NARRADOR DE HISTORIAS PARA NIÑOS BASADO EN SUS POSIBLES SENTIMIENTOS Y GESTOS

ALEJANDRO JÁUREGUI SALAMANCA

FUNDACIÓN UNIVERSITARIA KONRAD LORENZ

FACULTAD DE MATEMÁTICAS E INGENIERÍAS

PROGRAMA DE INGENIERÍA DE SISTEMAS

BOGOTÁ D.C.

2015

NARRADOR DE HISTORIAS PARA NIÑOS BASADO EN SUS POSIBLES SENTIMIENTOS Y GESTOS

AUTOR:

ALEJANDRO JÁUREGUI SALAMANCA

Trabajo de grado para optar al título de Ingeniero de Sistemas

**Director Trabajo de grado:**

Christian Orlando Benavides Erazo

FUNDACIÓN UNIVERSITARIA KONRAD LORENZ

FACULTAD DE MATEMÁTICAS E INGENIERÍAS

PROGRAMA DE INGENIERÍA DE SISTEMAS

BOGOTÁ D.C.

2015

Narrador de historias para niños basados en sus posibles sentimientos y gestos

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Christian Orlando Benavides Erazo

# Director del proyecto

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Nombre:

# Jurado

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Nombre:

# Jurado

DEDICATORIA

Dedico este trabajo de grado a toda mi familia y amigos, quienes fueron de gran apoyo, moral, económico y espiritual en todo el desarrollo de este proyecto. También a cada una de aquellas personas que siempre están pensando en el futuro y en el desarrollo del ser humano en diferentes áreas, pues mucho de su conocimiento me fue útil para culminar este trabajo.

Espero que este trabajo le sea útil a la comunidad que quiera desarrollar proyectos bajo el mismo tema y que pueda ser referencia para el desarrollo de mil y un nuevas tecnologías en el área de videojuegos.

AGRADECIMIENTOS

Mis agradecimientos especiales a cada una de las personas que mencionaré a continuación:

* Mi director Cristian Orlando Benavides Erazo, quien me impulsó a realizar este trabajo.
* Mi profesora Angélica Veloza Suan, quien fue mi primera guía en el mundo de la programación, quien siempre me apoyó a realizar mis sueños de crear videojuegos y a la que le debo mis conocimientos bases sobre programación.
* Mi profesor Felipe Fagua Barrera, quién me enseñó todo lo que acerca del desarrollo de videojuegos se necesita, además por ser un apoyo incondicional en el desarrollo de este trabajo, siendo un asesor y guía indispensable.
* A mis padres, quien sin su apoyo cada día me hubiese rendido, pues en cada momento que algún problema surgía ellos estaban allí para darme ánimo y continuar para alcanzar mi meta.
* A mi mejor amiga Valeria Santos, quien no solo fue un apoyo constante para alcanzar el objetivo de terminar este proyecto, también por haber aportado magnificas ideas para el desarrollo de la historia que va a contar el prototipo desarrollado.
* Por último, pero no menos importante, a Dios. Pues para mí, su energía que transformada en cada uno de los aquí mencionados, logró que me llenase de fuerza para hacer posible la culminación de este ambicioso proyecto.

INDICE

## Capítulo I: Preliminares

* 1. Introducción------------------------------------------------------------------------------------------ 9
  2. Problemática-----------------------------------------------------------------------------------------10
  3. Justificación------------------------------------------------------------------------------------------11
  4. Objetivo-----------------------------------------------------------------------------------------------12
     + - Objetivo General
       - Objetivos Específicos
  5. Alcances y limitaciones del trabajo de grado-----------------------------------------------12
  6. Metodología-----------------------------------------------------------------------------------------13

## Capítulo II: Marco Referencial

* 1. Estado del arte---------------------------------------------------------------------------------------15
     + - Callas Project-------------------------------------------------------------------------------------15

Interactive StoryTeller-----------------------------------------------------------------------15

Interactive Drama----------------------------------------------------------------------------16

* + - * Narrador de historias: La historia de los afroamericanos en la computación----16
      * Act, Play or Talking about story Project----------------------------------------------------17
      * Scene Driver, una técnica perfecta para el desarrollo de historias enfocadas a niños------------------------------------------------------------------------------------------------18
      * MEXICA, el sistema capaz de reconocer emociones----------------------------------19

## Capítulo III: Desarrollo del prototipo

* 1. Requerimientos--------------------------------------------------------------------------------------20
     + - Investigación con expertos en captar la atención de los niños--------------------20
       - Reconocimiento de requerimientos funcionales y no funcionales----------------21
       - Investigación con los futuros usuarios del prototipo---------------------------------23
  2. Análisis y Diseño-----------------------------------------------------------------------------------25
     + - Evaluación comportamental: Estudio Infantil------------------------------------------26
       - Diagramas de alto nivel del prototipo----------------------------------------------------27
  3. Desarrollo-------------------------------------------------------------------------------------------29
     + - Codificación, identificación y solución de errores------------------------------------29
* WebCam----------------------------------------------------------------------------------31
* Conexión del Web service-------------------------------------------------------------34
* JSonReader-------------------------------------------------------------------------------37
* WebServiceConnection----------------------------------------------------------------44
* Patrón de diseño Command----------------------------------------------------------50
* Interfaces de usuario--------------------------------------------------------------------70

**Capitulo IV: Conclusiones y una vista al futuro**

**4.1.** Conclusiones del prototipo----------------------------------------------------------------------74

**4.2.** Desarrollos futuros posibles---------------------------------------------------------------------74

**4.3.** Referencias------------------------------------------------------------------------------------------75

RESUMEN

Este documento narra el proceso que se realizó para desarrollar un prototipo capaz de reconocer los sentimientos de una persona y con base en ellos tomar decisiones para relatar una historia. Se aplicó una metodología de desarrollo iterativa para poder controlar los cambios que el desarrollo ameritaba mientras este estaba en curso; además se implementaron patrones de diseño que optimizaron la funcionalidad y el desarrollo del prototipo.

**ABSTRACT**

This document recounts the process performed to develop a prototype able to recognize a person's feelings and make decisions based on them to tell a story. Iterative development methodology was applied to monitor changes warranted as this development was underway; also design patterns optimized functionality and prototype development were implemented.

CAPÍTULO I: PRELIMINARES

# Introducción

La tecnología es una de las formas en que el ser humano ha logrado sistematizar procesos que antes realizaba manualmente, pero poco a poco el significado de tecnología no solo abarca el hecho antes descrito; este ha sufrido diferentes metamorfosis que lo han llevado a adaptarse. La industria del Entretenimiento ha llevado a que el ser humano desarrolle cada vez más tecnología que lo transporte a otros mundos, o simplemente los haga salir de su realidad por un breve momento. Sin embargo, es posible que se cuestionen los avances tecnológicos, pues estamos en una era en la cual chocan las tradiciones; con el vanguardismo de las costumbres que han creado las generaciones más recientes.

Hace unos cuantos años, los padres solían compartir mucho más tiempo de calidad con sus hijos, pero las nuevas obligaciones que demanda la sociedad, no permite que esto suceda; no es difícil imaginar una noche familiar de hace cincuenta años; la mayoría sostenían una comunicación permanente y cuando ya era momento de descansar; los padres solían sentarse al lado de la cabecera de la cama de sus hijos; leían una historia juntos para que se transportaran a un mundo ideado por ellos con base al relato; mientras que poco a poco iban quedando dormidos. Lamentablemente esto ya no sucede o son muy pocos los que mantienen la tradición.

¿Habrá alguna forma de que la tecnología aborde este problema que a simple vista puede ser algo totalmente superficial y no muy importante, pero que en realidad es tan crucial para el desarrollo de habilidades lógicas y artísticas de un infante? Bueno, el objetivo de este documento será dar respuesta a esta interrogante, basándose en el desarrollo de un prototipo funcional.

# 1.2 Problemática

El siglo XXI, es una era en la cual se han logrado avances tecnológicos impresionantes, han facilitado muchos de los procesos que el ser humano anteriormente hacía de manera manual y poco a poco los ha ido sistematizando, logrando así un avance paulatino al momento de realizar las tareas que debe hacer. Al ser una era que está en constante cambio, produce que las nuevas generaciones transformen el paradigma de aprendizaje y como consecuencia causa que los niños absorban mucho más rápido la información.

La imaginación poco a poco se ha visto más sesgada, pues los directos implicados con la tecnología son los niños, y de alguna manera se ha afectado ese proceso superior que les permite manipular sus sentidos para crear una representación de la información que albergan en sus cerebros. Hay muchas aplicaciones que permiten el desarrollo de ciertas capacidades [1], sin embargo, aquella tradición de hace cuarenta años aproximadamente, cuando un padre se sentaba con sus hijos a contar una historia antes de dormir; se ha ido desvaneciendo y actualmente son muy pocas personas quienes la siguen practicando, esto por diferentes preocupaciones del diario vivir, sean el trabajo u otros factores que puedan influenciar directamente el no poder dedicar un tiempo de calidad a sus hijos.

Los problemas que se pueden generar cuando los padres no están pendientes de sus hijos o que simplemente compartan muy poco tiempo con ellos; son muchos, sin embargo, este trabajo de grado se enfocará en los problemas que se pueden llegar a dar cuando la imaginación de un niño no es estimulada; problemas tales como, la poca creatividad, problemas al momento de solucionar una situación en la que se vea envuelta, su dificultad para investigar e incluso su poco manejo social ante niños de la misma edad. Muchos eruditos de la historia han catalogado la imaginación como una forma inferior al pensamiento crítico y lógico que se necesita para el desarrollo de la humanidad, tanto que por mucho tiempo las escuelas buscaron y apoyaron la eliminación de la imaginación, pues era visto como un defecto. Sin embargo este pensamiento fue cambiando paulatinamente, grandes pensadores como Hegel o Kant apoyaban la idea de que la imaginación era fundamental para el desarrollo creativo y conceptual de las cosas [2].

Actualmente se cuenta con avances tecnológicos impresionantes, sin embargo, aquella tecnología, no es usada o no es totalmente aprovechada por las aplicaciones que existen en el mercado. Como se mencionó anteriormente, hay aplicaciones que se encargan de ciertas áreas de estimulación infantil, pero no hay muchas que se centren en la imaginación de un niño y utilicen tecnologías que permitan el reconocimiento de voz y gestos, por lo tanto, es necesario que se diseñe, cree e implemente una aplicación de este tipo, la cual permita que la mente del niño que utilice la aplicación, incremente su proceso de imaginación.

# 1.3. Justificación

Desarrollar este proyecto permitirá contribuir con el desarrollo infantil actual; la imaginación es crucial para el desarrollo de un individuo pues es esta la que permite moldear diferentes soluciones o construcciones a partir de una visualización anterior. Un niño que entre en contacto con una aplicación de este tipo; desarrollará facultades de una persona creativa, investigativa, analítica y sociable [3].Este proyecto se apoyará en nuevas tecnologías que buscan lograr la interacción con el usuario; rescatando tradiciones que aunque olvidadas son efectivas para el desarrollo específico de la imaginación infantil.

Los expertos afirman que fomentar la lectura o que los niños escuchen historias, estimula las diferentes formas en las cuales un niño puede soñar, permitiéndole tener diferentes visiones del mundo y dándole la facultad de crear o solucionar situaciones en las que se vea directamente relacionado; la fantasía es una de las maneras en que una persona puede llegar a trazarse metas en un futuro, una de las herramientas más importantes para el desarrollo tanto de sí mismo, como el de la humanidad.

Crear una nueva forma de estimular el proceso de la imaginación de un niño con métodos que son totalmente aprobados por expertos, y que pueden llegar a dar una cierta evolución a tradiciones que están quedando en el olvido, son más que necesarias en una era en la cual un niño está inmerso totalmente en la tecnología; dándole una forma totalmente nueva e intuitiva para que se divierta e indirectamente desarrolle el proceso sistemático de la imaginación.

# 1.4. Objetivo

## Objetivo General

Desarrollar un prototipo funcional de cuento interactivo guiado por emociones para ejecución en computadores.

## Objetivos específicos

* + - * Elaborar el estado del arte teniendo en cuenta las estrategias de desarrollo de la narrativa interactiva.
      * Desarrollar un cuento interactivo, teniendo en cuenta las emociones de felicidad, tristeza y enojo, creando los diferentes caminos que este pueda tomar dependiendo del análisis que haga el prototipo en el momento de su uso.
      * Realizar pruebas de uso de la aplicación.

# 1.5. Alcances y limitaciones del trabajo de grado

* + - * El prototipo contará solamente con un cuento a desarrollar, por lo tanto, solo presentará un escenario, en el cuál se presentará imágenes que apoyen el relato presentado a los usuarios.
      * Se elegirán como máximo tres señales multimodales para que el prototipo pueda desarrollar perfectamente los diferentes giros argumentales que pueda llegar a tener el cuento.
      * El prototipo estará enfocado a niños en el rango de edades de tres a once años, pues es en estas edades donde se considera que la imaginación debe ser estimulada al máximo.

# 1.6. Metodología

Para desarrollar el trabajo de grado se planteará una metodología en la cual se pueda trabajar de la manera más óptima y la cual permita tener un máximo control en tiempo, alcance y costo del desarrollo del prototipo.

Cuando se planteó la metodogología se pensó en utilizar una ágil, sin embargo no se cumplía con los requisitos para desarrollarla, pues se contaba con solo una persona para desarrollar el prototipo; las métodologías clásicas por otra parte no permitían un control de cambios rápido, pues pasar por todas las fases de desarrollo no era una opción, así que se opto por desarrollar el prototipo bajo una metodología iterativa, que permitiese iterar cada una de las fases del desarrollo para poder minimizar el impacto de los errores que se pudiesen presentar.

A continuación se mencionará la manera en que se planeó realizar el trabajo, se enuncian los pasos que se siguieron al momento de realizar el proyecto; también se dividen en fases para poder mostrar con mayor claridad el trabajo realizado.

**Requerimientos**

En esta etapa se realizó el proceso de recolección de información pertinente, en el cuál se puedó identificar metodologías para captar la atención de los niños, que tal vez se pudiesen implementar en el prototipo a desarrollar; para así lograr la atención total del usuario a la hora de entrar en contacto con la aplicación.

* + - 1. Pactar reuniones con docentes de primaria y/o pedagogos expertos en cómo captar la atención de niños entre tres y once años.
      2. Construir preguntas pertinentes para la recolección de información con los expertos en las metodologías infantiles anteriormente mencionadas.
      3. Levantamiento de requerimientos funcionales y no funcionales del prototipo con los docentes o los pedagogos con los cuales se haya pactado las reuniones.
      4. Reconocimiento de las historias preferidas por niños de entre tres y once años.
      5. Construir una encuesta para niños entre tres y once años donde se permita identificar sus gustos con respecto a relatos fantásticos.

**Análisis y diseño**

En esta etapa del desarrollo se analizó toda la información recolectada y la cuál será necesaria para la construcción del prototipo, cumpliendo con los objetivos identificados de las metodologías para captar la atención de los niños y adaptándola al desarrollo de la imaginación infantil.

Además de los conocimientos recolectados con los expertos, se tomará en cuenta aquella información que se logró recolectar de los niños estudiados, pues el funcionamiento del prototipo debe acoplarse con las necesidades que estos requieren.

* + - 1. Definir el rumbo de la historia que se va a contar en el prototipo.
      2. Escribir una historia preliminar con base a los datos recolectados.
      3. Mostrar la historia preliminar a niños que se encuentren en el rango de edad antes mencionado.
      4. Identificar los posibles sentimientos que pueden llegar a tener los niños al escuchar la historia y elegir 3 de ellos para construir los caminos de decisiones que tomará el prototipo cuando esté en funcionamiento.
      5. Identificar los cambios que se deban hacer a la historia con respecto a la opinión de los niños estudiados.
      6. Construir el diagrama de decisiones que manejará el prototipo al momento de presentar la historia ante el usuario, en este momento se diseñará la historia final a contar.
      7. Escribir los diferentes giros argumentales que va a tener la historia, los cuales se debe tener en cuenta el diagrama de decisiones que manejará el prototipo, los sentimientos que se evaluarán en el diagrama antes mencionado están por definir.

**Desarrollo**

En el momento del desarrollo del prototipo se realizarón entregas parciales del prototipo al director de trabajo de grado, pues de esta manera se podrán mitigar posibles errores en la entrega final del prototipo.

Para que la mitigación de errores sea más efectiva, se realizarán pruebas del prototipo con niños, para así poder identificar cualquier problema de funcionamiento y abordarlo lo más pronto posible.

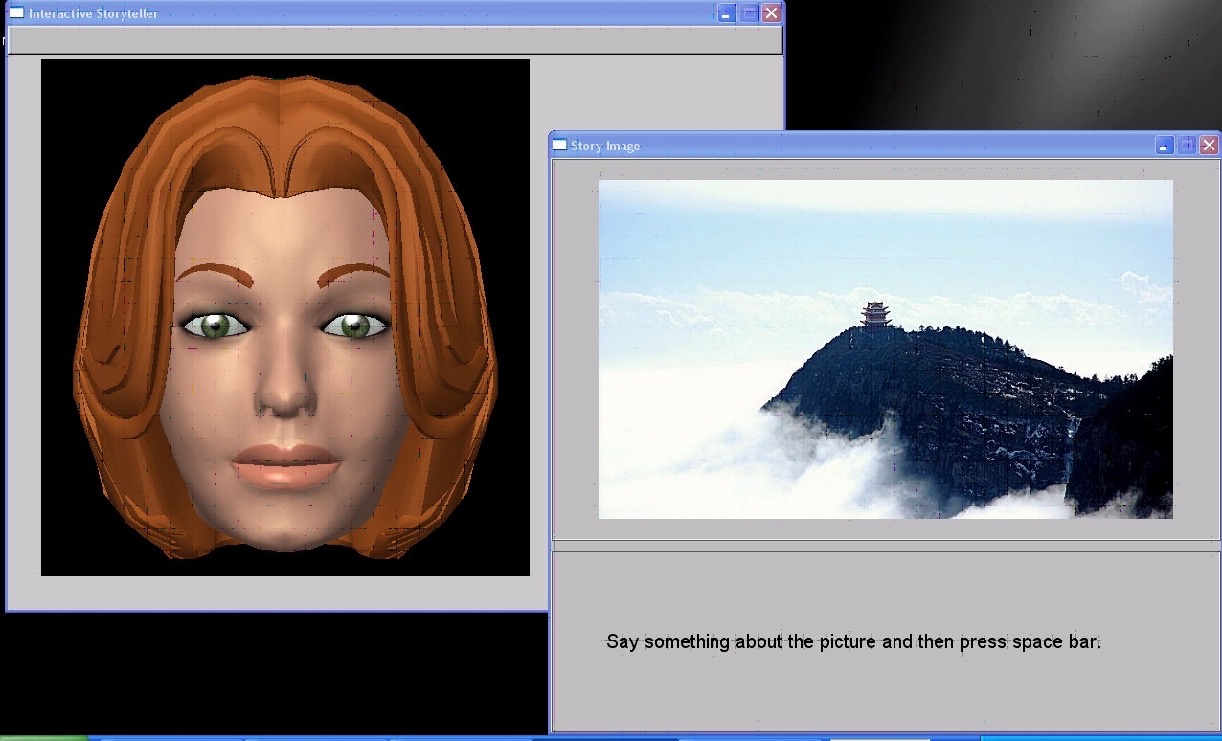
* + - 1. Investigar diferentes formas para el desarrollo de la aplicación, ya sea lenguajes de programación y/o metodologías que permitan optimizar el tiempo de desarrollo.
      2. Comenzar la fase de desarrollo de la aplicación, teniendo en cuenta que se utilizará la plataforma de desarrollo Unity.
      3. Implementar patrones de diseño que sirvan para optimizar el funcionamiento del prototipo.
      4. Resolver errores, defectos o bugs que pueda presentar la aplicación.

CAPÍTULO II: MARCO REFERENCIAL

# 2.1. Estado del Arte

* + - **CALLAS PROJECT**

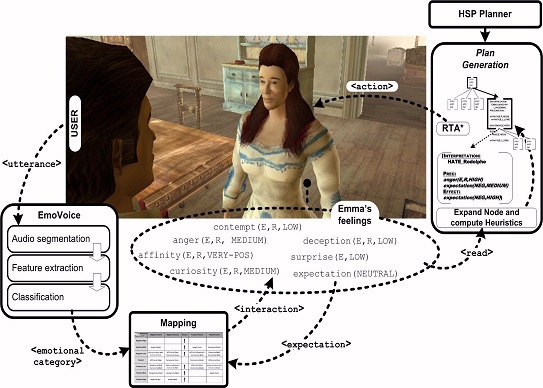
**Interactive storyteller**

Interactive Storyteller, es un proyecto que fue presentado por primera vez en la BBC Festival of Technology; la aplicación es un narrador de historias, la cual funciona a partir de los sentimientos del usuario. El funcionamiento de esta facultad se realiza presentando una serie de imágenes al usuario, dándole la oportunidad de comentarlas, es en este momento donde la tecnología de Callas, conocida como **Callas Shelf Technologies** se activa**;** lo que permite esta tecnología es analizar al usuario, teniendo en cuenta sus expresiones, para luego, dependiendo de esto tome las decisiones de que imágenes mostrar y así lograr una continuidad en la historia. Se muestra uno de los avances en la figura 1.

**Figura 1. Interacción de usuario con Interactive Storyteller** [7]

**Interactive Drama**

Interactive Drama, es un proyecto un poco más ambicioso que Interactive Storyteller, sin embargo se enfocan en el mismo objetivo, contar una historia. Al igual que en el anterior proyecto mencionado, Interactive Drama utilizó la misma tecnología logrando introducir al usuario en la historia, es decir es una interacción completa, sin necesidad de analizar los sentimientos del usuario.

Al momento de usar la aplicación, el usuario puede hablar con los personajes que se le van presentando en la historia y con eso dar diferentes giros argumentales a esta.

**Figura 2: Interacción de usuario con Interactive Drama** [7]

* **Narrador de historias: La historia de los afroamericanos en la computación**

Un proyecto ajeno al framework que se presentó anteriormente, fue desarrollado por un grupo de norte americanos quienes quieren incentivar el crecimiento en la población científica de las minorías del país; en este corto proyecto se realiza un sistema que tiene como pilar de funcionamiento el siguiente ideal **“Un narrador de historias interactivo es un medio por el cual una historia puede ser influenciada en tiempo real por el usuario del sistema”** [8]**.** El sistema buscaba realizar una historia interactiva acerca de los Afroamericanos en la computación, sus entrevistados conocían y hacían parte de la historia de aquel tema que estaba en discusión; la charla entre hombre y máquina duraba alrededor de noventa minutos y para que esta pudiese seguir su curso, el sistema reconocía gestos, discursos del usuario actual, lenguaje verbal y lenguaje no verbal, para acercarse lo más posible a la interacción entre una charla común entre dos personas cara a cara [8]. Este fue un proyecto piloto que no arrojo datos confiables, pues la muestra que se tomo era muy pequeña, sin embargo, logró avanzar en el concepto de **Interactive Storyteller.**

Poco a poco las tecnologías van avanzando y se logró realizar un proyecto que cumplía con características parecidas a las que ofrecía el framework del proyecto Callas, es así como nació un proyecto enfocado a un usuario más básico pero exigente, los niños.

* **Act, Play or Talking about story Project**

Se ha hablado por mucho tiempo que los niños deben desarrollar sus habilidades que giran en torno a la imaginación y para esto el concepto de Interactive Storyteller es perfecto. Los niños cada vez están más cerca de la tecnología y esto crea la necesidad de actualizar los medios por los cuales captar su atención. ***“Actuar, jugar o hablar sobre una historia”*** [9] es este el criterio que maneja el proyecto de este artículo. Tomaron una historia clásica y conocida a nivel mundial por grandes y chicos, Caperucita roja; el fin era lograr que sus sujetos de prueba, que en este caso eran niños con un rango de edad de ocho a once años, crearan su propia historia de Caperucita roja, cada uno de los niños tomó un rol, la abuela, Roja y el lobo feroz, representados por muñecos que podían mover encima de una pantalla táctil, la cual le sugería diferentes escenarios; la casa de Caperucita Roja, el oscuro bosque donde vive el Lobo, la casa de la dulce abuelita, un lago y una playa. A comparación con el proyecto anteriormente mencionado, este se realizó con más detalle. El experimento se realizó con dos sistemas, **Sawyer** que miraba la interacción entre usuarios y el sistema, el cual identificaba cuando los niños creaban la voz del personaje interactuando con los objetos presentados dentro del mundo, las voz creada por el niño para el personaje, pero refiriéndose a objetos del exterior, la voz del niño cuando salía del rol del personaje y por último la voz real del niño, la cual ya no estaba jugando con el sistema. El otro sistema que se utilizó en el proyecto es conocido como **Fine**, a diferencia de Sawyer, este identificaba el mundo real, es decir los participantes que estaban listos para jugar, el segundo criterio que tenía para identificar Fine, era cuando los participantes comenzaban a jugar dependiendo de las reglas y restricciones que tenía la historia; el último criterio que reconocía este sistema era la fantasía, es decir cuando sus participantes empezaban a asumir el rol del personaje que tuviesen en la historia [9].

* **Scene Driver, una técnica perfecta para el desarrollo de historias enfocadas a niños**

La narrativa digital en el uso de los videojuegos, es fundamental para saber cómo llegar a captar la intención entre niños de entre cinco y siete años, una técnica perfecta y reconocida es la de Scene-Driver [10] usadas para realizar las historias de dibujos animados en la televisión. La técnica anteriormente mencionada fue creada con la idea de crear pequeños mundos, donde los niños lograrán encontrar la coherencia de la historia y así completarla paulatinamente con sus ideas e imaginación, respetando cada una de las reglas y restricciones que este mundo presentaba. Para el diseño de historias interactivas, el concepto es perfecto, pues crea el algoritmo que el usuario va a seguir, teniendo en cuenta cada una de las restricciones, para que poco a poco vaya creando la historia como si fuesen piezas de domino pegadas una a la otra. Para que esto pueda suceder, el articulo menciona las diferentes fases por las cuales puede pasar un usuario y la historia; un usuario entra en dos fases, una activa y una pasiva, mientras una historia pasa por dos fases las cuales son la predeterminada y la generada, las cuatro fases se cruzan y generan las acciones que el usuario poco a poco va experimentando. Cuando un usuario es pasivo la historia le da una organización preestablecida con la cual podrá pasar paulatinamente a ser un usuario activo y determinará cuales son los cambios que puede hacer dependiendo de las salidas que se arrojen.

Cuando la historia poco a poco pasa de ser la predeterminada a la generada, el usuario experimenta los cambios que se van dando con respecto a las condiciones preestablecidas de la historia, siendo aún un usuario pasivo que rápidamente pasa a ser activo cuando la historia al fin se convierte en un relato totalmente dinámico y se evidencia la interacción que se dio en el proceso [10].

* **MEXICA, el sistema capaz de reconocer emociones**

Para el desarrollo de este proyecto es fundamental el lograr reconocer las posibles emociones que puede llegar a tener un usuario al momento de usar la aplicación, esto es fundamental a la hora de realizar una historia interactiva; ***MEXICA*** es el sistema que se utiliza para reconocer los sentimientos del usuario y así crear diferentes situaciones las cuales son características de una novela, es decir que este sistema es permisivo y cumple con el criterio de interacción entre usuario y máquina, dando por hecho una historia antes construida y permitiendo que las emociones reconocidas modifiquen las historias; lo interesante del artículo es la representación de cómo cada emoción afecta a los personajes, desde el odio de un caballero hacia un enemigo y el profundo amor que siente la princesa por su leal caballero, dándole puntos de agrado o quitándole estos, dependiendo el caso; este puede ser el criterio a utilizar a la hora de que una historia tenga giros argumentales dependiendo del rol que este siguiendo el usuario [11].

Por último, hay que saber cuáles son los criterios de interacción con una máquina, en este caso un sistema, precisamente este artículo menciona como surgió la idea de hablar con una máquina, pero lo que realmente es interesante y proporciona ayuda para el desarrollo de un Interactive Storyteller son los criterios que debe cumplir un sistema que interactúa con un ser humano, el más importante de estos es romper la barrera de instrucciones [12], es decir que el usuario no deba ingresar comandos para poder generar salidas del sistema, esto conlleva al siguiente atributo, el cual es, múltiples discursos dependiendo de las acciones del usuario, siendo así capaz de entablar una conversación donde la iniciativa de dialogo sea de parte y parte. Reconocer los problemas de lenguaje verbal y no verbal, es decir que el sistema sepa cómo responder a maneras de hablar y expresarse con gestos; esto conlleva a un aprendizaje y se logra evitar barreras a la hora de hablar, tanto como planear y proponer diferentes temas en la conversación. Para que este sistema sea perfecto, debe poder lograr tener recursos y servicios en línea y lograr mejorar su manera de interactuar [12].

CAPÍTULO III: Desarrollo del prototipo

**3.1. Requerimientos**

* + - * **Investigación con expertos en captar la atención de los niños**

Cuando se pensó en realizar un prototipo de narrador de historias para niños basado en sus posibles sentimientos y gestos; teniendo en cuenta los proyectos que se mencionaron en el capítulo anterior, era necesario contar con la opinión y experiencia de expertos en cómo tratar de captar la atención de un niño. El plan de acción consistió en buscar un colegio el cuál quisiera ser parte de un estudio con el único objetivo de identificar las diversas estrategias que un grupo de docentes y psicopedagogos utiliza para lograr el anterior objetivo mencionado.

Luego de varias solicitudes y presentaciones del proyecto a varias instituciones, el colegio que decidió participar en el estudio fue el **“Gimnasio Nuevo Modelia”,** otorgando el derecho de entrevistar a diferentes docentes de educación básica primaria; pues los usuarios que se piensa van a usar este prototipo son niños entre tres y once años; sin embargo es necesario conocer también las razones específicas del por qué aquellas técnicas que los docentes aplican en un sus clases funcionan, por ende, el colegio también permitió entrevistar a su psicopedagoga, experta en las diferentes formas de entender el lenguaje no verbal y como captar la atención de un niño con diferentes actividades que le permitan utilizar su imaginación.

En el capítulo I de este documento, se puede identificar que el foco de la justificación para realizar este prototipo es la estimulación de la imaginación infantil, las razones se mencionaron en aquel punto, no es necesario mencionarlas de nuevo, sin embargo, es necesario tenerlo claro y en cuenta para saber por qué las entrevistas se realizaron teniendo esta premisa.

La ronda de preguntas que se realizó en las diferentes entrevistas a docentes y psicopedagogos, se dividió en dos grandes grupos; “La imaginación: Sus causas y el cómo estimularla” y “La atención: Distracciones, Reacciones y medidas a tomar” (Para ver las entrevistas realizadas puede ver los vídeos adjuntos a este documento).

Entre los datos recolectados se identificó que muchos docentes utilizaban los juegos de rol como técnica para captar la atención de sus estudiantes más pequeños, acoplando algunas de sus clases y de temas anteriormente vistos en juegos donde los personajes resultaban siendo interpretados por sus estudiantes y dandoles una visión mas cercana al tema que estuviesen estudiando. Estas técnicas eran utilizadas por ellos en casos muy especiales, los cuales se determinaban con base en cómo se desarrollarán las clases anteriormente impartidas.

* **Reconocimiento de requerimientos no funcionales y no funcionales del prototipo**

Analizar la información recolectada en las entrevistas anteriormente mencionadas, dio la pauta para reconocer los requerimientos funcionales y no funcionales que debe tener el prototipo a construir. Con base en el conocimiento de los docentes y la psicopedagoga se logró identificar cuál es la forma en que el prototipo debe comportarse, pues las diferentes estrategias que ellos utilizan sirven para responder preguntas que se suelen hacer antes de comenzar con la fase de desarrollo del prototipo; las preguntas que se formularon son:

¿Cómo se debe contar una historia?

¿Cómo se debe captar la atención del niño?

¿Cuáles son los diferentes casos a los que hay que enfrentarse?

¿Cómo se puede lograr estimular la imaginación a través de un prototipo funcional que puede llegar a usar un niño de tres a once años?

Llegar a este nivel puntual de interrogantes, logró que el rumbo del proyecto tomase forma, teniendo presente que la información obtenida de proyectos desarrollados con un objetivo parecido es muy limitada; pues cómo se puede ver en el punto anterior, se demostraron diferentes recursos que los docentes y los psicopedagogos utilizan para estimular la imaginación de un niño.

Con esta información analizada se estipulo que el prototipo tiene los siguientes requerimientos funcionales y no funcionales:

**Requerimientos funcionales:**

|  |  |
| --- | --- |
| **RF#1: Iniciar relato** | |
| **Descripción** | El prototipo permitirá al usuario iniciar el relato, por ser el primer prototipo, solo podrá ejecutar una historia. |
| **Prioridad** | Alta |
| **Necesidad** | Alta |

|  |  |
| --- | --- |
| **RF#2: Pausar Relato** | |
| **Descripción** | El prototipo permitirá que el usuario pause la narración en cualquier momento |
| **Prioridad** | Media |
| **Necesidad** | Alta |

|  |  |
| --- | --- |
| **RF#3: Cambiar historia** | |
| **Descripción** | El prototipo permitirá que a través del análisis de sentimientos, la historia pueda cambiar dependiendo del sentimiento identificado a lo largo del relato. |
| **Prioridad** | Alta |
| **Necesidad** | Alta |

|  |  |
| --- | --- |
| **RF#4: Terminar** | |
| **Descripción** | El prototipo permitirá terminar el relato en cualquier momento de que se esté ejecutando si el usuario así lo desea |
| **Prioridad** | Alta |
| **Necesidad** | Alta |

**Requerimientos No funcionales:**

|  |  |
| --- | --- |
| **RNF#1: Conectar Web Service** | |
| **Descripción** | El prototipo deberá estar conectado a un Web Service (F.A.C.E. Sightcorp) que permite realizar la función del identificador de sentimientos |
| **Prioridad** | Alta |
| **Necesidad** | Alta |

|  |  |
| --- | --- |
| **RNF#2: Reproducir Audios** | |
| **Descripción** | El prototipo deberá reproducir los audios de la historia, dependiendo de las diferentes decisiones que pueda tomar este con base a los sentimientos identificados por el Web Service. |
| **Prioridad** | Alta |
| **Necesidad** | Alta |

|  |  |
| --- | --- |
| **RNF#3: Compatibilidad con MacOSX y Windows** | |
| **Descripción** | El prototipo se podrá ejecutar con las versiones de Windows 8 y 10, además de poderse ejecutar en MacOSX 10.8 o posteriores, sin embargo se debe tener en cuenta que esto dependerá de la versión de Unity3D que se tenga. Este prototipo se realizo en Unity3D 5.0.2. |
| **Prioridad** | Baja |
| **Necesidad** | Baja |

* **Investigación con los diferentes usuarios del prototipo**

Haber realizado la investigación con los docentes y psicopedagogos del colegio **“Gimnasio Nuevo Modelia”** determinó como debería funcionar el prototipo a alto nivel, sin embargo es importante tener en cuenta cuales son los gustos del directo implicado en el uso del prototipo, en este caso, los niños.

El objetivo de esta investigación fue reconocer cual era el tipo de historias que a niños entre los tres y once años les agrada escuchar, identificando los siguientes aspectos:

Atributos de los personajes principales

Atributos de los antagonistas de la historia

Escenarios posibles para el desarrollo de una historia

Argumentos por los cuales una aventura puede ocurrir

Qué giros argumentales en la historia podría sorprenderlos

Qué sentimientos afectan al usuario al momento de escuchar una historia con base a los giros argumentales identificados con anterioridad

Las moralejas de las historias que han escuchado

Género literario que más les llama la atención

Al ser una actividad con niños se necesitó de mucha creatividad para realizar la investigación, tomando alguna de las estrategias de docentes y psicopedagogos, anteriormente mencionadas en este capítulo. A todos los niños participantes en la investigación se les realizó una actividad lúdica en la cual se pudiesen sentirse en un juego, para que poco a poco la respuesta a los aspectos anteriormente mencionados se identificara de manera natural y sin que ningún elemento exterior los persuadiera en sus respuestas.

Para iniciar la investigación, se realizó una entrevista individual, con el objetivo de identificar la idea que tenían ellos de los aspectos anteriormente descritos.

Al momento de realizar cada una de las preguntas se buscaba que el niño no tuviese miedo, vergüenza o cualquier otro sentimiento que le impidiera responder cada una de las preguntas, así que se trató de realizar una entrevista donde ellos poco a poco fuesen respondiendo con más naturalidad.

Principalmente, lo que se quería era reconocer los aspectos importantes que podían influir en la total inmersión del cuento que se les pudiese llegar a contar, así que bastaba con preguntarles que aspectos les parecía llamativo en algún cuento que hayan leído o escuchado. El estudio se dividió en tres partes, las cuales fueron:

Cómo eran los cuentos que conocían

Cómo sería un cuento creado por ellos

Qué aspectos puede a llegar un cuento que ellos les gustaría escuchar

Los niños conocían cuentos clásicos cómo Blancanieves, Caperucita roja, la cenicienta, entre otros clásicos que se han hecho conocer a través del tiempo; sin embargo para sorpresa del estudio, mencionaban que aunque les gustaba escucharlos o leerlos de vez en cuando, se tornaban aburridos, pues estos ya no los impactaba. Muchos de estos niños encuestados tenían un hábito de lectura que apenas estaba despertando, pero aun así les da unas bases para saber cuáles serían los diferentes matices, giros argumentales y cualidades de los personajes que desarrollaran la historia.

Identificar lo anteriormente descrito, permitió que la entrevista fluyese de manera cada vez más natural, pues al momento de entrar a la segunda parte de la entrevista, donde se les preguntaba cómo sería un cuento que ellos crearían, limitándolos a una historia que se desarrollará en la época medieval; los niños poco a poco revelaron los diferentes personajes que participarían en esta, dando las características que estos debían tener, enfocándose en el héroe y en el villano. Posteriormente describieron los escenarios, dando a entender que podía desarrollarse en reinos gigantescos, en grutas tenebrosas o en campos donde hubiesen ruinas; la razón principal de la aventura de los héroes descritos era salvar algo o a alguien.

La tercera parte del estudio aterrizó las flamantes historias que crearon los niños, clasificándolas en géneros literarios, aspectos físicos y psicologicos de los personajes dependiendo de diferentes situaciones que puedan pasar en la historia y aspectos físicos de los escenarios.

Utilizando la técnica del juego de rol, mencionada en la investigación con los docentes, se les planteó una situación con la cual desarrollaran una historia.

**3.2. Análisis y diseño**

La fase de Análisis y diseño del prototipo se realizó paralelamente a la de requerimientos. Una vez identificados los requerimientos funcionales y no funcionales con los cuales debía cumplir el prototipo, se pasó a analizarlos; dando como resultado los requerimientos de usuario para así empezar a diagramar la solución de alto nivel del prototipo.

Antes de mencionar y mostrar los diagramas de alto nivel que se realizaron es importante mencionar los requerimientos de usuario que se identificaron.

* + - * + **Iniciar el prototipo:** El usuario, debe iniciar el prototipo para que este pueda interactuar con él y comience el relato.
        + **Cambiar el estado del prototipo:** La interacción del usuario con la aplicación se da por reconocimiento facial, así que basta con que el usuario realice los gestos que normalmente realiza cuando esta triste, enfurecido o feliz, para poder cambiar el estado del prototipo.
        + **Pausar:** El usuario podrá pausar el prototipo en cualquier momento de la ejecución.
        + **Terminar:** El usuario puede decidir interrumpir la historia en cualquier momento de la ejecución y dirigirse al menú principal.
* **Evaluación comportamental: Estudio infantil**

Cuando se estaba realizando la investigación con los niños, la cual se menciona en el punto 3.1 de este documento, se realizó una evaluación; esto consistió en la identificación de los posibles sentimientos que pueden mostrarse a lo largo de un relato pre-construido. Los sentimientos y gestos era la clave en este punto de la investigación, pues era la base para saber cómo construir la historia que se va a relatar en el prototipo e inclusive para saber cómo programar las decisiones que tomará el prototipo cuando este en ejecución.

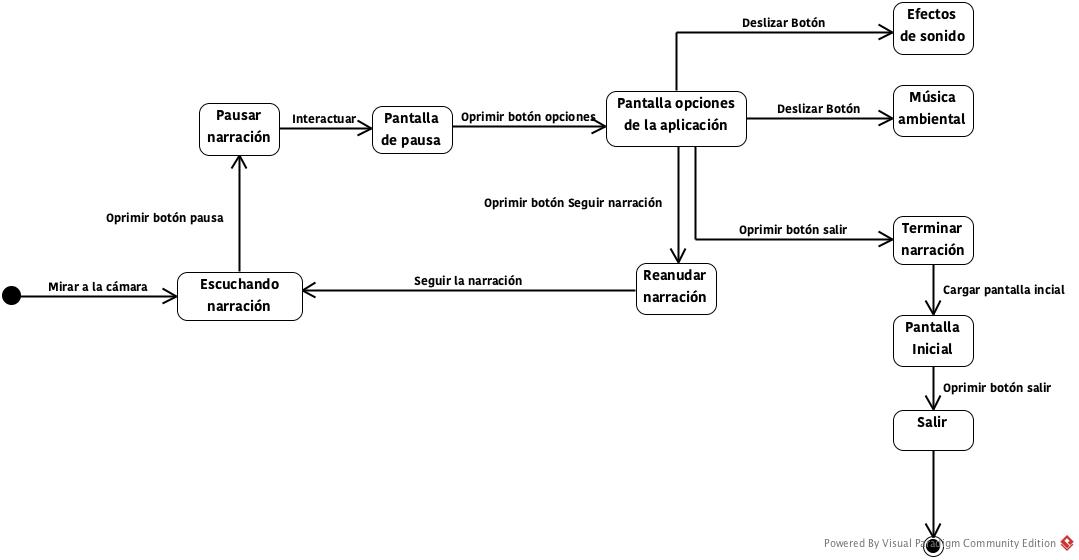
Se les pidió a los niños encuestados que escuchasen el inicio de una historia improvisada y estuviesen muy atentos a que sucedería, pues cada uno de ellos debía aportar un fragmento a la historia hasta que el narrador inicial le diese la pauta para finalizar aquel relato.

Cada uno tenía un turno para contar su parte de la historia con base a lo que el anterior había relatado, permitiendo observar cuales eran los estados de ánimo por los cuales un niño podía pasar, fue así como se eligieron los sentimientos a evaluar a la hora de poner en marcha el prototipo. Así como se menciona en los objetivos específicos, se eligieron tres de esos sentimientos evidenciados, los cuales fueron, Alegría, Tristeza y Enojo.

Teniendo en cuenta los requerimientos del usuario identificados, se realizaron los diagramas para mostrar la solución de alto nivel y se muestran en la siguiente pagina.

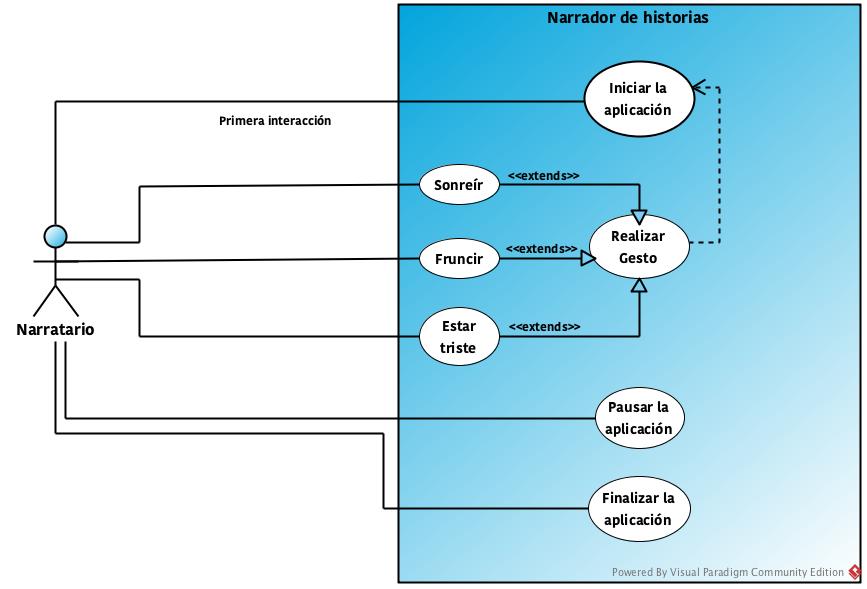
* + - * **Diagramas de Alto nivel del prototipo:**

**1. Diagrama de estado:**

****

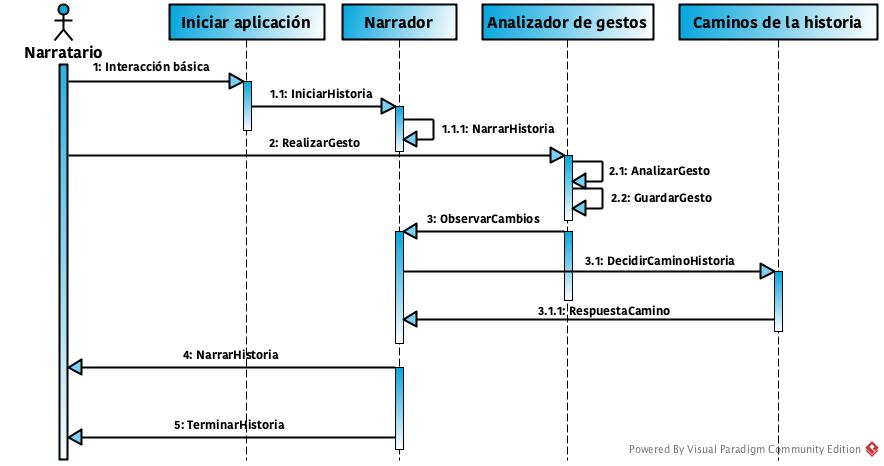
**Figura 4: Diagrama de estado**

**2. Diagrama de casos de uso:**

****

**Figura 4: Diagrama de casos de uso del prototipo**

**3. Diagrama de secuencia:**

****

**Figura 5: Diagrama de secuencia del prototipo**

Estos diagramas anteriormente presentados dieron una idea general de cómo el prototipo debía funcionar con base a los datos recolectados en la fase de Requerimientos, Análisis y diseño de la aplicación. Al ser la primera idea de diseño conceptualizada estaban sujetos a cambios.

**3.3. Desarrollo**

La fase de desarrollo del prototipo consiste en la codificación de este, el diseño de los personajes, el diseño de escenarios, la construcción de la historia a contar, el desarrollo de las pruebas del prototipo y la solución de posibles errores de este. Es importante que se tenga en cuenta que se va a ver en este punto del documento, pues así se tiene una idea general de cada uno de los puntos a tratar.

Esta fase de la construcción del prototipo conlleva varias tareas, así que se decidió realizarlas paralelamente, pues era absolutamente necesario alcanzar los objetivos propuestos en el desarrollo para poder tener tiempo de realizar las correcciones pertinentes que puedan arrojar las pruebas del prototipo.

* **Codificación, identificación y solución de errores**

Para el desarrollo del prototipo se eligió el motor de desarrollo para videojuegos y experiencias interactivas, Unity3D. Este motor tiene un entorno de desarrollo llamado **Monodevelop (IDE),** el cual soporta los lenguajes de **JavaScript** y **C#.** Por ser un lenguaje tipado y por preferencia propia, se eligió a **C#**, como el lenguaje de desarrollo del prototipo.

El objetivo del código fue solucionar cosas puntuales:

Conectar un web service capaz de brindar la información capaz de detectar los sentimientos de una persona, por medio de una foto.

Utilizar la WebCam de la laptop para poder tomar la foto que se podrá subir al servicio.

Activar los audios correspondientes a la historia con base a la conclusión que tome el prototipo y así crear los diferentes caminos de la historia.

**WebCam**

El primer objetivo que se realizó fue tomar las fotos con la WebCam del computador, pues era el de menos complejidad, pero el más necesario para poder realizar el llamado al web service.

Se creó una clase WebCam, heredando de la súper clase de monobehavior que trae por defecto Unity3D al crear cualquier script, por necesidad de funcionamiento es necesario heredar la mayoría de las clases que se creen posteriormente a esta súper clase, con el fin de poder utilizar cada una de las funcionalidades que trae esta, claramente habrá algunas excepciones, pero estas se verán mas adelante.

La clase WebCam que se creó, contiene todas las funciones que debe hacer la cámara integrada al computador cuando el prototipo comience su ejecución. Las funciones que debe hacer la cámara son:

Verificar que exista una cámara que el computador pueda utilizar.

Encender la cámara con la que cuenta el computador.

Tomar una foto cuando el audio que se esté reproduciendo este por terminar, de esta manera se podrá enviar al web service que se conectará.

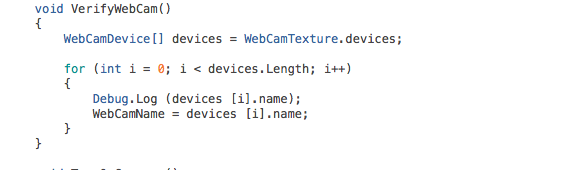
Antes de comenzar a explicar cómo se abordaron estas tareas, es importante saber que como criterio propio de codificación, se estipulo que se deben crear métodos que luego sean llamados en las funciones **Start()** y/o **Update(),** según se requiera,las cuales son funciones que Unity3D utiliza para saber en qué momento debe ejecutar las acciones a lo largo de la ejecución del prototipo. Para explicarse mejor, la función **Start()** se ejecuta al iniciar el prototipo, es decir, exactamente al momento en que este se empieza ejecutar. La función **Update()** por otro lado, ejecuta las acciones que se encuentren dentro de este en cada uno de los **FPS (Frames per second)** en los que se ejecute el juego. No es posible decir exactamente a cuántos FPS corre un juego, pues depende de la máquina en la que se ejecute.

Teniendo claro lo anteriormente descrito es más sencillo entender, por qué se decidió crear métodos para luego ejecutarlos en estas funciones; simplemente esto genera cierto orden en el código, permitiendo que cuando haya eventuales errores, sea más sencillo poder identificar qué está pasando y cómo se puede solucionar.

Para empezar con esta clase se decidió crear el primer método que permitiese verificar que existiese una cámara conectada al computador en el que se ejecute el prototipo:

**Void VerifyWebCam():**

Este es el método que verificará si existe una cámara que pueda tomar las fotos que necesita el prototipo, para esto se necesario realizar un arreglo de dispositivos, se utiliza la estructura que trae Unity3D llamada WebCamDevice, con la cual se crea un arreglo y se le asigna el valor que tenga la clase WebCamTexture con su método devices, de esta manera se accede a las Web Cam que se tengan conectadas al computador; teniendo esto, se puede acceder al arreglo de dispositivos, recorrerlo e imprimir por consola cual es el nombre de cada uno de estos dispositivos. El código se presenta en la figura 6 a continuación.



**Figura 6: ¿Existe una cámara conectada?**

**TurnOnCamera()**

Unity3D facilita encender la cámara utilizando la clase **WebCamTexture,** basta con instanciar una variable de este tipo y utilizar el método **Play(),** el cual trae esta clase y permite utilizarlo. En la figura 7 se puede apreciar el código donde se puede observar que se creó una variable para la Web Cam a la cual se le asigna lo que tiene la variable anteriormente instanciada, esto con el fin de poder utilizarla en el próximo método que se encargará de tomar los snapshots.



**Figura 7: Encender la cámara**

**TakeSnapshot()**

Este es el método más complejo de esta clase, sin embargo la complejidad estaba en saber qué resolución debía tener la foto tomada y cómo se iba a guardar la foto, así que estos fueron los temas que se debían solucionar:

Nombrar las fotos de la misma manera para que así el prototipo pueda acceder a ellas fácilmente a nivel de código.

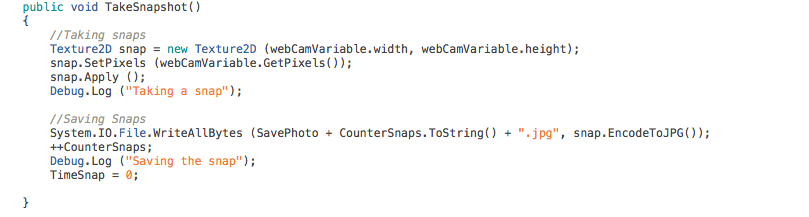
Utilizar un contador que fuese aumentando paulatinamente el prototipo fuese tomando las fotos; con el fin de asignárselo a cada una de estas y añadírselo al nombre de manera dinámica, con esto se logró que las fotos no se reemplazarán, por el contrario, simplemente se guardan como una secuencia y de esta manera fuese más sencillo utilizarlas posteriormente.

Crear una variable de tipo **WebCamTexture,** la cual tuviese la información de la cámara que se encendió en el método **TurnOnCamera()** y así poder utilizar la información de esa función sin necesidad de volver a hacer un llamado al método play().

Para crear una foto es necesario crear una textura 2D y Unity3D nuevamente facilita el proceso, basta con instanciar una variable de este tipo y asignarle los valores de **ancho (Width)** y **alto (height)** de la variable de tipo WebCamTexture que se creó anteriormente. El paso a seguir es obtener los píxeles que esta variable tiene guardados y que se obtienen por el método **GetPixels(),** que posteriormente son asignados a la variable de tipo **Texture2D** con el método **setPixels(),** pero aún el proceso no estará terminado si no se aplican estos cambios a la variable Texture2D, para esto se usa el método **Apply()** quien termina de asignar todos los atributos anteriormente obtenidos; es así cómo se puede tomar una foto con la cámara del computador, a través de Unity3D y utilizando C#.

Aún falta lo más importante de este método y es el guardar la foto para que luego el prototipo pueda acceder a ella y utilizarla para conocer los sentimientos del niño. Para esto se creó una variable de tipo string que pudiese guardar la ruta donde siempre se guardarán las fotos; acto seguido se utilizó la sentencia **System.IO.File.WriteAllBytes** para poder guardarlas, pasándole la variable que contenía la ruta, convirtiendo aquel contador que se describió anteriormente en un string y así asignárselo al nombre de la nueva foto. Por último, se utilizó el método **EncodeToJPG()** que permite convertir todos esos bytes a formato “.jpg”.

La solución anteriormente descrita se puede observar en la figura 8 de este documento.



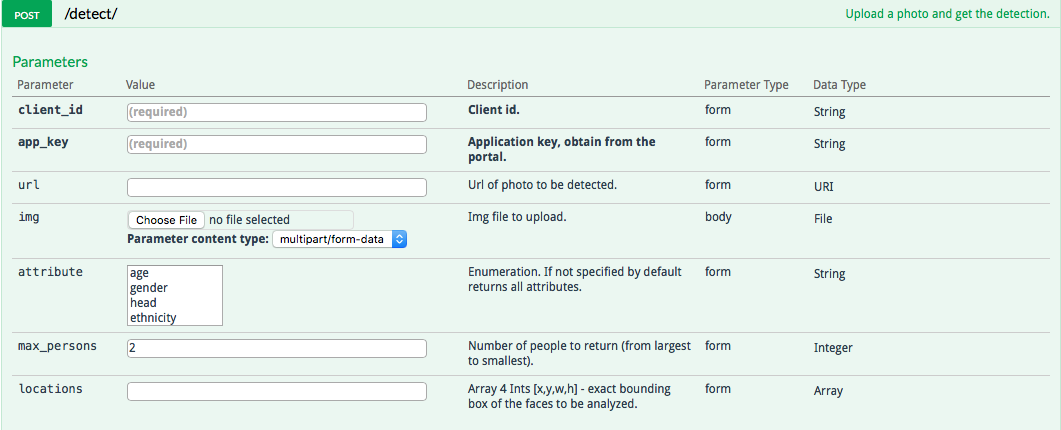
**Figura 8: Guardar y tomar las fotos necesarias para la evaluación de sentimientos**

**Conexión del Web Service:**

En el punto 3.1 de este documento, se mencionó la conexión de un Web Service; la decisión de conectar un Web service con el prototipo surgió por el hecho de que construir un analizador de sentimientos el cual realizará una función de tracking en tiempo real, era muy complicado, esto no era viable para el proyecto; por lo tanto, luego de realizar varias investigaciones, se encontró un analizador de sentimientos llamado **F.A.C.E by sightcorp**, el cual permite consumir un servicio que realiza un análisis de sentimientos con base a una foto que recibe el servicio, concluyendo con un JSon que brinda toda la información necesaria para identificar rasgos de la persona evaluada, su etnia, su edad aproximada y lo más necesario para el prototipo; el análisis de sentimientos.

Realmente este “Web service” es una API desarrollada para que diferentes desarrolladores puedan utilizarla de manera gratuita, con cierto límite al realizar los llamados a la URL que ellos exponen para poder consumir el servicio; es importante mencionar esto pues esto limitaba las funcionalidades del prototipo, pero era uno de los web service que se podía utilizar y del cual se tenían referencias de su aceptable funcionamiento.

Un Web service consta de un contrato, este servicio no es la excepción, así que es necesario observarlo y entenderlo pues es el modelo que ellos solicitan para poder ejecutar su servicio y dar la respuesta. F.A.C.E, claramente muestra este contrato para que cualquiera que quiera consumir su servicio, lo pueda hacer. El contrato de F.A.C.E. se puede encontrar en la siguiente url: <http://face.sightcorp.com/doc_swagger/>, sin embargo se muestra a continuación en la figura 9 como ejemplo de que se debe visualizar en aquella dirección.



**Figura 9: Contrato de F.A.C.E.**

Hay que tener en cuenta que tal como se muestran los datos en el contrato (figura 9) se han de utilizar al momento de codificar, pues de no hacerlo el servicio nunca recibirá los datos, por ende no habrá respuesta; esto se explicará con detalle más adelante.

Sightcorp, quien es la empresa desarrolladora del servicio, muestra diferentes ejemplos de cómo realizar la conexión y recibir la respuesta; el primer obstáculo que se encontró fue que F.A.C.E no mostraba ejemplos de cómo se debía construir la petición con lenguaje C#, su documentación muestra lenguajes como **PHP**, **HTML** y **Ajax**; pero da una idea de cuál es la estructura para llamar el servicio, consumirlo y recibir la respuesta. Entonces se tuvo claro las tareas que se necesitaban realizar para conectar el web service.

* + - * + Realizar una clase que permita la conexión con el web service.
        + Construir el modelo de respuesta que recibirá y mostrará el web service a través de C#

Los contra tiempos no se hicieron esperar, pues cuando se fue a realizar estas tareas se encontró que Unity3D, no puede interpretar scripts de tipo JSon, esta es la manera en la cual responde el servico, por lo tanto se debía encontrar la forma para poder traducirlo y realizar la conclusión de la evaluación que este mostraba.

Nació una nueva tarea que requería la construcción de esta respuesta y encontrar la manera de realizar una traducción que pudiese entender Unity3D, esecificamente el prototipo a través de C#. Luego de buscar en internet se encontró que muchos desarrolladores de Unity3D han pasado por el mismo problema a la hora de conectar servicios, así que algunos de ellos compartieron complementos que permitían realizar la traducción de las respuestas de los Web service que arrojaran un JSon; la mayoría están en el **Asset Store** de Unity3D, así que luego de una larga prueba con muchos de estos complementos que se ofrecían en la tienda de manera gratuita, se encontró uno que funcionaba perfecto.

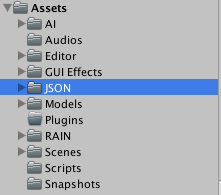
Realmente estos complementos no es más que un compendio de clases con ciertos métodos que se pueden usar para realizar las lecturas de scripts de tipo JSon y basta con agregarlo a la carpeta de Assets del proyecto, de esta manera se podrá usar en un nuevo o algún script que ya se tenga; por último, se instancian los scripts necesarios que se encuentran en la carpeta descargada para poder hacer uso de sus clases y los métodos que se necesiten.

Es importante separar siempre las funcionalidades, así que junto a la clase que conectaría el Web service, se tomó la decisión de construir otra clase que realizará la lectura del JSon que arrojaba el servicio, más adelante se continuará describiendo cómo se construyó la clase que permitía la conexión, pues es importante primero describir el lector de la respuesta del servicio.

Luego de pensar que debería hacer esa clase además de realizar la lectura del JSon, era claro que cuando se realiza una lectura de cualquier cosa se llega a una conclusión, así que esa era la siguiente función de esta clase, concluir dentro de toda la respuesta que arroja el servicio, cuál era el sentimiento del niño que utilice el prototipo.

**JSonReader**

Cómo se mencionó con anterioridad, para poder leer el JSon se utilizó un plugin que se descargó del Asset Store, cuando se descargan plugins, modelos o scripts de la tienda oficial de Unity3D, estas son agregadas como carpetas dentro de la carpeta principal de este, la cual es Assets, como se muestra en la figura 10 del cocumento.

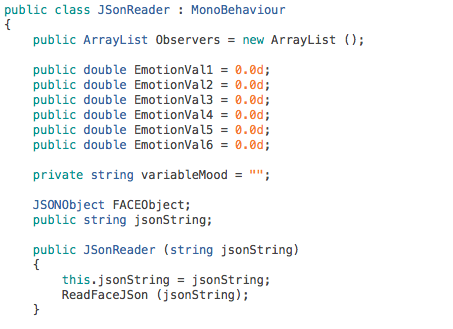


**Figura 10: Carpeta Assets del proyecto**

Teniendo la carpeta JSON en la carpeta principal, es posible acceder a los diferentes scripts que esta trae, los cuales permitirán instanciar objetos que faciliten la lectura de la respuesta del web service, con esto será posible traducir la información necesaria a C# para poder realizar la conclusión que será fundamental para elegir el camino de la historia.

Crear la clase JSonReader conlleva tener en cuenta ciertos criterios que poco a poco se irán explicando. En la figura 11, se puede observar que se han creado variables para:

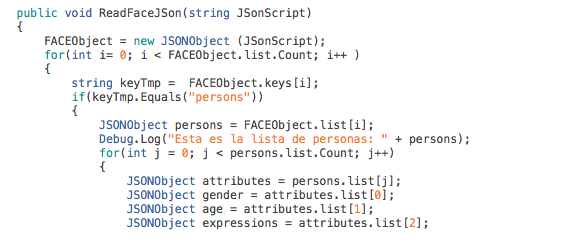
* Guardar los valores de las emociones. Esto se debe a que el web service le asigna un valor determinado a cada una de las emociones; esto viene perfecto para que el desarrollador pueda construir un método que realice una conclusión de los sentimientos obtenidos.
* Se necesita una variable que guarde el nombre del sentimiento concluido, esto es un string que se actualizará conforme se lea el JSon y se realice la conclusión.
* Se tiene una variable de tipo **JSONObject,** este tipo es uno de los cuales se puede utilizar gracias a la inclusión de la carpeta JSON a nuestra carpeta principal del proyecto, es acá donde se podrá obtener y guardar la respuesta del servicio.
* La última variable que se tiene es un string llamado **jsonString,** está se utiliza para guardar el JSon que responde el servicio y así poder empezar a utilizar lo que se necesite.

****

**Figura 11: Variables y constructor de la clase JSonReader**

Si se observa la figura 11, el constructor de la clase se ha definido, la razón es simple; era necesario estar seguros que cuando se obtuviese la respuesta del Web service se ejecutase el método ReadFaceJSon(), si esto no se hubiese estipulado, era posible que llegarán valores nulos a la clase, impidiendo que se hiciese la lectura del JSon y creando errores funcionales en el prototipo, también por esto es que el método mencionado se ejecuta tan pronto la clase recibe el JSon.

* + - * **ReadFaceJSon()**



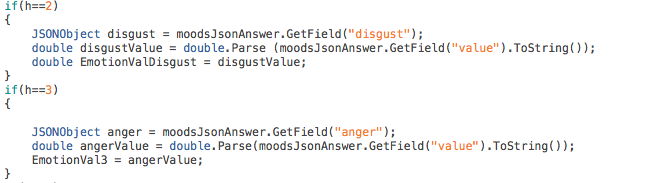
**Figura 12: El lector del JSon**

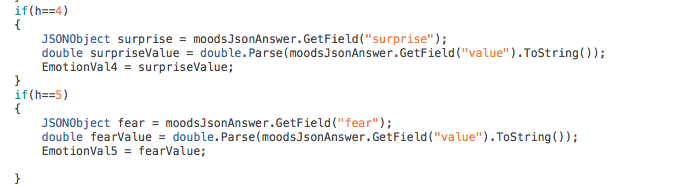
La figura 12, muestra la primera parte de la construcción del método que se encargará de leer los JSon que lleguen a la clase, recibiendolo en forma de string. FACEObject de tipo **JSONObject,** es la variable que anteriormente se creó, pero no ha sido inicializada, ni se había instanciado la clase de JSONObject, que será la que permitirá acceder y recorrer el script que ha de responder el servicio.

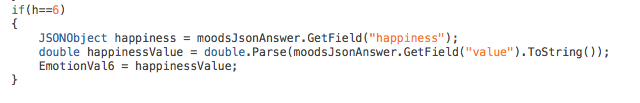
Los servicios web suelen tener llaves que permiten acceder a la información de cada una de las respuestas que estos arrojan, actuán como referencia para saber por que parte de la respuesta se han de mover; se dice mover, pues realmente esta información se ordena por listas, lo cual es un arreglo, por esta razón se han realizado los for necesarios para poder moverse entre las listas que el servicio responde; para este caso **“persons”** es la lista mas grande, es donde guarda la información de las personas que el servicio detecta y analiza, sin embargo la suscripción gratuita permite analizar a una sola persona, la cual es la que este mas cerca al lente de la web cam. Creando una variable de tipo **JSONObject** es posible moverse especificamente en la lista con el nombre que tiene la llave anteriormente creada, es acá donde se encuetrá la información de la respuesta, la cual esta organizada en listas y que algunas de ellas tienen otras listas dentro de si mismas.

Es importante tener en cuenta que se deben recolectar todos los datos de la respuesta, de no hacerlo se puede perder información, pues la respuesta no se va a visualizar tal como debe venir, puede que algunos valores se asignen en posiciones que no les corresponden, la causa de esto es porque se manejan for que recorren cada una de las listas.



****

****

****

**Figura 13: Obteniendo la respuesta del JSon, moviendose entre listas**

La figura 13 muestra un nuevo for, esto se debe a que mientras se realizaban pruebas, la respuesta del JSon en su lista de Expresiones, contenía otra lista que organizaba los sentimientos en forma de pequeñas cajas de información a las cuales solo era posible acceder si se tenía la posición de esa información, observando la respuesta fue como se obtuvieron los lugares del arreglo y por medio de if, se accedió a su información para guardar los valores de cada sentimiento, de esta manera se puede hacer la conclusión que había arrojado el servicio con respecto a la foto enviada.

Los valores de cada sentimiento son guardados en variables double, pues hay ocasiones donde el servicio arroja números muy grandes que deben ser manejados con este tipo de dato. Las variables globales para los valores de las emociones que se pudieron observar en la figura 11, se utilizan por primera vez acá, pues se les asigna el valor de cada una de las variables que obtienen del campo especifico de la respuesta y así se pueden utilizar en otros métodos, cómo en otras clases para realizar pruebas.

MacBook Pro:Users:macbook:Desktop:Screen Shot 2015-11-15 at 7.19.39 PM.png

**Figura 14: Los métodos de conclusión y la notificación de cambio de sentimiento**

**ConclusionFramework()** fue el método creado con el fin de realizar una conclusión a la respuesta del Web Service, esto es crucial para crear el factor de decisiones con el cual trabajará el prototipo, si esta conclusión no se realizará la traducción de la respuesta del Web Service no serviría para nada, pues solo se tendría un string que guarda cierta información muy larga. Era posible tomar ese string, pero si se tomaba el string que contenía toda esa información se tendría que recorrer todas las veces que se necesitara una respuesta para obtener el sentimiento de la persona, es por esto que existe el método ConclusionFramework().

A continuación se mostrará y se describirá cómo se creo el método que permite realizar la conclusión de la respuesta que ha dado el Web Service.



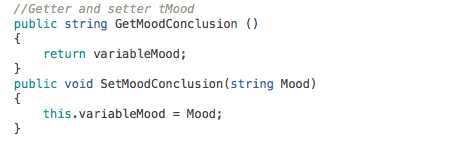
**Figura 15: ConclusionFramework(), el método que permite la conclusión**

El manejo de ArrayList fue vital para la elaboración del proyecto; JsonReader, fue la primera clase en la cual se decidió organizar la información en un arreglo de estos, por ser dinámico y por la fácilidad de agregar elementos a este. En la figura 14, se puede observar que al método de conclusión se le agregaron las variables que habían guardado los valores que arrojaba el servicio, además de eso se le pasan los nombres de los sentimientos para denotar a que valor corresponde cada sentimiento, cómo el servicio análiza seis sentimientos, era mas sencillo pasarle el nombre de estos sentimientos en el orden que ellos lo muestran. Los sentimientos que el servicio responde son:

* + - * + Tristeza.
        + Neutralidad.
        + Disgusto.
        + Enojado.
        + Sorprendido.
        + Aterrorizado.
        + Felicidad.

En la figura 15, se muestra lo que contiene el método de conclusión, donde se crearon las variables que la guardarían, los ArrayList que guardarían los valores de los sentimientos y los nombres de estos. Acto seguido, se agregan los elementos a cada uno de los ArrayList, por último se creo la forma de recorrer estos arreglos para determinar cuál es el valor mayor y a que sentimiento corresponde ese valor, para así guardar esa información en las variables de conclusión y permitiendo que el método retorne el nombre del sentimiento.

Dentro de este método se ve claramente que se llama un método llamado **SetMoodConclusion(),** esto se debe a que la variable **variableMood** es de tipo privado, con el fin de darle encapsulamiento a la clase, para que si otra clase debe acceder al valor de esta, solo pueda hacerlo por medio del metodo **GetMoodConclusion();** el cual se muestra en la figura 16.

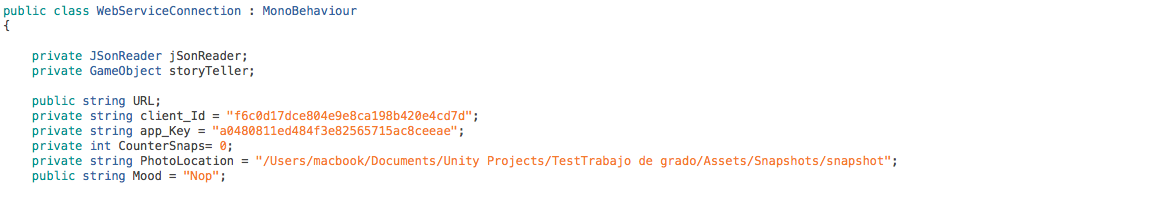


**Figura 16: El Get y el Set de la clase JsonReader**

Es así como se construyó el lector del JSon que responde el Web service, pero acto seguido, se construyó la clase que permitirá realizar la conexión al WebService de F.A.C.E., esta se explica a continuación.

**WebServiceConnection**

La construcción de la clase que se encargará de la conexión al Web service es un poco más sencilla que la anterior, pues hay clases propias de C# que permiten realizar el envío de datos de manera más sencilla.

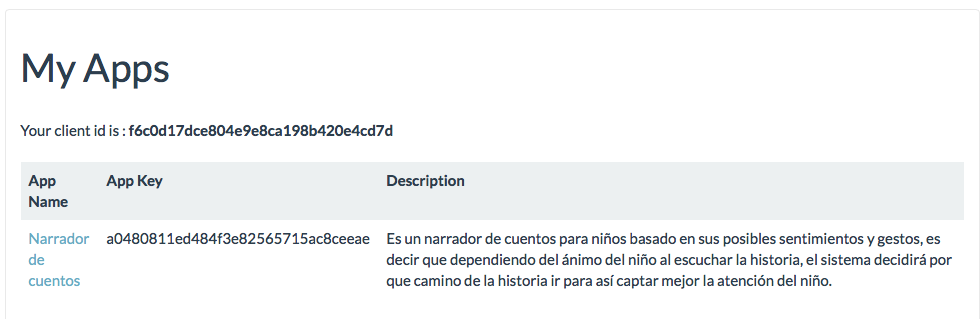


**Figura 18: Declaración de variables en la clase WebServiceConnection**

Es importante explicar las variables que se declararon de manera global en esta clase, pues algunas de estas son imprescindibles para la conexión del Web Service.

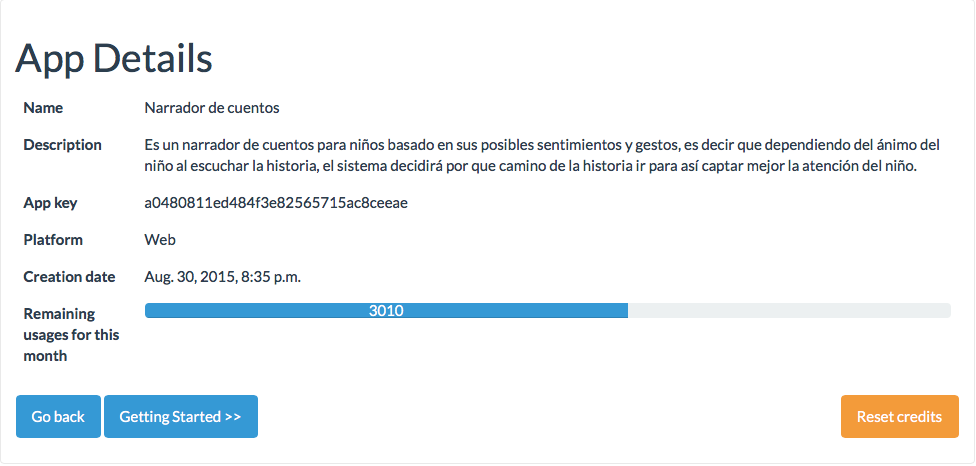
Las variables **URL, client\_id, app\_key** y **PhotoLocation,** son indispensables para poder realizar la conexión del servicio. La variable URL será quien guardará la dirección que expone F.A.C.E. para poder consumir el servicio, es pública para poder cambiarla desde el inspector de Unity, en caso de que en algún momento se tuviese que cambiar la URL, sea mucho más sencillo.

Por el contrario las variables de client\_id y app\_key son privadas y tienen un valor asignado, la razón es sencilla, cuando un desarrollador piensa utilizar el servicio de F.A.C.E. se debe registrar en la página de Sightcorp (http://face.sightcorp.com), para que desde la página se cree una referencia de la app que se va a crear y la cual va a consumir el servicio; una vez registrado y creada la aplicación dentro de su página, esta se direcciona a un Dashboard donde se puede ver las diferentes aplicaciones que se crearon para que estas puedan consumir el servicio.



**Figura 19: Dashboard de F.A.C.E by Sightcorp**

Cuando se observa el dashboard, es claro que la página asigna un código de cliente y una llave para la App, estos son los valores que guardan las variables de la clase WebServiceConnection, pues son datos requeridos para el consumo del servicio. Es importante mencionar nuevamente que F.A.C.E. permite realizar un limitado número de llamados para consumir el Web service; por mes se pueden hacer **cinco mil** request consumibles de manera totalmente gratuita y permitiendo hacerlo con una restricción de un llamado por segundo, para tener control de estos request, F.A.C.E permite ver cuantos llamados se han realizado, dando click en el nombre de la aplicación.



**Figura 20: F.A.C.E. Dashboard**

Como se ve en la figura 20, de esta manera se puede tener total control sobre el consumo del Web service, teniendo la oportunidad de reiniciar la cantidad de créditos o llamados una vez por mes.

Por último, la variable de PhotoLocation que se mencionó con anterioridad y se puede observar en la figura 18, es aquella que guardará la ruta donde debe acceder el prototipo al momento de enviar las fotos a analizar. Las demás variables se explicarán conforme se vaya describiendo toda la clase.

* **MakeFaceRequest()**

Este es el método de la clase que permite realizar el envío de la información necesaria para que el Web Service conteste.



**Figura 21: Método MakeFaceRequest**

Este método recibe la dirección donde se encuentran guardadas las fotos, pues es uno de los parámetros que debe enviar al servicio para que este responda. Para que todo este envío de información sea factible, en Unity3D se permite usar la clase **WWWForm,** la cual se encarga de mandar los datos a cualquier Web Service que reciba la información por POST, esto es perfecto, pues el servicio recibe los datos por este método.

**FaceForm** es la variable que permitirá el acceso a todos los métodos de la clase WWWForm, pero los únicos que interesan para este caso puntual son, **AddField** y **AddBinaryData,** pues son los métodos que permiten enviar la información por POST al Web Service. AddField permite enviar cadenas strings y AddBinaryData permite enviar archivos o imágenes, está información luego es interpretada por el servicio permitiendo completar los datos requeridos y estipulados en el contrato que se mostró en la figura 9.

Cuando se trabaja un Web service en Unity3D y específicamente en C# las clases que se utilizan y van de la mano son WWWForm y **WWW.** Ya se ha descrito que hace la clase WWWForm, pero no la clase WWW, básicamente la primera se encarga de preparar los datos para enviarlos y la segunda hace la conexión con el Web service, permitiendo pasar la URL del servicio y los datos que se van a enviar, es por esto que en la figura 21 se puede observar que se instancia una variable de tipo WWW para realizar la conexión.

Conectar un Web Service, el que sea conlleva tener en cuenta muchos sucesos que pueden pasar, entre estos es esperar a que la conexión esté lista y los datos hayan sido exitosamente enviados. Este es un manejo de tiempo que se debe controlar, al inicio de este capítulo se había mencionado que el manejo del tiempo en Unity depende de la máquina donde se ejecute el proyecto, pero para esto existen las “**Corutinas**”, que permiten ejecutar un método luego de pasado algún evento, en este caso puntual es esperar a que la conexión haya sido exitosa, para esto existe un método que pertenece a la súper clase de MonoBeahavior y se puede utilizar en cualquier momento. Esta clase hereda por defecto de la súper clase anteriormente mencionada, pero hay que tener en cuenta, además de todo lo ya mencionado, que este método recibe solamente otros métodos que retornen un **IEnumerator,** esto se debe a que esta es una cualidad en Unity que permite realizar tareas en cada uno de los frames en los que se ejecuta un juego, pero la “Corutina” permite que esas tareas se puedan pausar hasta obtener algún dato o acción del método IEnumrator para que cuando las tenga puedan ser ejecutadas en el siguiente Frame.



**Figura 22: WaitForRequest, obteniendo los datos**

En la figura 22 se puede apreciar la construcción del método WaitForRequest, el cual recibe el objeto WWW que contiene la dirección para conectarse y el WWWForm que es el formulario que se envió al Web Service para que se pudiese obtener la respuesta.

***Nota:*** *Los debug que se muestran en la imagen son para comprobar que los datos si están llegando y poder observarlos en la consola.*

Los **yield** que se pueden ver en la figura 22, son el atributo de los IEnumerator que permiten pausar las tareas hasta que se obtenga cierta respuesta, pera este caso puntual, se espera a que la conexión este lista y los datos hayan sido recibidos.

La condición que se encuentra en el if de esta función, permite corroborar que la conexión se haya podido establecer, el atributo **“.error”** es propio de toda variable de tipo WWW, este atributo siempre será nulo cuando se ha recibido una conexión de lo contrario este lanza una excepción, es por esto que si la conexión se efectúa, dos mensajes por consola son presentados, el primero hace referencia a cuanto le queda para terminar de realizar la conexión y el segundo que representa la conexión éxitosa. Gracias a lo anteriormente descrito se maneja la restricción principal del prototipo, **el dispositivo donde se ejecute siempre debe estar conectado a internet, de lo contrario este no podrá funcionar.**

Cuando el atributo .error es nulo, el método, además de hacer lo descrito anteriormente, obtiene la respuesta del servicio, esto se hace con el atributo **.text** que se asigna a la variable **jsonReader** de tipo **JSonReader,** la cual fue creada como variable global de esta clase y se puede observar en la figura 18 de este documento. Es importante recordar que la clase JSonReader se le definió un constructor, donde se obligaba a siempre pasarle una variable de tipo string, es por esto que en la instancia se le agrega la variable **URLConnection** junto con el atributo .text, pues este devuelve información de tipo string.

Cuando ya se tiene la respuesta del Web Service, se puede utilizar el método **GetMoodConclusion** de la clase JSonReader y asignárselo a la variable Mood propia de esta clase, sin embargo para asegurarse de que la conclusión se asigne a la variable Mood, se utilizó también el método **setMood,** que igualmente pertenece a la clase WebServiceConnection, todo estopara asegurarse de que se agregara correctamente el sentimiento a esta clase.

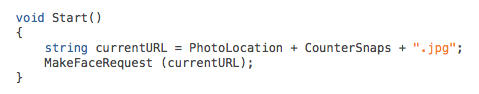
Seguido de la línea que utiliza el método setMood, se observa una manera poco común de modificar una variable que se encuentra en otra clase; resulta que en Unity3D, toda clase que herede de Monobeahavior no se debe instanciar con la palabra reservada **new**, pues al realizar esto es muy posible que algunos o todos los atributos de esa clase se vuelvan nulos, creando problemas que pueden volverse caoticos en un futuro, así que hay que tener muy claro cómo se hacen las modificaciones de atributos de una clase en otra.

En la figura 18, donde se puede apreciar las variables globales de esta clase se puede observar una variable pública de tipo **GameObject** estas variables son de gran ayuda a la hora de tener una referencia a otra clase ¿A que se debe esto? En Unity, todo script que se cree se debe agregar a un GameObject que se encuentre en la escena, de lo contrario nunca jamás se va a ejecutar. A Unity3D se le suele conocer como una herramienta **Drag and Drop**, la razón es simple y esas palabras lo explican a la perfección, la traducción literal de la frase es “**Agarrar y Arrojar**”, pero para llevarlo a un mejor argot se refiere a tomar y arrastrar, esta propiedad de Unity3D fácilita mucho las cosas, pues aquella variable pública se le puede asignar el GameObject que contiene el script **StorytTellerInvoker** seleccionandolo con el mouse y arrastrandolo desde la jerarquia hasta el inspector que debe estar mostrando el GameObject que contiene el script **WebServiceConnection.** Mas adelante, cuando se hable de la clase **AudioLibrary** se explicará con mayor detalle esta propiedad de Unity.

Ya se tiene claro por qué se debe evitar usar la palabra reservada new para instanciar clases que hereden de Monobeahavior y cómo se hace referencia a una clase cuando no se utiliza la palabra new, ahora es importante entender cómo se modifica un atributo de una clase en otra. Cuando ya se tiene la referencia al Gameobject que contiene la clase a la que se quiere accesar, se puede utilizar una propiedad de todos los Gameobjects de Unity, esta es **GetComponent<>()**, un método que puede obtener cualquiera de los componentes de un GameObject, en este caso serán los atributos de la clase que este tiene agregada, para este caso puntual el método de **setMood.**

Las siguientes lineas de código son semaforos que se utilizaron para activar y desactivar ciertas funcionalidades que se presentarán mas adelante cuando se comience a hablar del patrón de diseño Command que se implementó para la funcionalidad del prototipo.

Al final de la figura 22, se puede ver como la variable **counterSnaps** finalmente se utiliza dentro de esta clase, así como en la clase WebCam que se utiliza para asignarle un número a la foto y guardarla, la diferencia es que esta variable servirá para poder acceder a las fotos que ya se han guardado previamente y poder mandarlas al servicio. Ese contador se asigna a la variable **currentURL,** asignándole el valor de PhotoLocation y concatenándole el valor del contador mencionado. Se llama de nuevo el método MakeFaceRequest para que se vuelva a ejecutar y pueda volver a llenar el formulario con la información de una nueva foto,



**Figura 23: Iniciar el método MakeFaceRequest y la URL que debe leer por primera vez**

Se debe llamar el método que hace el request hacia el Web service y crear una nueva variable currentURL en el start para que no haya un error de referencia nula, pues de no indicarlo, no sabrá cuál es la primera foto que debe enviar al Web service.

Con esto se concluye la clase del Web service, esta clase fue fundamental para el funcionamiento del prototipo.

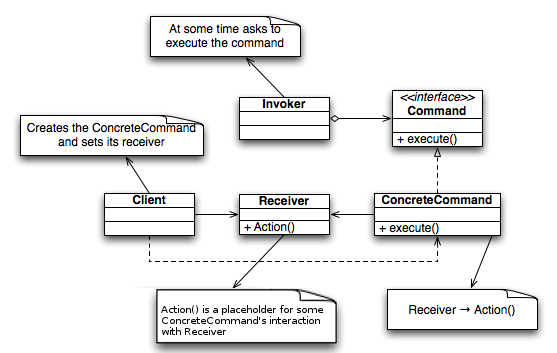
Luego de tener respuesta del web service y teniendo pleno conocimiento de que las clases estaban respondiendo tal como se esperaba, se empezó a pensar en buscar cual era la mejor forma de minimizar el impacto de desarrollo, así que se decidió implementar los patrones de diseño de tipo comportamental **Command y Observer** ¿Por qué se decidió implementar estos patrones? Bueno, el patrón comportamental Command es perfecto para invocar funciones de una clase dependiendo de una entrada específica, creando clases que permiten ejecutar cada uno de esos métodos.

El patrón observer se pensó porque era necesario notificar a la clase Web Service cuando se tomaba una foto para que este inmediatamente activará el servicio y respondiese el sentimiento del usuario, esto con el fin de poder utilizar esa conclusión para tomar los diferentes giros argumentales de la historia a contar.

Con esta breve introducción no queda más que entrar en materia y describir como se construyen estos patrones de diseño que servirán para el desarrollo del proyecto.

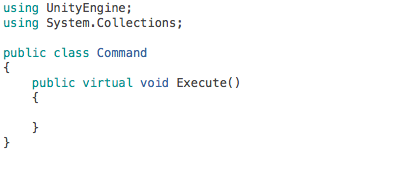
**Patrón de diseño Command**

Cómo se mencionó anteriormente el patrón Command sirve para poder invocar métodos de una clase específica dependiendo de una entrada especifica en una clase. Esta es la explicación a grosso modo de que hace un Command, pero conforme se vayan mostrando las clases creadas para el funcionamiento del prototipo se irá explicando de manera más técnica el patrón, haciendo referencia a cada una de las clases peresentadas en el diagrama modelo de la figura 24.



**Figura 24: Diagrama modelo de un patrón de diseño Command**

La figura 24 muestra el modelo estipulado para la construcción de un patrón de diseño Command, para ir entendiendo como funciona el patrón se hará una relación del diagrama mostrado, con las clases creadas en la construcción del prototipo.



**Figura 25: La clase Command**

La clase Command que se puede visualizar en la figura 25 es la clase interfaz, la cual permite estipular el modelo al que podrá acceder el receiver del prototipo para que luego de tenerla construida se puedan heredar todos los command concretos del prototipo. Un command concreto es una clase que hace referencia a un método específico de la clase que será el receiver del patrón. Para el prototipo, el receiver será la clase **StoryTeller,** quien se encargará de escoger la historia con respecto a los sentimientos que el Web service responda.

Para poder seguir con la estructura del Command que se implemento en el prototipo, es necesario mencionar la construcción de una clase, esta es la biblioteca de audios que contendrá todos los fragmentos que conformarán la gran historia que el prototipo contará.

* **AudioLibrary**

Se pensó en crear una clase que tuviese todos los audios que se fuesen a reproducir a lo largo de la ejecución del prototipo, donde la clase narrador o StoryTeller pudiese acceder a ellos y fuese el quien decidiese cuál audio reproducir, todo esto, dependiendo de que método pueda llegar a invocar el **Invoker** del patrón Command, es por esto que esta clase solo tiene métodos de acceso a los ArrayList contenedores de los audios que conforman la historia, además se crearon métodos para añadir cada uno de los audios a estos ArrayList y que se ejecutarán en el método **Awake(),** para estar seguros que se añadirán los audios tan pronto el prototipo inicie su ejecución. Este método se ejecuta antes de que incluso el Start() inicie, por eso se decidió que añadir los audios se debía hacer al inicio de la escena; con este método se tiene plena seguridad que los métodos para agregar los audios se ejecutará.

****

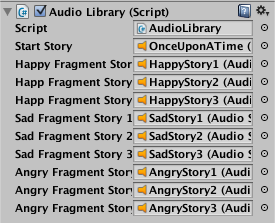
Figura 26: La biblioteca de audios, creación de variables

La razón de que estas variables de tipo **AudioSource,** el cual es el tipo que se utiliza para poder manejar los audios en Unity3D, sean públicas, es porque al momento de añadir los audios se debe hacer agregando cada uno de esos audios a un Gameobject vacío o 3D dentro de la escena, para que luego simplemente sean arrastrados desde el panel de la jerarquía de Unity (El cual se muestra en la figura 27 del documento) al inspector de este (Se puede ver en la figura 28), donde se debe seleccionar el GameObject que contenga el script de la biblioteca de audio, y donde aparecerán estas variables, preparadas para recibir los audios que se reproducirán en la escena de Unity. Es importante tener en cuenta que aquellos GameObjects que contengan los audios deben estar inactivos, de lo contrario sonarán todos al tiempo, se debe activar cada uno de ellos por medio de código.



**Figura 27: Jerarquía de Unity, GameObjects contenedores de los audios**

En la figura 27 se puede observar algunos de los GameObjects que están en la escena, en este preciso ejemplo se observan los GameObjects que tienen los audios, estos salen de un color gris, pues esto identifica que están inactivos, para activarlos se debe hacer por medio de código y cómo se mencionó anteriormente, quien se encargará de esto es la clase StoryTeller. Estos son los objetos que se deben arrastrar al GameObject que contenga el script de la clase AudioLibrary



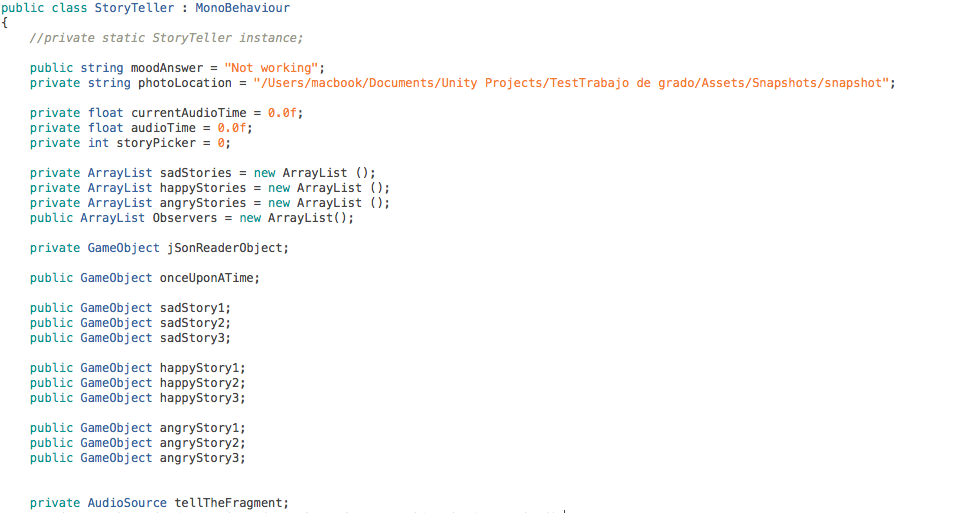
**Figura 28: Agregando los audios en el inspector a la clase de AudioLibrary**

A continuación se mostrarán en la Figura 29, los métodos que permiten agregar los audios a los ArrayList y los métodos que permitirán el acceso a estos en la clase StoryTeller, para poder cumplir con la función de tomar la decisión de cuál audio reproducir con respecto al sentimiento que el servicio haya respondido.



**Figura 29: Métodos de adición y Gets de la biblioteca de audios**

Teniendo claro que existe una clase AudioLibrary, es posible seguir hablando del receiver que se planeó para estructurar el patrón Command dentro del proyecto.

**Figura 30: Las variables de la clase StoryTeller**

Así como la clase AudioLibrary tiene públicas las variables que guardarán los audios, la clase StoryTeller también debe tener una referencia a los objetos que contienen los audios, es acá donde por medio del código cada uno de los GameObjects se va a activar en determinado momento, esto con el fin de que los audios no se reproduzcan uno encima del otro, tal como ya se había mencionado anteriormente, pero esto solo será para poder tener una referencia de que GameObjects se va a activar según el sentimiento que el Web service responda.

La clase StoryTeller debe tomar los audios de la biblioteca de audio, así que se crearon los ArrayList que tomará los arreglos que se encuentran en la clase AudioLibrary y los hace suyos, para que de esta manera esta clase decida que audio debe reproducir.

La decisión de que audio debe reproducir se planteó de la siguiente manera:

Inicio de la historia

Angry

Happiness

Sad

Angry

Happiness

Sad

Angry

Happiness

Sad

**Figura 31: Estructura de narración**

En la figura 31 se puede observar una estructura de narración, la cual fue planteada para crear los caminos de la historia, con la premisa que se pudo identificar en la fase de Análisis y diseño cuando se levantaron los requerimientos funcionales con los profesores y psicopedagogos, donde se identificó que al contarle una historia a un niño se debe manejar cada uno de los estados anímicos de ellos, es decir que se debe intentar que cada una de las emociones que siente un niño al escuchar una historia, está estrictamente ligado con los giros argumentales por los cuales el escritor decidió pasar, pero teniendo en cuenta que cada giro argumental le producirá un sentimiento al niño e inmediatamente cuando llegue al siguiente giro argumental debe lograr que ese sentimiento cambie.

Para el prototipo se determinó que el inicio de la historia siempre será el mismo, pues de esta manera se podrá evaluar cómo se sintió el niño al escucharlo. Cómo los sentimientos son totalmente aleatorios y depende de la respuesta del Web service. Ese es el plus del prototipo y lo que hace realmente interesante al proyecto.

En la figura 31 es fácil identificar que se muestran los sentimientos que serán evaluados, sin embargo aún no queda muy claro cuáles son los caminos que se plantea tomará el prototipo con respecto a la evaluación. Los ArrayList que se observaron en la figura 30 y de los cuáles se mencionó que guardarían los audios que se encontraban en la clase AudioLibrary están representados en la estructura de narración, si se observa detalladamente los sentimientos están organizados verticalmente, cada grupo de sentimientos representa un ArrayList; si se mira horizontalmente la estructura representa la posición de cada uno de esos ArrayList y por último las flechas, estás representan el camino que se va a tomar dependiendo de la respuesta del Web service. Sería ideal que se pudiese controlar los sentimientos que puede llegar a tener un niño a la hora de escuchar cierta parte de la historia, pero claramente esto es impredecible, cada quien puede sentirse, triste, feliz o enojado por un suceso que se le pueda llegar a contar, así que lo que hace el prototipo es recibir el sentimiento y ejecutar uno de los métodos de la clase StoryTeller. Si el niño se sintió triste al escuchar el fragmento anterior de la historia, el prototipo tomará la decisión de contarle el fragmento que se encuentre en la posición siguiente de cualquiera de los ArrayList, esto quiere decir que habrá una variable con la cual se debe comparar la posición de los ArrayList y con esta saber cuál es el camino, en la figura 30 se puede observar la variable **storyPicker** quien le dará la facultad al prototipo de realizar lo anteriormente descrito.

Ahora, la pregunta es ¿Cuál será el camino que el prototipo decidirá tomar con los otros sentimientos? Bueno, eso representan las flechas en la figura 31, si se le contó una historia feliz al niño, el prototipo le contará una historia que lo haga enojar, pero la idea no es hacer sentir enojado a un niño al escuchar x historia, el sentimiento enojado, en este contexto del prototipo, será evaluado cómo la oportunidad perfecta para contarle sucesos aún más emocionantes, con el único objetivo de que se interese aún más por la historia, la razón de que esto se pensará de esta manera es porque el Web service no tiene una respuesta para el sentimiento emoción, así que esta fue una manera de utilizar lo que este Web service ofrecía, acoplándolo a la lógica del prototipo.

Ahora, ¿Qué pasa si por el contrario el fragmento de la historia que se le contó al niño fue evaluado como enojo? Para cualquier narrador es importante despertar sentimientos con sus historias y esta debe ser la razón por las cuales debe hacer experimentar diferentes cosas al contar una, en el caso del prototipo, cuando el caso planteado anteriormente sucede, este decidirá contar un fragmento triste, el cuál pueda acoplarse con el camino de la historia anteriormente narrado, dándole matices que justifiquen el enojo del anterior fragmento, estos son los giros argumentales que se describían anteriormente, cada uno debe estar ligado y cada uno permite despertar interés por la historia que se está narrando.

Teniendo claro cómo se va a realizar la narración es momento de pasar a la parte funcional. La clase StoryTeller tendrá métodos que le permitan elegir la historia con respecto al sentimiento del Web service, pero es importante saber que el sentimiento no llegará a esta clase, simplemente porque no cumpliría con la estructura del patrón de diseño Command, así que el sentimiento llega es al **Invoker** del patrón.

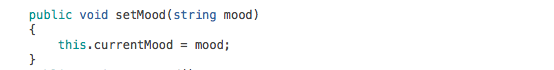
¿Por qué se explica de esta manera este patrón? Sucede que como están tan estrictamente ligados, la mejor forma de explicarlo es pasando por cada una de las clases del patrón mostrando cuál es el camino que ellas toman para poder ejecutarse correctamente.

En la figura 22 del documento, donde se describe la clase WebServiceConnection, aparece una instancia del invoker del patrón Command, de esta manera el patrón recibirá el sentimiento y podrá invocar los diferentes métodos de la clase StoryTeller dependiendo de la respuesta que le de la clase anteriormente mencionada.

El invoker tiene un patrón **Singleton** que buscaba que al momento de realizar la instancia anteriormente descrita fuese de la manera más sencilla posible y pudiese recibir el sentimiento que estaba llegando a esa clase.

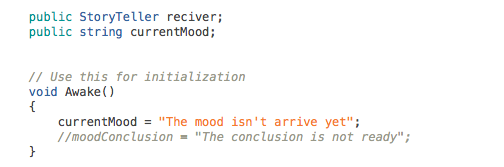
**MacBook Pro:Users:macbook:Desktop:Screen Shot 2015-10-31 at 6.11.17 AM.png****Figura 32: El singleton del Invoker**

El patrón singleton permite crear una instancia única de la clase en alguna otra clase que se necesite, así como se mostró en la figura 22. En el singleton de la clase se especifica cómo se instancia una clase que herede de la súper clase Monobehavior, para instanciar cierto elemento de una clase en otra, se debe tener un GameObject que pueda recibir la instancia y luego, por medio del método AddComponent, propio de cualquier GameObject se trae la instancia que se necesite de determinada clase. El patrón singleton permite que esto se realice solo una vez y que en lugar de tener que instanciar cierto elemento de esta clase, la instancia que se va a utilizar pueda llamar a cualquier elemento de esa clase en algún caso específico que lo necesite otra.



**Figura 33: El set que permite obtener el sentimiento respuesta del Web Service**

Una vez más los setter son una salvación en el manejo de estas clases, este es el método que se utiliza en la clase WebServiceConnection, de esta manera el invoker podrá guardar el sentimiento y saber que método se debe ejecutar apenas llegue.



**Figura 34: Las variables del Invoker**

CurrentMood es la variable que va a guardar el sentimiento y este se inicia en el método Awake de Unity3D, que se ejecuta antes del Start, esto con el fin de asegurarse de que siempre la variable se inicialice sin ningún sentimiento, pues se debe reproducir el inicio de la historia. En la figura 34 también se puede ver una variable de tipo StoryTeller llamada **receiver,** pues para que el patrón Command pueda funcionar se debe tener una referencia hacia aquel GameObject que actúe como el StoryTeller del prototipo.

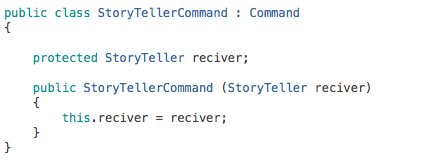
El invoker de un patrón Command, contiene métodos que pueden invocar los métodos del receiver y verificar la entrada que determinará cual es el método que se debe invocar, esto se puede ver en las figuras 35 y 36 presentadas a continuación. 

**Figura 35: Invokes de los métodos del receiver**



**Figura 36: Así se pueden invocar los métodos del receiver, rectificando cuál es la entrada, en este caso el sentimiento.**

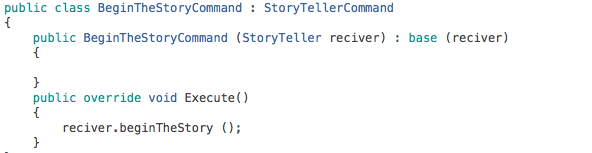
Sin embargo para poder convocar estos métodos que son propios del receiver se tiene que construir la clase que identificará a la clase StoryTeller como el receiver, esta heredará de la clase Command que se creó anteriormente y se mencionó su funcionalidad cuando se empezó a describir el patrón Command que se implementó.

****

**Figura 37: La verdadera clase Command del Prototipo**

Se dice que esta es la verdadera clase Command del prototipo simplemente porque cómo se mencionó anteriormente, de acá heredarán todos los comandos concretos quienes se encargarán realmente de llamar el método necesitado del receiver; esto sucederá cuando en el método CheckInput del Invoker cumpla con alguna de las condiciones estipuladas, para el caso de este desarrollo, el de cada uno de los diferentes sentimientos que se la van a pasar.

A continuación se presentarán los métodos propios del receiver, el cuál es el StoryTeller, junto a su comando concreto que le permite al invoker llamar cada uno de esos métodos.



**Figura 38: Comando Concreto, BeginTheStory**

****

**Figura 39: Método BeginTheStory, propio del StoryTeller, la clase receiver del patrón Command**

Este es el método que convocará la clase StoryTellerInvoker cuando el sentimiento que responde el Web service no ha llegado al invoker; por defecto el currentMood, variable de la que se habló en la página 58, tiene guardado **“The mood isn’t arrive yet”,** pues fue la manera en la que se podía ver por consola si el sentimiento del Web service, realmente estaba llegando a la clase Invoker, también dio la pauta para establecer la forma en la que se iba a reproducir el primer audio de la historia y en la cual se asegura que siempre comenzará por ese audio.

Es momento de activar los GameObjects que contienen los audios y fueron desactivados de la escena, para eso fue necesario crear variables de tipo GameObject que guardasen esos GameObjects que ya estaban creados en la escena cómo se mostró en la figura 27; al igual que como se agregaron los audios en la biblioteca, se agregaron al script StoryTeller, simplemente arrastrandolos al inspector de Unity que muestra este script cuando se selecciona el GameObject que lo contiene. Cuando se ha agregado cada uno de estos GameObjects que aparecen en la jerarquia al script se puede tener seguridad que ya hay una referencia creada y por medio de código se podrán activar.



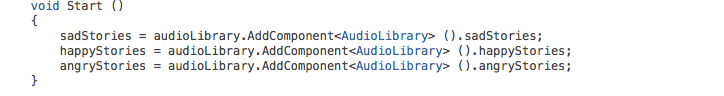
**Figura 40: La referencia de los GameObjects contenedores de los audios en la clase StoryTeller**

El resto del método mostrado en la figura 39, permite poner en reproducción el audio. Hay que tener en cuenta que cuando un audio es agregado a un GameObject en Unity3D, este, por defecto, viene preparado para que se cargue a la escena desde el método Awake, sin embargo esto se puede desactivar desde el inspector de Unity, a esto hay que sumarle que para el control de los audios en el proyecto e igualmente gracias a la interfaz gráfica que ofrece Unity3D se decidió activar la opción **mute** de todos los audios, para que por medio de código se pudiese desactivar y convocar el método propio de los AudioSource que permite reproducir el audio, este método es .Play(). Claramente se debe

Resulta que con base a la decisión de tener un mismo inicio para el prototipo, se pensó en evaluar el sentimiento del niño que fuese a utilizar la aplicación cuando el primer audio estuviese cerca de terminar, así que se investigó y se encontró que la clase AudioSource permite acceder al atributo del audio **time**, el cual es el tiempo en el que se encuentra actualmente, pero también esta clase permite acceder precisamente al clip de audio y con el atributo **length** devolver el tiempo total que dura un audio, así que venía perfecto para lo anteriormente ideado. El tiempo en Unity3D siempre se guarda en variable de tipo float, no es necesario un tipo que asigne más memoria en el computador, los atributos de los audios anteriormente descritos regresan un float, así que las variables que guardarán esto fueron creadas de manera global, pues los siguientes métodos que se van a describir también deben usarlos. La variable **tellTheFragment** propia de StoryTeller debe obtener aquel audio que se debe reproducir, así que la decisión fue traer el GameObject que contenía la clase AudioLibrary, al igual que los audios agregados a esta clase anteriormente, a la clase StoryTeller, ¿por qué? Pues se debe a que es necesario acceder a estas instancias varias veces, pues no solo se obtendrá el audio que iniciará la historia, también se obtendrán los ArrayList que contienen los demás fragmentos. Viendo la figura 39 es perceptible que se utilizó el método **AddComponent**, esta es la forma en la cual se instancian objetos que heredan de la clase MonoBeahavior y permite acceder a cada uno de los componentes públicos de otras clases.

Teniendo esto claro ahora se puede mencionar de donde se obtendrá el tiempo transcurrido y el tiempo total del audio para poder tomar la decisión de tomar la foto y notificar a la clase WebServiceConnection que haga el llamado. tellTheFragment ya puede acceder a los atributos de tiempo y tamaño del audio por ser de tipo AudioSource y como ya se le ha asignado el audio inicial, puede acceder al tiempo y al tamaño de ese preciso audio; la condición para tomar la foto es que si el tiempo actual es mayor o igual al tiempo, menos dos segundos del audio que se está reproduciendo se llame el método TakeSnapshot, propio de la clase que se creó llamada WebCam, así toma la foto y el servicio se puede llamar para que dé una respuesta que sea enviada al invoker, para que desde allí se elija cual método del receiver se va a invocar.

Los siguientes métodos se comportan de la misma manera, así que se mostrará su comando concreto y el método, explicando sus funcionalidades y sus pequeñas diferencias.

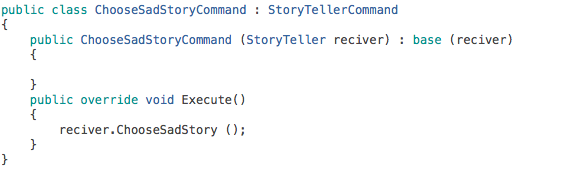


**Figura 41: Obteniendo los ArrayList de la biblioteca de audios**

Aquella variable que se creó para guardar el GameObject que actuaría cómo la biblioeca de audios, permitió instanciar los arreglos creados en la biblioteca para que el StoryTeller pueda recorrerlos y simplemente elegir el audio apropiado según el Invoker del prototipo. A continuación se mostrará como se recorrieron los arreglos para realizar lo anteriormente descrito.







**Figura 42: Método elector de historias tristes con su Command Concreto**

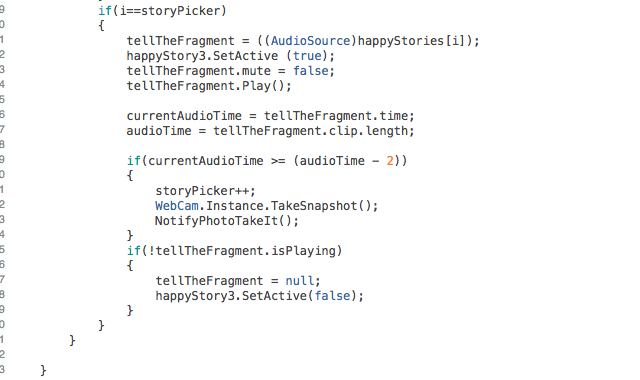
Recorrer los arreglos fue la forma que se decidió para poder obtener los audios indicados dependiendo de la invocación que realice la clase StoryTellerCommand. En la figura 37 se mostró el verdadero Command del prototipo, es esta clase quien permite identificar a la clase StoryTeller cómo el receiver del patrón, por lo tanto de acá heredarán cada uno de los comandos concretos, que son las puertas a un método especifico de la clase StoryTeller, cada uno con el método Execute, que le permite llamar al método requerido, simplificando en la clase Invoker, la cantidad de lineas de código, esto se puede evidenciar en la figura 35, donde se instancia cada uno de los comandos concretos en diferentes métodos para luego poder llamar su propio método Execute; con el fin de llamarlos en el CheckInput y que el invoker este pendiente de que a quien debe invocar.

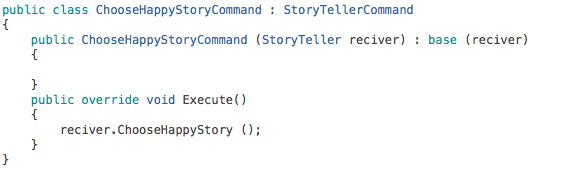
Teniendo claro esto se puede describir el recorrido de los arreglos a los que la clase StoryTeller ya tiene acceso y cuál será el proceso que el prototipo seguirá para poder reproducir el audio correcto. En la figura 42, en el método **ChooseSadStory(),** se puede observar que el recorrido del arreglo es simple, basta con un for que pueda recorrer todas las posiciones de este, se sabe que para el caso de este prototipo solo se tendrán tres posiciones por arreglo, sin embargo, si en un futuro se quiere hacer una historia mucho mas compleja o con mas caminos estos se pueden añadir a los diferentes ArrayList y los métodes funcionarán de la misma manera, pues el ciclo termina cuando llega a la última posición del arreglo, independientemente si no se sabe cual sea el tamaño de este. Es momento de usar la variable **storyPicker**, que realmente será un contador que determinará la posición a la cual debe acceder la clase StoryTeller, obtener su clip de audio y reproducirlo; la reproducción de este será de la misma manera que se planteó con el primer audio con el que se debe iniciar; la diferencia mas notoria es que para poder asignar ese audio del ArrayList a la variable tellTheFragment de la clase, es necesario realizar un cast de la posición, un cast es simplemente convertir aquella información al tipo de dato de la variable a la cual se le va a asignar, para el caso puntual de la elección de las historias será de tipo AudioSource. La variable storyPicker siempre, al inicio del prototipo iniciará en cero, pues debe cumplir con la estructura de narración descrita en la figura 31, pero para que esta cambié y no sea problema para cualquiera de los ArrayList a los que acceda la clase, una vez reproducido el audio y faltando dos segundos para que este acabe, la variable es iterada, por si debe acceder a otro Array y que continúe con la linea que ha de llevar la historia.

Por último, luego de que se toma la foto y se notifica a la clase que realiza la conexión al Web service, la variable tellTheFragment se pone en **Null,** para cuando acceda alguno de los otros métodos que eligen la historia no vaya a haber un problema de asignación de audios, además de esto el GameObject que contiene el audio reproducido se desactiva, evitando de igual manera que se pueda llegar a reproducir encima de algún otro audio que ya se pueda estar reproduciendo.

Los siguientes métodos de la clase receiver StoryTeller se mostrarán en la siguiente figura, teniendo en cuenta que todos se comportán de la misma manera que el método **ChooseSadStory(),** con la clara diferencia de que cada uno de estos métodos accede a un ArrayList diferente.







**Figura 43: Método elector de historias felices y su command concreto**

****

****

**Figura 44: Método elector de historias emocionantes y su command concreto**

Esta es la manera en que el prototipo funciona y en la cual se desarrolló, siempre siguiendo la metodología planteada y la cuál se itero varias veces para poder realizar los diferentes cambios que el prototipo debía sufrir.

* + **Interfaces de usuario**

Todo desarrollo de software de esta era necesita una interfaz gráfica de usuario y con mayor razón, cualquier aplicación que vaya a ser usada por niños. Este prototipo, quería tener una interfaz llamativa para cualquier niño que la usase, totalmente simple y que pudiese ser un acceso rápido y llamativo para el niño que llegue a utilizarla.

En el Asset Store de Unity se adquirió una interfaz de usuario que reflejara el sentido medieval de la historia, luego de realizar una búsqueda se seleccionó una llamada **Warforge Mobile UI,** pues aunque es perfecta para dispositivos móviles iba de maravilla para diseñar las interfaces para el prototipo ejecutable en computador. Durante la fase de análisis se recogieron datos de cómo un niño pensaba en que debería ser la interfaz con la cual ellos debían interactuar para poder iniciar el prototipo, claramente se recogió esta información de manera en que el niño entrevistado pudiese entender que era una interfaz. Mientras cada uno describía lo que se imaginaba se iba bosquejando sus ideas para que así al momento de diseñar la interfaz bajo las restricciones que se tuviesen de modelos, se pudiese traer a la vida tales interfaces.

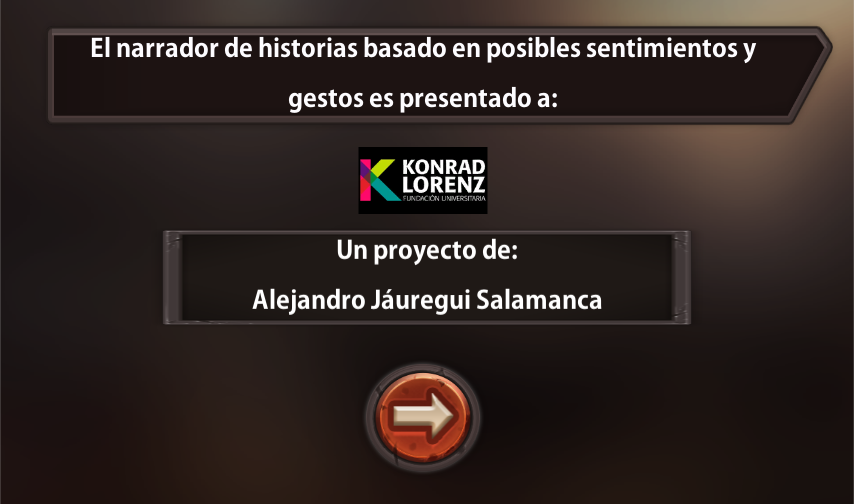
Por dificultades con el tiempo se tuvo que reducir la interfaz lo más posible, sin descuidar y dejar de lado las ideas que aportaron los niños cuando se les hizo la entrevista. La interfaz se redujo a una pantalla en la cual existiese un botón totalmente simple para comenzar, junto con varios íconos animados y una música de fondo que le permite al niño comenzar a imaginar cómo será la historia. Estos fueron los criterios elegidos para crear la primera escena del prototipo.

****

**Figura 45: Así nace el prototipo**

Es muy posible que se haga esta pregunta ¿Qué es Yorick…El narrador de historias”? ¿Por qué dice tal cosa en ese título? Bueno, cuando se hicieron las entrevistas, la mayoría de los niños coincidieron en que siempre es bueno conocer el nombre de quien les cuenta una historia, les hace sentir cierta relación con él, por esta razón se decidió nombrar al narrador Yorick, en honor a uno de los personajes más representativos de la historia, creado por William Shakespeare y presentado en su obra Hamlet, aunque con una aparición muy corta, Yorick la calavera que cuestiona el ser o no pareció ser la mejor elección. Aunque el prototipo no muestre al narrador, porque no se consiguió un modelo que se acoplara a esta exigencia y porque tampoco se cuenta con los conocimientos para la modelación y animación de uno, se decidió que Yorick fuese un narrador Omnipotente, es decir que aunque no se vea, el existe. Para que esto le quedase claro a cualquier niño que fuese a usar el prototipo, es el mismo Yorick quien presenta el inicio del juego, esto con el fin de que desde el inicio el niño se interese por escuchar la historia.

Cómo este prototipo se va a presentar a la Universidad Konrad Lorenz, se realizó una pantalla intermedia, entre la escena del menú principal y la escena donde comenzarán las historias, explicando que el proyecto presentado se llama **Narrador de historias para niños, basado en sus posibles sentimientos y gestos,** además de mencionar a la universidad y al desarrollador y diseñador del prototipo, Alejandro Jáuregui salamanca.



**Figura 46: La presentación oficial del proyecto**

Luego de que se pulsa el botón que aparece en la escena mostrada en la figura 46 de este documento, el prototipo se dirige a la escena donde todos los scripts que permitirán narrar la historia están agregados.

**Capitulo IV: Conclusiones y una vista al futuro**

**4.1. Conclusiones**

Luego de realizar todo el desarrollo del prototipo es importante decir que se logró desarrollar cada uno de los objetivos propuestos, además que se adquirió conocimiento fundamental para el desarrollo con el cual no se contaba al inicio del proyecto. Tal vez como desarrollador la imaginación se desarrolló y potencio al máximo, pues estructurar el proyecto no fue una tarea sencilla, sin embargo esto no significaba que fuese imposible. De gran ayuda fue cada una de las materias que se imparten a lo largo de la carrera de ingeniería de sistemas, desde lo más básico que es pensar cómo se planteará la solución, hasta lo más difícil que es la manera en que se puede optimizar el desarrollo, junto con el manejo de los recursos de la maquina donde se piensa ejecutar; cada uno de estos temas y los que se encuentran conectados entre estos, fueron vitales.

Conocer y aprender acerca de cómo se debe captar la atención de los niños fue fundamental para este desarrollo, pues aunque la parte técnica es importante, para el objetivo principal el cuál es estimular la imaginación de un niño que pueda a llegar a estar en contacto con este prototipo, por esto conocer las opiniones de ellos fue vital, de esta manera fue como se desarrolló la historia y se buscó hacerla lo más llamativa posible para el rango de edad estipulado del proyecto.

**4.2 Desarrollos futuros posibles**

Cuándo se piense en desarrollos futuros simplemente hay que explotar la imaginación, el cielo es el límite, pero siguiendo la misma línea del desarrollo acá presentado, estos son los desarrollos futuros que se podrían hacer, independientemente de quien quiera retomar el proyecto, tomando este como base.

* + - * Implementación de más historias de diferentes géneros literarios, para que el niño tenga varias posibilidades a descubrir.
      * Modelo de logros, es decir, medallas u objetivos, desbloqueando diferentes caminos de historias, todo para que el niño ame la aplicación y no se cansé de usarla.
      * Implementación de Inteligencia artificial a las escenas para que recreen lo que se está narrando.
      * Interacción del jugador con la historia por medio de movimientos en la escena.

Estas son las posibles funcionalidades que podría llegar a tener el proyecto en cualquier momento si alguien desea continuar con él, claramente si surgen otras ideas son más que aceptadas, pues al momento de imaginar, no hay nada que no se pueda realizar.

Terminando con este documento, solo queda por mencionar a las referencias que fueron fundamentales para crear la visión de este prototipo, muchas de las referencias que aparecen en el punto 4.3. De este documento hablan sobre proyectos que buscaron darle un giro a las narrativas actuales, por lo tanto no sobra revisarlas si alguna de la información presentada en este documento le ha parecido interesante y quiere profundizar en el tema.

**4.3 Referencias**

[1]. Eduapps.es, 'Aplicaciones educativas'. [Online]. Available: http://www.eduapps.es. [Accessed: 19- Aug- 2014].

[2] A. Sátiro, ‘Volar con las alas de la imaginación infantil. Página 4. Revista número 7.

’ 20-Aug-2014.

[3] D. M, ‘Early stimulation of rodents: a critical review of present interpretations’, British Psychological Society, Aug. 1973.

[4]*Historia del alfabeto y medios de escritura*, 1st ed. México, 2013.

[5]*Literatura de la edad media*, 1st ed.

[6]G. Polis, *Historia y evolución del teatro universal*, 1st ed. 2015.

[7] Callas-newmedia.eu, 'Welcome to CALLAS!', 2015. [Online]. Available: http://www.callas-newmedia.eu. [Accessed: 25- Feb- 2015].

[8]E. McNair, *Using interactive stroytelling agents to Broaden Participations in computing, Morehouse*, 1st ed. 2013, p. ScienceDirect.

[9]M. Theune, L. Jeroen and A. Thijs, *Acting, Playing or Talking about the Story: An Annotation Scheme for Communication during Interactive Digital Storytelling*, 1st ed. Twente: University of Twente, 2013.

[10]A. Wolff, P. Mulholland, Z. Zdenek and R. Joiner, *Re-using digital narrative content in interactive games, Department of Pshycology*, 1st ed. bath: University of bath, 2006.

[11]R. Perez and G. Oden, *Employing emotions to drive plot generation in a computer-based storyteller*, 1st ed. Cuajimalpa: Universidad autonoma Metropolitana Unidad Cuajimalpa, 2006.

[12]N. Mavridis and A. Paraskevi, *A review of verbal and non-verbal human–robot interactive communication*, 1st ed. Athens, 2015.