CONSTRUCCION E IMPLEMENTACION DE UN PROTOTIPO PARA RECONOCIMIENTO E IDENTIFICACION DE CLIENTES BANCARIOS USANDO TECNICAS DE RECOGNITION Y DEEP LEARNING

OSCAR JUNIOR TACHA GUTIERREZ

DIRIGIDO POR:

SANDRO JAVIER BOLAÑOS CASTRO

JORGE ULISES USECHE CUELLAR

UNIVERSIDAD DISTRITAL FRANCISCO JOSE DE CALDAS

FACULTAD INGENIERIA

BOGOTA D.C.

2021

CONSTRUCCION E IMPLEMENTACION DE UN PROTOTIPO PARA RECONOCIMIENTO E IDENTIFICACION DE CLIENTES BANCARIOS USANDO TECNICAS DE RECOGNITION Y DEEP LEARNING



OSCAR JUNIOR TACHA GUTIERREZ

TRABAJO DE GRADO PARA OPTAR AL TITULO DE

INGENIERO DE SISTEMAS

DIRIGIDO POR:

SANDRO JAVIER BOLAÑOS CASTRO

JORGE ULISES USECHE CUELLAR

REVISOR:

PAOLO ALEJANDRO DAZA CORREDOR

UNIVERSIDAD DISTRITAL FRANCISCO JOSE DE CALDAS

FACULTAD INGENIERIA

BOGOTA D.C.

2021

Resumen

A continuación, se encuentra la especificación del proyecto que será realizado en la empresa Banco De Bogotá, detallando el alcance y los objetivos de este, que tiene como tema principal el estudio de como reconocer e identificar a las personas al interior del banco de Bogotá utilizando técnicas de recognition y Deep learning ofreciendo un saludo si es un cliente del Banco de Bogotá y ofreciéndole obtener un producto si no es cliente beneficiando al banco con una solución propietaria y al usuario ofreciendo otro medio de identificación.

TABLA DE CONTENIDOS

Resumen *3*

Capítulo 1 ..... *11*

Proyecto **11**

1.1 Introducción 11

1.2 Problema de investigación 12

1.2.1 Planteamiento del problema 12

1.2.2 Pregunta problema 12

1.2.3 Formulación Del Problema 13

1.3 Objetivos 14

1.3.1 Objetivo general de la investigación 14

1.3.2 Objetivos específicos 14

1.4 Alcances 15

1.5 Limitaciones 16

Capitulo 2 ..... *17*

Marco referencial **17**

2.1 Marco teórico 17

2.2 Sistemas de gestión de identidad y acceso, o por sus siglas en ingles identity and access management (IAM): 18

2.3 Computación en la nube o cloud computing: 19

2.3.1 Características esenciales 19

2.3.1.1 Autoservicio bajo demanda 19

2.3.1.2 Amplio acceso a la red 19

2.3.1.3 Agrupación de recursos 19
2.3.1.4 Elasticidad rápida 20
2.3.1.5 Servicio medido 20
2.3.3 Modelos de servicio 20
2.3.3.1 Software como servicio (SaaS) 20
2.3.3.2 Plataforma como servicio (PaaS) 21
2.3.3.3 Infraestructura como servicio (IaaS) 21
2.3.4 Modelos de implementación 22
2.3.4.1 Nube privada 22
2.3.4.2 Nube comunitaria 22
2.3.4.3 Nube publica 22
2.3.4.4 Nube híbrida 23
2.4 Machine Learning 24
2.4 Recognition 25
2.5 Server-sent Events (SSE) 26
2.6 Websocket 27
2.7 Angular Framework 28
2.8 Estado del Arte 29
2.8.1 DLIB 30
2.9 GitHub 31
Capitulo 3 ..... ***32***
Técnica metodológica **32**
3.1 Investigación 32
3.1.1 Tipo de investigación 32
3.1.2 Diseño de Investigación 32
3.2 Metodología 34

3.2.1 Metodología cuantitativa 34
3.2.2 Metodología base 35
3.2.3 Marco de trabajo 35
3.2.3 Herramientas para manejo del marco de trabajo 37
Capitulo 4 39
Planeación 39
4.1 Participantes 39
4.2 Presupuesto 40
4.3 Recursos 42
4.4 Cronograma 43
4.4.1 Fase de inicio 43
4.4.2 Fase de elaboración: 44
4.4.3 Fase de Construcción: 44
4.4.4. Fase de transición: 45
Capitulo 5 46
Diseño y arquitectura 46
5.1 Arquitectura 46
5.1.1 Modelo 4+1 47
5.1.2 Vista lógica 49
5.1.2.1 Diagrama de clases AWS DeepLens 49
5.1.2.2 Diagrama de clases suscriptor de eventos 50
5.1.2.3 Diagrama de clases publicador de eventos 51
5.1.2.3 Diagrama de clases para la interfaz grafica. 52
5.1.3 Vista de desarrollo 53
5.1.4 Vista de proceso 54

5.1.5 Vista Física Topología Hardware 55

5.1.6 Escenarios 56

5.1.6.1. Partes de los escenarios propuestos: 56

5.1.6.1.1 Sistema 56

5.1.6.1.2 Actores 56

5.1.6.1.3 Casos de Uso: 56

5.1.6.2 Escenario Identificación facial 57

5.1.6.3 Escenario de registro de usuario nuevo. 58

Capitulo 6 59

Preparación del proyecto 59

6.1 Tecnologías a utilizar 59

6.1.1 JavaScript 59

6.1.2 HTML 59

6.1.3 Python 59

6.1.4 DLIB 60

6.1.5 Typescript 60

6.1.6 AWS-SDK 60

6.1.7 Boto3 60

6.2 Arquitectura de referencia 61

6.3 Creación del bucket S3 62

6.4 Diseño de la Interfaz grafica 63

6.5 Diagrama de secuencia para la visualización de resultados del prototipo de identificación de clientes 64

Capitulo 7 65

Desarrollo del proyecto 65

7.1 Obtención y guardado de metadatos faciales 65

7.2 Obtención de metadatos de rostros conocidos e indexados guardados en Bucket S3 66

7.3 Configuración del dispositivo de procesamiento de imágenes (AWS DeepLens). 67

7.3.1 Identificación de la red SSID y contraseña del dispositivo AWS DeepLens 67

7.3.2 Como conectar un equipo de escritorio (Computador) al WI-FI generado por el dispositivo AWS DeepLens 68

7.3.3 Configurar el dispositivo AWS DeepLens para conexión WI-FI 69

7.3.4 Configuración del dispositivo AWS DeepLens para asociarlo con una cuenta de Amazon Web Services. 70

7.3.4.1 Conectar el dispositivo AWS DeepLens a internet. 70

7.3.4.2 Configuración del dispositivo DeepLens en una cuenta de Amazon Web Services. 71

7.4.3.3 Verificación de la configuración del dispositivo DeepLens 72

7.4 Creación del proyecto para el dispositivo AWS DeepLens 73

7.4.1 Instalación de dependencias en el dispositivo AWS DeepLens 73

7.4.2 Creación de la función AWS Lambda de inferencia. 74

7.4.2.1 Generación del código para la función de inferencia 74

7.4.2.2 Creación de una función AWS lambda 75

7.4.2.3 Subir el proyecto de implementación anteriormente generado 76

7.4.2.4 Generación una nueva versión del proyecto de implementación 77

7.5 Implementación y despliegue del proyecto en el dispositivo AWS DeepLens 79

7.5.1 Creación del proyecto de implementación para desplegar en el dispositivo AWS DeepLens 79

7.5.2 Despliegue en el dispositivo AWS DeepLens del proyecto anteriormente creado 82

7.5.3 Verificación de la implementación del proyecto en el dispositivo 85

7.6 Generación del WebSocket en AWS API Gateway 86

7.7 Generación de la interfaz grafica 93

7.8 Integración continua 94

7.8.1 Calidad de código 94

Capitulo 8 ..... *98*

Análisis de los Resultados **98**

8.1 Resultados obtenidos por el dispositivo DeepLens 98

8.2 Beneficios obtenidos 100

Capitulo 9 ..... *101*

Conclusiones **101**

9.1 Verificación, contraste y evaluación de los objetivos 101

9.2 Síntesis del modelo propuesto 102

9.3 Aportes originales 102

9.4 Trabajos o publicaciones derivadas 104

Capitulo 10 ..... *105*

PROSPECTIVA DEL TRABAJO DE GRADO **105**

10.1 Líneas de investigación futuras 105

10.2 Trabajos de investigación futuros 105

Referencias *107*

Ilustración 1: Diagrama SSE _____ *26*

Ilustración 2: Diagrama WebSocket _____ *27*

Ilustración 3: Diagrama de flujo de la metodología SCRUM _____ *38*

Ilustración 4: Diagrama de Gantt del proyecto. _____ *43*

Ilustración 5: Diagrama de Gantt para la fase de inicio del proyecto. _____ *43*

Ilustración 6: Diagrama de Gantt para la fase de elaboración del proyecto. _____ *44*

Ilustración 7: Diagrama de Gantt para la fase de construcción del proyecto. _____ *44*

Ilustración 8: Diagrama de Gantt para la fase de transición del proyecto. _____ *45*

Ilustración 9: Diagrama vista 4+1. _____ *48*

Ilustración 10: Diagrama de clases para el dispositivo AWS DeepLens. _____ *49*

Ilustración 11: Diagrama de clases del componente subscriptor de eventos. _____ *50*

Ilustración 12: Diagrama de clases del componente publicador de eventos. _____ *51*

Ilustración 13: Diagrama de clases para la interfaz grafica. _____ *52*

Ilustración 14: Diagrama de componentes del prototipo de identificación de clientes. _____ *53*

Ilustración 15: Diagrama de flujo o actividades del prototipo de identificación de clientes. _____ *54*

Ilustración 16: Vista física topología del hardware del prototipo de identificación de clientes. _____ *55*

Ilustración 17: Escenario identificación facial de clientes. _____ *57*

Ilustración 18: Escenario registro de usuario nuevo. _____ *58*

Ilustración 19: Arquitectura de referencia del prototipo. _____ *61*

Ilustración 20: Diagrama de Amazon Web Services (AWS) S3. _____ *62*

Ilustración 21: Diseño de interfaz gráfica. _____ *63*

Ilustración 22: Diagrama de secuencia de la interfaz grafica. _____ *64*

Ilustración 23: Identificación de la red SSID y contraseña del dispositivo AWS DeepLens. _____ *67*

Ilustración 24: Conexión entre el ordenador y el dispositivo DeepLens vía WI-FI. _____ *68*

Ilustración 25: Como Configurar el dispositivo DeepLens para conexión WI-FI. _____ *69*

Ilustración 26: Interfaz de configuración inicial dispositivo AWS DeepLens _____ *70*

Ilustración 27: Interfaz de Configuración del dispositivo DeepLens en una cuenta de Amazon Web Services. _____ *71*

Ilustración 28: Interfaz de Verificación del estado del dispositivo DeepLens en una cuenta de Amazon Web Services. _____ *72*

*Ilustración 29: Interfaz de creación de función lambda en la consola de administración de cuenta de Amazon Web Services.*_____ *75*

*Ilustración 30: Interfaz de como subir un proyecto de código comprimido a una función AWS Lambda en consola de administración AWS.*_____ *76*

*Ilustración 31: Interfaz de como generar una nueva versión de una función AWS Lambda en consola de administración AWS.*_____ *77*

*Ilustración 32: Interfaz de como generar una nueva versión de una función AWS Lambda en consola de administración AWS.*_____ *78*

Ilustración 33: Creación del proyecto de implementación para desplegar en el dispositivo DeepLens. _____ *79*

Ilustración 34: Elección tipo de proyecto de implementación para AWS DeepLens. _____ *80*

Ilustración 35: Especificación de detalles del proyecto de implementación en el dispositivo AWS DeepLens. _____ *81*

*Ilustración 36: Despliegue del proyecto de implementación en el dispositivo AWS DeepLens.*_____ *82*

*Ilustración 37: Implementar un proyecto en el dispositivo AWS DeepLens.*_____ *83*

Ilustración 38: Selección del proyecto anteriormente creado para el dispositivo DeepLens. _____ *84*

Ilustración 39: Verificación de la implementación del proyecto en el dispositivo DeepLens. _____ *85*

Ilustración 40: Búsqueda de API Gateway en la consola de administración de Amazon Web Services. _____ *86*

Ilustración 41: Creación de API WebSocket en la consola de administración de Amazon Web Services. _____ *87*

Ilustración 42: Selección tipo de API a crear en la consola de administración de Amazon Web Services. _____ *88*

*Ilustración 43: Especificación de detalles del API a crear en la consola de Amazon Web Services.*_____ *89*

*Ilustración 44: Configuración de rutas para el API de WebSocket en la consola de administración de Amazon Web Services.*_____ *90*

Ilustración 45: Agregar Etapa de implementación de API WebSocket en la consola de administración de Amazon Web Services. _____ *91*

Ilustración 46: Creación final del API de WebSocket en la consola de administración de Amazon Web Services. __ *92*

Ilustración 47: Interfaz grafica final para mostrar los resultados al usuario. _____ *93*

Ilustración 48: Integración continua de los componentes del prototipo de identificación de clientes. _____ *94*

Ilustración 49: Resultados del análisis generado por SonarCloud para el prototipo de identificación de clientes. __ *95*

Ilustración 50: Resultado del análisis de Pylint sobre los componentes escritos en el lenguaje Python. _____ *97*

Ilustración 51: Resultado del análisis de ESLint sobre los componentes escritos en el lenguaje TypeScript. _____ *97*

Ilustración 52: Procesamiento de imágenes en el dispositivo AWS DeepLens en tiempo real y su salida en un video de tiempo real identificando los rostros conocidos. _____ *98*

Ilustración 53: Resultados obtenidos por el dispositivo DeepLens. _____ *99*

Tabla 1: Participantes del proyecto. _____ *39*

*Tabla 2: Presupuesto del proyecto.*_____ *40*

Tabla 3: Recursos del proyecto. _____ *42*

Capítulo 1

Proyecto

1.1 Introducción

A través del desarrollo del presente proyecto se plantea dar una alternativa de solución al problema de reconocimiento de clientes dentro de una oficina bancaria. La propuesta consta de un prototipo el cual tiene la capacidad de reconocer e identificar personas, registrando métricas importantes para el caso de estudio y con las mismas mejorar la confiabilidad en la identificación. El prototipo permitirá al usuario ser saludado por su nombre siempre y cuando sea un cliente del Banco de Bogotá.

El reconocimiento inequívoco de personas es una demanda constante en el mundo actual, un mundo globalizado, por lo cual es necesario el uso de la biometría, ciencia, la cual provee herramientas y métodos para realizar una exploración sobre las características de un individuo. "La biométrica es la ciencia de establecer la identidad de un individuo basada en los atributos físicos, químicos o de comportamiento de la persona" (Ross & Anil K, 2008). Los Sistemas de gestión de identidad también conocidos como IAM (por sus siglas en inglés (Identity and access management) "La gestión de identidad es el proceso organizativo para identificar, autenticar y autorizar a individuos o grupos de personas a tener acceso a aplicaciones, sistemas o redes asociando los derechos y restricciones de los usuarios con las identidades establecidas. Las identidades administradas también pueden referirse a procesos de software que necesitan acceso a sistemas organizacionales." (Rouse, 2019), estos sistemas están basados en técnicas de Deep Learning y Recognition.

1.2 Problema de investigación

1.2.1 Planteamiento del problema

En Colombia se funda el Banco de Bogotá el 15 de noviembre de 1870 constituida como la primera entidad bancaria, la cual continúa creando sucursales bancarias a lo largo de toda su historia. Estas sucursales permiten la interacción directa entre funcionarios del Banco y el usuario, debido a la alta demanda de estos servicios, el brindar una atención personalizada basada en el perfil del usuario con diversas variables para la entidad bancaria se vuelve una necesidad primaria. (Banco de Bogotá, 2012). Debido a la alta demanda de productos bancarios y a el constante aumento del fraude y la suplantación de identidad (Transunion, 2020) el Banco de Bogotá ha estudiado la implementación de sistemas de autenticación por biometría facial al interior de sus instalaciones con un proveedor externo de pero el alto costo por cada autenticación de cada usuario lo hace insustentable por ende se propone el presente proyecto para lograr la reducción de costos donde la principal diferencia se basa en cambiar un costo por autenticación de cada usuario a un costo de mantenimiento de un sistema.

1.2.2 Pregunta problema

¿Es posible implementar un sistema de reconocimiento facial para identificar clientes bancarios usando técnicas de Deep Learning y Recognition realizando un piloto controlado en las instalaciones del Banco de Bogotá específicamente en el laboratorio digital?

1.2.3 Formulación Del Problema

¿Cuáles son las variables que ayudan a realizar el reconocimiento facial e identificación de clientes, para brindar un saludo con su nombre al cliente, con el objetivo de mejorar la experiencia al interior de las oficinas del banco y así mismo se fortalezca el vínculo entre el cliente y el banco?

1.3 Objetivos

1.3.1 Objetivo general de la investigación

Construir e implementar un prototipo para reconocimiento e identificación de clientes bancarios, con un porcentaje de confiabilidad mayor al 80% (ochenta por ciento) usando técnicas de recognition y Deep learning con una infraestructura en la nube.

1.3.2 Objetivos específicos

- Realizar una vista en entorno web (Frontend) a través de la herramienta Angular, para poder visualizarse en cualquier dispositivo (responsive) con acceso a internet.
- Implementar una arquitectura cloud en la solución para que se accesible desde cualquier dispositivo con acceso a internet.
- Implementar un modelo de reconocimiento facial para poder procesar imágenes de las clientes obtenidas desde un dispositivo de captura de fotogramas.

1.4 Alcances

El presente estudio plantea la identificación de las personas en el interior del Laboratorio Digital del Banco de Bogotá con un porcentaje de fiabilidad mayor al 80% obteniendo y visualizando correctamente su nombre en un dispositivo con acceso a internet.

El entregable al finalizar este proyecto, es una herramienta de software, que sea capaz reconocer a una persona a través de biometría facial, conectándose a una base de datos alojada en Amazon Web Services (AWS) para obtener metadatos faciales de las personas registradas, extrayendo su documento de identificación para así poder recuperar los datos guardados de la persona si es cliente brindando un saludo con el nombre del individuo. Esta herramienta contará con estándares de calidad de código que permitirá su fácil modificación y extensión, además de ser stateless para permitir su implementación en diferentes oficinas del banco de Bogotá.

1.5 Limitaciones

- Datos faciales no recolectados y no indexados presentan una limitación en cuanto a tiempo ya que registrar más de diecinueve millones (19.000.000) de clientes activos (Banco de Bogotá, 2019).
- Costo de la capacidad de cómputo para realizar entrenamiento de modelo de reconocimiento facial debido a que entre más precisión se requiera el tiempo crecerá logarítmicamente (Chang Kai-Wei, 2008) y entre más imágenes de muestra se tengan en el dataset el tiempo crecerá exponencialmente (Soliman Mona, 2020) tomando como constante que el tiempo siempre incrementa y el costo de los equipos de entrenamiento se basa en el tiempo que se emplean significa que el costo estará en constante crecimiento.
- El tiempo para el desarrollo del proyecto se comprende de seis meses.

Capitulo 2

Marco referencial

2.1 Marco teórico

El Banco de Bogotá tiene como filosofía una atención centrada en el usuario para lo cual genera día a día diferentes métodos para mejorar la experiencia al usuario, una de ellas es la planteada en este proyecto la cual consiste en poder identificar al usuario a través de biometría facial y brindar un saludo en su etapa inicial lo cual puede seguir creciendo y ser utilizado para múltiples fines, también con el fin de mejorar fortalecer la relación cliente Banco. Se deben abordar la problemática de realizar una identificación inequívoca del usuario, por el ámbito en el cual estará la solución, un entorno web para ser usado en las oficinas del Banco de Bogotá, Aunque el alcance del presente proyecto va dirigido al laboratorio Digital.

Usando las mejores practicas de desarrollo para ofrecer al usuario final una aplicación segura y de calidad con posibilidad de ser extensible y ser trabajada por marcos iterativos.

2.2 Sistemas de gestión de identidad y acceso, o por sus siglas en ingles identity and access management (IAM):

La gestión de identidad y acceso se refiere a todas las políticas y herramientas utilizadas por los departamentos encargados de las tecnologías de la información y comunicación, para de esta manera garantizar que las personas tengan acceso solamente a los mínimos recursos para desarrollar su rol dentro de la organización.

El control de acceso no solo aplica para recursos tecnológicos, si no para recursos físicos como lo es el control de acceso a las instalaciones de la compañía haciendo posible que se tenga un control sobre las personas que están en una determinada instalación de alguna compañía.

2.3 Computación en la nube o cloud computing:

La computación en la nube es un modelo para permitir el acceso a de red ubicuo, conveniente y bajo demanda a un grupo de recursos informáticos configurables como lo son: redes, servidores, aplicaciones, almacenamiento y servicios que se pueden liberar o aprovisionar de manera rápida con un esfuerzo mínimo al momento de administrarlos o interactuar con el proveedor de estos servicios en Cloud computing, el modelo se compone de cinco características esenciales, tres modelos de servicios y cuatro modelos de implementación de la siguiente manera (National Institute of standarts and technology NIST, 2011) :

2.3.1 Características esenciales

2.3.1.1 Autoservicio bajo demanda

Un consumidor puede aprovisionar unilateralmente capacidades informáticas, como la hora del servidor y el almacenamiento en red, según sea necesario automáticamente sin requerir la interacción humana con cada proveedor de servicios.

2.3.1.2 Amplio acceso a la red

Las capacidades están disponibles a través de la red y se accede a ellas a través de mecanismos estándar que promueven el uso de plataformas heterogéneas de clientes delgados o gruesos (por ejemplo, teléfonos móviles, tabletas, computadoras portátiles y estaciones de trabajo).

2.3.1.3 Agrupación de recursos

Los recursos informáticos del proveedor se agrupan para servir a múltiples consumidores utilizando un modelo de múltiples inquilinos, con diferentes recursos físicos y virtuales asignados y reasignados dinámicamente de acuerdo con la demanda del consumidor. Existe una sensación de independencia de ubicación en el sentido de que el cliente generalmente no tiene control o

conocimiento sobre la ubicación exacta de los recursos proporcionados, pero puede especificar la ubicación en un nivel más alto de abstracción (por ejemplo, país, estado o centro de datos). Ejemplos de recursos incluyen almacenamiento, procesamiento, memoria y ancho de banda de red.

2.3.1.4 Elasticidad rápida

Las capacidades se pueden aprovisionar y liberar elásticamente, en algunos casos automáticamente, para escalar rápidamente hacia afuera y hacia adentro de acuerdo con la demanda. Para el consumidor, las capacidades disponibles para el aprovisionamiento a menudo parecen ser ilimitadas y pueden ser apropiadas en cualquier cantidad en cualquier momento.

2.3.1.5 Servicio medido

Los sistemas en la nube controlan y optimizan automáticamente el uso de recursos aprovechando una capacidad de medición1 en algún nivel de abstracción apropiado para el tipo de servicio (por ejemplo, almacenamiento, procesamiento, ancho de banda y cuentas de usuario activas). El uso de los recursos se puede monitorear, controlar e informar, proporcionando transparencia tanto para el proveedor como para el consumidor del servicio utilizado.

2.3.3 Modelos de servicio

- Software como servicio (SaaS)
- Plataforma como servicio (PaaS)
- Infraestructura como servicio (IaaS)

2.3.3.1 Software como servicio (SaaS)

La capacidad que se brinda al consumidor es utilizar las aplicaciones del proveedor que se ejecutan en una infraestructura en la nube.

Se puede acceder a las aplicaciones desde varios dispositivos cliente a través de una interfaz de cliente ligero, como un navegador web (por ejemplo, correo electrónico basado en la web) o una interfaz de programa. El consumidor no gestiona ni controla la infraestructura de la nube subyacente, incluida la red, los servidores, los sistemas operativos, el almacenamiento o incluso las capacidades de las aplicaciones individuales, con la posible excepción de la configuración de la aplicación limitada específica del usuario.

2.3.3.2 Plataforma como servicio (PaaS)

La capacidad proporcionada al consumidor es implementar en la infraestructura de la nube aplicaciones creadas o adquiridas por el consumidor creadas usando lenguajes de programación, bibliotecas, servicios y herramientas compatibles con el proveedor. El consumidor no administra ni controla la infraestructura de nube subyacente, incluida la red, servidores, sistemas operativos o almacenamiento, pero tiene control sobre las aplicaciones implementadas y posiblemente los ajustes de configuración para el entorno de alojamiento de aplicaciones.

2.3.3.3 Infraestructura como servicio (IaaS)

La capacidad que se brinda al consumidor es el aprovisionamiento de procesamiento, almacenamiento, redes y otros recursos informáticos fundamentales donde el consumidor puede implementar y ejecutar software arbitrario, que puede incluir sistemas operativos y aplicaciones. El consumidor no administra ni controla la infraestructura de nube subyacente, pero tiene control sobre los sistemas operativos, el almacenamiento y las aplicaciones implementadas; y posiblemente un control limitado de componentes de red seleccionados (por ejemplo, firewalls de host).

2.3.4 Modelos de implementación

- Nube privada
- Nube comunitaria
- Nube publica
- Nube hibrida

2.3.4.1 Nube privada

La infraestructura de la nube está aprovisionada para uso exclusivo de una sola organización que comprende múltiples consumidores (por ejemplo, unidades de negocios). Puede ser propiedad, ser administrado y operado por la organización, un tercero o alguna combinación de ellos, y puede existir dentro o fuera de las instalaciones.

2.3.4.2 Nube comunitaria

La infraestructura de la nube está provista para uso exclusivo de una comunidad específica de consumidores de organizaciones que tienen preocupaciones compartidas (por ejemplo, misión, requisitos de seguridad, políticas y consideraciones de cumplimiento). Puede ser propiedad, administrada y operada por una o más de las organizaciones de la comunidad, un tercero o alguna combinación de ellas, y puede existir dentro o fuera de las instalaciones.

2.3.4.3 Nube publica

La infraestructura de la nube está aprovisionada para uso abierto por el público en general. Puede ser propiedad, ser administrado y operado por una organización comercial, académica o gubernamental, o alguna combinación de ellos. Existe en las instalaciones del proveedor de la nube.

2.3.4.4 Nube híbrida

La infraestructura de la nube es una composición de dos o más infraestructuras de nube distintas (privadas, comunitarias o públicas) que siguen siendo entidades únicas, pero están unidas por una tecnología estandarizada o patentada que permite la portabilidad de datos y aplicaciones (por ejemplo, explosión de nubes para el equilibrio de carga entre nubes).

2.4 Machine Learning

Machine learning corresponde a un campo de ciencias de la computación, la cual se encarga de extraer conocimiento de una larga cantidad de datos, que no han sido previamente analizados, el objetivo principal de este campo después de haber extraído y entendido la información, es representarla y estructurarla para que se pueda estudiar el comportamiento de los datos.

2.4 Recognition

Recognition es el reconocimiento automatizado de patrones y regularidades en los datos. El reconocimiento de patrones está estrechamente relacionado con la inteligencia artificial y el aprendizaje automático, junto con aplicaciones como la minería de datos y el descubrimiento de conocimiento en bases de datos (KDD), y a menudo se usa indistintamente con estos términos. Sin embargo, estos se distinguen: el aprendizaje automático es un enfoque para el reconocimiento de patrones, mientras que otros enfoques incluyen reglas o heurísticas hechas a mano (no aprendidas); y el reconocimiento de patrones es un enfoque de la inteligencia artificial, mientras que otros enfoques incluyen la inteligencia artificial simbólica. (Bishop, 2006).

2.5 Server-sent Events (SSE)

Server-Sent Events es un estándar que describe cómo los servidores pueden iniciar la transmisión de datos hacia los clientes una vez que se ha establecido una conexión inicial del cliente. Se usan comúnmente para enviar actualizaciones de mensajes o secuencias de datos continuas a un cliente de navegador y están diseñadas para mejorar la transmisión nativa entre navegadores a través de una API de JavaScript llamada EventSource, a través de la cual un cliente solicita una URL particular para recibir una secuencia de eventos. (W3C, 2019).

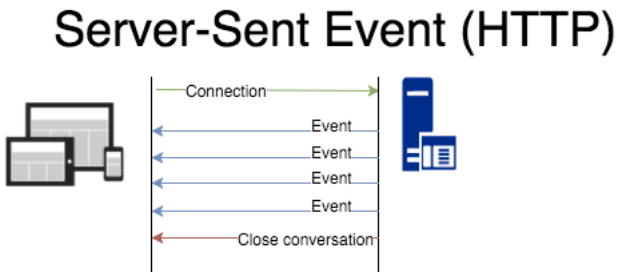


Ilustración 1: Diagrama SSE

Fuente: Extraída de https://medium.com/@bethecodewithyou/server-sent-events-concept-f5d34b3c2ecc

2.6 Websocket

El protocolo WebSocket permite la comunicación bidireccional entre un cliente que ejecuta código no confiable que se ejecuta en un entorno controlado con un host remoto que ha habilitado las comunicaciones de ese código. El modelo de seguridad utilizado para esto es el modelo de seguridad basado en Origin comúnmente utilizado por los navegadores web. El protocolo consiste en un apretón de manos de apertura seguido de un marco básico de mensajes, en capas sobre TCP. El objetivo de esta tecnología es proporcionar un mecanismo para aplicaciones basadas en navegador que necesitan comunicación bidireccional con servidores que no se basan en abrir múltiples conexiones HTTP.

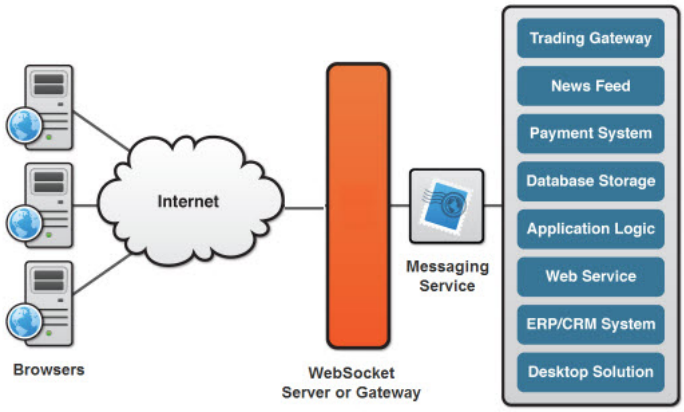


Ilustración 2: Diagrama WebSocket

Fuente: Extraída de https://www.websocket.org/aboutwebsocket.html

2.7 Angular Framework

Angular es un framework opensource desarrollado por Google para facilitar la creación y programación de aplicaciones web de una sola página, las webs SPA (Single Page Application). Angular separa completamente el frontend y el backend en la aplicación, evita escribir código repetitivo y mantiene todo más ordenado gracias a su patrón MVC (Modelo-Vista-Controlador) asegurando los desarrollos con rapidez, a la vez que posibilita modificaciones y actualizaciones. En una web Single Page Application, aunque la velocidad de carga puede resultar un poco lenta la primera vez que se abre, navegar después es totalmente instantáneo, ya que se ha cargado toda la página de golpe.

2.8 Estado del Arte

Basado en una investigación sobre una Red Neural Profunda (DNN) con un sistema de referencia "clásico". Se explota el sistema de código abierto OpenFace para generar un modelo de red neuronal convolucional profundo utilizando conjuntos de datos disponibles públicamente. Se estudia el impacto del tamaño de los conjuntos de datos, su calidad y se compara el rendimiento con un enfoque clásico de reconocimiento facial. El objetivo es tener un modelo totalmente reproducible. Para este fin, se utilizan conjuntos de datos disponibles públicamente (Face Recognition Grand Challenge, MS-celeb-lM, MOBIO, Labeled Faces in the Wild), así como también software disponible públicamente (OpenFace) para entrenar a el modelo a fin de hacer un reconocimiento facial. El modelo mejor entrenado logra un 97.52% de precisión en el conjunto de datos Etiquetado en el conjunto de datos Wild (LFW), que es inferior a los mejores resultados reportados de Google de 99.96% pero ligeramente mejor que el resultado informado de FaceBook de 97.35%. También se evalúa el mejor modelo en el desafiante dataset de video MOBIO y se informan los resultados competitivos con las mejores salidas reportadas en esta base de datos. (Mohamed and Dijana, 2018).

2.8.1 DLIB

El modelo tiene una precisión del 99,38% en los puntos de referencia estándar etiquetados en el Wild. Esto es comparable a otros modelos de vanguardia y significa que, dadas dos imágenes de rostros, predice correctamente si las imágenes son de la misma persona el 99.38% del tiempo. Para aquellos interesados en los detalles del modelo, este modelo es una red ResNet con 29 capas conv. Es esencialmente una versión de la red ResNet-34 del documento Deep Residual Learning for Image Recognition by He, Zhang, Ren y Sun con algunas capas eliminadas y la cantidad de filtros por capa reducida a la mitad. (King Davis, 2017).

2.9 GitHub

GitHub es un sistema de gestión de proyectos y control de versiones de código, así como una plataforma de red social diseñada para desarrolladores. ¿Pero para qué se usa GitHub? Bueno, en general, permite trabajar en colaboración con otras personas de todo el mundo, planificar proyectos y realizar un seguimiento del trabajo. (B. Gustavo, 2019).

Capitulo 3

Técnica metodológica

3.1 Investigación

3.1.1 Tipo de investigación

El presente proyecto propuesto llevará una investigación experimental en donde se estudiará si es posible autenticar a los clientes del Banco de Bogotá a través de biometría por reconocimiento facial mediante un sistema propietario del Banco de Bogotá.

Con el desarrollo del proyecto, se buscará que se conozca más acerca de los clientes que maneja el banco, así ofrecer un mejor servicio y un modo alternativo de autenticación a lo largo del tiempo.

3.1.2 Diseño de Investigación

Los metadatos faciales se extraen de una fotografía de la persona estos son indexados en una base de datos alojada en la nube, se observará como se asocia un rostro a un documento de identidad para poder obtener su nombre de una base de datos.

Se procederá de la siguiente manera.

- Se realizará una captura de fotogramas a través de un dispositivo especialmente diseñado para ello.
- Se compararán los metadatos faciales obtenidos de la captura del fotograma con los metadatos alojados en la base de datos.
- Se busca el índice de fiabilidad mas alto y se obtendrá el numero de identificación asociado al metadato facial.

- Se consulta el documento en la base de datos de clientes del banco (CRM).
- Se obtienen los nombres y apellidos de la persona.
- Se envía una notificación a la interfaz grafica de usuario para que se visualice los nombres y apellidos de la persona.

3.2 Metodología

3.2.1 Metodología cuantitativa

La metodología para implementar deriva de SCRUM la cual consta en Sprints de dos semanas (15 días).

Para cada Sprint, en una ceremonia anterior llamada planeación se definen las Historias de Usuario que serán desarrolladas en los siguientes quince días, cada historia de usuario define una funcionalidad que agrega valor al proyecto, cada historia debe ser independiente, medible y alcanzable en el Sprint a desarrollar, si la historia de usuario no es alcanzable se llama EPIC y se debe subdividir en más historias de usuario. Cada historia de usuario al ser medible debe ser probada exhaustivamente por un tester especializado.

En el transcurso del sprint se realizará la ceremonia de refinamiento que consiste en revisar y solucionar dudas técnicas de las historias a desarrollar en el Sprint siguiente mitigando la incertidumbre y aterrizando a las partes interesadas (StakeHolders) en lo que se realizará en el siguiente Sprint.

Cada día en la ceremonia llamada diario o Daily cada integrante o persona interesada expresara que hizo el día anterior, que realizara el día de hoy y si se presentó algún tipo de impedimento o solicitud en el desarrollo de su rol.

Todas las historias de usuario y su progreso serán observadas y analizadas desde la herramienta JIRA, los Bugs y errores de software serán reportados a través de la herramienta TestLink, el sistema de control de versiones de archivos se realizará en la plataforma GitHub, la automatización de los procesos de despliegue se realizará en la herramienta CircleCi.

3.2.2 Metodología base

La metodología base a utilizar es AGILE para esta metodología se definen tres hitos base:

- Inicio: Se escoge del total de objetivos del producto aquellos que serán implementados en el sprint, debiendo ser capaces de generar un producto funcional. En base a estos objetivos se define la duración del sprint (entre una semana y un mes), y las tareas que lo componen.
- Desarrollo del sprint: El equipo del proyecto planifica y ejecuta las tareas, las cuales se van supervisando en reuniones diarias donde se miran las tareas ejecutadas, en curso, y pendientes, así como posibles impedimentos y restricciones.
- Cierre: Al final del sprint se revisa que se hayan completado las tareas y objetivos definidos al inicio mediante la presentación de un producto funcional. El ensayo de este producto por parte del cliente puede hacer variar los objetivos del proyecto o su prioridad.

3.2.3 Marco de trabajo

Scrum es un marco dentro del cual las personas pueden abordar problemas adaptativos complejos, al tiempo que entregan productos productivos y creativos del mayor valor posible.
Scrum en sí mismo es un marco simple para la colaboración efectiva del equipo en productos complejos. Scrum es simple. Es lo contrario de una gran colección de componentes obligatorios entrelazados. Scrum no es una metodología. Scrum implementa el método científico del empirismo. Scrum reemplaza un enfoque algorítmico programado por uno heurístico, con respeto por las personas y la auto organización para hacer frente a la imprevisibilidad y resolver problemas complejos.

Se trabajará de la siguiente manera:

I. Objetivo

- Realizar una herramienta que reconozca e identifique a los clientes con biometría facial.
- Cumplirlo en un máximo de seis meses.

II. Backlog

- Llamado así, al lugar donde se crearán y se almacenarán todas las tareas necesarias para cumplir con el objetivo.
- Se crearán tareas específicas, que no sean dependientes entre sí, generen valor al desarrollo del proyecto individualmente y se puedan terminar en un periodo de tiempo no mayor a un sprint.

III. Sprint

- Unidad de tiempo en el que se dividirá el trabajo, para cumplir el objetivo y no extenderse más de lo acordado. (6 meses).
- Cada sprint, se debe elegir un número de historias para desarrollar y poder avanzar en el proyecto.
- Se elige un objetivo que ayuda para conseguir el objetivo principal.
- Cada Sprint consta de quince (15) días calendario aproximadamente dos (2) semanas.

IV. Historia

- Trabajo para realizar durante un sprint.
- Reunión de una o más tareas, definidas en el backlog.
- Contiene criterios de aceptación, y casos de prueba los cuales, se utilizan para validar que lo trabajado cumple con lo deseado.

V. Estado de una historia

- A lo largo del sprint, el estado en el que estará una historia va a ir cambiando, esto para controlar el tiempo y poder culminar exitosamente la tarea.
- To-Do: Todas las historias iniciarán en este estado, el cual, hace referencia a que no se han empezado las historias y están pendientes por realizar.
- In-progress: En este estado, las historias están en progreso.
- Test: Después de acabar cada una de las historias, se procede a ejecutar los casos de prueba y pruebas para poder certificar que cumplen con los criterios de aceptación definidos.
- Done: Cuando la historia cumple con los criterios de aceptación, pasa a este estado para llevar un control y poder trabajar en otras historias pendientes.

3.2.3 Herramientas para manejo del marco de trabajo

JIRA:

Los productos y aplicaciones creados sobre la plataforma Jira ayudan a los equipos a planificar, asignar, rastrear, informar y administrar el trabajo. La plataforma Jira reúne a los equipos para todo, desde el desarrollo ágil de software y la atención al cliente hasta la gestión de listas de compras y tareas familiares.

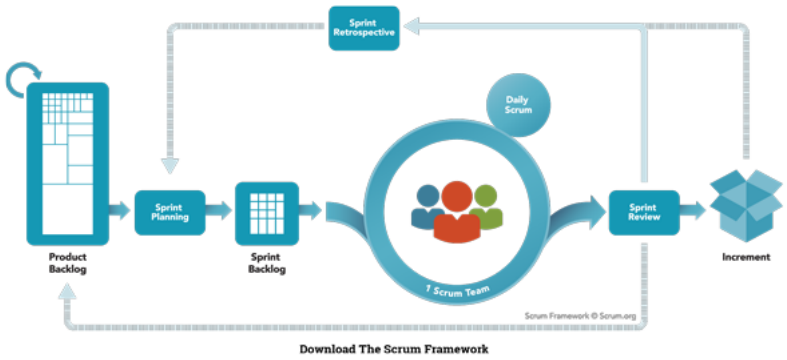


Ilustración 3: Diagrama de flujo de la metodología SCRUM

Fuente: extraída de https://www.scrum.org/resources/what-is-scrum.

Capitulo 4

Planeación

4.1 Participantes

Tabla 1: Participantes del proyecto.

Rol	Nombre
DIRECTOR ACADEMICO DEL PROYECTO	Sandro Javier Bolaños Castro
DIRECTOR EMPRESARIAL DEL PROYECTO	Jorge Ulises Useche Cuellar
SCRUM MASTER	Eliana Marcela Guavita Parrado
PRODUCT OWNER	Mónica Diazgranados Vásquez
ARQUITECTO	Eduardo Salas Buitrago
DESARROLLADORES	Oscar Junior Tacha Gutiérrez (Principal)

Nota. Fuente: Elaboración autor.

4.2 Presupuesto

Tabla 2: Presupuesto del proyecto.

CANTIDAD	UNIDAD	RECURSO	DESCRIPCION	COSTO MENSUAL	COSTO TOTAL
6	Meses	Amazon Web Services (AWS)	AWS API Gateway	$ 60.000	$ 360.000
			AWS API GreenGrass	$ 40.000	$ 240.000
			AWS Lambda	$ 50.000	$ 300.000
			AWS SNS	$ 30.000	$ 180.000
			AWS S3	$ 20.000	$ 120.000
6	Meses	Desarrollador	Encargado de realizar el proyecto Numero de personas una (1)	$ 1.012.000	$ 6.072.000
6	Meses	Arquitecto	Encargado de la infraestructura de todo el proyecto. Número de personas (1).	$ 8.000.000	$ 48.000.000
6	Meses	SCRUM MASTER	Encargado de superar cualquiera dificultad que puedan surger a lo largo del proyecto. Número de personas (1).	$ 6.000.000	$ 36.000.000
6	Meses	Product Owner	Maneja el conocimiento general del producto de identificacion de personas, busca generar estrategias	$ 7.000.000	$ 42.000.000

			seguras de identificacion Número de personas (1).		
6	Meses	Apoyo de alimentacion	Apoyo alimentario brindado por la entidad	$ 20.000	$ 120.000
6	Meses	Conectividad	Apoyo de conectivdad brindado por la entidad	$ 20.000	$ 120.000
6	Meses	Imprevistos	Presupuesto para imprevistos para la pasantia	$ 50.000	$ 300.000
1	Dispositivo	AWS DeepLens	Dispositivo de procesamiento de imágenes	$ 1.750.000	$ 1.750.000
Costo Aproximado del proyecto				$ 24.052.000	$ 135.562.000

Nota. Fuente: Elaboración autor.

4.3 Recursos

Tabla 3: Recursos del proyecto.

Cantidad	Recurso	Especificaciones
1	AWS DeepLens	Procesador: Intel Atom Memoria RAM: 8GB OS: Ubuntu 16-04 LTS Almacenamiento: 16GB Gráficos: Intel Gen9 Graphics Engine
1	Macbook Pro-15”	Procesador: Intel Core I7 2,6 GHz Memoria RAM: 16GB Almacenamiento: 512GB Gráficos: Radeon pro 560x

Nota. Fuente: Elaboración autor.

4.4 Cronograma

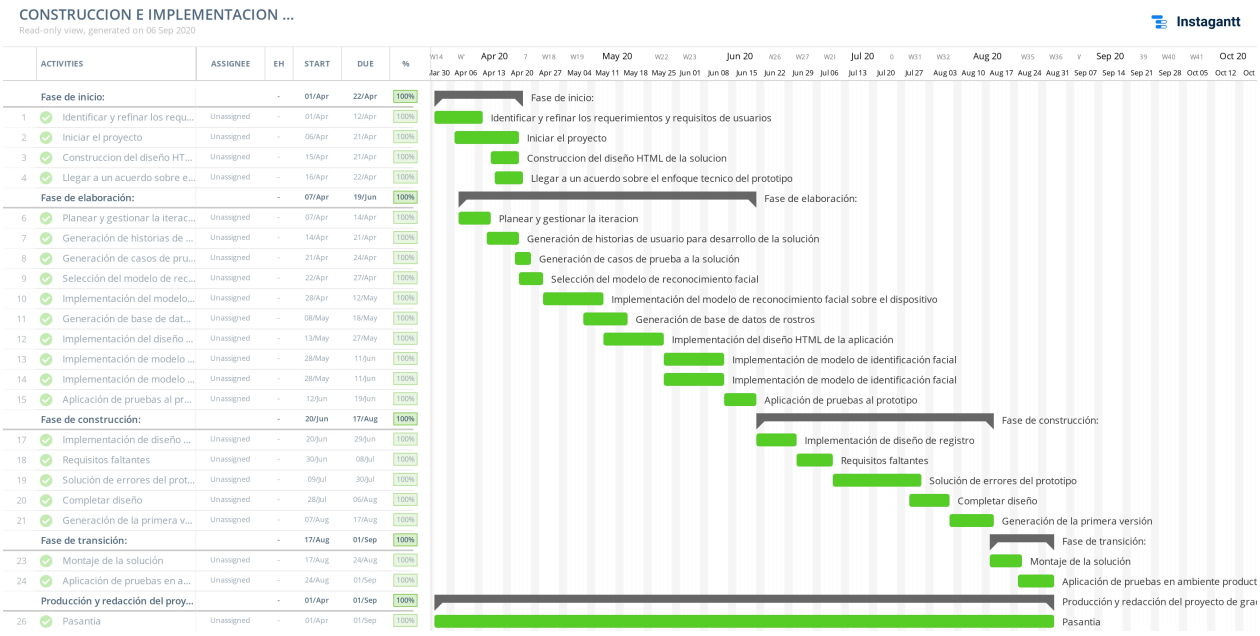


Ilustración 4: Diagrama de Gantt del proyecto.

Fuente Autor.

4.4.1 Fase de inicio

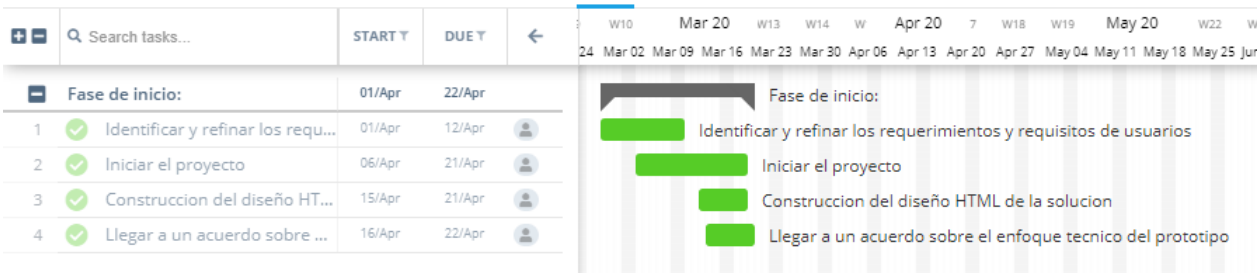


Ilustración 5: Diagrama de Gantt para la fase de inicio del proyecto.

Fuente: Autor.

4.4.2 Fase de elaboración:

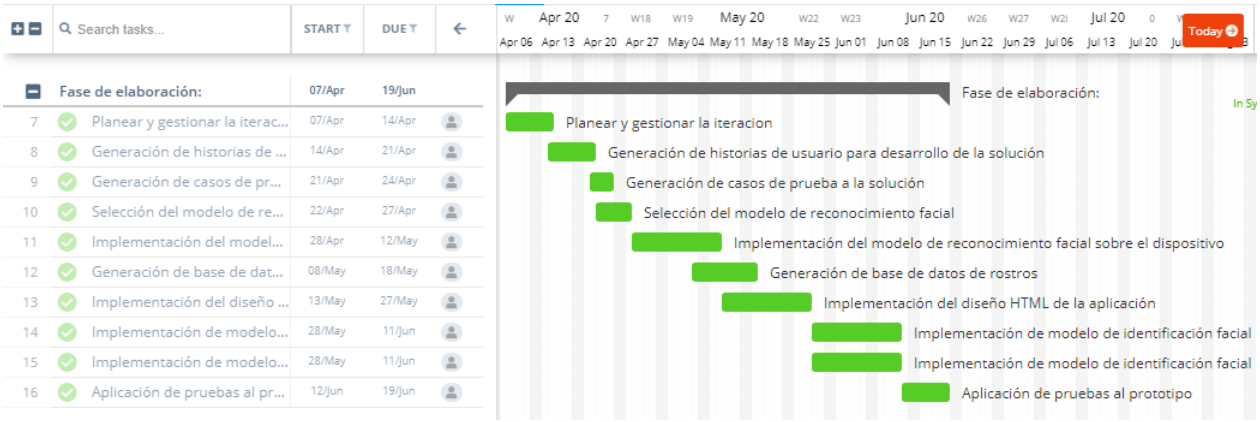


Ilustración 6: Diagrama de Gantt para la fase de elaboración del proyecto.

Fuente: autor.

4.4.3 Fase de Construcción:

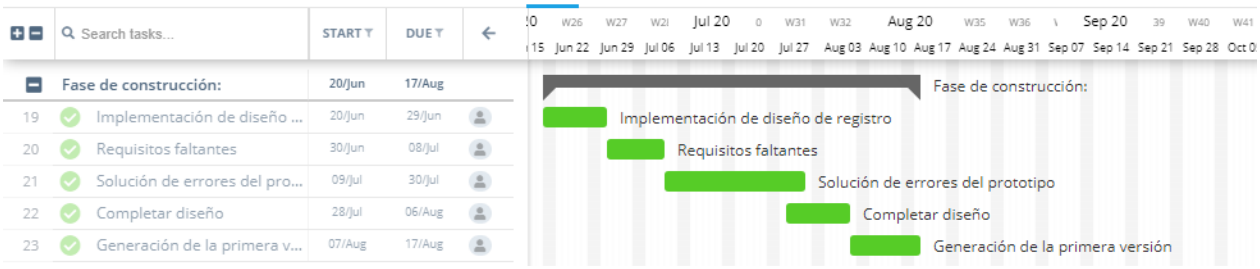


Ilustración 7: Diagrama de Gantt para la fase de construcción del proyecto.

Fuente: Autor.

4.4.4. Fase de transición:

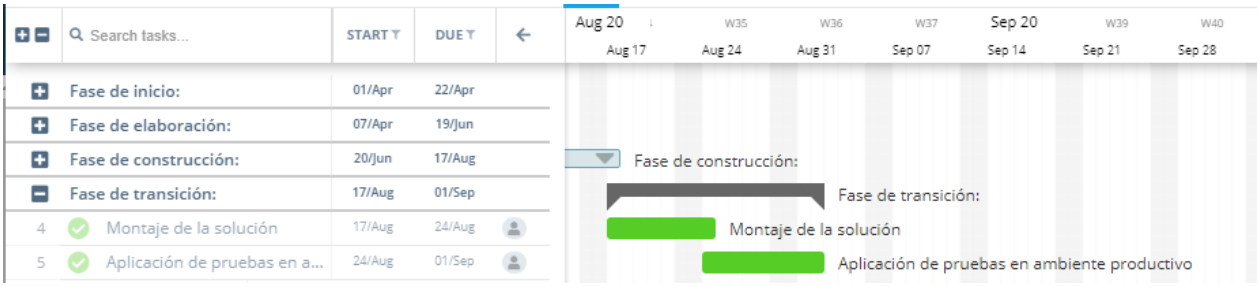


Ilustración 8: Diagrama de Gantt para la fase de transición del proyecto.

Fuente: Autor.

Capitulo 5

Diseño y arquitectura

5.1 Arquitectura

El objetivo en este capitulo es diseñar la arquitectura que contemple la actual problemática hallada en la identificación de personas al interior de las oficinas del Banco de Bogotá. La cual se realiza manualmente donde el usuario digita el numero de identificación.

Para abordar la construcción de la arquitectura debe contemplar la misión de la empresa, la cual busca ser un Banco líder en Colombia para el mercado de empresas, personas y el sector social, oficial. Un Banco siempre a la vanguardia para brindar a sus clientes soluciones anticipadas, que les permitan vivir una experiencia bancaria satisfactoria (Banco de Bogotá, 2021).

El mejor apoyo para el crecimiento y progreso de sus clientes, porque los valora y está dispuesto y disponible para asesorarlos, prestándoles un servicio ágil, oportuno, amable y de calidad. Un Banco que cumple con los objetivos de liderazgo en eficiencia, rentabilidad, utilidad y generación de valor que esperan los accionistas.

Banco de Bogotá es un banco centrado en el usuario y su experiencia generando las mejores soluciones, las cuales deben generar valor al cliente de manera rápida sin dejar de lado la seguridad y la calidad.

5.1.1 Modelo 4+1

Todos hemos visto muchos libros y artículos donde se intenta capturar todos los detalles de la arquitectura de un sistema usando un único diagrama. Pero si miramos cuidadosamente el conjunto de cajas y flechas que muestran estos diagramas, resulta evidente que sus autores han trabajado duramente para intentar representar mas de un plano que lo que realmente podría expresar la notación. ¿Es acaso que las cajas representan programas en ejecución? ¿O representan partes del código fuente? ¿O computadores físicos? ¿O acaso meras agrupaciones de funcionalidad? ¿Las flechas representan dependencias de compilación? ¿O flujo de control? Generalmente es un poco de todo. ¿Sera que una arquitectura requiere un estilo único de arquitectura? A veces la arquitectura del software tiene secuelas de un diseño del sistema que fue muy lejos en particionar prematuramente el software, o de un énfasis excesivo de algunos de los aspectos del desarrollo del software: ingeniería de los datos, o eficiencia en tiempo de ejecución, o estrategias de desarrollo y organización de equipos. A menudo la arquitectura tampoco aborda los intereses de todos sus "clientes". Varios autores han notado este problema, incluyendo a David Garlan y Mary Shaw, Gregory Abowd y Robert Allen, y Paul Clements. El modelo de 4+1 vistas fue desarrollado para remediar este problema. El modelo 4+1 describe la arquitectura del software usando cinco vistas concurrentes (Philippe Kruchten, 1995):

- La vista lógica describe el modelo de objetos del diseño cuando se usa un método de diseño orientado a objetos. Para diseñar una aplicación muy orientada a los datos, se puede usar un enfoque alternativo para desarrollar algún otro tipo de vista lógica, tal como diagramas de entidad-relación.

- La vista de procesos describe los aspectos de concurrencia y sincronización del diseño.
- La vista física describe el mapeo del software en el hardware y refleja los aspectos de distribución.
- La vista de desarrollo describe la organización estática del software en su ambiente de desarrollo.

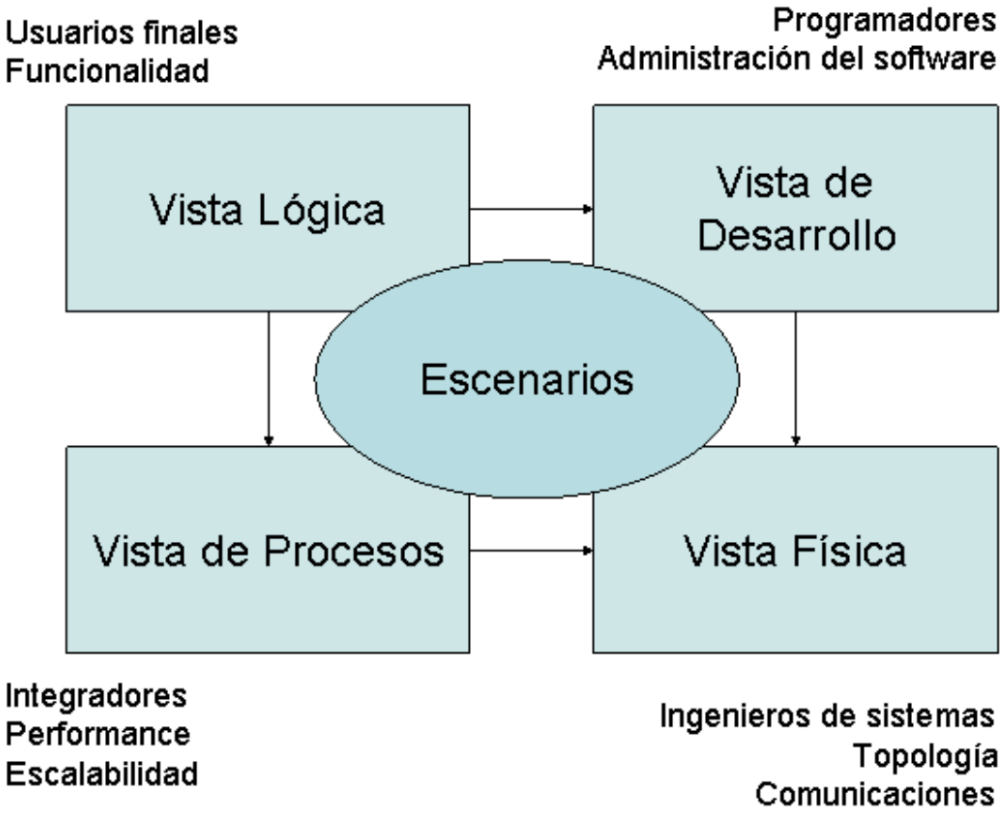


Ilustración 9: Diagrama vista 4+1.

Fuente: Recuperado de Kruchten, P. (1995). Planos Arquitectónicos: El Modelo de "4+1" Vistas de la Arquitectura del Software.

http://cic.puj.edu.co/wiki/lib/exe/fetch.php?media=materias:modelo4_1.pdf.

5.1.2 Vista lógica

5.1.2.1 Diagrama de clases AWS DeepLens

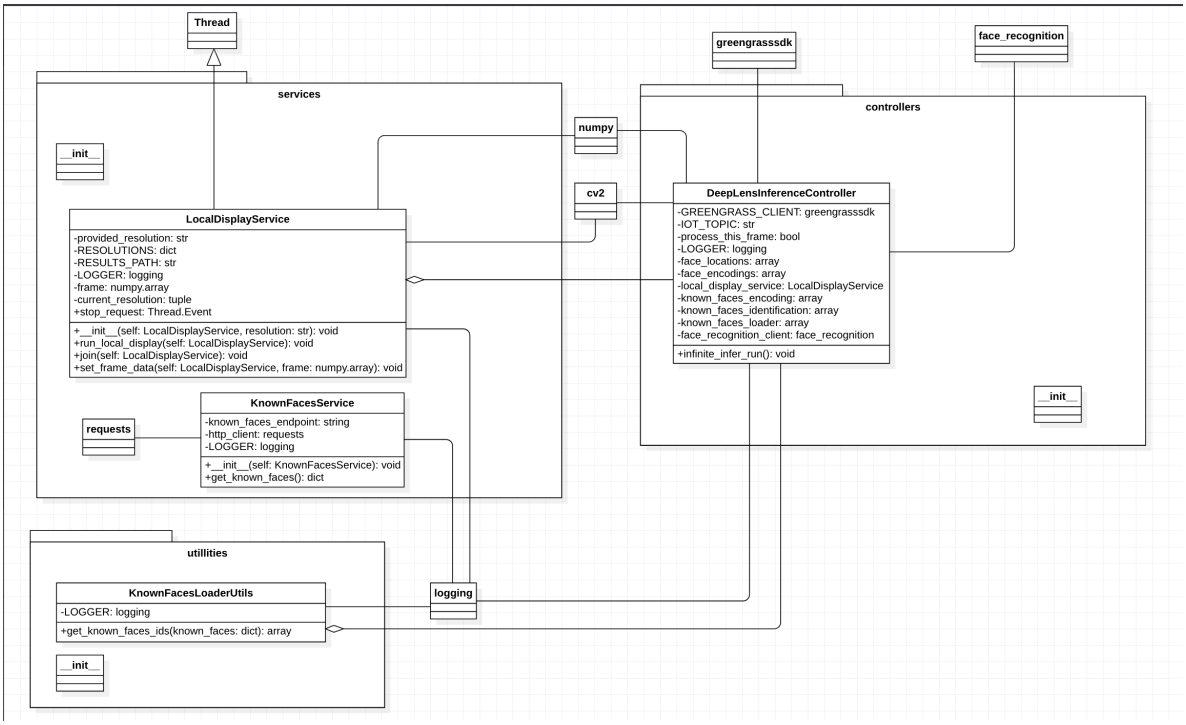


Ilustración 10: Diagrama de clases para el dispositivo AWS DeepLens.

Fuente: Autor.

5.1.2.2 Diagrama de clases suscriptor de eventos

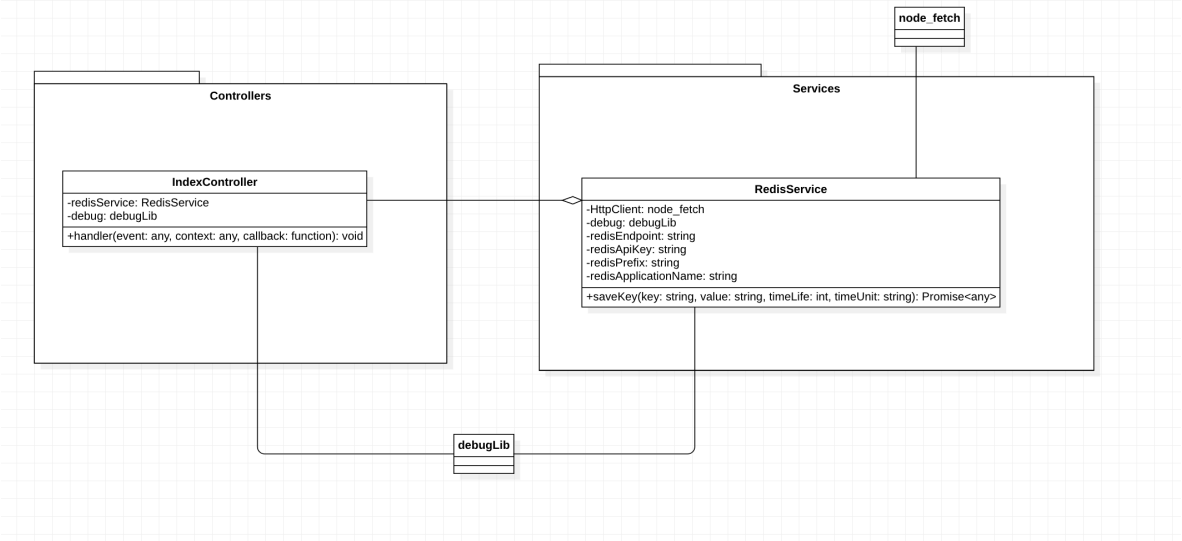


Ilustración 11: Diagrama de clases del componente subscriptor de eventos.

Fuente: Autor.

5.1.2.3 Diagrama de clases publicador de eventos

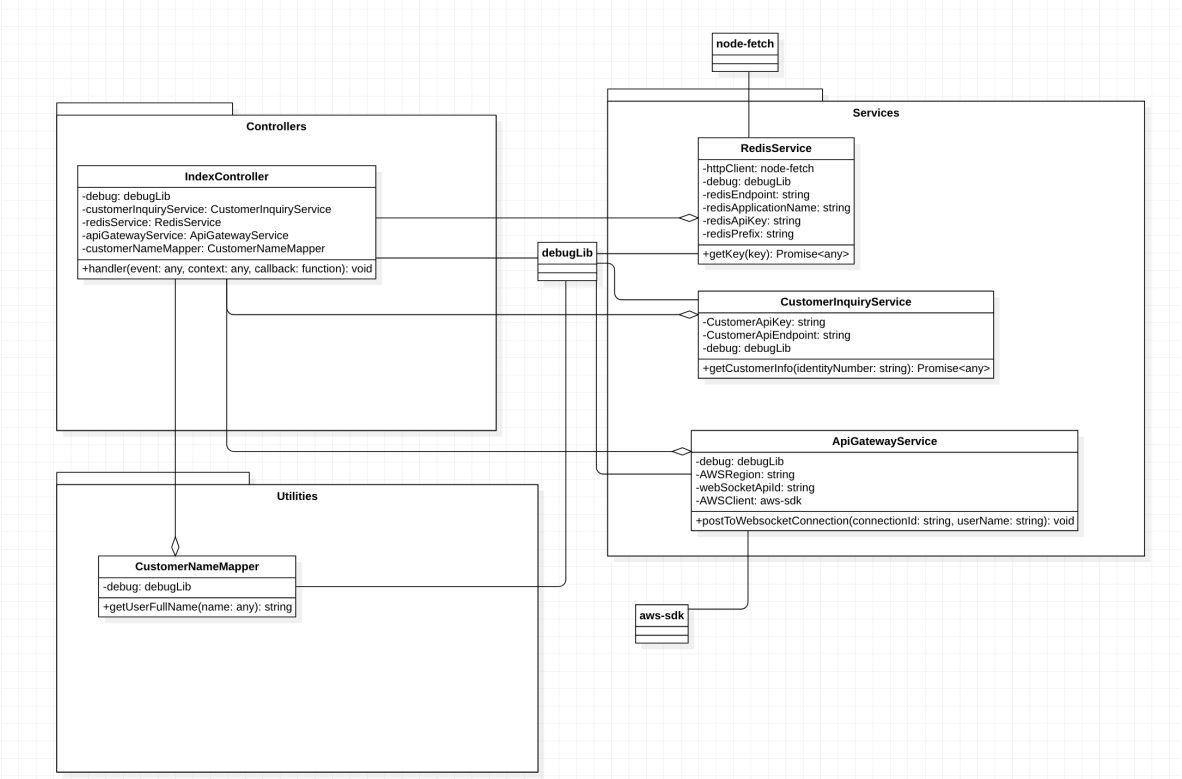


Ilustración 12: Diagrama de clases del componente publicador de eventos.

Fuente: Autor.

5.1.2.3 Diagrama de clases para la interfaz grafica.

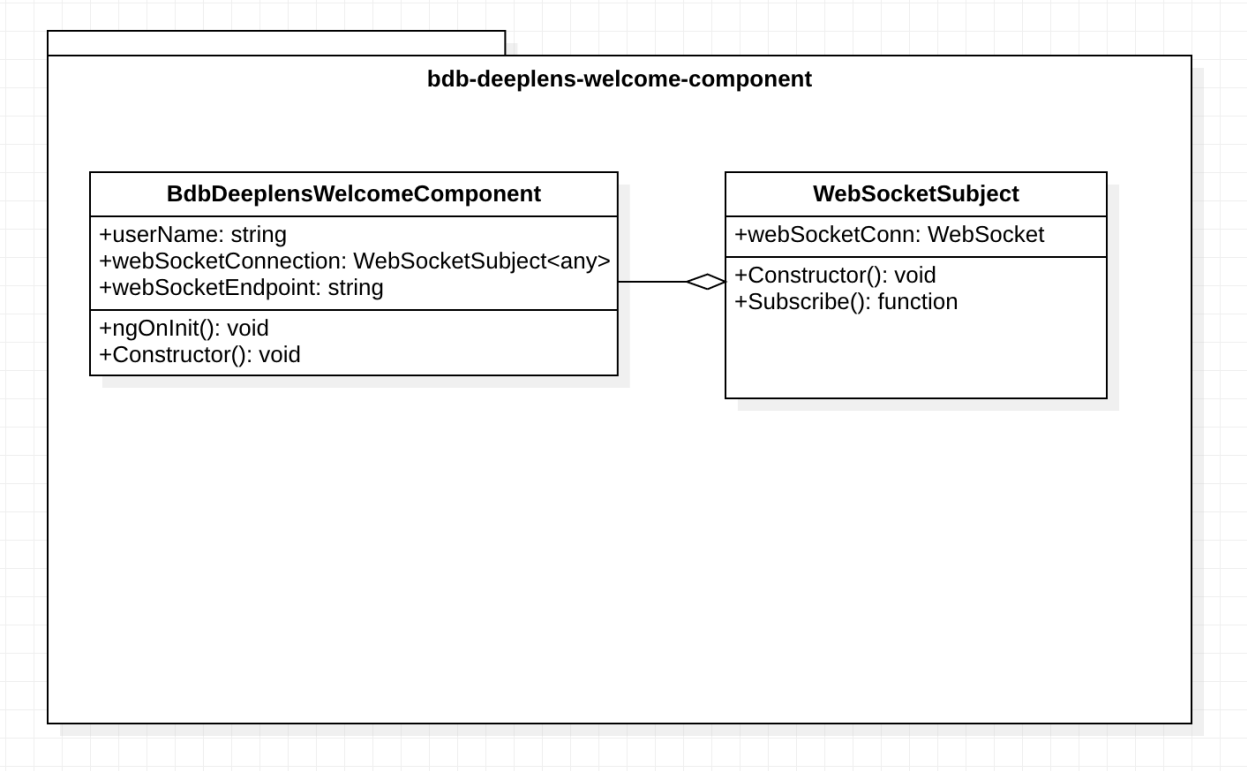


Ilustración 13: Diagrama de clases para la interfaz grafica.

Fuente: Autor.

5.1.3 Vista de desarrollo

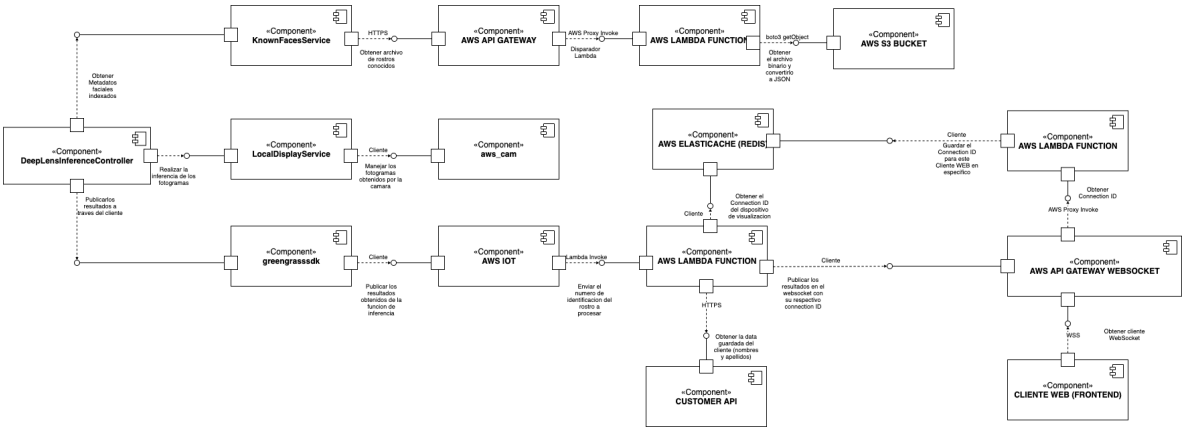


Ilustración 14: Diagrama de componentes del prototipo de identificación de clientes.

Fuente: Autor.

5.1.4 Vista de proceso

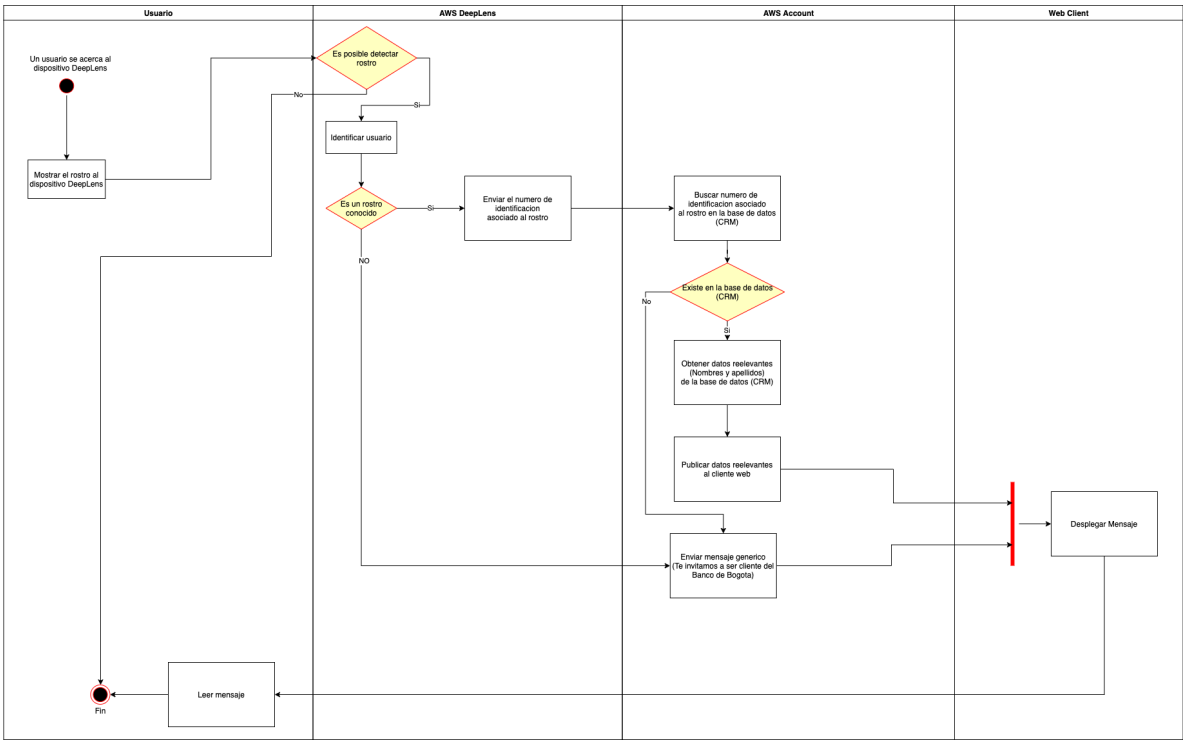


Ilustración 15: Diagrama de flujo o actividades del prototipo de identificación de clientes.

Fuente: Autor.

5.1.5 Vista Física Topología Hardware



Ilustración 16: Vista física topología del hardware del prototipo de identificación de clientes.

Fuente: Autor.

5.1.6 Escenarios

5.1.6.1. Partes de los escenarios propuestos:

5.1.6.1.1 Sistema

Como sistema esta el prototipo diseñado para identificar rostros y así reconocer quien es la persona que esta al frente del Dispositivo DeepLens.

5.1.6.1.2 Actores

- Usuario
- Administrador

5.1.6.1.3 Casos de Uso:

- Identificar al usuario.
- Obtener numero de identificación asociado al rostro.
- Obtener metadatos de rostros conocidos.
- Obtener datos del usuario (nombres y apellidos).
- Buscar en la base de datos el numero de identificación asociado al rostro.
- Mostrar nombres y apellidos del usuario.
- Registrar usuario nuevo.
- Ser cliente activo del Banco de Bogotá.
- Tomar Fotografía.
- Proveer numero de identificación, Nombres y apellidos.
- Actualizar los datos en la base de rostros conocidos.

5.1.6.2 Escenario Identificación facial

Para este caso de uso se identifica al usuario a través de biometría facial siempre y cuando sea posible reconocer un rostro en el dispositivo.

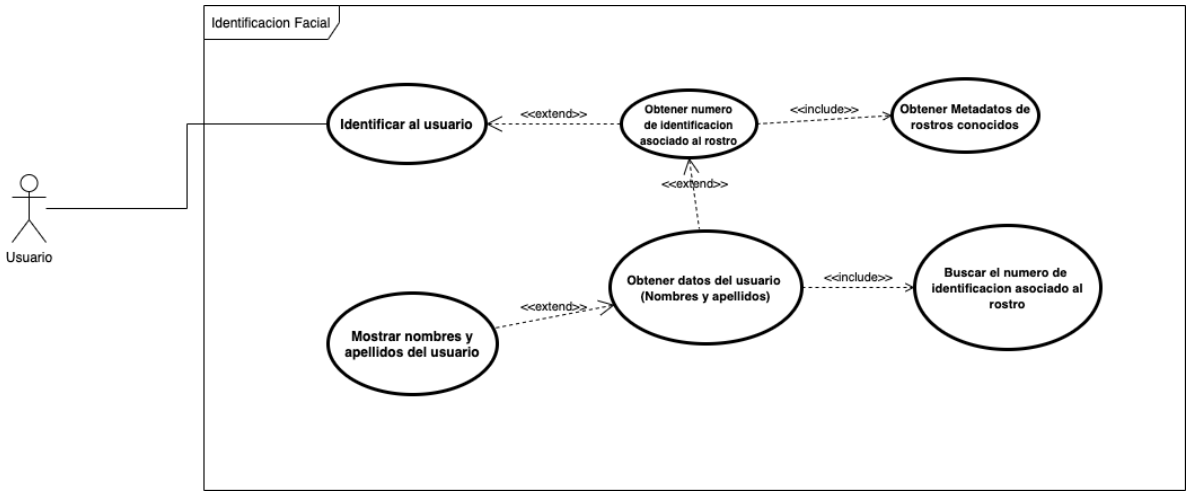


Ilustración 17: Escenario identificación facial de clientes.

Fuente: Autor.

5.1.6.3 Escenario de registro de usuario nuevo.

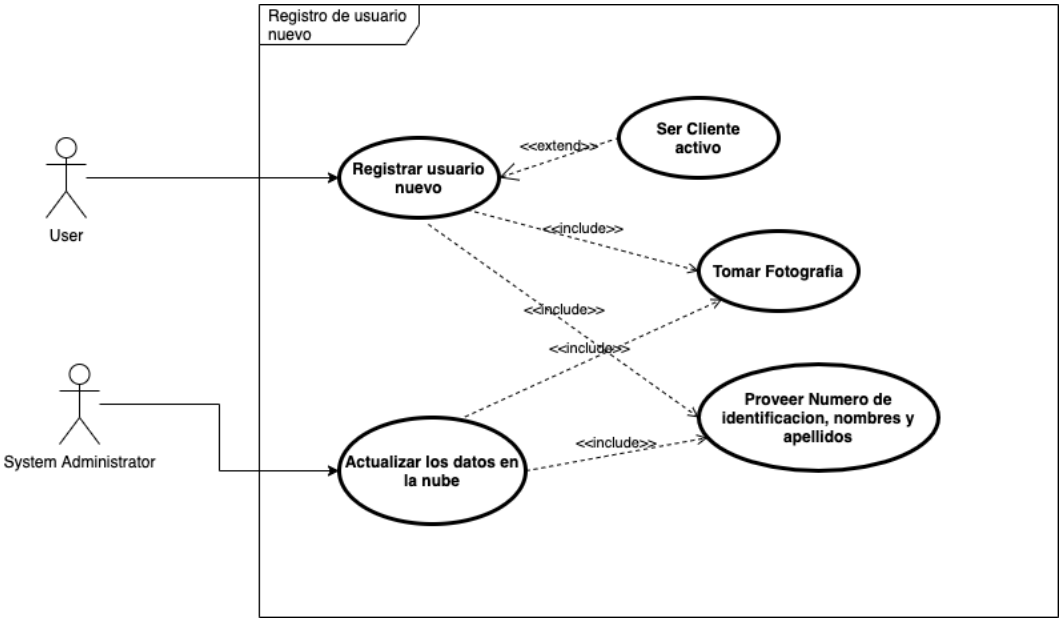


Ilustración 18: Escenario registro de usuario nuevo.

Fuente: Autor.

Capitulo 6

Preparación del proyecto

6.1 Tecnologías a utilizar

6.1.1 JavaScript

Debido al entorno en el cual se desarrollará la solución, un ambiente web JavaScript es un lenguaje idóneo por ser liviano y eficiente además provee librerías para mantener una conexión al servidor. Además, la plataforma a utilizar (Amazon Web Services) tiene compatibilidad con este lenguaje de programación.

6.1.2 HTML

HTML es un lenguaje de marcado que se utiliza para el desarrollo de páginas de Internet lo cual servirá para visualizar los resultados obtenidos después de obtener los datos de la persona previamente clasificada por el dispositivo de procesamiento de imágenes.

6.1.3 Python

Debido a que se va a tratar un volumen imágenes grande y se requiere procesar los mismos rápidamente, se necesita un lenguaje de programación que sea liviano y eficiente, Python cubre esta necesidad además de que ofrece una amplia variedad de librerías para tratamiento de información, clasificación de imágenes además es de código abierto y día a día miles de personas contribuyen a mejorar este lenguaje.

Por otro lado, cuenta con la facilidad de ser compatible con el dispositivo de procesamiento de imágenes (AWS DeepLens).

6.1.4 DLIB

Es una librería escrita en C++ con compatibilidad en Python la cual a través de realizar la carga de un modelo de aprendizaje permite clasificar imágenes con un porcentaje mayor al 80% de confiabilidad (Davis King), además es compatible con el dispositivo de procesamiento de fotogramas.

6.1.5 Typescript

Es un lenguaje de programación que a su ves es un superconjunto de JavaScript por ente se vuelve útil para desarrollo de grandes proyectos en lenguaje JavaScript.

6.1.6 AWS-SDK

El AWS SDK para JavaScript proporciona un conjunto de herramientas para operar sobre el api de cliente de los recursos en Amazon Web Services.(AWS) lo cual permite acceder a atributos y realizar acciones.

6.1.7 Boto3

Boto3 dispone de dos niveles distintos de API. Las API de cliente (o de “bajo nivel”) proporcionan asignaciones individuales a las operaciones HTTP subyacentes de las API. Las API de recursos ocultan llamadas a la red específicas, pero en vez de ello proporcionan recopilaciones y objetos de recursos para acceder a atributos y realizar acciones. (Amazon Web Services, 2021).

6.2 Arquitectura de referencia

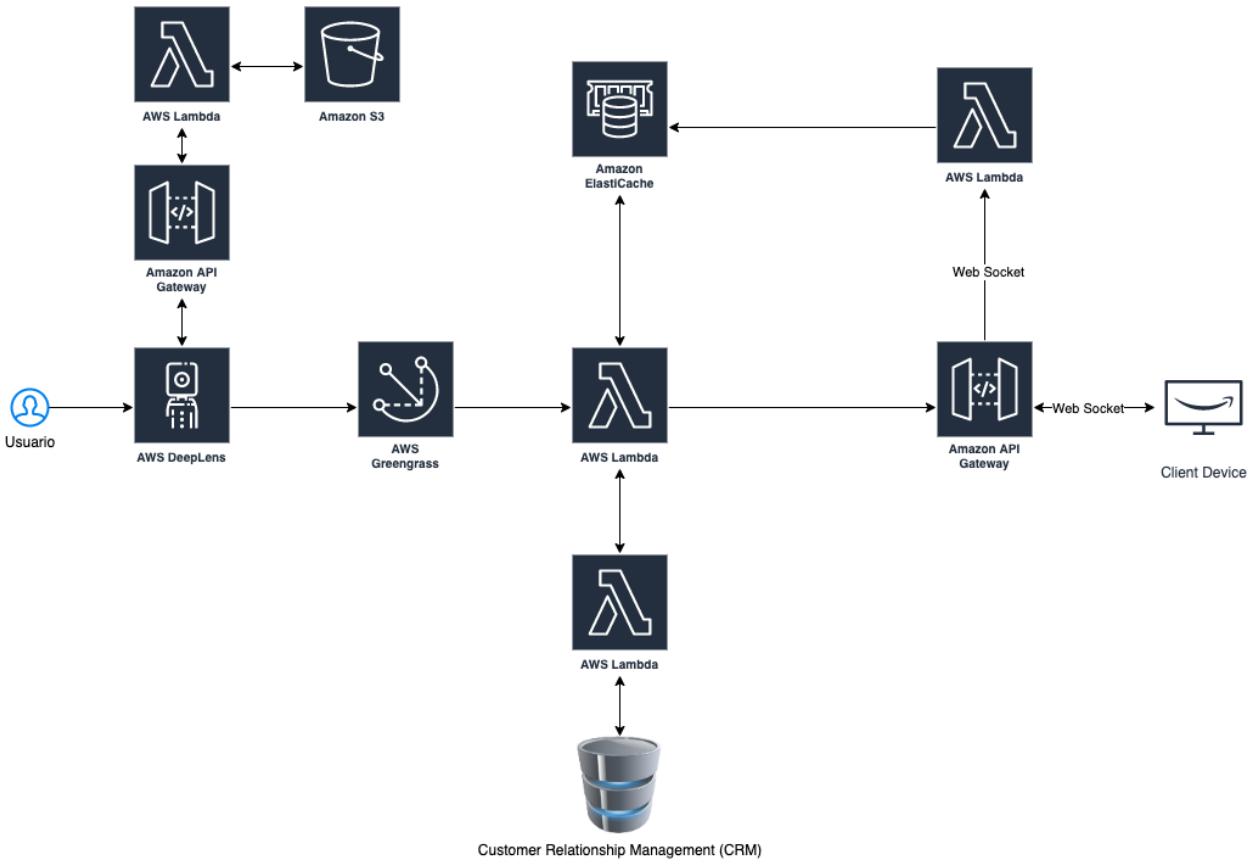


Ilustración 19: Arquitectura de referencia del prototipo.

Fuente: Autor.

En la arquitectura de referencia se ve como se comunican los diferentes componentes en su mayoría se comunican por llamados REST o de uso de librerías a las cuales son los clientes de la nube en este caso Amazon Web Services (AWS)

6.3 Creación del bucket S3

Se necesita configurar un bucket S3 de almacenamiento para poder las personas conocidas por numero de identificación junto a sus metadatos faciales en formato JSON.

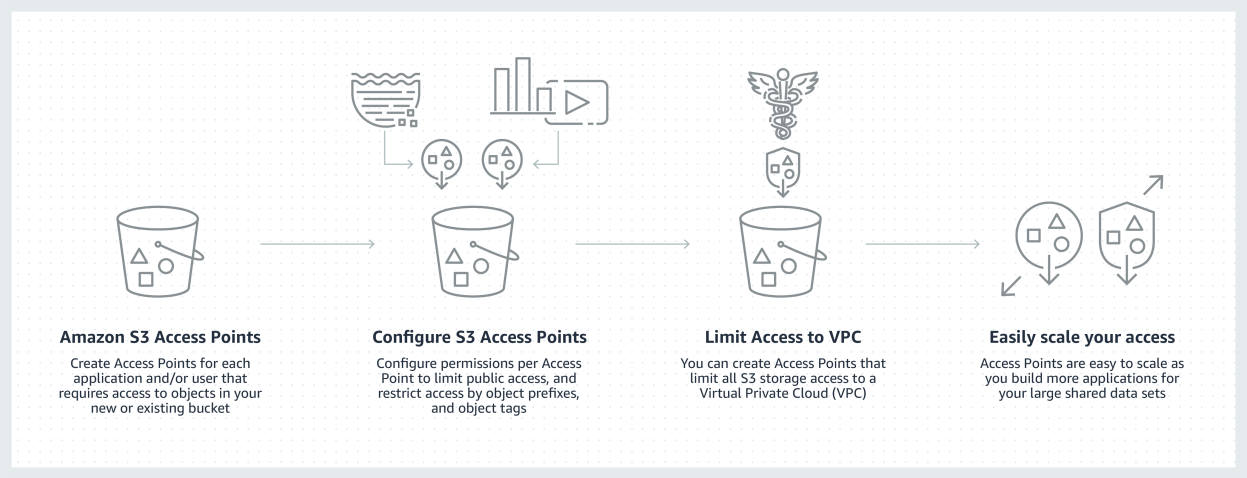


Ilustración 20: Diagrama de Amazon Web Services (AWS) S3.

Fuente: Recuperado de https://aws.amazon.com/es/s3/.

6.4 Diseño de la Interfaz grafica

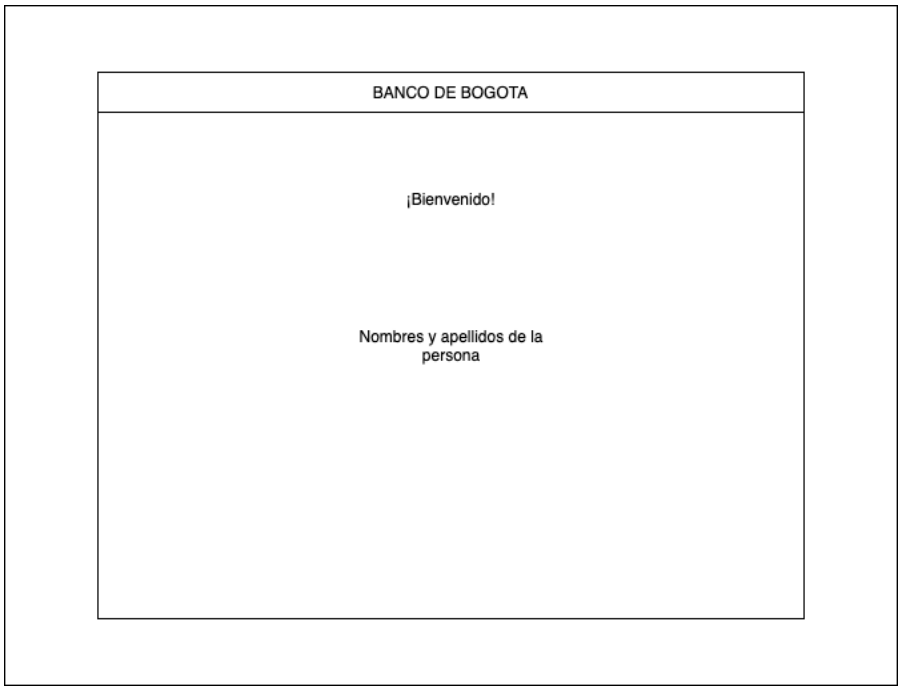


Ilustración 21: Diseño de interfaz gráfica.

Fuente: Autor.

El diseño de la interfaz grafica es bastante simple y sencilla consta de una barra de navegación con el logo del Banco de Bogotá, un mensaje de bienvenida y por ultimo el resultado arrojado por el dispositivo DeepLens cabe aclarar que el diseño de la interfaz es responsive es decir que se adapta a los diferentes tipos de dispositivos, y tamaños de pantalla.

6.5 Diagrama de secuencia para la visualización de resultados del prototipo de identificación de clientes

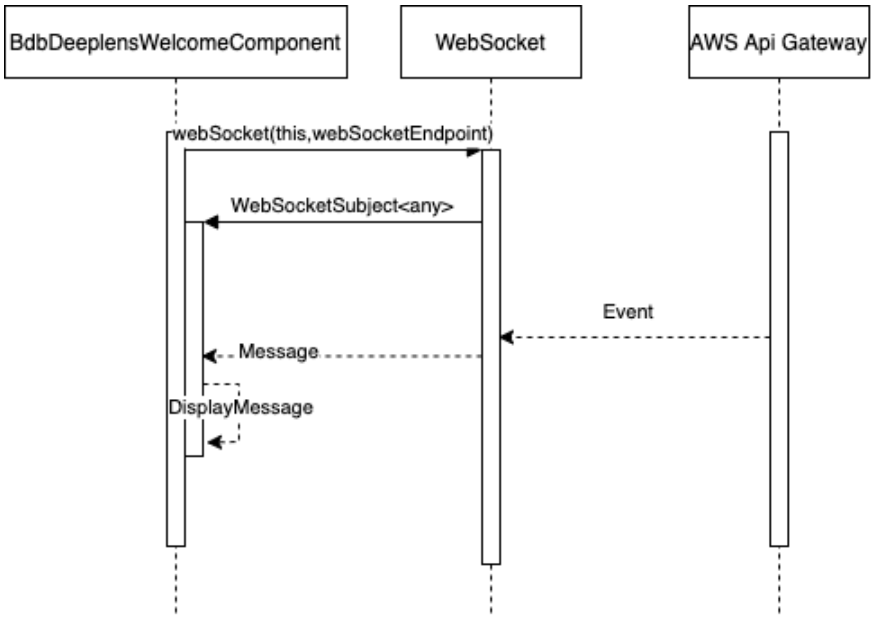


Ilustración 22: Diagrama de secuencia de la interfaz grafica.

Fuente: Autor.

Se ve la secuencia de los componentes, para lo cual el API Gateway de Amazon Web Services (AWS) esta publicando mensajes con el resultado que es obtenido desde el dispositivo DeepLens lo cual permite hacer múltiples implementaciones del mismo proceso lo cual es lo que se quiere para implementar en diversos sitios del Banco de Bogotá.

Capitulo 7

Desarrollo del proyecto

7.1 Obtención y guardado de metadatos faciales

Para hacer la obtención de metadatos se hace uso de la librería de Python face recognition la cual a su ves hace uso de las librerías DLIB y numpy lo cual retorna un arreglo con un tamaño de 128 en los cuales estará el rostro codificado. Este arreglo es guardado con su respectivo numero de identificación para este caso en especifico se usa el documento de identidad de la persona en un archivo JSON y este a su ves es almacenado en un bucket S3 de Amazon Web Services.

7.2 Obtención de metadatos de rostros conocidos e indexados guardados en Bucket S3

Para hacer la obtención de metadatos de los rostros conocidos con su respectiva identificación se expone el archivo JSON que esta almacenado en un Bucket S3 de Amazon Web Services (AWS) a través de una función AWS Lambda la cual es disparada a su ves por un evento que llega a AWS API Gateway, el llamado a API Gateway se hace con una llave (API-KEY) la cual asegura los llamados de las API y así evitar un ataque de denegación de servicios.

7.3 Configuración del dispositivo de procesamiento de imágenes (AWS DeepLens).

7.3.1 Identificación de la red SSID y contraseña del dispositivo AWS DeepLens

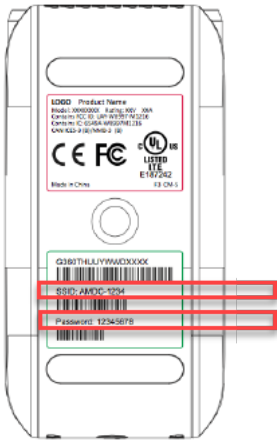


Ilustración 23: Identificación de la red SSID y contraseña del dispositivo AWS DeepLens.

Fuente: Recuperado de https://docs.aws.amazon.com/deeplens/latest/dg/deeplens-getting-started-connect.html.

Para empezar la configuración del dispositivo AWS DeepLens se debe identificar dos variables el SSID o nombre de la red WI-FI generada y la contraseña de la red contraseña para poder conectarse en un paso posterior.

7.3.2 Como conectar un equipo de escritorio (Computador) al WI-FI generado por el dispositivo AWS DeepLens

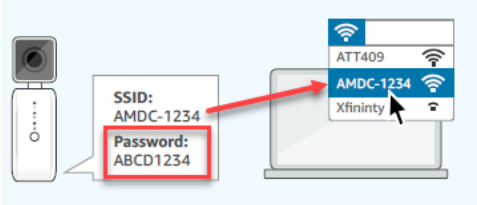


Ilustración 24: Conexión entre el ordenador y el dispositivo DeepLens vía WI-FI.

Fuente: Recuperado de https://docs.aws.amazon.com/deeplens/latest/dg/deeplens-getting-started-connect.html.

Una ves identificadas las credenciales de acceso WI-FI desde un ordenador se accede a la red WI-FI generada por el dispositivo AWS DeepLens.

7.3.3 Configurar el dispositivo AWS DeepLens para conexión WI-FI

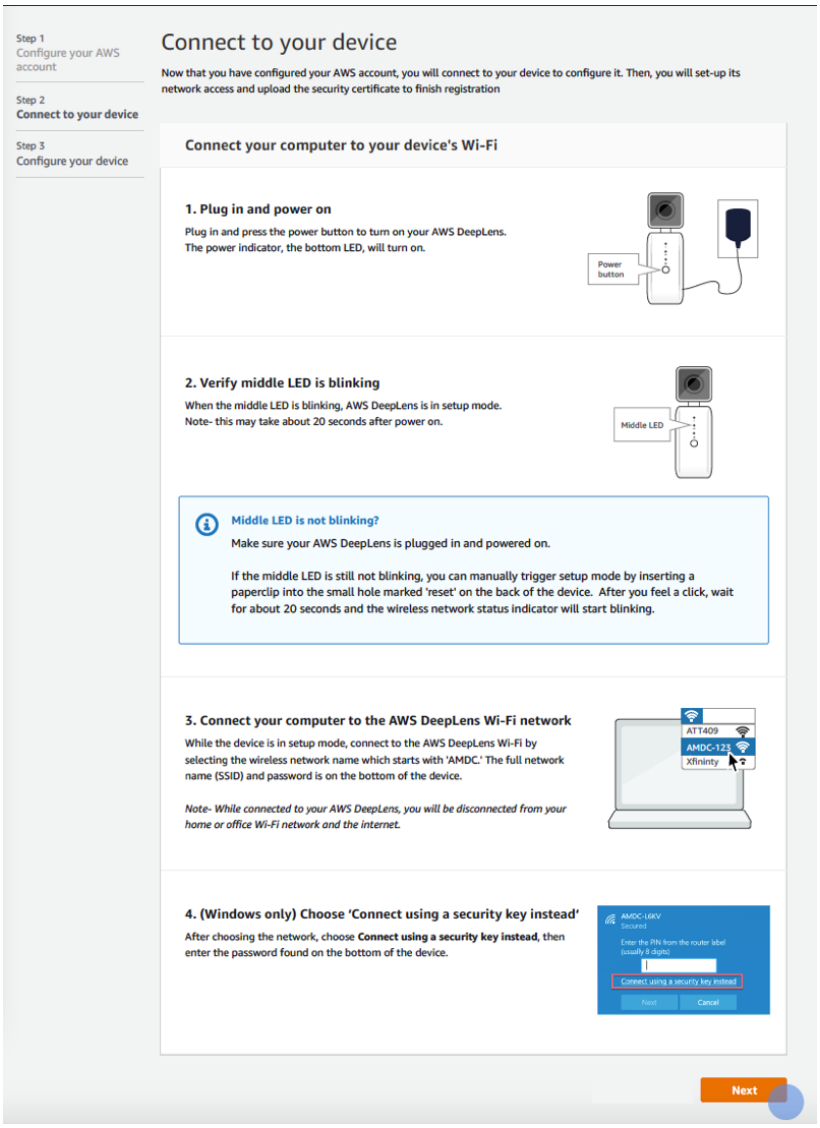


Ilustración 25: Como Configurar el dispositivo DeepLens para conexión WI-FI.

Fuente: Recuperado de https://docs.aws.amazon.com/deeplens/latest/dg/deeplens-getting-started-connect.html.

En la ilustración 25 se nota el paso a paso de como configurar el dispositivo AWS DeepLens para conexión vía WI-FI.

7.3.4 Configuración del dispositivo AWS DeepLens para asociarlo con una cuenta de Amazon Web Services.

7.3.4.1 Conectar el dispositivo AWS DeepLens a internet.

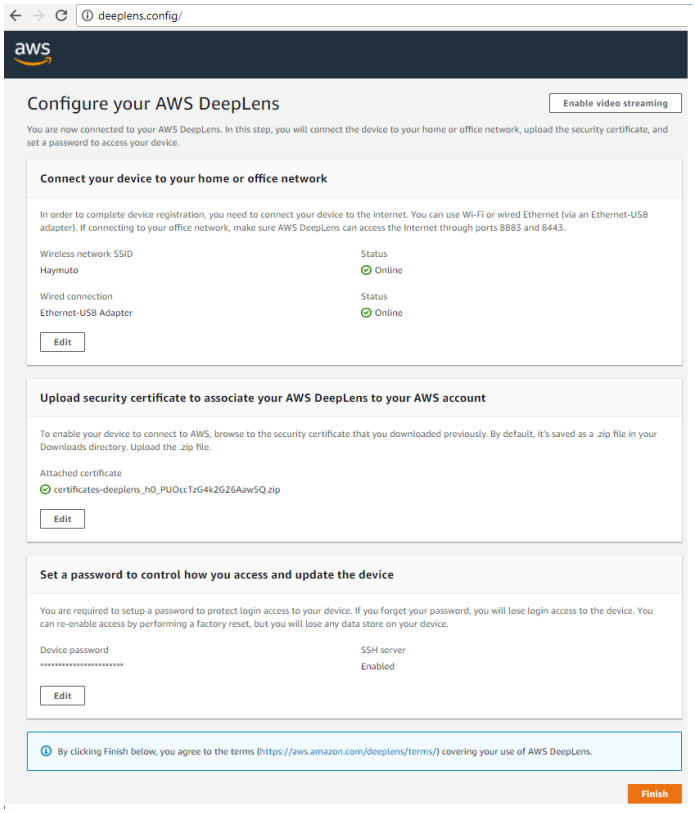


Ilustración 26: Interfaz de configuración inicial dispositivo AWS DeepLens

Fuente: Recuperado de https://docs.aws.amazon.com/es_es/deeplens/latest/dg/deeplens-getting-started-set-up.html.

Una ves conectado a el WI-FI generado por el dispositivo AWS DeepLens. En el computador a través de un navegador se debe ir a la dirección “http://deeplens.config” la cual provee una interfaz como se ve en la ilustración 26, en esta interfaz, se configura la red WI-FI local, se activa el acceso SSH y se asigna una contraseña para este acceso y se da clic en finalizar.

7.3.4.2 Configuración del dispositivo DeepLens en una cuenta de Amazon Web Services.

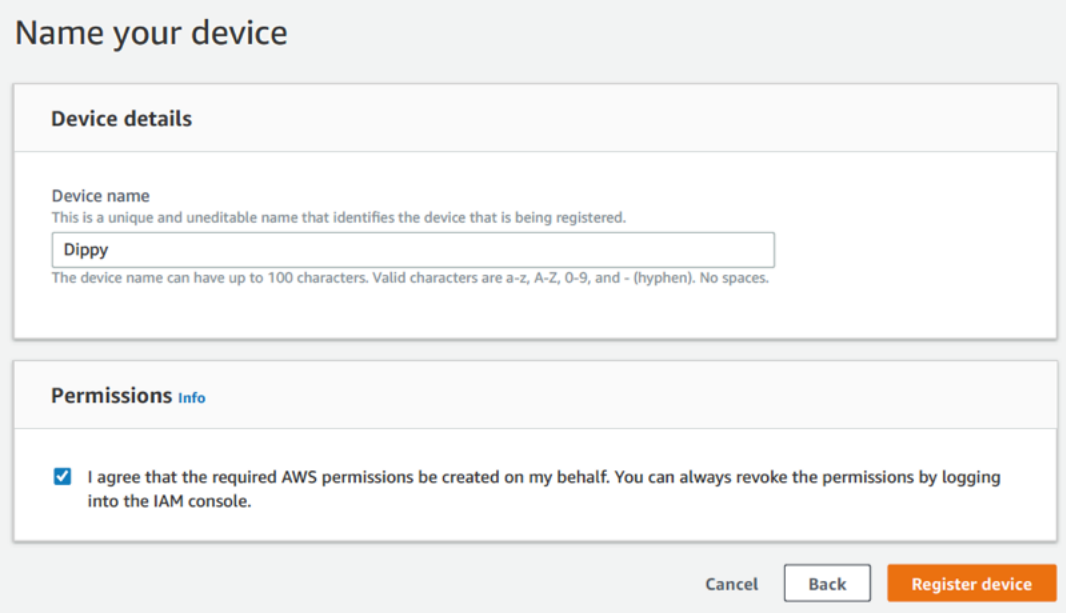


Ilustración 27: Interfaz de Configuración del dispositivo DeepLens en una cuenta de Amazon Web Services.

Fuente: Recuperado de https://docs.aws.amazon.com/es_es/deeplens/latest/dg/how-to-configure-deeplens-v1.1-device.html.

Una ves el dispositivo tenga conexión a internet se navega en la consola de AWS: y se busca DeepLens allí se da clic en registrar el dispositivo se asigna el nombre y los permisos dentro de la cuenta a través del servicio de AWS IAM (Identity and Access Management) como se ve en la ilustración 27.

7.4.3.3 Verificación de la configuración del dispositivo DeepLens

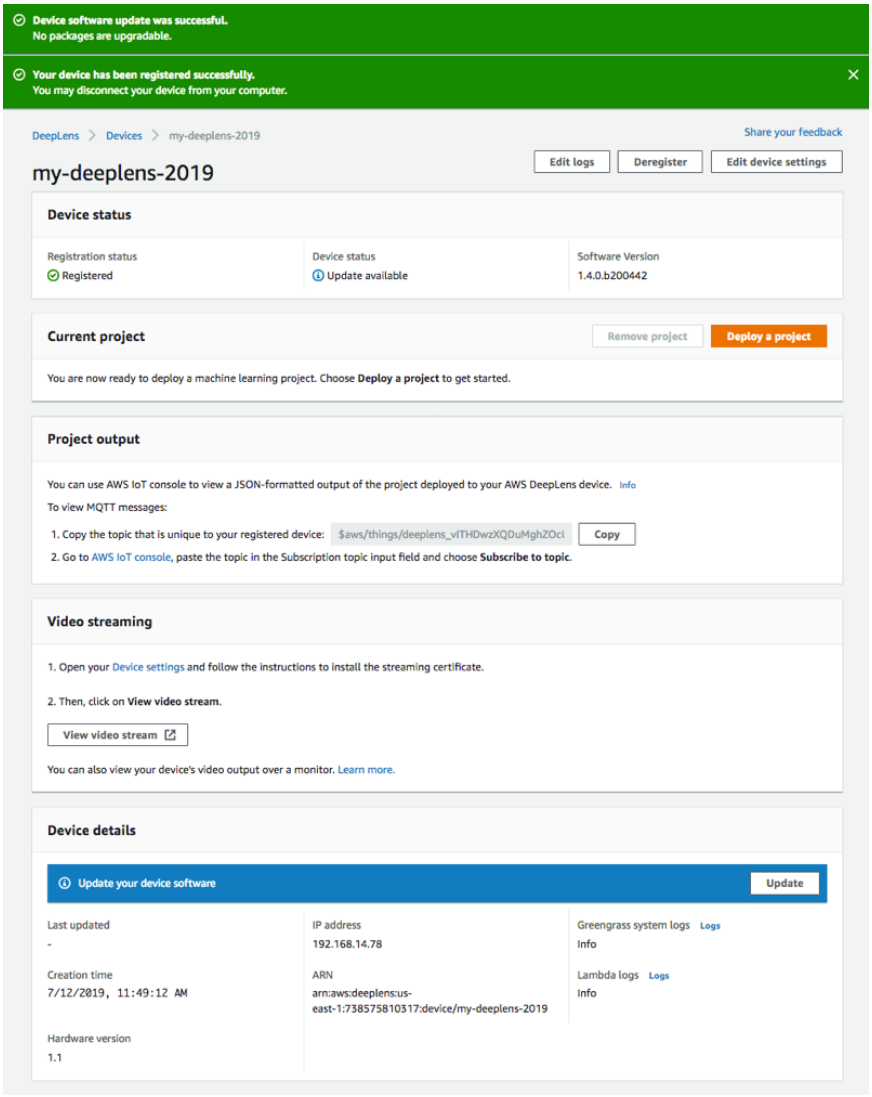


Ilustración 28: Interfaz de Verificación del estado del dispositivo DeepLens en una cuenta de Amazon Web Services.

Fuente: Recuperado de https://docs.aws.amazon.com/es_es/deeplens/latest/dg/how-to-configure-deeplens-v1.1-device.html.

Una ves finalizada la configuración se procede a revisar en la interfaz de la consola de AWS sea “Registrado” en el estado del dispositivo como se ve en la ilustración 28.

7.4 Creación del proyecto para el dispositivo AWS DeepLens

7.4.1 Instalación de dependencias en el dispositivo AWS DeepLens

Una ves se termina el registro y configuración del dispositivo, con el AWS DeepLens encendido y conectado a WI-FI se debe hacer una conexión SSH al dispositivo para poder manipular el sistema operativo desde la consola del ordenador también es posible hacer lo con mouse y teclado a través de un puerto nano-HDMI. para este caso se realiza la instalación de dependencias vía SSH se requiere actualizar las dependencias a la ultima versión estable e instalar la librería face-recognition.

7.4.2 Creación de la función AWS Lambda de inferencia.

7.4.2.1 Generación del código para la función de inferencia

Se debe obtener el modelo que provee AWS para funcionamiento del dispositivo DeepLens se encuentra en Amazon Web Services.

La Plantilla se puede encontrar en “https://docs.aws.amazon.com/deeplens/latest/dg/deeplens-inference-lambda-create.html”.

Con el modelo se procede a abrir el paquete en un editor de código y realizar los respectivos ajustes y desarrollos asegurándose de que se cumplen las métricas de calidad de código realizando la respectiva integración continua.

Por ultimo se debe comprimir en un Zip para ser subido a AWS posterior mente.

7.4.2.2 Creación de una función AWS lambda

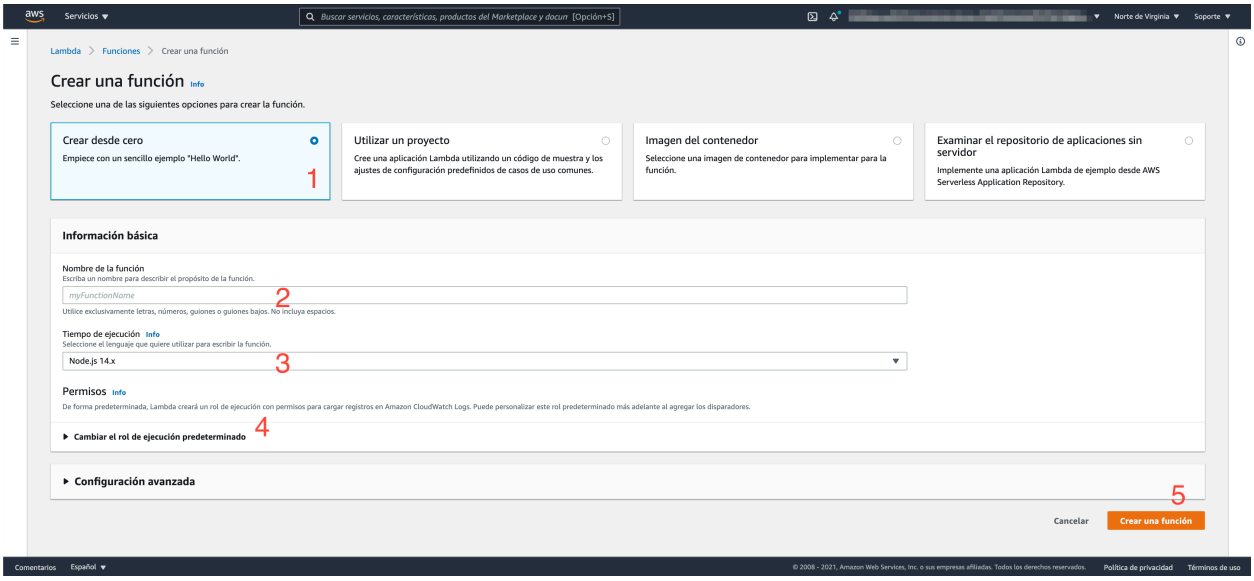


Ilustración 29: Interfaz de creación de función lambda en la consola de administración de cuenta de Amazon Web Services.

Fuente: Autor.

En la consola de administración se busca Lambda y se selecciona tal cual como se ve en la ilustración 29 una ves aquí se selecciona Crear desde cero (1) luego se le debe asignar un nombre para este caso bbog-sg-deeplens-face-recognition (2) en tiempo de ejecución se selección Python 3.8 (3) en permisos se selecciona el rol por defecto provisto por AWS (4) service-role/AWSDeepLensLambdaRole y por último crear función (5).

7.4.2.3 Subir el proyecto de implementación anteriormente generado

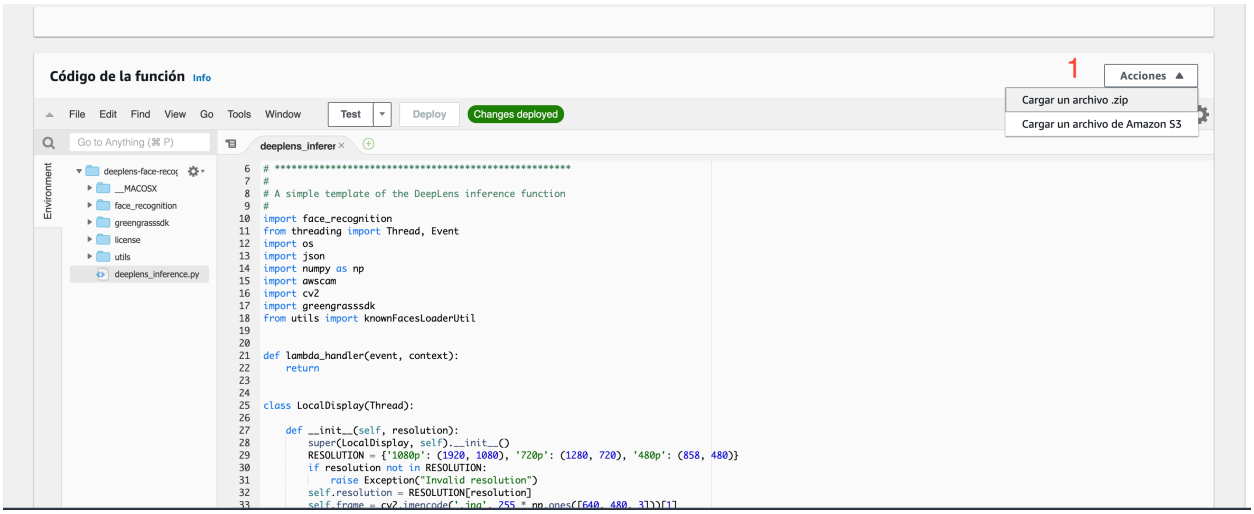


Ilustración 30: Interfaz de como subir un proyecto de código comprimido a una función AWS Lambda en consola de administración AWS.

Fuente: Autor.

Se debe importar el código generado anteriormente (7.4.2.1) en la interfaz de la función creada se debe añadir el código comprimido en Zip como se ve en la ilustración 30 (1).

7.4.2.4 Generación una nueva versión del proyecto de implementación

Se debe generar una nueva versión de la función AWS Lambda para posteriormente ser desplegada en el dispositivo AWS DeepLens.

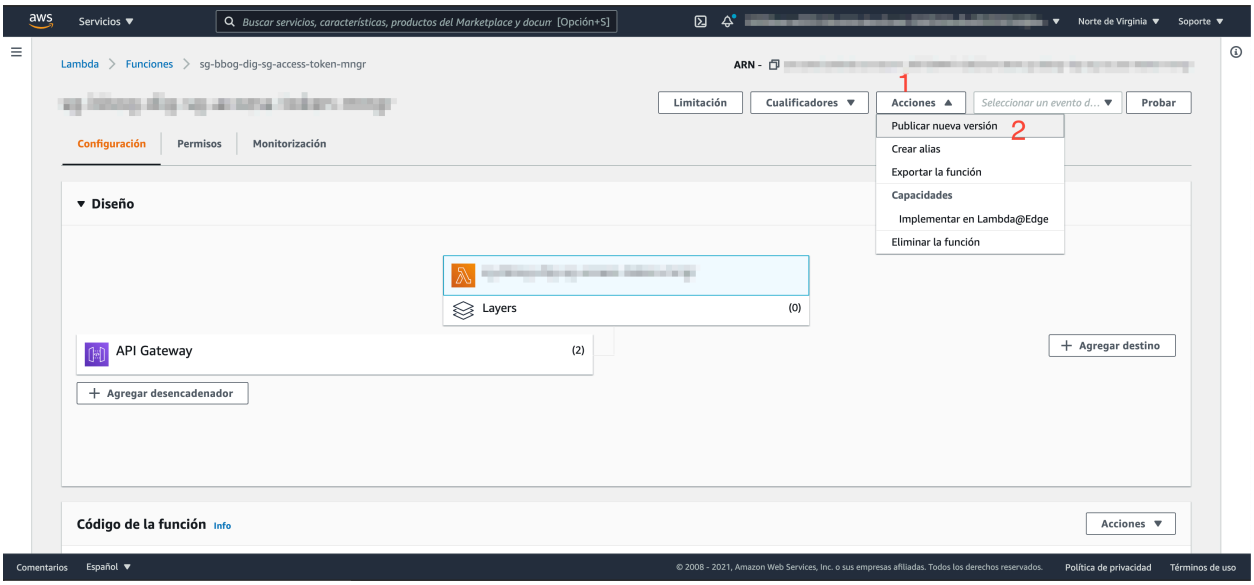


Ilustración 31: Interfaz de como generar una nueva versión de una función AWS Lambda en consola de administración AWS.

Fuente: Autor.

Para generar una nueva versión de la funcion Lambda se debe ir a acciones (1) luego se selecciona publicar nueva versión (2).

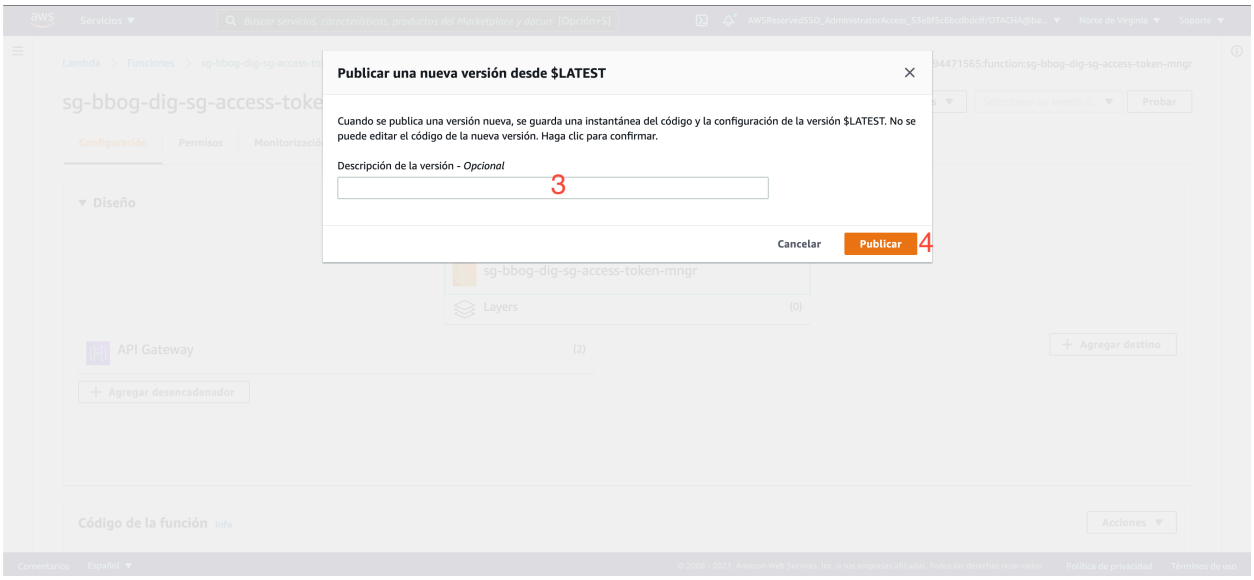


Ilustración 32: Interfaz de como generar una nueva versión de una función AWS Lambda en consola de administración AWS.

Fuente: Autor.

En esta ilustración se le asigna una descripción de la versión (3) la cual es opcional y por último publicar.

7.5 Implementación y despliegue del proyecto en el dispositivo AWS DeepLens

7.5.1 Creación del proyecto de implementación para desplegar en el dispositivo AWS DeepLens

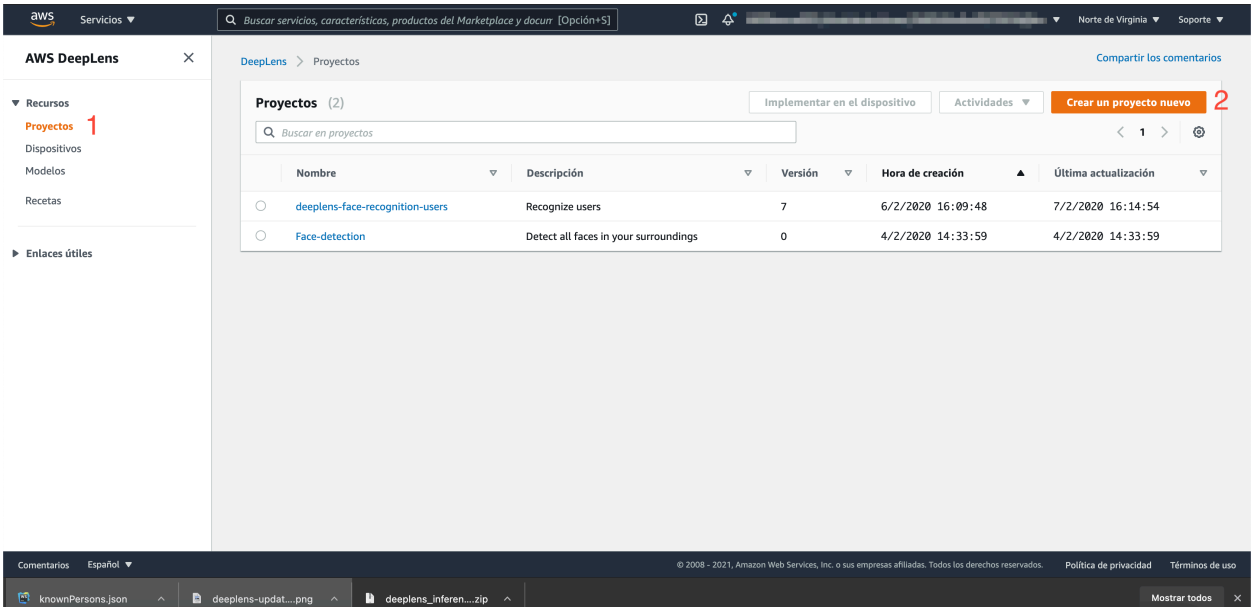


Ilustración 33: Creación del proyecto de implementación para desplegar en el dispositivo DeepLens.

Fuente: Autor.

En la consola de administración de la cuenta de Amazon Web Services (AWS) se busca DeepLens una ves allí se selecciona Proyectos (1), luego se selecciona Crear un proyecto (2).

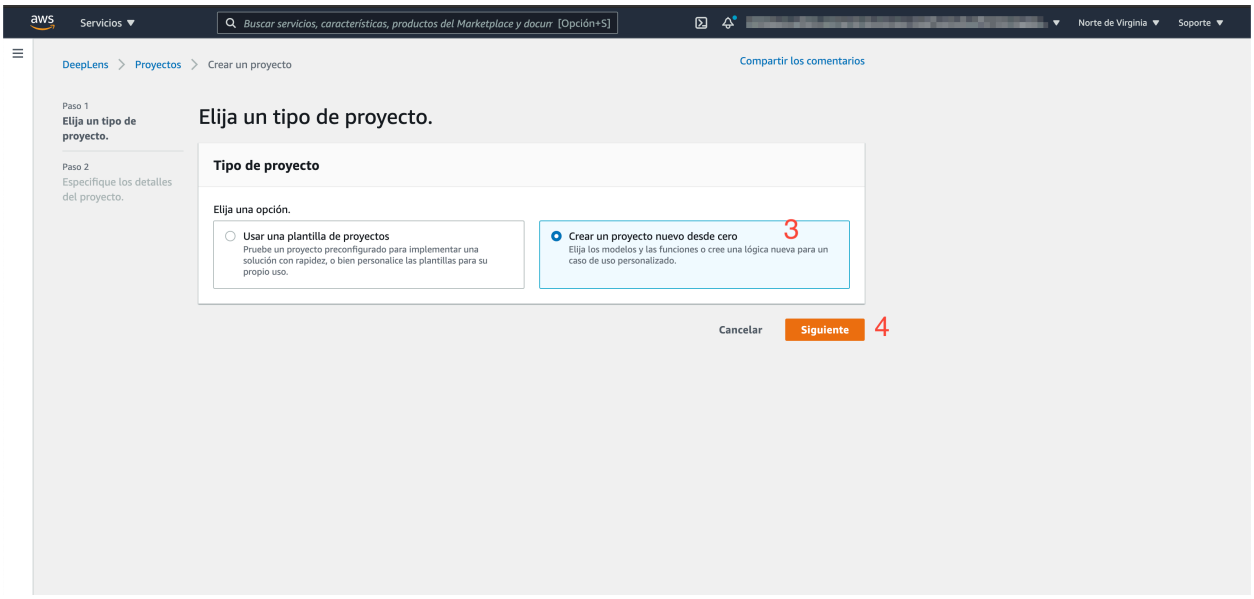


Ilustración 34: Elección tipo de proyecto de implementación para AWS DeepLens.

Fuente Autor.

En la selección de tipo de proyecto se selecciona “Crear un proyecto desde cero” (3) y se da clic en siguiente.

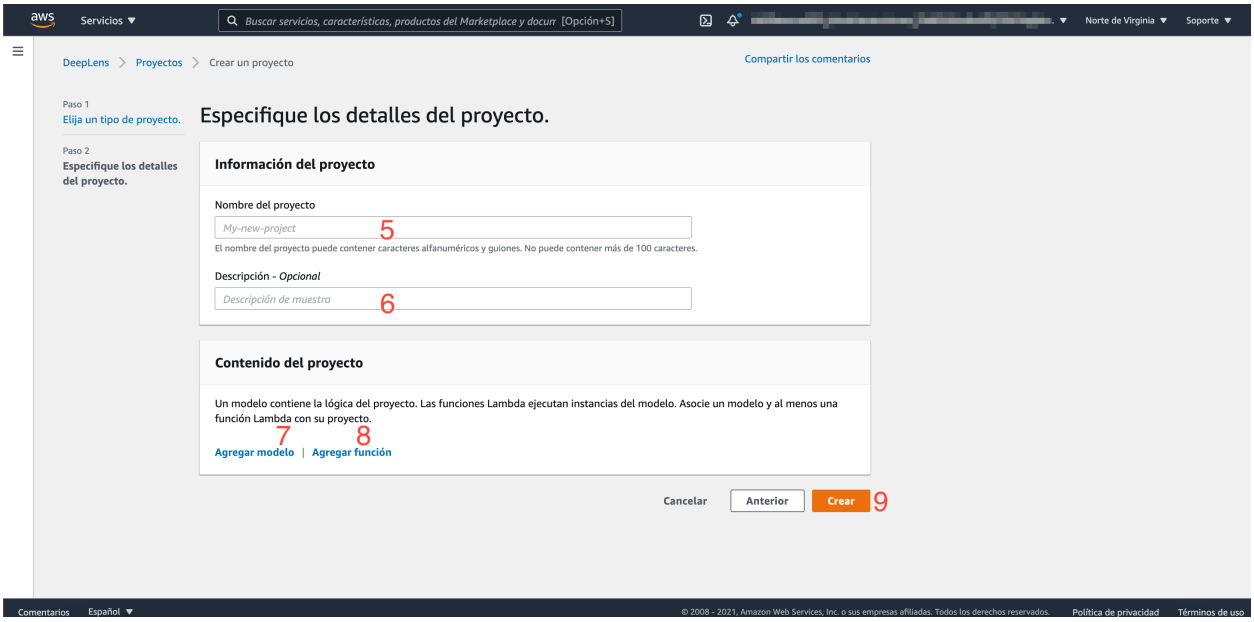


Ilustración 35: Especificación de detalles del proyecto de implementación en el dispositivo AWS DeepLens.

Fuente: Autor.

En esta interfaz se asigna un nombre al proyecto (5) para este caso el nombre asignado es “bbog-sg-deeplens-face-recognition”, la descripción del proyecto es opcional (6), en contenido del proyecto se agrega el modelo face recognition de ResNET (7) y en función se selecciona la función AWS Lambda generada anteriormente en el numeral 6.4.2 (8) y por ultimo se da clic en crear (9).

7.5.2 Despliegue en el dispositivo AWS DeepLens del proyecto anteriormente creado

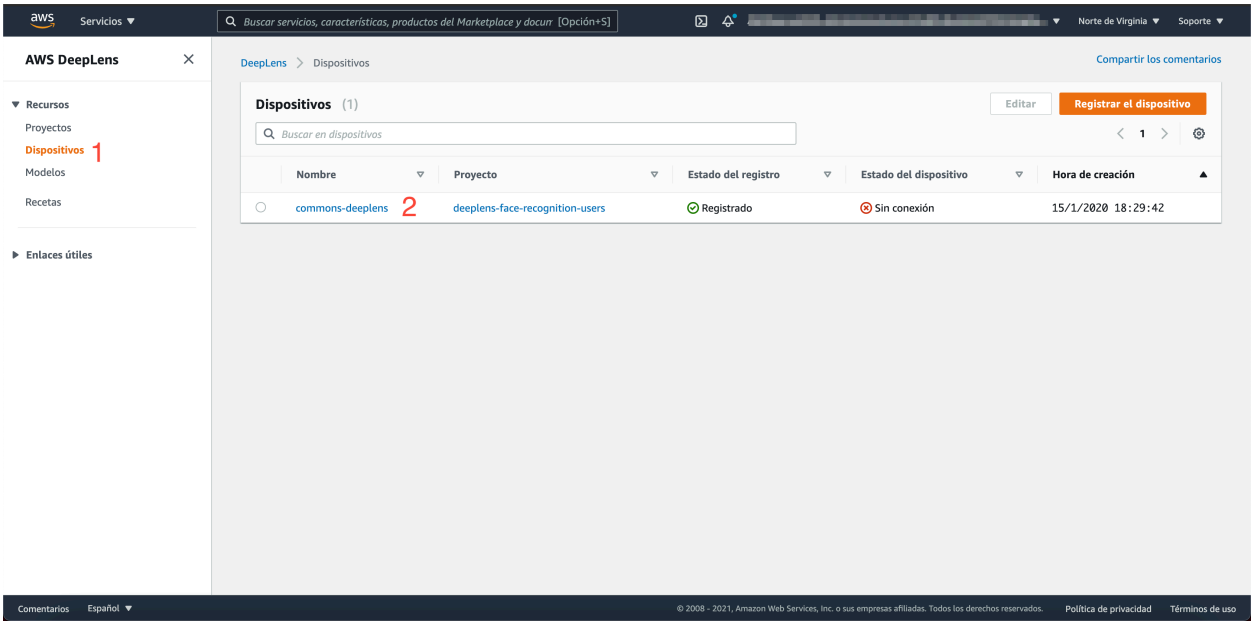


Ilustración 36: Despliegue del proyecto de implementación en el dispositivo AWS DeepLens.

En la consola de administración de la cuenta de Amazon Web Services (AWS) se busca DeepLens una ves allí se selecciona Dispositivos (1), luego se selecciona el nombre del dispositivo AWS Deeplens (2) anterior mente registrado.

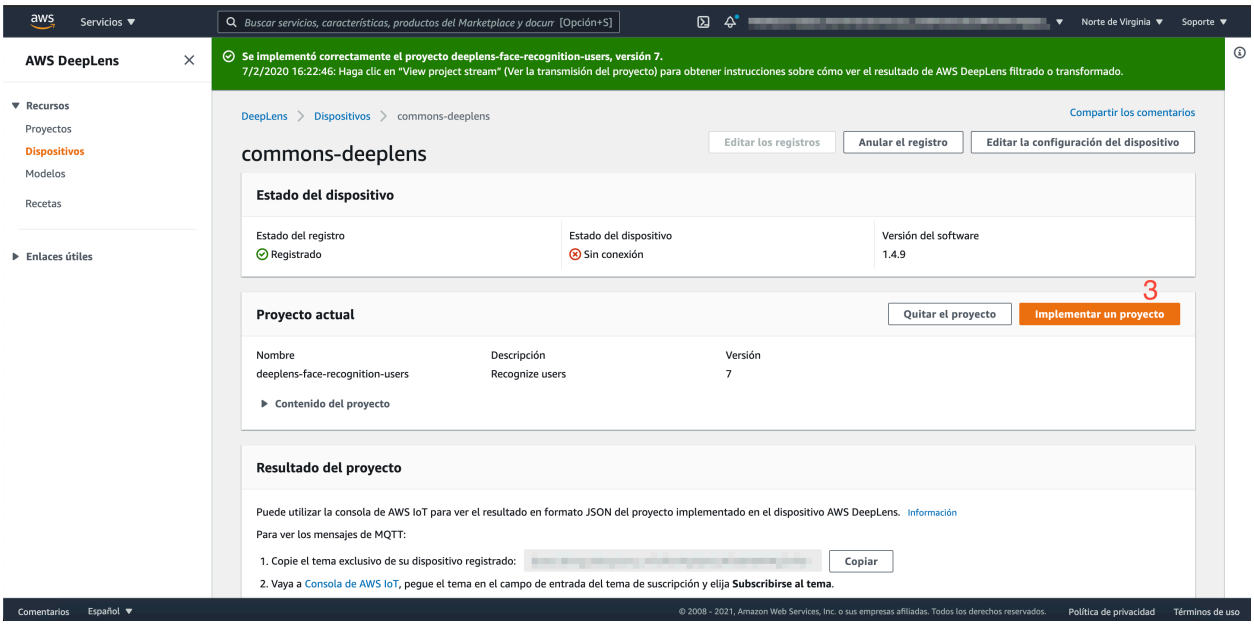


Ilustración 37: Implementar un proyecto en el dispositivo AWS DeepLens.

Fuente: Autor.

Una ves se seleccione el dispositivo se mostrará una interfaz como la ilustración 37, se selecciona "Implementar un proyecto" (3).

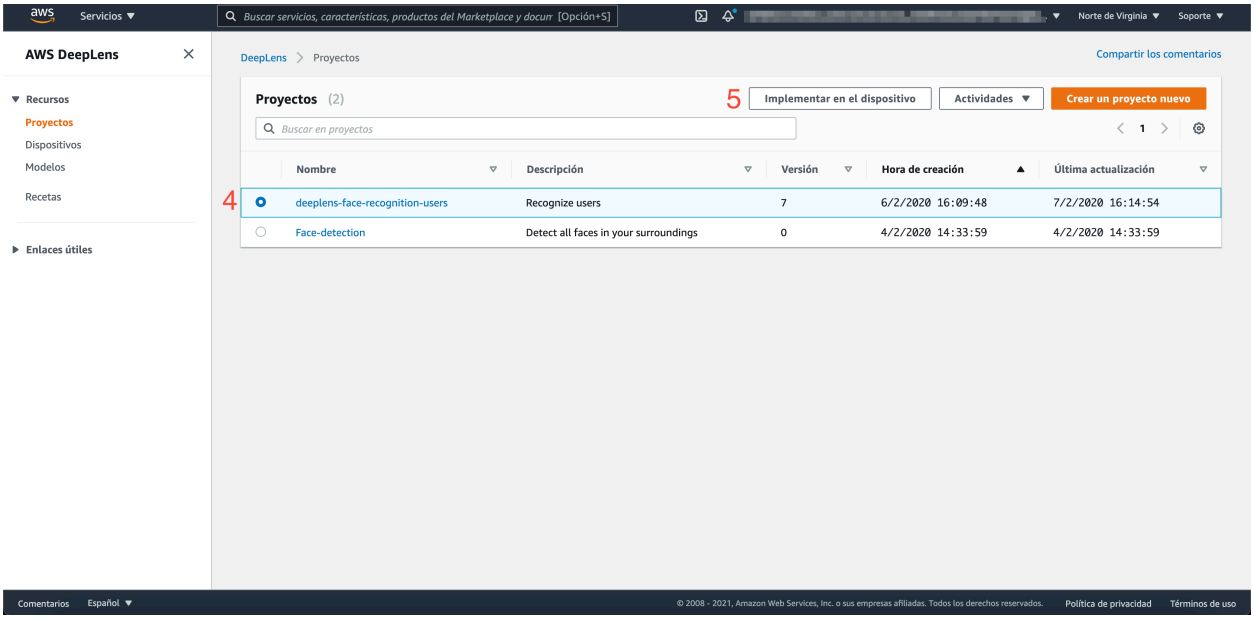


Ilustración 38: Selección del proyecto anteriormente creado para el dispositivo DeepLens.

Fuente: Autor.

Posteriormente se selecciona el proyecto anteriormente creado en el numeral 7.5.1 (4) y por ultimo se da clic en “Implementar en el dispositivo” (5).

7.5.3 Verificación de la implementación del proyecto en el dispositivo

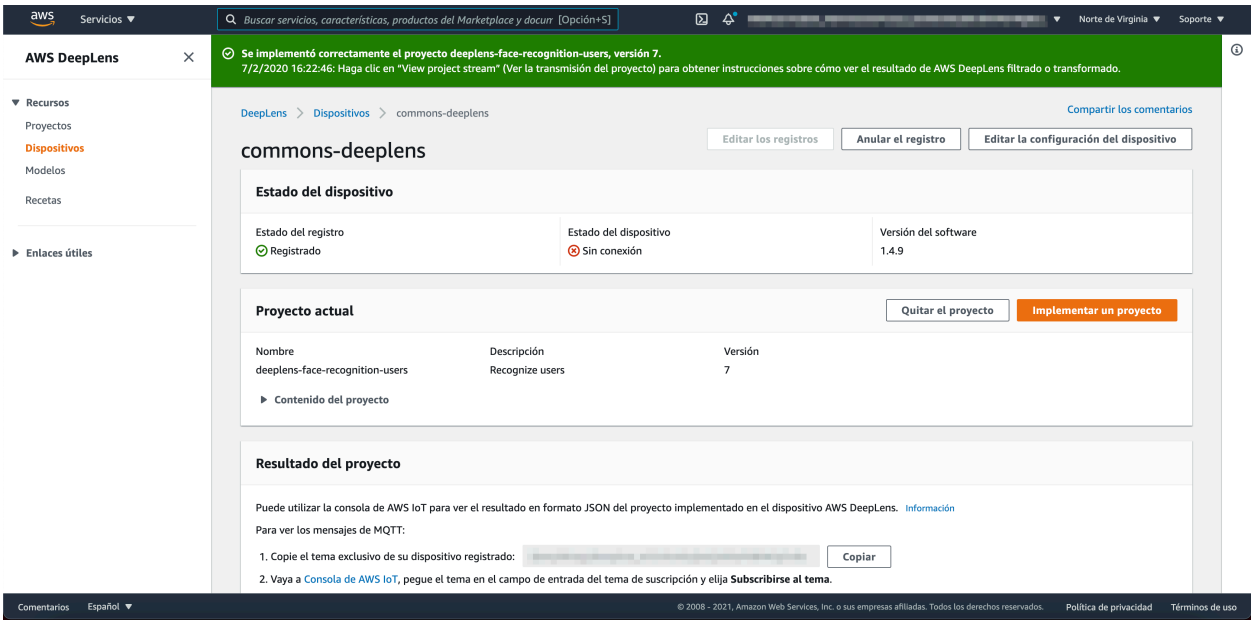


Ilustración 39: Verificación de la implementación del proyecto en el dispositivo DeepLens.

Fuente: Autor.

Para verificar el correcto despliegue del proyecto de implementación se navega a la consola de administración de Amazon Web Services (AWS) en el apartado de DeepLens y se selecciona dispositivos en la parte superior de la interfaz aparecerá un aviso de color verde el cual indica el correcto despliegue del proyecto en el dispositivo.

7.6 Generación del WebSocket en AWS API Gateway

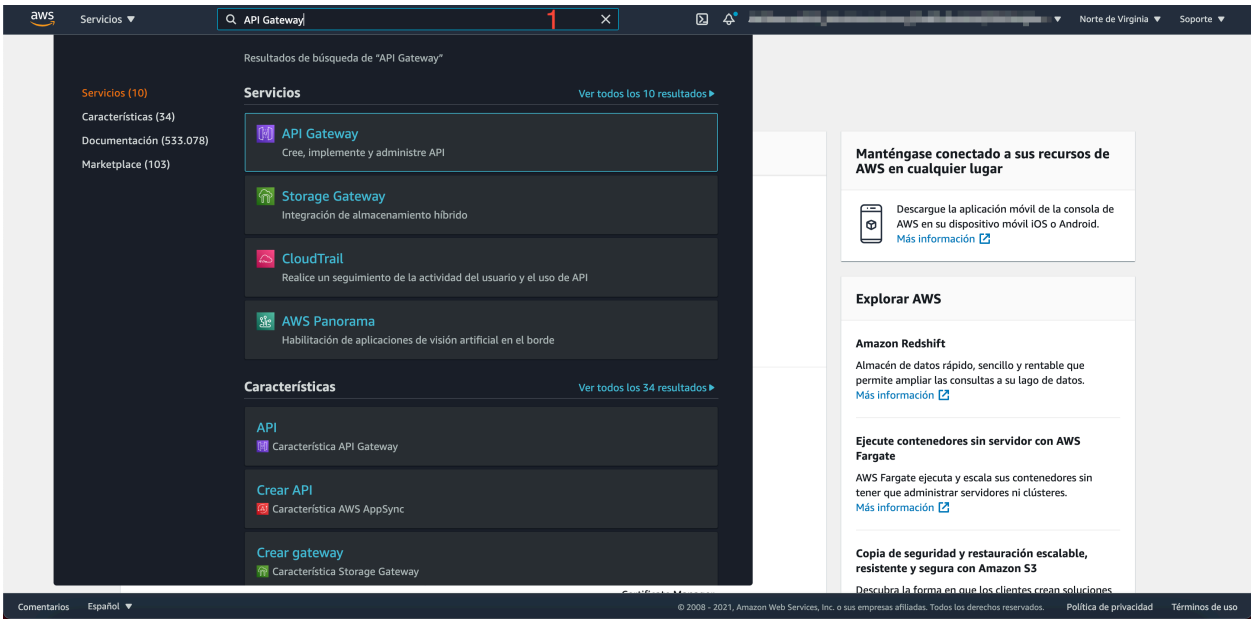


Ilustración 40: Búsqueda de API Gateway en la consola de administración de Amazon Web Services.

Fuente: Autor.

En la consola de administración de Amazon Web Services (AWS) se busca el apartado API Gateway (1) y se navega allí.

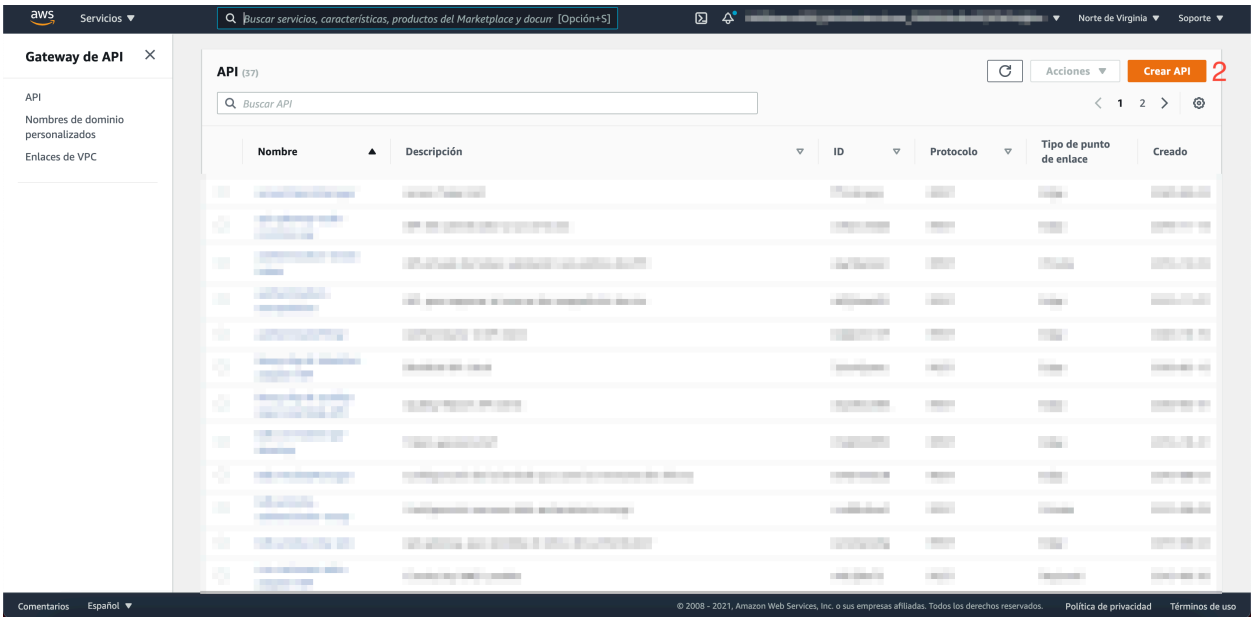


Ilustración 41: Creación de API WebSocket en la consola de administración de Amazon Web Services.

Fuente: Autor.

Paso seguido se da clic en el botón “Crear API” (2).

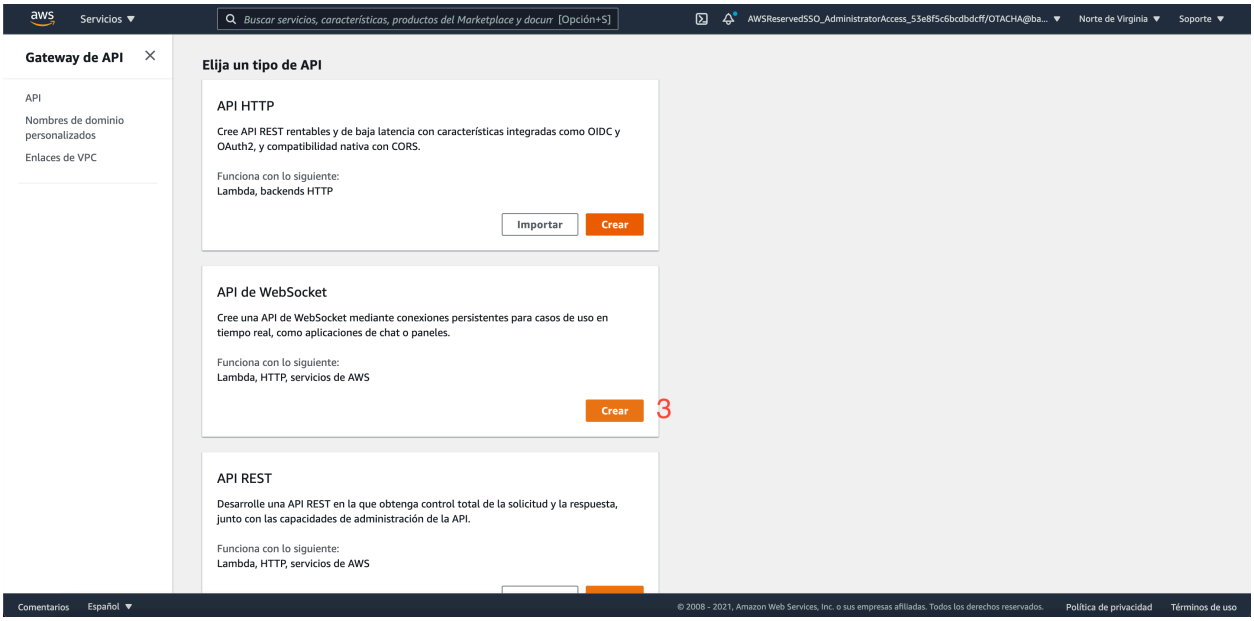


Ilustración 42: Selección tipo de API a crear en la consola de administración de Amazon Web Services.

Fuente: Autor.

En la selección de tipo de API se selecciona API de Web Socket en el botón de “Crear” (3).

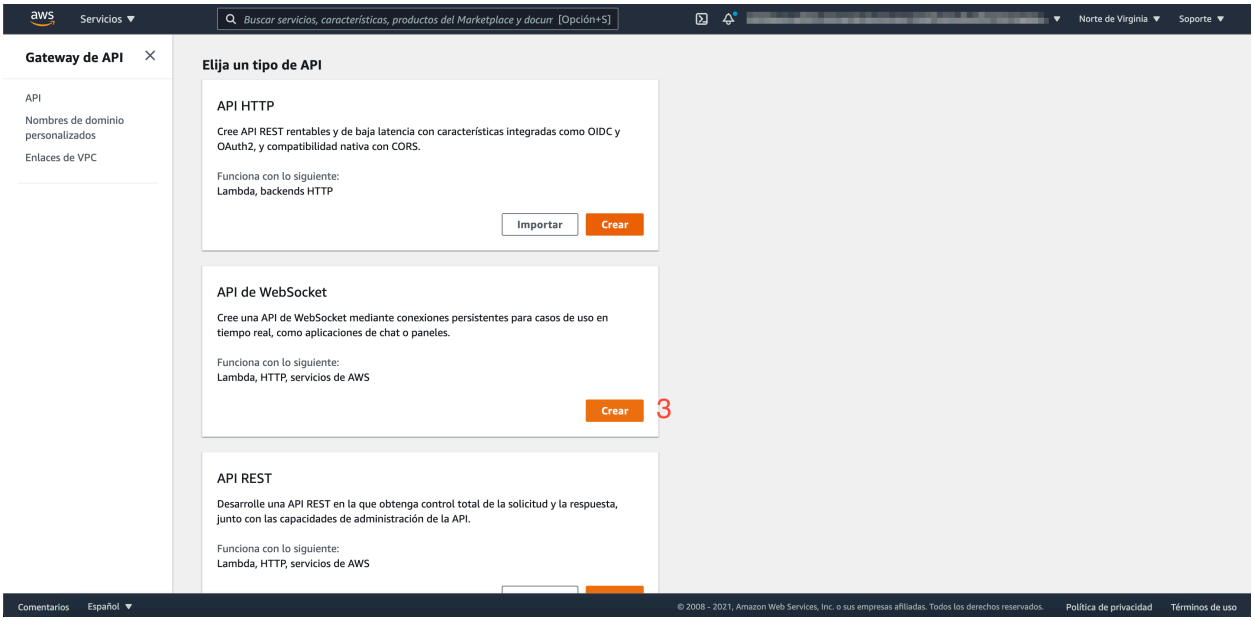


Ilustración 43: Especificación de detalles del API a crear en la consola de Amazon Web Services.

Fuente: Autor.

En la especificación de detalles del API de WebSocket a crear se le asigna un nombre (4) después se puede seleccionar de que manera se quiere que el cliente se conecte a una ruta (5) para este caso en especifico se selecciona “request.body.action” si no se asigna ninguna se asignara la ruta por defecto “default”.

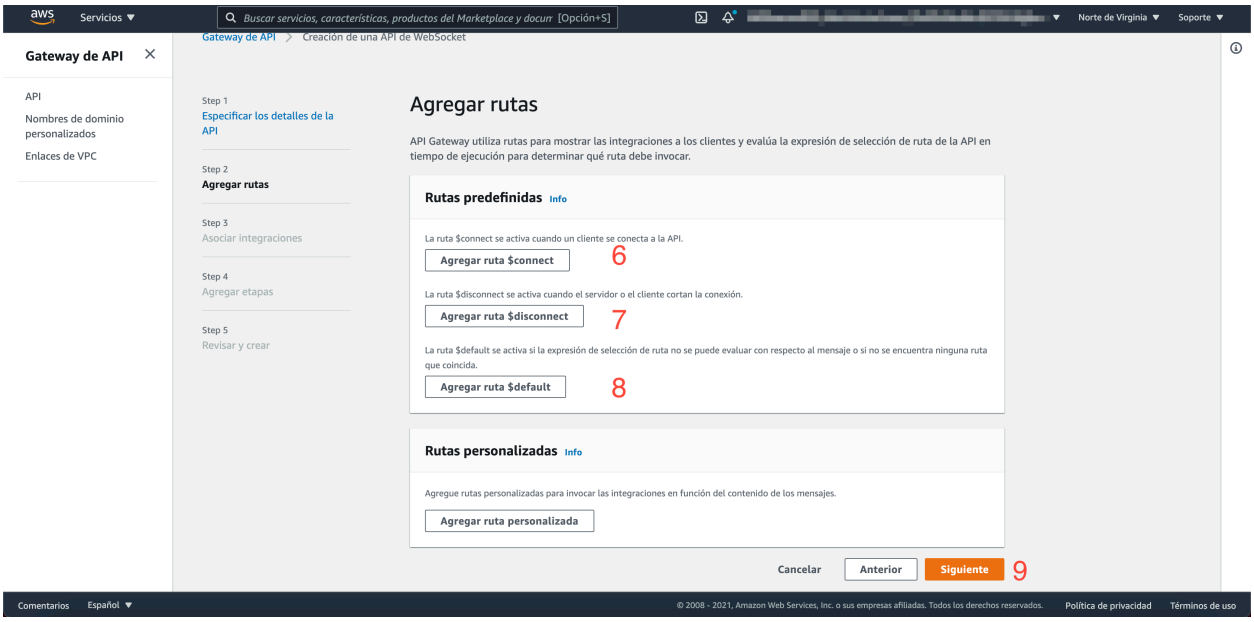


Ilustración 44: Configuración de rutas para el API de WebSocket en la consola de administración de Amazon Web Services.

Fuente: Autor.

En el apartado de agregar rutas se seleccionan las rutas sugeridas las cuales son $connect (6), $disconnect (7), $default(8) y por ultimo se da clic en “Siguiente” (9).

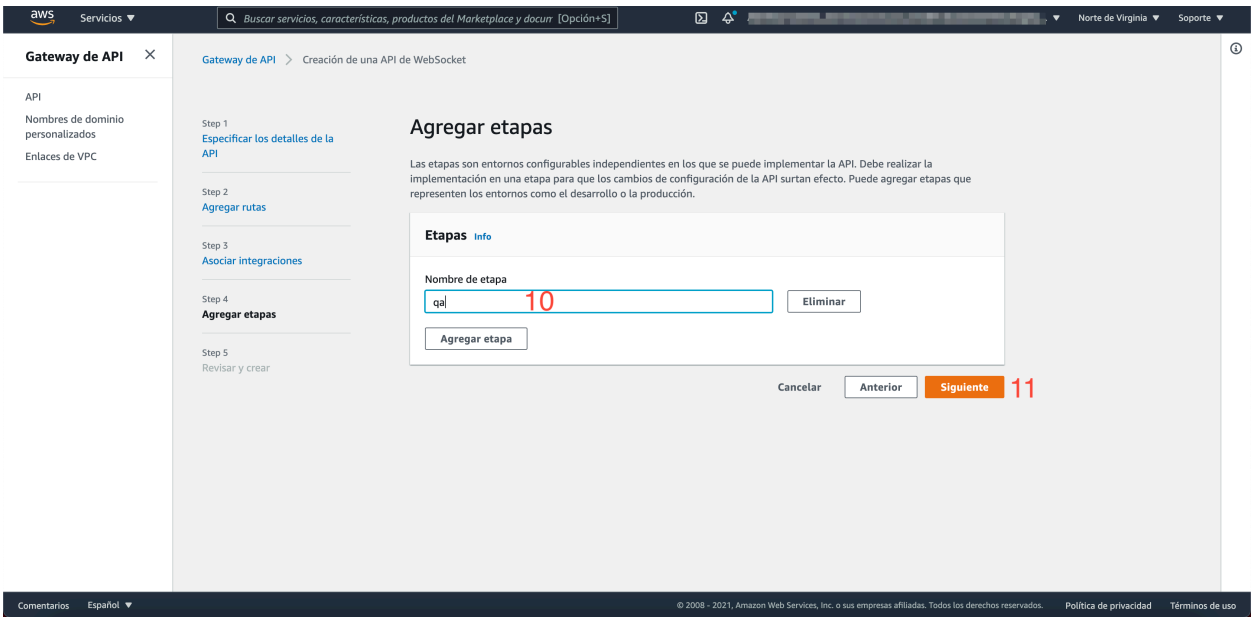


Ilustración 45: Agregar Etapa de implementación de API WebSocket en la consola de administración de Amazon Web Services.

Fuente: Autor.

En el apartado de “Agregar etapas” se pueden usar etapas previamente creadas, pero para este caso se usa una etapa nueva (11), las etapas son formas de manejar los distintos entornos de una API en este caso esta en QA “Quality Assurance” es decir aseguramiento de la calidad.

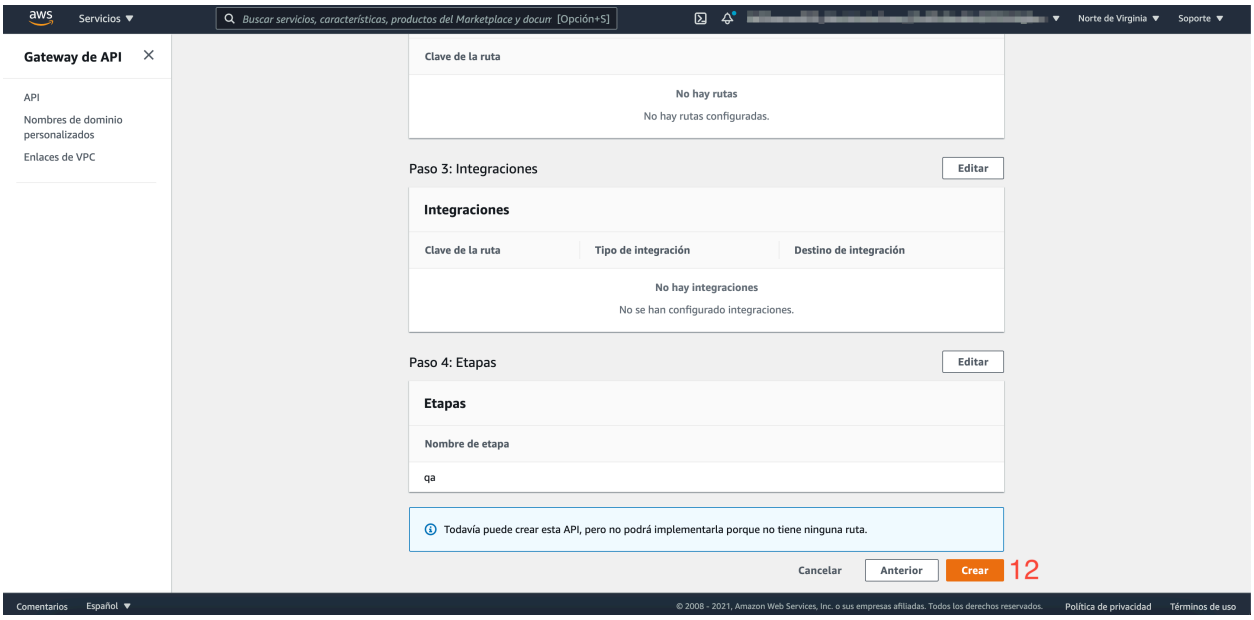


Ilustración 46: Creación final del API de WebSocket en la consola de administración de Amazon Web Services.

Fuente: Autor.

Como ultimo paso en la creación de un API de WebSocket en la consola de administración de Amazon Web Services (AWS) se hace una revisión del resumen generado del API a crear y si todo esta correcto se da clic en Crear (12).

7.7 Generación de la interfaz grafica

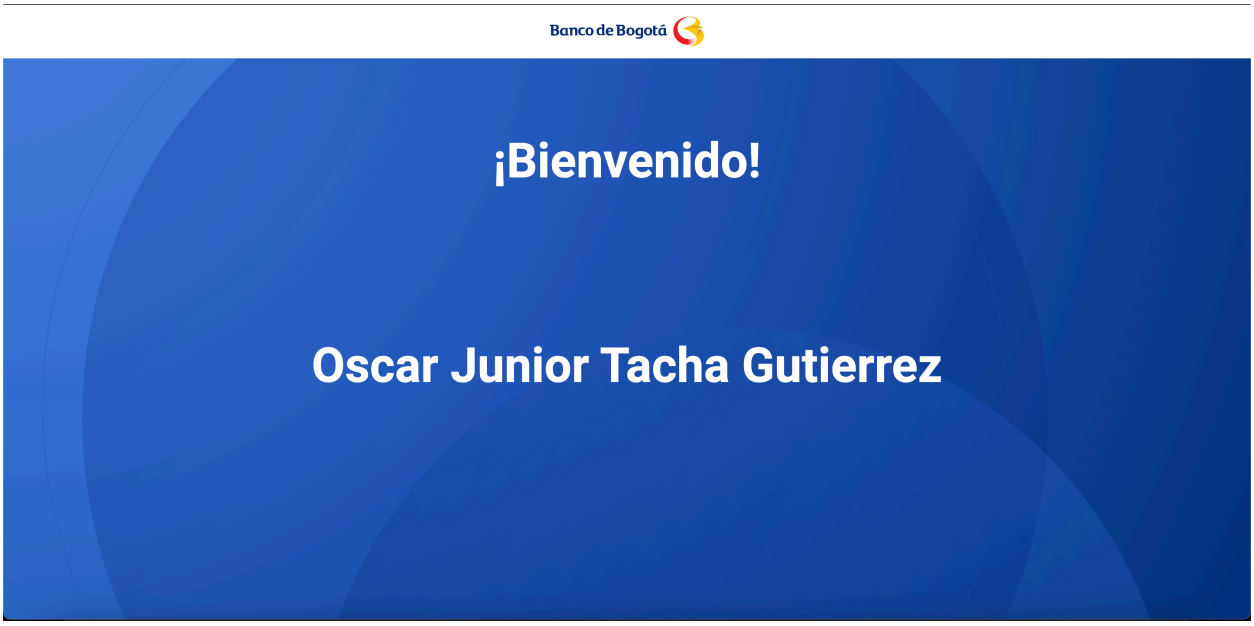


Ilustración 47: Interfaz grafica final para mostrar los resultados al usuario.

Fuente: Autor.

El diseño de la interfaz final incluye los colores característicos del Banco de Bogotá además incluye la barra de navegación con el logo del Banco de Bogotá, es una interfaz responsive la cual se adapta a los navegadores de internet mas utilizados (Google Chrome, Mozilla Firefox, Safari, Opera, Brave entre otros) y en diferentes resoluciones de dispositivos desde dispositivos móviles a grandes dispositivos como lo son Smart TV.

7.8 Integración continua

Para generar valor constante y de manera rápida evitando los posibles errores humanos se hace uso de la herramienta CircleCI para realizar la integración continua.

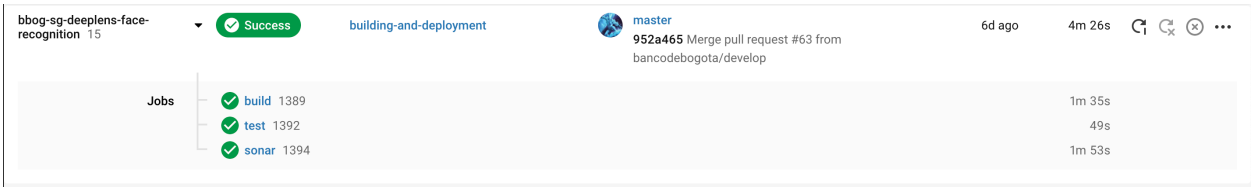


Ilustración 48: Integración continua de los componentes del prototipo de identificación de clientes.

Fuente: Autor.

7.8.1 Calidad de código

Para garantizar la extensibilidad y mantenibilidad de los componentes el Banco de Bogotá cumple con herramientas de auditoria de código como lo es SonarCloud y linters, los cuales son ejecutados en la integración continua de código como requerimientos fijos, además esto sirve para asegurar al cliente final que el producto de Software generado por el Banco de Bogotá cumple con los mas altos estándares de seguridad.

7.8.1.1 SonarCloud

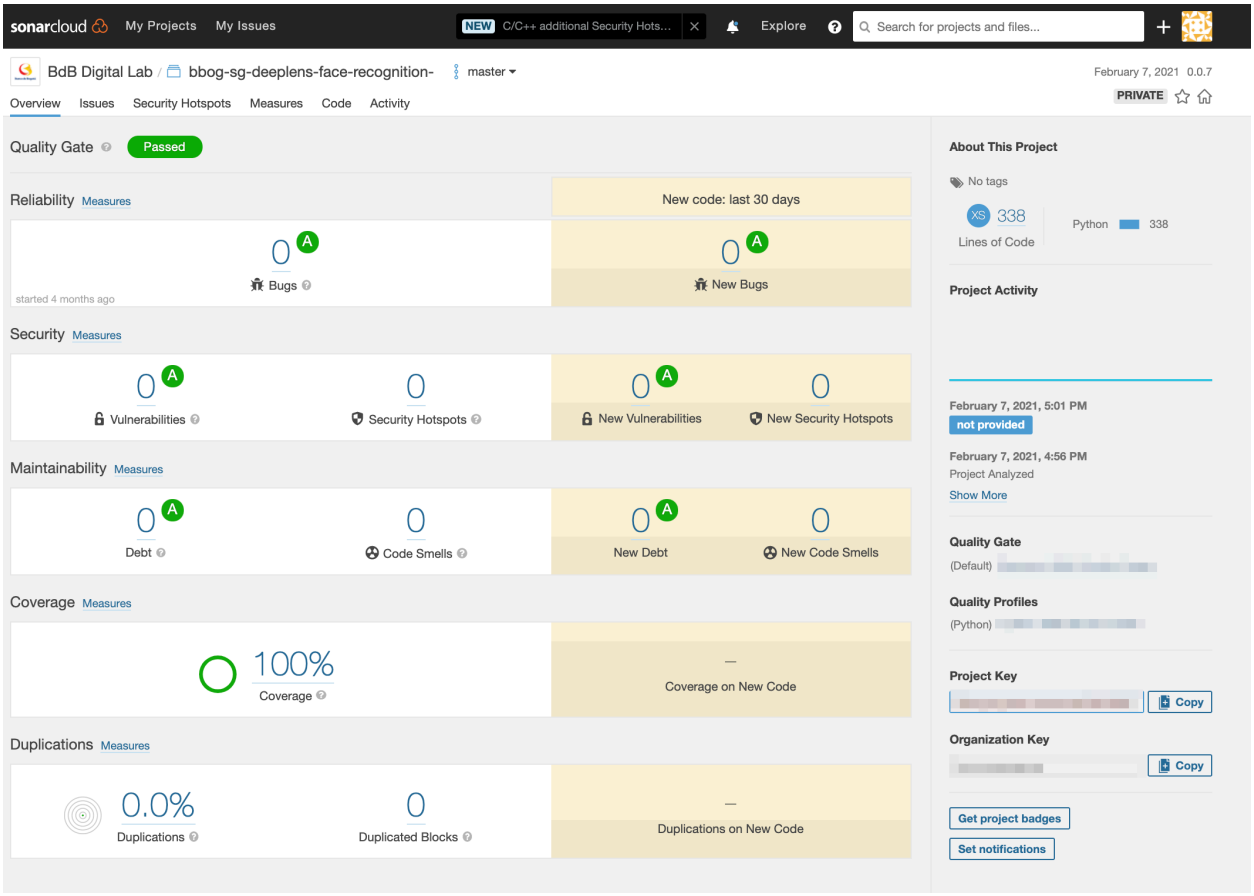


Ilustración 49: Resultados del análisis generado por SonarCloud para el prototipo de identificación de clientes.

Fuente: Autor.

En el análisis realizado por la herramienta SonarCloud se ve como se reporta la cobertura generada por las pruebas unitarias la cual es del 100% (cien porciento) además se pueden revisar otros aspectos relevantes dentro de la seguridad en el proyecto como lo son cero olores de código es decir se implementaron las mejores practicas para desarrollar este componente según la herramienta no posee “Bugs” o comportamientos inesperados del software, también tiene cero vulnerabilidades y cero “Security hotspots” o puntos de acceso de seguridad. Todo esto significa

que el proyecto generado cumple con los estándares de calidad obligatorios para el Banco de Bogotá.

7.8.1.2 Linters

Para asegurar la correcta sintaxis del código, evitar inconsistencias y mantener las mejores practicas de código limpio se usan las herramientas Pylint y eslint para realizar esta revisión en los lenguajes usados.

7.8.1.2.1 Pylint para Python

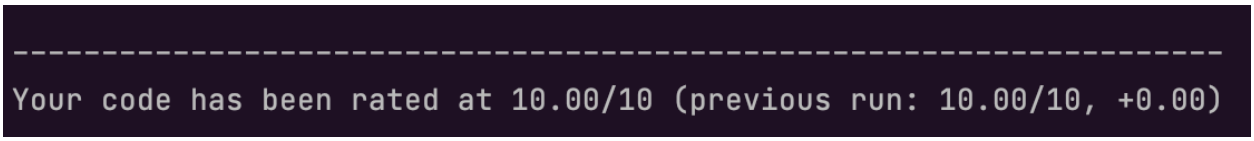


Ilustración 50: Resultado del análisis de Pylint sobre los componentes escritos en el lenguaje Python.

Fuente: Autor.

En la imagen se ve el puntaje generado para el análisis realizado por la herramienta PyLint el cual tiene un máximo de 10 (diez) y el obtenido es 10 (diez) cumpliendo con los estándares de calidad establecidos por el código.

7.8.1.2.1 ESLint para JavaScript y TypeScript

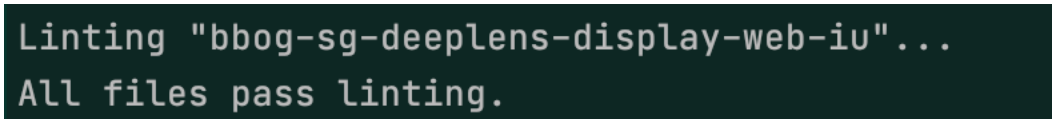


Ilustración 51: Resultado del análisis de ESLint sobre los componentes escritos en el lenguaje TypeScript.

Fuente: Autor.

En la revisión de los lenguajes TypeScript y JavaScript por la herramienta ESLint se aprecia que no maneja puntaje, por lo contrario, dice que todos los archivos pasaron correctamente la revisión.

Capitulo 8

Análisis de los Resultados

8.1 Resultados obtenidos por el dispositivo DeepLens

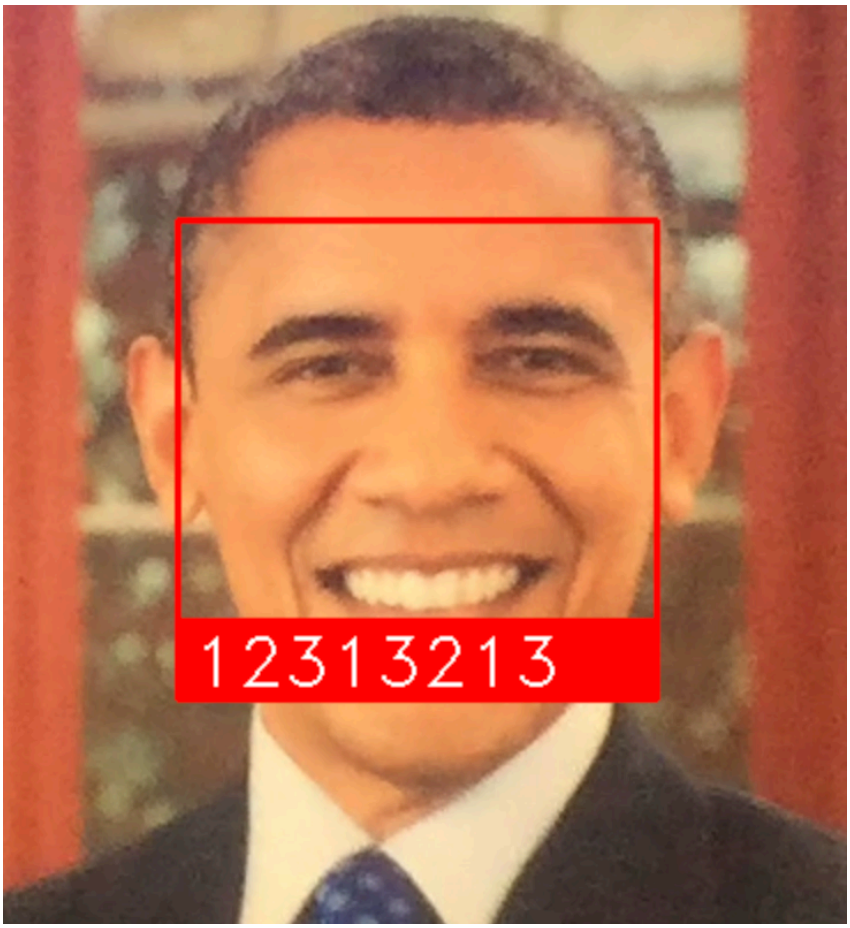


Ilustración 52: Procesamiento de imágenes en el dispositivo AWS DeepLens en tiempo real y su salida en un video de tiempo real identificando los rostros conocidos.

Fuente: Autor.

En la ilustración 52 se ve como un rostro conocido es identificado por su numero de identificación

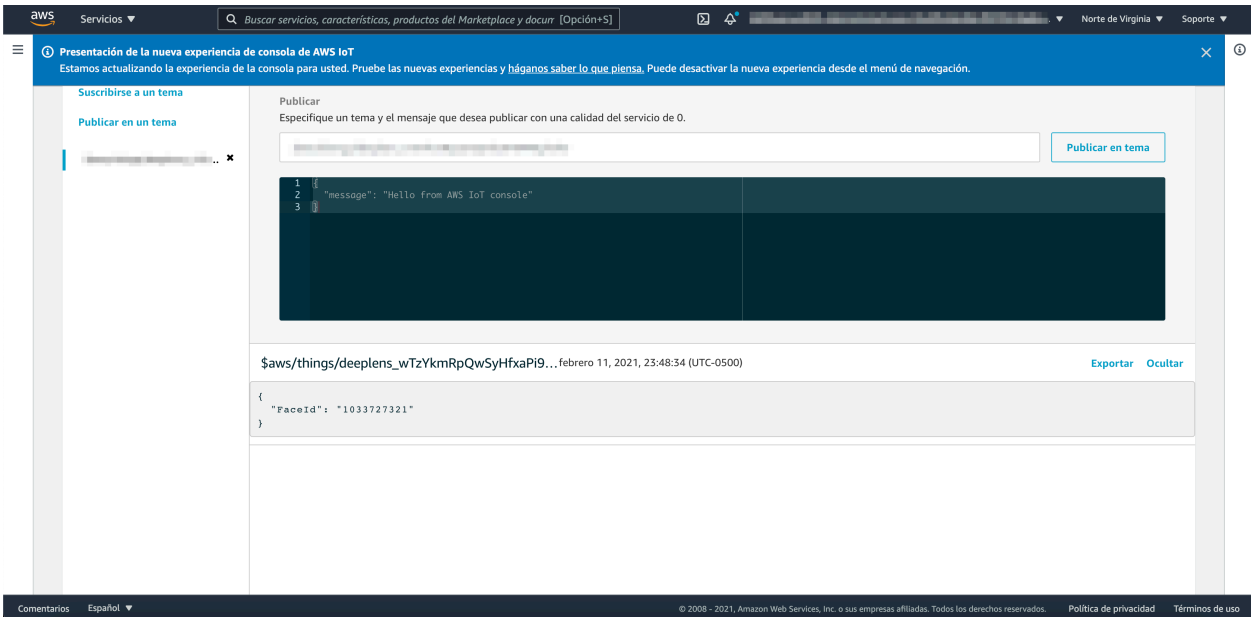


Ilustración 53: Resultados obtenidos por el dispositivo DeepLens.

Fuente: Autor.

Se revisan los resultados del proyecto de inferencia obtenidos por el dispositivo DeepLens el cual da un numero de cedula asociado al rostro capturado en la imagen el porcentaje de confiabilidad es del 80% mínimo en caso contrario se obtendrá como resultado “Unknown”.

Una ves se obtiene el numero de identificación de la persona asociada al rostro se puede consultar a la persona en la base de datos del Banco de Bogotá (CRM) y obtener sus nombres y apellidos.

8.2 Beneficios obtenidos

Amazon Web Services (AWS) provee una capa gratuita en todos sus servicios, lo cual resulta ser provechoso en cuanto a la reducción de costos por identificación de cada persona. Al ser un prototipo basado en servicios en la nube permite ser replicado e implementado en diferentes localizaciones, no solamente en el laboratorio digital del Banco de Bogotá, cabe aclarar que cada prototipo es independiente cualquier otro prototipo.

La arquitectura propuesta para el dispositivo provee una línea base para generar nuevos métodos de identificación y autenticación de los usuarios en las instalaciones del Banco de Bogotá, la generación de las vistas del "modelo 4+1" permite que las personas entiendan de una manera fluida como funciona el prototipo implementado.

Capitulo 9

Conclusiones

9.1 Verificación, contraste y evaluación de los objetivos

El objetivo principal que se presentó al inicio de este trabajo de investigación era el de desarrollar un prototipo de identificación de clientes bancarios el cual utiliza los principios planteados en los microservicios aplicándolos en el front-end y en el backend utilizando como línea base la aplicación de algoritmos de machine learning y recognition. Al lograr dicho modelo el cual se expone en el capitulo 5, se puede demostrar que es posible elaborar un proyecto que es una composición de pequeñas partes de otro los cuales llamamos componentes que exponen microservicios y que puede mantenido por equipos diferentes que se encargaran de mejorar las funcionalidades o dar soporte a las actuales.

Lo que se busca con la ultima afirmación es la identificación de clientes facilitando la manutención y extensibilidad del prototipo.

Dentro del desarrollo del proyecto se realizó la prueba de concepto del modelo arquitectural planteado, realizando un prototipo implementando alguna de las funcionalidades que se encontraban descritas dentro de el, esto con el fin de verificar el impacto que genera la identificación de clientes, tanto en la velocidad de despliegue como en los atributos de calidad que se proponen dentro del trabajo de la Dirección Digital del Banco de Bogotá, además de reducir los procesos manuales realizados por el cliente.

Al realizar la interfaz se desarrollo una interfaz completamente responsive que se adapta a múltiples navegadores y múltiples resoluciones de dispositivos para lo cual se puede utilizar en situaciones de seguridad.

9.2 Síntesis del modelo propuesto

El modelo de arquitectura basado en microservicios REST se compone de cuatro (4) componentes principales:

- Dispositivo DeepLens.
- Publicador de eventos.
- Subscriptor de eventos.
- Cliente web (FrontEnd).

Los cuales son independientes, pero se comunican entre si a través de llamados rest haciéndolos extensibles y mantenibles por si solos es decir no son componentes monolíticos que a largo plazo se vuelve una carga insustentable.

9.3 Aportes originales

Dentro de los aportes que se hacen al realizar el modelo de arquitectura y el prototipo inicial de implementación se pueden encontrar el uso de componentes stateless es decir que permiten fácilmente la implementación de varios prototipos y además por estar desarrollado en un entorno web permite que el prototipo sea desplegado en cualquier sitio solamente contando con una conexión a internet.

Además, permite que se realice una monitorización desde cualquier dispositivo móvil añadiendo un extra para revisar quien entra o sale de una instalación.

9.4 Trabajos o publicaciones derivadas

Del desarrollo del modelo de arquitectura y de la construcción del prototipo se deriva el proyecto inicial en el cual no se encuentran proyectos similares aparte establece una línea base para realizar identificación facial para lo cual es idóneo para un posible método de autenticación de usuarios. Este proyecto cumple con los estándares de desarrollo y de despliegue continuo que se proponen dentro de la Dirección Digital , además de estar desplegados en un ambiente de uso exclusivo de los integrantes del laboratorio digital, los cuales también pueden acceder al repositorio donde está alojado el código fuente.

Capitulo 10

PROSPECTIVA DEL TRABAJO DE GRADO

10.1 Líneas de investigación futuras

El desarrollo que se propone este trabajo fue el de realizar un prototipo capaz de identificar clientes bancarios solamente capturando un fotograma con su foto. Una línea que se puede abordar es el registro automático de la persona haciendo un proceso de autenticación previo y suministrando la fotografía de la persona si es cliente, si no lo es seria posible brindar al usuario la opción de obtener un producto con el Banco de Bogotá.

10.2 Trabajos de investigación futuros

En este trabajo de investigación es posible continuar en la implementación de los prototipos de identificación de clientes que se proponen en el modelo arquitectural tomando como base el proyecto creado en el prototipo.

Otro trabajo que se propone realizar en el futuro, teniendo en cuenta que ya se está realizando la captura de fotogramas es realizar una prueba de vida para disminuir el fraude que se pueda presentar con el uso de la herramienta, también se propone como método de autenticación dentro de las oficinas de Banco de Bogotá donde el cliente pueda solicitar algún servicio haciendo uso de su rostro demostrando que es una persona viva y no una fotografía, un video entre otros métodos de fraude que se puedan crear.

Un trabajo futuro propuesto es el de mejorar la base de rostros que para el presente proyecto se realizo con un archivo JSON alojado en un AWS bucket S3 lo cual se puede alojar en una base Entidad-Relación para mejorar la consistencia de la data, tener contingencia contra fallos y añadir datos relevantes para el análisis facial con lo cual se pueden actualizar los registros cada ves que el usuario hace uso del prototipo de identificación de clientes bancarios.

Referencias

Rouse, M. (16 de agosto de 2019). *Identity management (ID management)*. Obtenido de Search Security: https://searchsecurity.techtarget.com/definition/identity-management-ID-management.

Ross, A., & A. J. (2008). *Handbook of Biometrics.* Boston, MA: Springer.

Amazon Web Services. (17 de 08 de 2019). *Amazon Web Services*. Obtenido de ¿Que es informática en la nube?: https://aws.amazon.com/es/what-is-cloud-computing/.

Mell, P., & T. G. (2011, septiembre). The NIST Definition of Cloud Computing. *Instituto Nacional de Estándares y Tecnología (NIST)*, 2-3.

Lecun, Y., Y. B., & G. H. (2015). Deep Learning. *Nature International Journal of Science*.

Bishop, C. (2006). *Pattern Recognition and Machine Learning.* Cambridge: Springer.

AWS. (2019). *AWS DeepLens*. Obtenido de Amazon Web Services: https://aws.amazon.com/es/deeplens/.

W3C. (2019). *Server-Sent-Events*. Obtenido de W3C: https://www.w3.org/TR/eventsource/.

Ian Fette. (2011). Websocket protocol . Obtenido de RFC: https://tools.ietf.org/html/rfc6455.

Banco de Bogotá (2012). Historia Banco de Bogotá. Obtenido de https://www.bancodebogota.com/wps/portal/banco-de-bogota/bogota/nuestra-organizacion/nuestro-banco/historia-mision-vision.

King Davis. (2017). High Quality Face Recognition with Deep Metric Learning. Obtenido de Dlib C++ Library : http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html.

B Gustavo. (2017). ¿Qué es GitHub y para qué se utiliza? Obtenido de Hostinger: https://www.hostinger.co/tutoriales/que-es-github/.

Quality Devs. (2019). ¿Qué es Angular y para qué sirve? Obtenido de Quality Devs: https://www.qualitydevs.com/2019/09/16/que-es-angular-y-para-que-sirve/.

Mohamed Amine. Hmani and D. Petrovska-Delacrétaz, (2018). State-of-the-art face recognition performance using publicly available software and datasets.

Banco de Bogotá (2019) Informe de gestión 2019 Obtenido de http://bancodebogota.com/wps/themes/html/banco-de-bogota/pdf/relacion-con-el-inversionista/sobre-el-banco/informe-de-gestion/2019/ig-bdb-2019_cap-1.pdf.

Chang, Kai-Wei & Hsieh, Cho-Jui & Lin, Chih-Jen. (2008). Coordinate Descent Method for Large-scale L2-loss Linear Support Vector Machines. Journal of Machine Learning Research. 9. 1369-1398. 10.1145/1390681.1442778.

Soliman Habib, Mona. (2020). Biomedical Named Entity Recognition Using Support Vector Machines: Performance vs. Scalability Issues.

Amazon Web Services (2021) What is DeepLens Obtenido de https://docs.aws.amazon.com/es_es/deeplens/latest/dg/what-is-deeplens.html.

Amazon Web Services (2021) Getting Started with AWS DeepLens Obtenido de https://docs.aws.amazon.com/deeplens/latest/dg/deeplens-getting-started.html.

Amazon Web Services (2021) Create and Publish an AWS Deepens Inference Lambda Function Obtenido de https://docs.aws.amazon.com/deeplens/latest/dg/deeplens-inference-lambda-create.html.

Amazon Web Services (2021) About WebSocket APIs in API Gateway https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-websocket-api-overview.html.

Amazon Web Services (2021) Creating a bucket https://docs.aws.amazon.com/AmazonS3/latest/userguide/create-bucket-overview.html.

Amazon Web Services (2021) Create a Lambda Function (Console) https://docs.aws.amazon.com/lex/latest/dg/gs-bp-create-lambda-function.html.

Philippe Kruchten. (1995). Planos Arquitectónicos: El Modelo de "4+1" Vistas de la Arquitectura del Software. Recuperado de http://cic.puj.edu.co/wiki/lib/exe/fetch.php?media=materias:modelo4_1.pdf.

Amazon Web Services (2021). Boto3 Documentation. Recuperado de https://boto3.amazonaws.com/v1/documentation/api/latest/index.html.

Ageitgey. Face Recognition Python. Recuperado de https://github.com/ageitgey/face_recognition.

Banco de Bogotá (2020). Recuperado el 25 de diciembre del 2020 de https://www.bancodebogota.com/wps/portal/banco-de-bogota/bogota/nuestra-organizacion/nuestro-banco/historia-mision-vision.

Shirvany Y, Koka S, Mahdieh360, Kumar A. (2020) Recuperado el 27 de diciembre del 2020 de https://aws.amazon.com/es/deeplens/community-projects/OneEye/.q