UNIVERSIDAD TECNOLÓGICA DE BOLÍVAR

Integrantes:
Carlos Salas- Dylan Primera-Yean Pabon

Facultad de Ingeniería
Programa de Ingeniería de Sistemas y Computación

MIDDLEWARE

Introducción:

En los últimos años la sociedad ha visto un cambio enorme en el desarrollo de las "apps", desde aplicaciones monolíticas sin acceso a la red hasta aplicaciones cuyo funcionamiento requiere por obligación acceso a esta. En este caso se planteó el desarrollo de un conjunto de "apis" que recolectan datos de las redes sociales y que estos mismos serian transmitidos a un middleware para almacenarlos y que por último serian transmitidos al "dashboard "según sea necesario. En este Fragmento del proyecto se plantea como una manera de vincular o enlazar distintas Apis desarrolladas en plataformas diferentes para mostrar un conjunto de datos al usuario.

REQUISITOS FUNCIONALES:

RF 1: Comunicarse con las "apis" de recolección de datos

Descripción: Podra establecer comunicación entre el middleware y las "apis" de recolección de información.
Entrada: Archivos JSON
Prioridad: Alta
Salida: Archivo JSON
Acción: Solicita información y recibe Datos.
Requerimiento: Comunicarse con las "apis" de recolección de datos
Pre-condición: Acceso a la red (Prototipo será local).
Post-condición: Almacenar datos.
Efecto colateral: En caso de fallar la comunicación se procedera a usar una copia de seguridad previa.

RF 2: Almacenar datos en la "BD"

Descripción: Podrá almacenar los datos recolectados de las apis en una base de datos para su posterior uso.
Entrada: Archivos JSON
Prioridad: Alta
Acción: Guardar datos.
Requerimiento: Almacenar datos.

Pre-condición: Acceso a la red (Prototipo será local).
Post-condición: Sincronizar datos
Efecto colateral: En caso de que la base de datos quede comprometida se procedera a usar una de respaldo.

RF 3: Comunicarse con el "dashboard"

Descripción: Podra establecer comunicación entre el middleware y el "dashboard"
Entrada: Archivos JSON
Prioridad: Alta
Salida: Archivo JSON
Acción: Recibe solicitud y envía datos según lo que solicite.
Requerimiento: Comunicarse con el "dashboard"
Pre-condición: Acceso a la red (Prototipo será local).
Post-condición: Enviar datos.
Efecto colateral: En caso de fallar la comunicación se procedera a reiniciar el intento.

REQUISITOS NO FUNCIONALES:

RNF 1: Conexión a la red

Descripción: Propiedad de la aplicación para el acceso a la red.
Estabilidad: Media
Criterio Conceptual: El sistema o aplicación deberá ser capaz de conectarse a la red y establecer comunicación con las respectivas extensiones.
Prioridad: Alta

RNF 2: Disponibilidad

Descripción: Propiedad de la aplicación para responder al usuario cuando este lo necesite.
Estabilidad: Media
Criterio Conceptual: El sistema o aplicación deberá ser capaz de estar disponible cuando el usuario requiere el uso de esta.
Prioridad: Media.

REQUERIMIENTOS DE DESARROLLO:

.NET CORE

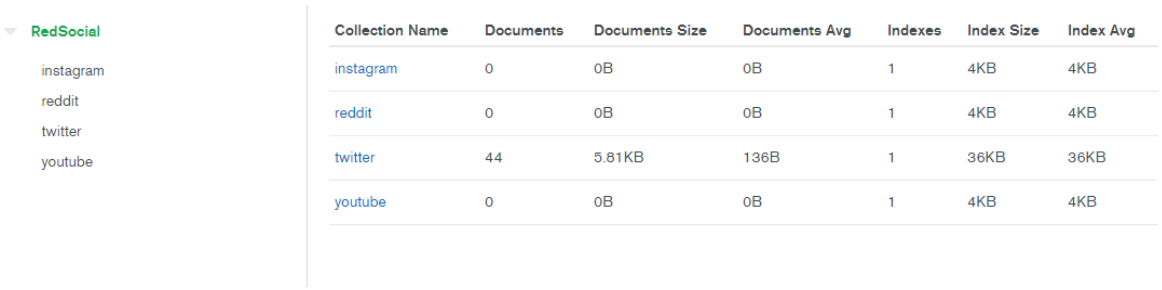
Para el desarrollo del middleware usamos este framework de código abierto y multiplataforma que permite crear aplicaciones modernas conectadas a internet, como aplicaciones web y APIs web.

Para el uso de .Net Core debemos contar con el SDK 3.0 para su correcto funcionamiento. Un entorno de desarrollo como Visual Studio Community 2019 ofrece herramientas que, como diseñadores, editores, depuradores, nos permite programar en diferentes lenguajes como C#, C++, Python, F# y también nos permite desarrollar utilizando el framework de desarrollo web ASP.NET CORE.

MONGO DB

Es una base de datos orientada a documentos. Esto quiere decir que, en lugar de guardar los datos en registros, guarda los datos en documentos. Estos documentos son almacenados en BSON, que es una representación binaria de JSON.

Creamos la base de datos en Mongo Atlas para guardar los datos de las APIs.



dentro de mongo atlas tenemos una colección para cada API donde se guardarán todos los datos devueltos por estas.

NOTA: la base de datos servirá como caché en caso de que se escriba un usuario que haya sido buscado previamente.

DOCKER

Docker es una herramienta para facilitar la creación, implementación y ejecución de aplicaciones mediante el uso de contenedores. Los contenedores permiten a un desarrollador empaquetar una aplicación con todas las partes que necesita, como bibliotecas y otras dependencias, y enviarla como un paquete.

Para instalar Docker se siguió la guía de instalación presente en el siguiente link https://docs.docker.com/v17.09/engine/installation/

METODOLOGÍA

Para la realización del Middleware se trabajó con la metodología ágil Kanban, cuya palabra viene del japonés y significa tarjetas visuales, este cuenta con tres columnas que son los estados de las tareas, la primera es cosas por hacer, la segunda es en progreso, y la última es realizado o hecho.

Con esta metodología buscamos disminuir el trabajo, limitar el trabajo en curso e ir mostrando el proceso de la tarea. Se comienza por lo que se va a realizar, es decir, en específico cada quien escoge una tarea a hacer.

El proceso con la metodología ágil va de esta forma, la plataforma que usamos para las tarjetas visuales es Trello:

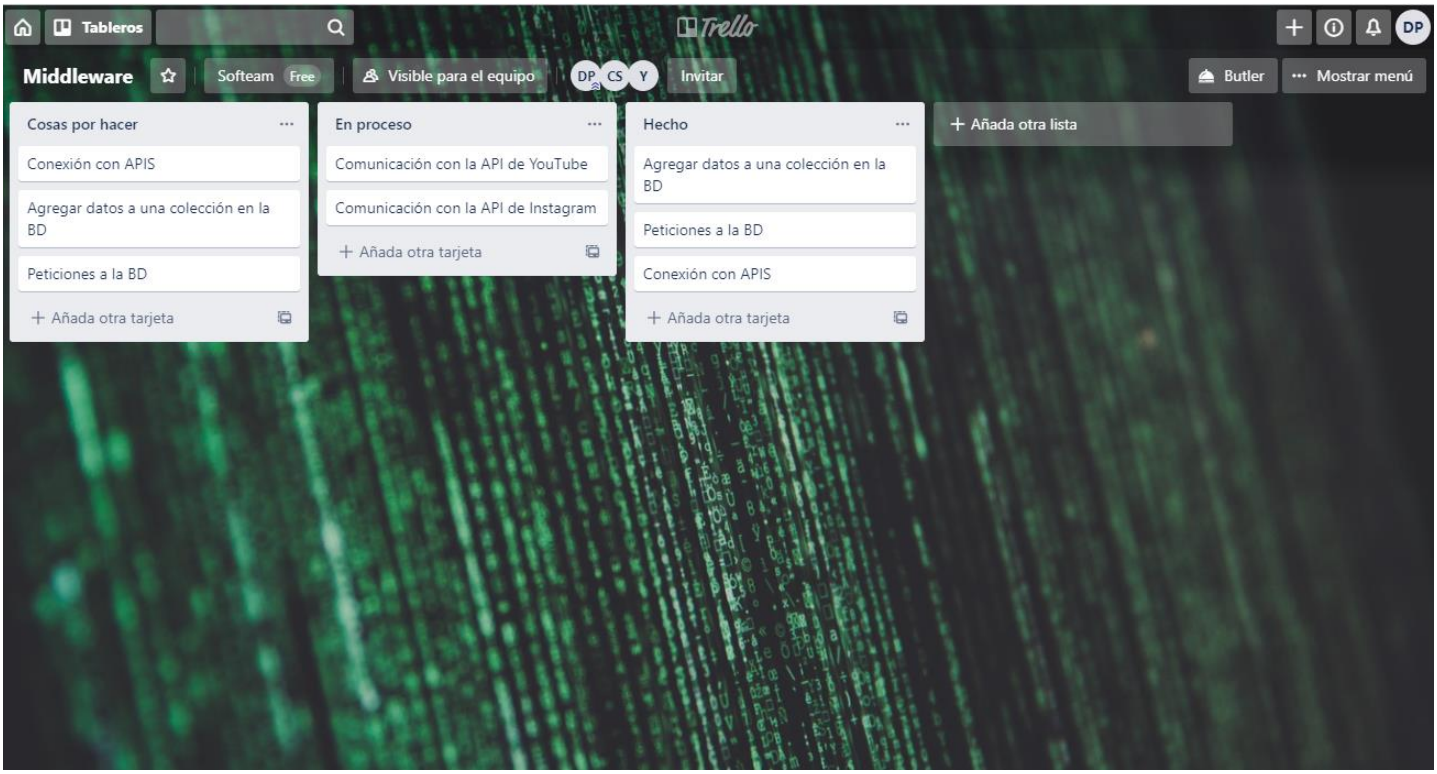


DIAGRAMA UML

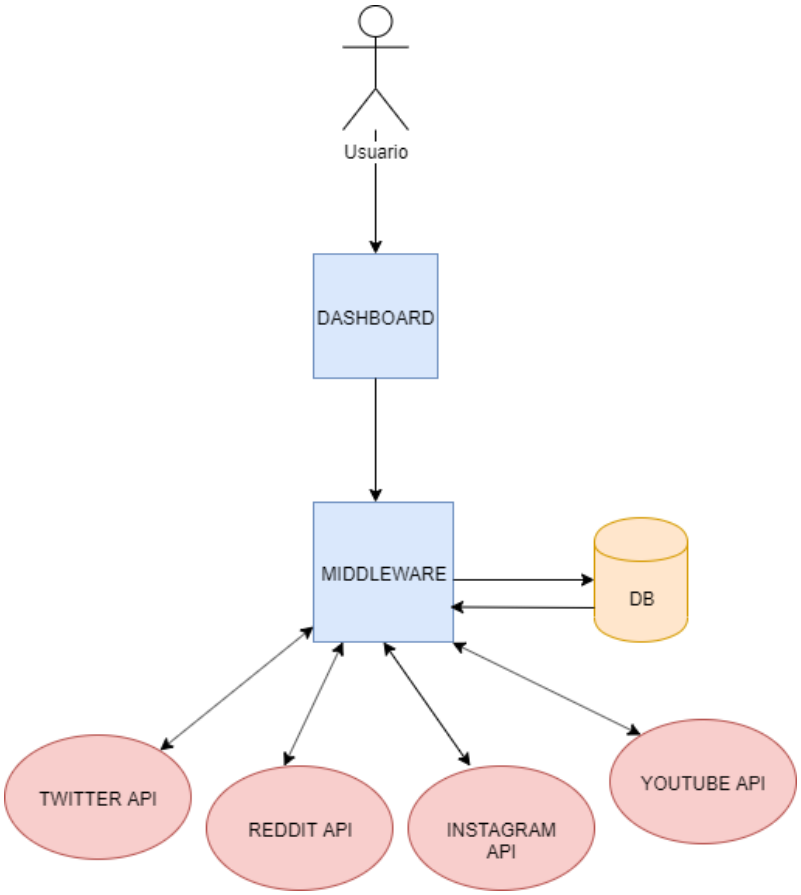
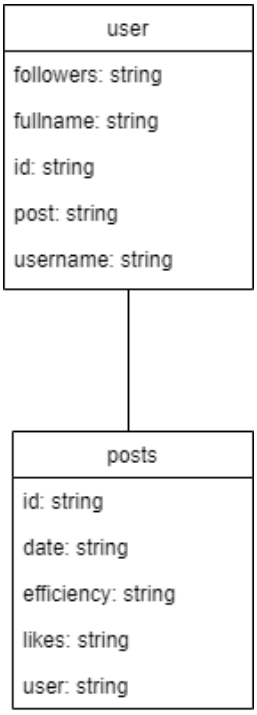


DIAGRAMA BASE DE DATOS



Código:

- https://github.com/IngenieriaDeSistemasUTB/ArcSoft2p2019/tree/master/middleware/WebApplication1

Referencias:

- https://www.crisp.se/gratis-material-och-guider/kanban
- http://www.proyectalis.com/documentos/KanbanVsScrum_Castellano_FINAL-printed.pdf