EGNSS4CAP IOS APPLICATION SOURCE CODE OVERVIEW

CONTENT

General Overview....2

System Structure....3

Global Structure....3

Taking Photos....4

Application Settings....4

Map and Paths....5

Login....5

GENERAL OVERVIEW

The application is developed as a regular application for the iPhone Operating System (iOS) in the Swift language build inside XCode IDE. The UI of application is built using the Storyboard. The application is written for minimal compatibility with iOS 13.6. The source code is structured according to the XCode project standard descripted at https://developer.apple.com/library/archive/featuredarticles/XcodeConcepts/Concept-Projects.html.

The source code tries to follow these group structure: Each screen of the application has its own controller. All controllers are located in separate files in the root group. All background processes, some foreground processes and all application data models are located in the Model group.

The files of source code can be divided based on these parts of the application:

- system structure,
- global structure,
- taking photos,
- application settings,
- map and paths,
- login.

SYSTEM STRUCTURE

This section includes all configuration files necessary for building, distributing and running the application. These files are defined according to the XCode requirement and do not define the application processes themselves or the application data structure itself.

GLOBAL STRUCTURE

This section describes the pieces of code, which are primally shared throughout the application.

The application uses User Default and Core Data as a local persistent storage. There is one Core Data model for the whole application defined in the PhotoModel.xcdatamodeld, which include following entities:

- PersitPhoto
  - The captured photo as a binary picture data with all metadata.
- PTPath
  - The metadata of an entire recorded path.
- PTPoint
  - The metadata of a point forming a recorded path.

Data about the user and application settings are stored in the User Default storage. This storage is accessed indirectly through classes UserStorage and SEStorage.

All UI screens are defined in Main.storyboad.

Description of files:

- **AppDelegate.swift**
  - The application delegate object that manages application's shared behaviors. The application delegate is effectively the root object of whole app, and it works in conjunction with UIApplication to manage some interactions with the system.
  - It loads and holds the Core Data PhotoModel persistent container.
- **SceneDelegate.swift**
  - It defines UIWindowSceneDelegate object to manage the life cycle of one instance of this application's user interface. The window scene delegate conforms to the UISceneDelegate property, and it is used to receive notifications when its scene connects to the application, enters the foreground, and so on.
- **extensions.swift**
  - It contains extensions of all system classes.
- **Model/PersistStorage.swift**
  - The base class registering all derived classes for access to User Default storage.
- **Model/ UserStorage.swift**
  - The class for accessing logged user data in the User Default storage.
- **Model/Setting/SEStorage.swift**
  - The class for accessing application settings in the User Default storage.
- **Model/ Util.swift**
  - The class containing general helper functions.
- **Model/Weak.swift**
  - The class holder for weak reference.
- **Model/DB/DB.swift**

    - The class that provides a private managed object context and a shared main managed object context for PhotoModel.
    -
- **Model/DB/PersistPhoto+CoreDataClass.swift**
    - It defines additional convenience methods inside the PersistPhoto managed object class.
- **Assets.xcassets**
    - It contains all pictures used in application.

TAKING PHOTOS

The code in this section defines how to take, edit, view and send photos.

Thera are included these screens:

- photo overview,
- photo detail,
- camera.

Description of files:

- **PhotosTableViewController.swift, PhotoTableViewCell.swift**
    - The classes to view captured photos in a list structure and to delete these photos.
- **PhotoDetailViewCotroller.swift**
    - The controller to view a captured photo in detail and to send this photo.
- **CameraViewController.swift**
    - The controller to display the camera preview and current location data on the screen, and to take a geotag photo from the camera.
    - It uses PhotoDataController class.
- **Model/PhotoDataController.swift**
    - The class that defines background processes for reading location data that are designed to be used to take a geotag photo.
- **Model/ConvexHull/…**
    - The classes for computing convex hull centroid over location data.

APPLICATION SETTINGS

The code in this section defines how to manage application settings.

There is included Settings screen.

Description of files:

- **SettingsViewController.swift**
    - The controller to view current settings in a list structure and to manipulate these settings.
- **Model/Setting/…**
    - Helper classes to display setting items and handling events on those items.
- **Model/Setting/SEStorage.swift**
    - The class for accessing application settings in the User Default storage.

MAP AND PATHS

The code in this section defines how to display photos on the map and to view, record, send and delete paths.

There are included these screens:

- map,
- path overview.

Description of files:

- **MapViewController.swift**
  - The controller to view map and to draw captured photos and recorded paths on this map.
  - It draws captured photos on this map using processes defined in Map/Photos group.
  - It controls the path recording process from UI and draws recorded or currently recording path on this map using the processes defined in Map/PathTracking group.
- **PathTrackTableViewController.swift, PathTrackTableViewCell.swift**
  - The classes to view recorded path in a list structure and to delete these paths.
- **Map/Photos/…**
  - These files define views and classes to draw the captured photos on the map in the form of annotations.
- **Map/PathTracking/…**
  - These files define views and classes to draw the recorded path or currently recording path on the map in the form of annotations, polyline and polygons.
- **Map/PathTracking/PTPath+CoreDataClass.swift, Map/PathTracking/PTPoint+CoreDataClass.swift**
  - There are defined additional convenience methods inside the PTPath and PTPoint managed object classes.

LOGIN

The code in this section defines user login process to the application.

There is included login screen.

Description of files:

- LoginViewController.swift
  - The controller to view login screen and to manage login process.
  - It stores data about logged user in the UserStorage.