ASISTENTE Y ORQUESTADOR INTELIGENTE DE VERSIONES

1. 🧭 PRODUCCIÓN

🎯 Objetivo: preparar carpetas con la versión, release, roadmap y correo de solicitud.

📚 Actividades:

1. Crear carpeta raíz de versión.

📥 Parámetros:

- **Ruta OneDrive**: `D:\OneDrive\Entregas` (o el path corporativo).

  La carpeta debe quedar generada dependiendo del nombre del cliente y versión. Por ejemplo: ENLACEAV2.0.1_250619.

2. Crear subcarpeta "FIRMA".

📥 Parámetros:

- **Cliente**: `ATC` (ejemplo)
- **Nombre de versión para el cliente:** `ENLACEAV` (ejemplo)
- **Terminal:** NEW6260 (ejemplo)
- **Versión base**: `2.0.1` (ejemplo de entrega oficial)
- **Versión de aumento**: `2.0.2` (ejemplo para versión para probar descarga remota)
- **Build** (AAMMDD): `250619` *(ejemplo)*
- **Ruta de compilación:** path desde donde se extraen los `.bin`

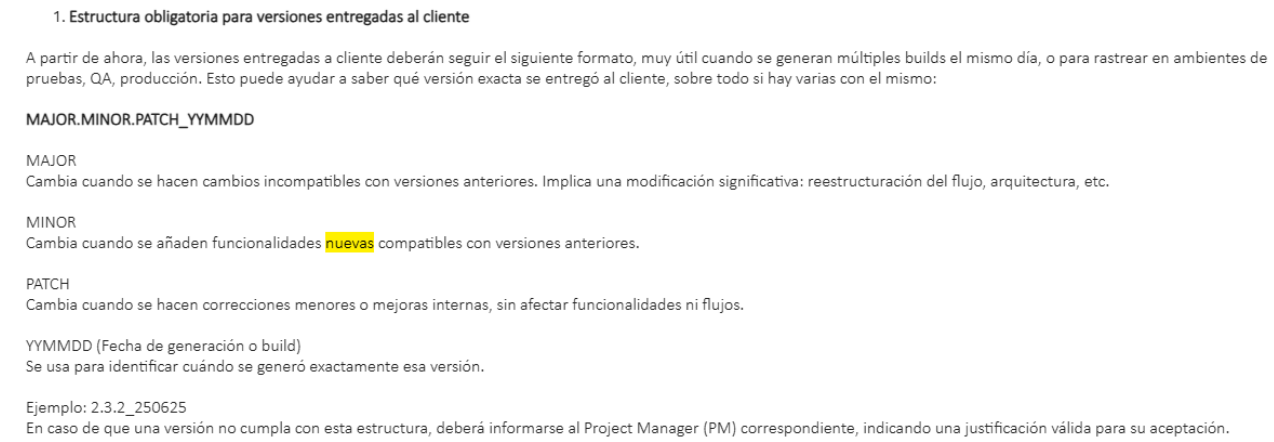
> ⚠️ **Nota:** Los datos mostrados en el documento son **solo de referencia**.

```
En ejecución real, el programa debe solicitar al usuario estos valores para evitar errores y asegurar que la versión y fecha correspondan a la entrega actual.
```

📏 Regla de nombres

{NOMBRE*VERSIÓN_CLIENTE}{VERSIÓN_BASE}* {BUILD}

Ejemplo: ENLACEAV2.0.1_250619



1. **Estructura obligatoria para versiones entregadas al cliente**

A partir de ahora, las versiones entregadas a cliente deberán seguir el siguiente formato, muy útil cuando se generan múltiples builds el mismo día, o para rastrear en ambientes de pruebas, QA, producción. Esto puede ayudar a saber qué versión exacta se entregó al cliente, sobre todo si hay varias con el mismo:

MAJOR.MINOR.PATCH_YYMMDD

MAJOR
Cambia cuando se hacen cambios incompatibles con versiones anteriores. Implica una modificación significativa: reestructuración del flujo, arquitectura, etc.

MINOR
Cambia cuando se añaden funcionalidades nuevas compatibles con versiones anteriores.

PATCH
Cambia cuando se hacen correcciones menores o mejoras internas, sin afectar funcionalidades ni flujos.

YYMMDD (Fecha de generación o build)
Se usa para identificar cuándo se generó exactamente esa versión.

Ejemplo: 2.3.2_250625
En caso de que una versión no cumpla con esta estructura, deberá informarse al Project Manager (PM) correspondiente, indicando una justificación válida para su aceptación.

📂 Estructura esperada dentro de FIRMA

1. **Carpeta Versión Base**
   - Nombre carpeta: {NOMBRE_VERSIÓN_CLIENTE}{VERSIÓN_BASE}
     - Ejemplo: `ENLACEAV2.0.0`
   - Contiene:
     - **Archivo binario:** Copia directa desde la **ruta de compilación**.
       El archivo se deja tal cual sale del proceso de compilación, sin cambios de nombre ni contenido. Ejemplo: ATC.bin
2. **Carpeta Versión Aumento**
   - Nombre carpeta: {NOMBRE_VERSIÓN_CLIENTE}{VERSIÓN_AUMENTO}
     - Ejemplo: `ENLACEAV2.0.2`
   - Contiene:
     - **Archivo binario:** Copia directa desde la **ruta de compilación**.
       El archivo se deja tal cual sale del proceso de compilación, sin cambios de nombre ni contenido. Ejemplo: ATC.bin
3. **Archivo `Checksums.txt`**
   Contiene el **hash MD5** de cada `.bin` incluido en las carpetas anteriores para validación de integridad.

   📄 **Formato de `Checksums.txt`**

   {NOMBRE_VERSIÓN_CLIENTE}{VERSIÓN_BASE}
   Checksum (MD5): <hash_base>

{NOMBRE_VERSIÓN_CLIENTE}{VERSIÓN_AUMENTO} - AUMENTO
Checksum (MD5): <hash_aumento>

Ejemplo con datos:
ENLACEAV2.0.0
Checksum (MD5): d41d8cd98f00b204e9800998ecf8427e

ENLACEAV2.0.1 - AUMENTO
Checksum (MD5): 098f6bcd4621d373cade4e832627b4f6

📌 **Nota de automatización**
Este proceso debe ser completamente automático:
1. Leer y editar el archivo de configuración del proyecto (`.h` o equivalente) para actualizar las macros de **versión** y build.
2. Compilar la versión base con los valores actualizados.
3. Incrementar automáticamente la versión con el valor de la versión base, editar nuevamente las macros pero ahora con la información ingresada de la versión de aumento y recompilar para generar la versión de aumento.
4. Copiar los binarios generados a sus carpetas correspondientes y generar el `checksums.txt`.
5. Validar integridad mediante comparación de hash MD5 local y en destino.

> 📌 **Notas importante**:

- Los `.bin` se copian inmediatamente después de cada build, asegurando que la carpeta de versión base y la de aumento contengan los archivos correctos.
- La versión de aumento siempre incrementa en **1** respecto a la versión base, tomando como referencia la posición que se modifique (mayor, menor o patch) según corresponda al cambio realizado.
- La distinción entre **versión base** y **versión aumento** depende de la compilación que se ejecute:
  - **Versión base:** Compilación tras ajustar el proyecto con la versión oficial de entrega (Compilar para generar la versión base y copiar a la carpeta base).
  - **Versión aumento:** Compilación tras realizar el incremento de versión (Realizar el incremento de versión en el archivo del proyecto, compilar nuevamente y copiar el archivo resultante a la carpeta de aumento).
- El incremento de versión en el archivo `.h` debe ser **automático** mediante script antes de compilar el bin de aumento.

- La carpeta completa se **sube a OneDrive** en la ruta de entregas definida, manteniendo nombres y estructura exactos.
- Tras la subida, validar integridad calculando MD5 del archivo en OneDrive y comparándolo con el local.
- Verificar **permisos de acceso** en OneDrive permitan a todo el equipo descargar y validar.

3. Crear y enviar correo para firma:
   🔹 **Parámetros requeridos:**

   - **Cliente**: `ATC` *(ejemplo)*
   - **Nombre de versión para el cliente**: `ENLACEAV` *(ejemplo)*
   - **Terminal/Dispositivo**: `NEW6260` *(ejemplo)*
   - **Versión base**: `2.0.1` *(ejemplo de entrega oficial)*
   - **Versión de aumento**: `2.0.2` *(ejemplo para pruebas adicionales, ej. descarga remota)*
   - **Build** (AAMMDD): `250619` *(ejemplo)*
   - **Ruta de compilación**: Path desde donde se extraen los `.bin`
   - **Tipo de firma**: Selección entre *Genérica* o *Personalizada*.
   - **CID**: Lista para elegir entre `0` *(default genérico)* o valor personalizado según cliente. Si es personalizada, desplegar campo para ingreso manual.
   - **Descripción breve de lo que se hizo**: Texto libre que incluya requerimientos, incidentes reportados por cliente/QA o bugs resueltos (para cuerpo del correo).

📦 Parámetros que el script obtiene automáticamente

(no se piden, se leen de la ruta y el proceso de FIRMA)

1. **Nombre del archivo** `.bin` – Se extrae tal cual de la carpeta de compilación.
2. **Checksum (MD5)** – Calculado en tiempo de ejecución y usado tanto para el `Checksums.txt` como para el correo.

📧 Formato del Asunto:
SOLICITUD DE FIRMA {NOMBRE*VERSIÓN_CLIENTE}{VERSIÓN_BASE}* {BUILD} {TERMINAL/DISPOSITIVO} | {CLIENTE}

📨 Formato del Cuerpo del correo:
Cordial saludo estimado/a @PERSONA,
Espero que te encuentres muy bien. Mediante el presente, es un placer informar que el

equipo de desarrollo POS para el proyecto {TERMINAL/DISPOSITIVO} ha resuelto satisfactoriamente {DESCRIPCIÓN*BREVE}. En este sentido, realizo la entrega formal de la versión {NOMBRE_VERSIÓN_CLIENTE}{VERSIÓN_BASE}* {BUILD} para su respectiva firma.

Detalles:
- {NOMBRE_VERSIÓN_CLIENTE}{VERSIÓN_BASE}
- Terminal: {TERMINAL/DISPOSITIVO}
- Tipo de firma: {TIPO_FIRMA}
- CID: {CID}
- Nombre del archivo: {NOMBRE_BIN}
- Checksum (MD5): {CHECKSUM_BASE}

- {NOMBRE_VERSIÓN_CLIENTE}{VERSIÓN_AUMENTO} - AUMENTO
- Terminal: {TERMINAL/DISPOSITIVO}
- Tipo de firma: {TIPO_FIRMA}
- CID: {CID}
- Nombre del archivo: {NOMBRE_BIN}
- Checksum (MD5): {CHECKSUM_AUMENTO}

> 📌 **Notas:**

- Los archivos `.bin` se **localizan automáticamente** desde la ruta de compilación indicada.
- El script obtiene de forma automática:
  - **Nombre del archivo** (exactamente como lo genera el compilador, sin renombrar).
  - **Checksum (MD5)** correspondiente.
- Una vez recopilada esta información, se debe **generar y enviar el correo de solicitud de firma** para recibir el bin firmado y continuar con el siguiente paso del flujo de entrega.

4. Generar Release Note para el cliente:
   En lo que se realiza el **proceso de firma** de la versión, es posible avanzar de forma paralela con la **generación del Release Note** y la actualización de las actividades de **producción** y **desarrollo**. El Release Note productivo o a entregar es word y pdf.

📩 Parámetros requeridos para Release Note

- **Cliente** → ya definido en pasos anteriores (ej. ATC).
- **Nombre de versión para el cliente** → ya definido en pasos anteriores (ej. ENLACEAV).
- **Versión nueva** (Versión que se entrega) → se obtiene de la carpeta **FIRMA**.
- **Versión base** (la versión anterior desde la que se desarrolló la nueva) → ingresar manualmente.
- **Fecha de entrega** → Tomar de la **Build** y formatear como dd/mm/aa.
- **Ambiente** → ingresar manualmente (`CERTIFICACIÓN` por defecto; seleccionar otro si aplica).
- **IP** → ingresar manualmente.
- **Puerto de descarga** → ingresar manualmente.
- **Puerto de inicialización** → ingresar manualmente.
- **Implementaciones y/o ajustes** → lista de tickets, incidentes o bugs QA corregidos (puede ingresarse manualmente o cargarse automáticamente desde commits o tablero de Kanban - Bitbucket).
- **Consideraciones** → texto libre con notas relevantes de la versión (ingresar manualmente).

📝 Lo que se debe hacer:

- **Generar el Release Note automáticamente**:
  - El sistema/script debe acceder a los **commits** asociados a la versión en desarrollo y extraer:
    - Lista de tickets, incidentes y bugs QA corregidos en cierto tiempo o rango de fecha.
  - Este proceso debe requerir **mínima o nula intervención humana**, limitándose a la verificación del contenido extraído.
  - La **versión nueva** se toma de la carpeta **FIRMA** y la **versión base** se ingresa manualmente.
- **Actualizar los documentos de producción y desarrollo**:
  - En producción, preparar la carpeta con los artefactos oficiales.
  - En desarrollo, actualizar la Wiki, roadmap y demás elementos del proyecto.
- **Verificar integridad y coherencia de la información**:
  - Confirmar que los datos extraídos de commits coinciden con los cambios efectivamente liberados.

- Validar que las consideraciones y parámetros manuales estén completos (IP, puertos, ambiente, notas).

💡 **Meta del flujo:**
Reducir la generación del Release Note a un proceso principalmente **automático**, donde la intervención humana se limite a revisar y aprobar el documento, sin tener que redactar o recopilar información manualmente. El Release productivo solo muestra la versión que se entregará.

5. Crear subcarpeta "CERTIFICACIÓN":

📥 Parámetros requeridos

- **Cliente** → ya definido en pasos anteriores (ej. ATC).
- **Nombre de versión para el cliente** → ya definido en pasos anteriores (ej. ENLACEAV).
- **Versión base** (entrega oficial).
- **Versión de aumento** (incrementada +1 sobre la base).
- **Build** (AAMMDD) → ya definida.
- **CID** → genérico (0) o personalizado.
- **Ruta de compilación** → desde donde se descargan los `.bin` y `.pkg` firmados.

🛠️ Pasos del proceso:

1. **Recibir y descargar el `.zip` de firma**
   - El `.zip` con las versiones firmadas llegan en un hilo de correo.
   - Descargar, descomprimir y verificar que contenga los `.bin` firmados correctamente.
2. **Preparar los `.pkg`**
   - Abrir el `.pkg` de la versión anterior con la herramienta correspondiente (Download Tools).
   - Cargar el .bin firmado correspondiente.
   - Cargarlo en la ruta que corresponda (Base o aumento).
3. Renombrar los `.bin` para descarga remota:
   - El `.bin` se toma tal cual viene firmado.
   - Renombrarlo al formato `{VERSIÓN SIN PUNTUACIÓN}{CID}.bin` → Ejemplo: `20149212.bin`.
   - Guardar y subir en la carpeta correspondiente en OneDrive (base o aumento).

📂 Estructura esperada dentro de CERTIFICACIÓN

4. **Carpeta Versión Base**
   - **Nombre de carpeta:** `{NOMBRE_VERSIÓN_CLIENTE}{VERSIÓN_BASE}
     - Ejemplo: `ENLACEAV2.0.0
   - **Contenido:**
     - **Archivo binario** → renombrado con el formato `{VERSIÓN SIN PUNTUACIÓN}{CID}.bin`
       - Ejemplo: `20149212.bin`
     - **Archivo** `.pkg` → nombrado con el formato `{NOMBRE_VERSIÓN_CLIENTE}{VERSIÓN_BASE}_{BUILD}.pkg`
       - Ejemplo: `ENLACEAV2.0.0_250619.pkg`
5. **Carpeta Versión de Aumento**
   - **Nombre de carpeta:** `{NOMBRE_VERSIÓN_CLIENTE}{VERSIÓN_AUMENTO}_{BUILD} - AUMENTO`
     - Ejemplo: `ENLACEAV2.0.1_250619 - AUMENTO`
   - **Contenido:**
     - **Archivo binario** → renombrado con el formato `{VERSIÓN SIN PUNTUACIÓN}{CID}.bin`
       - Ejemplo: `20149213.bin`
     - **Archivo** `.pkg` → nombrado con el formato `{NOMBRE_VERSIÓN_CLIENTE}{VERSIÓN_AUMENTO}_{BUILD}.pkg`
       - Ejemplo: `ENLACEAV2.0.1_250619.pkg`
6. **Archivo** `Checksums.txt`
   Contiene el **hash MD5** de cada `.bin` incluido en las carpetas anteriores para validación de integridad.

   📄 **Formato de `Checksums.txt**
   {NOMBRE_VERSIÓN_CLIENTE}{VERSIÓN_BASE}
   Checksum (MD5) del archivo binario: <hash_base>
   Checksum (MD5) del archivo pkg: <hash_base>

   {NOMBRE_VERSIÓN_CLIENTE}{VERSIÓN_AUMENTO} - AUMENTO
   Checksum (MD5) del archivo binario: <hash_aumento>
   Checksum (MD5) del archivo pkg: <hash_aumento>

> 📌 **Notas:**

- Siempre deben existir **dos** `.pkg` : uno de **versión base** y otro de **versión de aumento** (Cada uno en su carpeta correspondiente).
- Los nombres de `.pkg` y `.bin` son los únicos cambios permitidos; el contenido debe permanecer igual al recibido en el `.zip` .
- El MD5 debe calcularse sobre los archivos **ya renombrados** para garantizar coincidencia con lo entregado.

6. **CORREO CERTIFICACIÓN:**

📩 Parámetros requeridos para correo de Certificación

Ingreso manual (usuario)

- **Cliente** → ej. `ATC` .
- **Nombre de versión para el cliente** → ej. `ENLACEAV` .
- **Terminal/Dispositivo** → ej. `NEW6260` .
- **Tipo de firma** → Genérica o personalizada - Hacer seleccionable o lista.
- **CID** → Si se selecciona firma genérica `0` por default o valor personalizado según cliente.
- **Descripción breve de lo que se hizo** → Igual que en el correo de firma (requerimientos, incidentes, bugs resueltos).
- **Links OneDrive** → enlace a la carpeta de la versión a entregar (Carpeta Raíz) y el link del Roadmap productivo cargado.
- **Captura de evidencia** → imagen o link del Bitbucket con la versión entregada.

Obtenido automáticamente por el script

- **Versión base** → desde carpeta FIRMA.
- **Versión de aumento** → desde carpeta FIRMA.
- **Nombre de archivo** `.bin` → leído desde carpeta CERTIFICACIÓN (ya renombrado).
- **Nombre de archivo** `.pkg` → leído desde carpeta CERTIFICACIÓN.
- **Checksum (MD5)** del `.bin` y `.pkg` → calculado en ejecución.

📨 **Asunto**
SOLICITUD DE CERTIFICACIÓN {NOMBRE_VERSIÓN_CLIENTE}{VERSIÓN_BASE} | {PROYECTO} {TERMINAL/DISPOSITIVO}

📨 **Cuerpo del correo**

Cordial saludo estimado/a @PERSONA,

Espero que te encuentres muy bien. Mediante el presente, es un placer informar que el equipo de desarrollo POS para el proyecto {TERMINAL/DISPOSITIVO} ha resuelto satisfactoriamente {DESCRIPCIÓN*BREVE}. En este sentido, realizo la entrega formal de la versión {NOMBRE_VERSIÓN_CLIENTE}{VERSIÓN_BASE}* {BUILD].

Detalles:

- VERSIÓN {NOMBRE_VERSIÓN_CLIENTE}{VERSIÓN_BASE}
- Terminal: {TERMINAL/DISPOSITIVO}
- Tipo de firma: {TIPO_FIRMA}
- CID: {CID}
- Nombre del archivo .bin: {NOMBRE_BIN_BASE}
- Checksum (MD5) archivo .bin: {CHECKSUM_BIN_BASE}
- Nombre del archivo .pkg: {NOMBRE_PKG_BASE}
- Checksum (MD5) archivo .pkg: {CHECKSUM_PKG_BASE}

- **VERSIÓN DE AUMENTO** {NOMBRE_VERSIÓN_CLIENTE}{VERSIÓN_AUMENTO}
- Terminal: {TERMINAL/DISPOSITIVO}
- Tipo de firma: {TIPO_FIRMA}
- CID: {CID}
- Nombre del archivo `.bin` : {NOMBRE_BIN_AUM}
- Checksum (MD5) archivo `.bin` : {CHECKSUM_BIN_AUM}
- Nombre del archivo `.pkg` : {NOMBRE_PKG_AUM}
- Checksum (MD5) archivo `.pkg` : {CHECKSUM_PKG_AUM}

- **Versión:** {LINK_ONEDRIVE_VERSIÓN}
- **Roadmap:** [LINK_ONEDRIVE_ROADMAP]
- **Evidencia:** {CAPTURA_BITBUCKET}

NOTAS:

📌 **Notas clave:**

- La **descripción breve** es idéntica a la usada en el correo de firma.
- El script reutiliza la información obtenida en FIRMA y la complementa con la que se lee desde CERTIFICACIÓN.

- El envío se realiza únicamente cuando ambos `.pkg` (base y aumento) están listos y verificados.

7. **HERRAMIENTAS Y ESTRUCTURA DEL ONEDRIVE:**

📂 Estructura Genérica de OneDrive – Proyecto

{CLIENTE}/
├─ {PROYECTO}/
│ ├─ {NOMBRE*VERSIÓN*}{BUILD}/
│ │ ├─ FIRMA/
│ │ │ ├─ {NOMBRE*VERSIÓN_BASE}/*
│ │ │ │ └─ *{NOMBRE_VERSIÓN_BASE}.bin*
│ │ │ ├─ *{NOMBRE_VERSIÓN_AUMENTO} - AUMENTO/*
│ │ │ │ └─ *{NOMBRE_VERSIÓN_AUMENTO}.bin*
│ │ │ └─ *checksums.txt # Formato FIRMA*
│ │ │
│ │ ├─ *CERTIFICACIÓN/*
│ │ │ ├─ *{NOMBRE_VERSIÓN_BASE}/*
│ │ │ │ ├─ *{NOMBRE_VERSIÓN_BASE}_CID{NUM_CID}.bin*
│ │ │ │ └─ *{NOMBRE_VERSIÓN_BASE}*{BUILD}.pkg
│ │ │ ├─ {NOMBRE*VERSIÓN_AUMENTO} - AUMENTO/*
│ │ │ │ ├─ *{NOMBRE_VERSIÓN_AUMENTO}_CID{NUM_CID}.bin*
│ │ │ │ └─ *{NOMBRE_VERSIÓN_AUMENTO}*{BUILD}.pkg
│ │ │ └─ checksums.txt # Formato CERTIFICACIÓN
│ │ │
│ │ ├─ RELEASE NOTE/
│ │
│ └─ ROADMAP/

📂 Ejemplo aplicado – Proyecto ATC

ATC/
├─ NEW6260/
│ ├─ ENLACEAV2.0.0_250812/
│ │ ├─ FIRMA/
│ │ │ ├─ ENLACEAV2.0.0/
│ │ │ │ └─ ENLACEAV2.0.0.bin
│ │ │ ├─ ENLACEAV2.0.1 - AUMENTO/
│ │ │ │ └─ ENLACEAV2.0.1.bin
│ │ │ └─ checksums.txt # Formato FIRMA

```
│ │ │
│ │ ├─ CERTIFICACIÓN/
│ │ │ ├─ ENLACEAV2.0.0/
│ │ │ │ ├─ ENLACEAV2.0.0_CID20149212.bin
│ │ │ │ └─ ENLACEAV2.0.0_250812.pkg
│ │ │ ├─ ENLACEAV2.0.1 - AUMENTO/
│ │ │ │ ├─ ENLACEAV2.0.1_CID20149212.bin
│ │ │ │ └─ ENLACEAV2.0.1_250812.pkg
│ │ │ └─ checksums.txt # Formato CERTIFICACIÓN
│ │ │
│ │ ├─ RELEASE NOTE/
│ │
│ ├─ ENLACEAV2.0.2_250820/
│ │ ├─ FIRMA/
│ │ │ └─ ... (misma estructura)
│ │ ├─ CERTIFICACIÓN/
│ │ │ └─ ... (misma estructura)
│ │ ├─ RELEASE NOTE/
│ │
│ └─ ROADMAP/ # Único por proyecto, se mantiene fuera de las versiones
```

📌 Nota Importante

El patrón de carpeta `└─ [NOMBRE_VERSIÓN]_[BUILD]/` se repetirá para **todas las versiones consecutivas** del proyecto (por ejemplo: `ENLACEAV2.0.1_xxxxxx`, `ENLACEAV2.0.2_xxxxxx`, etc.), manteniendo exactamente la misma estructura interna. Para optimizar el flujo y minimizar errores, el programa o script que gestione este proceso debe:

- Almacenar y reutilizar los parámetros ingresados en la primera ejecución (cliente, nombre de versión base, dispositivo, CID, tipo de firma, etc.).
- Permitir que en ejecuciones posteriores **solo se modifiquen los campos variables** (número de versión, build, notas o tickets asociados).
- Evitar la reintroducción manual completa de datos en cada nueva entrega, reduciendo tiempos de preparación y riesgo de inconsistencias.

8. Crear carpeta de ROADMAP productivo:
   Mantener un registro visual y editable del roadmap productivo del proyecto, actualizado con cada nueva entrega de versión.

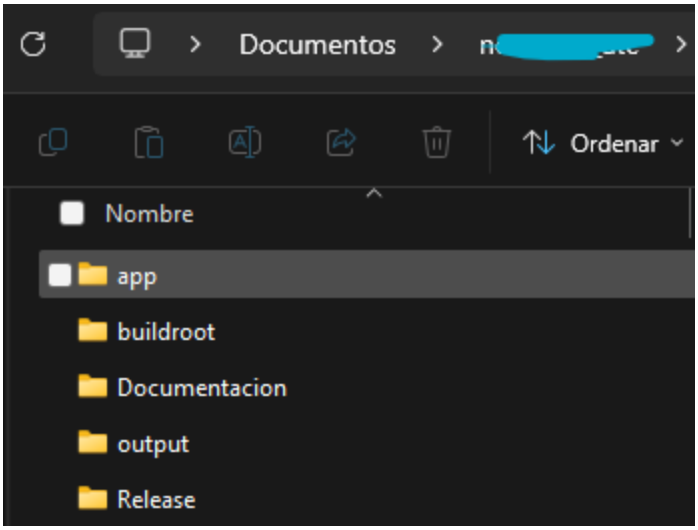
1. Estructura:
   - La carpeta **ROADMAP** se crea dentro del proyecto, **al mismo nivel** que las carpetas de versiones (`[NOMBRE_VERSIÓN]_[BUILD]/`).
   - Ejemplo de estructura:
     [CLIENTE]/
     ├─ [PROYECTO]/
     │ ├─ [NOMBRE_VERSIÓN]*[BUILD]/*
     │ ├─ *[NOMBRE_VERSIÓN]*[BUILD]/
     │ └─ ROADMAP/
2. 🛠 Pasos del proceso:
   1. **Creación de carpeta**
      - Ubicar la carpeta `ROADMAP/` dentro del directorio del proyecto (al mismo nivel de las versiones entregadas).
      - Asegurar que la carpeta exista; si no, crearla automáticamente.
      - Solo existe **un roadmap productivo**, que se actualiza conforme se entregan nuevas versiones.
   2. **Gestión de archivo** `.drawio:`
      - Si no existe, **generar un archivo base** `.drawio` siguiendo el formato de roadmap establecido para el cliente/dispositivo. Pero si ya existe, **abrirlo y editarlo** para agregar la nueva versión entregada:
        - Insertar un nuevo nodo/bloque con la nomenclatura correcta de la versión (`[NOMBRE_VERSIÓN]_[BUILD]`).
        - Seguir el estilo y formato visual del roadmap.
        - Guardar los cambios y obtener el archivo actualizado para cargarlo a la carpeta correspondiente.
   3. **Exportación a** `.png`
      - Generar la imagen actualizada del roadmap desde el `.drawio` .
      - Usar la exportación en alta calidad y formato estándar para adjuntar en entregas.
   4. **Versionado y respaldo**
      - Mantener **copias previas** del `.drawio` antes de modificarlo, en caso de necesitar revertir cambios. Antes de modificar el archivo `.drawio` del roadmap, se debe **generar una copia de seguridad** en una **ruta local separada** designada para respaldos (no dentro de `ROADMAP/`).

- La carpeta de respaldo debe estar organizada por fecha y hora, por ejemplo: `BACKUP_ROADMAP/YYYYMMDD_HHMM/roadmap_viejo.drawio`.
- Guardar tanto el `.drawio` como el `.png` en la carpeta `ROADMAP/`.

2. *DESAROLLO (DEV):*

🎯 **Objetivo:** Mantener actualizado el material interno de desarrollo (Release Notes y Roadmap DEV) y la Wiki actualizada.

📚 Actividades:
1. Verificar acceso Git (PAT/App Password) y que la **Wiki está habilitada** en Bitbucket.
2. Sincronizar/Clonar la Wiki del proyecto.
3. Construcción y actualización del Release Note de Desarrollo:
1. Ubicación: Carpeta `Release/` del proyecto.
Ejemplo:



2. Formato: Word + PDF acumulativo, reutilizando la tabla base del release productivo.
- Contenido:
- Mantener y listar todas las versiones
- Colocar la versión más reciente **primero** en la primera hoja.
- Las versiones anteriores deben ir en **orden cronológico descendente** hacia abajo.
- Respetar el formato precargado y si se inicia desde cero, adjuntar y seguir un formato de ejemplo aprobado.
- Reutilizar tabla del Release productivo
- Copiar y pegar la tabla correspondiente del Release productivo.
- Adaptar la información al entorno de desarrollo.
- Integrar en el Release de Desarrollo
- Pegar la tabla adaptada en el documento de desarrollo.

- Ordenar de forma descendente (última versión arriba en la primera hoja).
- Mantener sin alteraciones todas las versiones anteriores ya registradas.

Ejemplo:

VERSIÓN	ENLACEAV2.0.1_20250627	FECHA	27/06/25
VERSIÓN BASE	ENLACEAV2.0.1_20250624	AMBIENTE	CERTIFICACIÓN
IP	34.192.135.190		
PUERTO DE DESCARGA	5555		
PUERTO DE INICIALIZACIÓN	8[illegible]		
DESCRIPCIÓN DE DESARROLLO			
IMPLEMENTACIONES Y/O AJUSTES	• 45403 - POS_Ajuste a los mensajes del reverso. • 45407 - POS_Ajuste del tiempo para la carga de la tarjeta. • 45402 - POS_Revisión del timeout.		
CONSIDERACIONES	o Para llevar a cabo las pruebas correspondientes de descarga remota, es necesario comprimir el archivo con extensión bin al formato gz. o Se recomienda realizar pruebas puntuales de los tickets previamente mencionados.		

VERSIÓN	[illegible]	FECHA	24/06/25
VERSIÓN BASE	[illegible]	AMBIENTE	CERTIFICACIÓN
IP	[illegible]		
PUERTO DE DESCARGA	[illegible]		
PUERTO DE INICIALIZACIÓN	[illegible]		
DESCRIPCIÓN DE DESARROLLO			

Nueva ver. ↑

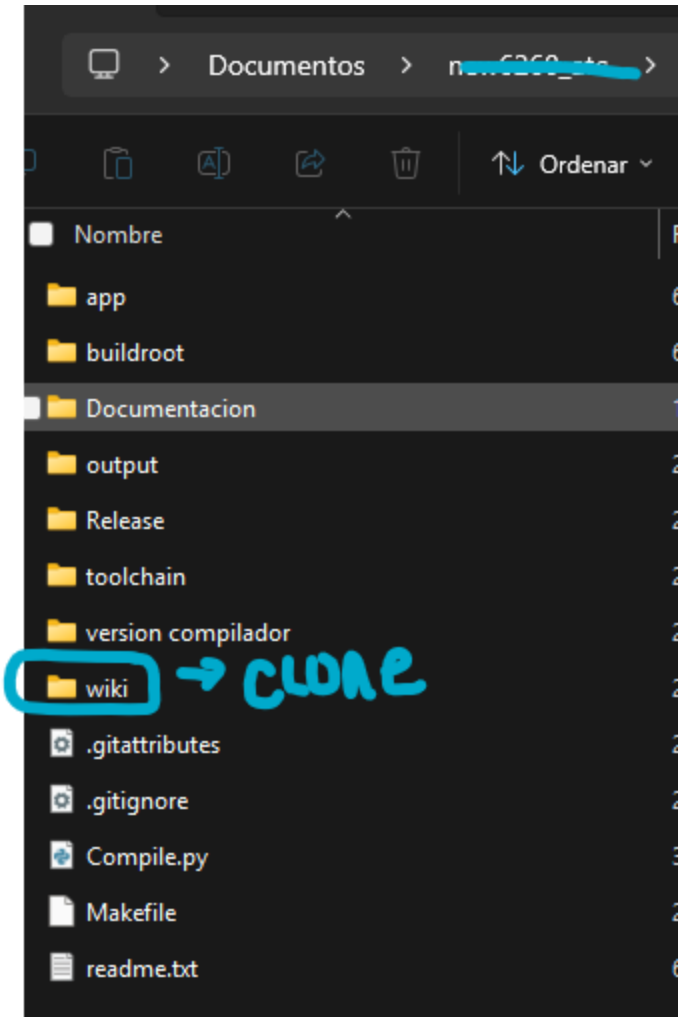
Otros ↓

- Guardar y generar PDF
- Actualizar y guardar el documento Word.
- Generar el PDF correspondiente.
- Ambos (Word y PDF) deben guardarse en **la misma carpeta donde estaba la versión anterior**.
- Si la carpeta no existe, **crear automáticamente** la estructura en el repositorio del proyecto.

3. Actualización de la Wiki de Desarrollo:

- Confirmar que la Wiki del proyecto está habilitada en Bitbucket.
- Sincronizar o clonar la Wiki en la carpeta del proyecto.

Ejemplo:



- Localizar y editar el archivo `.md` del Release:
- Buscar el archivo Markdown correspondiente al Release de Desarrollo en la estructura de la Wiki.
- Abrirlo y actualizarlo siguiendo el formato establecido en las nuevas directrices (mantener consistencia con la estructura y estilo acordados).
- Directrices:

2. **Manejo de roadmaps por repositorio**

Se deberán mantener dos roadmaps diferenciados en cada repositorio:

- Roadmap de desarrollo: versiones generadas y aprobadas internamente.
- Roadmap de cliente: versiones entregadas oficialmente al cliente.

Este segundo roadmap debe ser coordinado y revisado en conjunto con el PM del respectivo proyecto. Mismos roadmaps deberán ser incluidos en la Wiki del repositorio.

3. **Documentación de versiones en la Wiki del repositorio**

Al documentar una nueva versión en la wiki del repositorio, se deberá incluir un nuevo campo obligatorio (Fecha de entrega al cliente):
Los posibles valores para este campo son:

- " " (vacío): valor por defecto hasta que se confirme entrega.
- "N/A": cuando la versión ha sido devuelta por el equipo de QA, versiones de certificación de franquicias, versiones DEMO o aquellas versiones no oficiales productivas.
- "Fecha de entrega al cliente": se completa únicamente cuando el PM lo confirma mediante respuesta al mismo hilo de correo por el cual se compartió la versión desde desarrollo.

Ejemplo:

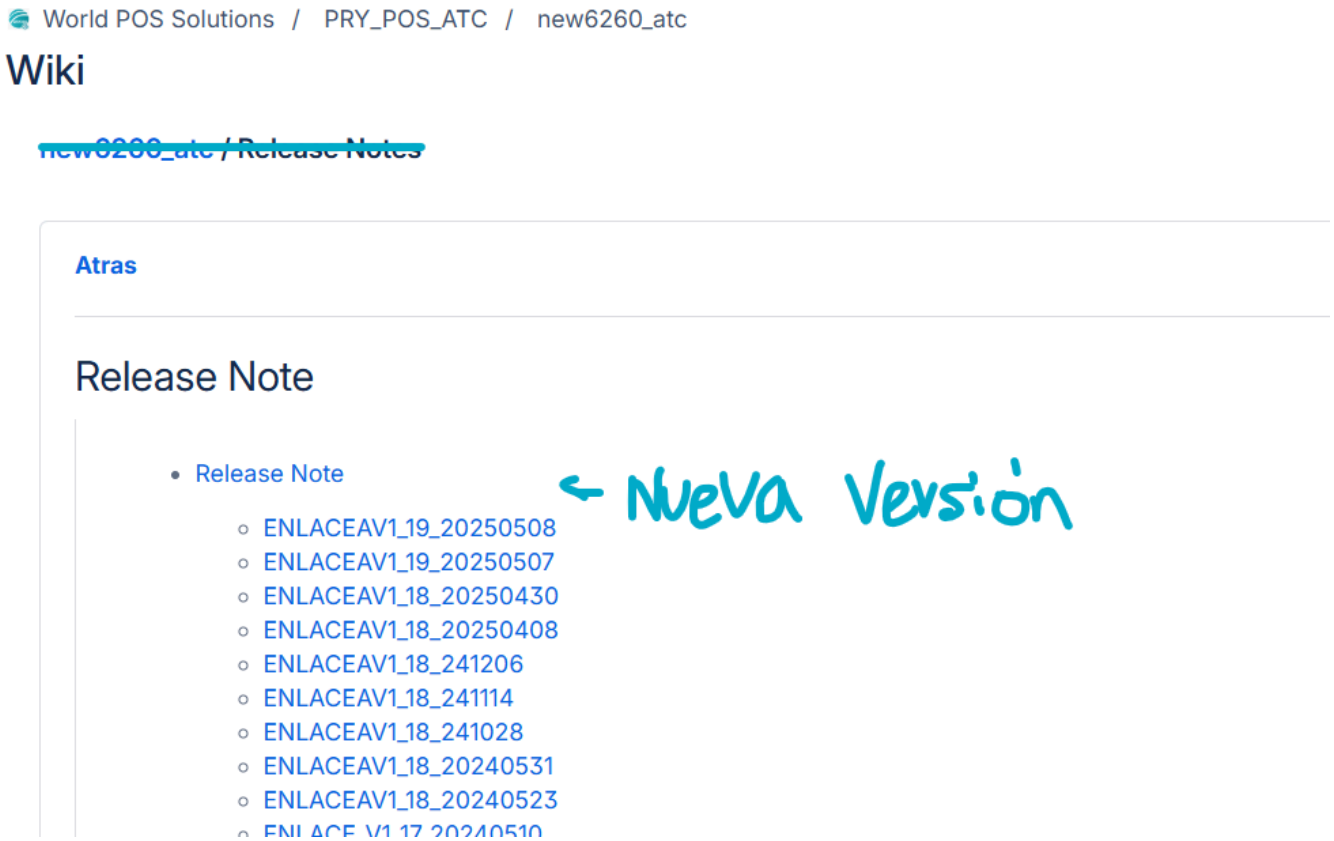
MNETV2.18.0_020725

Fecha: *02/07/2025* ***Creado:*** *Johan Criado* ***Revisado:*** *Rafael Sanchez* ***Version Base:*** *MNETV2.17.2_160625* ***Fecha entrega al cliente:*** *N/A*

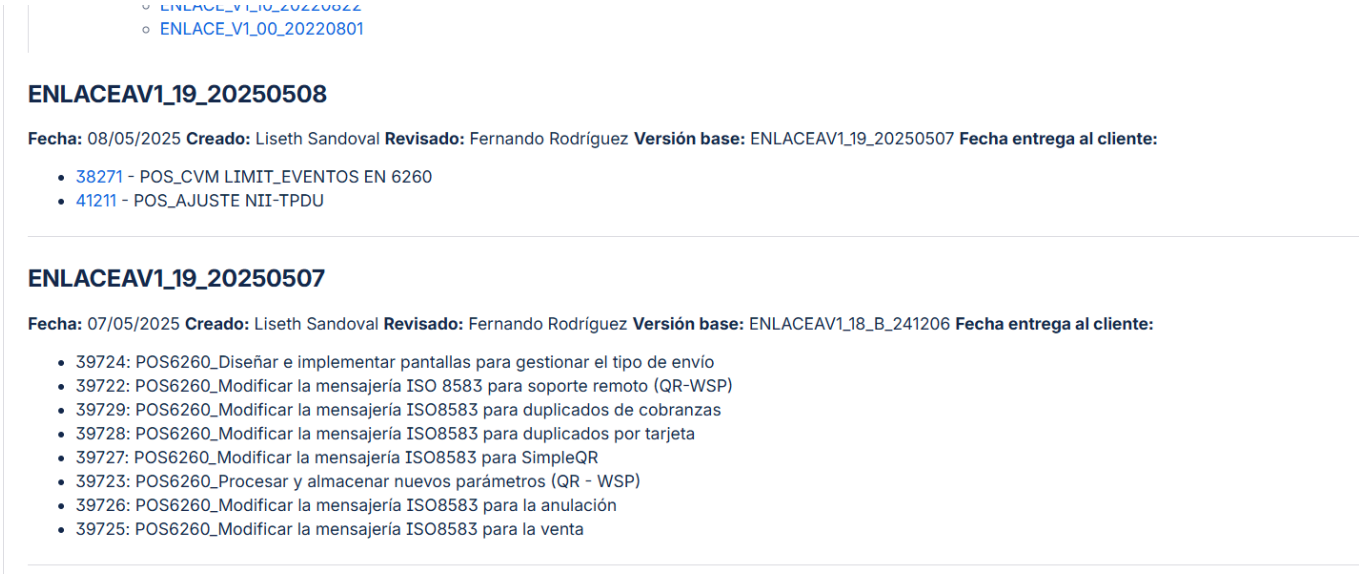
- 33677 - RQ20_Cierre automatico

- RELEASE WIKI:

1. Agregar el ítem de la versión en la sección de listado general de versiones:



2. Más abajo en el mismo documento, agregar la nueva versión con su detalle completo (Conservar las demás, solo es adicionar lo nuevo):



ENLACE_V1_00_20220801

ENLACEAV1_19_20250508

Fecha: 08/05/2025 **Creado:** Liseth Sandoval **Revisado:** Fernando Rodríguez **Versión base:** ENLACEAV1_19_20250507 **Fecha entrega al cliente:**

- 38271 - POS_CVM LIMIT_EVENTOS EN 6260
- 41211 - POS_AJUSTE NII-TPDU

ENLACEAV1_19_20250507

Fecha: 07/05/2025 **Creado:** Liseth Sandoval **Revisado:** Fernando Rodríguez **Versión base:** ENLACEAV1_18_B_241206 **Fecha entrega al cliente:**

- 39724: POS6260_Diseñar e implementar pantallas para gestionar el tipo de envío
- 39722: POS6260_Modificar la mensajería ISO 8583 para soporte remoto (QR-WSP)
- 39729: POS6260_Modificar la mensajería ISO8583 para duplicados de cobranzas
- 39728: POS6260_Modificar la mensajería ISO8583 para duplicados por tarjeta
- 39727: POS6260_Modificar la mensajería ISO8583 para SimpleQR
- 39723: POS6260_Procesar y almacenar nuevos parámetros (QR - WSP)
- 39726: POS6260_Modificar la mensajería ISO8583 para la anulación
- 39725: POS6260_Modificar la mensajería ISO8583 para la venta

Referencia del contenido:

- La información de la descripción y detalles del Release puede reutilizarse directamente del documento de Release (Word/PDF) elaborado para desarrollo o incluso del productivo si corresponde.
- El bloque de detalles se nutre de:
- Títulos de commits relevantes extraídos del control de versiones (Git/Sourcetree).
- Información ya registrada en el Release documentado.
- Esto asegura coherencia entre el documento de Release y la Wiki, evitando duplicar redacción y reduciendo el riesgo de inconsistencias.
- Actualizar RoadMap en la Wiki:

Mantener el archivo `.drawio` de desarrollo con **todas las versiones internas previamente registradas**, conservando su estructura y estilo visual. Al recibir una nueva versión, **agregarla como un nuevo nodo/bloque** siguiendo el formato establecido (`[NOMBRE_VERSIÓN]_[BUILD]`), sin eliminar o alterar las entradas existentes. Posteriormente, exportar el `.png` actualizado y subir ambos (archivo `.drawio` y `.png`) a la Wiki, asegurando que el roadmap refleje **todo el historial de versiones internas** junto con la más reciente.