eap

Last edited by **Marta Filipa Martins Tavares da Cruz** 3 weeks ago

EAP: Architecture Specification and Prototype

Connectify aims to revolutionize online social interactions by establishing a new paradigm in social networking that prioritizes user empowerment, trust, and community. We envision a platform where individuals can engage authentically, build meaningful relationships, and share their lives in a safe environment, ultimately creating a more connected and supportive digital world. Our commitment to user autonomy and privacy not only sets us apart but also inspires a movement towards more responsible and user-focused online experiences.

A7: Web Resources Specification

This artifact defines and documents the web resources and their functionalities, ensuring clear API specifications, while also specifying the appropriate permissions for different user roles.

1. Overview

Modules	Description
M01: Authentication	Web resources associated with user authentication. Includes the following system features: login/logout, registration, credencial recovery., view and edit individual user profile.
M02: Users	Web resources associated with users. Includes the following system features: view user profiles, creating, editing and deleting user.
M03: Posts and Comments	Web resources associated with posts and comments. Includes the following system features: create, edit, delete, and list posts and save posts for later viewing ; create, edit, delete, and view comments on posts and reactions to them.
M04: Friendship Requests	Web resources associated with friendship Requests. Includes, sending, accepting or rejecting friendship requests
M05: Groups	Web resources associated with group functionalities. Includes the following system features: create groups and manage group memberships; allow users to send and manage requests to join groups; approve or reject group membership requests.
M06: Notifications	Web resources associated with viewing users notifications
M07: Search	Web resources associated with users search, including exact match search and full text search for all types of search a user can do (user, group, post and comment)
M08: Administration	Web resources associated with administrative tasks. Includes the following system features: manage user accounts (view, block, or delete users), and update static pages.

2. Permissions

Identifier	Name	Description
GST	Guest	Unauthenticated User
USR	User	Authenticated users
OWN	Owner	Users owners of posts, comments, profiles, groups
ADM	Administrator	System administrators

3. OpenAPI Specification

Link to OpenAPI file

```
openapi: 3.0.0
info:
  version: '1.0'
  title: 'Connectify Web API'
  description: 'Web Resources Specification for Connectify API'

servers:
  - url: https://lbaw2453.lbaw.fe.up.pt
    description: 'Production server'
```

```yaml
externalDocs:
  description: Find more info here.
  url: https://gitlab.up.pt/lbaw/lbaw2425/lbaw2453/-/wikis/eap

tags:
  - name: 'M01: Authentication'
  - name: 'M02: Users'
  - name: 'M03: Posts and Comments'
  - name: 'M04: Friendship Requests'
  - name: 'M05: Groups'
  - name: 'M06: Notifications'
  - name: 'M07: Search'
  - name: 'M08: Administration'



paths:

#################################### Authentication ####################################

  /login:
    get:
      operationId: R101
      summary: 'R101: Login Form'
      description: 'Provide login form. Access: GST'
      tags:
        - 'M01: Authentication'
      responses:
        '200':
          description: 'Ok. Show log-in UI'
    post:
      operationId: R102
      summary: 'R102: Login Action'
      description: 'Processes the login form submission. Access: GST'
      tags:
        - 'M01: Authentication'
      requestBody:
        required: true
        content:
          application/x-www-form-urlencoded:
            schema:
              type: object
              properties:
                email:
                  type: string
                password:
                  type: string
              required:
                - email
                - password

      responses:
        '302':
          description: 'Redirect after processing the login credentials.'
          headers:
            Location:
              schema:
                type: string
              examples:
                Success:
                  description: 'Successful authentication. Redirect to user profile.'
                  value: '/users/{id}'
                Error:
                  description: 'Failed authentication. Redirect to login form.'
                  value: '/login'
```

```yaml
/logout:
  post:
    operationId: R103
    summary: 'R103: Logout Action'
    description: 'Logout the current authenticated used. Access: USR, ADM'
    tags:
      - 'M01: Authentication'
    responses:
      '302':
        description: 'Redirect after processing logout.'
        headers:
          Location:
            schema:
              type: string
            examples:
              302Success:
                description: 'Successful logout. Redirect to login form.'
                value: '/login'


/register:
  get:
    operationId: R104
    summary: 'R104: Register Form'
    description: 'Provide new user registration form. Access: GST'
    tags:
      - 'M01: Authentication'
    responses:
      '200':
        description: 'Ok. Show sign-up UI'

  post:
    operationId: R105
    summary: 'R105: Register Action'
    description: 'Processes the new user registration form submission. Access: GST'
    tags:
      - 'M01: Authentication'

    requestBody:
      required: true
      content:
        application/x-www-form-urlencoded:
          schema:
            type: object
            properties:
              username:
                type: string
              email:
                type: string
                format: email
              password:
                type: string
              confirmPassword:
                type: string
            required:
              - username
              - email
              - password
              - confirmPassword

    responses:
      '302':
        description: 'Redirect after processing the new user information.'
        headers:
          Location:
            schema:
              type: string
            examples:
              302Success:
                description: 'Successful authentication. Redirect to user profile.'
                value: '/users/{id}'
              302Failure:
                description: 'Failed authentication. Redirect to login form.'
                value: '/login'


  /forgot-password:
    get:
      operationId: R106
      summary: 'R106: Forgot Password Form'
      description: 'Displays a form to request a password recovery email. Access: GST'
      tags:
        - 'M01: Authentication'
      responses:
        '200':
          description: 'Ok. Show forgot password form.'
    post:
      operationId: R107
      summary: 'R107: Send Password Recovery Email'
      description: 'Processes the form submission to send a password recovery code. Access: GST'
      tags:
        - 'M01: Authentication'
      requestBody:
        required: true
        content:
          application/x-www-form-urlencoded:
            schema:
              type: object
              properties:
                email:
                  type: string
                  format: email
                  description: 'Registered user email address'
              required:
                - email
      responses:
        '302':
          description: 'Redirect after processing the request.'
          headers:
            Location:
              schema:
                type: string
              examples:
                Success:
                  description: 'Redirect to verify code page.'
                  value: '/verify-code'
                Error:
                  description: 'Redirect back with an error message.'
                  value: '/forgot-password'

  /verify-code:
    get:
      operationId: R108
      summary: 'R108: Verify Code Form'
      description: 'Displays the form to verify the password recovery code. Access: GST'
      tags:
        - 'M01: Authentication'
      responses:
        '200':
          description: 'Ok. Show verify code form.'
    post:
      operationId: R109
      summary: 'R109: Verify Recovery Code'
```

```yaml
      description: 'Processes the submitted recovery code and redirects appropriately. Access: GST'
      tags:
        - 'M01: Authentication'
      requestBody:
        required: true
        content:
          application/x-www-form-urlencoded:
            schema:
              type: object
              properties:
                email:
                  type: string
                  format: email
                  description: 'Email associated with recovery code'
                code:
                  type: integer
                  description: '6-digit recovery code'
              required:
                - email
                - code
      responses:
        '302':
          description: 'Redirect after verifying the recovery code.'
          headers:
            Location:
              schema:
                type: string
              examples:
                302Success:
                  description: 'Code verified successfully. Redirect to reset password page.'
                  value: '/reset-password'
                302Error:
                  description: 'Failed verification. Redirect back with error.'
                  value: '/verify-code'

  /reset-password:
    get:
      operationId: R110
      summary: 'R110: Reset Password Form'
      description: 'Displays the form to reset the user password. Access: GST'
      tags:
        - 'M01: Authentication'
      responses:
        '200':
          description: 'Ok. Show reset password form.'
    post:
      operationId: R111
      summary: 'R111: Reset Password Action'
      description: 'Processes the form submission to reset the user password. Access: GST'
      tags:
        - 'M01: Authentication'
      requestBody:
        required: true
        content:
          application/x-www-form-urlencoded:
            schema:
              type: object
              properties:
                email:
                  type: string
                  format: email
                  description: 'User email address'
                password:
                  type: string
                  format: password
                  description: 'New password'
                password_confirmation:
                  type: string
                  format: password
                  description: 'Confirmation of the new password'
              required:
                - email
                - password
                - password_confirmation
      responses:
        '302':
          description: 'Redirect after resetting the password.'
          headers:
            Location:
              schema:
                type: string
              examples:
                302Success:
                  description: 'Password reset successfully. Redirect to login page.'
                  value: '/login'
                302Error:
                  description: 'Failed to reset password. Redirect back with error.'
                  value: '/reset-password'

  /auth/google:
    get:
      operationId: R112
      summary: R112:Redirect to Google OAuth
      description: Redirects the user to the Google OAuth authorization page.
      responses:
        '302':
          description: Redirects to Google OAuth login page.
      tags:
        - Authentication
  /auth/google/call-back:
    get:
      summary: Google OAuth callback
      description: Handles the callback from Google OAuth after user login/authorization.
      responses:
        '200':
          description: Successfully authenticated and redirected to the home page.
        '400':
          description: Invalid OAuth callback data.
        '500':
          description: Server error or authentication failed.
      tags:
        - Authentication

##############################################Users#############################################

  /user/{userId}:
    get:
      operationId: R201
      summary: 'R201: View user profile'
      description: 'Show the profile for an individual user, Access: GST, USR, OWN, ADM'
      tags:
        - 'M02: Users'
      parameters:
        - name: userId
          in: path
          required: true
          description: The ID of the user
          schema:
            type: string
      responses:
        '200':
          description: 'OK. Show user profile'
          content:
            application/json:
```

```
              schema:
                type: object
                properties:
                  id:
                    type: string
                  username:
                    type: string
                  email:
                    type: string
                  isPublic:
                    type: boolean
        '404':
          description: User not logged in or user not found


  /user/editProfile:
    get:
      operationId: R202
      summary: 'R202: Edit user profile page'
      description: 'Provide user profile edit form. Access: OWN, ADM'
      tags:
        - 'M02: Users'
      responses:
        '200':
          description: 'Ok. Show user profile edit form'
        '401':
          description: 'Unauthorized, user not allowed to edit profile.'
          headers:
            Location:
              schema:
                type: string
              examples:
                401Unauthorized:
                  description: 'Unauthorized, redirect to current user profile.'
                  value: '/user/{userId}'

    post:
      operationId: R203
      summary: 'R203: Edit user profile'
      description: 'Edit user profile. Access: USR, OWN'
      tags:
        - 'M02: Users'
      requestBody:
        required: true
        content:
          application/x-www-form-urlencoded:
            schema:
              type: object
              properties:
                username:
                  type: string
                email:
                  type: string
                  format: email
                profilePicture:
                  type: string
                  format: uri
                isPublic:
                  type: boolean
                password:
                  type: string
                  format: password
                confirmPassword:
                  type: string
                  format: password
              required:
                - username
                - email
                - isPublic
      responses:
        '302':
          description: 'Redirect after editing user information.'
          headers:
            Location:
              schema:
                type: string
              examples:
                302Success:
                  description: 'Successful edit. Redirect to user profile.'
                  value: '/user/{userId}'
                302Failure:
                  description: 'Failed. Redirect back.'
                  value: '/user/editProfile'


  /user/deleteProfile:
    post:
      operationId: R204
      summary: 'R204: Delete user profile'
      description: 'Delete user profile. Access: OWN, ADM'
      tags:
        - 'M02: Users'
      requestBody:
        required: true
        content:
          application/x-www-form-urlencoded:
            schema:
              type: object
              properties:
                userId:
                  type: integer
              required:
                - userId
      responses:
        '302':
          description: 'Redirect after deleting user information.'
          headers:
            Location:
              schema:
                type: string
                example:
                  302Success:
                    description: 'Successful delete. Redirect to login page.'
                  302Failure:
                    description: 'Failed. Redirect back.'


  /home:

    get:
      operationId: R205
      summary: 'R205: View home page'
      description: 'Show home page. Access: USR, ADM'
      tags:
        - 'M02: Users'

      responses:
        '200':
          description: 'OK. Show home page for an individual user'
        '302':
          description: 'Redirect if not unauthorized.'
          headers:
            Location:
              schema:
                type: string
              example:
                  302Success:
                    description: 'You need to login first. Redirect to login page.'
                    value: '/login'

  /user/friends/{userId}:
    get:
      operationId: R206
      summary: 'R206: Get user friends list'
      description: 'Retrieve the list of friends for a specific user. Access: GST, USR, ADM'
      tags:
        - 'M02: Users'
      parameters:
        - name: userId
          in: path
          required: true
          description: 'The ID of the user whose friends list is being requested.'
          schema:
            type: string
      responses:
        '200':
          description: 'OK. Return the list of friends.'
          content:
            application/json:
              schema:
                type: array
                items:
                  type: object
                  properties:
                    friendId:
                      type: string
                    friendUsername:
                      type: string
        '404':
          description: 'User not found or no friends available.'

##########################POSTS AND COMMENTS######################

  /post/create:
    post:
      operationId: R301
      summary: 'R301: Create a new post action'
      tags:
        - 'M03: Posts and Comments'
      requestBody:
        content:
          application/x-www-form-urlencoded:
            schema:
              type: object
              properties:
                content:
                  type: string
                imageUrls:
                  type: array
                  items:
                    type: string
                  description: List of image URLs
                isPublic:
                  type: boolean
              required:
                - content
      responses:
        '201':
          description: 'Post created successfully'
          headers:
            Location:
```

```yaml
              schema:
                type: string
              examples:
                302Success:
                  description: 'Post created. Redirect to post page.'
                  value: '/post/{postId}'
                302Error:
                  description: 'Failed to create post. Redirect back.'
                  value: '/post/create'
        '400':
          description: 'Bad request, invalid input data'
        '500':
          description: 'Internal server error'

  /post/{postId}:
      get:
        operationId: R302
        summary: 'R302: View post'
        description: 'Show the post. Access: GST, USR, OWN, ADM'
        tags:
          - 'M03: Posts and Comments'
        parameters:
          - name: postId
            in: path
            required: true
            description: The ID of the post
            schema:
              type: string
        responses:
          '200':
            description: Post retrieved
            content:
              application/json:
                schema:
                  type: object
                  properties:
                    id:
                      type: string
                    content:
                      type: string
                    imageUrls:
                      type: array
                      items:
                        type: string
                    postDate:
                      type: string
                      format: date-time
                    isPublic:
                      type: boolean
                    userId:
                      type: string
          '404':
            description: Post not found

  /post/delete:
    delete:
      operationId: R303
      summary: 'R303: Delete post action'
      description: 'Delete a post. Access: OWN, ADM'
      tags:
        - 'M03: Posts and Comments'
      parameters:
        - name: postId
          in: query
          required: true
          description: 'ID of the post to delete'
          schema:
```

```
          type: integer
      responses:
        '200':
          description: Post deleted successfully
        '403':
          description: Forbidden, user not allowed to delete post
        '401':
          description: Unauthorized, user not allowed to delete post
          headers:
            Location:
              schema:
                type: string
              example: '/users/{userId}'
  # Redirecionar para o perfil do usuário

  /post/edit:
    put:
      operationId: R304
      summary: 'R304: Edit post. Access: OWN, ADM'
      description: 'Edit a post'
      tags:
        - 'M03: Posts and Comments'
      requestBody:
        required: true
        content:
          application/x-www-form-urlencoded:
            schema:
              type: object
              properties:
                postId:
                  type: integer
                content:
                  type: string
                imageUrls:
                  type: array
                  items:
                    type: string
                  description: List of image URLs
                isPublic:
                  type: boolean
              required:
                - postId
                - content
      responses:
        '200':
          description: Post edited successfully
        '403':
          description: Forbidden, user not allowed to edit post

  /post/save/{postId}:
    post:
      operationId: R305
      summary: 'R305: Save a post'
      description: 'Save a post for later reference. Access: USR'
      tags:
        - 'M03: Posts and Comments'
      parameters:
        - name: postId
          in: path
          required: true
          description: The ID of the post to be saved
          schema:
            type: string
      responses:
        '200':
          description: Post saved successfully
        '403':
```

```
        description: Forbidden, user not allowed to save this post
      '401':
        description: Unauthorized, user not logged in

/post/unsave/{postId}:
  post:
    operationId: R306
    summary: 'R306: Unsave a post'
    description: 'Remove a post from saved posts. Access: USR'
    tags:
      - 'M03: Posts and Comments'
    parameters:
      - name: postId
        in: path
        required: true
        description: The ID of the post to be unsaved
        schema:
          type: string
    responses:
      '200':
        description: Post unsaved successfully
      '403':
        description: Forbidden, user not allowed to unsave this post
      '401':
        description: Unauthorized, user not logged in

/post/saved:
  get:
    operationId: R307
    summary: 'R307: View saved posts'
    description: 'Retrieve all posts saved by the authenticated user. Access: USR'
    tags:
      - 'M03: Posts and Comments'
    responses:
      '200':
        description: List of saved posts retrieved successfully
        content:
          application/json:
            schema:
              type: array
              items:
                type: object
                properties:
                  id:
                    type: string
                  content:
                    type: string
                  imageUrls:
                    type: array
                    items:
                      type: string
                  postDate:
                    type: string
                    format: date-time
                  userId:
                    type: string
      '401':
        description: Unauthorized, user not logged in


/post/{postId}/comment:
  post:
    operationId: R308
    summary: 'R308: Add a comment to a post'
    description: 'Create a new comment for a specific post. Access: USR'
    tags:
      - 'M03: Posts and Comments'
    parameters:
      - name: postId
        in: path
        required: true
        description: The ID of the post to comment on
        schema:
          type: string
    requestBody:
      required: true
      content:
        application/x-www-form-urlencoded:
          schema:
            type: object
            properties:
              comment:
                type: string
            required:
              - comment
    responses:
      '201':
        description: Comment added successfully
        content:
          application/json:
            schema:
              type: object
              properties:
                message:
                  type: string
                comment:
                  type: object
                  properties:
                    id:
                      type: string
                    content:
                      type: string
                    userId:
                      type: string
                    postId:
                      type: string
      '400':
        description: Bad request, invalid input data
      '401':
        description: Unauthorized, user not logged in

/comments/{commentId}:
  put:
    operationId: R309
    summary: 'R309: Edit a comment'
    description: 'Edit an existing comment. Access: OWN, ADM'
    tags:
      - 'M03: Posts and Comments'
    parameters:
      - name: commentId
        in: path
        required: true
        description: The ID of the comment to be edited
        schema:
          type: string
    requestBody:
      required: true
      content:
        application/x-www-form-urlencoded:
          schema:
            type: object
            properties:
              content:
                type: string
```

```yaml
              required:
                - content
        responses:
          '200':
            description: Comment edited successfully
            content:
              application/json:
                schema:
                  type: object
                  properties:
                    success:
                      type: boolean
                    content:
                      type: string
          '403':
            description: Forbidden, user not allowed to edit comment
          '404':
            description: Comment not found

      delete:
        operationId: R310
        summary: 'R310: Delete a comment'
        description: 'Delete an existing comment. Access: OWN, ADM'
        tags:
          - 'M03: Posts and Comments'
        parameters:
          - name: commentId
            in: path
            required: true
            description: The ID of the comment to be deleted
            schema:
              type: string
        responses:
          '200':
            description: Comment deleted successfully
          '403':
            description: Forbidden, user not allowed to delete comment
          '404':
            description: Comment not found

    /post/{postId}/comments:
      get:
        operationId: R311
        summary: 'R311: Get comments for a post'
        description: 'Retrieve all comments for a specific post. Access: GST, USR, OWN, ADM'
        tags:
          - 'M03: Posts and Comments'
        parameters:
          - name: postId
            in: path
            required: true
            description: The ID of the post
            schema:
              type: string
        responses:
          '200':
            description: Comments retrieved successfully
            content:
              application/json:
                schema:
                  type: array
                  items:
                    type: object
                    properties:
                      id:
                        type: string
                      content:
```

```
                  type: string
                userId:
                  type: string
                postId:
                  type: string
        '404':
          description: Post not found

  /comment/{commentId}/reaction-count:
    get:
      operationId: R312
      summary: 'R312: Get reaction count for a comment'
      description: 'Retrieve the number of reactions for a specific comment. Access: GST, USR'
      tags:
        - 'M03: Posts and Comments'
      parameters:
        - name: commentId
          in: path
          required: true
          description: The ID of the comment
          schema:
            type: string
      responses:
        '200':
          description: Reaction count retrieved successfully
          content:
            application/json:
              schema:
                type: object
                properties:
                  reactionCount:
                    type: integer
        '404':
          description: Comment not found


#################################### Friendship Requests ####################################


  /friendship/request/send:
    post:
      operationId: R401
      summary: "R401: Send friendship request"
      description: "Allows a user to send a friendship request to another user. Access: USR"
      tags:
        - "M04: Friendship"
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                receiver_id:
                  type: integer
                  description: "ID of the user who will receive the friendship request."
      responses:
        '200':
          description: "Friendship request sent successfully."
          content:
            application/json:
              example:
                message: "Friendship request sent successfully."
        '400':
          description: "Invalid parameters or request already exists."
          content:
            application/json:
              example:
                error: "Friendship request already sent."
  /friendship/request/{id}/accept:
    post:
      operationId: R402
      summary: "R402: Accept friendship request"
      description: "Allows accepting a pending friendship request. Access: USR"
      tags:
        - "M04: Friendship"
      parameters:
        - name: id
          in: path
          required: true
          description: "ID of the friendship request."
          schema:
            type: integer
      responses:
        '200':
          description: "Friendship request accepted successfully."
          content:
            application/json:
              example:
                message: "Friendship added successfully."
        '400':
          description: "Invalid or not found request."
          content:
            application/json:
              example:
                error: "Invalid friendship request."
  /friendship/request/{id}/decline:
    post:
      operationId: R403
      summary: "R403: Decline friendship request"
      description: "Allows declining a pending friendship request. Access: USR"
      tags:
        - "M04: Friendship"
      parameters:
        - name: id
          in: path
          required: true
          description: "ID of the friendship request."
          schema:
            type: integer
      responses:
        '200':
          description: "Friendship request declined successfully."
          content:
            application/json:
              example:
                message: "Friendship request declined successfully."
        '400':
          description: "Invalid or not found request."
          content:
            application/json:
              example:
                error: "Invalid friendship request."



#################################### Groups ####################################

  /group/create:
    get:
      operationId: R501
      summary: R501:Create a new group
      responses:
        '200':
```

```yaml
          description: Group creation form displayed successfully.

  /group:
    post:
      operationId: R502
      summary: R502:Store a new group

      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                group_name:
                  type: string
                  description: Name of the group.
                  example: College Friends
                description:
                  type: string
                  description: Description of the group.
                  example: Group to share college memories.
                is_public:
                  type: boolean
                  description: Indicates if the group is public or private.
                  example: true
      responses:
        '201':
          description: Group created successfully.

  /group/{id}:
    get:
      operationId: R503
      summary: R503:Get details of a specific group

      parameters:
        - name: id
          in: path
          required: true
          schema:
            type: integer
            example: 1
      responses:
        '200':
          description: Group data retrieved successfully.

  /groups/{groupId}/join:
    post:
      operationId: R504
      summary: R504:Join a public group

      parameters:
        - name: groupId
          in: path
          required: true
          schema:
            type: integer
            example: 1
      responses:
        '200':
          description: Successfully joined the group.
        '400':
          description: Error joining the group.

  /groups/{groupId}/leave:
    get:
      operationId: R505
      summary: R505:Leave a group
      parameters:
        - name: groupId
          in: path
          required: true
          schema:
            type: integer
            example: 1
      responses:
        '200':
          description: Successfully left the group.
        '400':
          description: Error leaving the group.

  /group/{groupId}/members:
    get:
      operationId: R506
      summary: R506:View group members
      parameters:
        - name: groupId
          in: path
          required: true
          schema:
            type: integer
            example: 1
      responses:
        '200':
          description: Group member list retrieved successfully.

  /group/{groupId}/addFriend:
    post:
      operationId: R507
      summary: R507:Add a friend to the group
      parameters:
        - name: groupId
          in: path
          required: true
          schema:
            type: integer
            example: 1
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                friend_id:
                  type: integer
                  description: ID of the friend to be added.
                  example: 2
      responses:
        '200':
          description: Friend added to the group successfully.
        '400':
          description: Error adding friend to the group.

  /group/{groupId}/remove/{userId}:
    delete:
      operationId: R508
      summary: R508:Remove a member from the group
      parameters:
        - name: groupId
          in: path
          required: true
          schema:
            type: integer
            example: 1
        - name: userId
          in: path
          required: true
          schema:
            type: integer
            example: 2
      responses:
        '200':
          description: Member removed successfully.
        '400':
          description: Error removing member from the group.

  /posts/{post}/remove-from-group:
    patch:
      operationId: R509
      summary: R509:Remove a post from a group
      parameters:
        - name: post
          in: path
          required: true
          schema:
            type: integer
            example: 10
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                groupId:
                  type: integer
                  description: ID of the group from which the post will be removed.
                  example: 1
      responses:
        '200':
          description: Post removed from the group successfully.
        '400':
          description: Error removing the post from the group.


  ###############################Search##########################

  /api/search/exactMatch/:
    get:
      operationId: R701
      summary: 'R701: Search for exact match'
      description: 'Perform an exact match search for a given term. Access: USR'
      tags:
        - 'M07: Search'
      parameters:
        - name: query
          in: query
          required: true
          description: The exact term to search for.
          schema:
            type: string
        - name: type
          in: query
          required: true
          description: The type of search (e.g., "user", "group", "post", "comment"). Determines what type of entity to
          schema:
            type: string
            enum: ['user', 'group', 'post', 'comment']
      responses:
        '200':
          description: Results retrieved successfully. Returns a list of matching results.
          content:
            application/json:
              schema:
                type: object
                properties:
                  results:
                    type: array
                    items:
                      type: object
                      properties:
                        id:
                          type: string
                        type:
                          type: string
                        name:
                          type: string
                        snippet:
                          type: string
                  total:
                    type: integer
              example:
                results:
                  - id: "123"
                    type: "user"
                    name: "John Doe"
                    snippet: "John's profile description snippet."
                total: 1
        '400':
          description: Invalid search parameters (e.g., missing or invalid `query` or `type`).
          content:
            application/json:
              schema:
                type: object
                properties:
                  error:
                    type: string
              example:
                error: "Invalid query parameter. 'type' must be one of ['user', 'group', 'post', 'comment']."
        '404':
          description: No results found for the given search criteria.
          content:
            application/json:
              schema:
                type: object
                properties:
                  message:
                    type: string
              example:
                message: "No matching results found."

  /api/search/fulltext/:
    get:
      operationId: R702
      summary: 'R702: Search for full text'
      description: 'Search for documents or posts containing the specified keywords. Access: USR'
      tags:
        - 'M07: Search'
      parameters:
        - name: query
          in: query
          required: true
          description: The keywords or phrase to search for.
          schema:
            type: string
        - name: type
          in: query
          required: true
          description: The type of search (e.g., "user", "group", "post", "comment"). Determines what type of entity to
          schema:
            type: string
            enum: ['user', 'group', 'post', 'comment']
      responses:
        '200':
          description: Results retrieved successfully. Returns a list of matching results.
          content:
            application/json:
              schema:
                type: object
                properties:
                  results:
                    type: array
                    items:
                      type: object
                      properties:
                        id:
                          type: string
                        type:
                          type: string
                        name:
                          type: string
                        snippet:
                          type: string
              example:
                results:
                  - id: "456"
                    type: "post"
                    name: "Exploring APIs"
                    snippet: "This post discusses how to interact with APIs in a full-text search."
        '400':
          description: Invalid Parameters (e.g., missing or invalid `query` or `type`).
          content:
            application/json:
              schema:
                type: object
                properties:
                  error:
                    type: string
              example:
                error: "Invalid query parameter. 'type' must be one of ['user', 'group', 'post', 'comment']."


##### Administrator #################

  /admin/users/search:
    get:
      operationId: R801
      summary: 'R801: Search user accounts'
      description: 'Search for user accounts using filters like username, email, or roles. Access: ADM.'
      tags:
        - 'M08: Administration'
      parameters:
        - name: username
          in: query
          required: false
          schema:
            type: string
          description: Filter by username (partial match allowed)
        - name: email
          in: query
          required: false
          schema:
            type: string
          description: Filter by email (partial match allowed)
        - name: role
          in: query
          required: false
          schema:
            type: string
            enum: [user, moderator, admin]
          description: Filter by user role
        - name: isActive
          in: query
          required: false
          schema:
            type: boolean
          description: Filter by account status (active/inactive)
      responses:
        '200':
          description: Search results
          content:
            application/json:
              schema:
                type: array
                items:
                  type: object
                  properties:
                    id:
                      type: integer
                      description: User ID
                    username:
                      type: string
                      description: Username
                    email:
                      type: string
                      description: User email
                    role:
                      type: string
                      description: User role
                    isActive:
                      type: boolean
                      description: Account active status
        '403':
          description: Forbidden - User does not have administrator access
        '500':
          description: Internal server error

  /admin/users/{userId}:
    get:
      operationId: R802
      summary: 'R802: View user account details'
      description: 'Retrieve detailed information about a specific user account. Access: ADM.'
      tags:
        - 'M08: Administration'
      parameters:
        - name: userId
          in: path
          required: true
          schema:
            type: integer
          description: The ID of the user to retrieve
      responses:
        '200':
          description: User account details retrieved successfully
          content:
            application/json:
              schema:
                type: object
                properties:
                  id:
                    type: integer
                    description: User ID
                  username:
                    type: string
                  email:
                    type: string
                  role:
                    type: string
                    enum: [user, moderator, admin]
                  isActive:
                    type: boolean
                  createdAt:
                    type: string
                    format: date-time
                    description: Account creation date
                  updatedAt:
                    type: string
                    format: date-time
                    description: Last update date
        '403':
          description: Forbidden - User does not have administrator access
        '404':
          description: User account not found
        '500':
          description: Internal server error

  /admin/users/create:
    post:
      operationId: R803
      summary: 'R803: Create a new user account'
      description: 'Allow an administrator to create a new user account with specific attributes. Access: ADM.'
      tags:
        - 'M08: Administration'
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                username:
                  type: string
                email:
                  type: string
                  format: email
                password:
                  type: string
                  format: password
                role:
                  type: string
                  enum: [user, moderator, admin]
                isActive:
                  type: boolean
              required:
                - username
                - email
                - password
      responses:
        '201':
          description: User account created successfully
          content:
            application/json:
              schema:
                type: object
                properties:
                  id:
                    type: integer
                    description: ID of the newly created user
```

```yaml
                  username:
                    type: string
                  email:
                    type: string
                  role:
                    type: string
                  isActive:
                    type: boolean
        '400':
          description: Bad request - Validation error
        '403':
          description: Forbidden - User does not have administrator access
        '500':
          description: Internal server error

  /admin/users/edit:
    put:
      operationId: R804
      summary: 'R804: Edit user account details'
      description: 'Allow an administrator to update details of a user account. Access: ADM.'
      tags:
        - 'M08: Administration'
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                userId:
                  type: integer
                  description: ID of the user to edit
                username:
                  type: string
                  description: Updated username
                email:
                  type: string
                  format: email
                  description: Updated email address
                password:
                  type: string
                  format: password
                  description: Updated password
                role:
                  type: string
                  enum: [user, moderator, admin]
                  description: Updated role of the user
                isActive:
                  type: boolean
                  description: Updated account status
              required:
                - userId
      responses:
        '200':
          description: User account updated successfully
        '400':
          description: Bad request - Validation error
        '403':
          description: Forbidden - User does not have administrator access
        '404':
          description: User account not found
        '500':
          description: Internal server error

  /user/{userId}/promote:
    post:
      operationId: R805
      summary: R805:Promote user to administrator
      description: Grants administrator privileges to the specified user.
      parameters:
        - name: userId
          in: path
          required: true
          schema:
            type: integer
          description: The ID of the user to promote.
      responses:
        200:
          description: User promoted successfully.
          content:
            application/json:
              schema:
                type: object
                properties:
                  message:
                    type: string
                    example: User promoted to administrator.
```

A8: Vertical prototype

This artifact involves implementing the application's prototype with the high priority features. Its purpose is to validate the proposed architecture and help the team become familiarizes with the tools and technologies used in the development of the project.

1. Implemented Features

1.1. Implemented User Stories

User Story reference	Name	Priority	Description
US11	Sign-in	High	As an Unauthenticated User, I want to authenticate into the system so that I can access public or restricted content.
US12	Sign-up	High	As an Unauthenticated User, I want to register an account so that I can start using the platform.
US13	View Public Timeline	High	As an Unauthenticated User, I want to see the public timeline so that I can view posts shared by others.
US14	View Public Profiles	High	As an Unauthenticated User, I want to view other users' public profiles so that I can see their interests and posts.
US15	Exact Match Search	High	As an Unauthenticated User, I want to perform an exact match search for public users, posts, or comments so that I can quickly find specific content or individuals that match my search criteria without sifting through unrelated results.
US16	Full Text Search	High	As an Unauthenticated User, I want to utilize full-text search to explore public users, posts, or comments so that I can find relevant content based on keywords, phrases, or topics of interest, allowing for a more comprehensive search experience.
US21	View Personalized Timeline	High	As an Authenticated User, I want to see my personalized timeline so that I can view posts related to my interests and posts from my friends.
US22	View User Profiles	High	As an Authenticated User, I want to view user profiles so that I can see their posts and interests.
US23	Exact Match Search	High	As an Authenticated User, I want to perform an exact match search for public users, posts, groups or comments so that I can quickly find specific content or individuals that match my search criteria without sifting through unrelated results.
US24	Full Text Search	High	As an Authenticated User, I want to utilize full-text search to explore public users, posts, groups or comments so that I can find relevant content based on keywords, phrases, or topics of interest, allowing for a more comprehensive search experience.

User Story reference	Name	Priority	Description
US25	Create Post	High	As an Authenticated User, I want to create posts so that I can share my own content with friends or publicly.
US26	View My Own Profile	High	As an Authenticated User, I want to view my own profile so that I can see my personal details and posts.
US27	Edit My Own Profile	High	As an Authenticated User, I want to edit my own profile so that I can update my personal information.
US28	Log out	High	As an Authenticated User, I want to log out so that I can end my session and secure my account.
US31	Delete Post	High	As a Post Author, I want to delete my post so that it is no longer visible to other users.
US32	Edit Post	High	As a Post Author, I want to edit my post so that I can update the information.
US71	Administer User Accounts	High	As an Administrator, I want to search, view, edit, and create user accounts so that I can manage users and assist them with account-related issues.
US72	Manage User Content	Medium	As an Administrator, I want to manage and remove inappropriate content so that I can maintain community guidelines.

1.2. Implemented Web Resources

Module M01: Authentication

Web Resource Reference	URL
R101: Login Form	GET/ login
R102: Login Action	POST/ login
R103: Logout Action	POST/ logout
R104: Register Form	GET/ register
R105: Register Action	POST/ register

Module M02: Users

Web Resource Reference	URL
R201: View user profile	GET /users/{id}
R202: Edit user profile page	GET /user/editProfile
R203: Edit user profile	POST /user/editProfile
R205: View home page	GET /home

Module M03: Posts and Comments

Web Resource Reference	URL
R301: Create a new post action	POST /post/create
R302: View post	GET /post/{postId}
R303: Delete post action	DELETE /post/delete
R304: Edit post	put /post/edit

Module M07: Search

Web Resource Reference	URL
R701: Search for exact match	GET /api/search/exactMatch

Web Resource Reference	URL
R702: Search for full text	GET /api/search/fulltext

Module M08: Administration

Web Resource Reference	URL
R801: Search user accounts	GET /admin/users/search
R802: View user account details	GET /admin/users/{userId}
R803: Create a new user account	POST /admin/users/create
R804: Edit user account details	PUT /admin/users/edit

2. Prototype

The prototype Docker image is available at GitLab's Registry Container and can be run with:

- `docker run -d --name lbaw2453 -p 8001:80 gitlab.up.pt:5050/lbaw/lbaw2425/lbaw2453`

Acess http://localhost:8001 to use the application.

Credentials to test features:

- admin user: alice@example.com/securepassword1
- regular user: bob@example.com/securepassword2

Link to the source code: https://gitlab.up.pt/lbaw/lbaw2425/lbaw2453

Revision history

Changes made to the first submission: 23/12/2024: Updated OpenApi

---

GROUP2453, 04/11/2024

- Group member 1 Maria Vieira, up202204802@g.uporto.pt
- Group member 2 Marta Cruz [editor], up202205028@g.uporto.pt
- Group member 3 Bruna Scafutto, up202402704@g.uporto.pt
- Group member 4 Duarte Marques, up202204973@g.uporto.pt