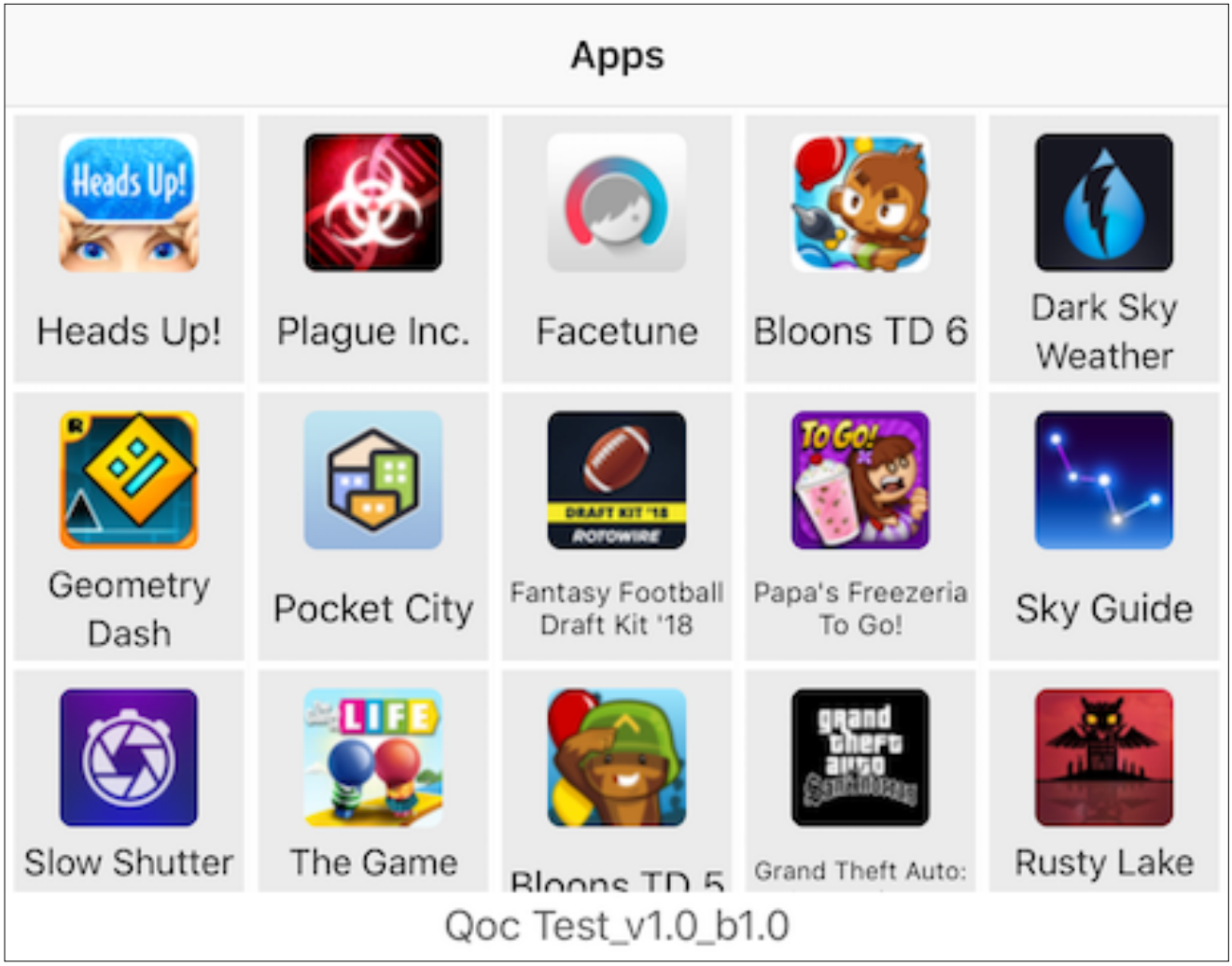


Top Paid Apps

Prepared for: Qoc Health

Prepared by: Sivakumar Boju, iOS Developer

August 19, 2018

Purpose: Interview

QOC HEALTH

IOS INTERVIEW PROJECT

Objective

Create an iPhone application that accomplishes the following goals

Goals

Please use the following link to display the top 100 paid apps from the following link
http://phobos.apple.com/WebObjects/MZStoreServices.woa/ws/RSS/toppaidapplications/limit=100/json

Screen 1 - List of Apps

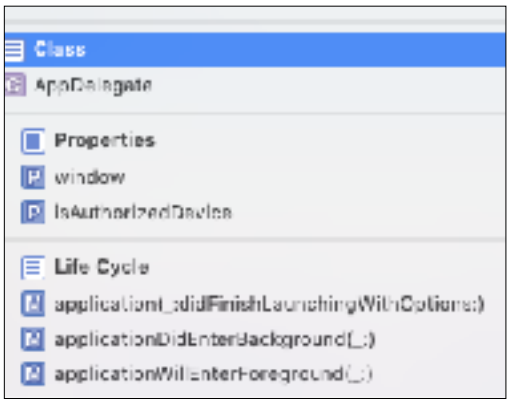
- Display application title and application thumbnail

Screen 2 - App Detail View

- Display application details such as
  - Title
  - App Icon
  - Release date in “MM/dd/yyyy” format
  - Summary
  - Price
  - Category
  - Link to app store
  - Publisher Name

Guidelines

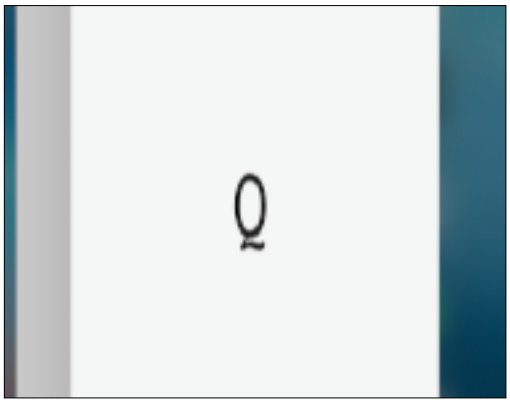
- Use Swift or Objective-C
- Use Apple’s native networking protocol



SOLUTION

Application Security

Our first focus to protect the application and its data. So in order to achieve that some simple functionalities have been added such as



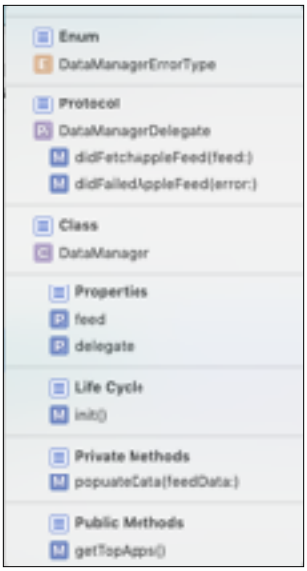
• Whenever the application is launched, the app will check to see if the target device (iPhone) is jail broken or not and its status stored for later reference.

(**Note**: Currently this status is not used anywhere. However we can use this status to control the application as required)

•Whenever the application goes into background, we do not want to display the application screen information. So we add a splash screen when in background and remove that splash screen when in foreground.

• App Info class has been added to provide app name, version and build no. This can be used to protect the web requests (as header data) , however it is currently not in use.

Application Data

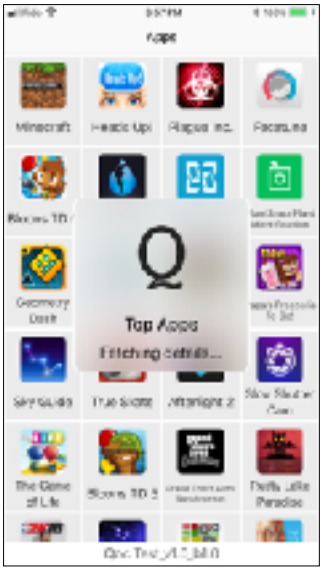


Data Manager class has been created to provide the top 100 paid apps from the source link. This class has one public method **getTopApps** and two delegate protocol methods such as **didFetchAppleFeed** and **didFailedAppleFeed**

The network request has been processed using Apple native APIs. No third party frameworks are used.

Lot of refactoring can be done to make it reusable, however it does the required job of fetching the feed from the link and to return errors.

List View



- List View uses a collection view to display Application Title and Icon.
- Status Label at the bottom is used to display the app info. It is also used to handle failure messages from **Data Manager Protocol** methods.

Localization

- SwiftGen - a third party library has been added via pod to handle localization
- Static texts in all screens are initialized with localized text.
- English and Spanish sample texts are added for now. (Spanish still uses the english text, translation pending)

Library

- AMStatus View component has been added to handle on demand activity messages.
- Components like this can be built as a framework and can be used by all projects. However in this case added directly to the project to show case the use case.

Detail View

- Below is a sample detail view for a selected application

Architecture in general

This app in general has the following concepts implemented

- to check if the device is jail broken
- to hide application information, to avoid other users to see the info, when the application is in background
- app info can be used to protect web service requests (but not used at this time)
- a sample library component that can become a framework
- a pod sample to handle localization
- a queue to handle background requests
- delegate protocol to handle data manager success and failures
- extension for objects such as Application, UIImageView, String and UIView (some may not be in use)

Please let me know, if you have any questions. I would be more than happy to provide explanations. Hope this helps.

Regards

Siva Boju

647 878 0004 | boju.siva@gmail.com