Master thesis : Amanote - A modern note-taking application

Auteur : Fery, Adrien
Promoteur(s) : Boigelot, Bernard
Faculté : Faculté des Sciences appliquées
Diplôme : Master en sciences informatiques
Année académique : 2016-2017
URI/URL : http://hdl.handle.net/2268.2/2612

MASTER THESIS

Amanote: A modern note-taking application

Author:
Adrien FERY

Supervisor:
Prof. Bernard BOIGELOT

A thesis submitted in fulfillment of the requirements for the degree of Master in Computer Science by Adrien FERY

in the

Faculty of Applied Sciences
Department of Electrical Engineering & Computer Science

Academic Year 2016-2017

Declaration of Authorship

I, Adrien FERY, declare that this thesis titled, "Amanote: A modern note-taking application" and the work presented in it are my own. I confirm that:

- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Build your own dreams, or someone else will hire you to build theirs."

Farrah Gray

University of Liège

Abstract

Faculty of Applied Sciences

Department of Electrical Engineering & Computer Science

Master in Computer Science

Amanote: A modern note-taking application

by Adrien FERY

This thesis discusses the reasoning - in terms of technological choices, methodologies, and workflow - behind the construction of a scalable and modern application intended for a large number of users.

The work takes the development of Amanote, a note-taking application for slides and syllabuses, as the main thread. Firstly, it covers the development, the deployment, and the testing of the client-side, favouring low-cost and open-source technologies. It then explains the construction of a serverless architecture for the server-side, enabling high scalability and availability.

The results tend to show that the technologies used were well suited and in accordance with the requirements. Two years after the start of development, the application is still maintainable and acquired more than 20,000 users in 7 months.

Acknowledgements

I wish to express my gratitude to Professor Bernard BOIGELOT for his supervision as well as to Thomas FRANÇOIS, Patrick FERY, and Matthias HURDEBISE for their proofreading.

I sincerely thank Olivier DE WASSEIGE and Vincent KEUNEN for their precious advice in both Management and Computer Science.

I also want to thank Laurine ROSSION, Nathan SORET, Etienne GERKENS, Baudouin DE HEMPTINNE, Remi COMTE-OFFENBACH, François RIGO, Noé DELÉPINE, Yoric PETITFRÈRE, and all those not mentioned for their support in Testing, Marketing and Management.

Contents

List of Figures

Chapter 1

Introduction

1.1 Why Amanote

1.1.1 The growth of digital communication supports

Since the invention of PowerPoint in the 90s[1] and the democratisation of computers in the 2000s, digital presentation slides have widely replaced overhead projectors. Nowadays, several millions of slides are created daily[2] for educational and corporate presentations. The advantages of digital slides are numerous: they are modifiable, persistent, easily shareable worldwide, ecological, and economical.

In the educational domain at the university level, at the beginning of the semester, professors usually provide slides to students and use them as a visual aid during the course. Syllabuses are also increasingly distributed to students in digital format leaving them the choice to study on a computer or paper.

Nonetheless, this does not prevent the students from having to take notes since slides are only a visual aid and are not intended to be complete. Moreover, note-taking helps, in addition to store information, clarify and to structure the topic.

1.1.2 Handwritten versus computer-based note-taking

Although a study has shown that taking notes by hand may improve short-term memorisation and allows to synthesise better[3], other factors have also

[1] *Microsoft PowerPoint - Wikipedia*. `https://en.wikipedia.org/wiki/Microsoft_PowerPoint`. (Accessed on 05/25/2017).

[2] *PowerPoint usage and Marketshare - Infogram, charts & infographics*. `https://infogr.am/PowerPoint-usage-and-Marketshare`. (Accessed on 05/25/2017).

[3] Daniel M. Oppenheimer Pam A. Mueller. "The Pen Is Mightier Than the Keyboard". In: (2014)

to be taken into consideration when choosing between paper or digital note-taking. This section compares handwritten versus computer-based note-taking.

The advantages of handwritten note-taking:

Memorisation
As mentioned above, information can be better memorised.[3].

Relevancy
As taking notes by hand ordinarily does not allow to be fast enough to record everything, handwritten notes are usually more selective and thus more synthesised.[3]

Flexibility
Handwriting usually brings more flexibility, especially for complex schema, formulas, or charts.

Fewer distractions
For some students, taking notes by hand can be less distractive than on a computer with an internet connection.

The advantages of computer-based note-taking:

Rapidity
For many students, it is faster to type and to structure text on a computer than with a pen.

Modifiability
One major advantage of computer-based note-taking is that it allows editing, restructuring, and reorganising notes easily afterwards.

Searchability
It is unquestionably faster to search in digital notes.

Space and order
All the notes can fit on a laptop, while handwritten note-taking produces several stacks of paper that have to be stored and carried for each lesson.

Shareability
Digital notes can be shared easily with others.

Persistence
It is possible to make backups which avoid notes to be lost or stolen.

Readability
Digital notes are cleaner and easier to read.

Economical
Computer-based note-taking is cheaper compared to the cost of printing all the slides and writing on them with a pen.

Ecological
Finally, it is also more ecological (if the student already has a computer).

Given the advantages of the two approaches, handwritten note-taking may seem more efficient for courses which do not require reworking the notes later and which contain few cross-references. However, computer-based note-taking has many advantages than can be more beneficial in the long term and for courses that require more work at home.

1.1.3 Note-taking solutions before Amanote

Text editor

A traditional way to take notes on a computer is using a text editor such as Microsoft Word or Open Office. Theses editors usually offer an extensive range of functionalities. However, they did not allow to link notes to slides, which makes them less adequate when slides are used as a course support.

In fact, being able to link notes to the source can be very helpful. It allows writing faster since the context does not have to be explained. It improves comprehension by gathering all information in the same place. And it allows better organisation of the information.

PDF Viewer

An alternative is to use a PDF viewer such as Preview or Adobe Reader. In fact, they allow viewing slides while adding annotations. This is very useful when one needs to write only a few comments. However, to make a real note taking, it quickly becomes disorganised, and it is not possible to format the text in a similar fashion to a text editor, neither to include mathematical formulas for instance.

1.2 The birth of Amanote

Given the lack of adequate solution and the frequent usage of slides in the context of education, it was essential to create a note-taking application appropriated to this technology. It is the reason why, the Amanote project was born in September 2015 with the main goal of allowing to follow courses and presentations while being able to take clear and structured notes at the same time, and to link them properly to the right slides.

This section gives an overview of what was originally expected, starting first with the basic features and then with the more advanced features. The results obtained are presented at the end of this work in Section 4.2.

1.2.1 Basic features

Basically, Amanote aims to combine a slide viewer with a rich text editor (Figure 1.1). As slides are often provided in PDF format, the slide viewer can be extended as a PDF viewer, thus allowing to take notes with any kind of PDF documents.

Each page of the PDF document has to be distinctly linkable with the text editor, enabling to write different notes for each of them. Naturally, the corresponding notes must appear when the pages of the document change. Thereby, when a page change occurs, the notes for the current page are stored, and the text editor is refreshed with the notes corresponding to the next page, if any were already produced, and blank otherwise.

Regarding the text editor, it should provide all the features required to structure and allow a rich note-taking. It should thus be possible to format the text, to include mathematical formulas, drawings, charts, images, and so on. Furthermore, it should be possible to refer to a specific area of the slides directly in the notes.

Finally, it must also be possible to save the notes to edit them later, as well as to print the PDF with the corresponding notes below each page.

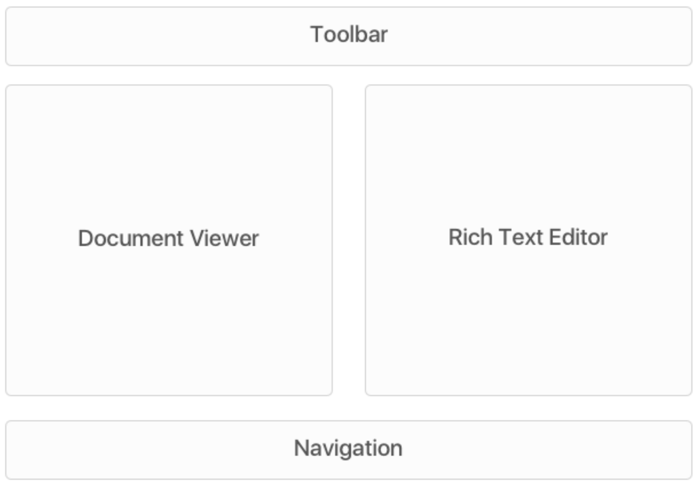


FIGURE 1.1: Basic schema of the application.

1.2.2 Advanced features

Amanote is intended to be free and accessible to all students. However, for development to be sustainable thereafter as well as to offset the costs of web services, the app should, somehow, generate profits. Thus, Premium accounts[4] should be developed to allow the users to access to more advanced features. Those are detailed in the following sections.

Cloud storage

As it is critical for a student not to lose his notes, a cloud storage system allowing to automatically backup the notes online and to synchronise them between all his devices should be available. This feature can also be a good way to share and collaborate on notes with other users.

Audio recorder

It is sometimes difficult to take note of all useful explanations during a course. In this respect, having an audio recorder synchronised with the slides allowing to matches the audio to the right slides, as illustrated in Figure 1.2, can be helpful. Students would no longer have to listen to the whole recording, but could instead focus on the explanations related to the slides they did not understand, which would save them considerable time.

[4]Note that there is no restriction on the usage of the basic features without Premium subscription.

Note: As with any recorder system, the students must, of course, have to get an authorisation before recording the speaker.

FIGURE 1.2: Audio recorder synchronised with the slides.

Statistics about the slides

All slides do not have the same importance: some may be skipped whereas others require long explanations. To reflect this, the app should record the time spent on each slide during the class and indicate it afterwards to the user.

1.3 Work overview

The goal of this work is to explain the reasoning behind the creation of modern and scalable application intended for a broad public using Amanote as the main thread. As we will see in chapter 2 and 3, respectively dedicated to the client-side and server-side, this includes the choices and motivations for the use of different technologies as well as the description of the methodologies applied to the development and the delivery of the application.

Chapter 3 reports the results obtained regarding both the developed application and the feedback received from users. Finally, chapter 4 concludes the work and considers some improvements of Amanote.

Note: The terms "front end" and "client-side" as well as "back end" and "server-side" will be used interchangeably in the rest of this work.

Chapter 2

Client-side

2.1 Introduction

As previously stated, the goal of this chapter is not to explain in detail how Amanote's client-side was designed but rather to give a general overview of front end development in terms of technological choices, workflow, and methodologies used to build a scalable and cross-platform application.

2.2 Requirements

The client-side is the most important part of Amanote. Its goal is, primarily, to allow students to take comprehensive notes, in a fast and structured way, while linking them to their course support (i.e., PDF documents).

Targeted platforms

As it is faster to type on a computer rather than on a mobile device and that, according to Pearson, 73% of college students already use a laptop during typical school days, and 24% use the tablets[1], the application should at least be **compatible with Windows and macOS**. Indeed, as it can be seen from Figure 2.1, Windows and macOS are by far the most popular operating systems.

Platforms	**Market share**
Windows	84.22%
macOS	11.63%
Others	2.48%
Linux	1.67%

FIGURE 2.1: Desktop OS market share.
StatCounter Global Stats. *Desktop Operating System Market Share Worldwide*. `http://gs.statcounter.com`. Apr. 2017

[1]Pearson, ed. *Student Mobile Device Survey 2015*. Pearson, 2015.

Given the growing use of tablets by college students[2], it should also be possible to target iOS and Android tablets later, if needed, without having to re-engineer all the code.

User-centered design

As the application is intended for a wide and heterogeneous public, it has to be especially **easy to use** and possess an **attractive and clean user interface**.

The **interactions of the users with the application have to be traceable** for analytic purposes. As it will be seen in section 3.5, this will allow getting direct feedback on how the users use the app, and it will help to better understand what has to be improved to increase retention.

Continuous delivery

Amanote is meant to evolve quickly, so it should be **easily maintainable on the long term**. The application should then be modular with smooth coupling between modules, which will make easy modifying some parts of the code without having to change the others. Furthermore, modularity facilitates collaboration with other developers.

In the same spirit, the front end has to be able to **update automatically and silently** to allow the users to be always up to date without bothering them by having to install each update which can come in short intervals of time. Moreover, it reduces the number of versions to deal with. But, on the other hand, it requires more vigilance during the tests phase to avoid introducing a buggy version to all users.

Cost

The cost of development has to be as low as possible while using the most appropriate technologies. Paid libraries can be used if they are open source and modifiable or if they are used for unessential features. The danger is to be stuck with a discontinued or buggy technology that cannot be modified.

[2] +11% from 2013 to 2015; (Pearson, ed. *Student Mobile Device Survey 2015*. Pearson, 2015)

Copyright

The application has to be **proprietary, protected by copyright, and closed source for commercial use**. It is, therefore, necessary to choose the technologies carefully in order to respect the terms of the licences.

2.3 Technology

When developing an application, one has usually to choose between a native, cross-platform or hybrid development. In this section, we will first compare native and cross-platform solutions, and then we will discuss the choice of a cross-platform framework.

Native. By definition, a code is said to be native when its compiled version is a code composed of instructions directly recognised by a processor. In the case of an application, it is said native when it is developed for a specific platform, and able to interact directly with the operating system interface. These are typically applications developed in C# or VB.NET for Windows and Objective-C or Swift for macOS.

Cross-platform. The cross-platform applications, on the other hand, are designed to run on several platforms with the same source code. These are interpreted languages, such as JavaScript or Java[3] for instance, or systems that convert the code to make it compatible with each targeted platform, such as Haxe[4].

Hybrid. Hybrid applications are cross-platform applications but with the particularity of being hosted inside a native application and executed in a Web view using Web technologies. We will not make a difference between cross-platform and hybrid in the rest of this work since hybrid applications are just a kind of cross-platform app and these terms are commonly used interchangeably.

[3]When compiled into Bytecode and interpreted by the Java virtual machine (JVM).
[4]We will see how Haxe works later in Section 2.3.3.

2.3.1 Native versus cross-platform comparison

Native

The main benefits of a native application are:

- Execution may be faster, and the user interface more responsive.
- Possibility to use all the native features of the platform directly compared to cross-platform frameworks that provide only a subset.
- User interface may be more user-friendly as it uses the standard components of the operating system.

The main disadvantages are:

- Depending on the language used, the whole code or at least some parts of it have to be rewritten for each platform; this increases the development time as well as complexity, and therefore the cost.
- It is harder to maintain since it requires a different version of the code for each platform.
- It more difficult to find developers that master the programming languages related to each platform.

Cross-platform

The main benefits of a cross-platform application are:

- It saves a lot of time since the source code is directly compatible with other platforms.
- It is much easier to maintain since there is only one version to manage.
- The result is the same whatever the platform.

The main disadvantages are:

- It is less responsive than a native app.
- Depending on the technology, it can be too slow for graphically or CPU intensive applications. However, if only some secondary tasks are intensive, it's generally possible to implement native plugins that perform the intensive tasks.
- Frameworks generally do not provide all the native features of the operating system (such as the possibility to use the Touch Bar on the new Macbook Pro for instance).
- It may take some time before the framework is compatible with new OS versions.

In view of the pros and cons of native compared to cross-platform development, as Amanote targets several platforms and that the front end is not intended to be intensive or to use special platform specific features, a cross-platform development is more advantageous from a cost, time, and maintainability point of view.

2.3.2 Selection criteria for a cross-platform framework

Open source with a large community

An important criterion is to choose a free open-source framework and with a large community. This will reduce the development costs and the risks that the framework will be discontinued later. Moreover, the fact of being open source, according to the licence, will allow the addition of missing features oneself if needed. Finally, having a large community also makes it easier to find plugins, tools, resources, or help.

The popularity of technologies can be estimated with different tools such as the number of Web searches (Google Trends), the number of dependent repositories, the number of followers on GitHub, or with survey results.

One thing to look at carefully when choosing an open-source framework or library is the licence that goes with it. In the case of Amanote, as the front end has to be closed source the licence must be non-copyleft and authorise commercial distribution. It is the case of the MIT License, LGPL, or Apache License but not of the GNU GPL for instance.

A popular programming language

A popular programming language can greatly facilitate the development of the application by allowing to find reusable code faster and in greater quantity. In addition, when the team grows up, it will be easier to find developers who master this programming language.

Fulfilled requirements

The requirements have, of course, to be fulfillable with the chosen framework. If the application needs to access specific native features, these should be available in the framework, and if not, it should be possible to add them without too much work.

Quality of existing applications and documentation

The quality of applications developed with the framework as well as the quality of the documentation can also help to make a choice. Indeed, a comprehensive documentation allows a faster development and avoids wasting time to understand how some features of the framework work.

2.3.3 Description of some popular cross-platform frameworks

In this section, we will review some popular open-source cross-platform desktop frameworks and technologies that can be used to develop the front end of Amanote.

JavaFX

Name	JavaFX
Developer	Sun Microsystems (Oracle Corporation)
Technologies	Java, CSS
License	Oracle Binary Code License, open source included in OpenJDK which is licensed under GPL v2 with Classpath Exception
Platforms	Windows Vista, Windows 7, Windows 8, Windows 10, macOS and Linux
GitHub	9,594 JavaFX public repositories, 682,000 Java public repositories (May 2017)
Example	SkedPal, Atlas Trader

FIGURE 2.2: JavaFX characteristics.

Java is an object-oriented programming language whose reputation is firmly established and which has the particularity of being portable on several platforms including Windows, macOS, and Linux. It is one of the oldest approaches to cross-platform applications.

JavaFX is a set of Java packages included with the standard JDK 8 that facilitates the development of graphical user interfaces, thanks to Cascading Style Sheets, as well as the deployment of Java cross-platform desktop applications.

The Java community is pretty broad, and with more than 682,000 Java public repositories on GitHub we can expect that many libraries can be reused to save time during the development. Successful software has been developed in Java (without necessarily using JavaFX), as is the case, for example, of IntelliJ or Eclipse.

The pros of JavaFX

- Java is a popular programming language with a large community.
- Cascading Style Sheets can be used for the GUI which is a big advantage compared to implement it in the code like with Swing.
- Possibility to embed Web pages within the application.

- The application can benefit from the hardware-accelerated graphics pipeline (Prism).
- It allows deploying a self-contained application package embedding the application resources as well as the Java runtimes.
- JavaFXPorts allows to port JavaFX application to iOS and Android.

The cons of JavaFX

- It is only partially open source; The JavaFX UI controls is part of the OpenJDK project licensed under GPL v2 with Classpath Exception and the runtime is licensed under Oracle Binary Code License (as Java) which prevents any modification and indicates that some Commercial Features may be chargeable.
- The JavaFX CSS parser is not a fully compliant CSS parser and is less powerful than Web CSS.

Haxe

Name	Haxe
Developer	Nicolas Cannasse
Technologies	Haxe
License	Haxe Licenses: GNU GPL and Licence MIT
Platforms	Windows, Linux, macOS, Android and iOS
GitHub	2,000 Haxe public repositories, 1,879 followers (May 2017)
Example	CastleDB

FIGURE 2.3: Haxe characteristics.

Haxe is an open-source toolkit under MIT Licence and developed originally since 2005 by an independent developer. The platforms are compatible with Windows, macOS, Linux but also Android and iOS.

It consists of a high-level object-oriented programming language influenced by Java, a cross-compiler that translates code written in Haxe into native code

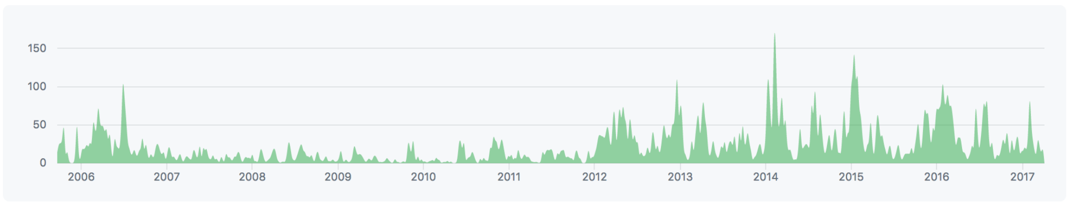


FIGURE 2.4: Contributions to development of Haxe on GitHub (excluding merge commits)

for each targeted platform, and a library implementing access to APIs for different platforms such as access to files or native dialogue boxes.

Although the community is small, it is active. However, there is no very popular desktop application that has been developed with Haxe until now. On its official website, Haxe lists only one desktop app called CastleDB. In contrast, several games are listed.[5]

The pros of Haxe

- Compatibility with Android and iOS.
- Better performance since the source code is compiled to native code.

The cons of Haxe

- Small community.
- Haxe is not a standard programming language.
- It is not yet widely used to develop desktop front end apps.
- There are not many resources (only 2,000 public repositories on GitHub).

[5] https://haxe.org/use-cases/

Electron

Name	Electron
Developer	Github
Technologies	JavaScript, HTML, CSS, Node.js
License	Licence MIT, open source
Platforms	Windows 7, Windows 8, Windows 10, macOS and Linux
GitHub	17,518 Electron public repositories, 715,000 JavaScript public repositories, 45,740 followers (May 2017)
Example	Atom, Slack, Visual Studio, Wordpress Desktop, MeisterTask, More examples are listed at https://electron.atom.io/apps/

FIGURE 2.5: Electron characteristics.

Electron is an open-source Node.js framework licensed under MIT licence and developed since 2013 by Github. It allows developing applications for Windows, macOS and Linux platforms using Web technologies (JavaScript, HTML and CSS). Nevertheless, it can be well integrated into the operating system and allows, for example, to use native menus, notifications or dialogues. Furthermore, the module C / C ++ Addons of Node.js allows creating some part in native code for intensive tasks.

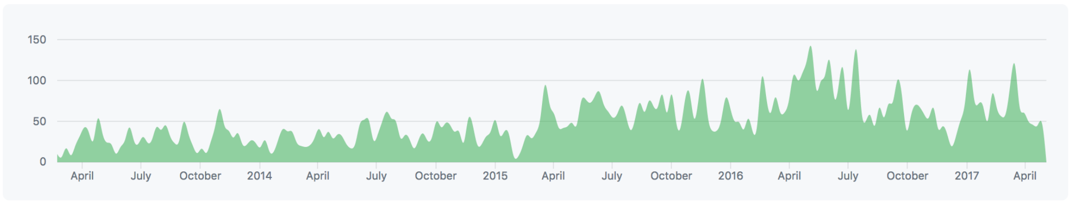


FIGURE 2.6: Contributions to development of Electron on GitHub (excluding merge commits)

With more than 45,740 followers on GitHub, the community is large and active. Electron is developed by a reputable company and has been used to develop popular modern applications such as Atom from GitHub, Visual Studio from Microsft, Wordpress Desktop or Slack.

The pros of Electron

- Many popular modern applications have been developed with this framework in recent years[6] and several are open source[7]. That allows looking at the source code to understand the way certain tasks are managed.
- Portable without too much work to mobile devices using a mobile framework that is based on the same mechanism such as Cordova, for instance.
- Web technologies are simple and with plenty of reusable code.
- According to Stack Overflow's 2016 Developer Survey Results, more developers use JavaScript than any other programming language.[8]
- HTML & CSS frameworks such as Bootstrap can be used to easily create an attractive user interface.
- Electron applications are simple to deploy and include an automatic update system provided directly by the framework.

The cons of Electron

- Execution of JavaScript is slower than compiled languages.
- Native addons have to be coded in C/C++, which are more complicated languages.

[6]More than 335 apps are listed on the Electron's official Website.
[7]134 are listed as open source.
[8]*Stack Overflow Developer Survey 2016 Results*. `https://insights.stackoverflow.com/survey/2016`.

Mono

Name	Mono
Developers	Xamarin and Microsoft
Technologies	C#, .NET
License	Class libraries licensed under Licence MIT and the runtime under GNU LGPL, open source
Platforms	Windows 7, Windows 8, Windows 10, macOS, Linux, Android and iOS
GitHub	3,000 Mono public repositories, 178,000 C# public repositories, 5,514 followers (May 2017)
Example	Unity3d, Sims3

FIGURE 2.7: Mono characteristics.

Mono is a .NET development platform developed since 2004 and acquired by Xamarin in 2011. Its functioning is similar to Java in the sense that the source code is compiled into an intermediate language (bytecode) and converted just-in-time into machine instructions by the Microsoft's .NET virtual machine called the Common Language Runtime (CLR).

The pros of Mono

- Compatible with a large range of platforms.
- Comparable to Java.

The cons of Mono

- Mono seems to be used more for games than for applications.
- GUI is more complex to implement since it has to be coded in C# with a GUI toolkit.

Others

Of course, other desktop cross-platform frameworks such as Qt or Kivy exists, but they are less appropriate for Amanote's client-side. In fact, Qt is a

C++ library rather used in embedded devices or intensive applications, while Kivy is highly multi-touch oriented which is not the case of Amanote.

2.3.4 Motivation for choosing Electron

At the end of this review step, it appears that the choice has to be made mainly between Electron and JavaFX. Indeed, Haxe has an interesting functioning, but it is a technology that remains a little immature, and that has not been much used yet for the development of desktop applications. Its community is small and its functioning complex, so there are more risks of being discontinued in the future. Moreover, as it is a particular technology, the code would be hardly portable to another framework. Haxe is therefore not suitable for the development of Amanote.

Regarding Mono, its main advantage is that it targets a greater number of platforms. However, the graphical user interface is more complicated to implement since it has to be coded with a GUI toolkit like GTK# what can be noticed when looking at the showcase of applications developed with Mono.

Electron versus JavaFX

First, as the front end is not intended to be graphically or CPU intensive, Electron is well suited for this kind of application and the better performances of Java are therefore not a real advantage in this case.

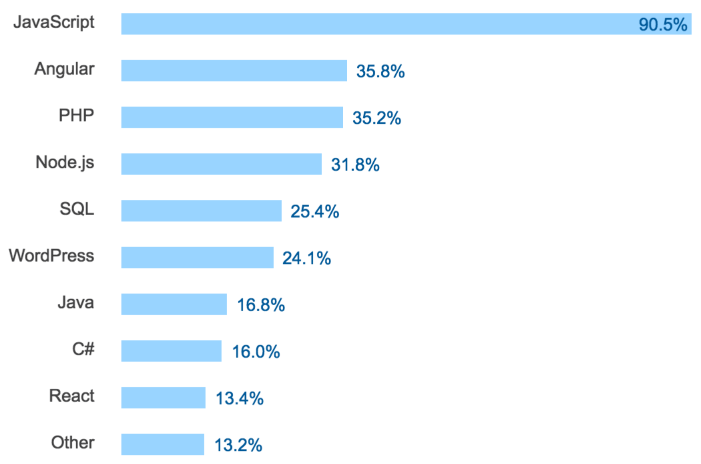


FIGURE 2.8: Front end most popular technologies according to Stack Overflow's 2016 Developer Survey Results.

Then, as shown in Figure 2.8, JavaScript is considered as the most commonly used programming language in front end technologies. We can then expect that there are more JavaScript's resources, helps, plugins, and developers. Moreover, npm (the package manager for JavaScript) is the world's largest software registry[9,10] allowing to quickly find many modules and reusable code.

One downside of JavaScript compared to Java is the maintainability. In fact, JavaScript allows more flexibility, as it is untyped for instance, and thus one must be careful not to use bad design. However, TypeScript[11] and JavaScript frameworks solve many of the problems of the code quality linked to JavaScript and allow to significantly increase its maintainability.

The fact that Electron uses Web technologies and the simplicity of its functioning (Web browser + Node.js) allow to not be too strongly bound to a particular technology. Thus, if one day Electron would be discontinued, most of the code could be reused with another framework using the same technologies, which is the case of nwjs[12] for instance. In the case of JavaFX, a large part of the user interface would have to be rewritten.

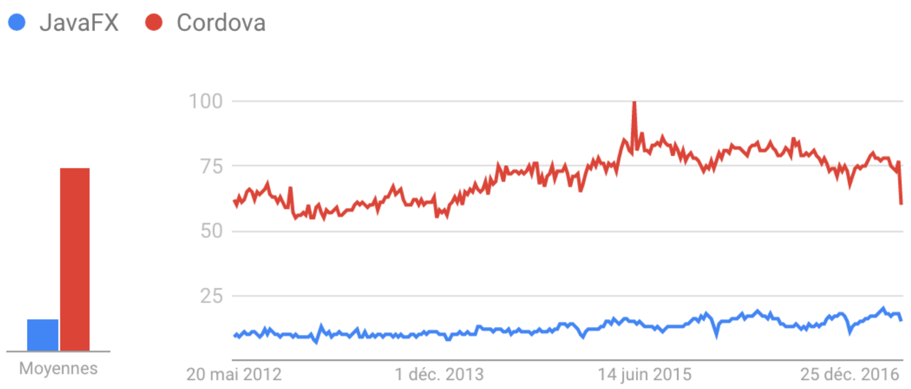


FIGURE 2.9: JavaFX versus Cordova on Google Trends.

Both can be ported to mobile platforms without too much work. However, as it can be seen in Figure 2.9, Cordova, a cross-platform framework with a similar functioning than Electron but for mobile, is much more popular than JavaFX.

[9]*npm now the largest module repository*. http://alexandros.resin.io/npm-now-the-largest-module-repository/.

[10]*npm*. https://www.npmjs.com/.

[11]TypeScript is a popular superset of JavaScript that adds static typing and class-based object-oriented programming.

[12]nwjs was not introduced before because it is in beta stage

Documentation of Electron and JavaFX are both comprehensive. However, contrary to JavaFX, the Electron website cannot be easily searched in, but that can be solved using a documentation browser such as Dash.

Regarding the licences, Electron is fully open source and licensed under the MIT License which is one of the most permissive licences while JavaFX is bound to Oracle.

Finally, Electron is used by big modern companies such as Facebook, Slack or Wordpress. The quality of application made with Electron is perfect, and the web technologies allow to achieve an appealing user interface more easily than with JavaFX thanks to CSS & HTML framework.

Remark: Electron only supports Windows 7 and higher. This is not a problem for an application like Amanote, but it is necessary to be aware that there are companies still using earlier versions (7% of the market share[13]), and this can be problematic for some B2B applications.

2.3.5 More detail about Electron

As it can be seen in Figure 2.10, Electron works by combining a variant of Node.js, an open-source cross-platform JavaScript run-time environment, and the Chromium rendering library, an open-source web browser which is used as the basis of the popular Google Chrome browser. It also includes a per-platform native API for auto-update, dialogue, notifications, and so on.

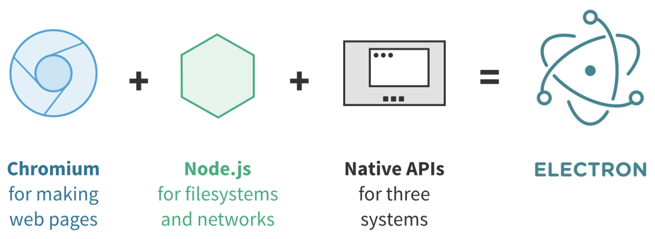


FIGURE 2.10: Electron's main components.
Essential Electron. http://jlord.us/essential-electron/.

[13] *Desktop windows versions market share Worldwide | StatCounter Global Stats*. http://gs.statcounter.com/os-version-market-share/windows/desktop/worldwide.

Node.js is slightly modified to be adapted for desktop applications instead of web servers, but its API remains the same. It is used mainly for low-level tasks such as file system access, networks, cryptography, process handling, etc. Its *C/C ++ Addon* and *Child process* API allow executing native code or starting another software if necessary.

Electron handles multiple processes. The first process executed is called the *MainProcess* and does not contain any graphical user interface. It is the entry point of the application from which different *RenderProcess* can be created via *BrowserWindow* and it is responsible for handling them as well as intensive or blocking tasks that should not be run in the *RenderProcess*. Figure 2.11 summarises the mechanism of the different processes.

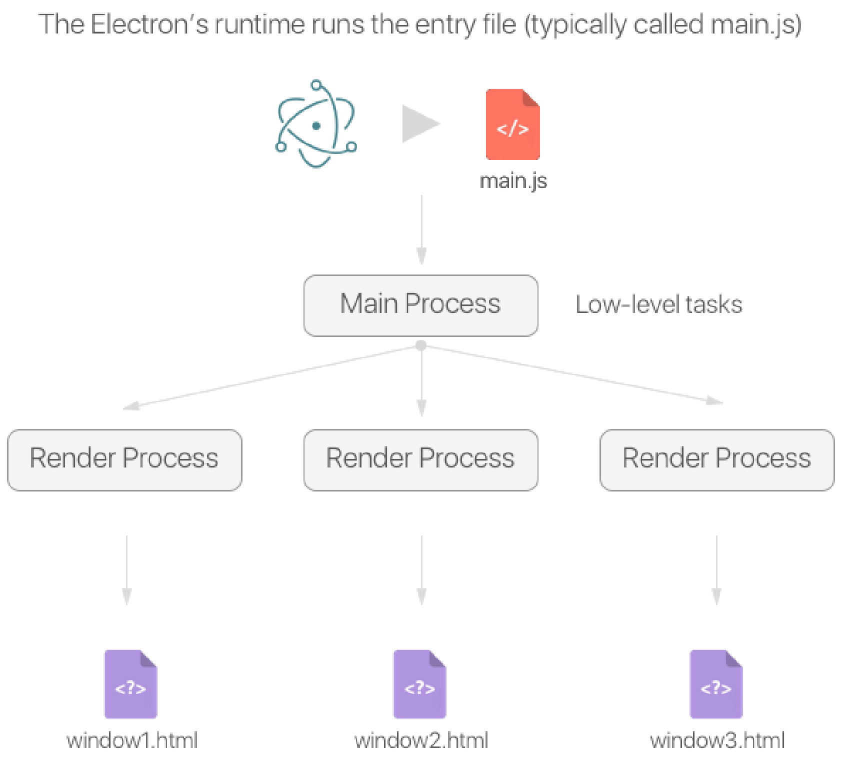


FIGURE 2.11: Electron's processes mechanism.

RenderProcess are responsible for rendering the GUI using Chromium. They are isolated from each other, but they can communicate with the *MainProcess* and vice versa via Electron's *ipcRender*, *ipcMain*, or *remote* modules. Electron does not include all the Chromium code, but only its rendering library.

Compared to a web page in a traditional browser that is in a sandbox, Node.js is also accessible directly on the *RenderProcess* side.

Basic Electron app example

Here is a basic example of an Electron app displaying *Hello World* and the node version. It is a simplified version of the code provided in Electron's Quick Start guide[14].

```
const {app, BrowserWindow} = require('electron');

const path = require('path');
const url  = require('url');

let mainWindow = null;

// Create the main window when the app is ready
// (i.e., Electron has finished initializing).
app.on('ready', function ()
{
  mainWindow = new BrowserWindow({width: 800, height: 600});

  // Load the index.html using the platform specific separator.
  mainWindow.loadURL(url.format({
    pathname: path.join(__dirname, 'index.html'),
    protocol: 'file:',
    slashes: true
  }));
});

// Close the app when all the window are closed.
app.on('window-all-closed', () => {
  app.quit();
});
```

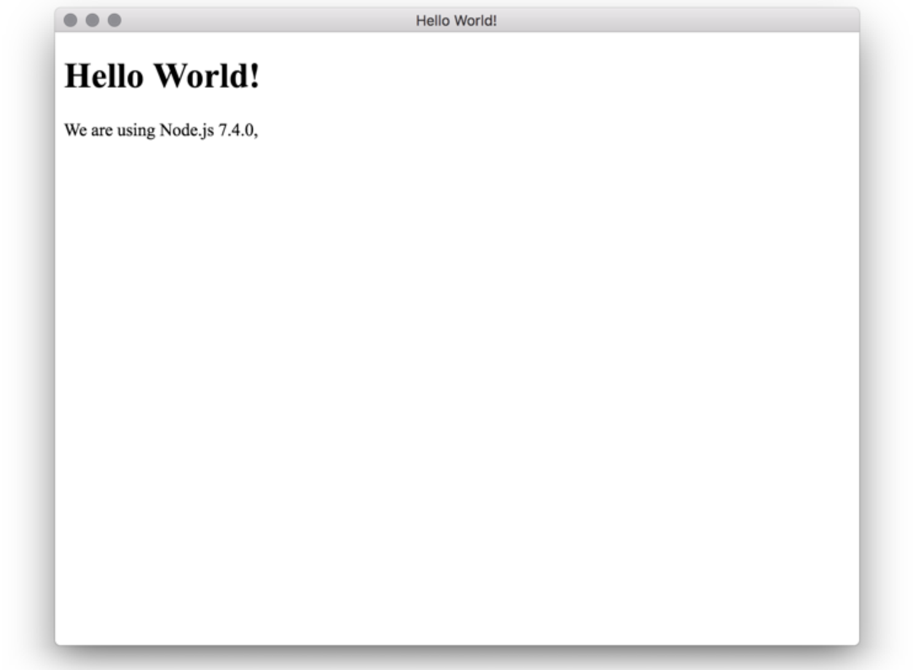
LISTING 2.1: main.js

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Hello World!</title>
  </head>
  <body>
    <h1>Hello World!</h1>
```

[14] *Quick Start | Electron*. https://electron.atom.io/docs/tutorial/quick-start/.

```
    <!-- All of the Node.js APIs are available in this renderer
       process. -->
    We are using Node.js <script>document.write(process.versions.
       node)</script>,
  </body>

  <script>
    // You can also require other files to run in this process
    require('./renderer.js')
  </script>
</html>
```

LISTING 2.2: index.html



Hello World!

We are using Node.js 7.4.0,

FIGURE 2.12: Electron Hello World app screen.

Auto-updater

Electron makes it easy to implement an automatic update system thanks to its *autoUpdater* module and the Squirrel framework.

Squirrel is a set of tools that manages the installation and updating of desktop applications. For the Windows and macOS platforms, there are Squirrel.Windows and Squirrel.Mac.

It just requires a back-end endpoint URL to be registered to check if an update is available and to update the application. The endpoint takes two arguments, the platform name, and the architecture and returns an URL indicating where the update, if any, can be downloaded.

Remark: Apple prevents automatic updates through the *autoUpdater* module when the application is published on the App Store.

2.3.6 JavaScript framework

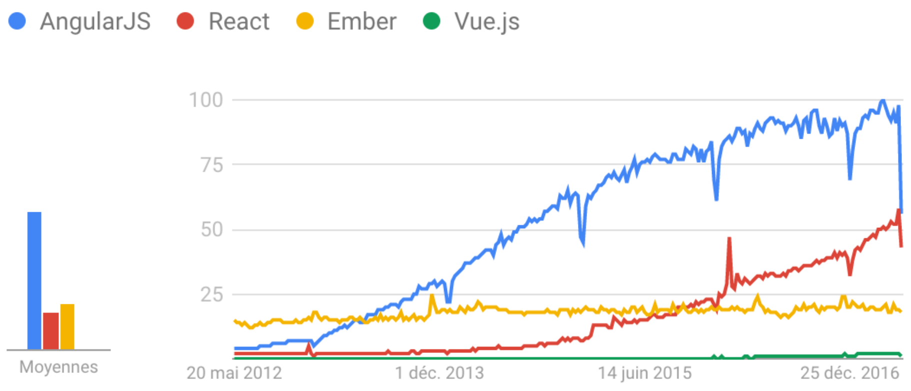


FIGURE 2.13: JavaScript Frameworks on Google Trends.

A JavaScript framework is also recommended in order to structure and facilitate the development. They are many possibilities: Ember, Vue.js, AngularJS from Google, or ReactJS from Facebook. However, the two most popular are AngularJS and ReactJS. They all have advantages and disadvantages, but AngularJS is richer for this kind of application. ReactJS is lighter and maybe more suited for Web apps requiring better Web performances.

2.4 Working environment

2.4.1 Directory structure

It is important to set up a development directory structure that allows to find and move from one file to another as quickly as possible even when the number of files increases dramatically.

That is why it seems interesting to adopt a different structure than the popular "type-based" approach that can be found in many projects. The latter consists of dividing the files into folders corresponding to their type: the controllers go in a *controllers* folder, the views in a *views* folder and so on, as illustrated in Figure 2.14.

```
app/
├── controllers/
│   ├── homeController.js
│   └── profileController.js
└── views/
    ├── homeView.html
    └── profileView.html
```

FIGURE 2.14: "Type-based" directory structure.

This approach works well with a limited number of files but when the number grows up, it ends up with lots of files in these folders, and it becomes difficult to quickly find a given file. In this case, a more scalable approach is to use a "component-based" structure in which files are divided into folders by feature. In this architecture, the controllers and the views corresponding to a specific feature are located in the same folder. This structure is illustrated in Figure 2.15.

```
app/
├── home/
│   ├── controller.js
│   └── view.html
└── profile/
    ├── controller.js
    └── view.html
```

FIGURE 2.15: "Component-based" directory structure.

This type of structure allows working on a feature without switching between folders. Moreover, if a feature has to be renamed or deleted, it makes simpler to rename/delete a single folder instead of renaming/deleting each file in different folders.

2.4.2 Automated workflow

Some tasks can be automated with automation tools such as Gulp or Grunt. The community of these tools already provides a lot of plugins that allow to easily implement scripts that run silently in the background in order to, for example:

- Compile files (such as SASS [15] files to CSS files).
- Automatically refresh the page when there is a change in the code.
- Scan the source code to detect errors.
- Replace patterns in the source code.
- Perform automatic tests.
- Commit changes every hour to a specific branch.
- And so on.

In the long term, the implementation of these scripts allows saving considerable time.

2.4.3 Version control system

Version control system (VCS) are tools that keep track of the source code modifications under all its different versions. It is essential for any significant project to set up a version control system for both front end and back end. The benefits are numerous.

First, it allows to have a backup and to keep a history of the changes in the source code. This way, it is possible to easily return to an earlier version in order to restore some part of the code, in case of a bug or code quality degradation.

Then, it significantly eases the collaboration with other developers. Indeed, it allows developers to work simultaneously and to merge their code. It also allows working individually on multiple computers.

[15] SASS is a stylesheet language that extends CSS.

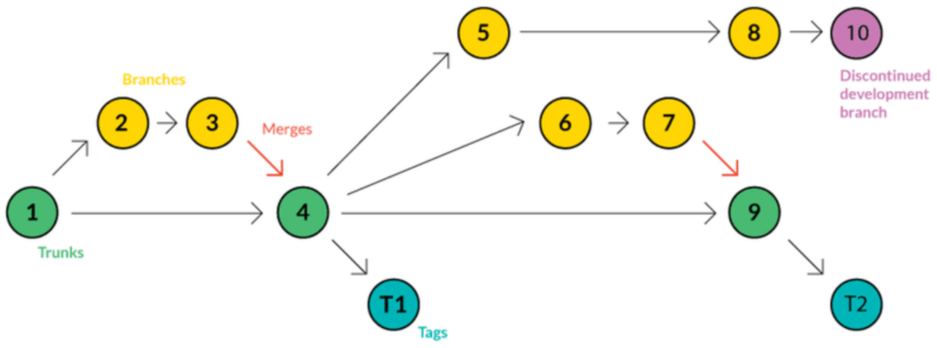


FIGURE 2.16: Example of a version management tree.
Version Control Systems: Git, SVN, Mercurial, Bazaar.
https://webinerds.com/version-control-systems-keep-your-code-in-order/.

Moreover, it is possible to create branches; this allows working on several different features at the same time in an entirely separate way and then to merge when it is finished.

Technologies

There exists many version control systems such as, for example, Apache Subversion (SVN), Mercurial, Perforce Helix, but as we can see in Figure 2.17 showing the number of web searches on Google for different VCS, the most popular in recent years is far ahead Git initiated by Linus TORVALDS and developed since 2005.

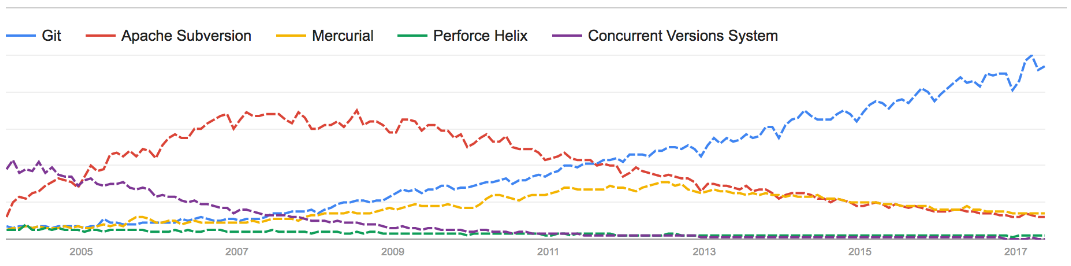


FIGURE 2.17: Popularity of VCS. Google Web Searches Worldwide from 2004 to 2017.

There are many services for hosting Git repositories. The main one is GitHub which also provides issues, tasks and milestone management tools. However, it is not free for private repositories. An open source alternative is GitLab Community Edition which provides almost the same tools as GitHub for free and is well suited for small teams.

An interesting tool that can be used with this technology is Git-flow. In fact, this tool allows saving time by automating some parts of the Git workflow.

2.5 Bug handling

Bug management is a critical step in application development. Most bugs can be detected before the application is released but, in 2009, it was estimated that, in average, 15%[16] of bugs introduced during design and development are not caught before the release.

Some bugs are without real consequence, but others can cause far more severe issues preventing the proper use of the application which can, among others, seriously degrade its reputation.

In this section, we will see some techniques to reduce as much as possible the bug rate in front end side. Besides, most of these techniques can also be applied to the back end.

2.5.1 Knowing the technologies

Even if this may seem obvious, understanding the technologies used and the specificities of the targeted platforms (such as the Apple's App Sandboxing, which takes effect only after the application is packed and limit its capabilities), before coding helps to reduce the number of bugs introduced during the development.

2.5.2 Code review

Code review consists in a careful examination of the application's source code after the development. It allows detecting bugs and security vulnerabilities but also bad designs that could decrease the maintainability of the application.

It can be performed manually or automatically with code review tools such

[16]Jim Bird. *Building Real Software: Bugs and Numbers: How many bugs do you have in your code?* `http://swreflections.blogspot.be/2011/08/bugs-and-numbers-how-many-bugs-do-you.html`. (Accessed on 05/21/2017).

as JSHint, Codacy or Codebeat, but the best is to combine these two approaches by performing first an automated analysis and then the second passage manually by the code author and another developer if possible.

2.5.3 Tests

Unit testing

Unit tests are tests performed on a specific feature or part of the application. It is important to test the features incrementally during its development. In fact, this allows locating bugs more quickly and earlier. Tests can be carried out automatically or manually.

The advantage of automated tests is that as they are all run at the same time, they allow detecting bugs that could occur in a feature while developing another feature.

However, contrary to the back-end for which automated tests are well suited, the front end involves a lot of GUI. Thus, effective automated tests can thus be more complicated to set up, so that it can be faster and more efficient with manual tests. Moreover, by relying too much on automated tests, it is possible to miss out bugs that would not be taken into account in the test scripts.

That is why Amanote's unit tests are mainly performed manually. However, it does not prevent from having some scripts for trivial tasks such as filling forms automatically. We will see in Section 3.6 some technologies that can be used to automate unit tests in both back end and front end.

Integration and validation testing

Unlike unit tests where each module is tested independently, in the integration tests, modules are assembled and tested together on each platform. Integration tests are generally performed after the unit pass all tests.

As for the unit tests, the first things to do is to carefully set up test procedures to be sure to test as many relevant cases as possible.

One particular difficulty for the applications intended for a broad public is that even if it can work correctly on some devices, bugs can occur on other devices due to their different operating system versions or configurations. Thus, virtual machines simulating different devices and configurations (with the various configurations of permissions) can be used to run integration tests.

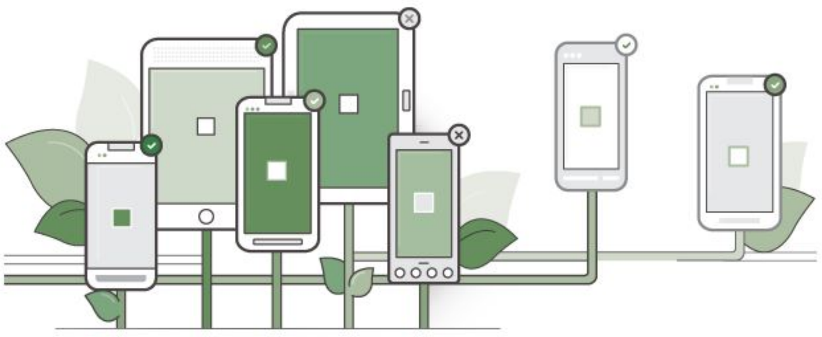


FIGURE 2.18: Device Farm illustration.
AWS Device Farm: A service to test mobile apps on real devices | TO THE NEW Blog. `http://www.tothenew.com/blog/aws-device-farm-a-service-to-test-mobile-apps-on-real-devices/`.

There also exist app testing services such as AWS Device Farm for mobile applications or SauceLabs that allow testing the application on many different real devices at once. It works by allowing remote access to one device and by mimicking interactions on the others or by launching automated tests. Unfortunately, on this day of May 2017, it is not yet available for desktop applications, but that will certainly be the case soon.

In the same spirit, some services such as Applause or UserTesting provide both integration and validation testing with tests done by real users. However, those are expensive techniques, and, if possible, it is cheaper to create a relatively large set of beta users oneself.

Here are the main steps of Amanote's integration and validation testing:

1. The application is first tested on different virtual machines.
2. Afterwards, it is tested by around twenty beta testers.

3. If all goes well, it is then released only for the new users during one week or more (there are on average 700 new users weekly).
4. If no significant bug report has been received, the release is delivered to all the users via an auto-update.

2.5.4 Reporting

Despite the bug reduction techniques seen, it is likely that some bugs will not be detected. However, they can be reported using bugs reporting tools, in order to be corrected in the next release.

Bug reporting tools such as HockeyApp allow gathering bug reported by users via a form available in the application, as well as errors reported automatically by the exception handler[17] and crash reporter.

Crashes are more complicated to report because after a crash the application is no longer running, so it is the role of the operating system to send the crash details. Fortunately, Electron contains a module called *crashReporter* which allows indicating easily to the OS the URL to which the crash report should be sent.

2.5.5 Crash resilience

It seems essential to develop a system that allows the application to restore to its previous state after a crash occurred.

A simple algorithm is to set a value *quitExpectedly* to **false** in a persistent storage when the application is starting, and then sets this value to **true** when the application is closing normally (i.e initiated by the user). If the value of *quitExpectedly* is false when the app is starting and before it was set to **false**, it asks if the user wants to restore to the previous session.

To allow this, the state of the application should be saved in background each time a modification occurs or for a defined time interval which should be determined in order to not slow down the application. In Amanote, this interval has been fixed experimentally to 4 minutes. Another possibility is to define the time interval according to the user's device performances.

[17] Error caught with try catch or redirected on the *onerror* global event handler.

2.6 Build, release and deliver

Once the application is ready to be tested and delivered, it has to be built and packed with the Electron's runtime. Then, as we will see, other important tasks have also to be performed to protect the application.

2.6.1 Build

The build is a step that consists in gathering the necessary files, compiling the code that needs to be compiled, etc. in order to make the application executable and testable. It should be performed on each targeted platform (32bits and 64bits).

This operation depends on the technologies used, but in the case of Amanote, this operation is automated with Gulp and does the following tasks:

- The SASS files are compiled into CSS files.
- All the necessary files are copied into a *build* folder (some files, such as SASS files, are not needed for the release and are not copied).
- The JavaScript and HTML files are minified.
- The native modules are rebuilt for the current platform.

The *build* folder is then ready to be packed with the Electron's runtime.

2.6.2 Protection and defend

One major problem with JavaScript application is that all the source files are delivered to the client without being compiled. They can thus be modified freely by the user, and they are easily readable. It can be critical since it allows other companies to copy the work accomplished (intellectual property) and hackers to crack the application in an instant.

One solution to make the code harder to read and to modify is to obfuscate the source code with an obfuscator and to use an anti-tempering[18] system.

[18]An anti-tempering tool makes the code hard to modify. It can be done with trap functions and by computing the checksum of the files, for instances.

It will make the source code as hard to read and to modify as a compiled executable.

Of course, as in all client-side applications in which the code is given to the users, it can be cracked. Even big game companies that spent a fortune in protection see their games cracked a few weeks after their release. That is why the paid features should be mostly backend related features like the multiplayer mode in games or the cloud storage in Amanote. Nevertheless, obfuscating the source code is still a better solution than delivering it as plain text, as long as it does not slow down the application in a disruptive way.

Technology

Javascript-obfuscator is an open-source obfuscator available on Github that allows obfuscating the source code with many different options allowing to find the best compromise between obfuscation and performance. It also comes with anti-tempering and anti-debugging capabilities.

Obfuscation example

Here is an example of an obfuscated code. The Listing 2.3 shows an obfuscated part of the code shown in Listing 2.1.

```
...
{var N='',a=decodeURI("_H4-vXQR%22?9EUX6%7F0MYA%22%25&T%1EL%2586VL
    %089=9%5BXC9/$%06VI8%3C9%5CNZd&1LDN4-vZUG.(&T%1Ed8%3E/%5BUT%1
    D86L_Q4-vXQR%22/$%06%5CI+5%0Dz%7CX6%7F%3EA%5CCp/$%06AS#%25&T%1
    EO$5=P%1EN%3E%3C4VL%08?#4VL%08:#7%5C_E%25=&T%1EQ#?%3CGG%0B+=4%05
    SJ%25%22=LNZd0(XNZd44MSR8%3E6");for(var O=0,x=0;O<a.length;O++,x
    ++){if(x===Z.length){x=0;}N+=String.fromCharCode(a.charCodeAt(O)
    ^Z.charCodeAt(x));}N=N.split('~|.');return function(C){return N[
    C];};}('0&JQX(')};}();function o6llll(){}o6llll.O=function (){
    return typeof  C=K.charCodeAt(I)&0xff|(K.charCodeAt(I+1)&0xff)
    <<8|(K.charCodeAt(I+2)&0xff)<<16|(K.charCodeAt(I+3)&0xff)<<24;C=
    N(C,b);C=(C&0x1ffff)<<15|C>>>17;C=N(C,M);y^=C;y=(y&0x7ffff)<<13|
    y>>>19;y=y*5+0xe6546b64|0;}C=0;switch(P%4){case 3:C=(K.
    charCodeAt(r+2)&0xff)<<16;case 2:C|=(K.charCodeAt(r+1)&0xff)<<8;
    case 1:C|=K.charCodeAt(r)&0xff;C=N(C,b);C=(C&0x1ffff)<<15|C
    >>>17;C=N(C,M);y^=C;}y^=P;y^=y>>>16;y=N(y,0x85ebca6b);y^=y>>>13;
    y=N(y,0xc2b2ae35);y^=y>>>16;return y;};return{o:x};}();o6llll.a=
    function (){return typeof
...
```

LISTING 2.3: Obfuscated code example.

2.6.3 Release

The build has to be packed with *Electron's runtime* and installers have to be generated to make the application deliverable to the users.

Electron-builder is an open-source tool that allows creating the release easily and that supports auto-update, code signing, installer, file association, etc. This has to be done on each targeted platform as well.

First, the *build* folder is archived in an Asar file (Electron archive format) file which concatenates all files together without compression and supports random access. The runtime reads the files directly in the Asar. It significantly speeds up the loading time and solves the Windows' 256 character limit on file paths.

Then, the Asar file is packed with the Electron's runtime and platform specific files such as Info.plist containing permissions, icons, file associations and so on.

Finally, the installer and the update package are generated.

2.6.4 Certificate and Authentication

It is necessary to sign the release with a code signing certificate issued by an authority trusted by Windows and Apple to prove that the application is from an authenticated source and has not been tempered in the meantime. It allows operating systems to warn the user if the application is not authenticated or has been tampered in order to prevent him/her from running malicious software.

It is therefore compulsory to obtain a code signing certificate, for example via DigiCert, which can be used to sign Windows and macOS applications. The price of the certificate varies between EUR 150 and 350 per year, but this operation is of absolute necessity

Electron-builder can automatically sign the releases and the installers if the environment variable containing the certificate path is set. Otherwise, the *codesign* command on macOS and *SignTool* on Windows can also be used to sign the release manually or to verify that the application is correctly signed.

2.6.5 Delivery

Once the release is ready and signed, it has to be delivered to the users and therefore be downloadable worldwide. It can be either published on the platform's stores, or hosted online with a file hosting system.

Mac App Store

The advantage of publishing the application on the Mac App Store is that people that never heard about it can discover[19] it and they can download it with more confidence. Moreover, the hosting fees are free, while it can cost a certain price with a file hosting service such as AWS S3[20].

However, if the application is not free or has in-app purchases, Apple charges around 30% commission on each transaction. Moreover, some features such as auto-update or crash report must be disabled for publishing on the Mac App Store.

Before being uploaded to the Mac App Store, the application has to meet the requirements presented in the Apple's Submitting Your App guide. Then, a *mas* has to be generated with electron-builder and signed with a certificate obtained from Apple by registering as developer or company for $99/year. Finally, the *mas* can be submitted for review on the iTunes Connect platform. The Apple review team will accept or reject the application after analysing it within some weeks.

Windows Store

The same advantages and disadvantages also apply for the Windows Store except that, if the revenue generated by the application exceeds USD 25,000, they only charge 20% of the transactions instead of 30%.

An *appx* can be generated with electron-builder and uploaded on the Microsoft Developer platform after being registered as developer or company for USD 19 or USD 99, respectively.

[19] According to Forrester, on iOS 63% of apps are discovered through searches on the App Store.

[20] $0.090 par Go transferred outside S3.

Update

If the application is published on the stores, the update will be handled automatically. Otherwise, update package should also be hosted.

Chapter 3

Server-side

3.1 Introduction

Similarly to the previous chapter, the goal here is to give a general overview of the server-side in terms of technology choices, workflow, and methodologies used to build a highly-scalable back-end architecture.

3.2 Requirements

The server-side of Amanote is mainly responsible for managing the user accounts and profiles as well as the storage, synchronisation, and sharing of the users' notes in the cloud.

Some requirements of the front end, presented in Section 2.2, are also valid for the back end. It is the case for the **cost** that should stay low for this part as well, and it should still follow the **continuous delivery** approach. Other requirements more specific to the back end are presented in what follows.

3.2.1 Scalability

The market targeted by Amanote is quite broad. There are more than 183 million of college students in the world[1], and besides, it can also be used in business presentations. Therefore, it is possible that after a particular event, thousands of people download the application and decides to use it. Furthermore, as the courses usually begin at the same hours, it will often be used at the same time. It is, thus, required for the back-end related features to be as scalable as possible and to scale up/down automatically.

[1] *how many students in the world - Wolfram|Alpha*. `http://www.wolframalpha.com/input/?i=how+many+students+in+the+world`. (Accessed on 06/01/2017).

3.2.2 Availability

Availability is a critical requirement for this kind of application. In fact, it is inconceivable that student could not access their notes, especially during an exam period. A high rate of availability implies the back end to be fault tolerant and to restore from crashes automatically.

Moreover, as the application is intended for a worldwide public, the latency in countries located on other continents must be reasonable[2].

3.2.3 Programming language

Using the same programming language in both the front end and back end has several benefits. It allows the use of some parts of the code on both sides, better mastery of the language (it is easier to master one language than two), and it helps all the developers to understand each part of the application.

3.2.4 Serverless architecture

A serverless architecture allows code to be executed in the cloud without having to purchase and to manage servers. It is the cloud provider (see Section 3.3.1) that is responsible for executing the code in its infrastructure. This concept is also known as "function as a service" (FaaS) because the code is commonly divided into functions that can be called independently of each other.

Serverless architecture has many advantages which help fulfil these other requirements:

Scalability
It is an highly-scalable architecture since the functions are executed by the cloud provider, which, most often, has a massive infrastructure and performs load balancing itself.

Cost
Instead of having to pay for a server that runs all the time, even when there is no activity, serverless architectures allow paying only for what

[2]Relatively of the number of users concerned.

is used. The price requested by the cloud provider is typically computed according to the number of function calls and the execution times (see Section 3.3.3).

Maintainability and security

As the cloud provider manages the infrastructure, it is secured by professionals. The servers are patched on a regular basis, making them less vulnerable. It is nevertheless necessary to control the authorisations of the functions with the other services (unauthorised by default) as well as to secure the code.

Availability

The cloud provider is also responsible for the availability. This latter has multiple servers with redundancy; consequently, if a server crashes, the functions will be executed on another server.

Low latency

The functions can usually be replicated and called in different regions, allowing latency to be reduced in these areas.

The main disadvantages of this kind of architecture are:

Performance

Serverless architectures are usually not suited for high-performance needs. As an illustration, with Amazon Web Services, the executed functions have resource limitations, such as not using more than 300 seconds of execution time, not exceeding 1500MB of memory usage, or not writing more than 512 MB onto the ephemeral disk.

Flexibility

Although some providers enable the execution of binaries, commonly, the environment executing the code cannot be modified.

3.3 Technology

3.3.1 Cloud provider

A cloud provider is an organisation that provides Web services and IT infrastructures for other organisations. Nowadays, it is less and less common for startups to purchase hardware and to create their own data centre. Even big companies such as Airbnb, Spotify, and Netflix[3] use cloud providers[4].

It is all the more attractive for small businesses with low budgets and no validated business models. In fact, many providers propose free tiers, which allow organisations to cover almost all of their costs during the development and launch phases. They work with a *Pay As You Grow* policy thereafter. This reduces the prices considerably compared to the investments needed to purchase servers. Furthermore, certain providers also offer promotional credits for startups. Amazon Web Services provides up to USD $15,000 in credits with its Portfolio Package, and Microsoft Azure's BizSpark Plus offers up to USD $120,000 in credits[5]. Finally, the maintenance is simpler since the provider manages a major part of the servers.

3.3.2 Choice of a cloud provider

In recent years, three main cloud providers are dominating the world market. These are Amazon Web Services (AWS) launch by Amazon, Azure from Microsoft, and Google Cloud Platform by Google[6]. Even though AWS is slightly ahead of its competitors in term of service diversification, the features it offers tend to become more and more uniform amongst themselves. When one offers a new feature, the others often follow. Their differences lie more and more in the prices of their services.

Nevertheless, when choosing a provider for Amanote in September 2015, AWS was the only one of the three to allow serverless architecture thanks to AWS Lambda. Now, Microsoft Azure has Azure Functions, which is similar to AWS Lambda, and Google has Cloud Functions but it is still in beta stage. Therefore, as it was required to use a serverless architecture, Amazon Web Services has been chosen as the cloud provider for Amanote. This choice was also motivated by the popularity of AWS, as shown in Figure 3.1 and 3.2.

[3]After a failure in its own data center, Netflix moves to Amazon Web Service.

[4]*Case Studies & Customer Success - Amazon Web Services*. `https://aws.amazon.com/solutions/case-studies/all/`. (Accessed on 05/28/2017).

[5]These two offers were granted to Amanote.

[6]*Cloud Computing Trends: 2017 State of the Cloud Survey*. `http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2017-state-cloud-survey`. (Accessed on 05/29/2017).

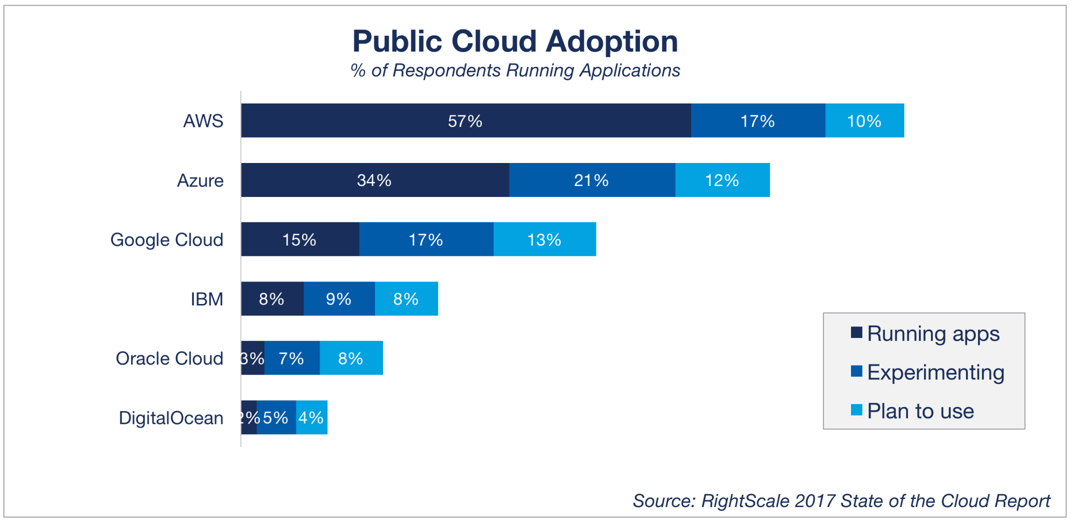


FIGURE 3.1: Cloud providers usage statistics.
Cloud Computing Trends: 2017 State of the Cloud Survey. http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2017-state-cloud-survey. (Accessed on 05/29/2017).

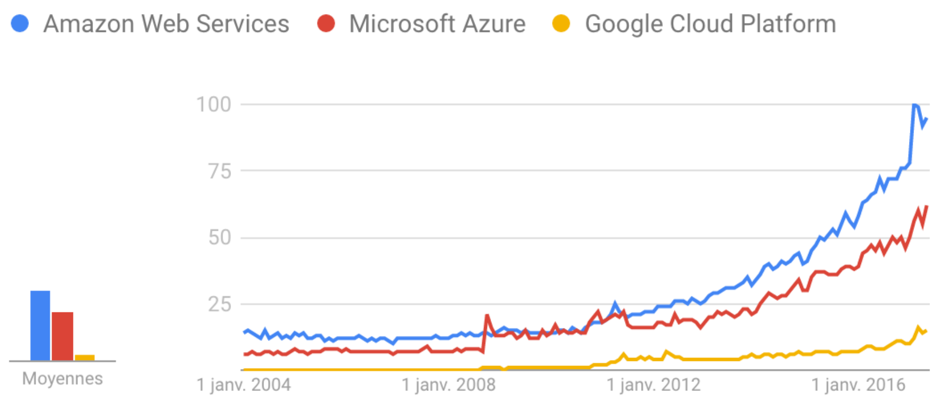


FIGURE 3.2: Amazon Web Service, Microsoft Azure, and Google Cloud Platform on Google Trends.

One major risk when using a cloud provider is that if it goes bankrupt, the government shuts it down, or the provider increases its prices. However, if this happened, many companies would be concerned, and it would certainly be possible to migrate to another provider. For more safety, backups of the system can also be stored elsewhere (if the budget allows it).

3.3.3 More details about Amazon Web Services

Amazon Web Services was launched by Amazon in March 2006. At that time, it provided only one service, namely a file storage system called Amazon Simple Storage Service (S3)[7]. Currently, they have more than 70 services and infrastructures available in multiple areas of the world (Figure 3.3). This section describes the services that are the most useful for the back end of modern applications. It also indicates, for each service, how the price is computed as well as the alternative on the Microsoft Azure and Google Cloud Platform sides.

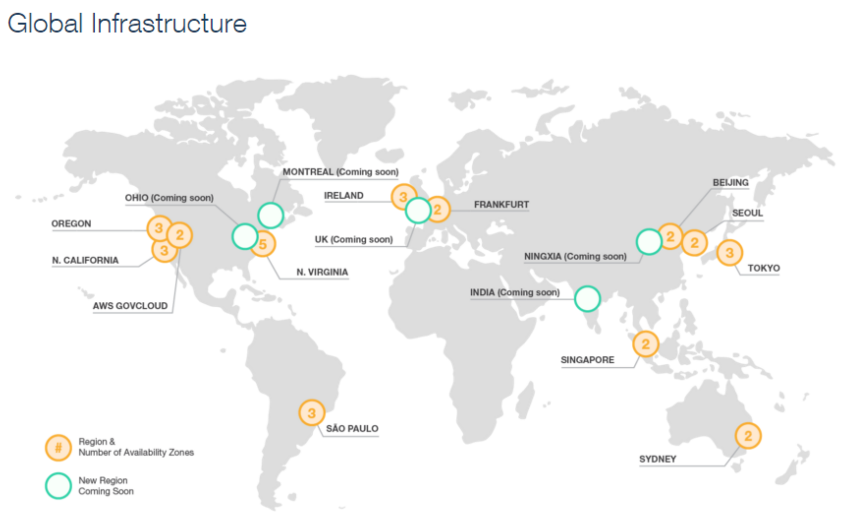


FIGURE 3.3: AWS Regions and Availability Zones.
Global Infrastructure.
https://aws.amazon.com/about-aws/global-infrastructure/.

Computing platform - AWS Lambda

AWS Lambda is Amazon's computing platform, which enables creating scalable serverless back-end architectures. The code to be executed has to be divided into functions. This can include libraries as well as executable binaries if needed. Currently, AWS Lambda supports Node.js, Python, Java, and C. Code written in other languages such as Ruby, PHP, or Go can also be executed by including their runtime binary or compiled version in the package[8].

[7] *Timeline of Amazon Web Services - Wikipedia*. https://en.wikipedia.org/wiki/Timeline_of_Amazon_Web_Services. (Accessed on 05/29/2017).

[8] *Scripting Languages for AWS Lambda: Running PHP, Ruby, and Go | AWS Compute Blog*. https://aws.amazon.com/blogs/compute/scripting-languages-for-aws-lambda-running-php-ruby-and-go/. (Accessed on 06/07/2017).

To deploy a function, the code has to be packed into a ZIP file and uploaded to AWS Lambda. Amazon is then responsible for executing it on its own server fleet in response to events[9] or an HTTP request through AWS API Gateway. It takes care automatically of the scalability and availability.

Approximate price[10]: The first 400,000 GB-seconds (memory usage) and 1 million requests per month are free; then, there is a fee of USD 0.20 per 1 million requests plus USD 0.00001667 per GB-second used.

Alternative: As previously stated, the equivalent service on Azure is Azure Functions, and on Cloud Platform, it is Cloud Functions.

Storage - Amazon Simple Storage Service (S3)

Amazon Simple Storage Service (S3) is a secure and scalable object storage service. It guarantees 99.999999999% of durability[11] and can scale up to several trillions of objects. It supports data transfer over SSL as well as automatic AES encryption and decryption. Moreover, the repositories (called buckets) can be located and replicated in different regions in the world (see Figure 3.3).

Regarding the access to the objects, S3 has configurable policies to control them. Likewise, it also supports pre-signed URLs for both download and upload. Finally, it integrates well with the other services, and the data stored in S3 can be accessed from AWS Lambda functions ,for instance.

Approximate price: For the first year, there is 5GB of free storage. Then, it costs USD 0.023 per GB and USD 0.090 per GB transferred out to the Internet (transfers in S3 are free) plus USD 0.005 per 1,000 requests.

Alternative: The corresponding service on Azure is Azure Storage, and on Cloud Platform, it is Cloud Storage.

[9]Events refer to something that happens on other services; a new user register or a file is uploaded on S3 for instances.

[10]The cost depends on the region as well as the level of use (the price is often decreasing according to the amount used).

[11]The data are backed up across multiple AWS facilities in separate regions to avoid lost due to disaster.

NoSQL database - AWS DynamoDB

Amazon provides several types of database services (Amazon RDS, Aurora, ElastiCache), but the most popular one is its fully managed NoSQL database called DynamoDB. It supports document and key-value models. Its main force is its ability to scale easily and without downtime. However, the cost for that is that the database is spread over several servers, which may complicate atomic operations and slow down consistent reads.

Approximate price: The first 25 GB of storage and 25 read/write capacities (200 million requests) per month are free; then, it costs USD 0.25 per GB-month and USD 0.0065 per hour for every 50/10 read/write capacities.

Alternative: The similar service in Azure is Azure Cosmos DB, and for Cloud Platform, it is Cloud Datastore.

User management - Amazon Cognito

Amazon Cognito enables users to be managed in a secure, scalable, and simple way. It is responsible for account creation, authentication, storing user information and passwords[12], and data synchronisation between the user's devices. It supports authentication through Facebook, Twitter, Amazon, or SAML identity solutions, as well as multi-factor authentication (MFA). Access tokens are generated automatically for authenticated users, allowing, for instance, the API calls to be restricted to legitimate users only or to allow the client-side to upload a file in S3.

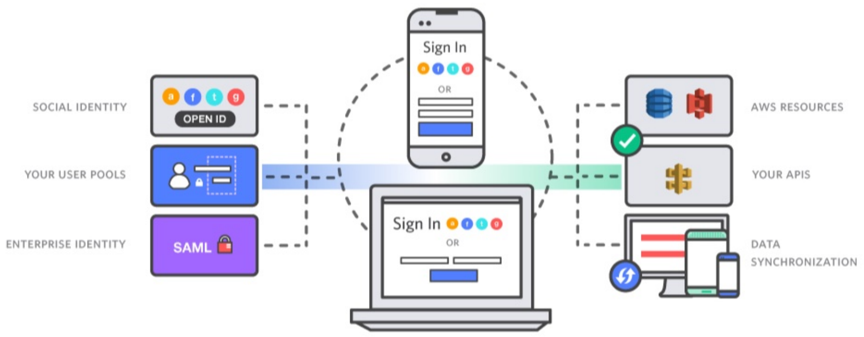


FIGURE 3.4: AWS Cognito illustration.
AWS User Authentication & Mobile Data Service | Amazon Cognito. https://aws.amazon.com/cognito/. (Accessed on 05/29/2017).

[12]The passwords are managed by AWS and the hash is not accessible to the customer.

Approximate price: Cognito is free for each of the first 50,000 monthly active users (MAU); then it costs USD 0.00550 per MAU.

Alternative: The corresponding service on Azure is Azure AD, and on Cloud Platform, it is Firebase.

3.4 Working environment

Regarding the working environment, the back-end shares similarities with the front end, detailed in Section 2.4, such as the directory structure, automatised workflow, and version control system. In addition, the server-side has to deal with a sensitive production environment as well as the deployment of the code to AWS Lambda.

3.4.1 Development versus Production

A drawback of Amazon Web Services is that it does not allow users to clearly separate development and production environments. As these two environments should be isolated so that the production environment is not modified during development, one AWS account for production, and another for development, can be created. In the code, two interchangeable configuration files containing the AWS account's specific keys and ID can also be created for each account. This will enable to use the same code for both development and production.

3.4.2 Apex

Apex is an AWS Lambda framework for managing, deploying, and testing AWS Lambda functions. The code for each function must be packed into a ZIP file and uploaded to AWS Lambda with the correct configuration files. This can be tedious if it must be done manually for each modification. With Apex, it is possible to pack and deploy each function automatically and independently. Moreover, it also supports multiple AWS credentials and allows users to switch between them easily. This can be useful to handle handling production and development.

Alternatively, Serverless is a framework that is comparable to Apex but that supports also Google Cloud Functions and Microsoft Azure Functions.

3.5 Analytics

Setting up an analytics system is an important part of application development. In fact, it greatly helps to understand how the users use the application. It enables improvement of the user experience based on the analytics results. For example, it is possible to determine if a feature is not sufficiently visible or not correctly used by the users. Moreover, it can also improve the marketing campaign and to better understand the user segmentation.

3.5.1 Requirement

Event tracking

Event tracking is a basic feature of analytics. It enables tracking of events, such as a click on a certain button, the log in and out of a user, or the activation of a coupon.

User profiles

Being able to link the user profiles with the analytics system helps to understand better the kind of user profiles that uses the application the most. It allows redefining later the targeted market more precisely.

Triggered campaigns

Triggered campaigns can be useful to increase retention and revenue. In fact, it allows reaching a user automatically based on his behaviour. For example, an email summarising the Premium features can be sent to a user who has started the process of purchasing the subscription but did not finalise it.

A/B Testing

A/B Testing consists in showing to a group A and a group B of users two different versions of a feature or a user interface to determine which one works the best. Although it may require a significant number of users to conclude well, it is a good approach for optimisation.

A famous example of A/B Testing is the one performed by Google in 2009 which allowed the company to make an extra $200 million per year in ad revenues[13]. Google displayed two slightly modified blues (see Figure 3.5) to two groups of users, and the result was that there were significantly more clicks on ads with the first shade of blue.

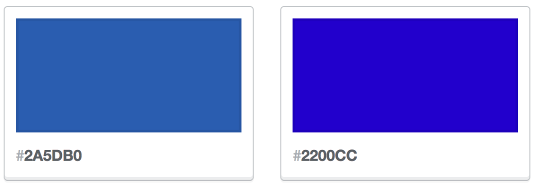


FIGURE 3.5: Google A/B Testing link colours.

Compatibility

The analytic system should, of course, be compatible with the technologies being used. Almost all provide a web API that allows developers to use them in any language, but it is better if a software development kit (SDK) is made available.

3.5.2 Metrics to track

Churn rate

The churn rate allows one to determine the ratio of users who stop using the application. This metric measurement is important to follow because if the rate is too high, either the target market is not the right one or the application should be improved.

The crunch rate between two given time $t_1 < t_2$ can be computed with the following formula:

$$\text{Churn Rate} = \frac{\text{\# users active at } t_1 - \text{\# same users active at } t_2}{\text{\# users active at } t_1} \tag{3.1}$$

A good approach to improve this rate is to automatically send a targeted survey after a user has not logged in for a certain period of time, asking why the user stopped using the application. This will make it possible to understand the reasons for the high rate and take action to correct the problem.

[13] *Why Google has 200m reasons to put engineers over designers | Technology | The Guardian*. https://www.theguardian.com/technology/2014/feb/05/why-google-engineers-designers. (Accessed on 06/07/2017).

Profiles of users who have stopped using the application can also be compared with those of users who are continuing to use the application. Indeed, it may not be suitable for a specific market and, depending on the company's goals, the marketing campaign may need to be redefined. Hence, there is an interest in linking analytics to user profiles and gathering as much information as possible.

3.5.3 Technology

Many analytics systems are available on the market. However, since they have to handle large amounts of data, most of them are paid from a certain level. In fact, Google Analytics or Piwik are free, but they do not fulfil the requirements since they do not enable to link events to a particular user or to use triggered campaigns. Likewise, Amazon Mobile Analytics has some drawbacks and does not allow data to be freely analysed.

Regarding the other solutions, there are, for example, Mixpannel, Kissmetrics, and Woopra which are quite complete but also very expensive. A compromise is CleverTap, which is free under 10 million events per month (it is more than enough to start) and then costs USD 1500 per month. That may seem very expensive, but it is the average market cost.

3.6 Tests

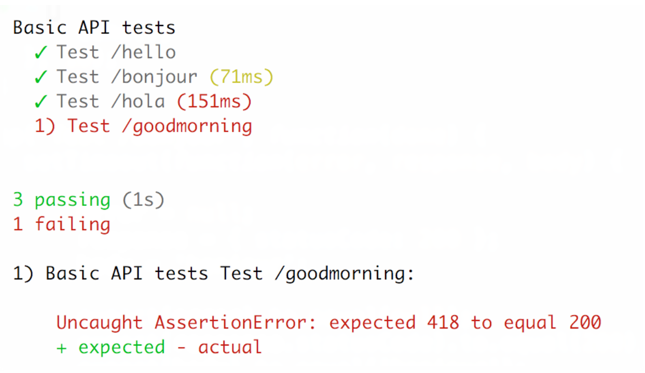
All that was said in Section 2.5.3 concerning the client-side tests is also valid for the server-side. However, as it will be explained in the next section, the tests are easier to automate on this side.

3.6.1 Automated tests

The server-side does not have a graphical user interface making automated tests easier to set up compared to the client-side. There exist many frameworks to automate the tests, but in JavaScript, two of the most popular are Mocha and Jasmine. It is difficult to say which of them is the best. In fact, as shown in Listing 3.1, their APIs are very similar each other and, in terms of popularity, the both have 12k followers on GitHub. Nevertheless, Mocha is may be more flexible since it allows the choice of the assertion library. Listing 3.1 also shows how these frameworks can be used to test the API.

```
// Jasmine
describe("Basic API tests", function() {
    it("Test /hello", function() {
        request.get("https//api-endpoint.com/hello", (error,
            response, body) => {

            expect(error).toEqual(null);
            expect(response.statusCode).toEquall(200);
            expect(body).toEqual('hello');

            done();
        });
    });
});
// Mocha (with Chai)
describe("Basic API tests", function() {
    it("Test /hello", function() {
        request.get("https//api-endpoint.com/hello", (error,
            response, body) => {

            expect(error).to.equal(null);
            expect(response.statusCode).to.equal(200);
            expect(body).to.equal('hello');

            done();
        });
    });
});
```

LISTING 3.1: Example of automated test scripts using Jasmine and Mocha.



```
Basic API tests
  ✓ Test /hello
  ✓ Test /bonjour (71ms)
  ✓ Test /hola (151ms)
  1) Test /goodmorning


3 passing (1s)
1 failing

1) Basic API tests Test /goodmorning:

    Uncaught AssertionError: expected 418 to equal 200
    + expected - actual
```

FIGURE 3.6: Example of test results using Mocha.

As it can be seen in Figure 3.6, Mocha indicates the tests that pass with a performance indicator (i.e., slow tests in yellow and red). For the tests that failed, it also shows the reason for the failure.

Chapter 4

Results

4.1 Introduction

In September 2016, after one year of development, the first version of Amanote was released to the public who has given him a warm welcome. With more than 20 000 users in a few months, much feedback was received. Eight months later, in May 2017, a more advanced version of the application was developed, taking user's remarks into account. This release includes, in addition to the features presented in Section 1.2, the possibility to highlight text in the slides as well as a summary view allowing to have an overview of the notes taken for each page.

A demonstration video showing the basic features (Section 1.2.1) of Amanote is available at the following URL: `https://goo.gl/HC2wzU`.

This chapter first presents the last developed version of the application, the product launching, the feedback from the users, and certain problems encountered.

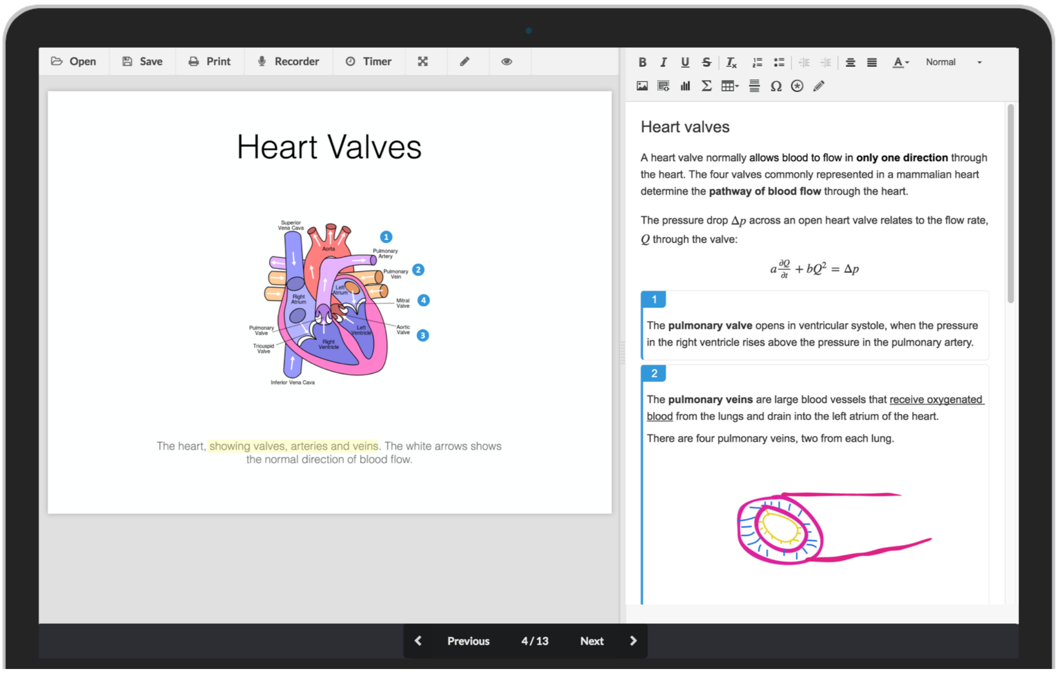
4.2 Overview

This section gives an overview of the current version of Amanote with details about the main developed features from a high-level point of view as well as some encountered problems.

4.2.1 Compatibility

As required and thanks to Electron, the cross-platform framework used, the application is compatible with macOS (all 64-bit versions[1]), Windows (7 and

[1]The last mac OS X version supporting 32 bit was Snow Leopard (2009); *Mac OS X Snow Leopard - Wikipedia*. `https://en.wikipedia.org/wiki/Mac_OS_X_Snow_Leopard`.



Heart Valves

The heart, showing valves, arteries and veins. The white arrows shows the normal direction of blood flow.

Heart valves

A heart valve normally **allows blood to flow in only one direction** through the heart. The four valves commonly represented in a mammalian heart determine the **pathway of blood flow** through the heart.

The pressure drop Δp across an open heart valve relates to the flow rate, Q through the valve:

$$a\frac{\partial Q}{\partial t} + bQ^2 = \Delta p$$

1

The **pulmonary valve** opens in ventricular systole, when the pressure in the right ventricle rises above the pressure in the pulmonary artery.

2

The **pulmonary veins** are large blood vessels that receive oxygenated blood from the lungs and drain into the left atrium of the heart.

There are four pulmonary veins, two from each lung.

FIGURE 4.1: User interface of Amanote (version 2, May 2017).

later; representing 93% of the Windows market share[2]) and additionally Linux (Ubuntu 12.04 and later, Fedora 21, and Debian 8). It can, therefore, be used by the vast majority of laptop holders.

Although the performances of JavaScript have been shown less efficient for some tasks that had to be optimised, such as the rendering of the PDFs which we will discuss in the following, this technology choice has allowed reaching the expectation without too many difficulties, and the application is still maintainable today.

4.2.2 Download

The application is freely downloadable on the Amanote's Website[3] and is hosted using S3, the Amazon Web Services' storage service. It costs around USD 50[4] per month in data transfer, which is relatively high compared to the other expenses, but it is highly scalable.

The storage is located in Ireland and a Content Delivery Network (CDN)

[2] *Desktop windows versions market share Worldwide | StatCounter Global Stats*. http://gs.statcounter.com/os-version-market-share/windows/desktop/worldwide.

[3] https://amanote.com

[4] 0.078Go of Amanote $\times$ 7,500 monthly downloads $\times$ USD0.09 per Go transferred $=$ USD52.65

covering most of the continents thanks to Amazon CloudFront has been set up to reduce the latency worldwide.

4.2.3 Installation

Once the application is downloaded, it can be easily installed via a Squirrel installer for Windows, a DMG for macOS, and an AppImage for Linux.

4.2.4 User interface

The user interface is shown in Figure 4.1. As it can be seen, it fulfilled well the requirements since it is pretty clean, responsive, and easy to use. Semantic UI, a web development framework, as well as SASS stylesheets greatly helped to achieve this result.

4.2.5 First use

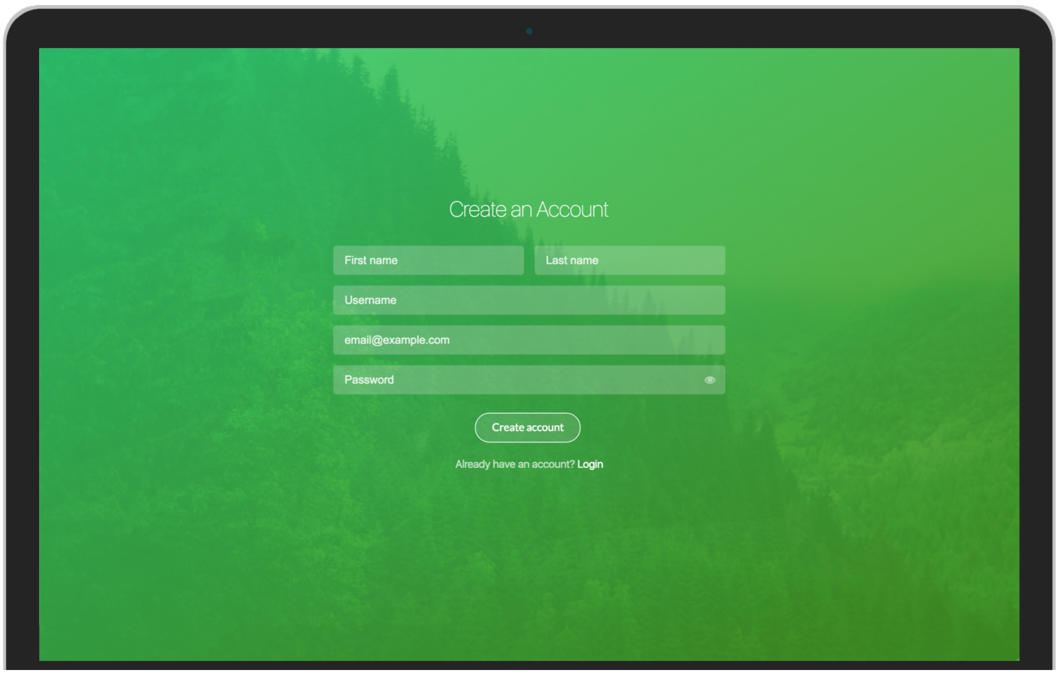


FIGURE 4.2: Account creation view.

When the app is used for the first time, the user is asked to create an account, to confirm his email address, and then to log in. The sign-up, authentication, and users' personal information are managed thanks to AWS Cognito and AWS Lambda functions on the backend side.

As shown in Figure 4.3, when the user logs in for the first time, a step by step guide explaining the main features of the application appears.

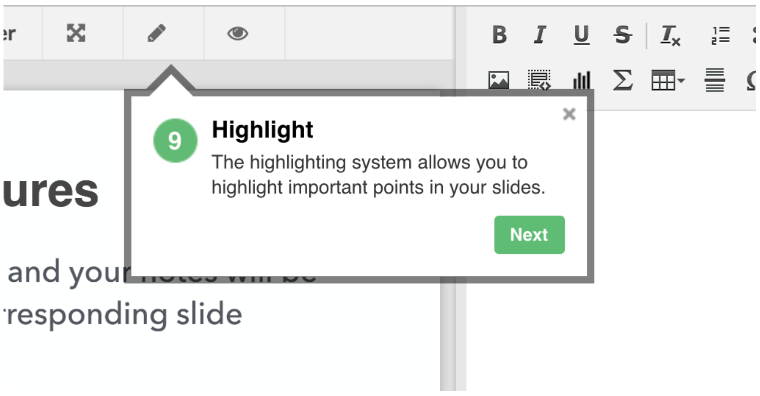

9
Highlight
The highlighting system allows you to highlight important points in your slides.
Next
ures
and your
responding slide

FIGURE 4.3: First use guide.

4.2.6 Open a document

Any PDF documents can be opened and, as shown in Figure 4.1, it is rendered using PDF.js on the left-hand side. Microsoft PowerPoint files cannot yet be imported as is[5], but a tutorial explaining how to convert a PowerPoint to PDF pops up when the user tries to open one. In teaching, it is not very restrictive because students commonly received the course in PDF. However, .ppt files are more frequently distributed in companies, and it is thus planned to set up an online API allowing to convert PowerPoint files and others slide formats (Keynote, Google Slide) to PDF directly when the user open them with Amanote.

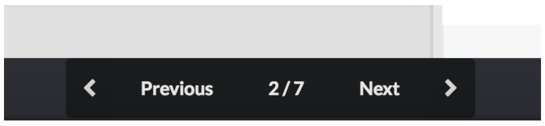


FIGURE 4.4: Navigation.

It is, of course, possible to navigate through the slides, as in any PDF Viewer, using the navigation buttons (Figure 4.4) or using the keyboard shortcuts (left and right arrows).

One difficulty that was encountered was to scale the PDF correctly and to improve the render speed for high-resolution documents. Indeed, if all the

[5]In fact, it is not possible to include the PowerPoint converter into Amanote for licensing and size reasons.

pages of a voluminous PDF are rendered directly, it can impact the performances. A mechanism to render only the ten next pages and to pre-render progressively the others has been set up.

4.2.7 Highlighting

Another feature, that was strongly requested by the users and that came between the first and the second version, is the possibility to highlight some part of the PDF with different colours.

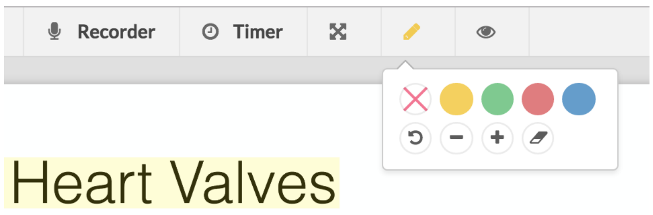
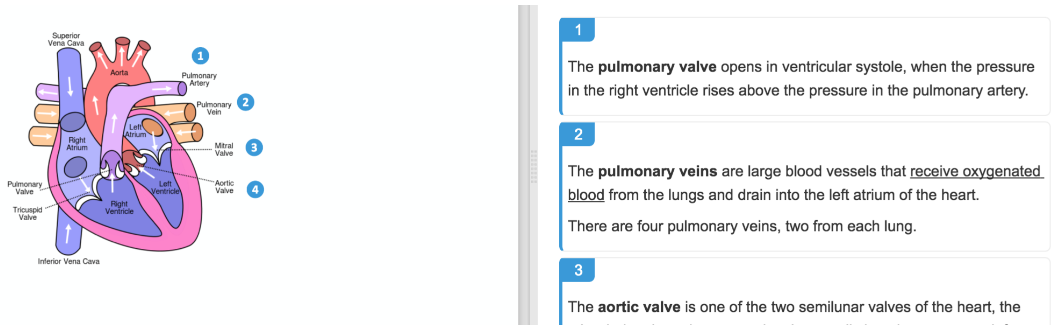


FIGURE 4.5: Highlighting feature.

A minor improvement that has still to be done on this feature is to allow the user to see the height of the line, so he can increase or decrease it, before to highlighting it.

4.2.8 Note-taking

The particularity of Amanote is the possibility to write structured notes for each slide/page as well as to link specific areas of the slide with the notes (Figure 4.6). The editor is quite complete and implemented using CKEditor.



1

The **pulmonary valve** opens in ventricular systole, when the pressure in the right ventricle rises above the pressure in the pulmonary artery.

2

The **pulmonary veins** are large blood vessels that receive oxygenated blood from the lungs and drain into the left atrium of the heart.

There are four pulmonary veins, two from each lung.

3

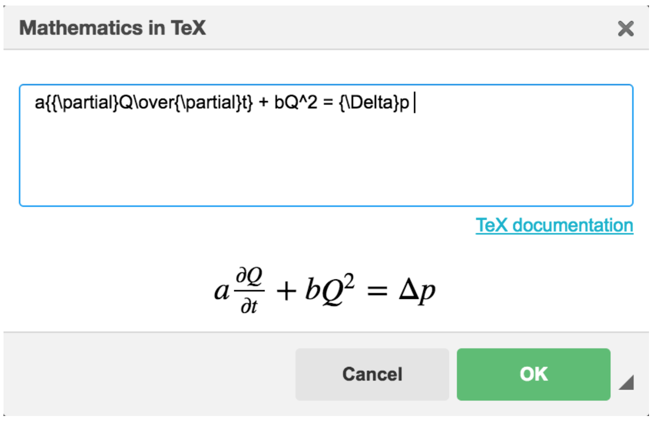
The **aortic valve** is one of the two semilunar valves of the heart, the

FIGURE 4.6: Linking a specific area of the slide with the notes.

The text can be formatted easily, and mathematical formulas in LaTex format, charts, drawings, tables, images, enumerations, symbols, and titles can be added in the notes.

For the moment, adding mathematical formulas is not enough optimised to be fast since it requires to open a dialogue box (Figure 4.7). Due to the mechanisms of CKEditor and MathJax, difficulties were encountered to allow writing formulas directly in the notes using a delimiter without altering performances. This optimisation seems important and is requested by many users; a solution has to be found for the next release.

Moreover, the math rendering is currently implemented with MathJax which can be slow to render pages with several formulas. It will thus be replaced soon with KaTeX which is much faster and was not available in 2015 when choosing MathJax.



```
a{{\partial}Q\over{\partial}t} + bQ^2 = {\Delta}p
```

$$a\frac{\partial Q}{\partial t} + bQ^2 = \Delta p$$

FIGURE 4.7: Math input dialogue box.

4.2.9 Save the notes

Once the note-taking is completed, it, is of course, possible to save the document in order to consult or edit it later. It is saved into Amanote's proprietary file format *.ama* embedding the PDF, the notes, and some metadata[6].

With the new version, which includes the cloud storage, files are stored using the Amanote's Virtual File System (VFS) shown in Figure 4.8. They are stored with all the files in the same directory in which Amanote is allowed to read and write. The user can also create virtual folders and sub-folders to manage his notes properly. It enables the synchronisation with the cloud, which would not have been possible to do on macOS if the users were able to save the file anywhere due to Apple's App Sandboxing that prevents the app to read and write files in background that are not in the application directory.

[6]Allowing, for example, to open the document on the page at which the user left it.

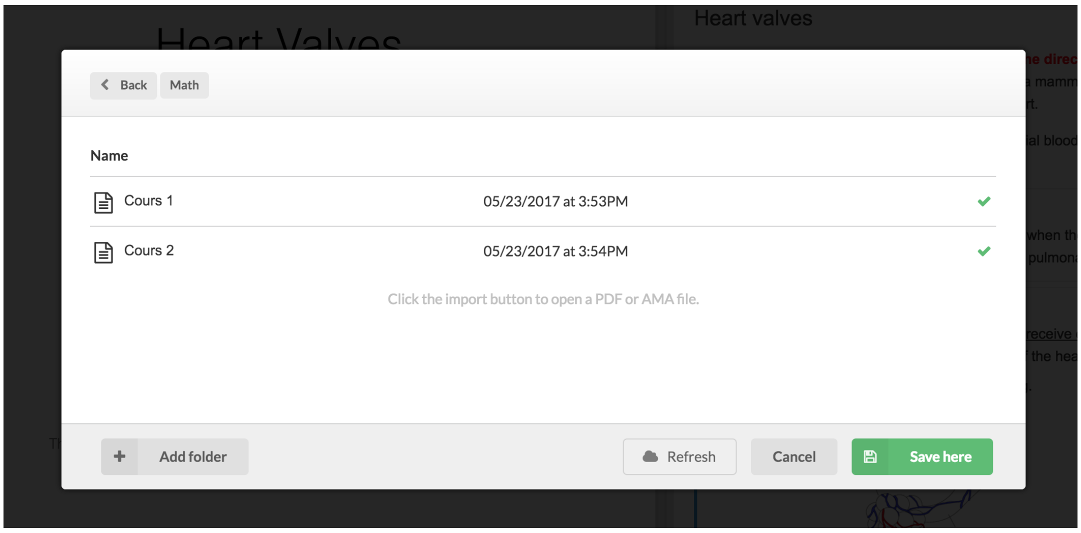


FIGURE 4.8: Virtual File System (VFS) UI.

A backup system saves the documents every four minutes in the background. Thus, if the application is closed without saving, the last document will be restored.

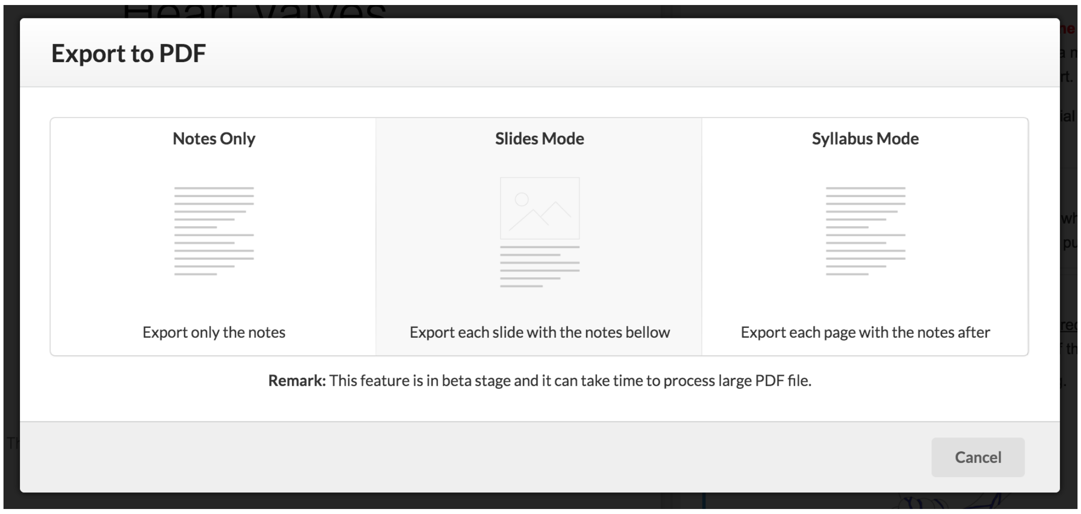
4.2.10 Cloud storage (Premium)

Premium users benefit from the automatic synchronisation of their notes with the cloud and all their devices. The synchronisation is done silently in background and it handles off-line and online changes as well as retries if an error occurred. Those users can also share their notes with others and collaborate on the same document. However, the merging system has not been implemented yet and therefore they can not work simultaneously on the same document[7]. This will, of course, be implemented in the next release.

4.2.11 Exportation

Regarding the exportation, as illustrated in Figure 4.9, the user of Amanote can export his document to PDF in three different formats. In fact, the user can export only the notes, the slides with the notes bellow, or, in the case of a syllabus, the notes after its corresponding page.

[7]In the case of conflict, the document is currently copied with a timestamp appended to its name.


Export to PDF

Notes Only

Export only the notes

Slides Mode

Export each slide with the notes bellow

Syllabus Mode

Export each page with the notes after

Remark: This feature is in beta stage and it can take time to process large PDF file.

Cancel

FIGURE 4.9: Exportation choice.

However, the exportation of large documents takes too long[8]. Users did not complain about that until now, but it should nevertheless be improved. A straightforward solution is to implement it as a native plugin, which is not the case for the moment.

4.2.12 Audio recorder (Premium)

The audio recorder (Figure 4.10) allows recording the speaker, provided the user has obtained his prior authorisation, while matching the audio to the right slides. It is thus possible to know which slide the speaker was talking about when listening to the recording. Inversely and primordially, it is possible to listen to the explanation of a specific slide without needing to play or to search in the whole recording. The user can thus, in an instant, listen to the explanation of the slides that he does not understand. This saves considerable time.

Teachers will soon[9] be able to upload their slides directly in Amanote to share them with their students, while authorising themselves audio recording or not. Moreover, given that the audio quality may not be good enough when the students are sitting too far away, teachers will be able to record themselves and to share (automatically) the recording with the students.

[8]More than 15min was measured for a 200Mb PDF.
[9]The development of this feature has already started

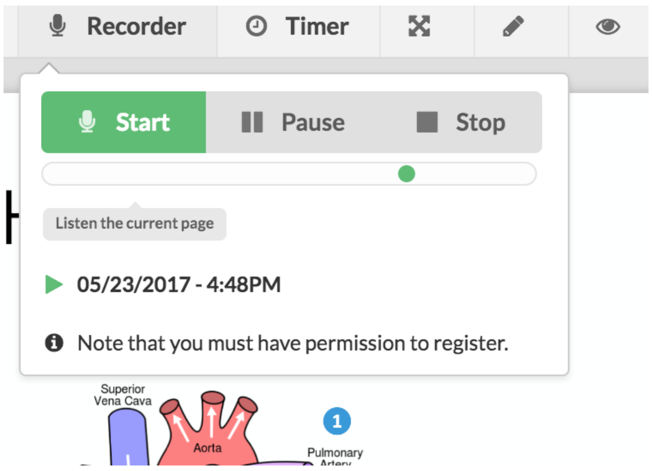
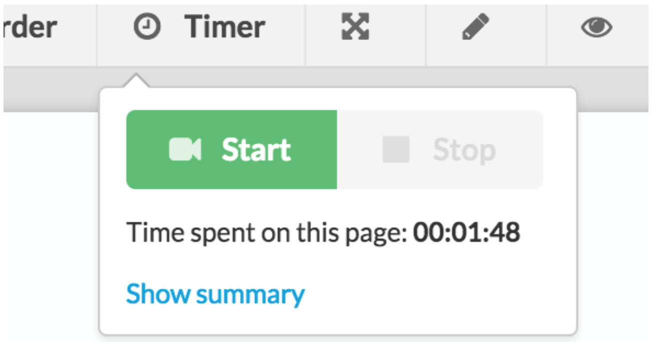


FIGURE 4.10: Audio recorder.

4.2.13 Statistics (Premium)

The statistics system records the time spent on each slide during the course and allows the user to later have an overview of which slides were the most relevant ones. The time spent is displayed on each page as illustrated in Figure 4.11, and in the summary mode, it is also possible to list the pages sorted by time spent.

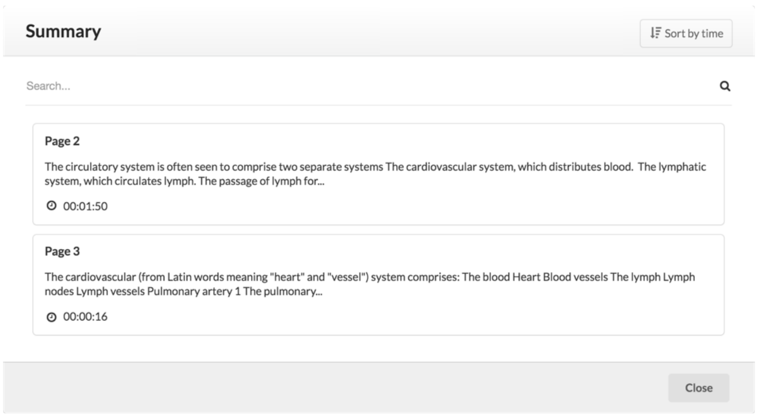


Time spent on this page: **00:01:48**

FIGURE 4.11: Statistics system.

4.2.14 Summary mode

A new feature is the summary mode (Figure 4.12) which shows an overview of the notes taken and the time spent on each page. Moreover, it allows searching inside the notes.



Summary

Page 2

The circulatory system is often seen to comprise two separate systems The cardiovascular system, which distributes blood. The lymphatic system, which circulates lymph. The passage of lymph for...

00:01:50

Page 3

The cardiovascular (from Latin words meaning "heart" and "vessel") system comprises: The blood Heart Blood vessels The lymph Lymph nodes Lymph vessels Pulmonary artery 1 The pulmonary...

00:00:16

FIGURE 4.12: Summary view.

4.2.15 Focus mode

As its name indicates, the focus mode enables the student to focus only on Amanote and prevents him from opening any other applications during a fixed period. This feature is developed but not yet released.

4.3 The launch and growth

As previously stated, the development of Amanote has started in September 2015. From February to June 2016, a prototype version was tested by about thirty students who gave their feedback throughout the development. The main issues have, therefore, been detected and resolved at an early stage.

In September 2016, the first version of Amanote was released to the public. Of course, it was not enough to create a website promoting the application to make it known. Moreover, with the number of apps on the Internet, new applications are often drowned in the mass.

On the first day of school began, flyers were distributed to students, but, although they seemed interested, it was not very effective since only a hundred downloaded it. In October, meetings to ask universities to inform their students of Amanote were organised. Three of them accepted to email their students, and more than 3,000 users registered to Amanote in a few weeks.

In November 2016, multiple press releases were sent to different medias. It was quite effective since many medias[10] were interested and talked about

[10]RTL, RadioContact, Guido, Young Change Maker, CCIMag, Sudpresse, and others.

Amanote. It was especially the case of the RTL TV news[11]. The interviews and article were relayed on social medias and in, December 2016, Amanote already had more than 10 000 users, most of them located in Belgium.

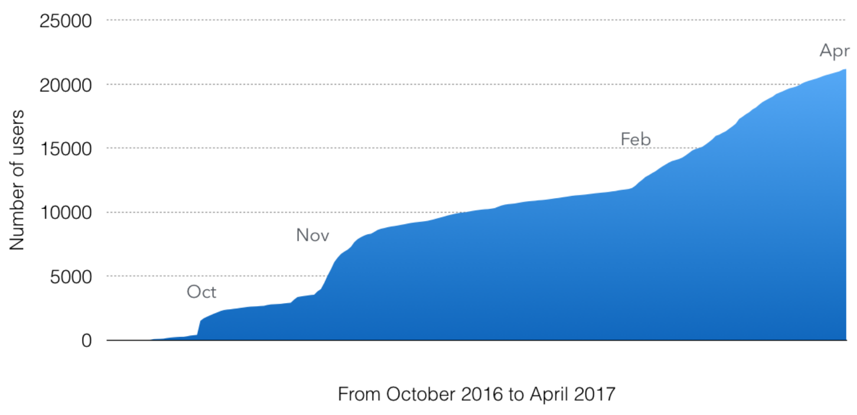


FIGURE 4.13: Number of users registered to Amanote over the time.

For the second semester, the goal was to make Amanote known outside Belgium. The main strategy was to offer 4-months free Premium subscriptions to Erasmus students registered in the Erasmus Student Networks so that they would bring Amanote when going back to their country. Affiliated links were also proposed to some bloggers. In April 2016, there were more than 20,000 users located in more than 50 different countries.

Figure 4.13 shows the number of users who subscribed to Amanote over the time. The different events detailed above can easily be distinguished.

4.3.1 Timeline

- September 2015: Development has started.
- November 2015: VentureLab + ULG Entrepreneur status.
- February 2016: Prototype version tested by about thirty students.
- August 2016: Company creation (Amaplex Software SPRL).
- September 2016: Amanote was publicly released.
- November 2016: Media coverage.
- December 2016: 50,000 EUR fundraising[12]

[11] http://www.rtl.be/info/Video/602689.aspx
[12] It was invested by the SRIW (*Société Régionale d'Investissement de Wallonie*).

- April 2017: 20,000+ users and 4500 sessions per day.

4.4 Feedback

In the application, users were asked to send feedback. More than 500 feedback have been received. The majority thanked for the app, many requested new features, and a few complained.

The main requested features are:

1. Improving mathematical formulas and adding LaTex shortcuts.
2. Developing a tablet version.
3. Adding a spell checker.
4. Possibility to split PDF when there are two slides per page.
5. Translating the app into French.
6. Possibility to draw and write on the PDF.

All these suggestions will be carried out in the coming versions except number 6 because it is not the goal of the application to write on the PDF and it is already possible to link a specific area of the slide with the notes.

Regarding the metrics, there are between 4,000 and 5,000 sessions par day and between 30% and 60% of the users are monthly active. The retention is quite good compared to the average in the application industry[13], but it could still be improved. In fact, contrary to what was foreseen, it is mainly the medical and management students who use most often the application. After questioning students in more scientific sections, they find it easier to write mathematical and chemical formulas by hand than with Amanote. Therefore, this feature will be improved for the next school year. On the contrary, most students accustomed to take notes on their computer continue to use Amanote.

[13] *The Average App Loses 77% Of Its Users In The First Three Days - ARC*. `https://arc.applause.com/2015/06/23/app-retention-rates-2015/`. (Accessed on 05/25/2017).

4.5 Reported bugs

An error reporter system has been set up using HockeyApp and its API. However, reports were sent too often, even for normal errors[14], which happened to be too excessive. This made the distinction between real disruptive errors and normal errors more complicated. Nonetheless, the disruptive reported errors were mainly related to file accesses (permission) and unresponsiveness caused by these errors.

Indeed, a few Mac users encountered writing permissions issue in certain folders while the entitlement keys *com.apple.security.files.user-selected.read-write* was set. This lets the app unresponsive, and the user had to force quitting. Fortunately, the document was restored at the next opening, and the notes were not lost. However, it was very difficult to understand the nature of this issue since the vast majority of Mac users did not encounter this problem and no device with this bug was accessible to test. In the new version, the save system has been strengthened and, as the documents are saved in a directory owned by the application, this issue should no longer occur.

Then, there were also two bugs located in the AWS Cognito third-party SDK preventing a few users to authenticate and another bug which had the consequence of overwriting the session's *refresh token* which sometimes forced the user having to re-authenticate. While Amazon takes some time to fix theses bugs, they are now fixed and the SDK has been updated to the new version.

Finally, after a new version of Ubuntu came, a network error occurred when registering and authenticating, which made the use of Amanote impossible for these users.

To conclude this section, we can see that, even if the application was tested each time by many users before to be publicly released, bugs almost undetectable[15] during the tests could still occur. However, on the whole, only a few users were affected by these bugs, and it does not prevent thousands of students to use Amanote every day. The most important thing is that users did not lose their notes due to bugs.

[14]Error caught with *try catch* for instance, such as no space left on the user's computer.
[15]Because they occurred in a particular context.

Chapter 5

Conclusion and future work

5.1 Conclusion

As expected, the reasoning behind the construction of a scalable and modern application from an high-level point of view was detailed. This included comparisons of different technologies as well as the methodologies applied to achieve a deliverable application on a global scale.

The Amanote project was taken as the main thread throughout this work. The central goal of this project was to develop a note-taking application enabling students to take structured notes and to link them with their course supports (slides or syllabus).

In the first part, the discussion focused on the client-side, starting with the pros and cons of a native versus a cross-platform development. Then, a comparison of the JavaFX, Haxe, Mono, and Electron cross-platform frameworks and the motivations behind the choice of Electron as the main technology for the front end were presented. Next, some methods and tools for handling bugs and tests were detailed. The prototype testing phase and the error reporter made it possible to detect certain problems and to locate and correct bugs early on in the project. Finally, explanations of the processes for releasing and delivering the application were given.

The results presented in chapter 4 show that Electron was well suited for Amanote since the application works correctly on both Windows and macOS. Although some bugs occurred, no major problem was caused by the chosen technologies. Nevertheless, JavaScript showed its limits when performing intensive tasks, which reveals that it may not be adapted for applications requiring a high performance on the client's side. Moreover, the results demonstrated that it was possible to develop a cross-platform client-side using free and open sourced technologies. In fact, the development costs were mainly for the code signing certificates and the data transfers for delivering the application.

The second part focused on the server-side, which required to be highly scalable and available. In order to fulfil such requirements, a serverless architecture using Amazon Web Services as a Cloud provider has been set up. This architecture proved to be beneficial since it made it possible to develop the back end quickly and without server management. Moreover, no problems have been encountered in production with this architecture. Until now, it has been automatically scalable and always available as it was required. Moreover, thanks to the Amazon Web Services' free tier and promotional credits offer, it has cost nothing so far.

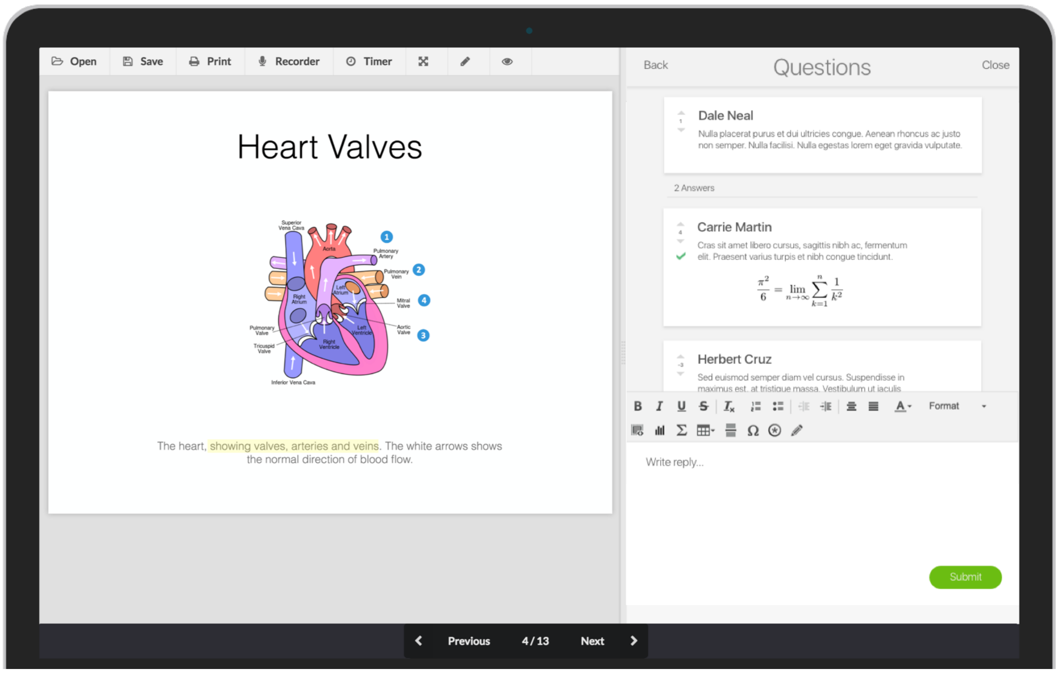
Finally, with more than 20,000 users registered to Amanote in seven months and with around 30% and 60% of daily active users, although some improvements still need to be made, theses analytic results confirm, in some way, that the main goal has been reached.

5.2 Future work

Regarding the future work, some features including the exportation to PDF, the highlighting system, the cloud storage, and mathematical formulas will be improved and optimised first. Then, the most requested features listed in Section 4.4 will be implemented. The note taking will be also sped up using auto-completion techniques based on the content of the slides and, if possible, the audio recorder.

Then, as there are numerous presentations in business, the application will be adapted to meet also the companies and conference needs. Especially, features capable to bring more interaction between the audience and the speaker will be added. For example, the possibility for the speaker to project his slides with Amanote and to share them in an instant with his public using a short code to enter in the app as well as a live chat allowing the public to ask questions.

To increase the virality and not force the audience to download the app before the presentations, a simplified version of Amanote will be ported as a web application (SaaS) allowing them to take notes and to interact without having to download the full application. The app will also be released on the platform stores to improve its visibility.



Heart Valves

The heart, showing valves, arteries and veins. The white arrows shows the normal direction of blood flow.

1
Dale Neal
Nulla placerat purus et dui ultricies congue. Aenean rhoncus ac justo non semper. Nulla facilisi. Nulla egestas lorem eget gravida vulputate.

4
Carrie Martin
Cras sit amet libero cursus, sagittis nibh ac, fermentum elit. Praesent varius turpis et nibh congue tincidunt.

$$\frac{\pi^2}{6} = \lim_{n\to\infty} \sum_{k=1}^{n} \frac{1}{k^2}$$

-3
Herbert Cruz
Sed euismod semper diam vel cursus. Suspendisse in maximus est, at tristique massa. Vestibulum ut iaculis

FIGURE 5.1: Prototype of the Questions/Answers feature.

Finally, features allowing more interactions between the users will also be developed. It will be possible, for instance, to merge notes with other users or to ask and answer questions on a specific slide as prototyped in Figure 5.1.

In order to be able to improve Amanote as much as possible, it is planned to raise 500,000 EUR in the course of September 2017. This will allows to hire developers but also to grow the company internationally.

References

[1] *android - What is the difference between cross platform app development and hybrid app development? - Stack Overflow*. http://stackoverflow.com/questions/32902009/what-is-the-difference-between-cross-platform-app-development-and-hybrid-app-dev. (Accessed on 05/16/2017).

[2] *AngularJS vs. ReactJS Comparison. What to Choose? | MLSDev*. https://mlsdev.com/blog/68-angularjs-vs-reactjs-comparison-what-to-choose. (Accessed on 05/21/2017).

[3] *AWS Device Farm: A service to test mobile apps on real devices | TO THE NEW Blog*. http://www.tothenew.com/blog/aws-device-farm-a-service-to-test-mobile-apps-on-real-devices/.

[4] *AWS User Authentication & Mobile Data Service | Amazon Cognito*. https://aws.amazon.com/cognito/. (Accessed on 05/29/2017).

[5] Jim Bird. *Building Real Software: Bugs and Numbers: How many bugs do you have in your code?* http://swreflections.blogspot.be/2011/08/bugs-and-numbers-how-many-bugs-do-you.html. (Accessed on 05/21/2017).

[6] *Case Studies & Customer Success - Amazon Web Services*. https://aws.amazon.com/solutions/case-studies/all/. (Accessed on 05/28/2017).

[7] *Cloud Computing Trends: 2017 State of the Cloud Survey*. http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2017-state-cloud-survey. (Accessed on 05/29/2017).

[8] *Desktop windows versions market share Worldwide | StatCounter Global Stats*. http://gs.statcounter.com/os-version-market-share/windows/desktop/worldwide.

[9] *Documentation | Electron*. https://electron.atom.io/docs/. (Accessed on 05/21/2017).

[10] *Essential Electron*. http://jlord.us/essential-electron/.

[11] *Global Infrastructure*. https://aws.amazon.com/about-aws/global-infrastructure/.

[12] *how many students in the world - Wolfram|Alpha*. http://www.wolframalpha.com/input/?i=how+many+students+in+the+world. (Accessed on 06/01/2017).

[13] *Interpreted language - Wikipedia*. https://en.wikipedia.org/wiki/Interpreted_language. (Accessed on 05/20/2017).

[14] *JavaFX CSS Reference Guide*. https://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/cssref.html#introlimitations. (Accessed on 05/21/2017).

[15] *JavaFX Documentation Home | JavaFX 2 Tutorials and Documentation*. http://docs.oracle.com/javafx/2/. (Accessed on 05/17/2017).

[16] *JavaFX - Wikipedia*. https://en.wikipedia.org/wiki/JavaFX. (Accessed on 05/17/2017).

[17] *Mac OS X Snow Leopard - Wikipedia*. https://en.wikipedia.org/wiki/Mac_OS_X_Snow_Leopard.

[18] *Microsoft PowerPoint - Wikipedia*. https://en.wikipedia.org/wiki/Microsoft_PowerPoint. (Accessed on 05/25/2017).

[19] *Native vs Cross-Platform App Development: Pros and Cons of PhoneGap, Titanium, and Xamarin - DZone Mobile*. https://dzone.com/articles/native-vs-cross-platform-app-development-pros-and. (Accessed on 05/16/2017).

[20] *npm*. https://www.npmjs.com/.

[21] *npm now the largest module repository*. http://alexandros.resin.io/npm-now-the-largest-module-repository/.

[22] Daniel M. Oppenheimer Pam A. Mueller. “The Pen Is Mightier Than the Keyboard”. In: (2014).

[23] Pearson, ed. *Student Mobile Device Survey 2015*. Pearson, 2015.

[24] *PowerPoint usage and Marketshare - Infogram, charts & infographics*. https://infogr.am/PowerPoint-usage-and-Marketshare. (Accessed on 05/25/2017).

[25] *Quick Start | Electron*. https://electron.atom.io/docs/tutorial/quick-start/.

[26] *Scripting Languages for AWS Lambda: Running PHP, Ruby, and Go | AWS Compute Blog*. https://aws.amazon.com/blogs/compute/scripting-languages-for-aws-lambda-running-php-ruby-and-go/. (Accessed on 06/07/2017).

[27] *Stack Overflow Developer Survey 2016 Results*. https://insights.stackoverflow.com/survey/2016.

[28] StatCounter Global Stats. *Desktop Operating System Market Share Worldwide*. http://gs.statcounter.com. Apr. 2017.

[29] *The Average App Loses 77% Of Its Users In The First Three Days - ARC*. https://arc.applause.com/2015/06/23/app-retention-rates-2015/. (Accessed on 05/25/2017).

[30] *Timeline of Amazon Web Services - Wikipedia*. https://en.wikipedia.org/wiki/Timeline_of_Amazon_Web_Services. (Accessed on 05/29/2017).

[31] *Version Control Systems: Git, SVN, Mercurial, Bazaar*. https://webinerds.com/version-control-systems-keep-your-code-in-order/.

[32] *What Is JavaFX? | JavaFX 2 Tutorials and Documentation*. http://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm. (Accessed on 05/17/2017).

[33] *What is native app? - Definition from WhatIs.com*. http://searchsoftwarequality.techtarget.com/definition/native-application-native-app. (Accessed on 05/20/2017).

[34] *Why Google has 200m reasons to put engineers over designers | Technology | The Guardian*. https://www.theguardian.com/technology/2014/feb/05/why-google-engineers-designers. (Accessed on 06/07/2017).