Proyecto Integrador: Sistema Automatizado de Facturación y Gestión de Usuarios para Tiendas Minoristas

Una cadena nacional de tiendas minoristas está modernizando sus operaciones de TI. En particular, busca automatizar la **generación y distribución de facturas electrónicas**, así como facilitar la **gestión de usuarios temporales** durante temporadas altas de ventas. Actualmente, el proceso es manual, propenso a errores, y consume recursos humanos que podrían dedicarse a tareas de mayor valor.

Para resolver esto, se ha diseñado un **sistema automatizado y modular** que aprovecha diversas tecnologías de scripting para simular un entorno real de trabajo que combina:

- Automatización de procesos con Bash y PowerShell,
- Procesamiento y simulación de datos con Python,
- Plantillas profesionales en LaTeX para facturación,
- Correos automáticos con archivos adjuntos,
- Control de errores y registros centralizados.

Objetivo General

Desarrollar un sistema automatizado y modular que simule la operación de facturación y gestión de empleados de una cadena de tiendas, usando herramientas de scripting para sustituir campos dinámicos en documentos LaTeX, generar compras y usuarios automáticamente, enviar correos, y ejecutar procesos periódicos con control de errores y logging.

Flujo General del Sistema

1. **Simulación de Compras (Python)**

   Un script llamado generador_compras.py simula transacciones comerciales generadas por clientes, utilizando la librería Faker. Los datos generados incluyen:

   - Nombre del cliente
   - Ciudad y dirección
   - Correo electrónico y teléfono
   - IP de compra
   - Monto total
   - Modalidad de pago (completo o fraccionado)
   - Estado del pago (exitoso o fallido)
   - Timestamp de la transacción

     Los registros se almacenan como archivos .csv, uno por cada lote simulado.

     También se generan errores aleatorios para simular casos de fallos.

2. **Generación de Facturas (Bash + sed/awk + LaTeX)**

   - Se toma un **template .tex** que contiene placeholders como {nombre}, {ciudad}, {monto}, {direccion}, {correo}, etc.
   - El script generador_facturas.sh sustituye esos campos con los datos del archivo .csv, usando sed.
   - Se compila automáticamente el documento con pdflatex. En caso de error, se revisa el .log generado buscando líneas con "!".
   - Se generan archivos .pdf por cliente.
   - Se crea un log individual por factura y un log centralizado con resumen diario, almacenado en log_diario.log.
   - Al finalizar el día, este log es enviado automáticamente al administrador por correo electrónico usando el script de envío.
   - Además, se genera un archivo pendientes_envio.csv en el siguiente formato:

factura_1234.pdf,cliente@correo.com
factura_5678.pdf,otro@correo.com

Este archivo servirá como base para el envío por correo en el paso siguiente.

3. **Envío de Correos (Python)**
    - A la hora siguiente (configurada mediante cron o Task Scheduler), se ejecuta enviador.py.
    - Este script abre el archivo pendientes_envio.csv, recorre cada línea, adjunta el PDF correspondiente y lo envía al correo indicado.
    - Valida correos por medio de expresoines regulares y maneja errores de envío, dejando un log por cada intento.
    - El script debe de eliminar las líneas exitosas del CSV para evitar reenvíos.

    - Debe de generar un archivo log_envios.csv en el siguiente formato:

      factura_1234.pdf,cliente@correo.com, exitoso
      factura_5678.pdf,otro@correo.com, fallido

4. **Automatización del Proceso (Bash + cron / Task Scheduler)**

    - Se configura una tarea en cron (Linux) o Task Scheduler (Windows) para ejecutar:
        - A la hora 1: generador_facturas.sh
        - A la hora 2: enviador.py
    - Al finalizar la ejecución diaria de enviador.py, se genera y envía automáticamente un reporte al administrador utilizando tambien log_diario.log.. Este reporte resume:

        - Total de correos procesados
        - Total vendido
        - Cuantos pedidos fueron pagados en su totalidad
        - Cuántos fueron exitosos y cuántos fallidos
        - Se genera usando awk sobre el archivo log_envios.csv

5. **Gestión de Usuarios Temporales (PowerShell)**

- Durante temporadas altas, se recibe un archivo diario empleados.csv (generado desde Python o manualmente), con los datos de los nuevos empleados temporales.
- El script usuarios.ps1 toma este archivo y crea cuentas de usuario en el sistema local con:
  - Nombre completo
  - Correo institucional
  - Contraseña aleatoria (que cumple con criterios de seguridad)
  - Privilegios administrativos
- Se guarda un log por cada usuario creado.

Importante: No utilice expresiones genéricas como s/{.*}/.../. Debe hacer la sustitución **campo por campo**, asegurándose de conservar intacta la sintaxis del documento .tex.

Estructura Modular

Módulo	Tecnología	Descripción
generador_compras.py	Python	Simula clientes y transacciones, exporta a CSV
plantilla_factura.tex	LaTeX	Documento base con campos reemplazables
generador_facturas.sh	Bash + sed/awk	Sustitución, validación, compilación PDF, crea CSV envío
pendientes_envio.csv	Bash	Lista generada con PDF y correos por enviar
enviador.py	Python	Envío de facturas por correo con validación
cron_job.sh	Bash	Automatiza la ejecución por hora/día
usuarios.ps1	PowerShell	Toma un CSV y crea cuentas locales con privilegios
log_diario.log	Bash/Python	Registro centralizado de errores y procesos

Lista de Placeholders a Sustituir

Placeholder en el .tex	Descripción del dato	Fuente esperada
`{id_transaccion}`	ID único de la transacción	Generado por `generador_compras.py`
`{fecha_emision}`	Fecha en que se genera la factura	Fecha actual o del timestamp
`{nombre}`	Nombre completo del cliente	CSV de compras
`{correo}`	Correo electrónico del cliente	CSV de compras
`{telefono}`	Número de teléfono del cliente	CSV de compras
`{direccion}`	Dirección física del cliente	CSV de compras
`{ciudad}`	Ciudad del cliente	CSV de compras
`{cantidad}`	Cantidad de productos comprados	CSV de compras
`{monto}`	Monto total en colones	CSV de compras
`{pago}`	Tipo de pago (completo o fraccionado)	CSV de compras
`{estado_pago}`	Estado del pago (exitoso o fallido)	CSV de compras
`{ip}`	IP desde la que se realizó la compra	CSV de compras
`{timestamp}`	Fecha y hora de la transacción	CSV de compras
`{observaciones}`	Texto opcional (ej: cliente frecuente, promoción aplicada)	Puede ser fijo o calculado

Reglamento del Proyecto Integrador

1. Cantidad de Integrantes

- **Equipos de 2 a 3 integrantes máximo.**
    - Permite distribuir eficientemente las responsabilidades del proyecto (simulación, automatización y scripting cruzado).
    - ***No se permiten equipos individuales*** **ni de más de 3 personas.**

- **Roles sugeridos (pero no fijos)**

    - **Integrante A:** Simulación y generación de datos con Python.
    - **Integrante B:** Automatización y scripting Bash + LaTeX.
    - **Integrante C:** Automatización en PowerShell y validación cruzada.

Todos deben conocer el flujo completo.

2. Entregables

Cada equipo deberá presentar los siguientes productos:

- **Código fuente funcional y organizado** de todos los módulos: *.py, *.sh, *.ps1, *.tex.
- **Al menos 3 logs** generados automáticamente por ejecución del sistema, incluyendo:
    - Log diario central (log_diario.log)
    - Log de envíos de correo (log_envios.csv)
    - Log de creación de usuarios (fecha, usuario, estado)
- **Capturas de pantalla o evidencia** verificable de:
    - Correos enviados exitosamente
    - Cuentas de usuario creadas localmente en el sistema operativo
- **Documento tipo reporte** en PDF, que debe incluir:

    1. **Portada** con nombre del proyecto, integrantes y fecha
    2. **Introducción** general al problema y solución propuesta

3. **Arquitectura implementada**, con diagrama y descripción de componentes
4. **Explicación detallada del desarrollo**, con evidencias (fragmentos de código, capturas de ejecución)
5. **Instrucciones para ejecutar el sistema** (dependencias, comandos, orden sugerido)
6. **Conclusión**: aprendizajes, evaluación general del proyecto
7. **Problemas encontrados y cómo fueron resueltos**, destacando decisiones técnicas clave

4. Evaluación

Criterio de Evaluación	*Valor Máximo*
1. Estructura general del sistema (integración y coherencia)	10 pts
- Flujo entre scripts, claridad del proceso y correcta automatización	
2. Código fuente funcional	20 pts
- Correcta implementación y ejecución de `generador_compras.py`	4 pts
- Funcionamiento y lógica de `generador_facturas.sh`	4 pts
- Sustituciones y compilación LaTeX con logs válidos	4 pts
- `enviador.py` con validaciones, envíos y limpieza de CSV	4 pts
- `usuarios.ps1` funcional, con contraseñas seguras y logs	4 pts
3. Automatización programada	5 pts
- Uso correcto de `cron` o `Task Scheduler`	3 pts
- Generación de reportes automatizados y envío al administrador	2 pts
4. Logs y evidencias de ejecución	10 pts
- Log diario funcional y comprensible (`log_diario.log`)	3 pts
- Log de correos enviados (`log_envios.csv`)	3 pts
- Log de creación de usuarios	2 pts
- Evidencia (capturas) de envíos y usuarios	2 pts
5. Documento tipo reporte (calidad y contenido)	30 pts
- Portada, introducción, arquitectura bien explicada	5 pts
- Explicación clara de lo desarrollado con evidencia. Sección de apéndice con los codigos implementados y los logs.	15 pts
- Instrucciones de ejecución completas y funcionales	4 pts
- Conclusión con reflexiones claras	3 pts
- Resolución de problemas bien documentada	3 pts
7. Calidad técnica y buenas prácticas de programación	15 pts
- Código claro, modular, bien documentado	6 pts

- Uso apropiado de cada tecnología según su propósito	6 pts
- Manejo de errores y validaciones	3 pts
Total	**100 pts**

5. Prohibiciones

- Penalizaciones por código quemado, no modular o poco profesional.
- No se permite el uso de herramientas gráficas para enviar correos o crear usuarios.
- No se permite sustituir el uso de LaTeX por editores de texto enriquecido.
- No se permite hardcodear rutas o datos sensibles.

Instrucciones Finales para el Reporte

Entrega Única: Documento PDF Integrado

- El **documento PDF será el único material que se revisará**.
- **Todo el proyecto debe estar contenido dentro del documento**.
- El PDF debe incluir:
    - Todo el código fuente con etiquetas claras por archivo.
    - Al menos uno de los logs generados.
    - Todas las capturas de pantalla requeridas.
    - Explicaciones técnicas, resultados, errores y reflexiones.

Reglas para incluir el código

- Todo el **código fuente debe estar contenido en una sección específica de Apéndice** al final del documento.
- Si el código es utilizado o referenciado durante el desarrollo del documento (por ejemplo, para explicar una decisión técnica o lógica implementada), se debe indicar claramente a qué parte del **Apéndice** corresponde.
- Cada archivo debe estar claramente indicado con un encabezado como:

  ### Código fuente: generador_compras.py

  seguido por el contenido completo del script con formato de bloque de código.

- Lo mismo aplica para .sh, .ps1, .tex, archivos .csv relevantes (fragmentos representativos), y cualquier otro necesario para comprender el sistema.
- En caso de que un script o archivo **no esté bien identificado o sea ilegible**, se considerará como **no entregado**.