Universidad de Ingenieria y Tecnologia
Departamento de Ciencias de la Computación

Ingeniería de Software I

Lab V: Design Part II
2024 - II

TEAM	**SegmentationFault**
INTEGRANTES	**Christopher Najarro Mancco** **Mauricio Salazar Hillenbrand** **Leonardo Matias Candio Ormeño** **Manyory Cueva Mendoza** **Angel Ulises Tito Berrocal**

Background:
Top-Down Design
Bottleneck
SPOF(single point of failure)
Scalability 101

(1 ptos) Evaluación Continua.

Preguntas

1. **Coloque el repo donde su team esta subiendo todas las tareas:**

https://github.com/utecsw20242/Autotarget.git

CASO DE ESTUDIO (level: intermediate):

Batcheros Inc. ha manejado por un periodo de tiempo considerable todos sus estados contables en archivos de texto planos en windows. Los archivos tienen la siguiente nomenclatura:

2024_09_EstadoContable.txt (cada archivo en promedio tienen 100k entradas)

Y el archivo tiene entradas como la siguiente:

Passive#Category#Amount#Currency

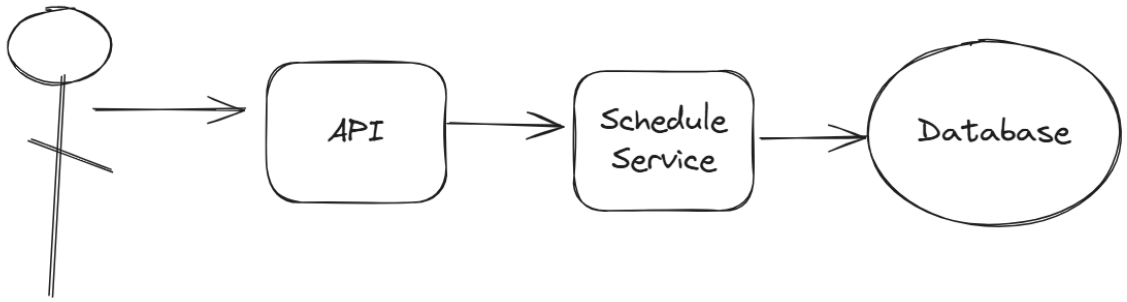
Por ejemplo:

Passive#Compras#20#Dolares

La moneda que manejan es soles y dolares.

Entonces la empresa decide contratar a la UTEC para que les ayude con este problema.

Actualmente ellos tienen un sistema que usan para registrar horas de sus empleados:



La base de datos que tienen actualmente es una instancia única de sqlite que la limpian cada 5 meses.

Problema:

- Identificar quienes son los usuarios si es que los hay.
- Identificar los posibles use-cases.
- Diseñar usando la técnica top-down todos los componentes/servicios/apis/base de datos que usarán para soportar el upload masivo de la data de 10 años.
    - Hacer una solución básica.
    - Escalar la solución de modo que ellos puedan registrar horas y el proceso de upload se haga en paralelo en días de oficina. Por razones de logística desconocidas no se puede hacer el proceso ni en fin de semana ni fuera de hora

de trabajo. Ya que el contador quisiera validar usando el sistema que todo esté OK.

- ¿Qué tablas se usarán ? Defina un bosquejo de todas las tablas y sus relaciones.
- Hacer un pseudocódigo como se llamaría el proceso de batch.

- **Identificar quienes son los usuarios si es que los hay.**

  Contador: Persona encargada de validar y revisar que todo el proceso de carga masiva de datos esté correcto. Este usuario necesita acceso para realizar validaciones y revisiones.
  Empleados: Usuarios del sistema que registran sus horas de trabajo.

- **Identificar los posibles use-cases:**

  Carga Masiva de Archivos Contables: Subir y procesar los archivos de los últimos 10 años.
  Validación de Carga: El contador valida que los datos cargados sean correctos.
  Registro de Horas: Los empleados registran sus horas de trabajo en el sistema.
  Paralelización de Procesos: Mientras se realiza la carga masiva, el registro de horas debe continuar funcionando sin interrupciones.

- **Diseñar usando la técnica top-down todos los componentes/servicios/apis/base de datos que usarán para soportar el upload masivo de la data de 10 años:**

  Niveles:

  1. **Componentes principales del sistema:**
     Frontend: Interfaz que permite a los usuarios interactuar con el sistema. Aquí los empleados registran horas y el contador valida el estado de la carga masiva.

     API Gateway: Intermediario entre el frontend y los servicios backend. Maneja las solicitudes y redirige a los servicios adecuados.

     Backend: Servicios responsables de realizar tareas como el procesamiento de archivos contables, registro de horas, y validación de datos.

     Base de Datos: Almacena los registros contables, horas trabajadas y el estado de validación de los archivos.

  2. **Backend - Servicios Principales:**

     a. **Servicio de Carga Masiva:**
     Este servicio se encarga de procesar los archivos contables, dividiendo los datos en bloques manejables (por ejemplo, de 10,000 entradas) y almacenándolos en la base de datos.
     - **Componentes:**
       - **División de archivos**: Divide los archivos grandes en fragmentos.

- **Conversión de moneda**: Convierte los valores en dólares a soles cuando sea necesario.

- **Validación de formato**: Revisa que los archivos sean válidos antes de procesarlos.

b. Servicio de Registro de Horas:

Permite a los empleados registrar sus horas de trabajo sin ser afectados por el procesamiento de los archivos contables.

- **Componentes:**
  - **Registro**: Almacena las horas trabajadas por los empleados.

  - **Validador de Duplicados**: Verifica que no haya registros duplicados.

c. Servicio de Validación Contable:

Permite al contador revisar y validar los datos procesados para asegurar que todo esté correcto.

- **Componentes:**
  - **Panel de Validación**: Muestra los archivos procesados para revisión.

  - **Confirmación de validación:** El contador puede marcar los archivos como validados o rechazar registros incorrectos.

3. **Base de Datos:**
   a. **Tabla de Registros Contables (`Registros_Contables`):**
      - **Campos:**
        - **`id_registros`: Identificador único del registro.**
        - **`tipo`: Tipo de cuenta (Pasivo/Activo).**
        - **`categoria`: Categoría del registro.**
        - **`monto`: Monto registrado.**
        - **`moneda`: Moneda del registro (Soles/Dólares).**
        - **`fecha`: Fecha del registro.**

   b. **Tabla de Monedas (`Monedas`)**
      - **Campos:**
        - **`id_monedas`: Identificador único de la moneda.**
        - **`moneda`: Tipo de moneda (Soles/Dólares).**
        - **`tasa_conversion`: Tasa de conversión a soles.**

c. **Tabla de Horas de Trabajo (`Horas_Trabajo`)**
   - **Campos:**
     - **`id_horas`: Identificador único del registro de horas.**
     - **`empleado_id`: Relación con el empleado.**
     - **`fecha`: Fecha del registro.**
     - **`horas_trabajadas`: Cantidad de horas trabajadas.**

d. **Tabla de Validaciones Contables (`Validaciones_Contables`)**
   - **Campos:**
     - **`id_validaciones`: Identificador único de la validación.**
     - **`registro_id`: Relación con el registro contable.**
     - **`validado`: Indica si fue validado (booleano).**
     - **`fecha_validacion`: Fecha de la validación.**

e. **Empleados (`Empleados`)**
   - **Campos:**
     - **`id_empleados`: Identificador único del empleado.**
     - **`nombre`: Nombre del empleado.**
     - **`cargo`: Cargo o posición que ocupa el empleado.**
     - **`fecha_contratacion`: Fecha en la que el empleado fue contratado.**

- **Solución Básica:**

  Objetivos:
  - El sistema debe ser capaz de procesar los archivos de 10 años de manera eficiente.
  - Los empleados pueden registrar sus horas de trabajo mientras se realiza la carga.
  - El contador debe poder validar los archivos cargados.

  Flujo de Trabajo:
  - **Carga de Archivos:** Los archivos de datos contables se suben al sistema y se dividen en bloques de 10,000 entradas. Estos bloques se procesan uno a uno, validando los datos y convirtiendo las monedas cuando sea necesario.
  - **Registro de Horas:** Los empleados registran sus horas mediante una API independiente, la cual no está afectada por la carga masiva.
  - **Validación Contable:** El contador puede verificar los archivos procesados en un panel de validación, asegurando que los datos estén correctos.

- **Escalabilidad de la Solución:**

  Escalar el sistema para manejar grandes volúmenes de datos y asegurar el funcionamiento en paralelo con el registro de horas.

  - **Procesamiento en Paralelo:** El servicio de carga masiva puede escalarse para procesar varios bloques de archivos en paralelo utilizando colas de trabajo. Esto permitirá que múltiples bloques de datos se procesan simultáneamente, mejorando el rendimiento general.

  - **Scheduler para Restricción de Tiempo:** Implementar un scheduler que gestione el tiempo de procesamiento:
    - Los archivos solo se procesan dentro del horario laboral (por ejemplo, de 9 AM a 6 PM).
    - El procesamiento se detiene fuera de este horario y los fines de semana.
    - Las cargas que no se completan en un día laboral pueden continuar el próximo día.

  - **Separación de Servicios:** Para evitar que el registro de horas se vea afectado por la carga masiva:
    - El Servicio de Registro de Horas debe estar separado lógicamente del Servicio de Carga Masiva, cada uno con sus propias bases de datos y recursos de procesamiento.
    - Esto asegura que ambos procesos puedan ejecutarse en paralelo sin impactar mutuamente.

  - **Tolerancia a Fallos y Respaldo:** El sistema debe ser resistente a fallos, por lo que debe incluir:
    - **Backup y Recuperación**: En caso de fallo en el procesamiento o en la base de datos, se debe poder recuperar fácilmente los registros procesados.
    - **Monitorización y Logs**: Implementar un sistema de monitorización que permita al contador y al administrador ver el estado del procesamiento, los errores ocurridos y los registros validados.

- **Bosquejo de Base de Datos:**

- **Hacer un pseudocódigo como se llamaría el proceso de batch:**

Algorithm 1 Procesamiento en batch de archivos contables

```
function PROCESAR_ARCHIVO_BATCH(archivo)
    bloques ← DIVIDIR_ARCHIVO_EN_BLOQUES(archivo, 10000)
    for cada bloque en bloques do
        for cada linea en bloque do
            (tipo, categoria, monto, moneda) ← PARSEAR_LINEA(linea)
            monto_en_soles ← CONVERTIR_MONEDA(monto, moneda)
            GUARDAR_REGISTRO_CONTABLE(tipo, categoria, monto_en_soles)
        end for
    end for
    NOTIFICAR_CONTADOR("Archivo procesado correctamente.")
end function
function DIVIDIR_ARCHIVO_EN_BLOQUES(archivo, tamano_bloque)
    abrir archivo
    while archivo no esté vacío do
        bloque ← leer_lineas(tamano_bloque)
        if bloque está vacío then
            break
        end if
        return bloque
    end while
end function
```