LAM Project 2021

Federico Montori, Luca Sciullo, Lorenzo Gigli

April 2021

Rules

In this document we describe one possible project for the exam of "Laboratorio di applicazioni mobili" course. The project is individual, no groups allowed. Each student can choose to develop the project proposed here (valid until Febraury 2022) or suggest something else based on his/her personal interests. In the latter case, project proposals should be submitted via e-mail to Dr. Luca Sciullo (luca.sciullo@unibo.it), with a brief description of the application goals, contents and a list of **all** the features that you propose to implement. The following project description contains a minimum set of requirements for the application. Students are strongly encouraged to expand the track, by adding new features to the application, and/or further customizing the contents. Any question regarding the projects have to be asked to Dr. Luca Sciullo via e-mail (`luca.sciullo@unibo.it`). Every student is required to produce:

1. *A mobile application*, there is no constraint on the language and the framework used: it can be native or hybrid, but it cannot be a Web Application (*i.e.* it has to run natively on the device).

2. *A project report*, which is a document that describes the application produced, focusing on the workflow and the design choices. In particular, the Report should be named `SURNAME_NAME.pdf` and contain:

   - Your name, surname, email and matriculation number.
   - Overview of the application with screenshots.
   - Implementation details on how you chose to implement the functionalities

   A good report is probably between 8 and 12 pages. Less than 5 pages is probably bad, more than 15 is probably too much. The quality of the report WILL be part of the evaluation.

3. *A presentation*, which consists in a set of slides that will help you in the project discussion. They should contain a brief recap of the report alongside with screenshots of the application. We suggest to produce max. 10 slides since the discussion time is approximately 10 minutes.

The **CODE** of the mobile application and the **REPORT** have to be uploaded exclusively on the Virtuale platform in the dedicated section[1] (there are 6 deadlines throughout the year). They must be enclosed in a single .zip file named `SURNAME_NAME.zip`. If the archive is too big for Virtuale, you can remove the directory "build" from your code (Android only). The SLIDES, instead, must be brought along the day of the oral examination.

Do not forget that the oral examination consists also in a theoretical part, therefore the knowledge of the topics discussed in class (both iOS and Android) is required.

The Project: "Tracking My Pantry"

The mobile application is designed for those who want to keep track of the groceries they buy through their barcode and build up a collaborative database of barcodes that can be used by the community. This means that each developed app will make use of both a local database and a unique remote knowledge base that will be updated and queried. The student **must** implement at least the following two features to pass the exam, bearing in mind that implementing **only** these will result in the minimal grade (18). In other words, implementing additional features is strongly encouraged and will result in a higher grade (they will be evaluated in terms of complexity and utility). If you want to ask in advance information about additional features, you can use the Forum on Virtuale: `https://virtuale.unibo.it/mod/forum/view.php?id=553649`. The features are as follows:

Feature 1: Update the barcode knowledge base

Products are registered by their barcode in a global remote database, which exposes a WebService interface (details about it in the next paragraph). A barcode is a sequence of digits that uniquely identifies a product. Every time the user buys a product, he/she must perform the following operations:

1. ask the WebService the details of the product by passing the barcode (you can simply make the user type in the digits). This returns a list of products. We will call this operation `GET PRODUCTS BY BARCODE`.
2. If the product is in the list, then the user selects it and notifies the WebServer about it. We will call this operation `POST PRODUCT PREFERENCE`.
3. If the product is not in the list (or the list is empty), the user must type in the details of the product and send it to the server along with the barcode, this creates a new entry in the remote database. We will call this operation `POST PRODUCT DETAILS`.

[1] `https://virtuale.unibo.it/course/view.php?id=18950`

The WebService answers to HTTP REST calls and the full documentation about them is in the appendix of this document. You can use the HTTP client library/framework that you prefer to accomplish this task (*e.g.* OKHttp, Volley, Retrofit, ...). In addition to these operations, the user must (only once) register to the WebService, by sending its username, email and password, so that the server records them. We will call this operation `REGISTER`. Then, every time the user uses the application, he or she needs to login to the WebService by using the email and the password. The WebService will reply with an authentication Token (a uniquely generated string) that will be used by the user in every subsequent call to prove his or her authentication.

Feature 2: Keep track of the Groceries bought

Once the operations against the WebService are done, the user has obtained a product with a barcode and its details. The products that the user buys have to be stored in a **local** database, which will act as your virtual pantry. In particular:

- The user should be able to insert new products in the pantry.
- The user should be able to delete products from the pantry as they get consumed.
- The user should be able to browse the full content of the pantry (this can be done in any way).

Additional features

The two features described above yield a very basic applications. The students are, again, encouraged to add their features on top of their implementation. Here are some suggestions (there is literally no limit):

- You can scan the barcode with the camera instead of typing it in.
- You can implement a search to look for specific products inside your pantry.
- You can browse your pantry with filters, by adding parameters to the product details (only in their local version).
- You can implement a daily notification that notifies you of some important products that you ran out of.
- You can implement a service that periodically scans your database and reminds you of some old entries that maybe you forgot to delete.
- You can also insert the expiry date of a product and set up a reminder on your calendar or a scheduled notification.
- You can even register the location where such products were bought and show it on the map; you may want to get back there someday...

Appendix: The WebService documentation

The full updated documentation of the API can be found at `https://lam21.modron.network/explorer/#/`, which is a Swagger endpoint. You can try out the calls and get familiar with them before implementing them into your application.

Operation `REGISTER` This operation is a POST call with the following endpoint: `https://lam21.modron.network/users`

The POST requires no header and has the following JSON parameters (example):

```
{
  "username": "stradivarius",
  "email": "federico.montori2@unibo.it",
  "password": "mypassword"
}
```

A successful response to the previous will return the code 201 and the following body:

```
{
  "id": "ckna223pm00002bn055hp3amj",
  "username": "stradivarius",
  "email": "federico.montori2@unibo.it",
  "password": "mypassword-ENCRYPTED",
  "createdAt": "2021-04-09T08:36:34.762Z",
  "updatedAt": "2021-04-09T08:36:34.763Z"
}
```

Bear in mind that you can register only ONCE with the same credentials, otherwise you'll get a 500. From here, you'll need only your email and password to login. The username will be used to anonymize you (so maybe don't use your name and surname if you don't want them exposed).

Operation `LOGIN` This operation is a POST call with the following endpoint: `https://lam21.modron.network/auth/login` The POST requires no header and has the following JSON parameters (example):

```
{
  "email": "federico.montori2@unibo.it",
  "password": "mypassword"
}
```

A successful response to the previous will return the code 201 and the following body:

```
{
  "accessToken": "eyJhbGhiOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
      eyJzdWIiOiJja25hMjIzcG0wMDAwMmJuMDU1aHAzYW1qIiwiaWF0IjoxNjE3OTU3Nzg4LCJleHAiOjE2MTg1NjI1ODh9
      .Ma1c4CarADZBpNRjxquf5QifRoqwKl15JZ92kw1esuY"
}
```


This is the **access token** that you need to include in *every* subsequent call to authenticate yourself. This token expires in 7 days, but, obviously, you can always request a new one which will replace the old one. Bear in mind that this access token needs to be used in the header of every call that you perform afterwards to prove that you are logged in. The header has the following form:

```
Authorization: Bearer <accessToken>
```

where `<accessToken>` is the access token that you received.

Operation `GET PRODUCTS BY BARCODE` This operation is a GET call with the following endpoint `https://lam21.modron.network/products?barcode=0000000000000` (this is obviously an example that uses 0000000000000 as a barcode). The GET requires the `<accessToken>` in the header.

A successful response to the previous call will return the code 200 and the following body (example with no products found):


```
{
  "products": [],
  "token": "ckna2l2js00172bn0sqgwkhxn"
}
```


(example with one product found):


```
{
  "products": [
    {
      "id": "ckna2s1wc00302bn0465hkboy",
      "name": "Testing Object",
      "description": "Hi, if you read this you have managed to obtain the
           first object! Congrats!",
      "barcode": "0000000000000",
      "userId": "ckna223pm00002bn055hp3amj",
      "test": false,
      "createdAt": "2021-04-09T08:56:45.468Z",
      "updatedAt": "2021-04-09T08:56:45.469Z"
    }
  ],
  "token": "ckna2l2js00172bn0sqgwkhxn"
}
```


If no products were found (or if what you found is different from what you want to insert), then you may need to insert a new one using `POST PRODUCT DETAILS`, otherwise, you may want to rate the products that you obtained by

giving a high rating to the one that corresponds to your product (and maybe a negative rating to others) via `POST PRODUCT PREFERENCE`. In both cases, you will need the `token` returned by `GET PRODUCTS BY BARCODE` (see example reply above). Let's call this the `sessionToken`, which will label the POST call as a reply to this GET.

Operation `POST PRODUCT DETAILS` This operation is a POST call with the following endpoint `https://lam21.modron.network/products`. The POST requires the `<accessToken>` in the header and has the following JSON parameters (example):

```
{
  "token": "ckna2l2js00172bn0sqgwkhxn",
  "name": "Testing Object",
  "description": "Hi, if you read this you have managed to obtain the
      first object! Congrats!",
  "barcode": "0000000000000",
  "test": false
}
```

Notice that this request has a `token` field. This is the `sessionToken` obtained from the last `GET PRODUCTS BY BARCODE`.

A successful response to the previous call will return the code 201 and the following body:

```
{
  "id": "ckna2s1wc00302bn0465hkboy",
  "name": "Testing Object",
  "description": "Hi, if you read this you have managed to obtain the
      first object! Congrats!",
  "barcode": "0000000000000",
  "userId": "ckna223pm00002bn055hp3amj",
  "test": false,
  "createdAt": "2021-04-09T08:56:45.468Z",
  "updatedAt": "2021-04-09T08:56:45.469Z"
}
```

At this point your flow is over. If you want this product to disappear instantly (because you are only testing this call), just set the "test" field to *true*.

Operation `POST PRODUCT PREFERENCE` This operation is a POST call with the following endpoint `https://lam21.modron.network/votes`. The POST requires the `<accessToken>` in the header and has the following JSON parameters (example):

```
{
  "token": "ckna2l2js00172bn0sqgwkhxn",
```

```
  "rating": 1,
  "productId": "ckna2s1wc00302bn0465hkboy"
}
```

Notice that this request has a `token` field. This is the `sessionToken` obtained from the last `GET PRODUCTS BY BARCODE`.

A successful response to the previous call will return the code 201 and the following body:

```
{
  "id": "ckna30be300462bn0oql9rzg0",
  "rating": 1,
  "productId": "ckna2s1wc00302bn0465hkboy",
  "userId": "ckna223pm00002bn055hp3amj",
  "createdAt": "2021-04-09T09:03:11.019Z",
  "updatedAt": "2021-04-09T09:03:11.020Z"
}
```
