Feat 103: Acting bot users

Issue: https://github.com/Team-TAU/tau/issues/103

Tables

New table to support this. Can be `ChatUser` for table `chat_users` or `BotUser`:

- `bot_users` as model `BotUser`
  - `access_token` : String
  - `refresh_token` : String
  - `username` : String
  - `user_id` : String or Number

Bot Authentication

Expose an additional Twitch callback endpoint `/twitch-bot-callback` with handler `def process_twitch_bot_callback_view`.

- Endpoint to copy: https://github.com/Team-TAU/tau/blob/main/tau/urls.py#L81
- Handler to copy: https://github.com/Team-TAU/tau/blob/main/tau/core/views.py#L256-L259

Changes to above mentioned functionality:

- Instead of Constance use a model backed by the table `bot_users`, i.e. `BotUser` to be able to perform an operation as follows to save data received from the Twitch callback success:

```
BotUser.create!(
  access_token: request[:access_token],
  refresh_token: request[:refresh_token],
  username: request[:username],
  user_id: request[:user_id]
)
```

Developers can express intent to perform actions on behalf of their bot by providing an extra header, e.g. `x-tau-on-behalf-of: techydrrroid` alongside their existing `Authorization: Token {token}` header.

Using bot actors for Twitch passthrough endpoints

A method can be used before evaluating which user to perform an action on behalf of:

```
def my_helix_passthrough_handler
  if not request.user.is_authenticated:
    return JsonResponse({'error': 'You must be logged into access this endpoint.'})

  actor_user = get_bot_user_from_request(request) || request.user
  # Alternatively some kind of middleware that can evaluate request.user based on the auth token plus tau header

  TwitchClient.get('/path/to/whatever', headers: {
    'Authentication': "Bearer #{actor_user.token}"
  })
```

Before performing actions, if we have a bot user, we can use that user's token instead. We can check the headers and use the `x-tau-on-behalf-of` header as a secondary criteria.

```
def get_bot_user_from_request(request)
  raise Unauthorized unless request.user.is_authenticated
  return nil unless request[:headers]['x-tau-on-behalf-of'].present?

  bot_username = request[:headers]['x-tau-on-behalf-of']
  bot_user = BotUser.find_by(username: bot_username)

  raise NotFound unless bot_user.present?

  return bot_user
end
```

Notes:

- header acts as secondary criteria combined with existing auth checks
- if no bot found for the TAU header, throw because you should only try to make requests for bots that are authorized (rather than fallback to your own user)

Using bot actors for IRC

TBD (I'm not familiar with the IRC protocols)