ZfdG

Zeitschrift für digitale Geisteswissenschaften

Projektvorstellung aus:
Zeitschrift für digitale Geisteswissenschaften, Heft 11 (2026)

Titel:
Digitale Editionen als statische Webseiten. Zur nachhaltigen Publikation TEI-XML-basierter Editionen mit dem dse-static-cookiecutter

Autor*in:
Peter Andorfer

Kontakt: peter.andorfer@oeaw.ac.at
Institution: Austrian Centre for Digital Humanities, Österreichische Akadamie der Wissenschaften
GND: 1043833846 ORCID: 0000-0002-9575-9372
Contribution (CRediT): Writing – original draft | Writing – review & editing

DOI des Beitrags:
10.17175/2026_003

Nachweis im OPAC der Herzog August Bibliothek:
1961596229

Erstveröffentlichung:
08.04.2026

Letzte Überprüfung aller Verweise:
23.02.2026

Format:
PDF ohne Paginierung, Lesefassung

GND-Verschlagwortung:
Digitale Edition | Edition | Web-Seite | Standardisierung

Empfohlene Zitierweise:
Peter Andorfer: Digitale Editionen als statische Webseiten. Zur nachhaltigen Publikation TEI-XML-basierter Editionen mit dem dse-static-cookiecutter. In: Zeitschrift für digitale Geisteswissenschaften 11 (2026). 08.04.2026. HTML / XML / PDF. DOI: 10.17175/2026_003

Peter Andorfer

Digitale Editionen als statische Webseiten. Zur nachhaltigen Publikation TEI-XML-basierter Editionen mit dem dse-static-cookiecutter

Abstract

Wie können digitale Editionen stabil und nachhaltig online publiziert werden? Eine Antwort auf diese Frage bietet der *dse-static-cookiecutter*. Dabei handelt es sich um einen am Austrian Centre for Digital Humanities (ACDH) entwickelten Static-Site-Generator. Mehrere digitale Editionsprojekte basieren bereits auf dem Tool, das aus TEI-XML-kodierten Textdateien Webseiten generiert und kostenfrei über den Service GitHub-Pages veröffentlicht. Diese Projektvorstellung behandelt die drei Konzepte im Kern des *dse-static-cookiecutter*: Erstens das Paradigma statischer Webseiten, zweitens das Prinzip definierter Workflows (im Sinne automatisch ablaufender Prozesse, die in einer zentralen, sowohl von Menschen als auch Maschinen lesbaren Textdatei konfiguriert werden), und drittens die Verwendung von Code Templates, die helfen, unterschiedliche Softwareprojekte ähnlich zu strukturieren. Außerdem werden die Vor- und Nachteile einer solchen statischen digitalen Edition thematisiert und der *dse-static-cookiecutter* in der aktuellen Landschaft digitaler Editionen verortet.

How can digital editions be published online in a stable and sustainable manner? One answer to this question is provided by *dse-static-cookiecutter*, a static site generator developed at the Austrian Centre for Digital Humanities (ACDH). Several digital edition projects are already based on the tool, which generates websites from TEI-XML-encoded text files and publishes them free of charge via the GitHub Pages service. This project presentation discusses the three concepts at the core of *dse-static-cookiecutter*: firstly, the paradigm of static websites; secondly, the principle of defined workflows (in the sense of automatically running processes that are configured in a central text file readable by both humans and machines); and thirdly, the use of code templates that help to structure different software projects in a similar way. In addition, the advantages and disadvantages of such a static digital edition are discussed, and the *dse-static-cookiecutter* is contextualised in the current landscape of digital editions.

1. Probleme und Begriffe

[1] Digitale Editionen werden oftmals im Rahmen von auf zwei bis drei Jahren befristeten drittmittelfinanzierten Projekten realisiert.[1] Während dieser Zeit werden ausgewählte Texte im TEI-XML-Format kodiert. Die Art der Auszeichnung und der verwendeten TEI-Tags orientiert sich an der jeweiligen Fragestellung und dem zu edierenden Material. Die TEI-XML-Kodierung ist für die Forscher*innen dabei primär Mittel zum Zweck der Veröffentlichung einer digitalen Edition. In der Regel verstehen sie darunter eine Webseite zur Darstellung, Publikation und Durchsuchbarkeit (Volltextsuche, registerbasierte Suche) ihrer edierten Texte. Für die Realisierung dieser Webseite werden Kooperationen mit einschlägigen Institutionen wie etwa dem Austrian Centre for Digital Humanities (ACDH) eingegangen. Von diesen Institutionen wird erwartet, dass sie nach der Entwicklung einer qualitativ hochwertigen und aktuellen Standards[2] entsprechenden digitalen Edition das stabile und langfristige Hosting der Edition betreiben. Seitens der Projektpartner*innen und Fördergeber*innen stehen für gewöhnlich nach Projektabschluss keine finanziellen Ressourcen mehr zur Verfügung.[3] Was ist nun konkret als Zeitraum für den Bestand einer Edition anzunehmen? Zwei Jahre scheint zu kurz, zehn Jahre scheint lang. Wer weiß schon, wie das Internet in zehn Jahren funktioniert? Am ACDH wird grundsätzlich garantiert, die im Projekt entwickelten Applikationen, worunter auch digitale

[1] Vgl. folgende Auswahl digitaler Editionsprojekte, welche mit Beteiligung des ACDH aktuell realisiert wurden bzw. werden, allesamt vom Österreichischen Wissenschaftsfonds FWF finanziert: Edition von Tillichs Vorlesung über Religion und Kultur, Laufzeit: 2024–2026; Edition des Briefwechsels von Paul Tillich (1887–1933), Laufzeit: 2021–2024; Die Korrespondenz Paul Tillichs 1933–1951, Laufzeit: 2024–2027; Die Entstehung des Bundes-Verfassungsgesetzes 1920, Laufzeit: 2023–2026; Die Protokolle des österreichischen Kabinettsrates 1919–1920, Laufzeit 2026–2029; Familiensache. Dynastische Handlungsspielräume von Frauen, Laufzeit: 2021–2025; Briefwechsel Arthur Schnitzlers mit Schriftstellern 3, Laufzeit: 2025–2029; Auden in Österreich Digital, Laufzeit: 2024–2027; Eine analytische Rekonstruktion von Hanslicks Ästhetik, Laufzeit: 2025–2027. Für eine aktuelle Liste von Editionsprojekten, die mit dem *dse-static-cookiecutter* realisiert wurden, vgl. Andorfer et al. 2024: Readme.md.

[2] Vgl. Sahle 2014.

[3] Vgl. Helling 2024.

Editionen fallen, zwei Jahre nach Projektende online zu halten. Nach diesem Zeitraum werden diese weiter automatisch überwacht[4] und technisch notwendige Updates – im Falle von überschaubarem Aufwand und unter Berücksichtigung der personellen Ressourcen – durchgeführt. Als zusätzliche Erschwernis kommt es immer wieder vor, dass Applikationen auf veralteter Technologie basieren und die ursprünglichen Entwickler*innen der Applikation nicht mehr zur Verfügung stehen. Derartige Applikationen müssen nach einer abschließenden technischen Evaluierung und der Sicherung der Daten und des Quellcodes vom Netz genommen werden.[5]

[2] Was bedeutet ›langfristig‹ nun im Kontext dieses Beitrages? Langfristig heißt hier, dass eine auf den Prinzipien des *dse-static-cookiecutter*[6] basierende digitale Edition so lange online bleiben kann, wie der institutionelle Kontext, in dem die Edition publiziert wurde, nicht strukturell geändert oder aufgelöst wird. Das heißt auch, dass das Überleben der Applikation weder an die Entwicklung einer bestimmten Technologie gekoppelt ist, noch an die Karriere einer einzelnen Person.

[3] Bleibt noch zu klären, was der vorliegende Beitrag unter einer digitalen Edition versteht.[7] Eine digitale Edition beschreibt hier eine Reihe von TEI-XML-Dateien – in der Regel handelt es sich dabei um eine »erschließende Wiedergabe historischer Dokumente«[8] – die in einer kuratierten Form als HTML-Dateien im Kontext einer Webseite im Internet zugänglich gemacht werden. Eine digitale Edition besteht also neben den Daten auch aus einem spezifisch für diese Daten entwickelten Interface, einer je nach Machart mehr oder weniger individualisierbaren Leseansicht.

2. Statisch, automatisch und ähnlich

[4] Am ACDH versucht man dem Versprechen, digitale Edition langfristig am Leben zu erhalten, mit Hilfe dreier Prinzipien gerecht zu werden. Diese Prinzipien werden von den Schlagworten ›statisch‹, ›automatisch‹ und ›ähnlich‹ skizziert.

[5] ›Statisch‹ – im Gegensatz zu ›dynamisch‹ – bezieht sich hier auf ein bestimmtes Konzept von Webseiten, nämlich solche, deren Inhalte bereits als eigenständige (statische) HTML-Dateien vorliegen. Im Unterschied zu vielen komplexeren Webseitenlösungen stellt das sicher, dass eine einzelne Seite nicht bei jeder Abfrage, bei jedem Aufruf neu berechnet werden muss.[9] Das reduziert den Anspruch beim Hosting und bei der Maintenance solcher Webseiten deutlich, da weder eine Datenbank benötigt wird noch ein webseiten-spezifischer Code am Server ausgeführt werden muss. Somit fallen gleich zwei Komponenten dynamischer Webseiten weg, die von regelmäßig notwendigen Updates betroffen wären und potentielle Sicherheitslücken beinhalten könnten. Vor allem bedeutet jedes Update, dass an solchen Schnittstellen danach eine Bruchstelle entstehen kann, die die ganze Seite unbrauchbar macht. Bis zum Ausrollen des *dse-static-cookiecutters* wurden digitale Edition am ACDH als dynamische Webapplikation mittels eXist-db, einer Java basierten, nativen XML-Datenbank publiziert. Dies war insofern eine naheliegende Entscheidung, als *eXist-db* neben der Verwaltung von XML-Dateien auch Tools und Funktionen bereitstellt, die es erlauben, dynamische, datengetriebene Webapplikationen zu entwickeln und zu publizieren. Dazu zählen etwa ein integrierter Webserver (Jetty), eine XQuery-basierte Templating-Engine, ein leistungsstarker Suchindex (Lucene), eine integrierte Entwicklungsumgebung (eXide), eine Anbindung an den Oxygen-XML Editor und vor allem auch eine in den digitalen Geisteswissenschaften verankerte Community. Diesen Vorteilen standen die Nachteile solcher dynamischen Lösungen gegenüber, vor allem das regelmäßige Nachziehen von Updates von neuen

[4] Dies erfolgt mit der Open-Source-Software Icinga.
[5] Beispiele hier: rest in peace.
[6] Andorfer et al. 2024.
[7] Vgl. Driscoll / Pierazzo (Hg.) 2016.
[8] Sahle 2016, S. 23.
[9] Vgl. Wikipedia: Statische Webseite.

eXist-db Versionen als auch der benötigten Java-Umgebung. Unter dem Gesichtspunkt der Langfristigkeit muss außerdem bedacht werden, dass es sich bei einer Software wie *eXist-db* um ein vergleichsweise kleines Open-Source-Projekt handelt, welches eng an einige wenige Developer*innen geknüpft ist.[10]

[6] Das Schlagwort ›automatisch‹ beschreibt die Aktionen, die notwendig sind, um die fertigen HTML-Dateien der statischen Webseite aus den TEI-XML-Dateien der Edition zu generieren und online zu publizieren. Dieser Build-Prozess kann je nach Projekt mehrere Schritte umfassen. Welche Schritte dies sind, welche Umgebung (Betriebssystem, Software, welche Versionen) diese benötigen und in welcher Reihenfolge diese auszuführen sind, ist dabei in einer menschen- wie maschinenlesbaren Datei konfiguriert. Ein solches Setup erlaubt es, die digitale Edition unabhängig von den ursprünglichen Entwickler*innen respektive deren Arbeitsgeräten neu zu bauen und zu publizieren. Im *dse-static-cookiecutter* wird ein solcher automatischer Build-Prozess mit Hilfe einer YAML-Datei konfiguriert, welche vom CI/CD-Service GitHub-Actions interpretiert und ausgeführt wird. Alternativ dazu generiert der *dse-static-coociecutter* auch ein Dockerfile, wodurch das Prinzip ›automatisch‹ auch ohne Rückgriff auf die Infrastruktur von GitHub realisiert ist.

[7] Das dritte Schlagwort ›ähnlich‹ bezieht sich auf den im Build-Prozess ausgeführten Programmcode und dient dazu, die digitalen Editionen noch stärker von ihren Entwickler*innen zu entkoppeln. Statische digitale Editionen, die mit dem *dse-static-cookiecutter* realisiert werden, sind hinsichtlich der verwendeten Technologien, der Organisation der TEI-XML-Daten, des Quellcodes sowie des Build-Prozesses ähnlich. Das erleichtert es für Entwickler*innen und Systemadministrator*innen, allfällig notwendige Adaptionen an den Editionen vorzunehmen, sobald sie sich mit der Codebase einer einzelnen Edition vertraut gemacht haben.[11]

3. dse-static-cookiecutter

[8] Der *dse-static-cookiecutter* ist ein Static-Site-Generator für TEI-XML-Dateien. Das Ziel ist es, *digital (scholarly) editions* – abgekürzt *DSE* – als statische (*static*) Webseiten zu veröffentlichen. Implementiert ist dies auf Basis der in Python geschriebenen »cross-platform command-line utility« cookiecutter, womit Name und Funktionalität in aller Kürze beschrieben wären.

[9] Um den *dse-static-cookiecutter* für ein neues Projekt einzusetzen, müssen sowohl Python als auch das Python-Package *cookiecutter* installiert sein. Danach wird mit dem Befehl `cookiecutter gh:acdh-oeaw/dse-static-cookiecutter` der Initialisierungsprozess einer neuen, auf *dse-static-cookiecutter* basierten digitalen Edition gestartet. Nun werden ein Dutzend Fragen gestellt, beispielsweise nach dem Titel der Edition, der zukünftigen URL, dem Speicherort der TEI-XML-Dateien,[12] der Standardsprache des User Interfaces bzw. ob Internationalisierung[13] implementiert werden soll.

[10] Vgl. dazu etwa die Parameter ›Starred‹ und ›Contributors‹ der GitHub-Repositoriums von eXist-db (449 und 80) und des Python-basierten Webframeworks Django (85.000 und 2679). Zahlen vom 15.09.2025.
[11] Hinsichtlich Form und Aufbau der einzelnen XML-Dateien der Editionseinheiten gibt es seitens des *dse-static-cookiecutter* allerdings keine technisch sichergestellten Vorgaben wie ein dezidiertes Schema.
[12] Empfohlen wird je ein eigenes Repositorium für die Daten und die Webseite.
[13] Vgl. Wikipedia: Internationalisierung (Softwareentwicklung).

Abb. 1: Screenshot aus dem Initialisierungsprozess eines neuen digitalen Editionsprojektes. [Screenshot: Peter Andorfer 2025]

Alle Fragen verfügen über vorgefertigte Antwortmöglichkeiten. Die meisten Antworten können außerdem nachträglich in einer globalen Konfigurationsdatei (`xslt/partials/params.xsl`) geändert werden. [10]

Das Ergebnis dieses Initialisierungsprozesses ist ein vom *dse-static-cookiecutter* angelegtes Dateiverzeichnis, welches aktuell 38 Ordner und über 2300 Dateien umfasst. Von diesen Dateien entfällt der Großteil auf Open-Source-Drittbibliotheken wie etwa das CSS-Framework Bootstrap, den Image-Viewer OpenSeadragon oder den XSLT-Prozessor Saxon. Zieht man diese ab, bleiben nur noch rund 60 Dateien übrig. [11]

Den eigentlichen Kern des Quellcodes bilden XSLT-Dateien, welche die TEI-XML-Dokumente der Edition nach HTML transformieren. Der Transformationsprozess ist in einer ANT-XML-Datei namens `build.xml` konfiguriert und wird von dem bereits erwähnten XSLT-Prozessor *Saxon* ausgeführt. Vereinfacht gesagt, definiert man in `build.xml`, welche Dateien mit welchem Stylesheet konvertiert werden und wohin und unter welchem Namen die erzeugten HTML-Dateien gespeichert werden sollen. Die Konvertierung kann mit dem Kommandozeilenbefehl `ant` ausgeführt werden, wofür ANT am Rechner installiert sein muss. Alternativ kann die Konvertierung über den XML-Editor Oxygen durchgeführt werden. Dieser gängige XML-Editor wird vom *dse-static-cookiecutter* von Haus aus unterstützt, indem bereits während der Initialisierung einer neuen Edition eine Oxygen-Projektdatei angelegt wird, die die Transformationsszenarien zur Konvertierung der TEI-XML-Dateien nach HTML mittels XSLT enthält. [12]

Die XSLT-Dateien sind modular aufgebaut. Sogenannte *partials* sind für Teile des zu generierenden HTML-Codes zuständig und können von anderen XSLT-Dokumenten eingebunden werden. Beispiele dafür wären etwa: `partials/html_footer.xsl`, `partials/html_navbar.xsl` oder `partials/html_head.xsl`. Damit wird die Duplizierung von Code reduziert und Elemente, die auf allen generierten HTML-Seiten anzutreffen sind, wie etwa ein Navigationsmenü oder eine Fußzeile, lassen sich in nur einer einzigen Datei ändern. [13]

Spezialseiten wie eine Startseite (`index.html`), das Impressum (`imprint.html`) oder eine 404-Seite (`404.html`) werden mit entsprechenden XSLT-Dateien (`index.xsl`, `imprint.xsl`, `404.xsl`) generiert. [14]

Die eigentlichen TEI-XML-kodierten Editionsdateien werden vom *dse-static-cookiecutter* in dem Verzeichnis `data/editions` erwartet. TEI-XML-kodierte Paratexte (z.B. ›Über das Projekt‹, ›Editionsrichtlinien‹, ›Team‹, …) in `data/meta`. Vom *dse-static-cookiecutter* werden gleichnamige XSLT-Dateien (`editions.xsl` und `meta.xsl`) zur Verfügung gestellt, welche sämtliche in den genannten Verzeichnissen abgelegten Dateien nach HTML konvertieren. [15]

[16] Die HTML-Standarddarstellung der einzelnen Editionseinheiten ist bewusst einfach gehalten, was der großen Heterogenität von Editionsprojekten, den zu edierenden Dokumenten und der jeweiligen TEI-Kodierung geschuldet ist. Für häufig anzutreffende Kodierungsmuster stellt der *dse-static-cookiecutter* mit dem Stylesheet `partials/shared.xsl` aber einige XSL-Templates bereit, etwa für TEI-kodierte Tabellen (tei:table, tei:row, tei:cell) und Listen (tei:list, tei:item), für Absätze (tei:p) und Zeilenumbrüche (tei:lb) oder für Verweise (tei:ref), Anmerkungen (tei:note), unklare Lesarten (tei:unclear), Streichungen (tei:del).

[17] Ein eigenes Stylesheet `toc.xsl` iteriert über alle Dateien in `data/editions`, liest die Informationen aus dem TEI-Header aus und erstellt daraus ein Inhaltsverzeichnis der editierten Dokumente (`toc.html`). Dieses Inhaltsverzeichnis wird als Tabelle dargestellt. Dank der JavaScript-Bibliothek tabulator.info können Nutzer*innen die Sortierung der Tabelle ändern, die einzelnen Spalten durchsuchen, Spalten ein- und ausblenden und die Tabelle exportieren (HTML, JSON, CSV). Jede Zeile der Tabelle verlinkt auf die HTML-Seite des entsprechenden edierten Dokumentes. Da im Falle des *dse-static-cookiecutters* als die Datengrundlage für *tabulator.info* eine HTML-Tabelle (html:table) verwendet wird (möglich wäre auch eine JSON-Repräsentation der Daten), bleibt die Basisfunktion dieses Inhaltsverzeichnisses – Überblick zu den Inhalten der Editionseinheiten und Verlinkung zu diesen – aufrecht, selbst wenn zukünftige Browsergenerationen die verwendete *tabulator.info*-Version nicht mehr unterstützen sollten.

[18] Als Registerdaten werden standardmäßig unterstützt: Personen-, Orts-, Organisations- und Werkregister. Dafür ist das Verzeichnis `data/indices` vorgesehen. XSLT-Dateien (`listperson.xsl`, `listplace.xsl`, `listorg.xsl`, `listbibl.xsl`) sind für deren Konvertierung verantwortlich, wobei diese Dateien eine spezifische TEI-Kodierung der Registereinträge erwarten (tei:listPerson, tei:listPlace, tei:listOrg, tei:listBibl). Für Registereinträge werden tabellarische Listenansichten generiert, welche ebenfalls mit Hilfe von *tabulator.info* interaktiv sind. Das Ortsregister wird außerdem durch eine Kartenansicht ergänzt (implementiert mittels Leaflet), sofern die dafür notwendigen Geodaten von den Editor*innen erhoben wurden.

Abb. 2: Ortsregister mit Kartenansicht aus dem Editionsprojekt *Familiensache*. Angezeigt werden hier nur die in der Tabelle gefilterten Absendeorte. [Screenshot: Peter Andorfer 2025]

[19] Die Kartenansicht ist mit der Ortstabelle verknüpft, das heißt, nur die in der Tabelle angezeigten Einträge werden auf der Karte dargestellt, so dass sich der Kartenausschnitt ändert, wenn Filter in der Ortstabelle angewandt werden. Zusätzlich wird für jeden Registereintrag, also für jede Person, jeden Ort, jedes Werk eine eigene HTML-Seite generiert. Diese präsentiert neben den im Registereintrag verzeichneten Informationen wie etwa dem Namen der Person, Lebensdaten, Normdaten (z.B. GND, Wikidata) auch eine Auflistung sämtlicher Editionseinheiten, in welchen besagte Entität erwähnt wird. Entitäten von Editionsprojekten sind somit eindeutig über ihre projektspezifischen URLs referenzierbar, wodurch andere Projekte auf diese verlinken können. Die Datei `beacon.xsl` verwendet diese URLs, um eine GND-Beacon-Datei[14] zu erstellen.

4. Lösungsansätze für Einschränkungen statischer Webseiten

4.1 Register

[20] Um ein Register in der eben beschriebenen Art und Weise im Kontext einer statischen Webseite umsetzen zu können, müssen die dafür benötigten Daten in einem entsprechenden Format vorliegen. Ein Personenregister wird häufig – so zumindest bei Projekten im Umfeld des ACDH – wie folgt angelegt: In einer TEI-XML-Datei, z.B. `listperson.xml`, werden projektrelevante Informationen zu Personen kodiert, welche in den zu edierenden Dokumenten erwähnt werden. Jeder Registereintrag erhält eine projektspezifische,

[14] Vgl. Wikipedia: BEACON.

eindeutige Identifikationsnummer, die etwa als `xml:id` im jeweiligen tei:person-Element in der Registerdatei hinterlegt ist (`<person xml:id=person_0003”>...</person>`). In den einzelnen TEI-XML-Dokumenten der Editionseinheiten können dann Erwähnungen einer im Register erfassten Person nach einem Muster ähnlich wie: `heute habe ich die <rs type=”person” ref=”#person_0003”>Königin</rs> getroffen` codiert werden. Eine herkömmliche, dynamische Webapplikation kann mit einer derartigen Kodierung, basierend etwa auf einer XML-Datenbank wie *eXist-db*, mittels XQuery eine Abfrage nach dem Muster: ›Zeige mir alle Dokumente, in denen person_0003 erwähnt wird‹ formulieren und das Ergebnis im Kontext der Webapplikation ansprechend darstellen.

[21] In einem statischen Kontext ist dies nicht möglich, da es weder eine Datenbank für die XML-Dateien noch eine Laufzeitumgebung gibt, durch die spontane Abfragen ausgeführt werden können. Die Registerdaten müssen denormalisiert werden,[15] also vervielfacht. Dafür wird einerseits in der jeweiligen Liste der erwähnten Entitäten erfasst, welches Dokument eine Erwähnung aufweist. Weiters werden in jede Editionsdatei die Angaben zu den darin erwähnten Entitäten aus der Registerdatei kopiert. Der Vorteil einer solchen Datenstruktur liegt darin, dass nun das jeweilige edierte Dokument alle Informationen selbst enthält und nicht erst durch einen Abgleich mit anderen Dokumenten ›vollständig‹ ist. Die Geschlossenheit der Register-, vor allem auch der Editionsdateien vereinfacht deren Verarbeitung in einem statischen Kontext massiv. Ein bedeutsamer Nachteil ist die Duplizierung von Information, was generell größere Datenmengen bedeutet. Das ist in Hinblick auf die Synchronität der Informationen problematisch, da die gleiche Information in mehreren Dateien gespeichert ist. Dabei handelt es sich jedoch eher nur um ein theoretisches Problem. In der Praxis, sprich in den Projekten, die mit dem *dse-static-cookiecutter* realisiert werden, läuft die Denormalisierung automatisiert als ein Schritt des Build-Prozesses der statischen Seite. Editor*innen müssen demnach Registerinformationen nicht händisch kopieren und synchron halten. Dafür wird ein Kommandozeilenbefehl des am ACDH entwickelten Python Packages acdh-tei-pyutils ausgeführt. Hinsichtlich der Größe dieser denormalisierten Dateien sind bis dato noch keine Probleme aufgetreten. Hier sei auf das umfangreiche Projekt *Schnitzler-Tagebuch* verwiesen, welches über 16.000 Editionseinheiten umfasst, wobei jede Einheit einem Tagebucheintrag bzw. einer einzelnen TEI-XML-Datei entspricht. Das Personenregister verzeichnet ca 8.500 Personen[16] und die entsprechende TEI-XML-Serialisierung[17] ist 25 MB groß. Schwierigkeiten gibt es hier weder beim Prozessieren auf einem gängigen Rechner noch beim Speichern der Dateien auf GitHub (mit einem Limit von 100 MB Dateigröße).

4.2 Volltextsuche

[22] Der *dse-static-cookiecutter* hat in der aktuellen Version keine integrierte Volltextsuche. Das war nicht immer so. Bis zur Version 1.0 war die Volltextsuche durch staticSearch implementiert. *StaticSearch* wurde von Personen aus dem DH / TEI-Umfeld und für statische Webprojekte aus diesem Umfeld entwickelt. Darüber hinaus verwendet *staticSearch* für den Bau des Suchindexes den gleichen Technologie-Stack (*ANT*, *Saxon*, *Java*) wie der *dse-static-cookiecutter* für den Bau der Webseite. Im Unterschied zu anderen gängigen Suchmaschinen wie Solr, Elasticsearch, Algolia oder Typesense benötigt *staticSearch* keine Serverkomponente, sondern generiert einen dateibasierten Suchindex. Vereinfacht formuliert legt es für jeden Token im Textkorpus eine Datei an, welche den Kontext des Tokens (*KWIC*), die Namen der Dateien, in denen der Token zu finden ist, und verschiedene weitere Informationen abspeichert. Je nach Größe des Editionsprojektes und Variabilität des verwendeten Vokabulars kann ein solcher Suchindex mehrere hunderttausend Einzeldateien umfassen. Entsprechend lange kann auch das Bauen des Index dauern, von wenigen Sekunden bis zu Stunden. Allerdings hat dies bei der praktischen Anwendung von *staticSearch* im Kontext von *dse-static-cookiecutter* Projekten nie zu Problemen geführt, auch nicht zu Time-Outs bei GitHub-Actions[18].

[15] Vgl. Wikipedia: Normalisierung (Datenbank).
[16] Vgl. Austrian Center for Digital Humanities (Hg.) 2021: Personenregister.
[17] Vgl. Austrian Center for Digital Humanities (Hg.) 2021: Personenregister (XML).
[18] GitHub-Workflows in der kostenlosen Version brechen nach sechs Stunden ab, längere Laufzeiten sind nur gegen Bezahlung möglich.

[23] Dass *staticSearch* aus der Standardkonfiguration von *dse-static-cookiecutter* entfernt wurde, liegt daran, dass das ACDH mittlerweile einen eigenen *Typesense*-Server betreibt, welcher für die Volltextsuche der im Kontext des ACDH entwickelten digitalen Editionen verwendet wird. *Typesense* erweist sich einerseits aus Perspektive unserer Systemadministratoren als äußerst stabil und pflegeleicht, andererseits stellt *Typesense* gute Werkzeuge[19] zur Verfügung, welche es Entwickler*innen erlauben, mit geringem Aufwand Indices zu bauen und aktuell zu halten, sowie Suchinterfaces für diese Indices zu gestalten. Mit *Typesense* kann eine Volltextsuche im Kontext einer *dse-static-cookiecutter*-basierten digitalen Edition schneller und projektspezifischer als mit *staticSearch* implementiert werden. Ein Grund dafür ist, dass während mit *staticSearch* nur XHTML-Dateien indiziert werden konnten, *Typesense* für das Bauen des Index direkt auf die TEI-XML-Daten zurückgreifen kann. Es sei erwähnt, dass auch die Langfristigkeit bei Typesense (24.300 Stars, 51 Contributors)[20] besser gewährleistet zu sein scheint als beim ungleich kleineren *staticSearch*-Projekt (59 Stars, 4 Contributors).[21] Ungeachtet dessen, für welche Suche man sich entscheidet, sei festgehalten, dass auch bei einem Ausfall der Volltextsuche, egal ob statisch oder serverseitig implementiert, die restliche digitale Edition weiterhin funktioniert, sprich angesehen und referenziert werden kann. Im Catalogue of Digital Editions verfügen von rund 350 verzeichneten Editionen nur rund zwei Drittel über eine einfache Volltextsuche.[22]

4.3 Schnittstellen

[24] Wie man dem *Catalogue of Digital Editions* ebenfalls entnehmen kann, verfügen deutlich weniger digitale Editionen über eine Automatische Programmierschnittstelle (API) als über eine Volltextsuche. Der *Catalogue* gibt aber keinerlei Vorgaben, was unter API zu verstehen ist, sondern fragt lediglich, ob »the digital edition comes with an API«.[23] Die Auffindbarkeit solcher Schnittstellen ist in den einzelnen Editionen höchst unterschiedlich. Während einige Projekte APIs gut sichtbar in den Hauptmenüs verlinken,[24] findet man andere Schnittstellen, über die man auf die (TEI-XML-)Daten der Edition zugreifen kann, nur zufällig.[25] Ähnliches kann auch über die Ausgestaltung der APIs festgestellt werden. Hier reicht das Angebot an Schnittstellen von einfachen, statischen, aber wohldefinierten Austauschformaten wie dem bereits erwähnten BEACON-Format oder dem für Briefeditionen relevanten CMIF[26] über mehr oder weniger sichtbare Links zu den Quelldaten der Edition[27] oder standardisiert dokumentierten Schnittstellen bis hin zur Implementierung etablierter Protokolle wie beispielsweise OAI-PMH.[28] Zu erwähnen sind an dieser Stelle auch Initiativen, die das Ziel haben, spezielle API-Protokolle für digitale Editionen bzw. digitale Textsammlungen zu definieren.[29]

[25] Der *dse-static-cookiecutter* ist aufgrund seiner Anlage grundsätzlich auf die Auslieferung statischer Daten beschränkt. In der Standardkonfiguration umfasst das nicht nur die im Build-Prozess erzeugten HTML-Dateien, inklusive der bereits erwähnten BEACON-Datei, es werden auch sämtliche TEI-XML-Dateien aus dem ›data‹-Verzeichnis in das Webseitenverzeichnis kopiert und damit online zugänglich gemacht. Dies bedeutet, dass das TEI-XML-Dokument, welches der HTML-Seite https://schnitzler-tagebuch.acdh.oeaw.ac.at/entry__1900-02-21.html zugrunde liegt, unter der URL https://schnitzler-tagebuch.acdh.oeaw.ac.at/

[19] Clients in diversen gängigen Programmiersprachen wie Python, JavaScript, PHP oder Ruby (vgl. Typesense Inc. (Hg.) 2025) sowie JavaScript-Komponenten für Abfrageinterfaces (vgl. Typesense Inc. (Hg.) 2020–2026).
[20] Vgl. Typesense Inc. 2016–2026 (Stand: 17.09.2025).
[21] Vgl. Holmes / Takeda 2019 (Stand: 17.09.2025).
[22] Vgl. Andorfer / Franzini 2025a.
[23] Andorfer / Franzini 2025b: CONTRIBUTING.md.
[24] Vgl. Allroggen (Hg.) 2025; API Dokumentation; Hitchcock et al. (Hg.) 2023: Old Bailey API oder Müller et al. (Hg.) 2018–[2029]: Quelldaten und API.
[25] Vgl. Radecke (Hg.) 2026. Hier können Daten über die Standard-REST-Schnittstelle von *eXist-db* abgefragt werden, auf diese wird bei der Übersicht zu den Editionseinheiten aber nur dezent verlinkt.
[26] Vgl. Bernauer et al. (Hg.) 2018–2026: Daten und Schnittstellen.
[27] Vgl. Allroggen (Hg.) 2025: Swagger OpenAPI Documentation.
[28] Vgl. Ette (Hg.) 2025: OAI 2.0 Request Results.
[29] Vgl. Hegel 2025.

entry__1900-02-21.xml heruntergeladen werden kann. Was dem *dse-static-cookiecutter* aktuell noch fehlt, ist ein Skript, das im Kontext des Build-Prozesses eine maschinenlesbare Datei erstellt, welche Links zu allen TEI-XML-Dateien sämtlicher Editionseinheiten enthält.

Welche statischen Daten in einer *dse-static-cookiecutter*-basierten digitalen Edition ausgeliefert werden, liegt aber letztendlich am jeweiligen Projekt. Idealerweise sollte die Erstellung dieser Dateien, wie etwa eine CMIF-Datei im Falle von Briefeditionen,[30] Teil des Build-Prozesses und somit aktualisierbar und reproduzierbar sein. [26]

Seit Version 1.3 beinhaltet der *dse-static-cookiecutter* Skripte, welche aus den TEI-XML-Daten statische XML-Dateien generieren, die dem OAI-PMH-Standard entsprechen. Dabei lässt sich das OAI-PMH-Protokoll aber nicht zur Gänze im Kontext einer statischen Webseite umsetzen, unter anderem deshalb, weil dafür auch HTTP-POST-Anfragen unterstützt werden müssen. Um dieses weitverbreitete Protokoll dennoch bedienen zu können, läuft auf den Servern des ACDH eine einfache Applikation namens dse-static-oai-pmh, welche die statischen OAI-PMH-XML-Daten entsprechend den Vorgaben des OAI-PMH-Protokolls verarbeitet und ausliefert. Wie schon bei der Volltextsuche wird somit auch hier auf eine externe und dynamische Serverkomponente zurückgegriffen, um die Limitierung der statischen Seite zu entschärfen. Allerdings kann ein einziger *dse-static-oai-pmh*-Server, wie auch ein einziger *Typesense*-Server, viele einzelne Editionsprojekte bedienen. Das hält den Wartungsaufwand beider Services in Grenzen. [27]

4.4 Faksimiles

Neben der Suche und der API-Integration gibt es eine dritte Funktion, die an einen spezifischen Service ausgelagert wird, der von vielen digitalen Editionen konsumiert wird: die Einbindung von Faksimiles. Sofern Faksimiles der edierten Dokumente vorhanden sind (und diese angezeigt werden dürfen), werden diese meist von externen Quellen wie beispielsweise von ARCHE (dem Forschungsdatenrepositorium des ACDH), von einem am ACDH gehosteten IIIF-Server, von den Servern von Transkribus oder von anderen Bibliotheken und Forschungseinrichtungen geladen. Im Code des *dse-static-cookiecutters* ist die Javascript-Bibliothek *OpenSeadragon* inkludiert, welche für die Darstellung der Bilder auf den Editionswebseiten verantwortlich ist und den gängigen IIIF-Standard unterstützt. [28]

Natürlich wäre es auch möglich, die Bilder gemeinsam mit der Webseite auszuliefern; sofern man aber die kostenlosen Services von GitHub (Code-Repo) in Anspruch nehmen möchte, sollte man allfällige Beschränkungen hinsichtlich der Dateigrößen beachten. [29]

4.5 Versionierung

Keine mitgelieferten Lösungen bietet der *dse-static-cookiecutter* für den Themenkomplex maschineller Versionierung,[31] im Unterschied etwa zur Darstellung von im revisionDesc-Element manuell protokollierten Änderungen im TEI-Dokument, welche in der HTML-Version natürlich sehr wohl angezeigt werden können. Versionierung der Daten wie auch des Codes läuft im Kontext eines Git-basierten Arbeitsumfeldes im Hintergrund mit. Nicht versioniert werden standardmäßig die im Build-Prozess generierten HTML-Dateien, also die Webseite der digitalen Edition. Dies wäre zwar technisch einfach möglich, allerdings liefern GitHub-Pages, wie auch der im Dockerfile des *dse-static-cookiecutters* konfigurierte Webserver, nur die jeweils letzte Version dieser Dateien aus. Um unterschiedliche Versionen der Webseite bzw. die »Versionierung des Gesamtsystems«[32] der digitalen Edition für Nutzer*innen zitierbar zu gestalten, bräuchte es ein Setup, welches nicht nur alle Versionen (von Code und Daten) über eindeutig zitierbare URLs über einen [30]

[30] Vgl. Keller et al. (Bearb.) 2024: cmif.xml.
[31] Vgl. Bürgermeister 2023.
[32] Bürgermeister 2023, S. 136.

Webserver zurückgibt, sondern auch ein System, das Nutzer*innen darauf hinweist, welche Version gerade betrachtet wird und wie man von einer Version auf die nächste, vorige oder aktuellste Version kommt. Ein solches System ließe sich vermutlich in einem statischen Kontext mit Hilfe von Commit-Hashes in Pfadnamen und einer spezifischen Konfiguration des Webservers aufsetzen. Allerdings hat bis dato noch kein Projektpartner die Frage nach Versionierung der HTML-Dateien im Sinne des oben erwähnten ›Gesamtsystems‹ aufgeworfen.

4.6 FAIR data

[31] Abseits des Problems der Versionierung erfüllen *dse-static-coociekutter*-basierte digitale Editionen die technischen FAIR-Kriterien.[33] Dies trifft insbesondere auf das Kriterium »F1: (Meta) data are assigned globally unique and persistent identifiers«[34] zu. Wie erwähnt sind sowohl die einzelnen XML-Dateien als auch die daraus resultierenden HTML-Ansichten über URLs adressierbar. Und im Gegensatz zu dynamischen Applikationen sind diese URLs nicht an die darunterliegende Technologie gekoppelt. Die Verwendung von TEI entspricht ebenfalls den FAIR-Kriterien; wie detailliert die notwendigen Informationen darin aber kodiert werden, liegt an den Einzelprojekten. Sehr wohl wird bei der laufenden Entwicklung des *dse-static-cookiecutters* aber auf Accessibility im Sinne von Barrierefreiheit, geachtet. So erfüllen die mit den standardmäßig ausgelieferten XSL-Templates generierten HTML-Seiten den WCAG 2.1 AA-Standard.

5. Abschluss und Ausblick

[32] Die vom *dse-static-cookiecutter* implementierten Prinzipien ›statisch‹, ›automatisch‹ und ›ähnlich‹ sind weder neu noch originell. Static-Site-Generatoren gibt es schon lange – die erste Version (v1.0.0) von Jekyll wurde am 14. Juni 2013 veröffentlicht – und neue Frameworks zum Bau statischer Webseiten, wie beispielsweise eleventy oder Astro – werden laufend entwickelt.[35] Statische Seiten finden auch in den digitalen Geisteswissenschaften ihre Verwendung, häufig im Kontext des Begriffs *minimal computing*, wie in: »Minimal computing is static site generation«.[36] So handelt es sich beim Webauftritt von DHQ: Digital Humanities Quarterly um eine mittels XSLT und ANT generierte statische Seite, welche mit Hilfe von GitHub-Actions gebaut und veröffentlicht wird (und ihre Volltextsuche mit *staticSearch* implementiert).[37] Ähnlich funktioniert auch der Webauftritt von The Programming Historian, der auf *Jekyll* basiert.[38] Und auch digitale Editionen werden abseits *dse-static-cookiecutter*-basierter Projekte in Form von statischen Seiten realisiert.[39] Es bestehen auch Pläne, bestehende dynamische Editionen im Sinne einer besseren Langlebigkeit in statische Seiten zu konvertieren.[40]

[33] Dass sich Projekte effizienter realisieren und warten lassen, wenn sie auf einen gemeinsamen respektive einen ähnlichen Quellcode zugreifen und den gleichen Technologie-Stack verwenden, dürfte offensichtlich sein. Dies trifft auch auf digitale Editionen zu. Dafür gibt es eine Reihe von Tools, welche sich zum Ziel gesetzt haben, die Erstellung und Publikation von TEI-XML-basierten digitalen Editionen zu unterstützen. Solche Tools reichen von generischen XSLT-Stylesheets zur Konvertierung von TEI-XML-Dateien nach HTML[41] oder der JavaScript-Bibliothek CETEIcean für die Darstellung TEI-XML-Dateien im Webbrowser über out-of-

[33] Go Fair 2026: FAIR Principles. Vgl. für eine Analyse digitaler Editionen vor dem Raster der FAIR-Prinzipien Dängeli / Stuber 2020.
[34] Go Fair 2026: F1: (Meta) data are assigned globally unique and persistent identifiers.
[35] Am ACDH wurde seit etwa 2024 damit begonnen, bestehende Wordpress-Instanzen, die vornehmlich als Projekt- oder Konferenzwebseiten verwendet wurden, durch *Astro*-Webseiten zu ersetzen. Das Ziel ist, den Maintenance-Aufwand stark zu reduzieren.
[36] Risam / Gil 2022, S. 1.
[37] Vgl. Cummings 2023.
[38] Vgl. Lincoln et al. 2022.
[39] Vgl. etwa Akademie der Wissenschaften und der Literatur Mainz (Hg.) 2025; Trautmann / Schrade (Hg.) 2018; Grallert (Hg.) 2022; Dumont (Hg.) 2025.
[40] Vgl. Jüngerkes / Kruse 2024.
[41] Vgl. TEI-Boilerplate oder TEIC/Stylesheets bzw. den entsprechenden Webservice teigarage.

the-box Lösungen für spezifische Editionstypen wie dem EVT-Viewer oder dem Edition-Crafter bis hin zu vielfach verwendeten Komplettlösungen wie etwa dem TEI Publisher.[42] Bei letzterem handelt es sich zwar um eine auf *eXist-db* basierte, dynamische Applikation, allerdings wurde daran gearbeitet, statische Lösungen zu implementieren.[43] Hinzu kommen die vielen Frameworks, Setups und Systeme, die an spezialisierten Institutionen wie Bibliotheken, Universitäten oder Akademien gepflegt werden.[44] Eine aktuelle Übersicht wäre sicherlich sehr gewinnbringend.[45]

[34] Der *dse-static-cookiecutter* versucht den Spagat zwischen einem spezifisch auf die Bedürfnisse des ACDH zugeschnitten Werkzeug und einem allgemein zugänglichen Tool zum Entwickeln und Publizieren von digitalen Editionen. Der spezifische Gebrauch am ACDH zeigt sich etwa in der im Initialisierungsprozess einer neuen Instanz gestellten Frage nach einer ›Redmine-ID‹ (welche anschließend benötigt wird, um eine aktuelle Impressum-Seite zu generieren) oder in der fehlenden Unterstützung von *staticSearch*. Andererseits werden technologische und konzeptionelle Entscheidungen immer mit Blick darauf getroffen, einen technologisch möglichst niederschwelligen Einstieg zu ermöglichen. Während die bereits erwähnten Alternativen zum *dse-static-cookiecutter* sehr stark auf JavaScript setzen, baut der *dse-static-cookiecutter* primär auf XSLT-Transformationen zur Generierung der HTML-Seiten. Grundkenntnisse in X-Technologien sind in den digital affinen Geisteswissenschaften häufig vorhanden.[46] Dies trifft auch auf die starke Integration des *dse-static-cookiecutters* mit GitHub und vor allem GitHub-Pages zu. Das ACDH hätte zwar eigene Ressourcen und Infrastruktur, womit sich der Build- und Deploy-Prozess individueller und flexibler gestalten ließe,[47] setzt aber bewusst auf GitHub und GitHub Pages, um auch jene Projekte zu unterstützen, die nicht auf eine vergleichbare Infrastruktur zurückgreifen können. Das Nicht-Unterstützen aktueller JavaScript-Tools wie etwa eines Package-Managers[48] zur Installierung der verwendeten Bibliotheken oder eines Build-Tool wie Vite für die Entwicklung und Implementierung projektspezifischer interaktiver Elemente ist ebenfalls eine bewusst getroffene Entscheidung mit dem Zweck, den Zugang zum *dse-static-cookiecutter* einfach zu gestalten.

[35] Ob Editionen, die mit dem *dse-static-cookiecutter* realisiert wurden und werden, das eingangs gegebene Versprechen der Langfristigkeit einlösen können, wird die Zukunft zeigen. Die bisherigen Erfahrungen stimmen allerdings positiv. Was die weitere Entwicklung des *dse-static-cookiecutters* betrifft, so stehen aktuell die Bereiche Schnittstellen und Zitierbarkeit im Vordergrund. Darüber hinaus wird am ACDH gerade an einem zentralen Triplestore unter anderem für Entitäten aus digitalen Editionen[49] und an einem linguistischen Suchservice[50] zu automatisch linguistisch annotierten Volltexten digitaler Editionen gearbeitet, die grundsätzlich auch für Projekte außerhalb des ACDH offenstehen sollen.

[42] Vgl. Sutor / Mertgens 2024.
[43] Vgl. Meier 2022.
[44] Vgl. Andorfer et al. 2017.
[45] Vgl. dazu etwa die Sondergausgaben von RIDE zu Tools and Environments.
[46] Vgl. etwa den im Rahmen der TEI-Konferenz 2025 angebotenen Workshop: ›Navigating and Processing Data from the TEI with XPath and XSLT‹ oder die im Curriculum für das Masterstudium Digitale Geisteswissenschaften der Universität Graz vorgesehenen Kurse ›X-Technologien I und II‹ (Universität Graz (Hg.) 2021).
[47] Gedacht ist hier etwa an die Möglichkeit, sogenannte Feature-Branches zu deployen, oder an bessere Konfigurationsmöglichkeiten des Webservers.
[48] Ein Beispiel hierfür wäre NPM.
[49] Dies findet im Rahmen des Projekts PFP – Prosopographische Forschungsplattform Österreich, Laufzeit: 2024–2026, statt.
[50] Austrian Center for Digital Humanities (Hg.) 2025.

Bibliografie

Akademie der Wissenschaften und der Literatur Mainz (Hg.): Burchards Dekret Digital. 2025. HTML. [online]

Gerhard Allroggen (Hg.): Carl-Maria-von-Weber-Gesamtausgabe. Digitale Edition. Version 4.13.1 vom 12.12.2025. HTML. [online]

Peter Andorfer / Daniel Elsner / Carl Friedrich Haak / Martin Anton Müller / Kinga Sramó / Stefan Probst / Timo Frühwirth: acdh-oeaw/dse-static-cookiecutter. Zenodo. 02.12.2024. Version v1.10 vom 05.12.2025. DOI: 10.5281/zenodo.17831445.

Peter Andorfer / Greta Franzini (2025a): Catalogue Digital Editions. 2025. HTML. [online]

Peter Andorfer / Greta Franzini (2025b): dig-ed-cat-static. GitHub. 2025. Version v1.1 vom 08.10.2015. [online]

Peter Andorfer / Matej Durco / Thomas Stäcker / Christian Thomas / Vera Hildenbrandt / Hubert Stigler / Sibylle Söring / Lukas Rosenthaler: Nachhaltigkeit technischer Lösungen für digitale Editionen. Eine kritische Evaluation bestehender Frameworks und Workflows von und für Praktiker_innen. In: Elisabeth Burr (Hg.): DHd 2016. Modellierung – Vernetzung – Visualisierung. Die Digital Humanities als fächerübergreifendes Forschungsparadigma. 3. Jahrestagung des Verbands Digital Humanities im deutschsprachigen Raum. Konferenzabstracts (Leipzig, 07.–12.03.2016). 2., überarbeitete und erweiterte Ausgabe. Leipzig 2017, S. 55–58. PDF. DOI: 10.5281/zenodo.3679331

Austrian Centre for Digital Humanities (Hg.): Arthur Schnitzler Tagebuch 1879–1931 digital. Wien 2021. HTML. [online]

Austrian Centre for Digital Humanities (Hg.): corpus-search. GitHub. 2025. [online]

BEACON. In: Wikipedia. Die freie Enzyklopädie. Letzte Aktualisierung: 11.02.2026. [online]

Markus Bernauer / Norbert Miller / Frederike Neuber (Hg.): Jean Paul – Sämtliche Briefe digital. 2018–2026. HTML. [online]

Martina Bürgermeister: Versionierung von digitalen Editionen in der Praxis. In: Roman Bleier / Helmut Klug W. (Hg.): Digitale Edition in Österreich (= Schriften des Instituts für Dokumentologie und Editorik, 16). Norderstedt 2023, S. 133–150. PDF. URN: urn:nbn:de:hbz:38-704525

James Cummings: Academics Retire and Servers Die: Adventures in the Hosting and Storage of Digital Humanities Projects. In: Digital Humanities Quarterly 17 (2023), H. 1. HTML. [online]

Peter Dängeli / Martin Stuber: Nachhaltigkeit in langjährigen Erschliessungsprojekten. In: xviii.ch – Schweizerische Zeitschrift für die Erforschung des 18. Jahrhunderts 11 (2020), H. 11. PDF. DOI: 10.24894/2673-4419.00004

Matthew James Driscoll / Elena Pierazzo (Hg.): Digital Scholarly Editing. Theories and Practices. Cambridge, UK 2016. PDF. DOI: 10.11647/OBP.0095

Stefan Dumont (Hg.): Julius & Barbara. Ein rheinischer Familienbriefwechsel aus der Mitte des 19. Jahrhunderts. Berlin 2025. HTML. [online]

Ottmar Ette (Hg.): edition humboldt digital. Version 11 vom 04.06.2025. HTML. [online]

GO FAIR. Letzter Zugriff: 16.01.2026. HTML. [online]

Till Grallert (Hg.): Open Arabic Periodical Editions. A Framework for Bootstrapped Digital Scholarly Editions outside the Global North. 2022. HTML. [online]

Philipp Hegel: APIs for Text-Based Editions. In: Digital Humanities im deutschsprachigen Raum (Hg.): DHd-Blog. 28.08.2025. HTML. [online]

Patrick Helling: To whom it may Concern – Strategies for Ensuring Sustainable Management of Digital Scholarly Editions. Präsentation zur Tagung »404 not found« Approaches to Sustainable Editions (Oslo, 09.04.2024). Zenodo. 24.10.2024. PDF. DOI: 10.5281/zenodo.13985876

Tim Hitchcock / Robert Shoemaker / Clive Emsley / Sharon Howard / Jamie McLaughlin (Hg.): The Proceedings of the Old Bailey, 1674–1913. Version 9.0 vom Herbst 2023. HTML. [online]

Martin Holmes / Joey Takeda: staticSearch. GitHub. 2019. Version v2.0.0 vom 24.01.2026. [online]

Internationalisierung (Softwareentwicklung). In: Wikipedia. Die freie Enzyklopädie. Letzte Aktualisierung: 19.02.2026. [online]

Sven Jüngerkes / Maximilian Kruse: Das Editionsprogramm ›Fraktionen im Deutschen Bundestag 1949–2005‹. In: Zeitschrift für digitale Geisteswissenschaften 9 (2024). 21.11.2024. HTML. DOI: 10.17175/2024_007

Katrin Keller / Ines Peper / Dorota Vargová / Anna Spitzbart (Bearb.): Die Korrespondenz der Kaiserin Eleonora Magdalena (1655–1720) von Pfalz-Neuburg. Wien 2024. HTML. [online]

Matthew Lincoln / Jennifer Isasi / Sarah Melton / François Dominic Laramée: Relocating Complexity: The Programming Historian and Multilingual Static Site Generation. In: Digital Humanities Quarterly 16 (2022), H. 2. HTML. [online]

Wolfgang Meier: Generating a High Voltage Site by Going Static. 2022. [online]

Martin Anton Müller / Gerd-Hermann Susen / Laura Untner / Selma Jahnke (Hg.): Arthur Schnitzler. Briefwechsel mit Autorinnen und Autoren. 1885–1931. Wien 2018–[2029]. HTML. [online]

Normalisierung (Datenbank). In: Wikipedia. Die freie Enzyklopädie. Letzte Aktualisierung: 09.12.2025. [online]

Gabriele Radecke (Hg.): Theodor Fontane: Notizbücher. Digitale genetisch-kritische und kommentierte Edition. Letzter Zugriff: 12.02.2026. HTML. [online]

Roopika Risam / Alex Gil: Introduction: The Questions of Minimal Computing. In: Digital Humanities Quarterly 16 (2022), H. 2. HTML. [online]

Patrick Sahle: What is a Scholarly Digital Edition? In: Matthew James Driscoll / Elena Pierazzo (Hg.): Digital Scholarly Editing. Theories and Practices. Cambridge, UK 2016, S. 19–40. HTML / PDF. DOI: 10.11647/OBP.0095.02

Patrick Sahle: Kriterien für die Besprechung digitaler Editionen. Version 1.1. 2014. HTML. [online]

Statische Webseite. In: Wikipedia. Die freie Enzyklopädie. Letzte Aktualisierung: 17.12.2025. [online]

Nadine Sutor / Andreas Mertgens: TEI Publisher. In: RIDE – A review journal for digital editions and resources 19 (2024). HTML. DOI: 10.18716/ride.a.19.1

Marjam Trautmann / Torsten Schrade (Hg.): DER STURM. Digitale Quellenedition zur Geschichte der internationalen Avantgarde. Mainz 2018. HTML. [online]

Typesense Inc. (Hg.): Typesense Guide. Installing a Client. Letzte Aktualisierung: 06.11.2025. HTML. [online]

Typesense Inc. (Hg.): Typesense Instantsearch Adapter. GitHub. 2020–2026. Version v2.9.0 vom 02.04.2025. [online]

Typesense Inc. (Hg.): Typesense. GitHub. 2016–2026. Version 30.1 vom 29.01.2026. [online]

Universität Graz (Hg.): Curriculum für das Masterstudium Digitale Geisteswissenschaften. Digital Humanities. Curriculum 2021 (= Mitteilungsblatt der Karl-Franzens-Universität Graz, Sondernr. 85). Graz 2021. PDF. [online]

Abbildungsverzeichnis