TIENDA NEST

Jaime Medina

Eva Gómez

Jorge Benavente

David Jaraba

ÍNDICE

Introducción

En el dinámico mundo de la tecnología, la gestión eficiente es clave para el éxito de cualquier negocio, especialmente en el sector de la informática. La capacidad de manejar de manera efectiva inventarios, proveedores, clientes y empleados es fundamental para garantizar operaciones fluidas y satisfacer las demandas del mercado.

Con este fin, hemos desarrollado una aplicación innovadora utilizando Nest, un framework de Node.js altamente escalable y robusto. Esta aplicación está diseñada específicamente para abordar las necesidades de gestión de una tienda de informática, ofreciendo una solución óptima que simplifica la administración de productos, proveedores, clientes y empleados.

En esta presentación, os mostraré las funcionalidades clave de nuestra aplicación y cómo Nest nos ha permitido construir una plataforma confiable y flexible para optimizar las operaciones comerciales. Desde la gestión de inventario hasta la administración de clientes y personal, nuestra aplicación ofrece una experiencia completa que impulsa el crecimiento y la eficiencia de cualquier tienda de informática.

Funcionalidades principales

Gestión de Productos

Para la gestión de productos en nuestra aplicación, utilizamos una entidad **Product** que nos permite almacenar información detallada sobre cada artículo en nuestro inventario. Aquí está cómo se gestionan los productos y algunas de las operaciones disponibles:

Descripción de cómo se gestionan los productos y sus categorías asociadas:

- Cada producto está representado por una instancia de la clase **Product**, que contiene atributos como nombre, peso, precio, imagen, stock y descripción.
- Para asociar un producto con una categoría, utilizamos una relación de muchos a uno (**ManyToOne**) con la entidad **Category**. Esto nos permite organizar nuestros productos en diferentes categorías para una mejor organización y navegación.
- Del mismo modo, asociamos cada producto con un proveedor utilizando una relación de muchos a uno (**ManyToOne**) con la entidad **Supplier**. Esto nos ayuda a rastrear qué proveedor suministra cada artículo en nuestro inventario.

Algunas de las operaciones que podemos realizar en nuestro programa por lo tanto serían crear, actualizar o borrar un producto, verificando anteriormente que las entidades relacionadas, como son **Supplier** y **Category,** existan en la base de datos. Además, podemos asignarle una imagen al producto que no sea la que viene por defecto, si ya tenía una imagen anteriormente que no fuese la de por defecto, esta se eliminaría de la carpeta donde están almacenadas y se sustituiría por la nueva, para ello tendremos una gestión de almacenamiento.

Gestión de Almacenamiento

El servicio de almacenamiento se encarga de manejar los archivos de imágenes asociados a los productos, proporciona funciones para buscar y eliminar archivos de imágenes en el sistema de archivos. Tendremos unas validaciones de imagen como por ejemplo el tamaño máximo, o las extensiones permitidas.

Gestión de Proveedores

En esta entidad tendremos información relevante sobre cada proveedor, como su nombre, contacto, dirección, fecha de contratación, categoría y productos asociados.

Tendrá un id generado con UUID ya que nuestro interés en esta entidad viene en los productos asociados, con los que tendrá una relación uno a muchos, esto significa que cada proveedor puede suministrar múltiples productos de la tienda.

Esta relación permite un seguimiento efectivo de los proveedores y los productos que suministran, lo que garantiza una gestión eficiente de los recursos.

Gestión de Clientes

Los clientes tienen la capacidad de acceder a la lista de productos Para esto, los clientes deben autenticarse en la aplicación utilizando JSON Web Tokens (JWT). Una vez autenticados, se verifica el rol del usuario para asegurar que tengan los permisos adecuados para acceder a la funcionalidad de visualización de productos. Si su rol de usuario es 'client', se les permite acceder a la ruta correspondiente que muestra la lista de productos disponibles.

Para gestionarlo, tendremos un controlador que se encarga de manejar las solicitudes de los clientes y asegurarse de que solo tengan acceso a las funcionalidades permitidas según su rol.

El sistema de autenticación basado en JWT y la verificación de roles garantizan que solo los usuarios autorizados puedan acceder a la funcionalidad de ver los productos disponibles, mejorando así la seguridad y la protección de los datos de la aplicación.

Gestión de Empleados

Los empleados tienen acceso completo a todas las funcionalidades de la aplicación, lo que les permite realizar operaciones CRUD en productos, clientes y otros recursos de la tienda de informática. Para acceder a estas funcionalidades, los empleados deben autenticarse en la aplicación utilizando también JSON Web Tokens (JWT). Una vez autenticados, se les otorga un token JWT que contiene su información de identificación y roles. Así podremos verificar su identidad y sus permisos en cada solicitud que realice a la aplicación. En el caso de no estar autorizado se deniega el acceso y se devuelve un error de autorización.

Gestión de Categorías

Cada categoría está representada por una instancia de la clase **Category**, que contiene atributos como nombre, fecha de creación, fecha de actualización y estado de activación. El nombre de la categoría se utiliza para identificarla de manera única y es obligatorio. El estado de activación indica si la categoría está activa o no. Por defecto, las categorías se crean como activas, pero este estado puede modificarse según sea necesario.

Cada categoría puede estar asociada con múltiples productos a través de una relación de uno a muchos. Esto nos permite asignar productos a categorías específicas para una mejor organización.

Asimismo, una categoría puede estar asociada con múltiples proveedores a través de otra relación de uno a muchos. Esto puede ser útil para rastrear los proveedores que suministran productos dentro de una categoría particular.

Gestión de Pedidos

Cada pedido contiene atributos como el id, el id del usuario que realizó el pedido, el cliente asociado, las líneas de pedido, el total de elementos, el total del pedido, la fecha de creación, la fecha de actualización y el estado de eliminación.

El cliente asociado al pedido contiene información como el nombre, el correo electrónico, el teléfono y la dirección de envío del cliente.

La dirección contiene atributos como la calle, el número, la ciudad, la provincia y el código postal.

Cada línea de pedido contiene detalles como la cantidad, el identificador único del producto, el precio unitario del producto y el total de esa línea de pedido.

El total de elementos y el total del pedido se calculan en base a las líneas de pedido incluidas en el pedido.

Cada pedido está asociado a un cliente a través de la entidad **Client**, que proporciona información detallada sobre el cliente que realizó el pedido. Además, cada pedido puede contener múltiples líneas de pedido.

PRECIOS

ESPECIFICACIONES TÉCNICAS PARA LAS BASES DE DATOS

POSTGRESQL

Para alojar nuestras bases de datos, hemos decidido utilizar las instancias RDS para PostgreSQL disponibles en AWS. En particular, para el entorno de producción, recomendamos comenzar con la instancia **db.t4g.medium**. Esta instancia proporciona una base de datos con las siguientes características:

- CPU virtual: 2
- Memoria RAM: 4GB
- 30 GB de almacenamiento SSD

En caso de que nuestras necesidades de base de datos aumenten, tenemos la flexibilidad de mejorar la instancia fácilmente. Al igual que con las instancias EC2, podemos optar por una instancia de mayor capacidad en momentos específicos de manera simple y directa.

También existen otras opciones dentro de AWS como T3, T2, M7g... pero hemos elegido la T4g ya que es la que más nos conviene para nuestra aplicación por precio, rendimiento y escalabilidad.

Dentro de T4g existen distintas opciones a las que podemos optar en caso de aumento de demanda.

Modelo	CPU virtual	Memoria (GiB)	Ancho de banda de EBS con ráfagas (Mbps)	Rendimiento de red (Gbps)
db.t4g.micro	2	1	Hasta 2085	Hasta 5
db.t4g.small	2	2	Hasta 2085	Hasta 5
db.t4g.medium	2	4	Hasta 2085	Hasta 5
db.t4g.large	2	8	Hasta 2780	Hasta 5
db.t4g.xlarge	4	16	Hasta 2780	Hasta 5
db.t4g.2xlarge	8	32	Hasta 2780	Hasta 5

Si deseas consultar las distintas opciones disponibles respecto a la elección del sistema de alojamiento de base de datos acceda al siguiente enlace: AMAZON RDS

MONGODB

Optaremos por utilizar las instancias de Amazon DocumentDB, que son compatibles con MongoDB. Estas instancias operan bajo un modelo de pago por uso, lo que significa que no hay costos iniciales; en su lugar, pagamos según el almacenamiento utilizado, las operaciones de entrada/salida y el tráfico de datos. En esta situación, al igual que con RDS, recomendaremos la instancia **db.t4g.medium**, que ofrece las mismas características:

- CPU virtual: 2
- Memoria RAM: 4GB

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

Instancias optimizadas para memoria, generación actual	Memoria	vCPU	Amazon DocumentDB Estándar (precio por hora)	Amazon DocumentDB Optimizado para E/S (precio por hora)
db.r6g.large	16 GiB	2	0,263 USD	0,290 USD
db.r6g.xlarge	32 GiB	4	0,526 USD	0,579 USD
db.r6g.2xlarge	64 GiB	8	1,053 USD	1,158 USD
db.r6g.4xlarge	128 GiB	16	2,105 USD	2,316 USD
db.r6g.8xlarge	256 GiB	32	4,210 USD	4,631 USD
db.r6g.12xlarge	384 GiB	48	6,316 USD	6,947 USD
db.r6g.16xlarge	512 GiB	64	8,419 USD	9,261 USD

Si deseas consultar las distintas opciones disponibles respecto a la elección del sistema de alojamiento de base de datos acceda al siguiente enlace: AMAZON DocumentDB

PROPUESTA RECOMENDADA PARA LA APLICACIÓN

La idea es utilizar los servidores de AWS para alojar nuestra aplicación en producción. Optamos por AWS porque ofrece una variedad de servidores y facilita la escalabilidad horizontal, lo que significa que podemos aumentar la capacidad de nuestra aplicación de manera flexible para adaptarnos a cambios en la demanda. Además, confiamos en la fiabilidad y el rendimiento de la infraestructura de AWS.

Para comenzar, para aprovechar al máximo la infraestructura de AWS, planeamos implementar nuestra aplicación en una instancia EC2. Estas instancias proporcionan máquinas virtuales donde podemos ejecutar nuestras aplicaciones. Para nuestro caso específico, hemos elegido una instancia **t4g.medium**, que tiene las siguientes características:

- CPU virtual: 2
- Memoria RAM: 4GB
- Ancho de banda: hasta 5Gb por segundo

En situaciones en las que la demanda de nuestra aplicación aumente, tenemos la flexibilidad de mejorar la instancia actual a una de mayor capacidad o aumentar el número de instancias que estamos utilizando. A continuación, mencionamos las instancias a las que podríamos escalar partiendo de instancia **t4g.medium:**

Tamaño de instancia	CPU virtual	Memoria (GiB)	Rendimiento base/CPU virtual	Créditos de CPU obtenidos/hora	Ancho de banda de red con ráfagas (Gbps)	Ancho de banda de EBS con ráfagas (Mbps)
t4g.nano	2	0,5	5 %	6	Hasta 5	hasta 2085
t4g.micro	2	1	10 %	12	Hasta 5	hasta 2085
t4g.small	2	2	20 %	24	Hasta 5	hasta 2085
t4g.medium	2	4	20 %	24	Hasta 5	hasta 2085
t4g.large	2	8	30 %	36	Hasta 5	hasta 2780
t4g.xlarge	4	16	40 %	96	Hasta 5	hasta 2780
t4g.2xlarge	8	32	40 %	192	Hasta 5	hasta 2780

Si desea consultar dicha información diríjase al siguiente enlace: AMAZON EC2

Ventajas de la propuesta recomendada

La instancia EC2 t4g.medium se destaca como la mejor opción para nuestra aplicación en este momento por varias razones. Ofrece un equilibrio óptimo entre rendimiento y costo, lo que nos permite maximizar la eficiencia de nuestros recursos.

Además, su capacidad para manejar la carga actual de la aplicación garantiza un funcionamiento fluido y sin interrupciones. La flexibilidad inherente de esta instancia nos permite escalar nuestros recursos según sea necesario, lo que nos brinda la tranquilidad de adaptarnos fácilmente a cambios en la demanda sin comprometer el rendimiento.

Consideraciones de Seguridad y Confiabilidad

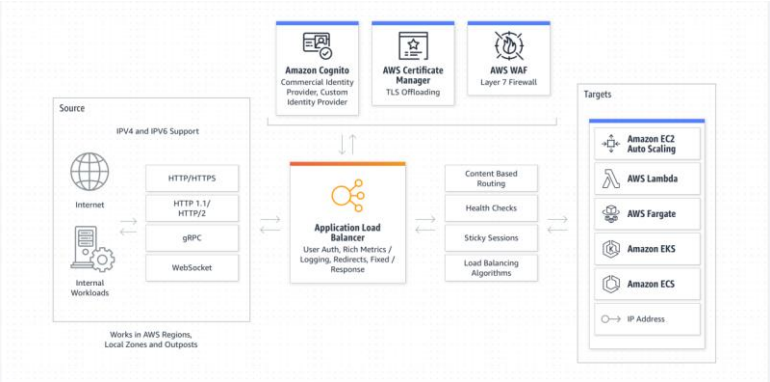
La elección de AWS como nuestra plataforma de alojamiento no solo se basa en consideraciones de rendimiento y escalabilidad, sino también en su destacada reputación en seguridad y confiabilidad.

AWS ofrece una amplia gama de herramientas y servicios diseñados para proteger nuestros datos y garantizar la integridad de nuestra aplicación. Esto incluye características como cortafuegos, controles de acceso, monitoreo en tiempo real y protección contra ataques DDoS. Además, la infraestructura de AWS está respaldada por una sólida red de centros de datos y una infraestructura global, lo que garantiza una alta disponibilidad y redundancia para nuestra aplicación.

SERVICIOS RECOMENDADOS

ELASTIC LOAD BALANCING

Si nuestra aplicación requiere escalar, es altamente aconsejable utilizar el servicio de Elastic Load Balancing. Elastic Load Balancing (ELB) es un servicio de AWS diseñado para distribuir automáticamente el tráfico entrante de las aplicaciones en múltiples instancias de Amazon EC2, contenedores de Amazon ECS o direcciones IP de clientes en varias zonas de disponibilidad.



Estas son algunas de sus características:

- **Alta disponibilidad**: ELB garantiza la fiabilidad al redirigir el tráfico solo a instancias saludables.
- **Escalabilidad automática**: Ajusta el número de instancias EC2 según la carga de la aplicación.
- **Balanceo de carga inteligente**: Distribuye uniformemente el tráfico entre las instancias para evitar sobrecargas.
- **Integración con servicios de AWS**: Se integra con Route 53, Certificate Manager y CloudWatch.

Página oficial: AMAZON ELB

SERVICIO DE ALMACENAMIENTO SIMPLE (S3)

El Servicio de Almacenamiento Simple (S3) de Amazon sería una excelente opción para almacenar los archivos de imágenes que nuestra aplicación necesitará subir. Este servicio nos proporciona almacenamiento por gigabyte (GB) a un costo principal de 0.02 USD por GB al mes, además de facturar la cantidad de solicitudes GET y PUT que realicemos.

Estimación de costes

Se estima un gasto de:

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

* Los costos mencionados son una estimación para un uso moderado de la aplicación. Amazon factura según el consumo real, por lo que los costos podrían ser más altos si la aplicación tiene una demanda mayor.

* 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.

Costes de Producción

El sueldo mensual de cada uno de los trabajadores, habiendo durado el trabajo un mes, será de 1500€ aproximadamente.

En estos se incluyen las horas semanales invertidas en el proyecto, el uso de distintas herramientas de elaboración de código y la elaboración de tests unitarios y de integración.

También se incluyen las futuras tareas de mantenimiento y actualización que se llevarán a cabo a medida que vayan surgiendo distintos problemas.

Clowns Informatics (Empresa)
C/ Paseo Grande, 1 (Dirección)
NIF: A12345678
Teléfono: 612 456 789
Mail: info@clownsinformatics.com

FACTURA Nº: 764563

12 febrero 2023

Datos cliente

Cliente:
Dirección:
NIF:
Mail:
Teléfono

Descripción / Producto	Cantidad	Base	IVA	Total
Hora de programación	200	25	5,25	6050,00 €
Mantenimiento	1	100	21	121,00 €
Servidores	1	296,33	62,23	358,56 €

BASE		5396,33 €
IVA	21%	1133,22 €
DESCUENTO	NO	0,00 €
Total		**6529,55 €**

El pago se realizará en un plazo de tres meses desde la emisión de esta factura, se realizará mediante transferencia bancaria.

Elección de tecnologías

NodeJS con Nest.js y TypeScript

Hemos optado por utilizar Node.js con el framework Nest.js y TypeScript en nuestro proyecto por las siguientes razones:

Seguridad y robustez: TypeScript es un superconjunto tipado de JavaScript que proporciona un sistema de tipos estático. Esto nos permite detectar errores en tiempo de compilación y mejorar la robustez y seguridad de nuestro código, reduciendo la probabilidad de errores en tiempo de ejecución.

Mejora la calidad del código: TypeScript fomenta prácticas de desarrollo más sólidas al permitirnos definir tipos de datos para nuestras variables, parámetros de función y valores de retorno. Esto hace que nuestro código sea más legible, mantenible y fácil de entender, especialmente en proyectos grandes y complejos.

Productividad mejorada: TypeScript ofrece características modernas de programación, como la inferencia de tipos, la finalización automática y el refactorizado seguro. Estas características aumentan la productividad del desarrollo al reducir el tiempo necesario para escribir y depurar código, y al proporcionar herramientas poderosas para la refactorización y el mantenimiento del código.

Compatibilidad con JavaScript: TypeScript es un superset de JavaScript, lo que significa que todo el código JavaScript existente es válido en TypeScript. Esto nos permite utilizar las bibliotecas y herramientas de JavaScript existentes en nuestro proyecto, mientras disfrutamos de las ventajas adicionales que ofrece TypeScript.

Soporte para características avanzadas de ECMAScript: TypeScript permite utilizar las últimas características de ECMAScript, como los decoradores, las clases, las funciones de flecha y los módulos, incluso en entornos donde no están completamente soportados. Esto nos permite escribir código más moderno y limpio, manteniendo la compatibilidad con versiones anteriores de JavaScript.

En resumen, elegimos Node.js con el framework Nest.js y TypeScript para nuestro proyecto debido a los beneficios que ofrece TypeScript en términos de seguridad, calidad del código, productividad y compatibilidad con JavaScript, además de aprovechar las características organizadas y modernas proporcionadas por Nest.js. Esta combinación nos permite desarrollar aplicaciones web de manera más segura, eficiente y mantenible.

POSTGRESQL

Hemos seleccionado PostgreSQL para gestionar los datos de empleados y clientes en nuestra aplicación debido a las siguientes razones:

Libre y de código abierto: PostgreSQL es una opción gratuita y de código abierto, lo que significa que no incurre en costos de licencia. Esta característica lo convierte en una elección atractiva para proyectos con recursos limitados, como startups y empresas emergentes.

Amplia comunidad y abundante documentación: PostgreSQL cuenta con una comunidad activa de desarrolladores y una extensa documentación. Esto garantiza que siempre haya recursos disponibles en línea, como guías, debates y tutoriales, lo que facilita tanto el aprendizaje como la resolución de problemas.

Adherencia a estándares de SQL: PostgreSQL sigue de cerca los estándares de SQL, lo que hace que aprender a utilizarlo sea altamente transferible a otras bases de datos relacionales. Esta consistencia en la sintaxis y la funcionalidad es beneficiosa para el desarrollo profesional a largo plazo.

Escalabilidad y rendimiento: PostgreSQL puede manejar eficientemente grandes volúmenes de datos a medida que nuestra aplicación crece. Su capacidad de escalar horizontal y verticalmente nos permite adaptarnos al crecimiento del proyecto sin sacrificar el rendimiento.

Funcionalidades avanzadas y extensibilidad: PostgreSQL ofrece una amplia gama de funcionalidades avanzadas, desde la gestión de datos espaciales hasta la compatibilidad con JSON y capacidades de indexación avanzada. Esto nos proporciona herramientas potentes para el manejo de datos y la optimización de consultas.

Seguridad sólida: La seguridad es una prioridad en PostgreSQL, ofreciendo mecanismos robustos de autenticación y autorización. Esto nos permite controlar con precisión quién accede a los datos y cómo lo hacen, protegiendo así la información confidencial de nuestros empleados y clientes.

En resumen, PostgreSQL se destaca como una opción de base de datos relacional por su gratuidad, su amplia comunidad y documentación, su adherencia a estándares de SQL, su escalabilidad, sus funcionalidades avanzadas y su sólida seguridad, convirtiéndolo en la elección ideal para almacenar los datos críticos de empleados y clientes en nuestra aplicación.

MongoDB

Hemos optado por utilizar MongoDB para gestionar los pedidos en nuestra aplicación por las siguientes razones:

Flexibilidad de esquema: MongoDB nos ofrece la libertad de guardar datos sin una estructura predefinida. Esto nos permite adaptarnos con facilidad a los cambios en nuestras aplicaciones, evitando la rigidez de las bases de datos relacionales.

Modelo de datos documental: Almacenar los datos en documentos similares a JSON simplifica enormemente su manipulación para los desarrolladores. Este modelo se alinea naturalmente con la estructura de datos común en muchas aplicaciones web modernas.

Escalabilidad horizontal sin complicaciones: MongoDB está diseñado para escalar horizontalmente sencillamente. Esta capacidad nos permite distribuir los datos en múltiples servidores, lo que resulta ideal para manejar grandes volúmenes de información y una mayor cantidad de usuarios sin sacrificar el rendimiento.

Eficiencia y velocidad: La optimización de MongoDB para un acceso rápido a los datos, junto con su capacidad de indexación, agiliza significativamente las consultas. Esto es fundamental para aplicaciones web que requieren respuestas rápidas y eficientes.

Amplia comunidad y recursos de aprendizaje: La activa comunidad de MongoDB y su documentación amigable nos proporcionan una gran cantidad de recursos en línea, como tutoriales y debates. Esto facilita tanto el aprendizaje como la resolución de problemas durante el desarrollo y la gestión de la aplicación.

Compatibilidad con datos no estructurados o semi-estructurados: MongoDB es una excelente elección cuando trabajamos con datos que no tienen un formato fijo o son semi-estructurados. Nos permite almacenar estos datos sin necesidad de ajustarlos a una estructura predefinida, lo que simplifica el proceso de almacenamiento y recuperación de la información.

En resumen, MongoDB destaca como una base de datos NoSQL que ofrece flexibilidad de esquema, eficiencia, escalabilidad horizontal y facilidad de uso para los desarrolladores, convirtiéndola en la elección ideal para gestionar los pedidos en nuestra aplicación.

En nuestro caso, usamos mongo para pedidos ya que es algo que va a tener una estructura flexible y nos facilita en el momento que queramos poder cambiar las propiedades de los pedidos o modificar algo en su estructura, a diferencia del resto de endpoints (productos, empleados, proveedores, categorías o clientes) que si van a tener una estructura más solida y definida y que además van a tener definidas relaciones claras entre ellos.

DIAGRAMA ENTIDAD RELACIÓN