ServiceNow iOS App

# Location

The code for the app is located on the inergex-dev GitHub page: <https://github.com/inergex-dev/ServiceNowApp>.

# Overview

The purpose of this app is to offer a condensed, mobile alternative to the service-now website. The app provides support for creating, editing, and viewing tickets.

This app supports iPhone and iPad.

The project is built using xCode 4.6.2, using storyboards, with the project primarily using ARC (Automatic Reference Counting). Some classes don’t use ARC, and are indicated by going to the project’s target > Build Phases > Compile Sources and adding the *-fno-objc-arc* compiler flag.

# Forms

There are 3 forms: Login, Create Ticket, and Update Ticket. Each of these have certain commonalities:

* When a text-box is clicked out of, the on-screen keyboard should disappear.
* When next/go is clicked, the next text field is clicked or a submit action is performed.
* In cases where a drop-down menu would normally be used, the *PickerController* class is used.
* The tag is used as a way to differentiate between different UI elements

In the cases of Create Ticket and Update ticket, these are dynamically created tables, and thus are created in code. This is to offer as much flexibly as possible for adding fields, as well as potentially having custom fields for specific businesses.

# Classes

The naming convention for classes that are used by UI elements on the storyboard is “NameType” (example: LoginView**Controller**). TVC is an abbreviation for “TableViewController”, so as to shorten class names.

## Support Classes

### Utility

*Utility* is a static class, which holds variables / methods that are used across multiple classes, to avoid repetition. This includes functions for

* converting a severity/impact/state integer value to the corresponding string
* getting the username/password of the current user for sending requests
* displaying/dismissing a loading alert window

### PickerController

This class is responsible for handling how the user would choose an option from a range of set values. When the user indicates they wish to choose a value, a segue should be performed with this object, passing along an array of choices, as well as a pointer to a “*SelectedRow*” object. Once the user chooses an option and returns, the class will update the value of the *SelectedRow*, so that the form can access the new value.

### Ticket

The ticket class holds all ticket/incident data, for easy transportation between ViewControllers.

### SOAPRequest

Sends SOAP requests which take a method name and a list of comma separated *SOAPRequestVariable*s, and sends the SOAP request to service-now. The result is returned through the use of delegate methods, after being converted into a *TBXMLElement* (or an *NSError*). The delegate method should ideally be the caller of the *SOAPRequest*.

### PullToRefresh

An open-source set of files that enable functionality for pulling down a UITableView and refreshing the content of the table, as well as indicating the table is refreshing.

### TBXML

An open-source set of files, that enables a tree style XML parser; much more manageable than the default objective-c XML parser available at the time.

## Core Classes

### LoginViewController

This is where the program starts. Once the user successfully logs in (Using *SOAPRequest*), their username and password are stored in the *Utility* class, so that they can be retrieved for SOAP calls. If the user wishes to have the device remember their credential, they are also stored in the **standardUserDefaults**.

### MainMenuTVC

This is a static table view that acts as the top level of navigation, with 4 options:

* **Create Ticket** - Performs a segue to *CreateTicketTVC*
* **View Open Tickets** - Performs a segue to *OpenTicketsTVC*
* **View Closed Tickets** - Performs a segue to *ClosedTicketsTVC*
* **Sign Out** - Clears stored passwords, and returns to *LoginViewController*

### CreateTicketTVC

A dynamic table view form, which creates all fields in code; it uses *SOAPRequest* as well as *PickerController*. Once all fields are filled in and valid, a ticket request is sent, and if it returns positive, the user is sent back to *MainMenuTVC*, with an alert window informing them the ticket was successfully sent.

### OpenTicketsTVC

This dynamic table view shows a table of all the user’s open incidents. The user may pull down on the view to refresh the table (Using *PullToRefresh*). The view refreshes every time it appears. Once a row is selected, a segue brings the user to *ViewOpenTicketTVC*, passing along the selected ticket information.

### ViewOpenTicketTVC

This dynamic table view displays all of the information pertaining to the open status of a ticket. It is also possible to edit/update a ticket, but hitting the soft “Update” key on the view’s navigation bar, which will segue the view to *EditTicketTVC*, passing along the current ticket.

### EditTicketTVC

A dynamic table view form, which creates all fields in code; it uses *SOAPRequest* as well as *PickerController*. Once a change has been made (so as to avoid sending a pointless update) and all fields are filled in and valid, an edit request is sent, and if it returns positive, the user is sent back to viewing the ticket (*ViewOpenTicketTVC*), with an alert window informing them the ticket was successfully sent.

### ClosedTicketsTVC

This dynamic table view shows a table of all the user’s closed incidents. The user may pull down on the view to refresh the table (Using *PullToRefresh*). The view refreshes every time it appears. Once a row is selected, a segue brings the user to *ViewClosedTicketTVC*, passing along the selected ticket information.

### ViewClosedTicketTVC

This dynamic table view displays all of the information pertaining to the closed status of a ticket.