**Documento de Arquitectura**

***Sistema de recomendación, búsqueda y venta de   
productos***

**Versión 2.0**

***Fecha: 24/10/2019***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CONTROL DE VERSIONES | | | | | | |
| Versión | Hecha por | Revisada por | Aprobada por | Fecha | Motivo | |
| 1.0 |  |  |  | 24/10/2019 | Versión original | |
| 2.0 |  |  |  | 01/11/2019 | Actualización de arquitectura | |
|  |  |  |  |  |  | |

INDICE

## 

[**Descripción general**](#_bincxbgu0gj3) **4**

[**Propósito del sistema**](#_f2sohfcu7gkg) **4**

[**Definiciones**](#_4d8omjua5m7b) **4**

[**Arquitectura general del sistema**](#_oz2pgjl1tm8e) **5**

[Diagrama de contexto](#_m0rsquw9s66z) 5

[(Diagrama de contexto realizado en Draw.io)](#_jzdnzhlsmixc) 6

[Especificación de requerimientos](#_vpy5htc410kh) 6

[Requerimientos no Funcionales](#_5ifg7yxcb17k) 7

[Arquitectura del sistema](#_fkbqlurbd5ta) 9

[Tecnologías aplicadas](#_z9uaat3vilro) 9

[Servidor AWS:](#_4p5hcckpkje6) 9

[API Firebase](#_ry8e0vpu3o6m) 10

[Java](#_4hbur8bvestm) 10

[Vue JS](#_nekjx7oelyut) 10

[PostgreSQL](#_7u9n1tsow0m0) 10

[**Evaluación de la arquitectura**](#_uk86jenhm4c0) **10**

[Atributos de calidad identificados](#_hms7dwneqk8j) 10

[Tácticas a implementar](#_hms7dwneqk8j) 12

# **Descripción general**

El sistema a realizar será una plataforma web y mobile que permitirá brindar facilidades a dos tipos de usuarios: compradores finales y tiendas, para poder adquirir productos y visualizarlos en una plataforma online.

Las tiendas podrán registrar sus productos en la plataforma con todos sus detalles de compra y estos productos serán encontrados de forma interactiva por medio de una aplicación móvil que realizar reconocimiento de imágenes y te recomienda los productos similares a la foto que se envió.

# **Propósito del sistema**

Lo que se busca con el sistema es dar solución a la poca visibilización de productos que tienen las empresas medianas o pequeñas, desde bazares, tiendas tecnológicas, ropas, entre otros.  
El sistema permitirá una rápida búsqueda y recomendación de diversos productos relacionados separados por categorías, aumentando así el posicionamiento online de estas tiendas y por ende sus ventas.

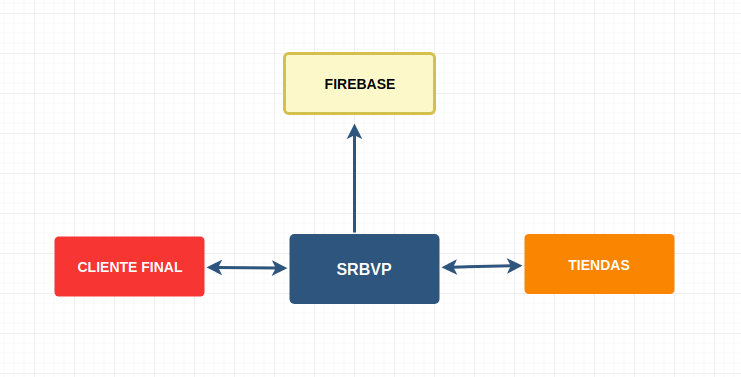
# **Definiciones**

|  |  |
| --- | --- |
| Término | Definición |
| Arquitectura de Software | Conjunto de elementos estáticos, propios del diseño intelectual del sistema, que definen y dan forma tanto al código fuente, como al comportamiento del software en tiempo de ejecución. Naturalmente este diseño arquitectónico ha de ajustarse a las necesidades y requisitos del proyecto. |
| Actor | Usuario del sistema que puede participar de un caso de uso. |
| Paquetes | Agrupaciones de casos de uso y actores por funcionalidad que proveen. |
| HTTP | El Protocolo de transferencia de hipertexto (en inglés: Hypertext Transfer Protocol o HTTP) es el protocolo de comunicación que permite las transferencias de información en la World Wide Web. |
| UML | Unified Modeling Language (Lenguaje de Modelado Unificado) |
| Servidor | Server: Un servidor es una aplicación en ejecución (*software*) capaz de atender las peticiones de un cliente y devolverle una respuesta en concordancia |
| Módulo | En informática un módulo es una parte autónoma de un programa de ordenador. |
| PostgreSQL | Es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual. |
| Controlador JDBC | Es un componente de software que les permite a las aplicaciones acceder a una base de datos |
| Amazon Web Services | Es una plataforma en la nube que ofrece más de 165 servicios integrales de centros de datos a nivel global. |
| M.V.C | Patrón de diseño Modelo - Vista - Controlador Sirve para separar en capas el diseño de la lógica en el Service Web y Web. |
| M.V.P | Patrón de diseño Modelo - Vista - Presentador Sirve para separar en capas el diseño de la lógica en el aplicativo móvil. |
| Api RESTFUL | Enlaces de comunicación entre el aplicación y la web con los datos en la base de datos. |

# **Arquitectura general del sistema**

## Diagrama de contexto

El diagrama de contexto del sistema SRBVP, podemos observar que la aplicación interactúa, con dos tipos de usuarios, cliente final y cliente tiendas; a su vez interactúa con firebase como plataforma de notificaciones.



## (Diagrama de contexto realizado en Draw.io)

## 

## Especificación de requerimientos

* Requerimientos Funcionales

## 

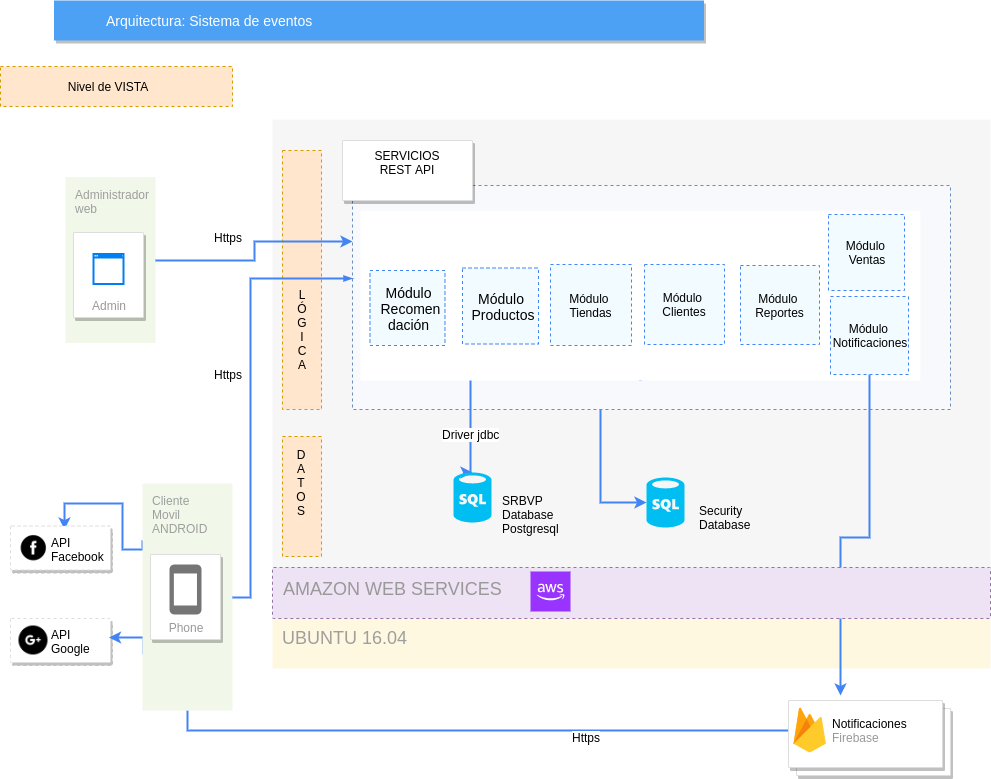
|  |  |
| --- | --- |
| **Requerimiento funcional** | **Descripción** |
| RF - 001 | El sistema debe poder registrar dos tipos de usuario, las tiendas que se afilien al sistema y los clientes que buscan productos |
| RF - 002 | La aplicaciónmóvil debe tener acceso a la cámara del dispositivo |
| RF - 003 | El usuario “cliente” debe poder buscar un producto en las tiendas afiliadas, así como obtener la información del mismo |
| RF - 004 | El usuario “cliente” debe poder generar un ticket de compra mediante la aplicación móvil |
| RF - 005 | El usuario “tienda” debe poder registrar los datos de su tienda mediante la página web del sistema. |
| RF - 006 | Los usuarios “tienda” deben poder registrar sus productos, el stock de cada producto, el precio y las características del producto. |
| RF - 007 | La aplicación móvil debe poder listar productos que tengan relación con el producto que el usuario “cliente” fotografió. |
| RF - 008 | El usuario “tienda” deberá poder editar los datos generales de sus productos. |
| RF-009 | Los usuarios deberán poder dar de baja a su cuenta. |
| RF-010 | Los usuarios podrán cambiar su contraseña |

## 

## Requerimientos no Funcionales

|  |  |
| --- | --- |
| **Requerimiento no funcional** | **Descripción** |
| RNF - 001 | USABILIDAD  Los errores mostrados por el sistema deben ser legibles al usuario. |
| RNF-002 | USABILIDAD  El sistema estará dirigido a usuarios con conocimientos básicos en el uso de tecnologías de la información. |
| RNF-003 | USABILIDAD  Se seguirán los estándares de diseño, para las interfaces del sistema. |
| RNF-004 | USABILIDAD  El sistema contará con una interfaz amigable con el usuario. |
| RNF – 005 | CONFIABILIDAD  El sistema validará todo tipo de información que se le ingrese al mismo, en los distintos formularios que éste dispone. |
| RNF - 006 | CONFIABILIDAD  El sistema debe ser tolerante ante los fallos y las operaciones a realizar, deben ser transaccionales. |
| RNF – 007 | CONFIABILIDAD  La información almacenada estará protegida por distintos procesos de seguridad |
| RNF – 008 | SEGURIDAD  La información no estará permitida para usuarios que no tengan los permisos establecidos; además el sistema tendrá procedimientos para validación de usuarios. |
| RNF – 009 | RESPUESTA  El tiempo de respuesta del sistema dependerá de la cantidad de transacciones que requerirá una operación para realizar un requerimiento dado o en todo caso será no mayor de 4 segundos. |
| RNF – 010 | RESPUESTA  El sistema estará disponible las 24 horas del día. |
| RNF – 011 | RESPUESTA  El sistema dispondrá de mecanismos de detección de errores y de copias de seguridad. |

## Arquitectura del sistema



## Tecnologías aplicadas

### **Servidor AWS:**

Amazon Web Services, es una nube pública, que nos permitirá desplegar el sistema en internet, aportandonos la seguridad, la estabilidad, el manejo y control de la aplicación.

* + 1. **Amazon RDS:**   
       Es sencillo de configurar, utilizar y escalar una base de datos relacional en la nube. El servicio suministra capacidad rentable y escalable al mismo tiempo que automatiza las arduas tareas administrativas, como el aprovisionamiento de hardware, la configuración de bases de datos, la implementación de parches y la creación de copias de seguridad.
    2. **Amazon Backup:**   
       AWS Backup es un servicio de copias de seguridad completamente administrado que facilita la tarea de centralizar y automatizar el respaldo de los datos en los servicios de AWS en la nube y en las instalaciones mediante AWS Storage Gateway.

### **API Firebase**

Servicio de Google que nos permite la administración de los mensajes Push hacía los dispositivos móviles, está basado en una estructura de colas

### **Java**

Para el desarrollo mobile se utilizará el lenguaje java sobre Android Studio, ya que nos permite un desarrollo más rápido de un producto funcional y eficiente para la gestión del evento y el manejo de notificaciones.

### **Vue JS**

Es un framework progresivo para construir interfaces de usuario. A diferencia de otros *frameworks* monolíticos, Vue está diseñado desde el inicio para ser adoptado incrementalmente. La biblioteca principal se enfoca solo en la capa de la vista, y es muy simple de utilizar e integrar con otros proyectos o bibliotecas existentes. Por otro lado, Vue también es perfectamente capaz de soportar aplicaciones sofisticadas de una sola página (en inglés *single-page-application* o SPA) cuando se utiliza en combinación con [herramientas modernas](https://es-vuejs.github.io/vuejs.org/v2/guide/single-file-components.html) y [librerías compatibles](https://github.com/vuejs/awesome-vue#libraries--plugins).

### **PostgreSQL**

Gestor de Base de datos.  
Postgresql nos permite realizar copias de seguridad de la base de datos en caliente

# **Evaluación de la arquitectura**

## Atributos de calidad identificados

1. **Disponibilidad**: El tiempo de inactividad debe ser mínimo ante una falla.
2. **Performance**: El sistema debe debe dar respuesta rápida e inmediata ante fallos que se puedan presentar.Además; La latencia del sistema debe ser lo mínimo posible para que el usuario no perciba a la solución como lenta o colgada.
3. **Seguridad**: El sistema debe validar el tipo de usuario y darle permisos correspondientes. Además; El sistema no debe permitir ingresar sin una autenticación correcta.
4. **Usabilidad**:El sistema debe ser intuitivo y amigable, facilitando su uso a nuevos usuarios.

**Escenarios**

|  |  |
| --- | --- |
| **Escenario de disponibilidad** | |
| **Origen del Estímulo** | Usuario del sistema |
| **Estímulo** | Solicitud de cualquier usuario válido para el sistema. |
| **Artefacto** | El sistema |
| **Entorno** | El sistema en condiciones normales |
| **Respuesta** | El sistema atiende la solicitud satisfactoriamente y genera una respuesta al usuario. |
| **Medida de respuesta** | el sistema debe responder el 100% de solicitudes |

|  |  |
| --- | --- |
| **Escenario de Seguridad** | |
| **Origen del Estímulo** | Usuario intentando acceder al sistema |
| **Estímulo** | Ingresar datos de ingreso |
| **Artefacto** | El sistema |
| **Entorno** | El sistema en condiciones normales |
| **Respuesta** | El sistema valida los datos enviados y autoriza si estos son válidos |
| **Medida de respuesta** | el sistema debe negar el 100% de solicitudes con las credenciales erradas. |

|  |  |
| --- | --- |
| **Escenario de Usabilidad** | |
| **Origen del Estímulo** | Usuario del sistema |
| **Estímulo** | usar el sistema |
| **Artefacto** | El sistema |
| **Entorno** | sistema finalizado y operativo |
| **Respuesta** | El sistema presentará un estructura simple y fácil de utilizar |
| **Medida de respuesta** | El usuario debe aprender a utilizar el sistema en menos de 5 minutos. |

|  |  |
| --- | --- |
| **Escenario de Performance** | |
| **Fuente** | Usuario |
| **Estímulo** | Usar el sistema |
| **Artefacto** | Sistema |
| **Entorno** | el sistema finalizado y operativo |
| **Respuesta** | El sistema debe dar respuesta al usuario de manera rápida |
| **Medición** | La respuesta del sistema debe demorar menos de 2 segundos |

## Tácticas a implementar

1. **Disponibilidad**:

Se implementará un control de errores y manejo de excepciones para evitar las caídas del sistema.

Contamos con una base de datos de respaldo de la cual se puede obtener la información en caso la base de datos principal presenta fallos.  
Al tener los servicios en la nube aseguramos aún más la alta disponibilidad gracias a los servicios de Amazon.

1. **Performance**

Se considera, para los flujos con transacciones, mejorar los algoritmos y así tener recursos más eficientes.  
En las consultas se maneja un time out, para medir el tiempo de respuesta.  
Las consultas pesadas de las fotos se realizan en la plataforma de google para evitar sobrecargar nuestros servidores.

1. **Seguridad**

Utilizar la identificación por usuario y contraseña; se trabajará con tokens de autenticación para evitar el acceso de usuarios maliciosos. Este permitirá la revocación del acceso en caso se detecte una intrusión.

1. **Usabilidad**

Se realizarán interfaces de usuario amigables e interactivas para que el cliente tenga la facilidad de usar tanto aplicación móvil como la web y su experiencia sea agradable.