Approach:

- I use a simplified VIPER Pattern approach because it uses SOLID methodology. In my case, this is not a FULL VIPER but a taste of it or a better version. The drawback of this pattern is the fact that we have to declare all variable in VIEW (main) file since Interactor and Presenter are just extensions of View.
- I used some 3rd part library such as:
  - Alamofire - to fetch images from the internet. (I would use it to get JSON file as well)
  - Realm - to save data locally - actually we don't need it but it already prepared for next version.
  - SCLAlertView - to show cool popup instead of original one
  - GridLayout - custom layout for UI Collection View. There are many different approaches to make a custom layout, I used this one for several times in other projects so that is why I use it now.
  - XML parser
- I use POD so you should run “pod install” before running the project and you must you xcworkspace instead of xcodeproj.
- This demo should run OK on different devices from SE to 7s Plus
- I didn’t use testUnit because of deadline and simplicity of app. I know, it sounds as an excuse, but it’s true.
- Home View and Favorite View are using the HomeViewCell since they required almost the same functionality. So I could reuse the code.

For Home View (main list) I user custom Layout. The layout logic is happening inside the GridLayout.swift For Favorite View, I used a normal one.

Note:

I used REALM in my app but I could create this app and achieve the same result without it in much easier way by just saving images inside the some specific Album and when user would open Favorites, I could simply show those images (by deleting 1 favorite image I would simply deleted the image, that’s it, very simple and easy).However, I decided to built something more complex with local data base.

You will see some warning messages regarding XML parse. Don’t worry, the have no impact. I would fix it right after doing few more tests. They are generated by 3rd party library.

On Home View you can perform:
- pull to refresh data (I know I could add a button to the very last position and add new 20 images every time user clicks on it, but for sake of simplicity and time dead line I used “pull to refresh” approach)
- reorder the cell with long click
- as you can see, that API gives gives us no longer existing images. This simple app will detect this situation and won’t add “fav” button and will put some thumbnail so user could understand what is happening
- when user add an image to the “fav” app removes the button… however, even if he could add twice, the REALM will prevent it from happening, so se never have 2 same objects in our data base

but we could have duplicates of images.

On Favorite View you can just remove the image. When you try to remove you will see 1 popup where you need to confirm and another one to delete the image from Album. You can remove the entity from database without deleting the image

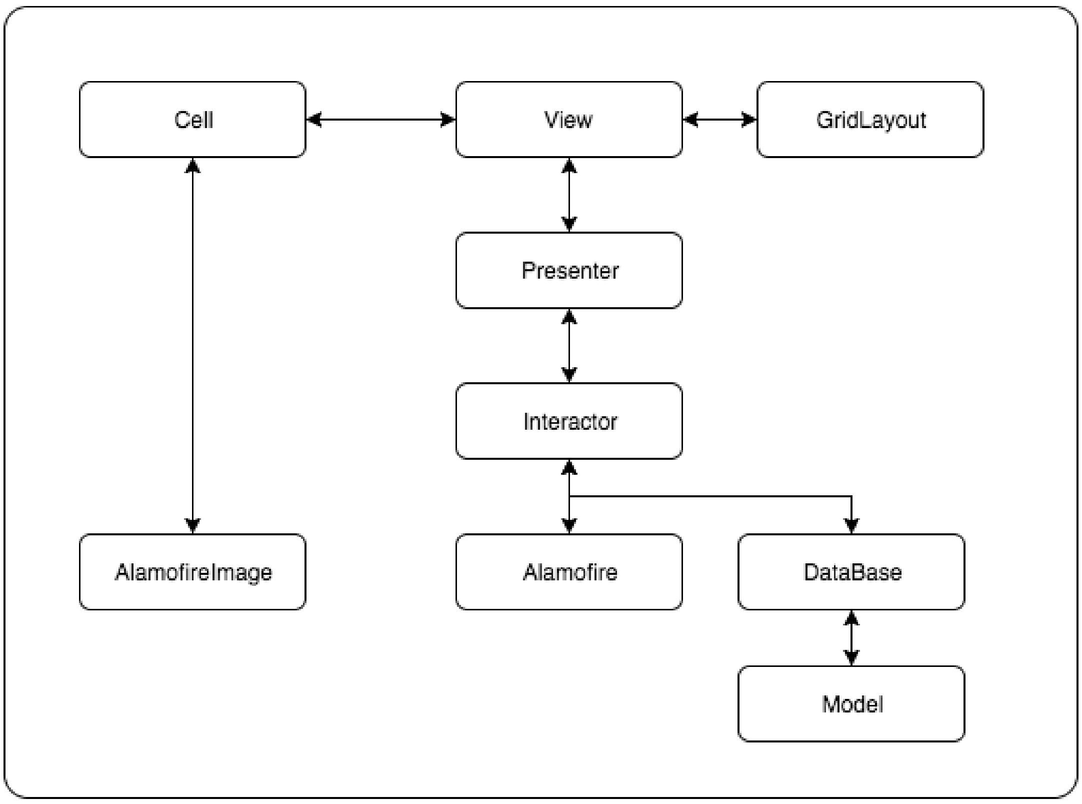
Known issue:

- Enter the app with Internet Connection, wait until the list is loaded, turn off internet, try to "pull to refresh"… app will stuck

- If user deletes Image from Album and than enter the App. He/She will see empty image.

- haven't tested with iPad.

The logic of the app is:



Need to tell that "cell" and "Interactor" is also using our Model as a data class.

Conclusion:

I know I could use MVP or MVVM but I saw that VIPER pattern is much more productive in long run. It has its own drawback but I think we should give him a try. Especially if there are juniors in the team, because of SOLID they are forced to make think in right direction.
If you don't like this pattern I can change it to MVP in 1-2h or so.

Since the details also count, I have added some icons and splash screen to this demo and made this introduction.

Hope you like and we can move to the next stage.

Thanks