# Estudio de Factibilidad Técnica, Arquitectura y Diseño para un Sistema de Liquidación de Sueldos SaaS Multitenant en el Marco Normativo Argentino

## 1. Introducción y Contexto del Mercado

La administración de nóminas en la República Argentina presenta uno de los ecosistemas más complejos y volátiles del mundo en términos de cumplimiento normativo y tributario. Históricamente, este mercado ha sido dominado por soluciones de software de escritorio ("On-Premise"), siendo el sistema **Bejerman** (actualmente parte del portafolio de Thomson Reuters) el estándar de facto para estudios contables y grandes empresas.[1] La hegemonía de Bejerman se fundamenta no en su interfaz gráfica, que a menudo es considerada anacrónica, sino en su **motor de cálculo flexible** basado en conceptos y fórmulas definibles por el usuario, lo que permite adaptar el sistema a cualquier Convenio Colectivo de Trabajo (CCT) sin necesidad de esperar actualizaciones del proveedor.[2]

Sin embargo, la transformación digital impulsada por la Administración Federal de Ingresos Públicos (AFIP), especialmente con la obligatoriedad del **Libro de Sueldos Digital (LSD)**, ha expuesto las limitaciones de las arquitecturas cliente-servidor tradicionales. Los contadores modernos requieren accesibilidad remota, colaboración en tiempo real y actualizaciones normativas automáticas que solo una arquitectura en la nube (SaaS - Software as a Service) puede ofrecer de manera eficiente.[3] El objetivo de este reporte es delinear la arquitectura técnica, los requisitos funcionales profundos y el plan de ejecución para desarrollar un sistema de liquidación web multitenant que replique y supere la flexibilidad lógica de Bejerman, garantizando al mismo tiempo una integración nativa y transparente con los servicios de AFIP y la normativa laboral vigente (LCT 20.744, Ley 24.241 y Ganancias 4ta Categoría).

La propuesta se centra en un sistema diseñado para el "Contador Multicliente", un perfil de usuario que no gestiona una sola nómina, sino decenas o cientos de empresas (tenants) simultáneamente. Esto impone requisitos no funcionales críticos de aislamiento de datos, escalabilidad y la capacidad de gestionar "fórmulas globales" (como el cálculo de la retención de jubilación) que se propaguen a todos los clientes, coexistiendo con "fórmulas locales" específicas de cada empresa.[5]

# 2. Análisis del Dominio y Requisitos Normativos

Para construir un "Bejerman Web", es imperativo deconstruir la lógica de negocio que subyace a la liquidación de haberes en Argentina. A diferencia de otros países donde el salario bruto se multiplica por tasas fijas, en Argentina la liquidación es un proceso algorítmico dependiente de variables temporales, acumuladores anuales y condiciones gremiales específicas.

## 2.1. El Paradigma del "Concepto" y la Flexibilidad Lógica

El núcleo de cualquier sistema de sueldos argentino competente es la entidad "Concepto". Un concepto no es simplemente una línea en el recibo de sueldo; es una unidad lógica autónoma que posee comportamiento, atributos fiscales y relaciones de dependencia. El análisis de la documentación de Bejerman revela que la flexibilidad del sistema radica en permitir al usuario definir si un concepto es remunerativo (sujeto a aportes), no remunerativo o de descuento, y asociarle una fórmula matemática que puede referenciar a otros conceptos.[2]

La arquitectura debe soportar tres categorías taxonómicas fundamentales de conceptos, las cuales determinan el tratamiento en el Libro de Sueldos Digital y en el Formulario 931:

- **Haberes Remunerativos:** Son aquellos montos sobre los cuales se calculan los aportes a la seguridad social (Jubilación, PAMI, Obra Social) y contribuciones patronales. Ejemplos incluyen el Sueldo Básico, Antigüedad, Adicionales de Convenio y Horas Extras. La Ley de Contrato de Trabajo (20.744) establece que estos conceptos forman la base de cálculo para el Sueldo Anual Complementario (SAC) y las indemnizaciones.[2]
- **Haberes No Remunerativos:** Pagos que, por acuerdos sindicales o decretos gubernamentales, están exentos de cargas sociales (total o parcialmente). Es crítico que el sistema permita configurar excepciones, ya que muchos acuerdos (como el de Comercio o UOM) estipulan que ciertos conceptos no remunerativos sí deben tributar Obra Social pero no Jubilación. Esta "excepción a la regla" es donde fallan los sistemas rígidos.[8]
- **Descuentos (Retenciones):** Deducciones aplicadas al bruto del empleado. Incluyen los descuentos de ley (11% SIPA, 3% Ley 19.032, 3% Obra Social), cuotas sindicales y el Impuesto a las Ganancias.

El sistema propuesto debe implementar un **Motor de Fórmulas** capaz de resolver dependencias complejas. Por ejemplo, el cálculo de la "Antigüedad" en el convenio de Empleados de Comercio no es un porcentaje fijo del básico, sino que se calcula sobre el básico *más* el presentismo. A su vez, el "Presentismo" se calcula sobre el básico *más* la antigüedad. Esta dependencia circular (A depende de B, B depende de A) se resuelve matemáticamente mediante bases teóricas, pero el sistema debe permitir al usuario modelar estas relaciones sin caer en errores de recursividad infinita.[1]

## 2.2. Marco Normativo de AFIP: El Libro de Sueldos Digital (LSD)

El cumplimiento con el Libro de Sueldos Digital no es una funcionalidad accesoria, sino el requisito de validación principal del sistema. AFIP exige que cada liquidación sea subida a su plataforma mediante un archivo de texto plano con un diseño de registro estricto. El análisis de las especificaciones técnicas de AFIP [9] indica que el sistema debe gestionar una capa de abstracción que "traduzca" los conceptos internos del usuario a la codificación estandarizada de AFIP.

La estructura de datos debe contemplar la generación de los siguientes registros obligatorios para la interfaz de importación del LSD:

**Tabla 1: Estructura de Registros para la Interfaz del Libro de Sueldos Digital**

| Registro | Descripción Técnica | Implicancia para el Diseño del Sistema |
|---|---|---|
| **Registro 01** | Cabecera del archivo. Contiene el CUIT de la empresa, el período fiscal (AAAAMM) y el tipo de liquidación (Mensual, Quincenal, Final). | El sistema debe validar que el CUIT del tenant coincida con el certificado digital o la configuración de la empresa antes de generar el archivo. |
| **Registro 02** | Datos del empleado en la liquidación. Incluye CBU, fecha de pago y fecha de rúbrica. | Se requiere almacenar el CBU y la fecha efectiva de pago en la tabla de liquidaciones, separada de la fecha contable. |
| **Registro 03** | Detalle de conceptos. Es el registro más crítico. Vincula el código interno del concepto, la cantidad (días/horas), el importe y la marca de Débito/Crédito. | El sistema debe poseer una tabla de mapeo (afip_concept_mapping) donde cada concepto creado por el usuario se asocie a un código AFIP (ej. 110000 para Sueldo). [12] |
| **Registro 04** | Datos para el F.931. Contiene la situación de revista, condición, | El sistema debe calcular automáticamente las bases imponibles 1 a 10 (SIPA, |

| | | |
|---|---|---|
| | actividad y bases imponibles. | PAMI, OS, etc.) sumando los conceptos remunerativos, aplicando los topes legales vigentes. [9] |

La complejidad radica en que AFIP valida la consistencia matemática entre el Registro 03 (la suma de los conceptos) y el Registro 04 (las bases imponibles declaradas). Si el sistema permite un error de redondeo de un centavo entre la suma de los ítems del recibo y el total declarado en el F.931, el archivo será rechazado. Por tanto, el motor de cálculo debe operar con precisión aritmética de punto fijo (Decimal), nunca con punto flotante.[13]

### 2.3. Impuesto a las Ganancias (4ta Categoría)

El cálculo de retención de Impuesto a las Ganancias es el desafío lógico más grande para cualquier sistema de sueldos en Argentina. A diferencia de las cargas sociales que son porcentajes directos del mes, Ganancias es un impuesto **anual acumulativo**. El sistema debe mantener una "memoria fiscal" del empleado que persista a través de los meses.[14]

El algoritmo requerido implica:

1. **Proyección de Ingresos:** Tomar el sueldo acumulado hasta el mes actual, sumarle el sueldo del mes corriente, y proyectar esa suma para los meses restantes del año fiscal.
2. **Deducciones Acumuladas:** Restar las deducciones generales (Jubilación, Obra Social) y las deducciones personales (Mínimo no imponible, Cargas de familia) acumuladas al mes.
3. **Aplicación de Escala:** Aplicar la tabla de alícuotas progresivas (Art. 94) sobre la ganancia neta sujeta a impuesto acumulada.
4. **Determinación del Impuesto:** Calcular el impuesto total anual estimado, restarle las retenciones ya efectuadas en meses anteriores del mismo año fiscal, y la diferencia es la retención del mes actual.

Este requerimiento dicta que la base de datos no puede simplemente guardar "liquidaciones cerradas"; debe tener una estructura de tablas agregadas (tax_accumulators) que se actualicen con cada cierre de liquidación, permitiendo recalcular el impuesto ante pagos retroactivos o ajustes.[14]

## 3. Arquitectura del Sistema Multitenant

Para satisfacer la necesidad de un sistema web escalable para un estudio contable que gestione múltiples clientes, la arquitectura debe basarse en un modelo **SaaS (Software as a Service)**. El patrón de diseño seleccionado es el de **Base de Datos Compartida con Esquemas Compartidos (Shared Database, Shared Schema)**, utilizando una columna discriminadora (tenant_id) en todas las tablas transaccionales. Este enfoque es superior al modelo de "Base de Datos por Tenant" en términos de eficiencia de costos y mantenibilidad

para miles de pequeñas empresas (PyMEs), que es el mercado objetivo típico de un contador en Argentina.[5]

## 3.1. Stack Tecnológico Propuesto

- **Servidor**: Linux + Apache (Tu entorno actual).
- **Backend**: PHP 8.x. Se utilizará Programación Orientada a Objetos (POO) para encapsular la lógica de las fórmulas.
- **Base de Datos**: MySQL. Motor relacional estándar.
- **Frontend**: HTML5 + Bootstrap 5. Para un diseño responsivo y limpio sin necesidad de compilar JavaScript.
- **Interacción**: jQuery (o JavaScript nativo) para las grillas de carga de novedades, enviando datos al servidor vía AJAX/Fetch.

## 3.2. Estrategia de Aislamiento de Datos (Multi-Tenancy)

El documento original sugiere "Base de datos compartida"1. En PHP/MySQL, esto se implementa así:

1. **Tabla** Empresas (Tenants): Cada empresa que liquidás tiene un ID único.

2. **Aislamiento Lógico**: Todas las tablas de datos (empleados, liquidaciones) deben tener una columna cliente_id.

3. **Seguridad en PHP**: En lugar de "Middleware", crearás una clase SessionManager. Al loguearse el contador y seleccionar una empresa, guardás el cliente_id en $_SESSION.

o **Regla de Oro**: Toda consulta SQL debe incluir WHERE cliente_id = $_SESSION['cliente_id'].

## 3.3. Diagrama de Entidad-Relación (ERD) y Diseño de Base de Datos

El esquema de base de datos debe reflejar la naturaleza flexible de los conceptos y la rigidez de las liquidaciones.

### 3.3.1. Tablas Nucleares

Tabla: tenants (Empresas)
Almacena la configuración fiscal y patronal.

| Columna | Tipo de Dato | Restricciones | Descripción |
|---|---|---|---|
| id | UUID | PK | Identificador único del cliente. |
| cuit | VARCHAR(11) | UNIQUE, NOT NULL | Clave fiscal, esencial para LSD y F.931. |
| razon_social | VARCHAR | NOT NULL | Nombre legal de la empresa. |
| cct_default_id | UUID | FK | Convenio colectivo por defecto (ej. Comercio). |
| tipo_empleador | INTEGER | NOT NULL | Código según tabla AFIP (ej. PyME, Privado). |

Tabla: conceptos (Definición Lógica)
Esta tabla es el corazón del motor de cálculo. Permite herencia: si tenant_id es NULL, es un "Concepto Global" (creado por el sistema, como Jubilación). Si tiene tenant_id, es una personalización del cliente.2

| Columna | Tipo de Dato | Descripción |
|---|---|---|
| id | UUID | Identificador del concepto. |

| | | |
|---|---|---|
| tenant_id | UUID (Nullable) | Si es NULL, aplica a todos los clientes. |
| codigo | VARCHAR(10) | Código visible (ej. "0100"). |
| nombre | VARCHAR(100) | Descripción en el recibo (ej. "Sueldo Básico"). |
| tipo | ENUM | REM (Remunerativo), NO_REM, DES (Descuento). |
| formula | TEXT | La lógica matemática (ej. BASICO * 0.01 * ANT). |
| codigo_afip | VARCHAR(6) | Mapeo al nomenclador LSD (ej. "110000"). [12] |
| orden_calculo | INTEGER | Prioridad de ejecución. |

Tabla: empleados (Legajos)
Debe soportar versionado histórico para manejar cambios de categoría retroactivos.23

| **Columna** | **Tipo de Dato** | **Descripción** |
|---|---|---|
| id | UUID | Identificador del legajo. |
| tenant_id | UUID | FK a la empresa. |
| cuil | VARCHAR(11) | Identificador laboral único. |
| fecha_ingreso | DATE | Crítica para cálculo de antigüedad. |
| categoria_id | UUID | FK a la escala salarial del CCT. |
| cbu | VARCHAR(22) | Para exportación bancaria. |
| estado | ENUM | ACTIVO, LICENCIA, BAJA. |

Tabla: novedades (Inputs Variables)
Almacena los valores que varían mes a mes, como horas extras o días de ausencia.

| Columna | Tipo de Dato | Descripción |
|---|---|---|
| id | UUID | PK. |
| liquidacion_id | UUID | FK a la liquidación abierta. |
| empleado_id | UUID | FK al empleado. |
| concepto_id | UUID | El concepto afectado (ej. Horas Extras 50%). |
| cantidad | DECIMAL(10,2) | El valor ingresado (ej. 5.5 horas). |

# 4. Diseño del Motor de Fórmulas y Cálculo

El requisito de "fórmulas personalizadas" exige un compilador de expresiones que entienda el contexto de la nómina argentina. No basta con aritmética simple; el motor debe resolver variables contextuales.

## 4.1. Sintaxis y Variables de Contexto

Se propone una sintaxis inspirada en Excel, familiar para los contadores, pero parseada en Python. El sistema expondrá un diccionario de variables globales y locales en tiempo de ej4.1. Lógica de "Pasadas"

1. P**aso 0 - Contexto**: Cargar datos fijos (Básico, Antigüedad calculada según fecha_ingreso).
2. **Paso 1 - Remunerativos**: Calcular todos los conceptos tipo REM. El sistema va sumando un acumulador $total_remunerativo.
3. **Paso 2 - No Remunerativos**: Calcular tipo NO_REM.
4. **Paso 3 - Descuentos**: Calcular tipo DES. Aquí las fórmulas pueden usar la variable TOTAL_REMUNERATIVO (que ya se calculó en el paso 1) para calcular Jubilación (11%), etc.
5. **Paso 4 - Neto**: Total REM + Total NO_REM - Total DES.

### 4.2. Seguridad (Reemplazo de asteval)

Como vimos en el ejemplo de código anterior:

- Usar str_replace para cambiar variables (BASICO -> 500000).
- Usar preg_match para prohibir letras y símbolos raros antes de ejecutar.
- Usar eval() solo si pasa la validación de seguridad.

## 5. Módulo de Integración con AFIP (LSD)

Esta es la funcionalidad que te dará clientes. AFIP exige un TXT estricto.

### 5.1. El Desafío

Debes generar un archivo donde cada línea tenga una longitud exacta. Si te falta un espacio, falla.

### 5.2. Solución PHP

PHP es excelente para esto. Usarás la función str_pad.

- Requisito AFIP: Importe de 15 caracteres (13 enteros, 2 decimales, sin punto)10.
- Código PHP:

| PHP |
| --- |
| // Ejemplo para convertir 12345.50 a "00000001234550"<br>$monto = 12345.50;<br>$monto_limpio = number_format($monto, 2, '', ''); // Quita el punto: "1234550"<br>$texto_afip = str_pad($monto_limpio, 15, "0", STR_PAD_LEFT); // Rellena con ceros<br>Deberás generar los Registros 01 (Cabecera), 02 (Liquidación), 03 (Detalle Conceptos) y 04 (Bases Imponibles) |

# 6. Hoja de Ruta (Roadmap) Sugerida

Dado que trabajás sola, dividiremos esto en "pequeñas victorias".

## Fase 1: El Núcleo Matemático (Mes 1)

- Objetivo: Lograr calcular un sueldo "en duro" (sin interfaz gráfica bonita).
- Tareas:

1. Crear la BD en MySQL (Tablas empleados, conceptos).
2. Programar la ClaseConcepto en PHP.
3. Crear un script test_liquidacion.php que tome un empleado y le calcule Jubilación y Obra Social.
4. Validar que los redondeos sean perfectos (2 decimales).

## Fase 2: Gestión de Datos (Mes 2)

- Objetivo: Que puedas cargar empleados y conceptos desde el navegador.
- Tareas:

1. Crear ABM (Alta/Baja/Modificación) de Empleados con Bootstrap.
2. Crear ABM de Conceptos. Aquí agregarás un campo de texto para escribir la fórmula.
3. Implementar el Login simple (Usuario/Contraseña) y la selección de Cliente (Tenant).

## Fase 3: Proceso de Liquidación (Mes 3)

- Objetivo: Generar recibos de sueldo reales.
- Tareas:

1. Pantalla "Nueva Liquidación": Seleccionar periodo y empleados.
2. Pantalla "Carga de Novedades": Una tabla HTML donde cargues horas extras manuales.
3. Botón "Liquidar": Ejecuta el motor PHP y guarda en liquidaciones_detalle.
4. Generar PDF del Recibo (usando librería FPDF o TCPDF, clásicos de PHP).

## Fase 4: Exportación y AFIP (Mes 4)

- Objetivo: Cumplir con la ley.
- Tareas:

1. Crear el botón "Exportar LSD".
2. Programar la generación del TXT usando str_pad.
3. Validar contra el aplicativo de prueba de AFIP.

## Fase 5: Ganancias 4ta Categoría (Futuro)

- Dejar esto para el final. Requiere tablas acumuladoras anuales13131313. Es lo más difícil, mejor tener el sistema básico funcionando primero.

# 7. Estrategia de Migración desde Bejerman

Dado que el cliente objetivo ya utiliza Bejerman, la barrera de entrada más alta es la migración de datos históricos. No se puede esperar que un contador cargue manualmente 500 legajos y sus acumulados.

## 7.1. Herramienta de Importación Masiva

Se debe desarrollar un "Asistente de Migración" que acepte los archivos de exportación estándar de Bejerman (generalmente DBF o Excel exportados).

- **Mapeo de Datos:** El asistente debe permitir mapear las columnas del Excel de Bejerman a los campos de la nueva base de datos (ej. Columna "F_INGRESO" -> empleados.fecha_ingreso).
- **Importación de Acumulados:** Es crucial importar los "Acumulados de Ganancias" del año en curso. Sin esto, el primer cálculo de impuestos en el nuevo sistema será erróneo. El sistema debe permitir ingresar "Saldos Iniciales" para cada empleado.[14]

# 8. Conclusión

El desarrollo de un sistema de liquidación de sueldos web multitenant para el mercado argentino es un desafío de alta complejidad técnica y regulatoria. La clave del éxito no reside en la interfaz visual, sino en la robustez del **Motor de Fórmulas** y la precisión absoluta en la generación de los archivos para el **Libro de Sueldos Digital**.

La arquitectura propuesta, basada en Python/Django para el procesamiento lógico seguro y React para una experiencia de usuario fluida, proporciona la base necesaria para competir con gigantes establecidos como Bejerman. Al adoptar un enfoque de "Base de Datos Compartida" con aislamiento lógico riguroso, el sistema será económicamente viable para comercializarse a estudios contables que gestionan carteras masivas de PyMEs, ofreciendo una solución moderna a un problema administrativo crónico. La implementación rigurosa de las validaciones de AFIP desde la etapa de diseño de datos garantizará que el producto no solo sea funcional, sino legalmente compliant desde el día uno.

### Fuentes citadas


1. Manual RRHH | PDF | Ventana (informática) | Gestión de recursos humanos - Scribd, acceso: diciembre 5, 2025, https://es.scribd.com/document/609977229/ManualRRHH
2. Sistemas Bejerman, acceso: diciembre 5, 2025, http://www.bejerman.com/Servicios/soporte/Faqs/sjw/55008old.htm
3. Sistema de liquidación de sueldos online - Xubio, acceso: diciembre 5, 2025, https://xubio.com/ar/sueldos
4. Bejerman Sueldos | Sistema de liquidación de sueldos - Thomson Reuters,

acceso: diciembre 5, 2025, https://www.thomsonreuters.com.ar/es/soluciones-fiscales-contables-gestion/soluciones-de-gestion-para-pymes/bejerman-sueldos.html
5. Multitenant SaaS database tenancy patterns - Azure - Microsoft Learn, acceso: diciembre 5, 2025, https://learn.microsoft.com/en-us/azure/azure-sql/database/saas-tenancy-app-design-patterns?view=azuresql
6. Multi-Tenant Database Architecture Patterns Explained - Bytebase, acceso: diciembre 5, 2025, https://www.bytebase.com/blog/multi-tenant-database-architecture-patterns-explained/
7. Sistemas Bejerman, acceso: diciembre 5, 2025, https://www.bejerman.com.ar/servicios/soporte/faqs/sj/34016.htm
8. Libro de Sueldos Digital - ARCA, acceso: diciembre 5, 2025, https://www.afip.gob.ar/librodesueldosdigital/documentos/nuevos/LSDetalleConceptos.pdf
9. Guía N.º 27 LSD: Ajustes para liquidaciones aceptadas INTRODUCCIÓN DESARROLLO, acceso: diciembre 5, 2025, https://www.afip.gob.ar/librodesueldosdigital/documentos/nuevos/g27-ajuste-de-liq-aceptadas.pdf
10. Libro de sueldos digital - Ayuda - Diseños - ARCA, acceso: diciembre 5, 2025, https://www.afip.gob.ar/librodesueldosdigital/ayuda/disenios.asp
11. La aplicación Libro de Sueldos Digital (LSD), cuenta con tres módulos, los que se presentan en la pantalla de inicio, acceso: diciembre 5, 2025, https://www.afip.gob.ar/LibrodeSueldosDigital/documentos/G03_Funciones_LSD.pdf
12. danwild/decision-tree-builder: Build serialisable flowchart-style decision trees with D3., acceso: diciembre 5, 2025, https://github.com/danwild/decision-tree-builder
13. Libro de Sueldos Digital - ARCA - Consulta Frecuentes, acceso: diciembre 5, 2025, https://servicioscf.afip.gob.ar/publico/abc/ABCpaso2.aspx?cat=2663
14. Como configurar tu módulo de sueldos en Bejerman ERP y Estudio ONE para cumplir con la AFIP RG 5008 - YouTube, acceso: diciembre 5, 2025, https://www.youtube.com/watch?v=2QW2_6IFRVA
15. Django Development Outsourcing Company in Argentina - Siblings Software, acceso: diciembre 5, 2025, https://siblingssoftware.com.ar/en/software-outsourcing-company/django-development/
16. Django Ledger is a double entry accounting system and financial analysis engine built on the Django Web Framework. - GitHub, acceso: diciembre 5, 2025, https://github.com/arrobalytics/django-ledger
17. jay3332/expr.py: A safe and simple math evaluator for Python, built with rply. - GitHub, acceso: diciembre 5, 2025, https://github.com/jay3332/expr.py
18. ASTEVAL: Minimal Python AST evaluator, acceso: diciembre 5, 2025, https://lmfit.github.io/asteval/

19. Monaco Editor Custom Language & Code Completion - Checkly, acceso: diciembre 5, 2025, https://www.checklyhq.com/blog/customizing-monaco/
20. Add a new language and its formatter to Monaco Editor · Issue #690 - GitHub, acceso: diciembre 5, 2025, https://github.com/microsoft/monaco-editor/issues/690
21. Managing object dependencies in PostgreSQL – Overview and helpful inspection queries (Part 1) | AWS Database Blog, acceso: diciembre 5, 2025, https://aws.amazon.com/blogs/database/managing-object-dependencies-in-postgresql-overview-and-helpful-inspection-queries-part-1/
22. Multi-Tenant Database Design Patterns 2024 - Daily.dev, acceso: diciembre 5, 2025, https://daily.dev/blog/multi-tenant-database-design-patterns-2024
23. Effective Dating Series Part I - The Problem – SQLServerCentral Forums, acceso: diciembre 5, 2025, https://www.sqlservercentral.com/forums/topic/effective-dating-series-part-i-the-problem/page/2
24. LSD Conceptos Basicos y Guia de Uso - ARCA, acceso: diciembre 5, 2025, https://www.afip.gob.ar/librodesueldosdigital/documentos/nuevos/LS_Conceptos_Basicos_y_Guia_de_Uso_V2.0.pdf
25. handsontable/formula-parser: Javascript Library parsing Excel Formulas and more - GitHub, acceso: diciembre 5, 2025, https://github.com/handsontable/formula-parser
26. Instructivo Acreditación Haberes Galicia | PDF | Archivo de computadora | Microsoft Excel, acceso: diciembre 5, 2025, https://es.scribd.com/document/711464841/Instructivo-acreditacion-de-haberes
27. DEPÓSITOS DE AHORRO, CUENTA SUELDO Y ESPECIALES Texto ordenado al 01/09/2021 - Banco Central, acceso: diciembre 5, 2025, https://www.bcra.gob.ar/pdfs/texord/texord_viejos/v-depaho_21-10-27.pdf