Komunikator

Generated by Doxygen 1.12.0

1 Documentation for "Komunikator" backend

1.0.1 Overview

Komunikator app backend is built with Flask, Flask-SocketIO, and SQLAlchemy. It provides a REST API and Web↩ Socket interface for user management, room-based messaging, and real-time presence tracking.

1.0.2 Features

1.0.2.1 User Management

- **Authentication**: JWT-based user registration and login system
- **User Statistics**: Track message counts and room memberships
- **Online Status**: Real-time presence detection with automatic offline detection after inactivity

1.0.2.2 Room Management

- **Public Rooms**: Open rooms visible to all users
- **Private Rooms**: Invite-only rooms with unique access keys
- **Room Operations**: Create, edit, delete, join, and leave rooms
- **User Lists**: View all users in a room with real-time updates
- **Room Moderation**: Room owners can kick users from their rooms

1.0.2.3 Messaging

- **Real-Time Messages**: WebSocket-based instant message delivery
- **Chat History**: Persistent message storage with timestamps
- **Message Validation**: Automatic validation of message length and content
- **Broadcast System**: Efficient message distribution to room members

1.0.2.4 Activity Tracking

- **Presence System**: Tracks user online/offline status across all rooms
- **Watchdog Mechanism**: Periodic heartbeat to maintain user presence
- **Automatic Cleanup**: Background task removes inactive users after timeout
- **Cross-Room Updates**: User status changes broadcast to all relevant rooms

1.0.3 Architecture

1.0.3.1 Database Layer (`db_objects.py`)

Four main models represent the application's data:

- **Users**: User accounts with credentials
- **Rooms**: Chat rooms with public/private access control
- **Users_room**: Many-to-many relationship between users and rooms
- **Chat_history**: Persistent message storage

1.0.3.2 REST API Routes

- `/api/register` - User registration
- `/api/login` - User authentication
- `/api/user/change_password` - Password modification
- `/api/user/get_info` - User statistics
- `/api/users/online` - List of users online
- `/api/rooms` - Public room listing
- `/api/room/*` - Room management operations
- `/api/user_rooms` - User's room memberships
- `/api/user/list` - List of users in a room
- `/api/user_kick` - Forceful removal of a user by the room owner
- `/api/chat_history` - Historical messages

1.0.3.3 WebSocket Events

- `join` - Connect to a room (sent to server)
- `leave` - Disconnect from a room (sent to server)
- `message` - Send a message to a room (sent to server)
- `watchdog` - Maintain user presence (sent to server)
- `user_list_updated` - Notify when user list was updated (sent to client)
- `room_list_updated` - Notify when room list was updated (sent to client)
- `user_online` - Notify when user status was changed to online (sent to rooms)
- `user_offline` - Notify when user status was changed to offline (sent to rooms)
- `user_joined` - Response for successful join event (sent to room)
- `user_left` - Response for successful leave event (sent to room)
- `new` - Response for message event (sent to room)
- `error` - Response on event failure (sent to client)

1.0.3.4 Application State (`app_state.py`)

In-memory state management:

- **online_users**: Set of currently active users
- **user_last_seen**: Timestamp tracking for activity detection
- **room_users**: Mapping of rooms to their current users
- **user_rooms**: Mapping of users to their joined rooms

1.0.4 Technology Stack

- **Flask**: Web framework
- **Flask-SocketIO**: WebSocket support
- **SQLAlchemy**: ORM and database management
- **Flask-JWT-Extended**: JWT authentication
- **MySQL**: Database backend
- **Flask-CORS**: Cross-origin resource sharing

1.0.5 Getting Started

Configure the following environment variables:

- `DB_USER` - Database username
- `DB_PASSWORD` - Database password
- `DB_HOST` - Database host
- `DB_PORT` - Database port
- `DB_NAME` - Database name
- `JWT_SECRET_KEY` - Secret key for JWT tokens

Authors

Paweł Dyczek, Paweł Herzyk, Mikołaj Całus

2 Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

3 Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

db.Model

4 Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

5 File Index

5.1 File List

Here is a list of all files with brief descriptions:

6 Namespace Documentation

6.1 app_state Namespace Reference

Functions

- update_user_room_maps (int room_id, str user_name, bool remove=False)

  Add or remove a user from user_rooms and room_users maps.
- remove_room (int room_id)

  Remove a room and clean up all associated user references.

Variables

- socketio = SocketIO(cors_allowed_origins="*")
- online_users = set()
- dict user_last_seen = {}
- dict room_users = {}
- dict user_rooms = {}

6.1.1 Function Documentation

remove_room()

```
app_state.remove_room (
            int room_id)
```

Remove a room and clean up all associated user references.

Parameters

	room_id	The ID of the room to remove.

Returns

None

update_user_room_maps()

```
app_state.update_user_room_maps (
            int room_id,
            str  user_name,
            bool  remove = False)
```

Add or remove a user from user_rooms and room_users maps.

Parameters

	room_id	The ID of the room.
	user_name	The name of the user.
	remove	If True, remove the user from the room.

Returns

None

6.1.2 Variable Documentation

online_users

```
app_state.online_users = set()
```

room_users

```
dict app_state.room_users = {}
```

socketio

```
app_state.socketio = SocketIO(cors_allowed_origins="*")
```

user_last_seen

```
dict app_state.user_last_seen = {}
```

user_rooms

```
dict app_state.user_rooms = {}
```

6.2 authorization Namespace Reference

Functions

- register ()
  Register a new user.
- login ()
  Login an existing user.

Variables

- bp = Blueprint("auth", __name__, url_prefix="/api")

6.2.1 Function Documentation

login()

```
authorization.login ()
```

Login an existing user.

Parameters

	user_name	The username of the existing user.
	password	The password of the existing user.

Returns

A dictionary containing the access token of the logged in user.

Exceptions

	400	Missing user_name or password.
	401	Invalid user credentials.

register()

```
authorization.register ()
```

Register a new user.

Parameters

	user_name	The username of the new user.
	password	The password of the new user.

Returns

A dictionary containing the access token of the newly registered user.

Exceptions

	400	Missing user_name or password.
	409	User with given user_name already exists.

6.2.2 Variable Documentation

bp

```
authorization.bp = Blueprint("auth", __name__, url_prefix="/api")
```

6.3 chat_history Namespace Reference

Functions

- get_chat_history ()

  Get the chat history of a room by its id.

Variables

- bp = Blueprint('chat_history', __name__, url_prefix='/api')

6.3.1 Function Documentation

get_chat_history()

```
chat_history.get_chat_history ()
```

Get the chat history of a room by its id.

Parameters

	room_id	The id of the room to get the chat history for.

Returns

A list of chat history messages, each containing the message_id, room_id, user_name, message, and create_date of the message.

Exceptions

	400	If the room_id parameter is missing.
	404	If the room is not found.

6.3.2 Variable Documentation

bp

```
chat_history.bp = Blueprint('chat_history', __name__, url_prefix='/api')
```

6.4 db_objects Namespace Reference

Classes

- class Chat_history

  Chat_history model representing a message in a chat room.
- class Rooms

  Database model representing a chat room.
- class Users

  Users model representing a user account in the database.
- class Users_room

  Database model representing the association between users and rooms.

Variables

- db = SQLAlchemy()

6.4.1 Variable Documentation

db

```
db_objects.db = SQLAlchemy()
```

6.5 main Namespace Reference

Functions

- initialize_room_users ()

  Initialize the room_users and user_rooms maps by iterating over all rooms and their users.
- create_app ()

  Create a Flask app with the necessary configurations and routes.

Variables

- app = create_app()
- debug
- True
- host
- port

6.5.1 Function Documentation

create_app()

```
main.create_app ()
```

Create a Flask app with the necessary configurations and routes.

This function initializes a Flask app with CORS enabled, JWTManager configured, and SQLAlchemy configured with the database URI from the environment variables.

It also registers the necessary blueprints for the routes.

Returns

The created Flask app.

initialize_room_users()

```
main.initialize_room_users ()
```

Initialize the room_users and user_rooms maps by iterating over all rooms and their users.

Uses the update_user_room_maps function to update the maps for each user in each room.

Returns

None

6.5.2 Variable Documentation

app

```
main.app = create_app()
```

debug

```
main.debug
```

host

```
main.host
```

port

```
main.port
```

True

```
main.True
```

6.6 room Namespace Reference

Functions

- handle_join (data)
  Handle a user joining a room.
- handle_leave (data)
  Handle a user leaving a room.
- handle_message (data)
  Handle a message sent by a user to a room.
- handle_watchdog (data)
  Handle a watchdog event sent by a user.

6.6.1 Function Documentation

handle_join()

```
room.handle_join (
              data)
```

Handle a user joining a room.

Parameters

	data	A dictionary containing the user_name and room_id.

Note

If the user_name or room_id is missing, an error is emitted.

If the room is private and the user is not in the room, an error is emitted.

If the user joins the room successfully, user_joined is emitted to the room.

handle_leave()

```
room.handle_leave (
              data)
```

Handle a user leaving a room.

Parameters

	data	A dictionary containing the user_name and room_id.

Note

If the user_name or room_id is missing, an error is emitted.

If the user leaves the room successfully, user_left is emitted to the room.

handle_message()

```
room.handle_message (
              data)
```

Handle a message sent by a user to a room.

Parameters

	data	A dictionary containing the room_id, user_name, and message.

Note

If the user_name or room_id is missing, an error is emitted.

If the message is missing, an error is emitted.

If the message is too long, an error is emitted.

If the message is sent successfully, new_message is emitted to the room.

handle_watchdog()

```
room.handle_watchdog (
              data)
```

Handle a watchdog event sent by a user.

Parameters

	data	A dictionary containing the user_name.

Note

If the user_name is missing, an error is emitted.

If the watchdog event is sent successfully, update_user_last_seen is called with the user_name.

6.7 rooms Namespace Reference

Functions

- get_rooms ()
  Get a list of all public rooms.
- get_room ()
  Get a room by its id.
- join_public_room ()
  Join a public room by its id.
- join_private_room ()
  Join a private room by its access key.
- leave_room ()
  Leave a room by its id.
- create_room ()
  Create a room.
- delete_room ()
  Delete a room by its id.
- edit_room ()
  Edit a room by its id.

Variables

- bp = Blueprint('rooms', __name__, url_prefix='/api')

6.7.1 Function Documentation

create_room()

```
rooms.create_room ()
```

Create a room.

Parameters

	room_name	The name of the room to create.
	is_private	Whether the room is private or not.
	access_key	The access key for the room.

Returns

A dictionary containing the message of the create action.

Exceptions

	400	If the room_name, room_owner, or access_key parameter is missing.
	400	If a room with the given access key already exists.

delete_room()

```
rooms.delete_room ()
```

Delete a room by its id.

Parameters

	room_id	The id of the room to delete.

Returns

A dictionary containing the message of the delete action.

Exceptions

	400	If the room_id parameter is missing.
	404	If the room is not found or the user is not the room owner.

edit_room()

```
rooms.edit_room ()
```

Edit a room by its id.

Parameters

	room_id	The id of the room to edit.
	new_access_key	The new access key for the room.
	is_private	Whether the room is private or not.
	new_name	The new name for the room.

Returns

A dictionary containing the message of the edit action.

Exceptions

	400	If the room_id parameter is missing.
	404	If the room is not found.
	403	If the user is not the room owner.

get_room()

```
rooms.get_room ()
```

Get a room by its id.

Parameters

	room_id	The id of the room to get.

Returns

A dictionary containing the room_id, room_name, room_owner, create_date, is_private, and access_key of the room.

Exceptions

	400	If the room_id parameter is missing.
	404	If the room is not found.

get_rooms()

```
rooms.get_rooms ()
```

Get a list of all public rooms.

Returns

A list of public rooms, each containing the room_id, room_name, room_owner, create_date, is_private, and access_key.

join_private_room()

```
rooms.join_private_room ()
```

Join a private room by its access key.

Parameters

	access_key	The access key of the private room to join.

Returns

A dictionary containing the message of the join action.

Exceptions

	400	If the access_key parameter is missing.
	404	If the room is not found or the user is not found.

join_public_room()

```
rooms.join_public_room ()
```

Join a public room by its id.

Parameters

	room_id	The id of the public room to join.

Returns

A dictionary containing the message of the join action.

Exceptions

	400	If the room_id parameter is missing.
	404	If the room is not found.

leave_room()

```
rooms.leave_room ()
```

Leave a room by its id.

Parameters

	room_id	The id of the room to leave.

Returns

A dictionary containing the message of the leave action.

Exceptions

	400	If the room_id or user_name parameter is missing.
	404	If the room is not found or the user is not in the room.

6.7.2 Variable Documentation

bp

```
rooms.bp = Blueprint('rooms', __name__, url_prefix='/api')
```

6.8 user_activity Namespace Reference

Functions

- None set_user_online (str user_name)

  Set a user as online.
- None update_user_last_seen (str user_name)

  Update the last seen timestamp of a user and set them as online.
- set get_online_users (str user_name)

  Get a set of online users that are in the same rooms as the given user.
- start_activity_tracking ()

  Start a background task to track user activity.

6.8.1 Function Documentation

get_online_users()

```
set user_activity.get_online_users (
            str user_name)
```

Get a set of online users that are in the same rooms as the given user.

Parameters

	user_name	The name of the user to get online users for.

Returns

A set of online users that are in the same rooms as the given user.

set_user_online()

```
None user_activity.set_user_online (
            str user_name)
```

Set a user as online.

If the user is already online, do nothing. Otherwise, add the user to the set of online users and emit a "user_online" event to all rooms that the user is a member of.

Parameters

	user_name	The name of the user to set as online.

Returns

None

start_activity_tracking()

```
user_activity.start_activity_tracking ()
```

Start a background task to track user activity.

This task will check the last seen timestamp of each user every second. If the difference between the current timestamp and the last seen timestamp is greater than the timeout threshold, it will set the user as offline and emit a "user_offline" event to all rooms that the user is a member of.

Returns

None

update_user_last_seen()

```
None user_activity.update_user_last_seen (
            str user_name)
```

Update the last seen timestamp of a user and set them as online.

Parameters

	user_name	The name of the user to update.

Returns

None

6.9 user_rooms Namespace Reference

Functions

- get_user_rooms ()
  Get a list of user rooms for the current user.
- get_user_list ()
  Get a list of users in a room.
- kick_user ()
  Kick a user from a room.

Variables

- bp = Blueprint('user_rooms', __name__, url_prefix='/api')

6.9.1 Function Documentation

get_user_list()

```
user_rooms.get_user_list ()
```

Get a list of users in a room.

Parameters

	room_id	The id of the room to get the user list for.

Returns

A list of users in the room, each containing the user_name, room_id.

get_user_rooms()

```
user_rooms.get_user_rooms ()
```

Get a list of user rooms for the current user.

Returns

A list of user rooms, each containing the user_name, room_id, room_name, room_owner, and is_private.

kick_user()

```
user_rooms.kick_user ()
```

Kick a user from a room.

Parameters

	room_id	The id of the room to kick the user from.
	user_to_kick	The name of the user to kick.

Returns

A dictionary containing the message of the kick action.

6.9.2 Variable Documentation

bp

```
user_rooms.bp = Blueprint('user_rooms', __name__, url_prefix='/api')
```

6.10 users Namespace Reference

Functions

- get_online_users ()

  Get a list of online users for the current user.
- change_password ()

  Change the password of the current user.
- get_user_info ()

  Get information about the current user.

Variables

- bp = Blueprint('users', __name__, url_prefix='/api')

6.10.1 Function Documentation

change_password()

```
users.change_password ()
```

Change the password of the current user.

Parameters

	old_password	(str): The current password of the user.
	new_password	(str): The new password to set for the user.

Returns

A dictionary containing the message of the password change action.

get_online_users()

```
users.get_online_users ()
```

Get a list of online users for the current user.

Returns

A list of online users, each containing the user_name.

get_user_info()

```
users.get_user_info ()
```

Get information about the current user.

Returns

A dictionary containing the count of messages sent by the user, the count of private and public rooms owned by the user, and the count of private and public rooms that the user is a member of.

6.10.2 Variable Documentation

bp

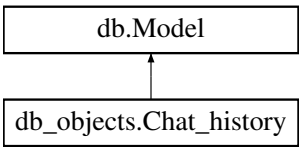
```
users.bp = Blueprint('users', __name__, url_prefix='/api')
```

7 Class Documentation

7.1 db_objects.Chat_history Class Reference

Chat_history model representing a message in a chat room.

Inheritance diagram for db_objects.Chat_history:



Public Member Functions

- to_dict (self)

  Converts the Chat_history instance to a dictionary representation.

Static Public Attributes

- message_id = db.Column(db.Integer, primary_key=True, autoincrement=True)

  Primary key, unique identifier for each message.
- room_id = db.Column(db.Integer, nullable=False)

  Foreign key reference to the chat room containing this message.
- user_name = db.Column(db.String(48), nullable=False)

  Name of the user who sent the message (max 48 characters).
- message = db.Column(db.String(1000), nullable=False)

  The content of the chat message (max 1000 characters).
- message_date = db.Column(db.DateTime, nullable=False, server_default=func.now())

  Timestamp when the message was created.

Static Private Attributes

- str __tablename__ = "chat_history"

7.1.1 Detailed Description

Chat_history model representing a message in a chat room.

7.1.2 Member Function Documentation

to_dict()

```
db_objects.Chat_history.to_dict (
            self)
```

Converts the Chat_history instance to a dictionary representation.

Returns

Dictionary with keys: chat_id, room_id, user_name, message, create_date.

7.1.3 Member Data Documentation

__tablename__

```
str db_objects.Chat_history.__tablename__ = "chat_history"  [static], [private]
```

message

```
db_objects.Chat_history.message = db.Column(db.String(1000), nullable=False)  [static]
```

The content of the chat message (max 1000 characters).

message_date

```
db_objects.Chat_history.message_date = db.Column(db.DateTime, nullable=False, server_default=func.↩
now())  [static]
```

Timestamp when the message was created.

Defaults to the current server time.

message_id

```
db_objects.Chat_history.message_id = db.Column(db.Integer, primary_key=True, autoincrement=True)
[static]
```

Primary key, unique identifier for each message.

Automatically incremented.

room_id

```
db_objects.Chat_history.room_id = db.Column(db.Integer, nullable=False)  [static]
```

Foreign key reference to the chat room containing this message.

user_name

```
db_objects.Chat_history.user_name = db.Column(db.String(48), nullable=False)  [static]
```

Name of the user who sent the message (max 48 characters).

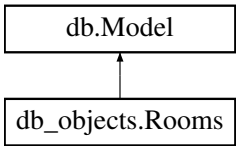
The documentation for this class was generated from the following file:

- db_objects.py

7.2 db_objects.Rooms Class Reference

Database model representing a chat room.

Inheritance diagram for db_objects.Rooms:



Public Member Functions

- to_dict (self)

  Converts the room object to a dictionary representation for serialization.

Static Public Attributes

- room_id = db.Column(db.Integer, primary_key=True, autoincrement=True)

  Unique identifier for the room.
- room_name = db.Column(db.String(64), nullable=False)

  Name of the room.
- room_owner = db.Column(db.String(48), nullable=False)

  Username of the room owner.
- create_date = db.Column(db.DateTime, nullable=False)

  Timestamp when the room was created.
- is_private = db.Column(db.Boolean, nullable=False)

  Flag indicating whether the room is private (True) or public (False).
- access_key = db.Column(db.String(64), nullable=True, unique=True)

  Unique access key for private rooms.

Static Private Attributes

- str __tablename__ = "rooms"

7.2.1 Detailed Description

Database model representing a chat room.

This model defines the structure for storing room information in the database. Rooms can be public or private, and private rooms require an access key for entry.

7.2.2 Member Function Documentation

to_dict()

```
db_objects.Rooms.to_dict (
            self)
```

Converts the room object to a dictionary representation for serialization.

Returns

Dictionary containing all room attributes.

7.2.3 Member Data Documentation

__tablename__

```
str db_objects.Rooms.__tablename__ = "rooms"  [static], [private]
```

access_key

```
db_objects.Rooms.access_key = db.Column(db.String(64), nullable=True, unique=True)  [static]
```

Unique access key for private rooms.

Maximum 64 characters. Optional and unique.

create_date

```
db_objects.Rooms.create_date = db.Column(db.DateTime, nullable=False)  [static]
```

Timestamp when the room was created.

Required.

is_private

```
db_objects.Rooms.is_private = db.Column(db.Boolean, nullable=False)  [static]
```

Flag indicating whether the room is private (True) or public (False).

Required.

room_id

```
db_objects.Rooms.room_id = db.Column(db.Integer, primary_key=True, autoincrement=True)  [static]
```

Unique identifier for the room.

Auto-incremented primary key.

room_name

```
db_objects.Rooms.room_name = db.Column(db.String(64), nullable=False)  [static]
```

Name of the room.

Maximum 64 characters. Required.

room_owner

```
db_objects.Rooms.room_owner = db.Column(db.String(48), nullable=False)  [static]
```

Username of the room owner.

Maximum 48 characters. Required.

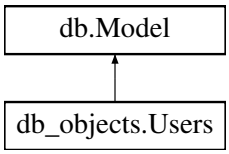
The documentation for this class was generated from the following file:

- db_objects.py

7.3 db_objects.Users Class Reference

Users model representing a user account in the database.

Inheritance diagram for db_objects.Users:



Public Member Functions

- to_dict (self)
  Converts the Users instance to a dictionary representation.

Static Public Attributes

- user_name = db.Column(db.String(48), primary_key=True)
  The unique username, serves as the primary key.
- password = db.Column(db.String(128), nullable=False)
  The password for the user account.

Static Private Attributes

- str __tablename__ = "users"

7.3.1 Detailed Description

Users model representing a user account in the database.

7.3.2 Member Function Documentation

to_dict()

```
db_objects.Users.to_dict (
            self)
```

Converts the Users instance to a dictionary representation.

Returns

Dictionary containing user_name and password.

7.3.3 Member Data Documentation

__tablename__

```
str db_objects.Users.__tablename__ = "users"  [static], [private]
```

password

```
db_objects.Users.password = db.Column(db.String(128), nullable=False)  [static]
```

The password for the user account.

Maximum 128 characters, required field.

user_name

```
db_objects.Users.user_name = db.Column(db.String(48), primary_key=True)  [static]
```

The unique username, serves as the primary key.

Maximum 48 characters.

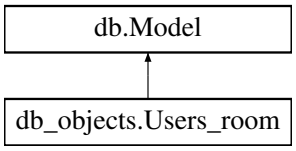
The documentation for this class was generated from the following file:

- db_objects.py

7.4 db_objects.Users_room Class Reference

Database model representing the association between users and rooms.

Inheritance diagram for db_objects.Users_room:



Public Member Functions

- to_dict (self)

  Converts the Users_room instance to a dictionary representation.

Static Public Attributes

- user_name = db.Column(db.String(48), ForeignKey("users.user_name"), primary_key=True)

  Foreign key reference to the user's user_name in the users table.
- room_id = db.Column(db.Integer, ForeignKey("rooms.room_id"), primary_key=True)

  Foreign key reference to the room's ID in the rooms table.

Static Private Attributes

- str __tablename__ = "users_room"

7.4.1 Detailed Description

Database model representing the association between users and rooms.

This model creates a many-to-many relationship between the Users and Rooms tables, allowing multiple users to be associated with multiple rooms.

7.4.2 Member Function Documentation

to_dict()

```
db_objects.Users_room.to_dict (
              self)
```

Converts the Users_room instance to a dictionary representation.

Returns

Dictionary containing user_name and room_id.

7.4.3 Member Data Documentation

__tablename__

```
str db_objects.Users_room.__tablename__ = "users_room"  [static], [private]
```

room_id

```
db_objects.Users_room.room_id = db.Column(db.Integer, ForeignKey("rooms.room_id"), primary_↩
key=True)  [static]
```

Foreign key reference to the room's ID in the rooms table.

Part of the composite primary key.

user_name

```
db_objects.Users_room.user_name = db.Column(db.String(48), ForeignKey("users.user_name"),
primary_key=True)  [static]
```

Foreign key reference to the user's user_name in the users table.

Part of the composite primary key.

The documentation for this class was generated from the following file:

- db_objects.py

8 File Documentation

8.1 app_state.py File Reference

Namespaces

- namespace app_state

Functions

- app_state.update_user_room_maps (int room_id, str user_name, bool remove=False)

  Add or remove a user from user_rooms and room_users maps.
- app_state.remove_room (int room_id)

  Remove a room and clean up all associated user references.

Variables

- app_state.socketio = SocketIO(cors_allowed_origins="*")
- app_state.online_users = set()
- dict app_state.user_last_seen = {}
- dict app_state.room_users = {}
- dict app_state.user_rooms = {}

8.2 db_objects.py File Reference

Classes

- class db_objects.Users

  Users model representing a user account in the database.
- class db_objects.Rooms

  Database model representing a chat room.
- class db_objects.Users_room

  Database model representing the association between users and rooms.
- class db_objects.Chat_history

  Chat_history model representing a message in a chat room.

Namespaces

- namespace db_objects

Variables

- db_objects.db = SQLAlchemy()

8.3 docs/mainpage.dox File Reference

8.4 main.py File Reference

Namespaces

- namespace main

Functions

- main.initialize_room_users ()

  Initialize the room_users and user_rooms maps by iterating over all rooms and their users.
- main.create_app ()

  Create a Flask app with the necessary configurations and routes.

Variables

- main.app = create_app()
- main.debug
- main.True
- main.host
- main.port

8.5 room.py File Reference

Namespaces

- namespace room

Functions

- room.handle_join (data)

  Handle a user joining a room.
- room.handle_leave (data)

  Handle a user leaving a room.
- room.handle_message (data)

  Handle a message sent by a user to a room.
- room.handle_watchdog (data)

  Handle a watchdog event sent by a user.

8.6 routes/authorization.py File Reference

Namespaces

- namespace authorization

Functions

- authorization.register ()
  Register a new user.
- authorization.login ()
  Login an existing user.

Variables

- authorization.bp = Blueprint("auth", __name__, url_prefix="/api")

8.7 routes/chat_history.py File Reference

Namespaces

- namespace chat_history

Functions

- chat_history.get_chat_history ()
  Get the chat history of a room by its id.

Variables

- chat_history.bp = Blueprint('chat_history', __name__, url_prefix='/api')

8.8 routes/rooms.py File Reference

Namespaces

- namespace rooms

Functions

- rooms.get_rooms ()
  Get a list of all public rooms.
- rooms.get_room ()
  Get a room by its id.
- rooms.join_public_room ()
  Join a public room by its id.
- rooms.join_private_room ()
  Join a private room by its access key.
- rooms.leave_room ()
  Leave a room by its id.
- rooms.create_room ()
  Create a room.
- rooms.delete_room ()
  Delete a room by its id.
- rooms.edit_room ()
  Edit a room by its id.

Variables

- rooms.bp = Blueprint('rooms', __name__, url_prefix='/api')

8.9 routes/user_rooms.py File Reference

Namespaces

- namespace user_rooms

Functions

- user_rooms.get_user_rooms ()

  Get a list of user rooms for the current user.
- user_rooms.get_user_list ()

  Get a list of users in a room.
- user_rooms.kick_user ()

  Kick a user from a room.

Variables

- user_rooms.bp = Blueprint('user_rooms', __name__, url_prefix='/api')

8.10 routes/users.py File Reference

Namespaces

- namespace users

Functions

- users.get_online_users ()

  Get a list of online users for the current user.
- users.change_password ()

  Change the password of the current user.
- users.get_user_info ()

  Get information about the current user.

Variables

- users.bp = Blueprint('users', __name__, url_prefix='/api')

8.11 user_activity.py File Reference

Namespaces

- namespace user_activity

Functions

- None user_activity.set_user_online (str user_name)

  Set a user as online.
- None user_activity.update_user_last_seen (str user_name)

  Update the last seen timestamp of a user and set them as online.
- set user_activity.get_online_users (str user_name)

  Get a set of online users that are in the same rooms as the given user.
- user_activity.start_activity_tracking ()

  Start a background task to track user activity.

Index