PRACTICA PASOS

Etiquetas	
Fecha de creación	@3 de diciembre de 2023 18:26

registro

1. generar RSA kpriv y kpub
2. generar klog y datos
3. encriptar kpriv con kdatos
4. almacenar kprivdatos y klogin, kpub

login:

1. comprobar klog y generar kdatos
2. descargar kprivdatos y klogin, kpub
3. obtener kpriv de kprivdatos desencriptando con kdatos

encriptar

1. usar kpub para encriptar keys
2. con keys encriptar archivo

desencriptar

1. con kpriv desdencriptar keys
2. con key desencriptar archivo

Servidor:

SERVER_URL = "http://127.0.0.1:5000"

endpoints :

/register POST

recibe:

```
data_server={
        "user":user,
        "klogin":k_login,
        "kpub":public_key_str,
        "kprivkdatos":kprivkdatos
    }
```

debe devolver el codigo de status : 200 si paso, otro si no se registro, almacena los datos enviados pero al klogin le hace un hash

/login: POST

recibe

```
payload ={
        'user': user,
        'password' : password
    }
```

debe devolver tanto la kpub como la kprivkdatos despues de verificar el klogin para el usuario correspondiente

/user: POST

recibe

```
payload ={
        'k_publica': k_publica,
    }
```

debe devolver la kpub del usuario correspondiente

/upload POST

recibe:

```
payload_bruto ={
        'user': user,
        'archivos':  {
            (0 a n): {
                'archivo': archivo64,
                'llave': llave64
               }
```

```
        }
    }
```

debe almacenar cada archivo en la carpeta correspondiente de la forma (puede ser tambien en un json o lo que sea mas facil, no necesariamente un archivo en local, sino un database.json

/user

/local

/0

archivo.enc

llave.bin

/download/user GET

no recibe nada, obtiene el user de la peticion GET en la URL

debe devolver todos los archivos y llaves del usuario en un json de la siguiente forma:

```
response ={
        'archivos':  {
                    (0 a n): {
                        'archivo': archivo64,
                        'llave': llave64
                       }
       }
    }
```

`/share POST SUBIR SE VAN A COMPARTIR 1 a 1`

despues de hacer llamado a /user y tener la kpub se llama al oro endpoint que recibe recibe

```
payload_bruto ={
        'receptor': receptor, <---------- Nombre de quien recibe p.e. Christian
        'data':  {
            'archivo': encriptarCompartido(fichero,receptorkpub)[0],
            'llave': encriptarCompartido(fichero,receptorkpub)[1]
```

```
        }
    }
```

debe almacenar cada archivo en la carpeta correspondiente al usuario ***RECEPTOR*** de la forma (puede ser tambien en un json o lo que sea mas facil, no necesariamente un archivo en local, sino un database.json

/receptor

/share

/0

archivo.enc

llave.bin

/share/user GET

no recibe nada, obtiene el user de la peticion GET en la URL

debe devolver todos los archivos y llaves del usuario en un json de la siguiente forma:

```
response ={
        'archivos':  {
                    (0 a n): {
                        'archivo': archivo64,
                        'llave': llave64
                       }
       }
    }
```

los archivos devueltos son los de la “BBDD” ***share, no local***

GET /share/”Christian”

Descarga : Christian/compartida/

POST /share

Usuario

- Propios
  - 0
    - archivo
    - key
  - 1
    - archivo
    - key
- Compartidos
  - 0
    - archivo
    - key
  - 1
    - archivo
    - key

key → key_str → key_str.encode() → key_crypted → key_crypted_str

key_crypted_str → key_crypted → key_str.encode() → key_str → key