High School of Applied Sciences and Management

University SESAME

Internship report:

[LONGVIEW PLATFORM]

A FRAMEWORKE: DASHBOARD GENERATOR

Med Amine Zaouia

Supervisor: **Ahmed Charfeddine**

Pedagogic advisor: **Chiheb Ben Ncir**

At Linedata

Academic Year: **2017/2018**

Table of contents

List of tables

List of figures

GENERAL INTRODUCTION

As the volume of data has grown and the methods of analysing them have improved, organizations have been integrating data more firmly into the decision-making process. However, increasing numbers of traditional and non-traditional data sources are inundating companies with data in volumes and types they may not have seen before. Companies are finding an increasing gap between the acquisition of data and their meaningful use. Companies have focused extensively on the opportunities and challenges presented by “big data”, recognizing that leveraging it to gain competitive advantage can yield significant payoffs. Research from Sloan School of Business indicates that companies that engage in “data-driven decision-making” enjoy a 5 to 6% increase in output and productivity over firms that do not. These results are replicated in other metrics, such as asset utilisation, return on equity and market value.

Executives love dashboards, and why wouldn’t they? Single-screen “snapshots” of operational processes, marketing metrics, and key performance indicators (KPIs) can be visually elegant and intuitive. They show just-in-time views of what’s working and what isn’t, no need to wait for weekly or monthly reports from a centralized data centre. A quick scan of a dashboard gives frontline managers transparency and, ideally, the opportunity to make rapid adjustments.

Organizations need dashboards because everyone from senior management to “front-line” employees are inundated with data in the form of reports, memos, emails and phone calls, among other electronic and non-electronic formats. This data needs to be synthesized into the critical information that will enable them to make decisions, take actions or delegate work in a timely way.

An effective dashboard should allow users to change the type of information they see, or how it is displayed, without help from IT as business challenges evolve. Many solutions exist to automatize the creation of dashboards, they allow users to create extremely powerful dashboards at a glance, without having any programming skills.

Chapter 1: Overview

1.1 Context of the internship

This internship is part of the Last year of the education of engineers at the High School of Applied Sciences and Management (University SESAME). I was led to make a 7 months experience in Linedata.

1.2 Presentation of Linedata

1.2.1 Generalities

Linedata was founded in January 1998 as the result of a merger of three firms: GSI Division, Line Data and BDB Participation. A leveraged management buyout backed by AXA Private Equity Fund (APEF) gave employees at GSI Division majority control over their company in December 1997. The subsequent acquisition of Linedata and BDB Participation led to the creation of a new company: Linedata. New, but not a startup, the new corporate entity benefited at the outset from the best management, technology and operational expertise drawn from its three component companies.
Linedata achieved a major milestone in May 2000 when it was admitted to the Nouveau Marché on the Paris Stock exchange. Linedata (Ticker LIN.FP on Bloomberg) is now quoted on Euronext Paris and has become one of the most respected and independent technology firms in its field.

Figure 1 Linedata across the world

1.2.2 Awards of Linedata

Linedata is a global leader recognized worldwide for its breakthrough solutions. These awards demonstrate its commitment to deliver superior solutions driven by long term relationships with the clients:

27 Apr 2017: Linedata recognized as **"Best Technology Provider"**.

19 Mar 2017: Linedata recognized **"National Champion"** by the European Business Awards.

24 Feb 2017: Linedata Wins Award for **"Best PMS"**.

25 Oct 2016: Linedata recognized as **"Best Portfolio Management Software Provider"**.

06 May 2015: Linedata recognized as the **"Best portfolio management software provider"**.

13 Mar 2015: Linedata recognized in the **Global Custodian** Awards for Excellence, Europe, 2015.

11 Mar 2015: Linedata named **Best Trading Platform – Client Service** by Wall Street Letter for Third Consecutive Year.

16 Dec 2014: Linedata wins **Best Fund Accounting Technology Platform in Global Custodian** Awards.

10 Dec 2014: Linedata wins **Best Disruptive Technology in Global Custodian North America** Awards.

04 Dec 2014: Linedata honoured by The TRADE with **"Outstanding Technology Vendor"** Hall of Fame award.

1.2.3 Sales revenue

Linedata realized a huge increase of sales revenue, it was **158.0 M$** in **2014** and it reaches **172.3 M$** in **2015**.

1.2.4 Products of Linedata:

Linedata have 26 products for its clients: Linedata Admin Edge, Linedata Bdb,Linedata Capitalstream, Linedata Chorus, Linedata Compliance, Linedata Disclosure Manager, Linedata Ekip 360, Linedata Global Hedge, Linedata I-BOR, Linedata I-CIPS, Linedata Icon, Linedata Icon Retail, Linedata Longview, Linedata Longview for Wealth Managers, Linedata Lynx, Linedata Master I, Linedata Mfact, Linedata Mshare, Linedata Navquest, Linedata Noee, Linedata Profinance, Linedata Reporting, Linedata Star, Linedata Trader+, Linedata Uniloan et Linedata Webpass.

1.2.4 Longview Linedata

Award-winning order management solution provides a highly configurable set of tools to streamline the portfolio management, compliance, trading, risk, and decision making. Real-time insights across the investment lifecycle help quickly identify and respond to opportunities. Single, modular platform reduces cost and operational risk, driving efficiencies. Access a scalable, flexible configuration, asset specific workflows and an interface designed for various roles across the front office. It helps clients optimize portfolios, seamlessly integrating hundreds of trading destinations, and supporting them with agile technology that evolves as their business does. [1]

1.3 Project Overview

1.3.1 Project Background

This internship aimed to develop a platform, its main purposes are creating dashboards and make sure that this platform is easy to maintain, update and add on to by the developer.

1.3.2 Project Presentation

Before describing the work to do, it would be useful to present some concepts and some Technologies. This will properly introduce the detailed work.

a- Dashboards

A dashboard is a visual display of the most important information needed to achieve one or more objectives; consolidated and arranged on a single screen so the information can be monitored at a glance. [2]

b- KPI

A Key Performance Indicator is a measurable value that demonstrates how effectively a company is achieving key business objectives. Organizations use KPIs at multiple levels to evaluate their success at reaching targets. [3]

c- Dashboard Generator

It is an application (in our case it is a framework) that allows the users to create and generate their own dashboards.

d- Why dashboards

Making valuable decisions has always been one of the most important things in business. A dashboard is a tool that gives business leaders valuable information, and it provides a quick outlook of all the most valuable numbers, which allows business leaders to perform an accurate business analysis.

1.3.3 Work goals

Linedata Company is in dire need to a platform dedicated to dashboard creators and developers that automatizes the implementation and the customization of dashboards. The aim of our project is to implement a dynamic platform which allows to create dashboards, implement new widget types, and add pages and widgets dynamically to a dashboard.
This platform is useful for both developers and clients, using this platform developers can add customized widget types to the widget library, make the branding of the dashboard after "forking" shell code, while the end user (which can be a developer) adds pages to the dashboard, and then adds widgets to those pages, representing his data in a structured way.

We named our application ‘Longview Platform’, it is an interactive web application for building real-time dynamic dashboards.

- It allows users to customise its dashboard, add widgets, connect to data sources or APIs, automate data, and then manipulate and visualize the data.
- It allows the developer to add new types of widgets to the code in a flexible and easy way.

1.4 The used methodology

Having decided on a general approach to managing the project, a decision must be made on how best to execute the project plan.

This part will cover the details explanation of methodology that is being used to make this project complete and working well.

As we're working in a team of developers, adopting a methodology is not a choice as it helps us achieve our project goals, it makes the development process more determined and clear by adapting to the project specifications. Here we are going to present some known methodologies and compare them in order to determine which one to choose according to our project.

1.4.1 Existent methodologies

When it comes to software development, there are two well-known methodologies:

- Waterfall
- Agile

The **waterfall** model is a sequential (non-iterative) design process, used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of conception, initiation, analysis, design, construction, testing, production, implementation and maintenance. This also means that the client is not allowed to change or add constraints when the product is being developed.

On the other hand, **agile** method is not about the sequential procedure. It follows an incremental approach. The project starts from a simple design template and evaluates as more constraints and features are introduced. Requirements are not fixed before starting, they are to be discovered in the development process. This allows developers to go back and fix Bugs or add functionalities. As a methodology it is first introduced as an alternative to the waterfall thanks to its flexibility

1.4.2 Scrum

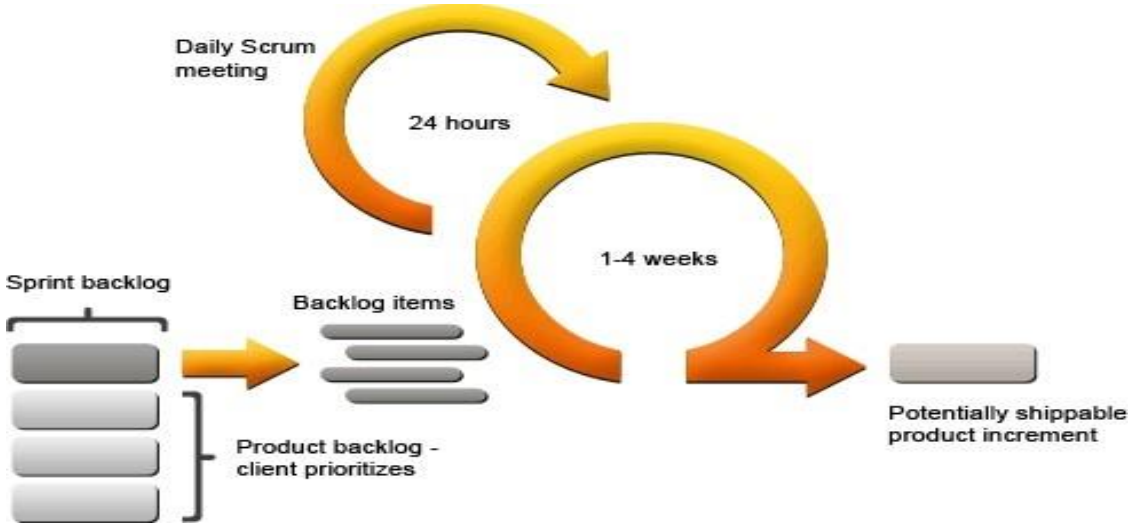

Figure 2: Scrum diagram

Scrum is an agile software development process, best suited for projects with rapidly changing requirements.
This part briefly covers the scrum theory and how we used Scrum ourselves in the project. [4]

1.4.3 Scrum Roles

Product Owner - acts as a spokesman for the customer, and defines features of the product based on each Backlog item or each specific request of the customer. He should prioritize these features according to the market value, decide on a release date for the product, and is responsible for the profitability of the product. The product owner should also adjust the contents of the features and their priority after every Sprint, and decide if what has been produced is acceptable.

Scrum Master - responsible for making sure a Scrum team lives by the values and practices of Scrum, and for removing any impediments to the progress of the team. As such, he should shield the team from external interferences, and ensure that the Scrum process is followed, including issuing invitations to the daily Scrum meetings, sprint reviews and the sprint planning meetings.
Scrum Team - the group of people developing the product. There is no personal responsibility in Scrum, the whole team fails or succeeds as a single entity.

1.4.4 Sprint Planning Meeting

The sprint planning meeting is attended by the product owner, the Scrum master, the team and any interested stakeholders. During the meeting, the product owner and the team will go through the vision, the roadmap, the release plan, and the Product backlog to develop a detailed plan for the upcoming sprint.

Together, the product owner and the team define a sprint goal, which is a short description of what should be completed at the end of the sprint. The success of the sprint will later be assessed during the sprint review meeting against the sprint goal, rather than against each specific item selected from the product backlog.

Since the product owner decides the scope of a feature, the team usually requires the product owner to answer all of their questions to estimate the time it will take to finish a backlog item.

The team needs to be well prepared for this meeting. They should dedicate some time during a sprint to think ahead how the upcoming backlog items could be designed and implemented. Especially if the team is inexperienced and cannot produce a confident estimate of the effort required to complete a backlog item on their own and needs some additional information.

The team decides how much work they can successfully take into the sprint based on the team size, available hours, and the estimated level of productivity. When the team has selected and committed to deliver a specific set of features from the highest priority items in the backlog, the Scrum master leads the team in a planning session to break these features down into Tasks, and form the Sprint backlog.

1.4.5 Daily Scrum Meeting

The daily Scrum meeting is a short meeting where the team members inform each other of what they have done, what they plan to do and if they have any impediments. This is to make sure that the whole team knows what is happening and to make sure none is stuck alone with a single problem for several days. It is timeboxed to 15 minutes to ensure that the team spends more time developing than talking about developing.

1.4.6 Sprint Review Meeting

At the end of each sprint, a sprint review meeting is held. The first half of the meeting is set aside to demonstrate the potentially shippable prototype to the product owner that has been developed during the sprint. The product owner leads this part of the meeting and invites all interested stakeholders to attend. The product owner determines which items on the product backlog have been completed in the sprint. It is usually not allowed to use PowerPoint slides and spend more than two hours preparing for the demonstration.

1.4.7 Retrospective

After the sprint review meeting the Scrum team goes through a retrospective of the sprint with the Scrum master. The team assesses the way they worked together in the sprint and identifies things that went well and should be encouraged in the future, as well as things that could be improved. The results of this should be visible to the team during the following sprints. It is vital, therefore, that the feedback received at the retrospective be followed up on. Otherwise, the team may quickly see it as a waste of time.

1.5 Versioning and code management

• **Git**: git is a tool for versioning and managing codes, it's well known for its scale and easy to use API,

• **GitHub**: GitHub is a managed Git server where users can use it to create, clone and push git project.

The simplified workflow used when implementing or modifying a feature is the following: A new branch is created for that feature, and code changes related to it are committed (saved) there. Once done a pull request is created, which means that the feature is ready to be merged with the master branch and to be deployed. Every pull request passes through two phases:

• **Code review**: Analysing the code changes,

• **Test**: Testing if the changes will break other components. If all tests pass and everything is OK, the pull request is merged with the master branch and later deployed.

Conclusion

In this chapter we specify the goals of our project, a presentation of Linedata and the methodology we are going to be used during this internship.

Chapter 2: Sprint 0: Requirements analysis and specification

Introduction

The very first sprint, sprint 0, was a very informal sprint which focused mostly on writing reports and planning and preparing the sprints ahead. This sprint took about a month. Work in this sprint started very slowly since we spent a lot of time learning the new technologies that we will be using and developing small side projects to prepare for the main project.
The successful completion of an application depends on understanding and analysis its functionality and its requirement. In this chapter we describe the functional requirements and non-functional implementation and then, for better clarity, the semi-formal language UML will be used to represent these requirements.

2.1 Existing solutions

2.1.1 Klipfolio

Klipfolio offers an online dashboard platform for building real-time business dashboards. It allows business users to connect to many data services, automate data retrieval, and then manipulate and visualize the data. Klipfolio uses a schema-less architecture that allows non-technical end users to more easily connect to data sources, and separates data from presentation to more efficiently use and reuse data sources throughout the platform.

Klipfolio has built-in formula editing, allowing end-users to transform, combine, slice, and filter any data before visualizing it. Users are able to access the dashboard from their desktop, tablet, TV, and mobile phone, and share it with colleagues by granting access to the dashboard, or by scheduling email reports. [5]

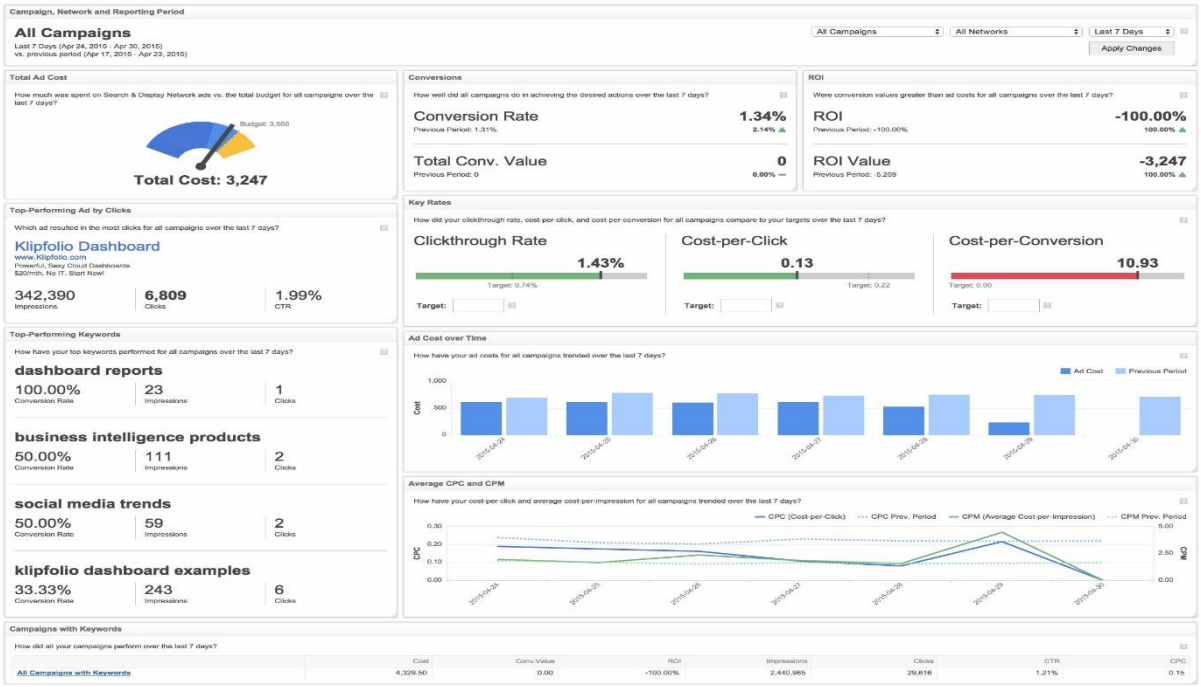

Figure 3 Klipfolio

2.1.2 Dundas BI

Dundas BI is an enterprise-ready Business Intelligence (BI) platform for creating and viewing interactive dashboards, reports, scorecards and more. Users can deploy Dundas BI as the central data portal for their organization, or integrate it into an existing website as part of a custom BI solution. [6]

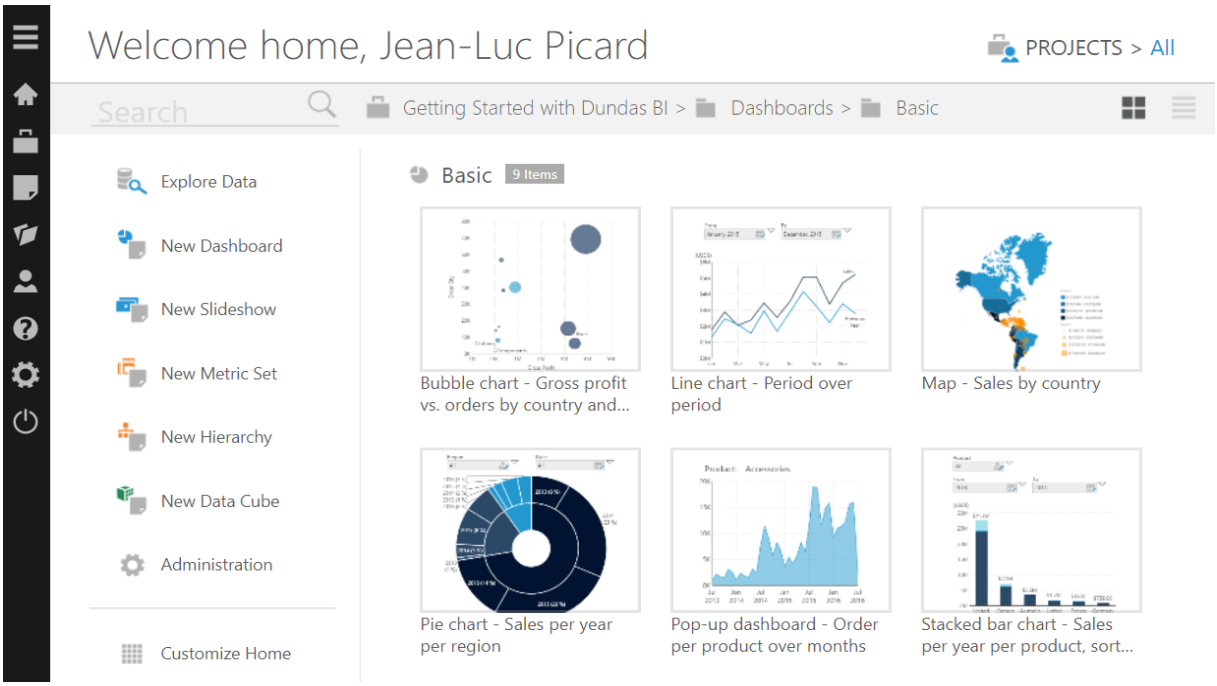

Figure 4 Dundas BI

2.1.3 Limitations and problems

In this previous examples, the main and only user is the client (the end user), in this case he/she can only use the functionalities provided by the application; but in some cases the widgets provided are not what the client wants or they look for different widgets. And this changes require the source code of the application to modify and customise it to response the client's needs and this modifications can be made by developers.

2.2 Functional requirements

The typical users of the Longview Platform are:

- Developers (in this case, they will be using the shell of the project).
- End user (this type of clients will be using the final version of the application).

The first user of this platform will be the developer. In this case we are focusing on what the developer may need to easily create a dashboard by giving him the best atmosphere.
So, to facilitate the developer's work, our project shall:

- Allow to download the project (the shell) via GitHub.
- Allow to install the project (the shell) as a package.
- Allow the developers to easily implement new developed widgets.
- Allow the developers to easily customize the brand and change the template.

But the client's needs should be taken in consideration during the developing of the framework because no developer will use a framework that is not a friendly user application. So the application in the final phase is used by the end user (the client).
So in this case the Longview platform shall:

- Allow the user to login.
- Be possible to modify the Layout.
- Allow the user to load the previous dashboard created.
- Allow the user to create one or more dashboard and manage them.
- Allow the user to add, remove and customize widgets.
- Be possible to drag and drop the widgets inside the dashboard.
- Allow the user to connect to a data source.
- Allow the user to choose a Key Performance Indicator for each widget.
- Be possible to download the results of a query as an Excel-file, XML, CSV, PDF...

2.3 Non-functional requirements

- **Reliability: T**he platform should visualise the right data at the right moment.
- **Dynamic:** refresh rates of data is in real-time (or near real-time).

- **Concise and responsive:** Information must be displayed in the briefest or concise way to enable easy navigation and reading. Moreover, the dashboard must be quick and smooth as well. It must be able to enable access to huge amounts of information effortlessly without difficulties in navigation.
- **Flexible:** it is important to create the platform in such a way that it can be changed or updated according to the changes in the business's requirements.
- **Clear communication and easy exporting:** A stunning dashboard is one which communicates clearly and sends the message in the simplest possible way.
- **Stability**
- **Effectiveness**: the platform must be efficient.
- **Maintainability**: the application must be legible and understandable in order to maintain and update it quickly and easily by the developer.
- **Flexible**
- **Reusability**: the platform allows the addition and editing eventual functionalities optionally (implement widgets, change brand, template…).

2.4 Use case diagram

In the Unified Modelling Language (UML), a use case diagram can summarize the details of our system's users (also known as actors) and their interactions with the system. This diagram presents the main functionalities of this framework that we have to provide for the end user and the developer.

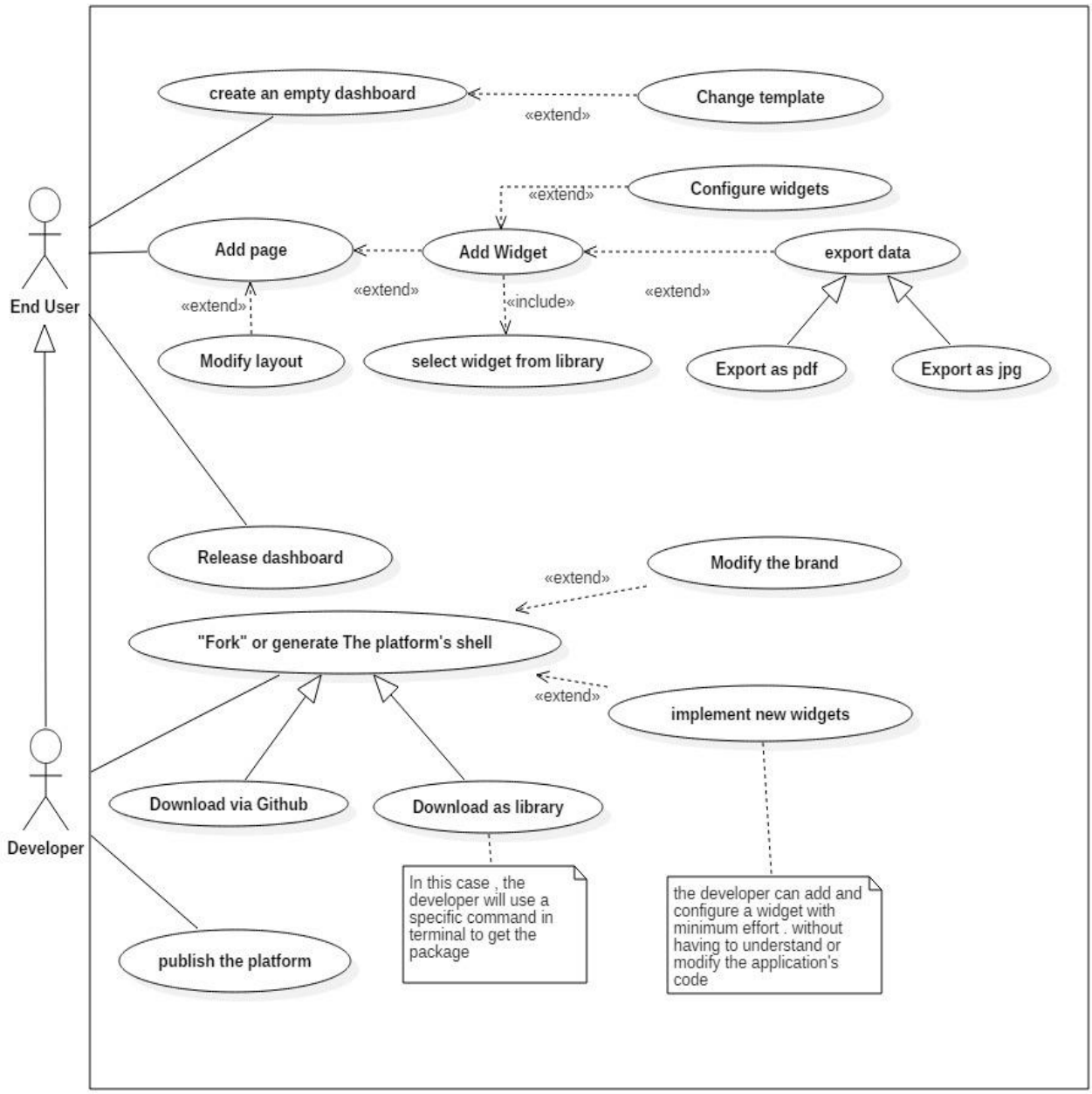

Figure 5 Use case diagram of the Longview platform

2.5 The model

This simple model shows the basic functionalities that the end user will find.

As the model shows, the end user can add more pages, these pages represent dashboard, in each dashboard he/she can add new widgets by clicking on" + " ,this button will shows all the types of widgets and all the charts in every type with description ,and after selecting widgets he/she just add them to the page.

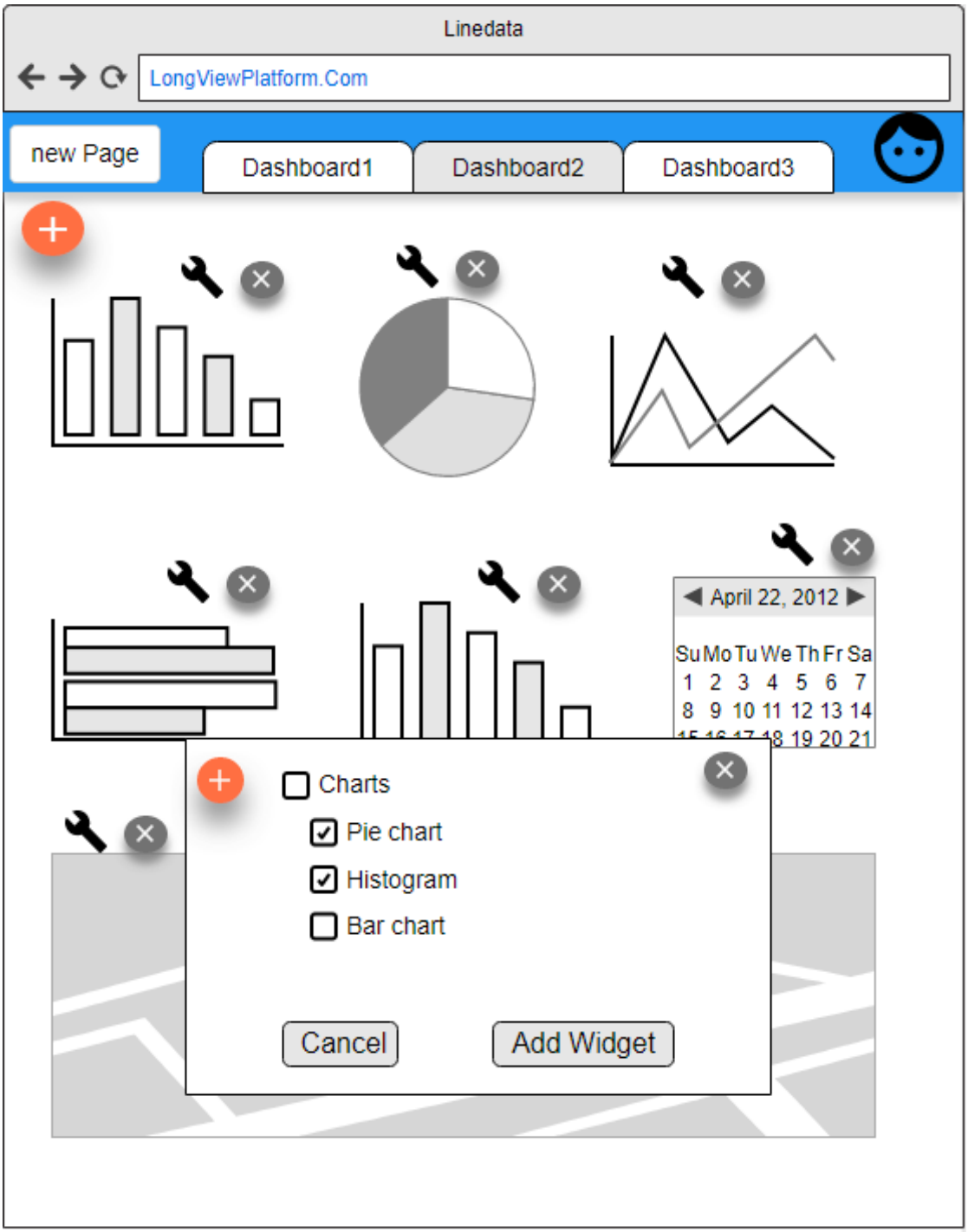

Figure 6 a preview of the Platform‘s graphic template

2.6 Product backlog

The product backlog is a list of all the desired features for the product. We present ours in this table. The priority levels used vary between 1 the least priority and 3 the highest priority.

Table 1 Product backlog

Id	User story	Priority
1	As a developer, i want to "fork" the shell code in order to use it.	3
2	As a developer, i want to create a new widget type.	3
3	As a developer, i want to publish my new created widget easily to the widget library.	3
4	As a developer, i want to be able to brand the dashboard so that it meets my client's needs	2
5	As an end user, i want to be able to create a new dashboard	3
6	As an end user, i want to customise the layout	3
7	As an end user, i want to discover existing widgets and use them in my dashboard pages, so that i get to visualize my data	3
8	As an end user, i want to restore my last modification in each widget every time I access the dashboard	2
9	As an end user, i want to customize my widgets, and organize their layout	2
10	As an end user, i want to drag and drop the widgets as I like in dynamic why inside the dashboard	3
11	As an end user, i want to visualise the data in real time.	3
12	As a developer, i want to modify and manage the source code in all platforms (Linux, Ubuntu, Windows…)	3
13	As an end user, i want to log in to the platform	3
14	As an end user, i want my data to load when I log in the platform	3
15	As an end user, i want to export the charts and the content of the widget as pdf, xml or jpg files.	2

2.7 Technological choices

Even though some choices were already made before i join the team, during my internship I was able to understand those choices. I also had the opportunity to make some choices later. In order to implement the desired global architecture the technological choices were not that obvious, mainly it's because we needed to explore many choices before making one, since we never worked with any of them before. Key factors we took in consideration when choosing were: performance and cost.

2.7.1 Angular 5.0

Angular is a platform that makes it easy to build applications with the web. Angular combines declarative templates, dependency injection, end to end tooling, and integrated best practices to solve development challenges. Angular empowers developers to build applications that live on the web, mobile, or the desktop.

Figure 7 Angular

The key features of Angular 5.0 are as follows:

- Simpler Progressive Web Applications
- Material Design
- Build optimizer
- Angular Universal API and DOM
- Improved Compiler and Typescript
- Number, date and currency pipes update

Simpler Progressive Web Applications

Progressive web applications are generating much hype these days. Understanding this trend, Angular Team has put emphasis to simplify the PWA making process. Not only that, with Angular 5.0 it is possible to get the features of native mobile applications with the mobile web apps such as push notifications and offline experience. This is made possible as Angular can create code and configuration with Angular-CLI on its own.

Material Design Components

Another major update in Angular 5.0 is that Material Design components are now made compatible with server-side rendering.

Built Optimizer

Angular 5.0 comes with build optimizer tool. It makes the application lighter and faster by removing unnecessary runtime code as well as unnecessary additional parts. Ultimately the size of the JavaScript decreases, and application becomes much faster.

The AOT compiler is much faster than it was in Angular 4 and it is also switched on by default. Basically, the AOT compiler converts Angular Typescript into efficient JavaScript code before the browser downloads it and runs it.

There is no doubt that Angular is the most popular JavaScript framework. Google is putting efforts to make it better and the progress can be observed by comparing the initial release of AngularJS to the latest version. Angular 5.0 is packed with some amazing features along with its core ones like dependency injections. All these updates make it faster, lighter and easy to use. Indeed, Angular is a super heroic framework that is beneficial to both developers and users.

Angular is the most preferred framework for creating interactive components of a website. It was designed as a full-featured JavaScript framework to enhance simplicity and efficiency. Developers find Angular very effective especially in creating dynamic, single page apps, and supporting MVC (Model View Controller) programming structure, so we will be using it for developing the front-end of the Longview platform.[7]

2.7.2 .Net Core

Figure 8 Dotnet Core

For the backend we will be using Dotnet Core.it is a general purpose development platform maintained by Microsoft and the .NET community on GitHub. It is cross-platform, supporting Windows, macOS and Linux, and can be used in device, cloud, and embedded/IoT scenarios.

The following characteristics best define .NET Core:

Flexible deployment: Can be included in your app or installed side-by-side user- or machine-wide.

Cross-platform: Runs on Windows, macOS and Linux; can be ported to other operating systems. The supported Operating Systems (OS), CPUs and application scenarios will grow over time, provided by Microsoft, other companies, and individuals.

Command-line tools: All product scenarios can be exercised at the command-line.

Compatible: .NET Core is compatible with .NET Framework, Xamarin and Mono, via the .NET Standard.

Open source: The .NET Core platform is open source, using MIT and Apache 2 licenses. Documentation is licensed under CC-BY. .NET Core is a .NET Foundation project.
Supported by Microsoft: .NET Core is supported by Microsoft, per .NET Core Support

2.7.3 Visual code

Figure 9 Visual Code

Visual Studio Code is a source code editor developed by Microsoft for Windows, Linux and macOS. It includes support for debugging, embedded Git control, syntax highlighting, intelligent code completion, snippets, and code refactoring. It is also customizable, so users can change the editor's theme, keyboard shortcuts, and preferences. It is free and open-source, although the official download is under a proprietary license.

Visual Studio Code is based on Electron, a framework which is used to deploy Node.js applications for the desktop running on the Blink layout engine. Although it uses the Electron framework, the software does not use Atom and instead employs the same editor component (codenamed "Monaco") used in Visual Studio Team Services (formerly called Visual Studio Online).[8]

2.7.4 Typescript

Figure 10 Typescript

TypeScript is a free and open-source programming language developed by Microsoft that aims to improve and secure the production of JavaScript code. It was created by Anders Hejlsberg in 2012 with the addition of new features to the JavaScript language. [9]

2.7.5 WebSocket

Figure 11 Websocket

WebSocket is a computer communications protocol, providing full-duplex communication channels over a single TCP connection. The WebSocket protocol enables interaction between a web client (such as a browser) and a web server with lower overheads, facilitating real-time data transfer from and to the server. [10]

2.7.6 GitHub

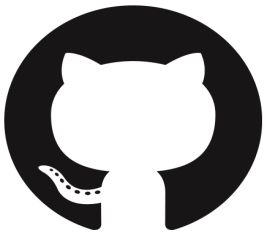

Figure 12 GitHub logo

GitHub (originally known as Logical Awesome LLC) is a web-based hosting service for version control using git. It is mostly used for computer code. It offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project. [11]

Conclusion

In this chapter we gave an overview of the technologies we will be using during the actual implementation, the Non-functional requirements, functional requirements, the backlog and a list of diagrams.

Chapter 3: Sprint1: The main missions for the frontend

Introduction

After a lot of hard work and a lot of side projects to learn the new technologies we mentioned in the last chapter, we used what we learned to develop the main functionalities. Our project was open for changes based on the difficulties we faced. We start preparing the main functionalities of our dashboard. So the three first months were divided between documentation and learning the technologies and developing the functionalities presented in this sprint. Since these technologies are new to us and it is very important to understand it and learn most of the techniques we will be using like ngrx/Store, effects, services in angular…

3.1 Sprint Backlog

Table 2: Sprint 1 backlog

Functionalities	Estimation
Add a page	3
Delete a page	1
Modify layout	1
Discover existing widgets	1
Add a widget to page dynamically.	4
Delete widget.	1
Configure widgets.	3
Save and restore the state of the pages and the widgets.	5
Export the widget as jpeg or pdf	1

This shows the list of all the functionalities assigned to this sprint and the duration that each will take.

3.2 Sprint description

In this part we are going to describe and get in the details of these functionalities:
In each dashboard, the user will create one or more pages, these pages represent a dashboard itself, for example a client wants different type of dashboards in the same web application like Strategic dashboard, Tactical dashboard, Operational dashboard…
When the user press on **'add page'**, a form with name of page, description and other inputs will appear. And a user can **delete a page** which means the data and the widgets in it will be lost too**.**
Modify layout: This application offers the user the ability to **modify the layout** of the pages any time. After adding at least one page, the user can add the widgets that he/she will select from a list that will appears after pressing the 'add widget' button, this list will allows the user to **discover the existing widgets** and by clicking on one of them, this widget will be **added dynamically to the page.**
All of the different type of widgets can be **deleted** and **configured** at any moment and this modification will be **saved** and the client will find his/her pages at the same state because each time he/she opens the dashboard, the **last state will be restored**.
The user can **export the data** that every widget represents and save it as jpeg or pdf file so that he/she can share it.

3.3 Analysis

3.3.1 Sequence diagram

UML sequence diagrams model the flow of logic within our system in a visual manner, enabling us both to document and validate our logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modelling, which focuses on identifying the behaviour within our system.
This diagram presents a simple workflow that the client can make with our platform when.

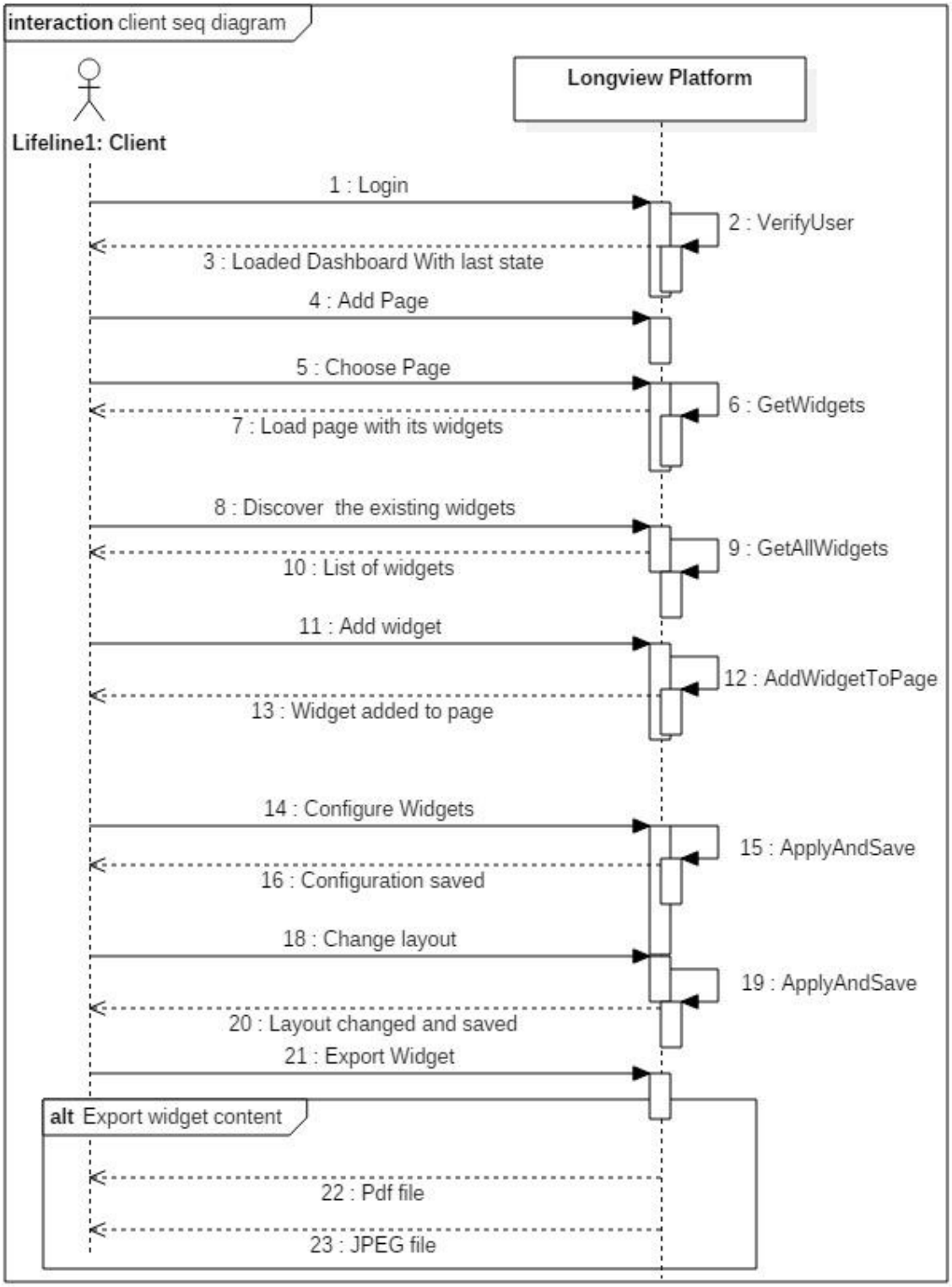

Figure 13 Client Sequence Diagram

3.3.2 Diagram of the participating classes

We present below the diagram of the participating classes to understand more the interaction of the user with the most important interfaces in our web application.

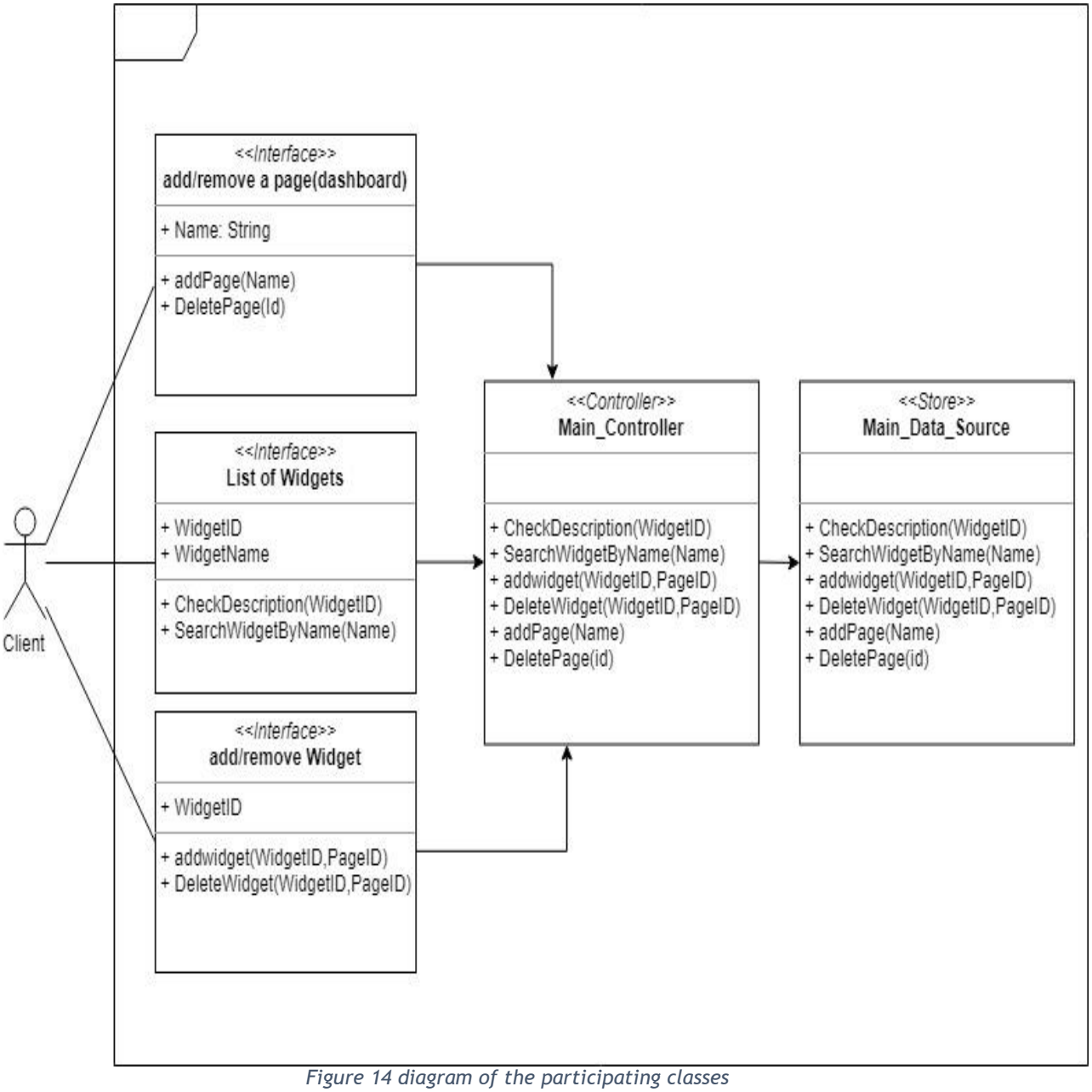

Figure 14 diagram of the participating classes

3.4 Design

After completing the analysis phase, we start the design phase. It translates into sequence diagram and the class diagram of sprint 1.

3.4.1 Detailed sequence diagram

Sequence diagrams, commonly used by developers, model the interactions between objects in a single use case. They illustrate how the different parts of a system interact with each other to carry out a function, and the order in which the interactions occur when a particular use case is executed.
In simpler words, a sequence diagram shows different parts of a system work in a 'sequence' to get something done.

Before we continue the design part, it is important to present the way our data will be managed.

@Ngrx/Store:

Like a traditional database represents the point of record for an application, the **store** can be thought of as a client side '*single source of truth*', or database. By adhering to the store contract when designing your application, a snapshot of store at any point will supply a complete representation of relevant application state. This becomes extremely powerful when it comes to reasoning about user interaction, debugging, and in the context of Angular, performance.

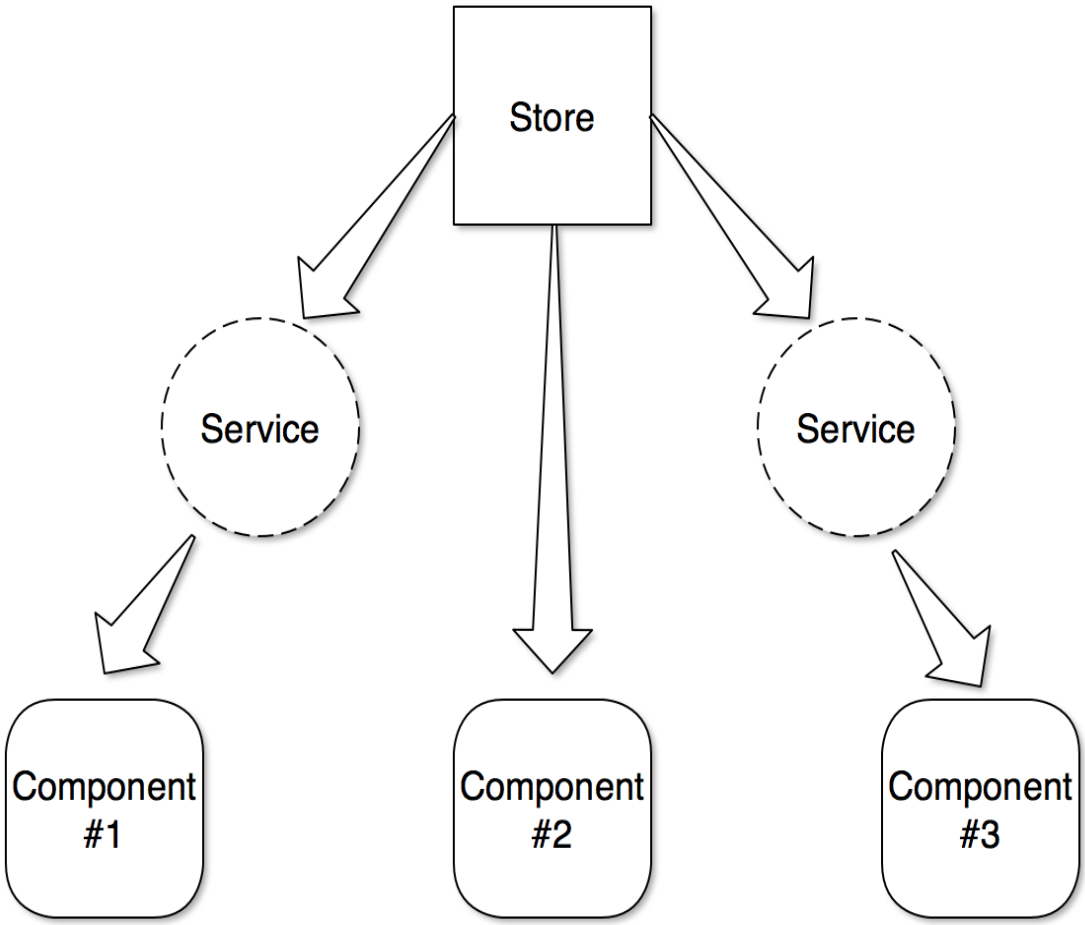

Figure 15 Store: Data management

a) Sign in and add a dashboard (page)

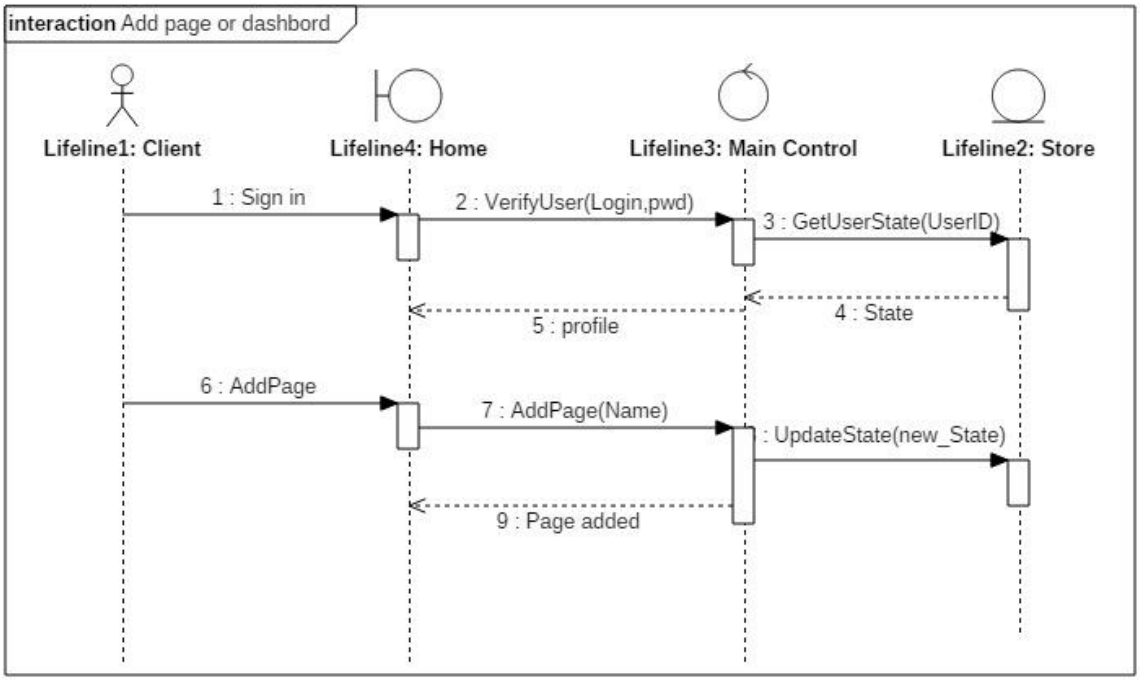

Figure 16 Detailed sequence diagram1

b) Add Widget to page

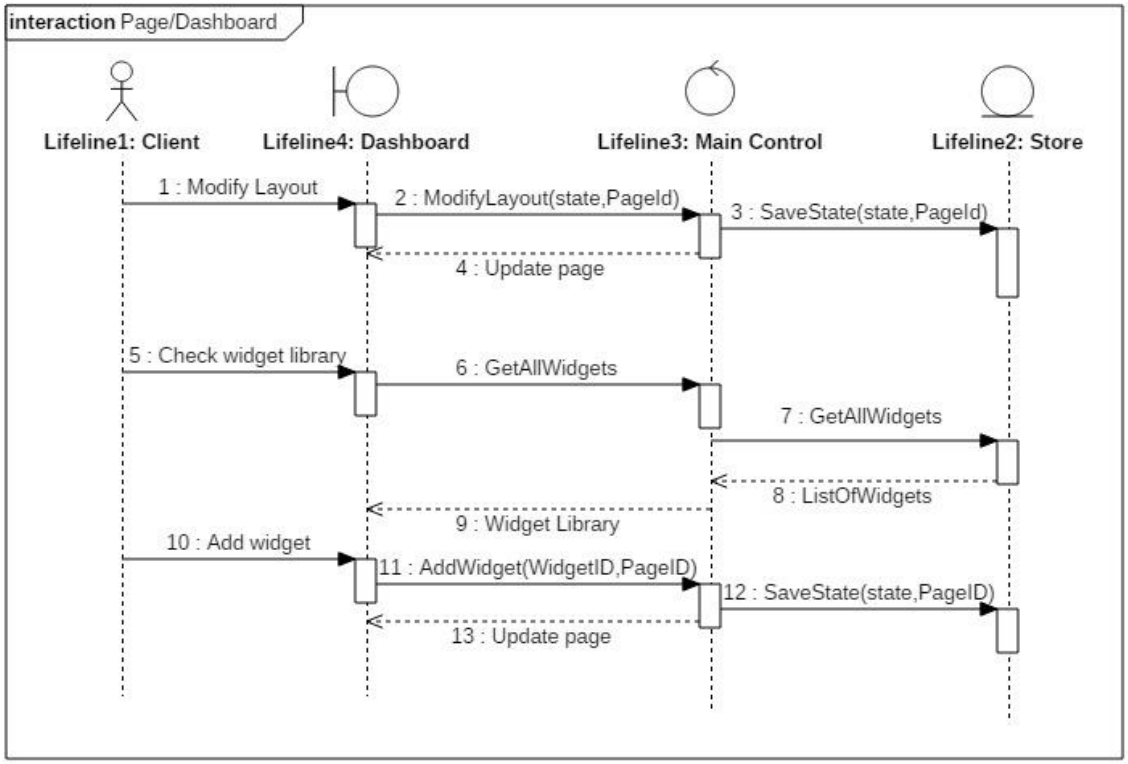

Figure 17 Detailed sequence diagram 2

3.4.2 Class diagram

Class diagrams are one of the most useful types of diagrams in UML as they clearly map out the structure of a particular system by modelling its classes, attributes, operations, and relationships between objects.

It is a static diagram. It represents the static view of an application.

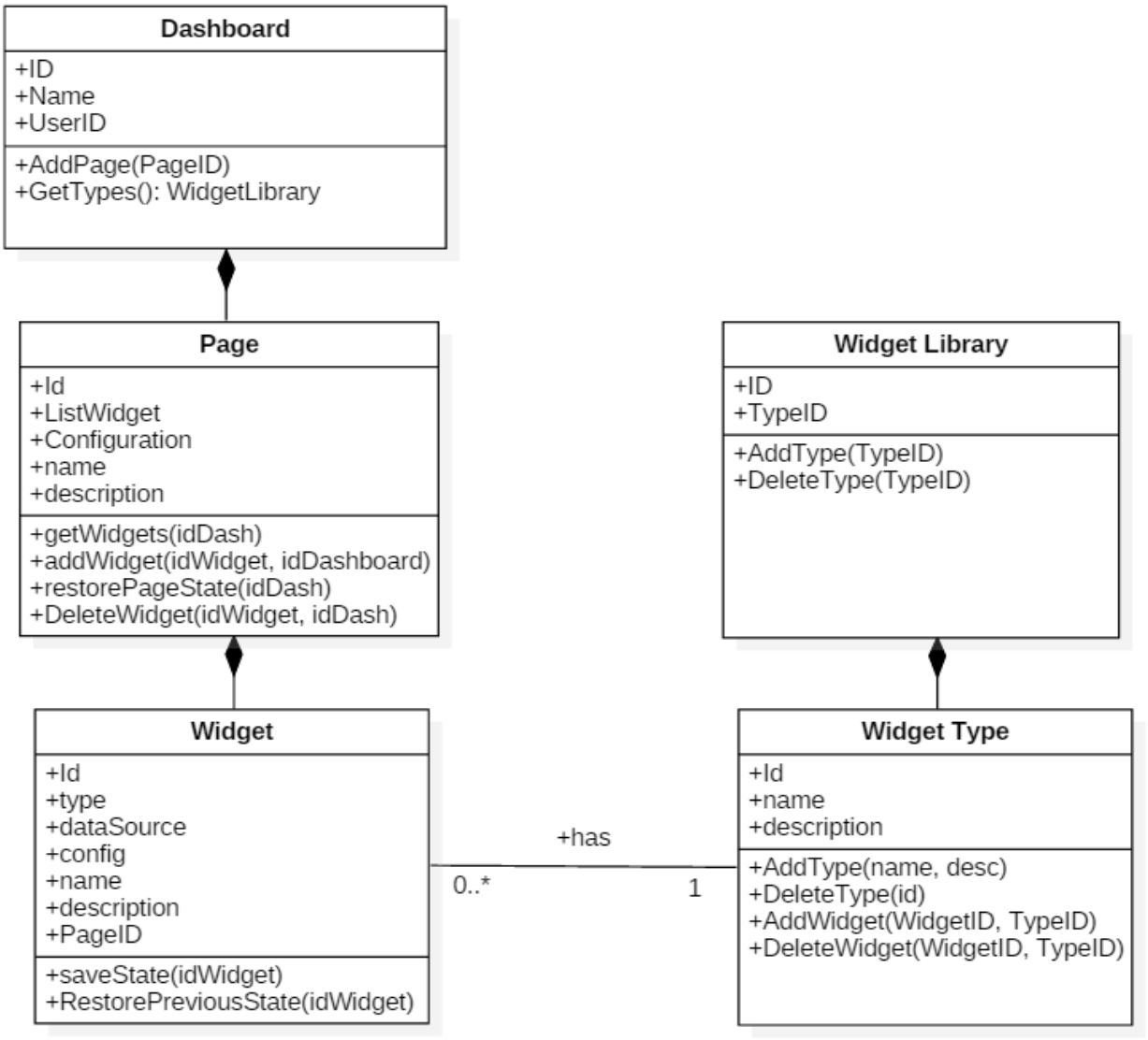

Figure 18 Class diagram

In this diagram we focused on the main objects of our framework.it presents the relations between the most important components.

3.4.3 Activity diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams are intended to model both computational and organizational processes.

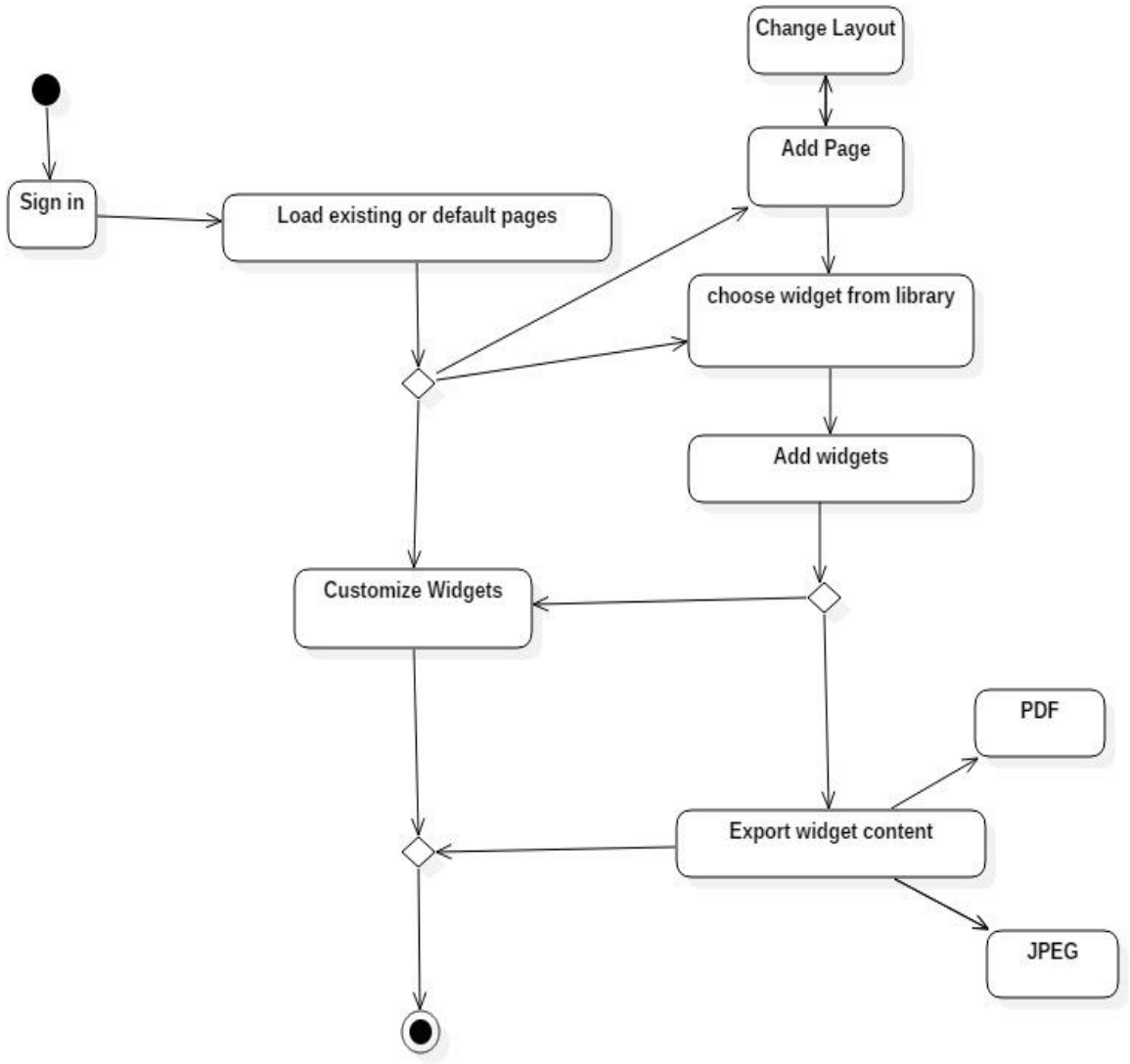

Figure 19 Activity diagram

This diagram presents the main workflow for the client. In this case the client will interact with the web application using the navigator to create dashboards, to visualise his/her or the company's data and then take the best decision.

3.5 Implementation

In this sprint we develop the main functionalities that the developer would like to find so he/she can satisfy their client's needs. So in this case the developer will find an easy way to implement his widgets based on what the end user wants and then publish the dashboard so that it will be ready use by the client.

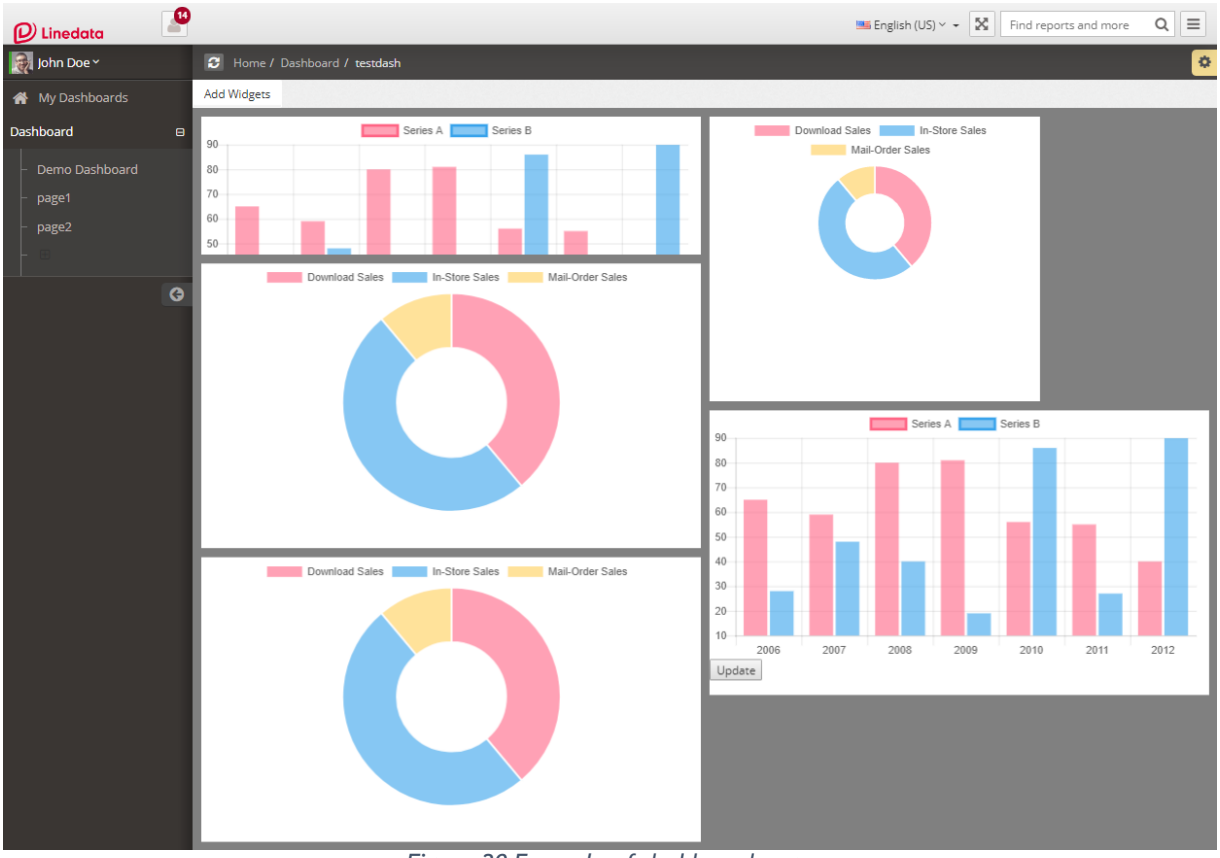

Figure 20 Example of dashboard page

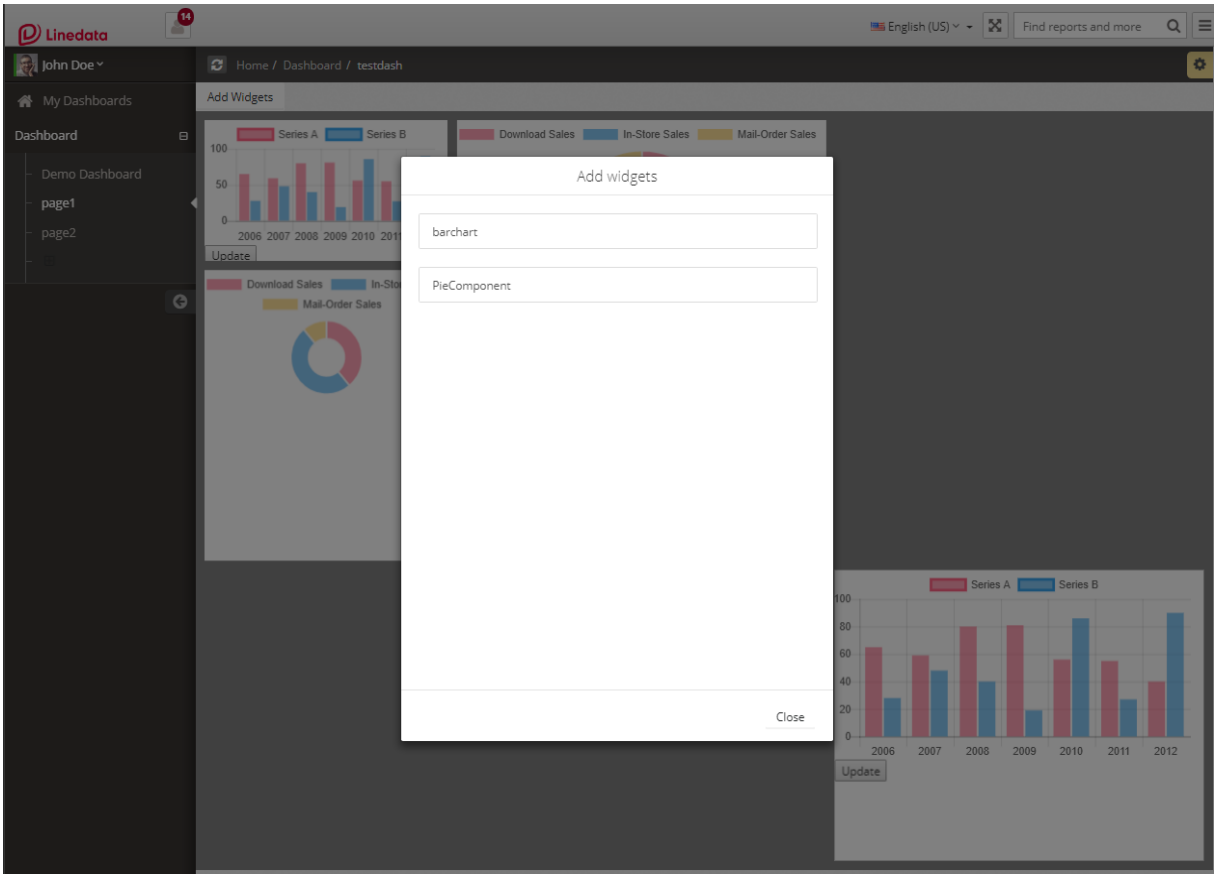

Figure 21 Add widget interface

The first figure shows tow type of widget, in this case we did the job of the developer and added tow type of widgets to the framework to show a demo of what the application will be when it is ready to use.
In the second figure, a list of the widgets added by the developer will be added automatically to the widget library so that the client navigate and choose any type of widget to be added.

3.6 Test

Web Testing in simple terms is checking our web application for potential bugs before it's made live or before code is moved into the production environment.
During this stage issues such as that of web application security, the functioning of the site, its access to handicapped as well as regular users and its ability to handle traffic is checked.
In this testing we are using **Jasmine**, it is a behaviour-driven development framework for testing JavaScript code. It does not depend on any other JavaScript frameworks. It does not require a DOM. And it has a clean, obvious syntax so that we can easily write tests.
The '*ng test*' command builds the app in watch mode, and launches the karma test runner.

Figure 22 the console output after testing the application

The last line of the log is the most important. It shows that Karma ran three tests that all passed.
A chrome browser also opens and displays the test output in the "Jasmine HTML Reporter" like this:

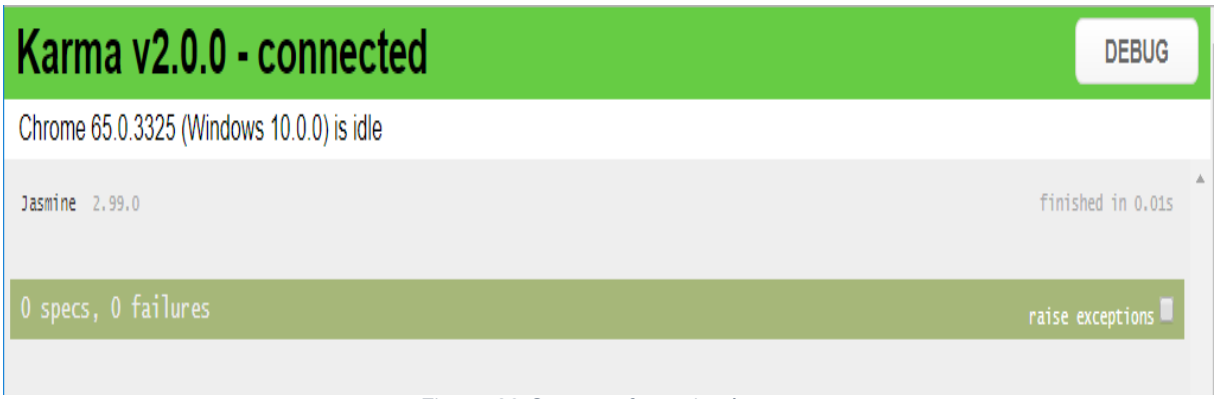

Figure 23 Output of test in chrome

Conclusion

In this chapter we describe the main functionalities that were done in this sprint, we started by the backlog of this sprint and then we present the four main steps in each sprint: analysis, design, implementation and testing.

General Conclusion

Working on this project was a little bit hard, not only the technical perspective, but also features were in production from day zero, so a good care about the quality of deliverables was not a choice but an obligation. But this graduation project was a rich opportunity to cover diverse engineering aspects and it was a wonderful experience and an interesting one with all different type of people and the different technologies that I have used.

Bibliography

- [1] : <http://www.linedata.com/company/about-us>
- [2]: < http://www.dashboardinsight.com/articles/digital-dashboards/fundamentals/what-is-a-dashboard.aspx >
- [3]:< https://www.klipfolio.com/resources/articles/what-is-a-key-performance-indicator >
- [4]: < https://www.envisionit.com/about-us/our-process >
- [5]: < https://en.wikipedia.org/wiki/Klipfolio_dashboard >
- [6]: < https://www.dundas.com/dundas-bi>
- [7]: < https://angular.io/>
- [8]: < https://fr.wikipedia.org/wiki/Visual_Studio_Code>
- [9]: < https://fr.wikipedia.org/wiki/TypeScript>
- [10]: < https://en.wikipedia.org/wiki/WebSocket >
- [11]: < https://en.wikipedia.org/wiki/GitHub >