Bachelor's thesis

Degree programme in Information and Communications Technology

KVTES16

2017

Jorge Bueno

DEVELOPMENT OF UNITY 3D MODULE FOR REST API INTEGRATION

– Unity 3D and REST API Technology

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and Communications Technology

2017 | 31

Jorge Bueno

DEVELOPMENT OF UNITY 3D MODULE FOR REST API INTEGRATION

- Unity 3D and REST API Technology

Communications constitute an important feature in current technologies, they keep leading the studies and research projects as technology evolves. In this context videogame development tools, such as Unity 3D, include new tools that might be useful for other purposes than what they were meant to be used for when they were developed.

Through the study of HTTP protocol, and several other communications elements this thesis gives information about how Unity deals with this HTTP protocol interactions when trying to communicate with an API that follows the REST constrains. The thesis also includes research about how Unity 3D handles data formats such as JSON as well as the management of plugins or modules so that code from different projects can be easily reusable.

KEYWORDS:

data communication, software architectures, protocols, information technology, REST, API

CONTENTS

FIGURES

LIST OF ABBREVIATIONS

HTTP	Hypertext Transfer Protocol
API	Application Programming Interface
JSON	JavaScript Object Notation
REST	Representational State Transfer

1 INTRODUCTION

1.1 Objetive, Scope and Structure of the Thesis

The objective of this thesis is to analyze how data can be transferred between a Unity 3D mobile application and a Restful API, located on a remote server, to avoid information to be stored on the device itself and requiring such data transfer to be executed on-demand.

Data transference is rather globalized nowadays, a lot of information can be found through different sources. This document tries to focus on those specific elements that will be critical to solve the problem suggested as topic for this thesis. Due to this fact, this thesis includes information about HTTP, being one of the main communication protocols that will influence the communication between Unity and the API. Besides that, is necessary to include data about how Unity implements this protocol as well as knowing how the data may be formatted so that the API is able to send information that Unity can analyze and understand. Furthermore, including data about REST services and how they are supposed to work will give the perfect context to understand and solve this communication issue.

This thesis introduce overviews of broad subjects such data structures and communication protocols, needed as part of the research, as well as an introduction to Unity technology and its core functionality in relation to data transfer and processing. However, this thesis is not a guideline or tutorial for any of these main technologies related to the objective and outcome. The scope of this thesis focuses on solving the problem raised by the research question through the use of such technologies.

At the time of writing of this thesis, the official stable release of Unity used for this thesis is version 5.5.1, besides Microsoft Visual Studio 2015 is the tool used for the development of Unity C# Scripts. In order to simulate interaction with a real API, in this thesis will be used endpoints from free access APIs, as this thesis is not meant to explain or analyze problems that could be derived from accessing an specific private API. Although some guidelines would be given about that topic, each case could be different therefore there is no point in giving a general solution. Besides this point, and although some guidelines will be given about which certain data structures might

generate better interaction between server and client, this thesis is not suppose to analyze latency or bandwidth problems either.

The structure of this document is divided in 6 chapters. The first chapter gives a generalist description and introduction to the thesis. The second chapter gives information about HTTP protocol as well as an analysis of REST services. Following, the third chapter includes the study about some common data structures that are nowadays used in several fields within the software development, adding a comparison between them. The fourth chapter introduces Unity and links it with the information seen in former chapters. The fifth chapter will provide a practical context to the theorical matters showed in this document. Finally in the sixth chapter will be shown the conclusions about the project.

1.2 Company and Project Background

Stormcloud is a Finnish company located in Tampere which is currently working in the development of an application that displays data on a 3D environment, using Unity 3D for this purpose. As this technology relatively new, and originally oriented to videogames, information is mostly oriented to this field. Due to this, the company required researching information about how Unity might be able to interact with an API that works following the REST specifications.Besides that, the research should give enough information that Unity applications once developed using the outcome of this thesis should satisfy at least both of the following problems:

- The application is supposed to be used in mobile devices. This kind of systems are not able to store unlimited information, at least not as much as a PC, therefore using the device as storage system is not a suitable option. This limitation brings up the necessity of using an API REST as a solution.
- Another characteristic that is forced by the usage of mobile devices is that data should only be actualized when is required to do so. Trying to do something else, might interfere with the efficiency of the system and may generate problems

2 HTTP AND REST

2.1 HTTP Protocol : Context and Functionality

Hypertext Transfer Protocol or HTTP, as it is commonly known, is one of the main communication protocols implemented within the Internet protocol suite that in general makes possible the interaction and communication between all the devices connected to the Internet or other kind of networks that are limited to a much more restricted area. Being both a quite "young" protocol and one of the more extended ones over the amount of devices used all over the world, asking for the reason of its creations seems to be logical .What should be taken into account is that the development of HTTP cannot be understood on its own. This protocol was developed along with other new tecnologies in order to allow communication among different computers. HTTP and all these new technologies end up setting a common environment for devices that may not have anything in common at first sight.

HTTP functionality has been growing and evolving with the global technology since its creation. Initially meant to be a simple contribution to the solution that was supposed to solve a communication problem, this protocol would allow an easy communication between two computers running over other protocols such as TCP/IP and sharing only HTML documents. That set the bases from which HTTP would evolve into what is being used right now all over the World Wide Web.

Since 1991 until now the evolution of this communication protocol lead to the development of three different HTTP implementations. Each new version added functionalities adapted to the necessities that the global technology evolution was requiring at that moment, therefore constantly updating the problems and goals that were supposed to be solved.

The first implementation of the HTTP protocol was supposed to be simple enough that could be a starting point for other protocol developers at that time. Version 0.9 of the HTTP protocol was designed as a prototype to show some of the functionality that Tim Berners-Lee considered that the protocol was supposed to include. (Grigorik, 2016)

At this version HTTP was able to set the communication between client and server in simple steps using ASCII character strings as request lines, as well as using HTML as

response language, all of this running over a TCP/IP communication link. At this point whenever the transference of the HTML response document was complete, the communication will end.

As it was, HTTP worked, allowing the communication between a server and a client.However, and after the development of the first version, the World Wide Web and therefore the Internet started their growth which meant a change or an evolution of the characteristics that HTTP was supposed to have in order to be used as data transference protocol. This evolution lead to the creation of HTTP 1.0, developed as a response to the needs required by the fairly new Internet community.

These modifications changed HTTP structure and added functionalities, one of the most important ones is the fact that the response element had not longer to be an HTML document but whatever file type the client requested and this allowed the request of images, videos or other kind of different files. This in some way allowed the Internet to change, from complete text interface, to something more similar to what we have now, images, videos, banners and all kinds of interactive content. On the other hand, HTTP incorporated new data encodings, new characters sets and formats and other characteristics that improved the protocol in general.

After HTTP 1.0 improvements, the new HTTP protocol had improved much more compared with HTTP 0.9, adding new functionalities and characteristics that made HTTP attractive to be the option to be the standard all over internet. Yet with all the improvements that HTTP 1.0 brought, there was still room for something more and that happened soon after with HTTP 1.1, as this version is the one that has been used over the most amount of time of all the HTTP versions. HTTP 1.1 was focused on the optimization of what HTTP already able to do, as well as adding new elements that improved the communication set up with this protocol.

One of this optimizations and maybe the most important one, was that in this version changed how HTTP used TCP. With version 1.1,HTTP no longer needed to create multiple TCP connections for a single conversation. This would allow using the same TCP connection for several transmissions. Besides HTTP required an explicit end of communication request in order to shut down the TCP link.

The next HTTP version, that is the one being used nowadays was again caused by the community needs and once again it is related to introducing optimization to the protocol, this time due to the growing number of embedded systems that are using HTTP as their communication protocol. Although HTTP works fine through normal work conditions, embedded systems usually need more data sharing, and, therefore, are more sensitive to latency and delays, HTTP 2.0 is meant to solve this issues introducing improvements in how the protocol sets the communication, and gets the information.

As it can be noticed, HTTP has gone through a lot of modifications and improvements along the years, always trying to satisfy the needs of both users and developers, being the second ones who really choose HTTP instead of other protocols when creating new applications or web servers that required a communication protocol as HTTP is. Once set the context of this well known protocol there might be, at least, one more question about it that must be solved. How does HTTP work? Of course this question is the main topic of many of the several books written about this protocol. Yet it seems important to explain at least, how is the message interchange and which tools does this protocol provide when a client-server interaction is taking place.As the main action that HTTP is based on is message interchange between servers and clients, the explanation contained in this document will be based in that aspect, considering it enough to understand in a basic level how HTTP grants devices the ability to communicate between themselves.

What HTTP does is providing both server and clients the necessary tools for information interchange, as servers usually have the information that clients seek.The main tool, as in any kind of communication, are messages.Messages allow both endpoints to send and receive requests and responses, the protocol grants the ability to understand those messages, as they have certain format that both systems are going to be able to understand. And basically and HTTP communication may be composed by an interchange of request and response messages although of course there are different kind of request messages as response ones.

Either one type of message or the other these communication units are set by the protocol which specify how the message is going to be define, and what information is going to be carrying from one end to the other. Being some of that information, what can be called as meta-information as it explains the contents of the message, allowing

the receiver to understand them. HTTP messages can be divided in three different parts these being :

- Start Line : As the name shows, all HTTP message will start with this, no matter which type of message the client is sending, response or request. Although in both cases this line gives information to either the server or the client about the message. In the case of request messages, this line will give information about what is being requested with the HTTP method that has been used in order to generate the request. In the other hand if the message is a response, it will show both status codes and reason phrases that will give information about what happened, if everything went right or if there was any problems. In both cases information about the HTTP version is given.
- Headers: Headers are meant to specify more the information given about the message that can be about what the message is going to do, delimitating the information given at the start line, or which is the content of the message that it is being stored in the body. Although HTTP standard has its own headers defined each application is able to generate their own ones that might suit their system context.
- Body: It is an optional part of the message that will carry the information that the messages were design to contain.

Messages are the unit of communication that HTTP uses in order to provide the service that it was designed for. It is important to notice the difference between the actual information the message is carrying, and the HTTP information. The first one is important for the client user, the second one is important for both client and server so they can understand the message and what to do with or because of it. There are certain elements that are crucial for this protocol. Those are the actions that HTTP provide to set the interaction between server and client(Gourley, et al., 2002). These actions are represented by verbs. They are usually called HTTP methods :

- GET: Used when asking for a resource to a server
- HEAD: Similar use as the method GET, but in this case asking only for the headers that describe the element, not the resource itself.
- PUT: Writes documents to the server.
- POST: Used for sending input data to the server. Usually used in HTML forms.

- TRACE: Sends back the original message that the server received to the client, so the client can compare if the message has suffered any modifications due to firewalls or proxies.
- OPTIONS: Used by a client for asking information about the capabilities of the server.
- DELETE: Request to a server of deleting a resource URL.

2.2 REST ARCHITECTURAL STYLE

As exposed and explained in this document, HTTP by itself allows communication between servers and clients all over the Internet. Yet every time some new technology is developed, users professionals that have to work with that new element might try to improve it with sets of rules and guides so everyone can have the best out of this new component in their daily lives. HTTP was not the exception and since its creation this paradigms have been created in order to set how the communication between two network components is going to work, and more specifically using HTTP as the communication protocol or at least being one of the options to use.

One of this software arquitecture styles is known as REST which stands for Representational State Transfer. To be as exact as possible, REST should not be defined as an architecture by itself although when combined with other elements its able to generate arquitectures that follow the REST "rules", services that are designed in this way are usually called Restful. REST can be considered as a series of constraints that have to be followed when designing a web service, and that is how was described and defined at the dissertation from Roy Thomas Fielding for the University of California.(Fielding, 2000)

As stated before, Restful APIs follow specific rules or guides that create the style that REST represents, detailing those characteristics can actually be a good way of understanding what is REST, and at some extend how it really works, or how Restful services can be identified. When explaining these characteristics, it is both useful and important to remember that REST is not an architecture by itself. It is not a structure you build something on, but a tool that allows the modification of a certain system so it can follow certain rules or conditions that improve that system. Of course if the system

has not been created yet, its design can take into account REST specifications so it can take some benefit from them already in the design process not needing a modification after, yet keeping in mind how REST was developed for seems to be a good approach to this "architectural style".

The first constraint that REST specifies is that the system should follow the client-server architecture. This has several advantages, allowing to distribute the problems making clear the difference where the data is going to be and where the user is going to interact besides other characteristics that this kind of architecture allows.

Once the system has this characteristic, the server session is suppose to be stateless. This means that each message, has all the information to be understandable by both the server and the client. This certainly generates to keep sending information that has already been sent, yet it allows the system to be checked easily as you have all the information within each message. It also allows recovering from partial errors to be easier and as the server does not have to keep track of anything that is happening, it allows that part of the system to be simpler. On the other hand this might generate excessive traffic by sending redundant information, other constraints will help with this problem. In fact the next constraint helps with this issue. With this former constraint some clients are allowed to have a cache. If the elements in a specific conversation between the server and client are too big, the client could tag some of them as cacheable. Therefore those resources won't be sent again, but saved in the client to be reused. Although the problem about excessive traffic might be solved, now there might be problems related with client-server coherence, as now the system will have to compare the two versions and check that they are the same or at least similar. Next step for getting a system that follows REST guidelines is that the system should have a Uniform Interface, this constraint is one of the most representative ones as it allows to distinguish REST from other architectural styles. A Restful system will have a uniform interface between its components allowing the execution of the most usual case of communication that meaning that the usual case is going to be satisfied but not that well other kind of architectural interactions. This description might not give enough or specific information to be completely understood. That is why within Fielding's dissertation there are specified four interface constraints that help defining how to design a Uniform Interface all over the different components of the system. The first of this interface constrains refers to the identification of resources, every resource will be identify by an URI. All request that is made within the system should have this

identification of the resource that is suppose to work with. Once a client has the URI of a certain resource, it can either modify or delete the resources through the HTTP methods, it is exactly what second constrain stands for, manipulation of resources through representation. Once that the resources can be easily identified and modified, the next constraint sets that messages should give enough information about themselves so the servers will always know how to process them. The last constraint is referred to HATEOAS, which stands for Hypermedia As The Engine of Application State, description that only by itself may not give too much information.This constraint indicates that while using an API designed through the REST constrains, the output of any of the endpoints will be able to provide the user with links to other parts of the API.(Wikipedia, 2017) As this main seem a little bit abstract Wikipedia provides an example that may give more information about this constraint :

```
GET /accounts/12345 HTTP/1.1
Host: bank.example.com
Accept: application/xml
...
```

Here is the response:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: ...

<?xml version="1.0"?>
<account>
   <account_number>12345</account_number>
   <balance currency="usd">100.00</balance>
   <link rel="deposit" href="https://bank.example.com/accounts/12345/deposit" />
   <link rel="withdraw" href="https://bank.example.com/accounts/12345/withdraw" />
   <link rel="transfer" href="https://bank.example.com/accounts/12345/transfer" />
   <link rel="close" href="https://bank.example.com/accounts/12345/close" />
 </account>
```

Figure 1. Uniform Constraint Interface(Wikipedia)

Related to the uniform interface, the system should be layered. As the internet itself, a Restful system distribute its functionalities in different layers allowing a much simpler system although of course this will add latency to the communications as the message should go through all the layers, as it happens in usual communication between computers. Last but not least, Restful systems have an optional constraint, that is the availability of Code-On-Demand. In this case allowing clients to send request to the

server which ask for a certain resource. This resource would be an applet or script that the client could execute. Therefore adding that functionality only when it is required by the client and being always available at the server. This simplifies the system as there are certain functionalities that can be implemented later on, although at the same time the visibility and clearness of the system is not as good as if everything would be implemented into the client in the first place (Sandoval, 2009).

2.2.1 URI HIERARCHY

Once explained and detailed all the constraints that shape the APIs that follow REST architectural style, it is quite clear that resource identification and representation is quite an important element.

When working with APIs, the URIs is the mean to identify the elements within the API, yet on the other hand is a really good way of organize and give more information about the structure, relationships and hierarchy among all the elements that form the API.

It depends on the developer which structure the API is going to follow, therefore the information the users are going to see when working with the URIs. Each endpoint for the API, represented by an URI, will provide either a collection of elements or just one element. In general is quite easy to differentiate those two situations :

http://www.thesisexampleAPI.com/chapters

http://www.thesisexampleAPI.com/chapters/1

These two URIs represent those different scenarios, the first one if used in a GET HTTP request, it will send back as a response all the chapters from this API, in case it is specified, as it is in the second URI, it will return just the chapter standing for the id that is specificity within the last part of the URI.

Besides if there are different elements related in the structure of the API, the URI will show that hierarchy :

http://www.thesisexampleAPI.com/chapters/3/images

In this case for example that in a chapter there can be images, of course the URI could continue with the specification the images that would be required to show up, in case of a GET request, or any other action possible through HTTP request.

Another important property of the URIs that form an API is that, there may be not only one possible endpoint to each resource, depending on how it is structured:

http://www.thesisexampleAPI.com/chapters/3/images/21546

In this case the URI shows the images with id 21546, specifying that it belongs to chapter 3, yet this could be not really necessary, in case the user does not know where exactly is the image :

http://www.thesisexampleAPI.com/images/21546

In this case the user only knows that there is an image with id 21546. The resource can be accessed in different ways which provides versatility to use the API.

Working with this kind of identifiers can be quite intuitive and give a lot of information about the API itself, but that also depends on how the structure has been generated, and sometimes even the names used in a URI can be confusing too. Usually and it has been seen in former examples, in the URI there is only referents to the resources in the API, in this case chapters and images. There is not any reference to actions, as that is well covered by HTTP actions.(Fredrich, 2017) (Webber, et al., 2010)

.

3 DATA FORMATS

HTTP as a protocol and REST as the architectural style allows the interchange of data between servers and clients as it has been explained and defined in the previous chapter. Besides the rules that involve communications, provided by the protocol, and how the rules are going to be implemented, provided by the architectural style, there is still need of something else that allows communication. Both HTTP and REST have their own ways of defining, manipulating and identifying messages, yet what is left to specify is how this messages are going to be in the inside, how information is going to be stored in the server until a client requires it by the usage of the tools defined before. Within HTTP messages, the body segment can be used to send different kind of digital elements. As can be seen all over the internet when requesting a document it can be a pdf, in case we are requesting a text file, if it is an image there are several other formats. In the context of this thesis, the RESTful API will not work with those "complex" formats, yet something much more simple, as it could be a raw text data format.

Data Formats are meant to be different ways of structuring information. As the speech is adjusted in a conversation, when talking with different people, data formats allow to organize the information adjusting it to the activity that is going to be involved in. Which specific Data format is going to be used is usually a question that should be solved by the necessities that the project requires, as well as the programming language. If several tools that are being used have already implemented means to deal with certain data format, might be interesting to use that one, as coders will have more time to deal with other problems than trying to fit a strange Data Format into the tools that are being used. On the other hand if that Data Format duplicates the amount of data that is being sent, and the system needs to be fast, maybe other Data Formats allow a shorter representation of the same data. Therefore the choice, as many things within the Information Technologies, is about balance.

3.1 CSV

One of the main branches of Computer Science or at least one of the most important ones is the management and development of Databases, having a specialized language as SQL used in big databases as the Oracle one, it can be said that all of them have a common antecessor in spreadsheets programs.Being a much simpler way of storing and organizing data, still are being used and are quite practical for those professions that require this kind of data storage.

Of course nowadays databases have evolved quite a few levels over the functionalities that a spreadsheets can provide.Yet they are many times a start point for those who want to learn about computer science or just people interested in this kind of software as it can be really useful for business and economics professionals.And as SQL is a language that is directly related with databases, CSV is a data format that can be easily associated with spreadsheets.

CSV stands for comma separated values, and it is actually a quite accurate description of how this data format is supposed to organize and represent the information. In general lines CSV manage to organize the information separating the different values within the same line by comas, of course there are a series of rules that must be followed when using this Data format, as there are certain situations that might compromise the stability of this format in order to be used as a data storage tool. With this guidelines CSV became to be in the past a de facto standard, maybe one of the reasons was that Microsoft chose it to be implemented within their spreadsheet management software Microsoft Excel, along with its simplicity, which made it both easy to learn and use, as well as being quite easy to parse when being received any information formatted with CSV.

Although in general views CSV is still a valid choice when working with data that is supposed to be organize, usually when there is a specific necessity of interchanging that information between different devices or applications that need a common format in order to understand the information that is being sent, nowadays there are already more complex formats that provide more functionalities or characteristics than those that CSV is able to contribute with.

Either way, as lots of the systems designed some years ago were supposed to be using this data format, CSV is still in use. And in general a good knowledge of this

specific format might be useful when working with languages as Cobol, or when trying to translate from this data format to any of the modern options that a new system might be implementing. This may happen if the system is going to have any kind of communication with an old system that still uses CSV as their main data format. Because of this, having a basic knowledge about the CSV guidelines can get to be a practical information to know and therefore here there are described the rules that conform and describe how CSV format data should look when seen within a document:

- Besides using commas as the separator of each value within a CSV document, group of values get to be records, the first rule says that each record is represented within a single line, that implies that any representation of line change will be read as a change to the next record. With the exception of special cases.
- Each data is represented within a field and each field is separated by commas.
- Space characters might be ignored if they are the only element between the data and the commas
- If a data string contains embedded commas, will be delimited by double quotes instead of using commas
- If data string contains embedded double quotes, the delimitation of that field will be done by double quotes and the embedded double quotes must be written two times
- As well if the user wants to keep several lines within the same field, the field should be delimited by double quotes, this would be an exception to the first rule that says that a single record must be represented within a line.
- In case that the user wants to avoid the elimination of certain space characters then the field should be delimited by double quotes.
- In general fields may be always be delimited by double quotes, although the delimiters should always be discarded
- The first record within a CSV file is supposed to represent the column names of the rest of the elements within the document.

Of course these are general rules and sometimes users might not follow them exactly. Yet these are the rules that CSV formatted information should follow, with this rules the problems that could appear from the usage of this way of organizing information might be solved or at least not being as problematic as they could be without them

As every other element within computer science each data format has their own advantages and disadvantages, and CSV is not special in that sense. The main advantage that nowadays can be noticed about CSV comes from its simplicity.

Being so simple allows this data format to avoid adding too much information to what is going to be send.This difference between data sent and useful data sent can be

nowadays critical, especially when talking about systems that generate a lot information, or a constant stream of continuous information such as embedded systems.

On the other side, this might be its main disadvantage too. Now applications are much more complex programs than a spreadsheet can be, therefore sometimes they will need more functionality from the data format they are using. Because of these using CSV might translate into an increase in development code, as CSV is not complex enough for providing those kind of functionality.(Repici, 2017)

3.2 XML

XML was developed along with the creation of the Web, and therefore related with HTML, being both of them mark up languages based on the SGML or Standard Generalized Markup Language. With the creation of the web by Tim Berners Lee, it was necessary to create something that will allow the display of the data, in that sense, HTML was design and developed for that purpose, yet with that specific target this mark-up language was not suitable for data manipulation or data storage, in that way was XML created.At first being used along with HTML for web pages display, proving afterwards that it could be much more useful than that, being currently able to be used as a storage format within databases, as well as being quite used common language for data sharing between different systems or application. Besides being used to store configuration settings in certain programs, for example in Android based applications. Since its creation XML has proven to be quite useful, being nowadays widely used all over the Information Technologies world, not only on the web but in every other field that requires a simple and complete data storage system.

XML stands for extensible markup language and explained before is being used either for both storing and interchange of information.In order to do so, XML documents are suppose to be done in a certain way following strict guidelines. Being strict is quite a distinctive feature from this data format. Also when talking about XML, is necessary to understand that it is used for many different activities therefore there are many features that will not be included in this document. Within this document it will be exposed how XML is defined in a document with the basic rules that have to be followed so the system that is using the document is able to get the information from the document without any errors.

The first characteristic, that actually depends on the rest of rules, is that a XML document must be well-formed and valid. Those two adjectives are quite important as they define that the XML must follow the XML structure so it is well-formed and it should be able to be validated by a DTD or Document Type Definition document so it can be considered valid, in this case a XML schema document can be used instead depending on which features the user is looking forward to use. When using XML though, it is not requested that it is actually valid depending on what is going to be done with the XML document, yet making sure that the XML document is valid allows to solve errors and incompatibilities faster, and sometimes even avoid them directly.Specifically a DTD is suppose to specify the structure of the XML document as well as the definition of certain attributes and elements that may be included in that document. Sometimes instead of using a DTD, a XML schema can be used as it was mentioned before, being this former one written in XML as well, which is one of the main differences between both documents. DTD is defined by some specific SGML DTDs based syntax.Within both documents there will be information about which data type would be the different elements that the XML will be using.

Besides following the guidelines of either a DTD or XML Schema document, which will turn the XML document valid. This document should follow a specific structure so it can be checked as well-formed. First element that should be included within the XML document is the identification as one XML document :

<?xml version="" encoding="" standalone=""?>

This line defines the document itself as an XML, specifying both the version that is being used as well as the encoding, as well as the standalone directive that specifies if the document is supposed to be loaded by itself or with other documents. After this line, the next ones include information about those documents that the xml will be loaded with, those could be the DTD or style sheets :

<?xml-stylesheet href="stylesheet.css" type="text/css"?>

<!DOCTYPE DTDdocument SYSTEM "DTDdocument.dtd">

Now that all the introductory lines had been included into the document as the XML structure specifies, the next step would be adding the actual elements that will form the data the XML is going to store. This elements are defined by specific tags that delimit the element itself:

<element><element/>

Between those element tags, the element can contain several things as other elements, just characters or be empty just as in the example above :

<element>This is the main element<element/>

<element><number></number><letter></letter><element/>

When within an objects there are one or more objects it is said that those objects are child elements of the parent element in which they are defined, this following a tree structure. On the other hand it is possible to define empty objects with the tag:

</name>

Being name, the actual tag name's, avoiding with this having to written both start and end element tags. Element tags have certain rules that have to be follow in order to create a well-formed document. This would be that the tags cannot include any spaces or start with numbers. Also the coder has to be aware that the names are case sensitive therefore <element> and <Element> would not be the same tag. Furthermore elements can have attributes defined within the start tag of the element or if the element is empty in the start and end tag being included between single or double quotations :

<element values="up,down">

Another way of writing this would be using only elements :

<element>

<values>up<values/>

<values>down<values/>

<element/>

This way instead of attributes that modify or add information to the element, the XML structure is defining an object with two child objects, the information is going to be the same but with a different organization.

As well as in other programming languages is possible to include comments inside a XML document :

<!- Comment ->

On the other hand if it is necessary to include within the code any illegal value, such as < or >, that could generate errors when parsing the document XML has already a solution for this problems, those are entities which are strings that are supposed to be used instead of those symbols, for example instead of using <, it can be used the entity &lt. XML has already defined some entities for the most common illegal values. The ampersand on its own is actually an special character in XML where it can be used for printing any available character writing it as:

&#code;

Being the code written in either decimal or hexadecimal.

Besides all of these features, XML includes a special tag that allows the inclusion of lines that will not be parsed, this can be done using the following tag:

<![CDATA[non parsed information]]>

With all this basic information it is possible to create a well structured XML document.

With all of these features XML has become a quite popular language\data format, present all over web applications to different programs, that use it in different ways. Due to this, using XML or better said knowing how to use it has become to be quite an important ability. Due to its extension all over the digital world, it is easy to find quite efficient parsers, and languages that have already implemented ways of interacting with this kind of files, being much more easier to use than other data formats.

On the other hand, XML adds a lot of metainformation that is only useful for the program to understand what is the document that is receiving, this means that in systems where latency is important or that have a lot of data interchange maybe XML is not the best option, as it will increase even more the amount of data sent. Besides some users do not find XML as clear to use as other data formats thus using others instead of XML .(Deane & Henderson, 2004)

3.3 JSON

Tightly related with JavaScript another quite used nowadays data format is JSON which means itself JavaScript Object Notation. Being defined as a subset of this language yet being completely independent from it, this open source data format has been increasingly used over the last years, as data interchange format. Lately being used instead of XML which has been for long time the data format chosen in data interchange activities between different systems. In general one of the reason to which this format owes is popularity, is its simplicity. Being easy to work with, and still being able to give the data a proper structure, more intricate than for example CSV, but more readable than a XML document. JSON has proven to be quite useful to perform this kind of task, being used more and more all over the Internet. Due to its growing popularity most of the main programming languages started to develop built-in tools so it can be easy to implement systems including JSON parsing or management in them.

An easy way to understand how JSON is by explaining first the data types included within this format. As mentioned before both XML documents DTDs and XML schemas, allowed to define which data types defined each element used within the main XML. JSON has its on data types that can implement. Depending on the data type it would be written within a JSON in one way or another:

- strings : JSON string always go between double quotations
- number: Are written as they are, they can only be integer or a floating point.
- Objects : In order to define an object it has to be between bracket .
- Arrays: Defining an array in JSON is done through square bracket .
- booleans: Can take as value true or false, and they are defined writing those words without quotations.
- null : Json allows the creation of a null element which would be define writing null without quotations.

Once specified the data types that JSON works with, the next step is knowing how a JSON document must be structured or formatted. In general a JSON document is supposed to be a JSON object by itself, therefore everything that is contained within the document must be written within brackets. Next thing to point out is that JSON objects are suppose to be compound by different elements. This elements are formed

by pairs name/value of data. If for example there is suppose to exist an id element, this element would be defined as :

"id":" First Element"

This element is named id, and its value type is a string which is : First Element.

Of course if this element was included into a JSON document or object it should be surrounded by brackets :

{ id":" First Element"}

This can be related in certain way to the elements defined by tags in XML, although JSON does not have tags, and in XML the data type is specified by the DTD or the XML schema, not by how is formatted the value of the element.

One JSON object can be compound by many element each one can be of any of the JSON types mentioned above, allowing the JSON to become as complex as the author wants. In order to set it clear here is an example of how could a JSON document look alike:

```
{
   "DataFormats":[
      {
         "name":"CSV",
         "position":1
      },
      {
         "name":"XML",
         "position":2
      },
      {
         "name":"JSON",
         "position":3
      }
   ]
}
```

As it can be seen in the example above the main JSON object name is Data Formats, its value is specified as an array of multiple JSON objects which are compound by a string and a number. In the example it can be seen that each element has to be separated by a comma from the others whether they are JSON objects or any other kind of data type. Through this structure JSON is able to represent the same information as XML, with less data. And this is one of the main reason why a lot of developers are using JSON instead of XML, specifically when the environment is quite

sensitive to the amount of data that is going to be sent. On the other hand, even if it is actually ease to read whenever the JSON is simple enough, if a human face quite a long JSON document might be quite confusing if there is not any tool that allows to display the document on a friendly display, not just plain text.(Sriparasa, 2013)

3.4 COMPARISON

Of course there are several data formats available to use within our system, there is always the question about which one is better than the others. As a matter of fact the three data formats listed in this document are not the only ones that could be used, there are already new versions of JSON that concentrates even more the data that is being send, or many other different formats that would be suitable to be chosen depending on the necessities of the project, as it usually happens with every other technology that is going to be used, or included in it.

In general, it is not about which format is better but which is going to suit better our system, which is going to required less effort and money to implement, or even which one is going to be easier to use. This last question might seem ridiculous at some point, yet, when a professional has to spend too much time learning how to use a new technology, it is time that is not using to develop, therefore there has to be always a study that points out what is better for the project or even looking to projects that will come later on.

Along the descriptions of each of the formats has been mentioned multiple times, how simple or complex is each one. This might be one of the main features that should be checked when comparing several languages, as a format being simple, means much more than just if it is easy to use or not.

From the three formats, the one being considered the simplest might be CSV. It can be said that is the simplest as the procedure for creating a CSV document is quite simple. Yet this simplicity might lead to certain problems, one of them, for example, is that it is not easily readable for humans. As mentioned before this also happens with JSON, when the document is big enough, yet with CSV, the amount of data required in order to a CSV to be not easily readable is even less than for JSON. In this case XML strict structure, and in some way redundant display of the information through easily

differentiable tags, allows users to comfortably read and XML with not much trouble. Of course this comes at a cost of adding great amounts of data, to the information that is going to be shared, maybe through systems as wifi, that might not generate an efficient communication, if the amount of that sent is too much. Using JSON instead of XML is usually the solution that most developers had found, yet if the system is really sensitive to this kind of problems, developers could even think about using CSV as suitable format. This would mean on the other hand leaving all the functionalities that more complex formats as JSON and XML, these could be features that are implemented within the data format, or features that can be used due to the fact of using one or the other format. In case of CSV for example, is quite easy to use with spreadsheets as most of this kind of programs have already implemented ways to read CSV documents and include the information directly to the spreadsheet, yet CSV is not able to include any format to the information it is just plain data. When using XML for example, the data can be formatted in several ways, through data types, and the correct usage of the validation documents XML can provide way more functionalities than CSV and JSON, even when the former is being used instead of XML due to the reasons given before.

If considering the problematic that this document is analyzing, one of the three formats, stands out clearly as a better choice among the others,although the three of them could be used.

By definition the problematic of communication between a program and an API, tells the developer that is going to be quite an amount of constant data flowing from one side to the other throughout the whole system, because of these, could seem right choosing CSV as the format, yet is easy to see that this could not be appropriate at all. As pointed out before, the system is going to be interacting with an external API which means that most probably the data sent is going to be more complicated than just numbers, and therefore CSV can be not the best option to choose. Besides there is not implemented within Unity any tool to manage CSV files, so selecting this format will forced to chose another external tool, or developing it in order to parse and manage those files, which could easily drop even more the efficiency of the system. Discarded CSV, two options are left, yet with the different reasons given already it seems that XML might not be the best option either. With this format there is already tools built-in Unity 3D so XML documents are easily and efficiently used, but using this format would mean adding more data to the specific information being sent among the API an

Unity.This could drop the efficiency or even generate errors in specific occasions if the data flow overloads the system.

Due to all of these reasons seems that JSON is the most suitable format, as it is simple enough for a human to use, Unity already implements tools that allows a quite easy interaction with JSON objects, besides limiting the amount of extra data that will be sent when using this data format.

4 UNITY

Currently being one of the most successful game development softwares, Unity starts getting attention from not only more game developers, but other companies that may be interested in the potential of this software to create other kind of projects, such as architects, engineers or investigators of all kind, or more specifically Stormcloud, company for which this thesis is being written.

Having such a modest beginning and still being quite new compared to other elements of this research, Unity has grown to be a quite useful and important tool for game developers, mostly for those that do not have the budget that a big company can provide them off, being this,one of the main thoughts that its developers took into account when creating this software. As to step into the software development content, providing an expensive tool would not be successful at all as the game industry was leading forward small developers, who would not be able to purchase a developing tool if it was too expensive. Besides having the big companies developing their own specialized software instead of buying it, as they know what the need for developing their own games.

Since 2005, when the Unity 1.0 version was released until now, this software has been growing, adding new functionalities and adapting itself to the requirements that the users are asking for, considering that this version was only supported under Mac OS X, it seems like a long journey (Technologies, 2017)

4.1 UNITY AND HTTP

Nowadays internet connection has become quite an important feature for almost every videogame in the market. From simple mobile apps to PC games with quite high requirements.Being such an important element within the gaming community would have been quite unusual not finding any tool, included in Unity, that allowed the communication through web, more specifically a HTTP connection. Checking Unity documentation page nowadays this software provides two different tools for HTTP. The main difference is when those tools where included to the program, besides of course the functionalities that were added with the second tool. As considered important to

specify this documents analyzes both tools, in order to provide the proper information about Unity's capabilities when referred to HTTP communication :

- WWW and WWWForm : This classes included in the UnityEngine since version 4.0 grant Unity with a basic HTTP communication tool. Each element provide different functionality, while WWW alone is used to do a GET request, using both allows to do a POST request.
  Being WWW a static element, it is created through its constructor to which is specified an URL as parameter :

  WWW httpRequest = WWW(url)

  Despite doing the request the element created have different variables that are used to get the information from the request, as well as other useful information that could be interesting depending on the context. If the there is only interest in the data returned by the request, the variable .text would return as a string the data retrieved, there is also the option of using a byte array or getting textures.
  With the creation of the WWW form, there is another constructor of the WWW element that can be used :

  WWW httpRequest = WWW(url,form)

  The WWWform would add the post information to the request, both the element that is meant to be posted, and the data necessary for that specific request.

With these elements Unity community was not quite comfortable, as it only provided two types of HTTP request, GET and POST, and since the rising of REST technologies more and more people were asking about a better solution that could provide better tools when working with this kind of system. Answering those requirements Unity developers included new tools for HTTP request connection through the upgrade to the version 5.4 adding the class UnityWebRequest :

- UnityWebRequest is a much more complex tool meant to solve the problems that the former tools caused, when accessing specific elements. Or in case that users were whiling to implement other HTTP requests than GET and POST. Configured by default to achieving a GET request, this tool can be modified so it can perform most of HTTP request : GET, POST, CREATE, HEAD, PUT and DELETE.Being true that Unity also adds methods related to the gaming

community as it is normal, it also provides a really good foundation for any app that requires from specific HTTP needs, solving most of the problems that the former solution was causing (Technologies, 2017).

4.2 UNITY AND JSON

As it was explained before in this document, JSON will be the data format used both by the API and Unity so both systems display and manipulate the information in the same way, so both can understand what each other is sending.One of the reasons for choosing this data format among others, was that Unity has already a built-in tool that allows easy parsing and manipulation of JSON strings. Of course this is not the only reason for selecting JSON, yet it is quite a profitable feature in case that it can be used, as having a tool that it is already included in the system would allow spending more time in the development of the product and not in the tools for creating the product. However before making the choice of using this element, studying how it works, what are the things that can be done with it and which ones might not be implemented yet would help to save time, as if eventually the tool was not meant be used in certain way, getting to know that in the last moment would be a waste of time and resources. The tool included in the Unity Engine at version 5.3 provides coders a useful interface to either serialize or deserialize Json strings, the class JsonUtility, as it has been named, provides 3 static functions in order to meet the functionalities that it claims to be able to give in the Unity Manual. Besides those functions, it is specified within Unity documentation that JSON Utility only supports certain data types, therefore and besides having this specific tool provided by Unity developers, checking what it actually can and cannot do is required, as depending on how the specific output from the API that is being used, those functionalities might not be enough.In order to solve this problem, the development of a support class that provides JSONUtility with the functionality that It could lack of might be one solution. Also researching other solutions besides the tool that Unity provides could be suitable depending on the amount of resources and time that should be invest into that kind of request.

Analyzing JSONUtiliy class and its functions will allow to clear up the solution that would suit better the project, and would give more information about how to solve Unity's JSON string management problem, either using its own tool or importing other

solutions into a Unity project or even developing a new solution, in the case that no other option is found to be suitable enough, being this the least probable scenario.

As mentioned before JSON Utility is a class that Unity Engine provides so it can be used to manipulate JSON strings both for serializing and deserializing Json objects. This class provides three functions :

- FromJson : This function is supposed to get a string that represents json and being able to create a new object from the data that it will get from the string. Of course this has its limitations as it will not work with every possible Json, and certain conditions should be fulfilled before this function works. In the first place analyzing the software definition will show a little bit more about how it operates:

  public static T FromJson(string json);

  As it can be seen, the parameter is just a string that in this case, in order to make the function work, has to represent a JSON structure. Another interesting feature about the function is the T generic type. When using this function this type should be specified, this meaning that the type or better called class should exists beforehand. Certainly is quite improbable, mostly impossible, that a class that represents a specific JSON is already built in this kind of engine, therefore this mean that before using this function the JSON should be analyzed and then create a class that fits the JSON structure then when using the function it should look like something like this:

  JsonUtility. FromJson<JsonClass>(string json);

  Being JsonClass the one that was created by the user to fit the JSON structure. Besides that the class should be defined with the serializable label:

  [System.Serializable]

And all the types within the class should be understandable for the Unity Serializer, on the contrary the function would ignore those fields, besides Unity documentation specifies that the classes should be plain, as other kind of classes such as the one that derived from UnityEngine. Object are not supported. Besides this function can be used with another overloaded version were the type that the JSON represents is specified as a parameter.

- FromJsonOverwrite: One thing that might not been completely clear in the last function is that FromJson is used to create a new object. In case there is already an instance of an object, if there is a json that represents that object this function allows to Overwrite its attributes with new values defined by a JSON string. Although the functionality of both functions could be seen as similar there are certain differences that are quite interesting and that could be useful when using JSONUtility.In order to see those differences, checking the definition of the function as It was done with FromJson would give information about this one too:

```
public static void FromJsonOverwrite(string json,object objectToOverwrite)
```

First difference that can be seen, is that this function does not return any value. Seems quite obvious as the information will be kept in the object that is set as a parameter for the function instead of creating a new one. It is important to mention that Unity Documentation says that this class allows the usage of classes that derived from UnityEngine. Object, which was not supposed to be done with FromJson, therefore in certain occasions might be useful creating an object and overwriting its values instead of creating it directly through FromJson function. On the other hand this function keeps the limitations about type restriction. Another interesting feature of this function is that the object that is set as parameter which values are going to be overwritten can be represented by "this", in case the function is used from inside the class :

```
public void refresh (string newData){
JsonUtility.FromJsonOverwrite(newData,this)
}
```

Being this code written as a function of the class that represents the instance of the object represented by the parameter "this".

- ToJson : This is the last function that JsonUtility provides in order to manipulate Json strings, in this case specifically it allows creating a new JSON from the instance of an object that has been already created. As it is specified in Unity documentation this functions follows similar limitations as the former one, being able to work with classes that derive from MonoBehaviour and ScriptableObject or plan classes or structs that are defined with the label serializable, yet not being able to deal with arrays itself, and serializing only the public instance of primitive types, being usually the result an empty object. This function is overloaded with two possible definitions :

```
public static string toJson(object obj);
public static string toJson(object obj,bool prettyPrint);
```

Being the difference just that the second definition allows the json string to be formatted.

With the description of these functions it is easy to see that even if Unity provides with tools, so users can have an easy interaction with Json string without having to create a solution for it, it is still limited. And it can be seen that for those users that will be working with a complex JSON that includes arrays, using JsonUtility might not be the better option, on the other hand, and as it is specified in Unity documentation, it is possible to interact with arrays defined within a Json by adding a Wrapper class that allows that interaction

4.2.1 EXISTING LIBRARIES

With all the information gathered about the tools provided by Unity for Json interaction, has been proved that depending on the context this elements might not be enough in order to work properly with the input obtained by the API, as JSON structures generated by these systems tend to be usually complicated and often using data types that JSONUtility might not be able to Serialize. Despite of the possibility of not being able to use Unity's native tools, there is always the option of importing other tools that might perform better under certain conditions. Choosing Unity as a software to work

with does not only depend on the functionality that Unity itself is able to grant the user with, but whether it is able to get better services by importing external tools or not, it is a feature that should be taken into account when choosing a software. About this matter, Unity grants the opportunity to include the libraries that are needed in the project folder, after that those libraries can be accessed from the scripts so they can be used as every other function from UnityEngine, or those from the C# or JavaScript versions that Unity includes. Besides Unity allows the creation of a folder, that if named as Plugin, allows the import of new functionalities that can be added directly to Unity, depending on which kind of plug-in the user is interested to include, native or not, this feature would require the payment of the plus, or pro version.

In the case of JSON string manipulation there are a lot of libraries that can be imported into Unity, that would add more functionalities than those ones provided by JSONUtility in the first place, solution that a lot of users consider to implement as soon as the JSON string becomes complex enough or the application is suppose to manipulate the string in a more detailed way. As a sample of what kind of libraries can be used in Unity this document will include some information about LitJson and SimpleJson, although other libraries that could be useful might be pointed out at the end of this section

- LitJson: Maybe not the latest solution for Json manipulation, jet quite suitable for the purpose of this document.The last version available for this software is 0.9 released on 2014 .There has been no upgrades for this software since then. This library claims to be a small and fast solution for JSON handling.Besides that features, would be interesting using this library instead of Unity JSONUtility as it is possible to work with more complex JSON strings, such as strings that include arrays or dictionaries. When working with LitJson, user has two options, as it provides both a low and high level interfaces in order to manipulate JSON strings, being the former one build using the low level interface. High level interface is represented by the JsonMapper class which provides two methods:

  JsonMapper.ToObject<ObjectType>(string json)
  JsonMapper.ToJson(Object object)

  Not being compulsory specifying the type in the ToObject function.In the case that this happens the function will return a JsonData typed object. With these

two methods, LitJson provides the same functionality as JsonUtility from Unity besides adding the compatibility to work with dictionaries and arrays .

Furthermore the low level interface might be useful for those developers who have greater skills or knowledge about how to handle JSON strings, this interface is compound by two classes JsonWriter, and JsonReader, which provide the functionality to either get or create JSON strings as well as the high interface class, but with more specific details that might not be used by everyone despite being useful in certain contexts.

In order to use this library within Unity, it is only required placing the LitJson.dll file that can be find in the official GitHub page (https://lbv.github.io/litjson/), into either the Assets folder directly, or creating a plug-in folder within our Unity project and locating the file there, besides the file should be accessible at the references from Visual Studio.

- SimpleJson: Wherever it is check SimpleJson libraries claim to be an easy solution to use, Whether they are python or C# oriented. Realizing that there were multiple versions from this software I just chose one of them checking the version developed by Nathan Totten, Jim Zimmerman and Prabir Shrestha. With this version as well as with the other tools the user is able to serialize and deserialize JSON objects. Yet the result in this case was not as satisfactory as in the case of LitJson as in order to get the code right I had to cast the results to specific data types which usually is not a great solution.

Having the option of using external libraries provides with a wider range of options to Unity. However not every library that can be found through the internet can be useful, for example, the version of SimpleJson that was tested for this document. Yet sometimes, it can be found something that allows a better solution to a problem that the tools that Unity gives, like LitJson.

Besides those tools for Json manipulation there are other that are usually mention when talking about Json and Unity. One of them being quite popular is Json.Net which is available both for free and paying for the asset in the Unity store. Besides that library others as MiniJson or even other versions of SimpleJson can be used. In general the choice between all the options available will depend on the functionality that the project will need to use. Having so many options makes Unity suitable to be used in a wider range of projects. Choosing the specific elements required in each case (Technologies, 2017).

4.3 PACKAGES AND PLUGINS

As seen in this chapter, Unity whether by itself or using complementary software, is able to provide tools that make possible the interaction with RESTful APIs, as well as being able to understand and manipulate the information that those APIs would retrieve. Yet this capacity might not be useful if there are no means to export those elements so they can be reused as long as the API does not change, as well as the functionality required from the project in which is going to be used.

In this case, and researching again through Unity documentation seems to be two possible ways that Unity projects share elements, without having to modify elements from those projects, or having to work again to create the module. As said before, this is only possible if the API remains the same, and the project will use the module in the same way as the other project.

On one hand Unity has the capability of importing Plugins.Creating a folder inside the premade Assets folder within a Unity project, provides the system to recognize the software in that folder as external, and load it in so it can be used later on for scripting purposes.

In this case the files contained in this folder, are usually code that is external to Unity, functionalities that are not included within Unity.For example the dll files that were imported into Unity when using LitJson. Those files can be included as a plug-in in this folder so Unity recognizes them.

On the other hand Unity allows to export assets from one project to other, checking the dependencies so everything remains stable. This means, that once created the scripts that grants the project with REST API communication, selecting those files to export as a unity package would allow to import them again in other project, easily without the necessity of checking that all the dependencies are all right, as Unity itself would have included in the exportation those files the script is dependent of (Technologies, 2017).

5 PRACTICAL APPROACH

In general this document contains plenty information about how Unity can be able to communicate through HTTP protocol with a REST API. Providing an example of how this can be achieved using Unity tools, might provide a little more information, and prove that it is indeed possible, giving a practical demonstration of the data displayed before. When working with Unity, the first step is of course creating a project, this will lead the user to the Unity interface where all the Assets, scenes and in general different materials can be managed and modified.

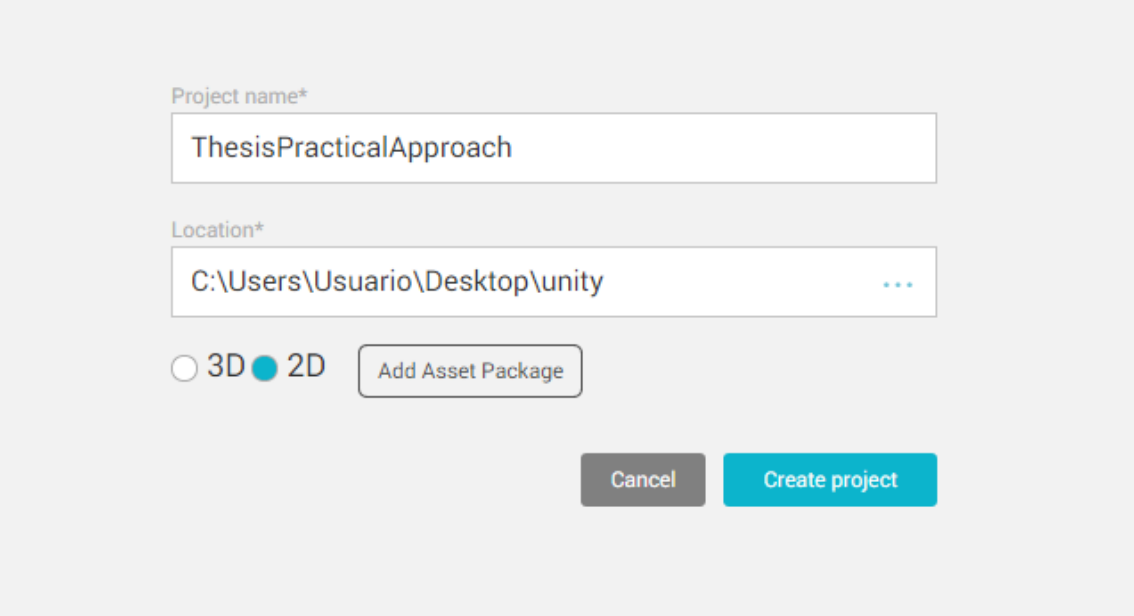

Figure 2. Creating a new Unity Project

This interface allows direct interaction with the elements of a scene, instead of coding. In this case, in order to create a project that can interact in some way with an API, it is necessary to create a script, which will be storage in the Assets folder within our project:

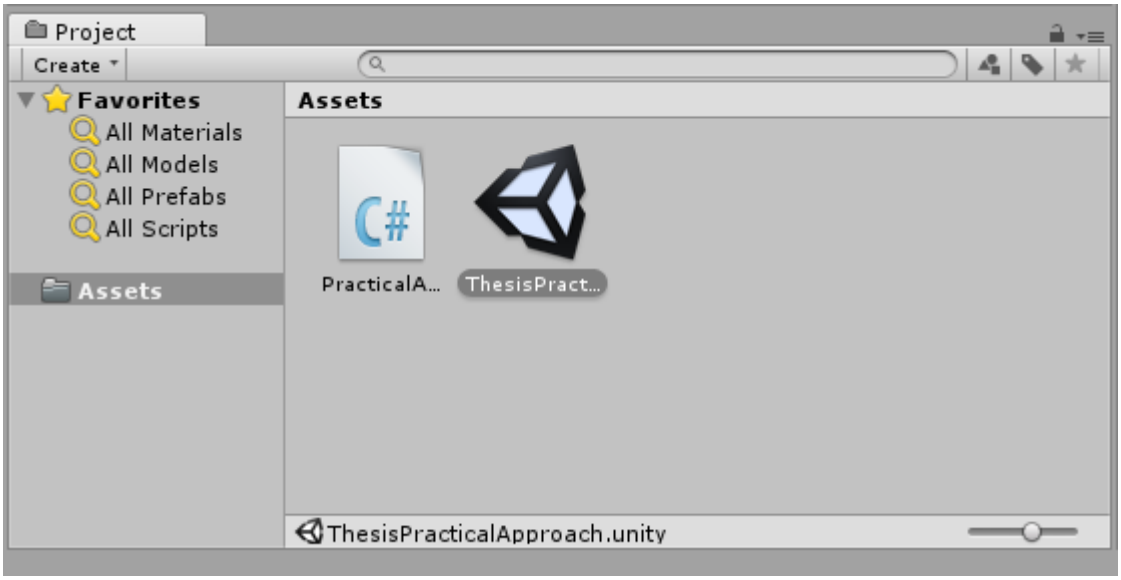

Figure 3. Assets folder

Is in this kind of scripts where the different functionalities of the project can be coded. In this case a simple example for an interaction with a free access API. For that mean, https://jsonplaceholder.typicode.com will be used as a free access solution, as this is just a practical demonstration, it is not relevant to use veridical data, in this case jsonplaceholder provides with a considerate amount of random data that can be used for testing or demonstration purposes

Once the script is created, it will contain just a class named after the name of the script that inherits from MonoBehavior, besides that two methods, start and update will be always created by default. As Unity scripts are supposed to be attached to specific objects, there is usually need of using this to methods. So the script is executed when the object is initiated or its state changes.

As this script is going to manage JSON structures, it is required to create a second internal class that allows the json strings to be turned into an object :

```
[System.Serializable]
public class Post
{
    public int userId;
    public int id;
    public string title;
    public string body;
}
```

Figure 4. Internal class for Json representation.

With this class created, now the methods to get the information from the API should be created, in this case the two methods that will be implemented are GET and DELETE. The other methods could be implemented in similar ways as this two, furthermore this two methods are prove enough to see that Unity is actually able to communicate with the API, and manipulate its output turning it into an object within the application.

```
public class PracticalApproach : MonoBehaviour
{

    void Start()
    {
        string apiEndpoint = "https://jsonplaceholder.typicode.com/posts";
        int postnumber = 4;
        StartCoroutine(GetRequest(apiEndpoint+"/"+postnumber));
        StartCoroutine(DeleteRequest(apiEndpoint + "/" + postnumber));

    }
```

Figure 5. Start() method

As the main class is define the first method, as mentioned before is Start(), in this case, as the "game" is only going to execute this script, it will be attached to the main camera object, so when it is initialized this method will be called executing the script. In this case this method will start two different Coroutines, one would be a Get request, the other one a delete request. In order to be able to execute this, those methods have to be implemented:

```
IEnumerator GetRequest(string url)
{
    using (UnityWebRequest www = UnityWebRequest.Get(url))
    {
        yield return www.Send();

        if (www.isError)
        {
            Debug.Log(www.error);
        }
        else
        {
            Post post = JsonUtility.FromJson<Post>(www.downloadHandler.text);


            Debug.Log("id: "+ post.id);
            Debug.Log("userId: "+post.userId);
            Debug.Log("title: "+post.title);
            Debug.Log("body: "+post.body);
        }
    }
}
```

Figure 6. Get Request

First of all, the GetRequest method will send the HTTP request to the API, using the UnityWebRequest class, after that, once the data has returned and using the JsonUtility and object with the json parameters will be created. In order to being able to see if it is working, the logger will send messages with the different elements of the JSON.

```
IEnumerator DeleteRequest(string url) {

    using (UnityWebRequest webRequest = UnityWebRequest.Delete(url)){
        yield return webRequest.Send();

        if (webRequest.isError) {
            Debug.Log(webRequest.error);
        } else {
            Debug.Log("Post deleted");
        }
    }

}
void Update()
{

}
}
```

Figure 7. Delete Request

In the case of the Delete request, besides the request itself, the log will only send a confirmation, in case everything went well. It is easy to notice that in this case, the update method is empty as there is nothing done when the object is updated.

In order to execute the script, Unity has to execute the scene where the object to which the script is attached is. After that the log messages will appear in the console :

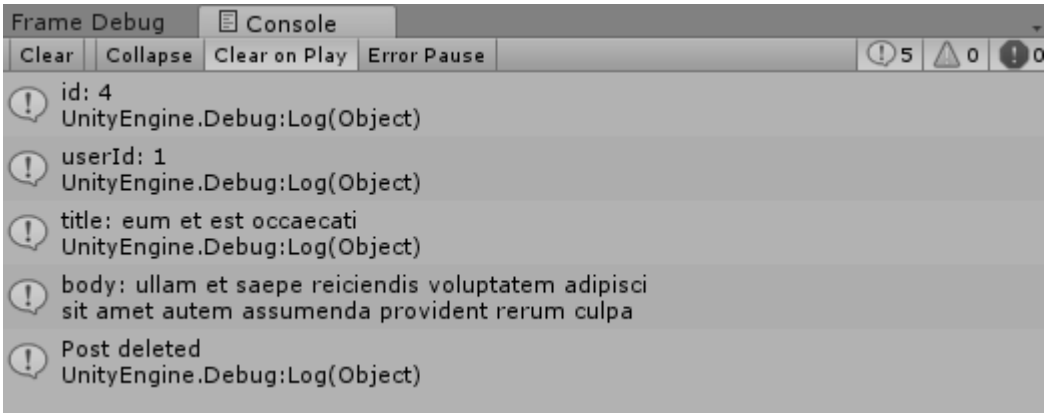

Figure 8. Debug Log Output

6 CONCLUSIONS

As far as technology keeps evolving, communication will be a problem to be solved, as it will continue evolving. Right now solutions that were conceived in a different context are still valuable and can be used for the same purpose. Of course those tools have evolved since then, as seen with the progression of the HTTP protocol through time. Although self-improvement is not always the way that technology evolves, REST was conceived much later than HTTP became a standard "de facto", yet it complements the protocol and allows that communication keeps evolving with them.

On the other hand, new technologies cannot avoid implementing solutions for those systems that are still working, maybe adapting their own tools over the time.

The main goal of this thesis was to provide enough information about the possibility of developing a module for Unity, proving whether it is possible or not. The research about this tool has lead to the conclusion that Unity provides adequate tools that allows the creation of a module that would be able to grant this program projects with communication with the API.

However, in certain aspects, these tools may not be the best solution. Therefore researching into the Unity capacities for including software from other sources, allowing new functionalities to be imported, defined even more the great potential of this tool, not only in the videogame environment, but in every other context that requires a tool as Unity.

Eventually, It was also clear that Unity is able, once the new functionality is achieved, to export those files into modules, so that they can be reused in different projects without having to deal with compatibility issues as the main program, Unity, would remain constant to the module.

REFERENCES

Deane, S. & Henderson, R., 2004. *XML made simple.* s.l.:Made Simple Books.

Fielding, R., 2000. *Architectural Styles and the Design of Network-based Software Architectures,* s.l.: s.n.

Fredrich, T. a. e. P., 2017. *REST API Tutorial.* s.l.:s.n.

Gourley, D., Totty, B. & Sayer, M., 2002. *HTTP.* s.l.:OŔeilly.

Grigorik, I., 2016. *High-performance browser networking.* s.l.:OŔeilly.

Repici, J. D., 2017. *CSV Comma Separated Value File Format - How To - Creativyst - Explored,Designed,Delivered.(sm).* s.l.:s.n.

Sandoval, J., 2009. *RESTful Java web services.* s.l.:Packt Publishing.

Sriparasa, S. S., 2013. *JavaScript and JSON essentials.* s.l.:Packt Publishing.

Technologies, U., 2017. *Unity - Scripting API: WWW.* s.l.:s.n.

Webber, J., Parastatidis, S. & Robinson, I., 2010. *REST in practice.* s.l.:OŔeilly.

Wikipedia, 2017. *Wikipedia.org.* [En línea] Available at: https://en.wikipedia.org/wiki/Representational_state_transfer