2023

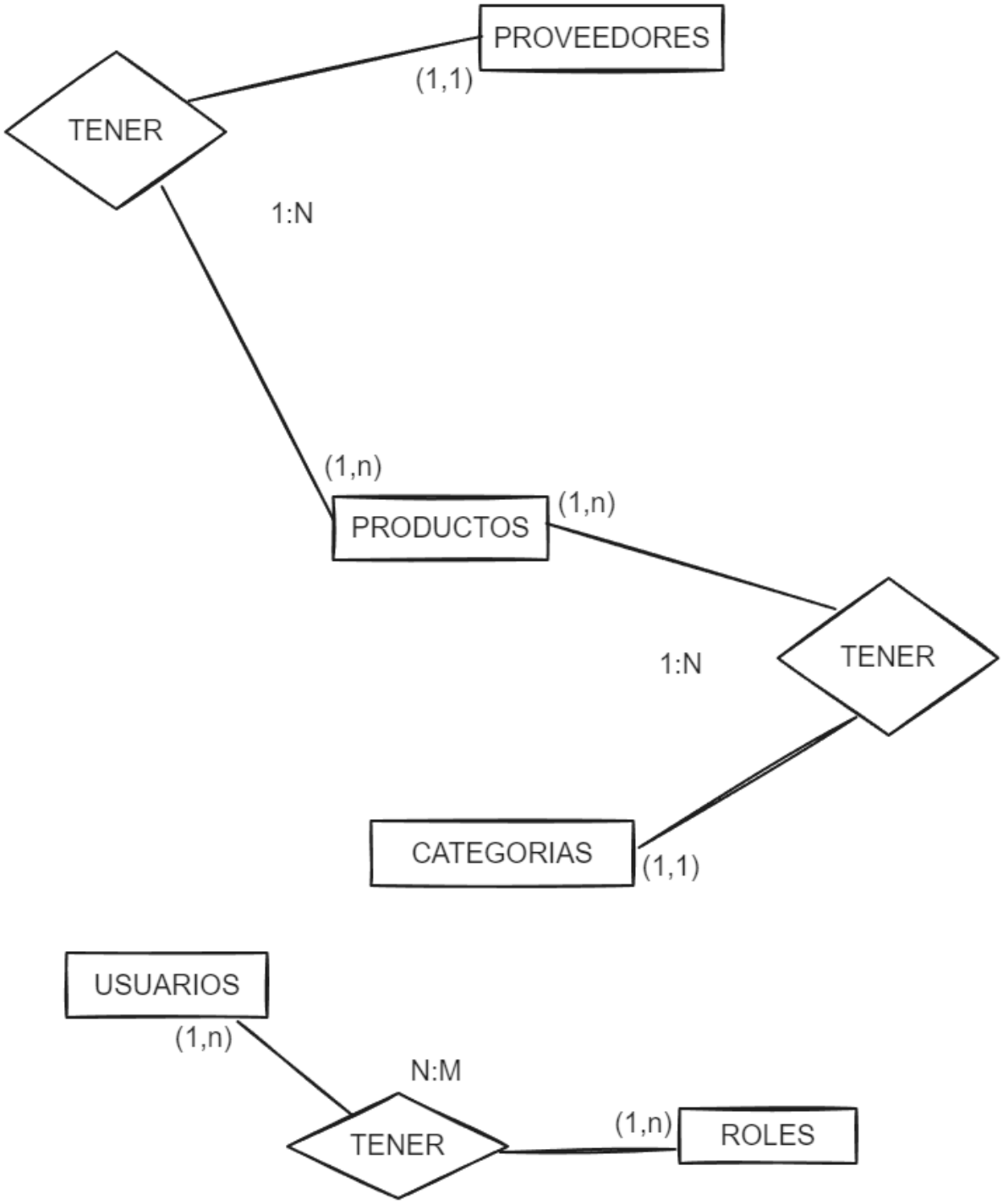
Tienda Informatica

DETALLES Y PRESUPUESTO

CLOWN INFORMATICS

Índice

MODELO ENTIDAD RELACIÓN



Especificaciones técnicas del servidor recomendado

La propuesta de servidor para ejecutar la aplicación en producción va a ser en la red de AWS, debido a que AWS nos ofrece una amplia gama de servidores y nos permite de una forma muy sencilla escalar nuestra aplicación de manera horizontal y asumir sin problema las variaciones en la demanda de nuestra aplicación. Además, AWS ofrece una infraestructura muy confiable y con un alto rendimiento.

Instancias para ejecutar la aplicación JAVA

En primer lugar, para aprovecharnos al máximo de la infraestructura de AWS, vamos a desplegar nuestra aplicación en una instancia de tipo EC2. Las instancias EC2 nos ofrecen máquinas virtuales donde poder ejecutar nuestras aplicaciones. En nuestro caso vamos es escoger una **t4g.medium**, la cual tiene las siguientes especificaciones:

- 2 vCPU
- 4GiB de memoria RAM
- Hasta 5 Gigabit de conexión a la red

En caso, de que a causa del aumento de la demanda nuestra aplicación requiriese de más recursos, se podría mejorar la instancia a una superior o bien aumentar la cantidad de instancias contratadas.

Estas son las instancias a las que sería escalable la que nosotros recomendamos.

Tamaño de instancia	CPU virtual	Memoria (GiB)	Rendimiento base/CPU virtual	Créditos de CPU obtenidos/hora	Ancho de banda de red con ráfagas (Gbps)
t4g.nano	2	0,5	5 %	6	Hasta 5
t4g.micro	2	1	10 %	12	Hasta 5
t4g.small	2	2	20 %	24	Hasta 5
t4g.medium	2	4	20 %	24	Hasta 5
t4g.large	2	8	30 %	36	Hasta 5
t4g.xlarge	4	16	40 %	96	Hasta 5
t4g.2xlarge	8	32	40 %	192	Hasta 5

Cabe recalcar que en este tipo de instancias el precio del internet de entrada no tiene ningún tipo de coste, pero el de salida se paga a 0.05 USD por GB.

Aquí puedes consultar toda la información oficial: AMAZON EC2

Instancias para ejecutar BD de PostgreSQL

Para alojar estas bases de datos, vamos a usar las instancias RDS para PostgreSQL que nos ofrece AWS. Para empezar, nosotros vamos a recomendar el uso de la **db.t4g.medium** para producción. Esta instancia nos ofrece una DB con estas especificaciones:

- 2 vCPU
- 4GiB de memoria RAM
- 30GB de almacenamiento SSD, lo cual seria ampliable cuando se necesitase.

En caso de ser necesario son igual de mejorables que las instancias EC2, podríamos de una manera muy sencilla escoger una instancia superior en momentos puntuales.

Modelo	CPU virtual	Memoria (GiB)	Ancho de banda de EBS con ráfagas (Mbps)
db.t4g.micro	2	1	Hasta 2085
db.t4g.small	2	2	Hasta 2085
db.t4g.medium	2	4	Hasta 2085
db.t4g.large	2	8	Hasta 2780
db.t4g.xlarge	4	16	Hasta 2780
db.t4g.2xlarge	8	32	Hasta 2780

Aquí tienes toda la información oficial: AMAZON RDS

Instancias para ejecutar la BD de MongoDB

Para este tipo de instancias vamos a escoger las de Amazon DocumentDB que son totalmente compatibles con MongoDB, en este tipo de instancia no se paga nada inicialmente ya que todo el precio funciona bajo demanda, se paga el almacenamiento, las operaciones E/S y el tráfico de datos. En este caso, al igual que en RDS vamos a recomendar una instancia **db.t4g.medium**, la cual dispone de lo mismo:

- 2 vCPU
- 4GiB de memoria RAM
- El almacenamiento se cobra en 0.116 USD / mes (no es fijo)

La transferencia de los datos tiene un coste medio de 0.05USD.

Al igual que las otras instancias, está también es igual de escalable con estas opciones superiores.

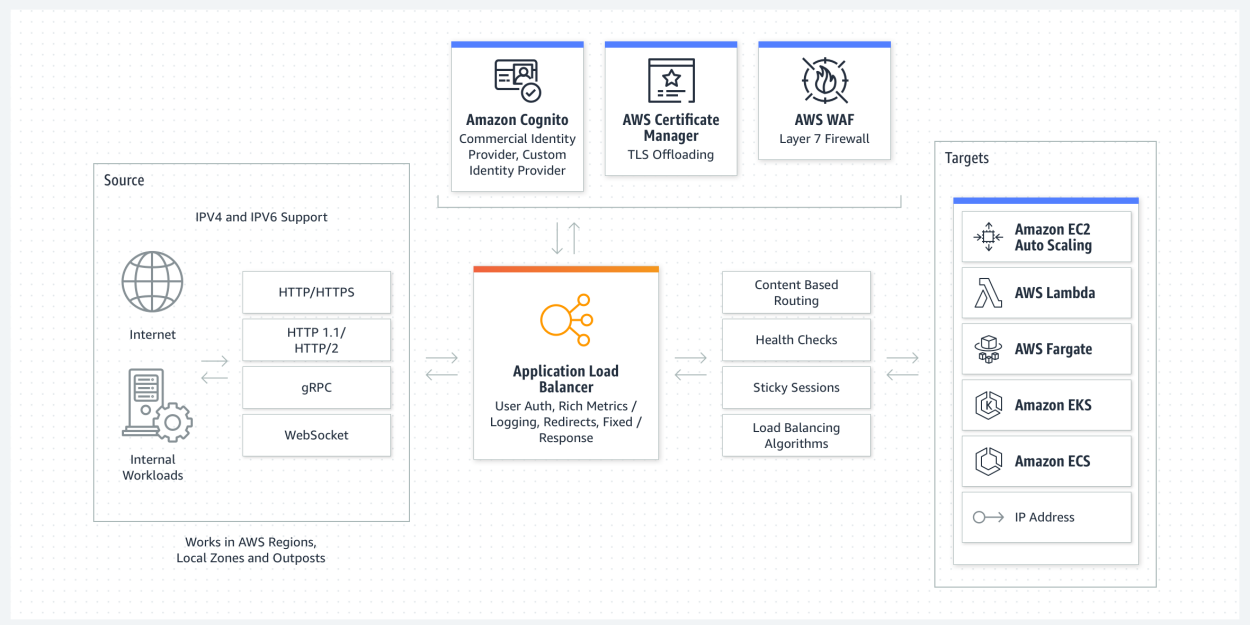
Instancias optimizadas para memoria, generación actual	Memoria	CPU virtual	Precios
db.r6g.large	16 GiB	2	0,304 USD
db.r6g.xlarge	32 GiB	4	0,608 USD
db.r6g.2xlarge	64 GiB	8	1,216 USD
db.r6g.4xlarge	128 GiB	16	2,432 USD
db.r6g.8xlarge	256 GiB	32	4,86286 USD
db.r6g.12xlarge	384 GiB	48	7,296 USD
db.r6g.16xlarge	512 GiB	64	9,72572 USD

Aquí dejo más información a la web oficial: [AMAZON DocumentDB]

Servicios adicionales recomendados

Elastic Load Balancing

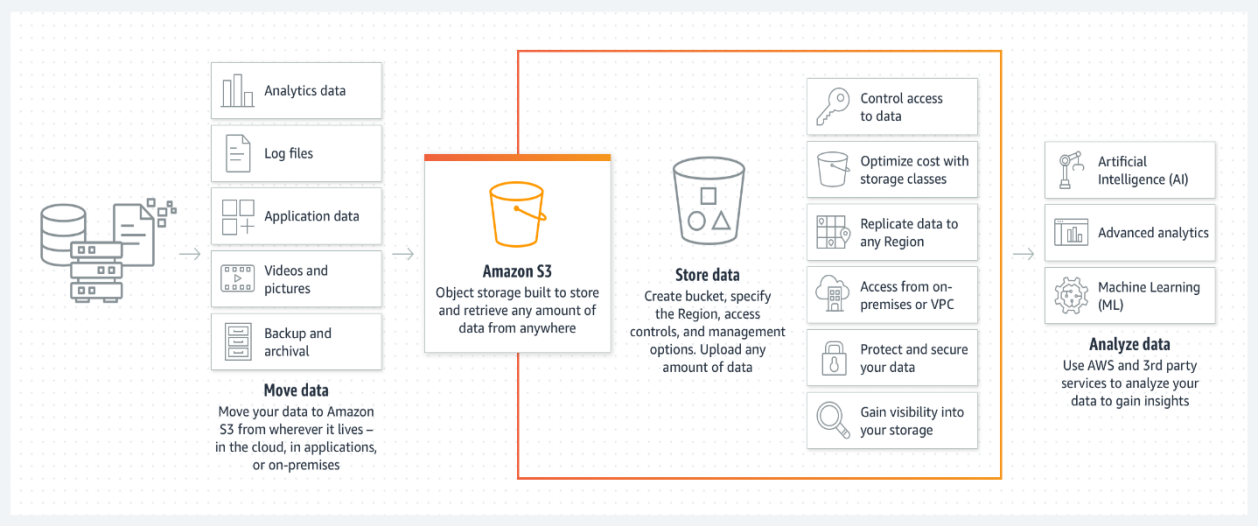
En caso de que la aplicación que tenemos necesite escalar, es muy recomendable adquirir el servicio de Elastic Load Balancing que es un balanceador de carga que reparte el tráfico entrante entre nuestras distintas instancias EC2.



Página oficial del servicio: [AMAZON ELB]

Servicio de Almacenamiento Simple (S3)

Como nuestra aplicación va a tener almacenamiento de ficheros, para las imágenes que se van a subir, seria recomendable adquirir el servicio de almacenamiento por GB que nos ofrece Amazon. Este tiene principalmente un coste de 0.02 USD / por GB al mes, aunque también se factura la cantidad de solicitudes GET y PUT que se hacen.



Estimación de los costes de AWS por mes

* Esto es algo aproximado para un consumo no muy elevado de la aplicación, puede ser superior ya que Amazon cobra en base a lo que se consume.

Aproximación:

- 2 instancias EC2 (**t4g.medium**) = 44.09 USD / mes
    - 2 vCPU
    - 4 GiB de RAM
    - Hasta 5 Gigabit de conexión

- 1 instancia RDS for PostGreSQL (**db.t4g.medium**) = 109.79 USD / mes
    - 2 vCPU
    - 4GiB de memoria RAM
    - 30GB de almacenamiento SSD.

- 1 instancia DocumentDB (**db.t4g.medium**) = 109.79 USD / mes
    - 2 vCPU
    - 4GiB de memoria RAM
    - 30GB de almacenamiento SSD.

- 1 instancia Elastic Load Balancing = 47,60 USB /mes
    - 2 GB por hora

- 1 instancia S3 = 11,51 USD / mes
    - 200GB de almacenamiento
    - 100000 solicitudes PUT, COPY, POST y LIST
    - 200000 solicitudes GET y SELECT

Total 296,33 € / mes

**No se incluye el precio que agrega AWS de 0,05 USD / GB al mes en los datos de salida a Internet para ninguno de los servicios.

Presupuesto (Número)

Clown Informatics (Empresa)	Jose Luis
C/ Paseo Grande, 1 (Dirección)	Calle De Los Pageables
NIF: A12345678	**NIF**:
Teléfono: 612 456 789	**Teléfono**:
Mail: info@clowninformatics.com	**Mail**:

Fecha del presupuesto	01/12/2023	**Validez**	30 días

DESCRIPCIÓN	UNIDADES	PRECIO	TOTAL
Hora de programación	300	25 €	7500,00 €

SUB-TOTAL	7500,00 €
DESCUENTO	00,00 €
IVA %	21,00%
TOTAL PRESUPUESTADO	**9075,00 €**

Firma	*Firma del cliente*
-	-

Elección de tecnologías

Java + Spring

Java con Spring es una elección sólida para desarrollar una REST API por varias razones:

1. **Fácil de usar**: Con Spring Framework, especialmente Spring Boot y Spring MVC, puedes crear APIs rápidamente sin tener que hacer mucha configuración. Es una forma rápida de empezar y hacer que tu API esté lista para usarse.
2. **Perfecto para APIs REST**: Spring está diseñado para ayudarte a hacer APIs REST. Tiene anotaciones (como @GetMapping, @PostMapping) que hacen que sea fácil definir tus endpoints y manejar las solicitudes y respuestas.
3. **Organización y mantenimiento sencillos**: Spring hace que sea fácil organizar tu código y mantenerlo. Con cosas como la inyección de dependencias, puedes hacer tu código modular y más fácil de entender.
4. **Pruebas integradas y desarrollo ágil**: Spring ofrece un entorno propicio para realizar pruebas integradas con facilidad. Mediante herramientas como JUnit y Mockito, puedes escribir pruebas unitarias y de integración para tu API de manera sencilla. Estas pruebas te permiten asegurar el buen funcionamiento de tu API y detectar posibles errores durante el desarrollo, lo que agiliza el proceso de corrección. Además, la estructura modular de Spring facilita la escritura de pruebas, lo que contribuye a un ciclo de desarrollo más rápido y robusto.
5. **Seguridad integrada**: Spring tiene funciones para ayudar a asegurar tu API. Puedes implementar cosas como autenticación y autorización fácilmente para proteger tus servicios.
6. **Muchos recursos y ayuda disponible**: La comunidad de desarrolladores de Spring es enorme. Hay muchos tutoriales, documentación y foros donde puedes encontrar ayuda si te atascas.

En resumen, Java con Spring es una elección muy buena para hacer una REST API porque es fácil de usar, está hecho para APIs RESTful, te ayuda a mantener tu código organizado, tiene herramientas de documentación y pruebas, ofrece funciones de seguridad y hay mucha ayuda disponible en la comunidad. Es una combinación excelente para desarrollar APIs REST.

PostgreSQL

PostgreSQL es una opción excelente como base de datos relacional por varias razones:

1. **Es gratuita y de código abierto**: Lo mejor de PostgreSQL es que es gratis y su código está abierto para que todos lo usen y contribuyan. Esto significa que no tienes que preocuparte por pagar por su uso, lo que lo hace genial para proyectos pequeños o startups con presupuestos limitados.
2. **Gran comunidad y abundante documentación**: Hay una comunidad activa de desarrolladores que utilizan PostgreSQL, por lo que siempre encontrarás ayuda en línea, tutoriales y foros. Además, su documentación es extensa y fácil de entender, lo que facilita aprender a usarlo y resolver problemas.
3. **Compatibilidad con estándares de SQL**: PostgreSQL sigue muy de cerca los estándares de SQL, lo que significa que si aprendes a usarlo, tus habilidades serán fácilmente transferibles a otras bases de datos relacionales. Esto es genial para tu crecimiento profesional.
4. **Escalabilidad y rendimiento**: A medida que tu aplicación crece, PostgreSQL puede manejar grandes cantidades de datos sin perder rendimiento. Puede ser escalado de manera efectiva, permitiéndote adaptarte al crecimiento de tu proyecto sin grandes problemas.
5. **Características avanzadas y extensibilidad**: PostgreSQL ofrece muchas características avanzadas. Desde la gestión de datos espaciales hasta la compatibilidad con JSON y capacidades de indexación avanzada, brinda muchas herramientas poderosas para el manejo de datos.
6. **Seguridad robusta**: PostgreSQL se toma en serio la seguridad. Ofrece mecanismos sólidos de autenticación y autorización, lo que te permite controlar quién tiene acceso a tus datos y cómo se accede a ellos.

En resumen, PostgreSQL es una base de datos relacional que, siendo gratuita, cuenta con una gran comunidad, es compatible con estándares de SQL, escala bien, tiene muchas características avanzadas y proporciona una seguridad sólida.

MongoDB

Para la parte de los pedidos, hemos utilizado MongoDB por varias razones:

1. **Esquema flexible**: A diferencia de las bases de datos relacionales, en MongoDB no necesitas definir una estructura de datos rígida desde el principio. Puedes guardar datos sin una estructura fija, lo que te da mucha flexibilidad para adaptarte a cambios en tus aplicaciones a medida que evolucionan.
2. **Modelo de datos basado en documentos**: En MongoDB, los datos se almacenan en documentos similares a JSON. Esto facilita mucho el manejo de datos para los desarrolladores, ya que coincide con la estructura de datos de muchas aplicaciones web modernas.
3. **Escalabilidad horizontal sencilla**: MongoDB está diseñado para escalar horizontalmente fácilmente. Puedes distribuir tus datos en varios servidores, lo que es genial para manejar grandes cantidades de información y un mayor número de usuarios sin sacrificar el rendimiento.
4. **Rendimiento y velocidad**: MongoDB está optimizado para un rápido acceso a datos. Su modelo de almacenamiento de documentos y la capacidad de indexar esos documentos hacen que las consultas sean más rápidas, lo que es importante para aplicaciones web que necesitan respuestas rápidas.
5. **Gran comunidad y documentación amigable**: Siempre hay mucha gente hablando de MongoDB y compartiendo conocimientos en línea. La documentación oficial es bastante fácil de entender y hay un montón de tutoriales disponibles para aprender a usarlo rápidamente.
6. **Soporte para datos no estructurados y semi-estructurados**: Si tus datos no tienen un formato fijo o si estás trabajando con datos semiestructurados, MongoDB es una excelente opción. Puedes almacenar estos datos sin tener que ajustarlos a una tabla predefinida.

En resumen, MongoDB es una base de datos NoSQL que destaca por su flexibilidad de esquema, velocidad, escalabilidad horizontal y facilidad de uso para desarrolladores.

Estructura de la API REST para múltiples recursos y bases de datos

El diseño de una API REST para gestionar diversos recursos almacenados en diferentes bases de datos implica considerar varios aspectos fundamentales para su correcta implementación y funcionalidad. Aquí se detalla un esquema general para estructurar esta API, tomando en cuenta los recursos de Categorías, Clientes, Empleados, Pedidos, Productos, Proveedores y Usuarios, almacenados en bases de datos como PostgreSQL y MongoDB.

Diseño de la API

1. **Definición de Rutas y Recursos:**
    a. Se establecen rutas claras y significativas para cada recurso, por ejemplo:
        i. /categories
        ii. /clients
        iii. /employees
        iv. /orders
        v. /products
        vi. /suppliers
        vii. /users
2. **Métodos CRUD y Controladores:**
    a. **Se implementan métodos HTTP estándar (GET, POST, PUT, PATCH, DELETE) para realizar operaciones CRUD en cada recurso.**
    b. Se desarrollan controladores o endpoints para cada recurso, manejando las solicitudes HTTP correspondientes y la lógica de negocio asociada.
3. **Conexión con Bases de Datos:**
    a. Se establece la interacción con las bases de datos pertinentes:
        i. PostgreSQL para Categorías, Clientes, Empleados, Productos, Proveedores y Usuarios.
        ii. MongoDB para almacenar Pedidos.
4. **Autenticación y Seguridad:**
    a. Se implementan mecanismos de autenticación y autorización para garantizar la seguridad de la API y restringir el acceso a los recursos según roles de usuario.
5. **Documentación y Pruebas:**
    a. **Se genera documentación detallada sobre el funcionamiento de la API, describiendo las rutas, métodos, parámetros y estructura de respuestas.**
    b. Se realizan pruebas exhaustivas para verificar el correcto funcionamiento de la API en diferentes escenarios.

Este enfoque proporciona una estructura flexible y escalable para gestionar múltiples recursos a través de una API REST, manteniendo la coherencia en el acceso y manipulación de datos a través de diferentes plataformas de almacenamiento.