**Oswaldo Isaias Hernández Santes A01199004**

**SICT0201 – Determinación de patrones / SICT0202 – Interpretación de variables**

La OSF (Organización Socio-Formadora) nos extendió su problemática la cual básicamente consistía en el que seguían un proceso de gestión de ventas entre empresas externas que hacen lives en la plataforma Facebook, pero lo más relevante fue que mencionaron que este proceso es manual (cálculos manuales con ayuda de hojas de cálculo y entre otras cosas) las cuales hacen el trabajo muy ineficiente al punto de administrar hasta 40 hojas de cálculo en un solo libro y eso era exclusivamente para productos de sus vendedores (las empresas externas).

Entonces gracias a varios datos que nos proporcionó la OSF se pudo pensar en un dashboard en el cual pudieran acceder de forma rápida a este sistema, donde pudieran visualizar las ventas, los vendedores registrados, los productos que tienen sus vendedores, un apartado que les dijera cuando se modifican datos sensibles como lo son los productos de los vendedores, las cuentas por cliente de cada vendedor, etc.

Transformamos esos datos proporcionados a este dashboard en el cual tomar decisiones será más fácil, por ejemplo al momento de usar la aplicación que filtra los datos esto ya lo hace de forma automática y le muestra al usuario los datos que no fueron procesados de manera correcta y el usuario solo tiene que verificar una mínima cantidad de datos.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Para el análisis de estos mismos datos mostraré un fragmento el cuál interpreta formas de pedir un producto con su cantidad.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Es un código en Python que analiza un archivo .csv y lo interpreta para solo obtener datos útiles para la OSF.

Texto

El contenido generado por IA puede ser incorrecto.

Por último se le brindan 2 archivos a la OSF los cuales tiene completa libertad de revisar, estos archivos se brindan en formato .csv y dentro de la misma app pueden editar estos archivos.

Otro ejemplo donde la OSF puede tener acceso es a los registros del sistema, es decir, cuando se edita algún campo dentro de la base de datos esta misma mediante un trigger (en la base de datos) genera un registro de la modificación dependiendo el caso y estos registros pueden visualizarlos ellos mismos en caso de que algún vendedor haya tenido un problema siempre hay un registro que los respalde.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Los triggers siguen una misma lógica, a continuación el pseudocódigo y el algoritmo de los triggers para cada tabla en la base de datos:

PARA CADA TABLA EN EL SISTEMA:

// TRIGGER AFTER INSERT

CUANDO se\_inserta\_registro EN tabla ENTONCES:

INSERTAR EN log:

accion\_evento = 'INSERT'

entidad\_evento = nombre\_de\_la\_tabla

fecha\_evento = fecha\_y\_hora\_actual

descripcion\_evento = 'Nuevo [entidad] creado/registrado [identificador]'

id\_usuario = usuario\_actual O usuario\_por\_defecto(1)

registro\_id = ID\_del\_nuevo\_registro

valor\_anterior = NULL

valor\_nuevo = campos\_relevantes\_concatenados\_del\_nuevo\_registro

FIN CUANDO

// TRIGGER AFTER UPDATE

CUANDO se\_actualiza\_registro EN tabla ENTONCES:

INSERTAR EN log:

accion\_evento = 'UPDATE'

entidad\_evento = nombre\_de\_la\_tabla

fecha\_evento = fecha\_y\_hora\_actual

descripcion\_evento = '[Entidad] actualizado [identificador]'

id\_usuario = usuario\_actual O usuario\_por\_defecto(1)

registro\_id = ID\_del\_registro\_actualizado

valor\_anterior = campos\_relevantes\_concatenados\_del\_registro\_anterior

valor\_nuevo = campos\_relevantes\_concatenados\_del\_registro\_nuevo

FIN CUANDO

// TRIGGER AFTER DELETE (o BEFORE DELETE para usuario)

CUANDO se\_elimina\_registro EN tabla ENTONCES:

INSERTAR EN log:

accion\_evento = 'DELETE'

entidad\_evento = nombre\_de\_la\_tabla

fecha\_evento = fecha\_y\_hora\_actual

descripcion\_evento = '[Entidad] eliminado [identificador]'

id\_usuario = usuario\_actual O usuario\_por\_defecto(1)

registro\_id = ID\_del\_registro\_eliminado

valor\_anterior = campos\_relevantes\_concatenados\_del\_registro\_eliminado

valor\_nuevo = NULL

FIN CUANDO

FIN PARA CADA TABLA

Funciones auxiliares necesarias para saber qué usuario modifica los campos

FUNCIÓN obtener\_usuario\_actual():

SI @current\_user\_id EXISTE ENTONCES:

RETORNAR @current\_user\_id

SINO:

RETORNAR 1 // Usuario por defecto

FIN SI

FIN FUNCIÓN

FUNCIÓN concatenar\_campos\_relevantes(registro):

cadena\_resultado = ""

PARA CADA campo\_importante EN registro:

SI campo NO es NULL ENTONCES:

cadena\_resultado += nombre\_campo + ": " + valor\_campo

SINO:

cadena\_resultado += nombre\_campo + ": NULL"

FIN SI

SI no\_es\_ultimo\_campo ENTONCES:

cadena\_resultado += " | "

FIN SI

FIN PARA CADA

RETORNAR cadena\_resultado

FIN FUNCIÓN

FUNCIÓN generar\_descripcion(accion, entidad, identificador):

SEGÚN accion:

CASO 'INSERT':

RETORNAR "Nuevo " + entidad + " creado/registrado " + identificador

CASO 'UPDATE':

RETORNAR entidad + " actualizado " + identificador

CASO 'DELETE':

RETORNAR entidad + " eliminado " + identificador

FIN SEGÚN

FIN FUNCIÓN

Patrón de Datos por Acción del usuario

ESTRUCTURA del registro de log:

accion\_evento: ['INSERT', 'UPDATE', 'DELETE']

entidad\_evento: nombre\_de\_la\_tabla\_afectada

fecha\_evento: timestamp\_actual

descripcion\_evento: mensaje\_descriptivo\_de\_la\_acción

id\_usuario: ID\_del\_usuario\_que\_ejecuta\_la\_acción

registro\_id: ID\_del\_registro\_afectado

valor\_anterior: {

INSERT: NULL

UPDATE: estado\_antes\_del\_cambio

DELETE: estado\_del\_registro\_eliminado

}

valor\_nuevo: {

INSERT: estado\_del\_nuevo\_registro

UPDATE: estado\_después\_del\_cambio

DELETE: NULL

}

Procedimientos de configuración que obtienen usuario actual (que edita los campos)

PROCEDIMIENTO establecer\_usuario\_actual(id\_usuario):

@current\_user\_id = id\_usuario

FIN PROCEDIMIENTO

PROCEDIMIENTO obtener\_usuario\_actual():

RETORNAR @current\_user\_id O 1

FIN PROCEDIMIENTO

Básicamente ese es el pseudocódigo y en base a ese se hicieron los triggers necesarios el cual reciben los campos modificados (un antes y un después o un nuevo dato o una eliminación) y se registra quién lo hizo.

**STC0201 – Aplicación de metodologías de software**

Para la parte de la gestión del proyecto se esperaba utilizar alguna metodología como ScrumBan, Kaban, etc; se tenía previsto un plan de trabajo, pero muchas veces no se ajustó a los tiempos de cada quien por lo cual yo empecé a trabajar por cuenta propia y luego verificábamos avances para revisar qué podíamos agregar, mejorar o eliminar cada cierto periodo de tiempo (2-3dias), adjunto captura del plan de trabajo que estaba siguiendo.

Texto

El contenido generado por IA puede ser incorrecto.

**STC0202 – Definición de requerimientos de software**

Para esta parte de definición de requerimientos de software propuse específicamente el uso de la aplicación que desarrollé para la filtración de los datos y su misma edición dentro de esa aplicación y que de la aplicación de filtrado de datos derivan 2 archivos en formato .csv los cuales son editables dentro de la misma aplicación.

También propuse la inserción de los datos a partir de un archivo en formato .csv hacia la aplicación web siguiendo el mismo formato del archivo .csv y que se pudiera almacenar en la base de datos además de que se mostrara a la OSF.

Tabla

El contenido generado por IA puede ser incorrecto.

Para obtener una definición más clara de los requerimientos se le solicitaba información a la OSF en donde contemplaba casos más específicos como la paleta de colores de la página, los datos que pudieran procesarse, etc (adjunto capturas).

Texto

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

**STC0203 – Diseño de componentes de software**

Para el diseño del backend se propuso el uso de arquitectura MVC el cual consiste en una arquitectura Modelo Vista Controlador, el cual separa cada módulo para hacerlo independiende uno del otro y así evitar dañar los demás componentes que están funcionando en el sistema. Al final se terminó usando Vista-Controlador, el cual no nos causó ningún problema trabajando en el proyecto. A continuación, una captura de la estructura del proyecto:

Texto

El contenido generado por IA puede ser incorrecto.

**STC0204 – Desarrollo de componentes de software**

Para el diseño de la página se contempló tener una barra superior donde pudieran seleccionar a donde quisieran navegar en la página, por lo tanto se hizo un nav.ejs en views en el cual solo se importaba en la parte superior de cada vista (para cada usuario, en este caso al administrador), este consiste en mostrar el logo de la app y al ser presionado se redirige al dashboard (dependiendo del usuario, en este caso el administrador), al pasar el cursor sobre el Menú desplegable se mostraban las diferentes funciones de la app.

Texto

El contenido generado por IA puede ser incorrecto.

Este es el código del cual se despliegan las funciones de la app.

Texto

El contenido generado por IA puede ser incorrecto.

En esta parte del código del dashboard se agrega esta barra de navegación, por lo tanto es cómodo para el usuario (en este caso el administrador) navegar en la página.

Para obtener la app creada para uso externo (fuera de la página) se usó un método conocido como fetch() el cuál es integrado para peticiones asíncronas y así no interferir con los demás procesos de la página.

Texto

El contenido generado por IA puede ser incorrecto.

Para el desarrollo del proyecto se usó como convención de código camelCase el cual consiste en definir las funciones o variables al inicio con una minúscula y lo demás que empiece en mayúsculas, por ejemplo “obtenerUsuario()”, “patronSoloNumeros”, etc.

Para trabajo colaborativo siempre se estuvo usando GitHub para poder trabajar a la par y verificar los cambios, a partir de eso sugerir modificaciones.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Siendo “tuzobus” mi compañero y “taqueritospro” yo.

**STC0205 – Elaboración de pruebas de software**

En el repositorio del equipo se encuentran los casos de prueba para la aplicación que estamos creando para la OSF (Organización Socio-Formadora), se utilizó la metodología vista en el curso para convertir diagramas de casos de uso a un caso de prueba.

**Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.STC0206 – Implantación de software**

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

El sistema está corriendo en EC2 AWS (si no mal recuerdo es Elastic Container de Amazon Web Services), el cual como presentación de proyecto es más que suficiente ya que se está usando un plan gratuito del servicio. Para un uso demandante es recomendable comprar más recursos para este servicio.

Se hizo todo en una arquitectura monolítica con microservicios (módulos que le llamamos nosotros) ya que no es muy complejo, en dado caso de que fuera más extenso se podría considerar tener un servidor dedicado a base de datos, otro para por ejemplo el módulo de la app de filtrado de datos (TwenBFive App).