EGNSS4CAP IOS APPLICATION SOURCE CODE OVERVIEW

Content

[Content 1](#_Toc68115376)

[General Overview 2](#_Toc68115377)

[System Structure 3](#_Toc68115378)

[Global Structure 3](#_Toc68115379)

[Taking Photos 4](#_Toc68115380)

[Task Management 5](#_Toc68115381)

[Application Settings 6](#_Toc68115382)

[About Screen 7](#_Toc68115383)

[Map and Paths 8](#_Toc68115384)

[Login 9](#_Toc68115385)

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. 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 aims to follow this 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 the source code can be divided based on these parts of the application:

* system structure,
* global structure,
* taking photos,
* task management,
* application settings,
* about screen,
* 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 the following entities:

* PersitPhoto
  + The captured photo as a binary picture data with all metadata.
* PersistTask
  + The metadata of a task.
* 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.storyboard.

Description of files:

* **AppDelegate.swift**
  + The application delegate object that manages the 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:
    - Initializing your app’s central data structures.
    - Configuring your app’s scenes.
    - Responding to notifications originating from outside the app, such as low-memory warnings, download completion notifications, and more.
    - Responding to events that target the app itself, and are not specific to your app’s scenes, views, or view controllers.
    - Registering for any required services at launch time, such as Apple Push Notification service.
  + Loads and holds the Core Data PhotoModel persistent container.
* **SceneDelegate.swift**
  + 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**
  + Contains extensions of all system classes.
* **Model/PersistStorage.swift**
  + The base class registering all derived classes for access to the 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**
  + Defines additional convenience methods inside the PersistPhoto managed object class.
* **Model/DB/PersistTask+CoreDataClass.swift**
  + Defines additional convenience methods inside the PersistTask managed object class.
* **Assets.xcassets**
  + Contains all pictures used in application.

Taking Photos

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

This is linked to the following 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 geotagged photo from the camera.
  + 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 geotagged photo.
* **Model/ConvexHull/…**
  + The classes for computing the convex hull centroid over location data.

Task Management

The code in this section defines how to download, edit, view and send tasks.

This is linked to the following screens:

* Task overview,
* Task detail,
* Photo gallery.

Description of files:

* **TasksTableViewController.swift, TasksTableViewCell.swift**
  + Classes to download and view tasks in a list structure.
* **TaskViewController.swift**
  + The controller to view a task in detail, to add a note, to list task photos and to send a task.
* **GalleryViewController.swift**
  + The controller to display full screen task photos as a gallery.

Application Settings

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

This is included in the 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.

About Screen

The code in this section defines how to show information about the application.

This is included in the About screen.

Description of files:

* **AboutViewController.swift**
  + The controller to view the About page.

Map and Paths

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

This is linked to the following screens:

* Map,
* Path overview.

Description of files:

* **MapViewController.swift**
  + The controller to view the map and to draw captured photos and recorded paths on this map.
  + Draws captured photos on this map using processes defined in Map/Photos group.
  + 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, polylines and polygons.
* **Map/PathTracking/PTPath+CoreDataClass.swift, Map/PathTracking/PTPoint+CoreDataClass.swift**
  + Additional convenience methods inside the PTPath and PTPoint managed object classes.

Login

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

This is linked to the Login screen.

Description of files:

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