4 Mobile Design Patterns

In order to understand if a mobile app has a good or bad mobile design, it is useful to read bad reviews on app stores because they give a lot of information about how we can improve such an app. In fact, good reviews are usually not useful for this purpose.
One way to approach the design of UIs is to learn from examples that have proven to be successful in the past. *Design patterns* are solutions to a recurrent problem within a specific application domain. We now examine each of them in detail.

4.1 Navigation

It is about how users move through the app views. Apps with good navigation just feel simple and make it easy to accomplish any task. We divide such patterns in primary and secondary.

4.1.1 Primary Navigation Patterns

They consist of navigation from one primary category to another (like in top-level menus of a desktop app); they are divided into

- *Persistent Navigation* which concerns interactive navigation components that are permanently visible
- *Transient Navigation* that must be explicitly revealed with a tap or a gesture.

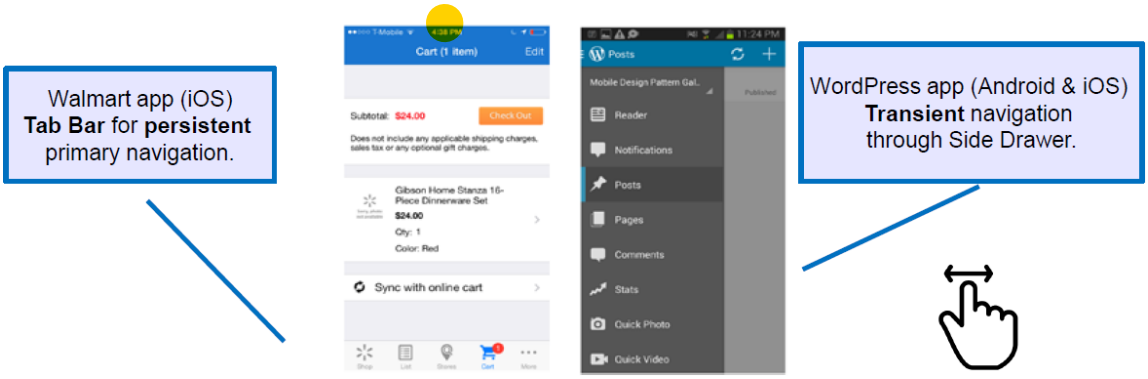

Figure 18: Primary Navigation Patterns.

For persistent patterns, we have a series of elements that we will be now explained in detail.

- *Springboard* is a landing screen with options that act as launch points into the application, using a grid layout for items of equal importance, or irregular layout to emphasize some items more than others.

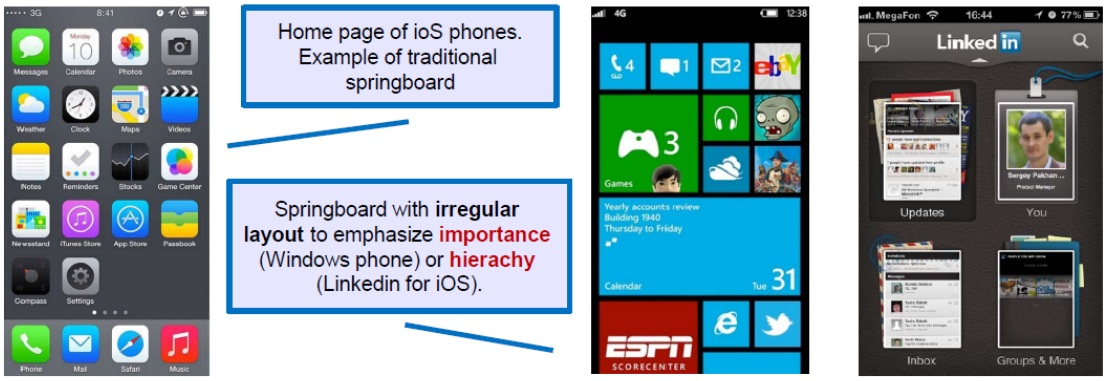

Figure 19: Springboard Layouts.

- *List Menu* is a view where each list item is a launch point into the application, and switching modules requires navigating back to the list. Expanding lists are the ones that expand with respect to the user's action, like in Android's CollapsingToolbarLayout.

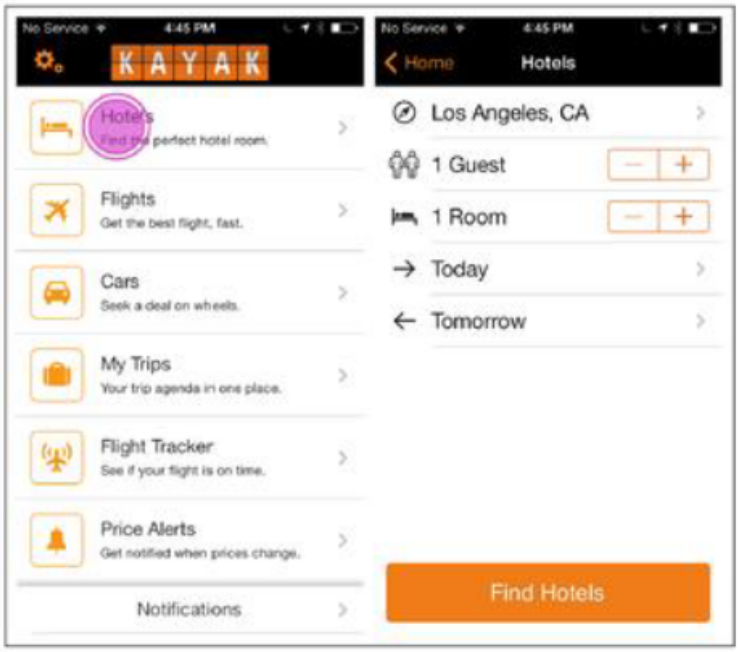

Figure 20: List Menu.

- *Cards* are small containers that logically encapsulate relevant information. Card navigation is based on a card deck metaphor, often including common card deck manipulations such as stacking, shuffling, discarding, and flipping. They are good for presenting similr objects whose size or supported actions can vary considerably.

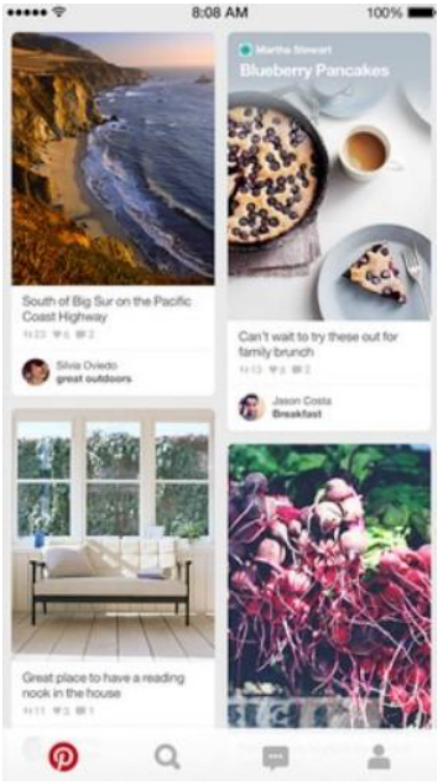

Figure 21: Cards.

- *Gallery* pattern displays live content (news, recipes, movies or photos) arranged in a grid or a carousel.

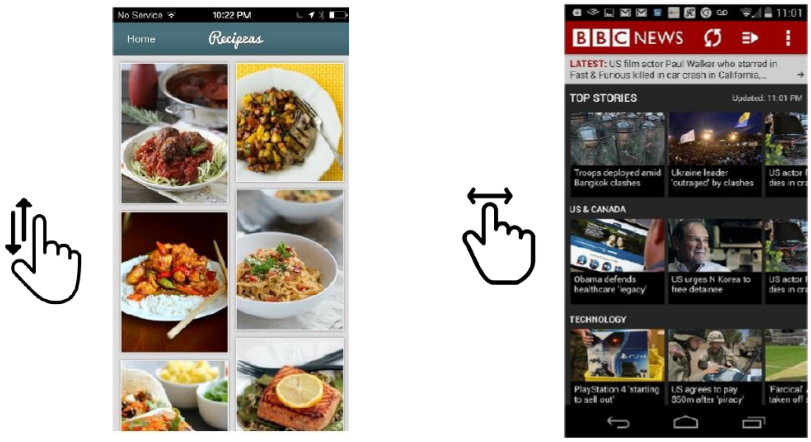

Figure 22: Gallery, grid (left) and carousel (right).

- *Tab* bars are suggested to navigate flat information structures: users can navigate directly from one primary category to another because all primary categories are accessible from the main screen.

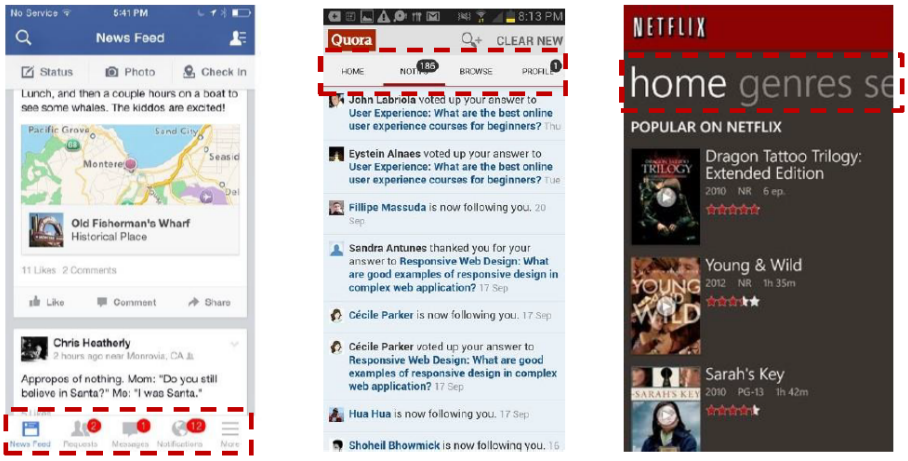

Figure 23: Tabs.

- *Metaphor* is characterized by an interface designed to match its real-world counterpart.

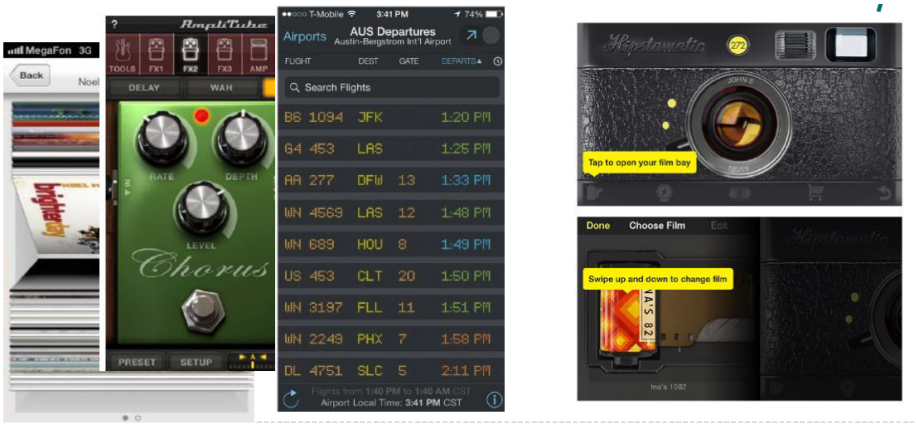

Figure 24: Metaphors.

- *Infinite Area* is a pattern where the entire data set can be considered to be a large, navigable 2D graphic, like it happens in Google Maps.

For transient patterns instead, we have to start from the meaning of the word "transient": it means "staying a short time", which is exactly how such navigation components work. They are hidden until we reveal them; then we make a selection and they disappear again. Let's examine them singularly:

- *Retracting Tab*, when tab bar collapses or appears when the user is scrolling or swiping through content;

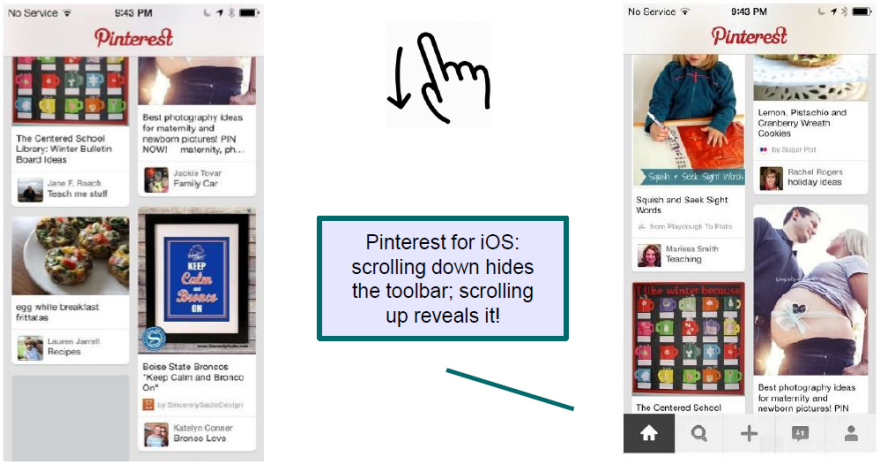

Figure 25: Retracting Tab.

- *Side Drawer*: there are two styles of Side Drawers:
  - *overlay*: a swipe gesture will reveal a drawer that partially covers or overlaps the original screen content
  - *inlay*: a swipe gesture will open a drawer that pushes the original screen content partially off-canvas

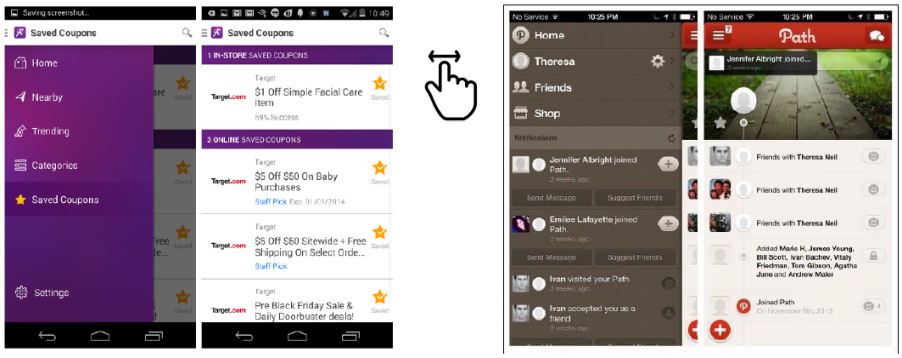

Figure 26: Side Drawer: overlay (left) and inlay (right).

- *Toggle Menu*, which can be an inlay that pushes the content down below the menu, or an overlay that appears as a layer above the content. The overlay design is the more common option in native mobile apps; it should not cover the whole screen but instead let the background peek through. Tapping anywhere in the background should also hide the menu.

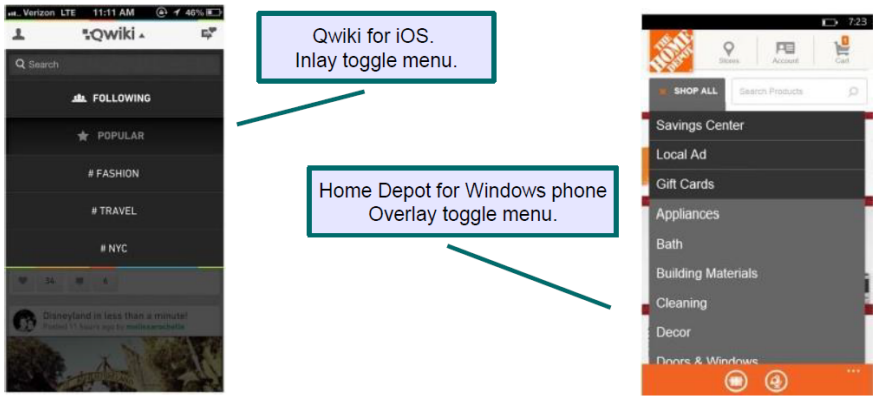

Figure 27: Toggle Menu.

Persisent vs Transient Navigation: when deciding which to take into consideration, we have to answer a few questions:

- Is your app "flat"?
- Are the menu categories equivalent in hierarchy and are there just a few primary categories (i.e. three to five) in the app?
- Do your users need the menu to be always visible for quick access?
- Do the menu categories have status indicators, like the number of unread emails, for instance?

if the answer is "yes" to one or more, it is better to choose persistent navigation.

4.1.2 Secondary Navigation Patterns

It regards the moving and navigating within a selected module.

- *Page Swiping*: this pattern for 2ndary navigation can be used to navigate quickly through content using the swipe gesture. The most common way to communicate this navigation pattern is via page indicators (three little dots) or with cards.

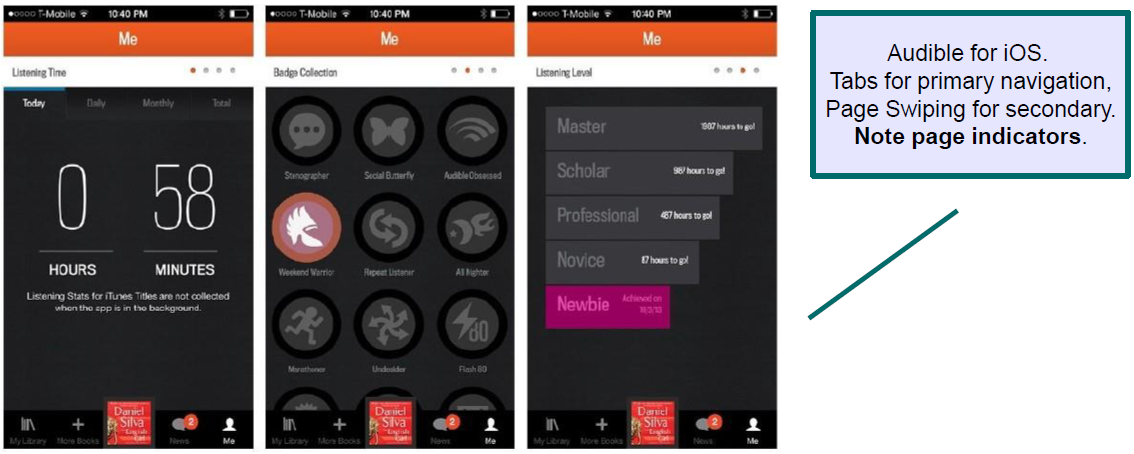

Figure 28: Page Swiping.

4.2 Forms

Forms are used for data entry and config features.

4.2.1 Sign In

Sign in forms require a minimal number of inputs, like username, password, command button, password help and so on. Often, they are visible in the same form of, or alongside with, sign up forms but sometimes they are separated (accessible via buttons). Remember that registration has to be short and done only if necessary.

4.2.2 Check-In

Check-In allows people to use the GPS on their mobile devices to let friends know exactly where they are. We have to keep it ultra-short and design it for speed and efficiency.

Figure 29: Check In.

4.2.3 Comments

App sections used for inviting and allowing users to leave comments. We always have to clarify what is being commented and show other people's ones over time. Timeline should be in minutes ago, hours ago, day of the week etc. depending on the range of viewed past comments.

4.2.4 User's Profile

We put the name and the picture in evidence. We also show users' contribution to the app or to the social network and provide action controls.

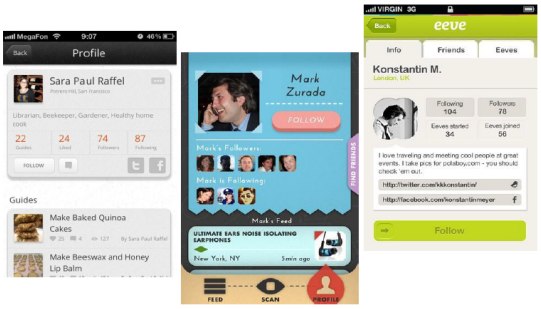

Figure 30: User Profiles.

4.2.5 Share

Always provide an "off social" way to share, e.g. by email. Remark what is being shared and keep track of past logins.

4.2.6 Empty Datasets

Avoid white-screens, explain why the dataset is empty, avoid error messages. Let's say you have a history, display "Empty history" with a fancy drawing!

4.2.7 Multi Steps

Show the user where they are and where they can go. These steps are often organized as sequential workflows. We also have to minimize the number of pages and steps.

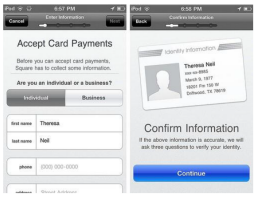

Figure 31: Multi Steps.

4.2.8 Settings

They have to be put inside the app, clear and grouped. Easy to understood!

4.3 Search

We subdivide search in four types.

4.3.1 Explicit Search

We have to offer a clear button in the field, provide an option to cancel the search and use feedback to show that the search is being performed.

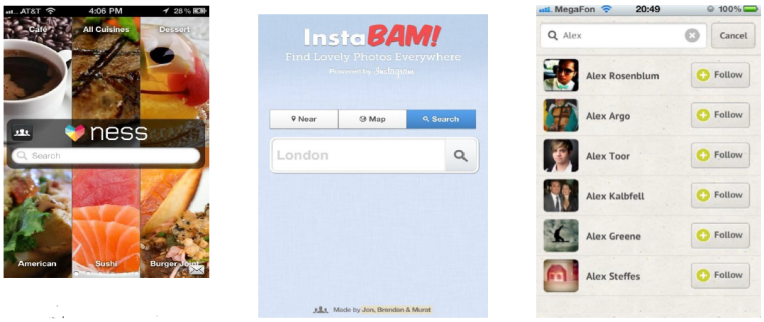

Figure 32: Explicit Search.

4.3.2 Dynamic Search

Automatically filters a given list of items dinamically, during typing. Works well for constrained datasets, like an address book.

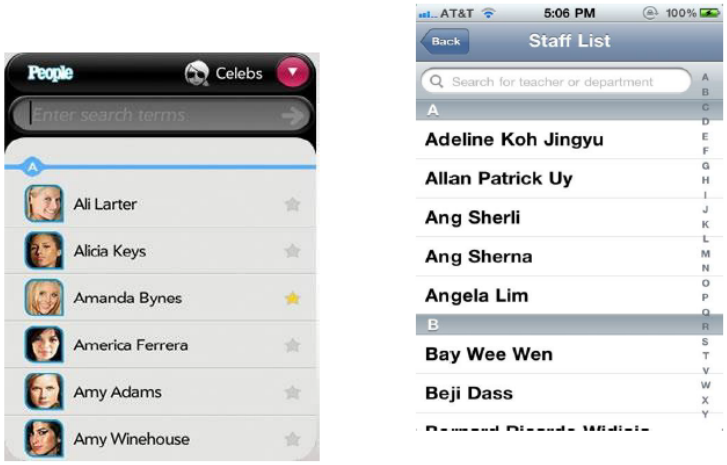

Figure 33: Dynamic Search.

4.3.3 Search Form

We have to minimize the number of input fields, follow form design best practices and use only when strictly needed.

Figure 34: Search Form.

4.3.4 Search Results

We scroll down to analyze the results, apply a reasonable default sort order and call for action.

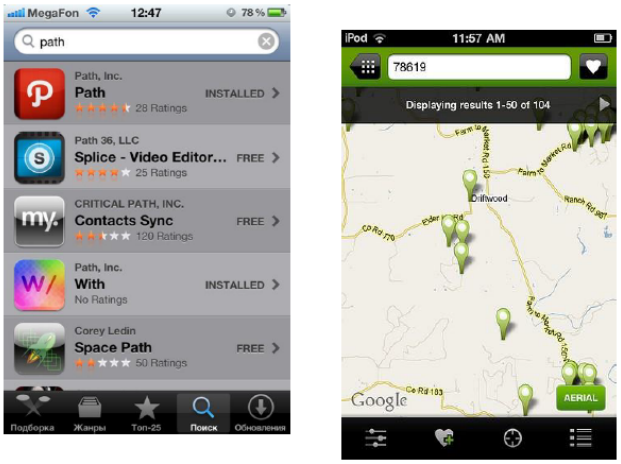

Figure 35: Search Results.

4.4 Tools

4.4.1 Toolbar

A toolbar contains screen level actions, generally displayed at the bottom of the screen. Different from the tab bar! We have to choose icons that are easy to recognize, or use lables plus icons.

4.4.2 Contextual Buttons

If buttons are necessary, they should be displayed in proximity to the actionable object. We have to choose a familiar icon or use a text label for such buttons.

4.4.3 Inline Actions

They should be in proximity to the actionable object. We have to choose, again, a familiar icon or use a text label. Inline actions are the ones present in an Android's RecyclerView, for example, which do something with respect to the selected item. No more than 2 inline actions should be added for item.

4.4.4 Call to Action Buttons

We don't have to hide the main call to action in a menu or disguise it as an unrecognizable icon in a toolbar. We have to choose a good contrast and clear label. Think about the FloatingActionButton of Android.

4.5 MultiState Buttons

They work well for a series of tightly correlated actions that will be performed in succession (like an enable/disable button).

4.6 Actions on Maps

We provide visible markers, using as much screen as possible.

4.7 Helps and Tutorials

They are helpful tips displayed the first time a user launches an app.

4.7.1 Dialog

We keep dialog content short and make sure there is an alternate way to access instructions from within the app. Think about the AlertDialog of Android.

4.7.2 Tips

We place tips in proximity to the feature they refer to, keeping the content short and removing the tip once interaction begins.

4.7.3 Tour

A tour should highlight key features of the application, preferably from a user goal perspective. We keep it short and visually engaging. They are usually presented only on the first run of the app.

4.7.4 Video

Demos and screencasts should showcase key features or show how to use the app.

4.7.5 Transparency

Transparencies are not meant to compensate for poor screen designs. We remove it once interaction begins.

4.8 Feedback and Affordance

4.8.1 Feedback

Feedbacks are composed by *errors* in plain language that propose a solving method, *confirmation* provided when an action is taken and *system status* which provides feedback about the system's status.

4.8.2 Affordance

In affordance we have *Tap*, *Flick* and *Drag*.