HUBZILLA

als Social Media CMS verwenden

Ein Leitfaden

Inhalt

Einleitung - Was ist Hubzilla?

Hubzilla ist eine ausgeklügelte und einzigartige Kombination aus einem Open-Source Content Management System und einem dezentralen Identitäts-, Kommunikations- und Berechtigungs-Framework sowie einer Protokollsuite, die unter Verwendung gängiger Webserver-Technologie entwickelt wurde. Das Endergebnis ist ein Maß an Systemintegration, Datenschutzkontrolle und Kommunikationsfunktionen, das Sie weder in einem Content-Management-System noch in einem dezentralisierten Kommunikationsnetzwerk für möglich gehalten hätten. Es bringt auch ein neues Maß an Zusammenarbeit und Datenschutz ins Web und führt das Konzept des persönlich verwalteten „Single Sign-On" für Webdienste im gesamten Internet ein.

Hubzilla kann als normale Zugangssoftware für die Teilnahme am Fediverse dienen und wird oft auch genau dafür verwendet. Die weit darüber hinausgehenden Funktionen und Möglichkeiten sind vielen nicht bekannt. Hubzilla ist ein vollwertiges Content Management System und damit durchaus geeignet umfangreiche Web-Präsenzen zu erstellen, die in Qualität und Funktion mit den Webseiten anderer CMS mithalten können.

Weil viele Funktionen nicht allgemein bekannt sind und teilweise auch eine andere Herangehensweise erfordern, als es bei den bekannten CMS der Fall ist, gibt es nun dieses Buch, welches reinen Nutzern vermittelt, wie sie Hubzilla für eben diese Zwecke verwenden können.

Für das Verständnis werden Grundkenntnisse über Hubuzilla, wie etwa das Installieren von Apps und das Berechtigungssystem vorausgesetzt. Wer hier noch Defizite hat, sollte sich die entsprechenden Artikel in der Hubzilla KnowledgeDB durchlesen.

Die Cloud

Obwohl Cloud-Funktionalität nicht unbedingt zu den Kern-Funktionen eines CMS gehören, soll diese hier gleich zu Beginn vorgestellt werden.

Hubzilla bietet Cloud-Funktionalität. Das bedeutet, dass jeder Kanal über ein Dateiverzeichnis verfügt, in dem weitere Unterverzeichnisse erzeugt und Dateien abgelegt werden können. Für jedes Verzeichnis, ja sogar für jede einzelne Datei können genaue Zugriffsrechte festgelegt werden. Das geht von einer Sichtbarkeit für die Allgemeinheit, eine Sichtbarkeit für Mitglieder bestimmter Gruppen bis hin zur einzelnen Freigabe für einzelne Mitglieder aus den eigenen Verbindungen. Es ist sogar möglich, Dateien mit Nutzern zu sharen, die keine Hubzilla-Identität haben. Dazu verwendet man Gastzugangs-Token.

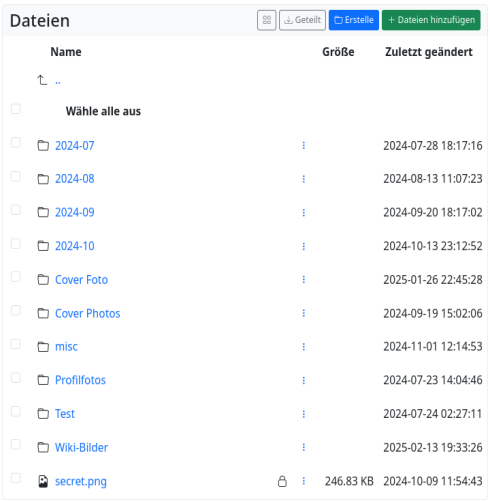
Die Verwaltung von Dateien und Verzeichnissen kann auf zwei Arten erfolgen.

Dateiverwaltung mit der Weboberfläche

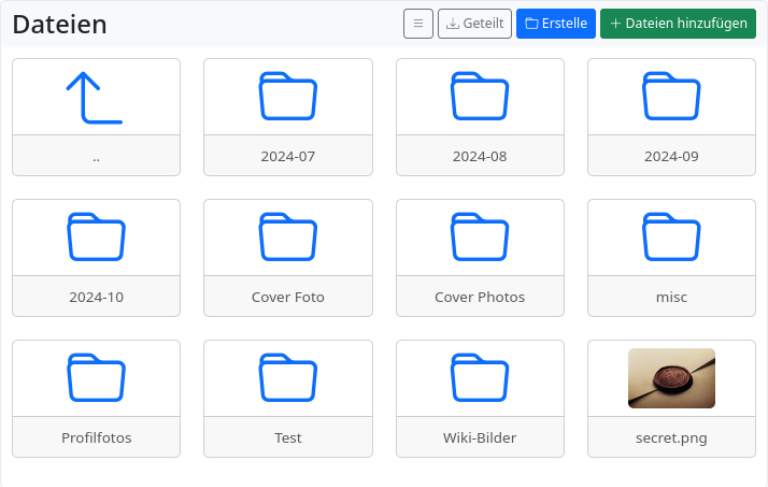
Einmal besteht die recht einfache Möglichkeit, die mit der Weboberfläche von Hubzilla zu erledigen. Die Oberfläche der Cloud ist über das App-Menü und die App „Dateien“ zu erreichen.

Die App zeigt nach Aufruf das Wurzelverzeichnis der eigenen Cloud. In der Weboberfläche kann man nun durch einfaches Klicken navigieren, also Ordner öffnen und betreten und sich bei der Auswahl einer einzelnen Datei diese anzeigen lassen (sofern der Webbrowser eine Voransicht für das jeweilige Dateiformat bietet) oder auf den eigenen Rechner herunterladen.

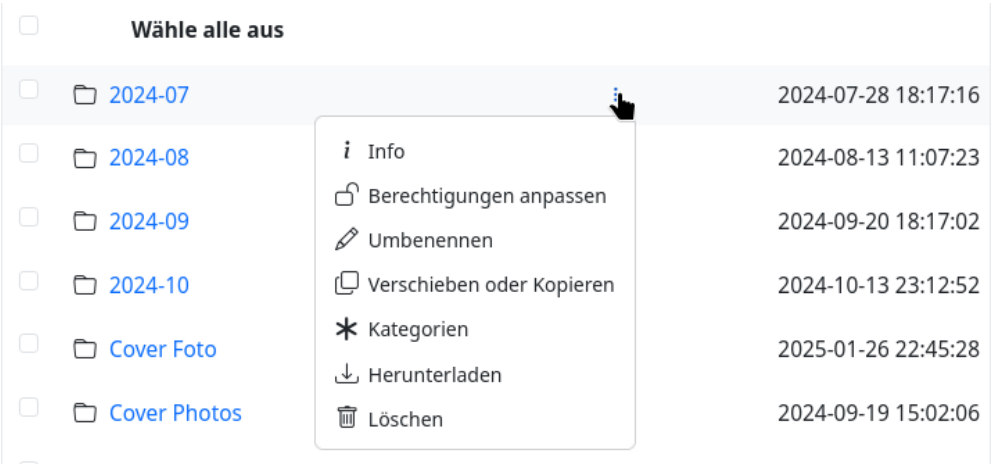
Es gibt zwei unterschiedliche Arten der Ansicht. Einmal die Listenansicht, welche auch Dateioperationen erlaubt,



oder in der Symbolansicht, die zwar mehr nach "Desktop" ausschaut, aber keine Dateioperationen zur Verfügung stellt.



Die Dateioperationen welche man über das Datei-/Ordnermenü (" ") erreicht, ermöglichen die Anzeige eines Datei-Infos, das Anpassen der Datei- bzw. Verzeichnisberechtigungen, Umbenennen, Verschieben oder Kopieren, das Zuweisen von Dateien bzw. Verzeichnissen zu einer Kategorie, das Herunterladen einer Datei bzw. eines Verzeichnisses, sowie das Löschen.



Dem Anpassen der Berechtigungen kommt eine besondere Bedeutung zu, weil man damit steuern kann, wer die Dateien bzw. Ordner sehen kann und wer auf sie zugreifen darf. Bilder und Dateien, die auch für Besucher sichtbar sein sollen, müssen zwingend die Berechtigung "Öffentlich" bekommen. Das gilt auch zu beachten, falls man Web-Frameworks für Webseiten in seiner Cloud selbst hosten möchte.

Dateiverwaltung mittels WebDAV

Wesentlich komfortabler und empfohlen ist die Dateiverwaltung mittels WebDAV. Dieses Verfahren erlaubt den Up- und Download über spezielle Client-Apps, vor allem aber auch das Einbinden in das Dateisystem des eigenen Endgerätes. Mit WebDAV kann man die Cloud so in das Dateisystem des eigenen Rechners einbinden, dass es sich verhält, als würden die Dateien lokal auf dem eigenen Rechner vorliegen. Auf diese Weise kann man Dateien schnell und einfach mit den bevorzugten Mitteln kopieren, verschieben, umbenennen und sogar bearbeiten.

Berechtigungen

Bei Verwendung von WebDAV wird die Datei mit den Standard-Dateiberechtigungen Ihres Kanals erstellt, die innerhalb des Betriebssystems nicht geändert werden können. Möglicherweise sind diese auch nicht so restriktiv, wie Sie es gerne hätten. Die bevorzugte Methode, um Dateien privat zu machen, besteht darin, zunächst Ordner oder Verzeichnisse zu erstellen, dann die Cloud aufzurufen, das Verzeichnis auszuwählen und die Berechtigungen zu ändern. Tun Sie dies, bevor Sie etwas in das Verzeichnis legen. Die Verzeichnisberechtigungen haben Vorrang, sodass Sie dann Dateien oder andere Ordner in diesen Container legen können und diese durch die Verzeichnisberechtigungen vor unerwünschten Betrachtern geschützt sind. Es ist üblich, einen „persönlichen” oder „privaten” Ordner zu erstellen, der nur für Sie selbst zugänglich ist. Sie können diesen als persönliche Cloud verwenden, um alles aus dem Internet oder von jedem Computer aus zu speichern, und er ist vor anderen geschützt. Sie können auch Ordner für „Familie” und „Freunde” erstellen und den entsprechenden Datenschutzgruppen Berechtigungen erteilen.

Cloud-Desktop-Clients

Cloud-Desktop-Clients – Windows

WebDAV mit dem Assistenten der grafischen Benutzeroberfläche von Windows 10/11:

1. Öffnen Sie den Datei-Explorer.
2. Rechtsklick auf "Dieser PC" "Netzlaufwerk verbinden..."
3. Wählen Sie einen Laufwerksbuchstaben.
4. Unter "Ordner" geben Sie die DAV-URL ein (z. B. https://example.com/remote.php/dav/files/benutzer).
5. Setzen Sie Haken bei "Verbindung mit anderen Anmeldeinformationen herstellen".
6. Klicken Sie auf "Fertig stellen".
7. Geben Sie Benutzernamen und Passwort ein (ohne "@").
8. Optional: Haken bei "Anmeldeinformationen speichern", wenn gewünscht.

Cloud-Desktop-Clients – Linux

Als Dateisystem einbinden

Um Ihr Cloud-Verzeichnis als Dateisystem zu installieren, müssen Sie zunächst davfs2 installieren. In 99 % der Fälle ist dies in den Repositorys Ihrer Distribution enthalten. In Debian installieren sie es mit

```
apt-get install davfs2
```

Unter Arch Linux erfolgt die Installation mit

```
pacman –S davfs2
```

Wenn Sie normalen Benutzern das Mounten des Dateisystems erlauben möchten, rufen Sie unter Debian

```
dpkg-reconfigure davfs2
```

auf und wählen Sie „Ja“ bei der Eingabeaufforderung.

Jetzt müssen Sie alle Benutzer, die dav mounten sollen, zur Gruppe davfs2 hinzufügen:

```
usermod –aG davfs2 <DesktopUser>
```

Hinweis: Auf einigen Systemen kann die Benutzergruppe unterschiedlich sein, z. B. „network“ unter Arch Linux. Wenn Sie sich nicht sicher sind, lesen Sie die davfs-Dokumentation für Ihr Betriebssystem.

Bearbeiten Sie /etc/fstab

`nano /etc/fstab`,

um Ihr Cloud-Verzeichnis hinzuzufügen, indem Sie folgendes hinzufügen

```
<domain_name>/dav/ /mount/point davfs user,noauto,uid=<DesktopU-
ser>,file_mode=600,dir_mode=700 0 1
```

wobei `<domain_name>` die URL Ihres Hubs ist, `/mount/point` der Ort, an dem Sie die Cloud mounten möchten, und `<DesktopUser>` der Benutzer, mit dem Sie sich an Ihrem Computer anmelden. Beachten Sie, dass sich der Mount-Punkt in Ihrem Home-Verzeichnis befinden muss, wenn Sie als normaler Benutzer (nicht als Root) mounten.

Wenn ich beispielsweise meine Cloud in einem Verzeichnis namens „cloud” in meinem Home-Verzeichnis mounten möchte und mein Benutzername „bob” lautet, würde meine fstab wie folgt aussehen

```
https://<domain_name>/dav/ /home/bob/cloud davfs user,noau-
to,uid=bob,file_mode=600,dir_mode=700 0 1
```

Erstellen Sie nun den Mount-Punkt.

```
mkdir /home/bob/cloud
```

Erstellen Sie außerdem ein Verzeichnis, in dem Sie Ihre Anmeldedaten speichern können

```
mkdir /home/bob/.davfs2
```

Erstellen Sie eine Datei namens „secrets"

```
nano /home/bob/.davfs2/secrets
```

und fügen Sie Ihre Cloud-Anmeldedaten hinzu

```
https://<domain_name>/dav <username> <password>
```

Dabei sind `<username>` und `<password>` der Benutzername und das Passwort *für Ihren Hub*.

Verhindern Sie, dass diese Datei von Personen geschrieben werden kann, die sie nicht benötigen, mit

```
chmod 600 /home/bob/.davfs2/secrets
```

Hängen Sie schließlich das Laufwerk ein.

```
mount https://<domain_name>/dav
```

Sie finden Ihre Cloud nun unter `/home/bob/cloud` und können sie wie einen Teil Ihres lokalen Dateisystems verwenden – auch wenn die von Ihnen verwendeten Anwendungen selbst keine DAV-Unterstützung bieten.

Fehlerbehebung

Bei einigen Webservern und bestimmten Konfigurationen kann es vorkommen, dass davfs2 Dateien mit einer Größe von 0 Byte erstellt, während andere Clients einwandfrei funktionieren. Dies wird in der Regel durch Cache und Sperren verursacht. Wenn Sie von diesem Problem betroffen sind, müssen Sie Ihre davfs2-Konfiguration bearbeiten.

```
nano /etc/davfs2/davfs2.conf
```

Ihre Distribution stellt eine Beispielkonfiguration zur Verfügung, und diese Datei sollte bereits vorhanden sein, jedoch ist der größte Teil davon mit einem # am Anfang der Zeile auskommentiert.

Der erste Schritt besteht darin, die Sperren zu entfernen.

Bearbeiten Sie die Zeile `use_locks` so, dass sie `use_locks 0` lautet.

Hängen Sie Ihr Dateisystem aus, hängen Sie es erneut ein und versuchen Sie, eine Datei über die Befehlszeile zu kopieren. Beachten Sie, dass Sie für diesen Test eine neue Datei kopieren und keine alte überschreiben sollten. Warten Sie ein oder zwei Minuten und führen Sie dann `ls -l -h` aus, um zu überprüfen, ob die Dateigröße Ihrer neuen Datei nun größer als 0 Byte ist. Wenn ja, hören Sie hier auf und unternehmen Sie nichts weiter.

Wenn das immer noch nicht funktioniert, deaktivieren Sie den Cache. Beachten Sie, dass dies Auswirkungen auf die Leistung hat und daher nur durchgeführt werden sollte, wenn das Deaktivieren der Sperren Ihr Problem nicht gelöst hat. Bearbeiten Sie die Zeile `cache_size` und setzen Sie sie auf `cache_size 0`. Setzen Sie außerdem `file_refresh` auf `file_refresh 0`. Hängen Sie Ihr Dateisystem aus, hängen Sie es wieder ein und testen Sie es erneut.

Wenn es *immer noch* nicht funktioniert, können Sie noch eine weitere Maßnahme ergreifen. (Dieses Problem wird durch einen Fehler in älteren Versionen von dav2fs selbst verursacht, sodass ein Update auf eine neue Version ebenfalls Abhilfe schaffen kann). Aktivieren Sie das Löschen schwacher ETAGs, indem Sie `drop_weak_etags 1` festlegen. Hängen Sie Ihr Dateisystem aus und hängen Sie es erneut ein, um die Änderungen zu übernehmen.

Dolphin

Rufen Sie `webdavs://example.com/dav` auf, wobei „example.com” die URL Ihres Hubs ist.

Wenn Sie zur Eingabe eines Benutzernamens und eines Passworts aufgefordert werden, geben Sie Ihren Kanalnamen (den ersten Teil Ihres Webbies – ohne @ oder Domainnamen) und das Passwort für Ihr normales Konto ein.

Nautilus

1. Öffnen Sie ein Dateibrowser-Fenster (das ist Nautilus).
2. Wählen Sie im Menü „Datei“ > „Mit Server verbinden“.
3. Geben Sie davs://<domain_name>/dav/<your_channelname> ein und klicken Sie auf „Verbinden“.
4. Sie werden zur Eingabe Ihres Kanalnamens (wie oben) und Ihres Passworts aufgefordert.
5. Ihr persönliches DAV-Verzeichnis wird im Fenster angezeigt.

Nemo

Für (Dateibrowser) Nemo 1.8.2 unter Linux Mint 15, Cinnamon 1.8.8. Nemo ist dort der Standard-Dateibrowser.

1. Möglichkeit

Geben Sie „`davs://<domain_name>/dav/<your_channelname>`" in die Adressleiste ein.

2. Möglichkeit

Menü > Datei > Mit Server verbinden

Füllen Sie das Dialogfeld aus

- Server: `hubzilla_domain_name`

- Typ: Sicheres WebDAV (https)

- Ordner: `/dav`

- Benutzername: `Ihr_Kanalname`

- Passwort: `IhrPasswort`

Sobald die Verbindung hergestellt ist, können Sie ein Lesezeichen setzen.

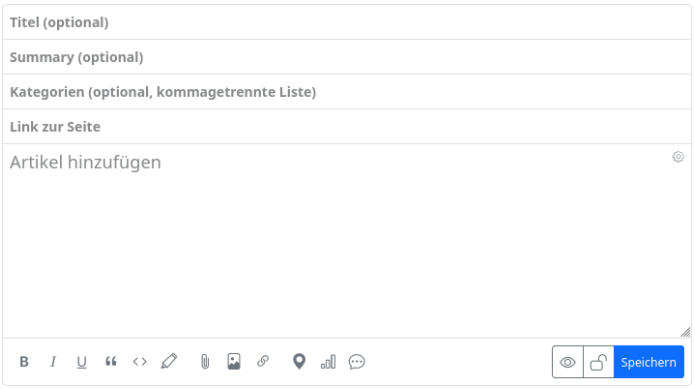
Artikel

Möchten Sie in Ihrer Web-Präsenz einen Blog-Bereich oder einen entsprechenden Bereich mit Einzelartikeln anbieten, so ist die App "Artikel" die richtige Wahl. Die App "Artikel" muss im Kanal zunächst installiert werden.

Diese App bietet Macroblogging-Fähigkeiten. Ein Artikel hat einen Titel, wenn gewünscht, eine Zusammenfassung und kann einer oder mehreren Kategorien zugeordnet werden. Jeder Artikel verfügt über eine Link-Adresse, die entweder von hubzilla automtisch erzeugt wird, oder von Ihnen festgelegt werden kann. Über diesen Link zur Seite ist der Artikel direkt adressierbar und kann unter

`<URL_DES_HUB>/articles/<KANALNAME>/<Link_zur_Seite>`

aufgerufen werden.



Die Textgestaltung erfolgt mit bbCode. Eine Übersicht über die Tags und Gestaltungsmöglichkeiten, finden Sie in der Hubzilla-Hilfe und in der Hubzilla KnowledgeDB.

Die Nutzung von Artikel-Kategorien ermöglicht die Strukturierung der Artikel. In der Artikelansicht ist in der Seitenleiste ein Widget vorhanden, in welchem die einzelnen Kategorien aufgelistet werden. Über diese Liste sind die Artikel einer bestimmten Kategorie filterbar.

Es existiert außerdem ein weiteres Widget, welches eine Kategorie-Cloud anzeigt, das man selbst hinzufügen muss.

Artikel erscheinen in chronologischer Reihenfolge, wobei nachträgliche Änderungen an einem Artikel diesen wieder in der Auflistung an die erste Stelle bringt (es gibt eine Möglichkeit, die Reihenfolge auf das ursprüngliche Erstellungsdatum zu erzwingen, die jedoch Administrator-Rechte erfordert).

Artikel werden nicht föderiert, sondern sind nur über den Kanal selbst abrufbar.

Karten

Die App "Karten"ist eng verwandt mit der App "Artikel". Auch diese App muss zuerst explizit in einem Kanal installiert werden. Sie unterscheidet sich in Hinsicht auf Erstellung und Gestaltung nicht von der App "Artikel". Sie bietet ebenfalls ein Artikel-Widget und Karten werden ebenfalls nicht föderiert.

Sie bietet sich an, wenn man einen weiteren, unabhängigen Artikel-Bereich auf seiner Web-Präsenz anbieten möchte.

Karten-Artikel sind über

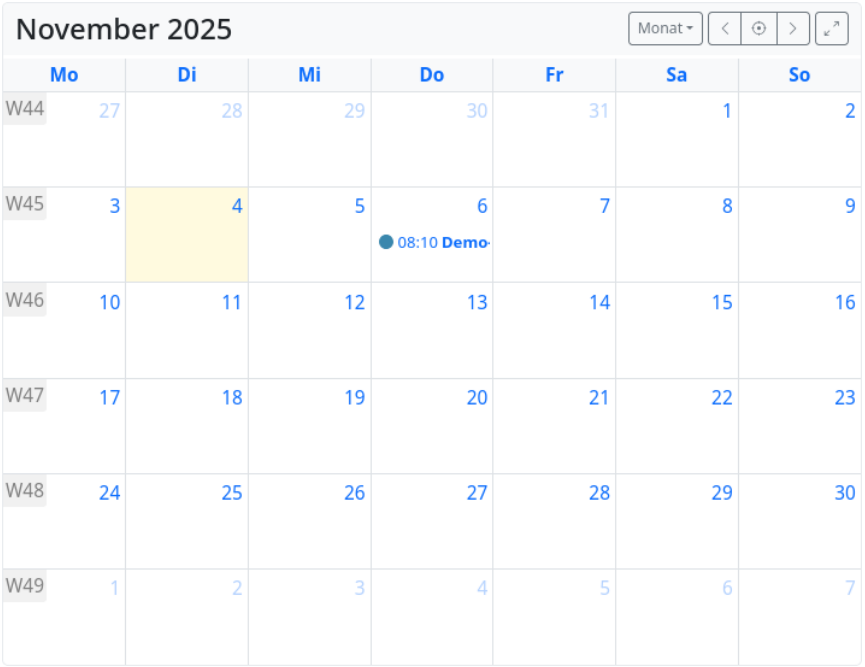
```
<URL_DES_HUB>/cards/<KANALNAME>/<Link_zur_Seite>
```

adressierbar.

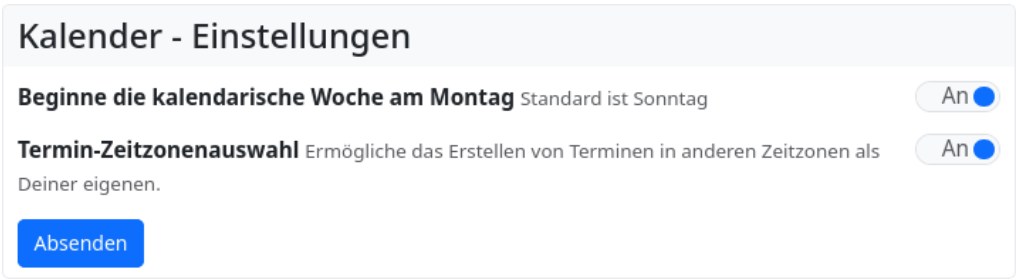
Kalender

Jeder Kanal verfügt über einen Standard-Kalender. Zu erreichen ist dieser über die App "Kalender".

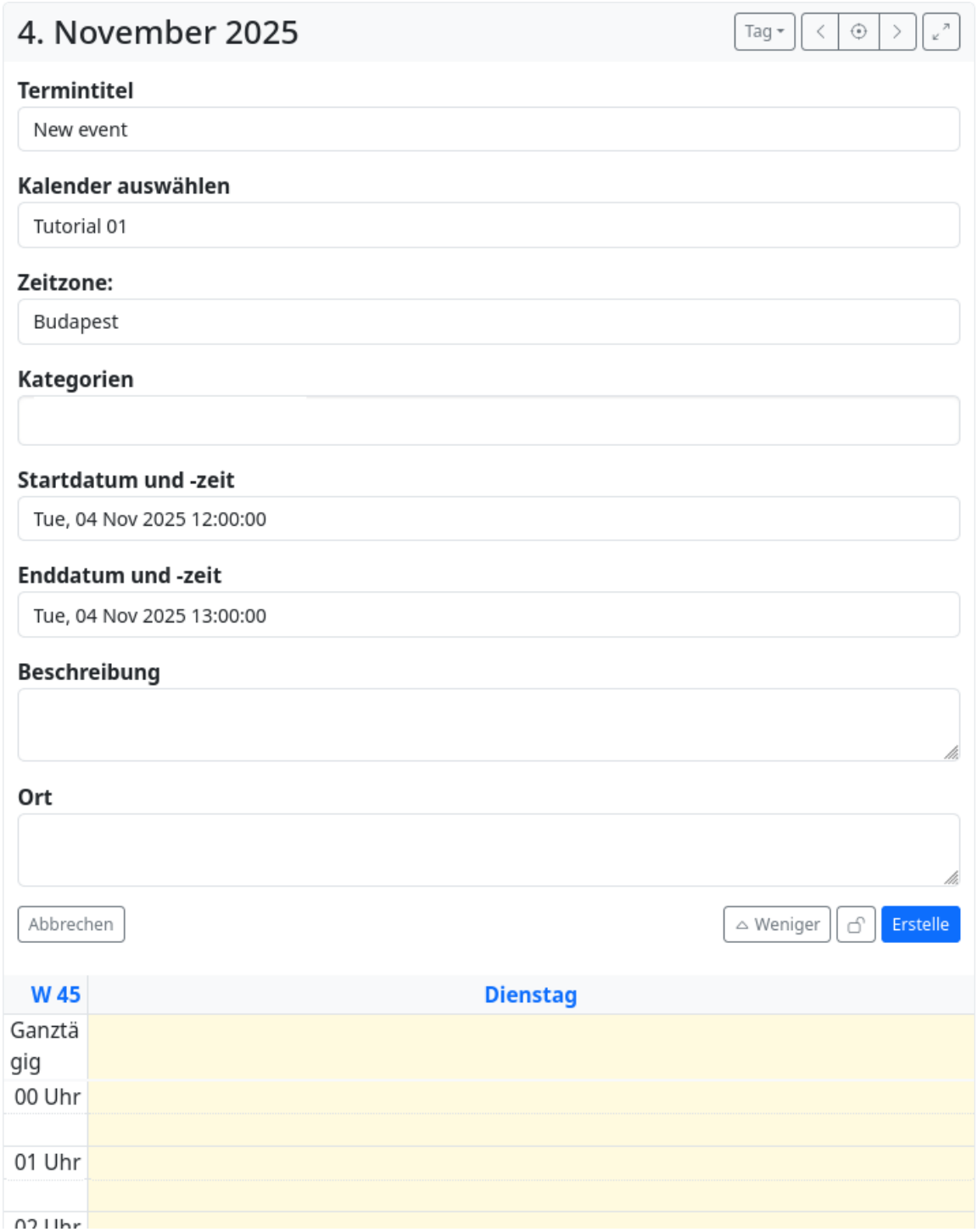
Die Kalender-App erlaubt es, seine eigenen Termine zu verwalten, aber auch als Event-Kalender Termine für die Besucher der Webpräsenz anzuzeigen. Verbundenen Nutzern mit einem Hubzilla-Kanal werden die freigegebenen Termine in ihrem Stre- am angezeigt. Sie können über das Beitrags-Menü (" ") dem jeweils eigenen Kalender zugefügt werden. Der Kalenderbeitrag kann vom Empfänger eine Zusage, Absage oder den Status "unentschlossen" erhalten, was dem Kalender-Inhaber angezeigt wird.



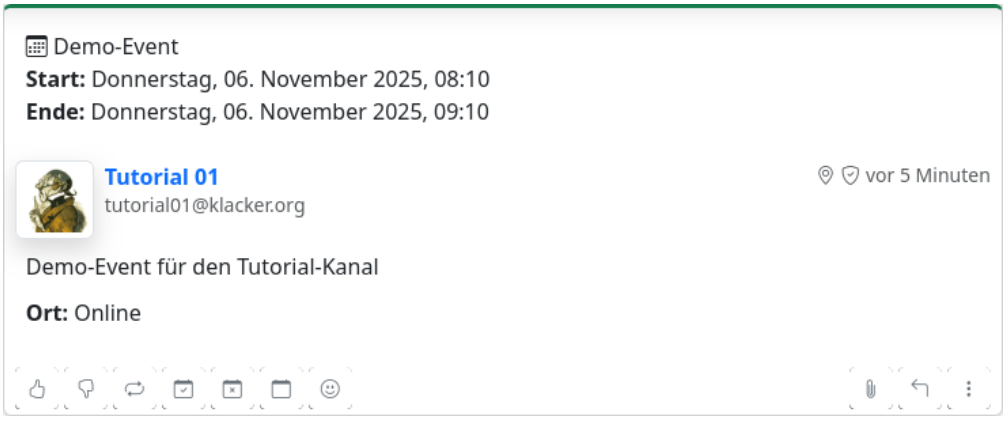
Mit den Kalender-Einstellungen kann man festlegen, ob die Woche mit dem Montag oder dem Sonntag beginnt. Außerdem kann man optional die Nutzung von Zeitzonen einstellen.



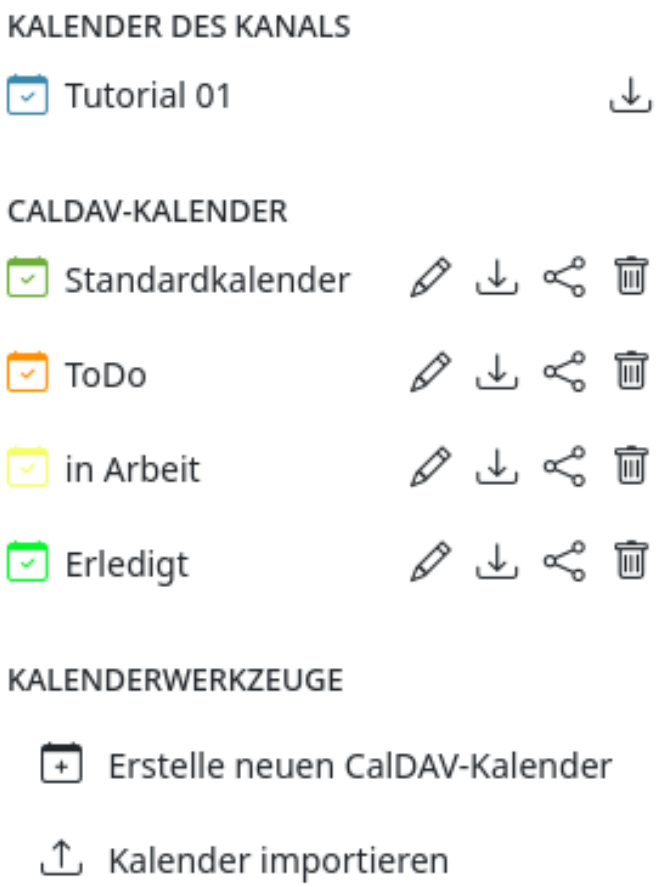
Neue Termine können Sie durch einen Klick auf den entsprechenden Tag hinzufügen. Tun Sie dies in der Monatsansicht, so werden ganztägige Termine erstellt. Aus der Wochen- und Tagesansicht heraus können sie Beginn und Ende des Termins genau festlegen.



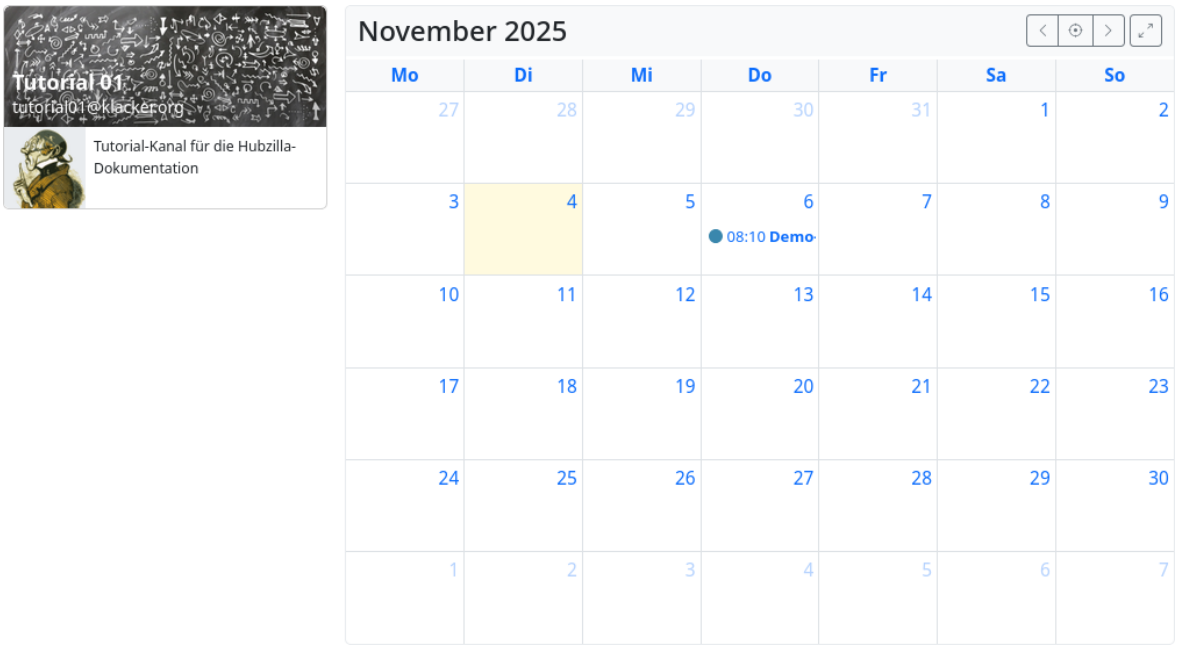
Haben Sie einen Termin erstellt, so erscheint dieser auch in Ihrem Kanal-Stream und, soweit sie Verbindungen mit anderen Kanälen haben und sie die entsprechende Berechtigung für den Termin erteilt haben, auch in den Streams der verbundenen Kanäle.



Innerhalb Ihrer Kalender App können Sie beliebig viele weitere Kalender für unterschiedliche Zwecke hinzufügen. Außerdem können Sie Kalender als ics-Dateien exportieren und Kalender aus ics-Dateien importieren.



Wenn Sie in Ihrem Kanal die App Kalender aufrufen, so werden Ihnen Ihre Kalender mit den Einträgen angezeigt. Diese App-Ansicht hat die URL `<IHR_HUB>/cdav/calendar` und steht für Besucher Ihrer Seite nicht zur Verfügung. Um Ihren Kalender auch Besuchern anzuzeigen, müssen Sie die Ansicht mit der URL `<IHR_HUB>/cal/<KANALNAME>` verwenden.



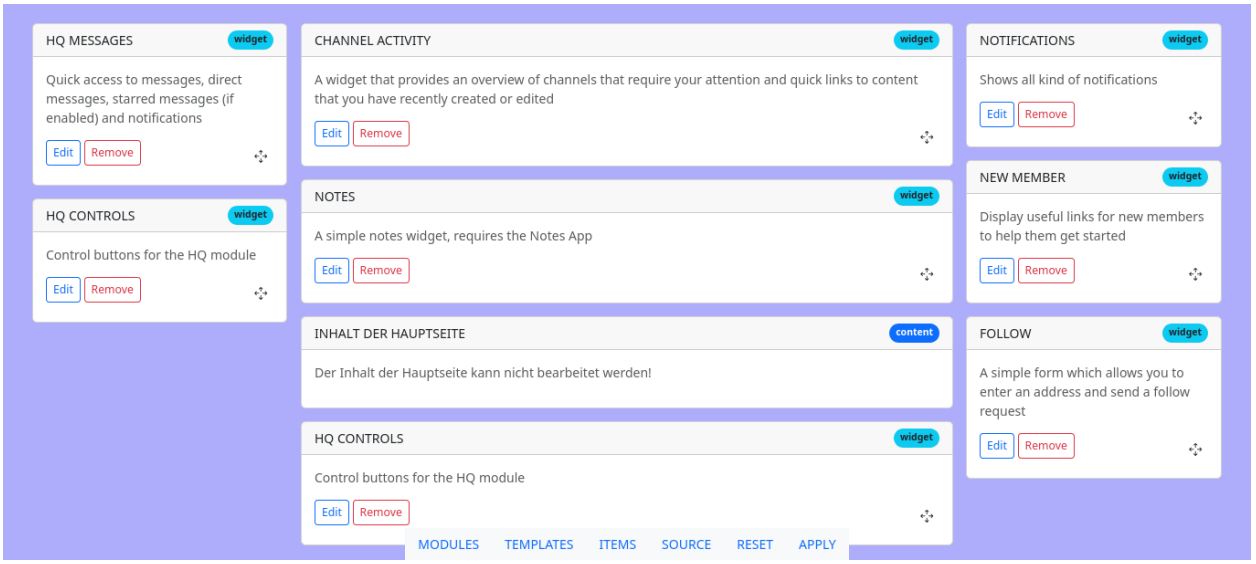
Das Kanal-Layout

Wenn Besucher Ihre Präsenz besuchen, landen sie in der Regel auf der Kanal-Seite Ihres Kanals.

Je nachdem, welche öffentlich zugänglichen Apps Sie in Ihrem Kanal installiert haben, stehen dem Besucher außerdem im App-Menü eben diese Apps, sowie die System-Apps Ihres Hubs zur Verfügung.

Die Layouts dieser Seiten sollten Sie führ Ihre Zwecke entsprechend anpassen. Abgesehen davon sind auch Anpassung der Layouts von Apps und App-Ansichten, welche nur Sie selbst verwenden, sinnvoll und teilweise erforderlich.

Die Layouts werden mit Hilfe das PDL-Editors, welchen Sie als App zunächst installieren müssen, bearbeitet. PDL steht für Page Description Language und liegt bei Hubzilla in der Comanche Layout Beschreibungssprache vor.



Das Hauptmenü des befindet sich am unteren Bildschirmrand:

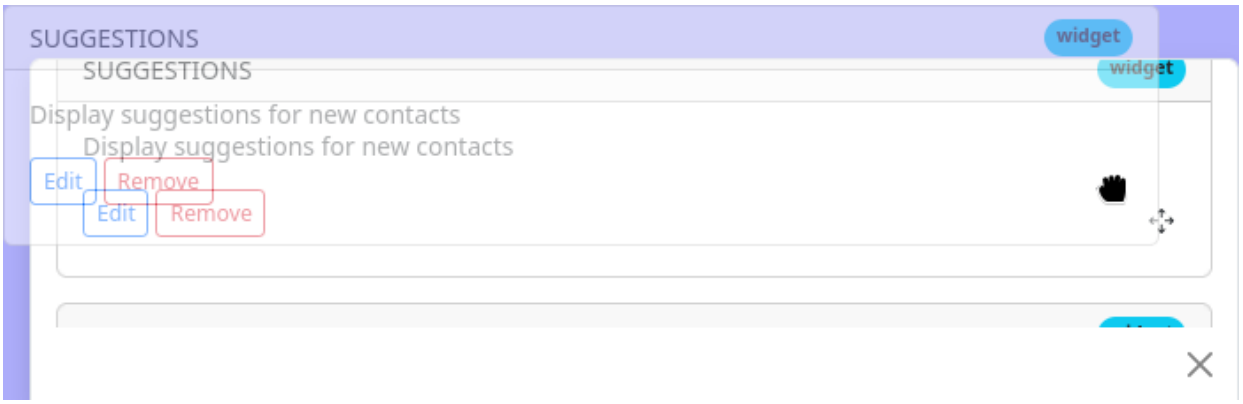
Über den Menüpunkt "MODULES" können Sie das zu bearbeitende Modul auswählen. Die Module steuern die verschiedenen Ansichten. Öffnen Sie z.B. die App "Artikel", so wird das Modul "articles" verwendet.

Mit dem Menüpunkt "TEMPLATES" können Sie die zu verwendende Layout-Vorlage wählen. Standard ("default") ist ein Layout mit einer linken und einer rechten Seitenleiste und einem Inhaltsbereich in der Mitte. "doubleleft" ist ein zweispaltiges Layout mit einer linke Seitenleiste und dem Inhaltsbereich. "doubleright" ist ein zweispaltiges Layout mit einer rechten Seitenleiste und dem Inhaltsbereich. "full" ist ein einspaltiges Layout nur mit der Navigationsleiste und dem Inhaltsbereich und "minimal" ist ebenfalls ein einspaltiges Layout, aber ohne die Navigationsleiste.

Im Normalfall müssen Sie hier vorerst nichts ändern.

Der Menüpunkt "ITEMS" enthält die in dem Modul verfügbaren Inhaltselemente. Hier können die von Hubzilla (und den Addons) angebotenen Widgets ausgewählt werden, aber auch alle Blöcke, welche Sie selbst erstellt haben.

Die Inhaltselemente werden per Drag-and-Drop an die gewünschte Position im Layout verschoben.



Einzelne bereits integrierte Elemente können per Klick auf den entsprechenden Button gelöscht oder bearbeitet werden.

Über den Menüpunkt "SOURCE" können Sie sich das Comanche-Layout des aktuellen Moduls anzeigen lassen und auch bearbeiten.

Haben Sie Ihr Layout verändert und wünschen, es wieder auf die Standardeinstellungen zurückzusetzen, wählen Sie den Menüpunkt "RESET".

Der Menüpunkt "APPLY" schließlich speichert Ihre Änderungen. Das veränderte Layout gestaltet nun die jeweilige Ansicht.

Einige Widgets erlauben es auch, sie über die Verwendung von Variablen zu modifizieren oder anzupassen.

Als Beispiel werde ich in der Artikelansicht das Widget "Album" in die rechte Seitenleiste integrieren. Dieses Widget zeigt die Bilder eines Albums an, also sämtliche Bilder eines Ordners in der Cloud.

Verwendet man das Widget ohne Bearbeitung, werden sämtliche Bilder im Wurzelverzeichnis der Cloud angezeigt. Möchte man die Bilder eines bestimmten Verzeichnisses anzeigen lassen, muss man das Widget mit der entsprechenden Variable konfigurieren. Das Album-Widget zeigt im PDL-Editor an, welche Variablen verwendet werden können. Hier sind es "album" für den gewünschten Ordner in der Cloud und "title" für einen Titel.

Variablen werden gesetzt, indem man sie innerhalb der Widget-Tags mit dem Tag-Paar [var=<VARIABLENNAME>]<WERT_DER_VARIABLE>[/var] definiert.

Als Album sollen die Bilder des Verzeichnisses "Die Alten" angezeigt werden. Als Titel soll der Text "Die alten Hunde" oben im Album-Widget zu sehen sein.

Zuerst schiebe ich das Widget Album an die unterste Stelle der rechten Seitenleiste. Danach muss das Widget durch Klick auf den Button "Edit" bearbeitet werden.

Der Quelltext wird in einem Editor-Fenster angezeigt:

```
[widget=album][/widget]
```

Nun werden die beiden Variablen gesetzt, so dass der Quelltext so aussieht:

```
[widget=album][var=album]Die Alten[/var][var=title]Die alten
Hunde[/var][/widget]
```

Mit Klick auf den Button "Submit" werden die Änderungen übernommen.

Abschließend muss auf den Menüpunkt "APPLY" geklickt werden, damit die Modifikation der Ansicht "Artikel" übernommen wird. Wird nun die App "Artikel" geöffnet, erscheint das Widget mit dem Fotoalbum wie gewünscht in der rechten Seitenleiste.

Durch das Anpassen der Layouts der für die Web-Präsenz relevanten Module kann diese sehr individuell gestaltet werden. Dadurch, dass auch selbst erstellte Blöcke hier zur Verfügung stehen, lässt sich z.B. eine einheitliche Navigation beispielsweise über einen Menü- oder Navigationsblock realisieren.

Das Kanal-Design

Das Standard-Theme eines Hubzilla-Kanals heißt "redbasic". Es handelt sich um ein einfach und klar strukturiertes dreispaltiges Layout und basiert auf Bootstrap. In der Grundeinstellung ist es ziemlich "neutral". Hubzilla erlaubt es, etliche Parameter zu verändern, um die Optik individueller zu gestalten.

Die Einstellungen zum Design stehen in den Anzeige-Einstellungen zur Verfügung. Die Anzeige-Einstellungen erreichen Sie im Hauptmenü unter dem Menüpunkt "Einstellungen".

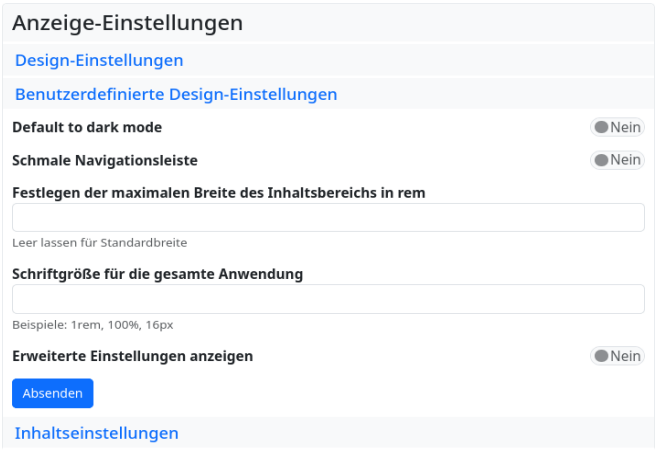
Die Anzeige-Einstellungen ist in drei Unterabschnitte (als Tabs) unterteilt.

Der erste Abschnitt ermöglicht die Auswahl des zu verwendenden Themes. Bei einem neu angelegten Kanal ist hier "redbasic" eingestellt. Sollte der Administrator Ihres Hubs weitere Themen installiert und aktiviert haben, können Sie an dieser Stelle auch ein anderes Theme auswählen. Je nach Gestaltung des anderen Themes, gibt es auch andere benutzerdefinierte Design-Einstellungen und ggf. wird ein anderes Seitenlayout verwendet. Die folgenden Ausführungen beziehen sich auf das Theme "redbasic", welches auf jedem Hub vorhanden ist.

Im ersten Abschnitt kann man auch einen Stil für das Theme auswählen. Bei redbasic sind dies "Focus" und "Focus-Boxy". Der zweite Style unterscheidet sich nur dadurch vom Standard-Style "Focus", dass einzelne Elemente einen feinen Rahmen aufweisen.

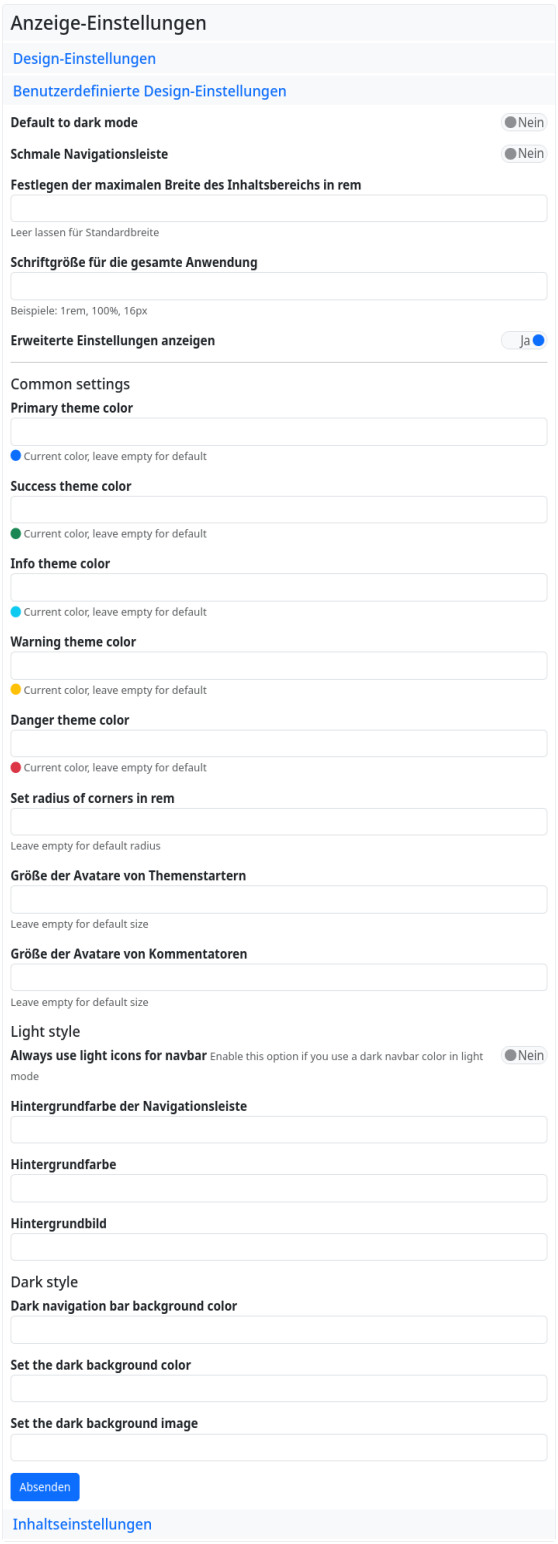
Der zweite Abschnitt bietet "Benutzerdefinierte Design-Einstellungen". Bei einem neu angelegten Kanal sind das nur sehr wenige. Sie können ein dunkles Theme erzwingen, auf eine "schmale" (das bedeutet in der Höhe reduzierte) Navigationsleiste umstellen, die Breite des Inhaltsbereichs in rem festlegen, sowie die Schriftgröße für die gesamte Anwendung modifizieren.

Mehr Einstellungen stehen nicht zur Verfügung.



Nehmen Sie hier Änderungen vor, müssen diese durch Klick auf den Button "Absenden" bestätigt werden.

Am Ende der Einstellungen gibt es den Schalter "Erweiterte Einstellungen anzeigen". Stellen Sie diesen auf "Ja", klicken Sie auf absenden und rufen nun erneut die benutzerdefinierten Design-Einstellungen auf, so haben Sie wesentlich mehr Möglichkeiten für die Einstellungen.



Das Theme "redbasic" verwendet die Hintergrundfarben von Bootstrap. Dies sind bg-primary, bg-success, bg-info, bg-warning und bg-danger. Diese Standard-Farben können Sie an dieser Stelle nun überschreiben. Dazu können Sie diese entweder im Hex-Format `#RRGGBB` eingeben, oder druch Klick in das Eingabefeld den Farbwähler verwenden.

Sie können außerdem den Eckradium von Elementen in rem, sowie die Größe der Avatare von Themenstartern und von Kommentatoren festlegen.

Weiterhin gibt es noch Einstellungen für den "Light Style", also den hellen modus und den "Dark Style", den dunklen Modus.

Für den "Light Style" können Sie festlegen, dass trotzdem die Icons des "Dark Style" verwendet werden. Das ist sinnvoll, falls Sie für die Farbe der Navigationsleiste im hellen Modus trotzdem eine relativ dunkle Farbe einstellen.

Sowohl für den hellen, wie auch für den dunklen Modus können Sie hier nun die Farbe der Navigationsleiste und die Hintergrundfarbe der gesamten Webseite festlegen. Beides geschieht in der rgba-Notation, also `rgba(RRR,GGG,BBB,A)`, wobei A für den Alpha-Kanal, also einen möglichen Transparenz-Effekt steht.

Außerdem können sie festlegen, dass für den gesamten Webseiten-Hintergrund ein Bild verwendet wird. Dafür müssen Sie die URL eines Hintergrundbildes eintragen, sinnvollerweise sollte es in der Cloud des Kanals liegen.

Der dritte Abschnitt ist für die Inhaltseinstellungen. Hier kann man einstellen, dass Emoticon als Bilder angezeigt, also in solche umgewandelt werden. Außerdem besteht die Möglichkeit, festzulegen, dass Beitragstitel immer zum Originalbeitrag verlinken. Für die Nutzung auf mobilen Endgeräten kann man den Zoom-Modus einschalten und man kann a dieser Stelle die Anzeige des Widgets mit Hinweisen für neue Benutzer abschalten. Schließlich kann man die Thread-Darstellung von Konversationen ein- bzw. abschalten.

Durch sinnvolle Einstellungen lässt sich die Webpräsenz hier den eigenen Bedürfnissen und ggf. an die CI anpassen.

Webseiten

Die App "Webseiten", die in einem Kanal explizit installiert werden muss, ist der Dreh- und Angelpunkt bei der Nutzung von Hubzilla als CMS. Sie bietet nicht nur die Möglichkeit, komplette Webseiten zu erstellen, sondern auch die Erstellung von Blöcken, welche dann z.B. in Seitenlayouts, also auch in den Layouts des System-Seiten und -Apps, integriert werden können.

Außerdem gehört zur App "Webseiten" auch der Menü-Editor, mit dem sich Menüs, sowie die Bookmarks von Hubzilla erstellen und bearbeiten lassen.

Die Oberfläche von Hubzilla basiert auf dem Frontend-CSS-Framework Bootstrap (Version 5). Für die Gestaltung von Webseiten und Blöcken können damit, ohne gesonderte Einbindung des Frameworks, die Features von Bootstrap5 genutzt werden.

Eine gute Einführung in die Benutzung von Bootstrap5 findet man bei W3Schools: Bootstrap 5 Tutorial.

Wer mit Hubzilla als CMS arbeiten möchte, sollte über grundlegende Kenntnise von HTML, CSS und ggf. Javascript verfügen. Auch hier wird man bei W3Schools füdig. Ebenfalls sehr empfehlenswert ist das SelfHTML Wiki.

Nach Aufruf der App "Webseiten" erscheint die Übersicht der vorhandenen Webseiten, welche in der Seitenleiste über ein Auswahlmenü verfügt, über welches man die Gestaltungswerkzeuge für die einzelnen Komponenten (Blöcke, Menüs, Layouts und Webseiten) erreichen kann und das die Möglichkeit für einen Im- und Export von Webseiten bereitstellt.

Webseiten: Blöcke

Mit dem Gestaltungswerkzeug "Blöcke" kann man wiederverwendbare Inhalte erstellen. Sie können in einem Webseiten-Layout und dem Layout einer Ansicht mittels des PDL-Editor verwendet werden.

Blöcke können Inhalte verschiedener Inhaltstypen enthalten:

1. Bbcode
2. HTML
3. Markdown
4. Text (Plaintext)
5. Comanche-Layout
6. PHP

Die üblichen und sinnvollen Inhaltstypen sind BBcode, HTML und Markdown.

BBcode ist für eine einfache Gestaltung von Blöcken sehr gut geeignet. Außerdem hat er den Vorteil, dass er in Hubzilla einige hubzilla-spezifische Tags nutzbar macht, die z.B. den derzeit genutzten Hub referenzieren oder auf Inhalte verweisen kann, die mittels Open Web Auth und Magic Auth auch zugriffsbeschränkte Inhalte zugänglich gemacht werden können.

Markdown bietet ähnliche Vorteile, ist aber in der Gestaltungsmöglichkeit beschränkter. Das Format bietet sich insbesondere für Textdokumente an.

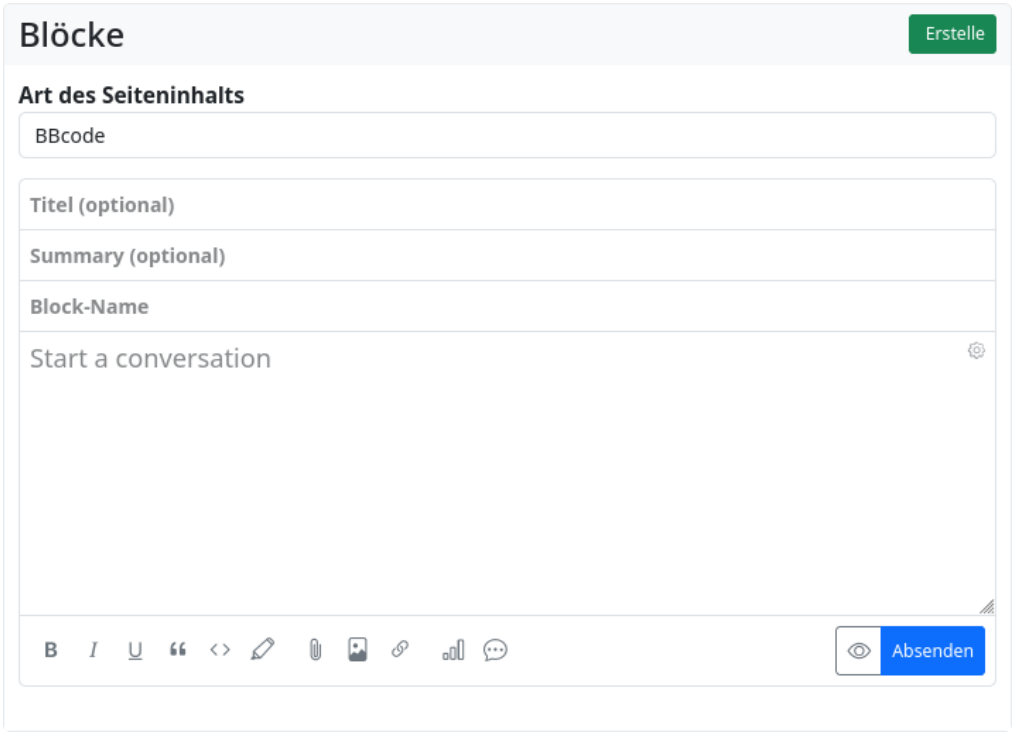
Mit HTML hat man den größten Gestaltungsspielraum. Damit ist in einem Block nahezu alles möglich, was mit HTML generell machbar ist. Man kann Frameworks einsetzen (insbesondere Bootstrap, welches ohnehin in Hubzilla schon zur Verfügung steht), eigene Stylesheets inline oder mittels separater Dateien verwenden, PHP-Code inline nutzen, ebenso Javascript (selbstverständlich können auch JS-Bibliotheken genutzt werden).

Plaintext mag für spezielle Anwendungsfälle auch eine Berechtigung haben, beschränkt sich aber auf reine Textdarstellungen ohne Gestaltungsmöglichkeiten.

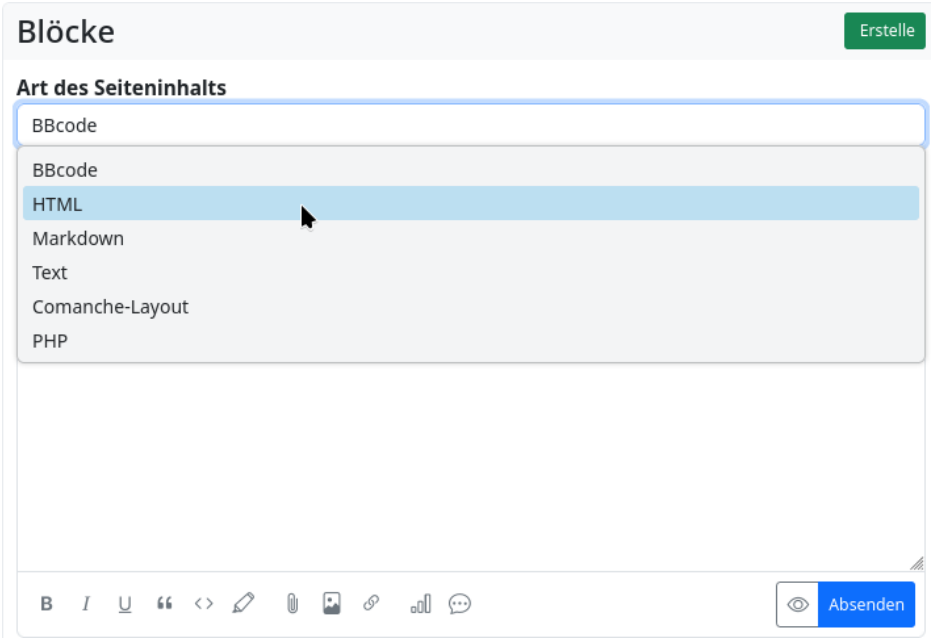
Commanche-Layout ist in Blöcken eher von theoretischer Natur und auf wirklich sehr spezielle Anwendungsfälle beschränkt.

Schließlich lässt sich auch PHP-Code zur Seitenerstellung nutzen. Hier sind der Fantasie kaum Grenzen gesetzt. In Verbindung mit einer Datenbank sind auch komplexe Webanwendungen möglich, ohne als Administrator zusätzliche Module für Hubzilla erstellen zu müssen.

Klicken Sie auf den Button "Erstelle", öffnet sich der Block-Editor Dialog.



Im Auswahlfeld "Art des Seiteninhalts" kann nun der Inhaltstyp ausgewählt werden.



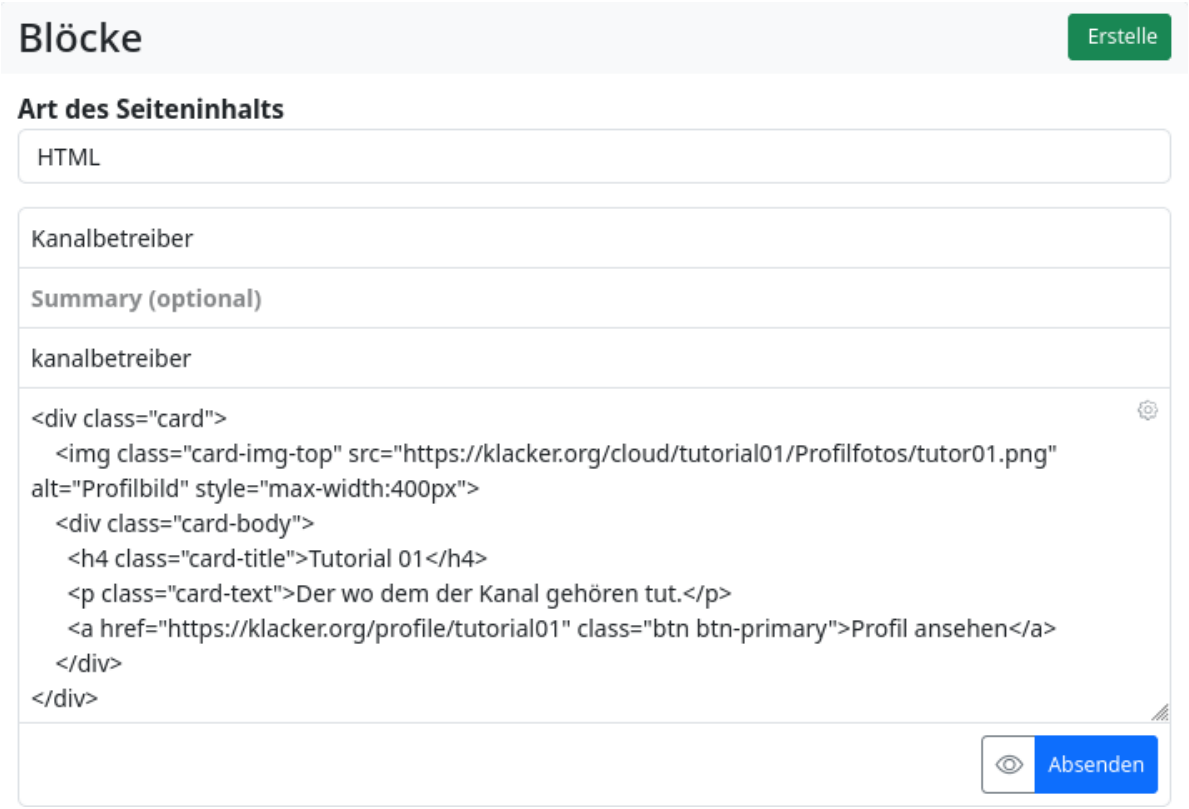
Ein Block kann optional mit einem Titel versehen werden. Dieser Titel erscheint als Überschrift im Header des dargestellten Blocks.

Ebenfalls optional und nur bei artikel-artigen Blöcken sinnvoll ist die Zusammenfassung ("Summary") des Inhalts. Bei der ersten Darstellung des Blocks wird diese Zusammenfassung angezeigt. Darunter befindet sich der anklickbare Text "Artikel ansehen". Klicken Sie darauf, wird der Block-Inhalt angezeigt. Über dem Inhalt erscheint der anklickbare Text "Zusammenfassung ansehen", was zum erneuten Einklappen des Inhalts dient.

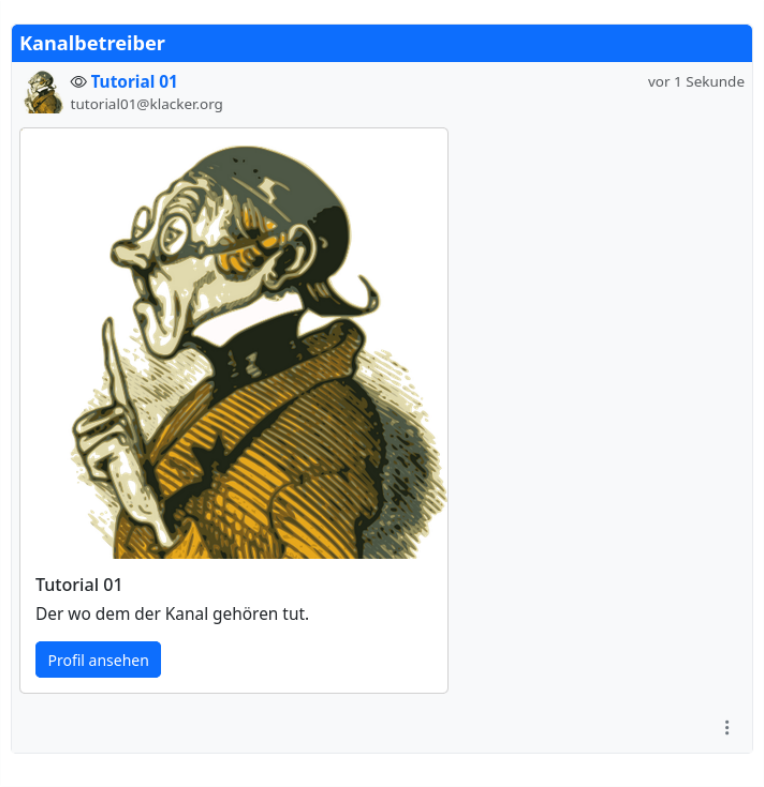
Unter dem Summary-Feld befindet sich das Eingabefeld für den Block-Namen. Hier ist es sinnvoll, einen passenden kurzen Namen zu vergeben. Lässt man das Feld leer, erzeugt Hubzilla einen Hash-Wert, der als Block-Name fungiert. Der Name lässt sich aber, falls man die Eingabe vergessen hat, nachträglich noch ändern, wenn man den Block editiert.

Schließlich gibt es ein Editor-Feld, in welches Sie den Inhalt des Blocks eingeben können.

Hier nun ein Beispiel-Block, der eine Karte des Kanalinhabers anzeigt.



Hat man die notwendigen Informationen eingegeben, kann man sich eine Vorschau des Blocks anzeigen lassen. Dafür klickt man auf das Augen-Symbol neben dem Button "Absenden".

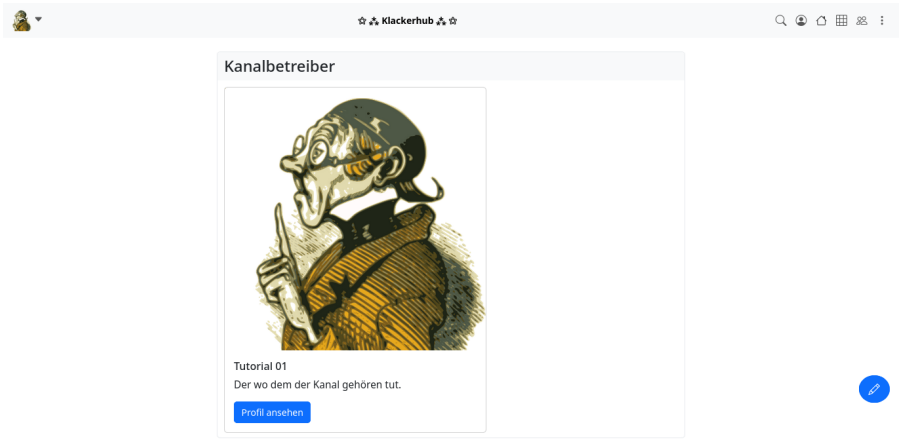


Ist man mit dem Ergebnis zufrieden, klickt man auf den Button "Absenden". Der Block wird erstellt und erscheint nun in der Liste der Blöcke.

Der Block kann hier durch Klick auf das Bleistift-Symbol bearbeitet werden. Daneben befindet sich ein "Download-Symbol", welches hier aber nicht dem Download dient, sondern den Block im eigenen Kanal teilt, wodurch er föderiert, also an die Verbindungen verteilt wird. Das Mülleiner-Symbol dient dem Löschen des Blocks.

Der Block kann hier durch Klick auf das Bleistift-Symbol bearbeitet werden. Danebn befindet sich ein "Download-Symbol", welches hier aber nicht dem Download dient, sondern den Block im eigenen Kanal teilt, wodurch er föderiert, also an die Verbindungen verteilt wird. Das Mülleiner-Symbol dient dem Löschen des Blocks.

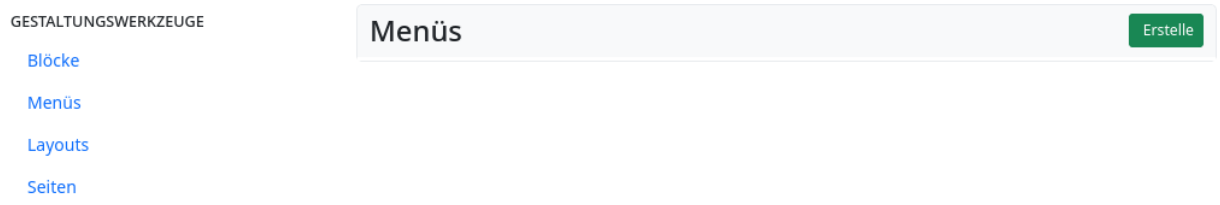
Klickt man auf den Block-Namen, wird der Block angezeigt.



Um den Beispiel-Block nun irgendwie sinnvoll zu verwenden, können wir ihn – ebenfalls als Beispielanwendung – mit dem PDL-Editor in die Seitenleiste der Kanal-Ansicht einfügen.

Webseiten: Menüs

Das Gestaltungswerkzeug "Menüs" dient der einfachen Erstellung von Menüs. Es handelt sich nicht um besonders designte Menüs, sondern um Listen mit Links.



Ein Klick auf den Button "Erstelle" öffnet den Menü-Editor.

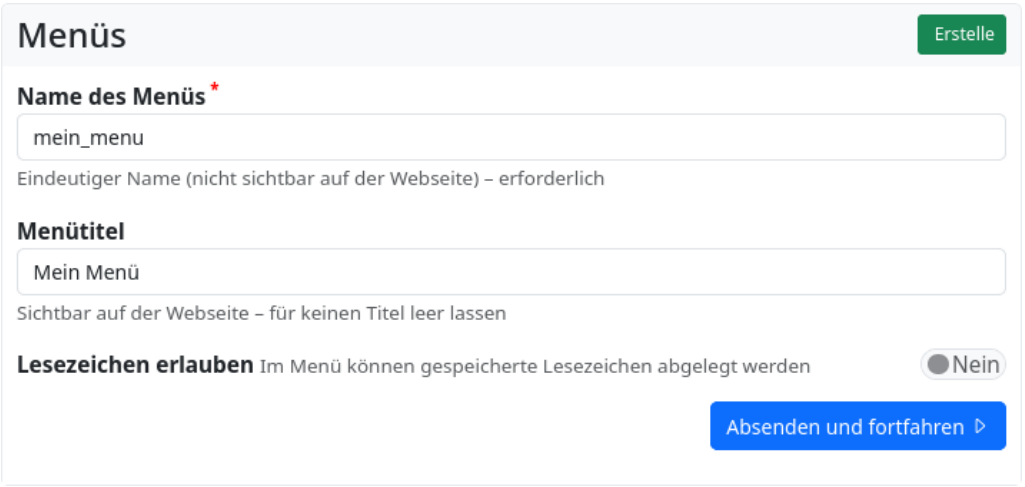
Im ersten Schritt erzeugen Sie damit das Menü an sich. Sie müssen dem Menü einen Namen geben. Dieser taucht nicht auf der Webseite auf und ist dafür gedacht, das Menü zu referenzieren.

Sie können dem Menü auch einen Titel geben, der als Kopfzeile über dem Menü auf der Webseite erscheint (optional).

Schließlich können Sie das Menü mit der Möglichkeit ausstatten, dort auch Lesezeichen abzulegen. Das ist eine spezielle Funktion von Hubzilla. Wenn Sie die App "Lesezeichen" installiert haben, können Sie Links, die mit "#^" bzw. dem Lesezeichen-Symbol 🔖 beginnen, durch klick auf das Zeichen bzw. Symbol dem Bookmark-Menü hinzufügen. Erlauben sie Ihrem selbst erstellten Menü das Hinzufügen von Bookmarks, so können Sie auf diese Weise auch entsprechend formatierte Links dem eigenen Menü hinzufügen.

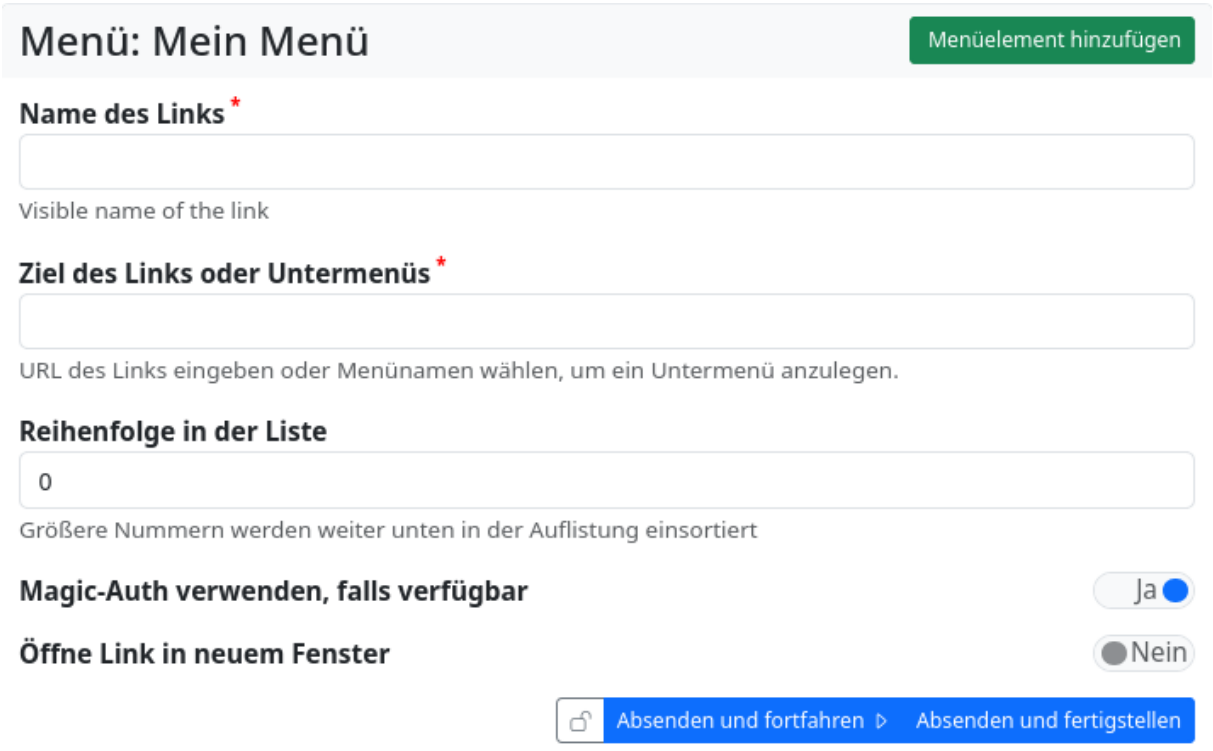
Hier erzeugen wir ein Beispiel-Menü mit dem Namen "mein_menu" und dem Titel

"Mein Menü". Bookmarks lassen wir nicht zu.



Anschließend klicken wir auf "Absenden und fortfahren". "Fortfahren" deshalb, weil wir dem Menü noch Einträge hinzufügen müssen.

Nun wird der Menüeintrags-Editor angezeigt, mit dem wir Menüeinträge zu unserem neu erstellten Menü hinzufügen können.

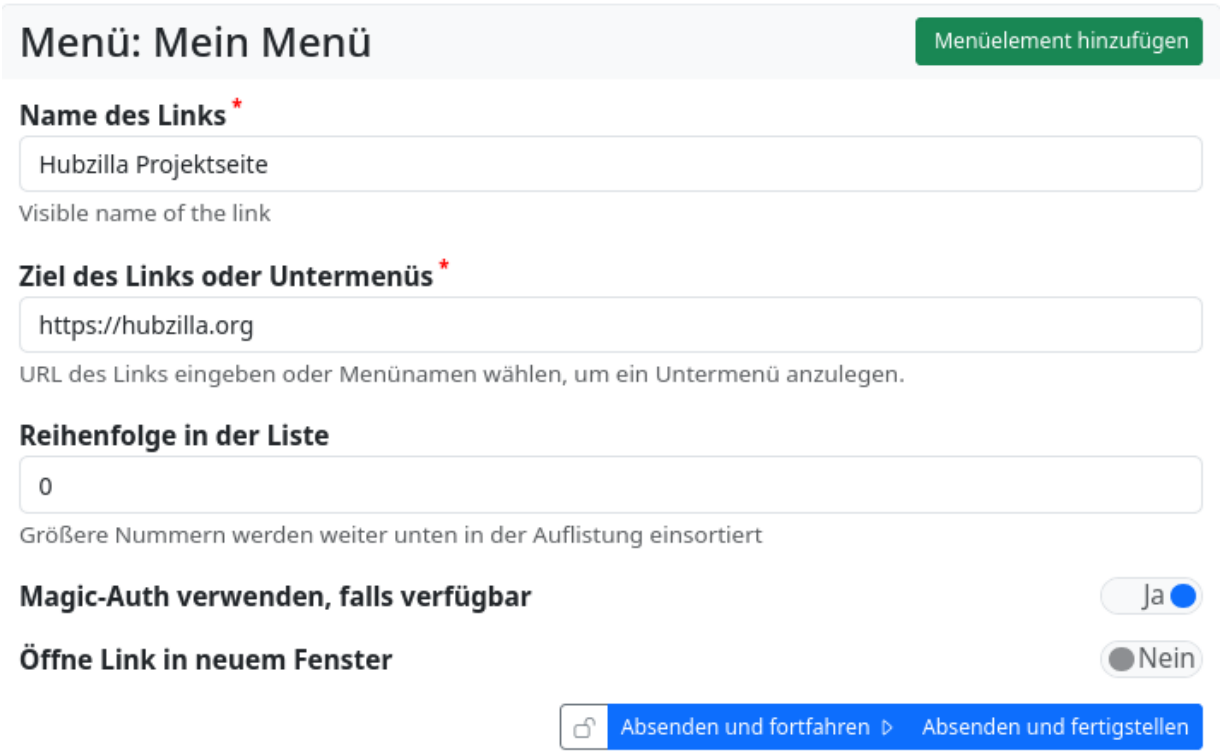


In das erste Textfeld "Name des Links" tragen wir den auf der Webseite sichtbaren Namen des Links ein. In das zweite Textfeld "Ziel des Links oder Untermenüs" muss das Linkziel als URL eingetragen werden. Es ist auch möglich den Menünamen eines anderen Menüs einzutragen. Dann wird ein Untermenü erzeugt, welches aus dem eingetragenen Menü besteht.

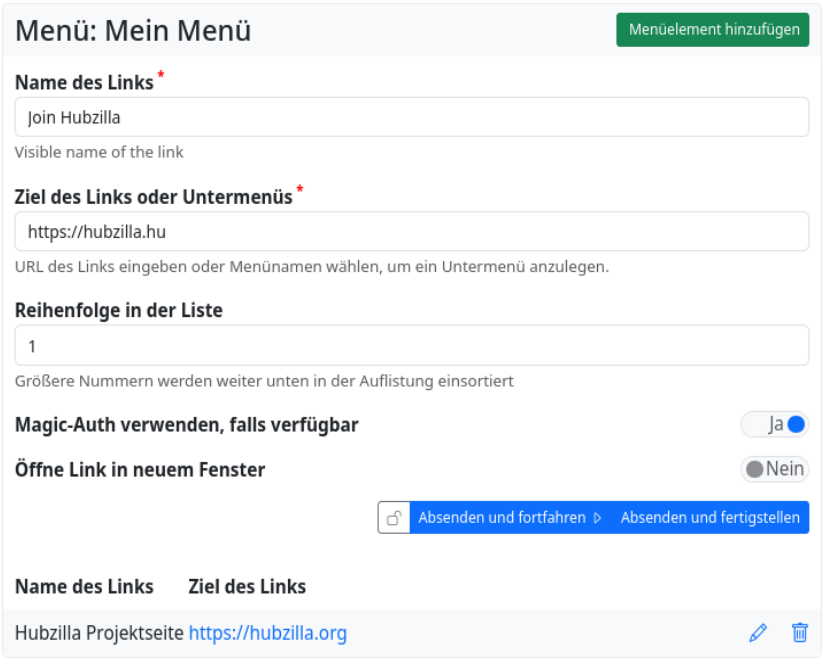
Mit dem Feld "Reihenfolge in der Liste" können wir, sobald wir mehrere Einträge haben, die Position in des Liste festlegen.

Der Schalter "Magic-Auth verwenden, falls verfügbar" hat eine besondere Bedeutung. Führt der Link zu Hubzilla-Seiten oder -Objekten, welche eine Authentifizierung erfordern, wird diese bei Auswahl des Menü-Links automatisch mittels Magic-Auth durchgeführt.

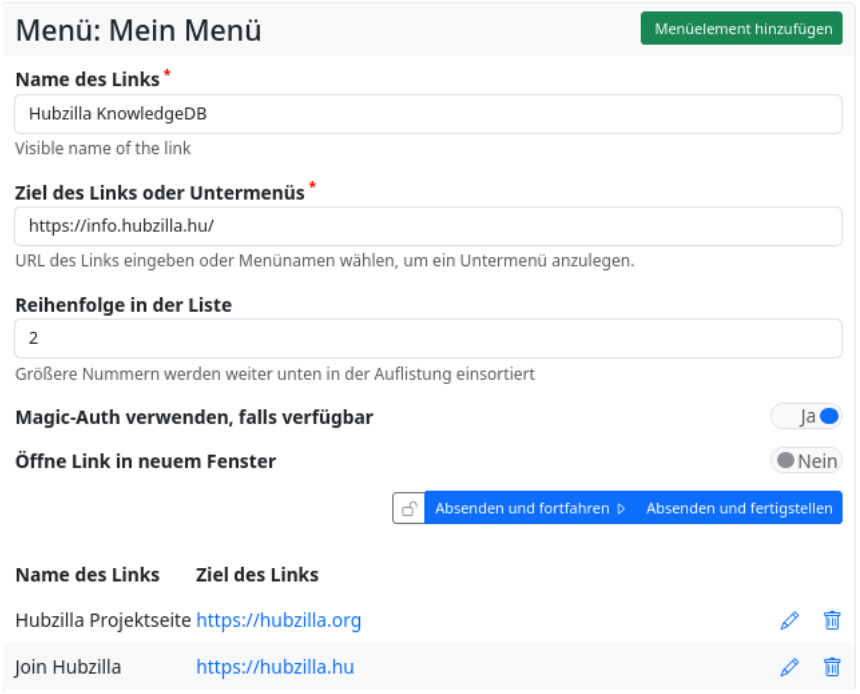
Schließlich können wir noch festlegen, ob das Ziel in einem neuen Fenster bzw. Browsertab geöffnet werden soll.



Klickt man auf "absenden und fertigstellen", wird die Menübearbeitung beendet. Möchte man nacheinander aber weitere Menüpunkte hinzufügen, klickt man auf "Absenden und fortfahren", worauf sich ein neues, leeres Formular des Menüeintrags-Editors.



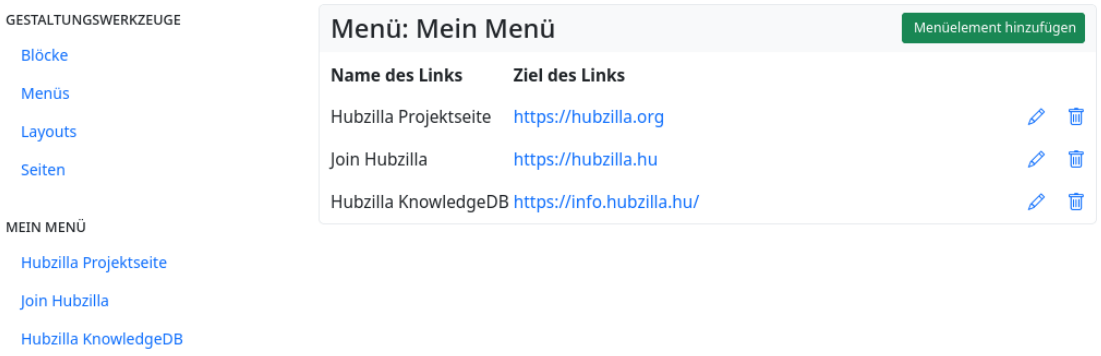
Wir fügen hier im Beispiel noch einen dritten Eintrag hinzu.



Nach dem jeweiligen Absenden eines Eintrags wird dieser unter dem Editor in einer Liste angezeigt. Hier kann man einen Eintrag auch, sofern erforderlich, noch einmal bearbeiten oder löschen.

Nach dem Fertigstellen taucht das Menü nun in der Liste der Menüs im Gestaltungswerkzeug "Menüs" auf.

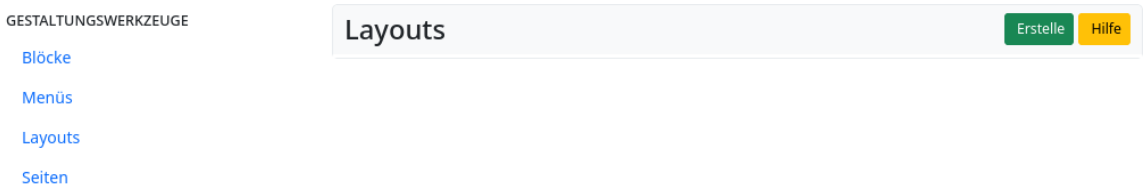
Ein Klick auf in Menü in der Liste öffnet einerseits die Liste der Menüeinträge mit den eben genannten Bearbeitungsmöglichkeiten und stellt es außerdem in der linken Seitenleiste dar, wo man die Funktionalität testen kann.



Zu Beachten ist, dass Menüs immer vertikal, also mit untereinander aufgelisteten Einträgen erscheint. Es taugt also, sofern im Inhaltsbereich noch andere Inhalte angezeigt werden sollen, eher für die Nutzung in einer Seitenleiste. Horizontale Menüs müssen per Hand mittels eines HTML-Blocks erstellt werden. Möchten Sie ein horizontal (also mit nebeneinander dargestellten Menüeinträgen) verwenden, bietet es sich an, dieses mit der Bootstrap-Klasse nav oder noch schicker navbar zu erstellen.

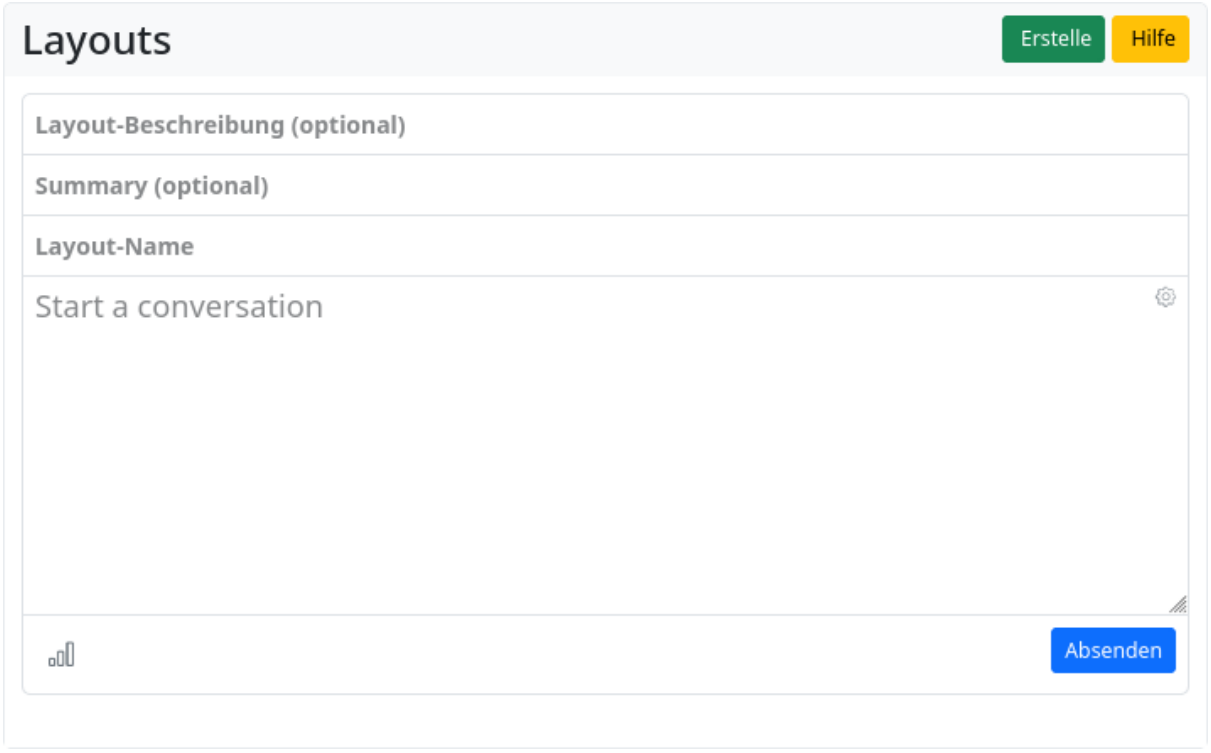
Webseiten: Layouts

Für die Gestaltung der Webseiten haben die Layouts eine entscheidende Bedeutung.



Layouts werden mit der Comanche Seitenbeschreibungssprache, welche dem BBcode-Format ähnelt, erstellt. Comanche ist eine bbCode-ähnliche Auszeichnungssprache, mit der aufwendige und komplexe Webseiten erstellt werden können, indem sie aus einer Reihe von Komponenten zusammengesetzt werden, von denen einige vorgefertigt sind, und andere selbst definiert werden können. Comanche wählt in erster Linie aus, welche Inhalte in den verschiedenen Bereichen der Seite erscheinen sollen. Die verschiedenen Bereiche haben Namen, und diese Namen können sich je nach gewählter Layoutvorlage ändern.

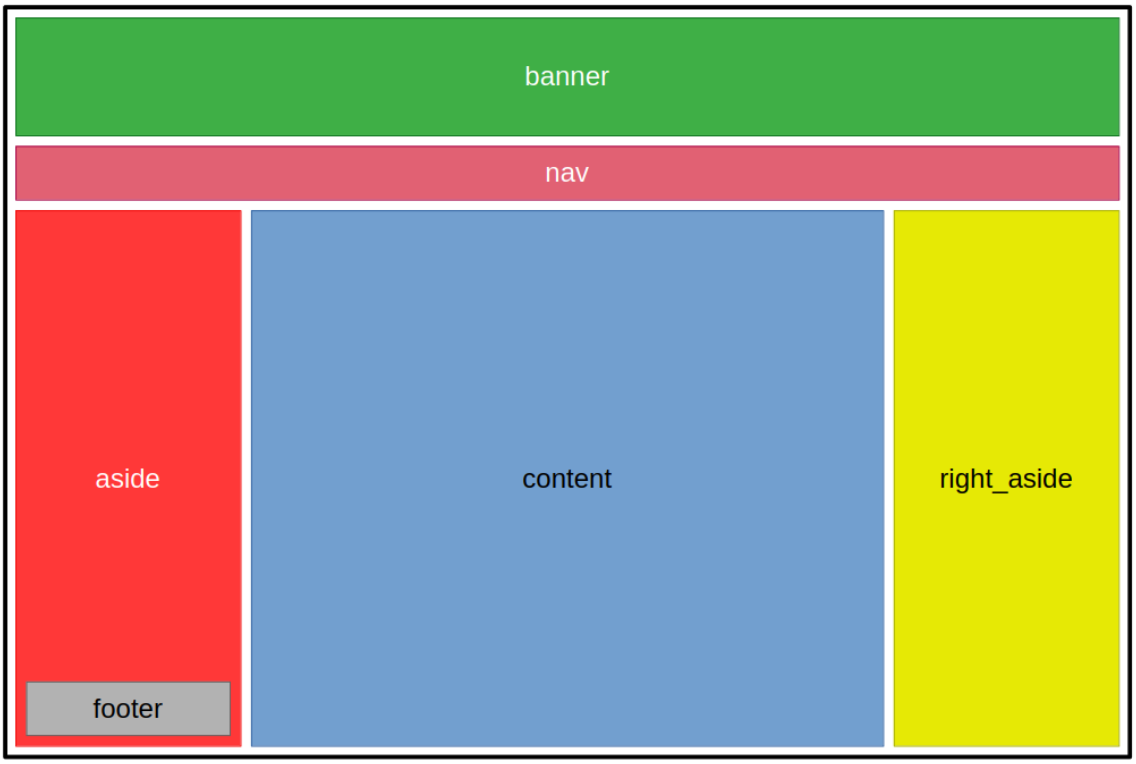
Der Layout-Editor ist simpel und selbsterklärend. Erforderlich ist es, dem Layout einen Namen zu geben. Über diesen Namen kann das Layout später bei Erstellung einer Webseite dieser zugewiesen werden.



Aktuell gibt es fünf verschiedene vordefinierte Vorlagen: default, full, choklet, redable und zen, wobei choklet in sechs verschiedenen "Flavours" daher kommt (default, bannertwo, three, edgestwo, edgesthree und full).

Um eine Layout-Vorlage auszuwählen, verwendet man die Tags [template][/template]. Im Fall von choklet wird der "Geschmack" (Flavour) über das öffnende Tag festgelegt `[template=<flavour>]choklet[/template]`. Lasst man für choklet die Auswahl des Flavour weg, wird "default" verwendet.

Es gibt verschiedene Bereiche einer Webseite:



In der Region "content" sollte in der Regel die Variable $content eingetragen werden. Damit erscheint der Inhalt einer Webseite, wenn man diese mit dem Gestaltungswerkzeug "Webseiten" erstellt, im Inhaltsbereich.

```
[region=content]
$content

[/region]
```

Die Region `[htmlhead][/htmlhead]` kann dazu verwendet werden, andere css- und js-Definitionen zu verwenden. Derzeit werden als Standard jquery (js), bootstrap (css/js) und foundation (css/js) verwendet.

Diese Region wird nicht dargestellt und dient den Zwecken, welche der gleichnamige Bereich in HTML-Dateien ebenfalls erfüllt.

```
<head>

…

</head>
```

Vorhanden sind folgende Vorlagen:

1. **default** verwendet folgende Bereiche: nav, aside (mit fester Breite), content und footer
2. **full** verwendet folgende Bereiche: nav, content, footer
3. **readable** zum Lesen längerer Texte hat keinen nav Bereich und verwendet folgende Bereiche: aside, content und right_aside
4. **zen** ist eine völlig leere Vorlage und verwendet lediglich den content Bereich.
5. **choklet** ist eine floating Layoutvorlage in verscheidenen Flavours:
    - ***default*** verwendet folgende Bereiche: nav, aside, content und footer
    - ***bannertwo*** verwendet folgende Bereiche: banner, nav, aside, content und footer
    - ***three*** verwendet folgende Bereiche: nav, aside, content, right_aside und footer
    - ***edgestwo*** verwendet folgende Bereiche: (fixed size) nav, aside und content
    - ***edgesthree*** verwendet folgende Bereiche: (fixed size) nav, aside, content und right_aside
    - ***full*** verwendet folgende Bereiche: (fixed size) nav, (darunter!) banner, aside, content und right_aside

In die einzelnen Bereiche, die sich aus der gewählten Vorlage ergeben, können nun Blöcke, Menüs und Widgets eingefügt werden. Schema:

```
[region=<bereich>]


[block]<block>[/block]

[menu]<menü>[/menu]

[widget]<widget>[/widget]


[/region]
```

Es können hier den Blöcken auch Klassen aus der CSS-Definition zugewiesen werden (`[block=<klasse>]<block>[/block]`).

Ebenso ist es möglich, Variablen, die für die Darstellung genutzt werden, zu definieren (`[var=<variable>]<variablenwert>[/var]`).

Sie können die Layout-Definition auch mit Kommentaren versehen. Dies geschieht mit den Tags `[comment]<kommentar>[/comment]`. Kommentare werden in der Webseite nicht dargestellt.

Schließlich erlauben die Layout-Definitionen auch eine bedingte Ausführung. Damit werden bestimmte Inhalte nur dann angezeigt, wenn eine bestimmte Bedingung erfüllt ist. Für die Bedingungen können derzeit ausschließlich die Systemvariablen von Hubzilla genutzt werden.

```
[if $<systemvariable>]
  ...
[else]
  ...
[/if]
```

Es sind natürlich auch komplexere Bedingungen möglich.

```
[if $<systemvariable> == <wert>]    // ist gleich
  ...
[/if]
```

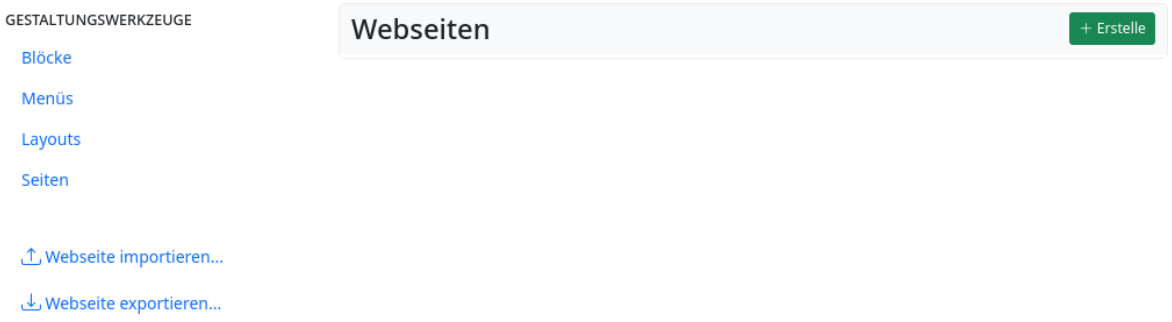
```
[if $c<systemvariable> != <wert>]   // ist ungleich
  ...
[/if]
```

```
[if $<systemvariable> {} <wert-array> ]  // das Array enthält
  ...
[/if]
```

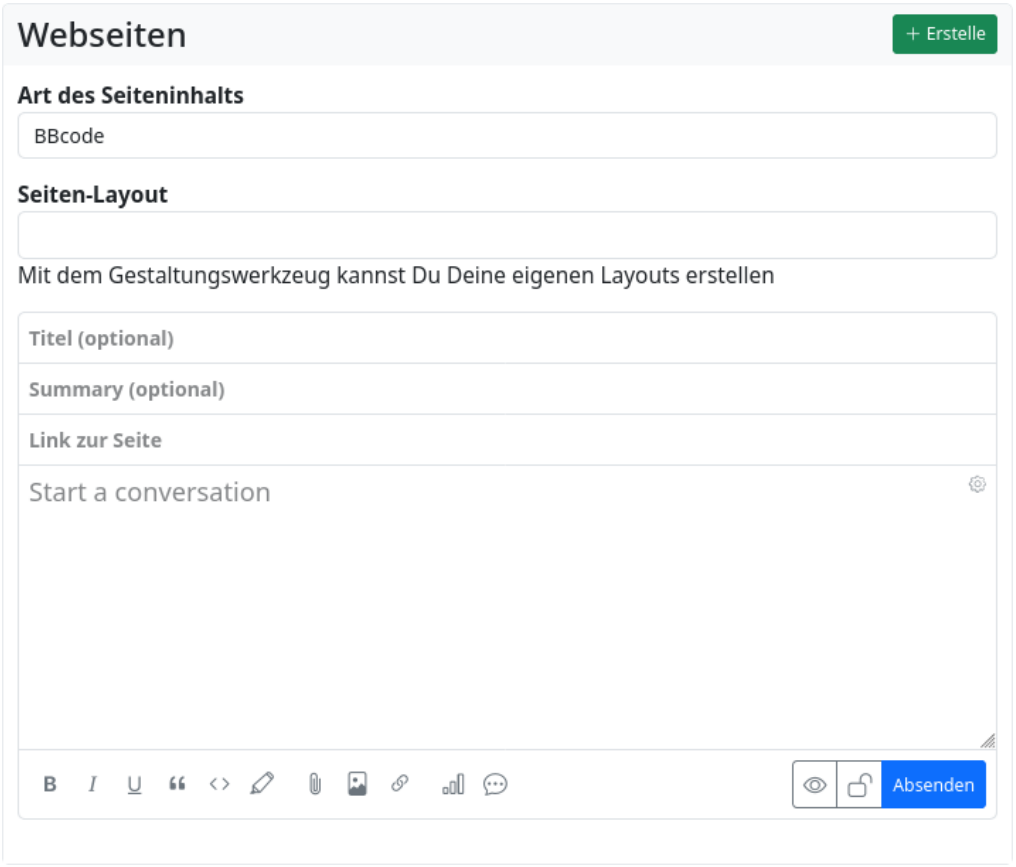
```
[if $c<systemvariable> {*} <wert>]   // der Wert enthält
  ...
[/if]
```

Webseiten: Seiten

Das Gestaltungswerkzeug "Seiten" dient dazu, die eigene Webseite mit Leben zu erfüllen. Mit ihm wird einer Webseite außerdem ein Layout und ein Link-Name ("Link zur Seite") zugewiesen.



Durch Klick auf den Button "+ Erstelle" öffnen Sie den Seiten-Editor.



Auch für Seiten haben Sie die Wahl bezüglich des Inhaltstyps, wie bei den Blöcken ("Art des Seiteninhalts").

In der Auswahlbox darunter können Sie nun das für die Darstellung genutzte Layout auswählen. Wählen Sie hier nichts, wird das Standard-Layout verwendet.

Geben Sie einen Titel für die Seite ein (optional), wird dieser im Kopf des Inhaltsblocks und in der Titelzeile des Browserfensters bez im Reiter des Browser-Tabs angezeigt.

Das vorhandene Eingabefed für die Zusammenfassung ist ohne Funktion und sollte unbedingt leer gelassen werden.

Mit dem Link zur Seite geben Sie einen referenzierbaren Namen für die Seite an, über welchen diese zu öffnen ist. Die Seite wird über den Link `<URL_DES_HUB>/page/<KANALNAME>/<LINK_ZUR_SEITE>` geöffnet.

In den Editor-Bereich tragen Sie nun den Inhalt Ihrer Seite ein. Dieser wird angezeigt, sofern im gewählten Layout der Inhaltsbereich die Variable `$content` enthält.

Webseiten: Widgets

Widgets sind ausführbare Anwendungen, die vom System bereitgestellt werden und die man mittels Layout auf der Seite platzieren kann. Einige Widgets benötigen Argumente, mit denen das Widget an einen bestimmten Zweck angepasst werden kann. System-Widgets sind hier aufgelistet. Widgets können auch von Plugins, Themes oder dem Website-Administrator erstellt werden, um zusätzliche Funktionen bereitzustellen.

Widgets und Argumente werden mit den Tags "widget" und "var" angegeben.

```
[widget=<widgetname>][var=<variable>]<variablenwert>[/var][/widget]
```

Widgetname	**Beschreibung**	**Variable/Argument**
clock	fügt eine digitale Uhr ein	military (1 oder 0) – 24-Stunden-Zeit im Gegensatz zu AM/PM verwenden
profile	zeigt eine Profil-Seitenleiste auf Seiten an, die Profile laden	
tagcloud	eine Tagcloud der Elemente einer Webseite anzeigen	count - – Anzahl der Objekte, die zurückgegeben werden sollen (Standardwert 24)
collections	Datenschutzgruppenauswahl für den aktuell angemeldeten Kanal	mode - je nach Modul einer von "conversation", "group", "abook"
suggestions	Vorschläge von Freunden für den aktuellen Kanal, in dem man angemeldet ist	
follow	zeigt ein Textfeld an, um einem anderen Kanal zu folgen	
notes	Privater Notizbereich für den aktuellen angemeldeten Kanal, wenn die Funktion „private_notes“ aktiviert ist	
savedsearch	Netzwerk-/Matrixsuche mit Speicherung – Sie müssen angemeldet sein und die Suchfunktion muss aktiviert sein	
filer	Ausgewählte abgelegte Elemente aus dem Netzwerk-/Matrix-Stream – Anmeldung erforderlich	
archive	Datumsbereich-Auswahl für Netzwerk- und Kanalseiten	wall – 1 oder 0, Beschränkung auf Pinnwandeinträge oder Netzwerk-/Matrix-Posts (Standard)

Widgetname	Beschreibung	Variable/Argument
fullprofile	identisch mit "profile"	
categories	Kategorienfilter (Kanalseite)	
tagcloud_wall	Tagcloud nur für Kanalseite	limit – Anzahl der zurückzugebenden Tags (Standardwert 50)
catcloud_wall	Kategorie-Cloud für Kanal-Seitenkategorien	limit – Anzahl der zurückzugebenden Kategorien (Standardwert 50)
affinity	Affinitätsregler für Netzwerkseite – Anmeldung erforderlich	
settings_menu	Seitenmenü für die Einstellungsseite, muss angemeldet sein	
design_tools	Menü für Design-Tools zum Erstellen von Webseiten, Anmeldung erforderlich	
findpeople	Tools, um andere Kanäle zu finden	
photo_albums	Liste der Fotoalben des aktuellen Seiteneigentümers mit Auswahlmenü	
vcard	Mini-Profil-Seitenleiste für die Person von Interesse (Seiteninhaber, was auch immer)	
dirsafemode	Auswahlwerkzeug für Verzeichnisse – nur auf Verzeichnisseiten	
dirsort	Auswahlwerkzeug für Verzeichnisse – nur auf Verzeichnisseiten	
dirtags	Verzeichnis-Tool – nur auf Verzeichnis-Seiten	
menu_preview	Vorschau eines Menüs – nur auf Menüseiten bearbeiten	
chatroom_list	Liste der Chatrooms für den Seiteninhaber	
bookmarkedchats	Liste der auf dieser Website für den aktuellen Hub gesammelten Chatrooms mit Lesezeichen	
suggestedchats	„interessante“ Chatrooms, die für den aktuellen Hub ausgewählt wurden	
item	zeigt ein einzelnes Webseitenelement nach Seiten- oder Seitentitel an	channel_id - Kanal, dem der Inhalt gehört, standardmäßig ist dies die profile_uid
		mid - message_id der anzuzeigenden Webseite (muss eine Webseite sein, kein Konversationselement)

Widgetname	Beschreibung	Variable/Argument
		title – URL-Seitentitel der Webseite (entweder „title“ oder „mid“ muss angegeben werden)
photo	display a single photo	src - URL des Fotos, muss http oder https sein
		zrl - Verwenden Sie einen authentifizierten zid-Link
		style – CSS-Stilstring
photo_rand	Zeigen Sie ein zufälliges Foto aus einem Ihrer Fotoalben an. Fotogenehmigungen werden berücksichtigt.	album – Name des Albums (dringend empfohlen, wenn Sie viele Fotos haben)
		scale – in der Regel 0 (Originalgröße), 1 (1024 Pixel), 2 (640 Pixel) oder 3 (320 Pixel)
		style – CSS-Stilstring
		channel_id – falls nicht Ihre eigene
random_block	Zeigt ein zufälliges Blockelement aus Ihrer Sammlung von Design-Tools für Webseiten an.	contains - nur Blöcke zurückgeben, die die Zeichenfolge „contains“ im Blocknamen enthalten
		channel_id - falls nicht Ihre eigene
tasklist	eine Aufgabe oder To-do-Liste für den aktuell angemeldeten Kanal bereitstellen	alle – abgeschlossene Aufgaben anzeigen, wenn „alle“ ungleich 0 ist
forums	eine Liste der verbundenen öffentlichen Foren mit nicht angezeigten Zählungen für den aktuell angemeldeten Kanal bereitstellen	
activity	eine Liste der Autoren von ungelesenen Netzwerkinhalten für den aktuell angemeldeten Kanal bereitstellen	
album	stellt ein Widget bereit, das ein vollständiges Fotoalbum aus Alben enthält, die dem Seiteninhaber gehören; dies kann zu groß sein, um in einem Seitenleistenbereich dargestellt zu werden, und wird am besten als Inhaltsbereich-Widget implementiert	album – Name des Albums
		title – optionaler Titel, falls nicht vorhanden, wird der Name des Albums verwendet
channel_activity	Widget, das einen Überblick über Kanäle bietet, die Ihre Aufmerksamkeit erfordern, und schnelle Links	

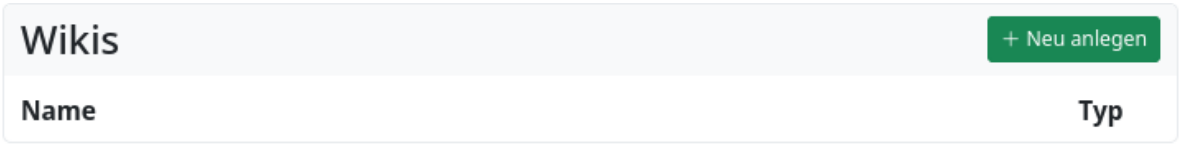
Widgetname	Beschreibung	Variable/Argument
	zu Inhalten, die Sie kürzlich erstellt oder bearbeitet haben	
hq_controls	Steuertasten für das HQ-Modul	
hq_messages	Schneller Zugriff auf Nachrichten, Direktnachrichten, mit Sternchen markierte Nachrichten (falls aktiviert) und Benachrichtigungen	
help_index	Index der Hilfeseiten	
new_member	nützliche Links für neue Mitglieder anzeigen, um ihnen den Einstieg zu erleichtern	
notifications	zeigt alle Arten von Benachrichtigungen an	
public_stream_tags	Öffentliche Stream-Tags in einer Cloud anzeigen	
wiki_list	ein Menü mit Links zu allen vorhandenen Wikis anzeigen	
zcard	Ihre Standard-Profilkarte mit Ihrem Titelbild	
activity_filters	Filter für den Netzwerk-Stream	
activity_order	sortieren Sie den Netzwerk-Stream nach dem Datum des Beitrags, dem letzten Kommentar oder dem Datum, an dem der Thread beendet wurde	

Wiki

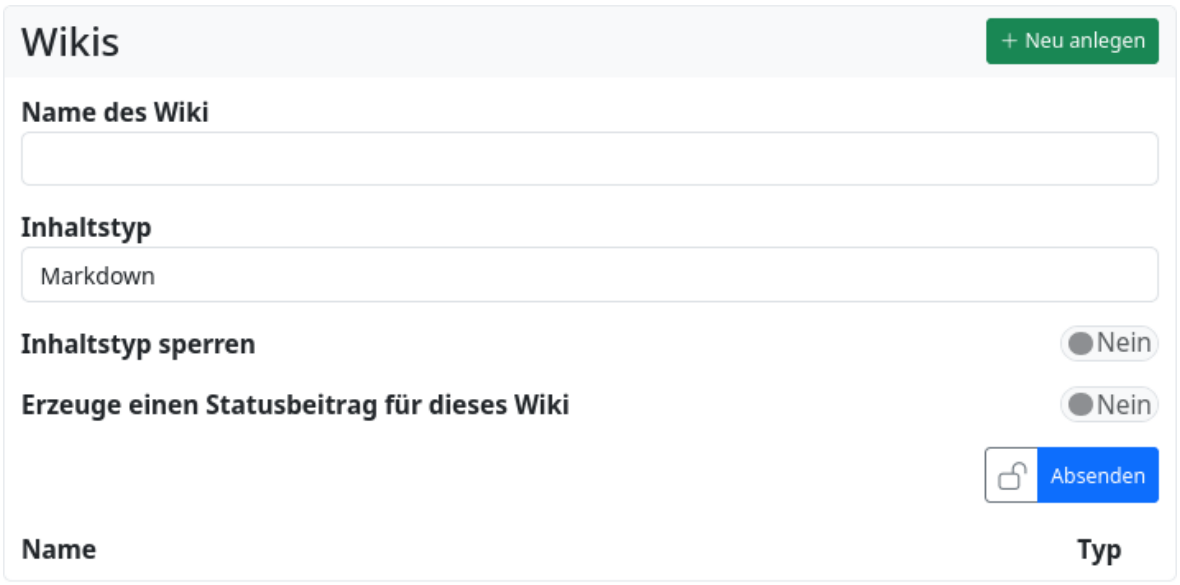
Mit der App "Wiki" können Sie Ihrer Webseite Wikis hinzufügen. Wiki-Seiten werden nicht föderiert und verbleiben auf dem eigenen Hub.

Die Wiki-App bietet eine einfache, klassische Wiki-Funktionalität. Wiki-Beiträge können als reiner Text, als Markdown-Text oder als bbCode-Text erstellt werden.

Der Aufruf der App "Wiki" führt zur Übersichtsseite der Wikis, wo die vorhandenen Wikis aufgelistet werden und wo Sie neue Wikis anlegen können.



Wenn Sie auf den Button "+ Neu anlegen" klicken, öffnet sich der Wiki-Editor.

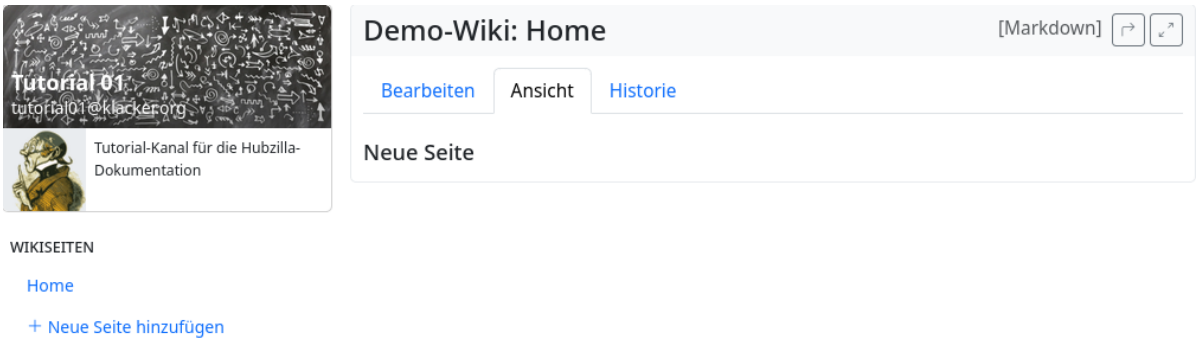


Sie müssen einen Namen für das Wiki vergeben. Außerdem können Sie den Inhaltstyp für das Wiki festlegen: Markdown, BBcode oder Text (Plaintext).

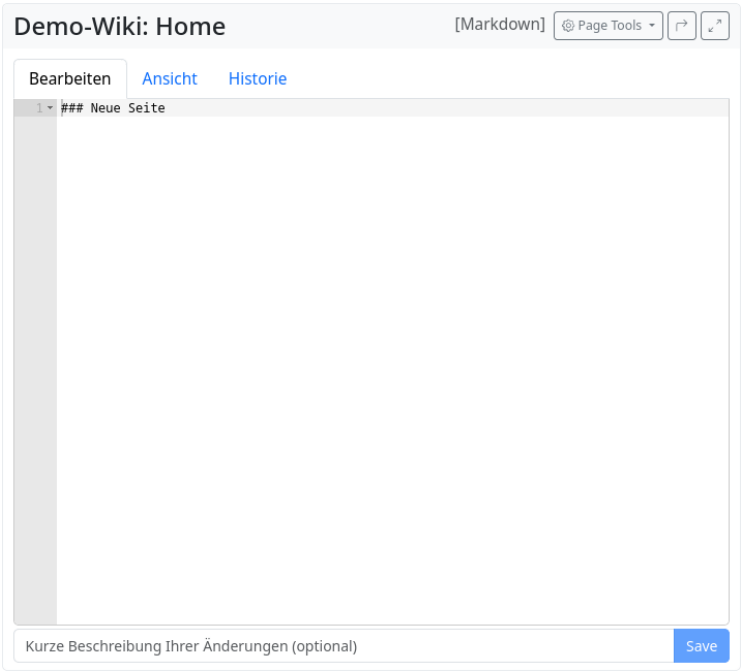
Über die Option "Inhaltstyp sperren" können Sie festlegen, ob generell sämtliche Einträge des WIkis in dem gewählten Grundformat erstellt werden, oder ob bei jedem Eintrag optional ein anderer Inhaltstyp gewählt werden kann.

Die Option "Erzeuge einen Statusbeitrag für dieses Wiki" ermöglicht es, dass die Erzeugung des WIkis von Ihrem Kanal geteilt wird, so dass sämtliche Verbindungen Kenntnis davon erhalten.

Durch Klicken auf "Absenden" wird das Wiki erstellt und es wird die erste Wiki-Seite angezeigt, welche automatisch den Namen "Home" erhält und lediglich den Text (als Heading formatiert) "Neue Seite" enthält.



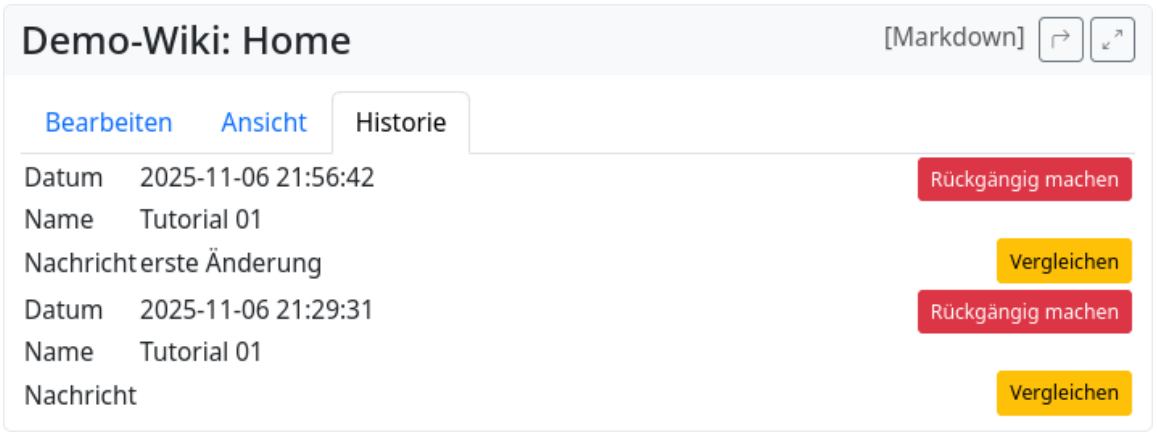
Die Seite weist (bei berechtigten Nutzern) drei Reiter auf: Bearbeiten, Ansicht, Historie. Wenn Sie nun auf den Reiter "Bearbeiten" wechseln, wird der Seiteneditor angezeigt, in welchem Sie die Seite bearbeiten können.



Sie können, nachdem Sie die Seite bearbeitet haben, im unteren Eingabefeld kurz vermerken, was Sie getan haben. Das ist zwar optional, wird aber empfohlen, weil damit das Änderungsprotokoll im Reiter "Historie" aussagekräftiger wird und Änderungen leichter nachzuvollziehen.

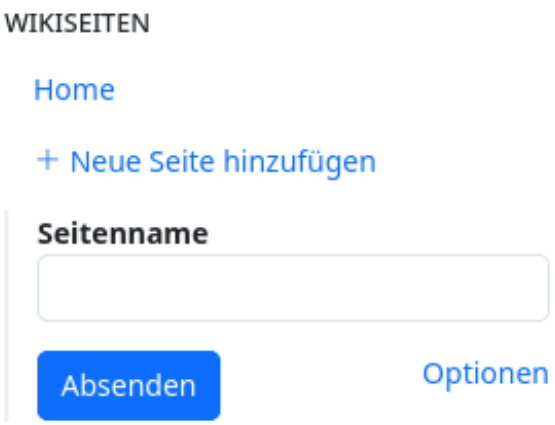
Haben Sie die Seite bearbeitet und die Bearbeitung mit "Save" gespeichert, erscheint der neue Inhalt auch im Reiter Ansicht.

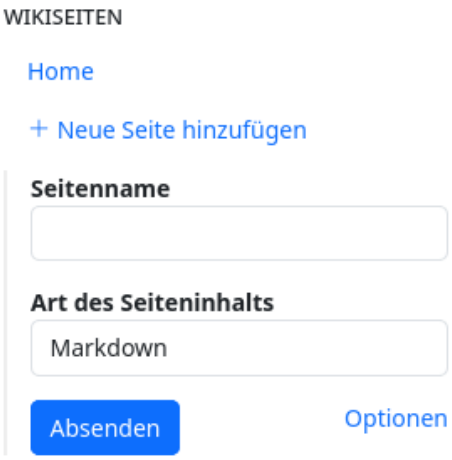
Der Reiter "Historie" dient dazu, den Änderungsverlauf der Seite anzuzeigen.



Dort können Sie Änderungen rückgängig machen und verschiedene Versionen miteinander vergleichen.

Neue Seiten werden durch Klick auf den Menüpunkt "+ Neue Seite hinzufügen" im Wikiseiten-Menü in der linken Seitenleiste erstellt.





Eine Liste der Wikis Ihrer Webpräsenz können Sie auf einer System-Ansicht oder einer selbst erstellten Webseite durch Hinzufügen des Widgets "wiki_list" hinzufügen.

Menüs II

Menüs sind zentrale und verbindende Element einer Web-Präsenz. Häufig möchte man ein Hauptmenü haben, das – vergleichbar mit dem Hauptmenü z.B. bei WordPress – die Navigation zu den verschiedenen Bereichen des Webauftritts ermöglicht und auf allen Seiten wiederkehrend an der selben Stelle vorhanden ist.

Ein solches Menü kann, vertikal gestaltet, gut in einer Seitenleiste eingebaut werden, sofern alle Seiten, wo es erscheinen soll, eine solche Seitenleiste im Layout vorsieht. Dafür kann man durchaus die von der Webseiten-App gebotene Menü-Funktion verwenden. Allerdings ist sie nicht besonders attraktiv gestaltet, sondern lediglich eine Link-Liste (ohne Aufzählungszeichen).

Für ein attraktiveres Menü oder ein horizontales Menü erstellt man am besten selbst ein als Block. Ein horizontales Menü als Hauptmenü hat den Vorteil, dass man es im Layout über dem eigentlichen Content platzieren kann und somit auch Layouts genutzt werden können, die über keine Seitenleiste verfügen.

Die einfachste Methode ist, den Menüblock mit BBcode zu gestalten. Dafür reiht man Links mittels `[url=<ZIEL_URL>]<LINKTEXT>[/url]` einfach nebeneinander.

Beispiel:

```
[url=https://pepecyb.hu]Link 1[/url] [url=https://hubzilla.hu]Link 2[/url] [url=https://klacker.org/channel/tutorial01]Link 3[/url]
```

Link 1 Link 2 Link 3

Der Nachteil bei dieser Methode ist, dass Links auf diese Weise immer in einem neuen Browserfenster bzw. Browsertab geöffnet werden.

Besser in Hinblick auf die Steuerung der Links lässt sich das Menü mit HTML erzeugen. Dafür reiht man Links mittels `<a href="<ZIEL_URL>" target="_self"><LINKTEXT></a>` einfach nebeneinander. Durch `target="_self"` wird erreicht, dass die gewählte Zielseite im selben Browsertab geöffnet wird.

Beispiel:

```
<a href="https://pepecyb.hu">Punkt 1</a> <a href="https://hubzilla.hu">Punkt 2</a> <a href="https://klacker.org/channel/tutorial01" target="_self">Punkt 3</a>
```

Punkt 1 Punkt 2 Punkt 3

Beachte: Um die Menüpunkte besser zu trennen, könnte man mehrere Leerzeichen zwischen die einzelnen href einfügen. Allerdings unterdrückt HTML traditionell mehrere aufeinander folgende Leerzeichen, weshalb hier geschützte Leerzeichen mit ` ` eingefügt werden müssen. Ein Trennzeichen kann auch eine gute Idee sein.

Das kann dann so aussehen:

Wesentlich flexibler und featurereicher kann man ein Menü mit dem Framework Bootstrap5 gestalten. Von Vorteil ist, dass Hubzilla selbst dieses Framework nutzt und es bereits eingebunden ist. Es ist also nicht erforderlich, es für die Verwendung in eigenen Blöcken selbst einzubinden.

Für ein horizontales Menü verwenden Sie einfach eine unsortierte Liste (`<ul></ul>`) und fügen die Klasse `.nav` zum `<ul>` Element hinzu: `<ul class="nav">`.

Den einzelnen Listenelementen (`<li></li>`) fügen Sie die Klasse `nav-item` hinzu: `<li class="nav-item">`.

Innerhalb der Listenelement-Tags fügen sie dem `<a href=>` Element noch die Klasse `nav-link` hinzu: `<a class="nav-link" href=>`.

Die Definition des Beispiel-Menüs würde dann z.B. so aussehen:

```
<ul class="nav">
  <li class="nav-item">
    <a class="nav-link" href="https://pepecyb.hu">Punkt 1</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="https://hubzilla.hu">Punkt 2</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="https://klacker.org/channel/tutori-
al01">Punkt 3</a>
  </li>
</ul>
```

Und als Ergebnis:

Ergänzen Sie die in `<ul>` verwendete Klasse `nav` um `justify-content-center`, wird das Menü zentriert dargestellt, mit `nav justify-content-end` rechtsbündig:

Sie können auch ein vertikales Menü mit diese Bootstrap-Klasse erzeugen: `<ul class="nav flex-column">`.

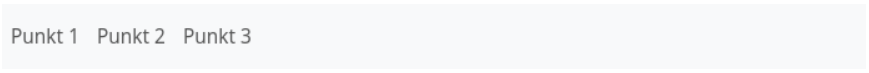
Noch moderner können Sie Ihr Menü gestalten, indem Sie die Klasse "`navbar`" verwenden. Sie kann in der Grundfunktion so verwendet werden, wie die nav-Klasse. Die Navigationsbar kann responsiv konfiguriert werden, so dass sie auf eine vertikale Darstellung ab einer bestimmten Bildschirmgröße umstellt. Dazu wird die Klasse mit

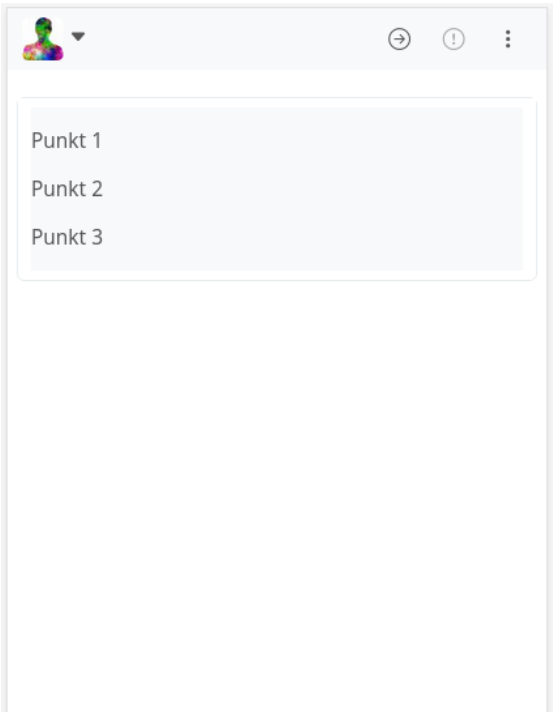
- `navbar-expand-sm` kleiner Bildschirm
- `navbar-expand-md` mittlerer Bildschirm
- `navbar-expand-lg` großer Bildschirm
- `navbar-expand-xl` sehr großer Bildschirm
- `navbar-expand-xxl`extrem großer Bildschirm

Außerdem kann die Hintergrundfarbe der Navigationsleiste unter Verwendung der Klassen für die Bootstrap-Hintergrundfarben festgelegt werden.

Das Beispielmenü als Navigationsleiste mit blauem Hintergrund (`bg-primary`), welche bei kleinen Bildschirmen in den vertikalen Modus umschaltet, würde also so erstellt werden:

```
<nav class="navbar navbar-expand-sm bg-light">

  <ul class="navbar-nav">
    <li class="nav-item">
      <a class="nav-link" href="https://pepecyb.hu">Punkt 1</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="https://hubzilla.hu">Punkt 2</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="https://klacker.org/channel/tuto-
rial01">Punkt 3</a>
    </li>
  </ul>

</nav>
```





Wenn Sie die Klassenangabe für das responsive Umschalten weglassen, wird die Navigationsleiste generell vertikal dargestellt.

Mit der Klasse `justify-content-center` werden die Menüpunkte in der Navigationsleiste zentriert dargestellt.

Verwenden Sie eine Hintergrundfarbe für die Leiste, legen Sie mit der Klasse `navbar-dark` fest, dass die Menüpunkte in weißer Schrift dargestellt werden, mit der Klasse `navbar-light` ist die Schrift schwarz.

Weitere Möglichkeiten finden Sie in der Bootstrap-Dokumentation.

In meinem Repo mit Gestaltungswerkzeugen und Vorlagen gibt es ein Python-Skript, welches den Code für eine Navigationsleiste mit Logo-Grafik erzeugt. Weitere Features müssen Sie dann von Hand in die entsprechende Klasse einfügen: gen_navbar.py

Der Aufruf erfolgt in der Konsole:

```
python3 gen_navbar.py OUTFILE LOGO_URL LOGO_LINK [LINK_TITLE LINK_URL]...
```

Für ein Beispiel habe ich ein Hauptmenü mit dem Skript erstellt. Es verwendet eine Logo-Datei, welche zur Profilseite des Kanals führt. Darauf folgen Links zur Homepage (eine Beispiel-Webseite), zu der Artikel-, der Kanalstream- und der Wiki-Seite.

```
python3 gen_navbar.py tutnavbar.txt https://klacker.org/cloud/tutorial01/misc/tutlogo.png https://klacker.org/profile/tutorial01 Home https://klacker.org/page/tutorial01/home Artikel https://klacker.org/articles/tutorial01 Kanal https://klacker.org/channel/tutorial01 Wikis https://klacker.org/wiki/tutorial01
```

Das Ergebnis

```
<nav class="navbar navbar-expand-lg bg-primary">
  <div class="container-fluid">
    <a class="navbar-brand" href="https://klacker.org/profile/tu-
torial01">
      <img src="https://klacker.org/cloud/tutorial01/misc/tutlo-
go.png" alt="Logo" height="30" class="d-inline-block align-text-
top">
    </a>
    <button class="navbar-toggler" type="button" data-bs-tog-
gle="collapse" data-bs-target="#navbarSupportedContent" aria-con-
trols="navbarSupportedContent" aria-expanded="false" aria-la-
bel="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedCon-
tent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
      <li class="nav-item"><a class="nav-link text-white"
href="https://klacker.org/page/tutorial01/home">Home</a></li>
      <li class="nav-item"><a class="nav-link text-white"
href="https://klacker.org/articles/tutorial01">Artikel</a></li>
      <li class="nav-item"><a class="nav-link text-white"
href="https://klacker.org/channel/tutorial01">Kanal</a></li>
      <li class="nav-item"><a class="nav-link text-white"
href="https://klacker.org/wiki/tutorial01">Wikis</a></li>
      </ul>
    </div>
  </div>
</nav>
```

wurde als Block "main_menu" genutzt. Diesen Block habe ich dann auf allen "beteiligten" Seiten (Profil, Artikel, Kanal, Wiki) über den Inhaltsbereich geschoben und in der Layout-Datei für die Demo-Homepage ebenfalls über den Inhaltsbereich `$content`:

```
[region=aside]
[widget=notifications][/widget]

[/region]
[region=content]
[block]main_menu[/block]
$content

[/region]
[region=right_aside]
[widget=profile][/widget]
[widget=articles_categories][/widget]

[/region]
```

Einblendbare Seitenleiste

Je nach Layout-Vorlage verfügen Ihre Webseiten über eine oder zwei Seitenleisten, in welchen Sie Blöcke und Menüs unterbringen können. Mittels Bootstrap haben Sie die Möglichkeit, eine weitere Seitenleiste zu nutzen, die nur bei Bedarf als Overlay eingeblendet wird (Sie sind nicht nur auf Seitenleisten beschränkt, sondern können mit dieser Funktionalität auch Header- und Footerleisten erstellen).

Dafür nutzen Sie die Bootstrap-Klasse `offcanvas`.

Um die Offcanvas-Seitenleiste mit einem Button bei Bedarf zu öffnen, müssen sie ihr ein Id zuweisen.

Sie erzeugen in einem Block einen `<div>`-Container der Klasse `offcanvas`:

```
<div class="offcanvas offcanvas-start" id="<CONTAINER-ID>">
  ...
</div>
```

Der Inhalt der Sidebar kann mit einem Header versehen werden:

```
<div class="offcanvas-header">
   <HEADER>
</div>
```

Außerdem benötigt die Seitenleiste natürlich Inhalt, welcher in einen Container der Klasse offcanvas-body eingebaut wird:

```
<div class="offcanvas-body">
  ...
</div>
```

Damit die Seitenleiste eingeblendet werden kann, bietet es sich an, einen Button zu verwenden. Meist werden solche einblendbaren Seitenleisten über einen Klick auf ein "Hamburger-Symbol" geöffnet.

Es muss also eine `button`-Klasse erzeugt werden, wobei der Button den Container mit der vergebenen `Id` öffnet:

```
<button class="btn" type="button" data-bs-toggle="offcanvas" data-
bs-target="<CONTINER-ID>">

</button>
```

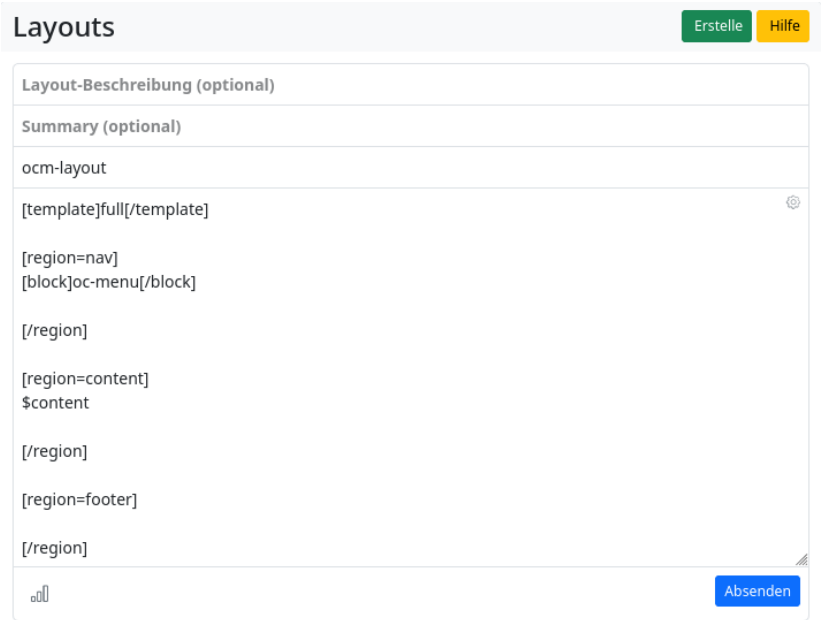
Wenn Sie diesen Block nun z.B. in den nav-Bereich ihres Layouts einbetten, wird in der Navigationsleiste der Button mit dem Symbol angezeigt und bei Klick auf den Button wird die Seitenleiste eingeblendet.

Weitere Features zur Offcanvas-Klasse können Sie in der Bootstrap-Dukumentation nachlesen: Offcanvas.

Als Beispiel erstelle ich im Folgenden eine Beispiel-Webseite mit einem Layout vom Typ "full" (Navigationsbereich, Inhaltsbereich, Footerbereich). Die Webseite selbst wird mit eine Platzhalter-Text befüllt. In den Navigationsbereich wird dann der Off-canvas-Block, den ich ebenfalls erstelle, eingesetzt.

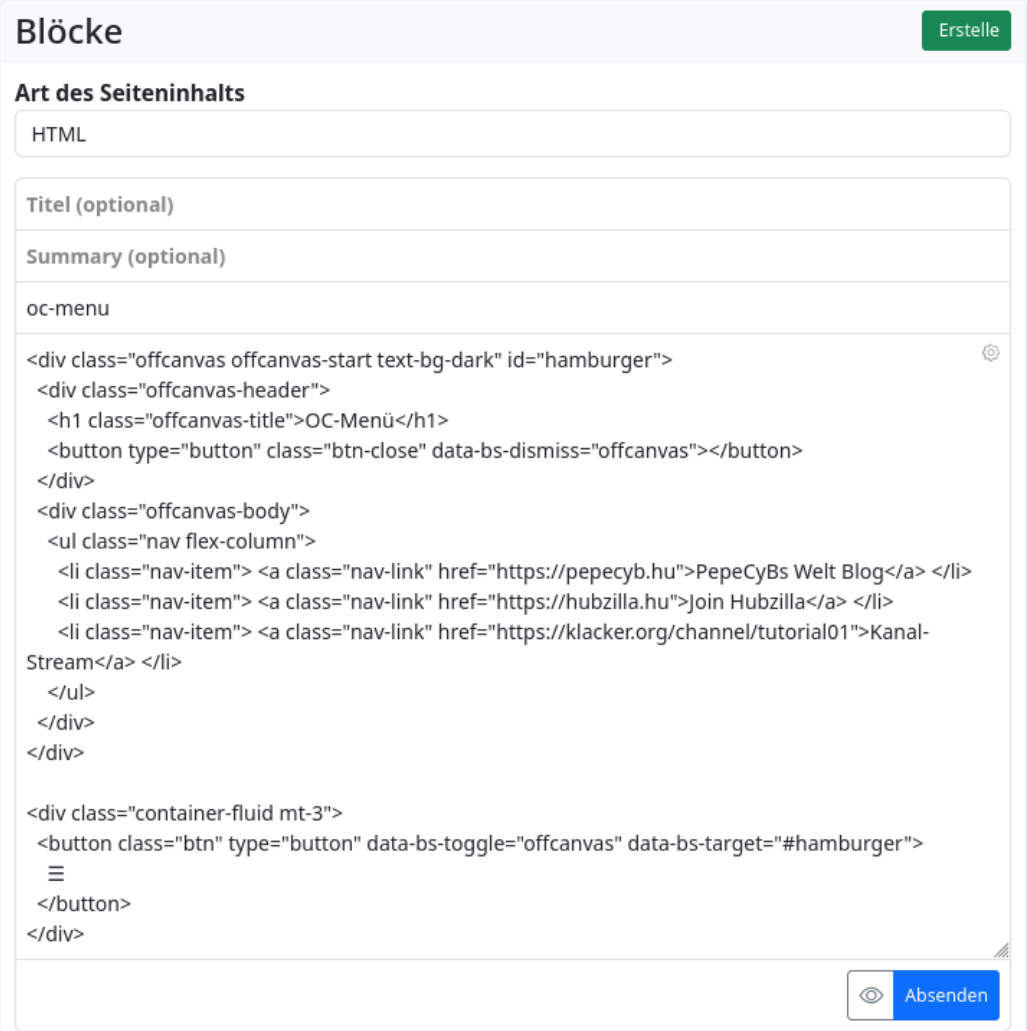
Der Block mit dem Offcanvas-Menü erhält den Namen "oc-menu". Und damit sieht das Layout (mit dem Namen "ocm-layout") so aus:

```
[template]full[/template]

[region=nav]

[block]oc-menu[/block]

[/region]

[region=content]

$content

[/region]

[region=footer]

[/region]
```



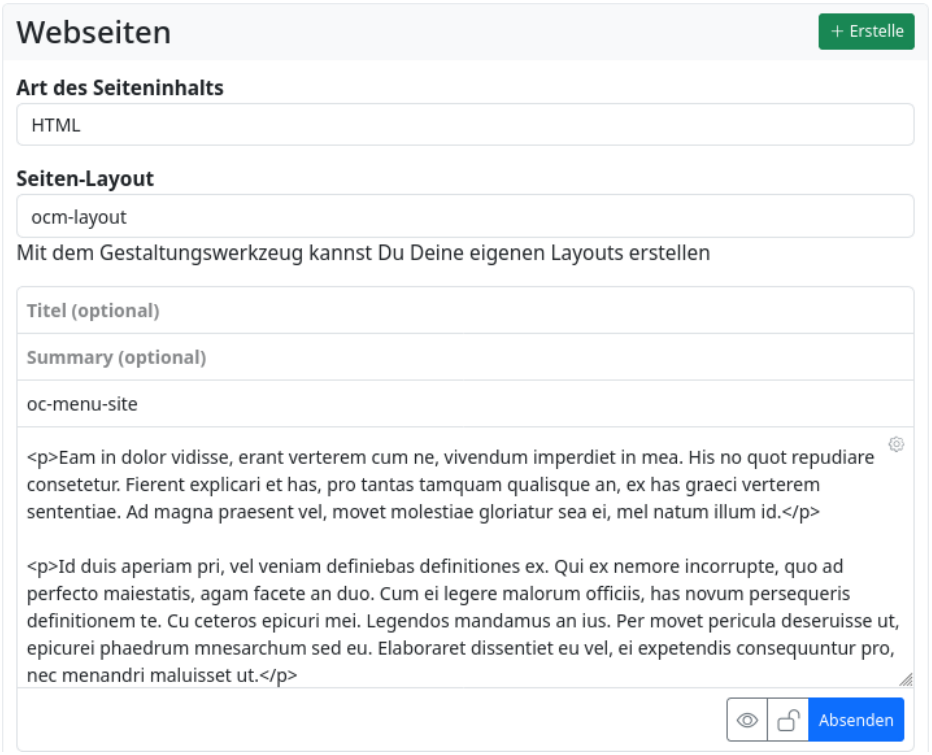
Nun erstelle ich den Block "oc-menu" (im dunklen Design), der einige Beispiel-Links enthält und als Header (Heading 1) den Text "OC-Menü":

```
<div class="offcanvas offcanvas-start text-bg-dark" id="hambur-
ger">

  <div class="offcanvas-header">

    <h1 class="offcanvas-title">OC-Menü</h1>

    <button type="button" class="btn-close" data-bs-dismiss="off-
canvas"></button>

  </div>

  <div class="offcanvas-body">

    <ul class="nav flex-column">

      <li class="nav-item"> <a class="nav-link" href="https://pe-
pecyb.hu">PepeCyBs Welt Blog</a> </li>

      <li class="nav-item"> <a class="nav-link" href="https://hub-
zilla.hu">Join Hubzilla</a> </li>

      <li class="nav-item"> <a class="nav-link" href="https://kla-
cker.org/channel/tutorial01">Kanal-Stream</a> </li>

    </ul>

  </div>

</div>



<div class="container-fluid mt-3">

  <button class="btn" type="button" data-bs-toggle="offcanvas"
data-bs-target="#hamburger">



  </button>

</div>
```

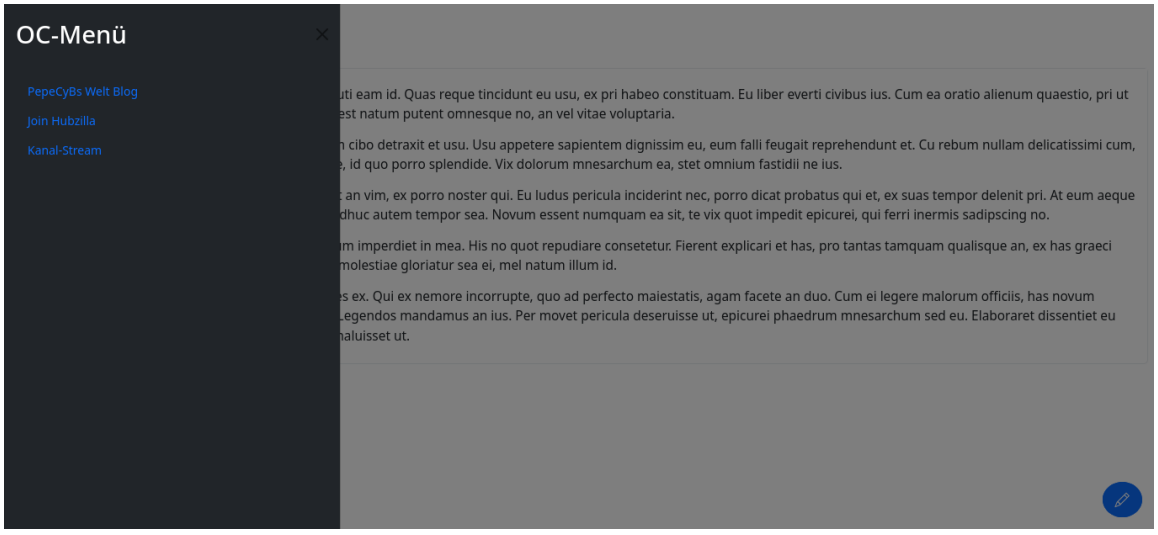


Und erzeuge eine Beispielseite "oc-menu-site", für welche ich das Layout auf "ocm-layout" festlege und welches den Platzhaltertext enthält:

```
<p>Lorem ipsum dolor sit amet, quem consulatu persecuti eam id.
Quas reque tincidunt eu usu, ex pri habeo constituam. Eu liber
everti civibus ius. Cum ea oratio alienum quaestio, pri ut quod
stet expetenda. Minim aeterno delicata vel ea, est natum putent
omnesque no, an vel vitae voluptaria.</p>

<p>Est case veritus ex. Ea vivendum sapientem vim, enim cibo de-
traxit et usu. Usu appetere sapientem dignissim eu, eum falli feu-
gait reprehendunt et. Cu rebum nullam delicatissimi cum, utinam
laoreet vel no. Et eos odio copiosae splendide, id quo porro
splendide. Vix dolorum mnesarchum ea, stet omnium fastidii ne
ius.</p>

...
```



Die Seite bietet nun eine einblendbare Seitenleiste:



Bildergalerie

Hubzilla bietet mit den Apps "Fotos" und "Galerie" zwei Möglichkeiten, Bilder anzuzeigen. Allerdings sind beide Apps von der Funktion und der Gestaltung her eher simpel und nicht wirklich gut nutzbar.

Besser verwendbar ist das Widget "Album".

Für eine wirklich gut nutzbare Möglichkeit, Bilder aus einem bestimmten Verzeichnis anzuzeigen, empfiehlt es sich, einen entsprechenden Block selbst zu erstellen.

Um die Handhabung so einfach wie möglich zu machen und ihn flexibel für verschiedene Bilderordner einsetzen zu können, sollte der Block selbst alle Bild-Dateien aus einem Verzeichnis, dessen URL man angibt, auslesen und für die Galerie verwenden.

In der Regel werden wir Bilder aus der eigenen Cloud des Kanals verwenden. Es ist aber auch möglich, Bilde zu nutzen, die an anderer Stelle vorliegen. Sollten sie auf einem anderen Server liegen, bei welchem CORS greift und die Bilder so nicht abgefragt werden können, soll es die Möglichkeit geben, die URLS der einzelnen Bilder einzeln in ein Array einzutragen.

Außerdem soll ein Statusanzeige existieren, die anzeigt, dass die Bilder geladen werden, falls dieser Vorgang aufgrund einer größeren Zahl von Bildern länger dauert. Außerdem sollen in der Statusanzeige Fehlermeldungen erscheinen, falls der Block falsch genutzt wurde (Fehler bei der URL etc.). Dem `<div>` Container geben wir die `Id` "`status`".

Die Bilder sollen dann in einen `<div>` Container der Bootstrap-Klasse `grid` eingefügt werden, welchem wir die `Id` "`galleryRow`" geben.

Einfache Galerie

```
<script>

const baseUrl = '<BASIS_URL_MIT_DEN_BILDERN>'; // HIER die URL
eintragen

/* Falls CORS greift, in fallbackImages[] sämtliche Bild-URLs ma-
nuell eintragen */
const fallbackImages = [
  // '<URL_ZUM_EINZELBILD>',
];

const galleryRow = document.getElementById('galleryRow');
const status = document.getElementById('status');

function looksLikeImage(u){
  return /\.(jpe?g|png|gif|webp|bmp|svg)(\?.*)?$/i.test(u);
}

function joinUrl(base, path){
  try{ return new URL(path, base).href; }catch(e){ return base +
path; }
}

function clearGallery(){ galleryRow.innerHTML = ''; }

function addCard(src){
  const col = document.createElement('div');
  col.className = 'col-6 col-sm-4 col-md-3 col-lg-2';
  const a = document.createElement('a');
  a.href = src; a.target = '_blank'; a.rel = 'noopener'; a.class-
Name = 'thumb-link';
  const img = document.createElement('img');
  img.src = src; img.alt = src.split('/').pop().split('?')[0];
img.className = 'thumb';
  a.appendChild(img); col.appendChild(a); galleryRow.append-
Child(col);
}

async function fetchImagesFromUrl(url){
  const res = await fetch(url, { mode:'cors' });
  if(!res.ok) throw new Error('Abruf fehlgeschlagen: ' + res.sta-
tus);
  const text = await res.text();
  const found = new Set();
  // <img src="...">
```

```
  const imgRegex = /<img[^>]+src=["']([^"']+)["'][^>]*>/gi;
  let m;
  while((m = imgRegex.exec(text)) !== null) if(looksLikeIma-
ge(m[1])) found.add(joinUrl(url, m[1]));
  // <a href="..."> (typisch für Verzeichnis-Listings)
  const aRegex = /<a[^>]+href=["']([^"']+)["'][^>]*>/gi;
  while((m = aRegex.exec(text)) !== null) if(looksLikeImage(m[1]))
found.add(joinUrl(url, m[1]));
  return Array.from(found);
}

(async function init(){
  clearGallery();
  if(!baseUrl || !baseUrl.trim()){ status.textContent = 'Keine Ba-
sis-URL gefunden. Bitte im Script eintragen.'; return; }
  status.textContent = 'Lade Bilder von ' + baseUrl;
  try{
    const imgs = await fetchImagesFromUrl(baseUrl);
    if(imgs.length){
      imgs.forEach(u => addCard(u));
      status.textContent = `Bilder in der Galerie:
${imgs.length}`;
      return;
    } else {
      status.textContent = 'Keine Bilder auf der Seite gefunden.
Prüfe URL oder nutze fallbackImages.';
    }
  }catch(err){
    console.warn('Ladefehler:', err);
    status.textContent = 'Automatisches Laden fehlgeschlagen (mög-
licher CORS-/Netzwerkfehler). Nutze fallbackImages.';
  }

  if(fallbackImages.length === 0){
    status.textContent = 'Keine Bilder verfügbar. Entweder CORS
auf Zielseite aktivieren oder fallbackImages befüllen.';
    return;
  }
  fallbackImages.forEach(u => addCard(joinUrl(baseUrl || loca-
tion.href, u)));
  status.textContent = `Galerie geladen (Fallback): ${fallbackIma-
ges.length}`;
})();

</script>
```

Der eigentlich HTML-Code zur Darstellung sieht so aus:

```
<div class="container gallery-container py-3">
  <div id="status" class="mb-2 text-muted">Lade…</div>
  <div id="galleryRow" class="row g-3" aria-live="polite"></div>
</div>
```

Und eine passende Inline-CSS-Definition könnte so aussehen:

```
<style>
  :root{ --thumb-size:180px; }
  body, .gallery-container { margin:0; background: inherit; }
  .thumb{ display:block; width:100%; height:var(--thumb-size); ob-
ject-fit:cover; border-radius:.375rem; }
  .thumb-link{ display:block; overflow:hidden; height:var(--thumb-
size); }
  @media (max-width:480px){ :root{ --thumb-size:120px; } }
</style>
```

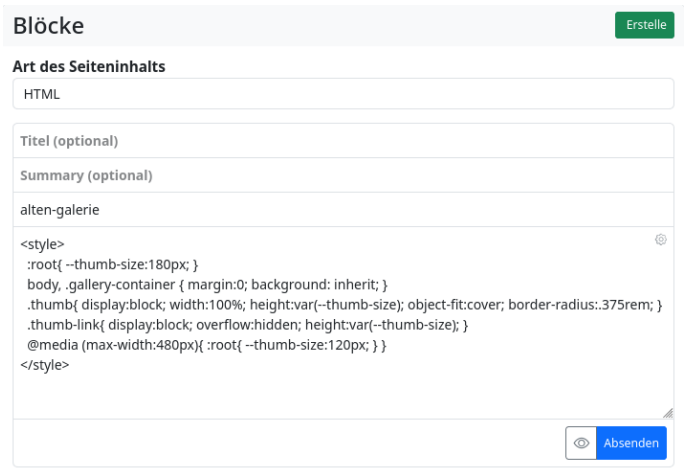
Der gesamte Blockinhalt vom Typ HTML wäre also dieser:

```
<style>
  :root{ --thumb-size:180px; }
  body, .gallery-container { margin:0; background: inherit; }
  .thumb{ display:block; width:100%; height:var(--thumb-size); ob-
ject-fit:cover; border-radius:.375rem; }
  .thumb-link{ display:block; overflow:hidden; height:var(--thumb-
size); }
  @media (max-width:480px){ :root{ --thumb-size:120px; } }
</style>


<div class="container gallery-container py-3">
  <div id="status" class="mb-2 text-muted">Lade…</div>
  <div id="galleryRow" class="row g-3" aria-live="polite"></div>
</div>


<script>

const baseUrl = '<BASIS_URL_MIT_DEN_BILDERN>'; // HIER die URL
eintragen

/* Falls CORS greift, in fallbackImages[] sämtliche Bild-URLs ma-
nuell eintragen */
const fallbackImages = [
  // '<URL_ZUM_EINZELBILD>',
];
```

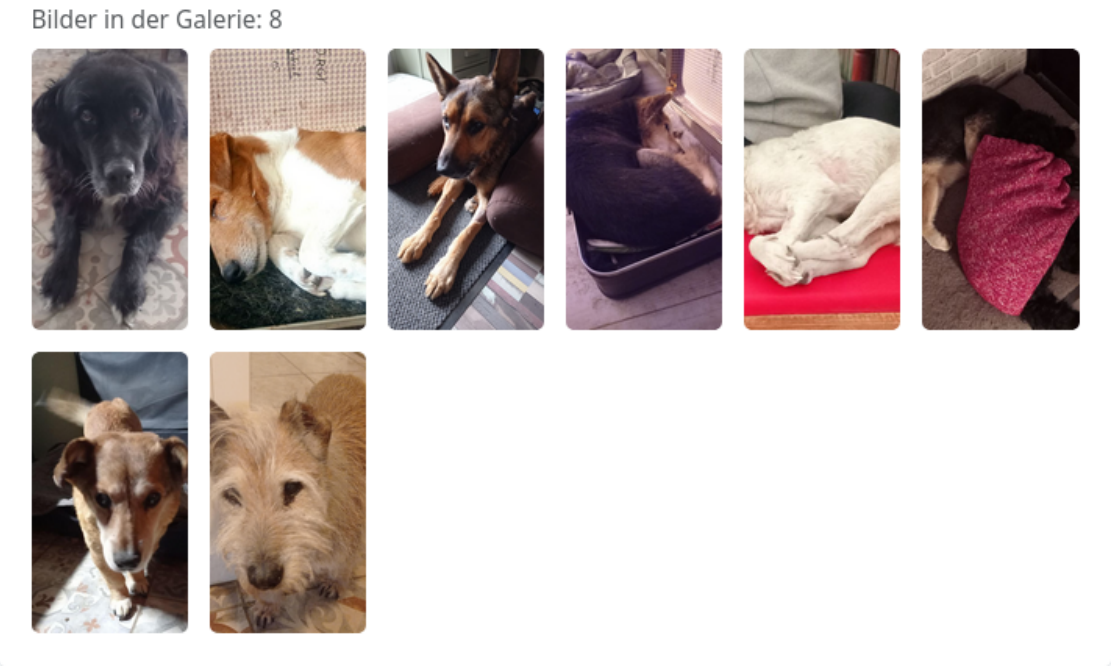
```
const galleryRow = document.getElementById('galleryRow');
const status = document.getElementById('status');

function looksLikeImage(u){
  return /\.(jpe?g|png|gif|webp|bmp|svg)(\?.*)?$/i.test(u);
}

function joinUrl(base, path){
  try{ return new URL(path, base).href; }catch(e){ return base +
path; }
}

function clearGallery(){ galleryRow.innerHTML = ''; }

function addCard(src){
  const col = document.createElement('div');
  col.className = 'col-6 col-sm-4 col-md-3 col-lg-2';
  const a = document.createElement('a');
  a.href = src; a.target = '_blank'; a.rel = 'noopener'; a.class-
Name = 'thumb-link';
  const img = document.createElement('img');
  img.src = src; img.alt = src.split('/').pop().split('?')[0];
img.className = 'thumb';
  a.appendChild(img); col.appendChild(a); galleryRow.append-
Child(col);
}

async function fetchImagesFromUrl(url){
  const res = await fetch(url, { mode:'cors' });
  if(!res.ok) throw new Error('Abruf fehlgeschlagen: ' + res.sta-
tus);
  const text = await res.text();
  const found = new Set();
  // <img src="...">
  const imgRegex = /<img[^>]+src=["']([^"']+)["'][^>]*>/gi;
  let m;
  while((m = imgRegex.exec(text)) !== null) if(looksLikeIma-
ge(m[1])) found.add(joinUrl(url, m[1]));
  // <a href="..."> (typisch für Verzeichnis-Listings)
  const aRegex = /<a[^>]+href=["']([^"']+)["'][^>]*>/gi;
  while((m = aRegex.exec(text)) !== null) if(looksLikeImage(m[1]))
found.add(joinUrl(url, m[1]));
  return Array.from(found);
}

(async function init(){
  clearGallery();
  if(!baseUrl || !baseUrl.trim()){ status.textContent = 'Keine Ba-
```

```
sis-URL gefunden. Bitte im Script eintragen.'; return; }
  status.textContent = 'Lade Bilder von ' + baseUrl;
  try{
    const imgs = await fetchImagesFromUrl(baseUrl);
    if(imgs.length){
      imgs.forEach(u => addCard(u));
      status.textContent = `Bilder in der Galerie:
${imgs.length}`;
      return;
    } else {
      status.textContent = 'Keine Bilder auf der Seite gefunden.
Prüfe URL oder nutze fallbackImages.';
    }
  }catch(err){
    console.warn('Ladefehler:', err);
    status.textContent = 'Automatisches Laden fehlgeschlagen (mög-
licher CORS-/Netzwerkfehler). Nutze fallbackImages.';
  }

  if(fallbackImages.length === 0){
    status.textContent = 'Keine Bilder verfügbar. Entweder CORS
auf Zielseite aktivieren oder fallbackImages befüllen.';
    return;
  }
  fallbackImages.forEach(u => addCard(joinUrl(baseUrl || loca-
tion.href, u)));
  status.textContent = `Galerie geladen (Fallback): ${fallbackIma-
ges.length}`;
})();

</script>
```

Um den Block nun in der Praxis zu testen, habe ich für ein Beispiel den Bilderordner "Die Alten" mit Bildern aktueller und ehemaliger Bewohner unseres Hunde-Altersruhesitzes gewählt und die URL in den Block eingetragen:



Die Galerie sieht nun so aus, wenn der Block dargestellt wird (ein Klick auf ein Bild öffnet dieses im Original in einem neuen Tab):



Erweiterte Galerie mit Zoom-Effekt

Das ist schon sehr schön. Aber weil wir ohnehin schon dabei sind, peppen wir den Block noch etwas weiter auf. Die Bilder werden quadratisch beschnitten,ihre Ecken leicht abgerundet und sie werden ein Stück weit gezoomt, sobald sich der Mauszeiger über ihnen befindet. Auf jedem Bild wird als Overlay der Dateiname angezeigt. Ein Klick auf ein Bild öffnet dieses wiederum in einem neuen Browser-Tab.

Der Code für einen solchen Block sieht dann so aus:

```
<style>
  :root{
    --gap: 0.75rem;
    --thumb-size: 200px;
    --zoom-scale: 1.8;
    --transition: 300ms;
  }

  body, .gallery-container {
    margin: 0;
    font-family: system-ui, -apple-system, "Segoe UI", Roboto,
"Helvetica Neue", Arial;
    color: #ddd;
    background: inherit;
  }

  .page-title {
    color: inherit;
    margin-bottom: 0.75rem;
  }

  .gallery {
    display: grid;
```

```
    grid-template-columns: repeat(auto-fill, minmax(var(--thumb-
size), 1fr));
    gap: var(--gap);
  }

  .card-thumb {
    position: relative;
    overflow: hidden;
    border-radius: 0.5rem;
    height: var(--thumb-size);
    display: block;
    text-decoration: none;
    background: rgba(0,0,0,0.06);
  }

  .card-thumb img{
    width:100%;
    height:100%;
    object-fit:cover;
    transition: transform var(--transition) ease, filter var(-
-transition) ease;
    transform-origin: center center;
    will-change: transform;
    display:block;
    user-select:none;
    pointer-events:none;
  }

  .card-thumb:hover img{
    filter: brightness(1.05);
```

```
  }

  .card-thumb:hover img.zoomed{
    transform: scale(var(--zoom-scale));
  }

  .meta {
    position:absolute;
    left:0.5rem;
    bottom:0.5rem;
    background:rgba(0,0,0,0.4);
    color:#fff;
    padding:0.25rem 0.5rem;
    border-radius:0.375rem;
    font-size:0.8rem;
    max-width:90%;
    white-space:nowrap;
    overflow:hidden;
    text-overflow:ellipsis;
  }

  @media (max-width:480px){
    :root{ --thumb-size:120px; }
  }
</style>


<div class="container gallery-container py-3">
    <div id="status" class="mb-2 text-muted">Lade Bilder…</div>
    <div id="gallery" class="gallery" aria-live="polite"></div>
```

```
</div>



<script>


const baseUrl = "<BASIS_URL_DER_BILDER>"; // <- HIER die URL mit
den Bildern einfügen

const gallery = document.getElementById('gallery');

const status = document.getElementById('status');


/* Manuelle Liste als Fallback */

let images = [];


function looksLikeImage(url){

  return /\.(jpe?g|png|gif|webp|bmp|svg)(\?.*)?$/i.test(url);

}


function joinUrl(base, path){

  try { return new URL(path, base).href; } catch(e){ return base +
path; }

}


function addImageCard(src, filename){

  const a = document.createElement('a');

  a.href = src;

  a.className = 'card-thumb';

  a.target = '_blank';

  a.rel = 'noopener';


  const img = document.createElement('img');
```

```
  img.src = src;
  img.alt = filename || '';
  img.classList.add('zoomed');

  const meta = document.createElement('div');
  meta.className = 'meta';
  meta.textContent = filename || '';

  a.appendChild(img);
  a.appendChild(meta);
  gallery.appendChild(a);

  a.addEventListener('mousemove', (e) => {
    const rect = img.getBoundingClientRect();
    const x = ((e.clientX - rect.left) / rect.width) * 100;
    const y = ((e.clientY - rect.top) / rect.height) * 100;
    img.style.transformOrigin = `${x}% ${y}%`;
  });

  a.addEventListener('mouseleave', () => {
    img.style.transformOrigin = 'center center';
  });
}

async function fetchImagesFromUrl(url){
  try{
    const res = await fetch(url, { mode:'cors' });
    if(!res.ok) throw new Error('Abrufen fehlgeschlagen: ' +
res.status);
    const text = await res.text();
```

```
    const imgRegex = /<img[^>]+src=["']([^"']+)["'][^>]*>/gi;
    let match;
    while((match = imgRegex.exec(text)) !== null){
      const src = match[1];
      if(looksLikeImage(src)) images.push(joinUrl(url, src));
    }

    const aRegex = /<a[^>]+href=["']([^"']+)["'][^>]*>/gi;
    while((match = aRegex.exec(text)) !== null){
      const href = match[1];
      if(looksLikeImage(href)) images.push(joinUrl(url, href));
    }

    images = Array.from(new Set(images));
    return images;
  }catch(err){
    console.warn('Fehler beim Laden:', err);
    throw err;
  }
}

(async function init(){
  gallery.innerHTML = '';
  status.textContent = 'Lade Bilder von ' + baseUrl;
  try{
    if(baseUrl && baseUrl.trim() !== ""){
      const found = await fetchImagesFromUrl(baseUrl);
      if(found.length){
```

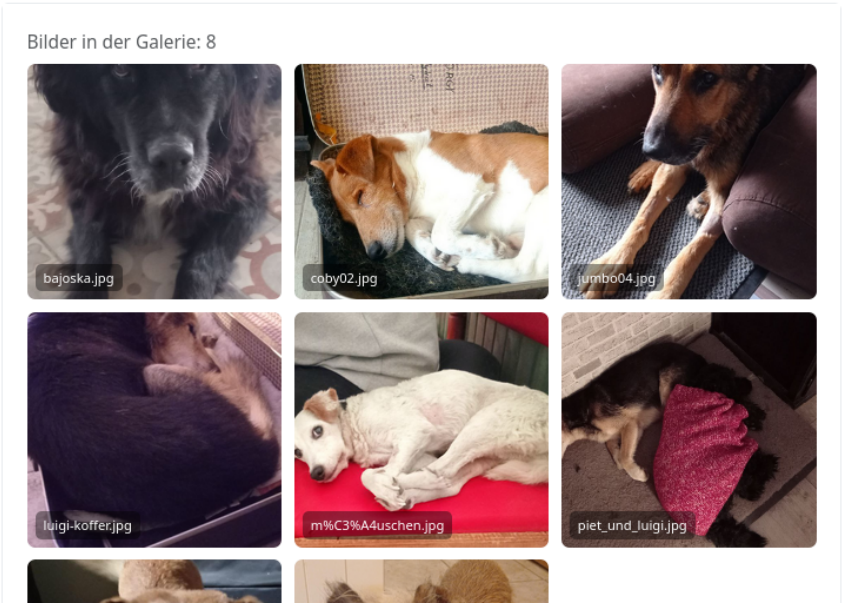
```
        found.forEach(u => addImageCard(u,
u.split('/').pop().split('?')[0]));

        status.textContent = 'Bilder in der Galerie: ' +
found.length;

        return;

      } else {

        status.textContent = 'Keine Bilder auf der Seite gefunden.
Fallback zur manuellen Liste.';

      }

    } else {

      status.textContent = 'Keine baseUrl angegeben. Fallback zur
manuellen Liste.';

    }

  }catch(e){

    status.textContent = 'Automatisches Laden fehlgeschlagen (CORS
oder kein Directory-Listing). Verwende manuelle image-Liste.';

  }


  // Fallback manuell befüllen:

  if(images.length === 0){

    images = [

      // Beispiel:

      // 'https://example.com/images/beispiel1.jpg',

      // 'https://example.com/images/beispiel2.png'

    ];

  }


  if(images.length === 0){

    status.textContent = 'Keine Bilder verfügbar. Bitte baseUrl
setzen oder images[] manuell befüllen.';

    return;

  }
```

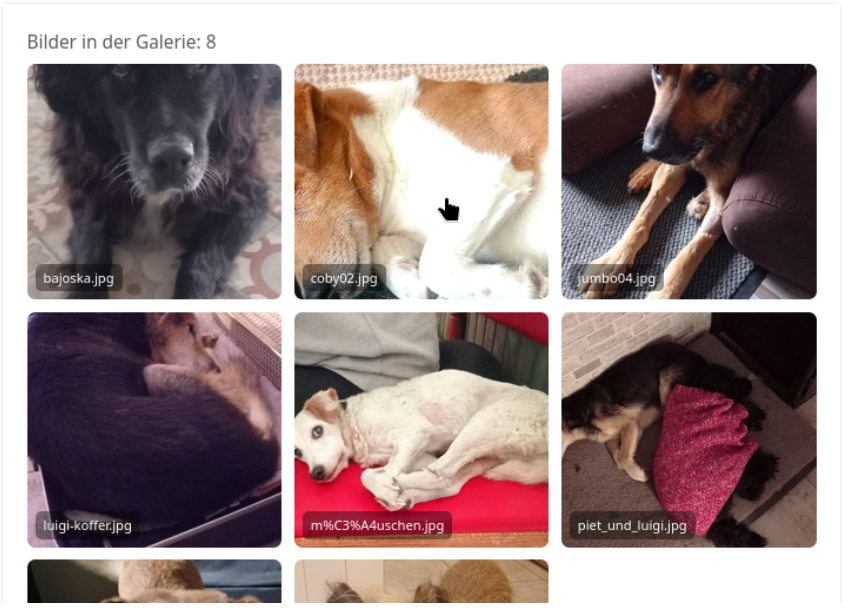
```
  images.forEach(u => addImageCard(joinUrl(baseUrl || loca-
tion.href, u), (u.split('/').pop().split('?')[0])));

  status.textContent = 'Galerie geladen (manuelle Liste).';

})();

</script>
```

Nun zum Testen wieder das Verzeichnis mit den Hundesenioren eingebaut:

```
const baseUrl = "https://klacker.org/cloud/tutorial01/Die%20Al-
ten"; // <- HIER die URL mit den Bildern einfügen
```

Und der Galerie-Block sieht dann so aus:





Galerie als Carousel

Als dritte Variante wäre auch ein Block mit einem Bilder-Carousel denkbar. Der Code dafür ist von der Basis her ähnlich:

```
<style>
  body, .gallery-container { margin:0; background: inherit; }
  .carousel-img {
    width:100%;
    height:60vh;
    object-fit:contain;
    background: #111;
  }
  @media (max-width:576px){
    .carousel-img { height:40vh; }
  }
</style>



<div class="container gallery-container py-3">
  <div id="status" class="mb-2 text-muted">Lade…</div>

  <!-- Carousel-Wrapper: wird dynamisch befüllt -->
  <div id="carouselWrapper" style="display:none;">
    <div id="imageCarousel" class="carousel slide" data-bs-
ride="carousel">
      <div class="carousel-indicators" id="carouselIndica-
tors"></div>
      <div class="carousel-inner" id="carouselInner"></div>
      <button class="carousel-control-prev" type="button" data-bs-
target="#imageCarousel" data-bs-slide="prev">
```

```
        <span class="carousel-control-prev-icon" aria-hid-
den="true"></span>

        <span class="visually-hidden">Vorheriges</span>

      </button>

      <button class="carousel-control-next" type="button" data-bs-
target="#imageCarousel" data-bs-slide="next">

        <span class="carousel-control-next-icon" aria-hid-
den="true"></span>

        <span class="visually-hidden">Nächstes</span>

      </button>

    </div>

  </div>

</div>




<script>


const baseUrl = '<BASIS_URL_MIT_DEN_BILDERN>'; // <- Hier die URL
mit den Bildern eintragen


/* Fallback-Liste falls CORS greift */

const fallbackImages = [

  // '<URL_DES_BILDES>>',

];


const status = document.getElementById('status');

const carouselWrapper = document.getElementById('carouselWrap-
per');

const carouselIndicators = document.getElementById('carouselIndi-
cators');

const carouselInner = document.getElementById('carouselInner');
```

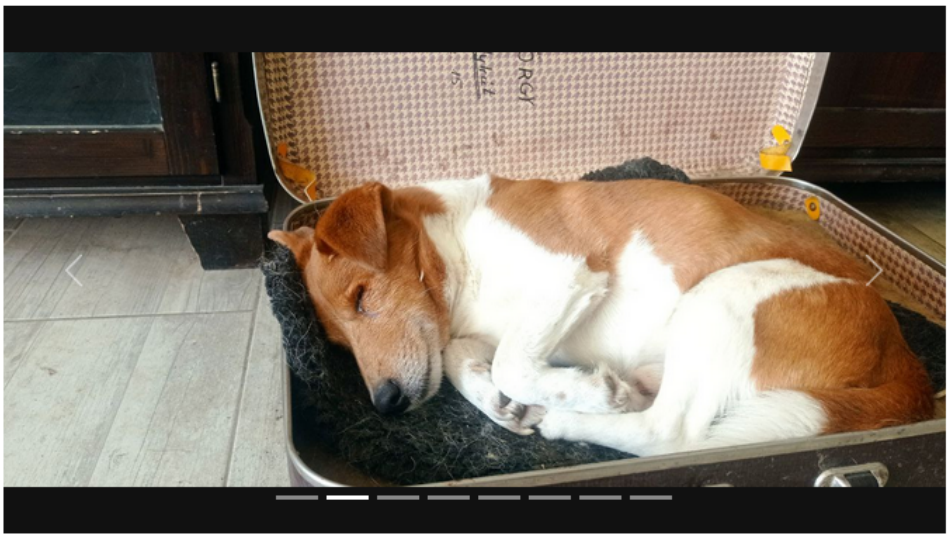
```
function looksLikeImage(u){
  return /\.(jpe?g|png|gif|webp|bmp|svg)(\?.*)?$/i.test(u);
}
function joinUrl(base, path){
  try{ return new URL(path, base).href; }catch(e){ return base + path; }
}

async function fetchImagesFromUrl(url){
  const res = await fetch(url, { mode:'cors' });
  if(!res.ok) throw new Error('Abrufen fehlgeschlagen: ' + res.status);
  const text = await res.text();
  const found = new Set();
  const imgRegex = /<img[^>]+src=["']([^"']+)["'][^>]*>/gi;
  let m;
  while((m = imgRegex.exec(text)) !== null) if(looksLikeImage(m[1])) found.add(joinUrl(url, m[1]));
  const aRegex = /<a[^>]+href=["']([^"']+)["'][^>]*>/gi;
  while((m = aRegex.exec(text)) !== null) if(looksLikeImage(m[1])) found.add(joinUrl(url, m[1]));
  return Array.from(found);
}

function buildCarousel(images){
  carouselIndicators.innerHTML = '';
  carouselInner.innerHTML = '';
  images.forEach((src, idx) => {
    // Indicator
    const btn = document.createElement('button');
    btn.type = 'button';
```

```
    btn.setAttribute('data-bs-target', '#imageCarousel');
    btn.setAttribute('data-bs-slide-to', String(idx));
    btn.ariaLabel = 'Slide ' + (idx + 1);
    if(idx === 0) { btn.className = 'active'; btn.setAttribu-
te('aria-current','true'); }
    carouselIndicators.appendChild(btn);

    // Slide
    const slide = document.createElement('div');
    slide.className = 'carousel-item' + (idx === 0 ? ' active' :
'');
    const link = document.createElement('a');
    link.href = src;
    link.target = '_blank';
    link.rel = 'noopener';
    const img = document.createElement('img');
    img.src = src;
    img.alt = src.split('/').pop().split('?')[0];
    img.className = 'd-block w-100 carousel-img';
    link.appendChild(img);
    slide.appendChild(link);
    carouselInner.appendChild(slide);
  });

  carouselWrapper.style.display = images.length ? '' : 'none';
}

(async function init(){
  status.textContent = 'Lade Bilder von ' + baseUrl;
  if(!baseUrl || !baseUrl.trim()){ status.textContent = 'Keine ba-
seUrl gesetzt. Bitte im Script eintragen.'; return; }
```

```
  try{
    const imgs = await fetchImagesFromUrl(baseUrl);
    if(imgs.length){
      buildCarousel(imgs);
      status.textContent = `Bilder in der Galerie:
${imgs.length}`;
      return;
    } else {
      status.textContent = 'Keine Bilder auf der Seite gefunden.
Nutze fallbackImages.';
    }
  }catch(err){
    console.warn('Ladefehler:', err);
    status.textContent = 'Automatisches Laden fehlgeschlagen (mög-
licher CORS-/Netzwerkfehler). Nutze fallbackImages.';
  }


  if(fallbackImages.length === 0){
    status.textContent = 'Keine Bilder verfügbar. Entweder CORS
aktivieren oder fallbackImages befüllen.';
    return;
  }
  buildCarousel(fallbackImages.map(u => joinUrl(baseUrl || loca-
tion.href, u)));
  status.textContent = `Carousel geladen (Fallback): ${fallback-
Images.length} Bild(er).`;
})();
</script>
```

Im Ergebnis werden die Bilder dann so dargestellt:

Bilder in der Galerie: 8



Galerie mit Zoom und Overlay-Slider

Zur "Krönung" hier noch der Quelltext für eine Galerie mit Zoom, die beim Klicken auf ein Bild auf eine Slider-Ansicht also Overlay wechselt.

```
<style>

  :root{

    --gap:10px;

    --thumb-size:200px;

    --zoom-scale:1.8;

    --transition:300ms;

  }

  body, .gallery-container{ margin:0; font-family:system-ui,
-apple-system, "Segoe UI", Roboto, "Helvetica Neue", Arial; back-
ground:inherit; color:#222; }

  .gallery{ display:grid; grid-template-columns: repeat(auto-fill,
minmax(var(--thumb-size), 1fr)); gap:var(--gap); }

  .card{ position:relative; overflow:hidden; border-radius:8px;
height:var(--thumb-size); background:#f3f4f6; box-shadow:0 6px
18px rgba(0,0,0,0.06); cursor:pointer; }

  .card img{ width:100%; height:100%; object-fit:cover; transi-
tion: transform var(--transition) ease, filter var(--transition)
ease; transform-origin:center center; will-change:transform; dis-
play:block; user-select:none; pointer-events:none; }

  .card:hover img{ filter:brightness(1.03); }

  .card:hover img.zoomed{ transform:scale(var(--zoom-scale)); }

  .meta{ position:absolute; left:8px; bottom:8px; back-
ground:rgba(0,0,0,0.45); color:#fff; padding:6px 8px; border-ra-
dius:6px; font-size:12px; max-width:90%; white-space:nowrap; over-
flow:hidden; text-overflow:ellipsis; }



  /* Overlay / Viewer */

  .viewer{

    position:fixed; inset:0; display:none; align-items:center;
justify-content:center;

    background:rgba(0,0,0,0.85); z-index:1050;
```

```
  }

  .viewer.visible{ display:flex; }

  .viewer-content{ position:relative; max-width:95%; max-
height:95%; width:calc(80vw); height:calc(80vh); display:flex;
align-items:center; justify-content:center; }

  .viewer-img{ max-width:100%; max-height:100%; object-fit:con-
tain; border-radius:6px; box-shadow:0 10px 40px rgba(0,0,0,0.6);
background:#000; }


  /* Controls */

  .viewer-btn{

    position:absolute; top:50%; transform:translateY(-50%);

    color:#fff; background:rgba(0,0,0,0.35); border:0; width:56px;
height:56px; border-radius:50%;

    display:flex; align-items:center; justify-content:center;
font-size:28px; cursor:pointer;

    transition:background .15s;

  }

  .viewer-btn:hover{ background:rgba(0,0,0,0.6); }

  .viewer-prev{ left:12px; }

  .viewer-next{ right:12px; }


  .viewer-close{

    position:absolute; top:12px; right:12px; color:#fff; back-
ground:rgba(0,0,0,0.35); border:0; width:44px; height:44px;
border-radius:6px;

    display:flex; align-items:center; justify-content:center;
font-size:20px; cursor:pointer;

  }

  .viewer-caption{ position:absolute; bottom:12px; left:12px;
right:12px; color:#fff; text-align:center; font-size:14px; opaci-
ty:0.9; }
```

```
  @media (max-width:480px){ :root{ --thumb-size:120px; } .viewer-
btn{ width:44px;height:44px;font-size:22px; } .viewer-close{
width:36px;height:36px;font-size:18px; } .viewer-content{
width:92vw; height:60vh; } }

</style>




  <div class="container gallery-container py-3">

    <div id="status" class="mb-2 text-muted">Lade Bilder…</div>

    <div class="gallery" id="gallery" aria-live="polite"></div>

  </div>



  <!-- Viewer Overlay -->

  <div id="viewer" class="viewer" role="dialog" aria-modal="true"
aria-hidden="true">

    <div class="viewer-content">

      <button class="viewer-btn viewer-prev" id="prevBtn" aria-la-
bel="Vorheriges">‹</button>

      <img id="viewerImg" class="viewer-img" src="" alt="">

      <button class="viewer-btn viewer-next" id="nextBtn" aria-la-
bel="Nächstes">›</button>

      <button class="viewer-close" id="closeBtn" aria-la-
bel="Schließen">✕</button>

      <div class="viewer-caption" id="viewerCaption"></div>

    </div>

  </div>




<script>

/* === URL hier eintragen (oder leer lassen und fallbackImages
nutzen) === */

const baseUrl = 'https://<IHR_HUB>/cloud/<IHR_KANAL>/<BILDERVER-
ZEICHNIS>/'; // <- anpassen: https://<IHR_HUB>/cloud/<IHR_KA-
NAL>/<BILDERVERZEICHNIS>
```

```
const fallbackImages = [
  // 'https://<IHR_HUB>/cloud/<IHR_KANAL>/<BILDERVERZEICH-
NIS>/<BILDDATEI>>',
];

const gallery = document.getElementById('gallery');
const status = document.getElementById('status');
const viewer = document.getElementById('viewer');
const viewerImg = document.getElementById('viewerImg');
const viewerCaption = document.getElementById('viewerCaption');
const prevBtn = document.getElementById('prevBtn');
const nextBtn = document.getElementById('nextBtn');
const closeBtn = document.getElementById('closeBtn');

let images = [];
let currentIndex = -1;

function looksLikeImage(u){ return
/\.(jpe?g|png|gif|webp|bmp|svg)(\?.*)?$/i.test(u); }
function joinUrl(base, path){ try{ return new URL(path,
base).href; }catch(e){ return base + path; } }

function addImageCard(src, filename, idx){
  const a = document.createElement('a');
  a.href = '#';
  a.className = 'card';
  a.dataset.index = String(idx);
  a.title = filename || src;
  a.addEventListener('click', (ev) => { ev.preventDefault(); open-
Viewer(Number(a.dataset.index)); });
```

```
  const img = document.createElement('img');
  img.src = src; img.alt = filename || '';
  img.className = 'zoomed';

  const meta = document.createElement('div');
  meta.className = 'meta';
  meta.textContent = filename || '';

  a.appendChild(img);
  a.appendChild(meta);
  gallery.appendChild(a);

  a.addEventListener('mousemove', (e) => {
    const rect = img.getBoundingClientRect();
    const x = ((e.clientX - rect.left) / rect.width) * 100;
    const y = ((e.clientY - rect.top) / rect.height) * 100;
    img.style.transformOrigin = `${x}% ${y}%`;
  });
  a.addEventListener('mouseleave', () => { img.style.transformOri-
gin = 'center center'; });
}

function clearGallery(){ gallery.innerHTML = ''; }

async function fetchImagesFromUrl(url){
  const res = await fetch(url, { mode:'cors' });
  if(!res.ok) throw new Error('Abrufen fehlgeschlagen: ' +
res.status);
  const text = await res.text();
  const found = new Set();
```

```
  const imgRegex = /<img[^>]+src=["']([^"']+)["'][^>]*>/gi;

  let m;

  while((m = imgRegex.exec(text)) !== null) if(looksLikeIma-
ge(m[1])) found.add(joinUrl(url, m[1]));

  const aRegex = /<a[^>]+href=["']([^"']+)["'][^>]*>/gi;

  while((m = aRegex.exec(text)) !== null) if(looksLikeImage(m[1]))
found.add(joinUrl(url, m[1]));

  return Array.from(found);

}



/* Viewer controls */

function showViewer(){ viewer.classList.add('visible'); viewer.se-
tAttribute('aria-hidden','false'); document.body.style.over-
flow='hidden'; }

function hideViewer(){ viewer.classList.remove('visible');
viewer.setAttribute('aria-hidden','true'); document.body.sty-
le.overflow=''; currentIndex = -1; }

function updateViewer(){

  if(currentIndex < 0 || currentIndex >= images.length) return;

  viewerImg.src = images[currentIndex];

  viewerImg.alt = images[currentIn-
dex].split('/').pop().split('?')[0];

  viewerCaption.textContent = viewerImg.alt;

}



function openViewer(idx){

  currentIndex = idx;

  updateViewer();

  showViewer();

}
```

```
function showPrev(){ if(images.length===0) return; currentIndex =
(currentIndex -1 + images.length) % images.length; updateViewer();
}

function showNext(){ if(images.length===0) return; currentIndex =
(currentIndex +1) % images.length; updateViewer(); }


prevBtn.addEventListener('click', (e)=>{ e.stopPropagation();
showPrev(); });

nextBtn.addEventListener('click', (e)=>{ e.stopPropagation();
showNext(); });

closeBtn.addEventListener('click', (e)=>{ e.stopPropagation(); hi-
deViewer(); });


viewer.addEventListener('click', (e)=>{

  if(e.target === viewer) hideViewer();

});


document.addEventListener('keydown', (e)=>{

  if(viewer.classList.contains('visible')){

    if(e.key === 'Escape') hideViewer();

    if(e.key === 'ArrowLeft') showPrev();

    if(e.key === 'ArrowRight') showNext();

  }

});


(async function init(){

  clearGallery();

  status.textContent = 'Lade Bilder von ' + baseUrl;

  try{

    if(baseUrl && baseUrl.trim()){

      const found = await fetchImagesFromUrl(baseUrl);

      if(found.length){
```

```
        images = found;

        found.forEach((u,i)=> addImageCard(u,
u.split('/').pop().split('?')[0], i));

        status.textContent = 'Bilder in der Galerie: ' +
found.length;

        return;

      } else {

        status.textContent = 'Keine Bilder gefunden. Nutze Fall-
back-Liste.';

      }

    } else {

      status.textContent = 'Keine baseUrl gesetzt. Nutze Fallback-
Liste.';

    }

  }catch(err){

    console.warn('Fehler beim Laden:', err);

    status.textContent = 'Automatisches Laden fehlgeschlagen (mög-
liche CORS-Einschränkung). Nutze Fallback-Liste.';

  }



  if(fallbackImages.length === 0){

    status.textContent = 'Keine Bilder verfügbar. Bitte baseUrl
setzen oder fallbackImages befüllen.';

    return;

  }

  images = fallbackImages.map(u => joinUrl(baseUrl || loca-
tion.href, u));

  images.forEach((u,i)=> addImageCard(u,
u.split('/').pop().split('?')[0], i));

  status.textContent = 'Galerie geladen (Fallback): ' +
images.length;

})();

</script>
```


Klackerhub
m%C3%A4uschen.jpg

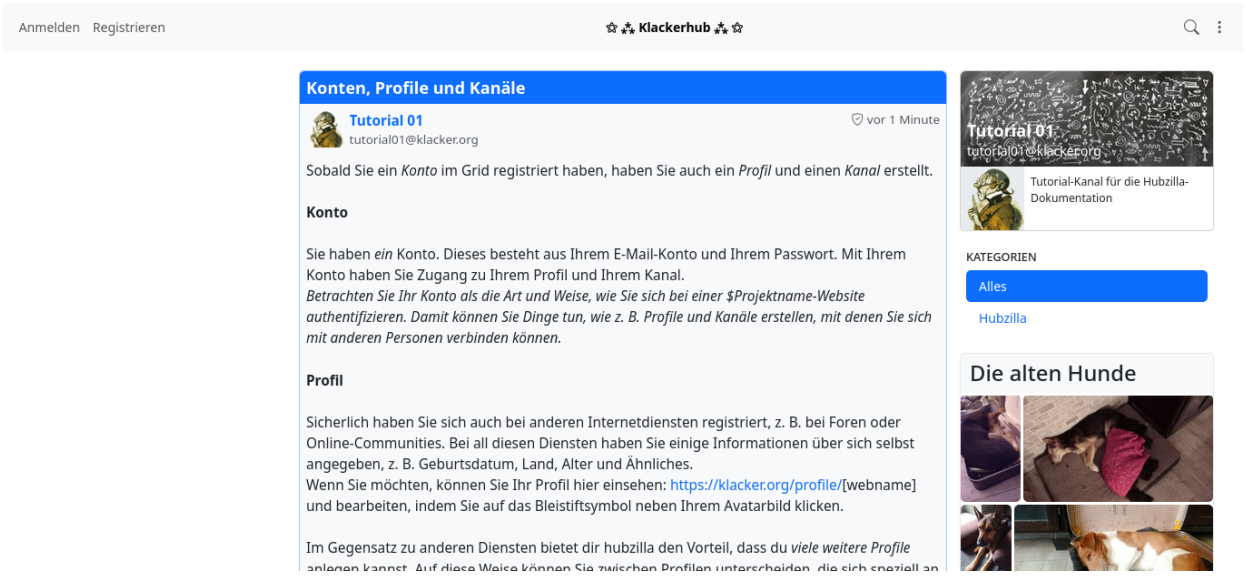
Navigationsleiste ersetzen

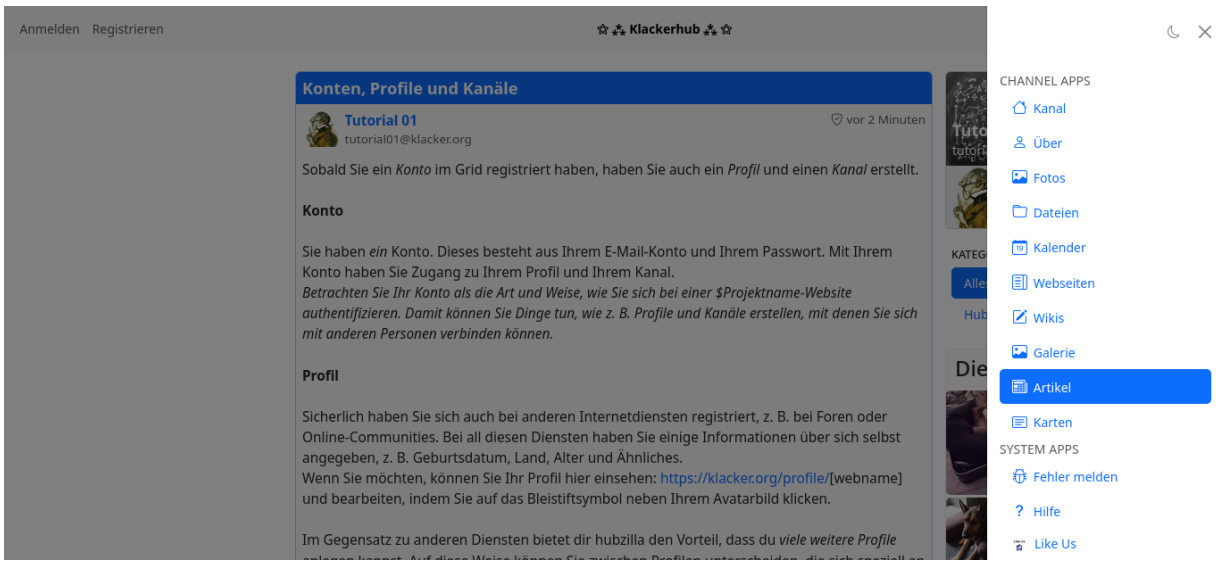
Wenn Sie den Besuchern/Nutzern Ihrer Web-Präsenz Zugriff auf Hubzilla-Apps, wie z.B. die Artikel App einräumen, dann befindet sich am oberen Bildschirmrand die Navigationsleiste, wie si von Hubzilla erzeugt wird.

Das bedeutet, dass bei Redbasic links oben zwei Links zum Einloggen und Registrieren vorhanden sind. Auf der rechten Seite werden Apps, die für Besucher nutzbar sind, als Icons angezeigt, sofern Sie diese an die Navigationsleiste angepinnt haben, sowie das Symbol " ", über welches Besucher das App-Menü mit den System-Apps nutzen können.

Einerseits signalisiert diese Optik dem Besucher, dass es sich um eine Präsenz handelt, die mit Hubzilla betrieben wird (sofern er dies erkennt), andererseits können Sie nicht wirklich beeinflussen, auf welche Seiten er besser keinen schnellen Zugriff haben sollte.

Betreiben Sie mit Hubzilla also beispielsweise eine Seite, die auch über einen Blog-Bereich verfügt, so bietet es sich an, für das Blog die App Artikel zu nutzen. Haben Sie auf Ihrer Landingpage und den anderen Webseiten Ihrer Präsenz nun im Bereich "nav" eine Bootstrap-Navbar integriert, können Sie einen Link zur Artikel-App dort einfügen. Leider ist es dann so, dass Ihre spezielle Navigationsleiste nach dem Öffnen der Aktikel-App nicht mehr angezeigt wird, sondern die Navigationsleiste von Hubzilla selbst (wie oben beschrieben).





Das ist einerseits ein Bruch in der Benutzerführung und andererseits auch damit verbunden, dass der Besucher damit auf Bereiche stößt, welche Sie für Ihn im direkten Zugriff nicht vorgesehen haben.

Deshalb sollten Sie in einem solchen Anwendungsfall die Standard-Navigationsleiste in der App "Artikel" durch ihre eigene ersetzen.

Das ist auch kein großer Aufwand. ich erkläre Ihnen jetzt, wie das ganz einfach umzusetzen ist und Sie dabei eine kleine Unzulänglichkeit in der Darstellung auch gleich umschiffen können.

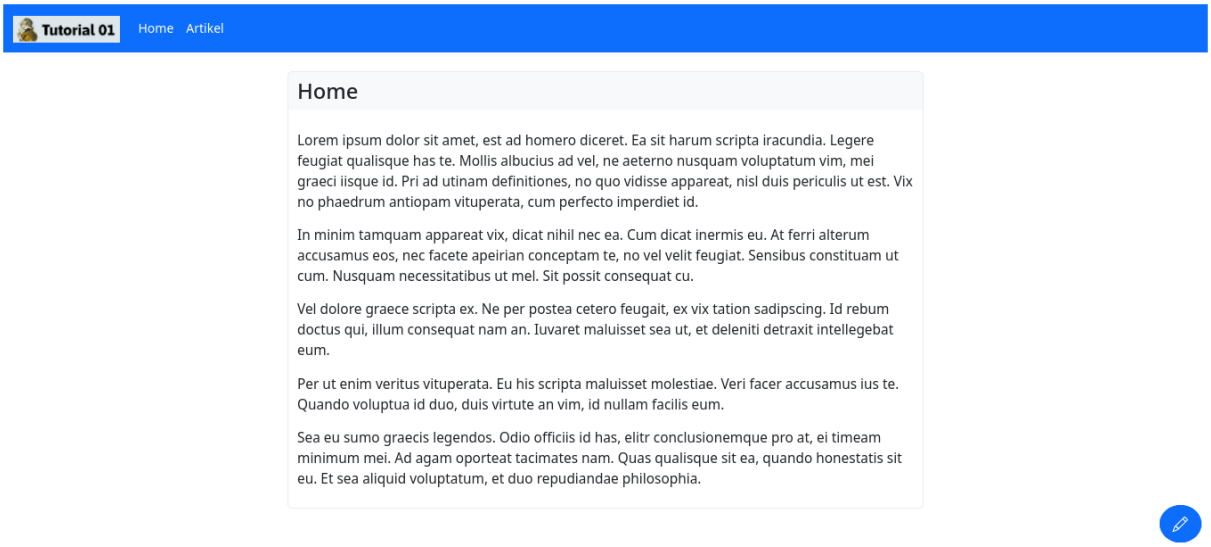
Zuerst erstellen wir uns eine Navigationsleiste z.B. mit dem Namen "main_menu":

```
<nav class="navbar navbar-expand-lg bg-primary">

  <div class="container-fluid">

    <a class="navbar-brand" href="https://klacker.org/page/tutori-
al01/home"><img src="https://klacker.org/cloud/tutori-
al01/misc/tutlogo.png" alt="Logo" height="30" class="d-inline-
block align-text-top"></a>



    <button class="navbar-toggler" type="button" data-bs-tog-
gle="collapse" data-bs-target="#navbarSupportedContent" aria-con-
trols="navbarSupportedContent" aria-expanded="false" aria-la-
bel="Toggle navigation">

      <span class="navbar-toggler-icon"></span>

    </button>

    <div class="collapse navbar-collapse" id="navbarSupportedCon-
tent">

      <ul class="navbar-nav me-auto mb-2 mb-lg-0">

      <li class="nav-item"><a class="nav-link text-white"
href="https://klacker.org/page/tutorial01/home">Home</a></li>

      <li class="nav-item"><a class="nav-link text-white"
href="https://klacker.org/articles/tutorial01">Artikel</a></li>

      </ul>

    </div>

  </div>

</nav>
```

Diese Navigationsleiste fügen wir auch in das Layout "basis-layout1" für unsere Homepage in den nav-Bereich ein.

```
[region=nav]
[block]main_menu[/block]

[/region]
[region=aside]

[/region]
[region=content]
$content

[/region]
[region=right_aside]

[/region]
```

Nun erscheint die Navigationsleiste, wenn wir unsere Beispiel-Homepage aufrufen.



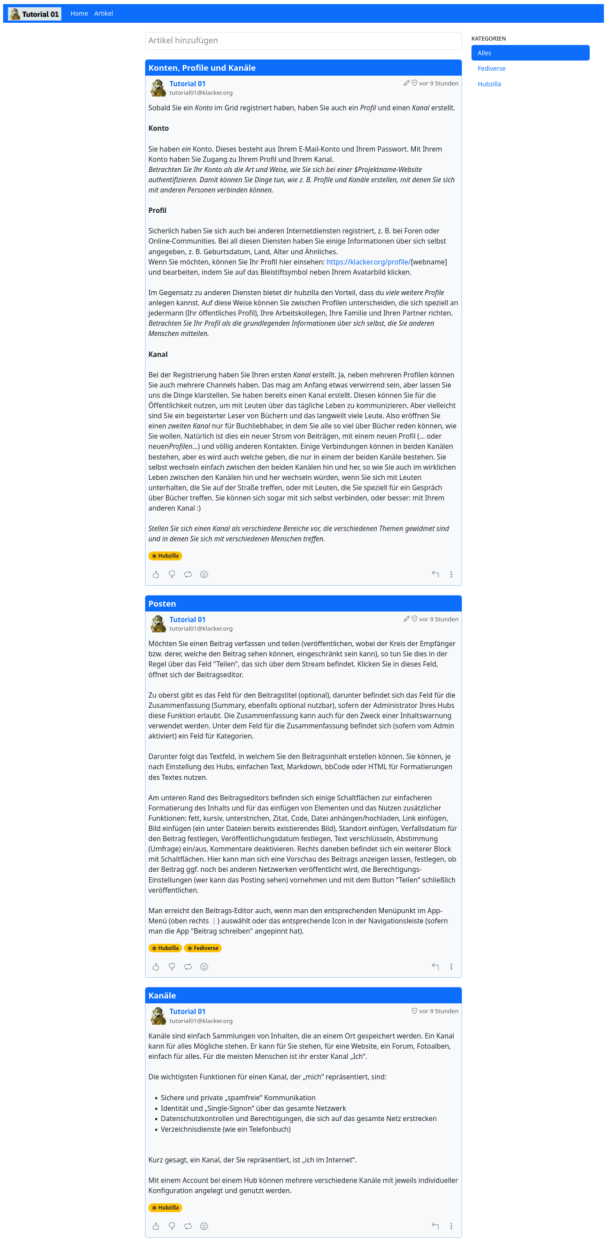
Damit diese Navigationsleiste nun auch in der App "Artikel" erscheint, müssen wir das Modul für diese App im PDL-Editor abändern. Nebenbei räumen wir auch (vorerst) die Seitenleisten auf.

Wir rufen den PDL-Editor auf und wählen im Menüpunkt "MODULES" das Modul "articles". Weil ich es schön "aufgeräumt" haben möchte, sollen auch die Widgets "Album" (das wir kürzlich hinzugefügt haben, um es auszuprobieren), "Profile" (welches die Profilkarte anzeigt) und "Notifications" (zeigt Benachrichtigungen der Kanalaktivitäten an) entfernt werden. Dafür genügt ein Klick auf den jeweiligen Remove-Button. Das Widget "Articles Categories" lassen wir drin, denn das ist eine sinnvolle Angelegenheit zum Auffinden und Navigieren durch die Artikel.

Nun fällt auf, dass es in der GUI des PDL-Editors gar keinen Bereich für die Navigationsleiste gibt. Wir können den Standard aber trotzdem überschreiben. Dafür wählen wir im Menü den Punkt "SOURCE", mit welchem sich der Quelltext-Editor für die Layout-Definition öffnet. Hier fügen wir ganz oben folgendes ein:

```
[region=nav]

[block]main_menu[/block]



[/region]
```

Ein Klick auf "APPLY" und die App "Artikel" sieht nun so aus:



Unzulänglichkeit Nr. 1 umschiffen

Wenn Sie nun als eingeloggter Kanalbesitzer auf die Webseite oder die App "Artikel" wechseln, sind Sie quasi auch in der Welt der eigenen Navigationsleiste "gefangen". Es fehlt dort der Zugriff auf die angepinnten Apps und das App-Menü, so dass Sie nicht einfach schnell beispielsweise zum PDL-Editor wechseln können, um da noch mehr anzupassen. Das, was der Sinn der Sache für nicht eingeloggte Besucher ist, fällt uns hier also selbst auf die Füße.

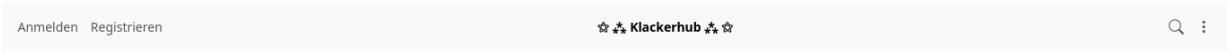
Aber das ist kein Problem. Wir nutzen einfach den Block unserer Menüleiste und hinterlegen das Logo mit dem Link zum "HQ" (es geht auch jede andere Ansicht, die nur eingeloggten Nutzern zur Verfügung steht, HQ bietet sich aber an):

```
<nav class="navbar navbar-expand-lg bg-primary">

  <div class="container-fluid">

    <a class="navbar-brand" href="https://klacker.org/hq"><img
src="https://klacker.org/cloud/tutorial01/misc/tutlogo.png"
alt="Logo" height="30" class="d-inline-block align-text-top"></a>



    <button class="navbar-toggler" type="button" data-bs-tog-
gle="collapse" data-bs-target="#navbarSupportedContent" aria-con-
trols="navbarSupportedContent" aria-expanded="false" aria-la-
bel="Toggle navigation">

      <span class="navbar-toggler-icon"></span>

    </button>

    <div class="collapse navbar-collapse" id="navbarSupportedCon-
tent">

      <ul class="navbar-nav me-auto mb-2 mb-lg-0">

      <li class="nav-item"><a class="nav-link text-white"
href="https://klacker.org/page/tutorial01/home">Home</a></li>

      <li class="nav-item"><a class="nav-link text-white"
href="https://klacker.org/articles/tutorial01">Artikel</a></li>

      </ul>

    </div>

  </div>

</nav>
```

Hier ist in der vierten Zeile nun als Ziel-URL für das Logo-Bild der Link zum HQ hinterlegt.

Wenn wir nun selbst, als eingeloggter Kanalbesitzer auf die Logo-Grafik klicken, landen wir beim HQ mit der üblichen Hubzilla-Navigationsleiste.

Besucher hingegen landen auf einer leeren Seite, denn sie verfügen ja über kein "HQ".

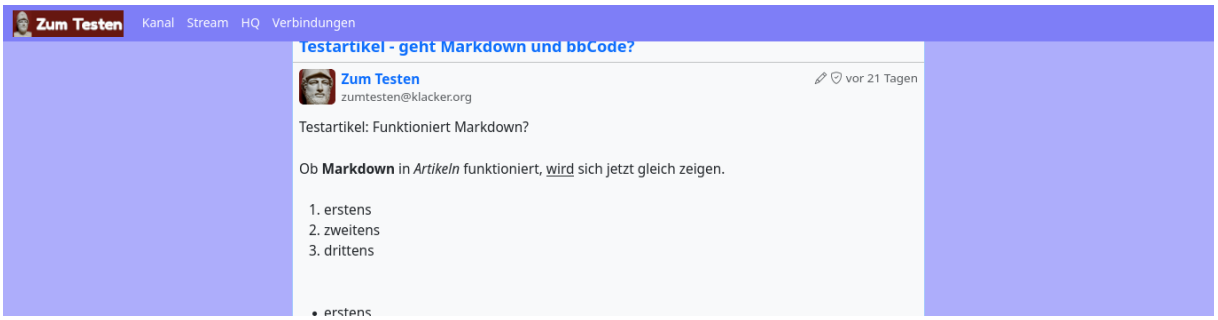


Unzulänglichkeit Nr. 2 umschiffen

Je nachdem wie sie Ihr Theme konfiguriert haben, kann es sein, dass der Inhaltsbereich der Artikel-App "zu weit oben" ansetzt. Damit wird er dann oben teilweise abgeschnitten, was optisch nicht wirklich professionell aussieht.

Eingeloggt:

Nicht eingeloggter Besucher:



Dieses Problem lässt sich beseitigen, indem wir in der nav-Region direkt unter den Block mit der Navigationsleiste einen zusätzlichen Abstandhalter-Block mit der Hintergrundfarbe einfügen.

```
Block "nav-spacer":

<style>

.spacer {

  background-color: var(--hz-body-bg);

}

</style>

<div class="spacer">


</div>
```

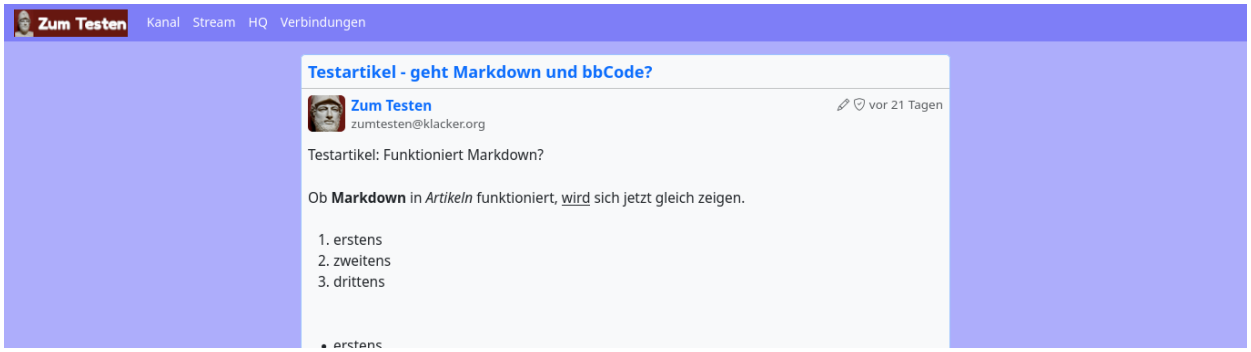
nav-Region in der PDL-Datei des Moduls "articles":

```
[region=nav]
[block]main_menu[/block]
[block]nav-spacer[/block]

[/region]
```

Nun wird es korrekt dargestellt.

Eingeloggt:

Nicht eingeloggter Besucher:



HTML-Editor - simple Methode

Wenn Sie Hubzilla einfach nur als CMS benutzen möchten, Sie es aber bequemer haben möchten und HTML-Inhalte einfach "runterschreiben" wollen, sind die Bordmittel von Hubzilla im Webseiten- und Blockeditor nicht optimal.

Wählen Sie HTML als Inhaltstyp, verschwinden sogar die Icons unten am Editor, die ja speziell für BBcode gedacht sind. Den gesamten HTML-Content müssen Sie "zu Fuß" codieren.

Solange Sie nur ein wenig Textformatierung nutzen möchten, also z.B. fette oder kursive Schrift und ab und an einmal einen Link einfügen, dann ist es zwar unbequem, aber trotzdem noch machbar. Sie müssen ja nur ein paar Tags hinzufügen.

Allerdings könnten Sie dann auch den Inhaltstyp BBcode oder Markdown verwenden, wobei bei BBcode immerhin noch die Helfer aus der unteren Toolbar zur Verfügung stehen und der Editor selbst nach dem Tippen einer öffnenden Tag-Klammer ("[") Ergänzungsvorschläge direkt im Editor angezeigt werden.

Für den Inhaltstyp HTML haben Sie solche Hilfen nicht. Es lässt sich aber, anstatt nun einen externen HTML-Editor zu nutzen (die auch nicht immer optimal sind), einfach ein Editor mit auf die App-Seite platzieren, wo Sie Ihre Inhalte schreiben und anschließend den Quelltext in das eigentliche Editorfeld kopieren.

Dafür bietet sich z.B. TinyMCE in der GPL-Version an. Die Distribution kann man auf der TinyMCE-Seite herunterladen: Get TinyMCE's WYSIWYG editor (die Free-Version herunterladen).

Die Zip-Datei entpacken Sie auf Ihrem Rechner. Es wird ein Verzeichnis "tinymce" erzeugt, welcher die Datei CHANGELOG.md und ein Unterverzeichnis "js" enthält. In diesem finden Sie wiederum ein Unterverzeichnis, welches wiederum "tinymce" heißt. Und genau dieses Verzeichnis kopieren Sie in das Wurzelverzeichnis der Cloud Ihres Hubzilla-Kanals. (Sie können auch das entpackte Archiv einfach so dort hin kopieren, das macht aber einen fehleranfälligen längeren Pfad im Editor-Block erforderlich, den Sie sich auf diese Weise ersparen können.)

Jetzt müssen Sie nur noch einen Editor-Block "tmce-block" erzeugen:

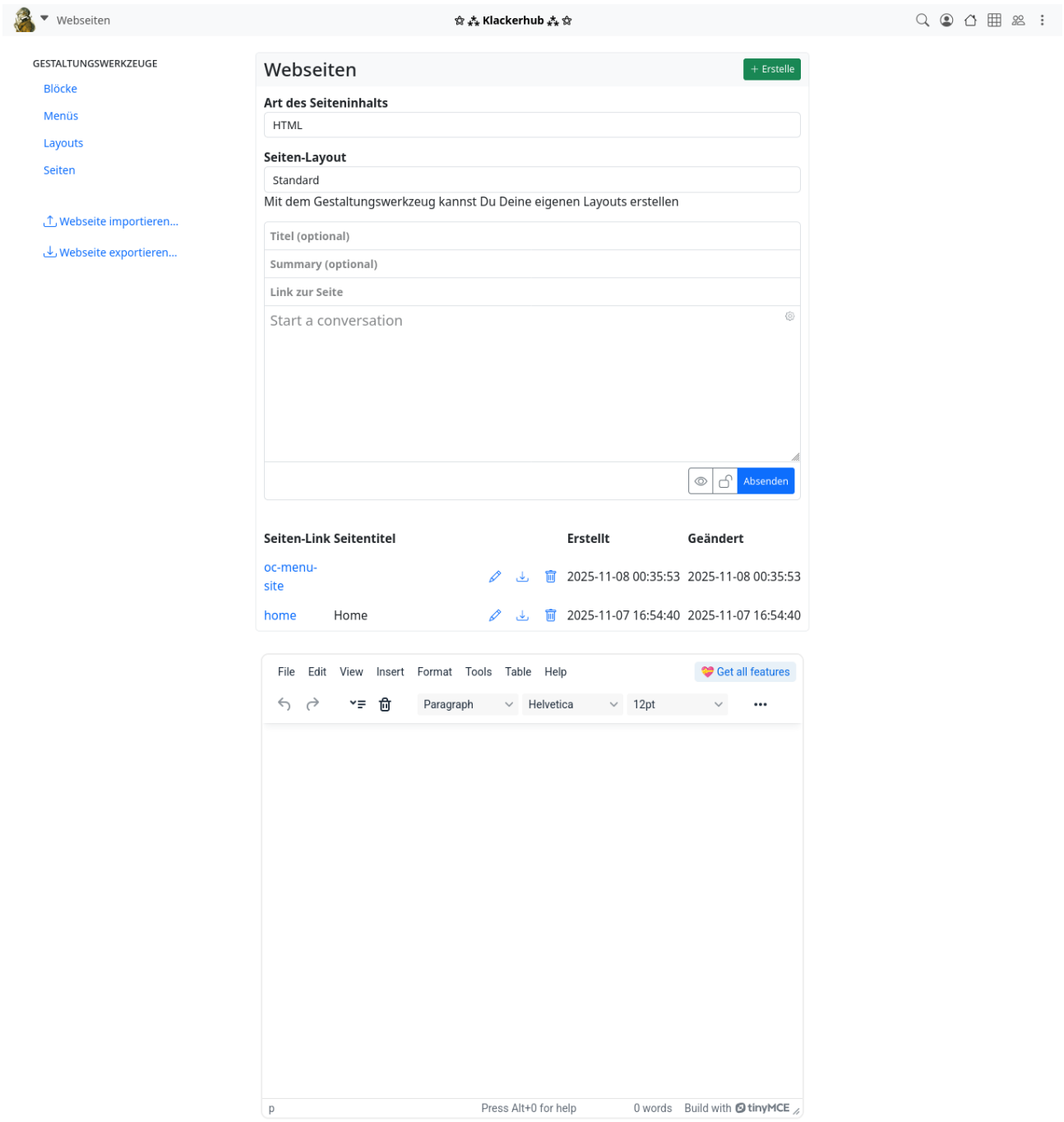
```
<!--  In der folgenden Zeile die URL der Cloud eintragen!  -->

<script src="<URL_DES_HUB>/cloud/<KANALNAME>>/tinymce/tinym-
ce.min.js" referrerpolicy="origin" crossorigin="anony-
mous"></script>

<script>
```

```
  const useDarkMode = window.matchMedia('(prefers-color-scheme:
dark)').matches;
  const isSmallScreen = window.matchMedia('(max-width:
1023.5px)').matches;

  tinymce.init({
  selector: '#mytextarea',
  license_key: 'gpl',
  plugins: 'preview importcss searchreplace autolink autosave save
directionality code visualblocks visualchars fullscreen image link
media codesample table charmap pagebreak nonbreaking anchor in-
sertdatetime advlist lists wordcount help charmap quickbars emoti-
cons accordion',
  editimage_cors_hosts: ['picsum.photos'],
  menubar: 'file edit view insert format tools table help',
  toolbar: "undo redo | accordion accordionremove | blocks fontfa-
mily fontsize | bold italic underline strikethrough | align num-
list bullist | link image | table media | lineheight outdent in-
dent| forecolor backcolor removeformat | charmap emoticons | code
fullscreen preview | save print | pagebreak anchor codesample |
ltr rtl",
  autosave_ask_before_unload: true,
  autosave_interval: '30s',
  autosave_prefix: '{path}{query}-{id}-',
  autosave_restore_when_empty: false,
  autosave_retention: '2m',
  image_advtab: true,
  link_list: [
    { title: 'My page 1', value: 'https://www.tiny.cloud' },
    { title: 'My page 2', value: 'http://www.moxiecode.com' }
  ],
  image_list: [
    { title: 'My page 1', value: 'https://www.tiny.cloud' },
    { title: 'My page 2', value: 'http://www.moxiecode.com' }
  ],
  image_class_list: [
    { title: 'None', value: '' },
    { title: 'Some class', value: 'class-name' }
  ],
  importcss_append: true,
  file_picker_callback: (callback, value, meta) => {
    /* Provide file and text for the link dialog */
    if (meta.filetype === 'file') {
      callback('hhttps://hubzilla.org/cloud/info/web-
site_images_small/hubzilla-logo-roland-1.png', { text: 'My text'
});
    }

    /* Provide image and alt text for the image dialog */
```

```
    if (meta.filetype === 'image') {
      callback('https://hubzilla.org/cloud/info/web-
site_images_small/hubzilla-logo-roland-1.png', { alt: 'My alt
text' });
    }

    /* Provide alternative source and posted for the media dialog
*/
    if (meta.filetype === 'media') {
      callback('movie.mp4', { source2: 'alt.ogg', poster:
'https://hubzilla.org/cloud/info/web-
site_images_small/S%20Hubzilla%20Connect.png' });
    }
  },
  height: 600,
  image_caption: true,
  quickbars_selection_toolbar: 'bold italic | quicklink h2 h3
blockquote quickimage quicktable',
  noneditable_class: 'mceNonEditable',
  toolbar_mode: 'sliding',
  contextmenu: 'link image table',
  skin: useDarkMode ? 'oxide-dark' : 'oxide',
  content_css: useDarkMode ? 'dark' : 'default',
  content_style: 'body { font-family:Helvetica,Arial,sans-serif;
font-size:16px }'
});
</script>

<style>
/* For other boilerplate styles, see:
https://www.tiny.cloud/docs/tinymce/8/editor-content-css/ */
/*
* For rendering images inserted using the image plugin.
* Includes image captions using the HTML5 figure element.
*/

figure.image {
  display: inline-block;
  border: 1px solid gray;
  margin: 0 2px 0 1px;
  background: #f5f2f0;
}

figure.align-left {
  float: left;
}

figure.align-right {
  float: right;
```

```
}

figure.image img {
  margin: 8px 8px 0 8px;
}

figure.image figcaption {
  margin: 6px 8px 6px 8px;
  text-align: center;
}

/*
 Alignment using classes rather than inline styles
 check out the "formats" option
*/

img.align-left {
  float: left;
}

img.align-right {
  float: right;
}

/* Basic styles for Table of Contents plugin (tableofcontents) */
.mce-toc {
  border: 1px solid gray;
}

.mce-toc h2 {
  margin: 4px;
}

.mce-toc li {
  list-style-type: none;
}
</style>

<form method="post">
  <textarea id="mytextarea" style="visibility: hidden;"></text-
area>
</form>
```

Ist der Block gespeichert, öffnen Sie den PDL-Editor, wählen das Modul "editwebpage" und fügen Sie den Block unterhalb des Inhaltsbereichs ein. Dann mit "APPLY" fixieren. Ebenso fügen Sie diesen Block im Modul "editblock" ein, und künftig erscheint bei Nutzung des Webpage- und Block-Editors unterhalb des hubzilla-eigenen Editors zusätzlich der TinyMCE.



Sie können nun Im TinyMCE Ihren Inhalt im WYSIWYG-Modus gestalten. Dann schalten Sie auf die Code-Ansicht kopieren den nun angezeigten HTML-Code und fügen diesen in den Hubzilla Editorblock ein.

Kontaktformular

Webseiten bieten oft ein Kontaktformular an. Die einfachste Methode ist es, ein Webformular für die Nachricht zu erstellen und diese dann über den E-Mail-Klienten des Besuchers abschicken zu lassen.

Der Block dafür sieht so aus:

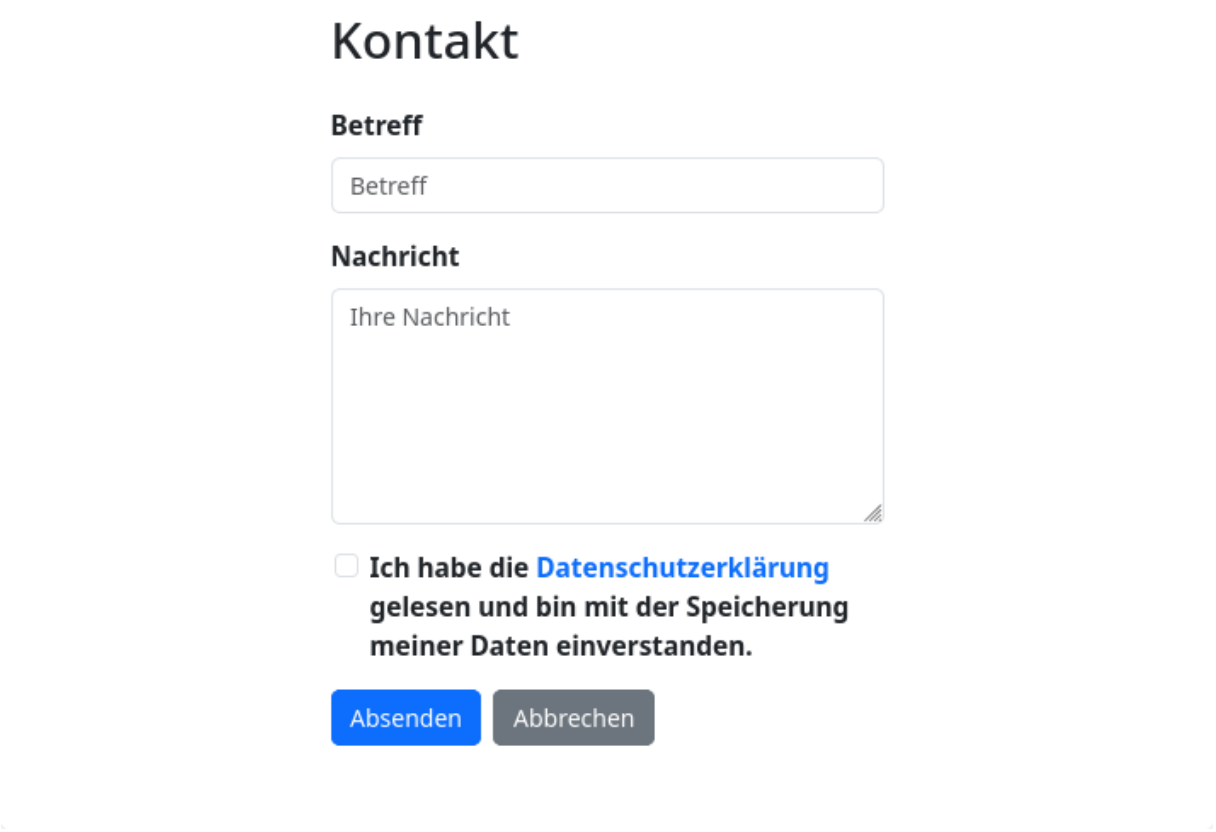
```
<div class="container py-5">
  <div class="row justify-content-center">
    <div class="col-md-8 col-lg-6">
      <h1 class="mb-4">Kontakt</h1>

      <form id="contactForm" novalidate>
        <div class="mb-3">
          <label for="subject" class="form-label">Betreff</label>
          <input type="text" class="form-control" id="subject" name="subject" required placeholder="Betreff">
          <div class="invalid-feedback">Bitte geben Sie einen Betreff ein.</div>
        </div>

        <div class="mb-3">
          <label for="message" class="form-label">Nachricht</label>
          <textarea class="form-control" id="message" name="message" rows="6" required placeholder="Ihre Nachricht"></textarea>
          <div class="invalid-feedback">Bitte geben Sie eine Nachricht ein.</div>
        </div>

        <div class="form-check mb-3">
```

```
          <input class="form-check-input" type="checkbox" value="1" id="privacy_agree"
required>


          <!-- Hier die URL zur Datenschutzerklärung einfügen -->


          <label class="form-check-label" for="privacy_agree">

            Ich habe die <a href="<URL_DER_DATENSCHUTZERKLÄRUNG>" tar-
get="_blank" rel="noopener">Datenschutzerklärung</a> gelesen und bin mit der
Speicherung meiner Daten einverstanden.

          </label>

          <div class="invalid-feedback">Sie müssen zustimmen, um die Nachricht zu sen-
den.</div>

        </div>


        <div class="d-flex gap-2">

          <button type="submit" class="btn btn-primary" id="btnSend">Absenden</but-
ton>

          <button type="button" class="btn btn-secondary" id="btnReset">Abbre-
chen</button>

        </div>

       </form>


     </div>

   </div>

 </div>


 <script>

 (function () {
```

```
  // Konfiguration (hier die Empfängeradresse anpassen)

  const recipient = '<ZIEL_MAILADRESSE_DES_KANALBETREIBERS>'; // <-- Empfänger-
adresse hier eintragen

  const form = document.getElementById('contactForm');
  const subjectEl = document.getElementById('subject');
  const messageEl = document.getElementById('message');
  const privacyEl = document.getElementById('privacy_agree');
  const btnReset = document.getElementById('btnReset');

  // Helper: mailto-encode
  function encodeMailto(text) {
    return encodeURIComponent(text).replace(/%0A/g, '%0D%0A');
  }

  form.addEventListener('submit', function (e) {
    e.preventDefault();
    // HTML5 validity
    if (!form.checkValidity()) {
      form.classList.add('was-validated');
      return;
    }
    if (!privacyEl.checked) {
      privacyEl.classList.add('is-invalid');
      return;
    } else {
      privacyEl.classList.remove('is-invalid');
```

```
    }

    const subject = subjectEl.value.trim();
    const message = messageEl.value.trim();

    // Build mailto link
    let mailto = 'mailto:' + encodeURIComponent(recipient)
            + '?subject=' + encodeMailto(subject)
            + '&body=' + encodeMailto(message);

    // Open default mail client
    window.location.href = mailto;

    // Optional: reset form after triggering mailto
    form.reset();
    form.classList.remove('was-validated');
  });

  btnReset.addEventListener('click', function () {
    form.reset();
    form.classList.remove('was-validated');
  });
})();
</script>
```

Hier muss dann nur noch die URL zur Datenschutzerklärung und die eigene Mail-adresse eingetragen werden.



Ein Mail-Versand mittels PHP wäre ebenfalls denkbar, ist jedoch nicht trivial, zumal es darauf ankommt, welches Mailversand-System der Hub verwendet.

Ein Blog mit Hubzilla statt mit WordPress

Sehr , sehr viele Blogs werden mit WordPress verwirklicht. Das auch aus gutem Grund, weil es sich bei WordPress um ein ausgereiftes, flexibles und umfangreiches CMS handelt, welches sich gerade für Blogs hervorragend eignet.

Im Folgenden werden wir hier ein Blog mit Hubzilla erstellen, welches sehr gut mit einem entsprechenden WordPress Blog mithalten kann. Um hier nicht ein völlig neues Blog zu "erfinden", für das auch noch Artikel kreiert werden müssen, habe ich mein Blog "Die Dampfdruck-Presse" zur Verfügung gestellt. Es existiert nun schon seit dem 13. Juli 2013 und umfasst mittlerweile 1.071 Artikel, sowie mehrere Serien zu DIY-Themen und etliche Informationsseiten.

Das alles komplett in Hubzilla zu übernehmen, wäre eine extreme Aufgabe, aber die Serien, die Info-Seiten, den Download-Bereich und ein paar Dutzend Artikel zu übernehmen, ist machbar und zeigt dann, inwieweit das Hubzilla-Blog dem WordPress-Blog ebenbürtig sein kann.

Anhand der Umsetzung werden Sie auch in der Praxis erfahren, was bisher in diesem Buch erwähnt wurde und wie man bestimmte Features nutzen kann.

Ein Blog: Blog-Kanal

Für das Blog muss natürlich zunächst ein eigener Kanal angelegt werden (was in Hubzilla ja problemlos möglich ist).

Weil das Forum öffentlich sein soll und Postings über neue Artikel unbeschränkt im Fediverse verteilt werden sollen, kann man ganz einfach die Kanalrolle "Öffentlich" dafür wählen. Ich habe für mein Blog trotzdem die Kanalrolle "Benutzerdefiniert" gewählt, weil ich einige Kleinigkeiten anders handhaben möchte, das spielt aber für die Erläuterungen hier keine Rolle. Gehen wir also von der Kanalrolle "Öffentlich" aus.

Für den Kanal braucht es ansonsten einen anständigen Kanalnamen und einen nicht zu komplizierten Kurznamen (Handle-Namen). Außerdem sollte das Profil mit den wichtigen Informationen bestückt werden und ein Profilbild, sowie ein Titelbild sind quasi Pflicht, um dem Blog auch ein "Gesicht" zu geben.

Für die Dampfdruck-Presse habe ich die Grafiken verwendet, die auch im WordPress-Blog schon immer genutzt wurden.

Das Profilbild:

Und das Titelbild:

Und die Profilinformationen für die Dampfdruck-Presse habe ich auch größtenteils übernommen.

- Name für diesen Kanal: "*Die Dampfdruck-Presse*"
- Kurze Beschreibung des Kanals: "*Die Dampfdruck-Presse (DDP) ist ein Blog rund um das Thema E-Dampfen.*"
- Geburtstag: "*13.07.1023*" (hier habe ich das Datum der Erzeugung des Originals eingetragen)
- Erzähle uns ein wenig von Dir: "*Informationen rund um das Thema E-Dampfen, DIYS und mehr...*
- *Zur Philosophie der DDP gehört, KEIN Blatt vor den Mund zu nehmen... selbst dann nicht, wenn ich damit einer vermeintlichen „Größe" der Dampferszene auf die Füße trete. Und ganz wichtig ist mir meine absolute Unabhängigkeit, weshalb ich mich nicht mit Wirtschaftsteilnehmern einlasse.*"

Wichtig: Für den Kanal muss das Ausführen von Code erlaubt sein! Beim eigenen Hub ist das kein Problem. Man schaltet die Codeausführung in der Admin-Oberfläche einfach mit einem einfachen Klick frei.Nutzt man einen öffentlichen Hub, so sollte das aber auch nicht die Hürde sein. Man schreibt den Admin an und bittet ihr, die Codeausführung zu erlauben. Dann muss dieser den Klick erledigen. Wenn man erläutert, weshalb man das benötigt (ein Blog betreiben und Webseiten auch mit Bootstrap gestalten), werden die meisten Admins dem zustimmen.

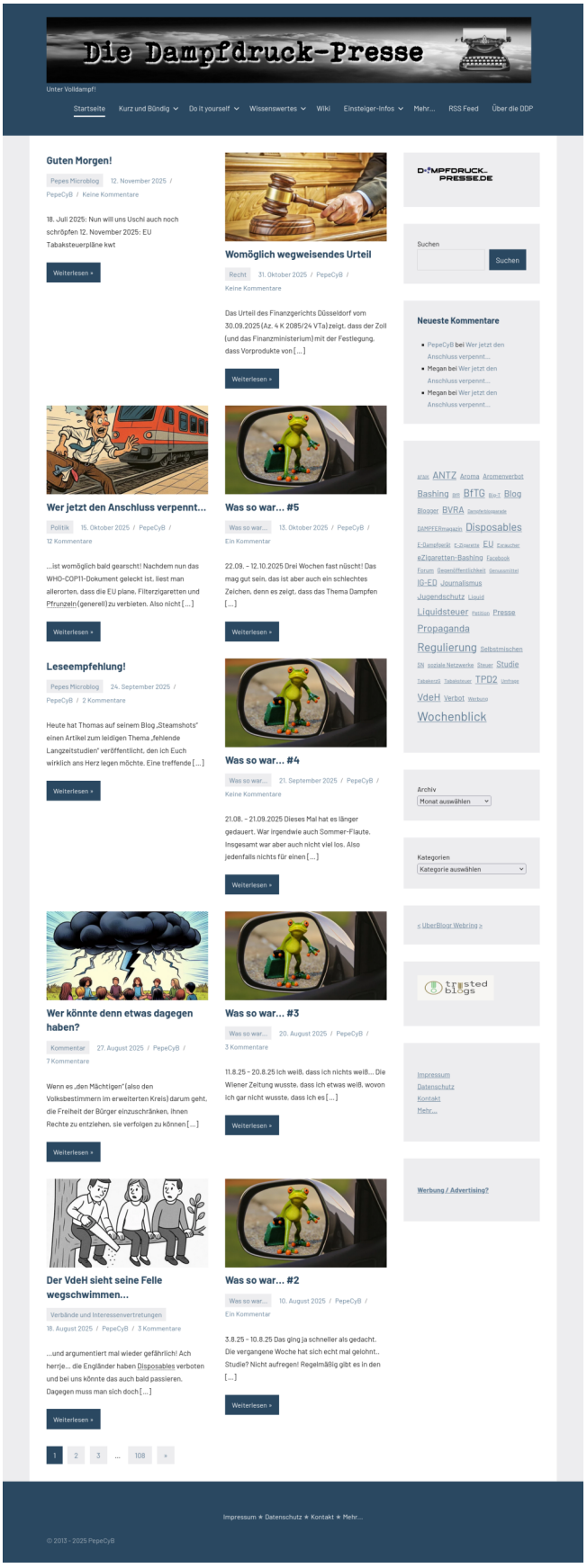
Damit wäre der Kanal bereit, zu einem Blog zu werden. Also im Beispielfall hier "Die Dampfdruck-Presse".

Ein Blog: Das Design

Weil die wenigsten Hubs bisher zusätzliche Themes installiert haben, bleibe ich hier beim Standard-Theme "redbasic". Hier kann man nachschauen: Einstellungen Anzeige-Einstellungen Design-Einstellungen. Für die Dampfdruck-Presse muss hier "redbasic" mit dem Style "Focus" eingestellt sein.

Die Dampfdruck-Presse hat im Laufe der Jahre immer wieder einmal Änderungen im Design erlebt. Die letzte liegt jetzt schon eine Weile zurück. Ich werde für die Hubzilla-Version versuchen, so nahe wie möglich am Original zu bleiben.

Die Landing-Page der Dampfdruck-Presse mit WordPress (künftig als DDP_WP bezeichnet) sieht aktuell so aus:



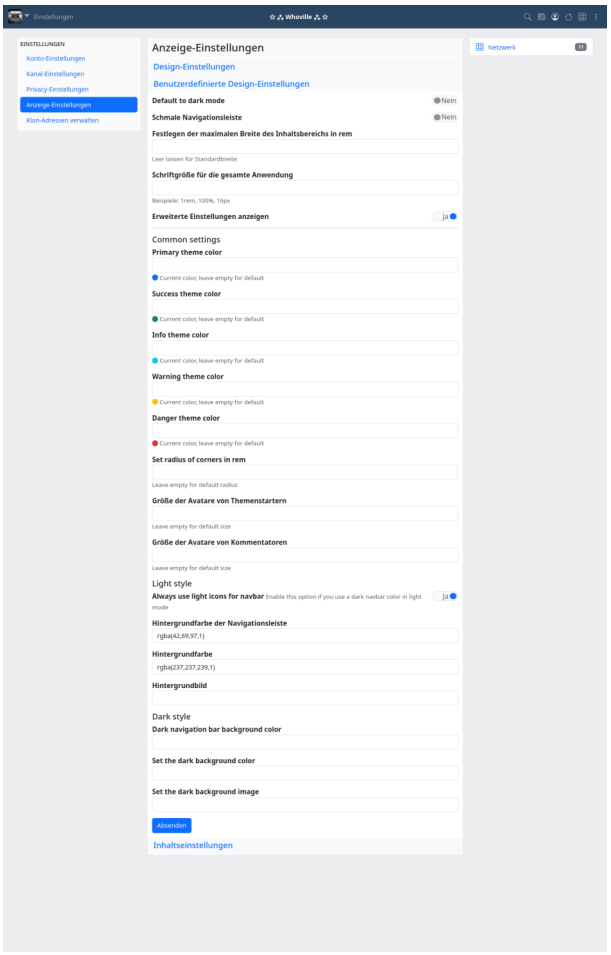
Die Farben der verschiedenen Bereiche und Elemente sollten entsprechend übernommen werden.

- Der Hintergrund des Inhaltsbereichs ist weiß (#FFFFFF).
- Kopf und Fußzeile haben einen stahlblauen Hintergrund (rgba(42,69,97,1)).
- Der Hintergrund der Seite ist hellgrau (rgba(237,237,239,1))

Diese Einstellungen erfolgen unter Einstellungen Anzeige-Einstellungen Benutzerdefinierte Design-Einstellungen.

Ich habe die Farbe für die Navigationsleiste auf rgba(42,69,97,1) gesetzt und die Option "Always use light icons for navbar" auf "Ja", weil die Navigationsleiste durch die Farbwahl nun recht dunkel ist und dunkle Icons und Schrift darauf nicht zu erkennen. Die Option sorgt dafür, dass in der Navigationsleiste helle Schrift und helle Icons erscheinen.

Die Hintergrundfarbe habe ich auf rgba(237,237,239,1) eingestellt.



Unter Einstellungen Anzeige-Einstellungen Inhaltseinstellungen habe ich die Option "Links für neue Mitglieder" abgeschaltet. Brauchen wir nicht, sind ja schon halbe Profis.

Ein Blog: Die Artikel

Die App "Artikel" bietet alles, was man sich für Artikel in einem Blog wünscht. Sicherlich wäre es möglich, Blog-Artikel auch als eigene Webseiten zu erzeugen. Das mag sinnvoll sein, wenn man spezielle Gestaltungsmöglichkeiten benötigt, die mit BBcode nicht darstellbar sind. Diese Vorgehensweise bedingt aber einen erheblichen Mehraufwand, weil man die Artikel-Übersicht, wahrscheinlich ziemlich bald auch paginiert, selbst erstellen und auf dem Laufenden halten muss.

Für die meisten Blogs genügen die Möglichkeiten von Hubzillas BBcode aber völlig aus, weshalb ich für die Dampfdruck-Presse die Artikel mit der App erstelle.

Titelbilder und Artikelbilder organisiert man am besten in der Cloud in dafür angelegten Ordnern. Ich habe mich dafür entschieden, für die Artikel einen Ordner "Artikel" zu erstellen und in diesem jeweils für jeden Artikel einen Ordner mit dem Namen "JJJJ-MM-TT" (für Jahr, Monat und Tag der Artikelerstellung).

Ich habe nun zahlreiche Artikel aus dem DDP_WP-Blog einfach übernommen und in die Artikel-App des Kanals übernommen. Eine gute Methode ist es, den Original-Artikel aufzurufen, alles zu markieren und dann in einen BBcode-Editor (z.B. den SCEditor, welchen man auch sehr gut lokal betreiben kann) zu kopieren. Wenn man dann (nach einigen wenigen Nachbearbeitungen) auf die Code-Ansicht umschaltet, kann man den Inhalt des Editors einfach in den Artikel-Editor von Hubzilla hineinkopieren.

Sinnvoll und ausgesprochen wichtig ist es, mit Kategorien zu arbeiten. Das macht die Navigation insbesondere für Leser wesentlich einfacher.

Ein Blog: Seiten und Layout

Ein Blog benötigt auch klassische Webseiten. So ist es z.B. in den meisten Fällen, dass ein Impressum angeboten wird und dass es eine Datenschutzerklärung gibt. Verbreitet ist auch eine "Über"-Webseite, welche das Blog, seine Thematik und ggf. den oder die Autoren vorstellt.

Ein WordPress Blog zeichnet sich unter anderem dadurch aus, dass alle seine Komponenten mit einem relativ einheitlichen Layout dargestellt werden. Ruft man einen Artikel auf, so unterscheidet er sich nicht wesentlich von der Darstellung einer Seite. Vor allem ist die Navigation der gesamten Präsenz in der Regel einheitlich.

Dies muss auch mit Hubzilla sichergestellt werden. Da z.B. die App "Artikel" genutzt wird und ich für die Dampfdruck-Presse auch die Kanalseite und die Profilseite für Besucher anbieten möchte, muss man sich Gedanken darüber machen, wie das Layout für diese App-Ansichten und die selbst erstellten Webseiten möglichst ähnlich gestaltet werden können.

Der generelle Aufbau einer Präsenz ist natürlich ein Frage des persönlichen Geschmacks. Ich habe mich bei der DDP_WP für ein Layout entschieden, welches am oberen Rand einen Navigationsbereich bietet, darunter befindet sich der eigentliche Inhaltsbereich und es gibt eine rechte Seitenleiste für weitere Informationen und weitere spezielle Navigationsmöglichkeiten (z.B. Artikelkategorien, Archive etc.). Schließlich gibt es noch einen Footer für einige Links, die auf jeder Seite zu sehen sein müssen (Impressum, Datenschutzerklärung) und ein bisserl "Firlefanz".

Diese Struktur musste ich nun auf Hubzilla übertragen.

Die Haupt-Navigation ganz oben lässt sich, wie hier bereits erläutert, sehr gut mit einer Bootstrap-Navbar realisieren. Diese sollte und müsste aber die eigentliche Navigationsleiste von Hubzilla auf den für Blog-Besucher zugänglichen Seiten einheitlich ersetzen. Es soll ja keine Web-Präsenz werden, die irgendwie innerhalb des Sozialen Netzwerks von Hubzilla wie eine embedded Seite dargestellt wird, sondern um eine Web-Präsenz, die als solche erscheint und die lediglich intern von Hubzilla "angetrieben" wird.

Für die App-Ansichten ist das Template "doubleright" das passende für die Dampfdruck-Presse. Hier muss man nur die gewünschten Blöcke und Widgets in die rechte Seitenleiste schieben und im Quelltext die Bootstrap Navar in den Bereich "nav" eintragen. Und für die Webseiten ist das Standard-Layout derzeit am besten geeignet, wobei man dort die vorhandene linke Seitenleiste einfach leer lässt.

Der Footer-Bereich "footer" von Hubzilla ist für das beabsichtigte Layout leider ungeeignet, weil er als schwebendes Overlay und nicht über die gesamte Seitenbreite angezeigt wird. Hier wird der Footer einfach als unterstes Element in den Inhaltsbereich eingefügt.

Abweichend von der DDP_WP enthält der Footer bei der Dampfdruck-Presse unter Hubzilla (ich nenne diese künftig nur noch DDP_HZ) lediglich den Hinweis, mit welchem System sie betrieben wird. Die Links zu Impressum und Datenschutz bringe ich in einem extra Block jeweils in der Seitenleiste auf allen App-Ansichten und im Webseiten-Layout unter.

In der DDP_HZ nenne ich den Block für die Navigationsleiste "menubar", den Block für den Footer "footer1" und den Block für eben erwähnte Seitenleisten-Menü heißt "plinks".

Außerdem werde ich noch einen Block mit einem grafischen Statement zum Thema der Webseite erstellen, welches den Namen "statement" erhält. Und schließlich soll an oberster Stelle der rechten Seitenleiste die kurze Profilkarte des Kanals angezeigt werden.

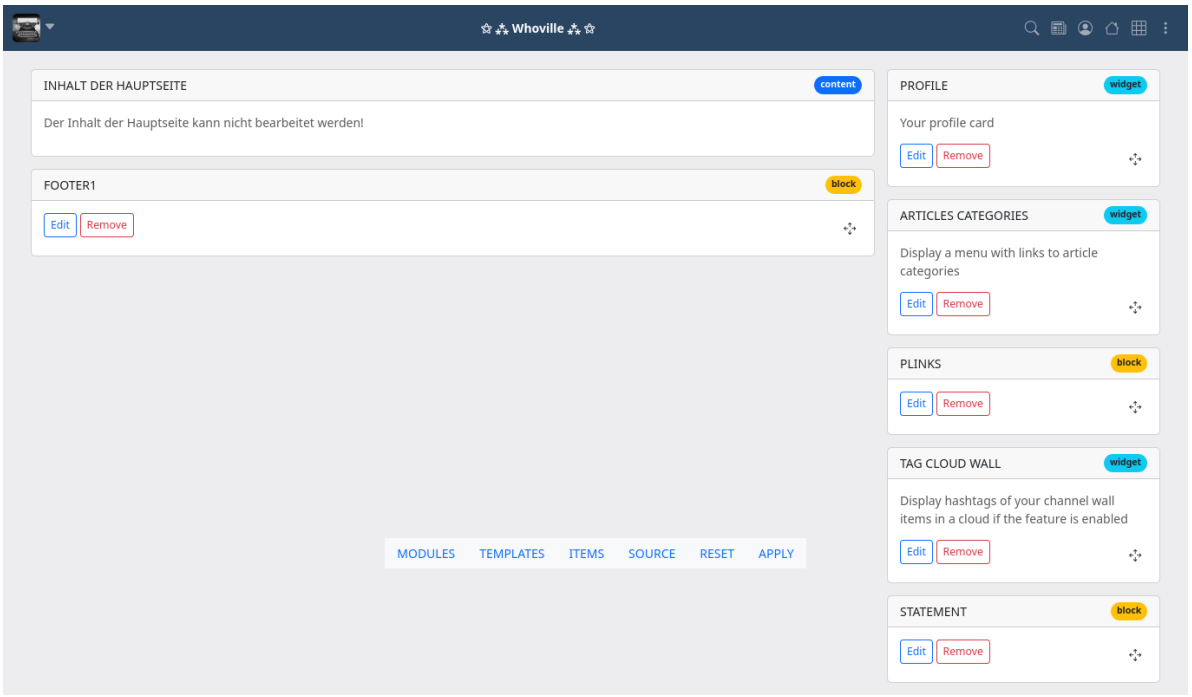
Damit sieht das Webseiten-Layout mit dem Namen "ddpbaselayout" so aus:

```
[region=nav]
[block]menubar[/block]


[/region]
[region=aside]
[/region]


[region=content]
$content
[block]footer1[/block]


[/region]
[region=right_aside]
[widget=profile][/widget]
[block]plinks[/block]
[block]statement[/block]


[/region]
```

Nun müssen nur noch die Layouts für die in der DDP_HZ verwendeten App-Ansichten "Artikel", "Kanal" und "Profil" entsprechend mit dem PDL-Editor angepasst werden.

Für jede Ansicht wird das Template "doubleright" ausgewählt.

Für die Artikel-Ansicht sieht das dann so aus:



Hier sind, zusätzlich zu den im Webseiten-Layout vorhandenen Elemente der rechten Seitenleiste, noch die Widgets "Articles Categories", welches anklickbare Links zur Filterung von Kategorien bietet, und "Tag Cloud Wall" für eine Tag-Cloud der verwendeten Hashtags eingefügt.

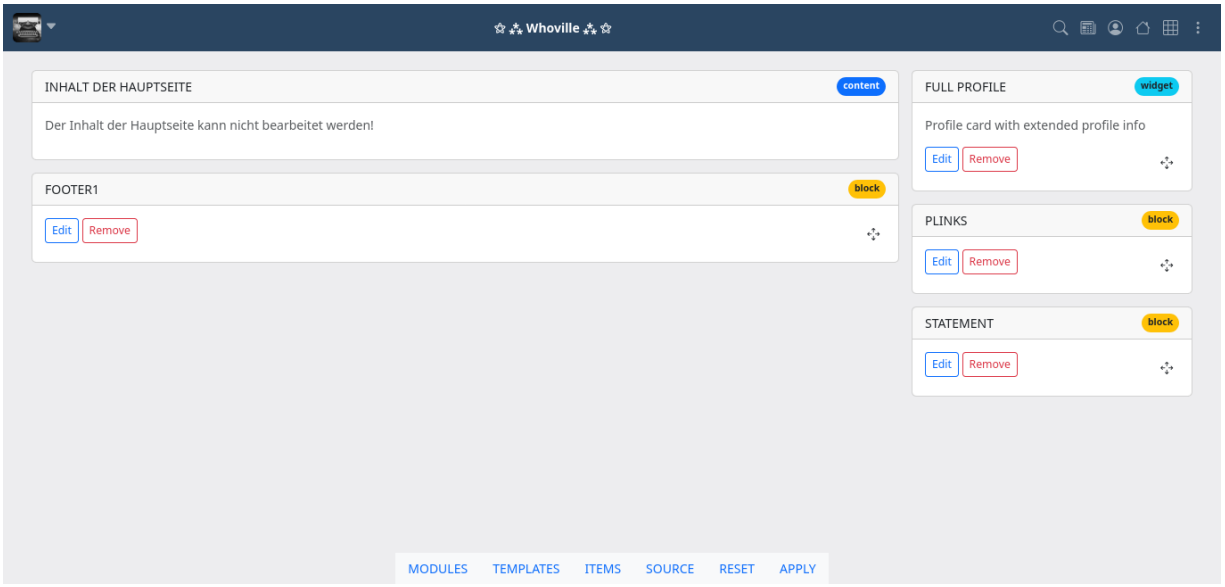
Unter dem "Inhalt der Hauptseite" wurde der Block "footer1" eingefügt.

Der PDL-Editor erlaubt das grafische Bearbeiten der Navigationsleiste nicht. Diese ist bei Modulen grundsätzlich mit der Hubzilla-Navigationsleiste vorbelegt, muss also nicht extra angegeben werden.

Um nun aber diese Navigationsleiste mit der eigenen zu überschreiben, öffnet man im PDL-Editor den Quelltexteditor "Source" und fügt ganz oben `[region=nav] [block]menubar[/block] [/region]` ein:

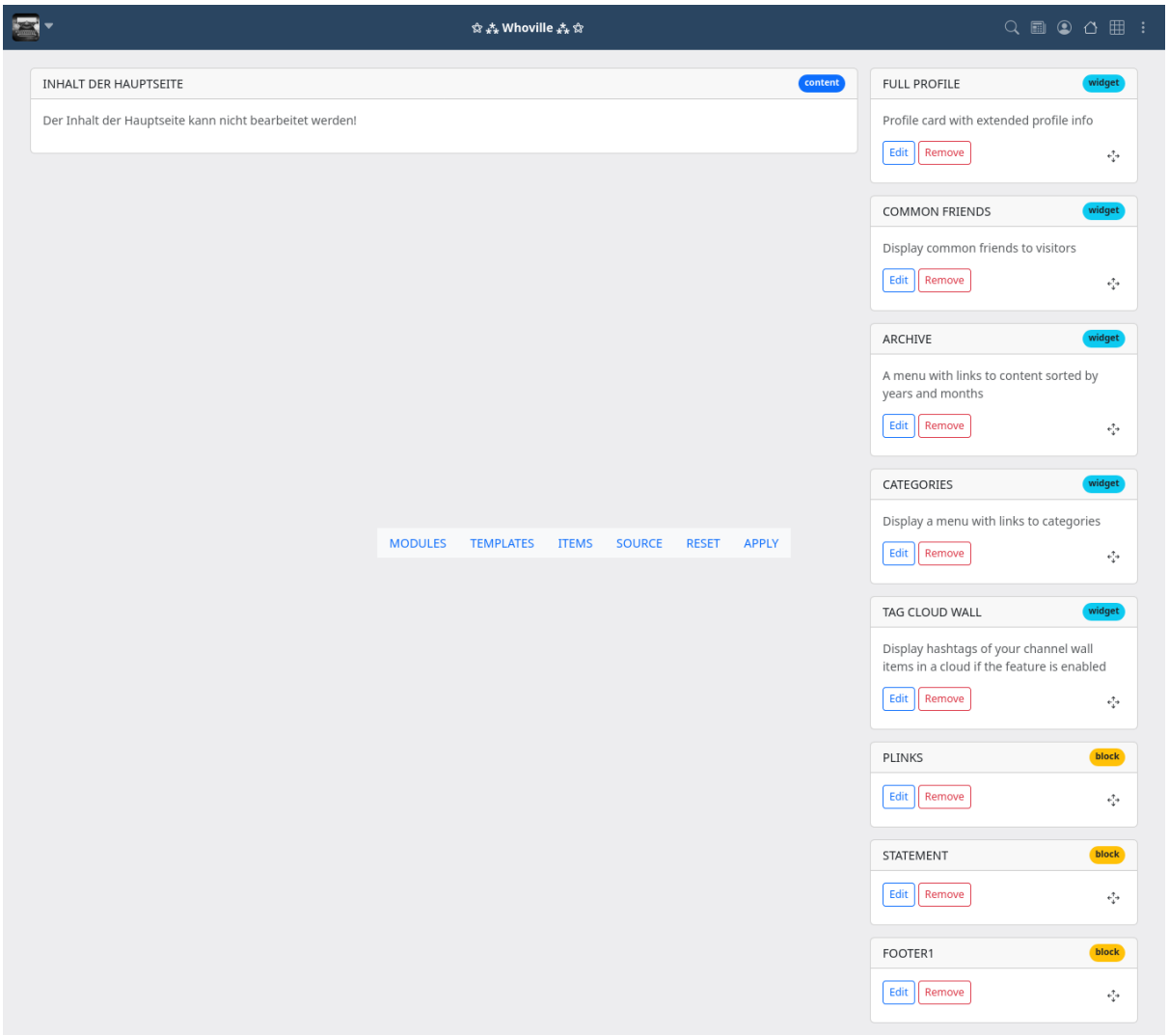
```
[template]doubleright[/template]
[region=nav]
[block]menubar[/block]

[/region]
[region=aside]

[/region]
[region=content]
$content
[block]footer1[/block]

[/region]
[region=right_aside]
[widget=profile][/widget]
[widget=articles_categories][/widget]
[block]plinks[/block]
[widget=tagcloud_wall][var=limit]24[/var][/widget]
[block]statement[/block]

[/region]
```

Die Profil-Ansicht wird so konfiguriert:



Auch hier muss über den Quelltexteditor wieder die "menubar" in die Nav-Region eingefügt werden.

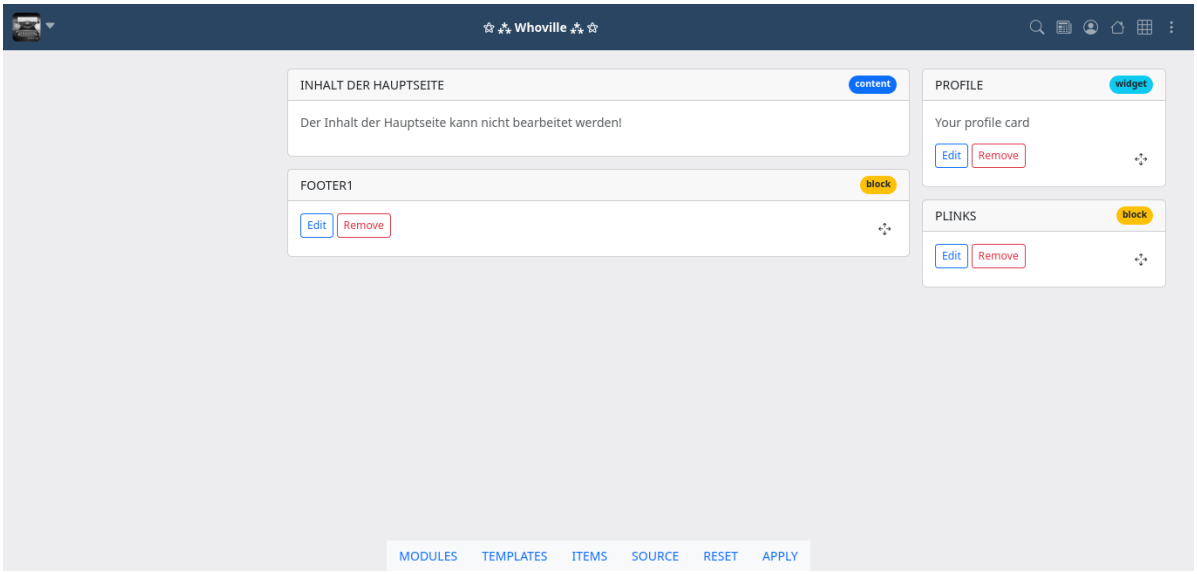
Für die Kanalansicht sieht die PDL-Datei bezüglich des Footers etwas anders aus:



Hier wurde der Block "footer1" nicht unter den Inhaltsbereich gepackt, sondern zuunterst in die rechte Seitenleiste. Der Grund dafür ist, dass es bei Kanal zunächst einmal kein "Ende" des Inhaltsbereichs gibt, unter welchem dann der Footer zu sehen wäre. Das Ende ist erst erreicht, wenn man bis zum ersten Posting des Kanals durchgescrollt hat. Das ist für einen präsenten Footer schlicht ungeeignet.

Und natürlich muss auch hier wieder der Quelltext-Editor für die Navigationsleiste bemüht werden.

Die Download-Seite für eBooks, welche ich anbiete öffnet über Links den jeweiligen öffentlich zugänglichen Ordner der kanaleigenen Cloud, wo die Dateien liegen. Damit wird also auch die Ansicht Cloud geöffnet, die ebenfalls angepasst werden muss:

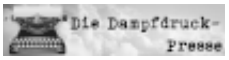


Auch hier darf nicht vergessen werden, die Navigationsleiste im Quelltext-Editor hinzuzufügen.

Damit sind die Arbeiten am Layout für die DDP_HZ erst einmal abgeschlossen.

Ein Blog: Blog-Navigation

Anschließend habe ich die Navigationsleiste für das Blog, also die gesamte Webpräsenz erstellt. Hierfür nutze ich die Navbar von Bootstrap5. Diese bietet, wie hier bereits beschrieben, etliche Vorteile gegenüber einem selbst erstellten horizontalen Menü.

Ich habe zunächst eine Logo-Grafik für das Blog erstellt: 

Diese wird in der Navbar mit dem Link zum HQ hinterlegt, um mir selbst die Möglichkeit zu geben, aus der Webpräsenz "auszubrechen" und die Hubzilla-Navigationsleiste zurück zu bekommen.

Die DDP_HZ wird folgende Webseiten enthalten:

- "impressum"
- "datenschutz"
- "kontakt"
- "aboutddp" ("Über die DDP")
- "koffein0" (Webseite mit Links zu den einzelnen Kapiteln der Koffein-Serie)
- "fasern0" (Webseite mit Links zu den einzelnen Kapiteln der Fasern-Serie)
- "hardware" (Langlebige Hardware - Serie auf einer einzelnen Webseite)
- "liquidsteuerspck" (der Liquidsteuer-Spickzettel)
- "wlinks" (Seite mit Link-Bannern zu verschiedenen Webseiten)
- "downloads" (Seite mit Download-Links)

Die Navigationsleiste "menubar" wird neben dem grafischen Logo

- einen Link zur Artikel-Ansicht ("Home"),
- einen Link zur Kanal-Ansicht ("Kanal)",
- einen Pulldown-Menü Eintrag mit Links zur Koffein-, Fasern- und Hardware-Serie ("DIY"),
- einen Pulldown-Menü Eintrag mit dem Link zum Liquidsteuer-Spickzettel ("Wissenswertes"),
- einen Link zur About-Seite ("Über"),
- einen Link zur Link-Banner Seite ("Mehr...") und
- einen Link zur Profilansicht ("Profil")

enthalten.

Im Pulldown-Menü "DIY" sind noch weitere Einträge zu Serien enthalten, die erst später folgen und derzeit zur Home-Seite führen, enthalten.

Der Quelltext für die Navigationsleiste "menubar" sieht damit wie folgt aus:

```
<nav class="navbar navbar-expand-sm navbar-dark" style="back-
ground-color:#2a4861;">

  <div class="container-fluid">

    <a class="navbar-brand" href="https://hub.hubzilla.hu/hq">

      <img src="https://hub.hubzilla.hu/cloud/dampfdruckpres-
se/misc/navlogo.png" alt="Avatar Logo">

    </a>

  </div>

  <ul class="navbar-nav">

      <li class="nav-item">

        <a class="nav-link" href="https://hub.hubzilla.hu/artic-
les/dampfdruckpresse">Home</a>

      </li>

      <li class="nav-item">

        <a class="nav-link" href="https://hub.hubzilla.hu/chan-
nel/dampfdruckpresse">Kanal</a>

      </li>



      <li class="nav-item dropdown">

        <a class="nav-link dropdown-toggle" href="#" role="button"
data-bs-toggle="dropdown">DIY</a>

        <ul class="dropdown-menu">

          <li><a class="dropdown-item" href="https://hub.hubzil-
la.hu/page/dampfdruckpresse/koffein0">Koffein</a></li>

          <li><a class="dropdown-item" href="https://hub.hubzil-
la.hu/page/dampfdruckpresse/fasern0">Fasern</a></li>

          <li><a class="dropdown-item" href="https://hub.hubzil-
la.hu/articles/dampfdruckpresse">Aromen</a></li>

          <li><a class="dropdown-item" href="https://hub.hubzil-
la.hu/articles/dampfdruckpresse">Selbstmischen</a></li>

          <li><a class="dropdown-item" href="https://hub.hubzil-
la.hu/articles/dampfdruckpresse">Keulkulator</a></li>
```

```
          <li><a class="dropdown-item" href="https://hub.hubzil-
la.hu/page/dampfdruckpresse/hardware">Hardware</a></li>

        </ul>

      </li>



      <li class="nav-item dropdown">

       <a class="nav-link dropdown-toggle" href="#" role="button"
data-bs-toggle="dropdown">Wissenswertes</a>

       <ul class="dropdown-menu">

         <li><a class="dropdown-item" href="https://hub.hubzil-
la.hu/page/dampfdruckpresse/liquidsteuerspck">Liquidsteuer-
Spickzettel</a></li>

       </ul>

      </li>



      <li class="nav-item">

        <a class="nav-link" href="https://hub.hubzil-
la.hu/page/dampfdruckpresse/aboutddp">Über</a>

      </li>

      <li class="nav-item">

        <a class="nav-link" href="https://hub.hubzil-
la.hu/page/dampfdruckpresse/wlinks">Mehr...</a>

      </li>

      <li class="nav-item">

        <a class="nav-link" href="https://hub.hubzilla.hu/profi-
le/dampfdruckpresse">Profil</a>

      </li>

    </ul>



</nav>
```

Der Quelltext des Blocks "plinks", der ein einfaches vertikales Link-Menü ist, sieht so aus:

```
<p>  <a href="https://hub.hubzilla.hu/page/dampfdruckpresse/im-
pressum">Impressum</a></p>
<p>  <a href="https://hub.hubzilla.hu/page/dampfdruckpresse/daten-
schutz">Datenschutz</a></p>
<p>  <a href="https://hub.hubzilla.hu/page/dampfdruckpresse/kon-
takt">Kontakt</a></p>
<p>  <a href="https://hub.hubzilla.hu/page/dampfdruckpresse/down-
loads">Downloads</a></p>
```

Und der Quelltext des Footer-Blocks "footer1" so:

```
<style>
  .my-container {
    /* lokale Variable-Überschreibung nur innerhalb dieses Contai-
ners */
    --bs-link-color: #6eb7ff;
    --bs-link-hover-color: #68acf1;
  }
</style>

<div class="my-container" style="background-color:#2a4861; pad-
ding:10px; text-align:center;">
        <p style="color:#FFFFFF;margin:0;">Powered by <a
href="https://hubzilla.org">Hubzilla</a> - the Social Networking
CMS</p>
</div>
```

Hier musste ich eine Style-Definition vornehmen, weil der Hintergrund des Footers auf dunkelstahlblaue Farbe gesetzt wurde und die Standard-Farbe für Links (es wird ja zur Hubzilla-Projektseite verlinkt) nach meine Geschmack einen zu geringen kontrast zur Hintergrundfarbe besaß. Ich habe die Link-Farben (normal und hover) auf hellere Blautöne gesetzt.

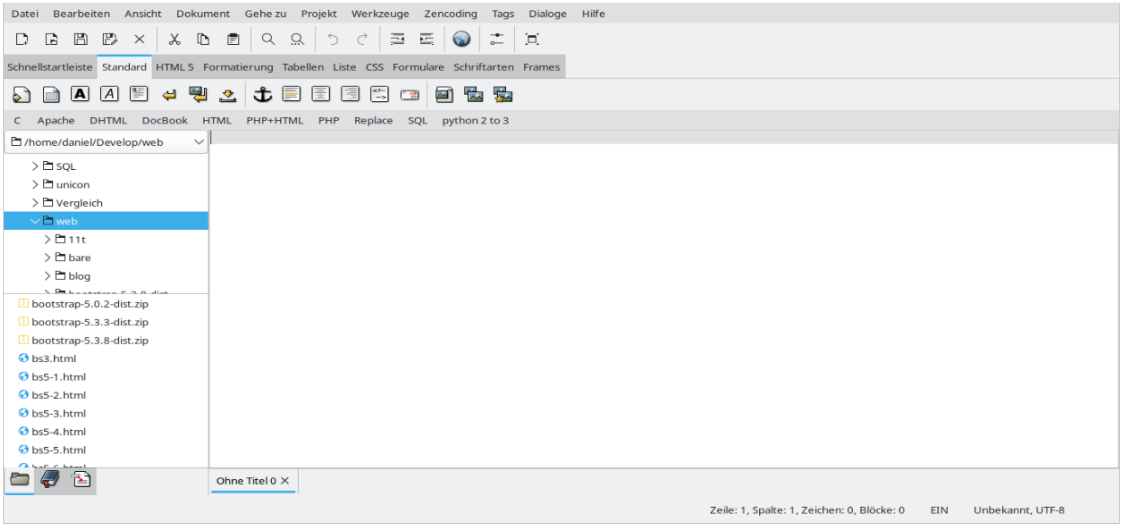
Ein Blog: Blog-Webseiten

Wie bereits im Unterkapitel Blog-Navigation aufgezählt, enthält die Dampfdruck-Presse etliche Webseiten. Einige davon sind eine Art Inhaltsverzeichnis für Serien, die lediglich die Kapitel der Serie als einzelne Webseiten verlinken.

Ich werde hier nicht ausführlich alle Seiten im Detail erläutern, weil ich davon ausgehe, dass das erstellen von Webseiten mit HTML (oder auch mit BBcode) von jedem selbst erledigt werden kann. Große Tricks und Geheimnisse gibt es in der DDP_HZ bei den Webseiten ohnehin nicht. Lediglich auf die Seite Kontakt werde ich noch gesondert eingehen.

Wie man den Inhalt erstellt, ist Geschmackssache. Man kann ihn per Hand im Webseiten-Editor von Hubzilla erstellen, man kann z.B. den TinyMCE wie hier bereits beschrieben als Unterstützung einbinden, oder auf einen externen Editor zurückgreifen und den erzeugten Code in das Editorfeld einfügen.

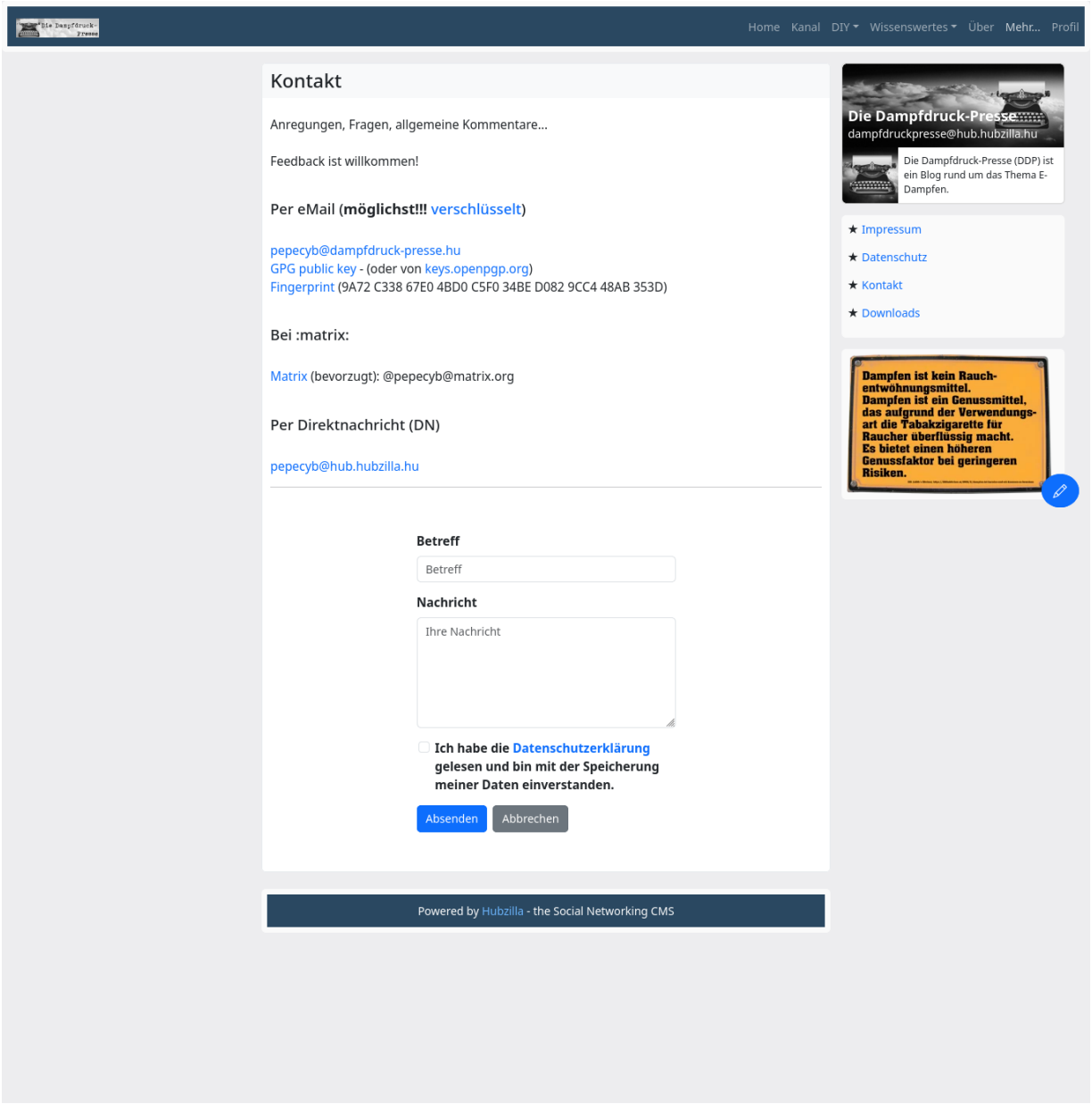
Ich weiß nicht, was die nette Zahnarztfrau empfiehlt, aber ich empfehle Bluefish. Scherz beiseite! Ich persönlich nutze den Bluefish-Editor, weil er zu den ausgefeiltesten HTML-Editoren gehört. Er verfügt zwar über keinen WYSIWYG-Modus (bietet aber eine Vorschau), erleichtert das Erstellen von HTML durch seinen Feature-Reichtum aber enorm.



Für alle Webseiten nutze ich das im Kapitel Seiten und Layout erstellte Layout "ddpbaselayout".

Ein Blog: Die Kontakt-Seite

Auch die Kontakt-Webseite habe ich der original DDP_WP entlehnt. Für das Kontaktformular, welches ich ans Ende der Seite gestellt habe, nutze ich die bereits beschrieben Variante aus dem Kapitel Kontaktformular. Als Empfänger Mailadresse ist im Formular-Quelltext meine DDP-Mailadresse eingetragen und der Link zur Datenschutzerklärung ist mit der Hubzilla-Webseite der Datenschutzerklärung hinterlegt.



```
<p>Anregungen, Fragen, allgemeine Kommentare...<br /><br />Feed-
back ist willkommen!<br /><br /></p>

<h3>Per eMail (<strong>möglichst!!!</strong> <a
href="https://www.kuketz-blog.de/verschluesselte-e-mails-mit-
gnupg-als-supergrundrecht/" rel="nofollow noopener">verschlüs-
selt</a>)</h3>

<p><br /><a href="mailto:pepecyb@dampfdruck-presse.hu" rel="nofol-
low noopener">pepecyb@dampfdruck-presse.hu</a><br /><a
href="https://hub.hubzilla.hu/cloud/dampfdruckpresse/misc/pepe-
cyb%40dampfdruck-presse_pubkey.asc" rel="nofollow noopener">GPG
public key</a> - (oder von <a
href="https://keys.openpgp.org/vks/v1/by-finger-
print/9A72C33867E04BD0C5F034BED0829CC448AB353D" rel="nofollow noo-
pener">keys.openpgp.org</a>)<br /><a href="https://hub.hubzil-
la.hu/cloud/dampfdruckpresse/misc/pepecyb%40dampfdruck-pres-
se_fingerprint.asc" rel="nofollow noopener">Fingerprint</a> (9A72
C338 67E0 4BD0 C5F0 34BE D082 9CC4 48AB 353D)<br /><br /></p>

<h3>Bei :matrix:</h3>

<p><br /><a href="https://matrix.org/" rel="nofollow noopener">Ma-
trix</a> (bevorzugt): @pepecyb@matrix.org<br /><br /></p>

<h3>Per Direktnachricht (DN)</h3>

<p><br /><a href="https://hub.hubzilla.hu/channel/pepecyb"
rel="nofollow noopener">pepecyb@hub.hubzilla.hu</a></p>

<hr>

  <div class="container py-5">

    <div class="row justify-content-center">

      <div class="col-md-8 col-lg-6">


        <form id="contactForm" novalidate>

          <div class="mb-3">

            <label for="subject" class="form-label">Betreff</la-
bel>

            <input type="text" class="form-control" id="subject"
name="subject" required placeholder="Betreff">

            <div class="invalid-feedback">Bitte geben Sie einen
Betreff ein.</div>

          </div>
```

```
          <div class="mb-3">
            <label for="message" class="form-label">Nachricht</label>
            <textarea class="form-control" id="message" name="message" rows="6" required placeholder="Ihre Nachricht"></textarea>
            <div class="invalid-feedback">Bitte geben Sie eine Nachricht ein.</div>
          </div>

          <div class="form-check mb-3">
            <input class="form-check-input" type="checkbox" value="1" id="privacy_agree" required>

            <!-- Hier die URL zur Datenschutzerklärung einfügen -->

            <label class="form-check-label" for="privacy_agree">
              Ich habe die <a href="https://hub.hubzilla.hu/page/dampfdruckpresse/datenschutz" target="_blank" rel="noopener">Datenschutzerklärung</a> gelesen und bin mit der Speicherung meiner Daten einverstanden.
            </label>
            <div class="invalid-feedback">Sie müssen zustimmen, um die Nachricht zu senden.</div>
          </div>

          <div class="d-flex gap-2">
            <button type="submit" class="btn btn-primary" id="btnSend">Absenden</button>
            <button type="button" class="btn btn-secondary" id="btnReset">Abbrechen</button>
          </div>
        </form>
```

```
      </div>
    </div>
  </div>

  <script>
  (function () {

    // Konfiguration (hier die Empfängeradresse anpassen)

    const recipient = 'pepecyb@dampfdruck-presse.hu'; // <-- Emp-
fängeradresse hier eintragen

    const form = document.getElementById('contactForm');
    const subjectEl = document.getElementById('subject');
    const messageEl = document.getElementById('message');
    const privacyEl = document.getElementById('privacy_agree');
    const btnReset = document.getElementById('btnReset');

    // Helper: mailto-encode
    function encodeMailto(text) {
      return encodeURIComponent(text).replace(/%0A/g, '%0D%0A');
    }

    form.addEventListener('submit', function (e) {
      e.preventDefault();
      // HTML5 validity
      if (!form.checkValidity()) {
        form.classList.add('was-validated');
        return;
```

```
    }
    if (!privacyEl.checked) {
      privacyEl.classList.add('is-invalid');
      return;
    } else {
      privacyEl.classList.remove('is-invalid');
    }

    const subject = subjectEl.value.trim();
    const message = messageEl.value.trim();

    // Build mailto link
    let mailto = 'mailto:' + encodeURIComponent(recipient)
               + '?subject=' + encodeMailto(subject)
               + '&body=' + encodeMailto(message);

    // Open default mail client
    window.location.href = mailto;

    // Optional: reset form after triggering mailto
    form.reset();
    form.classList.remove('was-validated');
  });

  btnReset.addEventListener('click', function () {
    form.reset();
    form.classList.remove('was-validated');
  });
})();
</script>
```

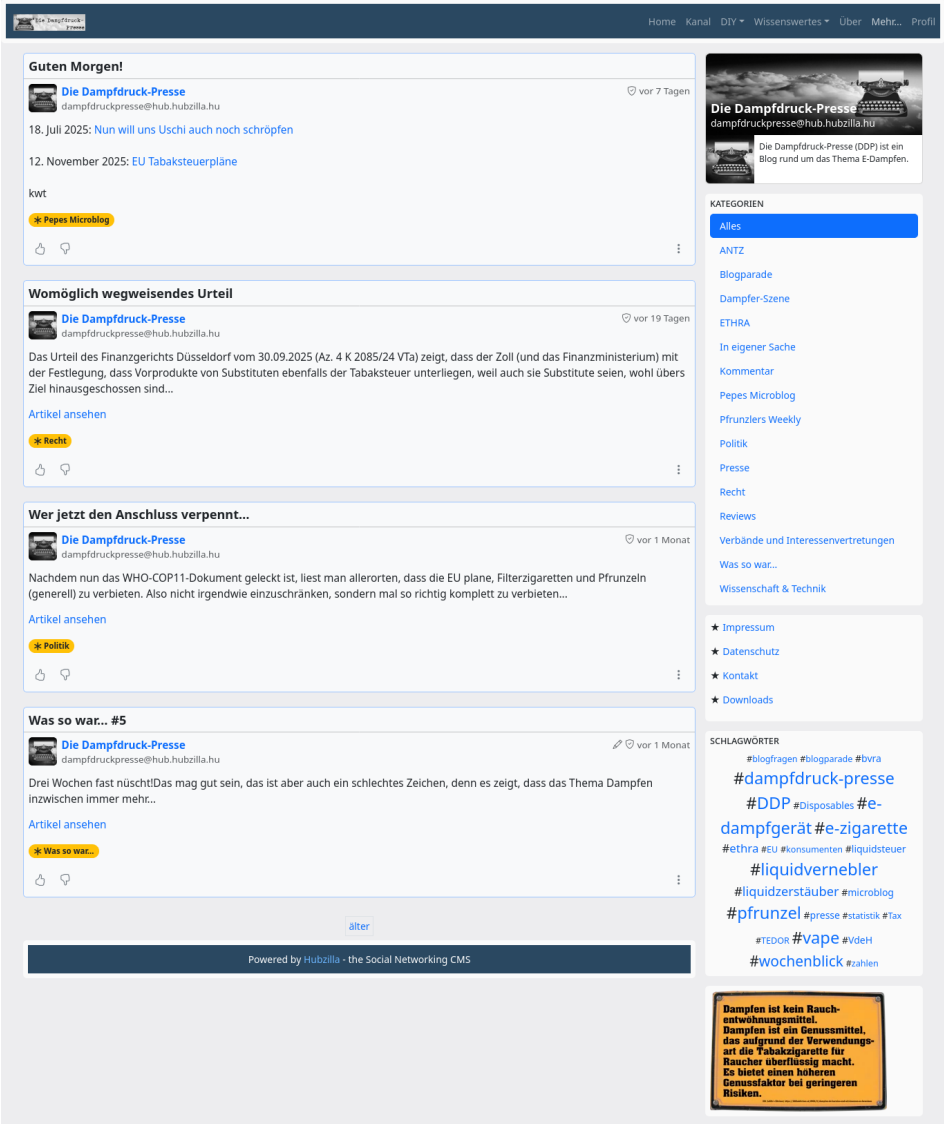
Ein Blog: Fertig!

Viel mehr braucht es nicht, um ein Blog als in sich stimmige Webpräsenz mit Hubzilla zu erstellen und zu pflegen.

Die URL für das Blog ist dann natürlich immer der Link zu den Artikeln des Kanals: `<URL_DES_HUB>/articles/<KANALNAME>`. Also für die DDP_HZ: https://hub.hubzilla.hu/articles/dampfdruckpresse.

Wen man über eine Domain verfügt, die man für das Blog als möglichst einprägsame Web-Adresse verwenden möchte, bietet sich eine permanente Umleitung auf die Artikel-App an.

So habe ich das auch mit der Dampfdruck-Presse gemacht. Während https://dampfdruck-presse.hu auf das WordPress Blog verweist, erreicht man die Hubzilla Dampfdruckpresse über https://dampfdruck-presse.de.



Statische Seiten anbieten

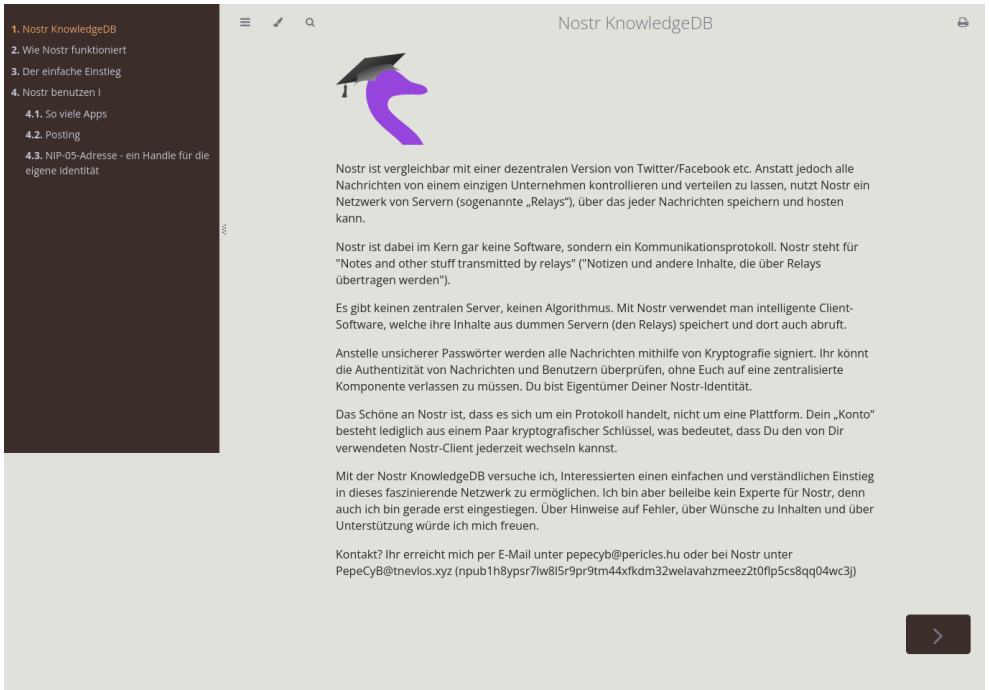
Mit Hubzilla sind Sie auch in der Lage, statische Webseiten anzubieten.

Beispiel: mdBook-Seiten

So ist dieses Buch hier nicht mit Hubzilla, sondern mit dem Seitengenerator mdBook erstellt. Ich nutze dieses System gerne und habe den Workflow inzwischen verinnerlicht. Den gesamten Inhalt erstellt man damit mit Marksown-Dateien, die dann in einer weiteren Marksown-Datei (SUMMARY.md) in eine unsortierte Liste verlinkt werden. mdBook kann die Seite mit einem eigenen kleinen Webserver lokal so anzeigen, wie sie letztlich auch auf dem Server ausgeliefert wird. Änderungen an den Quell-Dateien werden dabei ständig sofort übernommen, sodass man jederzeit sieht, wie sich diese auf das Gesamtwerk auswirken.

Mit einem anderen Befehl erzeugt mdBook dann die komplette Struktur der statischen Webseite in einem eigenen Verzeichnis. Diesen Inhalt muss man nur auf den Server hochladen und die Seite ist online.

Ich habe versuchsweise die mdBook-Webseite "Nostr KnowledgeDB", die ich normalerweise unter https://ninfo.tnevlos.xyz/ öffentlich anbiete, in das Verzeichnis "nostr_kdb" meines Kanals "thesmokinggnu@klacker.org" hochgeladen. Ruft man nun die Datei https://klacker.org/cloud/thesmokinggnu/nostr_kdb/index.html auf, wird die mdBook-Seite ganz normal dargestellt und man kann innerhalb der Seite mit sämtlichen Features, die mdBook bietet, surfen.



Beispiel: Hugo-Seiten

Ein sehr bekanntes System zur Erzeugung statischer Webpräsenzen ist Hugo. Auch Hugo-Wenseiten kann man benutzbar in der Hubzilla-Cloud anbieten. Man muss dafür in der Datei hugo.toml bei baseURL die URL des Hubzilla Cloud-Verzeichnisses eintragen und (wichtig) einen weiteren Eintrag uglyURLs = true anfügen (dieser sorgt dafür, dass Verweise innerhalb der Seitenstruktur immer als vollständige Dateinamen referenziert werden).

Als Beispiel habe ich das "berühmte" Quickstart-Beispiel aus der Hugo-Dokumentation verwendet und die erzeugte Seitenstruktur in das Verzeichnis "gohugoqs" hochgeladen. Ruft man nun https://hub.hubzilla.hu/cloud/pepecyb/gohugoqs/index.html auf, wird die Hugo-Webseite vollständig nutzbar dargestellt.