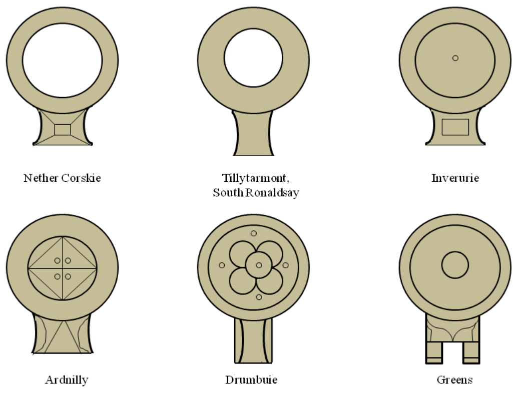
T7 - Application Development

T-DEV-800

PictsManager

A photo app to take, store, tag and handle pictures



1.5.4

PictsManager

binary name:	pictsManager_*$AcademicYear_$GroupNumber*.zip
delivery method:	Moodle
language:	whatever works

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.

You are about to build a mobile application and a server which takes and stores pictures. It will allow the user to perform a variety of actions on the database.

You can re-use the structure of the application you developed during a previous project, but this time you will have to focus on more advanced topics, including storing and parsing data efficiently.

Tools and Techniques

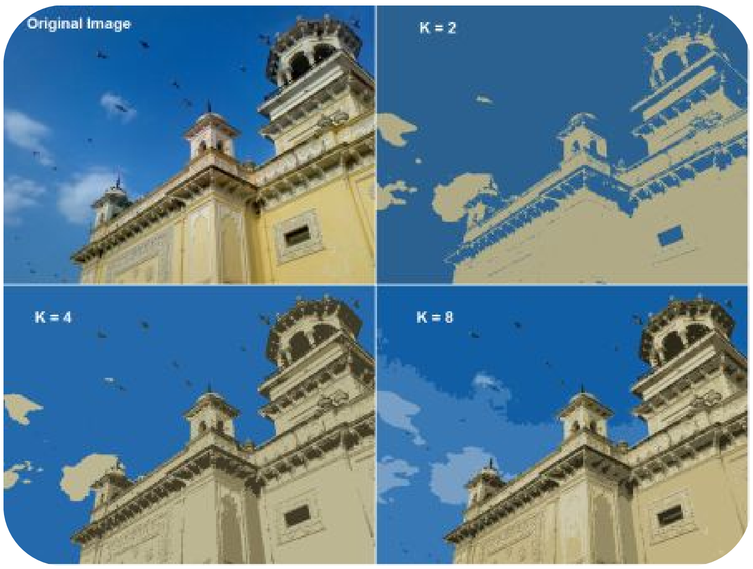
+Image Compression

Pictures you take from your phone camera need to be compressed before they are sent to the back. You have to find a compromise between performance and quality.

Do all views require the same compression ratio?

A pretty basic way to compress images consists in reducing the number of colors it contains.Three steps are needed to do this:

1. **read** the image and **extract** the colors of each pixel,
2. **cluster** these colors, and **replace** each color of a given cluster by the mean color of this cluster,
3. **index** the means of the cluster, and **create** the compressed image.



Many algorithms allow you to do this clustering, pick the one you that suits you best!

Machine Learning libraries are welcome, but you have to document the process and must be able to explain the complexity of your solution to your clients.

ML libraries are allowed, but automated image compression tools are prohibited

+UI/UX

UX is fundamental for any application. You should find guidelines for design development, or you could create and justify yours.

Accessibility is important. By integrating accessibility features and services, you could improve your app's usability, particularly for users with disabilities.

You must respect the application lifecycle

+Database models

You need to properly store extra contextual information with your photos, in order to perform various tasks (tags, timetables, sorts, ...) within reasonable time.

Choice of DB management might strongly impact your performances.

+Quick algorithms

Mastering algorithmic complexity is the key for designing really efficient algorithms. Assume our computer can handle 10^8 operations per second, and we want to sort a table of size 10^7:

- A selection sort needs 10^6 seconds (~300 hours) to perform 10^{14} operations
- A merge sort needs less than 1 second to perform 7×10^7 operations

Some available resources may drive you further than the scope of this project, but it might be as useful as pleasant for those of you that want to dive deep into it.

Specifications

+Information storage

Mobile application only stores pictures' names and associated ID. Pictures, metadatas and other informations are stored in the distant back server and can be retrieved and/or updated with specific requests.

Database's design, both model and implementation, is up to you. Just make sure that the necessary information can be stored and retrieved efficiently.

No need for your back server to have an admin for completing this project. However, it might be useful for dealing with the database and you are free to do it yourself or use any third-party solution.

+Permissions and security

User needs login and password to connect to the terminal, with section open until manual disconnection.

- User adding a picture|album becomes its only owner. This cannot be shared, transferred or resigned.
- Ownership of an album means ownership of all pictures that belong to it.
- A picture can't be part of several albums from different owners.
- Only the owner of a picture|album can delete it.
- Only the owner of an album can add a picture in it.
- Only the owner of a picture|album can give or remove (reading) access to other users.
- Modifying access to an album will automatically modify access to all pictures in it.
- Users can only see pictures|albums|informations they own or have been granted access to.
- Any research a user performs can only reach pictures|albums (s)he has access to.

This is not a project about security, so you can keep it minimal. However, feel free to secure it better if you like it.

Requirements

+HMI

You must be compliant with OS Guidelines on the differents functionalities:

1. **sign in**: users connect the terminal to the server and identify oneself with a password
2. **camera**: users take a photo, name it, save it or discard it
3. **gallery**: users browse many albums|pictures, select one picture for viewing it
4. **search**: users search among pictures, according to several criteria (tags, date, ...)
5. **view**: users view a picture, modify its metadata (grant or remove accesses), delete the picture.

Feel free either to split or combine several functionalities on one or more screens for achieving those goals.

+Other Requirements

1. Only operations performed directly in the terminal are taking pictures and compressing them.
2. Other tasks are performed on the server (where the database is located), which returns an answer.
3. Pictures must be compressed enough for consuming less time transfer and space storage.
4. Pictures' quality must stay good enough for a human user to be able to recognize them.
5. Search function must be implemented explicitly (no external library is allowed)
6. Search part must be surrounded with testing, to handle exceptions and evaluate computation time.
7. Atomicity of requests is mandatory: cancel operation and inform the terminal if anything fails.
8. Architecture must be designed with scalability in mind.

+Delivery

We require you to containerize each of the services with docker and orchestrate them with **docker-compose**.

The construction of your mobile applcation and your back must be automated and run within the container. The docker project will compile, test, package and deploy your program, including the database.

As usual, the code will be duly documented *(preferably following the JavaDoc syntax)* and you must provide Software Architecture Specification and Software Qualification.

To fully demonstrate all the functionalities of your app during the keynote, you should fill it with **many** albums and pictures from **various** users.

+Bonuses

You can improve this project in many ways, including:

- adding a real authentication protocol
- increasing your applcation's security
- building an admin view for the server
- adding formal specification documents
- conducting a full functional testing process
- providing automated tag suggestion based on ML