Innopolis Library Media Keeper

Here you can find all the documentation concerning the inner work of the ILMK.

scroller.py

File with class for interacting with user, generating 'menu' messages.

class Scroller

1. def **init** (*self*, state, list)
. **Description**: Initializer for object of class.
. **Input**: state - what type of menu is this; list - list with items to scroll through
. **Output**: none

2. def **update** (*self*, list_update)
. **Description:** updates the list from database
. **Input:** list_update - new list
. **Output:** none

3. def **convert_to_emoji** (state)
. **Description:** converts states to emojis
. **Input:** state - state to convert
. **Output:** emoji, depending on state

4. def **create_message** (*self*):
. **Description:** creates a message, depending on state, defined during initialization process
. **Input:** none
. **Output:** created message

5. def **create_keyboard** (*self*):
   - **Description:** creates a keyboard (buttons under message) according to a type of request
   - **Input:** none
   - **Output:** created keyboard

6. def **increase_cursor** (*self*):
   - **Description:** increases cursor used in def create_keyboard(self)
   - **Input:** none
   - **Output:** none

7. def **decrease_cursor** (*self*):
   - **Description:** decreases cursor used in def create_keyboard(self)
   - **Input:** none
   - **Output:** none

8. def **approve_request** (*self*, bot, update)
   - **Description:** approves the request for registration of user
   - **Input:** bot - bot object; update - log of bot-user interaction
   - **Output:** none

9. def **reject_request** (*self*, bot, update)
   - **Description:** rejects the request for registration of user
   - **Input:** bot - bot object; update - log of bot-user interaction
   - **Output:** none

10. def **book_media** (*self*, bot, update)
    - **Description:** books items
    - **Input:** bot - bot object; update - log of bot-user interaction
    - **Output:** none

11. def **accept_booking_request** (*self*, bot, update)
    - **Description:** accepts booking request
    - **Input:** bot - bot object; update - log of bot-user interaction
    - **Output:** none

12. def **reject_booking_request** (*self*, bot, update)
    - **Description:** rejects booking request
    - **Input:** bot - bot object; update - log of bot-user interaction
    - **Output:** none

13. def **generate_expiry_date** (*self*, media, patron, issue_date)
    - **Description:** generates expiry date based on type of media and user
    - **Input:** media - media item that is booked; patron - person that books media; issue_date - date of booking
    - **Output:** expiry date

user.py

File that performs connection with database. Configuration values are hidden in config.py

class Patron

The class makes user interaction easier, holds user's values and allows to work with database (get/set values).

1. def **init** (*self*)
    - **Description:** Initializer for object of class.
    - **Input:** none
    - **Output:** none

2. def **set_user** (*self*, json_line)
    - **Description:** parses the JSON-line record from MySQL database into class attributes.
    - **Input:** json_line - JSON-line with information about user
    - **Output:** none

3. def **set_request** (*self*, json_line)
    - **Description:** parses the JSON-line record from MySQL database into class attributes. Request has another attributes, not like User.
    - **Input:** json_line - JSON-line with information about request

. **Output:** none

4. def **insert_in_base** (*self*)
. **Description:** inserts user into database. Used for moving request to user.
. **Input:** none
. **Output:** none

5. def **update** (*self*)
. **Description:** updates user info. If user attributes are changed, they need to be uploaded to the database.
. **Input:** none
. **Output:** none

6. def **add_request** (*self*)
. **Description:** adds formed request to a specific request table.
. **Input:** none
. **Output:** none

7. def **make_media_request** (*self*, media_id)
. **Description:** adds request for taking media from library.
. **Input:** media_id - media that is taken
. **Output:** none

8. def **find** (*self*, telegram_id)
. **Description:** finds current user in table using telegram id and creates an object of this user.
. **Input:** telegram_id - Telegram ID of user
. **Output:** none

class ItemCard

The class makes media interaction easier, holds media's values and allows to work with database (get/set values).

1. def **init** (*self*)
   - **Description:** Initializer for object of class.
   - **Input:** none
   - **Output:** none

2. def **set_item** (*self*, json_line)
   - **Description:** parses the JSON-line record from MySQL database into class attributes.
   - **Input:** json_line - JSON-line with information about item
   - **Output:** none

3. def **update** (*self*)
   - **Description:** updates media info. If attributes are changed, they need to be uploaded to the database.
   - **Input:** none
   - **Output:** none

4. def **find** (*self*, media_id)
   - **Description:** Finds current media in table using media id and creates an object of this item.
   - **Input:** media_id
   - **Output:** none

class BookingRequest

This class works with booking requests and allows to work with database (get/set values).

1. def **init** (*self*)
   - **Description:** Initializer for object of class.
   - **Input:** none
   - **Output:** none

2. def **set_request** (*self*, json_line)
   - **Description:** moves data from JSON-line database to values in class attributes.
   - **Input:** json_line - JSON-line with data
   - **Output:** none

telebot.py

This file contains conversation handlers that help to get user's information and search items.

Handlers - specific functions that trigger actions by commands. Command can be /command or just text. Handlers and commands can also be connected by ConversationHandlers (used with /enroll command below)

1. def **search_functions** (bot, update)
   - **Description:** query handler. Each time user sends inline buttonbot gets a callback query. This handler chooses the appropriate way to run further.
   - **Input:** bot - bot object, update - log of bot-user interaction
   - **Output:** none

2. def **all_numbers** (input_string)
   - **Description:** checks if all the elements of input are digits
   - **Input:** input_string - string with information
   - **Output:** whether all the elements of input are digits

3. def **ask_name** (bot, update)
   - **Description:** handler that starts registration process. Handles from /enroll button. Here user's profile is starting filling
   - **Input:** bot - bot object; update - log of bot-user interaction
   - **Output:** next state

4. def **ask_phone** (bot, update)
   - **Description:** handler that asks for phone.
   - **Input:** bot - bot object; update - log of bot-user interaction
   - **Output:** next state

5. def **ask_address** (bot, update)
   - **Description:** handler that asks for address.
   - **Input:** bot - bot object; update - log of bot-user interaction
   - **Output:** next state

6. def **ask_faculty** (bot, update)
   - **Description:** handler that asks if the user is faculty member.
   - **Input:** bot - bot object; update - log of bot-user interaction

- **Output:** moving to buttons. After buttons moves to end_of_registration.

7. def **end_of_registration** (bot, update)
- **Description:** ends the registraton. If everything is correct, send request to database.
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** end of registry_handler

8. def **get_list** (table):
- **Description:** gets the list of all records from the table. Used for update.
- **Input:** table - name of table to update
- **Output:** list with all the records from the table

9. def **create_request_card** (bot, update)
- **Description:** is called when commands are getting called for the first time. Creates a request card.
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

10. def **create_media_card** (bot, update)
- **Description:** is called when commands are getting called for the first time. Creates a media card.
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

11. def **issue_media** (bot, update)
- **Description:** is called when commands are getting called for the first time. Creates an issue media card.
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

12. def **edit_request_card** (bot, update)
- **Description:** edits request card
- **Input:** bot - bot object; update - log of bot-user interaction
- **Output:** none

13. def **edit_media_card** (bot, update)
. **Description:** edits media card
. **Input:** bot - bot object; update - log of bot-user interaction
. **Output:** none

14. def **edit_issue_media** (bot, update)
. **Description:** edits issue meda card
. **Input:** bot - bot object; update - log of bot-user interaction
. **Output:** none

15. def **librarian_authentication** (user_id)
. **Description:** checks if user is a librarian or not
. **Input:** user_id - Telegram ID of user that we check
. **Output:** whether user is a librarian or not

16. register_conversation
. This is a connector which connects command /enroll and many handlers. Each handler asks user for single parameter (phone, address and etc). register_conversation calls handlers one-by-one.