UNIVERSIDAD DE SONORA
DIVISIÓN DE INGENIERÍA
Departamento de Ingeniería Industrial

APLICACIÓN DE INTELIGENCIA AMBIENTAL PARA APOYAR EN LA MEDICACIÓN DE ADULTOS MAYORES

Reporte de Prácticas Profesionales

PRESENTA:
KARINA JANETH MONTERO LÓPEZ

INGENIERO EN SISTEMAS DE INFORMACIÓN
Director: Dra. Raquel Torres Peralta

CONTENIDO

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

En la reglamentación de Universidad de Sonora se tiene contemplado que en todos los planes de estudio se incluyan actividades de vinculación con el sector social o productivo con el propósito de complementar la formación de los estudiantes a través de la aplicación de los conocimientos obtenidos en las diversas materias de las carreras. Así, la carrera de Ingeniería en Sistemas de Información, del Departamento de Ingeniería Industrial, incluye en su plan de estudios las prácticas profesionales con valor 20 créditos, que son equivalentes a 340 horas.

Para cumplir con este requisito se solicitó un espacio para colaborar como practicante en el proyecto de investigación "Aplicación de inteligencia ambiental para apoyar en la medicación de adultos mayores" a cargo del Dr. Rene Navarro en la Universidad de Sonora, a lo cual el responsable del proyecto respondió positivamente.

1.1. EXPLICACIÓN DEL PROYECTO

Este proyecto surge como una propuesta de investigación de un área desprotegida, ya que se estima que cerca del 60% de los adultos mayores tienen problemas para adherirse al régimen de medicación prescrito por su médico. Este es un problema grave si consideramos que el 86% de las personas de edad avanzada padecen de al menos una enfermedad crónica que requiere medicación[1]. Las consecuencias de la falta de adherencia a la medicación incluyen pérdida de control de la enfermedad, costos elevados para los sistemas de salud pública por el aumento en ingresos y readmisiones hospitalarias, y lo principal, la perdida de la calidad de vida del adulto mayor.

Para atender esta problemática se propone desarrollar un sistema basado en despliegues ambientales para apoyar la adherencia a la medicación en adultos mayores. Un despliegue ambiental es una tecnología de AmI que tiene como objetivo presentar información relevante a los usuarios mediante modalidades ambientales (e.g., Representaciones abstractas de la información basadas en imágenes, patrones de luz y color, sonido y/o sus combinaciones), que resulten fáciles de interpretar por los usuarios.

En base a esta definición se formuló y desarrollo una propuesta para atender este problema la cual podemos ver a continuación:

El proyecto desarrollado consiste en un espejo inteligente que apoyara a los usuarios con la toma de sus medicamentos diarios, entendiendo como usuario al adulto mayor.

- La aplicación debe mostrar al usuario información relevante de uso diario como la fecha, hora y el clima, esto para brindar el contexto necesario a la toma de medicamentos.

- Debe contar con una sección principal donde muestre al usuario la lista de medicamentos que este debe de tomar en ese momento o próximos al momento de consulta.
- Se debe lanzar una alarma o recordatorio de las tomas de medicamentos.
- El sistema debe de contar con la opción de consultar el historial de medicamentos del día y el estado de cada toma.
- El sistema debe lanzar alarmas en caso de que el usuario tenga una cita médica registrada.
- El usuario operará el espejo por medio de comandos de voz sin necesidad de interacción manual o directa.
- Son opciones de interacción los medios de comandos por voz.
- Opcionalmente se plantea que el sistema pueda contar con un módulo de reconocimiento facial para personalizar el servicio cuando sea utilizado por varios usuarios.
- Se debe seguir siempre una metodología orientada al diseñado centrado en el usuario.

1.2. OBJETIVOS

1.2.1. Objetivo general

El objetivo general de este proyecto es implementar y evaluar la usabilidad de un visualizador ambiental para apoyar la medicación en adultos mayores.

1.2.2. Objetivos específicos

- Diseñar el prototipo de un visualizador ambiental para apoyar la medicación en adultos mayores.
- Implementar un prototipo de un visualizador ambiental para apoyar la medicación en adultos mayores.
- Realizar la evaluación de la usabilidad de visualizador ambiental para apoyar la medicación en adultos mayores

2. DESCRIPCIÓN DEL CONTEXTO

La Universidad de Sonora es una institución pública autónoma que tiene como misión formar, en programas educativos de calidad y pertinencia, a profesionales integrales y competentes a nivel nacional e internacional, articulando la docencia con la generación y aplicación del conocimiento, la difusión de la cultura y la extensión de los servicios, para contribuir al desarrollo sustentable de la sociedad. Así, la Universidad de Sonora, como parte de su responsabilidad ante la sociedad, está obligada a contribuir a la solución de los problemas de su entorno. Es en el departamento de ingeniería industrial de la universidad de sonora donde surge la propuesta del desarrollo del proyecto abordado en este documento, bajo la dirección del Dr. Rene Navarro Hernández.

2.1. EQUIPAMIENTO E INSTALACIONES

Se puso a disposición para el desarrollo del proyecto un cubículo en el edificio 5O segunda planta del departamento de ingeniería industrial, el cual contaba con las herramientas e infraestructura, como conexión a internet, mobiliario, etc., necesaria para desempeñar las actividades de desarrollo, implementación y pruebas del sistema.

2.2. ENTORNO DONDE SE UBICA LA UNIDAD RECEPTORA

La organización se encuentra en un edificio del Departamento de Ingeniería Industrial en la Universidad de Sonora, el cual abarca un ambiente social universitario, con aplicaciones de sistemas para distintos sectores, enfocados al sector empresarial y al apoyo de organismos como la misma alma máter.

2.3. NORMATIVIDAD DE LA UNIDAD RECEPTORA

Con lo que respecta a los lineamientos que se proponen dentro del ambiente de la institución donde se realizaron las prácticas profesionales, se incluyen generalmente las reglas que se estipulan por la Universidad de Sonora, que además se anexan reglas con respecto al buen uso del equipo de cómputo y las instalaciones.

3. FUNDAMENTO TEÓRICO DE LAS HERRAMIENTAS Y CONOCIMIENTOS APLICADOS

3.1. INTELIGENCIA AMBIENTAL

La inteligencia ambiental (AmI) y el Internet de las Cosas (IoT, por sus siglas en inglés), estas están asociadas a tecnologías emergentes y aplicaciones de software destinadas a poblar el entorno cotidiano de las personas con artefactos inteligentes y asistentes personales cooperativos que proveen servicios en forma flexible y proactiva [2].

El bajo consumo de energía, la comunicación inalámbrica, nuevas formas de interacción, la miniaturización de componentes electrónicos, la movilidad y la capacidad de censar el ambiente para obtener información útil sobre el contexto de una actividad, son factores que permiten el diseño de sistemas y aplicaciones de cómputo embebidas en el medio ambiente.

3.2. DISEÑO DE INTERFACES CENTRADO EN EL USUARIO

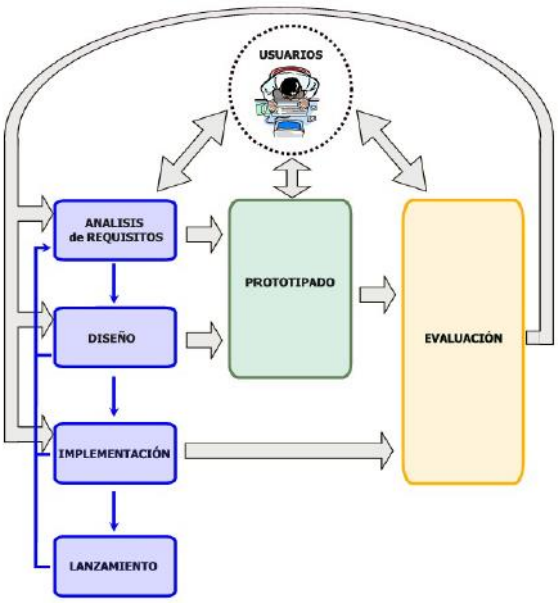
El Diseño Centrado en el Usuario es una filosofía de diseño que tiene por objeto la creación de productos que resuelvan necesidades concretas de sus usuarios finales, consiguiendo la mayor satisfacción y mejor experiencia de uso posible con el mínimo esfuerzo de su parte.

Toma forma como un proceso en el que se utilizan una serie de técnicas multi disciplinares y donde cada decisión tomada debe estar basada en las necesidades, objetivos, expectativas, motivaciones y capacidades de los usuarios.

La mayoría de los procesos que hacen Diseño Centrado en el Usuario suponen las siguientes etapas:

- Conocer a fondo a los usuarios finales, normalmente usando investigación cualitativa o investigación cuantitativa.
- Diseñar un producto que resuelva sus necesidades y se ajuste a sus capacidades, expectativas y motivaciones.
- Poner a prueba lo diseñado, normalmente usando test de usuarios.

La construcción de un sistema interactivo implica un proceso cíclico de diseño, desarrollo y evaluación. La realimentación que proporciona la evaluación sobre el diseño es fundamental para refinar y pulir aspectos que son muy dependientes de los usuarios finales (el factor humano) una vez que el sistema se ha puesto en marcha[3].



3.3. HARDWARE

3.3.1. Raspberry Pi 3

Raspberry Pi es un computador de placa reducida, computador de placa única o computador de placa simple (SBC) de bajo costo desarrollado en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas. La raspberry pi 3 modelo B fue lanzada en el año 2016, con procesador Quad-Core de la compañía Broadcom de 1.20GHz, posee RAM de 1GB, Incluye Wi-Fi y Bluetooth (4.1 Low Energy) sin necesidad de adaptadores.

3.3.2. Asus Tinker Board

Su tamaño físico y la asignación de pines GPIO están diseñados para ser compatibles con la Raspberry Pi 2 y modelos posteriores. La primera placa lanzada cuenta con 4K de video, 2 GB de RAM incorporada, Gigabit Ethernet y un procesador Rockchip RK3288 funcionando a 1.8 GHz.

3.4. PLATAFORMAS Y HERRAMIENTAS DE DESARROLLO

3.4.1. Git

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente.

3.4.2. Javascript

JavaScript (JS) es un lenguaje ligero e interpretado, orientado a objetos con funciones de primera clase, más conocido como el lenguaje de script para páginas web, pero también usado en muchos entornos sin navegador, tales como node.js o Apache CouchDB. Es un lenguaje script multi-paradigma, basado en prototipos, dinámico, soporta estilos de programación funcional, orientada a objetos e imperativa.

3.4.3. Node.JS

Node.js® es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. Node.js usa un modelo de operaciones E/S sin bloqueo y orientado a eventos, que lo hace liviano y eficiente. El ecosistema de paquetes de Node.js, npm, es el ecosistema más grande de librerías de código abierto en el mundo.

3.4.4. Electron.Js

Electron.js es una plataforma para desarrollar aplicaciones de escritorio usando tecnologías web (HTML, CSS y JavaScript) creada y mantenida por Github. Electron.js funciona creando dos tipos de procesos, el proceso main y el proceso renderer. El primero es un proceso de Node.js, este es su proceso principal, este proceso ayuda a comunicar con el SO y realizar distintas acciones o efectos. El segundo (renderer) es un proceso de Chromium, con una diferencia, este

Chromium tiene un Node.js incorporado y acceso a todos sus módulos y los que se instalen con npm.

3.4.5. JSON

JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript, Standard ECMA-262 3rd Edition - Diciembre 1999. JSON es un formato de texto que es completamente independiente del lenguaje, pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

3.4.6. Snowboy

Snowboy es un motor de detección de palabras clave (hotwork, keywork) altamente personalizable que se integra en tiempo real y siempre escucha (incluso cuando no está conectado) compatible con Raspberry Pi, (Ubuntu) Linux y Mac OS X.

Una palabra clave (también conocida como palabra de activación o palabra de activación) es una palabra clave o frase que la computadora escucha constantemente como una señal para desencadenar otras acciones.

3.4.7. API

La interfaz de programación de aplicaciones, abreviada como API del inglés: Application Programming Interface,1 es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas de programación.

3.4.7.1. Google Calendar API

La API de Google Calendar permite mostrar, crear y modificar eventos de calendarios, así como trabajar con muchos otros objetos relacionados, como calendarios o controles de acceso. Esto se logra usando llamadas RESTful y bibliotecas de cliente en varios lenguajes de programación como Java, PHP, .NET, JavaScript, NodeJs, Ruby, Python, Go, Android y iOS.

3.4.7.2. Google Cloud Speech API

La API Speech de Google Cloud permite convertir audio en texto aplicando potentes modelos de redes neuronales en una API fácil de usar. La API reconoce más de 80 idiomas y variantes, lo que ayuda a gestionar una clientela internacional. Puede transcribir el texto que los usuarios dictan al micrófono de una aplicación, habilitar el control por voz o transcribir archivos de audio, entre muchas otras funciones. Es posible reconocer el audio subido en la solicitud e integrarlo al almacenamiento de audio de Google Cloud Storage.

4. DESCRIPCIÓN DETALLADA DE LAS ACTIVIDADES REALIZADAS

Las actividades se realizaron durante el periodo comprendido entre el 4 de septiembre y el 25 de noviembre del 2017, de lunes a sábado en un horario de 12 a 17 horas. En general la dinámica de trabajo fue la siguiente:

A partir de las instrucciones emitidas por el asesor del proyecto, se realizaba una propuesta y planificación de las actividades a desarrollar, propuesta que era retroalimentada y validada por el asesor del proyecto. Parte de las actividades se desarrollaban en la oficina, equipada con acceso a internet y el hardware requerido para el desarrollo de las actividades.

Las actividades se realizaban la mayor parte del tiempo de forma individual o en conjunto con el asesor del proyecto cuando estas así lo requerían.

Las actividades concretas que se desarrollaron se enlistan a continuación para cada una de las tres etapas del periodo de realización de las prácticas profesionales.

4.1. PRIMERA ETAPA

Al inicio del proyecto se llevaron a cabo una serie de reuniones para definir las especificaciones de la aplicación de inteligencia ambiental a desarrollar, se definió el hardware necesario para lograr la interacción del usuario y se plantearon los requerimientos del sistema a integrar con el espejo inteligente.

Posteriormente se llevaron a cabo una serie de investigaciones sobre las aplicaciones y tecnologías existentes que se han utilizado para desarrollar espejos inteligentes y se analizaron los lenguajes de programación que ofrecían una buena opción de integración con la Raspberri Pi 3, siendo este último el hardware propuesto para montar el sistema.

De esta investigación surgió una presentación que incentivo la discusión para definir la forma de desarrollo del sistema, siento la seleccionada la aplicación Magic Mirror 2, que es una aplicación open source disponible en Github, con un enfoque modular que permite la integración de módulos y la reutilización de los existentes. Magic Mirror 2 está basado en el framework Electron.JS y es soportada por Raspberry pi 2 y 3, por lo

cual, dados todos sus beneficios y la gran comunidad con la que cuenta fue la opción seleccionada.

Una vez definida la forma de desarrollo se procedió a hacerse una definición más clara de los tipos de interacción con el sistema, se definieron los puertos de entrada y salida requeridos y se verifico el funcionamiento de estos una vez integrados en la Raspberry. A la par de esto se creó una cuenta de Google Calendar para pruebas que permita agendar las tomas de los medicamentos por fecha y hora, razón de toma y descripción de los medicamentos.

También se hizo una propuesta de diseño de la interfaz de usuario, utilizando herramientas del diseño centrado en el usuario, haciendo énfasis en el despliegue de la información de los medicamentos y se hizo una planificación detallada de la estructura del proyecto, definiéndose los tiempos de desarrollo del sistema.

Con la metodología de desarrollo clara, se procedió a instalar las herramientas necesarias para ejecutar el sistema Magic Mirror 2 en un ordenador personal utilizado para desarrollar el sistema. Las herramientas instaladas fueron el sistema de control de versiones Git, el entorno de ejecución para Javascript Node.js, el editor de texto Sublime Text y se tomaron en cuenta algunas cuestiones referentes a Electron.Js. Después se descargó por consola el repositorio de Magic Mirror 2, disponible en Github, y se realizó un análisis detallado de los módulos a reutilizar, utilizando el editor de texto Sublime Text, en la figura 1 podemos ver la estructura del proyecto.

Figura 1. Estructura del proyecto.

Se procedió a realizar la configuración de estos en el archivo config.js, comenzando con la implementación de la librería de lenguaje español al sistema y su pertinente configuración, ya que el sistema no lo incluía por default.

Para dar contexto al sistema se conservó el módulo de reloj y clima actual, este último modulo se tuvo que configurar para que muestre el clima de la ciudad en cuestión. Para ello se tuvo que crear una cuenta en openweathermap para obtener acceso a la información de su API, se creó una key de acceso y se analizó la documentación para hacer uso de esta, como se muestra en la figura 2.

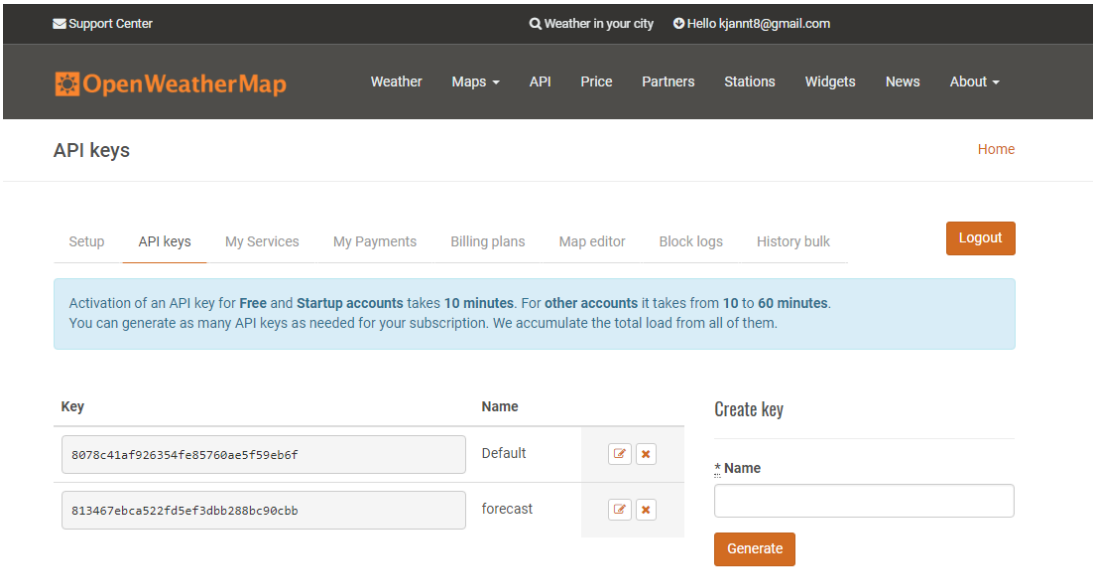


Figura 2. API keys de openweathermap.

Con esta información se configuro el módulo por default currentweather, en el archivo config.js, como se muestra en la figura 3.

```
58          {
59              module: "currentweather",
60              position: "top_right",
61
62              config: {
63                  onlyTemp: true,
64                  appendLocationNameToHeader: true,
65                  location: "Hermosillo, Sonora",
66                  locationID: "4004898", //Location ID from http://openweathermap.org/help/city_list
67                  appid: "8078c41af926354fe85760ae5f59eb6f" //openweathermap.org API key.
68              }
69          },
```

Figura 3. Configuración de modulo clima actual.

4.1.1. Módulo Calendario de Medicamentos

Después se procedió a implementar el módulo calendario, para ello primeramente se crearon dos tipos de calendario Medicamentos y Citas y se fijaron una serie de alarmas, en la cuenta de Google Calendar, como lo vemos en la figura 4, estas sirvieron a modo de prueba.

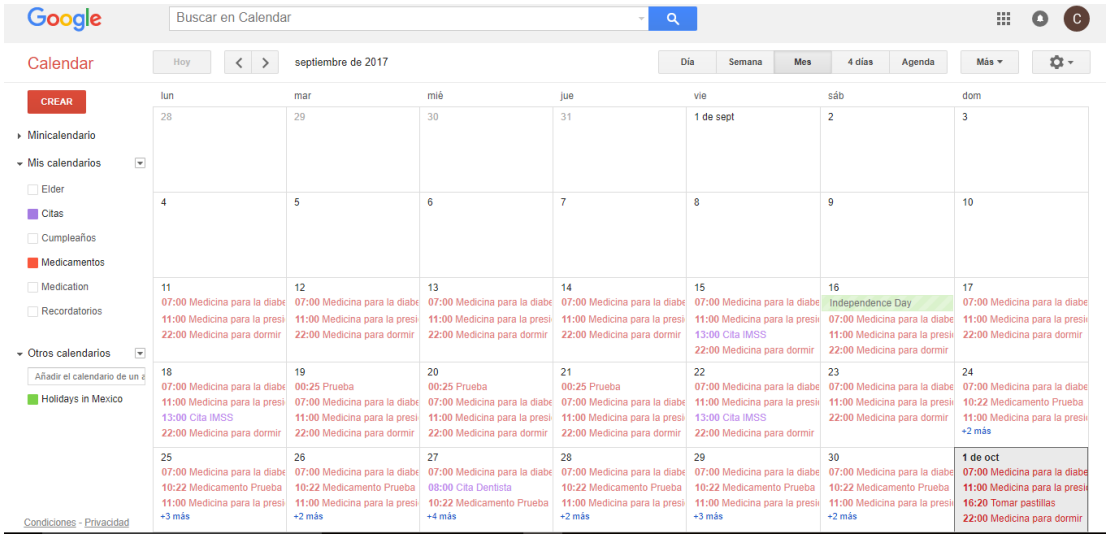


Figura 4. Alarmas médicas configuradas en Google Calendar.

Se determino que para el uso del sistema solo era necesario recabar la información del calendario, por ello se utilizó la dirección privada del calendario y con esta información se configuro el módulo calendario, figura 5.

```
    {
        module: "calendar",
        header: "Medicamentos del día",
        position: "top_left",

        config: {
            maximumNumberOfDays: 1,
            fetchInterval: 60000, // se actualiza cada minuto
            broadcastEvents: true,
            //maximumEntries: 5,

            calendars: [
                {
                    //Calendario de medicamentos
                    symbol: 'calendar',
                    url: 'webcal://calendar.google.com/calendar/ical/oijboakef5g5083at9j7csqa4g%40gr
                },
                {
                    // Calendario de citas
                    symbol: 'calendar-check-o',
                    url: 'https://calendar.google.com/calendar/ical/6uu3hlb94rbf1cu0rh1tfrav00%40gro
                }
            ]
        }
    },
```

Figura 5. Configuración de modulo calendario.

A la par de todas estas actividades, se realizaron una serie de pruebas de visualización para seleccionar la mejor opción de espejo a utilizar y determinar sus dimensiones, en la figura 6 podemos ver un prototipo de espejo que fue desechado por no proveer la nitidez deseada.

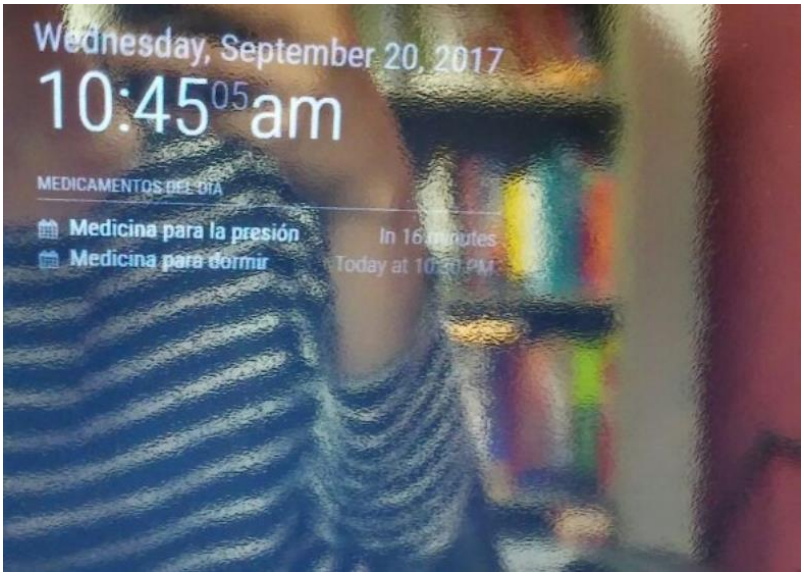


Figura 6. Prototipo de espejo.

4.1.2. Modulo Alerta de Medicamentos

Con el calendario listo, se procedió a desarrollar el módulo de alarmas, Magic Mirror 2 ya contaba con un módulo de despliegue de alarmas a través de la comunicación entre módulos que se envían notificaciones usando Socket.io de Electron.JS, para hacer esto se le tiene que enviar un mensaje que contenga, en formato html, el título y el mensaje a desplegar en el tiempo deseado. Para ello se creó el módulo MMMAM-MedicalAlert (siguiendo la nomenclatura recomendada, Iniciales del proyecto-Nombre del módulo) que contiene el archivo JavaScript a implementar y las carpetas con audios mp3 e iconos a desplegar en formato png, como vemos en la figura 7.

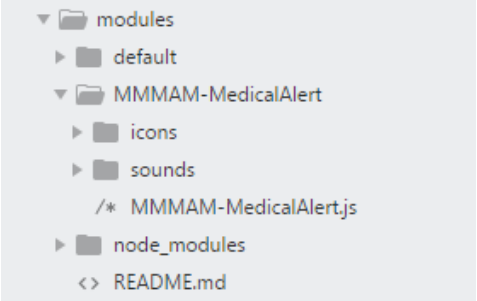


Figura 7. Estructura de módulo de alertas médicas.

Este módulo implementa las funciones necesarias para recibir notificaciones del módulo calendario con la lista de tomas de medicamentos, establecer la siguiente alarma, definir el formato html a enviar al módulo alert, verificar una alarma y resetearla. Además, contiene la lógica necesaria para identificar entre un medicamento y una cita médica, los resultados de esto los podemos ver en las figuras 8 y 9.

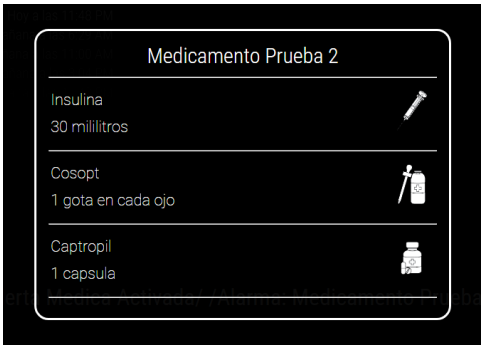


Figura 8. Display alerta médica.

Así mismo define las variables necesarias para lanzar un sonido de alarma y gestionar los iconos a enviar en el mensaje y define un método para gestionar un historial de tomas.

Figura 9. Display de cita médica.

El resultado del software desarrollado durante la primera etapa se puede apreciar en la figura 10 y el despliegue de alarmas en las figuras 8 y 9.

Figura 10. Display de sistema.

Durante toda esta etapa se estuvieron haciendo las pruebas pertinentes y se estuvieron gestionando las versiones a través de la herramienta de gestión de versiones Github.

4.2. SEGUNDA ETAPA

Durante la segunda etapa se realizaron algunas mejoras de la interfaz de usuario, se implementó un módulo de asistente médico por comandos de voz y se solventaron algunos problemas técnicos.

En un principio se continuó trabajando con el módulo calendario y el módulo de alertas para implementar mejoras gráficas que fueron requeridas, en si los cambios consistieron en:

- ✓ Actualización de iconos de medicamentos desplegados en las alertas, se agregaron colores.
- ✓ Mostrar todas las tomas médicas del día actual, sin importar que estas hayan pasado.
- ✓ Se optó por quitar el degradado para facilitar la lectura a los usuarios.
- ✓ Mostrar las horas de las tomas a modo de lista.
- ✓ Implementar un control estilo checklist para mayor claridad y gestión de los medicamentos.

El último punto se implementó desarrollando funciones que permitieran controlar por medio de iconos y colores las tomas médicas, el resultado es el mostrado en la figura 11.

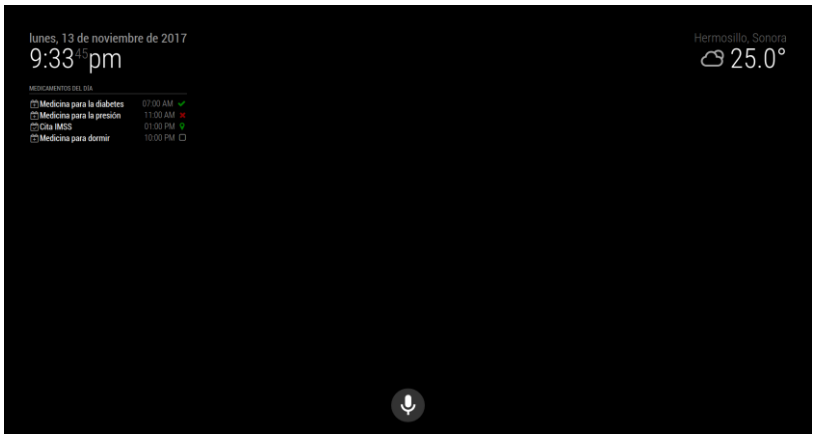


Figura 11. Actualización interfaz módulo calendario.

Así mismo los cambios de diseño en los iconos desplegados por las alertas, se pueden ver en la figura 12.

Figura 12. Cambio de iconos de módulo de alertas.

Con estos módulos actualizados y antes de continuar con el desarrollo del módulo de comandos por voz, se hicieron algunas pruebas instalando el sistema en una Raspberry Pi 3 con sistema operativo Raspbian para probar el rendimiento del sistema. Para nuestra sorpresa, el sistema tuvo algunos problemas gráficos para mostrar las animaciones de los despliegues de las alarmas. En lo que se trabajaba en actualizaciones y verificaciones de librerías y hardware del sistema operativo Raspbian, se probó el sistema en un dispositivo Tinker con un sistema operativo Debian Jessie, en este dispositivo no se tuvo ningún problema con el sistema, por lo cual se continuo con el desarrollo del sistema a la par que se trabajaba en las cuestiones técnicas de los dispositivos de hardware.

4.2.1. Módulo Comandos de voz

Un elemento esencial en la interface es el reconocimiento de comandos de voz. En esta etapa encaminé mis esfuerzos en mejorar esta función.

Primeramente, se evaluaron las opciones para el módulo comandos por voz, se determinó continuar utilizando los servicios de Google, en este caso la API Google Cloud Speech, para reconocimiento de voz a texto y de texto a voz. Así mismo, se determinó que por ser esta API un servicio de paga, con un periodo de prueba gratuito y limitado, se utilizaría un hotword (palabra clave) para activar al asistente médico, evitando así que el sistema esté realizando llamadas a la API cada vez que capte voz y de esta manera ahorrar recursos tanto de rendimiento del sistema como económicos.

El servicio de palabra clave seleccionado para programar las llamadas a la API de Google fue Snowboy. Snowboy es un motor de detección de palabras claves, altamente personalizable que se integra en tiempo real y siempre está escuchando de manera local. Además de ser gratuito, es compatible con Raspberry Pi, Linux y Mac OS X.

Analizando los módulos disponibles para reconocimiento de voz dentro de la familia Magic Mirror 2, se optó por reutilizar el módulo MMM-Assistant. Este módulo integra los servicios de Google Assistant API, Google Cloud Speech API y Snowboy. Mas sin embargo por las características del servicio de Snowboy fue necesario migrar el entorno de desarrollo, hasta ahora Windows 10, a un sistema Linux, siendo el seleccionado Ubuntu 16.04.

En la plataforma Ubuntu se realizaron las configuraciones pertinentes para continuar con el desarrollo del módulo de reconocimiento de comandos por voz, se realizaron las siguientes acciones como se listan:

- ✓ Se hicieron las actualizaciones necesarias del sistema por consola.

- ✓ Se instalaron las últimas versiones para Linux de Node.JS y GIT.
- ✓ Se instaló el editor de texto Atom.
- ✓ Se clono el proyecto de Github con los módulos desarrollados, se instalaron dependencias y se realizaron configuraciones pertinentes.

Continuando con el desarrollo del Asistente Médico por comandos de voz, se descargó e instaló en la carpeta de modules del proyecto, el módulo anteriormente nombrado, MMM-Assistant, el cual para fines de este proyecto renombramos a MMMAM-Assistant y se siguieron las instrucciones de instalación y configuración disponibles en la documentación. Estas inician probando dispositivos de entrada y salida de audio e instalando librerías, requeridas e instalando dependencias en el módulo MMMAM-Assistant. Posteriormente fue necesario crear la conexión con Google Cloud Speech y Google Assitant, para ello se utilizó una cuenta de Google diferente a la del módulo calendario, esto para separar la gestión de información de los medicamentos de la API del Speech. Para obtener acceso a los recursos y servicios de Google fue necesario crear un proyecto en Console Cloud de Google, activar las APIS de Google Assitant y Google Cloud Speech (se pueden ver las APIS activadas en la figura 13), así como realizar las configuraciones de permisos pertinentes en el proyecto.

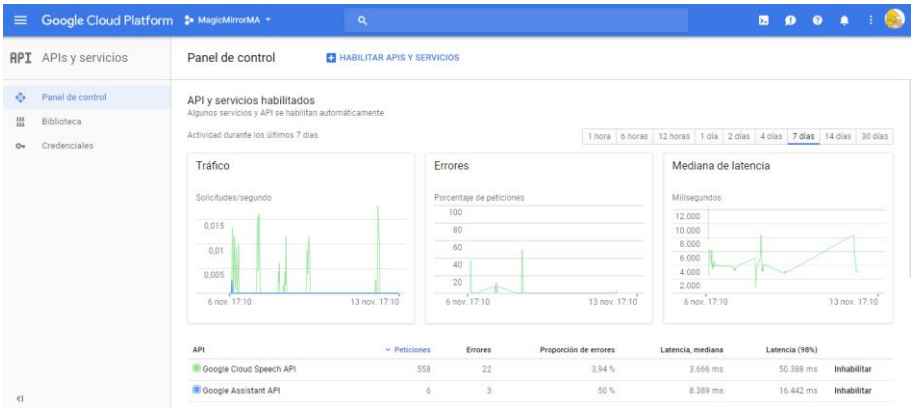


Figura 13. APIs Activadas para proyecto MagicMirrorAM y su consumo.

Lo siguiente, con nuestro modulo instalado, fue configurar el módulo en el archivo `config.js` del sistema MagicMirrorAM y realizar las configuraciones de idioma necesarias.

Se decidió cambiar el hotword por uno más fácil de pronunciar por hispanohablantes. Se sugirió que el nombre del asistente fuera EMMA, por la facilidad de pronunciación y por ser la abreviación del término en ingles *External Memory Aid.*

Para esto se generó un archivo pmdl personalizado con la voz del usuario y los datos de rango de edad y sexo en el sitio oficial de Snowboy, como se muestra en la figura 14. Gracias a este archivo, la detección de la palabra clave se realiza sin necesidad de conexión a internet,

ajustando las variables de sensibilidad al nivel de ruido en el ambiente, este en una escala 0-1, siendo 1 el máximo de ruido.

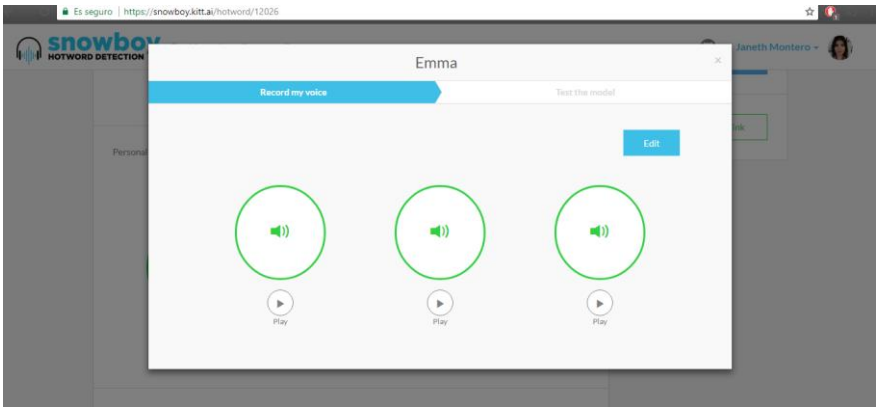


Figura 14. Configuración del archivo Hotword.

En las primeras pruebas, aun ajustando la sensibilidad al ruido de la habitación, la palabra clave se detectaba muy fácilmente, aunque la palabra no estuviera en contexto, por lo cual se determinó agregar otra palabra a la palabra clave, la nueva hotword propuesta fue *Oye Emma,* Esta palabra clave permitió continuar con las pruebas y el desarrollo sin mayores problemas.

4.2.1.1. Desarrollo de comandos de voz

Continuando con el diseño del módulo se plantearon los siguientes comandos por voz y funcionalidad para la interacción mínima entre el sistema y el usuario:

Comandos:

- ✓ **“Dime el siguiente medicamento”**: Dice el nombre de la toma y en cuanto tiempo se debe realizar.
- ✓ **“Dime el detalle del siguiente medicamento”**: Muestra una alerta con la información de la siguiente toma, y dice el nombre, cuanto falta y el detalle de los medicamentos.
- ✓ **“Actualiza la lista de medicamentos”**: Actualiza la lista del calendario.
- ✓ **“Oculta el mensaje”**: Si hay una alerta desplegada en la ventana, la oculta.
- ✓ **“Ya tomé mi medicamento”**: actualiza el estatus del medicamento en el historial y oculta la alerta, si esta activada.
- ✓ **“Dime el clima”**: lee el clima.

Funcionalidad:

- ✓ Leer la alerta cuando se despliega.
- ✓ Respuesta verbal de los comandos, por confirmación o detalles solicitados.

Para lograr implementar esta funcionalidad se tuvieron que implementar métodos en los módulos involucrados para recibir la información requerida y se gestionó a partir del envió de notificaciones entre módulos.

La forma de agregar un comando al diccionario de comandos del módulo consiste en registrarlo en un array de comandos para posteriormente mandarlos a registrar. Cuando se detecta un comando se llama a la función que ejecutara la funcionalidad requerida.

Para que el Asistente de una respuesta verbal se pueden implementar varios métodos del objeto handler. Los utilizados fueron: response y reply.

El primero lee el string de retorno y despliega la información en una alerta, el segundo solo lee el texto. En la mayoría de los casos fue necesario implementar otros métodos en los módulos para recibir el texto de detalle de los medicamentos o una respuesta como en el caso del clima. Además, en estas llamadas se debe enviar un objeto de configuración que contenga el idioma y otras configuraciones.

Con el comando "*Ya tomé mi medicamento*" se actualiza un array interno que sirve de historial para las tomas del día, este array se gestiona en el módulo MMMAM-CalendarDay, queda pendiente implementar métodos para actualizar el estatus en Google Calendar y obtener mejoras en la convergencia de los datos.

En la figura 15 podemos observar un prototipo de espejo seleccionado para montar el espejo inteligente, quedando pendiente para la siguiente etapa montar el espejo inteligente e implementar el sistema para pruebas.

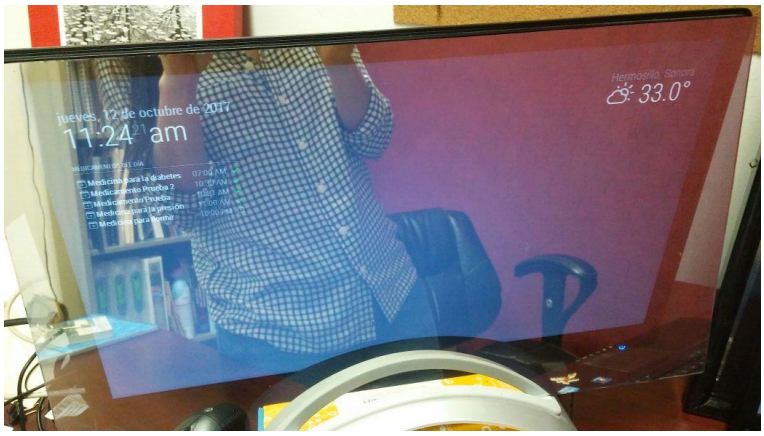


Figura 15. Prototipo de espejo seleccionado.

A lo largo de esta etapa se continuaron gestionando las actualizaciones y versiones de los módulos por medio del controlador de versiones Github, de igual forma se estuvieron realizando las pruebas pertinentes de integración y rendimiento entre los módulos.

El resultado de esta etapa fue la implementación de un módulo de comandos por voz que, aunque cumple con las funcionalidades mencionadas, requiere mejoras, ya que aún tiene problemas cuando la red está demasiado lenta, requiere implementar un método para autoajustar la sensibilidad de la hotword y hace falta implementar un método para actualizar el estatus de la toma medica en Google Calendar.

4.3. TERCERA ETAPA

Durante la tercera etapa se pulieron algunos detalles de los módulos desarrollados, se realizó una propuesta para las etapas posteriores del desarrollo del proyecto de espejo inteligente y se realizaron algunas actividades relacionadas con la difusión del proyecto.

Esta etapa comenzó con la adaptación del módulo calendario de medicamentos para actualizar el estado de la toma medica en Google Calendar, anteriormente, el módulo recuperaba la información del calendario de Google, mas no realizaba modificaciones en el.

Para tener acceso a la API de Google Calendar se utilizó el mismo proyecto de Console Cloud de Google creado en la etapa anterior para el Módulo Asistente Médico. De igual forma se activó la API de Google Calendar y se realizaron configuraciones pertinentes.

Se continuo con el desarrollo de una clase, con apoyo de Node.Js, para implementar la funcionalidad necesaria que permite actualizar el estado de la toma. Para esto último se utilizó el campo *location* del evento del calendario de medicamentos, ya que no estaba siendo utilizado. El desarrollo de este último paso brindo mayor convergencia en los datos de la lista de medicamentos.

Con esto se dio por terminado el desarrollo de modulo Asistente Médico por comandos de voz. Las siguientes actividades realizadas durante el proyecto de prácticas profesionales estuvieron encaminadas a la planificación de las siguientes etapas del proyecto.

Se realizo una propuesta de planificación de las etapas posteriores del proyecto, esta propuesta consiste en la ampliación de la funcionalidad del espejo inteligente a tres módulos más:

- Sensor de movimientos: permitirá ahorrar recursos en los otros módulos.
- Reconocimiento facial: permitirá personalizar el servicio brindado por el espejo.
- Lector de medicamentos: una alternativa al asistente médico para actualizar el estado de las tomas.

Y el desarrollo de una aplicación móvil para administrar y gestionar el espejo:

- Control remoto: permitirá configurar y gestionar el espejo inteligente.
- Gestión de tomas de medicamentos: permitirá crear, programar, modificar y eliminar tomas de medicamentos.
- Historial de tomas médicas: mostrara el histórico de las tomas medicas anteriores y mostrara estadísticas de adherencia a medicamentos.

Para ello se propone seguir utilizando un enfoque modular que permita el desarrollo de las aplicaciones de forma escalable y fácil de probar. Esta propuesta puede verse más a detalle en el diagrama de arquitectura disponible en el anexo 1.

Así mismo se participó en actividades de diseño para un poster que permitirá la divulgación científica del proyecto, este poster puede ser consultado en el anexo 2.

A la par de estas actividades se estuvo implementando el sistema del espejo inteligente en el dispositivo Tinker y se realizaron actualizaciones y configuraciones pertinentes para la utilización de las APIS de Google.

Algunos elementos para armar el prototipo de espejo inteligente se pueden ver en las figuras 16 y 17.

Figura 16. Dispositivo Tinker durante configuración.

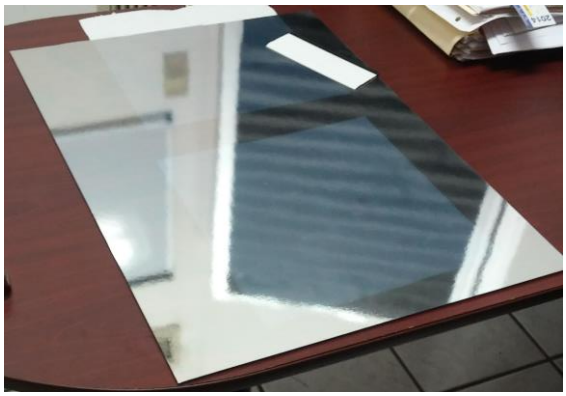


Figura 17. Espejo doble cara.

Durante todo el desarrollo se estuvieron realizando las pruebas necesarias para garantizar la funcionalidad y usabilidad de la aplicación del espejo inteligente, se realizaron las adaptaciones necesarias en los módulos cuando fue necesario y se realizaron mejoras que permitieron mejorar el rendimiento del sistema. Igualmente se depuro y comento el código para mayor legibilidad. Durante la última etapa de implementación en la Tinker se realizaron las pruebas de rendimiento necesarias. Igualmente, durante toda la etapa se estuvo documentación el

avance del sistema y se redactó un artículo científico referente a los módulos del espejo inteligente desarrollados y presentados en este documento.

El resultado del prototipo de espejo inteligente hasta el alcance de este documento aún no se encuentra armado por completo, quedan pendientes detalles del diseño físico del espejo y la integración de los componentes en la placa, podemos observar el display del sistema en las figuras 18 y 19.

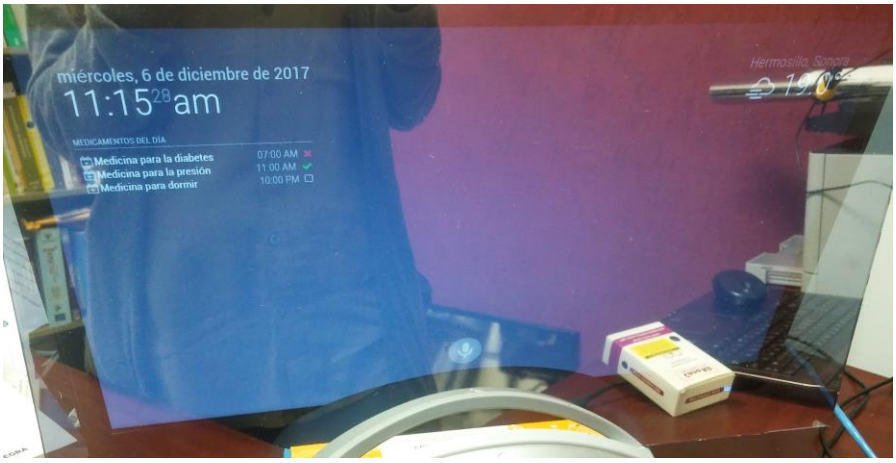


Figura 18. Prototipo de espejo inteligente - Display.

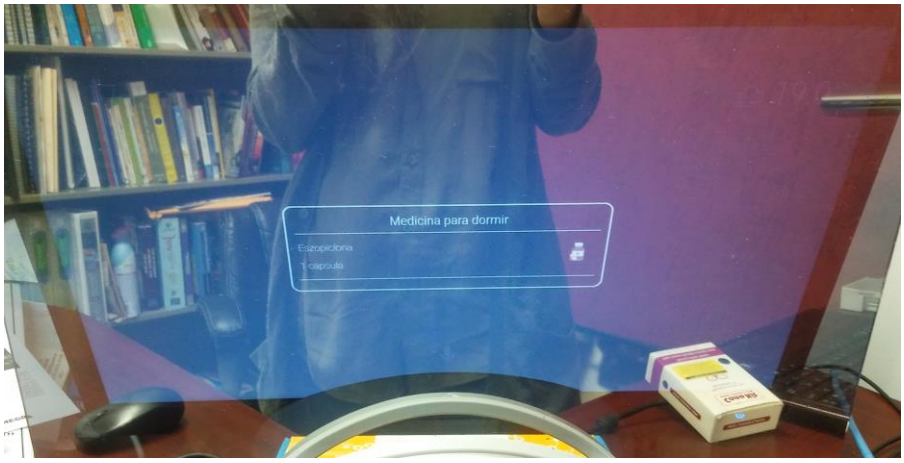


Figura 19. Prototipo de espejo inteligente - Display Alerta.

5. ANÁLISIS DE LA EXPERIENCIA ADQUIRIDA

En este apartado se presenta la valoración de los aprendizajes y lecciones aprendidas durante mi estancia profesional en la Universidad de Sonora, tanto en términos del proyecto en general, los objetivos que se plantearon para su realización, las actividades desarrolladas y la metodología que se utilizó.

5.1. ANÁLISIS GENERAL DEL PROYECTO

En general la organización y operación del desarrollo del proyecto de investigación trabajado en el departamento de ingeniería industrial y abordado en el presente documento fue bastante interesante y me permitió explotar mi creatividad e ingenio al desarrollar las propuestas del espejo inteligente. Se trabajó bajo una metodología de diseño centrado en el usuario, en el cual se trabajó con un grupo reducido conformado por dos practicantes, un tesista y dos profesores que guiaron y aprobaron el avance del proyecto en todo momento. Al tratarse de un grupo pequeño se trabajó de forma fluida y coordinada, además, resaltó la autoorganización y el autodesarrollo de sus miembros que permitieron impulsar nuevas ideas en los ámbitos que rodean la carrera de ingeniería de software.

5.2. ANÁLISIS DE LOS OBJETIVOS DE LAS PRACTICAS

Los objetivos del proyecto consistieron en diseñar, implementar y probar un visualizador ambiental para apoyar la medicación de los adultos mayores.

En un principio este proyecto fue todo un reto para mí, el análisis de las necesidades del usuario y su convergencia con las tecnologías existentes me permitieron conocer y comprender un poco más de la complejidad inherente en las necesidades de los usuarios, en este caso de los adultos mayores, y las plataformas inteligentes.

En cuanto al diseño del visualizador, considero que se alcanzó este objetivo ya que la arquitectura del sistema quedo bien asentada en la documentación. En cuanto a la implementación, se logró desarrollar e implementar la funcionalidad mínima del espejo inteligente. Se logro preparar los escenarios para probar el espejo y realizar la evaluación de usabilidad del dispositivo.

5.3. ANÁLISIS DE LAS ACTIVIDADES REALIZADAS

Una de las ventajas de realizar las prácticas profesionales en un proyecto de la universidad de sonora es que la comunicación y la libertad al desarrollar las actividades es muy amplia y permiten abordar los problemas desde varias perspectivas. La comunicación con el cliente, en este caso el responsable del proyecto fue en todo momento muy fluida y continua y se proporcionaron las herramientas requeridas a la brevedad.

Participar en el diseño y desarrollo de un proyecto de inteligencia ambiental desde sus cimientos me dio un panorama amplio sobre el ciclo de vida de un producto de software y me permitió probar mis conocimientos en todas las áreas del ciclo de vida, desde el análisis, diseño, desarrollo, implementación, testing y retroalimentación.

5.4. ANÁLISIS DE LA METODOLOGÍA UTILIZADA

En cuanto a la metodología seguida, se trabajo siempre siguiente un diseño centrado en el usuario, para ello fue muy importante entender y comprender las necesidades y el contexto del usuario, así como el producto a desarrollar y las plataformas o tecnologías que aportarían una solución, a la problemática planteada.

En cuanto a la forma de trabajo individual fue siempre muy abierta y con tendencias al autoaprendizaje y la participación continua, por lo cual me permitieron ampliar mis habilidades de liderazgo, superación y orientación a los resultados.

6. CONCLUSIONES Y RECOMENDACIONES

Las prácticas profesionales se desarrollaron en un equipo pequeño, pero en un proyecto muy interesante y con un gran potencial de aplicación y aprendizaje que creció alimentado por la retroalimentación y conocimientos de la comunidad académica.

El realizar las prácticas profesionales en este contexto me permitió darme cuenta de que no importa el tamaño o rubro de la empresa para adquirir una excelente experiencia, lo importante es ser participe de forma activa y positiva de las actividades y sacar el máximo provecho a los recursos y tiempo brindado por la institución.

Considero que algunas áreas de oportunidad que pudieran atenderse son:

- Mayor difusión dentro de la Universidad de Sonora de los proyectos realizados.
- Seguir fomentando la participación de los alumnos de ingeniería en sistemas en el desarrollo de proyectos, pero tratar de integrar a estudiantes de otras carreras afines a los proyectos que dieran otro panorama de la problemática.

A lo largo de los tres meses que estuve realizando mi estancia profesional aprendí bastante sobre el diseño centrado en el usuario, el desarrollo de aplicaciones en entornos web y la resolución de problemas en entornos académicos.

7. REFERENCIAS BIBLIOGRÁFICAS Y VIRTUALES

[1] A. Lehmann *et al.*, "Assessing medication adherence: Options to consider," *Int. J. Clin. Pharm.*, vol. 36, no. 1, pp. 55–69, 2014.

[2] G. Acampora, D. J. Cook, P. Rashidi, and A. V Vasilakos, "A Survey on Ambient Intelligence in Healthcare," *Proc. IEEE*, vol. 101, no. 12, pp. 2470–2494, 2013.

[3] Miguel Gea; and Francisco Luis Gutiérrez, "El diseño," *Libr. AIPO*, 2002.

[4] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

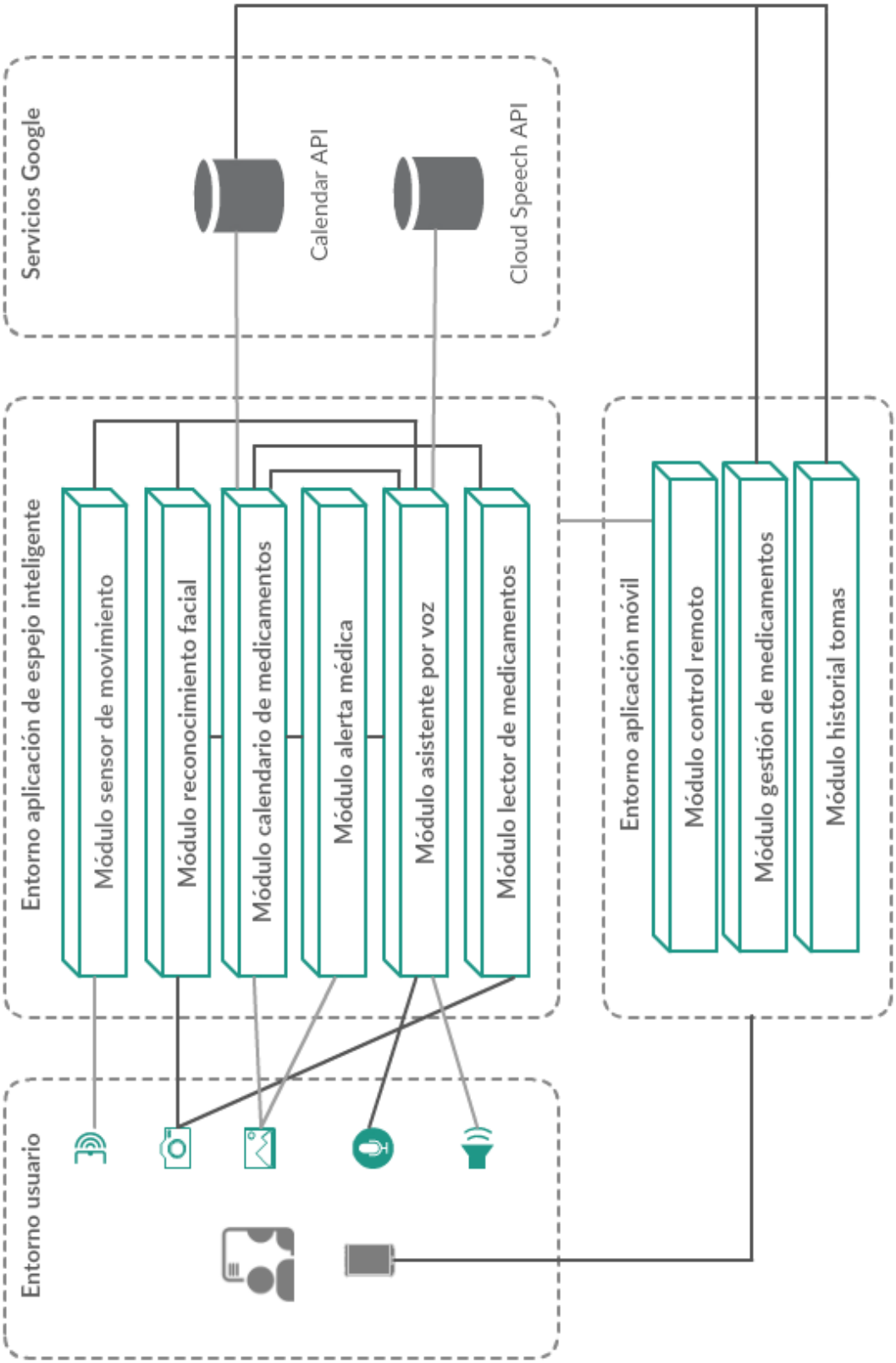
[5] https://www.asus.com/uk/Single-board-Computer/TINKER-BOARD/

[6] https://git-scm.com/

[7] https://developer.mozilla.org/es/docs/Web/JavaScript

ANEXOS

Anexo 1: Diagrama de arquitectura del proyecto.



Anexo 2: Póster del proyecto.