# Sorteazo

## Descripción

Sorteazo es una aplicación web para gestionar sorteos en línea: creadores autentican y configuran sorteos; compradores eligen y pagan números sin necesidad de registro.

## Tabla de Contenidos

1. [Casos de Uso](#casos-de-uso)
2. [Requisitos Funcionales](#requisitos-funcionales)
3. [Requisitos No Funcionales](#requisitos-no-funcionales)
4. [Modelo de Datos](#modelo-de-datos)
5. [Flujo de Pago y Webhooks](#flujo-de-pago-y-webhooks)
6. [Arquitectura](#arquitectura)
7. [API Endpoints](#api-endpoints)

## Casos de Uso

### 1. Registro y Autenticación

* **Actor**: Creador de sorteo
* **Precondición**: Usuario no autenticado.
* **Flujo principal**:
  1. Accede a registro.
  2. Ingresa email y contraseña.
  3. Obtiene token JWT.

### 2. Creación de Sorteo

* **Actor**: Creador autenticado
* **Flujo principal**:
  1. Completa título, descripción de premios, rango de números y alias/CBU.
  2. Sistema guarda sorteo y genera rango de números.

### 3. Actualización de Sorteo

* **Actor**: Creador autenticado
* **Flujo principal**:
  1. Selecciona sorteo.
  2. Modifica descripción de premios.

### 4. Visualización de Sorteo y Números

* **Actor**: Comprador
* **Flujo principal**:
  1. Consulta listado público de sorteos.
  2. Selecciona sorteo y ve estado de cada número (libre/ocupado).

### 5. Compra de Número

* **Actor**: Comprador
* **Flujo principal**:
  1. Elige número libre.
  2. Ingresa nombre y contacto.
  3. Realiza pago (MercadoPago o tarjeta).

### 6. Notificación de Ganador

* **Actor**: Creador
* **Precondición**: Sorteo cerrado o listo para definir ganador.
* **Flujo principal (a botón)**:
  1. El creador accede al panel del sorteo y presiona el botón “Sortear Ganador”.
  2. El sistema toma todos los números vendidos (estado “ocupados”) y selecciona uno de forma aleatoria.
  3. Genera la notificación al creador con los datos del comprador ganador.
* **Flujo alternativo (manual externo)**:
  1. El creador revisa los resultados externos (ej. lotería, quiniela).
  2. Accede a la sección “Definir Ganador” en el panel del sorteo.
  3. Selecciona manualmente el número ganador de la lista de números vendidos.
  4. El sistema valida que el número exista y esté pagado.
  5. Genera la notificación al creador con datos del comprador ganador.

### 7. Pausar Sorteo

* **Actor**: Creador autenticado
* **Precondición**: Sorteo activo y con números aún disponibles.
* **Flujo principal**:
  1. El creador selecciona la opción “Pausar Sorteo” en el panel de administración.
  2. El sistema cambia el estado del sorteo a “Pausado”, impidiendo nuevas compras.
  3. El creador recibe confirmación de que el sorteo está en pausa.

## Relaciones entre Casos de Uso

En el diagrama de casos de uso, las relaciones se representan con flechas y estereotipos:

* **<>** (incluir):
  + **Flecha punteada con un triángulo hueco** que apunta **del caso de uso base** al **caso incluido**.
  + Indica que la ejecución del caso base siempre incorpora el comportamiento del caso incluido.
* **<>** (extensión):
  + **Flecha punteada con un triángulo hueco** que apunta **del caso de uso extendido** al **caso base**.
  + Señala que el caso de uso extendido añade comportamiento opcional al caso base bajo ciertas condiciones.

### Relaciones definidas

* **Registro y Autenticación <>**
  + *De → a*: de **Creación de Sorteo**, **Actualización de Sorteo**, **Pausar Sorteo** y **Notificación de Ganador** → a **Registro y Autenticación**.
* **Visualización de Sorteo y Números <>**
  + *De → a*: de **Compra de Número** → a **Visualización de Sorteo y Números**.
* **Notificación de Ganador <>**
  + *De → a*: de **Notificación de Ganador** → al caso base **Pausar Sorteo** (o cierre de sorteo implícito), según el flujo de “Sortear Ganador”.
* **Pausar Sorteo <>**
  + *De → a*: de **Pausar Sorteo** → **Actualización de Sorteo**.

## Requisitos Funcionales

* **RF1**: Autenticación JWT para creadores.
* **RF2**: CRUD de sorteos y generación de números.
* **RF3**: Endpoint público para consulta de sorteos y números.
* **RF4**: Proceso de compra: captura de datos y creación de preferencia de pago.
* **RF5**: Webhook de MercadoPago para actualizar estado de pagos.
* **RF6**: Envío de notificaciones al creador con datos del comprador.

## Requisitos No Funcionales

* **RNF1**: Respuesta <200 ms en consultas de lotes de números.
* **RNF2**: Disponibilidad 99.9%.
* **RNF3**: Seguridad (HTTPS, validación de inputs, CORS).
* **RNF4**: Escalabilidad para hasta 10.000 números por sorteo.

## Modelo de Datos

UsuarioCreador(  
 id PK,  
 email UNIQUE,  
 password\_hash,  
 created\_at  
)  
  
Sorteo(  
 id PK,  
 creador\_id FK,  
 titulo,  
 descripcion\_premios,  
 rango\_min,  
 rango\_max,  
 alias\_cbu,  
 fecha\_inicio,  
 fecha\_cierre,  
 created\_at,  
 updated\_at  
)  
  
Numero(  
 id PK,  
 sorteo\_id FK,  
 valor INT,  
 estado ENUM(libres,ocupados),  
 comprador\_nombre NULL,  
 comprador\_contacto NULL,  
 pago\_id FK NULL  
)  
  
Pago(  
 id PK,  
 numero\_id FK,  
 preferencia\_mp\_id,  
 estado ENUM(pendiente,completado,fallido),  
 monto,  
 metodo ENUM(mp,tarjeta),  
 fecha  
)

## Flujo de Pago y Webhooks

1. **Crear preferencia**: Backend llama API de MercadoPago con ítem (número, monto).
2. **Checkout**: Frontend redirige o integra popup de pago.
3. **Webhook**: MercadoPago envía notificación al endpoint.
4. **Actualizar estado**: Backend valida firma y actualiza Pago.estado y Numero.estado.
5. **Confirmación**: Frontend notifica resultado al comprador.

## Arquitectura

Para soportar notificaciones push en Sorteazo, se integra un **Service Worker** que:

* Solicita permiso al usuario para recibir notificaciones.
* Gestiona las suscripciones al Push API.
* Escucha eventos de push y muestra notificaciones emergentes.

**Frontend**: SPA (React/Vue/Angular), maneja UI y llamadas API.

* **Backend**: Node.js/Express (o similar), gestiona autenticación, lógica, pagos.
* **Base de Datos**: PostgreSQL o MySQL.

## API Endpoints

| Método | Ruta | Descripción |
| --- | --- | --- |
| POST | /api/auth/register | Crear cuenta de creador |
| POST | /api/auth/login | Obtener JWT |
| GET | /api/sorteos | Listar sorteos |
| POST | /api/sorteos | Crear sorteo |
| GET | /api/sorteos/:id | Detalle sorteo |
| GET | /api/sorteos/:id/numeros | Listar números |
| PATCH | /api/sorteos/:id | Actualizar sorteo |
| DELETE | /api/sorteos/:id | Eliminar sorteo |
| POST | /api/sorteos/:id/comprar | Iniciar compra y crear preferencia |
| POST | /api/webhooks/mercadopago | Recibir notificaciones de pago |

## Infraestructura y Despliegue en AWS (Free Tier)

Para profundizar en la nube y desplegar Sorteazo usando servicios AWS gratuitos:

1. **Frontend – AWS Amplify**
   * Conecta tu repositorio (GitHub/GitLab) a Amplify.
   * Define el comando de build (por ejemplo, npm run build) y la carpeta de publicación (build o dist).
   * Amplify provee hosting en CDN con SSL automático y manejo de dominios.
   * **Free Tier**: 5 GB almacenamiento, 15 GB transferencia/mes.
2. **Backend – AWS Lambda + API Gateway**
   * Empaqueta cada endpoint de tu API como función Lambda (Node.js).
   * Configura API Gateway para exponer rutas HTTP hacia tus Lambdas.
   * Gestiona variables de entorno (JWT\_SECRET, MP\_ACCESS\_TOKEN) en la consola de Lambda.
   * **Free Tier**: 1 M invocaciones; 400 000 GB‑segundo cómputo/mes.
3. **Base de Datos – Amazon RDS (PostgreSQL)**
   * Crea una instancia db.t2.micro en RDS con PostgreSQL.
   * Permite acceso desde tus Lambdas (configura VPC y grupos de seguridad mínimos).
   * **Free Tier (12 meses)**: 750 horas de instancia; 20 GB almacenamiento.
4. **Webhooks y Notificaciones**
   * Define un endpoint en API Gateway para /api/webhooks/mercadopago, apuntando a una Lambda.
   * Para notificaciones push, registra tu Service Worker y usa Amazon SNS (Simple Notification Service) si quieres enviar mensajes programados.
5. **CICD y Monitorización**
   * Amplify incluye pipelines automáticos (push → build → deploy).
   * Usa CloudWatch para logs de Lambda y métricas de API Gateway.
6. **Puntos de Aprendizaje Clave**
   * Gestión de identidades y permisos (IAM Roles para Lambda y API Gateway).
   * Networking básico (VPC, Subredes, Security Groups).
   * Monitorización y alertas con CloudWatch.

*Fin de la documentación de especificación.*