0.1 GitHub

Beim Nachforschen und beim Arbeiten mit GitHub wurde der Artikel [*Was ist eigentlich dieses GitHub?* [1]] und die Webseite [*GitHub* [2]] verwendet.

GitHub ist ein System zur kollaborativen Versionsverwaltung für Software-Entwicklungsprojekte. Für GitHub gibt es zahlreiche Clients, die meisten von diesen sind Kommandozeilen-Tools. Mit diesen Clients ist es möglich, Änderungen an Projekten zu speichern und jederzeit wieder auf vorherige Versionen zugreifen zu können. GitHub bietet viele Statistiken und ermöglicht auch Benutzern, ohne Erfahrung mit Kommandozeilen-Befehlen, einen sehr einfachen Umgang.

0.1.1 Repositories

Jedes Projekt wird in einem Repository, kurz auch Repo, abgespeichert.

Public Repositories Public Repositories können mit einem kostenlosen GitHub Account erstellt werden. Jeder kann dieses Repo sehen, aber der Entwickler entscheidet wer Commits in dieses Repo absetzen kann.

Private Repositories Will man allerdings ein Private Repository einrichten muss man den Status seines Accounts auf Micro upgraden. Dieses Upgrade kostet \$7.00/Monat. In einem Private Repository kann der Besitzer entscheiden wer dieses Repo sehen und wer Commits in dieses Repo absetzen kann.

0.1.2 Branches

Innerhalb eines Repositories gibt es mehrere Versionen einer Software. Diese Versionen werden in verschiedenen Branches abgespeichert. Es gibt eine Version auf der master-Branch die lauffähig und funktionstüchtig sein sollte. Neue Versionen werden in Branches entwickelt und können durch Pull Requests in die Version der master-Branch übernommen werden. Auf diese Weise werden parallele Entwicklungen ermöglicht. Beispielsweise kann an einer Version weiterentwickelt und alle Fehler ausgebessert werden. Gleichzeitig kann in einer Branch an größeren Änderungen, zum Beispiel an dem Wechsel auf eine alternative Technologie, gearbeitet werden.

0.1.3 Commits

Ein Commit ist der Vorgang eine neue Version einzureichen. Das bedeutet, dass diese neue Version auf die aktuelle Branch übernommen wird. Zu diesen Commits kann zurückgesprungen werden. Dies ermöglicht eine sehr hohe Transparenz im Entwicklungsvorgang. Das Projekt sollte zum Zeitpunkt eines Commits immer lauffähig sein!

0.1.4 Pull Request

Wurde eine Funktion eingebaut oder ein Bug gefixt, kann ein Entwickler die Änderungen seiner Branch in die master-Branch mittels einer Pull-Request übertragen. Ein Administrator des Repositories begutachtet die Änderungen und entscheidet ob er diesen Code in die master-Branch übernehmen will.

0.1.5 Community

Durch die große Community und Features wie den Entwicklerprofilen ermöglicht GitHub eine große Vernetzung in der Entwicklergemeinde. Personen können einem Entwickler folgen und so alle seine Projekte und Updates verfolgen. Jeder kann sich kostenlos die neueste Version eines Projekts herunterladen, diese einsetzen und daran mitentwickeln.

0.1.6 Issues

Da GitHub auf Kollaboration ausgelegt ist, bietet die Webseite auch ein Ticketsystem. Jeder mit einem kostenlosen Account kann Tickets erstellen. Eine Issue kann, von einem Administrator des Repositories, einem Entwickler zugeteilt werden. Im Rahmen dieses Systems ist ein Austausch der Entwickler mit der Community möglich. Diese Tickets weisen auf Bugs im Projekt hin, schlagen Verbesserungen des Codes vor oder stellen den Entwicklern fragen. Ist das Problem gelöst kann eine Issue geschlossen werden.

0.1.7 Integration

GitHub läss sich sehr einfach in die meisten Entwicklungsumgebungen integrieren. Durch den hohen Bekanntheitsgrad wird es von den populärsten IDE's unterstützt. Sollte so ein Support nicht standardmäßig vorliegen, werden zahlreiche Extensions angeboten die so einen Support ermöglichen.

0.1.8 .gitignore

GitHub bietet die Möglichkeit mehrere .gitignore Dateien anzulegen. In diesen .gitignore Files werden Dateien aufgezählt die von GitHub ignoriert werden sollen. Dies umfasst hauptsächlich generierte Dateien oder Konfigurationsdateien. Dabei werden unnötige Änderungen und Konflikte, mit den Konfigurationsdateien anderer Entwicklern, vermieden.

```
1  FIPLY/App/app/app.iml
2  FIPLY/App/.idea/misc.xml
3  FIPLY/App/.idea/gradle.xml
4  FIPLY/App/.idea/vcs.xml
5  *.log
6  *.toc
7  *.aux
8  *.synctex.gz
9  *.blg
10 *.bbl
11 *.bak
12 *.bcf
13 *.run.xml
```

Das .gitignore im root Ordner `2015\_fiply`

```
1 .gradle
2 /local.properties
3 /.idea/workspace.xml
4 /.idea/libraries
5 .DS_Store
6 /build
7 /captures
```

Das .gitignore im Ordner unseres Androidprojekts `2015_fiply\FIPLY\App`

Einträge wie "*.log" ignorieren alle Dateien mit dieser Dateiendung.
Einträge wie "FIPLY/App/app/app.iml" ignorieren genau diese spezifische Datei.
GitHub bietet standardmäßige .gitignore Dateien an, die auf bestimmte Technologien, wie beispielsweise Android, Latex, JavaScript..., abgestimmt sind.

0.1.9 GitHub Desktop

GitHub Desktop ist der Standardclient für GitHub. Ist dieser Client installiert, ist eine enge Zusammenarbeit mit der GitHub Webseite möglich. So kann man beispielsweise durch einen einzelnen Klick auf der Webseite ein Repository oder eine bestimmte Branch herunterladen. Die meisten Basisfunktionen sind in GitHub Desktop vorhanden und sehr einfach zu bedienen. Im Laufe der Arbeit stellte sich jedoch heraus, dass dieser Funktionsumfang nicht immer ausreicht. Für diese Fälle ist die bei GitHub Desktop beiliegende Git Shell zu verwenden. Hier ist ein weitaus größerer Funktionsumfang gegeben. Dieser wird auch benötigt wenn Probleme, wie komplexe Konflikte bei Commits oder Pull Requests, auftreten.