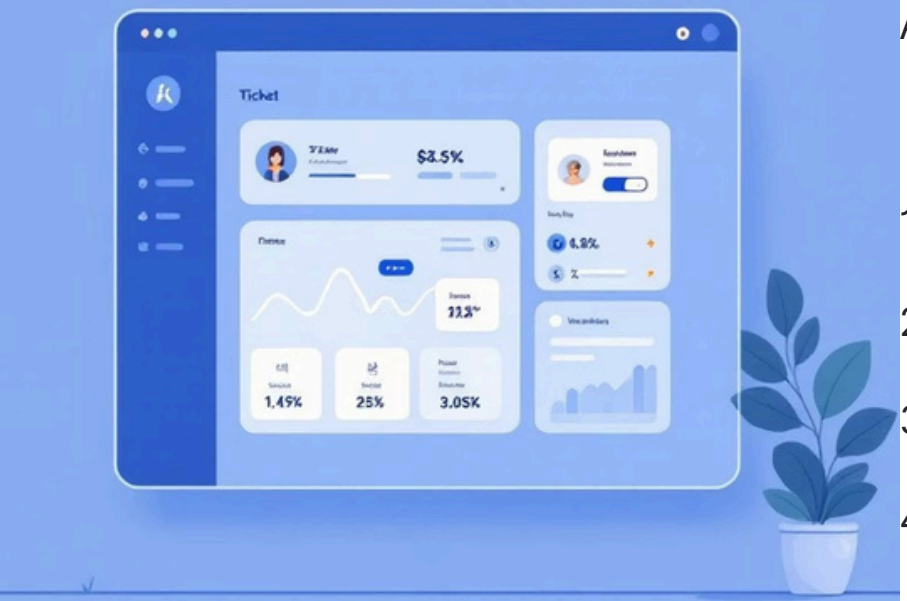


Sistema de Gestión de Tickets

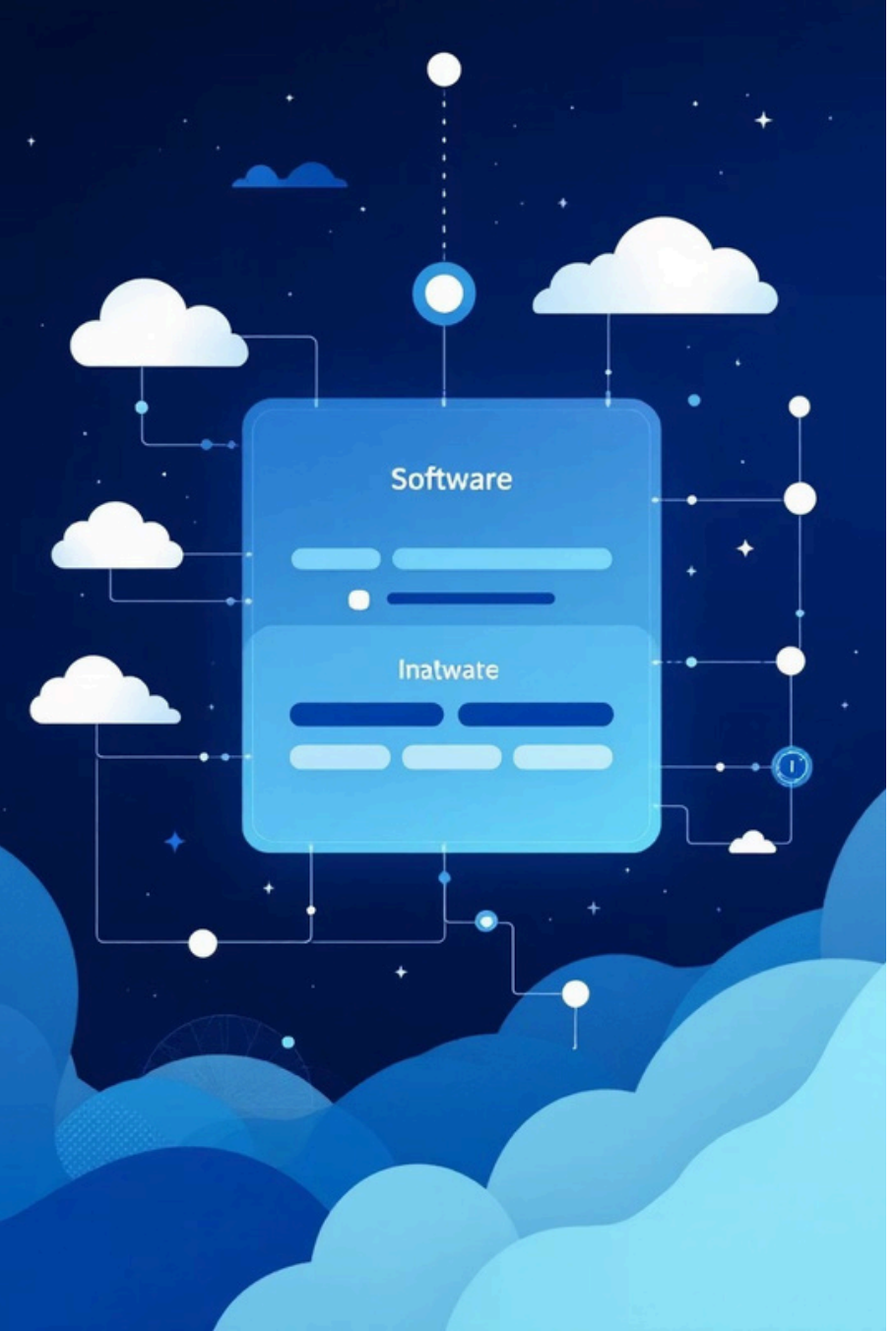
Arquitectura de seguridad y backend para plataforma de venta de entradas

1-Jose Adderly Oncebay Pumaccari

2-Yoel Alex Huamani Tancayllo

3-Emerson Franz Sarayasi Pinto

4-Jonathan Wilbert Bautista Cayllahua



Resumen de la Solución

Sistema integral que combinagestión de usuarios, procesamiento seguro
de pagos y administración completa del ciclo de vida de los tickets.
Arquitectura diseñada para escalabilidad y máxima seguridad.

1 **Base de datos robusta**

MySQL/PostgreSQL con estructura optimizada

2 **Autenticación segura**

JWT tokens con cifrado HS256

3 **Operaciones CRUD completas**

Gestión integral de tickets y órdenes

Arquitectura de Base de Datos

Estructura Principal

Sistema basado en **MySQL** o **PostgreSQL** que garantiza integridad referencial y rendimiento óptimo.

- **users:** Información de usuarios y credenciales
- **tickets:** Catálogo de eventos y disponibilidad
- **orders:** Historial de transacciones y estados

Relaciones normalizadas que optimizan consultas y mantienen consistencia de datos.

Librerías y Dependencias Clave

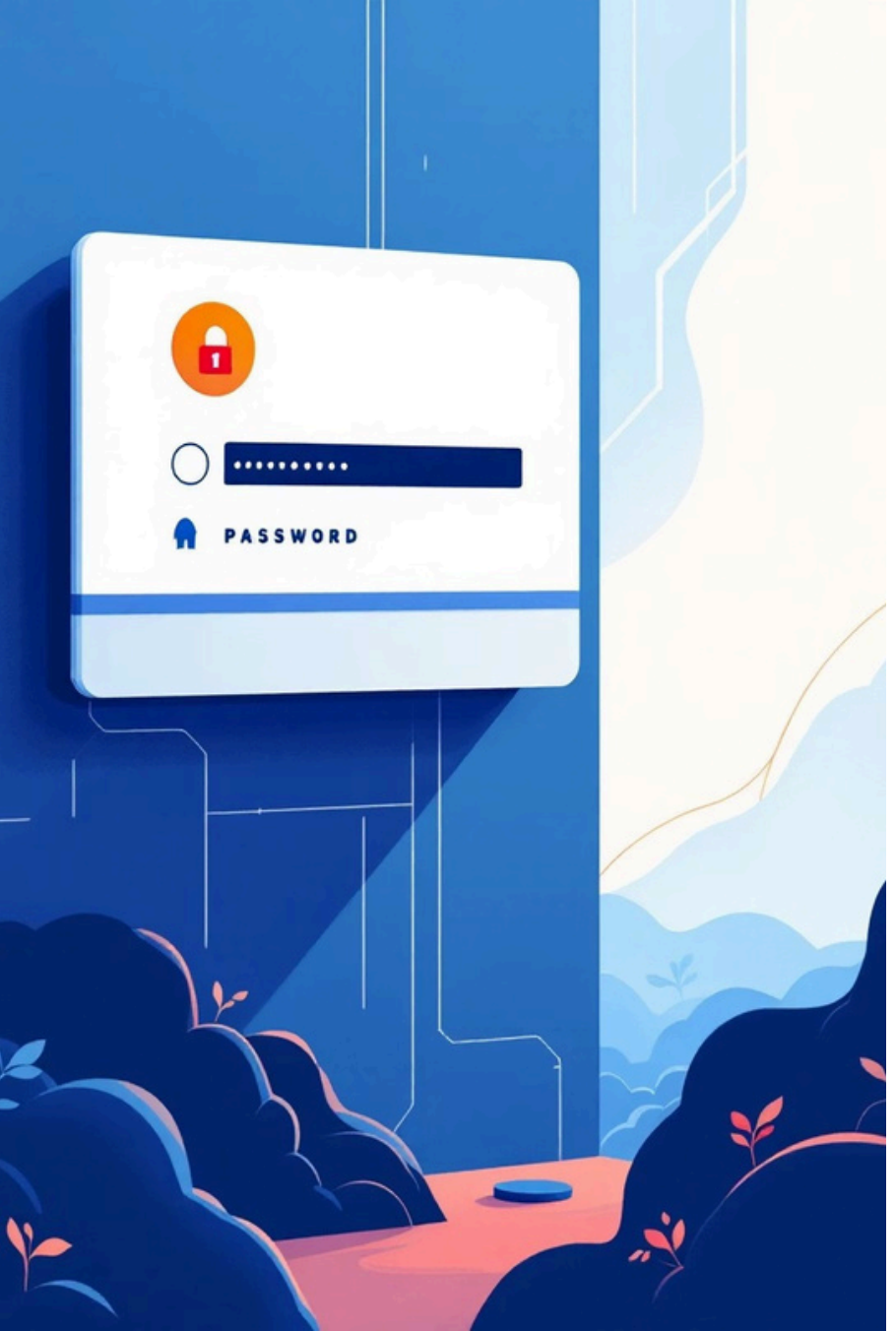
Para asegurar un desarrollo eficiente y seguro,se recomiendan las siguientes librerías para la implementación del sistema de gestión de tickets:

Backend (Node.js + Express)

- **express:** Framework web robusto para construir la API RESTful de forma escalable.
- **bcrypt:** Esencial para el hash seguro de contraseñas de usuarios, protegiéndolas contra ataques.
- **jsonwebtoken:** Gestión de la generación, firma y verificación de JSON Web Tokens (JWT) para una autenticación sin estado.
- **dotenv:** Para una configuración segura y fácil del entorno,
  cargando variables de un archivo .env.
- **pg** o **mysql2:** Controladores de bases de datos para establecer conexiones con PostgreSQL o MySQL, respectivamente.
- **sequelize** o **typeorm:** ORM (Object-Relational Mapper) para modelar y manipular datos de la base de datos con sintaxis orientada a objetos.
- **express-validator:** Middleware robusto para la validación
  de datos de entrada, garantizando la integridad de la información.

Frontend (Opcional, con React)

- **axios:** Cliente HTTP basado en promesas que facilita el consumo de APIs RESTful desde el navegador.
- **react-router-dom:** Librería estándar para gestionar rutas de usuario y navegación en aplicaciones React de una sola
  página (SPA).
- **jwt-decode:** Utilidad ligera para decodificar JWTs en el cliente, permitiendo el acceso a su contenido sin verificación de firma.



Sistema de Registro y Login

Proceso de Registro

Validaciónde correo electrónico único y contraseña con criterios de seguridad establecidos.

Cifrado de Contraseñas

Implementación de **bcrypt** librería de encriptación

No se deben guardar contraseñas en texto plano, siempre se usan hashes.

Autenticación

Verificacióndecredenciales contra base de datos con manejo de intentos fallidos.

Autenticación JWT

> LostokensJWT(JSON WebTokens) conalgoritmo **HS256** proporcionan autenticación stateless y segura entre cliente y servidor.

01

Generación de Token

Al login exitoso, se genera JWT con payload de usuario y expiración configurable

02

Validación en Endpoints

Middlewar sirve para verificar el token antes de que el usuario acceda a un endpoint protegido

03

Renovación Automática

Sistema de refresh tokens para mantener sesión activa sin reautenticación

Operaciones de Tickets

Gestión completa del ciclodevidadelosticketsconvalidaciones de negocio ycontrol de estados.

Compra de Tickets

Requisito: Usuario autenticado con token válido

Verificación de disponibilidad en tiempo real y reserva temporal durante el proceso de pago

Consulta de Historial

Acceso a tickets comprados, estados actuales y detalles de transacciones pasadas

Modificación de Reservas

Cambios permitidos según políticas de evento y disponibilidad de nuevas fechas/asientos

Cancelación

Proceso de reembolso basado en términos y condiciones del evento específico

Implementación de Seguridad

Cifrado HTTPS

(eselprotocolo seguro que permite transferir información en la web de manera cifrada y protegida)

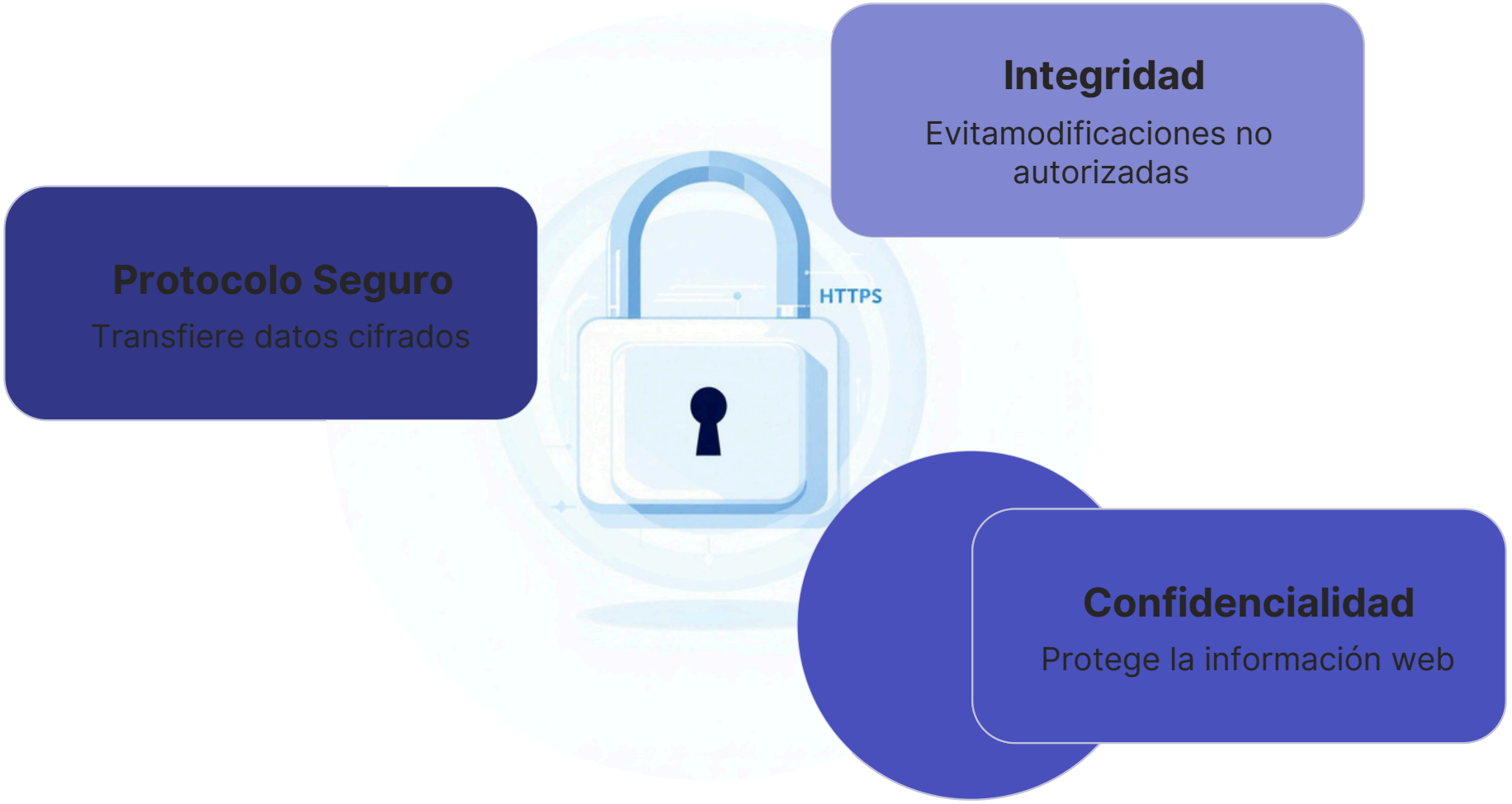
Todas las comunicaciones cifradas con certificados SSL/TLS válidos

Validación de Entrada

Sanitización yvalidacióndetodos los inputs para prevenir inyecciones

Control de Roles

Sistema depermisos granular: usuario estándar vs administrador



Integridad

Evitamodificaciones no autorizadas

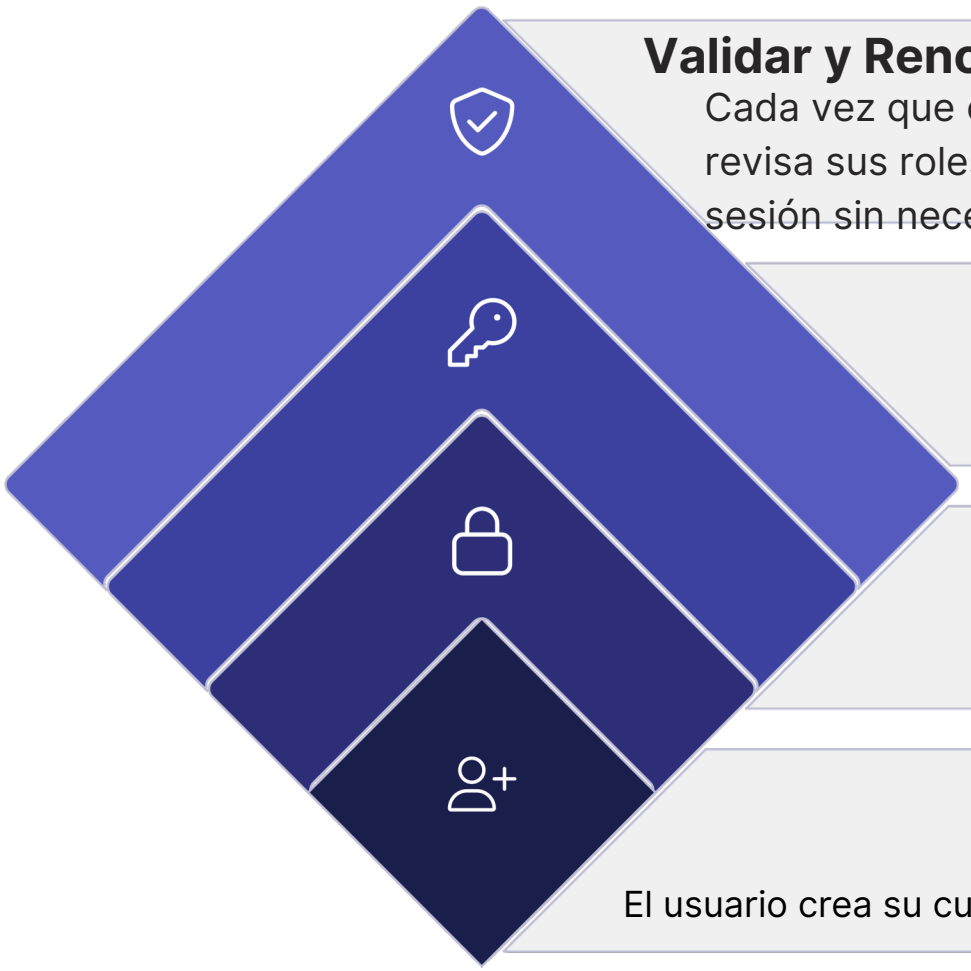
Protocolo Seguro

Transfiere datos cifrados

Confidencialidad

Protege la información web

Flujo de Autenticación



Validar y Renovar

Cada vez que el usuario hace una petición, el sistema valida el token y revisa sus roles. Si el token expira, se puede renovar para seguir con la sesión sin necesidad de volver a loguearse.

Emitir JWT

Si las credenciales son correctas, el sistema genera un token JWT con información del usuario y sus permisos.

Login

Cuando el usuario quiere entrar, se verifican sus credenciales (usuario y contraseña)

Registro

El usuario crea su cuenta, define sus datos y se le asigna un rol.

Diagrama que ilustra el proceso completo de autenticación y autorización, incluyendo puntos de validación y manejo de errores.

Flujo Detallado de Autenticación

Elprocesodeautenticaciónaseguraquesolousuarioslegítimospuedanaccederalos recursos protegidos del sistema.

01

Registro de Usuario

El usuario crea su cuenta con un correo y una contraseña. La contraseña se cifra utilizando **bcrypt** para garantizar su seguridad, almacenando solo el hash.

```
bcrypt.hash(password, 10);
```

02

Inicio de Sesión

Tras ingresar sus credenciales, si son correctas, el servidor genera un **JSON Web Token (JWT)** firmado con el algoritmo **HS256**.

```
jwt.sign({ id: user.id },
"CLAVE_SECRETA");
```

03

Envío del Token

El token actúa como una llave de acceso y se incluye en el encabezado `Authorization` (`Bearer Token`) de cada petición subsiguiente al servidor.

```
Authorization: Bearer <token>
```

04

Verificación por Middleware

Un middleware en el servidor intercepta las peticiones, verifica la validez y autenticidad del JWT. Si es inválido, devuelve un error **401 No Autorizado**.

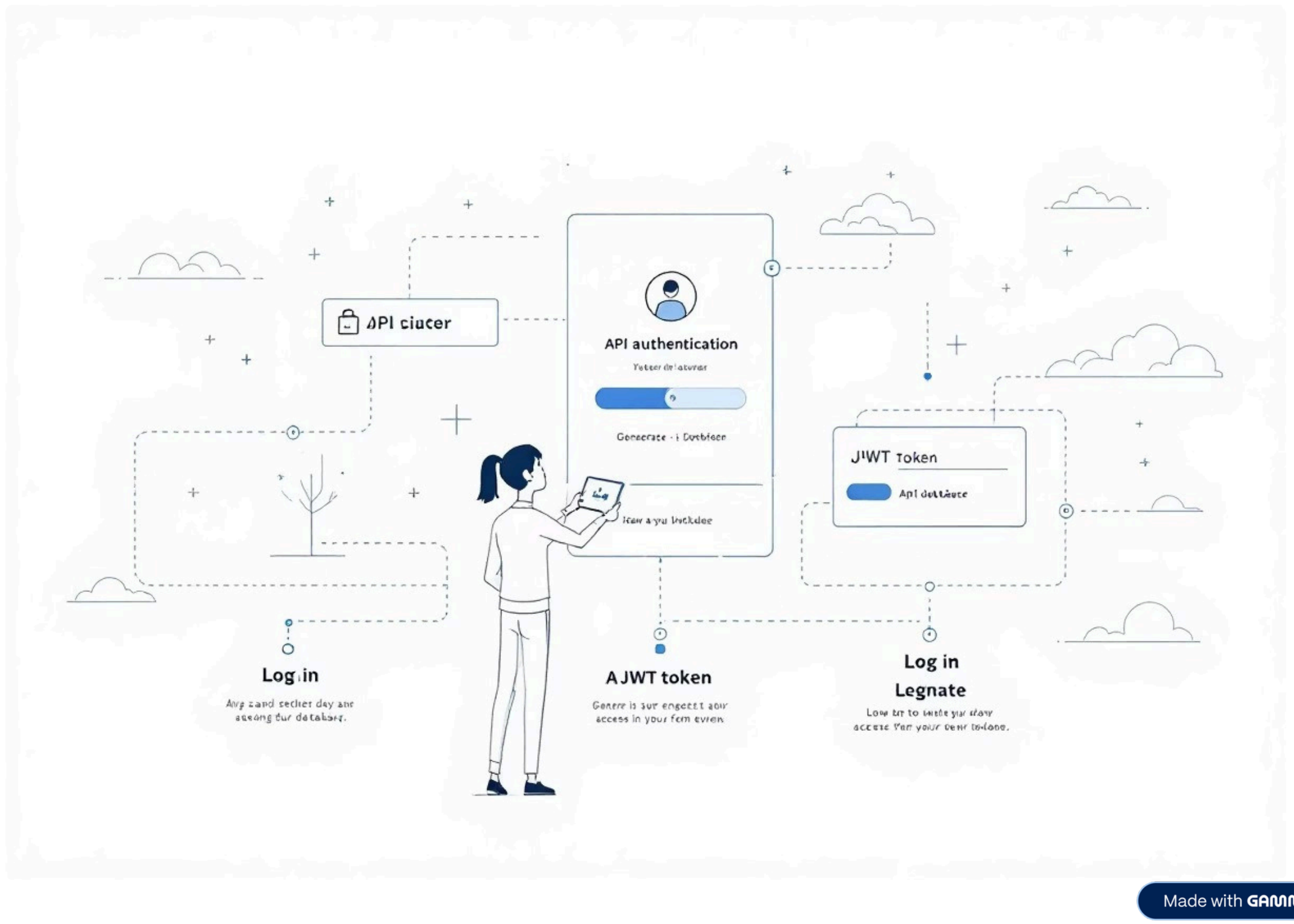
```
jwt.verify(token, "CLAVE_SECRETA");
```

05

Acceso a Recursos

Si el token es válido, se permite el acceso al recurso solicitado (ej., comprar un ticket), ejecutando la operación correspondiente en la base de datos.

```
INSERT INTO orders (user_id, ticket_id)
```



Consideraciones de Implementación

1

Configuración Inicial

Setupdebasededatos,variablesdeentorno y certificados SSL

2

Desarrollo de APIs

Implementacióndeendpoints REST con documentación OpenAPI

3

Testing y Validación

Pruebasunitarias,integración y tests de seguridad

4

Deployment

Configuraciónde entorno productivo con monitoreo continuo

Próximos Pasos

Configurar Entorno

Setupdebasededatosyvariables de configuración

Desarrollar APIs

Implementacióndeendpoints con validaciones de seguridad

Testing Integral

Pruebasdefuncionalidad, rendimiento y seguridad

Contacto: Para dudas técnicas sobre la implementación, documentación adicional está disponible en el repositorio del p roye c to .