GitHub Username: viniciusbonifacio

Don't Let Behind

Description

Lot of us just need a little heads up or some scrap of paper to remind what is not possible to do in the exact same point in the time that an event happens or a task appears. Maybe we can spend some time in a day during a coffee or a taxi ride to set the things up.

Imagine that you want to buy flowers to your girl or you got an email by a friend that still needs your piece of code to save the galaxy or even your mummy calling you to lend a hand with an social media post.

You can can keep this needs altogether in a single view to take care of it in some free time in your agenda. Just make a point and choose when to get remembered for.

Have fun!

Intended User

Anyone who needs a help in remembering short things during or between days. You, mom, dad, dog, friend, girlfriend, whatever ...

Features

Main features:

- Save information about tasks
- Show a list of tasks to conclude

Paid Features:

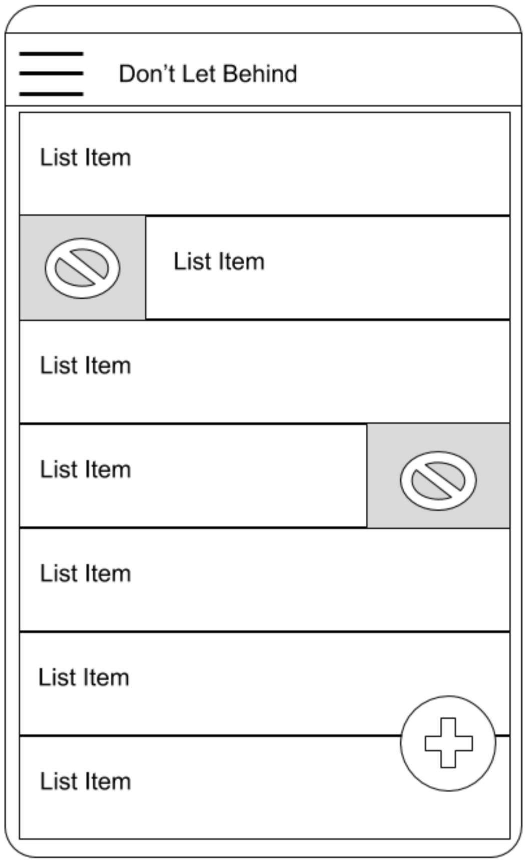
- No ads interpolation
- Tasks stored in cloud

Other features:

- Notification reminder for tasks
- App Widget button shortcut to add a task
- App Widget list of tasks to conclude

User Interface Mocks

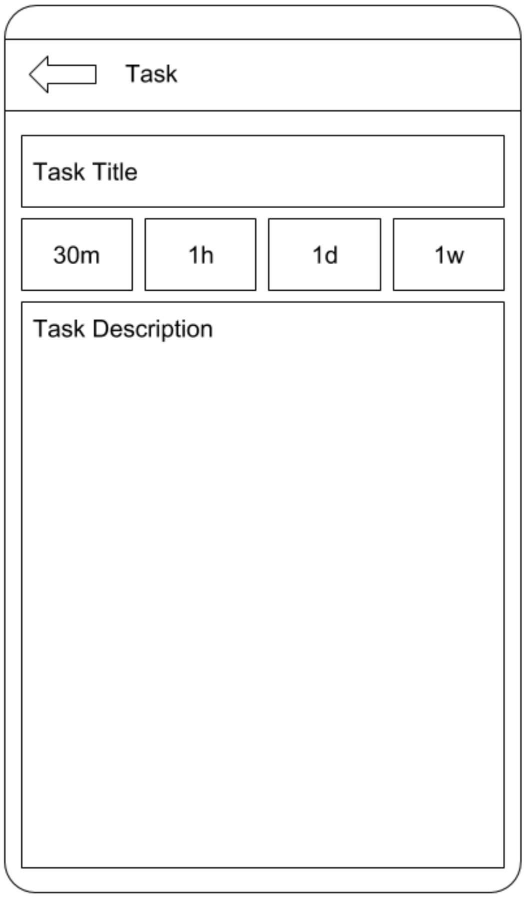
Screen 1



App - Task List

- It's the app main screen. It will show all the tasks that are not concluded.
- The app stores all tasks locally and a concluded task will be removed from the local storage.
- To conclude a task the user will swipe (right or left) activating the delete behavior.
- The float action button will start an empty Task Detail activity to register a new Task.
- All the items in list can be touched starting the Task Detail activity filled with the Task data to edition.

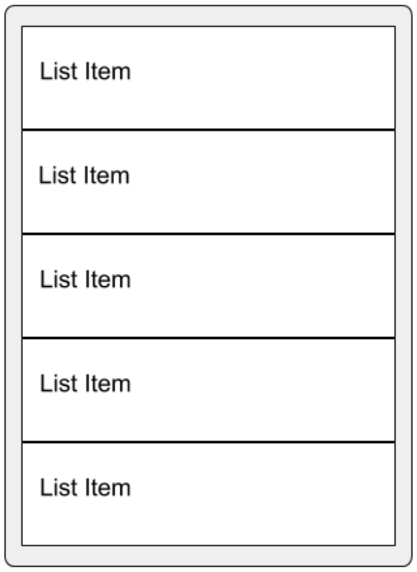
Screen 2



App - Task Detail

- This activity will be empty for a New Task action or filled for a Edit Task action.
- This activity is called by the App Widget Task List (when will be filled with data) and by the App Widget Button Shortcut (when will be empty).
- The “timer” buttons sets the trigger for a notification to remind the user about a task. Theses buttons will not be mandatory as the user can choose to not be reminded.
- I chose to not have a save button as I will take all the inputs and changes of data persisted in local storage by the focus change listener.

Screen 3



App Widget - Task List

- This App Widget will show the first 5 items in Task List that are not concluded.
- All the items in list can be touched starting the Task Detail activity filled with the Task data to edition.

Screen 4

App Widget - Button Shortcut

- This App Widget will start an empty Task Detail activity to register a new one.
- This widget is intended to be a fast way to create a Task to be reminded.

Key Considerations

How will your app handle data persistence?

As this app is intended to be a light and simple, the data is stored locally handled by a content provider. The Task entity have only 3 attributes and all the concluded tasks are removed from local storage.

In paid version the app stores Tasks data on Firebase Database storage. It is intended to permit data recovery between installations of the app, among different devices or maybe different platforms.

Describe any edge or corner cases in the UX.

The native android navigation will be sufficient to deliver the app's expected behavior.
As the app have 2 app widget and both activates the Task Detail activity by an "on touch event" there is no expected behavior other than return to Task List activity or close the app.

In Free version of the app, when the user hit Up Navigation the app interpolates an Interstitial Ad.

Describe any libraries you'll be using and share your reasoning for including them.

For the main features there is no expectation to apply third party libraries besides the Butter Knife (com.jakewharton:butterknife) as the functional requirements are well supported by the Android Framework itself.

Among the native libraries some packages like "com.android.support:design", "com.android.support:appcompat", "com.android.support:recyclerview", "com.android.support:cardview" and "com.android.support.test.espresso" are in need to be used.

For Paid version, since it brings the cloud persistence for Tasks, the Retrofit 2 library (com.squareup.retrofit2) is used in support for consuming cloud Firebase API.

Describe how you will implement Google Play Services or other external services.

For Free version the app interpolates Interstitial Ad from Firebase service. Basically, the process to work with Firebase involves adding the Firebase SDK to app's configuration file, create an account and register the app and creates and associate a project on Firebase console if not done yet.

From the app's side it is necessarily to initiate the AdMobs SDK only once at the onCreate method of an Activity, create an Interstitial AdMob Object, load and show the ad.

For Paid version the app store the Tasks on Firebase Realtime Database and it requires add the Firebase SDK to app’s configuration file and create an account and register the app and creates and associate a project on Firebase console if not done yet.

From the app’s side it is necessarily to set the access configuration for cloud storage, identify the user, define a structure of JSON data to persist, and create a component to provide an instance of Firebase database and the data transaction behaviors. The app count with offline local data storage to cache remote transactions until the device become online again.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

Task 1: Project Setup

This project is intended to be a fast and lightweight way to register and store eventually short tasks that occur during a day and can be taken care in a properly time. Keeping that in mind, the expectation is that the app can maintain a wide compatibility with older versions of Android platform.
The key expectations for the project configuration are:

- The app code is written in Java language
- Target Sdk Version: 28 (Pie, most recent at the writing of this document)
- Minimum Sdk Version: 19 (KitKat, used by 7.6% devices and is the least expressive version at the writing of this document)
- Use of google’s repository:
  - google() for gradle 4.x+, Android Studio 3.x+ and gradle plugin 3.x+
  - maven custom url "https://dl.google.com/dl/android/maven2/" for other versions of gradle, android and gradle plugin
- Declare the use of Firebase SDK dependency
  - Realtime database service
  - AdMob Ad service
- Create and config the project for digital signature for debug and release APKs

Task 2: Implement UI for Each Activity and Fragment

- Build UI for Task List activity
  - Build UI behavior onTouchEventListener

- Build UI for Task List item
  - Build UI beavavior of swiping items

- Build UI for Task Detail activity
  - Build UI Widget for reminder options

- Build UI for App Widget Task List
  - Build UI behavior onTouchEventListener

- Build UI for App Widget Button Shortcut
  - Build UI behavior onTouchEventListener

Task 3: Create Local Storage Provider

- Create the local storage entity classes
- Create the local storage database configuration classes
- Create the local storage provider classes
- Create the Loader classes to delivery data to the views

Task 4: Create Notification Component

- Create the layout for notification reminder
- Create a component class for the notification task reminder feature

Task 5: Create Task List Widget Provider

- Create the app widget provider class to the Task List widget

Task 6: Create Button Shortcut Widget Provider

- Create the app widget provider class to the Button Shortcut widget

Task 7: Review Instrumented and Unit Tests

- Review unit tests and instrumented tests classes

Task 8: Review Material Design Guidelines

- Create a new app theme that inherits from AppCompat

- Review the use and material specifications for appbars and toolbars
- Review the use and configuration of activities transitions
- Review the application of material design guidelines and key concepts

Task 9: Review Usability Requirements

- Content description
- D-Pad navigation
- RTL layout

Task 10: Review Literals in Code

- Extract the literal values to values files (Ex: strings.xml, dimensions.xml, numbers.xml, colors.xml)