MessengerServer Documentation

Jakub Kucharski, Adam Kuder

Mar 28, 2019

Reference documentation

CHAPTER 1

Modules

1.1 messenger_server.database module

exception messenger_server.database.**ContactExistingError**

class messenger_server.database.**DatabaseConnection**

Single SQLite3 Database connection.

Parameters

- **_connection** (sqlite3.Connection) – Connection object for the database
- **_cursor** (sqlite3.Cursor) – Cursor object returned by _connection

add_contact(*user*, *contact*)

Insert a record representing a (user, contact) pair.

Parameters

- **user** (*str*) – Login of user who is adding a new contact
- **contact** (*str*) – Login of user who is being added as contact

Raises *messenger_server.database.ContactExistingError* if such pair already exists in the database

Raises *messenger_server.database.UserNotFoundError* if contact does not exist in the users table

add_user(*email*, *login*, *password*)

Insert a record representing a single user.

Parameters

- **email** (*str*) – User's email address
- **login** (*str*) – User's login
- **password** (*str*) – User's password

Raises *messenger_server.database.LoginTakenError* if a record with identical login already exists

get_contacts_list(*user*)
Get a list of given user's contacts.

Parameters **user** (*str*) – Login of user whose contacts are being retrieved

Returns a list of logins

Return type list(str)

get_users_with_contact(*contact*)
Get a list of users who have given contact.

Parameters **contact** (*str*) – Login of user being searched as a contact

Returns a list of logins

Return type list(str)

setup()
Prepare initial database structure in case it doesn't exist yet.

This method should be called once before starting the server.

verify_login(*login*, *password*)
Verify if given login matches given password in the database.

Parameters

- **login** (*str*) – Login being matched
- **password** (*str*) – Password being matched

Raises *messenger_server.database.LoginNotFoundError* if there is no record with given login

Raises *messenger_server.database.PasswordError* if password retrieved from the database does not match the password provided

exception messenger_server.database.**LoginNotFoundError**

exception messenger_server.database.**LoginTakenError**

exception messenger_server.database.**NoContactsError**

exception messenger_server.database.**PasswordError**

exception messenger_server.database.**UserNotFoundError**

1.2 messenger_server.request_handler module

exception messenger_server.request_handler.**InvalidRequestError**
Bases: Exception

class messenger_server.request_handler.**RequestHandler**(*request*, *client_address*, *server*)
Bases: socketserver.BaseRequestHandler

Class used to handle each TCP request.

As required by socketserver.BaseServer this class inherits from socketserver.BaseRequestHandler and is provided to the socketserver.TCPServer to be instantiated for every incoming request and call the *handle()* function.

Parameters

- **request** (socket.socket) – a socket created by the request
- **prefixes_to_methods** (*dict(int, function)*) – mapping of requests' prefixes to appropriate handling methods

handle()
Retrieve incoming data and execute adequate handler function.

When data has been received from the socket, the first byte is used to determine the type of the request, in accordance with the protocol. Corresponding method is then passed the received data minus the first byte.

handle_add_contact_request(*add_contact_request*)
Perform adding user's contact and send adequate response.

Parameters **add_contact_request** (*bytes*) – Add-contact request without the prefix

handle_get_contacts_request(*get_contacts_request*)
Perform fetching user's contact list and send adequate response.

Parameters **get_contacts_request** (*bytes*) – Get-contacts request without the prefix

handle_incoming_message(*incoming_message_request*)
Perform passing a message to another user and other adequate actions.

Parameters **incoming_message_request** (*bytes*) – Incoming message without the prefix

handle_login_request(*login_request*)
Perform user log-in and send adequate response.

Parameters **login_request** (*bytes*) – Log-in request without the prefix

handle_signup_request(*signup_request*)
Perform user sign-up and send adequate response.

Parameters **signup_request** (*bytes*) – Sign-up request without the prefix

1.3 messenger_server.user_management module

The following classes are oriented toward working with users as they are signed-in (i.e. have established a socket connection with the server and authenticated themselves).

class messenger_server.user_management.**OnlineUser**(*login*, *key*, *socket*)
Class represents a user at the time of being signed-in.

Parameters

- **login** (*str*) – User's login
- **key** (*int*) – User's identification key
- **socket** (socket.socket) – Socket established between the user and the server

class messenger_server.user_management.**UserManager**
This class keeps track of users currently logged in.

classmethod get_online_user_by_key(*key*)
Get online user with matching identification key.

Parameters **key** (*int*) – Identification key

Returns Online user whose key has been matched

Return type *messenger_server.user_management.OnlineUser* or None

classmethod get_online_user_by_login(*login*)
Get online user with matching login.

Parameters **login** (*str*) – User's login

Returns Online user whose login has been matched

Return type *messenger_server.user_management.OnlineUser* or None

classmethod sign_in(*login*, *socket*)
Generate unique identification key and add new online user.

Parameters

- **login** (*str*) – User's login
- **socket** (socket.socket) – Socket established between the user and the server

Returns Online user with newly generated key

Return type *messenger_server.user_management.OnlineUser*

1.4 messenger_server.user_notifying module

class messenger_server.user_notifying.**UserNotifier**
Class for sending packets that are not direct responses to incoming requests.

static notify_users_of_contact_login(*contact*)
Send status update to everyone who has the user in their contacts and is online.

Parameters **contact** (*messenger_server.user_management.OnlineUser*) – User whose status was updated

static send_message(*sender*, *recipient*, *message*)
Send message to specified recipient.

Parameters

- **sender** (*messenger_server.user_management.OnlineUser*) – Sender of the message
- **recipient** (*messenger_server.user_management.OnlineUser*) – Recipient of the message
- **message** (*str*) – Contents of the message

Python Module Index

m

Index

R

S

U

V