### **NOMBRE DEL PROYECTO: Gym Pro**

**Integrantes del proyecto:**

**Lucas Juarez**

**Felipe Charabora**

**Juan Cruz Cetro**

**Guillermo Saracini**

### 

### **1. Descripción de la Problemática del MVP**

En el contexto actual, el cliente requiere una solución automatizada para gestionar la venta de planes mediante cobros electrónicos, tanto en modalidad de débito automático como en pagos en cuotas. Actualmente, este proceso se realiza de forma manual, lo que genera ineficiencias, riesgos de error humano, y dificultades para escalar las operaciones de forma ordenada y transparente.

No existe una plataforma centralizada donde los usuarios puedan contratar un servicio directamente con su tarjeta de forma segura, ni un sistema automatizado que gestione los cobros periódicos y registre los resultados en tiempo real. Además, se requiere una trazabilidad clara de los pagos, incluyendo su estado (aprobado o rechazado), así como un mecanismo automatizado de notificación al equipo administrativo.

Por lo tanto, se plantea como necesidad principal el desarrollo de una **Landing Page** funcional e intuitiva que permita a los usuarios ingresar sus datos, elegir la modalidad de pago, y procesar los cobros mediante la API de **Payway**. A su vez, se busca integrar esta solución con **Google Apps Script** y **Google Sheets** para registrar los tokens de las tarjetas, ejecutar cobros diarios de forma automática, almacenar el estado de las transacciones, y generar notificaciones por correo electrónico al backoffice, sin depender de un backend tradicional.

Esta problemática impacta directamente en la eficiencia operativa del negocio, la experiencia del usuario final y la seguridad del proceso de cobro.

### **2. Descripción de la Solución**

Este proyecto implementa una solución integral compuesta por una **Landing Page interactiva** y un sistema de **automatización de cobros** conectado a la API de **Payway**, utilizando **Google Apps Script** y **Google Sheets** como backend ligero.

#### **Componentes principales:**

**Landing Page (Frontend)** Diseñada con HTML, CSS y JavaScript puro (o framework según necesidad), la interfaz permite a los usuarios:

* Seleccionar entre *Débito Automático* o *Pago en Cuotas*.
* Aceptar términos y condiciones para activar la operación.
* Ingresar los datos de su tarjeta, que se envían directamente a la API de Payway para tokenización.
* Enviar el token y los datos del cliente a una hoja de cálculo en Google Sheets vía una integración con Apps Script.

**Google Apps Script (Backend ligero)** Un script vinculado a Google Sheets gestiona:

* Recepción de datos de clientes y tokens.
* Procesamiento automático de pagos mediante la API de Payway (en modalidad diaria, 23:30 hs).
* Registro de cada operación (exitosa o fallida) directamente en la hoja de cálculo.
* Envío automático de correos electrónicos al backoffice con el resultado de cada transacción.

#### **Flujo General:**

1. Usuario completa formulario en la Landing.
2. Datos se tokenizan y se envían a Google Sheets.
3. Un script ejecutado diariamente realiza los cobros.
4. Los resultados se registran y notifican automáticamente.

#### **Tecnologías utilizadas:**

* HTML, CSS, JavaScript
* Google Apps Script
* Google Sheets (como base de datos)
* API REST de Payway

Esta solución permite gestionar cobros de forma segura, automatizada y sin servidores dedicados, facilitando el control y seguimiento administrativo.

### **3. Análisis de Requerimientos**

#### **Requerimientos Funcionales (RF)**

1. **RF01 - Formulario Web Dinámico** Permite a los usuarios seleccionar entre distintas modalidades de pago, ingresar datos personales y de tarjeta.  
    *Justificación*: Es el punto de contacto principal con el usuario y esencial para iniciar el flujo de contratación y pago.
2. **RF02 - Tokenización vía API de Payway** Envía los datos ingresados por el usuario a Payway y recibe un token único de transacción.  
    *Justificación*: Cumple con los estándares de seguridad PCI-DSS al evitar el almacenamiento de datos sensibles.
3. **RF03 - Registro de datos en Google Sheets** Almacena token, datos del cliente y fecha/hora de envío.  
    *Justificación*: Permite trazabilidad simple sin necesidad de una base de datos tradicional.
4. **RF04 - Automatización de cobros diarios (23:30 hs)** Ejecuta cobros recurrentes utilizando Google Apps Script.  
    *Justificación*: Ahorra tiempo operativo y minimiza errores humanos.
5. **RF05 - Registro de estados de cobro** Guarda en Google Sheets si un cobro fue exitoso o rechazado.  
    *Justificación*: Clave para seguimiento, auditoría y toma de decisiones.
6. **RF06 - Notificaciones por correo al backoffice** Informa al área administrativa sobre el resultado de los cobros.  
    *Justificación*: Mejora la coordinación operativa y atención al cliente.

#### **Requerimientos No Funcionales (RNF)**

1. **RNF01 - Seguridad** La solución no almacena datos de tarjetas localmente y utiliza tokenización con Payway.  
    *Justificación*: Minimiza riesgos legales y técnicos al delegar la seguridad a un proveedor certificado.
2. **RNF02 - Disponibilidad 24/7** La landing debe estar siempre disponible, al igual que el script de cobros.  
    *Justificación*: Los usuarios deben poder contratar el servicio en cualquier momento.
3. **RNF03 - Escalabilidad básica** Aunque es una solución simple, debe soportar múltiples registros diarios sin fallos.  
    *Justificación*: Permite crecer sin rediseñar el sistema desde cero.
4. **RNF04 - Costo mínimo de infraestructura** Se evita usar servidores dedicados u hosting complejos.  
    *Justificación*: Ideal para emprendimientos o proyectos pequeños sin grandes presupuestos iniciales.

### **4. Listado de Funcionalidades**

* Landing Page funcional y responsiva.
* Formulario de contratación con selección de modalidad de pago.
* Validación y envío seguro de datos a Payway (tokenización).
* Envío automático de token + datos a Google Sheets.
* Script automático diario que procesa cobros.
* Registro de pagos en Google Sheets.
* Notificación por correo con estado del pago.
* Diseño adaptable a dispositivos móviles (responsive).
* Aceptación explícita de términos y condiciones por parte del usuario.

### **5. Elección de Entorno**

#### **Entorno Elegido: Web**

##### **Justificación:**

* Permite el acceso multiplataforma sin necesidad de instalación.
* Más accesible para usuarios finales y administradores.
* Compatible con Payway y Google Apps Script.

##### **Pros:**

* Acceso desde cualquier navegador (PC o móvil).
* ️Despliegue simple.
* Costo de mantenimiento muy bajo.
* Ideal para campañas, formularios de suscripción, e integraciones.

##### **Contras:**

* Requiere conexión a Internet constante.
* Menor personalización que un entorno mobile nativo.
* Limitado en notificaciones push y acceso a hardware.

##### **Por qué NO Mobile o Escritorio:**

* Mobile nativo implicaría costos y complejidad innecesarios para una funcionalidad simple.
* Escritorio carece de sentido en este contexto de contratación de servicios online.

### **6. Elección de Lenguajes para su Desarrollo**

#### **Lenguajes y tecnologías seleccionadas:**

* **Frontend (Landing Page)**:  
  + HTML + CSS + JavaScript
  + Framework opcional: Bootstrap o Tailwind (para diseño rápido)
* **Backend (ligero)**:  
  + Google Apps Script (basado en JavaScript)

#### **Justificación:**

* **HTML/CSS/JS**: Estándar web, ideal para formularios dinámicos.
* **Google Apps Script**: Fácil de integrar con Google Sheets, sin servidores ni base de datos externos.
* **JavaScript**: Unifica el lenguaje entre frontend y backend.

#### **Pros:**

* Curva de aprendizaje baja.
* Alta compatibilidad con navegadores.
* No se requieren servidores ni bases de datos externos.
* Rápido desarrollo y fácil mantenimiento.

#### **Contras:**

* Google Apps Script puede tener limitaciones de rendimiento si el volumen crece mucho.
* El stack es poco robusto para soluciones empresariales de gran escala.
* JS en frontend requiere una buena validación para evitar errores o malas prácticas.

**10. Estimación de entrega: 2 semanas**

**15. Propuesta Funcional**

La propuesta se estructura en dos fases, comenzando con la entrega de un Producto Mínimo Viable (MVP) funcional, y continuando luego con una evolución tecnológica integral para garantizar mayor escalabilidad, mantenimiento y robustez a futuro.

#### Fase 1 - MVP Inicial (Entregado al Cliente)

* Implementación de una Landing Page estática, desarrollada con HTML, CSS y JavaScript.
* Conexión directa con la API de Payway para tokenización de tarjetas.
* Integración con Google Apps Script y Google Sheets para registro de clientes, ejecución de cobros automáticos y envíos de notificaciones.
* Automatización de tareas mediante scripts programados (23:30 hs diarios).
* Solución económica, funcional y de rápida implementación.

Objetivo: Validar el flujo completo de contratación y cobro con mínima inversión y sin infraestructura compleja.

#### Fase 2 - Mejora Evolutiva (Post-MVP)

Tras la validación del MVP, se propone una reingeniería del software, enfocada en robustecer la solución y prepararla para un crecimiento sostenido:

##### Backend — Node.js + Express

* Reemplazo de Google Apps Script por un servidor backend propio.
* Desarrollo de una API REST que permita:  
  + Registrar usuarios y tokens.
  + Ejecutar cobros programados o manuales.
  + Consultar historial de pagos.
  + Integrarse con un dashboard de administración.
* Mayor control sobre la lógica del negocio, rendimiento y seguridad.

##### Frontend — React.js

* Migración de la Landing Page a una SPA (Single Page Application) desarrollada en React.
* Mejoras en la experiencia del usuario (UX) y rendimiento.
* Validaciones de formularios más robustas.
* Navegación más fluida y modular.

##### Base de datos — MongoDB o PostgreSQL

* Reemplazo de Google Sheets por una base de datos real para mayor integridad de datos y consultas eficientes.
* Posibilidad de añadir funcionalidades como reportes, auditoría y estadísticas de uso.

##### Beneficios esperados:

* Mayor escalabilidad ante crecimiento de usuarios.
* Seguridad mejorada y trazabilidad más robusta.
* Desacoplamiento entre frontend y backend para mayor mantenibilidad.
* Integración futura más simple con nuevos módulos (ej: suscripciones, usuarios, backoffice).
* Posibilidad de desplegar en servidores propios o nube (Ej: Vercel + Render, DigitalOcean, etc).