HAW HAMBURG

BACHELORARBEIT

Erstellung eines webbasierten Tools zur Generierung individueller Datenreports

vorgelegt am 08.Juli 2024

Henrike Bohnet

Erstprüfer: Prof. Dr. Nils Martini
Zweitprüfer: Christopher Gudat

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Department Medientechnik
Finkenau 35
20081 Hamburg

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorstehende Bachelorthesis mit dem Titel

„Erstellung eines webbasierten Tools zur Generierung individueller Datenreports“

selbstständig ohne fremde Hilfe gefertigt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quelle kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

_____

Ort, Datum Unterschrift

Inhalt

Bildverzeichnis

1 Einführung

1.1 Motivation

Wir in unserem Unternehmen bieten Software für verschiedene Banken an. Um Aufgaben, Fehler und Probleme aus dem Test und dem laufenden Betrieb zu erfassen und zuzuordnen, benutzen wir die das Softwaretool Jira von Atlassian. Hier werden die Aufgaben in Tickets erfasst und Projekten zugeordnet.

Für ein Projekt soll monatlich eine Übersicht erstellt werden, wie viele offene Tickets vorhanden sind und wie sich diese aufschlüsseln. Dafür kann man bei Jira Filtereinstellungen nutzen, sodass einem nur die gewünschten Tickets angezeigt werden. Für die angeforderte Übersicht bedarf es allerdings mehrerer Filtereinstellungen, sodass ein externes Dokument erstellt wurde mit allen benötigten Tabellen. Diese werden dann mit Copy und Paste aus den jeweiligen Filtereinstellungen gefüllt. Diese aufwendige und fehleranfällige Arbeit soll erleichtert werden. In Zukunft soll eine Software alle benötigten Informationen automatisch zusammenführen und Reports erstellen.

Dabei ist erstmal wichtig, dass alle Parameter der Tabellen, mit denen der zu erstellenden Übersicht übereinstimmen. Im weiteren Verlauf soll es jedoch möglich sein mit dem Programm auch weitere individuelle Reports zu erstellen aus anderen Projekten und auch mit anderen Konfigurationen der Tabellen.

1.2 Zielsetzung

Im Rahmen dieser Bachelorarbeit soll ein Programm angefertigt werden, das eine Ticketstatistik erstellt und geeignet visualisiert. Diese Tickets wurden erstellt mit Hilfe der Software „Jira“ des Softwareherstellers Atlassian. Über eine REST-API wird auf diese Daten zugegriffen. Anschließend können sie mit einer webbasierten Anwendung angezeigt werden.

Des Weiteren sollen der Webanwendung Filtermöglichkeiten hinzugefügt werden, damit Daten nicht nur statisch abgerufen werden können, sondern individuell einstellbar sind. Dabei ist darauf zu achten, welche Filter Sinn ergeben und wie sie dargestellt werden können. Am Ende soll der Benutzer die Möglichkeit haben, gewisse auswählbare Daten individuell in verschiedenen Tabellen und Grafiken darzustellen.

1.3 Struktur

Zunächst gehe ich auf vergleichbare Programme ein und werde eine Abgrenzung zu meinem Programm herstellen. Als nächsten werde ich in den Grundlagen erläutern, die für das Verständnis der Arbeit notwendig sind. Dazu gehört unser Ticket-System Jira, die dazugehörige Rest-API und JQL, welche eine eigene Sprache für Jira ist, um gezielt nach Informationen zu filtern. Außerdem gehe ich auf node.js und express.js ein, die ich für meine Webanwendung benötigt habe. Im nächsten Kapitel werde ich beschreiben, welche Anforderungen das Programm erfüllen soll, warum ich mich für diese Art der Umsetzung entschieden habe und wie die Architektur des ersten Programmes aussieht. Anschließend folgt die Beschreibung der Umsetzung des Programmes. Ich werde noch weiter auf die Filtermöglichkeiten und Konfiguration eingehen mit besonderem Hinblick auf die Erweiterung des ersten Programmes. Als letztes folgt ein Fazit und der Ausblick.

2 Vergleichbare Programme und Abgrenzung

2.1 Power BI

Hervorgegangen aus dem Projekt Crescent, entwickelt 2011 von Ron George, veröffentlichte Microsoft die Software unter dem Name Power BI im Jahr 2013 (Luber, 2018). Stetig weiterentwickelt ist es nun ein Tool, das es Benutzern ermöglicht, ihre Daten zu analysieren und visualisieren. Es bietet Werkzeuge für die Erstellung interaktiver Berichte und Dashboards. Power BI unterstützt eine Vielzahl von Datenquellen, einschließlich Datenbanken (SQL Server, Oracle, etc.), Cloud-Dienste (Azure, Salesforce, etc.), Excel-Dateien und Web-APIs unteranderem auch zu JIRA. Benutzer können Daten aus verschiedenen Quellen importieren, transformieren und kombinieren.
Um nun Tabellen individuell zu erstellen, müssen die Daten importiert, die Datensätze bereinigt und anschließend die Spalten und Zeilen definiert und angeordnet werden. Für jede Tabelle muss dieser Prozess wiederholt werden. Dabei muss der Ersteller der Tabellen sich in das Programm einarbeiten und die ganzen Funktionalitäten kennen (Microsoft, 2024).

2.2 Tableau

Tableau ist ein leistungsfähiges und vielseitiges Werkzeug zur Datenvisualisierung. Entwickelt von Tableau Software, einem Salesforce-Unternehmen, ermöglicht es Nutzern, Daten aus verschiedenen Quellen zu kombinieren, zu analysieren, zu visualisieren und interaktiv darzustellen.

Ein herausragendes Merkmal von Tableau ist seine Fähigkeit, aus einer Vielzahl von Quellen Daten zu integrieren, darunter relationale Datenbanken, Cloud-basierte Systeme, Excel-Tabellen und sogar Webdaten. Durch Dashboards wird es ermöglicht, Daten in Echtzeit zu erkunden und zu analysieren.

Tableau unterstützt auch die Zusammenarbeit und den Austausch von Erkenntnissen. Mit Tableau Cloud können Dashboards und Berichte einfach

veröffentlicht und mit anderen geteilt werden. Nutzer können Berechtigungen verwalten, um sicherzustellen, dass nur autorisierte Personen Zugriff auf sensible Daten haben. Dies fördert eine kollaborative Umgebung, in der Teams gemeinsam an Datenanalysen arbeiten und fundierte Entscheidungen treffen können (Salesforce, 2024).

2.3 Meine Software

Es gibt noch einige weitere Datenanalyse-Tools auf dem Markt. Für viele werden allerdings Lizenzen für jeden Benutzer benötigt, der Berichte und Dashboards erstellen möchte. Außerdem richten sie sich bei der Erstellung von Dashboards vor allem an Datenanalysten oder Leute mit vergleichbaren Aufgaben. Dadurch kann die Erstellung für Benutzer mit wenig Erfahrung in diesem Bereich erschwert werden, da es eine längere Einarbeitungszeit bedarf. Durch die Komplexität und Menge an Möglichkeiten, die diese Tools bieten,

Mein Tool soll jedem Benutzer uneingeschränkt erlauben Reports zu erheben. Es kann individuell an unser Unternehmen und seine Bedürfnisse angepasst werden. Dadurch soll es übersichtlich bleiben und einfach zu ändern.

3 Grundlagen

3.1 Jira / Atlassian (Ticketsystem)

Jira ist ein Software-Tool, entwickelt von Atlassian. Laut eigener Beschreibung dient Jira als Projektmanagementtool zur zuverlässigen Planung, Nachverfolgung und Unterstützung eigener Software (Atlassian, 2024). Der wichtigste Bestandteil von Jira ist sein Issue-Tracking-System. Ein "Issue", oder auch Ticket, kann eine Aufgabe, ein Fehler, eine Funktion oder jede andere Arbeitseinheit sein. Jira ermöglicht die Erstellung, Zuweisung, Priorisierung und Nachverfolgung von Tickets. Benutzer können Tickets kommentieren, Dateien anhängen, Arbeitsprotokolle führen und den Fortschritt durch verschiedene Workflow-Status verfolgen. Administratoren können benutzerdefinierte Workflows, Felder, Bildschirmmasken und Berechtigungsschemata erstellen, um die spezifischen Anforderungen ihres Teams zu erfüllen. Es ist möglich sogenannte Dashboards zu erstellen, mit den es möglich ist einen visuellen Überblick über das gesamte Projekt zu erhalten. Darüber hinaus bietet Atlassian den Atlassian Marketplace, auf dem eine Vielzahl von Plugins und Erweiterungen verfügbar sind, um Jira-Funktionalitäten weiter auszubauen. Jira ermöglicht eine zentralisierte Verwaltung aller Projektaktivitäten. Dies fördert die Transparenz und erleichtert die Zusammenarbeit zwischen Teammitgliedern. Jira bietet umfangreiche Berichterstattungsfunktionen, die es Teams ermöglichen, den Fortschritt, die Leistung und Engpässe in Echtzeit zu überwachen. Berichte wie Burndown-Charts, Velocity-Charts und Cumulative Flow-Diagramme bieten wertvolle Einblicke in den Projektstatus und helfen bei der Entscheidungsfindung.

3.2 Jira REST-API

Die Jira REST API ist ein leistungsfähiges Werkzeug, das Entwicklern ermöglicht, mit Jira zu interagieren. Wie der Name schon sagt, folgt sie der REST-Architektur. Das heißt, sie kommuniziert über http-Anfragen, um Standard-Datenbankfunktionen wie das Erstellen (POST-Methode), Lesen (GET-Methode), Aktualisieren (PUT-Methode) und Löschen (DELETE-Methode) von Datensätzen (auch als „CRUD“ für Creating, Reading, Updating und Deleting bekannt) innerhalb einer Ressource durchzuführen (IONOS, 2020). Die Antwort erfolgt im JSON-Format

Die Jira REST API stellt verschiedene Endpunkte zur Verfügung, die den Zugriff auf unterschiedliche Jira-Ressourcen ermöglichen. Zu den Hauptressourcen gehören:

- **Issues**: Repräsentiert Aufgaben, Fehler oder andere Arbeitseinheiten.
- **Projects**: Verwaltung und Konfiguration von Projekten.
- **Users**: Benutzerverwaltung und Berechtigungen.
- **Boards**: Verwaltung von Scrum- und Kanban-Boards.
- **Sprints**: Verwaltung von Sprints in Scrum-Projekten.
- **Worklogs**: Protokollierung und Nachverfolgung von Arbeitszeiten.

Um auf die Jira REST API zuzugreifen, müssen Benutzer authentifiziert werden. Die gängigsten Methoden zur Authentifizierung sind Basic Authentication und OAuth. Basic Authentication verwendet Benutzername und API-Token, während OAuth ein sichereres, tokenbasiertes Authentifizierungsverfahren bietet. (IBM, 2024)

Beispiel: Abrufen eines Issues

GET /rest/api/3/issue/ISSUE-KEY

Die Antwort ist typischerweise ein JSON-Dokument, das die angeforderten Daten enthält.

Beispielantwort:

```
{
   "id": "10000",
   "key": "ISSUE-KEY",
   "fields": {
      "summary": "Issue Summary",
      "description": "Issue Description",
      ...
   }
}
```

3.3 JQL

Jira Query Language (JQL) ist eine leistungsstarke und flexible Abfragesprache, die speziell für die Suche und Filterung von Daten in Jira entwickelt wurde. Sie ermöglicht es Benutzern, komplexe Suchanfragen zu erstellen, um spezifische Informationen innerhalb eines Jira-Projekts zu finden.

JQL basiert auf einer Syntax, die es erlaubt, Bedingungen und Kriterien für die Suche zu definieren. Diese Bedingungen können sowohl einfache als auch komplexe Filter beinhalten, wie beispielsweise die Suche nach bestimmten Feldern, Werten oder Attributen von Jira-Issues. Ein grundlegendes JQL-Statement könnte beispielsweise so aussehen: `project = "ABC" AND status = "Open"`, was alle offenen Issues im Projekt „ABC“ zurückgeben würde. Benutzer können eine Vielzahl von Schlüsselwörtern und Operatoren verwenden, um ihre Suchanfragen genau zu definieren. Dazu gehören Standardfelder wie Projekt, Status, Priorität, Reporter und Bearbeiter, sowie benutzerdefinierte Felder, die spezifisch für ein bestimmtes Jira-Setup konfiguriert wurden. Darüber hinaus unterstützt JQL auch die Nutzung von logischen Operatoren wie AND, OR und NOT, die es ermöglichen, mehrere Bedingungen zu kombinieren und komplexe Abfragen zu erstellen. (Atlassian, 2024)

3.4 Node.js

Node.js ist eine plattformübergreifende, Open-Source-JavaScript-Laufzeitumgebung, die auf der V8-Engine von Google Chrome basiert. (Flaviocopes et al., 2024). Es wurde 2009 von Ryan Dahl (TrainingDotCom, 2016) entwickelt und hat seitdem große Popularität in der Webentwicklung gewonnen. Node.js ermöglicht es, serverseitige Anwendungen in JavaScript zu schreiben, was eine einheitliche Sprache für sowohl die Client- als auch die Serverseite einer Anwendung bietet.

Das Herzstück von Node.js ist sein asynchrones, nicht-blockierendes I/O-Modell. Dieses Modell ermöglicht es, I/O-Operationen (wie das Lesen und Schreiben von Dateien oder den Zugriff auf eine Datenbank) auszuführen, ohne den Hauptthread zu blockieren. Dadurch können Node.js-Anwendungen tausende von gleichzeitigen Verbindungen effizient verwalten. Dieses bietet jedoch auch eine große Herausforderung für Entwickler, die das asynchrone Programmieren nicht gewohnt sind. Die Verwaltung von Callbacks und die Verwendung von Promises oder async/await erfordert ein Umdenken.

Node.js nutzt die V8-Engine von Google Chrome, die JavaScript in Maschinencode kompiliert und eine sehr hohe Ausführungsgeschwindigkeit bietet. Die V8-Engine ist bekannt für ihre Leistungsfähigkeit und Optimierung, was zu schnellen und effizienten Anwendungen führt.

3.5 Express

Express, häufig als Express.js bezeichnet, ist ein minimalistisches und flexibles Webanwendungs-Framework für Node.js. Es bietet eine robuste Reihe von Funktionen, um Einzel- und Mehrseiten-Webanwendungen sowie verschiedene Webdienste zu erstellen. In der wissenschaftlichen und industriellen Praxis hat sich Express als Standardframework für Node.js-Anwendungen etabliert, dank seiner Effizienz, Einfachheit und Erweiterbarkeit.

Express fungiert als eine Middleware-Schicht, die HTTP-Anfragen und -Antworten verarbeitet. Dies wird durch den Einsatz von Routen und Middleware-Komponenten erreicht. Die Routen in Express erlauben die Definition von Endpunkten für HTTP-Methoden (wie GET, POST, PUT und DELETE), die jeweils spezifische Funktionen ausführen, wenn Anfragen an diese Endpunkte gesendet werden.

Middleware sind Funktionen, die Zugriff auf das Anforderungsobjekt (req), das Antwortobjekt (res) und die nächste Middleware-Funktion im Anforderungs-Antwort-Zyklus haben. Diese Middleware-Funktionen können beliebig komplex sein und verschiedene Aufgaben übernehmen, wie das Parsen von JSON-Objekten, das Verarbeiten von Dateien oder das Handhaben von Fehlern.

Ein weiterer bedeutender Vorteil von Express ist seine hohe Leistungsfähigkeit. Durch den Verzicht auf unnötige Abstraktionen und die direkte Nutzung der nicht-blockierenden I/O-Fähigkeiten von Node.js, kann Express hochperformante Webanwendungen unterstützen.

Darüber hinaus zeichnet sich Express durch seine starke Community und umfangreiche Dokumentation aus (StrongLoop, Inc. , 2024).

4 Anforderungen und Entwurf

4.1 Anforderungsanalyse

1. Die Benutzer entscheiden, welche Jira-Version sie benutzen wollen, da zwei verschiedene im Unternehmen in Verwendung sind.
2. Aus einer Liste aller möglichen Projekten, wird das gewünschte ausgewählt.
3. Die Benutzer wählen aus, nach welchen Kriterien eine Tabelle erstellt werden soll.
    a. Welche Tickets sollen berücksichtigt werden
    b. Wonach sollen die Tickets aufgeschlüsselt werden, welche Spalten und Zeilen soll die Tabelle beinhalten
4. Es soll möglich sein, weitere Tabellen hinzuzufügen mit anderen Parametern
5. Am Ende soll eine PDF-Datei erstellt werden mit allen generierten Tabellen

4.2 Design

Ich habe mich dazu entschieden die JIRA-Api zu nutzen. Die http-Anfragen kann ich dynamisch generieren und so gezielt auf gewünschte Informationen zugreifen. Die Anwendung soll im Browser laufen, weshalb für die Oberflächenprogrammierung in HTML, CSS und JavaScript erfolgen kann. Mit Node.js und Express.js habe ich eine Möglichkeit gefunden auch für den Server eine JavaScript-Anwendung zu entwickeln. Dadurch, dass ich auch für die Oberfläche JavaScript benutze, muss die Datenverarbeitung nicht zwangsläufig auf der Serverseite erfolgen. Es kann beispielsweise eine Liste an Tickets im JSON-Format an die Website übergeben werden. Nun kann browserseitig alle Tickets gefiltert werden ohne das erneute Anfragen über das Backend laufen müssen.

Möglich wäre es auch gewesen eine Anwendung in einer anderen Programmiersprache, wie Java oder Python, zu entwickeln. Wichtig ist, dass http-Request geschickt und verarbeitet werden können, um mit der Jira-Schnittstelle zu kommunizieren. Ich habe mich allerdings für die Webbasierte Lösung entschieden. Zum einen bietet JavaScript eine gute Möglichkeit JSON-Objekte zu verarbeiten. Auch ist man mit HTML und CSS sehr flexibel, was Design-Möglichkeiten angeht. Außerdem bin ich mit diesen Sprachen bereits vertraut und habe schon mit ihnen gearbeitet.

4.3 Architektur

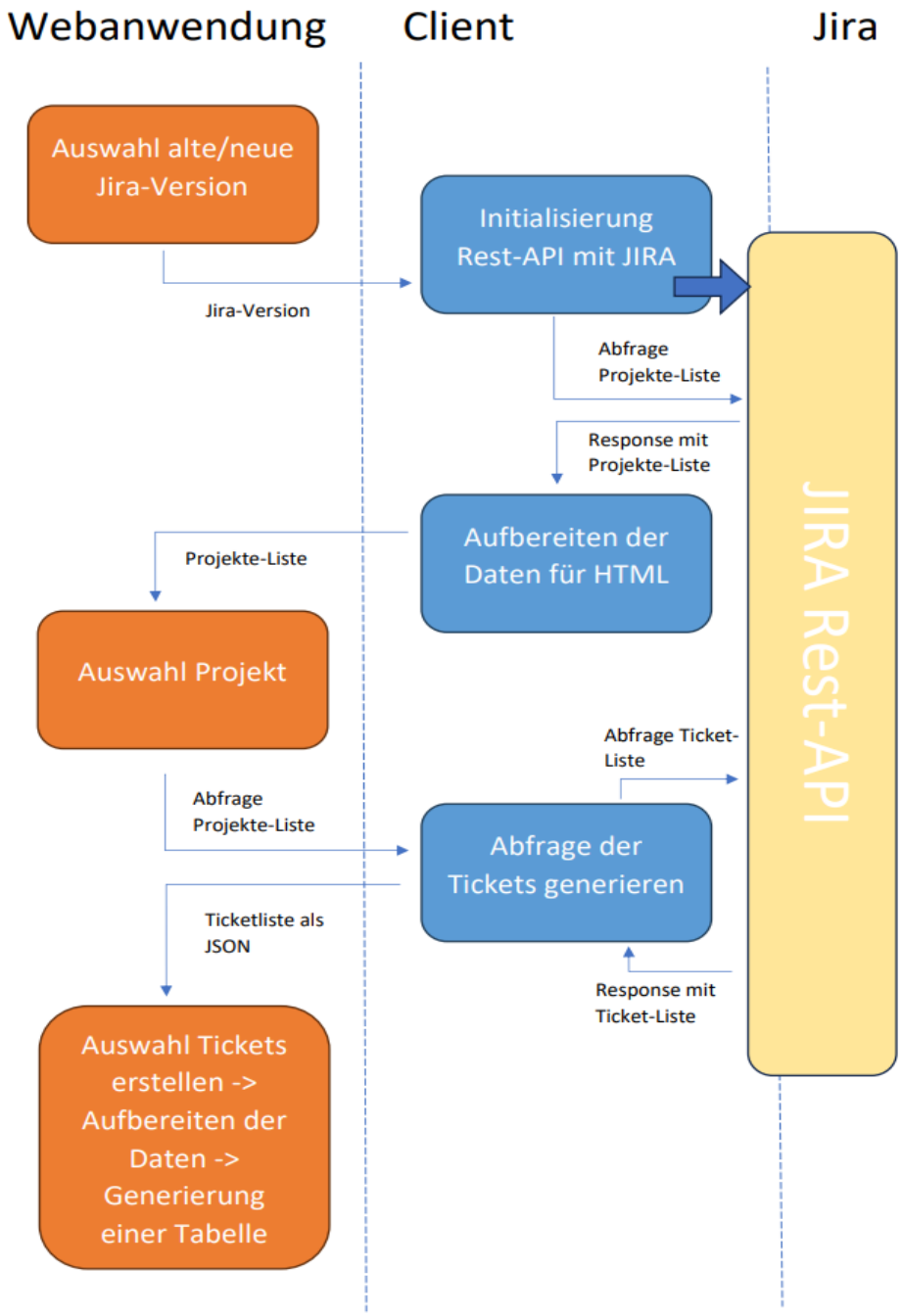

Abbildung 1 – Architektur des ersten Programmes (eigene Abbildung)

Zu sehen ist hier eine Skizzierung des ersten Programmes. Auf der linken Seite befindet sich die Komponenten auf der Browserseite. In der Mitte ist die Node.js-Anwendung und auf der rechten Seite ist die Jira-Plattform dargestellt. Kommuniziert wird mit über die JIRA Rest-API.

5 Umsetzung

5.1 Erste Funktion

Das Programm besteht aus zwei verschiedenen Komponenten. Zum einen im Backend, ein JS-Datei. Diese implementiert eine Node.js-Anwendung, um eine Webanwendung zu erstellen und mit der Jira-API zu interagieren. Dazu habe ich folgende Bibliotheken genutzt:

- ‚jira-client': Zur Kommunikation mit der Jira REST API (Bernardo, 2022)
- ‚http' zum Erstellen eines http-Server
- ‚url' zur Analyse von URL-Adressen
- ‚fs' um mit dem Datei-System zu arbeiten
- ‚express' zur Erstellung einer Webanwendung
- ‚path' um mit Dateipfaden zu arbeiten

Gestartet wird das Programm über die Konsole mit dem Aufruf „node programName.js".

Als erstes wird mit „Express" ein Server erstellt

```
var app = express();
var server = http.createServer(app);
```

Abbildung 2 – Beispielcode zum Erstellen eines Servers (eigene Abbildung)

Nachdem der Server wie folgt gestartet wurde, kann man über die URL „localhost:8080/… auf die abgelegten htmls zugreifen.

```
app.listen(8080, function (err) {
    if (err) console.log(err);
    console.log("Server listening on PORT", 8080);
});
```

Abbildung 3 – Beispielcode zum Starten des Servers (eigene Abbildung)

Um die Benutzung einfach zu halten, wird beim Aufruf der URL http://localhost:8080/ direkt die erste HTML angezeigt.

Abbildung 4

Diese ist lediglich eine statische HTML, auf der der Benutzer entscheiden muss, welche Jira-Version er benutzen möchte, da wir zwei verschiedene zurzeit in Verwendung haben. Zum einen „Helpdesk“ zum anderen „Servicedesk“. Da beide eine unterschiedliche URL haben, muss auch dementsprechend die JIRA-Schnittstelle konfiguriert werden. Mit Drücken des „Weiter“-Buttons erfolgt ein Routing auf die nächste Website und das Weitergeben der Version als Query-String in der URL. Hier wird nun die Jira-API initialisiert. Im Anschluss erfolgt die erste Jira-Abfrage mit der Funktion .listProjects(), was dem http-Request https://*hostURL*/rest/api/2/project entspricht. Dies ist ein asynchroner Aufruf. Als Antwort wird ein JSON-Object zurückgegeben. Dieses beinhaltet unteranderem die Namen und IDs der vorhandenen Projekte. Für jedes Projekt wird ein ein HTML-Input-Element erstellt. Diese werden dann in die HTML-Datei eingefügt. Anschließend wird diese dann angezeigt. Dort kann der Benutzer eins der aufgelisteten Projekte auswählen.

Abbildung 5 – Beispiel Webseite mit dynamisch erstellter Projektliste (eigene Abbildung)

Im Anschluss erfolgt ein weiteres Routing auf die nächste Website. Es wird eine weitere Jira-Abfrage gestartet. Diesmal wird eine Suche durchgeführt mit JQL. Dabei sollen alle Tickets mit einem bestimmten Status ermittelt werden, sprich keine, die bereits abgearbeitet und geschlossen wurden. Nur diese werden in der Statistik später auftauchen. Die zurückgegebene JSON wird so an die anzuzeigende Website weitergegeben. Damit ist auf dieser Seite alles abgeschlossen. Die Datenverarbeitung findet im Browser statt. Dort wird zunächst die Wahl getroffen welche Tickets berücksichtigt werden sollen.

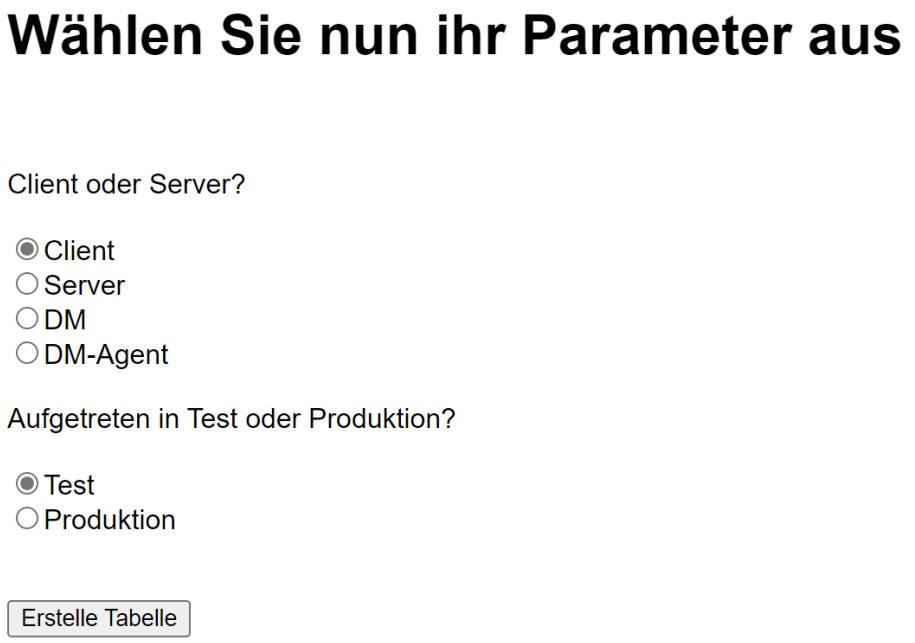

Abbildung 6 – Beispiel Auswahl auf HTML-Seite zur Generierung von Tabellen (eigene Abbildung)

Beim Klicken auf „Erstelle Tabelle“ wird anhand der der Einstellungen eine Tabelle generiert. Dazu wird zuerst gecheckt, ob alle Felder gefüllt sind, eine Tabelle in die HTML hinzugefügt, und anschließend durch die Tickets, die Priorisierungen und Kategorien iteriert, um die Tickets zu zählen, die genau diesen Kriterien

entsprechen. Kategorie und Priorisierung sind dabei festgelegte Parameter für die Tabelle.

Am Ende erscheint eine Tabelle.

eCMS-Client, Test				
Kategorie	**P1**	**P2**	**P3**	**P4**
Fehler	0	0	2	0
Kein Fehler	0	0	8	0
Doppelt / Bekannter Fehler	0	0	0	0
Änderungswunsch	0	0	8	0
Wartung	0	0	0	0
Keine Kategorie	0	0	10	1
Gesamt	0	0	28	1

Abbildung 7 – Beispiel einer generierten Tabellen betreffend Tickets aus Test für den ecms-Client (eigene Abblidung)

Man kann die Auswahl beliebig verändern und weitere Tabellen erstellen. Außerdem gibt es einen Button „PDF erstellen“. Beim Drücken erfolgt ein Download aller generierten Tabellen in einer PDF. Dazu benutze ich die Bibliothek „jsPDF“ (Kenneth Glassey et al., 2021) und „html2canvas“ (Hertzen, 2022). Es wird zunächst ein Screenshot von allen Tabellen erstellt, in einer PDF gespeichert und anschließend gedownloadet. Damit wird die Datei automatisch im Downloadverzeichnis abgelegt.

5.2 Filtereinstellungen / Konfigurierbarkeit

Die erste Funktion hat noch keine komplexen Auswahlmöglichkeiten. Sie zeigt jedoch welche Abfragen funktionieren und was für Ergebnisse sie zurückliefern. So zeigt sich, dass bei der JQL-Abfrage der Tickets eine Vielzahl an Informationen für die jeweiligen Tickets zurückgeliefert werden. Unteranderem alle Kommentare und Beschreibungen für das Ticket. Diese Informationen sind für mein Programm unwichtig und um zu verhindern, dass die Datenmengen zu groß werden, sollten diese Felder nicht an die Website weitergeleitet werden. Es gibt die Möglichkeit bei der Abfrage zu definieren, welche Felder in der Response zurückgegeben werden. Hier sollte eine sinnvolle Auswahl getroffen werden, damit genug Informationen mitkommen, ohne dass es zu viele werden. Es sollten genau die Felder sein, nach denen später auch gefiltert werden kann. Also auf jeden Fall „Priorität“, „Aufgetreten in“, „Kategorie“ und „Komponente“. Diese Abfrage kann man statisch gestalten und alle Felder im Code definieren. Alternative ist, dass man vor der Generierung der Tabellen eine weitere Website einbaut. Auf der können alle möglichen Felder aufgelistet werden, die Jira für Tickets bereitstellt. Die Abfrage erfolgt über die Funktion listFields(). Aus dieser Liste kann der Benutzer wählen, welche er später benutzen möchte. Dies könnte zur Folge haben, dass der Benutzer überfordert ist an Entscheidungen, die er treffen muss oder später doch nach einem Kriterium filtern möchte, dass er vergessen hat auszuwählen.

Außerdem ist zu beachten, dass es zwei verschiedene Arten der Filter gibt, um eine Tabelle zu erstellen. Zum einen soll ausgewählten werden, welche Tickets überhaupt berücksichtigt werden sollen, zum anderen gibt es die Auswahl, nach welchen Kriterien Spalten und Zeilen gefüllt werden sollen. Hier ist es sinnvoll eine visuelle Unterscheidung der beiden Auswahlmöglichkeiten zu machen.

6 Schluss

6.1 Fazit

Ich habe ein Programm entwickelt, dass eine kleine Auswahl an Filtermöglichkeiten bietet. Es werden Tabellen mit dieser Auswahl dynamisch generiert und es können beliebig viele hinzugefügt werden. Am Ende kann eine PDF erstellt werden. Ich habe mich viel mit Node.js auseinandergesetzt. Schwierigkeiten kamen vor allem bei asynchroner Programmierung auf, da ich diese nicht gewohnt bin. Auch war es nicht so leicht nachzuvollziehen, welche Daten bei einer Abfrage über die JIRA-API tatsächlich zurückgegeben werden. Die Dokumentation dazu war nicht immer hilfreich. Ein Grundgerüst ist geschaffen. Die Schnittstelle zu Jira ist aufgebaut, eine dynamische Liste wurde erstellt und die Tickets werden für eine Tabelle aufbereitet.

6.2 Ausblick

Die erste Anwendung zeigt, wie es möglich ist, dynamische Listen zu erstellen. Dies kann genutzt werden, um auch weitere Auswahlmöglichkeiten bezüglich der Ticketauswahl zu treffen.

Auch kann sich mit der Authentifizierung noch auseinandergesetzt werden. Zurzeit benutze ich meine Anmeldedaten im Programmcode. Hier wäre es eine schöne Möglichkeit, dass jeder Benutzer das Programm nutzen kann, ohne den Code vorher anzufassen. Dafür kann zu Anfang eine Abfrage erfolgen über die Zugangsdaten. Hier ist auf eine erhöhte Sicherheit zu achten.

Außerdem kann man sich noch mit der Visualisierung der Daten auseinandersetzen, sodass es nicht nur möglich ist, Tabellen zu generieren, sondern Grundlage der getroffenen Auswahl auch verschiedene Diagramme. Hierfür gibt es einige JavaScript-Bibliotheken und Frameworks, die eine geeignete Visualisierung ermöglichen.

7 Literaturverzeichnis

Atlassian. (2024). *JQL: Erste Schritte mit der erweiterten Suche in Jira.* Abgerufen am Juni 2024 von https://www.atlassian.com/de/software/jira/guides/jql/overview#what-is-jql

Atlassian. (2024). *Willkommen bei Jira.* Abgerufen am 24. 05 2024 von https://www.atlassian.com/de/software/jira/guides/getting-started/introduction#dig-into-specific-features

Bernardo, R. (2022). *JiraApi Github.* Abgerufen am Juni 2024 von https://jira-node.github.io/class/src/jira.js~JiraApi.html

Flaviocopes et al. (2024). *Introduction to Node.js.* node.js. Abgerufen am Juni 2024 von https://nodejs.org/en/learn/getting-started/introduction-to-nodejs

Hertzen, N. v. (2022). *html2canvas 1.4.1.* Abgerufen am Juni 2024 von https://cdnjs.cloudflare.com/ajax/libs/html2canvas/1.4.1/html2canvas.min.js

IBM. (2024). *Was ist eine REST-API?* Abgerufen am Juni 2024 von https://www.ibm.com/de-de/topics/rest-apis

IONOS. (2020). *CRUD: die Basis der Datenverwaltung.* Abgerufen am Juni 2024 von https://www.ionos.de/digitalguide/websites/web-entwicklung/crud-die-wichtigsten-datenbankoperationen/

Kenneth Glassey et al. (2021). *jsPDF - PDF Document creation from JavaScript.* Abgerufen am Juni 2024 von https://cdnjs.cloudflare.com/ajax/libs/jspdf/2.4.0/jspdf.umd.min.js

Luber, D.-I. S. (2018). *Was ist Power BI?* Bigdata Insider. Abgerufen am Juni 2024 von https://www.bigdata-insider.de/was-ist-power-bi-a-676381/

Microsoft. (2024). *Power BI.* Abgerufen am Juni 2024 von https://www.microsoft.com/de-de/power-platform/products/power-bi?market=de

Salesforce. (2024). *Was ist Tableau.* Abgerufen am Juli 2024 von https://www.tableau.com/de-de/why-tableau/what-is-tableau

StrongLoop, Inc. . (2024). *Express.* Abgerufen am Juni 2024 von https://expressjs.com/de/

TrainingDotCom. (2016). *About Node.js, and why you should add Node.js to your skill set?* training.com. Abgerufen am Juni 2024 von https://web.archive.org/web/20170401061100/http://blog.training.com/2016/09/about-nodejs-and-why-you-should-add.html

8 Anlagen

8.1 HTML-Seiten

8.1.1 Start.html

```
<!DOCTYPE html>
<html lang="de">
<head>
	<meta charset="utf-8">
	<title>BSP</title>
</head>
<body>
	<h1>Welche Jiraversion willst du verwenden</h1>
	<br>
	<form action="/wait.html" method="get">
	<input type="radio" id="alt" value="alt" name="jira">
	<label for="alt">Altes Jira</label>
	<br>
	<input type="radio" id="neu" value="neu" name="jira">
	<label for="neu">Neues Jira</label>
	<br>
	<input type="submit" value="Weiter">
	</form>
</body>
</html>
```

8.1.2 chooseProject.html

```
<!DOCTYPE html>
<html lang="de">
<head>
	<meta charset="utf-8">
	<title>BSP</title>
	<link rel="stylesheet" href="style.css">
</head>
<body>
	<h1>Folgende Projekte stehen zur Auswahl</h1>
	<br>
	<form action="/table.html" method="get">
		<input id="placeholder">
		<input type="submit" value="Wählen">
	</form>


</body>
</html>
```

8.1.3 listeWaehlen.html

```
<!DOCTYPE html>
<html lang="de">
<head>
	<meta charset="utf-8">
	<title>BSP</title>
	<link rel="stylesheet" href="style.css">
	<script
src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/2.4.0/jspdf.umd.min.js"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/html2canvas/1.4.1/html2canvas.min.js"></sc
ript>
</head>
<body>
	<div id="tableContent"></div><br>
	<button id="pdfButton" onclick="savePDF()" style="visibility:hidden">Als PDF
speichern</button><br><br>
	<h1>Wählen Sie nun ihr Parameter aus</h1><br><br>
	<span>Welches Produkt ist betroffen?</span><br><br>
	<input type="radio" name="component" id="client" value="eCMS-Client"><label
for="client" id="clientLabel">Client</label><br>
	<input type="radio" name="component" id="server" value="eCMS-
Server"><label for="server">Server</label><br>
	<input type="radio" name="component" id="dm" value="eCMS-DM"><label
for="dm" id="clientLabel">DM-Client</label><br>
	<input type="radio" name="component" id="agent" value="eCMS-Agent"><label
for="agent" id="clientLabel">DM-Agent</label><br><br>
	<span>Aufgetreten in Test oder Produktion?</span><br><br>
	<input type="radio" name="testProd" id="test" value="Test"><label for="test"
id="clientLabel">Test</label><br>
	<input type="radio" name="testProd" id="prod" value="Produktion"><label
for="prod">Produktion</label><br><br>
	<br><button id='button'>Erstelle Tabelle</button>
	<br><br><span id="hint" style="visibility:hidden">Wähle erst alle Fragen
aus<span>




</body>
</html>
<script>
	var ticketListe = jsonTicketListe;
</script>
<script src='liste.js'></script>
```

8.2 CSS-Datei

```
body{
	font-family: Arial;
}

table {
	width: 100%;
	border-collapse: collapse;
}
th, td {
	border: 1px solid black;
	padding: 8px;
	text-align: center;
}
th {
	background-color: #f2f2f2;
}

#hint{
	color: red;
}
```

8.3 Javascript-Datein

8.3.1 Liste.js

```
var countTable=0;
var priorities = ["1","2","3","4"]
var categories = ["10202","10203","10201","10600","10205", "10306"];
var compCheck;
var testCheck;
var valTable=[[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0]];
var cellID;

//Buttonlistener zur Erstellung der Tabellen
document.getElementById('button').addEventListener("click", function(){
	var checked = checkRadioButtons();
	if(!checked){
		document.getElementById('hint').style.visibility = 'visible';
		document.getElementById('pdfButton').style.visibility = 'hidden';
		return;
```

```
        }
        else{
                document.getElementById('pdfButton').style.visibility = 'visible';
                document.getElementById('hint').style.visibility = 'hidden';
        }

        addTable();
        countTickets();
});

//HTML-Tabelle hinzufügen
function addTable(){
        var table="<br><table><thead><tr><th colspan='5'
id='options"+countTable+"'></th></tr><tr><th>Kategorie</th><th>P1</th><th>P2</th>
<th>P3</th><th>P4</th></tr></thead><tbody><tr><td>Fehler</td><td
id='table"+countTable+"11'>Zelle 1.1</td><td id='table"+countTable+"12'>Zelle
1.2</td><td id='table"+countTable+"13'>Zelle 1.3</td><td
id='table"+countTable+"14'>Zelle 1.4</td></tr><tr><td>Kein Fehler</td><td
id='table"+countTable+"21'>Zelle 2.1</td><td id='table"+countTable+"22'>Zelle
2.2</td><td id='table"+countTable+"23'>Zelle 2.3</td><td
id='table"+countTable+"24'>Zelle 2.4</td></tr><tr><td>Doppelt / Bekannter
Fehler</td><td id='table"+countTable+"31'>Zelle 3.1</td><td
id='table"+countTable+"32'>Zelle 3.2</td><td id='table"+countTable+"33'>Zelle
3.3</td><td id='table"+countTable+"34'>Zelle
3.4</td></tr><tr><td>Änderungswunsch</td><td id='table"+countTable+"41'>Zelle
4.1</td><td id='table"+countTable+"42'>Zelle 4.2</td><td
id='table"+countTable+"43'>Zelle 4.3</td><td id='table"+countTable+"44'>Zelle
4.4</td></tr><tr><td>Wartung</td><td id='table"+countTable+"51'>Zelle 5.1</td><td
id='table"+countTable+"52'>Zelle 5.2</td><td id='table"+countTable+"53'>Zelle
5.3</td><td id='table"+countTable+"54'>Zelle 5.4</td></tr><tr><td>Keine
Kategorie</td><td id='table"+countTable+"61'>Zelle 6.1</td><td
id='table"+countTable+"62'>Zelle 6.2</td><td id='table"+countTable+"63'>Zelle
6.3</td><td id='table"+countTable+"64'>Zelle 6.4</td></tr><tr><td
id='gesamt'>Gesamt</td><td id='table"+countTable+"71'>Zelle 7.1</td><td
id='table"+countTable+"72'>Zelle 7.2</td><td id='table"+countTable+"73'>Zelle
7.3</td><td id='table"+countTable+"74'>Zelle 7.4</td></tr></tbody></table>";
        document.getElementById("tableContent").insertAdjacentHTML("beforeend",
table);
        countTable++;
}

//checken ob Felder ausgewählt wurden
function checkRadioButtons(){
        compCheck = document.querySelector('input[name="component"]:checked');
        testCheck = document.querySelector('input[name="testProd"]:checked');
        if(compCheck!=null&&testCheck!=null){

                return true;
```

```
	}
	else{
		return false;
	}
}

var issues = ticketListe.issues
var field;
function countTickets(){

	//Durch alle Tickets iterieren
	for (var i=0;i<issues.length;i++){
		field = issues[i].fields;
		if(field.components.length>0){
			//Berücksichtigung aller Tickets, die mit Produkttyp und
Zeitpunkt des Auftretens übereinstimmen
			if(field.components[0].name=="eCMS-Client, eCMS-
Server"||field.components[0].name==compCheck.value&&field.customfield_10000.valu
e==testCheck.value){
				//Durch alle Prioritäten iterieren
				for(var j=0;j<priorities.length;j++){
					if(field.priority.name==priorities[j]){
						//Durch alle Kategorie-IDs iterieren
						for(var k=0;k<categories.length;k++){

	if(field.customfield_10203.id==categories[k]){
								//Wert an der Stelle für
Priorität und Kategorie hochzählen
								valTable[j][k]++;
							}
						}
					}
				}
			}
		}
	}
	//Berechung der Gesamtticketanzahl und füllen der Tabelle
	for(var j=1;j<=priorities.length;j++){
		var totalAmount=0;
		for(var k=1;k<=categories.length;k++){
			var cellAmount =valTable[j-1][k-1];
			cellID= "table"+(countTable-1)+k+j;
			document.getElementById(cellID).innerHTML=cellAmount;
			totalAmount=totalAmount+cellAmount;
		}
		document.getElementById("table"+(countTable-
1)+"7"+j).innerHTML=totalAmount;
	}
```

```
            //Ausgewählte Parameter in Überschrift der Tabelle setzten
            document.getElementById("options"+(countTable-
1)).innerHTML=compCheck.value+", "+testCheck.value;
}

async function savePDF(){
            const { jsPDF } = window.jspdf;

        // Erfassen des Inhalts als Canvas
        const content = document.getElementById('tableContent');
        const canvas = await html2canvas(content);

        // Konvertieren von Canvas zu Bilddaten
        const imgData = canvas.toDataURL('image/png');

        const pdf = new jsPDF();

        // Hinzufügen des Bildes zum PDF
        const imgProps = pdf.getImageProperties(imgData);
        const pdfWidth = pdf.internal.pageSize.getWidth();
        const pdfHeight = (imgProps.height * pdfWidth) / imgProps.width;

        pdf.addImage(imgData, 'PNG', 0, 0, pdfWidth, pdfHeight);

        // Speichern des PDFs
        pdf.save('Tabelle.pdf');
}
```

8.3.2 Webserver.js

```
// Import packages
var JiraApi = require('jira-client');
var http = require('http');
var url = require('url');
var fs = require('fs');
var express = require('express');
const path = require('path');

var app = express();
var server = http.createServer(app);

// Set path
const options = path.join(__dirname);
var jira;

// Initialize Jira API
function setApi(hostUrl) {
   jira = new JiraApi({
      protocol: 'https',
      host: hostUrl,
      username: 'bohnet',
      password: 'Seevetal24',
      apiVersion: '2',
      strictSSL: true
   });
}

// Start server
app.listen(8080, function (err) {
   if (err) console.log(err);
   console.log("Server listening on PORT", 8080);
});

app.use('/', express.static(options));

app.get('/project.html', function (req, res) {
   var q = url.parse(req.url, true);
   var version = q.query.jira;
   console.log(version + " " + typeof(version) + " ");

   if (version == "neu") {
      console.log("helpdesk");
      setApi('helpdesk.dps.de');
   } else {
```

```
      console.log("servicedesk");
      setApi('servicedesk.dps.de');
   }

   var liste = "";
   jira.listProjects()
      .then(function (list) {
         console.log(list.length);
         for (var i = 0; i < list.length; i++) {
            liste += '<div><input type="radio" id="list' + i + '" name="project" value="' +
list[i].key + '"><label for="list' + i + '">' + list[i].name + '</label></div>';
         }
         fs.readFile(options + '/chooseProject.html', 'utf-8', function (err, data) {
            data = data.replace('<input id="placeholder">', liste);
            res.write(data);
            return res.end();
         });
      })
      .catch(function (err) {
         console.error(err);
      });
});

app.get('/table.html', function (req, res) {
   var q = url.parse(req.url, true);
   console.log("Search" + q.search);
   console.log("Query" + q.query.project);

   fs.readFile(options + '/listeWaehlen.html', 'utf-8', function (err, data) {
      var searchString = 'project = DEUBA AND status in (ANALYSE, RÜCKGABE,
RÜCKFRAGE, KLÄRUNG, NACHFRAGE, ANTWORT, INFORMATION, UMSETZUNG, TEST,
TESTDOKUMENTATION, RELEASENOTES, NUTZERDOKUMENTATION, FREIGABE, "TEST
NICHT OK", ABLEHNUNG, ENTWICKLUNGSFREIGABE, DESIGN)';
      jira.searchJira(searchString, optional = { fields: ['priority', 'components',
'customfield_10000', 'project', 'customfield_10203'] })
         .then(function (list) {
            data = data.replace("jsonTicketListe", JSON.stringify(list));
            console.log(data);
            res.write(data);

            return res.end();
         })
         .catch(function (err) {
            console.error(err);
            var error = toString(err);
            res.write(error);
            return res.end();
         });
```

```
    });
});

app.get('/', function (req, res) {
    fs.readFile(options + '/start.html', 'utf-8', function (err, data) {
        res.write(data);
    });
})
```