources contemplate the number of machines, RAM, CPU, and others.

4.2.2.5 Supportability

Supportability covers aspects regarding adaptability, maintainability, internationalization and configurability.

- NFR23 – Consistency between logging files' names across all products.
- NFR24 – Consistency between logging files' locations across all products.

- NFR25 – Log files must have a specific purpose.
- NFR26 – Transactions errors must be logged.
- NFR27 – Database exceptions must be logged.
- NFR28 – Production logs must be configured at the warning level.
- NFR29 – Logging must be configurable.
- NFR30 – Every request should generate an accompanying log entry.
- NFR31 – Monitoring should be detached from a specific piece of monitoring software.
- NFR32 – Monitoring reports should expose detailed status information.
- NFR33 – Monitoring should expose the application status as fast as possible.
- NFR34 – Alerts should be trigger when the SLA's are not met.
- NFR35 – Every component and service must have unit tests implemented.
- NFR36 – Must only support Chrome browser latest versions.

4.2.2.6 Other Restrictions (+)

The plus represents the implementation, interface, operations, packaging and legal details of the system.

- NFR37 – The cookbook (where the system configurations are defined), package and service names must be the same for consistency.
- NFR38 – The solution must leverage on the Paddy Power Betfair's infrastructure.
- NFR39 – Each application should be deployed in its own cluster.
- NFR40 – IAP Website should be a Progressive Web App [52]

5 Value Proposition

A value proposition is the promise of value that will be delivered to the customer [53].

Building a value proposition is a crucial step in the development of any project. It is within a value proposition that the reasons for such project to exist and why should the customers use it can be found.

5.1 The New Concept Development Model

According to Peter Koen, the New Concept Development (NCD) Model is a model that should be followed in the innovation process. Figure 41 shows in a visual manner the NCD Model.

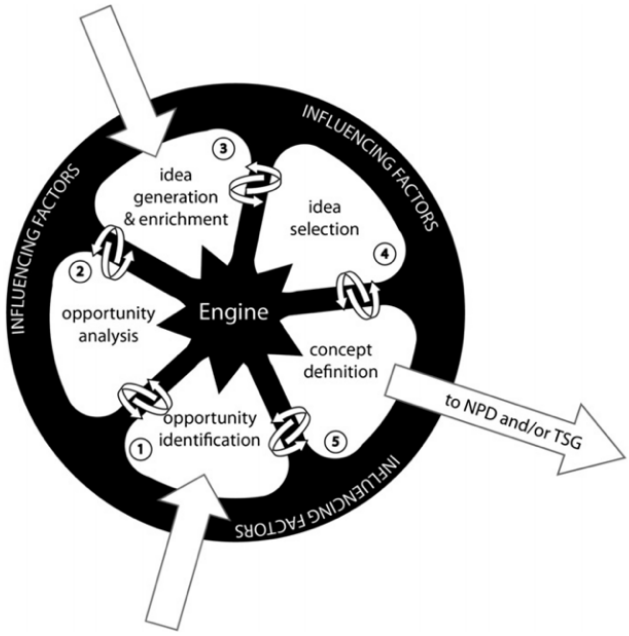


Figure 41 – The NCD Model according to Peter Koen [54]

This model is divided into three major parts: the engine, the wheel and the rim. According to the article Managing the Front End of Innovation – Part II:

"The engine consists of the core elements that provide power to the frontend process—organizational attributes, such as senior management involvement in the front end, vision, strategy, resources, culture, and teams and collaboration. The wheel, the inner part of the model, comprises five activity elements: opportunity identification, opportunity analysis, idea generation, idea selection, and concept definition. The third element of the NCD model, the rim, consists of the external environmental factors that influence the engine and the activity elements. "

After understanding how the NCD Model works, it is time to apply it to this project by identifying and explaining the five activity elements of the wheel of this model.

5.1.1 Opportunity Identification

The opportunity arose from a series of technological debts and process problems that were delaying the development process of the teams as well as making it harder for the framework maintainers to deal with problems that were being found through the lifetime of the project. For instances, the technological stack becoming outdated and the releases painful process (all these problems were stated in the sub-chapter 1.2) were two of the main reasons that led to the opportunity identification.

A market study in form of a survey was sent to the FCN maintainers, developers and users to prove that there was indeed an opportunity in these problems. Such a survey can be consulted in the Annex FCN Framework Pulse.

The sources and methods used were brainstorming and mind maps, from a creative perspective, and process mapping to help the team to solve a serious of problems.

5.1.2 Opportunity Analysis

After identifying the opportunity, an analysis of the same is needed to model it "into specific business and technology opportunities" [55].

In the scope of this project, this analysis consists in understanding what the market (technological or within the same businesses) is offering and how can we leverage on it. This is the point where a lot of effort is put into Proof of Concepts (PoC) with the goal of understanding what and how the new technologies can contribute to improving the technological stack, to ease the complex theme scheme.

The study of the AWS, GCP and Azure solutions is also a crucial point to understand how they evolved and if or how could such evolution be put into the project to improve the user experience.

5.1.3 Idea Generation and Enrichment

The idea generation and enrichment (or idea genesis) consists in "the birth, development and maturation of the opportunity into a concrete idea. This represents an evolutionary process in which ideas are built upon, torn down, combined, reshaped, modified, and upgraded" [55].

Since the opportunity of this project was raised due to a series of problems (enumerated in the sub-chapter 1.2), this process is repeated for each problem, since they have a prioritization that must be respected and, therefore, not all the work can be parallelized.

The method used in this project consists in the generation of several solutions from the FCN and IAP team members that, after documented, are shown to the team. The team, then, discusses the several ideas and tries to reach a consensus. This consensus can go through choosing one of the presented solutions to a new solution being built by fetching parts of the presented ones. At least one alternative is also accorded during this discussion.

During this process, the FCN and IAP team establishes contact with other interested parties (software architects, FCN users, developers that use the framework) to try to understand if the idea that is being thought respect their needs and opinions.

5.1.4 Idea Selection

In the idea selection phase, a meeting is scheduled where all the interested parties are present. Due to the heavy technological component of this project the software architects are, most of the time, a constant in these meetings. Here, the selected ideas that came from the idea genesis are presented and, again, discussed. In the end, it is expected that a final solution is reached and agreed between all interested parties. High-level estimations are also discussed. "In most businesses, there are so many product/process ideas that the critical activity is to choose which ideas to pursue in order to achieve the most business value" [55].

Due to the magnitude of the project, the solution also needs to be pitched to higher levels of the companies. Usually, these levels are the ones responsible for financing the project development.

5.1.5 Concept Definition

The concept definition implicates the construction of a business case following a series of parameters. These parameters must include estimations of the customer needs, market

potential, overall project risk, investment requirements, technology unknowns and competitor assessments [55].

In this step, the idea is broken into several milestones to get a more granular idea of the problems and in what order they need to be worked on. Another market analysis may be needed or the responses from the previous one can already show what are the market most urgent needs. Stakeholders and software architects may play an important role here since they have a more crosscutting knowledge of the company overall needs, and how this project is inserted into such needs.

5.2 Benefits and Sacrifices for the Clients

This section presents the definitions of value, value for the customer and perceived value. It ends with an analysis of the benefits and sacrifices that are expected to be required to the FCN and IAP clients.

5.2.1 Value, Value for the Costumer and Perceived Value Definitions

According to Susana Nicola, Eduarda Pinto Ferreira and João José Pinto Ferreira "the creation of value is key to any business, and any business activity is about exchanging some tangible and/or intangible good or service and having its value accepted and rewarded by customers or clients, either inside the enterprise or collaborative network or outside" [56].

Value is not an easy concept to understand, accomplish, model and/or conceptualize since many authors consider it to be a trade-off between the benefits and sacrifices that the clients perceive during the supplier's offer [57]. Furthermore, companies tend to struggle a lot in their definition of value and how they can measure it [56].

Value can and it is perceived differently by customers and companies. The same way customers perceive different value for the same products and services, companies have different perceptions of customers' value delivery [58].

Value for the customer is well explained by the author Tony Woodall. He describes it as being the perception that the customer has about an advantage coming from an organization (being it a product, service, or anything that the company offers). Such advantage that the customer perceives, can be the result of factors as "the reduction in sacrifice, presence of benefit, the resultant of any weighted combination of sacrifice and benefit, or an aggregation, over time, of any or all these" [59].

5.2.2 Benefices and Sacrifices

As seen in the previous sub-chapter, the reduction of the sacrifice and the increase of benefits is a decisive point to increase the value that we are delivering to the customers.

According to Lapierre, there are thirteen key drivers of customer perceived value: ten benefits and three sacrifices. These key drivers can be a product, relationship and service related. The following list enumerates these key drivers [60].

- Benefits
    - Alternative Solutions – Product Related;
    - Product Quality – Product Related;
    - Product Customization – Product Related;
    - Responsiveness – Service Related;
    - Flexibility – Service Related;
    - Reliability – Service Related;
    - Technical Competence – Service Related;
    - Supplier's Image – Relationship Related;
    - Trust – Relationship Related;
    - Supplier Solidarity with Customers – Relationship Related;
- Sacrifices
    - Price – Product and Service Related;
    - Time/Effort/Energy – Relationship Related;
    - Conflict – Relationship Related;

There is also the need to understand these benefits and sacrifices through time. For that, Woodall identifies four distinct temporal positions: Pre-Purchase, At the Point of Trade or Experience, Post Purchase, After-User experience [59].

After studying the Lapierre's value key drivers and the four distinct temporal positions identified by Woodall, it possible to build the following longitudinal value proposition.

Table 2 – Longitudinal Value Proposition

	Benefits	**Sacrifices**
Pre-Purchase	Alternative Solutions. Product Customisation. Flexibility. Responsiveness. Technical Competence. Supplier's Image.	Time, Conflict
At the Point of Trade or Experience	Product Quality. Reliability. Trust. Technical Competence.	Price
Post Purchase	Product Customisation. Flexibility. Responsiveness. Product Quality. Technical Competence.	Effort, Energy
After/User experience	Trust. Product Quality.	

At the pre-purchase time, clients try to understand the product and conversations take place between all the parties. These conversations generate some alternative solutions and drive the product/service to be customised to the client needs. The FCN and IAP team is always available for discussions and, being a technical team that works with the product for several years, technical competence is assured. Since this project is being developed inside the company and it is for the company itself, the supplier's image is assured. On the other hand, the clients spend much time discussing and understanding the product. This leads to conflicts between the different actors.

At the point of trade or experience, it is expected that the client has a quality product at hands. Since it was involved in all the process, it is expected that the major benefits are the product quality, reliability, trust and, once again and for the same reason, technical competence. However, this is the point where a monetary price needs to be paid.

The post-purchase phase reflects the results based on the customer's choices. Since the product/service, this project offers also contains all the maintenance and the continued improvements required, this phase most strong benefits are the product customisation, flexibility, responsiveness and technical competence. Effort and energy will be required to integrate the new product.

Finally, the after/user experience reflects the point where the user is working with our product with no difficulties whatsoever. It reflects the experience that the customers are having with our product. It's expected that trust, due to the product quality that they are now using, is the major benefit to the customers. No sacrifices are expected at this stage.

5.3 The Value Proposition

A value proposition "is an overall view of a company's bundle of products and services that are of value to the customer" [61]. A value proposition must be concise but must also answer several questions. By reading a value proposition, anyone, even those who do not have any knowledge of the business in which the project is inserted must understand what is the product, who are the target customers and for who is the value being provided, what is that value and why the product is unique. The following paragraph presents the value proposition of the IAP.

The Internal Applications Platform is a platform for rapid front-end development of internal applications that bundles solutions for common requisites in a medium to large company, applying the don't repeat yourself principle to a product delivering level. By the end of the day, a new application developed on top of IAP will look and feel like all the other internal applications developed inside the company, enhancing the user experience and learning curve for, especially, non-tech users, and having all of them available under a single website.

5.4 Business Model Canvas

The business model canvas is a strategic tool or framework to document a business model. By analysing a Canvas, it must be possible to understand how a business creates, delivers and captures value [62].

According to Osterwalder and Pigneur, the Business Model Canvas has nine building blocks:

- **Costumer Segments** - An organization serves one or several Customer Segments;
- **Value Propositions** - It seeks to solve customer problems and satisfy customer needs with value propositions;
- **Channels** - Value propositions are delivered to customers through communication, distribution, and sales Channels;
- **Customer Relationships** - Customer relationships are established and maintained with each Customer Segment;
- **Revenue Streams** - Revenue streams result from value propositions successfully offered to customers;
- **Key Resources** - Key resources are the assets required to offer and deliver the previously described elements;
- **Key Activities** – Activities performed to deliver the previously described elements;
- **Key Partnerships** - Some activities are outsourced and some resources are acquired outside the enterprise;
- **Cost Structure** - The business model elements result in the cost structure

Now that the definition of Business Model Canvas is clear, it is time to model the business of this project into a Canvas model.

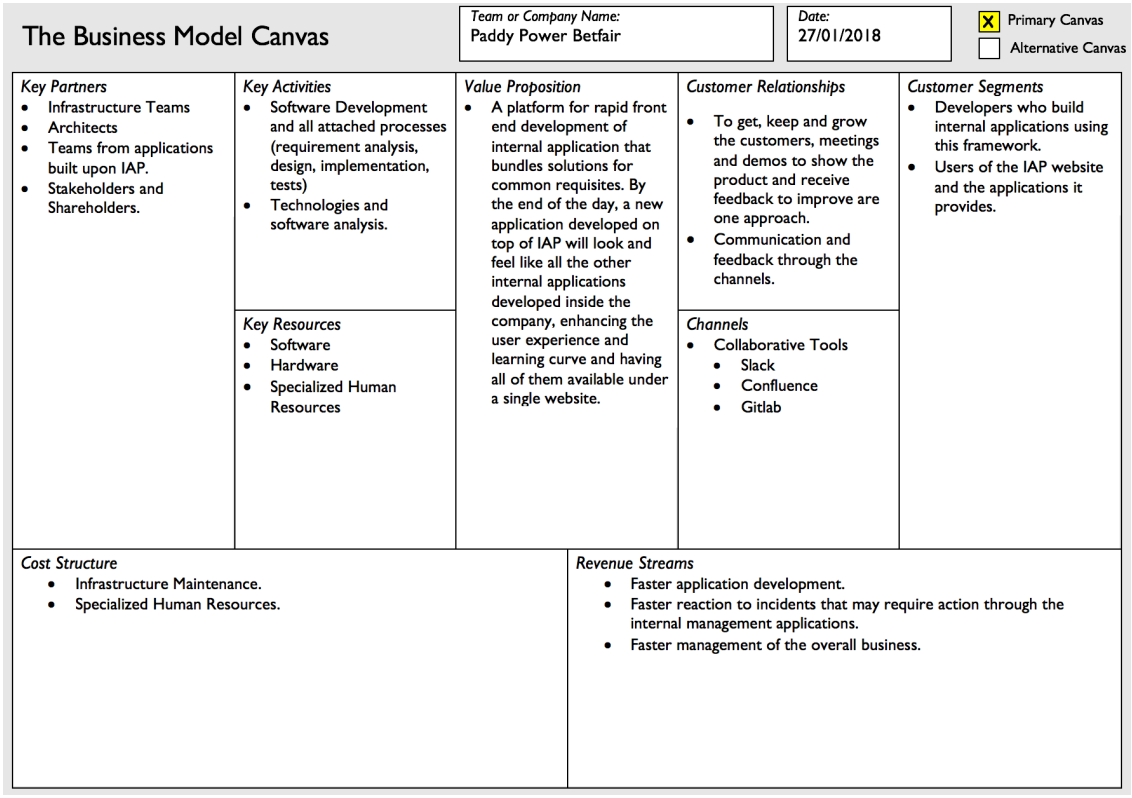


Figure 42 – Business Model Canvas

To better understand Figure 42, an analysis of each key block is needed.

Key Partners – Since this an internal project, the key partners are the ones that, inside the company, can help this project to grow and reach its goal. For that, infrastructure teams are needed to deal with the hardware, software architects to help with the architecture of the solution, the teams from applications that use the IAP to help the project grow according to their needs and, finally, the stakeholders and shareholders, since they have the last word in the approval our not of this project.

Key Activities – The analysis of the current technologies and software that can replace the current ones or take an important role in the system is a mandatory activity. Since this project is technology driven, software development processes are a must have since day one.

Key Resources – Software, hardware and specialized personnel are mandatory to achieve a technological solution.

Value Proposition – The detail needed to understand the value proposition is given in chapter 4.3.

Customer Relationship – Our customers are internal teams and users that interact with the FCN and IAP. To get, keep and grow our customer base there is the need to show to the rest of the company the product and how can they leverage their applications and their usage to

improve their own business. It is also important to collect feedback to understand if the project is going according to the customer needs.

Channels – Since the customers are internal personnel, the channels used to communicate are internal collaborative tools like Slack, Confluence and Gitlab.

Customer Segments – The customers of this project are also their actors. This project targets the developers who build applications on top of the FCN and IAP and the users of such applications.

Cost Structure – The main costs will be based on the hardware needed to deploy the solution and in the specialized human resources needed to develop it.

Revenue Streams – The revenues from this project do not come directly from it. However, the capabilities that it provides reduces costs and improves revenues, since it allows for a faster development of applications. Faster reaction to incidents that may require action through the internal management applications and faster management of the overall business is also achieved.

5.5 Value

According to Valen Allee, "People naturally network as they work so why not model itself as network" [63]. The question that such statement raises is how do people create value through such network.

Valen Allee also answers that question. "People can create value by assuming or creating roles to convert their tangible and intangible assets into deliverables that be conveyed to other roles through the execution of a transaction. In turn, value is realized by companies when they convert inputs into gains" [64].

So, a value network can be defined as being "any set of roles and interactions in which people engage in both tangible and intangible exchanges to achieve economic or social good" [64].

Another way to analyse the business value is to study the value chain. The value chain can be defined as being a set of actions that a company implements to create value for its customers. [65]. Instead of studying the departments or accounting cost types, Porter proposes a value chain analysis that focuses on all the company activities and how are they connected. The way such "activities are performed determines costs and affects profits, so this tool can help understands the sources of value of an organization" [65]. This is the tool that is used in this chapter to evaluate the value generated by this project.

5.5.1 Porter's Value Chain

Porter's value chain can be divided into two axes: primary and support activities. The primary activities encompass the actions regarding physical creation, sale, maintenance and support of services and products. On the other hand, the support activities are responsible for helping the primary ones. Each axis of activities is subdivided into more specific activities:

- Primary Activities
    - Inbound Logistics
    - Operations
    - Outbound Logistics
    - Marketing and Sales
    - Service
- Support Activities
    - Procurement (purchasing)
    - Human Resource Management
    - Technological Development
    - Infrastructure

Table 3 represents the value chain of the Paddy Power Betfair company;

Table 3 – Paddy Power Betfair Value Chain

Firm Infrastructure	Legal, Accounting, Financial, Quality Management IT Services			
Human Resources Management	Recruitment, Payroll Training, Benefits			
Technology Development	Product and process design and development. Production Engineering Market Research.			
Procurement	Subcontracting and Outsourcing. Collaborative Tools. Travel System.			
Inbound Logistics	**Operations**	**Outbound Logistics**	**Marketing and Sales**	**Service**
Real-time data about sports games and events; Customer usage and journeys;	Software development and maintenance; Content Management;	Large Betting markets offer; Complete information displayed about the events and its players;	Advertising; Promotions;	Sports Betting websites and applications; Customer Services;

Support Activities

Margin

Primary Activities

Getting into a more detailed analysis. The firm infrastructures comprise the Legal, Accounting, Finance and Quality management as well as the IT services.

The human resources management is responsible for the recruitment process, payroll, personal training and formation as well as the benefits and compensations.

The technology development is responsible for the product and process design and development as well as the production engineering. These activities encompass the software development required to create and maintain not only the betting products that the company offers to the customers, but all the internal management software (which is where this project is inserted) needed to support those products. A lot of market research is also put into effort to assure that the best technologies are being used as well into understanding the competitors technological level.

The procurement needs consists in subcontracting specialized personnel and outsourcing some activities that support the business like, for instances, Time Form, which is a company that provides data about horse racing events. The collaborative tools and the company travelling system are also acquired by the company.

The inbound logistics consist mainly of the real-time data about sports games and events as well as the data collected about our costumer's usage and journeys inside our channels.

By applying the defined operations (software development and maintenance, content management) to the inbounded logistics, the company can generate a large offer in betting markets and give the customers complete information about the players in such markets. To attract customers, effort is put into advertising and giving promotions personalised to certain events. In the end, the company services consist of sports betting websites and applications and customer services to help them in case they had any problems with the products.

5.6 AHP Method

The Analytic Hierarchy Process (AHP) is one of the main methods developed in the Discrete Multi-Criteria Decisions context created by professor Thomas L. Saaty in 1980. This method is used to divide a certain decision problem into hierarchical levels to ease its comprehension and evaluation [66].

This method is divided into seven phases:

1. Building the hierarchical decision tree;
2. Comparison between the alternatives and criteria;
3. Relative priority of each criterion;
4. Evaluate the consistency of the relative priorities;

5. Building the parity comparison matrix for each criterion considering each one of the selected alternatives;
6. Obtaining the compost priority for each alternative;
7. Selecting the alternative;

To demonstrate the use of such a method in the context of this project, an example will be put into practice. The example will be given using the independent releases problem.

The objective is to allow that the releases of the IAP and its applications can be independent of each other. The criteria's to be used are backward compatibility, changes required to the framework, changes required to the applications built upon FCN and maintenance.

Three alternative approaches were developed by the team. Technical details will be put aside for this analysis since a more detailed study will be presented in the Design Chapter.

The first alternative (Alternative 1) consists in the usage of a CDN (Content Delivery Network), a noSQL database and an API to make the communication between the noSQL database and the IAP framework. The second alternative (Alternative 2) consists in the usage of two APIs: one to expose the static assets of the applications and another one to fetch those assets into the IAP framework. The third alternative (Alternative 3) consists of the usage of manifest files and the artifactory (an artefact repository manager) where such manifests and other artefacts would be stored. These manifests would contain all the required data to reproduce the applications.

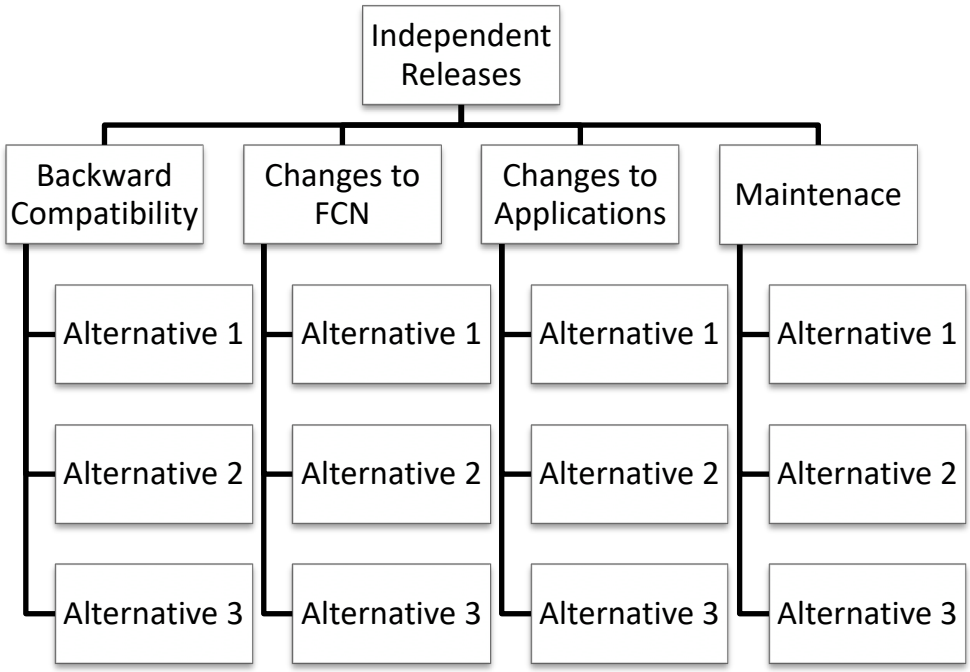


Figure 43 – Hierarchical Decision Tree Diagram

The following step consists of the comparison between the hierarchical criteria defined in the previous diagram. To do so, a scale with the different comparison importance levels is used.

Table 4 – The fundamental scale of absolute numbers [66]

Intensity of Importance	*Definition*	*Explanation*
1	Equal Importance	Two activities contribute equally to the objective
2	Weak or slight	
3	Moderate importance	Experience and judgement slightly favour one activity over another
4	Moderate plus	
5	Strong importance	Experience and judgement strongly favour one activity over another
6	Strong plus	
7	Very strong or demonstrated importance	An activity is favoured very strongly over another; its dominance demonstrated in practice
8	Very, very strong	
9	Extreme importance	The evidence favouring one activity over another is of the highest possible order of affirmation
Reciprocals of above	If activity *i* has one of the above non-zero numbers assigned to it when compared with activity *j*, then *j* has the reciprocal value when compared with *i*	A reasonable assumption
1.1–1.9	If the activities are very close	May be difficult to assign the best value but when compared with other contrasting activities the size of the small numbers would not be too noticeable, yet they can still indicate the relative importance of the activities.

To do so, a table was built to compare the different criteria using the above values.

Table 5 – Comparison of Criteria

	Backward Compatibility	**Changes to FCN**	**Changes to Applications**	**Maintenance**
Backward Compatibility	1	4	2	3
Changes to FCN	1/4	1	1/2	1/3
Changes to Applications	1/2	2	1	3
Maintenance	1/3	1/2	2	1

From Table 5, it is possible to write the following matrix:

$$\begin{bmatrix} 1 & 4 & 2 & 3 \\ 1/4 & 1 & 1/2 & 1/3 \\ 1/2 & 2 & 1 & 3 \\ 1/3 & 1/4 & 2 & 1 \end{bmatrix}$$

The next step consists of finding the relative priority of each criterion. To do so, the first stage goal is to normalize the values from the previous matrix.

$$\begin{bmatrix} 1 & 4 & 2 & 3 \\ 1/4 & 1 & 1/2 & 1/3 \\ 1/2 & 2 & 1 & 3 \\ 1/3 & 1/4 & 2 & 1 \end{bmatrix} => Normalize => \begin{bmatrix} 0.4800 & 0.5517 & 0.3636 & 0.4090 \\ 0.1200 & 0.1379 & 0.0909 & 0.0454 \\ 0.2400 & 0.2759 & 0.1818 & 0.4090 \\ 0.1600 & 0.0345 & 0.3636 & 0.1364 \end{bmatrix}$$

After obtaining the normalized matrix, it is now possible to calculate the priority vector.

Table 6 – Normalized Matrix with the Relative Priority

	Backward Compatibility	**Changes to FCN**	**Changes to Applications**	**Maintenance**	**Relative Priority**
Backward Compatibility	0.4800	0.5517	0.3636	0.4090	0.4510
Changes to FCN	0.1200	0.1379	0.0909	0.0454	0.0986
Changes to Applications	0.2400	0.2759	0.1818	0.4090	0.2767
Maintenance	0.1600	0.0345	0.3636	0.1364	0.1736

In the end, the relative priority of the different criteria is:

- Backward Compatibility – 0.45
- Changes to FCN – 0.1
- Changes to Applications – 0.28
- Maintenance – 0.17

The next step in the AHP method consists in calculating Consistency Ratio (CR) of the judgements made regarding the previous point. That encompasses the following stages:

1. The λ_{max}, that represents the biggest self-value of the normalized matrix, is obtained through the following equation:
$$Ax = \lambda_{max} x$$

2. The Consistency Index (CI) must be calculated using the next equation, where *n* is the number of lines of the matrix:
$$CI = \frac{\lambda_{max} - n}{n - 1}$$

3. The Consistency Ratio is calculated using the next equation, where RI is the random index for a big number of comparisons:
$$CR = \frac{CI}{RI}$$

The RI can be obtained through a table generated by the Oak Ridge National Laboratory from the USA.

Table 7 – RI values for square matrixes of order *n.*

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49	1.51	1.48	1.56	1.57	1.59

Following the previous steps, the Ax is calculated:

$$Ax = \begin{bmatrix} 1 & 4 & 2 & 3 \\ 1/4 & 1 & 1/2 & 1/3 \\ 1/2 & 2 & 1 & 3 \\ 1/3 & 1/4 & 2 & 1 \end{bmatrix} \begin{bmatrix} 0.45 \\ 0.1 \\ 0.28 \\ 0.17 \end{bmatrix} = \begin{bmatrix} 1.92 \\ 0.41 \\ 1.22 \\ 0.93 \end{bmatrix}$$

Now the λ_{max} can be calculated:

$$\lambda_{max} = average\{1.92/0.44, 0.41/0.1, 1.2/0.28, 0.93/0.17\} = 4.53$$

Next, the CI is calculated:

$$CI = \frac{4.53 - 4}{4 - 1} = 0.18$$

Since the CI value was found and, according to Table 7, the RI value is 0.90, the CR can be calculated:

$$CR = \frac{0.18}{0.90} = 0.2$$

A CR of 10% or less implies that the judgments made about the criteria are consistent. Although the obtained value is superior (20%) it is still a low value and, even though some adjustments may need to be made, indicate that the judgments are consistent.

Now that the consistency of the relative priority has been measured, a parity comparison matrix for each criterion must be built, having in consideration the selected alternatives.

Table 8 – Backward Compatibility Criteria

	Alternative 1	**Alternative 2**	**Alternative 3**
Alternative 1	1	4	1
Alternative 2	1/4	1	1/4
Alternative 3	1	4	1

The previous results in the following priority value for the backward compatibility criteria:

$$\begin{bmatrix} 0.44 \\ 0.12 \\ 0.44 \end{bmatrix}$$

Table 9 – FCN Changes Criteria

	Alternative 1	**Alternative 2**	**Alternative 3**
Alternative 1	1	1/2	1/3
Alternative 2	2	1	1/2
Alternative 3	3	2	1

Table 9 results in the following priority value matrix for the FCN changes criteria:

$$\begin{bmatrix} 0.16 \\ 0.30 \\ 0.54 \end{bmatrix}$$

Table 10 – Changes to Applications Criteria

	Alternative 1	**Alternative 2**	**Alternative 3**
Alternative 1	1	5	3
Alternative 2	1/5	1	1/2
Alternative 3	1/3	2	1

Table 10 results in the following priority value matrix for the changes to applications criteria:

$$\begin{bmatrix} 0.65 \\ 0.12 \\ 0.23 \end{bmatrix}$$

Table 11 – Maintenance Criteria

	Alternative 1	**Alternative 2**	**Alternative 3**
Alternative 1	1	2	4
Alternative 2	1/2	1	3
Alternative 3	1/4	1/3	1

Table 11 results in the following priority value matrix for the maintenance criteria:

$$\begin{bmatrix} 0.56 \\ 0.32 \\ 0.12 \end{bmatrix}$$

After obtaining the priority matrix from each of the alternatives, it is time to obtain the compose priority of them. That calculation is made by multiplying the priority matrix by the matrix with the weight of the criteria:

$$\begin{bmatrix} 0.44 & 0.16 & 0.65 & 0.56 \\ 0.12 & 0.30 & 0.12 & 0.32 \\ 0.44 & 0.54 & 0.23 & 0.12 \end{bmatrix} \times \begin{bmatrix} 0.45 \\ 0.1 \\ 0.28 \\ 0.17 \end{bmatrix} = \begin{bmatrix} 0.49 \\ 0.17 \\ 0.34 \end{bmatrix}$$

After analysing the resultant matrix, Alternative 1 seems to be the most indicated alternative of the three to implement independent releases in the FCN.

6 Analysis and Design

In this chapter, the solutions and respective design for the problems raised in this project are presented. This chapter is organized by approaching the problems, grouping them when such makes sense and respective solution in one different section.

6.1 Independent Releases and Applications Running on the Same Machine

Both independent releases and applications running on the same machine can be approached together. By allowing applications to have independent releases, the architecture as a whole can be rethought. To solve them, two different approaches were designed. However, the final solution gathers the best of all the approaches. In the end, the designed solution contributes to the NFR7 and NFR39 completion.

6.1.1 Approach 1

To better understand what is the desired deployment process for Approach 1, Figure 44 illustrates it through a UML activity diagram.

On the deployment process of IAP's Applications, there is the need to change the process to contemplate (I) a process to upload the static files from the applications to the CDN; (ii) and a series of calls that are going to be made to the IAP API in order to register the applications. All the applications pipeline's need to be updated to follow the new deployment process.

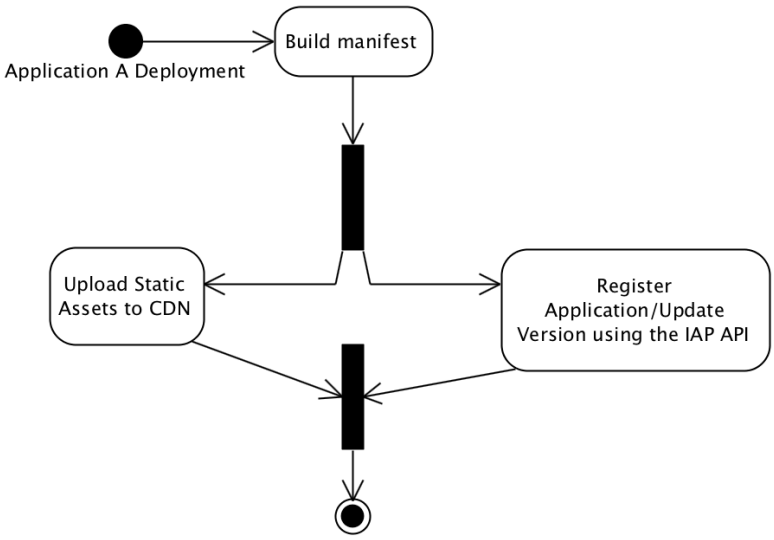


Figure 44 – Approach 1 Deployment Process

The core concepts of the first approach consist in (i) the applications only having strands and static files deployed in their machines, being the static files the result of the web application itself, leading to a homogenous pipeline; (ii) the deployment process registers the application in the IAP API and uploads static files to a CDN; and (iii) to endow IAP of an API and NoSQL data storage. The NoSQL as a data storage was chosen instead of a SQL one because it gives more dynamism regarding the data schema used [67].

Having the major components of the solution being identified, it is possible to draw the Figure 45 components diagram.

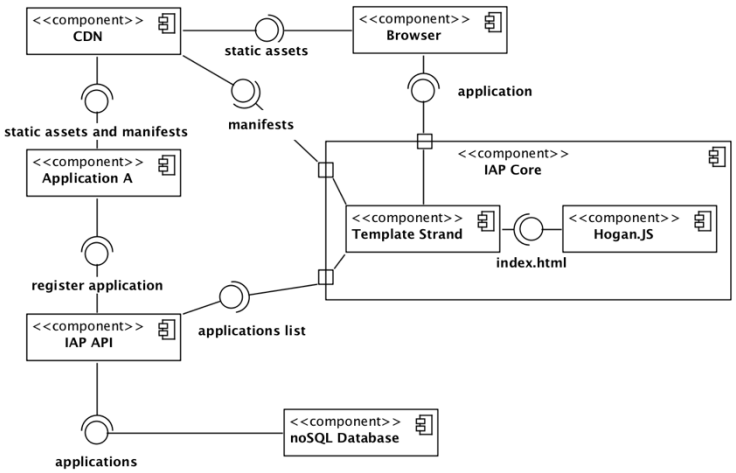


Figure 45 – Approach 1 UML Components Diagram

Analysing Figure 45 it is possible to see that IAP Core is similar to the FCN component in Figure 2, with the exception that the application now leaves outside the framework (IAP) scope.

The major differences are the introduction of an API (IAP API) that provides (i) an API for the applications to register/update themselves, as designed in Figure 44, (ii) and another API for the Template Strand to fetch the available applications for the users to know what are the applications they can use.

Figure 46 shows the components interactions during the applications' releases and deployments.

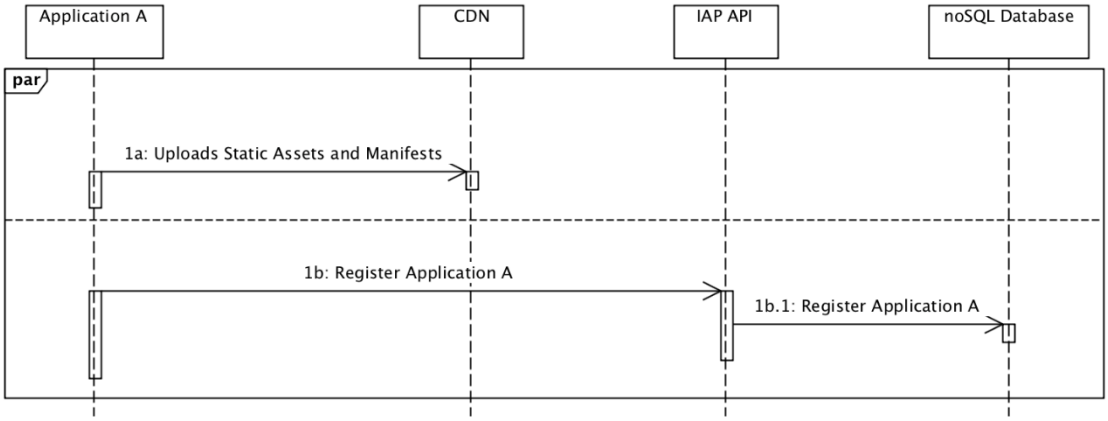


Figure 46 – Approach 1 Application Registration/Update during Releases and Deployments UML Sequence Diagram

Instead of the applications being part of IAP, the application static assets are uploaded to a CDN. This way, there is a clear separation between IAP and the applications. IAP only fetches and loads what the applications registered in the manifest and what it uploads to the CDN. Such can be depicted in Figure 47, where the interactions needed for a page to be loaded are exposed.

The described processes allow to establish that the IAP API must support (i) a get request to fetch all the available apps; (ii) a get request to fetch the manifest file of a specific app; (iii) and a put method to update the manifest of a specific application. A noSQL database must also be created to store the apps manifests. Finally, the template strand must be refactored to get the applications from the IAP API and the assets from the CDN.

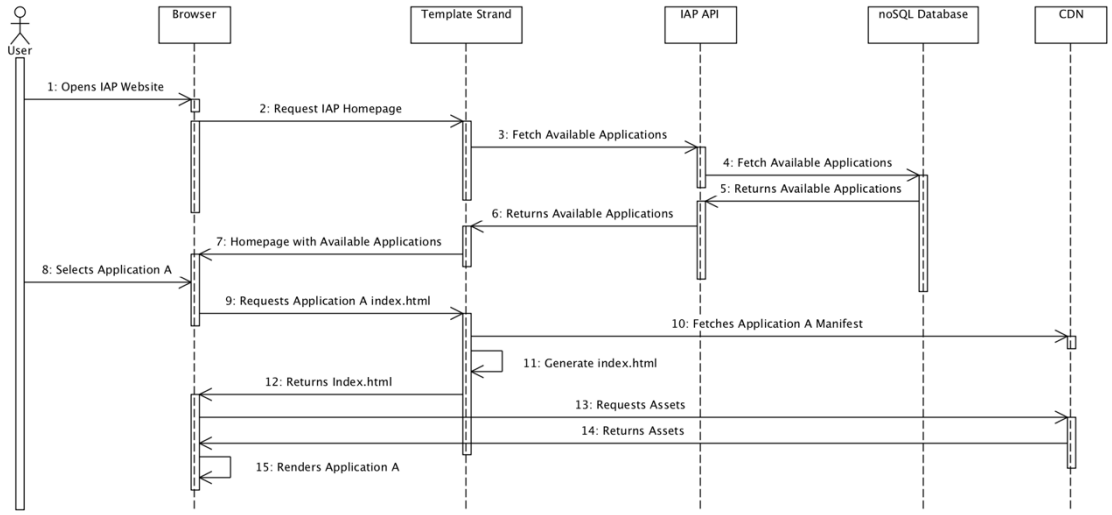


Figure 47 – Approach 1 Application A Loading UML Sequence Diagram

Since the responsibilities of IAP and the applications are well defined, it is possible to design a deployment diagram that also enables that every application has its own cluster. With that purpose, Figure 48 was designed.

Every application has its own cluster and IAP does not need to be aware of what execution environment it is on since the point of contact is the CDN. IAP Core and IAP API are in different clusters to isolate and prevent errors propagation. This way, even if IAP API fails, IAP Core can still generate some error pages and give the users some visual feedback.

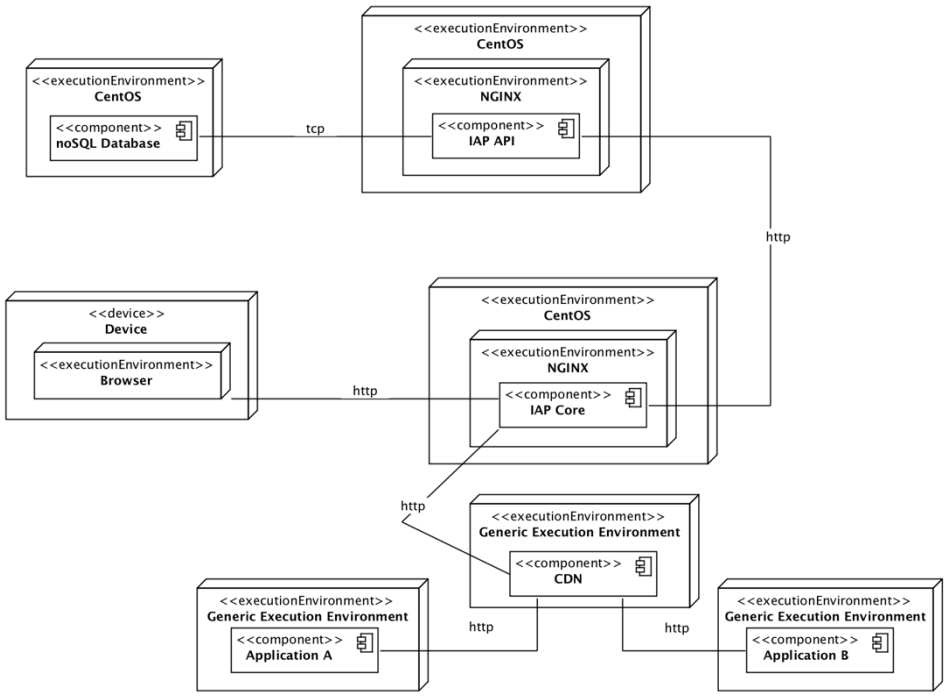


Figure 48 – Approach 1 UML Deployment Diagram

6.1.1.1 Advantages

The main advantages of this approach are (i) the app owners really own what is deployed into the different environments; (ii) homogeneous pipeline; (iii) the power given to IAP by having a database and an API will allow a more powerful set of features and independence to be provided; (Iv) and has low impact on migrating the applications.

6.1.1.2 Disadvantages

As for disadvantages, this approach requires the study of which step of the pipeline is the best suited to register the applications and the amount of work to maintain the new database and API.

6.1.2 Approach 2

The second approach has many similarities with the first one, but it tries to shift more responsibility into the application teams. This shift of responsibilities would also simplify the deployment process of an application, as it can be seen in Figure 49.

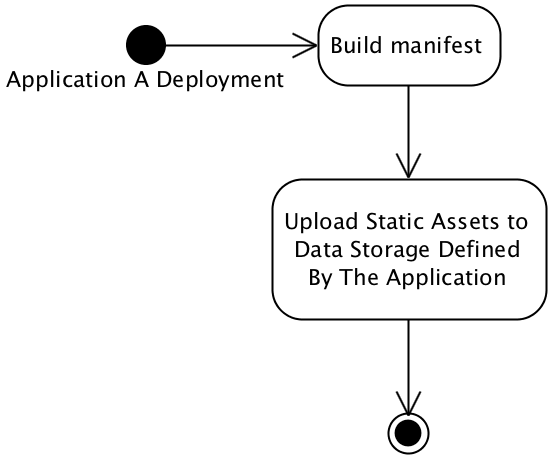


Figure 49 – Approach 2 Deploy Process

Similarly to Approach 1, there would be a step in the applications' pipelines to upload the static files that would be consumed by the Application API. Where such static files would be stored is not the concern of IAP, since in this approach it is detached from its scope. To the application teams, this imposes the creation of an API to expose the static assets and manifests files.

For instance, the IAP would create an API to fetch static files from the applications. These static files should be exposed by the applications through a well-defined API. Teams would have the freedom to decide what would be the best way to store and serve the files. Figure 50 depicts the components that are part of this solution.

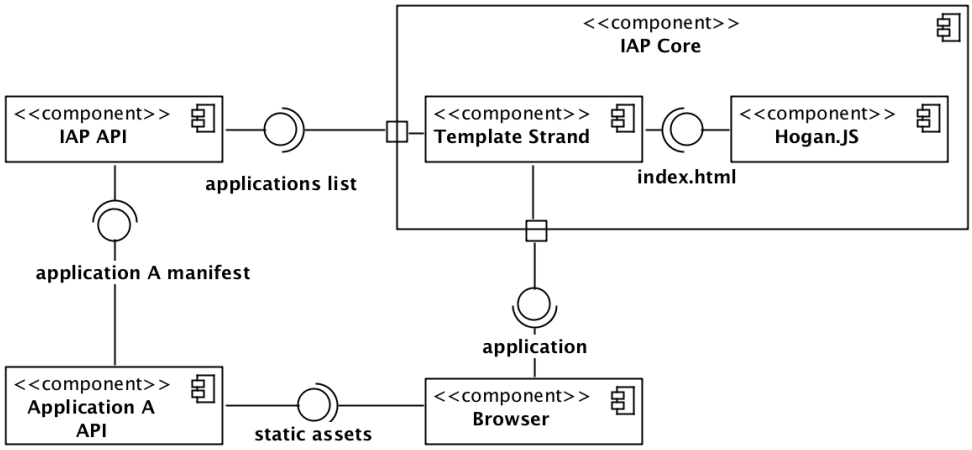


Figure 50 – Approach 2 UML Components Diagram

IAP API would have the list of applications and their respective API's endpoints hardcoded since adding and removing applications from the IAP is not a recurrent process. IAP API would behave only as a Façade to fetch the requested application manifests. How those applications' resources and manifests are stored is completely up to their respective team.

Figure 49 already portrays a simpler deployment process since the applications do not have a registration step. Regarding how an application is loaded and shown to the user, the process is in all similar to the one portraited in Figure 47. They diverge when it comes to fetching the application assets and manifest as described in Figure 51.

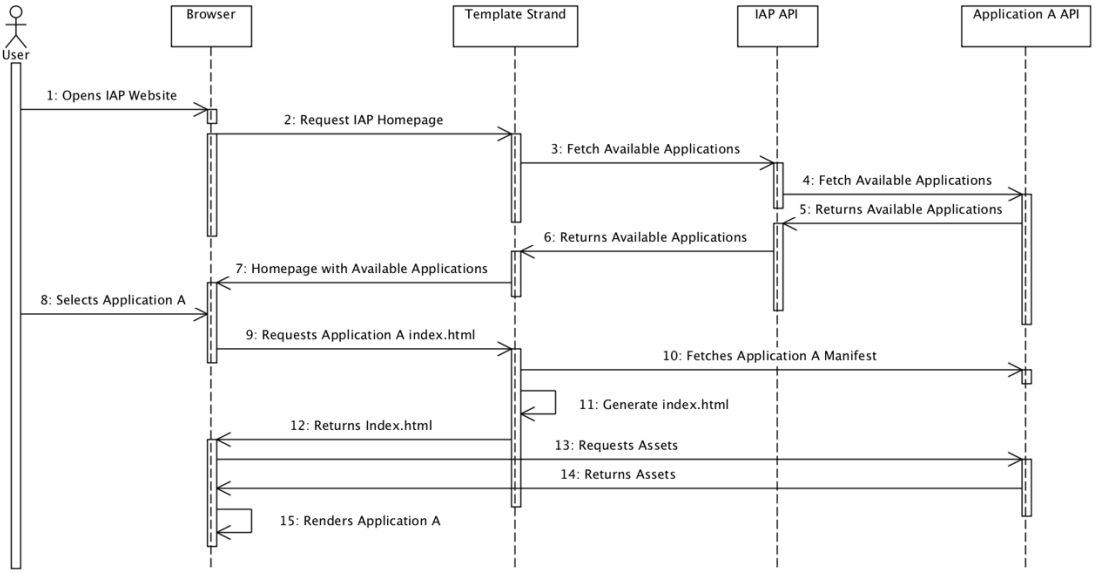


Figure 51 - Approach 2 Application Registration/Update during Releases and Deployments UML Sequence Diagram

Having the boundaries well defined, it is possible to draw the deployment diagram present in Figure 52. It is in all similar to the one designed in Figure 48, only changing the components that are deployed and reducing the IAP interactions with other components (like the CDN and noSQL database when compared with section 6.1.1).

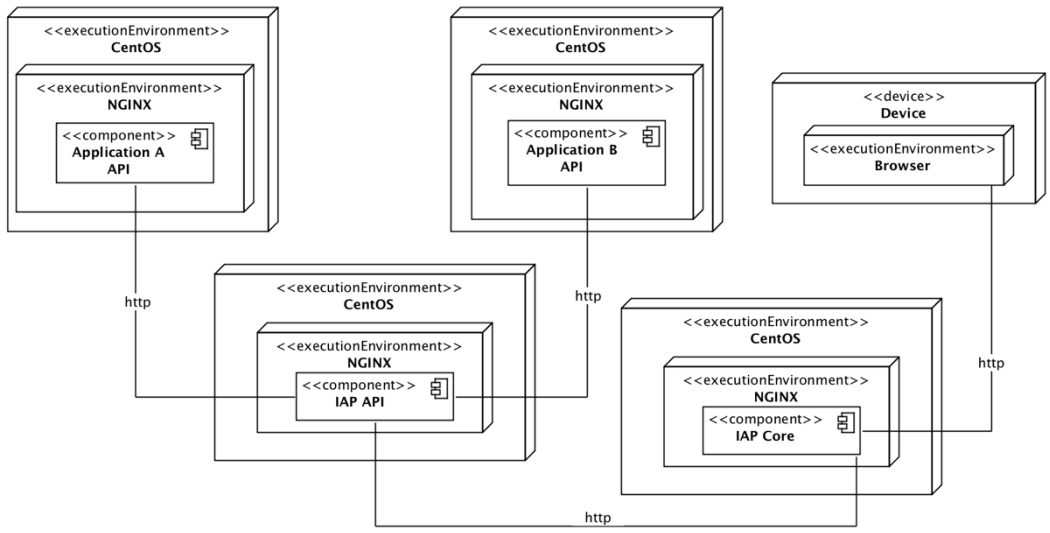


Figure 52 – Approach 2 UML Deployment Diagram

IAP is only responsible to maintain two blocks, them being the (i) IAP API and (ii) IAP Core. The rationale behind splitting IAP API and IAP Core into two different execution environments is the same used in Approach 1. Even if IAP API fails, IAP Core can still generate some error pages and give the users some visual feedback.

6.1.2.1 Advantages

The main advantages of this approach are (i) the app owners really own what is deployed into the different environments; (ii) homogeneous pipeline; (iii) IAP no longer aware of applications versions. Teams would be responsible for the build that is in each environment; (Iv) teams control their applications and releases, only needing to obey to a well-defined API; (v) and it allows to control what applications can be part of IAP or not;

6.1.2.2 Disadvantages

As disadvantages, this approach requires (i) the study of which step of the pipeline is the best suited to register the applications; (ii) has a high impact on applications to migrate to this approach; (iii) since every application would have to expose an API, that could lead to a management nightmare; (iv) and every time there is the need to add or remove an application from IAP a new deployment is needed.

6.1.3 Selected Approach

After understanding each approach and the respective advantages and disadvantages of each one, it was possible to draw a solution by using the best of what was learned from the developed approaches. The desired deployment process can be found in Figure 53.

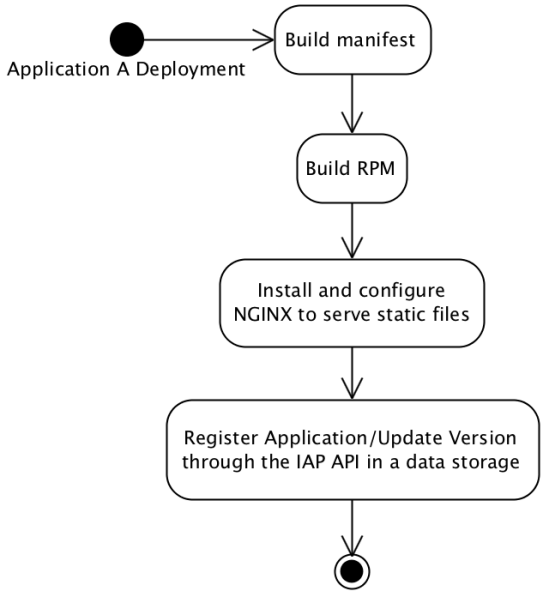


Figure 53 – Selected Approach Deploy Process

On the deployment process, there is the need to change the process to contemplate (I) a process to configure NGINX to serve the static files, strands and define any reverse proxies required; (ii) and a call that is going to be made to the IAP API in order to register the applications; All the applications pipeline's need to be updated to follow the new deployment process.

The selected solution, consists of (i) the application pipeline deploying their own strands and serve their static assets through a web server, NGINX (having a homogenous pipeline); (ii) the deployment process register the application in the IAP API through the manifest; (iii) endow IAP of an API and database; (IV) improve the development environment and process. Figure 54 depicts the identified components that are part of the solution.

This approach requires some changes to the FCN. In the IAP scope, a new API, the IAP API, must be created and must support (i) a get request to fetch all the available apps; (ii) a get request to fetch the manifest file of a specific app; (iii) a put method to update the manifest of a specific application; (iv) and the logic from the template strand must be added. A database must also be created to store the apps manifests.

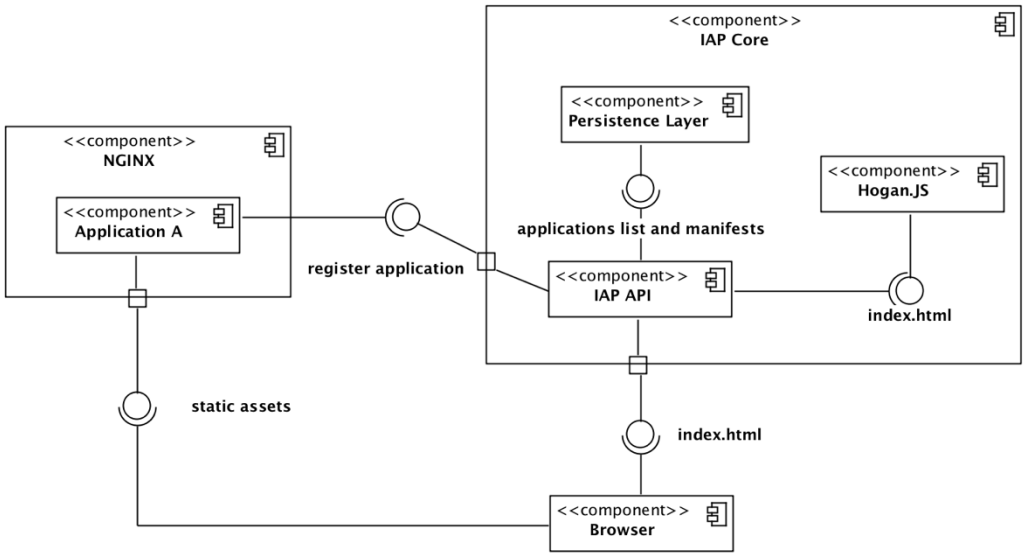


Figure 54 – Selected Approach UML Components Diagram

This solution has a similar application loading sequence with both Approach 1 and Approach 2. The main difference between them is where the assets and manifests are fetched, as seen in Figure 55.

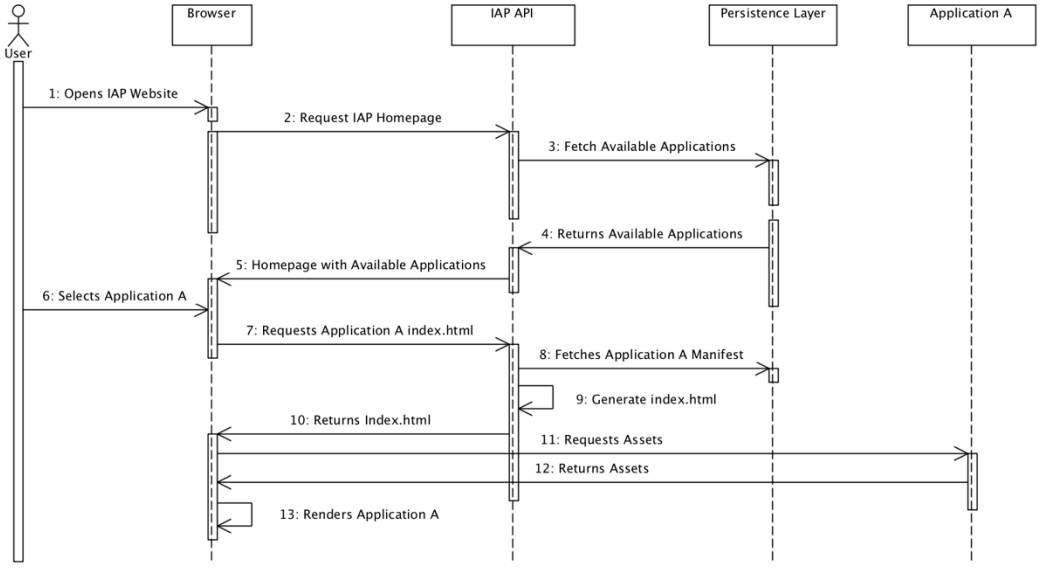


Figure 55 – Selected Approach UML Sequence Diagram for Application Loading

IAP API is now responsible for fetching the manifests from the persistence layer while the browser, when it loads the application index.html, is going to fetch the resources where the application is deployed.

When it comes to the deployment design, Figure 56 depicts the selected approach. It follows the rationale of both Figure 48 and Figure 52.

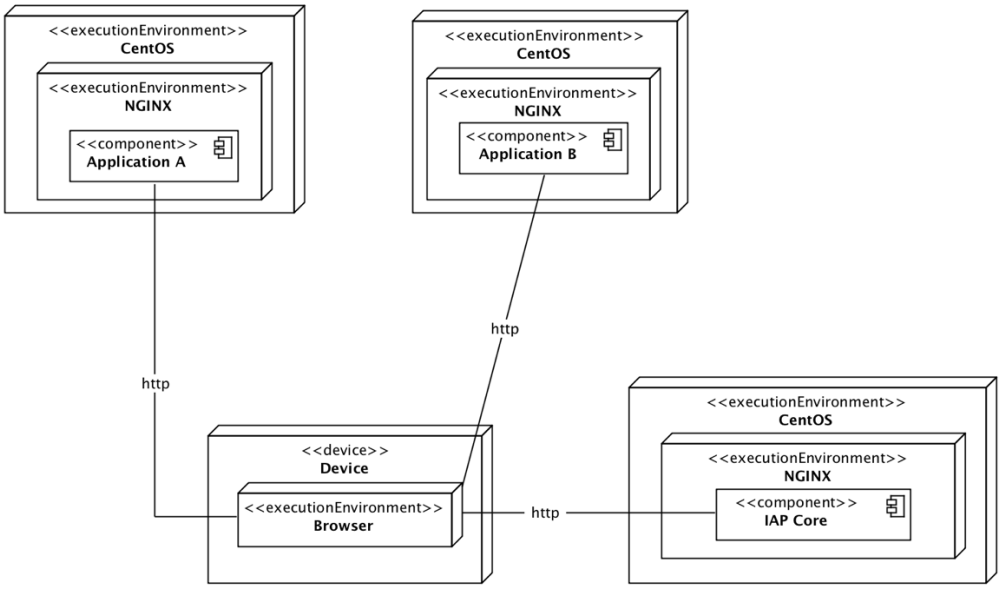


Figure 56 – Selected Approach UML Deployment Diagram

There are some considerations and some discussion that can be raised when analysing this solution. In comparison with FCN, this approach may look like it needs more resources. This is because while in FCN every application was on every machine, now, every application has their own cluster.

That statement is not necessarily true. While overall there are more machines in place, IAP is now responsible for a smaller number. Every application is now responsible for their own cluster which reduces the number of machines in IAP (c.f. NFR22). Moreover, the company has no resources limitation at the time of this writing so there is no problem if, overall, there is the need for more machines.

There are also two ways of storing the manifests: (i) the one presented in this solution (a database); (ii) and the usage of Artifactory, which is a third party. Even though the database was the desired approach due to the flexibility and independence provided, the software architects decided that the usage of the Artifactory would provide a quicker integration, making the delivery of this chunk of work a lot faster.

It is important to stress out that this solution makes leverage of the company's infrastructure (c.f. NFR38). Infrastructure is built upon OpenStack [68] and, therefore, many of the work is simplified. NFR19 to NFR21 are good examples of the advantages of leveraging on the company's infrastructure.

6.2 Complex Theme Scheme

The complex theme scheme and consequent lack of a consistent look and feel between the applications is one of the more intricate problems. On one hand, there is the need to simplify the way components and themes are provided. On the other hand, one of the philosophies of IAP is to give freedom of choice to the teams.

6.2.1 Approach 1 - Material Design

One of the first approaches to be discussed was the idea of grouping all the applications under a single visual language that had standards and well-defined patterns and could be easy for applications to use. The solution that best fit into the requirements was the usage of Material Design. Material Design consists of a visual language that joins the standard principles of good design with the technological and scientific innovations. It unifies the experience across platforms and devices by providing a single underlying system to applications [69].

6.2.1.1 Advantages

One advantage of the usage of Material Design is that there are implementations of several components for multiple frameworks using these principals. For instances, for Angular, there is Angular-Material [70], for React, there is Material-UI [71] and for Vue, there is Vue Material [72].

6.2.1.2 Disadvantages

Such offer reduces not only the IAP teamwork of maintaining the components but it also reduces the teams that use them since there are a lot of well-defined components ready to use in a lot of frameworks.

However, using Material Design would force teams to use a specific Material Design implementation, which would go against the freedom that this new version wants to bring to developers.

Another problem is that the projects that are already built with FCN or projects that would want to migrate to the new version can offer resistance or have problems migrating their current approach to Material Design. That can happen because the way that the projects handle look and feel can be totally disruptive regarding the Material Design approach.

Lastly, although today we can assure what are the most used frameworks to build applications, those may not be the same in the future. This leads to the problem of assuring that there is a Material Design implementation for every single one of them. While today the usage of Material Design implementations can make sense, the same cannot be sure in the future.

6.2.2 Approach 2 - Common Theme

Most of the CSS frameworks offer ways to override and customize its components. Not only the previously mentioned Material Design implementations but also Bootstrap [73] and its variations, like React Bootstrap [74] offer ways to do so.

One discussed solution was the possibility to create a theme that could be used by the CSS frameworks that the teams use for their applications.

6.2.2.1 Advantages

The major advantage of this approach is that it would not imply that teams followed a certain technology since the ones being currently used and the most used globally support customisation.

6.2.2.2 Disadvantages

However, the mechanisms that each of the technologies uses for customisation is different. Such would imply that for each framework being used, a new theme, or at least a new structure, should be built. Let's take as an example React-Bootstrap and Material-UI, both built for React applications.

In React Bootstrap, there is a *bsStyle* property that can be used to map a Bootstrap class for styling, as seen in Code 7.

```
bootstrapUtils.addStyle(Button, 'custom');
const customButtonStyle =(
<div>
 <style type="text/css" >{`
        .btn-custom {
               background-color: purple;
               color: white;
        }
 `}</style>
 <Button bsStyle="custom">Custom</Button>
</div>

render(customButtonStyle);
```

Code 7 – React Bootstrap Customisation

In Material UI, there is an API called *MuiThemeProvider* that takes a theme and makes it available down the react tree. Every component used will inherit the defined theme. An example of the code can be seen in Code 8.

```
import React from 'react';
import { render } from 'react-dom';
import { MuiThemeProvider, createMuiTheme } from '@material-ui/core/styles';
import purple from '@material-ui/core/colors/purple';
import green from '@material-ui/core/colors/green';
import Root from './Root';
```

```
const theme = createMuiTheme({
  palette: {
        primary: purple,
        secondary: green
  },
  status: {
        danger: 'orange'
  }
});

function App() {
  return (
        <MuiThemeProvider theme={theme}>
              <Root />
        </MuiThemeProvider>
  );
}

render(<App />, document.querySelector('#app'));
```

Code 8 – Material-UI Customisation

Since the mechanisms used by each technology can be different, the overhead of creating and maintaining such an approach would be unmanageable. Not only the combinations of possible cases can be enormous as the complexity of a solution that could abstract as much as possible the theme to be adapted to every possible technology could be even more complex than the FCN. For such reasons, this approach was left out.

6.2.3 Approach 3 - UI Kit + Implementation

Since it is expected that the UI/UX teams deliver a UI Kit with the components' styles, one approach could be to implement them using a naming convention like BEM [75]. In the end, regardless of the technology chosen to implement the UI components, we would only distribute a single CSS file per component.

The Code 9 shows an example of this approach being used. It shows a component being customized by CSS classes, using the BEM convention [75].

```
<button class="button" >Normal Button</button>
<button class="button button--state-success" >Success Button</button>
<button class="button button--state-danger" >Danger Button</button>
```

Code 9 – Approach 3 Usage Example

6.2.3.1 Advantages

Such an approach would be easy to maintain since the CSS files would only need to be updated if any change in the design was needed. No component behaviour logic would be implemented, which would ease the payload of the maintainers. Concerning the developers

that benefit from this approach, they would have more flexibility and ownership of the components they develop.

6.2.3.2 Disadvantages

However, this approach does not guarantee that the different applications use the provided styles or even if they do, does not guarantee that they would not be overwritten or changed. A good guide and applications follow-ups can be put into practice to avoid such changes.

6.2.4 Approach 4 - Web Components + UI Kit + Implementation

Instead of providing only the styles, web components that allow for customization can also be offered. This approach is a mix of Approach 3 but applied in web components.

Code 10 shows an example of this approach being used. It shows a component where its colour, width and height are configurable.

```
<burguer-menu color=”#ffc835” width=”32” height=”32” ></burguer-menu>
```

Code 10 – Approach 4 Usage Example

6.2.4.1 Advantages

The major advantages of this approach are that it is easier to use since it offers a simpler and cleaner API. It also encapsulates the component structure (CSS and HTML), leaving developers with the single responsibility of configuring the available properties.

6.2.4.2 Disadvantages

However, this adds a lot of costs regarding maintainability since the team that owns the framework is accountable for every change that needs to be made. Moreover, the number of components can scale in uncountable ways which would only add more load to the team.

6.2.5 Selected Approach

The selected approach end up being a mix of the approach 3 and 4, being each one more appropriate for specific cases.

For simple components (buttons, dropdowns, radio buttons, amongst others) that are already provided by other frameworks or CSS Libraries, Approach 3 is applied. This way, the teams' technological choices are not undermined and the IAP team is only focused on providing the appropriate styles.

For more complex components (form, headers, menus) Approach 4 is the most appropriate one. Not only it is easier for the teams to use, but it also allows the IAP maintainers to control the component behaviour, enhancing the overall platform Look-and-Feel.

One important aspect that was also thought through, is the case that when a team is developing an application on a specific framework, it can create a component that can be used by other teams using the same framework. Following this line of thinking, another approach was created. When a component is tightly coupled with a framework, but it is generic enough to be shared and fit different needs this component must be part of the framework as well. Furthermore, it helps to cultivate a Corporate Open-Source mindset.

To better understand when each approach is more suited, Figure 57 containing a decision diagram was created.

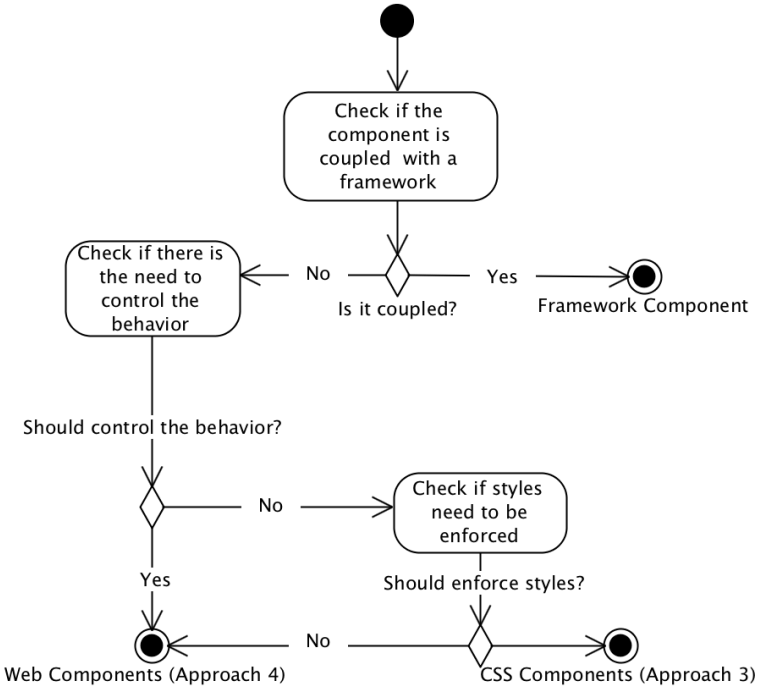


Figure 57 – Components UML Decision Diagram

When planning to create a component that leverages in specific framework functionalities, that component should be provided as a framework component. This way, teams that also use that framework can make use of it. Moreover, its development and maintenance can be eased due to the framework capabilities. It was not depicted in an approach since it is not in IAP scope to develop specific framework components.

If, on the other hand, there is the need to control the behaviour of a specific component between applications (for instance, to keep a consistent behaviour across applications) web components should be used. By using web components, it is possible to create framework agnostic components that can be used in any application and whose behaviour can be enforced.

Lastly, if no control behaviour is needed but there is the need of enforcing a specific style, for UI consistency, then CSS Components must be created. CSS Components only offer styles, leaving the behaviour and component implementation up to the teams. If no styles enforcing are needed, that web components should be created since it allows to share the components between every framework.

This solution is a great contribution to both NFR10 and NFR16. For NFR10, Figure 57 helps to understand how are the UI components now managed and always try to use the simplest approach. CSS Components can be used to provide different themes and Web Components and Framework Components can extend the CSS Component styles (c.f. NFR16)

6.3 Technological Dependencies

One of the biggest challenges of this project is how to provide a framework that is agnostic. By other words, how can a framework be provided without constraining the applications (c.f. NFR11) technologies. The FCN already uses a working and tested templating mechanism (see subchapter 2.4) and, therefore, a decision was made to keep the same mechanism.

Currently, there is only one mustache template that only works with Angular. To embrace all possible technologies, an agnostic template is going to be created. To make a template agnostic, there is the need to only allow the injection of CSS, HTML and Javascript. Besides that, only crosscutting concerns (monitoring scripts, storage services and others) can be present in this template. Teams will have the freedom to create a more specific template for their needs. For instance, if a React application needs to have some React logic to the template, a new one should be created so it can be shared with those who may have the same needs.

However, it is important to understand how does an application choose between different templates. For that, a series of approaches are discussed in the following sections.

6.3.1 Approach 1 – Metadata File

The metadata file would contain all the information that allows the IAP API to serve an application. This application is served through an index.html that is generated by the IAP API using the metadata file of the application and one of the many provided templates.

Since all the information from an app comes from the metadata, it makes sense to add the template information to the metadata.

6.3.1.1 Advantages

The major advantages of this approach are the flexibility and usability, which allows teams to easily change it and update it.

6.3.1.2 Disadvantages

One typo can break the application and leave it unavailable. For instance, if the wrong template or an inexistent one is chosen, the application will return a 404 error page. Mechanisms to prevent it, such as validation of the metadata file structure, must be implemented.

6.3.2 Approach 2 – Pipeline

The same way there are lists with several options when triggering a new build, instead of being read from the project metadata, it could be passed as an argument into the register app script. This would happen in the last step of Figure 53. When registering/updating the application, it would update the manifest with the chosen template. Figure 58 shows where the developer would choose the manifest.

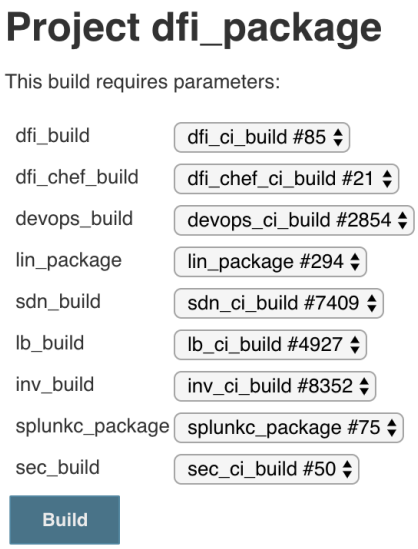


Figure 58 – Application Build Parameters

Figure 58 shows all the dependencies that are going to be installed in the machines once the application is deployed. This is the last step of an application deployment and, therefore, a new dependency could be added for the developer to choose what manifest would the application use.

6.3.2.1 Advantages

With this approach, no problem would occur due to typos.

6.3.2.2 Disadvantages

This approach introduces another kind of error: selecting the wrong template. The same way a typo can be introduced, it can be easy for a developer to select the wrong template. Moreover, this solution does a lot of overengineering to overcome a typo error.

6.3.3 Selected Approach

Approach 1 is already being partially used and it overcomes the objectives. The second one not only is harder to implement but is also harder to maintain and does not fully overcomes the problem raised by the first approach. Having this in mind, the chosen approach is the first one.

6.4 Documentation

One of the major complains about FCN was regarding the outdated documentation or worst, the lack of it. No documentation can lead to code duplication and can frustrate the framework/platform users.

However, one must not fall short into thinking that only formal documentation must be created. Martin Fowler classifies code as being the primary documentation of any software system but that does not mean it should be the only one [76].

Being that said, one of the major approaches to overcome this problem is to create a cleaner and more readable code. Not only it will improve the frameworks maintainers knowledge of the code but will help those who want to use it or contribute.

Regarding formal documentation, several approaches were discussed. Such approaches are expected to help to achieve the NFR12, NFR13, NFR14 and NFR15.

6.4.1 Approach 1 – READMEs and Wiki on Gitlab

READMEs and Wiki on Gitlab are already being used in FCN.

The major advantage is that the documentation is close to the code. When a developer is updating the repository, he/she can easily update the README and the wiki without big effort.

However, due to the Microservices and Microfrotends [77] approaches, there are dozens of projects. This could mean that in order to find the needed documentation, a developer must have a deep knowledge of the framework architecture and organization which, in some cases, may not be needed. It can also lead to duplicated documentation.

Another problem that arises is where should documentation that covers multiple modules be stored.

6.4.2 Approach 2 – Confluence

Confluence is the company's central repository of documentation. It is usually used to document Software Architectures (c.f. NFR14), APIs, SLAs, project drafts and others. Since it is

the company's standard, it is a known point for the IAP users and developers to find the documentation.

Nevertheless, Confluence is not commonly used to document code. Furthermore, the Confluence search engine proves to be a challenge when used to find the desired documentation.

6.4.3 Approach 3 – Central Documentation Website

Creating a Central Documentation Website would create a single point of access to all documentation regarding IAP. Not only it could contain more generic material like getting started guides, and organisation, as it could contain documentation on how to use a certain component and what API does it offer (c.f. NFR12, NFR13, NFR15). It would be a clear and simple way of accessing the documentation. One example of this approach is the React's [78] and Angular's [79] documentation websites.

However, such flexibility comes with a cost. This approach has the overhead of the website creation, maintenance and deployment. Every change made to the documentation requires a new deployment.

6.4.4 Selected Approach

Despite the fact that it is the one that is harder to implement and to maintain, the benefits of Approach 3 make it the best choice for the raised needs. Such kind of documentation is easy to distribute, find and it only presents content related to the framework/platform, reducing the context switching of the other approaches.

However, as stated previously, this three approaches can be used together to overcome the defects of each other.

It is also important to stress that more than a technological problem, this is a problem regarding the will of developers to create documentation and, more important, good and meaningful documentation.

6.5 Logging

One of the problems with FCN is the lack of logging (c.f. NFR9). Logging is extremely important to trace the errors and reduce the action time needed to find problems in the system.

There are already mechanisms in the company that ease the work of having logs of the system (c.f. NFR38). Splunk [80] is the company's log management system. It is used to search, analyse and visualize the logs that are previously collected from a website, machine and others [81].

This problem can be approached in two different areas: Frontend and Backend logging.

6.5.1 Frontend Logging

By abstracting the mechanisms needed to generate and collect logs from the frontend, not only code duplication is being prevented, but the teams that develop the applications can focus only in understanding where should they place the loggers in order to collect the most meaningful data possible.

Figure 59 and Figure 60 illustrate the solution for the frontend logging.

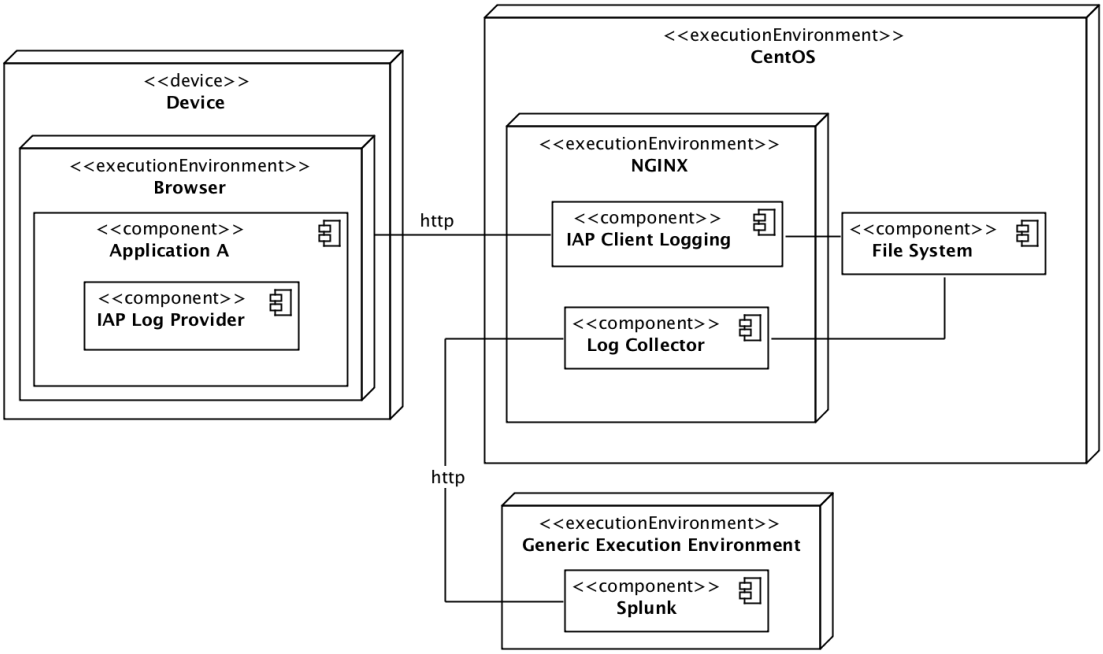


Figure 59 – Frontend Logging UML Deployment Diagram

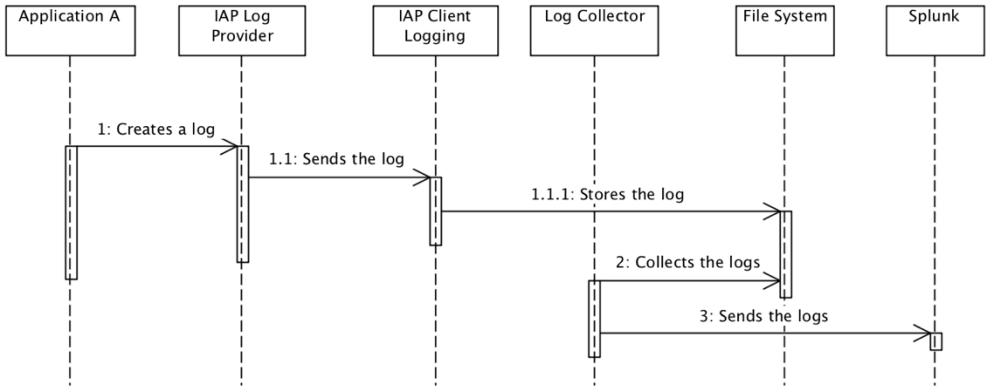


Figure 60 -Frontend Logging UML Sequence Diagram

There are some aspects worth noticing. Three of the components that are used in the solution are already implemented and are company standards, being them the (i) Log Collector; (ii) the File System, that is part of the CentOS operative system; (iii) and the Splunk. Only the IAP Log Provider and IAP Client Logging would require some work to implement.

The IAP Log Provider would be responsible to send the application logs to the IAP Client Logging. This last one would then store them in the File System of the operative system where it is running. This is necessary since the log collector is going to search for log files in that same file system.

However, this solution has the problem of all metrics being stored in the same cluster. All applications that would use this solution would end up with their logs in the IAP cluster. This raises the concern of the storage available. This log centralization if in one hand saves work for the applications, on the other hand, one single application can compromise all others if not managed carefully.

One solution to fix this problem would be to deploy the IAP Client Logging in each application cluster instead of the one from IAP. This way, each application would also be responsible to deal with the storage available in its cluster and there the problem would not propagate to other applications.

6.5.2 Backend Logging

The IAP will also provide several backend services, such as Authentication, Authorisation, Templating and many others for both the applications and the framework itself. The approach to have Backend Logging is similar to the one used in 6.5.1. Three of the components will still be used being them the (i) Log Collector; (ii) the File System; (iii) and Splunk.

Since all the backend services are going to be Strands, aggregation functions ran by Fabric [3], a company's standard for NodeJS services, no work is needed. Fabric already offers logging mechanisms, contributing to the achievement of the NFR23 until the NFR34. Nevertheless, Figure 61 helps to understand the workflow.

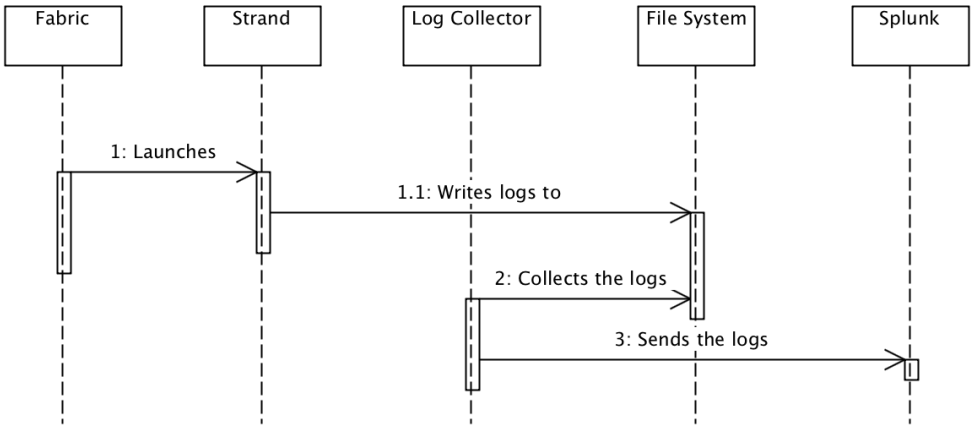


Figure 61 – Backend Logging UML Sequence Diagram

6.6 Conventions

Having coding conventions is mandatory. To assure a smooth and similar experience between different applications, a set of guiding rules must be created. Since everyone writes code differently without no convention, the code can become a mess. This way, new developers adaptation to a project is faster and the written code is easily understood by everyone.

There are multiple popular code style guides available in the market like (i) Airbnb Javascript Style Guide [82]; (ii) Google Javascript Style Guide [83]; (iii) Idiomatic Javascript [84]; (iv) and Javascript Standard Style [85];

However, style guides most important aspect is that every team member is comfortable with the selected guiding lines. If the developers do not feel comfortable or show resistance to the guidelines, it can have the adverse effect. Developers can be demotivated by being forced to use such rules or they can even ignore them. As stated in subchapter 2.8, different technologies, different projects and different teams can have different needs regarding such aspects. If, on one hand, the goal is to give teams more freedom of choice, forcing conventions can undermine such goal.

Having this in mind, the solution consists in revisiting the *eslint-config-fcn component* and updating it, having in mind that some decisions may not be accepted by every team. However, since this component can be extended and therefore some rules are overwritten to fit other teams needed, a guide on how to do so must also be provided.

This approach does not entirely fix the no conventions between projects problem, but it reaches a middle ground between having some conventions amongst projects and allowing teams to have their own conventions, using the *eslint-config-fcn* as a basis.

6.7 Authentication and Authorisation

This is one of the most sensitive topics. Not for its complexity, but because there are a lot of dependencies of the company software architects. It was important to discuss and study if the company already provided an Authorisation mechanism. The Authentication mechanism, discussed in 2.6, will remain the same as in the FCN and since in FCN the NFR2 to NFR6 are completed, the same is going to be true in IAP.

For the Authorisation (c.f. NFR8) there is a company's service, Admin Identity Service (AIS) [86]. AIS aims to offer a unified way to retrieve admin identity information associated with a given account. To clarify what an admin is in this context, an admin is a company internal user. In other words, is the account of a Paddy Power Betfair employee, which are the ones that have and will have access to IAP.

AIS allows the existence of permissions by product so a parallelism can be drawn between a product and an application. Every IAP application will be mapped into an AIS product.

An AIS product has associated a list of configurations, each one of them having three properties:

- Action – The action that is being performed in the product.
- Permission – The permission that a user must have to perform the action (can take the form of VIEW, EDIT or UNABLE_VIEW)
- Role – The role of the user (for instance, if it is a developer, a trader, a manager, amongst others)

The process of adding or updating Roles, Permissions and Actions to an AIS product has a well-defined yet bureaucratic process. A request with the desired changes to the Roles, Permissions and Actions should open in the company's requests management portal. Once opened, the AIS team is the one responsible for implementing such changes. This raises the concern of the applications' teams not owning the entire application development cycle. However, there is no solution or workaround.

After studying the AIS API [87], there are two methods that can be extremely important for providing Authorisation in IAP:

- retrieveActionsAndPermissions - When retrieving Actions and Permissions it is necessary to specify the product and the token session of the operator to retrieve all actions and permissions that a user has.
- retrieveAllRolesAndPermissions() - Is not possible to get permissions of a specific product, this method returns all the permissions for all existing products.

To better understand how would AIS be used, Figure 62 represents a practical use case in form of a UML sequence diagram

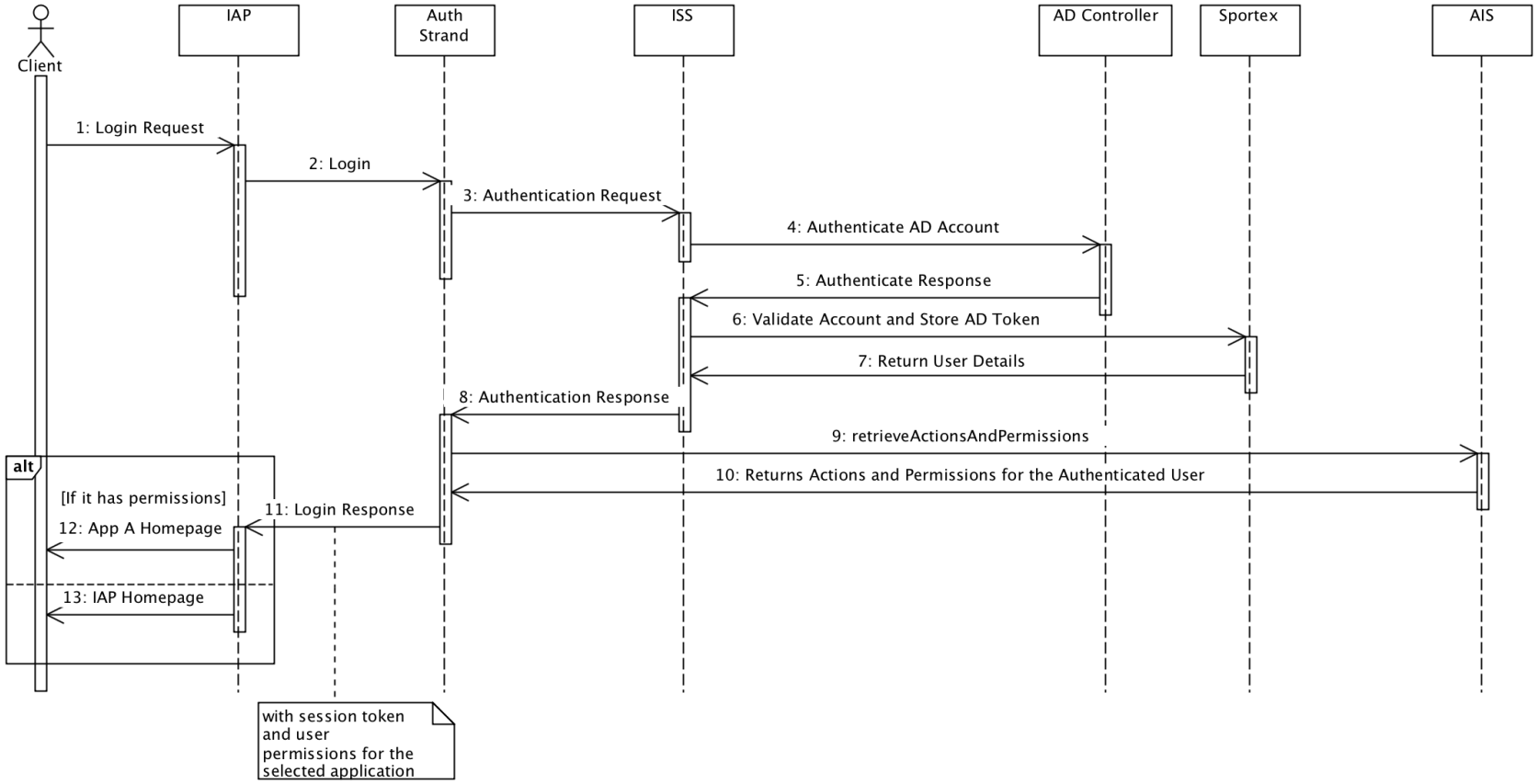


Figure 62 – Authentication and Authorisation UML Sequence Diagram

Everything is similar to Figure 8 with the exception of the Authorisation steps. When a user tries to login into a particular application the Auth Strand will communicate with AIS to retrieve the user permissions regarding that application. If it has permissions to login into it, IAP will redirect the user to the desired application. If not, the user will stay logged in into the IAP, but will be redirected to the IAP dashboard and will only see the applications to which he has access to (*retrieveAllRolesAndPermissions* would be used in this case to find what applications is the user able to see).

Another advantage of this approach is that applications will be able to access the user permissions. This way, the applications can restrict the user actions according to its role and permissions. As explained in this chapter, applications can add and update Roles, Actions and Permissions in AIS, so the responsibility is inherited for each application. IAP only provides mechanisms for the applications to know what are the permissions of the logged user.

6.8 Lack of User Interface Consistency

Regarding the lack of user interface consistency (c.f. NFR17), all the design was developed by the UI and UX teams of the company following the provided analysis of the existing solutions in subchapter 3.1. The provided design can be found in Annexe G.1. However, there are some improvements that the IAP team is still waiting. The application name and the background image to not reflect the new name of the platform. The header still refers to Fusion Console while the background image refers to another application called Yoda, an application that was developed with FCN but is deployed outside FCN scope.

It is important to notice that this does not solve the problem. However, it gives guidelines that applications must follow. It can also be an approval point of whether an application can use IAP or not.

6.9 Summary

Through this chapter, several approaches for each problem raised in sub-chapter 1.2 were developed and discussed with the objective of reaching a solution that allows that all of them can be overcome.

Regarding the independent releases and applications running in the same machine, there are two big chunks of work, them being (i) the IAP API, responsible for registering/updating the applications manifests and generating the index.html, (ii) and the persistence layer, responsible for storing the applications' manifests. The selected approach can be found in more detail in section 6.1.3

For the complex theme scheme, a decision diagram was created with the purpose of helping to understand what approach should be applied for each case. For IAP's scope, it is important

to understand how the (i) Web Components (section 6.2.3) and (ii) CSS Components (section 6.2.4) should be implemented.

When it comes to technological dependencies, a new template is going to be created. This template must be agnostic and, therefore, must not impose any kind of technology to the applications. The manifests currently being used must be updated to comport what kind of template must be chosen. More details can be found in section 6.3.3.

Documentation is now focused on a stand-alone website responsible to centralise all IAP related artefacts. These artefacts can go from high-level architecture, to how can the provided components be used. More details can be found in section 6.4.4.

Logging was divided into two separate pieces of work, (i) Frontend Logging (c.f. section 6.5.1); (ii) and Backend Logging (c.f. section 6.5.2). Regarding the Frontend Logging, the solution leverages on some components already developed by the company. The components that need to be implemented are the IAP Logging Client, that the applications use to generate the logs and send to a backend service; (ii) and IAP Client Logging Service, service that is going to receive the logs generated by the apps and stored them. When it comes to the Backend Logging, the services provided by IAP are going to use strands, that leverage on Fabric's existing logging mechanisms, so no work is expected at this level.

Conventions are a sensitive topic, as discussed in sub-chapter 6.6. To approach this problem, the solution consists in revisiting the already existing rules and update them, having in mind that other teams can overwrite them.

For the Authentication and Authorisation, it was defined that AIS is going to be used to provide Authorisation mechanisms. Mechanisms that allow the applications to know what are the permissions of the logged user must be created. More details can be found in sub-chapter 6.7.

Finally, the specs provided by the UI and UX Teams must be implemented to provide a smoother experience to the end user. More details can be found in sub-chapter 6.8.

7 Implementation

In this chapter, it is shown how the design was implemented. This chapter is organized by approaching each problem and respective design implementation in one different section.

7.1 Independent Releases and Applications Running in the Same Machine

To better drive this chapter, Figure 54 is used as a basis. Each component present in the solution has a description of its implementation.

One thing worthy of notice is that the IAP API was divided into two strands to promote a microservices approach and to use Fabric, which is an internal standard when building such kind of services (c.f. NFR38).

7.1.1 Persistence Layer

As stated in section 6.1.3, Artifactory was chosen as the Persistence Layer of the solution.

Artifactory follows as folder structure. Figure 63 shows the folders created to store the different applications manifests.

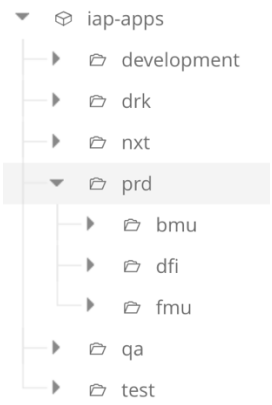


Figure 63 – Artifactory structure created to store IAP Applications.

There are three folder levels: (i) iap-apps, which is the root folder for this solution; (ii) the environment folders (e.g. production, quality, testing), which contains all the IAP Environments; (iii) and the applications folders that contain the manifests and are created and updated by the applications themselves through the IAP API. Figure 64 shows the structure inside each application folder.

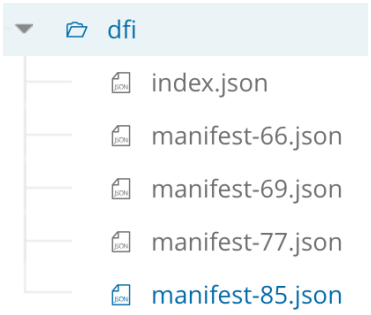


Figure 64 –Application Folder Content

Inside each of the application folder, there are two kinds of files: (i) an *index.json* (cf. Code 11), containing the current version to be used; (ii) and the manifests (cf. Code 12), that contains the information for that application in a specific version.

```
{
  "tla": "85",
  "version": "manifest-85.json"
}
```

Code 11 – Content of an application *index.json*

```
{
  "buildNumber": 85,
  "description": "DFI dashboard",
  "fullname": "IAP Dashboard",
```

```
    "hidden": true,
    "module": "dfi.dashboard",
    "name": "dfi-dashboard",
    "scripts": [
      { "src": "https://dfi.prd.internal/scripts/environment.85.js" },
      { "src": "https://dfi.prd.internal/scripts/dashboard.85.js" }
    ],
    "styles": [
      {
        "rel": "stylesheet",
        "href": "https://dfi.prd.internal/styles/dashboard.85.css"
      }
    ] ,
    "tla": "dfi",
    "bootstrap": { "root": "/" }
  }
```

Code 12 – Content of an application manifest in JSON.

It is important to understand why each of the properties present in the manifest were created and what are its impacts. A description of each property is provided as follows:

- **fullname** - Application's full name, to be displayed in the application selector menu.
- **hidden** – Boolean that indicates if the application should not be displayed in the application's selector menu.
- **module** – If creating an Angular.JS application, the application's angular module name must be configured**.** This was created with the purpose of allowing retro-compatibility with the apps created with FCN.
- **scripts** - This property is responsible for containing all the scripts that one application wants to inject in its index.html. This object is composed by one array for where each entry is an object containing an *src* property and its value. This structure was chosen to mimic the script HTML element [88].
- **styles** - This property is responsible for containing all the styles related files that one application wants to inject in its index.html. This object is composed by one array for where each entry is an object containing a *rel* and an *href* property and their respective values. This structure was chosen to mimic the link HTML element [89].
- **tla** – Three letter acronym that identifies the application.
- **bootstrap** – Allows the definition of some initial configurations. At the moment the only used property is root, which allows the configuration of the initial route.
- **template (optional)** – Option that indicates which template should be loaded in this application. Currently, there are only two: (i) agnostic; (ii) and legacy. It defaults to legacy.

The advantage of this approach is that it can easily scale if there is the need to add more features. There are already more features implemented, but these are the core ones and are the ones that are needed to develop this project.

Now that the persistence layer and its structure have been described, the next step is to document the layer that registers these manifests, the IAP API.

7.1.2 Registry Strand

One challenge of using Fabric is that each strand is responsible for one and one operation only. For a service responsible for the registration, many operations can be provided going from the registration itself to the consult of the registered applications and its manifests.

To overcome this problem and still use Fabric, GraphQL is used to implement this strand. GraphQL introduces two new concepts: (i) queries, that are used to retrieve information; (ii) and mutations, that are used to change and manipulate data [90]. All these are provided under one single endpoint [91], overcoming the problem introduced by Fabric. Another advantage is that GraphQL provides a type system that lets developers to create types to validate if the queries and mutations are valid or not (NFR1). This way, it is possible to prevent that misconfigured manifests won't be registered [92].

Having clear the purpose of using GraphQL in this strand, two queries and one mutation was created: (i) *getAppManifest* query, that given an application TLA returns the application's manifest; (ii) getAppsManifests query, that returns all applications' manifests; (iii) and the createAppManifest mutation, that given an object containing the manifest it registers that manifest in Artifactory. To better understand the implementation of the strand but at the same time to not overwhelm the reader with too much code detail, Code 13 represents the Registry strand core code content.

```
'use strict';

const Promise = require('bluebird');
const ArtifactoryError = require('../errors/artifactory');
const { IAP_CONFIG_PATH = '../config/config' } = process.env;
const config = require(IAP_CONFIG_PATH);
const env = config.IAP.env.toUpperCase();

/**
 * Fetches all app manifests
 * @param {Object} root - root configuration
 * @param {Object} args - query arguments
 * @param {Object} context - graphql context
 * @returns {Promise<Array>} promise resolved with array of all apps
manifests
 */
const getAppsManifests = (root, args, { $iapRegistry }) =>
    $iapRegistry
        .getIndex(env)
        .then(({ children }) =>
            Promise.map(children, child =>
                $iapRegistry.getIndexForTla(env, child.uri.slice(1))
            )
        )
        .then(apps =>
            Promise.map(apps, ({ tla, version }) =>
```

```
                $iapRegistry.getManifestContent(env, tla, version)
            )
        )
        .catch(error => new ArtifactoryError(error));

/**
 * Fetches a specific tla's manifest
 * @param {Object} root - root configuration
 * @param {Object} args - query arguments
 * @param {String} [args.tla] - application tla
 * @param {Object} context - graphql context
 * @returns {Promise<Object>} promise resolved with the manifest object for
a given tla
 */
const getAppManifest = (root, args, { $iapRegistry }) =>
    $iapRegistry
        .getIndexForTla(env, args.tla)
        .then(({ tla, version }) =>
            $iapRegistry.getManifestContent(env, tla, version)
        )
        .catch(error => new ArtifactoryError(error));

/**
 * Create a manifest for a specific tla
 * @param {Object} root - root configuration
 * @param {Object} args - query arguments
 * @param {Object} [args.app] - application manifest object
 * @param {Object} context - graphql context
 * @returns {Promise<Object>} promise resolved with the manifest object
 */
const createAppManifest = (root, { app }, { $iapRegistry }) =>
    $iapRegistry
        .createAppManifest(env, app.tla, app)
        .then(() =>
            $iapRegistry.updateAppIndex(
                env,
                app.tla,
                `manifest-${app.buildNumber}.json`
            )
        )
        .then(() => app)
        .catch(error => new ArtifactoryError(error));

module.exports = {
    Query: {
        getAppsManifests,
        getAppManifest
    },
    Mutation: {
        createAppManifest
    }
};
```

Code 13 – *resolvers.js* file content of the registry strand

$iapRegistry is a service provided by Fabric that offers an API to communicate with the Artifactory IAP folders and respective content. Fabric uses a dependency injection mechanism that allows the usage of this service in every strand. Besides the two queries and one

mutation, at the end of the code, it is possible to be that an object containing queries and mutations is exported. This object is loaded by GraphQL so that the created Queries and Mutations are available.

Another core piece of code is the strand initialization file, where the endpoint verb, route, schema and entry point function is defined. In the entry point function, GraphQL is initialised using the defined types (not shown due to their dimension) and resolvers (queries and mutations seen in Code 13). This file can be seen in Code 14.

```
'use strict';

const { graphql } = require('graphql');
const { makeExecutableSchema } = require('graphql-tools');
const joi = require('joi');
const apps = require('./apps');

/**
 * Handle any requests that hit this strand
 * @param {Object} $params - parameters passed to the request
 * @param {Object} $iapRegistry - fabric client to communicate with iap's
artifactory registry
 * @returns {Object} graphql instance
 */
function handle($params, $iapRegistry) {
    const schema = makeExecutableSchema({
        typeDefs: apps.schema,
        resolvers: apps.resolvers
    });

    const context = { $iapRegistry };

    return graphql(
        schema,
        $params.query,
        null,
        context,
        $params.variables,
        $params.operationName
    );
}

module.exports = {
    verb: 'POST',
    route: '/graphql',
    schema: {
        query: joi.string().required(),
        variables: joi.object().optional(),
        operationName: joi.string().optional()
    },
    fn: handle
};
```

Code 14 – Registry strand *index.js*

Looking at the tests, this strand uses Jest, a unit test framework, to build and run its unit tests. Code 15 shows a sample of some implemented tests.

```
jest.mock('graphql');
jest.mock('graphql-tools', () => ({
    makeExecutableSchema: jest.fn(() => 'EXECUTABLE-SCHEMA')
}));
jest.mock('../apps', () => ({
    schema: 'APP-SCHEMA',
    resolvers: 'APP-RESOLVERS'
}));

const joi = require('joi');
const victim = require('../index');
const graphql = require('graphql');

const graphqlTools = require('graphql-tools');

describe('index', () => {
    const $iapRegistry = {};
    const $params = {
        query: 'QUERY',
        variables: 'VARIABLES',
        operationName: 'OPERATION-NAME'
    };
    it('should have POST verb', () => {
        expect(victim.verb).toEqual('POST');
    });
    it('should have /graphql route', () => {
        expect(victim.route).toEqual('/graphql');
    });
    it('should have a schema', () => {
        expect(victim.schema).toEqual({
            query: joi.string().required(),
            variables: joi.object().optional(),
            operationName: joi.string().optional()
        });
    });
}
```

Code 15 – Registry Strand Unit Test Code Sample

7.1.3 Template Strand

As seen in Figure 2, a Template Strand already existed in FCN. The IAP Template Strand has the same responsibilities: (i) fetching the applications manifest; (ii) and building the HTML. The main difference between them is where they fetch the manifests and the templates available. Regarding the available templates, they will be further discussed in subchapter 7.3, since they were thought as a way to provide a solution agnostic from any technology.

As in section 7.1.2, one of the core pieces of code is the strand initialisation file. The same can be found in Code 16.

```
'use strict';
const Promise = require('bluebird');
const joi = require('fabric-input').validator.joi;
const createLayoutBuilder = require('iap-layout-builder');
const { IAP_CONFIG_PATH = './config/config' } = process.env;
const config = require(IAP_CONFIG_PATH);
/**
 * @param {Object} $params - get http request params
 * @param {Object} $iapRegistry - fabric's iapRegistry client
 * @param {Object} $hogan - hogan client
 * @returns {String} html page
 */
function handle($params, $iapRegistry, $hogan) {
    const render = createLayoutBuilder($hogan, config);
    /**
     * Get all apps selected manifest
     * @returns {Promise<Manifests>} manifests
     */
    const getAllManifests = () =>
        $iapRegistry
            .getIndex(config.IAP.env)
            .then(({ children }) =>
                Promise.map(children, child =>
                    $iapRegistry.getIndexForTla(
                        config.IAP.env,
                        child.uri.slice(1)
                    )
                )
            )
            .then(apps =>
                Promise.map(apps, ({ tla, version }) =>
                    $iapRegistry.getManifestContent(
                        config.IAP.env,
                        tla,
                        version
                    )
                )
            );
    return getAllManifests().then(manifests => {
        const tla = $params.tla || config.IAP.defaultTla;
        return render(manifests, tla);
    });
}
/**
 * Export the behavior using the Fabric component interface
 */
module.exports = {
    fn: handle,
    mime: 'text/html',
    route: '/:tla?',
    schema: {
        tla: joi
            .alternatives()
            .try(joi.string().optional(), joi.number().optional())
    }
};
```

Code 16 - Template strand *index.js*

The structure is in all similar to the Registry Strand, which is one of the advantages of using Fabric and Strands, it provides a homogenous development process. The entry point of this strand also uses the *$iapRegistry* service to retrieve the selected manifest of every application. This is needed because besides generating the HTML for the selected application, it also needs every application metadata so they can be displayed in an IAP Menu for a user to navigate between them.

The *iap-layout-builder* has the templates and the logic to generate the HTML but as stated before, it will only be approached in subchapter 7.3

Regarding unit tests, it follows the same patterns and technologies of the Registry Strand so they will be left out of this chapter since it would not bring any value to this point.

7.1.4 Application Registration

In order for an Application to register or update itself in IAP, as seen in Figure 53, its deployment process must be changed to contemplate that. For that, two changes are required: (i) a script must be created that abstracts the registration for every application; (ii) and the pipelines must be updated to call that script.

First, the applications must have a manifest file available in the root of their project. The manifest will end up stored in Artifactory as seen in Code 12 but it must have a different structure in the project since the build number is generated only in the pipelines, and the files will be stored in different locations for each environment. Code 17 shows an example of an application manifest.

```
{
  "name": "dfi-dashboard",
  "fullname": "IAP Dashboard",
  "description": "DFI dashboard",
  "tla": "dfi",
  "module": "dfi.dashboard",
  "hidden": true,
  "assets": {
    "scripts": [{
      "src":
"https://@@TLA.@@ENVIRONMENT.internal/scripts/environment.@@BUILD.js"
      },{
      "src":
"https://@@TLA.@@ENVIRONMENT.internal/scripts/dashboard.@@BUILD.js"
      }],
    "styles": [{
      "rel": "stylesheet",
      "href":
"https://@@TLA.@@ENVIRONMENT.internal/styles/dashboard.@@BUILD.css"
     }]
   }
}
```

Code 17 – Application *manifest.json*

The structure is in all similar to the one seen in Code 12. However, some differences can be noticed: (i) the lack of build number; (ii) and the tags *@@TLA, @@ENVIRONMENT and @@BUILD.* The files will be stored in different locations regarding the application TLA and environment and will have different names according to the build number. By creating these tags and transforming the manifest file into a template, it is possible that when the pipeline of each environment runs, it replaces the tags and generates a manifest file accordingly. This approach abstracts the applications owners of a lot of work and with only one manifest all the manifests for all the environments can be correctly generated.

The script, in the Annexe B.1, does the whitelisting of the properties that can be present in the final manifest. It receives the application TLA, environment, build number and the manifest content itself. Then, it replaces the tags with the correct values and calls the Registry Strand for the respective environment. For instance, if the script is running in the QA environment, it will point to the Registry Strand running in QA and so on.

Regarding the unit tests (c.f. NFR35), it follows the same technologies used in the previous one. Code 18 shows a sample of some unit tests.

```
const path = require('path');
const fs = require('fs');
const mockApolloFetchSuccess = jest.fn().mockResolvedValue('DUMMY-RESULT');
const mockApolloFetchFail = jest.fn().mockRejectedValue(new Error('DUMMY-
ERR'));
const mockApolloFetchSuccessWithErrors = jest
  .fn()
  .mockResolvedValue({ errors: [{ message: 'error' }] });
const expectedManifest = {…};
const mocks = {};

describe('Manifest File Generated Correctly', () => {
  beforeEach(() => {
      process.cwd = jest.fn().mockReturnValue(path.join(__dirname,
'mocks'));
      process.exit = jest.fn();
      mocks.joinSpy = jest.spyOn(path, 'join');
  });
  afterEach(() => {
    mocks.joinSpy.mockReset();
   mocks.joinSpy.mockRestore();
    jest.resetModules();
  });

  describe('When manifest file is passed as a param', () => {
    describe('When manifest file exists', () => {
      describe('When it succeedes', () => {
        let iapManifest;
          beforeEach(() => {
            jest.mock('apollo-fetch', () => ({
              createApolloFetch: createApolloFetch =>
                mockApolloFetchSuccess
            }));
            iapManifest = require('../lib/iap-manifest');
```

```
            return iapManifest.publishManifest(
              'tst',
              'dev',
              21,
              '../mocks/manifest.json'
            )
            .then(result => {
              mocks.result = result;
            });
          });

        it('should register application using manifest file', () => {
          const query ='mutation CreateApp($app: AppManifestInput!) ' +
            '{createAppManifest(app: $app) {tla}}';
          const variables = {
            app: Object.assign({}, expectedManifest, {
              buildNumber: 21
            }
          )};
          expect(mockApolloFetchSuccess).toHaveBeenCalledWith({
            query,
            variables
          });
          expect(mocks.result).toEqual('DUMMY-RESULT');
        });
      });
    });
  });
});
```

Code 18 – Application Registration Script Unit Test Sample

To complete the process it is necessary to change the pipelines or them to call this script. As seen in subchapter 2.4, Infrastructure is built upon OpenStack [68] (c.f. NFR19 to NFR21) and one tool available under the OpenStack is the Jenkins Job Builder that with simple YAML and JSON files, is capable of configuring Jenkins Jobs [93].

In order to provide a simple way for the applications to call the script, a template job [94] was created so it could be shared and used by every IAP application. In Code 19 it is possible to see the template job implementation.

```
- job-template:
    name: '{tla}_register_application'
    description: 'IAP {tla} Application Registration'
    branch: 'origin/master'
    project-type: freestyle
    logrotate:
      numToKeep: 20
      daysToKeep: 30
    wrappers:
      - timestamps
      - ansicolor
      - workspace-cleanup
      - nodejs-installator:
```

```
            name: 'v6.7.0'
        parameters:
          - string:
              name: upstream_build
              description: 'A parameter with the upstream build url'
          - string:
              name: environment
              description: 'A parameter with the environment'
        scm:
          - clone_gitlab_project_branch_ignore_notify:
              project_path: '{project_path}'
              branch: '{branch}'
        builders:
         - shell: |
          export build_number=$(echo ${{upstream_build}} | grep -oP '[0-
    9]{{1,}}')

          npm install
          npm update
          npm run iap:manifest -- -t {tla} -e ${{environment}} -b $build_number
```

Code 19 – IAP Registration Template Job

The applications that wish to use IAP must use this job in their pipelines. This job receives as parameters the upstream build URL, which is the URL containing the build of the application that is calling this job, and the environment where this job will run. After that, it clones the project that is calling the job and runs the commands that are in the shell property inside builders. This will call the script in the Annexe B.1, registering, therefore, the manifest in the Artifactory.

Jenkins Job Builder already comes with a suite of tests that runs for every file in the project, so no additional tests had to be implemented for this development.

This was the biggest piece of work of all the raised problems. However, it was also the one that achieved that made possible to overcome one of the biggest problems for the FCN users: being able to control their releases and deployment and, therefore, control their own development cycle (c.f. NFR7). Moreover, applications are now released and deployed into their own clusters (c.f. NFR39).

7.2 Complex Theme Scheme

For the complex theme scheme problem, the solution designed end up being a mixture of several approaches as illustrated by Figure 57. In this subchapter, an example of each approach implementation is shown. The presented implementation is set to achieve the NFR10 by simplifying the complex theme scheme.

For the case of the frameworks' components, no work from the IAP maintainers is expected. It is the applications' teams full responsibility for finding the best components for each

framework and to find one that they feel comfortable with. However, there are some technologies that have a wide use around the industry such as Material UI [71] and React-Bootstrap [95] for React, Ng-Bootstrap [96] and Angular Material for Angular [97], Bootstrap [73] for components not attached to a particular framework.

For more complex components, web components were implemented using StencilJS. By using StencilJS, we are assuring that the developed components are supported not only in the latest versions of Chrome (c.f. NFR36) but also in other browsers. That is because StencilJS comes with a set of polyfills, as discussed in section 3.3.3. To demonstrate the implementation, the header component, that represents the header that is presented in IAP, is used as an example. The implementation can be seen in Annexe C.1 while the styles for such implementation can be seen in Annexe C.2.

As it can be seen, this component is a complex one and it has a lot of logic and other components within it. It has the logic to redirect to other applications, it handles and triggers events that can be used by other applications to perform other operations and it also knows when to render the IAP menu, another component that is not going to be approached here but follows the same structure as this one.

Regarding unit tests (c.f. NFR35), a sample can be found in Code 20.

```
import { TestWindow } from '@stencil/core/testing';
import { IapHeader } from './iap-header';
describe('iap-header', () => {
    it('should build', () => {
        expect(new IapHeader()).toBeTruthy();
    });
    describe('rendering', () => {
        let element: HTMLIapHeaderElement;
        let testWindow: TestWindow;
        beforeEach(async () => {
            testWindow = new TestWindow();
            element = await testWindow.load({
                components: [IapHeader],
                html: '<iap-header> <div class="mock"><div> </iap-header>'
            });
        });
        it('should work without parameters', () => {
            expect(element.textContent.trim()).toEqual(
                'Internal Applications Platform'
            );
        });
        it('should initialize the menu as not visible', () => {
            const document = testWindow.document;
            const slot = document.getElementsByTagName('slot')[0] || null;
            expect(element.contains(slot)).toEqual(false);
        });
        it('should show slot', async () => {
            expect(
                element.querySelector('nav').innerHTML.includes('mock')
            ).toEqual(false);
            element.querySelector('button').click();
            await testWindow.flush();
```

```
            expect(
                element.querySelector('nav').innerHTML.includes('mock')
            ).toEqual(true);
        });
    });
});
```

Code 20 – IAP Menu Component Unit Test

It also uses Jest, but it is encapsulated under a StencilJS library [98] that adds other functionalities like, for instances, mocked browser variables, such as window and document, that make unit testing visual components easier.

Every Web Component developed with StencilJS follows this structure: (i) one *TSX* file, which is similar to JSX, with the only difference of using Typescript instead of Javascript; (ii) one style file, that can use other formats as SASS but will end up transpiled as CSS (iii) and one unit test file. This makes the development and maintenance of the created web components simpler and easier since the development experience between all of them are the same.

Last, IAP can also provide pure CSS components. At the time of this writing, no CSS component was yet developed since the need to do so has yet no arisen. However, to demonstrate how much is planned to be implemented, let's use Code 9 as an example, and Code 21 represents one potential implementation of the styles of such components.

```
.button {
 display: inline-block;
 border-radius: 3px;
 padding: 7px 12px;
 border: 1px solid #D5D5D5;
 background-image: linear-gradient(#EEE, #DDD);
 font: 700 13px/18px Helvetica, arial;
}
.button--state-success {
 color: #FFF;
 background: #569E3D linear-gradient(#79D858, #569E3D) repeat-x;
 border-color: #4A993E;
}
.button--state-danger {
 color: #900;
}
```

Code 21 – Potential CSS Component Styles Implementation [75]

This example is written in CSS. However, the components can be written in whatever CSS Framework the team desires or feels comfortable with. The mandatory is that once the components are published for global usage, they are transpiled into pure CSS files. This way, not only the developer experience of the IAP maintainers is enhanced, but the teams that use these components will end up with pure CSS files that do not force them to use any particular technology. CSS components can be extended by other components. This way, it is possible to keep a consistent theme across applications. (c.f. NFR10).

These three implementations ease one of the most complex development pieces of the FCN. In IAP, developing themes, styles and UI components is easier and gives applications' and IAP's developers much more freedom.

7.3 Technological Dependencies

As demonstrated in Code 4, the template used in the Template Strand of the FCN uses angular to load the applications and several modules that the apps depend on, therefore forcing that every application uses the same Angular version as the one that is used to load them. In section 6.3.1, the selected approach consisted in adding a field to the applications manifest file where the desired template could be chosen. In section 7.1.3 the template strand was already approached but this subchapter gets into detail on how the template strand contributes to a technological agnostic solution.

iap-layout-builder component has the templates and the logic to generate the HTML. First, it must parse the data in the manifest to understand what template is going to be used and what data is going to be used by the template. That logic can be found in Code 22, that is part of the *iap-layout-builder*.

```
/**
 * Renders the selected app
 * @returns {string} html
 */
const renderAppPage = () => {
  const iapConfig = Object.assign({}, getConfig(tla, config), {
    trackers,
    apps: appsRegistry
  });

  const templateParams = Object.assign(
    {
      coreBuildNumber: buildNumber,
      active: tla,
      apps: stringifyApps(manifests),
      fcn: appsRegistry,
      fcnBuildNumber: process.env.FCN_BUILD_NUMBER||config.FCN.buildNumber,
      headerLabel: selectedApp.header
        ? selectedApp.header.value
        : 'Internal Applications Platform',
      hasIapHeader:
        selectedApp.hasIapHeader === undefined ||
        selectedApp.hasIapHeader,
      iapConfig: JSON.stringify(iapConfig),
      isTestEnvironment: iapConfig.IS_TEST_ENVIRONMENT,
      hasGoogleAnalytics: trackers.some(
        tracker => tracker.type === 'GOOGLE_ANALYTICS'
      )
    },
    selectedApp
  );

  return $hogan.render(
```

```
    selectedApp.template || 'legacy',
    templateParams
  );
};
```

Code 22 – Template Strand Code Sample

At the moment of this writing, there are two templates: (i) legacy, in Annexe D.1, that is responsible for keeping the retro-compatibility between the FCN applications so they can be moved to IAP without changing their structure; (ii) and agnostic, in Annexe D.2, that does not force the use of any technology, it just injects the desired files and styles into the HTML. By default, the legacy one is rendered. For the agnostic one to be used, the template field in the manifest file should explicitly have the value agnostic.

In the legacy template, the logic seen in Code 4 is kept. This way, when migrating an application from FCN to IAP, besides changing its infrastructure to be deployed to its own machines and registering itself in IAP, no further development is needed. This assures that no changes to the application code are needed which is a big plus for applications that have no longer an active development.

Looking at the agnostic template, this one is a lot simpler and flexible than the legacy one. Not only it does not have any angular or other framework associated with the application loading, but it also allows customization. Applications can define if they wish to have tracking mechanisms (at the moment only Google Analytics is supported) or if they want to use the IAP Header or not. Applications are now responsible to inject the desired files, from scripts to styles. If the application is developed using a specific framework, that framework should be inside the application bundle or the application should add the URL of the framework script to the manifest. Either way, it is the application full responsibility, IAP only provides the mechanism to do so.

Regarding tests, there are unit tests (c.f. NFR35) that cover the logic that feeds the data to the template. However, there are no tests to validate the final *index.html*. Code 23 presents a sample

```
const createVictim = require('../index');

describe('index', () => {
  let $hogan;
  let baseConfig;

  beforeEach(() => {
    baseConfig = {
      IAP: {
        env: 'TEST',
        loggingEndpoint: 'LOGGING-ENDPOINT',
        gaKey: 'the ga key'
      },
      FCN: {
        buildNumber: 31
      }
    };
```

```
    $hogan = {
      render: jest.fn().mockReturnValue('HOGAN-RENDER-RESPONSE')
    };
  });

  describe('when selected app has no template defined', () => {
    let result;

    beforeEach(() => {
      const victim = createVictim($hogan, baseConfig);
      const manifests = [{ tla: 'a' }, { tla: 'b' }];

      result = victim(manifests, 'b');
    });

    it('should return hogan render response', () => {
      expect(result).toEqual('HOGAN-RENDER-RESPONSE');
    });

    it('should call $hogan.render for the legacy template', () => {
      expect($hogan.render).toHaveBeenCalledWith('legacy', {
        active: 'b',
        apps: JSON.stringify(['a', 'b']),
        coreBuildNumber: 'TAG',
        fcn: JSON.stringify({ a: { tla: 'a' }, b: { tla: 'b' } }),
        fcnBuildNumber: 31,
        hasIapHeader: true,
        headerLabel: 'Internal Applications Platform',
        iapConfig: JSON.stringify({
          IS_TEST_ENVIRONMENT: true,
          LOGGING_ENDPOINT: 'LOGGING-ENDPOINT',
          ENV: 'TEST',
          TLA: 'b',
          GA_KEY: 'the ga key',
          trackers: [],
          apps: JSON.stringify({ a: { tla: 'a' }, b: { tla: 'b' } })
        }),
        isTestEnvironment: true,
        tla: 'b',
        hasGoogleAnalytics: false
      });
    });
  });
});;
```

Code 23 – *iap-layout-builder* unit test sample

By providing a way for the applications to leverage on IAP but at the same time allowing them to use whatever technologies (c.f. NFR11) they desire gives more freedom to applications and can make IAP a time-proof framework and platform.

7.4 Documentation

In subchapter 6.4, there were three approaches that were created to overcome this problem: (i) READMEs and Wiki pages on Gitlab; (ii) Confluence pages; (iii) and a dedicated website.

However, when doing the implementation of the approaches, the group came to the conclusion that having READMEs and Wiki pages in Gitlab would make the information harder to reach. Since IAP's architecture is based in microservices and micro frontends, there are dozens of projects and the information would be too scattered around Gitlab. With that in mind, the team decided that a confluence page would be created and it would contain high-level information about IAP: (i) architecture (c.f. NFR14); (ii) releases; (iii) SLA approval form; (iv) environments; (v) spikes; (vi) team information amongst others. Figure 65 shows the homepage for the IAP confluence page.

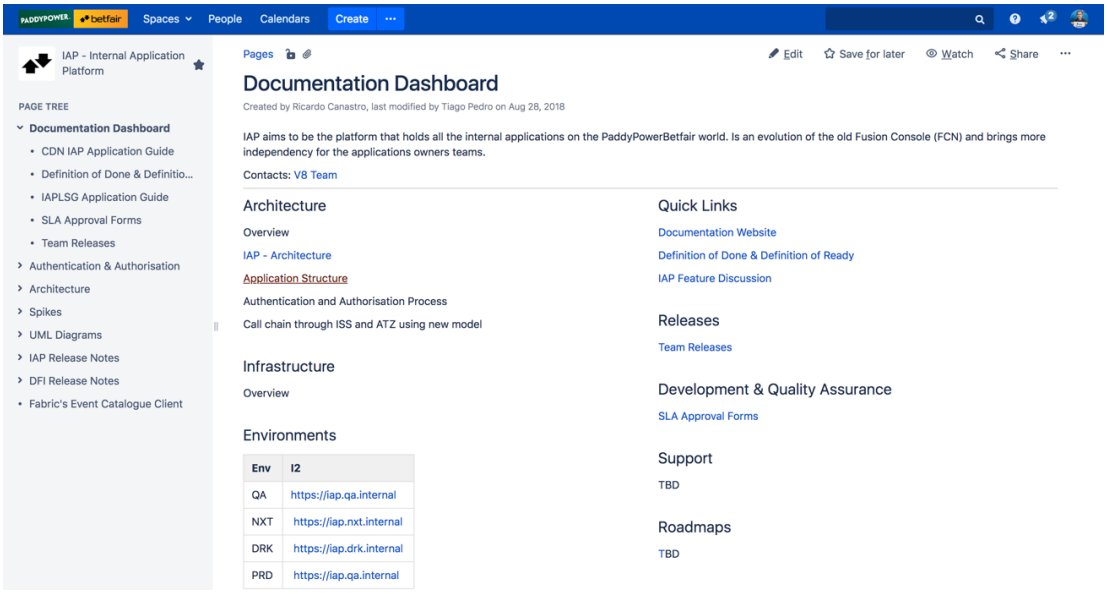


Figure 65 – IAP Confluence Page

This page lets applications' teams and stakeholders keep track of what is being built, releases that have been made and what is being discussed without getting into too much detail or code level information.

For the dedicated website, Docusaurus [99] was used. Docusaurus is a tool that allows the creation of a website where documentation can be published without any concern about the webpage design or infrastructure. The documentation must be provided via markdown files and Docusaurus provides default styles, site formatting, documentation navigation, blog support, internationalization, search, and versioning. All this can be customized through some configurations or simple code changes [100].

This website is divided into two major sections: (i) a User Guide, where documentation can be found for configuring the applications to use IAP and how to use the different services and

components that IAP provides (c.f. NFR13 and NFR15); (ii) and a developer guide, which contains the rules and recommendations for the developers that wish to contribute to IAP codebase; Figure 66 represents the website homepage that not only shows the two sections, but it also enumerates some IAP features and applications that already use IAP.

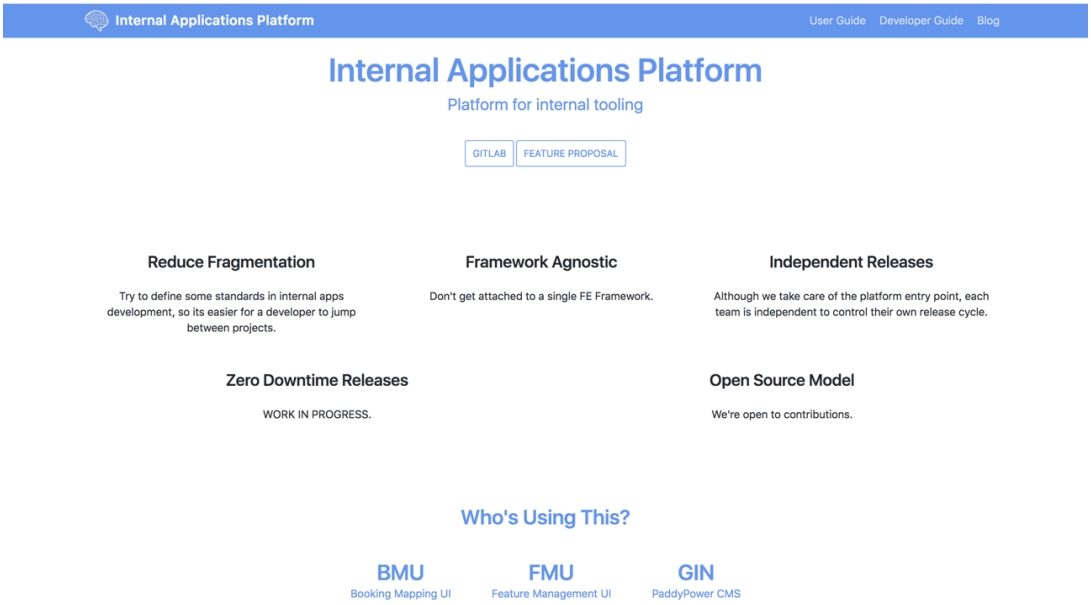


Figure 66 – IAP Documentation Website Homepage

Code 24 shows a simple documentation page built in markdown. Figure 67 represents the markdown after being loaded by Docusaurus.

```
---
id: creating-app
title: Creating App
---

Let's get started!

* [Install Tools](install-tools.html)
* [Configure metadata](metadata-package.html)
* [Install development environment](development-environment.html)
* [Create infrastructure](infrastructure-setup.html)
* [Configure CDN](cdn-setup.html)

Now you have an IAP's compatible application. You can start to develop,
test and leverage of the IAPs ecosystem. Make sure you check our UI/UX
guidelines, our Frontend Providers and our Functional Test tools.
```

Code 24 – Creating App Documentation Markdown

Figure 67 represents the markdown after being loaded by Docusaurus.

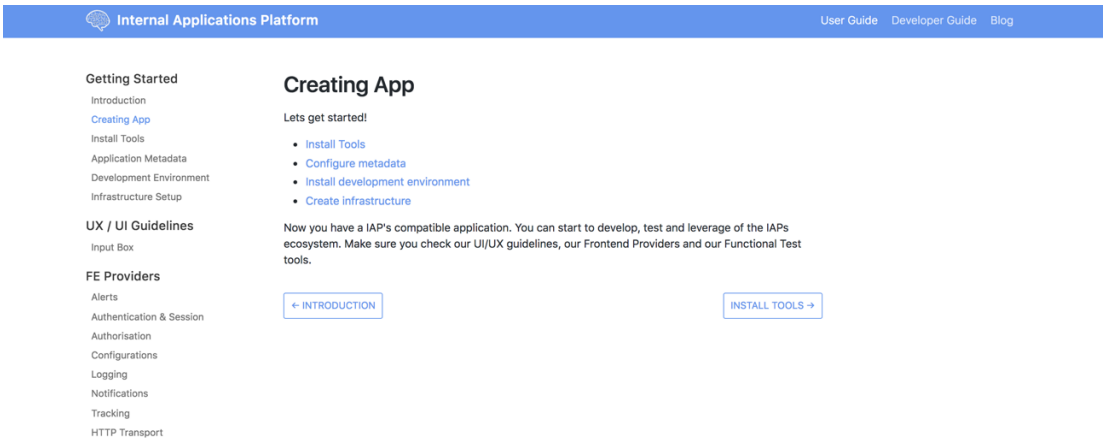


Figure 67 – Creating App Documentation Page

It is important to notice that the header is the same as the one in the home page which makes the navigation between pages more fluid and the left menu makes it easier to navigate between documentation pages.

Having a Confluence page that follows the company standard of keeping Confluence pages for more generic information and a dedicated website that centralizes all the low-level details about IAP, documentation is now easier to maintain and to reach. However, the impact on IAP users must be evaluated.

7.5 Logging

In subchapter 6.5 logging was divided into Frontend Logging and Backend Logging (c.f. NFR9). In this subchapter, only the Frontend Logging implementation is deeply described since the Backend logging will use the same mechanisms.

As seen in Figure 60, there are two IAP players working to provide Frontend Logging: (i) a IAP Logging Client, that the applications use to generate the logs and send to a backend service; (ii) and IAP Client Logging Service, service that is going to receive the logs generated by the apps and stored them.

The IAP Logging Client consists in a Javascript library that applications can import and use it to generate logs. It provides a factory, Annexe E.1, that applications can use to generate several loggers. All these loggers share the same client, Annexe E.2, (that is responsible for sending the logs to the backend service) but are different from each other, which allows applications to have multiple loggers per application. That can be a benefit if, for instance, an application wants to create a logger for each kind of action that a user can make.

Looking at the logger, it offers six levels of logs, each one with incremental value, stating in 0: (i) trace, value 0; (ii) debug, value 1; (iii) info, value 2; (iv) warn, value 3; (v) error, value 4; (vi) and silent, value 5; It is not possible to use a log level that is higher than the current one.

These levels can be used to filter the logs when searching in Splunk. Code 25 shows the function responsible for generating the warning messages. All the other levels are similar with the exception of the level silent, that does not allow any log to be created.

```
/**
 * Warn message
 * @param {String} message - message to be logged
 * @returns {Promise} http promise
*/
warn(message) {
    if (this.level > LEVELS.WARN.value) return;

    if (this.shouldOutputToConsole) {
        console.warn(`${this.name}:: ${message}`);
    }

    return this.client.send(
        getMessage({
            name: this.name,
            tla: this.tla,
            level: LEVELS.WARN.label,
            message
        })
    );
}
```

Code 25 – Logger Function responsible for generating warning messages

Jest was once again used. Code 26 shows a sample of the unit tests (c.f. NFR35) created for this project.

```
import LoggerClient from '../client';

describe('#send()', () => {
    test('should call window.fetch', () => {
        window.fetch = jest.fn();

        const target = new LoggerClient('http://dummy');
        target.send({ message: 'DUMMY-ENTRY' });

        expect(window.fetch.mock.calls[0][0]).toEqual('http://dummy');
        expect(window.fetch.mock.calls[0][1]).toEqual({
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({ entries: [{ message: 'DUMMY-ENTRY' }] })
        });
    });
});
```

Code 26 – *client.spec.js* sample unit test

The IAP Client Logging Service was also implemented as a strand, leveraging on the logging service that fabric also provides via dependency injection, already approached in section 6.5.2. The Strand initialisation file is in all similar to the previous strands, changing only the entry point function logic as demonstrated in Code 27.

```
/**
 * Iterates all entries and logs them
 * @param {Object} $params - request params
 * @param {Object} $log - log provider
 * @returns {Promise} Promise
 */
function main($params, $log) {
    return Promise.map(
        $params.entries,
        entry =>
            new Promise(resolve => {
                $log[entry.level.toLowerCase()](
                    util.format(
                        'TLA="%s", LOGGER="%s", URL="%s", USER:"%s",
MESSAGE="%s", DETAILS=%j',
                        entry.tla,
                        entry.name,
                        entry.url,
                        entry.username,
                        entry.message,
                        entry
                    )
                );

                return resolve(
                    util.format(
                        'IAP-CLIENT-LOGGER: logged %s::%s',
                        entry.level,
                        entry.tla
                    )
                );
            })
    );
}
```

Code 27 – Client Logging Strand Entry Point Function

$log is the service that Fabric provides to handle logs. This strand leverages on that service to store the logs that it receives the applications. This is the service that is also used in all the Strands that IAP provides. Authentication, Register and Template Strands only need to inject the *$log* service in their entry point function, and logging is available out of the box.

By leveraging on Fabric already available services, the logging implementation, both in Frontend and Backend was eased (c.f. NFR23 to NFR34 and NFR38). It is now the applications full responsibility to use the provided mechanisms to leverage on the advantages of having good logging in their applications.

7.6 Conventions

In subchapter 6.6, the solution developed for the lack of conventions was to revisit the *eslint-config-fcn* and to create a new one with updated configurations. As also stated in that

subchapter, this does not entirely fix the problem, but by updating it and showing how to use it in their applications, it may be possible to spread its usage. The final result can be seen in

```
module.exports = {
  "extends": [
    "eslint:recommended"
  ],
  "parserOptions": {
    "sourceType": "module"
  },
  "rules": {
    "arrow-parens": ["error", "as-needed"],
    "comma-dangle": ["error", "never"],
    "indent": ["error", 4, { "MemberExpression": 1, "SwitchCase": 1 }],
    "key-spacing": ["error", { "mode": "strict" }],
    "eol-last": ["error", "always"],
    "linebreak-style": ["error", "unix"],
    "max-len": ["warn", 100, 4],
    "no-cond-assign": ["error", "always"],
    "no-console": ["warn"],
    "no-debugger": ["warn"],
    "no-duplicate-imports": ["error"],
    "no-multiple-empty-lines": ["error", { "max": 1 }],
    "no-plusplus": ["error", { "allowForLoopAfterthoughts": true }],
    "quotes": ["error", "single"],
    "semi": ["error", "always"],
    "space-unary-ops": [2, {
      "words": true,
      "nonwords": false,
      "overrides": { "new": false }
    }]
  }
};
```

Code 28 – IAP Eslint Configurations

A guide is also provided. It explains the motivation behind providing IAP Eslint rules, how to use it and an example. Figure 68 shows that guide but it also mentions an *iap/jsdoc*. This is responsible for forcing the applications to have *jsdocs*, which consists in comments containing markup language that is used to describe functions, variables and the application API [101]. It was not mentioned before because the one used in IAP is an exact copy of the one used in the FCN.

iap recommended ESLint rules!

ESLint *extends* options

- `iap` : similar to `eslint:recommended` , but for iap!
- `iap/jsdoc` : for having `jsdoc` awareness! > This is highly recommended!

Usage

1. Create a `.eslintrc` on `root` path of the module;
2. Install as a `devDependency` by running `npm i -D eslint-config-iap` ;
3. Add the following line `{ "extends": ["iap"] }` ;
    - > `eslint-config-iap` already extends the `eslint:recommended` .
4. Extend or add as needed like adding `globals` and `env` or `plugins` .

Example

```
{
  "extends": [
    "iap",
    "iap/jsdoc"
  ],
  "globals": {
    "angular": true,
    "mocha": true
  },
  "env": {
    "es6": true,
    "browser": true
  }
}
```

Figure 68 – IAP Eslint Guide

Although not fixing the problem, updating and showing how to use these rules may help to standardize the way applications are built inside IAP, making the boilerplate and learning curve when a developer changes between applications lower.

7.7 Authentication and Authorisation

Unfortunately, due to the solution raised in subchapter 6.7 being dependent on third parties, the Authorisation has not been implemented yet. However, in this subchapter Authentication is approached.

The Authentication is made available through a Strand. There was already a Strand for FCN and for IAP a copy of the same was made since the services and the flow of the Authentication process have not changed. A sample of that strand can be found in Code 29.

```
/**
 * Aggregation function for authentication
 * @param {Object} $params - Object with params (domain, username, etc)
```

```
 * @param {Object} $identitySso - Object with params (domain, username, etc)
 * @param {Object} $adminIdentity - Object with params (domain, username,
etc)
 * @param {Object} $log - Object with params (domain, username, etc)
 * @return {Function} - returns
 */
function authAggregation($params, $identitySso, $adminIdentity, $log) {
    /**
     * ISS function to authenticate an operator
     * @param {String} domain - domain where to authenticate
     * @param {String} username - user that wants to authenticate
     * @param {String} password - passphrase
     * @return {Promise} - result of the operatorLogin action
     */
    const iss = (domain, username, password) => {
        $log.warn(msg(domain, username));
        return $identitySso.operatorLogin(domain, username, password, {
            loginContext: {}
        });
    };

    /**
     * AIS function to authenticate an operator
     * @param {String} username - user that wants to authenticate
     * @param {String} password - passphrase
     * @param {String} domain - domain where to authenticate
     * @return {Promise} - result of the operatorLogin action
     */
    const ais = (username, password, domain) => {
        $log.warn(msg(domain, username));
        return $adminIdentity.operatorLogin(username, password, {
            domain: domain
        });
    };

    switch ($params.service) {
        case 'isso':
        case 'iss':
        case 'sso':
        case 'identity-sso':
            return iss($params.domain, $params.username, $params.password);
        case 'admin-identity':
        case 'ais':
            return ais($params.username, $params.password, $params.domain);
        default:
            return ais($params.username, $params.password, $params.domain);
    }
}
```

Code 29 – IAP Authentication Strand Entry Point Function

It is possible to see that this Strand also takes leverage on the *$log* service injected by Fabric, but it also makes use of other two services (c.f. NFR2 to NFR6) to perform the user authentication, having in consideration what was the chosen service. These services are the (i) *$identitySso*; (ii) and *$adminIdentity;* This is one of the major advantages of using Fabric to

build microservices. A lot of clients to consume other services are already provided by dependency injection.

The major change in the Authentication Strand between FCN and IAP is the unit tests framework. In the FCN, the unit tests were implemented using Mocha, Sinon and Chai while in IAP they follow the same technologies and structure as seen in Code 15.

Authorisation was not implemented and, therefore, NFR8 was not delivered. Due to third party dependencies, in this particular case the Admin Identity Service, it was not possible to provide a way of having Authorisation in IAP and its applications. However, some mechanisms have already been thought.

7.8 Lack of User Interface Consistency

Since this new problem/requirement (c.f. NFR17) was only raised half-way the progress of this thesis it was not expected that there would be any implementation. However, there is work already being done.

The new dashboard is being built in React since it is also an application. This also helps to demonstrate that IAP is already framework agnostic and supports more than just Angular. Figure 69 shows how the current state of the dashboard.

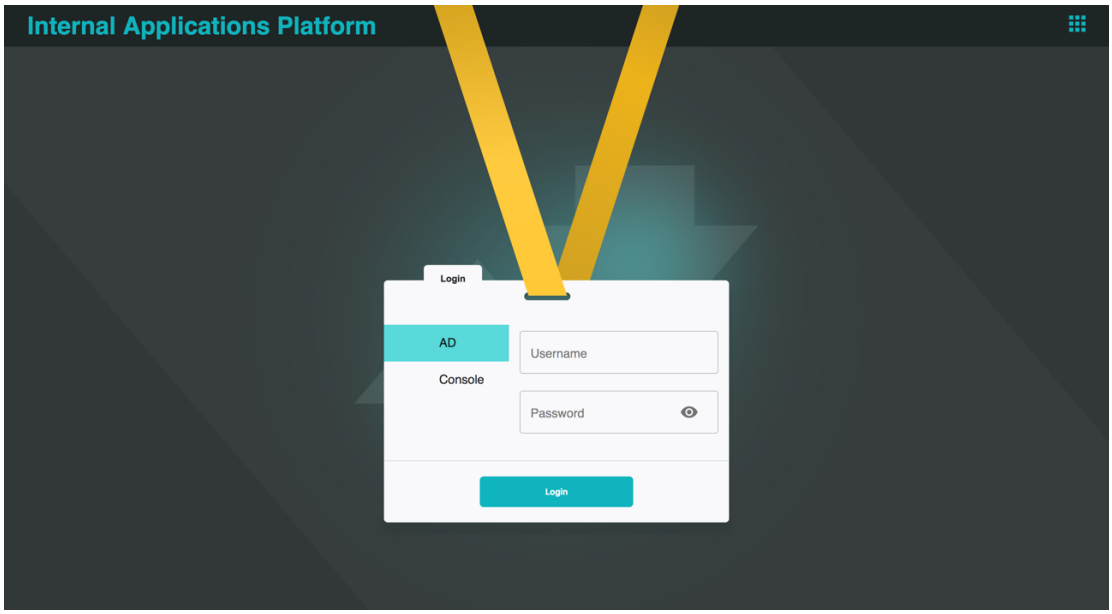


Figure 69 – IAP Dashboard, work in progress.

One thing that clearly jumps to the eye is the yellow stripes. This happens because the UI team has not yet provided the respective SVG files. Besides that, there are some changes to the design seen in Annexe G.1 that were approved.

The header, side menu, and the other components present in Figure 70 are being built using StencilJS since they must be shared across applications regardless of the technologies being used.

Figure 70 – Header, Side Menu, User Menu and Apps Menu in a sample Application

The notifications' menu is still being developed. Both figures follow what was discussed in subchapter 7.2.

8 Evaluation

To evaluate the solution presented in this thesis, there is the need to define a set of metrics and methodologies to guide such evaluation. It is important to understand what needs to be evaluated and how should it be evaluated.

Since this project follows an iterative approach to solve the problems, it is inevitable that the evaluation must be divided and thought out for each one of the problems that whose solutions were implemented.

8.1 Hypotheses

The hypotheses to be tested should provide answers to the goals that this project purposes to solve. Each hypothesis is associated with a series of requirements (c.f. section 4.2.2) and respective metrics and procedures that were applied to evaluate it. At the end of each hypothesis, the results and respective analysis can be found.

8.1.1 The applications developed using the IAP framework are independently release and deployed

This alternative hypothesis is related with the NFR7, NFR19 to NFR21 and NFR38. To reject the null hypothesis (The applications developed using the IAP framework are not independently deployed and released) there is the need to evaluate the achievement of such non-functional requirements. Since the design (c.f. section 6.1.3) and implementation (c.f. section 7.1) already clarify that the solution was built by leveraging in the company's infrastructure and services (c.f. NFR38) there is no need to evaluate such requirements.

Nevertheless, an inspection of the deployment process of applications built using IAP is presented. There are currently two applications built upon IAP and it is possible to affirm that there no dependencies over the IAP pipeline.

Moreover, NFR19 to NFR21, related with (i) availability, (ii) scalability and (iii) reliability of the solution, is achieved through the company's infrastructure. It is also important to evaluate if the solution did not increase the needed resources (machines) when compared with FCN.

8.1.1.1 Metrics

For the independent releases (c.f. NFR7) the metric to be used is the existence of one or more applications being released and deployed independently from IAP.

Deployment and release durations are expected to decrease due to the implementation of independent releases. However, to evaluate that such is true, there is the need to measure the duration of these processes in both versions of the framework, and then, compare the results.

In order to understand if the machines used increased between (c.f. NFR22) FCN and IAP, the OpenStack Platform must be consulted, since it is where all resources related information: (i) machines, (ii) instances, (iii) volumes, (iv) images and others; is centralized and available.

8.1.1.2 Methodologies

To understand if the established hypothesis can be accepted or rejected, there is the need to apply a set of tests. P-Value is the selected approach [102].

Being both null and alternative hypotheses defined, the best suitable statistical test must be applied, assuming that the null hypothesis is true. The selected statistical tests are the Student's test for paired values and the one-sample t-test.

The Students T-test [102] is a statistical test that compares two averages and calculates if they are different from each other and how meaningful such difference is. Since the raised hypotheses are related to the problems that this project aims to solve, there is the need to understand the values before and after the solution has been in place. Such aspect demands that a paired T-Test is used since dependent samples are used.

8.1.1.3 Results and Analysis

While developing IAP, two other applications were created: (i) BMU – Booking Mapping UI; (ii) and FMU – Feature Management UI;

Both these applications were built using IAP and the implementation for the independent releases and different clusters per application. As it can be seen in Figure 71, FMU has its own pipelines including every environment, with no dependency whatsoever of IAP.

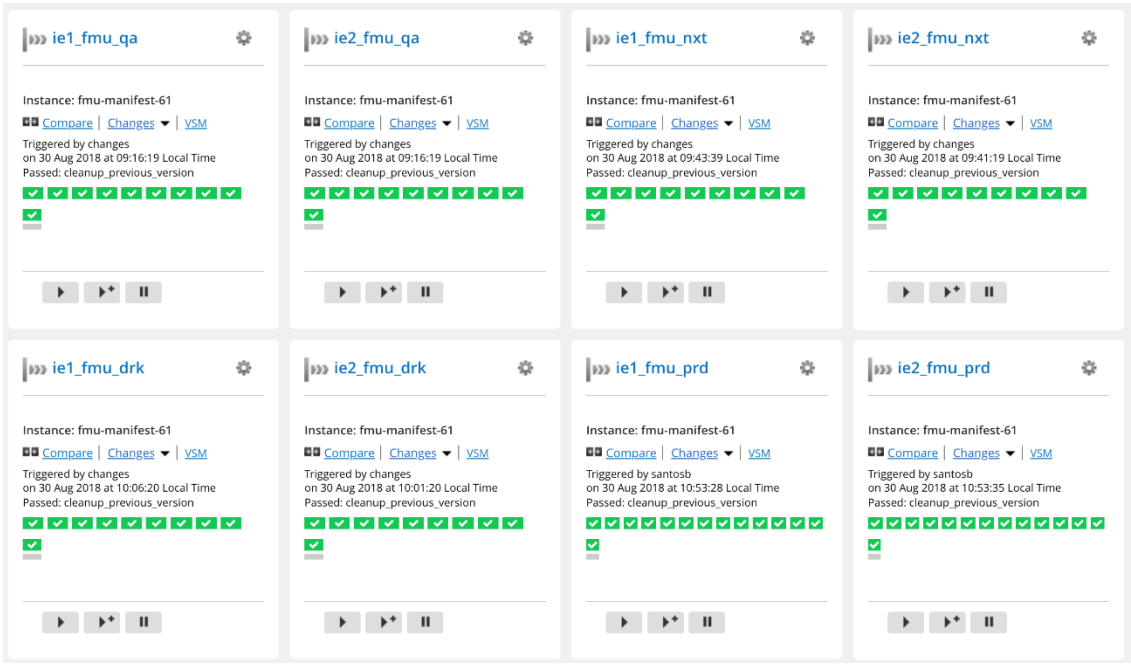


Figure 71 – FMU Pipelines

When looking at an application's pipelines developed in FCN, like IST in Figure 72, it is possible to see that the applications only have development environments.

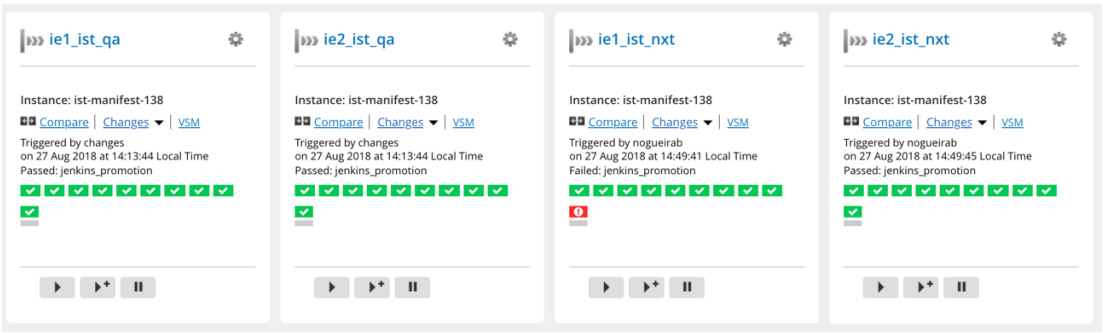


Figure 72 – IST Pipelines

Since it is possible to prove this hypothesis through these two applications, there is no need to perform a statistical test.

However, a statistical test to prove that the deployments and release time did not deteriorate from FCN to IAP can be performed.

The first step consists in collecting the time it takes for a build to go from QA (the first environment) until Production. In IAP, only the applications pipelines and environments are to be considered, while in FCN, the duration consists on the sum of the application's pipeline plus the FCN pipelines. Table 12 represents such times for FCN and for IAP's applications

where each entry represents the exact time it took for the applications to go from the first environment (QA) to the last one (Production).

Table 12 – FCN's IST and GIN Applications and IAP's FMU and BMU Applications Releases and Deployments Duration

FCN	IAP
143 minutes	105 minutes
161 minutes	101 minutes
172 minutes	111 minutes
151 minutes	106 minutes
162 minutes	108 minutes

At the time of this writing, no applications living in FCN have yet been migrated to IAP, meaning that a Paired T-Test cannot be applied. Instead, an Independent Samples T-Test is used using the following formula [103]:

$$t = \frac{\mu_{FCN} - \mu_{IAP}}{\sqrt{\left[\frac{\left(\sum FCN^2 - \frac{(\sum FCN)^2}{n_{FCN}}\right) + \left(\sum IAP^2 - \frac{(\sum IAP)^2}{n_{IAP}}\right)}{n_{FCN} - n_{IAP} - 2}\right] \cdot \left[\frac{1}{n_{FCN}} + \frac{1}{n_{IAP}}\right]}}$$

Where:

- μ_{FCN} – Mean of FCN's data set.
- μ_{IAP} – Mean of IAP's data set.
- $\sum FCN^2$ – Sum of the squares of FCN's data set.
- $\sum IAP^2$ – Sum of the squares of IAP's data set.
- $(\sum FCN)^2$ – Sum of FCN's data set, squared.
- $(\sum IAP)^2$ – Sum of IAP's data set, squared.
- n_{FCN} – Number of items in FCN's data set.
- n_{IAP} – Number of items in IAP's data set.

The *t-value* was then calculated by replacing the variables with its values:

$$t = \frac{157.8 - 106.2}{\sqrt{\left[\frac{494.8 + 54.8}{8}\right] \cdot \left[\frac{2}{5}\right]}} \Leftrightarrow t \approx 9.844$$

After the *t-value* calculation, there is the need to calculate the critical value. To do so, there was the need to calculate the Degree of Freedom ($n_{FCN} - 1 + n_{IAP} - 1 = 8$) and using an alpha level of 0.05, the critical value could be found in the Two Tails Distribution Table [104]. Being the critical value 2.306, and being the *t-value* higher than the critical value, by conventional criteria, this difference is considered to be extremely statistically significant. It is

possible to conclude, then, that not only there was no deterioration, as there was improvements in the durations between FCN and IAP.

Figure 73 and Figure 74 show the machines used for both FCN and IAP. Both machines have exactly the same configuration and specs.

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created
ie1-iap01b-qa.qa.betfair	centos-6-base-296	10.84.234.66	ie1_iap_qa_cd6534098a	-	Active	qa	None	Running	1 day, 2 hours
ie1-iap02b-prd.prd.betfair	centos-6-base-295	10.112.49.69	ie1_iap_prd_9cfd5982db	-	Active	prd	None	Running	2 weeks, 3 days
ie1-iap01b-prd.prd.betfair	centos-6-base-295	10.112.49.67	ie1_iap_prd_9cfd5982db	-	Active	prd	None	Running	2 weeks, 3 days
ie1-iap03b-prd.prd.betfair	centos-6-base-295	10.112.49.68	ie1_iap_prd_9cfd5982db	-	Active	prd	None	Running	2 weeks, 3 days
ie1-iap04b-prd.prd.betfair	centos-6-base-295	10.112.49.66	ie1_iap_prd_9cfd5982db	-	Active	prd	None	Running	2 weeks, 3 days
ie1-iap01b-drk.drk.betfair	centos-6-base-295	10.112.7.194	ie1_iap_drk_9cfd5982db	-	Active	prd	None	Running	3 weeks, 2 days
ie1-iap01b-nxt.nxt.betfair	centos-6-base-295	10.85.56.131	ie1_iap_nxt_3475e31831	-	Active	nxt	None	Running	3 weeks, 2 days
ie1-iap02b-nxt.nxt.betfair	centos-6-base-295	10.85.56.130	ie1_iap_nxt_3475e31831	-	Active	nxt	None	Running	3 weeks, 2 days

Figure 73 – IAP Machines from one of the Data Centers

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created
ie1-fcn01a-prf.prf.betfair	centos-6-base-296	10.85.141.130	ie1_fcn_prf_46b1789d5c	-	Active	prf	None	Running	1 day, 1 hour
ie1-fcn02b-nxt.nxt.betfair	centos-6-base-296	10.85.14.195	ie1_fcn_nxt_50c99d2cbe	-	Active	nxt	None	Running	1 day, 1 hour
ie1-fcn01b-nxt.nxt.betfair	centos-6-base-296	10.85.14.194	ie1_fcn_nxt_50c99d2cbe	-	Active	nxt	None	Running	1 day, 1 hour
ie1-fcn01a-qa.qa.betfair	centos-6-base-296	10.84.143.131	ie1_fcn_qa_cd6534098a	-	Active	qa	None	Running	1 day, 2 hours
ie1-fcn03a-prd.prd.betfair	centos-6-base-295	10.84.14.4	ie1_fcn_prd_9cfd5982db	-	Active	prd	None	Running	2 weeks, 4 days
ie1-fcn04a-prd.prd.betfair	centos-6-base-295	10.84.14.3	ie1_fcn_prd_9cfd5982db	-	Active	prd	None	Running	2 weeks, 4 days
ie1-fcn01a-prd.prd.betfair	centos-6-base-295	10.84.14.5	ie1_fcn_prd_9cfd5982db	-	Active	prd	None	Running	2 weeks, 4 days
ie1-fcn02a-prd.prd.betfair	centos-6-base-295	10.84.14.2	ie1_fcn_prd_9cfd5982db	-	Active	prd	None	Running	2 weeks, 4 days

Figure 74 - FCN Machines from one of the Data Centers

It is possible to affirm the IAP does not consume more resources than FCN. However, there is the need to understand if the used resources can be reduced. Since every application has now its own cluster and that is not the responsibility of IAP, probably, in comparison with FCN, the number of machines, processes, CPU and RAM allocated can be reduced. In this particular aspect, although the requirement is completed, it falls short of its potential.

8.1.2 Each application developed using the IAP has its own cluster

This alternative hypothesis is portraited in the NFR39. It shares the same concerns as the hypothesis presented in section 8.1.1, them being the (i) availability (c.f. NFR19 and NFR20), (ii) scalability (c.f. NFR21) and (iii) reliability (c.f. NFR20) of the solution.

8.1.2.1 Metrics

To understand if the applications have indeed their own clusters, it is important to show evidence. A metric to be used is the existence of a list of machines of a particular application that was built using IAP. From that list, it must be possible to see the presence of machines for every company's environment.

8.1.2.2 Methodologies

In order to collect the list of machines, the OpenStack Platform must be consulted. There is where all resources related information, (i) instances, (ii) volumes, (iii) images and others are centralized.

8.1.2.3 Results and Analysis

Regarding the alternative hypothesis, that stated that each application developed using IAP has its own cluster, it is also possible to prove it without the need to apply a statistical test. Both BMU and FMU have their own clusters. An evidence of this claim can be seen in Figure 75 where every BMU machine from one of the Data Centers can be consulted.

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State
ie2-bmu02b-prd.prd.betfair	centos-6-base-294	10.168.50.133	ie2_bmu_prd_9cfd5982db	-	Active	prd	None	Running
ie2-bmu03b-prd.prd.betfair	centos-6-base-294	10.168.50.132	ie2_bmu_prd_9cfd5982db	-	Active	prd	None	Running
ie2-bmu04b-prd.prd.betfair	centos-6-base-294	10.168.50.131	ie2_bmu_prd_9cfd5982db	-	Active	prd	None	Running
ie2-bmu01b-prd.prd.betfair	centos-6-base-294	10.168.50.130	ie2_bmu_prd_9cfd5982db	-	Active	prd	None	Running
ie2-bmu01b-nxt.nxt.betfair	centos-6-base-294	10.165.61.3	ie2_bmu_nxt_3475e31831	-	Active	nxt	None	Running
ie2-bmu02b-nxt.nxt.betfair	centos-6-base-294	10.165.61.2	ie2_bmu_nxt_3475e31831	-	Active	nxt	None	Running
ie2-bmu01b-qa.qa.betfair	centos-6-base-294	10.168.143.130	ie2_bmu_qa_cd6534098a	-	Active	qa	None	Running
ie2-bmu01b-drk.drk.betfair	centos-6-base-275	10.164.102.67	ie2_bmu_drk_9cfd5982db	-	Active	prd	None	Running
ie2-bmu02b-drk.drk.betfair	centos-6-base-275	10.164.102.66	ie2_bmu_drk_9cfd5982db	-	Active	prd	None	Running

Figure 75 – BMU Machines from one of the Data Centers

Analysing Figure 75 against Figure 76, that portraits and FCN application called GIN, it is possible to assess that an application in FCN only had development environments. The two environments portraited in Figure 76 are development environments usually used to run functional tests and manual tests before notifying the FCN maintainers that the application can be released and deployed in FCN.

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created
ie2-gin02a-nxt.nxt.betfair	centos-6-base-296	10.165.70.130	ie2_gin_nxt_3475e31831	-	Active	nxt	None	Running	1 week, 1 day
ie2-gin01a-nxt.nxt.betfair	centos-6-base-296	10.165.70.131	ie2_gin_nxt_3475e31831	-	Active	nxt	None	Running	1 week, 1 day
ie2-gin01b-qa.qa.betfair	centos-6-base-296	10.164.208.194	ie2_gin_qa_002f95c9b0	-	Active	qa	None	Running	1 week, 1 day

Figure 76 – GIN Machines from one of the Data Centers

8.1.3 The theme scheme is easy to manage and maintain;

The theme scheme being easy to manage and maintain is a fairly generic and hard to measure requirement (c.f. NFR16). Nevertheless, since it was one of the goals of this project, it is important to evaluate its achievement.

8.1.3.1 Metrics

The metrics that can be used to help are the IAP maintainers perception of usability and maintainability.

8.1.3.2 Methodologies

To assess the metrics discussed in 8.1.3.1, a satisfaction survey to FCN and IAP maintainers was developed whose results are present in section 8.2.1

8.1.3.3 Results and Analysis

In section 8.2.1, through the satisfaction survey sent to IAP Maintainers, it is possible to see that IAP maintainers find it easy to add a feature and to debug IAP.

These results do not allow to quantify an absolute value for how easy to manage and maintain is the new solution. However, the survey shows that in comparison with FCN, IAP maintainers agree that the theme scheme is easier to manage and maintain.

8.1.4 IAP framework does not constrain the technological decisions of the applications' teams.

For the developer day-to-day job, this is one of the most valuable requirements (c.f. NFR11). The freedom given by allowing any technology to be used can make developers more motivated and engaged with their own applications. This hypothesis

8.1.4.1 Metrics

The metric that should be used is the ability to build an application with a different framework than the one used in FCN and integrate it with IAP. This metric can only have a binary result: (i) the application is successfully integrated (ii) or not.

8.1.4.2 Methodologies

Since no new application was yet developed using a different stack, the methodology used to evaluate this alternative hypothesis is the creation of Proof of Concepts. These proof of concepts

8.1.4.3 Results and Analysis

Regarding hypothesis (iv), that states that the IAP framework does not constrain the technological decisions of the applications' teams, the same was also proved without the need of statistical tests. The same was proved through a manual integration test with a POC – Proof of Concept – where a mock application was created using React. An application that is currently being designed to be built upon IAP is also going to use React as its Client Side

Framework. These two factors are evidence that teams are no longer constrained technologically. At the moment of this writing, two frameworks are being used, when in FCN only a particular version of Angular could be used. Code 30 shows the POC manifest created to prove that IAP can work with other frameworks.

```
{
    "assets": {
        "scripts": [{
            "src": "https://@@TLA.@@ENVIRONMENT.internal/bundle.js"
        }],
        "styles": []
    },
    "bootstrap": { "root": "/" },
    "description": "IAP React Sample App",
    "fullname": "IAP React App",
    "name": "IAP React App",
    "hidden": false,
    "tla": "ira",
    "version": "0.0.1",
    "template": "agnostic",
    "hasIapHeader": true
}
```

Code 30 – React POC Manifest

8.1.5 IAP provides updated documentation that covers all the areas of the platform, from how each component and service works, to how an application can be integrated.

Measuring the success of IAP documentation depends on many factors (c.f. NFR12 to NFR15). As stated in section 6.4.4, this is more of a human problem and whether or not the developer writes the documentation and the quality of the same.

8.1.5.1 Metrics

Nevertheless, since it was one of the goals of this project, it is important to assess if there were improvements in comparison with FCN and if the developers that have only used IAP find it useful.

8.1.5.2 Methodologies

To assess the metrics discussed in 8.1.5.1, a satisfaction survey to FCN and IAP maintainers was developed (c.f. Annexe F.1), whose results are present in section 8.2.1. Another one was developed for the users that have only worked with IAP (c.f. Annexe F.2) where results can be found in section

8.1.5.3 Results and Analysis

Scrutinising the hypothesis (v), that states that IAP provides updated documentation that covers all the areas of the platform, from how each component and service works, to how an application can be integrated, the same was proven through the satisfaction survey in section 8.2.1 and 8.2.2

The assessment that not only developers who worked with FCN acknowledge the improvements but new IAP users also find the documentation useful helps to conclude that the raised requirements (c.f. NFR12 to NFR15) were successfully delivered.

8.1.6 FCN services and components have logging mechanisms implemented that are integrated with the company logging collectors.

This is the goal that raised the more requirements. As discussed in the subchapters 6.5 and 7.5, the NFR20 to NFR31 are abstracted by the use of internal tools and infrastructure (c.f. NFR35). Also, in subchapters 6.5 and 7.5, it is demonstrated how are the logging mechanisms provided (c.f.NFR9).

The only thing left to evaluate is the integration between the provided mechanisms and the company's log storage, Splunk.

8.1.6.1 Metrics

The metrics that are desired to be collected are the presence of IAP related logs in Splunk.

8.1.6.2 Methodologies

Manual integration tests are going to be performed to understand if the logs are being collected and stored in the company's log collector, Splunk.

8.1.6.3 Results and Analysis

This hypothesis was proven as seen in subchapter 7.5 and logs can already be seen in Splunk. Figure 77 shows some logs collected from a particular IAP machine. One particular aspect that is important to clarify is the index being called *ppb-fcn*. At the time of this writing, Splunk has an index limit that was already reached and, therefore, IAP metrics are being sent to the FCN index.

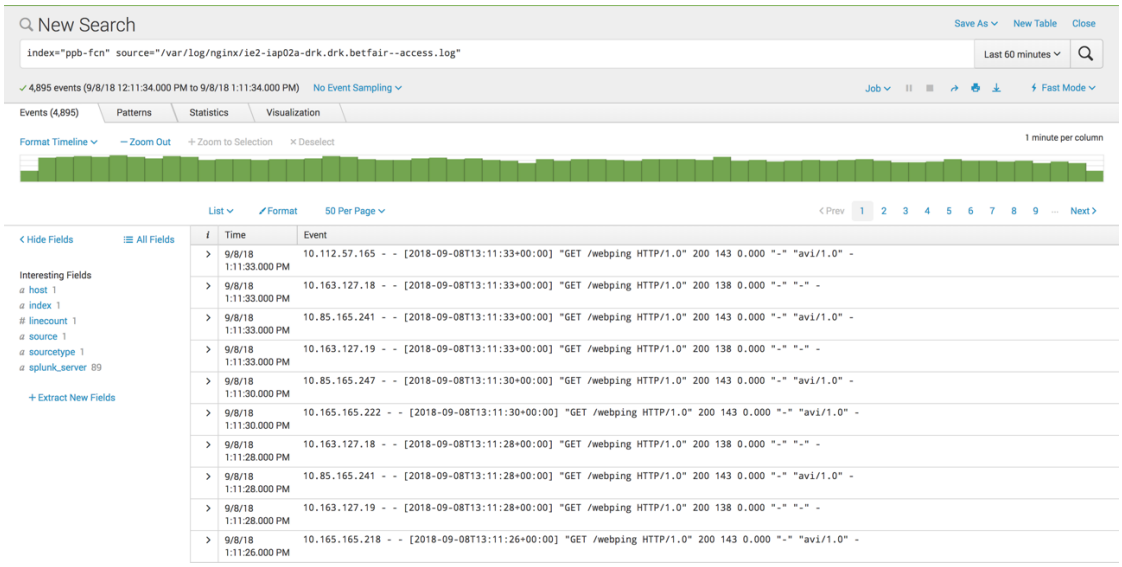


Figure 77 – Splunk with IAP logs

8.2 Satisfaction Survey

This project creates three distinct groups that can be enquired about their satisfaction with IAP: (i) developers that have worked with FCN and now IAP (ii) the FCN and IAP maintainers, that not only maintain the platform but also develop applications that take leverage in it; (iii) and developers that have only worked with FCN;

For both group (ii) and (iii), two separate surveys were created. For group (ii), the survey in the Annexe F.1 focuses on understanding if IAP brought benefits over FCN. It is important to take in consideration that this group has deep knowledge on both FCN and IAP since it is composed by the maintainers and creators themselves. For Group (iii), the survey in Annexe F.1 goal is to understand if IAP helped them leverage their applications and how well was that accomplished. The questions in the survey have four possible answers: (i) 1 - Strongly Disagree; (ii) 2 – Disagree; (iii) 3 – Agree; (iv) 4 – Strongly Agree.

Unfortunately, at the time of this writing, there are no developers, besides the IAP and FCN maintainers, that have developed or maintained applications on both platforms. Another aspect to notice is the shortage of answers to the survey on Annexe F.2. Only a few people of a specific team have worked with IAP which led to only two answers.

8.2.1 IAP Maintainers' Survey Results

Regarding whether IAP's documentation is easier to find, 100% strongly agree with that statement, as seen in Figure 78. This indicates that the platform documentation is now easy to reach, which may help reduce the need for direct help from the maintainers.

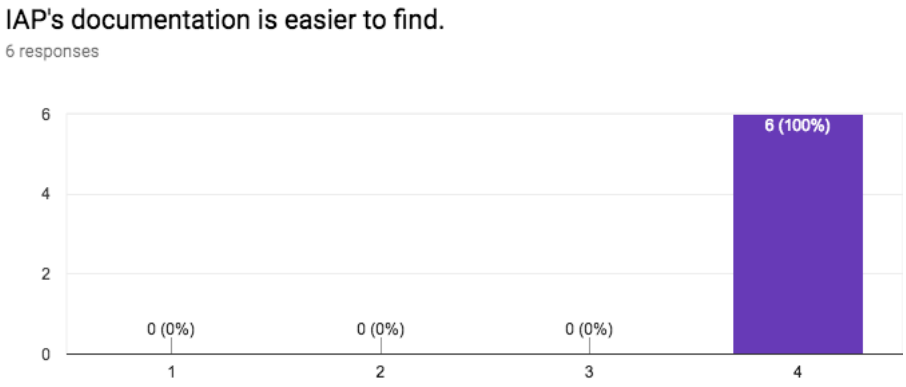


Figure 78 – IAP's Documentation Findability in Comparison with FCN

When trying to understand if IAP's documentation is more clear and meaningful, Figure 79 shows that five subjects strongly agreed and one agreed with the statement. It is possible to conclude that it becomes easier for a developer to understand what IAP offers and how to use it. Developers are now able to spend less time reading the documentation to find what they need.

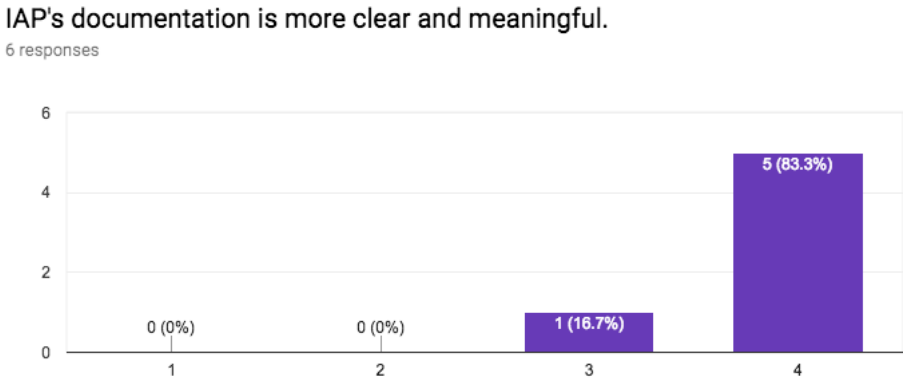


Figure 79 – IAP's documentation content in comparison with FCN

One of the bottlenecks of FCN's documentation was that it did not cover all the needed areas for a developer to use the platform. When enquired, all the enquiries agreed, four of them strongly, that IAP's documentation overcame that pitfall. Such can be verified in Figure 80.

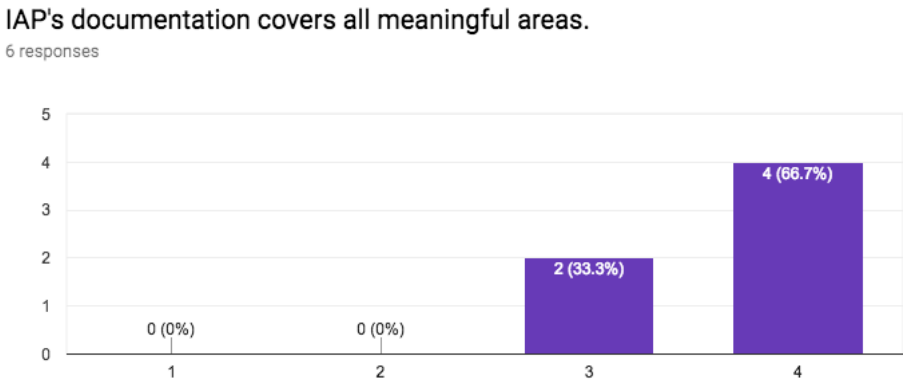


Figure 80 – IAP's documentation subjects coverage in comparison with FCN

One of the goals of this survey was to understand how does the documentation impacts the engagement with the framework. A good documentation can lead developers to a more constant use since it eases its job. By analysing Figure 81, it can be concluded that this topic does not meet a consensus. Half of the population strongly agreed with the statement, while the other half disagreed. Many reasons can lead to such a result, depending on how much they use documentation. Many members of the team had deep knowledge on both FCN and IAP while others only entered half-way through IAP and documentation may have helped them get started.

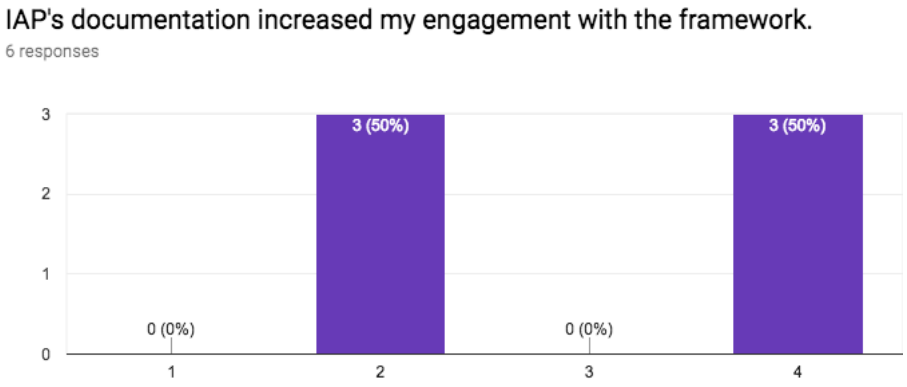


Figure 81 – IAP's documentation engagement's impact

A consensus is reached when the enquiries are asked if they are happier with IAP. All the population strongly agreed, Figure 82, that they are happier with IAP more than they were with FCN. This is an excellent result to prove that those who had experience with both frameworks can see a clear improvement.

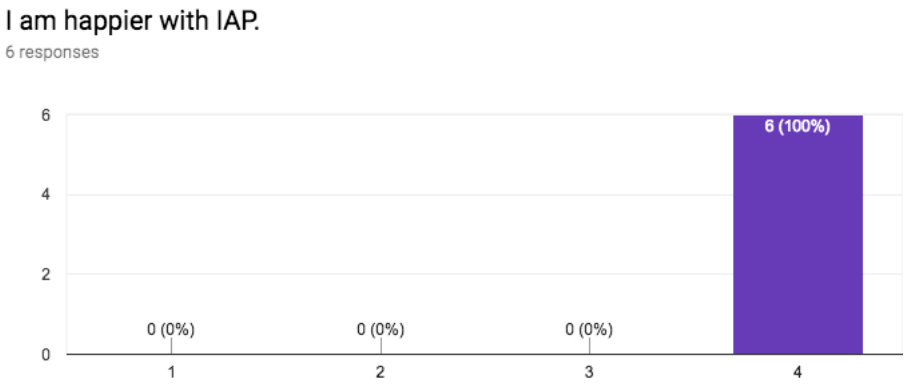


Figure 82 – IAP's satisfaction in comparison with FCN.

When trying to understand, from an IAP's user point of view, if IAP was easier to use, Figure 83 shows that everyone agrees, five of them strongly, that IAP is indeed easier to use. This shows that many development problems were overcome which makes the development process less painful and it can become a selling point for other teams.

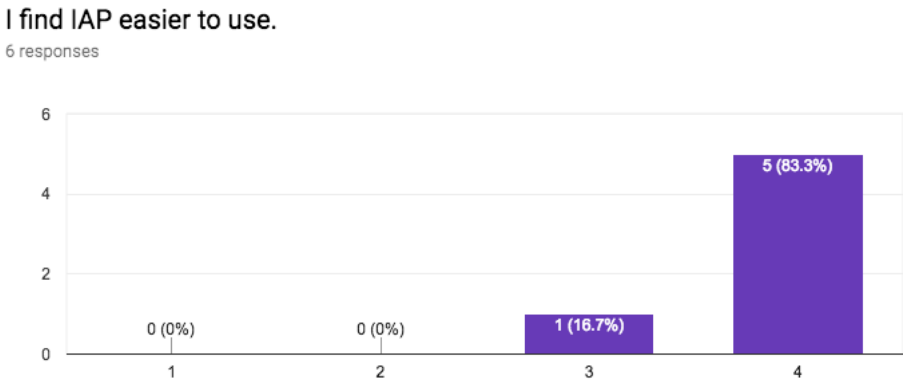


Figure 83 – IAP's usability in comparison with FCN

Since the group chosen to answer this survey are the FCN and IAP's maintainers, it is important to understand how is IAP's maintainability when compared with FCN. Figure 84 makes possible to draw the conclusion that in IAP it is easier to add a feature. Four of them strongly agree while the other two simply agree. These last ones can be explained by the fact that IAP introduces more moving parts, which can make it difficult to reason about a solution.

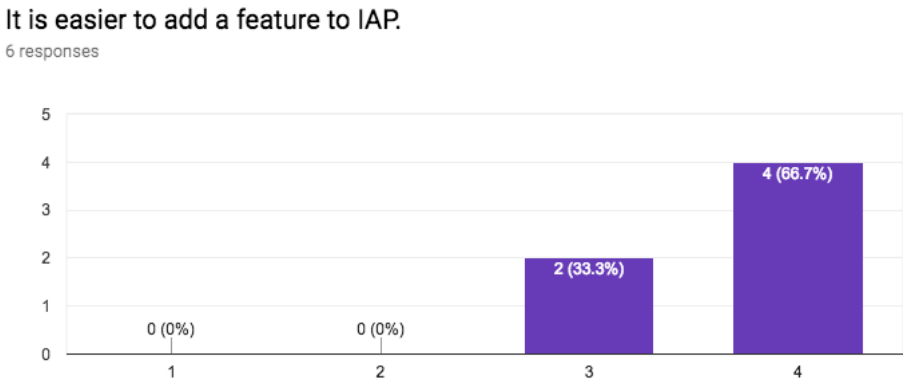


Figure 84 – IAP's feature development in comparison with FCN

Finally, Figure 85 shows the answers regarding IAP's debuggability. There are four persons that strongly agree, one that agrees, and one that disagrees. The conclusions to be drawn from these results are the same ones from Figure 84. Many moving parts can be it difficult to reason about a specific workflow, debug more painful. However, there was a clear improvement in comparison with FCN.

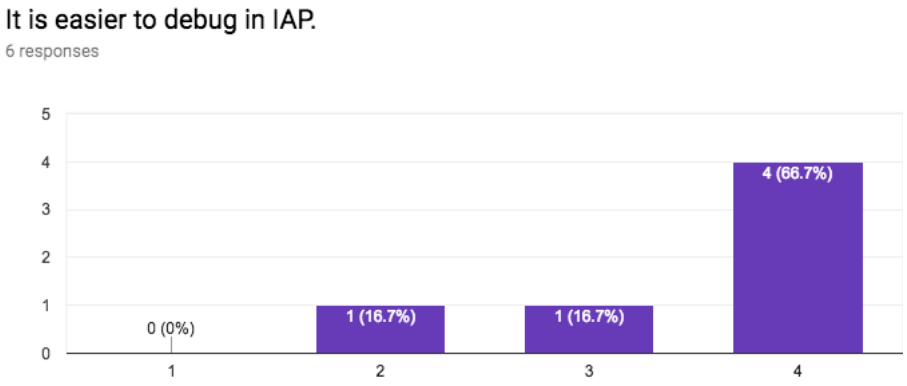


Figure 85 – IAP's debuggability in comparison with FCN

Through this survey, it is clear that all areas were greatly improved in IAP. However, there is room for improvements, being the most notable the IAP debuggability and the engagement that the provided documentation between its users and the platform.

8.2.2 First Time IAP Users Survey Results

Since the target group of this survey was a population that has never worked with FCN, a comparison analysis cannot be performed. However, it is important to understand the satisfaction level with their IAP's usage.

Regarding whether IAP's documentation is easier to find, 50% strongly agree and 50% agree with that statement, as seen in Figure 86. These results confirm the conclusions drawn in Figure 78.

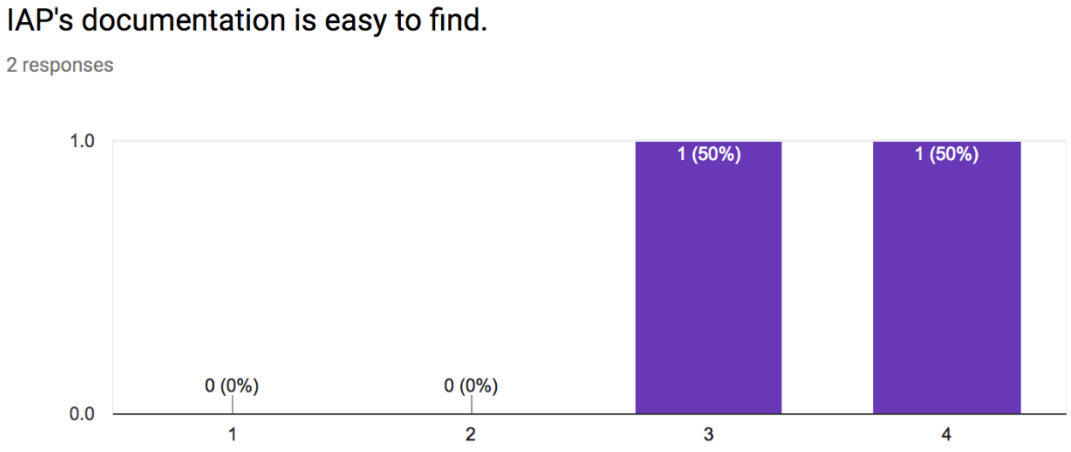


Figure 86 – IAP's Documentation Findability

When trying to understand if IAP's documentation is more clear and meaningful, Figure 79 shows once again that one enquiry agreed and other strongly agree. Once again, the conclusion goes hand to hand to hand with the conclusions from Figure 79.

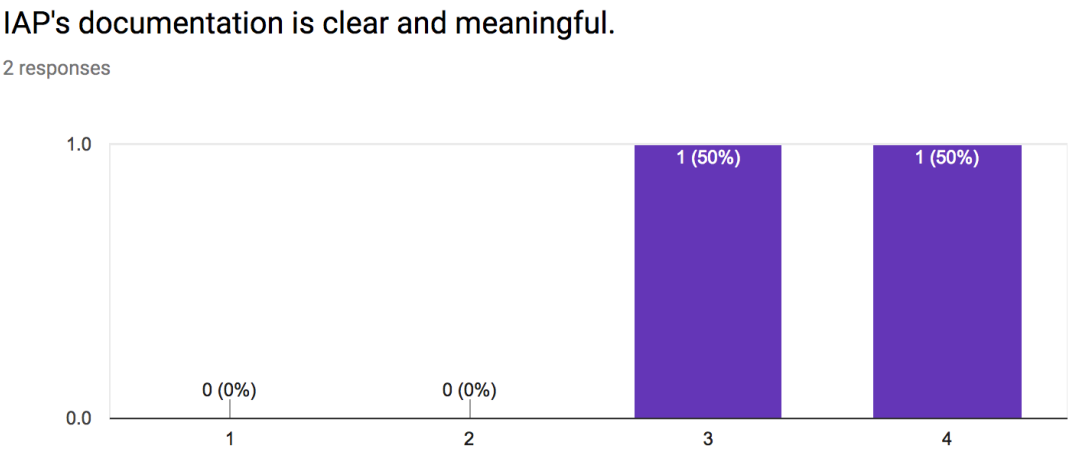


Figure 87 – IAP's documentation content

Looking into what does the documentation covers, Figure 88 shows that all developers that have only used IAP strongly agreed that its documentation covers all meaningful areas. This is extremely important since it does force developers to uncover hidden details that could deteriorate the development process.

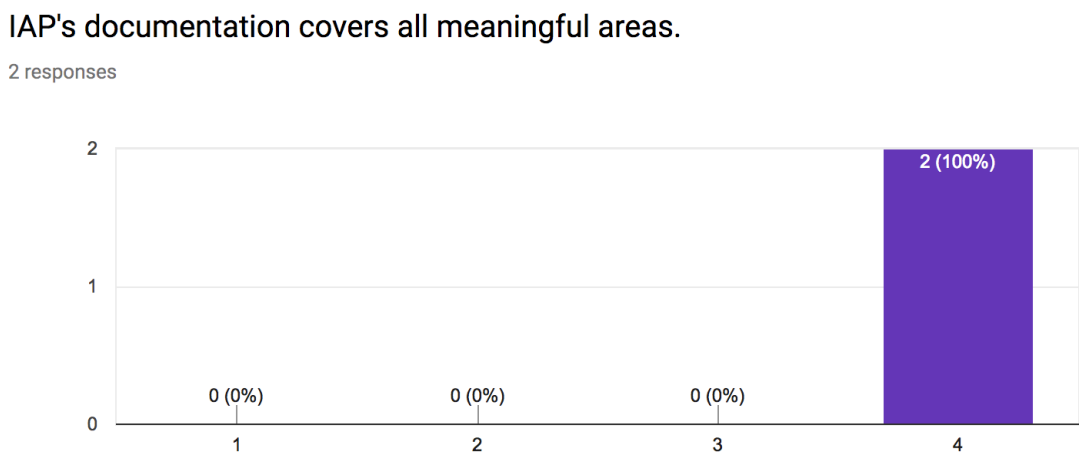


Figure 88 – IAP's documentation subjects coverage

The engagement that IAP's documentation provides, Figure 89, shows equal results when compared with Figure 81. However, the reasons may not be related. and due to the small number of participants, it is hard to draw a conclusion from two so contrasting values.

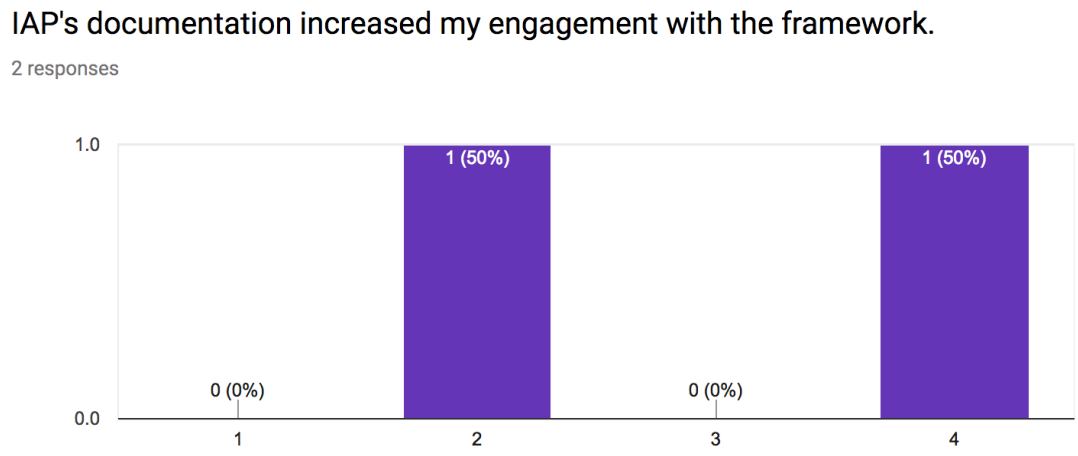


Figure 89 – IAP's documentation engagement's impact

When stated that they were happy with IAP, 50% strongly agree and 50% agree, as seen in

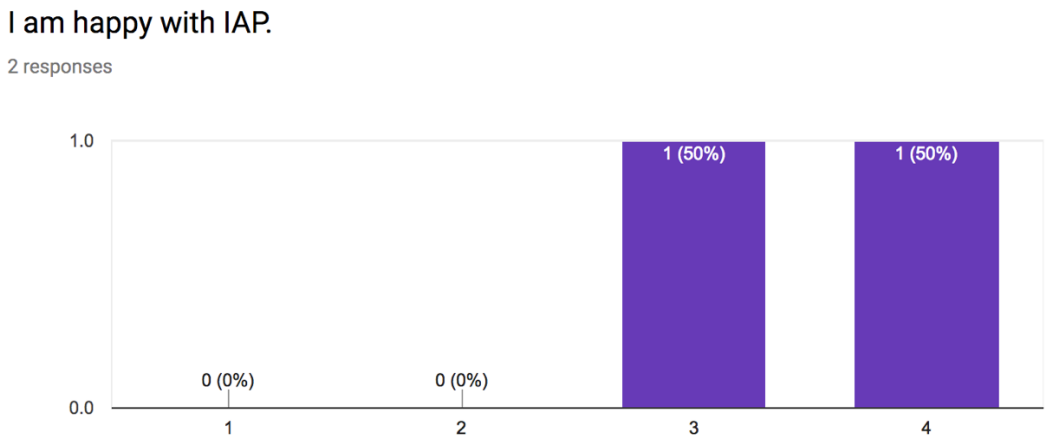


Figure 90. The results only strengthen the conclusion of Figure 82 and stress out that IAP was a success not only to its maintainers but to the ones using it for the first time.

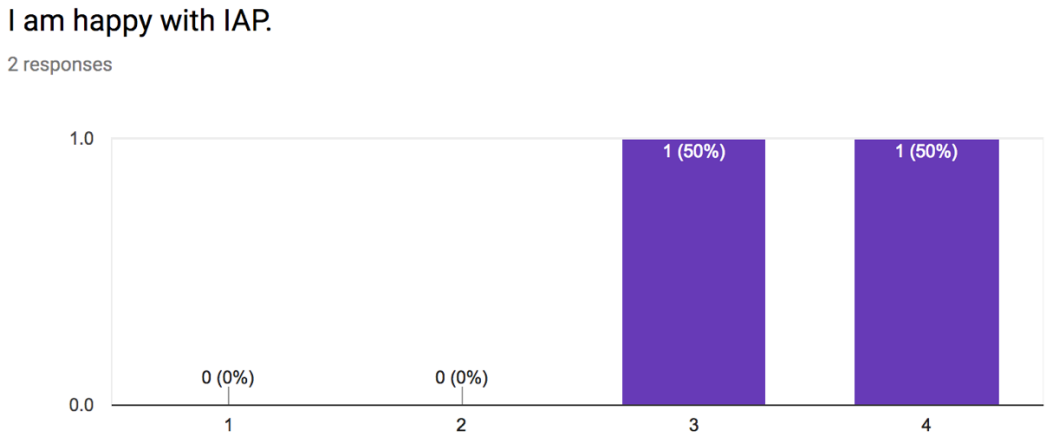


Figure 90 – IAP's satisfaction.

When trying to understand, from an IAP's user point of view, if IAP was easy to use, Figure 91 shows that everyone agrees, 50% strongly, that IAP is indeed easy to use. This shows that many development problems were overcome leading that the IAP's first-time users have had an experience that they enjoyed.

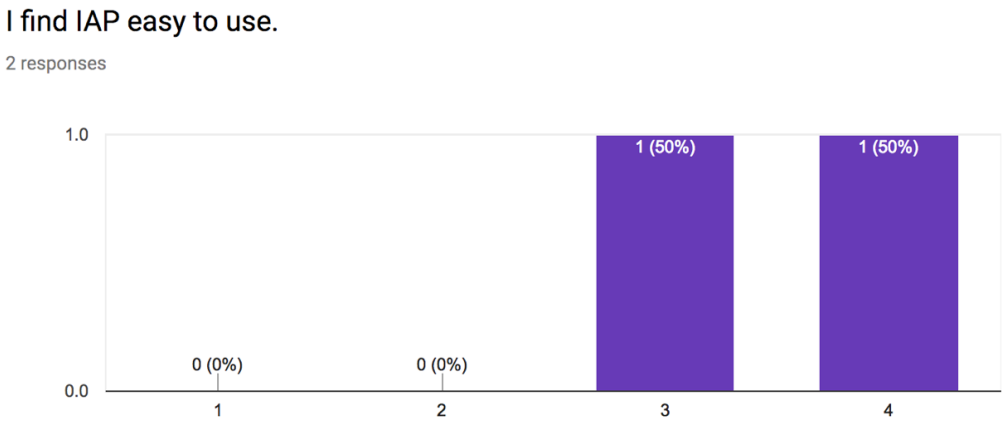


Figure 91 – IAP's usability in comparison with FCN

Due to the lack of participation in this survey, it is hard to draw conclusions with a high level of confidence since the population used was too small. However, at the light of the available results at the time of this writing, it is possible to stress out, even more, the success that IAP was, but now taking into consideration a non-bias opinion from people who never worked with IAP or FCN before.

9 Conclusion

This project came from the need to improve and refactor an existing framework from the Paddy Power Betfair's technological stack. There were a bunch of challenges and problems that needed to be approached to improve the framework efficiency for its maintainers, application developers and users.

The solutions presented in this document made it possible to overcome many challenges that were constraining the development process of many teams. With IAP, the company is now in a position where all internal tooling can be centralized and it's development eased. Teams can now fully own their own applications and can focus on developing value for the company. Not only both users and developers can become more productive, but the company is now in a position where IAP can become a global standard for developing internal applications.

Overall, the project was a success. Almost every goal was fully achieved and the evaluation, chapter 8, shows that not only were achieved but were achieved with outstanding results.

9.1 Goals Achievement

At the beginning of the project, the problems and consequently nine goals were defined: (i) applications should have independent releases; (ii) applications should run in separate clusters; (iii) complex theme scheme should be refactored; (iv) the framework should not force specific technologies; (v) documentation should be vast and updated; (vi) the platform should provide authentication and authorisation mechanisms; (vii) there should be logging mechanisms; (viii) conventions should be established between projects; (ix) user interface should be consistent across applications. Table 13 resumes the achieved goals.

Table 13 –Goals, Respective Requirements and Degree of Achievement

Goals	Requirements	Degree of Achievement
Goal (i) and Goal (ii)	NFR7, NFR18 to NFR22, NFR38 and NFR39	Fully Achieved
Goal (iii)	NFR10, NFR16 and NFR36	Fully Achieved
Goal (iv)	NFR11	Fully Achieved
Goal (v)	NFR12 to NFR15	Fully Achieved
Goal (vi)	NFR2 to NFR6 and NFR8	Not Achieved
Goal (vii)	NFR9, NFR23 to NFR34 and NFR38	Fully Achieved
Goal (viii)	No Requirement Created	Fully Achieved
Goal (ix)	NFR17	Partially Achieved

Goal (i) and Goal (ii) were achieved through the same solution. Not only was successfully designed (cf. 6.1) and implemented (cf. 7.1), as it improved the release and deployment times, as demonstrated in section 8.1.1.

To achieve the Goal (iii), many solutions were thought and the trade-offs of each solution had to be well measured to avoid getting back to a complex theme scheme. The solutions studied can be found in subchapter 6.2 while the implementation can be seen in subchapter 7.2. There are proofs of concept developed and the components are already available for usage.

Goal (iv) was one of the goals that users were most eager to have available. Right now, the IAP dashboard is being rebuilt with React (the previous version was in Angular) and two new applications are being built with React. Design can be found in subchapter 6.3 while implementation in subchapter 7.3. Section 8.1.4 also evaluates this solution.

To complete Goal (v), a mix of solutions were followed to tackle different areas. In, subchapter 6.4 it is possible to see how each solution can overcome problems found in different areas. The respective implementation is seen in subchapter 7.4 and its success was proved through the satisfaction survey in both section 8.2.1 and 8.2.2.

Goal (vii) was also successfully achieved by leveraging on the current company's solutions regarding logging. The design, in subchapter 6.5, shows how two different solutions were thought to provide backend and frontend logging. Subchapter 7.5 shows the respective implementation and in evaluation chapter, it is possible to see the collected logs.

Goal (viii) was approached having in mind that it was a subject matter and forcing a specific convention over every application can go against what IAP stands for. However, some approaches were design and implemented (subchapter 6.6 and 7.6). However, no evaluation was performed regarding conventions due to its subjective nature.

Finally, Goal (ix) was a goal that appeared half-way through the project with no expectations of being achieved. However, in subchapter 7.8 it is possible to see that the provided design in

Annexe G.1 is almost implemented. Moreover, these designs are already being integrated with two new applications that are being developed at the time of this writing.

Unfortunately, not every goal was fully achieved.

Goal (vi) was not achieved. IAP already provides authentication mechanisms but regarding authorisation, only the designed was developed (subchapter 6.7). In the implementation subchapter 7.7, it is possible to see what already exists in term of authentication. However, no work was done in implementing the authorisation.

There are many factors who contributed to this outcome. The dependency on a third party to provide feedback on how could we handle authorisation played a major role. Only in a later stage of the project were these details provided. Moreover, the constant context switching between projects made it difficult to plan an implementation.

9.2 Future Work

IAP is an ongoing work in progress. As some needs are satisfied, others arise. Authorisation is a top priority and a new requirement was identified: federation.

The focus now is on migrating the FCN applications to IAP so FCN can be deprecated. By having a lot of applications working successfully under IAP, it becomes easier to attract new applications and teams.

New services are being thought to ease even more work and provided more functionalities to developers. User preferences are one of them. A storage using MySQL to store the preferences by users is already being implemented.

At the end of the day, the greatest goal of IAP is to become a standard inside the company when it comes to the development internal tooling web applications and, who knows, selling this solution to other companies.

9.3 Final Considerations

However, there were many challenges that were not estimated when planning this project. Many team members left the company and many projects gained higher priority. Slowing IAP's development and removing the focus from it. This intermittent development style made it really difficult to manage the project while the loss of valuable team members slowed production. There were also third-party dependencies that undermined some project goals.

It is, without a doubt, the most challenging project I have ever had the pleasure to work on and has made me grow as professional in a way that I thought it would not possible for a few years.

References

[1] Wikipedia, “Software Deployment,” 07 December 2017. [Online]. Available: https://en.wikipedia.org/wiki/Software_deployment. [Accessed 25 April 2018].

[2] Twitter, “Hogan.js,” 17 June 2017. [Online]. Available: https://github.com/twitter/hogan.js/.

[3] Paddy Power Betfair, “Fabric Documentation Central,” Paddy Power Betfair, 10 January 2017. [Online]. Available: https://confluence.app.betfair/display/SCO/Fabric+Documentation+Central.

[4] The Sinon.JS Committers, “Sinon.js,” 2017. [Online]. Available: http://sinonjs.org/. [Accessed 22 January 2018].

[5] Amazon, “About AWS,” 2018. [Online]. Available: https://aws.amazon.com/about-aws/. [Accessed 25 January 2018].

[6] Amazon, “Types of Cloud Computing,” 2018. [Online]. Available: https://aws.amazon.com/types-of-cloud-computing/. [Accessed 31 January 2018].

[7] M. Rouse, “Amazon Web Services (AWS),” December 2017. [Online]. Available: http://searchaws.techtarget.com/definition/Amazon-Web-Services. [Accessed 31 January 2018].

[8] D. Poccia, F. Liverani and P. Latella, *AWSome Days,* Milano, 2015.

[9] Google, “Why Google,” 2018. [Online]. Available: https://cloud.google.com/why-google/. [Accessed 25 January 2018].

[10] K. Marko, “A Closer Look At Google Cloud: Don’t Dismiss It,” 11 November 2017. [Online]. Available: http://markoinsights.com/2016/11/11/google-cloud-update/. [Accessed 1 February 2018].

[11] Google, “Architecture: Web Application on Google App Engine,” 4 October 2016. [Online]. Available: https://cloud.google.com/solutions/architecture/webapp. [Accessed 1 February 2018].

[12] Managility, “Microsoft Azure Services,” Managility, 2018. [Online]. Available: http://www.managility.co/microsoft-azure-services/. [Accessed 02 February 2018].

[13] Microsoft Azure, “Solution architecture: Social mobile and web app with authentication,” 2018. [Online]. Available: https://azure.microsoft.com/en-us/solutions/architecture/mobile-app-social-with-authentication/.

[14] Wikipedia, “ List of JavaScript libraries,” 6 Febraury 2018. [Online]. Available: https://en.wikipedia.org/wiki/List_of_JavaScript_libraries#Web-application_related_(MVC,_MVVM).

[15] L. Voss, “State of JavaScript Framworks 2017,” 3 January 2018. [Online]. Available: https://www.npmjs.com/npm/state-of-javascript-frameworks-2017-part-1.

[16] ReactJS, “ReactJS,” 2018. [Online]. Available: https://reactjs.org/. [Accessed 03 February 2018].

[17] Facebook, “React Native,” 2018. [Online]. Available: https://facebook.github.io/react-native/.

[18] Tutorials Point, “ReactJS Overview,” [Online]. Available: https://www.tutorialspoint.com/reactjs/reactjs_overview.htm.

[19] Facebook, “Flux Concepts,” 19 January 2017. [Online]. Available: https://github.com/facebook/flux/tree/master/examples/flux-concepts.

[20] Facebook, “In Depth Overview,” 2015. [Online]. Available: https://facebook.github.io/flux/docs/in-depth-overview.html#content.

[21] Angular, “Architecture,” 2018. [Online]. Available: https://angular.io/guide/architecture.

[22] Vue.js, “What is Vue.js?,” 2018. [Online]. Available: https://vuejs.org/v2/guide/.

[23] VueJS, “Comparison with Other Frameworks,” 2018. [Online]. Available: https://vuejs.org/v2/guide/comparison.html.

[24] Vue.js, “Reactivity,” 2018. [Online]. Available: https://vuejs.org/v2/guide/reactivity.html.

[25] A. Gore, “React or Vue: Which Javascript UI Library Should You Be Using?,” 22 December 2016. [Online]. Available: https://medium.com/js-dojo/react-or-vue-which-javascript-ui-library-should-you-be-using-543a383608d.

[26] J. Neuhaus, “Angular vs. React vs. Vue: A 2017 comparison,” 2017 August 2017. [Online]. Available: https://medium.com/unicorn-supplies/angular-vs-react-vs-vue-a-2017-comparison-c5c52d620176.

[27] J. Potter, “angular vs react vs vue,” 04 February 2018. [Online]. Available: http://www.npmtrends.com/angular-vs-react-vs-vue.

[28] S. Krause, “JS Framework Benchmark,” [Online]. Available: https://rawgit.com/krausest/js-framework-benchmark/master/webdriver-ts-results/table.html.

[29] WebComponents.org, “What are web components?,” 2018. [Online]. Available: https://www.webcomponents.org/introduction.

[30] W3C, “Custom Elements,” 04 December 2017. [Online]. Available: https://w3c.github.io/webcomponents/spec/custom/#custom-elements.

[31] E. Bidelman, “Custom Elements,” 3 January 2018. [Online]. Available: https://developers.google.com/web/fundamentals/web-components/customelements.

[32] A. Deveria, “Custom Elements V1,” 2018. [Online]. Available: https://caniuse.com/#feat=custom-elementsv1.

[33] A. Deveria, “HTML Imports,” 2018. [Online]. Available: https://caniuse.com/#feat=imports.

[34] A. Deveria, “Shadow DOM V1,” 2018. [Online]. Available: https://caniuse.com/#search=Shadow%20DOM.

[35] A. Deveria, “HTML Templates,” 2018. [Online]. Available: https://caniuse.com/#search=HTML%20templates.

[36] Mozilla Developer Network, “Web Components,” 7 February 2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Web_Components.

[37] Web Components Organization, “Introduction,” [Online]. Available: https://www.webcomponents.org/introduction.

[38] SkateJS, “SkateJS,” 03 February 2018. [Online]. Available: https://github.com/skatejs/skatejs.

[39] Stencil, “Stencil: A Compiler for Web Components,” 2018. [Online]. Available: https://stenciljs.com/docs/intro/.

[40] Polymer, “Roadmap,” 19 January 2018. [Online]. Available: https://github.com/Polymer/project/blob/master/Roadmap.md.

[41] Polymer Project, “Polymer Project,” 2017. [Online]. Available: https://www.polymer-project.org/.

[42] R. Sharp, “What is a polyfill?,” 08 October 2010. [Online]. Available: https://remysharp.com/2010/10/08/what-is-a-polyfill/.

[43] Polymer Project, “Data Binding,” 09 January 2018. [Online]. Available: https://www.polymer-project.org/2.0/docs/devguide/data-binding.

[44] Reddit, “Solutions for Framework Agnostic Web Components,” [Online]. Available: https://www.reddit.com/r/Frontend/comments/6efdll/solutions_for_framework_agnostic_web_components/.

[45] platosha, “Angular Polymer,” 9 June 2017. [Online]. Available: https://github.com/platosha/angular-polymer.

[46] jscissr, “React Polymer,” 30 October 2017. [Online]. Available: https://github.com/jscissr/react-polymer.

[47] Ionic Open Source, “Intro,” [Online]. Available: https://stenciljs.com/docs/intro.

[48] sinedied, “Stencil.js: it’s finally time for vanilla web components!,” 29 August 2017. [Online]. Available: https://medium.com/@sinedied/stencil-js-its-finally-time-for-vanilla-web-components-927d26b573e1.

[49] C. Codeman, “Polymer 2 - the 10Kb Web Framework,” 31 March 2017. [Online]. Available: https://www.captaincodeman.com/2017/03/31/polymer-2-10kb-web-framework.

[50] StencilJS, “StencilJS,” 2018. [Online]. Available: https://stenciljs.com/.

[51] devlucky, “Building a custom tag input with Skate.js,” 2017 January 2017. [Online]. Available: https://hackernoon.com/building-a-custom-tag-input-with-skate-js-fbd4cdf744f.

[52] Wikipedia, “Progressive Web Apps,” 24 September 2018. [Online]. Available: https://en.wikipedia.org/wiki/Progressive_Web_Apps.

[53] P. Vallender, “What is value proposition and why is it important,” 19 March 2013. [Online]. Available: https://www.blendb2b.com/blog/what-is-value-proposition. [Accessed 25 January 2018].

[54] P. Koen, H. Bertels and E. Kleinschmidt, “Managing the Front End of Innovation—Part II,” *RESEARCH-ON-RESEARCH ,* 2014.

[55] G. A. R. B. A. C. J. D. R. D. C. E. K. H. M. I. A. J. R. K. R. S. A. S. a. K. W. Peter Koen, "PROVIDING CLARITY AND A COMMON LANGUAGE TO THE "FUZZY FRONT END"," *Industrial Research Institute,* 2001.

[56] S. Nicola, E. P. Ferreira and J. J. P. Ferreira, "A NOVEL FRAMEWORK FOR MODELING VALUE FOR THE CUSTOMER, AN ESSAY ON NEGOTIATION," *International Journal of Information Technology & Decision Making,* vol. 11, no. 03, May 2012.

[57] D. Walters and G. Lancaster, "Implementing value strategy through the value chain," *Management Decision,* vol. 38, no. 3, p. 19, 2000.

[58] A. Lindgreen and F. Wynstra, "Value in business markets: What do we know? Where are we going?," *Industrial Marketing Management,* 2003.

[59] T. Woodall, "Conceptualising 'Value for the Customer': An Attributional, Structural and Dispositional Analysis," *Academy of Marketing Science Review,* vol. 12, 2003.

[60] J. Lapierre, "Customer-perceived value in industrial contexts," *Journal of Business & Industrial Marketing,* vol. 15, pp. 122-140, 2000.

[61] A. Osterwalder, "THE BUSINESS MODEL ONTOLOGY A PROPOSITION IN A DESIGN SCIENCE APPROACH," UNIVERSITE DE LAUSANNE ECOLE DES HAUTES ETUDES COMMERCIALES, 2004.

[62] A. Osterwalder and Y. Pigneur, Business Model Generation, 2010.

[63] V. Allee, *What is Value Network Analysis,* 2011.

[64] V. Allee, "Value network analysis and value conversion of tangible and intangible assets," *Journal of Intellectual Capital,* vol. 9, no. 1, pp. 5-24, 2008.

[65] Mind Tools Content Team, "Porter's Value Chain - Understanding How Value Is Created Within Organzations," Mind Tools, [Online]. Available: https://www.mindtools.com/pages/article/newSTR_66.htm.

[66] T. Saaty, "Decision making with the analytic hierarchy process," *Inderscience Enterprises,* vol. 1, no. 1, pp. 83-98, 2008.

[67] Xplenty, "The SQL vs NoSQL Difference: MySQL vs MongoDB," 28 September 2017. [Online]. Available: https://medium.com/xplenty-blog/the-sql-vs-nosql-difference-mysql-vs-mongodb-32c9980e67b2.

[68] Rackspace Cloud Computing, "OpenStack," Rackspace Cloud Computing, 2018. [Online]. Available: https://www.openstack.org/.

[69] Google, “Material Design,” [Online]. Available: https://material.io/design/introduction/. [Accessed 2018].

[70] Google, “Angular Material,” 2018. [Online]. Available: https://material.angular.io/.

[71] Material-UI, “Material-UI,” 2018. [Online]. Available: https://material-ui.com/.

[72] M. Moura, “Vue Material,” 2018. [Online]. Available: https://vuematerial.io/.

[73] Bootstrap, “Costumize,” 2018. [Online]. Available: https://getbootstrap.com/docs/3.3/customize/.

[74] React-Bootstrap, “Custom Styles,” 2018. [Online]. Available: https://react-bootstrap.github.io/utilities/custom-styles/.

[75] getbem, “BEM - Block Element Modifier,” 2018. [Online]. Available: http://getbem.com/.

[76] M. Fowler, “CodeAsDocumentation,” 22 March 2005. [Online]. Available: https://www.martinfowler.com/bliki/CodeAsDocumentation.html.

[77] Neuland, “Micro-Frontends,” [Online]. Available: https://micro-frontends.org/.

[78] Facebook Open Source, “React Docs,” 2018. [Online]. Available: https://reactjs.org/docs/getting-started.html.

[79] Google, “Angular Docs,” 2018. [Online]. Available: https://angular.io/docs.

[80] Splunk Inc, “Splunk,” Splunk Inc, 2018. [Online]. Available: https://www.splunk.com/.

[81] Vardhan, “What is Splunk,” 30 January 2018. [Online]. Available: https://www.edureka.co/blog/what-is-splunk/.

[82] Airbnb, “Airbnb JavaScript Style Guide,” 5 July 2018. [Online]. Available: https://github.com/airbnb/javascript.

[83] Google, “Google JavaScript Style Guide,” 30 January 2018. [Online]. Available: https://google.github.io/styleguide/jsguide.html.

[84] R. Waldron, “Idiomatic,” 28 June 2018. [Online]. Available: https://github.com/rwaldron/idiomatic.js/.

[85] Standard JS, “Javascript Standard Style,” 13 August 2018. [Online]. Available: https://github.com/standard/standard.

[86] Aero, “AIS - Admin Identity Service,” Paddy Power Betfair, 09 August 2018. [Online]. Available: https://confluence.app.betfair/display/CET/AIS+-+Admin+Identity+Service.

[87] G. Fecior, “Admin Identity Service - BSIDL,” Paddy Power Betfair, 10 October 2010. [Online]. Available: https://confluence.app.betfair/display/CET/Admin+Identity+Service+-+BSIDL.

[88] Mozilla Developers Network, “<script>: The Script element,” 27 August 2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/script.

[89] Mozilla Developer Network, “<link>: The External Resource Link element,” 1 September 2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/link.

[90] Facebook Open Source, 2018. [Online]. Available: https://graphql.org/learn/queries/.

[91] Facebook Open Source, “Best Practices,” 2018. [Online]. Available: https://graphql.org/learn/best-practices/.

[92] Facebook Open Source, “Validation,” 2018. [Online]. Available: https://graphql.org/learn/validation/.

[93] Jenkins Job Builder Maintainers, “Jenkins Job Builder,” 2012. [Online]. Available: https://docs.openstack.org/infra/jenkins-job-builder/.

[94] Jenkins Job Builder Maintainers, “Job Definitions,” 2012. [Online]. Available: https://docs.openstack.org/infra/jenkins-job-builder/definition.html#job-template.

[95] React Bootstrap, “React Bootstrap,” 2018. [Online]. Available: https://react-bootstrap.github.io/.

[96] ng-bootstrap team, “Ng Bootstrap,” 2018. [Online]. Available: https://ng-bootstrap.github.io/#/home.

[97] Google, “Angular Material,” 2018. [Online]. Available: https://material.angular.io/.

[98] Ionic Open Source, “Unit Testing,” Ionic, 2018. [Online]. Available: https://stenciljs.com/docs/unit-testing/.

[99] Facebook Open Source, “docusaurus,” Facebook, 2018. [Online]. Available: https://docusaurus.io/.

[100] J. Marcey, “Introducing Docusaurus,” 14 December 2017. [Online]. Available: https://docusaurus.io/blog/2017/12/14/introducing-docusaurus.

[101] Wikipedia, “JSDoc,” 18 June 2018. [Online]. Available: https://en.wikipedia.org/wiki/JSDoc.

[102] The Pennsylvania State University, “3.2 - Hypothesis Testing (P-value approach),” The Pennsylvania State University, 2018. [Online]. Available: https://onlinecourses.science.psu.edu/statprogram/node/138.

[103] S. H. To, “Independent Samples T-Test,” Stephanie, 11 July 2015. [Online]. Available: http://www.statisticshowto.com/independent-samples-t-test/.

[104] Stephanie, “Two Tails T Distribution Table,” Statistics How To, 12 October 2017. [Online]. Available: http://www.statisticshowto.com/tables/t-distribution-table/#two.

[105] Microsoft, “What is Azure,” 2018. [Online]. Available: https://azure.microsoft.com/en-gb/overview/what-is-azure/. [Accessed 25 January 2018].

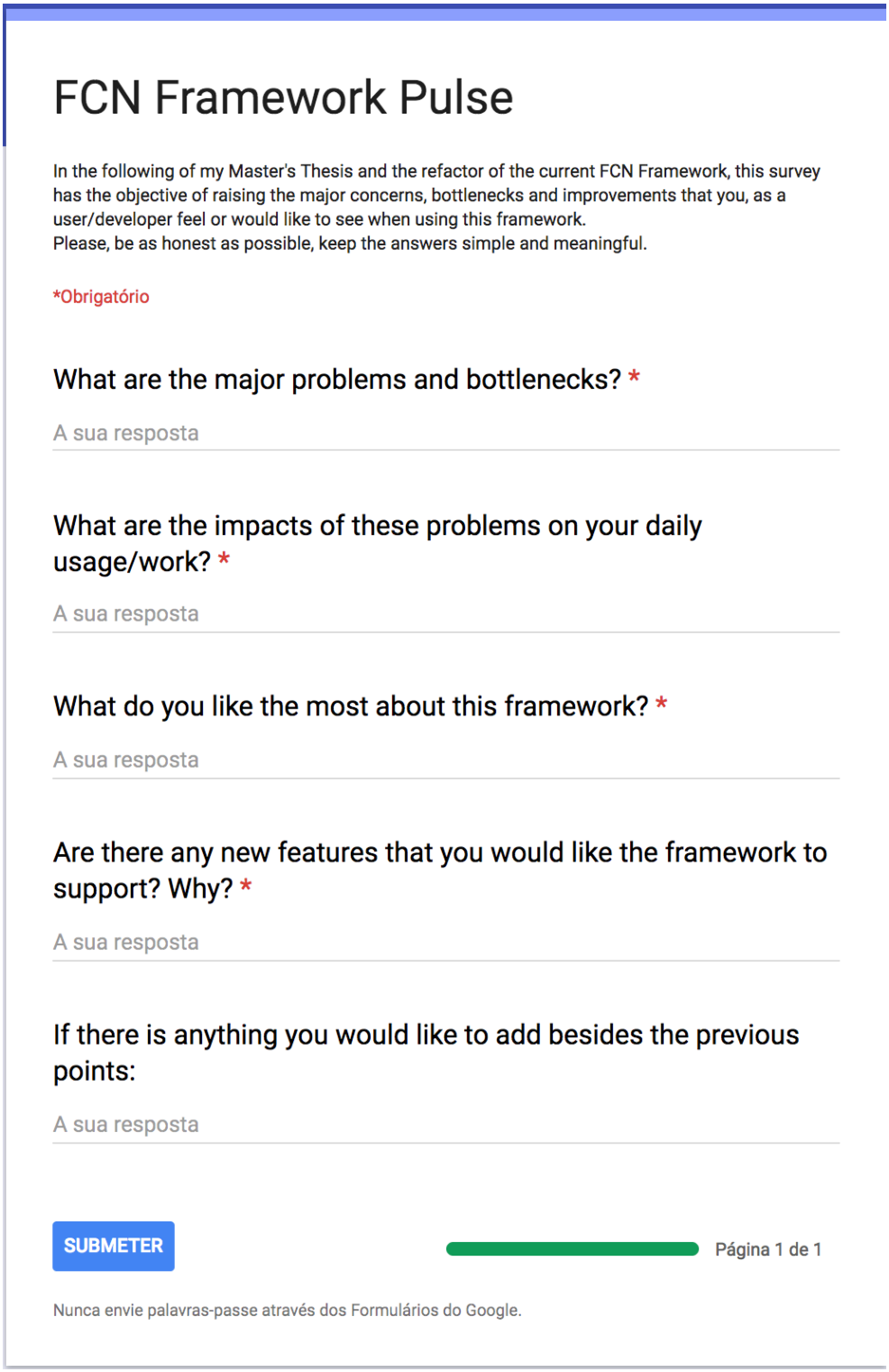
[106] Stephanie, “T Test (Student’s T-Test),” 6 January 2018. [Online]. Available: http://www.statisticshowto.com/probability-and-statistics/t-test/.

[107] R. Lima, “Approach 03 - manifests and artifactory,” Paddy Power Betfair, 11 January 2018. [Online]. Available: https://confluence.app.betfair/display/SportsbookPlatform/Approach+03+-+manifests+and+artifactory.

[108] C. Coles, “Overview of Cloud Market in 2017 and Beyond,” [Online]. Available: https://www.skyhighnetworks.com/cloud-security-blog/microsoft-azure-closes-iaas-adoption-gap-with-amazon-aws/.

Annexes

A.1 FCN Framework Pulse



FCN Framework Pulse

In the following of my Master's Thesis and the refactor of the current FCN Framework, this survey has the objective of raising the major concerns, bottlenecks and improvements that you, as a user/developer feel or would like to see when using this framework.
Please, be as honest as possible, keep the answers simple and meaningful.

*Obrigatório

What are the major problems and bottlenecks? *

A sua resposta

What are the impacts of these problems on your daily usage/work? *

A sua resposta

What do you like the most about this framework? *

A sua resposta

Are there any new features that you would like the framework to support? Why? *

A sua resposta

If there is anything you would like to add besides the previous points:

A sua resposta

SUBMETER

Página 1 de 1

Nunca envie palavras-passe através dos Formulários do Google.

B.1 iap-manifest.js

```
'use strict';

const debug = require('debug')('iap-registration');
const path = require('path');
const fs = require('fs');
const { createApolloFetch } = require('apollo-fetch');

process.env.NODE_TLS_REJECT_UNAUTHORIZED = 0;

const whitelisted = [
    'tla',
    'name',
    'module',
    'fullname',
    'description',
    'version',
    'bootstrap',
    'access',
    'hidden',
    'header',
    'assets',
    'template',
    'hasIapHeader',
    'trackers'
];

/**
 * Parser to insert the assets into the manifest
 * @param {Object} manifest - manifest being built
 * @param {Object} assets - object containing the assets from the
package.json
 * @param {String} tla - string containing the tla
 * @param {String} env - string containing the environment
 * @param {Number} buildNumber - number containing the build number
 * @returns {Object} manifest updated with the assets
 */
const parseAssets = (manifest, assets, tla, env, buildNumber) => {
    Object.keys(assets).forEach(assetKey => {
        manifest[assetKey] = [];
        assets[assetKey].forEach(asset => {
            const assetItem = Object.keys(asset).reduce((item, key) => {
                item[key] = asset[key]
                    .replace('@@BUILD', buildNumber)
                    .replace('@@ENVIRONMENT', env)
                    .replace('@@TLA', tla);
                return item;
            }, {});
            manifest[assetKey].push(assetItem);
        });
    });

    return manifest;
};

/**
 * Calls iap API to upload the manifest
 * @param {Object} manifest - application manifest
```

```
 * @param {String} env - string containing the environment
 * @returns {Promise} response to the api call
 */
const uploadManifest = (manifest, env) => {
    const uri = `https://iap.${env}.internal/api/graphql`;
    const apolloFetch = createApolloFetch({ uri });

    const query =
        'mutation CreateApp($app: AppManifestInput!) {createAppManifest(app:
$app) {tla}}';

    /**
     * Verify if res param has errors
     * @param {Object} res - result of a call from graphql
     */
    const getErrors = res => res && res.errors;

    const variables = { app: manifest };
    return apolloFetch({ query, variables })
        .then(result => {
            const errors = getErrors(result);

            if (errors) {
                debug(
                    'failed to register application: ',
                    JSON.stringify(errors)
                );
                process.exit(1);
            }

            debug(
                `successfully registered application:\n ${JSON.stringify(
                    result
                )}`
            );
            return result;
        })
        .catch(err => {
            debug('failed to register application: ', JSON.stringify(err));
            process.exit(1);
        });
};

const generateManifest = (tla, env, buildNumber, manifestFile) => {
    if (!buildNumber) {
        throw new Error('Build Number is mandatory');
    }
    const cwd = process.cwd();

    let pkg;
    const packagePath = manifestFile
        ? manifestFile
        : path.join(cwd, 'metadata.json');
    try {
        pkg = fs.existsSync(packagePath)
            ? require(packagePath)
            : require(path.join(cwd, 'package.json'));
    } catch (error) {
        throw new Error(`No ${manifestFile} found in ${packagePath}`);
```

```
    }

    const finalManifest = Object.keys(pkg).reduce((manifest, key) => {
        if (whitelisted.indexOf(key) === -1 || pkg[key] == undefined) {
            return manifest;
        }

        if (key === 'assets') {
            return parseAssets(manifest, pkg[key], tla, env, buildNumber);
        }

        manifest[key] = pkg[key];

        return manifest;
    }, {});
    finalManifest.buildNumber = buildNumber;

    debug(`iap registration payload:\n ${JSON.stringify(finalManifest)}`);

    return finalManifest;
};

module.exports = {
    publishManifest: (tla, env, buildNumber, manifestFile) =>
        uploadManifest(
            generateManifest(tla, env, buildNumber, manifestFile),
            env
        ),

    exportManifest: (tla, env, buildNumber, packageFile) =>
        JSON.stringify(
            generateManifest(tla, env, buildNumber, packageFile),
            null,
            2
        )
};
```

C.1 iap-header.tsx

```
import {
    Component,
    Event,
    EventEmitter,
    Listen,
    Prop,
    State
} from '@stencil/core';

import 'iap-icons';
import 'iap-components';

declare global {
    interface Window {
        iap: any;
    }

    interface Object {
        values: Function;
    }
}

/**
 * IAP Header Component
 *
 * @export Component iap-header
 * @class IapHeader
 */
@Component({
    tag: 'iap-header',
    styleUrl: 'iap-header.css'
})
export class IapHeader {
    /**
     * Emitted event when the menu icon is clicked
     * @type {EventEmitter} iapMenuClicked
     */
    @Event()
    iapMenuClicked: EventEmitter;

    /**
     * Label to be shown in the header
     * @type {string} headerLabel
     */
    @Prop()
    headerLabel = 'Internal Applications Platform';

    /**
     * State property that defines the menu visibility
     * @type {boolean} menuVisible
     */
    @State()
    menuVisible: Boolean = false;

    /**
     * Listener event when a card is clicked
     */
```

```
@Listen('window:iapCardClicked')
redirectToApp(event: CustomEvent) {
    const app = this.getClickedApp(event);
    window.location.href = this.createUrl(app);
}

/**
 * Helper to get clicked app from menuContent
 * @returns {any} appEntry
 */
getClickedApp(event) {
    return this.menuContent.find(
        (entry: any) => entry.tla === event.detail
    );
}

/**
 * Helper to create url of clicked app
 * @returns {string} url
 */
createUrl(app) {
    const tla = app.tla.toLowerCase();
    const root = app.bootstrap.root;
    return `${window.location.origin}/${tla}#${root}`;
}

/**
 * Menu content to display in iap-apps-menu
 */
menuContent: Array<Object>;

/**
 * Component Lifecycle Event
 */
componentWillLoad = () => {
    const rawMenu = window.iap.ConfigService.get({
        tla: 'iap',
        prop: 'apps'
    });

    this.menuVisible = false;
    this.menuContent = this.translateMenu(
        Object.values(JSON.parse(rawMenu))
    );
};

/**
 * Translate menu entries into iap-apps-menu format
 */
translateMenu = rawMenu =>
    rawMenu.map(entry => ({
        ...entry,
        tla: entry.tla.toUpperCase(),
        name: entry.fullname
    }));

/**
 * Function that handles the menu clicking action
 */
onMenuClick = () => {
```

```
        this.menuVisible = !this.menuVisible;
        this.iapMenuClicked.emit(this.menuVisible.toString());
    };

    /**
     * Renders the iap-header element
     * @returns {JSX.Element}
     */
    render = (): JSX.Element => {
        return (
            <div class="iap-header" id="iap-header">
                <nav class="iap-nav">
                    <button class="app-selector"
onClick={this.onMenuClick}>
                        <burger-menu color="#ffc835" width="32" height="32"
/>
                    </button>
                    <a href="/" class="iap-logo">
                        {this.headerLabel}
                    </a>
                    {this.menuVisible ? <slot /> : null}
                </nav>
                <div class="options-row">
                    <iap-apps-menu
                        fullScreen
                        card-list={JSON.stringify(this.menuContent)}
                    />
                </div>
            </div>
        );
    };
}
```

C.2 iap-header.css

```
.iap-header {
  display: block;
  align-content: center;
  align-items: center;
  height: 72px;
}
.iap-nav {
  background-color: var(--outter-space);
  color: var(--iron);
  display: flex;
  flex-direction: row;
  flex-wrap: nowrap;
  height: 72px;
  align-content: center;
  align-items: center;
}

.app-selector {
  transition: all, 0.3s, ease-in-out, 0s;
  border: 0;
  background-color: transparent;
  display: flex;
  height: 72px;
  justify-content: center;
  outline: 0;
  padding: 0 20px;
  align-items: center;
  min-width: 72px;
  cursor: pointer;
}

.app-selector:hover,
.app-selector.selected {
  transition: all, 0.3s, ease-in-out, 0s;
  background-color: var(--brown);
}

.iap-logo {
  cursor: pointer;
  text-decoration: none;
  color: var(--sunglow);
  font-size: 32px;
  font-weight: bold;
  padding-left: 10px;
}

.options-row {
  display: flex;
  position: relative;
  flex-direction: row;
  align-items: flex-end;
  width: 50%;
}
```

D.1 legacy.mustache

```
<!doctype html>
<html class="no-js">
    <head>
        <meta charset="utf-8">
        <title>Internal Applications Platform</title>
        <meta name="description" content="">
        <meta name="viewport" content="width=device-width">

        <link rel="stylesheet" type="text/css"
href="/core/styles/app.{{fcnBuildNumber}}.css">
        {{#styles}}
        <link rel="{{rel}}" type="text/css" href="{{href}}">
        {{/styles}}
    </head>
    <body ng-app="fcn">
        <fcn-alerts></fcn-alerts>

        <div ui-view="headerView"></div>

        <aside ui-view="asideView"></aside>

        <div
            id="root"
            ui-view="contentView"
            class="content"
            ng-class="{'aside-visible': data.isAsideVisible, 'aside-
hidden': !data.isAsideVisible}"
        ></div>

        <fcn-modals></fcn-modals>

        <script>
            window.ga = window.ga || function() {
                (ga.q = ga.q || []).push(arguments);
            };
            ga.l = +new Date;
        </script>
        <script async src="https://www.google-
analytics.com/analytics.js"></script>

        <script src="/core/scripts/vendor.{{fcnBuildNumber}}.js"></script>
        <script
src="/core/scripts/environment.{{fcnBuildNumber}}.js"></script>

        <script src="/iap/scripts/iap-
core.js?v={{coreBuildNumber}}"></script>
        <script>window.iap.setup({{{iapConfig}}});</script>

        <script>
        angular
            .module("fcn", [
                "ui.router",
                "fcn.core.templates",
                "fcn.core.config",
                "fcn.core.services",
                "fcn.core.storage",
                "fcn.core.modals",
```

```
                "{{ module }}"
            ])
            .constant('fcn.apps', {{{ apps }}})
            .constant('fcn.registry.active', "{{{ active }}}")
            .constant('fcn.registry', {{{ fcn }}});
        </script>
        <script src="/core/scripts/app.{{fcnBuildNumber}}.js"></script>
        {{#scripts}}
        <script src="{{src}}"></script>
        {{/scripts}}
    </body>
</html>
```

D.2 agnostic.mustache

```
<!doctype html>
<html class="no-js">
    <head>
        <meta charset="utf-8">
        <title>Internal Applications Platform</title>
        <meta name="description" content="">
        <meta name="viewport" content="width=device-width">

        {{#hasIapHeader}}
        <link rel="stylesheet" type="text/css"
href="/iap/ui/styles/base.css?v={{coreBuildNumber}}">
        <link rel="stylesheet" type="text/css"
href="/iap/ui/themes/default.css?v={{coreBuildNumber}}">
        <link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Ubuntu">
        {{/hasIapHeader}}
        {{#styles}}
        <link rel="{{rel}}" type="text/css" href="{{href}}">
        {{/styles}}
    </head>
    <body>
        {{#hasIapHeader}}
        <iap-header header-label="{{headerLabel}}"></iap-header>
        {{/hasIapHeader}}
        <div id="root"></div>

        {{#hasGoogleAnalytics}}
        {{> google-analytics}}
        {{/hasGoogleAnalytics}}

        <script src="/iap/scripts/iap-
core.js?v={{coreBuildNumber}}"></script>
        <script>window.iap.setup({{{iapConfig}}})</script>

        {{#hasIapHeader}}
        <script src="/iap/ui/components/iap-
skeleton.js?v={{coreBuildNumber}}"></script>
        {{/hasIapHeader}}

        {{#scripts}}
        <script src="{{src}}"></script>
        {{/scripts}}
    </body>
</html>
```

E.1 logger-factory.js

```
import LoggerClient from './client';
import Logger from './logger';

/**
 * Object where the loggers are stored by type
 * @type {Object.<string, Object>}
 */
const loggers = {};

/**
 * Returns the logger strand endpoint
 * @returns {string} logging endpoint
 */
const getDefaultUrl = () =>
    window.iap.ConfigService.get({ tla: 'iap', prop: 'LOGGING_ENDPOINT' });

/**
 * A factory that creates loggers.
 * The loggers are cached so, the 1st time you call the factory for a given
type,
 * you'll have to provide all initialisation parameters.
 * Further calls only need the name.
 * Works as a singleton, therefore you don't have to pass around the
instance of your tracker.
 */
class LoggerFactory {
    /**
     * Instantiate
     */
    constructor() {
        this.isConfigured = false;
    }

    /**
     * Sets the required configuration for the logger to work properly
     * @param {Object} payload - initialization object
     * @param {String} [payload.tla] - application tla
     * @param {String} [payload.url] - url to the logging strand
     * @param {Boolean} [payload.shouldOutputToConsole] - should output to
console
     */
    configure({ tla, url = getDefaultUrl(), shouldOutputToConsole = false })
{
        if (!tla) {
            throw new Error('LoggerFactory must be configured with a tla');
        }

        loggers = {};

        this.tla = tla;
        this.shouldOutputToConsole = shouldOutputToConsole;
        this.client = new LoggerClient(url);

        this.isConfigured = true;
    }

    /**
```

```
     * Get or create logger
     * @param {String} name - logger name
     * @param {String | Number} level - logging level
     * @returns {Logger} instance of Logger
     */
    getLogger(name, level) {
        if (!this.isConfigured) {
            throw new Error('LoggerFactory has not been properly
configured');
        }

        if (!loggers[name]) {
            loggers[name] = new Logger({
                tla: this.tla,
                client: this.client,
                shouldOutputToConsole: this.shouldOutputToConsole,
                level,
                name
            });
        }

        return loggers[name];
    }
}

export default new LoggerFactory();
```

E.2 client.js

```
/**
 * Client that sends logs to the strand
 */
class LoggerClient {
    /**
     * Instantiates a Logger Client
     * @param {string} url - strand url
     */
    constructor(url) {
        this.url = url;
    }

    /**
     * Send logs to the strand
     * @param {Object} entry - message object
     * @returns {Promise} http promise
     */
    send(entry) {
        return window.fetch(this.url, {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({ entries: [entry] })
        });
    }
}

export default LoggerClient;
```

F.1 IAP Pulse – Maintainers Team

IAP Pulse

*Required

IAP Documentation Support

All questions are in comparison with FCN.

IAP's documentation is easier to find. *

	1	2	3	4	
Strongly Disagree	○	○	○	○	Strongly Agree

IAP's documentation is more clear and meaningful. *

	1	2	3	4	
Strongly Disagree	○	○	○	○	Strongly Agree

IAP's documentation covers all meaningful areas. *

	1	2	3	4	
Strongly Disagree	○	○	○	○	Strongly Agree

IAP's documentation increased my engagement with the framework. *

	1	2	3	4	
Strongly Disagree	○	○	○	○	Strongly Agree

IAP Documentation Support

All questions are in comparison with FCN.

IAP's documentation is easier to find. *

	1	2	3	4	
Strongly Disagree	○	○	○	○	Strongly Agree

IAP's documentation is more clear and meaningful. *

	1	2	3	4	
Strongly Disagree	○	○	○	○	Strongly Agree

IAP's documentation covers all meaningful areas. *

	1	2	3	4	
Strongly Disagree	○	○	○	○	Strongly Agree

IAP's documentation increased my engagement with the framework. *

	1	2	3	4	
Strongly Disagree	○	○	○	○	Strongly Agree

IAP Learning Curve

All questions are in comparison with FCN.

I am happier with IAP. *

	1	2	3	4	
Strongly Disagree	○	○	○	○	Strongly Agree

I find IAP easier to use. *

	1	2	3	4	
Strongly Disagree	○	○	○	○	Strongly Agree

IAP Maintainability

All questions are in comparison with FCN.

It is easier to add a feature to IAP.

	1	2	3	4	
Strongly Disagree	○	○	○	○	Strongly Agree

It is easier to debug in IAP.

	1	2	3	4	
Strongly Disagree	○	○	○	○	Strongly Agree

F.2 IAP Pulse – IAP Only Developers

IAP Documentation Support

Description (optional)

IAP's documentation is easy to find. *

	1	2	3	4	
Strongly Disagree	○	○	○	○	Strongly Agree

IAP's documentation is clear and meaningful. *

	1	2	3	4	
Strongly Disagree	○	○	○	○	Strongly Agree

IAP's documentation covers all meaningful areas. *

	1	2	3	4	
Strongly Disagree	○	○	○	○	Strongly Agree

IAP's documentation increased my engagement with the framework. *

	1	2	3	4	
Strongly Disagree	○	○	○	○	Strongly Agree

IAP Learning Curve

Description (optional)

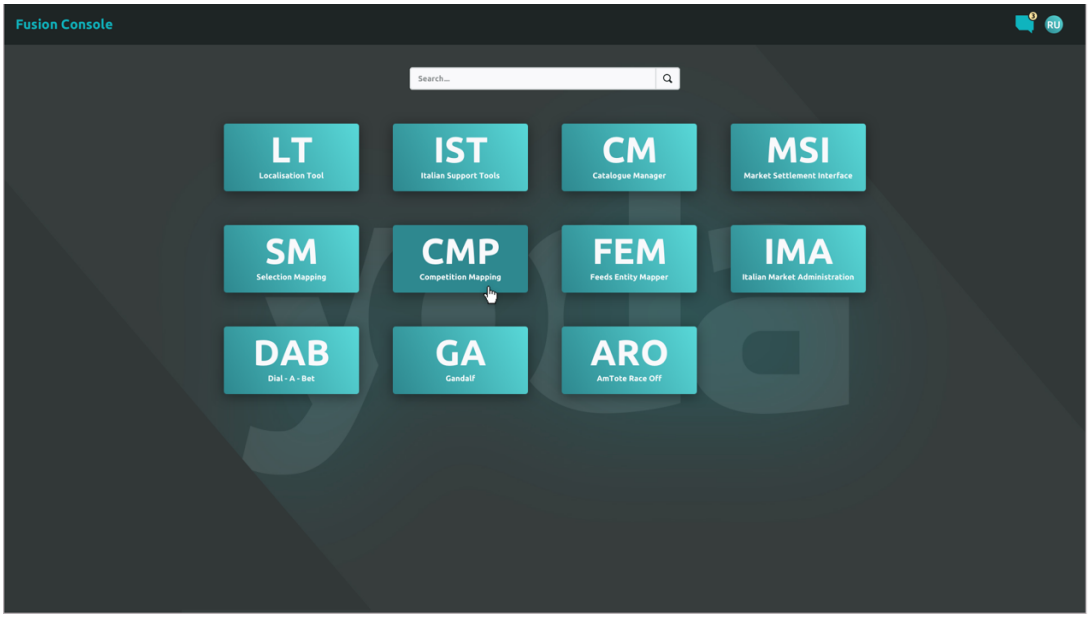
I am happy with IAP. *

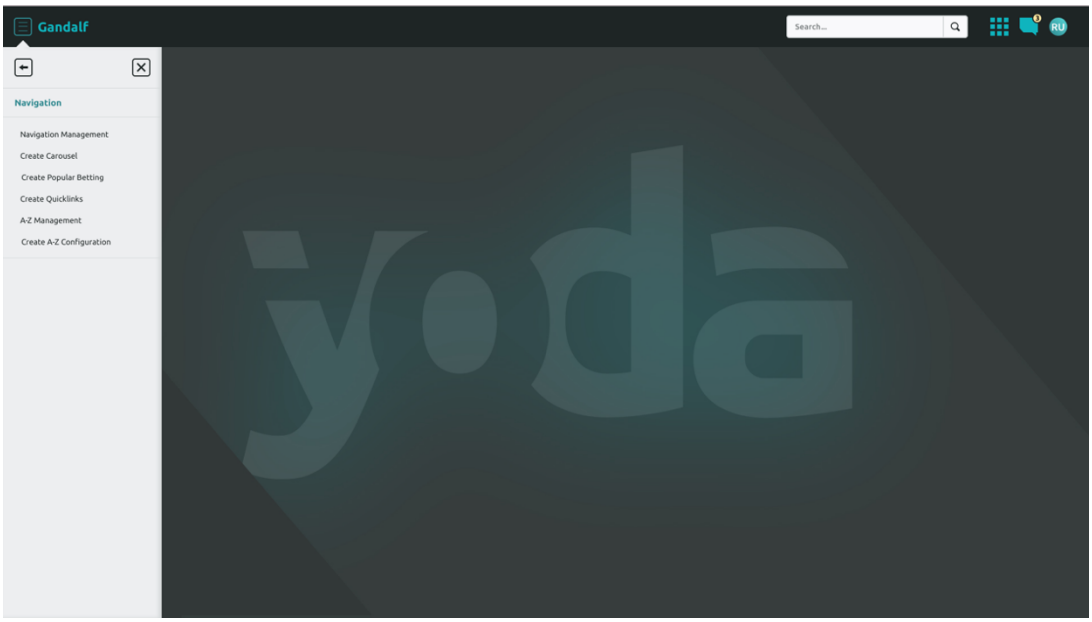
	1	2	3	4	
Strongly Disagree	○	○	○	○	Strongly Agree

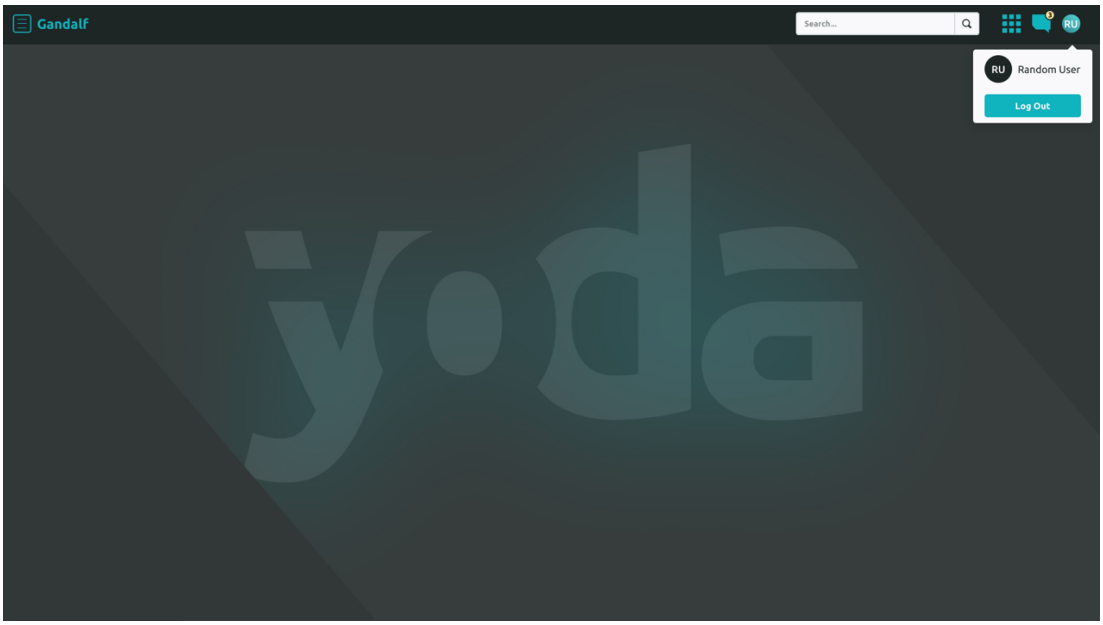
I find IAP easy to use. *

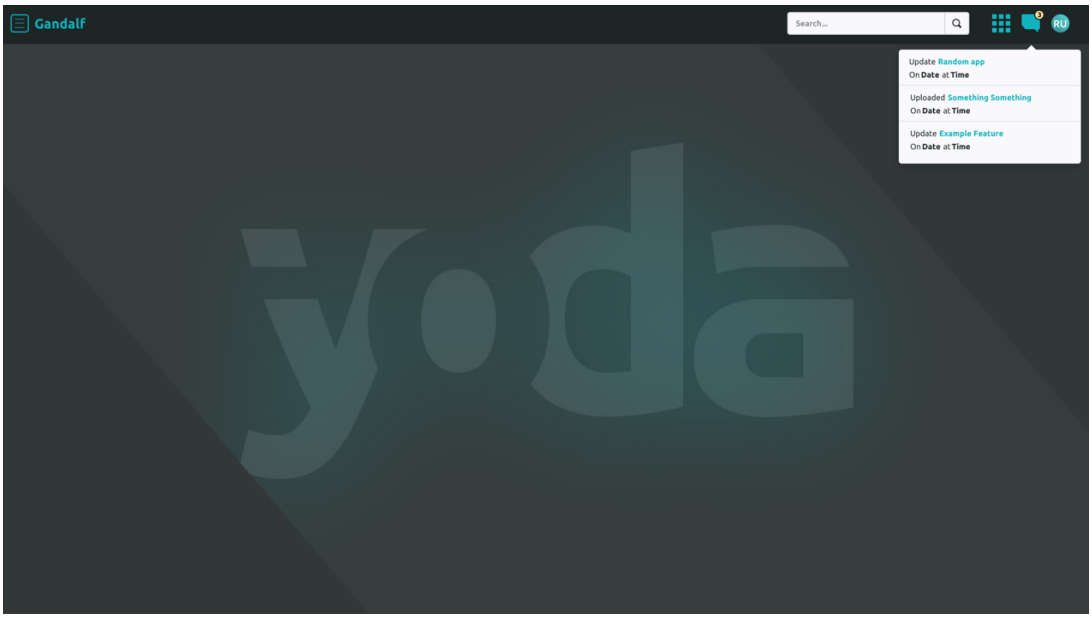
	1	2	3	4	
Strongly Disagree	○	○	○	○	Strongly Agree

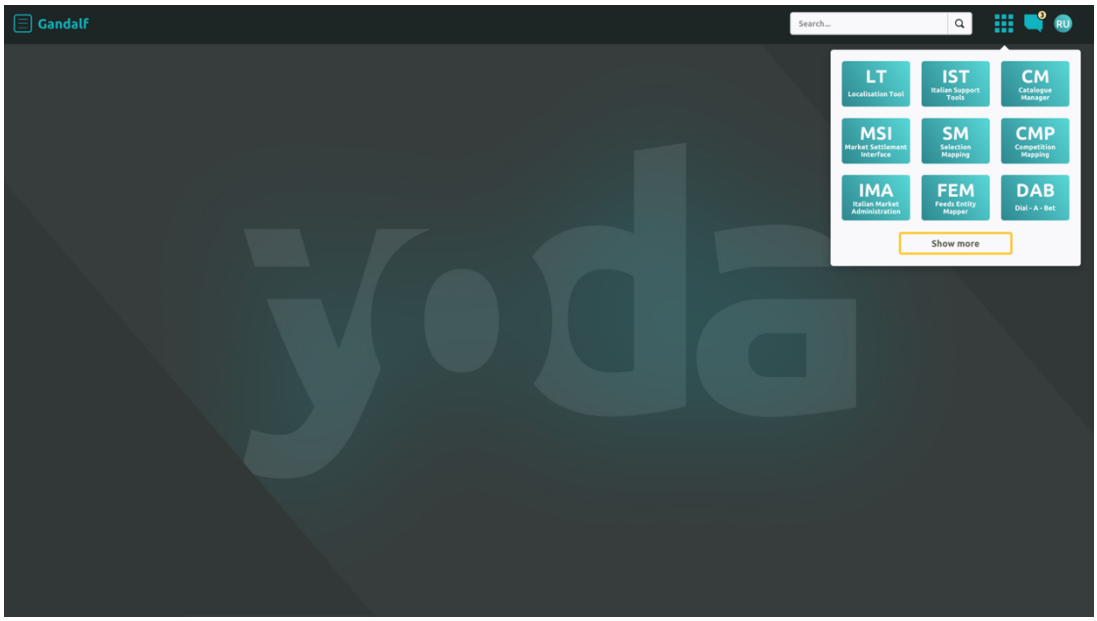
G.1 IAP Design

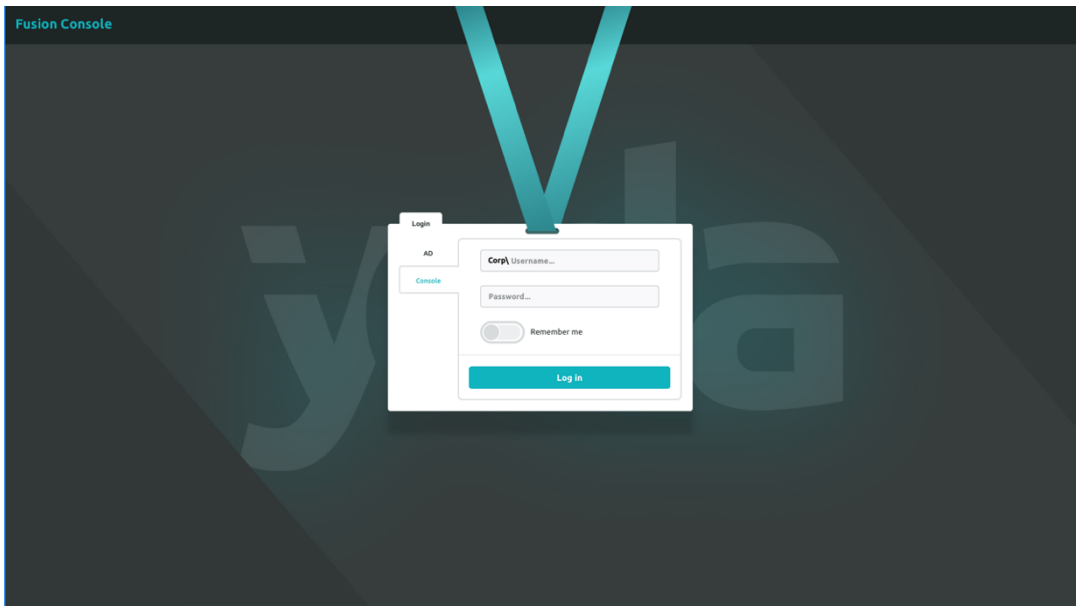





Gandalf
Search...
RU
Random User
Log Out


Gandalf
Search...
RU
Update Random app
On Date at Time
Uploaded Something Something
On Date at Time
Update Example Feature
On Date at Time


Gandalf
Search...
RU
LT
Localisation Tool
IST
Italian Support Tools
CM
Catalogue Manager
MSI
Market Settlement Interface
SM
Selection Mapping
CMP
Competition Mapping
IMA
Italian Market Administration
FEM
Feeds Entity Mapper
DAB
Dial - A - Bet
Show more


Fusion Console
Login
AD
Console
Corp\ Username...
Password...
Remember me
Log in