BackEnd Cuestionario Inicial

Indica tu experiencia/conocimiento sobre las siguientes tecnologías, servicios y entornos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tecnología | No lo conozco | Lo conozco | Lo he Administrado | Soy Experto |
| Java |  |  | x |  |
| Java Spring  Framework |  |  | x |  |
| Java Spring  Cloud |  | x |  |  |
| Java Spring boot |  |  | x |  |
| SQL |  |  | x |  |
| API restful |  |  | x |  |
| Diseño Open  API |  |  | x |  |
| Golang |  | x |  |  |
| Python |  | x |  |  |
| Unit Testing |  |  | x |  |
| End to End  Testing | x |  |  |  |
| VCS (GIT u otro) |  |  | x |  |
| Docker |  | x |  |  |
| Kubernetes |  | x |  |  |
| NoSQL |  |  | x |  |
| ElasticShearch | x |  |  |  |

# Ejercicio 1

Escribe, en el lenguaje de programación que desees, un programa que muestre en pantalla los números del 1 al 100, sustituyendo los múltiplos de 3 por el palabro “VIN” y, a su vez, los múltiplos de 5 por “CLE”. Para los casos que, al tiempo, son múltiplos de 3 y 5, utiliza el combinado “VINCLE”.

Desarrollado en C++, puede encontrar la solución aquí: https://godbolt.org/z/de6nhrba4  
Además adjunto el código aquí por si es que el link no funciona:

// Prueba para Virtual Tech Team

// Dominique Verellen

#include <iostream>

int main(int argc, char \*argv[]){

    for (int i = 0; i <100; i++){

        int t = i + 1;

        bool mark = false;

        if(t % 3 == 0){

            std::cout << "VIN";

            mark = true;

        }

        if (t % 5 == 0){

            std::cout << "CLE";

            mark = true;

        }

        if (!mark){

            std::cout << t;

        }

        std::cout << "\n";

    }

    std::cout << std::endl;

}

# Ejercicio 2

Diseña, documenta e implementa una PoC (entregable y ejecutable en docker) de un sistema para poder visualizar en tiempo real la evolución de creación, modificación, parada y destrucción de ítems.

Los Ítems podrán ser de tipo bebida, comida, salsas, especies. Cualquiera de los ítems podrá tener de manera aleatoria las siguientes características:

* Precisa nevera
* No precisa nevera

Cualquiera de los ítems podrá tener una capacidad de:

* 100 gr
* 1000 gr

Cualquiera de los ítems podrá tener un envase de:

* Botella
* caja

Cualquiera de los ítems tendrá:

* Nombre
* Identificador numérico único

Se deberá guardar:

* El nombre del cliente que ha lanzado el comando de creación
* Se deberá guardar el ID del ítem
* Se deberá guardar la hora + timestamp de la operación
* Se deberá guardar el estado
* WAITING: Está procesando una petición (creación, eliminación, cambio, etc.)
* CREATED: creado ok
* DELETED: Está eliminado

Finalmente se deberá implementar un sistema de simulación que lanzando peticiones aleatorias a la API cree, elimine, modifique ítems, para que estos cambios se puedan monitorizar en tiempo real en el dashboard.

En resumen, el sistema deberá proveer:

* Una API restful para gestionar el recurso “ítem” sobre una base de datos. (Diseño

Openapi 3)

* Una App para gestionar desde el backend el recurso ítem implementando la API.
* Una interficie gráfica para visualizar y analizar (drill down) en tiempo real los cambios en la colección de ítems desde el origen de datos.
* Una aplicación de test para lanzar la creación automática y aleatoria de ítems (CRUD completo) contra la API de N elementos en tiempo real.

Desarrollado en Java 17 se encuentran TRES aplicaciones, la aplicación solicitada que realiza consultas aleatorias, una aplicación que recibe las aplicaciones y almacena la información en una base de datos en memoria(H2) y una prueba de concepto utilizando una base de datos NoSQL(DynamoDB). La aplicación con los endpoints para bla BD SQL se encuentra dockerizada con un archivo DOCKERFILE el cual se puede ejecutar para lanzar la aplicación.

La aplicación NoSQL esta desarrollado con Docker Compose para poder lanzar la base de datos DynamoDB en un contenedor y otro contenedor con la misma aplicación.

Se presentas ambas soluciones como pruebas de concepto explorando diferentes tópicos y solucionas a un mismo problema.

Las dos imágenes de Docker se encuentran publicadas en Docker Hub y tienen como nombre:

* dominique120/vtt-sql-test
* dominique120/vtt-nosql-test

Se adjuntan los tres folders en un zip junto con este documento para ser revisado y ejecutado.

Finalmente, la aplicación de pruebas no requiere contenedores y se puede acceder a los dos ambientes de prueba utilizando Swagger 3.0. La aplicación de prueba esta diseñada para ser utilizada con la app SQL, requiere una modificación mínima para ser utilizada con la aplicación NoSQL

Dominique Verellen

Desarrollado entre Feb 7 y Feb 9 2023