**Tarea Flask para entregar DWES06**

Apertura: domingo, 1 de enero de 2023, 00:00

Cierre: viernes, 17 de marzo de 2023, 23:55

Introducción

En esta práctica dejamos el gimnasio para ver más el sentido de una API con autentificación. El objetivo de la práctica es diseñar un servidor de mensajería que llamaremos “whatspau”, por sus semejanzas con la aplicación WhatsApp.

Para la autenticación de usuarios en una API utilizaremos la validación “JSON Web Token” que nos proporciona la librería Flask\_JWT

https://pythonhosted.org/Flask-JWT/

Diseño de la base de datos

Puedes encontrar un archivo mySQL adjunto que contiene la base de datos de la aplicación. Tiene las siguientes tablas:

-users\_base: contiene los datos de los usuarios con 3 campos: id\_user, username y password. El password NO está cifrado para no añadir más dificultad a la tarea.

-mensajes: En esta tabla se almacenan todos los mensajes entre todos los usuarios. Los campos id\_sender y id\_receiver son FK de id\_user (users\_base). La clave primaria está formada por la tupla (fecha,id\_sender) puesto que un usuario sólo puede enviar un mensaje en un determinado instante de tiempo. También tiene un campo “status” que indica si el mensaje está enviado, recibido o leído (los ticks de whatsapp devora cada mensaje)

Características de la autenticación

La mayoría de servicios de la API estarán protegidos por @jwt\_required

Para obtener las credenciales se debe obtener el webtoken enviando usuario y password a /auth . Este path NO debe añadirlo a la API, lo añade automáticamente la librería flask\_JWT.

Una vez tengamos el webtoken, servirá para esta sesión y debemos añadirlo a los Headers con el nombre “Authorization”

Servicios protegidos por @jwt\_required (7 puntos)

Dado que el usuario está autenticado tenemos su identificador sin tener que indicarlo en la consulta, ni en la URI ni en el JSON (en caso de que se envíe un JSON). Puede acceder a ella en cualquier punto del código con current\_identity.id

Para realizar la tarea deberá implementar los siguientes servicios:

/users

GET: Recibe la lista de sus “amigos” (todos los usuarios excepto el usuario que hace la consulta)

/mensajes

GET: Recibe todos los mensajes nuevos con status “enviado”

/check

GET: Devuelve el número de mensajes con status “enviado” que tiene el usuario en el servidor.

/mensajes/id\_receiver

GET: Recibe todos los mensajes de la conversación con un usuario concreto

POST: Envía un mensaje a la conversación con un usuario concreto. A JSON sólo se envía el texto del mensaje, los otros campos son gestionados por la API.

/mensajes recibidos

PUT: Cambia el status de los mensajes incluidos en JSON por “Recibo”

/mensajesleídos

PUT:Cambia el status de los mensajes incluidos en JSON por “Leído”

Tarea avanzada (3 puntos)

Realiza los cambios necesarios para permitir la existencia de grupos de mensajería.

Para hacer esto deberás generar nuevas tablas en la base de datos y nuevos servicios a la API. Tenga en cuenta que los mensajes podrán estar "enviados" a un usuario, "recibidos" por otro usuario y "leídos" por un tercer usuario.

Asimismo, para optimizar el almacenamiento en la base de datos el “texto” de un mensaje sólo deberá guardarse una vez.

Entrega

Tienes que entregar un archivo comprimido que contenga la carpeta con todos los archivos utilizados en la tarea, entre los que tendremos el código python de la tarea y el archivo postman\_collection de Postman que permita probar los diferentes servicios.

El documento debe tener el formato

PrimerLinaje\_SegundoLinaje\_DWES06.zip o .rar

**Tasca Flask per lliurar DWES06**

**Apertura:** domingo, 1 de enero de 2023, 00:00

**Cierre:** viernes, 17 de marzo de 2023, 23:55

**Introducció**

En aquesta pràctica deixam el gimnàs per veure més el sentit d’una API amb autentificació. L’objectiu de la pràctica és dissenyar un servidor de missatgeria que anomenarem “whatspau”, per les semblances que tindrà amb l’aplicació WhatsApp.

Per l’autenticació d’usuaris a una API utilitzarem la validació “JSON Web Token” que ens proporciona la llibreria Flask\_JWT

<https://pythonhosted.org/Flask-JWT/>

**Disseny de la base de dades**

Pots trobar un arxiu mySQL adjunt que conté la base de dades de l’aplicació. Té les següents taules:

-users\_base: conté les dades dels usuaris amb 3 camps: id\_user, username i password. El password NO està encriptat per no afegir més dificultat a la tasca.

-missatges: En aquesta taula s’emmagatzemen tots els missatges entre tots els usuaris. Els camps id\_sender i id\_receiver son FK de id\_user (users\_base). La clau primària està formada per la tupla (fecha,id\_sender) ja que un usuari només pot enviar un missatge en un determinat instant de temps. També te un camp “status” que indica si el missatge està **enviat**, **rebut**o **llegit** (els ticks de whatsapp devora cada missatge)

**Característiques de la autenticació**

La majoria de serveis de la API estaran protegits per **@jwt\_required**

Per obtenir les credencials s’ha dobtenir el webtoken enviant usuari i password a **/auth** . Aquest path NO l’heu d’afegir a la API, l’afegeix la llibreria flask\_JWT automàticament.

Una vegada tenguem el webtoken, en servirà per aquesta sessió i l’hem d’afegir als Headers amb el nom “Authorization”

**Serveis protegits per @jwt\_required (7 punts)**

Com que l’usuari està autenticat tenim el seu identificador sense haver-ho d’indicar a la consulta, ni a la URI ni al JSON (en cas de que s’envii un JSON). Hi podeu accedir en qualsevol punt del codi amb **current\_identity.id**

Per fer la tasca haureu d'implementar els següents serveis:

* /users
  + GET: Rep la llista dels seus “amics” (tots els usuaris excepte l’usuari que fa la consulta)
* /missatges
  + GET: Rep tots els missatges nous amb status “enviat”
* /check
  + GET: Torna el número de missatges amb status “enviat” que te l’usuari al servidor.
* /missatges/id\_receiver
  + GET: Rep tots els missatges de la conversa amb un usuari concret
  + POST: Envia un missatge a la conversa amb un usuari concret. Al JSON només s’envia el texte del missatge, els altres camps els gestiona la API.
* /missatgesrebuts
  + PUT: Canvia l’status dels missatges inclosos al JSON per “Rebut”
* /missatgesllegits
  + PUT:Canvia l’status dels missatges inclosos al JSON per “Llegit”

 Tasca avançada (3 punts)

Fes els canvis necessaris per permetre l’existència de grups de missatgeria.

Per fer això hauras de generas noves taules a la base de dades i nous serveis a la API. Teniu en compte que els missatges podran estar “enviats” a un usuari, “rebuts” per un altre usuari i “llegits” per un tercer usuari.

Així mateix, per optimizar l’emmagatzament a la base de dades el “texte” d’un missatge només s’haurà de guardar una vegada.

Lliurament

Has de lliurar un fitxer comprimit que contingui la carpeta amb tots els arxius utilizats a la tasca, entre els que tendrem el **codi python**de la tasca i l’arxiu **postman\_collection** de Postman que permeti provar els diferents serveis.

El document ha de tenir el format

**PrimerLLinatge\_SegonLlinatge\_DWES06.zip** o .rar