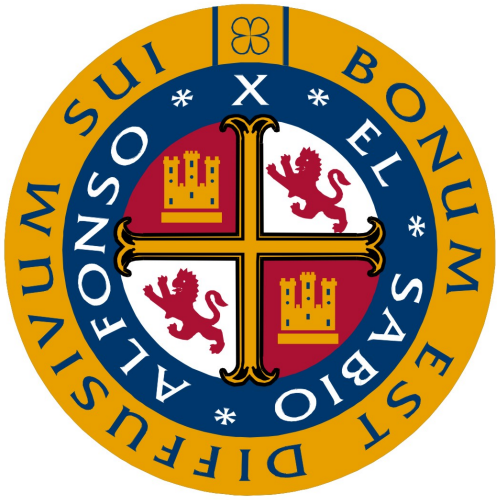

UNIVERSIDAD
ALFONSO X EL SABIO

Implementación de Programación Concurrente en Bases de Datos NoSQL

Concurrencia con MongoDB en React.js

Por

Luis Miguel Urbez

Np: 139314

Programacion Concurrente

Ingeniería Informática

ENUNCIADO DEL EJERCICIO

Objetivos de la práctica:

En esta práctica, nuestro objetivo será obtener una comprensión práctica de cómo aplicar los principios de la programación concurrente en un entorno de base de datos NoSQL. Implementaremos diferentes estrategias de programación concurrente y evaluaremos su rendimiento al manejar grandes volúmenes de datos.

En la era actual de Big Data, las organizaciones generan y consumen cantidades enormes de información diariamente. Con tanta información disponible, las bases de datos tradicionales SQL pueden resultar ineficientes para manejar volúmenes de datos tan masivos y estructuras de datos complejas. Por lo tanto, las bases de datos NoSQL se han convertido en una elección popular debido a su escalabilidad y flexibilidad. Sin embargo, la eficiencia de estas bases de datos puede mejorarse aún más con el uso de la programación concurrente.

Detalles de la tarea:

Para la base de datos NoSQL, puedes optar por utilizar MongoDB, que es una base de datos orientada a documentos, o Apache Cassandra, que es una base de datos orientada a columnas. Ambas opciones son open-source y tienen amplia documentación disponible.

En cuanto a los datos, puedes utilizar conjuntos de datos públicos disponibles en recursos como Kaggle, Google Dataset Search o el Portal de Datos Abiertos del Gobierno. Sin embargo, elige un conjunto de datos que sea lo suficientemente grande para que sea relevante el uso de programación concurrente.

Podrías optar por un conjunto de datos relacionados con transacciones de comercio electrónico, tweets de Twitter, datos meteorológicos históricos, registros de sensores de IoT, etc. Recuerda, cuanto más grande y complejo sea el conjunto de datos, más interesantes serán los desafíos al implementar la programación concurrente.

Para la implementación de la programación concurrente, puedes usar herramientas y bibliotecas como ExecutorService y Future en Java, o ThreadPoolExecutor en Python. Estas herramientas te permiten gestionar la creación de hilos, su ejecución y la recolección de resultados de manera eficiente.

En cuanto a las pruebas de rendimiento, existen varias herramientas que puedes usar para monitorear y registrar el rendimiento de tu aplicación, como JMeter, Gatling o incluso las herramientas de monitoreo integradas en tu IDE.

Detalles de la tarea:

Configuración inicial: Inicia configurando una base de datos NoSQL (puedes elegir entre una base de datos clave-valor, orientada a documentos, orientada a columnas o una base de datos de grafos).

Modelado de datos: Modela un conjunto de datos para tu base de datos NoSQL elegida. Los datos deben ser lo suficientemente complejos para requerir el uso de programación concurrente (por ejemplo, un número significativo de registros o una estructura de datos compleja).

Implementación de programación concurrente: Implementa la programación concurrente para realizar operaciones en tu base de datos. Esto puede implicar el uso de múltiples hilos o procesos para manejar las operaciones de lectura y escritura, la implementación de bloqueos y otras estrategias de control de concurrencia, o el uso de bibliotecas de programación concurrente de alto nivel.

Pruebas y evaluación de rendimiento: Realiza pruebas de rendimiento en tu implementación. ¿Cómo se compara el rendimiento de tu implementación con y sin programación concurrente? Considera el tiempo de ejecución, el uso de recursos y cualquier otro factor de rendimiento relevante.

Informe de la práctica: Escribe un informe que documente tu enfoque, los desafíos que encontraste y cómo los resolviste, los resultados de tus pruebas de rendimiento y cualquier hallazgo o conclusión interesante que hayas obtenido de la práctica.

Consideraciones finales:

Este proyecto está diseñado para ser un desafío. Te animamos a que experimentes, investigues y explores diferentes enfoques y soluciones. ¡Buena suerte!

Rúbrica:

Entendimiento y modelado del problema (25

Entiende el problema y el contexto de la práctica. Diseña e implementa correctamente un modelo de datos en MongoDB que se adapta al problema. Implementación de la programación concurrente (25

Implementa correctamente los conceptos de programación concurrente en la práctica. Usa adecuadamente los recursos de la computadora para optimizar la eficiencia y rendimiento. Maneja adecuadamente los errores y las excepciones que puedan surgir durante la ejecución concurrente. Interacción con la base de datos MongoDB (25

Establece correctamente la conexión con la base de datos MongoDB. Realiza correctamente operaciones de lectura y escritura en la base de datos. Maneja adecuadamente los errores y las excepciones que puedan surgir durante la interacción con la base de datos. Análisis del rendimiento (15

Realiza un análisis cuantitativo y cualitativo del rendimiento de la práctica. Compara y discute los resultados obtenidos con y sin la programación concurrente. Calidad del código (10

Escribe código limpio, legible y bien comentado. Sigue las mejores prácticas de codificación y estilo. Por favor, ten en cuenta que esta es una rúbrica sugerida y puede adaptarse según las necesidades y requisitos específicos de tu curso o proyecto.

RESUMEN DEL PROYECTO

En este proyecto se está buscando la implementación de la concurrencia aprendida en clase usando una base de datos no relacional como MongoDB. Aunque había más bases de datos disponibles, desde mi punto de vista he considerado MongoDB más eficiente por su sencillez y por ser una de las mas famosas en este ámbito de la programación. Me ha concedido una amplia variedad de informacion de otros programadores dado que cuenta con mucha documentación disponible de la base de datos.

Para el uso de la base de datos me he apoyado en la aplicacion de escritorio que la propia pagina de mongoDB Atlass (desde donde se maneja todo lo relacionado con los clusters y als baeses de datos) te recomienda, siendo esta MongoDb Compass. Esta aplicación me ha porpocionado una vista más gráfica y completa de mi base de datos. También me ha ofrecido muchas herramientas para trabajar mas comodo con mis datos.

El proyecto ha sido implementado en TypeScript con sus diferentes variables, como tsx y ts. al igual que con JavaScript el cual he usado jsx y jx. Para el proyecto me he basado la pagina web de alquiler de casas y apartamentos Airbnb. De la cual he usado su base de datos para implementarlos en mi cluster de MongoDB. Estos datos en la pagina se ven afectados por las politicas de Airbnb, la cual no permite usar fotos ni nombres reales de las casas, pero no ha sido un impedimento para el desarrollo del proyecto.

He usado los IDE IntelliJ y WebStorm, proporcionados por Jetbrains, los cuales han sido muy utiles para la implementación de mi proyecto, ya que oferecen una

ámplia variedad y gama de pluggings y herramientas para el desarrollo de este tipo de proyectos.

Todo el proyecto y sus links se encuentran en el repositorio de github publico (https://github.com/LuisMi01/airbnb-mongodb$_{LuisMiguelUrbez)}, aligualquetodaslasexplicacionesenformadecodigocomentadoycommitsqu$

Por ultimo, antes de entrar en detalles del desarrollo del proyecto, la pagina web que ha administrado el deployment de este proyecto ha sido vercel, la cual me ha proporcionado usar la pagina de forma dinamica y salir los proyectos estaticos que usaban localhost.

Los siguientes links son los respectivos para la central de deployments del proyecto y la pagina principal.

1.- https://vercel.com/luis-miguel-urbezs-projects/airbnb-mongodb-luis-miguel-urbez
2.- https://airbnb-mongodb-luis-miguel-urbez.vercel.app/

PARTE GRÁFICA

Para la parte grafica del proyecto he usado el lenguaje TypeScript, en concreto TSX. El cual es una extensión de TypeScript que combina JavaScript con la sintaxis de XML para el desarrollo de interfaces en React. Me ha permitido escribir componentes React con tipado estático y estructura declarativa, facilitandome el desarrollo de la aplicacion web. Este lenguaje, nuevo para mi, me ha parecido muy util para la creacion de las webs. Aunque al principio se me hizo un gran reto el uso de un lenguaje tan nuevo para mi, es facil hacerse a su estructura y sintaxis.

El lenguaje de TSX me a parecido un lenguaje muy completo, con infinidad de posibilidades gracias a su extensa documentacion la cual he usado para aprender parte dle lenguaje y por todas sus librerias. EStas ultimas han sido usadas de forma extensa y constante en el proyecto. Como librerias de imagenes como 'react-icons', la cual me ha proporcionado todos los iconos que se ven en la aplicacion. Otras librerias como 'react' la cual es la mas extensa del FrameWork que he usado, junto con Next.js me ha proporcionado metodos para aplicar logica de concurrencia a mi proyecto como Lazy o Suspense.

Ya que el lenguaje de TSX y TS ha sido completamente nuevo para mi he tenido que aprenderlo de fuentes externas esta ultima semana. La principal fuente para aprender la logica y la sintaxis del mismo ha sido la documentacion que se encuentra disponible en la pagina de React.js (https://es.react.dev/learn). Para diseñar el proyecto me he basdo en el siguiente tutorial que se encuentra en Youtube de forma publica: https://www.youtube.com/watch?v=c$_b_isI4vgt = 9743s.$

Este tutorial me ha permitido seguir una explicacion detallada de la cracion de la parte grafica y de mis primeros pasos a la hora d ela implementacion de clases y

componenetes en React.js y en Next.js.

La implementacion de librerias de diseño y la creacion de mis propios objetos los cuales he reutilizado en muchos lugares, (como las "tarjetas" de las casas reutilizados en la pagina principal, favoritos y propiedades). Esto me ha permitido agilizar el proyecto y no tener que realizar mucho codigo "basura" el cual esta rehaciendose constantemente. Por la parte grafica tambien destacar las oprtunidades que me han brindado algunas paginas externas como Boostrap, la cual me dio la idea de crear la opcion de filtrar por paises y dormitorios en la aplicacion.

PARTE LÓGICA

Dentro de la parte logica de al aplicacion, lo primero fue darle las opciones basicas de funcionamiento a la aplicacion. Como moverse entre oaginas y poder añadir y eliminar objetos. esta parte ha sido sin ninguna duda la mas complicada para mi, ya que la mezcla de creación de componentes gráficos junto con su propia logica la cual tiene q usar las logicas de más componentes a parte, se me ha dificultado mucho.

Emepezé por la implementacion de la logica más "basica" de mi proyecta, la creacion de los usuarios para iniciar sesion y registrarse. Con la creacion de esta logica he sido capaz de realizar mis primeras llamadas POST y GET a mi base de datos en mongo, lo que ha hecho que luego se me hiciese mucho mas facil para el resto de las llamadas ya que lo tenia ya implementado de antes. En esta parte no he añadido ningun tipo de concurrencia ya que al ser funciones mas sencillas no hacia falta ya que eran bastate rapidas de por si.

A estas funciones lo unico que se le ha añadido es la logica suficiente para que sean realistas, es decir, si el usuario no esta registrado o logeado no podra añadir casas, guardar casas en favoritos, etc. La funcion de registrarse con google, pese a darme problemas por las autenticaciones con la pagina que administra por parte de google su propia logica (Google Cloud), he conseguido implementarla correctamente y hacer que el inicio de sesion sea mucho mas sencillo.

Por parte del añadido de propiedades, esto solo es una funcion para usuarios registrados, ya que es necesario guardar las casas con un componente de UserID. La informacion que se guarda de este usuario que crea la casa es usada mas tarde para mostrar el host de la casa al querer entrar en los detalles de la misma, y tambien

permite ver a los usuarios dueños de sus propiedades las casas que han creado en la aplicacion en la seccion de "Mis propiedades".

Lo mas destacable del añadido de las casas es la posibilidad de añadir una imagen de la misma. Esta implementacion la cual se hace a traves del portal gratuito (hasta 25 imagenes por usuario) de Cloudinary. Esta herramienta me ha permitido incorporar imagenes a la web ocuando se crea una nueva casa. Esta herramienta tambien ha sido realmente util por la gran cantidad de documentacion que tiene en internet y los ejemplos que hay de la misma, haciendo mas amigable su uso.

Por ultimo, la parte de concurrencia de la aplicación. La mayor parte de esta se encuentra en la clase principal (./page.tsx), ya que al ser la calse que carga todos los componentes de la primera pantalla es la que mas carga de trabajo tiene. La concurrencia usada es principalmente para aliviar esa carga de trabajo del hilo principal para que la experiencia de usuario sea lo mas fluida posible.

El uso mas constante de concurrencia en mi aplicacion se encuentra en la libreria Memo (memorize) de React.js. ESta libreria es capaz de memorizar componentes y objetos para que sean reutillizados desde la cache de la aplicacion, haciendo que por ejemplo el renderizado de objetos como el banner principal (el cual se encuentra en todas las paginas) y los objetos como las tarjetas individuales de las casas sea mucho mas facil de cargar.

Otra libreria que he usado en la parte concurrente es Lazy de ’react’. Esta libreria se le atribuye a un objeto directamente dentro de tu clase y ayuda al hilo principal a cargar de forma dinamica ese objeto. Yo lo he usado a la hora de cargar las imagenes de las casas. Estas imagenes pese a no ser muy pesadas, tienen un peso grande en el hilo principal cada vez que pasas por la pantalla principal. El uso de Lazy para esto tambien ha ayudado con las imagenes principales cuando entras a los detalles de las casas, ya que esta libreria es especialmente buena para el renderizado de objetos pesados como las imagenes.

Otra forma de concurrencia aplicada es la llamada a los objetos de las casas cuando estas en la pantalla principal. Esta llamada es un tanto especial ya que no esta haciendo una consulta a la API, esta creando los objetos de forma recursiva usando el ID de los mismos y recurriendo en gran parte a la cache de la aplicacion, ayudando a reduir en unos segundos la carga de estos.

Otra libreria realmente util para la carga de componentes grandes ha sido la librería Promise de React. Esta permite cargar funciones que trabajen sobre el hilo principal en paralelo, haciendo asi que este se libere de carga de trabajo y permita al usuario tener una experiencia mas limpia y fluida. Tambien al librearle carga de trabajo en el hilo principal este puede encargarse de cargar otros componentes como el banner o el componente que comprueba la sesion del usuario mas rapido.

El mayor uso de mi libreria Promise se encuentra en la carga de la pantalla principal. Cada vez que entras a esta misma la libreria promise actua, haciendo un promise.all() a todos los objetos de tipo casas que se tienen que cargar en esta pantalla, liberando asi al hilo principal y administrando mejor a carga de trabajo.

En resumen, la parte logica de mi aplicacion esta diseñada para ser lo mas rapida posible, y poder reutilizar todos los elementos que se puede para que no haya redundancia a l ahora de cargar elementos. Clases como las Modal o las Action son claves en la implementacion de esto mismo ya que permite reutilizar funciones graficas y llamadas a la API para que no colapse la aplicación

CONCLUSIONES

Para terminar, quiero añadir que este proyecto me ha gustado mucho, la implementacion pienso que se podria haber mejorado en algunas partes al igual que la parte grafica. La creacion de un mapa el cual muestre las coordenadas de cada casa es la siguiente implementacion que estaria dispusto a aplicarle a mi web, como la posibilidad de reservar fechas para las casas, esto ultimo siendo algo "sencillo" ya que tengo los calendarios implementados y funcionando.

Este proyecto me ha enseñado un nuevo lenguaje y me ha proporcionado nuevas posibilidades con el framework de React, posibilidades que yo antes no conocia y me han parecido realmente útiles.

Por último„ el uso de concurrencia en este tipo de proyectos me parece vital para el funcionamiento de ellos, ya que las posibilidades que puede ofrecer una aplicacion rápida y fluida son casi infinitas. A parte cabe recalcar que al usar tantos datos (en mi caso 4000 objetos de casas), seria casi imposible poder usar la web si esta no es administrada de forma recursiva.