GitHub good practices

Table of Contents

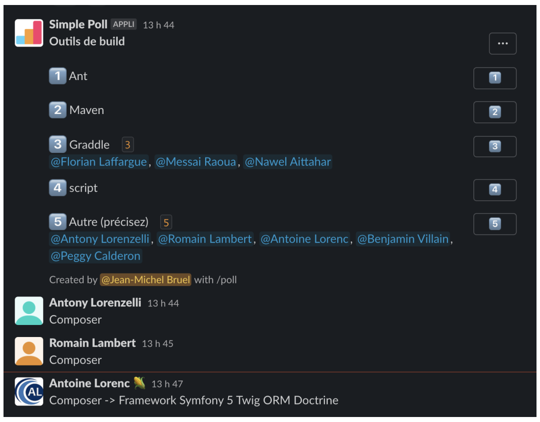
1. Correspondences Scrum/GitHub/Gitlab

Table 1. Correspondences between Artefacts

Scrum	GitHub	GitLab
User Story	Issues	Issues
Task	Task Lists/dependency issues	Task Lists
Epic	??	Epics
Points/Estimation	Weights ($)	Weights
Product Backlog	Issues Lists	Issues Lists
Priorities	Labels	Labels
Sprint	Milestone	Milestone
Burndown Chart	A nice app!	Burndown Chart
Agile board	Project board	Issue Board

2. Build

Let’s do a quick poll about build tools


Simple Poll APPLI 13 h 44
Outils de build

1 Ant
2 Maven
3 Graddle 3
@Florian Laffargue, @Messai Raoua, @Nawel Aittahar
4 script
5 Autre (précisez) 5
@Antony Lorenzelli, @Romain Lambert, @Antoine Lorenc, @Benjamin Villain, @Peggy Calderon

Created by @Jean-Michel Bruel with /poll

Antony Lorenzelli 13 h 44
Composer

Romain Lambert 13 h 45
Composer

Antoine Lorenc 13 h 47
Composer -> Framework Symfony 5 Twig ORM Doctrine

Figure 1. 2021 promotion build tool usage

2014 Study:

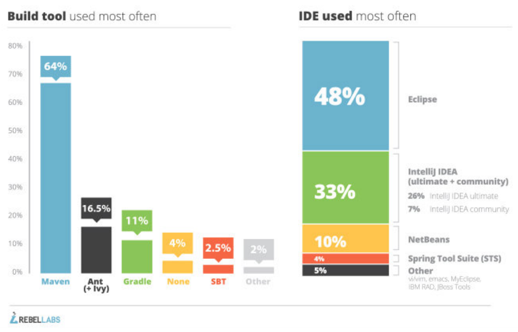


Figure 2. Ant vs Maven vs Gradle (source here)

Ant example (cf. source)

```
<project xmlns:ivy="antlib:org.apache.ivy.ant" name="java-build-tools" default="jar">

    <property name="src.dir" value="src"/>
    ...
    <path id="lib.path.id">
        <fileset dir="${lib.dir}" />
    </path>

    <target name="clean">
        <delete dir="${build.dir}"/>
    </target>

    <target name="compile">
        <mkdir dir="${classes.dir}"/>
        <javac srcdir="${src.dir}" destdir="${classes.dir}"
classpathref="lib.path.id"/>
    </target>

    <target name="jar" depends="compile">
        <mkdir dir="${jar.dir}"/>
        <jar destfile="${jar.dir}/${ant.project.name}.jar" basedir="${classes.dir}"/>
    </target>

</project>
```

Maven example (cf. source)

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/maven-v4_0_0.xsd">

    <modelVersion>4.0.0</modelVersion>
    <groupId>com.technologyconversations</groupId>
    <artifactId>java-build-tools</artifactId>
    <packaging>jar</packaging>
    <version>1.0</version>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.11</version>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>2.3.2</version>
            </plugin>
        </plugins>
    </build>

</project>
```

Gradle example (cf. source)

```
apply plugin: 'java'
apply plugin: 'checkstyle'
apply plugin: 'findbugs'
apply plugin: 'pmd'

version = '1.0'

repositories {
    mavenCentral()
}

dependencies {
    testCompile group: 'junit', name: 'junit', version: '4.11'
    testCompile group: 'org.hamcrest', name: 'hamcrest-all', version: '1.3'
}
```

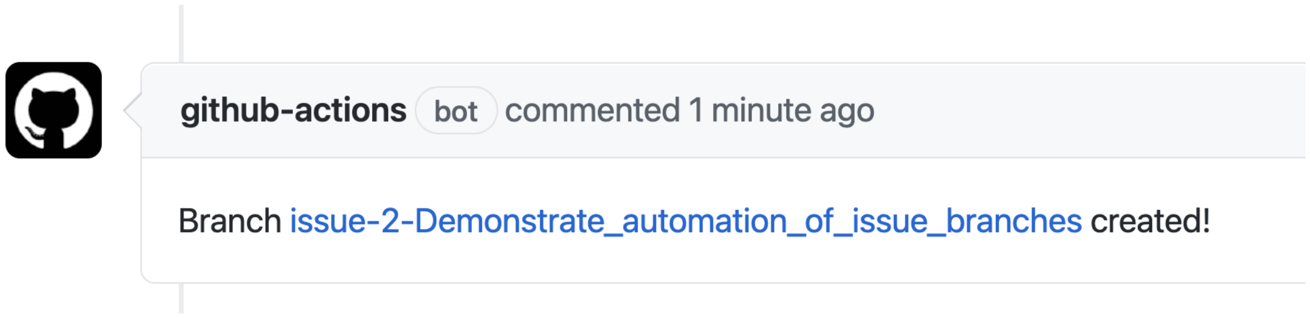
3. Automate issue branches

https://github.com/marketplace/actions/create-issue-branch

Add this to your workflow YAML configuration:

```
on:
    issues:
        types: [assigned]
    issue_comment:
        types: [created]
    pull_request:
        types: [closed]

jobs:
    create_issue_branch_job:
        runs-on: ubuntu-latest
        steps:
        - name: Create Issue Branch
          uses: robvanderleek/create-issue-branch@master
          env:
            GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```



github-actions bot commented 1 minute ago

Branch issue-2-Demonstrate_automation_of_issue_branches created!

Figure 3. As soon as the issue is assigned...

4. Use tags

```
git tag 1.1.0 -m "Release 1.1.0"
git push origin tag 1.1.0
```

Tagging practices:

- You don’t tag branches. You tag commits!
- You should add a tag to mark a released version. If you then need to make bug fixes to that release, you would create a branch at the tag
- If you checkout a tag, you will need to create a branch to start working from it

More here.

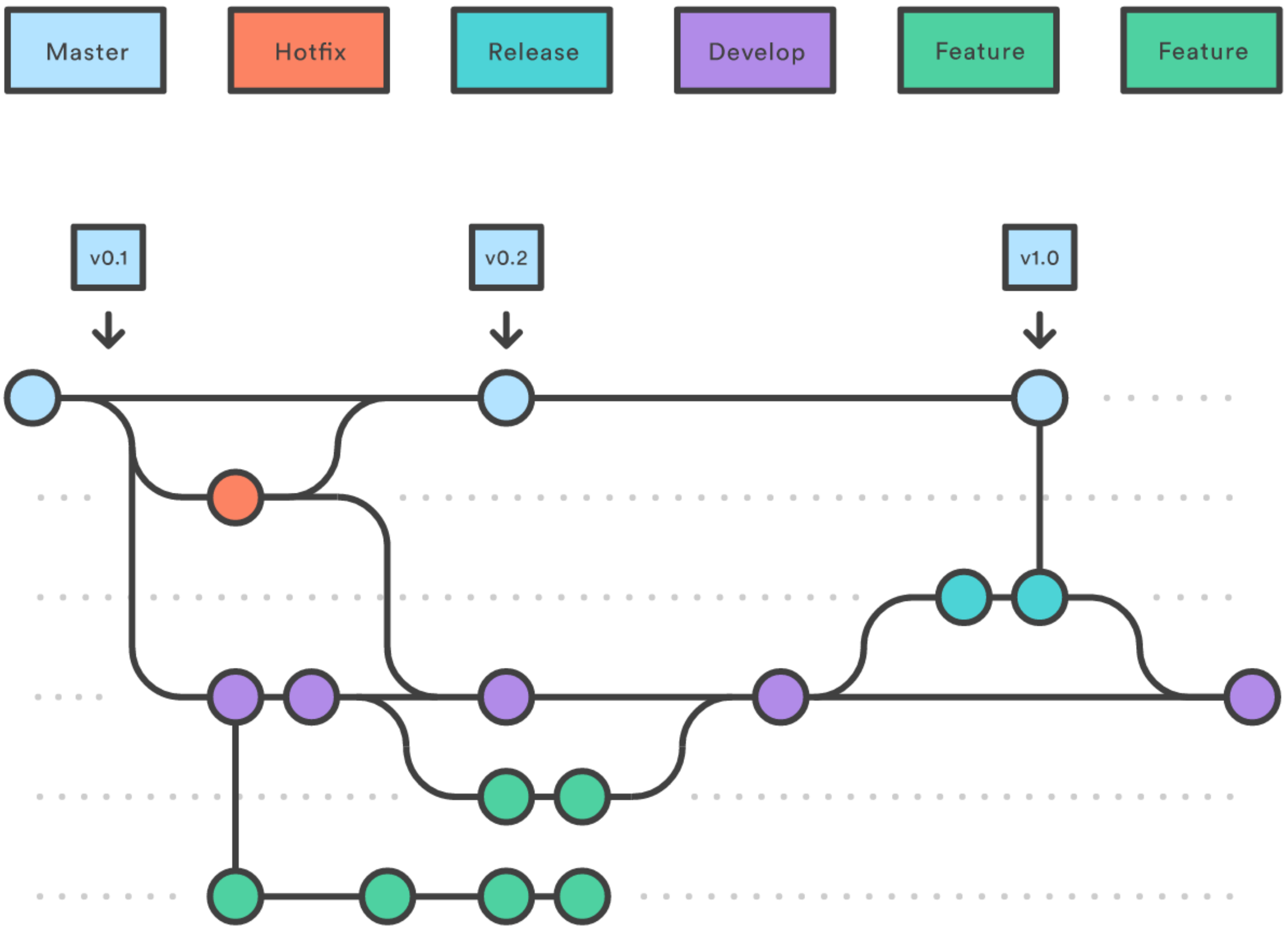


Figure 4. Tags in git flow (source here)

5. Meaningful `.gitignore` file

This repo `.gitignore` file

```
# Output directory for HTML files
output/
Gemfile.lock
*.html
.DS_Store


topics
node_modules
package-lock.json

# Ignore Gradle project-specific cache directory
.gradle

# Ignore Gradle build output directory
build
.project
.settings
settings.json
```

https://gitignore.io

Example for `Node.js` *(only beginning!)*

```
# Created by https://www.toptal.com/developers/gitignore/api/node
# Edit at https://www.toptal.com/developers/gitignore?templates=node

### Node ###
# Logs
logs
*.log
npm-debug.log*
yarn-debug.log*
yarn-error.log*
lerna-debug.log*
```

6. Meaningful commit messages

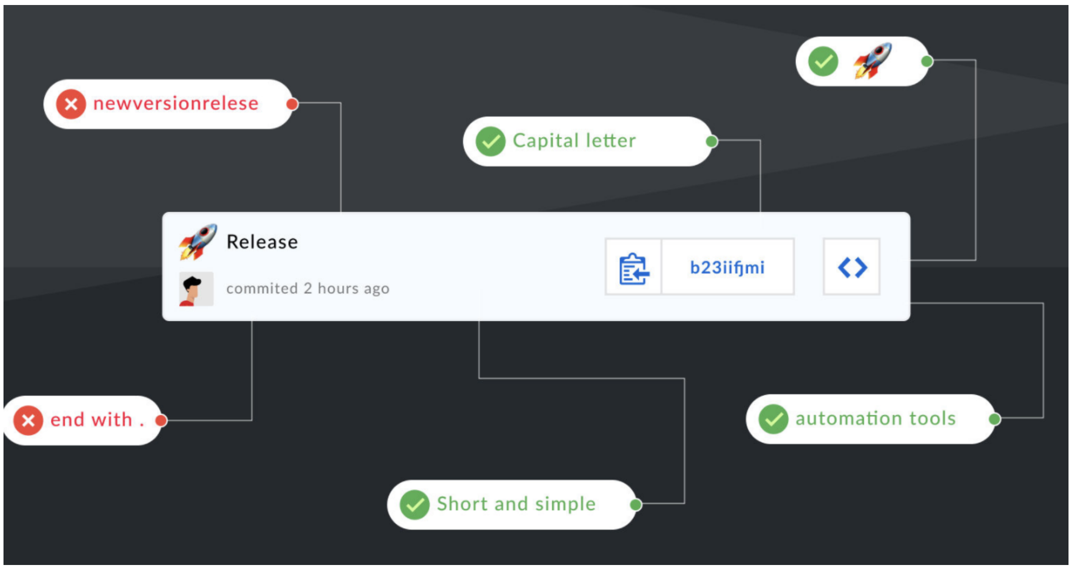


Figure 5. Example of conventions (source here)

Example of `~/.gitconfig` file

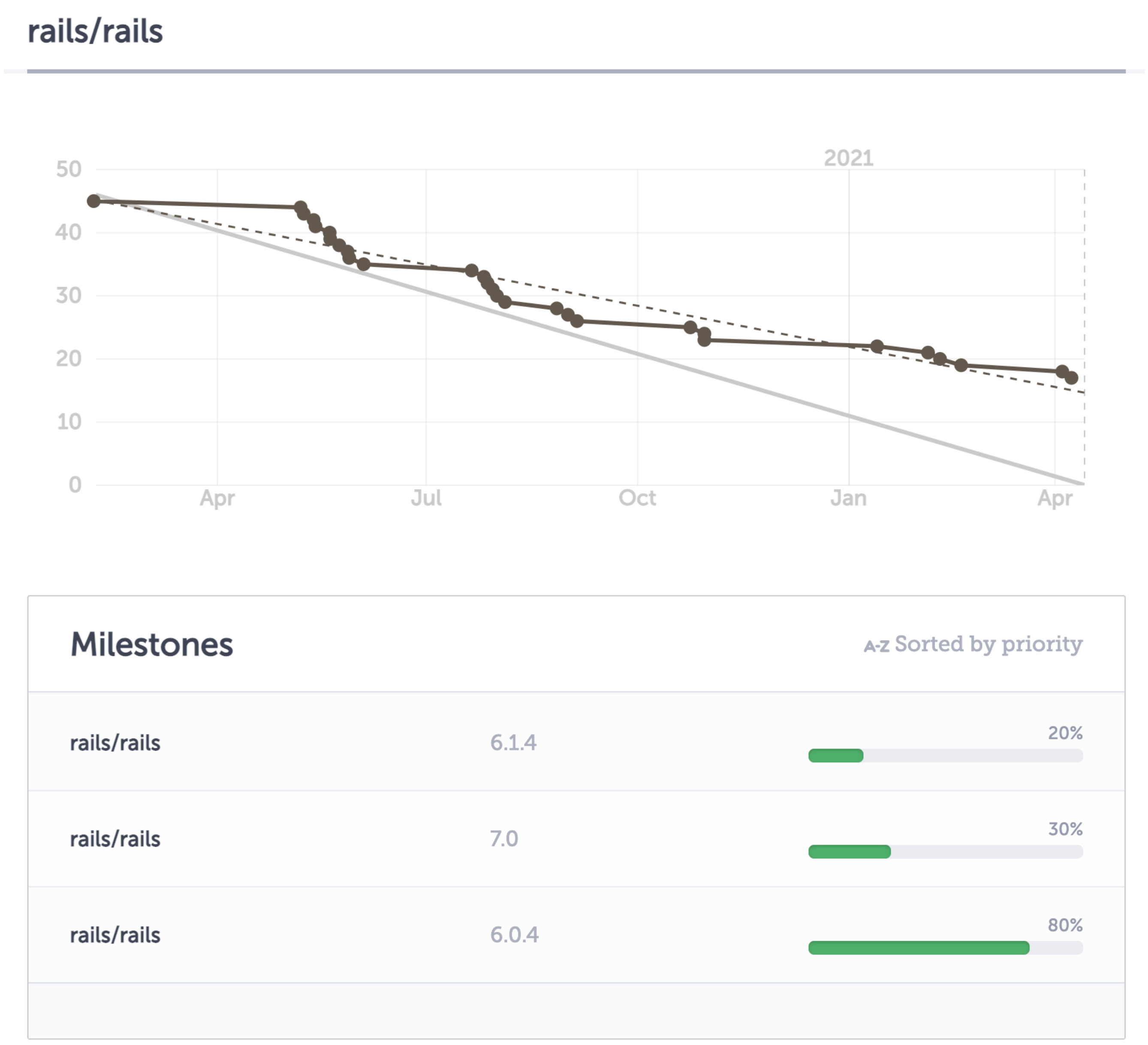
```
# Git Commit, Add all and Push — in one step.
cap = "!f() { git add .; git commit -m \"$@\"; git push; }; f"

# NEW.
new = "!f() { git cap \"▯ NEW: $@\"; }; f"
# IMPROVE.
imp = "!f() { git cap \"▯ IMPROVE: $@\"; }; f"
# FIX.
fix = "!f() { git cap \"▯ FIX: $@\"; }; f"
# RELEASE.
rlz = "!f() { git cap \"▯ RELEASE: $@\"; }; f"
# DOC.
doc = "!f() { git cap \"▯ DOC: $@\"; }; f"
# TEST.
tst = "!f() { git cap \"▯ TEST: $@\"; }; f"
```

6.1. Burndown charts

A nice add-on app that takes a repo and draws the current milestone (not closed) burndown chart :

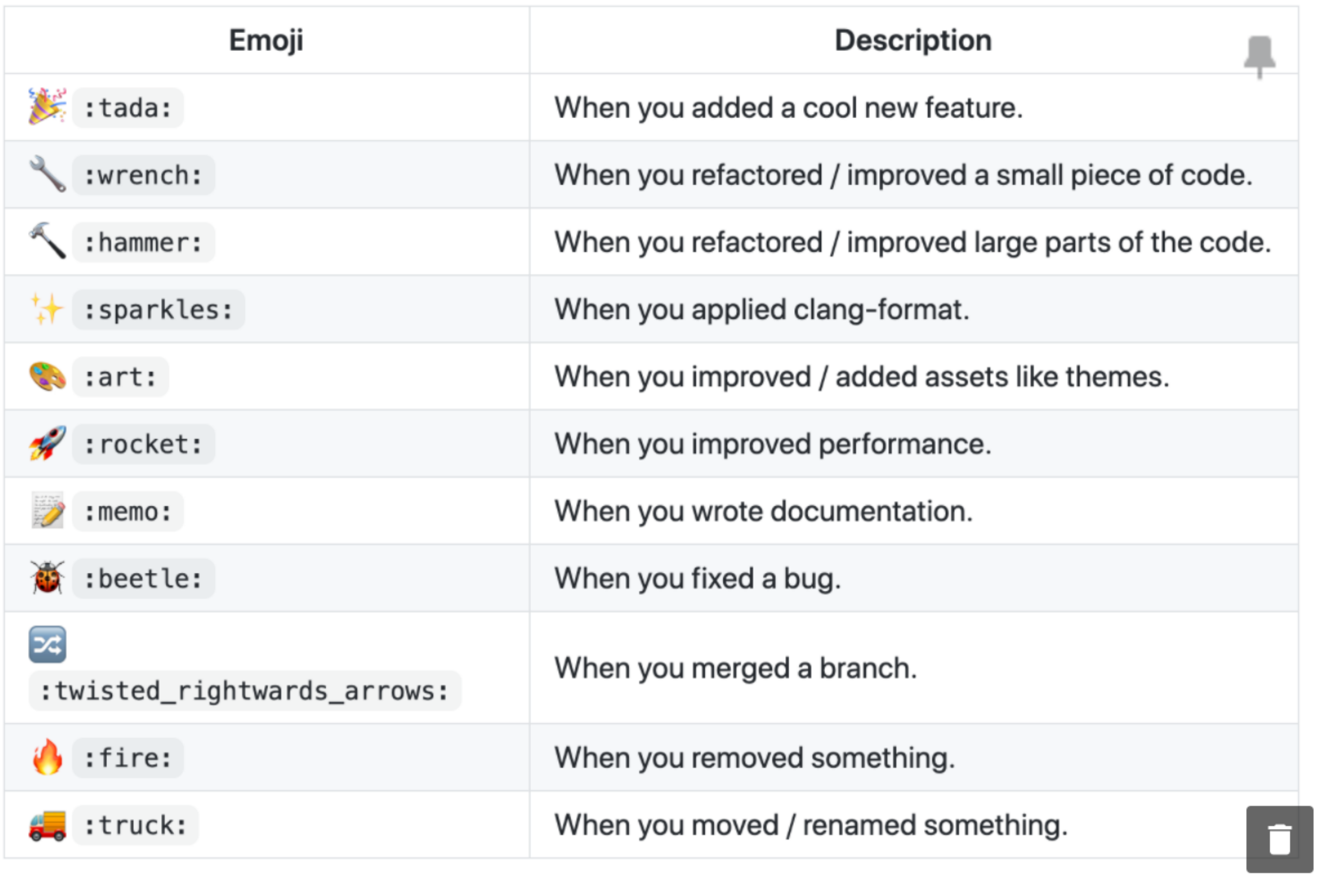
http://radekstepan.com



Milestones — A-Z Sorted by priority

rails/rails	6.1.4	20%
rails/rails	7.0	30%
rails/rails	6.0.4	80%

Figure 6. Example of Burndownchart (source here)



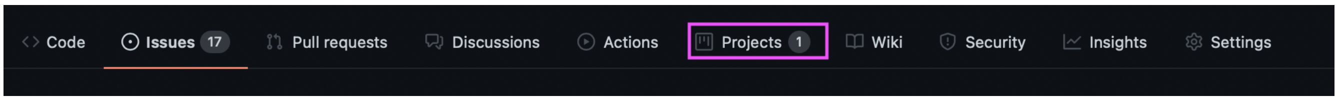
Emoji	Description
🎉 `:tada:`	When you added a cool new feature.
🔧 `:wrench:`	When you refactored / improved a small piece of code.
🔨 `:hammer:`	When you refactored / improved large parts of the code.
✨ `:sparkles:`	When you applied clang-format.
🎨 `:art:`	When you improved / added assets like themes.
🚀 `:rocket:`	When you improved performance.
📝 `:memo:`	When you wrote documentation.
🐞 `:beetle:`	When you fixed a bug.
🔀 `:twisted_rightwards_arrows:`	When you merged a branch.
🔥 `:fire:`	When you removed something.
🚚 `:truck:`	When you moved / renamed something.

Figure 7. Example of emoji usage convention

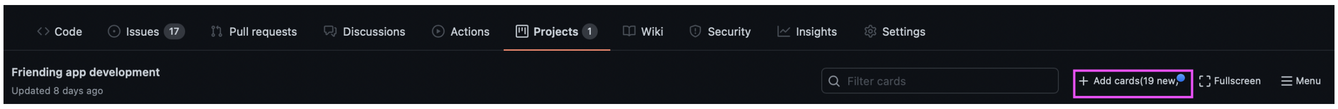
7. Issue management

It is very handful to connect you issues and tasks on a single project board!

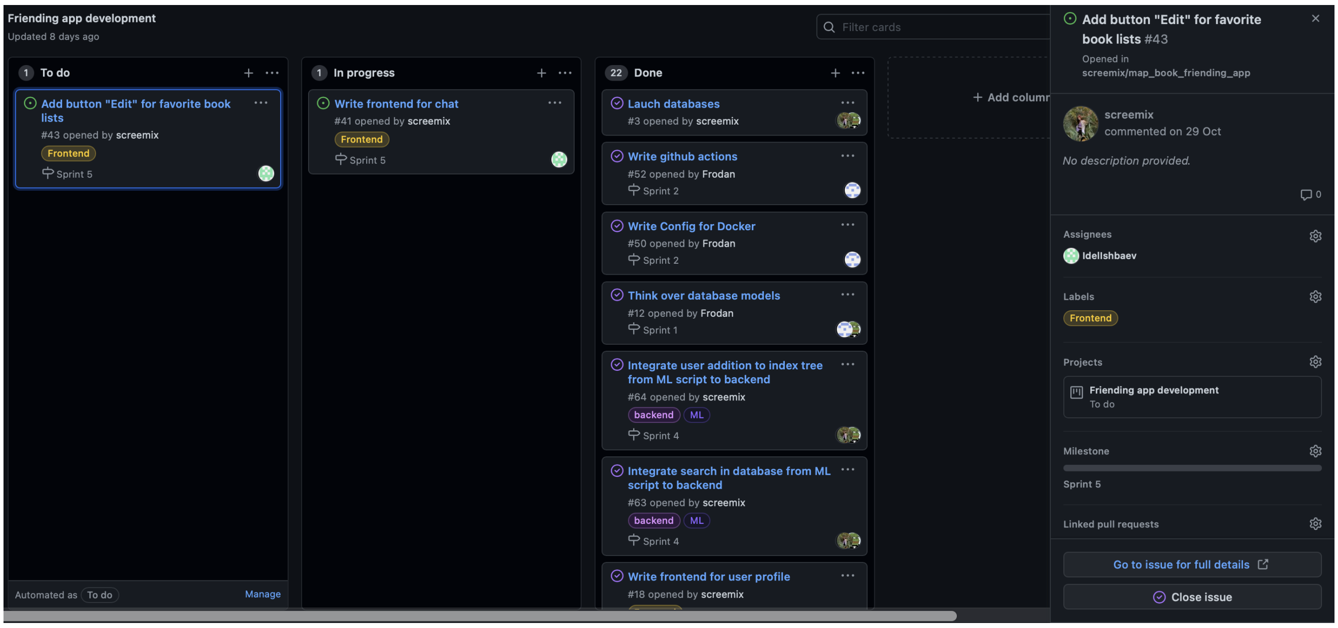
To access your project board or create a new one go to:



To add a new issue as a task on the project board go to:



And finally, enjoy your board with all the issues in there. Instead of simple kanban board you can even use techniques as YAGNI approach on a separate board to prioritize them.


Friending app development
Updated 8 days ago

Filter cards

1 To do

Add button "Edit" for favorite book lists
#43 opened by screemix
Frontend
Sprint 5

Automated as To do — Manage

1 In progress

Write frontend for chat
#41 opened by screemix
Frontend
Sprint 5

22 Done

Lauch databases
#3 opened by screemix

Write github actions
#52 opened by Frodan
Sprint 2

Write Config for Docker
#50 opened by Frodan
Sprint 2

Think over database models
#12 opened by Frodan
Sprint 1

Integrate user addition to index tree from ML script to backend
#64 opened by screemix
backend ML
Sprint 4

Integrate search in database from ML script to backend
#63 opened by screemix
backend ML
Sprint 4

Write frontend for user profile
#18 opened by screemix

+ Add column

Add button "Edit" for favorite book lists #43
Opened in screemix/map_book_friending_app

screemix commented on 29 Oct
No description provided.

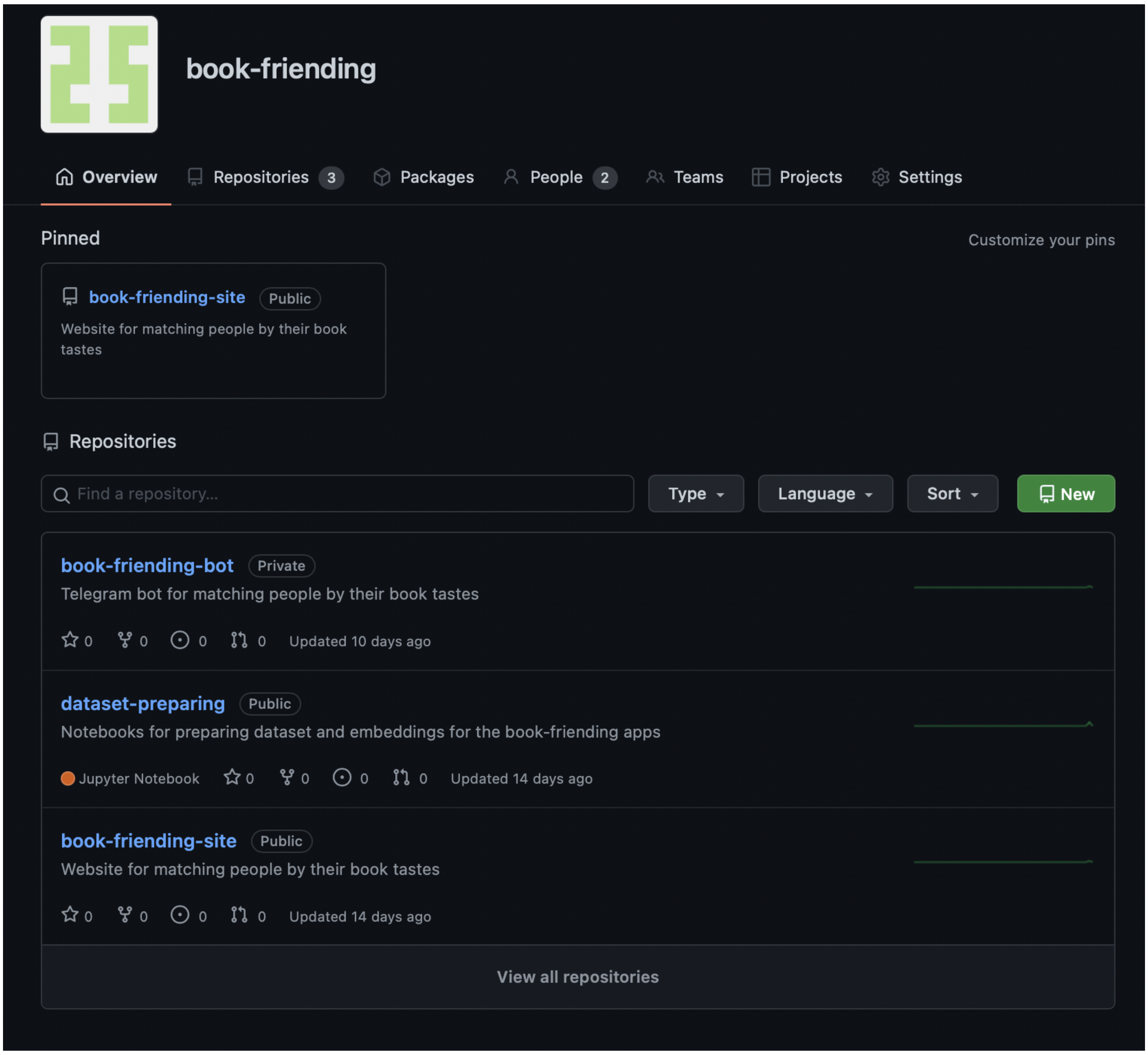
Assignees: Idellshbaev
Labels: Frontend
Projects: Friending app development — To do
Milestone: Sprint 5
Linked pull requests

Go to issue for full details
Close issue

8. Github organizations

Organizations are shared accounts where businesses and open-source projects can collaborate across many repositories at once.

If working on a project for several platforms is your case, this Github feature could be very convinient to use:



book-friending-site Public
Website for matching people by their book tastes

book-friending-bot Private
Telegram bot for matching people by their book tastes
0 0 0 0 Updated 10 days ago

dataset-preparing Public
Notebooks for preparing dataset and embeddings for the book-friending apps
Jupyter Notebook 0 0 0 0 Updated 14 days ago

book-friending-site Public
Website for matching people by their book tastes
0 0 0 0 Updated 14 days ago

9. US template good practice

Github provides with a possibilty to create templates of issues, so users will be following the same structure of US in a repository. The template should be in `.github/ISSUE_TEMPLATE/user-story-template.md`

Here is an example how it could look like:

```
---
[
The user story should have a reason to exist: what do I need as the user described in
the summary?
This part details any detail that could not be passed by the summary.
]
### Acceptance Criteria
1. [If I do A.]
1. [B should happen.]
[
Also, here are a few points that need to be addressed:
1. Constraint 1;
1. Constraint 2;
1. Constraint 3.
]
### Resources:
* Mockups: [Here goes a URL to or the name of the mockup(s) in inVision];
* Testing URL: [Here goes a URL to the testing branch or IP];
* Staging URL: [Here goes a URL to the feature on staging];
### Notes
[Some complementary notes if necessary:]
* > Here goes a quote from an email
* Here goes whatever useful information can exist…
* This tenplate is taken from https://github.com/AlphaFounders/style-
guide/blob/master/agile-user-story.md
---
```

10. Useful links

- https://www.datree.io/resources/github-best-practices