Centro de Electricidad Electrónica Y Telecomunicaciones (CEET)

Análisis y desarrollo en sistemas de información

Ficha:

2049891

Aprendices:

Andrés Felipe Sáenz Salazar

Lady Tatiana Chivito Caicedo

Andrés Esteban Sossa Rodríguez

Yesid Steven Valencia Rodríguez

Instructor:

Helberth Rodrigo Rojas Gacha

Noviembre 27 de 2020, Bogotá D.C

DOCUMENTO DE RESPALDO DE INFORMACIÓN

SOFTWARE DE GESTIÓN Y ORGANIZACIÓN DE INVENTARIOS EN LAS MISCELÁNEAS

Aprendices:

Andrés Felipe Sáenz Salazar

Lady Tatiana Chivito Caicedo

Andrés Esteban Sossa Rodríguez

Yesid Steven Valencia Rodríguez

Ficha:

2049891

Servicio Nacional de Aprendizaje Sena

Centro de Electricidad Electrónica Y Telecomunicaciones (CEET)

Programa Análisis y desarrollo en sistemas de información

CONTENIDO

TABLA DE DIAGRAMAS

ALCANCE

Este documento pretende dar conocer los tipos y etapas de pruebas de software que se van a desarrollar y a poner en marcha durante el desarrollo y ejecución del sistema de información, de igual forma pretende entregar las pautas y estrategias que se seguirán para llevar a cabo con la certificación de software que establecerá que las pruebas se realizaron adecuadamente y en el momento oportuno.

El objetivo general de este documento es establecer la cronología y condiciones para la aplicación de pruebas, de esta forma obtener, un sistema que pueda ser completo y que cumpla con las expectativas del cliente final. Además de esto poner el mismo en operación mostrando todas sus funcionalidades. Ya que va dirigido al cliente final y este espera que el sistema esté funcionando a la perfección con todos los requisitos que el solicito al comienzo del desarrollo del sistema.

DEFINICIÓN DE NIVELES DE PRUEBA

Las pruebas que se realizaran durante el desarrollo y ejecución del sistema de información serán aquellas que ayuden a encontrar problemas y solucionarlos en tiempo oportuno, Además de esto ayudan a comprobar si los resultados obtenidos son iguales que los resultados esperados y así garantizar de forma segura y eficaz que el sistema está libre de errores y defectos. Las pruebas de software que se usaran para el sistema de información **EFFECTIVE RECORD** han sido previamente investigadas y probadas en el mismo sistema.

PRUEBA UNITARIA O DE COMPONENETES

Este tipo de pruebas son ejecutadas normalmente por el equipo de desarrollo, básicamente consisten en la ejecución de actividades que le permitan verificar al desarrollador que los componentes unitarios están codificados bajo condiciones de robustez, esto es, soportando el ingreso de datos erróneos o inesperados y demostrando así la capacidad de tratar errores de manera controlada.

En el sistema de información **EFFECTIVE RECORD** se usa este tipo de prueba para verificar que los atributos de las clases estén recibiendo los parámetros correctamente sin ningún tipo de problema, es decir creamos un aprueba para validar que los datos que nombramos al crear esta clase estén bien escritos y concuerde con la sintaxis del lenguaje de programación que se usa para desarrollar este sistema, Además de esto se verifica que los métodos que pertenecen a la clase estén depurando bien al momento de inicializar la prueba. Esta prueba como se puede evidenciar en la imagen se realiza de acuerdo al registro de usuarios del sistema y se puede observar además que se ejecuta de forma satisfactoria.

Cabe recalcar que estas pruebas se realizaron en la etapa de análisis donde se crearon las clases y sus métodos.

PRUEBA UNITARIA

```
public static void main(String[] args) {

    USUARIOS_DAO mi_usuario_dao = new USUARIOS_DAO();
    USUARIOS mi_usuario = new USUARIOS();

    Scanner teclado = new Scanner(System.in);

    String nombresusuario = "";
    String apellidosusuario = "";
    String identificacionusuario = "";
    String claveusuario = "";
    String telefonousuario;
    String direccionusuario = "";
    String correousuario = "";
    int id_tipo_usuario;
    int id_estado_usuario;
    int id_genero_usuario;
    int id_doc_identificacion_usuario;
```

EJECUCIÓN DE PRUEBA

```
Output - Effective (run)
Carlos
Ingrese los apellidos del usuario
Andres
Ingrese la nueva clave del usuario
1212
Ingrese la dirección del usuario
Calle 68
Ingrese el correo del usuario
carlitos@gmail.com
Ingrese el teléfono del usuario
2003250
Ingrese el número de identificación del usuario
1000254110
Ingrese el id del tipo usuario
1
Ingrese el id del estado del usuario
1
Ingrese el id del genero del usuario
1
Ingrese el id del tipo de documento del usuario
1
Conexión establecida con la base de datos effective
Registro exitoso
BUILD SUCCESSFUL (total time: 50 seconds)
```

PRUEBAS DE INTEGRACIÓN

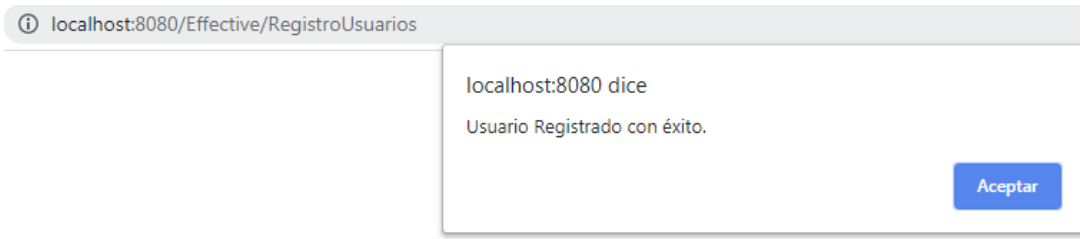
Este tipo de pruebas son ejecutas por el equipo de desarrollo de **EFFECTIVE RECORD** y consisten en la comprobación de que elementos del software interactúan entre sí, además de esto se puede observar la forma de cómo funcionan de manera correcta los distintos módulos que componen la solución una vez que han sido probados unitariamente con el fin de comprobar que interactúan correctamente a través de sus interfaces internas o externas, cubriendo así lo que es la parte de la funcionabilidad establecida por el cliente y que estos se ajusten a los criterios solicitados por el cliente final.

En este tipo de prueba se evidencio como los métodos de la clase interactúan entre sí con un formulario de registro para cumplir distintas funcionalidades, **EFFECTIVE RECORD** realiza una prueba donde se puede ver la funcionalidad de estos dos módulos trabajando entre sí para registrar un usuario en el sistema de información sin presentar ningún tipo de percance, ya que se realiza la prueba unitaria que nos ayuda a demostrar que las los métodos de la clase están recibiendo los parámetros sin ningún tipo de problema.

MÉTODO QUE RESIBE LOS PARAMETROS

```
*/
public class USUARIOS_DAO {

    public String AdicionarUsuarios(USUARIOS Usuarios) {

        String miRespuesta;
        conexion miConexion = new conexion();
        Connection nuevaCon;
        nuevaCon = miConexion.getConn();

        PreparedStatement sentencia;

        try {

            String Query = "Insert into USUARIOS (nombres_usuario, apellidos_usuario, identificacion_usuario, clave, telefo
            sentencia = nuevaCon.prepareStatement(Query);

            sentencia.setString(1, Usuarios.getNombres_usuario());
            sentencia.setString(2, Usuarios.getApellidos_usuario());
            sentencia.setString(3, Usuarios.getIdentificacion_usuario());
            sentencia.setString(4, Usuarios.getClave());
```

EJECUCIÓN DE PRUEBA



PRUEBA DE SISTEMA

Este tipo de pruebas deben ser ejecutadas idealmente por un equipo de pruebas ajeno al equipo de desarrollo, una buena práctica en este punto corresponde a la tercerización de esta responsabilidad. La obligación de este equipo, consiste en la ejecución de actividades de prueba en donde se debe verificar que la funcionalidad total de un sistema fue implementada de acuerdo a los documentos de especificación definidos en el proyecto.

EFFECTIVE RECORD realizara algunas de las pruebas no funcionales que le permitirán conocer de qué forma funciona y de qué forma se comporta el sistema de información en ciertos momentos, estas pruebas no funcionales serán las de **carga y las de estrés**.

De esta forma se conocerán sus problemas al someterlas a este tipo de pruebas, para así realizar ajustes y mejoras que contribuyan con un mejor funcionamiento del sistema.

Estas pruebas consisten en lo siguiente

PRUEBA DE CARGA.

La prueba de carga se refiere a la capacidad que tiene un software, para atender a un conjunto de usuarios al mismo tiempo, Por ello, las actividades de esta etapa tienen relación con comprobar, de manera anticipada, el funcionamiento que tendrá el software cuando esté en plena operación.

Las pruebas en este caso consisten en simular una carga de trabajo similar y superior a la que tendrá cuando el software esté funcionando, con el fin de detectar si los (programas y aplicaciones) cumple con los requerimientos de muchos usuarios simultáneos.

PRUEBA DE ESTRÉS

La prueba de *estrés*. Evalúa y pone a prueba la robustez y la confiabilidad del *software* sometiéndolo a condiciones de uso extremas. Entre estas condiciones se incluyen el envío excesivo de peticiones y la ejecución en condiciones de hardware limitadas. El objetivo es saturar el programa hasta un punto de quiebre donde aparezcan *bugs* (defectos) potencialmente peligrosos.

PRUEBAS DE ACEPTACIÓN

Las pruebas de aceptación son las últimas pruebas realizadas donde el cliente prueba el software y verifica que cumpla con sus expectativas. Estas pruebas generalmente son funcionales y se basan en los requisitos definidos por el cliente y deben hacerse antes de la salida a producción.

Estas pruebas se realizan una vez que ya se ha aprobado que cada módulo funcione bien por separado, que el software realice las funciones esperadas y que todos los módulos se integren correctamente.

EFFECTIVE RECORD una vez tenga probado todos los módulos y demuestren que están 100% funcionales, se reunirá con el cliente antes de sacar el sistema a producción y acordaran ciertas pruebas que necesita el cliente final, para posteriormente realizarlas a vista del mismo y mostrarle que el sistema de información funciona y realiza todos los procesos de forma exacta y sin ningún problema.

Esta última prueba busca que el cliente final, verifique las pruebas de software con el fin de que el cliente observe y defina si el sistema cumplió con sus expectativas.

DIAGRAMA DE CLASES

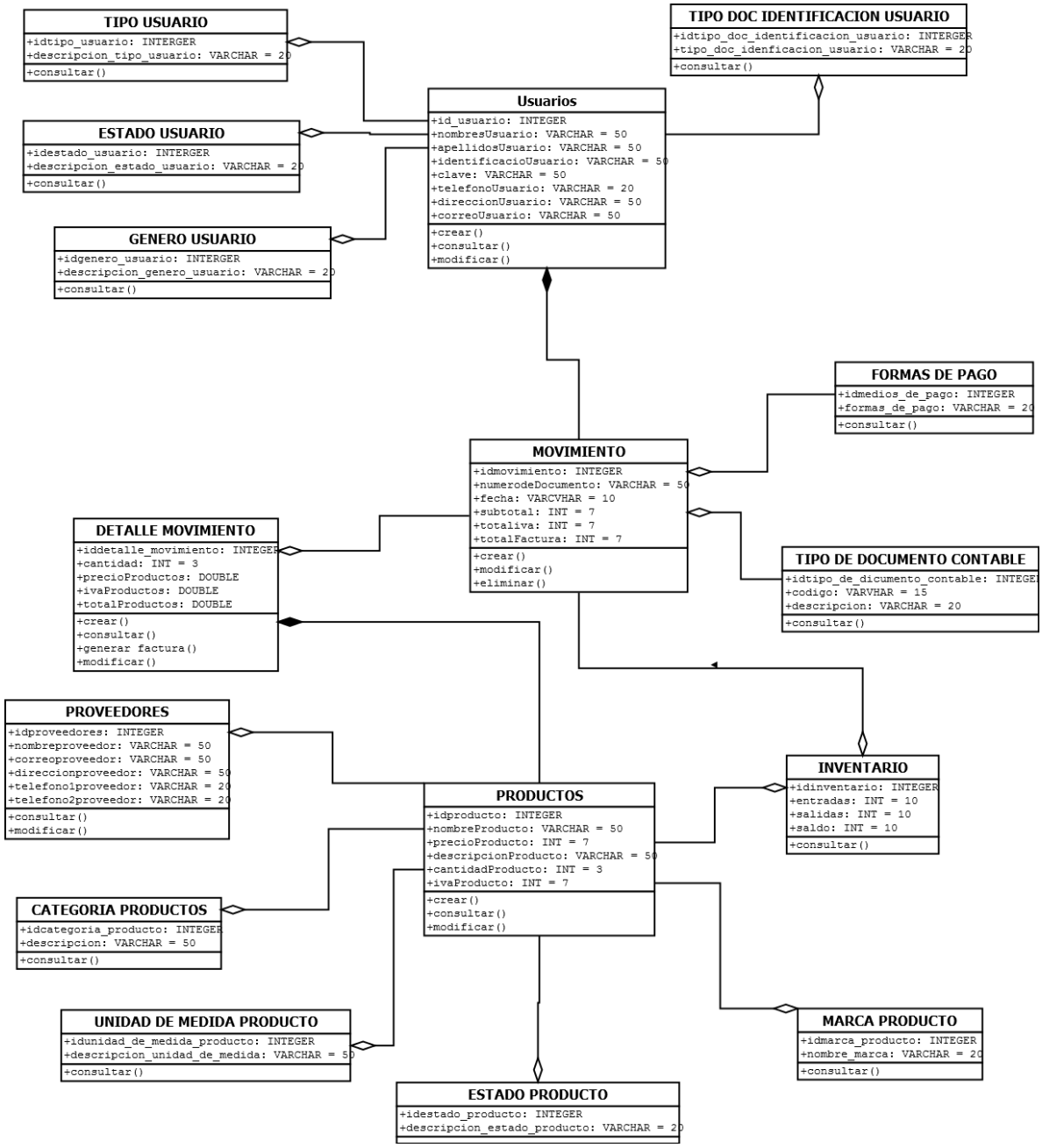


DIAGRAMA DE PAQUETES

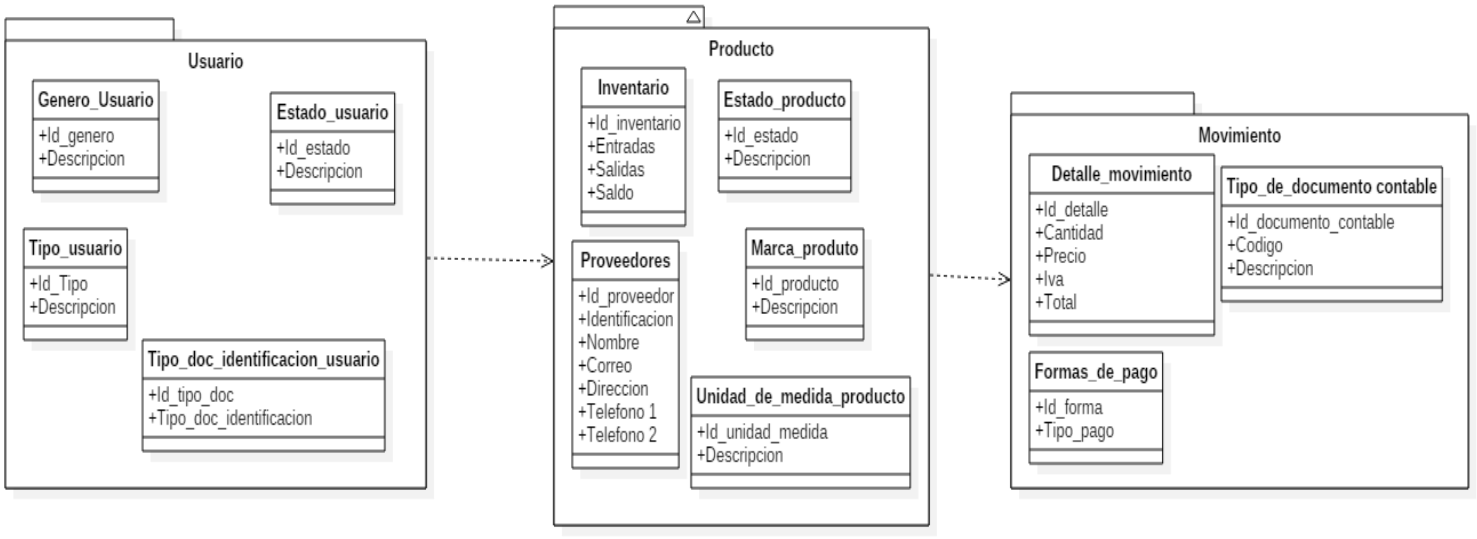


DIAGRAMA DE COMPONENTES

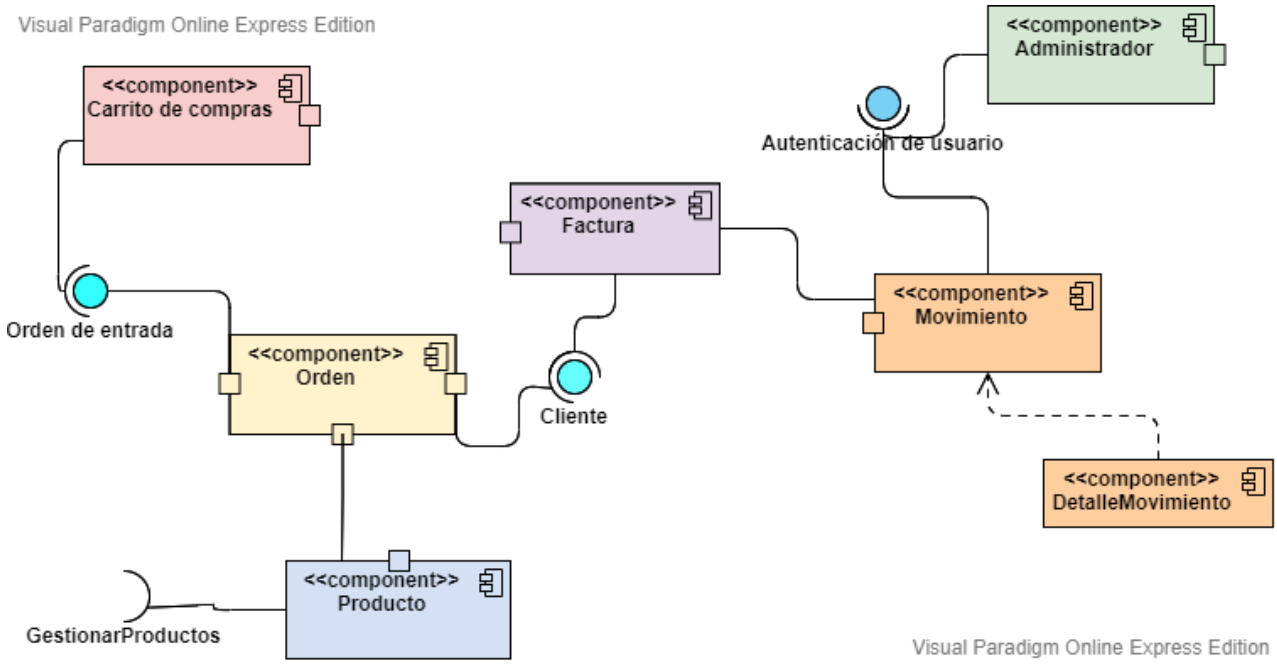


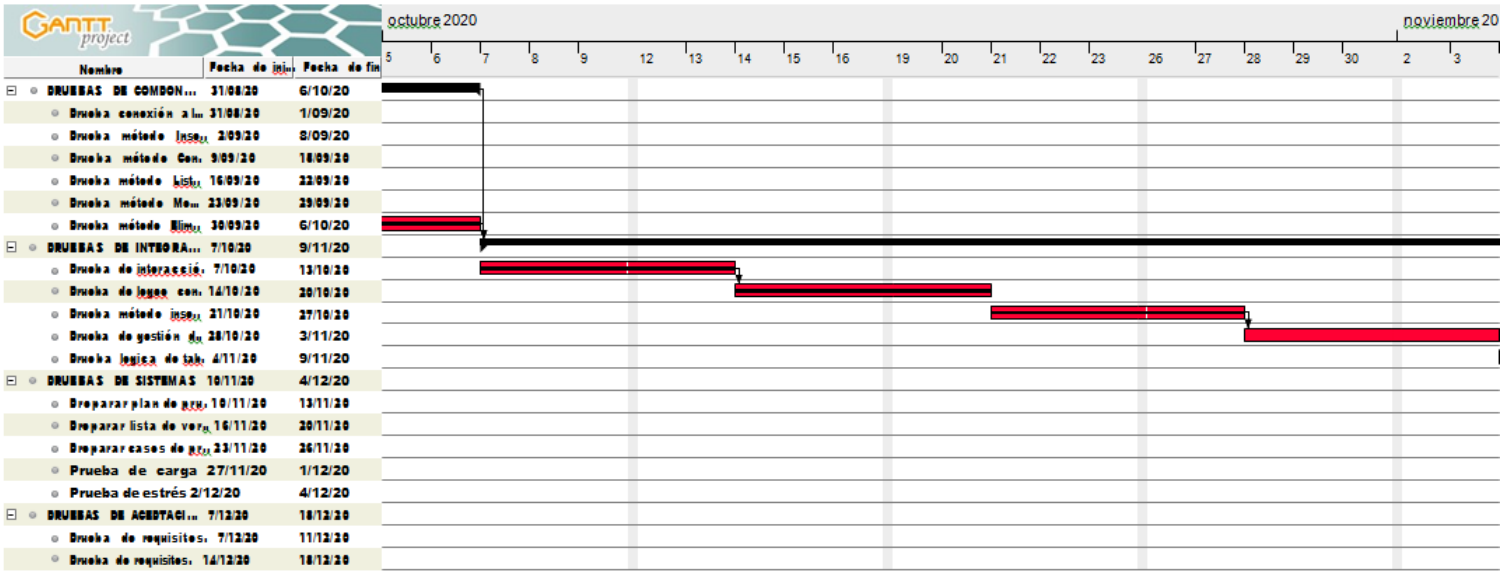
DIAGRAMA DE GANTT

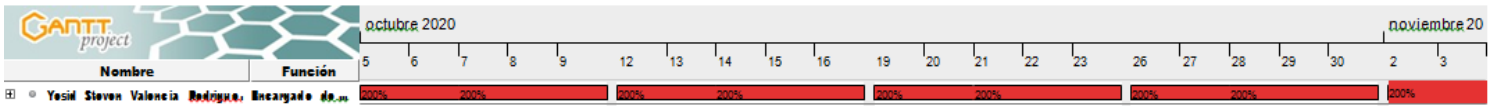
Encargado del proyecto
Fechas de inicio y fin del proyecto 31-ago-2020 - 19-dic-2020

Progreso 52%
Tarea 22
Recursos 1

Nombre	Fecha de inicio	Fecha de fin
PRUEBAS DE COMPONENTES	31/08/20	6/10/20
Prueba conexión a la BD	31/08/20	1/09/20
Prueba método Insertar datos	2/09/20	8/09/20
Prueba método Consultar datos	9/09/20	15/09/20
Prueba método Listar datos	16/09/20	22/09/20
Prueba método Modificar datos	23/09/20	29/09/20
Prueba método Eliminar datos	30/09/20	6/10/20
PRUEBAS DE INTEGRACIÓN	7/10/20	9/11/20
Prueba de interacción entre métodos	7/10/20	13/10/20
Prueba de logeo con la BD	14/10/20	20/10/20
Prueba método insertar con el formulario	21/10/20	27/10/20
Prueba de gestión de datos	28/10/20	3/11/20
Prueba logica de tabla movimiento y detalle movimiento	4/11/20	9/11/20
PRUEBAS DE SISTEMAS	10/11/20	4/12/20
Preparar plan de pruebas	10/11/20	13/11/20
Preparar lista de verificaciones de los requisitos	16/11/20	20/11/20
Preparar casos de prueba	23/11/20	26/11/20
Prueba de carga	27/11/20	1/12/20
Prueba de estrés	2/12/20	4/12/20
PRUEBAS DE ACEPTACIÓN	7/12/20	18/12/20
Prueba de requisitos funcionales del sistema	7/12/20	11/12/20
Prueba de requisitos no funcionales del sistema	14/12/20	18/12/20

Nombre	Función
Yesid Steven Valencia Rodriguez	Encargado de pruebas





PRUEBA DE DISEÑO QUE SE VA A APLICAR

PRUEBA DE CLASE: USUARIOS_DAO Y USUARIOS.

MÉTODO: INSERTAR.

PRUEBA REALIZADA POR: ANDRÉS FELIPE SÁENZ SALAZAR.

Esta prueba de componentes se realiza para examinar que la clase **USUARIOS_DAO,** este recibiendo los parámetros establecidos anteriormente por los Setters y Getters que provienen de otra clase llamada **USUARIOS,** estos sirven para asignar y recibir parámetros que establezcamos dentro de esa clase. Además de esto poder observar que el Query que se escribe y realiza en la clase **USUARIOS_DAO** este bien escrito y cumpla con los parámetros de inserción para la posterior adición de información a la bases de datos.

Esta prueba consiste en traer las clases de **USUARIOS_DAO** y **USUARIOS** para acceder a los métodos de estas, de esta forma obtendremos el método de insertar de la clase **USUARIOS_DAO** y los atributos de la clase **USUARIOS.**

MÉTODO INSERTAR

```
public class USUARIOS_DAO {

    public String AdicionarUsuarios(USUARIOS Usuarios) {

        String miRespuesta;
        conexion miConexion = new conexion();
        Connection nuevaCon;
        nuevaCon = miConexion.getConn();

        PreparedStatement sentencia;

        try {

            String Query = "Insert into USUARIOS (nombres_usuario, apellidos_usuario, identificacion_usuario, clave, telefon
            sentencia = nuevaCon.prepareStatement(Query);

            sentencia.setString(1, Usuarios.getNombres_usuario());
            sentencia.setString(2, Usuarios.getApellidos_usuario());
            sentencia.setString(3, Usuarios.getIdentificacion_usuario());
            sentencia.setString(4, Usuarios.getClave());
            sentencia.setString(5, Usuarios.getTelefono_usuario());
            sentencia.setString(6, Usuarios.getDireccion_usuario());
            sentencia.setString(7, Usuarios.getCorreo_usuario());
            sentencia.setInt(8, Usuarios.getTIPO_USUARIO_id_tipo_usuario());
            sentencia.setInt(9, Usuarios.getESTADO_USUARIO_id_estado_usuario());
            sentencia.setInt(10, Usuarios.getGENERO_USUARIO_id_genero_usuario());
            sentencia.setInt(11, Usuarios.getTIPO_DOC_IDENTIFICACION_USUARIO_id_doc_identificacion_usuario());
            sentencia.execute();
            miRespuesta = "";

        } catch (Exception ex) {

            miRespuesta = ex.getMessage();
            System.out.println("Ocurrio un error en AdicionarTipo_usuario_USUARIO_DAO\n" + miRespuesta);

        }

        return miRespuesta;

    }
```

PRUEBA DE COMPONENTE

```
USUARIOS_DAO mi_usuario_dao = new USUARIOS_DAO();
USUARIOS mi_usuario = new USUARIOS();

Scanner teclado = new Scanner(System.in);

String nombresusuario = "";
String apellidosusuario = "";
String identificacionusuario = "";
String claveusuario = "";
String telefonousuario;
String direccionusuario = "";
String correousuario = "";
int id_tipo_usuario;
int id_estado_usuario;
int id_genero_usuario;
int id_doc_identificacion_usuario;

System.out.println("Ingrese los nombres del usuario");
nombresusuario = teclado.nextLine();

System.out.println("Ingrese los apellidos del usuario");
apellidosusuario = teclado.nextLine();

System.out.println("Ingrese la nueva clave del usuario");
claveusuario = teclado.nextLine();

System.out.println("Ingrese la dirección del usuario");
direccionusuario = teclado.nextLine();

System.out.println("Ingrese el correo del usuario");
correousuario = teclado.nextLine();

System.out.println("Ingrese el teléfono del usuario");
telefonousuario = teclado.nextLine();

System.out.println("Ingrese el número de identificación del usuario");
identificacionusuario = teclado.nextLine();
```

EJECUCIÓN DE PRUEBA

```
Output - Effective (run)
Carlos
Ingrese los apellidos del usuario
Andres
Ingrese la nueva clave del usuario
1212
Ingrese la dirección del usuario
Calle 68
Ingrese el correo del usuario
carlitos@gmail.com
Ingrese el teléfono del usuario
2003250
Ingrese el número de identificación del usuario
1000254110
Ingrese el id del tipo usuario
1
Ingrese el id del estado del usuario
1
Ingrese el id del genero del usuario
1
Ingrese el id del tipo de documento del usuario
1
Conexión establecida con la base de datos effective
Registro exitoso
BUILD SUCCESSFUL (total time: 50 seconds)
```

PRUEBA DE CLASE: PROVEEDORES_DAO Y PROVEEDORES.

MÉTODO: ELIMINAR.

PRUEBA REALIZADA POR: YESID STEVEN VALENCIA RODÍGUEZ

Esta prueba de componentes se realiza para examinar que la clase **PROVEEDORES_DAO,** este recibiendo los parámetros establecidos anteriormente por los Setters y Getters que provienen de otra clase llamada **PROVEEDORES,** estos sirven para asignar y recibir parámetros que establezcamos dentro de esa clase. Además de esto poder observar que el Query que se escribe y realiza en la clase **PROVEEDORES_DAO** este bien escrito y cumpla con los parámetros de eliminación de información para la bases de datos.

Esta prueba consiste en traer las clases de **PROVEEDORES_DAO** y **PROVEEDORES** para acceder a los métodos de estas, de esta forma obtendremos el método de eliminar de la clase **PROVEEDORES_DAO** y los atributos de la clase **PROVEEDORES.**

MÉTODO ELIMINAR

```
public String EliminarProveedores(PROVEEDORES Proveedores) {

    String miRespuesta;

    conexion miConexion = new conexion();
    Connection nuevaCon;
    nuevaCon = miConexion.getConn();

    PreparedStatement sentencia;

    try {

        String Query = "delete from PROVEEDORES where identificacion_proveedor = ? and id_proveedor= ? ;
        sentencia = nuevaCon.prepareStatement(Query);
        sentencia.setString(1, Proveedores.getIdentificacion_proveedor());
        sentencia.setInt(2, Proveedores.getId_proveedor());
        sentencia.execute();
        miRespuesta = "";

    } catch (Exception ex) {

        miRespuesta = ex.getMessage();
        System.err.println("Ocurrio un error en PROVEEDORES_DAO.EliminarProveedor" + ex.getMessage());
    }
    return miRespuesta;
}
```

PRUEBA DE COMPONENTE

```
public class pruebaEliminarPROVEEDORES {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        PROVEEDORES_DAO mi_proveedor_dao = new PROVEEDORES_DAO();
        ArrayList<PROVEEDORES> mi_lista_proveedor = mi_proveedor_dao.ConsultarListadoProveedor("02","Scribe");

        for (PROVEEDORES C : mi_lista_proveedor) {

            System.out.println("Número de id - " + C.getId_proveedor()+ "\n" + "Nombre de proveedor - " + C.getNombre_proveed
        }

        System.out.println("El registro se elimino satisfactoriamnete");
        mi_proveedor_dao.EliminarProveedores(mi_lista_proveedor.get(0));
        mi_lista_proveedor = mi_proveedor_dao.ConsultarListadoProveedor("02","Scribe");

        for (PROVEEDORES C : mi_lista_proveedor) {

            System.out.println("Número de id - " + C.getId_proveedor()+ "\n" + "Nombre de proveedor - " + C.getNombre_provee
        }

    }
```

EJECUCIÓN DE PRUEBA

```
Output - Effective (run)
Correo proveedor - cartulina@
Dirección proveedorCalle 200
Telefono 1 proveedor2003256
Telefono 2 proveedor7578596

Número de id - 14
Nombre de proveedor - Cartulina
Correo proveedor - cartulina@
Dirección proveedorCalle 200
Telefono 1 proveedor2003256
Telefono 2 proveedor7578596

El registro se elimino satisfactoriamnete
Conexión establecida con la base de datos effective
Conexión establecida con la base de datos effective
Buscando parametro16
Número de id - 14
Nombre de proveedor - Cartulina
Correo proveedor - cartulina@
Dirección proveedorCalle 200
Telefono 1 proveedor2003256
Telefono 2 proveedor7578596

BUILD SUCCESSFUL (total time: 6 seconds)
```

PRUEBA DE CLASE: PROVEEDORES_DAO

MÉTODO: MODIFICAR

PRUEBA REALIZADA POR: ANDRÉS ESTEBAN SOSSA RODRIGUEZ

Esta prueba se realiza ya teniendo un registro realizado exitosamente en la base de datos. La prueba se hace con el fin de que la clase de **PROVEEDORES_DAO** encuentre los datos en la base de datos y por medio del registro modifique cualquier dato mal registrado o no registrado, una vez que recibe estos parámetros modificados se dirigen a la base de datos para que se realice su actualización.

La prueba de modificación de datos consiste en que se ingrese la identificación del proveedor para que el sistema encuentre el registro, después de encontrarlo pedirá que registre nuevamente los datos pedidos por el sistema, los datos que están bien registrados los puede escribir tal cual como en el registro y los datos que no fueron llenados correctamente, puede actualizarlos.

MÉTODO MODIFICAR

```
public String ModificarProveedores(PROVEEDORES Proveedores) {

    String miRespuesta;
    conexion miConexion = new conexion();
    Connection nuevaCon;
    nuevaCon = miConexion.getConn();

    PreparedStatement sentencia;

    try {
        String Query = "update PROVEEDORES set identificacion_proveedor=?, nombre_proveedor=?, correo_proveedor=?,
        sentencia = nuevaCon.prepareStatement(Query);
        sentencia.setString(1, Proveedores.getIdentificacion_proveedor());
        sentencia.setString(2, Proveedores.getNombre_proveedor());
        sentencia.setString(3, Proveedores.getCorreo_proveedor());
        sentencia.setString(4, Proveedores.getDireccion_proveedor());
        sentencia.setString(5, Proveedores.getTelefono_1_proveedor());
        sentencia.setString(6, Proveedores.getTelefono_2_proveedor());
        sentencia.setString(7, Proveedores.getIdentificacion_proveedor());

        sentencia.executeUpdate();
        miRespuesta = "";

    } catch (Exception ex) {

        miRespuesta = ex.getMessage();
        System.out.println("Ocurrio un error en ModificarProveedores_PROVEEDORES_DAO_MODIFICAR\n" + miRespuesta);

    }

    return miRespuesta;

}
```

PRUEBA MÉTODO MODIFICAR

```
String identificacionproveedor = "";
String nombreproveedor = "";
String correoproveedor = "";
String direccionproveedor = "";
String telefono1proveedor;
String telefono2proveedor;


System.out.println("Ingrese el nuevo número de identificación del proveedor para modificar");
identificacionproveedor  = teclado.nextLine();

System.out.println("Ingrese el nuevo nombre del proveedor ");
nombreproveedor = teclado.nextLine();

System.out.println("Ingrese el nuevo correo del proveedor");
correoproveedor = teclado.nextLine();

System.out.println("Ingrese la nueva dirección del proveedor");
direccionproveedor = teclado.nextLine();

System.out.println("Ingrese el nueva teléfono del proveedor");
telefono1proveedor = teclado.nextLine();

System.out.println("Ingrese el nueva teléfono del proveedor si lo tiene");
telefono2proveedor = teclado.next();

mi_proveedor.setIdentificacion_proveedor(identificacionproveedor);
mi_proveedor.setNombre_proveedor(nombreproveedor);
mi_proveedor.setCorreo_proveedor(correoproveedor);
mi_proveedor.setDireccion_proveedor(direccionproveedor);
mi_proveedor.setTelefono_1_proveedor(telefono1proveedor);
mi_proveedor.setTelefono_2_proveedor(telefono2proveedor);

mi_proveedor.setIdentificacion_proveedor(identificacionproveedor);
String miRespuesta = mi_proveedor_dao.ModificarProveedores(mi_proveedor);

if (miRespuesta.length() == 0) {

    System.out.println("Modificación exitosa");

} else {

    System.out.println("Ocurrio un error" + miRespuesta);

}
```

EJECUCION DE PRUEBA

```
run:
Ingrese el nuevo número de identificación del proveedor para modificar
1002358
Ingrese el nuevo nombre del proveedor
Norma
Ingrese el nuevo correo del proveedor
Productosnorma@norma.colombia.com.co
Ingrese la nueva dirección del proveedor
calle10c-45-b
Ingrese el nueva teléfono del proveedor
3208427889
Ingrese el nueva teléfono del proveedor si lo tiene
5263698
Conexión establecida con la base de datos effective
Modificación exitosa
BUILD SUCCESSFUL (total time: 1 minute 20 seconds)
```

PRUEBA DE CLASE: PROVEEDORES_DAO.

METODO: LISTAR.

PRUEBA REALIZADA POR: LADY TATIANA CHITIVO CAICEDO.

Esta prueba de componentes se usa para examinar que la clase **PROVEEDORES_DAO** esté recibiendo los parámetros correctamente, y de esta forma obtener el método listar que nos mostrará los registros que existen en la base de datos. Además de esto poder observar que el Query que se escribe y realiza en la clase **PROVEEDORES_DAO** este bien escrito y cumpla con los parámetros para listar la información que hay en la base de datos.

MÉTODO LISTAR

```
package controlador;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import modelo.PROVEEDORES;

public class PROVEEDORES_DAO {

    public String AdicionarProveedores(PROVEEDORES Proveedores) {

        String miRespuesta;
        conexion miConexion = new conexion();
        Connection nuevaCon;
        nuevaCon = miConexion.getConn();

        PreparedStatement sentencia;

        try {

            String Query = "Insert PROVEEDORES (identificacion_proveedor,nombre_proveedor,correo_p
            sentencia = nuevaCon.prepareStatement(Query);
            sentencia.setString(1, Proveedores.getIdentificacion_proveedor());
            sentencia.setString(2, Proveedores.getNombre_proveedor());
            sentencia.setString(3, Proveedores.getCorreo_proveedor());
            sentencia.setString(4, Proveedores.getDireccion_proveedor());
            sentencia.setString(5, Proveedores.getTelefono_1_proveedor());
            sentencia.setString(6, Proveedores.getTelefono_2_proveedor());

            sentencia.execute();
            miRespuesta = "";

        } catch (Exception ex) {

            miRespuesta = ex.getMessage();
            System.out.println("Ocurrio un error en AdicionarPROVEEDORES_DAO\n" + miRespuesta);

        }

        return miRespuesta;
```

PRUEBA MÉTODO LISTAR

```
package pruebas;

import controlador.PROVEEDORES_DAO;
import java.util.ArrayList;
import modelo.PROVEEDORES;

/**
 *
 * @author AndresSaenz
 */
public class pruebaListarPROVEEDORES {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        PROVEEDORES_DAO proveedor_dao = new PROVEEDORES_DAO();
        ArrayList<PROVEEDORES> mi_proveedor = new ArrayList<PROVEEDORES>();
        mi_proveedor = proveedor_dao.ConsultarListadoProveedor("", "");

        int size = mi_proveedor .size();
        System.out.println("<table border=\"1\"><br><td>id_proveedor</td><td>nombre_proveedor</td><td>correo_

        for (PROVEEDORES U : mi_proveedor) {

            System.out.println("<tr>");
            System.out.println("<td>" + U.getIdentificacion_proveedor() + "</td>");
            System.out.println("<td>" + U.getNombre_proveedor() + "</td>");
            System.out.println("<td>" + U.getCorreo_proveedor() + "</td>");
            System.out.println("<td>" + U.getDireccion_proveedor() + "</td>");
            System.out.println("<td>" + U.getTelefono_1_proveedor() + "</td>");
            System.out.println("<td>" + U.getTelefono_2_proveedor() + "</td>");
            System.out.println("</tr>");

        }

        System.out.println("</table>");
```

EJECUCION DE PRUEBA

```
Conexión establecida con la base de datos effective
Buscando parametro
<table border="1"><br><td>id_proveedor</td><td>nombre_proveedor</td><td>correo_proveedor</td><td>direccion_
<tr>
<td>08</td>
<td>Papermate</td>
<td>Papermate@gmail.com</td>
<td>Calle el dorado</td>
<td>2003250</td>
<td>7578596</td>
</tr>
<tr>
<td>08</td>
<td>Colbones </td>
<td>pega@gmail.com</td>
<td>Calle el dorado</td>
<td>7578596</td>
<td>45458596</td>
</tr>
<tr>
<td>09</td>
<td>Lego</td>
<td>Lego@gmail.com</td>
```

INFORME DE RESULTADOS

EFFECTIVE RECORD puede constatar que las diferentes pruebas que se realizaron en los métodos de las clases están funcionando en su totalidad ya que estas pruebas fueron diseñadas y estructuradas para que mostraran el funcionamiento completo de cada clase y sus métodos. Además de esto se puede evidenciar que se realizaron otras pruebas que muestran que algunas funcionalidades trabajando entre si con elementos del mismo sistema para realizar tareas en específico. Estas pruebas son las de integración que dejan examinar cómo funcionan componentes del sistema entre si sin presentar algún tipo de problema.

EFFECTIVE RECORD espera que las pruebas que se designaron anteriormente y que hacen falta por completarse cumplan con las expectativas finales, esto quiere decir que al momento de aplicarlas muestren algún tipo de error que se pueda corregir a tiempo, antes de que pase a pruebas de aceptación donde se le mostrara al cliente final que todos los requisitos que el planteo al comienzo del desarrollo del sistema están 100 % funcionales y no presentan ningún problema.