Diplomarbeit Yanick Arn

Schweizerische Fachschule TEKO

DOMAIN-DRIVEN MODELLING IN THE WEB

Projektteam:	Yanick Arn, Projektleiter
Auftraggeber/Firmenbetreuer:	Stefan Kapferer / mimacom ag
Schulischer Betreuer	Gregor von Flüe
Höhere Fachschule:	TEKO Olten
Ausbildungsstart	28. Oktober 2019
Abgabedatum	24. Oktober 2022

CONTEXT MAPPER

Management Summary

Das vorliegende Dokument befasst sich im Rahmen der Diplomarbeit für den Abschluss der höheren Fachschule Techniker mit dem Thema «DOMAIN-DRIVEN MODELLING IN THE WEB».

Die heutige Arbeitswelt in der Softwareentwicklung wird immer agiler. Rollen vermischen sich und es gibt eine steigende Anzahl an Tools. Die Informatikwelt und der Austausch zwischen Kunde und Entwickler werden immer komplexer. Es entstehen Ketten von Tools, welche voneinander abhängig sind und oftmals nur der Entwickler hat.

In dem vorliegenden Dokument wird beschrieben, wie man eines der erwähnten Tools, den «Context Mapper», von der Abhängigkeit einer Entwicklungsumgebung abkoppelt. Da gerade bei «Domain-Driven Design» das Gleiche Verständnis über die gesamte Organisation wichtig ist, sollen auch Rollen ausserhalb der Entwicklung dieses Tool zur Verfügung haben.

Die Arbeit befasst sich mit der Erstellung einer Webapplikation, welche das Schreiben von «Context Map» Diagrammen mithilfe einer von «Eclipse Xtext» erstellten domänenspezifische Sprache ermöglicht.

Inhaltsverzeichnis

Qualifikationsprofil

Yanick Arn
Dipl. Techniker HF

Qualifikationsprofil

Dipl. Techniker HF, Applikationsentwicklung

Menschen führen *Prozess 1*	Lernende & Bachelorstudenten einführen und coachen sowie Projekte von ihnen führen.
Entscheidungen fällen *Prozess 2*	Varianten für Lösungsmöglichkeiten von diversen Problemen untersucht und sich für eine davon entschieden.
Projekte planen und leiten *Prozess 3*	Ein Java Spring Projekt als Lead-Engineer geplant und mit dem Team bis zum MVP ausgeführt.
Wirkungsvoll präsentieren und kommunizieren *Prozess 5*	Durch Verwendung von diversen Medien wie PowerPoint, Confluence und Livedemonstrationsmitteln wie Flipcharts, Whiteboards dem Auftraggeber Lösungsansätze zu Herausforderungen präsentiert und sich auf eine begrenzte Auswahl geeinigt.
Unternehmensprozesse verstehen und mitgestalten *Prozess 6*	Das bisherige Einführungsprogramm für Lernende so weit neugestaltet, damit eine schnelle und effiziente Mitarbeit in produktiven Projekten möglich ist.
Geschäftsziele erreichen *Prozess 7*	Diverse Projekte innerhalb der erforderlichen Zeit realisiert und an Kunden verkauft.

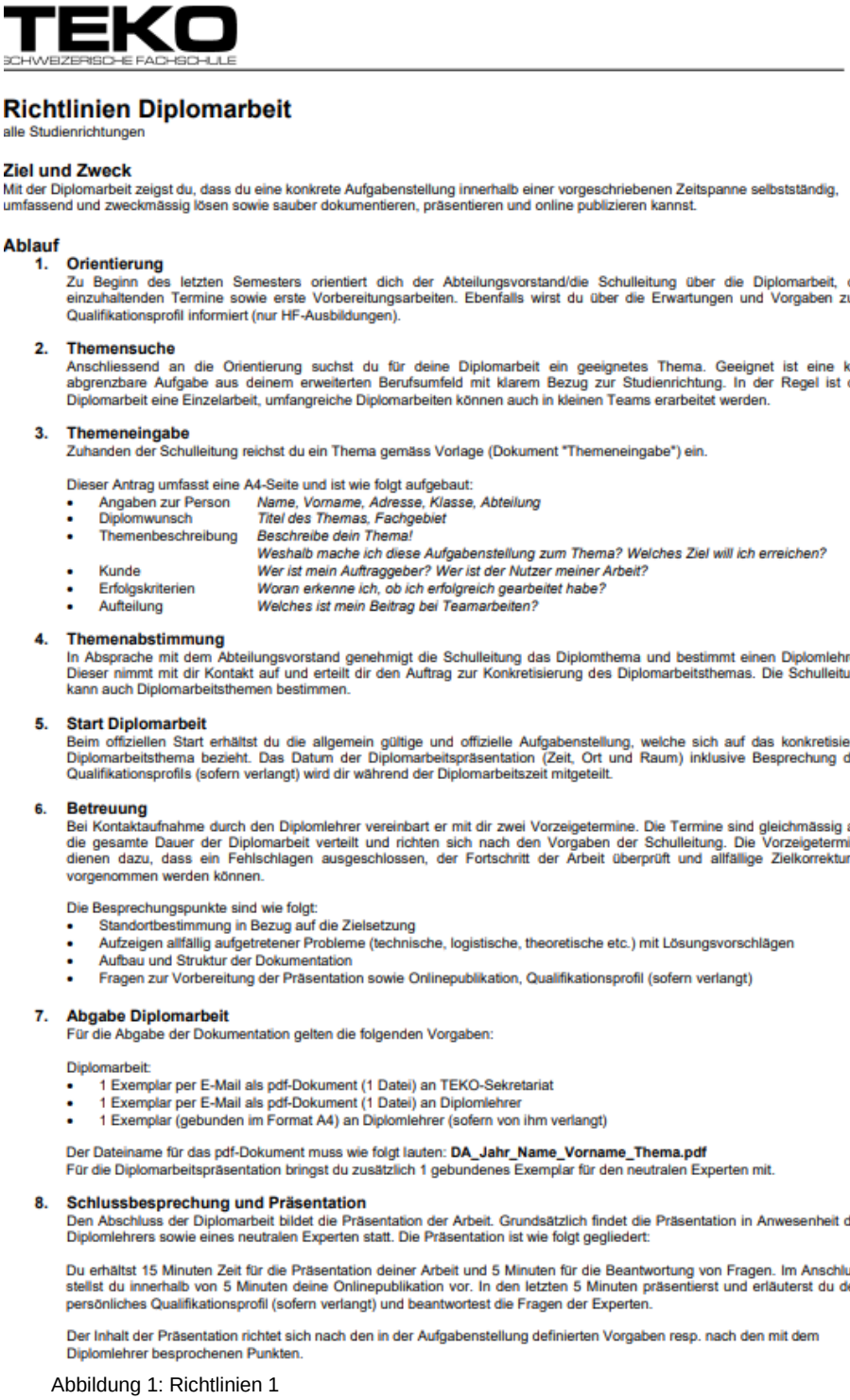
Umfeld berücksichtigen *Prozess 8*	Produktive Arbeitsumgebung für das Team eingerichtet, sowie den Teamzusammehalt gestärkt durch kleine Events.
Probleme analysieren und lösen *Prozess 9*	Probleme in jeglichen Bereichen analysiert und wenn nötig mithilfe von Medien wie Whiteboards/Flipcharts dokumentiert, um Algorithmen für bestimmte Softwareherausforderungen zu entwickeln
Sich persönlich weiterentwickeln *Prozess 10*	Fehlendes Wissen selbständig durch die Hilfe vom Internet oder Kursen angeeignet und eingesetzt.
Business Anforderungen analysieren und bestimmen *Prozess 12*	Mithilfe von SCRUM diverse Anforderungen analysiert und Anforderungen sowie Akzeptanzkriterien für die jeweilige Anforderungen konzipiert
Datenschutz und Datensicherheit gewährleisten *Prozess 14*	Diverse Authentifikationssysteme implementiert, sowie nur innerhalb eines bestimmten Netzwerkes verfügbar gemacht. «Secure»-Umgebungen für diverse Projekte aufgebaut.
Softwarearchitektur analysieren und bestimmen *Prozess 15*	Softwarearchitekturen anhand Anforderungen und bereits bestehender Infrastruktur analysiert und entsprechend bestimmt, um diverse Projekte von Scratch hochzuziehen
Applikationen entwickeln, Programme erstellen und testen *Prozess 16*	Eine Webapplikation mit einem Java Backend und einem Angular Frontend getestet, containerisiert in der AWS Cloud deployt und vollständig lauffähig gemacht
Spezifische Hardware programmieren *Prozess 20*	Badge Reader programmiert und einen Newsletter von einer Migration an die jeweilige Person über Mail zu schicken, damit diese Person alle nötigen Informationen hat. Arduino für die Messung von Temperatur, Luftfeuchtigkeit und ppm programmiert.

1 Projektinitialisierung

In diesem Kapitel sind die wichtigsten Informationen zum Projekt vorzufinden. Diese beinhalten zum Grossteil Daten aus dem Pflichtenheft.

1.1 Richtlinien

Anders als bei einem Projektauftrag, ist die Diplomarbeit nicht vollständig an ein Thema gebunden und orientiert sich an Richtlinien.



TEKO
SCHWEIZERISCHE FACHSCHULE

Richtlinien Diplomarbeit
alle Studienrichtungen

Ziel und Zweck
Mit der Diplomarbeit zeigst du, dass du eine konkrete Aufgabenstellung innerhalb einer vorgeschriebenen Zeitspanne selbstständig, umfassend und zweckmässig lösen sowie sauber dokumentieren, präsentieren und online publizieren kannst.

Ablauf

1. **Orientierung**
   Zu Beginn des letzten Semesters orientiert dich der Abteilungsvorstand/die Schulleitung über die Diplomarbeit, die einzuhaltenden Termine sowie erste Vorbereitungsarbeiten. Ebenfalls wirst du über die Erwartungen und Vorgaben zum Qualifikationsprofil informiert (nur HF-Ausbildungen).

2. **Themensuche**
   Anschliessend an die Orientierung suchst du für deine Diplomarbeit ein geeignetes Thema. Geeignet ist eine klar abgrenzbare Aufgabe aus deinem erweiterten Berufsumfeld mit klarem Bezug zur Studienrichtung. In der Regel ist die Diplomarbeit eine Einzelarbeit, umfangreiche Diplomarbeiten können auch in kleinen Teams erarbeitet werden.

3. **Themeneingabe**
   Zuhanden der Schulleitung reichst du ein Thema gemäss Vorlage (Dokument "Themeneingabe") ein.

   Dieser Antrag umfasst eine A4-Seite und ist wie folgt aufgebaut:
   - Angaben zur Person — *Name, Vorname, Adresse, Klasse, Abteilung*
   - Diplomwunsch — *Titel des Themas, Fachgebiet*
   - Themenbeschreibung — *Beschreibe dein Thema!*
     Weshalb mache ich diese Aufgabenstellung zum Thema? Welches Ziel will ich erreichen?
   - Kunde — *Wer ist mein Auftraggeber? Wer ist der Nutzer meiner Arbeit?*
   - Erfolgskriterien — *Woran erkenne ich, ob ich erfolgreich gearbeitet habe?*
   - Aufteilung — *Welches ist mein Beitrag bei Teamarbeiten?*

4. **Themenabstimmung**
   In Absprache mit dem Abteilungsvorstand genehmigt die Schulleitung das Diplomthema und bestimmt einen Diplomlehrer. Dieser nimmt mit dir Kontakt auf und erteilt dir den Auftrag zur Konkretisierung des Diplomarbeitsthemas. Die Schulleitung kann auch Diplomarbeitsthemen bestimmen.

5. **Start Diplomarbeit**
   Beim offiziellen Start erhältst du die allgemein gültige und offizielle Aufgabenstellung, welche sich auf das konkretisierte Diplomarbeitsthema bezieht. Das Datum der Diplomarbeitspräsentation (Zeit, Ort und Raum) inklusive Besprechung des Qualifikationsprofils (sofern verlangt) wird dir während der Diplomarbeitszeit mitgeteilt.

6. **Betreuung**
   Bei Kontaktaufnahme durch den Diplomlehrer vereinbart er mit dir zwei Vorzeigetermine. Die Termine sind gleichmässig auf die gesamte Dauer der Diplomarbeit verteilt und richten sich nach den Vorgaben der Schulleitung. Die Vorzeigetermine dienen dazu, dass ein Fehlschlagen ausgeschlossen, der Fortschritt der Arbeit überprüft und allfällige Zielkorrekturen vorgenommen werden können.

   Die Besprechungspunkte sind wie folgt:
   - Standortbestimmung in Bezug auf die Zielsetzung
   - Aufzeigen allfällig aufgetretener Probleme (technische, logistische, theoretische etc.) mit Lösungsvorschlägen
   - Aufbau und Struktur der Dokumentation
   - Fragen zur Vorbereitung der Präsentation sowie Onlinepublikation, Qualifikationsprofil (sofern verlangt)

7. **Abgabe Diplomarbeit**
   Für die Abgabe der Dokumentation gelten die folgenden Vorgaben:

   Diplomarbeit:
   - 1 Exemplar per E-Mail als pdf-Dokument (1 Datei) an TEKO-Sekretariat
   - 1 Exemplar per E-Mail als pdf-Dokument (1 Datei) an Diplomlehrer
   - 1 Exemplar (gebunden im Format A4) an Diplomlehrer (sofern von ihm verlangt)

   Der Dateiname für das pdf-Dokument muss wie folgt lauten: **DA_Jahr_Name_Vorname_Thema.pdf**
   Für die Diplomarbeitspräsentation bringst du zusätzlich 1 gebundenes Exemplar für den neutralen Experten mit.

8. **Schlussbesprechung und Präsentation**
   Den Abschluss der Diplomarbeit bildet die Präsentation der Arbeit. Grundsätzlich findet die Präsentation in Anwesenheit des Diplomlehrers sowie eines neutralen Experten statt. Die Präsentation ist wie folgt gegliedert:

   Du erhältst 15 Minuten Zeit für die Präsentation deiner Arbeit und 5 Minuten für die Beantwortung von Fragen. Im Anschluss stellst du innerhalb von 5 Minuten deine Onlinepublikation vor. In den letzten 5 Minuten präsentierst und erläuterst du dein persönliches Qualifikationsprofil (sofern verlangt) und beantwortest die Fragen der Experten.

   Der Inhalt der Präsentation richtet sich nach den in der Aufgabenstellung definierten Vorgaben resp. nach den mit dem Diplomlehrer besprochenen Punkten.

Abbildung 1: Richtlinien 1

TEKO *Richtlinien Diplomarbeit*

9. Onlinepublikation

Alle HF-Diplomanden sind verpflichtet, das Ergebnis ihrer Arbeit (inkl. Modelle etc.) online zu publizieren. Die Onlinepublikation erfolgt auf einer von der TEKO zur Verfügung gestellten Plattform und dient dazu, das Ergebnis einem breiten Publikum nachhaltig zugänglich zu machen. Die Vorstellung der Onlinepublikation ist fester Bestandteil der Diplomarbeitspräsentation.
Der Zugang zur Plattform wird im Verlauf der Diplomarbeit durch die Schulleitung bekanntgegeben.

Die Publikation umfasst folgende Inhalte: Kurzbeschreibung des Ergebnisses resp. des Produktes, Nutzen / Mehrwert für den Auftraggeber, Veranschaulichung mittels frei wählbarer Elemente (z.B. Fotos, Film etc.). Die Anleitung zur Benützung und zum Handling der Plattform ist auf dem Extranet im Abschnitt Diplomarbeit verfügbar.

Vertrauliche Daten in der Diplomarbeit

Vertrauliche Diplomarbeiten müssen durch die Schulleitung bewilligt werden und sind auf der Titelseite mit dem Vermerk "vertraulich" zu versehen. Bei vertraulichen Arbeiten erhältst du die Exemplare direkt vom Diplomlehrer nach der Präsentation zurück. Die Schulleitung unterzeichnet keine Vertraulichkeitsvereinbarung irgendwelcher Art.

Äussere Form, Aufwand

Die Diplomarbeit verfasst du mit einem Textverarbeitungsprogramm im Format A4, einseitig beschrieben, fortlaufend nummeriert, gebunden oder in einem Ordner (keine Zeigetaschen). Je nach Ausbildung ist etwa mit folgendem Arbeitsaufwand zu rechnen:

• Techniker HF	150-250 Std.	6 Wochen
• Betriebswirtschafter HF	150-250 Std.	6 Wochen
• Nachdiplomstudien NDS HF	150-250 Std.	6 Wochen

Die Diplomarbeit umfasst eine schriftliche Dokumentation und je nach Aufgabenstellung bzw. Fachrichtung einen zeichnerisch-konstruktiven und / oder experimentellen Teil (z.B. Modell, Prototyp, Labor-Aufbau etc.).

Sofern Arbeitsunterlagen aus deiner Unternehmung verwendet werden, solltest du das Einverständnis des betreffenden Vorgesetzten einholen. Allfällig benützte Quellen sind lückenlos aufzuführen.

Gliederung der Diplomarbeit

Im Rahmen des Studiums hast du ein breites Grundlagenwissen im Projektmanagement aufgebaut und in Projekt- und Semesterarbeiten angewendet. Die Dokumentation der Diplomarbeit orientiert sich an denselben Grundsätzen.

Die Arbeit soll beim Leser den Eindruck einer sinnvollen und geschlossenen Arbeit hinterlassen. Es muss ein "roter Faden" ersichtlich sein. Der chronologische Ablauf soll folgende Struktur aufweisen:

- Deckblatt: Titel der Diplomarbeit, Name der Schule, Name des Diplomanden, Ausbildung und Jahr, wenn vertraulich: Vermerk "vertraulich" auf der Titelseite
- Inhaltsverzeichnis
- Management Summary, max. 2 A4-Seiten
- Kurzer beruflicher Lebenslauf
- Aussagekräftiges Qualifikationsprofil (sofern verlangt, max. 2 A4-Seiten). Die Form ist grundsätzlich frei wählbar. Jedoch muss der Bezug zur Studienrichtung basierend auf dem Rahmenlehrplan klar erkennbar sein.
- Aufgabenstellung / Zieldefinition
- Terminplan SOLL/IST
- Lösung der Aufgabe: Berichte / Programme / Ablaufpläne / Lösungswege / Entwürfe / Zeichnungen / Modelle / usw.
- Reflexion Weg zum Ziel sowie "Lessons learnt"
- Persönliches Schlusswort, Verdankungen und Eigenständigkeits-Erklärung
- Anhang: Pflichtenheft, Vollständige Quellenangaben, Literaturverzeichnis

Bewertung der Diplomarbeit

Dein Diplomlehrer bewertet deine Diplomarbeit anhand eines vorgegebenen Beurteilungsrasters (Schwierigkeitsgrad, Projektinitialisierung und -planung, Realisierung, Dokumentation, Präsentation, Onlinepublikation) in Abstimmung mit dem Diplomexperten. Bei Diplomarbeiten, welche im Auftrag eines Betriebes gemacht werden, fliesst die Bewertung des betrieblichen Fachexperten in angemessener Form in die Gesamtbeurteilung ein. Bei Gruppenarbeiten kann jeder Diplomand einzeln bewertet werden.

Diplomierung

Zur Diplomierung wirst du zugelassen, wenn du bei der Diplomarbeit mindestens die Note 4.0 erreicht hast und sämtliche Verpflichtungen gegenüber der TEKO erfüllt sind. Die Arbeit kann einmal (jedoch mit einem neuen Thema) wiederholt werden.
Du bist selber für die Einhaltung sämtlicher Termine und Vorgaben verantwortlich. Unbegründet zu spät eintreffende Arbeiten gelten als nicht eingereicht; sie müssen mit neuem Thema wiederholt werden.

Fristerstreckung

Der festgelegte Abgabetermin der Diplomarbeit kann nur auf begründetes schriftliches Gesuch an die Schulleitung hinausgeschoben werden. Als Begründung für eine Fristerstreckung können die folgenden Punkte geltend gemacht werden: Nachgewiesene Krankheit (Arztzeugnis) und persönlicher Militärdienst (Marschbefehl).

Verwertungsrecht

Diplomarbeitskopien, welche im Besitze der TEKO bleiben, dürfen ohne Rückfrage für folgende Zwecke verwendet werden: Ausstellungen, Orientierungen, Anerkennungsverfahren und Rekursbeweise.

Beschwerden / Rekurs

Gegen promotionsrelevante Noten kannst du in begründeten Fällen innert 14 Tagen schriftlich Rekurs erheben. Erste Rekursinstanz ist die Schulleitung.

Abbildung 2: Richtlinien 2

1.2 Pflichtenheft

In diesem Kapitel sind die wichtigsten Kernpunkte des Pflichtenhefts aufgelistet. Diese beinhalten eine kurze Vorstellung der Firma, Details, wie man auf diesen Auftrag gekommen ist, sowie die Problemstellung.

1.2.1 mimacom ag

Die mimacom ag ist die erste Wahl, wenn ein Unternehmen Kunden durch Softwarelösungen erreichen und innovative Open Source Technologien für Ihren Unternehmenserfolg nutzen wollen. Das Unternehmen verfügt über eine breitgefächerte Branchenexpertise aus einer Vielzahl von Projekten in verschiedensten Anwendungsbereichen. Das Geschäft geht von der öffentlichen Verwaltung zur Telekommunikation, dem Handeln und vielen weiteren Bereichen hinaus.

Für jeden Anwendungsfall findet mimacom eine passende Lösung. Die Mitarbeiter werden oftmals in ein internes Team einer Firma eingebunden und nutzen von da ihr Knowhow optimal aus.

1.2.2 Ausgangslage

Bei der Strukturierung von Software und deren Komponenten gibt es diverse Herangehensweisen, Architekturen und Methoden. Eine sehr beliebte Methode ist das Domain-Driven Design oder kurz DDD, welches das Wissen und die Sprache der Fachdomäne ins Zentrum stellt. Domain-Driven Design stellt den Fokus auf die Verwendung von einer ubiquitären Sprache, damit in allen Bereichen, von Business, Requirements Engineer, bis zum Entwickler die gleiche Sprache gesprochen wird.

Um Domain-Driven Design optimal zu implementieren, gibt es Tools, um Architektur und Domänenmodelle visuell darzustellen. Eines dieser Tools ist das «Context Mapping» welches es erlaubt den Zusammenhang zwischen abgegrenzten Kontexten (Bounded Context) zu identifizieren.

Für die Generierung solcher «Context Maps» gibt es Frameworks wie den «Context Mapper» (contextmapper.org), welcher als Eclipse oder Visual Studio Code Plugin kommt und mithilfe einer domänenspezifischen Sprache (DSL) das Generieren solchen «Context Maps» ermöglicht.

1.2.3 Problemstellung

Da das Erstellen von «Context Maps» nicht nur den Entwicklern vorbehalten sein soll, braucht es eine Lösung, welche unabhängig von Entwicklungsumgebungen ist.
Eine Lösung, um dieses Problem zu beheben, ist es die Generierung von «Context Maps» auf dem Web mithilfe einer Webapplikation möglich zu machen.

Da diese Umsetzung in diesem spezifischen Kontext noch nicht existiert und keine genaue Dokumentation vorhanden ist, kam die Idee auf, dass sich das Ganze als Diplomarbeit eignen könnte.

Das Spezielle ist, dass die Arbeit im Auftrag der mimacom ag umgesetzt werden soll, aber das Endergebnis quellenoffen veröffentlich wird. Es handelt sich hierbei um einen Hybridprojekt welches sowohl firmenintern wie auch gemeinnützig genutzt werden kann.

1.3 Projektvertrag

Der Projektvertrag ist ein rechtsverbindliches Dokument zwischen dem Auftraggeber und dem Projektleiter. Deshalb ist diesem Dokument besondere Aufmerksamkeit zu schenken.

Projekttitel:	*DOMAIN-DRIVEN MODELLING IN THE WEB*

Projektauftraggeber:	Stefan Kapferer
Projektleiter:	Yanick Arn

Projektdaten			
Start:	*12.09.2022 00:00*	**Ende:**	*24.10.2022 23:59*

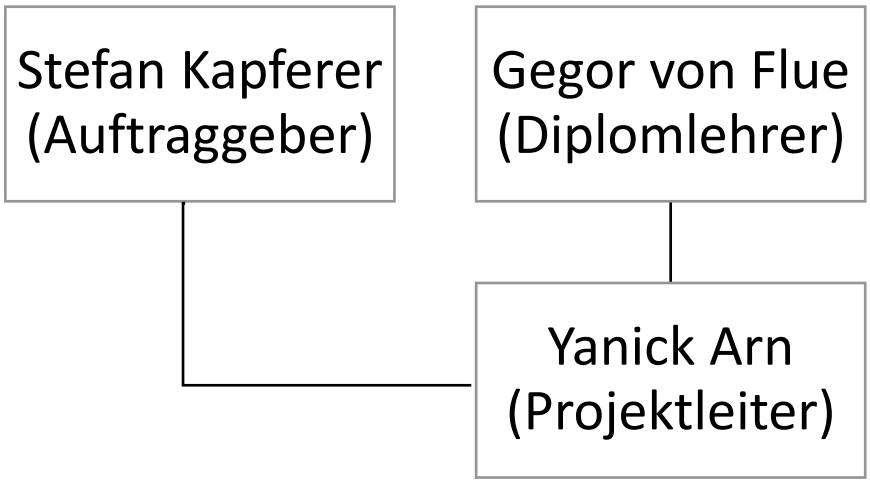
Projektbeschreibung	
Ausgangslage / Projektbegründung:	*Bei der Strukturierung von Software und deren Komponenten gibt es diverse Herangehensweisen, Architekturen und Methoden. Eine sehr beliebte Methode ist das Domain-Driven Design oder kurz DDD, welches das Wissen und die Sprache der Fachdomäne ins Zentrum stellt. Domain-Driven Design stellt den Fokus auf die Verwendung von einer ubiquitären Sprache, damit in allen Bereichen, von Business, Requirements Engineer, bis zum Entwickler die gleiche Sprache gesprochen wird.* *Um Domain-Driven Design optimal zu implementieren, gibt es Tools, um Architektur und Domänenmodelle visuell darzustellen. Eines dieser Tools ist das «Context Mapping» welches es erlaubt den Zusammenhang zwischen abgegrenzten Kontexten (Bounded Context) zu identifizieren.* *Für die Generierung solcher «Context Maps» gibt es Frameworks wie den «Context Mapper» (contextmapper.org), welcher als Eclipse oder Visual Studio Code Plugin kommt und es mit-hilfe einer domänenspezifischen Sprache (DSL) das Generieren solcher Maps ermöglicht.*
Sinn und Zweck / Nutzen:	*Die Generierung von «Context Maps» soll unabhängig von einer konfigurierten Entwicklungsumgebung möglich sein.*
Projektrichtziel:	*Domain-Driven Modelling mit dem Context Mapper via Webapplikation verfügbar machen*

Endergebnisse	**Erfolgskriterien**
Die Applikations-Oberfläche enthält ein Editor Feld mit einem Beispiel Code Segment	*Bei Aufrufen der URL lädt die Oberfläche mit einem Beispiel Code Segment im Editor Feld, welches wie ein Textfeld bearbeitbar ist. Das Beispiel Code Segment wird bei Erstaufruf als «Context-Mapper» Diagramm generiert und dargestellt.*
Ein Diagramm ist durch einen «Generate» Button generierbar.	*Wenn der «Generate» Button auf der Oberfläche gedrückt wird, und das Code Segment im Editor valide ist, wird ein aktualisiertes Diagramm auf der rechten Seite ausgegeben.*
Das momentan generierte Diagramm ist durch einen «Export» Button und einem Format Auswahlfeld exportierbar.	*Wenn ein vom Generator unterstütztes Exportformat ausgewählt und der «Export» Button gedrückt wird, startet der Download im jeweiligen Format*

Zielgenehmigung:	Die Ziele werden bewilligt. Datum: 11.09.2022 Unterschrift Auftraggeber: 

Projekttyp:	
☐ Routineprojekt ☐ komplexes Standardprojekt ☐ Potenzial- / Innovationsprojekt ☒ Pionierprojekt	**Begründung:** Die Entscheidung fiel auf ein Pionierprojekt, da die Aufgabenstellung sehr viele Möglichkeiten in der Umsetzung bietet und es komplexe Wirkungszusammenhänge zwischen Systemen geben wird.

Projektorganisation:



Organisationstyp:

Abbildung 3: Projektorganisation

Projektmitarbeiter:	**Name / Vorname / OE**	**Stellenprozent für Projekt**
Yanick Arn	*Yanick Arn / Projektleiter* *Stefan Kapferer / Auftraggeber* *Gregor von Flüe / Diplomlehrer*	100% - -
Steering Commitee: ☐ **Ja** ☒ **Nein**		
Sonstige Beteiligte:	-	

Projektplanung	
Projektphasen / Meilensteine:	*1. Projektinitialisierung & Planung* ♦ *MS1: Initialisierung & Planung abgeschlossen* *2. Projektrealisierung* ♦ *MS2: Abschluss MVP* ♦ *MS3: Abschluss Realisierung* *3. Projektabschluss* ♦ *MS4: Projekt abgeschlossen*

Projektentscheid:	☒ Das Projekt wird bewilligt. ☐ Das Projekt wird abgelehnt. Begründung: Datum: 11.09.2022 Unterschrift Auftraggeber: 

Tabelle 1: Projektvertrag

1.3.1 Zielscheibe

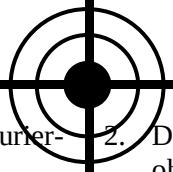
Richtziel: Domain-Driven Modelling mit dem Context Mapper via Webapplikation verfügbar machen

Endergebnisse

1. Es liegt eine schriftliche Projektdokumentation, im A4 Format als PDF, über DOMAIN-DRIVEN MODELLING IN THE WEB bis am 24.10.2022 um 23:59 mit folgenden Inhalten vor:
   1.1. Die Applikations-Oberfläche enthält ein Editor Feld mit einem Beispiel Code Segment
   1.2. Ein Diagramm ist durch einen «Generate» Button generierbar.
   1.3. Das momentan generierte Diagramm ist durch einen «Export» Button und einem Format Auswahlfeld exportierbar.

Kunde

Stefan Kapferer, mimacom ag



Sinn und Zweck

- Die Generierung von «Context Maps» soll unabhängig von einer konfigurierten Entwicklungsumgebung möglich sein.

Erfolgskriterien

2. Die Dokumentation entspricht den geforderten Richtlinien und ist ohne zeitlichen Verzug dem Verantwortlichen abgegeben.
   2.1. Bei Aufrufen der URL lädt die Oberfläche mit einem Beispiel Code Segment im Editor Feld, welches wie ein Textfeld bearbeitbar ist. Das Beispiel Code Segment wird bei Erstaufruf als «Context-Mapper» Diagramm generiert und dargestellt.
   2.2. Wenn der «Generate» Button auf der Oberfläche gedrückt wird, und das Code Segment im Editor valide ist, wird ein aktualisiertes Diagramm auf der rechten Seite ausgegeben.
   2.3. Wenn ein vom Generator unterstütztes Exportformat ausgewählt und der «Export» Button gedrückt wird, startet der Download im jeweiligen Format

2 Projektplanung

In diesem Kapitel sind die wichtigsten Planungen für die Realisation vorzufinden. Diese beinhalten Zeitpläne, Testpläne sowie eine allgemeine Risikoanalyse.

2.1 Vorgehensmodell

Die Planung setzt sich aus dem 4-Phasenmodell zusammen mit Teilen von HERMES.

2.2 Projektstrukturplanung

Im Rahmen der Projektstrukturplanung (kurz PSP) wird das Projekt in Arbeitspakete aufgeteilt, welche in der Summe die Endergebnisse des Projektes ergeben. Die Abbildung zeigt einen Projektstrukturplan als Strukturbaum, welcher nach dem 4-Phasenmodell aufgebaut ist.

Der Strukturplan befindet sich aufgrund der Grösse im Anhang.

2.3 Projektablaufplanung

Mithilfe des Projektstrukturplanes (PSP) wird nun ein detaillierter Projektablaufplan dargestellt, in welchem Arbeitspakete in logischer Reihenfolge in einen zeitlichen Ablauf gebracht werden. Für die Darstellung wird ein Balkendiagramm bzw. «Gnatt»-Diagramm verwendet.

Bei Krankheitsfällen werden die Termine um die jeweilige Zeit verschoben. Die Termine sind so geplant, dass sie mit einem täglichen Kleinaufwand oder bei einem Grossaufwand pro Tag gelöst werden können. Eine Verschiebung von wenigen Tagen ist daher nicht gefährlich für den Abschluss der Arbeit.

Da die Darstellung des Plans zu viel Platz einnimmt, ist der Gesamte Plan + «Critical Path» im Anhang zu finden.

Der weniger detaillierte Ablauf sieht folgendermassen aus:

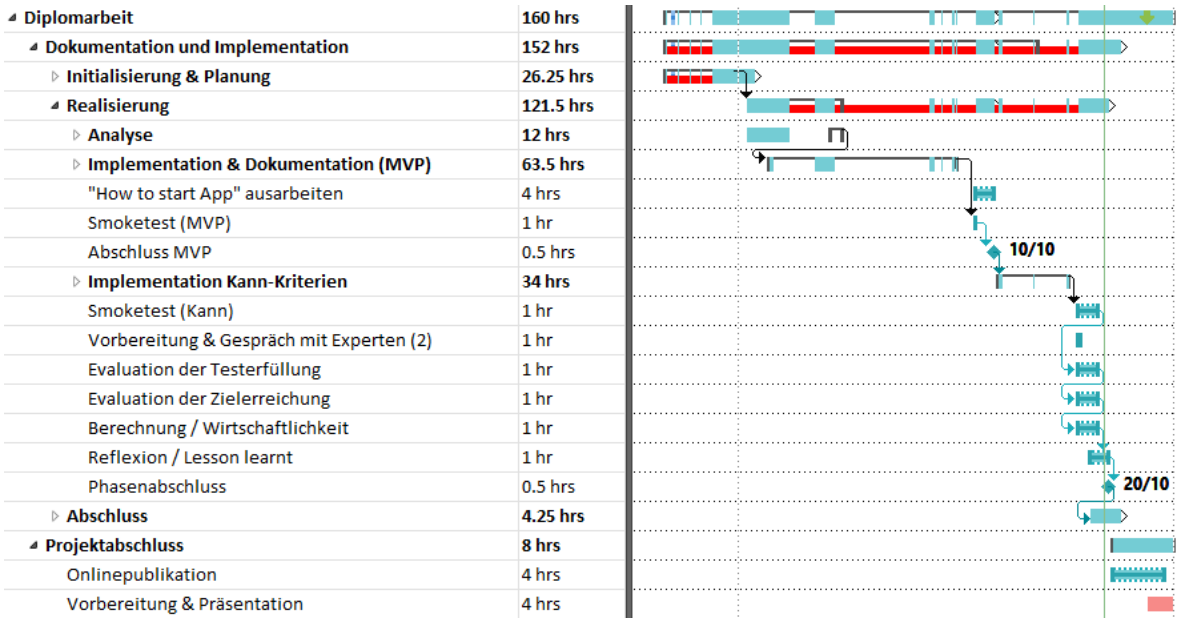


Abbildung 4: PAP gekürzt

2.4 Kommunikationsplanung

Die Kommunikationsplanung beinhaltet die Planung der Kommunikation zwischen den verschiedenen Stakeholdern. Da nicht jeder Stakeholder die gleichen Informationen braucht, hilft es oft diese darzustellen.

2.4.1 Stakeholderanalyse / Ansprechsgruppen

Unter einer Stakeholderanalyse versteht man die systematische Ermittlung von Interesseträgern, sowie deren Einfluss auf eine bestimmte Entscheidung. Unter dem Begriff «Stakeholder» werden normalerweise alle Individuen, Gruppen oder Institutionen zusammengefasst, die von einer bestimmten Massnahme direkt oder indirekt, positiv oder negativ betroffen sind.

Mit der Stakeholderanalyse sollen relevante Bezugsgruppen ermittelt werden, um so die Interessen der einzelnen Stakeholder besser zu verstehen. Im Zusammenhang mit der Diplomarbeit gestaltet sie sich simpel.

2.4.1.1 Stakeholder

Auftraggeber → Projektleiter → durchgehende Kommunikation bei der technischen Umsetzung
Diplomlehrer → Projektleiter → Wochenbericht + zwei individuelle Termine

2.4.2 Fieldforceanalyse / Kraftfeldanalyse

Unter Fieldforceanalyse versteht man eine einfache Methode zur Analyse der treibenden und rückhaltenden Faktoren in einer Situation. Die Kraftfeldanalyse stellt einen Vorstellungsrahmen für eine Situation dar. Sie betrachtet Kräfte, die entweder auf ein Ziel hintreibend (helfende Kräfte) oder blockierend wirken (hindernde Kräfte) und so die Situation als den Gleichgewichtszustand erzeugen.

Im Falle der Diplomarbeit gestaltet sie sich simpel und hat keine hindernden Kräfte:

2.4.2.1 Kraftfeld

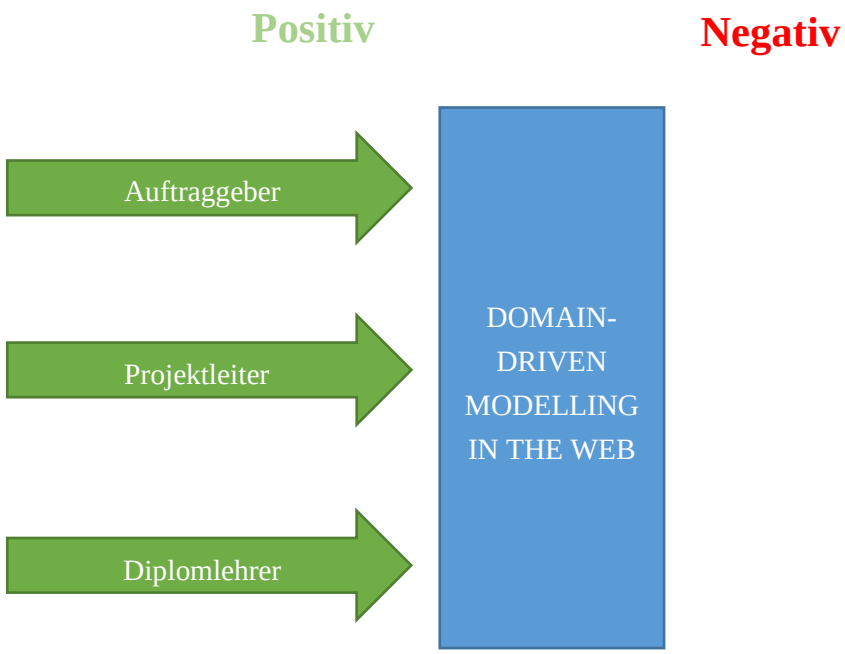


Abbildung 5: Kraftfeldanalyse

2.4.2.2 Analyse

Jeder Stakeholder wünscht sich eine Erfolgreiche Durchführung des Projektes, weshalb nichts im Wege steht.

2.5 Allgemeine Risikoanalyse

Da neben projektspezifischen Risiken auch noch Risiken existieren, welche vom Menschen teilweise nicht kontrolliert werden können, aber trotzdem grosse Auswirkungen auf das Projekt haben können, müssen diese auch analysiert werden.

Diese Risiken beinhalten:

- Krankheit und Unfälle
- Verpassen des Abgabetermins
- Datenverlust bei der Dokumentation / Code
- Probleme bei der Entwicklung
- Hardwareausfall
- Fehler in der Dokumentation

Die genaue Risikotabelle ist im Anhang, da sie für die Dokumentation selbst zu gross ist.

2.5.1 Risikomatrix

Die Visualisierung der Risiken erfolgt nach der Analyse mit der Darstellung der Risiken in einer 16er Matrix. Die Tabelle in der X-Richtung beinhaltet das Schadensausmass und in der Y-Richtung die «Eintrittswahrscheinlichkeit». Entsprechend ihrer Bewertung in der Risikotabelle werden die Risiken in die Matrix übertragen.

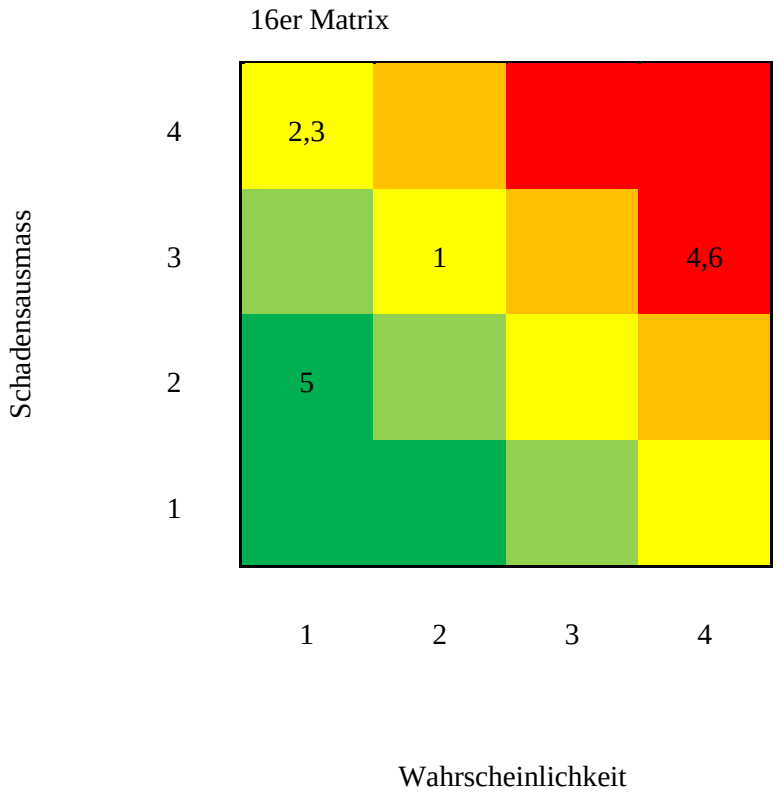
Die Kategorien:

Grüner Bereich:
Kein Handlungsbedarf erforderlich, da das Projekt nur sehr gering beeinflusst wird.

Gelber Bereich:
Die Risiken erfordern keinen unmittelbaren Handlungsbedarf, müssen aber beobachtet werden, um allfällige Tendenzen erkennen zu können. Massnahmen können vorgesehen werden.

Roter Bereich
Diese Risikokategorie beeinflussen das Projekt massiv und gefährden es. Vorbeugende Massnahmen müssen definiert werden. Für die Realisierung des Projektes muss die Massnahme umgesetzt werden.

Vor der Präventionsstrategie



Nach der Präventionsstrategie

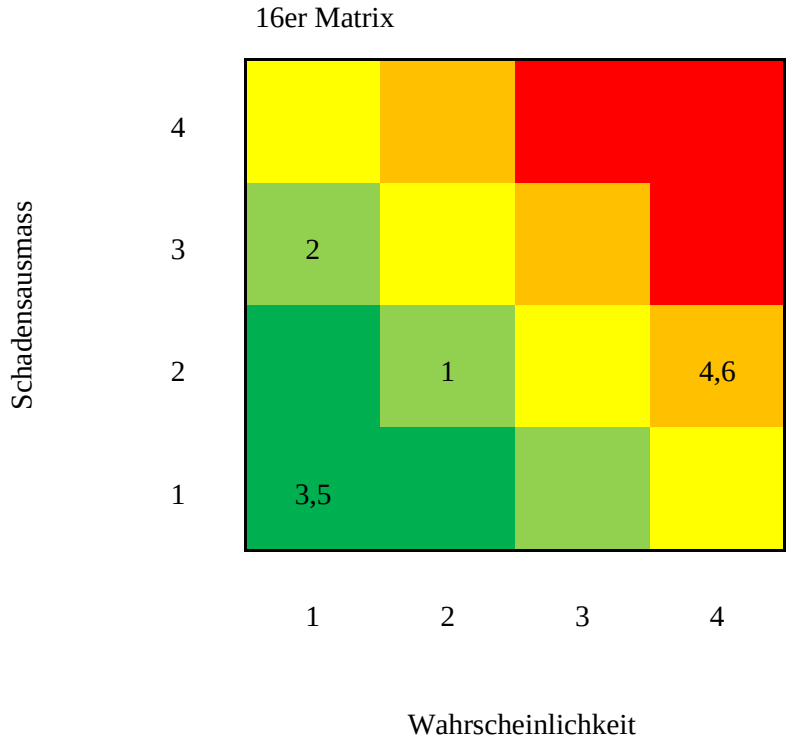


Tabelle 2: Allgemeine Risikomatrix

2.5.1.1 Fazit

Die allgemeinen Risiken umfassen zum Grossteil Risiken, welche jederzeit im Alltag eines Softwareentwicklers vorkommen können. Viele werden bereits durch Systeme wie z.B. «Git» für die Datenspeicherung ausgehebelt.

Im Allgemeinen sind alle der Risiken gut verkraftbar und kein Problem für die Arbeit insofern alle beteiligten Personen informiert werden und alles Nötige dokumentiert wurde.

Risiken wie Krankheiten kann man jedoch nicht gross beeinflussen, wenn zum Zeitpunkt der Arbeit noch eine Pandemie und viele Endemien wüten. Es gilt die gleichen Massnahmen zu befolgen, welche auch im Alltag befolgt werden.

Fehler in der Dokumentation können durch mehrfaches Lesen vermieden werden, allerdings gilt auch hier, dass Fehler machen, menschlich ist und ein perfektes Produkt nicht möglich ist. Es gilt sein Bestes zu geben, um das Beste aus dem Endresultat zu machen.

2.6 Testkonzept

Das Testkonzept beschreibt die Testziele, Testobjekte, Testarten, Testinfrastruktur sowie die Testorganisation. Es umfasst ebenfalls die Testplanung und die Testfallbeschreibungen. Für jeden Testfall wird eine detaillierte Testfallbeschreibung erstellt. Diese stellt die Spezifikation des Tests dar. Die Testplanung legt den logischen und zeitlichen Ablauf der Tests fest. Das Testkonzept bildet die Grundlage, auf der die Testorganisation und die Testinfrastruktur bereitgestellt und die Tests durchgeführt werden. Es wird bei neuen Erkenntnissen nachgeführt.[1]

2.6.1 Testziele

Die Ziele von Tests sind:

- Qualität des Produkts sicherstellen
- Schwerwiegende Fehler im Produkt frühzeitig erkennen
- Eine korrekte Implementation sicherstellen

Alle Tests werden während der Realisierungsphase durchgeführt. Auftretende Fehler werden dokumentiert und falls möglich anschliessend gelöst.

2.6.2 Testarten

In diesem Kapitel werden die verschiedenen Testarten erklärt. Dabei geht es um Testarten, die den Source Code direkt testen und manuelle Tests, welche vom Benutzer selbst durchgeführt werden.

2.6.2.1 JUnit

Um die Qualität des implementierten Codes im Backend sicherzustellen, werden JUnit Testfälle erstellt. JUnit Testfälle testen, wie der Name bereits sagt, eine «Unit» je Test. Es handelt sich hierbei um isolierte Testfälle, bei denen Abhängigkeiten normalerweise «gemockt» werden. Bei Mocks handelt es sich um Platzhalter, welche vom System als richtige Objekte behandelt werden. Falls die Implementation es ohne grossen Leistungsverlust zulässt, können auch richtige Objekte verwenden werden.

Es werden alle neu erstellten Klassen und Methoden durch JUnit abgedeckt.

2.6.2.2 Integrationstest

Um die Qualität des Gesamtsystems sicher zu stellen, werden Integrationstests angewendet. Diese bestehen aus Reihen von abgestimmten Einzeltests, die dazu dienen voneinander abhängige Komponenten im Zusammenspiel miteinander zu testen.

[1] Direkt übernommen: (Kanton Zürich, 2022)

2.6.2.3 Smoketest / manuelle Tests

Um Fehler bei der Ausführung von Funktionen zu vermeiden, wir das System von Hand oder u.a. manuell getestet. Diese können unter anderem mithilfe von Frameworks wie «Swagger» gemacht werden oder so, wie der Endbenutzer die Software anwenden wird.

Es geht bei diesen Tests unter anderem auch um das «out-of-the-box» denken, um Fälle zu testen die ein automatischer Ablauf nicht einfach testen kann.

2.6.2.4 E2E-Tests im Frontend

Ähnlich wie bei Integrationstests im Backend kann man auch das Frontend testen. Dafür gibt es heute Tools, welche vollständige Abläufe emulieren können. Eine dieser Frameworks ist zum Beispiel «Cypress» welches zum Testen der Applikation benutzt werden kann. Dabei wird der volle Ablauf, von der Anfrage bis zur Antwort getestet.

2.6.3 Testfälle

In diesem Kapitel werden die Testfälle aufgezeigt. Diese werden Schritt für Schritt aufgezeigt.

2.6.3.1 Muss-Kriterien

In diesem Kapitel sind die Testfälle für die Anforderungen beschrieben, welche zwingend umgesetzt werden müssen.

Korrektes Laden der Inhalte

ID	TF-M1
Testbeschreibung	Die Webapplikation ist über localhost:[PORT] aufrufbar und lädt eine Weboberfläche bei Aufruf der URL, welche ein Editorfeld mit einem Beispiel Code Segment beinhaltet und dem entsprechend generierten Diagramm in einem anderen Feld.
Vorbedingung	-
Testschritte	1. Die Webseite unter der URL «localhost:4200» aufrufen 2. Oberfläche überprüfen
Erwartetes Resultat	- Die Weboberfläche wird bei Aufruf korrekt geladen. - Es wird ein Beispieldiagramm durch Beispielcode im Editor-text generiert

Tabelle 3: TF-M1 - Korrektes Laden der Inhalte

Textfeld lädt und ist bearbeitbar

ID	TF-M2
Testbeschreibung	Bei Aufrufen der URL lädt die Oberfläche mit einem Beispiel Code Segment im Editor Feld, welches wie ein Textfeld bearbeitbar ist.
Vorbedingung	-
Testschritte	1. Die Webseite unter der URL «localhost:4200» aufrufen 2. Editorfeld anklicken 3. «Test» auf der Tastatur drücken
Erwartetes Resultat	- Die Weboberfläche lädt mit einem Beispiel .cml Code - Das Editorfeld spiegelt die Eingaben des Benutzers wieder («Test»)

Tabelle 4: TF-M2 - Textfeld lädt und ist bearbeitbar

Generieren eines Diagramms

ID	TF-M3
Testbeschreibung	Wenn der «Generate» Button auf der Oberfläche gedrückt wird, und das Code Segment im Editor valide ist, wird ein aktualisiertes Diagramm auf der rechten Seite ausgegeben.
Vorbedingung	-
Testschritte	1. Die Webseite unter der URL «localhost:4200» aufrufen 2. «Generate»-Button anklicken 3. Diagramm überprüfen
Erwartetes Resultat	- Das Diagramm wird dem Code entsprechend generiert

Tabelle 5: TF-M3 - Generieren eines Diagramms

Exportieren eines Diagramms

ID	TF-M4
Testbeschreibung	Wenn ein vom Generator unterstütztes Exportformat ausgewählt und der «Export» Button gedrückt wird, wird die Datei im jeweiligen Format runtergeladen
Vorbedingung	-
Testschritte	1. Die Webseite unter der URL «localhost:4200» aufrufen 2. «Export»-Button anklicken 3. Diagramm lokal aufrufen
Erwartetes Resultat	- Das Diagramm ist nach herunterladen korrekt aufrufbar

Tabelle 6: TF-M4 - Exportieren eines Diagramms

2.6.3.2 Kann-Kriterien

In diesem Kapitel sind die Testfälle für die Anforderungen beschrieben, welche nach Beendigung des MVPs umgesetzt werden können.

Generieren eines Alternativ-Diagramms

ID	TF-K1
Testbeschreibung	Auf der Oberfläche kann via Auswahlfeld «Context Map» oder «PlantUML» ausgewählt werden und via «Generate» Button wird das jeweilige Format als Diagramm generiert.
Vorbedingung	-
Testschritte	1. Die Webseite unter der URL «localhost:4200» aufrufen 2. Auf Dropdown oberhalb vom Editor klicken 3. «PlantUML» auswählen 4. «Generate» klicken 5. Überprüfen, ob das Diagramm «PlantUML» Format entspricht
Erwartetes Resultat	- Es wird ein Diagramm im «PlantUML» Format generiert und dargestellt

Tabelle 7: TF-K1 - Generieren eines Alternativ-Diagramms

IntelliSense

ID	TF-K2
Testbeschreibung	Beim Schreiben von Code im Editor werden Vorschläge für Autovervollständigung gegeben und Fehler werden hervorgehoben (beispielsweise rot unterstrichen).
Vorbedingung	-
Testschritte	1. Die Webseite unter der URL «localhost:4200» aufrufen 2. Editorfeld anklicken 3. «Bou» eingeben 4. Enter drücken 5. «abc» eingeben
Erwartetes Resultat	- Nach der Eingabe von «Bou» erscheint der Vorschlag «BoundedContext» - Bei Drücken von Enter wird «Bou» durch «ndedContext» erweitert - «abc» wird rot unterstrichen

Tabelle 8: TF-K1 - IntelliSense

Realtime Generierung

ID	TF-K3
Testbeschreibung	Beim Bearbeiten vom Code im Editorfeld wird bei validem Code eine Anfrage an das Backend geschickt, welches ein neues Diagramm zurückschickt und auf dem Frontend darstellt.
Vorbedingung	-
Testschritte	1. Die Webseite unter der URL «localhost:4200» aufrufen 2. Editor Feld anklicken 3. «`VoyagePlanningContext <-> LocationContext`» in der geschweiften Klammer einfügen
Erwartetes Resultat	- Nach dem Einfügen wird automatisch ein neues Diagramm generiert

Tabelle 9: TF-K3 - Realtime Generierung

3 Projektrealisierung

Die beiden abgeschlossenen Phasen Projektinitialisierung sowie Projektplanung bilden die Basis für die Phase Projektrealisierung.

In diesem Kapitel werden die Endergebnisse auf dem Weg zum Ziel erarbeitet.

Da der Inhalt der Arbeit bereits feststeht, wird auf eine Methode zur Ideenfindung verzichtet. Kleinere Analysen sind in den jeweiligen Kapiteln vorzufinden. Die Funktionsweise der «Context Mapper» Bibliothek ist durch Beispiele[2] in Github und der Dokumentation auf der Hauptseite[3] abgedeckt.

3.1 Variantenentscheid Editor

Ein grosser Teil dieser Arbeit ist ein Editor, welcher den geschriebenen Code entgegennimmt und richtig behandelt. Im «minimum viable product» (MVP) reicht ein einfaches Textfeld für die Eingabe, allerdings wäre die Verwendung von einem Xtext-Kompatiblen Editor ein Wunsch. Dieser ist direkt mit den Kann-Kriterien verbunden, da er «Syntax Highlighting», sowie «Autocompletion» für die DSL vom «Context Mapper» bereitstellen kann.

«Eclipse», der Erschaffer von «Xtext», bietet grundsätzlich Unterstützung für drei JavaScript Editoren an.

Diese sind als Tabellenform nach ihren Vor- und Nachteilen aufgereiht.

	Orion	Ace	CodeMirror
Content assist	✓	✓	✓
Validation	✓	✓	✓
Syntax highlighting	✓	✓	✓
Semantic highlighting	✓		✓
Mark occurrences	✓	✓	✓
Hover information	✓		
Formatting	✓	✓	✓
Code generator	✓	✓	✓
Persistence	✓	✓	✓

Abbildung 6: Features per Editor[4]

[2] https://github.com/ContextMapper/context-mapper-standalone-example
[3] https://contextmapper.org/docs/home/
[4] (Eclipse, 2022)

3.1.1 Kurzbeschreibung der Variante Orion

Der Orion Webeditor ist ein von Eclipse erstellter und verwalteter Webeditor. Er wurde erstellt, um eine Browser basierende offene Werkzeug Integrationsplattform zu erschaffen, welche das Entwickeln von Webprojekten im Web ermöglicht.

Er hat die höchste Kompatibilität mit Xtext Sprachen, da beide vom Haus Eclipse stammen.

3.1.2 Kurzbeschreibung der Variante Ace

Der Ace (Ajax.org Cloud9 Editor) Webeditor ist genau wie Eclipse Orion ein Webeditor, der entwickelt wurde um Webprojekte im Web zu entwickeln. Er wurde so gebaut, dass er mit der Performanz von nativen Editoren wie zum Beispiel «Vim» oder «Sublime» mithalten kann. Er wird als Primäreditor für die Cloud9 IDE verwendet, welche eine Entwicklungsumgebung auf der Cloud ist und das Kollaborieren zwischen mehreren Personen im Team ermöglicht.

Er unterstützt kein «semantic highlighting» und kann keine Informationen beim «hovern» anzeigen, ist allerdings native als «npm» Paket erhältlich.

3.1.3 Kurzbeschreibung der Variante CodeMirror

CodeMirror ist ein Open Source Editor, welcher wie die anderen beiden das Entwickeln von Webprojekten im Web ermöglicht. Er kommt bereits mit Support für diverse Sprachen und eignet sich daher für den Schnelleinstieg ins webbasierte Entwickeln.

Er unterstürzt alles ausser Informationsanzeige beim «hovern». Er ist der beliebteste Editor von den Drei.

3.2 Evaluation der geeignetsten Variante

Die drei genannten Varianten eignen sich alle für die Umsetzung mit der Xtext erstellten Sprache. Allerdings soll nun herausgefunden werden, welche Variante sich am besten für das Projekt eignet. Dafür werden verschiedene Kriterien definiert, bei welchen alle drei Varianten bewertet werden.

3.2.1 Kriterien

In diesem Kapitel werden die Kriterien aufgezählt, nach welchen die Editoren bewertet werden sollen. Kriterien wie Kosten oder Sicherheit werden nicht aufgezählt, da sie im Zusammenhang mit diesem Projekt irrelevant sind.

3.2.1.1 Features

Anzahl an Features wie zum Beispiel «Syntax Highlighting» oder «Formatting». Die Features können dem Abbildung 6: Features per Editor im Kapitel Variantenentscheid Editor entnommen werden.

3.2.1.2 Komplexität

Komplexität der Implementation des jeweiligen Editors. Darunter sind unter anderem Unterkriterien, wie die «out-of-the-box» Unterstützung durch «Eclipse Xtext» oder der Aufwand, um die volle Unterstützung von Features erreichbar zu machen.

3.2.1.3 Angular Unterstützung

Unterstützung in Angular «out-of-the-box». Angular ist seit der zweiten Version TypeScript basiert und hat daher ganz eigene Vorgehen bei der Verwaltung von Paketen. Beliebte Pakete wie «RequireJS» für Abhängigkeiten von Paketen werden daher nicht ohne Aufwand unterstützt.

3.2.1.4 Maintenance

Wartungs- und Aktualisierungsinterval der Editoren. Damit die Applikation auch in Zukunft weiterentwickelt werden kann, benötigt sie einen Editor, der auch noch in einigen Jahren brauchbar ist. Der Release-Zyklus von den jeweiligen Editoren ist daher für dieses Kriterium relevant.

3.2.2 Präferenzmatrix

Mit den Kriterien, die in Bezug auf alle Varianten angewendet werden müssen, soll eine Entscheidung herbeigeführt werden, die den Stellenwert die der einzelnen Kriterien innerhalb des Projektes definiert. Sie kann mit einer Präferenzmatrix dargestellt werden. Diese Entscheidungsfindung ist möglichst mit dem gesamten Team durchzuführen. Unstimmigkeiten sollten ausdiskutiert werden.

Die Präferenzmatrix ist ein einfaches, aber effektives Hilfsmittel, um den Kriterien das richtige Gewicht zu geben. Im Zentrum steht jeweils nur eine Frage: »was ist wichtiger: Kriterium A oder B?» diese Methode führt zu einer klaren Rangfolge der Kriterien. Dadurch wird die Bewertung der Kriterien wesentlich objektiver als durch irgendwelche Entscheidungen die schnell gefällt worden sind.

Die Tabelle der Nutzwertanalyse wir von links nach rechts gelesen und zeigt die jeweilige Priorisierung. Die Kriterien werden links und oben in der gleichen Reihenfolge aufgeschrieben, aufgezählt und miteinander verglichen.

DOMAIN-DRIVEN MODELLING IN THE WEB		**a**	**b**	**c**	**d**	**Anzahl**
		Features	Komplexität	NG-Unterstützung	Maintenance	
a	Features		b	c	d	3
b	Komplexität			b	b	2
c	NG-Unterstützung				d	1
d	Maintenance					0
Nennungen		0	3	1	2	6
Rang		4	1	3	2	10
Prozent		0%	50%	16.65%	33.35%	100%

Tabelle 10: Präferenzmatrix

Die Features sind insgesamt am irrelevantesten, da jeder Editor genug Features bietet, welche den Benutzer unterstützen. Da «Eclipse» und «Xtext» «Oldschool»-Applikationen sind, wird vor allem darauf geachtet, dass die Komplexität nicht zu hoch ist, da sonst die Implementation und das Warten der Applikation zu aufwendig sind.

3.2.3 Nutzwertanalyse

Gilt es Lösungsvarianten zu beurteilen, um zu einem Entscheid zu gelangen, welcher auf bekannten gewichteten Kriterien beruht, ist eine Nutzwertanalyse eine vielfach eingesetzte Methode.

In einer vorbereiteten Tabelle werden in der Spalte «Gewichtung» die errechneten Prozentzahlen der Präferenzmatrix eingesetzt. Dadurch wird ermöglicht, dass bei der Bewertung der Lösungsvarianten die Kriterien den entsprechenden Stellenwert erhalten.

3.2.3.1 Teilnutzen (TN)

Der Teilnutzen wird benötigt, um die Varianten untereinander zu bewerten. Zur Bestimmung des Teilnutzens in wenig komplexen Projekten ist der maximale Teilnutzen durch die Anzahl Varianten gegeben, das heisst in diesem Fall eins bis drei.

Die höchste Zahl erhält die Variante, für die ein bestimmtes Kriterium den höchsten Teilnutzen hat.

3.2.3.2 Gewichteter Teilnutzen (GTN)

Mit dem gewichteten Teilnutzen wird der Teilnutzen der einzelnen Varianten mit der Gewichtung und dem Ergebnis aus der Präferenzmatrix vernetzt

d.h. die Wertung im Teilnutzen (eins bis drei) multipliziert, man mit dem Prozentwert der vorangegangenen Präferenzmatrix.

3.2.3.3 Tabelle

DOMAIN-DRIVEN MODELLING IN THE WEB		Variante 1		Variante 2		Variante 3	
		Orion		Ace		CodeMirror	
Kriterien	**Gewichtung**	**TN**	**GTN**	**TN**	**GTN**	**TN**	**GTN**
Features	0	3	0	1	0	2	0
Komplexität	50	1	50	3	150	2	100
NG-Unterstützung	16.65	1	16.65	3	49.95	2	49.95
Maintenance	33.35	1	33.35	3	100.05	2	66.7
Total			**100**		**300**		**200**

Tabelle 11: Nutzwertanalyse

3.2.3.4 Fazit

Anhand unserer Nutzwertanalyse haben wir unsere drei Varianten bewertet. Dabei hat sich die Variante «Ace» als vielversprechend herausgestellt, da «Xtext» standardmässig mit diesem Editor arbeitet und genügend Funktionen bietet. «Ace» bietet eine TypeScript Implementation, weshalb damit auch die Angular Unterstützung gegeben ist.

«Ace» ist zudem am besten gewartet, da während der Arbeit alleine drei Versionen rauskamen.

Mehr Infos dazu hier: https://www.npmjs.com/package/ace-builds

«CodeMirror» bietet neben «Ace» einige Angular Wrapper an und ist im Allgemeinen sehr beliebt. «Orion» benötigt einige Kniffe, damit man diesen in eine Webapplikation einbauen kann und hat am wenigsten Unterstützung in Angular. Das «npm»-Package ist ausserdem schlecht gewartet.

3.2.4 Sensitivitätsanalyse

Um den Favoriten aus der Nutzwertanalyse zu verifizieren, kann man das Resultat via einer Sensitivitätsanalyse auf seine Aussagekraft und Stabilität überprüfen. Mittels Veränderung der Gewichtung der relevantesten Kriterien kann die Stabilität geprüft werden.

DOMAIN-DRIVEN MODELLING IN THE WEB		**Variante 1**		**Variante 2**		**Variante 3**	
		Orion		Ace		CodeMirror	
Kriterien	**Gewichtung**	**TN**	**GTN**	**TN**	**GTN**	**TN**	**GTN**
Features	33.35	3	100.05	1	33.35	2	66.7
Komplexität	0	1	0	3	0	2	0
NG-Unterstützung	50	1	50	3	150	2	100
Maintenance	16.65	1	16.65	3	49.95	2	33.3
Total			**166.7**		**233.3**		**200**

Tabelle 12: Sensitivitätsanalyse 1

Wenn die Gewichtung mehr auf die Angular Unterstützung und Features und gar nicht auf die Komplexität legt, dann kommen sich die drei Editoren alle einen Schritt näher. «Orion» holt stark auf, da er mehr Features hat. «CodeMirror» ist gleichbleiben, da er ein durchaus ausgewogener Editor ist.

Die bevorzugte Variante bleibt allerdings weiterhin «Ace».

DOMAIN-DRIVEN MODELLING IN THE WEB		Variante 1		Variante 2		Variante 3	
		Orion		Ace		CodeMirror	
Kriterien	Gewichtung	TN	GTN	TN	GTN	TN	GTN
Features	50	3	150	1	50	2	100
Komplexität	16.65	1	16.65	3	49.95	2	33.3
NG-Unterstützung	33.35	1	33.35	3	100.05	2	66.7
Maintenance	0	1	0	3	0	2	0
Total			**200**		**200**		**200**

Tabelle 13: Sensitivitätsanalyse 2

Das Verlagern der Gewichtung auf die Features ist der einzige Sonderfall, in dem alle drei Varianten Parität erreichen. In allen anderen Fällen, in denen das Kriterium «Features» nicht die stärkste Gewichtung hat, kommt «Ace» als bevorzugte Variante hervor.

3.2.5 Resultat der Variantenevaluation

Wenn man die Gewichtungen der Kriterien ändert, merkt man schnell, dass «Orion» vor allem bei den Features bevorzugt wird, während «CodeMirror» ein durchgängig ausgewogener Editor für diese Implementation mit «Xtext» ist.

Allerdings gewinnt «Ace» insgesamt bei der niedrigen Komplexität der Umsetzung und Angular Unterstützung, da er der Standardeditor von «Xtext» ist und er mit native TypeScript Unterstützung kommt. Zudem ist er ein sehr gut gewarteter Editor, was ihn zu einem Kandidaten für die Zukunft macht. Im Rahmen der Arbeit und allgemein macht daher «Ace» am meisten Sinn, mit «CodeMirror» als gute Zweitwahl. «Orion» ist im Gegensatz zu den anderen zwei zu komplex zum Implementieren und sollte man nur wählen, wenn man unbedingt alle Funktionen braucht.

Gewinner: «Ace»

3.3 Konzeptionelle Ausarbeitung der Variante «Ace»

Da es sich bei dem Variantenentscheid dieser Arbeit lediglich um einen Teilbereich der Applikation handelt, der im MVP nicht relevant ist, werden die Analysen kleiner gehalten und auf Sicht der Entwicklungen nach der Abgabe des MVPs.

Sonstige Variablen waren bereits vordefiniert und keine Auswahl für einen Variantenentscheid.

3.3.1 SWOT – Analyse

Jedes Projekt und jede Handlung bergen nebst Chancen auch Risiken. Um die möglichen Folgen für die Gruppe abschätzen zu können, müssen die Chancen, Risiken sowie die Stärken und Schwächen des Projektes aufgestellt werden.

Das Instrument der SWOT-Analyse mit den vier Aspekten Stärken (**S**trengths), Schwächen (**W**eaknesses), Chancen (**O**pportunities) sowie Risiken (**T**hreats) bilden den Bestandteil der Analyse.

Stärken (Strengths) • Gut gewarteter Editor • Stabiler Editor, welcher zu der Zeit gebaut wurde, als Editoren noch stabil sein mussten • Hohe Performanz • Primäreditor von Xtext	**Schwächen** (Weaknesses) • Hat weniger Features als andere Editoren • Nicht responsive wie andere Editoren und daher ungeeignet für mobile Plattformen
Chancen (Opportunities) • Schnelle Umsetzung • Kann weit in die Zukunft unterstützt werden • Open-Source für die selbständige Weiterentwicklung	**Risiken** (Threats) • Herausforderung bei Kompatibilität mit Handheld Geräten • Keine Unterstützung von sehr hilfreichen Features

Tabelle 14: SWOT "Ace"

3.3.2 Risikoanalyse

Neben der allgemeinen Risikoanalyse gibt es einige Risiken bei der Benutzung der bevorzugten Varianten. Diese sind, da sie kein Teil des MVPs sind nicht so schlimm wie die allgemeinen Risiken, bergen allerdings Auswirkungen wie Zeitverlust bei der Weiterentwicklung.

Diese Risiken beinhalten:

- Unterstützung durch Framework nicht gegeben
- Unterstützung durch «Xtext» nicht gegeben
- Der Editor wird nicht gewartet
- Der Editor ist nicht kompatibel mit dem Rest der Applikation
- Dokumentationen zur Umsetzung sind nicht vorhanden

Die genaue Risikotabelle ist im Anhang, da sie für die Dokumentation selbst zu gross ist.

3.3.2.1 Risikomatrix

Die Visualisierung der Risiken erfolgt nach der Analyse mit der Darstellung der Risiken in einer 16er Matrix. Die Tabelle in der X-Richtung beinhaltet das Schadensausmass und in der Y-Richtung die «Eintrittswahrscheinlichkeit». Entsprechend ihrer Bewertung in der Risikotabelle werden die Risiken in die Matrix übertragen.

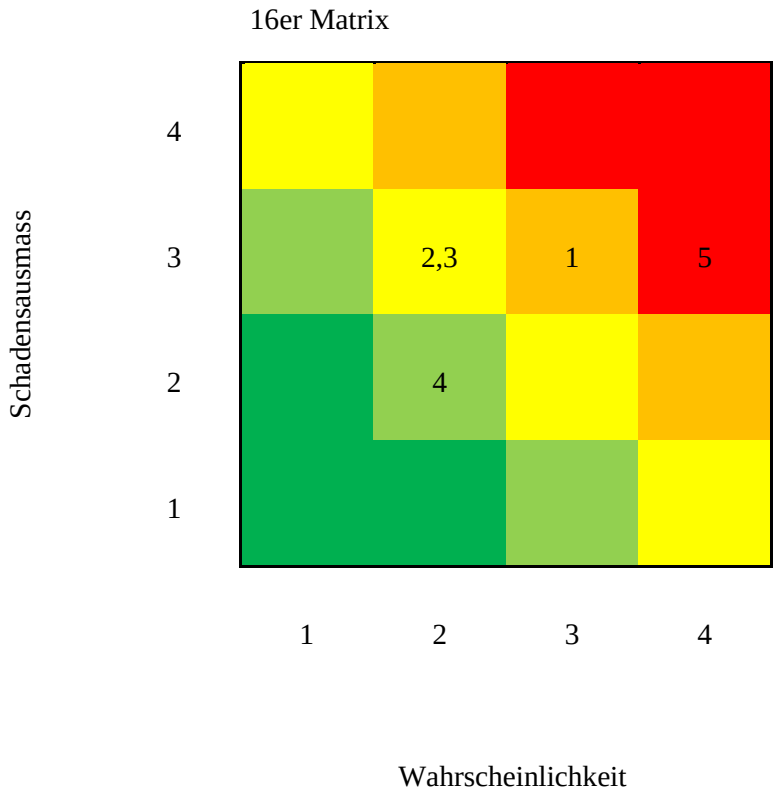
Die Kategorien:

Grüner Bereich:
Kein Handlungsbedarf erforderlich, da das Projekt nur sehr gering beeinflusst wird.

Gelber Bereich:
Die Risiken erfordern keinen unmittelbaren Handlungsbedarf, müssen aber beobachtet werden, um allfällige Tendenzen erkennen zu können. Massnahmen können vorgesehen werden.

Roter Bereich
Diese Risikokategorie beeinflussen das Projekt massiv und gefährden es. Vorbeugende Massnahmen müssen definiert werden. Für die Realisierung des Projektes muss die Massnahme umgesetzt werden.

Vor der Präventionsstrategie



Nach der Präventionsstrategie

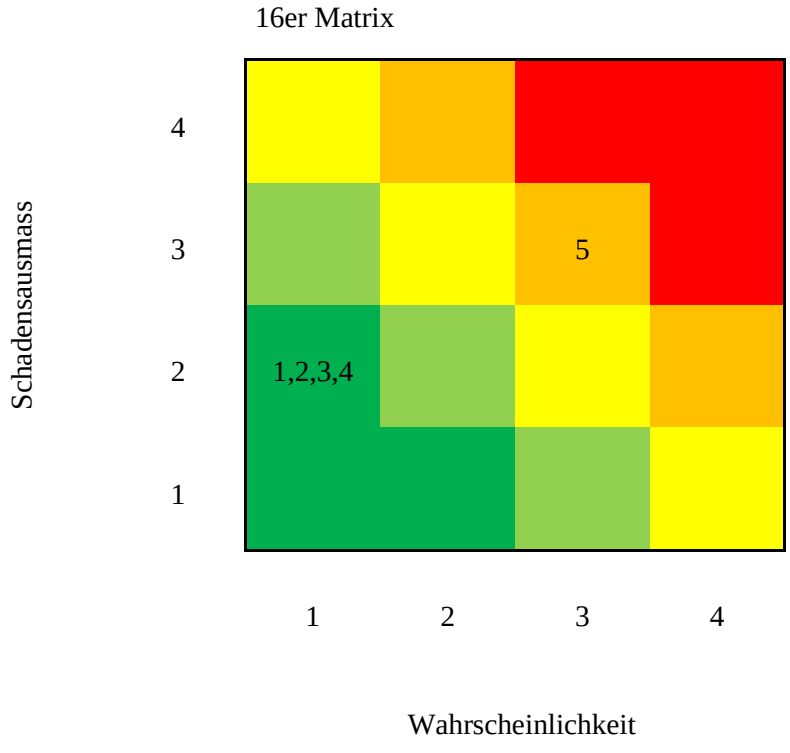


Tabelle 15: Risikomatrix "Ace"

3.3.2.2 Fazit

Die Risiken für die Umsetzung des Editors sind alle ähnlich, da es jederzeit zu einer Inkompatibilität mit einem Teil der Applikation kommen kann.

Zudem machen fehlende Dokumentationen die Umsetzung des «Xtext» Editors um einiges schwieriger.

Leider ist «Eclipse» dafür bekannt, dass die Kompatibilität zu neuartigen Technologien nicht sehr hoch ist und es daher auch wenige Dokumentationen gibt. Eine komplette Umsetzung von einem «Xtext» Editor mit «TypeScript» oder «Angular» konnte beispielsweise nicht gefunden werden.

3.4 Umsetzung Webapplikation

Die Webapplikation ist als 2-Layer Architektur aufgebaut, welche aus einem Angular Frontend, welches verantwortlich ist für die Darstellung, sowie einem Spring Backend, welches verantwortlich für die Verarbeitung aller Daten ist.

Das Frontend soll auf einen «XText» kompatiblen Editor setzen, welcher in einem Variantenentscheid evaluiert wurde.

Das Backend ist an den bereits vorhanden «Context Mapper» Generator, sowie die DSL (CML) via Library angebunden und es wird kein neuer «XText» Language Server aufgesetzt. Die Library unterstützt die wichtigsten Operationen wie z.B. das Generieren der Diagramme mit dem jeweiligen Generator.

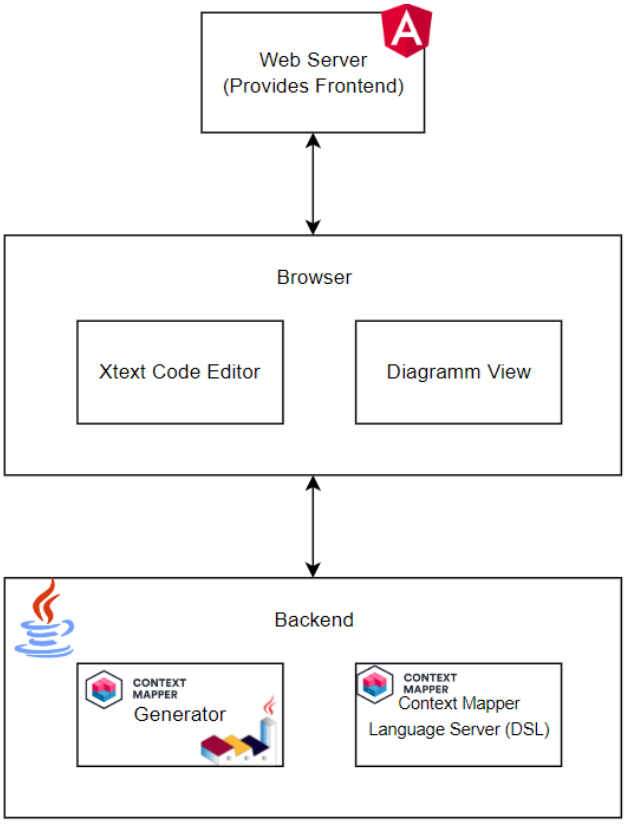


Abbildung 7: Architektur

Die Dokumentation teilt die zwei Layer auf und beschreibt diese in ihrem jeweiligen Kapitel. Übergreifende Herausforderungen werden dementsprechend beschrieben.

Das Frontend Kapitel enthält zwei nicht für den MVP relevante Kriterien, die auf Wunsch bereits umgesetzt wurden. Diese sind dementsprechend markiert. (Kann-Kriterien)

Das Backend Kapitel enthält auch ein nicht für den MVP relevantes Kapitel. Es wird auch dementsprechend erwähnt.

3.4.1 Backend

Das Backend stellt das Herzstück der Verarbeitung dar, welches die Eingabe nimmt und diese in das gewünschte Format mutiert sowie zurückgibt.

Im Backend wird im Rahmen dieses Projekt lediglich eine Implementation für die «Context Mapper» Bibliothek implementiert, wobei auf weiterführende Features im Rahmen der Arbeit je nach Zeitbedarf verzichtet wurde.

3.4.1.1 Die wichtigsten Fakten

Das Backend besteht zum Grossteil aus einer Implementation der «Context Mapper» Library, welche eine Abhängigkeit zu «Graphviz» hat.

3.4.1.1.1 «Context Mapper»

Die «Context Mapper» Library ist das Herzstück des ganzen Projektes. Sie stellt die Funktionen zur Verfügung, um Diagramme in der hauseigenen DSL zu verarbeiten und zu Diagrammen umzuwandeln.

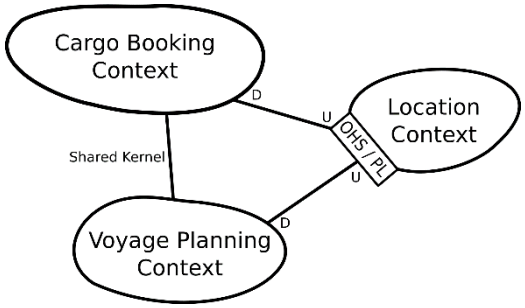


Abbildung 8: Context Map Beispiel

Die Library unterstützt «out-of-the-box» den hauseigenen «Context Mapper» Generator, sowie Generatoren für «PlantUML», «MDSL» und generische Anwendungsfälle. Es wird die neuste Version «6.6.1» verwendet.

Weitere Informationen unter: https://contextmapper.org/.

3.4.1.1.2 Graphviz

«Graphviz» ist eine Open Source Visualisierungs-Software, welche das Generieren von Diagrammen in verschiedenen Formaten bereitstellt. Die Software setzt auf die Sprache «DOT», welche eine Beschreibungssprache für das Darstellen von Graphen ist. Sie ermöglicht das textuelle Beschreiben eines Graphen, welcher wiederum durch «Graphviz» visualisiert wird.

Die Software muss für die Verwendung auf dem eigenen System installiert und als Umgebungsvariable konfiguriert sein.

Weiter Informationen unter: https://graphviz.org/.

3.4.1.1.3 Spring

Für die Umsetzung des Backends wurde sich vom Auftraggeber gewünscht, dass man dieses mit dem Spring-Framework macht, da dieses in grossen Teilen der Softwareindustrie gang und gäbe ist. Das Spring Framework unterstützt «out-of-the-box» alles, was man für die Webentwicklung braucht. Weiter Funktionen gibt es als «Starter» Abhängigkeiten zur Verfügung.

Für dieses Projekt wurde Spring Boot mit der neusten Version «2.7.4» verwendet.

Weitere Informationen unter: https://spring.io/.

3.4.1.1.4 Sonstige

Um maximale Unterstützung bei Entwicklungsumgebungen zu erzielen, ohne auf Funktionen zu verzichten, wurde die Java OpenJDK17 für die Entwicklung sowie für die Produktion genommen.

Für das Projektmanagement- sowie das Automatisierungstool wurde sich für «Gradle» entschieden, da dieses Tool bereits in Beispielen des «Context-Mappers» vorkam. Es bietet im Gegensatz zu Maven die fast gleichen Funktionen.

Weitere Informationen unter: https://gradle.org/.

3.4.1.2 Docker

Da das Projekt mit «Graphviz» bereits eine Abhängigkeit hat, welche nicht jeder Benutzer auf dem System hat, wurde entschieden, dass man die Applikation von Anfang an in einem Docker Container laufen lässt. Die Möglichkeit auf eine Docker-freie Applikation besteht allerdings weiterhin, auch wenn dafür jegliche Drittabhängigkeit erfüllt werden muss.

Damit Testfälle auch ohne lokale Abhängigkeiten ausgeführt werden können, habe ich mich für einen dualen Ansatz entschieden, bei dem es jeweils ein Image für einen Testdurchlauf gibt und eines für eine normale produktive Umgebung.

3.4.1.2.1 Produktives Image

Um das Produktiv-Image möglichst klein zu halten, wird es basierend auf «Alpine» Linux aufgebaut. Da «Graphviz» eine Abhängigkeit zu systeminstallieren Schriften hat, müssen diese manuell installiert werden, da «Alpine» standardmässig keine Schriften installiert hat. Die Abhängigkeiten, welche im Dockerfile vorkommen müssen, sind wie folgt:

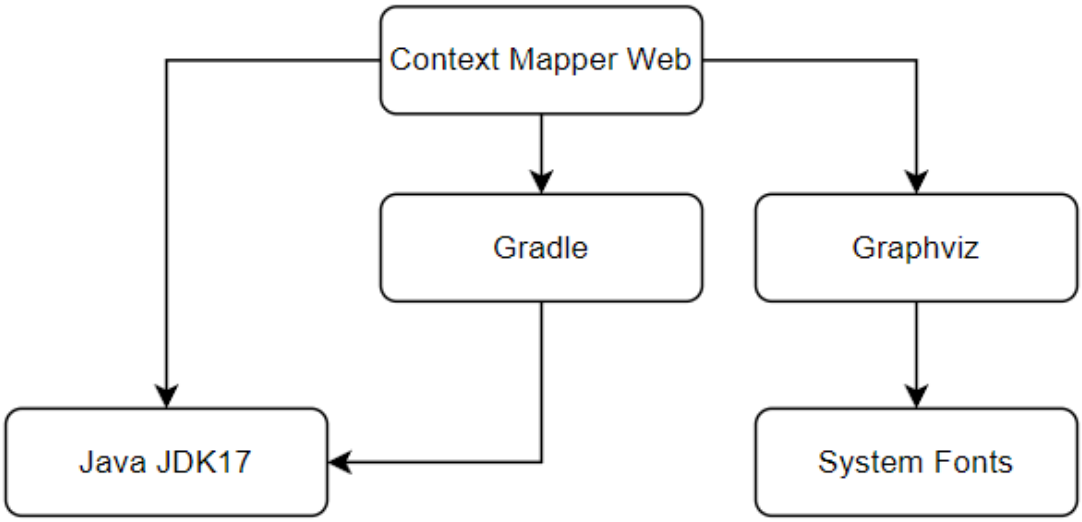


Abbildung 9: Dependency Graph Backend

Der Build vom JAR-File wird dabei in einem Vor-Schritt gemacht und dann von diesem in das Haupttimage kopiert.

```
FROM gradle:7.5.1-jdk17-alpine AS build
COPY --chown=gradle:gradle ../backend /home/gradle/src
WORKDIR /home/gradle/src
RUN gradle bootJar --no-daemon
```

Abbildung 10: Dockerfile Build

3.4.1.2.2 Test Image

Da das Test-Image voraussichtlich nur benutzt wird, um Funktionen sicherzustellen, wurden alle nötigen Funktionen berücksichtigt, ohne auf weitere einzugehen. Das Image basiert auf einem «Alpine» Linux, welches zusätzlich «Gradle» + «JDK17» im Bundle installiert haben. Wie das produktive Image werden «Graphviz» sowie Schriften installiert. Anschliessen wird der ganze Source Code auf das «Gradle»-Home kopiert, worauf schlussendlich ein «Gradle Build» gemacht wird, welcher eine Abdeckung der Testfälle beinhaltet.

```
FROM gradle:7.5.1-jdk17-alpine AS build

# Install graphviz + fonts for testing
RUN apk add graphviz
RUN apk add terminus-font ttf-inconsolata ttf-dejavu font-noto font-noto-cjk ttf-
font-awesome font-noto-extra

# Copy the whole source code into /home/gradle/src
COPY --chown=gradle:gradle . /home/gradle/src
WORKDIR /home/gradle/src
RUN mkdir /app

# Gradle Build + Tests with no-daemon
RUN gradle build --no-daemon
```

Abbildung 11: Dockerfile Test

3.4.1.3 Docker-Compose

Um die beiden Images individuell ansteuerbar zu machen, wird mit einer Docker-Compose Datei gearbeitet, welche zwei Services, einen für «web» (Produktives Images) und «test» (Testimage), bereitstellt.

Das Web-Image hat zusätzlich «Remote-Debugging» aktiviert, um Probleme live auf dem Container zu debuggen. Das Ganze wird über die Umgebungsvariable «JAVA_TOOL_OPTIONS» gesteuert und gibt in diesem Beispiel den Port «5005» als Debug-Port frei. Remote-Debugging kann jederzeit deaktiviert werden, indem die Environment Variable entfernt wird.

Der Standard-Port ist bei beiden Services «8080».

```
version: "3.9"
services:
  web:
    build:
      context: ../
      dockerfile: docker/Dockerfile-Spring
    ports:
      - "8080:8080"
      - "5005:5005"
    environment:
      - JAVA_TOOL_OPTIONS=-agentlib:jdwp=transport=dt_socket,server=y,sus-
pend=n,address=*:5005
  test:
    build:
      context: ../
      dockerfile: docker/Dockerfile-Gradle
    ports:
      - "8080:8080"
```

Abbildung 12: Docker-Compose web+test

3.4.1.4 Aufbau

Der Aufbau des Backends ist anders als bei einem konventionellen «Controller-Service-Repository» simpel aufgebaut, da neben den Schnittstellen (Controller) lediglich die jeweiligen Services für die Mutation der Daten existieren.

Um Daten in einer immer gleichen Form zu halten, werden diese, falls nötig oder schöner, mit jeweiligen Request/Response Modellen dargestellt. Diese sind auch, wie es für gewöhnlich wäre, nicht in «Data Access Object (DAO)» und «Data Transfer Objekt (DTO)» aufgeteilt, da die Daten sich durch die Linie nicht in der Form verändern.

Der Aufbau der Applikation in einem einfachen Klassendiagramm dargestellt würde folgendermassen aussehen:

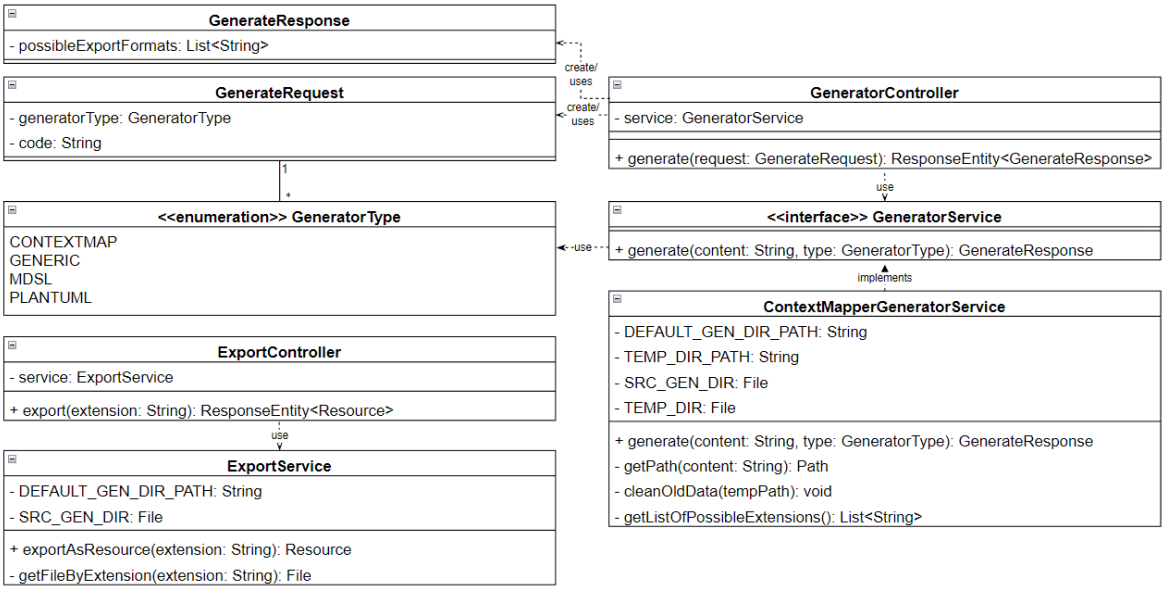


Abbildung 13: Klassendiagramm

Aus Platzgründen und um das Diagramm nicht zu komplex zu machen, wurden Spring-spezifische Klassen, wie Konfigurationsklassen oder Klassen mit Framework-spezifischen Annotationen ausgelassen.

3.4.1.4.1 Kann-Kriterium: Anbindung Generatoren abstrahieren

Vom Auftraggeber wurde sich gewünscht, dass man in Zukunft weitere Generatoren mit wenigem Aufwand in den Code einbinden kann. Das Ganze wurde vorerst so gelöst, dass allgemeine Funktionen in einem «GeneratorService»-Interface abstrahiert wurden. Dadurch können Implementationen individuell implementiert werden, während der Controller weiterhin auf die gleiche Funktion zugreift.

Das Ganze ist in Abbildung 13: Klassendiagramm ersichtlich.

3.4.1.4.2 API

Die Applikation besteht aus einer Schnittstelle mit zwei Operationen.

Die erste Operation ist die «generate» Funktion, welche ein Diagramm in einem gewünschten Format generieren kann. Die Operation benötigt den gewünschten Generator Typen, sowie dem Code, aus welchem das Diagramm generiert werden soll. Beides wird als JSON-Format «key-value pair» mitgegeben. Die Antwort auf einen Request besteht aus einer Liste von möglichen Exportformaten. Die generierte Datei wird in einem zweiten Schritt geholt.

Die zweite Operation ist die «export» Funktion, welche das Runterladen von anderen Formaten ermöglicht. Die Formate werden stets durch den Generator selbst entschieden, variieren also stark unter den Generatoren und werden dem Benutzer nach dem Generieren präsentiert. Die Operation verlangt lediglich das gewünschte Format als Pfadvariable (URL), da der Benutzer es von der Oberfläche auswählt, auf der das Diagramm bereits dargestellt ist. Die Antwort ist die jeweilige Datei als «Spring Resource».

Die URLs der Operationen korrespondieren jeweils mit den oben erwähnten Funktionsnamen.

```
@PostMapping(path = "/generate", produces = MediaType.APPLICATION_JSON_VALUE, con-
sumes = MediaType.APPLICATION_JSON_VALUE)
@Operation(summary = "Generates a diagram and return it",
        description = "Generates a diagram and return the file format corresponding
to the generator.")
public ResponseEntity<GenerateResponse> generate(@RequestBody @Valid GenerateRequ-
est request) {
    return ResponseEntity.ok(
            service.generate(request.getCode(), request.getGeneratorType()));
}

@PostMapping(path = "/export", consumes = MediaType.APPLICATION_JSON_VALUE)
@Operation(summary = "Exports a file-format",
        description = "Exports the selected file-format and returns it to make it
downloadable.")
public ResponseEntity<Resource> export(@RequestBody @Valid ExportRequest request)
throws IOException {
    return ResponseEntity.ok()
            .contentType(MediaType.APPLICATION_OCTET_STREAM)
            .body(service.exportAsResource(request.getExportFormat()));
}
```

Abbildung 14: API

3.4.1.4.3 Service

Das Mutieren der Daten passiert alles innerhalb eines Services, der Export wiederum in einem anderen Service. Hier wird die «ContextMapper»-API sowie die jeweiligen Generatoren aufgesetzt. Im MVP gibt es lediglich den «ContextMapGenerator».

Die «ContextMapper»-API hat eine Funktion «callGenerator(res, gen)» welche eine «CML-Resource» und einen Generator als Parameter annimmt. Die «CMLResource» kann instanziiert werden, indem eine Datei mit gültigem CML-Code geladen wird. Da der Code allerdings als String und nicht als Datei im Backend ankommt, wird mit temporären Dateien gearbeitet. Java stellt eine Klasse «Files» zur Verfügung, welches das Erstellen und Verwalten von temporär Dateien erlaubt. Dieses temporäre File zum Generieren wird danach wieder gelöscht. Die generierten Dateien werden zudem auch jedes Mal bei Neugenerieren gelöscht, sofern es mehr als zehn sind, damit das Dateisystem sauber bleibt. Dateien, welche beim Generieren entstehen werden als «DOT»-kompatible Dateien im Pfad «/tmp/graphviz» gespeichert und im «src-gen» werden Dateien abgelegt, welche die Generatoren erstellen.

In einem Diagramm abgebildet sieht der Ablauf folgendermassen aus:

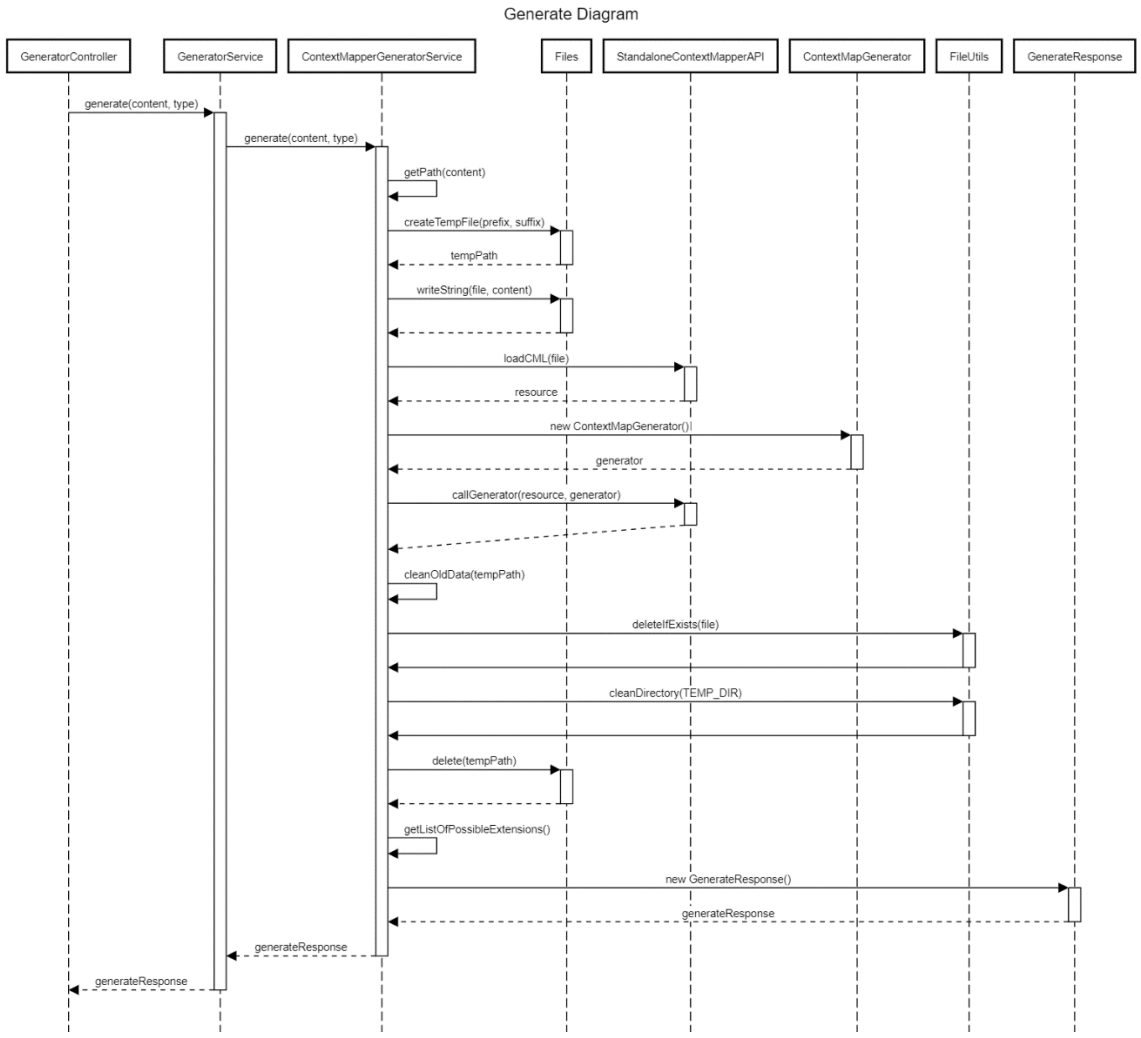


Abbildung 15: Generieren von Diagrammen Sequenzdiagramm

Dateien, welche mithilfe der «Export»-API zurück beim Benutzer landen, werden immer als «Spring Resource» zurückgegeben. Diese wird in Frontend als «Blob» behandelt.

Die Exportierfunktion sucht dabei lediglich die neuste Datei mit dem gewünschten Dateientyp innerhalb der eben erstellten Dateien im «src-gen» Ordner und gibt diese zurück. Die Auswahl der Typen werden nach dem Generieren als Antwort dem UI gesendet.

Der Ablauf des Exports ist deutlich simpler als das Generieren aufgebaut:

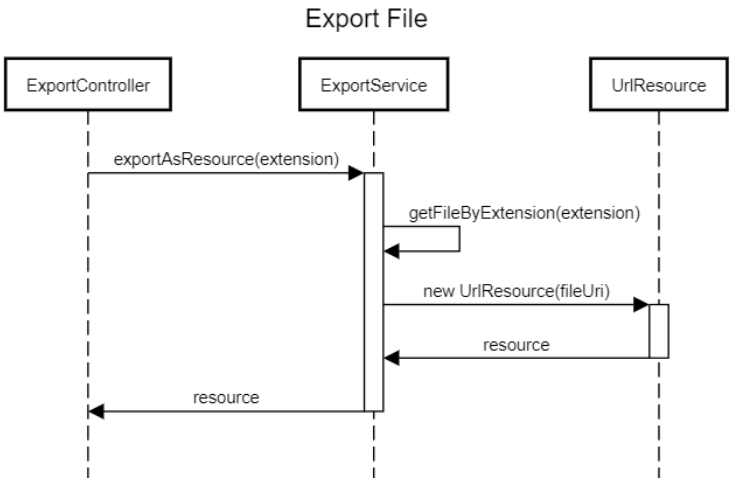


Abbildung 16: Export von Dateien Sequenzdiagramm

Für Generatoren, welche mehrmals den gleichen Dateientyp ausgeben, muss noch eine Lösung gefunden werden, wie zum Beispiel das Schicken einer ZIP-Datei mit allen Dateien oder dass der Benutzer diese auswählen kann. Zum Zeitpunkt des MVPs wird nur eine Datei pro Typ unterstützt.

«PlantUML» ist hierbei ein Problem, da der Generator mehrere «*.puml» Dateien generiert.

3.4.1.4.4 Testing

Für das Testing wurden übliche JUnit5 Testfälle und MockMVC Integrationstests geschrieben, welche wie folgt aussehen könnten:

```
@Test
public void exportAsPng_200() throws IOException {
    GenerateResponse res = contextMapperGeneratorService.generate(Te-
stUtils.cmlAsString(), GeneratorType.CONTEXTMAP);
    String extension = "png";

    Resource png = exportService.exportAsResource(extension);

    // Get actual file from directory
    final String DEFAULT_GEN_DIR = "./src-gen";
    File srcGenDir = new File(DEFAULT_GEN_DIR);
    File generatedPng = Arrays.stream(Objects.requireNonNull(srcGenDir.listFi-
les()))
            .filter(f -> f.getName()
                    .contains(extension))
            .toList().get(0);

    assertEquals(generatedPng.length(), png.contentLength());
}


@Test
public void givenGeneratorData_thenItProvidesJsonResponse() throws Exception {
    GenerateRequest req = new GenerateRequest(GeneratorType.CONTEXTMAP, Te-
stUtils.cmlAsString());

    this.mockMvc.perform(MockMvcRequestBuilders.post("/generate")
                    .content(asJsonString(req))
                    .contentType(MediaType.APPLICATION_JSON)
                    .accept(MediaType.APPLICATION_JSON))
            .andExpect(status().isOk())
            .andExpect(jsonPath("$.possibleExportFormats[0:3]", containsInAnyOr-
der("gv", "png", "svg")));
}
```

Abbildung 17: Backend Testing

3.4.2 Frontend

Das Frontend ist die Benutzerschnittstelle, welche es dem Benutzer ermöglicht mit den Backend-APIs zu kommunizieren.

Im Frontend wird im Rahmen dieses Projekt lediglich eine simple Benutzerschnittstelle gebaut, die aus einem Editor und einem Bilderanzeigefeld, sowie den jeweiligen Buttons besteht.

Als weitere Kann-Kriterien wurde «Syntax Highlighting» und eine «Real-Time Aktualisierung» implementiert.

3.4.2.1 Die wichtigsten Fakten

Das Frontend wird mit dem Angular Framework implementiert und hat an sich keine Drittabhängigkeiten. Es besteht neben den Angular Libraries aus Bootstrap für das Design, «Ace» als Editor, Cypress als Testing Framework und kleineren Abhängigkeiten, welche Funktionen wie das Speichern von Dateien übernehmen.

3.4.2.1.1 Angular

Für die Umsetzung des Frontends wurde sich vom Auftraggeber gewünscht, dass man dieses mit einem «state-of-the-art» Framework umsetzt. Da ich bereits Erfahrung mit Angular habe und dieses Framework in der Schweiz weit verbreitet ist, habe ich mich für Angular entschieden. Das Angular Framework ist eine TypeScript-basiertes Webapplikationsframework, welches alles Nötige für die Erstellung einer «Single-Page-Webanwendung» out-of-the-box unterstützt.

Für dieses Projekt wurde «Angular» mit der neusten Version «14.2.x» verwendet.

Weitere Informationen unter: https://angular.io/.

3.4.2.1.2 Cypress

Cypress ist ein für JavaScript entwickeltes End-to-End Testing Framework, welches Benutzereingaben nachahmen kann. Es wird über eigene «.cy» Dateien gesteuert, welche Anweisungen beinhalten. Darunter sind beispielsweise Anweisungen, um Textfelder auszufüllen oder Buttons zu drücken.

Weitere Informationen unter: https://www.cypress.io/.

3.4.2.1.3 Sonstige

Da es «state-of-the-art» ist «Node» für JavaScript/TypeScript basierte Anwendungen zu benutzen, wurde auch dieses für dieses Projekt benutzt. Da der Package Manager «Yarn» eine sehr gute Kompatibilität mit Node generell hat und viele Verbesserungen zum hauseigenen Packet Manager von Node «npm» hat, wurde sich für «Yarn» anstatt «npm» entschieden.

Für das Projekt wurde Node «18.9.0» benutzt, da die Kompatibilität mit Angular sehr wichtig ist.

3.4.2.1.4 Docker

Anders als beim Backend hat das Frontend keine Drittabhängigkeiten, ausser den Standardabhängigkeiten. Damit das Projekt allerdings durchgehend konsistent durchgezogen ist, wird auch für das Frontend ein Docker Image gebaut.

Dafür wird zudem ein Service in der «docker-compose» Datei angelegt. Das GUI ist über Port 80 erreichbar und kann dadurch via «localhost» (ohne Port) erreicht werden.

```
gui:
  build:
    context: ../
    dockerfile: docker/Dockerfile-Angular
  ports:
    - "80:80"
```

Abbildung 18: Docker-Compose gui

3.4.2.2 Aufbau & Design

Für den Aufbau des Frontends gab es keine Designanforderungen anders als, dass das Logo vorhanden ist. Ich habe mich allerdings dafür entschieden viele Teile der «Context Mapper» Webseite (https://contextmapper.org/) zu übernehmen, weshalb die Seite auf dem «Bootstrap» 3 Design «Bootswatch Paper» (https://bootswatch.com/3/paper) aufgebaut ist. Nebenbei wurde wie auf der Hauptseite das «Github» mithilfe von einem «Font Awesome» (https://fontawesome.com/) Icon verlinkt.

Das Design der Gesamtwebseite ist grundsätzlich sehr minimal aufgebaut und besteht nur aus einer Komponente mit Editor und Bildanzeige, sowie einer Komponente für eine grössere Darstellung des Bildes. Die Navigation ist eine eigene Komponente und ist getrennt vom Hauptinhalt.

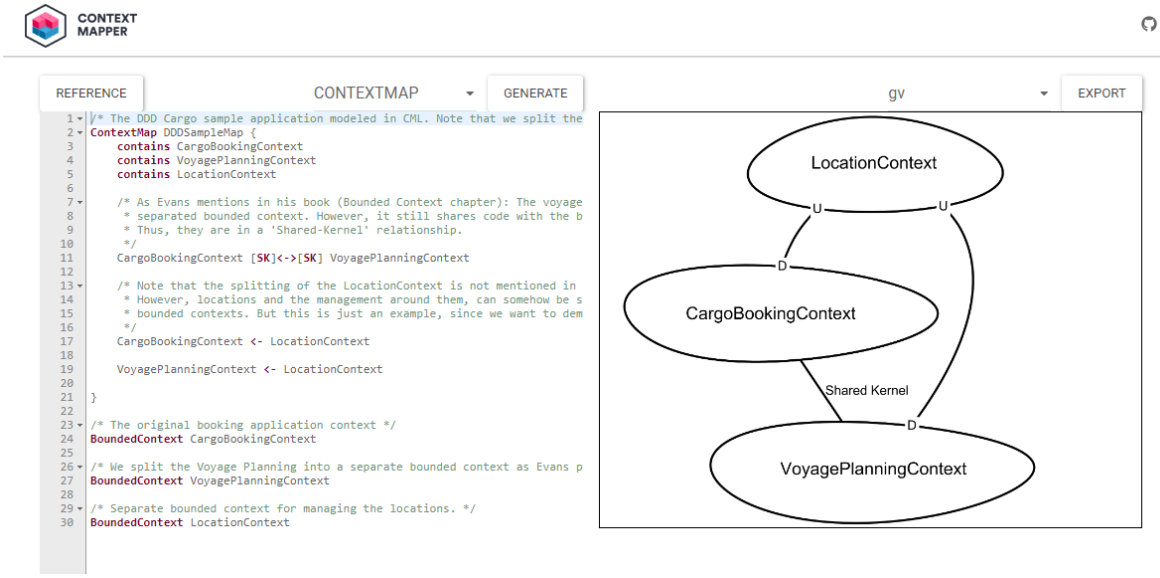


Abbildung 19: GUI

3.4.2.2.1 «Overview»-Komponente

Das Herzstück der Oberfläche ist die «Overview» Komponente, welche man in Abbildung 19: GUI sieht. Sie besteht aus zwei verschiedenen Containern. «Editor» für den Code und «Generate» für die Darstellung des generierten Diagramms.

3.4.2.2.1.1 Editor

Neben dem Editor bietet dieser Container noch einen Knopf für die Weiterleitung an eine Anleitung/Referenz, wie man die CML (Context Mapper Language) schreibt, ein Dropdown Menü, welches momentan nur den «CONTEXTMAP» Generator unterstützt, aber später weitere unterstützen soll und einen «Generate» Button, der eine Anfrage an das Backend schickt. Der Editor hat beim Erstaufruf der Seite standardmässig bereits ein Beispiel-Codesegment geschrieben.

Der benutze Editor ist der im Variantenentscheid ausgewählte «Ace».

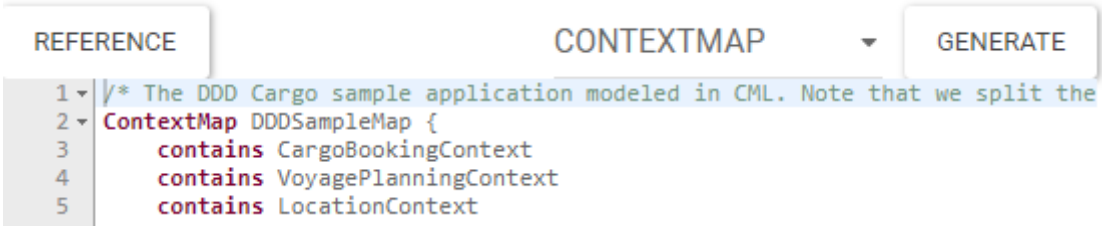


Abbildung 20: GUI Editor

Eine Referenz, wie solcher Code aufgebaut ist, kann auf der offiziellen «Context Mapper» Webseite[5] gefunden werden.

[5] https://contextmapper.org/docs/language-reference/

3.4.2.2.1.1.1 Kann-Kriterium: «Syntax Highlighting»

Der Editor soll in Zukunft nicht nur ein Textfeld sein, sondern auch Funktionen wie «Syntax Highlighting» und «Content Assist» unterstützen.

Der Editor «Ace» arbeitet mit «Modes», bei welchen es sich grundsätzlich um Spezifikationen einer Sprache handelt. Diese können mithilfe eines «Mode Creator»[6] Tools, welches «Ace» bereitstellt selbst geschrieben werden.

Glücklicherweise können diese auch direkt aus einer eigenen «DSL» mithilfe des «Eclipse» Plugins «DSL Forge»[7] generiert werden. Dieses Plugin generiert zudem das entsprechende Design für «Syntax Highlighting», sowie einem «Worker», welcher für «Context Assist» verantwortlich wäre, wenn es vollständig konfiguriert ist.

Die generierten Scripts müssen danach lediglich in Angular eingebunden und nach dem Initialisieren der View konfiguriert werden:

```
import '../../../assets/ace/mode-contextmappingdsl.js';
import '../../../assets/ace/theme-contextmappingdsl.js';
import '../../../assets/ace/worker-contextmappingdsl.js';


  ngAfterViewInit(): void {
    this.generateDiagram();
    ace.config.set('fontSize', '18px');

    const aceEditor = ace.edit(this.editor.nativeElement);
    aceEditor.session.setValue(this.content);

    aceEditor.session.setMode('ace/mode/contextmappingdsl');
    aceEditor.setTheme('ace/theme/contextmappingdsl');

    aceEditor.on('change', () => {
      this.content = aceEditor.getValue();
      this.generateDiagram();
    });
  }
```

Abbildung 21: Konfiguration Editor

Der Editor benutzt mit dieser Konfiguration den «contextmappingdsl»-Mode und das entsprechende Theme.

Die Schwierigkeit bei diesem Feature lag dabei, dass es nicht ganz simpel ist mit «Eclipse»-abhängigen Dingen zu arbeiten. Zudem würde «Eclipse» eine ähnliche Funktion bieten, welche aber nicht «out-of-the-box» Angular-kompatibel ist.

[6] https://ace.c9.io/tool/mode_creator.html
[7] https://dslforge.org/getting-started-generate-xtext-rap-editor/

3.4.2.2.1.1.2 Kann-Kriterium: «Real-Time Aktualisierung»

Damit der Benutzer nicht jedes Mal manuell auf den «Generate» Button drücken muss, sollen Diagramme in «real-time» aktualisiert werden.

Frontendtechnisch muss man dafür lediglich ein «onChange»-Event Handler machen, welcher bei bestimmten Voraussetzungen ein «generate»-Anfrage absetzt.

```
aceEditor.on('change', () => {
      this.content = aceEditor.getValue();

      clearTimeout(this.timer);
      this.timer = setTimeout(() => {
        this.generateDiagram();
      }, 1000);
    });
```

Abbildung 22: On-Change Event

Die Schwierigkeit hier ist, dass der Generator im Backend manchmal den vielen Anfragen nicht hinterherkommt, weshalb vorerst ein Buffer von zehn (10) Dateien eingebaut wurde und ein Timeout von einer Sekunde, welcher sich jeden Tastenanschlag zurücksetzt und auslöst, sobald für eine Sekunde kein Input mehr kommt.

Sobald der Editor alle Funktionalitäten von «Xtext» unterstützt, kann die Validität des Codes geprüft werden, bevor eine Abfrage abgesetzt wird.

3.4.2.2.1.1.3 Kann-Kriterium: «Autocompletion» - Nicht abgeschlossen

Wie im Kapitel «Syntax Highlighting» beschrieben soll der Editor in Zukunft viele Funktionalitäten bieten, welche auch eine «IDE» dem Entwickler bietet. Dazu gehört «Content Assist» oder ander gesagt «Autocompletion» dazu.

Im Rahmen des Projektes und der verfügbaren Zeit wurde versucht diese Umzusetzen. Leider war die Zusammenarbeit zwischen «Eclipse Xtext» und einem «Angular» Frontend zu komplex und hätte den Rahmen gesprengt.

Wenn man sich die Architektur nochmals anschaut und die Kommunikation zwischen Editor und «Lanugage Server» anschaut, hat man eine direkte Kommunikation vom GUI zu «Xtext» basierten Schnittstellen.

Weder die Schnittstellen noch die Services, welche die Logik ausführen sind schnell ersichtlich und man müsste sich tief in den Source Code eingraben. Zudem wurde die «Context Mapper» DSL ursprünglich ohne Web-Support generiert.

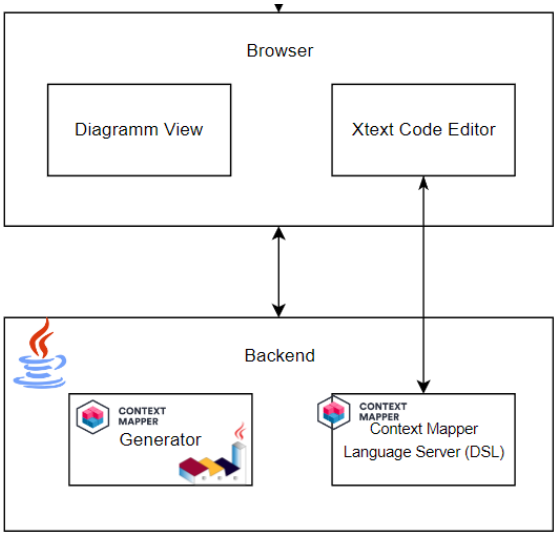


Abbildung 23: Architektur Kommunikation Editor <-> Language Server

Da «Eclipse» standardmässig viele Dinge einer mit «Xtext» erstellten «DSL» generiert passiert sehr viel «Magic» im Hintergrund, weshalb der Entwickler nicht auf die Schnelle alle «W-Fragen» beantworten kann. Ausnahmen sind Entwickler, welche bereits viel damit gearbeitet haben.

Zudem werden nötige Files in JavaScript-Format generiert und greifen oft auf «RequireJS», einem JavaScript File und Modul «Loader», zurück, welcher nicht «out-of-the-box» Kompatibel mit dem «Loader» von «Angular» ist.

Die Umsetzung ist definitiv möglich, allerdings mit der Überlegung, ob man sich die Zeit nehmen will, sich mit den «Loadern» von Angular und die Kompatibilität zu sogenannten «AMD» Modulen zu befassen oder ob man lieber auf ein anderes Framework oder Bibliothek wie «VueJS» ausweicht.

Die Herausforderung wird noch grösser, weil es genau für das Thema («Xtext» + «AMD» + «Angular 2» keine Ressourcen im Internet gibt).

3.4.2.2.1.2 Generate

Neben der Bildanzeige bietet dieser Container ein Dropdown Menü für die Auswahl eines Exportformats an und den «Export» Button, welcher eine Anfrage an das Backend schickt, um das jeweilige Format runterzuladen. Beim «CONTEXTMAP» Generator wird standardmässig das «png» Format geholt und dargestellt.

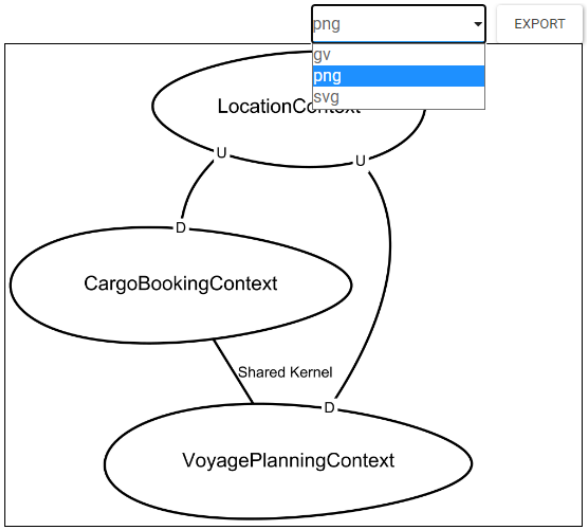


Abbildung 24: Exportformat Auswahl

3.4.2.2.2 «Expand-Image» Komponente

Diese Komponente hat die Funktion das Bild zu vergrössern, wenn man auf das Bild klickt. Sie wird mit einem «Flag» aktiviert und deaktiviert und kann entweder über das «x» oder wenn man ausserhalb der Vergrösserung klickt, wieder geschlossen werden.

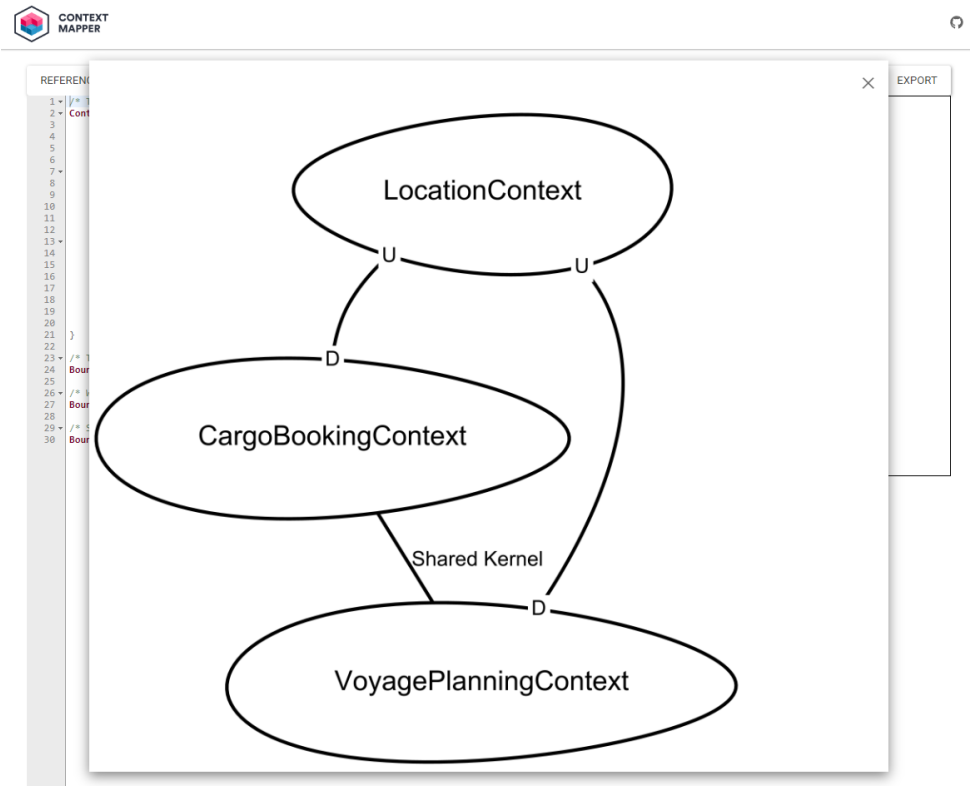


Abbildung 25: Expand Image GUI

3.4.2.2.3 Data-Service

Der Data-Service ist verantwortlich für die ganzen Anfragen und Antworten des Backends. Er sendet mithilfe der «HttpClient» Library Anfragen an das Backend und empfängt diese.

Damit die Antworten bereits im vorgefertigten Format ankommen, wurde mit Angular Interface Models gearbeitet, welche gleich aussehen, wie die Models im Backend. Die «Overview»-Komponente greift direkt auf den Service zu. Der Code wird vor dem Schicken in das Backend jeweils «escaped», damit Sonderzeichen auch als String behandelt werden.

Die Export Anfrage ist beispielsweise so aufgebaut:

```
public export(req: ExportRequest): Observable<Blob> {
    return this.http.get(this.EXPORT_URL + '/' + req.exportFormat, {
      responseType: 'blob',
    });
  }
```

Abbildung 26: API GUI

3.4.2.2.4 Testing

Testing im Frontend ist klein gehalten und mit «Cypress» umgesetzt.

«Cypress» erlaubt E2E-Testing, welches einen vordefinierten Ablauf durcharbeitet. Für das GUI wurde ein Testfall geschrieben, welcher eine «ContextMap» im Editor mit zwei Beispiel-«Bounded Context» definiert. Das Diagramm daraus wird in einem zweiten Schritt generiert und in einem dritten Schritt runtergeladen.

Die Abläufe werden in dieser Form geschrieben:

```
it('test generation and export', function () {
    cy.viewport(1920, 1009);
    cy.visit('http://localhost:4200/');
    cy.wait(1000);
    cy.get('.ace_text-input')
      .first()
      .focus()
      .clear()
      .type(
        'ContextMap Sample {\n contains Sample1\n contains Sample2\n Sample1 [SK]<->[SK]
Sample2 } \n BoundedContext Sample1 \nBoundedContext Sample2',
        { parseSpecialCharSequences: false }
      );
```

Abbildung 27: Cypress Test Anweisungen

Schlussendlich werden diese mit dem Framework umgesetzt, als würde ein richtiger Benutzer die Oberfläche bedienen.

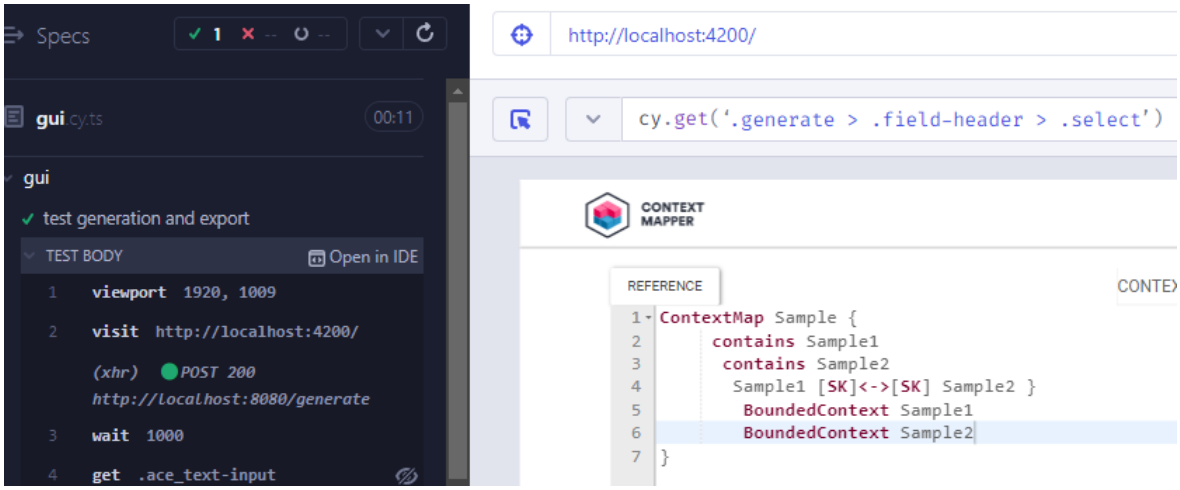


Abbildung 28: Cypress Test Interface

3.4.2.3 Standardablauf

Der Standardablauf der Applikation ist minimal und damit intuitiv aufgebaut. Der Zyklus kann sich so oft wiederholen, bis der Anwender zufrieden ist. Der Ablauf geht davon aus, dass es nur einen Generator zur Auswahl gibt. In Zukunft wird daher ein weiterer Schritt benötigt, wenn weitere Generatoren implementiert wurden.

Der Schritt «Generate Knopf drücken» wird mit der «real-time» Aktualisierung zudem überholt.

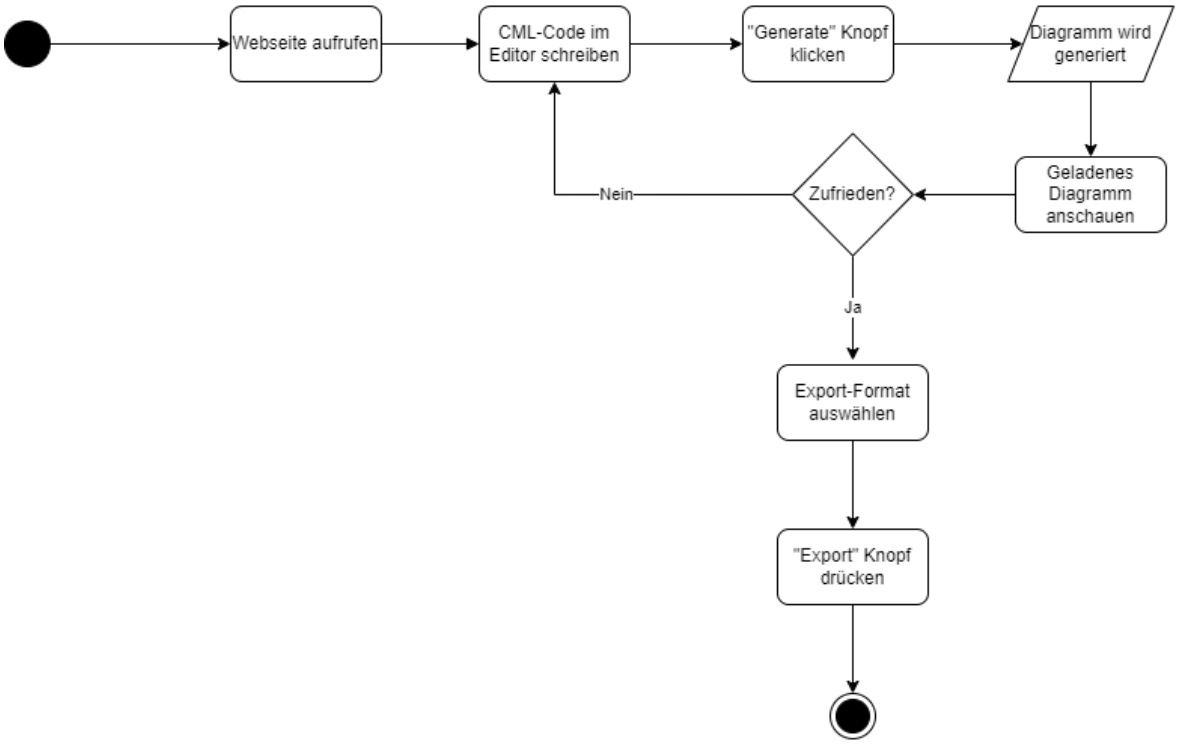


Abbildung 29: Standardablauf GUI

3.4.3 Evaluation der Testerfüllung

In diesem Kapitel werden die im Testkonzept erwähnten Testfälle darauf abgeglichen, ob diese bei einem manuellen Test erfolgreich waren. Dabei werden sie Schritt für Schritt wie im Konzept befolgt.

ID	Bezeichnung	Bemerkung	Ergebnisse
TF-M1	Korrektes Laden der Inhalte	-	OK
TF-M2	Textfeld lädt und ist bearbeitbar	-	OK
TF-M3	Generieren eines Diagramms	-	OK
TF-M4	Exportieren eines Diagramms	-	OK
TF-K1	Generieren eines Alternativ-Diagramms	Wurde aus Zeitgründen nicht umgesetzt	-
TF-K2	IntelliSense	Wurde aus Zeitgründen nicht umgesetzt	-
TF-K3	Realtime Generierung		OK

Tabelle 16: Testerfüllung

Alle Tests haben funktioniert, wie sie geplant waren mit Ausnahme von zwei «Kann»-Kriterien Testfällen, welche aus Zeitgründen nicht umgesetzt wurden. Für «TF-K1» wurde allerdings der Code vorbereitet, damit neue Generatoren möglichst einfach implementiert werden können.

3.4.4 Evaluation der Zielerreichung

In diesem Kapitel werden alle Erfolgskriterien und Endergebnisse mit dem tatsächlichen Resultat abgeglichen.

Erfolgskriterien	Endergebnisse	Bemerkung	Status
Die Applikations-Oberfläche enthält ein Editor Feld mit einem Beispiel Code Segment	Bei Aufrufen der URL lädt die Oberfläche mit einem Beispiel Code Segment im Editor Feld, welches wie ein Textfeld bearbeitbar ist. Das Beispiel Code Segment wird bei Erstaufruf als «Context-Mapper» Diagramm generiert und dargestellt	Bei Aufruf der Seite existiert ein «Ace» Editor Feld und das Diagramm wird ohne zusätzliche Benutzerinteraktion generiert.	✓
Ein Diagramm ist durch einen «Generate» Button generierbar.	Wenn der «Generate» Button auf der Oberfläche gedrückt wird, und das Code Segment im Editor valide ist, wird ein aktualisiertes Diagramm auf der rechten Seite ausgegeben.	Der Button ist normalerweise unnötig, da Diagramme «real-time» generiert werden, kann allerdings bei Systemfehlern angewendet werden.	✓
Das momentan generierte Diagramm ist durch einen «Export» Button und einem Format Auswahlfeld exportierbar.	Wenn ein vom Generator unterstütztes Exportformat ausgewählt und der «Export» Button gedrückt wird, startet der Download im jeweiligen Format	Funktioniert problemlos, kann allerdings Browserabhängig sein und daher teilweise nicht steuerbar von der App.	✓

Tabelle 17: Zielerreichung

Alle Erfolgskriterien und Endergebnisse konnten erfolgreich erfüllt werden. Die Webapplikation macht, was sie in einem ersten Entwurf machen soll, und ist für die Zukunft einfach und weiter ausbaufähig.

3.4.5 Berechnung / Wirtschaftlichkeit

Das Richtziel für das Projekt war «*Domain-Driven Modelling mit dem Context Mapper via Webapplikation verfügbar machen*» und beschreibt das Ziel den «Context Mapper»[8], welcher momentan als «Eclipse» und «Visual Studio Code» Plugin existiert und fähig ist Diagramme aus einem Code aus einer eigenen «DSL» (domain-specific language) in Diagramme umzuwandeln, im Web verfügbar zu machen, damit man solche Diagramme auch ohne Hilfe einer Entwicklungsumgebung generieren kann.

Die Webapplikation wurde als Erstentwurf abgeschlossen und erlaubt es die Diagramme mit dem hauseigenen Generator zu generieren. Die Kriterien für die Umsetzung des Projekts, mit Ausnahme des Editors, waren festgelegt, erfordern daher keinen Variantenvergleich mit anderen Methoden.

Sie erlaubt jeder Einzelperson, sowie Unternehmen eine Applikation zu starten, welche den «Domain-Driven Design» Prozess um einiges verschnellert und modularer macht.

Für die Umsetzung wurden 150 Stunden eingesetzt, bei welchen über das Richtziel hinausgeschossen wurde. Es existieren Kriterien und Lösungsansätze, welche die Weiterentwicklung erleichtern. Zudem kann sich jeder daran beteiligen.

Das Projekt wurde nicht mit dem direkten Ziel umgesetzt Gewinn in Form von Geld zu machen, kann allerdings die Effizienz des Prozesses erhöhen und somit Kosten sparen.

Die Webapplikation ist für jeden öffentlich unter folgendem Link zugänglich:

https://github.com/Akagitsunee/context-mapper-web-app

Das Richtziel wurde damit erfüllt. Die Zielerreichung ist in Tabelle 17: Zielerreichung ersichtlich.

[8] https://contextmapper.org/

3.4.6 Reflexion / Lesson learnt

Ich wollte ursprünglich mehr Anforderungen in die Arbeit reinnehmen, als es jetzt gibt. Mir wurde allerdings davon abgeraten, dass man die Arbeiten nicht unterschätzen sollte, auch wenn diese klein klingen.

Da ich bereits professionell tätig im Bereich des «Software Engineerings» bin, war mir bereits klar, dass es zwischendurch Aufgaben gibt, die ein riesiges Gerüst dahinter verstecken, wie auch simple Webseiten nicht gleich simpel sein müssen. Allerdings wollte der Perfektionist aus mir raus und zeigen, dass man alles in dieser Zeit umsetzen kann.

Ich habe zu der Zeit Variablen wie die persönliche Gesundheit aus dem Fenster geworfen und bin vom Besten ausgegangen, wurde allerdings schlussendlich überzeugt die Anzahl Ziele zu kürzen, da man diese auch als Zusatz reinnehmen kann.

In Rückblick bin ich froh darüber, dass man die obligatorischen Anforderungen gekürzt hat, da genau in dieser Zeit alle anderen Probleme ausserhalb des Projektes auf mich zu kamen.

Ich nehme für die Zukunft mit, dass das Stichwort «weniger ist mehr» in manchen Situationen gar nicht so falsch ist und man im Zusammenhang mit einem Softwareprodukt lieber zuerst eine gute Basis schafft, bevor man zum «fancy Stuff» übergeht.

In Retrospektive habe ich die Zeitplanung zu meinem eigenen Erstaunen allerdings sehr gut gemacht. Ich habe viele Aufgaben nicht auf die bestmögliche Zeit geplant, sondern auf eine realistische Zeit, in der nicht immer alles gleich funktioniert oder auch mal der Denkanstoss / die Inspiration fehlt. Ich habe damit allerdings bereits auf der Arbeit angefangen anstatt 10 Minuten für eine «kurze Änderung» einfach mal 30 Minuten einzuplanen, damit diese «kurz» geändert werden kann, aber danach auch funktioniert. Die Planung hat es mir ermöglicht trotz 16 Krankheitstagen mit 8 von diesen Tagen komplett arbeitsunfähig, die ganzen Muss- und fast alle Kann-Kriterien auszuführen.

Ich hatte anfangs Schwierigkeiten anzufangen, bis mal eine Substanz da war, aber konnte mich danach gut durch die 4-Phasen und somit die ganze Arbeit schlängeln. Die Initialisierungs- und Planungsphase verlief mehr oder weniger problemlos und bei der Realisierung kam es dann zu den ersten Problemen.

Ich bin grösstenteils ein «Backend/DevOps Engineer», habe daher keine grosse Erfahrung mit UI/UX, was sich auch gezeigt hat. Die Frontend Welt ist riesig. Auch hatte ich Schwierigkeiten mit der Verwendung älterer Technologien wie «Eclipse Xtext». Ich habe auf dem Weg des «Software Engineers» noch vieles zu lernen, was auch den Spass ausmacht. Ich bevorzuge allerdings weiterhin den agilen Weg, da ich oftmals lieber Frameworks und weiteres während der Arbeit geändert hätte.

Ich bin stolz darauf, dass alles trotz der Umstände geklappt hat und nehme mit, dass ich mich lieber auf das wesentliche konzentriere, bevor ich zu allem übergeordneten übergehe. Auch dass ich in Zukunft mehr Priorität auf meine eigene Gesundheit setze.

4 Projektabschluss

Der Projektabschluss beinhaltet eine Zusammenfassung über das ganze Projekt sowie einige Schlussworte.

4.1 Zusammenfassung

Die Zusammenfassung befasst sich mit den Inhalten über das ganze Projekt mit dem Richtziel «*Domain-Driven Modelling mit dem Context Mapper via Webapplikation verfügbar machen*». Sie umfasst die wichtigsten Punkte der gesamten Arbeit.

Ausgangslage

Bei der Strukturierung von Software und deren Komponenten gibt es diverse Herangehensweisen, Architekturen und Methoden. Eine sehr beliebte Methode ist das Domain-Driven Design oder kurz DDD, welches das Wissen und die Sprache der Fachdomäne ins Zentrum stellt. Domain-Driven Design stellt den Fokus auf die Verwendung von einer ubiquitären Sprache, damit in allen Bereichen, von Business, Requirements Engineer, bis zum Entwickler die gleiche Sprache gesprochen wird.

Um Domain-Driven Design optimal zu implementieren, gibt es Tools, um Architektur und Domänenmodelle visuell darzustellen. Eines dieser Tools ist das «Context Mapping» welches es erlaubt den Zusammenhang zwischen abgegrenzten Kontexten (Bounded Context) zu identifizieren.

Für die Generierung solcher «Context Maps» gibt es Frameworks wie den «Context Mapper» (contextmapper.org), welcher als Eclipse oder Visual Studio Code Plugin kommt und mithilfe einer domänenspezifischen Sprache (DSL) das Generieren solcher «Context Maps» ermöglicht.

Problemstellung

Da das Erstellen von «Context Maps» nicht nur den Entwicklern vorbehalten sein soll, braucht es eine Lösung, welche unabhängig von Entwicklungsumgebungen ist.
Eine Lösung, um dieses Problem zu beheben, ist es die Generierung von «Context Maps» auf dem Web mithilfe einer Webapplikation möglich zu machen.

Da diese Umsetzung in diesem spezifischen Kontext noch nicht existiert und keine genaue Dokumentation vorhanden ist, kam die Idee auf, dass sich das Ganze als Diplomarbeit eignen könnte.

Das Spezielle ist, dass die Arbeit im Auftrag der mimacom ag umgesetzt werden soll, aber das Endergebnis quellenoffen veröffentlich wird. Es handelt sich hierbei um einen Hybridprojekt welches sowohl firmenintern wie auch gemeinnützig genutzt werden kann.

Ziele

Die Endergebnisse und Erfolgskriterien auf einen Blick:

Endergebnisse	Erfolgskriterien
Die Applikations-Oberfläche enthält ein Editor Feld mit einem Beispiel Code Segment	Bei Aufrufen der URL lädt die Oberfläche mit einem Beispiel Code Segment im Editor Feld, welches wie ein Textfeld bearbeitbar ist. Das Beispiel Code Segment wird bei Erstaufruf als «Context-Mapper» Diagramm generiert und dargestellt.
Ein Diagramm ist durch einen «Generate» Button generierbar.	Wenn der «Generate» Button auf der Oberfläche gedrückt wird, und das Code Segment im Editor valide ist, wird ein aktualisiertes Diagramm auf der rechten Seite ausgegeben.
Das momentan generierte Diagramm ist durch einen «Export» Button und einem Format Auswahlfeld exportierbar.	Wenn ein vom Generator unterstütztes Exportformat ausgewählt und der «Export» Button gedrückt wird, startet der Download im jeweiligen Format

Tabelle 18: Ziele

Vorgehen

Das Projekt wurde nach dem 4-Phasenmodell in der Zeitspanne vom 12. September 2022 bis zum 24. Oktober 2022 aufgebaut. Das gesamte Projekt wurde von einer Person im Auftrag eines Auftraggebers aufgebaut. Es entstanden über das ganze Projekt keine Kosten, da das Produkt vollständig quelloffen ist.

Vor Beginn der Arbeit wurde ein Pflichtenheft geschrieben, welches alle Anforderungen beinhaltete. Die festgelegten Anforderungen waren dabei ein «Angular» Frontend mit Textfeld und Diagrammanzeigefeld, sowie ein, wenn möglich, Spring Backend.

Im Projekt wurden die vier Schritte «Initialisierung», «Planung», «Realisierung» und «Abschluss» der Reihe nach umgesetzt.

Zu Beginn wurden Projektstruktur- und Terminplan sowie eine grobe Dokumentstruktur erstellt. In einem zweiten Schritt wurden alle Angaben zum Projektleiter in Form von einem Qualifikationsprofil und einem Lebenslauf niedergeschrieben. In einem dritten Schritt alle Informationen zum Projekt.

Im Projekt wurden Methoden und Instrumente wie eine Zielscheibe, Risikoanalysen, Testkonzepte, Variantenentscheide mit Präferenzmatrix, Nutzwertanalyse und Sensitivitätsanalyse angewendet. Weiterhin UML für die Darstellung von Abläufen und Zusammenhängen in Form von Diagrammen.

Ergebnisse

Es gibt eine vollumfängliche Dokumentation über die Umsetzung einer Webapplikation, die es ermöglicht «Context Maps» aus der von «contextmapper.org» erstellten «DSL» (domain-specific language») in Form von Diagrammen zu generieren.

Zusätzlich ist der Code vollständig unter diesem Link erhältlich:

https://github.com/Akagitsunee/context-mapper-web-app

Alle Konzepte, Abläufe und die wichtigsten Zusammenhänge wurden in diesem Dokument dokumentiert. Alle Ziele wurden genau nach der Beschreibung der Ziele erreicht und dokumentiert.

Es wurde nach vollen Vorgaben der Anforderungen ein MVP entwickelt, welcher den durch Variantenentscheid bestimmten «Ace»-Editor (Ajax.org Cloud 9 Editor) als Herzstück der Oberfläche benutzt. Der geschriebene Code im Editor kann durch einen Knopf auf der Oberfläche in das Backend geschickt werden, worauf dieses diesen Code in eine temporäre Datei speichert. Diese Datei wird mit der bereitgestellten API vom «Context Mapper» von «contextmapper.org» in einen Generator gespiesen, welcher Diagramme in Form von z.B. Bildern («*.png», «*.svg») mithilfe von «Graphviz» generiert.

Standardmässig wird nach Abschluss der Generierung eine Anfrage geschickt, welche die «.*png» Datei auf die Oberfläche holt und diese darstellt. Das Bild kann durch einen Klick darauf vergrössert werden. Die zusätzlich generierten Formate können durch eine Auswahlbox und einem weiteren Knopf auf der Oberfläche runtergeladen werden. Temporäre Dateien werden laufend gelöscht, um Platz zu sparen.

Die Webapplikation kann als «Tomcat»-Server und «Angular»-Applikation gestartet werden oder innerhalb von Docker Containern. Die Applikation ist durch ein Start-Script startbar, welches in einem README innerhalb der Code Base dokumentiert wurde.

Das Projekt wurde ohne grosse Abweichungen durchgezogen.

Die Applikation kann von jeder Einzelperson und von jedem Unternehmen benutzt werden. Es steht jedem frei zur Verfügung einen eigenen Server darauf aufzubauen.

Die Software erleichtert das Erstellen von «Context Maps» im Zusammenhang mit Domain-Driven Design, da es die Anforderung an eine Entwicklungsumgebung abschafft.

4.2 Ausblick

Die Applikation läuft auf dem Stand vom MVP vollumfänglich mit dem Generator und der «DSL» von «contextmapper.org».

Die Entwicklung der Applikation gilt allerdings nicht als abgeschlossen, da geplant ist, dass weitere Generatoren wie z.B. «PlantUML», «MDSL» und ein «Generic» Generator unterstützt werden.

Der Editor soll in Zukunft zusätzlich so ausgebaut werden, dass dieser den vollen Nutzen aus der von «Eclipse Xtext» generierten «DSL» ziehen kann. Dazu gehören alle Features dazu, welche im Variantenentscheid in der Abbildung 6: Features per Editor erwähnt sind.

Als Alternative für den Editor können beide Editoren, welche im Variantenentscheid erwähnt, sind implementiert werden. Zusätzlich empfiehlt sich die Überlegung von «Angular» auf «VueJS» zu wechseln, da dieses die von «Xtext» generierten Script besser unterstützt.

Der Editor und die Bildanzeige sollen eventuell durch einen «Split»-Regler separiert werden, damit der Benutzer die Komponente grösser machen kann, welche er benutzen will. Die Schriftgrösse des Editors soll einstellbar sein.

4.2.1 Known Bugs

Der Editor hat einen bekannten Bug, dass er auf manchen Bildschirmauflösungen den Code auf die ersten paar Zeilen schiebt und nur diese scrollbar macht. Der Bug kann leider nicht nachgestellt werden, aber soll in zukünftigen Versionen gelöst werden.

Die «real-time» Aktualisierung von Diagrammen funktioniert zwar, jedoch kommt der Generator manchmal den vielen Anfragen nicht nach. Es kann dazu führen, dass bei der finalen Generierung zu wenige Files generiert werden und nur eine beschränkte Anzahl verfügbar ist. Das Problem kann momentan über ein einfaches Drücken auf den «Generate» Button gelöst werden und wird in der Zukunft mit mehr Editorfeatures gelöst. Der Bug sollte jedoch nur sehr selten vorkommen.

4.3 Persönliches Schlusswort

Die Zeit über die gesamten 6 Wochen war für mich eine sehr schwierige Zeit, da ich stark mit gesundheitlichen Problemen zu kämpfen hatte, welche sich über die Diplomarbeitszeit hinausziehen und bereits im Sommer angefangen haben. Ich hatte und anderem einen Knochenbruch, war ganze 16 Tage krank, davon 8 Tage komplett krankgeschrieben und musste daher oftmals den Zeitplan Tage nach vorne gegen Abschluss schieben, was bei der Aufgabe «Testing (Unit-/Integrationstests) (BE)» gut ersichtlich ist.

Ich bin dennoch froh und stolz, dass ich jetzt auf ein solches Resultat zurückschauen kann, auch wenn dieses noch bei weitem nicht perfekt ist.

Meine «Lesson» die ich vom Projekt mitnehme, ist im Kapitel «Lesson learnt» beschrieben.

4.4 Verdankung

Ich bedanke mich bei allen Personen, die mir bei der Arbeit zur Seite standen, und vor allem ein herzliches Dankeschön an meinen Auftraggeber Stefan Kapferer für die Idee und Unterstützung sowie an meinen Diplomlehrer Gregor von Flüe für die Unterstützung über die ganze Projektzeit.

5 Redlichkeitserklärung

Die Verfasserinnen und Verfasser bestätigen mit ihrer Unterschrift, dass die vorliegende Arbeit selbstständig, ohne fremde Hilfe und ohne Benutzung anderer als die angegebenen Hilfsmittel erstellt wurde.

Die aus fremden Quellen (einschliesslich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit ist in gleicher oder ähnlicher Form noch nicht vorgelegt worden.

Unterschriften:	Datum/Ort:
Yanick Arn 	23.10.2022

Abkürzungsverzeichnis

ABKÜRZUNG	BESCHREIBUNG
CML	Context Mapper Language und Datenformat
DSL	Domain-specific language – Formale Sprache in spezifischem Anwendungs-bereich
GUI	Graphical User Interface – Siehe UI
GV	Graphviz Dateiformat
MDSL	(Micro-)Service Contracts Generator
MVC	Model-View-Control – Pattern in der Softwareentwicklung
MVP	Minimum viable product – die Basis eines funktionellen Produktes
NPM	Node Package Manager – Packet Manager
PNG	Portable Network Graphics - Bildformat
PUML	PlantUML Datenformat
SVG	Scalable Vector Graphics – Bildformat für Vektoren
UI	User Interface – Oberfläche für die Bedienung
UML	Unified Modeling Lanugage – Vereinheitliche Modelierungssprache für die Modellierung von Software-Teilen und anderen System
UX	User Experience

Abbildungsverzeichnis

Tabellenverzeichnis

Diagrammverzeichnis

Literatur- und Quellenverzeichnis

Literaturverzeichnis

Ace C9.org. (19. 9 2022). *Ace C9 About*. Von https://ace.c9.io/ abgerufen

CodeMirror. (19. 9 2022). *CodeMirror Startpage*. Von https://codemirror.net/ abgerufen

Context Mapper / Stefan Kapferer . (12. 9 2022). *Context Mapper*. Von https://contextmapper.org/ abgerufen

Context Mapper / Stefan Kapferer . (12. 9 2022). *Context Mapper Github Organisation*. Von https://github.com/ContextMapper abgerufen

Cypress.io. (7. 10 2022). *Cypress Quickstart: Angular*. Von https://docs.cypress.io/guides/component-testing/quickstart-angular abgerufen

dslforge. (10. 2022 2022). *dslforge - Generate Web Editor from Xtext*. Von 10 abgerufen

Eclipse. (12. 9 2022). *Eclipse Xtext*. Von https://www.eclipse.org/Xtext/ abgerufen

Eclipse. (19. 9 2022). *Orion Documentation*. Von https://wiki.eclipse.org/Orion/Documentation/Developer_Guide/Plugging_into_the_editor abgerufen

Eclipse. (19. 9 2022). *Xtext Web Editor Support*. Von https://www.eclipse.org/Xtext/documentation/330_web_support.html abgerufen

Google. (19. 9 2022). *Angular.io*. Von https://angular.io/ abgerufen

Graphviz . (19. 9 2022). *Graphviz* . Von https://graphviz.org/about/ abgerufen

https://gradle.org. (19. 9 2022). *Gradle.org*. Von https://gradle.org abgerufen

Kanton Zürich. (18. 9 2022). *Hermes Zürich*. Von https://hermes.zh.ch/onlinepublikation/index.xhtml?element=ergebnis_testkonzept.html abgerufen

npm Inc / CodeMirror / marijn. (19. 9 2022). *npm codemirror*. Von https://www.npmjs.com/package/codemirror abgerufen

npm Inc. / Ace C9.org. (19. 9 2022). *npm ace-builds*. Von https://www.npmjs.com/package/ace-builds abgerufen

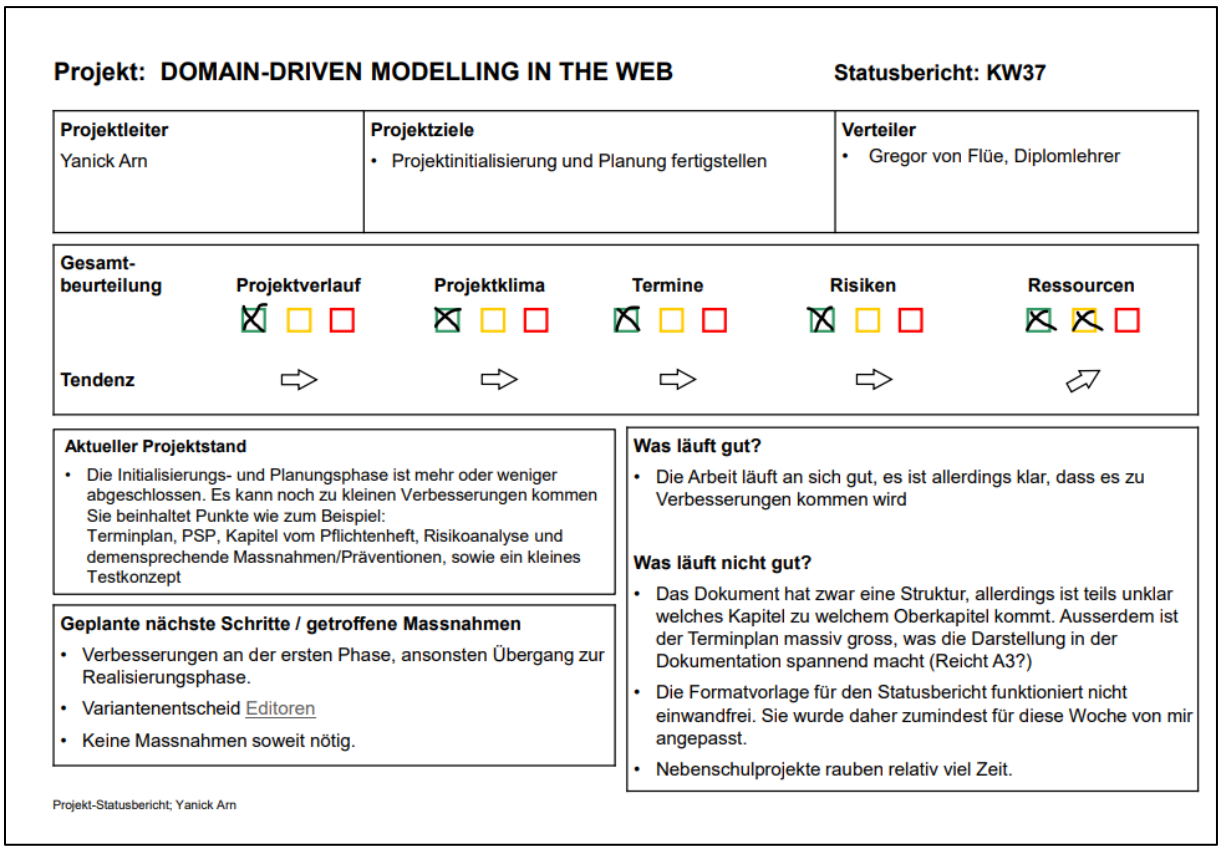
Shah, D. (10. 10 2022). *Blog - How to setup Ace editor in Angular?* Von https://blog.shhdharmen.me/how-to-setup-ace-editor-in-angular abgerufen

VMware. (19. 9 2022). *Spring.io*. Von https://spring.io/ abgerufen

6 Anhang

In diesem Kapitel sind alle Dinge zu finden, die nicht in die Projektdokumentation selbst gehören oder zu gross waren.

6.1 Projektstatusberichte



Projekt: DOMAIN-DRIVEN MODELLING IN THE WEB **Statusbericht: KW37**

Projektleiter	**Projektziele**	**Verteiler**
Yanick Arn	• Projektinitialisierung und Planung fertigstellen	• Gregor von Flüe, Diplomlehrer

Gesamt-beurteilung	**Projektverlauf**	**Projektklima**	**Termine**	**Risiken**	**Ressourcen**
	☒ ☐ ☐	☒ ☐ ☐	☒ ☐ ☐	☒ ☐ ☐	☒ ☒ ☐
Tendenz	⇨	⇨	⇨	⇨	⇗

Aktueller Projektstand

- Die Initialisierungs- und Planungsphase ist mehr oder weniger abgeschlossen. Es kann noch zu kleinen Verbesserungen kommen Sie beinhaltet Punkte wie zum Beispiel:
  Terminplan, PSP, Kapitel vom Pflichtenheft, Risikoanalyse und demensprechende Massnahmen/Präventionen, sowie ein kleines Testkonzept

Geplante nächste Schritte / getroffene Massnahmen

- Verbesserungen an der ersten Phase, ansonsten Übergang zur Realisierungsphase.
- Variantenentscheid Editoren
- Keine Massnahmen soweit nötig.

Was läuft gut?

- Die Arbeit läuft an sich gut, es ist allerdings klar, dass es zu Verbesserungen kommen wird

Was läuft nicht gut?

- Das Dokument hat zwar eine Struktur, allerdings ist teils unklar welches Kapitel zu welchem Oberkapitel kommt. Ausserdem ist der Terminplan massiv gross, was die Darstellung in der Dokumentation spannend macht (Reicht A3?)
- Die Formatvorlage für den Statusbericht funktioniert nicht einwandfrei. Sie wurde daher zumindest für diese Woche von mir angepasst.
- Nebenschulprojekte rauben relativ viel Zeit.

Projekt-Statusbericht; Yanick Arn

Projekt: DOMAIN-DRIVEN MODELLING IN THE WEB **Statusbericht: KW38**

Projektleiter	Projektziele	Verteiler
Yanick Arn	• ContextMapper im Backend implementieren	• Gregor von Flüe, Diplomlehrer

Gesamtbeurteilung	Projektverlauf	Projektklima	Termine	Risiken	Ressourcen
	☒ ☐ ☐	☒ ☐ ☐	☒ ☐ ☐	☒ ☐ ☐	☒ ☒ ☐
Tendenz	⇨	⇨	⇨	⇨	⇗

Aktueller Projektstand
- Zusätzlich zu letzter Woche wurde die Backend Implementierung des ContextMapper Generators funktionsfähig gemacht.

Was läuft gut?
- Die Arbeit läuft an sich weiterhin gut, es ist allerdings klar, dass es zu Verbesserungen kommen wird.

Was läuft nicht gut?
- Nebenschulprojekte rauben weiterhin relativ viel Zeit.
- Gesundheitliche Probleme rauben auch sehr viel Zeit
- Ich wurde krank

Geplante nächste Schritte / getroffene Massnahmen
- Dokumentation erweitern
- Frontend aufsetzen

Projekt-Statusbericht; Yanick Arn

Projekt: DOMAIN-DRIVEN MODELLING IN THE WEB **Statusbericht: KW39**

Projektleiter	Projektziele	Verteiler
Yanick Arn	• Dokumentation erweitern und Frontend aufsetzen	• Gregor von Flüe, Diplomlehrer

Gesamtbeurteilung	Projektverlauf	Projektklima	Termine	Risiken	Ressourcen
	☐ ☒ ☐	☐ ☒ ☐	☐ ☐ ☒	☐ ☐ ☒	☐ ☐ ☒
Tendenz	⇗	⇗	⇗	⇗	⇗

Aktueller Projektstand
- Die Backendimplementierung wurde gemacht und die Dokumentation um diese erweitert

Was läuft gut?
- -

Was läuft nicht gut?
- Ich wurde schwer krank und war die ganze Woche nicht arbeitsfähig. Ich bin mich seit Samstag (gestern) am erholen, aber eigentlich noch immer nicht voll gesund

Geplante nächste Schritte / getroffene Massnahmen
- Frontend aufsetzen
- Versuchen den Plan so gut aufzuholen wie nur möglich trotz Krankheit
- Tendenz kann nur besser werden, da sich Zeit genommen wurde für die kommende Woche

Projekt-Statusbericht; Yanick Arn

Projekt: DOMAIN-DRIVEN MODELLING IN THE WEB **Statusbericht: KW40**

Projektleiter	**Projektziele**	**Verteiler**
Yanick Arn	• MVP beenden	• Gregor von Flüe, Diplomlehrer

Gesamt-beurteilung	**Projektverlauf**	**Projektklima**	**Termine**	**Risiken**	**Ressourcen**
	☒ ☐ ☐	☒ ☐ ☐	☒ ☐ ☐	☒ ☐ ☐	☒ ☐ ☐
Tendenz	⇨	⇨	⇨	⇨	⇨

Aktueller Projektstand
- Der MVP wurde abgeschlossen und somit alle Muss-Kriterien
- Sowohl FE wie auch BE sind containerisiert startbar

Was läuft gut?
- Das Projekt ist an sich «abgeschlossen»
- Die Applikation läuft auf zwei Containern, welche neben Docker abhängigkeitsunabhängig sind

Was läuft nicht gut?
- Durch eine Krankheit liegen die Kann-Kriterien nicht mehr im realistischen Bereich der Umsetzung, da diese teils sehr komplex sind. Sie sind dennoch eingeplant, aber sind nicht relevant für die Bewertung wie besprochen beim ersten Gespräch.

Geplante nächste Schritte / getroffene Massnahmen
- Dokumentation wo nötig aufbessern
- Code wo nötig aufbessern
- Kann-Kriterien Umsetzung beginnen

Projekt-Statusbericht; Yanick Arn

Projekt: DOMAIN-DRIVEN MODELLING IN THE WEB **Statusbericht: KW41**

Projektleiter	**Projektziele**	**Verteiler**
Yanick Arn	• Kann Kriterien Umsetzen	• Gregor von Flüe, Diplomlehrer

Gesamt-beurteilung	**Projektverlauf**	**Projektklima**	**Termine**	**Risiken**	**Ressourcen**
	☒ ☐ ☐	☒ ☐ ☐	☒ ☐ ☐	☒ ☐ ☐	☒ ☐ ☐
Tendenz	⇨	⇨	⇨	⇨	⇨

Aktueller Projektstand
- Es wurden einige Bugfixes gemacht
- Zwei Kann-Kriterien konnten umgesetzt werden (Syntax Highlighting + Änderung Aufbau Code für das Hinzufügen weiterer Generatoren)

Was läuft gut?
- Der MVP ist seit einer Woche fertig. Alles was jetzt gemacht wird ist freiwillig und kein offizieller Teil der Bewertung.
- Der Code und die Dokumentation wurden ein wenig aufgeräumt.
- Die Arbeit ist trotz schlechter Gesundheit auf Kurs.

Was läuft nicht gut?
- Da ich gerade durch mehrere Behandlungen gehe nehme ich Medikamente, welche angefangen haben schlimme Nebenwirkungen (u.a. sehr starke Magenschmerzen und weitere) zu haben. Ich war die ganze Arbeit durch bisher 8 Tage krank geschrieben und rund 16 Tage (8 krank geschrieben + 8 nicht krank geschrieben) krank und teils arbeitsunfähig / nicht konzentrationsfähig.

Geplante nächste Schritte / getroffene Massnahmen
- Dokumentation aufbessern
- Code wo nötig aufbessern
- Abschluss beginnen

Projekt-Statusbericht; Yanick Arn

Projekt: DOMAIN-DRIVEN MODELLING IN THE WEB **Statusbericht: KW42**

Projektleiter	**Projektziele**	**Verteiler**
Yanick Arn	• Diplomarbeit beenden	• Gregor von Flüe, Diplomlehrer

Gesamt-beurteilung	**Projektverlauf**	**Projektklima**	**Termine**	**Risiken**	**Ressourcen**
	☒ ☐ ☐	☒ ☐ ☐	☒ ☐ ☐	☒ ☐ ☐	☒ ☐ ☐
Tendenz	⇨	⇨	⇨	⇨	⇨

Aktueller Projektstand
- Die Diplomarbeit wurde beendet
- Die Webpublikation wurde geschrieben

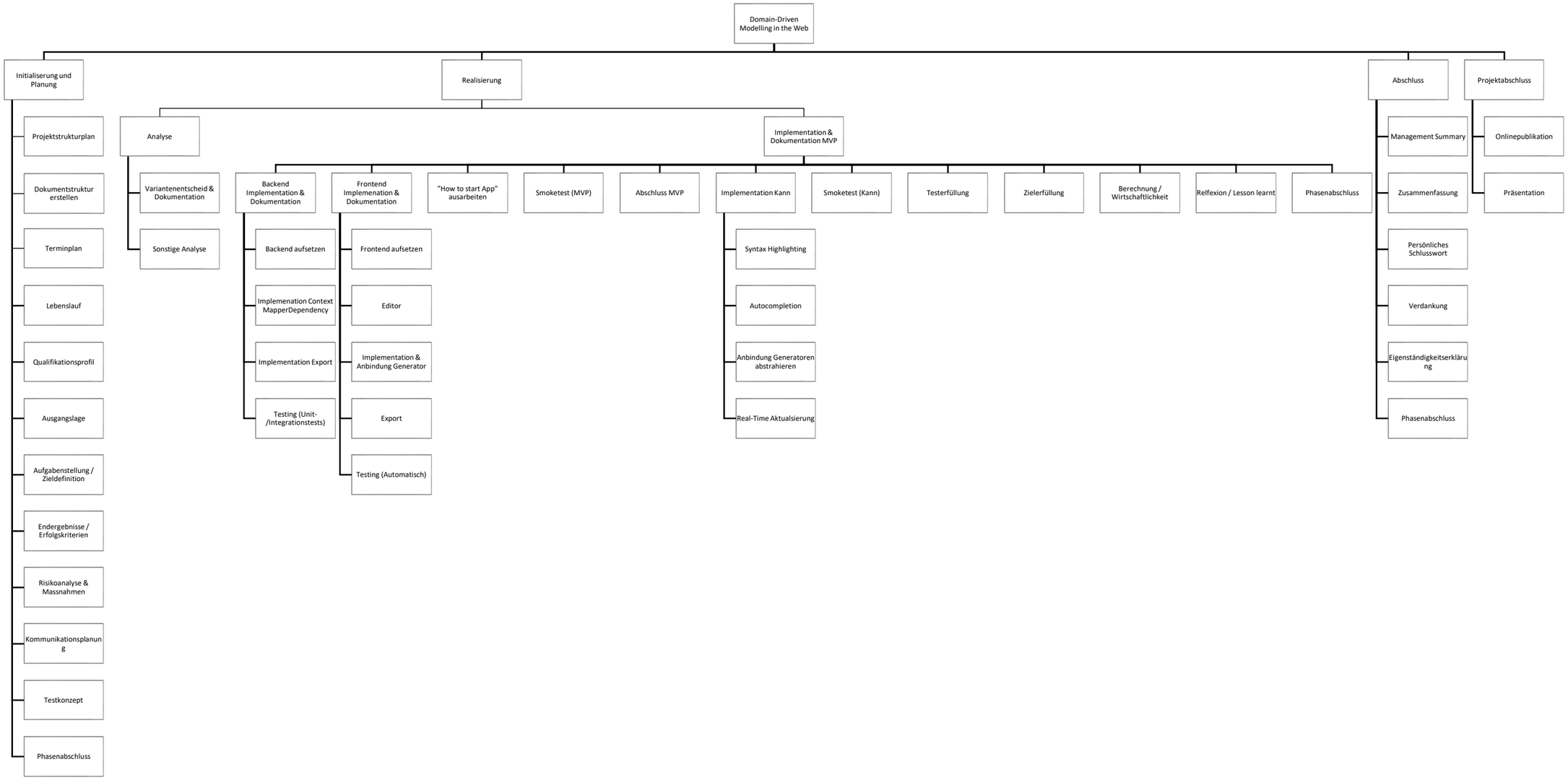
Was läuft gut?
- Die Arbeit konnte trotz teils schlechten Umständen erfolgreich abgeschlossen werden.

Was läuft nicht gut?
- -

Geplante nächste Schritte / getroffene Massnahmen
- Präsentation

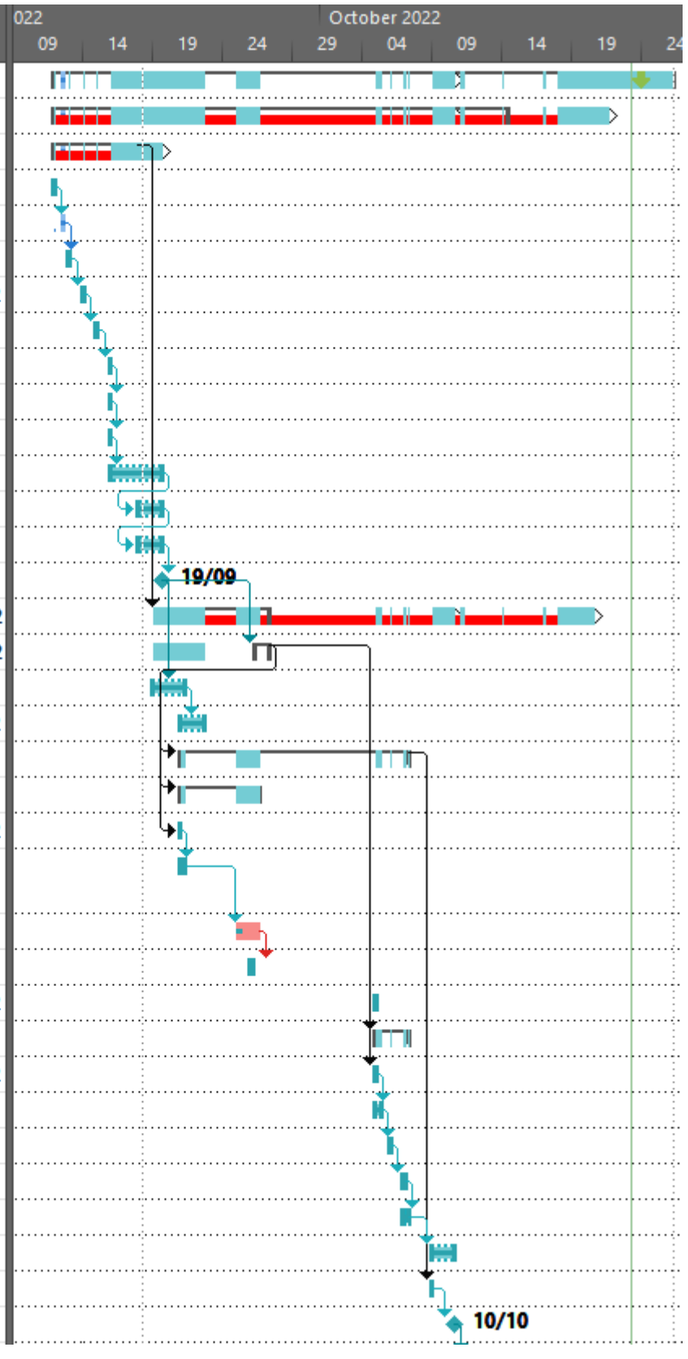
Projekt-Statusbericht; Yanick Arn

6.2 Projektstrukturplan

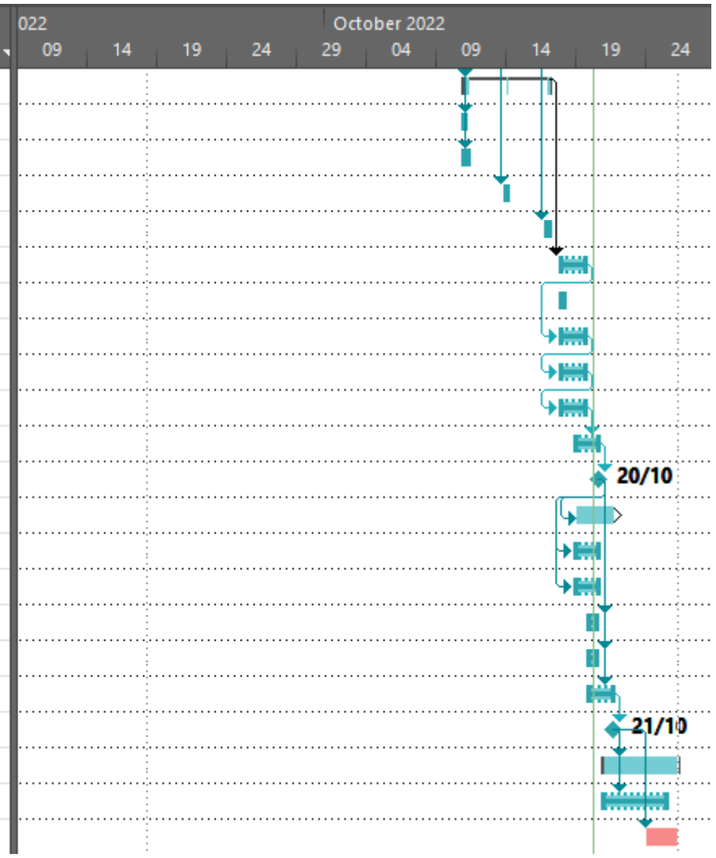


6.3 Projektablaufplan

Vorgangsname	SOLL	IST	Start	End
Diplomarbeit	**160 hrs**	**147 hrs**	**Mon 12/09/22**	**Sat 12/11/22**
Dokumentation und Implementation	**152 hrs**	**143 hrs**	**Mon 12/09/22**	**Fri 18/11/22**
Initialisierung & Planung	**26.25 hrs**	**26.25 hrs**	**Mon 12/09/22**	**Sun 18/09/22**
Projektstrukturplan	2 hrs	2 hrs	Mon 12/09/22	Mon 12/09/22
Terminplan	6 hrs	6 hrs	Mon 12/09/22	Mon 12/09/22
Dokumentstruktur erstellen	2 hrs	2 hrs	Tue 13/09/22	Tue 13/09/22
Lebenslauf	3 hrs	3 hrs	Wed 14/09/22	Wed 14/09/22
Qualifikationsprofil	3 hrs	3 hrs	Thu 15/09/22	Thu 15/09/22
Ausgangslage	0.25 hrs	0.25 hrs	Fri 16/09/22	Fri 16/09/22
Aufgabenstellung / Zieldefinition	0.25 hrs	0.25 hrs	Fri 16/09/22	Fri 16/09/22
Endergebnisse / Erfolgskriterien	0.25 hrs	0.25 hrs	Fri 16/09/22	Fri 16/09/22
Risikoanalyse & Massnahmen	4 hrs	4 hrs	Fri 16/09/22	Fri 16/09/22
Kommunikationsplanung	3 hrs	3 hrs	Sun 18/09/22	Sun 18/09/22
Testkonzept	2 hrs	2 hrs	Sun 18/09/22	Sun 18/09/22
Phasenabschluss	0.5 hrs	0.5 hrs	Sun 18/09/22	Sun 18/09/22
Realisierung	**121.5 hrs**	**112.5 hrs**	**Mon 19/09/22**	**Wed 19/10/22**
Analyse	**12 hrs**	**12 hrs**	**Mon 19/09/22**	**Wed 21/09/22**
Variantenentscheide & Dokumentation	8 hrs	8 hrs	Mon 19/09/22	Tue 20/09/22
Sonstige Analysen Software	4 hrs	4 hrs	Wed 21/09/22	Wed 21/09/22
Implementation & Dokumentation (MVP)	**63.5 hrs**	**58.5 hrs**	**Wed 21/09/22**	**Sun 09/10/22**
Backend Implementation & Dokumenation	**30 hrs**	**28 hrs**	**Wed 21/09/22**	**Tue 04/10/22**
Backend aufsetzen	2 hrs	2 hrs	Wed 21/09/22	Wed 21/09/22
Implementation Context Mapper Dependency (BE)	15 hrs	15 hrs	Wed 21/09/22	Sun 25/09/22
Implementation Export (BE)	3 hrs	2 hrs	Sun 25/09/22	Sun 25/09/22
Testing (Unit-/Integrationstests) (BE)	10 hrs	9 hrs	Mon 26/09/22	Tue 04/10/22
Vorbereitung & Gespräch mit Experten (1)	1.5 hrs	1.5 hrs	Wed 05/10/22	Wed 05/10/22
Frontend Implementation & Dokumenation	**32 hrs**	**29 hrs**	**Wed 05/10/22**	**Sun 09/10/22**
Frontend aufsetzen	8 hrs	8 hrs	Wed 05/10/22	Wed 05/10/22
Editor (FE)	8 hrs	8 hrs	Wed 05/10/22	Thu 06/10/22
Implementation & Anbindung Generator (FE)	6 hrs	5 hrs	Thu 06/10/22	Fri 07/10/22
Export (FE)	4 hrs	4 hrs	Fri 07/10/22	Fri 07/10/22
Testing (Automatisch) (FE)	6 hrs	4 hrs	Fri 07/10/22	Sun 09/10/22
"How to start App" ausarbeiten	4 hrs	4 hrs	Sun 09/10/22	Sun 09/10/22
Smoketest (MVP)	1 hr	1 hr	Sun 09/10/22	Sun 09/10/22
Abschluss MVP	0.5 hrs	0.5 hrs	Sun 09/10/22	Sun 09/10/22



Vorgangsname	SOLL	IST	Start	End
Implementation Kann-Kriterien	**34 hrs**	**30 hrs**	**Mon 10/10/22**	**Tue 18/10/22**
Syntax Highlighting	6 hrs	6 hrs	Mon 10/10/22	Tue 11/10/22
Autocompletion	20 hrs	17 hrs	Tue 11/10/22	Fri 14/10/22
Anbindung Generatoren abstrahieren	4 hrs	3 hrs	Fri 14/10/22	Mon 17/10/22
Real-Time Aktualisierung	4 hrs	4 hrs	Mon 17/10/22	Tue 18/10/22
Smoketest (Kann)	1 hr	1 hr	Tue 18/10/22	Tue 18/10/22
Vorbereitung & Gespräch mit Experten (2)	1 hr	1 hr	Tue 18/10/22	Tue 18/10/22
Evaluation der Testerfüllung	1 hr	1 hr	Tue 18/10/22	Tue 18/10/22
Evaluation der Zielerreichung	1 hr	1 hr	Tue 18/10/22	Tue 18/10/22
Berechnung / Wirtschaftlichkeit	1 hr	1 hr	Tue 18/10/22	Tue 18/10/22
Reflexion / Lesson learnt	1 hr	1 hr	Wed 19/10/22	Wed 19/10/22
Phasenabschluss	0.5 hrs	0.5 hrs	Wed 19/10/22	Wed 19/10/22
Abschluss	**4.25 hrs**	**4.25 hrs**	**Wed 19/10/22**	**Thu 20/10/22**
Management Summary	1 hr	1 hr	Wed 19/10/22	Wed 19/10/22
Zusammenfassung	1.5 hrs	1.5 hrs	Wed 19/10/22	Wed 19/10/22
Persönliches Schlusswort	0.5 hrs	0.5 hrs	Thu 20/10/22	Thu 20/10/22
Verdankung	0.5 hrs	0.5 hrs	Thu 20/10/22	Thu 20/10/22
Redlichkeitserklärung	0.25 hrs	0.25 hrs	Thu 20/10/22	Thu 20/10/22
Phasenabschluss	0.5 hrs	0.5 hrs	Thu 20/10/22	Thu 20/10/22
Projektabschluss	**8 hrs**	**4 hrs**	**Fri 21/10/22**	**Sat 12/11/22**
Onlinepublikation	4 hrs	4 hrs	Fri 21/10/22	Sun 23/10/22
Vorbereitung & Präsentation	4 hrs	0 hrs	Mon 24/10/22	Sat 12/11/22



6.4 Allgemeine Risikoanalyse

Identifikation & Klassifizierung	Eintrittsindikator	Auswirkung	Risiko				Massnahmen
			W	S	Risiko	Handlungsweise	
1. Krankheit/Unfall Menschlich / Höhere Gewalt	• Der Diplomand wird krank oder hat einen Unfall	• Zeitverlust	W2	S3	mittel	Risikominderung	• Frühzeitig bei Experten melden, wenn ein solches Problem Auftritt, um die Diplomarbeitsdauer zu verlängern • Arztzeugnis einholen • Gut planen
2. Verpassen des Abgabetermins Menschlich	• Der Diplomand gibt das Projekt nicht termingerecht ab.	• Notenabzug	W1	S4	mittel	Risikominderung	• Einhalten des Projektplans. • Diplomarbeit frühzeitig fertigstellen. • Erinnerung zum Abschluss erstellen
3. Datenverlust bei Dokumentation / Code Technisch	• Daten sind nicht mehr aufrufbar/auffindbar	• Verzögerungen im Zeitplan • Abgabe eines nicht vollständigen Produkts	W1	S4	mittel	Risikominderung	• Dokumentieren, dass es Probleme gab • Dokumentation auf der Cloud speichern und automatische Speicherung aktivieren • Code auf Github ablegen und mindestens 1x pro Tag pushen
4. Probleme bei der Entwicklung Menschlich / Technisch	• Die Entwicklung läuft nicht nach Plan und/oder die Implementation ist fehlerhaft.	• Verzögerungen im Zeitplan • Abgabe eines nicht vollständigen Systems	W4	S3	hoch	Risikominderung	• Testkonzept schreiben und einhalten • Frühzeitig Hilfe bei Fachexperte holen • Genügend Zeit für Bugfixing einplanen • Dokumentieren wieso ein Teil fehlerhaft/unvollständig ist
5. Hardwareausfall Technisch	• Benötigte Hardware, wie zum Beispiel das Arbeitsgerät fallen aus	• Verzögerungen im Zeitplan • Abgabe eines nicht vollständigen Produkts • Datenverlust	W1	S2	gering	Risikominderung	• Ersatzgerät beschaffen • Speichern der Daten nach Punkt 3
6. Fehler in der Dokumentation Menschlich	• Die Arbeit weisst Fehler in der Grammatik, Semantik oder dem Design auf	• Notenabzug	W4	S3	hoch	Risikominderung	• Mehrere Reviews der Arbeit durch externe Personen • Mehrere Reviews durch Diplomanden

Legende:

Schadensausmass:

S1 = führt zu keiner Abwertung

S2 = geringe Abwertung bis 1.0 Notenpunkte

S3 = hohe Abwertung bis über 1.0 Notenpunkte

S4 = führt zu Nichtbestehen

Eintrittswahrscheinlichkeit

W1 = unvorstellbar

W2 = unwahrscheinlich

W3 = eher vorstellbar

W4 = vorstellbar

W5 = hohe Wahrscheinlichkeit

6.5 Risikoanalyse Variante «Ace»

Identifikation & Klassifizierung	Eintrittsindikator	Auswirkung	Risiko				Massnahmen
			W	S	Risiko	Handlungsweise	
1. Unterstützung durch Framework nicht gegeben Technisch	• Lässt sich nicht einbetten • Editor funktioniert nicht	• Zeitverlust • Umsetzung nicht möglich	W3	S3	mittel	Risikominderung	• Gut informieren • Sandbox Projekte aufziehen • Alternativen bereithalten
2. Unterstützung durch Xtext nicht gegeben Technisch	• Der Editor unterstützt die Xtext basierte DSL nicht	• Zeitverlust • Umsetzung nicht möglich	W2	S3	mittel	Risikominderung	• Gut informieren • Sandbox Projekte aufziehen • Alternativen bereithalten
3. Der Editor wird nicht gewartet Technisch	• Es existieren nur alte Versionen des Editors • Der Editor wurde zu der Zeit des Projektes aufgegeben	• Zeitverlust • (Zukünftige) Umsetzung nicht möglich	W2	S3	mittel	Risikominderung	• Gut informieren • Alternativen bereithalten
4. Der Editor ist nicht kompatibel mit dem Rest der App Technisch	• Der Editor ist nicht sichtbar • Der Editor macht andere Komponenten kaputt	• Zeitverlust	W2	S2	gering	Risikominderung	• Mehrere Designkonzepte bereithalten • Alternativen bereithalten
5. Dokumentationen zur Umsetzung sind nicht vorhanden Technisch	• Dokumentationen zur Umsetzung sind nicht auffindbar	• Extremer Zeitverlust	W4	S3	hoch	Risikominderung	• Gut informieren • Xtext lernen und gut verstehen • JavaScript lernen und gut verstehen • Alle nötigen Libraries lernen und gut verstehen • Konzepte lernen und gut verstehen

Legende:

Schadensausmass:

S1 = führt zu keinem Zeitverlust

S2 = geringer Zeitverlust

S3 = hoher Zeitverlust

S4 = Umsetzung nicht möglich

Eintrittswahrscheinlichkeit

W1 = unvorstellbar

W2 = unwahrscheinlich

W3 = eher vorstellbar

W4 = vorstellbar

W5 = hohe Wahrscheinlichkeit