* Introduciton schreiben, sonst ohne kontext, mit problem statement, was in den chaptern kommt 10
* Rq in introduction schreiben 1
* Related work: mehr details bei rdf und shacl 10
  + Rdf: formal aufschreiben s p o. s ist entweder resource oder ... 5
  + Shacl mehr details 5
* Slr nach themen gruppieren, Überschriften umbenennen 5
* Background -> state of the art umbenennen 1
* Qse noch genauer beschreiben, besonders am schluss, begriffe, definitionen 5
* Bilder in pdf statt in png. Pdflatex 4
* Referenzen direkt nach wort. Bei projekten. Damit man es direkt zuordnen kann 3
* In methods bezug nehmen auf rq in introduction 2
* Methoden lassen, sie lest es sich das nächste mal durch
* 4: requirements and functionalities: was braucht man, verbindung zu rq, anforderungsanalyse 20
  + System architecture: was moment drinnen ist 10
  + Es sollen dann später 3 teile werden: requirements + functionalities, beitrag zur wissenschaft (sie überlegt sich noch namen), implementation 1
  + Für interface z.B. Client Server Architektur In requirements
  + Interface -> trennen -> eigenes unterkapitel in requirements: nicht webapp schreiben, anderen namen dafür überlegen (marketing)
  + Implementation: manche kapitel sind kurz (z.B. Version of QSE), screenshots teil dann lange
  + Bei screenshots teil sollte wirklich nichts mehr erklärt werden, nur beispiel
  + Text nur anpassen, sollte schon gut so passen
* In bilder highlights, box a, rot einzeichnen, wenn ein genauer teil beschrieben ist 1
* Webapp im interface schritte machen statt use-case diagramm, in functionalities 4
  + Vorher abstrakt dann konkret
* Datenbank erwähnen in architecture, dass es eine gibt. Später bei implementation dann das erd hinschreiben 6
* Ram size und cpu speed nachfragen und hineinschreiben 2
* Motivation evolving kowledge graphs in einleitung. Es gibt verschiedene versionen....
* Von mir: Sollte ordentliche Diagramme verwenden 10
* RQ4 schreiben 15
* Notizen aus Scan umschreiben 30 h

Fragen:

* Passt Titel jetzt?
* Was ist bie knowledge graphs auf pdf seite 19 mit Vituuouso gemeint?
* Bei SHACL seite 20 Introduce "target" and say that you focus on one of the cases?
* Seite 21: Evolving Knowledge unterstrichen -> kleinschreibung?
* Seite 22: Strich bei QSE?
* Seite 23: pdf nt nngor so -> kann ich nicht lesen
* Methoden: present oder past tence verwenden?
* List of figures -> war so im Template, wo hätte sie es hingegeben?

Todo im ganzen Text:

* Bilder PDF
* Beistrich vor And
* Groß/Kleinschreibung
* Referenzen direkt nach Wort
* Text nach Subkapitel
* Namen von WebApp austausschen

\begin{definition}

(Shape graph \cite{rabbani\_extraction\_2023}). A shape graph $S$ contains node shapes $N$, with \(\langle s, \tau\_{n}, \Phi\_{n} \rangle \in N\) where $s$ is the subject IRI of the node shape (or the name), \(\tau\_{n}\in C\) is the target class, and $\Phi\_{n}$ are the property shapes, in the form \(\phi\_{n}:\langle \tau\_{p}, T\_{p}, C\_{p}\rangle\). $\tau\_{p}$ is the target property, defined with "sh:path" and \(T\_{p} \subset R\) contains, in case of the node kind of $\tau\_{p}$ "sh:Literal", an IRI describing the literal type e.g. "xsd:string". In case of the node kind of $\tau\_{p}$ "sh:IRI", a set of IRIs is provided. $C\_{p}$ is a pair \((n,m)\in\mathbb{N}\times(\mathbb{N}\cup\{\infty\})\) with \(n\leq m\), which describes the min and max cardinality constraints.

Check, if the property shape has mutliple constraints linked together with SHACL-Or. If

Another aspect of performance optimization involved determining the most efficient method for extracting text segments for each shape from QSE. There were multiple approaches considered on how to use the information provided by QSE which includes a list of objects and the final SHACL file. However, inconsistencies between these objects were often encountered. In the final version, a combination of the two artifacts was used, the list of objects was used internally and the SHACL file was used for text generation. \newline

One attempt involved reading the SHACL file into an RDF4J or Jena model and then filtering it using SPARQL or provided methods to retrieve only the relevant text for each shape. Unexpectedly, this approach included dealing with blank nodes. The generated SHACL shapes included objects like SHACL-or and SHACL-in which are internally stored with blank nodes. To retrieve this full information, an algorithm had to be designed to recursively load the triples in the blank nodes from the model. However, this approach in general proved to be inefficient, therefore the final solution was to use regex to extract the shape segments directly from the SHACL file. Although this approach has several disadvantages, such as potential inconsistencies in recreating SHACL-In items and items in a SHACL-Or List, it is notably faster. However, QSE uses the TurtlePrettyFormatter to format SHACL files, therefore the general, consistent structure of the text is ensured which makes the comparison of shapes easy \cite{textor\_atextorturtle-formatter\_2023}. In the end, the support and confidence triples were removed by converting the text segment into a Jena file and then filtering the statements. To address the issue of SHACL-Or and SHACL-in items, algorithms were developed to order these text lines alphabetically.

\newline

The mapping between the shapes in different QSE runs is based on the name generated by QSE.

However, inconsistencies between these objects were often encountered.

An alternative approach would be to present the shape as an object with fixed properties, similar to the internal output format used by QSE. However, given the potential combinations and edge cases that could arise, it was determined that maintaining the output in its original SHACL Turtle format would be more practical