O'REILLY®

Inklusive Git-Basics

GitHub

Eine praktische Einführung

Von den ersten Schritten bis zu eigenen GitHub Actions

Anke Lederer

GitHub – Eine praktische Einführung

Von den ersten Schritten bis zu eigenen GitHub Actions

Anke Lederer

O'REILLY®

Anke Lederer

Lektorat: Ariane Hesse
Korrektorat: Sibylle Feldmann, *www.richtiger-text.de*
Satz: III-satz, *www.drei-satz.de*
Herstellung: Stefanie Weidner
Umschlaggestaltung: Michael Oréal, *www.oreal.de*
Druck und Bindung: mediaprint solutions GmbH, 33100 Paderborn

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über *http://dnb.d-nb.de* abrufbar.

ISBN:
Print 978-3-96009-141-7
PDF 978-3-96010-426-1
ePub 978-3-96010-427-8
mobi 978-3-96010-428-5

1. Auflage 2021

Wieblinger Weg 17
69123 Heidelberg

Dieses Buch erscheint in Kooperation mit O'Reilly Media, Inc. unter dem Imprint »O'REILLY«. O'REILLY ist ein Markenzeichen und eine eingetragene Marke von O'Reilly Media, Inc. und wird mit Einwilligung des Eigentümers verwendet.

Hinweis:
Dieses Buch wurde auf PEFC-zertifiziertem Papier aus nachhaltiger Waldwirtschaft gedruckt. Der Umwelt zuliebe verzichten wir zusätzlich auf die Einschweißfolie.

Schreiben Sie uns:
Falls Sie Anregungen, Wünsche und Kommentare haben, lassen Sie es uns wissen: kommentar@oreilly.de.

5 4 3 2 1 0

Inhalt

Vorwort

Liebe Leserin, lieber Leser,

schön, dass du hier bist! Warum ist dieses Buch entstanden? Als ich anfing, mich mit GitHub zu beschäftigen, war ich zunächst erschlagen von der Plattform. Ich fragte mich, wo und wie ich einsteigen sollte und was überhaupt wichtig sei. Ich habe daraufhin versucht, die unzähligen Quellen, die einen leichten Einstieg versprochen haben, zu durchforsten, um eben diesen Einstieg zu bekommen. Ich habe brillante Tutorials und grottenschlechte Videos gefunden und viel Zeit damit verbracht, die Infos zu filtern und nach Nützlichkeit zu sortieren.

Gewünscht hätte ich mir einen kompakten Einstieg, der an einer Stelle alles für eine Anfängerin Relevante zusammenfasst und erklärt – ohne einen neuen, unbekannten Fachbegriff in unverständlichem Fachchinesisch. Am liebsten wäre mir ein Buch gewesen, das mich an die Hand nimmt und mir schrittweise zeigt, was wichtig ist, wo ich was finde und wie das alles eigentlich geht. Das gab es aber in dieser Form nicht. Also habe ich versucht, eine möglichst praktische und hoffentlich verständliche Einführung zu schreiben, und du hast sie gerade vor deiner Nase.[1]

Ist dieses Buch das richtige für mich?

Du bist hier richtig, wenn du dich bei der einen oder anderen nachfolgenden Beschreibung wiederfinden kannst:

- Du programmierst in deiner Freizeit gern Apps und suchst einen Weg, um sie zu veröffentlichen. Vielleicht möchtest du sogar zusammen mit deinen Freunden gemeinschaftlich an einer App arbeiten.

1 Ich freue mich über Feedback: *githubbuch@ist-einmalig.de* – ja ich weiß, es klingt etwas überheblich, die Alternativen wären *@alphafrau.de* oder *@streber24.de* gewesen ...

- Du warst schon immer fasziniert von Open Source und hast dich stets gefragt, wie und wo man solche Projekte eigentlich unterstützen kann. Zudem fragst du dich, ob man dafür zwingend programmieren können muss.[2]
- Du benutzt schon lange eine App und möchtest die Entwicklerinnen und Entwickler über einen Fehler, der dich nervt, informieren. Beim Klicken auf *Fehler melden* bist du auf einer GitHub-Seite gelandet und bist erst einmal nicht weitergekommen. Das wurmt dich, und du möchtest es ändern.
- Du hast schon einmal versucht, dich mit GitHub auseinanderzusetzen, aber das war dir dann doch alles zu kompliziert. Du möchtest jetzt gern einen neuen Versuch starten und erhoffst dir, endlich wirklich zu verstehen, wie das alles läuft und warum Pull-Requests nicht Push-Requests heißen.[3]
- Du möchtest eigentlich nur wissen, wo und wie du den Quellcode zu einer bestimmten Software herunterladen kannst.[4]
- Du benutzt GitHub schon eine Weile, aber bis auf ein wenig Geklicke auf der Weboberfläche hast du dich noch nicht viel damit beschäftigt. Jetzt möchtest du intensiver einsteigen und vor allem auch dieses ominöse Git ausprobieren.
- Du benutzt Git schon eine Weile und möchtest jetzt die Funktionen von GitHub kennenlernen.

Ich habe dieses Buch primär für Anfängerinnen und Anfänger geschrieben, die sich entweder noch gar nicht oder erst ein bisschen mit GitHub beschäftigt haben. Profis hingegen werden inhaltlich vermutlich nicht viel Neues finden.

Ich setze voraus, dass du auf deinem Computer mit dem Betriebssystem deiner Wahl (beispielsweise Windows, macOS, Linux etc.) auf Anwendungsniveau zurechtkommst. Du kannst z. B. einen Browser öffnen und eine Webseite aufrufen, Software installieren, ein Verzeichnis anlegen sowie Dateien erstellen und kopieren.

Für wen ist dieses Buch nicht geeignet?

Für wen ist dieses Buch vermutlich nichts?

- Du möchtest dich intensiv mit Git beschäftigen und in die tiefsten Tiefen abtauchen, GitHub interessiert dich - wenn überhaupt - nur am Rande. Git werden wir jedoch nur relativ oberflächlich behandeln. Leseempfehlungen für einen tieferen Einstieg gebe ich in Kapitel 7, Abschnitt »Sich weiter schlaumachen über Git« auf Seite 160.
- Deine erste Aktivität nach dem Frühstück ist es, alle eingegangenen Pull-Requests zu überprüfen und ein paar Merges durchzuführen, bevor du dir

2 Spoiler: Nein, muss man nicht.

3 GitHub-Profis werden darüber vermutlich die Stirn runzeln, aber genau solche Fragen stellen sich Neulinge. Und Spoiler: Wir werden diese Frage gemeinsam klären.

4 Auch diese Frage klären wir, sollte das aber deine einzige zu GitHub sein, empfehle ich dir eher eine entsprechende Webrecherche als dieses Buch.

nach dem Mittag alle neuen Issues anschaust und zur Kaffeezeit überlegst, ob nicht mal ein neuer Branch fällig wäre.

- Du bist für dein Unternehmen auf der Suche nach einem Werkzeug, das das gemeinschaftliche Arbeiten unterstützen soll. Dafür interessieren dich die businessrelevanten Informationen, z.B. welche Business-Features GitHub anbietet, wie man es selbst hosten könnte und ob sich der Business Case rechnet.
- Du liebst knallharte IT-Fakten und findest es albern, wenn man versucht, IT anhand von Beispielen, Bildern oder Vergleichen zu erklären.[5]
- Du findest Fußnoten nervig.[6]

Da GitHub eine lebende Plattform ist, kann es sein, dass sich nach Druck des Buchs schon wieder einiges geändert hat. Buttons könnten an einer anderen Stelle sein, vorgestellte GitHub-Projekte (auch Repositories oder kurz Repos genannt, übersetzt »Aufbewahrungsort«) nicht mehr existieren oder verwaist sein oder neue Features zur Verfügung stehen. Das sollte allerdings kein Problem sein, da dieses Buch die grundlegenden Prozesse beschreibt, sodass du dich mit diesem Wissen auch auf einer veränderten Benutzungsoberfläche zurechtfinden solltest. Ich werde im Verlauf dieses Buchs die Begriffe Projekt und Repository synonym verwenden.

Der Leser oder die Leserin?

Ich persönlich bin ein großer Fan davon, alle Menschen gleichermaßen mit einzubeziehen, weswegen ich das generische Maskulinum[7] für nicht mehr zeitgemäß halte. Um Konstruktionen wie »Mein_e Leser_in, der_die mein Buch liest« oder »Mein*e Leser*in, der*die mein Buch liest« zu vermeiden, werde ich je nach Kontext entweder eine Beidnennung vornehmen (»Leserinnen und Leser«), das Gendersternchen (»Leser*innen«), das generische Maskulinum (»der Leser«) oder das generische Femininum (»die Leserin«) verwenden, gemeint sind damit immer alle Geschlechter.

Wie ist dieses Buch zu lesen?

Manche Menschen lesen ein Fachbuch von vorne bis hinten durch, andere springen in die Kapitel, die für sie attraktiv klingen. Ich habe dieses Buch so konzipiert, dass ein blutiger Anfänger es von vorne nach hinten durchlesen sollte. Sofern du bereits etwas Erfahrung hast, kann ein »Kapitel-Hopping« eventuell sinnvoll sein, beispielsweise wenn du schon weißt, wie man mit einem eigenen Projekt auf Git-

5 Wir werden in diesem Buch Häuser renovieren, Kinderkostüme aufhübschen, Teenager zum Küche putzen bewegen und Laster beladen ...

6 Schau mich nicht so an, ich habe mir das von Christina Czeschik und Matthias Lindhorst aus »Weniger schlecht über IT schreiben« abgeschaut. Ein Buch, das ich übrigens sehr empfehlen kann!

7 *https://de.wikipedia.org/wiki/Generisches_Maskulinum*

Hub umgeht, und nur wissen willst, wie du Open-Source-Projekte, die du unterstützen möchtest, finden kannst. Dann ist es eventuell durchaus sinnvoll, in das entsprechende Kapitel zu springen. Ich persönlich empfehle aber ein vollständiges, lineares Lesen, und das nicht nur, weil ich mir so viel Mühe gemacht habe :).

Dieses Buch enthält viele Links auf andere GitHub-Repositories und Websites. Da einige davon aufwändig abzutippen sind, findest du sie, um dir die Recherche zu erleichtern, auch in diesem Repository auf GitHub: *https://github.com/githubbuch/githubbuch.github.io* (siehe auch Abbildung 1).

Abbildung 1: Alle Links in diesem Buch sind in dem Repository https://github.com/githubbuch/githubbuch.github.io zu finden.

Konventionen in diesem Buch

Ich weiß nicht, wie es dir geht, aber immer wenn ich in einem Buch die Abschnittsüberschrift »Konventionen« lese, überspringe ich das Kapitel am liebsten, weil meist nichts Spannendes drinsteht. Insofern halte ich es kurz.

Wir werden teilweise auf der Konsole arbeiten (unter Windows häufig auch Eingabeaufforderung genannt, siehe Abbildung 1). Das werde ich folgendermaßen darstellen:

```
$ ls
datei.txt
```

Wenn ich exemplarisch Eingaben auf der Konsole zeigen möchte, also etwas, was du so nicht eins zu eins eintippen solltest, wähle ich Großbuchstaben:

```
$ vi DATEINAME
```

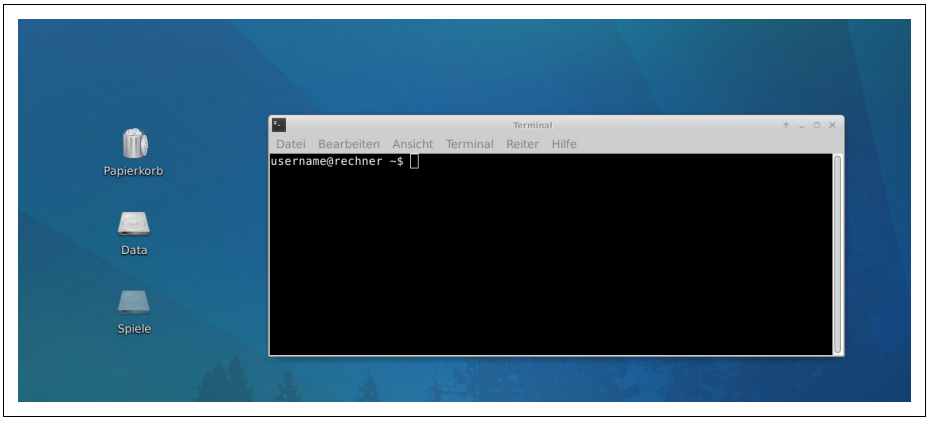

Abbildung 1: Die Konsole ist im Gegensatz zu einer grafischen Schnittstelle eine textbasierte Schnittstelle zum Betriebssystem und ermöglicht einen direkten Zugriff auf Betriebssystemressourcen und Programme.

Hier soll der Texteditor `vi` mit einer Datei deiner Wahl gestartet werden, `DATEINAME` dient als Platzhalter. Wer mit der Konsole und deren Ausgaben noch nicht gearbeitet hat, für den habe ich extra einen Abschnitt geschrieben (siehe Kapitel 7, Abschnitt »Exkurs: Umgang mit der Konsole« auf Seite 139).

Konsolenbefehle oder Ausschnitte aus Code-Beispielen im Fließtext, werden in Listingschrift dargestellt. URLs, E-Mail-Adressen, Dateinamen oder Dateierweiterungen sind in *Kursivschrift* formatiert.

Tipps, Anmerkungen und alles, was ich für hervorhebenswert halte, kommen in folgende schnuckelige Kästen:

Tippkasten
In einem solchen Kasten werde ich wichtige Hinweise, Tipps, Tricks und Best Practices kurz hervorheben, die meiner Meinung nach hilfreich und/oder interessant sein könnten.

Stolperfallen und alles, was das Leben schwerer machen könnte, wenn man es nicht kennt und beachtet, werde ich in solche Kästen schreiben:

Stolperfallenkasten
In einem solchen Kasten werde ich Stolperfallen kenntlich machen.

Für längere Erklärungen gibt es die »Erklärbärbox«:

> **Erklärbärbox**
>
> In diese Boxen kommen längere Erläuterungen zu einem bestimmten Thema, die den Lesefluss im »normalen« Text stören würden. Falls dich das Thema nicht interessiert oder du darüber schon gut Bescheid weißt, kannst du diese Boxen einfach überspringen.

Alles, was meiner Ansicht nach den Lesefluss stören würde, werde ich in Fußnoten packen: beispielsweise Webadressen, Details, die für das weitere Verständnis nicht unbedingt wichtig sind, und auch persönliche Kommentare.

Danksagung

Danksagung sind für Leser*innen häufig öde, daher mache ich es kurz und schmerzlos:

- Als Erstes möchte ich meinem Mann **Rüdiger** danken, der mich während des Schreibens oft genug aus meinem Schreibtunnel hervorlocken oder aber meine Begeisterung für irgendwelche fachlichen Details ertragen musste.
- Weiterer Dank gebührt meinen ganzen Querleser*innen, die das Buch in Teilen gelesen und mir Feedback gegeben haben: **Henry Hillje**, **Alina Robbers**, **Halil Ege**, **Jan Schnedler** und (wieder) **Rüdiger**.
- Dann gab es noch ein paar Verrückte, die das Buch als Ganzes gelesen und mir ebenfalls sehr hilfreiches Feedback gegeben haben: **Julia Barthel**, **Florian Diedrich**, **Sven Riedel**, **Nina Siessegger** und **Ayleen Weiß**. Dank eurer Unterstützung sind noch einige Praxistipps eingeflossen, konnte ich einige Unklarheiten ausräumen und habe zudem meinen wortreichen Schreibfluss etwas im Zaum gehalten ;-).
- Danke auch an meine Lektorin **Ariane Hesse**, die (hoffentlich) alle Stolpersteine gefunden und dadurch dieses Buch zu einem besseren gemacht hat.
- Nicht vergessen möchte ich **Christina Czeschik** und **Matthias Lindhorst**, beide Autor*innen des Buchs *Weniger schlecht über IT schreiben*, die mich durch ihr Buch überhaupt erst dazu inspiriert und motiviert haben, selber ein Buch zu schreiben. Danke dafür!
- Danken möchte ich auch der **Open-Source-Community**. Ohne die unermüdliche Unterstützung dieser Menschen wäre die (Software-)Welt eine ärmere. Ich werde daher ein Teil des Bucherlöses an einige Open-Source-Projekte spenden.

Git als Unterstützungstool für dieses Buch
Dieses Buch ist mithilfe von Git erstellt worden. Dafür habe ich 882 Commits auf dem master-Branch durchgeführt.

KAPITEL 1

Was ist GitHub, und wofür brauche ich es?

GitHub ist eine der wichtigsten Anlaufstellen im Internet für Softwareentwickler*innen, Open-Source-Enthusiast*innen und andere Interessierte. Viele bekannte Projekte werden dort weiterentwickelt und veröffentlicht, wie beispielsweise Googles User-Interface-Entwicklungskit *Flutter*[1], das Machine-Learning-Framework *TensorFlow*[2] (ebenfalls Google) oder das CAD-Programm *FreeCAD*[3] (CAD steht für *Computer Aided Design*). Auch wer Bibliotheken und Erweiterungen für Programmiersprachen oder Unterstützung bei Projekten mit dem günstigen Einplatinencomputer Raspberry Pi sucht, wird häufig auf einem GitHub-Repository landen. Spätestens seit der Corona-Warn-App[4], deren Quellcode auf GitHub veröffentlicht wurde,[5] ist die Plattform auch einem breiteren Publikum namentlich bekannt.

Unternehmen wie Facebook, Ford, Spotify, 3M und viele weitere haben die Plattform ebenfalls für sich entdeckt und entwickeln und veröffentlichen dort eigene Projekte. Mittlerweile ist es sogar nicht unüblich, das potenzielle Arbeitgeber*innen einen Blick auf den GitHub-Account einer sich bewerbenden Person werfen und diesen in ihre Entscheidungsfindung mit einfließen lassen. Es kann also viele Gründe geben, sich mit GitHub beschäftigen zu wollen.

Am Ende des Kapitels weißt du ...

- was GitHub ist und wofür es eingesetzt wird.
- wie Git und GitHub zusammenhängen.
- welche Funktionen von GitHub kostenlos nutzbar sind.

1 *https://github.com/flutter/flutter*

2 *https://github.com/tensorflow/tensorflow*

3 *https://github.com/FreeCAD/FreeCAD*

4 *https://www.coronawarn.app/de/*

5 *https://github.com/corona-warn-app*

Open Source

Auf den ersten Blick könnte man denken, mit Open Source sei ausschließlich gemeint, den Quellcode (Quelle, englisch *Source*) offenzulegen. Das ist aber nicht das einzige Merkmal. Vom Grundgedanken her gewährt Open Source die folgenden vier Freiheiten:

- **Verwenden:** die Software für beliebige Zwecke nutzen.
- **Verbreiten:** die Software uneingeschränkt an andere weitergeben.
- **Verändern/verbessern:** die Software an eigene Bedürfnisse anpassen.
- **Verstehen:** den Quellcode untersuchen.

Es existieren Hunderte von Open-Source-Lizenzen, die jeweils eine andere Kombination dieser Freiheiten regeln. Das sehen wir uns in Kapitel 5 noch etwas genauer an. Open-Source-Software ist meistens kostenlos, muss es aber nicht sein.

Rund um das Thema Open Source wirst du zudem eine Vielzahl von Begriffen wie *Free Software*, *Libre Software*, *Free and Open Source Software* (FOSS) sowie *Free/Libre and Open Source Software* (FLOSS) finden. Diese Begriffe haben nicht unbedingt die gleiche Bedeutung, sind inhaltlich aber verwandt. Lass dich davon jedoch nicht verwirren. Diese Namensvielfalt kommt daher, dass jeweils unterschiedliche Aspekte von Freiheit besonders betont werden.[6] Ich werde in diesem Buch den Begriff Open Source verwenden.

Was bietet GitHub?

GitHub ist ein webbasierter kollaborativer Hosting-Dienst für Git-Projekte. »Aha«, mag sich der eine oder die andere denken und ist immer noch nicht schlauer. Gehen wir das mal gemeinsam durch.

- **Webbasiert** bedeutet nichts anderes als »befindet sich im Internet«.
- **Kollaborativ** bedeutet »zusammenarbeiten«.
- Ein **Hosting-Dienst** ist ein Dienst, der Ressourcen wie beispielsweise Speicherplatz oder Softwareanwendungen bereitstellt. Häufig handelt es sich dabei um »Plattenspeicherplatz«, um Dateien im Internet ablegen zu können, beispielsweise für eine Webseite.
- **Git**[7] ist eine freie Software zur Versionsverwaltung von Dateien. Versionsverwaltungen bieten Werkzeuge an, um mit Arbeits- und Zwischenständen von Dateien besser arbeiten zu können (mehr dazu später im Abschnitt »Versionsverwaltung« auf Seite 136 in Kapitel 7). Git wird in der Regel lokal auf dem eigenen Rechner installiert und verwaltet häufig programmierten Code. Aber

6 Wer tiefer eintauchen möchte: *https://fsfe.org/freesoftware/basics/comparison.de.html*.

7 *https://git-scm.com/*

auch Webseiten, Firmenkorrespondenz oder Gedichte wären möglich (genau genommen alle Arten von Textdateien). Darüber hinaus gibt es die Möglichkeit, einen eigenen Git-Server zu installieren.

- Unter **Projekten** sind alle Dateien zu verstehen, die zu einem bestimmten Thema gehören. Ein Projekt könnte beispielsweise eine Vereins-Website sein, die neben den eigentlichen Seiten beispielsweise Bilder und Beitrittsformulare für den Verein beinhaltet.

GitHub bietet also *Projekten*, die mit der Software *Git verwaltet* werden, eine *Speichermöglichkeit im Internet* an, um *zusammen mit anderen* daran arbeiten zu können. Manche bezeichnen es auch als soziales Netzwerk rund um Softwareprojekte.

Vielleicht fragst du dich gerade, ob du Git jetzt installieren musst, um GitHub überhaupt nutzen zu können. Die Antwort lautet: Nein, GitHub geht auch ohne Git![8] Wir werden sogar den Großteil dieses Buchs ohne Git auskommen und erst in einem späteren Kapitel Git zu Hilfe nehmen (siehe Kapitel 7).

GitHub basiert auf vielen Konzepten von Git, wie beispielsweise dem Erstellen von Branches oder dem Durchführen von Merges. Es vereinfacht einige Git-Vorgänge dadurch, dass Aktivitäten durch einfaches Anklicken im Browser durchgeführt werden können, anstatt auf der Konsole Befehle mit mehreren Parametern angeben zu müssen. GitHub stellt aber auch noch einige zusätzliche Funktionen bereit, um das kollaborative Arbeiten zu erleichtern. All das wirst du im Verlauf des Buchs kennenlernen.

Einsatzgebiete von GitHub

Wofür kann ich GitHub denn jetzt konkret einsetzen? GitHub ist vor allem dann sinnvoll, wenn du vorhast, mit mehreren Personen gemeinschaftlich an einem Projekt zu arbeiten (Stichwort *kollaborativ*). Ich habe bisher folgende Anwendungsfälle für GitHub identifiziert:

1. Gemeinschaftlich mit mehreren Personen an einem Projekt arbeiten.
2. Ein Projekt bzw. die Ergebnisse eines Projekts veröffentlichen, häufig nach dem Prinzip »fire and forget« (sinngemäß: »einmal veröffentlichen und danach nicht weiter anpassen«) – das ist beispielsweise immer mal wieder auch bei (Programmier-)Büchern anzutreffen.
3. Ein Projekt veröffentlichen, um anderen interessierten Personen eine Schnittstelle zu bieten, beispielsweise um zu unterstützen oder Fehler zu melden.
4. Dateien »im Internet« abspeichern, um schnell und einfach von überall auf sie zugreifen zu können.[9] Oft ist das mit dem Wunsch verbunden, zeitgleich an-

8 Andersherum natürlich ebenfalls: Git geht auch ohne GitHub!

9 Ich kenne Menschen, die für ihren personalisierten Linux-Arbeitsplatz die entsprechenden Konfigurationsdateien auf GitHub stellen, um sie mit wenigen Befehlen auf ihren aktuellen Rechner ziehen zu können.

deren die Dateien ebenfalls zur Verfügung zu stellen (wie bei 2) sowie Feedback und Unterstützung zu bekommen (wie bei 3).

Wobei die Übergänge fließend sind. Zu einem gemeinschaftlichen Projekt können plötzlich – manchmal ungefragt – interessierte Personen dazustoßen, oder ein ehemals stark frequentiertes Projekt verwaist und ist nur noch eine historische Veröffentlichung ohne weiteres Leben.

Eine große Stärke von GitHub ist es, ein Repository relativ einfach mit anderen Diensten verknüpfen zu können, seien es Containerlösungen, die einem das Ausliefern von Software erleichtern (etwa Docker), Übersetzungsunterstützungstools, Projektmanagementanwendungen und vieles mehr.

Ein wichtiger Punkt ist das Thema Sprache. Wer sich auf GitHub tummelt, sollte englisch lesen und schreiben können, da viele Informationen, Tutorials und Projekte nur auf Englisch verfügbar sind und die meiste Kommunikation auf Englisch stattfindet. Deutschsprachige Projekte gibt es zwar auch, sie sind aber eher selten.

Grundsätzlich sind die auf GitHub veröffentlichten Dokumente erst einmal für jeden und jede offen einsehbar. Es gibt aber auch die Möglichkeit, private Repositories anzulegen, darauf werde ich in Kapitel 3, Abschnitt »Das erste eigene Repository anlegen« auf Seite 24, noch eingehen.

Git, GitHub, GitLab – alles das Gleiche?

Wir haben bereits geklärt, dass GitHub etwas mit der Versionsverwaltungssoftware Git zu tun hat. Die beiden arbeiten eng zusammen, sind aber nicht dasselbe oder würden einander gar ersetzen. Beide sind unabhängig voneinander nutzbar – ich kann ausschließlich mit Git arbeiten oder ausschließlich mit GitHub oder eben beide in Kombination einsetzen. Was ich wann brauche, hängt davon ab, was ich eigentlich genau machen will. Insbesondere bei Softwareentwicklungsprojekten werden beide häufig zusammen verwendet.

Namensgebung GitHub

Der Name GitHub kommt von seiner engen Verzahnung mit Git und lässt sich grob mit »Knotenpunkt für Git-Anwendungen« übersetzen. Initiator und Namensgeber von Git ist Linus Torvalds, der vor allem als Entwickler des Betriebssystems Linux bekannt ist.[10] Das Wort »Git« (ausgesprochen wie im Deutschen) bedeutet wörtlich übersetzt übrigens »Depp« oder »Blödmann«, und diese ungewöhnliche Namenswahl wird von Torvalds wie folgt begründet:

10 Aber nicht nur – Torvalds ist auch bekannt für seine Schimpftiraden, die mittlerweile auf eigenen Webseiten gesammelt werden, beispielsweise auf *https://www.reddit.com/r/linusrants/*.

> »I'm an egoistical bastard, and I name all my projects after myself. First ›Linux‹, now ›Git‹.«
>
> »Ich bin ein egoistischer Mistkerl, und ich benenne all meine Projekte nach mir. Zuerst ›Linux‹, jetzt eben ›Git‹.«
>
> *– Linus Torvalds*

Mit einem Augenzwinkern könnte man »GitHub« daher auch mit »Deppentreff« übersetzen. ;)

Neben GitHub gibt es weitere Plattformen, wie beispielsweise *GitLab* oder *Bitbucket*, die im Grunde einen ähnlichen Funktionsumfang bieten. Nicht unerwähnt lassen möchte ich, dass GitHub 2018 von Microsoft gekauft wurde und viele Entwickler*innen das bis heute kritisch betrachten. Deswegen gab es diverse Projekte, die von GitHub nach GitLab umgezogen sind, und manche Projekte findet man heute sogar auf beiden Plattformen.

In Tabelle 1-1 habe ich Git, GitHub und GitLab gegenübergestellt, damit du ein Gefühl für die Unterschiede bekommst. An den Zahlen erkennst du auch, dass GitHub deutlich größer ist als GitLab. Was teilweise auch darin begründet liegt, dass GitLab später gestartet ist.

Tabelle 1-1: Übersicht und Vergleich von Git, GitLab und GitHub

Name	Git	GitLab	GitHub
Logo			
Aufgabe	dezentrales Versionsverwaltungssystem	Onlineplattform für (Git-)Projekte	Onlineplattform für (Git-)Projekte
Webseite	*https://git-scm.com/*	*https://about.gitlab.com/*	*https://www.github.com*
Zahlen, Daten, Fakten	k. A.	100.000 Unternehmen und Organisationen, 30 Millionen registrierte Nutzer*innen (Stand Dezember 2020)	2,1 Millionen Unternehmen und Organisationen, 100 Millionen Repositories, 40 Millionen registrierte Nutzer*innen (Stand August 2019)
Erscheinungsjahr	2005	2011	2008

Im Verlauf unserer gemeinsamen Entdeckungsreise werden wir mit GitHub zunächst einsteigen und die ersten Schritte gehen. Später nehmen wir noch Git dazu, um die volle Bandbreite der Möglichkeiten ausschöpfen zu lernen. Am Ende wirst du selbst einschätzen können, wann du was am besten einsetzen kannst. GitLab und Bitbucket werden nicht Teil dieses Buchs sein, sollten aber mit den in diesem Buch vermittelten Grundlagen ebenfalls relativ leicht erlernbar sein. GitLab nutzt beispielsweise zum Teil dasselbe Vokabular wie GitHub.

Wie ich mein erstes fremdes Projekt unterstützt habe

Obwohl ich programmieren kann, habe ich zu meinem ersten fremden Projekt keinen Programmcode beigetragen. Ich möchte diese Geschichte erzählen, um auch Menschen ohne Programmierkenntnisse zu ermutigen, bei Open-Source-Projekten mitzuwirken.

Ich habe ein Open-Source-Spiel auf meinem Smartphone gespielt, bei dem mir die schlechte deutsche Übersetzung aufgefallen war. Manches wirkte zusammengewürfelt, und an vielen Stellen war das schönste »Denglisch«[11] zu lesen.

Ich stellte fest, dass der Autor sein Spiel auf GitHub weiterentwickelte, und begann damit, eine – in meinen Augen bessere – deutsche Übersetzung in kleineren Häppchen beizusteuern. Da der Autor sehr schnell auf meine Änderungsvorschläge reagierte, habe ich mich ermutigt gefühlt, weiterzumachen, bis ich das ganze Spiel einmal »generalüberholt« hatte. Das Gefühl, als ich dann das erste Mal »meiner« Übersetzung beim Spielen des Spiels begegnete, war unbeschreiblich!

Mit welchen Kosten muss ich rechnen?

GitHub kannst du grundsätzlich erst einmal kostenlos verwenden, ebenso wie das in diesem Buch vorgestellte Git. GitHub bietet allerdings auch kostenpflichtige Funktionen und Erweiterungen an. In diesem Buch werden wir uns aber ausschließlich mit den kostenfreien Features beschäftigen. Zum Zeitpunkt der Drucklegung dieses Buchs waren das unter anderem:

- Unbegrenzte Anzahl öffentlicher Repositories.
- Unbegrenzte Anzahl privater Repositories.
- Unbegrenzte Anzahl an Mitarbeitenden.
- 2.000 Action-Minuten pro Monat für öffentliche Repositories.
- Issues und Fehler-Tracking.
- Projektmanagement.

Das bedeutet: Man kann kostenfrei beliebig viele Projekte anlegen, und es können beliebig viele Menschen an diesen Projekten mitarbeiten. *Actions* erlauben es, das eigene Repository zu automatisieren, und es gibt dafür ein gewisses Freikontingent (das schauen wir uns in Kapitel 9 noch genauer an). Issues, Fehler-Tracking und Projektmanagement sind Werkzeuge zur Unterstützung bei der Projektabwicklung. Auch diese Werkzeuge lernst du noch kennen.

11 Kunstwort aus »Deutsch« und »Englisch«, denglische Texte sind ein Mischmasch dieser beiden Sprachen und finden sich auch in diesem Buch wieder, z. B. bei »Committe bitte …«

Für die meisten privaten Personen, die gerade ihre ersten Schritte in die Veröffentlichung von Software oder Ähnlichem gehen, ist der kostenlose Zugang völlig ausreichend.

Wer bereit ist, Geld auszugeben, kann beispielsweise das Actions-Kontingent erhöhen, bekommt sieben Tage die Woche rund um die Uhr Unterstützung bei Fragen oder Problemen (24/7-Support) oder kann die Anmeldung bei GitHub über Single Sign-on realisieren (siehe Erklärbärbox »Single Sign-on«). Die GitHub-Website[12] gibt Aufschluss darüber, was gegen Einwurf kleiner Münzen noch an weiteren Features mit welchem Preismodell möglich ist.

Single Sign-on

Single Sign-on (SSO) kann man mit »einmaliger Anmeldung« oder »Einmalanmeldung« übersetzen. Eine Benutzerin meldet sich bei einem einzelnen Dienst an, beispielsweise auf einer Website oder im Firmennetzwerk, und wird dann automatisch auch bei anderen Diensten angemeldet. Dadurch ist es für die Benutzerin nicht mehr notwendig, sich die Anmeldedaten für jeden einzelnen Dienst merken zu müssen (siehe auch Abbildung 1-1).

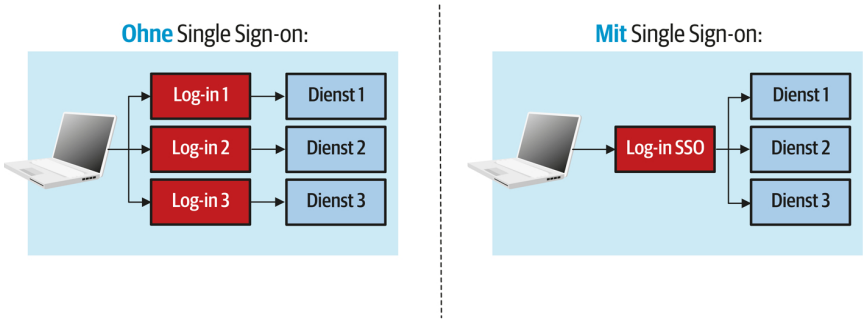

Abbildung 1-1: Mit Single Sign-on (SSO) muss sich eine Benutzerin nur noch die Zugangsdaten für den SSO-Dienst merken.

Vorteile sind unter anderem eine schnellere Anmeldung und ein erhöhter Komfort für die nutzende Person. Zudem sinkt das Risiko, dass Nutzer sich zu schwache oder leicht zu erratene Passwörter auswählen, da sie sich nicht mehr so viele unterschiedliche Passwörter ausdenken müssen (manchmal auch »Passwortmüdigkeit« genannt). Auf der anderen Seite sollte der SSO-Dienst sicherstellen, dass das gewählte SSO-Passwort stark und schwer zu knacken ist. Ein großer Nachteil: Fällt der SSO-Dienst einmal aus, kann man sich auch nicht mehr oder nur noch umständlich an den anderen Diensten anmelden.

12 *https://github.com/pricing*

KAPITEL 2

GitHub – Wir verschaffen uns einen Überblick

Die Startseiten vieler GitHub-Projekte wirken leider nicht sonderlich einladend. Man klickt auf einen Link in der Hoffnung, weitere Informationen zu einer bestimmten App zu bekommen, und das Erste, womit man begrüßt wird, ist ein Hinweis, sich einen GitHub-Account zuzulegen, und eine Auflistung von Dateien (siehe Abbildung 2-1).

Selbst für Menschen, die viel mit Dateien und Verzeichnissen arbeiten und programmieren können, wirken die meisten GitHub-Projekte daher auf den ersten Blick nicht besonders einladend. Es ist also völlig normal, wenn auch du nach etwas Herumgeklicke noch nicht schlauer bist.

Am Ende des Kapitels kannst du ...

- auf GitHub navigieren, und du weißt, wie du an benötigte Informationen kommst.
- Dateien auf GitHub finden und herunterladen.

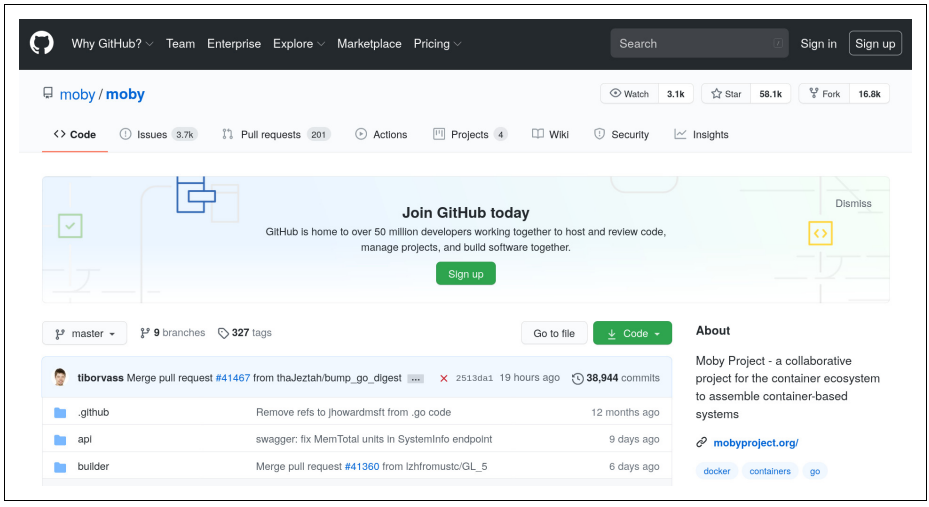

Abbildung 2-1: Ein x-beliebiges GitHub-Repository

Wir schauen uns daher zunächst an, was für den Anfang wichtig ist. In Abbildung 2-2 habe ich die Bereiche hervorgehoben, mit denen wir uns primär beschäftigen werden.

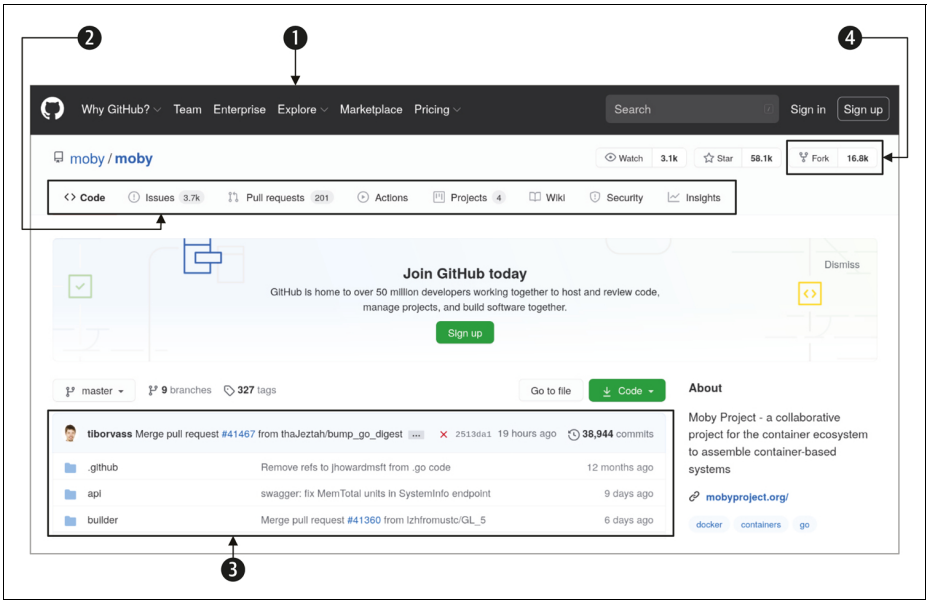

Abbildung 2-2: Die für den Anfang wichtigen Bereiche eines GitHub-Projekts

- **Bereich ❶** ist das Menü von GitHub, um auf die Startseite von GitHub zurückzukommen, GitHub zu durchsuchen oder um einen Account anzulegen.
- **Bereich ❷** ist das Menü des Repositorys. Hier kannst du zwischen den verschiedenen Funktionalitäten, die GitHub für ein Projekt bereitstellt, hin und her springen.
- **Bereich ❸** ist der Arbeitsbereich eines Repositorys. An dieser Stelle wirst du später unter anderem Projektdateien bearbeiten, Fehler im Projekt notieren und mit anderen kommunizieren.
- **Bereich ❹** ist der »magische Knopf«, um bei anderen Repositories zu unterstützen.

Wir besprechen zunächst Anwendungsfälle für GitHub und werden dann ein eigenes Projekt bzw. Repository aufbauen. Wir ignorieren aber erst einmal, dass es auch noch andere Repositories und Menschen da draußen gibt. Dafür werden wir überwiegend in den mit ❶ bis ❸ gekennzeichneten Bereichen arbeiten (siehe Kapitel 3).

Dann schauen wir uns an, was es so alles zu tun und zu beachten gibt, wenn man seine Freundinnen zum eigenen Projekt einlädt (siehe Kapitel 4). Auch hier werden wir in den Bereichen ❶ bis ❸ unterwegs sein.

Später, wenn du etwas erfahrener (und mutiger) bist, werden wir zudem schauen, wo und wie wir andere Repositories unterstützen können (siehe Kapitel 6). Dafür werden wir in den Bereichen ❶ bis ❹ arbeiten.

GitHub bietet aber noch viel mehr, z.B. den *Marketplace* oder den Menüpunkt *Explore*, der uns helfen kann, unser Projekt bekannter zu machen. Auch das Zusammenspiel mit Git hält weitere Möglichkeiten bereit. Das schauen wir uns aber erst an, wenn du einigermaßen trittsicher bist und wir die Dinge auch wirklich brauchen (das folgt in Kapitel 6, Kapitel 8 und Kapitel 9).

Das GitHub-Logo – Mona Lisa Octocat

Vielleicht ist dir schon das Logo von GitHub aufgefallen (etwa in Tabelle 1-1 in Kapitel 1): eine Katze mit Oktopus-Armen, auch als »Octocat« bekannt mit dem Namen »Mona Lisa«.[1] Diese Figur hat im Laufe der Zeit eine solche Beliebtheit unter GitHub-Nutzer*innen erlangt, dass es mittlerweile Folgendes gibt:

- Ein Blog[2], auf dem alle offiziell erstellten Octocat-Varianten veröffentlicht werden, das sogenannte »Octodex«.
- Einen Onlineshop[3], um das Maskottchen auf unterschiedlichsten Materialien zu erwerben, unter anderem auf T-Shirts, Stickern und Tassen.
- Einen eigenen Twitter-Account[4], auf dem Neuigkeiten rund um GitHub veröffentlicht werden.
- Und eine Applikation im Webbrowser[5], mit der man sich seine eigene Octocat erstellen kann.

Es gibt sogar kleine Comic-Verfilmungen mit Mona als Hauptfigur,[6] um beispielsweise Werbung für ein Event oder eine Neuerung zu machen. Auch in manchen Tutorials trifft man auf Mona.

1 Die Entstehungsgeschichte kannst du hier nachlesen: *http://cameronmcefee.com/work/the-octocat/*.

2 *https://octodex.github.com/*

3 *https://github.myshopify.com/*

4 *https://twitter.com/monatheoctocat*

5 *https://myoctocat.com/build-your-octocat/*

6 Zu finden im offiziellen GitHub-YouTube-Kanal: *https://www.youtube.com/user/github/*.

Anwendungsfälle für GitHub (oder: Was will ich da eigentlich?)

Jeder und jede hat unterschiedliche Gründe, sich auf GitHub zu tummeln. Nach meiner Erfahrung gibt es folgende Personengruppen:

1. Die **interessierte Anwenderin** möchte zu einem bestimmten Projekt ein paar weitergehende Informationen haben, z.B. Installationsanleitungen oder wie man mit einem bekannten Fehler der Software umgeht. Sobald sie einen Fehler melden möchte, wechselt sie in die Rolle der Contributorin (siehe weiter unten).
2. Der **Hilfe suchende Programmierer** programmiert gerade an etwas Eigenem (auf GitHub veröffentlicht oder auch nicht)[7] und sucht für ein spezifisches Problem Inspiration und Hilfe. Oder er sucht nur eine Software, die er herunterladen und nutzen möchte.
3. Der **Contributor** (englisch *contribute* = mitwirken, beisteuern) ist Teil eines oder mehrerer (nicht eigener) Projekte (oder möchte es mal werden) und unterstützt mit Code, Dokumentation und/oder möchte einen Fehler melden. Er macht in der Regel Vorschläge für Anpassungen, hat aber keine Rechte, diese auch durchzusetzen.
4. Die **Maintainerin** dagegen kümmert sich bei Projekten um deren Instandhaltung (englisch *maintain* = instand halten, GitHub nennt diese Rolle den *Collaborator*) und verfügt dafür über mehr Rechte als andere. Unter Instandhaltung kann alles Mögliche verstanden werden: die Codebasis aufräumen, Änderungsvorschläge annehmen oder verwerfen, eingehende Fehlermeldungen klassifizieren, das Wiki befüllen oder die entsprechende Website auf dem aktuellen Stand halten.
5. Die **Projekteignerin** hat vollen Zugriff auf ein oder mehrere Projekte. Das bedeutet, sie kann sowohl inhaltliche Änderungen als auch personelle Besetzungen vornehmen, beispielsweise Maintainer (Collaborator) benennen.

In diesem Kapitel schauen wir uns zunächst alle Aktivitäten genauer an, für die du keinen GitHub-Account benötigst. Konkret sind das die Suche nach Informationen und das Herunterladen von Dateien. Sobald du dich aktiv einbringen möchtest, beispielsweise in Form von Kommentaren oder durch das Einreichen von Änderungsvorschlägen, ist das Anlegen eines Accounts notwendig (siehe auch Tabelle 2-1). Die Konzepte, Begriffe und die typischen Arbeitsschritte, die du dafür benötigst, folgen in den restlichen Kapiteln.

7 Auf GitHub gibt es auch Inspiration zu anderen Themen als nur zu Programmierprojekten, diese anderen Themen sind hier ebenfalls gemeint. In Kapitel 11, Abschnitt »Ideen für eigene Repositories – ohne programmieren« auf Seite 259, sind ein paar solcher Repositories aufgelistet.

Tabelle 2-1: Übersicht darüber, welche Aktivitäten mit und ohne GitHub-Account durchführbar sind

Kein Account notwendig	Account notwendig
Informationen finden	kommentieren
Dateien finden	Wünsche äußern
Dateien herunterladen	Fehler melden
	Änderungsvorschläge einreichen
	Projekte veröffentlichen und bearbeiten

Informationen finden (interessierte Anwenderin)

Eine interessierte Anwenderin möchte sich über ein oder mehrere Projekte informieren. Die Startseite eines Projekts ist der beste Ausgangspunkt dafür. Oben links in der Ecke steht in der Regel der Name des Repositorys und auch der Accountname der Projekteignerin nach dem Muster »Accountname/Repositoryname«. Das Geheimnis, um an mehr Informationen zu kommen, besteht darin, einfach mal runterzuscrollen, und auf einmal zeigt sich eine ganz neue Welt an Informationen (siehe Abbildung 2-3).

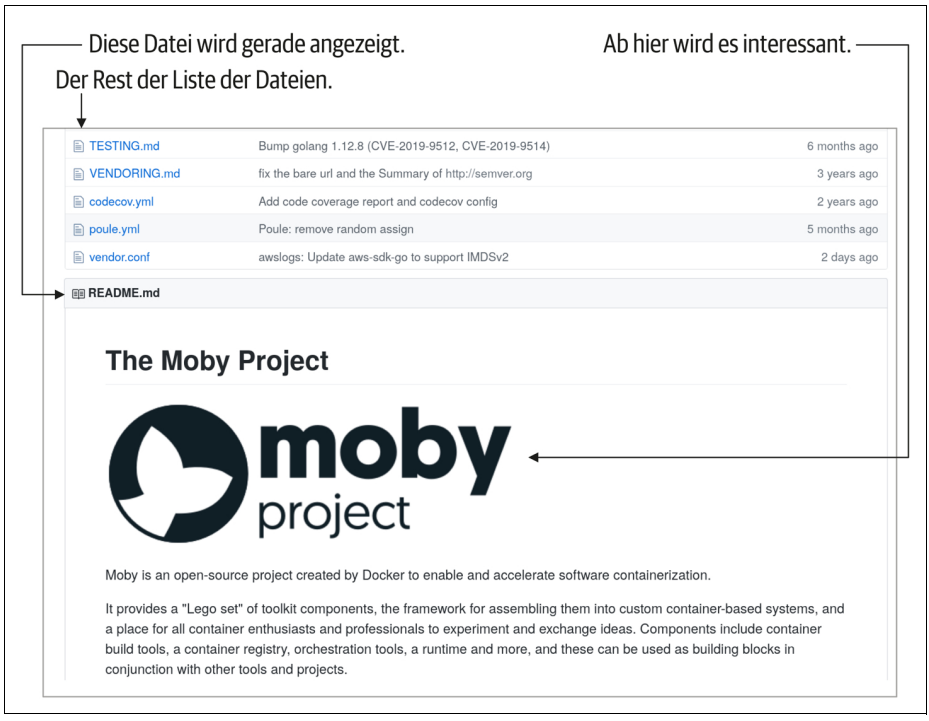

Abbildung 2-3: Nach dem Runterscrollen können die ersehnten Informationen zu einem Projekt auftauchen. Eine besondere Bedeutung hat die Datei README.md. Deren Inhalt wird hier nämlich angezeigt.

README.md

Bei GitHub gibt es eine Konvention bezüglich des Dateinamens *README.md*. Sobald sich eine solche Datei entweder im Hauptverzeichnis (auch Wurzel- oder Root-Verzeichnis genannt) oder in den Unterordnern *docs* oder *.github* (den Punkt vorweg beachten!) befindet, interpretiert GitHub diese als »Willkommensseite« und zeigt deren Inhalt den Besuchern des Repositorys an.

Idealerweise sollte in der Datei eine Reihe von Fragen zum Projekt beantwortet werden wie:

- Was macht das Projekt, und warum ist das irgendwie nützlich?
- Wie benutzt man das Projekt (z.B. wie installiert und konfiguriert man die Software)?
- Was kann eine Anwenderin machen, wenn sie Hilfe braucht?
- Eventuell: Wie kann man das Projekt unterstützen und in welcher Form?
- Eventuell: Wo findet man weiterführende Informationen?
- Eventuell: Wer entwickelt und betreibt das Projekt?

Die Endung *md* steht übrigens für »Markdown«. Das ist eine Auszeichnungssprache mit simplen Anweisungen, um optische Anpassungen und Formatierungen wie Überschriften, Fettschrift und Aufzählungszeichen zu erzeugen oder um Bilder einzufügen und Links zu setzen. Die Überschrift »The Moby Project« in Abbildung 2-3 kann man beispielsweise wie folgt erzeugen[8]:

```
# The Moby Project
```

Gute Auflistungen gängiger Befehle und wie deren Darstellung aussieht, findest du im Internet.[9] GitHub selbst bietet auf seinen Hilfeseiten ebenfalls weiterführende Informationen an.[10]

Vorlagen für eine gute *README.md* gibt es natürlich auch, beispielsweise unter *https://gist.github.com/jxson/1784669*.

Tipp: Man kann übrigens auch in Unterverzeichnissen jeweils eine *README.md* anlegen. In diesem Verzeichnis wird sie dann ebenfalls als Willkommensseite angezeigt.

Jetzt sich kann die interessierte Anwenderin ab sofort glücklich und wissend zurücklehnen.

8 Für Menschen mit Programmiererfahrung sieht es auf den ersten Blick wie ein Quellcode-Kommentar aus, es erzeugt aber tatsächlich eine Überschrift der ersten Ordnung.

9 Beispielsweise *https://github.com/tchapi/markdown-cheatsheet/blob/master/README.md*.

10 *https://help.github.com/en/github/writing-on-github/basic-writing-and-formatting-syntax*