inż. Jan Romaniak 184755
inż. Piotr Giedziun 184731

Wrocław, 21.04.2014

App. programing - java & xml

course leader:
Mgr inż. Jarosław Rudy

1. Project description

   First of all I would like to explain why we made Cookbook application. Our goal was to create modern, simple to use, but yet powerful Android app. As a iOS users, we used to well designed and easy to use apps. However it's not the case for many modern Android Applications. Most of them was developed for older API (Android versions), using default widgets. Back in the days of Android 2.x almost all apps were based on black background and white text. Since then, Android User Interface team improved a lot of build-in components. Today default android design is white background and black text (similar to iOS). We have chosen to create cookbook app due to visual aspects of food. Food brings only good feelings, everyone has to deal with it.

2. Research

   1. Available apps

      Android Market is rapidly catching up with the Apple App Store. As a market grow, developers have to compete in order to stay on top. This trends leads to general UI and usability improvements. As we were looking for existing apps, we came across well designed and badly made. Android Market (Google Play) provides user ability to express their feelings about app in two ways. First, the easiest one to perform by user is grade. It’s not always accurate way to describe application. If developer fails to adjust app for common devices, he will be rated low by theirs users. It will cause average grade to be low, hiding all positive feedback. Also developer can fail in one of many updates, making his app to froze or lose users data. This unfortunate action also will reflect average grade, long after cause of issue will be fixed. Another way to describe app is reading users comments. This option is more accurate, because we are able to determine what’s wrong with app itself. Our research was focused only on Cookbook-like apps. We have read several negative comments and defined following list of main issues:

      - crashes - this is main cause of negative comments. People value this aspects more than UI

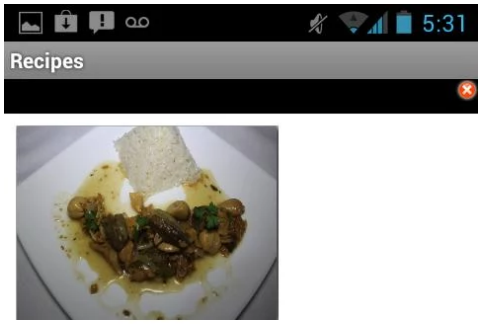
      - locking portrait mode - it’s the easiest way to fix several android bugs, also helps to create layout for predefined ratio. It does all it says, but people like to use their devices in landscape mode as well

      - default widgets - it’s mostly the case for older apps. As I said before, Android used to have ugly buttons, black and white colors, ugly overall

      - locking device, while executing networking tasks - it’s also common in Android applications. Android introduced dialogs, that were designed to take control of entire screen. While displaying loading dialog, user is not able to perform any action. When request will be taking longer time, user is not able to cancel it. This behaviour is unacceptable, all long running task can not be executed on main thread for this reason.

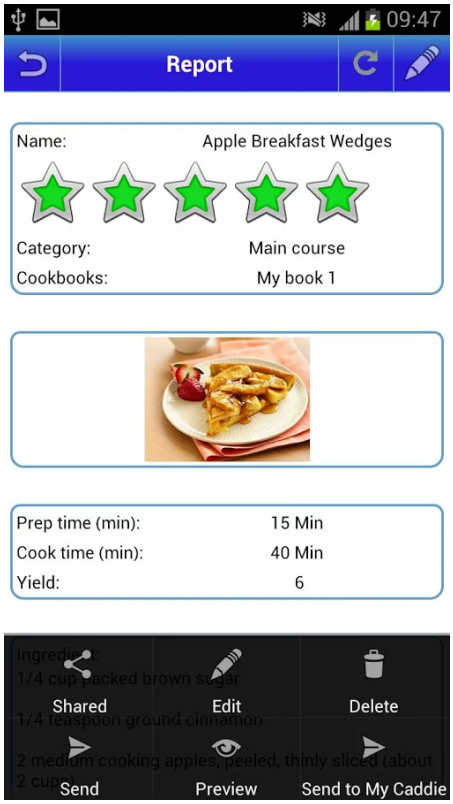
      Examples of poorly made apps:

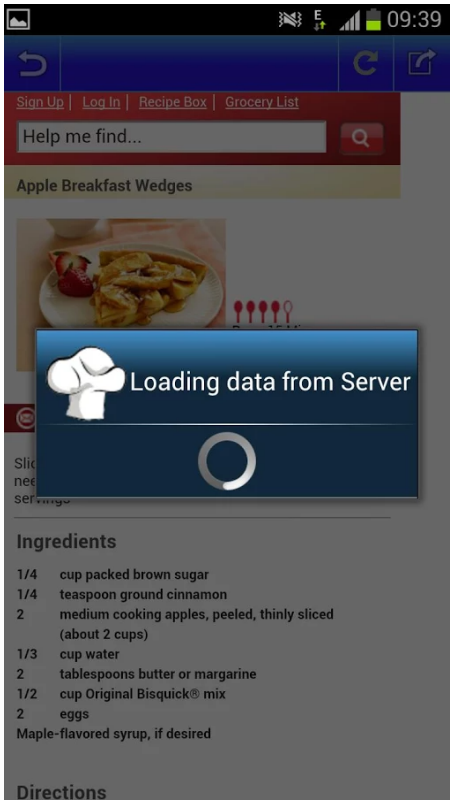
      1. **Haitian Cookbook** - this app is an good example of almost all we have listed before. Application works only in portrait mode, recipes are displayed in text list (no images in list). Recipe activity is displaying data in webview - default colors.



- 1-2lbs Salt Cod Filet (easier to use, has no bones)
- 1 Onions diced
- 1lbs of Arbre a pin (Chesnut or Chateigne) - boiled
- ½ lbs of Okra
- 1 bunch of parsley (chopped roughly)
- 1 Hot pepper (choice)
- 4oz of butter
- 1oz Olive Oil
- Adobo seasoning to taste (can be found in any Caribbean Market)
- Cracked black pepper to taste
- 4oz of seafood stock (preferred or

2. **My Recipes - cookbook** - similar to previous app. Designed with usage of default (old) Android widgets. Small font makes it hard to read. Request are performed with usage of progress dialogs, blocking app when it fails (happened few time during our tests).





2. User Interface designs patterns

We consider ourselves as a newbies in this subject. In order to gain knowledge, we have read several online tutorials involving this subject. We were surprised by the fact, we knew most of them already. We will explain them later on, in Prototype section.

3. Android platform specific patterns

Android application can be displayed on various range of devices (TVs, smartphones, fridges etc.). Android UI have to be flexible, unlikely to iOS where most of layouts is designed as absolute layout (each widget is positioned in fixed position). Android introduced many positionings patterns (for example RelativeLayout - where elements are positioned in relation to each other). Consistent navigation is an essential component of the overall user experience. Android 3.0 introduced common buttons required on all devices (if device doesn't have hardware buttons - required buttons will be displayed on screen). The most essential button is *back button*, this way user is able to navigate through previously visited application states. Another key feature of all new applications is Action Bar. Described as dedicated piece of real estate at the top of each screen that is generally persistent throughout the app. It's place for activity title and few fast access buttons. Another important pattern is Navigation Drawer (a panel that transitions in from the left edge of the screen and displays the app's main navigation option). The user can bring the navigation drawer onto the screen by swiping from the left edge of the screen or by touching the application icon on the action bar.

3. Resources

We were looking for the easiest way to get recipes to our cookbook application, and on our way we found *yummly* API. It provides photos in good resolution and proportions(we need one square and one panoramic), the information about recipe is detailed enough, so it exactly fit the bill. The only disadvantage to use it is that it is not free for commercial usage.

4. Prototype

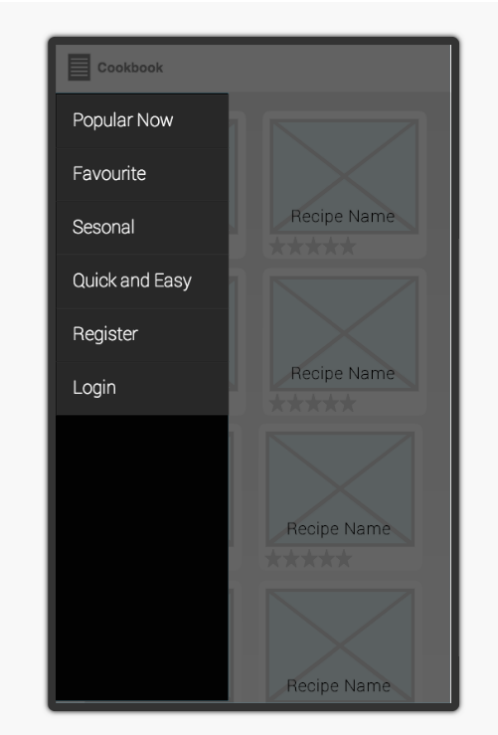
Each company have their own prototyping techniques. I came across few of them, the one I enjoyed the most was web based tool - FluidUI. This tool have free plan for non-commercial projects. The best function (at least I enjoy it the most) is ability to render prototype on mobile device. This way I was able to check how certain functionality feels like. Moreover there is plenty Android widgets to choose from. In our case prototype is fully functional (you can click to move across application). Link to preview: https://www.fluidui.com/editor/live/preview/p_xEHGZOGTSok7mT8Vm6uTXbBl00QreeDf.1400677785407

All laws I will be using in this section were mentioned in this article http://99designs.com/designer-blog/2014/01/15/7-unbreakable-laws-of-user-interface-design/.

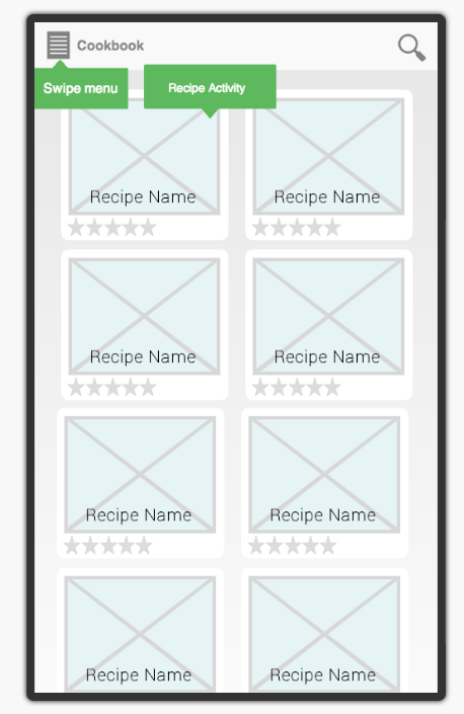
Before I will explain each view in details I want to give brief introduction to our thinking process. Our goal was to develop stable solution, that will take care of all mentioned in Research section issues. Android application structure is based on activities, only one activity is displayed, rest of them are stored in Back Stack (in the order in which each activity is opened).

Another good practice is access to Navigation Drawer across entire application. This way user will be able to access all available functionalities, without need to going back through all previous screens to access menu.

This is how application flow looks like. User is able to access Navigation Drawer (left side menu) from all listed views. Lets take a closer look at out menu. We have listed all quick links to application features. When user will successfully login, Register and Login buttons will be replaced with Logout button.



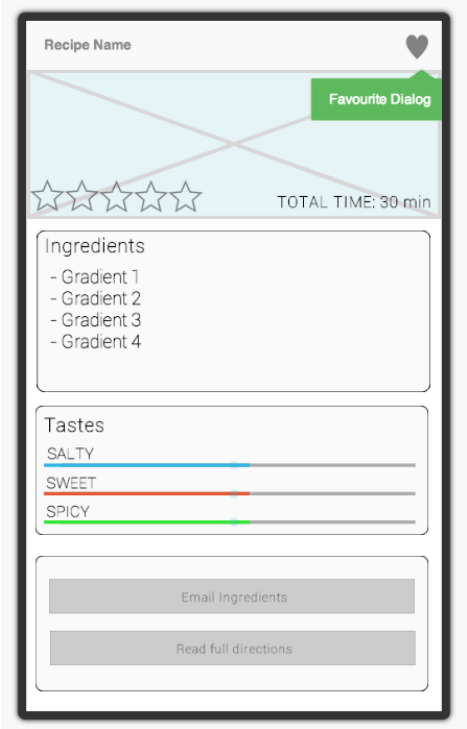
We believe, that user should experience essential app features right after opening it. Our app have certain features, that require user to login. We could for him to login (by opening login activity at the beginning), but this could discourage user. If he is not in a mood to enter any data, he will just close and most likely uninstall app. In our case user will be ask to login/register, when he will pick a functionality, that require logged account.



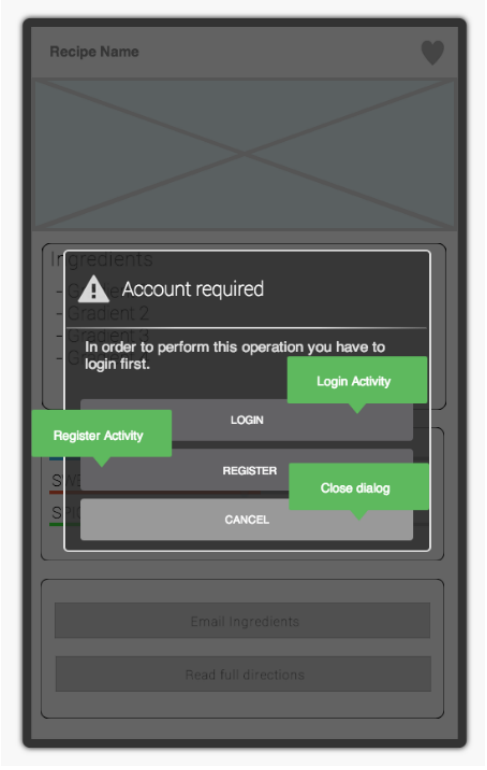
Here it is, our main screen - list of all recipes. We have used *Law of clarity* while designing this list. This app is all about the food, so the best way to experience a meal is to show it. Simple as that. The user will avoid interface elements without a clear meaning. We did all we could to make it as simple as it could be. All essential information about recipe are given in cards (recipe name, score - visual representation is stars and picture).

User is able to filter through list by using magnifying glass button (from navigation bar). When he will perform swipe gesture, left side menu will be opened. He can return to main list (Popular Now, Favourite, Seasonal, Quick and Easy) or access one of access management screens. As I mentioned previously Favourite tab will redirect to login/signup when user is not logged in. This is usage of another law, Law of guided action. It says "The user will probably do something if he is asked to do it.". We came across it everyday. For example facebook uses this technique to suggest you new friends.

When we will chose one of recipes application will open new activity - recipe details. This action require another call to remote API (in order to receive detailed data). This action require asynchronous call, somehow we have to give feedback to user, that he clicked button and action is being performed. We were trying different approaches. We decided to use progress dialog. In order to prevent application lock, we also implemented a way to cancel request. When user will use back key, request will be canceled and progress dialog closed. This way if user will change his mind, he don't have to wait until previous request will be finally executed.

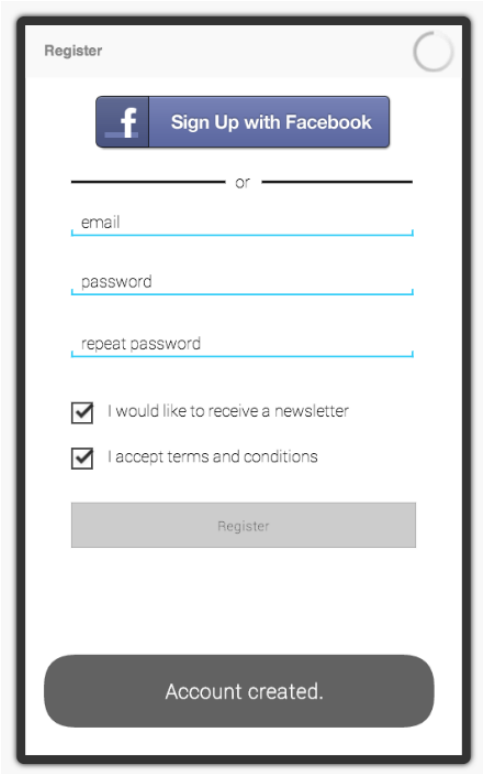


We also decided to use card layout in this view. Recipe is displayed in nice, clear and grouped view. Recipe name is displayed in Action Bar title, score and time is displayed on better quality image. Another used law is *Law of context*. The user expects to see interface controls close to the object he wants to control. In this case we replaced magnifying glass from previous screen with heart (star in final version). We also use *Law of guided action* here. When guest will click favourite button (heart) he will be ask to login/register. I was describing this view before, here it's how it looks like.



In final version (where heart was replaced with start) we are also using *Law of feedback*. The user will feel more confident if you provide clear and constant feedback. When user logged user will click star, star will change into full-star (if he just added recipe) or empty-star (when he removed it from favourites).

Lastly I will explain authentication views. First one is Register screen. This screen is compilation of almost all laws. Let's see it first.



Let's start from top to bottom. While user will enter data and click register button, he will be provided with visual feedback - register button will be blocked, loading indicator will be displayed (left-top corner). When registration process will succeed he will be provided with Tooltip (Account created). Another trick we used is Law of defaults. It's said, that the user will rarely change default settings. We used it against them and set "I would like to receive a newsletter" by default. We know, that people are lazy, this is why we decided to add Sing up with Facebook feature. This way without entering any data user is able to use register. We also decided to break registration process into two steps. This action is known as *Law of easing*. The user will be more inclined to perform a complex action if it's broken down into smaller steps.

5. Implementation

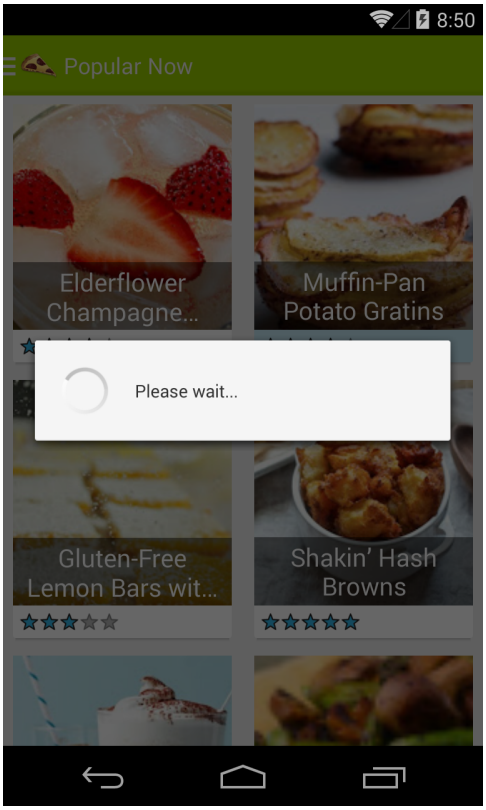
1. API

To receive cookbook recipes we are using *yummly REST API*. It offers possibility of getting data in JSON format. We actually using two methods from this API:

- Getting list of recipes
- Getting details of selected recipe

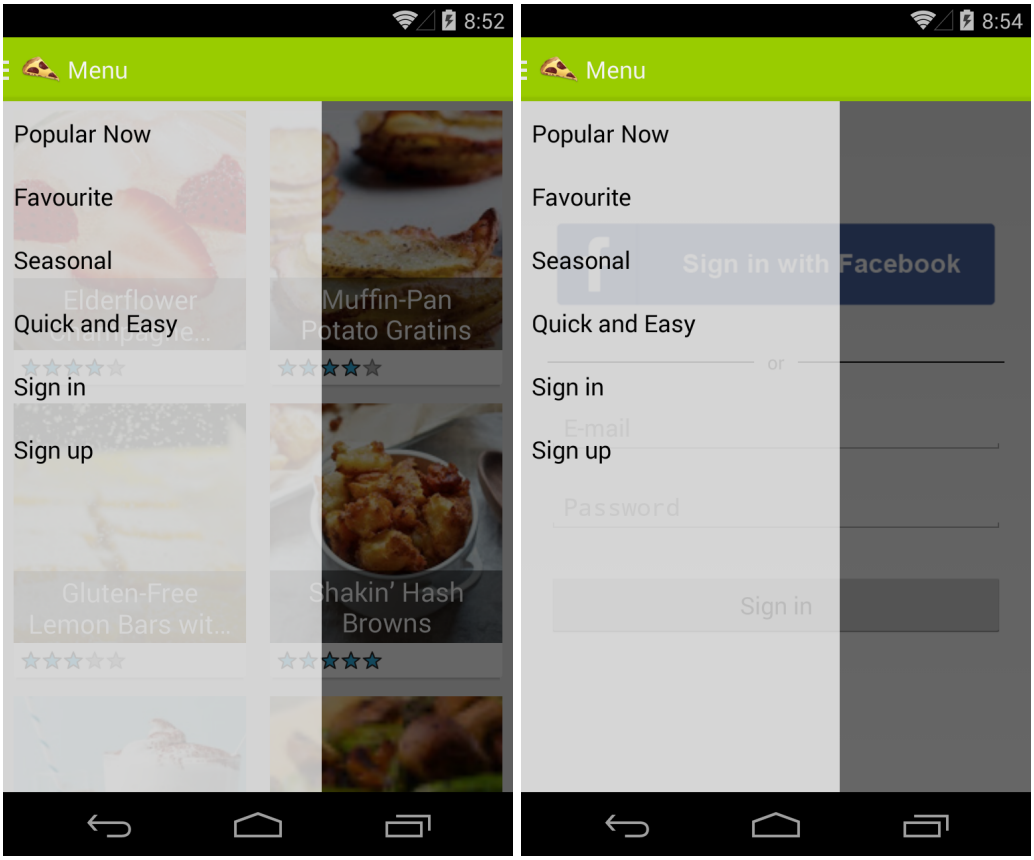
To parse fetched data to java objects we are using *Google GSON* library. Each API call is runned asynchronously. This way of calling methods which take long time allows to preserves the responsiveness of user interface. While request is executing, application

displays progress dialog with information about it. When the user decide to cancel actual task the dialog can be dismissed by back button or clicking around the dialog. To receive images is used *lazy loading* mechanism, the image is downloaded from internet in the moment when it is requested.



2. Main menu

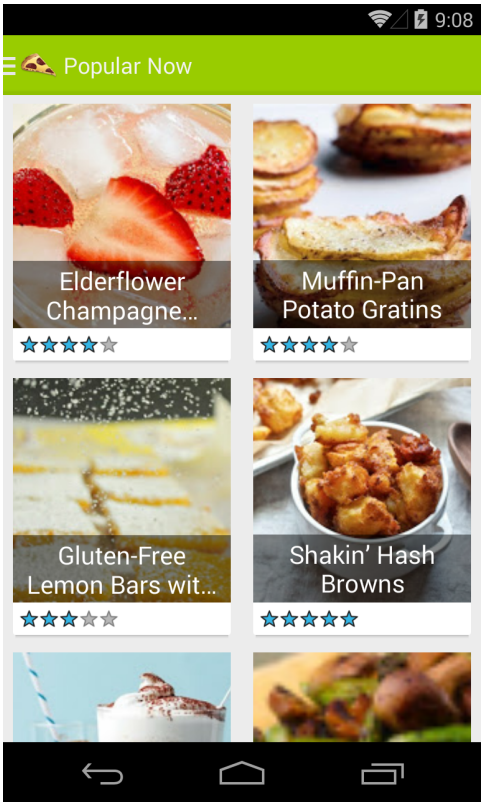
Application menu is available from each place in application, it allows to quick access to all major views. To get this feature we had to use *navigation drawer*. This decision forced us to use *fragments*.

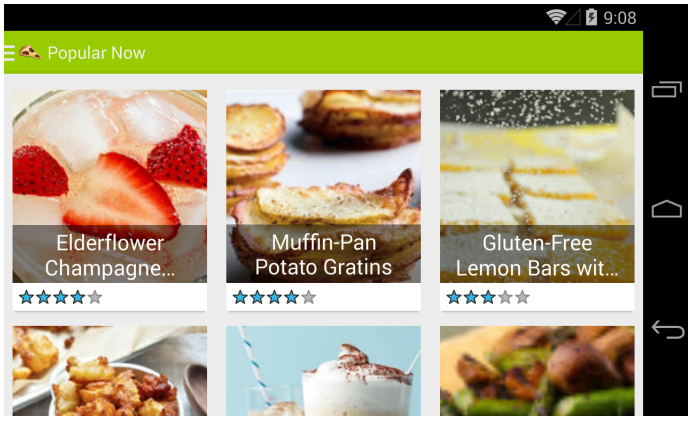


Fragments in this case are each of visible screen of application. After selecting i.e. one recipe the old fragment is replacing with new (with details of recipe). To improve usability we implemented stack of fragments, it allows to return to previous fragment after clicking on back button.

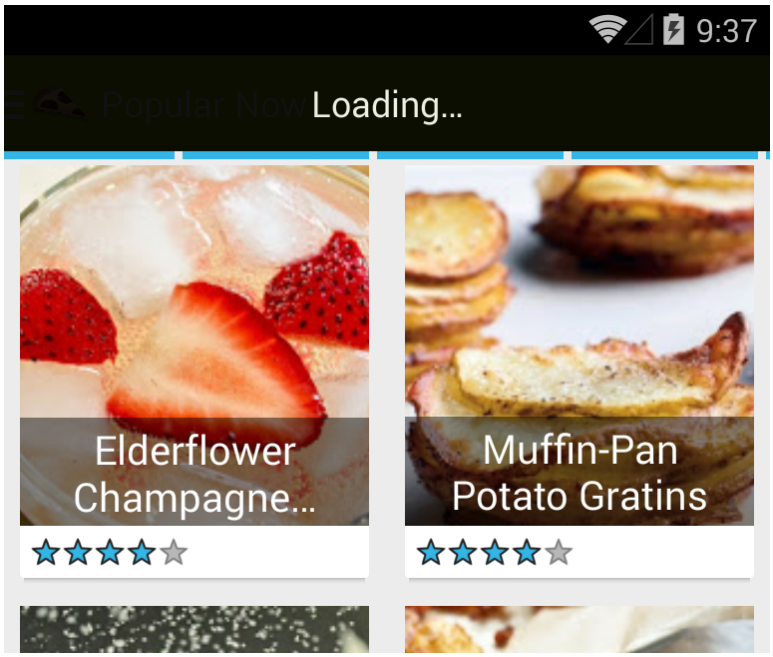
3. List of recipes

To create list of recipes we are using *cards grid*, each item is own card. Grid layout allows to use application on different screen resolutions and with landscape orientation. The number of columns is not fixed but is determined by screen with.



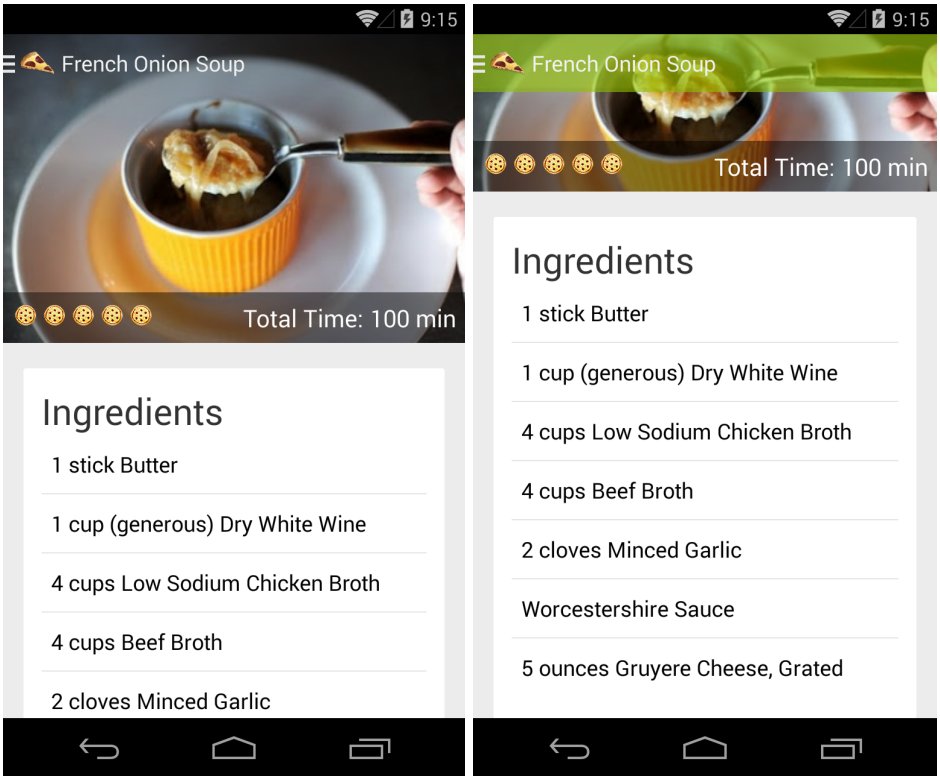


To refresh list is used pull to refresh feature (good known to google apps users, or iphone user). When we on top of the list and we pull it again we will see information about refreshing. Refreshing operation is also non blocking.



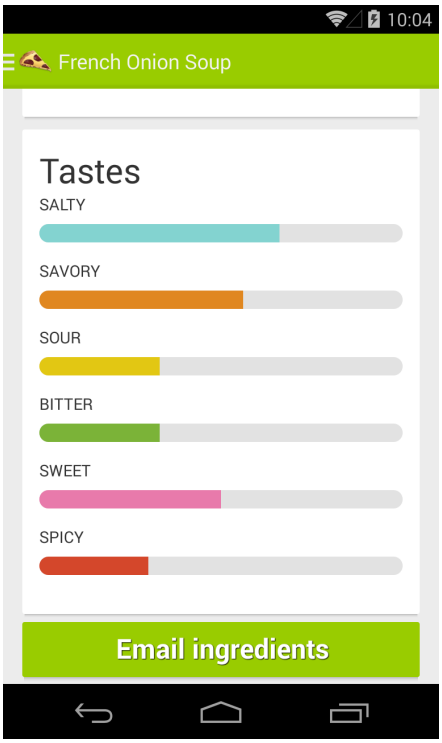
4. Recipe details

Main future of this screen is Fading Action Bar.



The image of food is behind *Action Bar* (title bar), and when view is scrolled down, green title bar is showing. To avoid illegibility of title, on top of the picture we are put dark gradient.

In details we also provide custom progress bars with different bar collars, it improve readability of presented data.



6. Sources

1. Project sources

    1. https://github.com/johniak/AndroidLayoutDesign/
    2.

2. Articles

    1. http://99designs.com/designer-blog/2014/01/15/7-unbreakable-laws-of-user-interface-design/
    2. https://developer.android.com/design/patterns/index.html
    3. https://developer.android.com/design/get-started/ui-overview.html
    4. https://developer.yummly.com/