Integración de QZ Tray en DixTPV

Resumen

Se ha enlazado el flujo de cobro de DixTPV con la infraestructura de impresión térmica del plugin Tickets, reutilizando QZ Tray para lanzar automáticamente el ticket de la factura recién generada. El objetivo era imprimir sin que el usuario tuviese que navegar al módulo Tickets ni interactuar con la vista previa salvo que fuese imprescindible.

Cambios principales

- **Respuesta del backend** (`Plugins/DixTPV/Controller/DixTPV.php`):
    - Las acciones `cobrarCuenta` y `cobrarCuentaDividida` devuelven ahora el array completo con `document`, `success`, `message` y `resultado`. Esto permite al frontal recuperar `modelClassName` y `modelCode` para solicitar el ticket ESC/POS contra `SendTicket`.
- **Flujo de impresión en el frontal** (`Plugins/DixTPV/Assets/JS/moduloCobro.js` y su utilitario replicado `Plugins/DixTPV/Assets/JS/Utilidades_Cobro.js`, así como las copias en `Dinamic/Assets/JS/...`):
    - Se guarda en `window.dixTpvLastTicketDoc` el último documento generado.
    - Nueva función `requestManualFallback()` que sólo abre el modal de vista previa cuando no hay conexión con QZ Tray, falta impresora configurada o el envío raw falla.
    - Se mantiene una promesa (`manualCompletionPromise`) que bloquea el refresco del TPV hasta que el usuario cierre el modal o confirme la impresión manual.
    - Ajustes para reutilizar la impresora almacenada, evitar dobles llamadas a `listSystemPrinters()` y cerrar el modal automáticamente tras una impresión manual exitosa.
- **Comunicación con QZ Tray** (`Plugins/Tickets/View/Modal/PrintTicketModal.html.twig` + versión `Dinamic/...`):
    - `sendEscposToPrinter()` emite un evento personalizado `dixTpvTicketPrinted` cuando el envío a QZ/USB finaliza con éxito. Este evento, enganchado en DixTPV, libera la promesa pendiente y cierra el modal en los escenarios manuales.

Flujo actualizado

1. Tras confirmar el cobro (`cobrarCuenta`/`cobrarCuentaDividida`), el backend devuelve la información del nuevo documento.
2. `automaticTicketPrint(docInfo)` intenta:
    - Asegurar sesión con QZ Tray (`ensureQZSession()`).
    - Resolver impresora, formato y ancho de papel.
    - Solicitar los datos ESC/POS al controlador `SendTicket` (acción `get-escpos`).
    - Enviarlos a QZ Tray a través de `sendEscposToPrinter`.

3. Si algo falla (QZ desconectado, impresora sin configurar, error en envío), se abre el modal de vista previa para que el usuario imprima o cierre manualmente.
4. Cuando QZ Tray confirma la impresión (automatizada o manual), se dispara `dixTpvTicketPrinted`, el modal se oculta y el TPV continúa con el refresco habitual.

Funciones destacadas

Función	Ubicación	Descripción
`requestManualFallback()`	`moduloCobro.js` / `Utilidades_Cobro.js`	Prepara el modal de vista previa sólo cuando la impresión automática no es viable. Gestiona eventos para resolver la promesa pendiente.
`automaticTicketPrint(docInfo)`	Id.	Orquesta la secuencia completa: obtiene docInfo, asegura QZ, pide ESC/POS y envía a la impresora.
`sendEscposToPrinter()`	`PrintTicketModal.html.twig`	Envía los datos a QZ Tray (por nombre de impresora o USB), muestra mensajes en el modal y lanza `dixTpvTicketPrinted` en caso de éxito.
`ensureQZSession()`	`moduloCobro.js` / `Utilidades_Cobro.js`	Alta de la conexión con QZ Tray, con espera activa hasta que el websocket está listo.
`requestTicketEscposData()`	`moduloCobro.js` / `Utilidades_Cobro.js`	Llama a `SendTicket` para recuperar el ESC/POS con el formato y ancho seleccionados.

Qué queda pendiente

Certificados para QZ Tray

La versión gratuita de QZ Tray muestra dos diálogos por impresión (`Signature Required` y `Allow Printing`). Para eliminar estas confirmaciones y habilitar impresión silenciosa, es necesario firmar los trabajos con un certificado digital confiable. Pasos recomendados:

1. **Elegir enfoque de firma**
   - *Autofirmado (entorno controlado)*: generar un certificado y distribuir manualmente la clave pública en los equipos con QZ Tray. Adecuado para instalaciones cerradas.
   - *Autoridad certificadora (AC) pública*: ideal para despliegues masivos, elimina advertencias adicionales al importar el certificado.
2. **Generar certificado**

- QZ Tray utiliza keystore PKCS12 (`.p12`).
- Puede generarse con `keytool` (Java) o `openssl`.
- Ejemplo con `keytool`:

```
keytool -genkeypair -alias dixtpv-qztray -keyalg RSA -keysize 2048 \
-keystore dixtpv-qztray.p12 -storetype PKCS12 \
-validity 365 -storepass <password>
```

- Exportar también la clave pública (`.pem`) que se instalará en los clientes.

3. **Implementar la firma en el frontal**
   - QZ Tray necesita dos promesas en `qz.security`: `setCertificatePromise()` y `setSignaturePromise()`.
   - Actualmente se resuelven en blanco (`resolve()`), lo que provoca los avisos.
   - Debe reemplazarse por lógica que:
     - Devuelva el certificado (`resolve(certificateBytes)`) desde localStorage o un endpoint que sirva el `.pem`.
     - Firme los datos (`resolve(sign(toSign))`) usando la clave privada almacenada de forma segura (preferiblemente en backend y vía API).
4. **Distribución en los equipos**
   - Importar la clave pública en QZ Tray (`QZ Tray > Advanced > Certificates > Import`).
   - Verificar que aparece como "Trusted".
5. **Pruebas**
   - Con QZ Tray reiniciado, realizar una impresión de prueba.
   - Deberían desaparecer los diálogos de confirmación, quedando únicamente los mensajes configurados en la app (si se desea mantenerlos).

> Consulta la documentación oficial de QZ Tray para más detalles: https://qz.io/wiki/2.0-signing y https://qz.io/wiki/2.0-certificates.

Próximos pasos sugeridos

6. **Implementar firma real** en `PrintTicketModal.html.twig`, almacenando claves en el servidor y sirviendo la firma mediante AJAX seguro.
7. **Gestionar certificados**: script o tarea que distribuya la clave pública en los terminales al instalar/actualizar DixTPV.
8. **Monitoreo**: añadir logs específicos cuando falle la firma o la conexión con QZ, facilitando soporte remoto.
9. **Documentación de usuario final**: redactar guía rápida para terminales indicando cómo recuperar la impresora por defecto y qué hacer si los diálogos reaparecen.

Documento generado automáticamente como base para informes internos del desarrollo.