Git für Maker, Teil 3

Ein cooles Projekt braucht auch eine ansprechende Präsentation und Dokumentation. GitHub bietet einfache und trotzdem mächtige Funktionen, um sowohl eine Website als auch eine Dokumentation mit wenigen Klicks auf die Beine zu stellen.

von Daniel Schwabe

Die Vorgänge werden beispielhaft an GitHub gezeigt, weil es sich dabei um die größte Git-Plattform handelt. Ähnliche Services funktionieren grundlegend gleich. Befehle und Konzepte lassen sich auch auf andere Anbieter anwenden.

Kurzinfo

» **Websites mithilfe von Commits anlegen**
» **Dokumentation in einem Wiki erstellen**
» **Im Bereich Issues mit der Community interagieren**

Mehr zum Thema

» Daniel Schwabe, Git für Maker, Make 1/25, S. 54
» Daniel Schwabe, Git für Maker, Teil 2, Make 2/25, S. 52
» Ákos Fodor, Von Arduino zur PlatformIO IDE, Make 1/25, S. 8

Alles zum Artikel im Web unter
make-magazin.de/xjnu

Das Wiki

In Teil 2 dieser Artikelreihe in der Make 2/25 haben wir die README.md-Datei erklärt, welche in einem Repository gespeichert ist und deren Inhalt auf GitHub direkt unter den Projektdateien angezeigt wird. Mit dieser Datei werden üblicherweise grundlegende Informationen über das Projekt vermittelt.

Wenn man für ein Projekt allerdings eine lange Dokumentation anlegen möchte, in der z. B. API-Beschreibungen, umfangreiche Tutorials für die Bedienung oder kleinteilige Code-Erklärungen zu finden sein sollen, kann man sich entweder die 13.000 Zeichen lange Installationsanleitung für die große, eigenständige Software MediaWiki (auf der z. B. auch Wikipedia basiert) durchlesen, sich einen eigenen Server mieten, eine Domain kaufen und sich mit Datenbanken auseinandersetzen, oder man aktiviert in den Repository-Settings des GitHub-Projektes die eingebaute Wiki-Funktion und kann mit der Doku loslegen. Diese Wiki-Lösung kann genau das Gleiche wie spezialisierte, große Standalone-Dokumentations-Softwares, ist aber kinderleicht einzurichten und zu konfigurieren. Man muss nicht installieren oder groß konfigurieren.

In einem Editor mit grafischer Benutzeroberfläche kann man direkt im Browser seine Wikiseiten anlegen und mit Inhalt füllen. Standardmäßig ist das Wiki für ein neues Repository auf GitHub aktiviert. Man greift über die Wiki-Schaltfläche in der Repository-Bedienleiste darauf zu. Sollte die Schaltfläche fehlen, kann man mit einem einzelnen gesetzten Haken unter „Settings“ und dort „Features“ das Wiki aktivieren.

Neben dem kompletten Aktivieren und Deaktivieren der Wiki-Funktion kann man in den Features noch festlegen, ob das Wiki auch von GitHub-Nutzern bearbeitet werden können soll, die keine Bearbeitungsrechte für das eigentliche Repository haben.

Klickt man bei einem selbst erstellten Repository in der Menüleiste auf die Wiki-Schaltfläche, bekommt man sofort eine Meldung, um eine neue Seite anzulegen.

Geht man hier auf „Create the first page“, wird – wie der Button es ankündigt – direkt die erste Wiki-Seite angelegt und in einem Editor zur Bearbeitung geöffnet. Dieser visuelle Editor erlaubt es Formatierungen (*kursiv*, **Fettdruck**, Aufzählungen, etc.) über Schaltflächen vorzunehmen.

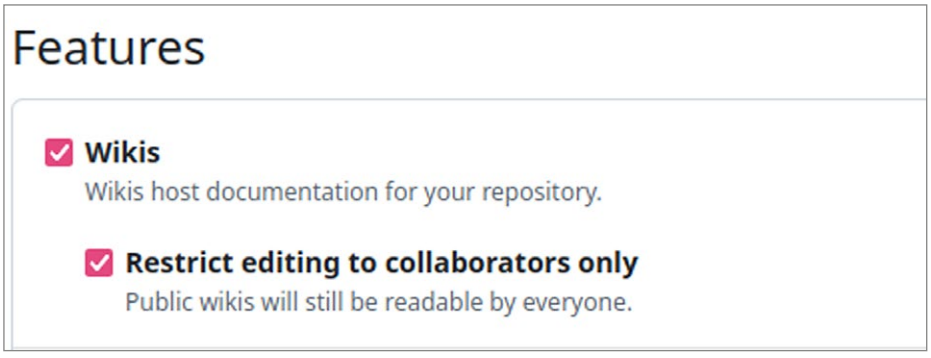
Der Wiki-Editor unterstützt beim Erstellen mit smarten Funktionen. Wenn man z. B. mithilfe der Markdown-Formatierung `[[Wiki-Seiten-Name]]` innerhalb des Wikis auf unterschiedliche Seiten verlinkt, aber es die entsprechende Seite aber noch nicht gibt, kann man sie direkt beim Erstellen der ersten Verlinkung anlegen.

Auch Bilder lassen sich direkt über einen Link mit dem Befehl `![Alt-Text](URL)` einbinden. Das können Bilder von anderen Websites oder aus dem Repository sein, zu dem das Wiki gehört.

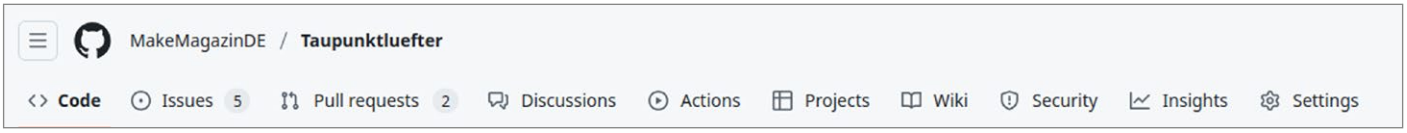
Das Layout des Wikis besteht zu Beginn aus folgenden Elementen: Auf der rechten Seite unter „Pages“ werden alle angelegten Wiki-Seiten in einer Art Inhaltsverzeichnis aufgelistet. Links daneben steht der Inhalt der aktuell ausgewählten Wiki-Seite (im Beispielbild „Welcome to the wiki“). Darüber befindet sich der Seiten-Name. Dieser wird auch in der Pages-Leiste am Rand angezeigt.

Die beiden Schaltflächen „+ Add a custom footer“ und „+ Add a custom sidebar“ erzeugen statische Elemente, die später jeweils unter oder rechts neben jeder Wikiseite angezeigt werden, ohne dass deren Inhalt manuell für jede neu erstellte Seite hinzugefügt werden muss. Die Inhalte für diese Elemente werden ebenfalls über den visuellen Editor bearbeitet.

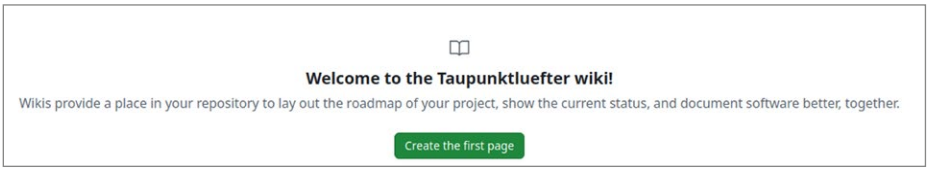
Eine tolle Funktion von GitHub-Wikis ist, dass es selbst ein Git-Repository ist. Das bedeutet, dass man es klonen, lokal bearbeiten, Änderungen committen und wieder hochladen kann – ganz genau so, wie man es mit Code macht. Jede Änderung wird versioniert; man kann sehen, wer wann was geändert hat, Seiten zurücksetzen und Unterschiede



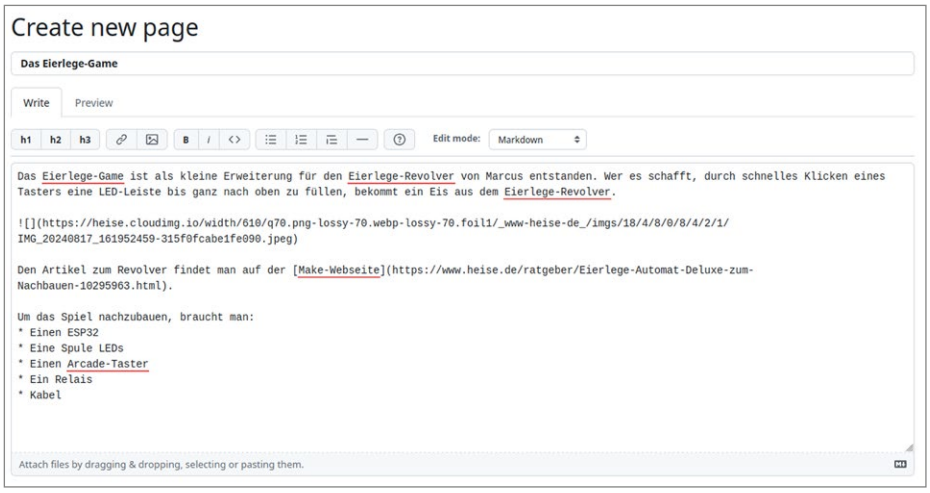
Wikis können pro Repository ein- und ausgestellt werden.



In der Navigationsleiste des Repositories findet sich das Wiki.



Ein Klick entfernt vom Wiki, im Browser geht das alles unkompliziert.



Direkt im Browser hat man einen vollumfänglichen, visuellen Editor für Wiki-Seiten.

vergleichen. Aber das ist vollkommen optional, denn für die meisten Projekte reicht es völlig aus, das Wiki einfach in dem Browser zu pflegen.

Issues

Issues (auf Deutsch: Probleme) sind ein Mehrzweckwerkzeug auf GitHub. Ein Issue ist ein forumartiger Beitrag mit Überschrift und Inhalt (Text und Bild), unter dem wiederum andere Nutzer kommentieren können. Issues sind ein Ort, an dem Code, Diskussion, Organisation und Ideen zu dem Projekt zusammenlaufen. Und das alles sehr einsteigerfreundlich: Man klickt einfach auf „New issue“, gibt einen Titel ein, schreibt sein Anliegen und klickt auf „posten“ – fertig.

Jedes Repository kann einen Issues-Reiter haben, wenn man unter „Setting/Features“ den Haken setzt, um sie zu aktivieren. Dann können zu einem Repository vom Besitzer und von fremden Nutzern Issues angelegt werden.

Wie diese forumartigen Strukturen dann genutzt werden, hängt vom Projekt ab:

- Als normales Forum: Nutzer können zu einer Software Fragen stellen oder Wünsche für zukünftige neue Features äußern. Dafür gibt es in dem Repository eigentlich auch GitHub-Discussions (Diskussionen), aber diese Funktion wurde erst 2020 eingeführt. Deshalb haben sich Issues als Diskussionsorte durchgesetzt.
- Aufgabenüberwachung: Ein Issue kann zu einem spezifischen, geplanten Feature-Update des Projektes angelegt werden. In ihm wird dann der Status der Entwicklung dokumentiert. Viele Teams nutzen Checklisten, Screenshots und automatische Verknüpfungen mit Commits, um den Fortschritt transparent sichtbar zu machen – ganz ohne ein zusätzliches Tool.
- Wissensdatenbank: Einige Projekte nutzen Issues auch als Wissensdatenbank, in der Nutzer des GitHub-Projektes Informationen über die Kompatibilität der Software etc. sammeln. Manche Repositories organisieren sogar ihre komplette Projekthistorie ausschließlich über Issues – inklusive Entscheidungen, Feedback und Dokumentation.

Neben selbst formuliertem Text gibt es im Issue-Editor auf der rechten Seite noch einige Optionen, welche über Drop-down festgelegt werden können.

„Assignees“, „Projects“ sowie „Milestone“ sind für die Issue-Verwendung als Aufgaben-Tracker vorgesehen. Mit ihnen kann man eine bestimmte Aufgabe definieren, welche dann einem Mitarbeiter des Repositories, einem Projekt und einem Meilenstein zugewiesen wird.

Die Option „Labels“ ist für alle möglichen Verwendungszwecke von Issues relevant. Damit legt man fest, was in dem Issue besprochen wird: ob man beispielsweise einen Bug gefunden hat (Label-Bug) oder ob man Hilfe bei der Verwendung des Projekt-Codes braucht (Label „Help wanted“). Labels helfen nicht nur bei der Übersicht – sie machen das gesamte Projekt filterbar und durchsuchbar.

„Type“ ist nur für GitHub-Organisationen relevant. Für mehr Informationen siehe Kasten „GitHub-Organisationen“. Hat man alles ausgefüllt, kann man unten rechts auf „Create“ klicken und ein Issue wird erstellt.

Möchte man in einem Issue auf ein anderes Issue, einen Pull-Request, Kommentar oder Commit eingehen, kann man im Editor mit `#<Nummer>` auf das entsprechende Element verweisen. Wenn man das # eintippt, bekommt man automatisch Vorschläge für verlinkte Datensätze angeboten.

Wiki-Formatierungen

Im Wiki-Editor kann man über ein Drop-down-Menü verschiedene Formatierungsarten für das Wiki auswählen.

Standardmäßig ist Markdown eingestellt, eine der gängigsten Sprachen, um Text direkt mit Befehlen im Text zu formatieren.

Auf den Editor hat die Wahl der Formatierung keine Auswirkungen. Er sieht immer gleich aus. Es ändert sich nur, wie die Dateien für das Wiki gespeichert werden.

Für besondere Anwendungen gibt es spezielle Formatierungen.

Websites auf GitHub hosten

Jetzt hat man ein cooles Projekt auf GitHub veröffentlicht und hätte gerne noch eine Website dazu, braucht dafür aber eigentlich noch einen Webserver und ein geeignetes Tool zum Erstellen der eigentlichen Webseite. Glücklicherweise kann man auf GitHub mit nur einer Datei direkt aus dem Repository heraus eine ansprechende, statische Website hosten.

Diese GitHub-Funktion heißt Pages. Sie ermöglicht das Website-Hosting über einen Commit in unter fünf Minuten – kostenlos, ohne sich bei irgendeinem Hosting-Dienst anmelden zu müssen und ohne Set-up. Zwei Dateien, ein Commit und live – einfacher geht es wirklich nicht.

Eine statische Website besteht meist aus einer HTML- und CSS-Datei und eventuell etwas JavaScript. So eine Seite lässt sich nicht dynamisch anpassen und lädt auch keine Informationen aus einer Datenbank. Man kann sich das wie ein gedrucktes Plakat vorstellen: Es sieht für alle gleich aus, und wenn man etwas ändern möchte, muss man den Code der Website direkt bearbeiten und neu hochladen.

Statische Inhalte für alle Wiki-Seiten können in ein Seitenelement oder einen Footer geschrieben werden.

Im Gegensatz dazu steht eine Website, welche mit einem Content-Management-System (CMS) wie WordPress oder Joomla betrieben wird. Es ermöglicht, Inhalte wie Texte oder Bilder direkt im Browser zu verändern, ohne dass man den Code anfassen muss. Man meldet sich einfach in einem Adminbereich an, nimmt Änderungen vor und speichert sie – das System kümmert sich um den Rest. So eine Website ist flexibler und leichter zu pflegen, besonders wenn sich Inhalte regelmäßig ändern oder wenn mehrere Personen daran arbeiten sollen.

Der große Vorteil statischer Websites liegt in ihrer Schnelligkeit und Sicherheit. Sie laden oft blitzschnell, weil der Server keine Datenbankabfragen machen muss, und es gibt weniger Angriffsflächen, weil keine komplexe Software im Hintergrund läuft. Dafür sind sie weniger praktisch, wenn Inhalte oft aktualisiert werden sollen. Welche Lösung besser ist, hängt also immer vom Zweck der Website und den eigenen Bedürfnissen ab.

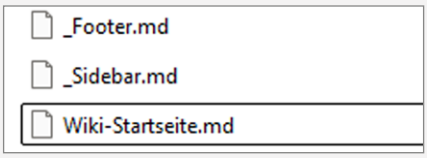
Es gibt zwei Möglichkeiten, um sich eine Website auf GitHub einzurichten. Die simpelste ist, direkt HTML-Dateien zu erstellen und hochzuladen. Dafür braucht man mindestens eine index.html-Datei. Index bedeutet, dass das die erste Seite ist, die beim Aufruf der Domain geöffnet wird. Von dieser Seite aus kann man dann natürlich auf weitere HTML-Dateien verlinken. Folgend der Minimalcode für eine funktionierende Website:

Wiki als Repository

Wenn man das Wiki wie ein Repository auf dem lokalen Computer bearbeiten will, benötigt man einen speziellen Klon-Link, der auf dem Wiki unten rechts angezeigt wird (wenn man die Berechtigung zum Bearbeiten hat). Dieser Link wird wie folgt gebildet: `https://github.com/<Nutzername>/<Repository-Name>.wiki.git`. Wenn man über diesen Link das Wiki klont, kann man die Dateien (deren Formatierung man frei wählen kann, siehe Kasten „Wiki-Formatierung“) in einem Texteditor der Wahl lokal auf dem Computer bearbeiten. Dann kann man wie bei einem regulären Repository einen Commit erstellen und ins Wiki-Repository pushen.

```
git clone https://github.com/<Nutzername>/<Repository-Name>.wiki.git
git add *
git commit -m „Mein Wiki-Commit“
git push
```

Wenn man lokal neue Wiki-Seiten anlegt und diese Änderungen pusht, werden sie automatisch in die Bedienoberfläche des Wikis (z. B. als Eintrag in die „Pages“-Seite) eingebunden. Änderungen, die in diesem Wiki-Repository gemacht werden, haben keine Auswirkungen auf das eigentliche Projekt-Repository.



Klont man das Wiki-Repository auf dem Computer, sind die Wiki-Seiten einfache Textdateien.

```
<!DOCTYPE html>
<html>
<head>
  <title>Titel</title>
</head>
<body>
  Hallo Welt!
</body>
</html>
```

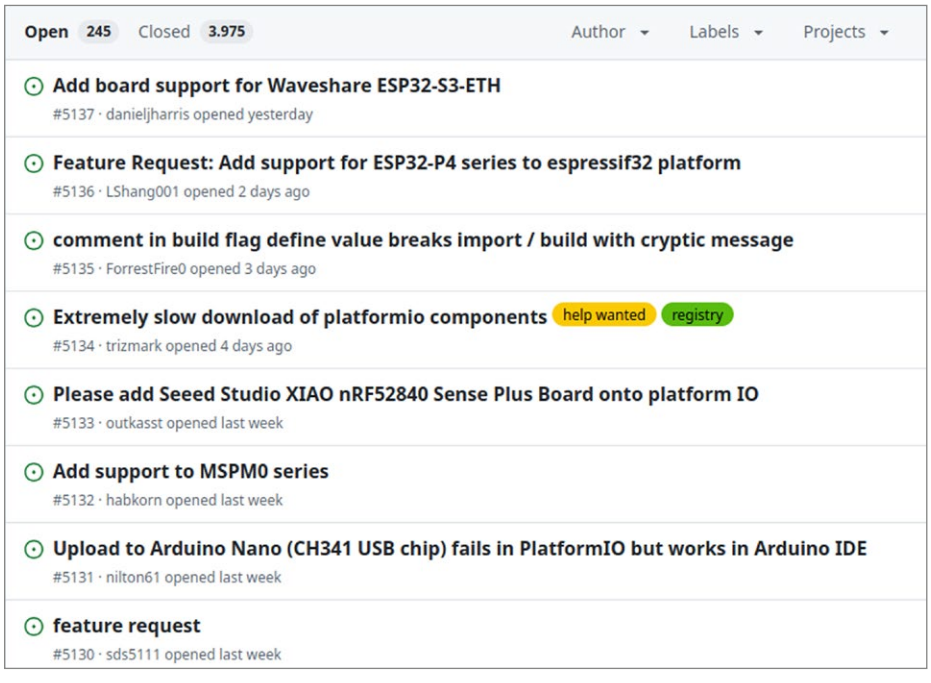
Damit man sich nicht gleich das ganze Repository zumüllt, sollte man für die Website einen eigenen Branch erstellen (wie das geht, steht in Make 1/25, S. 54).

Hat man eine index.html hochgeladen, kann man in den Repository-Einstellungen unter „Pages“ die Website aktivieren. Die Option „Source“ ist standardmäßig auf „Deploy from branch“ eingestellt. Das lässt man so. Darunter kann man dann den Branch, in den man die index.html hochgeladen hat, auswählen und auf „Save“ klicken. Jetzt ist die Website auf `https://dein-username.github.io/<Repository-name>/` erreichbar.

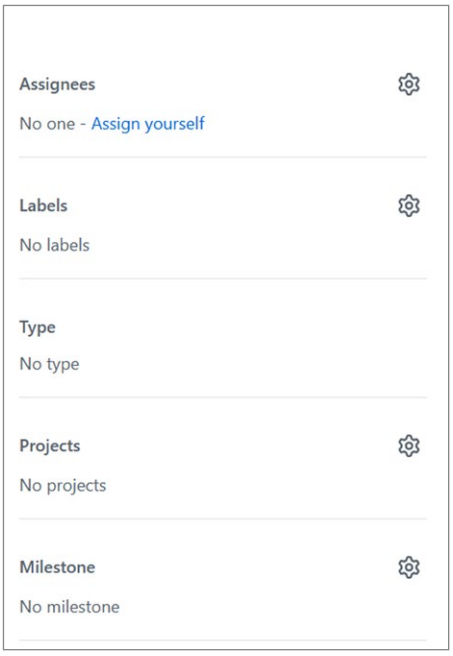
Wenn man eine umfangreichere Website hosten möchte, die mehr Inhalt hat, sich einfacher erweitern lässt oder man sich nicht mit HTML auseinandersetzen möchte, kann man den in GitHub eingebauten „Static Site Generator“ Jekyll benutzen. Dieses Programm baut nach einer einmaligen Konfiguration aus unformatiertem Text eine statische Website zusammen. Man kann sich dann komplett auf das Erstellen von Inhalten beschränken.

Der in GitHub-Pages eingebaute SSG Jekyll braucht dafür nur eine ausgefüllte Konfigurationsdatei _config.yml und Inhalte im MD-Format (Markdown). Diese Dateien kann man dann einfach in das Grundverzeichnis eines neuen Branches hochladen (auch direkt im Browser möglich) und unter „Settings/Pages“ für diesen Extra-Branch in /root die Pages-Funktion aktivieren. Dann geschieht das Bereitstellen der Webseite und das Generieren der HTML-Dateien automatisch.

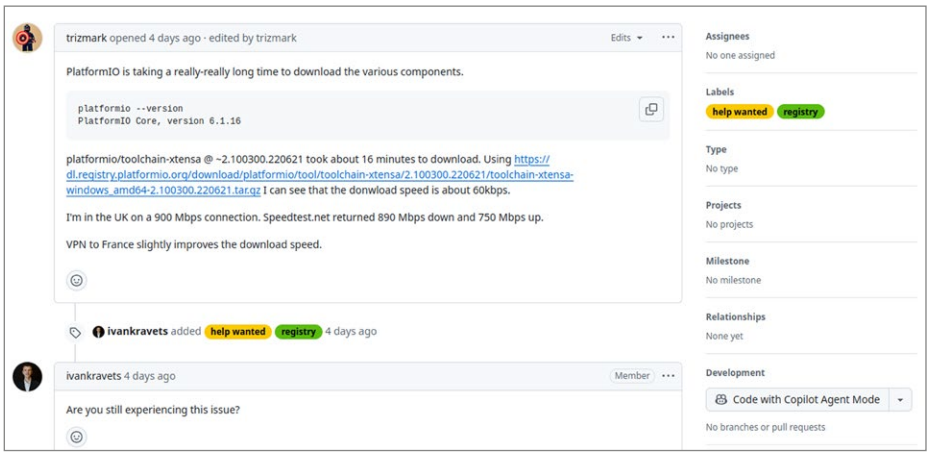
Auf diese Weise eine attraktive Website auf die Beine zu stellen, ist so einfach, dass die komplette Konfiguration für die GitHub-Pages des Eierlege-Games (Make Ausgabe 2/25) hier abgedruckt werden kann (siehe Kasten „Einfache Jekyll-Konfiguration“).



Die Issues für das Projekt platformio-core zeigen alle Möglichkeiten, was man mit Issues machen kann.



Für die Einordnung eines Issues gibt es verschiedene Einstellungsmöglichkeiten.

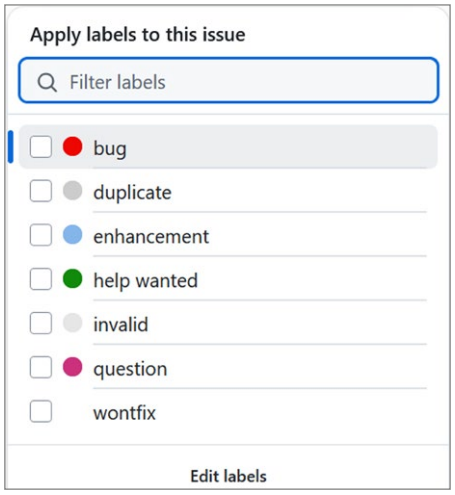


Ein Issue, versehen mit Labels, einer Beschreibung und der ersten Antwort.

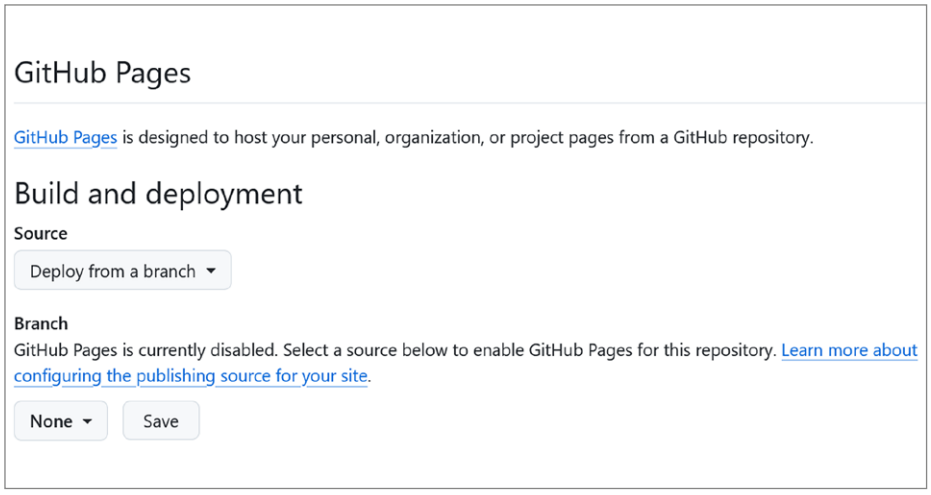
GitHub-Pages wirkt auf den ersten Blick wie ein kleines Extra – in Wirklichkeit ist es aber ein vollwertiges Publishing-Tool, das direkt im Entwicklungsprozess mitläuft. Ohne Hosting-Kosten, ohne komplizierte Set-ups, ohne externe Werkzeuge. Wer ein GitHub-Repository hat, kann auch in wenigen Minuten eine öffentlich zugängliche, gepflegte Projektwebsite veröffentlichen – technisch sauber, schnell, sicher und komplett integriert. Gerade in der Kombination mit Markdown und Jekyll ist das ein überraschend mächtiges Werkzeug,

GitHub-Organisationen

Eine GitHub-Organisation ist eine Art gemeinsames Konto, unter dem mehrere Personen zusammen an Projekten arbeiten können. Sie eignet sich z. B. für Teams, Firmen oder Gruppen. In einer Organisation können beliebig viele Repositories (Projekte) erstellt und verwaltet werden. Man kann Mitgliedern unterschiedliche Rollen geben, etwa als Entwickler oder Admin, und so die Zusammenarbeit gut strukturieren und kontrollieren. Man nutzt eine Organisation, um gemeinsam Code zu speichern, Änderungen zu verfolgen und Aufgaben zu organisieren – alles an einem zentralen Ort. Erstellen kann man sie direkt auf GitHub unter dem Punkt „Your organizations“.



Labels sind für grundlegende Einordnung schon vordefiniert.



In den Einstellungen des Repositories muss man nur auswählen, von wo eine Website erstellt werden soll und los geht es.

Einfache Jekyll-Konfiguration

Um aus Markdown eine HTML zu erstellen, zieht sich Jekyll folgende Informationen aus der _config.yaml-Datei:

- **title**: Der Titel der Website. Dieser kann frei gewählt werden.
- **description**: Beschreibt die Website, kann auch frei eingetragen werden.
- **baseurl**: Ergibt sich aus dem Namen des Repositories, wird hinter die URL angehängt.
- **url**: Ergibt sich aus dem Namen des GitHub-Kontos.
- **theme**: Das hier ist der wichtigste Punkt. Hier wird eingetragen, mit welcher Vorlage aus den Markdown-Dateien eine Website gebaut werden soll. In der Kurzinfo ist eine Liste mit verfügbaren Themes verlinkt.

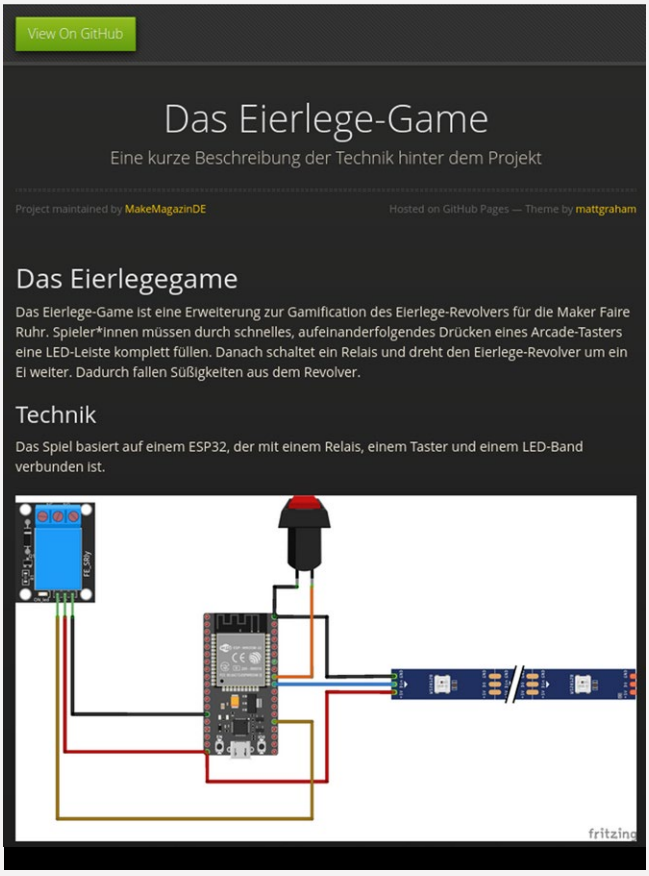
```
title: Das Eierlege-Game
description: Eine kurze Beschreibung..
baseurl: "/Eierlege-Game"
url: "https://makemagazinde.github.io"
theme: jekyll-theme-midnight
markdown: kramdown
```

Jetzt brauchen wir noch den Website-Inhalt. Die Konfiguration der index.md-Datei ist sehr übersichtlich.

- **title**: Der Titel der Seite wird oben im Browser angezeigt.
- **layout**: Bestimmt, welche Vorlage aus einem Theme für diese Seite genutzt werden soll. Der Parameter `default` gibt an, dass keine spezielle Formatierung verwendet werden soll.
- Danach kann man einfach den Inhalt im Klartext schreiben.

```
---
title: Das Eierlege-Game
layout: default
---
# Das Eierlege-Game
Das Eierlege-Game ist eine
Erweiterung zur Gamification....
```

Für jede weitere Seite braucht man eine eigene MD-Datei.



Und mit nur diesen beiden Dateien hat man direkt eine ansprechende Website gebaut.

um Inhalte sichtbar zu machen, egal ob es um Dokumentation, Portfolios oder kleine Web-Experimente geht.

Releases

Über das Releases-System lassen sich bestimmte Versionen eines Projekts markieren, mit zusätzlichen Dateien versehen und öffentlich als offizieller Download bereitstellen. So kann man anderen Nutzern einen stabilen, getesteten Stand zur Verfügung stellen – zum Beispiel eine fertige Firmware, eine ZIP-Datei mit allen Sketches oder eine ausführbare Version eines Tools.

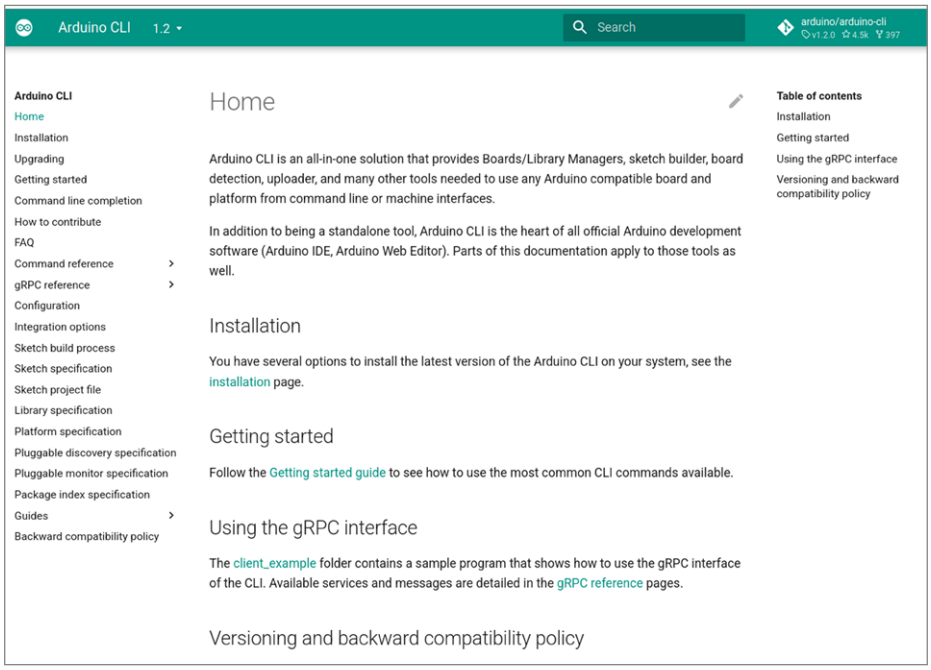
Um einen Release auf GitHub anzulegen, öffnet man zunächst das Repository, in dem die Veröffentlichung erfolgen soll. In der Menüleiste direkt unter dem Repository-Namen befindet sich der Punkt „Releases" – ein Klick darauf führt zur Übersicht aller bisherigen Versionen. Ist noch kein Release vorhanden, klickt man auf „Draft a new release", um eine neue Version anzulegen. Im sich öffnenden Formular gibt man als Erstes einen sogenannten Tag-Namen ein, zum Beispiel v1.0. Dieser Tag markiert den Stand des Codes, der veröffentlicht wird. Wenn der Tag noch nicht existiert, wird er automatisch neu erstellt. Danach wählt man den Branch aus, auf dem der Release basieren soll – in den meisten Fällen ist das „main" oder „master".

Anschließend vergibt man einen Titel für den Release, etwa „Version 1.0 – Neue Funktion XY". Darunter kann man in einem größeren Textfeld optional eine Beschreibung ergänzen, zum Beispiel eine kurze Zusammenfassung der Änderungen oder Hinweise zur Nutzung. Die Beschreibung unterstützt auch Markdown – man kann also Listen, Links oder Hervorhebungen einbauen. Falls man zusätzlich zum Quellcode noch Binärdateien anbieten möchte, etwa eine ZIP- oder HEX-Datei oder ein PDF, kann man diese einfach per Drag & Drop ins Formular ziehen oder über die Schaltfläche „Attach binaries" hinzufügen.

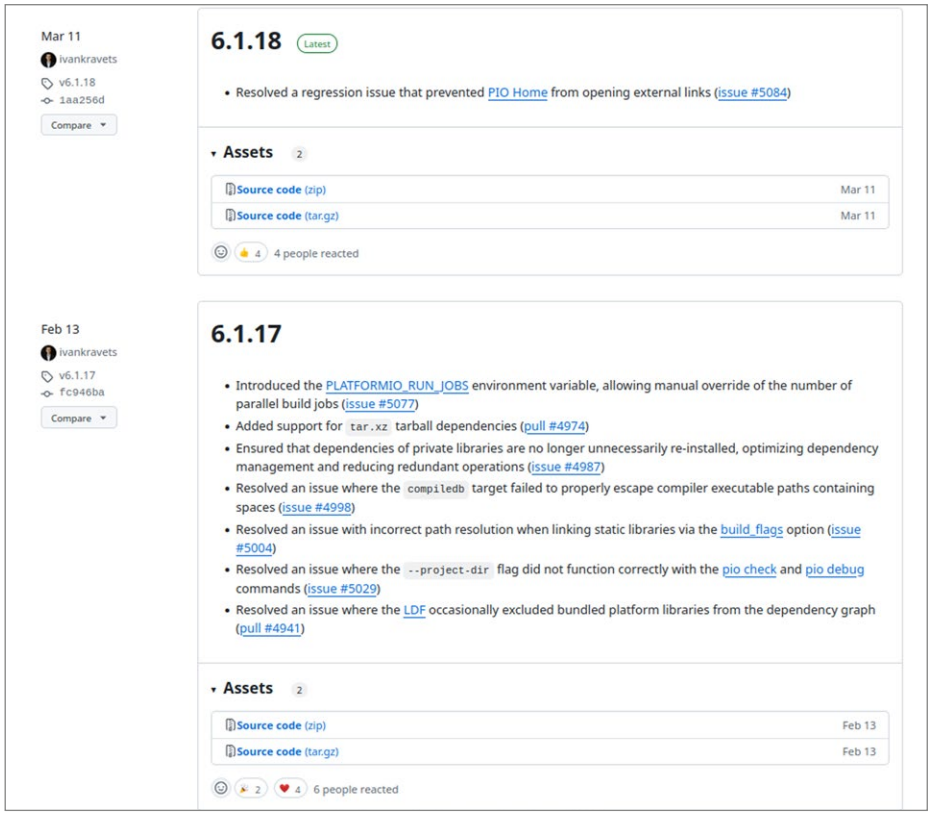
Wer den Release nicht als endgültige Version, sondern als eine Testversion markieren möchte – beispielsweise für eine Beta – kann ganz unten das Häkchen bei „This is a pre-release" setzen. Ist alles ausgefüllt, klickt man auf „Publish release" und sofort steht die neue Version öffentlich zur Verfügung, inklusive aller angehängten Dateien, mit dauerhaftem Download-Link und automatisch generierter Release-Seite.

Ein großer Vorteil: Releases sind unveränderlich. Sie frieren den Codezustand zum Zeitpunkt der Veröffentlichung ein – genau das, was man für Reproduzierbarkeit oder langfristige Nutzung braucht. Wer sich also „Version 1.0" eines Projekts herunterlädt, bekommt exakt das, was zu diesem Zeitpunkt online war – auch wenn das Repository sich später weiterentwickelt.

GitHub-Releases eignen sich perfekt, um ein Projekt professionell und zugänglich zu präsentieren – gerade für Maker, die nicht nur entwickeln, sondern ihre Projekte auch veröffentlichen und weitergeben wollen. Ohne zusätzlichen Webspace, ohne Versionschaos, mit direktem Download-Link und automatischer Versionierung. Einfach einen Tag setzen, Beschreibung reinschreiben, Datei hochladen – fertig ist das offizielle Release. *—das*



Ein Beispiel für ein Maker-relevantes Projekt, das GitHub-Pages benutzt.



Releases können eine Zusammenfassung der Neuerungen im Vergleich zum letzten Mal beinhalten und natürlich die Release-Datei an sich (Assets genannt).