QGIS Documentation Guidelines

QGIS Project

mai 22, 2024

Cuprins

Documentația QGIS este disponibilă la https://docs.qgis.org. Pe măsură ce are loc procesul de scriere, comanda de compilare este rulată automat în fiecare zi (v. partea de jos a paginii pentru ora exactă) pentru fiecare dintre versiunile acceptate (versiunea de testare, versiunea cu suport pe termen lung (LTR) și viitorul LTR).

Fișierele sursă ale documentației QGIS sunt disponibile la https://github.com/qgis/QGIS-Documentation. În principal, pentru scrierea documentației QGIS se folosește sintaxa formatului reStructuredText (reST), cu excepția unor scrípturi din setul de instrumente Sphinx, dedicate post-procesării fișierelor HTML. Pentru informații generale despre aceste instrumente, consultați https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html sau https://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html.

Următoarele capitole vă arată:

- cum să gestionați fișierele sursă ale documentației folosind sistemul git și platforma GitHub pe care sunt stocate
- cum să modificați textele și să pregătiți capturile de ecran… într-un mod adecvat
- cum să efectuați partajarea și să vă asigurați că modificările sunt transmise documentelor oficiale.

Pentru a cunoaște regulile generale privind contribuția la proiectul QGIS, ați putea utiliza informațiile din secțiunea Implicați-vă în Comunitatea QGIS.

CAPITOLUL 1

O contribuție pas cu pas

Notă: Cu toate că pentru a demonstra procesul se folosește Documentația QGIS, comenzile și etapele prezentate mai jos se aplică întregului site QGIS.

If you are reading these lines, it is certainly because you are willing to contribute to writing QGIS documentation and are looking for a how-to. You have come to the right place! The current document will guide you through the different ways to achieve this objective, showing you the main steps to follow, the tricks you can use and the traps you should be aware of.

For any help, do not hesitate to either ask in a comment on the issue report you are trying to fix or write to the QGIS-community-team list. More details at Get involved in documentation.

Let's now dive into the process.

Documentation sources are stored using the git version control system and are available on GitHub at https://github.com/qgis/QGIS-Documentation. A list of issues to fix and features to explain can be found at https://github.com/qgis/QGIS-Documentation/issues.

Sfat: If you are a first-time contributor and do not know where to start from, you may be interested in tackling our welcoming issue reports.

There are two main ways, not mutually exclusive, to modify the files:

1. *Using the GitHub web interface*
2. *Using Git command line tools*.

1.1 Using the GitHub web interface

The GitHub web interface allows you to do the following:

- edit files
- preview and commit your changes
- make a pull request to have your changes inserted into the main repository
- create, update, or delete branches

If you are not yet familiar with git and GitHub vocabulary, you may want to read the GitHub Hello-world project to learn some basic vocabulary and actions that will be used below.

Notă: If you are fixing a reported issue

If you are making changes to fix an issue, add a comment to the issue report to assign it to yourself. This will prevent more than one person from working on the same issue.

1.1.1 Fork QGIS-Documentation

Assuming you already have a GitHub account, you first need to fork the source files of the documentation.

Navigate to the QGIS-Documentation repository page and click on the Fork button in the upper right corner.

In your GitHub account you will find a QGIS-Documentation repository (`https://github.com/<YourName>/QGIS-Documentation`). This repository is a copy of the official QGIS-Documentation repository where you have full write access and you can make changes without affecting the official documentation.

1.1.2 Make changes

There are different ways to contribute to QGIS documentation. We show them separately below, but you can switch from one process to the other without any harm.

Alternative 1: Use the `Edit on GitHub` shortcut

Pages on the QGIS documentation website can be edited quickly and easily by clicking on the `Edit on GitHub` link at the top right of each page.

1. This will open the file in the `qgis:master` branch with a message at the top of the page telling you that you don't have write access to this repo and your changes will be applied to a new branch of your repository.
2. Do your changes. Since the documentation is written using the reStructureText syntax, depending on your changes, you may need to rely on the *writing guidelines*.
3. When you finish, make a short comment about your changes and click on *Propose changes*. This will generate a new branch (`patch-xxx`) in your repository.
4. After you click on *Propose changes*, github will navigate to the *Comparing changes* page.
   - If you're done making changes, skip to *Compare changes* in the *Share your changes via Pull Request* section below.
   - If there are additional changes that you want to make before submitting them to QGIS, follow these steps:
     1. Navigate to your fork of QGIS-Documentation (`https://github.com/<YourName>/QGIS-Documentation`)
     2. Click on

and search for the `patch-xxx` branch. Select this patch branch. The

button will now say *Branch: patch-xxx*
     3. Jump down to *Modify files* below.

Notă: The `Edit on GitHub` shortcut is also available in the drop-down menu at the bottom of the left sidebar.

Alternative 2: Create an ad hoc branch in your documentation repository

You can edit files directly from your fork of the QGIS Documentation.

First, make sure that your `master` branch is up to date with `qgis:master` branch. To do so:

1. Go to the main page of your repository, i.e. `https://github.com/<YourName>/QGIS-Documentation`. The `master` branch should be active with a mention whether it is up to date with `qgis/QGIS-Documentation:master` or not.

   If it has commits ahead the upstream branch, you better use the previous *shortcut button alternative* until you align your `master` branch.

   If it only has commits behind:

   1. Expand the *Fetch Upstream* drop-down menu on the right. You can
      - *Compare* the branches and see new changes in the main repository
      - *Fetch and merge*: takes changes from the upstream branch to yours.
   2. Let's click *Fetch and merge*: after the process, your branch is mentioned as up to date with `qgis/QGIS-Documentation:master`.

2. Click on Branch: master in the upper left corner of your forked QGIS-Documentation repository and enter a unique name in the text field to create a new branch . The name of the new branch should relate to the problem you intend to fix. The Branch: master button should now say *Branch: branch_name*
3. You are ready to start new changes on top of it.

Atenție: Do your changes in an ad hoc branch, never in the `master` branch

By convention, avoid making changes in your `master` branch except when you merge the modifications from the `master` branch of `qgis/QGIS-Documentation` into your copy of the QGIS-Documentation repository. Separate branches allow you to work on multiple problems at the same time without interfering with other branches. If you make a mistake you can always delete a branch and start over by creating a new one from the master branch.

1.1.3 Modify files

1. Browse the source files of your fork of QGIS-Documentation to the file that needs to be modified
2. Make your modifications following the *writing guidelines*
3. When you finish, navigate to the **Commit Changes** frame at the bottom of the page, make a short comment about your changes, and click on *Commit Changes* to commit the changes directly to your branch. Make sure *Commit directly to the branch_name branch.* is selected.
4. Repeat the previous steps for any other file that needs to be updated to fix the issue

1.1.4 Share your changes via Pull Request

You need to make a pull request to integrate your changes into the official documentation.

Notă: If you used an `Edit on GitHub` link

After you commit your changes GitHub will automatically open a new page comparing the changes you made in your `patch-xxx` branch to the `qgis/QGIS-Documentation` master branch.

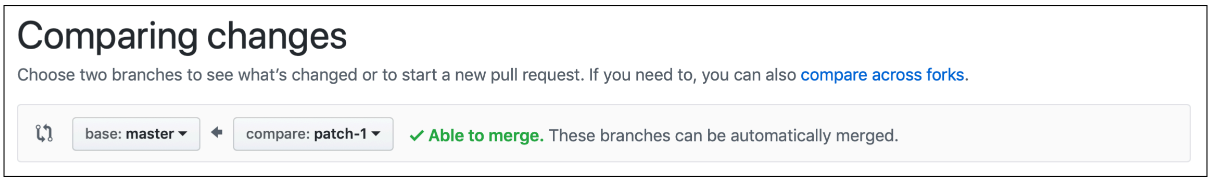
Skip to *Step 2* below.

Start a new pull request

Navigate to the main page of the QGIS-Documentation repository and click on *New pull request*.

Compare changes

If you see two dialog boxes, one that says `base:master` and the other `compare:branch_name` (see figure), this will only merge your changes from one of your branches to your master branch. To fix this click on the *compare across forks* link.


Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

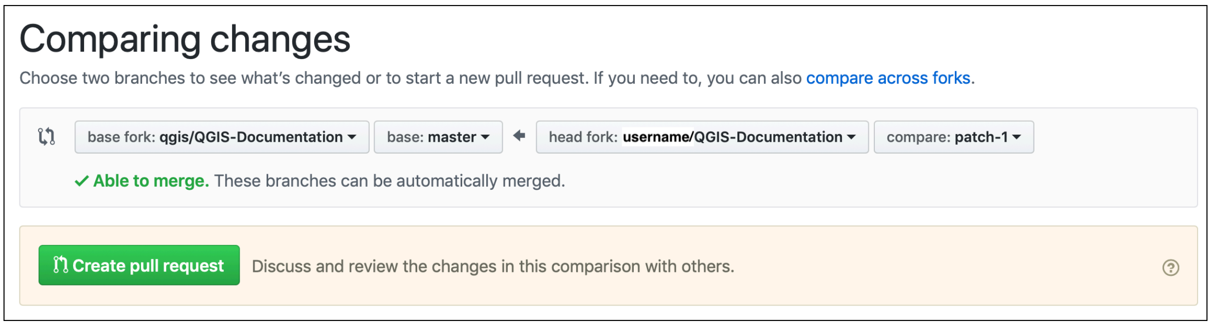
base: master ▾ ← compare: patch-1 ▾ ✓ Able to merge. These branches can be automatically merged.

Fig. 1.1: If your *Comparing changes* page looks like this, click on the *compare across forks* link.

You should see four drop-down menus. These will allow you to compare the changes that you have made in your branch with the master branch that you want to merge into. They are:

- **base fork**: the fork that you want to merge your changes into
- **base**: the branch of the base fork that you want to merge your changes into
- **head fork**: the fork that has changes that you want to incorporate into the base fork
- **compare**: the branch with those changes

Select `qgis/QGIS-Documentation` as the base fork with `master` as base, set the head fork to your repository `<YourName>/QGIS-Documentation`, and set compare to your modified branch.



Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

base fork: qgis/QGIS-Documentation ▾ base: master ▾ ← head fork: username/QGIS-Documentation ▾ compare: patch-1 ▾

✓ Able to merge. These branches can be automatically merged.

Discuss and review the changes in this comparison with others.

Fig. 1.2: Comparing changes between `qgis/QGIS-Documentation` and your repository

A green check with the words **Able to merge** shows that your changes can be merged into the official documentation without conflicts.

Click the *Create pull request* button.

Atenționare: If you see ✕ Can't automatically merge.

This means that there are conflicts. The files that you are modifying are not up to date with the branch you are targeting because someone else has made a commit that conflicts with your changes. You can still create the pull request but you'll need to fix any *conflicts* to complete the merge.

Sfat: Though being translated, the latest version of QGIS documentation is still maintained and existing issues are fixed. If you are fixing issues for a different release, change **base** from `master` to the appropriate `release_...` branch in the steps above.

Describe your pull request

A text box will open: fill in any relevant comments for the issue you are addressing.

If this relates to a particular issue, add the issue number to your comments. This is done by entering # and the issue number (*e.g.* `#1234`). If preceded by terms like `fix` or `close`, the concerned issue will be closed as soon as the pull request is merged.

Add links to any documentation pages that you are changing.

Click on *Create pull request*.

Review and comment pull request

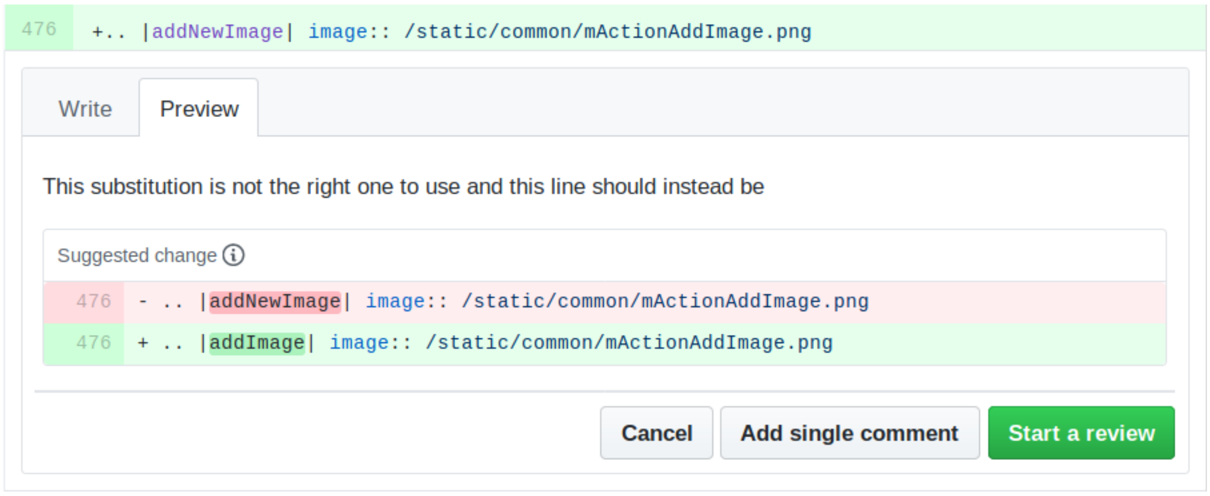
As seen above, anyone can submit modifications to the documentation through pull requests. Likewise anyone can review pull requests with questions and comments. Perhaps the writing style doesn't match the project guidelines, the change is missing some major details or screenshots, or maybe everything looks great and is in order. Reviewing helps to improve the quality of the contribution, both in form and substance.

To review a pull request:

1. Navigate to the pull requests page and click on the pull request that you want to comment on.
2. At the bottom of the page you will find a text box where you can leave general comments about the pull request.
3. To add comments about specific lines,
   1. Click on Files changed and find the file you want to comment on. You may have to click on *Display the source diff* to see the changes.
   2. Scroll to the line you want to comment on and click on the +. That will open a text box allowing you to leave a comment.

Specific line comments can be published either:

- as single comments, using the *Add single comment* button. They are published as you go. Use this only if you have few comments to add or when replying to another comment.
- or as part of a review, pressing the *Start a review* button. Your comments are not automatically sent after validation, allowing you to edit or cancel them afterwards, to add a summary of the main points of the review or global instructions regarding the pull request and whether you approve it or not. This is the convenient way since it's more flexible and allows you to structure your review, edit the comments, publish when you are ready and send a single notification to the repository followers and not one notification for each comment. Get more details.



```
476  +.. |addNewImage| image:: /static/common/mActionAddImage.png
```

This substitution is not the right one to use and this line should instead be

```
476  - .. |addNewImage| image:: /static/common/mActionAddImage.png
476  + .. |addImage| image:: /static/common/mActionAddImage.png
```

Fig. 1.3: Commenting a line with a change suggestion

Line comments can embed suggestions that the pull request writer can apply to the pull request. To add a suggestion, click the Insert a suggestion button on top of the comment text box and modify the text within the suggestion block.

Sfat: Prefer committing suggestions to your pull request as a batch

As a pull request author, when directly incorporating reviewers» feedback in your pull request, avoid using the *Commit suggestion* button at the bottom of the comment when you have many suggestions to address and prefer adding them as a batch commit, that is:

1. Switch to the Files changed tab
2. Press *Add suggestion to batch* for each rewording you'd like to include. You will see a counter increasing as you go.
3. Press any of the *Commit suggestions* button when you are ready to apply the suggestions to your pull request, and enter a message describing the changes.

This will add all the modifications to your branch as a single commit, resulting in a more legible history of changes and less notifications for the repository followers. Incidentally, proceeding as this will also save you many clicks.

Make corrections

A new pull request will automatically be added to the Pull requests list. Other editors and administrators will review your pull request and they may make suggestions or ask for corrections.

A pull request will also trigger automated build checks (eg, for rst formatting, python code syntaxes), and reports are displayed at the bottom of the page. If an error is found, a red cross will appear next to your commit. Click on the red cross or on `Details` in the summary section at the bottom of the pull request page to see the details of the error. You'll have to fix any reported errors or warnings before your changes are committed to the `qgis/QGIS-Documentation` repository.

You can make modifications to your pull request until it is merged with the main repository, either to improve your request, to address requested modifications, or to fix a build error.

To make changes click on the Files changed tab in your pull request page and click the pencil button next to the filename that you want to modify.

Any additional changes will be automatically added to your pull request if you make those changes to the same branch that you submitted in your pull request. For this reason, you should only make additional changes if those changes relate to the issue that you intend to fix with that pull request.

If you want to fix another issue, create a new branch for those changes and repeat the steps above.

An administrator will merge your contribution after any build errors are corrected, and after you and the administrators are satisfied with your changes.

1.1.5 Delete your merged branch

You can delete the branch after your changes have been merged. Deleting old branches saves you from having unused and outdated branches in your repository.

1. Navigate to your fork of the QGIS-Documentation repository (`https://github.com/<YourName>/QGIS-Documentation`).
2. Click on the *Branches* tab. Below *Your branches* you'll see a list of your branches.
3. Click on the Delete this branch icon to delete any unwanted branches.

1.2 Utilizarea instrumentelor Git pentru linia de comandă

The GitHub web interface is an easy way to update the QGIS-documentation repo with your contributions, but it doesn't offer tools to:

- group your commits and clean your change history
- fix possible conflicts with the main repo
- build the documentation to test your changes

You need to install git on your hard drive in order to get access to more advanced and powerful tools and have a local copy of the repository. Some basics you may often need are exposed below. You'll also find rules to care about even if you opt for the web interface.

În exemplele de cod de mai jos, liniile care încep cu `$` prezintă comenzile pe care ar trebui să le introduceți, în timp ce `#` prefixează comentariile.

1.2.1 Depozitul local

Now you are ready to get a local clone of **your** QGIS-Documentation repository.

You can clone your QGIS repository using the web URL as follows:

```
# move to the folder in which you intend to store the local repository
$ cd ~/Documents/Development/QGIS/
$ git clone https://github.com/<YourName>/QGIS-Documentation.git
```

The former command line is simply an example. You should adapt both the path and the repository URL, replacing `<YourName>` with your github user name.

Check the following:

```
# Enter the local repository
$ cd ./QGIS-Documentation
$ git remote -v
origin  https://github.com/<YourName>/QGIS-Documentation.git (fetch)
origin  https://github.com/<YourName>/QGIS-Documentation.git (push)
$ git branch
* master
```

- *originea* este numele depozitului dumneavoastră, accesibil de la distanță, cu Documentație QGIS.
- *master* este ramura principală, implicită. Nu ar trebui să introduceți codul dvs. aici! **Niciodată!**

Alternatively you can clone your QGIS repository using the SSH protocol:

```
# move to the folder in which you intend to store the local repository
$ cd ~/Documents/Development/QGIS/
$ git clone git@github.com:<YourName>/QGIS-Documentation.git
```

Sfat: Permission denied (publickey) error?

If you get a Permission denied (publickey) error with the former command, there may be a problem with your SSH key. See GitHub help for details.

Check the following if you used the SSH protocol:

```
# Enter the local repository
$ cd ./QGIS-Documentation
$ git remote -v
```

(continues on next page)

(continuare din pagina precedentă)

```
origin  git@github.com:<YourName>/QGIS-Documentation.git (fetch)
origin  git@github.com:<YourName>/QGIS-Documentation.git (push)
$ git branch
* master
```

You can start to work here but in the long term process you will get a lot of issues when you will push your contribution (called Pull Request in github process) as the master branch of the qgis/QGIS-Documentation repository will diverge from your local/remote repository. You then need to keep track of the main remote repository and work with branches.

1.2.2 Adăugarea unui alt depozit accesibil de la distanță

Pentru a putea urmări activitatea din proiectul principal, adăugați un nou depozit, accesibil de la distanță, în depozitul local. Acest nou depozit va fi depozitul Documentației QGIS, al proiectului QGIS:

```
$ git remote add upstream https://github.com/qgis/QGIS-Documentation.git
$ git remote -v
origin  https://github.com/<YourName>/QGIS-Documentation.git (fetch)
origin  https://github.com/<YourName>/QGIS-Documentation.git (push)
upstream        https://github.com/qgis/QGIS-Documentation.git (fetch)
upstream        https://github.com/qgis/QGIS-Documentation.git (push)
```

Similarly, you can use the SSH protocol to add a remote repository in your local repository:

```
$ git remote add upstream git@github.com:qgis/QGIS-Documentation.git
$ git remote -v
origin  git@github.com:<YourName>/QGIS-Documentation.git (fetch)
origin  git@github.com:<YourName>/QGIS-Documentation.git (push)
upstream        git@github.com:qgis/QGIS-Documentation.git (fetch)
upstream        git@github.com:qgis/QGIS-Documentation.git (push)
```

Deci, acum aveți posibilitatea de a alege între două depozite accesibile de la distanță:

- *origin* to push your local branch in **your** remote repository
- *upstream* to merge (if you have right to do so) your contribution to the official one OR to update your master branch of local repository from the master branch of the official repository.

Notă: *upstream* is just a label, a kind of standard name but you can call it as you want.

1.2.3 Update your base branch

Before working on a new contribution, you should always update your master branch in your local repository. Assuming you are willing to push changes to the testing documentation, run the following command lines:

```
# switch to master branch (it is easy to forget this step!)
$ git checkout master
# get "information" from the master branch in the upstream repository
# (aka qgis/QGIS-Documentation's repository)
$ git fetch upstream master
# merge update from upstream/master to the current local branch
# (which should be master, see step 1)
$ git merge upstream/master
# update **your** remote repository (aka <YourName>/QGIS-Documentation)
$ git push origin master
```

Now you have your local and remote repositories which both have their `master` branch up to date with the official `master` branch of QGIS-Documentation. You can start to work on your contribution.

Notă: Switch the branch if you wish to contribute to released doc

Along with the testing documentation, we continue to fix issues in the latest release, meaning that you can also contribute to it. Follow the previous section sample code, replacing `master` with the corresponding branch of the latest documentation.

1.2.4 Contribute to your production branch

Now that your base branch is updated, you need to create a dedicated branch in which you add your contribution. Always work on a branch other than the base branch! Always!

```
# Create a new branch
$ git checkout -b myNewBranch
# checkout means go to the branch
# and -b flag creates a new branch if needed, based on current branch
# Let's check the list of existing branches (* indicates the current branch)
$ git branch
master
release_2.18
...
* myNewBranch
# You can now add your contribution, by editing the concerned file(s)
# with any application (in this case, vim is used)
$ vim myFile
# once done
$ git add myFile
$ git commit
```

Few words about commit/push commands:

- try to commit only one contribution (atomic change) i.e. address only one issue
- try to explain carefully what you change in the title of your commit and in the description. The first line is a title and should start by an upper case letter and have 80 characters length, don't end with a `.`. Be concise. Your description can be longer, end with a `.` and you can give much more details.
- use a `#` with a number to refer to an issue. Prefix with `Fix` if you fix the ticket: your commit will close the ticket.

Now that your changes are saved and committed in your local branch, you need to send them to your remote repository in order to create pull request:

```
$ git push origin myNewBranch
```

1.2.5 Partajați Modificările Proprii

Now you can go to your github repository and *create a Pull Request* as exposed in a previous section. Ensure you create a PR from your branch to the remote branch you are targetting in the official QGIS-Documentation repository.

1.2.6 Curățarea depozitului local și a aceluia aflat la distanță

După ce PR-ul a fost încorporat în Documentația QGIS oficială, puteți șterge ramura. În cazul în care lucrați mult în acest mod, în câteva săptămâni, vă veți alege cu o mulțime de ramuri nefolositoare. De aceea, pentru a păstra curat depozitul curățați-l în acest fel:

```
# delete local branch
$ git branch -d myNewBranch
# Remove your remote myNewBranch by pushing nothing to it
$ git push origin :myNewBranch
```

Și nu uitați să actualizați ramura `master` din depozit local!

1.3 Lecturi suplimentare

- Other than the Github web interface and the git command line tools exposed above, there are also GUI applications you can use to create and manage your contributions to the documentation.
- When the changes in the pull request are conflicting with recent changes pushed to the target branch, the conflicts need to be resolved before a merge is possible:
  - if the conflict relates to few competing lines, a *Resolve conflicts* button is available in the GitHub pull request page. Press the button and resolve the issue as explained at Resolving a merge conflict on GitHub
  - if the conflict involves files renaming or removal, then you'd need to resolve the conflict using git command lines. Typically, you have to first rebase your branch over the target branch using `git rebase targetBranch` call and fix the conflicts that are reported. Read more at Resolving a merge conflict using the command line
- Sometimes, at the end of the proofreading process, you may end up with changes split into multiple commits that are not necessarily worth it. Git command lines help you squash these commits to a smaller number and more meaningful commit messages. Some details at Using git rebase on the command line

CAPITOLUL 2

Writing Guidelines

In general, when creating reST documentation for the QGIS project, please follow the Python documentation style guidelines. For convenience, we provide a set of general rules we rely on for writing QGIS documentation below.

2.1 Writing Documentation

2.1.1 Headlines

To each webpage of the documentation corresponds a `.rst` file.

Sections used to structure the text are identified through their title which is underlined (and overlined for the first level). Same level titles must use same character for underline adornment. In QGIS Documentation, you should use following styles for chapter, section, subsection and minisec.

```
********
Chapter
********

Section
=======

Subsection
----------

Minisec
.......

Subminisec
^^^^^^^^^^
```

2.1.2 Lists

Lists are useful for structuring the text. Here are some simple rules common to all lists:

- Start all list items with a capital letter
- Do not use punctuation after list items that only contain a single simple sentence
- Use period (`.`) as punctuation for list items that consist of several sentences or a single compound sentence

2.1.3 Indentation

Indentation in ReStructuredText should be aligned with the list or markup *marker*. It is also possible to create block quotes with indentation. See the Specification

```
#. In a numbered list, there should be
   three spaces when you break lines
#. And next items directly follow

   * Nested lists
   * Are also possible
   * And when they also have
     a line that is too long,
     the text should be naturally
     aligned
   * and be in their own paragraph

However, if there is an unindented paragraph, this will reset the numbering:

#. This item starts at 1 again
```

2.1.4 Inline Tags

You can use tags to emphasize items.

- **Menu GUI**: to mark a complete sequence of menu selections, including selecting submenus and choosing a specific operation, or any subsequence of such a sequence.

  ```
  :menuselection:`menu --> submenu`
  ```

- **Dialogs and Tab titles**: Labels presented as part of an interactive user interface including window titles, tab titles, button and option labels.

  ```
  :guilabel:`title`
  ```

- **Filenames and directories**

  ```
  :file:`README.rst`
  ```

- **Icons with popup text**

  ```
  |icon| :sup:`popup_text`
  ```

  (see *image* below).

- **Keyboard shortcuts**

  ```
  :kbd:`Ctrl+B`
  ```

  will show `Ctrl+B`

  When describing keyboard shortcuts, the following conventions should be used:

  – Letter keys are displayed using uppercase: `S`

  – Special keys are displayed with an uppercase first letter: `Esc`

  – Key combinations are displayed with a + sign between keys, without spaces: `Shift+R`

- **User text**

  ```
  ``label``
  ```

- **Layer names** When referring to layers, format as inline code:

  ```
  ``layer name``
  ```

2.1.5 Labels/references

Anchors inside the text can be used to create hyperlinks to sections or pages.

The example below creates the anchor of a section (e.g., Label/reference title)

```
.. _my_anchor:

Label/reference
---------------
```

To call the reference in the **same page**, use

```
see my_anchor_ for more information.
```

which will return:

see *my_anchor* for more information.

Notice that it will jump to the line/thing following the «anchor». You do not need to use apostrophes, but you do need to have empty lines after the anchor.

Another way to jump to the same place **from anywhere in the documentation** is to use the `:ref:` role.

```
see :ref:`my_anchor` for more information.
```

which will create a link with the caption instead (in this case the title of this section!):

see *Labels/references* for more information.

So, reference 1 (*my_anchor*) and reference 2 (*Labels/references*). Because the reference often displays a full caption, it is not really necessary to use the word *section*. Note that you can also use a custom caption to describe the reference:

```
see :ref:`Label and reference <my_anchor>` for more information.
```

which returns:

see *Label and reference* for more information.

2.1.6 Figures and Images

Pictures

To insert an image, use

```
.. figure:: /static/common/logo.png
   :width: 10 em
```

which returns

Replacement

You can put an image inside text or add an alias to use everywhere. To use an image inside a paragraph, first create an alias in the `source/substitutions.txt` file:

```
.. |nice_logo| image:: /static/common/logo.png
               :width: 1 em
```

and then call it in your paragraph:

```
My paragraph begins here with a nice logo |nice_logo|.
```

This is how the example will be displayed:

My paragraph begins here with a nice logo .

To allow preview rendering in GitHub that is as close as possible to HTML rendering, you will also need to add the image replacement call at the end of the file you changed. This can be done by copy-pasting it from `substitutions.txt` or by executing the `scripts/find_set_subst.py` script.

Notă: Currently, to ensure consistency and help in the use of QGIS icons, a list of aliases is built and available in the *Substituții* chapter.

Figure

```
.. _figure_logo:

.. figure:: /static/common/logo.png
   :width: 20 em
   :align: center

   A caption: A logo I like
```

The result looks like this:

Fig. 2.1: A caption: A logo I like

To avoid conflicts with other references, always begin figure anchors with `_figure_` and use terms that easily connect to the figure caption. While only the centered alignment is mandatory for the image, feel free to use any other options for figures (such as `width`, `height`, `scale`...) if needed.

The scripts will insert an automatically generated number before the caption of the figure in the generated HTML and PDF versions of the documentation.

To use a caption (*see My caption*) just insert indented text after a blank line in the figure block.

A figure can be referenced using the reference label like this:

```
see :numref:`figure_logo`
```

renders like this:

see Fig. 2.1

This is the preferred way of referencing figures.

Notă: For `:numref:` to work, the figure **must have a caption**.

It is possible to use `:ref:` instead of `:numref:` for reference, but this returns the full caption of the image.

```
see :ref:`figure_logo`
```

renders like this:

see *Un text explicativ: Un logo care-mi place*

Tables

A simple table can be coded like this

```
=======  =======  =======
x        y        z
=======  =======  =======
1        2        3
4                 5
=======  =======  =======
```

It will render like this:

x	y	z
1	2	3
4		5

Use a \ (backslash) followed by an empty space to leave an empty space.

You can also make more complicated tables and reference them:

```
.. _my_drawn_table:

+---------------+--------------------+
| Windows       | macOS              |
+---------------+--------------------+
| |win|         | |osx|              |
+---------------+--------------------+
| and of course not to forget |nix|  |
+------------------------------------+

My drawn table, mind you this is unfortunately not regarded as a caption

You can reference it like this: my_drawn_table_.
```

The result:

Windows	macOS
	X
și, desigur, să nu uităm de	

My drawn table, mind you this is unfortunately not regarded as a caption

Puteți să-l referențiați la fel ca în *my_drawn_table*.

For even more complex tables, it is easier to use `list-table`:

```
.. list-table::
   :header-rows: 1
   :widths: 20 20 20 40

   * - What
     - Purpose
     - Key word
     - Description
   * - **Test**
     - ``Useful test``
     - complexity
     - Geometry.  One of:

       * Point
       * Line
```

The result:

What	Scop	Key word	Descriere
Test	`Useful test`	complexity	Geometry. One of: • Punct • Line

2.1.7 Index

An index is a handy way to help the reader find information in a document. QGIS documentation provides some essential indices. There are a few rules that help us provide a set of indices that are really useful (coherent, consistent and really connected to each other):

- An index should be human readable, understandable and translatable; an index can be made from many words but you should avoid any unneeded `_`, `-`... characters to link them i.e., `Loading layers` instead of `loading_layers` or `loadingLayers`.
- Capitalize only the first letter of the index unless the word has a particular spelling. E.g., `Loading layers`, `Atlas generation`, `WMS`, `pgsql2shp`.
- Keep an eye on the existing Index list in order to reuse the most convenient expression with the right spelling and avoid unnecessary duplicates.

Several index tags exist in RST. You can use the inline `:index:` tag within normal text:

```
QGIS can load several :index:`Vector formats` supported by GDAL ...
```

Or you can use the `.. index::` block-level markup which links to the beginning of the next paragraph. Because of the rules mentioned above, it is recommended to use the block-level tag:

```
.. index:: WMS, WFS, Loading layers
```

It is also recommended to use index parameters such as `single`, `pair` and `see`, in order to build a more structured and interconnected index table. See Index generating for more information on index creation.

2.1.8 Comentarii Speciale

Sometimes, you may want to emphasize some points of the description, either to warn, remind or give some hints to the user. In QGIS Documentation, we use reST special directives such as `.. warning::`, `.. seealso::``, ``.. note::` and `.. tip::`. These directives generate frames that highlight your comments. See Paragraph Level markup for more information. A clear and appropriate title is required for both warnings and tips.

```
.. tip:: **Always use a meaningful title for tips**

   Begin tips with a title that summarizes what it is about. This helps
   users to quickly overview the message you want to give them, and
   decide on its relevance.
```

2.1.9 Secvențe De Cod

You may also want to give examples and insert code snippets. In this case, write the comment below a line with the `::` directive inserted. For a better rendering, especially to apply color highlighting to code according to its language, use the code-block directive, e.g. `.. code-block:: xml`. More details at Showing code.

Notă: While texts in note, tip and warning frames are translatable, be aware that code block frames do not allow translation. So avoid comments not related to the code and keep comments as short as possible.

2.1.10 Note de subsol

Please note: Footnotes are not recognized by any translation software and it is also not converted to pdf format properly. So, if possible, don't use footnotes within any documentation.

This is for creating a footnote (showing as example[1])

```
blabla [1]_
```

Care va indica:

2.2 Gestiunea Capturilor de Ecran

2.2.1 Adăugarea noilor capturi de ecran

Here are some hints to create new, nice looking screenshots. The images should be placed in an image (`img/`) folder that is located in the same folder as the referencing `.rst` file.

- You can find some prepared QGIS-projects that are used to create screenshots in the `./qgis-projects` folder of this repository. This makes it easier to reproduce screenshots for the next version of QGIS. These projects use the QGIS Sample Data (aka Alaska Dataset), which should be unzipped and placed in the same folder as the QGIS-Documentation Repository.
- Reduce the window to the minimal space needed to show the feature (taking the whole screen for a small modal window > overkill)
- The less clutter, the better (no need to activate all the toolbars)
- Don't resize them in an image editor; the size will be set into the `.rst` files if necessary (downscaling the dimensions without properly upping the resolution > ugly)
- Cut the background

[1] Actualizări ale plugin-urilor de bază

- Make the top corners transparent if the background is not white
- Set print size resolution to `135 dpi` (e.g. in Gimp set the print resolution *Image* ► *Print size* and save). This way, images will be at original size in html and at a good print resolution in the PDF. You can also use ImageMagick convert command to do a batch of images:

  ```
  convert -units PixelsPerInch input.png -density 135 output.png
  ```

- Save them as `.png` (to avoid `.jpeg` artifacts)
- The screenshot should show the content according to what is described in the text

Sfat: If you are on Ubuntu, you can use the following command to remove the global menu function and create smaller application screens with menus:

```
sudo apt autoremove appmenu-gtk appmenu-gtk3 appmenu-qt
```

2.2.2 Translated Screenshots

Here are some additional hints for those that want to create screenshots for a translated user guide:

Translated images should be placed in a `img/<your_language>/` folder. Use the same filename as the english «original» screenshot.

2.3 Documentarea Algoritmilor Processing

If you want to write documentation for Processing algorithms, consider these guidelines:

- Processing algorithm help files are part of QGIS User Guide, so use the same formatting as User Guide and other documentation.
- Each algorithm documentation should be placed in the corresponding **provider** folder and **group** file, e.g. the algorithm *Voronoi polygon* belongs to the *QGIS* provider and to the group *vectorgeometry*. So the correct file to add the description is: `source/docs/user_manual/processing_algs/qgis/vectorgeometry.rst`.

  Notă: Before starting to write the guide, check if the algorithm is already described. In this case, you can enhance the existing description.

- It is **extremely** important that each algorithm has an *anchor* that corresponds to the provider name + the unique name of the algorithm itself. This allows the Help button to open the Help page of the correct section. The anchor should be placed **above** the title, e.g. (see also the *Labels/references* section):

  ```
  .. _qgisvoronoipolygons:

  Voronoi polygons
  ----------------
  ```

  To find out the algorithm name you can just hover the mouse on the algorithm in the Processing toolbox.

- Avoid using „This algorithm does this and that…" as the first sentence in the algorithm description. Try to use more general expressions like:

  ```
  Takes a point layer and generates a polygon layer containing the...
  ```

- Avoid describing what the algorithm does by replicating its name and please don't replicate the name of the parameter in the description of the parameter itself. For example if the algorithm is `Voronoi polygon` consider to describe the `Input layer` as `Layer to calculate the polygon from`.
- Indicate in the description whether the algorithm has a default shortcut in QGIS or supports in-place editing.
- Add images! A picture is worth a thousand words! Use `.png` format and follow the general guidelines for documentation (see the *Figures and Images* section for more info). Put the image file in the correct folder, i.e. the `img` folder next to the `.rst` file you are editing.
- If necessary, add links in the „See also" section that provide additional information about the algorithm (e.g., publications or web-pages). Only add the „See also" section if there is really something to see. As a good practice, the „See also" section can be filled with links to similar algorithms.
- Give clear explanation for algorithm parameters and outputs: take inspiration from existing algorithms.
- Avoid duplicating detailed description of algorithm options. Add this information in the parameter description.
- Avoid adding information about the vector geometry type in the algorithm or parameter description, as this information is already available in the parameter descriptions.
- Add the default value of the parameter, e.g.:

```
* - **Number of points**
  - ``NUMBER_OF_POINTS``
  - [number]

    Default: 1
  - Number of points to create
```

- Describe the *type* of input supported the parameters. There are several types available you can pick one from:

Parameter/Output type	Descriere	Visual indicator
Point vector layer	`vector: point`	∴
Line vector layer	`vector: line`	
Polygon vector layer	`vector: poligon`	
Generic vector layer	`vector: any`	
Vector field numeric	`tablefield: numeric`	**1.2**
Vector field string	`tablefield: string`	**abc**
Vector field generic	`tablefield: any`	
Raster layer	`raster`	▦
Raster band	`raster band`	
HTML file	`html`	
Table layer	`table`	▤
Expresie	`expression`	ℰ
Point geometry	`coordinates`	
Extinderea	`extent`	
CRS	`crs`	
Enumeration	`enumeration`	… ▼
List	`list`	
Numărul	`number`	1,00
Șirul	`string`	Display name lakes.shp
Boolean	`boolean`	✓
Folder path	`folder`	
Fișier	`file`	

continues on next page

Tabelul 2.1 – continuare din pagina precedentă

Parameter/Output type	Descriere	Visual indicator
Matrix	`matrix`	
Strat	`layer`	
Same output type as input type	`same as input`	
Definition	`definition`	
Punct	`point`	
MultipleLayers	`multipleLayers`	
Intervalul	`range`	
AuthConfig	`authconfig`	
Suprafață poliedrică	`mesh`	
Layout	`layout`	
LayoutItem	`layoutitem`	
Color	`color`	
Scale	`scale`	

- Study an existing and well documented algorithm, and copy all the useful layouts.
- When you are finished, just follow the guidelines described in *O contribuție pas cu pas* to commit your changes and make a Pull Request

Here is an example of an existing algorithm to help you with the layout and the description:

```
.. _qgiscountpointsinpolygon:

Count points in polygon
-----------------------
Takes a point and a polygon layer and counts the number of points from the
point layer in each of the polygons of the polygon layer.
A new polygon layer is generated, with the exact same content as the input
polygon layer, but containing an additional field with the points count
corresponding to each polygon.

.. figure:: img/count_points_polygon.png
  :align: center

  The labels in the polygons show the point count

An optional weight field can be used to assign weights to each point.
Alternatively, a unique class field can be specified. If both options
are used, the weight field will take precedence and the unique class field
will be ignored.

``Default menu``: :menuselection:`Vector --> Analysis Tools`

Parameters
..........

.. list-table::
   :header-rows: 1
   :widths: 20 20 20 40

   * - Label
     - Name
     - Type
     - Description
   * - **Polygons**
     - ``POLYGONS``
     - [vector: polygon]
     - Polygon layer whose features are associated with the count of
       points they contain
```

(continues on next page)

(continuare din pagina precedentă)

```
   * - **Points**
     - ``POINTS``
     - [vector: point]
     - Point layer with features to count
   * - **Weight field**

       Optional
     - ``WEIGHT``
     - [tablefield: numeric]
     - A field from the point layer.
       The count generated will be the sum of the weight field of the
       points contained by the polygon.
   * - **Class field**

       Optional
     - ``CLASSFIELD``
     - [tablefield: any]
     - Points are classified based on the selected attribute and if
       several points with the same attribute value are within the
       polygon, only one of them is counted.
       The final count of the points in a polygon is, therefore, the
       count of different classes that are found in it.
   * - **Count field name**
     - ``FIELD``
     - [string]

       Default: 'NUMPOINTS'
     - The name of the field to store the count of points
   * - **Count**
     - ``OUTPUT``
     - [vector: polygon]

       Default: [Create temporary layer]
     - Specification of the output layer type (temporary, file,
       GeoPackage or PostGIS table).
       Encoding can also be specified.

Outputs
.......

.. list-table::
   :header-rows: 1
   :widths: 20 20 20 40

   * - Label
     - Name
     - Type
     - Description
   * - **Count**
     - ``OUTPUT``
     - [vector: polygon]
     - Resulting layer with the attribute table containing the
       new column with the points count
```

CAPITOLUL 3

Scrierea codului în Cartea de rețete PyQGIS

If you are planning to add or update some chapters of the PyQGIS-Developer-Cookbook, then you should follow some rules to enable automatic testing of the code snippets.

Testing is really important because it allows automatic checking of the code. Code snippets with errors or code that uses outdated methods will fail and the notification will help you fix the problems.

For testing, we use the Sphinx doctest extension. Refer to the extension documentation for more detailed information.

3.1 How to write testable code snippets

Writing testable code snippets is not so different from the *old* method. Basically, you need to use a different Sphinx *directive*.

3.1.1 Doctest sphinx directives

Instead of embedding the code in a `.. code-block:: python` directive (which would highlight the code syntax automatically), you now need to embed it in a `.. testcode::`. That is, instead of this:

```
.. code-block:: python

   crs = QgsCoordinateReferenceSystem("EPSG:4326")
   assert crs.isValid()
```

You now use this:

```
.. testcode::

   crs = QgsCoordinateReferenceSystem("EPSG:4326")
   assert crs.isValid()
```

After you wrote the example code, you should add some *assertion* that will evaluate the code and that will be run automatically.

In the above example, you are creating a crs and with `assert crs.isValid()` you **test** if it is valid. If the code has a wrong python syntax or the `crs.isValid()` returns `False`, this code snippet will fail during testing.

To successfully run the tests on snippets, you must import all the classes and declare any variables used in the code snippets. You can include those in the code snippet itself (visible in the HTML pages) or you can add them to a `.. testsetup::` directive (hidden in the HTML pages). The `.. testsetup::` needs to be placed before the `.. testcode::`:

```
.. testsetup::

   from qgis.core import QgsCoordinateReferenceSystem

.. testcode::

   crs = QgsCoordinateReferenceSystem("EPSG:4326")
   assert crs.isValid()
```

If the code snippet doesn't create objects (and therefore you cannot use something like `assert object.isValid()`), you can test the code using the `print()` method, then add the expected results within a `.. testoutput::` directive to compare the expected output:

```
.. testcode::

   print("QGIS CRS ID:", crs.srsid())
   print("PostGIS SRID:", crs.postgisSrid())

.. testoutput::

   QGIS CRS ID: 3452
   PostGIS SRID: 4326
```

By default, the content of `.. testoutput::` is shown in the HTML output. To hide it from the HTML use the *:hide:* option:

```
.. testoutput::
   :hide:

   QGIS CRS ID: 3452
   PostGIS SRID: 4326
```

Notă: If the code snippet contains any print statements, you MUST add a `testoutput` with the expected outputs; otherwise the test will fail.

3.1.2 Grouping tests

For each rst document, the code snippets are tested sequentially, which means you can use one `.. testsetup::` for all the following code snippets and that later snippets will have access to variables declared in earlier ones in the document.

Alternatively, you can use groups to break down the examples on the same page in different tests.

You add the code snippet to groups by adding one or more group names (separated by commas) in the respective directive:

```
.. testcode:: crs_crsfromID [, morenames]

   crs = QgsCoordinateReferenceSystem("EPSG:4326")
   assert crs.isValid()
```

The `doctest` will pick each group snippets and run them independently.

Notă: Use group names that make sense with the related content. Use something similar to <chapter>_<subchapter>, for example: crs_intro, crs_fromwkt. In case of failures, this will help identifying where the failures occur.

If you don't declare any group, the code snippet will be added to a group named `default`. If instead, you use `*` as a group name, the snippet will be used in all testing groups, something normally usefull to use in the test setup:

```
.. testsetup:: *

   from qgis.core import QgsCoordinateReferenceSystem
```

3.2 How to test snippets on your local machine

Notă: Instructions are valid for Linux system.

To test Python code snippets, you need a *QGIS* installation. For this, there are many options. You can:

- Use your system *QGIS* installation with *Sphinx* from a Python virtual environment:

  ```
  make -f venv.mk doctest
  ```

- Use a manually built installation of *QGIS*. You'd need to:

  1. Create a custom `Makefile` extension on top of the `venv.mk` file, for example a `user.mk` file with the following content:

     ```
     # Root installation folder
     QGIS_PREFIX_PATH = /home/user/apps/qgis-master

     include venv.mk
     ```

     Or

     ```
     # build output folder
     QGIS_PREFIX_PATH = /home/user/dev/QGIS-build-master/output

     include venv.mk
     ```

  2. Then, use it to run target `doctest`:

```
make -f user.mk doctest
```

- Run target `doctest` inside the official *QGIS* docker image:

  ```
  make -f docker.mk doctest
  ```

  You have to install Docker first because this uses a docker image with QGIS in it.

CAPITOLUL 4

Ghidul traducerilor

Acest manual este dedicat traducătorilor. Mai întâi este explicat procesul general de realizare a unei traduceri. Mai departe, este detaliată traducerea în olandeză a unui document scris, în original, în engleză. La final sunt rezumate *Regulile de traducere*.

Notă: Deși se concentrează pe documentația QGIS, metodele și regulile descrise mai jos se aplică, de asemenea, și traducerii aplicației QGIS și a site-ului web.

4.1 Procesul de traducere

QGIS Documentation is written in English with `.rst` files. In order to provide translations:

1. A prebuild script creates translation files named `.po` files for the English language in the folder `/QGIS-Documentation/locale/en`.
2. The sentences in the `.po` files are pushed to the Transifex web platform, and made available for translators who can begin to translate from English to their language with the editor.
3. When a file is translated at 100%, the translated strings are automatically pulled back to the documentation repository, under `/QGIS-Documentation/locale/<language>`.
4. At the next build of the documentation (which occurs at least once a day – see time at the bottom of the page), a script reuses the sentences to create translated output.
5. For files not fully translated, a script pulls every two weeks translated strings from Transifex to Github and these are as well published at the next build.

6. Whenever an `.rst` file is updated, the English `.po` file is updated and the changes are pushed to the corresponding file in Transifex. This means that when a new paragraph is added to an `.rst` document that was already translated, only the new/updated sentences are added to the translated `.po` file and needs to be translated.

Notă: Translating QGIS Desktop specificities

The main difference with translating QGIS applications is that instead of `.po` files, all the translatable strings in the `.py`, `.cpp`, `.yaml` files that shape a particular version of the application are pushed to and pulled from Transifex as a single `.ts` file (e.g. `qgis-application/qgis_en.ts (branch release-3_30)`). Translations are pulled to Github in development branch (daily), and at release time (for every released versions).

În prezent, sunt folosite două instrumente diferite pentru a efectua traducerile QGIS:

- The Transifex web platform, the easiest and recommended way to translate QGIS, transparently does the process described above and pulls all the translatable texts in one place for the translator. Just pick the files you want and translate. Translated files are stored in the platform until another release is pushed.
- Qt Linguist, a Qt development tool, requires the translator to pull locally the `.po` (or `.ts`) files from the source code, translate and then push back.

Rețineți că indiferent de instrumentul ales, regulile de traducere sunt aceleași.

4.2 Traducerea unui fișier

Pentru a explica cum funcționează traducerea, vom folosi plugin-ul Heatmap ca exemplu. În acest exemplu, îl vom traduce din engleză în olandeză, dar procedeul va fi, practic, similar pentru traducerea în oricare altă limbă.

Sursa unui document se poate găsi aici:

```
QGIS-Documentation/source/docs/user_manual/plugins/plugins_heatmap.rst
```

Deci, de ce am ales acest document?

1. It includes images, captions, headers, references and replacements.
2. Eu l-am scris, deci este mult mai ușor pentru mine să efectuez traducerea ;-)

The build process has created the English `.po` file which can be found here:

```
QGIS-Documentation/locale/en/LC_MESSAGES/docs/user_manual/plugins/plugins_heatmap.
↪po
```

The equivalent Dutch `.po` file (basically a copy) can be found here:

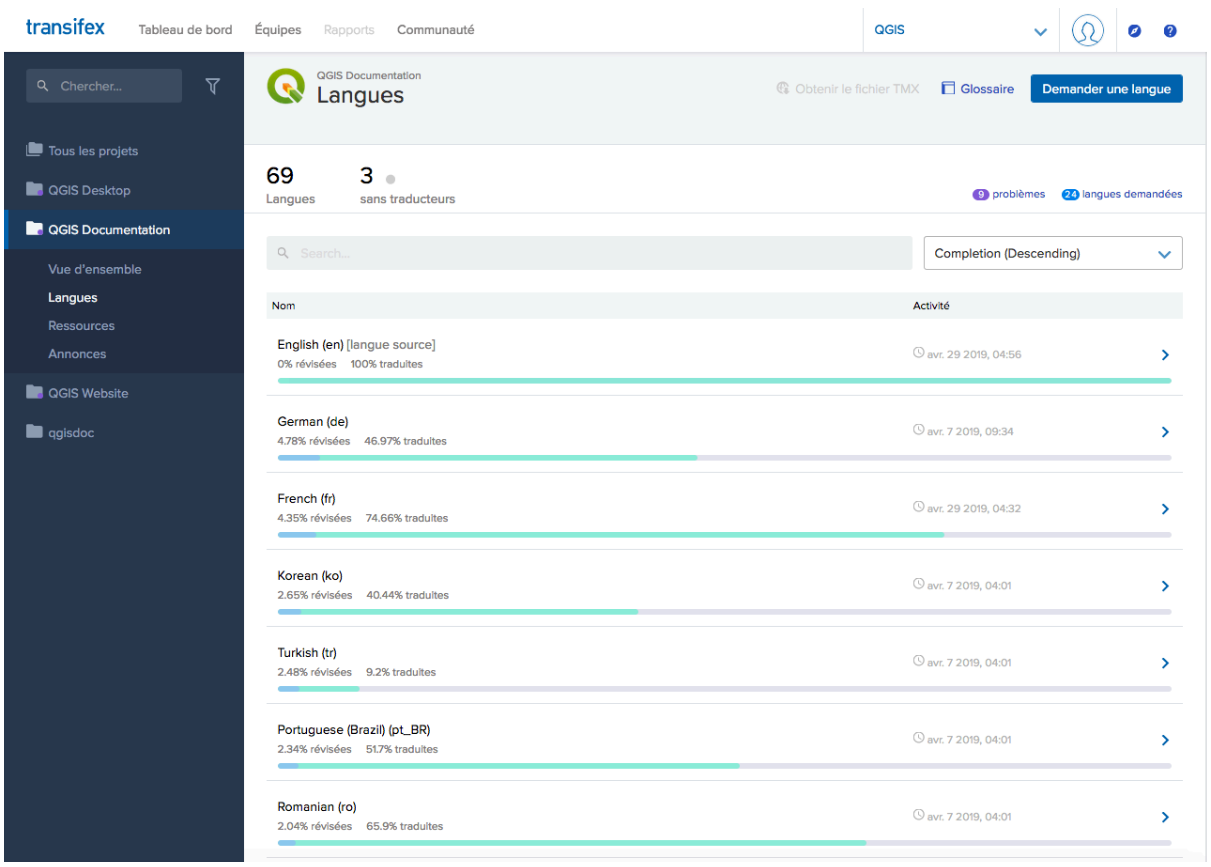
```
QGIS-Documentation/locale/nl/LC_MESSAGES/docs/user_manual/plugins/plugins_heatmap.
↪po
```

Along this file you will see a tiny `.mo` file which indicates that it does not hold any translations yet.

4.2.1 Traducerea pe Transifex

In order to translate using Transifex, you need to:

1. create an account on Transifex and join the QGIS project.
2. Once you are part of a language team, click on the corresponding project (in this case `QGIS Documentation`). A list of available languages with their ratio of translation is displayed.



69 Langues — 3 sans traducteurs

Nom	Activité
English (en) [langue source] — 0% révisées, 100% traduites	avr. 29 2019, 04:56
German (de) — 4.78% révisées, 46.97% traduites	avr. 7 2019, 09:34
French (fr) — 4.35% révisées, 74.66% traduites	avr. 29 2019, 04:32
Korean (ko) — 2.65% révisées, 40.44% traduites	avr. 7 2019, 04:01
Turkish (tr) — 2.48% révisées, 9.2% traduites	avr. 7 2019, 04:01
Portuguese (Brazil) (pt_BR) — 2.34% révisées, 51.7% traduites	avr. 7 2019, 04:01
Romanian (ro) — 2.04% révisées, 65.9% traduites	avr. 7 2019, 04:01

Fig. 4.1: Select language for translation in the Transifex menu

3. Hover over your language and click either:
   - *View resources*: translatable `.po` files with their ratio of translation, number of strings and some more metadata are now displayed.
   - or *Translate*: opens the interface of translation with all the available `.po` files
4. Identify the file you'd like to translate (in our case we are looking for the `docs_user-manual_plugins_plugins-heatmap`, the heatmap plugin file) or any unfinished file and click on it: strings in the files are loaded and you can use the interface to filter, translate, suggest translation…

   Sfat: For the documentation or the website, clicking the `Fix me` link in the footer of a page brings you directly to its corresponding translation page in Transifex.

5. All you need to do is select each text and translate following the *guidelines*.

For further information on the use of Transifex Web Editor, see https://help.transifex.com/en/articles/6318216-translating-with-the-web-editor.

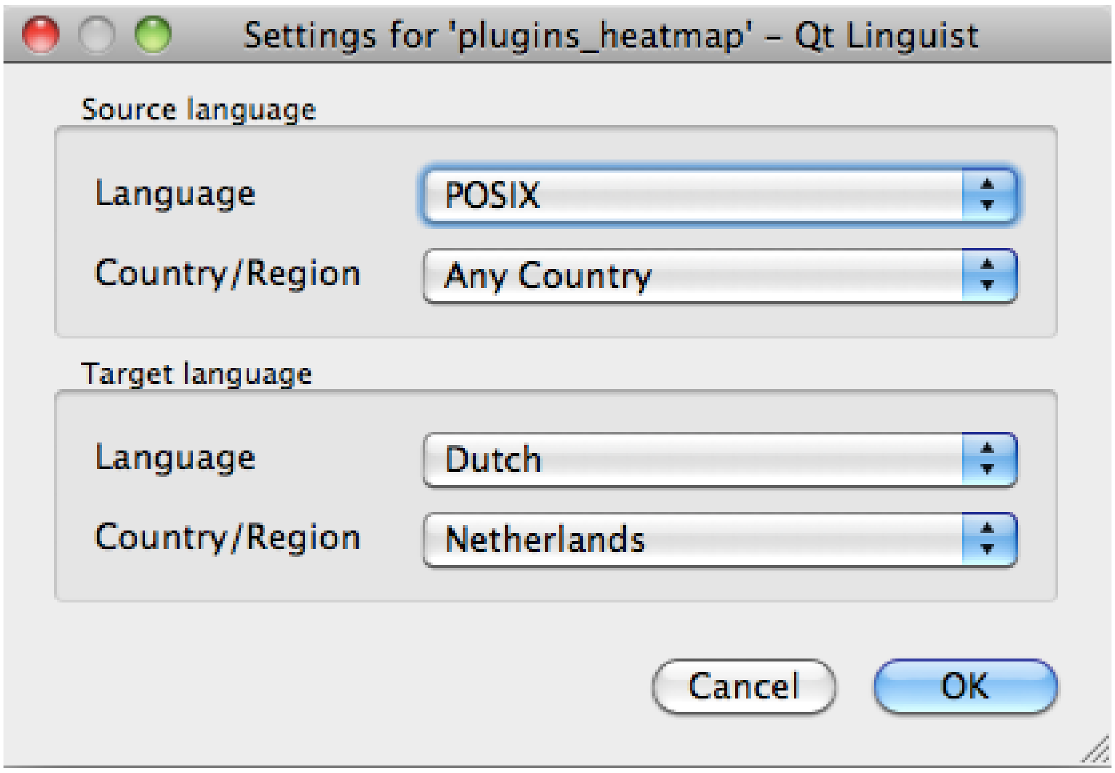
4.2.2 Traducere în Qt Linguist

With Qt Linguist, you need to:

1. manually grab the `.po` or `.ts` file(s). This can be achieved by downloading the file(s) either from Transifex platform or from the `locale/$language` folder of the source repository (in GitHub),
2. proceed to the translation locally
3. upload the modified files to their sources (Transifex or GitHub).

While downloading and uploading translatable files can be done with Transifex, it's not advised to use this process. Since there's no versioning system on Transifex, the file you upload will simply replace the existing one and potentially overwrite any modification made by others on the platform in the meantime.

When you open the file in Qt Linguist for the first time you will see the following dialog:



Settings for 'plugins_heatmap' – Qt Linguist

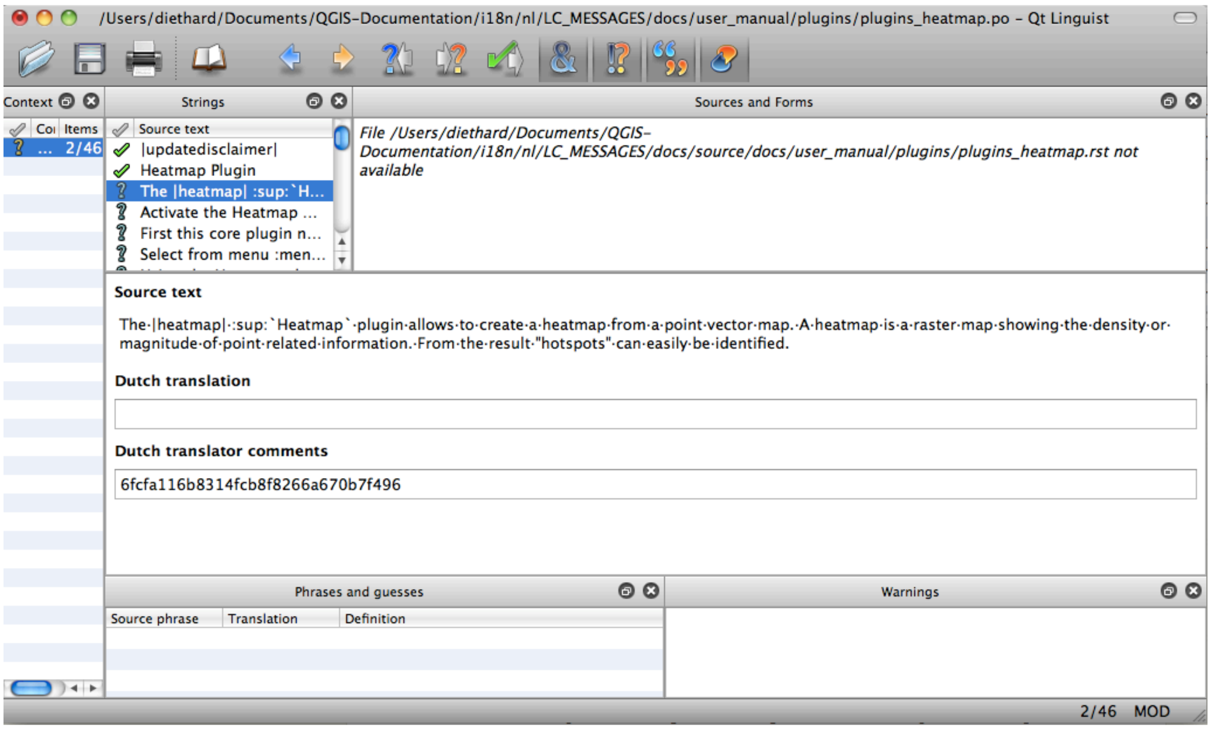
Source language
Language: POSIX
Country/Region: Any Country

Target language
Language: Dutch
Country/Region: Netherlands

Fig. 4.2: Selectați limba pentru traducere în meniul linguist

Limba Țintă trebuie să fie completată corect. Limba Sursă poate fi lăsată așa cum este, cu limba POSIX și Țara/Regiunea pe Orice Țară.

When you press the *OK* button Qt Linguist is filled with sentences and you can start translating, see Fig. 4.3.



/Users/diethard/Documents/QGIS-Documentation/i18n/nl/LC_MESSAGES/docs/user_manual/plugins/plugins_heatmap.po – Qt Linguist

File /Users/diethard/Documents/QGIS-Documentation/i18n/nl/LC_MESSAGES/docs/source/docs/user_manual/plugins/plugins_heatmap.rst not available

Source text

The·|heatmap|·:sup:`Heatmap`·plugin·allows·to·create·a·heatmap·from·a·point·vector·map.·A·heatmap·is·a·raster·map·showing·the·density·or·magnitude·of·point·related·information.·From·the·result·"hotspots"·can·easily·be·identified.

Dutch translation

Dutch translator comments

6fcfa116b8314fcb8f8266a670b7f496

Fig. 4.3: Traducerea cu ajutorul meniului Linguist

În meniu veți vedea următoarele butoane, ușor de utilizat.

- Butonul Translation Done Next este cel mai important. Dacă elementul are nevoie de traducere, introduceți o traducere în câmpul de text, apoi apăsați acest buton. Dacă elementul nu necesită traducere, lăsați gol câmpul de text și apăsați, de asemenea, acest buton, care indică faptul că elementul este în regulă, continuându-se cu următorul.
- Butonul Goto Previous, poate fi folosit pentru a merge la elementul tradus anterior.
- Butonul Goto Next, poate fi folosit pentru a merge la următorul element de tradus.
- Butonul Next Todo, efectuează saltul la prima traducere care necesită o traducere. Este foarte util atunci când documentul original s-a schimbat și numai câteva expresii noi/modificate trebuie să fie traduse.
- Butonul Previous Todo, caută înapoi și sare la primul element de traducere care necesită o traducere.

For further information on the use of Qt Linguist, see https://doc.qt.io/qt-5/linguist-translators.html

Atenționare: If you want to download content to translate from the source repository, never do this in the `master` branch. For translations there are always translation branches available, once a document is fully updated in English for a certain version. As an example, to translate the manual of QGIS 2.8, you have to use the manual_en_v2.8 branch.

4.2.3 Traduceți un manual

Acum vom începe traducerea manualului pentru pluginul_heatmap!

Traducerea celor mai multe propoziții ar trebui să fie simplă. Pe durata acestei sesiuni de traducere, vom sublinia părțile (expresiile rst) care necesită traducere.

Mai jos, vedem o propoziție de tradus, mai interesantă:

```
The |heatmap| :sup:`Heatmap` plugin allows to create a heatmap from a
point vector map. A heatmap is a raster map showing the density or
magnitude of point related information. From the result "hotspots" can
easily be identified.
```

Această frază conține două declarații rst:

1. `|heatmap|` cuvintele dintre `|` sunt înlocuiri și nu ar trebui să fie traduse niciodată! Acesta, de exemplu, va fi înlocuit de pictograma plugin-ului Heatmap!
2. `:sup:`Heatmap``, expresia `:sup:` este o declarație de superpoziție, care imprimă textul următor, un pic mărit. Se folosește pentru a prezenta textele într-un balon, care apare la trecerea pe deasupra elementului din bara de instrumente, el putând arăta în mod diferit dacă a fost tradus în aplicația QGIS. În cazul limbii olandaeze, nu a fost tradus!

Toate celelalte texte simple din această propoziție pot fi traduse!

Următorul element de tradus conține expresia `:ref:`, care este frecvent folosită pentru a face trimitere la o altă secțiune a manualului respectiv! Textul urmat de expresia `:ref:` nu ar trebui să fie schimbat, pentru că acesta reprezintă un identificator unic!

```
First this core plugin needs to be activated using the Plugin Manager
(see Section :ref:`load_core_plugin`). After activation the heatmap icon
|heatmap| can be found in the Raster Toolbar.
```

In this case `load_core_plugin` is a unique reference identifier placed before an rst item that has a caption. The ref statement will be replaced with the text of the header and turned into a hyperlink. When the header this reference is referring to is translated, all references to this header will be automatically translated as well.

Articolul următor conține eticheta rst `:menuselection:` urmat de textul afișat, în mod curent, într-un meniu din aplicația QGIS, acesta putând fi tradus în cerere și, prin urmare, ar trebui să fie schimbat atunci când este cazul.

```
Select from menu :menuselection:`View --> Toolbars --> Raster` to activate
the Raster Toolbar when it is not yet activated.
```

La punctul de mai sus „View –>"este, de fapt, tradus ca „Beeld –>", aceasta fiind traducerea olandeză utilizată din aplicația QGIS.

Un pic mai departe vom întâlni următoarele elemente greu de tradus:

```
The |heatmap| :sup:`Heatmap` tool button starts the Dialog of the Heatmap
plugin (see :numref:`figure_heatmap_settings`).
```

It holds a reference to a figure `figure_heatmap_settings_`, and like a reference to a section this reference should not be changed!! The reference definition from the rst-document is not included in the `.po` file and can therefore not be changed. This means the reference to figures can not be translated. When HTML is created you will see `figure_heatmap_settings`. When a PDF document is created `figure_heatmap_settings_` is replaced with a figure number.

Următorul element de traducere cu atribute rst este următorul:

```
**Input Point dialog**: Provides a selection of loaded point vector maps.
```

Nu eliminați steluțele din linia de mai sus. Textul conținut între ele va fi transcris îngroșat. Adesea, textul este inclus chiar în dialog și poate fi, la fel de bine, tradus în aplicație.

Următoarea traducere conține eticheta rst `:guilabel:`.

```
When the |checkbox| :guilabel:`Advanced` checkbox is checked it will
give access to additional advanced options.
```

Textul `Avansat` al etichetei poate fi tradus din aplicația QGIS și, probabil, chiar trebuie tradus!

Următorul element de tradus conține ``airports``. Apostrofurile sunt folosite pentru a atribui textului un alt font. În acest caz, este vorba de o valoare literală și nu are nevoie de traducere.

```
For the following example, we will use the ``airports`` vector point
layer from the QGIS sample dataset (see :ref:`label_sampledata`).
Another excellent QGIS tutorial on making heatmaps can be found on
`https://www.qgistutorials.com
<https://www.qgistutorials.com/en/docs/creating_heatmaps.html>`_.
```

This item also includes a hyperlink with an url and an external presentation. The url should of course be left intact, you are allowed to change the external text `https://www.qgistutorials.com` which is visible by the reader. Never remove the underscore at the end of the hyperlink which forms an essential part of it!!

4.2.4 Sumarul regulior pentru traducere

1. Nu schimbați textul dintre două caractere `|`, cum ar fi `|bronze|`, `|checkbox|`, `|labels|`, `|selectString|`, `|addLayer|` … Acestea sunt etichete speciale, folosite pentru a substitui imaginile
2. Do not change references that start with roles like `:ref:`, `:file:`, `:numref:` unless they include a title. In that case, you can translate the title but keep unchanged the link (i.e., the text between < and >)

   Sfat: When a title is provided for a reference, Transifex may display a number in the English source text in replacement of the link part. Click on the number in the source text to add the reference link next to the title being translated.

3. Nu schimbați referințe care se termină cu liniuțe de subliniere ca `figure_labels_1_`
4. Nu modificați adresa hiperlegăturilor, însă puteți modifica descrierea lor externă. Lăsați neatinsă liniuța de subliniere de la sfârșit, neintroducând spațieri adiționale (>`_)
5. Schimbați textul dintre ghilimele, care urmează după etichetele `:sup:`, `:guilabel:` și `:menuselection:`. Verificați dacă/cum sunt traduse în aplicația QGIS. Nu schimbați denumirea etichetelor.
6. Textul inclus între steluțe duble și apostrofuri duble indică adesea valori sau nume de câmpuri, uneori necesitând traducere, alteori nu.
7. Aveți grijă să utilizați exact același (număr de) caractere speciale, cum ar fi `` ` ``, ` `` `, `*`, `**`, `::`, similar textului sursă, . Acestea contribuie la cosmetizarea informațiilor furnizate
8. Nu puneți un spațiu, înainte sau după textul inclus între caractere speciale sau din cadrul etichetelor
9. Nu încheiați șirurile traduse cu un nou paragraf, în caz contrar, textul nu va fi tradus pe durata generării codului html.

Respectați regulile de mai sus iar documentul tradus va arăta foarte bine!

Pentru orice întrebare, vă rugăm să contactați Echipa Comunității QGIS sau Echipa de Traducere QGIS.

CAPITOLUL 5

Substituții

5.1 Folosire

Pentru a ușura utilizarea pictogramelor în manualele QGIS, substituțiile sunt definite pentru fiecare pictogramă din fișierul `/source/substitutions.txt` al `Depozitului Documentației QGIS <https://github.com/qgis/QGIS-Documentation>`_ , unele dintre acestea fiind enumerate mai jos. Astfel, atunci când doriți să utilizați o pictogramă din aplicația QGIS în documentație, există șanse mari să existe deja o înlocuire care poate/ar trebui să fie utilizată.

În cazul în care nu există nici o substituție:

1. verificați în depozitul documentației dacă pictograma este disponibilă în folderul `/static/common`. Dacă nu există nicio imagine, atunci trebuie să găsiți și să copiați fișierul imagine al pictogramei în depozitul QGIS (adesea în folderul temelor implicite) și să-l adăugați (în format `.png`) în folderul `/static/common`. Pentru păstrarea convențiilor și pentru a ușura actualizările, atunci când este posibil, se recomandă să păstrați numele fișierului.

2. creați referința către substituție în fișierul `/substitutions.txt` urmând exemplul de mai jos. Textul de înlocuire ar trebui să fie derivat din numele fișierului și să fie scris în camelCase:

```
.. |dataSourceManager| image:: /static/common/mActionDataSourceManager.png
   :width: 1.5em
.. |splitLayer| image:: /static/common/split_layer.png
   :width: 1.5em
```

3. Actualizați secțiun(ile) țintă din documente, folosind noua substituție.

4. (opțional, dar ar fi bine să nu lipsească) adăugați substituția în lista de mai jos.

5. Adăugați noua referință pentru substituție în lista de înlocuiri, la sfârșitul fișier(elor) în care se utilizează, sau pur și simplu rulați scriptul `scripts/find_set_subst.py`.

```
# from the repository main folder
python3 scripts/find_set_subst.py
```

5.2 Substituții comune

Mai jos sunt prezentate unele pictograme și substituțiile lor, pentru a fi utilizate la scrierea documentației. Pot fi folosite/găsite în multe locuri din manuale.

5.2.1 Pictogramele platformelor

Pictogramă	Substituție	Pictogramă	Substituție
	`|logo|`		
	`|kde|`		`|nix|`
	`|osx|`		`|win|`

5.2.2 Elemente de meniu

Pictogramă	Substituție	Pictogramă	Substituție
☑	|checkbox|	☐	|unchecked|
●	|radioButtonOn	○	|radioButtonOff|
1,00	|selectNumber|	... ▾	|selectString|
■ ▾	|selectColor|	▾	|selectColorRamp|
	|tab|	°	|degrees|
Display name lakes.shp	|inputText|		|slider|
☰	|hamburgerMenu		

5.3 Butoanele Barelor de Instrumente

5.3.1 Gestiune Straturi și vederi de ansamblu

Pictogramă	Substituție	Pictogramă	Substituție
	|dataSourceManager|		
	|addOgrLayer|		
	|addRasterLayer|		|addMssqlLayer|
	|addDelimitedTextLayer|		|addSpatiaLiteLayer|
	|addPostgisLayer|		|addOracleLayer|
	|addAfsLayer|		|addMeshLayer|
	|addVectorTileLayer|		|addXyzLayer|
	|addVirtualLayer|		|addWmsLayer|
	|addWcsLayer|		|addWfsLayer|
	|addPointCloudLayer|		|addGpsLayer|
	|addTiledSceneLayer|		|addHanaLayer|
	|newVectorLayer|		|newSpatiaLiteLayer|
	|newGeoPackageLayer|		|createMemory|
	|newVirtualLayer|		|newMeshLayer|
	|newGpx|		
	|dbManager|	GDAL	|gdal|
	|geoPackage|		|spatialite|
	|virtualLayer|		|wms|
	|wcs|		|wfs|
	|pointCloudLayer|		|gps|
	|tiledSceneLayer|		|hana|

continues on next page

Tabelul 5.1 – continuare din pagina precedentă

Pictogramă	Substituție	Pictogramă	Substituție
	`|dbSchema|`		
	`|inOverview|`		`|addAllToOverview|`
	`|removeAllFromOverview|`		`|removeLayer|`
	`|showAllLayers|`		`|hideAllLayers|`
	`|showPresets|`		`|showSelectedLayers|`
	`|hideSelectedLayers|`		`|hideDeselectedLayers|`
	`|toggleAllLayers|`		`|toggleSelectedLayers|`
	`|addLayer|`		
	`|indicatorTemporal|`		`|indicatorNonRemovable|`
	`|indicatorEmbedded|`		`|indicatorFilter|`
	`|indicatorMemory|`		`|indicatorNoCRS|`
	`|indicatorBadLayer|`		`|favourites|`
	`|indicatorLayerError|`		`|indicatorNotes|`
	`|indicatorLowAccuracy|`		`|indicatorOffline|`

5.3.2 Proiect

Pictogramă	Substituție	Pictogramă	Substituție
	`|fileNew|`		`|fileOpen|`
	`|fileSave|`		`|fileSaveAs|`
	`|fileExit|`		`|user|`

5.3.3 Editare

Pictogramă	Substituție	Pictogramă	Substituție
	|undo|		|redo|
	|editCopy|		|editPaste|
	|editCut|		|saveEdits|
	|editableEdits|		
	|circle2Points|		|circle2TangentsPoint|
	|circle3Points|		|circle3Tangents|
	|circleCenterPoint|		|ellipseCenter2Points|
	|ellipseCenterPoint|		|ellipseExtent|
	|ellipseFoci|		|rectangle3PointsDistance|
	|rectangle3PointsProjected|		|rectangleCenter|
	|rectangleExtent|		|regularPolygon2Points|
	|regularPolygonCenterCorner		|regularPolygonCenterPoint|

5.3.4 Rezultatul identificării

Pictogramă	Substituție	Pictogramă	Substituție
	|expandTree|		|collapseTree|
	|expandNewTree|		|formView|
	|deselectAll|		|editCopy|
	|filePrint|		
	|identifyByRectangle|		|identifyByPolygon|
	|identifyByFreehand|		|identifyByRadius|

5.3.5 Digitizare Simplă și Avansată

Pictogramă	Substituție	Pictogramă	Substituție
	|cad|		|cadConstruction|
	|cadParallel|		|cadPerpendicular|
	|floater|		
	|toggleEditing|		|allEdits|
	|tracing|		|snapping|
	|snappingVertex|		|snappingSegment|
	|snappingArea|		|snappingCentroid|

continues on next page

Tabelul 5.2 – continuare din pagina precedentă

Pictogramă	Substituție	Pictogramă	Substituție
	|snappingMiddle|		|snappingEndpoint|
	|capturePoint|		|capturePolygon|
	|captureLine|		|captureCurveFromFeature|
	|deleteSelectedFeatures|		
	|circularStringCurvePoint|		|circularStringRadius|
	|vertexTool|		|vertexToolActiveLayer|
	|digitizeWithSegment|		|digitizeShape|
	|streamingDigitize|		|digitizeWithCurve|
	|moveFeature|		|moveFeatureCopy|
	|moveFeatureLine|		|moveFeatureCopyLine|
	|moveFeaturePoint|		|moveFeatureCopyPoint|
	|rotateFeature|		|rotatePointSymbols|
	|scaleFeature|		
	|offsetCurve|		|offsetPointSymbols|
	|simplify|		|reshape|
	|addRing|		|addPart|
	|fillRing|		
	|deleteRing|		|deletePart|
	|mergeFeatures|		|mergeFeatureAttributes|
	|splitFeatures|		|splitParts|
	|reverseLine|		
	|allowIntersections|		|avoidIntersectionsCurrentLayer|
	|avoidIntersectionsLayers|		|snappingSelf|

5.3.6 Suprafață poliedrică

Pictogramă	Substituție	Pictogramă	Substituție
	|meshDigitizing|		|meshReindex|
	|meshSelectExpression|		|meshSelectPolygon|
	|meshTransformByExpression		|meshEditForceByVectorLines|
	|vertexCoordinates|		

5.3.7 Explorarea hărții și a atributelor

Pictogramă	Substituție	Pictogramă	Substituție
	|pan|		|panToSelected|
	|zoomIn|		|zoomOut|
	|zoomActual|		|zoomFullExtent|
	|zoomToLayer|		|zoomToSelected|
	|zoomLast|		|zoomNext|
	|zoomInXAxis|		|refresh|
	|identify|		|mapTips|
	|showBookmarks|		|newBookmark|
	|measure|		|measureArea|
	|measureBearing|		|measureAngle|
	|newMap|		|new3DMap|
	|tiltUp|		|tiltDown|
	|3dNavigation|		|play|
	|temporal		|temporalNavigationOff|
	|temporalNavigationFixedRang		|temporalNavigationAnimated|
	|newElevationProfile|		

5.3.8 Selecția și Expresiile

Pictogramă	Substituție	Pictogramă	Substituție
	|selectRectangle|		|selectPolygon|
	|selectFreehand|		|selectRadius|
	|selectAll|		|deselectAll|
	|invertSelection|		|expressionSelect|
	|deselectActiveLayer|		
	|selectDistance|		|selectLocation|
	|selectAllTree|		|select|
	|selectAdd|		|selectRemove|
	|formSelect|		|dataDefine|
	|expression|		|dataDefineOn|
	|dataDefineExpressionOn|		|dataDefineError|
	|dataDefineExpressionError|		
	|addExpression|		
	|expressionFilter|		|filterMap|

5.3.9 Etichete și Diagrame

Pictogramă	Substituție	Pictogramă	Substituție
	|labelingSingle|		|labelingNone|
	|labelingRuleBased|		|labelingObstacle|
	|piechart|		|diagramNone|
abc	|text|		|histogram|
	|stackedBar|		
	|createAnnotationLayer|		|annotationLayer|
	|textAnnotation|		|svgAnnotation|
	|formAnnotation|		|htmlAnnotation|
	|actionText|		|textAlongLine|
	|labelbackground|		|labelbuffer|
	|labelformatting|		|labelplacement|
	|labelshadow|		|render|
	|labelcallout|		
	|labelAnchorCenter|		|labelAnchorCustom|
	|labelAnchorEnd|		|labelAnchorStart|
	|pinLabels|		|showHideLabels|
	|moveLabel|		|rotateLabel|
	|showPinnedLabels|		|showUnplacedLabel|
	|changeLabelProperties|		|autoPlacementSettings|

5.3.10 Decorațiuni

Pictogramă	Substituție	Pictogramă	Substituție
	|copyrightLabel|		|addGrid|
	|titleLabel|		|northArrow|
	|scaleBar|		|addMap|
	|addImage|		

5.3.11 Ajutorul

Pictogramă	Substituție	Pictogramă	Substituție
?	|helpContents|	🏠	|qgisHomePage|
✓	|success|		
●	|helpSponsors|	?	|contextHelp|

5.3.12 Culori

Pictogramă	Substituție	Pictogramă	Substituție
▨	|colorBox|		|colorPicker|
▦	|colorSwatches|	●	|colorWheel|

5.4 Alte pictograme de bază

Pictogramă	Substituție	Pictogramă	Substituție
◀	|arrowLeft|	▶	|arrowRight|
▼	|arrowDown|	▲	|arrowUp|
✚	|symbologyAdd|	▬	|symbologyRemove|
✎	|projectProperties|	🔧	|options|
	|interfaceCustomization|		|keyboardShortcuts|
©	|copyrightLabel|		|northArrow|
	|scaleBar|		|tracking|
📊	|gpsTrackBarChart|		
	|gpsConnect|		|gpsDisconnect|
⌖	|gpsDestinationLayer|		|addTrackPoint|
⌖	|recenter|		|reset|
	|folder|		|extents|
⚙	|settings|	▶	|start|
🔧	|properties|	🗑	|deleteSelected|
▶	|browserExpand|	▼	|browserCollapse|
	|codeEditor|	✚	|add|
	|relations|		|layoutItem3DMap|
⏱	|stopwatch|	((●))	|sensor|
✖	|clearItem|		

5.5 Tabela de Atribute

Pictogramă	Substituție	Pictogramă	Substituție
	|openTable|		|openTableSelected|
	|openTableVisible|		|openTableEdited|
	|selectedToTop|		
	|selectAll|		|invertSelection|
	|panToSelected|		|zoomToSelected|
	|copySelected|		|editPaste|
	|expressionSelect|		|deleteSelectedFeatures|
	|newAttribute|		|deleteAttribute|
	|editTable|		
	|newTableRow|		|calculateField|
	|refresh|		|formView|
	|conditionalFormatting|		|multiEdit|
	|dock|		|actionRun|
	|duplicateFeature|		|zoomTo|
	|panTo|		|highlightFeature|
★	|handleStoreFilterExpressionChecke		
☆	|handleStoreFilterExpressionUnchec		

5.6 Proiecții și Georeferențiere

Pictogramă	Substituție	Pictogramă	Substituție
	|geographic|		|crs|
	|customProjection|		|setProjection|
	|projectionDisabled|		|projectionEnabled|
	|transformation|		|gdalScript|
	|georefRun|		|pencil|
	|linkQGisToGeoref|		|linkGeorefToQGis|
	|fullHistogramStretch|		

5.7 Aspect Pagină Imprimată

Pictogramă	Substituție	Pictogramă	Substituție
	`|newLayout|`		`|layoutManager|`
	`|duplicateLayout|`		
	`|newReport|`		`|newPage|`
	`|atlasSettings|`		`|atlas|`
	`|filePrint|`		`|saveMapAsImage|`
	`|saveAsSVG|`		`|saveAsPDF|`
	`|addBasicShape|`		`|addBasicCircle|`
	`|addBasicTriangle|`		`|addBasicRectangle|`
	`|addNodesShape|`		`|editNodesShape|`
	`|addPolygon|`		`|addPolyline|`
	`|addArrow|`		`|northArrow|`
	`|add3DMap|`		`|addMap|`
	`|elevationProfile|`		`|copyProfileSettings|`
	`|addLegend|`		`|addHtml|`
	`|addManualTable|`		`|addTable|`
	`|addImage|`		`|addMarker|`
	`|label|`		`|scaleBar|`
	`|select|`		`|moveItemContent|`
	`|setToCanvasScale|`		`|setToCanvasExtent|`
	`|viewScaleInCanvas|`		`|viewExtentInCanvas|`
	`|raiseItems|`		`|lowerItems|`
	`|moveItemsToTop|`		`|moveItemsToBottom|`
	`|alignLeft|`		`|alignRight|`
	`|alignHCenter|`		`|alignVCenter|`
	`|alignTop|`		`|alignBottom|`
	`|distributeLeft|`		`|distributeRight|`
	`|distributeTop|`		`|distributeBottom|`
	`|distributeHCenter|`		`|distributeVCenter|`
	`|distributeHSpace|`		`|distributeVSpace|`
	`|resizeShortest|`		`|resizeTallest|`
	`|resizeNarrowest|`		`|resizeWidest|`
	`|resizeSquare|`		`|groupItems|`
	`|lockItems|`		`|unlockAll|`

continues on next page

Tabelul 5.3 – continuare din pagina precedentă

Pictogramă	Substituție	Pictogramă	Substituție
🔒	|locked|	🔓	|unlocked|
🔒	|lockRepeating|	🔒	|lockedGray|

5.8 Proprietăţile Stratului

Pictogramă	Substituție	Pictogramă	Substituție
	|symbology|		|labelingSingle|
	|sourceFields|	⚒	|general|
ℹ	|metadata|	⚙	|action|
💬	|display|		|rendering|
◀	|join|	📊	|diagram|
	|labelmask|	🕒	|temporal|
	|legend|		|dependencies|
	|3d|		|system|
	|elevationscale|		|layerTree|
	|editMetadata|		|overlay|
	|digitizing|		|auxiliaryStorage|
	|history|		|stylePreset|
🔍	|search|		|pyramids|
	|transparency|		|rasterHistogram|
	|singleSymbol|		|nullSymbol|
	|graduatedSymbol|		|categorizedSymbol|
	|25dSymbol|		|ruleBasedSymbol|
	|invertedSymbol|	●	|heatmapSymbol|
◎	|pointDisplacementSymbol|		|pointClusterSymbol|
	|mergedFeatures|		
	|meshcontours|		|meshcontoursoff|
↘	|meshvectors|	↘	|meshvectorsoff|
▦	|meshframe|		|meshaveraging|
🟩	|singleColor|		|paletted|
	|singlebandPseudocolor|		|multibandColor|
⬚	|pointCloudExtent|		
Σ	|sum|		|sort|

continues on next page

Tabelul 5.4 – continuare din pagina precedentă

Pictogramă	Substituție	Pictogramă	Substituție
⭐	|paintEffects|		|mapIdentification|
	|styleManager|		|iconView|
	|joinNotEditable|		|joinedLayerNotEditable|
	|joinHasNotUpsertOnEdit|		|filterTableFields|
	|symbologyEdit|		
	|sharingImport|		|sharingExport|

5.9 Plugin-uri

5.9.1 Procesare

Pictogramă	Substituție	Pictogramă	Substituție
	|processingAlgorithm|		|processingModel|
	|processingHistory|		|processingResult|
	|menu|		
	|processSelected|		|editHelpContent|
	|saveAsPython|		|modelOutput|
	|qgsProjectFile|		|addToProject|
123	|fieldInteger|		
	|meanCoordinates|		|extractLayerExtent|
	|selectRandom|		|vectorGrid|
	|convexHull|		|buffer|
	|intersect|		|union|
	|symmetricalDifference|		|clip|
	|difference|		|dissolve|
✓	|checkGeometry|		|exportGeometry|
	|delaunay|		|centroids|
	|polygonToLine|		|extractVertices|
	|lineToPolygon|		|nearestNeighbour|
	|splitLayer|		|heatmap|
	|showRasterCalculator|		|showMeshCalculator|
	|regularPoints|		|addGeometryAttributes|
Σ	|basicStatistics|		|uniqueValues|
	|collect|		|simplify_2|
	|createGrid|		|distanceMatrix|

continues on next page

Tabelul 5.5 – continuare din pagina precedentă

Pictogramă	Substituție	Pictogramă	Substituție
	|lineIntersections|		|mergeLayers|
	|sumPoints|		|sumLengthLines|
	|randomPointsInPolygons|		|randomPointsWithinPolygon|
	|randomPointsOnLines|		|randomPointsWithinExtent|
	|multiToSingle|		
	|grid|		|tiles|
	|merge|		|rasterClip|
	|contour|		|proximity|
	|polygonize|		|rasterize|
	|sieve|		|nearblack|
	|projectionAdd|		|projectionExport
	|8To24Bits|		|24To8Bits|
	|rasterInfo|		|rasterOverview|
	|vrt|		|voronoi|
	|translate|		|warp|
	|iterate|		|terminal|

5.9.2 Diverse Plugin-uri de Bază

Standard, cu instalare minimală, dar neîncărcate la instalarea inițială

Pictogramă	Substituție	Pictogramă	Substituție
	|showPluginManager|		|installPluginFromZip|
	|pythonFile|		|runConsole|
	|showEditorConsole|		|clearConsole|
	|offlineEditingCopy|		|offlineEditingSync|
	|plugin|		|metasearch|
	|geometryChecker|		|topologyChecker|
	|fromSelectedFeature|		|sqlQueryBuilder|

5.9.3 Integrare GRASS

Pictogramă	Substituție	Pictogramă	Substituție
	`|grassLogo|`		`|grassRegion|`
	`|grassTools|`		`|grassNewMapset|`
	`|grassOpenMapset|`		`|grassCloseMapset|`