PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2513 - Sección 1 — Tecnologías y Aplicaciones Web

Tarea 3

Entrega

- **Fecha y hora**: Sábado 4 de noviembre del 2023, a las 22:00
- **Lugar:** Repositorio individual en la organización del curso en GitHub

Objetivos

- **Comprender** los fundamentos de autenticación y autorización.
- **Producir** documentación efectiva y clara del proceso realizado.

Descripción

En esta tarea, tendrán la oportunidad de crear un sistema de autenticación basado en *tokens* utilizando el lenguaje de programación Python. De esta forma, podrán entender cómo funciona el proceso de autenticación y autorización, lo que los preparará para abordar desafíos del mundo real en el desarrollo de aplicaciones web, donde la seguridad y la protección de datos son preocupaciones críticas.

Para esta tarea, se les proveerá un código base creado por el equipo docente que tiene como propósito simular proceso de inicio de sesión en una aplicación web. Para ello, deberán hacer uso de las clases `User` y `Token`, donde la primera representa al cliente que intenta iniciar sesión, mientras que la segunda es el *token* que este debiese utilizar para acceder a los distintos recursos reservados para cada uno de los roles en la aplicación. En específico, deberán implementar un JSON Web Token (JWT), que corresponde a un mecanismo para poder transmitir información entre dos partes de forma segura.

Tendrán que completar este *script* para que cumpla con las funcionalidades que se especifican a continuación.

Clase User

Esta clase representa al usuario, que tiene como atributos correo electrónico, contraseña y rol (que sirve para la autorización de acceso de rutas protegidas). Además, tiene las siguientes funciones que deben completar:

- Método `encrypt_password()`: recibe la constraseña del usuario y se encarga de encriptarla utilizando el algoritmo de *bcrypt*, que es una función de *hash* de contraseña segura. Retorna el *hash* resultante, que es el valor seguro que se almacena en la base de datos en lugar de la contraseña en texto plano.
- Método `check_password()`: recibe una contraseña en texto plano ingresada por el usuario. Deben verificar si esta coincide con la contraseña almacenada en la instancia de la clase User (que debe ser un *hash* encriptado). Retorna un booleano que indica si la contraseña es correcta o no.

Clase Token

Esta clase se encarga del manejo de *tokens*. Estos son de suma relevancia para autenticar a los usuarios y otorgarles permisos específicos dado su rol específico. Se compone de los siguientes métodos, que tendrán que completar:

- Método `encode()`: recibe el *payload* y el *header* del *token*, es decir, la parte de un *token* que contiene la información útil o los datos que se transportan. Deberán componer el JWT a partir de estos datos, agregando además el tiempo de expiración de este. Retorna el *token* encriptado como un *string*.
- Método `decode()`: recibe el *token* de un usuario. Deben completar esta función tal que desencripte este *token* y retorne el *header* y el *payload* de este.
- Método `validate()`: este método se debe utilizar cuando un usuario esté intentando acceder a cierto recurso reservado para un rol en particular. Recibe el *token* del usuario y el *rol* necesario para acceder. Deben completar esta función tal que se revise si el *token* es válido, vigente y corresponde al de un usuario autorizado. Retorna un booleano que indica la validez de este.

Manejo de sesión

Para probar el manejo de sesión utilizando las clases mencionadas anteriormente, se utilizarán dos funciones que deberán completar:

- Función `register()`: recibe el correo electrónico, contraseña y rol del usuario. Retorna un booleano que indica si el registro ha sido exitoso, en caso que el usuario se encuentre registrado, deberán imprimir un mensaje y retornar falso. Deben registrar al nuevo usuario con los datos dados.

  Nota: para asegurar el correcto funcionamiento de la base de datos de usuario, el correo electrónico de cada usuario debe ser único. Consideren este detalle al realizar el registro de usuarios.
- Función `login()`: recibe el correo electrónico y contraseña del usuario. Realiza el proceso de inicio de sesión y retorna el *token* asociado al usuario. Si los datos del usuario no están previamente registrados, retorna `None`.

Documentación

Este ítem tiene como propósito:

- Explicar cómo funciona el código que implementaron.
- Indicar apropiadamente cómo correr su programa y que librerías hay que instalar (si es que aplica).

Además, en sus respectivos repositorios encontrarán una serie de preguntas teóricas acerca de su tarea y del uso de *tokens* en general. Deberán responder estas preguntas de forma más completa posible.

Para un mejor entendimiento sobre cómo escribir la documentación, se les hará entrega de una plantilla con la estructura que debe seguir, su descripción y las preguntas que deben responder.

Consideraciones

- Prohibición de bibliotecas:

  Está estrictamente prohibido utilizar la biblioteca `jwt` o cualquier otra biblioteca similar para la generación o validación de *tokens* en esta tarea. El uso de esta librería o de similares en este contexto implicará una nota máxima de 4 en la tarea. El objetivo principal de esta tarea es que adquieran la comprensión y habilidades necesarias para crear un sistema de autenticación y autorización personalizado, en lugar de depender de soluciones preconstruidas.

- Política de revisión antiplagio:

  Les recordamos que se aplicará una política de revisión antiplagio rigurosa en esta tarea. Se espera que trabajen de manera independiente y cumpla con los principios éticos de la integridad académica. Cualquier forma de copia o plagio, ya sea de fuentes en línea, de otros estudiantes o de cualquier otra fuente, resultará en consecuencias académicas negativas, que pueden incluir la anulación de la tarea y medidas disciplinarias adicionales, de acuerdo con las políticas de integridad académica de la institución.

- Estandar de codificación de caracteres:

  Es esencial llevar a cabo los procesos de codificación y decodificación utilizando el estándar de codificación de caracteres `utf-8`. Esto es fundamental para mantener la integridad y la precisión de los datos, así como para garantizar la interoperabilidad en entornos multilingües y diversas plataformas.

Entregables

Cada entrega deberá incluir los siguientes archivos:

- `README.md` con la documentación de tu programa. Deben mencionar brevemente aquello que pudiste implementar y también lo que no. Deben indicar cómo correr tu programa y responder las preguntas indicadas en el *template* que te facilitamos. Para finalizar, debes explicar claramente cómo funciona cada función de tu programa.
- Archivo `.py` desarrollado en base al material entregado.

Rúbrica

A continuación, se describe el criterio de cada uno de los ítems de la rúbrica con la que se evaluará esta entrega. La nota se calcula al 50 % considerando un total de 25 puntos.

- **Clase User: función** `encrypt_password()` **[1.5 puntos]**: Se espera que utilicen la librería `bcrypt` para encriptar la contraseña recibida y la retornen.
- **Clase User: función** `check_password()` **[1 punto]**: Se espera que este método valide que la contraseña guardada en la base de datos coincida con la que se está entregando como *input*.
- **Clase Token: función** `encode()` **[4 puntos]**: Se espera que generen el JWT a partir del *payload* y *header* entregado y el tiempo de expiración. El *token* debe contener todas las partes de un *jason web token*. Utilizar la librería `jwt` NO es válido. Recuerda que para que te funcione el *enconding* deben mandar un *header* y *payload* válido en sus formatos correspondientes.
- **Clase Token: función** `decode()` **[4 puntos]**: Se espera que este método sea capaz de desencriptar el *token* que recibe y retornar las partes de este que se piden (*header* y *payload*). Utilizar la librería `jwt` NO es válido.

- **Clase Token: función** `validate()` **[1.5 puntos]**: Se espera que validen el *token* para acceder a una ruta protegida, verificando el tiempo de expiración, la existencia del *token* y la autorización del rol específico del usuario que está intentando acceder.
- **Función** `register()` **[2 puntos]**: Se espera que registren a los usuarios con sus respectivos atributos y lo añadan a su base de datos. Deben tener el cuenta que no pueden existir dos usuarios con el mismo correo electrónico asociado.
- **Función** `login()` **[3 puntos]**: Se espera que verifiquen los datos para iniciar sesión y, en caso de que sean correctos, entreguen el token asociado al usuario que intenta hacer *login*.
- **Documentación [8 puntos]**: Se espera que expliquen cómo funciona su código, cómo correr el programa, además de responder las preguntas dadas en el *template*.

Dudas

Cualquier duda que se presente acerca del enunciado debes consultarla en las ***issues*** del repositorio del curso. Recuerden que como equipo docente estaremos atentos para poder ayudarlos :)