**Sistema Inteligente de Seguridad para el Hogar**

**«SafeHome»**

*Documento de Diseño*

**Versión 1.0**

**Historial de Revisiones**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| 12/10/2019 | 1.0 | Creación del documento de diseño | Carlos La Rosa Sánchez |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Tabla de contenido

[1 INTRODUCCIÓN 4](#_Toc21807136)

[1.1 Alcance 4](#_Toc21807137)

[1.2 Objetivos 4](#_Toc21807138)

[2 DEFINICIÓN DE LA ARQUITECTURA DEL SISTEMA 5](#_Toc21807139)

[2.1 Definición de Niveles de Arquitectura del Sistema 5](#_Toc21807140)

[3 DISEÑO DEL MODELO DE CLASES DEL SISTEMA 6](#_Toc21807141)

[4 MODELO FÍSICO DE DATOS 8](#_Toc21807142)

[4.1 Modelo Físico de Datos 8](#_Toc21807143)

# INTRODUCCIÓN

El presente documento, documento de diseño del sistema, revela todo lo concerniente al diseño (en alto nivel) del sistema inteligente de seguridad para el hogar (SISH). Se documenta la arquitectura del sistema, diseño de las clases y el modelo de datos, así como la definición de los documentos para las colecciones de datos del sistema.

## Alcance

Realizar un sistema de control de seguridad que implementa algoritmos de reconocimiento facial, manteniendo un registro y enviando notificaciones siempre que este detecte a una persona. Además, permitirá a los habitantes ingresar a su hogar haciendo uso del reconocimiento facial y un pin numérico único por habitante.

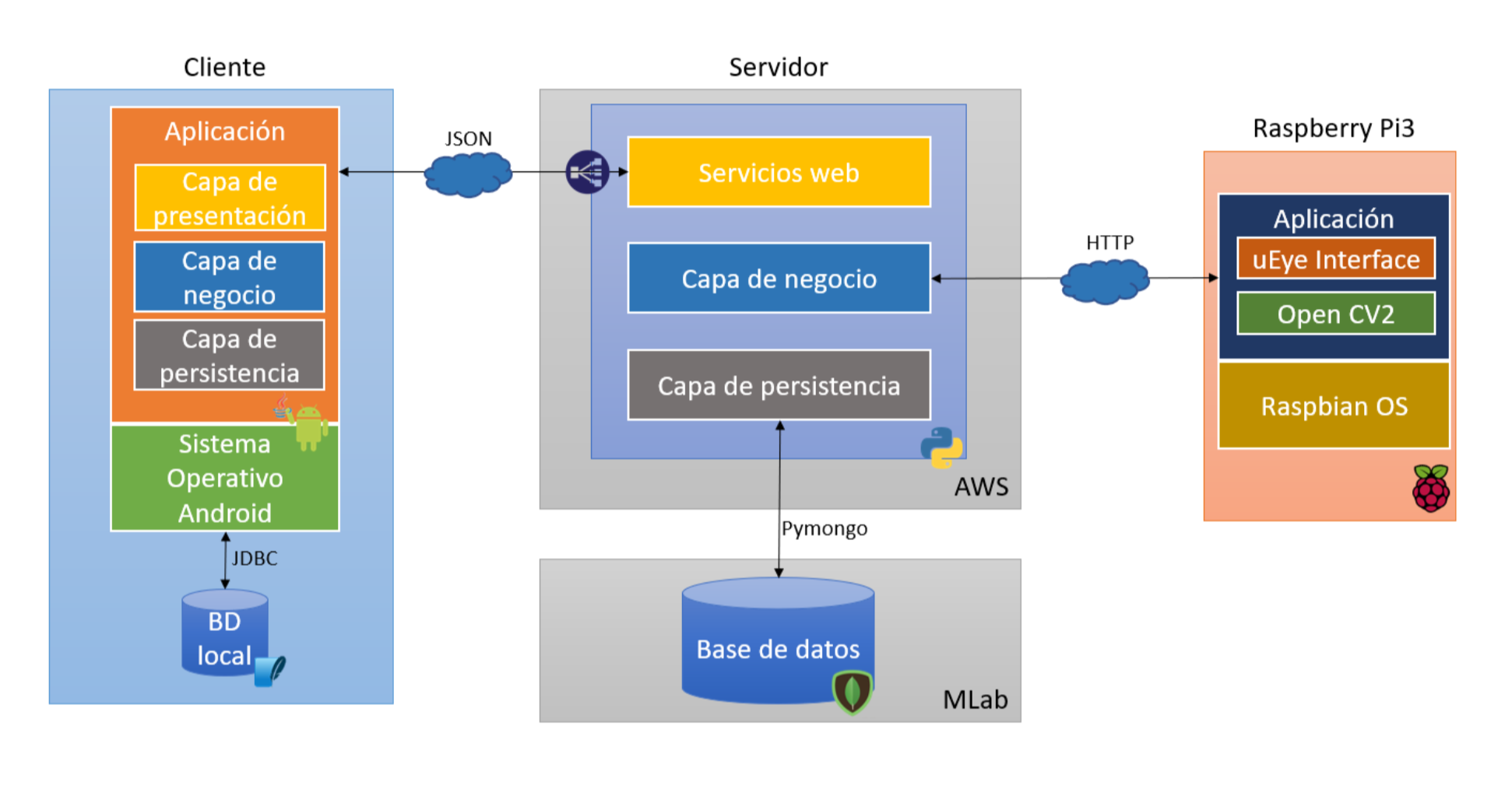
## Objetivos

Los objetivos principales del documento de diseño son los siguientes:

* Dar a conocer el diseño de la arquitectura de SISH.
* Presentar los diagramas de clases, tanto del nivel del cliente como del RaspBerry.
* Presentar el modelo de datos, así como una vista previa a su implementación.

# DEFINICIÓN DE LA ARQUITECTURA DEL SISTEMA

Considerando las tres: Cliente, Servidor y RaspBerry, para gestionar el desarrollo del sistema de una manera más ordenada y clara además de permitir un acceso remoto se ha decidido usar una arquitectura cliente/servidor con 3 capas.



## Definición de Niveles de Arquitectura del Sistema

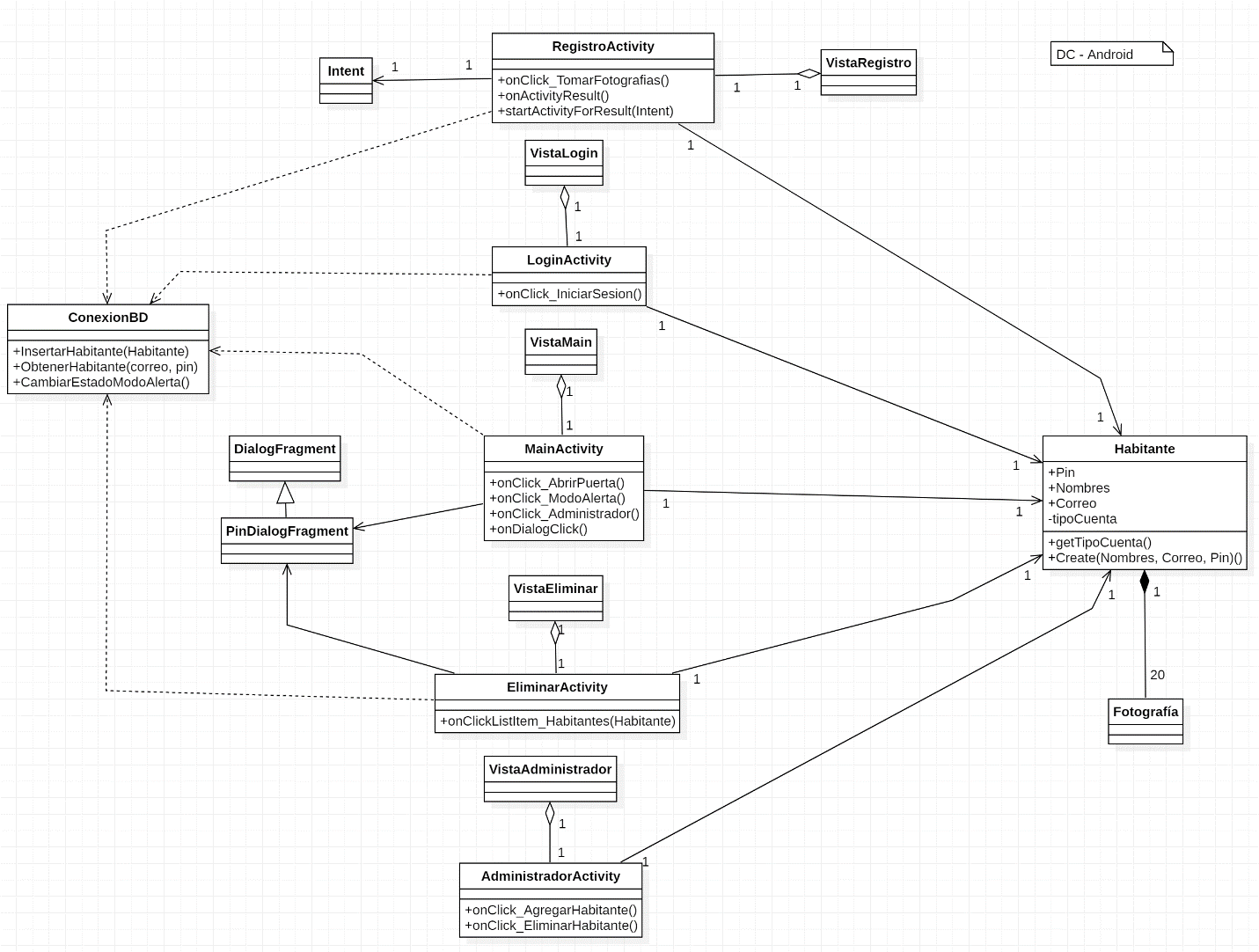
* Nivel cliente: Capa donde estarán desplegadas las aplicaciones que tendrán acceso a la nube además de solicitar el historial de los mensajes enviados. Se utilizará una aplicación desarrollada en Android para desplegar la interfaz en la cual el cliente puede visualizar los datos obtenidos del servidor en la nube. Para ello la aplicación contará con 3 capas, la capa de presentación, de negocio, y de persistencia ya que cuenta con base de datos local del propio dispositivo móvil.

Este nivel se comunicará con el servidor en la nube para la creación de usuarios, subir datos y para recibir notificaciones.

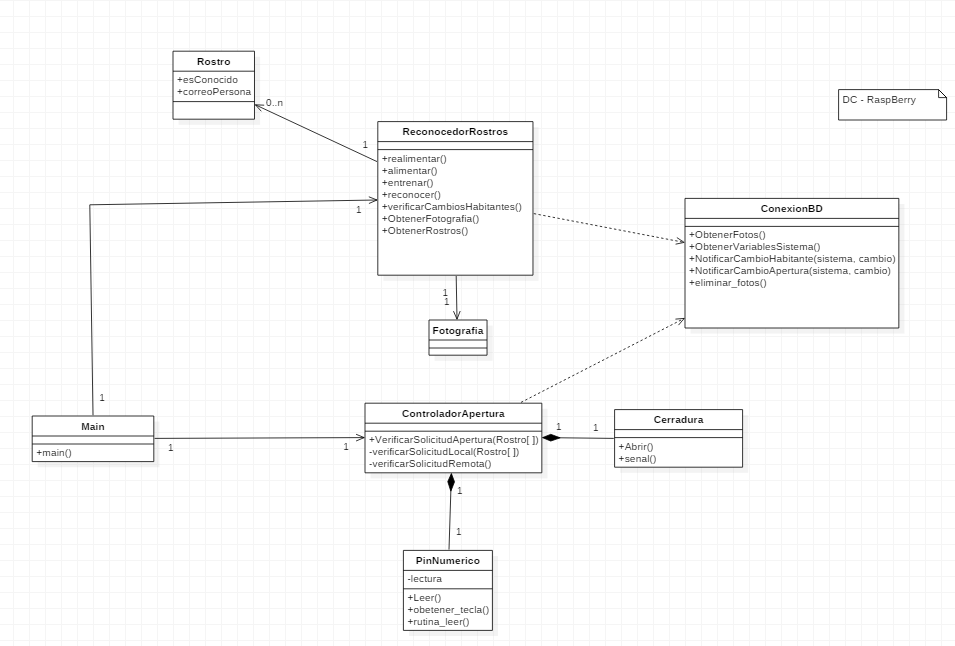
* Nivel servidor: Capa que soportará la lógica que esta alojada en la nube (almacenar los mensajes y enviarlos a la capa de hardware). Para esta capa se utilizará MongoDB en la nube MLab donde se almacenará datos de los usuarios y de las características faciales que se quieran admitir como válidas y el modelo y comunicación será visto en la capa de persistencia. Al mismo tiempo la lógica de la aplicación también se ejecutará en la capa de negocio del servidor alojado en Amazon Web Services. Los datos que se recibe o envía a la aplicación Android del nivel cliente se manejan en la capa de servicios web.
* RaspBerry Pi3: Hardware que recibe información del servidor en la nube. El hardware utilizado para controlar este sistema será el Raspberry pi 3 el cual recibirá datos y lógica del servidor en la nube para operar con estas luego retornará una respuesta a esta última.

# DISEÑO DEL MODELO DE CLASES DEL SISTEMA

Se consideran dos diagramas de clases generales dentro del sistema, los cuales son:



*Diagrama de clases nivel Cliente (Android).*



*Diagrama de clases nivel RaspBerry Pi3 (Python).*

# MODELO FÍSICO DE DATOS

El sistema tendrá como gestor de base de datos a MongoDB. MongoDB es un gestor de base de datos No Relacional, esto significa que no presenta una estructura similar a los gestores de base de datos Relacionales (MariaDB, SQL Server, PostgreSQL, MySQL, etc.). MongoDB tiene cuatro características fundamentales que lo hace un candidato perfecto para nuestro proyecto: MongoDB almacena datos en documentos BSON flexibles, es decir, cada documento puede contener diferentes campos y las estructuras de datos se pueden ir modificando. El modelo de documentos concuerda con los objetos del código de la aplicación, lo que facilita trabajar con datos.

Las consultas ad-hoc, la indexación y la agregación en tiempo real permiten acceder a los datos y analizarlos con gran eficacia.

MongoDB es una base de datos distribuida, por lo que es fácil de usar y proporciona una elevada disponibilidad, escalabilidad horizontal y distribución geográfica.

## Modelo Físico de Datos

**Colección de documentos de tipo “habitante”**:

Se almacenarán los datos de los habitantes del hogar en documentos de tipo habitante, siendo los atributos los siguientes:

* “\_id”: Object ID de cada documento.
* “nombres”: Nombres del habitante.
* “apellidos”: Apellidos del habitante.
* “correo”: Correo del habitante.
* “tipoCuenta”: El tipo de cuenta del habitante, siendo 0 si es habitante normal y 1 si es de tipo administrador del hogar.
* “primeraVez”: Será verdadero si la cuenta de habitante es nueva, esto permitirá crear el pin al primer inicio de sesión.
* “pin”: Pin del habitante.
* “fotografías”: Arreglo de enlaces donde estarán alojadas las fotografías del habitante registrado.



**Colección de documento único de tipo “variables\_sistema”**:

Se almacenarán las variables del sistema para su funcionamiento, será un único documento dentro de la colección y se actualizará siempre que sea necesario. Los atributos serán los siguientes:

* “\_id”: *Object ID* de cada documento.
* “modoAlerta”: Será verdadero cuando el modo alerta está activado y falso cuando no lo esté.
* “puertaAbierta”: Será verdadero cuando algún habitante haya abierto la cerradura por cualquier medio, esto funcionará para poder generar notificaciones cada vez que se abra una puerta.
* “notificarNuevosRostros”: Será verdadero cada vez que la cámara encuentre rostros, almacenando también información de los rostros encontrados.
* “últimosRostrosReconocidos”: Arreglo de los últimos rostros reconocidos por la cámara.