LibreOffice for Web / Mobile

In the past 10 years, TDF has invested into providing web and mobile support for LibreOffice. Although there were some releases, both Mobile and Online products were abandoned for non-technical reasons. Around 5 years have been lost since then, but there is still room for providing usable and competitive web and mobile applications.

This proposal is written to address the needs that have been discussed over the past decade. The goal is to hire in-house developers to continue the development of web and mobile support for LibreOffice.

Market Analysis

While desktop will remains a strong platform, users demand that their office productivity of choice remains accessible in the newer platforms, including web and mobile.

Statista estimates the market for productivity software in 2025 to be around $80 billion in which office software holds the biggest share, around $30 billion. Consainsights estimates[1] that cloud based solutions that provide real-time collaboration features will grow from $36.44 billion in 2023 to $71.61 billion by 2033. The value estimated for 2023 is more than 86% of the total market value estimated for the office productivity software in the same year.

The market is dominated by a duopoly, Google and Microsoft, which provide their online productivity suites that include online office software alongside other productivity tools. Google Docs is estimated to have 1 billion active users per month and over 6-8 million paying customers[2] in 2025, in comparison to 321 million monthly active users of Microsoft and 345 million paying subscribers[3] in the same year. It is important to know that the Microsoft 365 paid subscription includes the desktop office software, but also provides access to online and mobile[4]. This shows the importance of platform convergence for gaining more users.

While TDF provides a well-know office productivity suite for desktops, it provides almost nothing in the area of web and mobile. In order to become relevant in next decade, it is essential to use the potential of the huge LibreOffice codebase which supports many different formats with high fidelity to the original design to something that is available across different and modern platforms.

It is also important to mention that nearly all of the companies who provide office software for desktop, provide their version of web/cloud and mobile. That is another key factor for adoption of the office

1 https://consainsights.com/reports/office-productivity-software-market
2 https://explodingtopics.com/blog/google-workspace-stats
https://www.patronum.io/key-google-workspace-statistics-for-2023
3 electroiq.com/stats/microsoft-365-statistics
4 https://www.microsoft.com/EN/microsoft-365/buy/compare-all-microsoft-365-products
Note that separate office license can be obtained, but it is only a small fraction of the Microsoft revenue, and Microsoft actively promotes and advertises 365 as the best value buy.

software, as it is valuable for the users to choose a product that is also available on the other platforms other than desktop.

The strong and steady growth in the market, which is estimated to be ~15% annually[5] from 2023 to 2030, alongside the effect of platform convergence shows the necessity that TDF provides web and mobile versions of LibreOffice.

Rationale for Hiring

These are some of the reasons that were previously cited for hiring in-house developers, which are still relevant and were proved to be correct with the previous developer hirings:

- Increasing TDF code contribution: statistics in LibreOffice core commits shows significant increase in the number of commits from TDF.
- Making TDF a regular code contributor: the steady stream of patches and bug fixes by TDF developers shows this was a realistic goal. With the next round of hiring, it is expected that more areas of the code that was neglected for several years will receive improvements.
- Spending more money on development: this goal, which was demanded by the members of the ecosystem and others, is now a reality. But, there is still room for improvements within the TDF budget.
- Bug fixing: the fact that some of the bugs have been neglected for years, and some important bugs were untouched more than 10 years are now being addressed by the developers who work on specific areas of LibreOffice including RTL/CTL/CJK and accessibility. Building on this background, and the focus areas of the suggested hirings, it is expected to see the improvements in relatively short amount of time.
- Building internal skills: understanding of the code and growing internal development capabilities are crucial for TDF. Sharing the knowledge for increasing the number of contributors and external suppliers, ability to evaluate tenders and deliverables, and developing software where others are not able or willing to invest are among the reasons.
- Supporting new contributors: to support new contributors, it is important to have people who can mentor new contributors in the diverse LibreOffice code base. Hired developers are expected to be able to do mentoring in the corresponding areas of the code that they work on.
- App store revenues: sales of LibreOffice on the app store now provides an important revenue source for TDF. Building on that success, it is expected to provide new applications on various app stores.

These goals will be out of reach without having internal developers focusing on the areas of web, mobile and some other areas that are neglected by other developers in the past years. Considering

5 https://www.linkedin.com/pulse/cloud-office-services-market-trends-growth-rate-insights-nbx3f

successful TDF hirings for a11y, RTL/CTL/CJK justifies further investment into hiring developers who can bring LibreOffice into the popular platforms of web and mobile.

Technologies and Focus Areas

The underlying technologies used in LibreOffice Online, and also mobile version of LibreOffice is the same: LibreOfficeKit. Therefore, it is helpful to discuss things in a single document, rather than discussing things in different proposals.

LibreOfficeKit itself is written in C/C++. It is the back-end technology used to open, render and deliver document tiles to LibreOffice Online and also mobile versions of LibreOffice. LibreOfficeKit is used as a back-end library.

On the other hand, different front-end technologies have been used in web and mobile, which is platform-specific. For each of them, different programming language(s) are used, but any of these languages are well-known and widely used on that specific platform.

The below table is the proposal in short:

Focus areas for the proposed developer hiring

Focus Area	Programming language / Technology	Hire count
Back-end for Online	C/C++	1
Front-end for Online	JavaScript, TypeScript	1
Android	Java, C/C++	1
iOS	Objective C, Swift, C/C++	1

LibreOffice Online

LibreOffice Online source code repository is frozen since 2020, around 5 years ago, and was atticized in 2022. The freezing was supposed to be temporary. It is now stored as a read-only repository, where no contribution is possible[6].

Since then, there has been various improvements in LibreOffice itself, LibreOfficeKit, and Online counterparts in other repositories. It is important that LibreOffice Online can use these improvements.

LibreOffice Online consists of various modules, but it can be divided into two main parts:

1. Server side: mostly written in C/C++
2. Client side: mostly written in JavaScript/TypeScript

First, one back-end developer, which should be fluent in C/C++. And second, one front-end developer, focusing on JavaScript/TypeScript development.

6 https://git.libreoffice.org/online/

These are some of the possible focus areas of the developers, which are areas for future improvements of LibreOffice Online:

1. Wasm integration
2. E2EE
3. Improve language support
4. Better administration page
5. LibreOfficeKit stability
6. Improve server side performance
7. Improve multi-process
8. Improve storage back-end
9. Document LibreOfficeKit
10. Document LibreOffice Online

Each of the above areas of improvements are discussed below.

1. Wasm integration

WebAssembly[7] (Wasm) is a way to compile native applications from source code to run inside a browser. There has been experimental ports of LibreOffice to Webassembly, and this paves the way to also use it for LibreOffice Online.

There can be two main benefits for this approach:

1. Solving latency issues: one of the main disadvantages of LibreOffice Online tile based approach is latency, which is a major issue: high latency means dissatisfaction of the users, and make the web application hard to use. With Wasm, LibreOffice will run inside the user's browser, thus removing that huge latency with client/server approach.

2. Offloading server load: LibreOffice Online creates a considerable load on the server. With Wasm port of LibreOfficeKit, this issue will be resolved, as the server will only provide the some download, and that can happen only the first time.

Efforts to port LibreOffice to WebAssembly was done by different companies and funding sources, even with the name of LibreOffice Online[8], and the result is also present on different source code repositories. The current results are considered experimental, and not production-ready. The goal here is to improve the situation and consolidate different efforts.

7 https://en.wikipedia.org/wiki/WebAssembly
8 https://web.archive.org/web/20250620104150/https://nlnet.nl/project/LibreOfficeP2P
https://community.documentfoundation.org/t/questions-on-nlnet-fundings-with-the-libreoffice-name/13065

2. E2EE

E2EE (End-to-end encryption) means only communicating users will see the un-encrypted contents. In other words: "No one else, including the system provider, telecom providers, Internet providers or malicious actors, can access the cryptographic keys needed to read or send messages"[9].

At the moment, what users are editing is visible for the server (if not others), and the administrator can easily download or modify the document. This is not always desirable. There has been some efforts to make E2EE possible for online editing, but no viable product provides that functionality yet.

This feature is tied to WASM support, and will not be possible without it. Yet it will require additional development efforts. Therefore, it is listed as a separate working area for improvement.

3. Improve language support

While LibreOffice Online works well with English and some other western languages with Latin script, there are serious issues with other languages, including RTL/CTL/CJK. Some of the documents do not load at all, and some lead to complete hang of the server. Using fonts lead to problems, and there are many other issues affecting the non-western languages.

Improving the overall language support of LibreOffice Online and properly testing it is an important focus area, which can gather more users across the world.

4. Better administration page

One can access administration page in “Admin Console”[10] of LibreOffice Online. It has few pages with brief contents like:

Overview, Analytics, History, Log, Settings

It provides very basic information about the server and the software, while most of the configurations are done through the XML configuration file.

There can be obvious improvements for this administration page, including:

- More details about
  - Status of the server
  - LibreOffice version and settings
  - LibreOffice Online version and settings
- Means to
  - Install add-ons, fonts, etc.
  - Mange server, users, etc.

9 https://en.wikipedia.org/wiki/E2EE
10 Default address: http://localhost:9980/browser/dist/admin/admin.html

    - Configuring the system (more important on the first run)
- Display of
    - Help/manuals
    - Information about the software and contributors
- Better login page, instead of browser based authentication

Some of these features may be available elsewhere, for example in NextCould, but it makes sense to have such features inside LibreOffie Online's administration page.

5. LibreOfficeKit stability

LibreOfficeKit is the underlying technology behind LibreOffice Online and mobile applications. It is basically a tiling and controlling mechanism for LibreOffice that makes it possible to use LibreOffice in another application, while displaying the screen via small image tiles that can be sent over network and sending mouse/keyboard interactions back to the LibreOffice.

The current LibreOfficeKit implementation is scattered throughout the LibreOffice source code, while there is a small folder in LibreOffice named `libreofficekit/` which is dedicated to tests and an example application called `LibreOfficeKitGtk`. Other than that, there are header files inside `include/LibreOfficeKit/`.

At the moment, most of the LibreOfficeKit API is marked as unstable. Only the file conversion part, which lets one convert a document to another format is considered stable. Other than that, almost everything is usable only when you add the `LOK_USE_UNSTABLE_API` definition to be able to use the unstable API. This prevents other users from using it meaningfully in their application, exactly because it is unstable.

Therefore, it is meaningful to consider adhering to some guidelines, as semantic versioning[11]. Define stable API, and support it.

Also, the unstable API does not even work on macOS[12]. That deserves a fix, which is not trivial and needs important changes including headless VCL implementation[13] on macOS.

6. Improve server side performance

While LibreOffice itself provides good performance in startup and also in loading files, Online creates overhead because of the way it works and its architecture, therefore consuming huge amount of CPU and memory.

11 https://semver.org/
12 https://bugs.documentfoundation.org/show_bug.cgi?id=145127
13 https://bugs.documentfoundation.org/show_bug.cgi?id=145127#c4

Although several optimizations including the tile caching and compression techniques was implemented in Online, the load on the servers are relatively expensive on CPU/RAM, and there is still room for improvement on how tiles are created and served.

7. Improve multi-process

LibreOffice Online uses multi-process architecture. On the other hand, there are many cases where the current architecture fails to address problems that has happened for the underlying processes. There are situations where underlying process hangs, and the whole application becomes non-responsive.

The idea here is to revise the multi-process architecture, and get insight from the architecture in modern multi-process browsers, and improve the situation in LibreOffice Online.

8. Improve storage back-end

LibreOffice Online uses file back-end for storage which causes issues, and is not scaleable. On the other hand, conventional database back-ends cause high latency. The goal is to use low latency databases that can serve as back-end for LibreOffice Online, and provide it as an alternative solution to file back-end.

9. Document LibreOfficeKit

The users of LibreOfficeKit are currently limited to LibreOffice Online and a few example applications, and there is a little amount of documentation for the way LibreOfficeKit should be properly used. Recently, some examples were added for LibreOfficeKit stable API. But, the most important part, the unstable API which is used to create LibreOffice Online is mostly undocumented. This is another focus area to improve.

10. Document LibreOffice Online

The amount of documentation around LibreOffice Online internals is also limited. Adding more documents and examples can be helpful for the future developers of LibreOffice Online. This focus area is around developer documentation, and not user manuals.

LibreOffice Mobile

The initial Android version of LibreOffice was funded by TDF[14] [15], but then it was ignored for non-technical reasons. The result was dissatisfaction from the users, leading to a low rating on the app stores like Google Play. In 2020, it was decided to remove it from the app stores.

On the other hand, in the past 2 years, there was an on-going maintenance work by TDF staff, which improved the situation, providing updated versions of the application with newer underlying LibreOffice. The Android Viewer was put back on Google play[16] and F-Droid[17] on 2023. This effort was well received by the users, and the software rating was improved on the online software markets. But, there are still shortcomings in the Android application that prevents users from using the potential that Android version of LibreOffice provides. There is no new development work and no new features, therefore it is not possible to provide any improvements as users request.

Market Analysis

While desktop will remain as a strong platform in many cases, mobile platforms including devices based on Android and iOS has grown vastly. Statista estimates the number of mobile devices to be ~18 billion[18], in other estimates > 3.5 billion active users for Android, and > 2 billion active users for iOS in 2025. Also, 1 billion Android devices are shipped each year[19].

The huge number of mobile devices including mobile phone and tablets creates a big opportunity to expand the LibreOffice user base, while presents new challenges: a mobile application needs its own UI design, usable on displays with different resolutions and different aspect ratios compared to the desktop environment. A native mobile application usually means extra development work on different operating systems, in order to provide a good user interface for the user on their platform.

Scope

This proposal is written to address the needs to have a working Android editor that has been discussed over the past years. The goal is to hire 1 in-house developer to continue the development of Android support for LibreOffice, which includes a working editor which is enabled by default.

While using a web version is possible for creating a mobile application that provides the content via a web view, the accessibility and usability of such an approach is not convincing for many users. Native applications provide much better usability and performance compared to such an approach. Therefore, it is important to develop and provide native mobile applications for better user experience.

14 https://blog.documentfoundation.org/blog/2014/09/04/tender-for-base-framework-for-an-android-version-of-libreoffice-with-basic-editing-capabilities-201409-01/
15 https://blog.documentfoundation.org/blog/2015/01/27/the-document-foundation-announces-the-results-of-the-android-tender/
16 https://play.google.com/store/apps/details?id=org.documentfoundation.libreoffice
17 https://f-droid.org/en/packages/org.documentfoundation.libreoffice/
18 https://www.statista.com/statistics/245501/multiple-mobile-device-ownership-worldwide/
19 https://www.businessofapps.com/data/android-statistics/

The current Android viewer application is a native Android application written in Java, which provides good support for the UI/UX expected by Android users. A native application is also planned for iOS devices as described in the latter part of this document.

Android Version

Currently, TDF provides two different Android applications for LibreOffice, which are for two different purposes:

1. LibreOffice Viewer (with experimental editing support)

2. LibreOffice Impress Remote

The plan is to consolidate these two applications, and improve the editing capabilities, promoting it out of experimental status to a working by default status.

1 developer hiring is proposed for this task.

These are some of the focus areas of the Android developer.

Improve Text Editing capabilities

The current editing capabilities are limited to basic editing features. On the other hand, the infrastructure facilitates complex LibreOffice actions via UNO commands. Therefore, adding some of the more complex editing capabilities of LibreOffice with native UI/UX is feasible, but needs UI work.

Ability to Work with Tables

While inserting tables in the current version of Android Viewer is possible, important modification features are missing. Modifying tables is another important feature that is expected to be implemented. A separate toolbar will be required for modifying various properties of the tables.

Ability to Work with Graphics/Objects

The current version lacks the ability to modify the graphics and other objects. This is another important editing capability that should be added to LibreOffice for Android. Expected features are the ability to resize graphics, in addition to modify various properties of it via a new toolbar.

Required Expertise

The candidate should be fluent in Java and related tools, including Gradle and Maven. Having Android development experience is a requirement. Also, it is required that the candidate has C/C++ knowledge, and has done works in interfacing Java and native code. The candidate needs to understand threading in Java and C++.

The candidate should be familiar with user interface design, and human interface guidelines, and should be able to work with the design team to provide an acceptable user interface from the user point of view.

iOS Version

Apple iOS is another popular mobile operating system, and there is a high demand for providing LibreOffice application on this platform.

Although there is no iOS LibreOffice Viewer/Editor from TDF at the moment, the underlying architecture and technologies are mostly similar to what is used in LibreOffice Android viewer. Therefore, an early beta is expected to be released in the short term. The goal is to bootstrap a minimal native iOS application that will be improved in the next steps.

There is an older “LibreOffice Impress Remote” application from TDF[20] that has not been updated since 2013. The plan is to consolidate that application into the new LibreOffice application for iOS.

1 developer hiring is proposed in this task.

These are some of the development milestones expected for the iOS application:

LibreOfficeKit Bootstrapping

Providing a basic iOS application that uses LibreOfficeKit to load a document, pass tiles to the application and displays it on the iOS. This is the minimum iOS application that will be the foundation for implementing other features.

iOS Viewer

The goal is to provide LibreOffice Viewer comparable to LibreOffice Viewer for Android. Although the architecture is almost similar, the implementation is system-specific as the application is planned to be a native one.

iOS Basic Editor

The next milestone is to provide basic editing capabilities. The basic editing capabilities require processing input and making a minimum editing toolbar.

These are some of the expected editing features:

- Input handling: keyboard and pointer
- Character customization
- Paragraph customization
- Inserting images and modifying properties

20 https://apps.apple.com/us/app/libreoffice-remote-for-impress/id806879890

- Inserting tables and modifying properties
- Exporting document to PDF and other formats
- Undo/redo

Improving the editing capabilities to provide More complex features are handled at later stages.

Required Expertise

It is expected that the developer is fluent in both Swift/Objective C and C/C++, to be able to do the development in both backend and the UI of the application. The candidate should know build tools, and have done iOS development experience. The candidate needs to understand threading.

The candidate should be familiar with user interface design, and human interface guidelines, and should be able to work with the design team to provide an acceptable user interface from the user point of view.

Conclusion

The goal is to hire 4 in house developers to cover LibreOffice web and mobile support. The Executive Director, or other staff member appointed by the ED, shall direct management for the developers to ensure they effectively focus on their development areas, will follow the progress and address eventual issues.

The development work of the in-house developers will include fixing bugs and implementing features in full, fixing bugs that are blockers for community contributors and to help evaluating which complex tasks should be tackled by external specialists.

Regular weekly exchanges between the team and ESC will help the team to independently define the tasks to be performed so that the most relevant topics will be prioritized and overlaps will be avoided.

While the main areas of relevance are listed in this document, there may be situations where the in-house developers will work with the team to help fixing urgent issues, such as implementing security patches or app stores required changes, if skill sets and experience allow.

The in-house developers will have the same rules, rights and conditions as any other volunteer or corporate contributors. Overlaps or prioritization that find no clear conclusion within team will be evaluated by our Executive Director with the support of our mentors.

In-house developers will follow agile development methodologies, including short-term goals and delivery times.

In case the developers need to spend time mentoring community developers, the expectations may be adjusted by the Executive Director with the assistance of our current mentors.

In-house developers should be able to provide performance levels comparable to other developers working in the same area after an on-boarding and mentoring period of approximately 6 months.

In-house developers will greatly benefit from the experience accumulated by the team while also providing our current mentors a way to improve their mentoring techniques within the team to better serve other developers. The new members of the team may also provide mentoring services, once deemed ready by our current senior mentors, which will be scheduled in a way that will be compatible with their primary role as in-house developers.

In-house developers should be able to present their activities, discuss their achievements and possible obstacles on a regular basis in team and/or ESC meetings depending on the relevance of the topics to be discussed.

It is expected from the developers to set together with the team the planning of their activities and to participate to weekly team meetings. TDF’s staff will evaluate and report to our ED any attempt made by community members and/or third parties to propose tasks for our in-house developers that are distracting them from the work assigned to them by the ED. If ED suspects the request is driven by a personal or commercial interest, the ED will consult the Board to decide about eventual further actions.