

Development of a Progressive Web Application for stray pets reporting

Rita Da Silva Moreira



Author(s) Rita Da Silva Moreira	
Degree program Business Information Technology	
Report/thesis title Development of a Progressive Web Application for stray pets reporting	Number of pages and appendix pages 57 + 7
<p>In the last decade, the pet industry experienced continued growth due to the increasing number of domestic animals. This phenomenon also results in more animals being lost. To overcome this, many countries offer several websites that register lost animals. However, these sites are not user-friendly and they generally do not use the latest technologies.</p> <p>This project aimed to develop the prototype of a Progressive Web Application for reporting stray pets. This application attempts to provide a visual and local dimension by using geolocation and a notification system.</p> <p>Started in February 2020, the project lasted three months. The prototype was implemented following the lean startup approach, and therefore, two sprints were planned. At the end of each iteration, a week was devoted to user testing, where surveys were sent to target users to determine their needs and the usability of the application. Before this, theoretical research on PWA, React.js and Firebase was conducted. Also, an initial vision of the platform was explained using use cases.</p> <p>The surveys revealed that amongst professionals, veterinarians were the most interested in this platform. Indeed, many shelters have their own tool or use an already existing platform that veterinarians and individuals are not aware of. Moreover, social integration is a high priority for target users. In particular, they want social media integration and a notification system to reach a wide audience.</p> <p>Regarding the lean startup approach, this project has shown that it is an effective way to develop a viable product in a minimum of time. However, it has exposed that this methodology can be followed up to a certain limit. This limit depends on the time planned per sprint, the team size and the product itself.</p> <p>At the end of the project, the final product provides a basis for further development. A strategy on how to reach a future audience was also briefly discussed.</p>	
Keywords PWA, React.js , Firebase, Lean Startup, Geolocation	

Table of content

Table of figures	1
Terms and abbreviations	2
1 Introduction	3
1.1 Topic.....	3
1.2 Objectives	4
1.3 Delimitations	4
2 Theoretical background	5
2.1 Web application	5
2.2 Mobile application	5
2.3 Progressive Web Application	7
2.4 How Progressive Web App works	9
2.4.1 Web App Manifest.....	9
2.4.2 Service Workers.....	11
2.5 Why PWA over native application	14
2.6 React.js.....	15
2.6.1 Virtual DOM	15
2.6.2 Components, states and props.....	16
2.6.3 Popularity of React.....	17
2.6.4 Why React	17
2.7 Firebase.....	18
2.7.1 Firestore.....	19
2.7.2 Cloud storage.....	20
2.7.3 Authentication	21
2.7.4 Why Firebase.....	21
2.8 Lean Startup methodology	22
2.8.1 Software Development Methodology.....	22
2.8.2 Agile methodologies.....	22
2.8.3 Lean Startup	23
2.8.4 Why Lean startup.....	26
3 System overall introduction and initial vision	27
3.1.1 Use Case Descriptions.....	28
4 Project implementation.....	32
4.1 Development plan	32
4.2 Setup of the project.....	32
4.3 1 st iteration	33
4.3.1 Show reports with a map.....	35
4.3.2 Add a lost report.....	37

4.3.3	User testing and analysis	41
4.4	2nd iteration	45
4.4.1	Addition of a list of reports with filter options.....	45
4.4.2	Management of own reports.....	46
4.4.3	Other changes	48
4.4.4	User testing and analysis	49
5	Discussion.....	52
5.1	Results of the lean startup approach.....	52
5.2	Usability of the prototype.....	53
5.3	Author's own learning.....	54
5.4	Conclusion	55
6	References.....	56
Appendices		60
Appendix 1. MVP at the end of the first iteration.....		60
Appendix 2. Questionnaire and results of the first iteration.....		62
Appendix 3. MVP at the end of the second iteration		63
Appendix 4. Questionnaire and results of the second iteration		65
Appendix 5. Source code of the final implementation		66

Table of figures

<i>Figure 1: an example of a Web app manifest for a racing game (W3C, 2019)</i>	10
<i>Figure 2: Tag for importing the web app manifest in all HTML files</i>	10
<i>Figure 3: examples of a browser displaying an install banner (Osmani, 2016)</i>	11
<i>Figure 4: Architecture of a service worker (Sheppard, 2017)</i>	12
<i>Figure 5: code checking if the service worker API is available</i>	12
<i>Figure 6: Code caching some assets after an 'install' event</i>	13
<i>Figure 7: diagram of the service worker lifecycle</i>	14
<i>Figure 8: The official logo of React</i>	15
<i>Figure 9: DOM manipulation with React virtual DOM</i>	16
<i>Figure 10: Two ways of defining a React component</i>	16
<i>Figure 11: The official logo of Firebase</i>	18
<i>Figure 12: Creation of a document in a "cities" collection (Firebase, 2020)</i>	19
<i>Figure 13: Getting data from "LIS" document within "cities" collection (Firebase, 2020)</i>	19
<i>Figure 14: Query using Firestore (Firebase, 2020)</i>	20
<i>Figure 15: Cloud Storage console listing files in a simulated folder named "images"</i>	21
<i>Figure 16: The phases of agile methods through multiple sprints</i>	23
<i>Figure 17: The lean startup loop</i>	24
<i>Figure 18 : Initial use cases of the stray animal platform</i>	28
<i>Figure 19: Example of the Create React App command line to setup a React app</i>	32
<i>Figure 20: Firebase configuration of the project in the app</i>	33
<i>Figure 21: Simple data structure of a report for the first MVP</i>	34
<i>Figure 22: Implementation of a Leaflet map showing the reports</i>	36
<i>Figure 23: Code to get all reports from Firestore</i>	37
<i>Figure 24: stepper implementation</i>	37
<i>Figure 25: Example of a Yup Validation Schema</i>	38
<i>Figure 26: validation function</i>	38
<i>Figure 27: Autocomplete demo of lists based on research</i>	39
<i>Figure 28: Method that uploads an image into Firebase Cloud Storage</i>	40
<i>Figure 29: Reverse geocoding method: from latitude and longitude, an address is fetched</i>	41
<i>Figure 30: definition of the columns for the list of reports</i>	45
<i>Figure 31: creation of a user using email and password in Firebase</i>	47
<i>Figure 32: redirection to the edit page with a report passed as prop</i>	48
<i>Figure 33: Update of a report stored in Firestore</i>	48
<i>Figure 34: install banner and launching screen of the app</i>	49
<i>Figure 35: The main map showing all the reports</i>	60
<i>Figure 36: Display of the details of a report</i>	60
<i>Figure 37: The review of the lost pet report before publishing</i>	61
<i>Figure 38: The form with stepper and validation for lost pet reporting</i>	61
<i>Figure 39: The main map with search filters on the right</i>	63
<i>Figure 40: List of reports under the main map</i>	63
<i>Figure 41: The form for the account creation</i>	64
<i>Figure 42: The personal page with account information and reports management</i>	64

Terms and abbreviations

UML:	Unified Modeling Language
OMG:	Object Management Group
SPCA:	Society for the Prevention of Cruelty to Animals
OS:	Operating system
SDK:	Software development kit
HTTPS:	HyperText Transfer Protocol Secure
URL:	Uniform Resource Locator
UI:	User Interface
DOM:	Document Object Model
SDM:	Software Development Methodology
MVP:	Minimum viable product
XP:	Extreme programming
RAD:	Rapid application development
CSS:	Cascading Style Sheet

1 Introduction

1.1 Topic

The pet industry is the market related to domestic animals. This past decade, this industry experienced continual growth. For instance, the European Pet Food Industry declared having an annual increase of 2.5% in 2018 and 2% in 2016.

In their 2018 annual report, they estimated the number of European households owning at least one pet to 80 million. The number of pet dogs is said to be of 85 million and 103 million for cats (The European Pet Food Industry - Fediaf, 2019). This growing trend can be seen at the scale of a country as well. In Finland, for example, the number of households with at least one animal has increased from 30% to 35% between 2012 and 2016 (Official Statistics of Finland, 2016). In Switzerland, this rate rose 57% in 2018 against 44% in 2016 (Société pour l'alimentation des animaux familiers, 2018).

Even though these numbers are just estimations as there is no strict identification and registration system in many European countries, they nevertheless show the expansion of the animal industry. And along with these increasing numbers, the number of pets lost is also rising.

As a result, the identification and registration of pets became mandatory in some countries such as Belgium. However, these regulations differ from a country to another. For instance, in Switzerland, dogs must be microchipped and registered in a national database while cats do not (Office fédéral de la sécurité alimentaire et des affaires vétérinaires, 2019). Thanks to the traceability it gives, this regulation helps authorities in many ways. Indeed, it facilitates investigations into illegal pet trade, the antecedents of an animal can be easily checked for health or welfare situations, and it has been proven that microchipping increases the return rates of lost pets (Ohio State University, 2009).

Although these systems were introduced to limit the number of animals lost, pets continue to get lost. Indeed, the ASPCA declared that in the United States in 2018, 6.5 million pets entered a shelter, and among them, 710'000 stray animals found their owners.

In almost all countries, platforms exist to register lost and found pets such as "www.karkurit.fi" in Finland or "www.stmz.ch" in Switzerland. They, however, have not yet taken advantage of technology to increase their efficiency. For instance, none of them use geolocation or notifications. They instead only list the reports and have a simple form for searching purposes.

1.2 Objectives

It is why, with this project, we aim to create a prototype of a platform for stray pet reporting so that all reports are centralized in only one place.

This platform is intended to be innovative by adding a new dimension, which is the use of geolocation. In order to be accessible for all devices and regardless of network connection, a Progressive Web Application will be created. The platform is meant to be a free service that can be used by both individuals and professional institutions.

Before the implementation, we will determine the users' needs with initial use cases. Furthermore, a discussion will be conducted to explain why the chosen tools and technologies are the most suitable for this project. Hence, we will present Progressive Web Application, Firebase and React.js.

To develop this application while delivering the maximum user value, it will be implemented following the lean start-up methodology. A chapter will be dedicated to that approach.

The implementation will consist of two iterations, including user testing and analysis of the results. After that, a discussion will be made around the results of this project, its implementation using the lean start-up approach, and the usability of the product.

1.3 Delimitations

As the time available for this project is short and the application is significant, we will only develop a prototype. Thus, the mobile responsiveness will not be included. However, to learn and practice PWA, at least one mobile-friendly page will be created.

Also, some use cases will not be developed during the project, such as:

- Adding report for a found pet
- Search with a form
- Share to social media
- Contact through a report

2 Theoretical background

In this chapter, we will discuss the technologies used for this project and why they are the most suitable compared to other solutions. Progressive web applications will be presented. Before that, we will first expose the situation of mobile and web development nowadays to explain the emergence of PWA. Then we will present Firebase that will be used as backend and React as front-end. And finally, we will discuss about software methodologies and why the Lean start-up methodology was chosen for this project.

2.1 Web application

A web application is a server-side software program that is delivered over the Internet through a browser. Web services are considered web apps and many websites contain them. (Rouse, 2019) For the end-users, there is no difference between websites and web apps because in both cases they will use a web browser and type the URL to access them. Until now, the differences between the two are still being discussed by developers. As a rule, the primary purpose of a website is to convey information, whereas web apps are usually intended for interactivity with end-users. They do still transmit information, but users are also able to manipulate that information or, through interaction, request a different type of information. As examples of web applications, we find webmail, online banking, e-commerce shops or even words processors such as Google Sheets (Gibb, 2016).

Unlike mobile applications, a web app runs on multiple platforms regardless of the devices or the OS. Therefore, it is easier to develop and maintain the app's code as there is only one code.

In addition to this, a web app needs no installation compared to a native application, and thanks to that, developers do not have to go through app stores' approval process to release their apps. However, because of that lack of regulation, web apps are prime targets for hackers. Indeed, the 2019 Veracode State of Software Security reported that 83% of applications had at least one vulnerability. (Gates, 2018)

2.2 Mobile application

Over this past decade, mobile applications have become a new way for businesses to reach and serve their customers. Indeed, they can help companies to improve customer loyalty by having the ability to interact directly with them (with push notifications, for example).

Compared to web apps, they are software that run on the device itself. They need to be installed and thus require storage capacity. As they are downloaded on an app store, they must pass its approval process, which ensures security and compatibility. The main advantage of mobile applications compared to web apps is that they have access to the device's features such as the camera, payment options or GPS (Stevens, 2018).

Native mobile applications, which are currently the most popular type of mobile application, are built for one platform (IOS, Android, Windows Phone). They usually are more expensive to develop because each platform uses a specific programming language. In addition to that, the process of validating an app through an app store can be long and laborious for developers. Nevertheless, native mobile developers have access to a software development kit (SDK) that can facilitate the creation of the app (Existek — Software Development Company, 2019).

Another type of mobile application is a hybrid application. Like native apps, it runs on the device, but it is written with web technologies like web apps (HTML5, CSS and Javascript). In other words, it is a web app wrapped in a native container and it uses the browser of the device to render its content. The principal advantages of hybrid apps are that they are multi-platform compatible and have, in opposition to web apps, access to the device's features (Cowart, 2012).

Since the launch of the first iPhone in 2007, mobile applications took over websites. Indeed, the possibilities that offered native applications were a real game-changer for developers and end-users. Next to the number of mobile features available to them (notification push, offline availability, home screen icons, geolocation, camera), web sites and web applications seemed boring for developers. Therefore, the number of native mobile applications grew since then and the mobile app industry was thriving. However, this era is slowly changing.

Even though we spend more time than ever before on our smartphones and apps, we tend to limit our use to a diminishing number of them. Indeed, according to the 2019 comScore report on 'Global State of Mobile', most users spend 77% of their time on their top 3 apps (96% on top 10). Also, people in the United States keep downloading fewer new applications (33% downloaded new apps in June 2019 compared to 49% in June 2017). That trend makes it harder for any new mobile application to break into the market. It is even more complicated to stand out among all existing applications when it is known that 75% of apps are used only once after being downloaded. (BuildFire, 2019)

It is why in response to that competitive market and the shortcomings of web apps, progressive web applications emerged since 2015.

2.3 Progressive Web Application

Progressive web applications are falling in between web applications and native applications. Indeed, as we will see, they are enhanced websites that feel like native apps. And it is not just only about responsive design, but also about the features from native mobile applications.

The term “progressive web app” appeared for the first time in 2015 by Frances Berriman and Alex Russel, a Google engineer, over dinner (Russell, 2015). There is nothing new with progressive web application. It is mainly a set of strategies, techniques and APIs that give that native app experience for the end-user. These practices have been developed and promoted actively by Google (Google Developers, 2020). According to them, PWAs should follow these three pillars:

- **Capable:**
Today, the web is more capable than ever. Modern browsers now have features that did not exist before, and it will keep enhancing over time. For instance, some of the features that before were only claimed by native apps are now within the web’s reach.
- **Reliable:**
A PWA needs to be reliable regardless of network connection. It should display the most recent content users have interacted with and be usable even if the network is low or down. Also, PWA needs to be fast to build trust with users and encourage them to use it.
- **Installable:**
PWA can be moved out of a browser tab into a standalone app window. Doing so, PWAs are launchable from the device home screen, searchable and jumpable using the app switcher. They feel like part of the device.

In order to reach these pillars, PWA must meet the following characteristics:

- **Responsive**
PWA’s UI should fit any device: mobile, desktop, tablet or any form yet to come. For this purpose, conventional solutions for responsive websites can be used, such as grids, CSS libraries, frameworks and media queries.
- **Connectivity independent**
Service workers, key components for PWA, allow apps to work offline. By caching part of an app or its totality, they make it available offline. In most cases, the UI is cached,

and the dynamic data requires a connection to the server. Service workers will be explained in more detail in the next section.

- **App-like-interactions**

A PWA should follow an Application Shell Architecture for better performance. This architecture allows similar properties of native apps, such as instant loading or regular updates. It is done by separating the content of the app from the static UI elements (header, menus, drawers...). That way, when the connection is lost, the user still has access to the app interface instead of an empty page. (Google Developers, 2019)

- **Fresh:**

A PWA is transparently always up to date, thanks to the service worker. By updating the service worker, developers can release any changes to the app. Otherwise, a PWA will always load from the local storage that has been previously cached.

- **Safe:**

The use of a service worker requires an HTTPS connection to protect the sensitive data being handled.

- **Discoverable:**

PWA is identifiable on the web as “application” thanks to a Web App Manifest file. This file allows developers to specify metadata such as the icon, name, category and description of the PWA but also some specific behaviors. The Web App Manifest will be further explained in this document.

- **Re-engageable:**

Now that push notifications are not limited to native applications anymore, PWAs can use this feature to help engage users.

- **Installable:**

Through browser-provided prompts, it is possible to save a PWA to the home screen of a device. If the app follows a series of requirements, the browser will automatically display an installation pop-up asking to add it to the home screen.

- **Linkable:**

PWAs can be accessed simply via their URL. It means that they are easy to share, and no app store is required to find or install them.

As PWAs are products of the web, their market reach is more significant than native applications because they can reach everyone regardless of the OS of the devices. Also, they

are lighter than native apps as they only take what they need. Thanks to that, the downloading process is also faster. Also, by bypassing app stores, the process of launching and updating a PWA is easier for developers. In addition to that, as PWAs only have one codebase for multiple platforms, the development is cheaper and faster than for native apps where a codebase is necessary for each OS.

Despite all these qualities, PWAs have shortcomings. Firstly, they have limited browser support. Indeed, not all browsers support the full ranges of PWA features yet. If Chrome, Firefox and Opera supports most of them, Safari and Edge are lagging. For instance, Safari does not support Web App Manifest or push notification (Love, 2019). However, since their 11.3 IOS release, they do support service workers, which already gives a great PWA experience with offline support.

PWAs also have limited access to device-specific functionalities. For example, they are not able to use NFC, Bluetooth, Contacts or Geofencing.

In addition to that, their non-presence on app stores can delegitimize them. Indeed, app stores not only provide legitimacy and proof of quality through reviews, but they also are catalogs where users search for apps instead of using search engines. Because of that, they can miss PWAs. Moreover, users may have less confidence in downloading a PWA from a website than from an app store where applications are regulated. (Siavosh, 2019)

Nevertheless, PWAs have benefited many companies since their emergence. For instance, AliExpress, which is the most visited e-commerce site in Russia, has increased by 104% the conversion from new users and the time spent on their site by 74%. Twitter Lite, a PWA version of twitter, also had positive results: a 75% increase in Tweets sent, a 65% increase in pages per session and 20% decrease in bounce rate (users entering and leaving the site). (Google Developers, 2020)

2.4 How Progressive Web App works

In the following sections are explained the two minimum components required to develop a PWA: a web app manifest and a service worker. Other components can be used for specific PWA features. For instance, the push and the notification APIs are used for push notifications. Cache Storage is another component that is needed for caching.

2.4.1 Web App Manifest

A Web App Manifest is a JSON file where we specify details about the app to help devices create the best experience for users. For the browser, it is only this file that makes it possible

to distinguish between a classic website and a PWA. In this file, we specify for instance the icons for different device resolution, the name of the app, the options to make the app appears without the browser frame, the splash screen while fetching the content from the network instead of the cache or even the preferred URL to open when users launch the app. The next figure is an example of a Web App Manifest.

```
{
  "lang": "en",
  "dir": "ltr",
  "name": "Super Racer 3000",
  "description": "The ultimate futuristic racing game from the fu-
ture!",
  "short_name": "Racer3K",
  "icons": [{
    "src": "icon/lowres.png",
    "sizes": "64x64"
  }, {
    "src": "icon/hd_hi",
    "sizes": "128x128"
  }],
  "scope": "/racer/",
  "start_url": "/racer/start.html",
  "display": "fullscreen",
  "orientation": "landscape",
  "theme_color": "aliceblue",
  "background_color": "red",
}
```

Figure 1: an example of a Web app manifest for a racing game (W3C, 2019)

Web App Manifests are an easy and straightforward procedure. Once all the options have been specified in the JSON file, that file needs to be referenced in all HTML files using the following tag inside the head tag:

```
<link rel="manifest" href="manifest.json">
```

Figure 2: Tag for importing the web app manifest in all HTML files

It is the web app manifest that enables the installation of the PWA in the device. However, the browser is the one to decide to prompt the install banner. In order to have that happening, the app must fulfill three requirements: it has to be served over HTTPS, a service worker must have been registered for the site, and the web app manifest should have the following four mandatory fields: name, start_url, icons and display.

When these criteria are fulfilled, the browser will prompt the install banner after multiple visits.

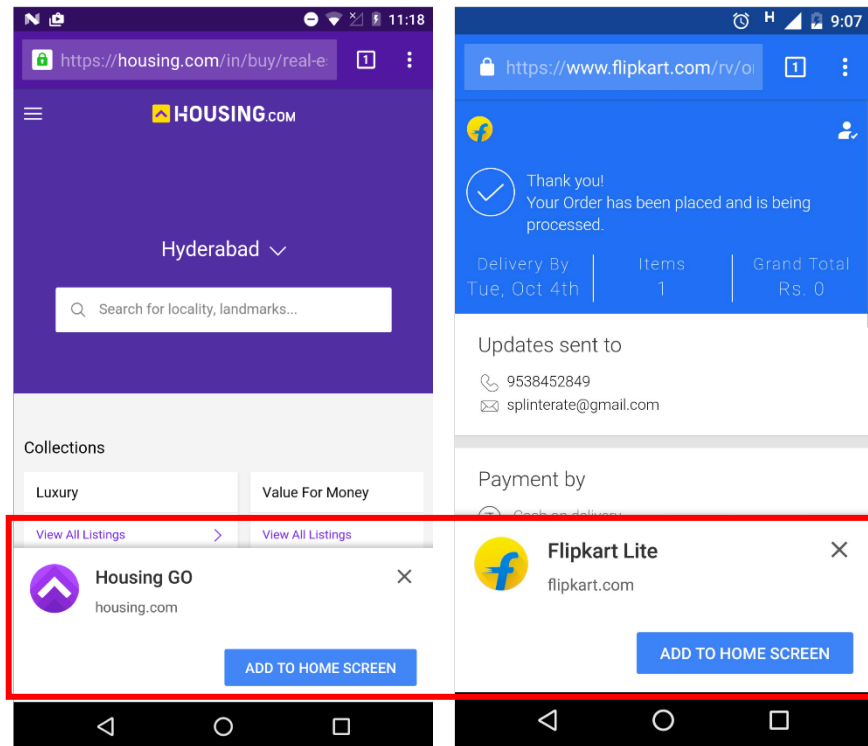


Figure 3: examples of a browser displaying an install banner (Osmani, 2016)

2.4.2 Service Workers

A Service Worker is an intermediate component that sits between the application and the network. It is written in Javascript and acts like a script that runs outside the context of the application.

It works on an event-driven basis and can actively modify the way an application behaves (caching, handling push notifications, background synchronization, availability offline). Events such as data requests from the web can be intercepted, modified, passed on, and returned to the page. Service workers can be used for man-in-the-middle attacks because they act as an intermediate between the network and the app. To prevent that, service workers need to be registered on web pages served over HTTPS.

Service workers are always listening and thus can intercept requests even when the app or tabs are closed. Also, as they work in a layer before the network, they can respond to request independently of a network connection. Indeed, they can detect when there is slow or no response from the server and return cached content instead. In addition to that, they make sure that any performed action gets delivered to the server when the connection is back.

As explained earlier, service workers are the key component managing caching for PWA.

When an app requests an image, the service worker intercepts that request and verifies first in the local cache if the image is already stored. If it is the case, it returns it from the cache and no network request is made. However, if it does not find it, it lets the request go through the network and after downloading the image, it stores it in the cache. Also, as the service worker runs in a separate thread from the UI, it does not block the UI while it operates. (Sheppard, 2017) (see Figure 4).

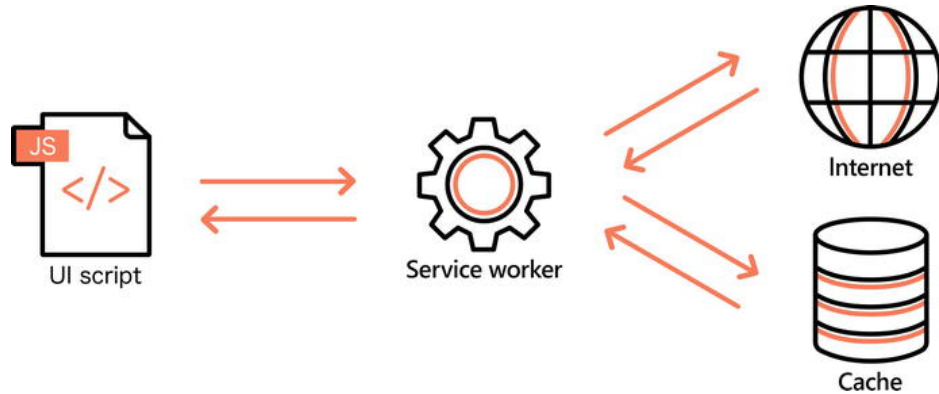


Figure 4: Architecture of a service worker (Sheppard, 2017)

A service worker has a lifecycle consisting of mainly 3 phases: registration, installation and activation. In order to install a service worker, we must beforehand register it in our pages. The following code first checks if the browser supports service workers and then it registers it by informing the browser where its file is stored (sw.js).

```
if ('serviceWorker' in navigator) {
  window.addEventListener('load', function() {
    navigator.serviceWorker.register('/sw.js').then(function(registration) {
      // Registration was successful
      console.log('ServiceWorker registration successful');
    }, function(err) {
      // registration failed :(
      console.log('ServiceWorker registration failed: ', err);
    });
  });
}
```

Figure 5: code checking if the service worker API is available

After being successfully registered, the service worker can be installed. The JS script is downloaded, and the browser tries to install it. It can be successfully installed only if it has never been registered before or if the script has changed. Once it has been installed, an “install” event is sent and, by listening to it, it is possible to perform application-specific tasks such as caching static assets:

```
const assetsToCache = [
  '/index.html',
  '/about.html',
  '/css/app.css',
  '/js/app.js',
]

self.addEventListener('install', function (event) {
  self.skipWaiting();
  event.waitUntil(
    caches.open('staticAssetsCache').then(function (cache) {
      return cache.addAll(assetsToCache);
    })
  );
});
```

Figure 6: Code caching some assets after an ‘install’ event

After the installation, the service worker is waiting to be activated. It can be activated in three situations:

- if there is no service worker currently active
- if in the install event handler, it is declared ‘self.skipWaiting()’ to force the waiting service worker to become the active one
- if the page is refreshed.

Once activated, the service worker has full control of the pages and can thus handle events. If it does not receive any events, it goes into an idle state. After a while, the service worker changes into a “terminate” state until a new event is handled, becoming again idle. (Enyinnaya, 2019)

The next figure is a diagram of the service worker lifecycle to illustrate the last paragraphs.

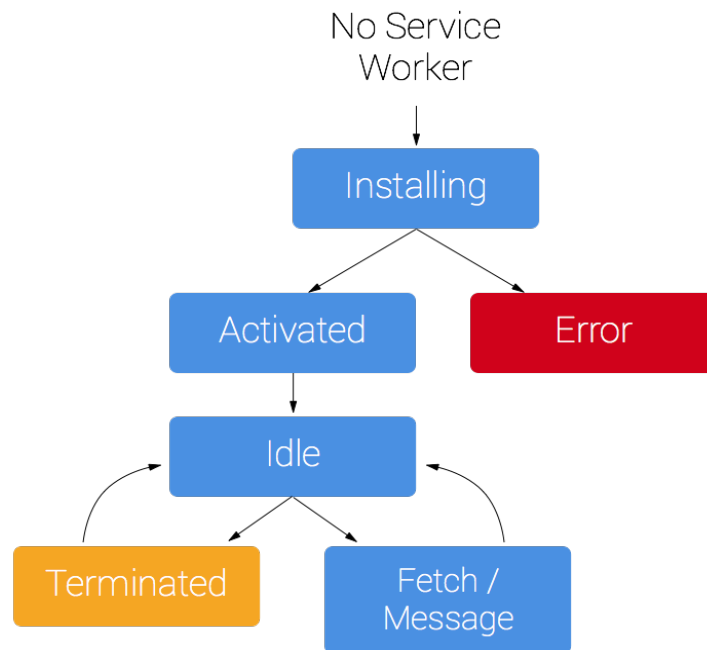


Figure 7: diagram of the service worker lifecycle

2.5 Why PWA over native application

As this project is intended to be a platform accessible in all circumstances, the choice of a PWA seemed to be the most logical one.

Indeed, when reporting a stray animal, it happens, for example, that the person is in the middle of the forest where the network is often poor. In this situation, a website would not be able to support this low network and the user would grow impatient.

Furthermore, even if it is possible to make a mobile app work offline, it would not have been an optimal choice as it restrains the use of the platform to mobiles only. And in order to reach the highest number of users, it is necessary to offer a solution that combines mobiles but also computers and tablets.

In addition to that, as this platform is intended to be used episodically (mostly when people lose or find animals), end-users probably would not like the idea of installing a native application that would heavily use their local storage. In contrast, PWAs are better since they only store what is needed for caching.

On top of that, they now have native-specific features such as push notifications, camera access and GPS. These three functionalities are decisive for this platform. For instance, push notifications are necessary to help the sharing of new reports and informing the user of any received messages.

Finally, a single code for multiple platforms is a real time-saver in order to set up this platform as quickly as possible.

2.6 React.js

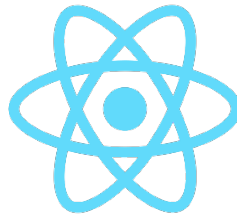


Figure 8: The official logo of React

React is a “Javascript library used for building user interfaces” (Facebook Inc., 2020). It was developed by Facebook Inc. in 2013. This library manages only the front-end of an application, considered as the view layer in the MVC model. It is component-based, which means that developers can break down the UI into independent, reusable elements, allowing them to consider each element separately. The library stands out from its competitors by its flexibility and performance, working with a virtual DOM and updating the rendering in the browser only when necessary.

2.6.1 Virtual DOM

The React virtual DOM has been created to solve the common problem of slow DOM manipulation. Indeed, DOM manipulation is an essential part of web development. The DOM (Document Object Model) represents the UI of an app with a tree data structure. As we want the user interface to change dynamically with the content, the DOM must be updated. These modifications in the browser DOM are known to be very slow. In addition to that, most Javascript frameworks update the DOM much more than what they need to (Chinnathambi, 2016).

With React, developers never change the DOM directly. They instead modify the virtual DOM, which is an in-memory copy of the real DOM. When a virtual DOM object is updated, React compares the virtual DOM with a virtual DOM “snapshot” that was taken right before the update. This process is called “diffing”. By doing so, React figures out exactly which DOM objects have changed and hence only update these objects in the real DOM. That final step is called “patching” (see the next figure to illustrate the overall process). For instance, without React, a list of items would have been entirely rebuilt if an item had been changed. With React, only the specific item is updated in the DOM. (Codecademy, 2020)

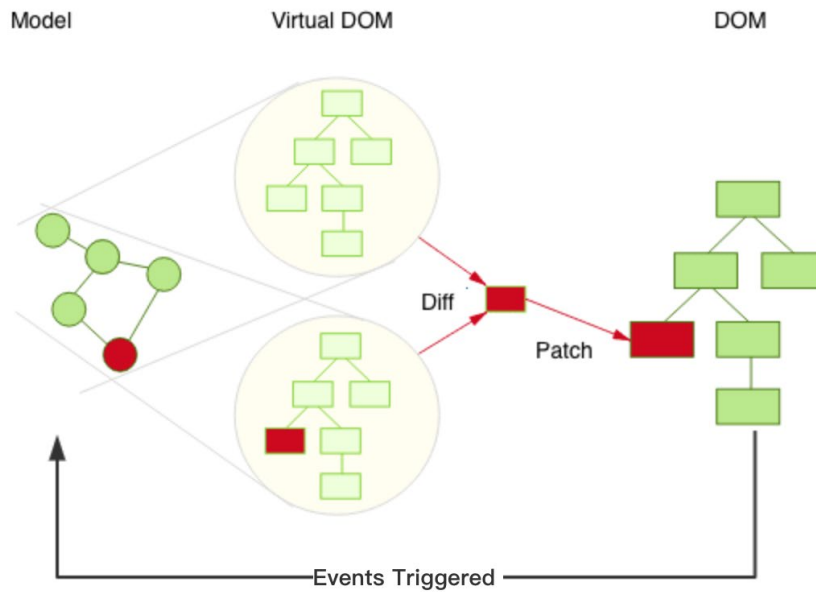


Figure 9: DOM manipulation with React virtual DOM

The manipulation of a virtual DOM is faster firstly because it does not get drawn at each update compared to the real DOM, but also because when the real DOM is redrawn, it is strictly the changed objects that are updated.

2.6.2 Components, states and props

In React, components are reusable and independent code blocks that divide the UI into smaller elements. React has two types of components: functional components or class components. Functional components are defined by a Javascript function, while class components are defined by an ES6 class (see the next figure to understand the differences between the two). (Facebook Inc., 2020)

```
// Functional component (with Javascript function)
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

//Class component (with ES6 class)
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

Figure 10: Two ways of defining a React component

Developers can create their own components. These custom components must have a name starting with a capital letter and they can pass attributes that become props.

These props are used to pass data between components. They are read-only, which means they cannot change after the first render of the component. And because React has a uni-directional data flow, props can only be passed from parent to child.

In contrast to static props, state is the dynamic part of React components. It allows them to create and manage their data internally. To update a state, a method called “setState(..)” must be used. This method re-renders the components and all the child components. In the early days of React, only class components could use states. However, since the introduction of React Hooks, functional components can now also use them. (Ravichandran, 2018)

2.6.3 Popularity of React

React is part of the most popular frameworks and libraries nowadays, amongst Angular and Vue.JS. Indeed, according to a survey made by (StackOverFlow, 2019), React.js is the most loved web frameworks by developers. And there are many reasons why.

Firstly, component-based frameworks help to have a clean code that is easy to read and reusable. Secondly, as this library is an open-source platform since 2013, there is a large and active community that contributed to a rich package ecosystem and documentations. React.js is also recognized for its fast performance with its virtual DOM. Moreover, the fact that many large-scale companies are using React for their web apps is reassuring for developers trying to create their app. Among these big companies, we find Netflix, Airbnb, Dropbox, Reddit and of course Facebook. Finally, having a company like Facebook backing React and maintaining it for their own sake contribute to the evolution of it. (Code, 2019)

2.6.4 Why React

To be effective for this project, the use of a web framework for the front-end is necessary. Nowadays, there are numerous frameworks, but the most famous ones for PWA development are Angular and React. (Rajput, 2019)

Even though Angular is a mature and complete framework with a significant number of contributors and backed by Google, the learning curve is too steep to dive into this framework in the time available. Indeed, to get to understand and use Angular, it is necessary to know TypeScript, an open-source language. And as this project is focused on creating a PWA prototype with geolocation, the learning content is big enough to not add two new concepts to it (TypeScript and Angular).

Also, as we already have knowledge in React, this project can hone our skills and make us feel more at ease with this technology. The fact that there is comprehensive documentation and great support from the community and Facebook is encouraging. In addition to that, as this project will have a map and geolocation functionalities, it is decisive to choose a flexible framework that can download packages easily. For all these reasons, React has been selected as the front-end library for this project.

2.7 Firebase



Figure 11: The official logo of Firebase

Firebase is a platform for software development created by Firebase Inc. but acquired by Google in 2014. It is a Backend-as-a-Service solution that covers many developer's needs. It includes numerous services, such as Cloud Storage, Authentication, Realtime Database, Hosting, Cloud functions, and Crashing reports. Firebase defines itself as a Rapid Application Development (RAD) platform because its purpose is to allow developers to build apps fast without worrying about the back-end infrastructure but focusing instead on the need of the end-user.

Firebase divides its services into three goals: building better apps, improving the quality and growing the business. It integrates Google Cloud Platform, allowing developers to use its services such as Google Ads, Google Marketing Platform or Google Analytics. (Firebase, 2020)

Some of the services that will be used for this project are presented in more detail in the following sections.

2.7.1 Firestore

Firestore is a NoSQL cloud database for mobile, web and server development. It is the successor of Firebase Realtime Database. Because it is a NoSQL database, there are no tables or rows like for SQL databases. Data are instead stored in documents that contain sets of key-value pairs. These documents are then organized in collections. It is also possible to create subcollections within documents. In addition to that, Firestore supports many data types such as simple strings and numbers to more complex objects like nested arrays. For custom objects, Firestore uses a Converter called “FirestoreDataConverter” to convert objects into supporter data. Thanks to all these features, Firestore data model supports any data structure. This flexibility allows developers to structure their database as it suits them the best.

```
// Add a new document with a specific id ("LIS")
db.collection("cities").doc("LIS").set
  name: "Lisbon",
  country: "Portugal",
  capital: true,
})
.then(function() {
  console.log("Document successfully written!");
})
.catch(function(error) {
  console.error("Error writing document: ", error);
});
```

Figure 12: Creation of a document in a “cities” collection (Firebase, 2020)

The figure above shows how to create a document in a collection named “cities”. With the set method, we generate a meaningful Id (here “LIS”). However, if the id is not essential, Firestore can auto-generate Ids by using add() method instead of set(). Also, retrieving data with Firestore is said to be expressive and efficient. Indeed, apps can retrieve data at the document level without retrieving the whole collection (see the following figure).

```
var docRef = db.collection("cities").doc("LIS");

docRef.get().then(function(doc) {
  if (doc.exists) {
    console.log("Document data:", doc.data());
  } else {
    // doc.data() will be undefined in this case
    console.log("No such document!");
  }
}).catch(function(error) {
  console.log("Error getting document:", error);
});
```

Figure 13: Getting data from “LIS” document within “cities” collection (Firebase, 2020)

It is also possible to use queries to retrieve specific data using a where method. This method takes three parameters: a field to filter, an operator (<, >, ==, in,...) and a value.

```
var query = citiesRef.where("capital", "==", true);
```

Figure 14: Query using Firestore (Firebase, 2020)

2.7.2 Cloud storage

As for any application, it is necessary to have a place to store assets or user-generated content such as photos, videos or audio files. Firebase offers Cloud Storage for this purpose. It inherits features from Google Cloud Storage, but it also has its own features.

Compared to Google Cloud Storage, Firebase storage is easier to use, and it is more robust in terms of data transfer. Indeed, it can perform downloads or uploads regardless of network quality. If the connection is weak, these operations can stop and resume from where they were left, which preserve users time and internet bandwidth.

It is also highly scalable because Firebase is backed by Google infrastructure. If needed, Google storage environment can quickly adapt to any growth demand.

In addition to that, Cloud Storage is secure. Again, as it is a Google product, it has its power in terms of security. Moreover, Cloud Storage can integrate Firebase Authentication and it is also possible to declare security rules to restrict or grant access to some files.

Cloud Storage can be compared to a filesystem, but it is different in one particular aspect: there are no directories or folders. It works instead with buckets that are like objects containers. All objects inside a bucket have a unique name. By using slashes (/), we can make objects appears like they are part of a directory structure. For instance, we can call an image "users/uid/images/profile1.jpg". This way of naming files is useful for browsing in the console as Firebase shows these objects as part of a hierarchical structure.

Nevertheless, for Cloud Storage, they are only individual objects. The following figure shows the Cloud Storage console on Firebase with a simulated directory "images". (Mayur Tanna, 2018)

Nom	Taille	Type	Dernière modification
dog2.png	20.79 KB	image/png	9 déc. 2019
eHC3eyoXbXhMBCXagG8V7yzumU63-1574977380885.jpg	1.64 MB	image/jpeg	28 nov. 2019
eHC3eyoXbXhMBCXagG8V7yzumU63-1574977406897.jpg	1.64 MB	image/jpeg	28 nov. 2019
eHC3eyoXbXhMBCXagG8V7yzumU63-1574977424560.jpg	1.64 MB	image/jpeg	28 nov. 2019
eHC3eyoXbXhMBCXagG8V7yzumU63-1574977443594.jpg	312.37 KB	image/jpeg	10 déc. 2019

Figure 15: Cloud Storage console listing files in a simulated folder named "images"

2.7.3 Authentication

Firebase Authentication is a solution to manage user authentication for applications. It supports multiple methods of authentication, such as the traditional form-based authentication with email and password, phone numbers or third-party providers, including Facebook, Twitter, Google or Microsoft.

2.7.4 Why Firebase

Firebase has been chosen for this project firstly because it is a complete back-end solution that helps to cut down the development time, which is imperative to create a working prototype without delay. Indeed, by centralizing most of the services needed for this application, such as Cloud Storage, Firestore and Authentication, Firebase simplifies the configuration and management of these resources.

Also, Firebase offers multiple pricing plans, including the Spark Plan, which allows a free start for any application. This free plan offers enough resources for the project: 20.000 documents writes/reads per day for Firestore and 5 Gigabytes of storage. Moreover, it is possible to upgrade the current plan as the app grows.

In addition to that, as Firebase is provided by Google, its services meet Google requirements in terms of speed and reliability. It helps to have a fast working application. Also, Google has set up a rich documentation, which also saves time.

Finally, from a personal point of view, Firebase Realtime Database and Cloud Storage has already been used for school projects. Despite that basic knowledge, not only is there room for improvement, but there is also a personal interest in discovering particularly Firestore and Authentication with third-party providers.

2.8 Lean Startup methodology

The purpose of this section is to explain what a software development methodology is and how useful it is. The differences between heavyweight and lightweight methodologies will be presented. As the lean start-up is an agile methodology, a small explication about agile will be given. We will end this chapter by presenting the Lean Startup approach and explain why we chose it.

2.8.1 Software Development Methodology

Software have been part of society for many years. At first, they were developed messily, without much planning and decisions were made based on immediate needs. Even though it was working fine at first, as soon as projects started to become more complex, developers faced more difficulties in fixing bugs and adding new features. That is why methodologies appeared.

A Software Development Methodology (SDM) is a framework used to plan, structure and control the process of building software. Through their organized processes, these methodologies make software development more predictable and efficient. (Awad, 2005)

The first methodologies to be developed were called heavyweight. This term comes from the fact that these methodologies focused on detailed documentation of requirements, planning, architectural and design development. These models had many flaws. For instance, they did not handle dynamic changes easily, they were said to be too predictable, bureaucratic and laborious and they resulted in many unsuccessful projects. As a result, a new kind of methodology appeared, called lightweight. These methodologies are based on iterative enhancements and are less document-oriented. They adapt to change well; they promote knowledge sharing and project teams are usually smaller. These lightweight methods are now considered as Agile methodologies (Charvat, 2002)

2.8.2 Agile methodologies

As stated above, agile methodologies are an alternative to heavyweight methods. They focus more on interaction, working software, customer collaboration and change, rather than on plans and processes. They help teams to respond to the unpredictability of software development through incremental and iterative works called sprints.

These sprints usually last between one to four weeks, and in the end, the goal is to demonstrate a working product to stakeholders. Within each iteration, there are multiple phases such as planning, analysis, design, coding, testing and review (the next figure illustrates this process). These iterations allow the product to adapt to change quickly. More importantly, they allow stakeholders to redefine the product according to new business needs. For this kind of methodology, face-to-face communication is crucial to minimize lost time by asking questions by email, phone or wiki.

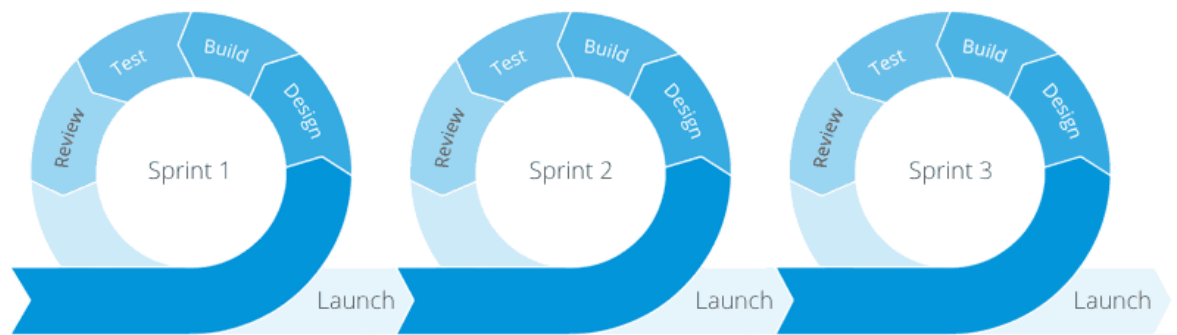


Figure 16: The phases of agile methods through multiple sprints

Many agile methodologies exist, such as Scrum, Extreme programming (XP), Rapid application development (RAD) and Lean startup. Even if these methods have their own approach, they all incorporate iteration and continuous feedback. (Miski, 2014)

2.8.3 Lean Startup

It was in his bestseller "The Lean Startup" in 2008 that Eric Ries first evoked the concept of Lean Startup. His analysis of startups leads him to notice that one of the difficulties they encounter is that they create the company from an existing technology rather than designing it based on the needs of the users. By following this approach, startups, most of the time, bias the approach by projecting their own interpretation of the needs, which may be far from the perception of the users. Therefore, many companies design products that are little or not used. The Lean Startup method allows startups to instead focus on the elements that matter for the growth of their business. (Dufour, 2020)

The most important thing with this methodology is to develop a product that the market wants. It is done by frequently evaluating the product with as many target users as possible to know, as quickly and as early as possible, their needs and reactions to the product. This method makes it possible to design, with a minimum of investment, innovative solutions adapted to users' expectations.

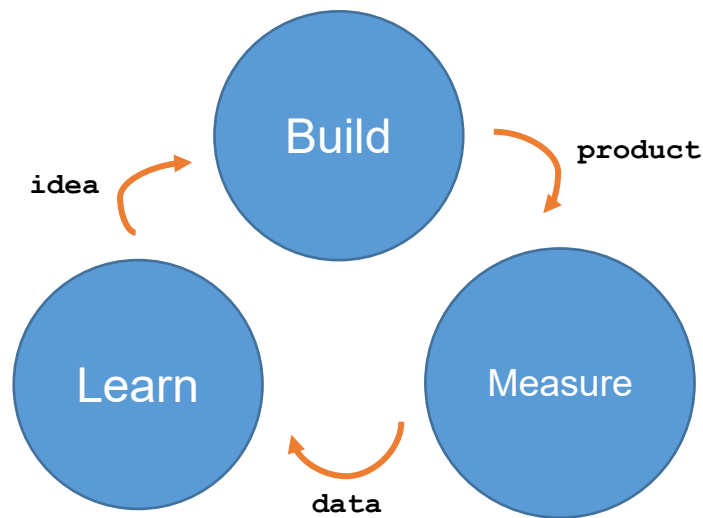


Figure 17: The lean startup loop

As it is considered an agile method, the lean startup follows the principle of iteration. The figure above depicts the Lean Startup loop consisting of three phases: build, measure and learn.

Build

At first, the startup builds a minimum viable product (MVP). It is sort of a prototype of the product. It is called “minimum” because its implementation requires the minimum possible effort and time. It is, therefore, evident that the PMV will not be perfect and that it will lack many features before it looks like the perfect product. Nevertheless, this minimal version makes it possible to test hypotheses with customers.

Measure

The role of this phase is to measure the effectiveness of MVP in the market. It is measured through end-users evaluation. In order to measure the progress of MVP effectively, it is essential to define useful metrics.

Indeed, metrics must be chosen according to their ability to help make decisions. Also, “vanity” metrics should be avoided as they only make companies feel good but do not reflect the real progress of the product. For instance, the number of hits on a webpage is a vanity metric as it can quickly reach a large number, but it does not tell any useful information such as where do these hits come from or how was the experience for users.

Eric Ries suggests focusing on actionable metrics. They are metrics that are often linked to specific and repeatable actions or features. They produce results that are indisputable in contrast to vanity metrics. A/B testing is a way of measuring these metrics in software development.

Learn

Then comes the time for conclusions and decisions: should the startup persevere or deviate from its initial strategy? A startup must be able to question itself and, if necessary, adapt, change, or reinvent its strategy. According to Eric Ries, nothing is more dangerous than a company that is stagnating. Furthermore, a successful change of direction can put a company back on the path to long-term business. These changes of direction are called pivots.

There are different types of pivots:

- restrictive pivot (a feature becomes the product in its own)
- extensive pivot (the initial product becomes a feature of a larger product)
- customer segment change (when customers differ from the one originally intended)
- customers' needs change (when customers have other needs more important than the one for which the product exists).

It is during the learning phase that startups can make these potential changes. A pivot is emotionally hard to accept because people generally see it as a failure. Nevertheless, failure is necessary for innovation. It allows companies to realize that the path to success is different from what they had imagined. Additionally, failure contains useful information to avoid repeating the same mistakes. (Xhofferay, 2016)

This phase aims to define a new business hypothesis to be developed in the next iteration or to persevere with the same one.

Lean Startup methodology offers entrepreneurs the opportunity to design their product while retaining great flexibility to adapt their product during the implementation of the project. This approach has several advantages:

- help creating a business quickly
- bring entrepreneur to develop a product adapted to the users' expectations thanks to an active customer validation approach
- develop products through short iterative cycles
- limit the amount of financial investments
- validate the product's commerciality through measurable evaluations.

Despite these multiples benefits, lean techniques are far from unanimous. Indeed, they are accused of intensifying work, which could have an impact on the well-being of workers. By getting rid of wasted time, workers may not have that time of recuperation, which is sometimes vital.

Another critic about this methodology is that it mostly focuses on the short term. As the lean startup key action is to pivot when it is necessary, it is difficult to maintain a long-term vision of the business. And without a vision, finding investors often becomes more complicated. In addition to that, products developed following this methodology are often criticized too. Mostly because, when they are launched in the market, they are incomplete. This could

refrain customers from using the product that does not yet have all the essential features. These critics use the example of large companies such as Facebook or Google, whose success is due to the launch of a good quality product right from the start.

2.8.4 Why Lean startup

At first, when planning this project, the waterfall methodology, which divides projects into linear sequential phases, was favored because this linear process looked more logical for a project where there is only one developer.

However, during the starting meeting, the advisor suggested to follow the Lean Startup methodology instead. Because, as he explained, it is crucial when doing a product-oriented thesis, to be able to evaluate the value of the product and it is more efficient with real end-users' feedback. That is why this methodology suits this project better because it focuses on the users' evaluation.

Also, as explained before, this methodology strategy is to build a minimum viable product with minimum time and resources in order to test it as soon as possible. This approach promotes fast product development cycles, which is optimal for this project and the available time left.

3 System overall introduction and initial vision

This chapter aims to present the initial vision of the application using use cases. The role of a use case model is to “get to the heart of what a system must do.” (Spence & Bittner, 2002). It is a simple and powerful way of representing the behavior of a system. It is basically a list of actions and interactions between actors and the system. Actors can either be a real user or an external system that interacts with the software. A use-case model is actor-oriented in order to ensure that it solves the user’s requirements.

Besides the actors, it usually includes two other main components: use cases and relationships between the actors and the use cases.

A use case describes “the way a system behaves to meet a requirement” (Hamilton & Miles, 2006). It has a title and a brief description. It can also have detailed descriptions to specify for instance, how the actors use the system and how it respectively responds to it. Relationships between actors and use cases are referred to as behavioral relationships. There are four types of relationships: associations, include, extend and generalization (Nishadha, 2019).

The next figure is the use-case diagram for this project. There are two main actors: a visitor and a member. As this platform aims to be a public service, they both share some use cases, such as the ability to see the map displaying reports and search for something. They also can contact a member through the report or share the report through social media. A visitor has another use case that is to sign in in order to become a member. After logging in, a member can add reports either for a found pet or for a lost pet. A member also has access to a private section where he can view his own reports and manage them. There he can also see any messages received from his reports. For more clarity, each use case has been described in the next chapter.

These use cases are in their first version. They represent the initial vision of the platform. They serve as a basis for the minimum viable product (MVP), but they may be subject to change since the lean startup methodology may cause the application to change according to the users’ needs.

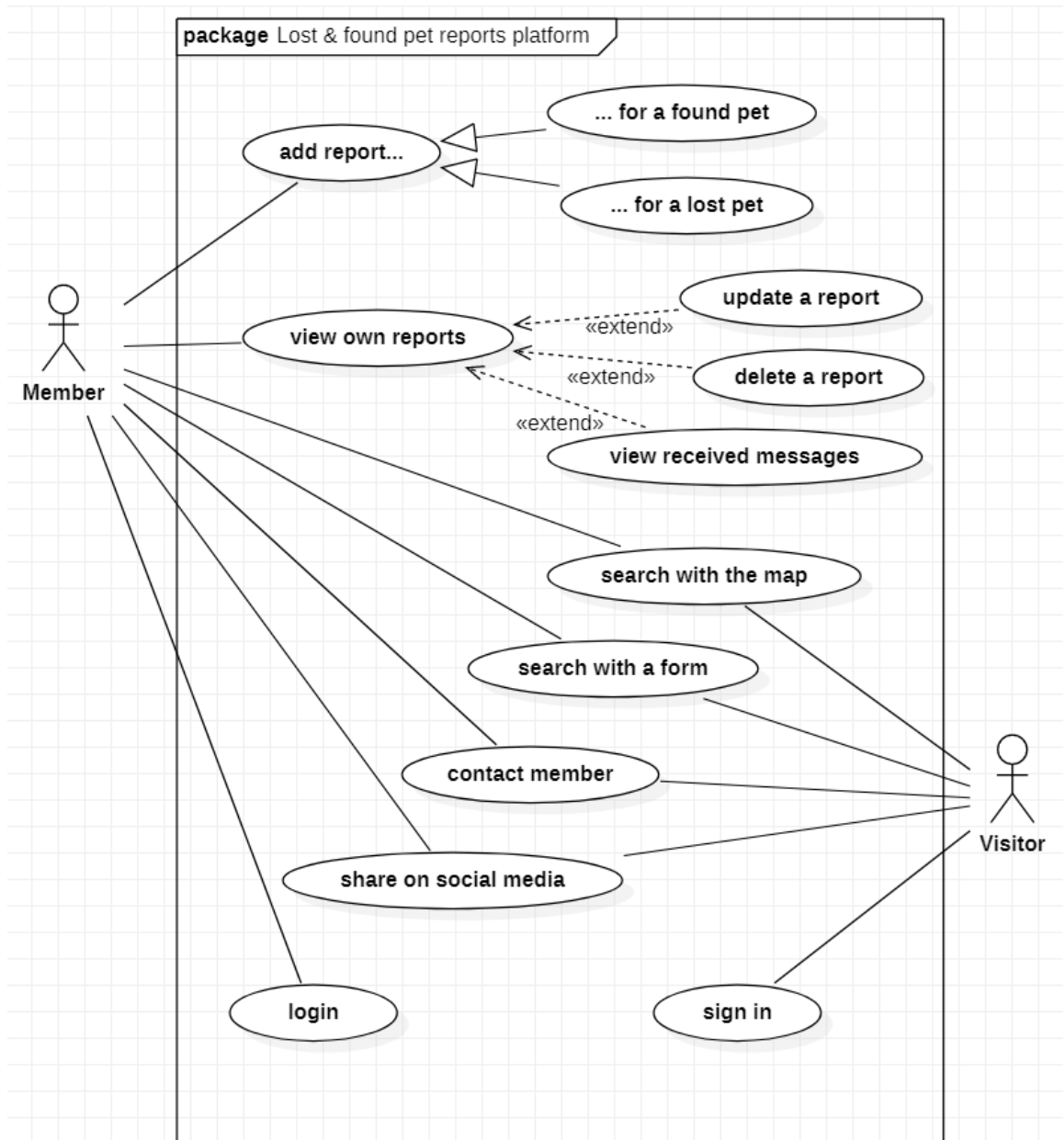


Figure 18 : Initial use cases of the stray animal platform

3.1.1 Use Case Descriptions

Use case name:	Sign in
Actors:	Visitor
Description:	To add a report, a visitor has to create an account. Through a form, he will give personal information such as email, first name, last name, address, phone number. He will also set a password and a username. The system will ask him if he allows to be notified of lost pets near his location (5-10 km of perimeter).

Use case name: Login
Actors: Member
Description: To have access to his own reports and add a new report, a member has to login through a simple form (email and password)

Use case name: Search with the map
Actors: Member, Visitor
Description: A map showing all the reports is accessible for anyone. They can navigate through the map to discover ads and display their details. They can also focus the map on a specific location by adding an address or postal code.

Use case name: Search with a form
Actors: Member, Visitor
Description: If a user needs to be more specific, he can use a form to filter the reports. For instance, he can add information about his pet (color, race, age, breed, color of collar, etc.). After that, he will press a button "search" that will display all reports corresponding to his search.

Use case name: View own reports
Actors: Member
Description: When the member is logged in, he has access to a private section where he can see his current reports. On this page, he can either update them or delete them. This page also displays possible messages received from users through his ads.

Use case name: Update a report
Actors: Member
Description: From his private section, the member can update his report by clicking on the update button. It will open a page with a form with all the current data. He can either change, add, or remove data (unless it is a mandatory field). For saving the changes, he has to click on a save button and will be redirected back to his private page.

Use case name: Delete a report
Actors: Member
Description: From his private section, the member can delete his report by clicking on the delete button. A dialog box will ask to confirm the deletion. Once confirmed, the report is deleted and removed from his private page.

Use case name: View received messages
Actors: Member
Description: From his private section, the member can see a short preview of the messages. He can display the totality of a message by clicking on it. Along with each message, there is information about the sender (email, phone number, first name and last name). He can respond to the sender by email.

Use case name: Contact member
Actors: Member, Visitor
Description: When displaying a report, anybody can click on a “contact” button. It will show a dialog box with a form. The user will have to fulfill personal information such as first name, last name and email. If he is a logged-in user, personal information is retrieved by the system. The user can then type his message.

Use case name: Add report (inherited)
Actors: Member
Description: A member can click on an add button to add a report. After clicking, the system will first ask him if he wants to report a missing pet or a found pet. According to his answer, a personalized form will be displayed.

Use case name: Add a report for a **lost** pet (inheriting)
Actors: Member
Description: After answering that it is a lost pet report, the member will have to fulfill information about the pet and his last location. A picture of the pet can be added too. Also, the user can either enter a location manually or set a point on the map. After validating, the report is sent to the system. When successfully published, the user is redirected to the “search page” and a success message is displayed.

Use case name: Add a report for a **found** pet (inheriting)

Actors: Member

Description: After answering that it is a lost pet report, the member will have to fulfill information about the pet and the approximate location where he found him. In this form, only a few fields are mandatory compared to the lost report. A picture of the pet can be added too. Also, the user can either enter a location manually or set a point on the map. After validating, the report is sent to the system. When successfully published, the user is redirected to the “search page” and a success message is displayed.

4 Project implementation

This chapter corresponds to the practical part of the project. It contains an explanation of the development plan and the setup of the project. The work done in each iteration will be detailed, including the functionalities created, the user testing and the analysis of the results for the next sprint.

4.1 Development plan

As this implementation follows the lean startup methodology, it is developed through iterations. Based on the time available, it has been decided to make two iterations of three weeks. By choosing two sprints, it is possible to test the very principle of lean startup, which is to adapt the product to users' needs.

Each iteration includes two weeks of development and one week of user testing and analysis. The user testing includes a questionnaire along with the actual testing of the prototype. For this purpose, the prototype had to be deployed in order to be accessible for all testers. Firebase Hosting was used in that sense. The testers are a mix of individuals pet owners, employees of veterinary cabinets and employees of animal shelters.

At the end of each iteration, the questionnaire results are collected and analyzed. This analysis is then used to plan the next iteration.

4.2 Setup of the project

Before going into developing the prototype, it was necessary to set up the environment for this project.

The code editor chosen for this project is Visual Studio Code. Along with it, a plugin called Prettier has been used to format the code correctly.

A React project has been created using Create React App. It is a tool that allows the creation of a basic React application without worrying about the setup and configuration. A single command line is enough to create the app:

```
npx create-react-app strayPetsFinder
```

Figure 19: Example of the Create React App command line to setup a React app

At its creation, the project already has a Web App Manifest file and an unregistered Service Worker. These two files will be used later in the implementation of PWA features. Moreover,

Finally, for this iteration, the data structure on Firestore was decided to be as simple as possible in order to maximize time on building the functionalities. It is for this reason that all information about a report is concentrated in a single document. The figure below lists all the properties stored for a report.

```
animal: "Dog"
circumstances: "He got scared and ran away from my car while we were on t...
▶ date: t {seconds: 1585702440, nanoseconds: 0}
email: "rita.moreira96@gmail.com"
firstname: "Paul"
imageUrl: "https://firebasestorage.googleapis.com/v0/b/straypetsfinder.ap...
lastname: "Luck"
lat: 46.47940673650972
lng: 6.855253706234405
location: "Route de Nant, Corsier-sur-Vevey"
petAge: "9"
petBreed: "Labrador Retriever"
petCoatColor: "gold"
petCoatType: "short"
petEyesColor: "brown"
petGender: "Male"
petName: "Fox"
petParticularInfo: "He has a blue collar and is very scared of cars and k...
petSize: "large"
status: "lost"
```

Figure 21: Simple data structure of a report for the first MVP

4.3.1 Show reports with a map

For this functionality, the first thing that was tried was the implementation of a simple map with markers on it. Using only Google Maps API was not ideal because it was said to be challenging to customize its map. During our research, a react library called google-map-react was found instead. This library wraps Google Maps API and allows to render any react component on the map. This library seemed perfect for our needs as we wanted to custom the markers and display the details of the report in a react component. Thus, a map with fake data was first implemented. It went well at first. However, as more complexity was added, we realized that the documentation was weak and that the demos provided were not up to date according to the documentation. Worried that the library would soon become obsolete, we started to look for another library for the map.

While discussing this issue with another student, he presented a library that he has been using for that purpose. This library is called Leaflet and is a recognized open-source Javascript library for mobile-friendly maps. The documentation was comprehensive, making the coding much more comfortable. In addition to that, a react library called React-Leaflet, which converts Leaflet map elements into React components, was available, making the coding even easier.

The figure 24: Implementation of a Leaflet map showing the reports below shows the implementation of the Leaflet Map with react components. As it is shown, all the elements of the map are React components: Map, TileLayer, Marker and Popup.

The Map component is the top-level element containing the map. TileLayer is used to load and display tile layers on the map. The latter are web-accessible bits of a map located on a server and are fetched through an URL template like "https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png". The "s" corresponds to one of the available subdomains, "z" the zoom level of the map, "x" and "y" the tile coordinates. Markers are clickable icons on the map, and Popup is the box that appears when a marker is clicked on for instance.

```

<Map
  center=[[this.state.lat, this.state.lng]]
  zoom={this.state.zoom}
  style={{ width: "100%", height: "85vh" }}
>
  <TileLayer
    url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
    attribution='&copy; <a href="http://osm.org/copyright">OpenStreetMap</a> contributors'
  />
  { //loop into each report and show marker
  this.props.reports.map((report, key) => {
    const point = [report.lat, report.lng];

    return (
      <div key={key}>
        <Marker
          position={point}
          key={key}
          icon={report.status === "found" ? MarkerFoundPet : MarkerLostPet}
          onMouseOver={(e) => {e.target.openPopup()}}
          onMouseOut={(e) => {e.target.closePopup()}}
          onMouseDown={() => this.handleClickOpen(key)}
        >
          <Popup>
            <span>
              {report.status.toUpperCase()} {report.animal.toUpperCase()}(
                {report.date.toDate().toLocaleDateString()}) by {" "}
                {report.firstname} {report.lastname.substring(0, 1)}.
            </span><br />
            <img src={report.imageUrl} width="200" height="auto" alt="pet"/><br />
          </Popup>
        </Marker>
        {this.state.selectedPost === key //Show dialog when it's selected
        ? this.dialogDetail(report)
        : null}
      </div>
    );
  })
</Map>

```

Figure 22: Implementation of a Leaflet map showing the reports

In order to display a marker for each report, we are using a `map()` method to loop into the array of reports and return a marker. The marker icon changes based on the type of report: the marker is green if it is a found report or red if it is a lost report.

To get all reports, an API call is made to Firestore when the page containing the map is mounted. The next figure represents the call to Firestore. Firestore services are imported under the constant "db". Once imported, it is almost the same as the example given in the chapter about Firestore (Figure 13: Getting data from "LIS" document within "cities" collection. First, the Firestore service is called, then the collection is indicated (here "reports").

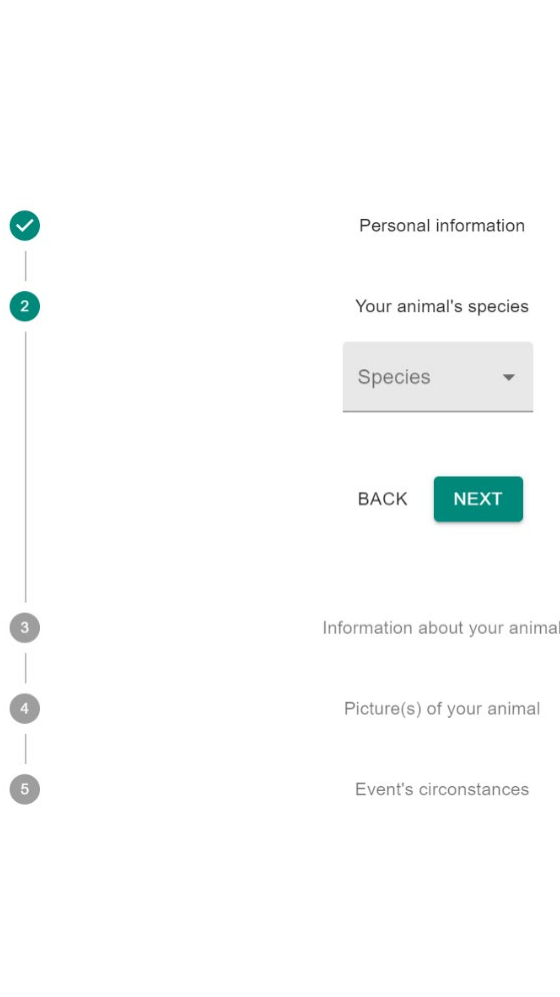
And, as we want all documents within this collection, the `get()` method is used at that level. Once the documents are fetched, they are saved in the “reports” state.

```
8   state = {
9     reports: []
10  };
11
12  componentDidMount() {
13    //Get all reports from collection
14    db.collection("reports").get().then((snapshot)=>{
15      var reports = snapshot.docs.map(doc => doc.data());
16      this.setState({ reports: reports });
17    })
18  }
```

Figure 23: Code to get all reports from Firestore

4.3.2 Add a lost report

This functionality was the most time-consuming functionality of the iteration. Indeed, as there is much data to fill in for a report, we chose to use a stepper to divide the form in multiple smaller forms (see figure below), making however the development more difficult.



```
19  function getSteps() {
20    return [
21      "Personal information",
22      "Your animal's species ",
23      "Information about your animal",
24      "Picture(s) of your animal",
25      "Event's circumstances",
26    ];
27  }
28
29  function getStepContent(
30    stepIndex,
31    handleNext,
32    handleBack,
33    activeStep,
34    handleChange,
35    handleLocation,
36    handleImageUrl,
37    report,
38    petOptions
39  ) {
40    switch (stepIndex) {
41      case 0:
42        return (
43          <PersonalInformationForm
44            handleNext={handleNext}
45            handleBack={handleBack}
46            activeStep={activeStep}
47            handleChange={handleChange}
48            report={report}
49          />
50        );
51      case 1:
52        return (
53          <AnimalSpecies
54            handleNext={handleNext}
55            handleBack={handleBack}
56            handleChange={handleChange}
```

Figure 24: stepper implementation

A stepper is a Material-UI component. According to the documentation, the steps are first defined (see the figure above, lines 21 to 25), then a switch returns the content of the step based on the step number. Here each step corresponds to a mini form and is in fact, a custom component. For instance, in the figure above, PersonalInformationForm is a custom component that is rendered when the user is at the first step (index 0).

As explained in the theory (see 2.6.2), custom components can receive data as properties. These properties then become props within the component. The object “report” that is being filled in through the form is passed as a prop to them (line 48). The same happens for the actions of the stepper (handleNext, handleBack,...). Moreover, we explained in the theory that React has a uni-directional data flow, which means that the mini forms cannot update the “report” object in the parent directly. Instead, functions in the parent that are responsible for changing the state of the object are passed as props too (handleChange, handleLocation, handleImageUrl,...).

Validation of the fields

The validation is made thanks to “Yup” a Javascript Object Schema Validator. By defining the type of data or any other requirements we are expecting from a form, Yup can determine if the form is valid or not. In the following figure, we can see the Validation Schema of the personal information form, stating that two text fields are required and that one email field is required too. Also, in parenthesis, an error message can be specified.

```
const validationSchema = yup.object().shape({
  firstname: yup.string().required("Firstname is required").max(20),
  lastname: yup.string().required("Lastname is required").max(20),
  email: yup.string().email("Invalid email").required("Email is required"),
});
```

Figure 25: Example of a Yup Validation Schema

The validation is done when the next button is pressed (see the Figure 26: validation function). ValidationNext is the function in charge of comparing the values with the schema. If the form is valid, the user goes to the next step (props.handleNext()). If it is not, the errors are saved with setErrors() and then rendered in red in the form.

```
<Button
  variant="contained"
  color="primary"
  className={classes.button}
  onClick={() =>
    validationNext({
      validationSchema,
      props.report,
      props.handleNext,
      setErrors
    })
  }
/>
Next
</Button>
```

Figure 26: validation function

Options and list of breeds

For now, the implemented species are limited to cats, dogs and rabbits. Since each of these species has a different list of breeds, it was necessary to do some research to collect all the breeds. The length of each list varies a lot based on the species. For instance, we found more than 300 species of dogs. These lists have been retrieved from Wikipedia.org and then formatted with Excel. For now, these breeds are stored locally in a JS file, but they may be moved to Firestore to lighten the app.

Other lists have been stored in Firestore. For instance, all the options regarding the appearance of the pet, such as its size, its coat type, its coat color are fetched when the page containing the form is rendered.

These lists have been implemented using a Material-UI component called Autocomplete, which is already published on their website but not part of the core of the library yet. This component looks like a select input with autocomplete. It has been chosen especially for the breed field because, as some lists are very long, the possibility to type and show the breeds corresponding to the research is better for the user (Figure 31: Autocomplete demo of lists based on research as a demo).

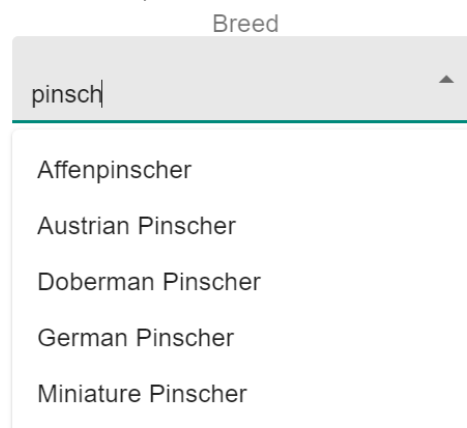


Figure 27: Autocomplete demo of lists based on research

Image handling

At some point in the process of creating a lost pet report, it is required to add a picture of the pet. For that purpose, it is both Cloud Storage and Firestore services that are needed. First, the user must pick an image. When the image is picked, the local URL of the image is saved in a variable in order to preview the selected image. The raw image is also saved in another variable to upload it when the user clicks on the "next" button. When that button is clicked, a validation method is called to determine if there is a selected image. If it is the case and if that selected image has not already been uploaded, the raw image is uploaded

to the cloud and the user can go to the next step. If no image has been picked, an error message defined with Yup is rendered. The figure below shows the upload method.

```
//image upload (with blob)
const handleUpload = () => {
  console.log(image);
  const uploadTask = storage
    .ref("images")
    .child(props.report.email)
    .child(image.name)
    .put(image); //raw image

  //monitoring of the upload
  uploadTask.on(
    "state_changed",
    (snapshot) => { //progress function
    },
    (error) => {
      // error function
      console.log(error);
    },
    () => {
      // complete function
      //Retrieve download URL of the image stored in Firebase
      storage
        .ref("images")
        .child(props.report.email)
        .child(image.name)
        .getDownloadURL()
        .then((urlimage) => {
          console.log("url download", urlimage);
          props.handleImageUrl(urlimage); //save that URL
        });
    }
  );
};
```

Figure 28: Method that uploads an image into Firebase Cloud Storage

As explained in the theory, Cloud Storage can be compared to a filesystem, even if there are no real directories or folders. Indeed, by using slashes (/), we can make objects appear like they are part of a directory structure. Here that directory structure corresponds to “images/user email/image name”.

Once the path of the image is defined, by simply using a put() method with the raw image as a parameter, the uploading service of Firebase is launched. When the upload succeeded, the URL of the uploaded image is retrieved by first referring to its path and then calling getDownloadURL(). After that, the URL is saved in the “report” state through the method props.handleImageUrl() implemented in the parent component.

Geocoding to set the location of the event

In order to show the report on the map, it is necessary to know the location of the event. Geocoding is the process used to get coordinates from an address (called forward geocoding) and vice-versa (called reverse geocoding). For this first iteration, only the reverse geocoding has been implemented. Thus, the user was asked to click on a point on the map to obtain its address.

By clicking on the map, coordinates are returned by the Leaflet map. With these coordinates, the address is fetched from a geocoding API developed by MapQuest. This service is free up to 15.000 requests per month. The coordinates and the translated address are then returned in a callback function. In this case, it is a method that saves the location and the coordinates into the “report” state in the parent component.

```
//geocodeReverse : from lat,lng, return address in a callback
export const geocodeReverse = async (lat, lng, callback) => {
  let locationString;
  fetch(
    "https://www.mapquestapi.com/geocoding/v1/reverse?key=XDt7bT6BSbau4x8QXPYR3AqtRMKf1dPR&location=" +
    lat +
    ", " +
    lng +
    ""
  )
  .then((response) => response.json())
  .then((data) => {
    locationString =
      (data.results[0].locations[0].street !== ""
        ? data.results[0].locations[0].street + ", "
        : "") + data.results[0].locations[0].adminArea5;
    callback(lat, lng, locationString); //return result
  })
  .catch((err) => console.error(err));
};
```

Figure 29: Reverse geocoding method: from latitude and longitude, an address is fetched

4.3.3 User testing and analysis

According to the lean startup methodology, after implementing the MVP, there is the product validation phase. One week in the sprint was allocated for that purpose.

A questionnaire was sent to potential users. It included private individuals, veterinary cabinets and animal shelters. In total, almost 30 persons were reached out with 50% of answers. Of these respondents, 3 were employees of veterinary cabinets, 1 employee of an animal shelter and 13 private pet owners. As this platform will potentially be further developed at the end of this project in Switzerland, the testers were mainly Swiss residents (88%). However, it would be interesting for the next iteration to test the prototype to other nationalities.

For this first iteration and in addition to feedback on the prototype, we wanted to know how the testers behaved towards stray pets reporting. Therefore, the first part of the questionnaire was devoted to finding out their habits in this area. The second part of the questionnaire was about testing the prototype. For this, the prototype was deployed on the web and the URL (<https://strappetsfinder.firebaseio.com>) given to testers.

The following table presents a summary of the most relevant questions. For the full survey and results, see Appendix 2. Questionnaire and results of the first iteration

Table 1: Summary table of the first survey

1. Have you ever lost one of your pets?				
Yes		No		
41% (7)		59% (10)		
2. Did you report his disappearance by any means?				
Yes		No		
86%(6)		14%(1)		
3. If yes, what mean(s) did you use?				
Post an ad on social media	Hang posters in your neighborhood		Post an ad on a website	
50%	67%		17%	
Contact the police	Contact a shelter		Contact the veterinarian	
17%	-		-	
4. Do you think there is room for improvement in this area?				
Yes	Surely	I do not know	Hardly	No
57%	43%	-	-	-
5. Have you ever shared lost pet ads?				
Yes		No		
59%		41%		
6. If you answered yes, for what reason(s) did you share?				
It was the pet of a friend	It was the pet of a neighbor.		I am sensitive to the animal cause	
40%	10%		30%	
It was an act of kindness	It was a pet from my region		It is part of my job	
40%	20%		30%	

7. If you answered no, for what reason(s) did you not share?			
The ad was out of my area	I did not know the owner	I did not have the idea of sharing	
86%	28%	14%	
8. What do you think of the map system?			
Helpful color code, good, nice design, smooth map manipulation		No possibility to filter/search reports, No possibility to share the ad,	
9. What do you think of the form for adding a lost pet?			
Simple, comprehensive, easy to use, lot of choices, very good		Lack of multiple choice for coat/eyes colors and mixed breeds, the form is in English	
10. If you had to choose 3 features to develop among them, which ones would you choose? Choose 3 to rank by importance			
(Display a list of ads (besides the map), the addition of several images per ad, ability to share on social networks, management of my ads (modify, delete), Notification when reports are published in my region, the addition of "sighted" pet ads)			
1	2	3	
List of reports in addition to the map	Notification when ads are posted on my area	Report management (update, delete reports).	
Possibility to share a report on social networks	Addition of "sighted" animal reports		
11. If this platform were to be finished and put online, do you plan to use it?			
Yes	Maybe	No	For professional use only
70,5%	17,5%	6%	6%

Analysis of the results

Based on these results, it is possible to realize that for these potential end-users, whether individuals or professionals, stray pets' reporting are a frequent and common occurrence (41% has ever lost one of his pet and 59% has ever shared a lost report). Although the current means of doing seem to suit most, there is still a general feeling that improvements can be made (57% answered "yes" and 43% "surely"). Also, being able to share reports quickly is an essential element for them as it was the most voted feature (see question 10). Finally, the fact that they tend to share reports from their region (by 87%) is a reason to develop a notification system.

When they were asked if they had encountered any problems using the app, 50% of them claimed everything went well. For the rest, 30% said they were able to publish their report but could not find it on the map. After checking in the database, the reports were well saved,

except for the coordinates, which were set to zero. This may be because they typed the address instead of picking a location on the map. The validation of the coordinates must be revised for the next iteration in order to avoid this.

About the system of geolocation and the map (question 8), most of the testers found it great. Some appreciated the smoothness of the navigation, the different colors to distinguish lost/found ads. Others pointed out the lack of filter options (to display only the lost dogs for instance), and one suggested adding a “share” button for social media.

About the form of a new report (question 9), most of the respondents declared that it is easy to use and comprehensive. However, some testers pointed out the lack of possibility to select several colors for the coat of the animal. Another one noted the lack of mixed breeds. A respondent also suggested the opportunity to choose the language of the form.

At the end of the questionnaire, the respondents could give any ideas for the application. One suggested the possibility to directly comment on the report to facilitate the research thanks to testimonials. Another recommended the creation of a profile with personal information and thus making the user more legitimate. The idea of rewards as motivation was mentioned too.

Generally speaking, the initial vision we had of the platform corresponds reasonably well to the needs of the users. Indeed, the testers showed interest in the map as it offers an engaging visual approach to pet reporting. However, they put more emphasis on the "social" side of the app with the integration with social networks and the notification system.

Besides, some bugs have been detected, such as coordinates set to zero for some testers or the lack of multiple colors for the coat. Through this, it is possible to realize that indeed, by creating an MVP in a short time with the lean startup methodology, the quality is diminished. However, at the same time, the advantage of this is that the bugs are noticed more easily thanks to user testing.

For the next iteration, these bugs are going to be fixed. In addition to that and following the feedbacks, a list of reports and the management of own reports will be developed. This last functionality will require moreover user authentication. If time allows, the implementation of notifications will be made.

4.4 2nd iteration

This iteration was about adding new functionalities on top of the minimum viable product created in the last iteration. These functionalities were decided at the end of the latter based on the results of the user testing:

- listing of reports
- management of reports (deletion and edition)
- user authentication
- bugs fixing (zero coordinates, lack of multiple colors for the description of the pet)

The result of the prototype (version 0.2) by the end of the sprint can be found in appendices (Appendix 4. Questionnaire and results of the second iteration).

4.4.1 Addition of a list of reports with filter options

In order to build the list of reports, a custom component called “ListOfReports” was created. This component is rendered on the home page of the platform. This page oversees the passing of fetched reports to both the map and the list.

The list is made with “MaterialTable”, a component for complex data tables recommended by Material-UI. This component offers features such as edition, deletion, sorting, and filtering. The configuration of this component is straightforward. The data to display, which must be an array of objects, is declared. Then the columns of the table are defined by a title and linked to a field stored in the data defined earlier. It is possible to create custom columns by using the “render” property of MaterialTable. This property has been used in this project to show the picture of each pet. The figure below shows this custom render from line 28 to 34.

```
21 | <MaterialTable
22 |   icons={TableIcons}
23 |   columns={[
24 |     {
25 |       title: " ",
26 |       field: "imageUrl",
27 |       sorting: false,
28 |       render: (rowData) => (
29 |         <img
30 |           alt={"image of"+rowData.petName+" for report published on "+rowData.date}
31 |           src={rowData.imageUrl}
32 |           style={{ width: 150, borderRadius: "2%" }}
33 |         />
34 |       ),
35 |     },
36 |     { title: "Date", field: "date", type: "date" },
37 |     {
38 |       title: "Distance from you",
39 |       field: "distanceFromPosition",
40 |       render: (rowData) => (rowData.distanceFromPosition+" km"
41 |     ),
42 |     hidden: props.manualLocation?true:false
43 |   },
44 |   { title: "Nom", field: "petName" ,sorting: false},
45 |   { title: "Status", field: "status" },
```

Figure 30: definition of the columns for the list of reports

It has also been used in lines 38 to 42 to display the distance in kilometers from the current user position. The line 42 tests if the location is given manually (meaning that the user has not allowed to get his current location). If it is true, the distance field is hidden.

In order to allow users to see the details of a report from the list, the dialog component used in the map to show the detail of an ad has been reused. As stated in theory, the reusability is one of the significant advantages of component-based frameworks like React.js.

A filter system, another functionality asked by some testers, has been implemented along with the list of reports. It seemed more appropriate to build it right after the listing since they are linked. This filter system updates the list of reports on both the list and the map. It is also possible to type an address to show it on the map. For this purpose, forward geocoding has been implemented using the MapQuest API again.

4.4.2 Management of own reports

A majority of the testers from the last survey selected the management of own reports as the third most crucial functionality to be developed. As a user must be registered to retrieve his reports, user authentication has been implemented using Firebase Authentication.

User authentication

For the sake of speed and simplicity, only the method using email and password has been activated in Firebase. A sign-up form was created to register a user. When the form is submitted, the user is created in Firebase, and if it is successfully created, its data (address, email, names) is stored in Firestore. After that, the user is redirected to the home page and is automatically signed in. The following figure shows the code for the creation of an account.

```

//create user account
auth
.createUserWithEmailAndPassword(userInfo.email, userInfo.password)
.then(() => {
  let uidUser = auth.currentUser.uid;
  //save user info
  db.collection("users")
  .doc(uidUser)
  .set(userInfo)
  .then(() => {
    //message account saved + redirect to home page
    showSnackbar({
      severity: "success",
      message: "Your account have been successfully created",
    });
    //redirect after 2 seconds
    setTimeout(() => {
      history.push("/");
    }, 2000);
  })
  .catch(() => {
    //show error
    showSnackbar({
      severity: "warning",
      message: "Your account couldn't be saved. Try later.",
    });
  });
})
.catch((error) => setErrors({ path: "", message: error.message }));

```

Figure 31: creation of a user using email and password in Firebase

The interface of the platform needs to be different whether a user is logged in or not. For instance, a logged-in user will have access to a private page for reports management. He also can add new reports while a non-registered user will be asked to log in for that. Because of these differences, the state of the current user is saved in a “context”. The latter is used to ensure that there is only one instance of that data throughout the application. The values passed to this context are then available to any components in the app.

Edition and deletion of reports

A user can manage his reports and personal information on a private page created for this purpose. His ads are again listed using MaterialTable. This time, however, actions such as editing and deleting have been added. The deletion operation proposed by MaterialTable has been reused, while the editing one, on the other hand, was too limited for the needs of the app. A custom action was created instead.

When editing an ad, the user is redirected to a page containing the same form as for its creation. Simultaneously, the selected report is also passed as a prop, and its data is loaded into the form. The following figure shows the custom edit action with the redirection to another page (`history.push()`) and the report being passed.

```
actions=[  
  {  
    icon: TableIcons.Edit,  
    tooltip: "Edit",  
    onClick: (event, rowData) =>  
      history.push({  
        pathname: "/edit-report",  
        state: { report: rowData },  
      }),  
  },  
]
```

Figure 32: redirection to the edit page with a report passed as prop

The form used initially for the creation had to be modified in order to also handle the editing. In particular, the upload of changes had to be modified so that Firebase call is an update when editing and not an add. Because of that, another call to Firestore had to be implemented. This one uses an `update()` method and needs the id of the report to save the changes in the right document (see Figure 33: Update of a report stored in Firestore).

```
const sendReport = () => {  
  //EDIT MODE : update report  
  if (report.id) {  
    db.collection("reports")  
      .doc(report.id)  
      .update(report)  
      .then(function () {  
        console.log("Document successfully updated!");  
      })  
      .catch(function (error) {  
        // The document probably doesn't exist.  
        console.error("Error updating document: ", error);  
      });  
  } else {  
    // ADD MODE : Add a new document with a generated id.  
  }  
}
```

Figure 33: Update of a report stored in Firestore

4.4.3 Other changes

The other main task of this second iteration was fixing bugs found in the last version of the prototype. The first bug to be fixed was the problem of the position set to zero for some users. The possibility of getting the location from a typed address was implemented to solve this issue. The second bug was the lack of possibility to set multiple colors for the coat and

eyes of a pet. This functionality has been added by modifying the Autocomplete field used for these fields. They now allow numerous selections and the user can type other colors if needed.

Finally, regarding the PWA implementation, the Manifest file generated during the project creation has been updated with information and icons related to the platform. In addition to this, the Service Worker present by default in the react app has been registered, allowing browsers to now show the install banner. Now, once installed, the application can be launched from the device of the user and without the browser frame. The figure on the left shows the install banner and the launching screen of the app without a frame.

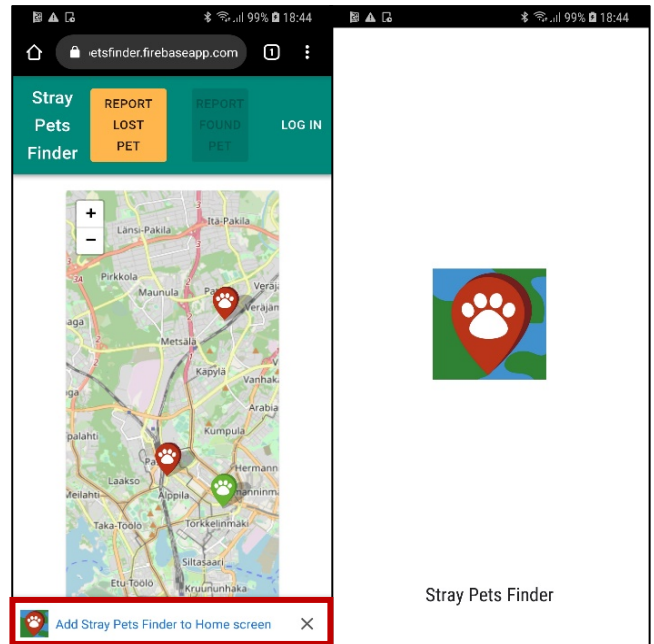


Figure 34: install banner and launching screen of the app

4.4.4 User testing and analysis

As done for the first iteration, a questionnaire was sent to target users. This time the survey was sent, besides Switzerland, to professional institutions in Portugal and Finland. Because of that, a second version of the questionnaire was created in English. The survey and its results can be found in the appendices (Appendix 4. Questionnaire and results of the second iteration). In total, 43 persons, including professional institutions, were contacted, and the response rate is 32%, with 13 answers among which 2 professionals.

In this questionnaire, testers were asked to navigate through the reports using both map and list, create an account, add a report, edit it, and finally delete it. They then were asked to vote for future functionalities and for a way to contact the author of a report.

The following table shows a summary of the relevant results of this questionnaire. The results in bold mean that these answers were the most frequent ones.

Table 2: Summary table of the second survey

1. What do you think of the location and map system?				
Easy to use, excellent, very good, simple and effective, nice design, works well, reports are easily found, good idea		lack of caption to differentiate the colors of the ads		
2. What do you think of the search system on the right side of the map?				
effective, great, simple, nothing to say, useful for refining searches quickly		cannot find a city with the only the zip code, location not very precise, lack of species for filtering		
3. What do you think of the list of reports below the map?				
great, intuitive, sort by criteria is good, logical display of ads		did not see the list, lack of information displayed, not aesthetic		
4. What do you think of the form to create an account?				
simple, ordinary, nothing to say, ok, only the essential is asked, quick to fill		add a "cell phone" field, add a "confirm password" field, add an "owner description" field, one tester could not create the account		
5. If you had to choose three functionalities to develop among these, which ones would you choose? Choose 3 to rank by importance				
(additions of several images per report, ability to share on social networks, notification when reports in my area are published, adding reports of animals "sighted", possibility of "monetized rewards" for motivation, possibility to comment directly on the report, choice of language for the platform)				
1		2		3
Possibility to share a report on social networks		Notification when reports in my area are published		Adding reports of animals "sighted"
6. Which of the following options below would you prefer to contact a report?				
Click on a "contact" button and send a private message		Click on a "contact" button and send an email		Comment below the report directly
Obtain the telephone number and call directly.				
46% (6)		23% (3)		23% (3)
				30% (4)
7. Would you be willing to receive notifications when a lost pet report is published near you?				
Yes			Maybe	
85% (11)			15% (2)	
8. If so, up to what range? (km)				
2 km	5 km	7 km	10 km	+10 km
23% (3)	39% (5)	-	23% (3)	15% (2)
9. If this platform were to be finished and put online, do you plan to use it?				

Yes	Maybe	For professional use only
69% (9)	23% (3)	8% (1)

Analysis of the results

To start with, the response rate is lower compared to the first questionnaire (50%). There are several reasons for this. Firstly, the time available (1 week) may be too short. Also, testers may have given up when they saw that they had to create an account for security reasons. Alternatively, maybe the length of the test demotivated them.

Comparing with the first questionnaire, it is possible to confirm that users are looking for a social and local dimension as the two primary voted functionalities were the share on social media and notifications based on location. Regarding the means to contact, they prefer by 46% to send private messages on the application. Now that user authentication has been implemented, these functionalities can be developed for the next iteration.

Some problems were encountered in this version. For instance, the implementation of the forward geocoding (convert an address into coordinates) must be improved to show the right place when the user enters a zip code or the name of a city. One solution would be to ask the country before the address for more precision or otherwise change the API. Other minor details were revealed, such as the too hidden design of the list and the missing fields in forms. Besides these issues and considering their positive comment, the actual implementation seems to meet the needs of target users well.

The next iteration that will be done outside the scope of this work will include the implementation of the two voted functionalities, the addition of found reports, and the contact for a report. The next iterations will last one week longer to implement these.

5 Discussion

In this final chapter, the results of this project will be analyzed and discussed. First, an analysis of the implementation of the lean startup approach will be made. The usability of the product will be evaluated, as well as the own learning of the author. A conclusion will finally summarize the results found.

5.1 Results of the lean startup approach

Regarding the lean startup approach, the first point that can be observed is that the importance and quality given to the test phase is essential to have useful results.

As stated in the theory, vanity measures should be avoided since they do not reflect real product progress. For this reason, many questions in the surveys were open-ended questions in which testers are invited to write their opinion. Despite this precaution, many of the answers were brief and generic such as "simple and effective" and "very good".

This leads to the idea that doing user testing remotely using a questionnaire is not an ideal approach. Indeed, by doing the test remotely, we miss valuable information, such as testers' first impressions. On top of that, with only written feedback, it is difficult to understand what the user has written. Indeed, without a context or being with him to ask for clarification, his answers can be misinterpreted or not understood. A solution to these disadvantages would have been to perform live user testing. However, since the project is intended to focus mainly on Switzerland and most of the testers were therefore in Switzerland, it would not have been possible to carry out these live tests. What is more, with live user testing, there would surely have been less responses since more time and organization would have been required to run the tests.

Regarding the response rate, it dropped considerably in the last questionnaire, from 50% to 32%. Several reasons can be found for this. Firstly, the planned response time of 5 working days is probably too short, especially for professionals. Secondly, contacting the testers without prior notice may surprise them and hold them back from responding. Indeed, by contacting potential testers by e-mail without asking for their prior consent and without presenting the context of the tests, they may not feel sufficiently involved to take the time for it. All these observations raise the idea that the time put into the planning of the test phase is critical in order to obtain the best possible results.

This project has also led to the realization that it is not always possible to follow the primary needs of the target users strictly. Indeed, sometimes to be able to develop one of their needs, it will be necessary to implement another functionality not primarily requested. This

was the case for the request of notifications and sharing on social networks. In the first questionnaire, testers stated that they wanted these functionalities, but to develop them, it was necessary to first implement the authentication of users.

Furthermore, this project has highlighted the fact that by focusing on a few features at each iteration, initial ideas and requirements tend to be set aside. For example, while looking for productivity in order to deliver a functional product by the end of each iteration, responsiveness and many PWA features such as offline availability or notifications have been shelved due to lack of time.

All of this showed that the lean startup approach could be followed up to a certain limit. This limit will vary depending on the time allocated to each sprint, the number of developers working, and the product itself. If more time was planned per iteration and more people were working on this project, some of the points discussed earlier would not have happened.

In the end, the lean startup approach seems to be an effective methodology in terms of the speed of implementation of a minimum viable product. The constant validation of the product through user testing saves time and energy, as it is no longer necessary to make assumptions. However, this project has shown that this approach would work better with a team of developers and with more time available per sprint. Indeed, by distributing the workload, the quality of what is developed would better meet the requirements set initially, and the test phases would be better organized and planned.

5.2 Usability of the prototype

Concerning the usability of the prototype, the questionnaires showed that veterinarians were more interested in this product than animal shelters. Unlike for the private sector, the response rate for professionals was low. Out of more than 30 institutions contacted, only 6 responded to the questionnaires. Of these 6, 4 worked in veterinary practices and 2 in shelters. In addition to that, two of the shelters contacted replied that they already have their own tool for stray pet reporting. One of them also mentioned that the Swiss platform "stmz.ch" (mentioned in the introduction) was used by many of the shelters in their region and therefore, they believe that an additional platform would only disperse the research. This may explain why the response rate of shelters were lower than for veterinarians. It also seems that, compared to shelters, many veterinary cabinets do not use this type of tool. They are indeed more interested in this platform. One even replied that if this platform could reach a large audience, he would recommend it to their clients. Another thought that it was a way to highlight the number of disappearances, something they are

generally not aware of. Individuals also do not seem to know the platforms used by professionals. They are therefore, much more interested in this platform. Several said that it was an "innovative" idea that is currently lacking.

The results of the questionnaires also showed that the success of this platform would depend on its integration with social media. Indeed, the two most voted features in each survey were the possibility to share on social networks and a notification system. This focus is justified by the objective of reaching as many people as possible. This interest is beneficial for the platform as it would increase its visibility and attract more users.

However, in order to make the platform known and to help it grow, a strategy will have to be put in place. Firstly, instead of going into production with the MVP right from the start, a practice recommended by the lean startup, it would be better to wait for the app to be finalized with almost all the essential features. This would include the implementation of the contact, the addition of found pets, and social network integration. As explained before, the testers stated that they would be ready to use the application if it can reach many people, and for this, at least the integration of social media must be implemented. This waiting seems to be the best solution to avoid a shaky first impression.

Once the application is judged ready to launch, several tactics can be put in place to promote and grow the app. It will first be necessary to decide on a target region where to carry out the different marketing actions to make this application known. After that, the second action will be to continuously post lost and found reports of that region into the app. By adding content to the application, it becomes more realistic and useful for potential users. Once this is done, all the Facebook groups that share lost pet ads in this region become the prime target users to reach. If their administrators agree to share this platform with their subscribers, it will give high visibility. It would be even more efficient if they would use this platform to share their reports.

In parallel, as the testers in this project saw a raw prototype without much design, once it is launched, it would be useful to contact them again to present the platform in production. The before/after effect could surprise them in a good way and push them to talk about it around them. For this reason, the email of all the people contacted for the surveys have been saved.

5.3 Author's own learning

This work helped to deepen the author's personal knowledge of React and Firebase. She was able to clarify some aspects of React.js that had remained elusive before this project, such as the extensive use of react-router, useContext and useEffect. The research on the

PWA has been enlightening, and she is now aware of the growing need for solutions adapted to both computers and mobiles.

In addition to the technical skills learned, the author realized the importance of user validation. Of course, it takes time, requires organization and it usually means giving less significance to the coding phase, but this approach ensures a realistic vision of the project being developed. Whether the feedback is positive or negative, the author found it to be a source of motivation to continue offering a product as close as possible to what the market needs. In the past, she was often apprehensive about evaluating her work, but thanks to this project, she learned to let go and see the positive in this process. This project allowed her to learn how to organize herself by planning deadlines in advance and splitting the different phases of implementation as well. She additionally realized that the time set for the test phase was not enough. Also, in order to have more answers to the questionnaires, a better outreach to professional institutions could have been done.

Because of the time and the features requested by the users, the author did not have the possibility during this project to implement sophisticated PWA features such as notifications. However, she will be able to do it in the future development of the application since notifications are planned to be implemented.

5.4 Conclusion

This thesis work consisted of the development of a PWA prototype of a platform for stray pet reporting. Its implementation was carried out following the lean startup approach. Because of this approach, user testing was conducted to know the needs of potential users and validate the usability of this platform. These tests showed that private and veterinarians were the most interested in this product and that the success of this platform will depend on its ability to integrate into social networks.

To promote this product, a strategy that has briefly been introduced will have to be put in place after the launch of the product.

Regarding the way this project was implemented, it was determined that one week was not enough for user testing and that remote testing was not sufficient to get an accurate interpretation of answers.

Regardless of the future outcome of this platform, this project has highlighted the general lack of communication of existing platforms for individuals and veterinarians in Switzerland. These same platforms could integrate the use of geolocation to give the social and visual dimension that testers appreciated in this prototype.

6 References

- American Society for the Prevention of Cruelty to Animals - ASPCA, 2018. *Facts about U.S. Animal Shelters*. [Online]
Available at: <https://www.asPCA.org/animal-homelessness/shelter-intake-and-surrender/pet-statistics>
[Accessed 08 05 2020].
- Awad, M. A., 2005. *A Comparison between Agile and Traditional*, s.l.: The University of Western Australia,.
- BuildFire, 2019. *Mobile App Download and Usage Statistics (2019)*. [Online]
Available at: <https://buildfire.com/app-statistics/>
[Accessed 11 03 2020].
- Charvat, J., 2002. *Heavyweight vs. lightweight methodologies: Key strategies for development*. [Online]
Available at: <https://www.techrepublic.com/article/heavyweight-vs-lightweight-methodologies-key-strategies-for-development/>
- Chinnathambi, K., 2016. *Learning React*. s.l.:Addison-Wesley Professional.
- Codecademy, 2020. *React: The Virtual DOM*. [Online]
Available at: <https://www.codecademy.com/articles/react-virtual-dom>
[Accessed 14 03 2020].
- Code, C. o., 2019. *React Popularity and When Not to Use React*. [Online]
Available at: <https://scotch.io/starters/react/react-popularity-and-when-not-to-use-react>
[Accessed 16 03 2020].
- Cowart, J., 2012. *What is a Hybrid Mobile App?*. [Online]
Available at: <https://www.telerik.com/blogs/what-is-a-hybrid-mobile-app->
[Accessed 10 03 2020].
- Dufour, L., 2020. *Qu'est-Ce Que Le Lean Startup ?*. [Online]
Available at: <https://www.leblogdudirigeant.com/quest-ce-que-le-lean-startup>
- Enyinnaya, C., 2019. *Demystifying The Service Worker Lifecycle*. [Online]
Available at: <https://www.digitalocean.com/community/tutorials/demystifying-the-service-worker-lifecycle>
[Accessed 12 03 2020].
- Existek — Software Development Company, 2019. *Hybrid VS Native App: Which one to choose for your business?*. [Online]
Available at: <https://medium.com/existek/hybrid-vs-native-app-which-one-to-choose-for-your-business-e51542554078>
[Accessed 10 03 2020].

Facebook Inc., 2020. *React*. [Online]
Available at: <https://reactjs.org/>
[Accessed 15 03 2020].

Firebase, 2020. *Firebase*. [Online]
Available at: <https://firebase.google.com/?hl=fr>

Gates, S., 2018. *Why Web Application Security is Important*. [Online]
Available at: <https://dyn.com/blog/why-web-application-security-is-important/>
[Accessed 10 03 2020].

Gibb, R., 2016. *What is a Web Application?*. [Online]
Available at: <https://blog.stackpath.com/web-application/>
[Accessed 10 03 2020].

Google Developers, 2019. *Instant Loading Web Apps with an Application Shell Architecture*. [Online]
Available at: <https://developers.google.com/web/updates/2015/11/app-shell>
[Accessed 10 03 2020].

Google Developers, 2020. *What are Progressive Web Apps?*. [Online]
Available at: <https://web.dev/progressive-web-apps/>
[Accessed 11 03 2020].

Hamilton, K. & Miles, R., 2006. *Learning UML 2.0*. s.l.:O'Reilly Media Inc..

Love, C., 2019. *Apple Safari Ships Service Worker and Progressive Web App (PWA) Support on iOS 11.3*. [Online]
Available at: <https://love2dev.com/blog/apple-ships-service-workers/>
[Accessed 11 03 2020].

Mayur Tanna, H. S., 2018. *Serverless Web Applications with React and Firebase: Develop real-time*. s.l.:Packt Publishing Ltd.

Miski, A., 2014. *Development of a Mobile Application Using the*, s.l.: s.n.

Mozilla Developer Network (MDN), 2019. *CacheStorage*. [Online]
Available at: <https://developer.mozilla.org/en-US/docs/Web/API/CacheStorage>
[Accessed 12 03 2020].

Nishadha, 2019. *Use Case Diagram Relationships Explained with Examples*. [Online].

Nnamdi, C., 2019. *Introduction to the Cache Storage (A New Browser Cache) PWA API*. [Online]
Available at: <https://blog.bitsrc.io/introduction-to-the-cache-storage-a-new-browser-cache-pwa-api-a5d7426a2456>
[Accessed 14 03 2020].

Office fédéral de la sécurité alimentaire et des affaires vétérinaires, 2019. *Identification*. [Online]
Available at: <https://www.blv.admin.ch/blv/fr/home/tiere/transport-und->

[handel/tierverkehrskontrolle/kennzeichnung.html](http://handel.tierverkehrskontrolle/kennzeichnung.html)

[Accessed 07 05 2020].

Official Statistics of Finland, 2016. <http://www.stat.fi/>. [Online]

Available at: https://www.stat.fi/til/ktutk/2016/ktutk_2016_2016-11-03_en.pdf

[Accessed 07 05 2020].

Ohio State University, 2009. *Microchips Result In Higher Rate Of Return Of Shelter Animals To Owners*. [Online]

Available at: <https://www.sciencedaily.com/releases/2009/10/091013185154.htm>

[Accessed 08 05 2020].

Osmani, A., 2016. *Progressive Web Apps with React.js: Part I — Introduction*. [Online]

Available at: <https://medium.com/@addyosmani/progressive-web-apps-with-react-js-part-i-introduction-50679aef2b12>

[Accessed 12 03 2020].

Rajput, M., 2019. *Best Frameworks for Building Progressive Web Apps*. [Online]

Available at: <https://www.mindinventory.com/blog/best-progressive-web-apps-frameworks/>

Ravichandran, A., 2018. *Props and State in React Native explained in Simple English*.

[Online]

Available at: <https://codeburst.io/props-and-state-in-react-native-explained-in-simple-english-8ea73b1d224e>

[Accessed 15 03 2020].

Rouse, M., 2019. *Web application (Web app)*. [Online]

Available at: <https://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app>

[Accessed 10 March 2020].

Russell, A., 2015. *Progressive Web Apps: Escaping Tabs Without Losing Our Soul*.

[Online]

Available at: <https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without-losing-our-soul/>

[Accessed 11 03 2020].

Sheppard, D., 2017. *Beginning Progressive Web App Development: Creating a Native App Experience on the Web*. s.l.:Apress.

Siavosh, K., 2019. *why use The PWA? what are Advantages and Disadvantages of PWA? part 2*. [Online]

Available at: <https://avengeriq.com/en/why-use-the-pwa-what-are-advantages-and-disadvantagesof-pwa-part-2/>

[Accessed 11 03 2020].

Société pour l'alimentation des animaux familiers, 2018.

<https://www.vhn.ch/fr/statistiques/animaux-familiers-en-suisse/>. [Online]

Available at: <https://www.vhn.ch/wp-content/uploads/2018/04/Statistik-VHN-Heimtierpopulation-2018.pdf>

[Accessed 07 05 2020].

Spence, I. & Bittner, K., 2002. *Use Case Modeling*. s.l.:Addison-Wesley Professional.

StackOverFlow, 2019. *Developer Survey Results 2019*. [Online]

Available at: <https://insights.stackoverflow.com/survey/2019/#most-loved-dreaded-and-wanted>

[Accessed 16 03 2020].

Stevens, E., 2018. *What Is The Difference Between A Mobile App And A Web App?*.

[Online]

Available at: <https://careerfoundry.com/en/blog/web-development/what-is-the-difference-between-a-mobile-app-and-a-web-app/>

[Accessed 10 03 2020].

The European Pet Food Industry - Fediaf, 2019. *EUROPEAN STATISTICS*. [Online]

Available at: <http://www.fediaf.org/who-we-are/european-statistics.html>

[Accessed 08 05 2020].

W3C, 2019. *Web App Manifest*. [Online]

Available at: <https://www.w3.org/TR/appmanifest/>

[Accessed 12 03 2020].

Xhoffray, X., 2016. *Lean Startup - Résumé et analyse du livre d'Eric Ries*, s.l.: 50minutes.fr.

Appendices

Appendix 1. MVP at the end of the first iteration

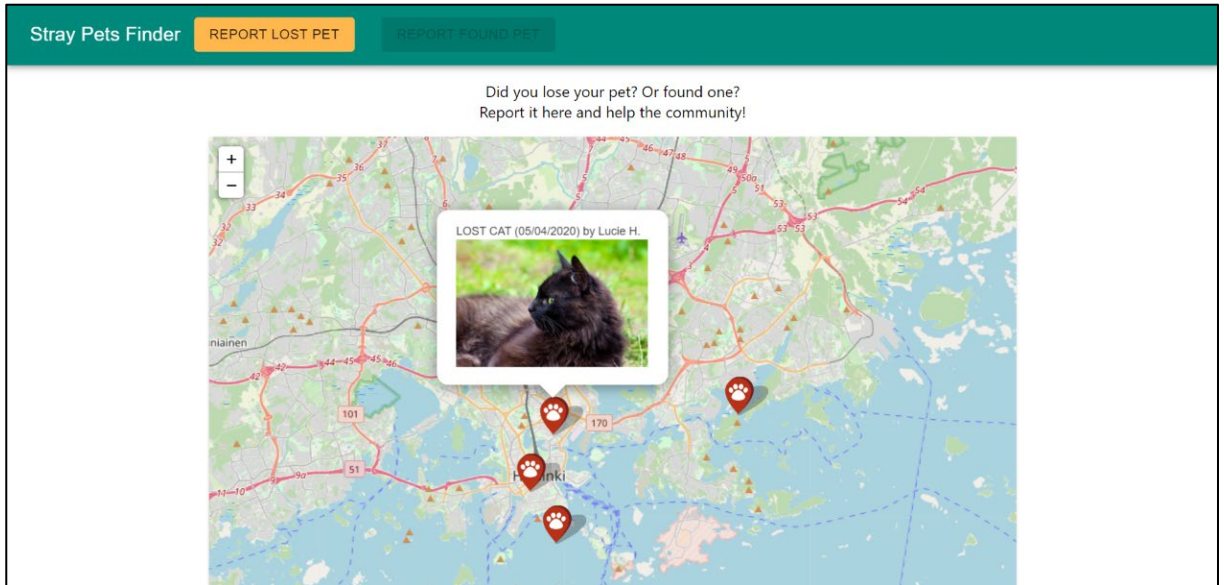


Figure 35: The main map showing all the reports

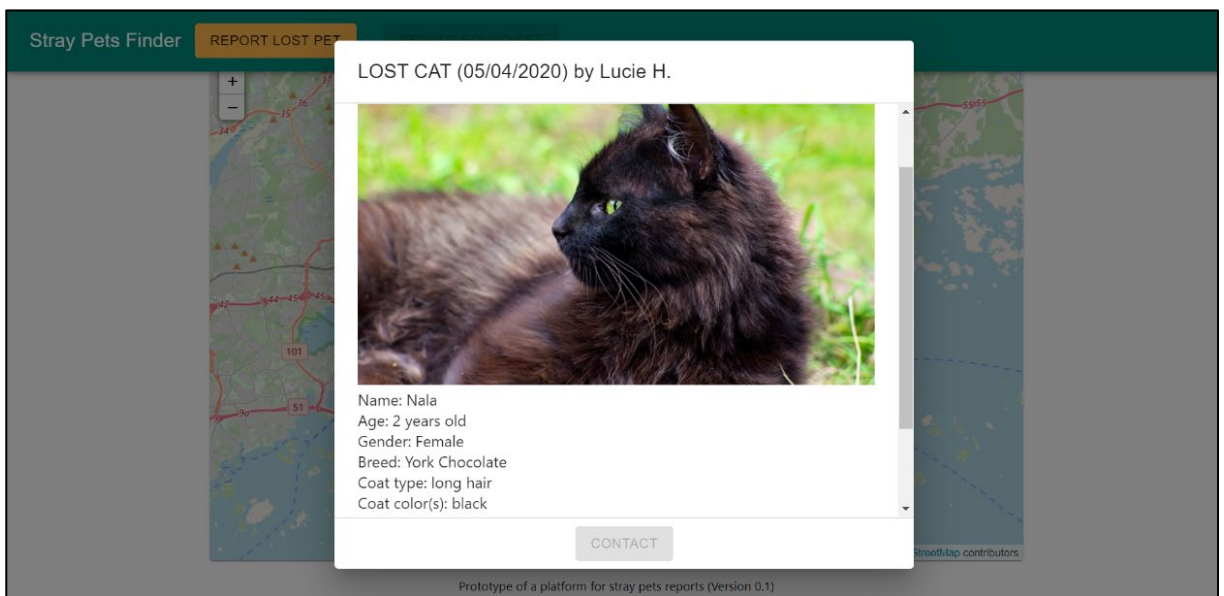


Figure 36: Display of the details of a report

Stray Pets Finder **REPORT LOST PET** REPORT FOUND PET

1

Personal information

Firstname *
Rita

Lastname *
Moreira

Email *

Email is required

BACK NEXT

2

Your animal's species

3


Information about your animal

Figure 38: The form with stepper and validation for lost pet reporting

Stray Pets Finder **REPORT LOST PET** REPORT FOUND PET

Event's circumstances

Please review your report before publishing



Your name: Rita Moreira
Your email: rita.moreira96@gmail.com
Name: Luke
Age: 3 years old
Gender: Female
Breed: Labrador Retriever
Coat type: long hair
Coat color(s): brown
Eyes color: blue
Pet size: medium
Particular information: none
Circstances: He ran away when I opened the car's door.
Location : Helsinki

BACK PUBLISH MY REPORT

Figure 37: The review of the lost pet report before publishing

Appendix 2. Questionnaire and results of the first iteration

1. Questionnaire

Personal information

1. What group do you belong to? (I'm a private individual, I'm an employee of an animals shelter, I'm an employee of a veterinary cabinet)
2. What age group are you in?
3. Do you have (had) any pets?
4. What species have you had?
5. What country do you live in?

Regarding lost pet ads

6. Have you ever lost one of your pets?
7. Did you report his disappearance by any means?
8. If yes, what mean(s) did you use?
9. What was your level of satisfaction with the means used?
10. Why?
11. Do you think there is room for improvement in this area?
12. How much do you pay attention to lost pet ads?
13. Have you ever shared lost pet ads?
14. If you answered yes, for what reason(s) did you share?
15. If you answered no, for what reason(s) did you not share?

Prototype testing

16. Generally speaking, did you encounter any problems during your experience?
17. What do you think of the map system? (what you liked, disliked, what is missing, doesn't work, etc.).
18. Do you think this system is useful for identifying lost/found animals?
19. What do you think of the form for adding a lost pet?
20. If you had to choose 3 features to develop among them, which ones would you choose? Choose 3 to rank by importance
21. Do you have other ideas for this platform?
22. If this platform were to be completed and put online, do you plan to use it?
23. Why?
24. Do you agree to answer a second questionnaire on this same application within 3-4 weeks?
25. If yes, enter your email address or any other information that will allow me to contact you.

2. Complete table of results :

<https://docs.google.com/spreadsheets/d/1YPZvVQHmflfOF-SCI7rUI9V5JhxfAoAwdu8bnVelzRs/edit?usp=sharing>

Appendix 3. MVP at the end of the second iteration

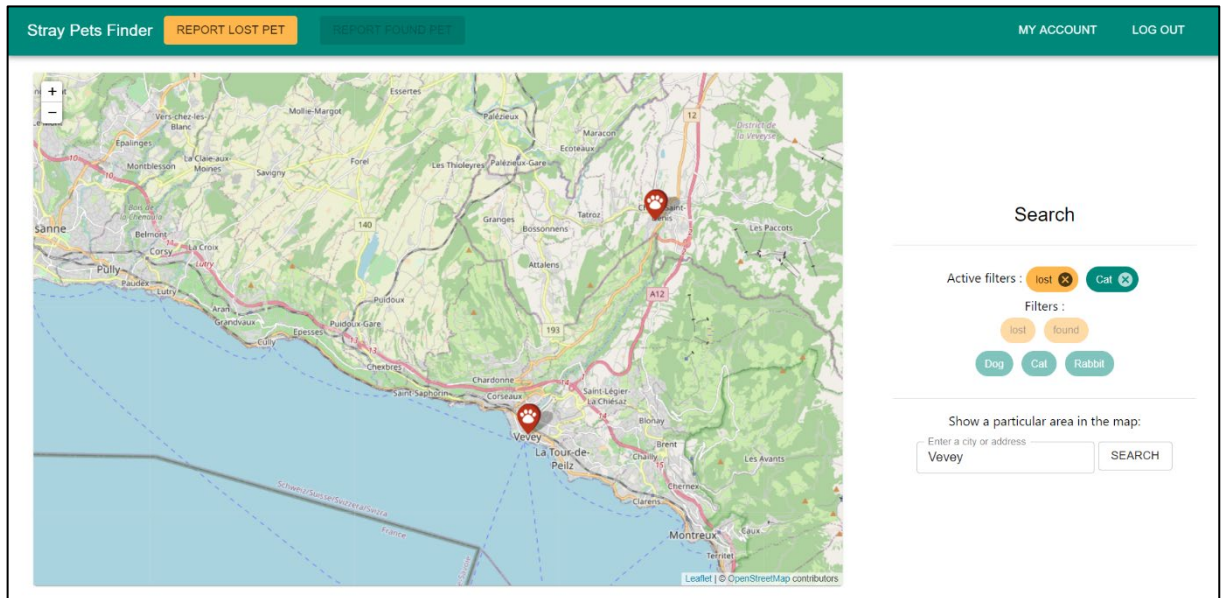


Figure 39: The main map with search filters on the right

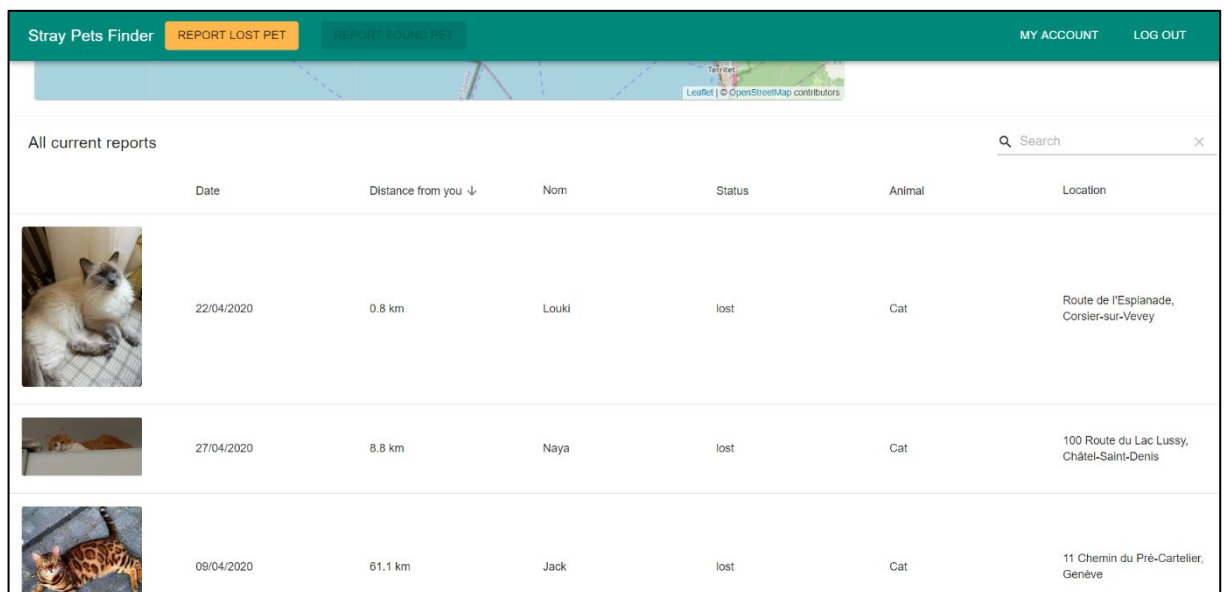


Figure 40: List of reports under the main map

Stray Pets Finder [REPORT LOST PET](#) [REPORT FOUND PET](#) [LOG IN](#)

Create your account

I agree to receive notifications when a new report in my area is posted. (NOT DEVELOPPED YET)

[SIGN UP](#)

[ALREADY HAVE AN ACCOUNT? LOGIN](#)

Figure 41: The form for the account creation

Stray Pets Finder [REPORT LOST PET](#) [REPORT FOUND PET](#) [MY ACCOUNT](#) [LOG OUT](#)

Personal Information

Name : Rita Moreira

Email : rita.moreira96@gmail.com

Address : Rte. Des Pléiades 74

City : Châtel-St-Denis, 1618

Your reports 🔍 Search ✕







Actions	Nom	Status	Animal	Location	Date
  	Louki	lost	Cat	Route de l'Esplanade, Corsier-sur-Vevey	22/04/2020
  	Jack	lost	Cat	11 Chemin du Pre-Carteller, Geneve	09/04/2020

Figure 42: The personal page with account information and reports management

Appendix 4. Questionnaire and results of the second iteration

1. Questionnaire

Personal information

1. What group are you in?
2. What age group are you in?
3. What country do you live in?

Prototype testing

4. Does the map show your position? (It is centered on your current location when you refresh the page)
5. What do you think of the location and map system?
6. What do you think of the list of reports below the map?
7. What do you think of the search system on the right side of the map?
8. What do you think of the form to create an account?
9. After creating the account, can you see a new tab "MY ACCOUNT" at the top right?
10. What do you think of the form for adding a lost pet?
11. After creating your ad, click on "MY ACCOUNT". Do you see your report listed?
12. If so, edit it and publish it. Are the changes visible?
13. Finally, delete your ad. Has it been removed from the list?
14. What do you think of the "MY ACCOUNT" page? (what you liked, disliked, what is missing, doesn't work, etc.).
15. If you had to choose 3 functionalities to develop among these, which ones would you choose? Choose 3 to rank by importance
16. Which of the following options below would you prefer to contact a report?
17. Would you be willing to receive notifications when a lost pet report is published near you?
18. If so, up to what range?
19. If not, for what reason(s)?
20. Do you have other ideas for this platform?
21. If this platform were to be finished and put online, do you plan to use it?
22. Why?
23. Finally, do you already know/use such a platform? If yes, please indicate its name.

2. Complete table of the results (English and French version are merged):

https://docs.google.com/spreadsheets/d/1_P2rwX3xeWGW1eZLZZ7ML9hnmbZqtCVUb-DWTTcklpYY/edit?usp=sharing

Appendix 5. Source code of the final implementation

<https://github.com/RtaMoreira/StrayPetsFinder>