**DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN**

**OPCION I.- TESIS**

TRABAJO PROFESIONAL

# "DESARROLLO DE UN INTÉRPRETE DE LENGUA DE SEÑAS MEXICANA CON VOCABULARIO CONFIGURABLE SEGÚN EL CONTEXTO"

QUE PARA OBTENER EL GRADO DE:

**MAESTRO EN**

**SISTEMAS COMPUTACIONALES**

PRESENTA:

**ROBERTO HERNÁNDEZ DE LA LUZ**

DIRECTOR DE TESIS:

**M.C. MARÍA ANTONIETA ABUD FIGUEROA**

CODIRECTOR DE TESIS:

**DRA. LISBETH RODRIGUEZ MAZAHUA**

ORIZABA, VER. MÉXICO

ABRIL 2019.

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Orizaba

"2019, Año del Caudillo del Sur, Emiliano Zapata"

FECHA: 30/05/2019
DEPENDENCIA: POSGRADO
ASUNTO: Autorización de Impresión
OPCIÓN: I

C. **ROBERTO HERNANDEZ DE LA LUZ**
CANDIDATO A GRADO DE MAESTRO EN:
**SISTEMAS COMPUTACIONALES**

De acuerdo con el Reglamento de Titulación vigente de los Centros de Enseñanza Técnica Superior, dependiente de la Dirección General de Institutos Tecnológicos de la Secretaría de Educación Pública y habiendo cumplido con todas las indicaciones que la Comisión Revisora le hizo respecto a su Trabajo Profesional titulado:

**"DESARROLLO DE UN INTERPRETE DE LENGUA DE SEÑAS MEXICANA CON VOCABULARIO CONFIGURABLE SEGUN EL CONTEXTO".**

Comunico a Usted que este Departamento concede su autorización para que proceda a la impresión del mismo.

A T E N T A M E N T E

**DR. RUBEN POSADA GOMEZ**
**JEFE DE LA DIV. DE ESTUDIOS DE POSGRADO**

C.A. TITULACIÓN
SECRETARIA DE EDUCACIÓN PÚBLICA
INSTITUTO TECNOLÓGICO DE ORIZABA

FECHA : 06/03/2019

ASUNTO: **Revisión de Trabajo Escrito**

**C. DR. RUBEN POSADA GOMEZ**
JEFE DE LA DIVISION DE ESTUDIOS
DE POSGRADO E INVESTIGACION.
P R E S E N T E

Los que suscriben, miembros del jurado, han realizado la revisión de la Tesis del (la) C. :

**ROBERTO HERNANDEZ DE LA LUZ**

la cual lleva el título de:

**"DESARROLLO DE UN INTERPRETE DE LENGUA DE SEÑAS MEXICANA CON VOCABULARIO CONFIGURABLE SEGUN EL CONTEXTO".**

Y concluyen que se acepta.

A T E N T A M E N T E

PRESIDENTE : **M.C. MARIA ANTONIETA ABUD FIGUEROA** — FIRMA

SECRETARIO : **DRA. LISBETH RODRIGUEZ MAZAHUA** — FIRMA

VOCAL : **DR. ULISES JUAREZ MARTINEZ** — FIRMA

VOCAL SUP. : **M.C. CELIA ROMERO TORRES** — FIRMA

EGRESADO(A) DE LA MAESTRIA EN **SISTEMAS COMPUTACIONALES**

OPCION: **I** **Tesis**

Esta tesis está dedicada a mi familia,
a todos ustedes que de una u otra forma
me formaron e impulsaron.

**Agradecimientos:**

*A mi directora de tesis, por creer en mí y darme la oportunidad de explorar y desarrollar esta idea.*

*Al Doctor Mario Rojas Meza y mis compañeros del curso de LSM por apoyarme en el aprendizaje y desarrollo practico de este proyecto.*

*A mi tío Guillermo por ese primer impulso para aprender computación.*

*A mi hermano Abraham por permitirme ver sus éxitos y recordarme lo mucho que puedo mejorar profesional y personalmente.*

*A mi padre Roberto por ser un constante ejemplo de perseverancia y por recordarme que uno debe buscar sonreír siempre.*

*A mi hijo Roberto por regalarme su risa y ser un oasis dentro de los mil pendientes que tenía.*

*Pero sobre todo a las mujeres más hermosas de mi vida, a mi madre Adela y a mi esposa Margaret, por tolerarme, por aguantar mis desvelos y malos humores, por estar ahí, cada una, a su manera, serán siempre la constante en mi vida para seguir creciendo, para: "saltar la raya que me pinten".*

# Índice

## Índice de Tablas

## Índice de Figuras

## Resumen

Actualmente el 35% de la población mexicana sufre algún tipo de discapacidad auditiva, además de que a pesar de que la lengua de señas mexicana se considera una lengua oficial, existe un déficit de intérpretes y las personas con esta discapacidad ven su calidad de vida mermada, debido a la falta de acceso a servicios como el resto de la población, además de que ven limitada su comunicación con todas aquellas personas que no dominan la lengua de señas.

Teniendo en cuenta esta problemática, el presente proyecto busca diseñar y construir un intérprete de lengua de señas mexicana con vocabulario configurable según el contexto, apoyado en el dispositivo de captura de movimiento 3D LeapMotion y técnicas de procesamiento de imágenes, brindando una mejora sustancial en la calidad de vida de las personas con discapacidad auditiva.

**Abstract**

Currently 35% of the Mexican population suffers from some type of hearing disability and although the Mexican sign language is considered an official language, there is a deficit of interpreters, so people with this disability see their quality of life diminished, because they cannot access to services like the rest of the population, in adition to seeing their communication limited with all those people who do not know the sign language.

Taking into account this problem, the present project seeks to create a Mexican sign language interpreter with configurable vocabulary according to the context, supported by the 3D motion capture device LeapMotion and image processing techniques, providing a substantial improvement in the quality of life of people with hearing disabilities.

## Introducción

La comunicación por medio de un lenguaje común es una característica inherente de la vida diaria del ser humano, sin embargo, existen personas que sufren de algún tipo de discapacidad que les impide comunicarse, lo cual repercute en su calidad de vida.

Por otra parte, en la actualidad existen avances importantes en el desarrollo de dispositivos de captura de movimiento en 3D, además de una constante mejora en la capacidad de los equipos de cómputo, lo cual permite obtener el máximo provecho de diversas bibliotecas de captura y procesamiento de imágenes.

Por lo cual, en la presente tesis se propone la creación de un intérprete de lengua de señas mexicana por medio de la implementación del dispositivo de captura de movimiento en 3D LeapMotion, en combinación además de la biblioteca de procesamiento de imágenes OpenCV para mejorar el porcentaje de aciertos en la identificación de señas.

La presente tesis se compone de cinco capítulos, donde el primer capítulo se enfoca en dar a conocer los conceptos básicos del dominio del problema, así como el planteamiento del mismo, objetivo general y específicos y la justificación del proyecto, el capítulo dos se compone de una revisión de las investigaciones y publicaciones que se realizaron anteriormente, enfocadas en el uso de dispositivos de captura de movimiento en intérpretes de lenguas de señas, además de contener la propuesta de solución planteada, en el capítulo tres se describe el desarrollo del proyecto, indicando la aplicación de la metodología seleccionada, el capítulo cuatro contiene los resultados obtenidos durante el desarrollo de la tesis, en el capítulo cinco se indican las conclusiones y recomendaciones a las que se ha llegado y finalmente se incluyen las secciones de anexos y referencias.

# Capítulo 1. Antecedentes

En este primer capítulo se explican los conceptos más relevantes relacionados con el trabajo presentado. Se da lugar a la problemática a resolver, los objetivos a alcanzar y la justificación de lo que se pretende realizar.

## 1.1. Marco teórico

En esta sección se presentan los conceptos relacionados con el tema de investigación.

### 1.1.1. Lenguaje

Una definición aceptada por el diccionario de la lengua española [1] para lenguaje es: “Conjunto de señales que dan a entender algo”, de igual forma como señala Pelayo et al. [2] una lengua existe como código en la medida en que sus signos concurren en situaciones comunicativas. Así pues el lenguaje humano se vale de operaciones de sustitución simbólica para cumplir el objetivo de comunicar estados de realidad interna y externa.

Dauder [3] indica que existen dos tipos usuales de lenguaje convencional: el hablado y el escrito. El primero es más expresivo que el segundo porque cuenta con más modulaciones de la voz, que son sumamente difíciles de expresar (y de entender bien) con signos escritos. En este sentido son más ventajosos y pertinentes los medios de comunicación que transmiten la voz, como los telefónicos, que los que solo transmiten mensajes escritos. Hasta hace unas décadas el lenguaje oral contaba con una desventaja: la de ser más pasajero. El escrito era más inexpresivo, pero más perdurable. La tecnología hoy no sólo permite guardar el material escrito de modo mucho más fácil y perdurable que en un libro, sino también la posibilidad de guardar archivos de lenguaje oral por medio de sofisticadas y fidedignas grabaciones.

Sin embargo, superior a los medios de comunicación que transmiten la voz son aquellos que la acompañan de imágenes en las que se aprecian los gestos que se realizan con el rostro, las manos, entre otros, porque todos esos signos están en función de la inteligencia humana.

### 1.1.2. Lenguaje de señas

Los sordos se comunican por señas y el lenguaje de señas tiene la característica de haber sido creado por ellos mismos para satisfacer sus propias necesidades de comunicación. No hay una lengua de señas universal, sino que en cada comunidad estas lenguas mudas tienen su propia historia [4].

### 1.1.3. Lengua de Señas Mexicana

La Lengua de Señas Mexicana (LSM) es la lengua que utilizan las personas sordas en México. Como toda lengua, posee su propia sintaxis, gramática y léxico.

La Lengua de Señas Mexicana se compone de signos visuales con estructura lingüística propia, con la cual se identifican y expresan las personas sordas en México como se observa en la figura 1.1. Para la gran mayoría de quienes nacieron sordos o quedaron sordos desde la infancia o la juventud, ésta es la lengua en que articulan sus pensamientos y sus emociones, la que les permite satisfacer sus necesidades comunicativas, así como desarrollar sus capacidades cognitivas al máximo mientras interactúan con el mundo que les rodea [5].

Figura 1.1 Lengua de Señas Mexicana

### 1.1.4. Dactilología

Se conoce como dactilología en el lenguaje de señas a lo que en lengua oral se conoce como deletreo y está representada sobre todo con el abecedario. Cada palabra se representa con la articulación de mano correspondiente de cada letra que la conforma. De acuerdo con esto, *mamá* se representa con cada una de las señas que forman la palabra completa [6].

### 1.1.5. Ideograma

Los ideogramas representan una palabra con una o varias configuraciones de mano, incluidas repeticiones y movimientos, de acuerdo con esta realización, la palabra *mamá* se articula con la letra *m* de ideograma sobre los labios, con la que se golpean varias veces los labios [6].

### 1.1.6. Configuración Manual

Son estrategias que se usan en la Lengua de Señas Mexicana para describir las características inherentes y sobresalientes de un objeto como su material, forma, consistencia, tamaño, ubicación, orden y número [7].

### 1.1.7. SignWriting

"*SignWriting*" es un sistema de escritura visual que hace posible leer, escribir y teclear cualquier lenguaje de señas en el mundo. "*SignWriting*" usa símbolos visuales (conocidos también como glifos) para representar la forma de las manos, los movimientos y las expresiones faciales de cualquier lenguaje de señas. Es posible comparar el alfabeto de "*SignWriting*" con el alfabeto romano, el cual se utiliza para escribir inglés, español, francés y otras lenguas habladas.

Los símbolos en el alfabeto de "*SignWriting"* son internacionales y se utilizan para escribir los movimientos de cualquier lengua de señas en el mundo, de esta manera se provee de un mundo de literatura a las personas nacidas sordas, incluyendo libros, periódicos y diccionarios, que se utilizan para enseñar matemáticas, ciencias e historia a los niños sordos [8].

Para facilitar el aprendizaje, los símbolos que utiliza "*SignWriting*" representan diversas partes del cuerpo, en distintas posiciones, así como movimiento, como se observa en las figuras 1.2, 1.3, 1.4 y 1.5.

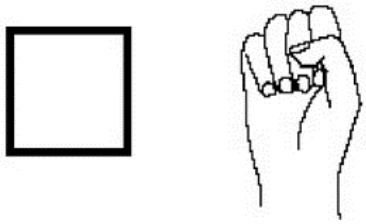


Figura 1.2 Representación del puño cerrado en "*SignWriting*".

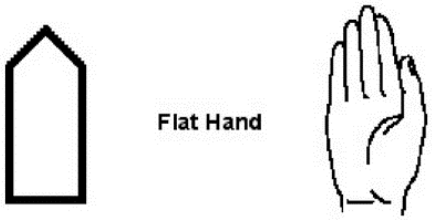



Figura 1.3 Representación de la palma extendida en "*SignWriting*".

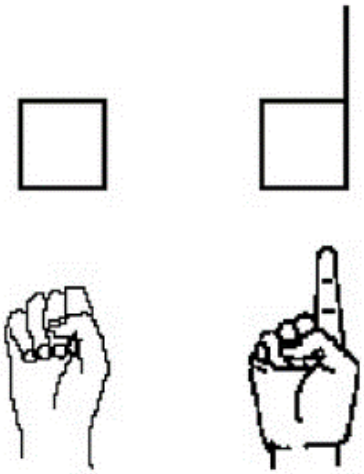


Figura 1.4 Representación de la letra S con el puño cerrado y el número uno en *"*SignWriting*".*

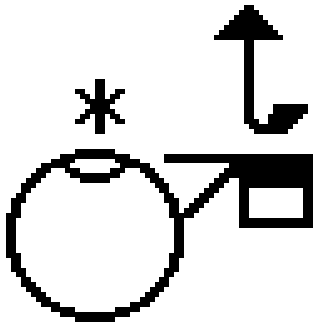


Figura 1.5 Representación de la seña de México de la LSM en "*SignWriting*".

### 1.1.8. Símbolo

Elemento u objeto material que, por convención o asociación, se considera representativo de una entidad, de una idea, de una cierta condición, entre otros [1]. En el caso de "*SignWriting*" es el concepto principal en el que se apoya para la creación de su alfabeto y la extrapolación de este a cualquier lengua de señas, sin importar lo diferentes que puedan ser, ya que permite asignarle un significado diferente a la combinación de sus símbolos.

### 1.1.9. Intérprete

El diccionario de la lengua española [1] define intérprete como aquella persona que explica a otras, en lengua que entienden, lo dicho en otra que les es desconocida.

### 1.1.10. Algoritmos de aprendizaje

Los algoritmos de aprendizaje se utilizan para predecir las señas ejecutadas por las personas y capturadas por los intérpretes de señas y aunque en la actualidad hay

descritas un número ingente de reglas de aprendizaje, estas se clasifican en dos grupos o categorías atendiendo a la presencia o no de un agente externo o supervisor que guíe el proceso de aprendizaje. Si dicho elemento supervisor está presente durante el aprendizaje, entonces se dice que el aprendizaje es supervisado, en caso contrario es no supervisado [9].

#### 1.1.10.1. Aprendizaje Supervisado

El aprendizaje supervisado se caracteriza por que la regla de aprendizaje incorpora un agente externo o supervisor que evalúa el proceso de aprendizaje, enseñándole como debería corregir su comportamiento dinámico durante el aprendizaje [9].

##### 1.1.10.1.1. Red Neuronal

Una red neuronal es un modelo conexionista cuyos elementos o nodos, conectados entre sí, simulan las funciones desempeñadas por las células cerebrales denominadas neuronas. Las redes neuronales artificiales exhiben sorprendentemente muchas de las funciones del cerebro siendo capaces de aprender, memorizar un conjunto de patrones, clasificarlos, inferir o generalizar a qué clase pertenece un nuevo objeto a partir de la experiencia acumulada por la red neuronal previamente [9].

##### 1.1.10.1.2. Máquina de Vectores de Soporte

Centrándose en la solución de problemas de clasificación binaria, donde el objetivo es estimar una respuesta que solo tiene dos estados, teniendo en cuenta una función de pérdida de clasificación, se define el objetivo de las Máquinas de Vectores de soporte como encontrar una solución que obtenga el menor riesgo posible en la clasificación [10].

#### 1.1.10.1.3. Máquinas de Vectores de Soporte Latente

La máquina de vectores de soporte latente es un algoritmo creado por Pedro Felzenszwalb que se utiliza para el reconocimiento de objetos deformables (como por ejemplo los peatones), ya que conceptualiza explícitamente la idea de múltiples subcomponentes que están enlazados entre sí por una estructura deformable [11], por lo que brindan excelentes resultados en el reconocimiento de señas.

### 1.1.10.2. Aprendizaje no supervisado

Se considera aprendizaje no supervisado, cuando el aprendizaje tiene lugar en ausencia de un elemento supervisor [9], en el caso de las redes neuronales en lugar de existir un conjunto de ejemplos en forma de pares entrada-salida, tomados de un proceso previo, el sistema de aprendizaje modifica los coeficientes de la red basándose en las características estadísticas del ambiente [12].

#### 1.1.10.2.1. Agrupamiento de datos

El agrupamiento de datos también llamado *"Clustering",* análisis de agrupamiento, análisis de segmentación o clasificación no supervisada, es un método de creación de grupos de objetos, donde los objetos en una unidad de agrupamiento son similares, mientras que los objetos en una unidad de agrupamiento distinta son ligeramente diferentes, por otra parte, en el agrupamiento las clases de los objetos no están definidas, deben ser descubiertas [13].

#### 1.1.10.2.2. Aprendizaje hebbiano

Los algoritmos de aprendizaje no supervisado de carácter hebbiano se basan en el postulado formulado por O. Hebb en 1949: “Cuando un axón de una celda A está suficientemente cerca para conseguir excitar una celda B y repetida o persistentemente toma parte en su activación, algún proceso de crecimiento o cambio metabólico tiene lugar en una o ambas celdas, de tal forma que la eficiencia de A, cuando la celda activa es B, aumenta”. De esta forma se identifican las celdas con neuronas fuertemente conectadas [14].

### 1.1.11.Detección de movimiento

La capacidad de detección de movimiento es una característica presente en los sistemas de visión de los seres vivos. Esta característica es tan importante para el sistema visual que en general predomina como detonante de la atención del sistema de visión. Es decir, el movimiento de un objeto tiende a captar la atención del sistema de visión en forma predominante [15].

### 1.1.12.Captura de movimiento en 3D

La captura de movimiento, control de movimiento, o “*Mocap”* (“*Motion Capture*”, Captura de Movimiento) son términos usados para describir el proceso de grabación de movimiento y la traducción de ese movimiento a un modelo digital. Se utiliza en el terreno militar, entretenimiento, deportes, aplicaciones médicas y para la robótica [16].

### 1.1.13.Leap Motion

Es un pequeño dispositivo USB ("*Universal Serial Bus*", Bus Serial Universal) que contiene tres emisores de luz infraroja y dos cámaras que capturan las luces infrarrojas de regreso, tiene la capacidad de detectar las palmas de las manos y los movimientos de los dedos; los datos de seguimiento que contienen la posición de ambos, así como la dirección y velocidad son accedidos mediante su SDK (Software Developer Kit); tiene un rango de detección de aproximadamente 0.025m – 0.6m [17].

### 1.1.14.OpenCV

OpenCV ("*Open Source Computer Vision Library*", Biblioteca de Visión por Computadora de Código Abierto) es una biblioteca de visión artificial y de aprendizaje por computadora de código abierto, construida para proveer una infraestructura común para las aplicaciones de visión por computadora y para acelerar el uso de la percepción de las máquinas en productos comerciales.

La biblioteca cuenta con más de 2500 algoritmos optimizados para detectar y reconocer rostros, identificar objetos, clasificar acciones humanas en videos, extraer modelos 3D de objetos, encontrar imágenes similares en una base de datos, entre muchos otros usos.

Tiene interfaces en C++, C, Python y Java, además de soportar los sistemas operativos Windows, Linux, Android y Mac OS [18].

### 1.1.15.Python

Python es un lenguaje de programación interpretado, multiparadigma, con tipado dinámico, multiplataforma y de código abierto, cuya última versión estable para el año 2018 es la 3.7, dada la naturaleza del proceso de aprendizaje automático donde se

llevan a cabo múltiples pruebas para lograr una configuración correcta, Python se adapta correctamente a este enfoque, además de incluir un amplio número de bibliotecas orientadas al área de inteligencia artificial [19].

### 1.1.16.Scikit-learn

Es una biblioteca de aprendizaje automático para el lenguaje de programación Python, incluye funcionalidades de clasificación, regresión y agrupamiento, además de diversos algoritmos de aprendizaje supervisado y no supervisado, está diseñada para trabajar con las bibliotecas científicas de Python NumPy y SciPy [20].

### 1.1.17.Pandas

Pandas es una biblioteca que sirve de extensión a NumPy para la manipulación y análisis de datos, ofrece estructuras de datos y operaciones a tablas numéricas y series temporales [21].

### 1.1.18.NumPy

Es una extensión del lenguaje de programación Python que permite mejorar el soporte para el manejo de vectores y matrices, incluyendo funciones matemáticas de alto nivel para operar con los mismos [22].

### 1.1.19.Scrum

Scrum es un proceso de control y administración que reduce la complejidad de los proyectos para enfocarse en crear productos que cumplan con las necesidades del

negocio, por otra parte, es un modelo de referencia que define un conjunto de prácticas y roles, y que puede tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará durante un proyecto.

El marco de trabajo Scrum induce a preguntarse por qué hacer algo implica tanto tiempo y esfuerzo y por qué se tienen tantos problemas para calcular cuánto tiempo y esfuerzo absorberá, por otra parte, abraza la incertidumbre y la creatividad, dota de estructura al proceso de aprendizaje, lo que permite a los equipos evaluar qué crearon e igualmente importante cómo lo hicieron, por medio de un ciclo de inspección y ajuste.

Los roles principales en Scrum son el "*Scrum Master*", que procura facilitar la aplicación de scrum y gestionar cambios, el "*Product Owner"*, que representa a los interesados y el equipo que ejecuta el desarrollo y demás elementos relacionados con él. Durante cada "*sprint"*, un periodo entre una y cuatro semanas, el equipo crea un incremento de software utilizable. El conjunto de características que forma parte de cada "*sprint*" viene del "*Product Backlog"*, que es un conjunto de requisitos de alto nivel priorizados que definen el trabajo a realizar [23].

### 1.1.20.Taiga

Taiga es una plataforma de administración de proyectos ágiles, enfocada en usuarios de tipo desarrollador, diseñador y gerentes de proyecto que permite el manejo de proyectos según los marcos de trabajo Scrum y Kanban, en caso de Scrum permite administrar historias de usuario, tareas, *sprints*, pizarras, roles de usuario, puntos de historias y genera a su vez de manera automática las gráficas de tareas pendientes para el producto y el sprint activo [24].

## 1.2. Planteamiento del problema

De acuerdo con los resultados de la Encuesta Nacional de la Dinámica Demográfica (ENADID) 2014, de los 119.9 millones de personas que habitan el país, 6% (7.2 millones) tienen discapacidad, de las cuales para el 33.5% (2.4 millones) su problema es auditivo [25].

Por otra parte, aunque existe una normativa para la certificación de intérpretes y la Lengua de Señas Mexicana (LSM) se declaró en el 2003 oficialmente como una lengua nacional, señalándola en la Ley General para la Inclusión de las Personas con Discapacidad en el artículo 14 : "La Lengua de Señas Mexicana es reconocida oficialmente como una lengua nacional y forma parte del patrimonio lingüístico con que cuenta la nación mexicana.", no se reportan programas para la capacitación de esta lengua en el sistema educativo nacional, de hecho se estima que existen menos de 40 intérpretes certificados [26].

La LSM es independiente de las versiones de lenguaje de señas de otros países, incluidos los latinoamericanos, por lo que es considerado un idioma único.

En la actualidad existen diversas investigaciones sobre desarrollo de intérpretes utilizando el periférico Kinect para hacer un seguimiento en tiempo real del cuerpo y las manos, ya que el mismo ofrece un marco de trabajo con seguimiento de formas y esqueletos con profundidad, es decir, un eje Z para el análisis de imágenes [27], [28], por otra parte, existen acercamientos al enfoque de una arquitectura basada en escenarios [29], que permita un mejor manejo de la complejidad del vocabulario y las interacciones necesarias para las actividades cotidianas de una persona muda, según el contexto y el idioma utilizado para entrenamiento, sin embargo, más allá del planteamiento, no se reporta una

implementación que utilice un periférico como Kinect o Leap Motion y la arquitectura de escenarios de manera conjunta, para la LMS.

## 1.3. Objetivo general y específicos

En la siguiente sección se detallan el objetivo general y los objetivos específicos de la tesis.

### 1.3.1. Objetivo general

A continuación, se presenta el objetivo general de la tesis.

Diseñar y construir un intérprete de LSM que permita la traducción en tiempo real de señas, en un contexto específico.

### 1.3.2. Objetivos específicos

En la siguiente lista se dan a conocer los objetivos específicos de esta tesis:

- Diseñar la arquitectura para el intérprete de señas.
- Seleccionar la tecnología a utilizar tanto de software como de hardware.
- Diseñar una interfaz gráfica que sea controlada con movimientos gestuales.
- Construir el intérprete con la tecnología seleccionada.
- Seleccionar el contexto para aplicar al caso de estudio.
- Probar el intérprete en el caso de estudio seleccionado.

## 1.4. Justificación

Existen muy pocos proyectos formales que abordan el problema de los intérpretes de LSM, entre estos, el proyecto desarrollado para el DIF de Jalisco, ofrece una traducción de español a LSM, sin embargo, no ofrece una solución de LSM a español, por otra parte, se indica que el LSM tiene características propias, que no permiten una extrapolación directa de su alfabeto (señas) a otros lenguajes [30].

El proyecto propuesto busca generar un intérprete que sirva como herramienta para una mejor integración de un segmento de la sociedad que actualmente se encuentra limitado en su interacción con otras personas en sus actividades diarias, permitiéndoles ser más independientes y ayudando a tener una sociedad más inclusiva mediante la aplicación de la tecnología existente.

# Capítulo 2. Estado de la práctica

En este capítulo se dan a conocer algunos trabajos relacionados directa o indirectamente con el proyecto de tesis.

## 2. 1. Trabajos relacionados

Leigh et al. [31] realizaron una serie de pruebas para determinar las fortalezas y debilidades del dispositivo "*Leap Motion"* aplicado en el reconocimiento del lenguaje de señas australiano conocido como "*Auslan"*, estas pruebas consistieron en evaluar el reconocimiento de la mano y los dedos, en distintas posiciones, así como la capacidad del dispositivo para identificar correctamente toda la mano al realizar movimientos propios del lenguaje de señas australiano, es importante señalar que estas pruebas se realizaron con un dispositivo de primera generación en el año 2013, por lo que también en [31] se señaló que varios de los problemas se corregirán en futuras actualizaciones del dispositivo.

Algunos de los problemas que encontraron Leig et al. [31] fueron originados por una API ("*Application Programming Interface",* Interfaz de Programación de Aplicaciones) aún incompleta y en etapas tempranas de desarrollo, que no permitía una correcta detección de los dedos de la mano en ciertas posiciones o al realizar rotaciones, sin embargo, también señalaron como fortaleza del dispositivo el costo del mismo, que permite que sea accesible a un mayor número de personas, por último mencionaron las limitaciones que se tienen en cuanto a señas que involucran contacto con el cuerpo o la cara, ya que el dispositivo está diseñado para el reconocimiento de las manos.

Barragan et al. [30] resaltaron la importancia de las características únicas que tiene el lenguaje de señas mexicano, y sobre todo el hecho de que estas características inherentes al lenguaje propio de México hacen que sea difícil extrapolar una solución

ya existente al mismo, por lo que es importante contar con una solución que contemple la estructura gramatical única con la que cuenta.

En [30] se mencionó la existencia de soluciones previas para el lenguaje de señas español europeo, utilizando avatares 3D, sin embargo, estos trabajos no son aplicables al contexto del idioma en México, por otra parte, se indicó la importancia de contar con un dispositivo de bajo costo, ya que a pesar de existir proyectos anteriores, el costo de los mismos es muy elevado y su exactitud por el contrario es muy reducida, por lo que su propuesta usa "*Java Media Framework API"*, MySQL, cámara y teclado, dividiendo el proyecto en cuatro etapas: reconocimiento de caracteres, sintaxis, semántica y análisis sintáctico. Se obtiene como resultado final el módulo de reconocimiento de caracteres y voz.

Simos et al. [32] exploraron las capacidades del dispositivo "*Leap Motion"* aplicadas al reconocimiento del alfabeto del lenguaje de señas griego, combinando los datos de posicionamiento 3D del dispositivo y usando algoritmos de SVM ("*Support Vector Machines"*, Máquinas de Vectores de Soporte) para aumentar el porcentaje de clasificación correcto llevándolos sobre el 99%.

En [32] también se mencionaron las diferentes técnicas y tecnologías usadas para el reconocimiento de señas, desde el uso de dispositivos "*wereables"*, técnicas de procesamiento de imágenes en 2D y dispositivos de captura de movimiento en 3D, resaltando las fortalezas de la última opción mencionada, para demostrarlo llevaron a cabo un experimento usando las señas de seis personas de prueba y una extra para la validación cruzada, por último, Simos et al. [32] propusieron dos métodos de reconocimiento de señas basados en los puntos reconocibles por el control "*Leap Motion*" llamados *"palmTranslation"* (Traduccion de palma) y *"boneTranslation"* (Traduccion de hueso), indicando como trabajos futuros la implementación de la traducción de palabras o frases.

Rafael et al. [29] esbozaron la idea de una arquitectura de interacción por escenarios para la gente sorda, destacando el hecho de que en México y el resto del mundo, a pesar de existir legislaciones que buscan promover la integración de las minorías en la sociedad, se necesitan herramientas y propuestas que ayuden a mejorar la inclusión, ya que un alto porcentaje de sordos en México aprenden el lenguaje de señas mexicano en la adolescencia o en la edad adulta, debido a que en las escuelas públicas los profesores no tienen la habilidad de dominar el lenguaje de señas, además de la falta de herramientas informáticas, por ello teniendo en cuenta todos los puntos antes mencionados introducen el concepto de escenario, entendido como pequeñas unidades de interacción que ocurren a diario, las cuales tienen diversos niveles de complejidad y vocabulario. Por último, en [29] llevaron a cabo pruebas del concepto de la arquitectura presentada con un prototipo utilizando Kinect y videos, obteniendo una precisión de alrededor del 90%.

Mapari et al. [33] realizaron pruebas para verificar la viabilidad del uso del control *"Leap Motion"* en el reconocimiento de señas del lenguaje de señas americano, concentrándose en el reconocimiento de señas "estáticas", es decir el alfabeto y los números del uno al diez, con las excepciones de las letras J y Z, además de los números 2 y 6, ya que la posición de la mano en estas señas es similar a las letras V y W.

Por tanto en [33] se establecieron un total de 48 características para identificar las señas del lenguaje de señas americano utilizando una red neuronal artificial, específicamente MLP ("*Multilayer Perceptron*", Perceptrón Multicapa), con una fuente de datos de 146 usuarios, los cuales realizaron 32 señas, para un total de 4672 registros, donde se obtuvo una exactitud en la clasificación del 90%.

Huenerfauth et al. [34] exploraron los beneficios de recibir retroalimentación de la ejecución de las señas para las personas que están aprendiendo el lenguaje de señas americano por medio de video, para llevar a cabo este experimento, examinaron y compararon los métodos tradicionales por medio de los cuales los estudiantes reciben

retroalimentación de su ejecución del lenguaje de señas, de esta forma diseñaron una serie de estímulos para indicarle a los estudiantes si la seña se había realizado correctamente o si tenía algún error.

Los métodos para mostrar retroalimentación comparados en [34] demostraron que los estudiantes mejoraron su desempeño cuando se les indicaba con mensajes de texto sobrepuestos las correcciones que debían hacer o los errores que tuvieron en la ejecución de las señas, cabe destacar que los autores realizaron un estudio del tipo "Mago de Oz" en las pruebas, es decir, se le hizo creer a los participantes de los experimentos que un sistema computacional fue el que evaluó su desempeño, sin embargo, fue un ser humano el que los evaluó.

Sun et al. [28] desarrollaron un experimento utilizando el sensor "Microsoft *Kinect*" en conjunto con la aplicación de un modelado de LSVM ("*Latent Support Vector Machine*", Máquina de Vectores de soporte Latente) para complementar los datos obtenidos por el sensor, es decir, los datos de imágenes 2D, y estructuras tridimensionales capturadas por "Microsoft Kinect" se utilizaron para mejorar la eficiencia en la captura de información relevante, que se usó para apoyar el LSVM. Un dato relevante en [28] es la comprobación de la eficacia de su modelo, para la predicción a nivel de palabras y sentencias, presentando una eficacia por encima del 82% y 84%, respectivamente.

Shang et al. [35] propusieron un sistema de reconocimiento de lenguaje de señas utilizando las señales "*Wi-Fi",* partiendo de la idea de que los diferentes movimientos de las manos y brazos generan distorsiones únicas en las señales inalámbricas, que a su vez se clasifican como patrones correspondientes con las señas de un lenguaje de señas, bajo el nombre de "*WiSign"* se presentó el sistema compuesto por tres periféricos, específicamente utilizaron un "*router"* TP-Link TL-WR1043ND y dos computadoras portátiles Lenovo.

En [35] partieron de la información que se obtiene desde el CSI ("*Channel State Information*", Canal de Información de Estado) de los dispositivos "*Wi-Fi*", la cual ya

había sido utilizada anteriormente en otras propuestas para dar seguimiento a las personas dentro de una habitación, registrar las conversaciones o las teclas presionadas en un teclado, como resultado final de su experimento obtuvieron un porcentaje de 93.8% de fiabilidad en el reconocimiento de señas.

Tal como Bianchi et al. [36] señalaron, las personas sordas se comunican esencialmente a través de gestos visuales según el lenguaje de señas que dominen, los cuales tienen una estructura diferente de los lenguajes vocales, por lo que las personas sordas tiene dificultades para aprender y usar las formas escritas de los lenguajes vocales, lo cual limita el acceso a textos y su consiguiente generación, una solución prometedora es "*SignWriting",* un marco de trabajo que permite escribir mediante símbolos, es por esto que los autores desarrollaron *"*SWift*" ("SignWriting improved fast transcriber"*, SignWriting transcriptor rápido mejorado*)*, el cual funciona como una aplicación Web que permite a los usuarios crear signos simples llamados glifos a través de una interfaz de usuario simple y amigable, aplicando los conceptos de "*SignWriting*", lo cual se espera ayude a romper la barrera electrónica, permitiendo que las personas sordas puedan escribir y consumir textos en su propio idioma.

Boulares et al. [37] resaltaron la falta de servicios de traducción de textos a lenguaje de señas en conjunto con los problemas que presentan algunas personas sordas con los tiempos de los verbos y la relación entre los géneros y los números al armar oraciones, además de las dificultades para formar imágenes mentales de ideas abstractas desde un texto, ya que prefieren la comunicación con lenguaje de señas en lugar del uso del idioma escrito u oral, por ello proponen un nuevo enfoque usando un intérprete de señas basado en Web, utilizando servicios Web y avatares, apoyándose en adaptaciones de XML para el manejo de los datos que componen la descripción de las animaciones de las señas, de esta manera optimizaron el uso de datos para que el servicio Web sea consumido por dispositivos móviles.

Ghanem et al. [38] realizaron un análisis y clasificación de las distintas soluciones existentes orientadas al reconocimiento de lenguaje de señas basadas en dispositivos móviles, resaltando el hecho de que para los usuarios de lenguajes de señas, comunicarse con personas que sí escuchan, representa un verdadero reto, de igual manera las herramientas de conectividad social no están disponibles para los usuarios de lenguaje de señas, a menos que estos tengan acceso a estas herramientas usando un lenguaje hablado y escrito como el español o el inglés con el cual podrían no sentirse confortables.

Tal como se señaló en [38], la computación móvil entró en una nueva era donde los dispositivos cuentan con múltiples procesadores, GPU ("*Graphics Processor Unit*", Unidad de Procesamiento Grafico) de alta calidad y cámaras de alta resolución, además se mencionó que las investigaciones muestran que el ASL ("*American Sign Language*", Lenguaje de Señas Americano) tiene cuatro componentes básicos: configuración de los dedos, movimiento de las manos, orientación de las manos y ubicación de las manos con respecto al cuerpo, todo ello permite la clasificación de las técnicas en dos enfoques, el primero basado en sensores usando guantes o los sensores internos del teléfono y el enfoque basado en visión usando diferentes técnicas para identificar la piel y movimiento de las manos.

Miller et al. [39] señalaron que los estudiantes sordos deben cambiar constantemente entre diversos recursos visuales para obtener toda la información necesaria cuando se realizan lecturas en los salones de clase, por ello se propuso probar el potencial que tiene el uso de lentes inteligentes en la reducción de los efectos negativos de los cambios de campo visual entre un recurso y otro, por medio de la consolidación de recursos en una sola vista.

En [39] se indicó que no se encontraron datos estadísticos suficientes para confirmar el beneficio sobre el uso de lentes inteligentes, sin embargo, los estudiantes que participaron en la investigación informaron que les resultó más fácil seguir la clase con la ayuda de los lentes, por lo que se espera continuar la investigación.

Gugenheimer et al. [40] indicaron que las personas sordas experimentan dificultades en la comunicación cara a cara con personas que escuchan, aun cuando se haga uso de tecnologías de asistencia centradas en traducción de lenguaje de señas en tiempo real, por ello la investigación que se realizó se centró en el impacto de esta tecnología en la calidad de las comunicaciones entre las personas sordas y que escuchan, por medio de un estudio de tipo “Mago de Oz”, donde se encontró que la calidad de la comunicación se degrada debido a las pausas en la conversación.

En [40] se propuso un cambio en la perspectiva de diseño de tecnologías asistidas, basados en la teoría Co-Cultural permitiendo a las personas que escuchan hacer señas en lugar de a los sordos escuchar, por lo que las tecnologías de asistencia no deben ser vistas solo como una herramienta para sordos, sino como una tecnología colaborativa.

En [41] se presentó la propuesta de un sistema de selección de caracteres que cualquier persona pudiera utilizar; el sistema facilita la comunicación entre personas sordas y aquellos usuarios que no utilizan el lenguaje de señas o el alfabeto de señas, utilizando solo los dedos; esta interacción se realiza sin el uso de teclado, ratón o equipos de cómputo que se utilicen físicamente como guantes o marcadores, ya que se calcula la letra seleccionada con las coordenadas de la cabeza, hombros y manos, para ello se usó un sensor “*Microsoft Kinect”*.

Miliccho et al. [42] mencionaron el crecimiento que tiene el campo de turismo accesible, el cual busca integrar a las personas con alguna discapacidad en actividades de viaje y placer, aprovechando los avances tecnológicos en dispositivos inteligentes, por ello desarrollaron dos aplicaciones que permiten a las personas sordas visitar el museo nacional romano “*Palazzo Massimo*” y la excavación arqueológica de “*Ostia Antica*”, haciendo uso de metodologías ágiles y del software Corona.

Cervantes et al. [43] señalaron los avances en el poder de cómputo, la miniaturización de los componentes y la mejora en los resultados obtenidos mediante el aprendizaje automático, su investigación se centró en la extracción de características, para ello seleccionaron inicialmente aquellas características que describen las propiedades básicas de la región a conocer, en total fueron nueve características: área de la región, redondez de la mano, longitud del borde de la mano, elongación de la mano definida por la longitud y ancho de la mano, las coordenadas x e y del centro de gravedad, la densidad, definida por la longitud de los bordes de la mano y área de esta, por otra parte, se apoyaron en el uso de máquinas de vectores de soporte para realizar la clasificación de las señas correspondientes, posteriormente generaron datos artificiales para mejorar la precisión del modelo, generando nuevos vectores de soporte.

Espejel et al. [44] aplicaron la técnica de máquina de vectores de soporte, para procesar un total de 15 015 imágenes, divididas en grupos de 500 imágenes para cada una de las 27 señas seleccionadas, correspondientes a las letras del alfabeto, el conjunto de imágenes que componen esta muestra la obtuvieron de un grupo de 40 diferentes voluntarios, con una resolución de 640 x 480 pixeles, con esta selección se realizó un preprocesamiento de las imágenes, utilizando una serie de características geométricas y el análisis de componentes principales, obteniendo una precisión general del clasificador de máquina de vectores de soporte del 91%.

## 2. 2. Análisis Comparativo

La tabla 2.1 muestra la información del análisis comparativo realizado a los artículos que tienen relación con el proyecto que se presenta.

Tabla 2.1 Análisis comparativo de los artículos relacionados

| Artículo | Problema | Objetivo | Resultados | Estado | Tecnología |
|---|---|---|---|---|---|
| Potter et al. [31] | Estadísticamente la gran mayoría de niños sordos nacen en hogares con padres que sí escuchan, por lo que existe una deficiencia en la comunicación dentro de la familia por la falta de conocimiento del lenguaje de señas, además de que esto deriva en problemas de lenguaje en los niños. | Exploración de la posibilidad de integración del control "*Leap Motion*" en soluciones de intérpretes de lenguaje de señas australiano. | Las pruebas se realizaron con un dispositivo de primera generación a unos meses de su lanzamiento, por lo que se encontraron problemas de identificación con ciertas señas, debido a una API incompleta. | Finalizado | "*Leap Motion"* |
| Barragán et al. [30] | En México existe un gran segmento de la población con problemas auditivos que están desatendidos y no logran una correcta integración en la sociedad, por otra parte, prácticamente no existen implementacio- | Crear una interfaz visual para un intérprete de lenguaje de señas mexicano, que en una primera etapa manejara las señas – escritura. | Se desarrolló el reconocimiento de voz, que busca y muestra una secuencia de video con la seña correspondiente, donde el módulo de semántica está completo al 70% | Finalizado | Java<br>MySQL<br>Video Cámara<br>Teclado<br>Pantalla |

Tabla 2.1 Análisis comparativo de los artículos relacionados (continuación).

| Artículo | Problema | Objetivo | Resultados | Estado | Tecnología |
|---|---|---|---|---|---|
| | nes orientadas al lenguaje de señas mexicano, además de que las soluciones desarrolladas tienen costos muy elevados para su adopción masiva. | | debido a la complejidad que representa. | | |
| Simos et al. [32] | Existen diferentes formas de capturar las señas de un lenguaje y según estas opciones a su vez varía la forma en que se procesan e identifican las mismas, desde la captura y procesamiento de imágenes hasta la captura de datos de posiciones en 3D, siendo estos últimos los que representan mejores resultados. | Presentar los métodos propuestos para el correcto reconocimiento del alfabeto del lenguaje de señas griego con el dispositivo *"Leap Motion".* | Se propusieron dos métodos de identificación de señas, basados en la información que brinda la API de "*Leap motion*", los resultados de clasificación correcta son superiores al 99%, además de haber identificado problemas con la interpretación. | Finalizado | *"Leap Motion"* |

Tabla 2.1 Análisis comparativo de los artículos relacionados (continuación).

| Artículo | Problema | Objetivo | Resultados | Estado | Tecnología |
|---|---|---|---|---|---|
| Rafael et al. [29] | Las personas sordas tienen problemas para interactuar con personas no sordas en diferentes ámbitos o escenarios y satisfacer correctamente sus necesidades de seguridad y bienestar. | Presentar una arquitectura modular donde cada módulo realiza una tarea específica, introduciendo el concepto de escenarios, definidos como pequeñas unidades de interacción que ocurren en la vida diaria. | Se desarrolló un prototipo donde se implementó la arquitectura de escenarios propuesta, manejando como escenario inicial el alfabeto del lenguaje de señas mexicano obteniendo un porcentaje de reconocimiento del 90%. | Finalizado | No se menciona |
| Mapari et al. [33] | Los avances en la tecnología crean nuevas oportunidades en el campo de la interacción humano-computadora, siendo el uso del sensor "*Leap Motion*" una nueva op - | Comprobar la viabilidad de usar el sensor "*Leap Motion*" para traducir las señas estáticas del lenguaje de señas americano. | Se realizaron pruebas de reconocimiento de señas combinando los datos recabados por el dispositivo "*Leap Motion*", un MLP | Finalizado | *"Leap Motion"* |

Tabla 2.1 Análisis comparativo de los artículos relacionados (continuación).

| Artículo | Problema | Objetivo | Resultados | Estado | Tecnología |
|---|---|---|---|---|---|
| | ción para interpretar el lenguaje de señas americano. | | para analizar un conjunto de datos conformado por 4672 señas registradas, obteniendo un promedio de clasificación correcta de 90%. | | |
| Huenerfauth et al. [34] | Se demostró previamente que los niños sordos se benefician de la exposición al lenguaje a temprana edad, sin embargo, los padres no tienen acceso a herramientas que les ayuden en el reforzamiento del aprendizaje del lenguaje de señas americano. | Realizar un estudio del tipo “Mago de Oz” para comparar los resultados de brindar retroalimentación a los niños por medio de grabaciones en video, donde se indicara en tiempo real las correcciones o errores que estos habían tenido. | Se encontró que los niños mejoraban su ejecución al recibir retroalimentación en forma de texto sobrepuesta al video grabado previamente, así lograban identificar de mejor manera las mejoras a realizar. | Finalizado | Video |

Tabla 2.1 Análisis comparativo de los artículos relacionados (continuación).

| Artículo | Problema | Objetivo | Resultados | Estado | Tecnología |
|---|---|---|---|---|---|
| Sun et al. [28] | El reconocimiento de lenguajes de señas por medio de propuestas basadas en visión generó mucho interés en los investigadores del área de visión por computadora, además aprovechando el bajo costo del sensor "*Microsoft Kinect"* y las amplias capacidades con las que cuenta para detectar profundidad, se exploran nuevas formas de mejorar la identificación de palabras y sentencias de los lenguajes de señas. | Se propuso un experimento para verificar la utilidad de seleccionar determinados cuadros de la secuencia de video como discriminativos y representativos para las señas del lenguaje de señas americano complementando los datos obtenidos por el sensor "*Microsoft Kinect"* con un modelado de *"Latent SVM".* | Se obtuvo un 82.9% y 84.1% de identificación correcta para las palabras y sentencias, respectivamente, con lo cual se comprueba la eficacia del enfoque propuesto, se continúa con la investigación para mejorar los porcentajes. | En progreso | *"Microsoft Kinect"* *"Latent SVM"* |
| Shang et al. [35] | El lenguaje de señas es importante ya que permite la comunicación de las personas sordas y al | Los diferentes movimientos de las manos y brazos generan distorsiones en las señales Wi-Fi, | Los resultados de los experimentos demostraron que se cuenta con una exactitud en el re- | Finalizado | CSI TP-Link TL-WR1043ND Lenovo X100e SVM |

Tabla 2.1 Análisis comparativo de los artículos relacionados (continuación).

| Artículo | Problema | Objetivo | Resultados | Estado | Tecnología |
|---|---|---|---|---|---|
| | mismo tiempo es el medio de generación de cultura para la comunidad sorda, además ya que el lenguaje de señas utiliza el cuerpo para dar significado, las técnicas de HAR ("*Human Activity Recognition*", Reconocimiento de Actividades Humanas) se utilizan para aplicaciones de traducción. | que a su vez generan patrones únicos que son identificados y asignados a señas específicas, utilizando para esto la información generada en el CSI. | conocimiento de señas del 93.8%, al utilizar un "*router*" y dos computadoras portátiles. | | |
| Bianchi et al. [36] | Las personas sordas se comunican esencialmente por medio de gestos y señas, aun cuando conozcan un lenguaje vocal, aunado a esto la estructura de los lenguajes de señas y | Se presentó el editor *Swift* basado en web¸ que permite escribir y transcribir usando el marco de trabajo "*SignWriting"*. | Se creó el editor con una interfaz gráfica amigable y con un alto grado de usabilidad , además de ser diseñado de manera modular para que se imple- | Finalizado | "*SignWriting*" |

Tabla 2.1 Análisis comparativo de los artículos relacionados (continuación).

| Artículo | Problema | Objetivo | Resultados | Estado | Tecnología |
|---|---|---|---|---|---|
| | los lenguajes vocales es diferente, lo cual genera problemas a las personas sordas, ya que terminan la traducción desde su lenguaje de señas al lenguaje vocal cuando intentan escribir o leer. | | mente en cualquier lugar donde el uso de "*SignWriting*" sea recomendado como una alternativa a la escritura de lenguajes vocales. | | |
| Boulares et al. [37] | Actualmente las tecnologías Web representan una forma eficiente de asegurar la comunicación entre audiencias grandes y heterogéneas, sin embargo, la información Web existente está en su mayoría basada en contenido de texto y multimedia, por lo que las personas sordas tienen dificultad para acceder a la misma. | Crear un intérprete de lenguaje de señas basado en Web, que permita la traducción automática de texto usando tecnología de avatares y servicios Web. | Se presentó un servicio Web de traducción basado en X3D ("*Extensible 3D Graphics",* Graficos 3D Extensibles) y SML ("*Sign Modelling Language*", Lenguaje de Modelado de Señas) para dispositivos Android, que permite la traducción de texto | Finalizado | *"Blender"*<br>SML<br>X3D |

Tabla 2.1 Análisis comparativo de los artículos relacionados (continuación).

| Artículo | Problema | Objetivo | Resultados | Estado | Tecnología |
|---|---|---|---|---|---|
| | | | a lenguaje de señas ya sea ingresado directamente o tomado desde los mensajes sms recibidos en el teléfono. | | |
| Ghanem et al. [38] | Las personas sordas en todo el mundo usan los lenguajes de señas para cubrir sus necesidades de comunicación, además de que los avances en la tecnología de los dispositivos móviles ofrecen nuevas posibilidades para proveer aplicaciones más amigables con este tipo de usuarios. | Clasificar las técnicas más recientes de los sistemas de reconocimiento de lenguaje de señas basadas en dispositivos móviles. | Se clasificaron las técnicas de reconocimiento de lenguaje de señas en basadas en visión y basadas en sensores. | Finalizado | No se menciona |
| Miller et al. [39] | Los estudiantes sordos deben obtener información de diferentes fuentes vi- | Realizar una investigación para comprobar el beneficio de usar len- | No se obtuvieron datos estadísticos contundentes sobre la mejora directa en | Finalizado | Google Glass<br>Moverio BT-200<br>Java<br>VLC media player |

Tabla 2.1 Análisis comparativo de los artículos relacionados (continuación).

| Artículo | Problema | Objetivo | Resultados | Estado | Tecnología |
|---|---|---|---|---|---|
| | suales para poder comprender correctamente las clases. | tes inteligentes para ayudar a los alumnos sordos durante las clases. | la compresión de las clases al utilizar los lentes inteligentes, sin embargo, los estudiantes que participaron en la investigación indicaron que les resultó más fácil seguir la clase al utilizar los lentes. | | |
| Gugenheimer et al. [40] | Las personas sordas enfrentan dificultades en las experiencias de comunicación cara a cara, aun con el uso de tecnología de asistencia de traducción en tiempo real, originadas principalmente por las pausas. | Llevar a cabo una serie de pruebas para validar la propuesta de modificar el enfoque de diseño de tecnologías de asistencia en tiempo real, para permitir que las personas que escuchan puedan hacer señas. | Se realizaron pruebas de concepto utilizando cartulinas, proyectores y teléfonos, con la limitante en algunos casos de usar sinónimos para utilizar solo palabras que incluyeran señas con las manos y no | Finalizado | Cámara<br>Celular<br>Proyector |

Tabla 2.1 Análisis comparativo de los artículos relacionados (continuación).

| Artículo | Problema | Objetivo | Resultados | Estado | Tecnología |
|---|---|---|---|---|---|
| | | | el resto del cuerpo, aun así se demostró que las tecnologías de asistencia de traducción afectan directamente la calidad de la comunicación y que un cambio en la perspectiva de diseño permitiría balancear las relaciones de poder durante las conversaciones. | | |
| Shin et al. [41] | El lenguaje de señas y el alfabeto usando los dedos son usados para realizar la comunicación entre las personas sordas, sin embargo, las personas que escuchan pueden tener problemas debido a la | El principal problema que hay con las propuestas existentes para el reconocimiento de señas y caracteres es que se necesita un conocimiento previo con respecto a las se- | Se generó un sistema que facilita la comunicación entre personas sordas y usuarios que no usan el lenguaje de señas o el alfabeto en señas, mediante el | Terminado | *"Microsoft Kinect"* |

Tabla 2.1 Análisis comparativo de los artículos relacionados (continuación).

| Artículo | Problema | Objetivo | Resultados | Estado | Tecnología |
|---|---|---|---|---|---|
| | falta de conocimiento de estos lenguajes. | ñas, por ello se busca validar un sistema que detecta coordenadas para seleccionar letras. | sensor "*Microsoft Kinect*". | | |
| Miliccho et al. [42] | El turismo accesible busca integrar a las personas con problemas de movilidad, visión o escucha en actividades de viajes y placer y gracias a los avances tecnológicos este campo ha crecido en la última década. | Mostrar el emprendimiento de aplicaciones de turismo accesible para celulares inteligentes y tabletas para la comunidad sorda. | Se diseñaron y desarrollaron dos aplicaciones para teléfonos inteligentes y tabletas, dirigidos a la comunidad sorda, específicamente orientadas en las visitas a sitios arqueológicos, implementados en el museo nacional romano *"Palazzo Massimo*" y en la excavación arqueológica de "*Ostia Antica*". | En proceso | *"Corona software development kit"* Metodología Ágil Android |

Tabla 2.1 Análisis comparativo de los artículos relacionados (continuación).

| Artículo | Problema | Objetivo | Resultados | Estado | Tecnología |
|---|---|---|---|---|---|
| Cervantes et al. [43] | Obtener una buena precisión de generalización en los clasificadores, requiere de un conjunto de datos amplio que pueden no ser representativos o relevantes. | Mejorar la precisión obtenida al utilizar el clasificador de máquina de vectores de soporte, identificar características relevantes y generar datos artificiales. | Se obtuvo una mejora en la precisión del clasificador al introducir los datos artificiales generados. | Concluido | Máquina de vectores de soporte Generación de datos artificiales |
| Espejel et al. [44] | A pesar del elevado número de personas con discapacidad auditiva, resulta difícil encontrar personas que puedan entender la lengua de señas mexicana, lo cual limita a las personas sordas en el manejo de la comunicación con el resto de la comunidad. | Obtener un intérprete de lengua de señas mexicana utilizando técnicas de extracción de características y análisis de componentes principales. | Se comprobó la metodología experimental propuesta para el reconocimiento de la lengua de señas mexicana, obteniendo una precisión del 91% para la clasificación. | Concluido | Máquina de vectores de soporte Análisis de componentes principales |

Después de analizar los artículos citados, se concluye que el desarrollo de intérpretes de lenguaje de señas es un área de interés con diversas propuestas, integrándose recientemente nuevas tecnologías de hardware de reconocimiento 3D, que permiten desarrollar prototipos y soluciones de manera más efectiva, aprovechando los bajos costos de estos y el aumento en la capacidad de computadoras y dispositivos móviles, por ello se considera que si bien en los trabajos mencionados ya se hicieron pruebas con lenguajes de señas de otros países, es una excelente oportunidad el desarrollar un intérprete de lenguaje de señas mexicano usando el hardware "*Leap Motion"* adaptado a las necesidades del país.

## 2. 3. Propuesta de solución

Para mejorar la calidad de vida de las personas sordas al interactuar con su entorno social, se propone el diseño y desarrollo de un intérprete de lengua de señas mexicana, que permita la interpretación de las señas en tiempo real, traduciéndolas al idioma español, con un módulo que permita el entrenamiento de nuevo vocabulario y su agrupamiento en contextos, para de esta manera facilitar la implementación del interprete en diversos ámbitos, como por ejemplo el reforzamiento del aprendizaje de la lengua de señas mexicana o en la solicitud de información en instituciones públicas, brindando una opción reutilizable para contrarrestar el déficit de intérpretes que existe en el país.

En la tabla 2.2 se muestra la información de los elementos seleccionados para la presente propuesta del intérprete de lengua de señas mexicana.

Tabla 2.2 Elementos propuestos para conformar el intérprete de lengua de señas mexicana.

| Dispositivo de captura de movimiento en 3D | Biblioteca de software para la captura de movimiento | Lenguaje de programación | Metodología |
|---|---|---|---|
| Leap Motion | OpenCV | Python | Scrum |

Se sugiere el uso del dispositivo de captura de movimiento 3D Leap Motion en conjunto con la biblioteca OpenCV, utilizando el lenguaje de programación Python, ya que existen múltiples bibliotecas de inteligencia artificial de alta calidad y disponibles sin costo, de igual manera se sugiere utilizar el marco de trabajo Scrum para llevar a cabo el manejo del proyecto.

### 2.3.1. Justificación de la tecnología seleccionada

Esta decisión se fundamenta en el análisis de la relación costo beneficio de la combinación de tecnologías de hardware y software que se especifica a continuación:

**Dispositivo de captura de movimiento en 3D:** El control Leap Motion es el dispositivo que ofrece mayores prestaciones a un menor costo, mostrando además estabilidad en su desarrollo, ya que se encuentra disponible en el mercado desde el año 2013 y a lo largo de los años la compañía que lo comercializa mantiene actualizaciones constantes de su API, corrigiendo errores y mejorando el rendimiento del producto en cuanto a la detección de distintas posiciones de las manos, por otra parte, durante la investigación del estado de la práctica, se comprobó que puede ser utilizado satisfactoriamente en proyectos relacionados con intérpretes de lenguaje de señas.

**Biblioteca de Software para la captura de movimiento:** Se seleccionó la biblioteca OpenCV como biblioteca para el procesamiento de imágenes, debido a que es una solución de software robusta y utilizada ampliamente como se comprobó en el estudio del estado de la práctica, de esta manera será utilizada en el proyecto como auxiliar en la interpretación de ideogramas, por otra parte, presenta múltiples interfaces para

implementarse en distintos lenguajes, por lo que es la opción que resulta más flexible, cabe destacar que los lenguajes de los cuales dispone interfaces son compatibles con los lenguajes soportados por el control Leap Motion.

**Lenguaje de Programación:** Se seleccionó el lenguaje de programación Python debido al buen desempeño que presenta para proyectos de aprendizaje automático, dada su naturaleza explorativa, al ser un lenguaje interpretado de alto nivel, permite realizar pruebas de concepto de manera rápida y eficiente, además de contar con un ecosistema de bibliotecas orientadas específicamente al manejo de técnicas de aprendizaje automático [19].

**Metodología:** Como metodología de desarrollo de software se seleccionó el marco de trabajo Scrum debido a que el producto final en esta propuesta de tesis es el desarrollo de un intérprete de señas, por lo que la orientación de desarrollo iterativo e incremental que maneja el marco de trabajo permite enfocarse en la entrega del producto de software, dando flexibilidad para la inclusión de los artefactos de ingeniería de software que se consideren necesarios para sustentar los requerimientos y características de la solución presentada, ya que se buscará adaptar el marco de trabajo para acelerar el desarrollo y mantener la calidad del software generado con la documentación correspondiente.

## Capítulo 3. Aplicación de la metodología

El presente capítulo tiene como objetivo presentar el desarrollo de la solución, siguiendo las pautas recomendadas en la metodología seleccionada, que en este caso fue Scrum.

### 3.1. Descripción de la solución

En la siguiente sección se realiza una descripción detallada de la solución propuesta.

**1. Módulo de captura de información:**

1.1. **Captura de información 3D**: Este módulo realiza la captura directa de la información correspondiente a la posición y desplazamiento de los brazos y manos del usuario, por medio del dispositivo de captura de movimiento en 3D LeapMotion, de esta manera se almacenan o utilizan directamente estos valores para alimentar repositorios o compararlos con la información almacenada previamente.

1.2. **Captura de imágenes:** Este módulo realiza la captura de imágenes a través de una cámara Web, que posteriormente se procesan mediante la biblioteca openCV.

2.1. **Módulo de Comparación**

2.1.1. **Comparación con repositorio:** Los datos capturados directamente con el dispositivo de captura de movimiento en 3D se comparan directamente con los guardados en el repositorio, ya que mediante su API se obtienen los valores originales, sin tener que transformarlos, de esta manera se busca

una coincidencia de las posiciones y movimientos en el vocabulario de contexto para identificarlos y enviarlos al siguiente módulo.

2.2. **Módulo de procesamiento**

2.2.1. **Procesamiento de imágenes:** Las imágenes capturadas con la videocámara se procesan mediante la biblioteca de visión por computadora openCV, para obtener la posición de los marcadores y mapearlos a un sistema de coordenadas con ejes *x* y *y* que se creó, ya que en el repositorio deberán guardarse estos valores y no las imágenes.

2.3. **Módulo de comparación:**

2.3.1. **Comparación con repositorio:** Una vez obtenidas las características de las imágenes correspondientes al video capturado, se procede a realizar la comparación con el repositorio de manera rápida y eficiente, para buscar coincidencias con las señales guardadas según el vocabulario del contexto.

**3. Vocabulario de contexto:** Este módulo contendrá la información referente a las señas que conforman el vocabulario del contexto que se esté trabajando en ese momento, una vez identificada una coincidencia se enviará ésta al siguiente módulo.

4. **Módulo de análisis compartido**

4.1. **Análisis de información obtenida:** Este módulo se encargará de determinar si es suficiente la información obtenida mediante los datos del dispositivo de captura de movimiento en 3D o si es necesario apoyarse en los resultados del procesamiento y análisis de la videocámara, procediendo a un análisis de manera conjunta de ambas fuentes de información.

5. **Módulo de comparación de diccionario de contexto**

5.1. **Repositorios:** Este módulo contiene la información relacionada con el vocabulario del contexto actual almacenada en repositorios y accedida por el

módulo de análisis compartido, que lo utiliza para determinar la correcta traducción de las señas.

6. **Salida Procesada:** Será el mensaje traducido según el resultado del análisis de los datos obtenidos a través de la videocámara y el dispositivo de captura de movimiento en 3D, apoyadas en el vocabulario del contexto seleccionado.

La arquitectura preliminar se muestra en la figura 3.1.

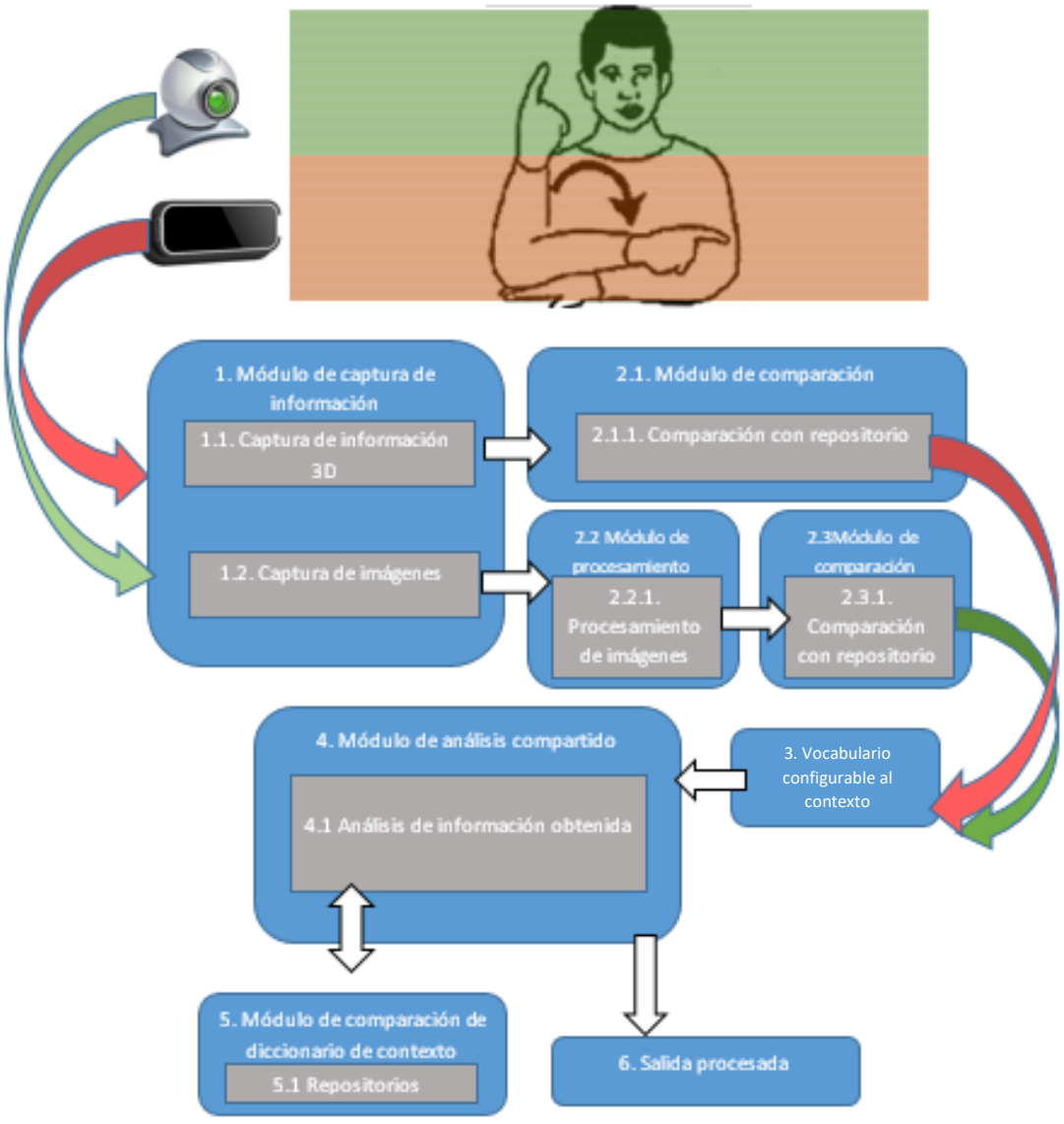



Figura 3.1 Módulos propuestos para el intérprete de lengua de señas mexicana.

## 3.2. Marco de trabajo Scrum

En la siguiente sección se detalla la implementación e iteración del marco de trabajo Scrum aplicado a este proyecto.

### 3.2.1 Historias de Usuario

En las tablas 3.1 a la 3.14, se observan las historias de usuario que corresponden a la planificación original de los módulos propuestos.

Tabla 3.1 Historia de usuario INV-01: Análisis de dispositivos de captura de imágenes en 3D.

| Código | INV-01 | |
|---|---|---|
| Descripción | Realizar un Análisis de los dispositivos de captura de imágenes en 3D, tomando en cuenta los puntos a favor y en contra para elegir el que se considere la mejor opción. | |
| Tipo | Obligatoria | |
| Restricciones | | Relaciones |
| Debe ser de las primeras actividades a realizar | | Ninguna |

Tabla 3.2 Historia de usuario INV-02: Análisis de biblioteca de captura de imágenes.

| Código | INV-02 | |
|---|---|---|
| Descripción | Realizar un Análisis de las bibliotecas de captura y procesamiento de imágenes, tomando en cuenta los puntos a favor y en contra para elegir la que se considere la mejor opción. | |
| Tipo | Obligatoria | |
| Restricciones | | Relaciones |
| Debe ser de las primeras actividades a realizar | | Ninguna |

Tabla 3.3 Historia de usuario INV-03: Análisis de tecnologías de almacenamiento.

| Código | INV-03 | |
|---|---|---|
| Descripción | Analizar las tecnologías de almacenamiento existentes, buscando la que tenga índices óptimos para datos de captura de movimiento | |
| Tipo | Obligatoria | |
| Restricciones | | Relaciones |
| Debe ser de las primeras actividades a realizar | | Ninguna |

Tabla 3.4 Historia de usuario DIA-01: Diseño de arquitectura.

| Código | DIA-01 | |
|---|---|---|
| Descripción | Diseñar la arquitectura del intérprete de Lengua de Señas Mexicana. | |
| Tipo | Obligatoria | |
| Restricciones | | Relaciones |
| Debe ser de las primeras actividades a realizar | | Ninguna |

Tabla 3.5 Historia de usuario DMC-01: Desarrollo del módulo de captura de información.

| Código | DMC-01 | |
|---|---|---|
| Descripción | Desarrollo del módulo de captura de información. | |
| Tipo | Obligatoria | |
| Restricciones | | Relaciones |
| Debe realizarse primero la actividad MOD-01 | | Ninguna |

Tabla 3.6 Historia de usuario DMI-01: Desarrollo del módulo de procesamiento de imágenes configuración de entorno.

| Código | DMI-01 |
|---|---|
| Descripción | Desarrollo del módulo de procesamiento de imágenes configuración de entorno. |
| Tipo | Obligatoria |
| Restricciones | Relaciones |
| Debe realizarse primero la actividad MOD-01 | DMI-02 |

Tabla 3.7 Historia de usuario DMI-02: Desarrollo del módulo de procesamiento de imágenes selección y aplicación de algoritmos.

| Código | DMI-02 |
|---|---|
| Descripción | Desarrollo del módulo de procesamiento de imágenes selección y aplicación de algoritmos. |
| Tipo | Obligatoria |
| Restricciones | Relaciones |
| Debe realizarse primero la actividad DMI-02 | Ninguna |

Tabla 3.8 Historia de usuario DRD-01: Diseño del repositorio.

| Código | DRD-01 |
|---|---|
| Descripción | Diseño del repositorio de datos de captura de movimiento. |
| Tipo | Obligatoria |
| Restricciones | Relaciones |
| Debe realizarse primero la actividad MOD-01 | Ninguna |

Tabla 3.9 Historia de usuario DMC-02: Desarrollo del módulo de comparación con repositorio.

| Código | DMC-02 |
|---|---|
| Descripción | Desarrollo del módulo de comparación con repositorio. |
| Tipo | Obligatoria |
| Restricciones | Relaciones |
| Debe realizarse primero la actividad DRD-01, DMC-01 | Ninguna |

Tabla 3.10 Historia de usuario DMV-01: Desarrollo del módulo de vocabulario de contexto.

| Código | DMV-01 |
|---|---|
| Descripción | Desarrollo del módulo de vocabulario de contexto. |
| Tipo | Obligatoria |
| Restricciones | Relaciones |
| Ninguna | Ninguna |

Tabla 3.11 Historia de usuario DMA-01: Desarrollo del módulo de análisis compartido.

| Código | DMA-01 |
|---|---|
| Descripción | Desarrollo del módulo de análisis compartido. |
| Tipo | Obligatoria |
| Restricciones | Relaciones |
| Debe realizarse primero la actividad DMC-02 | Ninguna |

Tabla 3.12 Historia de usuario SCE-01: Selección caso de estudio.

| Código | SCE-01 |
|---|---|
| Descripción | Selección de un caso de estudio. |
| Tipo | Obligatoria |
| Restricciones | Relaciones |
| Ninguna | Ninguna |

Tabla 3.13 Historia de usuario CDC-01: Desarrollo del módulo de comparación del diccionario de contexto.

| Código | CDC-01 |
|---|---|
| Descripción | Desarrollo del módulo de comparación de diccionario de contexto. |
| Tipo | Obligatoria |
| Restricciones | Relaciones |
| Ninguna | Ninguna |

Tabla 3.14 Historia de usuario PCE-01: Pruebas asociadas al caso de estudio seleccionado.

| Código | PCE-01 |
|---|---|
| Descripción | Realizar pruebas con el caso de estudio seleccionado. |
| Tipo | Obligatoria |
| Restricciones | Relaciones |
| Ninguna | Ninguna |

### 3.2.2 Roles

Scrum plantea tres roles principales, dueño del producto, equipo de desarrollo y Scrum *master*, sin embargo, para la ejecución de este desarrollo estos roles son ejecutados completamente por el tesista.

### 3.2.3 *Sprint*

El *Sprint* es un bloque de tiempo de un mes o menos, durante el cual se crea un incremento de producto “terminado”, utilizable y potencialmente desplegable, en el caso del desarrollo de este proyecto, el *Sprint* se ejecuta en periodos mensuales.

### 3.2.4 *Sprint* 1 Enero 2018

Para el primer *Sprint* se contó con la siguiente lista de producto (*Product Backlog*) basada en las historias de usuario:

- INV-01 Revisión dispositivos 3D.
- INV-02 Revisión de bibliotecas.
- INV-03 Analizar las tecnologías de almacenamiento.
- DIA-01 Diseño de arquitectura.
- DMC-01 Desarrollo del módulo de captura de información.
- DMI-01 Desarrollo del módulo de procesamiento de imágenes.
- DMI-02 Desarrollo del módulo de procesamiento de imágenes
- DRD-01 Diseño del repositorio de datos de captura de movimiento.
- DMC-02 Desarrollo del módulo de comparación con repositorio.

- DMV-01 Desarrollo del módulo de vocabulario de contexto.
- DMA-01 Desarrollo del módulo de análisis compartido.
- SCE-01Selección de un caso de estudio.
- CDC-01 Desarrollo del módulo de comparación de diccionario de contexto.
- PCE-01 Realizar pruebas con el caso de estudio seleccionado.

Las historias de usuario generadas inicialmente totalizan 370 puntos, lo cual genera la gráfica de pendientes de producto que se muestra en la figura 3.2.

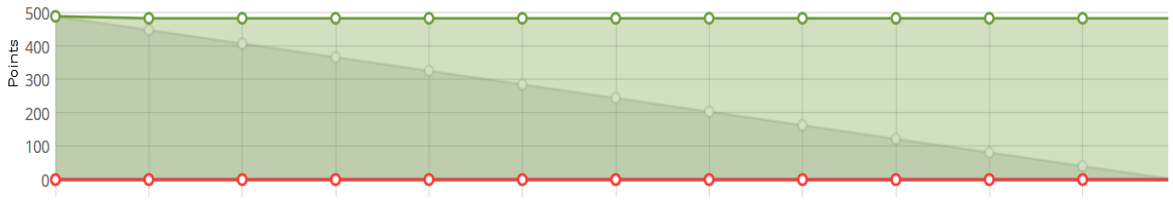



Figura 3.2 Gráfica de pendientes de producto inicial.

A su vez estas historias de usuario derivaron en la siguiente lista de pendientes del primer *sprint* (*Sprint Backlog*), junto a sus correspondientes tareas:

- INV-01 Revisión dispositivos 3D.
  - Revisar las características del dispositivo de captura 3d *LeapMotion*
- INV-02 Revisión de bibliotecas.
  - Revisión de compatibilidad de bibliotecas con los lenguajes de programación manejados por el dispositivo de captura.
- INV-03 Analizar las tecnologías de almacenamiento.
  - Revisión de las tecnologías de almacenamiento reportadas en el estado del arte para elegir la que cubra las necesidades del proyecto.

Después de realizar un análisis de las diferentes tecnologías de almacenamiento disponibles, se determinó que para la realización del proyecto se buscaría seleccionar y utilizar un ORM (*Object Relationship Mapper*), para obtener independencia y flexibilidad con respecto a la sección realizada, debido a que el desarrollo del proyecto será de manera incremental, pero siempre buscando tener un producto utilizable, es decir, se busca tener la flexibilidad de cambiar de sistema gestor de base de datos según se considere conveniente al agregar nuevas características al sistema que no estaban contempladas anteriormente, la opción seleccionada fue SQLAlchemy, debido a que es la opción más utilizada para los proyectos desarrollados en Python y permite una gran flexibilidad al soportar múltiples gestores de base de datos.

A continuación, se muestra la gráfica de trabajo pendiente generada para ese *sprint* en la figura 3.3.

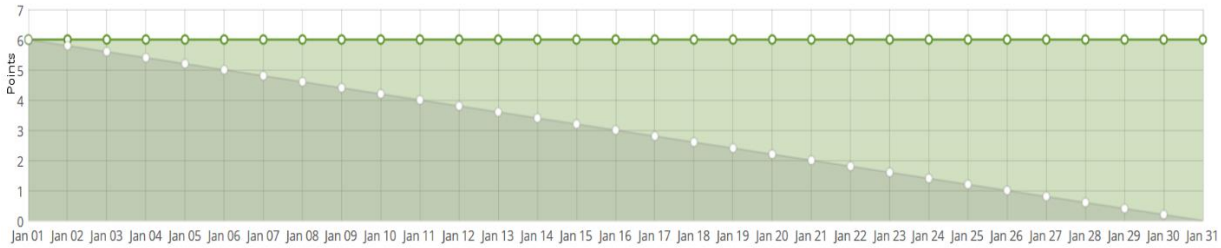



Figura 3.3 Gráfica de trabajo pendiente para el *Sprint* 1 Enero 2018.

El total de puntos de las tareas cerradas para este *sprint* fue de seis tal como se observa en la captura de pantalla de la interfaz de Taiga que se muestra en la figura 3.4.

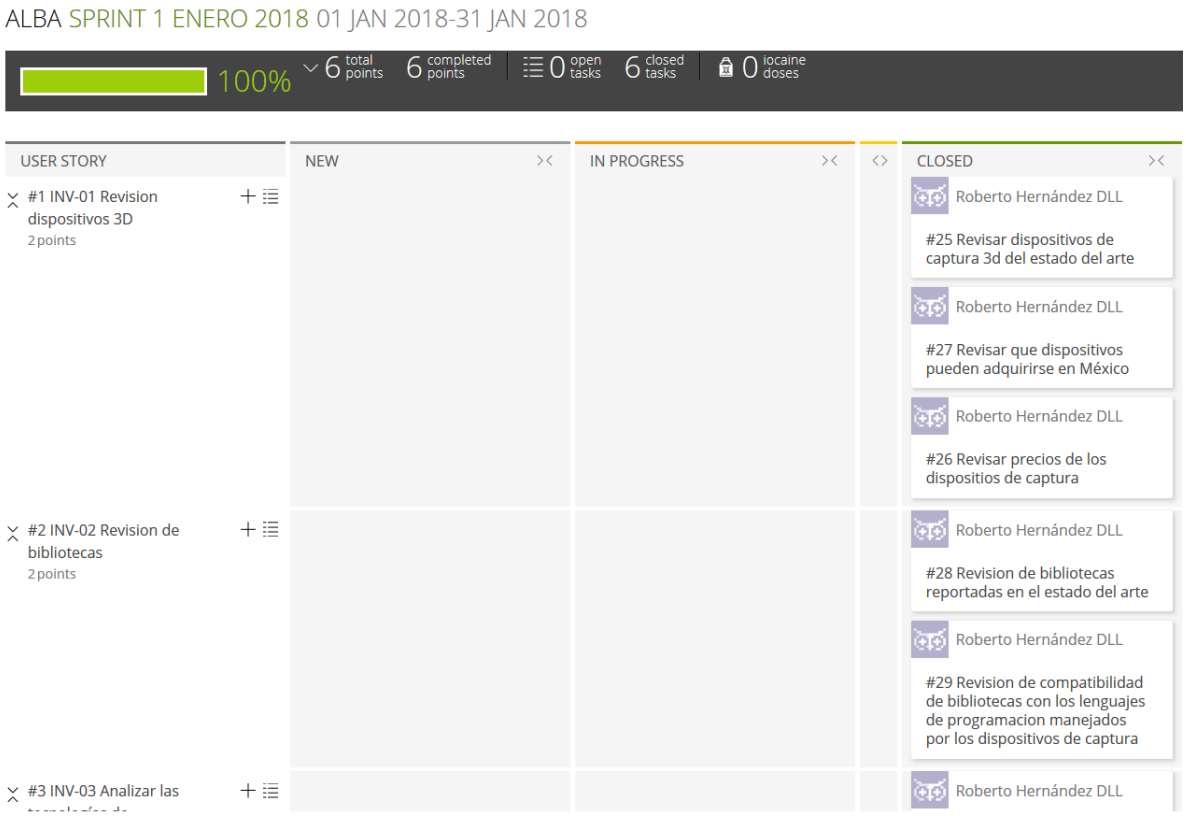



Figura 3.4 Interfaz de Taiga para manejo de tareas de Sprint 1 Enero 2018.

### 3.2.5 *Sprint* 2 Febrero 2018

Para el segundo *Sprint* se contó con la siguiente lista de producto (*Product Backlog*) basada en las historias de usuario:

- DIA-01 Diseño de arquitectura.
- DMC-01 Desarrollo del módulo de captura de información.
- DMI-01 Desarrollo del módulo de procesamiento de imágenes.
- DMI-02 Desarrollo del módulo de procesamiento de imágenes
- DRD-01 Diseño del repositorio de datos de captura de movimiento.
- DMC-02 Desarrollo del módulo de comparación con repositorio.
- DMV-01 Desarrollo del módulo de vocabulario de contexto.
- DMA-01 Desarrollo del módulo de análisis compartido.
- SCE-01Selección de un caso de estudio.
- CDC-01 Desarrollo del módulo de comparación de diccionario de contexto.
- PCE-01 Realizar pruebas con el caso de estudio seleccionado.

Que a su vez derivaron en la siguiente lista de pendientes del *sprint* (*Sprint Backlog*), junto a sus correspondientes tareas:

- MOD-01 Arquitectura.
    - Selección de tecnología a utilizar.
    - Instalación y configuración del entorno de desarrollo y herramientas necesarias.
    - Creación del modelo de base de datos.

Como parte de los resultados obtenidos durante el desarrollo de las tareas de este Sprint, se generó una primera versión de la estructura de base de datos para almacenar los datos obtenidos del dispositivo Leap Motion que se observan en la figura 3.5.

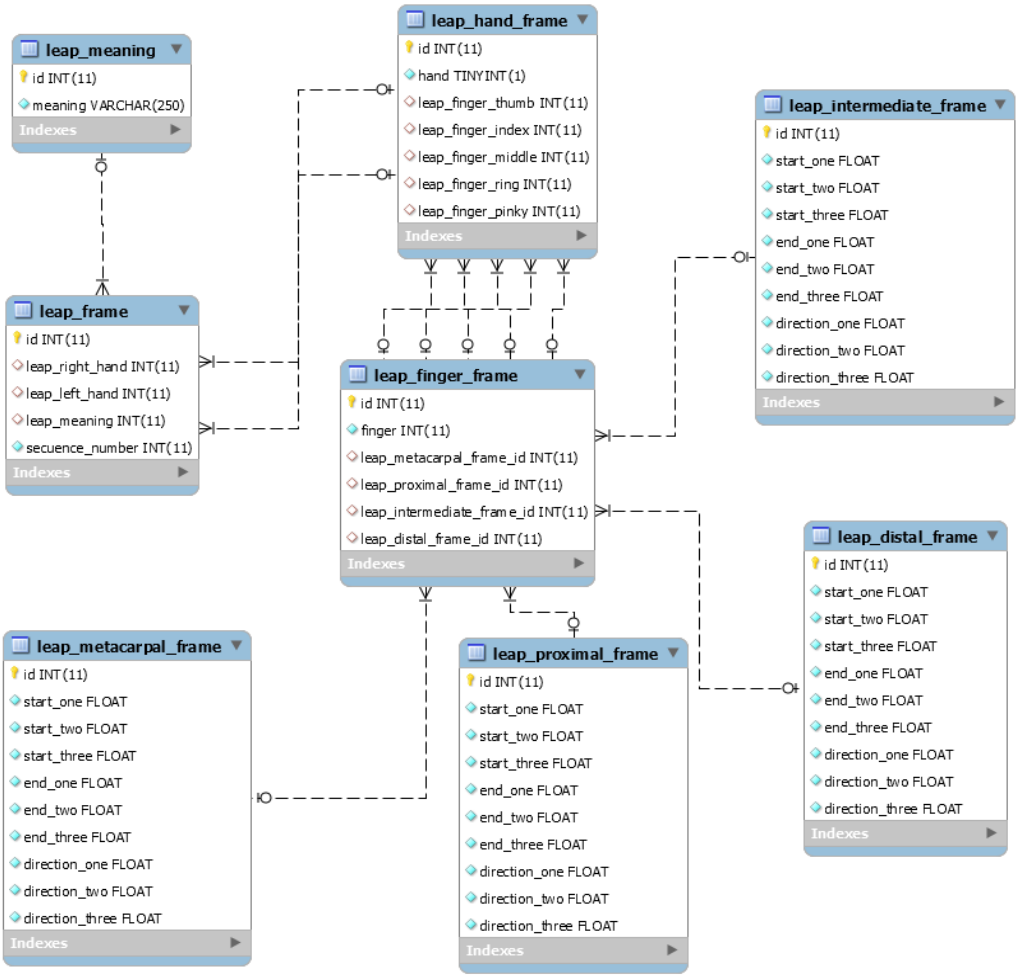



Figura 3.5 Modelo de base de datos utilizado para el almacenamiento de los datos obtenidos por el control Leap Motion correspondiente al Sprint 2 Febrero 2018.

El total de puntos de las tareas cerradas para este *sprint* fue de doce tal como se observa en la captura de pantalla de la interfaz de Taiga que se muestra en la figura 3.6.

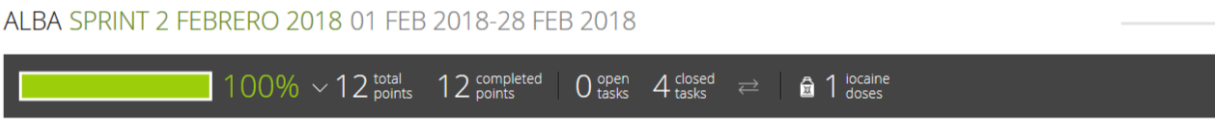



Figura 3.6 Interfaz de Taiga para manejo de tareas de *Sprint* 2 Febrero 2018.

### 3.2.6 *Sprint* 3 Marzo 2018

Para el tercer *Sprint* se contó con la siguiente lista de producto (*Product Backlog*) basada en las historias de usuario:

- DMC-01 Desarrollo del módulo de captura de información.
- DMI-01 Desarrollo del módulo de procesamiento de imágenes.
- DMI-02 Desarrollo del módulo de procesamiento de imágenes.
- DRD-01 Diseño del repositorio de datos de captura de movimiento.
- DMC-02 Desarrollo del módulo de comparación con repositorio.
- DMV-01 Desarrollo del módulo de vocabulario de contexto.
- DMA-01 Desarrollo del módulo de análisis compartido.
- SCE-01Selección de un caso de estudio.
- CDC-01 Desarrollo del módulo de comparación de diccionario de contexto.
- PCE-01 Realizar pruebas con el caso de estudio seleccionado.

Que a su vez derivaron en la siguiente lista de pendientes del *sprint* (*Sprint Backlog*), junto a sus correspondientes tareas:

- DMC-01 Desarrollo del módulo de captura de información.
    - Revisión de información disponible para utilizar el api de *LeapMotion* con Python 3.6.
    - Configuración de Visual Studio y capa de compatibilidad de Python.
    - Creación de adaptador para Python 3.6 en su versión de 64 bits.
    - Selección de características relevantes según el estado del arte.
    - Realizar pruebas de conexión con el dispositivo *LeapMotion*.
    - Revisión de marcos de trabajo de interfaces gráficas para Python.
    - Pruebas de integración de interfaces y *LeapMotion*.

Se utilizó la biblioteca Swig para generar un adaptador para que la biblioteca de LeapMotion se utilizara con la versión 3.6 de Python, ya que por defecto solo funciona con la versión 2, para ello se utilizó el IDE Visual Studio Community 2018 y se siguieron las instrucciones del blog de soporte de LeapMotion con título "*Generating a Python 3.3.0 Wrapper with SWIG 2.0.9*" [45].

Se llevaron a cabo pruebas para la integración de las interfaces gráficas *Kivy* [46], *PyForms* [47] y *Tkinter* [48], seleccionando finalmente *Tkinter* debido a su facilidad de uso, estabilidad y fácil integración en el proyecto, ya que es la biblioteca por defecto

integrada como parte de Python, además de que se encontraron algunos problemas para integrar los ciclos de refrescamiento de los hilos de *Kivy* y *PyForms*, por lo que sumado al tiempo requerido para poder corregir estos problemas se desechó su uso en esta versión del proyecto.

El total de puntos de las tareas cerradas para este *sprint* fue de treinta y seis tal como se observa en la captura de pantalla de la interfaz de Taiga que se muestra en la figura 3.7.

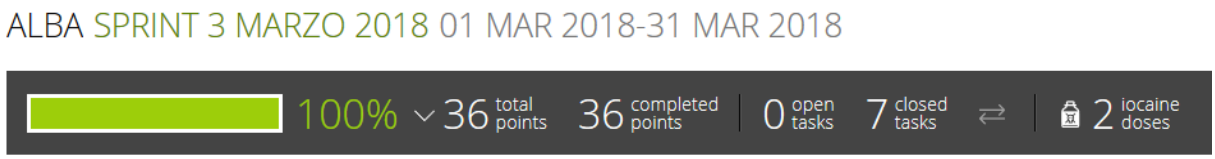



Figura 3.7 Interfaz de Taiga para manejo de tareas de *Sprint* 3 Marzo 2018.

### 3.2.7 *Sprint* 4 Abril 2018

Para el cuarto Sprint se contó con la siguiente lista de producto (*Product Backlog*) basada en las historias de usuario:

- DMC-01 Desarrollo del módulo de captura de información.
- DMI-01 Desarrollo del módulo de procesamiento de imágenes.
- DMI-02 Desarrollo del módulo de procesamiento de imágenes.
- DRD-01 Diseño del repositorio de datos de captura de movimiento.
- DMC-02 Desarrollo del módulo de comparación con repositorio.
- DMV-01 Desarrollo del módulo de vocabulario de contexto.
- DMA-01 Desarrollo del módulo de análisis compartido.
- SCE-01Selección de un caso de estudio.
- CDC-01 Desarrollo del módulo de comparación de diccionario de contexto.
- PCE-01 Realizar pruebas con el caso de estudio seleccionado.

Que a su vez derivaron en la siguiente lista de pendientes del *sprint* (*Sprint Backlog*), junto a sus correspondientes tareas:

- DMC-01 Desarrollo del módulo de captura de información.
  - Diseño de interfaz gráfica para capturar datos de *LeapMotion*.
  - Pruebas sobre el uso del manejador grid y pack en Tkinter.
  - Pruebas de acceso a la base de datos.

    - Creación de clases contenedoras para los datos capturados por el dispositivo *LeapMotion*
- DRD-01 Diseño del repositorio de datos de captura de movimiento.
    - Creación del archivo que usa SQLAlchemy para la creación de la base de datos.
    - Instalación y configuración del paquete para manejar la conexión a MySQL.
    - Realización de pruebas sobre la implementación de las relaciones entre las tablas.

Se realizaron los bocetos correspondientes a la interfaz básica de selección de opciones, entrenamiento e identificación de señas, tal como se observa en las figuras 3.8 y 3.9.

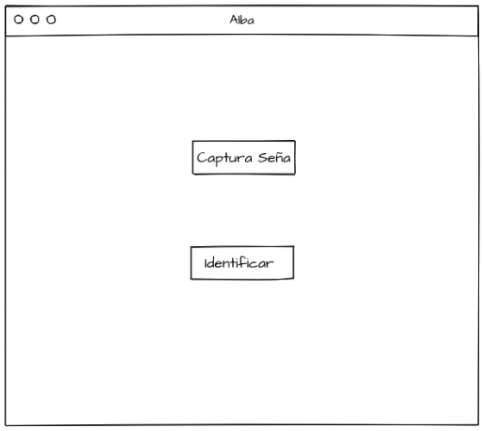



Figura 3.8 Boceto de interfaz de selección de opciones inicial.

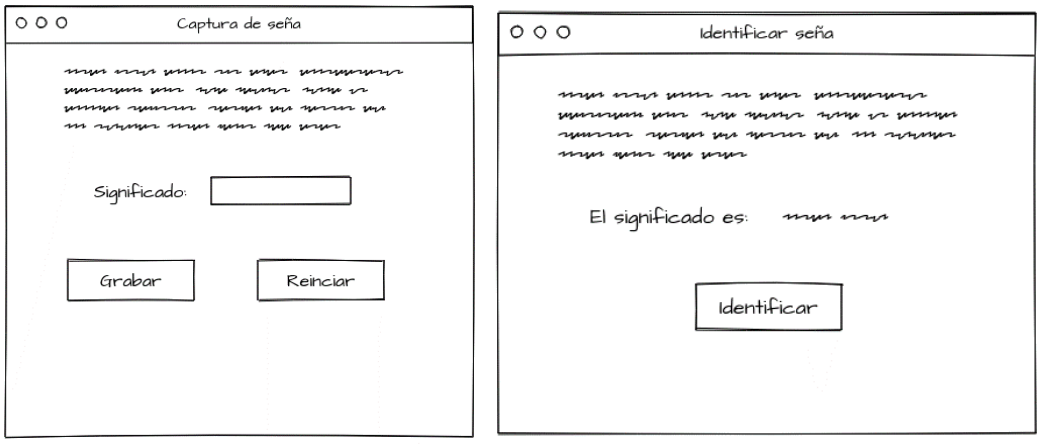



Figura 3.9 Boceto de interfaz de captura de seña e identificar seña.

Como parte del incremento alcanzado durante la ejecución de este *sprint*, se realizó una revisión de la estructura del repositorio de datos acorde a la historia de usuario DRD-01, realizando una serie de mejoras, al simplificar las entidades y hacerlas más genéricas, por lo cual la nueva estructura se muestra en la figura 3.10, como se observa la información correspondiente a los dedos de las manos se unificó en una sola tabla, ya que comparten la misma información, es decir, los dedos están compuestos de los mismos elementos (posiciones de los huesos en un espacio de coordenadas tridimensional x, y, z).

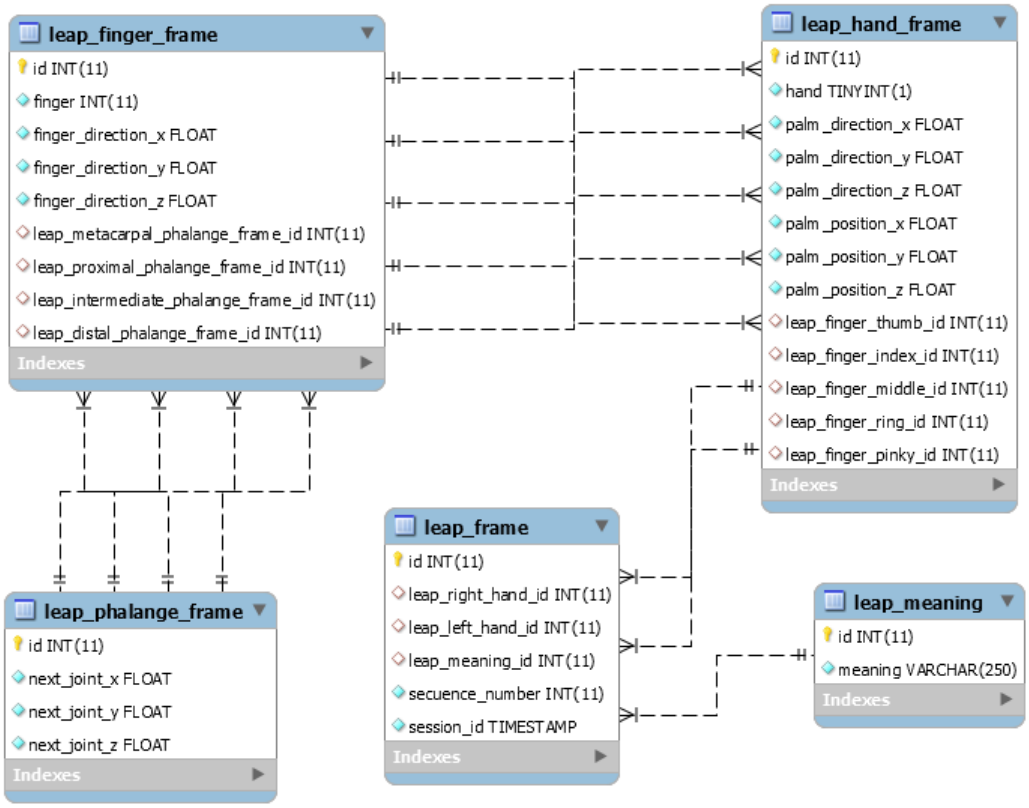



Figura 3.10 Modelo de base de datos utilizado para el almacenamiento de los datos obtenidos por el control Leap Motion correspondiente al *Sprint* 4 Abril 2018.

Se instaló el dialecto *PyMySQL* [49] en *SQLAlchemy* para poder conectarse a la base de datos y realizar las pruebas correspondientes entre el *ORM* y las clases generadas.

Finalmente se generó la clase base *LeapFrame* que utiliza *SQLAlchemy* para crear la base de datos y manejar las relaciones y objetos por medio de su *ORM*, cuyo extracto

de código se muestra a continuación, se observa la creación de clases en las líneas 9 y 23, su diagrama de clases se incluye en la figura 3.11.

```
1 from sqlalchemy import Column, Integer, String, Float, Boolean, TIMESTAMP, Table
2 from sqlalchemy.ext.declarative import declarative_base
3 from sqlalchemy import create_engine
4 from sqlalchemy.sql.schema import ForeignKey
5 from sqlalchemy.orm import relationship
6
7 Base = declarative_base()
8
9  class LeapFrame(Base):
10    __tablename__='leap_frame'
11
12    id = Column(Integer, primary_key = True)
13    leap_right_hand_id = Column(Integer, ForeignKey('leap_hand_frame.id'),
nullable = True)
14    leap_left_hand_id = Column(Integer, ForeignKey('leap_hand_frame.id'), nullable
= True)
15    leap_meaning_id = Column(Integer, ForeignKey('leap_meaning.id'))
16    secuence_number = Column(Integer, nullable = False)
17    session_id = Column(TIMESTAMP, nullable=False)
18
19    right_hand = relationship("LeapHandFrame", foreign_keys=[leap_right_hand_id])
20    left_hand = relationship("LeapHandFrame", foreign_keys=[leap_left_hand_id])
21    meaning = relationship("LeapMeaning", foreign_keys=[leap_meaning_id])
22
23 class LeapHandFrame(Base):
24
25    __tablename__ = 'leap_hand_frame'
26
27    id = Column(Integer, primary_key=True)
28
29    hand = Column(Boolean, nullable = False)
30
31    palm_direction_x = Column(Float, nullable=False)
32    palm_direction_y = Column(Float, nullable=False)
33    palm_direction_z = Column(Float, nullable=False)
34    # La posicion de la mano con respecto al leapMotion
35    palm_position_x = Column(Float, nullable=False)
36    palm_position_y = Column(Float, nullable=False)
37    palm_position_z = Column(Float, nullable=False)
38    leap_finger_thumb_id = Column(Integer, ForeignKey('leap_finger_frame.id'))
39    leap_finger_index_id = Column(Integer, ForeignKey('leap_finger_frame.id'))
40    leap_finger_middle_id = Column(Integer, ForeignKey('leap_finger_frame.id'))
41    leap_finger_ring_id = Column(Integer, ForeignKey('leap_finger_frame.id'))
42    leap_finger_pinky_id = Column(Integer, ForeignKey('leap_finger_frame.id'))
43
44    thumb = relationship("LeapFingerFrame", foreign_keys=[leap_finger_thumb_id])
45    index = relationship("LeapFingerFrame", foreign_keys=[leap_finger_index_id])
46    middle = relationship("LeapFingerFrame", foreign_keys=[leap_finger_middle_id])
47    ring = relationship("LeapFingerFrame", foreign_keys=[leap_finger_ring_id])
48    pinky = relationship("LeapFingerFrame", foreign_keys=[leap_finger_pinky_id])
```

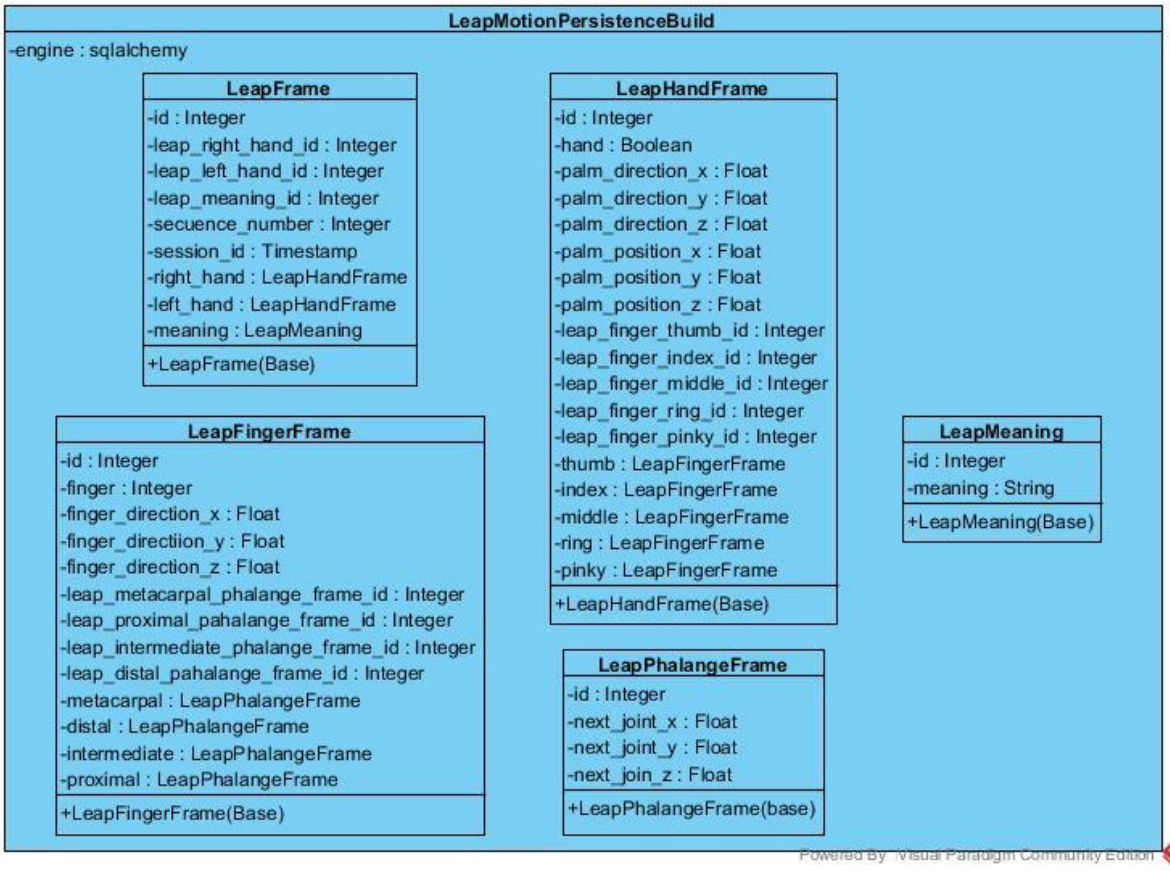



Figura 3.11 Diagrama de la clase LeapMotionPersistenceBuild que a su vez contiene diversas clases que se utilizan con el *ORM* de *SQLAlchemy* para generar la base de datos y manejar las relaciones.

El total de puntos de las tareas cerradas para este *sprint* fue de cuarenta y uno tal como se observa en la captura de pantalla de la interfaz de Taiga que se muestra en la figura 3.12.

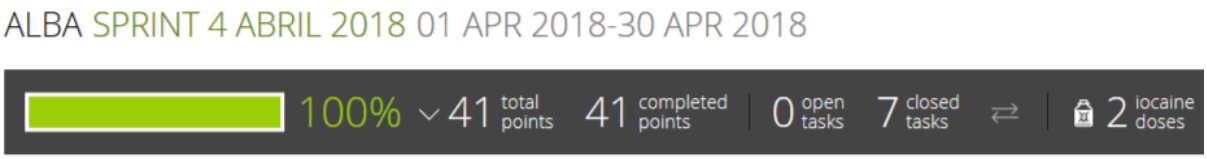



Figura 3.12 Interfaz de Taiga para manejo de tareas de *Sprint* 4 Abril 2018.

### 3.2.8 *Sprint* 5 Mayo 2018

Para el quinto *Sprint* se contó con la siguiente lista de producto (*Product Backlog*) basada en las historias de usuario:

- DMI-02 Desarrollo del módulo de procesamiento de imágenes.
- DRD-01 Diseño del repositorio de datos de captura de movimiento.
- DMC-02 Desarrollo del módulo de comparación con repositorio.
- DMV-01 Desarrollo del módulo de vocabulario de contexto.
- DMA-01 Desarrollo del módulo de análisis compartido.
- SCE-01Selección de un caso de estudio.
- CDC-01 Desarrollo del módulo de comparación de diccionario de contexto.
- PCE-01 Realizar pruebas con el caso de estudio seleccionado.

Que a su vez derivaron en la siguiente lista de pendientes del *sprint* (*Sprint Backlog*), junto a sus correspondientes tareas:

- DMI-01 Desarrollo del módulo de procesamiento de imágenes.
    - Instalación de la biblioteca OpenCV.
    - Configuración de las variables de entorno de OpenCV.
    - Análisis del comportamiento del hilo de ejecución de OpenCV.
    - Análisis de las técnicas y dispositivos disponibles para la captura y transformación de imágenes.
    - Creación de clases auxiliares para ejecutar diversos métodos de control de la interfaz.
    - Pruebas de funcionalidad de refrescamiento con el canvas de Tkinter.
    - Pruebas de integración de la biblioteca Scikit-learn.

Para construir el módulo de procesamiento de imágenes se realizó un análisis de las técnicas y dispositivos disponibles para la captura y transformación de imágenes, incluyendo la opción de acceder a los datos obtenidos por las cámaras infrarrojas del dispositivo *LeapMotion* [50], sin embargo, se descartó utilizar esta opción, ya que como se muestra en la figura 3.13 la vista obtenida estaría limitada a la posición del *LeapMotion* y lo que se busca con la captura de imágenes es ampliar el ángulo de visión y por lo tanto ser capaces de superar los puntos ciegos que tiene el dispositivo actualmente.

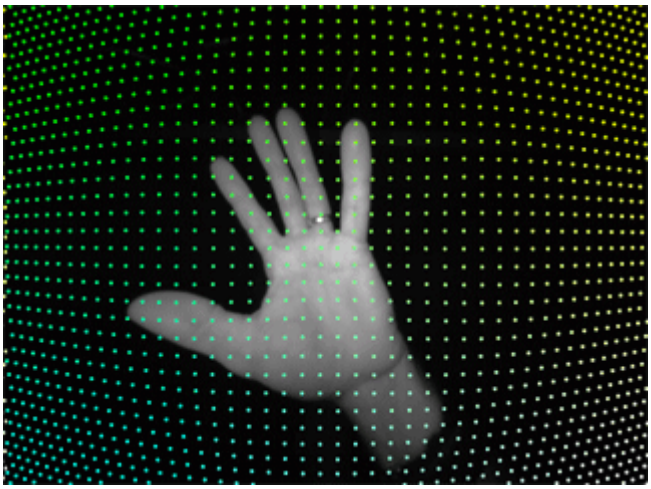


Figura 3.13 Ejemplo de imagen obtenida por el dispositivo *LeapMotion* donde se aprecia la distorsión generada por el lente instalado.

Como se observa en la figura anterior, las imágenes obtenidas desde las cámaras infrarrojas del dispositivo *LeapMotion*, presentan una distorsión en las áreas alejadas del centro, esto se debe a las especificaciones del diseño, ya que gracias a esta distorsión se obtiene un mayor ángulo de visión, lo que permite mantener el seguimiento de las manos aun cuando se alejan del dispositivo, utilizar estas imágenes para su análisis conlleva a una constante corrección de imagen en tiempo real, lo que a su vez impacta negativamente en la velocidad de respuesta y el rendimiento de la aplicación en general aun en el caso de utilizar funciones de procesamiento de *OpenCV* como *convertMaps()*, tal como se muestra en el siguiente extracto de código en la línea 26, que facilita la documentación de *LeapMotion* para Python [50] :

```
1  Import cv2, Leap, math, ctypes
2  import numpy as np
3
4  def convert_distortion_maps(image):
5
6      distortion_length = image.distortion_width * image.distortion_height
7      xmap = np.zeros(distortion_length/2, dtype=np.float32)
8      ymap = np.zeros(distortion_length/2, dtype=np.float32)
9
10     for i in range(0, distortion_length, 2):
11         xmap[distortion_length/2 - i/2 - 1] = image.distortion[i] * image.width
12         ymap[distortion_length/2 - i/2 - 1] = image.distortion[i + 1] *
image.height
13
14     xmap = np.reshape(xmap, (image.distortion_height, image.distortion_width/2))
15     ymap = np.reshape(ymap, (image.distortion_height, image.distortion_width/2))
16
17     resized_xmap = cv2.resize(xmap,
18                               (image.width, image.height),
19                               0, 0,
20                               cv2.INTER_LINEAR)
```

```
21      resized_ymap = cv2.resize(ymap,
22                                (image.width, image.height),
23                                0, 0,
24                                cv2.INTER_LINEAR)
25
26      coordinate_map, interpolation_coefficients = cv2.convertMaps(resized_xmap,
27                                                                   resized_ymap,
28                                                                   cv2.CV_32FC1,
29                                                                   nninterpolation =
False)
30
31      return coordinate_map, interpolation_coefficients
```

Por otra parte, las cámaras instaladas son infrarrojas, lo que, si bien permite capturar e identificar las partes del cuerpo en distintas condiciones de luz, resulta en una reducción de las técnicas de tratamiento de imágenes disponibles, ya que no se puede trabajar con los algoritmos destinados a identificar colores.

Derivado de los puntos anteriores, después de realizar una serie de pruebas, se optó por descartar el uso de las cámaras infrarrojas instaladas en el dispositivo y utilizar en su lugar una cámara secundaria.

Se realizó el proceso de análisis y selección de una cámara Web con conexión *USB* que pudiera utilizarse en el proyecto, obteniendo una buena imagen en tiempo real y cuyo costo fuera reducido, por ello se seleccionó la cámara Logitech C525, que cuenta con una resolución de 8 megapíxeles para fotografías, videos en 720p / 30fs y autoenfoque [51], siendo esta última característica determinante, ya que para poder obtener imágenes nítidas, deben estar correctamente enfocadas y considerando que el usuario final podrá variar constantemente y no se tendrá un control absoluto de su ubicación, el autoenfoque permite aumentar las posibilidades de obtener una imagen de alta calidad, en la figura 3.14 se observa la cámara secundaria seleccionada, cabe destacar que el diseño de los algoritmos de análisis de imagen se realizaron de manera genérica, es decir, podrán funcionar con cualquier cámara.


Figura 3.14 Cámara web secundaria Logitech C525

Para el manejo de la interfaz gráfica se desarrolló la clase principal del programa llamada Ventana, que usa diversos *widgets* de la biblioteca Tkinter para permitir al usuario interactuar con el programa, el enfoque inicial es brindarle al usuario dos opciones a elegir, la primera llamada “Capturar seña” referente a la interfaz utilizada para capturar los datos de posición en 3D e imágenes, y la segunda llamada "Identificar seña”, que sería utilizada para clasificar la seña realizada por el usuario, en la figura 3.15 se observa el diagrama de esta clase para esta iteración y en la figura 3.16 y 3.17 se observan las ventanas generadas.

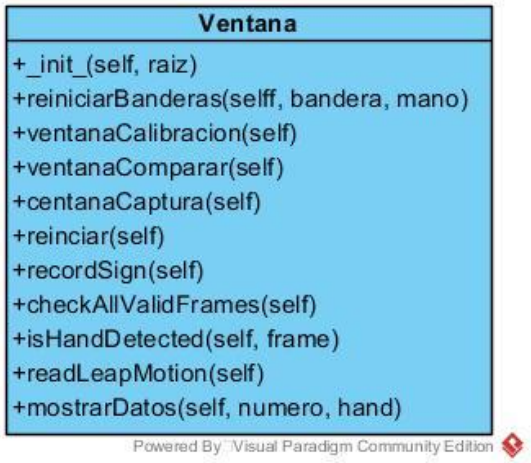



Figura 3.15 Diagrama de la clase Ventana.

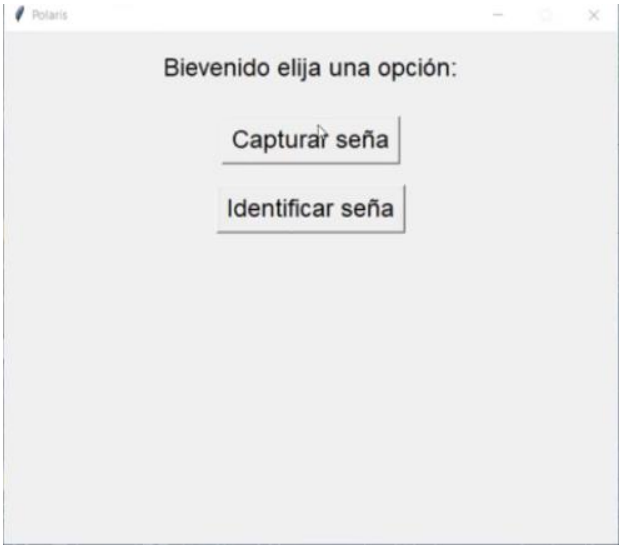



Figura 3.16 Ventana principal generada durante el *sprint* 5.

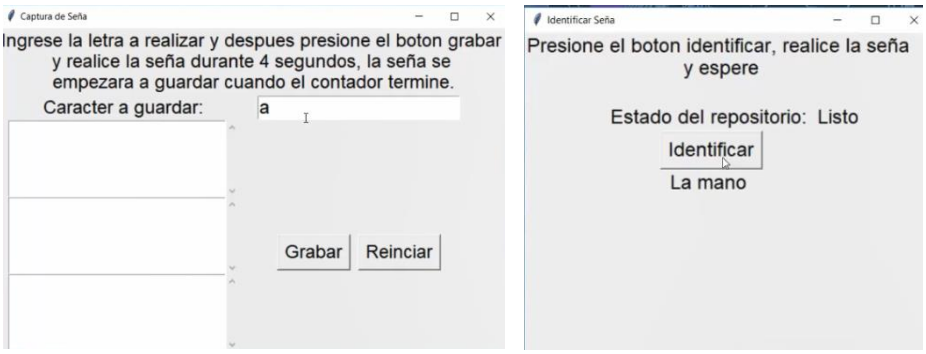



Figura 3.17 Ventanas secundarias para recolectar y predecir información.

El enfoque que se tomó para la interfaz en esta iteración, fue probar la correcta integración de la biblioteca Tkinter y los dispositivos de captura.

El total de puntos de las tareas cerradas para este *sprint* fue de cuarenta tal como se observa en la captura de pantalla de la interfaz de Taiga que se muestra en la figura 3.18.


Figura 3.18 Interfaz de Taiga para manejo de tareas de *Sprint* 5 Mayo 2018.

### 3.2.9 *Sprint* 6 Junio 2018

Para el sexto *Sprint* se contó con la siguiente lista de producto (*Product Backlog*) basada en las historias de usuario:

- DMI-02 Desarrollo del módulo de procesamiento de imágenes.
- DRD-01 Diseño del repositorio de datos de captura de movimiento.
- DMC-02 Desarrollo del módulo de comparación con repositorio.
- DMV-01 Desarrollo del módulo de vocabulario de contexto.
- DMA-01 Desarrollo del módulo de análisis compartido.
- SCE-01Selección de un caso de estudio.
- CDC-01 Desarrollo del módulo de comparación de diccionario de contexto.
- PCE-01 Realizar pruebas con el caso de estudio seleccionado.

Que a su vez derivaron en la siguiente lista de pendientes del *sprint* (*Sprint Backlog*), junto a sus correspondientes tareas:

- DMI-02 Desarrollo del módulo de procesamiento de imágenes.
  - Realizar pruebas con el clasificador de Haar en cascada en OpenCV.
  - Algoritmo de etiquetado de componentes conectados.
  - Selección adaptativa.
  - Segmentación OTSU.

La detección de objetos por medio del uso de clasificadores de características de Haar es un método sumamente efectivo propuesto por Paul Viola and Michael Jones, donde una función en cascada es entrenada con un alto volumen de imágenes positivas y negativas, en el caso de OpenCV, la biblioteca cuenta con un detector y un entrenador para estas características, que utilizan y generan respectivamente un archivo *XML* que contiene las funciones en cascada necesarias para la detección de los objetos.

Al ser un método que puede ser entrenado, se utiliza para detectar no solo partes del cuerpo, si cualquier objeto específico, siempre y cuando se cuente con la suficiente cantidad de imágenes positivas (imágenes del objeto) e imágenes negativas (imágenes sin el objeto).

Por otra parte, están disponibles un gran número de archivos *XML* clasificadores para el rostro y las manos.

Una vez detectado el objeto buscado en la imagen (cara, ojos o manos), se procede a generar una región de interés en forma cuadrangular o rectangular, según sea el caso, para facilitar a su vez un análisis más detallado.

Tomando en cuenta lo anterior, se utilizó la función *CascadeClassifier* del módulo *cv2* de la biblioteca OpenCV, así como los archivos *XML* clasificadores incluidos en la misma en la ruta opencv/data/haarcascades/, sin embargo, la Lengua de Señas Mexicana incluye distintas posiciones de las manos, desde las palmas a los dedos, por lo que un entrenamiento detallado sería requerido, para este *sprint* solo se decidió probar la facilidad de implementación de esta funcionalidad, a continuación se incluye el código utilizado para probar la funcionalidad de detección de manos, palmas y rostro, con la clase *face_cascade* en la línea 14:

```
1  import cv2
2
3  face_cascade = cv2.CascadeClassifier('../varios/haarcascade_frontalface_default.xml')
4  hand_cascade = cv2.CascadeClassifier('../varios/haarcascade_hand.xml')
5  hand_cascade_second = cv2.CascadeClassifier('../varios/haarcascade_hand_second.xml'
6  open_hand_cascade = cv2.CascadeClassifier('../varios/haarcascade_openhand.xml')
7
8  cap = cv2.VideoCapture(0)
9
```

```
10 while True:
11     ret, img = cap.read()
12     # print(ret)
13     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
14     faces = face_cascade.detectMultiScale(gray,1.3,5)
15     xo,yo,wo,ho = 0,0,0,0
16     for(x,y,w,h) in faces:
17         xo=x
18         yo=y
19         wo=w
20         ho=h
21 
22         cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
23         roi_gray=gray[y:y+h,x:x+w]
24         roi_color = img[y:y+h, x:x+w]
25 
26     cv2.rectangle(gray, (xo, yo), (xo + wo, yo + ho), (255, 0, 0), 2)
27     # cv2.imshow('img', gray)
28     cv2.imshow('img',img)
29     k=cv2.waitKey(30) & 0xff
30     if k == 27:
31         break
32 
33  cap.release()
34  cv2.destroyAllWindows()
```

La detección de objetos funciona ejecutando el método *detectMultiScale()* aplicado a un objeto *CascadeClassifier*, generado al cargar el correspondiente archivo *XML* como se observa en la variable *face_cascade*, cabe destacar que esta función requiere una imagen convertida a escala de grises, además de los parámetros de factor de escala y mínimo de vecinos, que se refieren a cuánto será reducida la imagen en cada iteración y cuántos vecinos debe tener cada posible objeto para que sea mantenido respectivamente.

Se observa en la figura 3.19 el resultado de ejecutar el clasificador para rostro y en la figura 3.20 al ejecutarlo con el clasificador de manos.

El enfoque buscado con los clasificadores *Haar* es determinar un área de interés, es decir, cuando se ubica un objeto en específico, es posible determinar un *ROI* (Región of Interes, Región de Interés), que a su vez se extrae y analiza mediante otras técnicas, esto es altamente útil, ya que se reduce el área a analizar y como consecuencia se reduce la complejidad y la demanda de recursos, brindando una mejora en el tiempo de respuesta del programa, lo cual es relevante para un sistema en tiempo real como el propuesto.

El algoritmo de etiquetado de componentes conectados es otro enfoque utilizado en la resolución de problemas de visión por computadora, se usa para detectar regiones en imágenes binarias, aunque es factible utilizar imágenes de color, se prefieren las imágenes binarias para lograr mejores resultados, una vez identificadas las regiones estas pueden ser contadas, filtradas o seguidas, lo cual permite aplicarlo a una amplia

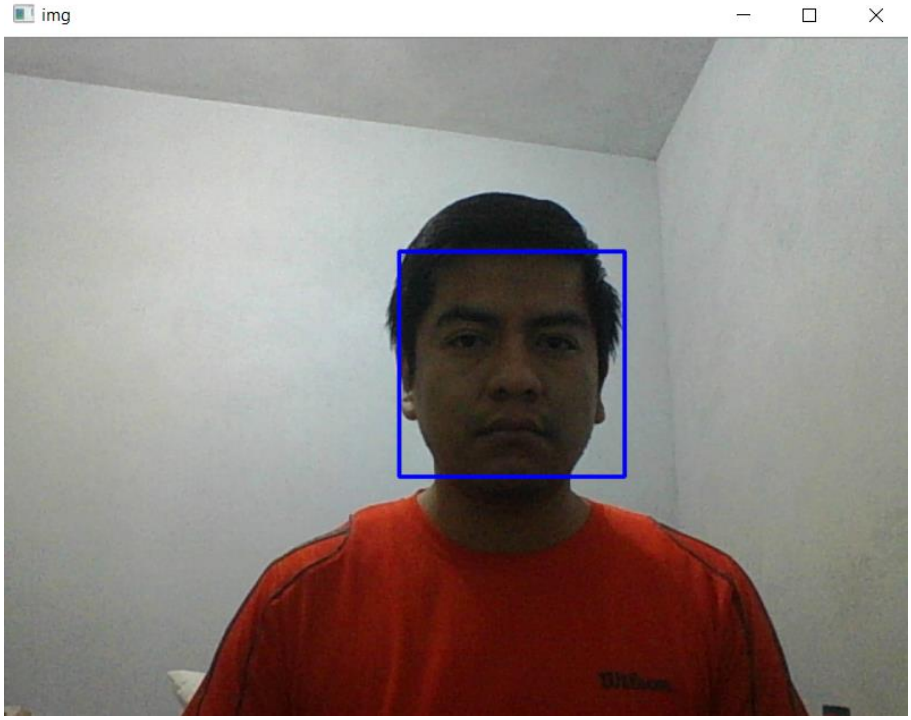



Figura 3.19 Ventana resultante al ejecutar el clasificador de rostros.


Figura 3.20 Ventana resultante al ejecutar el clasificador de manos.

gama de funcionalidades, en el caso del intérprete de lengua de señas, permitiría una vez detectada la mano en un área de interés, depurar la imagen resultante, eliminando objetos o áreas que no pertenecen a la mano pero que fueron detectados en un procesamiento previo, para lograrlo, es posible utilizar la conectividad de cuatro u ocho vecinos, es decir, un pixel solo será integrante de una región siempre y cuando tenga cuatro u ocho vecinos que también sean pixeles válidos, de esta forma se pueden eliminar pixeles no pertenecientes a la región principal, estos pixeles también son llamados ruido, a continuación se muestra el código de una implementación básica del etiquetado de componentes conectados en Python obtenida de la documentación de la biblioteca scikit-image [52] para la función *find_contours* en la línea 10 y en la figura 3.21 se observa la imagen etiquetada resultante:

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  from skimage import measure
4
5
6  x, y = np.ogrid[-np.pi:np.pi:100j, -np.pi:np.pi:100j]
7  r = np.sin(np.exp((np.sin(x)**3 + np.cos(y)**2)))
8
9  # Find contours at a constant value of 0.8
10 contours = measure.find_contours(r, 0.8)
11
12 fig, ax = plt.subplots()
13 ax.imshow(r, interpolation='nearest', cmap=plt.cm.gray)
14
15 for n, contour in enumerate(contours):
16    ax.plot(contour[:, 1], contour[:, 0], linewidth=2)
17
18 ax.axis('image')
19 ax.set_xticks([])
20 ax.set_yticks([])
21 plt.show()
```

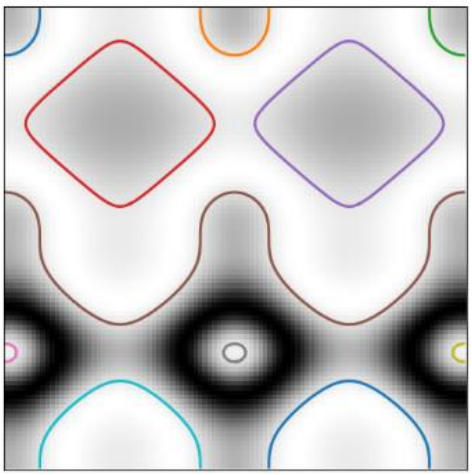


Figura 3.21 Regiones etiquetadas con la biblioteca scikit-image y el algoritmo de etiquetado de componentes conectados.

La selección adaptativa es otra técnica utilizada en el tratamiento de imágenes, que se basa en la manipulación de la información obtenida en el histograma, el cual se representa como un gráfico de frecuencias relativas de la ocurrencia de cada nivel de intensidad en la imagen, su aplicación en el intérprete de lengua de señas permite tener una mejor calidad de la imagen analizada, eliminando distorsiones en la misma y permitiendo una mejor selección de características relevantes, la representación matemática del histograma de una imagen con niveles de gris en el rango [0,L-1], donde k es el k-ésimo nivel de gris, k=0,1,...,L−1, con nk píxeles, se expresa de la siguiente forma:

$$h(k) = \delta(f(x,y) - k)_{x,y}\sum, \forall k = 0,1,\ldots,L-1$$

Para mejorar la aplicación de esta técnica, se procesa la imagen por subregiones o regiones contextuales, esta técnica suele aplicarse a imágenes en escala de grises, sin embargo, puede aplicarse a imágenes en color, pero convertidas al espacio de color HSL y ejecutado solo en el canal L, relacionado con la luminosidad, se procedió a probar el comportamiento de la función *adaptiveThreshold* de OpenCV, a continuación se muestra el código utilizado para la prueba de la función en la línea 9 y 10, obtenido de la documentación oficial de OpenCV[53] y en la figura 3.22 se muestra la imagen resultante:

```
1  import cv2 as cv
2
3  from matplotlib import pyplot as plt
4
5  img = cv.imread('../1542983849.5482736_normalize.jpg',0)
6  img = cv.medianBlur(img,5)
7
8  ret,th1 = cv.threshold(img,127,255,cv.THRESH_BINARY)
9  th2 = cv.adaptiveThreshold(img,255,cv.ADAPTIVE_THRESH_MEAN_C, cv.THRESH_BINARY,11,2)
10 th3 = cv.adaptiveThreshold(img,255,cv.ADAPTIVE_THRESH_GAUSSIAN_C,
cv.THRESH_BINARY,11,2)
11
12 titles = ['Imagen original', 'Selección Global (v = 127)',
13            'Selección adaptativa promedio', 'Selección adaptativa gaussiana']
14 images = [img, th1, th2, th3]
15
16 for i in range(4):
17    plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')
18    plt.title(titles[i])
19    plt.xticks([]),plt.yticks([])
20
21 plt.show()
```

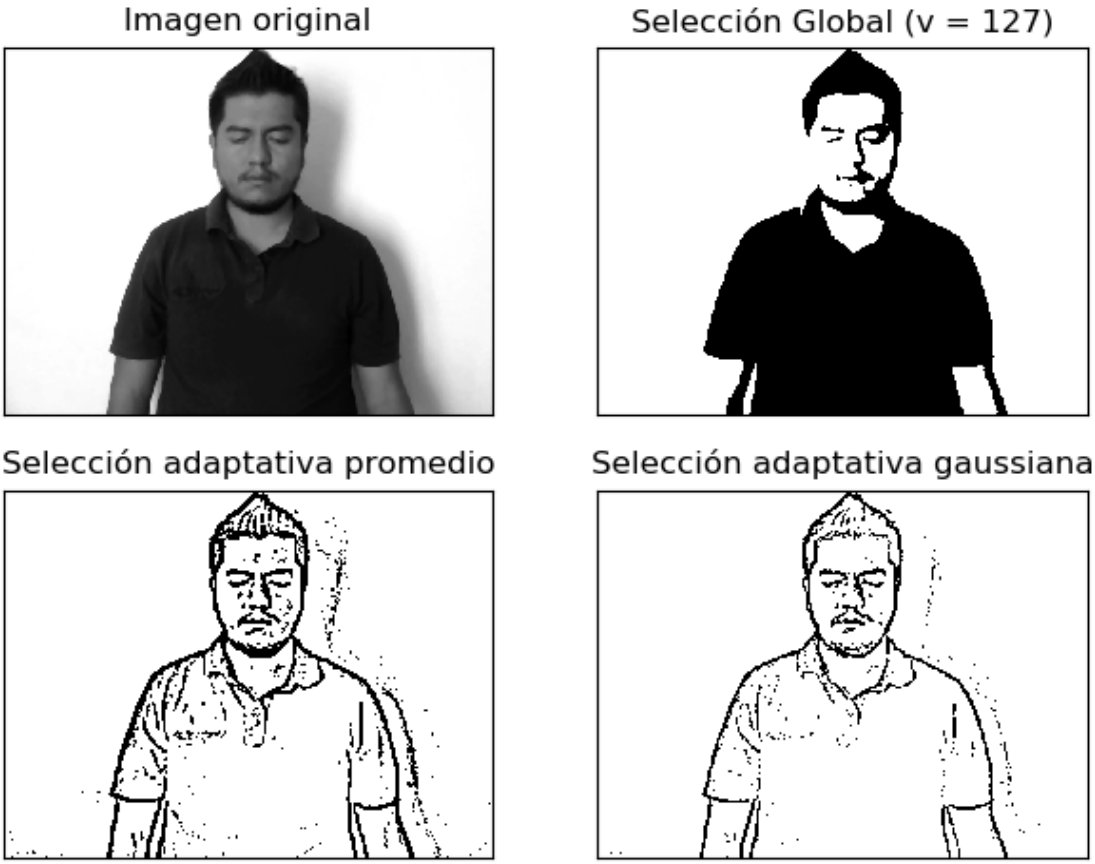



Figura 3.22 Resultado de la aplicación de la selección adaptativa con diferentes valores de umbral.

La selección de características utilizando el valor de umbral global se apoya en el algoritmo *OTSU*, especializado en imágenes bimodales, es decir, aquellas imágenes que en su histograma tienen dos picos, estos picos representan los objetos al frente y al fondo de la imagen, por lo que para esa imagen se pueden tomar los valores en el área entre los picos como un valor de umbral, *OTSU* calcula de manera automática el valor de umbral del histograma de la imagen, sin embargo, solo brindará buenos resultados cuando la imagen sea bimodal, es por ello que para la aplicación de este método en el intérprete de lengua de señas, debe asegurase que en la imagen la persona tenga un contraste diferenciado con el fondo para eliminar correctamente esos elementos durante el tratamiento de la imagen y de esta forma obtener una imagen con características relevantes para su análisis, a continuación se muestra el código utilizado para realizar la prueba del algoritmo, usando la función *threshold* en la línea 12 tomado de la documentación de OpenCV[53] y en la figura 3.23 se observa el resultado:

```
import cv2 as cv

from matplotlib import pyplot as plt

img = cv.imread('../1542983849.5482736_normalize.jpg',0)
# Umbral global
ret1,th1 = cv.threshold(img,127,255,cv.THRESH_BINARY)
# Umbral Otsu
ret2,th2 = cv.threshold(img,0,255,cv.THRESH_BINARY+cv.THRESH_OTSU)
# Umbral Otsu's despues del filtro Gaussiano
blur = cv.GaussianBlur(img,(5,5),0)
ret3,th3 = cv.threshold(blur,0,255,cv.THRESH_BINARY+cv.THRESH_OTSU)

images = [img, 0, th1,
          img, 0, th2,
          blur, 0, th3]

titles = ['Imagen original','Histograma','Umbral global (v=127)',
          'Imagen original','Histograma',"Umbral Otsu",
          'Imagen con filtro gaussiano','Histograma',"Umbral Otsu"]

for i in range(3):
    plt.subplot(3,3,i*3+1),plt.imshow(images[i*3],'gray')
    plt.title(titles[i*3]), plt.xticks([]), plt.yticks([])
    plt.subplot(3,3,i*3+2),plt.hist(images[i*3].ravel(),256)
    plt.title(titles[i*3+1]), plt.xticks([]), plt.yticks([])
    plt.subplot(3,3,i*3+3),plt.imshow(images[i*3+2],'gray')
    plt.title(titles[i*3+2]), plt.xticks([]), plt.yticks([])

plt.show()
```

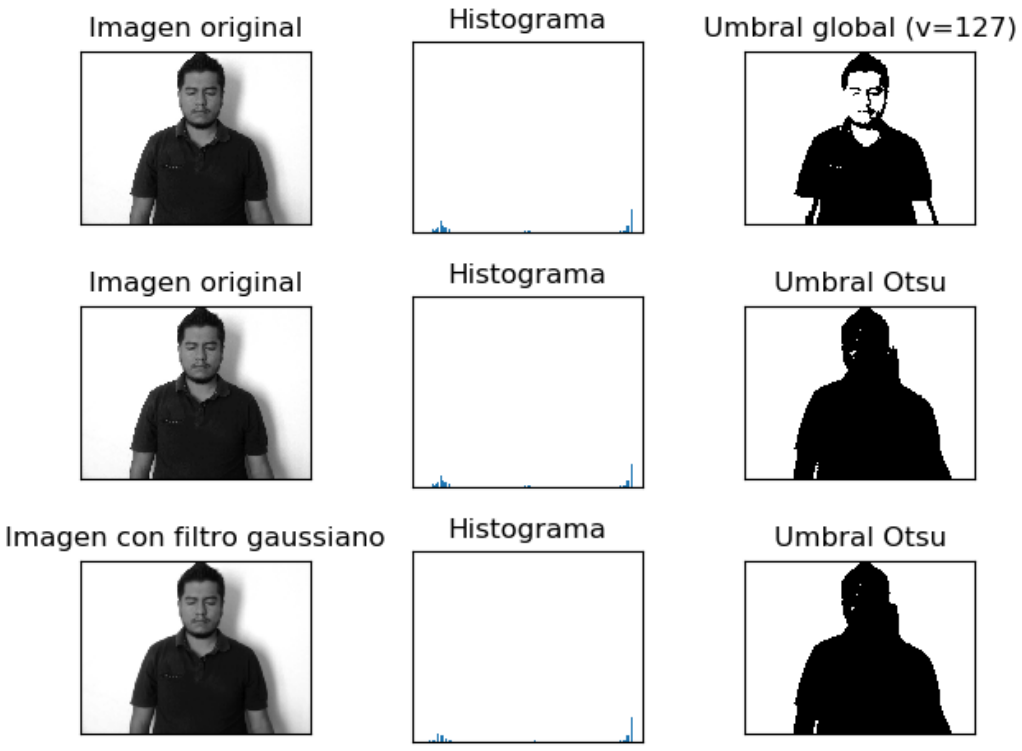



Figura 3.23 Resultado de la aplicación de la selección adaptativa con el algoritmo OTSU y filtro gaussiano.

El total de puntos de las tareas cerradas para este *sprint* fue de cuarenta y ocho tal como se observa en la captura de pantalla de la interfaz de Taiga que se muestra en la figura 3.24.

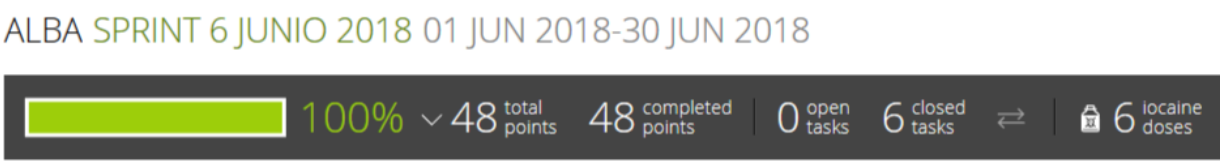



Figura 3.24 Interfaz de Taiga para manejo de tareas de *Sprint* 6 Junio 2018.

### 3.2.10 *Sprint* 7 Julio 2018

Para el séptimo *Sprint* se hizo una revisión de la información recolectada y se realizaron ajustes en la estructura del proyecto, trabajando en un rediseño de la interfaz, la base de datos y las clases necesarias para la interacción del usuario con el sistema, por lo que se agregaron las siguientes historias de usuario:

Tabla 3.15 Historia de usuario RDI-01: Rediseño de interfaz de usuario.

| Código | RDI-01 | |
|---|---|---|
| Descripción | Realizar un rediseño de la interfaz del usuario, para integrar todos los componentes y vistas necesarias. | |
| Tipo | Obligatoria | |
| Restricciones | | Relaciones |
| Deben definirse todas las ventanas a utilizar | | Ninguna |

Tabla 3.16 Historia de usuario RBD-01: Rediseño de la base de datos.

| Código | RBD-01 | |
|---|---|---|
| Descripción | Realizar un rediseño de la base de datos | |
| Tipo | Obligatoria | |
| Restricciones | | Relaciones |
| Deben definirse todos atributos de las clases que utilizará el ORM | | Ninguna |

Tabla 3.17 Historia de usuario MEC-01: Desarrollo del módulo de entrenamiento de comandos con gestos visuales.

| Código | MEC-01 | |
|---|---|---|
| Descripción | Creación de módulo de entrenamiento para comandos activados por gestos visuales para el control de la interfaz. | |
| Tipo | Obligatoria | |
| Restricciones | | Relaciones |
| Deben estar definidas completamente los controles que pueden aplicar a la interfaz. | | Ninguna |

Tabla 3.18 Historia de usuario MCG-01: Desarrollo del módulo de interpretación de gestos visuales.

| Código | MCG-01 | |
|---|---|---|
| Descripción | Creación de módulo de interpretación de gestos visuales. | |
| Tipo | Obligatoria | |
| Restricciones | | Relaciones |
| Debe estar complementado la historia de usuario MEC-01 | | Ninguna |

De igual manera la historia de usuario DMC-02 (Desarrollo del módulo de comparación con repositorio) absorbió a la historia de usuario CDC-01 (Desarrollo del módulo de comparación de diccionario de contexto), ya que se buscó simplificar y unificar el procesamiento de datos y la historia de usuario RBD-01 (Rediseño de base de datos) absorbió la historia de usuario DRD-01 (Diseño del repositorio de datos de captura de movimiento).

El esquema de la arquitectura reflejando los ajustes se observa en la figura 3.25.

La lista de producto (*Product Backlog*) basada en las historias de usuario quedó conformada de la siguiente manera:

- RDI-01 Rediseño de interfaz de usuario.
- RBD-01 Rediseño de base de datos.
- MEC-01 Desarrollo del módulo de entrenamiento de comandos con gestos visuales.
- MCG-01Desarrollo del módulo de interpretación de gestos visuales.
- DMI-02 Desarrollo del módulo de procesamiento de imágenes.
- DMC-02 Desarrollo del módulo de comparación con repositorio.

- DMV-01 Desarrollo del módulo de vocabulario de contexto.
- DMA-01 Desarrollo del módulo de análisis compartido.
- SCE-01Selección de un caso de estudio.
- PCE-01 Realizar pruebas con el caso de estudio seleccionado.

Por lo que los nuevos puntos totales del proyecto quedaron definidos en 398 con 181 cerrados al día 31 de Junio con un 45% de avance como se observa en la interfaz de Taiga para la gráfica de pendientes de producto que se muestra en la figura 3.26.

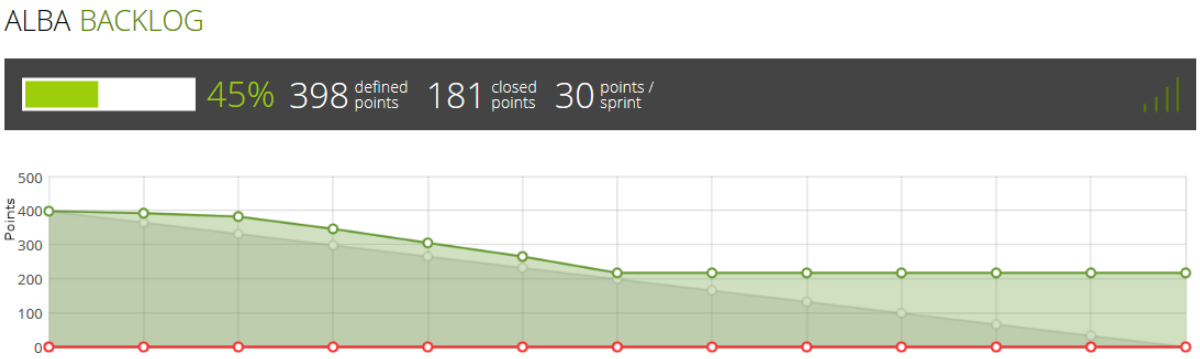



Figura 3.25 Gráfica de pendientes de producto al inicio del *Sprint* 6, el día 2 de Julio, reflejando los ajustes realizados.

La lista de pendientes del *sprint* actual (*Sprint Backlog*), junto a sus correspondientes tareas comprendió los siguientes puntos:

- RDI-01 Rediseño de interfaz de usuario.
  - Creación de bocetos para todas las pantallas y opciones disponibles.
- RBD-01 Rediseño de base de datos.
  - Revisión de las tablas necesarias para almacenar la información necesaria del sistema.
  - Modificación del archivo de clases de *SQLAlchemy* para integrar los nuevos atributos y clases.
- MEC-01 Desarrollo del módulo de entrenamiento de comandos con gestos visuales.
  - Creación de las clases necesarias para manejar el entrenamiento de comandos.
  - Realizar entrenamiento para reconocimiento de comandos.
  - Modificación del archivo de clases de *SQLAlchemy* para integrar los nuevos atributos y clases.

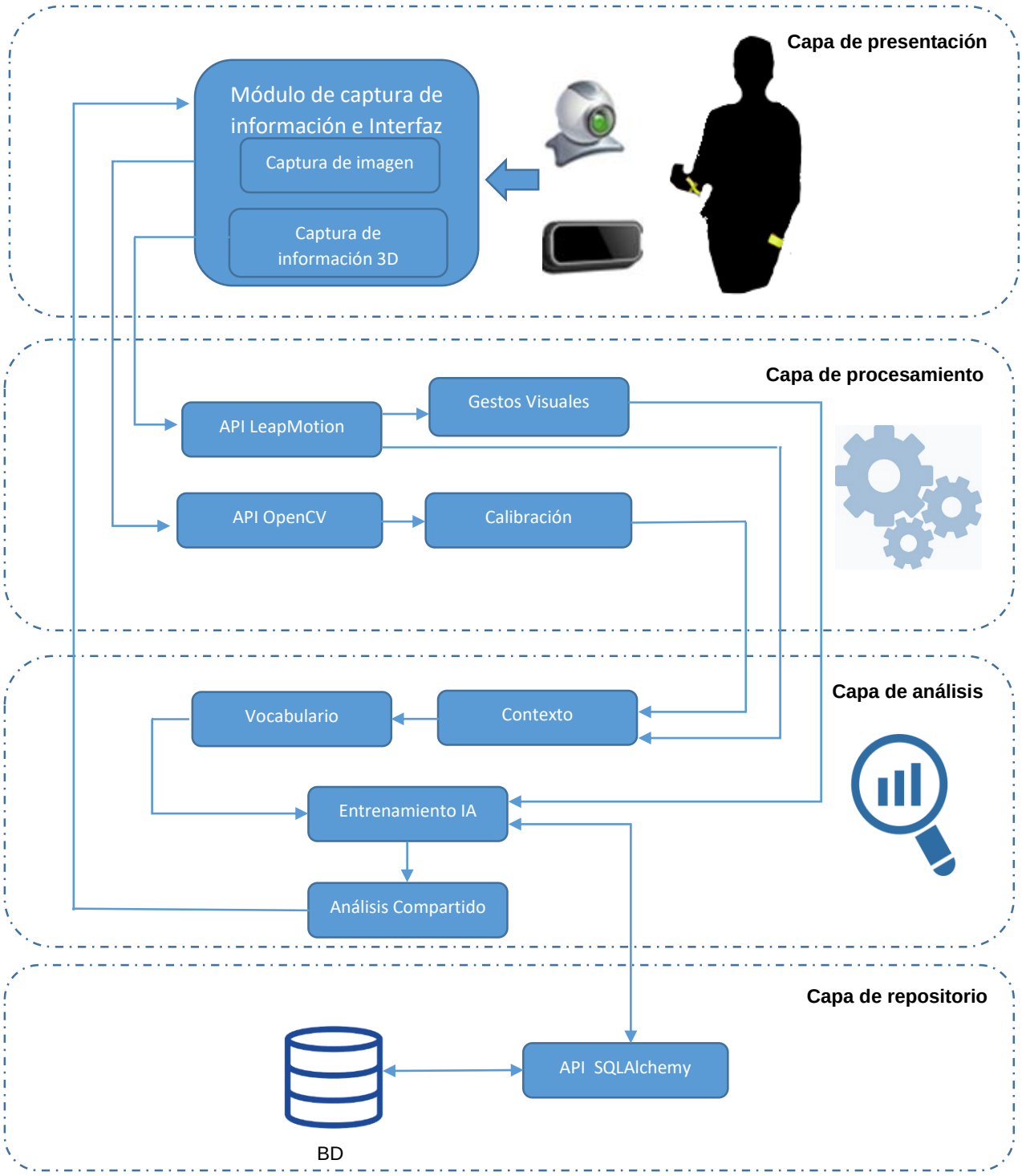



Figura 3.26 Arquitectura Actualizada del intérprete para el *sprint* 7 Julio 2018.

Se realizaron los bocetos correspondientes a la interfaz básica de todas las pantallas disponibles, incluyendo las opciones que tendrá el usuario, esto incluyó rediseñar las interfaces que ya estaban realizadas, tal como se observa en las figuras 3.27, 3.28, 3.29, 3.30, 3.31 y 3.32.

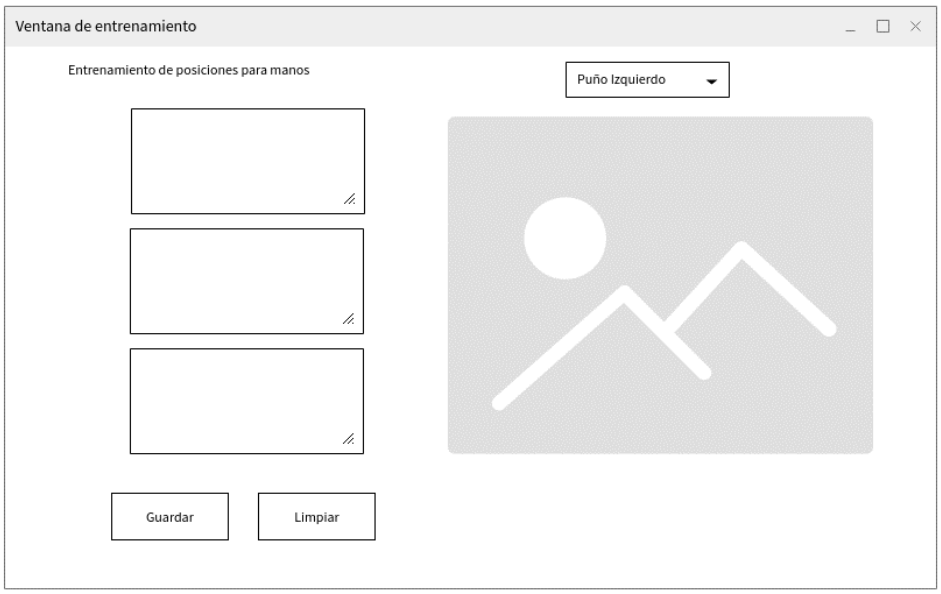



Figura 3.27 Boceto de interfaz de captura de información para el entrenamiento de gestos visuales.

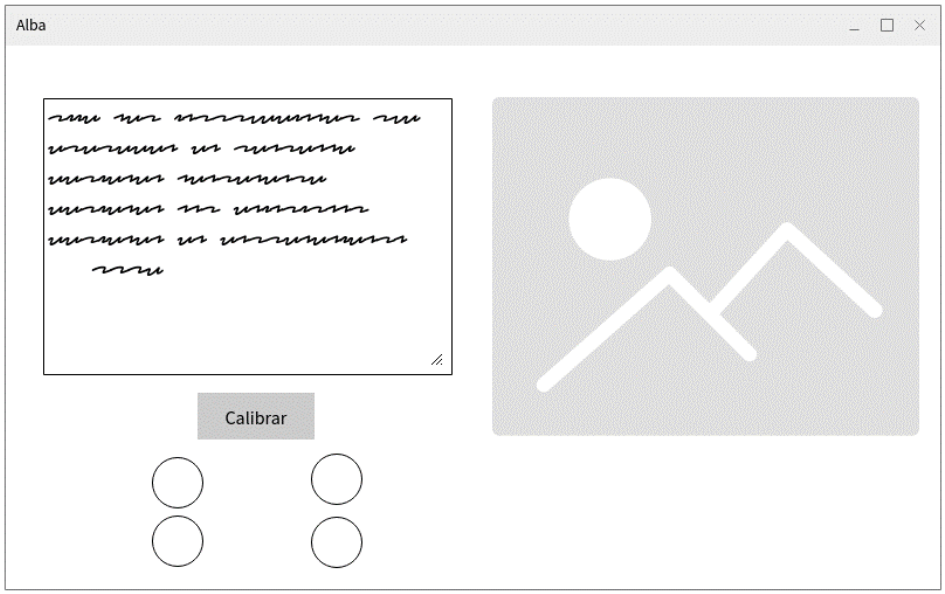



Figura 3.28 Boceto de interfaz inicial del intérprete para activar la calibración para mejorar el procesamiento de imágenes.

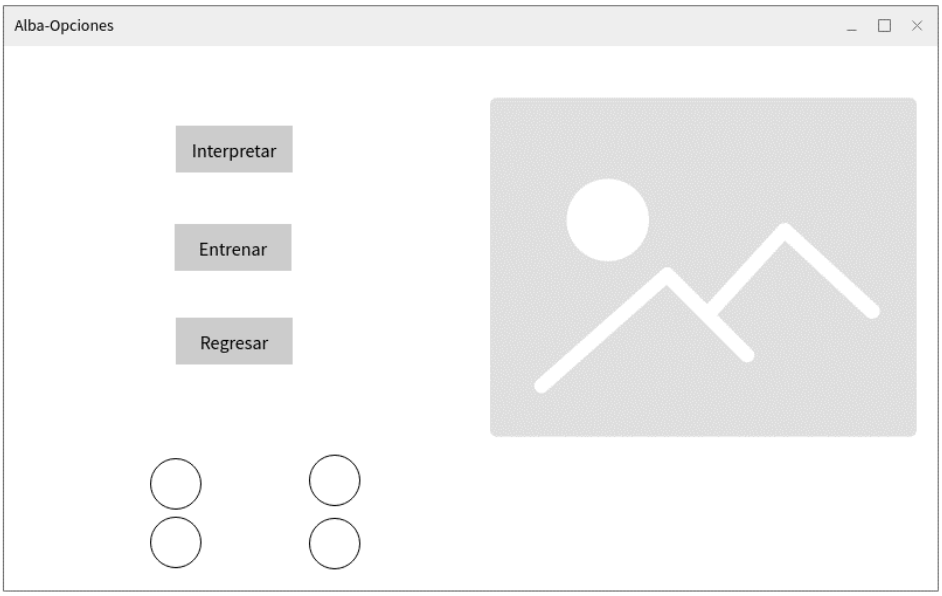



Figura 3.29 Boceto de interfaz de selección de opciones para entrenar o interpretar.

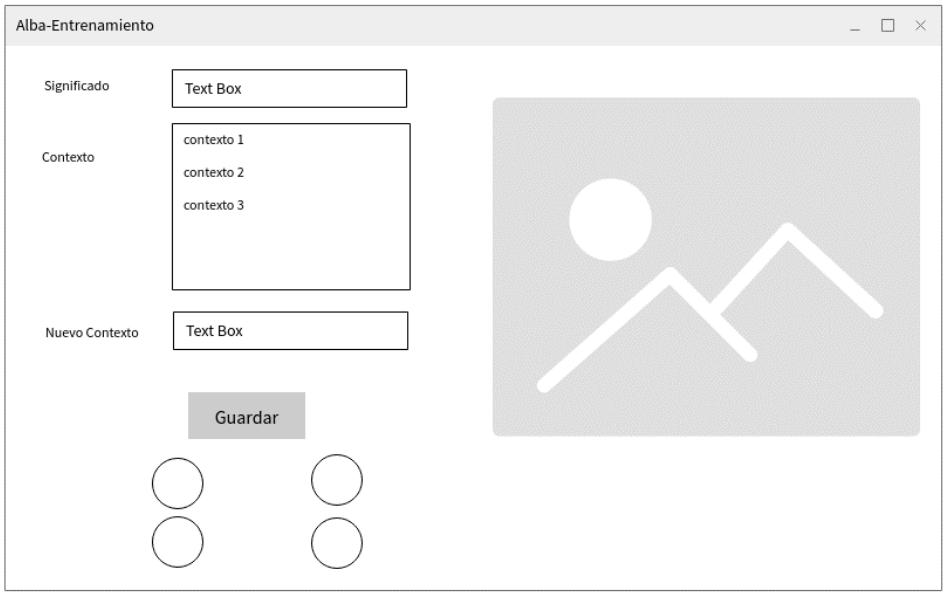



Figura 3.30 Boceto de interfaz de entrenamiento de significados, con selección de contexto y opción para cargar seleccionar significados nuevos o existentes.

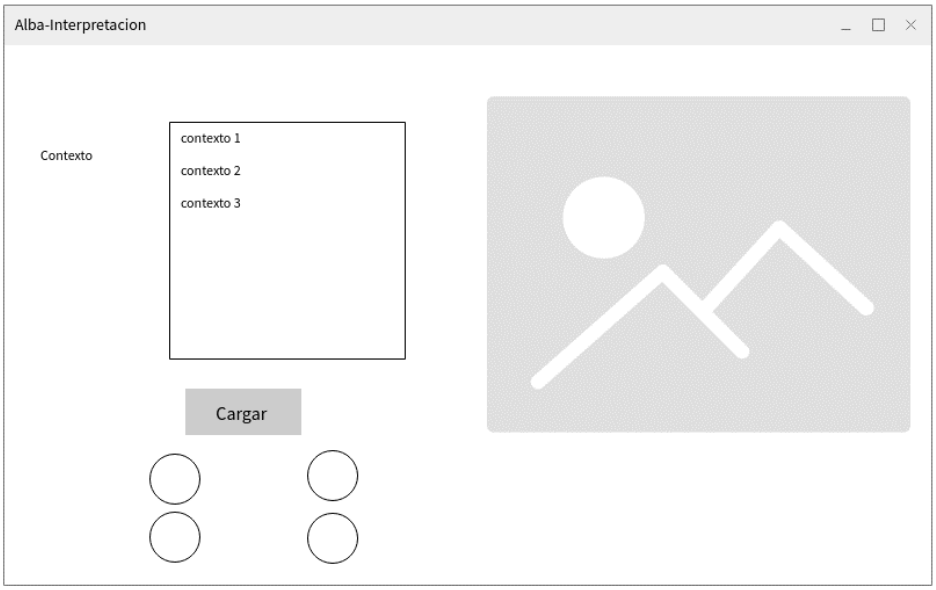



Figura 3.31 Boceto de interfaz de selección de contexto para iniciar la interpretación.

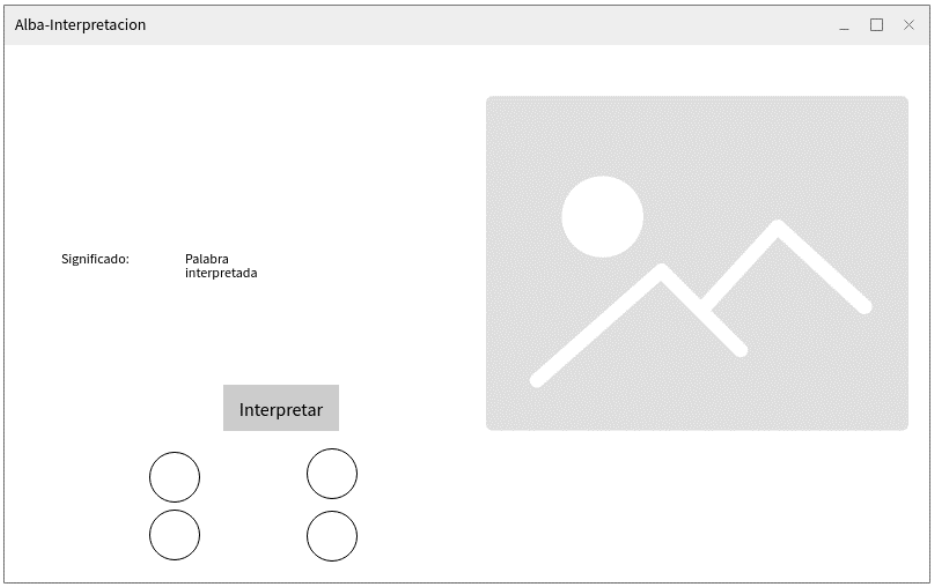



Figura 3.32 Boceto de interfaz de interpretación de seña con vocabulario disponible según el contexto seleccionado.

Como se observa en la figura 3.26, se creó el boceto de una nueva interfaz destinada al entrenamiento de los gestos visuales destinados a permitir el control de la interfaz por parte del usuario, para ello se utilizan solo los datos obtenidos por el dispositivo LeapMotion, permitiendo seleccionar entre cuatro posibles comandos, correspondientes a la mano extendida y la mano en forma de puño para cada brazo, por otra parte el resto de las interfaces contendrán diversos elementos para capturar y mostrar información, siendo recurrentes el área destinada a mostrar la cámara y los círculos en la parte inferior izquierda de la pantalla que representan los comandos disponibles.

Se trabajó en la estandarización de los nombres utilizados en el proyecto, por lo cual se optó por utilizar nombres en inglés descriptivos, para de esta manera permitir una lectura más sencilla y que en caso de que este proyecto sea consultado, esté disponible para un público más amplio y de esta forma se pueda entender el funcionamiento del sistema.

Se agregó la tabla *command_frame*, que almacena la información correspondiente a los gestos visuales, los cuales consisten en la mano derecha o izquierda colocada a la altura de la cabeza y que con la palma extendida o el puño cerrado permiten cuatro posibles configuraciones diferentes, que se utilizan para controlar la interfaz, se determinó que la información obtenida por el dispositivo LeapMotion es suficiente para controlar la interfaz, por lo que la tabla creada aprovecha la estructura existente por medio de múltiples relaciones con la tabla *leap_finger_frame* y solo agrega información extra referente a la posición de los brazos y codos como se observa en las líneas 17 a 22, a continuación se muestra un extracto de la modificación hecha a la clase *LeapMotionPersistenceBuild*, que es la que contiene la declaración utilizada por *SQLAlchemy*:

```
1  class CommandFrame(Base):
2
3      __tablename__='command_frame'
4
5      id = Column(Integer, primary_key=True)
6      command = Column(Integer, nullable=False)
7      hand = Column(Integer, nullable=False)
8
9      palm_direction_x = Column(Float, nullable=False)
10     palm_direction_y = Column(Float, nullable=False)
11     palm_direction_z = Column(Float, nullable=False)
12
13     palm_position_x = Column(Float, nullable=False)
14     palm_position_y = Column(Float, nullable=False)
15     palm_position_z = Column(Float, nullable=False)
16
17     arm_direction_y = Column(Float, nullable=False)
18     arm_direction_z = Column(Float, nullable=False)
19
```

```
20    elbow_position_x = Column(Float, nullable=False)
21    elbow_position_y = Column(Float, nullable=False)
22    elbow_position_z = Column(Float, nullable=False)
23
24    wrist_position_x = Column(Float, nullable=False)
25    wrist_position_y = Column(Float, nullable=False)
26    wrist_position_z = Column(Float, nullable=False)
27
28    command_leap_finger_thumb_id = Column(Integer,
ForeignKey('leap_finger_frame.id'))
29    command_leap_finger_index_id = Column(Integer,
ForeignKey('leap_finger_frame.id'))
30    command_leap_finger_middle_id = Column(Integer,
ForeignKey('leap_finger_frame.id'))
31    command_leap_finger_ring_id = Column(Integer,
ForeignKey('leap_finger_frame.id'))
32    command_leap_finger_pinky_id = Column(Integer,
ForeignKey('leap_finger_frame.id'))
33
34    thumb_command = relationship("LeapFingerFrame",
foreign_keys=[command_leap_finger_thumb_id])
35
36    index_command = relationship("LeapFingerFrame",
foreign_keys=[command_leap_finger_index_id])
37
38    middle_command = relationship("LeapFingerFrame",
foreign_keys=[command_leap_finger_middle_id])
39
40    ring_command = relationship("LeapFingerFrame",
foreign_keys=[command_leap_finger_ring_id])
41
42    pinky_command = relationship("LeapFingerFrame",
foreign_keys=[command_leap_finger_pinky_id])
43
44 class Context(Base):
45    __tablename__='context'
46    id = Column(Integer, primary_key=True)
47    context = Column(String(250), nullable=False)
48
49 class ContextMeaning(Base):
50    __tablename__="context_meaning"
51
52    meaning_id = Column(Integer, primary_key=True, nullable=False)
53    context_id = Column(Integer, primary_key=True, nullable=False)
```

Se agregaron las tablas *context* y *context_meaning*, que almacenan información referente los contexto disponibles y su relación con los significados respectivamente, ya que un significado podría ser válido para más de un contexto, como en el caso del español con la palabra banco que se refiere al mueble para sentarse y banco que se refiere a la institución para guardar dinero, la estructura de la base de datos final se muestra en la figura 3.33.

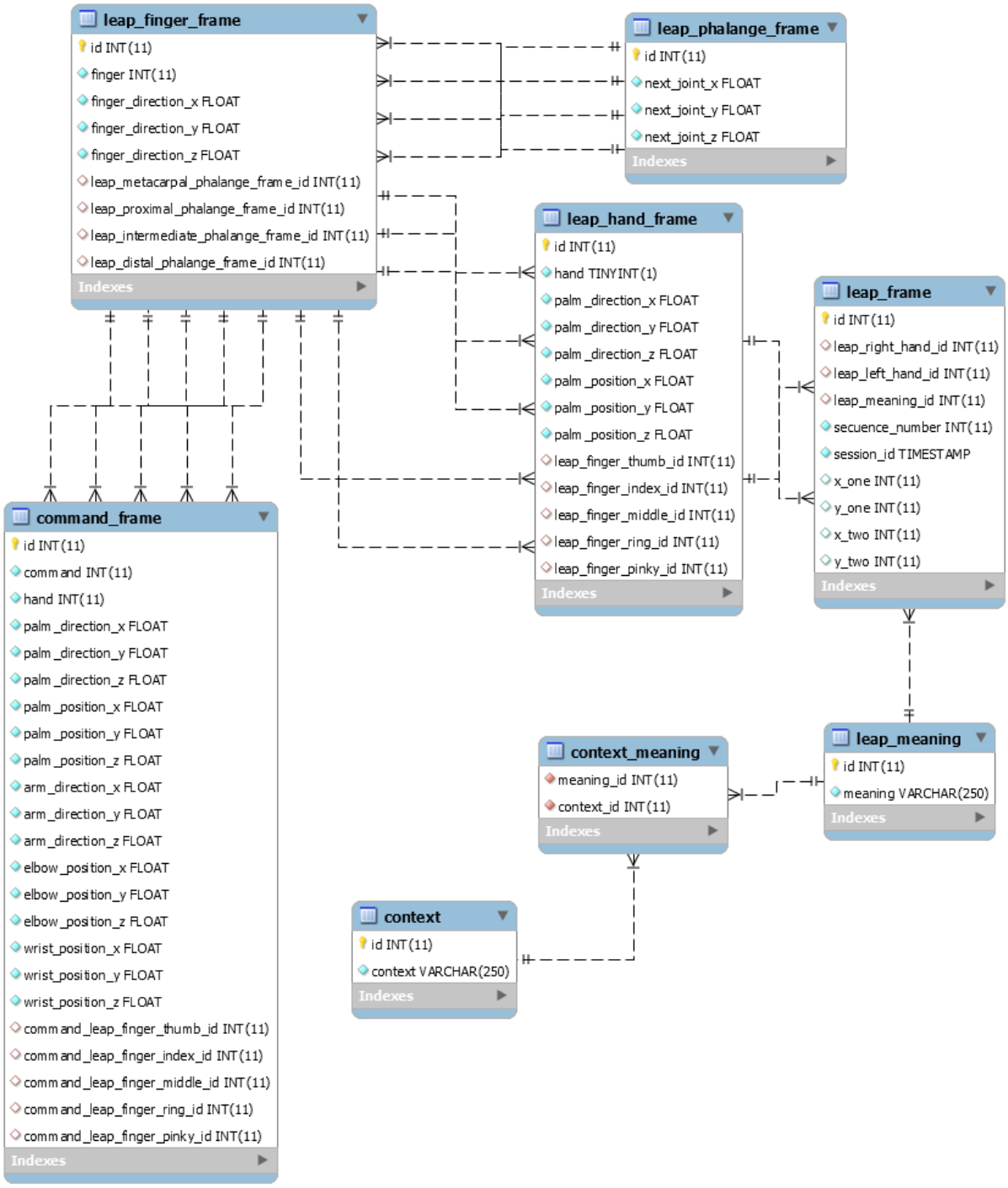



Figura 3.33 Modelo de base de datos final utilizado para el almacenamiento de los datos obtenidos por el control LeapMotion y la cámara Web, correspondiente al *Sprint* 7 Julio 2018.

Se tomó como base el boceto de la figura 3.26 Interfaz de captura de información para el entrenamiento de gestos visuales, para crear el archivo *traingConmmands.py* que contiene la clase *WindowCamLeapMotion* que contiene el código necesario para crear la interfaz gráfica en Tkinter, se muestra un extracto del código generado para dicha clase a continuación y en la figura 3.34 se observa el resultado obtenido:

```
1  import os, sys, inspect
2  import cv2
3  import PIL.Image, PIL.ImageTk
4  import time
5  from tkinter import Frame as FrameInterface
6  from tkinter import Canvas, Button, NW, Tk, Label, StringVar, OptionMenu, Text,
END
7  from collections import OrderedDict
8
9  # Personalize directories
10
11 arch_persistencia = '../bd'
12 src_dir = os.path.dirname(inspect.getfile(inspect.currentframe()))
13 sys.path.insert(0, os.path.abspath(os.path.join(src_dir, arch_persistencia)))
14
15 from LeapMotionCommandFrame import LeapMotionCommandFrame
16
17 arch_leap = '../x64'
18 src_dir = os.path.dirname(inspect.getfile(inspect.currentframe()))
19 sys.path.insert(0, os.path.abspath(os.path.join(src_dir, arch_leap)))
20
21 import Leap
22
23 class WindowCamLeapMotion:
24    # We construct initialize with the window object, the title and the source of
the video
25    def __init__(self, window, window_title, video_source=0):
26
27        # Declaration of variables
28
29        # Variables of the main windows and widgets elements
30        self.window = window
31        self.window.title(window_title)
32        self.video_source = video_source
33
34        # Variables for manipulation and data
35        # Variable of time delay to update the canvas
36        self.delay = 15
37
38        # List to save the data from leap motion and past to the persist method
39        self.frames = list()
40
41        # List to control the validation of valid frames
42        self.valid_frames = list()
43
44        # Variable to identify the number of snapshot
45        self.snapshop_count = 1
46
47        # Object of the actual Leap Motion Controller
48        self.leap_motion_controller = Leap.Controller()
.
.
```

```
.
122
123         def update(self):
124             # Get a frame from the video source
125             ret, frame = self.video.get_frame()
126 
127          if ret:
128              self.photo = PIL.ImageTk.PhotoImage(image =
PIL.Image.fromarray(frame))
129              self.canvas.create_image(0, 0, image = self.photo, anchor = NW)
130 
131          self.window.after(self.delay, self.update)
132 
133         # Method to clear the text area
134         def clearText(self):
135             self.textContenidoLectura.delete("1.0",END)
136             self.textContenidoLectura2.delete("1.0",END)
137             self.textContenidoLectura3.delete("1.0",END)
138 
.
.
.
336 
337 class MyVideoCapture:
338     def __init__(self, video_source=0):
339 
340         # Open the video source
341         self.video = cv2.VideoCapture(video_source)
342         if not self.video.isOpened():
343             raise ValueError("Unable to open video source", video_source)
344 
345         # Get video source width and height
346         self.width = self.video.get(cv2.CAP_PROP_FRAME_WIDTH)
347         self.height = self.video.get(cv2.CAP_PROP_FRAME_HEIGHT)
348 
349     def get_frame(self):
350         if self.video.isOpened():
351             ret, frame = self.video.read()
352             if ret:
353                 # Return a boolean success flag and the current frame converted
to BGR
354                 return (ret, cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
355             else:
356                 return (ret, None)
357         else:
358             return (ret, None)
359 
360     # Release the video source when the object is destroyed
361     def __del__(self):
362         if self.video.isOpened():
363             self.video.release()
364 
365 
366 # This condition check if the file is used individual or as a module call:
367 
368 if __name__ == '__main__':
369     WindowCamLeapMotion(Tk(), "Ventana de entrenamiento",0)
370 
```


Figura 3.34 Interfaz de captura de información para el entrenamiento de gestos visuales.

Para manejar la manipulación de los datos obtenidos por el dispositivo LeapMotion se creó en las primeras iteraciones la clase *LeapMotionFrame*, cuya función radica en servir de intermediario para ajustar la estructura de los datos enviados por el dispositivo a la esperada por las clases contenidas en *LeapMotionPersistenceBuild,* y para el caso específico de los comandos de gestos visuales se creó la clase *LeapMotionCommandFrame* que hereda y sobrescribe los métodos de *LeapMotionFrame* como se observa en la línea 15, para mantener una mejor estructura y control de las características específicas que requiere cada conjunto de datos, a continuación se muestra un extracto de esta clase:

```
1  from sqlalchemy import create_engine
2  from sqlalchemy.orm import sessionmaker
3
4  from LeapMotionPersistenceBuild import Base, LeapFrame, LeapMeaning,
LeapHandFrame, LeapFingerFrame, LeapPhalangeFrame, CommandFrame
5  from LeapMotionFrame import LeapMotionFrame
6
7  engine = create_engine('mysql+pymysql://alba:alba@localhost:3306/alba')
8
9  Base.metadata.bind = engine
10
11 DBSession = sessionmaker(bind=engine)
12 session = DBSession()
13
14
15 class LeapMotionCommandFrame(LeapMotionFrame):
16    def __init__(self):
17        self.left_hand = list()
```

```
18          self.right_hand = list()
19          self.meaning = None
20          self.secuence_number = 0
21
22      def generarListaDatosLeap(self, hand, command):
23          fingers = hand.fingers
24
25          listaDatos = []
26
27          listaDatos.append(hand.direction)
28
29          # Get fingers
30          for finger in fingers:
31
32            dedo = list()
33            dedo.append(finger.type)
34
35              # Get bones
36              for b in range(0, 4):
37                  bone = finger.bone(b)
38                  dedo.append(bone.next_joint)
39                  if b == 3:
40                      dedo.append(bone.direction.to_float_array())
41
42              listaDatos.append(dedo)
43
44          listaDatos.append(hand.palm_position)
45
46          if int(command) < 2:
47              self.left_hand = listaDatos
48
49          if int(command) > 1:
50              self.right_hand = listaDatos
51
52      def saveData(self, command):
53
54          if not (self.left_hand == None):
55              self.saveFeatures(self.left_hand, command, 0)
56          if not (self.right_hand == None):
57             self.saveFeatures(self.right_hand, command, 1)
```

Los datos recolectados y almacenados en la base de datos por medio de esta interfaz se almacenan en la tabla command_frame que cuenta con 23 campos además de la información almacenada en las tablas con las que tiene relaciones, a continuación, se muestra un extracto de la información guardada en la base de datos en las tablas 3.19 y 3.20:

Tabla 3.19 Extracto de la información almacenada en la tabla command_frame.

| command | hand | palm_direction_x | palm_direction_y | palm_direction_z |
|---|---|---|---|---|
| 0 | 0 | 0.47365 | 0.850016 | -0.230496 |
| 0 | 0 | 0.418428 | 0.89675 | -0.144074 |
| 0 | 0 | 0.50005 | 0.831999 | -0.240265 |
| 0 | 0 | 0.239515 | 0.963631 | 0.118524 |

Tabla 3.20 Extracto de los id de las columnas de relación con las tablas de leap_finger_frame almacenadas en la tabla comand_frame.

| command_leap_finger_thumb_id | command_leap_finger_index_id | command_leap_finger_middle_id | command_leap_finger_ring_id |
|---|---|---|---|
| 2366 | 2367 | 2368 | 2369 |
| 2371 | 2372 | 2373 | 2374 |
| 2376 | 2377 | 2378 | 2379 |
| 2381 | 2382 | 2383 | 2384 |
| 2386 | 2387 | 2388 | 2389 |
| 2391 | 2392 | 2393 | 2394 |
| 2396 | 2397 | 2398 | 2399 |
| 2401 | 2402 | 2403 | 2404 |
| 2406 | 2407 | 2408 | 2409 |
| 2411 | 2412 | 2413 | 2414 |
| 2416 | 2417 | 2418 | 2419 |
| 2421 | 2422 | 2423 | 2424 |
| 2426 | 2427 | 2428 | 2429 |

Se cuenta con 480 registros para los comandos con gestos visuales, que a su vez se dividen en 120 registros para cada uno de los 4 comandos disponibles.

El total de puntos de las tareas cerradas para este *sprint* fue de cuarenta y ocho tal como se observa en la captura de pantalla de la interfaz de Taiga que se muestra en la figura 3.35.

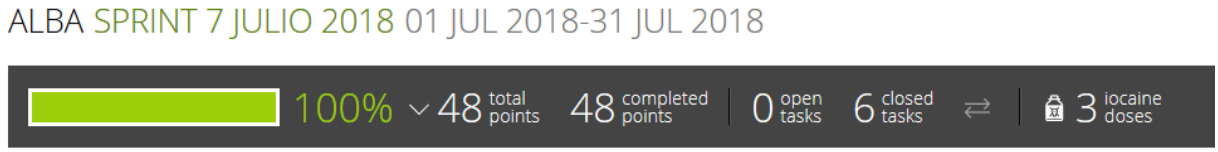



Figura 3.35 Interfaz de Taiga para manejo de tareas de *Sprint* 7 Julio 2018.

### 3.2.11 *Sprint* 8 Agosto 2018

Para el octavo *Sprint* se contó con la siguiente lista de producto (*Product Backlog*) basada en las historias de usuario:

- MCG-01Desarrollo del módulo de interpretación de gestos visuales.
- DMI-02 Desarrollo del módulo de procesamiento de imágenes.
- DMC-02 Desarrollo del módulo de comparación con repositorio.
- DMV-01 Desarrollo del módulo de vocabulario de contexto.
- DMA-01 Desarrollo del módulo de análisis compartido.
- SCE-01Selección de un caso de estudio.
- PCE-01 Realizar pruebas con el caso de estudio seleccionado.

Que a su vez derivaron en la siguiente lista de pendientes del *sprint* (*Sprint Backlog*), junto a sus correspondientes tareas:

- MCG-01Desarrollo del módulo de interpretación de gestos visuales.
  - Creación de la clase especializada para identificar comandos.
  - Integración de la clase especializada en el flujo de ejecución de *Tkinter*.
  - Comprobación de la activación de comandos según la sección de la interfaz de usuario activa.

Se creó la clase *ControlMenu* como clase auxiliar para llevar a cabo la clasificación de las señas detectadas por el dispositivo *LeapMotion* y que a su vez derivan en comandos específicos previamente entrenados como se observa en las líneas 16 a la 36, a continuación, se muestra un extracto del código:

```
1  import Leap
2  from sklearn.model_selection import train_test_split
3  from sklearn import svm
4  from sklearn.neural_network import MLPClassifier
5  import pandas as pd
6  from LeapMotionCommandFrame import LeapMotionCommandFrame
7
8  class ControlMenu():
9
10    def __init__(self, controladorLeap):
11        self.controlador_leap = controladorLeap
12        self.ai = None
13        self.frame_item = LeapMotionCommandFrame()
14
15    # Function to identify the current comand
16    def identifyCommand(self, ail, air):
```

```
17
18          frame = self.controlador_leap.frame()
19          if (not frame.hands.is_empty) and (len(frame.hands) < 2):
20              for hand in frame.hands:
21                  if self.isLeft(hand):
22
23                      # We need to generate a pandas dataframe to use wit the ail
24                      df = pd.DataFrame(self.frame_item.generarDataFrameLeap(hand,
1), index=[0])
25                      r = ail.predict(df)
26
27                  else:
28                      df = pd.DataFrame(self.frame_item.generarDataFrameLeap(hand,
2), index=[0])
29                      r = air.predict(df)
30                      rr = air.predict_proba(df)
31                      print(rr)
32                      # r=0
33
34              return True, r
35          else:
36              return False, 0
37
38      # Function to train command network
39      def trainNetworkCommand(self, repository, network):
40
41          self.repository = repository
42          self.network=network
43
44          self.X = self.repository.drop(['command'], axis=1)
45          self.y = self.repository.command
46
47          X_train, X_test, y_train, y_test = train_test_split(self.X, self.y,
random_state=42)
48
49          if self.network == 0:
50              self.ai = svm.SVC(cache_size=500, C=100, gamma=0.00000001,
decision_function_shape='ovo', kernel='rbf', probability=True)
51          else:
52              self.ai = MLPClassifier(alpha=0.0001, solver='lbfgs',
learning_rate='invscaling', activation='logistic')
53
54          return self.ai.fit(X_train, y_train)
55
56  if __name__ == '__main__':
57      ControlMenu(Leap.Controller())
```

Como se observa en el extracto anterior, la función *trainNetworkCommand* está preparada para retornar un clasificador entrenado con el algoritmo de máquina de vectores de soporte o un clasificador entrenado con el algoritmo de red neuronal de perceptron multicapa, según el valor de la variable de entrada *network*, este diseño permitió mayor flexibilidad para realizar pruebas.

Los métodos pertenecientes a la clase *ControlMenu*, se integraron a la clase principal *MainWindows* que controla la ejecución de la ventana del programa y sus diversas

secciones, el funcionamiento del componente que detecta y analiza los gestos visuales, tiene la siguiente lógica:

- El componente permanentemente obtiene información del dispositivo para detectar gestos a interpretar.
- Cuando detecta una sola mano en su campo de visión recolecta datos del dispositivo.
- Analiza constantemente los datos con el modelo entrenado previamente.
- Una vez clasificado como uno de los cuatro posibles comandos lo almacena de manera temporal en memoria de ejecución.
- Repite el proceso para detectar un nuevo gesto después de 900 milisegundos.
- Cuando se detecta cinco veces el mismo gesto, es decir, cuando el usuario mantiene el mismo gesto durante cuatro segundos y medio, se activa el comando correspondiente según la sección activa.

El esquema de interpretación de gestos visuales se muestra en la figura 3.36.

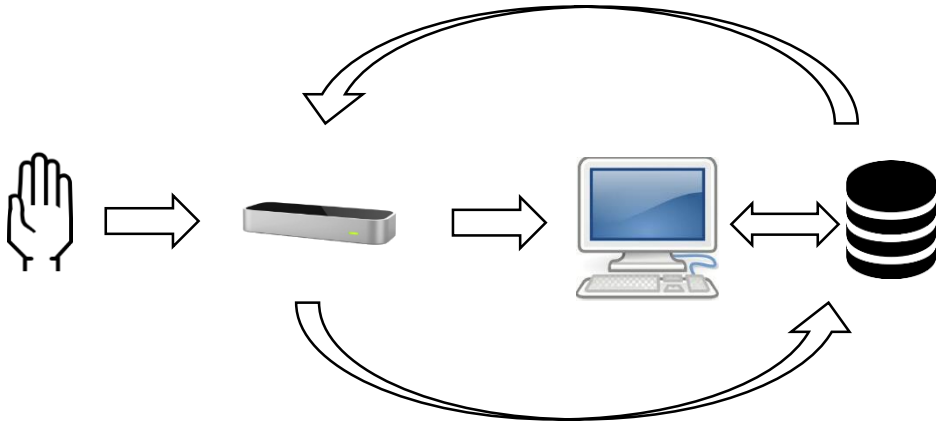


Figura 3.36 Esquema de interpretación de gestos visuales.

La función *checkLeapMotionGesture* encargada de manejar la revisión de los gestos visuales, se ejecuta cada 900 milisegundos como se aprecia en la línea 38, a continuación, se incluye el código de dicha función:

```
1  def checkLeapMotionGesture(self):
2    # Check if the hand is detected
3
4      if self.menu_control.isOnlyOneHandDetected():
5        # Identify the gesture
6        r, command = self.menu_control.identifyCommand(self.ai_left, self.ai_right)
7        if r:
8            # Check the last gesture detect and the current and acumulate it
9            if self.last_gesture_result == command:
```

```
10
11                 self.accumulator_gesture = self.accumulator_gesture + 1
12                 if command == 0:
13                     self.left_fist_indicator.set(self.left_fist_indicator.get() +
"|")
14                 if command == 1:
15                     self.left_palm_indicator.set(self.left_palm_indicator.get() +
"|")
16                 if command == 2:
17                     self.right_fist_indicator.set(self.right_fist_indicator.get() +
"|")
18                 if command == 3:
19                     self.right_palm_indicator.set(self.right_palm_indicator.get()
+ "|")
20             else:
21                 self.accumulator_gesture = 0
22                 self.last_gesture_result = command
23
24                 self.left_fist_indicator.set("")
25                 self.left_palm_indicator.set("")
26                 self.right_fist_indicator.set("")
27                 self.right_palm_indicator.set("")
28
29             if self.accumulator_gesture == 5:
30                 self.reinitializeGestures()
31                 self.activateMotionCommand(command)
32
33         else:
34             self.reinitializeGestures()
35     else:
36         self.reinitializeGestures()
37
38     self.window.after(900, self.checkLeapMotionGesture)
```

La función *activateMotionCommand* es la encargada de activar el comando correspondiente una vez que se detectó la incidencia del mismo comando cinco veces, la ejecución de dicha opción se da de acuerdo a la evaluación de la sección activa en las líneas 4 a la 26, un extracto del código correspondiente se incluye a continuación:

```
1   # Funtion to activate the motion command accord to the actual page
2   def activateMotionCommand(self, command):
3
4       if self.actual_section == 0:
5           if command == 1:
6               self.startCalibration(0)
7
8       # If the section is 1 we need to choose between
9       if self.actual_section == 1:
10          # In this section are two commands avaliable
11          if command == 0:
12              self.configure_change_section(0)
13          if command == 1:
14              # We show the select context for interpreting window
15              self.changeToInterpreting()
16          if command == 2:
17              # We show the training window
18              self.changeToTraining()
19
```

```
20      # If the section is 2
21      if self.actual_section == 2:
22          if command == 0:
23              self.configure_change_section(1)
24
25      # If section is 3
26      if self.actual_section == 3:
27          if command == 0:
28              # We empty the list of data to save
29              self.data_to_save_in_training = list()
30              self.configure_change_section(1)
31          if command == 1:
32              # Activate the count down
33              self.saveMeaningContext()
```

Se agregaron indicadores de gestos detectados para que el usuario visualice las opciones disponibles, en la figura 3.37 se observa el cambio gradual.

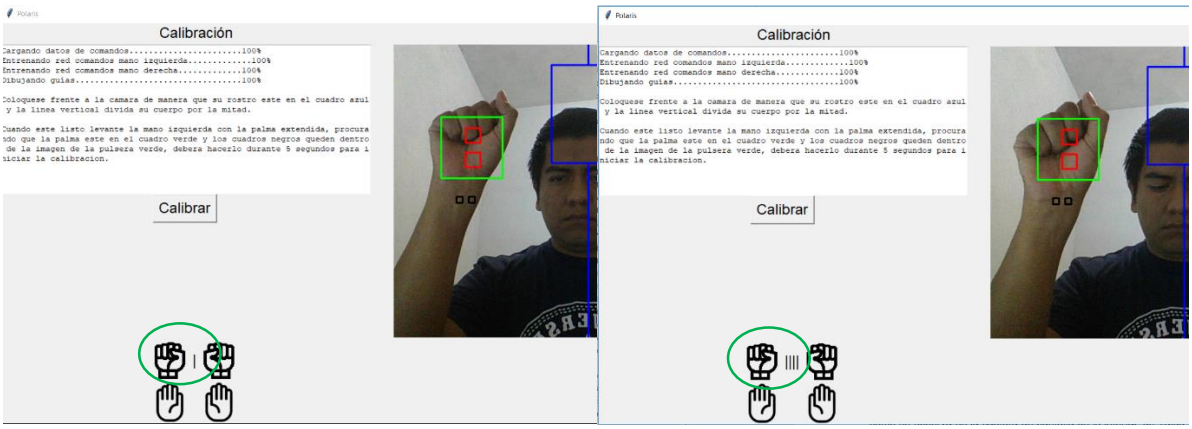



Figura 3.37 Indicador de detección de gesto visual en la interfaz inicial.

El total de puntos de las tareas cerradas para este *sprint* fue de cuarenta y tres tal como se observa en la captura de pantalla de la interfaz de Taiga que se muestra en la figura 3.38.

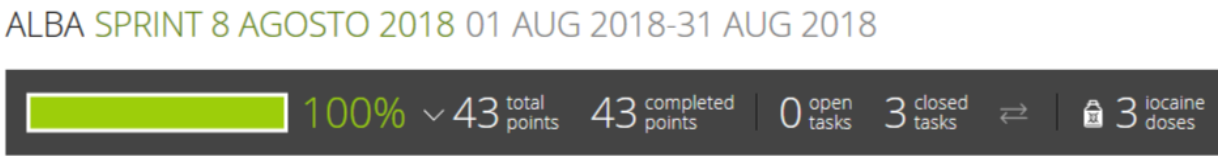



Figura 3.38 Interfaz de Taiga para manejo de tareas de *Sprint* 8 Agosto 2018.

### 3.2.12 *Sprint* 9 Septiembre 2018

Para el noveno *Sprint* se contó con la siguiente lista de producto (*Product Backlog*) basada en las historias de usuario:

- DMI-02 Desarrollo del módulo de procesamiento de imágenes.
- DMC-02 Desarrollo del módulo de comparación con repositorio.
- DMV-01 Desarrollo del módulo de vocabulario de contexto.
- DMA-01 Desarrollo del módulo de análisis compartido.
- SCE-01Selección de un caso de estudio.
- PCE-01 Realizar pruebas con el caso de estudio seleccionado.

Que a su vez derivaron en la siguiente lista de pendientes del *sprint* (*Sprint Backlog*), junto a sus correspondientes tareas:

- DMI-02 Desarrollo del módulo de procesamiento de imágenes.
    - Creación de las funciones necesarias para acceder a los cuadros de la cámara Web.
    - Creación de las funciones necesarias para la manipulación de las imágenes y selección de las áreas de interés.
    - Creación de las funciones necesarias para la calibración de valores del objeto a seguir.
    - Aplicación de las técnicas de procesamiento de imagen.

Se creó la clase *MyVideoCapture*, para concentrar las funciones destinadas a acceder a la cámara Web y obtener los cuadros de manera recurrente, durante toda la ejecución del programa, a continuación, se muestra un extracto del contenido de dicha clase:

```
1  import cv2
2  import os
3  import numpy as np
4
5  class MyVideoCapture:
6      def __init__(self, actual_section, video_source=0):
7
8          self.actual_section = actual_section
9
10         # Open the video source
11         self.video = cv2.VideoCapture(video_source)
12         if not self.video.isOpened():
13             raise ValueError("Unable to open video source", video_source)
14
```

```
15          # Get video source width and height
16          self.width = self.video.get(cv2.CAP_PROP_FRAME_WIDTH)
17          self.height = self.video.get(cv2.CAP_PROP_FRAME_HEIGHT)
18
19      def get_frame(self, original):
20          if self.video.isOpened():
21              ret, frame = self.video.read()
22              if ret:
23                 return (ret, cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
24
25            else:
26                  return (ret, None)
27          else:
28              return (ret, None)
29
30      # Release the video source when the object is destroyed
31      def __del__(self):
32          if self.video.isOpened():
33              self.video.release()
34
35
36  if __name__ == '__main__':
37      MyVideoCapture(0, 0)
```

La función *get_frame* es llamada de manera recurrente en la línea 399 desde el hilo principal de ejecución del programa, controlado por *Tkinter* desde la clase principal *MainWindows*, lo cual permite mostrar al usuario una serie de imágenes actualizadas en tiempo real en un elemento canvas, que funcionan como un video en vivo, a continuación, se muestra un extracto del código que controla el ciclo de ejecución:

```
10 arch_movement_user_interface = '../mui'
11 src_dir = os.path.dirname(inspect.getfile(inspect.currentframe()))
12 sys.path.insert(0, os.path.abspath(os.path.join(src_dir,
arch_movement_user_interface)))
13
14 from ControlMenu import ControlMenu
15 from MyVideoCapture import MyVideoCapture
16 from TrainingAuxiliarFunctions import TrainingAuxiliarFunctions
17
18 class MainWindow():
19     def __init__(self, window, window_title, video_source=0):
20
22         # Declaration of variables
         .
         .
         .
146         # This will try to open the Camera in the source received
147         self.video = MyVideoCapture(self.actual_section, self.video_source)
148         # Variable of time delay to update the canvas
149         self.delay = 15
         .
         .
         .
228         # Create a canvas with the video size
229         self.canvas = Canvas(self.frame_video, width=self.video.width, height =
self.video.height)
230         self.canvas.grid(row=0,column=0)
         .
         .
```

```
        .

342         # The update method for the canvas-openCV will be automatically called every
delay milliseconds
343         self.update()
        .
        .
        .
393         self.checkLeapMotionGesture()
394
395         self.window.mainloop()
396
397     def update(self):
398         # Get a frame from the video source
399         ret, frame = self.video.get_frame(self.actual_section)
400
401         if ret:
402             self.photo = PIL.ImageTk.PhotoImage(PIL.Image.fromarray(frame))
403             self.canvas.create_image(0, 0, image = self.photo, anchor = NW)
404
405         self.window.after(self.delay, self.update)
```

Tal como se observa en el extracto anterior, una vez inicializada la variable *self.video* con la sección actual y la cámara Web que la alimentará, se procede a declarar la variable *self.delay* que controla el tiempo de refrescamiento por medio de la función *self.update* que se llama a sí misma de manera recurrente y actualiza el canvas con la imagen correspondiente al cuadro actual, permitiendo experimentar una transición fluida simulando un video en tiempo real, también se observa que el tamaño del video se utiliza para determinar el tamaño de algunos elementos de la interfaz gráfica, por medio de los atributos *width* y *height*.

En las figuras 3.39 y 3.40 se observa el canvas actualizado en diversas secciones.

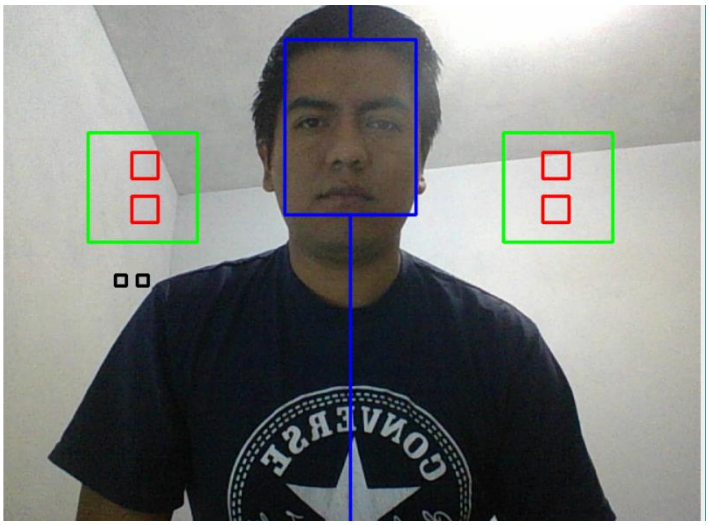


Figura 3.39 Canvas actualizado en la sección de calibración.

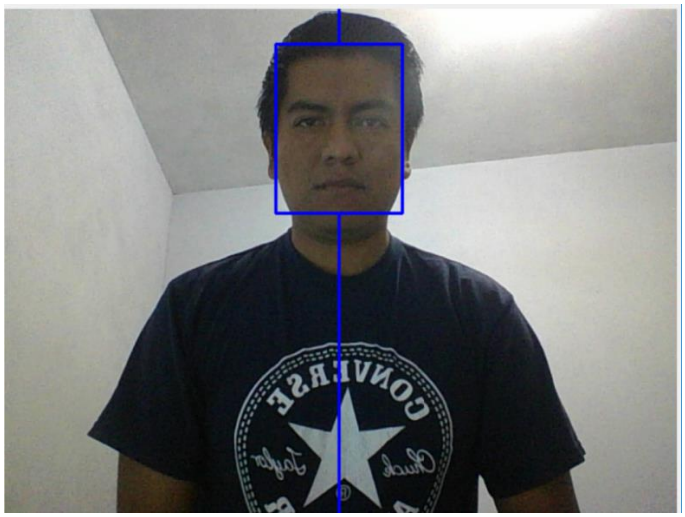


Figura 3.40 Canvas actualizado en la sección de selección de actividad.

Durante el desarrollo del proyecto se propusieron diversas secciones candidatas a ser utilizadas como áreas de interés, sin embargo, después de analizar las técnicas de tratamiento de imágenes y realizar los rediseños necesarios durante el *sprint* 7 Julio, se determinó utilizar un marcador en forma de pulsera de un color altamente diferente al de la piel humana (para las pruebas se han utilizado pulseras de color verde fluorescente y fucsia), para poder dar seguimiento a la posición estimada de la mano, por ello se decidió crear marcadores en las imágenes que alimentan al canvas, para ayudar al usuario a ubicarse en una posición óptima para la detección de las manos, estos marcadores se dividen de la siguiente forma:

1. Un rectángulo de color azul que ayuda al usuario a colocar su rostro dentro de él, acompañado por una línea que divide la imagen por la mitad, cuya funcionalidad es permitir que el usuario se ubique correctamente al centro de la cámara Web.
2. Dos cuadrados verdes cuya funcionalidad es brindar al usuario una guía para colocar la palma de su mano para lograr una posición óptima para el reconocimiento de gestos visuales, a su vez cuentan con dos cuadrados rojos cuya funcionalidad original era indicar el área de donde se extrae el área de interés de la palma de la mano (aunque esta área de interés no se está utilizando, se decidió dejar el marcador, ya que permite a los usuarios centrar su mano de manera más precisa).
3. Dos rectángulos negros que indican el área de interés del marcador en forma de pulsera que ayudan al usuario a ubicar los cuadros dentro de su pulsera durante la calibración, permitiendo de esta manera mejorar el rendimiento del módulo de procesamiento de imágenes.

Solo el marcador número uno se mantiene durante todo el programa, los otros dos, se eliminan una vez que se terminó la calibración, para evitar distracciones.

A continuación, se muestra el código correspondiente a la función *get_frame* de la clase *MyVideoCapture*, encargada de dibujar los marcadores en tiempo real:

```
28 def get_frame(self, original):
29     if self.video.isOpened():
30         ret, frame = self.video.read()
31         if ret:
32             if original < 7:
33                 # Draw a line to use for guide
34                 cv2.line(frame, (int(self.width / 2), 0),(int(self.width / 2),
35), (255, 0, 0), 2)
35                 cv2.line(frame, (int(self.width / 2), 195), (int(self.width / 2),
int(self.height)), (255, 0, 0), 2)
36                 # Draw a square to use for guide
37                 cv2.rectangle(frame, (int(self.width / 2) - 60, 35),
(int(self.width / 2) + 60, 195), (255, 0, 0), 2)
38
39                 #If the actual section is 0 we need to draw guides for the hands
40                 if original == 0:
41                     # Right
42                     cv2.rectangle(frame, (int(self.width / 2) - 240, 120),
(int(self.width / 2) - 140, 220), (0, 255, 0), 2)
43                     # Left
44                     cv2.rectangle(frame, (int(self.width / 2) + 140, 120),
(int(self.width / 2) + 240, 220), (0, 255, 0), 2)
45                     # Visible sections to take colors
46                     cv2.rectangle(frame, (int(self.width / 2) - 200, 138),
(int(self.width / 2) - 176, 162), (0, 0, 255), 2)
47
48                     cv2.rectangle(frame, (int(self.width / 2) - 200, 178),
(int(self.width / 2) - 176, 202), (0, 0, 255), 2)
49
50                     # left hand
51                     cv2.rectangle(frame, (int(self.width / 2) + 200, 138),
(int(self.width / 2) + 176, 162), (0, 0, 255), 2)
52
53                     cv2.rectangle(frame, (int(self.width / 2) + 200, 178),
(int(self.width / 2) + 176, 202), (0, 0, 255), 2)
54
55                     # Bracelete left hand
56                     cv2.rectangle(frame, (int(self.width / 2) + 215, 250),
(int(self.width / 2) + 205, 260), (0, 0, 0), 2)
57
58                     cv2.rectangle(frame, (int(self.width / 2) + 195, 250),
(int(self.width / 2) + 185, 260), (0, 0, 0), 2)
59
60                 final_frame = cv2.flip(frame, 1)
61                 return (ret, cv2.cvtColor(final_frame, cv2.COLOR_BGR2RGB))
62             else:
63                 # We only return the original image to manipulate the roi
64                 final_frame = cv2.flip(frame, 1)
65                 return (ret, cv2.cvtColor(final_frame, cv2.COLOR_BGR2RGB))
66         else:
67             return (ret, None)
68     else:
69         return (ret, None)
```

Tal como se aprecia en el código anterior la función devolverá la imagen con los marcadores correspondientes o la imagen original, de acuerdo al valor de la variable de entrada original, lo cual permite reutilizar este código tanto para refrescar el canvas como para obtener las áreas de interés a analizar.

Con el fin de permitir la máxima eficiencia en el procesamiento de las imágenes y a su vez lograr la mayor flexibilidad posible frente a variables de las que no se tiene control (iluminación, disponibilidad de marcadores, entre otras), se dotó al sistema de una fase de calibración, dicha fase permite seleccionar un conjunto de pixeles presentes en un área de interés determinada y obtener el valor promedio de dicha área, para realizar el posterior análisis de las imágenes en busca de la posición del marcador, de esta forma en conjunto con diversas técnicas de procesamiento de imágenes se busca superar los retos que implica el uso del programa en un ambiente de pruebas real.

La función *startCalibration* de la clase *MainWindow* es la encargada de iniciar el proceso de calibración, obteniendo las áreas de interés y pasando dichas áreas a la función *getAverageFromRoi* en las líneas 582 a la 585, a continuación se muestra un extracto de la función:

```
.
.
.
574 def startCalibration(self, area):
575     if area == 0:
576         ret, frame = self.video.get_frame(7)
577         if ret:
578 
579             self.text_calibration_instructions.insert(END, "\nInicia Calibración
de Pulsera: ..... \n")
580 
581             # We get the first square for bracelete)
582             roi_left_bracelet_upper = frame[250:260, 105:115]
583             # we get the second square for bracelete
584             roi_left_bracelet_down = frame[250:260, 125:135]
585             self.average_bracelet_value =
self.video.getAverageFromRoi(roi_left_bracelet_upper, roi_left_bracelet_down, frame)
586             self.wait_to_change(1)
587 
588         else:
589             self.text_calibration_instructions.insert(END, "\nError en la
conexion de la camara vuelva a intentar.\n")
```

Por otra parte, la función *getAverageFromRoi* de la clase *MyVideoCapture* es la encargada del cálculo del valor promedio en las líneas 116 a la 127, su extracto de código puede observarse a continuación:

```
112 def getAverageFromRoi(self, roi_upper, roi_lower, frame):
113 
```

```
114     color_average = list()
115 
116     upper_hsv = cv2.cvtColor(roi_upper, cv2.COLOR_BGR2HSV)
117     lower_hsv = cv2.cvtColor(roi_lower, cv2.COLOR_BGR2HSV)
118 
119     au = int(np.mean(upper_hsv[:, :, 0])) + 50
120     au2 = int(np.mean(upper_hsv[:, :, 1])) + 50
121 
122     al = int(np.mean(lower_hsv[:, :, 0])) - 30
123     al2 = int(np.mean(lower_hsv[:, :, 1])) - 30
124 
125     average_upper = np.array([au, au2, 255])
126 
127     average_lower = np.array([al, al2, 40])
128 
129     color_average.append(average_lower)
130     color_average.append(average_upper)
131 
132     return color_average
```

Tal Como se observa en el código anterior se hace uso de algunas técnicas de procesamiento de imágenes, específicamente del cambio de espacio de color *RGB* a *HSV*, para ignorar el análisis de los valores del canal *V* referente al brillo de la imagen, ya que de esta manera se minimizan las variaciones de color debido a la posición de fuentes de luz en el ambiente, también se aprecia el uso de las funciones *mean* y *array* de la biblioteca *NumPy* que permiten obtener los valores promedio de los pixeles de las áreas de interés en aquellos canales que son relevantes para su análisis, de esta forma se obtienen dos matrices de tres dimensiones que contienen los valores promedios de máximos y mínimos que se utilizan en el análisis de las imágenes en el resto del intérprete.

Una vez obtenidos los valores promedios de las áreas de interés se utilizan en conjunto con otras técnicas de procesamiento de imágenes para buscar e identificar las pulseras utilizadas como marcadores, para ello se toma como entrada la imagen correspondiente al cuadro actual y los valores promedio obtenidos, iniciando un cambio en el espacio de color de la imagen a HSV, para posteriormente generar una mascara por medio de la función *inRange* de la biblioteca openCV, para obtener las coordenadas de los pixeles que cumplen con los valores mínimo y máximo obtenidos en el proceso de calibración, con esta mascara se extraen los pixeles originales de la imagen que coincidan para después transformarlos en una imagen en escala de grises para poder aplicar a su vez una conversión binaria por medio de la aplicación del algoritmo *OTSU*, de esta forma se procede a realizar una búsqueda de componentes conectados y una serie de transformaciones morfológicas, cuyo objetivo es eliminar secciones no relevantes de la imagen, para finalmente obtener los centroides de estas áreas y calcular sus coordenadas con base a una cuadrícula imaginaria conformada por siete columnas y siete filas.

En la figura 3.41 se muestran las técnicas de procesamiento de imágenes utilizadas, así como el orden seguido en dicho análisis.

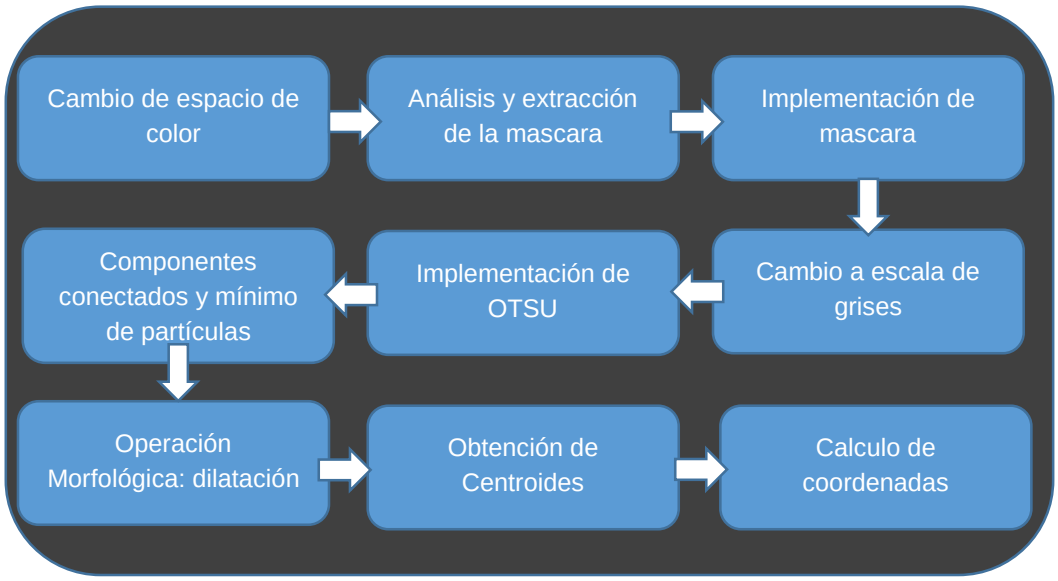



Figura 3.41 Técnicas utilizadas durante el proceso de análisis de la imagen.

A continuación, se incluye un extracto del código correspondiente a la función *getCoordenates*, *clearImage*, *getCentroids* y *determineCoordenates* que forman parte de la clase *TrainingAuxiliaryFunctions*, encargadas del procesamiento de la imagen:

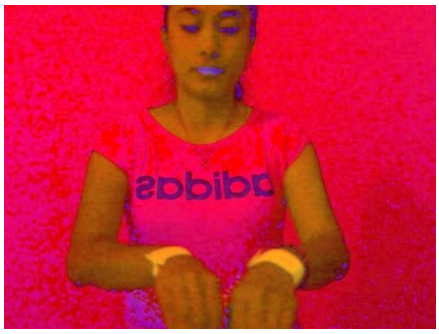
```
89 def getCoordenates(self, raw_training_data_to_save, average_bracelet_value):
90     result = list()
91 
92     n = 0
93 
94     for i in raw_training_data_to_save:
95         result.append(self.clearImage(i, average_bracelet_value, n))
96         n = n + 1
97 
98     return result
99 
100 def clearImage(self, i, average_bracelet_value, n=0):
101     result = list()
102     centroids = list()
103 
104     # Timestamp
105     self.times = time.time()
106 
107     normalizate = cv2.cvtColor(i[1], cv2.COLOR_BGR2RGB)
108 
109     frame2 = cv2.cvtColor(i[1], cv2.COLOR_BGR2HSV)
110 
111     mask = cv2.inRange(frame2, average_bracelet_value[0],
average_bracelet_value[1])
112 
```

```
113     res = cv2.bitwise_and(i[1], i[1], mask=mask)
114 
115     # We convert the color image with mask to gray
116     gray = cv2.cvtColor(res, cv2.COLOR_BGR2GRAY)
117 
118     # We convert the gray image to binary one with otsu
119     thresh, im_bw = cv2.threshold(gray, 128, 255, cv2.THRESH_BINARY |
cv2.THRESH_OTSU)
120 
121     # find all the connected components (white blobs in the image)
122     nb_components, output, stats, centroids =
cv2.connectedComponentsWithStats(im_bw, connectivity=8)
123     sizes = stats[1:, -1]
124     nb_components = nb_components - 1
125 
126     # minimum size of particles we want to keep (number of pixels)
127     min_size = 250
128 
129     # resulting image
130     img2 = np.zeros(output.shape)
131     # for every component in the image, you keep it only if it's above min_size
132     for i in range(0, nb_components):
133         if sizes[i] >= min_size:
134             img2[output == i + 1] = 255
135     # Change the type of the array to use it with cv2 functions
136     img2 = img2.astype(np.uint8)
137 
138     kernel = np.ones((5, 5), np.uint8)
139     dilation = cv2.dilate(img2, kernel, iterations=1)
140     centroids = self.getCentroids(dilation, normalizate, n)
150     result = self.determineCoordenates(centroids)
151 
152     return result
153 
154 def getCentroids(self, dilation, normalizate, n=0):
155 
156     result = list()
157 
158     im2, contours, hierarchy = cv2.findContours(dilation, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
159 
160     for c in contours:
161         M = cv2.moments(c)
162 
163         cX = int(M["m10"] / M["m00"])
164         cY = int(M["m01"] / M["m00"])
165 
166         result.append([cX,cY])
167 
168     return result
169 
170 def determineCoordenates(self, centroids):
171     result = list()
172 
173     for c in centroids:
174         x_coordinate = self.searchCoordinates(c[0], self.limits_matrix_x)
175         y_coordinate = self.searchCoordinates(c[1], self.limits_matrix_y)
176 
177         result.append([x_coordinate, y_coordinate])
```

```
178
179     return result
```

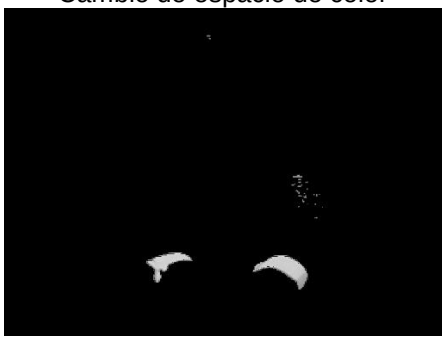
El procesamiento de las imágenes se hace de forma automática y transparente para el usuario, sin embargo, en la figura 3.42, se muestra el resultado de las técnicas de procesamiento de imágenes aplicadas a un cuadro en específico para visualizar la transformación que se sigue:

Imagen original

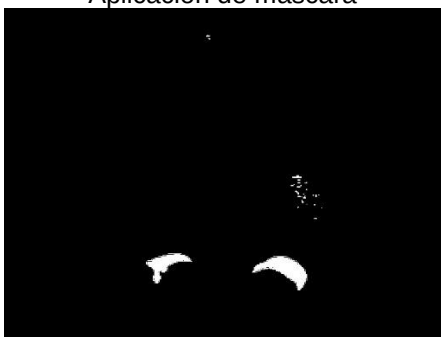


Cambio de espacio de color

Aplicación de máscara



Cambio a escala de grises



Implementación OTSU

Componentes conectados

Transformación morfológica Ubicación de centroides

Figura 3.42 Procesamiento de imágenes con las técnicas seleccionadas.

Por último, una vez que se obtienen los centroides, se procede a ubicarlos en una cuadrícula imaginaria, para obtener los valores de coordenadas que serán guardados en la base de datos, tal como se muestra en la figura 3.43.

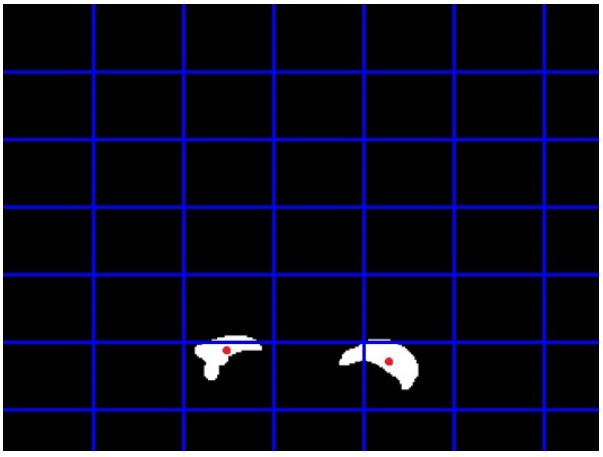


Figura 3.43 Cálculo de coordenadas de los centroides obtenidos.

El total de puntos de las tareas cerradas para este *sprint* fue de sesenta tal como se observa en la captura de pantalla de la interfaz de Taiga que se muestra en la figura 3.44.


Figura 3.44 Interfaz de Taiga para manejo de tareas de *Sprint* 9 Septiembre 2018.

### 3.2.13 *Sprint* 10 Octubre 2018

Para el décimo *Sprint* se contó con la siguiente lista de producto (*Product Backlog*) basada en las historias de usuario:

- DMC-02 Desarrollo del módulo de comparación con repositorio.
- DMV-01 Desarrollo del módulo de vocabulario de contexto.
- DMA-01 Desarrollo del módulo de análisis compartido.
- SCE-01Selección de un caso de estudio.
- PCE-01 Realizar pruebas con el caso de estudio seleccionado.

Que a su vez derivaron en la siguiente lista de pendientes del *sprint* (*Sprint Backlog*), junto a sus correspondientes tareas:

- DMV-01 Desarrollo del módulo de vocabulario de contexto.
    - Creación de la interfaz para la selección de opciones de entrenamiento e interpretación.
    - Creación de la interfaz de selección de contexto.
    - Creación de funciones necesarias para guardar los datos recolectados.
- DMC-02 Desarrollo del módulo de comparación con repositorio.
    - Creación de las funciones necesarias para entrenar las redes neuronales según el vocabulario del contexto seleccionado.

Tomando en cuenta el boceto de las figuras 3.28 y 3.29 se crearon las interfaces que se muestran en la figura 3.45 y 3.46, correspondientes a la selección de opciones de entrenamiento y la selección y creación de contextos y significados:


Figura 3.45 Interfaz de selección de opción de entrenamiento o interpretación.

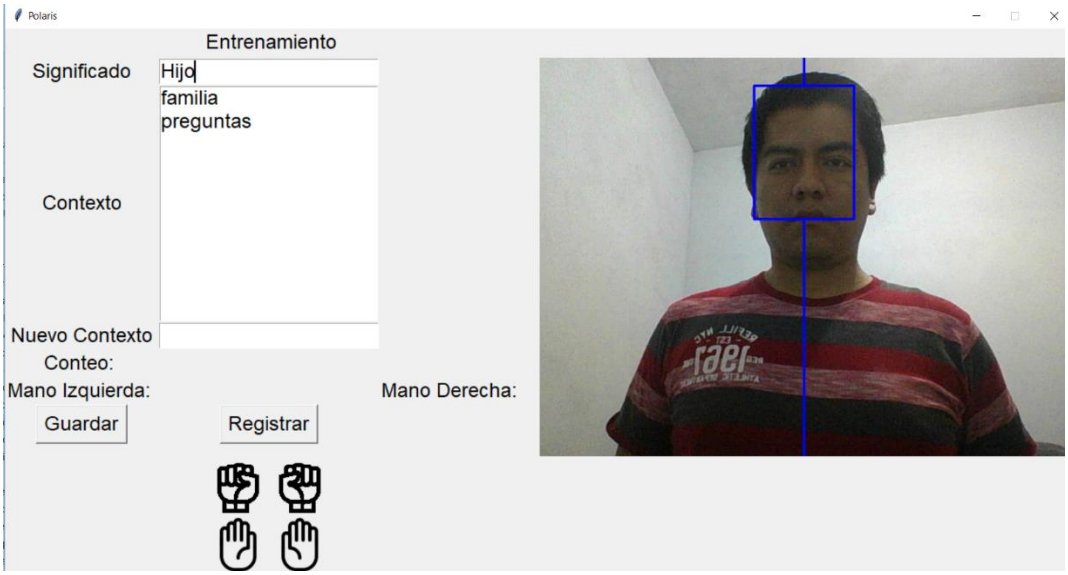



Figura 3.46 Interfaz de creación y selección de contextos o significados.

Se creó la función *saveMeaningContext* en la clase *MainWindows* para registrar los nuevos significados y contextos, asi como la relación entre estos, como se observa en la línea 730, a continuación, se muestra un extracto de esta función:

```
722 def saveMeaningContext(self):
723 
724     value = self.listbox_context_selector.curselection()
725     if len(value) < 1:
726         list_box_value = None
727     else:
```

```
728         list_box_value = self.listbox_context_selector.get(value)
729 
730     r, id_context, self.id_meaning = self.raf.saveMeaningContext(
731         self.new_context_value.get(), self.input_mean_to_save.get(),
732         list_box_value)
733 
734     if r:
735         # If is true We need to start the count down and save the leapmotion
values and images snapshots
736 
737         self.raw_training_data_to_save = list()
738         self.count_down_training_data_getting = 1
739         # Start the count down
740         self.start_count_down_training()
```

Tal como se observa en el extracto anterior se obtienen los valores de los campos correspondientes al contexto y significado y se evalúan en la función *saveMeaningContext*, que forma parte de la clase auxiliar *RepositoryAuxiliarFunctions*, en las líneas 248 a la 256 y cuyo extracto se incluye a continuación:

```
243 def saveMeaningContext(self, context_value, meaning_value, context_selection):
244     r = False
245     context = None
246     meaning = None
247 
248     if not (meaning_value is None or meaning_value is "") and (context_selection
is not None or context_value is not None):
249         r = True
250         # If the context value is empty we need to check the list value
251         if context_value is None or context_value is "":
252             # Call the function to get the id of the context selection value
253             context = self.getContextId(context_selection)
254             if context is None:
255                 r = False
256         else:
257             # Call the funtion to check and save a new context
258             context = self.getNewContextId(context_value)
259 
260         meaning = self.getMeaningId(meaning_value)
261 
266         # we need to related the meaning table and he context table with the
intermediate table
267         self.saveMeaningContextRelation(meaning, context)
268     return r, context, meaning
269 
270 def getContextId(self, context):
271     result = None
272 
273     if context is not None:
274         Base.metadata.bind = self.engine
275         DBSession = sessionmaker(bind=self.engine)
276         session = DBSession()
277         total = session.query(func.count(Context.id)).filter(Context.context ==
context).scalar()
278 
279         if total > 0:
280             result = session.query(Context.id).filter(Context.context ==
```

```
context).scalar()
281     return result
```

Las funciones *getContextId* y *getMeaningId* consultan en la base de datos si el valor enviado por el usuario existe previamente en la base de datos, de esta manera se evita guardar valores repetidos, en caso de que exista previamente, regresa el id del registro en la base de datos, en caso contrario, lo guarda en la base de datos y obtiene el nuevo id generado.

El siguiente paso después de obtener los id correspondientes al contexto y el significado, es iniciar una cuenta regresiva para indicarle al usuario que se grabarán los datos correspondientes a la seña indicada, en la figura 3.47 se observa el conteo.

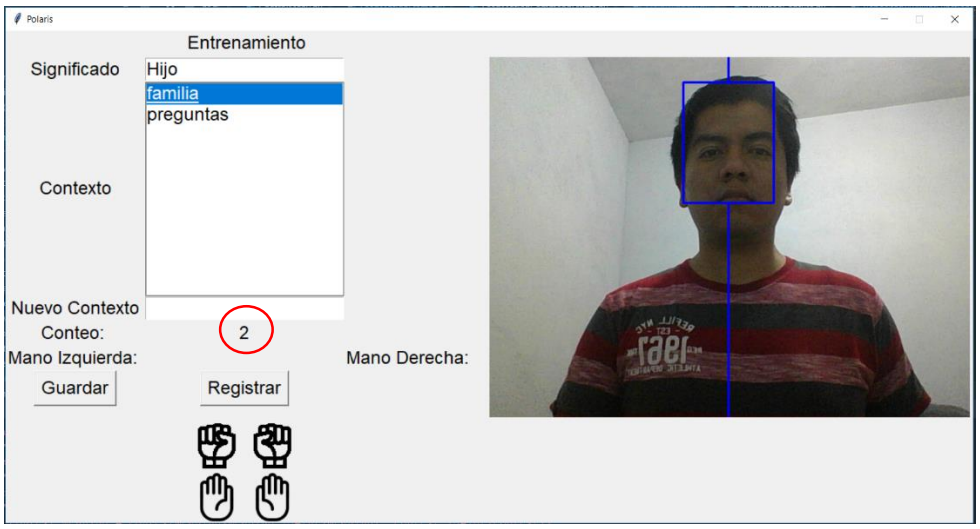



Figura 3.47 Interfaz de entrenamiento con conteo en decremento.

Cuando el conteo finaliza se ejecuta la función *readInformation* de manera recurrente cuatro veces cada 500 milisegundos, ejecutando a su vez la función *get_frame* de la clase *MyVideoCapture* y la función *getFrameData* de la clase *TrainingAuxiliaryFunctions* para obtener la imagen de cuadro actual y los datos del dispositivo LeapMotion, a continuación, se muestra un extracto de la función *getFrameData*:

```
61 def getFrameData(self):
62     frame = self.controlador_leap.frame()
63     data = list()
64     if self.isHandDetected(frame):
65         frame_item = LeapMotionFrame()
66         for hand in frame.hands:
67             # Check if exist two or one hand
68             if len(frame.hands) < 2:
69                 if hand.is_left:
70                     data.append(self.frame_item.generarListaDatosLeap(hand))
71                     # righ hand is None
```

```
72                      data.append(list())
73                  else:
74                      # Left hand is None
75                      data.append(list())
76                      data.append(self.frame_item.generarListaDatosLeap(hand))
77
78              else:
79                  data.append(self.frame_item.generarListaDatosLeap(hand))
80      else:
81          data.append(list())
82          data.append(list())
83
84      return data
```

A su vez la función anterior delega la generación de la estructura de datos correspondientes a la función *generarListaDatosLeap* para guardar los datos de cada dedo dentro de un ciclo *for* que se aprecia en la línea 91 y se muestra a continuación:

```
85 def generarListaDatosLeap(self, hand):
86      fingers = hand.fingers
87      listData = []
88
89      listData.append(hand.direction)
90
91      for finger in fingers:
92          dedo = list()
93
94          dedo.append(finger.type)
95
96          # Get bones
97          for b in range(0, 4):
98              bone = finger.bone(b)
99              dedo.append(bone.next_joint)
100              if b == 3:
101                  dedo.append(bone.direction.to_float_array())
102
103         listData.append(dedo)
104
105     listData.append(hand.palm_position)
106
107     return listData
```

Cuando se tienen los datos estructurados se procede a procesar los cuadros correspondientes a las imágenes almacenadas de forma temporal para buscar los marcadores en forma de pulsera y generar sus coordenadas, para finalmente ejecutar la función *saveFramesTraining* de la clase *RepositoryAuxiliaryFunctions* cuyo extracto es el siguiente:

```
404 def saveFramesTraining(self, centroids, leap_data, session_time, id_meaning):
405     result = False
406     left = None
407     right = None
408
409     for index, c in enumerate(centroids):
410         # print("index: "+str(index))
```

```
411         # For every hand we will save values if is not empty
412         if len(leap_data[index][0][0]) > 0:
413             left = leap_data[index][0][0]
414             print("Tiene mano izquierda")
415 
416         if len(leap_data[index][0][1]) > 0:
417             right = leap_data[index][0][1]
418 
419         result = self.lmf.saveAllFeatures(left, right, c, session_time, index,
id_meaning)
420 
421     return result
```

La función *saveAllFeatures* de la clase *LeapMotionFrame* es la encargada de persistir los datos procesados en la base de datos, guardando cada una de las instancias necesarias en las líneas 1521 a la 1545, es decir, los huesos, dedos y manos correspondientes, se muestra parte del código:

```
1506 def saveAllFeatures(self, left_h, right_h, centroid, session_time, index,
id_meaning):
1507     result = False
1508     lh = None
1509     rh = None
1510     x_one = None
1511     y_one = None
1512     x_two = None
1513     y_two = None
1514     query = session.query(LeapMeaning).filter(LeapMeaning.id == id_meaning)
1515     meaning = query.first()
1516 
1517     if left_h is not None or right_h is not None:
1518         # We check the default values for the hands if are detected or not
1519         if left_h is not None:
1520             # Save the hand
1521             lh = self.saveFeatures(left_h, 0)
1522 
1523         if right_h is not None:
1524             # Save the hand
1525             rh = self.saveFeatures(right_h, 1)
1526 
1527         if len(centroid) > 0:
1528             x_one = centroid[0][0]
1529             y_one = centroid[0][1]
1530 
1531             if len(centroid) > 1:
1532                 x_two = centroid[1][0]
1533                 y_two = centroid[1][1]
1534 
1535     LeapFrameToSave = LeapFrame(right_hand=rh,
1536                                 left_hand=lh,
1537                                 meaning=meaning,
1538                                 secuence_number=index,
1539                                 session_id=session_time,
1540                                 x_one=x_one,
1541                                 y_one=y_one,
1542                                 x_two=x_two,
1543                                 y_two=y_two)
1544     session.add(LeapFrameToSave)
```

```
1545    session.commit()
1546
1547    result = True
1548    return result
```

Se creó la función *loadRepositoryToInterpretation* que obtiene el valor del contexto seleccionado y llama a su vez a la función *getRepositoryForContext* en la línea 629 para inicializar el predictor con los datos correspondientes a los significados en la base de datos restringidos al contexto seleccionado, se incluye un extracto del código correspondiente:

```
621 def loadRepositoryToInterpretation(self):
622
623    value = self.listbox_context_selector_interpretation.curselection()
624    if len(value) < 1:
625        list_box_value = None
626    else:
627        list_box_value = self.listbox_context_selector_interpretation.get(value)
628    if list_box_value is not None:
629        self.interpreting_repository_context, self.current_meanings_values =
self.raf.getRepositoryForContext(list_box_value)
630        # This variable has a list of leapmotionPersistenceBuild
631        for i in self.current_meanings_values:
632
633            self.meanings_avaliable.insert(END, i.meaning + "\n")
634
635        self.meanings_avaliable.configure(state=DISABLED)
636        self.trainNetworkForInterpretation()
637
638
639 def trainNetworkForInterpretation(self):
640    self.ai_current_context =
self.menu_control.trainNetworkInterpretation(self.interpreting_repository_context,
0)
641    self.configure_change_section(4)
```

El total de puntos de las tareas cerradas para este *sprint* fue de cincuenta y seis tal como se observa en la captura de pantalla de la interfaz de Taiga que se muestra en la figura 3.48.

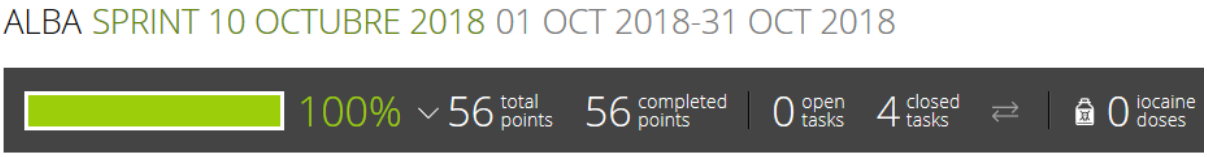



Figura 3.48 Interfaz de Taiga para manejo de tareas de *Sprint* 10 Octubre 2018.

### 3.2.14 *Sprint* 11 Noviembre 2018

Para el onceavo *Sprint* se contó con la siguiente lista de producto (*Product Backlog*) basada en las historias de usuario:

- DMA-01 Desarrollo del módulo de análisis compartido.
- SCE-01Selección de un caso de estudio.
- PCE-01 Realizar pruebas con el caso de estudio seleccionado.

Que a su vez derivaron en la siguiente lista de pendientes del *sprint* (*Sprint Backlog*), junto a sus correspondientes tareas:

- DMA-01 Desarrollo del módulo de análisis compartido.
    - Creación de interfaces de selección de contexto y predicción.
    - Creación de funciones necesarias para interpretar el resultado de las señas según el contexto.

Se crearon las interfaces necesarias correspondientes a los bocetos presentados en las figuras 3.30 y 3.31, cuyo resultado final se muestra en las figuras 3.49 y 3.50.

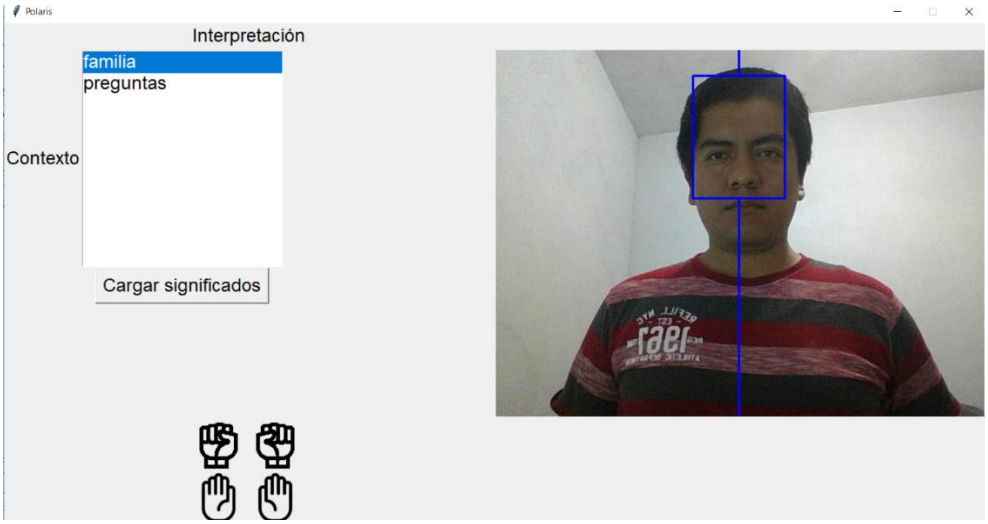



Figura 3.49 Interfaz de selección de contexto.


Figura 3.50 Interfaz de interpretación según vocabulario de contexto disponible

Se creó la función *finalInterpretation* encargada de ejecutar un conteo regresivo para indicar al usuario que está por iniciarse la recolección de datos de la ejecución actual para clasificarla como un significado disponible, según el contexto seleccionado previamente, para ello se reutiliza la función *readInformation* en la línea 664 pero con el parámetro *to_train* con un valor falso para indicar que los datos no serán guardados, sino interpretados, el extracto del código para dicha función se incluye:

```
649 def finalInterpretation(self):
650     # We start the count down to record
651     if self.interpretation_text_countdown.get() == "":
652         self.interpretation_text_countdown.set(3)
653         self.window.after(1000, self.finalInterpretation)
654     else:
655         if int(self.interpretation_text_countdown.get()) > 0:
656 
self.interpretation_text_countdown.set(int(self.interpretation_text_countdown.get())
- 1)
657             self.window.after(1000, self.finalInterpretation)
658         else:
659             self.raw_training_data_to_save = list()
660             self.count_down_training_data_getting = 1
661 
662             self.interpretation_text_countdown.set("Capturando")
663             # We call the method to capture the leap motion and image section for
1 second
664             self.readInformation(False)
```

Una vez recolectados los datos, se envían a la función *getRowStructure* de la clase *RepositoryAuxiliaryFunctions* para obtener una estructura óptima para poder compararlas, el extracto de esta función se muestra a continuación:

```
300 def getRowStructure(self, centroids, leap_data):
301     result = []
```

```
302     temp = list()
303     data = list()
304 
305     # Get all the hands
306     for index, c in enumerate(centroids):
307         if type(leap_data[index][0][0]).__name__ == "Hand":
308             temp.append(leap_data[index][0][0])
309         else:
310             temp.append(list())
311 
312         if type(leap_data[index][0][1]).__name__ == "Hand":
313             temp.append(leap_data[index][0][1])
314         else:
315             temp.append(list())
316         temp.append(c)
317         data.append(temp)
318         temp = list()
319 
320     result = pd.DataFrame(self.lmf.getStructuredData(data),index=[0])
321 
322     return result
```

Para llevar a cabo la clasificación de las señas se analizan cuatro cuadros o instantes de la ejecución guardada, a su vez cada uno de estos cuadros están compuestos por 167 características, correspondientes a los valores de los ejes *x*, *y* y *z* de los huesos de las manos, aportados por el dispositivo *LeapMotion*, además de los cuatro valores correspondientes a las coordenadas de la cuadrícula imaginaria utilizada para clasificar la posición de los marcadores, lo que da un total de 668 características, la función encargada de estructurar estos datos es *getStructuredData* de la clase *LeapMotionFrame*, de la cual se muestra el siguiente extracto:

```
317 def getStructuredData(self, data):
318     final_data = None
319 
320     for index, d in enumerate(data):
321 
322         temp_left = self.buildListFrame(d[0])
323         temp_right = self.buildListFrame(d[1])
324 
325         x_one = None
326         y_one = None
327         x_two = None
328         y_two = None
329         if len(d[2]) > 0:
330             x_one = d[2][0][0]
331             y_one = d[2][0][1]
332 
333             if len(d[2]) > 1:
334                 x_two = d[2][1][0]
335                 y_two = d[2][1][1]
336 
337         if index == 0:
338             final_data = {
339                 'palm_direction_x_r': temp_right[0][0],
340                 'palm_direction_y_r': temp_right[0][1],
341                 'palm_direction_z_r': temp_right[0][2],
```

```
342             }
343
344             # Bones for finger thumb right hand
345
346             final_data['finger_thumb_tip_x_r'] = temp_right[1][5][0]
347             final_data['finger_thumb_tip_y_r'] = temp_right[1][5][1]
348             final_data['finger_thumb_tip_z_r'] = temp_right[1][5][2]
349
350             final_data['metacarpal_thumb_x_r'] = temp_right[1][1][0]
351             final_data['metacarpal_thumb_y_r'] = temp_right[1][1][1]
352             final_data['metacarpal_thumb_z_r'] = temp_right[1][1][2]
353
354             final_data['proximal_thumb_x_r'] = temp_right[1][2][0]
355             final_data['proximal_thumb_y_r'] = temp_right[1][2][1]
356             final_data['proximal_thumb_z_r'] = temp_right[1][2][2]
357
358             final_data['intermediate_thumb_x_r'] = temp_right[1][3][0]
359             final_data['intermediate_thumb_y_r'] = temp_right[1][3][1]
360             final_data['intermediate_thumb_z_r'] = temp_right[1][3][2]
```

Para optimizar y simplificar la búsqueda de valores se crearon cuatro vistas en la base de datos con los nombres *secuence_zero*, *secuence_one*, *secuence_two* y *secuence_three*, cuyo contenido son las 167 caracteristicas para cada uno de los cuatro cuadros y que son utilizadas por la función *getRepositoryForContext* para ejecutar una consulta a la base de datos por medio de la función *read_sql* de la biblioteca pandas, como se observa en la línea 54, el contenido de dicha función se transcribe a continuación:

```
34 def getRepositoryForContext(self, context):
35
36     # Get the context id
37     context_id = self.getContextId(context)
38     # Get the avaliable meaning ids for the current context
39     Base.metadata.bind = self.engine
40     DBSession = sessionmaker(bind=self.engine)
41     session = DBSession()
42     meanings_ids = [r.meaning_id for r in
session.query(ContextMeaning.meaning_id).filter(ContextMeaning.context_id
43
== context_id).all()]
44
45     meanings_values =
session.query(LeapMeaning).filter(LeapMeaning.id.in_(meanings_ids)).all()
46
47     s = text('SELECT * FROM secuence_zero as z '
48              'LEFT JOIN secuence_one as o ON o.session_id_s_one = z.session_id '
49              'LEFT JOIN secuence_two as t ON t.session_id_s_two = z.session_id '
50              'LEFT JOIN secuence_three as tr ON tr.session_id_s_three =  '
51              'z.session_id')
52
53     # To use we need to drop the session_id, meaning, secuence_number and ids
fields
54     repository = pd.read_sql(s, self.conn, index_col='id', params={'meanings':
meanings_ids})
55
56     repository.fillna(value=0, inplace=True)
```

```
57
58     return repository, meanings_values
```

Por último, la función *showClassification* utiliza los datos recolectados y el clasificador entrenado para determinar el significado de la seña realizada como se observa en la línea 835:

```
835 def showClassification(self):
836
837     r = self.ai_current_context.predict(self.current_data_row_for_predict)
838
839     # Search in db the meaning with this id
840     meaning_text = self.raf.getMeaningText(r)
841
842     self.interpretation_text_result.set(meaning_text)
843
844     self.interpretation_text_countdown.set("")
```

El total de puntos de las tareas cerradas para este *sprint* fue de diez tal como se observa en la captura de pantalla de la interfaz de Taiga que se muestra en la figura 3.51.

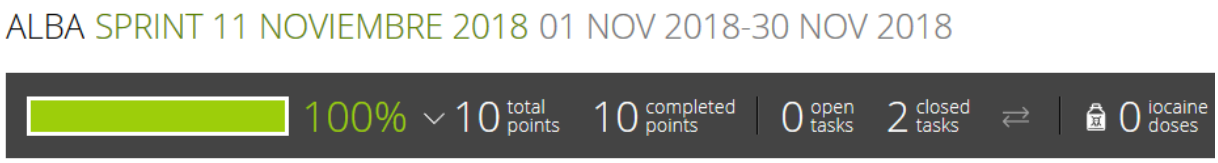



Figura 3.51 Interfaz de Taiga para manejo de tareas de *Sprint* 11 Noviembre 2018.

### 3.2.15 *Sprint* 12 Diciembre 2018

Para el doceavo Sprint se contó con la siguiente lista de producto (*Product Backlog*) basada en las historias de usuario:

- SCE-01Selección de un caso de estudio.
- PCE-01 Realizar pruebas con el caso de estudio seleccionado.

Que a su vez derivaron en la siguiente lista de pendientes del *sprint* (*Sprint Backlog*), junto a sus correspondientes tareas:

- SCE-01Selección de un caso de estudio.
  - Analizar las posibles aplicaciones.
  - Seleccionar un caso de estudio aplicable.
- PCE-01 Realizar pruebas con el caso de estudio seleccionado.
  - Recolección de datos.
  - Ejecución de pruebas.

Dado el déficit de facilidades e intérpretes certificados para la población sorda en el país, las aplicaciones de un intérprete automatizado de lengua de señas mexicana son amplias, por lo que se enumeran algunas de las mismas:

- Intérprete automatizado para consultas médicas.
- Intérprete automatizado para tramites en oficinas de gobierno.
- Intérprete automatizado para información turística.
- Intérprete automatizado para información en terminales aéreas o terrestres.
- Intérprete automatizado como herramienta para tomar declaraciones o denuncias por parte de los organismos de justicia.
- Intérprete para uso en el hogar.
- Intérprete automatizado como herramienta para aprender y reforzar la lengua de señas.

Dado el tiempo disponible para realizar pruebas y las facilidades encontradas en el curso “Comunicación eficaz a través de la lengua de señas mexicana” impartido por el Doctor en educación Mario Rojas Meza, en el DIF de la ciudad de Fortín, se eligió probar el caso de estudio de orientación educativa:

Intérprete automatizado como herramienta para aprender y reforzar la lengua de señas mexicana.

El planteamiento del caso de estudio y los resultados de la ejecución del caso de estudio se explican a detalle en el capítulo cuatro.

El total de puntos de las tareas cerradas para este *sprint* fue de diez tal como se observa en la captura de pantalla de la interfaz de Taiga que se muestra en la figura 3.52.

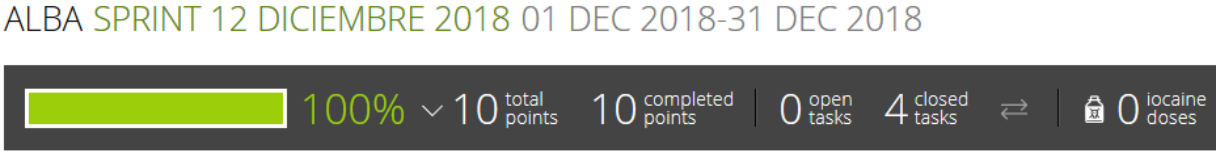



Figura 3.52 Interfaz de Taiga para manejo de tareas de *Sprint* 11 Noviembre 2018.

Al finalizar los doce *sprints* planificados se concluyeron historias de usuario con un total de 408 puntos y una velocidad de *sprint* de 34 puntos, aunque el proyecto inicio con menor velocidad, se logró mejorar el desempeño, tal y como se observa en la gráfica final de pendientes de producto que se muestra en la figura 3.53.

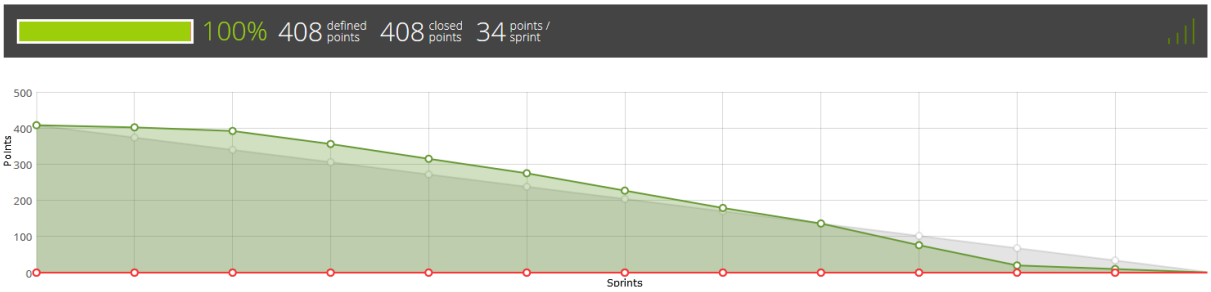



Figura 3.53 Gráfica final de pendientes de producto.

# Capítulo 4. Resultados

En la siguiente sección se detallan los resultados obtenidos, desde la selección del caso de estudio, las características de la recolección de datos y la eficacia del intérprete.

## 4.1 Planteamiento del caso de estudio

Se realizó una búsqueda de instituciones que realizan actividades para la promoción de la lengua de señas mexicana en diversos municipios de la zona centro del estado de Veracruz, incluyendo Orizaba, Nogales, Ciudad Mendoza, Córdoba y Fortín, y se constató que en todos ellos se efectúan diversos eventos de promoción, así como también se lleva a cabo la impartición de cursos para su aprendizaje, sin embargo, el horario disponible de los profesores es limitado, por lo que los alumnos deben esperar a los días de clases para poder practicar o coordinar con otras personas que conozcan la lengua de señas, es por ello que se busca probar el intérprete de lengua de señas mexicana como auxiliar en el aprendizaje y reforzamiento de vocabulario, para que los alumnos lo utilicen con la presencia del profesor o sin ella.

Debido al alto porcentaje de personas con discapacidad auditiva es de suma importancia dar a conocer esta lengua, para que más personas la conozcan y exista una mejora en la comunicación entre la comunidad sorda y los oyentes.

Para la delimitación del caso de estudio se tomó en cuenta la metodología de aprendizaje aplicada en el curso de lengua de señas mexicana que imparte el Doctor Mario Rojas Meza, la cual consiste en el estudio de un conjunto de vocabulario delimitado a una categoría en específico por cada sesión, ya que esta metodología puede adaptarse al uso de los contextos que se han planteado en el diseño y desarrollo del intérprete, por ello se eligió la categoría alfabeto, compuesta específicamente de las primeras doce letras del alfabeto del idioma español y la categoría preguntas compuesta por doce señas.

Este conjunto de señas, se considera óptimo como vocabulario de pruebas debido a los siguientes puntos:

- Las señas correspondientes a la categoría alfabeto:
    - Las señas se realizan con una sola mano.
    - Cuenta con señas estáticas.
    - Cuenta con señas con movimiento.
    - Es el primer tema que se ve en el curso.
- Las señas correspondientes a la categoría preguntas:

- Cuenta con señas que utilizan una mano.
- Cuenta con señas que utilizan dos manos.
- Cuenta con señas que intercalan durante su ejecución entre una y dos manos.
- Cuenta con señas que presentan los dedos en distintas posiciones bien diferenciadas.
- Cuenta con señas que presentan los dedos en posiciones similares.
- Las señas correspondientes a las preguntas forman parte de los primeros significados que se aprenden en la lengua de señas mexicana.

- El aprendizaje del vocabulario necesario para realizar preguntas en conjunto con el alfabeto es de suma importancia en cualquier lenguaje, ya que facilita la tarea de continuar aprendiendo al permitir interactuar con otras personas.

## 4.2 Vocabulario seleccionado para el caso de estudio

Las señas correspondientes a las 12 pociones del alfabeto español se muestran a continuación en la figura 4.1.

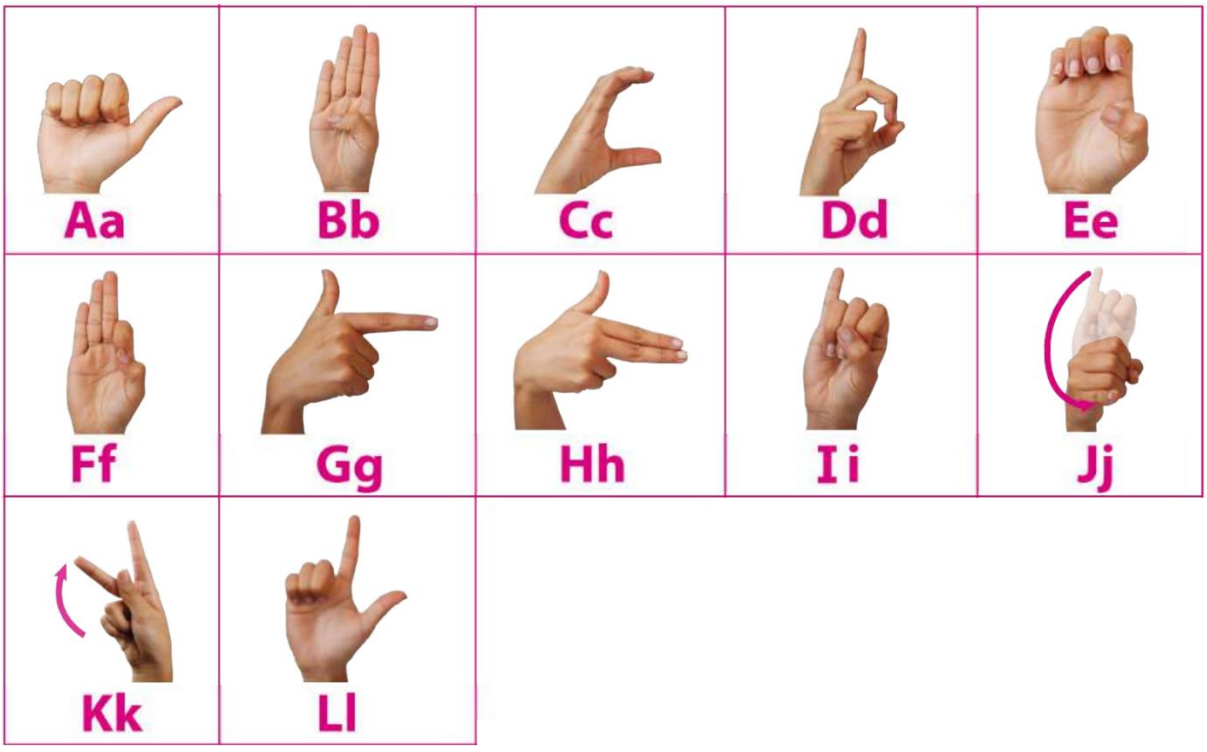



Figura 4.1 Conjunto de señas seleccionadas para el caso de estudio de la categoría alfabeto.

Las doce señas de preguntas correspondientes al vocabulario seleccionado se muestran en la figura 4.2.

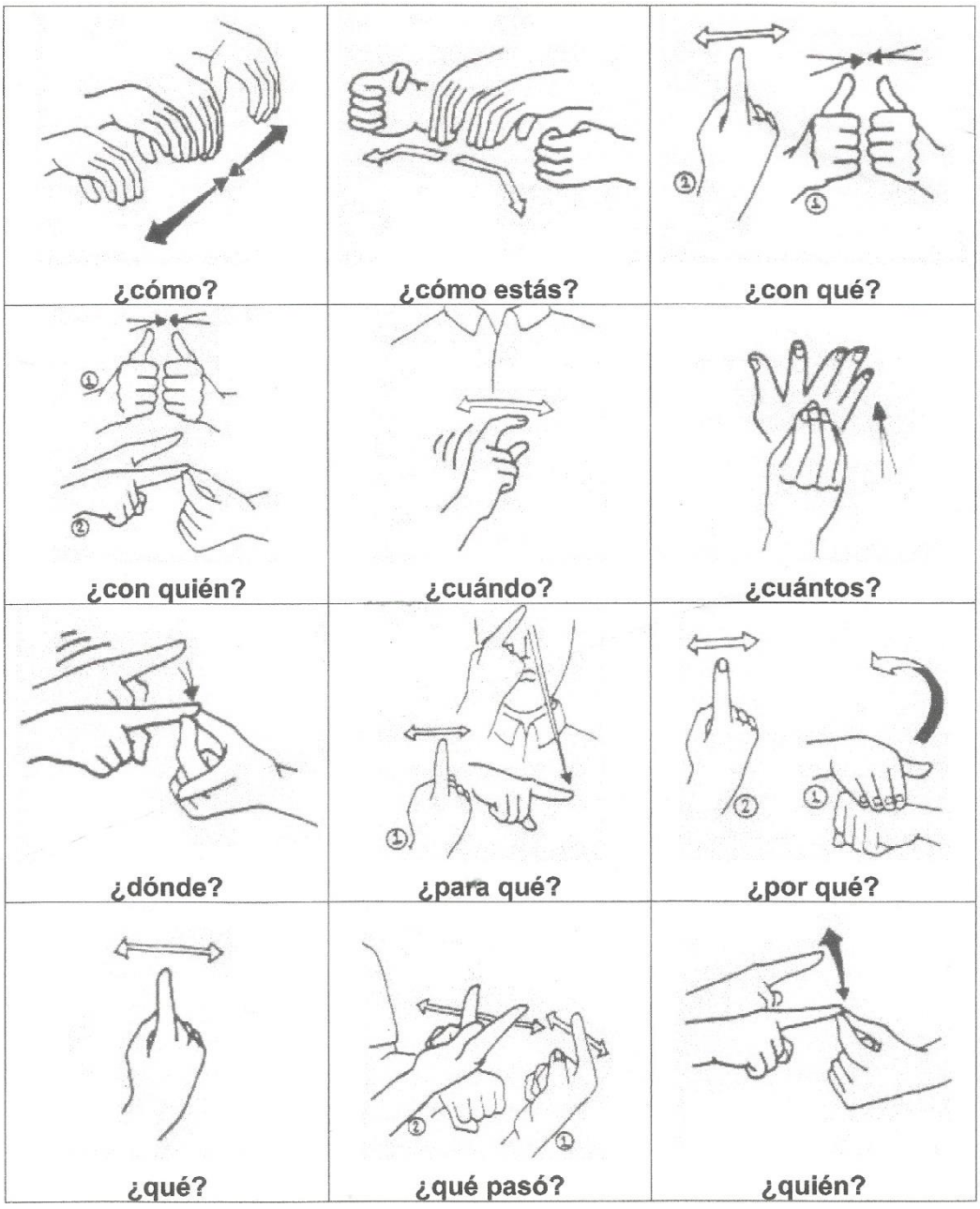



Figura 4.2 Conjunto de señas seleccionado para el caso de estudio de la categoría preguntas.

## 4.3 Características de la recolección de datos para la construcción del repositorio

Se contó con la participación de los alumnos del nivel 2 de lengua de señas mexicana que asisten a tomar clases en la biblioteca municipal “Lic. Antonio M. Quirasco” con las siguientes características:

- 10 alumnos.
- 9 alumnos de sexo femenino.
- Un alumno de sexo masculino.
- El rango de edad de los alumnos va de los 25 a los 50 años.
- Se tomaron quince muestras para cada seña por parte de cada alumno.
- Llevan estudiando la lengua de señas mexicana más de seis meses.
- Son oyentes.

La configuración utilizada para obtener los datos desde el dispositivo LeapMotion y la cámara Web fue la siguiente:

- Computadora Dell Inspiron 5000.
- Procesador i5-7300.
- 8 Gb de memoria RAM.
- Windows 10 de 64 bits.
- Tarjeta gráfica Nvidia Geforce GTX de 4 Gb.
- Python 3.6.5
- SDK LeapMotion Orion versión 3.2.1+45911.
- Dispositivo LeapMotion.
- Cámara web Logitech C525
- Trípode fotográfico de altura ajustable 100 cm.
- Minitrípode de altura ajustable 19 cm.
- Marcadores en forma de pulsera de cartulina color verde.

En la figura 4.3 y 4.4 se aprecia la configuración de los equipos utilizados para la recolección de datos y los marcadores utilizados:

Figura 4.3 Configuración utilizada con los diversos dispositivos utilizados para realizar la recolección de datos.

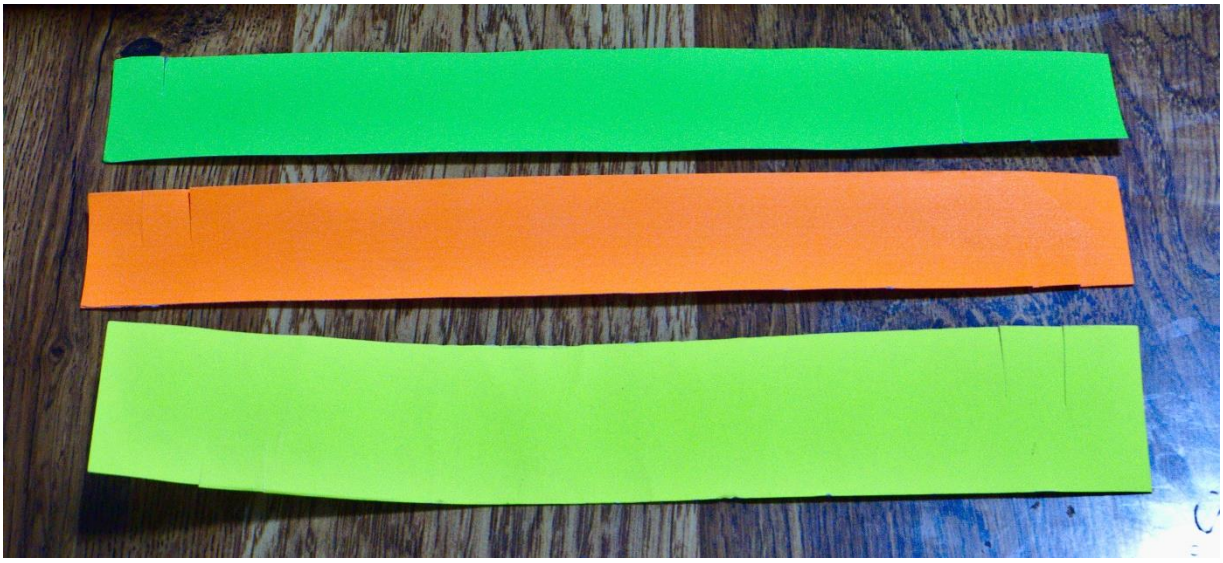


Figura 4.4 Pulseras utilizadas como marcadores para la recolección de datos.

La recolección de los datos además contó con las siguientes características:

- Se realizó en cinco sesiones con fechas distintas.
- La ubicación dentro del área disponible de la biblioteca cambió.
- Las horas a las que se realizó fueron distintas.
- La iluminación disponible varió en cada sesión.

## 4.4 Resultado de las pruebas realizadas al Intérprete de Lengua de Señas mexicana con vocabulario del caso de estudio seleccionado

Se recolectaron diez registros por cada alumno, para cada seña, lo cual genera un total de 3600 registros que alimentan el clasificador que utiliza el intérprete de lengua de señas mexicana para el caso de estudio seleccionado, 1800 registros para cada categoría.

En la figura 4.5 se observan dos ejemplos de la recolección de datos con los alumnos:

Figura 4.5 Recolección de datos con la configuración seleccionada.

Una vez recolectados los datos necesarios que servirán para entregar al clasificador, se procedió a probar el módulo de interpretación, para ello los alumnos ejecutaron los

movimientos frente al dispositivo para que el intérprete indique el significado de la seña ejecutada, tal como se observar en la figura 4.6.

Figura 4.6 Prueba del módulo de interpretación.

Los resultados obtenidos para las pruebas realizadas corresponden a la ejecución de validación cruzada de 10 pliegues y una repetición del experimento 10 veces, el promedio de los resultados obtenidos al realizar las pruebas con el clasificador de máquina de vectores de soporte se muestra en la tabla 4.1. y en la tabla 4.2 y 4.3 se visualiza la matriz de confusión obtenida para el vocabulario seleccionado.

Al realizar la recolección de datos se detectó un caso particular que generaba problemas para la correcta identificación de las pulseras utilizadas como marcadores, se realizaron pruebas con dos colores para los marcadores, amarillo y verde fluorescente, sin embargo, la calibración debió realizarse tres veces hasta conseguir una imagen limpia para los marcadores, se determinó que el problema guardaba relación con el color de piel del alumno y con la configuración de la iluminación disponible, que generaba sombras en la imagen, lo que a su vez ocasionaba una alta variabilidad en los colores detectados, en la figura 4.7 se observa al alumno ejecutando señas durante la recolección de datos.

Tabla 4.1 Resultado de la ejecución de pruebas al módulo de interpretación.

| Seña | Precisión |
|---|---|
| **A** | 85.3 % |
| **B** | 92.6 % |
| **C** | 91.3 % |
| **D** | 97.3 % |
| **E** | 97.3 % |
| **F** | 96 % |
| **G** | 93.3 % |
| **H** | 91.3 % |
| **I** | 94.6 % |
| **J** | 98 % |
| **K** | 98.6 % |
| **L** | 96 % |
| **¿Cómo?** | 100% |
| **¿Cómo estás?** | 84% |
| **¿Con que?** | 96% |
| **¿Con quién?** | 96% |
| **¿Cuándo?** | 100% |
| **¿Cuántos?** | 96% |
| **¿Dónde?** | 96% |
| **¿Para qué?** | 90% |
| **¿Por qué?** | 96% |
| **¿Qué?** | 100% |
| **¿Qué paso?** | 84% |
| **¿Quién?** | 92% |

Tabla 4.2 Matriz de confusión para el vocabulario seleccionado de la categoría abecedario.

| | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** | **J** | **K** | **L** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **A** | 128 | 0 | 0 | 0 | 20 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| **B** | 0 | 139 | 1 | 2 | 2 | 3 | 0 | 1 | 0 | 2 | 0 | 0 |
| **C** | 0 | 0 | 137 | 9 | 0 | 1 | 0 | 0 | 0 | 3 | 0 | 0 |
| **D** | 0 | 0 | 4 | 146 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **E** | 2 | 0 | 0 | 0 | 146 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| **F** | 0 | 6 | 0 | 0 | 0 | 144 | 0 | 0 | 0 | 0 | 0 | 0 |
| **G** | 0 | 0 | 0 | 0 | 0 | 0 | 140 | 8 | 0 | 0 | 2 | 0 |
| **H** | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 137 | 0 | 0 | 10 | 0 |
| **I** | 0 | 1 | 4 | 0 | 0 | 3 | 0 | 0 | 142 | 0 | 0 | 0 |
| **J** | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 147 | 0 | 0 |
| **K** | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 148 | 0 |
| **L** | 2 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 144 |

Tabla 4.3 Matriz de confusión para el vocabulario seleccionado de la categoría preguntas.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1- ¿Cómo?** | 150 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **2- ¿Cómo estás?** | 12 | 126 | 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **3- ¿Con que?** | 0 | 0 | 144 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 0 |
| **4- ¿Con quién?** | 0 | 0 | 0 | 144 | 0 | 3 | 0 | 0 | 0 | 3 | 0 | 0 |
| **5- ¿Cuándo?** | 0 | 0 | 0 | 0 | 150 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **6- ¿Cuántos?** | 0 | 0 | 0 | 0 | 0 | 144 | 0 | 0 | 0 | 6 | 0 | 0 |
| **7- ¿Dónde?** | 0 | 0 | 0 | 0 | 0 | 0 | 144 | 0 | 0 | 0 | 0 | 6 |
| **8- ¿Para qué?** | 0 | 0 | 0 | 0 | 0 | 9 | 0 | 135 | 0 | 6 | 0 | 0 |
| **9- ¿Por qué?** | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 3 | 144 | 0 | 0 | 0 |
| **10- ¿Qué?** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 150 | 0 | 0 |
| **11- ¿Qué paso?** | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 12 | 0 | 6 | 126 | 0 |
| **12- ¿Quién?** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 12 | 0 | 138 |

Figura 4.7 Alumno identificado con tono de piel que requirió repetir la calibración inicial para detectar correctamente los marcadores.

# Capítulo 5. Conclusiones y Recomendaciones

En la siguiente sección se incluyen las conclusiones obtenidas al finalizar el proyecto y las recomendaciones para trabajos futuros que se obtuvieron con la implementación del intérprete.

## 5.1. Conclusiones

Como se observa en las tablas 4.2 y 4.3, la clasificación con el algoritmo de máquina de vectores de soporte presenta en general una precisión superior al 90% en las pruebas realizadas, lo cual representa un nivel de aceptación alto para las señas estáticas del alfabeto de la lengua de señas mexicana y para el conjunto de señas de la categoría preguntas, con ello se confirma la utilidad y buen desempeño del control LeapMotion, la biblioteca OpenCV y el conjunto de características seleccionadas.

De igual forma se identificaron una serie de posiciones únicas e inherentes al vocabulario de la LSM, cuyas características presentan problemas para la correcta identificación por parte del dispositivo, debido a la posición de los dedos, cuando estos quedan colocados en medio de otros o se cruzan con la palma de la mano, en la sección recomendaciones se indican posibles soluciones a explorar en posteriores desarrollos, estos problemas pueden verse reflejados en los resultados de la interpretación de la letra A y las preguntas ¿Cómo estás? Y ¿Qué paso?, donde se puede inferir que se ven afectadas debido a la similitud con otras señas o debido a que incluyen posiciones donde las manos se sobreponen una sobre otra, lo que ocasiona que el dispositivo LeapMotion no proporcione un seguimiento correcto.

Durante la aplicación de pruebas de interpretación en tiempo real con los alumnos, se identificaron una serie de problemas relacionados con la calibración del intérprete para la correcta detección de los marcadores de color utilizados, que, si bien el diseño actual fue capaz de solventar, se detectó una posibilidad de mejora, por lo que en la siguiente sección se indican posibles correcciones en el diseño de la interfaz que permitan mejorar la experiencia de uso por parte del usuario final al solventar estos problemas.

Si bien existen investigaciones anteriores sobre la aplicación del control LeapMotion para la identificación de significados en diversas lenguas de señas, no se encontró ninguno relacionado específicamente con la lengua de señas mexicana, por lo que la presente tesis permite validar la funcionalidad y efectividad de la arquitectura propuesta, que incluye el uso conjunto del dispositivo de captura de movimiento en 3D y el tratamiento de imágenes para la recolección de datos que puedan ser analizados y procesados mediante técnicas de inteligencia artificial, finalmente la prueba del

dispositivo con usuarios finales permitió validar e identificar patrones de comportamiento por parte del usuario al interactuar con el programa, generando una valiosa retroalimentación que sirve como base a las recomendaciones propuestas que se detallan en la siguiente sección, se estima que estas mejoras permitirán elevar la calidad de la clasificación y mejoren las posibilidades de una implementación del intérprete en un ambiente de uso cotidiano, ya sea en el caso de estudio presentado o en cualquiera de los otros casos mencionados.

De igual forma se comprobó una buena recepción por parte de los usuarios finales, los cuales mostraron interés y entusiasmo al utilizar una herramienta tecnológica para reforzar su conocimiento, expresando sentirse más motivados a practicar el lenguaje.

Al tener en cuenta lo anteriormente expresado, se considera que el proyecto puede seguir mejorando y podrá ser implementado en más áreas de uso, además de servir como una herramienta atractiva para la promoción de la lengua de señas mexicana.

## 5.2. Recomendaciones

Como resultado de las pruebas realizadas al intérprete de lengua de señas mexicana para el caso de estudio seleccionado se consideran las siguientes recomendaciones para continuar con su desarrollo a futuro, buscando mejorar la calidad de la interpretación realizada y mejorar la experiencia de uso por parte de los usuarios:

- Rediseñar la interfaz para permitir una verificación constante de la calidad del procesamiento de imágenes con respecto a los marcadores utilizados, para agilizar el proceso de detección de problemas, tal como se mencionó en la sección 4.4 y en la figura 4.7, con el fin de mejorar la calidad de los datos recolectados.
- Se propone como posible solución a la pérdida de seguimiento de las manos en las señas donde estas se sobreponen y el dispositivo LeapMotion no tiene una clara imagen de los dedos o de las manos, explorar el funcionamiento de un segundo dispositivo LeapMotion en paralelo, ubicado en otra posición para tener un segundo ángulo de visión, ya que el fabricante lanzó el 20 de diciembre de 2018 una actualización experimental que permite el uso de más de un dispositivo [54].
- Una mejora que puede explorarse para mejorar el desempeño del intérprete de lengua de señas mexicana es modificar las áreas de interés manejadas, para permitir segmentar de mejor forma la posición de las manos y brazos, subdividiendo la imagen en secciones correspondientes a las áreas donde interactúan las manos en el lenguaje, por ejemplo, el área del pecho, la cabeza, al lado de los hombros.

- Agregar soporte para clasificadores de características en cascada *Haar* que permitan identificar y dar seguimiento a la posición de las manos y de esta forma eliminar el uso de marcadores visuales, tales como las pulseras utilizadas en esta versión.
- Realizar pruebas con algoritmos de aprendizaje profundo o algoritmos genéticos para validar la selección de características.
- Probar diversas técnicas para la selección de momentos relevantes durante la grabación de datos y la interpretación para mejorar la precisión de la clasificación.
- Realizar pruebas con un segundo caso de estudio, para obtener retroalimentación de los usuarios con respecto a la experiencia de uso en diversos contextos.
- Agregar un módulo de control automático de altura para los trípodes utilizados mediante servomotores, que permitan mejorar la calidad de la información recolectada de manera automática.
- Utilizar marcadores en forma de pulsera que tengan el mismo color en toda su superficie y sea de un material poco reflexivo para mejorar el análisis de imágenes.

## Productos Académicos

Roberto Hernández-De-La-Luz, Ma. Antonieta Abud Figueroa, Lisbeth Rodríguez Mazahua, Ulises Juárez Martínez, Celia Romero Torres.

*Prototipo de intérprete de Lengua de Señas Mexicana usando el control Leap Motion.*

Research in Computing Science, ISSN 1870-4069.

Estado: Presentado

Registro de derechos de autor del “Intérprete de Lengua de Señas Mexicana con vocabulario configurable según el contexto” ante INDAUTOR

Estado: En trámite

## Referencias


[1] Asociación de Academias de la Lengua Española, *Diccionario de la lengua española (23.ª edición)*. Madrid: Espasa, 2014.

[2] N. Pelayo y A. Cabrera, *Lenguaje y comunicación: conceptos básicos, aspectos teóricos generales, características, estructura, naturaleza y funciones del lenguaje y la comunicación*. Los Libros de el Nacional, 2001.

[3] J. F. S. Dauder, *Antropología para inconformes: Una antropología abierta al futuro*. Ediciones Rialp, 2006.

[4] M. Infante, *Sordera: mitos y realidades*. Editorial de la Universidad de Costa Rica, 2005.

[5] "Lengua de Señas Mexicana (LSM)", *Lengua de Señas Mexicana (LSM)*, 19-sep-2017. [En línea]. Disponible en: https://www.gob.mx/conadis/articulos/lengua-de-senas-mexicana-lsm?idiom=es. [Consultado: 19-sep-2017].

[6] M. E. Serafín de Fleischmann y R. Pérez González, *Manos con voz Diccionario de Lengua de Señas Mexicana*, I. México: Consejo Nacional para Prevenir la Discriminación, 2011.

[7] INDEPEDI CDMX, *Diccionario de Lengua de Señas Mexicana de la Ciudad de México*. INDEPEDI CDMX, 2017.

[8] V. Sutton, *Lessons in Sign Writing: Textbook*. Deaf Action Committee for Sign Writing, 1995.

[9] R. Lahoz-Beltrá, *Bioinformática: simulación, vida artificial e inteligencia artificial*. Diaz de Santos, 2004.

[10] I. Steinwart y A. Christmann, *Support Vector Machines*. Springer New York, 2008.

[11] A. Kaehler y G. Bradski, *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*. O'Reilly Media, 2016.

[12] M. J. de la Fuente Aparicio y T. C. Cano, *Aplicaciones de las redes de neuronas en supervisión, diagnosis y control de procesos*. Equinoccio, 1999.

[13] G. Gan, C. Ma, y J. Wu, *Data Clustering: Theory, Algorithms, and Applications*. Society for Industrial and Applied Mathematics, 2007.

[14] R. F. López y J. M. F. Fernández, *Las Redes Neuronales Artificiales*. Netbiblo, 2008.

[15] M. I. CHACÓN MURGUÍA, *PERCEPCIÓN VISUAL - Aplicada a la robótica:* Editorial Ink, 2016.

[16] M. A. Crespo, *Dirección Cinematográfica: Manual Avanzado de Aprendizaje Creativo*. 2013.

[17] Lin Shao, "Hand movement and gesture recognition using Leap Motion Controller". 2016.

[18] "About OpenCV Library". [En línea]. Disponible en: http://opencv.org/about.html. [Consultado: 01-oct-2017].

[19] L. P. Coelho y W. Richert, *Building Machine Learning Systems with Python - Second Edition*. Packt Publishing, 2015.

[20] "Scikit-learn", *scikit-learn Machine Learning in Python*. [En línea]. Disponible en: https://scikit-learn.org/. [Consultado: 01-ene-2018].

[21] "pandas: Python Data Analysis Library", *Python Data Analysis Library*. [En línea]. Disponible en: https://pandas.pydata.org. [Consultado: 01-ene-2018].

[22] "NumPy", *NumPy*. [En línea]. Disponible en: http://www.numpy.org. [Consultado: 01-ene-2018].

[23] J. Sutherland, *Scrum: El arte de hacer el doble de trabajo en la mitad de tiempo*. Oceano, 2016.

[24] "Taiga", *Taiga.io*, 24-may-2018. [En línea]. Disponible en: https://taiga.io/. [Consultado: 24-may-2018].

[25] "Encuesta Nacional de la Dinámica Demográfica 2014", *Encuesta Nacional de la Dinámica Demográfica 2014*, 2014. [En línea]. Disponible en: http://www.beta.inegi.org.mx/proyectos/enchogares/especiales/enadid/2014/. [Consultado: 19-sep-2017].

[26] "En México se hacen ciegos ante los sordos", *En México se hacen ciegos ante los sordos*. [En línea]. Disponible en: http://sipse.com/mexico/sordos-discapacidad-gobierno-mexico-224324.html. [Consultado: 19-sep-2017].

[27] H. Fuyang, S. Zelong, X. Quiang, S. Yim Binh, T. Wai Lan, y W. Xiaogang, "Real-time Sign Language Recognition using RGBD Stream: Spatial-Temporal Feature Exploration", en *Proceedings of the 2nd ACM symposium on Spatial user interaction*, Holulu, 2014, pp. 149–149.

[28] S. Chao y Tianzhu, "Latent Support Vector Machine Modeling for Sign Language Recognition with Kinect", *J. ACM Trans. Intell. Syst. Technol. TIST*, vol. VI, núm. 2, pp. 1–20, 2015.

[29] R.-C. J. Rafael, S.-B. Héctor, M.-G. M. Eleazar, G. Molero-Castillo, y J. A. Ortega-Carrillo, "Designing an interaction architecture by scenarios for Deaf people", en *Proceedings of the XVII International Conference on Human Computer Interaction*, Salamanca, 2016, pp. 1–2.

[30] J. Barragán *et al.*, "Spanish sign language interpreter for mexican linguistics", *J. Comput. Sci. Technol.*, vol. 13, núm. 1, pp. 32–37, abr. 2013.

[31] L. E. Potter, J. Araullo, y L. Carter, "The Leap Motion controller: a view on sign language", en *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration*, Adelaide, 2013, pp. 175–178.

[32] M. Simos y N. Nikolaidis, "Greek sign language alphabet recognition using the leap motion device", en *Proceedings of the 9th Hellenic Conference on Artificial Intelligence*, Thessaloniki, 2016, pp. 1–4.

[33] R. B. Mapari y G. Kharat, "American Static Signs Recognition Using Leap Motion Sensor", en *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies*, Udaipur, 2016, pp. 1–5.

[34] M. Huenerfauth, E. Gale, B. Penly, M. Willard, y D. Hariharan, "Comparing Methods of Displaying Language Feedback for Student Videos of American Sign Language", en *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility*, Lisbon, 15, pp. 139–146.

[35] J. Shang y J. Wu, "A Robust Sign Language Recognition System with Multiple Wi-Fi Devices", en *Proceedings of the Workshop on Mobility in the Evolving Internet Architecture*, Los Angeles, 2017, pp. 19–24.

[36] C. S. Bianchini, F. Borgia, P. Bottoni, y M. D. Marsico, "SWift: a SignWriting improved fast transcriber", en *Proceedings of the International Working Conference on Advanced Visual Interfaces*, Capri Island, 2012, pp. 390–393.

[37] M. Boulares y M. Jemni, "Mobile sign language translation system for deaf community", en *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility*, Lyon, 2012, pp. 1–4.

[38] S. Ghanem, C. Conly, y V. Athitsos, "A Survey on Sign Language Recognition Using Smartphones", en *Proceedings of the 10th International Conference on PErvasive Technologies Related to Assistive Environments*, Island of Rhodes, 2017, pp. 171–176.

[39] A. Miller, J. Malasig, B. Castro, V. L. Hanson, H. Nicolau, y A. Brandão, "The Use of Smart Glasses for Lecture Comprehension by Deaf and Hard of Hearing Students", en *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, Denver, 2017, pp. 1909–1915.

[40] J. Gugenheimer *et al.*, "The Impact of Assistive Technology on Communication Quality Between Deaf and Hearing Individuals", en *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, Portland, 2014, pp. 669–682.
[41] J. Shin y C. M. Kim, "Character Input System using Fingertip Detection with Kinect Sensor", en *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, Odense, 2016, pp. 74–79.
[42] F. Milicchio y M. Prosperi, "Accessible Tourism for the Deaf via Mobile Apps", en *Proceedings of the 9th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, Corfu, 2016, pp. 1–7.
[43] J. Cervantes, F. García-Lamont, J. H. Santiago, J. Espejel Cabrera, y A. Trueba, "Clasificación del lenguaje de señas mexicano con SVM generando datos artificiales", *Rev. VÍNCULOS*, vol. 10, núm. 1, pp. 328–341, abr. 2013.
[44] J. Espejel-Cabrera, J. Cervantes, A. Trueba Espinosa, y J. SergioRuiz Castilla, "Reconocimiento de Lenguaje de Señas Usando Características Geométricas y SVM", *Congr. Int. Comput. E Informática 2012*, sep. 2012.
[45] "Generating a Python 3.3.0 Wrapper with SWIG 2.0.9", *LeapMotion*. .
[46] "Kivy", *Kivy - Open source Python library for rapid development of applications*. [En línea]. Disponible en: https://kivy.org/. [Consultado: 15-mar-2018].
[47] "PyForms", *PyForms*. [En línea]. Disponible en: https://github.com/UmSenhorQualquer/pyforms. [Consultado: 15-mar-2018].
[48] "Graphical User Interfaces with Tk". [En línea]. Disponible en: https://docs.python.org/3/library/tk.html. [Consultado: 15-ene-2018].
[49] "Pure Python MySQL Client", *PyMySQL*. [En línea]. Disponible en: https://github.com/PyMySQL/PyMySQL. [Consultado: 15-abr-2018].
[50] "Camera Images - Leap Motion Python SDK v3.2", *Camera Images - Leap Motion Python SDK v3.2*. [En línea]. Disponible en: https://developer-archive.leapmotion.com/documentation/python/devguide/Leap_Images.html?proglang=python. [Consultado: 08-may-2018].
[51] "Logitech C525 Portable HD Webcam", *Logitech C525 Portable HD Webcam*. [En línea]. Disponible en: https://www.logitech.com/es-mx/product/hd-webcam-c525. [Consultado: 20-may-2018].
[52] "Contour finding-scikit-image", *Contour finding*. [En línea]. Disponible en: http://scikit-image.org/docs/dev/auto_examples/edges/plot_contours.html#sphx-glr-auto-examples-edges-plot-contours-py. [Consultado: 18-jun-2018].
[53] "Image Thresholding", *Image Thresholding*. [En línea]. Disponible en: https://docs.opencv.org/3.4.1/d7/d4d/tutorial_py_thresholding.html. [Consultado: 20-jun-2018].
[54] "Experimental Release #2: Multiple Device Support", *Experimental Release #2: Multiple Device Support*. [En línea]. Disponible en: http://blog.leapmotion.com/multiple-devices/. [Consultado: 20-dic-2018].