

Desarrollo de un Software Web para el control de inventario, con aplicación móvil de consulta

Kelvin Arrobo¹, Erica Vásquez², Liliana Enciso³

³Dpto. de Ciencias de la Computación y Electrónica,
Universidad Técnica Particular de Loja
Loja, Ecuador

¹krarrobo1@utpl.edu.ec, ²etvasquez@utpl.edu.ec, ³lenciso@utpl.edu.ec

Abstract—El presente trabajo propone el desarrollo de una aplicación móvil que ayudará a los proveedores a mantener un control de su inventario de una forma segura en cualquier momento, en la cual, el beneficiado podrá observar cuantos productos han sido vendidos, cual es el estado de sus productos, además recibirá un mensaje de alerta cuando uno de sus productos haya caducado. Esta información será consumida desde un portal de compras al cual tiene acceso el cliente para realizar sus compras y el proveedor en caso que desee agregar, modificar o eliminar productos de su inventario.

Keywords—REST, web services, API.

Abstract—This work proposes the development of a mobile application that will help suppliers to keep control of their inventory in a safe way at any time, in which the beneficiary will be able to observe how many products have been sold, what is the status of their products, you will also receive an alert message when one of your products has expired. This information will be consumed from a shopping portal to which the customer has access to make their purchases and the supplier in case you want to add, modify or eliminate products from your inventory.

Keywords—REST, web services, API.

I. INTRODUCCIÓN

Actualmente, la globalización mundial y el gran nivel competitivo ha provocado que muchas empresas proveedoras de productos busquen nuevas maneras de llevar un control preciso de su inventario, con la finalidad de proveer a sus clientes un mejor servicio y alcanzar una etapa de prosperidad económica y estabilidad de los miembros que la integran dirigiéndose principalmente a tomar decisiones orientadas a la reducción de costos, mejoramiento de la calidad y agilidad en los procesos para alcanzar los máximos resultados económicos. [1].

Los negocios buscan detectar y responder con rapidez a la demanda cambiante de los clientes, reducir inventarios a los niveles más bajos posibles y lograr posiciones más altas de eficiencia operacional. Las cadenas de suministro se han vuelto más aceleradas, en donde empresas de todos los tamaños dependen del inventario justo a tiempo para reducir sus costos indirectos y llegar más rápido al mercado. La base de toda empresa comercial es la compra productos; de aquí la importancia de suministrar los productos adecuadamente por parte del proveedor [2].

A través de un sistema de control de inventario permanente de mercadería el proveedor podrá conocer a ciencia cierta

la rotación de cada uno de los productos, saber cuáles con los más solicitados y en base a qué criterios solicitar una nueva provisión de tal manera que pueda extraer estrategias específicas para aprovechar dicha situación; igualmente indicara los productos que menos rotación tienen de tal manera que se pueda establecer tácticas para que su demanda se incremente o sencillamente que la empresa deje de invertir en ese tipo de artículos ya que no son muy negociables [3]. El comercio es una actividad que necesita ser manejada por personas que actúen con inteligencia, aplicando la creatividad y las habilidades posibles para lograr resultados que justifiquen su accionar [4].

Sin un sistema de inventarios que actúe con total fiabilidad, es posible que las ventas no sean las estimadas, puesto que no contaremos con la mercancía necesaria para satisfacer las necesidades del cliente. Con el sistema de inventario el proveedor evitará la demora en el despacho de la mercancía y se garantiza el buen estado de la misma, puesto que esta va a tener una rotación adecuada. De este modo podemos llevar a cabo una planeación de reabastecimiento adecuada [5].

II. PROYECTOS RELACIONADOS

Existen un sinnúmero de aplicaciones que ayudan a las empresas a mantener un control de su inventario, pero ninguna de estas está dirigida a solventar las necesidades de los proveedores, entre las aplicaciones investigadas tenemos las siguientes. La implementación de aplicación web con acceso a base de datos para manejo de inventario de la empresa Orange Business Services Colombia S.A. En este trabajo se describe un proceso completamente distinto, se propone tener la base de datos en un computador, que permita compartir la información de la misma, sin la necesidad de volver pública la base de datos. Asimismo, en este trabajo no se hace referencia al desarrollo web, más bien, al uso de web services, esto diferenciará totalmente el desarrollo del proyecto, pues en el trabajo de la aplicación web se realiza la conexión de manera directa a la base de datos por el uso de la página, en el trabajo propuesto se implementarán los servicios web para acceder a la información, protegiendo de mejor manera la información de la base de datos y a su vez, proponiendo un reto a los desarrolladores del proyecto, que es implementar

distintos métodos de conexión con una base de datos, como lo son los drivers de MySQL, para poder comunicarse con los programas (de naturaleza de Java) que deseen ver la información [2]. El diseño de una aplicación Android, que recolecta información de tráfico y su comunicación a un servidor. En este proyecto se desarrolla una herramienta que es una aplicación móvil para los dispositivos Android. La App posee la capacidad de establecer un enlace cliente – servidor, en la red se permite compartir una base de datos en MySQL, esta base de datos se almacena en un servidor. De manera que este antecedente aporta información en cuanto al desarrollo de la aplicación Android y manejo de MySQL [6]. Otro proyecto destacado trata sobre el desarrollo de una aplicación móvil Android para control remoto de un servicio web. El objetivo de esta herramienta consiste en el desarrollo de una aplicación para el sistema operativo Android, que permita obtener una interfaz de usuario conectada a un servicio web. Donde resulta un vínculo con el proyecto, en vista al desarrollo e información de la aplicación Android. La Aplicación Android para la empresa Travelling-Service. Esta fue una herramienta desarrollada con la finalidad de asociar una página web de la 7 empresa Travelling Service mediante el uso de servicios de telecomunicaciones y bases de datos. De ahí se puede referenciar unos parámetros básicos para tener en cuenta en el momento de diseñar una herramienta móvil para una empresa [7]. En base a las proyectos previamente revisados y a la necesidad que se intenta solventar, se plantea una propuesta que permita aplicar el problema encontrado en una aplicación móvil desarrollada en Android, misma que, ayudará a los proveedores a estar al día sobre el estado en el cual se encuentran las ventas efectuadas de sus productos de una forma rápida y cómoda desde su teléfono y cualquier lugar, a través del portal web de compras desde la cual se va consumiendo los datos actualizados cada vez que se registre una compra del producto ofertado en la empresa de venta.

III. MARCO TEÓRICO

A. Servicios Web

El consorcio W3C define los Servicios Web como sistemas software diseñados para soportar una interacción interoperable máquina a máquina sobre una red. Los Servicios Web suelen ser APIs Web que pueden ser accedidas dentro de una red (principalmente Internet) y son ejecutados en el sistema que los aloja [8]. Los Servicios Web albergan muchos tipos diferentes de sistemas, pero el caso común de uso de refiere a clientes y servidores que se comunican mediante mensajes XML que siguen el estándar SOAP.

En los últimos años se ha popularizado un estilo de arquitectura Software conocido como REST (Transferencia de estado representacional). Este nuevo estilo ha supuesto una nueva opción de estilo de uso de los Servicios Web. A continuación, se listan los tres estilos de usos más comunes [9]:

- **Remote Procedure Calls (RPC, Llamadas a Procedimientos Remotos):** Los Servicios Web basados en RPC presentan una interfaz de llamada a procedimientos y

funciones distribuidas, lo cual es familiar a muchos desarrolladores. Típicamente, la unidad básica de este tipo de servicios es la operación WSDL (WSDL es un descriptor del Servicio Web, es decir, el homólogo del IDL para COM).

- **Arquitectura orientada a servicios (Service-oriented Architecture, SOA).** Los Servicios Web pueden también ser implementados siguiendo los conceptos de la arquitectura SOA, donde la unidad básica de comunicación es el mensaje, más que la operación. Esto es típicamente referenciado como servicios orientados a mensajes.
- **REST (Representación de transferencia de estado).** Los Servicios Web basados en REST intentan emular al protocolo HTTP o protocolos similares mediante la restricción de establecer la interfaz a un conjunto conocido de operaciones estándar. Por tanto, este estilo se centra más en interactuar con recursos con estado, que con mensajes y operaciones.

B. REST

Es un estilo de arquitectura basado en un conjunto de principios que describe cómo se definen y se abordan los recursos en red. Los servicios RESTful ofrecen una alternativa simple, liviana y escalable a los servicios basados en SOAP. REST utiliza los métodos de interacción remotos HTTP integrados básicos (PUT, POST, GET, y DELETE) aplicando su semántica prevista para acceder a cualquier recurso referenciable URI [10]. En la Figure 1, se puede observar una interacción de REST para servicios web, en la cual existen solicitudes de servicio desde el servicio cliente al servidor. Un recurso podría ser cualquier dato en la Web, como un documento, una imagen, un tweet o un pronóstico del tiempo. Además, los servicios RESTful exhiben cuatro propiedades [11]:

- 1) Los recursos representan una abstracción para el estado y las entidades de la aplicación del servidor, es decir, cualquier elemento que pueda ser el objetivo de una referencia de hipertexto es un recurso.
- 2) Cada recurso es direccionable utilizando un identificador mundial único (URI).
- 3) Todos los recursos comparten una interfaz uniforme (métodos HTTP) para interactuar con las aplicaciones cliente.
- 4) La interacción con un recurso es sin estado.



Figure 1. Uso de REST para servicios web.

C. Hypertext Transfer Protocol

La mayoría de las aplicaciones incluyen funcionalidades HTTP, el protocolo seguro de transferencia de hipertexto, este es un protocolo de comunicaciones utilizado para la comunicación segura. HTTPS se usa frecuentemente para mensajes basados en XML, pero también se puede usar para archivos binarios o sin formato. Cuando se emplea el protocolo HTTPS, se puede observar el icono de un candado en la barra del navegador que se esté usando, este protocolo de seguridad empezó a ser usado por distintas entidades que trabajan con información personal e intransferible. Si en una página Web se usa HTTPS, esta página se protege por medio de una codificación con certificado digital SSL, creando así un canal más seguro para el transporte de datos del usuario o cliente [12]. Los Web Services basados en REST son considerados como Web services, la organización de Netbeans considera que "... REST es una nueva forma de crear y comunicarse con los servicios web. En REST, los recursos tienen URIs y se manipulan a través de operaciones de encabezado HTTP." [13].

1) *Método de autenticación básica*: El protocolo HTTP proporciona un mecanismo de autenticación desafío-respuesta que puede ser usado por un servidor para desafiar una solicitud de un cliente para proveer información de autenticación. El esquema de autenticación "básico" se basa en el modelo que el cliente debe autenticarse con un ID de usuario y una contraseña para cada realm. El valor de realm debe considerarse una cadena opaca que solo se puede comparar por la igualdad con otros realm en ese servidor. El servidor atenderá la solicitud solo si puede validar el ID de usuario y la contraseña para el espacio de protección del URI de solicitud. No hay parámetros de autenticación opcionales [11].

2) *Métodos de solicitud*: El token de método de solicitud es la fuente principal de la semántica de solicitud; indica el propósito para el cual el cliente ha realizado esta solicitud y lo que el cliente espera como resultado exitoso. Dicho token distingue entre mayúsculas y minúsculas y se utiliza como puerta de acceso a sistemas basados en objetos [14]. Esta especificación define una serie de métodos estandarizados que son comúnmente utilizados en HTTP, como se describe en la siguiente Figure 2. Por convención, los métodos estandarizados se definen en letras mayúsculas (US-ASCII).

Método	Descripción
GET	Transfiere la representación actual de la fuente objetivo
HEAD	Al igual que GET, pero solo transfiere la línea de estado y la sección de encabezado
POST	Realiza el procesamiento específico de recursos en la carga útil de la solicitud
PUT	Reemplaza todas las representaciones actuales del recurso objetivo con la carga útil de la solicitud
DELETE	Elimina todas las representaciones actuales del recurso objetivo
CONNECT	Establece un túnel al revidor identificado por el recurso objetivo
OPTIONS	Describe las opciones de comunicación para el recurso objetivo
TRACE	Realiza una prueba de bucle de retorno de mensaje a lo largo de la ruta al recurso de destino.

Figure 2. Métodos de solicitud HTTP.

D. Plataformas de Desarrollo

1) *Java*: Java es un lenguaje de programación creado por Sun Microsystems, (empresa que posteriormente fue comprada por Oracle) para poder funcionar en distintos tipos de procesadores. Su sintaxis es muy parecida a la de C o C++, e incorpora como propias algunas características que en otros lenguajes son extensiones: gestión de hilos, ejecución remota, etc. El código Java, una vez compilado, puede llevarse sin modificación alguna sobre cualquier máquina, y ejecutarlo. Esto se debe a que el código se ejecuta sobre una máquina hipotética o virtual, la Java Virtual Machine, que se encarga de interpretar el código (archivos compilados .class) y convertirlo a código particular de la CPU que se esté utilizando [13].

2) *Restful*: Se trata de una API que permite manejar servicios web con métodos definidos, manteniendo la simpleza del protocolo como XML, para que cada servicio sea identificado únicamente con un solo URI. NetBeans proporciona un asistente que puede generar automáticamente el código Java del cliente que invoca nuestros métodos de servicio web RESTful a través de las solicitudes HTTP correspondientes [15].

3) *Android Studio*: Es el entorno desarrollado por la empresa Google, Android Studio fue anunciado por primera vez en el año 2013 en la conferencia de Google I/O, como el primer IDE estable para el desarrollo de aplicaciones móviles. Android Studio es el IDE oficial para desarrollo de aplicaciones Android [13]. Asimismo, "Android Studio proporciona las herramientas más rápidas para la creación de aplicaciones en todos los tipos de dispositivos Android." Android Studio permite desarrollar para todos los dispositivos Android, desde los Smartwatch hasta televisores inteligentes con el SO Android [16].

4) *Retrofit*: Es una biblioteca de redes desarrollada por Square a través de la cual podemos capturar JSON sin problemas. Ahora no necesitamos ningún analizador JSON para analizar los datos. Retrofit literalmente compuesto de todas las características requeridas para servicios web. No necesita ninguna AsyncTask, HttpURLConnection o JsonParser [17].

5) *GlassFish Server*: Es un servidor de aplicaciones desarrollado por Sun Microsystems que implementa las tecnologías definidas en la plataforma Java EE/Web y permite ejecutar aplicaciones que siguen esta especificación. La versión comercial es denominada Sun GlassFish Enterprise Server. Soporta las últimas versiones de tecnologías como: JSP, Servlets, EJBs, Java API para Servicios Web (JAX-WS), Arquitectura Java para Enlaces XML (JAXB), Metadatos de Servicios Web para la Plataforma Java 1.0, y muchas otras tecnologías [18].

E. Metodologías de desarrollo de software

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo [19]. Las metodologías ágiles están mostrando su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo, pero manteniendo una alta calidad [20]. En la Figure 3, se puede observar

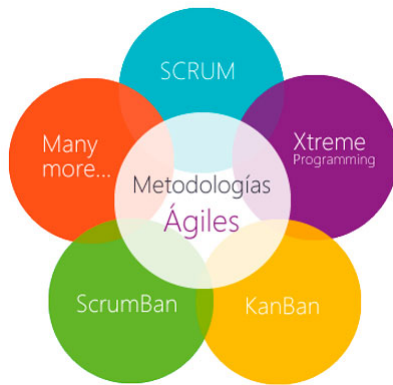


Figure 3. Metodologías Ágiles de Desarrollo de Software..

los diferentes tipos de metodologías ágiles existentes. Entre ellas tenemos programación extrema, Kanban, ScrumBan, Lean, entre otras. La metodología SCRUM define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente [21]. Scrum se centra en la división del trabajo complete en distintos apartados o bloques que pueden ser abordados en periodos cortos de tiempo los cuales son denominados Sprint. Los pilares o características de la metodología Scrum más importantes son [22]:

- **Transparencia:** Todos los implicados tienen conocimiento de qué ocurre y en el proyecto y cómo ocurre. Esto hace que haya un entendimiento “común” del proyecto, una visión global.
- **Inspección:** Los miembros del equipo Scrum frecuentemente inspeccionan el progreso para detectar posibles problemas. La inspección no es un examen diario, sino una forma de saber que el trabajo fluye y que el equipo funciona de manera auto-organizada.
- **Adaptación:** Cuando hay algo que cambiar, el equipo se ajusta para conseguir el objetivo del sprint. Esta es la clave para conseguir éxito en proyectos complejos, donde los requisitos son cambiantes o poco definidos y en donde la adaptación, la innovación, la complejidad y flexibilidad son fundamentales.

IV. METODOLOGÍA

A. Diseño

El desarrollo del proyecto se divide en dos partes, la primera, referente a la creación de una plataforma web, desde la cual los clientes podrán realizar la compra de productos que se encuentran disponibles, adicional a ello, los proveedores podrán ingresar a la plataforma web con sus credenciales, para poder administrar su inventario (introducir, modificar o eliminar algún producto). La segunda parte, hace mención a

la elaboración de una aplicación móvil, que será utilizada únicamente por el proveedor, en la cual podrá observar la productividad de las venta de sus productos en la plataforma.

A continuación, se presentan los diseños de casos de uso planteados y la Arquitectura sobre la cual funcionará el aplicativo a desarrollar.

1) *Modelos de Casos de Uso:* Los modelos de caso de uso son de gran utilidad para ilustrar la interacción que tendrán los usuarios (en este caso cliente, proveedor) con el sistema a implementar. En la siguiente Figure 4, se puede constatar, el modelo de caso de uso resultante de las actividades que se pueden realizar mediante la plataforma web. Desde la cual el actor cliente tendrá la capacidad de visualizar el listado de productos, añadir productos a su orden de compra para posteriormente finalizar con el pago de la orden creada. Así mismo del lado del proveedor quien tendrá las tareas de ingresar, modificar y eliminar productos. El modelo de caso

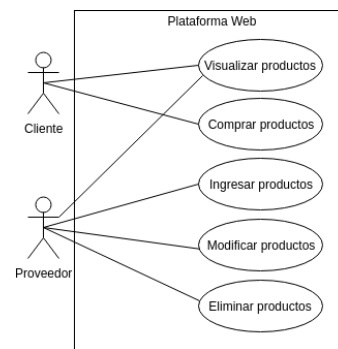


Figure 4. Modelo caso de uso, referente a la Plataforma Web.

de uso de la aplicación móvil, se muestra a continuación, en la Figure 5, en la cual se encuentran las operaciones que podrá realizar el proveedor dentro del aplicativo, las cuales son ver el estado en que se encuentran los productos (disponibles, agotados, caducados) y ver las ganancias que generan las ventas efectuadas por los clientes desde el portal web.

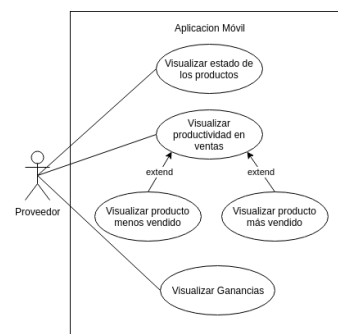


Figure 5. Modelo caso de uso, referente a la Aplicación móvil.

2) *Arquitectura:* El modelo Cliente/Servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Las aplicaciones Clientes realizan peticiones a una o varias aplicaciones Servidores, que

deben encontrarse en ejecución para atender dichas demandas diferentes.

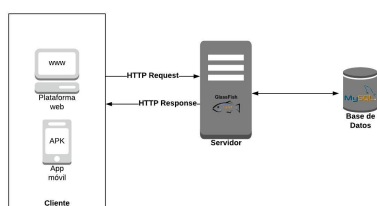


Figure 6. Arquitectura del proyecto.

En el caso de nuestro aplicativo definiremos la plataforma web y la aplicación móvil como clientes. Los cuales realizan las peticiones al servidor local (en este caso GlassFish), el cual se encargará de responder dichas peticiones y enviar su respuesta a los clientes mediante el protocolo HTTP. Así mismo se encargada de gestionar las conexiones con la base de datos MySQL la misma que almacenará todos los datos emitidos por los clientes.

B. Implementación

En lo que respecta al desarrollo de la plataforma web en Java, tenemos distribuido el proyecto mediante una arquitectura Modelo Vista Controlador, de manera en que se pueda llevar a cabo un proyecto con funcionalidades distribuidas entre los componentes de la arquitectura.

Por un lado, tenemos el componente Modelo Figure 7 definimos los objetos Proveedor, Producto, Categoría, Estado, Pedido de manera en que se pueda tener una estructura sólida de tipado fuerte, que caracteriza a Java como lenguaje orientado a objetos, con los cuales trabajaremos en los demás componentes en cada instancia.

```

1 package modelo;
2
3 public class Producto {
4     private int idproducto;
5     private String nombre;
6     private double preciounitario;
7     private int cantidadStock;
8     private String fechaemision;
9     private String fechaexpedicion;
10    private Categoria idcategoria;
11    private EstadoProducto idestado;
12    public Producto() {
13    }
14
15
16
17    public Producto(int idproducto, String nombre, double preciounitario, int cantidadStock, String
18
19
20
21
22    public Producto(String nombre, double preciounitario, int cantidadStock, String fechaemision,
23        String fechaexpedicion, Categoria idcategoria, EstadoProducto idestado) {... lines }
24
25
26
27    public int getIdproducto() {... lines }
28
29
30    public void setIdproducto(int idproducto) {... lines }
31
32
33    public String getNombre() {... lines }
34
35
36    public void setNombre(String nombre) {... lines }
37
38    public double getPreciounitario() {... lines }

```

Figure 7. Clase modelo producto.

Así mismo tenemos el componente Vista en el cual está contemplado toda la parte visual de nuestra aplicación, es decir las interfaces con las que el usuario interactuara a cada momento, para diseñar las vistas usamos lo que se conocen

como JSP (JavaServer Pages) las cuales son de ayuda para construir plataformas web dinámicas basadas en HTML, XML entre otros tipos de documentos.

[illegible]

Figure 8. JSP Index.

Ahora bien, el componente controlador el cual fue desarrollado con Servlets encargados de manejar los datos que serán mostrados en la Vista, dicho componente actúa como una interfaz entre el Modelo y la Vista, si recibe peticiones de la vista se encarga de procesarlas y validarlas, así mismo envía resultados de las peticiones hacia la Base de datos y hacia la misma vista. En resumen, se encarga del backend de la plataforma web en sí.

```

81 //Hacemos clic en el boton de editar para ir al metodo de actualizar
82 private void cargarProducto(HttpServletRequest request, HttpServletResponse response) throws Exception {
83     // Lee id del producto
84     int idProveedor = Integer.parseInt(request.getParameter("idProveedor"));
85     //Leer id proveedor
86     int idProveedor = Integer.parseInt(request.getParameter("idProveedor"));
87     // Leer id producto
88     int idProducto = Integer.parseInt(request.getParameter("idProducto"));
89     // Consultar datos del producto
90     Producto objProducto = controlProducto ObtenerInfoProducto(idProducto);
91     // Retornar la respuesta
92     List<EstadoProducto> estados = controlProducto.PresentarEstados();
93     List<Categoria> categorias = controlProducto.PresentarCategorias();
94     request.setAttribute("producto_actualizar", objProducto);
95     request.setAttribute("listaProductos", categorias);
96     request.setAttribute("listaEstados", estados);
97     request.setAttribute("idProveedor", idProveedor);
98     RequestDispatcher dispatcher = request.getRequestDispatcher("/actualizarProducto.jsp");
99     dispatcher.forward(request, response);
100 }
101
102 private void actualizarProducto(HttpServletRequest request, HttpServletResponse response) throws Exception {
103     int idProveedor = Integer.parseInt(request.getParameter("idProveedor"));
104     int idProducto = Integer.parseInt(request.getParameter("id_producto"));
105     String nombre = request.getParameter("nombre");
106     double precio = Double.parseDouble(request.getParameter("precio"));
107     int cantidad = Integer.parseInt(request.getParameter("cantidad"));
108     String fecha = request.getParameter("fecha");
109     String estado = request.getParameter("estado");
110     String categoria = request.getParameter("categoria");
111     String estado = request.getParameter("estado");
112     int idCategoria = controlProducto.getIdCategoria(categoria);
113     boolean esNuevo = controlProducto.isEstado(estado);
114     // Crear o actualizar objeto
115     Producto obj = new Producto(idProveedor, nombre, precio, cantidad, fecha, estado, idCategoria);
    
```

Figure 9. Servlet Controlador Productos .

La implementación de proyecto, por parte de Android, se dividió en lo que es la configuración de los servicios a ser consumidos. En la figure 10 se muestra el código utilizado para la solicitud requerida por el cliente. En la cual se mida la parte de seguridad mediante el uso de método autenticación básica para recuperar la información.

Una vez obtenida la autenticación, se devuelve un Gson en cual contiene los datos personales del proveedor, mediante los cuales podrá tener acceso a la aplicación. En la Figure 11 se puede observar el método creado para que el usuario ingrese

```

private void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);
    loginButton.setOnClickListener() {
        login()
    }
}

private void login() {
    String email = emailEditText.getText().toString().trim();
    String password = passwordEditText.getText().toString().trim();

    if (email.isEmpty() || password.isEmpty()) {
        Toast.makeText(this, "Por favor, ingrese un correo electrónico y una contraseña.", Toast.LENGTH_SHORT).show();
        return;
    }

    // Aquí se llamaría a un método que verifique los datos de login
    // Por ejemplo: login(email, password)
}

```

Figure 10. Configuración del Servicio Android .

a la plataforma. Por otro lado, la figure 12 indica la vista que será visible por el proveedor al momento del intentar acceder a la plataforma.

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);
    loginButton.setOnClickListener() {
        login()
    }
}

private void login() {
    String email = emailEditText.getText().toString().trim();
    String password = passwordEditText.getText().toString().trim();

    if (email.isEmpty() || password.isEmpty()) {
        Toast.makeText(this, "Por favor, ingrese un correo electrónico y una contraseña.", Toast.LENGTH_SHORT).show();
        return;
    }

    // Aquí se llamaría a un método que verifique los datos de login
    // Por ejemplo: login(email, password)
}

```

Figure 11. Código del Login .

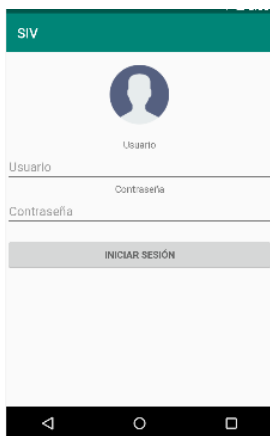


Figure 12. Pantalla Login .

Una vez que el usuario se haya logueado y tenga acceso a la plataforma, se le mostrara el menú principal, ver Figure 13. Mediante el cual podrá navegar y ver la lista de los productos que aún están disponibles, los que ya caducaron y los agotados. Además hay la opción de que pueda ver las ganancias y el producto más y menos vendido.

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);
    loginButton.setOnClickListener() {
        login()
    }
}

private void login() {
    String email = emailEditText.getText().toString().trim();
    String password = passwordEditText.getText().toString().trim();

    if (email.isEmpty() || password.isEmpty()) {
        Toast.makeText(this, "Por favor, ingrese un correo electrónico y una contraseña.", Toast.LENGTH_SHORT).show();
        return;
    }

    // Aquí se llamaría a un método que verifique los datos de login
    // Por ejemplo: login(email, password)
}

```

Figure 13. Código del menú principal .

La figure 14 muestra de forma gráfica el código implementado de la Figure 13, en la cual interactuará el usuario, en los Tabs y Layout incorporados. Cabe recalcar que dicha pantalla se encuentra vacía ya que tiene los fragmentos son llamados según los servicios que solicite el usuario. En cada Fragmentos

se encuentran desplegados todo el resultado de los servicios solicitados.

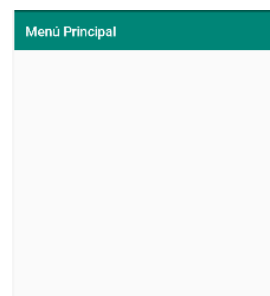


Figure 14. Pantalla menú principal .

V. RESULTADOS

Al finalizar la etapa de desarrollo en Java , tenemos como resultado una plataforma llamada Portal de Compras en la cual el cliente podrá agregar a su canasta uno o varios productos según sus necesidades. Figure 10

#	Nombre Producto	Precio	Stock	Categoría	Proveedor	Stock	Acción
1	Leche de Vaca	12.0	1000	Lacteos	Elva Vaca	1000	Ver Detalles
2	Leche de Cabra	15.0	500	Lacteos	Elva Cabra	500	Ver Detalles
3	Leche de Oveja	18.0	200	Lacteos	Elva Oveja	200	Ver Detalles
4	Leche de Asno	20.0	100	Lacteos	Elva Asno	100	Ver Detalles
5	Leche de Mula	22.0	50	Lacteos	Elva Mula	50	Ver Detalles
6	Leche de Puma	25.0	20	Lacteos	Elva Puma	20	Ver Detalles
7	Leche de Guepardo	28.0	10	Lacteos	Elva Guepardo	10	Ver Detalles
8	Leche de Tigre	30.0	5	Lacteos	Elva Tigre	5	Ver Detalles
9	Leche de León	35.0	2	Lacteos	Elva León	2	Ver Detalles
10	Leche de Oso	40.0	1	Lacteos	Elva Oso	1	Ver Detalles

Figure 15. Portal de Compras.

Posteriormente podrá dirigirse al icono de la canasta para visualizar y confirmar su pedido, así mismo podrá eliminar uno o más productos que ya no desee comprar. Figure 11

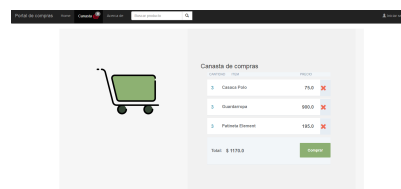


Figure 16. Canasta de compras.

Luego al pulsar el botón "Comprar" se desplegará un formulario Figure 12 en el cual el cliente deberá ingresar su información personal, método de pago, etc.

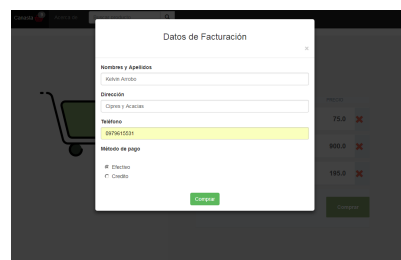
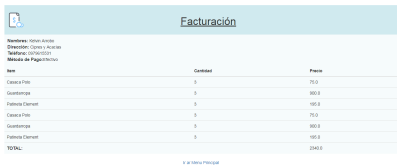


Figure 17. Formulario de facturación.

Finalmente redirigirá al cliente a su respectiva facturación, donde podrá visualizar los productos que ha adquirido y el monto total de su compra, adicionalmente si este lo desea , lo redirecciona a la pantalla principal del portal de compras. Figure 13



Item	Cantidad	Precio
Caca de Pato	5	75.0
Contenedor	5	80.0
Contenedor	5	80.0
Caca de Pato	5	75.0
Contenedor	5	80.0
Contenedor	5	80.0
Total:		290.0

Figure 18. Facturación.

La plataforma dirigida para proveedores, en cambio loguea a los proveedores Figure 14, una vez validadas sus credenciales, los redirige hacia el panel de control Figure 15 de inventario donde podrán ingresar, modificar y eliminar sus productos.

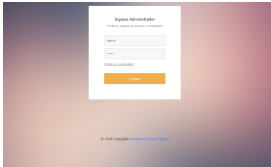


Figure 19. Login para proveedores.



Nombre	Cantidad	Precio	Estado
Yogurt	10	100.0	Disponible
Yogurt	10	100.0	Disponible
Yogurt	10	100.0	Disponible
Yogurt	10	100.0	Disponible
Yogurt	10	100.0	Disponible
Yogurt	10	100.0	Disponible
Yogurt	10	100.0	Disponible
Yogurt	10	100.0	Disponible
Yogurt	10	100.0	Disponible
Yogurt	10	100.0	Disponible

Figure 20. Panel de Control de Inventario.

Los resultados obtenidos en el desarrollo de la aplicación Android se muestran a continuación. En la Figure 21 se puede observar la pantalla de logueo en la cual el usuario ingresa sus credenciales para acceder a la aplicación. Si los datos son correctos se direcciona a al menú principal, Figure 22, en la cual podrá navegar y ver los resultados de sus ventas. En caso de que los datos sean erróneos notificará el error al usuario.

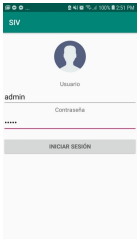


Figure 21. Login Aplicación.

Una vez logueado el usuario, dentro de la aplicación podrá observar enlistados en cada menú los productos y su estado. En el menú ventas, vera las ganancias obtenidas, el producto

más vendido y el menos vendido, para en base a esto poder tomar decisiones y mejorar su negocio mediante técnica de mejora.



Figure 22. Menú Principal de la aplicación.

VI. CONCLUSIONES

Con la realización del presente proyecto, hemos concluido que se pueden mejorar de manera significativa las plataformas de comercio en linea, si se añaden sistemas de gestión de inventario que faciliten la administración de productos, asi mismo se pueden desarrollar aplicativos que permitan medir el desempeño de la tienda en base a las ventas realizadas , de manera en que el proveedor pueda saber las preferencias de sus clientes, asi como también pueda monitorear el estado de sus productos.

REFERENCES

- [1] L. Mindiolaza, “Sistema de Inventario,” 2012.
- [2] A. Carlos, “Diseño y desarrollo de un sistema de gestión de inventarios con integración a SAP,” 2016.
- [3] U. R. Palma, A. Ing, and C. Oquelize, ““ Sistema de control de inventarios del almacén de productos terminados en una empresa metal mecánica ,”” 2009.
- [4] M. Puig, ““Creación de una aplicación, programada en Java, para smart-phones basados en el sistema operativo Android para un portal turístico,”” 2012.
- [5] M. Lozano Perez, “DESARROLLO DE UNA APLICACIÓN MÓVIL ANDROID PARA CONTROL REMOTO DE UN SERVICIO WEB.”
- [6] Á. G. Caballero, “Aplicación Android para obtener información de tráfico en carreteras,” 2016.
- [7] V. Flores and J. Waldo, “Implantación de un aplicativo móvil comercial para incrementar las ventas en una empresa administradora de camposantos,” 2017.
- [8] L. Richardson and S. Ruby, RESTful Web Services Copy Editor: Peggy Wallace Printing History: Cover Designer: Karen Montgomery Interior Designer: David Futato.
- [9] R. N. Marset, “v s W e b S e r v i c e s.”
- [10] T. R. Fielding, “Architectural Styles and the Design of Network-based Software Architectures,” 2000.
- [11] M. Garriga, C. Mateos, A. Flores, A. Cechich, and A. Zunino, “Journal of Network and Computer Applications RESTful service composition at a glance: A survey,” vol. 60, pp. 32–53, 2016.
- [12] Universidad de Chile, “La Web,” 2016.
- [13] J. R. Dean, Introducción al Lenguaje Java. 2009.
- [14] R. Fielding and J. Reschke, “Hypertext Transfer Protocol,” pp. 1–101, 2014.
- [15] J. Sandoval, RESTful Jave. 2009.
- [16] V. Basterra, “Android OS Documentation,” 2017.
- [17] Navneet, “Retrofit Android Tutorial: Example of Retrofit 2 . 0 capturing JSON Array and JSON Object from URL,” pp. 1–9, 2016.
- [18] D. S. Manchado and D. F. Puentes, “Estudio del servidor de aplicaciones Glassfish y de las aplicaciones J2EE,” 2010.
- [19] P. Letelier and C. Penadés, “Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP),” 2004.
- [20] V. Jornadas, “Metodologías Ágiles en el Desarrollo de Software,” 2003.

- [21] I. Innovación, "METODOLOGÍA AGILE Y SCRUM," 2018.
- [22] G. Mousqués, "Metodología SCRUM," 2003.
- [23] I. E. Marini, "El Modelo Cliente / Servidor," pp. 1–11, 2012.