# DixTPV - Documentacion Funcional y Tecnica

## Tabla de contenidos


1. Introduccion
2. Vision general del plugin
3. Arquitectura de alto nivel
4. Flujo operativo principal
5. Funcionalidades complementarias
6. Componentes tecnicos clave
7. Modelo de datos
8. Configuracion y parametrizacion
9. Guia para el equipo de desarrollo
10. Riesgos conocidos y mejoras sugeridas
11. Apendece: estructura de carpetas


---

## 1. Introduccion

### 1.1 Proposito

El plugin **DixTPV** extiende FacturaScripts para cubrir un caso de uso de TPV orientado a hosteleria: gestion de salones, mesas, comandas, turnos de caja y cobros vinculados a la facturacion del ERP. Este documento ofrece una referencia funcional y tecnica para el nuevo equipo de desarrollo que se integra al proyecto.

### 1.2 Alcance

- Analiza las capacidades funcionales expuestas al usuario final.
- Describe la arquitectura tecnica, componentes y puntos de integracion.
- Detalla el modelo de datos y los procesos criticos.
- Recolecta riesgos actuales y oportunidades de mejora para asegurar continuidad y escalabilidad.

### 1.3 Publico objetivo

- Desarrolladores y arquitectos que asumiran mantenimiento o nuevas evoluciones.
- Integradores que necesiten adaptar DixTPV a despliegues concretos.
- Stakeholders tecnicos que requieren comprender capacidades y limitaciones.

---

## 2. Vision general del plugin

### 2.1 Resumen funcional

- Gestion integral de turnos de caja para terminales TPV dedicados.
- Carpeta visual centrada en la operativa de camareros: seleccion de salones, mesas y productos agrupados por familia.
- Flujo completo de venta: aparcar comandas, editar lineas, cobrar en bloque o dividido, emitir documentos de FacturaScripts (factura, albaran, pedido, presupuesto) y registrar cierres de caja.
- Integracion con impresoras de tickets mediante el plugin requerido `PrintTicket` y builders propios para vouchers y apertura de cajon.
- Alta rapida y busqueda de clientes desde la propia interfaz de venta.

### 2.2 Dependencias y compatibilidad

- Version minima de FacturaScripts: `2024.00` (`facturascripts.ini`).
- Dependencia directa declarada: plugin `PrintTicket`.
- Se apoya en modelos nativos de FacturaScripts (`Agente`, `Producto`, `Cliente`, `FormaPago`, `Serie`, etc.) y en nuevas tablas especificas definidas en `Table/`.

### 2.3 Requisitos de sistema recomendados

- PHP 8.1 o superior (en linea con requisitos de FacturaScripts 2024).
- Servidor web con acceso de escritura a la ruta del plugin para assets y caches.
- Impresoras compatibles con `Mike42\Escpos` cuando se usa printing directo.
- Navegador moderno (Chrome, Edge, Firefox) para la UI de ventas.

### 2.4 Roles y permisos implicitos

- Los **agentes** (camareros) son usuarios de FacturaScripts con permisos sobre el menu `DixTPV`.
- Los terminales se asignan a cajas y agentes mediante `Controller/DixTPV.php`.
- No se define un sistema de roles propio; se hereda el control de FacturaScripts sobre accesos al menu.

---

## 3. Arquitectura de alto nivel

### 3.1 Capas principales

- **Presentacion**: plantillas Twig (`View/`) y modulos JavaScript (`Assets/JS/`) renderizados desde `Controller/DixTPV.php`.
- **Logica de negocio**: controladores bajo `Controller/` y biblioteca `Lib/UtilsTPV.php`.
- **Datos**: modelos personalizados en `Model/`, integrados con las tablas definidas en `Table/`. Se emplea el ORM nativo de FacturaScripts (ModelClass + ModelTrait).
- **Integraciones**: generacion de tickets via `PrintTicket`, aperturas de cajon (`Lib/Ticket/Builder/CashDrawer.php`) y generacion de documentos comerciales de FacturaScripts.

### 3.2 Flujo request-response principal

1. `Controller/DixTPV.php` procesa la peticion en `privateCore()`.
2. Se cargan datos basicos (agentes, familias, mesas, salones, terminales, cajas, clientes, series, etc.).
3. Segun el parametro `action` se ejecuta logica AJAX especifica (`execAfterAction()`); en caso contrario se monta la vista principal.
4. Las interacciones de UI se apoyan en endpoints AJAX que devuelven JSON o HTML parcial (renderizado via `Html::render()`).
5. Las operaciones de cobro invocan `UtilsTPV` para persistir comandas y generar documentos/facturas, actualizando resumenes de venta asociados a la caja activa.

### 3.3 Componentes front-end

- Plantilla principal `View/DixTPV/Hosteleria/Ventas/DixTPV.html.twig` mas componentes reutilizables bajo `View/DixTPV/Hosteleria/Ventas/Componentes`.
- CSS especifico en `Assets/CSS/` (temas `tpv-bonito.css`, `mesacss.css`, etc.).
- JavaScript modular organizado por responsabilidad (`moduloCesta.js`, `moduloCobro.js`, `moduloMesasSalones.js`, `Utilidades_Cobro.js`, etc.). Cada modulo maneja eventos y actualiza la UI mediante llamadas AJAX al controlador principal.

### 3.4 Servicios y helpers

`Lib/UtilsTPV.php` centraliza operaciones de negocio complejas:

- Generacion de facturas, pedidos, albaranes y presupuestos a partir de comandas o cestas.
- Actualizacion de resumenes por forma de pago, tipo de IVA y producto (`dix_formas_pago_resumen`, `dix_tipo_iva_resumen`, `dix_productos_resumen`).
- Busqueda de productos y variantes, resolucion de impuestos y normalizacion de claves de producto para analitica.

---

## 4. Flujo operativo principal

### 4.1 Preparacion del turno

1. Un usuario accede a `DixTPV` desde el menu principal.
2. `createView()` verifica datos indispensables: agentes (camareros), terminales y clientes. Si falta alguno redirige a los listados correspondientes.
3. Con una caja abierta (`turnoAbierto()`), se carga la interfaz de ventas; en caso contrario se muestra la vista de apertura (`nuevoTurnoSlid.html.twig`).

### 4.2 Apertura de caja

- `execAfterAction('abrirCaja')` invoca `abrirCaja()` que registra una nueva fila en `dixtpv_cajas` con hora de apertura, camarero responsable y terminal asociado.
- El efectivo inicial se almacena en `dineroini` y se registra el agente en `camareroini`.

### 4.3 Gestion de salones y mesas

- Los salones se configuran en `Model/DixSalon.php` y se consumen desde la vista principal.
- `Assets/JS/moduloMesasSalones.js` coordina la seleccion de salon/mesa, su estado (`dix_estado`) y permite aparcar comandas por mesa.
- `Controller/DixTPV.php::obtenerMesasDisponibles()` responde con mesas libres para asignaciones rapidas.

### 4.4 Construccion de la comanda

- La UI organiza productos por familia (`familia.html.twig` + `Assets/JS/moduloCesta.js`).
- Al agregar lineas se mantiene una cesta en memoria (frontend) y se sincroniza con el backend al aparcar o cobrar.
- `aparcarCuenta()` (action `aparcarCuenta`) persiste la comanda y sus lineas en `dix_comanda` y `dix_details_comanda`, manteniendo mesa y salon ocupados.
- `recuperarAparcado()` reconstruye la cesta desde BD para continuar la venta.

### 4.5 Proceso de cobro completo

1. Se recopilan datos desde la UI (`cesta`, `formapago`, `idcliente`, `camarero`, `serie`).
2. `cobrarCuenta()` valida la mesa, almacena lineas si la comanda no existia y actualiza totales de caja (`dinerofin` para efectivo, `dinerocredito` para medios no efectivos).
3. Se invoca `generarDoc()` que delega en `UtilsTPV::generarFactura()` (actualmente fuerza factura como tipo por defecto) para crear el documento en FacturaScripts.
4. Tras generar el documento se actualiza `dix_listafins` (historico de ventas) y los resumenes analiticos.

### 4.6 Cobro dividido

- `cobrarCuentaDividida()` replica la logica anterior pero trabajando con subconjuntos de lineas (cesta parcial) y llama a `UtilsTPV::generarFacturaDiv()` para emitir un documento segmentado. Tambien actualiza caja y resumenes.
- El flujo espera que la UI gestione cuantos parciales se cobran y mantenga consistencia de cantidades.

### 4.7 Cierre de caja

- `cerrarCajaAuto()` cierra automaticamente la caja ajustando `dinerocierre` y `descuadre` a cero.
- `cerrarCaja()` permite introducir un conteo final y opcionalmente aceptar un descuadre; registra `fechahoracierre`, `camarerofin` y guarda la diferencia en `descuadre`.
- Tras cerrar, la interfaz vuelve a pedir apertura de turno.

### 4.8 Impresion y apertura de cajon

- `printTicket()` y `printVoucher()` generan tickets usando builders de `Lib/Ticket/Builder`.
- `openCashDrawer()` dispara la secuencia ESC/POS definida en `CashDrawer.php` para abrir el cajon portamonedas.

### 4.9 Gestion de clientes

- Busqueda incremental via `execAfterAction('searchclient')` que usa `searchClient()` con filtros `nombre` y `razonsocial`.
- Alta rapida con `anhadirCliente()` que crea el `Cliente` y su `Contacto` asociado.

---

## 5. Funcionalidades complementarias

- **Reservas**: `Model/DixReserva.php` y `Table/dix_reservas.xml` permiten vincular reservas a mesas, comandas y franjas horarias. El controlador principal puede extenderse para explotarlas (actulamente sin endpoints especificos).
- **Tarifas**: `Model/DixTarifa.php` y `Table/dix_tarifas.xml` definen horarios, incrementos y descuentos aplicables por salon.
- **Estados de mesa**: `dix_estado` fija el flujo visual (libre, ocupado, limpieza, etc.), con datos iniciales en `Data/Codpais/ESP/dix_estado`.
- **Tipos de mesa**: `dix_tipo_mesa` permite clasificar mesas por capacidad.
- **Resumenes de venta**:
  - `dix_formas_pago_resumen` acumula importes por forma de pago y caja.
  - `dix_tipo_iva_resumen` agrega ventas por tipo impositivo.
  - `dix_productos_resumen` consolida unidades e importes por producto o descripcion.
- **Movimientos de caja**: `dixtpv_movimientos` registra ingresos/egresos manuales y conciliaciones.
- **Comanderos**: `dix_comanderos` y `Model/DixTPVComandero.php` permiten configurar dispositivos o roles que reciben las comandas (cocina, barra).
- **Billetes**: `dix_billetes` almacena denominaciones para paneles de efectivo y arqueo.

---

## 6. Componentes tecnicos clave

### 6.1 Controladores (`Controller/`)

- `Controller/DixTPV.php`: nucleo del TPV. Gestiona sesiones, acciones AJAX, cobros, cierres, impresion y utilidades varias.
- `Controller/InicioTPV.php`: punto de entrada para anclar vistas de inicio y gestion de permisos.
- `Controller/DixSelectAgent.php`, `List*` y `Edit*`: controladores genericos para CRUD de nuevas entidades (salones, mesas, estados, tarifas, terminales, cajas, comandas, reservas, etc.). Extienden clases base de FacturaScripts para mantener

consistencia con la UI de administracion.
- `Controller/crearEmpresaPrueba.php`: script auxiliar (probablemente para onboarding) que crea una empresa de prueba.

### 6.2 Modelos (`Model/`)

- Cada clase representa una tabla definida en `Table/`. Ejemplos:
    - `Model/DixTPVComanda.php`: gestiona encabezados de comanda, asigna nombres legibles a cliente, mesa y salon durante `save()`.
    - `Model/DixDetailComanda.php`: detalle de lineas, con valores por defecto y claves foraneas a productos.
    - `Model/DixTerminal.php`: configuracion de terminales (almacen, serie, formato de ticket, flags de UI).
    - `Model/DixTPVCaja.php`, `Model/DixMovimiento.php`, `Model/DixListaFin.php`, `Model/DixFormasPagoResumen.php`, etc. soportan seguimiento financiero del turno.

### 6.3 Biblioteca de negocio (`Lib/UtilsTPV.php`)

- Implementa funciones reutilizables para generar documentos de venta (`generarFactura`, `generarPedido`, `generarAlbaran`, `generarPresupuesto`) y sus variantes divididas.
- Resuelve productos y variantes (`getVariant`, `getProduct`), calcula impuestos (`resolveTaxCode`) y mantiene resumenes acumulados (`upsertResumenFormasPago`, `upsertResumenIVA`, `upsertResumenProducto`).
- Contiene logica pendiente de pulir (ver seccion de riesgos), pero centraliza la mayoria de integraciones con FacturaScripts Dinamic.

### 6.4 Extensiones (`Extension/`)

- **Modelos**: se amplian clases nativas (`Familia`, `FormaPago`, `FacturaCliente`) para anadir campos y comportamientos. Ejemplo: `Extension/Model/Familia.php` agrega colores de fondo/texto para los botones de familia en la UI.
- **Tablas**: archivos XML agregan columnas a tablas existentes (`agentes.xml` anade `clave`, `facturascli.xml` incorpora `idcaja`, etc.).
- **XMLView**: se ajustan formularios de edicion de familias, agentes y formas de pago para exponer los nuevos campos en la UI generica de FacturaScripts.

### 6.5 Frontend (`Assets/` y `View/`)

- **Plantillas Twig**: estructuran la interfaz de venta y los modales (`modalCierre.html.twig`, `modalCamareros.html.twig`, `modalNuevoCliente.html.twig`, etc.).
- **JavaScript**:
    - `DixTPV.js`, `moduloGeneral.js`: bootstrap de la app, registro de listeners generales.
    - `moduloCesta.js`, `newLine.js`: gestion de la cesta, calculos locales, renderizado dinamico.
    - `moduloCobro.js`, `Utilidades_Cobro.js`: prepara el payload de cobro, controla dialogos de pago dividido, integra con resumentes de caja.
    - `moduloMesasSalones.js`, `salonesMesas.js`: actualizacion visual del mapa de mesas.
    - `moduloAparcados.js`, `moduloCaja.js`, `mantenerCajaAbierta.js`: sincronizacion de estados, recordatorios y polling.
    - `nuevoCliente.js`, `moduloModifProducto.js`: formularios secundarios.
- **CSS**: archivos tematicos para adaptar la UI a pantallas tactiles y dispositivos de punto de venta.

### 6.6 Datos semilla y traducciones

- `Data/Codpais/ESP` y `Data/Lang/ES` incluyen seeds para `dix_estado` y `dix_tipo_mesa`.
- `Translation/es_ES.json` almacena cadenas traducidas al castellano para la interfaz.

---

## 7. Modelo de datos

### 7.1 Nuevas tablas principales

| Tabla | Descripcion | Campos clave y notas |
|---|---|---|
| `dixtpv` | Terminales TPV configurables | `idtpv`, `codalmacen`, `codpago`, `codserie`, flags de comportamiento (sonido, grupos, mostrar camareros). |
| `dixtpv_cajas` | Cajas abiertas por terminal | `idcaja`, `pertenenciaterminal`, `dineroini`, `dinerofin`, `dinerocredito`, `dinerocierre`, `descuadre`, `camareroini`, `camarerofin`. |
| `dixtpv_movimientos` | Movimientos manuales de caja | `id`, `idcaja`, `idtpv`, `amount`, `motive`, `idasiento` (enlace contable). |
| `dix_comanda` | Cabecera de comanda | `idcomanda`, `fecha`, `idcliente`, `idmesa`, `idsalon`, `idestado`, totales y observaciones. |
| `dix_details_comanda` | Lineas de comanda | `iddetalle`, `idcomanda`, `idproducto`, `descripcion`, `cantidad`, precios, |

| | | |
|---|---|---|
| | | codimpuesto. |
| dix_salones | Salones del establecimiento | idsalon, codsalon, nombre, codtarifa. |
| dix_mesa | Mesas fisicas | idmesa, idsalon, nombre, tipo_mesa, estado, comensales. |
| dix_estado | Estados de mesa/comanda | idestado, nombre, orden. |
| dix_tipo_mesa | Clasificacion de mesas | idtipomesa, nombre, personasaprox. |
| dix_tarifas | Tarifas avanzadas | Horarios y factores de precio (incremento1/2, descuento1/2). |
| dix_reservas | Reservas de mesas | idreserva, idmesa, idcomanda, fecha-hora, numtarjeta. |
| dix_listafins | Historico de ventas por documento | idlista, codpago, idcaja, idfactura, idagente, total. |
| dix_formas_pago_resumen | Resumen por forma de pago | Unique (idcaja, codpago) con acumulado total. |
| dix_tipo_iva_resumen | Resumen por impuesto | Unique (idcaja, codimpuesto) con total. |
| dix_productos_resumen | Resumen por producto | Unique (idcaja, claveprod) con unidades, total. |
| dix_billetes | Configuracion de denominaciones | idbillete, coddivisa, icono, valor, cantidad. |
| dix_comanderos | Dispositivos/comanderos | idcomandero, tipocomandero, idterminal. |

### 7.2 Alteraciones a tablas existentes

- agentes: se anade clave (usado para identificar camareros en la UI).
- facturascli: se extiende para guardar idcaja e idagente, permitiendo trazabilidad de ventas por turno.
- familias: campos color_fondo y color_texto para personalizar botones.
- formaspago: bandera efectivo para discriminar acumulados de caja fisica vs credito.

---

## 8. Configuracion y parametrizacion

1. **Instalacion**: copiar el plugin a Plugins/DixTPV y activar desde el panel de FacturaScripts.
2. **Dependencias**: asegurar que PrintTicket esta instalado y configurado.
3. **Terminales (dixtpv)**:
    - Definir almacen, forma de pago por defecto, serie, tipo de documento y formato de ticket.
    - Parametrizar opciones de interfaz (mostrarcamareros, grouplines, sound).
4. **Cajas (dixtpv_cajas)**: abrir turno desde la UI indicando efectivo inicial y camarero.
5. **Familias y productos**: asignar colores y agrupaciones para mejorar la navegacion tactil.
6. **Formas de pago**: marcar cuales se consideran efectivo para el arqueo.
7. **Impresoras**: configurar PrintTicket con conectores ESC/POS compatibles. El builder VoucherTicket asume line width segun Tools::settings('ticket', 'linelength', 50).
8. **Datos semilla**: revisar estados y tipos de mesa predefinidos; ajustar segun el negocio.

---

## 9. Guia para el equipo de desarrollo

### 9.1 Puntos de entrada clave

- Controller/DixTPV.php::execAfterAction() centraliza las acciones AJAX. Cualquier nueva funcionalidad UI deberia declararse aqui y devolverse con JSON coherente.
- Lib/UtilsTPV.php es la capa para logica de negocio reutilizable; evita duplicar reglas en controladores o JS.
- Plantillas Twig y modulos JS siguen una convencion Componentes/ + Vistas/ para piezas reutilizables.

### 9.2 Buenas practicas recomendadas

- Mantener la sincronizacion entre estado de mesa en BD y front-end para evitar carreras en escenarios multi terminal.
- Reducir logica duplicada entre cobrarCuenta() y cobrarCuentaDividida() moviendola a servicios compartidos.
- Anadir validaciones server-side previas a la emision de documentos (existencia de caja abierta, totales coherentes, impuestos definidos).
- Garantizar consistencia de codimpuesto: actualmente se normaliza concatenando IVA a codigos numericos; conviene establecer un mapeo explicito.

### 9.3 Extensiones y personalizacion

- Cualquier nuevo campo debe declararse en Table/*.xml, extender el modelo en Model/ y, si afecta a tablas base, anadir el XML correspondiente en Extension/Table/.
- Para nuevas vistas de administracion, reutilizar los controladores genericos Edit* y List*, definiendo XMLView para adaptar

formularios.
- La UI se actualiza mayoritariamente via AJAX; preferir respuestas parciales (HTML renderizado) cuando el cambio afecte a bloques grandes (familias, cesta) y JSON para actualizaciones puntuales.

### 9.4 Testing y observabilidad

- No existen pruebas automaticas ni logs estructurados. Considerar agregar:
    - Logs contextuales en `UtilsTPV` (actualmente se usa `error_log`) con niveles diferenciados.
    - Pruebas funcionales de generacion de documentos y cierre de caja.
    - Validaciones en JS para evitar peticiones incompletas.
- Para diagnosticos en produccion, habilitar el log de FacturaScripts y monitorear `error_log` del servidor web.

---

## 10. Riesgos conocidos y mejoras sugeridas

- **Forzado de tipo documental**: `generarDoc()` y `generarDocDiv()` reasignan `doctype = 'FacturaCliente'`, ignorando la configuracion del terminal. Revisar para respetar la preferencia del usuario.
- **Funciones incompletas en `UtilsTPV`**: existen metodos como `generarAlbaranDiv()` que referencian `$comandaId` sin estar definido, y placeholders (`PedidoDiv`, `PresupuestoDiv`) pendientes de implementar. Catalogar y corregir antes de activar cobros divididos en produccion.
- **Extension clases**: en `Extension/Model/Agente.php` el nombre de clase (`Familia`) parece inconsistente con el archivo, lo que puede provocar conflictos de autoload. Validar y corregir.
- **Sincronizacion de mesas**: `aparcarCuenta()` elimina comandas previas para una mesa sin validar estados concurrentes. Considerar bloqueos optimistas/pesimistas o control por terminal.
- **Validacion de cierres**: `cerrarCaja()` no impide cerrar cajas con descuadres sin registrar motivo. Evaluar registrar movimientos de ajuste en `dixtpv_movimientos`.
- **Normalizacion de impuestos**: la conversion `IVA` + numero puede fallar si el codigo real del impuesto difiere de esa nomenclatura. Implementar una tabla de equivalencia o usar `codimpuesto` recibido sin transformaciones.
- **Traducciones parciales**: la UI mezcla cadenas hardcoded en castellano y textos localizables. Completar `Translation/es_ES.json` y preparar internacionalizacion futura.

---

## 11. Apendece: estructura de carpetas

```
Plugins/DixTPV
|-- Assets/
|   |-- CSS/
|   `-- JS/
|-- Controller/
|-- Data/
|-- DataSrc/
|-- Extension/
|   |-- Model/
|   |-- Table/
|   `-- XMLView/
|-- Lib/
|   `-- Ticket/
|       `-- Builder/
|-- Model/
|-- Table/
|-- Translation/
|-- View/
|   `-- DixTPV/
|       `-- Hosteleria/
|           `-- Ventas/
`-- docs/
    `-- DixTPV_Documentacion.md
```

---

**Contacto**: El repositorio mantiene como autor original a `Alexis Cambeiro`. Para dudas o PRs, seguir el flujo estandar de FacturaScripts y revisar compatibilidad con versiones futuras (renombrados de namespace, cambios en ORM, etc.).