

# App Maintenance Manual

Last Revised October 3rd, 2019

---

<b>Introduction</b>	4
Anatomy of the App	4
Why Do We Need to Update the App?	5
<b>Unzipping the App Contents</b>	5
<b>Using the Command Line</b>	5
Opening the Command Line	6
Enabling Developer Tools	7
More Resources	7
Other Command Line Features	8
<b>Software Preparation</b>	9
Microsoft Visual Studio Code	10
Homebrew	10
Node.js	10
Apache Cordova	11
Software for Apple	11
XCode	11
ios-deploy	12
Cocoapods	12
Software for Android	13
Java	13
Gradle	13
Android Studio	13
Android SDK's	14
Checking Software Installation	19
<b>Before You Update</b>	20
The File Structure	20
CleanCopy.zip	21
EditingFolder	21
El Caño Tour	21
KeysAndUploads	21
Logs	21
openIOS	21

RawImages	21
Scripts	22
Shelf	22
SigningInfo	22
tour	22
UploadsAndStoreInfo	22
Shell Scripts	22
<b>Updating the App</b>	<b>23</b>
Changing the Version Code	23
Updating the Native Code	24
Publishing the Android Version	24
Opening Android Studio	25
Generating a Signed App Bundle	28
Uploading to the Android App Store	35
Android Build Troubleshooting	42
<b>Modifying the App</b>	<b>42</b>
Looking at the code	42
Editing the Donation Link	42
Changing out Images	43
Editing the Content (HTML)	44
Editing the Style (CSS)	44
Looking at the App	46
<b>Directing People to the App</b>	<b>49</b>
Android App Store	50
Progressive Web App	51
<b>Managing App Store Listing</b>	<b>51</b>
Demonetization and Price Changes	51
Android Price Changes	52
Apple Price Changes	53
App Listing Changes	53
Android App Listing	53
Generating More Screenshots	55
<b>Reverting to a Progressive Web App</b>	<b>56</b>
Switching on GitHub Pages	56
Modifying the PWA	59
Managing the Serviceworker	60



## Introduction

Fundación El Caño's app was originally coded by a team of four students from Worcester Polytechnic Institute as their Interactive Qualifying Project. This manual describes how to update and modify the app to maintain it.

This manual is designed for a Mac computer, as this was the computer the IQP team originally used, and this is the only type of computer that can update an iPhone app as of the writing of this manual.

Furthermore, this manual was designed originally for a user account with Admin privileges. We have done our best to adapt the manual and the updating and installing processes to work on a user account without Admin privileges.

The original language for the manual is English. If the text is translated, be careful not to translate the commands that appear in monospace code snippets, such as the following:

example monospace code snippet

Additionally, some folder names are based on the English version of Apple's default file system. On a Spanish Mac, these names may need to be changed (e.g. User may need to be Usario).

It may be easier to view this manual without Print Layout turned on, if the medium through which you're reading it allows this.

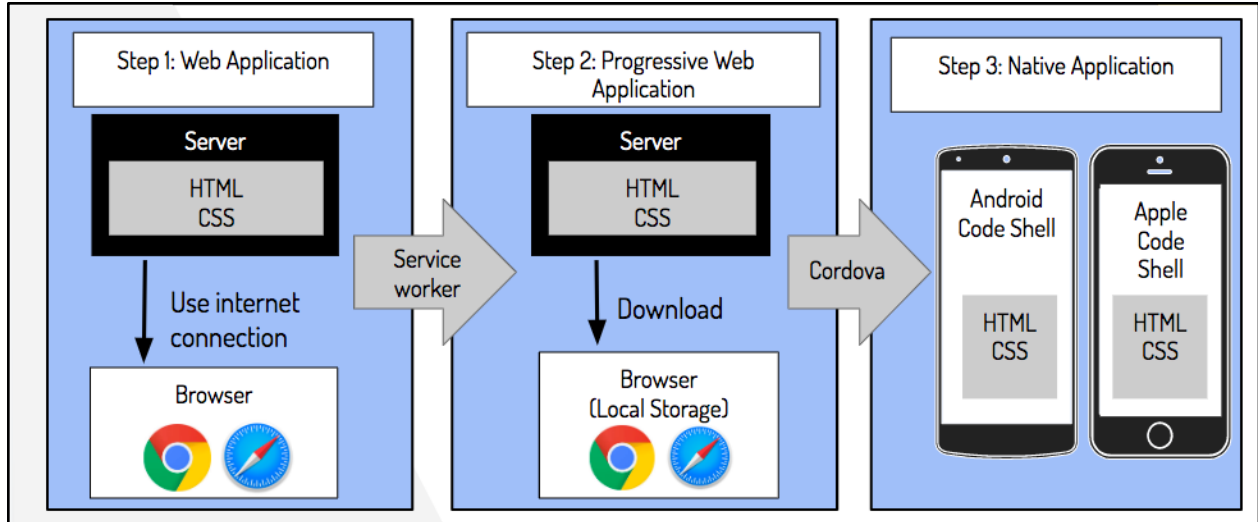
## Anatomy of the App

The app has an interesting anatomy to it, and understanding this anatomy on some level will help you work with the source code. At the core, the application is a website made up of HTML and CSS files. The content that appears in the application is written in the HTML files, and the content is styled using CSS files. These files together create a website that can be [hosted on GitHub Pages](#).

A service worker file was created in Javascript to accompany the HTML and CSS files. The purpose of the serviceworker file is to create the browser cache that allows all the files to be downloaded into the local storage of a browser. The serviceworker source file is also hosted on GitHub Pages alongside our HTML and CSS files, and it installs itself in the browser as a serviceworker. These files together create the progressive web application.

From there, we used Cordova to create the native application. Cordova is a software that allows cross-platform development of applications from standard web languages. The software writes native code shells around the HTML and CSS files to make a native application that can run on Apple and Android operating systems. This software allows the user to avoid coding in the native language of each mobile platform.

A diagram that details this process is shown below.



## Why Do We Need to Update the App?

About once a year, Apple and Android release a major update to their mobile Operating Systems. All apps on their respective app stores need to update their software to conform to the new Operating System. This would normally mean that we need to change a lot of small and very technical things within the app's code. Anticipating this, we built the app through a third-party software called [Cordova](#).

## Unzipping the App Contents

If you have been provided with a Zip file that contains the app and other things, unzip it in your root directory. If the folder is already in your root directory, skip this step.

You can find your root directory on Finder by opening your downloads folder and pressing Command + Up Arrow repeatedly until further presses have no effect. Then navigate forward through your files on Macintosh HD to /Users/(your user name)/. Place EICañoApp.zip inside the folder with your username, then unzip the file by double-clicking it. Updating the app will only work if the EICañoApp folder is in the right place. For more information on how to use Finder, [see here](#).

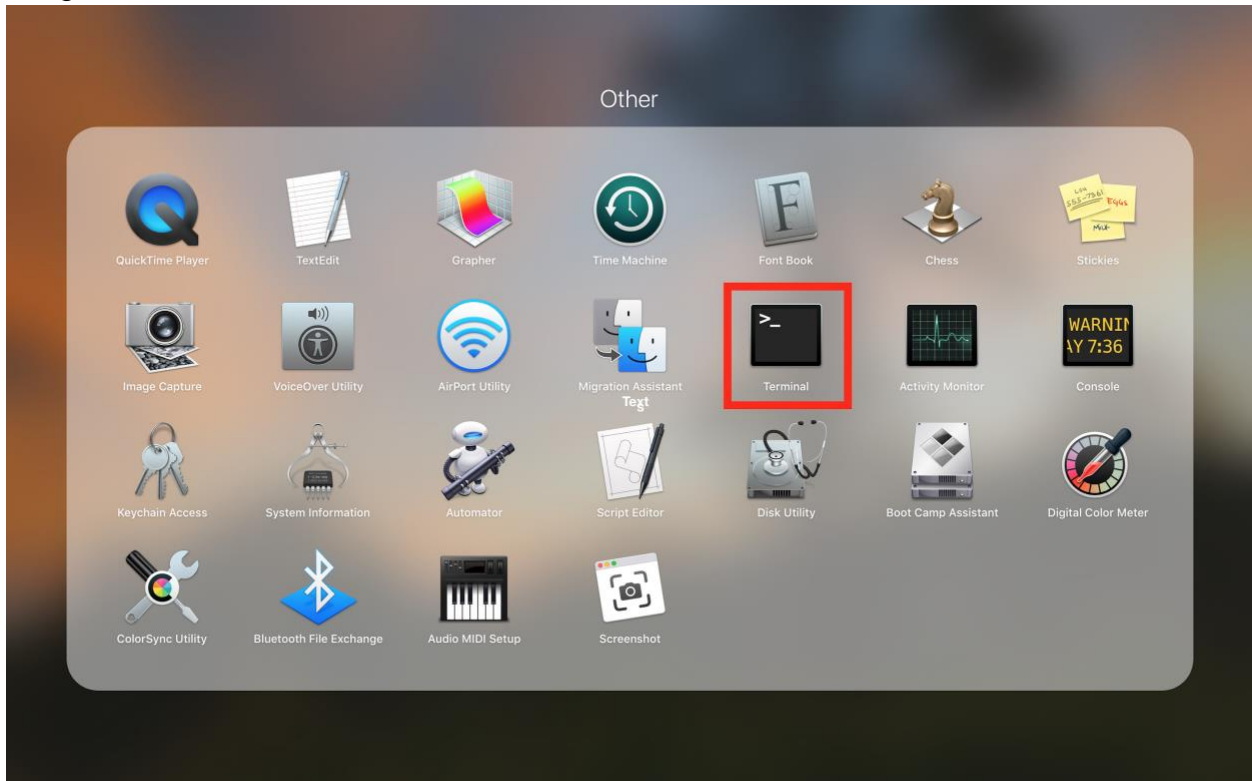
## Using the Command Line

The Command Line is a way to interface with your computer on a low level using just textual commands. You may also hear it called the terminal, the command prompt,

or the command shell<sup>1</sup>. Using the command line is crucial to updating or modifying the app.

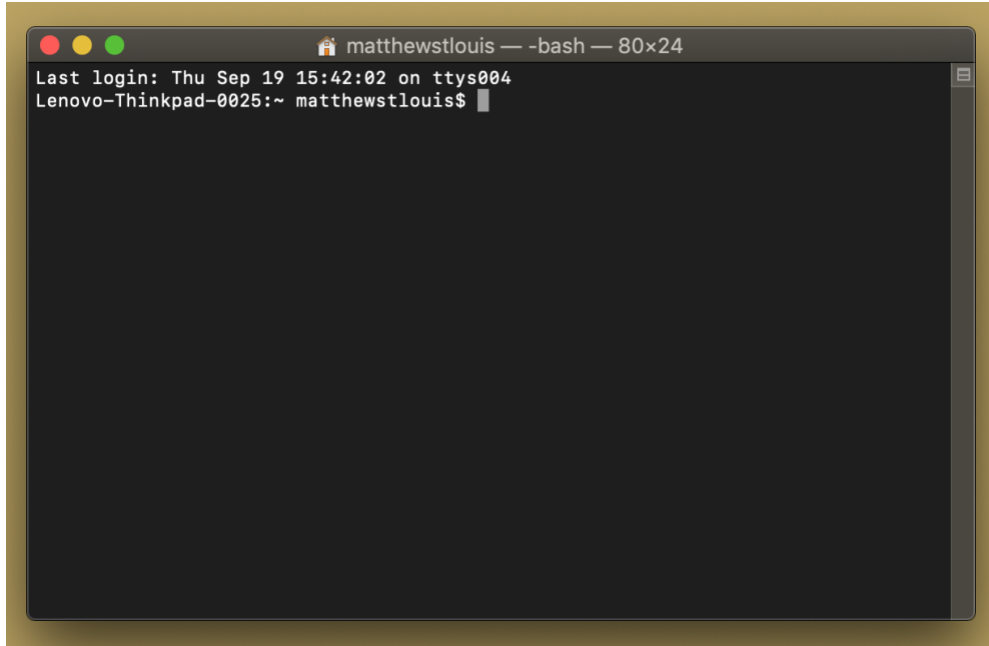
## Opening the Command Line

To get to the command line, you can find it in your launchpad, your list of applications, or using Spotlight Search. The icon for the Command Line is shown in the image that follows.



If you've opened the correct program, you should see something like the following image.

<sup>1</sup> The difference used to be that the Terminal was the physical machine you would sit at to type commands, the command line or command prompt was the line where you would type the command, and the shell was the software that would receive, interpret, and execute the commands.



To see the directory (folder) you're working from, type **pwd** and hit enter. To see all of the folders and files in your directory, type **ls** and hit enter. To enter another folder, type **cd**, then a space, then the name of the folder, then hit enter. You can use **pwd** or **ls** to see where you are<sup>2</sup>. These are a few sample commands to get used to the command line.

## Enabling Developer Tools

To enable the developer tools from the command line, type the following command and press enter:

```
xcode-select --install
```

You should only need to do this once per Mac.

## More Resources

The particular command shell that Apple uses is called Bash, and these commands will not necessarily work on other shells. You do not really need knowledge of bash beyond what this manual explains in order to update the app.

However, if you want to get more familiar with Bash commands, you can check out some of these tutorials and resources. Note that many of them are for Linux or Unix, not Posix (Apple), so some of the commands may not carry perfectly, but for the most part, they are the same.

<sup>2</sup> If you're curious, *pwd* is "print working directory," *ls* is "list," and *cd* is "change directory."

- [Short Video](#)<sup>3</sup>
  - Be careful with the `rm` command, as this can delete files that are necessary to the app
- [Longer Video](#)
- [Short Tutorial](#)
- [Longer Tutorial](#)
- In-Depth [Written Tutorial](#) with a Table of Contents
- Type “`man man`” into the terminal to see the manual page for how to use the manual.
  - [How to use the Man pages](#)
  - On the man pages, you can use `j` and `k` to scroll (`j` is down; `k` is up). Press `q` to exit the man page. Press `h` for help on navigating the man page. Press `q` to leave the help view.
  - Type “`man`” followed by a space and another command to see how to use the command.
- Type “`man bash`” for a confusing and technical explanation of how to type shell commands.

## Other Command Line Features

Another notable feature of the command line on Mac is that you can open a new tab with `Command + T`, and you can open another window with `Command + N`. The new tab or window will start in the directory of the last tab or window in focus. Different tabs can be in different directories, which can be handy.

In case you didn't read the footnote, don't use `vi` or `vim` if something asks you to, as these are complicated text editors with steep learning curves. Instead, follow the advice in [Modifying the App](#).

Using **`sudo`** before a command will make you execute the command as a superuser rather than as an admin. In many cases, this is harmless and sometimes necessary. However, using `sudo` for some tasks may damage your computer (e.g. `rm`, `mv`, `cp`), and it may sometimes allow viruses additional access to your machine. As long as you stick to this manual for using `sudo`, you should be fine. **`sudo`** may also be necessary for gaining more privileges for users without Admin privileges, especially when put in front of commands like **`chown`** to change the ownership of a directory.

<sup>3</sup> When she types “`vi`,” the equivalent text editor for mac would be “`vim`.” `Vi` is an older version of `Vim` for certain linux computers. `Vim` is a very complicated text editor with a steep learning curve, so I would not recommend using it. If you need to edit a file's text, see the section on [modifying the app](#).



Another special feature of the command line to be aware of is escape characters. Certain characters, such as a new line or a space, have no way to be typed directly into a command because they have a different assigned meaning (a space separates tokens of the command, and a new line ends the command). The workaround for this is escape characters. Most escape sequences start with a backslash (\). Some examples of escape characters are shown as follows:

- \n is a new line
  - On the terminal, this allows multi-line commands
- Backslash + Space is a space
  - This is **very** important for file paths, especially because the app is in a folder called “El Caño Tour,” which needs to be typed as El\ Caño\ Tour
- \" is a quote inside of a string
- \\ is a backslash
- See [this reference](#) for more examples if needed
- See [this reference](#) if you want another explanation of escape characters

If you need to see the contents of a log or a file, you can use the **cat** command followed by the file name. You can use the formula **cat FILE\_NAME | less** to see the contents in a view separate from the rest of the commands you have entered (use **q** to escape the less view).

File paths are something you will come across often. A file path can either be absolute or relative. An absolute file path usually starts with a / and starts at the root of whatever file system you are in. In this case, it would be the root of the computer’s file system. A relative file path starts without a slash. It may start with the name of a folder within your working directory, a . to represent your working directory, or a double dot (..) to represent the directory above your working directory. You may up by multiple directories with consecutive double dots (../..).

## Software Preparation

In order to modify or update the app, you will need access to a computer with specialized software. This section describes what software to install and how to install it. To skip ahead to [updating the app, click here](#).

Most of these steps are fairly quick, but some downloads and installations can potentially take hours depending on your network speed. The longest steps are listed here:

- [XCode](#)
- [Android Studio](#)
- [Android SDK's](#)

This tutorial was originally written for a user account with Admin privileges, which was the case for the original laptop used to compile the app. The computer at El Caño's office at the time of writing this does not have admin privileges, so some modifications have been made, and you may occasionally need to edit the permissions of certain directories with the **chown** command. For more information on how this command works, type **man chown** into the command line.

## Microsoft Visual Studio Code

To update the app, you're going to need to change one line of code. To do this, you will need a text editor. If you're unfamiliar with a text editor, it's like a word processor, except it works with text files and can only encode characters (no bolding, font size, italics, etc.).

You could use just about any text editor, but I would recommend [Microsoft Visual Studio Code](#). Note that Microsoft Visual Studio Code and Microsoft Visual Studio are two distinct pieces of software. Do not download Microsoft Visual Studio.

## Homebrew

[Homebrew](#) is a package manager for Mac. A package manager is a program that has powerful access to your computer to manage low-level files to facilitate the installation of third-party software. To download homebrew, paste the following command into the command line:

```
/usr/bin/ruby -e "$(curl -fsSL \n\nhttps://raw.githubusercontent.com/Homebrew/install/master/install)"
```

If you are unfamiliar with the command line, [see the section above](#).

One notable feature of Homebrew is that it will not work in conjunction with the command **sudo**. **sudo** is the keyword to execute a command as the superuser. This means that sometimes Homebrew may ask you to change the permissions for certain directories if your user profile does not have Admin privileges. These commands will be in the error messages that Homebrew displays. Please type them and try the installation again if you see any.

## Node.js

Node.js is the software through which Cordova runs. Download it [here](#). Choose the installation that is recommended for most users.

## Apache Cordova

Apache's Cordova software is the thing that wraps our web pages in a native shell to let the app run on smartphones. To download Cordova using the Node Package Manager, run the following command on the command line:

```
sudo npm install -g cordova
```

This command will prompt you to enter your password, as will any command preceded by **sudo**. This is necessary to allow Cordova to access certain folders within the Node Package Manager. You may also need to change the ownership of some files with the **chown** command if your user account on the computer does not have Admin privileges. If you see error messages, read them to look for the recommended command to resolve the issue.

Once you have Cordova installed, you will need to install a few pieces of software to develop for Apple and Android. Run the following:

```
sudo npm install -g ios-deploy
```

## Software for Apple

This is the software that Cordova needs to make an Apple app. For assistance, see Cordova's page on [Installing Apple Requirements](#). If that link is outdated, look for the iOS platform guide on [Cordova's website](#).

### XCode

XCode is Apple's app development platform. Download it from the [Mac App Store](#). If you don't have a Mac, you won't really be able to upgrade the iPhone app. Once you download this, you will have to open XCode and accept the terms and conditions before you can install further software for updating the iPhone version of the app. After you accept the terms and conditions, XCode will download more Software Developer Kits (SDK's).

There may be an error where your computer does not recognize that XCode has been installed even after you have accepted the terms and conditions and downloaded the rest of the SDK's. You will only see this error when installing the next steps. The error will say something about how some developer directory points to the command line tools, not XCode. If you get this error, [this webpage](#) describes some potential solutions and causes. Try opening XCode and accepting the terms and conditions (this will start more downloads). Then make sure that XCode is in the Applications folder. If you have done both of these, try the following command to resolve the issue:

```
sudo xcode-select -s /Applications/Xcode.app/Contents/Developer
```

## ios-deploy

ios-deploy is a set of command line tools that can build ios apps from the command line. Cordova uses it to help write the native code shell for Apple. You cannot install ios-deploy until you have installed XCode. You can use the command below to install ios-deploy. The source code is on [GitHub](#).

```
npm install -g ios-deploy
```

If this does not work, and if you get an error message that XCode has not been installed, try [the fixes above](#).

If you get an error message about a not having permissions to edit a certain folder or that you should set a certain flag because of your MacOS installation, try installing with the command below instead:

```
sudo npm install -g --unsafe-perm=true ios-deploy
```

## Cocoapods

[Cocoapods](#) is something Apple uses. I'm not sure exactly what it does, but it feels similar to how Android uses Gradle. Cocoapods will only install if you have already installed XCode and [if your command line recognizes that you have installed XCode](#).

You can install Cocoapods with the following command:

```
sudo gem install cocoapods
```

Once it's installed, use the following command to get it ready:

```
pod setup
```

If this command does nothing or leaves the terminal hanging for longer than about 20-30 minutes, you can troubleshoot with [this website](#). As the website explains, the following commands may be able to clear up the installation:

```
mkdir -p ~/.cocoapods/repos  
git clone https://github.com/CocoaPods/Specs \  
~/.cocoapods/repos/master
```

This process may take a while. If you want to open another tab within the terminal while you wait, use Command T.

For more information and to see if the Apple Software installation worked, verify your installations with [cordova requirements](#).

## Software for Android

This is the software that Cordova needs to make an Android app. For assistance, see Cordova's page on [Installing Android requirements](#).

### Java

Java is a programming language and development kit that Android uses intermediately for its development environment. Java is not proprietary to Android. To download with [Homebrew](#) (recommended), use the following command:

```
brew cask install homebrew/cask-versions/adoptopenjdk8
```

Unfortunately, java may be uncooperative, especially if you have installed Java or JDK on your computer previously. Here are some potential solutions to potential problems you may encounter:

- [How to Manually Install Multiple Versions of Java](#)
- [Java 11 package javax.xml.bind does not exist \[duplicate\]](#)
- [Failed to install android-sdk: "java.lang.NoClassDefFoundError: javax/xml/bind/annotation/XmlSchema"](#)

As a last resort, to install Java manually, use [Oracle's website](#). The reason this is a last resort is that Homebrew can easily manage and uninstall anything it originally installed, but manually uninstalling Java downloaded manually can prove tricky. If you are going on Oracle's website, get the Mac x64 version of Java SE Development Kit 8u221 (or whatever is on that page at the time).

### Gradle

Gradle is a tool that Android uses to manage packages and to help interface with your computer. To install Gradle using [Homebrew](#), enter the following command:

```
brew install gradle
```

If you run into problems installing Gradle, and if you have already installed Homebrew correctly, then see [Gradle's website](#) for assistance.

### Android Studio

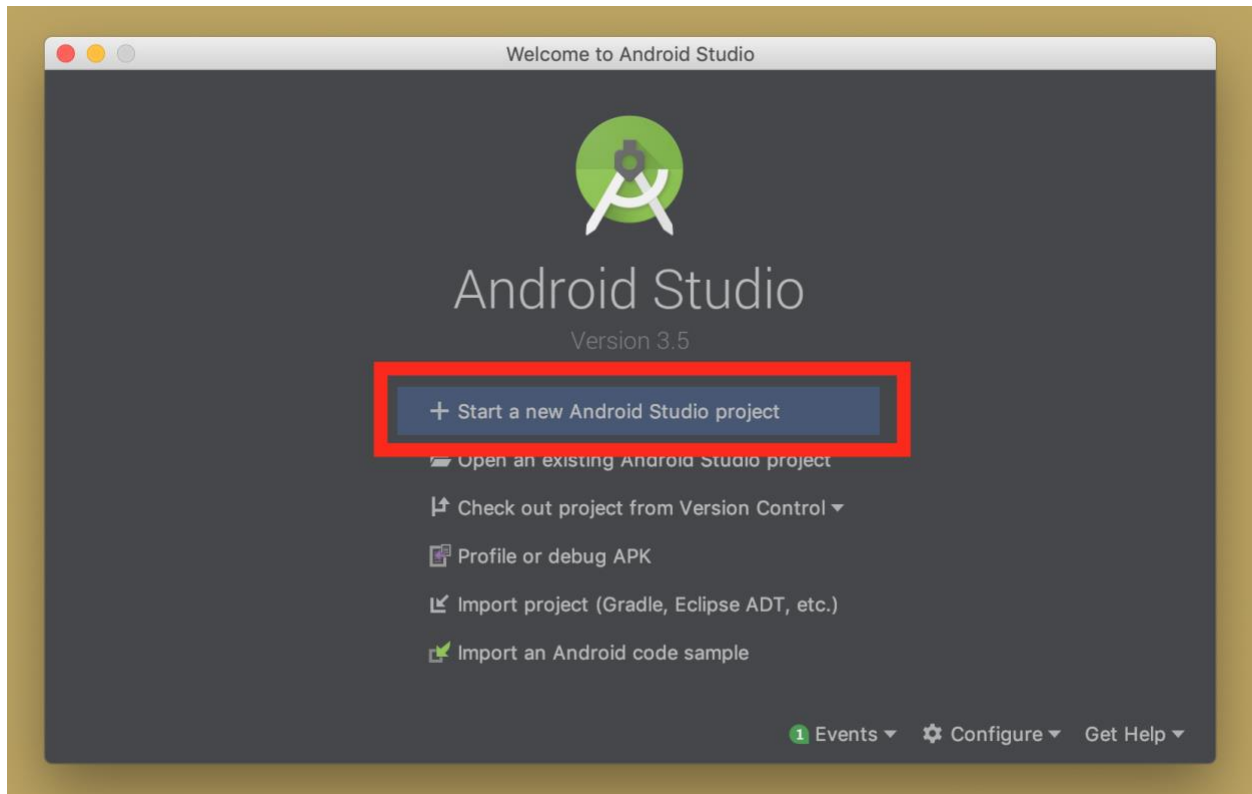
Android Studio is the development environment for Android apps. To download Android Studio, [visit their website](#). Note that the initial download is only a fraction of the installation process for this step. Once you open Android Studio, the program will download other resources and dependencies it needs to operate. This may take multiple hours, depending on your download speed.

## Android SDK's

The Software Development Kit (SDK) installation is one of the trickiest and longest steps in this whole process. It can take tens of gigabytes.

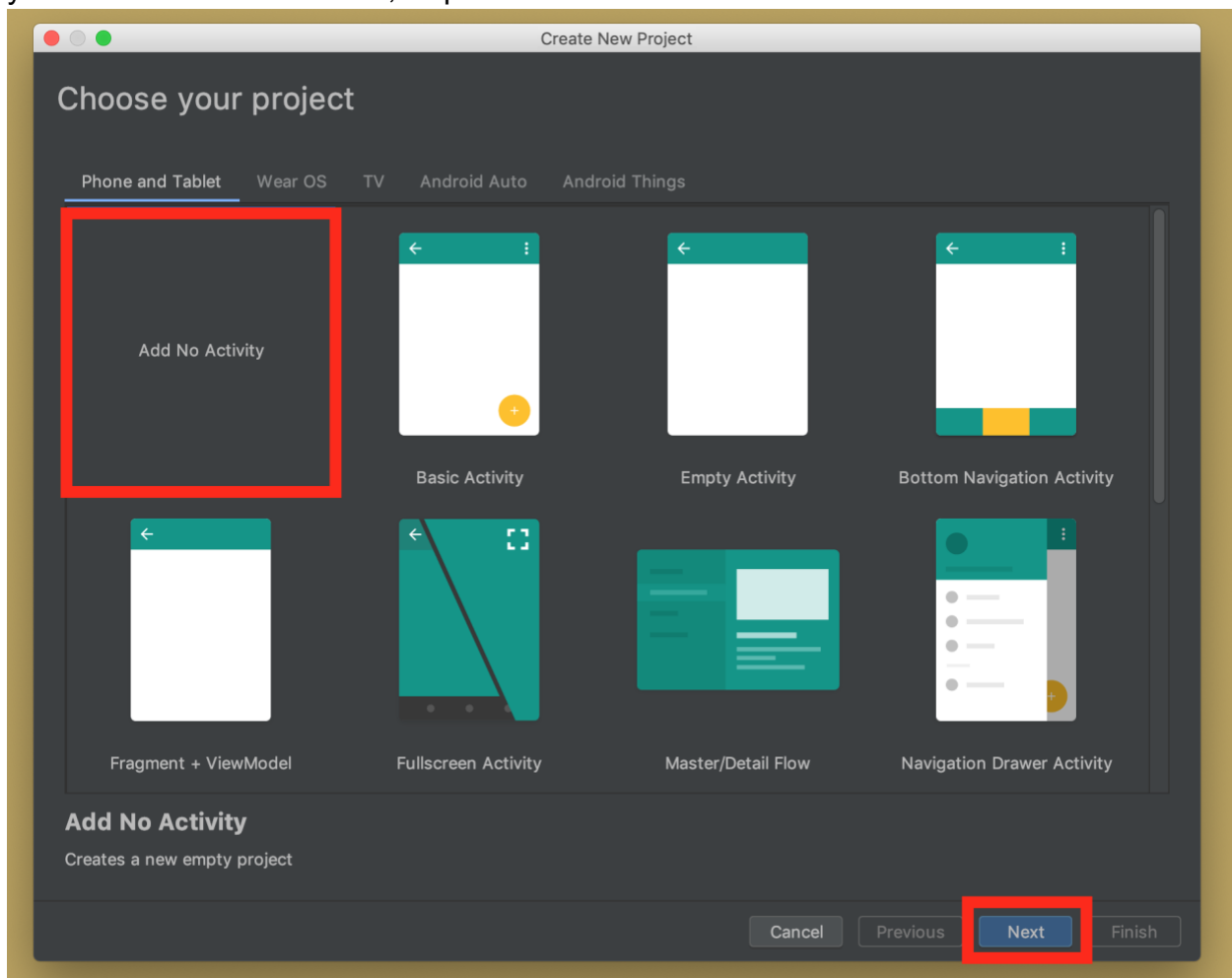
The first step to installing Android SDK's is to open Android Studio. However, to open Android Studio, you will need to create or open an Android Project. This part of the manual will describe how to create a new project, which you can delete later. A [later part of the manual](#) will describe how to open the existing Android Project. Skip to page 13 if you have opened Android Studio with Fundación El Caño's App.

When you launch Android Studio, the first screen you will see will look something like the screenshot below.

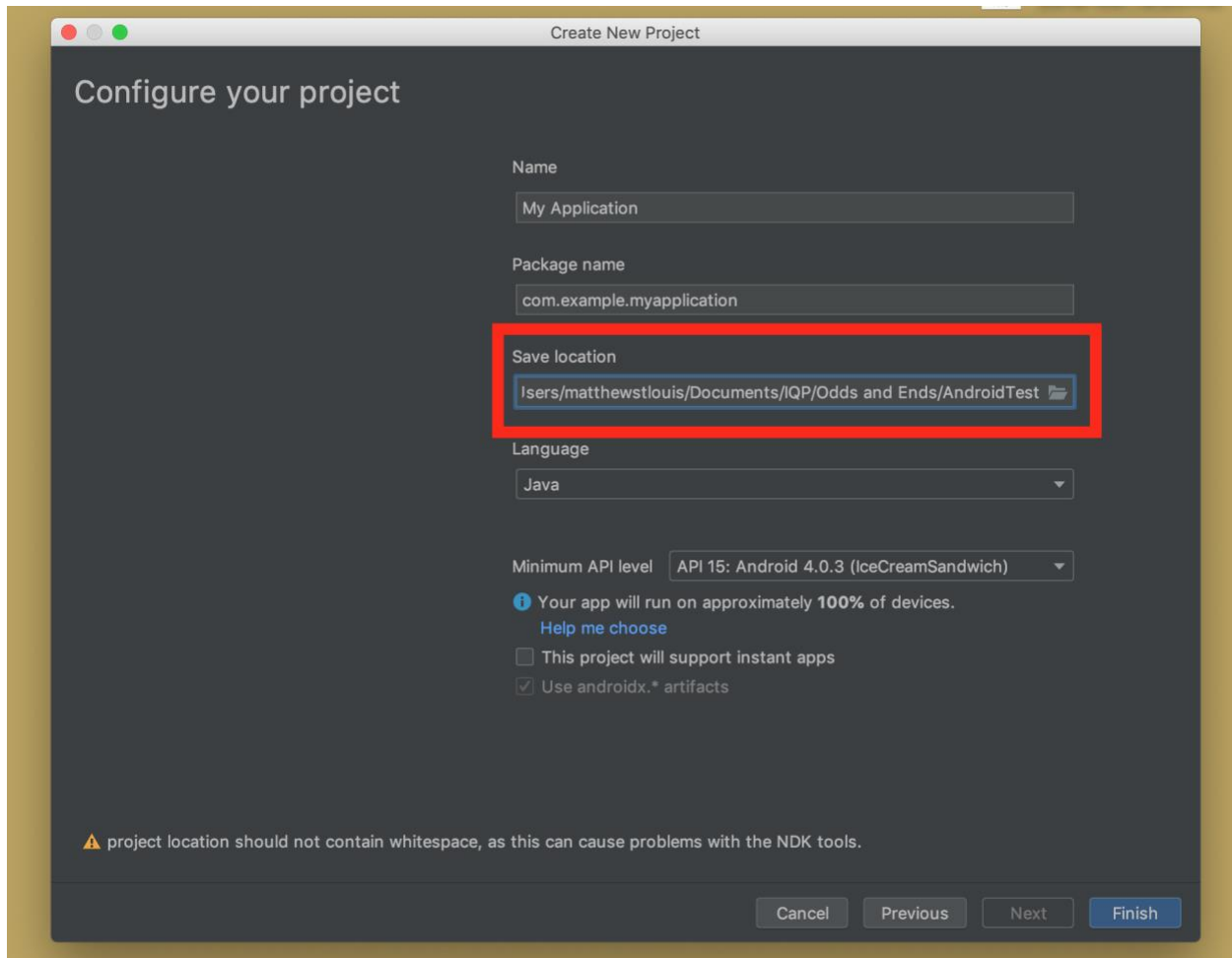


Select the option to create a new project as shown.

Once you do that, you will either see a screen like the one below, or you may be brought straight to a screen about downloading Software Development Kits (SDK's). If you see a screen on SDK's, skip ahead to the SDK menu later in this section.



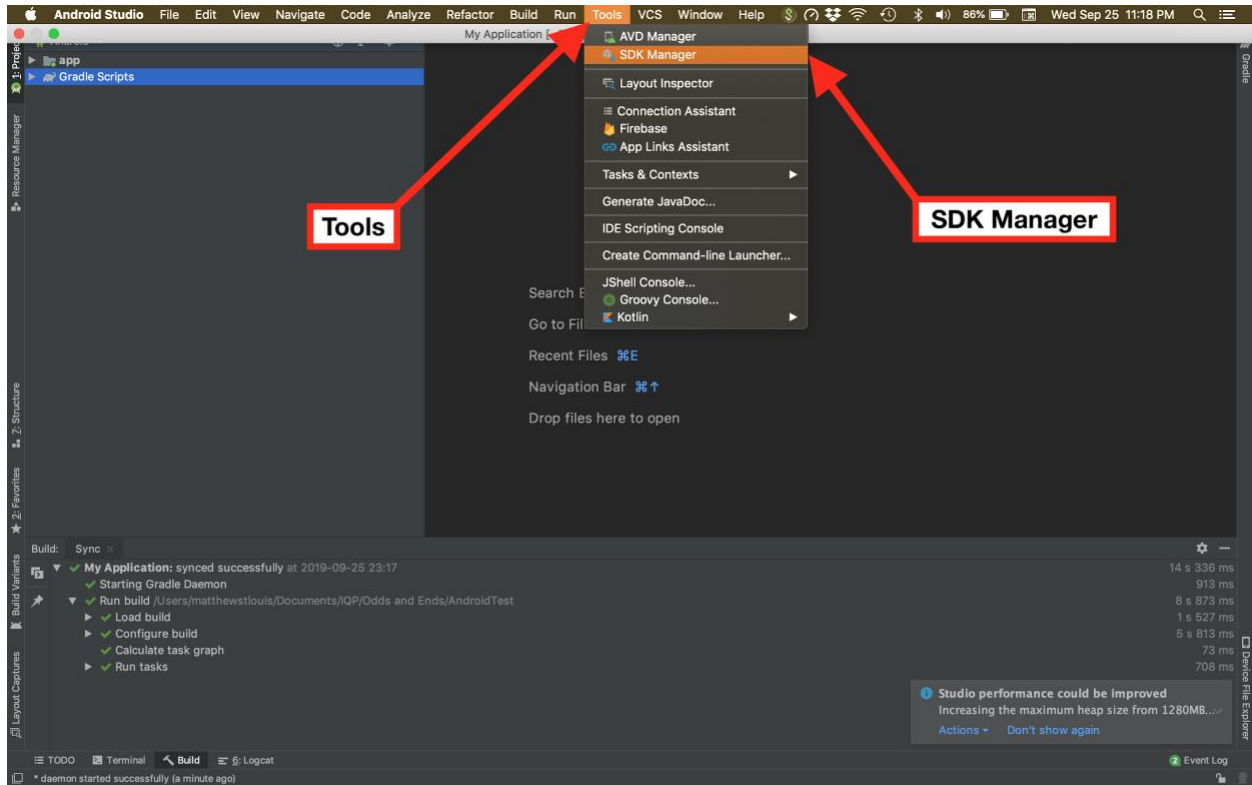
If this is the screen you see, select that you want to add no activity, then hit "Next." You will then be asked to configure your project, as shown below.



This step might be important. There will be a directory listed in the Save Location, as shown above. The default directory will likely be fine, but make sure it is not being saved in the EICañoApp directory or one of its subdirectories. You can make a note of the directory where this dummy project is being saved if you want to delete it later. Press “Finish” in the lower right when you are done.

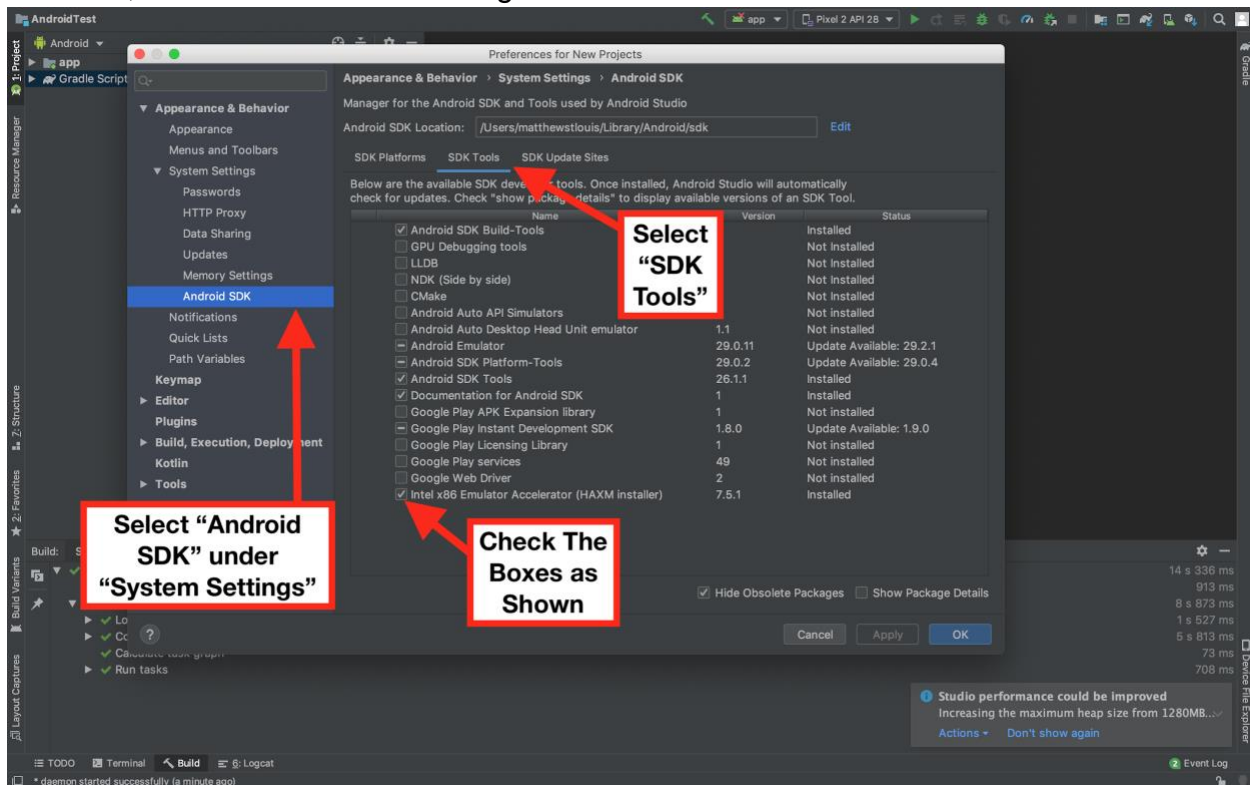


This should open a new window. Welcome to Android Studio! From here, you will open the SDK manager from the Tools menu, as shown below.



If you do not see this option under “Tools,” you should be able to access the SDK manager from settings. Press Command , to open the settings, or press “Android Studio” from the menu at the top and select preferences. An image of the system preferences is on the next page of this manual. There should be a series of dropdowns on the left. If “Appearances and Behavior” is not expanded, expand it. If “System Settings” is not expanded, expand it. Under System Settings, you should see “Android SDK.” If this is not the case, type “SDK” into the search bar for the system preferences.

This should bring up a new window. Make sure that Android SDK is selected on the left, that SDK Tools is selected in the top tab, and that the appropriate boxes are checked, as shown in the next image.



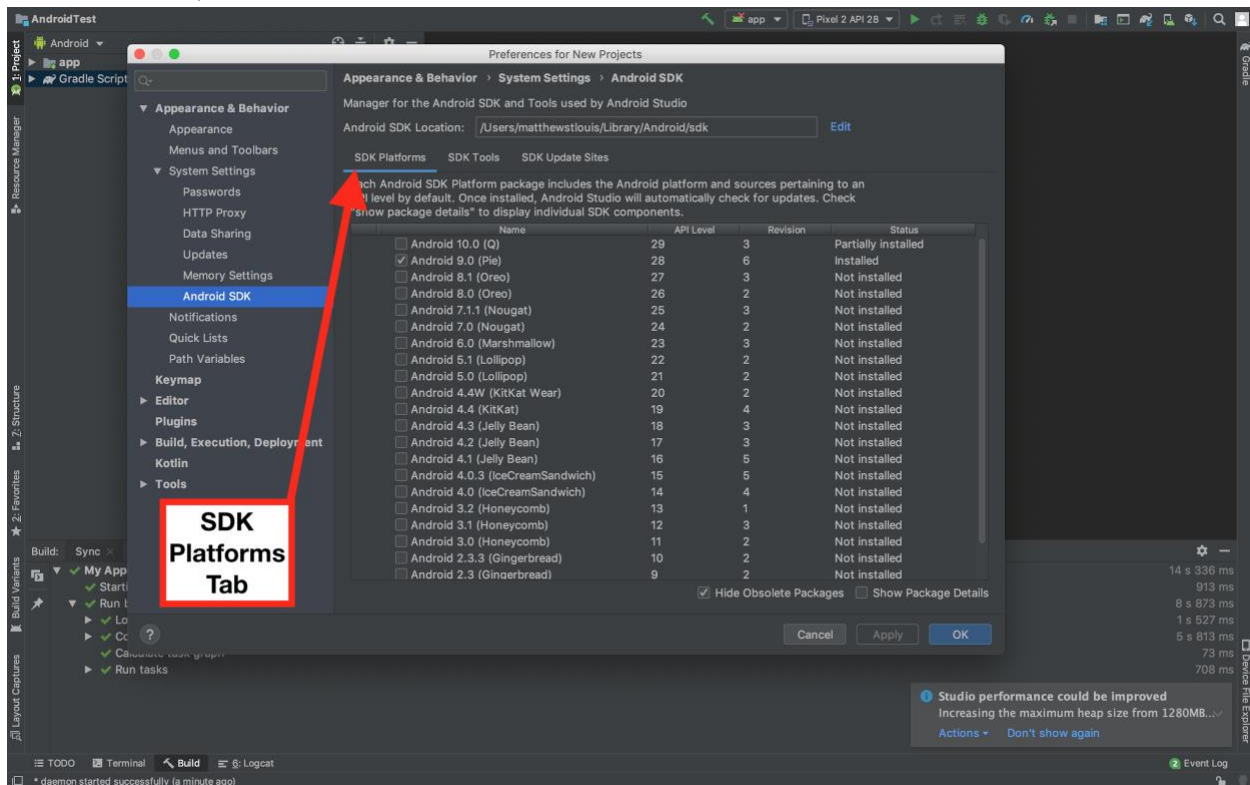
The SDK's checked are as follows:

- Android SDK Build-Tools
- Android Emulator (Optional)
- Android SDK Platform Tools
- Android SDK Tools
- Documentoxy for Android SDK (Optional, but recommended)
- Google Play Instant Development SDK
- Intel x86 Emulator Accelerator (HAXM installer)

In the screenshot used, I'm displaying which SDK's we needed when we first designed this app. However, depending on when you are reading this, the required SDK's may have changed. To see which SDK's you may need, [ask Cordova](#).

Once you have selected the appropriate software, select "Apply," then "OK." This will begin downloading the SDK's. You may need to wait a few hours for this to complete.

When you are ready, install the appropriate SDK platforms. You can do this by bringing up the same window as for the SDK tools, except instead of selecting the tab for SDK tools, select the tab for SDK Platforms as shown below.



You should only really need one SDK platform. The one with the highest or second-highest API level is preferable. The original package used was API level 28. If you want to see more specifics within the packages, select “Show Package Details” from the lower right corner of the dialog box. This expands each SDK Platform. The ones I have installed from this menu are as follows:

- Android 10.0 (Q)
  - Android SDK Platform 29
- Android 9.0 (Pie)
  - Android SDK Platform 28
  - Sources for Android 28
  - Intel x86 Atom System Image
  - Google Play Intel x86 Atom System Image

Once you have selected the desired SDK’s, select “Apply” and “OK.”

## Checking Software Installation

To see if you have installed the software correctly, open the [Command Line](#) and use the following command to enter the Cordova Project directory:

```
cd ~/ElCañoApp/El\ Caño\ Tour/
```

Type the following command:

```
cordova requirements
```

If you see any error messages, try entering the command that the message suggests, or try Googling the error message to resolve the problem.

## Before You Update

As previously explained, the app needs to be updated about once a year to keep it compliant with the latest mobile operating systems. We have tried to make this process as painless as possible.

First, this manual discusses the file structure we have used and which folders you should and should not modify. Then, we discuss how to use shell scripts.

## The File Structure

To work with the files that contain the app, it may help to understand the file system we used. The outer folder is called “ElCañoApp,” and this folder should be inside your root directory (the one with your username). You can check this by opening the command line and typing **cd** and **ls**. If you see ElCañoApp listed, the folder is in the right place. If this is not the case, see [this section](#) on how to unzip the folder into the right directory.

There are several folders and files inside ElCañoApp. The ones you **can edit** are listed below in green. The ones you **can read** are listed below in blue. The ones you **should not edit** are in red. These recommendations are not enforced, so please be careful.

- CleanCopy.zip
- EditingFolder
- El Caño Tour
- KeysAndUploads
- Logs
- openIOS
- RawImages
- Scripts
- Shelf
- SigningInfo
- tour
- UploadsAndStoreInfo

Each folder and file is described below.

## CleanCopy.zip

**CleanCopy.zip** should be the website source code as the developers originally left it. Feel free to unzip this to get a fresh copy of the code if anything goes wrong.

## EditingFolder

**EditingFolder** starts off as a copy of the source code from CleanCopy. This is the folder that you can edit to change the content of the Native App. Any changes made here will affect the native app's next update, so consider editing a copy of these files for testing.

## El Caño Tour

**El Caño Tour** is the Cordova Project. Some of it can technically be edited safely, but changes will likely either be overwritten or possibly damage the app. Please do not edit any files within the El Caño Tour directory.

## KeysAndUploads

**KeysAndUploads** contains the private keys used to sign the apps and the app bundles to upload to the app stores. You will be asked to upload the bundles to the Android App store, but please do not touch the key files or the keystore files.

## Logs

**Logs** contains some of the logs from updating the app. If something in the build goes wrong, the information may be recorded in one of the logs. The files in this directory get regenerated with every update, so the logs will only display the status of the most recent update.

## openIOS

**openIOS** is a short script I wrote to open the Apple text editor because it's slightly easier than writing out the steps to open the project manually. It should work every time.

## RawImages

**RawImages** is a folder of some of the image assets used in the app, specifically those made with the GNU Image Manipulation Program (GIMP). These images are here in case the foundation would like to edit the images from the app in any way. To edit the .xcf versions, you will need to download GIMP at [their website](#) or with the following command:

```
brew tap caskroom/cask && brew cask install gimp
```

## Scripts

The **Scripts** folder contains the scripts you will run to work with the app. A script is an executable file that enters terminal commands for you. Rather than using a script, you could potentially enter each line into the command line yourself. If a script fails, this may be a good option for debugging.

Running any script will modify some of the files in this file tree, so be careful not to run a script when you don't mean to. To run a script from the command line, you can type `./` before the script's name then hit enter.

## Shelf

The **shelf** folder is where we put a few loose files that aren't supposed to be edited. Most of these files get copied elsewhere when the scripts execute. Please don't edit the shelf folder.

## SigningInfo

**SigningInfo** is a text file with the information on how to sign the Android App. Some of this information is sensitive, and anyone with access to this file could potentially maliciously edit the app, so be a bit careful with it.

## tour

You may or may not see a **tour** folder in the EICañoApp folder. The EICañoApp folder starts without this folder by default. The tour folder is a local copy of the Progressive Web App version of the app [that can be hosted on GitHub Pages](#). Any edits made to the tour folder will change the progressive web app. See a later section of the manual for information on [how to push these changes to GitHub Pages](#).

## UploadsAndStoreInfo

**UploadsAndStoreInfo** is a folder containing the information that's gone into the app store. This includes the app bundles to upload to the app store, as well as promotional screenshots of the app. If you make additional screenshots, it might be a good idea to store them here.

## Shell Scripts

To update the app, you will use a few shell scripts. A shell script is an executable program that enters commands into the command line for you.

To execute a shell script, navigate to the directory where the script is located, then type the following command<sup>4</sup> where SCRIPT\_NAME should be replaced by the name of the target script:

```
./SCRIPT_NAME
```

This will run every command within the shell script. You will normally see some output from the script on the command line. If an error occurs within a shell script, you can try entering each command by hand, one at a time.

Some shell scripts are only meant to be run once. To enforce this, the scripts remove their own status as an executable file. This essentially makes the shell scripts into text files. If you for some reason need to restore the executable nature of these scripts, enter the following command where SCRIPT\_NAME should be replaced by the name of the target scripts:

```
chmod 0700 SCRIPT_NAME
```

Most of the scripts are in the scripts folder. Be careful only to execute scripts when instructed by this manual, and make sure to execute the correct script.

## Updating the App

### Changing the Version Code

Part of releasing an app to any app store is updating the version number. The version number system our app currently uses is  $10000 * \text{MAJOR\_RELEASE} + 100 * \text{MINOR\_RELEASE} + \text{PATCH}$ . The line of code to edit uses the following form:

```
version = "MAJOR_RELEASE.MINOR_RELEASE.PATCH"
```

It would likely be best to modify the Minor Release number with annual updates.

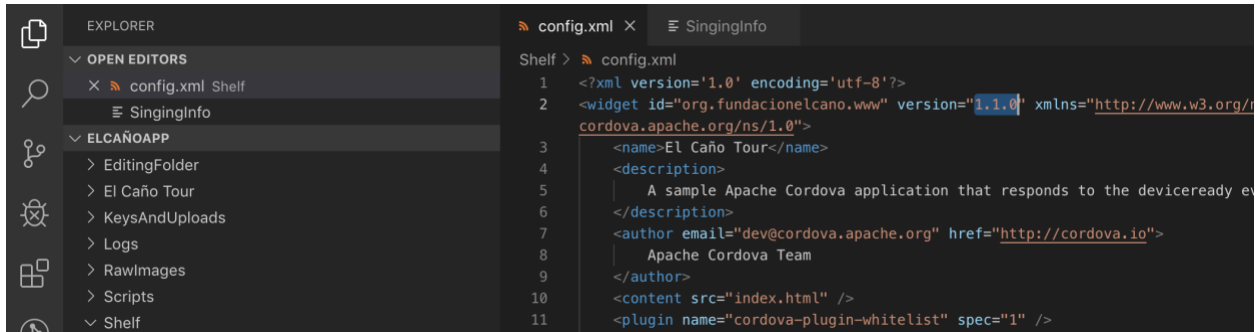
To update the version, you will have to use a text editor to modify the configure.xml file prior to building the app. If you have already built the app, you will need to build it again after versioning it. Open [Microsoft Visual Studio Code](#) (or any text editor), and open the config.xml file (located in the Shelf folder of the app, or at the absolute file path `~/ElCañoApp/Shelf/config.xml`)

The line you need to edit is shown in the image below.

<sup>4</sup> The command to execute a shell script is really any file path (absolute or relative) to get to the shell script other than the unaccompanied name of the shell script. The dot represents the current directory.

<sup>5</sup> The Chmod command can be confusing in a base-10 number system. The 7 in binary is 111, where the first digit represents read permissions, the second represents write permissions, and the third represents execute permissions. Each base-10 digit of the chmod command has a different meaning: the first is a system thing, the second is admin privileges, the third is privileges for a different user, and the fourth is privileges for anyone else. Use `man chmod` for more.





```

1 <?xml version='1.0' encoding='utf-8'?>
2 <widget id="org.fundacionelcano.www" version="1.1.0" xmlns="http://www.w3.org/1
  cordova.apache.org/ns/1.0">
3   <name>El Caño Tour</name>
4   <description>
5     A sample Apache Cordova application that responds to the deviceready ev
6   </description>
7   <author email="dev@cordova.apache.org" href="http://cordova.io">
8     Apache Cordova Team
9   </author>
10  <content src="index.html" />
11  <plugin name="cordova-plugin-whitelist" spec="1" />

```

Increment the middle number. Each number can be up to two digits. For more information on versioning, see [Cordova's documentation](#).

## Updating the Native Code

Before you update the app, you should probably make sure that the contents of the EditingFolder make a fully-functioning website. If you have not modified the contents of this folder, they will work just fine.

To update the Android and Apple source code of the app, execute the update shell script with the following command from the ElCañoApp directory:

```
./Scripts/updateScript
```

This will destroy the previous version of the app that was located in the Cordova project (the El Caño Tour folder), and it will replace it with new versions for both Apple and Android. These new versions of the app will only be different from the old versions if Cordova has actually updated their Android and Apple shell writers. To see if Cordova has updated, check [their blog](#).

If you see the following error for the ios build, it should be harmless:

```
error: Signing for "El Caño Tour" requires a development team. Select a
development team in the project editor. (in target 'El Caño Tour')
```

```
** ARCHIVE FAILED **
```

## Publishing the Android Version

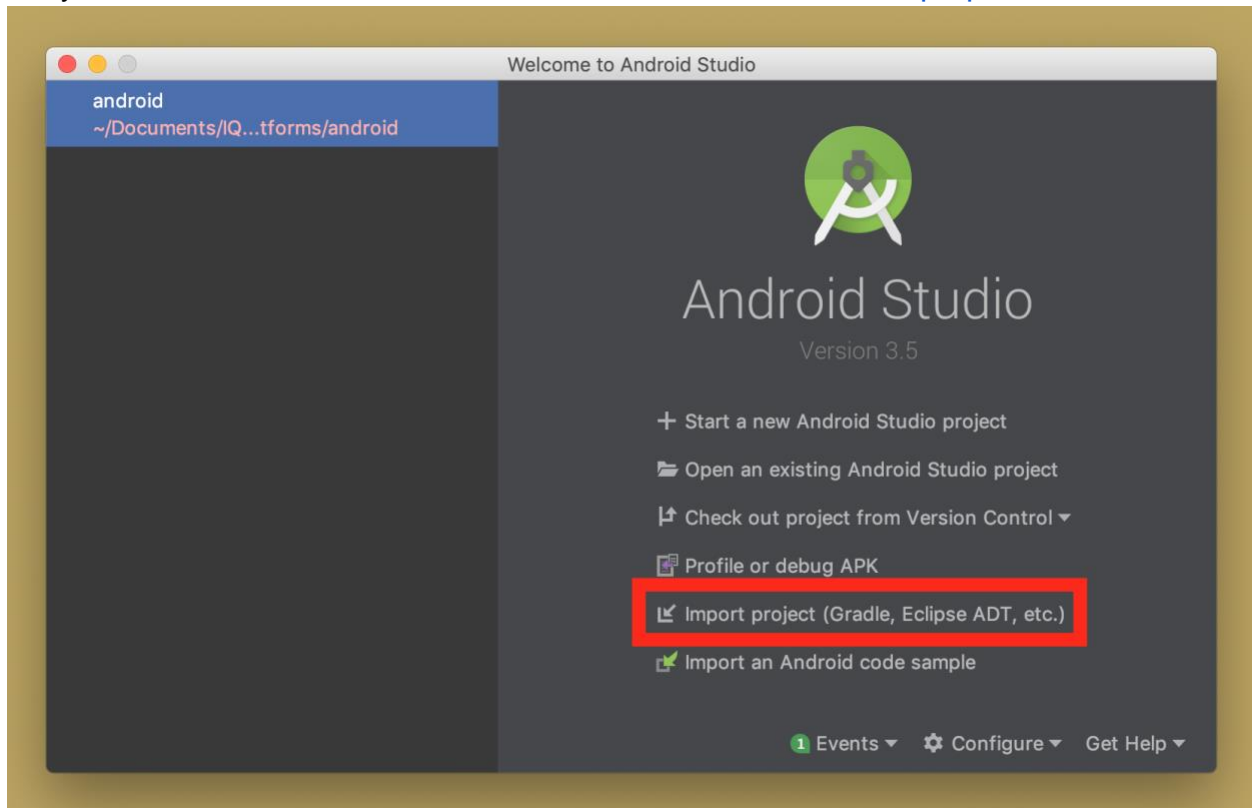
To publish the Android version of the app, you first need to have a built version of the Android app. To build the app, run the [update script](#) as described in the previous section.

Once you have built the Android app, the first step to publishing is to open Android Studio and use the menus to build a signed app bundle. A signed app bundle is a file that you upload to the Android App Store.



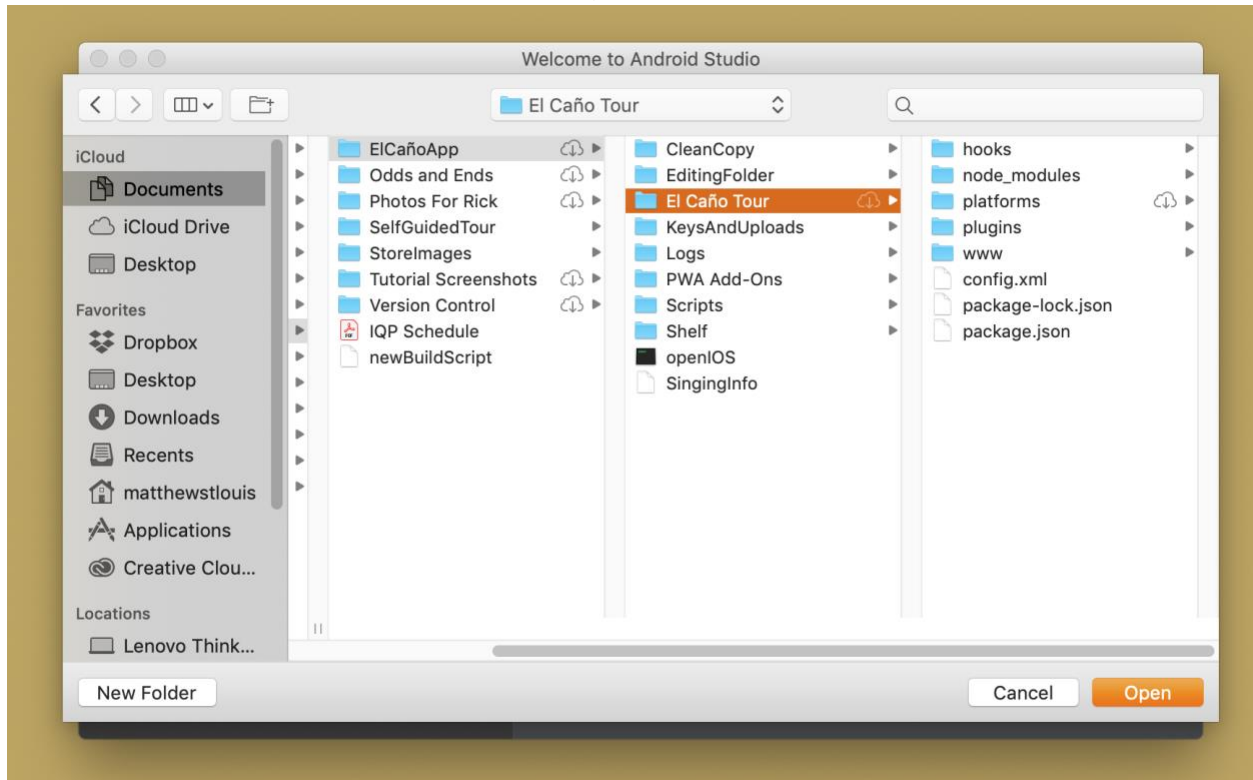
## Opening Android Studio

First, double-click the Android Studio icon to launch Android Studio. If you have not yet downloaded Android Studio, see the section on [software preparation](#).

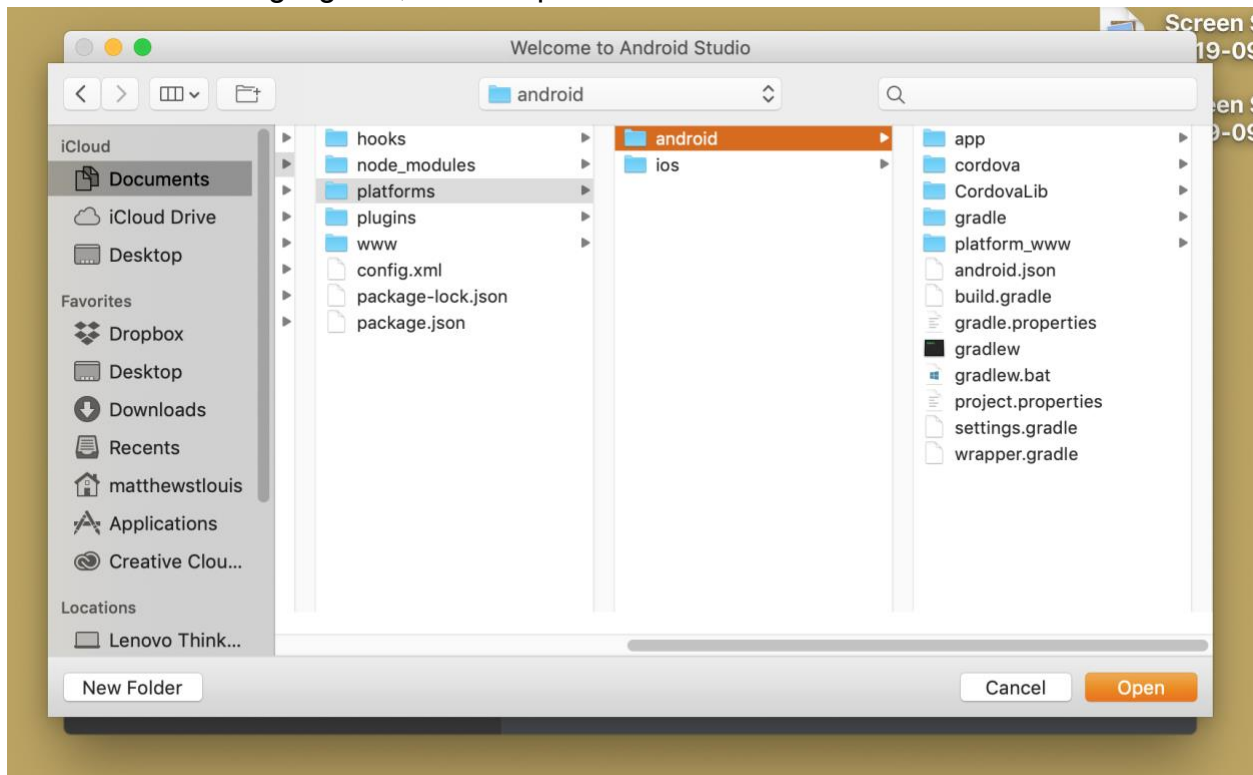


From this menu, select “Import Project (Gradle, Eclipse ADT, etc.)”. This should open your file system. From this, navigate to the EICañoApp folder as demonstrated in the following image.

From the El Caño App folder, navigate to the platforms folder.



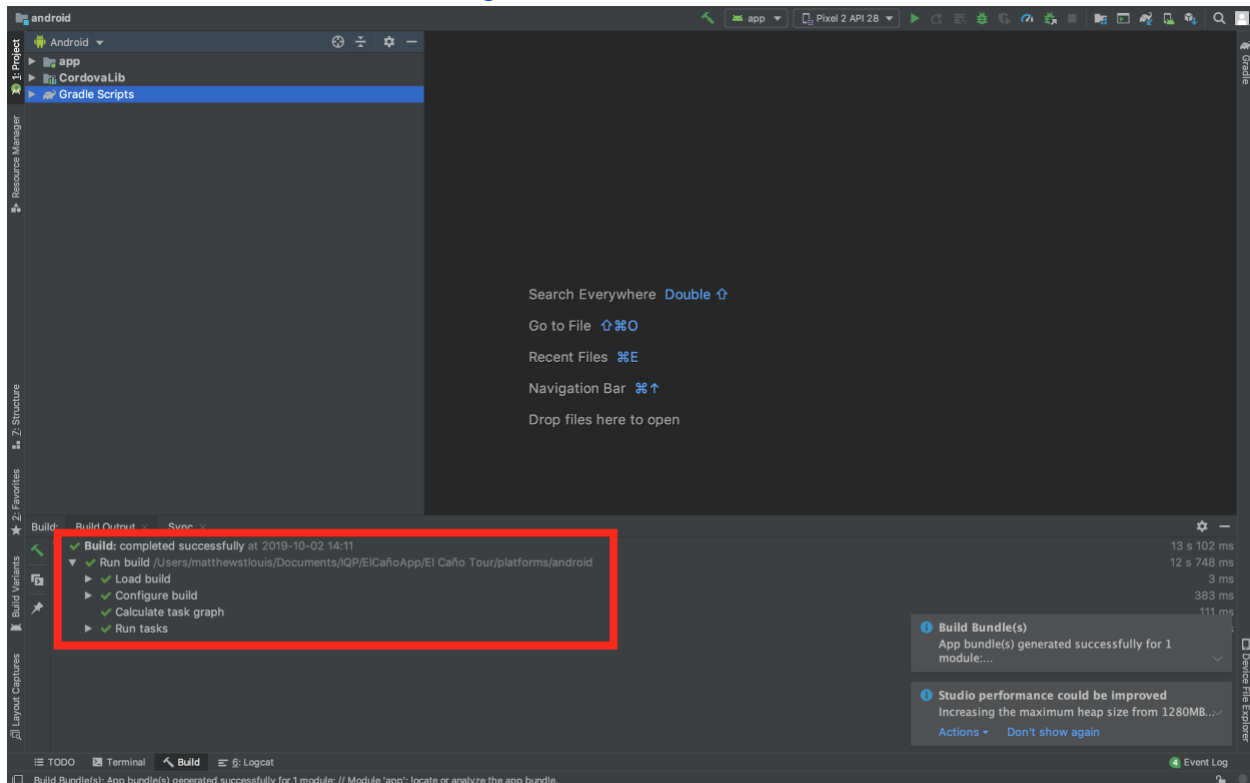
Within the platforms folder, navigate to the android folder. If you do not see an android folder, the build script failed. Try [troubleshooting the shell script](#). When the android folder is highlighted, select “open”



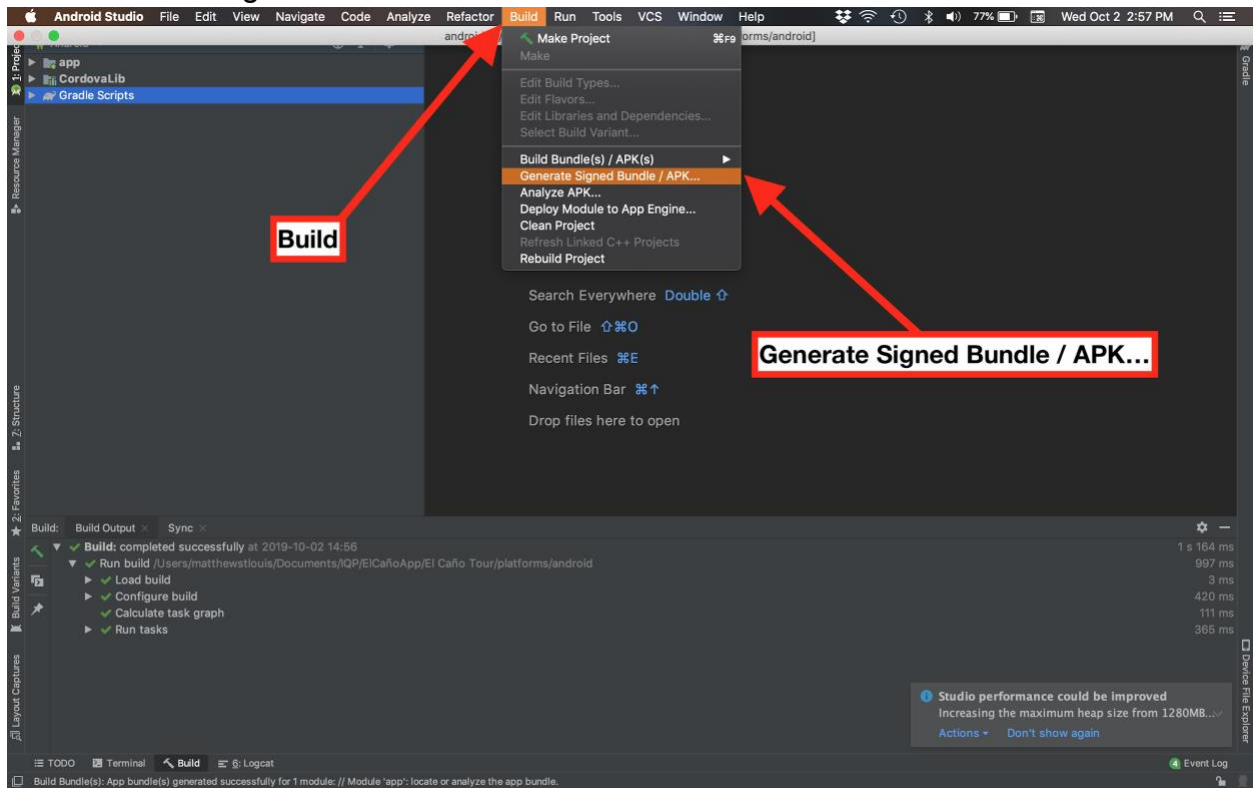
This should launch Android studio. If you see a screen asking you to install Software Development Kits (SDK's), you may not have downloaded the proper SDK's to update the app. You will need to refer to [an earlier section of the manual](#) to do this before you can proceed.

## Generating a Signed App Bundle

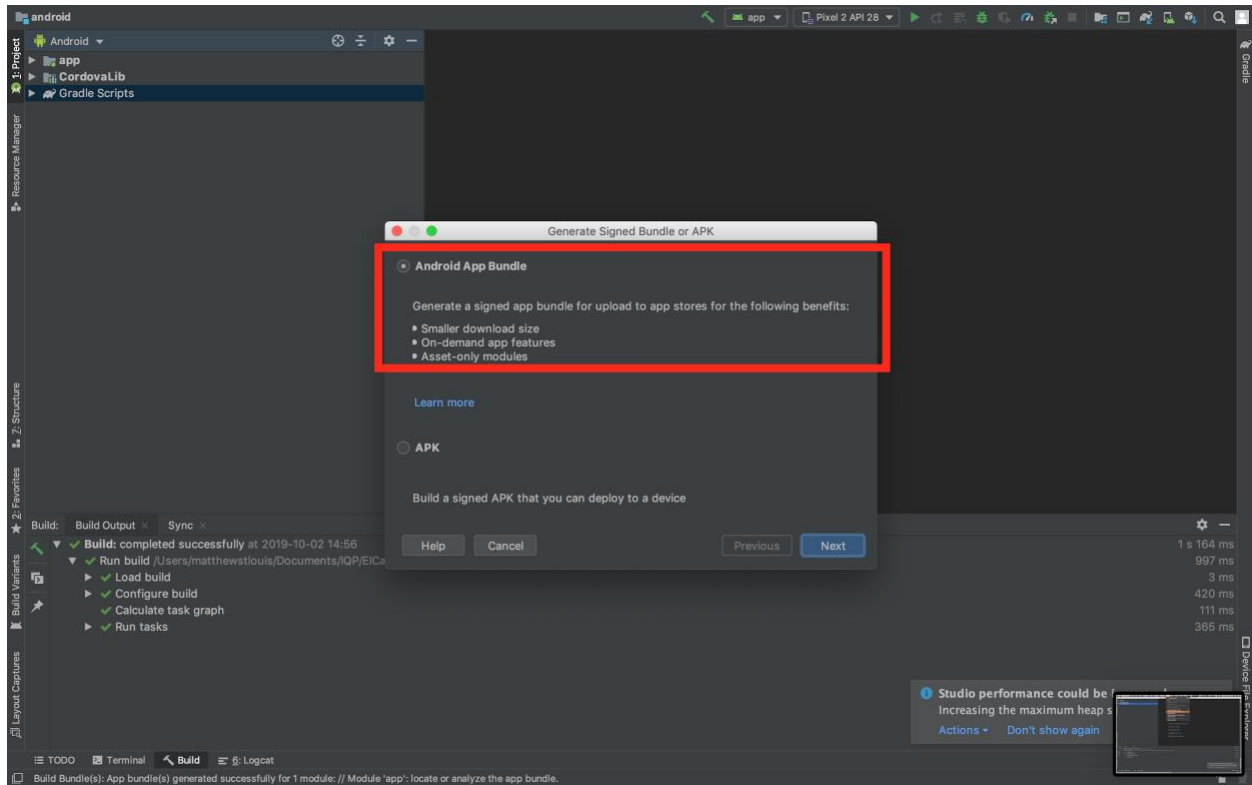
Once you have the app open in Android Studio, you should see the following screen. Before you can build a signed app bundle, the project needs to build properly. If the project builds properly, you should see a screen resembling the one below where there are green checkmarks in the lower left corner. If there are errors instead of checkmarks, try clicking on the dropdowns by the errors for more information as described in [Build Troubleshooting](#).



If the project has built properly, you should be able to make the app bundle. From the menu at the top of your screen, you can select build. A menu will open, from which you should select “Signed Bundle/SDK” as shown below.

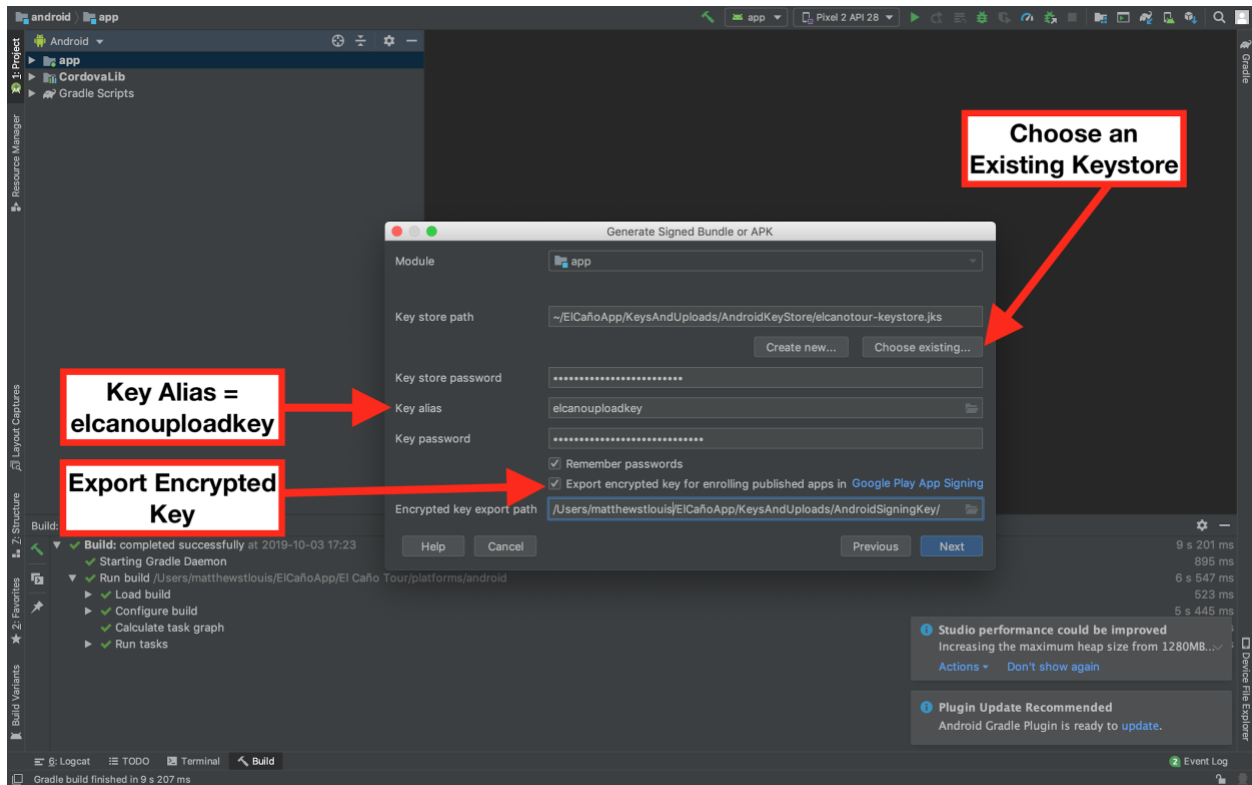


This should bring up a dialogue box that will ask you if you want to build an Android App Bundle or an APK. We want an Android App Bundle.



Once you have done that, you should see a screen asking for key and keystore information. This page has a lot of fields to fill. Please take care to fill them accurately.

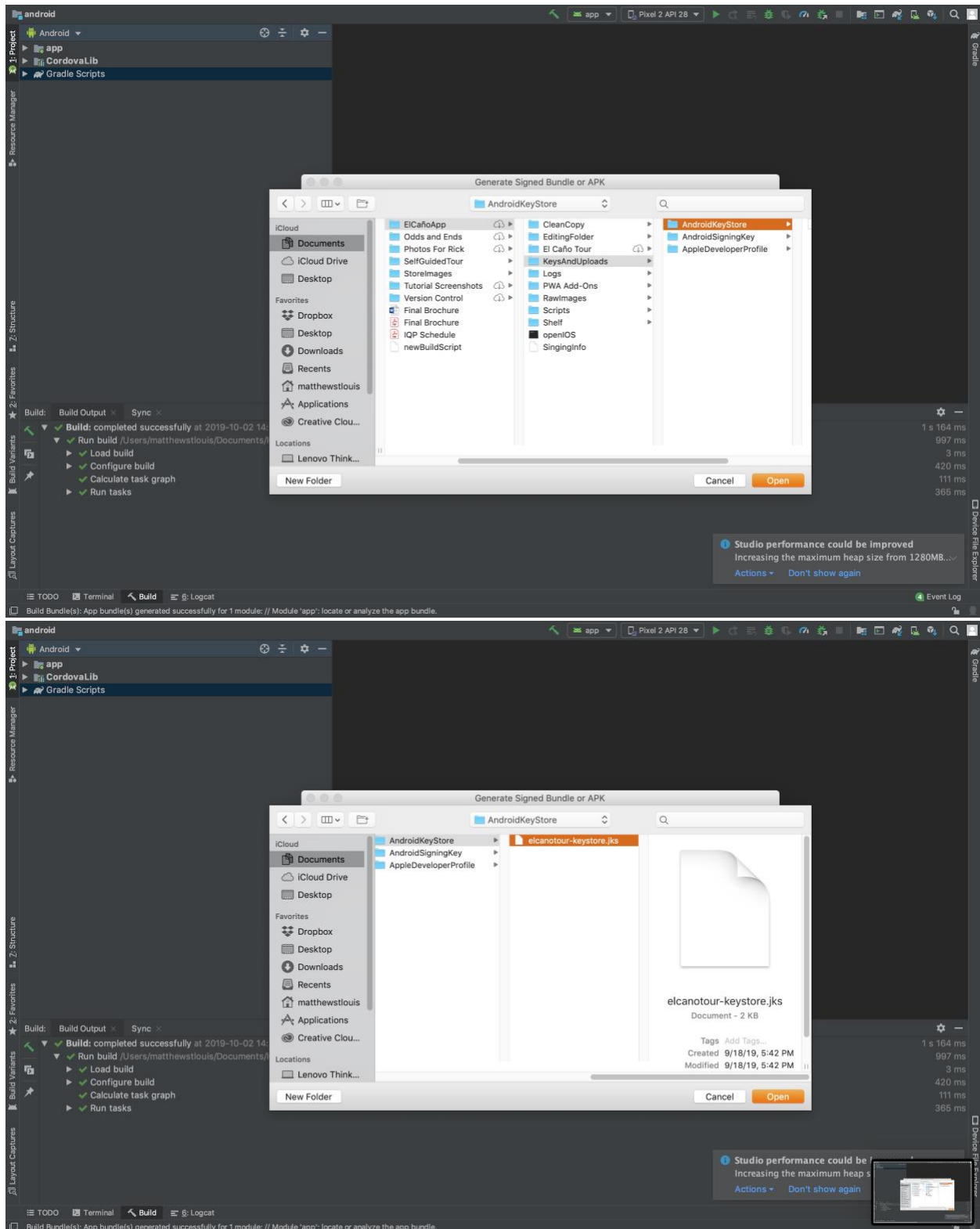
This information has been centralized in the text file “SigningInfo” in the ElCañoApp folder.



The Module will be “app” by default. Leave that alone.

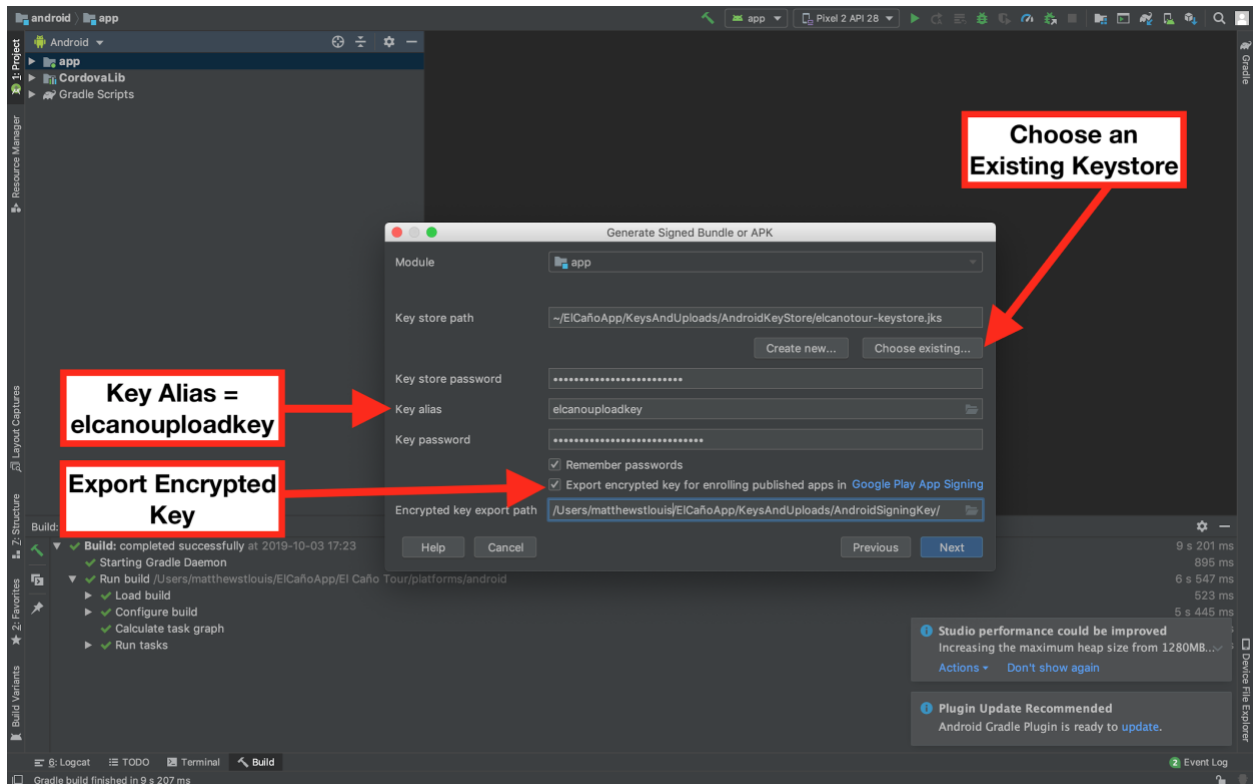
For the key store path, you can either type a path manually, or you can press the folder button to the right to select a path from your file system. The latter method is generally less error-prone if you can find the right file. The correct file path should be  
~/ElCañoApp/KeysAndUploads/AndroidKeyStore/elcanotour-keystore.jks

This file path is shown in the following images.



The next field is the Key Store password. This may be on your computer's keychain. If not, ask someone from the foundation for it. (Image reprinted for reference)





The fourth field is the Key Alias. The key alias is elcanouploadkey.

The fifth field is a Key password (distinct from the keystore password). This may also be on your computer's keychain. If not, ask someone from the foundation for it.

Below these fields are two checkboxes: one to remember your password and another to export the encrypted key. I would recommend you check the first, but you need to check the second. I'm not sure that the upload will work at all without exporting the key.

Checking the second box will bring up a new field for the Encrypted Key Export Path. Either type the file path manually or press the folder icon to the right to select the folder from your file system.

The correct folder name will have the following form where USER\_NAME is replaced by your username on the computer (in the foundation's case, FEC):

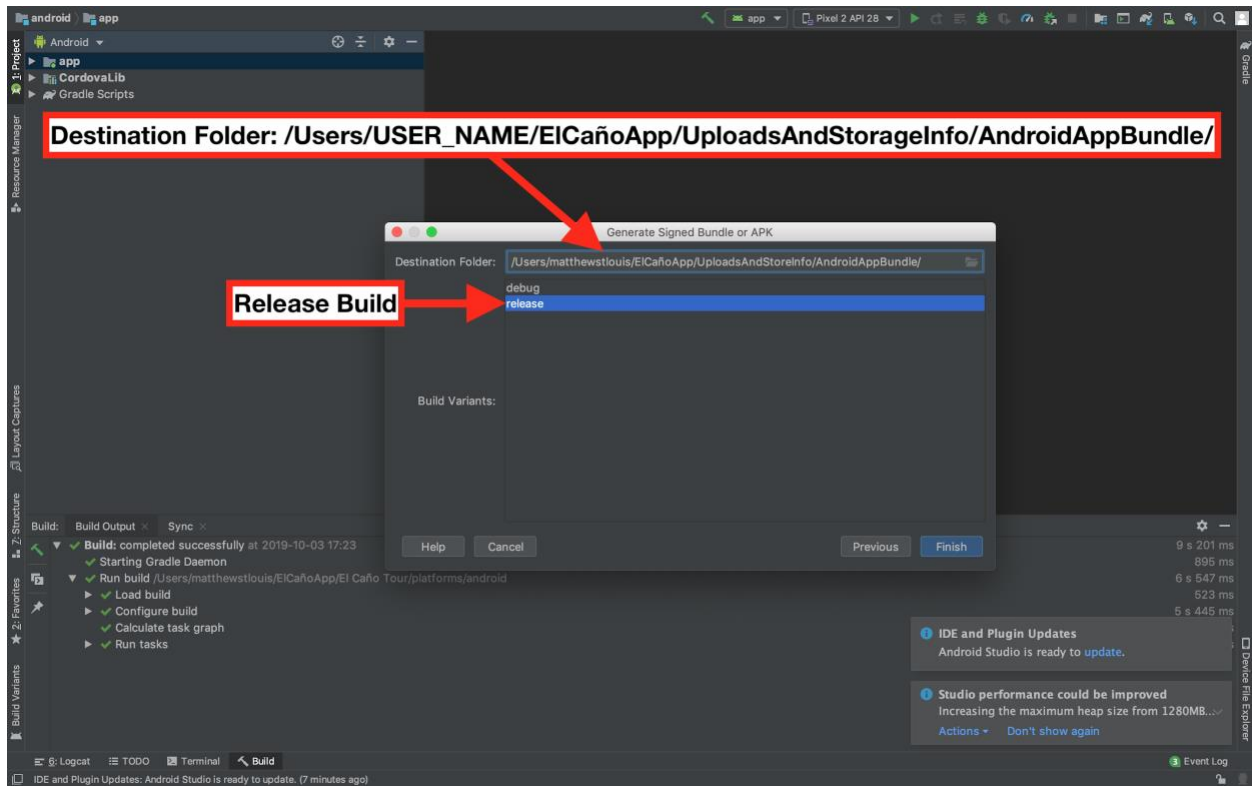
`/Users/USER_NAME/EICañoApp/KeysAndUploads/AndroidSigningKey/`

Once you have entered all of that information and double-checked it, hit "next."

The next screen will let you select between a debug build and release build. Since we intend to put this version on the app store, it should be a release build.

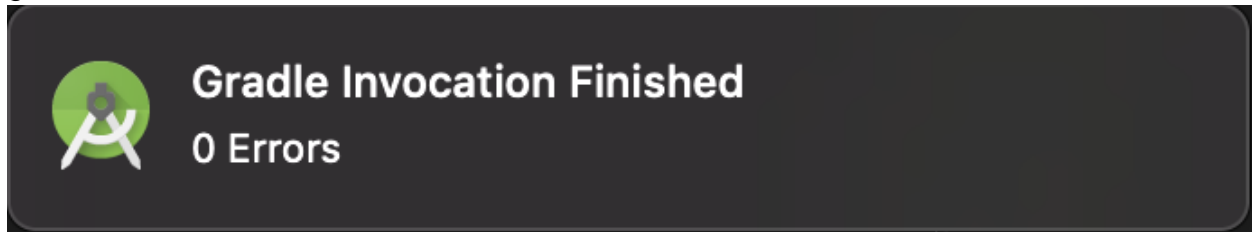
At the top, there is a field for a file path for the project build. Please put the bundle into the following folder where USER\_NAME should be replaced by your username on the computer (in the foundation’s case, FEC):

/Users/USER\_NAME/EICañoApp/UploadsAndStorageInfo/AndroidAppBundle/



When this is done, press “Next”

If you have entered all of the information right, you should see a few spinning wheels over the green checkmarks in the lower left hand corner of the screen. You may get a notification like the one shown below:



To see whether or not the build succeeded, you can check the folder where the folder was placed: `~/ElCañoApp/UploadsAndStorageInfo/AndroidAppBundle/`

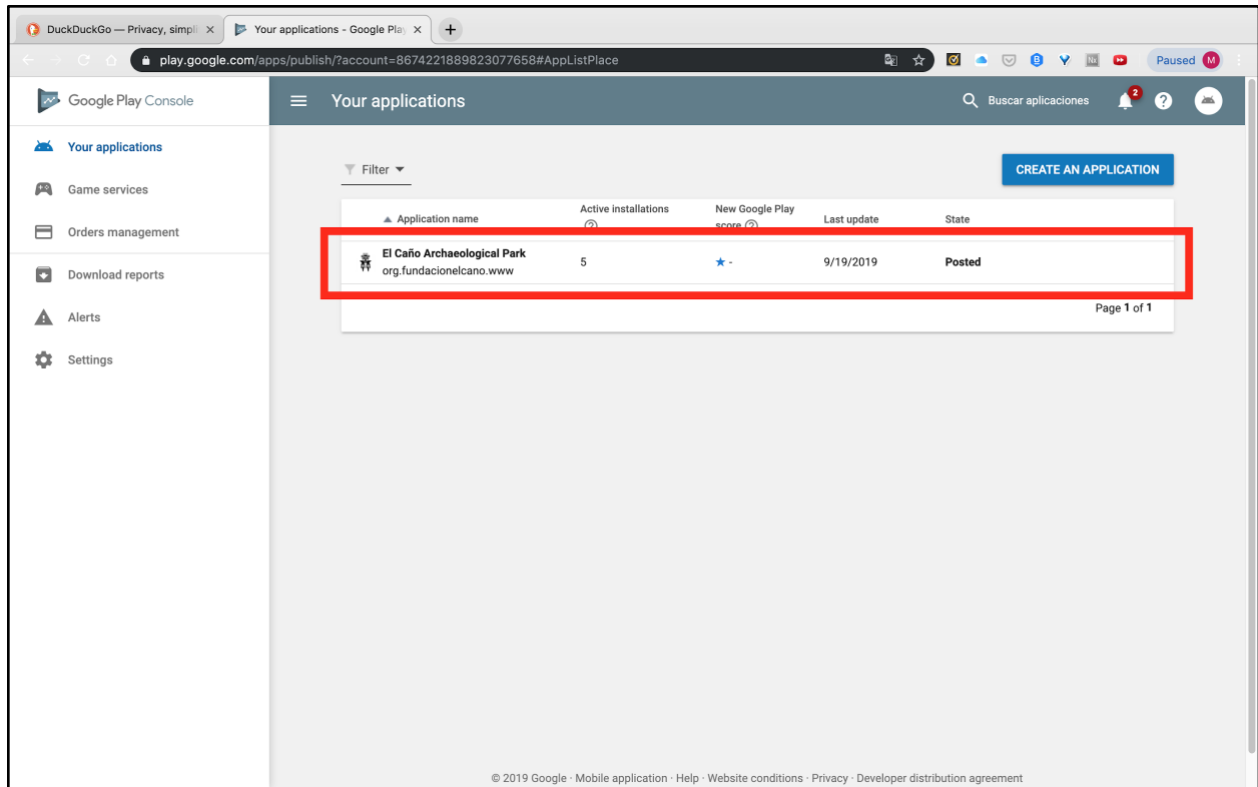
If this folder is empty, the build failed silently. Unfortunately, Android Studio does not let you know that the build failed directly, nor does it let you know what failed. You may have to enter all of this information again.

If you do see a new app bundle (app.aab), you did it! That's the file we will upload to the Android App Store.

## Uploading to the Android App Store

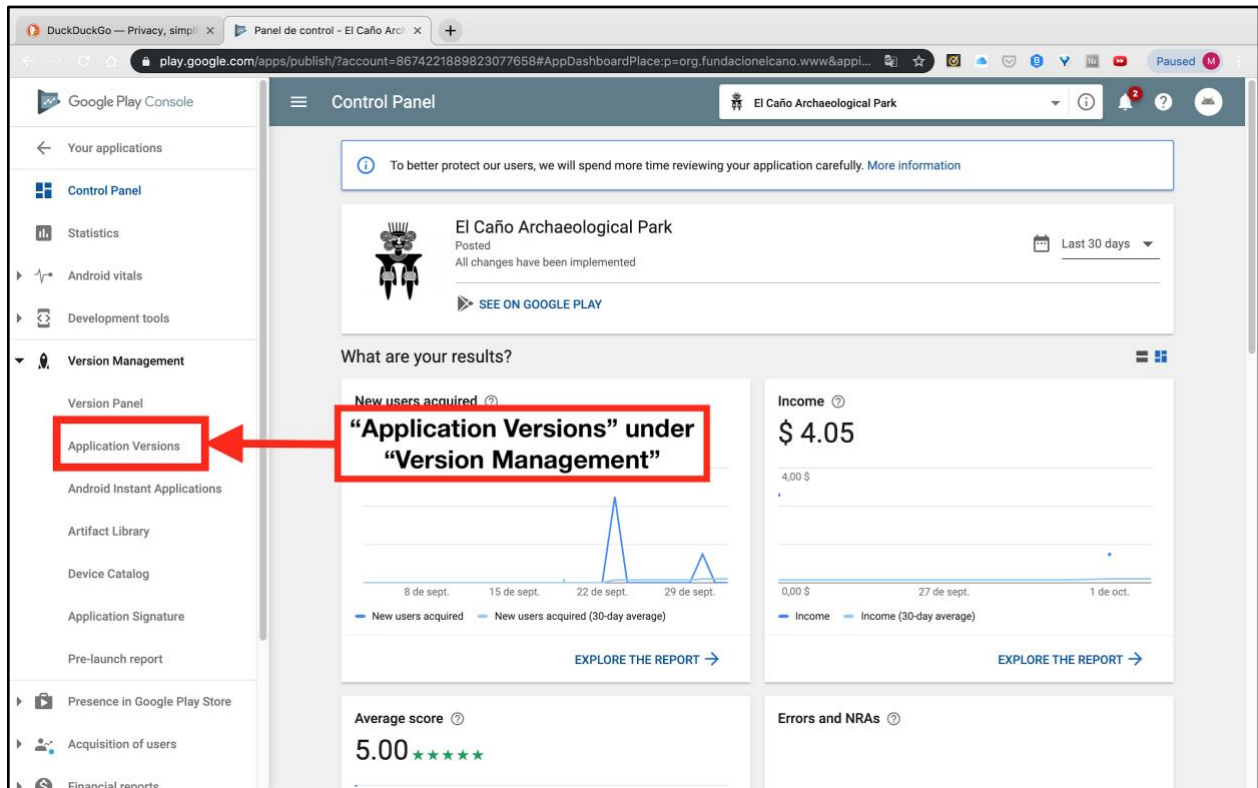
Now that you have the Android App Bundle ready, you can load it onto the Android App Store through the [Google Play Console](#). To use the Google Play Console, you will need to use the foundation's Google Account (fundacionelcano@gmail.com).

Once you log in, you should see the following screen.



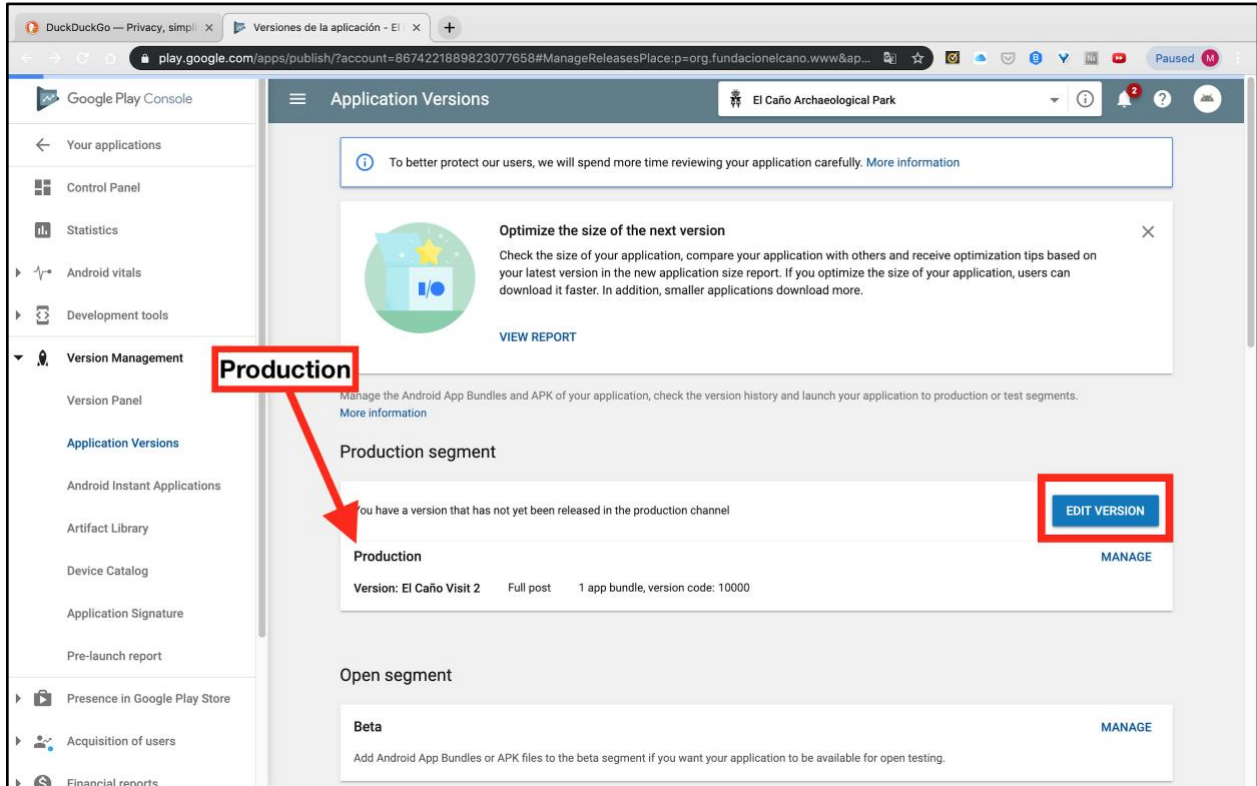
This is your home screen for the entire Google Play Console. You're free to explore it, but most of the features relevant to the app are only accessible if you click on the app in the red box as shown.

This screen is your “Control Panel.” It has a bunch of neat statistics about app purchases.

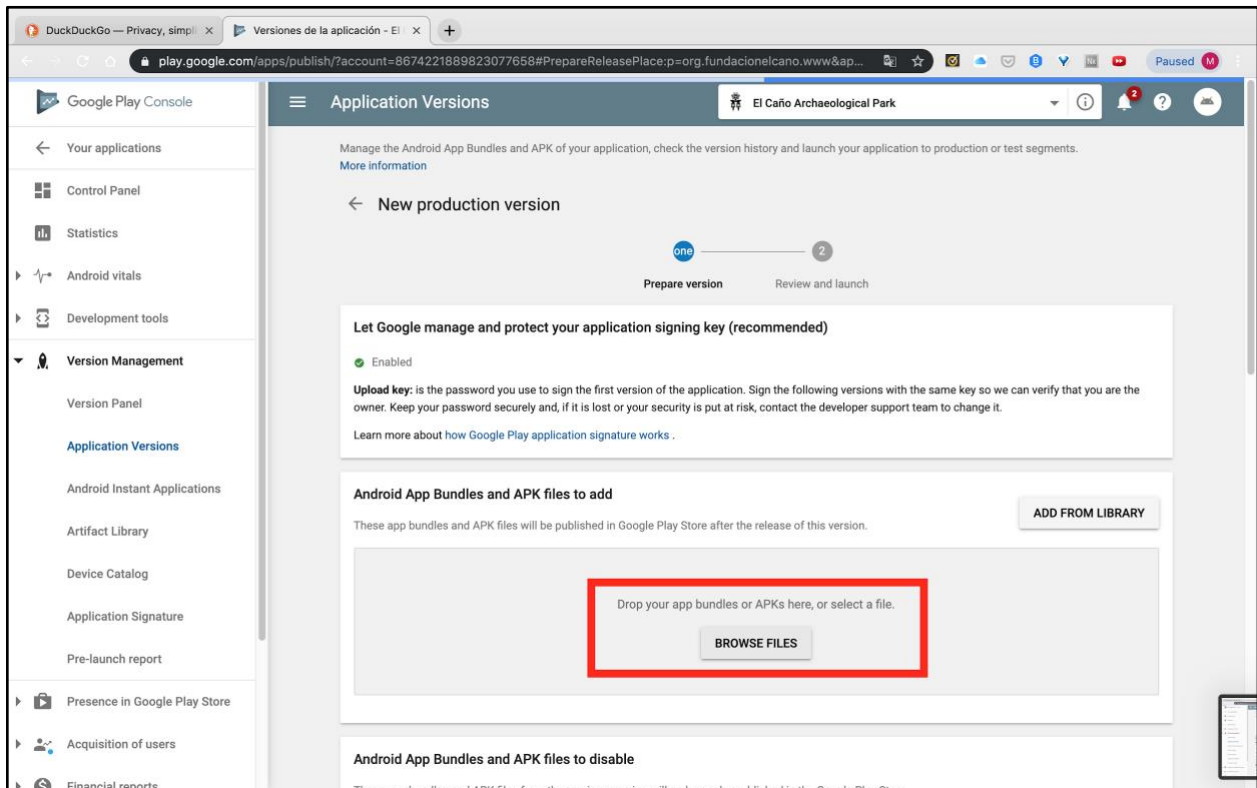


To upload a new version of the application, select “Application Versions” under the “Version Management” option from the menu on the left as shown above.

This will bring up a menu of the different versions of the app. As of the writing of this manual, there only exist production versions. Unless you have added a new experimental feature to the app or something, you will want to launch the version you made as a production version. To do this, press “Edit Version” on the panel for production versions.



You should now see a screen similar to the following. This is where you will upload the app bundle that you have generated. If you have not generated one, [see the section above](#).

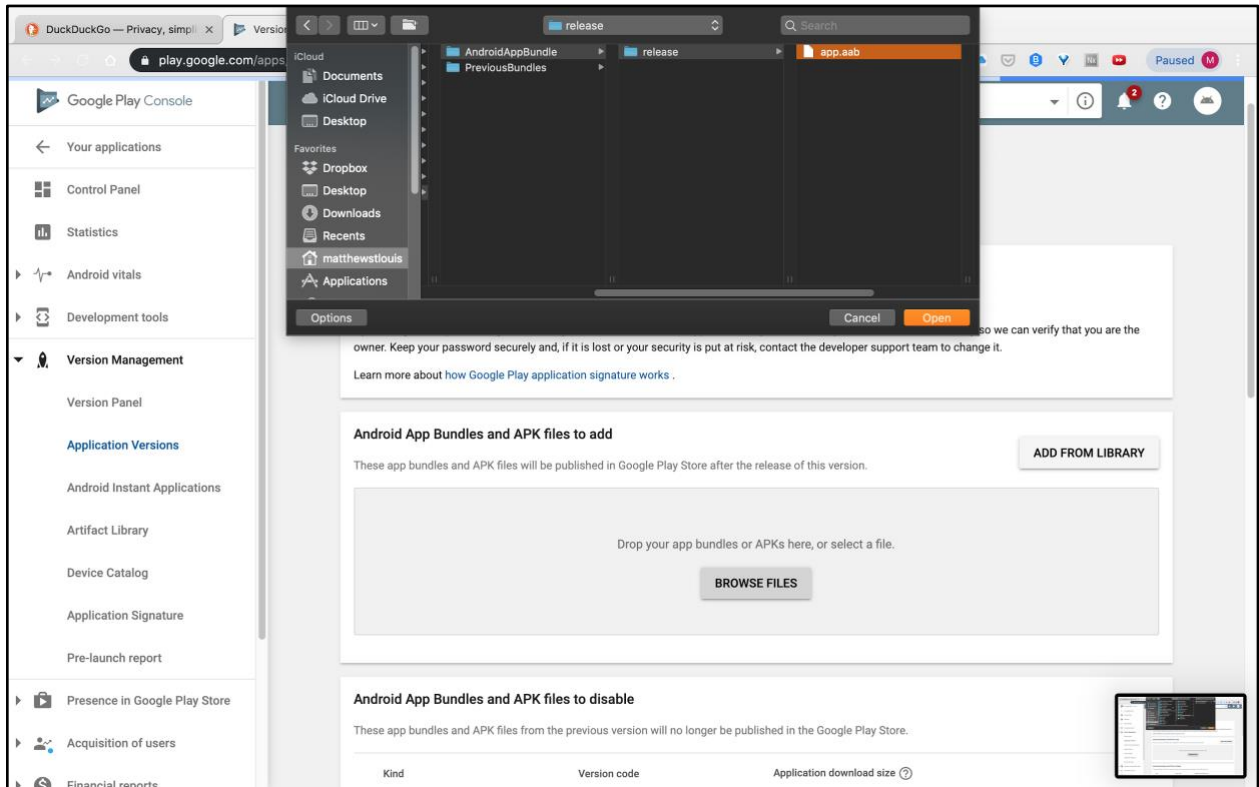
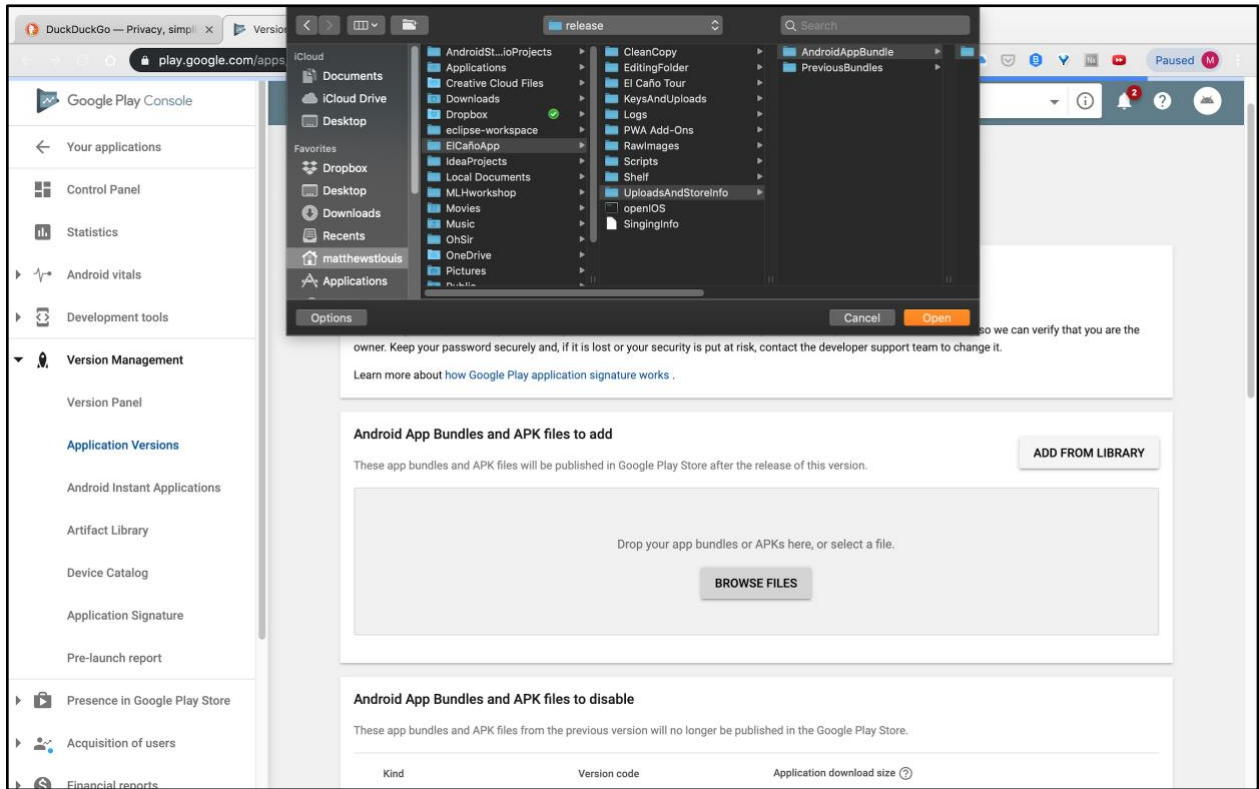


Some other information you may see is Google letting you know that they manage our signing key. This option lets Google manage our upload key, which is separate from the key used to sign the app to build a new version. This option makes it so that if we lose our signing key, we don't lose the ability to update the app altogether.

From this screen, choose the option to load an Android App Bundle from your files, as shown above.

The next two images show the correct folder to select. The absolute file path is as follows:

~/ElCañoApp/El\ Caño\ Tour/UploadsAndStoreInfo/AndroidAppBundle/release/

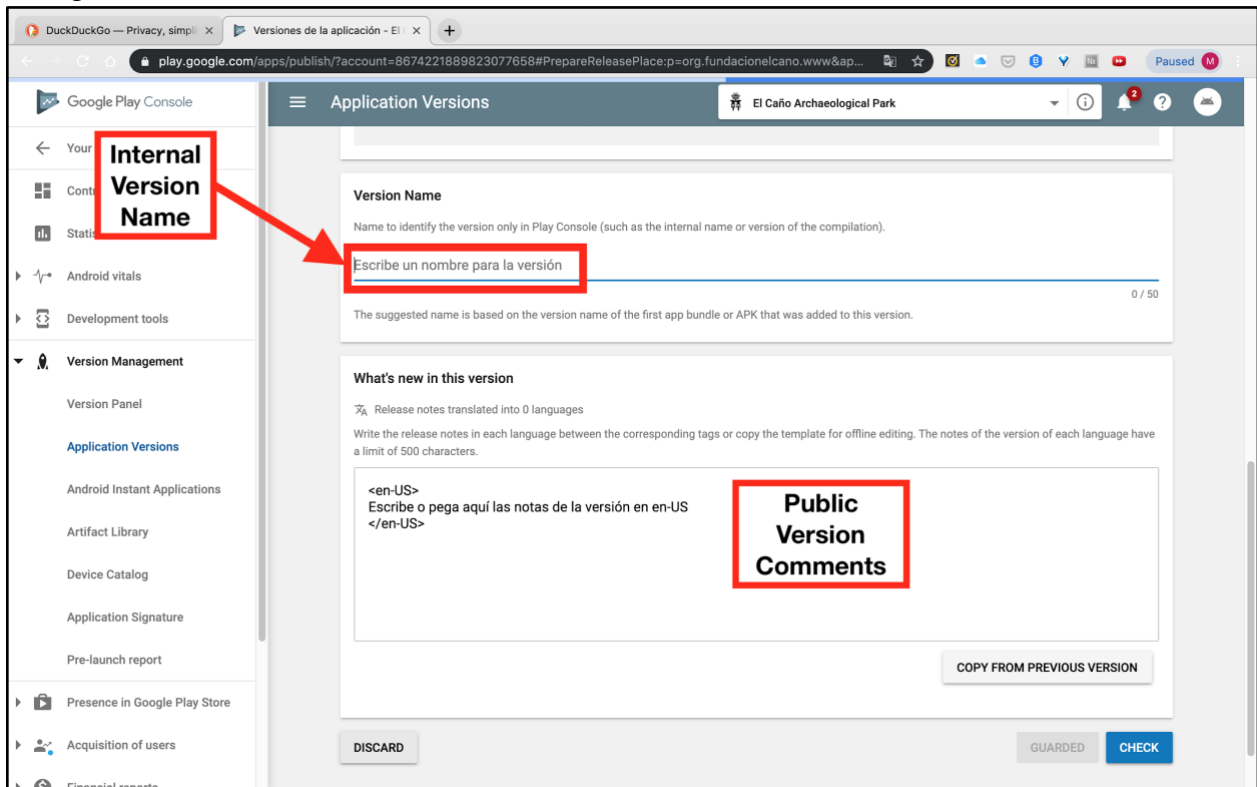




The upload may take a few minutes. If it is successful, you should see a version code for the app. If the webpage complains about a version code that is too low, you will need to [change the version code and remake the app](#).

When the upload is done (or while it's happening), you can scroll down to the part with the version name and the version comments. The version name is something internal for your reference. Pick something that will help you identify when it was released (e.g. the Month/Year, what you changed, the version number, some combination of these).

The second box is the version comments. This text will be displayed to let users know what changed since the last version. These version comments will be on the app store for the life of the app, so keep that in mind when writing them. Most apps use the version comments to thank their users for their support or to get excited about the app, though we don't need to.



When everything is ready, press the button in the lower right corner of the page to save your changes. Once the changes have been saved, press the button in the lower right to continue to the next screen.

Once you have loaded the app into the Android App Store, please rename the Android App Bundle and move it from the AndroidAppBundles folder to the PreviousBundles folder. This helps keep track of past versions of the app in case you need them. Google also keeps past versions in the Play Console, so you don't necessarily need to do this. If you decide not to keep past versions of the app, please

still delete the generated .aab bundle file so that you can tell when a new one is made for the next build, as Android Studio does not let you know when a build fails.

## Android Build Troubleshooting

### Modifying the App

You may want to update the app to change the content, to add support for another language, or to change the appearance. The first two require some programming in HTML, and the third will likely take a combination of HTML and CSS. HTML can be relatively easy to pick up, but be warned that CSS is notoriously difficult to work with.

Please do not rename any of the files unless you are prepared to change their name anywhere they appear (including in HTML files and in the [serviceworker](#)).

### Looking at the code

To edit the app, you will need to use [Microsoft Visual Studio Code](#) (or another text editor). To open a folder in VS Code, you will need to open VS Code, then press “Open” under the “File” menu. From here, select the folder you want.

Opening code in a text editor can be helpful for finding what’s going wrong with shell scripts.

### Editing the Donation Link

If the foundation ever switches donation services from GlobalGiving or changes to a different GlobalGiving campaign, someone will need to update the donation link within the app. The donation link is located in the index page of each language folder shown below. Open up these files in Microsoft Visual Studio Code (or any text editor) and replace the absolute link shown with the correct one. If the app is in the app store,

you will need to [update it](#).

```
PagesEN > <> index.html > <html > <body > <h1
1  <!DOCTYPE html>
2  <html lang = "en">
3    <head>
4      <title>
5        English Landing Page
6      </title>
7      <meta charset="UTF-8">
8      <meta name="viewport" content="width=device-width, initial-scale=1">
9      <link rel="manifest" href=" ../manifest.json">
10     <meta name="apple-mobile-web-app-capable" content="yes">
11     <meta name="apple-mobile-web-app-status-bar-style" content="black">
12     <meta name="apple-mobile-web-app-title" content="El Caño Tour">
13     <link rel="apple-touch-icon" href="https://mastlouis.github.io/SelfGuidedTour/Images/WebIcons/
HighRes.png">
14     <meta name="description" content="A self-guided tour of the El Caño Archaeological Park in Coclé,
Panamá.">
15     <meta name="theme-color" content="#ae954c"/>
16     <link href="https://fonts.googleapis.com/css?family=Open+Sans&display=swap" rel="stylesheet">
17     <link rel="stylesheet" href=" ../Styles/indexEN.css" type="text/css">
18   </head>
19   <body>
20     <script src = "https://mastlouis.github.io/SelfGuidedTour/serviceworker.js"></script>
21     <a href=" ../index.html"></a>
22     <h1>
23       Welcome to El&nbsp;Caño Archaeological Park
24     </h1>
25     <div>  </div>
26     <div class="Forcentering"><p class="parkmap"><a href=" ../PagesEN/parkmap.html"
style="text-decoration:none">Park Map</a></p></div>
27     <div><p class="sponsors"><a href="sponsor.html" style="text-decoration:none">Sponsors</a></
p></div>
28     <div><p class="donations"><a href="https://www.globalgiving.org/projects/
rescue-the-el-cano-archaeological-park-museum/" style="text-decoration:none">Donations</a></
p></div>
29   </body>
30 </html>
```

## Changing out Images

If there are significant changes to the areas in the park, you may want to update some of the images within the app. The easiest way to do this is to put an image into the correct image folder (either park, museum, or top level) and to give it the exact name of the image you want to replace. The exact name means that the new image will need to be in the same file format as the old image (e.g. jpeg, jpg, png, gif). You may need to rename or remove the old image to do this.

If you insert an image this way, try to give it a similar aspect ratio (width-to-height ratio) as the old image so it can cleanly replace the old one. To save space and to make the app smaller, we scaled down our images to 720 pixels in width while preserving the aspect ratio. This brought the app from about 26MB to about 11MB, so we would recommend scaling any new images you add. We chose 720px because it's twice the pixel density of the smallest common smartphone viewport.

## Editing the Content (HTML)

In the event that the foundation needs to change the content within the application, this can be done in [Visual Studio Code](#) (or another text editor). The HTML files hold all the text and pictures within the application. Most HTML is written between a pair of tags: an open tag and a close tag. The close tag usually contains a slash. You can specify some attributes within a tag using an equals sign (=). The most basic HTML functions for text are listed below:

1. `<html> </html>`

This tag defines the start and end of the whole document.

2. `<body> </body>`

This tag defines the start and end of the body of the document.

3. `<p> </p>`

This tag defines a paragraph of text. These tags go around any amount of text that is to be shown in the application.

4. `<h1> </h1>`

This tag defines a header. There are headers 1 through 6. These can be used to style certain phrases of text including titles.

5. `<img src="">`

This tag defines an image. The relative file path should be inserted in the quotation marks.

6. `<a href=""> </a>`

The anchor tag defines a link. The relative file path<sup>6</sup> of the page that you would like to link to should be inserted in the quotation marks.

To learn more, you can visit the HTML homepage of [W3Schools](#). This website has an immense amount of information on other HTML functions. Some other notable HTML features we used in the app are [tables](#) (for the list of pages) and [image maps](#) (to make the park map and museum map numbers clickable).

## Editing the Style (CSS)

The content contained in the HTML files is styled using CSS files. In the event the foundation would like to change an aesthetic feature of the application, this can be done within the CSS files in [Visual Studio Code](#) (or another text editor). The CSS files are a series of selectors and properties. A selector selects one or more tags within the HTML files, and the properties style the selected tags.

<sup>6</sup> You can use either a relative file path or an absolute file path for any of these. A relative file path starts in the directory of the file from which the file path is written. You can go out one directory in a file path with two dots (`..`). You can go out multiple directories with consecutive double dots (`../..`). An absolute file path for the progressive web app will start with <https://fundacionelcano.github.io/tour/>. To link to an external site (like we did for the donation link), you will need to use an absolute file path.

For example, to style the text paragraphs, headers, or images you would identify the tags and style accordingly. The image below shows a sample of CSS code in which the body, h1, h2, p and image tags are styled.

```
9 }
10 body{
11     background-color: #white;
12 }
13 h1{
14     font-family: 'Brandon Grotisque', 'Klill Light', 'Open Sans', sans-serif;
15     background-color: #white;
16     color: #black;
17     font-size: 2.75em;
18     text-decoration: none;
19     text-align: center;
20     margin-right: auto;
21     margin-left: auto;
22     width: 100%;
23     margin-top: 0px;
24 }
25 h2{
26     font-family: 'Brandon Grotisque', 'Klill Light', 'Open Sans', sans-serif;
27     background-color: #white;
28     color: #AE954C;
29     text-align: center;
30     width: 40%;
31     margin-bottom: 50px;
32 }
33 p{
34     font-family: 'Brandon Grotisque', 'Klill Light', 'Open Sans', sans-serif;
35     color: #black;
36     text-align: justify;
37     font-size: 1.2em;
38     margin: 5%;
39 }
40 img{
41     height: 90%;
42     width: 90%;
43     display: block;
44     margin-left: auto;
45     margin-right: auto;
46     margin: 5%;
47     box-shadow: 10px 10px 5px #grey;
```

In order to specify a specific image or group of text you can use classes and id's. The class is identified within the HTML tag and can then be styled separately from similar tags. The classes are labeled with a period (.) and the id's are labels with a pound sign (#). The HTML file containing the class "backarrow" is shown below and the CSS files that styles it is shown underneath.

```
<script src = "https://maatlouis.github.io/SelfGuidedTour/serviceworker.js"></script>
<a href=" ../index.html" </a>
<h1>
  Welcome to El Caño Archaeological Park
</h1>
```

```
135 .backarrow{
136     height: 40px;
137     width: 40px;
138     background-color: white;
139     border: none;
140     margin-top: 0;
141     margin-left: 0;
142     margin-bottom: 0;
143     box-shadow:none;
144 }
145 #logos{
146     width:40%;
147     margin: auto;
148     box-shadow:none;
149     padding-bottom: 5px;
150 }
```

There are many properties you can use within CSS to style a page. These include font, text size, color, background color, location within the page, margins, and other many more. You can learn more by going to the CSS page of [W3Schools](https://www.w3schools.com/css/).

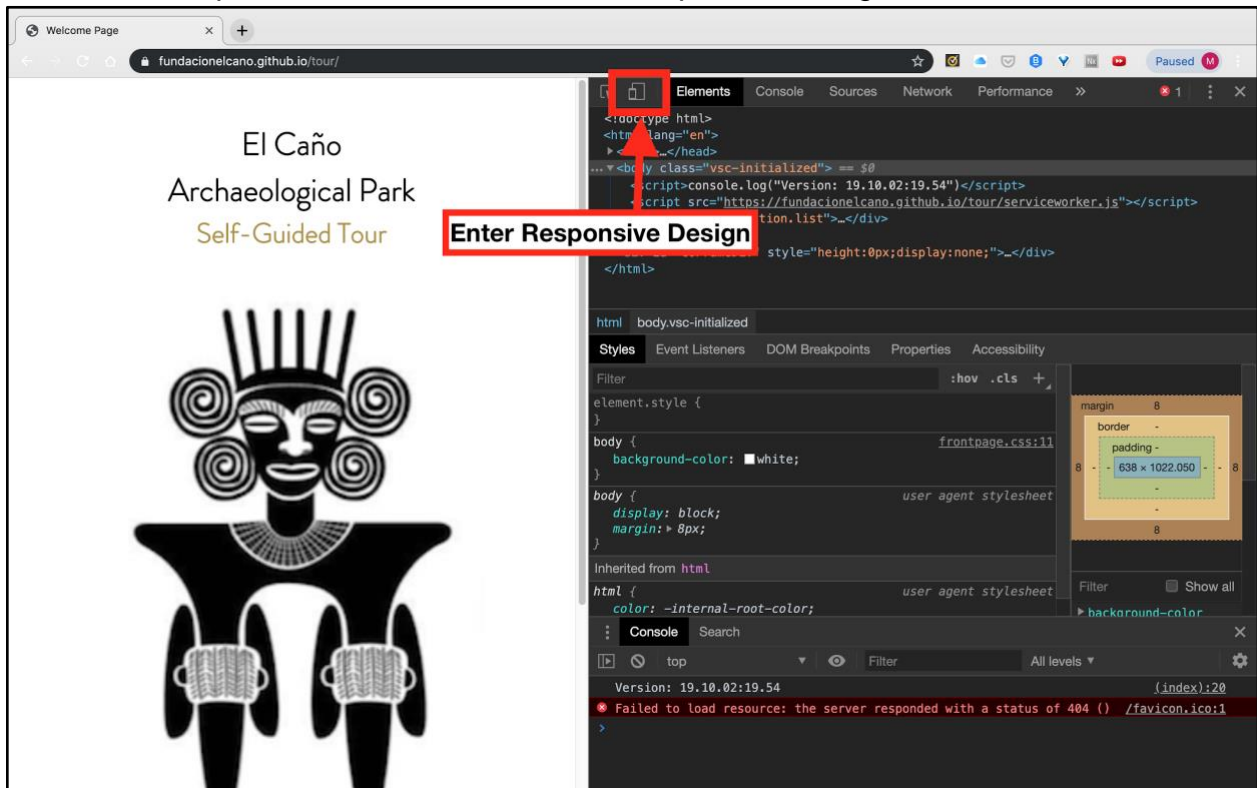
## Looking at the App

While editing or modifying the application, it is helpful to be able to view the application the Chrome Dev Tools. To do this, right click on the HTML file and scroll down to "reveal in finder." We recommend using the outermost index.html file for this. In the finder view, right click again and scroll down to "open with" and choose Google Chrome. You should be looking at the application on Google Chrome as though it were being served from the internet.

The proportions and dimensions of the app will probably look weird to you. This is because the app was only designed to be viewed on a phone screen. To preview what the app would look like on a phone screen, open the Chrome Developer Tools. One way to do this is by pressing the "option", "command" and "i" keys together. Another way is by pressing F12 (or fn and F12 together, for most users).



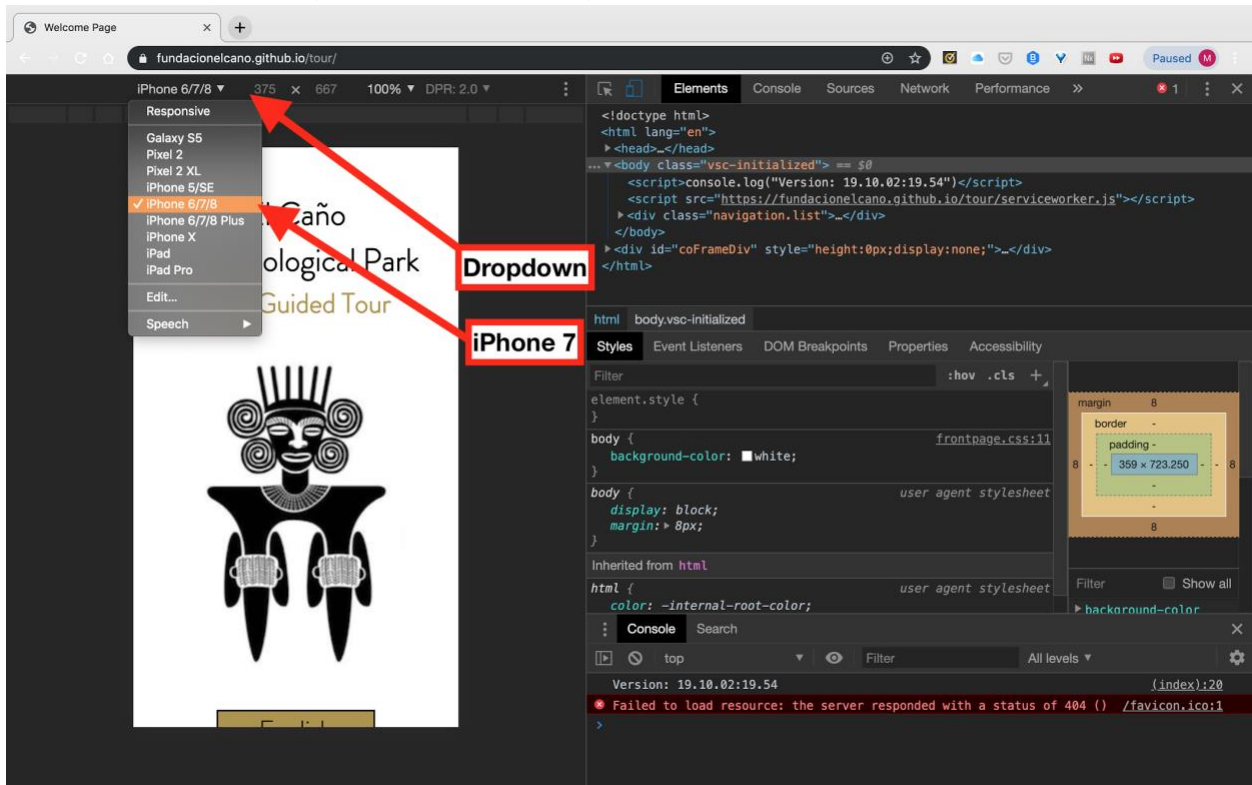
Once the Chrome developer tools are open, you should see a screen like the one below. As far as browser developer tools go, Google Chrome has by far the best. Press the icon of the phone and the tablet to enter responsive design mode.



Once you have done that, you will have more control over the dimensions of the window you are working with.



There should be a dropdown as shown below that will let you change the dimensions of the web page window to match those of popular phones. For developing the app originally, we used the iPhone 7 as a guideline. It's generally a good idea to design for the smallest available device (which would be the iPhone 5 in this case) and to check that the design still works on larger devices.



Previewing the app in Google Chrome like this is a good way to take [screenshots for the app store](#) if you change the layout of the app.

## Directing People to the App

Most users who download the app will likely already be planning to visit El Caño Archaeological Park. The main place they will likely look when planning their visit is the website ([www.fundacionelcano.org](http://www.fundacionelcano.org)). This is where we should advertise the app with a link, and potentially with a QR code (you can use whatever you find best). The links and codes to download the various versions of the app are in the sections below. These links should theoretically all be stable through version changes and updates.

If any of the links change, you can generate a QR code with just about any QR generator online. A QR code is just a 2D format of encoding plain text. Here is one example of a [QR generator](#).

## Android App Store

<https://play.google.com/store/apps/details?id=org.fundacionelcano.www>



## Progressive Web App

See [this section](#) on how to activate the Progressive Web App  
<https://fundacionelcano.github.io/tour/>



## Managing App Store Listing

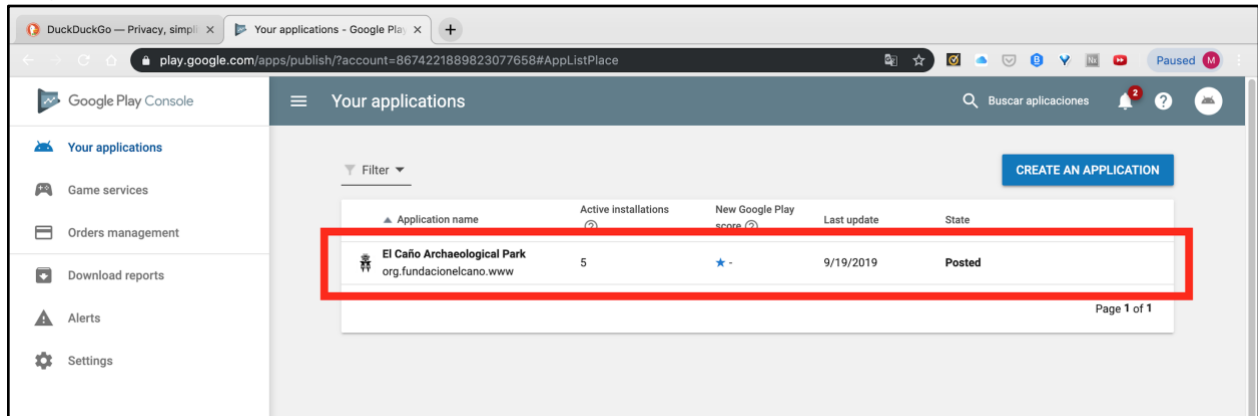
This section discusses how to change various aspects of the app's store listing pages for Apple and for Android. This includes how to change the price of the apps, how to make the app

## Demonetization and Price Changes

If Fundación El Caño would like to demonetize the app for any reason, this section succinctly describes how. Be warned that for Android, an app that has been made free can never be re-monetized. Re-monetization would likely require re-uploading the app as a new listing entirely and deleting the old one.

## Android Price Changes

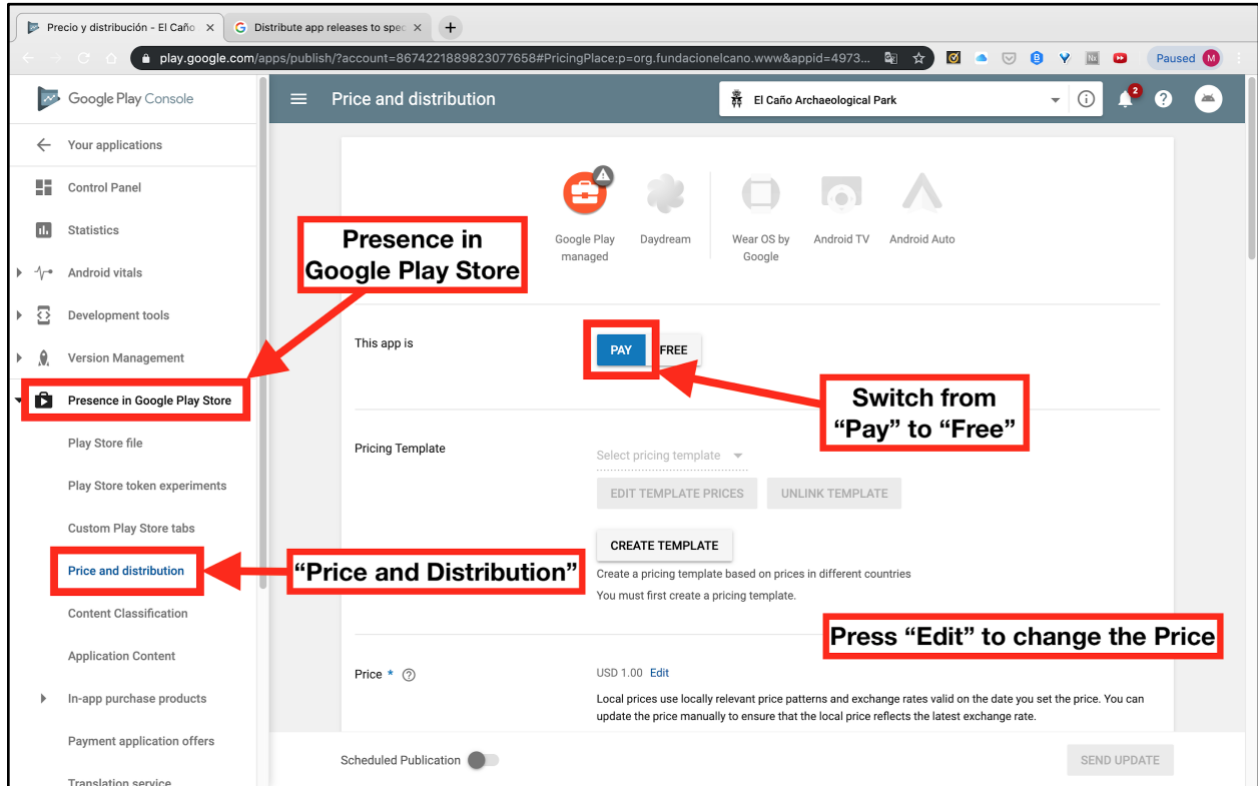
Log onto the [Google Play Console](#). Select the El Caño App in the middle of the page as shown below.



This should bring you to the app's “Control Panel.”

From the series of options on the left, select “Presence in Google Play Store” as shown below. From the dropdown beneath that, select “Price and Distribution.” This will bring you to the page shown below. You will have the option to change the app from “Paid” to “Free.” **Warning:** If you demonetize the app, there is not currently a way to remonetize it. You would have to delete the app listing from the Android app store and make a new one.

If you just want to change the price, scroll down to the Price option and select “Edit.” As of the writing of this manual, you can only change the price in whole dollar amounts.



## Apple Price Changes

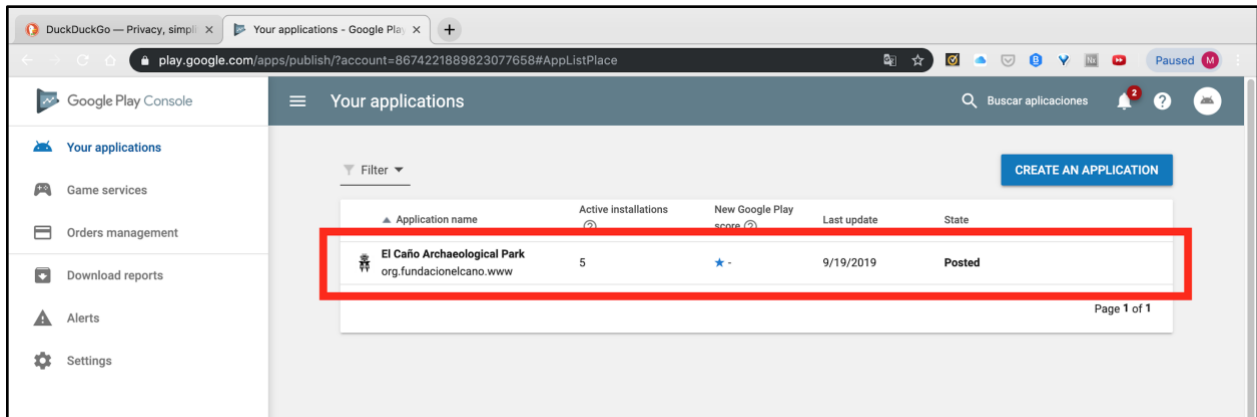
## App Listing Changes

This section covers how to change the name or photos of an app listing. For Android, this also covers how to change the countries in which the app is available.

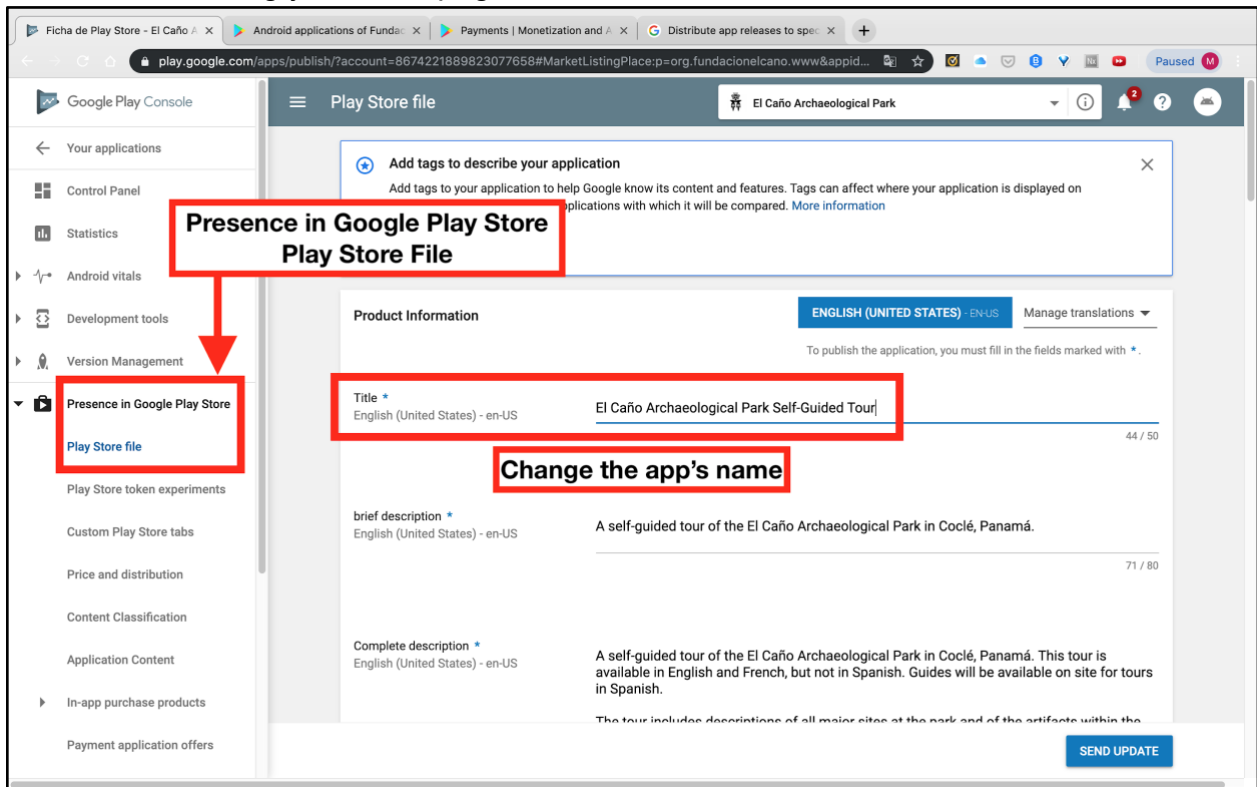
## Android App Listing

To change the listing information, log onto the [Google Play Console](#). Click the foundation’s app to get started. This will bring you to the app’s “control panel,” where you

will have a series of options on the left side of the screen.



To change the app's name as it's displayed on the app store, select the dropdown "Presence in Google Play Store." Under that dropdown, select "Play Store File." This will bring you to the page shown below.



The box in the image indicates where to change the app's name. This screen is also where to change the app's description. I believe there is a way to set a different description for a different language. We would recommend eventually setting a description of the app in French to let French-speakers know to download the app and what it's about. We would also recommend setting a description of the app in Spanish to let Spanish-speakers know not to download the app because it is not in Spanish.

You can also change the countries where the app is available. To do this, select “Presence in Google Play Store” on the left, and select “Price and Distribution” from the dropdown below that. If you scroll down on the page this bring you to, you will see a series of boxes and countries as shown in the photo below. If you change any of the countries from “Not Available” to “Available,” the app should become available in those countries within about a week or so.

The screenshot shows the Google Play Console interface for the app "El Caño Archaeological Park". The main heading is "Price and distribution". On the left sidebar, "Presence in Google Play Store" is selected, and "Price and distribution" is highlighted in a dropdown menu. The main content area shows a table of countries with their current availability status and radio buttons to toggle between "Not available" and "Available".

Country	State	Not available	Available	Local price	Taxes
Albania	Not available	<input checked="" type="radio"/>	<input type="radio"/>		
Germany	Not available	<input checked="" type="radio"/>	<input type="radio"/>		Operator Options
Angola	Not available	<input checked="" type="radio"/>	<input type="radio"/>		
Old and bearded	Not available	<input checked="" type="radio"/>	<input type="radio"/>		
Netherlands Antilles	Not available	<input checked="" type="radio"/>	<input type="radio"/>		
Saudi Arabia	Not available	<input checked="" type="radio"/>	<input type="radio"/>		Operator Options
Argentina	Not available	<input checked="" type="radio"/>	<input type="radio"/>		

## Generating More Screenshots

If you change the appearance of the app, and if you would like to generate new screenshots to show off this new appearance in store listings, there are a few ways you can do this.

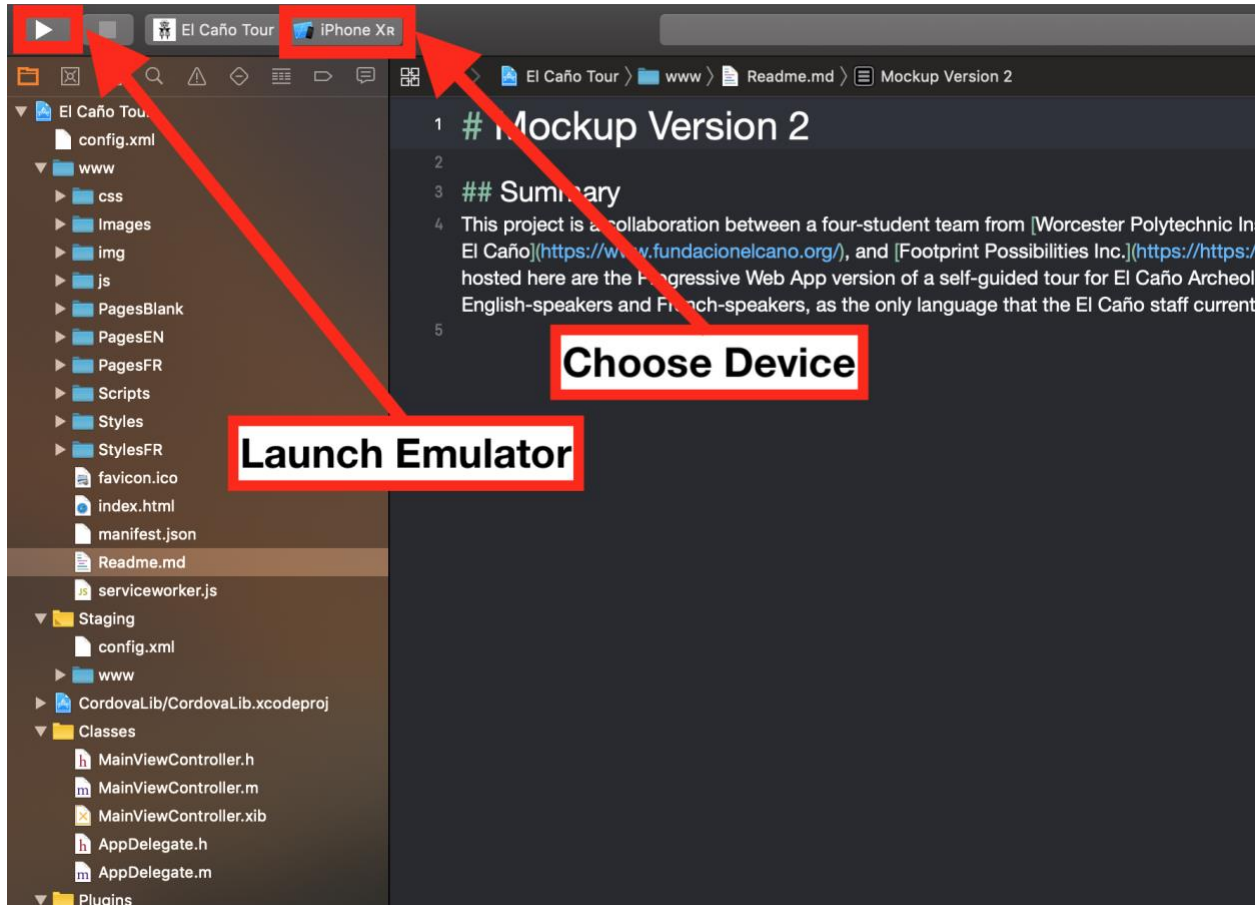
One option is to view the app in Google Chrome’s Developer Tools and to take and trim screenshots. How to use these tools is described in [another section of this manual](#).

The other option is to use emulators. An emulator is something that simulates one type of computing machine on another. In this case, the emulators are for [Android](#) and Apple. The Apple emulator is hard to find documentation for, but steps for it are shown below.

First, launch Xcode with the openiOS script using the following command:

```
~/ElCañoApp/openiOS
```

That should bring you to the screen shown below. From this screen, you can select the model of phone you want to emulate, then you can launch the emulator from the play button.



## Reverting to a Progressive Web App

If you run into an issue in updating the app, and you cannot find the solution for it by googling error messages, this manual may no longer be a viable method for keeping the app up to date. If this is the case, the webpages that comprise the app can still be served from online as a Progressive Web App that users can download from their browsers.

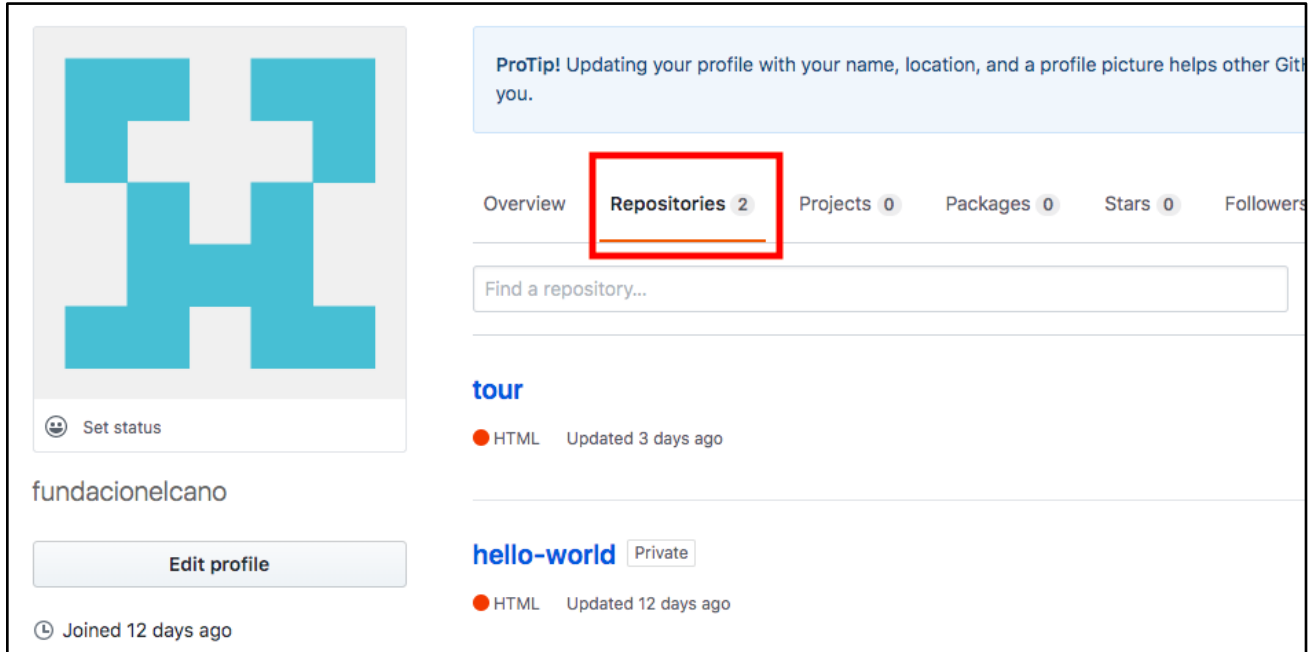
## Switching on GitHub Pages

The Foundation has a [GitHub account](#). The GitHub account has the web pages that comprise the app uploaded into it. GitHub has a service called GitHub Pages that

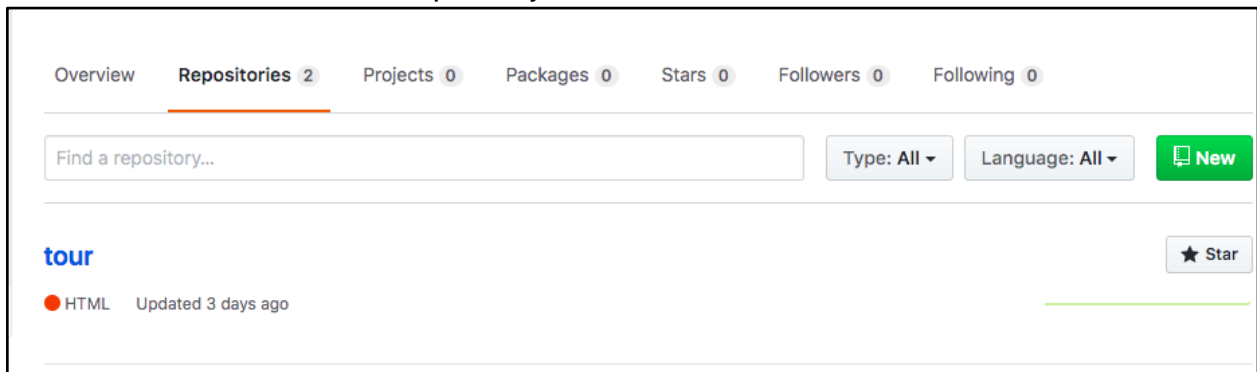


can host the app for free. At the completion of the project the pages were taken down to accommodate the use of the native application. In order to relaunch the progressive web application log into GitHub Pages and go to the profile page.

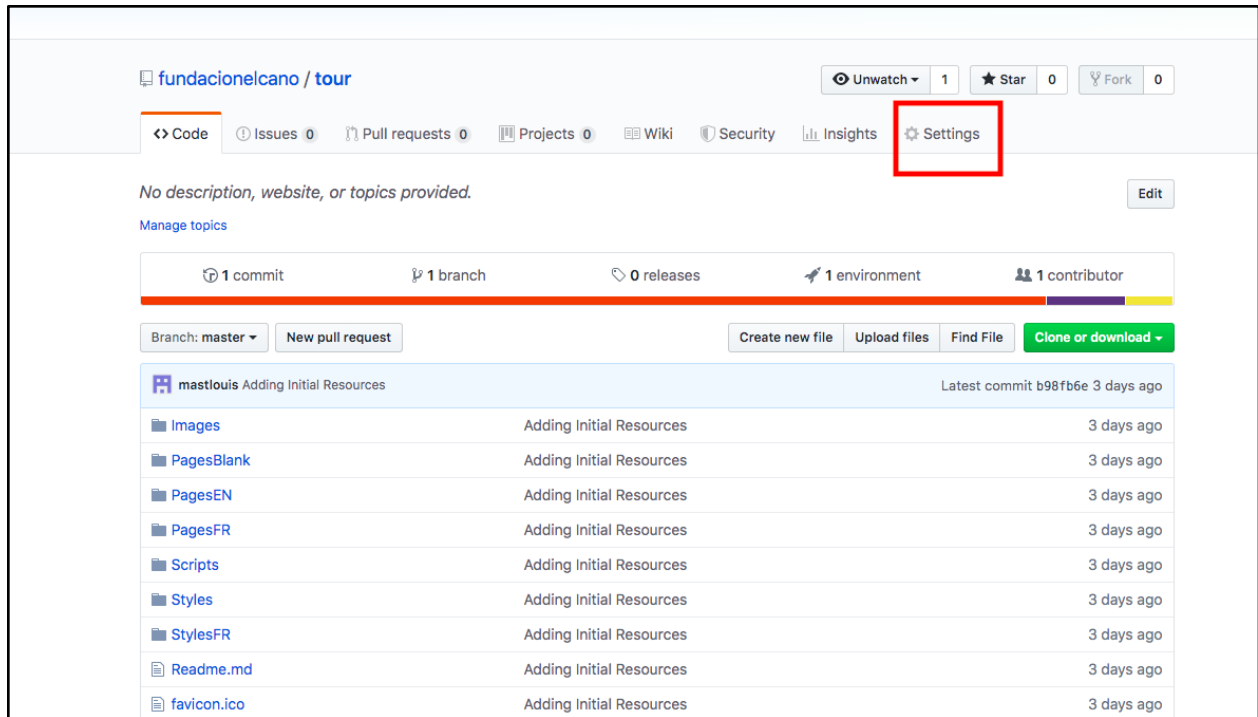
From the profile page click on Repositories.



From there click on the tour repository.



Then click on Settings.



fundacionelcano / tour

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights Settings

No description, website, or topics provided. Edit

Manage topics

1 commit 1 branch 0 releases 1 environment 1 contributor

Branch: master New pull request Create new file Upload files Find File Clone or download

mastlouis Adding Initial Resources		Latest commit b98fb6e 3 days ago
Images	Adding Initial Resources	3 days ago
PagesBlank	Adding Initial Resources	3 days ago
PagesEN	Adding Initial Resources	3 days ago
PagesFR	Adding Initial Resources	3 days ago
Scripts	Adding Initial Resources	3 days ago
Styles	Adding Initial Resources	3 days ago
StylesFR	Adding Initial Resources	3 days ago
Readme.md	Adding Initial Resources	3 days ago
favicon.ico	Adding Initial Resources	3 days ago

Finally scroll down to the header “GitHub Pages”. There is a drop down menu where you will switch from “none” to “master branch”.

## GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

✓ Your site is published at <https://fundacionelcano.github.io/tour/>

**Source**  
Your GitHub Pages site is currently being built from the master branch. [Learn more.](#)

master branch ▾

Select source

- ✓ master branch  
Use the master branch for GitHub Pages.
- master branch /docs folder  
Use only the /docs folder for GitHub Pages.
- None  
Disable GitHub Pages.

Save

**Enforce HTTPS**  
— Required for your site because you are using the default domain (fundacionelcano.github.io)

HTTPS provides a layer of encryption that prevents others from snooping on or tampering with traffic to your site. When HTTPS is enforced, your site will only be served over HTTPS. [Learn more.](#)

These steps will re-establish the functioning Progressive Web application.

## Modifying the PWA

If you want to make changes to the Progressive Web App, you'll need to make changes to a folder on your computer, then transfer those changes to GitHub. To start this process, you will clone the folder onto your local computer. You can execute the gitInitiate script with the following command:

```
~/ElCañoApp/Scripts/gitInitiate
```

This script will set your computer's git credentials to those of the foundation, and it will clone the repository into a folder called "tour" that will be in the ElCañoApp directory. The script is only meant to execute once, so it removes its own execution permissions. To make the script executable again, run the command below:

```
chmod 0700 ~/ElCañoApp/Scripts/gitInitiate
```

Once you have cloned the tour folder using the gitInitiate script, you should be able to edit the folder using [Microsoft Visual Studio Code](#) (or any other text editor). Once these updates are made, you can [transfer them to GitHub](#).

Since Git<sup>7</sup> is a version management software, it can break down if different people have made different changes to the same lines of the same files. This is called a Merge Conflict. The easiest way to avoid merge conflicts is to only have the tour folder on one computer at a time. When you want to give the tour folder to a new computer, you can delete it on the old one and run the gitInitiate script on the new one (assuming the new computer has the file system designed for this project).

## Managing the Serviceworker

The service worker is essential for the progressive web application to work properly. The serviceworker worked as it was last left, but if anyone has added, moved, or renamed files in the Progressive Web App, you will need to update the serviceworker. Every file that you want included in the progressive web app must be listed with absolute file paths. If an absolute file path is listed that does not exist, the progressive web app will crash.

<sup>7</sup> People who are new to Git and GitHub often confuse the two or don't know the difference. Git is a version control software that can track a branching path of development. GitHub is an online service that hosts a Git repository (folder) and that adds a few neat features onto Git using its existing commands (e.g. a Pull Request).

```

var urlsToCache = [
  'https://mastlouis.github.io/SelfGuidedTour/',
  'https://mastlouis.github.io/SelfGuidedTour/favicon.ico',
  'https://mastlouis.github.io/SelfGuidedTour/index.html',
  'https://mastlouis.github.io/SelfGuidedTour/manifest.json',
  'https://mastlouis.github.io/SelfGuidedTour/serviceworker.js',

  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/Area1Excavation.html',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/index.html',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/InformationPanels.html',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/Mound3.html',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/museummap.html',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/parkmap.html',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/Riverrockcauseway.html',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/Rowofmonoliths.html',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/Sculptures.html',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/sponsor.html',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/ticketbooth.html',

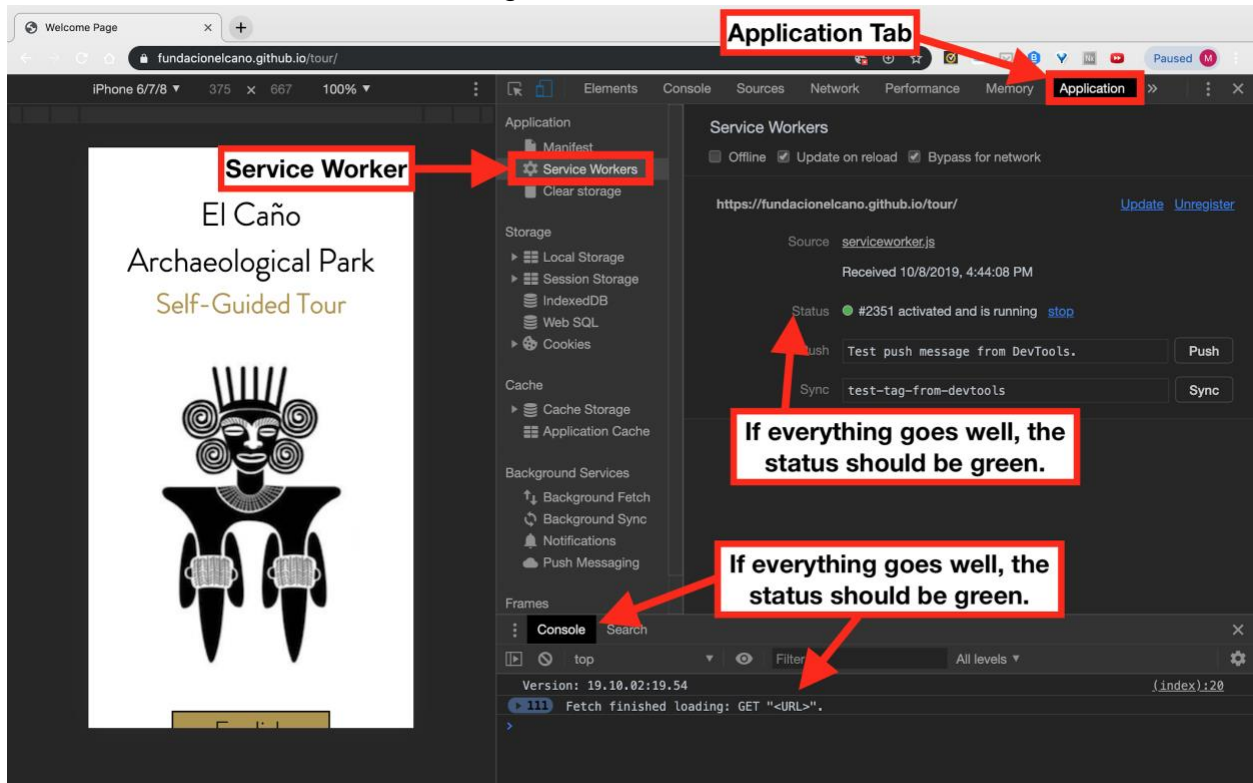
  //English Museum
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/Museum/AS-V1.html',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/Museum/AS-V2.html',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/Museum/Panel1.html',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/Museum/Panel2.html',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/Museum/Panel3.html',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/Museum/Panel4.html',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/Museum/Panel5.html',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/Museum/Panel6.html',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/Museum/Panel7.html',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/Museum/SP-V1.html',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/Museum/SP-V2.html',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/Museum/SP-V3.html',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/Museum/SP-V4.html',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/Museum/SP-V5.html',
  'https://mastlouis.github.io/SelfGuidedTour/PagesEN/Museum/SP-V6789.html',

```

Each file path must be listed in single quotes with a comma written after each entry. The last file to cache should not have a comma after it. This structure is vital for the serviceworker to function properly. The titles in green are simply for the structure of the document and are commented out of the code using the //.

If you would like to test the service worker to make sure that it still works, the easiest way to do so is with the Google Chrome Developer Tools. The results of this test will only be valid [if GitHub Pages is switched on](https://mastlouis.github.io/SelfGuidedTour/). If it is, you should be able to access the Progressive Web App from <https://fundacionelcano.github.io/tour>. Once you're there, one way to open Developer Tools is by pressing the “option”, “command” and “i” keys together. Another way is by pressing F12 (or fn and F12 together, for most users).

There should be a series of tabs to the right. To see whether or not the serviceworker is registering, click the Applications Tab. From the list on the left, select Serviceworker as shown in the image below.



If the serviceworker registers successfully, the status will be green.

You can see errors in the console below or in the network tab. To see each resource load, you can use the network tab. If you want to test the service worker repeatedly, disable the cache as shown below. To test offline functionality, re-enable the cache.

**Check "Disable Cache" if you want to test the serviceworker.**

**If anything is red, the resource did not load. This may be because the name on the serviceworker was wrong.**

Name	Size	Time
SP-V4.JPG	172 KB	14.09 s
SP-V5-2.JPG	344 KB	14.74 s
SP-V7.jpg	327 KB	15.93 s
SP-V8.PNG	448 KB	16.51 s
HighRes.png	114 KB	1.05 s
MedRes.png	73.0 KB	16.54 s
area1excav...	168 KB	16.88 s
InfoPanel1.j...	95.9 KB	17.16 s
monoliths&t...	196 KB	17.49 s
Mound3.2.j...	103 KB	17.74 s
mound3tom...	101 KB	17.94 s
ParkMap.png	350 KB	18.78 s
ParkMapFre...	1.6 MB	22.08 s

This can show you problems in the serviceworker. The most common error is that the file names in the serviceworker may be incorrect, which would cause them to request resources that don't exist. This would cause the entire cache to fail. If this happens, correct the file names that show up in red in the Network Tab so that they're spelled accurately in the service worker.

Google has plenty of resources on [what a service worker is](#), [debugging service workers](#), [how to use promises](#), and how to use the [Chrome Developer Tools](#). Editing the service worker beyond adding, removing, or renaming files will be exceedingly complicated, but if you would like to, you will need to [learn JavaScript](#).

## Updating with Git

Because the progressive web app is written in web languages, you should never really need to update it. However, if you want to update the progressive web app to change the appearance or to add content, these are the steps to follow.

The progressive web app's code is hosted on GitHub, and the progressive web app uses Git for version control. Git is a very useful software for version control, but it can be difficult to learn. To simplify the process, we made an update script that will



transfer the changes you made to GitHub. To use the script, type the following command:

```
~/ElCañoApp/Scripts/gitUpdate
```

You may be prompted for your username and password on GitHub. The username should theoretically have been set by the `gitInitiate` script, and the instructions for setting the password print when the `gitInitiate` script runs (read the script's source code to see these instructions).

If you want to learn more about Git and GitHub, read through the [Git Documentation](#). Git is incredibly robust for users who have a thorough understanding of its commands and internals, but it can be frustratingly fragile to everyone else. Remember that the PWA tour folder should only be edited on one computer at a time.