Software Analysis for Omni-Notes

Guowei Li, Dongxin Xiang, Jing Chen

1.System Architecture

Software architecture is the set of principled design decisions about the system. Architectural recovery is the process of determining a software system's [as-implemented] architecture from its implementation-level artifacts.

1.1 Grouping

Based on the folders created in Omni-Notes, we categorized all the classes into several packages with classes not related to other classes removed in the UML diagram. The processed UML is shown in Figure 1-1-1.

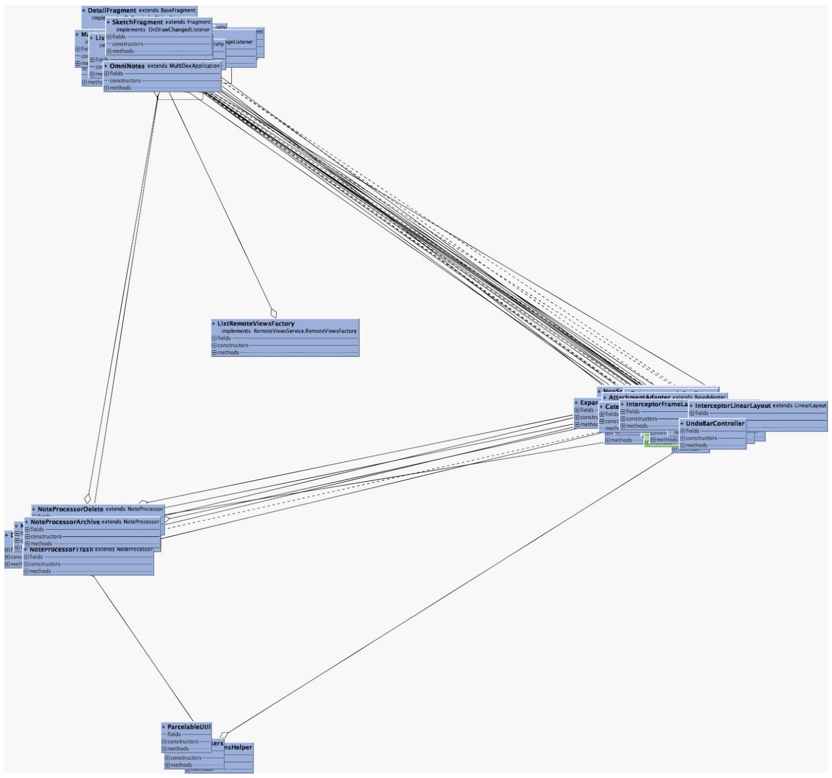

DetailFragment extends BaseFragment
SketchFragment extends Fragment implements OnDrawChangedListener
OmniNotes extends MultiDexApplication
ListRemoteViewsFactory implements RemoteViewsService.RemoteViewsFactory
AttachmentAdapter extends BaseAdapter
InterceptorFrameLayout
InterceptorLinearLayout extends LinearLayout
UndoBarController
NoteProcessorDelete extends NoteProcessor
NoteProcessorArchive extends NoteProcessor
NoteProcessorTrash extends NoteProcessor
ParcelableUtil

Figure 1-1-1 Categorized UML Diagram for Omni-Notes

We checked relationships among the classes in the UML diagram by looking through the source code. Finally, we did abstraction and organization to these classes and their relationships. As a result, the system architecture of Omni-Notes is shown in Figure 1-1-2.

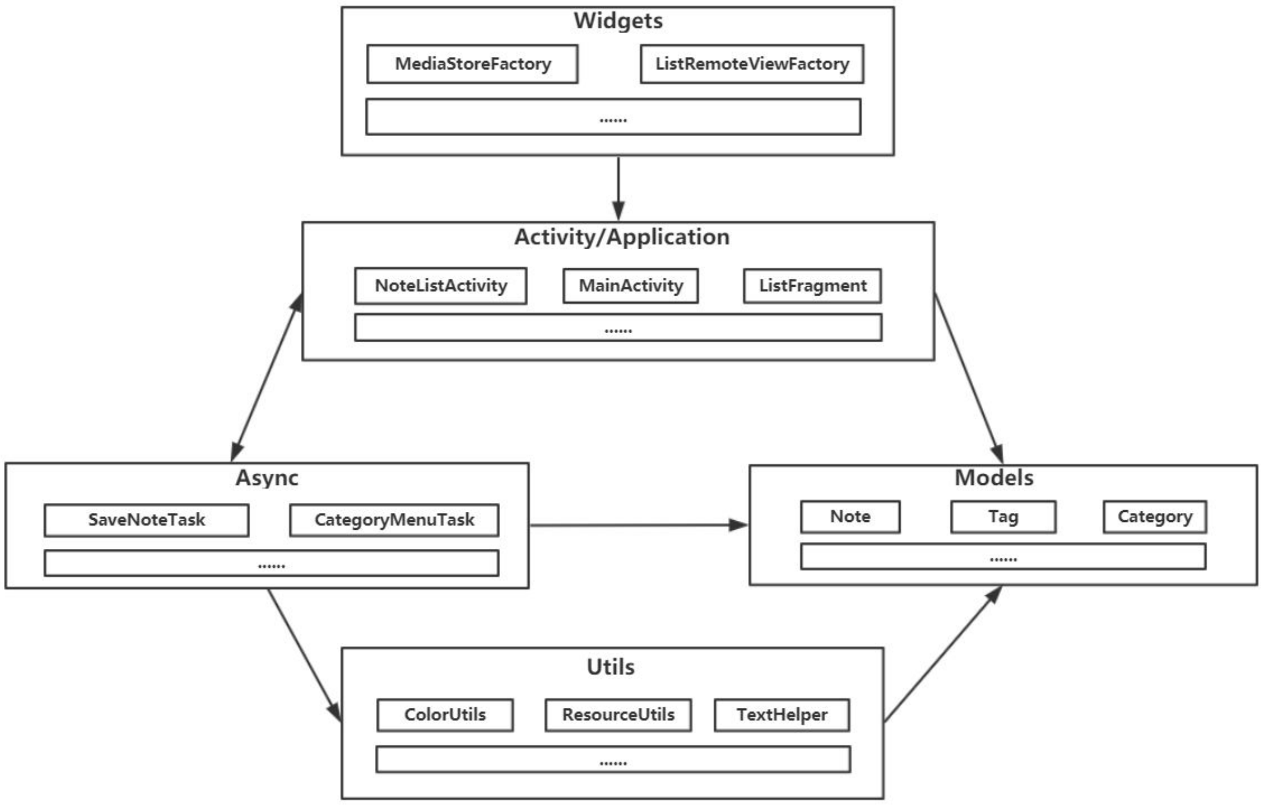


Figure 1-1-2 Software Architecture of Omni-Notes

1.2 Functionality & Architecture Analysis

Figure 1-1-2 shows a high-level software architecture of Omni-Notes. Packages work in collaboration through the relationships revealed in the figure. It is clear that there are five main packages in it, which play different main roles in the system.

The application/activity package basically contains Android activities and fragments which provide UI and can interact with users. For example, the home screen is created by *MainActivity. ListFragment* is loaded with a shortcut list view of existing notes on the home screen by default. When a user clicks on a specific note or the plus button (to create a note), *DetailedFragment* will be loaded with a note editing view.

The widgets package is made up of subclasses of WidgetProvider, which extends a basic component type in android named BroadcastReceiver class. They provide activities with miniature application views that can be embedded and receive periodic updates. In our case, some factory classes which help update those widgets are also involved in the package.

The async package contains subclasses of *AsyncTask*. AsyncTask is intended to enable proper and easy use of the UI thread. Some of them are responsible for dealing with all kinds of note tasks like note creation, update, delete, categorization and loading in the background. For example, *SaveNoteTask* will be created by a method in *DetailedFragment* once a user finishes creating or editing a note and presses the back button. It will save the new note in the background. The others are designed to offer real-time updates on menu views in activities. For example, *CategoryMenuTask* will be invoked by *MainActivity* once the user adds or deletes categories in the category menu (calling chain: *MainActivity* -> *DetailedFragment* -> *NavigationDrawerFragment* -> *CategoryMenuTask*). It will provide a real-time update on this menu view.

The models package has various models such as *Note*, *Tag* and *Category* for Omni-Notes. These models are intermediate data structures between the application and the database. They describe the properties of entities that can be processed in classes from any other packages as a whole in the software. For Omni-Notes, the developer also keeps adapters and listeners here.

The utils package basically provides util and helper classes that contain useful methods that are related to operations on everything. Those methods are widely called in classes from the async package.

According to the functionality of each package, we found the software pattern of Omni-Notes is very close to MVC pattern. The relationship among packages can be described in Figure 1-2-1. Arrows in the diagram represent interactions between different layers. The application/activity and widgets parts can be considered as the View in MVC pattern. The asnyc and utils packages play a role as the Controller. And the models package is used as the Model. Specifically, the View obtains commands from users and invocates the Controller. The Controller then performs logical business and finally, the Model receives updated information and stores it. At the same time, the Model provides data for the Controller to process. The Controller sends the processed data to the View for display. Users will be able to see the most up-to-date information on the screen. The only violation is that a view component - application/activity directly uses classes from the models package, which is a model component.

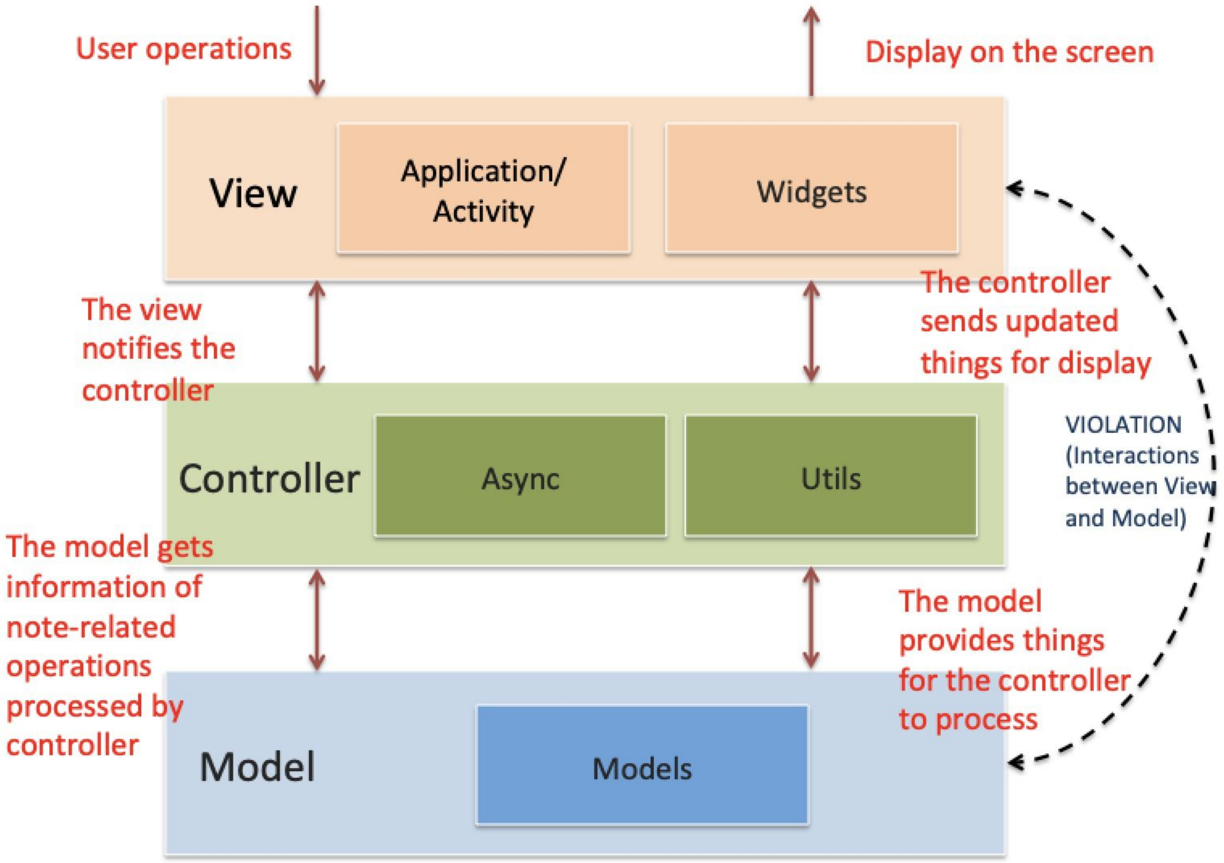

User operations
Display on the screen
View
Application/ Activity
Widgets
The view notifies the controller
The controller sends updated things for display
Controller
Async
Utils
VIOLATION (Interactions between View and Model)
The model gets information of note-related operations processed by controller
The model provides things for the controller to process
Model
Models

Figure 1-2-1 MVC Pattern for Omni-Notes

2.Social Context

2.1 Key Developers

Omni-Notes is basically a self-motivated open-source application developed by Federico Iosue (Github: federicoiosue).

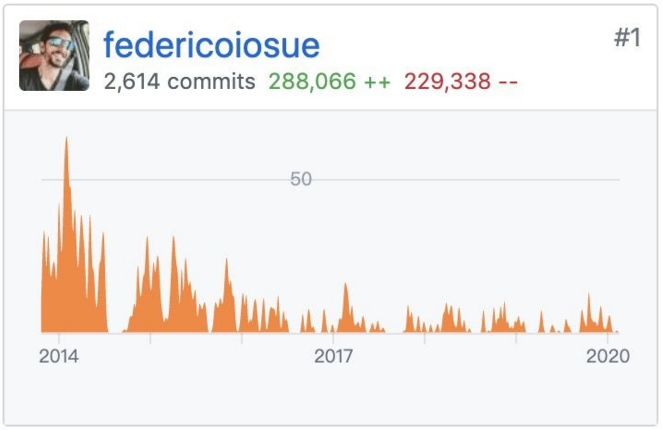


Figure 2-1-1 The Screenshot of Iosue's Contributing History

Iosue first started the development in October 2013. After that he has been contributing to this app throughout all these years, implementing almost all functions by himself. There were 19 other contributors before 2019, but they did not make many commits and generally. Their work was to fix bugs instead of implementing new features. Therefore, we believe Iosue is the only key developer for Omni-Notes.

2.2 State of the Project

2.2.1 State Overview

- Level of usage: 100,000+ installs (Google Play)
- Rate: 4.0/5.0 (Google Play)
- Last release date: Nov 16th, 2019 (v6.0.5)
- Number of commits as of late: 2988
- Number of open issues as of late: 147
- Number of active developers: 1

2.2.2 Release Roadmap

So far there have been 127 releases for Omni-Notes. To make our report readable and focus on the point, we would only list milestone releases here.

Version	Release Date	Description of New Release
3.0.0	Nov 13th, 2013	N/A
4.0.0	Dec 27th, 2013	N/A
5.0.0	Aug 17th, 2015	☆ Material design UI ☆ Customizable floating action button

		☆ Exceptionally improved image loading! ☆ Recurrent reminders ☆ Support for animated gifs ☆ Filterable reminders: selectively include or not past reminders ☆ Dynamic menu: only used items are shown within navigation list (option in settings) ☆ Application size reduced by 20% ☆ Choose to keep or delete merged note after merge succeeded ☆ Crash reports can now be deactivated from settings (not recommended)
5.1.0	Aug 17th, 2015	★ App intro ★ Filter archived notes inside categories ★ Statistics available ★ New version available check ★ Merging notes will now keep the first valid category, position, and reminder from selected notes ☆ Optionally disable swipe action in the notes list ☆ Checklists layout and functionality improvements ☆ Bug fixes
5.2.0	Jan 15th, 2016	★ Material design color chooser for categories ★ Decide whether to display attachments above or below note content ★ More options long-clicking attachments in detail ★ Widget to start writing a new note in a single touch ★ Marshmallow's runtime permissions support ★ Set a reminder to multiple notes at the same time ☆ Notifications will now survive to reboots if not dismissed (activation from settings for Android versions major than Jelly Bean) ☆ Added labels to floating action button
5.2.13	Jun 3rd, 2016	☆ Archived notes are now hidden within home widgets ☆ Updated Dutch, Polish and Spanish translations (thanks translators!) ☆ Navigation change issue when app is opened from home widget ☆ Broken share with ON function when attachments are included on Android Marshmallow ☆ Geolocation runtime permission request when not needed ☆ Sharing notes with audio recordings

5.3.0	Feb 21st, 2017	★ FOSS version available on F-Droid issue ★ Enjoy the renewed help page (accessible from inside settings) that makes use of animated tutorials ★ Added Serbian Cyrillic language (thanks Slobodan Simic) ☆ New checklist items are now capitalized ☆ Category deletion button no more is shown on creation ☆ Notes without title are now shown as they into list instead of using part of the content to improve performances ☆ Updated translations ✓ Memory leaks related to checklists and geolocation ✓ Hidden attachments from locked notes into home widget
5.4.0	Dec 11th, 2017	★ Added file size to attachments action dialog ★ When editing a note's reminder, if that's in the past, a future date will be proposed ★ Added an option to switch timestamps ON or OFF into home widget ☆ Revisited contextual actions when multiple notes are selected ☆ Improved memory usage ☆ Different text size in the "Text size" settings ✓ Bug fixes Thanks a lot to Dima-1 for his contributions!
5.5.0	Apr 26th, 2018	★ Rapidly search your unfinished to-do lists ★ New "attachment" available to easily add the current timestamp ★ Now an option is available into the editor to check currently edited note's information ★ Persian translation ☆ Removed outdated transparent navigation bar: this will save a lot of CPU avoiding calculating dynamic layout, after all, today all devices are very much bigger than when this optimization has been introduced ☆ Removed semi-transparent overlay when using floating action button to speed-up the flow ✓ Notes list update after password reset and password request bypass using swipe actions ✓ Language switching on Android Oreo and later versions ✓ First title duplication during notes' merging

6.0.0	Oct 21st, 2019	★ New backup mode using JSON: older backups can still be restored with "legacy" restore option ★ Included SDK 28 fixed and improvements ★ Added Finnish and Romanian translations ☆ Translations update ☆ Much faster images loading ☆ Easily recognizable shortcut widget ✓ Broken share-with-ON when receiving images ✓ Reloaded notes after password reset displaying an explicit message ✓ Ignored case when sorting notes ✓ Broken notifications on Android Oreo and following
6.0.5 (Latest Release)	Nov 16th, 2019	☆ Confirmation required when undoing note changes ✓ Fixed notification on Oreo devices

★ New features ☆ Improvements ✓ (Bug) fixes

2.3 Standards to Which We Should Adhere

2.3.1 Coding Standards

The implemented code must follow the specific code style of Omni-Notes. According to the *README.md* file in the project on Github, the base code style is Google Java Style. On top of that, Iosue made some changes to it:

- Hard wrap at 120 chars (instead than 100)
- Single space before method declaration parenthesis
- Simple lambdas in one line
- Simple methods in one line
- Simple classes in one line
- Chained method calls: do not wrap
- Chained method calls: align when multilined

All these rules make up the coding standards of Omni-Notes. A Java source file is described as being in Omni-Notes' style if and only if it adheres to the rules herein.

2.3.2 Testing Standards

The *README.md* file in the project on Github only covers that to execute all tests included in the project connect a device or emulator, then run the following command:

```
./gradlew connectedAndroidTest
```

However, we found from past work on Omni-Notes that there are other testing standards that are actually applied but not covered in the documentation. For example, any contributor should at least add unit tests if they implement new methods/ functions/ features. Contributors should improve test coverage. These are particular concerns of adequate testing[1]:

- Is every public function and class tested?
- Are a reasonable set of parameters, their values, value types, and combinations tested?
- Do the tests validate that the code is correct and that it is doing what the documentation says the code is intended to do?
- If the change is a bug fix, is a non-regression test included?
- Do the tests pass the continuous integration build?
- Do the tests cover every line of code? If not, are the exceptions reasonable and explicit?

2.3.3 Documentation Standards

Since the base code style is Google Java Style, Omni-Notes also obeys rules about Java doc format in it.
For documentation, it's recommended to use .md files like *README.md*, thus contributors need to obey Markdown grammar. There are best practices for documentation[2]:

- Focus on user intent and audience.
- Use every-day words and keep sentences short.
- Use consistent sentence construction, wording, and capitalization.
- Use headings and lists to make your docs easier to scan.
- The Google Developer Docs Style Guide is helpful.

2.3.4 Submission Standards[3]

- Read contributing guidelines.
- Read the Code of Conduct.
- Check if your changes are consistent with the guidelines.
- Check if your changes are consistent with the coding style.
- Run unit tests.
- Check if your modification in one pull request only fixes one issue.

2.4 Typical Process of Contributing

2.4.1 Process Description

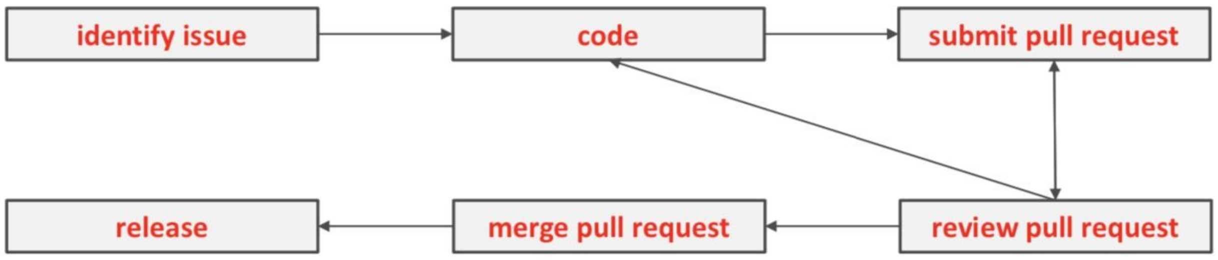


Figure 2-4-1 Standard Contributing Process for Omni-Notes

Anyone can feel free to add themselves to contributors.md file if they want to make a contribution to Omni-Notes. To make a contribution, the first thing is to identify an issue from existing usages of the app or the source code. There are many features/ improvements that are not on Iosue's roadmap but a contributor could decide to work on them anyway. Iosue suggests that contributors hunt for issues in the open issues section, especially issues with tag "Help wanted".

Once an issue is identified, contributors can work on it with any IDEs/editors they feel comfortable to use. It's recommended to create a "develop" branch in Git when making contributions to the software. When implementations are done, all code changes and additions should be tested (unit and UI tests). Contributors are also responsible for avoiding introducing technical debt like code repetition, missing exception handling, hard-coded text strings, preferences management in the main thread, commented code rows.

The next step is to submit a pull request. Iosue will review every pull request and decide if they can be merged based on the rules described above in section 2.3. If the contributor does not follow the rules, his/ her request will be closed and the contributor needs to review the work again. A new pull request is needed when a contributor wants to submit the work again. If merged, the contribution will appear in the next release of Omni-Notes.

2.4.2 Tools Support the Process

2.4.2.1 Version Control/Source Code Repository

Git is used as the version control system.

2.4.2.2 Issue Tracker

(Not specified) Options: Zendesk, Freshdesk

2.4.2.3 Build Tool

Gradle is used to build the project.

2.4.2.4 Test Tool

(Not specified) Options: Selenium, Cucumber

2.4.2.5 Code Review Tool

SonarQube is used as the code review tool for the project.

2.4.2.6 Security Checker Tool

(Not specified) Options: Ostorlab, Appvigil

2.4.2.7 Continuous Integration Tool

Travis CI is used for continuous integration and testing.

3.Five “Interesting” Pull Requests

Pull Requests 1: Added fast scroll functionality and some layout changes inside the note view #643

- Url: https://github.com/federicoiosue/Omni-Notes/pull/643
- Creation date: Dec 22nd, 2018
- Current status: Open
- Summary:
  This pull request was intended to provide a solution for two open issues back then. One issue (#523) was the search and scroll bar feature request. A user claimed that there should be a scroll bar for viewing through long notes instead of swiping the screen. The other (#434) was a layout modification request for the note editing view. Hisham Mahfouz (Github: HM201) implemented *FastScrollDelegate.java* and *FastScrollScrollView.java* to fix issue #523 and modified layout of the note editing view based on the suggestion made in issue #434. The pull request is still open. Iosue believed that issue #523 was not completely fixed by Mahfouz’s modification since his code missed both UI and unit testing. On top of that, it introduced a new problem with the position of the scrollbar. Iosue has been trying to fix those issues by himself till now and this open pull request stays as a reminder for further implementations.
  It’s believed “interesting” because we realize from it that there should be a balance between the real implementation and users’ expectations for features. Actually we think the current layout design is good enough. Sometimes it’s hard or unnecessary to add all

features they want to software and we can’t make everyone totally satisfied with the software.

Pull Requests 2: Delete home screen shortcut #115

- Url: https://github.com/federicoiosue/Omni-Notes/pull/115
- Creation date: Jan 23rd, 2015
- Current status: Merged (Jan 26th, 2015)
- Summary:
  This pull request was about fixing a functional bug. Antoine Saliba (Github: antoinesaliba) claimed that when a note got deleted in the app, the shortcut of it on the home screen would not automatically disappear, instead, an error occurred when the shortcut was clicked. He changed code in *DetailFragment.java* by adding a method **removeshortCut()** and calling it in both **trashNote()** and **deleteNote()** so that the shortcut would disappear as expected after a note got trashed/deleted. This pull request got merged. Later Iosue implemented a class named *ShortcutHelper*. **removeshortCut()** got moved to *ShortcutHelper.java* together with other newly implemented methods related to shortcut updating.
  There were many functional bugs in Omni-Notes. Most of them were minor bugs that could be fixed with modification within several lines of code. We think it’s “interesting” because it’s relatively more complicated to solve and it reminds us of the importance of software testing. As the only key developer of Omni-Notes, Iosue cannot capture all functional bugs without users and other software testers. Software needs to pass a large number of tests to be mature.

Pull Requests 3: Just made the entire code more readable (read description) #709

- Url: https://github.com/federicoiosue/Omni-Notes/pull/709
- Creation date: Oct 16th, 2019
- Current status: Closed (Oct 21st, 2019)
- Summary:
  This pull request was not about to solve any technical issues. It aimed to improve the readability of code. Muhammad Aqib (Github: lionuncle) changed 33 java files by clearing extra spaces and adding spaces where needed so that the code could look more comprehensive and readable for other developers. This pull request was closed by Iosue. He couldn’t merge it because he’d have to check too much code to make sure there were not any unexpected changes that may undermine the functionality of Moni-Notes. However, Iosue realized from the pull request that:
    - ❏ Create a formal code style for the project.
    - ❏ Perform whole Java classes code style refactoring when it’s necessary.

- ❏ Add instructions for contributors.

From our own experiences, code that has good readability is easy for developers to understand and maintain. To ensure a smoothly running software development, it's necessary to make code readable.

Pull Requests 4: Improve RTL #636

- Url: https://github.com/federicoiosue/Omni-Notes/pull/636
- Creation date: Dec 15th, 2018
- Current status: Merged (Jan 1st, 2019)
- Summary:
  This pull request is intended to improve the RTL layout. Mostafa Ahangarha (Github: ahangarha) was a user of Omni-Notes. He found it couldn't display Persian in a proper way - from right to left. He then improved the RTL layout by replacing left/right layout properties with relevant start/end. In that case, the layout could automatically switch between LTR mode and RTL mode based on the language the user set in Omni-Notes. The pull request got merged after Iosue added relative unit tests.
  The pull request looks "interesting" because Iosue, the creator of Omni-Notes, is from Italy. Italian and most other languages in the world are LTR. This bug may have existed since he first created Omni-Notes. Then Omni-Notes has gradually gained popularity. Some users are from many countries whose official languages are RTL. At this time, it's increasingly necessary to fulfill their need to make the software available all over the world.

Pull Requests 5: Add a word counter #437

- Url: https://github.com/federicoiosue/Omni-Notes/pull/437
- Creation date: Feb 12th, 2018
- Current status: Closed (Mar 24th, 2018)
- Summary:
  This pull request is "interesting" because it does not seem like a real pull request. Someone (Github: nosenses) talked about his/ her feature request - to add a word counter in it without actually modifying any files. The pull request got closed. We believe the person used pull requests in the wrong way. Creating a new issue about this in "Issues" is the right thing to do.

4.Five “Interesting” Open Issues

Issue 1: Search within Text Note #622

- Url: https://github.com/federicoiosue/Omni-Notes/issues/622
- Context (if applicable): N/A
- Description: Some users have text notes with a lot of text, so it would be very useful if Omni-Notes allows users to search within the text note itself and make found words highlighted.
- Expected behavior: Described above.
- Summary: This open issue is “interesting” because it’s still among unsolved issues while we may be able to contribute to implementing this feature. First, we need to add an EditText for search, maybe along with a search button in the layout of the note editing view. Then we may add methods in *DetailedFragment.java* for searching within a note and highlighting found words. We are definitely interested to give it a try if it’s a task for the next homework or there’s enough spare time.

Issue 2: Bug when splitting windows/multitasking #456

- Url: https://github.com/federicoiosue/Omni-Notes/issues/456
- Context (if applicable): N/A
- Description: Omni-Notes does not seem to be working well when using the Multi-window view, the split window or the multitasking thingie.
- Expected behavior: It’s better if Omni-Notes can work normally in a Multi-window view.
- Summary: This open issue is a feature request. We believe it is necessary to add this feature. Some Android apps support multi-windows mode. For a note-taking app, supporting multi-windows mode is a must to have. For example, users are able to open a window with a text-image and write at the same time in the note-taking app.

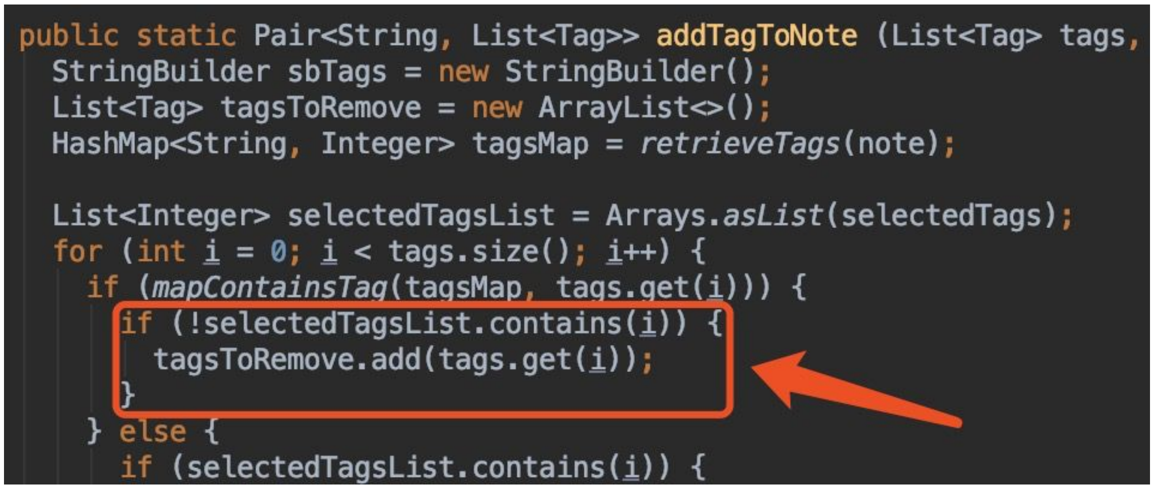
Issue 3: Test coverage report #383

- Url: https://github.com/federicoiosue/Omni-Notes/issues/383
- Context (if applicable): N/A
- Description: Some sort of code testing coverage should be prepared to allow SonarQUBE to include that data.
- Expected behavior: Described above.
- Summary: This issue is opened by Iosue. His purpose is to remind himself of improving testing coverage and validating code quality with SonarQUBE. This is one of these lessons we learned from reading these open issues - test cases should cover as much code as possible to efficiently avoid potential bugs.

Issue 4: Unexpected behavior on tag assignment #517

- Url: https://github.com/federicoiosue/Omni-Notes/issues/517
- Context (Where the issue is confirmed by our group):
  - Device: Nexus 5X
  - OS version: Android 7.1.1
  - App version: 25

- Description: When users try to add new tags to a selection of multiple notes, the operation would remove all old tags that are already assigned but not among these new tags.
- Expected behavior: Omni-Notes is supposed to simply add new tags that have never been assigned to selected notes while their old tags should be kept.
- Summary: The issue is about an unexpected behavior of Moni-Notes. However, according to the source code in *TagsHelper.java,* we can be pretty sure that the issue lies in the implementation of the method **addTagToNote()**. It may probably be fixed by removing the inner if block in the piece of code below:



```
public static Pair<String, List<Tag>> addTagToNote (List<Tag> tags,
  StringBuilder sbTags = new StringBuilder();
  List<Tag> tagsToRemove = new ArrayList<>();
  HashMap<String, Integer> tagsMap = retrieveTags(note);

  List<Integer> selectedTagsList = Arrays.asList(selectedTags);
  for (int i = 0; i < tags.size(); i++) {
    if (mapContainsTag(tagsMap, tags.get(i))) {
      if (!selectedTagsList.contains(i)) {
        tagsToRemove.add(tags.get(i));
      }
    } else {
      if (selectedTagsList.contains(i)) {
```

Figure 4-1-1 Code Snippet of **addTagToNote()**

Iosue might want to make Omni-Notes update tags on notes while the tag assigning operation, but somehow he implemented it in the wrong way by adding unnecessary code block to it.

Issue 5: Crash on note swiping #740

- Url: https://github.com/federicoiosue/Omni-Notes/issues/740
- Context (Where the issue is confirmed by our group):
  - Device: Nexus 5X

  - OS version: Android 7.1.1
  - App version: 25
- Description: Swipe any (to the left or right) in the note list on the home screen and the application would crash and show an error message toast “Something went wrong in Omni-Notes\n A report has been sent to the developer!”.
- Expected behavior: The swipe motion is supposed to work like a normal delete operation. The note that gets swiped should be moved to trash.
- Summary: According to the description, the error message shows that “NoClassDefFoundError for it.feio.android.omninotes.alpha java.lang.NoClassDefFoundError: Failed resolution of: Lcom/nineoldandroids/view/ViewHelper;”. It is probably an issue resulting from using an out-of-date library named NineOldAndroid. It has been deprecated in recent Android versions. It is better to use another library such as SmartSwipe, which is compatible with new Android versions, to implement the swiping feature. It is “interesting” because it reveals the fact that developers should pay attention to the software’s compatibility for different Android versions when updating it since each new Android version changes some libraries from the old version, which may bring compatibility issues.

References

[1] Contribute to the TensorFlow code - Test and improve test coverage.
https://www.tensorflow.org/community/contribute/code#test_and_improve_test_coverage
[2] TensorFlow documentation style guide.
https://www.tensorflow.org/community/contribute/docs_style
[3] Contribute to the TensorFlow code - Contributor checklist.
https://www.tensorflow.org/community/contribute/code#contributor_checklist