GitHub

Projekt: DigitalSchoolNotes

Projekt Team: Adler, Brinnich, Hohenwarter, Karic, Stedronsky

Version 2.0 **16.12.2015** **Status: [RELEASE]**

	Datum	Name	Unterschrift
Erstellt	07.10.2015	Selina Brinnich	
Geprüft	16.12.2015	Niklas Hohenwarter	
Freigegeben			
Git-Pfad: */doc/technologien*		**Dokument:** *github_technologie.doc*	

Inhaltsverzeichnis

1 Changelog

Version	Datum	Status	Bearbeiter	Kommentar
0.1	2015-10-07	Erstellt	Selina Brinnich	Dokument erstellt
1.0	2015-10-07	Geprüft	Niklas Hohenwarter	QA
1.1	2015-12-15	Bearbeitet	Adin Karic	Abbildungsverzeichnis, Codeverzeichnis, Code-Korrekturen, Formatierung
2.0	2015-12-16	Geprüft	Niklas Hohenwarter	QA

2 Was ist GitHub

GitHub.com [1] ist ein Hosting-Dienst für Software-Entwicklungsprojekte, welches das Versionsverwaltungs-System Git verwendet. Git ist eine freie Software zur verteilten Versionsverwaltung von Dateien. Alternativ zu GitHub gibt es GitLab, das jedoch für lokale Netzwerke angewendet wird. Hauptaufgabe ist das Versionieren von Software, auch wenn mehrere Personen darauf zugreifen, somit kann ein Projektteam gleichzeitig und produktiv an der Software arbeiten.

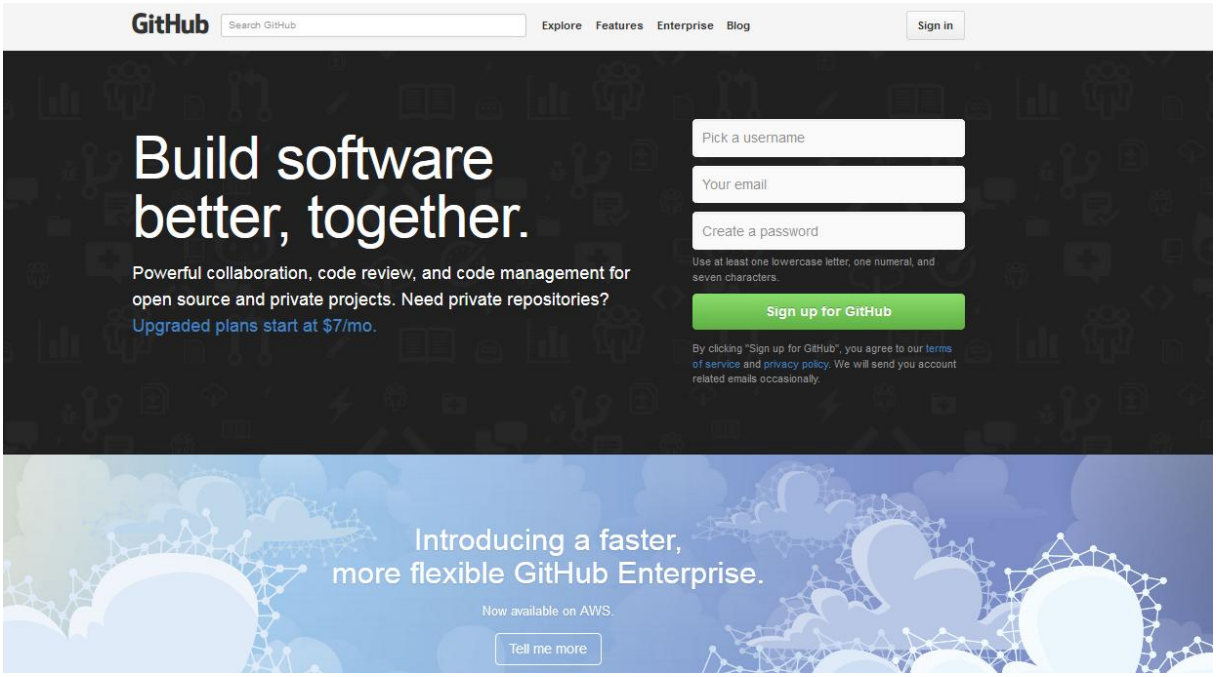

Abbildung 1 github.com Registrierung

3 Collaborators

3.1 Was ist ein Collaborator

Ein Collaborator ist eine Person, die Lese und Schreib-Rechte für ein Repository besitzt, das jemand anderem gehört. Man kann eine Person als Collaborator zu seinem eigenen Repository hinzufügen, indem man diese Person unter den Einstellungen des Repositories, unter dem Reiter „Collaborators“, den Benutzernamen der Person angibt.

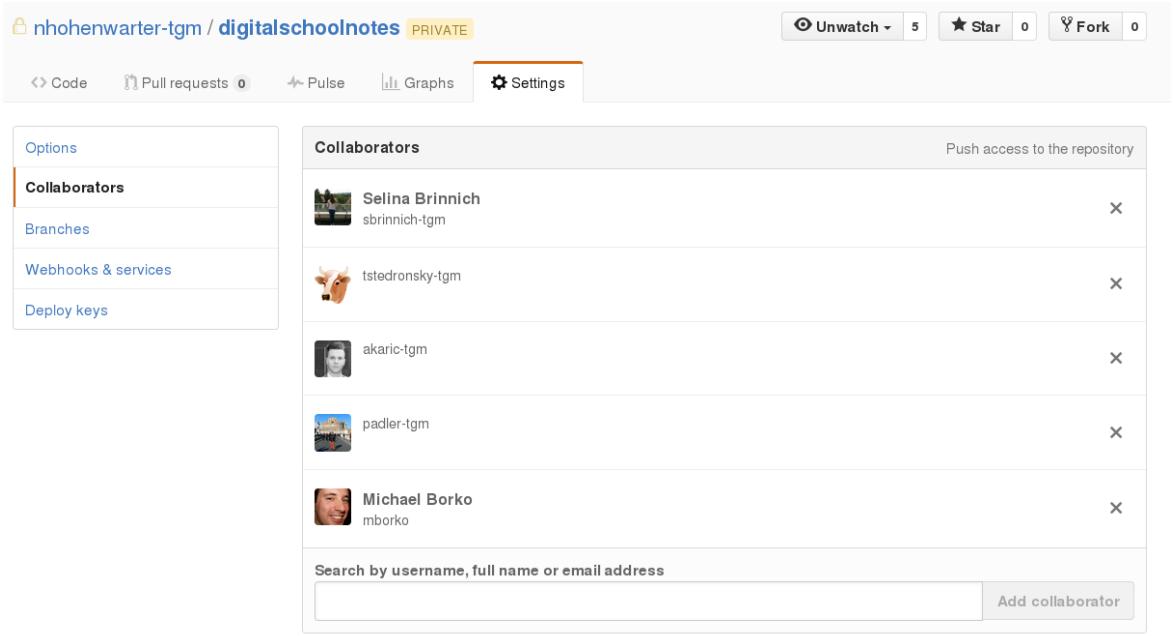

Abbildung 2 github Collaborator Ansicht

4 Git Commands

Mithilfe des Programmes „Git-Bash“ unter Windows bzw. direkt auf der Konsole unter Linux kann man mit einfachen Befehlen seine Software versionieren und das Repository verwalten.

4.1 Befehl: „clone“

Wenn man lokal ein Repository haben möchte, verwendet man den Befehl `git clone <URL>`. Zu finden ist die einzufügende URL im richtigen Repository, rechts.

Abbildung 3 https clone URL

Dafür gibt es 3 Arten:

- HTTPS
- SSH
- Subversion

HTTPS ist die einfachste Möglichkeit. Einfach die URL kopieren und mit `git clone <URL>` herunterladen. Dies funktioniert überall, auch wenn man hinter einer Firewall ist. Bei Befehlen wie `git pull` oder `git push` im Repository wird zur Überprüfung der Identität Username/Email und Passwort gefragt. [2]

SSH kann verwendet werden, wenn der SSH-Key des Computers in Github unter "Einstellungen -> SSH" ebenso angeführt ist. Entweder erstellt man sich einen SSH-Key (siehe Github-Hilfe) oder verwendet einen bereits vorhandenen. [2,3]

Ebenso kann man einen Superversion Client benutzen um lokal Zugang zum Repository zu erhalten. Es beinhaltet aber auch viele andere Funktionen. [2]

Wenn man sich für eines entschieden hat, kann man den Befehl in der Commandline eingeben. Bsp:

```
$ git clone https://github.com/mgoebel-tgm/Bring-Buy.git
```

Code 1 git clone

4.2 Befehl: „status“

Mit dem Befehl `git status` kann man sehen, welche Dokumente des Repositories bearbeitet, verändert oder gelöscht wurden.

```
$ git status
On branch development
Your branch is up-to-date with 'origin/development'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        github_technologie.docx
        ~$thub_technologie.docx
        ~WRL0162.tmp
        ../richtlinien/~$orlage.dotx
```

Abbildung 4 git status

4.3 Befehl: „add“

Der Befehl `git add` fügt ein Dokument zu den Dokumenten hinzu, die committed werden.

```
$ git add github_technologie.docx
```

Code 2 git add

```
$ git status
On branch development
Your branch is up-to-date with 'origin/development'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   github_technologie.docx
```

Abbildung 5 git status nach git add

4.4 Befehl: „commit“

Mit dem `git commit -m` Befehl fügt man zu den hochzuladenden Dokumenten eine Nachricht hinzu, die sogenannte „Commit-Message“. Diese ist im Logfile sowie auf Github.com sichtbar und sollte den Inhalt des Commits sinnvoll beschreiben. Commiten sollte man dann wenn etwas (fast) vollständig Funktionsfähiges vorhanden ist und der committete Inhalt andere Teammitglieder nicht behindert. (Fehler etc.)

```
$git commit -m "Dies ist eine Commit-Message"
```

Code 3 git commit

4.5 Befehl: „push“

Mit `git push` lädt man die bereits committeten Dokumente auf den Server hoch.

```
$git push
```

Code 4 git push

4.6 Befehl: „pull“

Mithilfe des Befehls `git pull` lädt man alle Dokumente herunter, die von anderen Benutzern geändert und hochgeladen wurden. Wenn ein anderes Projektmitglied also den `git push` Befehl benutzt hat, muss man selbst mit `git pull` seinen lokalen Ordner auf den neuesten Stand bringen.

```
$ git pull
remote: Counting objects: 20, done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 20 (delta 14), reused 11 (delta 5), pack-reused 0
Unpacking objects: 100% (20/20), done.
From github.com:mgoebel-tgm/Bring-Buy
   09dfb03..d725d79  development -> origin/development
Updating 09dfb03..d725d79
Fast-forward
 doc/Technologien/JSON_Technolgie.docx          | Bin 49122 -> 56429 bytes
 doc/Technologien/JSON_Technolgie_20150218.pdf | Bin 0 -> 703765 bytes
 doc/Technologien/selenium_technologie.docx     | Bin 205413 -> 205034 bytes
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 doc/Technologien/JSON_Technolgie_20150218.pdf
```

Abbildung 6 git pull

5 Branches

Eine Branch in GitHub bezeichnet einen anderen „Zweig“ des Programmes.
Standardmäßig befindet man sich im „master“ Branch. Man könnte neue Branches, zum Beispiel für optionale Features, erstellen, welche erst in ferner Zukunft in die Software einfließen sollen. Bei DSN werden zwei Branches definiert: development und stable. Auf dem development-Branch findet dich hauptsächliche Entiwicklung statt und auf dem stable-Branch werden stabile Versionen der Software veröffentlicht.

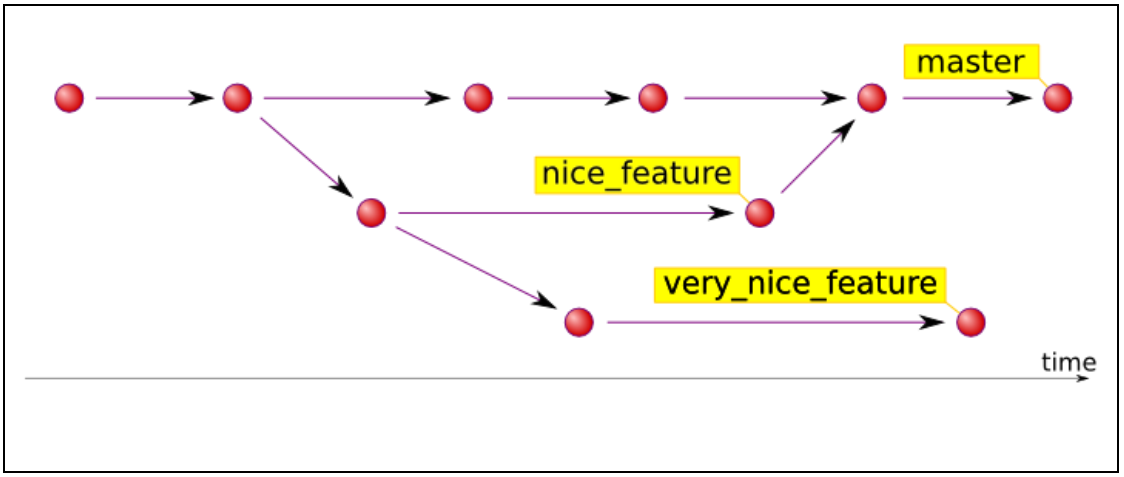

Abbildung 7 verschiedene Branches [5]

6 Merge Konflikt

Ein Merge Konflikt kommt dann zustande, wenn zwei oder mehr Personen ein und dasselbe File an derselben Stelle ändern und dann versuchen das File hochzuladen und zusammen zu führen, zu „mergen“. Um diesen Konflikt zu beheben, muss man das File so ändern, dass bei allen Personen dasselbe im File steht und es dann hochladen. Zusätzlich kann hier ein `git stash` hilfreich sein.

```
$ git status
# On branch branch-b
# You have unmerged paths.
#   (fix conflicts and run "git commit")
#
# Unmerged paths:
#   (use "git add ..." to mark resolution)
#
# both modified:      planets.md
#
no changes added to commit (use "git add" and/or "git commit -a")
```

Abbildung 8 Merge-Konflikt [6]

7 Abbildungsverzeichnis

8 Codeverzeichnis

9 Quellen

[1] Julia Meindl , „Unterschied zwischen Git und Github“,
https://alphanodes.com/de/unterschied-zwischen-git-github, zuletzt besucht: 18.02.2015

[2] Github Inc, “Which remote URL should I use?”,
https://help.github.com/articles/which-remote-url-should-i-use/, zuletzt besucht: 18.02.2015

[3] Github Inc, “Generating a SSH-Key”,
https://help.github.com/articles/generating-ssh-keys/, zuletzt besucht: 18.02.2015

[4] GitHub Inc., „Github Hauptseite“,
https://github.com/, zuletzt besucht: 18.02.2015

[5] Edward „Hades“ Toroshchin, „The House Of Hades“,
http://hades.name/blog/tag/git/, zuletzt besucht: 18.02.2015

[6] GitHub Inc., „GitHub Help“,
https://help.github.com/articles/resolving-a-merge-conflict-from-the-command-line/, zuletzt besucht: 18.02.2015