# **Sistema de Comisiones Comerciales - SERSA SAECA**

## **Documentación Técnica y Funcional Completa**

## **ÍNDICE RÁPIDO**

### **INICIO RÁPIDO**

* **1. Resumen Ejecutivo** → Descripción general
* **14. Instalación** → Cómo instalar y ejecutar

### **REGLAS CLAVE**

* **4. Reglas de Negocio** → Llaves críticas (LEER PRIMERO)
* **6. Fórmula de Cálculo** → Cálculo completo paso a paso

### **CONFIGURACIÓN**

* **3. Niveles y Estructura** → Configuración de niveles y pagos
* **5. Multiplicadores** → Sistema de bonificaciones/penalizaciones

### **ARQUITECTURA**

* **2. Arquitectura Técnica** → Estructura y tecnologías
* **13. Arquitectura Modular** → Organización del código
* **15. Estándares de Código** → Patrones y mejores prácticas

### **DESARROLLO**

* **7. Interfaz de Usuario** → Layout y componentes
* **10. Consideraciones Técnicas** → Performance y compatibilidad
* **16. Testing y Calidad** → Estrategia de pruebas

### **FUNCIONALIDADES**

* **8. Funcionalidades Especiales** → Features avanzados
* **9. Casos de Uso** → Ejemplos prácticos
* **19. Sistema de Perfiles** → Múltiples configuraciones
* **20. Panel de Administración** → Gestión de configuración
* **21. Generación de PDF Profesional** → Reportes ejecutivos

### **MANTENIMIENTO**

* **11. Mantenimiento** → Cómo hacer cambios
* **17. Documentación del Código** → JSDoc y diagramas

## **1. RESUMEN EJECUTIVO**

### **1.1 Descripción General**

Aplicación web para el cálculo automático de comisiones de asesores comerciales de SERSA SAECA. Permite calcular pagos mensuales basados en múltiples indicadores de desempeño, incluyendo volumen de ventas, calidad de servicio y trabajo en equipo.

### **1.2 Características Principales**

* Cálculo automático de comisiones con múltiples componentes
* Sistema de niveles progresivos (6 niveles: Capilla a Genio)
* Multiplicadores de calidad que afectan la comisión final
* Interfaz visual con barras de progreso interactivas
* Sugerencias personalizadas para mejorar ingresos
* Guardado automático de datos (borrador)
* Generación de PDF para impresión

## **2. ARQUITECTURA TÉCNICA**

### **2.1 Estructura de Archivos Requerida**

La aplicación DEBE seguir una arquitectura modular con la siguiente estructura:

commission-calculator/

├── src/

│ ├── modules/

│ │ ├── config.js # Configuración y constantes

│ │ ├── calculations.js # Lógica de cálculo pura

│ │ ├── validators.js # Validaciones de datos

│ │ ├── ui.js # Manipulación del DOM

│ │ ├── storage.js # Persistencia local

│ │ └── state.js # Gestión de estado

│ ├── styles/

│ │ ├── main.css # Estilos principales

│ │ ├── components.css # Estilos de componentes

│ │ └── responsive.css # Media queries

│ ├── main.js # Punto de entrada

│ └── index.html # HTML principal

├── tests/

│ ├── unit/

│ │ ├── calculations.test.js

│ │ └── validators.test.js

│ └── integration/

│ └── commission.test.js

├── docs/

│ ├── architecture.md

│ └── api.md

├── LICENSE

├── README.md

└── package.json

### **2.2 Tecnologías y Principios**

* **Frontend**: JavaScript ES6+ modular, CSS3 con custom properties, HTML5 semántico
* **Arquitectura**: Modular con separación de responsabilidades
* **Estado**: Gestión centralizada con patrón Observer
* **Persistencia**: LocalStorage con versionado
* **Testing**: Jest para tests unitarios e integración
* **Build**: Sin transpilación para desarrollo, opcional para producción
* **Patrones**: Factory, Strategy, Observer, Module Pattern

## **3. SISTEMA DE NIVELES Y ESTRUCTURA**

### **3.1 Niveles de Carrera**

****const niveles = ['Capilla', 'Junior', 'Senior A', 'Senior B', 'Máster', 'Genio'];

const iconos = ['🏠', '👤', '⭐', '💎', '👑', '🏆'];

### **3.2 Metas por Nivel**

****const metas = {

montoInterno: [600000000, 800000000, 900000000, 1000000000, 1100000000, 1200000000],

montoExterno: [50000000, 100000000, 150000000, 200000000, 300000000, 400000000],

montoRecuperado: [40000000, 60000000, 80000000, 100000000, 120000000, 150000000],

cantidad: [6, 8, 9, 10, 12, 13]

};

### **3.3 Sistema de Pagos**

****const pagos = {

base: 3000000, // Base fija para todos

carrera: [0, 0, 500000, 1000000, 1500000, 2000000], // Por nivel alcanzado

montoInterno: [500000, 600000, 1000000, 1400000, 2000000, 2500000],

montoExterno: [800000, 1000000, 1500000, 2000000, 2500000, 3300000],

montoRecuperado: [300000, 400000, 500000, 600000, 800000, 1000000],

cantidad: [0, 400000, 600000, 700000, 1000000, 1200000],

equipo: [0, 0, 0, 500000, 800000, 1000000] // Solo desde Senior B

};



## **4. REGLAS DE NEGOCIO CRÍTICAS**

### **4.1 Sistema de Llaves**

#### **4.1.1 Llave de Montos (6 desembolsos)**

* **Requisito**: Mínimo 6 desembolsos en el mes
* **Afecta a**: Premios de Monto Interno, Externo y Recuperado
* **Si no cumple**: NO cobra ningún premio de montos (aunque alcance las metas)

#### **4.1.2 Llave Semanal (2/semana)**

* **Requisito**: Mínimo 2 desembolsos en la peor semana del mes
* **Afecta a**: Premio de Cantidad
* **Comportamiento**:
  + < 2/sem: Sin premio de cantidad (nivel = -1)
  + ≥ 2/sem: Premio completo según nivel alcanzado

### **4.2 Cálculo de Nivel de Carrera**

1. Se determina el nivel alcanzado en cada categoría (Interno, Externo, Recuperado, Cantidad)
2. El nivel de carrera del mes = MENOR nivel entre las 4 categorías
3. El nivel final = MENOR entre el nivel del mes actual y el mes anterior
4. Este nivel determina el "Premio Carrera"

### **4.3 Premio de Equipo**

* **Requisitos**:
  + El asesor debe estar en Senior A o superior
  + El menor nivel del equipo debe ser Senior A o superior
* **Cálculo**: Se paga según el menor nivel del equipo

## **5. SISTEMA DE MULTIPLICADORES**

### **5.1 Tablas de Multiplicadores**

#### **5.1.1 Conversión**

****{min: 10, mult: 1.1}, // 10%+ → 110%

{min: 8, mult: 1.0}, // 8-9% → 100%

{min: 7, mult: 0.8}, // 7% → 80%

{min: 6, mult: 0.7}, // 6% → 70%

{min: 5, mult: 0.6}, // 5% → 60%

{min: 4, mult: 0.5}, // 4% → 50%

{min: 0, mult: 0.3} // <4% → 30%

#### **5.1.2 Empatía**

****{min: 96, mult: 1.0}, // 96%+ → 100%

{min: 90, mult: 0.9}, // 90-95% → 90%

{min: 80, mult: 0.5}, // 80-89% → 50%

{min: 70, mult: 0.3}, // 70-79% → 30%

{min: 0, mult: 0} // <70% → 0%

#### **5.1.3 Proceso**

****{min: 95, mult: 1.0}, // 95%+ → 100%

{min: 90, mult: 0.95}, // 90-94% → 95%

{min: 85, mult: 0.8}, // 85-89% → 80%

{min: 70, mult: 0.3}, // 70-84% → 30%

{min: 0, mult: 0} // <70% → 0%

#### **5.1.4 Mora (inverso - penaliza)**

****{min: 0, mult: 1.05}, // 0-2% → 105% (bonifica)

{min: 3, mult: 0.95}, // 3-7% → 95%

{min: 8, mult: 0.9}, // 8-9% → 90%

{min: 10, mult: 0.85}, // 10-14% → 85%

{min: 15, mult: 0.7} // 15%+ → 70%

### **5.2 Cálculo del Multiplicador Total**

****multiplicadorTotal = conversion × empatia × proceso × mora

* Se aplica solo a la parte variable (todo menos la base fija)
* Mínimo: 0.1 (10%) para evitar valores negativos

## **6. FÓRMULA DE CÁLCULO COMPLETA**

### **6.1 Componentes**

1. **Base Fija**: 3,000,000 Gs (siempre)
2. **Premio Carrera**: Según nivel de carrera (0 a 2,000,000)
3. **Premio Monto Interno**: Si cumple llave de 6 desembolsos
4. **Premio Monto Externo**: Si cumple llave de 6 desembolsos
5. **Premio Recuperados**: Si cumple llave de 6 desembolsos
6. **Premio Cantidad**: Si cumple llave semanal (2/sem)
7. **Premio Equipo**: Si asesor y equipo ≥ Senior A

### **6.2 Cálculo Final**

****subtotal = base + carrera + interno + externo + recuperado + cantidad + equipo

parteVariable = subtotal - base

totalVariable = parteVariable × multiplicadorTotal

comisionFinal = base + totalVariable



## **7. INTERFAZ DE USUARIO**

### **7.1 Layout**

* **Panel Izquierdo (320px)**:
  + Header con título y botones de acción
  + Formulario de entrada de datos
  + Campos obligatorios marcados con asterisco rojo
  + Valores predeterminados optimistas en campos de calidad
* **Panel Derecho (flexible)**:
  + Estadísticas principales en la parte superior
  + Barras de progreso clickeables para cada categoría
  + Tablas de multiplicadores interactivas
  + Cálculo detallado de comisión
  + Sugerencias personalizadas

### **7.2 Características Visuales**

* **Barras de Progreso**:
  + Segmentadas por nivel (6 segmentos)
  + Click en segmento carga automáticamente ese valor
  + Colores: gris (no alcanzado), verde (alcanzado), azul oscuro (actual)
* **Campos de Entrada**:
  + Borde verde: campo lleno
  + Borde naranja + fondo amarillo: campo vacío no crítico
  + Borde rojo: campo requerido vacío
* **Multiplicadores**:
  + Tablas con código de color (verde/amarillo/rojo)
  + Filas clickeables para cargar valores
  + Resaltado de rango actual

### **7.3 Responsive**

* < 1200px: Paneles apilados verticalmente
* < 992px: Sidebar colapsable con botón toggle
* < 768px: Layout optimizado para móvil

## **8. FUNCIONALIDADES ESPECIALES**

### **8.1 Guardado Automático**

****// Guarda en localStorage cada 500ms después de cambios

localStorage.setItem('draftCommission', JSON.stringify(draft));

* Indicador visual "⚡ Guardando..."
* Restaura valores al recargar página

### **8.2 Validaciones**

* **Recuperados**: Alerta si superan 50% del monto interno
* **Rangos de porcentajes**: 0-100%
* **Formato de montos**: Separador de miles automático
* **Límites**: Hasta 10 mil millones en montos

### **8.3 Sugerencias Inteligentes**

El sistema analiza los datos y genera 5 tipos de sugerencias:

1. **Limitante Principal** (🚨): Identifica qué indicador limita la carrera
2. **Oportunidades Rápidas** (💰): Metas cercanas (< 20% faltante)
3. **Mejora de Multiplicadores** (⚡): Sugiere mejorar el multiplicador más bajo
4. **Objetivos Motivacionales** (🎯): Proyecciones de ingresos futuros
5. **Alertas** (⚠️): Problemas críticos (llaves no cumplidas, mora alta)

### **8.4 Generación de PDF**

* Abre ventana nueva con resumen formateado
* Incluye todos los datos relevantes
* Listo para imprimir con estilos optimizados

## **9. CASOS DE USO Y EJEMPLOS**

### **9.1 Caso Base (Senior A completo)**

****Entrada:

- Nivel anterior: Senior A

- Monto Interno: 900,000,000

- Monto Externo: 150,000,000

- Recuperados: 80,000,000

- Cantidad: 9

- Menor semana: 2

- Calidad: Conv 8%, Emp 96%, Proc 95%, Mora 2%

Resultado:

- Nivel carrera: Senior A

- Subtotal: 3,000,000 + 500,000 + 1,000,000 + 1,500,000 + 500,000 + 600,000 = 7,100,000

- Multiplicador: 1.0 × 1.0 × 1.0 × 1.05 = 105%

- Total: 3,000,000 + (4,100,000 × 1.05) = 7,305,000 Gs

### **9.2 Caso con Penalizaciones**

****Sin llave de montos (5 desembolsos):

- Pierde premios de Interno, Externo y Recuperado

- Solo cobra: Base + Carrera + Cantidad (si cumple 2/sem)

Multiplicadores bajos:

- Conversión 3%: × 0.3

- Empatía 69%: × 0

- Total: 0% (solo cobra base fija)



## **10. CONSIDERACIONES TÉCNICAS**

### **10.1 Performance**

La aplicación DEBE implementar estas optimizaciones:

#### **10.1.1 Optimizaciones de Renderizado**

****// Debounce para inputs con cálculos costosos

const debounce = (func, wait) => {

let timeout;

return function executedFunction(...args) {

const later = () => {

clearTimeout(timeout);

func(...args);

};

clearTimeout(timeout);

timeout = setTimeout(later, wait);

};

};

// RequestAnimationFrame para actualizaciones visuales

const updateUI = (changes) => {

requestAnimationFrame(() => {

// Batch DOM updates

changes.forEach(change => change());

});

};

#### **10.1.2 Memoización de Cálculos**

****const memoize = (fn) => {

const cache = new Map();

return (...args) => {

const key = JSON.stringify(args);

if (cache.has(key)) return cache.get(key);

const result = fn(...args);

cache.set(key, result);

return result;

};

};

// Aplicar a cálculos costosos

const calcularNivelMemo = memoize(calcularNivel);

#### **10.1.3 Lazy Loading**

* Cargar módulos bajo demanda
* Diferir carga de funcionalidades no críticas
* Optimizar el Critical Rendering Path

### **10.2 Seguridad**

La aplicación DEBE implementar estas medidas de seguridad:

#### **10.2.1 Validación y Sanitización**

****// Sanitizar todos los inputs

const sanitizeInput = (value, type) => {

if (type === 'number') {

return parseInt(value.toString().replace(/[^0-9]/g, ''), 10) || 0;

}

if (type === 'text') {

return value.toString().replace(/[<>]/g, '');

}

return value;

};

// Validar rangos estrictos

const validateRange = (value, min, max) => {

const num = Number(value);

if (isNaN(num)) return min;

return Math.max(min, Math.min(max, num));

};

#### **10.2.2 Protección de Datos**

* No almacenar información sensible en localStorage
* Implementar límites de tamaño para prevenir DoS local
* Validar integridad de datos almacenados

### **10.3 Compatibilidad**

#### **10.3.1 Navegadores Soportados**

* Chrome 90+
* Firefox 88+
* Safari 14+
* Edge 90+
* Opera 76+

#### **10.3.2 Polyfills Requeridos**

****// Para navegadores antiguos (opcional)

if (!String.prototype.includes) {

String.prototype.includes = function(search, start) {

return this.indexOf(search, start) !== -1;

};

}

### **10.4 Accesibilidad (A11y)**

La aplicación DEBE ser accesible:

<!-- ARIA labels obligatorios -->

<input

type="text"

id="montoInterno"

aria-label="Monto interno en guaraníes"

aria-required="true"

aria-invalid="false"

aria-describedby="montoInterno-error"

/>

<!-- Navegación por teclado -->

<div

class="progress-segment"

role="button"

tabindex="0"

aria-label="Seleccionar nivel Senior A"

onkeypress="if(event.key === 'Enter') selectLevel('senior-a')"

>

### **10.5 Internacionalización (i18n)**

Preparado para múltiples idiomas:

const i18n = {

es: {

'commission.base': 'Base fija',

'commission.total': 'Comisión total',

'level.capilla': 'Capilla'

},

en: {

'commission.base': 'Fixed base',

'commission.total': 'Total commission',

'level.capilla': 'Chapel'

}

};

const t = (key) => i18n[currentLang][key] || key;



## **11. INSTALACIÓN Y DESPLIEGUE**

### **11.1 Desarrollo Local**

****# Clonar repositorio

git clone [repository-url]

cd commission-calculator

# Instalar dependencias de desarrollo

npm install

# Ejecutar tests

npm test

# Ejecutar con servidor de desarrollo

npm run dev

# Build para producción

npm run build

### **11.2 Package.json Requerido**

****{

"name": "commission-calculator",

"version": "1.0.0",

"description": "Sistema de Comisiones Comerciales SERSA SAECA",

"type": "module",

"scripts": {

"dev": "serve src",

"test": "jest",

"test:watch": "jest --watch",

"test:coverage": "jest --coverage",

"build": "node scripts/build.js",

"lint": "eslint src/\*\*/\*.js"

},

"devDependencies": {

"jest": "^27.0.0",

"eslint": "^8.0.0",

"serve": "^14.0.0"

}

}

### **11.3 Configuración de Build**

****// scripts/build.js

import fs from 'fs';

import path from 'path';

const build = async () => {

// Crear directorio dist

const distDir = './dist';

if (!fs.existsSync(distDir)) {

fs.mkdirSync(distDir);

}

// Concatenar módulos en orden correcto

const modules = [

'config.js',

'validators.js',

'calculations.js',

'storage.js',

'state.js',

'ui.js',

'main.js'

];

let bundled = '';

for (const module of modules) {

const content = fs.readFileSync(`./src/modules/${module}`, 'utf8');

bundled += content + '\n';

}

// Generar archivo final

fs.writeFileSync('./dist/app.js', bundled);

// Copiar archivos estáticos

fs.copyFileSync('./src/index.html', './dist/index.html');

// Concatenar CSS

const styles = ['main.css', 'components.css', 'responsive.css'];

let css = '';

for (const style of styles) {

css += fs.readFileSync(`./src/styles/${style}`, 'utf8') + '\n';

}

fs.writeFileSync('./dist/styles.css', css);

console.log('✅ Build completado en ./dist');

};

build();

### **11.4 Despliegue en Producción**

#### **11.4.1 Servidor Web Estático**

****# Nginx configuration

server {

listen 80;

server\_name comisiones.example.com;

root /var/www/commission-calculator/dist;

location / {

try\_files $uri $uri/ /index.html;

}

# Cache estático

location ~\* \.(js|css|png|jpg|jpeg|gif|ico|svg)$ {

expires 1y;

add\_header Cache-Control "public, immutable";

}

}

#### **11.4.2 CDN y Optimización**

****<!-- index.html optimizado -->

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta name="description" content="Sistema de Comisiones Comerciales SERSA SAECA">

<!-- Preload crítico -->

<link rel="preload" href="styles.css" as="style">

<link rel="preload" href="app.js" as="script">

<!-- CSS -->

<link rel="stylesheet" href="styles.css">

<!-- Favicons -->

<link rel="icon" type="image/png" href="favicon.png">

</head>

<body>

<!-- Contenido -->

<!-- JavaScript al final -->

<script src="app.js" defer></script>

</body>

</html>

### **11.5 Configuración de Entorno**

****// config/environments.js

const environments = {

development: {

debug: true,

apiUrl: 'http://localhost:3000',

storage: 'localStorage'

},

staging: {

debug: false,

apiUrl: 'https://staging-api.example.com',

storage: 'localStorage'

},

production: {

debug: false,

apiUrl: 'https://api.example.com',

storage: 'localStorage',

analytics: true

}

};

export default environments[process.env.NODE\_ENV || 'development'];

### **11.6 Monitoreo y Analytics**

****// Integración con Google Analytics o similar

if (config.analytics) {

window.dataLayer = window.dataLayer || [];

function gtag(){dataLayer.push(arguments);}

gtag('js', new Date());

gtag('config', 'GA\_MEASUREMENT\_ID');

// Track eventos importantes

gtag('event', 'commission\_calculated', {

'event\_category': 'engagement',

'event\_label': 'total\_amount',

'value': calculatedCommission

});

}



## **12. MANTENIMIENTO Y ACTUALIZACIONES**

### **12.1 Proceso de Cambios**

Para mantener la integridad del sistema, TODOS los cambios deben seguir este proceso:

#### **12.1.1 Cambios en Configuración**

****// src/modules/config.js

// Para cambiar metas o premios:

// 1. Actualizar el objeto CONFIG

export const CONFIG = {

// Versión de configuración (incrementar con cada cambio)

version: '1.2.0',

// Modificar valores aquí

metas: {

montoInterno: [600000000, 800000000, ...], // Nuevos valores

},

// Agregar changelog

changelog: {

'1.2.0': 'Actualización de metas de monto interno',

'1.1.0': 'Ajuste de multiplicadores de mora'

}

};

// 2. Actualizar tests correspondientes

// 3. Documentar el cambio en CHANGELOG.md

#### **12.1.2 Migración de Datos**

****// src/modules/migrations.js

const migrations = {

'1.0.0': (data) => data, // Sin cambios

'1.1.0': (data) => {

// Ejemplo: agregar nuevo campo

return {

...data,

nuevosCampo: 0

};

},

'1.2.0': (data) => {

// Ejemplo: cambiar estructura

if (data.montoTotal) {

data.montoInterno = data.montoTotal;

delete data.montoTotal;

}

return data;

}

};

export const migrate = (data, fromVersion, toVersion) => {

let migrated = data;

const versions = Object.keys(migrations).sort();

for (const version of versions) {

if (version > fromVersion && version <= toVersion) {

migrated = migrations[version](migrated);

}

}

return migrated;

};

### **12.2 Tipos de Cambios Comunes**

#### **12.2.1 Agregar Nuevo Nivel**

****// 1. Actualizar config.js

export const CONFIG = {

niveles: ['Capilla', 'Junior', 'Senior A', 'Senior B', 'Máster', 'Genio', 'Leyenda'], // Nuevo nivel

metas: {

montoInterno: [...existentes, 1500000000], // Nueva meta

},

pagos: {

montoInterno: [...existentes, 3000000], // Nuevo premio

}

};

// 2. Actualizar UI para mostrar 7 segmentos

// 3. Actualizar tests

// 4. Verificar cálculos de carrera

#### **12.2.2 Modificar Multiplicadores**

****// 1. En config.js

multiplicadores: {

conversion: [

{min: 12, mult: 1.2}, // Nuevo rango superior

{min: 10, mult: 1.1},

// ... resto igual

]

}

// 2. Actualizar documentación

// 3. Notificar a usuarios del cambio

#### **12.2.3 Agregar Nueva Categoría de Premio**

****// 1. Definir en config.js

pagos: {

...existing,

nuevaCategoria: [0, 0, 100000, 200000, 300000, 400000]

}

// 2. Implementar lógica en calculations.js

calculateNuevaCategoria(data) {

// Lógica específica

}

// 3. Agregar a UI

// 4. Actualizar fórmula total

// 5. Agregar tests

### **12.3 Versionado Semántico**

La aplicación DEBE seguir versionado semántico:

* **MAJOR** (X.0.0): Cambios incompatibles en reglas de negocio
* **MINOR** (0.X.0): Nuevas funcionalidades compatibles
* **PATCH** (0.0.X): Corrección de bugs

// package.json

{

"version": "1.2.3",

// Mayor.Minor.Patch

}

### **12.4 Testing de Regresión**

Antes de cada release:

# 1. Ejecutar suite completa

npm test

# 2. Verificar cobertura (mínimo 80%)

npm run test:coverage

# 3. Test manual con casos extremos

# - Todos los niveles

# - Valores límite

# - Combinaciones de llaves

# - Multiplicadores extremos

# 4. Test de migración

npm run test:migrations

### **12.5 Documentación de Cambios**

****# CHANGELOG.md

## [1.2.0] - 2024-03-15

### Added

- Nuevo nivel "Leyenda" con metas superiores

- Exportación a Excel

### Changed

- Metas de monto interno aumentadas 10%

- Multiplicador de conversión mejorado para rangos altos

### Fixed

- Cálculo incorrecto de mora en casos extremos

- Error de redondeo en totales grandes

### Breaking Changes

- Ninguno (retrocompatible)

### **12.6 Procedimiento de Hotfix**

Para correcciones urgentes:

# 1. Crear rama de hotfix

git checkout -b hotfix/calculation-error

# 2. Implementar fix mínimo

# 3. Test específico del bug

# 4. Test de regresión rápido

# 5. Deploy directo a producción

npm run build

npm run deploy:production

# 6. Merge a main

git checkout main

git merge hotfix/calculation-error

### **12.7 Monitoreo Post-Actualización**

****// Agregar telemetría para detectar problemas

window.addEventListener('error', (event) => {

const errorInfo = {

message: event.message,

source: event.filename,

line: event.lineno,

column: event.colno,

stack: event.error?.stack,

version: CONFIG.version,

timestamp: new Date().toISOString()

};

// Enviar a servicio de logging

if (window.analytics) {

window.analytics.track('javascript\_error', errorInfo);

}

});



## **13. ARQUITECTURA MODULAR REQUERIDA**

### **13.1 Estructura de Módulos**

La aplicación DEBE estar organizada en módulos separados para mantener el principio de responsabilidad única:

// config.js - Configuración y constantes

export const CONFIG = {

niveles: ['Capilla', 'Junior', 'Senior A', 'Senior B', 'Máster', 'Genio'],

metas: { /\* ... \*/ },

pagos: { /\* ... \*/ },

multiplicadores: { /\* ... \*/ },

ui: {

colors: {

primary: '#006D77',

success: '#4CAF50',

warning: '#FFA726',

danger: '#F44336'

},

animations: {

duration: 300,

easing: 'ease-in-out'

}

}

};

// calculations.js - Lógica de cálculo pura

export const Calculations = {

calcularNivel(monto, tipo) { /\* ... \*/ },

calcularMultiplicador(tipo, valor) { /\* ... \*/ },

calcularComisionTotal(data) { /\* ... \*/ }

};

// validators.js - Validaciones

export const Validators = {

schemas: {

montoInterno: { min: 0, max: 10000000000, required: true },

conversion: { min: 0, max: 100, required: true }

},

validate(field, value) { /\* ... \*/ }

};

// ui.js - Manipulación del DOM

export const UI = {

updateProgressBar(tipo, valor, containerId) { /\* ... \*/ },

showError(field, message) { /\* ... \*/ },

updateCalculationDisplay(results) { /\* ... \*/ }

};

// storage.js - Persistencia de datos

export const Storage = {

KEYS: { draft: 'draftCommission', history: 'commissionHistory' },

save(data) { /\* ... \*/ },

load() { /\* ... \*/ },

clear() { /\* ... \*/ }

};

### **13.2 Gestión de Estado**

La aplicación DEBE implementar un gestor de estado centralizado:

// state.js

export const AppState = {

\_state: {

formData: {},

calculations: {},

ui: { loading: false, errors: {} }

},

\_listeners: [],

subscribe(callback) {

this.\_listeners.push(callback);

return () => this.\_listeners = this.\_listeners.filter(l => l !== callback);

},

update(path, value) {

// Actualizar estado inmutablemente

this.\_state = this.\_updatePath(this.\_state, path, value);

this.\_notify();

},

get(path) {

return this.\_getPath(this.\_state, path);

},

\_notify() {

this.\_listeners.forEach(listener => listener(this.\_state));

}

};

### **13.3 API Pública**

La aplicación DEBE exponer una API consistente:

// main.js

export const CommissionCalculator = {

initialize(config = {}) {

this.config = { ...CONFIG, ...config };

this.state = AppState;

this.bindEvents();

this.loadDraft();

},

calculate(data) {

const validation = this.validate(data);

if (!validation.isValid) {

return { error: validation.errors };

}

return Calculations.calcularComisionTotal(data);

},

reset() {

Storage.clear();

this.state.update('formData', {});

UI.resetForm();

},

export(format = 'pdf') {

const data = this.state.get('calculations');

return Exporters[format](data);

}

};



## **14. INSTALACIÓN Y CONFIGURACIÓN**

### **14.1 Instalación Básica**

****# Opción 1: Uso directo

1. Descargar los archivos

2. Abrir index.html en el navegador

# Opción 2: Con servidor local

npx serve .

# o

python -m http.server 8000

### **14.2 Instalación Modular (Recomendada)**

****# Estructura de proyecto

commission-calculator/

├── src/

│ ├── modules/

│ │ ├── config.js

│ │ ├── calculations.js

│ │ ├── validators.js

│ │ ├── ui.js

│ │ └── storage.js

│ ├── styles/

│ │ ├── main.css

│ │ ├── components.css

│ │ └── responsive.css

│ ├── main.js

│ └── index.html

├── tests/

│ ├── calculations.test.js

│ └── validators.test.js

├── docs/

│ └── architecture.md

└── package.json

### **14.3 Configuración Personalizada**

****// custom-config.js

CommissionCalculator.initialize({

niveles: ['Nivel 1', 'Nivel 2', ...], // Personalizar niveles

metas: { /\* valores custom \*/ },

ui: {

colors: {

primary: '#YOUR\_COLOR'

}

}

});



## **15. ESTÁNDARES DE CÓDIGO Y PATRONES**

### **15.1 Principios de Diseño**

La aplicación DEBE seguir estos principios:

1. **DRY (Don't Repeat Yourself)**: Reutilizar código común
2. **SOLID**: Especialmente Single Responsibility y Open/Closed
3. **KISS**: Mantener las soluciones simples
4. **Composición sobre herencia**: Usar funciones componibles

### **15.2 Patrones Implementados**

#### **15.2.1 Observer Pattern**

****// Para actualización reactiva de UI

class Observable {

constructor() {

this.observers = [];

}

subscribe(fn) {

this.observers.push(fn);

}

notify(data) {

this.observers.forEach(fn => fn(data));

}

}

#### **15.2.2 Factory Pattern**

****// Para crear diferentes tipos de multiplicadores

const MultiplierFactory = {

create(type, config) {

const strategies = {

'conversion': new ConversionMultiplier(config),

'empatia': new EmpathyMultiplier(config),

'proceso': new ProcessMultiplier(config),

'mora': new MoraMultiplier(config)

};

return strategies[type];

}

};

#### **15.2.3 Strategy Pattern**

****// Para diferentes estrategias de cálculo

class CalculationStrategy {

calculate(data) {

throw new Error('Must implement calculate method');

}

}

class InternalAmountStrategy extends CalculationStrategy {

calculate(data) {

if (!data.meetsKeyRequirement) return 0;

return this.getPaymentForLevel(data.level);

}

}

### **15.3 Convenciones de Código**

#### **15.3.1 Nomenclatura**

****// Constantes: UPPER\_SNAKE\_CASE

const MAX\_RETRIES = 3;

const DEFAULT\_TIMEOUT = 5000;

// Clases: PascalCase

class CommissionCalculator { }

// Funciones y variables: camelCase

function calculateTotal() { }

let currentLevel = 0;

// Privadas: prefijo underscore

\_privateMethod() { }

// Eventos: prefijo 'on'

onInputChange() { }

#### **15.3.2 Estructura de Funciones**

****// Funciones DEBEN ser pequeñas y hacer una sola cosa

// Máximo 20 líneas por función

function calculateInternalBonus(amount, level, hasKey) {

// Validación primero

if (!hasKey) return 0;

if (level < 0 || level > 5) return 0;

// Lógica principal

const bonus = PAYMENT\_TABLE.internal[level];

// Return explícito

return bonus;

}

### **15.4 Manejo de Errores**

****// Toda operación crítica DEBE tener manejo de errores

class ErrorHandler {

static handle(error, context) {

console.error(`Error in ${context}:`, error);

// Notificar al usuario apropiadamente

if (error instanceof ValidationError) {

UI.showValidationError(error.field, error.message);

} else if (error instanceof CalculationError) {

UI.showCalculationError(error.message);

} else {

UI.showGenericError();

}

// Log para debugging

this.logError(error, context);

}

static logError(error, context) {

const errorLog = {

timestamp: new Date().toISOString(),

context,

error: error.message,

stack: error.stack,

userAgent: navigator.userAgent

};

// En producción, enviar a servicio de logging

console.table(errorLog);

}

}



## **16. TESTING Y ASEGURAMIENTO DE CALIDAD**

### **16.1 ¿Qué es Testing y Por Qué es Importante?**

**Testing = Probar que todo funciona correctamente**

Imagina que construyes un auto. Antes de venderlo, necesitas probar:

* ¿Los frenos funcionan?
* ¿Las luces prenden?
* ¿El motor arranca?

Lo mismo con el software. El testing verifica que:

* Los cálculos dan resultados correctos
* Los botones hacen lo que deben hacer
* No hay errores cuando ingresas datos

### **16.2 Guía PASO A PASO para Testing (Para No Programadores)**

#### **PASO 1: Preparar el Entorno de Testing**

****# En la carpeta del proyecto, abrir terminal y escribir:

npm install --save-dev jest

# Esto instala Jest, una herramienta para hacer tests

#### **PASO 2: Crear Carpeta de Tests**

****commission-calculator/

├── src/

├── tests/ # ← Crear esta carpeta

│ ├── manual/ # ← Tests manuales (checklist)

│ └── automatic/ # ← Tests automáticos

#### **PASO 3: Tests Manuales (Los Más Importantes)**

Crear archivo tests/manual/checklist.md:

# CHECKLIST DE PRUEBAS MANUALES

## 🟢 PRUEBAS BÁSICAS (Hacer siempre)

### 1. Prueba de Carga Inicial

- [ ] Abrir la aplicación

- [ ] ¿Aparece el selector de perfiles?

- [ ] ¿Los campos tienen valores predeterminados?

- [ ] ¿No hay errores en la consola? (F12 → Console)

### 2. Prueba de Cálculo Básico

- [ ] Seleccionar perfil "Ágil 1"

- [ ] Ingresar estos valores exactos:

- Monto Interno: 900,000,000

- Monto Externo: 150,000,000

- Cantidad: 9

- Menor semana: 2

- [ ] ¿El nivel muestra "Senior A"?

- [ ] ¿La comisión total es mayor a 0?

### 3. Prueba de Llaves

- [ ] Poner Cantidad en 5 (menos de 6)

- [ ] ¿Los premios de montos muestran 0?

- [ ] Poner Cantidad en 6

- [ ] ¿Los premios de montos ahora muestran valores?

### 4. Prueba de Llave Semanal

- [ ] Poner Menor Semana en 1

- [ ] ¿El premio de cantidad muestra 0?

- [ ] Poner Menor Semana en 2

- [ ] ¿El premio de cantidad muestra valor?

## 🟡 PRUEBAS DE PERFILES

### 5. Cambio de Perfiles

- [ ] Cambiar a perfil "Empresarial 1"

- [ ] ¿Los cálculos se actualizan?

- [ ] Cambiar a perfil "Ágil 2"

- [ ] ¿Se mantienen los valores ingresados?

## 🔴 PRUEBAS DE ADMIN

### 6. Acceso a Admin

- [ ] Ir a /admin.html

- [ ] Ingresar PIN incorrecto (111111)

- [ ] ¿Muestra error?

- [ ] Ingresar PIN correcto (123456)

- [ ] ¿Entra al panel?

### 7. Modificar Valores

- [ ] Cambiar Base Fija de Ágil 1 a 3,500,000

- [ ] Guardar cambios

- [ ] Volver a la aplicación principal

- [ ] Seleccionar Ágil 1

- [ ] ¿La base muestra 3,500,000?

## 📄 PRUEBAS DE PDF

### 8. PDF Simple

- [ ] Click en PDF → PDF Simple

- [ ] ¿Se descarga un PDF?

- [ ] ¿Tiene todos los datos?

### 9. PDF Profesional

- [ ] Click en PDF → PDF Profesional

- [ ] Ingresar nombre "Juan Pérez"

- [ ] ¿Se descarga PDF con gráficos?

- [ ] ¿Aparece el nombre en el PDF?

#### **PASO 4: Tests Automáticos Básicos**

Crear archivo tests/automatic/calculos.test.js:

// TESTS AUTOMÁTICOS - No te asustes, son como fórmulas de Excel

// Importar las funciones a probar

const { calcularNivel, calcularMultiplicador } = require('../../src/modules/calculations');

// GRUPO 1: Probar cálculo de niveles

describe('Pruebas de Niveles', () => {

// Prueba 1: Nivel Capilla

test('Monto de 700 millones debe dar nivel Capilla (0)', () => {

const resultado = calcularNivel(700000000, 'interno');

expect(resultado).toBe(0); // Esperamos nivel 0 (Capilla)

});

// Prueba 2: Nivel Senior A

test('Monto de 950 millones debe dar nivel Senior A (2)', () => {

const resultado = calcularNivel(950000000, 'interno');

expect(resultado).toBe(2); // Esperamos nivel 2 (Senior A)

});

// Prueba 3: Sin alcanzar ningún nivel

test('Monto de 100 millones no alcanza ningún nivel', () => {

const resultado = calcularNivel(100000000, 'interno');

expect(resultado).toBe(-1); // Esperamos -1 (sin nivel)

});

});

// GRUPO 2: Probar multiplicadores

describe('Pruebas de Multiplicadores', () => {

// Prueba de conversión

test('Conversión 8% debe dar multiplicador 1.0', () => {

const resultado = calcularMultiplicador('conversion', 8);

expect(resultado).toBe(1.0);

});

// Prueba de empatía baja

test('Empatía 50% debe dar multiplicador 0', () => {

const resultado = calcularMultiplicador('empatia', 50);

expect(resultado).toBe(0);

});

});

// GRUPO 3: Probar cálculo completo

describe('Pruebas de Comisión Total', () => {

test('Caso completo Senior A con todos los bonos', () => {

const datos = {

nivelAnterior: 2,

montoInterno: 900000000,

montoExterno: 150000000,

montoRecuperado: 80000000,

cantidad: 9,

menorSemana: 2,

conversion: 8,

empatia: 96,

proceso: 95,

mora: 2,

nivelEquipo: 2

};

const resultado = calcularComisionTotal(datos);

// Verificar que hay comisión

expect(resultado.total).toBeGreaterThan(0);

// Verificar que el subtotal es correcto

expect(resultado.subtotal).toBe(7100000);

// Verificar multiplicador

expect(resultado.multiplicador).toBeCloseTo(1.05);

});

});

#### **PASO 5: Ejecutar los Tests**

****# Para ejecutar tests automáticos:

npm test

# Verás algo así:

# ✓ Monto de 700 millones debe dar nivel Capilla (0)

# ✓ Monto de 950 millones debe dar nivel Senior A (2)

# ✓ Sin alcanzar ningún nivel

# ... etc

# Si algún test falla, verás:

# ✗ Conversión 8% debe dar multiplicador 1.0

# Expected: 1.0

# Received: 0.8

### **16.3 Plan de Testing Completo**

#### **16.3.1 Cuándo Hacer Tests**

| **Momento** | **Tests Manuales** | **Tests Automáticos** |
| --- | --- | --- |
| Cada cambio pequeño | ❌ | ✅ |
| Antes de mostrar al jefe | ✅ | ✅ |
| Después de cambiar valores | ✅ | ✅ |
| Una vez por semana | ✅ | ✅ |

#### **16.3.2 Qué Probar Siempre**

**1. CASOS FELICES** (Todo correcto)

* Datos normales
* Flujo esperado
* Valores típicos

**2. CASOS LÍMITE** (Valores extremos)

* Monto = 0
* Monto = 999,999,999,999
* Porcentajes = 0% y 100%

**3. CASOS DE ERROR** (Cosas que pueden fallar)

* Campos vacíos
* Letras donde van números
* Clicks múltiples rápidos

### **16.4 Documento de Casos de Prueba**

Crear archivo tests/casos-de-prueba.xlsx con estas hojas:

#### **Hoja 1: Casos de Cálculo**

| **ID** | **Descripción** | **Entrada** | **Salida Esperada** | **¿Pasó?** |
| --- | --- | --- | --- | --- |
| C01 | Senior A completo | Interno: 900M, Externo: 150M | Nivel: Senior A, Total: ~7.3M | ☐ |
| C02 | Sin llave montos | Cantidad: 5 | Premios montos: 0 | ☐ |
| C03 | Sin llave semanal | Menor sem: 1 | Premio cantidad: 0 | ☐ |

#### **Hoja 2: Casos de Perfiles**

| **ID** | **Descripción** | **Pasos** | **Resultado Esperado** | **¿Pasó?** |
| --- | --- | --- | --- | --- |
| P01 | Cambiar perfil | 1. Seleccionar Empresarial 1 | Recalcula todo | ☐ |
| P02 | Guardar perfil | 1. Cambiar valores 2. Recargar | Mantiene cambios | ☐ |

#### **Hoja 3: Casos de Admin**

| **ID** | **Descripción** | **Pasos** | **Resultado Esperado** | **¿Pasó?** |
| --- | --- | --- | --- | --- |
| A01 | Login correcto | PIN: 123456 | Entra al panel | ☐ |
| A02 | Cambiar base | 1. Cambiar a 4M 2. Guardar | Se refleja en app | ☐ |

### **16.5 Reporte de Bugs**

Cuando algo no funciona, documentar así:

# BUG #001

\*\*Fecha\*\*: 25/03/2024

\*\*Encontrado por\*\*: Juan

\*\*Severidad\*\*: Alta/Media/Baja

\*\*Descripción\*\*:

Al poner monto interno en 0, la aplicación muestra error

\*\*Pasos para reproducir\*\*:

1. Abrir aplicación

2. Borrar monto interno

3. Poner 0

\*\*Resultado actual\*\*:

Error en consola, cálculo se detiene

\*\*Resultado esperado\*\*:

Debe mostrar nivel -1 y continuar

\*\*Capturas\*\*:

[Adjuntar imagen del error]

### **16.6 Automatización Simple**

Para que los tests se ejecuten solos:

// En package.json agregar:

{

"scripts": {

"test": "jest",

"test:watch": "jest --watch",

"test:manual": "open tests/manual/checklist.md"

}

}

### **16.7 Herramientas Útiles**

**Para Capturas de Pantalla**:

* Windows: Win + Shift + S
* Mac: Cmd + Shift + 4

**Para Grabar la Pantalla** (mostrar bugs):

* Windows: Win + G
* Mac: QuickTime Player

**Para Compartir Resultados**:

* Crear carpeta "resultados-tests" con fecha
* Guardar capturas y videos
* Compartir en drive/slack

### **16.8 Certificación de Calidad**

Antes de decir "está listo", verificar:

# CERTIFICACIÓN DE RELEASE v1.0

## ✅ Tests Automáticos

- [ ] Todos los tests pasan (npm test)

- [ ] Coverage > 80%

## ✅ Tests Manuales

- [ ] Checklist completo sin errores

- [ ] Probado en Chrome, Firefox, Safari

- [ ] Probado en móvil

## ✅ Documentación

- [ ] README actualizado

- [ ] Casos de prueba documentados

- [ ] Bugs conocidos listados

## ✅ Performance

- [ ] Carga en menos de 3 segundos

- [ ] Cálculos instantáneos

- [ ] Sin errores en consola

\*\*Certificado por\*\*: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\*\*Fecha\*\*: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\*\*Versión\*\*: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



## **17. DOCUMENTACIÓN DEL CÓDIGO**

### **17.1 JSDoc Completo**

Todo el código DEBE estar documentado con JSDoc:

/\*\*

\* Calcula el nivel alcanzado según el monto y tipo

\* @param {number} monto - Monto en guaraníes (sin decimales)

\* @param {('interno'|'externo'|'recuperado')} tipo - Tipo de monto a evaluar

\* @returns {number} Nivel alcanzado (0-5) o -1 si no alcanza ninguno

\* @throws {TypeError} Si el tipo no es válido

\* @example

\* // returns 2 (Senior A)

\* calcularNivel(950000000, 'interno')

\*/

function calcularNivel(monto, tipo) {

if (!['interno', 'externo', 'recuperado'].includes(tipo)) {

throw new TypeError(`Tipo inválido: ${tipo}`);

}

const metas = CONFIG.metas[tipo];

for (let i = metas.length - 1; i >= 0; i--) {

if (monto >= metas[i]) return i;

}

return -1;

}

/\*\*

\* @typedef {Object} CalculationResult

\* @property {number} base - Monto base fijo

\* @property {number} bonusCarrera - Bonus por nivel de carrera

\* @property {number} bonusInterno - Bonus por monto interno

\* @property {number} bonusExterno - Bonus por monto externo

\* @property {number} bonusRecuperado - Bonus por recuperados

\* @property {number} bonusCantidad - Bonus por cantidad

\* @property {number} bonusEquipo - Bonus por equipo

\* @property {number} subtotal - Suma de todos los bonuses

\* @property {number} multiplicadorTotal - Multiplicador final aplicado

\* @property {number} total - Comisión total a pagar

\*/

/\*\*

\* Calcula la comisión total con todos los componentes

\* @param {Object} data - Datos de entrada

\* @returns {CalculationResult} Resultado del cálculo

\*/

function calcularComisionTotal(data) {

// Implementation

}

### **17.2 Diagramas de Arquitectura**

#### **17.2.1 Flujo de Datos**

****┌─────────────┐ ┌──────────────┐ ┌────────────┐

│ Usuario │────▶│ Formulario │────▶│ Validators │

└─────────────┘ └──────────────┘ └────────────┘

│ │

▼ ▼

┌──────────────┐ ┌────────────┐

│ State │◀────│ Valid? │

└──────────────┘ └────────────┘

│

▼

┌──────────────┐

│ Calculations │

└──────────────┘

│

┌───────┴────────┐

▼ ▼

┌──────────────┐ ┌──────────────┐

│ UI │ │ Storage │

└──────────────┘ └──────────────┘

#### **17.2.2 Componentes del Sistema**

****┌─────────────────────────────────────────────┐

│ index.html │

├─────────────────────────────────────────────┤

│ main.js │

├──────────┬──────────┬──────────┬───────────┤

│ config.js│ calc.js │ ui.js │storage.js │

├──────────┴──────────┴──────────┴───────────┤

│ validators.js │

├─────────────────────────────────────────────┤

│ utilities.js │

└─────────────────────────────────────────────┘

### **17.3 Guía de Contribución**

Para mantener la calidad del código:

1. **Antes de codificar**: Leer esta documentación completa
2. **Durante el desarrollo**:
   * Seguir los estándares definidos
   * Escribir tests para nueva funcionalidad
   * Documentar con JSDoc
3. **Antes de integrar**:
   * Pasar todos los tests
   * Code review por pares
   * Actualizar documentación si es necesario

## **19. SISTEMA DE PERFILES COMERCIALES**

### **19.1 Estructura de Perfiles**

La aplicación DEBE soportar 4 perfiles comerciales con configuración independiente:

// src/modules/profiles.js

export const PROFILES = {

'agil\_1': {

id: 'agil\_1',

name: 'Ágil 1',

description: 'Comercial Ágil Nivel 1',

config: {

// Base fija configurable

base: 3000000,

// Niveles (mismo estructura, valores configurables)

niveles: ['Capilla', 'Junior', 'Senior A', 'Senior B', 'Máster', 'Genio'],

// Metas por nivel

metas: {

montoInterno: [600000000, 800000000, 900000000, 1000000000, 1100000000, 1200000000],

montoExterno: [50000000, 100000000, 150000000, 200000000, 300000000, 400000000],

montoRecuperado: [40000000, 60000000, 80000000, 100000000, 120000000, 150000000],

cantidad: [6, 8, 9, 10, 12, 13]

},

// Pagos por nivel

pagos: {

carrera: [0, 0, 500000, 1000000, 1500000, 2000000],

montoInterno: [500000, 600000, 1000000, 1400000, 2000000, 2500000],

montoExterno: [800000, 1000000, 1500000, 2000000, 2500000, 3300000],

montoRecuperado: [300000, 400000, 500000, 600000, 800000, 1000000],

cantidad: [0, 400000, 600000, 700000, 1000000, 1200000],

equipo: [0, 0, 0, 500000, 800000, 1000000]

},

// Multiplicadores configurables

multiplicadores: {

conversion: [

{min: 10, mult: 1.1, text: '10%+'},

{min: 8, mult: 1.0, text: '8%'},

{min: 7, mult: 0.8, text: '7%'},

{min: 6, mult: 0.7, text: '6%'},

{min: 5, mult: 0.6, text: '5%'},

{min: 4, mult: 0.5, text: '4%'},

{min: 0, mult: 0.3, text: '<4%'}

],

empatia: [

{min: 96, mult: 1.0, text: '96%+'},

{min: 90, mult: 0.9, text: '90%'},

{min: 80, mult: 0.5, text: '80%'},

{min: 70, mult: 0.3, text: '70%'},

{min: 0, mult: 0, text: '<70%'}

],

proceso: [

{min: 95, mult: 1.0, text: '95%+'},

{min: 90, mult: 0.95, text: '90%'},

{min: 85, mult: 0.8, text: '85%'},

{min: 70, mult: 0.3, text: '70%'},

{min: 0, mult: 0, text: '<70%'}

],

mora: [

{min: 0, mult: 1.05, text: '0-2%'},

{min: 3, mult: 0.95, text: '3-7%'},

{min: 8, mult: 0.9, text: '8-9%'},

{min: 10, mult: 0.85, text: '10-14%'},

{min: 15, mult: 0.7, text: '15%+'}

]

}

}

},

'agil\_2': {

id: 'agil\_2',

name: 'Ágil 2',

description: 'Comercial Ágil Nivel 2',

config: {

// Copia de la estructura con valores iniciales iguales

// El admin los modificará según necesidad

base: 3000000,

// ... resto igual que agil\_1

}

},

'empresarial\_1': {

id: 'empresarial\_1',

name: 'Empresarial 1',

description: 'Comercial Empresarial Nivel 1',

config: {

// Misma estructura, valores configurables

base: 3000000,

// ... resto igual

}

},

'empresarial\_2': {

id: 'empresarial\_2',

name: 'Empresarial 2',

description: 'Comercial Empresarial Nivel 2',

config: {

// Misma estructura, valores configurables

base: 3000000,

// ... resto igual

}

}

};

// Gestión del perfil activo

export const ProfileManager = {

currentProfile: null,

loadProfile(profileId) {

const profile = PROFILES[profileId];

if (!profile) throw new Error(`Perfil no encontrado: ${profileId}`);

this.currentProfile = profile;

this.applyProfileConfig(profile.config);

this.saveCurrentProfile(profileId);

return profile;

},

applyProfileConfig(config) {

// Actualizar CONFIG global con valores del perfil

window.CONFIG = { ...window.CONFIG, ...config };

// Notificar a la UI para recalcular

window.dispatchEvent(new CustomEvent('profileChanged', {

detail: { config }

}));

},

saveCurrentProfile(profileId) {

localStorage.setItem('currentProfile', profileId);

},

getCurrentProfile() {

const saved = localStorage.getItem('currentProfile');

return saved || 'agil\_1'; // Default

},

getAllProfiles() {

return Object.values(PROFILES).map(p => ({

id: p.id,

name: p.name,

description: p.description

}));

}

};

### **19.2 Integración en la UI Principal**

#### **19.2.1 Selector de Perfil**

****// Agregar en el header de la aplicación

const ProfileSelector = {

render() {

const profiles = ProfileManager.getAllProfiles();

const current = ProfileManager.getCurrentProfile();

return `

<div class="profile-selector">

<label>Perfil Comercial:</label>

<select id="profileSelect" onchange="changeProfile(this.value)">

${profiles.map(p => `

<option value="${p.id}" ${p.id === current ? 'selected' : ''}>

${p.name}

</option>

`).join('')}

</select>

<span class="profile-desc">${profiles.find(p => p.id === current)?.description}</span>

</div>

`;

}

};

// Función para cambiar perfil

function changeProfile(profileId) {

if (confirm('¿Cambiar de perfil recalculará todos los valores. Continuar?')) {

ProfileManager.loadProfile(profileId);

updateCalculations(); // Recalcular todo

UI.showNotification('Perfil cambiado exitosamente');

}

}

#### **19.2.2 Indicador Visual de Perfil**

****/\* styles/profiles.css \*/

.profile-selector {

background: linear-gradient(135deg, #83C5BE 0%, #006D77 100%);

padding: 10px 15px;

border-radius: 8px;

color: white;

display: flex;

align-items: center;

gap: 10px;

margin-bottom: 15px;

}

.profile-selector select {

padding: 5px 10px;

border-radius: 4px;

border: none;

font-weight: bold;

}

.profile-desc {

font-size: 12px;

opacity: 0.9;

font-style: italic;

}

/\* Indicadores por tipo de perfil \*/

.profile-agil\_1 { border-left: 5px solid #4CAF50; }

.profile-agil\_2 { border-left: 5px solid #8BC34A; }

.profile-empresarial\_1 { border-left: 5px solid #2196F3; }

.profile-empresarial\_2 { border-left: 5px solid #3F51B5; }

### **19.3 Persistencia de Configuración por Perfil**

****// storage.js extension

export const ProfileStorage = {

STORAGE\_KEY: 'commission\_profiles',

saveProfiles(profiles) {

const data = {

version: CONFIG.version,

timestamp: Date.now(),

profiles: profiles

};

localStorage.setItem(this.STORAGE\_KEY, JSON.stringify(data));

},

loadProfiles() {

try {

const stored = localStorage.getItem(this.STORAGE\_KEY);

if (!stored) return null;

const data = JSON.parse(stored);

// Verificar versión

if (data.version !== CONFIG.version) {

console.warn('Versión de perfiles desactualizada, usando defaults');

return null;

}

return data.profiles;

} catch (e) {

console.error('Error cargando perfiles:', e);

return null;

}

},

exportProfiles() {

const profiles = this.loadProfiles() || PROFILES;

const blob = new Blob(

[JSON.stringify(profiles, null, 2)],

{ type: 'application/json' }

);

const url = URL.createObjectURL(blob);

const a = document.createElement('a');

a.href = url;

a.download = `perfiles\_comisiones\_${new Date().toISOString().split('T')[0]}.json`;

a.click();

},

importProfiles(file) {

return new Promise((resolve, reject) => {

const reader = new FileReader();

reader.onload = (e) => {

try {

const profiles = JSON.parse(e.target.result);

this.validateProfiles(profiles);

this.saveProfiles(profiles);

resolve(profiles);

} catch (error) {

reject(error);

}

};

reader.readAsText(file);

});

},

validateProfiles(profiles) {

// Validar estructura de cada perfil

const requiredFields = ['base', 'niveles', 'metas', 'pagos', 'multiplicadores'];

Object.values(profiles).forEach(profile => {

requiredFields.forEach(field => {

if (!profile.config[field]) {

throw new Error(`Perfil ${profile.id} falta campo: ${field}`);

}

});

});

}

};



## **20. PANEL DE ADMINISTRACIÓN**

### **20.1 Estructura del Panel Admin**

El panel de administración DEBE estar en una URL separada (/admin.html) con su propia estructura:

<!-- admin.html -->

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Admin - Sistema de Comisiones</title>

<link rel="stylesheet" href="styles/admin.css">

</head>

<body>

<div id="adminApp">

<!-- Login Screen -->

<div id="loginScreen" class="login-container">

<h1>🔐 Administración del Sistema</h1>

<form id="loginForm">

<input type="password" id="adminPin" placeholder="PIN de 6 dígitos" maxlength="6">

<button type="submit">Ingresar</button>

</form>

</div>

<!-- Admin Panel (oculto inicialmente) -->

<div id="adminPanel" class="admin-panel" style="display: none;">

<!-- Se carga dinámicamente -->

</div>

</div>

<script type="module" src="admin.js"></script>

</body>

</html>

### **20.2 Lógica del Panel Admin**

****// admin.js

import { PROFILES } from './modules/profiles.js';

import { ProfileStorage } from './modules/storage.js';

class AdminPanel {

constructor() {

this.profiles = this.loadProfiles();

this.currentProfileId = 'agil\_1';

this.changes = [];

this.isAuthenticated = false;

}

// Autenticación simple

authenticate(pin) {

// PIN hardcodeado por simplicidad - en producción usar hash

const ADMIN\_PIN = '123456';

if (pin === ADMIN\_PIN) {

this.isAuthenticated = true;

this.showAdminPanel();

return true;

}

return false;

}

loadProfiles() {

// Cargar perfiles guardados o usar defaults

return ProfileStorage.loadProfiles() || PROFILES;

}

showAdminPanel() {

document.getElementById('loginScreen').style.display = 'none';

document.getElementById('adminPanel').style.display = 'block';

this.renderAdminPanel();

}

renderAdminPanel() {

const panel = document.getElementById('adminPanel');

panel.innerHTML = `

<header class="admin-header">

<h1>🛠️ Panel de Administración - Comisiones</h1>

<div class="admin-actions">

<button onclick="admin.saveChanges()" class="btn-save">💾 Guardar Cambios</button>

<button onclick="admin.exportConfig()" class="btn-export">📥 Exportar</button>

<button onclick="admin.importConfig()" class="btn-import">📤 Importar</button>

<button onclick="admin.logout()" class="btn-logout">🚪 Salir</button>

</div>

</header>

<div class="profile-tabs">

${Object.values(this.profiles).map(profile => `

<button

class="profile-tab ${profile.id === this.currentProfileId ? 'active' : ''}"

onclick="admin.selectProfile('${profile.id}')"

>

${profile.name}

</button>

`).join('')}

</div>

<div class="config-sections">

${this.renderConfigSections()}

</div>

<div class="preview-section">

<h3>Vista Previa de Cambios</h3>

<div id="changePreview"></div>

</div>

<div class="history-section">

<h3>Historial de Cambios</h3>

<div id="changeHistory"></div>

</div>

`;

}

renderConfigSections() {

const profile = this.profiles[this.currentProfileId];

const config = profile.config;

return `

<!-- Base Fija -->

<section class="config-section">

<h2>💰 Salario Base</h2>

<div class="config-field">

<label>Base Fija (Gs):</label>

<input

type="number"

value="${config.base}"

onchange="admin.updateValue('base', this.value)"

class="money-input"

>

</div>

</section>

<!-- Metas -->

<section class="config-section">

<h2>🎯 Metas por Nivel</h2>

${this.renderMetasTable(config.metas, config.niveles)}

</section>

<!-- Pagos -->

<section class="config-section">

<h2>💵 Pagos/Bonos por Nivel</h2>

${this.renderPagosTable(config.pagos, config.niveles)}

</section>

<!-- Multiplicadores -->

<section class="config-section">

<h2>📊 Multiplicadores</h2>

${this.renderMultiplicadores(config.multiplicadores)}

</section>

`;

}

renderMetasTable(metas, niveles) {

return `

<table class="config-table">

<thead>

<tr>

<th>Nivel</th>

<th>Monto Interno</th>

<th>Monto Externo</th>

<th>Recuperados</th>

<th>Cantidad</th>

</tr>

</thead>

<tbody>

${niveles.map((nivel, i) => `

<tr>

<td><strong>${nivel}</strong></td>

<td>

<input

type="number"

value="${metas.montoInterno[i]}"

onchange="admin.updateMeta('montoInterno', ${i}, this.value)"

class="table-input money"

>

</td>

<td>

<input

type="number"

value="${metas.montoExterno[i]}"

onchange="admin.updateMeta('montoExterno', ${i}, this.value)"

class="table-input money"

>

</td>

<td>

<input

type="number"

value="${metas.montoRecuperado[i]}"

onchange="admin.updateMeta('montoRecuperado', ${i}, this.value)"

class="table-input money"

>

</td>

<td>

<input

type="number"

value="${metas.cantidad[i]}"

onchange="admin.updateMeta('cantidad', ${i}, this.value)"

class="table-input"

>

</td>

</tr>

`).join('')}

</tbody>

</table>

`;

}

renderPagosTable(pagos, niveles) {

return `

<table class="config-table">

<thead>

<tr>

<th>Nivel</th>

<th>Carrera</th>

<th>Interno</th>

<th>Externo</th>

<th>Recuperado</th>

<th>Cantidad</th>

<th>Equipo</th>

</tr>

</thead>

<tbody>

${niveles.map((nivel, i) => `

<tr>

<td><strong>${nivel}</strong></td>

<td>

<input

type="number"

value="${pagos.carrera[i]}"

onchange="admin.updatePago('carrera', ${i}, this.value)"

class="table-input money"

>

</td>

<td>

<input

type="number"

value="${pagos.montoInterno[i]}"

onchange="admin.updatePago('montoInterno', ${i}, this.value)"

class="table-input money"

>

</td>

<td>

<input

type="number"

value="${pagos.montoExterno[i]}"

onchange="admin.updatePago('montoExterno', ${i}, this.value)"

class="table-input money"

>

</td>

<td>

<input

type="number"

value="${pagos.montoRecuperado[i]}"

onchange="admin.updatePago('montoRecuperado', ${i}, this.value)"

class="table-input money"

>

</td>

<td>

<input

type="number"

value="${pagos.cantidad[i]}"

onchange="admin.updatePago('cantidad', ${i}, this.value)"

class="table-input money"

>

</td>

<td>

<input

type="number"

value="${pagos.equipo[i]}"

onchange="admin.updatePago('equipo', ${i}, this.value)"

class="table-input money"

>

</td>

</tr>

`).join('')}

</tbody>

</table>

`;

}

renderMultiplicadores(multiplicadores) {

return `

<div class="multiplicadores-grid">

${Object.entries(multiplicadores).map(([tipo, valores]) => `

<div class="multiplicador-config">

<h4>${this.getTipoLabel(tipo)}</h4>

<table class="mult-table">

<thead>

<tr>

<th>Rango</th>

<th>Multiplicador</th>

</tr>

</thead>

<tbody>

${valores.map((item, i) => `

<tr>

<td>${item.text}</td>

<td>

<input

type="number"

value="${item.mult}"

step="0.05"

min="0"

max="2"

onchange="admin.updateMultiplicador('${tipo}', ${i}, this.value)"

class="mult-input"

> = ${Math.round(item.mult \* 100)}%

</td>

</tr>

`).join('')}

</tbody>

</table>

</div>

`).join('')}

</div>

`;

}

getTipoLabel(tipo) {

const labels = {

conversion: '% Conversión',

empatia: '% Empatía/Mystery',

proceso: '% Proceso/CRM',

mora: '% Mora'

};

return labels[tipo] || tipo;

}

// Métodos de actualización

updateValue(field, value) {

const numValue = parseInt(value, 10);

this.profiles[this.currentProfileId].config[field] = numValue;

this.trackChange(`Base fija`, `Cambio a ${this.formatMoney(numValue)}`);

this.showPreview();

}

updateMeta(tipo, index, value) {

const numValue = parseInt(value, 10);

this.profiles[this.currentProfileId].config.metas[tipo][index] = numValue;

const nivel = this.profiles[this.currentProfileId].config.niveles[index];

this.trackChange(`Meta ${tipo}`, `${nivel}: ${this.formatMoney(numValue)}`);

this.showPreview();

}

updatePago(tipo, index, value) {

const numValue = parseInt(value, 10);

this.profiles[this.currentProfileId].config.pagos[tipo][index] = numValue;

const nivel = this.profiles[this.currentProfileId].config.niveles[index];

this.trackChange(`Pago ${tipo}`, `${nivel}: ${this.formatMoney(numValue)}`);

this.showPreview();

}

updateMultiplicador(tipo, index, value) {

const numValue = parseFloat(value);

this.profiles[this.currentProfileId].config.multiplicadores[tipo][index].mult = numValue;

const porcentaje = Math.round(numValue \* 100);

this.trackChange(`Multiplicador ${tipo}`, `Rango ${index + 1}: ${porcentaje}%`);

this.showPreview();

}

// Tracking de cambios

trackChange(campo, detalle) {

this.changes.push({

timestamp: new Date(),

profile: this.currentProfileId,

campo,

detalle

});

}

showPreview() {

const preview = document.getElementById('changePreview');

if (this.changes.length === 0) {

preview.innerHTML = '<p class="no-changes">Sin cambios pendientes</p>';

return;

}

preview.innerHTML = `

<div class="changes-list">

${this.changes.map(change => `

<div class="change-item">

<span class="change-field">${change.campo}:</span>

<span class="change-detail">${change.detalle}</span>

<span class="change-profile">(${this.profiles[change.profile].name})</span>

</div>

`).join('')}

</div>

`;

}

// Guardar cambios

saveChanges() {

if (this.changes.length === 0) {

alert('No hay cambios para guardar');

return;

}

if (confirm(`¿Guardar ${this.changes.length} cambios?`)) {

// Guardar en localStorage

ProfileStorage.saveProfiles(this.profiles);

// Guardar historial

this.saveHistory();

// Limpiar cambios

this.changes = [];

this.showPreview();

alert('✅ Cambios guardados exitosamente');

}

}

saveHistory() {

const history = JSON.parse(localStorage.getItem('admin\_history') || '[]');

history.push({

date: new Date().toISOString(),

user: 'Admin',

changes: this.changes.length,

details: this.changes

});

// Mantener solo últimos 50 registros

if (history.length > 50) {

history.shift();

}

localStorage.setItem('admin\_history', JSON.stringify(history));

this.showHistory();

}

showHistory() {

const history = JSON.parse(localStorage.getItem('admin\_history') || '[]');

const historyDiv = document.getElementById('changeHistory');

historyDiv.innerHTML = `

<table class="history-table">

<thead>

<tr>

<th>Fecha</th>

<th>Usuario</th>

<th>Cambios</th>

<th>Acciones</th>

</tr>

</thead>

<tbody>

${history.reverse().slice(0, 10).map((entry, i) => `

<tr>

<td>${new Date(entry.date).toLocaleString('es-PY')}</td>

<td>${entry.user}</td>

<td>${entry.changes}</td>

<td>

<button onclick="admin.showHistoryDetail(${i})" class="btn-detail">

Ver detalle

</button>

</td>

</tr>

`).join('')}

</tbody>

</table>

`;

}

// Utilidades

formatMoney(value) {

return value.toString().replace(/\B(?=(\d{3})+(?!\d))/g, ".") + ' Gs';

}

selectProfile(profileId) {

this.currentProfileId = profileId;

this.renderAdminPanel();

}

exportConfig() {

ProfileStorage.exportProfiles();

}

async importConfig() {

const input = document.createElement('input');

input.type = 'file';

input.accept = '.json';

input.onchange = async (e) => {

try {

const file = e.target.files[0];

const profiles = await ProfileStorage.importProfiles(file);

this.profiles = profiles;

this.renderAdminPanel();

alert('✅ Configuración importada exitosamente');

} catch (error) {

alert('❌ Error al importar: ' + error.message);

}

};

input.click();

}

logout() {

if (confirm('¿Salir del panel de administración?')) {

this.isAuthenticated = false;

window.location.reload();

}

}

}

// Inicializar

const admin = new AdminPanel();

// Login handler

document.getElementById('loginForm').addEventListener('submit', (e) => {

e.preventDefault();

const pin = document.getElementById('adminPin').value;

if (!admin.authenticate(pin)) {

alert('❌ PIN incorrecto');

document.getElementById('adminPin').value = '';

}

});

// Exportar para uso global

window.admin = admin;

### **20.3 Estilos del Panel Admin**

****/\* styles/admin.css \*/

:root {

--admin-primary: #006D77;

--admin-secondary: #83C5BE;

--admin-danger: #F44336;

--admin-success: #4CAF50;

--admin-warning: #FFA726;

}

/\* Login Screen \*/

.login-container {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

height: 100vh;

background: linear-gradient(135deg, var(--admin-secondary) 0%, var(--admin-primary) 100%);

}

.login-container h1 {

color: white;

margin-bottom: 30px;

}

#loginForm {

background: white;

padding: 40px;

border-radius: 12px;

box-shadow: 0 8px 32px rgba(0,0,0,0.1);

}

#adminPin {

font-size: 24px;

text-align: center;

padding: 15px;

width: 200px;

border: 2px solid #ddd;

border-radius: 8px;

margin-bottom: 20px;

}

/\* Admin Panel \*/

.admin-panel {

min-height: 100vh;

background: #f5f6f8;

}

.admin-header {

background: var(--admin-primary);

color: white;

padding: 20px;

display: flex;

justify-content: space-between;

align-items: center;

}

.admin-actions {

display: flex;

gap: 10px;

}

.admin-actions button {

padding: 10px 20px;

border: none;

border-radius: 6px;

cursor: pointer;

font-weight: bold;

transition: all 0.3s;

}

.btn-save {

background: var(--admin-success);

color: white;

}

.btn-save:hover {

background: #45a049;

}

.btn-export, .btn-import {

background: var(--admin-secondary);

color: white;

}

.btn-logout {

background: var(--admin-danger);

color: white;

}

/\* Profile Tabs \*/

.profile-tabs {

background: white;

padding: 0;

display: flex;

border-bottom: 2px solid #ddd;

}

.profile-tab {

flex: 1;

padding: 15px;

border: none;

background: transparent;

cursor: pointer;

font-size: 16px;

font-weight: bold;

transition: all 0.3s;

}

.profile-tab:hover {

background: #f0f0f0;

}

.profile-tab.active {

background: var(--admin-primary);

color: white;

}

/\* Config Sections \*/

.config-sections {

padding: 20px;

}

.config-section {

background: white;

padding: 20px;

margin-bottom: 20px;

border-radius: 8px;

box-shadow: 0 2px 8px rgba(0,0,0,0.05);

}

.config-section h2 {

color: var(--admin-primary);

margin-bottom: 15px;

padding-bottom: 10px;

border-bottom: 2px solid #eee;

}

/\* Tables \*/

.config-table {

width: 100%;

border-collapse: collapse;

}

.config-table th {

background: var(--admin-secondary);

color: white;

padding: 12px;

text-align: left;

}

.config-table td {

padding: 8px;

border-bottom: 1px solid #eee;

}

.table-input {

width: 100%;

padding: 8px;

border: 1px solid #ddd;

border-radius: 4px;

font-size: 14px;

}

.table-input:focus {

border-color: var(--admin-primary);

outline: none;

}

.table-input.money {

text-align: right;

font-family: monospace;

}

/\* Multiplicadores Grid \*/

.multiplicadores-grid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));

gap: 20px;

}

.multiplicador-config {

background: #f8f9fa;

padding: 15px;

border-radius: 8px;

}

.multiplicador-config h4 {

color: var(--admin-primary);

margin-bottom: 10px;

}

.mult-table {

width: 100%;

font-size: 14px;

}

.mult-input {

width: 60px;

padding: 5px;

border: 1px solid #ddd;

border-radius: 4px;

text-align: center;

}

/\* Preview Section \*/

.preview-section {

background: white;

padding: 20px;

margin: 20px;

border-radius: 8px;

border: 2px solid var(--admin-warning);

}

.changes-list {

max-height: 200px;

overflow-y: auto;

}

.change-item {

padding: 8px;

margin: 5px 0;

background: #fff8e1;

border-radius: 4px;

display: flex;

justify-content: space-between;

}

.change-field {

font-weight: bold;

color: var(--admin-primary);

}

/\* History Section \*/

.history-section {

background: white;

padding: 20px;

margin: 20px;

border-radius: 8px;

}

.history-table {

width: 100%;

font-size: 14px;

}

.btn-detail {

padding: 5px 10px;

background: var(--admin-secondary);

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

font-size: 12px;

}

/\* Responsive \*/

@media (max-width: 768px) {

.profile-tabs {

flex-direction: column;

}

.multiplicadores-grid {

grid-template-columns: 1fr;

}

.admin-header {

flex-direction: column;

gap: 15px;

}

.admin-actions {

flex-wrap: wrap;

justify-content: center;

}

}



## **21. GENERACIÓN DE PDF PROFESIONAL**

### **21.1 Integración de jsPDF**

****// src/modules/pdf-generator.js

import jsPDF from 'jspdf';

import 'jspdf-autotable';

import Chart from 'chart.js/auto';

export class PDFGenerator {

constructor() {

this.doc = null;

this.pageHeight = 297; // A4

this.pageWidth = 210;

this.margin = 20;

this.currentY = this.margin;

// Colores corporativos

this.colors = {

primary: [0, 109, 119], // #006D77

secondary: [131, 197, 190], // #83C5BE

success: [76, 175, 80], // #4CAF50

danger: [244, 67, 54], // #F44336

warning: [255, 167, 38], // #FFA726

dark: [33, 33, 33], // #212121

gray: [117, 117, 117] // #757575

};

}

async generatePDF(data) {

// Inicializar documento

this.doc = new jsPDF({

orientation: 'portrait',

unit: 'mm',

format: 'a4'

});

// Metadata

this.doc.setProperties({

title: `Comisión ${data.comercial.nombre} - ${data.periodo}`,

subject: 'Liquidación de Comisiones',

author: 'SERSA SAECA',

keywords: 'comisiones, comercial, liquidación',

creator: 'Sistema de Comisiones v1.0'

});

// Generar contenido

await this.addHeader(data);

this.addExecutiveSummary(data);

await this.addVisualComposition(data);

this.addVolumeDetails(data);

this.addCalculationDetails(data);

this.addMultiplierDetails(data);

this.addFooter(data);

// Retornar blob para descarga

return this.doc.output('blob');

}

async addHeader(data) {

// Logo (placeholder - en producción usar imagen real)

this.doc.setFillColor(...this.colors.primary);

this.doc.rect(this.margin, this.margin, 170, 40, 'F');

// Título

this.doc.setTextColor(255, 255, 255);

this.doc.setFontSize(24);

this.doc.setFont('helvetica', 'bold');

this.doc.text('SERSA SAECA', this.margin + 10, this.margin + 15);

this.doc.setFontSize(16);

this.doc.setFont('helvetica', 'normal');

this.doc.text('LIQUIDACIÓN DE COMISIONES', this.margin + 10, this.margin + 25);

this.doc.setFontSize(14);

this.doc.text(data.periodo, this.margin + 10, this.margin + 35);

this.currentY = this.margin + 50;

// Información del comercial

this.doc.setTextColor(...this.colors.dark);

this.doc.setFontSize(12);

this.doc.setFont('helvetica', 'bold');

this.doc.text('DATOS DEL COMERCIAL', this.margin, this.currentY);

this.currentY += 7;

this.doc.setFont('helvetica', 'normal');

this.doc.text(`Nombre: ${data.comercial.nombre}`, this.margin, this.currentY);

this.currentY += 5;

this.doc.text(`Perfil: ${data.comercial.perfil}`, this.margin, this.currentY);

this.currentY += 5;

this.doc.text(`Fecha de generación: ${new Date().toLocaleString('es-PY')}`, this.margin, this.currentY);

// Línea divisoria

this.currentY += 7;

this.doc.setLineWidth(0.5);

this.doc.setDrawColor(...this.colors.secondary);

this.doc.line(this.margin, this.currentY, this.pageWidth - this.margin, this.currentY);

this.currentY += 10;

}

addExecutiveSummary(data) {

// Título sección

this.doc.setFontSize(14);

this.doc.setFont('helvetica', 'bold');

this.doc.setTextColor(...this.colors.primary);

this.doc.text('RESUMEN EJECUTIVO', this.margin, this.currentY);

this.currentY += 10;

// KPIs en cajas

const kpis = [

{ label: 'NIVEL ALCANZADO', value: data.nivel, color: this.colors.primary },

{ label: 'SUBTOTAL', value: this.formatMoney(data.subtotal), color: this.colors.secondary },

{ label: 'MULTIPLICADOR', value: (data.multiplicador \* 100).toFixed(1) + '%', color: this.colors.warning },

{ label: 'COMISIÓN TOTAL', value: this.formatMoney(data.total), color: this.colors.success }

];

const boxWidth = 40;

const boxHeight = 25;

const spacing = 5;

let boxX = this.margin;

kpis.forEach(kpi => {

// Caja

this.doc.setFillColor(...kpi.color);

this.doc.roundedRect(boxX, this.currentY, boxWidth, boxHeight, 3, 3, 'F');

// Texto

this.doc.setTextColor(255, 255, 255);

this.doc.setFontSize(8);

this.doc.setFont('helvetica', 'normal');

this.doc.text(kpi.label, boxX + boxWidth/2, this.currentY + 8, { align: 'center' });

this.doc.setFontSize(12);

this.doc.setFont('helvetica', 'bold');

this.doc.text(kpi.value, boxX + boxWidth/2, this.currentY + 17, { align: 'center' });

boxX += boxWidth + spacing;

});

this.currentY += boxHeight + 15;

}

async addVisualComposition(data) {

// Título sección

this.doc.setFontSize(14);

this.doc.setFont('helvetica', 'bold');

this.doc.setTextColor(...this.colors.primary);

this.doc.text('COMPOSICIÓN DE LA COMISIÓN', this.margin, this.currentY);

this.currentY += 10;

// Crear gráfico con Chart.js

const canvas = document.createElement('canvas');

canvas.width = 600;

canvas.height = 400;

const ctx = canvas.getContext('2d');

const chartData = {

labels: ['Base', 'Carrera', 'Interno', 'Externo', 'Recuperado', 'Cantidad', 'Equipo'],

datasets: [{

data: [

data.desglose.base,

data.desglose.carrera,

data.desglose.interno,

data.desglose.externo,

data.desglose.recuperado,

data.desglose.cantidad,

data.desglose.equipo

],

backgroundColor: [

'#006D77',

'#83C5BE',

'#4CAF50',

'#2196F3',

'#FF9800',

'#9C27B0',

'#795548'

]

}]

};

const chart = new Chart(ctx, {

type: 'doughnut',

data: chartData,

options: {

responsive: false,

plugins: {

legend: {

position: 'right'

},

tooltip: {

callbacks: {

label: (context) => {

const value = context.parsed;

const total = context.dataset.data.reduce((a, b) => a + b, 0);

const percentage = ((value / total) \* 100).toFixed(1);

return `${context.label}: ${this.formatMoney(value)} (${percentage}%)`;

}

}

}

}

}

});

// Esperar a que se renderice

await new Promise(resolve => setTimeout(resolve, 100));

// Agregar imagen al PDF

const imgData = canvas.toDataURL('image/png');

this.doc.addImage(imgData, 'PNG', this.margin, this.currentY, 170, 60);

this.currentY += 70;

// Limpiar

chart.destroy();

}

addVolumeDetails(data) {

// Nueva página si es necesario

if (this.currentY > 200) {

this.doc.addPage();

this.currentY = this.margin;

}

// Título sección

this.doc.setFontSize(14);

this.doc.setFont('helvetica', 'bold');

this.doc.setTextColor(...this.colors.primary);

this.doc.text('DETALLE DE VOLÚMENES', this.margin, this.currentY);

this.currentY += 10;

// Tabla de volúmenes

const volumeHeaders = ['Concepto', 'Valor', 'Meta', 'Nivel', 'Cumplimiento'];

const volumeData = [

[

'Monto Interno',

this.formatMoney(data.volumenes.montoInterno),

this.formatMoney(data.metas.montoInterno),

data.niveles.interno || 'No alcanzado',

this.getProgressBar(data.volumenes.montoInterno, data.metas.montoInterno)

],

[

'Monto Externo',

this.formatMoney(data.volumenes.montoExterno),

this.formatMoney(data.metas.montoExterno),

data.niveles.externo || 'No alcanzado',

this.getProgressBar(data.volumenes.montoExterno, data.metas.montoExterno)

],

[

'Recuperados +3M',

this.formatMoney(data.volumenes.montoRecuperado),

this.formatMoney(data.metas.montoRecuperado),

data.niveles.recuperado || 'No alcanzado',

this.getProgressBar(data.volumenes.montoRecuperado, data.metas.montoRecuperado)

],

[

'Cantidad Desembolsos',

data.volumenes.cantidad.toString(),

data.metas.cantidad.toString(),

data.niveles.cantidad || 'No alcanzado',

this.getProgressBar(data.volumenes.cantidad, data.metas.cantidad)

]

];

// Agregar llaves

if (data.llaves.montos) {

volumeData.push(['', '', '', '', '✅ Llave 6 desembolsos cumplida']);

} else {

volumeData.push(['', '', '', '', '❌ Llave 6 desembolsos NO cumplida']);

}

if (data.llaves.semanal) {

volumeData.push(['', '', '', '', '✅ Llave 2/semana cumplida']);

} else {

volumeData.push(['', '', '', '', '❌ Llave 2/semana NO cumplida']);

}

this.doc.autoTable({

head: [volumeHeaders],

body: volumeData,

startY: this.currentY,

margin: { left: this.margin },

theme: 'grid',

styles: {

fontSize: 10,

cellPadding: 3

},

headStyles: {

fillColor: this.colors.primary,

textColor: [255, 255, 255]

},

columnStyles: {

4: { cellWidth: 60 }

}

});

this.currentY = this.doc.lastAutoTable.finalY + 10;

}

addCalculationDetails(data) {

// Nueva página si es necesario

if (this.currentY > 200) {

this.doc.addPage();

this.currentY = this.margin;

}

// Título sección

this.doc.setFontSize(14);

this.doc.setFont('helvetica', 'bold');

this.doc.setTextColor(...this.colors.primary);

this.doc.text('CÁLCULO DETALLADO', this.margin, this.currentY);

this.currentY += 10;

// Tabla de cálculo

const calcHeaders = ['Concepto', 'Monto (Gs)'];

const calcData = [

['Base Fija', this.formatMoney(data.desglose.base)],

['Premio Carrera', this.formatMoney(data.desglose.carrera)],

['Premio Monto Interno', this.formatMoney(data.desglose.interno)],

['Premio Monto Externo', this.formatMoney(data.desglose.externo)],

['Premio Recuperados', this.formatMoney(data.desglose.recuperado)],

['Premio Cantidad', this.formatMoney(data.desglose.cantidad)],

['Premio Equipo', this.formatMoney(data.desglose.equipo)],

['', ''],

['SUBTOTAL', this.formatMoney(data.subtotal)],

['Multiplicador aplicado', (data.multiplicador \* 100).toFixed(1) + '%'],

['', ''],

['COMISIÓN TOTAL', this.formatMoney(data.total)]

];

this.doc.autoTable({

head: [calcHeaders],

body: calcData,

startY: this.currentY,

margin: { left: this.margin },

theme: 'striped',

styles: {

fontSize: 11,

cellPadding: 4

},

headStyles: {

fillColor: this.colors.primary,

textColor: [255, 255, 255]

},

bodyStyles: {

textColor: this.colors.dark

},

// Resaltar subtotal y total

didParseCell: (data) => {

if (data.row.index === 8 || data.row.index === 11) {

data.cell.styles.fontStyle = 'bold';

data.cell.styles.fontSize = 12;

if (data.row.index === 11) {

data.cell.styles.fillColor = this.colors.success;

data.cell.styles.textColor = [255, 255, 255];

}

}

}

});

this.currentY = this.doc.lastAutoTable.finalY + 10;

}

addMultiplierDetails(data) {

// Nueva página si es necesario

if (this.currentY > 200) {

this.doc.addPage();

this.currentY = this.margin;

}

// Título sección

this.doc.setFontSize(14);

this.doc.setFont('helvetica', 'bold');

this.doc.setTextColor(...this.colors.primary);

this.doc.text('DETALLE DE MULTIPLICADORES', this.margin, this.currentY);

this.currentY += 10;

// Tabla de multiplicadores

const multHeaders = ['Indicador', 'Valor', 'Multiplicador', 'Estado'];

const multData = [

[

'Conversión',

data.indicadores.conversion + '%',

(data.multiplicadores.conversion \* 100).toFixed(0) + '%',

this.getMultStatus(data.multiplicadores.conversion)

],

[

'Empatía/Mystery',

data.indicadores.empatia + '%',

(data.multiplicadores.empatia \* 100).toFixed(0) + '%',

this.getMultStatus(data.multiplicadores.empatia)

],

[

'Proceso/CRM',

data.indicadores.proceso + '%',

(data.multiplicadores.proceso \* 100).toFixed(0) + '%',

this.getMultStatus(data.multiplicadores.proceso)

],

[

'Mora',

data.indicadores.mora + '%',

(data.multiplicadores.mora \* 100).toFixed(0) + '%',

this.getMultStatus(data.multiplicadores.mora, true)

]

];

this.doc.autoTable({

head: [multHeaders],

body: multData,

startY: this.currentY,

margin: { left: this.margin },

theme: 'grid',

styles: {

fontSize: 11,

cellPadding: 4

},

headStyles: {

fillColor: this.colors.primary,

textColor: [255, 255, 255]

}

});

this.currentY = this.doc.lastAutoTable.finalY + 10;

// Cálculo del multiplicador total

this.doc.setFontSize(12);

this.doc.setFont('helvetica', 'bold');

this.doc.text('Cálculo multiplicador total:', this.margin, this.currentY);

this.currentY += 5;

this.doc.setFont('helvetica', 'normal');

const formula = `${(data.multiplicadores.conversion \* 100).toFixed(0)}% × ${(data.multiplicadores.empatia \* 100).toFixed(0)}% × ${(data.multiplicadores.proceso \* 100).toFixed(0)}% × ${(data.multiplicadores.mora \* 100).toFixed(0)}% = ${(data.multiplicador \* 100).toFixed(1)}%`;

this.doc.text(formula, this.margin, this.currentY);

this.currentY += 15;

}

addFooter(data) {

// Ir al final de la página

this.currentY = this.pageHeight - 40;

// Línea divisoria

this.doc.setLineWidth(0.5);

this.doc.setDrawColor(...this.colors.secondary);

this.doc.line(this.margin, this.currentY, this.pageWidth - this.margin, this.currentY);

this.currentY += 5;

// Código de verificación

const verificationCode = this.generateVerificationCode(data);

this.doc.setFontSize(10);

this.doc.setFont('helvetica', 'normal');

this.doc.setTextColor(...this.colors.gray);

this.doc.text(`Código de verificación: ${verificationCode}`, this.margin, this.currentY);

this.currentY += 5;

this.doc.text('Este documento es un comprobante oficial de la liquidación de comisiones.', this.margin, this.currentY);

this.currentY += 5;

this.doc.text('Sistema de Comisiones SERSA SAECA v1.0', this.margin, this.currentY);

// Número de página

const pageCount = this.doc.internal.getNumberOfPages();

for (let i = 1; i <= pageCount; i++) {

this.doc.setPage(i);

this.doc.setFontSize(10);

this.doc.setTextColor(...this.colors.gray);

this.doc.text(

`Página ${i} de ${pageCount}`,

this.pageWidth - this.margin - 20,

this.pageHeight - 10

);

}

}

// Utilidades

formatMoney(value) {

return value.toString().replace(/\B(?=(\d{3})+(?!\d))/g, ".");

}

getProgressBar(current, target) {

const percentage = Math.min((current / target) \* 100, 100);

const filled = Math.round(percentage / 10);

const empty = 10 - filled;

return '█'.repeat(filled) + '░'.repeat(empty) + ` ${percentage.toFixed(0)}%`;

}

getMultStatus(mult, inverse = false) {

if (inverse) {

if (mult >= 1) return '✅ Óptimo';

if (mult >= 0.9) return '⚠️ Regular';

return '❌ Crítico';

}

if (mult >= 0.9) return '✅ Óptimo';

if (mult >= 0.7) return '⚠️ Regular';

return '❌ Crítico';

}

generateVerificationCode(data) {

// Generar hash simple basado en los datos

const str = JSON.stringify({

comercial: data.comercial.nombre,

periodo: data.periodo,

total: data.total,

fecha: new Date().toISOString()

});

let hash = 0;

for (let i = 0; i < str.length; i++) {

const char = str.charCodeAt(i);

hash = ((hash << 5) - hash) + char;

hash = hash & hash;

}

return Math.abs(hash).toString(16).toUpperCase().substring(0, 8);

}

}

// Integración con la aplicación principal

export function setupPDFGeneration() {

window.generateProfessionalPDF = async () => {

try {

// Mostrar loading

UI.showLoading('Generando PDF profesional...');

// Recopilar datos

const data = {

comercial: {

nombre: prompt('Nombre del comercial:') || 'Juan Pérez',

perfil: ProfileManager.currentProfile.name

},

periodo: `${new Date().toLocaleDateString('es-PY', { month: 'long', year: 'numeric' }).toUpperCase()}`,

nivel: document.getElementById('statNivel').textContent,

subtotal: parseInt(document.getElementById('calcSubtotal').textContent.replace(/\D/g, ''), 10),

multiplicador: parseFloat(document.getElementById('calcMultiplicador').textContent) / 100,

total: parseInt(document.getElementById('totalComision').textContent.replace(/\D/g, ''), 10),

desglose: {

base: parseInt(document.getElementById('calcBase').textContent.replace(/\D/g, ''), 10),

carrera: parseInt(document.getElementById('calcCarrera').textContent.replace(/\D/g, ''), 10),

interno: parseInt(document.getElementById('calcInterno').textContent.replace(/\D/g, ''), 10),

externo: parseInt(document.getElementById('calcExterno').textContent.replace(/\D/g, ''), 10),

recuperado: parseInt(document.getElementById('calcRecuperado').textContent.replace(/\D/g, ''), 10),

cantidad: parseInt(document.getElementById('calcCantidad').textContent.replace(/\D/g, ''), 10),

equipo: parseInt(document.getElementById('calcEquipo').textContent.replace(/\D/g, ''), 10)

},

volumenes: {

montoInterno: getNumericValue('montoInterno'),

montoExterno: getNumericValue('montoExterno'),

montoRecuperado: getNumericValue('montoRecuperado'),

cantidad: getNumericValue('cantidadDesembolsos')

},

metas: {

// Obtener de la configuración actual

montoInterno: CONFIG.metas.montoInterno[2], // Ejemplo: Senior A

montoExterno: CONFIG.metas.montoExterno[2],

montoRecuperado: CONFIG.metas.montoRecuperado[2],

cantidad: CONFIG.metas.cantidad[2]

},

niveles: {

interno: 'Senior A',

externo: 'Senior A',

recuperado: 'Senior A',

cantidad: 'Senior A'

},

llaves: {

montos: getNumericValue('cantidadDesembolsos') >= 6,

semanal: getNumericValue('menorSemana') >= 2

},

indicadores: {

conversion: getNumericValue('conversion'),

empatia: getNumericValue('empatia'),

proceso: getNumericValue('proceso'),

mora: getNumericValue('mora')

},

multiplicadores: {

conversion: calcularMultiplicador('conversion', getNumericValue('conversion')),

empatia: calcularMultiplicador('empatia', getNumericValue('empatia')),

proceso: calcularMultiplicador('proceso', getNumericValue('proceso')),

mora: calcularMultiplicador('mora', getNumericValue('mora'))

}

};

// Generar PDF

const generator = new PDFGenerator();

const pdfBlob = await generator.generatePDF(data);

// Descargar

const url = URL.createObjectURL(pdfBlob);

const a = document.createElement('a');

a.href = url;

a.download = `Comision\_${data.comercial.nombre.replace(/\s/g, '\_')}\_${new Date().toISOString().split('T')[0]}.pdf`;

a.click();

// Limpiar

URL.revokeObjectURL(url);

UI.hideLoading();

UI.showNotification('PDF generado exitosamente');

} catch (error) {

console.error('Error generando PDF:', error);

UI.hideLoading();

UI.showError('Error al generar el PDF');

}

};

}

### **21.2 Actualización del Botón PDF**

****// Modificar el botón PDF existente en la UI principal

function actualizarBotonPDF() {

const headerButtons = document.querySelector('.header-buttons');

// Reemplazar botón simple por menú

const pdfButton = `

<div class="pdf-menu">

<button class="header-btn" onclick="togglePDFMenu()">

📄 PDF ▼

</button>

<div class="pdf-menu-options" id="pdfMenuOptions" style="display: none;">

<button onclick="descargarPDF()">📄 PDF Simple</button>

<button onclick="generateProfessionalPDF()">📊 PDF Profesional</button>

</div>

</div>

`;

// Reemplazar botón existente

const oldButton = headerButtons.querySelector('button[onclick="descargarPDF()"]');

oldButton.outerHTML = pdfButton;

}

function togglePDFMenu() {

const menu = document.getElementById('pdfMenuOptions');

menu.style.display = menu.style.display === 'none' ? 'block' : 'none';

}

// CSS para el menú

const pdfMenuStyles = `

.pdf-menu {

position: relative;

display: inline-block;

}

.pdf-menu-options {

position: absolute;

top: 100%;

right: 0;

background: white;

border: 1px solid #ddd;

border-radius: 4px;

box-shadow: 0 2px 10px rgba(0,0,0,0.1);

z-index: 1000;

min-width: 180px;

}

.pdf-menu-options button {

display: block;

width: 100%;

padding: 10px 15px;

border: none;

background: none;

text-align: left;

cursor: pointer;

color: #333;

font-size: 12px;

}

.pdf-menu-options button:hover {

background: #f0f0f0;

}

`;



## **22. CONCLUSIÓN**

Esta documentación proporciona una especificación completa y detallada del Sistema de Comisiones Comerciales de SERSA SAECA, ahora mejorado con tres características fundamentales:

### **Nuevas Características Implementadas:**

1. **Sistema de Perfiles Comerciales**:  
   * 4 perfiles configurables: Ágil 1, Ágil 2, Empresarial 1, Empresarial 2
   * Configuración independiente para cada perfil
   * Selector integrado en la UI principal
   * Persistencia y exportación/importación de configuraciones
2. **Panel de Administración**:  
   * URL separada (/admin.html) con autenticación PIN
   * Interfaz intuitiva para modificar TODOS los valores
   * Vista previa de cambios en tiempo real
   * Historial de modificaciones
   * Sin necesidad de tocar código
3. **Generación de PDF Profesional**:  
   * Diseño ejecutivo con logo y branding
   * Resumen con KPIs visuales
   * Gráficos de composición de comisión
   * Tablas detalladas con barras de progreso
   * Código de verificación único
   * Múltiples páginas con numeración

### **Arquitectura Mejorada:**

1. **Modularidad total** con separación clara de responsabilidades
2. **Configuración dinámica** por perfil sin modificar código
3. **Panel administrativo** para gestión no técnica
4. **Reportes profesionales** listos para archivo digital

### **Stack Técnico Actualizado:**

* JavaScript ES6+ modular
* jsPDF + Chart.js para PDFs profesionales
* Sistema de perfiles con hot-reload
* Panel admin con validaciones en tiempo real
* LocalStorage versionado con import/export

### **Checklist de Implementación Actualizado:**

* [ ] Implementar sistema de 4 perfiles (sección 19)
* [ ] Crear panel admin en /admin.html (sección 20)
* [ ] Integrar jsPDF para PDFs profesionales (sección 21)
* [ ] Configurar selector de perfiles en UI principal
* [ ] Implementar autenticación PIN para admin
* [ ] Crear templates de PDF con gráficos
* [ ] Testing de migración entre perfiles
* [ ] Validar export/import de configuraciones

Con estas mejoras, el sistema no solo calcula comisiones, sino que se convierte en una plataforma completa y flexible que:

* **Se adapta** a diferentes tipos de comerciales
* **Se gestiona** sin conocimientos técnicos
* **Genera reportes** de calidad profesional

**La aplicación está lista para escalar y adaptarse a las necesidades cambiantes del negocio, manteniendo la calidad y profesionalismo en cada aspecto.**