LAMK Lahden ammattikorkeakoulu
Lahti University of Applied Sciences

iFlow

Automated production flow tracking

LAHTI UNIVERSITY OF APPLIED SCIENCES
Master's Degree Programme in Information and Communications Technology
Master´s Thesis
Spring 2018
Ville Kauppinen

Lahti University of Applied Sciences
Master's Degree Programme in Information and Communications Technology

KAUPPINEN, VILLE: iFlow

Automated production flow tracking

75 pages

Spring 2018

ABSTRACT

The aim of the thesis was to plan and create an application for tracking the gravure printing process. In its current state, the process is divided between various subcontractors and viewing the overall status of the process requires a lot of manual interaction. The goal was to gather process information from various data sources and combine it in one dashboard view presenting the current status of production. The thesis presents a working application model for combining vendor and in-house data sources.

The thesis includes evaluating different open-source platforms and frameworks and using them to develop a modern, web-based production tracking dashboard. The goal was to create an application capable of handling various 3rd party data connections and providing a single overview of different parts of the printing process.

The successful development of the application proved that open-source platforms and frameworks used with Agile development methologies are applicable in a commercial environment.

Key words: agile, software development, integration, gravure printing, printing process

Lahti University of Applied Sciences
Master's Degree Programme in Information and Communications Technology

KAUPPINEN, VILLE:

iFlow

Automaattinen tuotantoprosessin seuranta

75 sivua

Kevät 2018

TIIVISTELMÄ

Opinnäytetyön tavoite oli suunnitella ja toteuttaa sovellus syväpainoprosessin kokonaisvaltaiseen seuraamiseen. Nykytilassaan tuotantoprosessin seuranta vaatii paljon manuaalista työtä. Tavoite on yhdistää eri tietolähteiden tarjoamat tiedot yhteen kojelauta -malliseen näkymään. Opinnäytetyön esittää toimivan applikaatiomallin alihankkijoiden järjestelmien yhdistämiseksi tuotantolaitoksen datalähteisiin.

Opinnäytetyö sisältää erilaisten open source -alustojen ja ohjelmistokehysten analysoinnin sekä niiden yhdistämisen nettipohjaiseen tuotannonseurantanäkymään. Opinnäytetyön tavoite oli luoda sovellus, joka pysty yhdistämään kolmansien osapuolien tietolähteet ja tuottamaan yhden yleiskuvan eri painoprosessien tilasta.

Sovelluksen onnistunut kehitys todistaa, että open source -alustat ja ohjelmistokehykset ketterän kehityksen kanssa ovat varteenotettava vaihtoehto kaupallisessa ympäristössä.

Asiasanat: agile, ohjelmistokehitys, integraatiot, syväpaino, painoprosessi

CONTENTS

1 INTRODUCTION

Digitalization, or process automation, is a strongly growing trend in all over the world. It can be defined simply as automating a manual process with technological solutions. This thesis will be done on behalf of Eracon Oy for Helio Charleroi. Helio Charleroi is one of the most modern printing plants in Europe.

Helio Charleroi has a 30-year history and is located in Charleroi, near Brussels. The plant has a good location compared to large market areas such as Brussels, Paris, Amsterdam, London, and Frankfurt. With two extra large presses, Helio Charleroi is an efficient and swift producer in medium and large volume Rotogravure printing.

1.1 Aim of the study

> *"Digitalization is the use of digital technologies to change a business model and provide new revenue and value-producing opportunities; it is the process of moving to a digital business." (Gartner 2017a).*

The thesis is based on a development project where the main goal is to generate a working, digitalized process by using modern software framework and open-source software solutions. The main goal is to develop and test a process model and evaluate different software development processes around frameworks.

In modern printing world, the process is divided between several actors and managing a print project is a time-consuming challenge utilizing traditional methods. A weekly magazine requires several contributors before it is ready for delivery and distribution. Usually, these contributors are a cover, miscellaneous inserts, and the pages manufacturer and all three operate in their own closed processes and environments. Keeping track of the phase of each contributor requires lots of emails and phone

calls. This consumes time and even in best case will not have a real-time view on the production overall status.

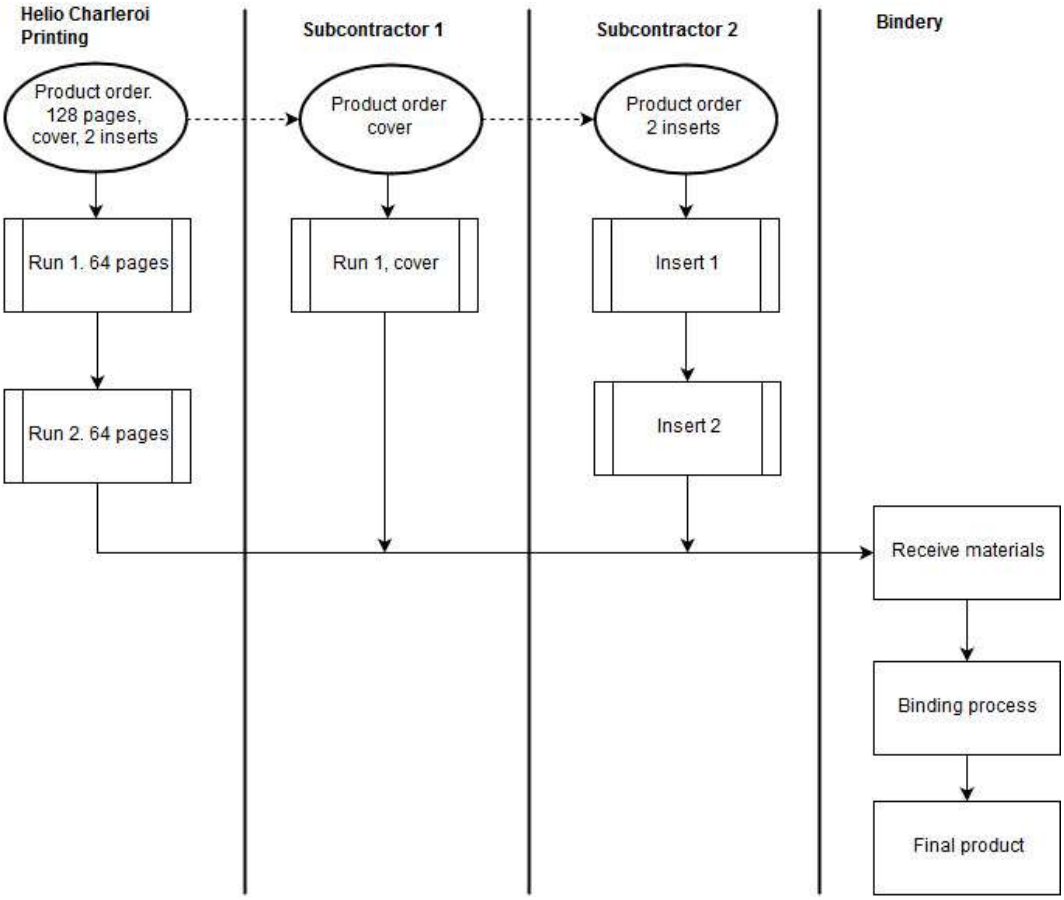

FIGURE 1. Basic printing process

Figure 1 describes a basic printing process for 128 pages wide title with cover and two inserts. Helio Charleroi (HCH) receives an order and separates it into different production units. HCH does the printing for the main title, 2x 64-page runs and subcontractors handle cover, inserts and the final binding.

1.2 Research goals and questions

The goal of the thesis is to build a collaborative tracking site for printing industry tying customer, manufacturer, and subcontractor together using open source tools.

This thesis aims to find out answers to the following questions:

1. How can open-source systems provide a competitive platform for digitalization projects?
2. What are software frameworks, and how they improve development efficiency?
3. Are agile methods effective in developing software for traditional printing industry ?

In addition to answering these questions, the goal is to build a working system with predefined parameters described in the later chapter.

1.3 Working methodology

The overall process map for building application is shown in Figure 2. In this model, the work is divided into five different phases. In the first phase, the development process is defined and technological questions answered. In the second phase, decisions are made based on research done in the first phase. The third phase concentrates on building suitable environment for the application. The fourth phase concentrates on building the application. After these four phases, the application should be up and running, and documentation can be created.

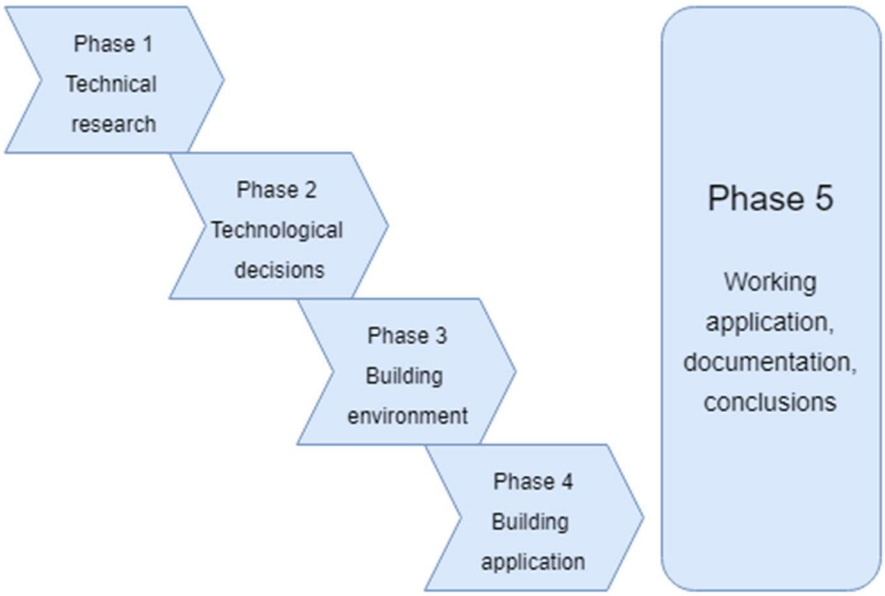

FIGURE 2. Phase map of the building process

1.4 iFlow

iFlow is the working title of the application. It describes automatical visualization of the printing process flow. As some parts of the printing process are usually handled by various subcontractors, it is essential that the main contractor is always on track what is going on with the project.

The aim of the application is to ease communication with various actors and provide a transparent view to the printing process. The application will be the first step in integrating various 3rd party production tracking systems to one single point.

2 TECHNICAL OVERVIEW AND DEVELOPMENT TOOLS

Application architecture has been pre-selected as a www-service. This narrows the possible environments down to Windows Server and number of Linux-distributions. Due to the overall economic situation in the printing industry, one of the primary goals is to use as many open source solutions as possible leaving the Linux-based system as a primary candidate.

The application will be deployed in a live production environment so the distribution should be an Enterprise version with long-term support. CentOs Project is an open source recompile of RedHat open source. CentOS is very close to RedHat or Amazon Linux which is used on Amazon EC virtual servers.

The environment configuration will be done as a part of this thesis and application deployment.

2.1 Zend Studio

Zend Studio is the integrated development environment for PHP developers. It incorporates a lot of useful features and tools such as code editor with code assist, code completion, automatical refactoring, error analysis and code validation.

iFlow will be built using GIT -version control system (VCS) and Zend Studio integrates seamlessly with it. Framework support is also included.

Figure 3 shows basic user interface of Zend Studio.

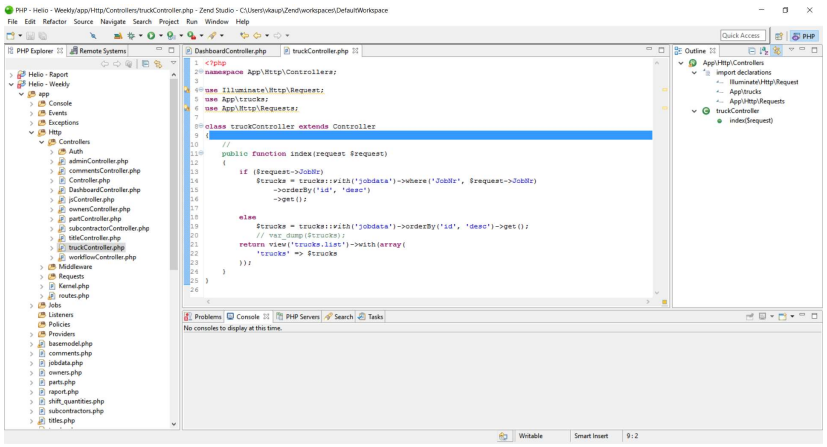

FIGURE 3. Zend Studio user interface

2.2 GIT

Git is a version control system (VCS). Version control systems software tools which help development team to manage source code and its changes over the development process. VCS keeps track of every modification committed to the code in software's internal database. The developers can choose to go back in version history to any version,

download it and fix possible bugs. The version history is stored as increments between commits (See Figure 4).

iFlow uses GitHub version control system as a storage for source code. Github can be accessed over the internet, so it is an excellent tool transferring source code between Finland and Belgium.

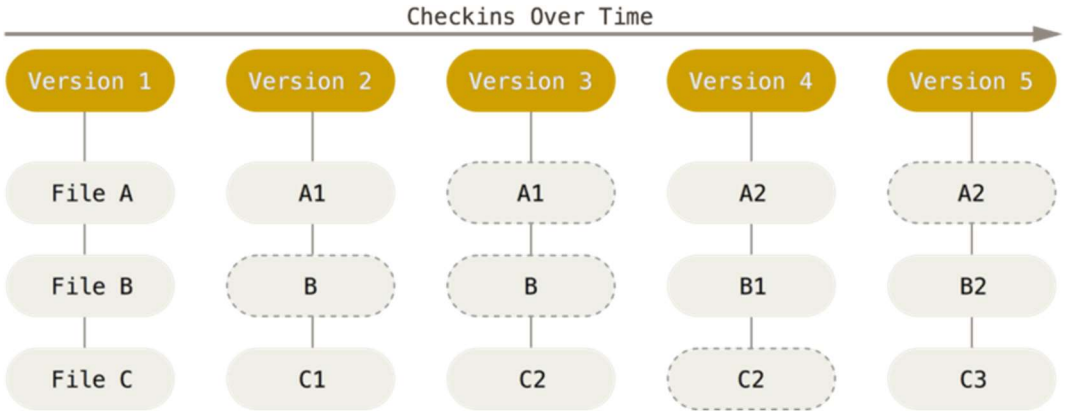

FIGURE 4. GIT version control explained

2.3 REST

> *"REST (Representational State Transfer) was introduced and defined in 2000 by Roy Fielding in his doctoral dissertation. REST is an architectural style for designing distributed systems. It is not a standard but a set of constraints, such as being stateless, having a client/server relationship, and a uniform interface. REST is not strictly related to HTTP, but it is most commonly associated with it." (Spring.io).*

REST is a software architectural structure instead of protocol. It defines means to exchange information between different parties rather than the information structure itself. The most common use case is a Web application where user or client is retrieving and updating information on a server via different web forms and views. It can also be used for

communication between different systems or software components in an intra- or internet environment.

IFlow utilizes REST -techniques for data transfer between all connected systems.

> *“Principles of REST*
>
> *Resources expose easily understood directory structure URIs.*
>
> *Representations transfer JSON or XML to represent data objects and attributes.*
>
> *Messages use HTTP methods explicitly (for example, GET, POST, PUT, and DELETE).*
>
> *Stateless interactions store no client context on the server between requests. State dependencies limit and restrict scalability. The client holds session state.” (Spring.io)*

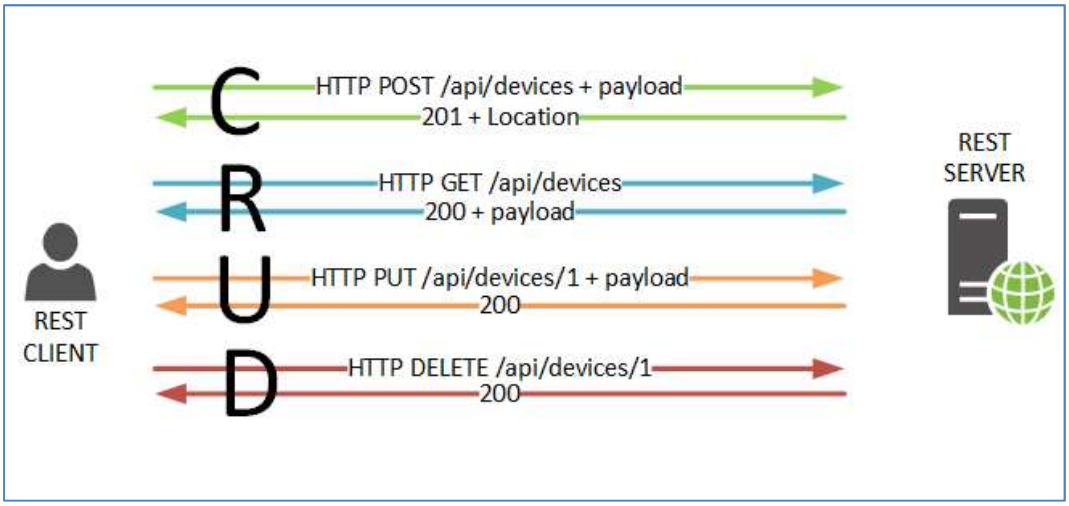

FIGURE 5. REST API

Operations performed via API are often referred as CRUD. This is an acronym for **c**reate, **r**ead, **u**pdate and **d**elete which are basic database or data handling functions (See Figure 5).

For example, if we want to **create** a new object, we need to construct a payload with object attributes and send it attached to an HTTP POST request.

To **read** an object or attributes from a data source, we can send an HTTP GET -request to the server. The server responds with a list of known results or attributes.

To **update** we send the object or attributes back to the server with HTTP PUT -method. The server evaluates the received data and updates the information if data is valid and changes are detected.

To **delete** an object or data, the method is HTTP DELETE.

There different methods separate the action and API can interpret the desired action correctly. For example, sending a delete request via GET would not work as delete is defined as DELETE -request.

2.4 Open Source Operating Systems

Open source software holds many advantages for businesses, some being more valuable than the non-existent price of the software. Most open source operating systems are considered to be highly efficient and stable compared to commercial counterparts.

Linux systems are known for their stability, and they can run years without a failure or need to reboot. System updates can be run directly on the server while it is running where Windows usually requires a reboot, especially after major updates.

Openness also creates security. As most, or all, of the source code, is visible to legions of developers vulnerabilities are found and fixed faster

than in closed, tightly controlled development process. Internally, Linux was designed from the start to be a multi-user environment creating more layers in user management and kernel access.

Efficiency comes from requiring less hardware to run. Most Linux servers are happy with a few-year old hardware and OS can accommodate itself to fewer resources.

2.5 CentOS

> *"The CentOS Project is a community-driven free software effort focused on delivering a robust open source ecosystem. For users, we offer a consistent manageable platform that suits a wide variety of deployments. For open source communities, we offer a solid, predictable base to build upon, along with extensive resources to build, test, release, and maintain their code."*
> *(https://www.centos.org/)*

The latest major version of CentOs is 7 and is selected as the operating system (OS) for the virtual machine (VM) providing iFlow the environment it needs to run properly.

If CentOS 7 is properly configured, it is a very stable operating system. The software packages included in the OS package manager are only stable versions and most common open source www-server packages such as Apache Web Server and MySQL are available via software package service. The downside is that newest development versions usually arrive a bit late since they need to test and verified as stable before adding to the package manager.

2.6 Virtualization

Virtualization is referred as means to create a virtual representation of a device or resource. These resources can be such as server, storage device, network or even a complete operating system. As a practical example partitioning a hard drive can be considered as virtualization because you take one drive and divide it to create two completely

separate hard drives. Devices and applications can to interact with the virtual resource as if it were a real resource. (Webopedia 2017a).

The concept of server virtualization is similar to hard drive partitioning. You take a part of larger resource pool and create a completely virtual environment for the required server to run in. In server virtualization, the physical hardware resources of the server itself are hidden from users and virtualization software is used to divide hardware server into multiple smaller virtual environments which share the original hardware resources. This is the complete opposite for dedicating a server to a single task or application (See Figure 6).

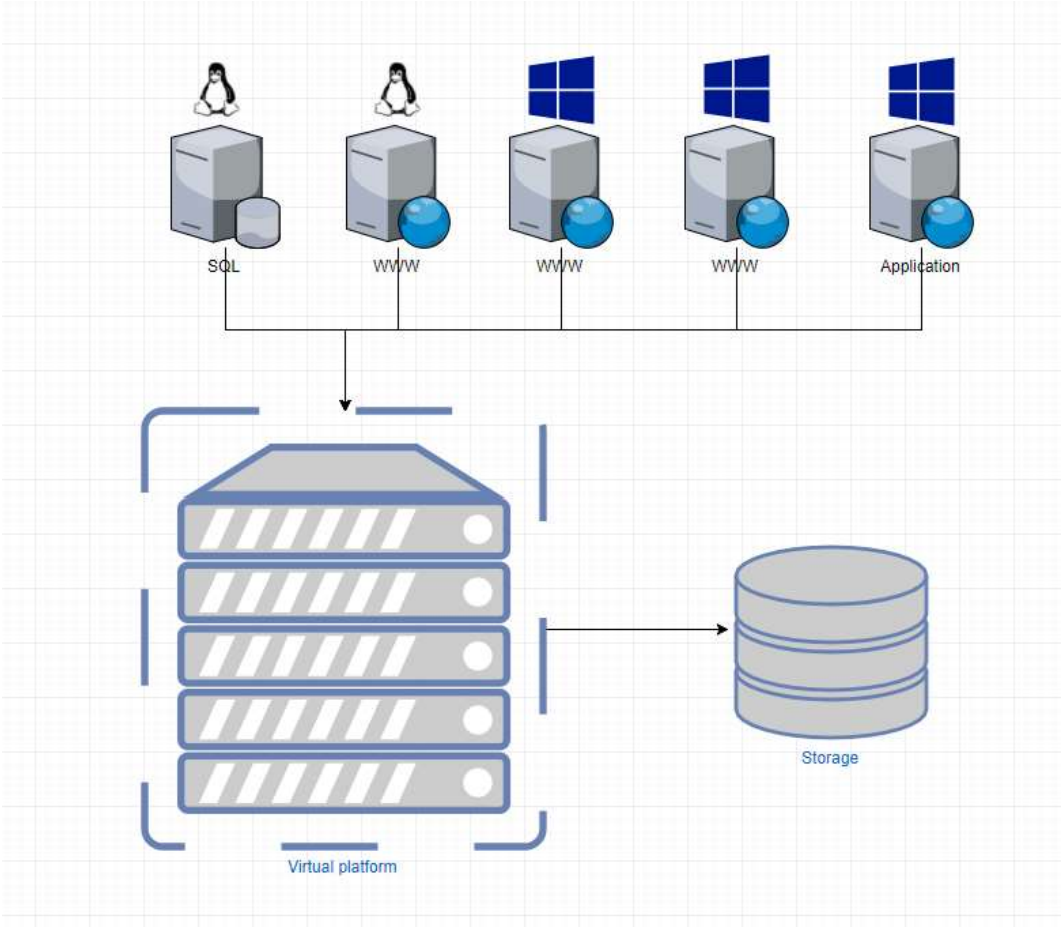

FIGURE 6. Virtualization

Most common usage of this technology is in Web servers, as seen in Figure 7.

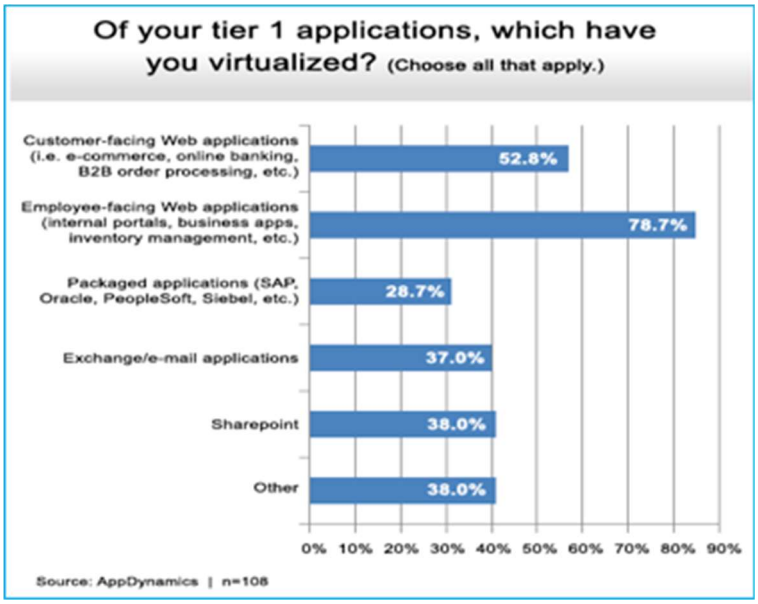

FIGURE 7. Virtualization targets (Virtual Realities 2018a).

Using virtual Web servers is a popular way to provide cheap Web hosting services. Instead of requiring a physical server for each Web server, multiple virtual servers can co-reside on the same computer. On top of that, one virtual server including Apache Web server can serve many virtual hosts or domains.

Virtualizing servers has many benefits. For example, each virtual server can run its own operating system and can be rebooted independently of one another. Managing one large hardware resource is easier and requires fewer hours than managing multiple separate hardware servers.

In addition to being cost effective, virtualization makes hardware fault tolerant. In case of hardware failure latest snapshot can be put online in a matter of minutes versus installing a completely new physical machine.

Moving a server to a new service provider is fast and easy. A snapshot of the virtual machine can be exported and loaded into a new platform

without any real hardware installations or limitations. This has made virtual server most popular platform choice (See Figure 8).

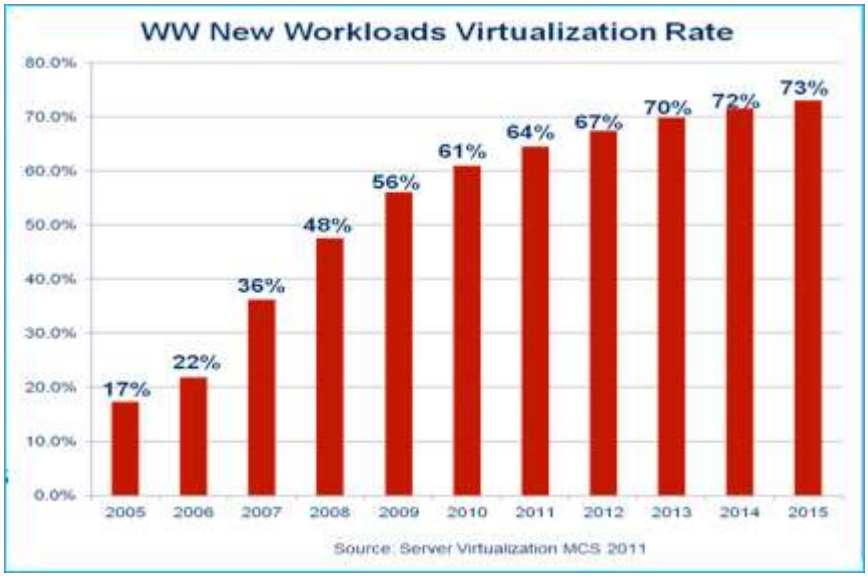

FIGURE 8. Virtualization rate (Virtual Realities 2018a).

3 PLATFORM AND APPLICATION ARCHITECTURE OVERVIEW

The main application, or web front, is deployed on the iFlow -server. This is located on DMZ-zone of Helio Charleroi's network. DMZ is an acronym of the demilitarized zone which is used to refer an area of the network outside Internet providers direct firewall. A server located in DMZ is neither as secure as the internal network nor as insecure as the public internet. (See Figure 9).

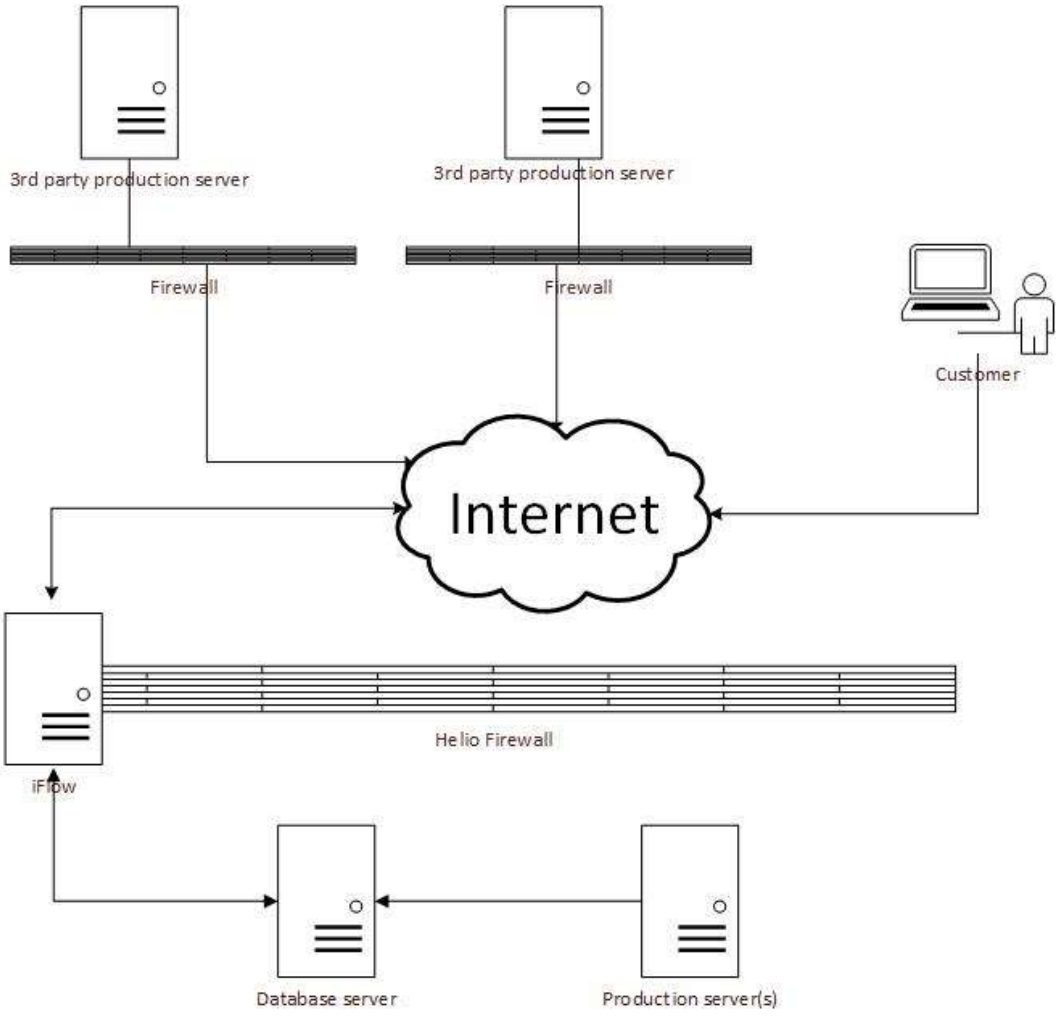

FIGURE 9. Production environment

Customers and external users can access iFlow directly via the Internet. 3rd party production data is transferred to the database server via REST API. Similar API can be found on the database server so internal production servers can update their data via more secure solution.

Depending on the security requirements 3rd party can be connected via iFlow API or database server API. The difference is that iFlow can be accessed directly through internet with security rules where DB server requires a dedicated VPN Lan to Lan -tunnel for increased security.

Customers or product owners can access the iFlow through web front with their own web browsers. In-built security is a username/password -combination defined in framework's authentication -functionality

3.1 Database

A database is a collection of information that is organized so that it can easily be accessed, managed, and updated. For this application, a relational database server was selected, as storage engine MySQL. Relational databases typically use id-fields to connect data between various tables. (See Figure 10). This establishes a well-defined relationship between database tables and data itself.

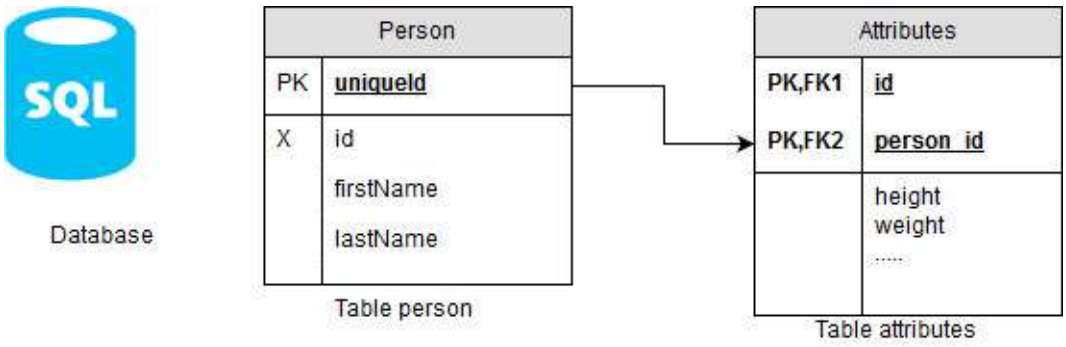

FIGURE 10. Relational database example

Relational databases organize data in relationships between tables and columns. Each table is known as a relation from database engines point of view. Tables contain one or more data columns for which to store data. Each record or row contains an id-column which is a unique identifier defined for a row. Accessing data row usually happens through that identifying column and relationships can cross between tables. One or more data or records relate to one or many records to form functional dependencies. (What Is 2017a).

Relational databases offer easy extendability, also known as scalability, as new data may be added without modifying existing records.

3.1.1 MySQL

> *"MySQL is an open source relational database management system (RDBMS) based on Structured Query Language, SQL. MySQL runs on virtually all platforms, including Linux, UNIX, and Windows. Although it can be used in a wide range of applications, MySQL is most often associated with web-based applications and online publishing and is an important component of an open source enterprise stack called LAMP. LAMP is a Web development platform that uses Linux as the operating system, Apache as the Web server, MySQL as the relational database management system and PHP as the object-oriented scripting language." (What is 2017c).*

In this thesis, MySQL will act as the database engine in iFlow -application. MySQL databases are also MariaDB or Amazon Aurora compatible which makes possible transition to new platform or engine easy. One identified post-development task is virtualization of the entire system into Amazon EC2 -cloud and connecting this environment to Helio and 3rd party contributors via Amazon VPN solution.

3.1.2 Amazon Aurora

> *Amazon Aurora (Aurora) is a fully managed, MySQL- and PostgreSQL-compatible, relational database engine. It combines the speed and reliability of high-end commercial databases with the simplicity and cost-effectiveness of open-source databases. It delivers up to five times the throughput of MySQL and up to three times the throughput of PostgreSQL without requiring changes to most of your existing applications." (Amazon 2018a).*

Amazon offers their own MySQL -compatible database engine in provisioned Relational Database System (RDS). It is clustered to different availability zones which mean the database is located in separate areas to avoid downtime. If one instance stops responding, another takes place immediately (See Figure 11).

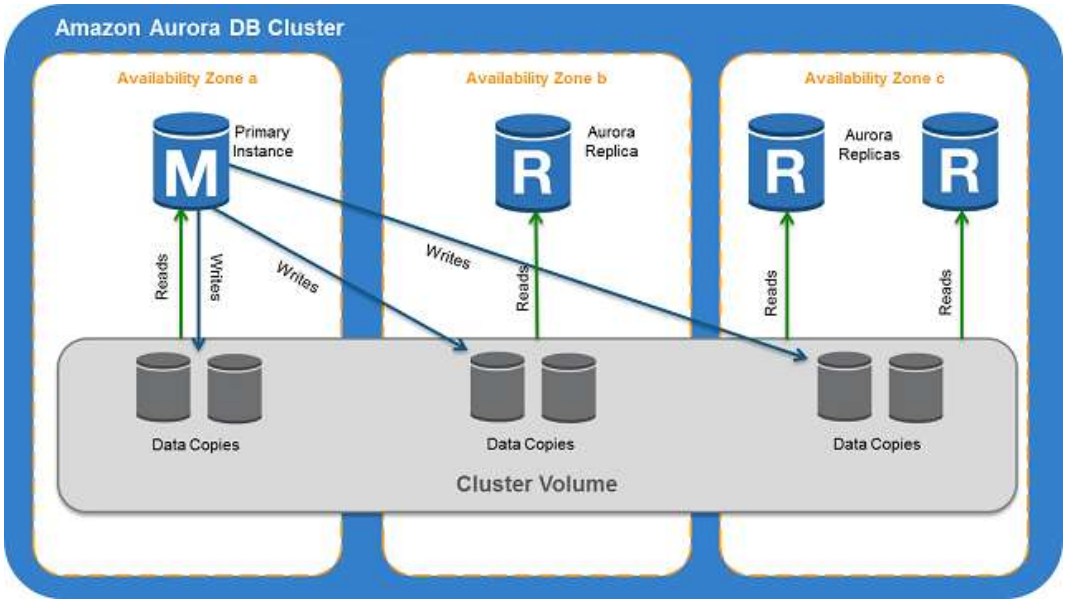

FIGURE 11. Amazon Aurora Architecture (Amazon 2018a).

3.2 Web and application server

For iFlow, the business logic and presentation layer will be set on the same server. These are called application server and web server also referred as web front. Application logic and presentation is handled with a Linux-based web server, Apache, and PHP.

All data is accessed, modified and stored via this server. In more complex cases it is sometimes practical to further separate application and web servers.

For application to work the Web server takes client requests. It passes those to a server-side program able to handle the request. The server-side program looks up the information from a database. Once retrieved, the program on the server uses the information to create an HTML response and then the Web server sends it back to the Web browser. On a larger scope with hundreds or thousands of client applications usually need multiple dedicated web servers and separate application servers to be able to process all requests without delay.

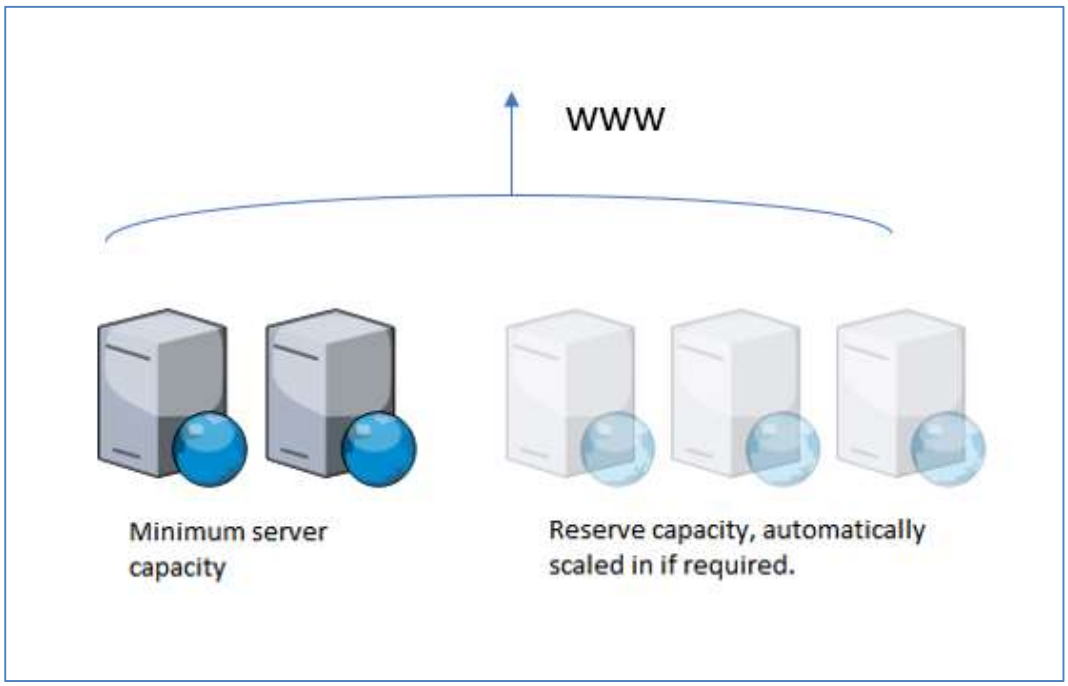

FIGURE 12. Auto Scaling Group

Using cloud-based solutions these servers can be provisioned with auto-scaling groups. This model automatically increases or decreases number of servers handling web requests based on pre-defined parameters such as CPU usage or page load times (See Figure 12).

3.2.1 Apache Web Server

A web server is a software that listens on specified TCP ports and receives clients request to read and access data. It runs a few security checks on an HTTP request and takes the client to the web page. Depending on the page client has requested, the page may ask the server

to run a few extra modules or even complete page requests while generating the document or application to serve (See Figure 13).

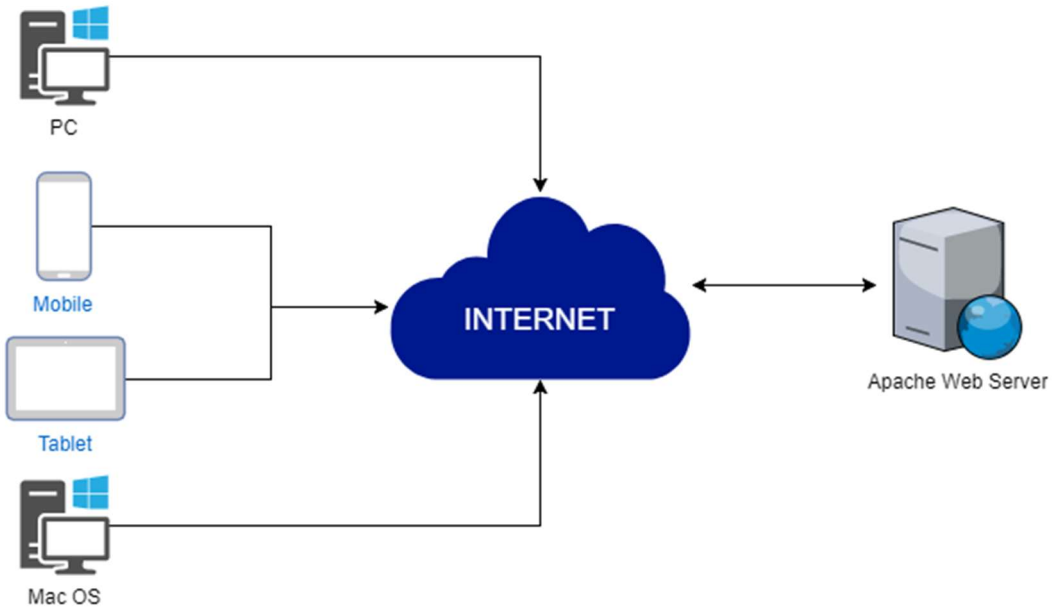

FIGURE 13. Apache Web Server

Apache is one of the most popular web server software as seen in Figure 14. It is maintained and developed by an independent Apache Software Foundation. It is an open source web server available for free and is considered to be fast, reliable, and secure. There are several customizations available to meet the needs of many environments by

using virtual hosts, different extensions, and number of modules. (Apache Project).

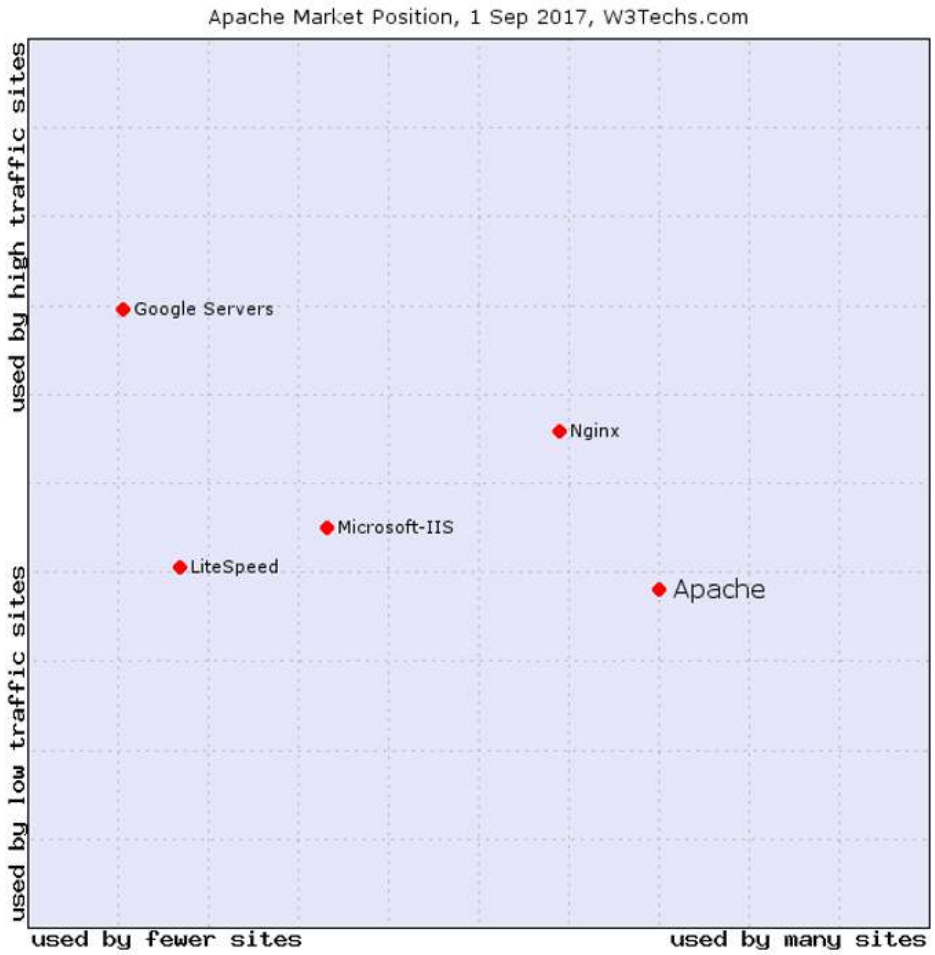

FIGURE 14. Apache Market Position (W3techs 2017a).

3.2.2 PHP

> *"PHP (recursive acronym for PHP: Hypertext Preprocessor) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. Much of its syntax is borrowed from C, Java and Perl with a couple of unique PHP-specific features thrown in. The*

goal of the language is to allow web developers to write dynamically generated pages quickly and effortlessly." (PHP 2017a).

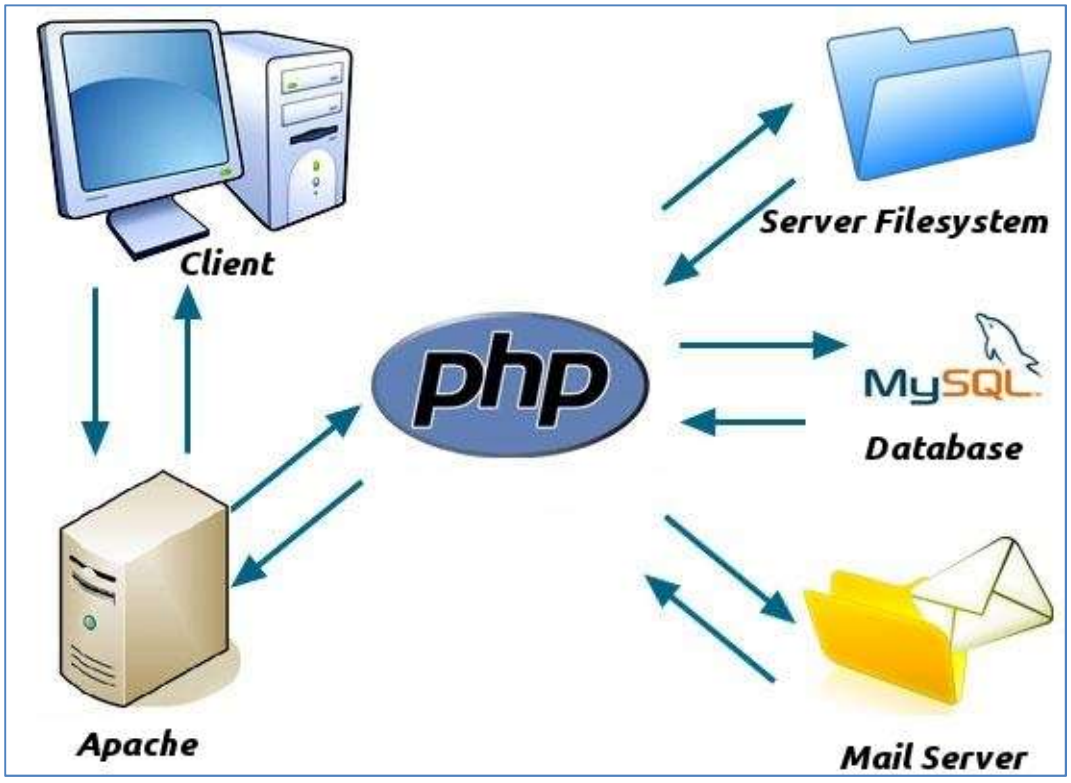

FIGURE 15. PHP handes the application logic (PHP 2017b).

PHP is a server-side language which means the code is executed on web server instead of client (See Figure 15). Most common example is that client accesses a web page. Apache web server receives the request, processes the program code, handles possible database queries and executions and generates a web page as a response back to the client. Famous example of a PHP application is Facebook, located in http://www.facebook.com

PHP request handling process:

1. The client sends an HTTP/HTTPS request to Apache Server through ports 80 or 443, depending on the protocol.

2. Web server receives the request and PHP engine executes possible scripts located in request path,
3. After engine completes its task it parses out HTML and server will return HTML output to Client.
4. HTML will be executed by the client browser and displays output as a web page.

3.3 PHP framework selection

There is a wide range of different PHP frameworks for developers to utilize. To ensure rapid development, a suitable framework needed to be selected. Basic requirements were listed as follows:

- built-in user authentication
- database connection
- MVC (explained in chapter 4.2)

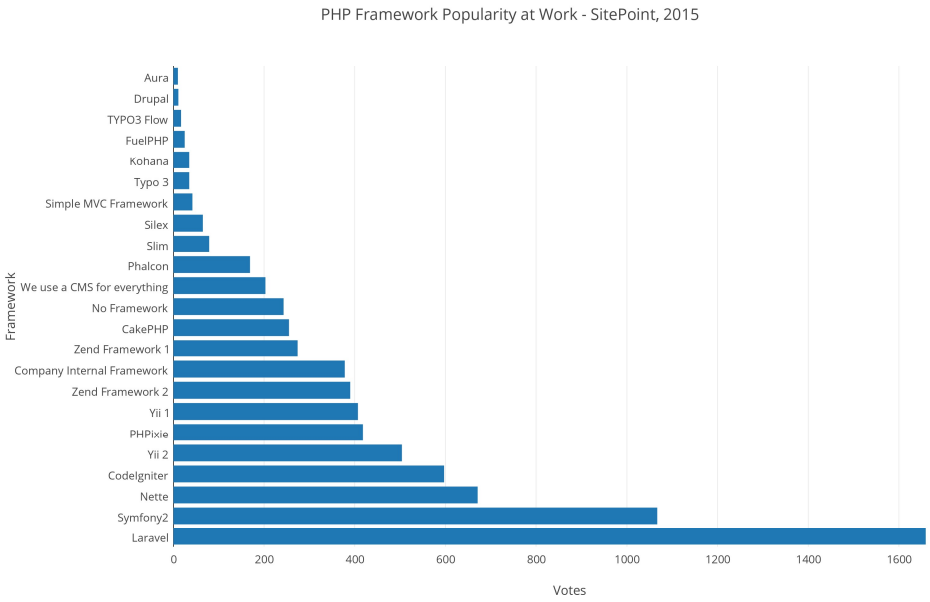

FIGURE 16. Framework popularity (Sitepoint 2015a).

Most popular options at the time were Laravel and Symfony (See Figure 16). Based on previous personal experience, Laravel was chosen as the project framework.

3.4 Laravel

Laravel is often referred as a full-stack framework. This is because it handles everything from serving web pages through database management down to HTML generation. Laravel framework has been created with PHP and runs on every web server capable of fulfilling the PHP version requirements, 7.0 to this date. (See Figure 17).

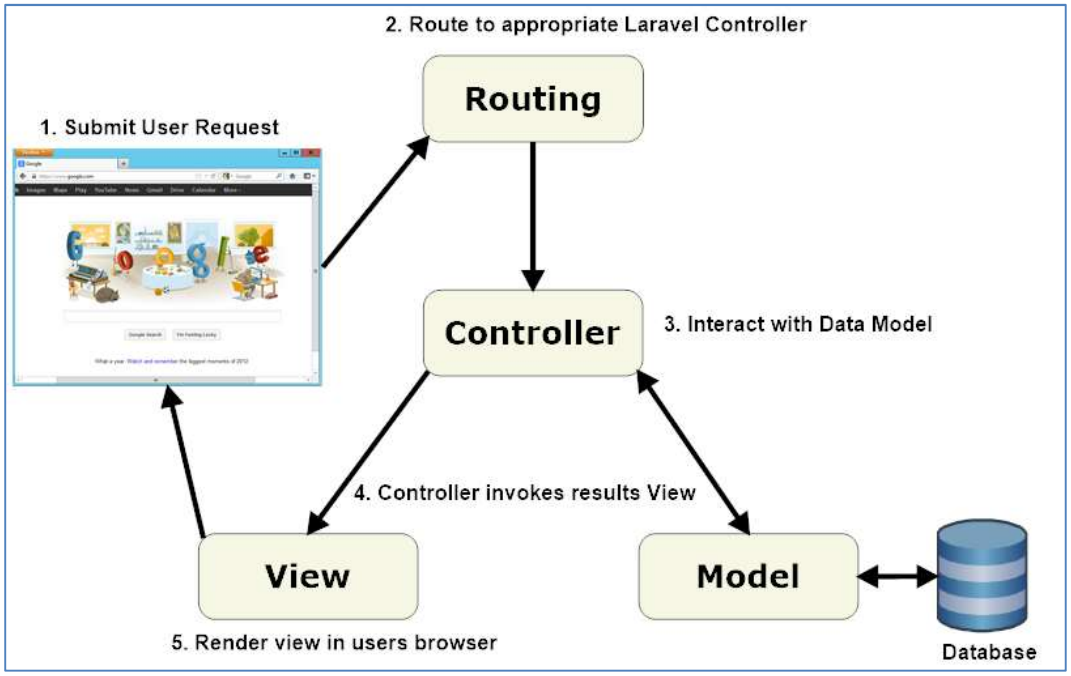

FIGURE 17. Laravel framework architecture (Laravel 2017c).

3.5 Laravel directory structure

The default Laravel application structure provides a clear starting point for both large and small applications. Laravel forces almost no restrictions on

where any given class or code is located - as long as Composer can autoload the class. Composer is a dependency manager for PHP. Composer will manage different dependencies required on a project and will pull in all the required libraries and manage them in one place.

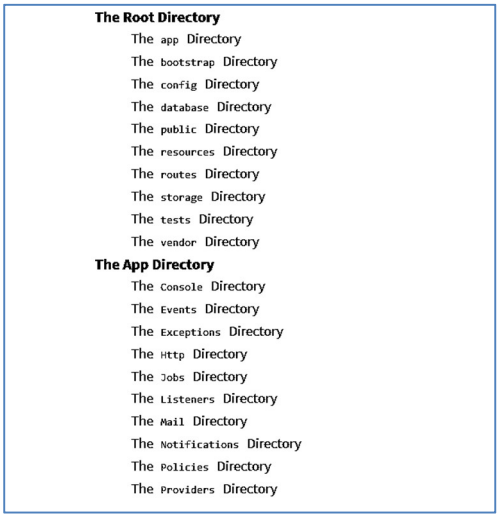

FIGURE 18. Laravel directory structure (Laravel 2017a).

The **app** directory contains the core code of applications and is explained in detail later. It contains application specific functionality such as controllers, models and jobs. (See Figure 18).

The **bootstrap** directory contains files that start up the framework and conFigure autoloading tasks. This directory also contains a cache directory for performance optimization such as the route and services cache files. In short, bootstrap can be described as the starter motor of the application.

The **config** directory contains application spesific configuration files such as creating database connections.

The **database** directory contains database migration and seeds. These are used to create database tables and seeding with test data.

The **public** directory is the www-root of an application. It contains the original index.php file, which is the entry point for all requests entering your application. This directory also houses external stores such as images, JavaScript, and CSS. This structure hides all the application logic from the browsers point of view as folders below public in the directory structure cannot be accessed.

The **resources** directory contains application resources. That includes views, language files and any uncompiled external assets such as JavaScript files.

The **routes** directory contains all HTTP route definitions in application. Every route must be registered in web.php -file before it can be accessed in the application.

The **storage** directory contains compiled Blade templates, sessions, file-based caches, and other cache files generated by the framework. The app directory can be used to store files generated by the application. The logs directory contains application's log files as the name implies.

The **tests** directory contains automated tests for the application.

The **vendor** directory is reserved for Composer dependencies.

(Laravel 2017a).

3.6 XML

> *“Extensible Markup Language (XML) is used to describe data. The XML standard is a flexible way to create information formats and electronically share structured data via the public Internet, as well as via corporate networks.*
>
> *XML code, a formal recommendation from the World Wide Web Consortium (W3C), is similar to Hypertext Markup Language (HTML). Both XML and HTML contain markup symbols to describe page or file contents.” (What Is 2017d).*

The data structure of XML data is defined as self-describing or self-defining. This means that the structure of the data is defined with the data. The receiver does not need to know the data structure beforehand; it can be parsed when the document or data arrives.

An XML document is built on elements and tags. XML can contain elements within elements, and the outermost element is known as the root element. This allows XML to have hierarchical structures. The names and tags describe the content of the elements and the relationships between the elements. (See Figure 19).

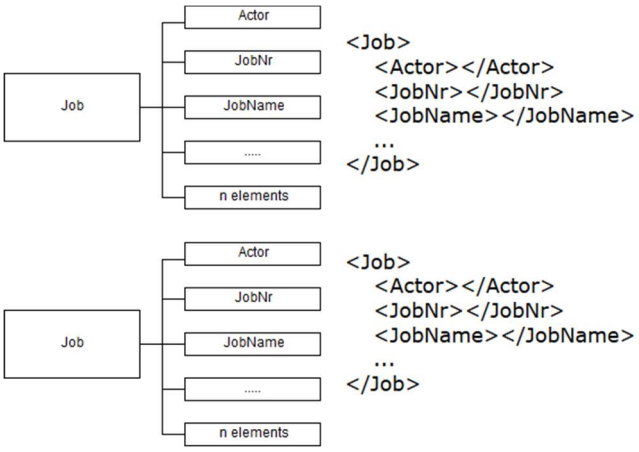

FIGURE 19. Example XML Structure

There are many forms of XML moving back and forth. A properly marked XML which format complies with the XML specification is considered to be well formed. Well formed XML documents can be read and understood by generic XML parsers when it is properly marked up, and elements are properly nested. XML also supports the ability to define attributes for elements and describe characteristics of the elements in the beginning tag of an element.

The practical applications for XML are infinite. For example, manufacturers might agree upon a standard or common way to describe the information about a product or data and then describe the information format with XML code. The standard way of describing any data makes data transfer and automated comparisons between systems applicable.

iFlow will be using XML to make a standard, easily understood mark-up and definition for data transfer between various production systems. On top of that it is easy to create and export from different databases as every participator has different resource planning softwares.

XML files will be transferred over secure FTP (SFTP) protocol to iFlow server where scheduled jobs (cronjobs) will process the incoming files.

4 FRAMEWORKS

> *“The purpose of a framework is to improve the efficiency of creating new software. Frameworks can improve developer productivity and improve the quality, reliability and robustness of new software. Developer productivity is improved by allowing developers to focus on the unique requirements of their application instead of spending time on application infrastructure.“ (Cimetrix 2017a).*

By using different frameworks and libraries software development cycles become more predictable since they provide a standardized code format and structure. This approach applies to generic services needed by most applications such as configuration, logging, different database access functions and caching. They provide methods that simplify development and make sure that all developers access functionality in same way instead of everyone writing their own solutions.

> *“Frameworks reduce the software development effort: Software applications consist of code related to the business logic and code related to the infrastructure that holds the application components together. When development projects include a framework, developers will spend less effort on the infrastructure and have more time to work on the business logic.”(Lansa 2017a).*

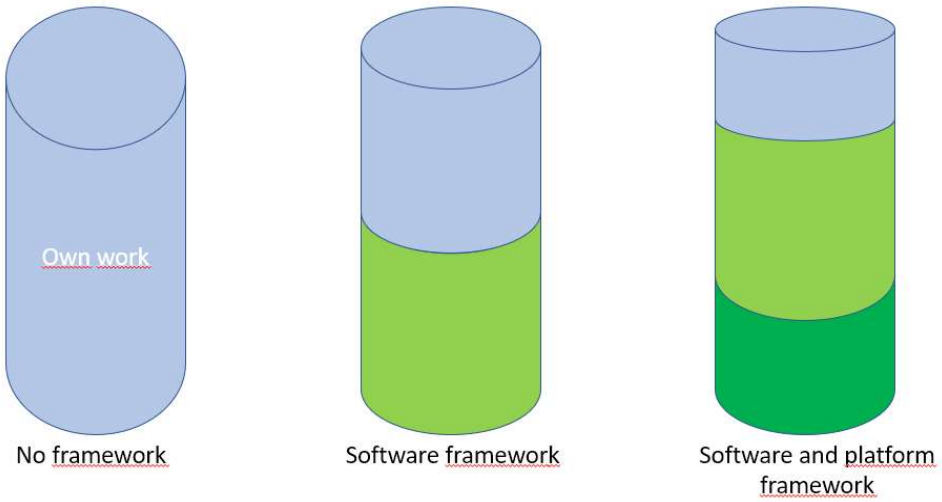

FIGURE 20. Illustration of required work

There are several classifications of frameworks (See Figure 20) depending on how comprehensive they are and how much they could speed up development process.

In conclusion, by using software frameworks, developers do not waste time reinventing the wheel. Also, less experienced developers will build better quality software by adding to a framework, rather than writing all (potentially buggy and messy) code from scratch.

4.1 Framework subcategories

Software frameworks are the models, templates, and libraries that we discussed earlier. They manage discrete components of applications. Developers use these frameworks to build applications.

Application frameworks are partially built applications. These frameworks are applications without actual business logic. Developers add the business logic to the project in order to complete the application.

Solution framework is a complete, working application. They include both business logic, backend- and frontend code. They provide a complete solution and may require some configuration and integration to activate in

production. Also, solution framework may require a some development efforts due to different environments.

Library frameworks are specific collections that provide some limited functionality to the project. Examples are JavaScript libraries (jQueryUI, Bootstrap) in the user interface layer. Application frameworks (CakePHP, Zend Framework, Laravel) in the business layer.

4.2 MVC

MVC is an acronym for model-view-controller -architecture. The model represents the data and only data. The fundamental idea behind MVC is that the model does not depend on the view or the controller. The view is used to display model data and controller handles all the actions the user does inside the application. (See Figure 21).

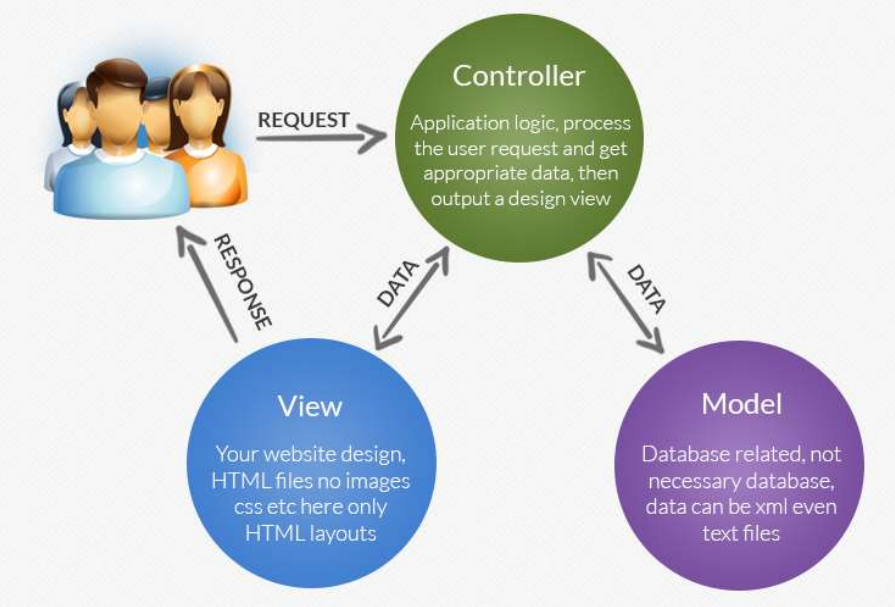

FIGURE 21. MVC logic

Utilizing MVC design also eases code maintentance and upkeep costs. Developers who are familiar with MVC logic and methods can pick up the application logic quickly by inspecting controllers. Also there is no need to know exactly the logics of the database as long as they understand the models communicating with it. This separation of responsibilites allow more flexibility and code base should be easy to use and possibly re-use.

4.3 Framework benefits

According to Fayed, the primary benefits of object-oriented application frameworks are generated from extensibility, modularity, reusability, and control they provide to developers (See Figure 22). Framework modularity improves software quality by providing a standard design and minimizing custom design and implementation changes, which reduces the effort required to understand, develop and maintain software. (Fayed, 1997).

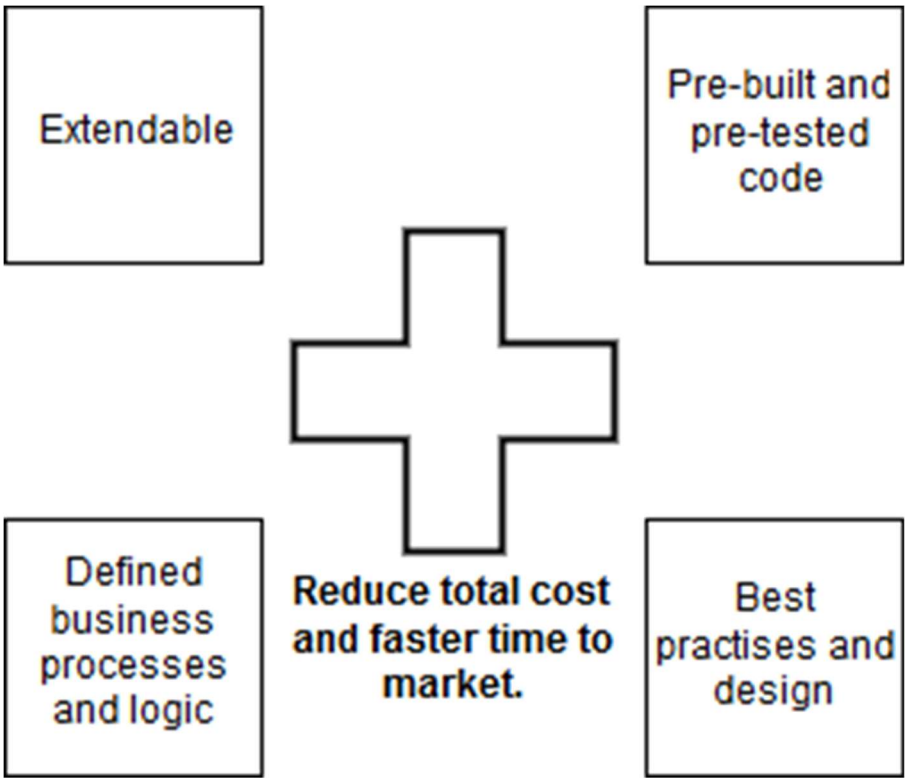

FIGURE 22. Advantages of software framework

As code is pre-built and pre-tested the development costs are minimal and on larger projects framework ensures that all contributors are using same standardized methods such as database handling. Internet community is providing easy extensions to most popular frameworks by building lots of different modules from where project can choose which to utilize to its direct needs. It is possible to build a basic working application just by selecting and picking functionality modules and configuring them to work with production databases.

4.4 Frameworks and coding efficiency

In a related study by Moreno-Lizarazu, three real-life applications were created using Model Driven Approach (MDA) on Java-Oracle environment with custom framework behind the development process. The three

applications proved to be highly maintainable and upgradeable, and developer efficiency improved about 25%. In this case, junior developers were able to leverage framework in less than 40h training. Additionally, the documentation improved.

The resources used by the framework (CPU, memory, disk space, database, and network traffic) were minimized. In the five years following the deployment of these three applications client satisfaction increased from 2.7 to 3.4 (measured on a scale from 0 to 4). Moreover, users showed increasingly positive replies during the same period. (Moreno-Lizarazu 2017).

5 IFLOW DEVELOPMENT METHOD

Due to digital disruption printing industry must embrace new techonologies and services to survive the changes digitalized world brings. First signs of digitalization is smaller quantities and reduced page quantities in products as advertisers are changing the focus from paper to web. This creates overcapacity, lowering prices, increases competition and this leads to lower margins and need for more efficient workmodels.

One of these challenges include adapting modern agile on top of old waterfall project models, described in following subchapters and as 3rd research question.

5.1 Waterfall model

The waterfall model is a sequential design process, used in software development processes. The progress through the process is seen like a waterfall falling through the different phases of project. This process is illustrated in Figure 23.

The waterfall model was initially created and adapted in the manufacturing and construction industries. These are structured physical environments and in which after-the-fact changes are costly, if not impossible. At that time no formal software development methodologies existed, so an originally hardware-oriented model was adapted as a base model. (Ebooklibrary 2017a).

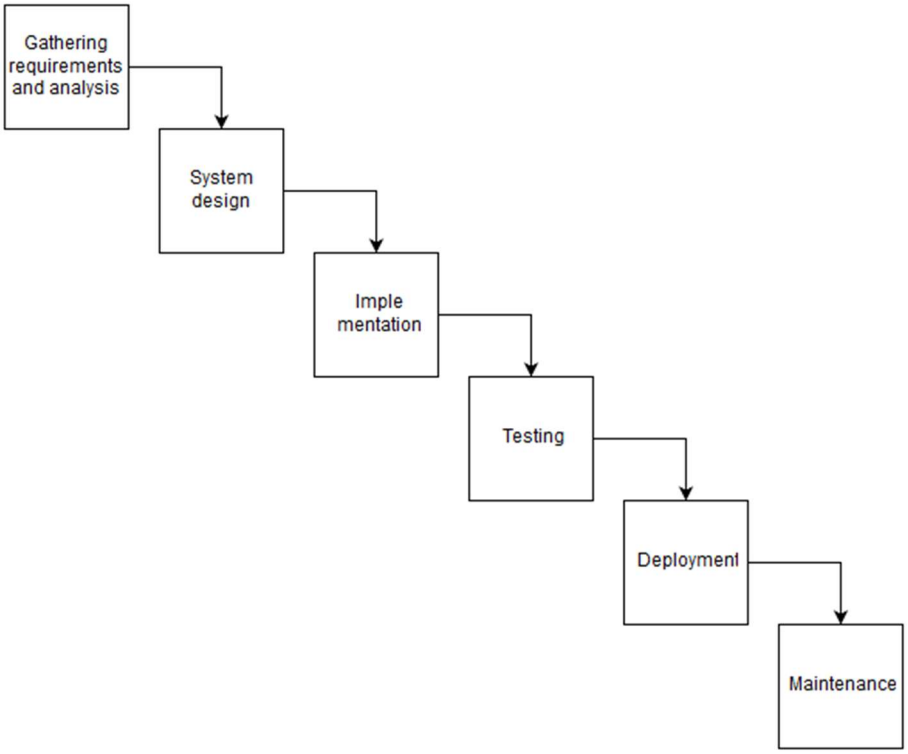

FIGURE 23. General overview of Waterfall model

5.1.1 Advantages and disadvantages of waterfall model

Using waterfall approach has some advantages as well as disadvantages. It can be used on many projects, especially with smaller scale. But requirements are prone to change during the project, especially larger projects, the scaling ability and readiness to answer changes is poor.

In overall waterfall model is simple and easy to understand. Due to this, it is easy to manage as each phase has specific milestones and processes before moving to next phase. One of the major disadvantages is that no working software or product is produced until late development cycle. It is very difficult to go back in phases and change something that was not well-thought during the concept stage.

There is also risk involved, does the process create a working product and more importantly; is it something the customer(s) want? As the constant

iterative process with customer validation feedback loops are missing the risk of missing out something important is high. There is a risk that once the product is out from development and any failure or changes occur then the cost of fixing or modifying are most likely high as application logic has been built on top of or relying on the issue.

5.2 Agile software development

Accoring to Cohn, many software development organizations are striving to become more agile. Successful agile teams are producing higher-quality software that better meets user needs more quickly and at a lower cost than are traditional teams. (Cohn 2009, 3).

The birth of Agile methods came from real-life project experiences with waterfall model. The most visible change is that most agile projects start coding for production immediately after designing or even during designing. The goal is to produce minimum viable product (MVP) as soon as possible and update or upgrade its functionality during additional iterations. Customer feedback loops are an essential part of agile development to ensure that the final product is answering to a specific need (See Figure 24).

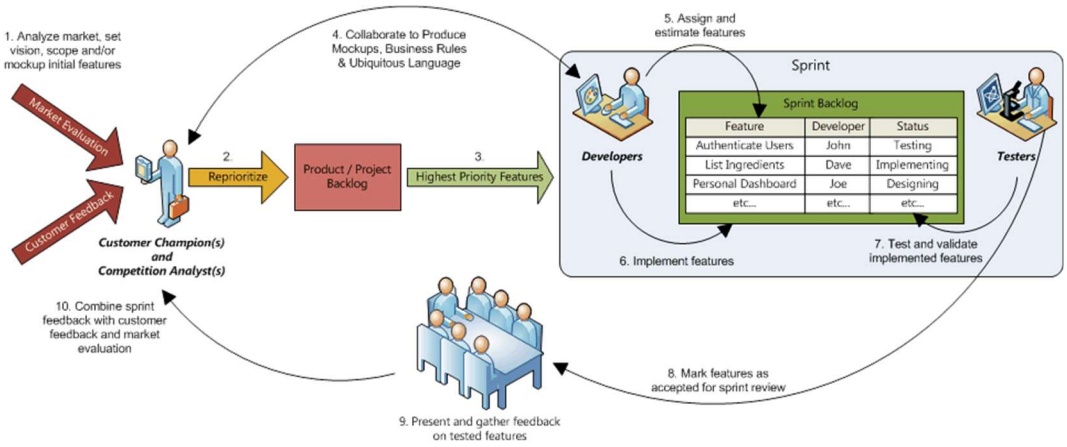

FIGURE 24. Agile Development Process

The core of the process is rapid response to changing requirements which come from the customer or end-user feedback loops.

5.2.1 Design sprint

> *"The design sprint is a multi-day process for answering critical business questions through design, prototyping, and testing ideas with customers." (GV 2017).*

A guide made by Google Ventures (GV) gives a rought timetable for design sprint contents.

Day 1: Map out the problem and pick an important place to focus.
Day 2: Sketch competing solutions on paper.
Day 3: Make decisions and turn your ideas into a testable hypothesis.
Day 4: Hammer out a high-fidelity prototype.
Day 5: Test prototype with real live end-users.

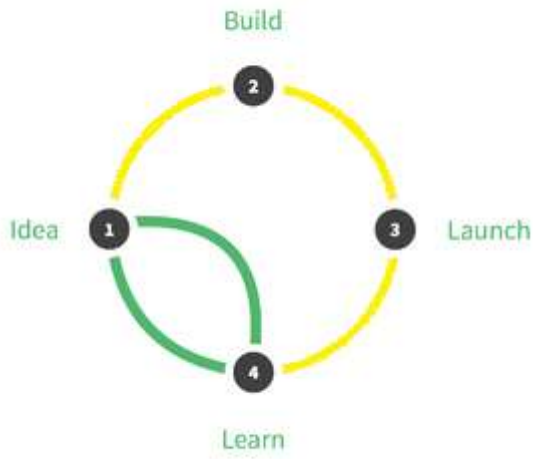

FIGURE 25. Design sprint process

Design sprint is based on building a feature or functionality and getting a response from the built feature. This loops keep on repeating itself until the product is complete (See Figure 25).

5.2.2 Development sprint

Development sprint is the core of agile process. Duration of the sprint varies from one week to four weeks, longer sprints are not advisable since they tend to lose focus when time increases. The final product functionality is split up in small pieces and tasks, called the backlog. In the beginning of the sprint the development goals are taken from the backlog. These do not need to be massive steps but rather small steps towards the final product. The purpose of the sprint is to add some functionality to the product and evaluate its functionality. By building the application layer by layer change requests or direction changes are handled as soon as possible keeping the costs low. The line-up of the sprint team varies by development technique used but usually involves few developers and a coach.

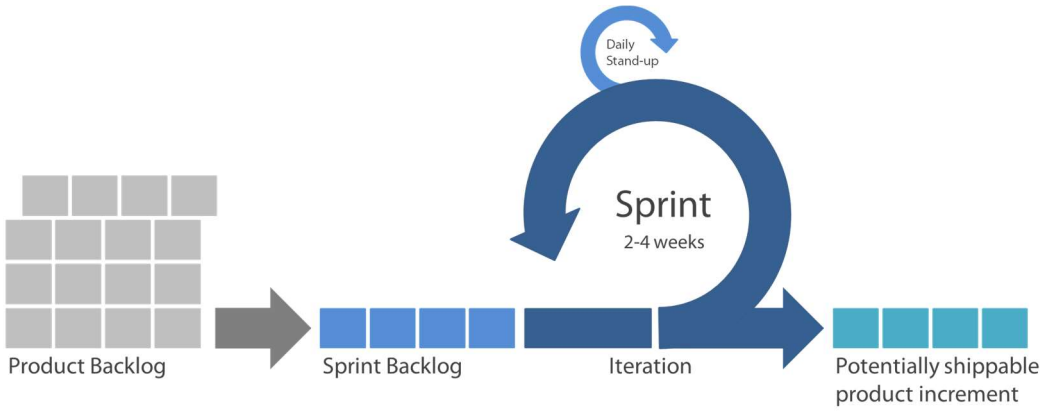

FIGURE 26. Development sprint (Diyoron 2018a).

At the end of each sprint team has produced a tested and usable increment to the existing software and a sprint review is kept. During sprint review the project is assessed against the produced goal and the initial target. If necessary, new items will be added to the backlog and the process starts again from start (See Figure 26).

5.2.3 Minimum Viable Product (MVP)

Minimum Viable Product (MVP) is a product which has minimum required features to enter the market. In iFlow -case MVP is a view which can display production status data from database. MVP is usually used to gather initial feedback from actual users or customers to define next development steps and to verify that the chosen solutions and paths are leading to desired result. Building a MVP also minimizes the risk as if the product proves to be a failure or does not meet the customer requirements it can be either redesigned or scrapped in an early stage (See Figure 27).

In software development MVP is usually used for gathering feedback and testing the use case after a successful proof of concept -has been done. It is also criticized for leaving the testing work and debugging to the end-users.

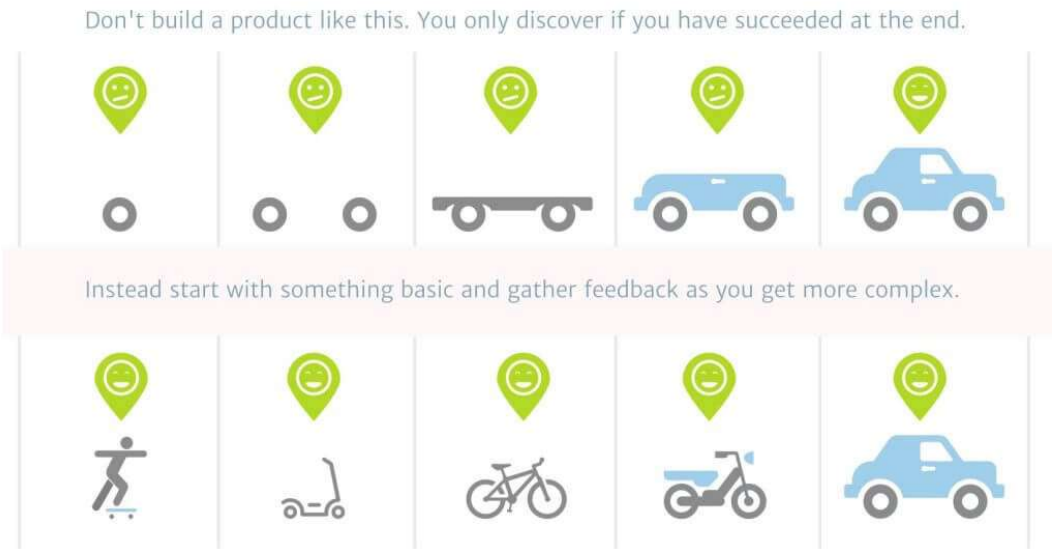

FIGURE 27. Development through MVP (Reinvently 2019a)

5.3 Scrum

Scrum is one the most used agile development method. The method is based upon different team member roles illustrated in Figure 28 inside the circle.

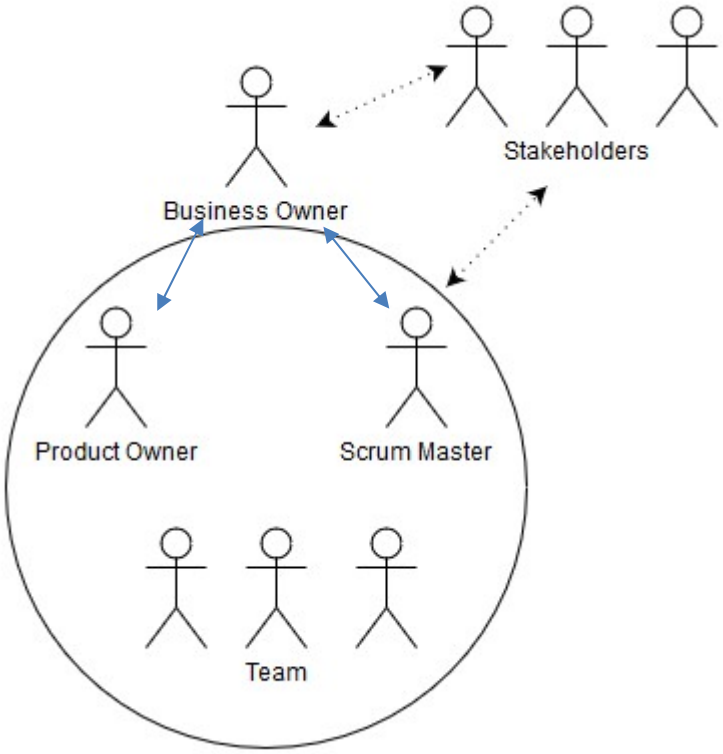

FIGURE 28. Scrum team

Typical Scrum team consists a product owner, scrum master and the development team. Business owner and different stakeholders are not considered as a part of the team. Product owner and scrum master handle all communications with external stakeholders so the development team do not have any external disturbances during development sprints.

5.3.1 Product backlog

Product backlog is a prioritized features list. It contains a short description of all functionality what the product should have. Prioritizing the product backlog is on the responsibility of a product owner. A typical backlog consists of features, bugs, design or research. The backlog is allowed to grow and change during the development.

Before each development sprint the scrum team takes items on sprint planning meeting. For this meeting, the product owner has prioritized the backlog items but the team determines which items they can complete during the sprint. These items become the targets for the sprint and cannot be changed during the sprint. New or changed backlog items can be taken on next spring planning meeting.

An example of product backlog is illustrated in Figure 29.

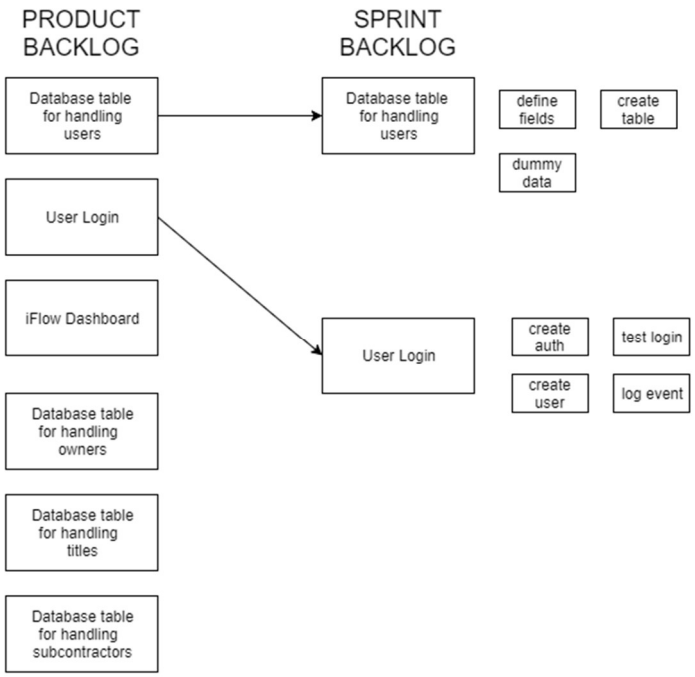

FIGURE 29. Product and sprint backlog

5.3.2 Sprint backlog

Development teams chooses sprint backlog items from the product backlog during the sprint planning meeting. This is illustrated in the figure 21. If necessary, the product backlog items are broken into smaller entities allowing team to distribute work. While putting the backlog together the team estimates hours they need to work in order to complete decided features. These hours should not exceed the total duration of the sprint.

During the sprint team members update the backlog and hour estimates as work progresses. This results in a sprint burndown chart, Figure 30.

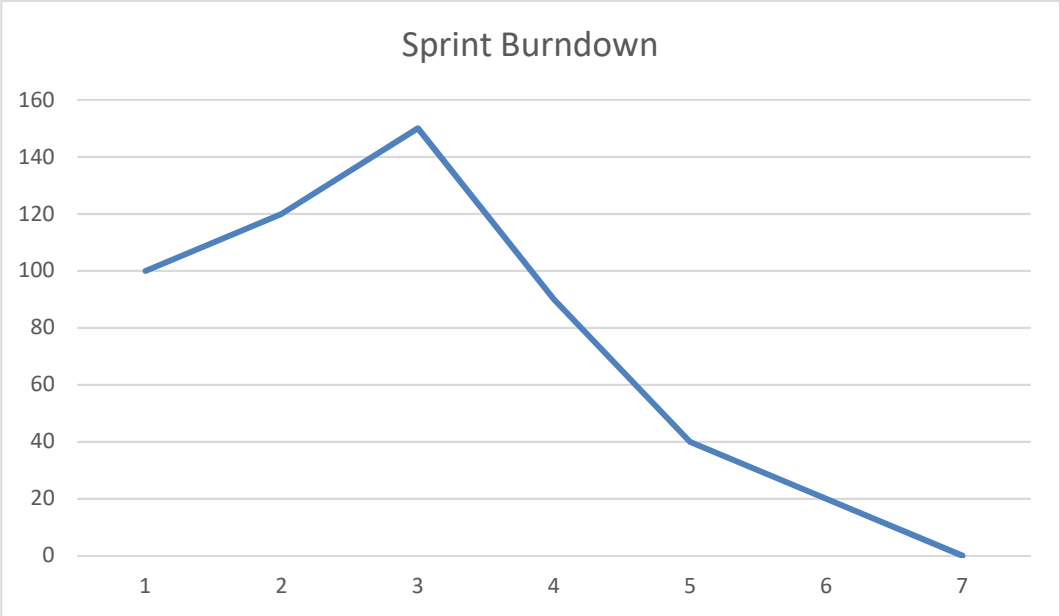

FIGURE 30. Sprint burndown, hours by days.

5.3.3 Product owner

An essential part of the Scrum-team is product owner. Product owner has a clear vision what he or she wishes to build and then relay that vision to the team. In agile world requirements can and will be changed during development process and product owner has the final word on these changes.

Product owner also prioritizes the product backlog and actively maintains the backlog for scrum team. This way team always has a clear view on the backlog items they should be developing.

FIGURE 31. Product owner (AgiVetta 2018a).

Compared to traditional methods, product owner could be described as a project manager, keeping all the strings attached (See Figure 31).

5.3.4 Scrum Master

Scrum master acts as a API between product owner and development team and facilitates the agile development process. Scrum master can be a part of the development team, especially in smaller organizations.

The main responsibility for a Scrum master is to facilitate the scrum - process. He or she is responsible that the team used the scrum - techniques and processes correctly and if necessary, helps the team to adopt the principles of Scrum.

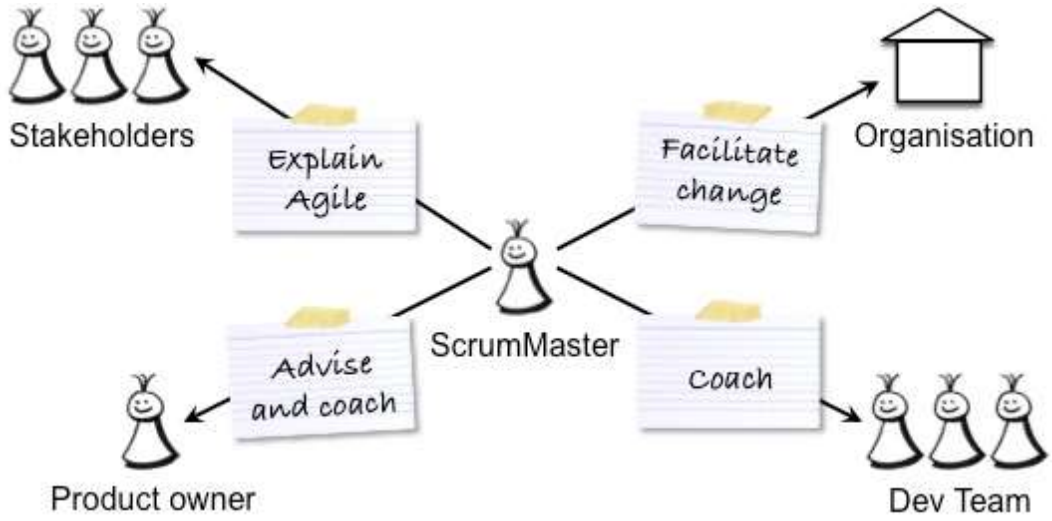

FIGURE 32. Scrum Master (GravityIT 2018a).

Scrum master does not manage the team, in Scrum the team itself is self-managing (See Figure 32). Scrum master can also act as a chairman for the daily scrum meetings.

6 IFLOW DEVELOPMENT

User stories are used to build an overview of the desired application functionality. To avoid user stories becoming too large and complicated they should focus on one specific function. If user story is too complex, it can be broken down to story points. Story points describe achievable functionality milestones in a larger entity.

User stories can be created on a whiteboard or flipchart and do not require any spefific template or software.

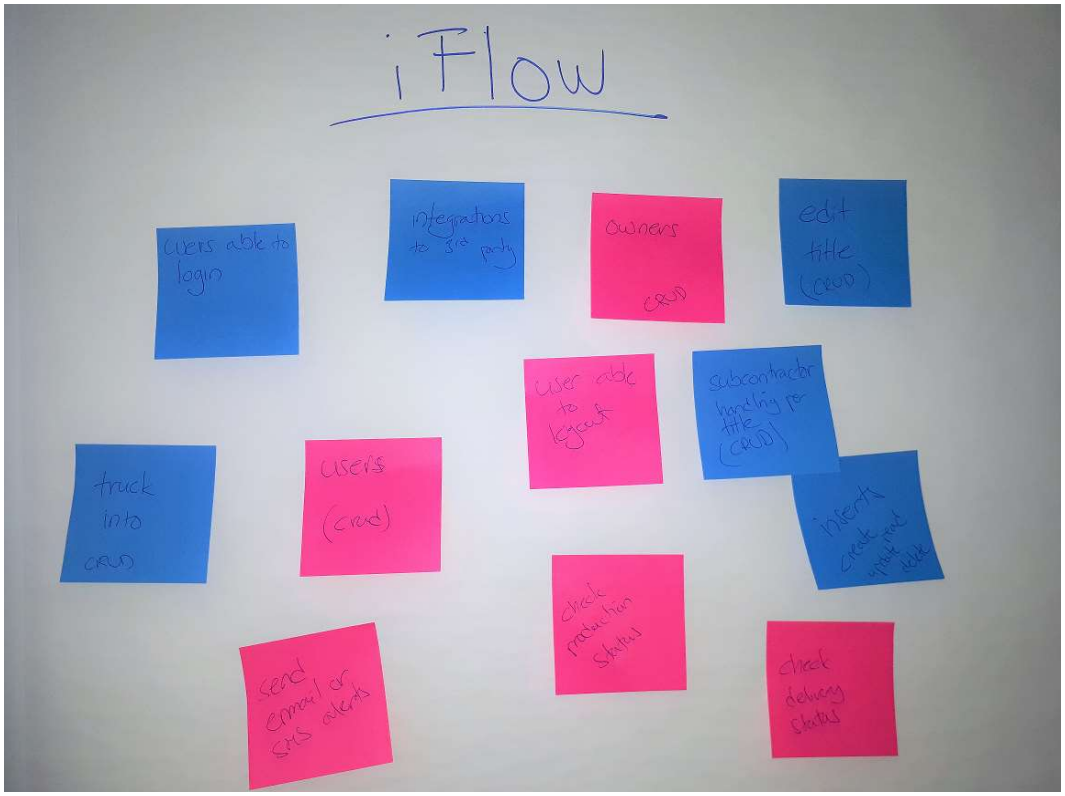

FIGURE 33. iFlow user stories

Figure 33 demonstrates a simple user story planning board created for iFlow application. These are use cases the application is required to perform.

6.1 Creating backlog

User stories are broken down to simpler backlog items. For example, user story ‘users able to log on’ can be broken down to backlog items such as:

1. Login – user interface
2. User and password database
3. Authentication
4. Authentication failure
5. Authentication success
6. Reset password

If necessary, development team can still break these items into smaller entities to ensure they are able to deliver the item at the end of the development sprint.

7 INITIAL SETUP

The selected methodology for building application was Scrum, described in chapter 5. Due to limited size of the development team some persons had multiple roles. This is well within Scrum definitions.

Development team roles were assigned to title owner, subcontractor and main contractor. First task was to create user stories based on the main idea of the application. These stories were broken down to form initial backlog for development sprints.

7.1 Deployment environment

Deployment environment can be divided in to three different sections. A local environment, test environment and production environment. Deployment environment is also commonly referred as development environment.

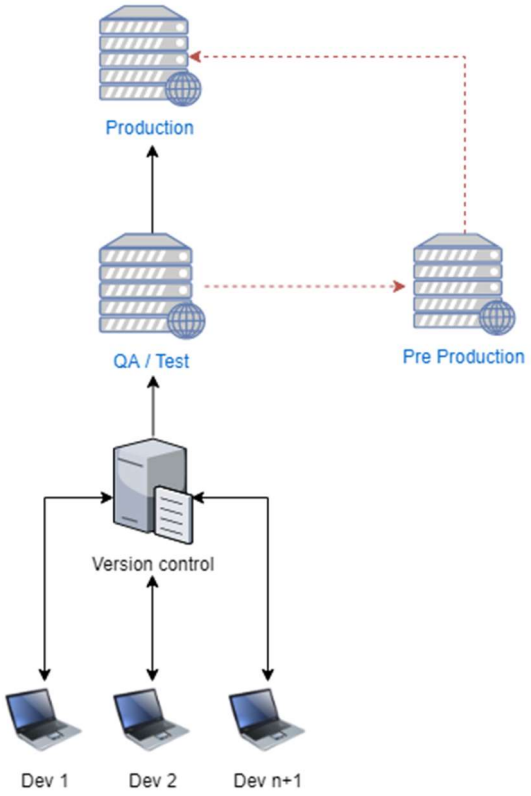

FIGURE 34. Deployment environment

Figure 34 demonstrates deployment environment. Due to small scope of the project separate pre-production environment was not deployed. In larger projects pre-production environment is copy of production environment where final validations are made before pushing new version to actual production environment.

Pre-production should be a direct copy of actual production environment with full integration support. Test environment can be a partial representation of production with limited integrations.

7.2 Local environment

The development environment was set up on a local laptop computer. Apache web server, MySQL database and Laravel framework were installed to provide running environment for the application. Zend Studio

was used as an integrated development environment (IDE) (See Figure 35).

Local development IDE Zend Studio which is a commercial product and one user licenses cost around $150. There is also open source - development editors available but lacking most of the commercial version features.

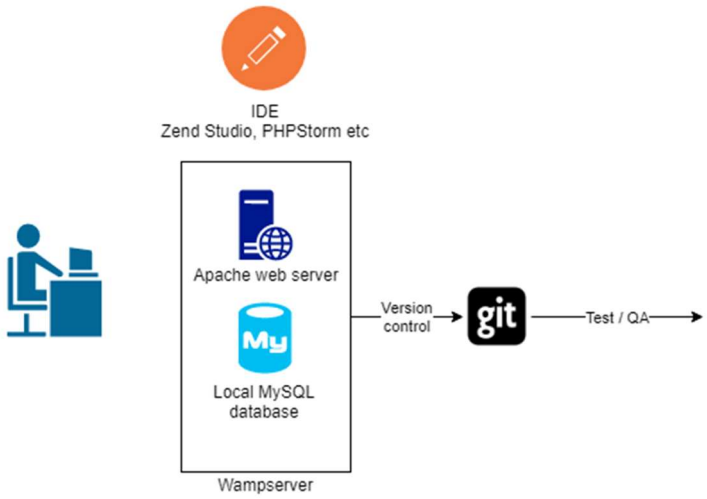

FIGURE 35. Local development environment

7.3 Test environment

Due to small scope of the project the test environment was created on the same server as production (See Figure 36). Separate Apache virtual host was created to respond at address http://test.iflow.helio.charleroi. This virtual host serves application separately from main production, on a separate container.

Test application has a separate MySQL database connection on database server to ensure no real production data is in danger.

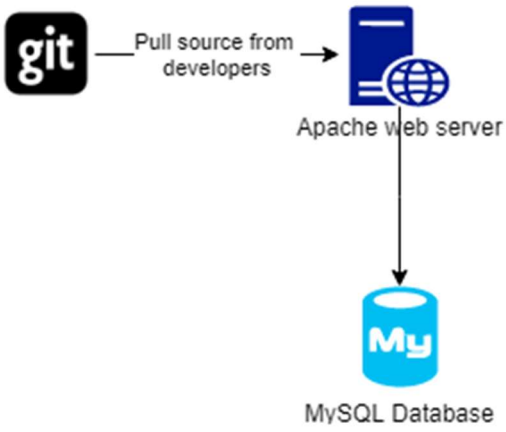

FIGURE 36. Test environment

After testing and validation on test server the code is ready for production. If problems arise, they are strictly restrained on testing environment and developers receive a list of errors. There are multiple testing tools available which automate application testing but due to project scope no real test automation was required.

7.4 Production

Production environment, as described in Figure 7, was set up inside Helio Charleroi intranet network consisting application server, database server and some 3rd party servers which are used to gather various production data. These 3rd party contributors communicate via secure FTP -connection to iFlow -server. To increase security, the iFlow server is configured to let only certain IP addresses connect and initiate XML-transfer (See Figure 37).

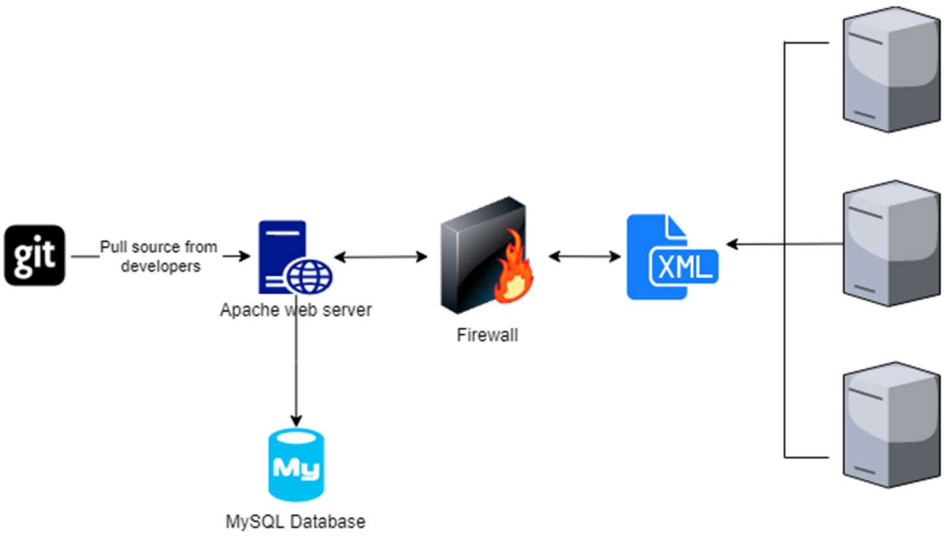

FIGURE 37. Production environment

The difference between test and production iFlow -environments is that test does not have external 3rd party connections. The production environment was built completely with open source software packages making it extremely cost effective solution.

7.5 Connecting 3rd party

3rd party data sources are essential for the application to work. As subcontractors provide different parts for the main product it is necessary to display the status and stage of their contribution. Due to various production management solutions, the data link is done with XML -files. (See Figure 38).

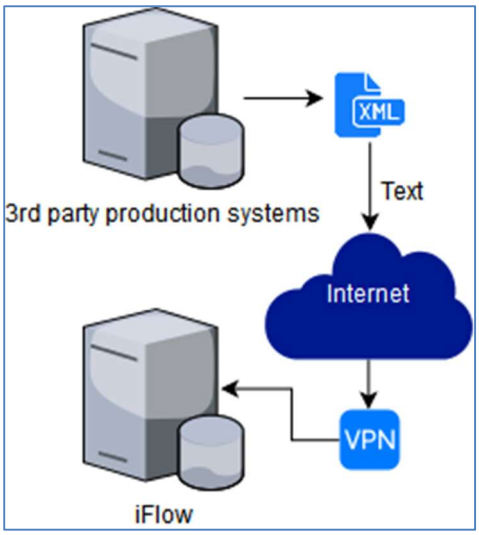

FIGURE 38. XML transfer from 3rd party to Helio Charleroi

The connection between 3rd party and iFlow is maintained via static VPN (Virtual Private Network) tunnel which allows the sending party to connect Helio Charleroi network.

7.5.1 XML Structure

Due to various ERP (Enterprise Resource Planning) and MIS (Management Information System) systems in printing industry, the transfer format must be simple for extraction and generation.

The format chosen for iFlow is XML. XML is used in various file-based data transfers between systems due to its easy mark up and readability (see Figure 39). The XML can be generated from almost any source through scripting and is not dependant of the source or receiving system. Most common use is to parse XML to respective systems in both ends of the data transfer.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Job>
    <Actor>HCH</Actor>
    <JobNr>1639740</JobNr>
    <JobNrInt>22676</JobNrInt>
    <JobLib>HELLO 1450/UK/CA1/56P</JobLib>
    <Press>CR1</Press>
    <PrintingOrder>52142</PrintingOrder>
    <RunNr>1</RunNr>
    <Status>3</Status>
    <OrderedCopies>338650</OrderedCopies>
    <PrintedCopies>361100</PrintedCopies>
    <ProducedPallets>338650</ProducedPallets>
    <DateHourStart>1474710753</DateHourStart>
    <DateHourEnd>1474762140</DateHourEnd>
    <NumSeq>1474762140</NumSeq>
    <IdJobData>6785</IdJobData>
    <IdSeq>7181</IdSeq>
</Job>
```

FIGURE 39. XML structure

The XML fields are illustrated in Table 1. There are 9 mandatory fields used in data transfer. Seven optional fields are used to carry extra data but are not required in all transfer functions.

The job number is used as the main key in data transfer. Every party contributing to the title is using the same general job number where JobNrINT, internal job jumber, can vary between parties.

Actor is used to define which party is submitting the data. Every contributor to the title is considered as a subcontractor so they can be identified from subcontractors -database table and assign work done as stated in the XML file.

TABLE 1. Data transfer XML Structure

Child	Description	Mandatory
Actor	Current actor supplying data	1
JobNr	**Job number. Main identifier.**	**1**
JobNrInt	Internal job number the system uses.	0
JobLib	Job name	1
Press	Press or production machine in question	1
PrintingOrder	Internal printing order number	0
RunNr	Run number	0
Status	Status flag	0
OrderedCopies	Copies ordered from production	1
PrintedCopies	Copies produced so far. Good + waste	1
ProducedPallets	Good copies	1
DateHourStart	Production start timestamp	1
DateHourEnd	Production end timestamp	1
NumSeq	Internal numbering	0
IdJobData	Internal numbering	0
idSeq	Sequence id, can be used to sort data sequences	0

8 BUILDING APPLICATION

The building of application was done in one week sprints. Most of the data was already available in different databases. Building a master iFlow -database was the first task and gather internal production data to one spot before connecting 3rd party data sources.

8.1 iFlow application logic

Application logic is centered around title. The title is the final product delivered to the customer. A title has an owner and can include various parts and comments. Helio Charleroi's existing ERP-solution, GPAO, handles production data around jobnumber and title and this approach was selected on base of iFlow -logic as well. It has numerous advantages, especially building an equivalent data relation and synchronization easily.

A part can have a subcontractor and provides information on extra fills on the title. A workflow is used to track title or part specific actions, such as receival of external materials or stitching a part into the main title.

Database tables, fields and relations are described in detail on following sub-chapters and illustrated in Figure 40.

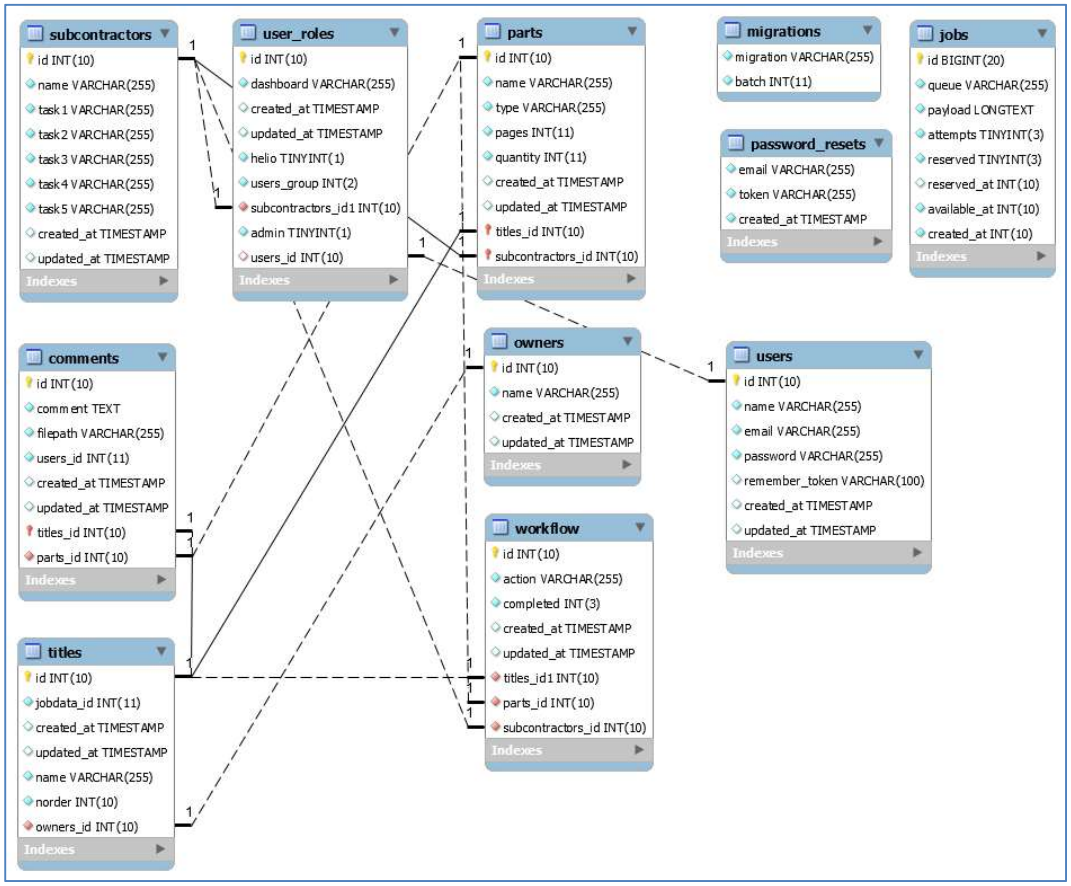

FIGURE 40. iFlow database structure

The database design was made with MySQL Workbench, a graphical tool for designing databases and visualizing relations between different tables. Data handling and testing was done with PHPMyAdmin which is a browser-based tool for inspecting and modifying MySQL databases.

8.2 Application structure

The main application view is the dashboard which connects to all operative functions. In the first phase there is only one generic dashboard for all users. Future development plans include customizable dashboard for users where users could select information blocks which are essential for their operations.

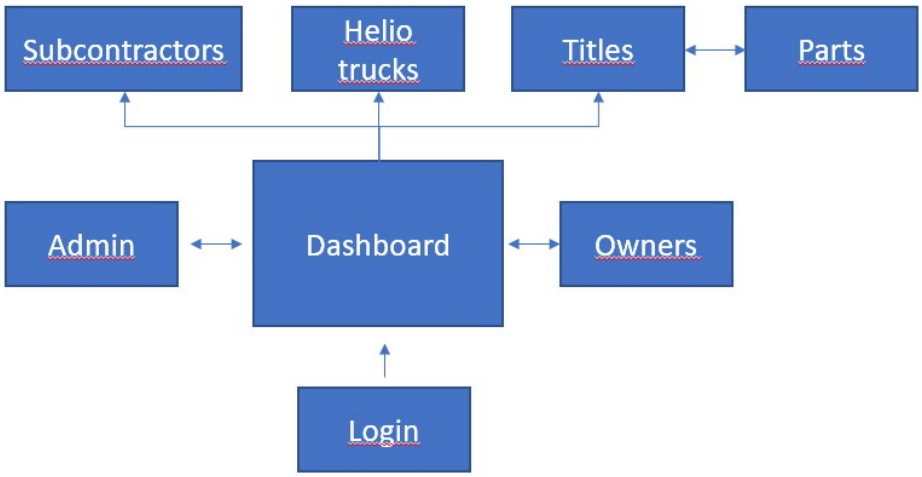

FIGURE 41. Application structure

Figure 41 shows the basic application structure which design was based on.

8.3 Dashboard

Dashboard provides an overall view of the weekly products defined into the system (see Figure 42). A script will look up automatically new titles inserted into the production planning system and add them on top of the list.

In development there were 4 stages of work defined: Data/copies received, M-R (Make Ready) status, Production status and delivery status.

These are the basic milestones of every printing job and reaching one milestone unlocks the new step in the process. For example, bindery cannot start binding before all the covers and runs have been received so information of their status is critical to the binding process.

Production and delivery status are automatically updated via Helio production management system and possibilities to monitor data and M-R are a case for further development. If there is no data and production starts both data/copies and M-R status are set to 100% since they are prerequisites for production.

Different stages can be added or removed if need arises.

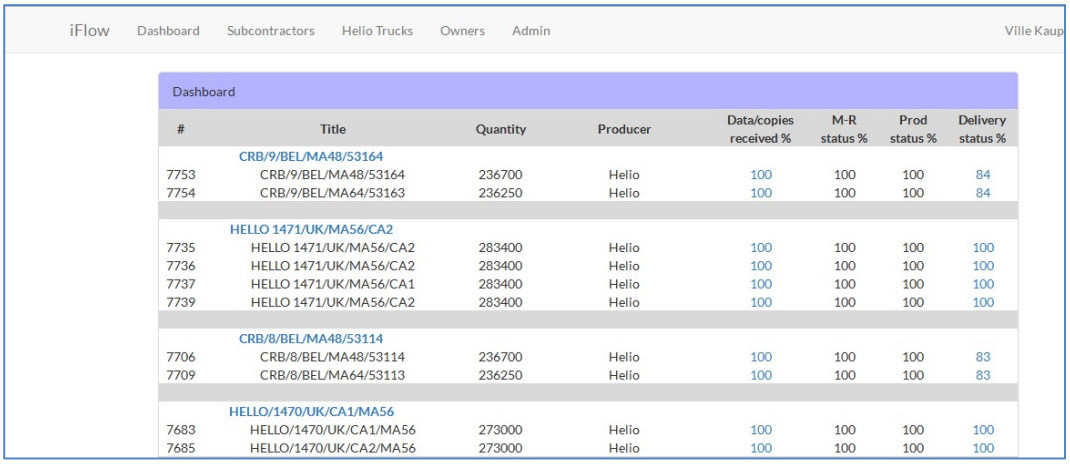

FIGURE 42. Dashboard

8.4 Detailed title view

Detailed title view provides a deeper view in the title and its parts (see Figure 43). On top is the title information. Second box provides list of title parts and their status. In most cases, the parts quantity equals the title quantity but in some special cases there can be for example territorial inserts which are distributed based on location.

Third box holds notes. A simple interface for adding note for a part with a comment and possible attachment. For example, if inserts have been damaged during transport the receiving party can insert a note of what has happened and when this has been detected. Possible reclamation cases are easy to track when appropriate note is made when something unsual happens during process.

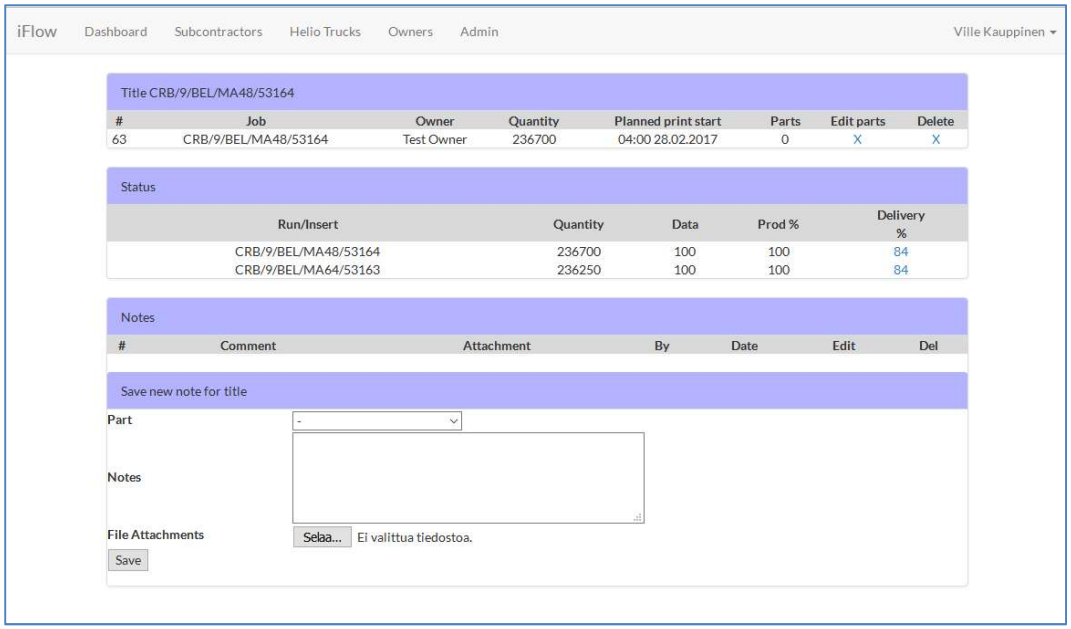

FIGURE 43. Detailed title view

8.5 Edit title

Edit title -view is a helper view to do basic edits on the title master data. Admin or owner can add different title parts and assign them to various subcontractors. Number of pages, quantity and external name can also be defined (see Figure 44).

The base title data come from the Helio production system automatically but there may be a need occasionally to give more detail or notes to the title itself.

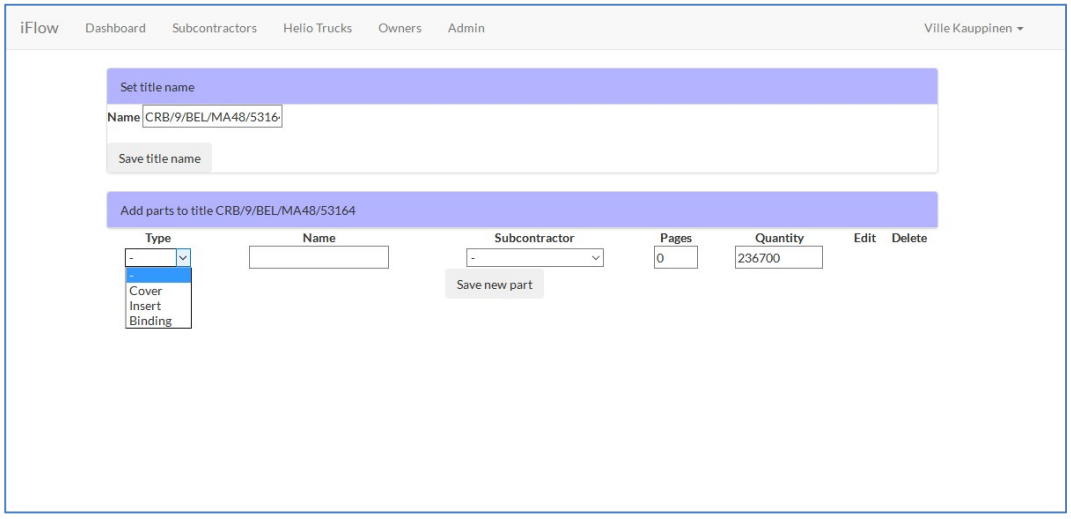

FIGURE 44. Edit title and add title parts

8.6 Owners

The owners view is used to add owners to the system. Owners own the title so mostly they are different publishing houses or their agents. The owner has a complete view and access to the data considering their title. The ownership is has a basic heritage function based on the title name. This means that the system will detect owner based on the title name when a new title is inserted into the production queue.

An owner cannot change the data regarding the title, it is a full read-only access. All the actions to titles require either an admin or helio -flag in the user object. This is because Helio Charleroi acts as a main contractor in titles and are in change of the printing process.

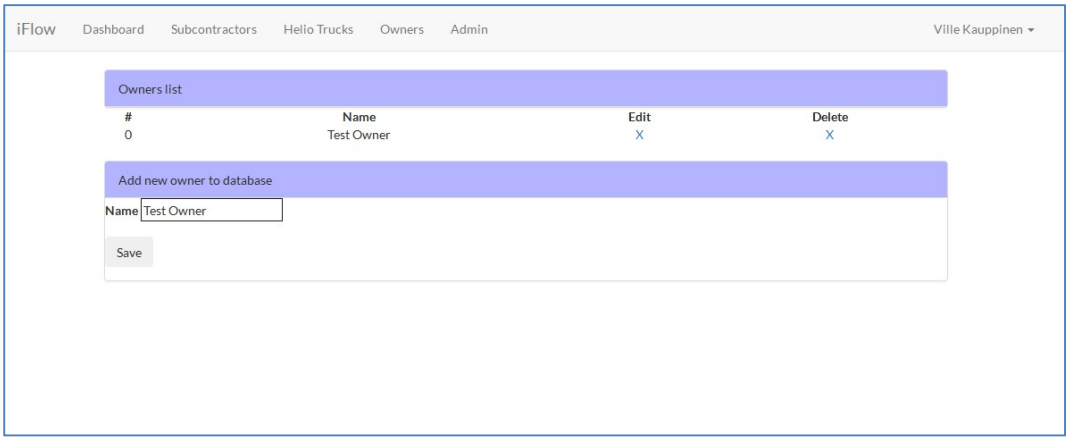

FIGURE 45. Title owners

Figure 45 shows title owner -administration template.

9 DATABASE STRUCTURE

The database was built around MySQL database. A relational database is usually made of tables are made of rows and columns. The database table is the basic component where all the data in a database is stored row by row.

Following chapters describe the structure and functionality of different tables used in application. Each table consists multiple rows and columns which are defined for storing specific data. Every table has one primary key column. The primary key is used to give unique identifier for each row in the database table. As a result, no two rows can have the same primary key value. Because of this, it is possible to select every single row by just knowing its primary key.

9.1 Titles

Titles - table handles the core data of the title. The relational jobdata has more attributes regarding the title. It is an internal table of the iFlow - system and can be accessed with iFlow so data duplication is not necessary. (See Table 2).

TABLE 2. Titles

titles		
Key	Value	Comment
id	INT	auto increment id
jobdata_id	INT, FK	relational id for jobdata from ERP
created_at	DATETIME	record created timestamp
updated_at	DATETIME	record updated timestamp
name	VARCHAR	title name
norder	INT	internal order number for ERP
owners_id	INT, FK	relational id for owner in owners-TABLE

9.2 Owners

Owners - table keeps track of different title owners. This can be used to give external access to title owners, e.g. publishers which can view all information of owned items. (See Table 3).

TABLE 3. Owners

owners		
Key	Value	Comment
id	INT	auto increment id
name	VARCHAR	owner name
created_at	DATETIME	record created timestamp
updated_at	DATETIME	record updated timestamp

9.3 Parts

Parts table stores information of title parts. These can include various inserts, runs or covers. The key connecting part to a title is foreign key field titles_id. Various parts can be connected to subcontractors TABLE indicating who is supplying the part via subcontractors_id -foreign key.

Various information is given for a part like pages and quantity. For example there may be a different regional cover for a magazine so half of the total quantity receives cover 1 and other half cover 2.

TABLE 4. Parts

parts		
Key	Value	Comment
id	INT	auto increment id
name	VARCHAR	owner name
type	VARCHAR	Part type, pre-defined types
pages	INT	Number of pages
quantity	INT	Quantity of part items
created_at	DATETIME	record created timestamp
updated_at	DATETIME	record updated timestamp
titles_id	INT, FK	relational id in titles-TABLE
subcontractors_id	INT,FK	relational id in subcontractors-TABLE

Table 4 shows the structure of ‘parts’ database table.

9.4 Workflow

Workflow stores different status of the title tasks in progress. Data is updated either via call or regular scripted interval and completed value is defined as percents. For example, make-ready is 90% complete or production of run 1 is 50% complete. (See Table 5).

TABLE 5. Workflow

Workflow		
Key	Value	Comment
id	INT	auto increment id
action	VARCHAR	name of the action
completed	INT	percent value
created_at	DATETIME	record created timestamp
updated_at	DATETIME	record updated timestamp
titles_id	INT, FK	relational id in titles-TABLE
parts_id	INT, FK	relational id in parts-TABLE

subcontractors_id	INT,FK	relational id in subcontractors-TABLE

9.5 Subcontractors

Subcontractors table saves the information of various subcontractors. There are slots for 5 tasks the subcontractor can perform, such as binding or transport. The primary key of the table is used to identify users and title parts the subcontractor is providing to the main title (See Table 6).

For example, one binder might have a task list:

1. Covers received
2. Runs received
3. Binding started
4. Binding ended
5. Title shipped

TABLE 6. Subcontractors

subcontractors		
Key	Value	Comment
id	INT	auto increment id
name	VARCHAR	name of the subcontractor
task1	VARCHAR	name of task performed by contractor
task2	VARCHAR	name of task performed by contractor
task3	VARCHAR	name of task performed by contractor
task4	VARCHAR	name of task performed by contractor
task5	VARCHAR	name of task performed by contractor
created_at	DATETIME	record created timestamp
updated_at	DATETIME	record updated timestamp

9.6 Comments

Comments TABLE is responsible of storing comments for a title or a part, or both. The comment is entered in free form via text field. There is also a possibility to attach a single file to a single comment. The path where the attachment can be found is stored to the table.

TABLE 7. Comments

comments		
Key	Value	Comment
id	INT	auto increment id
comment	TEXT	Free field text for comment
filepath	VARCHAR	Path to possible attachment
users_id	FK,INT	Comment creator id in users -TABLE
created_at	DATETIME	record created timestamp
updated_at	DATETIME	record updated timestamp
titles_id	INT, FK	relational id in titles-TABLE
parts_id	INT, FK	relational id in parts-TABLE

Table 7 shows the structure of comments -table.

9.7 User_roles

User_roles is responsible for storing roles for different types of users. Defined roles are presented in Table 8. Predefined roles are stored in user_roles database table. Different access levels can be defined for each view generating an efficient and light-weight access control solution.

The admin has access to everywhere in the application.

Helio represents the users inside the production facility of Helio-Charleroi. They are able to access and modify all views except admin-spesific functions.

Owners -role is created for title owners. They are able to access and modify views which belong to a specific title.

Subcontractor -role is for specific contributors, for example a cover, and they are able to access all views which belong to their part of the title.

Reader is a read-only role. They can access the system but cannot change any information.

TABLE 8. User roles -mapping

Role	Comment	ID
admin	Administrator, can access all views.	1
helio	Helio, can access and modify all views except admin-functions.	2
owner	Owner, can access and read all views in owned title.	3
subcontractor	Subcontractor, can access and read all views of own parts.	4
reader	Reader, can access and read all views. Designed for Helio internal use.	5

The roles are linked to a specific user by id-relation. Only one role can be defined per user. This data is stored in user_roles -table which is directly related to users -table (see Table 9).

TABLE 9. User_roles

users		
Key	Value	Comment
id	INT	auto increment id
users_id	INT	id in users-TABLE
subcontractors_id	INT	if provided the user has subcontractors role based on subcontractor -TABLE
dashboard	INT	designed for various frontpage dashboard paged
users_group	INT	group for user as described in Table 7.
helio	BOOLEAN	indicates a helio-user
admin	BOOLEAN	indicates an admin-user
read_only	BOOLEAN	indicates a read-only user
created_at	DATETIME	record created timestamp
updated_at	DATETIME	record updated timestamp

9.8 Users

Users table is responsibe for storing users and information related to user-object (See Table 10). This information includes created_at -timestamp, role_id and subcontractors_id. User object is combined from users and users_roles - table which determines the access level user has inside iFlow -system.

TABLE 10. Users

users		
Key	Value	Comment
id	INT	auto increment id
name	INT	id in users-TABLE
email	INT	if provided the user has subcontractors role based on subcontractor -TABLE
password	INT	designed for various frontpage dashboard paged
remember_token	DATETIME	record created timestamp
created_at	DATETIME	record updated timestamp

updated_at	BOOLEAN	indicates a helio-user

9.9 Migrations

Migrations table is an internal table of Laravel framework. It tracks status of different database migration actions.

9.10 Password_resets

Password_resets keeps track of user password reset -requests. This is an internal database table of Laravel framework.

9.11 Jobs

Jobs table is an internal table of Laravel framework. It is used to queue different jobs into a working cycle. This prevents system to run out fo resources because of multiple jobs running simultaneously.

10 CONCLUSION

The first research question was ‘How open-source systems can provide a competitive platform for digitalization projects’. The study presents a use-case for an open-source system with open-source programming framework. In conclusion and from personal experience with similar projects open-source fares well against most common closed-source platforms.

Regarding the environment, Linux-based servers are extremely stable and have excellent tools for setting up scheduled tasks. They can be operated from a terminal view which saves a bit of hardware resources. Most distributions are completely free of charge giving them an extra edge over commercial counterparts.

On top of low cost Linux servers are extremely reliable and require very little maintenance. Software package updates can be applied without rebooting the server itself resulting in practically zero downtime.

iFlow remains in production use today and has made the printing process a bit more transparent to the different parties involved. This has improved efficiency as the job owner does not have to gather data from different systems or by e-mail and same status can be seen throughout the organization.

10.1 Software frameworks and their impact on efficiency

Second research question was ‘What are software frameworks, and how they improve development efficiency’. Software frameworks are pre-built and tested packages of working code designed to ease application building. In this thesis work using framework made development a lot faster and easier.

This is because framework provides a lot of useful pre-tested libraries and packages which results in less time spent re-inventing the wheel. Common

functions such as database, view or model handling were provided by the Laravel PHP framework so developer does not need to worry too much about what goes where or how to implement X and Y. Framework also reduces possible technical debt. Technical debt is usually created by developers when they choose quick and dirty solution instead of using a better and more stable solution that would take longer to implement.

Frameworks also speed up maintenance process as they make software more transparent and easy to adopt. Every developer who knows the operation of Laravel framework needs a lot less time to be efficient with iFlow -project and with little preparation can alter, update or modify the functionality as it is based on common guidelines and practises. This results in smaller personnel risk as more people are qualified to read, understand and upkeep the code.

Overall conclusion is that using a framework can and will improve development and maintenance efficiency a lot.

10.2 Using agile methods in a traditional printing environment

Third research question was ‘Are agile methods effective in developing software for traditional printing industry?’ Agile methods provide an excellent base for developing software in traditional printing industry. The ability to adapt quickly to changing needs is a key component required in this environment. In this project there was a lot of moving factors and variables, like the 3rd party contributors and their individual systems, and ability to answer and react possible show-stoppers saved a lot of time. With traditional waterfall method these critical changes would have been noticed in a later phase and depending of the severity of the issue changing the existing code might have been a lot more time consuming.

The main advantage of Agile method used in project was redesign. Rather than trying to tie all the knots before hand, the requirements changed as the operating environment changed. On many occasions planned features

had to be redesigned or recoded as practise showed better ways to display and handle the available data.

One of the key elements behind success is training of employees. Agile methods need a different mindset compared to traditional project management and development processes.

REFERENCES

Printed sources and e-books:

Cohn, M. 2009. Succeeding with Agile: Software Development Using Scrum

Martin, R: 2008. Clean Code: A Handbook of Agile Software Craftsmanship

DIGITAL SOURCES

Amazon 2018a [accessed in March 2018]. Availabe in:

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Aurora.Overview.html

Digitalization - Gartner IT Glossary [referred 1.5.2017]. Gartner Inc. Available: https://research.gartner.com/definition-whatis-digitalization?resId=3237920

What is a Software Framework? And why should you like 'em? [accessed in March 2017]. Available in:https://info.cimetrix.com/blog/bid/22339/What-is-a-Software-Framework-And-why-should-you-like-em

Apache Project [accessed in March 2017]. Available in: https://www.apache.org/

CentOS Project [accessed in March 2017]. Available in: https://www.centos.org/

Laravel 2017a. Laravel Documentatioin [accessed in March 2017]. Available in: https://laravel.com/docs/

Laravel Book. [accessed in January 2017]. Available in: http://laravelbook.com

Laravel 2017c. Laravel 5 – Easy Learning [accessed in March 2017]. Available in:
http://learnlaravel5easy.blogspot.fi/

MySQL.com [accessed in February 2017]. Available in:
https://www.mysql.com/

PHP 2017a. PHP.net - Manual [accessed in February 2017]. Available in:
http://php.net/manual/en/index.php

PHP 2017b. PHP MySQL Learners [accessed in March 2017]. Available in:
https://phpmysqllearners.wordpress.com/

Spring.io – Understanding REST [accessed in March 2017]. Available in:
https://spring.io/understanding/REST

Towards a Framework for Improving Software Development Process Mediated with CMMI Goals and Agile Practices [accessed in January 2017]. Available in:
http://www.vtt.fi/inf/pdf/publications/2008/P695.pdf

Webopedia 2017a. Webopedia, Virtualization. [accessed in January 2017]. Available in:
http://www.webopedia.com/TERM/V/virtualization.html

What Is 2017a. What Is Relational Database [accessed in March 2017] Available in:
http://searchsqlserver.techtarget.com/definition/relational-database

What Is 2017b. What is XML [accessed in March 2017]. Available in:
http://searchmicroservices.techtarget.com/definition/XML-Extensible-Markup-Language

What is 2017c. What is MySQL? [Accessed in March 2017]. Available in:
http://searchoracle.techtarget.com/definition/MySQL

What is 2017d. What is XML? [Accessed in September 2017]. Available in: http://searchmicroservices.techtarget.com/definition/XML-Extensible-Markup-Language

W3techs 2017a. Apache Market Position Diagram [Accessed in September 2017]. Available in: https://w3techs.com/technologies/details/ws-apache/all/all

Ebooklibrary 2017a. Waterfall model [accessed in July 2017]. Available in: http://www.ebooklibrary.org/articles/eng/waterfall_model

Lansa 2017a. Where do Frameworks fit in Application Development? [accessed in July 2017]. Available in: http://www.lansa.com/resources/where-do-frameworks-fit-in-application-development.htm

Reinvently 2017a. What is an MVP (Minimum Viable Product?) [accessed in July 2017]. Available in: https://reinvently.com/blog/understanding-minimum-viable-product-mvp/

Virtual Realities 2018a. Virtual Realities - What Virtualization Rates Are Not Telling Us! [accessed in March 2018]. Available in: https://www.linkedin.com/pulse/virtual-realities-what-virtualization-rates-telling-jim-french/

Sitepoint 2015a. The Best PHP Framework for 2015. [accessed in September 2016]. Available in: https://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/