ASP.NET based REST API

The main task is to create a REST based API that consists of 5 major endpoints. The API should provide data about users and allow the editing of user profiles.

The API should be served by an ASP.NET Core application. The data has to be stored in SQL Server.

This backend system serves a hypothetical mobile application where you can view users, retrieve their profiles and edit your own profile. For simplicity, we do not deal with authentication.

Users

The application only manages users.

We store the following data about a user:

- First name
- Last name
- E-mail address
- Phone number
- E-mail address visibility
- Phone number visibility
- Profile picture

The visibility of the email address and phone number indicates whether the user wants these data to be publicly visible. This will be important at the endpoint that lists the users. These settings can have two possible values: visible or hidden. The default setting is hidden.

The format of the data is irrelevant. The following restrictions apply to the first name, last name, email address and phone number:

- They are mandatory
- They cannot be longer than 200 character
- Should be stored as strings

The following restrictions apply to profile picture:

- Only images can be accepted (jpg, png, etc.)
- Do not store files larger than 10 MB
- Images has to be resized automatically before storage: neither dimension can be longer than 1024 pixels. The resizing method must preserve the aspect ratio of the image.

The maximum number of users is 100.

One of the users has a special role: they represent the (logged in) user of the mobile application that communicates with the API. Without an authentication system, for simplicity, we designate the first user in the natural order of the records as the “logged in user”.

Endpoints

The endpoints should be designed in REST style. The communication format should be JSON.

The summary of the 5 major endpoints:

Name	Description
List of users	Lists all users in the system
User profile	Provides detailed information about a user, selected by a unique identifier
Own profile	Detailed information about our own (the logged in user’s) profile
Own profile update	Updates our own profile
Own profile picture upload	Uploads our own profile picture

User listing endpoint

This endpoint returns a list of all users. Pagination is not necessary.

The following data should be returned about the users:

- First name
- Last name
- Profile picture URL
- Unique identifier

The profile picture should be available through an URL in binary format, as an `image/jpeg` content type.

The identifier should uniquely identify the user through all endpoints.

User profile endpoint

This endpoint returns details about a selected user. The selection is made by the user’s unique identifier that is passed as a parameter.

Inbound parameters:

- The user's unique identifier

Returned data:
- First name
- Last name
- Profile picture URL
- Unique identifier
- E-mail address (based on visibility)
- Phone number (based on visibility)

The e-mail address and telephone number should only be returned if the user enabled visibility on the relevant data. If the visibility is set to hidden, then the relevant field should return a `null` value.

Own profile endpoint

This endpoint returns the owner's (logged in user) details.

This endpoint is similar to the previous user details endpoint, with the following differences:
- Instead of a unique identifier, a fix name should be used that identifies the logged in user's own profile. Either in the endpoint's name, or a constant parameter (like "me" of "myself") should be used.
- Because the endpoint represents the logged in user's own profile, both the e-mail and phone number should be returned, regardless of their visibility settings.
- The following data should be returned in addition:
    - Visibility setting of the e-mail address
    - Visibility setting of the phone number

In summary, all data should be returned about the logged in user.

Own profile update endpoint

This endpoint updates the logged in user's own profile. When processing the update, please observe the restrictions discussed at the description of the users. The selection of the user record should be done similarly to the previous, own profile details endpoint.

The inbound data received by the endpoint:
- First name
- Last name
- E-mail address
- Phone number
- E-mail address visibility
- Phone number visibility

The returned data upon success: the same data returned by the previous, own user details endpoint.

The returned data upon failure (for example, validation error): a list containing the errors in a readable way).

Own profile picture upload

This endpoint receives an uploaded image file and stores it as the profile picture of the logged in user. Please observe the previously discussed restrictions on the profile picture. The selection of the user should be done similarly to the method described at the own user details endpoint.

The endpoint should process a single uploaded file. The data transfer should be handled as a traditional multi-part form post (`Content-Type: multipart/form-data`) instead of JSON.

Other endpoints

Recommended for testing purposes. Irrelevant of the task's main purpose.

Creation of new user

Deletion of a user by id

Upload a profile picture to a user's profile by id