

Allgemeine Beschreibung

ediarum ist eine seit 2012 entwickelte digitale Arbeits- und Publikationsumgebung, die aus mehreren Softwarekomponenten besteht und im Grunde ein Werkzeugkasten aus verschiedenen auf TEI-Standard basierenden Modulen ist. Damit bietet *ediarum* eine Schnittstelle zwischen Editions-umgebung, XML-Datenbank und Rechercheportal, wobei die Kernfähigkeit jedenfalls in der Aufbereitung von XML-Dateien besteht. Die digitale Arbeitsumgebung basiert auf einer eXist-db und ermöglicht nicht nur die Transkription von Manuskripten und Drucken, sondern vor allem auch die TEI-konforme Annotation und Erstellung von Text- und Sachapparaten sowie Registern.

ediarum ist als ein Add-On für Oxygen konzipiert und seit 2015 gibt es eine generalisierte Eingabeoberfläche. Zu beachten ist, dass *ediarum* keine Plug-and-Play-Software ist, da immer ein:e DH-Entwickler:in für die Implementierung und den Betrieb von *ediarum* nötig ist. In laufenden Projekten werden nicht immer alle Module, sondern mitunter auch nur Frameworks eingesetzt.

Der größte Vorteil von *ediarum* liegt darin, dass Transkripte sehr benutzerfreundlich mit TEI-konformem XML in einer gut individualisierbaren Editions-umgebung ausgezeichnet werden können. Die Einrichtung von projektspezifischen Bearbeitungsmöglichkeiten bzw. Buttons erfordert aber jedenfalls Programmierkenntnisse oder die Betreuung durch eine:n DH-Entwickler:in. Die Suchmöglichkeiten im Autormodus des Register-Moduls sind außerdem leider auf wortwörtliche Eingaben beschränkt, die von den Editionsmitarbeitenden eine genaue Kenntnis der Daten verlangen.

→ Hinweis: Die nachfolgenden Ausführungen beschränken sich auf die Module *ediarum.BASE.edit* und *ediarum.REGISTER.edit*.

Anwendungsbereiche

- Transkription von Manuskripten und Drucken
- Text-Mark-up
- TEI-konforme Annotation
- Indexierung
- Erstellung von Text- und Sachapparaten sowie Registern
- Publikation in Web und Druck

Funktionsübersicht

- Kollaborationsmöglichkeit
- Möglichkeiten der Qualitätssicherung
- Module innerhalb unterschiedlicher Umgebungen (hier nur auszugsweise):
 - *eXist-db*:
 - **ediarum.DB** zur Konfiguration einer eXist-Datenbank - die Daten, die von beliebigen Webtechnologien genutzt werden können, werden dabei über eine JSON-API geliefert
 - **ediarum.WEB** als Library, die entsprechende Funktionalitäten für die Erstellung einer WebApp für ein Rechercheportal bereitstellt
 - *Oxygen XML-Editor*:
 - **ediarum.BASE.edit** beinhaltet nützliche Funktionen für die Bearbeitung von XML-Dateien wie zum Beispiel Stylesheets für den Autor-Modus, die eine benutzerfreundlichere Transkription (von Formatierungen wie Unterstreichungen, Hervorhebungen, Streichungen, Leserlichkeit, etc. oder Markierung fremdsprachlicher Einträge) mit Schaltflächen im Oxygen-Editor ermöglichen

- **ediarum.REGISTER.edit** beinhaltet nützliche Funktionen für die erleichterte Auszeichnung von Personen, Orten etc.
- **ediarum.PDF** für das Herunterladen und Ausdrucken von TEI-XML-kodierten Texteditionen (in Entwicklung)
- **ediarum.MEDIAEVUM** für die Edition mittelalterlicher (Prosa-)Texte
- Diverse Ansichtsmöglichkeiten: Web-Ansicht und PDF-Vorschau
- Nutzung des DTA-Basisformats als Schema (ODD/RNG)

Voraussetzungen

Jedes Tool kann einerseits bestimmte Vorkenntnisse der Benutzer:innen voraussetzen und andererseits auch hinsichtlich der Software-Umgebung gewisse Anforderungen stellen.




Erforderliche Kenntnisse

- [EDV-Grundkenntnisse](#)
- TEI-XML
- XPath/XQuery
- HTML/CSS von Vorteil
- Grundverständnis von GitHub

Benötigte Software

- Oxygen XML Author (kommerziell)
- eXist-db (optional - nur bei kollaborativem Arbeiten und für Registerverknüpfungen; Workarounds sind grundsätzlich möglich)
- Docker Desktop (für eine weniger fehleranfällige Installation von eXist-db)

Tool-Kompatibilität

	IIIF	Transkribus	FromThePage	FairCopy	OpenRefine	ba[sic?]	teiPublisher	ediarum.WEB
ediarum	✗	✗		✗		✗		✓

Kostenübersicht

- **ediarum & eXist-db:**
 - kostenlos
- **Oxygen XML Author:**
 - Halbjahres-Abo: \$190 - 244
 - Jahres-Abo: \$335 - 432

[Detaillierte Preisübersicht \(Oxygen\)](#)

Möglichkeiten & Grenzen

Da jedes Projekt unterschiedliche Anforderungen mit sich bringt, sollen nachfolgend mögliche Vor- und Nachteile des Tools aufgelistet werden, die während der Durchführung des jeweiligen [Beispielprojekts](#) festgestellt wurden.

Stärken

- Nutzung von Oxygen als komfortable und stabile Arbeitsumgebung mit ergiebiger Dokumentation
- Projektmitarbeitende können die Edition in einer benutzerfreundlichen "Autoransicht" bearbeiten und über eine eigene Werkzeugleiste per Mausklick Auszeichnungen vornehmen
- Bearbeitungsansicht kann für den jeweiligen Arbeitsschritt angepasst werden
- Validierung von Dokumenten gegen das eingebundene Schema sowie Ausgabe von entsprechenden Fehlermeldungen direkt während Bearbeitung
- Ständige Weiterentwicklung durch [BBAW](#)
- Einfache Nachnutzbarkeit von Funktionalitäten und Designs
- Möglichkeit Daten-Backups einzustellen (dabei sind jedoch Speicherkapazitäten zu beachten)
- [Zotero](#)-Integration möglich

Herausforderungen & Probleme

- Implementierung und projektspezifische Konfiguration erfordert DH-Entwickler:in
- Abhängigkeit von eXist-db und Oxygen
- Datenbank-Technologie ist bei größeren Datenmengen noch suboptimal, da eXist-db viel Arbeitsspeicher benötigt - für eine stabile Funktion der Datenbank sollten zumindest 2 GB Arbeitsspeicher zur Verfügung stehen
- Metadaten-Anreicherung bzw. -Ergänzung nur teilweise im benutzerfreundlicheren Autormodus möglich
- Suche im Register bei der Annotation bietet keine Substring-Suchmöglichkeit und auch keine Suche nach alternativen Bezeichnungen eines Eintrags - Bezeichnungen der Registereinträge (z. B. Flaschenkürbis, Gewürznelke, Echter Pfeffer) werden nicht gefunden, wenn man Teilbezeichnungen sucht (z. B. Kürbis, Nelke, Pfeffer)

Einrichtung & Erste Schritte

Anhand eines [Beispielprojekts](#), das zum Ziel hat, Kochrezepte aus dem Mittelalter computergestützt zu analysieren und später über eine Forschungsplattform zur Verfügung zu stellen, soll nachfolgend ein möglicher Arbeitsablauf für die Annotation mit *ediarum* beschrieben werden. Die dafür verwendeten Daten wurden bereits mit dem Tool [FromThePage](#) transkribiert und daraufhin für die weitere Bearbeitung vorbereitet (siehe [Transition](#)). In dieser Kurzanleitung soll nun mit *ediarum* die weitere Annotation und Indexierung der Kochrezepte des Beispielprojektes erfolgen und beschrieben werden.

1. Installation einzelner Komponenten

- **Installation von eXist-db:** Um die Abläufe für ein kollaborativ angelegtes Projekt genauer zu betrachten, soll nachfolgend zuerst das Einrichten einer XML-Datenbank beschrieben werden. Dies dient dazu, die bereits erstellten Transkriptionen für die weitere Annotation zu speichern und für die gemeinsame Bearbeitung bereitzustellen.
 - Hier geht es zur detaillierten [Installationsanleitung für Docker Desktop und eXist-db](#).
- **Installation von ediarum.DB in eXist-db:** Um später unser eigenes Projekt generieren zu können, müssen wir nun *ediarum* in eXist einrichten.
 - Dafür laden wir aus dem [ediarum-Github-Repository](#) die aktuellste Version von ediarum.DB als XAR-Datei herunter. {% include image.html url="../../../data/pipelines/pipeline_1/ediarum/img/github-ediarum.PNG" description="Download von ediarum auf Github" %}
 - Im eXist-Dashboard können wir nun über den Package Manager die XAR-Datei hochladen. {% include image.html url="../../../data/pipelines/pipeline_1/ediarum/img/ediarum-upload.PNG" description="Upload von ediarum im Package Manager von eXist" %} → Nach erfolgreichem Upload finden wir nun eine eigene Kachel mit der *ediarum*-App, die ab sofort außerdem auch im Dashboard über einen Link

verfügbar ist. {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/ediarum-app.PNG" description="ediarum-Applikation in eXist" %}

- **Installation des Oxygen XML-Editors:** Abschließend fehlt nur mehr der Download und die Installation des [Oxygen XML Author](#) - und zwar mindestens in Version 20.1. Alternativ kann auch der Oxygen XML Editor installiert bzw. genutzt werden.

2. Einrichtung des Projekts

- Zuerst wollen wir in unserer eXist-Datenbank für unsere Edition ein Projekt mit standardisierter Ablagestruktur sowie bestimmten Settings einrichten. Damit diese Standards korrekt angelegt werden, nutzen wir die ediarum.DB-App.
- Wir klicken also zunächst auf "ediarum" in unserem eXist-Dashboard. {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/ediarum-dashboard.PNG" description="ediarum.DB in eXist" %}
- Die *ediarum*-App öffnet sich nun in einem weiteren Fenster im Browser, wo wir im Menü zu **Verwaltung > Projekte** navigieren und dort im Feld "Neues Projekt" einen Projektnamen für unsere Edition eingeben und abschließend auf "Anlegen" klicken. {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/ediarum-project-setup.PNG" description="Einrichten eines Projektes in der ediarum-App" %} → Es dürfen keine Umlaute, Leer- oder Sonderzeichen verwendet werden.
- Außerdem möchten wir für unser Projekt in der Datenbank noch ein *ediarum*-Register aktivieren. In der geöffneten *ediarum*-App wählen wir dafür wieder unser Projekt aus und gelangen so auf die Projekt-Übersichtsseite. {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/ediarum-register.PNG" description="Projektkonfiguration in der ediarum-App" %} Dort legen wir ein neues Register an, indem wir neben **Neues Register** auf "Aktivieren" klicken. Für unser Beispielprojekt wählen wir hier "Sachbegriffe" aus dem Drop-Down-Menü des Ediarum-Registers und wählen bei Registeraufbau die Option "Register in einer Datei". {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/ediarum-register-setup.PNG" description="Einrichten eines projektspezifischen Registers" %}
- Wir können nun als Admin mit dem admin-Benutzernamen (ohne Passwort-Eingabe) eine Verbindung unseres Oxygen-Editors zur eXist-db herstellen. Für unsere Projektmitarbeitenden wollen wir aber noch weitere Benutzer:innen in der *ediarum*-App anlegen. Dafür gehen wir wieder auf die Übersichtsseite unseres Projekts und wählen in dem Menüpunkt **Projektkonfiguration > Benutzer** aus. Dort haben wir bereits *zimedgedtnt* als User angelegt und fügen jetzt noch *zim-student* als weiteres Projektmitglied hinzu. {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/ediarum-user.PNG" description="Anlegen von Projektmitgliedern in der ediarum-App" %} → Damit auch andere Benutzer:innen bzw. Projektmitarbeitende die Register nutzen können, müssen wir im User Manager der eXist-db entsprechende Zugriffsrechte vergeben. Dafür gehen wir zum eXist-db-Dashboard, wählen dort **User Manager** und klicken jenen Benutzernamen an, dem wir Zugriffsrechte einräumen wollen. {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/user-manager.PNG" description="User-Manager in der eXist-db" %} Am Ende der User-Ansicht gibt es ein Gruppenverwaltungssystem, wo wir unserem/unserer Benutzer:in auch Zugriff auf die Gruppe "oxygen" gewähren sollten, wenn wir unseren Projektmitarbeitenden Zugriff auf die erstellten Projekt-Register einräumen wollen. {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/user-permissions.PNG" description="Einstellung der Zugriffsrechte für Benutzer:innen" %} Damit hat der/die ausgewählte Projektmitarbeitende nun ebenfalls Zugriff auf das zuvor angelegte Register und kann dieses bei der Annotation nutzen.
- Im nächsten Schritt stellen wir im Oxygen XML Author eine Verbindung zu unserem soeben eingerichteten Projekt her, um alle in *ediarum* enthaltenen Funktionalitäten verfügbar zu machen. Wir öffnen dafür den Oxygen XML Author und wählen im Menü **Optionen > Einstellungen**, um anschließend in der linken Spalte "Datenquellen" anzuklicken und unter der Tabelle zu den Verbindungen auf das Plus zu klicken. Dort geben wir nun unsere Projektdaten an. {% include image.html

- url="../data/pipelines/pipeline_1/ediarum/img/oxygen-author.PNG" description="Erstellen einer Datenverbindung zu unserer ediarum-Datenbank" %} Ob unsere Datenbankverbindung erfolgreich war, können wir herausfinden, indem wir im Menü **Fenster > Ansicht zeigen > Datenquellen Explorer** auswählen. Dort sollte unser Projekt mit der darin angelegten Ordnerstruktur aufscheinen. {% include image.html url="../data/pipelines/pipeline_1/ediarum/img/oxygen-dataexplorer.PNG" description="Erfolgreiche Datenbank-Verbindung im Oxygen XML Author" %} Den Ordner mit den Briefen und dem Briefbeispiel.xml können wir schließlich löschen. Für unser Projekt legen wir stattdessen einen neuen Ordner mit der Bezeichnung "Manusripte" an, indem wir nach einem Rechtsmausklick auf unser Projekt im Datenquellen-Explorer "Verzeichnis erstellen" wählen. In diesen Ordner importieren wir nun unsere transformierten und auf das [DTABf angepasste XML-Dokumente](#). {% include image.html url="../data/pipelines/pipeline_1/ediarum/img/dataexplorer-manuscripts.PNG" description="Projektspezifischen Ordner erstellen und Dateien importieren" %} Wenn wir nun alle Verzeichnisse ausklappen, sieht die Ordnerstruktur und der Ordnerinhalt folgendermaßen aus: {% include image.html url="../data/pipelines/pipeline_1/ediarum/img/dataexplorer-new-project.PNG" description="Projektspezifische Datenstruktur" %}
- Als nächstes richten wir uns nun noch die grundlegenden *ediarum*-Frameworks in unserer Arbeitsumgebung ein. Dafür legen wir zuerst einen Ordner für diese Frameworks auf unserem lokalen Dateisystem an und legen dort die GitHub-Repositorien [ediarum.BASE.edit](#) und [ediarum.REGISTER.edit](#) ab, indem wir jeweils die ZIP-Dateien mit dem Source Code des letzten Releases herunterladen und diese dann entpacken. Da wir später auch ein editionsspezifisches Framework für unsere eigene Edition benötigen, legen wir zusätzlich noch einen projektspezifischen Ordner zur Erweiterung des Basis-Frameworks an. {% include image.html url="../data/pipelines/pipeline_1/ediarum/img/folder-frameworks.PNG" description="Anlegen der ediarum-Frameworks" %} In Oxygen müssen wir nun noch unter **Optionen > Einstellungen** in der linken Spalte zu **Dokumenttypen-Zuordnung > Orte** navigieren, dort "Benutzerdefiniert" aktivieren und dann den Dateipfad zu unserem Framework-Ordner angeben. {% include image.html url="../data/pipelines/pipeline_1/ediarum/img/oxygen-framework-connection.PNG" description="Framework-Konfiguration in Oxygen" %} Nach einem Neustart des Oxygen XML Author sollten nun unter **Optionen > Einstellungen > Dokumenttypen-Zuordnung** die *ediarum*-Frameworks aufscheinen. Hier wählen wir "ediarum.BASE.edit" aus und klicken "Erweitern" an, um unser eigenes Erweiterungsframework für unsere Edition einzubinden. Wir geben hierfür den Namen für unser editionsspezifisches Framework an, aktivieren beim Speicherort die Option "Extern" und geben dann den Pfad zu unserem zuvor angelegten Framework-Ordner an. {% include image.html url="../data/pipelines/pipeline_1/ediarum/img/oxygen-own-framework.PNG" description="Framework-Zuordnung" %}
 - Abschließend richten wir uns im Oxygen XML Author ein Projekt ein. Dafür navigieren wir in Oxygen auf **Projekt > Neues Projekt** und wählen dort den Ordner unseres projektspezifischen Frameworks aus. {% include image.html url="../data/pipelines/pipeline_1/ediarum/img/oxygen-project.PNG" description="Erstellen eines Projekts in Oxygen" %} Damit die Editorvariablen im Projekt gespeichert werden, gehen wir außerdem auf **Optionen > Einstellungen** und wählen dort "Benutzerspezifische Editorvariablen", um im Weiteren diese Projekt-Optionen anzuwenden. {% include image.html url="../data/pipelines/pipeline_1/ediarum/img/oxygen-project-variables.PNG" description="Speichern von Editorvariablen" %} → Am Ende sehen die Editorvariablen für unser spezifisches Projekt so aus: {% include image.html url="../data/pipelines/pipeline_1/ediarum/img/oxygen-editor-variables.PNG" description="Projektspezifische Editorvariablen" %}

3. Erweiterung des projektspezifischen Frameworks

a. Erstellen eines neuen Buttons

- Im vorangegangenen Kapitel haben wir uns zusätzlich zu dem über GitHub heruntergeladenen BASE-framework (ediarum.BASE.edit) auch ein eigenes projektspezifisches Framework mit dem Namen *ediarum.MARezepte.edit* angelegt. Dieses wollen wir nun erweitern. Nachdem es in unserem zuvor genutzten Transkriptionstool ([FromThePage](#)) nicht möglich war, Textstellen mit roter Schriftfarbe zu annotieren, legen wir uns nun einen Button für den Oxygen XML Author an, um Projektmitarbeitenden diese spezifische Annotation zu vereinfachen.
- Im Oxygen XML Author gehen wir unter **Optionen > Einstellungen** auf **Dokumenttypen-Zuordnung** und wählen dort das projektspezifische Framework *ediarum.MARezepte.edit*, um dieses zu bearbeiten. {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/framework-settings.PNG" description="Framework-Einstellungen" %}
- In dem neuen Dokumenttypen-Fenster wählen wir nun den Reiter **Autor** und in der linken Menüleiste **Aktionen**. Mit einem Klick auf das Plus-Zeichen öffnet sich schließlich ein weiteres Fenster, in dem wir eine neue Aktion bzw. einen neuen Button anlegen können. {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/create-action.PNG" description="Anlegen einer neuen Aktion bzw. eines neuen Buttons" %}
- Für die Konfiguration eines Buttons, der die Annotation von Textstellen mit roter Schrift erleichtert, haben wir die Felder wie folgt befüllt. {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/configure-action.PNG" description="Konfiguration der neuen Aktion" %} → Nach der Wahl einer ID, eines kurzen und verständlichen Namens sowie eines Menü-Tastenkürzels (a), haben wir einen kurzen Beschreibungstext gewählt (b), der im Oxygen XML Author als Tooltip angezeigt wird. Wir haben schließlich Icons für den Button in zwei unterschiedlichen Größen erstellt, innerhalb unseres projekteigenen Frameworks gespeichert und den Pfad zu den Icons referenziert, sowie außerdem ein Tastenkürzel gewählt (c), mit dem Projektmitarbeitende auch ohne Mausklick auf den Button Textstellen annotieren können. Im nächsten Feld (d) geben wir an, unter welcher Bedingung die Aktion überhaupt gesetzt werden darf. Da wir in unserem Fall eine Textpassage mit einem `<hi>`-Element umgeben wollen, geben wir bei der **XPath-Aktivierung** an, dass dieser Button bzw. diese Annotation nur dann möglich sein soll, wenn an dieser Stelle auch tatsächlich ein `<hi>`-Element erlaubt ist. Für diese Art der Annotation haben wir uns im Übrigen auf Grundlage der Vorgabe für **Einfärbungen** im DTABf entschieden. In dem Auswahlmenü zum Vorgang (e) wählen wir den gewünschten Prozess, den wir mit unserer Aktion ausführen wollen - in unserem Fall ist dies die "SurroundWithFragmentOperation", die bewirkt, dass die markierte Textstelle mit einem "Fragment" umgeben wird. Welches Element mit welchen Attributen dieses Fragment am Ende sein soll, legen wir schließlich etwas weiter unten fest, wo wir die Zeile mit dem Namen **fragment** auswählen und schließlich auf das Konfigurationssymbol klicken.
- In dem Fenster zur Bearbeitung des Fragments können wir angeben, dass wir die markierte Stelle mit einem `<hi>`-Element umgeben wollen, das ein Attribut `@rendition` mit dem Wert `"#red"` beinhaltet. {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/action-value.PNG" description="Auswahl der Annotation" %}
- Damit der Button letztlich auch in unserer Menüleiste erscheint und den Projektmitarbeitenden bei der Annotation zur Verfügung steht, fügen wir in dem Dokumenttypen-Fenster im Reiter **Autor** unter **Symbolleiste** unsere neue verfügbare Aktion zu den Textaktionen hinzu. {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/menu-button.PNG" description="Hinzufügen eines neuen Buttons zur Menüleiste im Autormodus" %}
- Um im Autormodus nach einem Klick auf den neu konfigurierten Button auch eine sichtbare Veränderung zu bewirken, muss im CSS noch eine entsprechende Änderung vorgenommen werden. Im Dokumenttypen-Fenster kann man einsehen, auf welches CSS zugegriffen wird. {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/css-ediarum.PNG" description="CSS für Autormodus" %} Es wäre nun entweder möglich, ein eigenes projektspezifisches CSS anzulegen, oder aber das im *ediarum.BASE.edit*-Framework bereits vorhandene `standard.css` zu erweitern. Wir haben uns dazu entschlossen, die entsprechenden Erweiterungen im `standard.css`, das im Frameworks-Verzeichnis in dem Ordner *ediarum.BASE.edit* zu finden ist, vorzunehmen. Wir fügen also folgenden CSS-Code hinzu:

```
quote hi[rendition="#red"],
text hi[rendition="#red"] {
  color: red;
  -oxy-display-tags: none;
}
```

- Wenn wir nun im Autormodus unseren neuen Button verwenden, wird unser markierter Text rot eingefärbt. Die Oxygen-Tags, die im Autormodus standardmäßig erscheinen, werden außerdem unterdrückt. {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/button-highlight.PNG" description="Sichtbares Ergebnis nach Auswahl der Aktion" %}

b. Anpassung bereits bestehender Buttons

- Grundsätzlich besteht im ediarum.BASE.edit-Framework bereits ein Button für die Annotation mit Registereinträgen, der ein Dropdown-Menü mit einer Registerliste öffnet. {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/register-shortcut.PNG" description="Vordefinierter Button für Annotation mit Registereinträgen" %} → Um die Annotation der Zutaten zu erleichtern und die Klickzahl zu minimieren, haben wir entschieden, den Listenpunkt "Sachbegriffe" aus dem Dropdown als eigenen Button in der Menüleiste anzulegen.
- Dafür navigieren wir über **Optionen > Einstellungen > Dokumenttypen-Zuordnung** zu unserem projektspezifischen Framework (ediarum.MaRezepte.edit), klicken auf "Bearbeiten" und wählen anschließend im Dokumenttypen-Fenster im Reiter **Autor** den Menüpunkt **Aktionen**. {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/register-action.PNG" description="Änderung vordefinierter Aktionen" %} Hier wählen wir die Aktion mit dem Namen "Sachbegriff" und klicken auf das Werkzeugsymbol, um diese Aktion zu bearbeiten.
- Im Fenster zur Bearbeitung der Aktion ändern wir die ID von rs[term] auf term, fügen neue Icons hinzu, die wir lokal im Ordner für das projektspezifische Framework gespeichert haben, wählen ein Tastenkürzel (Strg + I) für jene Projektmitarbeiter:innen, die die Arbeit mit der Tastatur bevorzugen, und ändern die XPath-Aktivierung dahingehend, dass wir bei der Annotation die entsprechende Textstelle nun anstelle mit einem `<rs>`-Element mit einem `<term>`-Element auszeichnen. {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/register-action-changes.PNG" description="Gestaltung des neuen Buttons für Registereinträge" %} In der Argumente-Tabelle wählen wir außerdem noch das Argument "element", um diese Anpassungen auch dort vorzunehmen.
- Bei der Bearbeitung des Argumentwerts ersetzen wir das `<rs>`-Element mit einem `<term>`-Element und wählen als Wert des Attributs `@type` die Bezeichnung "ingredient". Außerdem stellen wir vor die `$ITEMS`-Variable des `@key`-Attributs, das später auf die `@xml:id` im Register verweist, ein "#". {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/register-action-term.PNG" description="Änderung des Elements für Annotation" %}
- Damit der adaptierte Button auch in der Symbolleiste aufzufinden ist, wählen wir nun im Dokumenttyp-Fenster die Aktion Sachbegriff aus und fügen sie unter "Text(Benutzerdefinierten Author-Aktionen)" als Kindelement hinzu. {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/register-action-menu.PNG" description="Hinzufügen der adaptierten Aktion zur Symbolleiste im Autormodus" %} → Im Einstellungsfenster unter Dokumenttypen-Zuordnung sollte man im Übrigen nicht vergessen, auf "Anwenden" zu klicken, damit alle Änderungen auch tatsächlich übernommen werden.
- Zurück in der Autor-Ansicht im Oxygen XML Author verfügen wir jetzt über einen eigenen Button, der ein Fenster mit den Einträgen unseres Sachbegriff-Registers öffnet. (Weiteres zur Registerbearbeitung: siehe [Punkt 4d](#)) {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/register-ingredient-button.PNG" description="Mit einem Klick direkt zum Sachbegriff-Register" %} In der Text-Ansicht ist es

möglich, zu überprüfen, ob unsere Änderungen auch tatsächlich zum gewünschten Output führen (sofern wir bereits Einträge in unserem Sachregister haben). {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/register-tag.PNG" description="Ergebnis des Register-Buttons im Code" %}

4. Bearbeitung der Dokumente

- Bei jedem Start unserer Arbeitsroutine müssen wir zuerst Docker Desktop aktivieren und anschließend den exist-Container starten, indem wir auf die Play-Schaltfläche klicken. {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/exist-container-run.PNG" description="Starten des exist-Containers in Docker Desktop" %} → Mit einem Klick auf den Port (8080:8080) öffnet sich schließlich im Browser das eXist-db-Dashboard, über das wir uns anmelden können, falls wir Anpassungen an den Projektkonfigurationen in der *ediarum*-App vornehmen wollen.
- Wenn uns das Status-Feld unseres exist-Containers "Running" anzeigt, wurde die Verbindung zur eXist-Datenbank hergestellt und wir haben auch im Oxygen XML Author über den Datenquellen-Explorer Zugriff auf unsere Dateien. Wir öffnen das erste Manuskript und müssen nur mehr sicherstellen, dass wir uns im Autormodus befinden, um direkt mit der Bearbeitung beginnen zu können. {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/author-mode-start.PNG" description="Start der Editionsarbeit im Autormodus" %}
- Einige einfache Annotationen, die in unserem Workflow bereits über [FromThePage](#) möglich waren, haben wir übernommen bzw. im Zuge unserer [Transition](#) für die weitere Bearbeitung in *ediarum* transformiert. Für unser Beispielprojekt ergeben sich daher noch folgende Bearbeitungsschritte:
 - Bearbeitung der nach DTABf obligatorischen Metadaten
 - Ergänzung von Annotationen, die in FromThePage vorgenommen wurden, aber in *ediarum* (bzw. gemäß DTABf) bei der Validierung aufgrund eines fehlenden Attributs eine Fehlermeldung werfen
 - Annotation von Textstellen mit roter Schrift, für die es in FromThePage keine Annotationsmöglichkeit gab
 - Annotation von Zutaten aus dem eigens erstellten Zutatenregister → Auf all diese Schritte soll nachfolgend genauer eingegangen werden.

a. Bearbeitung der Manuskript-Metadaten

- Für die Bearbeitung der Metadaten ist es notwendig, den Projektmitarbeitenden, Metadaten zur Verfügung zu stellen, damit diese alle Felder, die während der [Transition von FromThePage zu ediarum](#) mit einem Hinweis versehen wurden bzw. über einen Platzhalter in eckigen Klammern verfügen, entsprechend ausfüllen können. {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/editing-metadata.PNG" description="Ausfüllen von Platzhaltern" %} Wir gehen hierzu einfach auf die zu bearbeitende Stelle, löschen den Platzhalter und überschreiben die Stelle mit den entsprechenden Metadaten. {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/edited-metadata.PNG" description="Befüllen der Metadaten" %} → Anhand der oberen Leiste können wir auch stets nachvollziehen, in welchem Element wir uns befinden.
- Leider werden nicht alle Elemente, die im Textmodus im `<teiHeader>` vorzufinden sind, im Autormodus angezeigt. So findet man im Autormodus das `<publicationStmnt>` beispielsweise weder in der `<fileDesc>`, noch in der `<sourceDesc>`. Diese Metadatenfelder können somit also nur im Textmodus bearbeitet bzw. ergänzt werden.
- **Kleiner Exkurs hinsichtlich der Möglichkeiten einer Metadaten-Anreicherung über die ediarum-Werkzeugeiste:** Für die Erweiterung der Metadaten über die *ediarum*-Oberfläche klicken wir in der Werkzeugleiste auf "Metadaten" und wählen zu Demonstrationszwecken für unser Projekt den Punkt "Ungefährer Erstellungszeitraum". {% include image.html url="../../data/pipelines/pipeline_1/ediarum/img/metadata-additions.PNG" description="Neues Metadatenfeld" %}

einfügen" %} Daraufhin erscheinen drei Eingabeaufforderungen zu dem frühest-möglichen Erstelldatum, dem spätest-möglichen Erstelldatum sowie einer Angabe, wie sicher man sich mit der Datierung ist. {% include image.html url="../../../data/pipelines/pipeline_1/ediarum/img/metadata-extras.PNG" description="Eingabeaufforderungen zu den Metadaten" %} In unserem Dokument finden wir schließlich eine neue Zeile mit dem entsprechenden Eintrag. {% include image.html url="../../../data/pipelines/pipeline_1/ediarum/img/metadata-adds.PNG" description="Anzeige der hinzugefügten Metadaten im Automodus" %} → Da wir das DTABf-Schema eingebunden haben, in dem das `<creation>`-Element im `<teiHeader>` nicht vorgesehen ist, wird diese Metadaten-Erweiterung rot unterwellt. Es müsste also an dieser Stelle entweder das DTABf-Schema an unsere projektspezifischen Metadaten-Anforderungen angepasst werden oder wir ignorieren die fehlerhafte Validierung. Da wir in unserem Projekt nicht auf die Datierung verzichten wollen, aber auch das Schema nicht anpassen wollen, entfernen wir diese zusätzliche Metadaten-Angabe trotz fehlerhafter Validierung nicht.

b. Annotationen mit eigenem Button

- Einige der Annotationen haben wir in unserem Beispielprojekt bereits mithilfe von [FromThePage](#) vorgenommen. Nicht möglich war es uns jedoch, farbliche Hervorhebung - also in unserem Fall rote Textstellen - als solche auszuzeichnen. Dies können unsere Projektmitarbeitenden nun mittels des eigens dafür erstellten Rotstift-Buttons. Für die Bearbeitung der Transkripte benötigen wir in diesem Fall für die Annotation die Digitalisate der Manuskripte. Wir öffnen dafür im Oxygen XML Author einerseits unter **Fenster > Ansicht zeigen** die **Bildvorschau** und wählen andererseits für die Projektansicht den Menüpunkt **Projekt** unter demselben Pfad. In der Projektansicht navigieren wir zu unserem ediarum.MaRezepte.edit-Ordner und legen in diesem einen neuen Ordner mit der Bezeichnung "faksimile" an, in den wir unsere Faksimiles hineinkopieren. Wenn wir hier nun auf eine der Bilddateien klicken, öffnet sich diese in der Bildvorschau und wir verfügen somit über eine Bild-Text-Ansicht, die es uns erleichtert, die entsprechenden in roter Farbe geschriebenen Textstellen im Original zu finden und diese Passage entsprechend im Transkript zu markieren und mittels Rotstift-Button zu annotieren. {% include image.html url="../../../data/pipelines/pipeline_1/ediarum/img/red-highlighting.PNG" description="Annotation von Textstellen mit roter Schrift" %}
- Sollten wir einmal aus Versehen eine Textstelle falsch ausgezeichnet haben, gibt es in *ediarum* auch die Möglichkeit, die Auszeichnung wieder zu entfernen. Da der Button hierfür aber nicht standardmäßig in der Werkzeugleiste aufscheint, müssen wir zuerst in dem Fenster, das sich über einen Rechtsmausklick auf den Freibereich im Menü öffnet, die Option "Werkzeugleisten konfigurieren..." auswählen. {% include image.html url="../../../data/pipelines/pipeline_1/ediarum/img/configure-menu-view.PNG" description="Änderungen an der Werkzeugleiste vornehmen" %} In dem darauf erscheinenden Bearbeitungsfenster haken wir die Checkbox "Auszeichnung" an und klicken anschließend auf "OK", um die Änderungen zu übernehmen. {% include image.html url="../../../data/pipelines/pipeline_1/ediarum/img/delete-annotation.PNG" description="Auswahl weiterer Buttons für die Werkzeugleiste" %} Nun erscheint die Button-Leiste zu den Auszeichnungen in der *ediarum*-Werkzeugleiste. Wenn wir unseren Cursor in eine annotierte Textstelle setzen und den entsprechenden Button betätigen, wird die bereits getätigte Annotation wieder entfernt. {% include image.html url="../../../data/pipelines/pipeline_1/ediarum/img/delete-annotation-menu.PNG" description="Erweiterte Werkzeugleiste" %}

c. Ergänzung von nicht validen Annotationen

- Außerdem befinden sich in den bereits teilweise über FromThePage annotierten Transkriptionen einige Auszeichnungen, die noch nicht dem DTA-Basisformat entsprechen. Dies trifft in unserem Beispielprojekt auf Tilgungen und Ergänzungen durch die ursprüngliche Schreiber:in zu, da wir hier über FromThePage mit den vorgegebenen Buttons nur Auszeichnungen mit ``- und `<add>`-Elementen vornehmen, aber keine Attribute setzen konnten. Im Oxygen XML Author sind diese nicht validen Stellen rot unterwellt. Mit einem

Klick auf die nicht valide Textstelle erscheint links ein kleines Glühbirnen-Symbol mit Sofort-Lösungsvorschlägen. {% include image.html url="../../../data/pipelines/pipeline_1/ediarum/img/attribute-problem1.PNG" description="Anzeige von nicht DTABf-konformen Auszeichnungen" %} Durch die Auswahl von "Attribute 'rendition' hinzufügen" erscheint ein neues Fenster, mit dem wir für das @rendition-Attribut schließlich über das Dropdown einen entsprechenden Wert aussuchen können. Da wir im Faksimile erkennen können, dass die getilgten Wörter durchgestrichen wurden, wählen wir das #s, welches für Streichungen steht. {% include image.html url="../../../data/pipelines/pipeline_1/ediarum/img/rendition-choice.PNG" description="Anpassung nicht valider Annotationen" %} → Die Unterwellung verschwindet schließlich, da dieses Element nun das im DTABf erforderliche @rendition-Attribut erhalten hat.

- Das Gleiche machen wir nun auch für die <add>-Elemente, die ebenfalls unterwellt sind, da diesen für ihre DTABf-Konformität das @place-Attribut fehlt. {% include image.html url="../../../data/pipelines/pipeline_1/ediarum/img/attribute-problem2.PNG" description="Ergänzung von Attributen zur gültigen Validierung nach dem DTABf" %}

d. Annotation mit Registereinträgen

- Für die Annotation der Zutaten, die in den Manuskripten zu finden sind, gibt es nun zwei Möglichkeiten. Entweder besteht bereits eine Zutatenliste und man überführt diese in die für *ediarum* geeignete XML-Struktur oder man fügt erst während des Annotationsprozesses in *ediarum* die in den Rezepten auftretenden Zutaten dem Register nach und nach hinzu. In unserem Beispielprojekt besteht bereits eine Zutatenliste, die aber noch nicht über alle Zutaten-Einträge verfügt. Daher wird im Folgenden einerseits der Workflow mit einem bereits bestehenden Register sowie das nachträgliche Hinzufügen von Registereinträgen beschrieben.
- Unsere (unvollständige) Zutatenliste, die zuerst nur als Exceltabelle verfügbar war, wurde über die [Transition OpenRefine → ediarum](#) mit Wikidata-Normdaten angereichert und schließlich in das Sachregister überführt, wodurch wir hier mit der Annotation der Zutaten fortsetzen können.
- Da wir aufgrund des DTABf-Schemas, das wir im Zuge der [Transition FromThePage → ediarum](#) eingebunden haben, aber das Problem haben, dass unsere <term>-Elemente, die wir für die Zutaten verwenden, im Fließtext nicht verwendet werden können und daher der Zutaten-Button noch nicht funktioniert, müssten wir nun entweder das eingebundene DTABf-Schema anpassen oder könnten als Workaround im Textmodus die ersten beiden <?xml-model>-Elemente entfernen. Wir haben uns im Rahmen unseres Projekts für Letzteres entschieden. {% include image.html url="../../../data/pipelines/pipeline_1/ediarum/img/register-delete-schema.PNG" description="Entfernung der eingebundenen Schemata" %}
- Sobald wir anschließend im Text auf eine Zutat stoßen, klicken wir auf den Zutaten-Button und wählen aus der Registeransicht, den entsprechenden Eintrag. {% include image.html url="../../../data/pipelines/pipeline_1/ediarum/img/ingredient-annotation.PNG" description="Annotation der Zutaten mit projektspezifischem Button" %}
- Sollte eine Zutat noch nicht in unserem Register auffindbar sein, können wir diese manuell hinzufügen. Dafür öffnen wir im Datenquellen-Explorer im Ordner "Register" die Datei "Sachbegriffe.xml" und legen dort einen neuen Eintrag an. {% include image.html url="../../../data/pipelines/pipeline_1/ediarum/img/register-new-entry.PNG" description="Neuen Registereintrag anlegen" %}
- Über den Punkt **Sachregister** in der untersten Menüleiste ist es uns außerdem möglich, eine "Alternative Bezeichnung" anzulegen (oder sogar Unterlisten zu erstellen). Dafür müssen wir auf das Label klicken, um nicht den gesamten Eintrag angewählt zu haben. {% include image.html url="../../../data/pipelines/pipeline_1/ediarum/img/register-label-alt.PNG" description="Weitere Bezeichnungen für einen Eintrag hinzufügen" %} → An dieser Stelle fügen wir im Sinne unseres Beispielprojekts die frühneuhochdeutsche Bezeichnung ein.
- In der Menüleiste unter **Allgemein** gibt es die Möglichkeit, eine Norm-ID hinzuzufügen. Diese ist im Register-Framework von *ediarum* vorerst aber nicht für Sachbegriffe aktiviert und daher ausgegraut. Um dies zu ändern, wählen wir unter **Optionen > Einstellungen > Dokumenttypen-Zuordnung**

"ediarum.REGISTER.edit" und gehen auf "Bearbeiten". {% include image.html url="../data/pipelines/pipeline_1/ediarum/img/register-new-action.PNG" description="Änderungen im Register-Framework" %} Im Reiter **Autor** wählen wir schließlich **Aktionen** und suchen in der Spalte Name nach "Norm-ID", um schließlich über das Werkzeugsymbol zur Bearbeitungsansicht zu kommen. {% include image.html url="../data/pipelines/pipeline_1/ediarum/img/register-id-action.PNG" description="Bearbeitung des Norm-ID-Buttons" %} Unter Vorgänge gehen wir auf einen der 4 Vorgänge und duplizieren diesen. {% include image.html url="../data/pipelines/pipeline_1/ediarum/img/register-copy-action.PNG" description="Kopie einer Aktion" %} Nun tauschen wir "org" mit "label" aus, um das Einfügen einer ID auch in unserem Sachregister zu ermöglichen, und speichern unsere Einstellungen. Hierbei sollte nicht vergessen werden, im Einstellungsfenster noch auf "Anwenden" zu klicken, damit unsere Änderungen auch wirklich übernommen werden. {% include image.html url="../data/pipelines/pipeline_1/ediarum/img/register-adapt-action.PNG" description="Anpassung der kopierten Aktion" %} Jetzt können auch im Sachregister Norm-IDs - wie in unserem Fall der entsprechende Wikidata-Link - hinzugefügt werden. {% include image.html url="../data/pipelines/pipeline_1/ediarum/img/register-wikidata-idno.PNG" description="Wikidata-Link zum Registereintrag hinzufügen" %}

- Zuletzt möchten wir für unseren Eintrag noch die automatisch generierte @xml:id ändern. Dafür gehen wir auf unseren Eintrag (und zwar am Besten über den Wikidata-Link, damit wir den gesamten Eintrag markieren und nicht nur das Label oder die Idno) und wählen nach einem Rechtsmausklick im daraufhin erscheinenden Menü "Attribute bearbeiten". {% include image.html url="../data/pipelines/pipeline_1/ediarum/img/register-add-xml-id.PNG" description="Änderung der automatisch generierten xml:id" %} Wir geben hier unseren neuen Wert - in unserem Fall die englische Übersetzung - ein und speichern unsere Änderung. {% include image.html url="../data/pipelines/pipeline_1/ediarum/img/register-xml-id.PNG" description="Eingabe einer individuellen xml:id" %} → Während dieses Prozesses hat sich ein kleines Problem ergeben: Leider ist das Register nicht auf die alternativen Bezeichnungen durchsuchbar, sondern man muss den genauen Wortlaut des Eintrags wissen, der die reguläre Schreibweise abbildet. So, hatten wir beispielsweise nach dem Begriff "Nelke" gesucht, wurden aber aufgrund dessen, dass es keine Substring-Suche gibt, nicht fündig. Denn die Nelke ist in unserem Register unter "Gewürznelke" gespeichert, und erst als wir die xml:id mit dem Wert "clove" anlegen wollten, haben wir die Meldung bekommen, dass diese xml:id (für den Eintrag Gewürznelke) bereits besteht. Ein weiterer Nachteil ist, dass die <label>-Elementen mit @alt-Attribut der einzelnen Einträge, in denen die frühneuhochdeutschen Schreibweisen gespeichert sind, nicht während der Annotation durchsucht werden können. Dadurch müssten wir jedes Mal in die Registeransicht wechseln und manuell überprüfen, ob die im Manuskript auftretende Schreibung bereits im Register festgehalten ist.

5. Export der Dokumente

- Um die XML-Dateien auf unserem lokalen Gerät zu speichern, gehen wir mit einem Rechtsmausklick auf jenen Ordner, dessen Inhalte wir herunterladen wollen, und klicken auf "Exportieren". {% include image.html url="../data/pipelines/pipeline_1/ediarum/img/export-data.PNG" description="Export der annotierten XML-Dateien" %}
- Im darauffolgenden Fenster geben wir den Speicherort an. {% include image.html url="../data/pipelines/pipeline_1/ediarum/img/export-data-path.PNG" description="Angabe des Speicherorts" %} → Die Dateien befinden sich jetzt in unserem lokalen Verzeichnissystem.

Kontakt

Unternehmensgröße:

Weblink: www.weblink.com

Allgemeiner Support	ediarum@bbaw.de
Mailingliste (für DH-Entwickler:innen)	dev.list@ediarum.org
Martin Fechner (ediarum Web + eXist)	fechner@bbaw.de

Ressourcen

- [Workshop- und MeetUp-Termine](#)

Dokumentation

- [ediarum Setup-Anleitung](#)
- [ediarum.BASE Benutzerhandbuch](#)
- [Oxygen XML Author](#)
 - [Erstellung von Buttons](#)
- [XPath](#)
- [eXist-db](#)
- [GitHub Repository \(Informationen zu einzelnen Modulen\)](#)

Tutorials

- [Eigenes Framework für Oxygen XML bauen](#)

Projekte, die dieses Tool genutzt haben

- [edition humboldt digital](#): Das Editionsprojekt "Alexander von Humboldt auf Reisen - Wissenschaft aus der Bewegung" ediert und publiziert Reisetagebücher, Briefe sowie andere Dokumente seit 2015 mit *ediarum*. In der edition humboldt digital werden die edierten Texte nicht nur präsentiert, sondern auch über Personen-, Orts-, Werk-, Institutions- und Pflanzenregister erschlossen.
- [Schleiermacher in Berlin 1808-1834](#): In diesem Projekt werden Briefwechsel, Tageskalender sowie Vorlesungen von Friedrich Schleiermacher während seiner Zeit in Berlin ediert und die Edition durch Personen-, Orts-, Werkregister und ein Verzeichnis zu den erwähnten Bibelstellen erschlossen.

Literatur

- Arndt, N., & Wegener, L. (2019). Überlegungen zur digitalen Edition mystischer Mosaiktraktate des Spätmittelalters. *Das Mittelalter*, 24(1), 15–30. <https://doi.org/10.1515/mial-2019-0003>
- Dumont, S., Arndt, N., Grabsch, S., & Klappenbach, L. (2021). *ediarum.BASE.edit v2.0.0* (v2.0.0) [Computer software]. Zenodo. <https://doi.org/10.5281/ZENODO.5897100>
- Dumont, S., & Fechner, M. (2014). Bridging the Gap: Greater Usability for TEI encoding. *Journal of the Text Encoding Initiative*, Issue 8. <https://doi.org/10.4000/jtei.1242>
- Dumont, S., & Fechner, M. (2019, April 2). *ediarum – Arbeits- und Publikationsumgebung für digitale Editionsprojekte*. <https://doi.org/10.5281/zenodo.2621062>
- Fechner, M. (2018, März 4). *A Standardized Interface for Digital Scholarly Editions*. DHd 2018 - Konferenz der Digital Humanities im deutschsprachigen Raum, Köln vom 26.02.-02.03.2018, Köln. urn:nbn:kobv:b4-opus4-33277
- Fechner, M., & Dumont, S. (2019). *ediarum - from bottom-up to generic programming*. <https://av.tib.eu/media/42492>

- Fechner, M., Dumont, S., JanWierzoch, Lauml, & Grobian. (2022). *ediarum/ediarum.DB: ediarum.DB 4.0.2* (v4.0.2) [Computer software]. Zenodo. <https://doi.org/10.5281/ZENODO.5940465>
- Kraft, T. (2018). Hybride Edition und analoges Erbe: Editionsphilologie und Alexander von Humboldt-Forschung in der digitalen Sattelzeit. *Informatik-Spektrum*, 41(6), 385–397. <https://doi.org/10.1007/s00287-018-01130-5>
- Kraft, T., & Thomas, C. (2022). *Datenmanagementplan des Akademienvorhabens „Alexander von Humboldt auf Reisen – Wissenschaft aus der Bewegung“*. Berlin-Brandenburgische Akademie der Wissenschaften. <https://edoc.bbaw.de/opus4-bbaw/frontdoor/index/index/docId/3725>
- Mertgens, A. (2019). *Ediarum: a toolbox for editors and developers*. *RIDE* 11. <https://doi.org/10.18716/RIDE.A.11.4>
- Vetter, A. (2022). *ediarum.MEDIAEVUM*. Eine Arbeitsumgebung zur Edition mittelalterlicher (Prosa)Texte. *Beiträge zur mediävistischen Erzählforschung, Themenheft 12*, 47–64 Seiten. <https://doi.org/10.25619/BME20223194>

Factsheet

System	
Scope des Tools	Annotation
Softwareumgebung/Softwaretyp (Remotesystem im Browser / Lokaler Client)	Applikation/Plug-In
Unterstützte Plattformen	Linux, Windows & Mac
Geräte	Desktop
Einbindung anderer Systeme (Interoperabilität)	☑
Accountsystem	☑ Hinzufügen von mehreren Usern möglich
Kostenmodell (Kostenübersicht/Open Source)	<u>ediarum & eXist-db</u> : kostenlos <u>Oxygen</u> : \$190 - 244/Halbjahres-Abo \$335 - 432/Jahres-Abo
Anforderungen & Methoden	
Erforderte Code Literacy	fortgeschritten
Interface-Sprachen (ISO 639-1)	de
Unterstützte Zeichenkodierung	Latin-1, UTF-8, UTF-16
Inkludierte Datenkonvertierung (Im Preprocessing mögliche Anpassung der Daten an für die Software erforderliches Format)	✗
Abhängigkeit von anderer Software (Falls ja, wird diese Software automatisch mitinstalliert?)	☑ eXist-db und Oxygen sind eigenständig zu installieren
Erforderliche Plug-Ins (bei web-basierten Anwendungen)	[nicht anwendbar]

Dokumentation & Support

Wartung und ständige Erweiterung	✓
Einbindung der Community	✓ GitHub
Dokumentation	✓
Dokumentationssprache	Deutsch, Spanisch
Dokumentationsformat	HTML
Dokumentationsabschnitte	Einführung, Allgemeine Bedienung, Metadaten, Text, Brief, Register
Verfügbarkeit von Tutorials	✓ für Oxygen XML Frameworks
Aktiver Support/Community (Forum, Slack, Issue Tracker etc.)	✓ GitHub-Issues-Mechanismus, Mailingliste

Nutzbarkeit & Nachhaltigkeit

Installationsablauf	fortgeschrittene Kenntnisse nötig
Test (Gibt es ein Test Suite, um zu überprüfen, ob die Installation erfolgreich war?)	✓
Lizenz, unter der das Tool veröffentlicht wurde	[GNU 3.0](https://www.gnu.org/licenses/gpl-3.0.en.html)
Registrierung in einem Repository	✓ GitHub
Möglichkeit zur Software-Entwicklung beizutragen	✓
Benutzerinteraktion & Benutzeroberfläche	
Benutzerprofil (erwartete Nutzer:innen)	GeWi-Forschungsinstitutionen und GeWi-Forschende als Tool-Nutzende
Benutzerinteraktion (erwartete Nutzung)	Hochladen von Dateien, Projektmanagement, Edition bzw. Annotation von Texten, Indexierung, Export
Benutzeroberfläche	GUI
Visualisierungen (Analyse-, Input-, Outputkonfigurationen)	✗

Benutzerverwaltung

Personenverwaltung	✓ inklusive Rollenzuweisung
Interne Kommunikationsmöglichkeiten (z. B. Annotationsrichtlinien, Kommentarfunktionen, ...)	✓ Kommentarfunktion

Daten- und Toolverwaltung

Zentrale/dezentrale Verwaltungsmöglichkeit	✓
Versionskontrolle	✗

Projektspezifische Einstellungen	<input checked="" type="checkbox"/>
API	<input type="checkbox"/>
Möglichkeit auf simultanes Arbeiten	<input checked="" type="checkbox"/>
Datenupload	
Unterstützte Dateiformate	XML, TEI-XML
Informationen zur Datensicherheit	[nicht anwendbar, da Datenbank und Daten auf einem von der Projektleitung selbst gewählten Server gespeichert sind]
Zugänglichkeit von verschiedenen Standorten/Geräten	<input checked="" type="checkbox"/>
Einschränkungen hinsichtlich der Datenmenge	[keine Angabe]
Verlustfreier Upload von bereits bearbeiteten Dokumenten	<input checked="" type="checkbox"/>
Unterstützung von IIIF-Import	<input type="checkbox"/>
Datenbearbeitung (Annotationstool)	
Komplexitätsgrad bei Annotation (z. B. Verfügbarkeit von Buttons, Drag&Drop-Funktion, ...)	Buttons für Annotationen (individuelle Buttons müssen jedoch zuerst erstellt werden - benötigt fortgeschrittene Kenntnisse)
Standardeinstellungen entsprechend bestimmten Standards für Digitale Editionen	<input checked="" type="checkbox"/> TEI, DTA-Basisformat
Anpassungsmöglichkeit und Validierung entsprechend projektspezifischen Konventionen/Schemata	<input checked="" type="checkbox"/>
Definition eigener/projektspezifischer Tags	<input checked="" type="checkbox"/>
Metadaten-Anreicherung	<input checked="" type="checkbox"/>
Eigene Indexierung	<input checked="" type="checkbox"/> Registerfunktion
Möglichkeit von Textvergleich bzw. Arbeit an Variantenapparat	<input type="checkbox"/>
Ansichtsmöglichkeiten (z. B. Bearbeitungsansicht, Synopsen-Ansicht, Vorschauansicht ...)	Vorschauansicht (=Autormodus), Bearbeitungsansicht (=Textmodus), Synopsen-Ansicht
Verlinkung von Entitäten, NER	<input checked="" type="checkbox"/> über Register möglich
Datenexport	
Unterstützte Dateiformate	XML, TEI_XML
Datenverlust (nicht vollständiger Erhalt von Annotationen, die bereits vor Verwendung des Tools gemacht wurden)	<input type="checkbox"/>

Validierungsmöglichkeit für TEI-XML vor Export**Datenaufbewahrung nach Export**

[nicht anwendbar, da Datenbank und Daten auf einem von der Projektleitung selbst gewählten Server gespeichert sind]