ingenieur
wissenschaften
htw saar

Hochschule für
Technik und Wirtschaft
des Saarlandes
University of
Applied Sciences

Projektbericht

zur Projektarbeit im Rahmen des Studiengangs Praktische Informatik
der Fakultät für Ingenieurwissenschaften

Assistenzsystem zur Eintragung wissenschaftlicher Arbeiten

vorgelegt von
Noel Vanelle Tchinda Kuegouo
Chrislie Briel Mohomye Yotchouen 5013415

betreut und begutachtet von
Prof. Dr. Maximilian Altmeyer

Saarbrücken, 06. 08 2025

Selbständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit (bei einer Gruppenarbeit: den entsprechend gekennzeichneten Anteil der Arbeit) selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich erkläre hiermit weiterhin, dass die vorgelegte Arbeit zuvor weder von mir noch von einer anderen Person an dieser oder einer anderen Hochschule eingereicht wurde.

Darüber hinaus ist mir bekannt, dass die Unrichtigkeit dieser Erklärung eine Benotung der Arbeit mit der Note „nicht ausreichend“ zur Folge hat und einen Ausschluss von der Erbringung weiterer Prüfungsleistungen zur Folge haben kann.

Saarbrücken, 06. 08 2025

Noel Vanelle Tchinda Kuegouo

Zusammenfassung

Kurze Zusammenfassung des Inhaltes in deutscher Sprache, der Umfang beträgt zwischen einer halben und einer ganzen DIN A4-Seite.

Orientieren Sie sich bei der Aufteilung bzw. dem Inhalt Ihrer Zusammenfassung an Kent Becks Artikel: `http://plg.uwaterloo.ca/~migod/research/beckOOPSLA.html`.

We have seen that computer programming is an art,
because it applies accumulated knowledge to the world,
because it requires skill and ingenuity, and especially
because it produces objects of beauty.

— Donald E. Knuth [1]

Danksagung

Hier können Sie Personen danken, die zum Erfolg der Arbeit beigetragen haben, beispielsweise Ihren Betreuern in der Firma, Ihren Professoren/Dozenten an der htw saar, Freunden, Familie usw.

Inhaltsverzeichnis

1 Einleitung

In diesem Kapitel wird das Projekt vorgestellt. Es wird die Motivation hinter der Entwicklung beschrieben, das zugrundeliegende Problem erläutert und schließlich das Ziel des Projekts klar formuliert.

1.1 Motivation

Die Idee für dieses Projekt entstand aus der Beobachtung, dass viele Professorinnen und Professoren beim Archivieren ihrer wissenschaftlichen Publikationen in Opus häufig auf unnötige Hürden stoßen. Der aktuelle Prozess erfordert oft eine manuelle Eingabe der Daten, was sowohl zeitintensiv als auch fehleranfällig ist. Besonders bei umfangreichen Publikationslisten kann dies zu einem erheblichen Aufwand führen.

Das Projekt verfolgt daher die Idee, ein einfaches und benutzerfreundliches Tool zu entwickeln. Mit diesem sollen Nutzerinnen und Nutzer eine .bib-Datei – ein gängiges Format zur Speicherung bibliographischer Daten – importieren können. Anschließend sollen sie die Möglichkeit haben, die Einträge einzeln einzusehen und bei Bedarf zu bearbeiten. Auf diese Weise wird der gesamte Vorgang transparenter und effizienter gestaltet.

Die Relevanz dieses Ansatzes liegt in der deutlichen Zeitersparnis und der Verbesserung der Datenqualität. Durch die Möglichkeit, die aufbereiteten Einträge direkt in Opus zu exportieren, wird der Publikationsprozess erheblich vereinfacht. Dies führt nicht nur zu einer besseren Organisation wissenschaftlicher Arbeiten, sondern unterstützt auch die Hochschulen bei der langfristigen und fehlerfreien Archivierung von Forschungsleistungen.

1.2 Problembeschreibung

In dem heutigen wissenschaftlichen Bereich spielt die Verwaltung von Literaturquelle eine große Rolle . In diesem Zusammenhang wird zur Verwaltung und Zitierung der Literaturquellen eine `.bib`-Datei verwendet, die eine strukturierte und automatisierte Einbindung der Referenzen im Dokument ermöglicht. Ein häufiges Problem besteht darin, dass die manuelle Bearbeitung von `.bib`-Datei zu Formatierungsfehlern oder sogar zu fehlenden Einträgen in der Literaturverzeichnis führen kann. Besonders bei umfangreichen Projekten mit vielen Quellen wird die Pflege der Datei schnell unübersichtlich und fehleranfällig. Auch kleinere Eingabefehler können dazu führen, dass Quellen nicht korrekt angezeigt oder überhaupt nicht übernommen werden. Deshalb haben wir ein ein benutzerfreundliches Assistenzsystem zur Eintragung wissenschaftlicher Arbeiten entwickelt , um dieses Problem zu lösen. Dieses System soll die Qualität der Literaturverwaltung verbessern und den Arbeitsaufwand für die Professoren und Studierenden deutlich reduzieren.

1.3 Ziel des Projekts

Das Ziel des Projekts ist die Entwicklung eines einfach zu bedienenden Tools zur Erfassung und Verwaltung wissenschaftlicher Arbeiten. Nutzerinnen und Nutzer sollen die Möglichkeit haben, eine bestehende .bib-Datei auszuwählen, deren Einträge sich einzeln anzeigen und bei Bedarf bearbeiten. Ein weiterer zentraler Bestandteil des Tools ist die direkte Exportfunktion in das Repositorium Opus. Damit wird der Publikationsprozess erheblich vereinfacht und beschleunigt. Darüber hinaus trägt das System dazu bei, die Qualität und Einheitlichkeit der bibliographischen Daten zu verbessern, indem Fehlerquellen minimiert und wiederkehrende Arbeitsschritte automatisiert werden. Langfristig ermöglicht das Tool eine bessere Organisation wissenschaftlicher Arbeiten und eine Zeitersparnis für die Nutzerinnen und Nutzer, wodurch der gesamte Archivierungs- und Publikationsprozess effizienter gestaltet wird.

2 Verwandte Arbeiten

3 Aufbau des Projektteams

In diesem Kapitel wird beschrieben, wie das Projektteam zusammengesetzt war und wie die Zusammenarbeit im Verlauf des Projekts verlief. Zunächst wird das Team vorgestellt, anschließend wird erläutert, wie die Arbeitsaufteilung und Kommunikation erfolgte. Zum Schluss erfolgt eine Darstellung der einzelnen Rollen und Aufgaben der Teammitglieder.

3.1 Übersicht

Das Projekt wurde ursprunglich in einer Dreiergruppe gestartet durchgeführt, bestehend aus Briel Mohomye, Vanelle Tchinda und Tony Nsangou. Aufgrund organisatorischer Schwierigkeiten konnte das dritte Teammitglied jedoch nicht offiziell am Projekt teilnehmen. Aus diesem Grund wurde das Projekt im weiteren Verlauf von Briel Mohomye und Vanelle Tchinda eigenständig durchgeführt. Beide Teammitglieder haben gleichberechtigt an der Konzeption, Entwicklung und Dokumentation des Tools mitgewirkt. Ziel war es, die Stärken und Kompetenzen beider Personen optimal einzubringen, um die Arbeit effizient und zielgerichtet zu gestalten.

3.2 Plan zur Zusammenarbeit im Team

Wir haben Git als Versionskontrollsystem genutzt, um den Code gemeinsam zu verwalten und Konflikte zu vermeiden. Regelmäßig fanden Meetings mit dem Professor statt, um den Fortschritt zu besprechen und Rückmeldungen einzuholen. Nach jedem Meeting mit dem Professor trafen wir uns im Team, um die Aufgaben zu verteilen und den weiteren Arbeitsplan zu besprechen. Jede Person arbeitete eigenständig an ihren zugeteilten Aufgaben. Dabei gaben wir uns gegenseitig Feedback und konstruktive Kritik, um die Qualität unserer Arbeit kontinuierlich zu verbessern. Dieses Vorgehen haben wir bis zum aktuellen Stand des Projekts beibehalten.

Bezüglich der Implementierung waren ursprünglich zwei Personen für das Backend zuständig, insbesondere für den Parser-Service und die API-Endpunkte, während eine Person das Frontend entwickelte. Nachdem eine Person das Team verlassen hatte, haben die verbleibenden zwei Teammitglieder die Arbeit gerecht untereinander aufgeteilt, um alle Aufgaben weiterhin abzudecken.

3.3 Funktion der einzelnen Personen und Umsetzungskette

Zu Beginn der technischen Umsetzung konzentrierten sich zwei Teammitglieder auf die Backend-Entwicklung, mit dem Fokus auf den Aufbau eines Parser-Services zur Konvertierung von `.bib`-Dateien in ein JSON-Format sowie auf die Gestaltung der benötigten API-Endpunkte. Parallel dazu begann ein drittes Teammitglied mit der Gestaltung des Frontends. Nach dem vorzeitigen Ausscheiden eines Teammitglieds übernahmen die beiden verbleibenden Personen zusätzlich die Weiterentwicklung der Benutzeroberfläche und stimmten ihre Aufgaben fortan eng miteinander ab. Die Umsetzung begann mit

der Entwicklung des Parsers im Backend, welcher die Einträge aus der `.bib`-Datei in ein JSON-Format überträgt. Darauf folgte die Erstellung der API-Endpunkte zur Kommunikation mit dem Frontend.

Im Anschluss wurde mit der Frontend-Entwicklung begonnen. Dabei wurde schrittweise folgendes umgesetzt:

- Implementierung eines **„Importieren"-Buttons** zur Auswahl einer `.bib`-Datei.
- Darstellung der Einträge in einer **aufklappbaren Struktur**.
- Möglichkeit zur **Bearbeitung einzelner Einträge** direkt in der Oberfläche.
- Einbindung einer **Suchkomponente** zur Filterung der Einträge.
- Implementierung der **Paginierung** zur besseren Übersicht bei vielen Einträgen.
- Nutzung von **Local Storage**, um Änderungen lokal zu speichern.
- Einfügen eines **„Speichern"-Buttons** zur endgültigen Sicherung der bearbeiteten Daten.
- Trennung der **Autoreninformationen** aus der `.bib`-Datei in einzelne Felder.
- Einfügen eines **Buttons zum Löschen aller Einträge**.
- Hinzufügen eines **„Exportieren"-Buttons** zur Ausgabe der modifizierten Daten.

Im Verlauf des Projekts wurden die einzelnen Entwicklungsschritte regelmäßig auf Grundlage der Besprechungen mit dem Betreuer angepasst. Dabei wurde großer Wert auf transparente Kommunikation und iterative Verbesserung gelegt. Durch kontinuierliche Rücksprachen und konstruktiven Austausch konnte die Arbeit effizient koordiniert und gemeinsam vorangetrieben werden.

4 Umsetzung

4.1 Systemarchitektur

Das entwickelte Assistenzsystem für wissenschaftliches Arbeiten basiert auf einer klar strukturierten 3-Schichten-Architektur. Es besteht aus einem Frontend zur Benutzerinteraktion, einem Backend zur Verarbeitung der Anfragen sowie einer lokalen Datenbank zur temporären Datenhaltung.

Das Frontend wurde mit React, JavaScript und CSS implementiert, um eine erfolgreiche Benutzerinteraktion zu ermöglichen. Über das Frontend können Professor:innen sowie andere Personen mit wissenschaftlichen Publikationen BibTeX-Dateien importieren, deren Einträge bearbeiten oder löschen und anschließend als XML-Datei exportieren, um sie in OPUS zu importieren. Das Backend wurde mit Node.js und Express realisiert und stellt über definierte Endpoints die Schnittstelle zwischen Frontend und Datenbank bereit. Es übernimmt die Verarbeitung der Import- und Exportvorgänge sowie die Datenpersistenz. Die lokale Datenspeicherung erfolgt im Local Storage des Systems, wodurch keine externe Datenbankverbindung notwendig ist. Dies vereinfacht die Architektur, ermöglicht aber dennoch das Zwischenspeichern von Daten zwischen verschiedenen Arbeitsschritten innerhalb einer Sitzung.

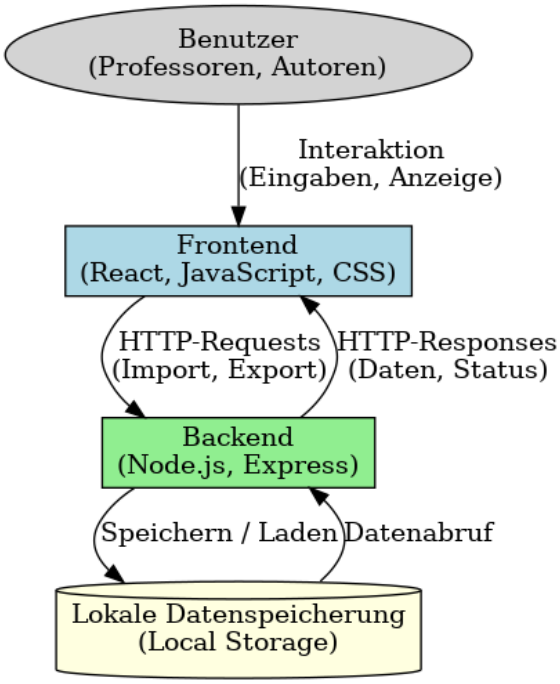

Abbildung 4.1: Systemarchitektur des entwickelten Assistenzsystems

4.2 Technologie-Stack

Das Assistenzsystem wurde mit moderne und effektive Technologie Stack entwickelt, der eine einfache Wartung ermöglicht. Der Stack umfasst sowohl Frontend und Backend Technologien sowie die lokalen Datenspeicherung.

4.2.1 Frontend

Für die Benutzeroberfläche wurde das React-Framework in Kombination mit JavaScript eingesetzt. React ermöglicht eine komponentenbasierte Entwicklung. Dadurch sind einzelne Funktionseinheiten wie Importformulare, Bearbeitungsansichten und Exportfunktionen klar voneinander getrennt und wiederverwendbar. Um eine ansprechende und benutzerfreundliche Oberfläche zu gewährleisten, wurden die Gestaltung und das Layout mit CSS umgesetzt.

4.2.2 Backend

Die serverseitige Logik basiert auf Node.js mit dem Express-Framework. Durch die Verwendung von Node.js wird eine ereignisgesteuerte und asynchrone Verarbeitung ermöglicht, wodurch die Verarbeitung von Benutzeraktionen beschleunigt werden kann. Express bietet eine klare Struktur für die Definition von Endpoints, über die das Frontend mit dem Backend kommuniziert. Die vorliegende Arbeit befasst sich mit REST-Endpoints, die insbesondere für die Steuerung von Import- und Exportprozessen sowie für den Zugriff auf die lokale Datenspeicherung genutzt werden.

4.2.3 Datenhaltung

Für die Datenspeicherung wird der Local Storage des Browsers verwendet. Dieser ermöglicht eine einfache, eingeschränkt persistente Ablage der importierten BibTeX-Dateien, die auch nach dem Schließen des Browsers erhalten bleibt. Da die Daten lediglich als temporäre Zwischenablage dienen und anschließend im OPUS-Format exportiert werden, ist der Einsatz einer vollwertigen Datenbank nicht erforderlich. Diese Entscheidung reduziert die Systemkomplexität und genügt den Anforderungen des Systems.

4.3 Implementierung der Features

4.3.1 Parsen und API-Kommunikation

Zu Beginn wurde ein Parser-Service implementiert, der eine hochgeladene `.bib`-Datei einliest und deren Inhalte in ein strukturiertes JSON-Format umwandelt. Dies ermöglicht eine einfache Weiterverarbeitung der Literaturdaten im Frontend. Dabei wurden Bibliotheken genutzt, um die einzelnen Felder wie Titel, Autoren, Jahr und Publikationsart korrekt zu extrahieren.

Für die Kommunikation zwischen Backend und Frontend wurden mehrere REST-API-Endpunkte erstellt. Diese liefern die geparsten Daten im JSON-Format, nehmen Änderungen vom Frontend entgegen und stellen Funktionen zum Speichern und Löschen bereit. Die Endpunkte wurden so gestaltet, dass sie sowohl einzelne Einträge als auch komplette Datensätze verarbeiten können.

4.3.2 Importieren, Bearbeiten und Trennen der Einträge

Im Frontend wurde ein Button „Importieren“ implementiert, der es dem Benutzer ermöglicht, eine lokale `.bib`-Datei auszuwählen. Nach der Auswahl wird die Datei an das Backend gesendet, dort geparst und die verarbeiteten Daten werden im Frontend angezeigt.

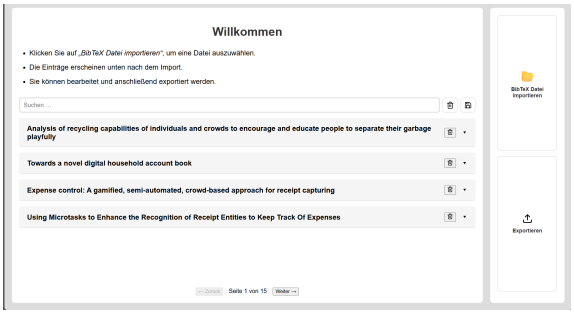

Abbildung 4.2: Benutzeroberfläche des Import-Buttons und Trennen der Einträge

Jeder Eintrag kann direkt in der Oberfläche bearbeitet werden. Änderungen an Feldern wie Titel, Autoren oder Jahr werden sofort in der aktuellen Ansicht übernommen und können später gespeichert werden. Eine spezielle Logik trennt die Autorenangaben aus der `.bib`-Datei in einzelne Felder. Dies erlaubt eine saubere Darstellung und erleichtert die Bearbeitung einzelner Namen.

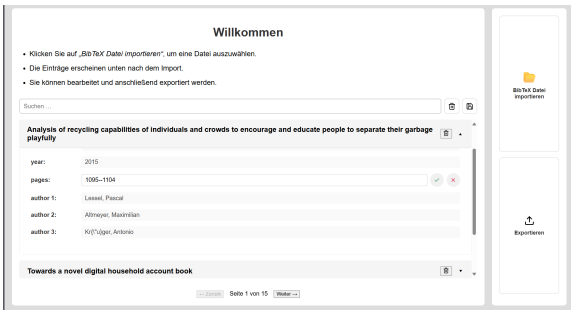

Abbildung 4.3: Bearbeitung eines Literatur-Eintrags

4.3.3 Anzeige und Navigation der Einträge

Um die Übersichtlichkeit zu erhöhen, werden die einzelnen Literatur-Einträge in einer aufklappbaren Struktur dargestellt. So kann der Benutzer gezielt die Details eines Eintrags einsehen oder ausblenden.

Um die Navigation bei vielen Einträgen zu vereinfachen, wurde eine Paginierung implementiert. Diese teilt die Liste der Einträge in mehrere Seiten auf und reduziert so die Ladezeit und die visuelle Überlastung.

4.3.4 Such- und Löschfunktionen

Eine Suchkomponente ermöglicht das Filtern von Einträgen anhand von Schlagwörtern. Dies erleichtert das Auffinden bestimmter Publikationen in großen Datenbeständen.

Mit dem Button „Alle löschen“ können sämtliche Einträge in der aktuellen Ansicht entfernt werden. Diese Funktion ist besonders nützlich, wenn ein neuer Datensatz importiert werden soll.

4.3.5 Speicherung und Export der Daten

Über den Local Storage des Browsers werden Änderungen lokal gespeichert, sodass diese auch nach einem Neuladen der Seite erhalten bleiben. Diese Funktion dient als Zwischenspeicher, bevor die Daten endgültig exportiert oder gespeichert werden. Der Button „Speichern“ ermöglicht es, alle vorgenommenen Änderungen dauerhaft zu sichern. Dabei werden die aktuellen Daten aus dem Local Storage übernommen und entweder im Backend oder als Exportdatei gespeichert.

Zum Schluss bietet der Button „Exportieren“ die Möglichkeit, alle gespeicherten Daten in eine XML-Datei umzuwandeln und herunterzuladen.Diese Datei kann später in OPUS importiert werden .

5 Diskussion

Das entwickelte Assistenzsystem zur Übertragung wissenschaftlicher Arbeiten in das OPUS-Publikationssystem erfüllt die wesentlichen Zielsetzungen des Projekts. Professoren und Professorinnen haben die Möglichkeit, BibTeX-Dateien zu importieren, die enthaltenen Einträge zu bearbeiten oder zu löschen und im Anschluss im OPUS-Format zu exportieren. Die Bedienung erfolgt vollständig über eine Weboberfläche, was eine einfache und intuitive Nutzung ermöglicht. Durch den gewählten Technologie-Stack aus React, Node.js und Local Storage konnte ein leichtgewichtiger Prototyp realisiert werden, der ohne zusätzliche Server- oder Datenbankinfrastruktur auskommt.

Die im Projekt definierten technischen Ziele wurden weitgehend erreicht. Sämtliche Kernfunktionen, nämlich Import, Bearbeitung, Suche, Speicherung, Löschung und Export von BibTeX-Daten, sind implementiert und funktionieren zuverlässig innerhalb der gewählten Systemarchitektur. Im Zuge der Optimierung der Benutzerfreundlichkeit wurden ergänzend Funktionen wie eine Echtzeitvorschau der Einträge und eine automatische Trennung einzelner Felder realisiert. Die vorliegende Umsetzung hat das Potenzial, den Publikationsprozess zu erleichtern und die mit manueller Eingabe und Formatierung häufig auftretenden Fehler zu reduzieren.

Während des Implementierungsprozesses traten jedoch verschiedene Herausforderungen auf. Die Analyse der vorliegenden Daten ergab, dass das Parsen von .bib-Dateien im Backend mit einer hohen Komplexität verbunden ist. Dies ist auf die Berücksichtigung unterschiedlicher Formatvarianten und Sonderzeichen zurückzuführen. Auch die dynamische Bearbeitung einzelner Autoren im Frontend stellte eine technische Herausforderung dar, da hierfür eine flexible und gleichzeitig stabile Datenstruktur erforderlich war. Darüber hinaus waren organisatorische Schwierigkeiten bei der parallelen Arbeit an Frontend und Backend zu verzeichnen, was eine präzise Abstimmung im Team erforderlich machte. Die zuvor genannten Probleme konnten durch spezifische Anpassungen in der Parserlogik, erweiterte Validierungsmechanismen und transparente Kommunikationswege innerhalb des Teams weitgehend gelöst werden.

Insgesamt hat sich die gewählte Architektur als zweckmäßig erwiesen, um die gestellten Anforderungen mit überschaubarem Entwicklungsaufwand umzusetzen. Der modulare Aufbau mit klarer Trennung von Frontend, Backend und Datenspeicherung ermöglicht eine einfache Erweiterbarkeit, beispielsweise durch die Integration einer Benutzeranmeldung, von Mehrsprachigkeit oder durch die direkte Anbindung an OPUS über eine API. Die Entwicklung hat zudem gezeigt, dass ein enger Austausch im Team und eine klare Aufgabenteilung für die termingerechte Umsetzung entscheidend waren.

Für eine zukünftige Weiterentwicklung wäre es sinnvoll, neben der Verbesserung der Datenpersistenz auch Funktionen wie Mehrbenutzerunterstützung, erweiterte Suchfilter oder eine direkte Online-Veröffentlichung zu integrieren. Durch diese Erweiterungen würde sich das System von einem reinen Übertragungswerkzeug zu einer vollwertigen Publikationsplattform entwickeln und den Mehrwert für die Zielgruppe nochmals deutlich steigern.

6 Manual

6.1 Starten der Projekts

6.2 Bedienung des Projekts

7 User Test

7.1 Methode und Ablauf

7.2 Ergebnisse

7.3 Interpretation der Ergebnisse und deren Auswirkungen

8 Fazit und Ausblick

Literatur

[1] Donald E. Knuth. „Computer Programming as an Art“. In: *Communications of the ACM* 17.12 (1974), S. 667–673.

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungsverzeichnis

Anhang

A Erster Abschnitt des Anhangs

In den Anhang gehören „Hintergrundinformationen“, also weiterführende Information, ausführliche Listings, Graphen, Diagramme oder Tabellen, die den Haupttext mit detaillierten Informationen ergänzen.

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Kolophon

Dieses Dokument wurde mit der LaTeX-Vorlage für Abschlussarbeiten an der htw saar im Bereich Informatik/Mechatronik-Sensortechnik erstellt (Version 2.25, August 2024). Die Vorlage wurde von Yves Hary und André Miede entwickelt (mit freundlicher Unterstützung von Thomas Kretschmer, Helmut G. Folz und Martina Lehser). Daten: (F)10.95 – (B)426.79135pt – (H)688.5567pt