UNIVERSIDAD POLITÉCNICA DE PUEBLA

Ingeniería en Informática

Proyecto de Estancia Práctica en Desarrollador en Sistemas de Software y Administrador de Redes

Sistema de reconocimiento e identificación personas que no hagan uso de cubrebocas

Área temática del CONACYT: VII

Ingenierías y tecnologías

Presenta:

Jesús David Rodríguez Escobedo

Asesor técnico

Dr. Caldera Miguel Javier

Asesor académico

Ma. Rebeca Rodriguez Huesca

Juan C. Bonilla, Puebla, México. 18 de diciembre de 2020

Resumen

En el presente documento se presentará el desarrollo de un sistema de detección de cubrebocas e identificación de las personas en tiempo real escrito en el lenguaje de programación de Python, utilizando las librerías de OpenCV, TensorFlow y Keras mediante la metodología ágil de Programación Extrema (XP).

Para el procesamiento de las imágenes, primero se utiliza un algoritmo “entrenado” para la detección de rostros evitando que se haga el análisis de cubrebocas en otro tipo de objetos, una vez detectado un rostro humano el sistema empezará a analizar si la persona porta o no un cubrebocas.

Si la persona hace uso del cubrebocas detectará al usuario y seguirá su proceso de detección sin mandar alerta alguna ya que el usuario no ha cometido ninguna falta, en caso de no tenerlo procederá a la identificación de la persona, seguido de la captura de su rostro asignando un nombre clave, esta imagen se almacenará en una carpeta local y se subirá a la nube la correspondiente información del evento.

Índice

1. Introducción

En este capítulo se introducirá al lector sobre la descripción del problema, su justificación y los objetivos del proyecto, a continuación, hablaremos daremos a conocer los motivos por los que se decidió realizar el proyecto, de dónde surge la necesidad y qué beneficio puede brindar nuestro proyecto, además de detallar los objetivos que deben cumplirse para la realización del proyecto.

1.1. Descripción del problema o necesidad

En diciembre de 2019 estalló el brote de un nuevo virus conocido como COVID-19 perteneciente a la familia de los coronavirus que pueden causar infecciones respiratorias que pueden ir desde el resfriado común hasta enfermedades más graves como el síndrome respiratorio de Oriente Medio (MERS) y el síndrome respiratorio agudo severo (SRAS). La enfermedad se propaga principalmente de persona a persona a través de las gotículas que salen despedidas de la nariz o la boca de una persona infectada al toser, estornudar o hablar. Una persona puede contraer la COVID-19 si inhala las gotículas procedentes de una persona infectada por el virus, estas gotículas pueden caer sobre los objetos y superficies que rodean a la persona, de modo que otras personas pueden infectarse si tocan esos objetos o superficies y luego se tocan los ojos, la nariz o la boca. Por ello es importante lavarse las manos frecuentemente con agua y jabón o con un desinfectante a base de alcohol[1].

La Organización Mundial de la Salud menciona que no hay suficientes pruebas a favor o en contra del uso de mascarillas[1] (médicas o de otro tipo) por personas sanas de la comunidad en general debido a muchos factores entre ellos el hecho que los poros de los cubrebocas desechables en promedio miden 20 micras y el virus mide entre 70 y 90 nanómetros, sin embargo, una persona infectada al hablar, estornudar o toser libera gotículas que pueden caer en las superficies o personas además de ser inhaladas, éstas gotículas incrementan el tamaño del virus por lo que pueden ser retenidas por el cubrebocas, pero cabe aclarar que no es una barrera 100% efectiva[2].

Sin embargo, se encontró que al utilizar un EPP(Equipo de Protección Personal) resultó en una reducción significativa (al menos 50% de prevalencia y 20% de incidencia acumulada) en el riesgo de respiradores N95 ajustados y no ajustados, mascarillas quirúrgicas de alta filtración y tanto de baja filtración como de mascarillas pediátricas de alta filtración. Una tasa de cumplimiento del 80% eliminó esencialmente el brote de influenza. *"Si 10 a 15% de la población usa respirador N95, se reduciría 20% la incidencia de infecciones respiratorias"*[3].

[1] Se usará de forma indistinta mascarilla o cubrebocas.

Como es mencionado en la revista migración y desarrollo, *“la pandemia de la covid-19 se da en condiciones de un contexto capitalista altamente desarrollado.”* Donde el sistema capitalista presenta al menos tres tendencias generales que le son intrínsecas y que se retroalimentan dialécticamente: la ampliación de la división social del trabajo, la profundidad de la diferenciación e inequidad social, y el desarrollo de las fuerzas productivas[4].Generando en diferentes escalas la posibilidad de propagación del virus; si las medidas sanitarias no son aplicadas de forma eficiente como el uso apropiado de los EPP o sanitizantes; determinadas por el sector: industrial, comercial, educativo; la fuerza productiva si es automática o mano de obra.

A dos siglos y medio de la Revolución industrial, la profundización y expansión de aquellas tres tendencias puede visualizarse fácilmente en las cadenas productivas globales, donde para la fabricación de un producto participan decenas de empresas en muchos países simultáneamente. Si prestamos atención a la pandemia, aquellas tendencias se expresan de múltiples formas directamente conectadas con la enfermedad. Se manifiestan en la expansión ilimitada y globalizada de la producción y el mercado, que llega a borrar las barreras naturales que antaño contenían las epidemias. A ello se agrega un fuerte movimiento no sólo de mercancías, sino de personas que atraviesan fronteras nacionales y regionales como el turismo[4].

Con las enormes cantidades de personas que cruzan las diversas instalaciones, puntos de control o diversas infraestructuras existe un alto nivel de contagio y/o propagación de la enfermedad, no todos los establecimientos pueden tener personal a cargo del monitoreo del personal o clientela en todo momento dando la posibilidad a la gente que no haga uso del cubrebocas por periodos cortos o largos de tiempo. La reactivación de las actividades en todos los sectores, económico, social, educativo, entre otros son fundamentales para poder continuar con nuestra vida cotidiana adaptándonos a “la nueva normalidad”. Pero ello no puede ser posible si los establecimientos no cumplen con las medidas de salud correspondientes, como es indicado en el los “lineamientos de protección a la salud para reanudar actividades hacia un regreso seguro a la nueva normalidad en la ciudad de México”. en ellos se comenta que: *“las tiendas departamentales son espacios de trabajo donde se tiene contacto constante con decenas de personas diariamente. Es por ello que las empresas empleadoras deben habilitar y asegurar la operación diaria del filtro de supervisión para el ingreso de las personas trabajadoras.”*[5]

1.2. Justificación

A raíz de la pandemia se retoma el término de “Nueva Normalidad”, que limitadamente se enfoca en un regreso a las actividades cotidianas bajo un esquema que refuerza el lavado frecuente de manos, el uso de un gel antimicrobiano, el distanciamiento físico (no distanciamiento social) caracterizado por un fenotipo humano cubierto con cubrebocas, mascarillas o caretas[6].

El uso de mascarillas y/o cubrebocas se ha vuelto parte ya de nuestra vestimenta y estándar de vida por el bien de nuestra salud hasta que se encuentre una cura, salir del confinamiento para realizar nuestras actividades diarias deber realizarse con respeto hacia los lineamientos sanitarios y, sobre todo, a la responsabilidad para el autocuidado de la salud, simplemente hay que cuidarse para no infectarse, desgraciadamente no toda la población hace el uso del mismo o simplemente no lo utiliza de manera apropiada, aún en lugares donde se “exige” el uso del cubrebocas, por ello algunas empresas y/o establecimientos requieren la detección en tiempo real del uso de las mascarillas para mantener las normas de salubridad dentro de ellos cuidando la salud de sus empleados y clientes.

El Desarrollo de las fuerzas productivas que caracteriza al capitalismo contemporáneo, sustentado en las TICs y la llamada revolución de las tecnociencias. El nivel de desarrollo científico y tecnológico es tan espectacular que en algunos sectores los sistemas productivos corrigen el curso de la producción automáticamente, con algoritmos y bases de datos (Big Data) que trascienden las capacidades de un mortal para darle seguimiento[5] permitiéndonos abarcar más con menos, para ello se propone un sistema autónomo capaz de identificar el uso mismo del cubrebocas de las personas en tiempo real para mantener la mínima probabilidad de contagio , y determinar el personal que no esté usándolo para emitir una sanción o aviso.

Esto permitirá a los establecimientos supervisar con mayor eficiencia a la gente que haga el uso del cubrebocas en todo momento, reduciendo la posibilidad de contagio en gran medida a su clientela y personal de trabajo. Al saber la dificultad económica por la que pasan diversos establecimientos o instituciones el modelo propuesto será desarrollado en el lenguaje de programación de Python al ser de código abierto usando las librerías de OpenCV, TensorFlow y Keras para el procesamiento de imágenes con el fin de determinar si en la imagen se está haciendo uso del cubrebocas, en caso de no ser así el sistema obtendrá una imagen de la persona que no lo porte y se hará la identificación del empleado que no lo porte para alguna amonestación o aviso.

1.3. Objetivo General y Específicos

El objetivo general es desarrollar un sistema informático para la identificación del uso de cubrebocas en personas en tiempo real.

Los objetivos específicos son:

1. Se analizarán los componentes necesarios para su implementación como son: cámara para la captura de video y un equipo de cómputo para el procesado de la información.
2. Se realizará el diseño arquitectónico de los componentes
3. Se desarrollará un módulo de aprendizaje y entrenamiento de reconocimiento del uso de cubrebocas en personas
4. Se desarrollará un módulo de reconocimiento de uso de cubrebocas en personas.
5. Se desarrollará un módulo de identificación de usuarios.
6. Se aplicarán pruebas de funcionalidad a los componentes para determinar las personas que porten o no el cubrebocas

2. Metodología y herramientas

En este capítulo describiremos la metodología empleada, sus etapas y cómo las implementaremos, además de mencionar y describir las herramientas utilizadas en el desarrollo del proyecto.

2.1 Metodología Ágil Programación Extrema (XP)

El software se ha incrustado profundamente en casi todos los aspectos de nuestras vidas y, como consecuencia, el número de personas que tienen interés en las características y funciones que brinda una aplicación específica ha crecido en forma notable[7].

Los individuos, negocios y gobiernos dependen cada vez más del software para tomar decisiones estratégicas y tácticas, así como para sus operaciones y control cotidianos. Si el software falla, las personas y empresas grandes pueden experimentar desde un inconveniente menor hasta fallas catastróficas, por ello se concluye que el software debe tener alta calidad y a medida que aumenta el valor percibido de una aplicación específica se incrementa la probabilidad de que su base de usuarios y longevidad también crezcan. Conforme se extienda su base de usuarios y el tiempo de uso, las demandas para adaptarla y mejorarla también crecerán. Se concluye que el software debe tener facilidad para recibir mantenimiento[7].

“La Ingeniería de software es la aplicación práctica del conocimiento científico al diseño y construcción de programas de computadora y a la documentación asociada requerida para desarrollar, operar y mantenerlos. Se conoce también como Desarrollo de Software o Producción de Software” (Bohem, 1976) [8].

Por estos motivos necesitamos una metodología apropiada para el correcto desarrollo de nuestro software, una metodología es una de las etapas específicas de un trabajo o proyecto que parte de una posición teórica y conlleva a una selección de técnicas concretas o métodos acerca del procedimiento para el cumplimiento de los objetivos. Es el conjunto de métodos que se utilizan en una determinada actividad con el fin de formalizarla y optimizarla. Determina los pasos a seguir y cómo realizarlos para finalizar una tarea[8].

Las primeras actividades con las ideas y los métodos asociados a Programación Extrema (XP) ocurrieron al final de la década de 1980, el trabajo fundamental sobre la materia había sido escrito por Kent Beck[7], autor del primer libro sobre la materia, Extreme Programming Explained: Embrace Change (1999). La metodología XP es la más destacada de los procesos ágiles de desarrollo de software. Al igual que los demás, la programación extrema se diferencia de las

metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad[9].

«Todo en el software cambia. Los requisitos cambian. El diseño cambia. El negocio cambia. La tecnología cambia. El equipo cambia. Los miembros del equipo cambian. El problema no es el cambio en sí mismo, puesto que sabemos que el cambio va a suceder; el problema es la incapacidad de adaptarnos a dicho cambio cuando éste tiene lugar.» Kent Beck.

Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos[10].

Extreme Programming (XP) surge como una nueva manera de encarar proyectos de software, proponiendo una metodología basada esencialmente en la simplicidad y agilidad. Las metodologías de desarrollo de software tradicionales (ciclo de vida en cascada, evolutivo, en espiral, iterativo, etc.) aparecen, comparados con los nuevos métodos propuestos en XP, como pesados y poco eficientes. La crítica más frecuente a estas metodologías “clásicas” es que son demasiado burocráticas. Hay tanto que hacer para seguir la metodología que, a veces, el ritmo entero del desarrollo se retarda. Como respuesta a esto, se ha visto en los últimos tiempos el surgimiento de “Metodologías Ágiles”. Estos nuevos métodos buscan un punto medio entre la ausencia de procesos

Beck define cuatro valores que nos establecen algunos criterios para todo trabajo realizado como parte de XP: comunicación, simplicidad, retroalimentación, valentía. Sin embargo, los valores son demasiado vagos para ayudarnos a decidir qué prácticas utilizar por ello necesitamos sintetizar estos valores en principios, estos principios nos ayudarán a elegir entre alternativas. Un valor puede ser vago, pero un principio es más concreto. Los principios fundamentales que Beck menciona son[11]: Retroalimentación rápida, asumir la simplicidad, cambio incremental, emprendiendo el cambio, trabajo de calidad.

La programación extrema usa un enfoque orientado a objetos como paradigma preferido de desarrollo, y engloba un conjunto de reglas y prácticas que ocurren en el contexto de cuatro actividades estructurales (Figura1):

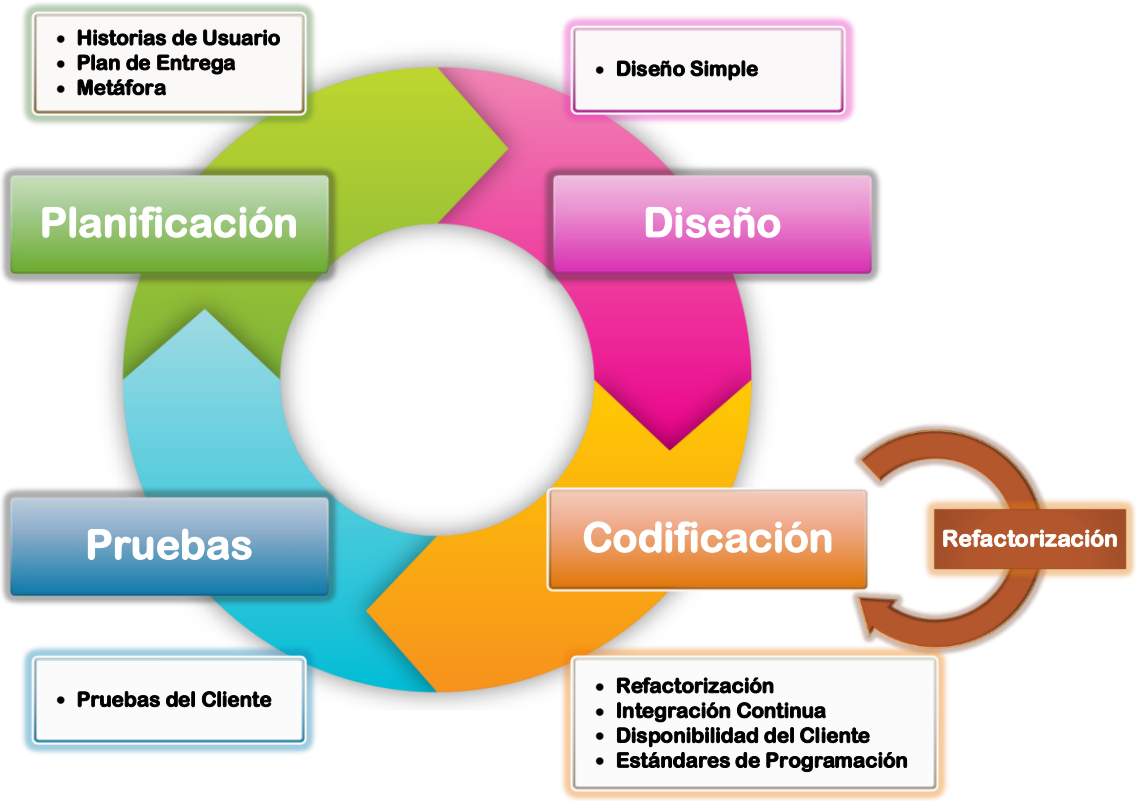

Figura 1 Fases de la metodología XP

A continuación, explicaremos cada fase y las respectivas etapas que aplicaremos en nuestro proyecto:

2.1.1 Planeación

La actividad de planeación comienza escuchando —actividad para recabar requerimientos que permite que los miembros técnicos del equipo XP entiendan el contexto del negocio para el software y adquieran la sensibilidad de la salida y características principales y funcionalidad que se requieren—.

2.1.1.1 Historias de Usuario

Escuchar lleva a la creación de algunas “historias de usuario” (Las cuales se componen de 3 partes “que, cómo, para qué”) describen la salida necesaria, características y funcionalidad del software que se va a elaborar.

Cada historia es escrita por el cliente y colocada en una tarjeta indizada[10].Constan de 3 o 4 líneas escritas por el cliente en un lenguaje no técnico. El cliente asigna un valor (es decir, una prioridad) a la historia con base en el valor general de la característica o función para el negocio.

Después, los miembros del equipo XP evalúan cada historia y le asignan un costo, medido en semanas de desarrollo. Si se estima que la historia requiere más de tres semanas de desarrollo, se pide al cliente que la descomponga en historias más chicas y de nuevo se asigna un valor y costo[10].

A medida que avanza el trabajo, el cliente puede agregar historias, cambiar el valor de una ya existente, descomponerlas o eliminarlas. Entonces, el equipo XP reconsidera todas las entregas faltantes y modifica sus planes en

consecuencia[7]. Si alguna de ellas tiene “riesgos” que no permiten establecer con certeza la complejidad del desarrollo, se realizan pequeños programas de prueba (*“spikes”*), para reducir estos riesgos.

2.1.1.2 Plan de Entrega

Una vez realizadas estas estimaciones, se organiza una reunión de planificación, con los diversos actores del proyecto (cliente, desarrolladores, gerentes), a los efectos de establecer un plan de entregas (“Release Plan”) en los que todos estén de acuerdo. Una vez acordado este cronograma, comienza una fase de iteraciones, en dónde en cada una de ellas se desarrolla, prueba e instala unas pocas “historias de usuarios”.

2.1.1.3 Metáfora del negocio

- Es una historia común compartida por el usuario y el equipo de desarrollo.
- Debe servir para que el usuario se sienta a gusto refiriéndose al sistema en los términos de ella.
- Debe servir a los desarrolladores para implementar las clases y objetos del sistema.

2.1.2 Diseño

El diseño XP sigue rigurosamente el principio MS (mantenlo sencillo). Un diseño sencillo siempre se prefiere sobre una representación más compleja. Además, el diseño guía la implementación de una historia conforme se escribe: nada más y nada menos. Se desalienta el diseño de funcionalidad adicional porque el desarrollador supone que se requerirá después.

Si en el diseño de una historia se encuentra un problema de diseño difícil, problemas técnicos, o cuando es difícil de estimar el tiempo para implementar una historia de usuario, XP recomienda la creación inmediata de un prototipo operativo de esa porción del diseño[7]. Entonces, se implementa y evalúa el prototipo del diseño, llamado “solución en punta” o “spike” para explorar diferentes soluciones. Estos programas son únicamente para probar o evaluar una solución, y suelen ser desechados luego de su evaluación[12]. El objetivo es disminuir el riesgo cuando comience la implementación verdadera y validar las estimaciones originales para la historia que contiene el problema de diseño.

2.1.3 Desarrollo del código o codificación

La codificación debe hacerse ateniendo a estándares de codificación ya creados. Programar bajo estándares mantiene el código consistente y facilita su comprensión y escalabilidad. Crear test que prueben el funcionamiento de los distintos códigos implementados nos ayudará a desarrollar dicho código. Crear estos test antes nos ayuda a saber qué es exactamente lo que tiene que hacer el código a implementar y sabremos que una vez implementado pasará dichos test sin problemas ya que dicho código ha sido diseñado para ese fin. Se puede dividir la funcionalidad que debe cumplir una tarea a programar en pequeñas

unidades, de esta forma se crearán primero los test para cada unidad y a continuación se desarrollará dicha unidad, así poco a poco conseguiremos un desarrollo que cumpla todos los requisitos especificados.

Un concepto clave durante la actividad de codificación (y uno de los aspectos del que más se habla en la XP) es la programación por parejas. XP recomienda que dos personas trabajen juntas en una estación de trabajo con el objeto de crear código para una historia. Esto da un mecanismo para la solución de problemas en tiempo real (es frecuente que dos cabezas piensen más que una) y para el aseguramiento de la calidad también en tiempo real (el código se revisa conforme se crea). También mantiene a los desarrolladores centrados en el problema de que se trate además que en la práctica, cada persona adopta un papel un poco diferente[7].

2.1.3.1 Refactorización.

Es muy común rehusar códigos ya creados que contienen funcionalidades que no serán usadas y diseños obsoletos. Esto es un error porque puede generar código completamente inestable y muy mal diseñado; por este motivo, es necesario refactorizar cuando se va a utilizar código ya creado.

La refactorización mejora la estructura interna del código sin alterar su comportamiento externo el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios. El diseño del sistema de software es una cosa viviente. No se puede imponer todo en un inicio, pero en el transcurso del tiempo este diseño evoluciona conforme cambia la funcionalidad del sistema.[13]

Cuando implementamos nuevas características en nuestros programas nos planteamos la manera de hacerlo lo más simple posible, después de implementar esta característica, nos preguntamos cómo hacer el programa más simple sin perder funcionalidad, este proceso se le denomina recodificar o refactorizar (refactoring). Esto a veces nos puede llevar a hacer más trabajo del necesario, pero a la vez estaremos preparando nuestro sistema para que en un futuro acepte nuevos cambios y pueda albergar nuevas características. No debemos de recodificar ante especulaciones si no solo cuando el sistema te lo pida.

Para mantener un diseño apropiado, es necesario realizar actividades de cuidado continuo durante el ciclo de vida del proyecto. De hecho, este cuidado continuo sobre el diseño es incluso más importante que el diseño inicial. Un concepto pobre al inicio puede ser corregido con esta actividad continua, pero sin ella, un buen diseño inicial se degradará[13].

2.1.3.2 Integración continua

La integración continua a menudo reduce la fragmentación de los esfuerzos de los desarrolladores por falta de comunicación sobre lo que puede ser reutilizado o compartido[13], cada pieza de código es integrada en el sistema una vez que

esté lista, así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día, esto ayuda a evitar los problemas de compatibilidad e interfaces y brinda un ambiente de "prueba de humo" ayuda a descubrir a tiempo los errores.

Todos los desarrolladores necesitan trabajar siempre con la "última versión", realizar cambios o mejoras sobre versiones antiguas causan graves problemas, y retrasan al proyecto. Es por eso que XP promueve publicar lo antes posible las nuevas versiones, aunque no sean las últimas, siempre que estén libres de errores. Idealmente, todos los días deben existir nuevas versiones publicadas. Para evitar errores, solo una pareja de desarrolladores puede integrar su código a la vez[12].

2.1.3.3 Disponibilidad del cliente

Uno de los requerimientos de XP es tener al cliente disponible durante todo el proyecto. No solamente como apoyo a los desarrolladores, sino formando parte del grupo. El involucramiento del cliente es fundamental para que pueda desarrollarse un proyecto con la metodología XP. contienen los detalles necesarios para realizar el desarrollo del código. Estos detalles deben ser proporcionados por el cliente, y discutidos con los desarrolladores, durante la etapa de desarrollo. No se requieren de largos documentos de especificaciones, sino que los detalles son proporcionados por el cliente, en el momento adecuado, "cara a cara" a los desarrolladores[12].

2.1.3.4 Estándares de programación

XP enfatiza la comunicación de los programadores a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación (del equipo, de la organización u otros estándares reconocidos para los lenguajes de programación utilizados). Los estándares de programación mantienen el código legible para los miembros del equipo, facilitando los cambios.

2.1.4 Pruebas

No debe existir ninguna característica en el programa que no haya sido probada, los programadores escriben pruebas para chequear el correcto funcionamiento del programa, los clientes realizan pruebas funcionales. El resultado un programa más seguro que conforme pasa el tiempo es capaz de aceptar nuevos cambios[13].

Wells [14] dice: *"Corregir pequeños problemas cada cierto número de horas toma menos tiempo que resolver problemas enormes justo antes del plazo final."* Las pruebas de aceptación XP, también llamadas pruebas del cliente, son especificadas por el cliente y se centran en las características y funcionalidad generales del sistema que son visibles y revisables por parte del cliente. Las pruebas de aceptación se derivan de las historias de los usuarios que se han implementado como parte de la liberación del software.

Sabemos que existen diversas metodologías y a su vez cada una tiene ventajas y desventajas entre ellas dependiendo del tipo de proyecto que se quiera realizar, dentro de la metodología ágil XP podemos encontrar[15][16]:

Ventajas

- Da lugar a una programación sumamente organizada.
- Ocasiona eficiencias en el proceso de planificación y pruebas.
- Cuenta con una tasa de errores muy pequeña.
- Propicia la satisfacción del programador.
- Fomenta la comunicación entre los clientes y los desarrolladores.
- Facilita los cambios.
- Permite ahorrar mucho tiempo y dinero.
- Puede ser aplicada a cualquier lenguaje de programación.
- El cliente tiene el control sobre las prioridades.
- Se hacen pruebas continuas durante el proyecto.
- La XP es mejor utilizada en la implementación de nuevas tecnologías

Desventajas

- Es recomendable emplearla solo en proyectos a corto plazo.
- En caso de fallar, las comisiones son muy altas.
- Requiere de un rígido ajuste a los principios de XP.
- Puede no siempre ser más fácil que el desarrollo tradicional.
- Dificultad para documentar, como esta metodología trabaja de forma rápida y con cambios constantes, es difícil llevar un registro e historial de lo que se ha hecho.
- Fuerte dependencia de las personas.

2.2 Herramientas

A continuación, se describen las herramientas utilizadas durante el desarrollo del proyecto

2.2.1 Python

Python[17] es un lenguaje de programación creado por Guido van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses “Monty Python”[18].

A continuación, se hablarán sobre las ventajas y desventajas de Python[19] [20]:

Ventajas

- **Estilo flexible:** Ello se debe a que te aporta muchas herramientas para crear código de manera flexible. Por ejemplo, si deseas tener listas para varios tipos de datos, no es necesario que declares cada uno de estos (esto se hace para determinar la clase del dato). Por otro lado, la sintaxis es

comprensible y en algunas funciones se asemejan a estructuras de otros lenguajes.

- **Ordenado y limpio:** Significa que este lenguaje es legible y entendible para cualquier programador que quiera trabajar sobre una estructura ya establecida y ello se debe a la organización de sus módulos.
- **Cuenta con una comunidad activa:** A diferencia de otros tipos de lenguajes menos populares, Python cuenta con una comunidad activa de usuarios comprometidos a ayudar en las actualizaciones.
- **Open Source:** Este lenguaje de programación es de software libre, significa que puedes emplearlo en cualquier momento para tus proyectos. Además, puedes utilizarlo como base para crear extensiones o desarrollar módulos.
- **Simplificado y rápido:** Es muy simplificado ya que cuenta con varios patrones orientados a acciones. Además, por ser un lenguaje interpretado su ejecución se hace de manera rápida debido a que no necesita ser compilado. Con esto te ahorras un montón de tiempo para programar y crear proyectos.
- **Estilo sano de programación:** Los programadores mantienen un estilo sano de programar, direccionado a las reglas perfectas. Además, ello sumado a sus otros beneficios hace que sea un lenguaje productivo.
- **Multiplataforma:** Este lenguaje lo puedes emplear en varios sistemas operativos como Linux, Windows o Mac OS. Por otra parte, incluye las librerías más populares dentro del intérprete, de manera que no debes perder tiempo en instalarlas como ocurre con otros lenguajes.

Desventajas

- **Problemas con hosting:** Existen muchos servidores que no soportan Python y en caso de hacerlo, su configuración suele ser compleja.
- **Librerías incorporadas :** A pesar de incluir un compendio de librerías populares (por ejemplo: kivy, requests, scapy, entre otras), algunas de esas no son necesarias o se usan muy poco. Respecto a estas últimas, te menciono como ejemplo aquellas que sirven para trabajar con HTTP. En este caso puedes optar por usar librerías de terceros.
- **Lentitud al ejecutar múltiples hilos:** A pesar de que no necesita compilarse, si quieres ejecutar múltiples hilos de programación puede que no aproveches toda la potencia de tu PC. Por ejemplo, pueden surgir errores por parte del intérprete o simplemente tener problemas para usar todos los núcleos del procesador.
- **No dispone de buena documentación:** Python no cuenta con buena documentación, por lo que puedes tener problemas para comprender algunas librerías y ciertas estructuras, sobre todo si no has programado antes. Todo ello comparado con otros lenguajes de programación como Java, PHP o C++.
- **Curva de aprendizaje:** Aunque el lenguaje es comprensible, para el desarrollo web puede tomar algo de tiempo aprenderlo, sobre todo si no se tiene ninguna idea sobre programación orientada a objetos.

- **No tiene identificadores protegidos:** A diferencia de otros tipos de lenguajes como PHP, Python no cuenta con identificadores protegidos, por lo que los métodos empleados son públicos.
- **Simulaciones:** Para simulaciones físicas el lenguaje Python puede resultar complejo, ya que no trabaja con matrices por defecto, tal como ocurre con otros lenguajes como Matlab. En definitiva, este lenguaje te es útil siempre que no dependas de una matriz o tengas que trabajar con un vector complejo, ya que de lo contrario debes importar bibliotecas.

2.2.2 OpenCV

OpenCV[21] es una herramienta Open Source publicada bajo licencia BSD. Es multiplataforma, pues soporta Windows, Linux, Mac OS, iOS y Android. Cuenta con interfaces Python, C++ y Java, por lo que es bastante completo. Fue diseñado para eficiencia computacional, enfocándose en aplicaciones en tiempo real. Está escrita en C/C++ y soporta procesamiento multinúcleo. Con OpenCL, puede tomar las ventajas de la aceleración de hardware[21].

Su uso está muy extendido, con una gran comunidad de usuarios. Los usos van desde el arte interactivo, a la inspección de minas o robótica avanzada. También se utiliza para detección facial y de diferentes objetos del entorno, como el proyecto YOLO, que hace uso de OpenCV.

A continuación, se hablarán sobre las ventajas y desventajas de OpenCV[20]:

Ventajas

- Open Source
- Es muy versátil.
- Multiplataforma.
- Tiene gran cantidad de documentación.

Desventajas

- La instalación es complicada
- Complicada ejecución

2.2.3 TensorFlow

TensorFlow[22] es una librería de código abierto para computación numérica de alto rendimiento. Cuenta con un buen soporte tanto para Machine Learning como para Deep Learning. Gracias a su flexibilidad, también se puede extrapolar a otros ámbitos científicos. Esta potente herramienta para Deep Learning ha sido desarrollada por Google y soporta diferentes lenguajes, entre los que están Python, Java y C++, además de soportar CUDA, dando soporte únicamente a GPUs de la firma NVIDIA y, además, se puede separar la forma de ejecución, sea en CPU o GPU, en dos paquetes de instalación diferentes. Las ventajas e

inconvenientes que tiene este framework de desarrollo para Deep Learning desarrollado por los trabajadores de Google, se enumeran a continuación[22].

A continuación, se hablarán sobre las ventajas y desventajas de TensorFlow[20] [23]:

Ventajas

- Escalabilidad.
- Se puede paralelizar muy bien.
- Soporta varios lenguajes.
- Puede ejecutarse a través de Docker.
- Está extendido, se pueden encontrar multitud de proyectos hechos con este framework.
- TensorFlow se puede aplicar a una variedad de casos de uso. Se puede usar para construir casi cualquier cosa que implique el procesamiento de datos ejecutando una serie de operaciones matemáticas en él. Sin embargo, TensorFlow se usa más comúnmente para construir redes neuronales, y toma la delantera en el aprendizaje profundo. Si está trabajando con pequeños conjuntos de datos, TensorFlow podría ser excesivo.
- TensorFlow no solo le brinda control total sobre su modelo, sino también sobre su lógica de preprocesamiento. El año pasado, Google anunció TensorFlow Transform, que permite al usuario definir tuberías de preprocesamiento que admiten pases completos sobre los datos para un procesamiento de datos distribuido, eficiente y a gran escala.

Desventajas

- Tiene una curva de aprendizaje grande. El diseño de las redes es algo tedioso.
- Alto tiempo de desbordamiento de pila
- Difícil de practicar, el lenguaje es muy estricto por lo que genera alta cantidad de errores sintácticos
- Solamente ofrece soporte a GPUs NVIDIA.
- A veces, las actualizaciones rompen la retrocompatibilidad.

Keras

Keras[23re] es una API de alto nivel para Deep Learning. Está escrita en Python y es capaz de ejecutarse sobre TensorFlow, Microsoft Cognitive Toolkit o Theano para simplificar el desarrollo de redes neuronales en estos Frameworks. Fue desarrollada enfocándose en la posibilidad de una experimentación rápida con redes neuronales. Permite un rápido y fácil prototipado de redes, soportando tanto redes convolucionales y recurrentes, así como una mezcla de ambas[24].

Una de las características más interesantes de Keras es la posibilidad de ejecutarse sobre CPU o GPU sin tener que especificarlo, ya que viene indicado por la herramienta que va por debajo. A pesar de ser un framework para

simplificar el desarrollo de redes neuronales en TensorFlow, por ejemplo, también cuenta con una serie tanto de ventajas como de inconvenientes. Estos se enumeran a continuación.

A continuación, se hablarán sobre las ventajas y desventajas de Keras[20][23]:

Ventajas

- Desarrollo rápido de modelos.
- Curva de aprendizaje pequeña.
- Puede ejecutarse utilizando diferentes frameworks que hacen de motor.
- Una comunidad grande de desarrolladores.
- Prioriza la experiencia del usuario al minimizar la carga cognitiva, constante, sencillo, reduce el número de acciones del usuario
- Respaldado por una comunidad grande y activa.
- Las capas neuronales, las funciones, los esquemas y los optimizadores son módulos independientes que puede combinar para crear nuevos modelos si lo desea.

Desventajas

- Si se desea desarrollar algo más específico, es muy difícil con Keras.
- Solamente tiene un formato de salida para los pesos: HDF5.
- Keras es una API bellamente escrita que no bloquea el acceso a marcos de nivel inferior. Dicho esto, Keras no está realmente diseñado para que cambies la arquitectura subyacente de tu modelo. Puede personalizar sus capas en Keras.

3. Resultados

Durante este capítulo se mostrará la realización de la metodología XP explicada en el capítulo 2 así como los resultados provenientes de cada fase y sus correspondientes etapas.

3.1 Planeación

Durante la planeación se realizaron diversas reuniones y entrevistas para determinar el objetivo del proyecto, esas ideas las representamos como "historias de usuario" y estas fueron agregándose durante el desarrollo del proyecto.

Las historias de usuario fueron siendo incluidas conforme le cliente solicitaba un cambio o pedía agregar algo más, normalmente era cada vez que se concluía una historia de usuario o estaba pronta su culminación(Figuras 2-6).

No. 1	Detección de Cubrebocas en Rostros
Como usuario me gustaría un programa que verifique que las personas porten cubrebocas, para tener la minima probabilidad de riesgo de contagio en el establecimiento	
Prioridad: Alta	Riesgo: Medio
Condiciones de Satisfacción ◦ El programa detecte personas con y sin cubrebocas. ◦ Funcione sin detenerse y en todo momento.	

Figura 2 Historia de Usuario #1

No. 2	Almacenamiento de Datos
Como usuario quiero que el programa tome una fotografía de las personas que no porten cubrebocas y guarde información en una carpeta dentro del sistema para tener una evidencia y poder actuar apropiadamente	
Prioridad: Alta	Riesgo: Alto
Condiciones de Satisfacción ◦ El programa detecte a la persona sin cubrebocas y tome una fotografía de ella. ◦ El nombre del archivo debe ser el correspondiente ◦ Funcione sin detenerse y en todo momento.	

Figura 3 Historia de Usuario #2

No. 3	Reconocimiento de Personas

Como usuario me gustaría que el programa reconozca a las personas que no porten cubrebocas, para automatizar el proceso de identificación y poder actuar más rápido

Prioridad: Alta Riesgo: Alto

Condiciones de Satisfacción
- El programa identifique a la persona sin cubrebocas y tome una fotografía de ella.
- El nombre de la imagen debe ser de entendible
- Funcione sin detenerse y en todo momento.

Figura 4 Historia de Usuario #3

No. 4	Mensaje de Voz al Usuario

Como usuario quiero que el programa mencione el nombre de las personas que no porten cubrebocas y guarde información en una base de datos, paa que reciban un aviso al instante de su falta, además de contar con un registro de los incidentes

Prioridad: Alta Riesgo: Alto

Condiciones de Satisfacción
- El programa identifique a la persona y mencione su nombre de forma clara
- La información pertinente debe guardarse sin problemas
- Funcione sin detenerse y en todo momento.

Figura 5 Historia de Usuario #4

Por medio de las historias de Usuario definidas para el desarrollo de nuestro proyecto, se elaboró un plan de entrega(Figura 6), donde se estipulan fechas de inicio y término de las iteraciones, además de ser una referencia en el progreso del proyecto.

N° de Historia	Iteración	Prioridad	Fecha inicio	Fecha término
1	1	Medio	07/09/2020	28/09/2020
2	2	Medio	05/10/2020	14/10/2020
3	3	Alto	19/10/2020	30/10/2020
4	4	Alto	02/11/2020	16/11/2020

Figura 6 Plan de Entrega

3.2 Diseño

Para seguir el principio MS (Mantenlo Sencillo) la metodología XP hace uso de la refactorización cuyo objetivo es “simplificar” el código; el proyecto fue realizado

en el lenguaje de programación de Python[17] por su “simplicidad” en código y facilidad de uso con las librerías de TensorFlow[22] y Keras[24] además.

A continuación (Figura 7) se muestra un ejemplo de simplificación (Refactorización) de código, donde se puede observar como el “spike_time2.py” posee un mayor código, tanto en líneas de comandos como en funciones y variables, algo que se simplifica drásticamente en el “spike_time3.py”. Al refactorizar el script, se buscó la manera de reducir la utilización de variables, así como la optimización de la estructura en funcionalidad y estética.

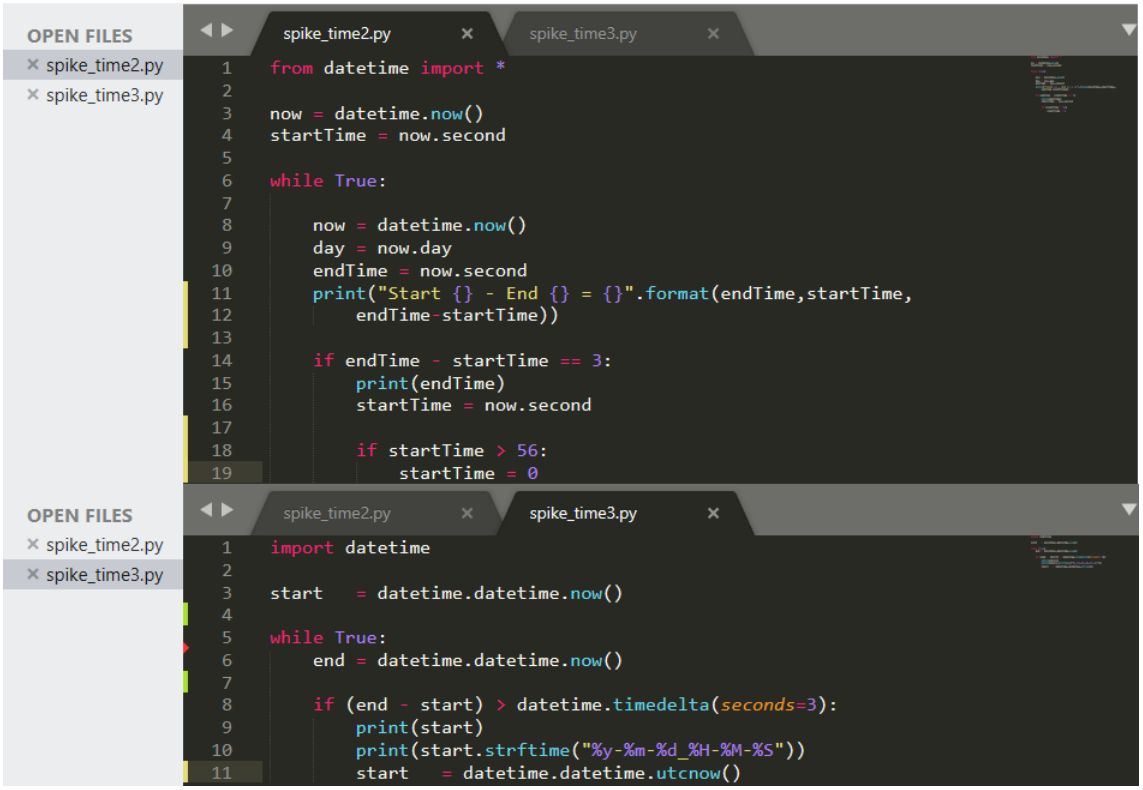

Figura 7 Ejemplo de simplificación (Refactorización) de spike

Durante la realización del proyecto, cada “spike” realizado(Figura 8) procuraba abarcar las características de la nueva historia de usuario evitando agregar cosas innecesarias trabajándola de manera “independiente” hasta ser refinada.

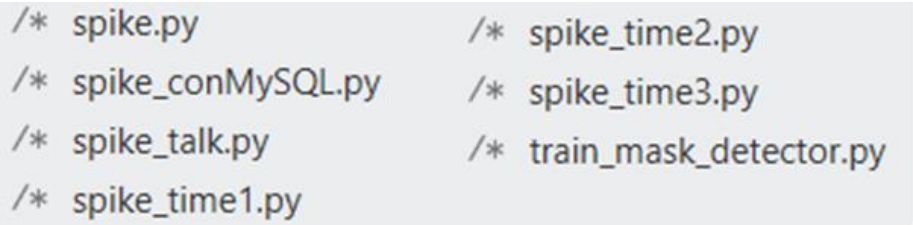

Figura 8 Listado de Spikes

Durante el diseño el cliente nos apoyó en la determinación de la nomenclatura que tendrían las imágenes capturadas(Figura 9) y la ruta de almacenamiento de las imágenes(Figura 10), como se muestra a continuación:

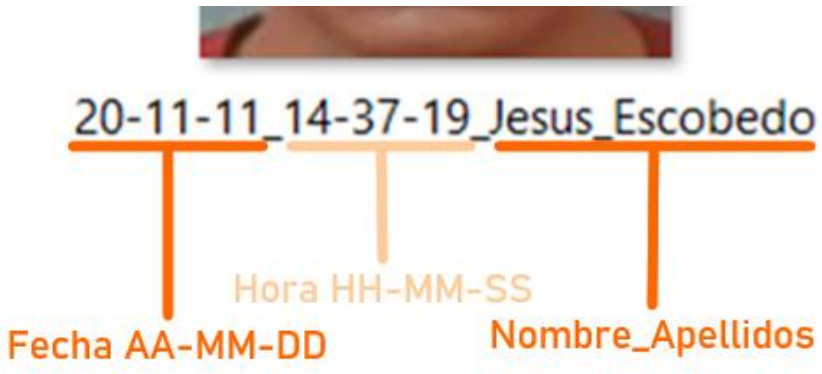

Figura 9 Nomenclatura de las imágenes capturadas

A continuación, mostramos el Diagrama de directorio de nuestros archivos:

- En el primer nivel podemos observar nuestra carpeta contenedora o raíz.
- En el segundo nivel se encuentran los scripts de programación en Python y los correspondientes archivos XML que fueron utilizados y la carpeta contenedora de los rostros capturados.
- En el tercer nivel se encuentran los rostros capturados en formato JPG con su nomenclatura correspondiente.

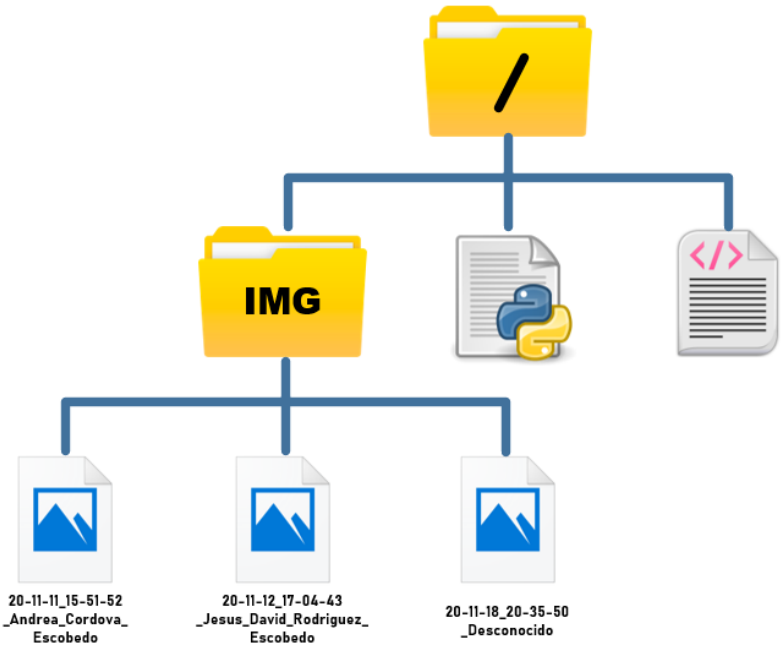

Figura 10 Diagrama del Directorio del Proyecto

3.3 Codificación

Con la adición de historias de usuario el código fue incrementando proporcionalmente, una vez el “spike” cumplía su función sin problemas se anexaba al código principal, mientras se refactorizaba, quitando cosas innecesarias y adaptando el código de prueba a nuestro código principal siguiendo los estándares de programación establecidos y el principio MS(Integración Continua) una vez integrado el “spike” se realizaba una prueba de funcionalidad para concluir la iteración pertinente. Cada historia de usuario

posee sus respectivas iteraciones, a su vez cada iteración posee sus respectivas tareas de ingeniería.

Primera Iteración

Para determinar el(los) objetivo(s) de la iteración actual nos apoyamos de la historia de usuario correspondiente, con la cuál podemos generar las tareas de ingeniería pertinentes para su realización(Figura 11).

HU-1 Detección de Cubrebocas en Rostros	
N° de Tarea	Nombre de la Tarea
1	Detección de rostros
2	Aprendizaje y entrenamiento de detección de cubrebocas
3	Detección de cubrebocas
4	Pruebas de Funcionalidad

Figura 11 Tareas de Ingeniería (Primera Iteración)

Durante la primera iteración se desarrolló el **módulo de aprendizaje y entrenamiento del uso de cubrebocas en personas** además del **módulo de reconocimiento de uso de cubrebocas en personas** mediante el lenguaje de programación Python, las librerías de TensorFlow y Keras para la detección de cubrebocas únicamente en personas. A continuación, detallamos las tareas de ingeniería de la iteración actual(Figuras 12-15).

Tarea de Ingeniería			
N° de Tarea:	1	N° de Historia:	1
Nombre de la Tarea:	Detección de Rostros		
Tipo de Tarea:	Desarrollo		
Fecha de Inicio: 07/09/20		Fecha de Término: 28/09/20	
Descripción de la Tarea: Se desarrollará un módulo de detección de rostros mediante TensorFlow y Keras para enfocar el procesamiento de información			

Figura 12 Tarea de Ingeniería 1 (UH-1)

Tarea de Ingeniería			
N° de Tarea:	2	N° de Historia:	1
Nombre de la Tarea:	Aprendizaje y entrenamiento de detección de cubrebocas		
Tipo de Tarea:	Desarrollo		
Fecha de Inicio: 07/09/20		Fecha de Término: 28/09/20	
Descripción de la Tarea: Se desarrollará un módulo de aprendizaje y entrenamiento de detección de cubrebocas mediante TensorFlow y Keras para que el sistema sea capaz de detectar los cubrebocas en los rostros de las personas			

Figura 13 Tarea de Ingeniería #2 (UH-1)

Tarea de Ingeniería			
N° de Tarea:	3	N° de Historia:	1
Nombre de la Tarea:	Detección de Cubrebocas		
Tipo de Tarea:	Desarrollo		
Fecha de Inicio: 07/09/20		Fecha de Término: 28/09/20	
Descripción de la Tarea: Se desarrollará un módulo de detección de cubrebocas empleando el modelo obtenido en la tarea previa para la detección del cubrebocas en tiempo real			

Figura 14 Tarea de Ingeniería 3 (UH-1)

Tarea de Ingeniería			
N° de Tarea:	4	N° de Historia:	1
Nombre de la Tarea:	Pruebas de Funcionalidad		
Tipo de Tarea:	Desarrollo		
Fecha de Inicio: 07/09/20		Fecha de Término: 28/09/20	
Descripción de la Tarea: Se realizarán pruebas de funcionalidad verificando que el sistema detecte los rostros de las personas y determine si hace uso o no de un cubrebocas, el sistema debe funcionar sin interrupciones de lo contrario deben aplicarse sus correcciones pertinentes.			

Figura 15 Tarea de Ingeniería 4 (UH-1)

Segunda Iteración

Para determinar el(los) objetivo(s) de la iteración actual nos apoyamos de la historia de usuario correspondiente, con la cuál podemos generar las tareas de ingeniería pertinentes para su realización(Figura 16).

HU-2	Almacenamiento de Datos
N° de Tarea	Nombre de la Tarea
1	Almacenamiento de datos
2	Pruebas de Funcionalidad

Figura 16 Tareas de Ingeniería (Segunda Iteración)

Durante la segunda iteración se desarrolló el **módulo de almacenaje de datos** el cuál asignará una nomenclatura(acordada con el cliente) y una ruta de almacenamiento para la gestión de los datos. A continuación, detallamos las tareas de ingeniería de la iteración actual(Figuras 17 y 18).

Tarea de Ingeniería			
N° de Tarea:	1	N° de Historia:	2
Nombre de la Tarea:	Almacenamiento de Datos		
Tipo de Tarea:	Desarrollo		
Fecha de Inicio: 05/10/20		Fecha de Término: 08/10/20	
Descripción de la Tarea: Se desarrollará un módulo para la captura de imágenes de las personas que no porten cubrebocas, se asignará una nomenclatura al archivo y se almacenará en una ruta específica			

Figura 17 Tarea de Ingeniería 1 (UH-2)

Tarea de Ingeniería			
N° de Tarea:	2	N° de Historia:	2
Nombre de la Tarea:	Pruebas de Funcionalidad		
Tipo de Tarea:	Desarrollo		
Fecha de Inicio: 05/10/20		Fecha de Término: 14/10/20	
Descripción de la Tarea: Se realizarán pruebas de funcionalidad verificando que el sistema capture una imagen de las personas que no porten un cubrebocas, asigne el nombre y ubicación del archivo apropiadamente, el sistema debe funcionar sin interrupciones de lo contrario deben aplicarse sus correcciones			

Figura 18 Tarea de Ingeniería 2 (UH-2)

Tercera Iteración

Para determinar el(los) objetivo(s) de la iteración actual nos apoyamos de la historia de usuario correspondiente, con la cuál podemos generar las tareas de ingeniería pertinentes para su realización(Figura 19).

HU-3	Reconocimiento de Personas
N° de Tarea	Nombre de la Tarea
1	Detección de rostros
2	Aprendizaje y entrenamiento del identificador de usuarios
3	Identificación de usuarios
4	Pruebas de Funcionalidad

Figura 19 Tareas de Ingeniería (Tercera Iteración)

Durante la tercera iteración se desarrolló el **módulo de identificación de usuarios** mediante un algoritmo implementando el algoritmo "*haarcascade"* de OpenCV para el entrenamiento y reconocimiento de personas en el sistema. A continuación, detallamos las tareas de ingeniería de la iteración actual(Figuras 20-23).

Tarea de Ingeniería			
N° de Tarea:	1	N° de Historia:	3
Nombre de la Tarea:	Detección de Rostros		
Tipo de Tarea:	Desarrollo		
Fecha de Inicio: 19/10/20		Fecha de Término: 20/10/20	
Descripción de la Tarea: Se desarrollará un módulo de detección de rostros mediante OpenCV para enfocar el procesamiento de información			

Figura 20 Tarea de Ingeniería 1 (UH-3)

Tarea de Ingeniería			
N° de Tarea:	2	N° de Historia:	3
Nombre de la Tarea:	Aprendizaje y entrenamiento del identificador de usuarios		
Tipo de Tarea:	Desarrollo		
Fecha de Inicio: 20/10/20		Fecha de Término: 22/10/20	
Descripción de la Tarea: Se desarrollará un módulo de aprendizaje y entrenamiento del identificador de usuarios mediante OpenCV para que el sistema sea capaz de reconocer a los empleados que no porten cubrebocas			

Figura 21 Tarea de Ingeniería 2 (UH-3)

Tarea de Ingeniería			
N° de Tarea:	3	N° de Historia:	3
Nombre de la Tarea:	Identificación de Usuarios		
Tipo de Tarea:	Desarrollo		
Fecha de Inicio: 23/10/20		Fecha de Término: 27/10/20	
Descripción de la Tarea: Se desarrollará un módulo de reconocimiento de usuarios empleando el modelo obtenido en la tarea previa para la identificación de los usuarios registrados.			

Figura 22 Tarea de Ingeniería 3 (UH-3)

Tarea de Ingeniería			
N° de Tarea:	4	N° de Historia:	3
Nombre de la Tarea:	Pruebas de Funcionalidad		
Tipo de Tarea:	Prueba		
Fecha de Inicio: 28/10/20		Fecha de Término: 30/10/20	
Descripción de la Tarea: Se realizarán pruebas de funcionalidad verificando que el sistema reconozca e identifique los rostros de las personas sin cubrebocas, el sistema debe funcionar sin interrupciones de lo contrario deben aplicarse sus correcciones pertinentes.			

Figura 23 Tarea de Ingeniería 4 (UH-3)

Cuarta Iteración

Para determinar el(los) objetivo(s) de la iteración actual nos apoyamos de la historia de usuario correspondiente, con la cuál podemos generar las tareas de ingeniería pertinentes para su realización(Figura 24).

HU-1	Mensaje de Voz al Usuario
N° de Tarea	Nombre de la Tarea
1	Voz del Sistema
2	Base de Datos
3	Pruebas de Funcionalidad

Figura 24 Tareas de Ingeniería (Cuarta Iteración)

Durante la cuarta iteración se desarrolló el **módulo de mensaje de Voz del y Base de Datos** para enviar un mensaje de aviso al personal, además de enviar determinada información(acordada con el cliente) a una base de datos. A continuación, detallamos las tareas de ingeniería de la iteración actual(Figuras 25-27).

Tarea de Ingeniería			
N° de Tarea:	1	N° de Historia:	4
Nombre de la Tarea:	Mensaje de Voz al Usuario		
Tipo de Tarea:	Desarrollo		
Fecha de Inicio: 02/11/20		Fecha de Término: 6/11/20	
Descripción de la Tarea: Se desarrollará un módulo de voz en python para dar mensaje de aviso a las personas que no porten el cubrebocas			

Figura 25 Tarea de Ingeniería 1 (UH-4)

Tarea de Ingeniería			
N° de Tarea:	2	N° de Historia:	4
Nombre de la Tarea:	Base de Datos		
Tipo de Tarea:	Desarrollo		
Fecha de Inicio: 09/11/20		Fecha de Término: 11/11/20	
Descripción de la Tarea: Se enviara la información correspondiente del usuario a una base de datos			

Figura 26 Tarea de Ingeniería 2 (UH-4)

Tarea de Ingeniería			
N° de Tarea:	3	N° de Historia:	4
Nombre de la Tarea:	Pruebas de Funcionalidad		
Tipo de Tarea:	Prueba		
Fecha de Inicio: 12/11/20		Fecha de Término: 16/11/20	
Descripción de la Tarea: Se realizarán pruebas de funcionalidad verificando que el sistema capture los rostros de las personas que no porten cubrebocas, asignando su nombre y ubicación correspondiente, el sistema debe funcionar sin interrupciones de lo contrario deben aplicarse sus correcciones pertinentes.			

Figura 27 Tarea de Ingeniería 3 (UH-4)

Cuando surgían dudas en el proyecto, se realizaban avances en las tareas de ingeniería o se finalizaba una historia de usuario se consultaba al cliente(**disponibilidad del cliente**), el cual nos daba una retroalimentación sobre lo que debía corregirse, lo que debería mejorarse y lo que quería en específico como el nombre de las imágenes, los datos que quería que se mandarán a la base de datos, los mensajes de voz hacia los infractores y las posibles adiciones que podrían plantearse al sistema.

Cuando la iteración era culminada y aprobada por el cliente, se añadía al código principal, esto forma parte del proceso llamado **integración continua,** pero no se podía incluir "directamente", debía adaptarse de manera apropiada, de manera en la cual el sistema se "refinaba" y mejoraba en cuestión de funcionalidad y "estética" del código haciéndolo de manera óptima, a este proceso se le conoce como **refactorización**.

Los **estándares de programación** fueron definidos por el desarrollador, centrándose en conseguir una fácil comprensión para futuros cambios, algunas de los estándares fueron:

- Nombres de las funciones y variables, siendo concisos con la información que almacenan.
- Identación y espaciado del código, para una lectura fácil del código, así como una fácil identificación de secciones.
- Comentarios dentro código para no haber pérdidas en su lectura y modificación al mencionar las acciones de determinadas líneas así cómo los títulos de las secciones pertinentes.

3.4 Pruebas

Nuestras pruebas se centraron en la funcionalidad del código, el código se fue testeando conforme se iban agregando módulos o funciones, al funcionar correctamente se guardaba una copia de esa versión.

El propósito de realizar pruebas y respaldos con cada “versión” del sistema fue para observar su evolución, además de tener siempre accesible un respaldo de la última versión funcional del código, ya que al haber constantes modificaciones e incluso en el proceso de refactorización pudieran surgir inconvenientes.

Para ello se realizaron durante cada iteración algunas pruebas de funcionamiento, como se muestran a continuación de forma general en la figura 28 y de manera detallada en las figuras 29-34:

N° de Historia	N° de Prueba	Nombre de la Prueba
1	1	Detección de Rostros
1	2	Detección de Cubrebocas
2	3	Almacenamiento de Datos
3	4	Identificación de Usuarios
4	5	Mensaje al Usuario
4	6	Alta de Datos

Figura 28 Pruebas de Aceptación (vista general)

UH-1	Caso de Prueba	N° 1
Nombre de la Prueba:	Detección de Rostros	
Condiciones de ejecución: 1 El usuario debe estar en un espacio con buena iluminación. 2 El usuario debe estar a una distancia entre 30cm y 3 mts de la cámara para ser detectado por el sistema.		
Pasos de Ejecución: 1 El usuario camina enfrente de la cámara. 2 La camara detecta el rostro de la persona.		
Resultado(s) Esperado(s): 1 El sistema dibuja un rectángulo verde alrededor del rostro detectado.		
Evaluación de la Prueba:	Exitosa	

Figura 29 Caso de Prueba “Detección de Rostros”

UH-1	Caso de Prueba	N° 2
Nombre de la Prueba:	Detección de Cubrebocas	
Condiciones de ejecución: 1 El usuario debe estar en un espacio con buena iluminación. 2 El usuario debe estar a una distancia entre 30cm y 3 mts de la cámara para ser detectado por el sistema.		
Pasos de Ejecución: 1 El usuario camina enfrente de la cámara. 2 La camara detecta el rostro de la persona.		
Resultado(s) Esperado(s): 1 El sistema detecta el rostro de la persona. a Si la persona porta un cubrebocas se dibujará un rectángulo verde alrededor del rostro e incluirá una leyenda que diga "Usa cubrebocas". b Si la persona no porta un cubrebocas se dibujará un rectángulo rojo alrededor del rostro e incluirá una leyenda que diga "No usa cubrebocas".		
Evaluación de la Prueba:	Exitosa	

Figura 30 Caso de Prueba "Detección de Cubrebocas"

UH-2	Caso de Prueba	N° 3
Nombre de la Prueba:	Almacenamiento de Datos	
Condiciones de ejecución: 1 El sistema detectó una persona sin cubrebocas.		
Pasos de Ejecución: 1 El sistema captura el rostro de la persona sin cubrebocas. 2 El sistema asigna una nomenclatura como nombre del archivo.		
Resultado(s) Esperado(s): 1 El sistema almacena de manera satisfactoria el archivo con su respectiva nomenclatura y ubicación.		
Evaluación de la Prueba:	Exitosa	

Figura 31 Caso de Prueba "Almacenamiento de Datos"

UH-3	Caso de Prueba	N° 4
Nombre de la Prueba:	Identificación de Usuarios	
Condiciones de ejecución: 1 El usuario debe estar en un espacio con buena iluminación 2 El usuario debe estar a una distancia entre 30cm y 3 mts de la cámara para ser detectado por el sistema		
Pasos de Ejecución: 1 El usuario camina enfrente de la cámara. 2 La camara detecta el rostro de la persona.		
Resultado(s) Esperado(s): 1 El sistema detecta el rostro de la persona y lo compara a Si el rostro de la persona se encuentra registrada en la base de datos se mostrará una leyenda con su respectivo nombre b Si el rostro de la persona no se encuentra registrada en la base de datos se mostrará una leyenda con el título "Desconocido"		
Evaluación de la Prueba:	Exitosa	

Figura 32 Caso de Prueba "Identificación de Usuarios"

UH-4	Caso de Prueba	N° 5
Nombre de la Prueba:	Mensaje al Usuario	
Condiciones de ejecución: 1 El usuario debe estar en un espacio con buena iluminación 2 El usuario debe estar a una distancia entre 30cm y 3 mts de la cámara para ser detectado por el sistema 3 El usuario no porta subrebocas		
Pasos de Ejecución: 1 El usuario camina enfrente de la cámara. 2 La camara detecta el rostro de la persona sin cubrebocas.		
Resultado(s) Esperado(s): 1 El sistema detecta el rostro de la persona y lo compara con su base de datos 2 a Si el rostro de la persona se encuentra registrada en la base de datos se reproducirá un audio dando un aviso dirigido al nombre de la persona identificada 3 b Si el rostro de la persona no se encuentra registrada en la base de datos se reproducirá un audio dando un aviso dirigido al nombre "Desconocido"		
Evaluación de la Prueba:	Exitosa	

Figura 33 Caso de Prueba "Mensaje al Usuario"

UH-4	Caso de Prueba	N° 6
Nombre de la Prueba:	Alta de Datos	
Condiciones de ejecución: 1 El sistema detectó una persona sin cubrebocas.		
Pasos de Ejecución: 1 El sistema captura los datos pertinentes del evento 2 El sistema sube los datos correspondientes a una base de datos		
Resultado(s) Esperado(s): 1 La base de datos almacena la información correspondiente		
Evaluación de la Prueba:	Exitosa	

Figura 34 Caso de Prueba "Alta de Datos"

4. Conclusiones y recomendaciones

Al culminar este proyecto se obtuvo un sistema capaz de detectar si las personas hacen o no uso de un cubrebocas, en caso del incumplimiento el sistema procede a la captura del rostro del infractor, colocando una nomenclatura(fecha, hora y nombre del infractor) como nombre del archivo, asignándolo en una ruta específica y mandando los datos de la nomenclatura a una Base de Datos. Todo esto se realiza en tiempo real y sin interrupciones.

El desarrollo del proyecto fue complicado al inicio por carecer de conocimientos acerca del tema, el desconocer las herramientas apropiadas para el sistema complica en gran medida los primeros pasos sobre todo porque puede ser necesario cambiar de herramienta por no ser la más adecuada en lo que se desarrolla, eso me ocurrió con OpenCV y TensorFlow, algo que ayudó de sobremanera en la realización del proyecto fue la continua investigación en las funciones que se podían aplicar en el proyecto y la experimentación continua con el mismo para obtener los mejores resultados.

Gracias al sistema se logrará tener un mejor control del personal en las medidas preventivas de salud que se requieren hoy en día para aminorar la probabilidad de contagio, ya que determinaremos el momento y las personas que incumplan con estas nuevas medidas de sanidad, teniendo un registro comprobable de que el usuario realiza una falta y poder actuar de manera propia. Ya que con la pandemia actual el uso de EPP es vital para asegurar entornos “libres” de COVID-19 brindado espacios donde los usuarios puedan sentirse seguros.

Recomiendo el uso de TensorFlow y Keras para el procesamiento de video e imágenes además de herramienta para el procesamiento de datos matemáticos ya que combinados ambos proveen una alta velocidad en su procesamiento y bajo consumo de espacio de memoria comparado a otras APIs.

5. Referencias bibliográficas

[1] URL: https://www.who.int/es/emergencies/diseases/novel-coronavirus-2019/advice-for-public/q-a-coronaviruses.Página oficial de la OMS, en ella se puede consultar información referente al área de la Salud. Fecha de consulta: 09/Agosto/20

[2] Dannia Colín Castelán. "¿Cubrebocas o no cubrebocas? Esa es la cuestión". CIENCIORAMA, México, 2020

[3] Jing Yan, Suvajyoti Guha, Prasanna Hariharan, Matthew Myers. "Modeling the Effectiveness of Respiratory Protective Devices in Reducing Influenza Outbreak". Society for Risk Analysis, 2018

[4] Guillermo Foladori, Raúl Delgado Wise. Para comprender el impacto disruptivo de la covid-19,un análisis desde la crítica de la economía política. Universidad Autónoma de Zacatecas.2020

[5] Gobierno de la ciudad de México. LINEAMIENTOS DE MEDIDAS DE PROTECCIÓN A LA SALUD QUE DEBERÁN CUMPLIR LOS CENTROS COMERCIALES PARA REANUDAR ACTIVIDADES HACIA UN REGRESO SEGURO A LA NUEVA NORMALIDAD EN LA CIUDAD DE MÉXICO. Gobierno de la ciudad de México, 2020. Fecha de consulta 07/Octubre/10 URL: https://medidassanitarias.covid19.cdmx.gob.mx/

[6] Agustín Zerón. "Nueva normalidad, nueva realidad". Revista ADM, México, 2020. Pág.120-123

[7] Roger S. Pressman. Ingeniería del Software un enfoque práctico 7ma Edición. The McGraw-Hill, México, 2010

[8] Maida, EG, Pacienzia, J. Metodologías de desarrollo de software. Universidad Católica Argentina, 2015

[9] José Bautista. PROGRAMACIÓN EXTREMA (XP). Universidad Unión Bolivariana. 2018

[10] Maida, Esteban Gabriel, Pacienzia, Julián. Metodologías de desarrollo de software. Universidad Católica de Argentina. Argentina. 2015

[11] Beck Kent, Extreme Programming Explained: Embrace Change 2a. Edición. Addison-Wesley. 2004.

[12] José Joskowicz. Reglas y Prácticas en eXtreme Programming. Universidad de Vigo, España. 2008

[13] Patricio Letelier, María del Carmen Penadés. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). Ciencia y Técnica Administrativa – CyTA. Buenos Aires, Argentina. 2006 URL: http://www.cyta.com.ar/ta0502/v5n2a1.htm

[14] URL: www.extremeprogramming.org/rules/unittests.html. Página web donde se encuentra información pertinente a la metodología XP. Fecha de consulta 29/Oct/20

[15] URL: https://iswugaps2extremeprogramming.wordpress.com/2015/09/14/ventajas-y- desventajas/ Página web perteneciente al Instituto Tecnológico de Sonora. Fecha de consulta 3/Nov/20

[16] URL: https://blog.comparasoftware.com/programacion-extrema-ventajas-desventajas/ Página web perteneciente a la empresa ComparaSoftware dedicada a la conexión de usuarios con proveedores de software, además de proporcionar información correspondiente de los sistemas que facilita. Fecha de consulta 3/Nov/20

[17] URL: https://www.python.org/ Página web oficial de Python, en ella se encuentran diversas versiones y descargas disponibles del software además de su información. Fecha de consulta 29/Oct/20

[18] Raúl González Duque. Python PARA TODOS. Creative Commons Reconocimiento 2.5 España. 2017

[19] URL: https://blogueropro.com/blog/ventajas-y-desventajas-de-usar-python-en-la-programacion-web. Página web que alberga diversidad de blogs informativos, culturales y tecnológicos para todos. Fecha de consulta 27/Octubre/20

[20] Cristian Domínguez. FacesDetector: Aplicación práctica de machine learning sobre imágenes para un contexto de seguridad. Universidad de Extremadura.2018

[21]URL: https://opencv.org/ Página web oficial de OpenCV, en ella se encuentran diversas versiones y descargas disponibles del software además de su información. Fecha de consulta 29/Oct/20

[22] URL: https://www.tensorflow.org/ Página web oficial de TensorFlow, en ella se encuentran diversas versiones y descargas disponibles del software además de su información. Fecha de consulta 29/Oct/20

[23] URL: https://es.mort-sure.com/blog/keras-vs-tensorflow-90977b/ Fecha de consulta 29/Oct/20

[24] URL: https://keras.io/ Página web oficial de Keras, en ella se encuentran diversas versiones y descargas disponibles del software además de su información. Fecha de consulta 29/Oct/20

Universidad Politécnica de Puebla
Ingeniería en Informática

Jesús David Rodríguez Escobedo
Caldera Miguel Javier
Rebeca Rodriguez Huesca