GitLab

Final Report

Contributing to an Open-Source Project
Practical Course WiSe 2020/21

Simon Stieger
simon.stieger@tum.de

Technical University Munich

March 16, 2021

Contents

Terminology

GitLab is a large application consisting of multiple components, many of them managed in their own dedicated Git repository[1].
In this report, "GitLab" and "GitLab project" always refer to the whole project, not just the "GitLab" main repository[2] or any other subcomponent of GitLab.
"GitLab Inc." refers to the company maintaining GitLab. Its employees (together with the core team, see 5.3 Community Involvement) are referred to as "GitLab team members" or just "team members"[3], in contrast to community contributors outside the company, which are referred to as "community contributors" or "wider community contributors". These naming conventions follow those applied on GitLab's web pages hosted on about.gitlab.com.

1 The GitLab Project

GitLab is an open core DevOps lifecycle software.
As its name suggests, GitLab's key functionality revolves around the management of Git repositories and is extended by issue management, security checks, continuous integration and deployment, wiki, integrated software package, container and public pages hosting, and many other features. GitLab denotes itself as "The complete DevOps platform"[4] and since its inception in 2011[5], it has effectively grown from a small team collaboration tool to a large, feature-rich software providing extensive possibilities for software lifecycle management.
The GitLab project is currently led by GitLab Inc., a company founded by the initial creators of GitLab. With GitLab.com[6], GitLab Inc. provides a hosted public GitLab instance based on a freemium model, with many features available free of charge, such as unlimited private repositories and contributors, but also Enterprise Edition features for public open source projects[7].

The following chapters describe my experiences made while contributing to GitLab over the past months. For more in-depth information on GitLab's free and open source software aspects as well as its maintaining, financing and project structure, please refer to the preliminary report.

2 Brief Retrospect

To briefly summarize my experiences from the past couple of months: GitLab has almost fully met the expectations I had built up after my initial deep dive into the project in the course of my research for the preliminary report.
I was particularly positively surprised by how the team members reacted to community contributions and how motivated and engaged they interacted with the corresponding authors. My impressions may be partly shaped by the less pleasant experiences made by other participants in this course with different open source projects, but I am convinced that the GitLab team members handle community contributions with a very

commendable attitude, and that the company is also managed with these values in mind (see 5 Community Interaction and Management).
In fact, all of my contributions and changes have been characterized by a very friendly and respectful communication from the team member's side. In particular, I always had the feeling of being shown a deep gratitude even for smaller changes and improvements. Critical feedback was always constructive. I never had the feeling of things related to me or my contributions being discussed behind my back and I never experienced team members closing community proposals without a comment at all. For a more detailed discussion of my experiences, please see 3 Contributions and Communication.
I also found the contribution workflow in general to be mostly enjoyable – some problems I had and smaller shortcomings I had noticed were even improved independently of me during the last months, which demonstrates that the GitLab team members working on community management are constantly trying to improve the contribution process and experience (see 6 Contribution Process Ideas).

3 Contributions and Communication

This chapter discusses my initial difficulties with setting up a well-functioning development environment for my contributions (and how I mitigated them), and gives a summary of the things I have worked on and my interactions with the team members, accompanied by some retrospective remarks.

3.1 Development Environment Odyssey

Getting started with contributing code to GitLab (which is only one of many possibilities to contribute, see my preliminary report) poses some hurdles, as it requires a working development environment in most cases (except for contributions exclusively containing documentation or code documentation changes). Due to GitLab consisting of multiple different components[1] with even more large external dependencies, it is definitely not reasonable to pursue the path of an all-manual local environment configuration. The GitLab team is obviously aware of this and created a fully-fledged development environment called GitLab Development Kit (GDK). On code-contribution related documentation pages, the GDK is recommended for an optimal and streamlined development experience for both GitLab's team members as well as its community contributors[8].
Therefore, with the beginning of the project phase, I started setting up the GDK, which – as somewhat in the nature of a development kit – should be a pretty straightforward thing to do. However, as an (enthusiastic) Arch Linux user, I had a hard time getting the GDK to work in a reliable way, because Arch was not among the distributions natively supported by the GDK.
For this reason I then decided to install Ubuntu, which is officially supported, along my Arch Linux installation in a dual-boot setup. Installing the GDK there turned out exceptionally easy and allowed me to start a local GitLab instance for the first time, even though the page loading times were very high. Unfortunately, I soon ran into other problems: various GDK components, such as the webpack webserver, crashed

again and again. I found out that this was mainly due to the RAM size on my system being too low (8 GB) and leading to the kernel randomly killing processes. I proceeded creating a 10 GB swapfile, which solved the out-of-memory problem, but significantly increased the already high page loading times while also making my system as a whole very unresponsive and almost unusable, especially with an IDE running simultaneously.
Among other things, this led me to ordering a new notebook, but the manufacturer's enormous delivery delays due to the COVID-19 pandemic thwarted my plans and decisively lowered my hopes of having a usable development environment soon.
My temporary rescue was to set up a virtual Ubuntu installation with lxc on a rented server and developing there remotely. This of course implied some delays, but was overall better than anything before.
After a long five weeks, my notebook finally arrived and from then on everything worked perfectly, especially when I discovered the distribution-agnostic GCK[9] (GitLab Compose Kit, a Docker Compose based development environment) in February, which even allowed me to work on Arch Linux.

Current State

Since the beginning of this course's practical phase, there have been two decisive changes for the better to the contribution possibilities with regard to the development environment.

GitPod GitPod[10] is a fully web-based development environment which is now supported by GitLab and it was enabled for community contributions in January 2020. A GitPod-based development environment can be started with a single click directly from GitLab.com for branches on a contributor's fork. This sets up the full GDK on a virtual AWS server and gives you a unique URL where it is possible to develop changes and test them live as well as to commit and push them. The cloud IDE is based on Eclipse Theia[11], whereby that in turn is based on the Monaco Editor[12] from VSCode. I found GitPods performance to be relatively solid, although of course it can not keep up with a local GDK on proper hardware. Nevertheless, I think GitPod makes it much easier for potential new contributors to get started.

GDK for Arch(-based) Linux If I had started just about one month later, I probably would have been spared one of the problems right from the start: getting the GDK set up on Arch Linux. At the end of January, GitLab team member Marcin Sedlak-Jakubowski introduced support for Arch Linux and popular Arch-based distributions like Manjaro to the GDK[13].
Unfortunately I didn't notice that change right away, but only after it had been added to the list of supported operating systems and distributions in the project's README file on March 3rd[14]. As discussed in the theory part of this course, the README file is of central importance in open source software projects, as it usually represents the first point of contact with the project for new users or contributors. Especially if a README contains a list of supported operating systems, users probably expect it to

be updated continuously with relevant changes. However, I don't think it's extremely severe that the documentation on the README has been a month in the coming in this case.

3.2 Communication Channels

The large majority of my communication with team members as well as other community contributors happened through public channels, I have never been contacted personally by a team member (and there would not even have been an option to do so, because I – just like most GitLab team members – did not provide my e-mail address on my public profile). Discussions on the topic of code took place exclusively on GitLab (GitLab.com) itself at the respective merge request pages.
I also used Gitter[15], especially the public channel "gitlab/contributors"[16] was a good point of reference to get informed about the latest developments concerning the open source community (such as news about the GitLab Hackathon) and to get in touch with team members. On Gitter, I also had a private development-related conversation with Rajendra Kadam, a member of the GitLab core team (see 5.3 Community Involvement) and a conversation about contribution process improvement ideas with GitLab's Code Contributor Program Manager[17].

3.3 Contribution Summary and Reflections

Include information in API endpoint A certain API endpoint did not include a field required by GitLab's frontend app[18]. I solved it in a merge request[19] and had a brief follow-up discussion with the backend engineer reviewing it, where he suggested to add a unit test for the changed file (there had not been a test for this at all before). After implementing that and some other minor change requests, the changes got accepted and merged.

For my second version of this merge request, I missed the first possible milestone release by few days, because I was working on other things in the meantime. Stephan Kulla[20], one of the developers of the Serlo project[21], talked about how he tries to get and keep open source contributors involved in a private talk held as part of this course. In particular, he said it is required to have a strong frustration resistance level if you want to get community contributors involved with your open source project. I think GitLab members may sometimes feel similarly when they see abandoned issues or merge requests and that we, as community contributors, also somewhat owe it to them to at least not disappear without comment (especially because of how they interact with community contributors).
In the case described before, I responded to his message with an emoji, but in retrospect I would rather write a clearer message, saying that it might take a little while (even though nobody really regarded the underlying issue as critical) or simply get back to it sooner instead.

On January 6th and 7th, the GitLab Hackathon took place[22]. The GitLab team members held a virtual kickoff meeting on Zoom, where three community contribu-

tors (me included) participated. There was a very friendly atmosphere, everyone was greeted personally and had the chance to talk to the team members.
Unfortunately I also had other commitments on these two days, but I created the following two merge requests in the frame of the hackathon.

Documentation nitpicking I submitted a small documentation-facing merge request just fixing a minor grammar mistake[23], because I could not resist submitting a merge request which would definitely secure me one of the hackathon prizes.

Dropdown closing problem Later, I submitted another merge request tackling a UX (user experience) problem[24] with some dropdowns as pointed out in an older issue[25]. Here again, there was a follow-up discussion after my first version, which eventually ended in a completely new implementation being merged. The review of my initial implementation marked the only time I felt that there wasn't a lot of effort put into reviewing, because the answers to some of the questions I was asked would, in my opinion, have been obvious right away after looking at the source code a little more closely. Of course I was nevertheless happy to provide more detailed information about my implementation, and after that (and an emoji-based feedback I received) my changes got merged soon.
A few weeks later there was a follow-up discussion on the original issue, after which I created another merge request[26], which also got merged.

Comment fullscreen mode Another issue I had contact with was a frontend bug noticed by GitLab team member Paul Slaughter on the latest development version ("master" branch)[27]: the full screen mode for code comments and discussions did not work correctly and broke the whole page layout instead. I notified the author that the stable version was also affected. The priority was lowered later (because they didn't regard this as a heavily used feature), so I took care of the matter and implemented a fix in a merge request[28], which got accepted and merged very soon thereafter.

Custom emoji database changes and migrations After the hackathon, I had a conversation with Rajendra Kadam on Gitter. I had noticed how many community contributions he had submitted in 2019 and asked him for a recommendation for a deeper dive into the backend in the context of the development of the custom emoji feature for GitLab (where he had been particularly involved before). He suggested to pick up where he had left off and assigned me one of his latest issues[29] in our chat, describing a model and database change request.
I spent a lot of time on getting familiar with the backend structure and especially database migrations and the GraphQL API. After that, I also wrote a fairly large amount of tested code. However, when I checked back on the issue for a minor detail, I found out that its original author had already implemented the exact same feature.

Looking back on it now, I think I should definitely have documented publicly that I was working on this issue. It would have been possible to foresee the possibility of misunderstandings arising, since the issue was never formally assigned to Rajendra

Kadam. However, a comment from the issue author suggested this, which is why I think that he could also have asked Rajendra about the current status before starting to work on it.
Moreover, I think I should have asked for code pointers or browsed the code itself more thoroughly instead of first reading through a lot of architecture, backend code and feature flags documentation.

"npm" spelling changes I submitted another smaller merge request[30] introducing spelling changes to multiple backend files, as suggested in a corresponding issue[31] by a team member I had got to know within the context of the hackathon in January. The changes here affected a large number of files, which is why I was asked to create a more verbose commit message as well as stating that multiple reviewers are required here.
Moreover, I had originally added changes to a machine-generated translation file (which I was not aware of). I was very kindly informed about that and of course followed up with a new version.

This merge request particularly also demonstrates how grateful team members react to community contributions.

Feature proposal I created an issue to propose a change in order to be able to reach created merge requests more efficiently[32]. My proposal was politely declined, however, the initial reviewer consulted a second opinion from another team member.

I think that with such explanations community contributors can live quite well with a rejection, especially because the comment of the second reviewer also suggested a very efficient alternative to achieve the same thing (which I simply wasn't aware of before).

Backend rendering bug(s) My following merge request[33] introduced a fix for a regression bug in the server side rendering code of the project dashboard page[34]. During my research into the cause of it, I noticed some more similar problems, which I fixed in separate merge requests [35][36] in order to follow the iteration principle (see 4.1 Iteration) as closely as possible (as also discussed on the initial merge request with Coung Ngo[37]).

Here, one of the very few disadvantages of consistently pursuing an iterative approach becomes clear in my opinion: for smaller merge requests it introduces quite a lot of overhead, because you have to create new branches, push them and submit new merge requests. Especially as a community contributor, you have some slight disadvantages in that regard (see 6 Contribution Process Ideas). In the two follow-up merge requests, I directly pinged the reviewers of the first merge request, which allowed them to be merged very fast too.

Version dropdown rendering bug During the above mentioned contributions, I found a graphical bug on the merge request changes view and created a corresponding issue[38]. The issue was later closed as a duplicate of another issue[39]. I had unfortunately not been able to find using the search functionality before. However, I also implemented a fix for this bug[40].

More frontend bug reports Recently, I noticed two more frontend bugs regarding the search filter functionality for issues and merge requests, and I created corresponding bug reports[41][42].

Participation in discussions Occasionally I gave feedback on contributions that were particularly valuable to me[13], furthermore I also started a discussion with the aim of increasing the visibility of the GDK on public contribution documentation pages[43][44] (see 3.1 Development Environment Odyssey).

Dark mode settings UX improvements and discussion

In the following paragraphs, I will describe one of my code contributions and the corresponding discussions in a slightly more detailed manner.

Since I am a strong supporter of dark modes in graphical software and an early adopter of GitLab's dark mode, I noticed a bug with the implementation on the configuration page already some months ago on my private GitLab instance. As I found out, a corresponding issue[45] already existed on GitLab.com and I chose to to work on it and notified the issue author about it[46].
As soon as I could manage to find a fix for the underlying problem, I created a merge request[47].
My first implementation needed to be slightly changed, as Simon Knox pointed out in his initial review[48]. After I had implemented these, I contacted him again with some user experience improvement suggestions I had developed in the course of dealing with this issue[49]. He was very open to my suggestion and I implemented it accordingly, together with extensive unit tests.
Simon Knox and another reviewer showed great appreciation (they had also been bothered by the problem for a long time) and also pointed out to me that for some of my remarks corresponding issues already exist (which I will address next).

I really felt on the same wavelength with the team members involved in this merge request, as we all shared the same views. This in my opinion creates a strong sense of belonging, an aspect certainly important for being motivated to contribute to an open source project in the long term.

Apart from the above contributions, I also provided some technical support and help on Gitter, especially with setting up the GDK.
I have also contributed a bit to GitLab's translations on CrowdIn[50], mainly to see how it works, because I was interested in that for another project I will be working on in the future.

3.4 Response Times

GitLab puts a lot of effort into ensuring that community contributors receive timely responses. Unassigned merge requests are automatically included in a triage report issue and assigned a team member for further triaging[51], which worked out very well for my issues and merge requests.
Moreover, when submitting a merge request as a community contributor, the GitLab bot immediately creates a public comment describing ways to get feedback even faster, for example by mentioning the merge request coaches[52] in a comment.

After submitting a merge request, I got a first comment from a team member after 62 hours on average. Leaving out one merge request where it took 14 days (which could certainly have been shortened by pinging the merge request coaches), we achieve 32.6 hours on average and I think that this is quite impressive (even though one should keep in mind that I pinged team members right after submitting the merge request in some cases).
Moreover, I was also particularly surprised by the pace at which GitLab's team members reacted to my comments mentioning them. At times, using GitLab.com almost felt like using an instant messenger.
This is on the one hand certainly due to the fact that the GitLab team members are paid employees of a very large company and that reviewing contributions is part of their job at GitLab Inc. But even with that in mind, the speed was extraordinarily high and it was also noticeable that many team members were really available at almost any time of day and night (in their respective timezones) for handling community contributions (see [47] for an example).

4 Lessons learned

During the project phase I have been able to get much more involved with GitLab's contribution workflow and to get in-depth insights about its project management and development philosophy, which I consider very valuable.

4.1 Iteration

GitLab is a very large software project and especially in such projects, efficiency and continuous feedback are enormously important. What I have really come to appreciate in that regard is one of GitLab's six core values: "Iteration"[53]. Even when reading the handbook page on GitLab's values[54], it can be difficult to really understand the essence of some of the values, but the imperative behind the "Iteration" value can be described very easily: always think in small changes, contribute nothing but small changes at a time and break down complex features into small building blocks. An important concept I have come to know here is that of the "MVP" (minimum viable product), which describes the absolute minimum deliverable state of a feature and what developers should aim for at the beginning when introducing a new feature.
It also needs to be emphasized that the iteration principle applies to any kind of

change. It might at first seem excessive to, for example, break down a change of less than 30 lines of a documentation page into smaller parts[55]. On closer inspection however, it becomes clear that this has multiple advantages: it makes reviewing much easier (especially with GitLab's complex reviewing policies) and allows for much better traceability, which significantly reduces the need for redundant thinking in the long run ("Why was that done like this?") and makes it easy to gain insights from the changes history[28]. Obviously, this applies to both code and technical writing.

4.2 Clear Communication

A contributor should never assume that something is known to all involved parties if there has not been any clear and explicit communication about it. Especially after my experiences described in 3.3 Contribution Summary and Reflections with the issue related to the custom emoji feature, I will certainly try hard to avoid similar misunderstandings in the future. I think that especially when working remotely, people are even more prone to not communicating things clearly to others.
Key takeaway: if there is even just a minimal risk of a misunderstanding, nip it in the bud by documenting everything publicly and mentioning involved collaborators.

4.3 Realistic planning and limited Scope

Unfortunately, I could not finish everything I had started to work on (but I will continue to work on it in the future). This is on the one hand due to the fact that I worked on too many issues at the same time at the end, but also because I underestimated how many different technologies are used in the GitLab project.
In retrospect, I think I should have tried to focus on either the frontend or the backend in order to be able to create some more valuable contributions. In particular, I think I should have sticked to working on backend issues after spending quite a bit of time there within the context of the database-facing custom emoji issue.
Nevertheless, I am very happy to have gotten to know the frontend framework Vue.js from close by in the course of my contributions.

5 Community Interaction and Management

This chapter discusses how the GitLab team members manage the open source community and how they try to ensure to adhere to open source and free software development best practices wherever possible.

5.1 Open Source Community Advocacy

Although GitLab Inc. as a company of course also operates for profit, they have never abandoned or lost their free and open source software roots (which in my opinion would also be a great disadvantage). I have noticed multiple times that GitLab's good attitude towards the open source community is very strongly promoted structurally, which numerous public pages demonstrate[56][52][57].

A powerful example is the handbook page dedicated to GitLab Inc.'s biggest risks[57]: one of them is titled "Loss of the open source community" and also describes how to mitigate this risk. The page has originally been authored by GitLab co-founder and CEO Sid Sijbrandij. This clearly indicates that GitLab Inc. is not a company merely half-heartedly trying out open source, and that they (up to the management board) consider its open source aspect and the related community to be an important part playing an important role also for GitLab's future.
Moreover, the handbook provides very concrete collaboration guidelines for team members involved with community contributions, but also the expected attitude of team members towards community contributors[52]:

> *With each merge request opened by a wider community member, it's important to note GitLab is not their main focus. They have contributed to GitLab out of kindness, and we should aim to give them the space they need to fulfill their merge request.*

This is exactly how I experienced their interactions with me and other contributors (see 3 Contributions and Communication).
I also find it particularly noteworthy that the team members regularly organize events for the community, such as the quarterly GitLab Hackathon[58], and that they have also launched the Heroes program[59] to reward community contributions.
With regard to the GitLab Hackathon, there was only one drop of bitterness: the successful participants of the Q4'2020 edition[22] (which took place on January 6th and January 7th, 2021) have not been contacted and the announced prizes have not been shipped to date, more than two months after the hackathon taking place.

5.2 Communication and Transparency

Most communication of GitLab team members, especially with community contributors, takes place publicly on GitLab (GitLab.com) itself. I consider this ideal for reasons of transparency and traceability, which I regard as highly important for very large projects with a large number of contributors in different sections. The GitLab project does not have a central point of authority, as we often have with smaller open source projects, therefore transparent communication certainly plays an even greater role. In fact, "Transparency" is also one of GitLab's values, the corresponding handbook page states that everything should be public by default[60], with exceptions being explicitely listed[61].
It is noticeable that in some places internal communication threads are referenced, these are mostly links to the CRM platform Salesforce and the customer service platform Zendesk[62][63][64]. However, this is mainly due to reasons of confidentiality, as it usually concerns feature requests of larger customers, and I am not aware of concrete early feature discussions done off GitLab.com. I therefore think that community contributors are not excluded from these, and in fact I have often seen non-team members participating in feature discussions on the corresponding issues or epics[64].
Apart from GitLab.com, team members also use (private) Slack channels for communication. These are also sometimes referenced on public issues[65], which of course

makes it difficult to follow along for community contributors. However, I never had the feeling that essential parts of a discussion were withheld from the public by taking place on a referenced Slack channel.
I found the language to be very inclusive on all public pages I visited, I never observed any kind of harassment.
Public communication is also very clear in general. Team members do use some abbreviations relatively frequently, especially in the context of reviewing and approving merge requests (for example "LGTM" for "looks good to me" or "MWPS" for "merge when pipeline succeeds"), but they become clear rather quickly.

5.3 Community Involvement

Because almost all communication is public (see 5.2 Communication and Transparency), everyone is invited to take part in discussions. Especially in discussions about new features there are often feedback posts and suggestions from community members[64].
Another option to get in touch with members is by contacting members of the GitLab core team[66], for example in case of questions regarding the GitLab project and its management in general. The core team has been established as an interface between the wider community and the team members (see the preliminary report for further details). Its members regularly take part in scheduled meetings with team members. Moreover, they have access to some private GitLab Slack channels, which is why they have to sign a non-disclosure agreement when joining the core team.
Currently, the core team consists of 14 members[67]. However, 6 of them (almost 43%) are team members employed of GitLab Inc. even though this should only be the case for up to two members[66]. It is currently discussed if this is still acceptable, because a core team member not employed by GitLab Inc. opened a corresponding issue[68]. I think that it does not necessarily have a decisive disadvantage to have multiple company team members in the core team too, but I think it somewhat distorts the statistics on the number of core team members.

6 Contribution Process Ideas

As described in 2 Brief Retrospect and 3 Contributions and Communication, my experiences contributing to GitLab were generally very positive, but some things still caught my eye as a community contributor and I think there is some room for improvement here.

6.1 Issue and Merge Request Triaging

Issue and Merge Request Labelling

Especially for more experienced community contributors, I would consider it very helpful if they could get the permissions to directly set labels themselves, both for issues and merge requests. GitLab makes heavy use of labels to associate issues and merge requests to its different sections. Allowing contributors to choose labels could often save a full triage level because it makes it possible to automatically assign the issue or a merge request to a team member of the correct section. I think that this could save the team members a lot of time without having any considerable disadvantages, especially if it only applies to experienced contributors trusted by the GitLab team members.

In the course of the practical phase of this course, there has already been an improvement to this aspect: for merge requests, it is now possible to assign so-called "group labels"[69] writing a comment mentioning the GitLab bot ("@gitlab-bot") and specifying the desired label, which works very well and fast[70]. The described functionality makes it possible to directly bring a merge request to the attention of the correct team, which of course helps in finding a reviewer faster. For other labels (for example "frontend" or "bug") however, this feature is not enabled.

Merge Request Reviewer Assignment

After multiple successful merge requests, I think that community contributors eventually get a sense of who is best suited to review their next merge request. Besides that, you often already deal with the history of the changed files in advance of the submission anyway and therefore know who has worked on them in the past. For these reasons it would be desirable to also be able to assign reviewers directly.
Moreover, the status of team members is always public (e.g. "OOO" with an accompanying date range on days off or holidays). I am therefore not aware of any disadvantages community contributors may have due to a lack of internal company knowledge about the current situation of the team members.
Here again, it would be conceivable to give the required permissions only to more experienced contributors.

For the inclined reader: as long as there is no such feature, it is of course also possible to ping team members directly by mentioning them ("@username") for example in a comment, which is what I did multiple times to have some merge requests merged much faster[35].

6.2 Merge Request Template

GitLab's merge request template is rather complex and includes a checklist with multiple points[47], most of them linked to a long documentation page. While this is commendable from a quality assurance point of view, I think it can be more of a discouragement for new contributors. It could be beneficial to include a notice as a comment inside the markdown template, stating that it is not a hard requirement to verify that everything is one hundred percent fulfilled, since reviewers can always point these things out eventually.

6.3 Work on Team Member Issues

Stephan Kulla[20] talked about his way of finding and recruiting new open source contributors in the context of a private presentation held as part of this course's

theory part. One of his approaches remained particularly memorable to me: giving new contributors tasks that he would be working on anyway, and offering pair programming whenever needed.
With GitLab, it is usually the case that issues that team members plan to work on in the near future are not labelled with “Accepting merge requests” (and are therefore mostly ignored by community contributors). I think it would be great if there was, for example, a label called “offering assisted contribution” to indicate that a team member would be soon working on the issue anyway and that they would be happy to help an interested community contributor for example by pair programming with them. In my opinion, this or a similar approach could considerably lower the inhibition threshold of new potential contributors, especially at their first contact with GitLab’s enormously large codebase.
Of course these contributions should mainly concern features of the free version of GitLab, as it should not be about recruiting free collaborators.
Regarding pair programming in general, there seems to have been a company-internal initiative at GitLab by the frontend team[73][74], but so far nothing for community contributors. I therefore discussed my idea with GitLab’s current Code Contributor Program Manager, and he informed me that this is currently being evaluated internally.

6.4 Automated Changelog Creation

While it is really cool that team members are allowed to create a changelog entry for every merge request they submit[75], it is quite time-consuming to create them. This especially holds if you want to add the correct merge request ID, which often needs a new commit or a forced push after submitting the merge request and getting the corresponding ID, which also increases the number of versions of the merge request. In my opinion, it would be very beneficial if contributors could have GitLab generate the changelog entry based on the merge request title on merge.

Conclusion

I consider the experiences made contributing to GitLab to be very valuable. It was great to meet so many people all around the world.
The team members were very friendly, welcoming and helpful. It was really cool to get answers almost instantly at the most unusual hours, and to always know and see (except on parts of the weekend) that countless other people are working on the same project they’re all passionate about at the same time. I think that the international aspect of GitLab is something really special that they should definitely preserve (which I’m sure they will).
I am also quite certain that without this course I would probably never have had the idea to contribute to GitLab (and I would definitely have missed out on something). In any case, the contributions described in this report were certainly not my last ones.

References

[1] GitLab architecture overview - GitLab, https://docs.gitlab.com/ee/development/architecture.html. Last accessed on 16 Mar 2021.

[2] GitLab.org - GitLab - GitLab, https://gitlab.com/gitlab-org/gitlab. Last accessed on 16 Mar 2021.

[3] Meet the GitLab team - GitLab - GitLab, https://about.gitlab.com/company/team/. Last accessed on 16 Mar 2021.

[4] DevOps Platform Delivered as a Single Application - GitLab, https://about.gitlab.com/. Last accessed on 16 Mar 2020.

[5] init commit (9ba12248) · Commits · GitLab.org / GitLab FOSS · GitLab, https://gitlab.com/gitlab-org/gitlab-foss/commit/9ba1224867665844b117fa037e1465bb706b3685. Last accessed on 16 Mar 2020.

[6] GitLab.com Feature Comparison - GitLab, https://about.gitlab.com/pricing/gitlab-com/feature-comparison/. Last accessed on 16 Mar 2020.

[7] GitLab Ultimate and Gold now free for education and open source - GitLab, https://about.gitlab.com/blog/2018/06/05/gitlab-ultimate-and-gold-free-for-education-and-open-source/. Last accessed on 16 Mar 2020.

[8] Contributing to Development - GitLab, https://about.gitlab.com/community/contribute/development/. Last accessed on 16 Mar 2021.

[9] GitLab.org / gitlab-compose-kit · GitLab, https://gitlab.com/gitlab-org/gitlab-compose-kit. Last accessed on 16 Mar 2021.

[10] Gitpod - Dev environments for the cloud, https://www.gitpod.io/. Last accessed on 16 Mar 2021.

[11] Theia - Cloud and Desktop IDE Platform, https://theia-ide.org/. Last accessed on 16 Mar 2021.

[12] Monaco Editor, https://microsoft.github.io/monaco-editor/. Last accessed on 16 Mar 2021.

[13] Add Arch and Manjaro to supported OSes (!1782) - GitLab Development Kit, https://gitlab.com/gitlab-org/gitlab-development-kit/-/merge_requests/1782. Last accessed on 16 Mar 2021.

[14] Add Arch and Manjaro to supported OS list (!1855) - GitLab Development Kit, https://gitlab.com/gitlab-org/gitlab-development-kit/-/merge_requests/1855. Last accessed on 16 Mar 2021.

[15] Gitter, https://gitter.im/gitlab/home. Last accessed on 16 Mar 2021.

[16] gitlab/contributors - Gitter, https://gitter.im/gitlab/contributors. Last accessed on 16 Mar 2021.

[17] Code contributor program manager - GitLab, https://about.gitlab.com/job-families/marketing/code-contributor-program-manager/. Last accessed on 16 Mar 2021.

[18] Fix icon for submodules in tree list of merge requests (#285287) · Issues · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/issues/285287. Last accessed on 16 Mar 2021.

[19] Include submodule information for files in diff metadata (!50346) · Merge Requests · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/merge_requests/50346. Last accessed on 16 Mar 2021.

[20] Über mich - Stephan Kulla, http://kulla.me/de/. Last accessed on 16 Mar 2021.

[21] Serlo Education - GitHub, https://github.com/serlo. Last accessed on 16 Mar 2021.

[22] What happened at the Q4'2020 GitLab Hackathon - GitLab, https://about.gitlab.com/blog/2021/02/08/q42020-hackathon-recap/. Last accessed on 16 Mar 2021.

[23] Fix minor error in JavaScript style guide docs (!51167) · Merge Requests · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/merge_requests/51167. Last accessed on 16 Mar 2021.

[24] Don't close auto suggest select boxes if only mouseup outside box (!51139) · Merge Requests · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/merge_requests/51139. Last accessed on 16 Mar 2021.

[25] Auto-suggest select boxes collapse on mouseUp instead of mouseDown (#33748) · Issues · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/issues/33748. Last accessed on 16 Mar 2021.

[26] Don't close issue sidebar label select box on click if only mouseup outside (!56721) · Merge Requests · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/merge_requests/56721. Last accessed on 16 Mar 2021.

[27] "Go full screen" comment button does not work in MR diffs (#294156) · Issues · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/issues/294156. Last accessed on 16 Mar 2021.

[28] Fix full screen mode for comments in merge request changes view (!53009) · Merge Requests · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/merge_requests/53009. Last accessed on 16 Mar 2021.

[29] Add creator_id to custom_emoji table (#284641) · Issues · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/issues/284641. Last accessed on 16 Mar 2021.

[30] Spell "npm" with lowercase letters in Package Registry UI (!54163) · Merge Requests · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/merge_requests/54163. Last accessed on 16 Mar 2021.

[31] Make npm lowercase in the GitLab Package Registry UI (#321420) · Issues · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/issues/321420. Last accessed on 16 Mar 2021.

[32] Efficiently finding merge requests as the author (but not assignee) in the frontend (#321924) · Issues · GitLab.org / GitLab · GitLab https://gitlab.com/gitlab-org/gitlab/-/issues/321924. Last accessed on 16 Mar 2021.

[33] Hide issue count and link in project list where issues are disabled (!54275) · Merge Requests · GitLab.org / GitLab · GitLab https://gitlab.com/gitlab-org/gitlab/-/merge_requests/54275. Last accessed on 16 Mar 2021.

[34] Issue counts (and a link) are still showing up in the project list for a project with no issues (#321593) · Issues · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/issues/321593. Last accessed on 16 Mar 2021.

[35] Hide merge request count and link in project list where merge requests are disabled (!56432) · Merge Requests · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/merge_requests/56432. Last accessed on 16 Mar 2021.

[36] MR 56520, https://gitlab.com/gitlab-org/gitlab/-/merge_requests/56432. Last accessed on 16 Mar 2021.

[37] Hide issue count and link in project list where issues are disabled (!54275) · Merge Requests · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/merge_requests/54275#note_521139547. Last accessed on 16 Mar 2021.

[38] MR changes view: version select dropdowns behind tabs bar on smaller screens (landscape) (#321488) · Issues · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/issues/321488, Last accessed on 16 Mar 2021.

[39] MR - Changes - Mr version control dropdown zindex bug (#277201) · Issues · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/issues/277201, Last accessed on 16 Mar 2021.

[40] Render version dropdowns in MR changes view above tab navbar (!54159) · Merge Requests · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/merge_requests/54159, Last accessed on 16 Mar 2021.

[41] Filtering by releases on the dashboard's issue (and merge request) search is broken (#324700) · Issues · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/issues/324700, Last accessed on 16 Mar 2021.

[42] Filtering by releases on the dashboard's issue (and merge request) search is broken (#324700) · Issues · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/issues/324700, Last accessed on 16 Mar 2021.

[43] GCK and GDK should be able to co-exist (#19) · Issues · GitLab.org / gitlab-compose-kit · GitLab, https://gitlab.com/gitlab-org/gitlab-compose-kit/-/issues/19#note_527284573, Last accessed on 16 Mar 2021.

[44] Co-existence of GDK and GCK (#1081) · Issues · GitLab.org / GitLab Development Kit · GitLab, https://gitlab.com/gitlab-org/gitlab-development-kit/-/issues/1081#note_527528657, Last accessed on 16 Mar 2021.

[45] The top bar in the user settings preferences page doesn't render correctly in dark mode (#321929) · Issues · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/issues/321929. Last accessed on 16 Mar 2021.

[46] The top bar in the user settings preferences page doesn't render correctly in dark mode (#321929) · Issues · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/issues/321929#note_512035680. Last accessed on 16 Mar 2021.

[47] Correctly style Dark Mode application header on profile preferences page (!54575) · Merge Requests · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/merge_requests/54575. Last accessed on 16 Mar 2021.

[48] Correctly style Dark Mode application header on profile preferences page (!54575) · Merge Requests · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/merge_requests/54575#note_520721885. Last accessed on 16 Mar 2021.

[49] Correctly style Dark Mode application header on profile preferences page (!54575) · Merge Requests · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/merge_requests/54575#note_520722344. Last accessed on 16 Mar 2021.

[50] GitLab translations in Crowdin, https://translate.gitlab.com/. Last accessed on 16 Mar 2021.

[51] 2020-12-21 Untriaged community merge requests requiring initial triage (#1229) · Issues · GitLab.org / quality / triage-reports · GitLab, https://about.gitlab.com/handbook/engineering/workflow/iteration/. Last accessed on 16 Mar 2021.

[52] Merge Request Coach - GitLab, https://about.gitlab.com/job-families/expert/merge-request-coach/. Last accessed on 16 Mar 2021.

[53] Iteration - GitLab, https://about.gitlab.com/handbook/engineering/workflow/iteration/. Last accessed on 16 Mar 2021.

[54] GitLab Values - GitLab, https://about.gitlab.com/handbook/values/. Last accessed on 16 Mar 2021.

[55] Clarify changelog guidance documentation (!51172) · Merge Requests · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/merge_requests/51172. Last accessed on 16 Mar 2021.

[56] Get Help - Contribute to GitLab - GitLab, https://about.gitlab.com/community/contribute/#getting-help. Last accessed on 16 Mar 2021.

[57] Biggest risks - GitLab, https://about.gitlab.com/handbook/leadership/biggest-risks/#loss-of-the-open-source-community. Last accessed on 16 Mar 2021.

[58] Past GitLab Hackathon Events, https://about.gitlab.com/community/hackathon/past-events/. Last accessed on 16 Mar 2021.

[59] GitLab Heroes - GitLab, https://about.gitlab.com/community/heroes/. Last accessed on 16 Mar 2021.

[60] Transparency - GitLab Values - GitLab, https://about.gitlab.com/handbook/values/#transparency. Last accessed on 16 Mar 2021.

[61] Not Public - GitLab Communication - GitLab, https://about.gitlab.com/handbook/communication/#not-public. Last accessed on 16 Mar 2021.

[62] Support and Certification of GitLab on Amazon Linux 2 (&2195) · Epics · GitLab.org · GitLab, https://gitlab.com/groups/gitlab-org/-/epics/2195#note_244510913. Last accessed on 16 Mar 2021.

[63] Project Integration Management - Supporting mass-integration at a Group and Instance Level (&2137) · Epics · GitLab.org · GitLab, https://gitlab.com/groups/gitlab-org/-/epics/2137#note_295496020. Last accessed on 16 Mar 2021.

[64] 'not' and 'or' in search and filter bars, including saved config in boards (&291) · Epics · GitLab.org · GitLab, https://gitlab.com/groups/gitlab-org/-/epics/291#note_135738274. Last accessed on 16 Mar 2021.

[65] Document challenges with re-using project level clusters (#61226) · Issues · GitLab.org / GitLab FOSS · GitLab, https://gitlab.com/gitlab-org/gitlab-foss/-/issues/61226. Last accessed on 16 Mar 2021.

[66] GitLab Core Team - GitLab, https://about.gitlab.com/community/core-team/. Last accessed on 16 Mar 2021.

[67] Group members · community-members · GitLab, https://gitlab.com/groups/gitlab-core-team/community-members/-/group_members. Last accessed on 16 Mar 2021.

[68] Keep the number of Team members in Core Team up to 2 (#78) · Issues · GitLab Core Team / general · GitLab, https://gitlab.com/gitlab-core-team/general/-/issues/78. Last accessed on 16 Mar 2021.

[69] Group labels - Issues workflow - GitLab, https://docs.gitlab.com/ee/development/contributing/issue_workflow.html#group-labels. Last accessed on 16 Mar 2021.

[70] Bot group label assignment - Don't close issue sidebar label select box on click if only mouseup outside (!56721) · Merge Requests · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/merge_requests/56721#note_529869806. Last accessed on 16 Mar 2021.

[71] .gitlab/issue_templates - master - GitLab.org / GitLab, https://gitlab.com/gitlab-org/gitlab/-/tree/master/.gitlab/issue_templates. Last accessed on 16 Mar 2021.

[72] .gitlab/merge_request_templates - master - GitLab.org / GitLab, https://gitlab.com/gitlab-org/gitlab/-/tree/master/.gitlab/merge_request_templates. Last accessed on 16 Mar 2021.

[73] Agile for developers: pairing sessions - GitLab, https://about.gitlab.com/blog/2019/08/20/agile-pairing-sessions/. Last accessed on 16 Mar 2021.

[74] Format for Pair Programming (#12) - Issues - GitLab.org / Frontend / general - GitLab, https://gitlab.com/gitlab-org/frontend/general/-/issues/12. Last accessed on 16 Mar 2021.

[75] changelogs/unreleased/321929-fix-dark-mode-app-header-on-profile-preferences-page.yml · 7f3d3186297af21f01e69f4675162eb295f65fc9 · GitLab.org / GitLab · GitLab, https://gitlab.com/gitlab-org/gitlab/-/blob/7f3d3186297af21f01e69f4675162eb295f65fc9/changelogs/unreleased/321929-fix-dark-mode-app-header-on-profile-preferences-page.yml. Last accessed on 16 Mar 2021.