# **APP “MMAstrología”**

## 🧱 **Estructura General de la App de Mary**

### 🛒 1. **Gestión de Productos**

* Crear, editar, eliminar, ver productos
* Subida de imágenes con Cloudinary
* Mostrar cards en la tienda
* Integración con Mercado Pago

### 📝 2. **Blog de artículos y novedades**

* Crear artículos con título, contenido, imagen, fecha
* Listado público tipo blog
* Admin para Mary o vos

### 🎓 3. **Venta de Cursos o Aulas**

* Crear cursos (nombre, descripción, imagen, precio, enlace al contenido)
* Integrar pasarela de pago
* Opción 1: Cursos grabados (YouTube privado, Vimeo, Drive)
* Opción 2: Cursos en vivo (Google Meet, Zoom, link de WhatsApp)

### 🔮 4. **Servicios espirituales (turnos o contacto)**

* Página con descripción de servicios:
  + Registros Akáshicos
  + Revolución Solar
  + Constelaciones Cuánticas
  + Runas
  + Reiki
* Botón de contacto directo por WhatsApp o formulario de contacto. Reservar turno desde la página.

## 🧭 Roadmap técnico

Te propongo el siguiente orden de desarrollo:

| **Etapa** | **Funcionalidad** | **Tecnologías clave** |
| --- | --- | --- |
| 1 | Backend base (TypeScript + Express + MongoDB) | Mongoose, JWT |
| 2 | Admin de productos (CRUD + Cloudinary) | Multer o Cloudinary SDK |
| 3 | Tienda pública con pasarela de pago | React + Mercado Pago Checkout |
| 4 | Blog de artículos (CRUD) | Mongoose, Markdown o Rich Text |
| 5 | Página de servicios con botón de contacto | React frontend |
| 6 | Cursos (publicación + pago + acceso) | React + MP + links externos |

## 🛠️ ¿Cómo seguimos?

Puedo prepararte el backend inicial con TypeScript para que comiences ya mismo, incluyendo:

* Conexión a MongoDB
* Modelo de Producto
* Rutas protegidas CRUD
* Subida a Cloudinary

Y después agregamos:

* Modelo de Artículo (blog)
* Modelo de Curso

## 🛍️ Función: **Creación y Gestión de Productos**

### 🔧 ¿Qué necesita Mary hacer?

* Crear un nuevo producto (nombre, descripción, precio, imagen, stock)
* Editar y eliminar productos
* Ver listado de productos
* Publicarlos en una tienda visible para sus clientas

## 🧱 Estructura básica

### 📦 1. **Modelo de Producto (Backend - Node.js + MongoDB o PostgreSQL)**

js

CopiarEditar

// Ejemplo en MongoDB con Mongoose

const productSchema = new mongoose.Schema({

name: String,

description: String,

price: Number,

imageUrl: String,

stock: Number,

createdAt: { type: Date, default: Date.now }

});

### 🔐 2. **Rutas protegidas para creación y edición (Node.js)**

* POST /api/products: crear producto
* PUT /api/products/:id: editar
* DELETE /api/products/:id: eliminar
* GET /api/products: listar todos
* GET /api/products/:id: uno solo

**Protegé estas rutas con autenticación básica** (aunque sea solo para vos por ahora).

### 💻 3. **Panel Admin en React para vos**

* Formulario con campos:
  + Nombre
  + Descripción
  + Precio
  + Imagen (podés usar Cloudinary para el upload)
  + Stock
* Listado editable con botones: editar 🖊️, borrar ❌

### 🛒 4. **Frontend público (tienda)**

* Página con cards de productos
* Cada producto puede tener:
  + Imagen
  + Nombre
  + Precio
  + Botón “Agregar al carrito” o “Comprar”

### 💸 5. **Integración con pagos**

Más adelante:

* Mercado Pago o Stripe Checkout para vender productos reales
* Link directo o botón dentro de la app

### 🌤️ ¿Qué podés usar para simplificar?

| **Necesidad** | **Solución recomendada** |
| --- | --- |
| Subida de imágenes | [Cloudinary](https://cloudinary.com/) (fácil, gratis, rápido) |
| Autenticación | JWT simple o una clave de admin hardcodeada al inicio |
| Base de datos | MongoDB (Atlas) o PostgreSQL (Railway) |
| Hosting | Frontend en Vercel, backend en Railway |