

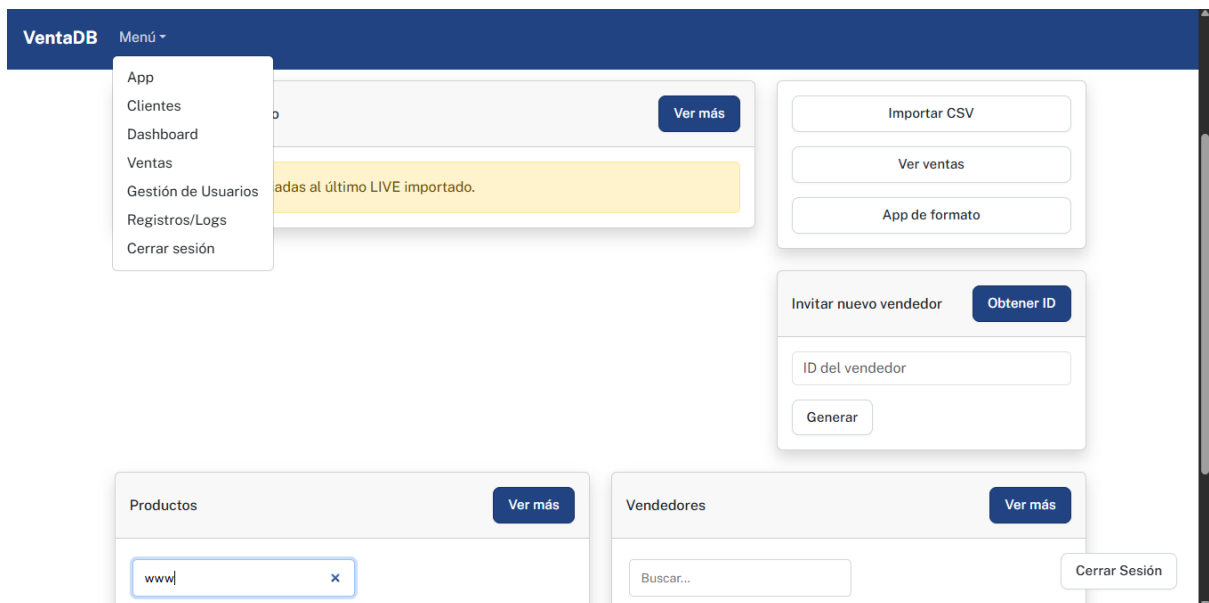
**Oswaldo Isaias Hernández Santes A01199004**

## **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.



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

```
# Verificar si tiene palabra clave "mio" o variantes
tiene_palabra_clave = any(palabra in text for palabra in ['mio', 'mia', 'yo'])

# Si no tiene palabra clave, verificar si es solo números (patrón: número cantidad o númeroxcantidad)
if not tiene_palabra_clave:
    # Buscar patrones como "203 2", "203x2", o solo "203"
    patron_solo_numeros = re.search(r'^\s*(\d{1,4})(?:\s*x\s*(\d+)|\s+(\d+))?\s*(.*)', text.strip())
    if patron_solo_numeros:
        product_id = patron_solo_numeros.group(1)
        # Cantidad puede estar en grupo 2 (con x) o grupo 3 (con espacio)
        quantity_x = patron_solo_numeros.group(2)
        quantity_space = patron_solo_numeros.group(3)
        specification = patron_solo_numeros.group(4).strip()

        if quantity_x:
            quantity = int(quantity_x)
        elif quantity_space:
            quantity = int(quantity_space)
        else:
            quantity = 1
```

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

```
# Nombres de archivos de salida con formato: YYYYMMDD_nombrearchivo_correcto/revision.csv
temp_file = os.path.join(output_dir, f"{fecha_csv}_{nombre_base}_temp.csv")
output_file_ok = os.path.join(output_dir, f"{fecha_csv}_{nombre_base}_correcto.csv")
output_file_error = os.path.join(output_dir, f"{fecha_csv}_{nombre_base}_revision.csv")
```

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.

VentaDB <span>Menú</span>								
Logs del Sistema								
ID	Acción	Entidad	Fecha	Descripción	Usuario	Registro ID	Valor Anterior	Valor Nuevo
5	INSERT	live	10/06/2025, 20:15:17	Nuevo live creado: p_20250610_1	tuxo	3	-	ID Vendedor: 123456786   Fecha live: 2025-06-10   Nombre CSV: p_20250610_1   Nombre live: p_20250610...
6	INSERT	producto_live	10/06/2025, 20:15:17	Nuevo producto agregado al live: 123	tuxo	3	-	ID Live: 3   Código: 123   Nombre: 123   Descripción: 123   Precio: 123.00   Cantidad inicial: 123

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 (2-3 días), adjunto captura del plan de trabajo que estaba siguiendo.

Actividad (Responsable)	Subtarea	Plan de trabajo Descripción	Fecha Estimada	Fecha Completada	Porcentaje
Avance del proyecto 2: Análisis y diseño de la solución. (Todos)	N/A	N/A	jueves, 17 de abril de 2025	jueves, 3 de abril de 2025	100.00%
	Diagrama de contexto	Haciendo uso formal de la nomenclatura descrita en el curso para la elaboración de este diagrama, elaborar el diagrama de contexto de nivel 0 para identificar a todos los interesados en el sistema de información que dará solución a la problemática descrita anteriormente. Recuerda detallar los flujos de datos entre interesados y sistema a un nivel de campos, ya que este será el punto de entrada para diseñar el modelo de persistencia.	jueves, 17 de abril de 2025	jueves, 3 de abril de 2025	100.00%
	Requisitos funcionales	Identificación completa de los requerimientos de la organización que pretenden cubrirse con el sistema. Se espera que se elabore un diagrama de casos de uso que cumpla con los lineamientos descritos en la Unidad de Formación Análisis de Requerimientos de Software. Tabla de priorización de requisitos que considere el riesgo, valor, complejidad y estabilidad. Detalle de los casos de uso que identifique en la tabla de alta prioridad, la plantilla a utilizar queda a su criterio pero debe incluir el diagrama de actividad de cada uno. Posteriormente se recomienda que los validen con el cliente y a partir de este momento se procede a definir el modelo de datos del proyecto.	jueves, 17 de abril de 2025	jueves, 3 de abril de 2025	100.00%
	Reglas de negocio	Identificar y describir las reglas de negocio definidas por la organización que deben ser atendidas por el	jueves, 17 de abril de 2025	jueves, 3 de abril de 2025	100.00%

## 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.

### 1. Gestión de Importaciones de Datos

#### 1.1. Automatización de recolección de comentarios

El sistema utiliza la aplicación en Python llamada *TwenBFive Editor* para realizar el formateo automático de comentarios, eliminando la necesidad de utilizar programas externos, los archivos generados tendrán el formato CSV.

Aceptado ☐ Rechazado ☐

#### 1.2. Revisión manual de comentarios

El sistema de *TwenBFive Editor* debe apartar los registros que no pudieron ser procesados automáticamente en un archivo en formato CSV para su posterior revisión manual.


Aceptado ☐ Rechazado ☐

#### 1.3. Importación a base de datos










El sistema debe importar los datos en formato CSV recolectados automáticamente a la base de datos para su almacenamiento y posterior procesamiento.

Aceptado ☐ Rechazado ☐

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).




**JOSE DOMINGO VICTORIA RODRIGUEZ**



Para: Oswaldo Isaías Hernández Santes; [marisolgarsan99@gmail.com](mailto:marisolgarsan99@gmail.com)

Mié 28/05/2025 8:13

CC: Marco Iván Flores Villanueva

**filter683718fc6da28\_csv.csv**

424 KB


No suele recibir correo electrónico de [jvictoria01@alumnos.uaq.mx](mailto:jvictoria01@alumnos.uaq.mx). [Por qué es esto importante](#)


**Buen día Oswaldo y equipo Veinte-25.**

Les mando en el siguiente correo un formato csv de unos de las empresas con las que trabajamos. Este formato se encuentra sin filtrar los códigos, es decir, se encuentra sin alteraciones, así como nos lo da el formato original.










Sin más que decir nos despedimos, esperando les sea de ayuda, de igual manera pueden decirnos si tienen alguna duda.

**Saludos**







**MARISOL ARCIAG**




Para: Oswaldo Isaías Hernández Santes










Jue 05/06/2025 12:09








**MARISOL ARCIAG**



Para: Oswaldo Isaías Hernández Santes

Jue 05/06/2025 12:16

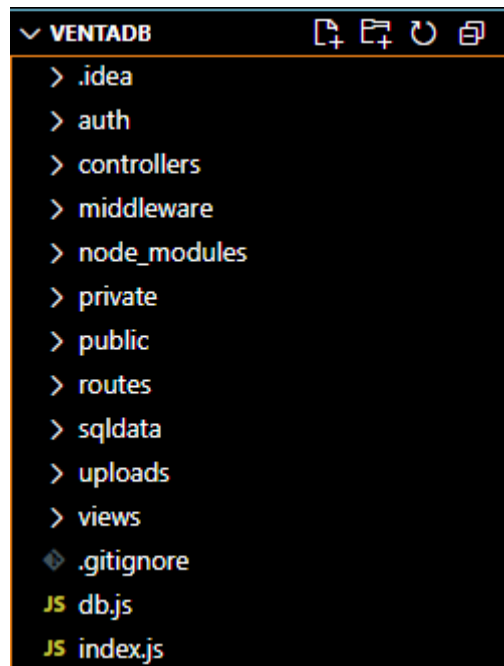
**1-Alondra Prospero Amaya.p...**

173 KB



### 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:



### 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.

```

<nav class="fixed-top navbar navbar-expand-md navbar-dark align-items-center justify-content-between px-4 py-4">
  <a class="navbar-brand fw-bold" href="/dashboard">VentaDB</a>
  <div class="collapse navbar-collapse">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item dropdown">
        <a class="nav-link dropdown-toggle">Menú</a>
        <div class="dropdown-menu">
          <a class="dropdown-item" href="/scrapping">App</a>
          <a class="dropdown-item" href="/clientes">Clientes</a>
          <a class="dropdown-item" href="/dashboard">Dashboard</a>
          <a class="dropdown-item" href="/ventas">Ventas</a>
          <a class="dropdown-item" href="/editarusuario">Gestión de Usuarios</a>
          <a class="dropdown-item" href="/logs">Registros/Logs</a>
          <a class="dropdown-item" href="/logout">Cerrar sesión</a>
        </div>
      </li>
    </ul>
  </div>
</nav>

```

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

```

<!DOCTYPE html>
<html lang="es">
<head>
  <%- include('includes/head.ejs') %>
</head>
<body class="bg-custom">
  <%- include('includes/nav.ejs') %>
<main role="main">
  <div class="container">
    <div class="container-fluid">
      <!--Primera fila-->
      <div class="d-sm-flex align-items-center justify-content-between mb-4">
        <h2 class="text-accent mb-0">Dashboard</h2>

```

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 cual es integrado para peticiones asíncronas y así no interferir con los demás procesos de la página.

```

<script>
  // Función para cargar archivos
  async function loadFiles() {
    const loadingEl = document.getElementById('loading');
    const containerEl = document.getElementById('files-container');
    const noFilesEl = document.getElementById('no-files');

    // Mostrar loading
    loadingEl.style.display = 'block';
    containerEl.innerHTML = '';
    noFilesEl.style.display = 'none';

    try {
      const response = await fetch('/scraping/api/files');
      const data = await response.json();
    }
  }

```

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.

- ☐  Modificado ventadb/<files> Razón: Modelo de scraping modificado para solo aceptar .exe y .dmg  
#41 by taqueritospro was merged 1 hour ago

---

- ☐  Agrega ejecutables de TwenBFiveApp  
#40 by tuzobus was merged 1 hour ago

---

- ☐  Modificado ventadb/db.js Razón: Confidencial  
#39 by taqueritospro was merged 3 hours ago

---

- ☐  Modificado ventadb/<files> Razón: Se añadió función de ver logs de la base de datos (sólo para admin)  
#38 by taqueritospro was merged 3 hours ago

---

- ☐  Modificado ventadb/<files> Razón: Mejora de vistas, se añadió el sql de la base de datos (contiene creación de la base de datos, tablas, triggers)  
#37 by taqueritospro was merged 4 hours ago

---

- ☐  Mejora dashboard y errores en código  
#36 by tuzobus was merged 6 hours ago

---

- ☐  Arregla problemas con módulo de invitaciones y AJAX  
#35 by tuzobus was merged 8 hours ago

---

- ☐  Completa funcionalidad de editarusuario  
#34 by tuzobus was merged 12 hours ago

---

- ☐  Actualiza a AJAX y complementa módulo de creación de usuarios  
#33 by tuzobus was merged 13 hours ago

---

- ☐  Modificado ventadb/<files> Razón: Errores corregidos en registro de usuarios  
#32 by taqueritospro was merged 20 hours ago


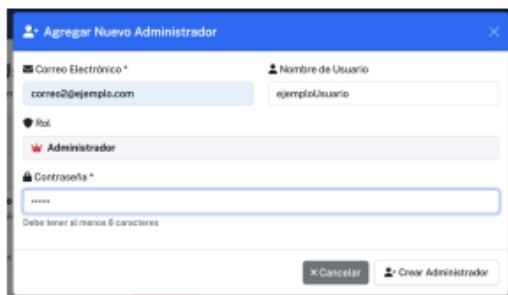
Siendo “tuzobus” mi compañero y “taqueritospro” yo.

## 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.

### *Caso de prueba 4*

El administrador añade un nuevo usuario de tipo administrador con un correo electrónico sin utilizar, pero con un usuario ya existente en la base de datos. Esto será permitido, pues los nombres de usuario no provocan conflictos a la hora de iniciar sesión (el inicio de sesión se lleva a cabo con el correo electrónico, el cual sí debe ser único).



ID	Usuario	Rol	Estado	Fecha Alta	Pa. Acciones
100	ejemplo.usuario	Administrador	Activo	10/10/2020	[iconos]
101	ejemplo.usuario	Administrador	Activo	10/10/2020	[iconos]

## STC0206 – Implantación de software



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).