User Microservice Task Summary

Data

For creating the User entity, I used the specified properties (first name, last name, nickname, password, email and country) as fields. User data is stored in its own table in an in-memory database.

As there weren't any requirements for password specifications, I assumed the password field would be holding an encoded pass and so I have tested it like this with my unit tests. For this reason, I didn't specify a size restriction on the password string field as other factors would have to be considered that are out of the scope of this task. And so, the other User fields just have sensible string length restrictions appropriate for the data they hold.

For exposing the User through the web response, I took the assumption that the password would not be included. Therefore, there is a service that handles returning a user response without the password. I decided to put that operation in its own service because if there would be any future requirements to refine the user response from its original user entity this could also be done in the service.

API

CRUD operations are exposed through the user API controllers. For create and update, the API simply takes a user entity. For delete it is done by the user ID. For reading a user I assumed the requirement meant that users should be filtered by a certain single user property at a time specified in the request. Therefore, I have added individual endpoints to get users, by each user property separately.

So, I took the assumption that there is not a requirement to return users by multiple parameters in one request. E.g. returning users based on 'country' and 'last name'. And instead users are returned based on a request with a single specified user property e.g. 'first name'.