PPR Dokumentation

24. März 2024

1 Klassenstruktur

1.1 Package core

Das Package Core enthält wichtige Klassen für die Funktionalität unserer Anwendung. Zwei der Klassen dienten zum einlesen der Abgeordneten und ihrer Fotos.

Die wichtigste Klasse dieses Packages ist die Klasse Main. Durch des Starts dieser Klasse wird die Verbindung zur Datenbank hergestellt für die Webseite und unsere Klasse gestartet, welche die Webrouten erzeugt. Die Webseite kann daraufhin unter localhost:8002 gestartet werden.

1.2 Package data

Das Package Data enthält alle wichtigen Klassen und Interfaces, die wir für die Verwaltung unserer Anwendung benötigen. Für Informationen über die Abgeordneten, Reden, Sitzungen, Tagesordnungspunkte und die NLP-Daten, wurden zuerst Interfaces erstellt und diese im Pacakge impl implementiert.

Die Klasse AbgeordneterImpl bildet abgeordnete ab und die Klasse AbgeordneterFotosImpl die Fotos der Abgeordneten mit ihren Metadaten.

Die Klasse RedeImpl speichert die Reden. Da die Reden in den XML-Dateien in verschiedene Abschnitte unterteilt sind, werden diese Abschnitte auch seperat in der Rede als RedeElementImpl gespeichert. RedeElementNLPImpl und RedeElementNLPNamedEntityImpl verwalten hierbei die Ergebnisse der NLP-Analyse. RedeCasImpl und RedeElementCasImpl bilden die Cas-Objekte ab, die bei der NLP-Analyse entstehen.

Das außerdem in diesem Package enthaltene Package usermanagement stellt Interfaces und Klassen bereit, welche für die Verwaltung von Usern und Gruppen auf unserer Webseite relevant sind.

1.3 Package database

Das Package database beinhaltet die Klassen, die für die Datenbanknutzung von Relevanz sind.

Zum einen befindet sich hier der MongoDBHandler als Interface und Klasse. Diese Klasse stellt Methoden bereit, um sich mit der Datenbank zu verbinden, Elemente in die Datenbank zu schreiben und die Datenbankelemente zu updaten, sowie verschiedene Methoden um Dokumente aus der Datenbank auszulesen.

Auch zu finden ist hier die Klasse und das Interface RedeNlpDBHelper, welche als Hilfsklasse dient, um NLP Daten aus der Datenbank auszulesen.

Die anderen in diesem Package enthaltenen Klassen und Interfaces bieten Hilfsmethoden für das Schreiben der Daten in die Datenbank.

1.4 Package downloads

Die Downloader, sowohl für die Bilder aus dem Bildarchiv des Bundestags als auch für die Protokolle von der Bundestagswebseite, befinden sich im Package downloader.

Fotos der Abgeordneten werden hier runtergeladen indem nach Fotos mit dem entsprechendem Vor- und Nachnamen gesucht wird.

1.5 Package parsing

Für die Webseite müssen verschiedene Informationen aus unterschiedlichen Quellen bezogen werden. Die hierfür benötigten Klassen mit den dazugehörigen Interfaces befinden sich im Package parsing.

Die Klasse AbgeordneterParser wird verwendet, um die xml-Datei, welche die Inhalte aller Abgeordneter enthält, zu parsen, und aus diesen Inhalten die Abgeordneten zu erstellen.

Die Klassen ProtocolParser, XMLParser und XMLTools dienen als Hilfestellungen um ein Protokoll zu Parsen und entsprechend zu erstellen und einzulesen. Hierbei werden Sitzungen einzeln geparst.

1.6 Package website

Das Package website enthält die Klassen, welche für die Webseite relevant sind. Am wichtigsten ist hier der SparkHelper. Dieser definiert die einezlenen Routen unserer Webseite und deren Funktionalitäten. (Erklärung der Routen im Nutzerhandbuch)

In diesem Package befinden sich außerdem verschiedene Klassen zum User Management, Sessions und Permissions.

Das Interface ProtokolToLaTeX mit seiner dazugehörigen Implementierung sind die Klassen, welche für die Erstellung der LaTeX-Datei aus den Sitzungsprotokollen zuständig ist und diese wiederum in eine PDF-Datei umwandelt.

2 Resources Ordner

2.1 MdB-Stammdaten

Der Ordner MdB-Stammdaten enthält die Stammdaten-XML der Abgeordneten, da diese, nach Aufgabenstellung, nicht aus dem Internet geholt werden muessen.

2.2 spark

Der Ordner spark enthält alle für die Webseite relevanten Dateien.

Der in diesem Ordner enthaltene Ordner other, enthält den header unserer Webseite inklusive der Menüleiste. Da die Menüleiste und der Header immer gleich aussehen sollen, haben wir uns dazu entschieden diesen einmal zu erstellen und dann in unsere Webseite einzubinden.

Der Ordner static enthält sowohl für die Diagramme als auch andere Funtkionalitäten der Webseite JavaScript Dateien, die dann auf unserer Webseite eingebunden werden, als auch die css-Dateien, welche das Aussehen der Webseite vorgeben. Zusätzlich zu selbstgeschriebenen css-Dateien haben wir auch bootstrap verwendet (im Tutorium abgesprochen). In diesem Ordner sind auch die Bilder enthalten, welche auf unserer Webseite eingebettet sind.

Der Ordner templates enthält, sortiert nach den Kategorien der Webseiteninhalte, die ftl-Dateien, welche die html für unsere verschiedenen Routen enthalten. Diese ftl-Dateien werden im SparkHelper verwendet und eingebunden.

3 tmp Ordner

Der tmp-Ordner wird verwendet als Ablageort für die gedownloadeten Protokolle und dafür benötigten Bilder.

4 Aufgabenaufteilung

Die Aufgabenaufteilung unseres Abschlussprojekt kann dem Gantt-Diagramm wie auch unserem Git und den Anmerkungen am Code entnommen werden.

Da während der Ausführung bestimmter Aufgaben zusätzliche Anforderungen auftauchten, wurden die Klassen von denen, welche die zusätzlichen Methoden brauchten, ergänzt und bearbeitet. An fast allen Klassen waren daher mehr als eine Person beteiligt.

Der Arbeitsaufwand für das Gesamtprojekt war gleichmäßig über alle Gruppenmitglieder verteilt.

5 Zeitliche Durchführung

Unserem Gantt-Diagramm kann die geplante zeitliche Aufteilung entommen werden. Für das spätere Tracken der Aufgaben und Deadlines, sowie das Ab-

haken erledigter Aufgaben und weiteres Zeitmanagement, haben wir die Tickets (Issues) in Git verwendet. Dort kann sowohl die geplante Deadline, der Zeitpunkt des Schließens des Tickets als auch die Beauftragten der Aufgabe eingesehen werden.

Während der Ausführung des Projekts haben wir weitestgehend an die im Gantt-Diagramm festgelegten Deadlines und Reihenfolgen gehalten. Gestartet hat unser Projekt pünktlich mit der Erstellung des Gantt-Diagramms sowie weiterer vorbereitender Dokumentation (Mock-Ups, UML-Diagramme) und einer Einplanung der verschiedenen Aufgaben. Die Zuteilung der Aufgaben haben wir auch mithilfe des Gantt-Diagramms gemacht, indem jeder die für sich möglichen Aufgaben auswählen konnte und wir dann eine faire Zuteilung der Aufgaben erledigten.

Das Programmieren haben wir dann mit der Erstellung des Backends, also dem erstellen der Klassenstruktur, der Verbindung der Datenbank, dem parsen der Abgeordneten und Protokolle, sowie dem Einlesen in die Datenbank, begonnen.

Während des auslesens der Bilder und Protokolle aus dem Internet stießen wir auf gewisse Schwierigkeiten, weswegen sich der Abschluss dieser Aufgaben leicht nach hinten verschob. Wir haben in der Zwischenzeit allerdings schon an anderen Aufgaben weitergearbeitet.

Während wir die Klassenstruktur erzeugt haben, haben wir bereits die Routen für unsere Webseite erstellt und aufgesetzt. Die lief im geplanten Zeitraum. Die Dokumentation der Api mit swagger.io haben wir nicht mehr geschafft. Daher ist dieses Ticket noch offen.

Das Erstellen der Diagramme für die Webseite geschah, sobald die ersten Daten in der Datenbank standen. Durch die anfänglichen Schwierigkeiten verzögterte sich dies leicht nach hinten. Da die Diagramme allerdings keine Abhängigkeiten zu anderen Inhalten der Webseite haben, war dies unproblematisch.

Die anderen htlm-Darstellungen, wie Reden, Abgeordnete, Filterfunktionen und Suchfunktionen, geschah weitestgehend im vorgegeben Zeitramen. Krankheitsbedingt verzögerten sich letzte Detailarbeiten wie das Einbinden der named Entities und des Fotos des Abgeordneten.

Die Erstellung der Benutzerverwaltung geschah im zeitlichen Rahmen, allerdings haben wir einige der Verwaltungsaufgaben der Webseite nicht mehr geschafft zu bearbeiten.

Die Umwandlung eines Protokolls in ein Latex-Dokument inklusive der entsprechenden Vorschau und des Downloaders für ein Dokument begann leicht nach dem dafür vorgesehenen Zeitabschnitt. Die Umwandlung der Protokolle in ein Latex-Dokument und wiederum eine PDF-Datei wurde dabei im angegebenen Zeitabschnitt beendet. Da wir bei der Latex-Vorschau und dem Downloader gewisse Komplikationen hatten, haben wir diesbezüglich nur den html-Rahmen rechtzeitig fertig gehabt. Den vollendeten Downloader mit Vorschau beendeten wir erst kurz nach der vorgegeben frist. Da hier allerdings wiederum keine Abhängigkeiten bestanden, war das unproblematisch.

Die Rot markierten Bonusaufgaben haben wir nicht bearbeitet, weswegen für diese weder eine Person eingeteilt noch ein Zeitfenster festgelegt ist.

Durch die leichten Verzögerungen in unserer Programmiertätigkeit verschoben sich auch die Beendigung der letzten Aufgaben, des Benutzerhandbuchs, des Promovideos und der Dokumentation, sowie die Überarbeitung der zu Beginn erstellten UML-Diagramme.

Die im Gantt-Diagramm grün markierten Tage haben wir als Puffer eingeplant, falls wir während unseres Projekts auf unerwartete Schwierigkeiten stoßen. Die Planung dieser Puffertage hat sich gelohnt, da wir in diesen Tagen die Dokumentation beenden, letzte Details programmieren und Kommentieren und unser Promovideo erstellen konnten.

Unsere Meilensteine haben wir durch die aufgetretenen Probleme nur mit leichter Verzögerung erreichen können. Da wir eineige Aufgaben nicht erledigt haben, bleiben manche Meilensteine somit offiziell unerfüllt, da das Projekt allerdings trotzdem zu unserer Zufriedenheit ist, haben wir von unserer Sicht unsere Meilensteine erreichen können.