Dokumentation zur schulischen Projektarbeit

Fachinformatiker(in) für Anwendungsentwicklung
Ausbildungsjahr 3

Moritz Hamminger, Mohammad Younus Jabari, Martin Manka, Jonas Hellwig,
Maximilian König

BETT – Berichtsheft Editor Transaction Tool.

Berufliche Schule ITECH Elbinsel Wilhelmsburg

Berufsschule	Berufliche Schule ITECH Elbinsel Wilhelmsburg (BS14) Dratelnstraße 26 21109 Hamburg
Durchführungszeitraum	07.08.2020 – 10.09.2020
Lehrkraft	Lena Sandmann
E-Mail	lena.sandmann@itech-bs14.de

Inhaltsverzeichnis

1 Projekthintergrund 1
1.1 Die Berufsschule 1
1.2 Das Projektumfeld 1
1.3 Projektauftrag 2
2 Projektplanung 2
2.1 IST-Analyse 2
2.2 IST-Kritik 3
2.3 SOLL-Konzept 4
2.4 Vergleich der Technologien 4
2.5 Lastenheft 5
2.1 Wirtschaftlichkeitsanalyse 6
2.1.1 Projektkosten 7
2.1.2 Amortisationsrechnung 7
3 Projektdurchführung 8
3.1 Implementierung des Backends 8
3.2 Implementierung des Frontends 9
3.3 Deployment 9
4 Abschlussphase 10
4.1 Übergabe an den Auftraggeber 10
4.2 SOLL / IST Vergleich 10
4.3 Gruppen-Fazit 10
4.4 Ausblick 11
Glossar I
Quellenverzeichnis II
Anhänge III
A: Projektzeitplan: Vergleich geplante und tatsächliche Zeitliche Dauer III
B: ERM Diagramm BETT III
C: Struktogramm BETT IV

1 Projekthintergrund

Diese Dokumentation ist im Rahmen des fünften Unterrichtsblockes der Berufsschule zur Ausbildung zum/zur Fachinformatiker*In entstanden. Der Inhalt dieser Dokumentation ist nach bestem Wissen und Gewissen von allen Gruppenmitgliedern erarbeitet und zusammengetragen worden. Alle verwendeten Quellen sind im Quellenverzeichnis aufgeführt und an den entsprechenden Stellen im Text gekennzeichnet. Um das Lesen der Dokumentation für alle Interessierten zu erleichtern, werden Abkürzungen wie z.B. BS (Berufsschule) verwendet.

1.1 Die Berufsschule

Die ITECH ist eine Berufsschule (BS) mit vielfältigen Möglichkeiten für seine Schuler. Hier bekommen die Schüler*Innen die Chance, sich umfangreiches Wissen in den Bereichen Informations-, Chemie- und Elektrotechnik, im Rahmen der dualen Ausbildung oder in unseren vollzeitschulischen Bildungsgängen (um einen höheren Bildungsabschluss zu erlangen) anzueignen.

Die BS ITECH bietet zusätzlich ein vielfältiges Angebot an Zusatzqualifikationen, wie Englisch-Sprachzertifikate oder finanziell unterstützte Praktika an, mit denen die beruflichen Chancen gezielt gesteigert werden können. Mit der Zusatzqualifikation „Dual Plus Fachhochschulreife“ erhalten besonders motivierte Auszubildende die Möglichkeit, parallel zur Ausbildung die Fachhochschulreife zu erlangen. Dafür wird ein freiwilliger und kostenloser Zusatzunterrichtangeboten. Die ITECH ist eine autorisierte Cisco Network Academy und kann ihren Schüler*Innen daher die Vorbereitungskurse für das renommierte Cisco Certified Network Associate (CCNA) samt Zertifikats-Prüfung anbieten.

Für angehende Fachinformatiker in den Bereichen Systemintegration und Anwendungsentwicklung bietet die ITECH jedes Jahr mindestens eine bilinguale Klasse angeboten. Hier wird in einzelnen Modulen der fachliche Inhalt auf Englisch vermittelt.

Die technische Ausstattung der BS ITECH reicht von modern eingerichteten Laboren für gewerblich-technische Chemie-Berufe, über vernetzte Unterrichts- und Projekträumen bis hin zu einem schulweitem WLAN, auch für private Notebooks, Handys und Tablets.

Die innovative Internetplattform der ITECH bietet den Schüler*Innen Zugriff auf verschiedene eLearning-Angebote, wie z.B. die Lernplattform Moodle. Außerdem haben die Schüler*Innen die Möglichkeit, ihr eigenes ePortfolio anzulegen. Mit Hilfe von Mahara können sich die Schüler*Innen ihren persönlichen Lern- und Arbeitsbereich schaffen.

1.2 Das Projektumfeld

Im Projektumfeld werden wir kurz alle Beteiligten des Projektes benennen. Die Lehrkräfte des Unterrichtsfaches Anwendungsentwicklung sind in diesem Projekt Auftraggeber und Ansprechpartner zugleich. Sie haben uns den Auftrag erteilt und stehen uns für Fragen und bei Problemen zur Verfügung. Auftragnehmer sind wir als Entwickler-Team. Das Projekt soll von uns alleine durchgeführt und ohne fremde Hilfe erarbeitet werden. Die gesamte interne und externe Kommunikation wird von

uns übernommen. Sämtliche Arbeitsprozesse, wie Planung, Realisierung und Implementierung der APP finden in der BS oder in der Heim-Beschulung statt.

Nun werden wir die technische und die organisatorische Umgebung des Projektes beschreiben.

Zuerst werden wir die organisatorische Umgebung etwas genauer beleuchten. Hier ist zu beachten, dass dieses Projekt im Rahmen des Hybriden Berufsschulblocks erarbeitet wird. Ein Teil der Gruppe nimmt von zuhause am Unterricht teil, der andere Part ist vor Ort in den Unterrichtsräumen der BS. Um das Projekt zu organisieren, muss viel über Videokonferenztools kommuniziert werden. Chat-Programme wie z.B. Discord werden benutzt, um Fragmente der Dokumentation oder des Projekts miteinander zu Teilen. Um die Programmierfortschritte zu dokumentieren und zu versionieren haben wir ein Repository auf der Internet-Seite „www.github.com“ angelegt und dieses gemeinschaftlich benutzt. Die Kommunikation mit den Lehrkräften findet überwiegend mündlich statt, Mail und Videotelefonie müssen, aufgrund des hybriden Unterrichts, ebenfalls des Öfteren genutzt werden.

Zu dem technischen Umfeld des Projekts gehören die Arbeitsmittel, in unserem Fall die Computer, Server, aber auch die Arbeitsplätze der einzelnen Gruppenmitglieder. Gearbeitet wird in unserem Fall mit den vom Betrieb bereitgestellten Notebooks. Auf einem von uns bereitgestellten Server wird unser fertiges Produkt laufen und über den Browser erreichbar sein.

1.3 Projektauftrag

Ziel des Projektes ist, die bisherige Art und Weise der manuellen Anfertigung des Berichtsheftes durch eine Webbasierte Lösung zu ersetzen. Die neue Anwendung soll den Auszubildenden ermöglichen, ihr Berichtsheft von überall anzufertigen, auf einer Datenbank abzuspeichern und die bereits angefertigten Seiten aufzurufen oder zu bearbeiten. So kann sichergestellt werden, dass der Auszubildende auch bei Wechsel des Standortes, der Abteilung oder des Arbeitsplatzes Zugriff auf sein Berichtsheft hat. Die zu verwendenden Programmiersprachen sind vom Entwickler-Team frei wählbar. Die Anwendung soll zwingend über ein „graphical user interface“´(GUI) verfügen.

Zusätzlich soll mindestens ein Design-Pattern bei der Entwicklung des Tools eingesetzt werden. Ebenfalls soll die Anwendung über eine Anbindung zu einer frei wählbaren Datenbank verfügen.

Über die Anwendung hinaus, soll ein Klassendiagramm, ein UML oder PAP/Struktogramm sowie ein ERD in dritter Normalform (NF) angefertigt werden. Die Anwendung soll am Ende der Projektzeit in einer Live-Demo vorgeführt und der Quellcode erklärt werden.

2 Projektplanung

2.1 IST-Analyse

Zum jetzigen Zeitpunkt gibt es weder eine offiziell unterstützte Lösung der Handelskammer noch in der Industrie verwendete Best Practises zum Führen und Verwalten der Tätigkeitsnachweise von dualen Auszubildenden. Jeder Betrieb ist somit selber dafür verantwortlich das Format, die Führung und die sichere Aufbewahrung aller Tätigkeitsnachweise über mehrere Jahre hinweg sicherzustellen.

Für das Format gibt es zwar von der Handelskammer bereitgestellte Vorlagen, jedoch ist deren Verwendung keine Verpflichtung, solange ein Tätigkeitsnachweis für den gesamten Zeitraum der

Ausbildung geführt wurde und minimale Formale Kriterien eingehalten wurden, wird der Tätigkeitsnachweis als gültig angesehen. Der Betrieb muss also individuell entscheiden, welche Vorlage verwendet wird. Im Zweifelsfall muss der Auszubildende selber die Korrektheit des Formats gewährleisten, wenn vom Betrieb keine konkreten Vorlagen gegeben sind. Sobald sich die formalen Anforderungen der Handelskammer an die von den Auszubildenden geführte Tätigkeitsnachweise ändern sollten, müssen alle bestehenden Einträge manuell angepasst werden.

Wenig Einheitlichkeit besteht ebenfalls bei der Wahl des genutzten Mediums für die Führung des Tätigkeitsnachweises. Oft werden diese noch auf Papier geführt. Und selbst bei einem digital erstellten Tätigkeitsnachweis wird dieser oft ausgedruckt, um diesen mit einer authentischen Unterschrift zu versehen. Bei den digital geführten Tätigkeitsnachweisen ist die Wahl des gewählten Textbearbeitungsprogramms ebenfalls nicht einheitlich. Vorlagen sind oft Software oder auch plattformspezifisch und somit nicht für jeden nutzbar. Beim Migrieren zu einem anderen Betriebssystem oder einem anderen Textbearbeitungsprogramm können oft Fehler auftreten, die im schlimmsten Fall die gesamte Formatierung zerstören können. Das Konvertieren zu einer Unterschiedlichen Vorlage ist oft selbst bei einem digitalen Format nicht oder nur sehr schwer möglich.

Die korrekte Sicherung des Tätigkeitsnachweises stellt oft eine besondere Hürde dar, obwohl bei einem Verlust der unweigerliche Verlust von Monaten oder sogar Jahren an gesammelten Einträgen droht. Oft können die Tätigkeitsnachweise nur teilweise oder mit einem Bruchteil der Details in mühevoller Handarbeit wiederhergestellt werden. Wenn der Tätigkeitsnachweiß in analoger Form geführt wird, ist die Aufbewahrung oft besonders fehleranfällig und aufwendig, da Einträge bei physikalischen Schäden ersatzlos verloren gehen und physischer Platz zur Aufbewahrung benötigt wird. Das Sortieren und Finden einzelner Einträge muss ebenfalls manuell erledigt werden, was bei einer Fehlenden Struktur sehr zeitaufwändig sein kann. Mit einem digital geführten Tätigkeitsnachweis erledigen sich zwar viele der organisatorischen Probleme im Vergleich zu seinem analogen Gegenstück, jedoch ist der Verlust der Daten jederzeit durch Hardwareausfälle oder sogar aus Versehen möglich. Jedoch bietet der digitale Tätigkeitsnachweis die Möglichkeit, diverse Cloud Anbieter oder bestehende Backup Lösungen zu nutzen und so die bereits geschriebenen Einträge vor einem kompletten Verlust zu bewahren

Um seinen Tätigkeitsnachweis in einem von der Handelskammer anerkannten Format zu führen und dessen Datensicherheit zu garantieren ist eine Menge Eigeninitiative und Wissen nötig, zu der nicht jeder Auszubildende die Bereitschaft oder die Möglichkeit hat.

2.2 IST-Kritik

In der jetzigen Situation ist es sowohl für Betriebe als auch für Auszubildende sehr unübersichtlich und intransparent wie der Tätigkeitsnachweis idealerweise geführt und verwaltet werden sollte. Außerdem ist das korrekte Führen eines Tätigkeitsnachweises mit einem hohen Maß an Eigeninitiative und Aufwand verbunden, der durch die fehlende Automatisierung eines solchen Prozesses den Betrieben und Auszubildenden zur Last fällt.

Jede Änderung des Formats ist umständlich und oft sogar frustrierend, da oft alle bereits bestehenden Einträge manuell angepasst werden müssen. Dies kann unter anderem mit einem enormen Zeitaufwand einhergehen, oder auch mit einem enormen Kostenaufwand, wenn der Azubi dadurch seine produktive Arbeit vernachlässigen muss. Insbesondere wenn Änderungen dieser Art öfter und relativ spät in der Ausbildung notwendig sind. Unabhängig ob der Tätigkeitsnachweis digital oder analog geführt wird gibt es also kaum eine Möglichkeit manuelle Anpassungsarbeit zu sparen.

Die Datensicherung des Tätigkeitsnachweises gestaltet sich ebenfalls schwierig für die meisten Betriebe und Auszubildenden. Lediglich mit einer durch Cloud oder Backup getriebenen Speicherungsstrategie kann eine verlustfreie und verhältnismäßig aufwandsarme Verwaltung der Tätigkeitsnachweise gewährleistet werden. Jede alternative Art der Aufbewahrung oder Speicherung ist meist ein Kompromiss aus Datensicherheit und Aufwand. Die plattformabhängigen Vorlagen machen eine Migration oft unmöglich.

2.3 SOLL-Konzept

Um die genannten Probleme anzugehen, soll ein Software Tool entwickelt werden, dass digitale Lösungen für die vorhandenen Probleme bieten und für eine breite Menge an Benutzer nutzbar ist. Hierbei soll eine grafische Oberfläche entstehen, auf der Berichtshefteinträge erstellt und bearbeitet werden können.

Die fehlende Automatisierung bei Veränderung der Vorlage oder der formalen Anforderungen soll dadurch gesteigert werden, dass ein fertiges Dokument aus den in der grafischen Oberfläche erstellten Einträgen generiert werden kann. So muss bei einer Veränderung lediglich die Generierung neu angestoßen werden, eine manuelle Bearbeitung fällt damit komplett weg.

Die Datensicherheit soll dadurch gegeben sein, dass die auf der grafischen Oberfläche erstellten Einträge in einer Datenbank gespeichert werden sollen, und somit der Benutzer von der Verpflichtung der sicheren Aufbewahrung befreit wird.

Die plattformspezifischen Einschränkungen sollen dadurch umgangen werden, indem eine grafische Oberfläche für das Web konstruiert wird. So kann die maximale Anzahl an Benutzer und die höchstmögliche Plattformunabhängigkeit erreicht werden.

2.4 Vergleich der Technologien

Der generelle Aufbau der Applikation besteht aus einem diskreten Frontend und Backend, die unabhängig voneinander eingesetzt und bereitgestellt werden können, sowie einer getrennten Datenbank. So sind eine voneinander unabhängige Entwicklung und Wartung möglich. Beide Teile kommunizieren über REST Endpunkte. Da REST ein sprachenagnostischer Standard ist, steht es den Entwickler jederzeit frei die Implementierung einer Softwarekomponente zu ändern oder sogar komplett auszutauschen.

Zur Authentifizierung zwischen Frontend und Backend wird ein JSON Web Token eingesetzt. So sind private Informationen des Users wie Username und Password sicher vor Angreifern, da sie nicht im Klartext übertragen werden, und ohne das Verschlüsselungspasswort des Backend zu kennen, ist es nicht möglich den Token wieder zu entschlüsseln. Somit bietet es dem Frontend die Möglichkeit ohne Kenntnisse vom Backend das Token zu erhalten und bei jeder Anfrage mitzugeben. Der Token kann im Session Speicher des Browsers festgehalten werden, damit die Anmeldung beim erneuten Laden der Seite nicht wiederholt werden muss. Somit bietet das JSON Web Token massive Vorteile in den Bereichen Sicherheit und Komfort.

Die Infrastruktur der Applikation basiert auf Docker Containern. Mit Docker kann der voneinander unabhängige Einsatz, der für die REST Architektur der Applikation benötigt wird, leicht umgesetzt werden. Ein Container hat darüber hinaus den Vorteil, dass er eine abgekapselte Umgebung bietet, bei denen sich keine Teile der Applikation gegenseitig beeinflussen können. Das Bauen und Konfigurieren eines Containers sind außerdem komplett reproduzierbar, was den Einsatz plattformunabhängig und flexibel macht und das spätere Wechseln der Plattform oder sogar des Betriebssystems ermöglicht. Um mehrere Container im Verbund einzusetzen wird Docker Compose verwendet. Dieses sehr gut in Docker integrierte Tool ermöglicht das kollektive bauen, starten und anhalten mehrerer Container und kann weitere Beziehungen zwischen ihnen konfigurieren. Somit bietet der Einsatz von Docker und Docker Compose große Vorteile in Vergleich zu traditionellen Methoden und Konfigurationen.

Das Backend ist in Java mithilfe des Frameworks Springboot geschrieben. Mit Springboot ist es vergleichsweises leicht und Programmierer freundlich, einen REST Endpunkt zu implementieren. Es ist ebenfalls sehr gut möglich, eine Authentifizierung durch zum Beispiel JSON Web Token einzubauen. Die benötigte Verbindung zu einer persistenten Datenbank ist durch die zusätzliche Library Hibernate beinahe trivial. Mit Hibernate muss dich der Programmierer werden um das Konvertieren von Datenbankrückgaben zu für die Programmierung nutzbaren Objekten, noch um das Anpassen der Datenbankstruktur bei Änderungen an der Applikationsstruktur befassen. Somit ist das Schreiben von REST Endpunkten und die erforderliche Datenbankanbindung so unbeschwert wie möglich.

Das Frontend ist in React mit TypeScript geschrieben. React ist ein Framework, was es massiv erleichtert eine reaktive und moderne Benutzeroberfläche für den Browser zu erstellen. Es erleichtert Prozesse wie State Management und Dokumentaktualisierung. Darüber hinaus gibt es eine vielfältige Bandbreite an weiteren Libraries, um eine diverse Liste an Aufgaben zu vereinfachen. Hierzu zählen unter anderem die optische Darstellung, das Versenden von REST Anfragen und das Routing innerhalb der Benutzeroberfläche. Außerdem wird anstatt JavaScript die alternative TypeScript verwendet. TypeScript garantiert die Typsicherheit vom zu JavaScript kompiliertem Quellcode durch statische Typen und Compile Time Checks. Somit ist es leicht möglich eine anprechende grafische Oberfläche zu programmieren und es können im Vergleich zu JavaScript teilweise sehr schwer zu findende Fehler vermieden werden.

2.5 Lastenheft

Durch ein Fachgespräch mit dem Auftragsgeber, vertreten durch die JAV, Personalabteilung und Ausbilder wurde folgendes Lastenheft entworfen.

Programm:

- Eine Web-Applikation, über die man sein Berichtsheft schreiben und in einer Datenbank speichern kann.
- Die User sollen sich einloggen und dort einen persönlichen Bereich vorfinden, hier sollen alle bereits geschriebenen Seiten aufgelistet sein, diese können bearbeitet werden.
- Auch neue Seiten können hier hinzugefügt werden.
- Aus der Applikation heraus sollen fertige Seiten des Berichtshefts ausgedruckt werden können.

Programmiersprachen und Datenbank:

- Java/ Springboot für das Backend
- React/ Typescript für das Frontend
- PostgreSQL als Datenbank

Must have:

- Loginmaske()
  - Die User sollen sich über eine Loginmaske anmelden können
- Registrierungsmaske
  - Die User sollen sich einmalig zu Beginn der Nutzung über eine Registrierungsmaske registrieren können
- Profilseite
  - Die User sollen die Möglichkeit haben, Name/Abteilung/Tel. Nr abzuändern. Die Daten werden zu Beginn einmalig aus der Registrierungsmaske gezogen und können jederzeit vom User geändert werden.
- Form-Vorlage für Berichtsheft
  - Die User sollen die Möglichkeit haben, das Berichtsheft über eine vorgefertigte Formvorlage zu schreiben
- Liste aller Berichte
  - Die User sollen die Möglichkeit haben, die einzelne Berichte einzusehen und, wenn nötig, zu bearbeiten
- Bearbeitungsfunktion
  - Die User sollen die Möglichkeit haben, die bereits geschriebenen Seiten des Berichtshefts bearbeiten zu können.
- Speicher/Druckfunktion (lokal speichern)
  - Die einzelnen Berichte sollen lokal als z.B. PDF gespeichert werden oder direkt aus der Applikation gedruckt werden können.
- DB-Anbindung
  - Die einzelnen Berichte sollen in einer Datenbank gespeichert werden können

Nice to have:

- Mail-Funktion
  - direkt an den Ausbilder zur Kontrolle.
- Feedback zu aktuellem Stand des Berichtsheftes an User.

2.1 Wirtschaftlichkeitsanalyse

Im Folgenden wird die Wirtschaftlichkeit des Projektes betrachtet. Dazu werden die Kosten, die im Rahmen dieses Projektes entstehen, dem Nutzen gegenübergestellt, der für den Aufraggeber mit der Einführung der neuen Entwicklungsinfrastruktur entsteht.

2.1.1 Projektkosten

Es wird ein pauschaler Arbeitslohn von 55 Euro pro Stunde angenommen. In der Tabelle 1 sind die einmaligen Investitionskosten aufgeführt, die durch das Projekt entstehen.

Table 1: Übersicht der Projektkosten

Kosten und Bezeichnung	Einheit	Preis pro Einheit	Anzahl	Gesamt
Personalkosten				
Entwicklungskosten	Std.	55,00€	250,00	13.750,00€
Deploymentkosten	Std.	55,00€	2	110,00€
			Gesamtkosten	13860,00€

Neben den einmaligen Projektkosten entstehen aus dem Betrieb des neuen Berichtshefttools außerdem laufende Kosten – Kosten für eine virtuelle Maschine und Wartungskosten. In Tabelle 2 sind diese variablen Kosten aufgeführt. Es wird 1 Stunde Wartung pro Monat veranschlagt.

Table2: Übersicht der Variablenkosten

Kosten und Bezeichnung	Einheit	Preis pro Einheit	Anzahl	Gesamt
Serverkosten				
Virtuelle Maschiene pro Monat	Stk.	18,44€	12,00	221,28€
Wartungskosten				
Deploymentkosten	Std.	55,00€	12,00	660,00€
			Gesamtkosten	881,28€

2.1.2 Amortisationsrechnung

Unter Verwendung der Durchschnittsmethode wird eine anteilige Amortisationsrechnung, in der die Zeitersparnis als Gewinn-Grundlage betrachtet werden, durchgeführt. Veranschlagt wird eine Zeitersparnis von 10 Minuten pro Berichtsheft, der Durchschnittsstundenlohn der am Prozess beteiligten Person wird auf 20€ veranschlagt und anhand von 30 Auszubildenden pro Lehrjahr berechnet.

```
Zeitersparnis pro Jahr in Stunden = 90 Berichte pro Woche * 52 Wochen * 10 Minuten
/ 60 = 780h
```

Jährliche Ersparnis durch Zeitersparnis = Zeitersparnis pro Jahr in Stunden * 20€ = 15.600,00€

Einmalige Investitionskosten	13.860 Euro
Laufende Wartungskosten pro Jahr	881,28 Euro
Ersparnis durch Zeitersparnis	15.600 Euro
Jährlicher Gewinn	15.600 Euro – 881,28 Euro = 14718.72 Euro
Amortisationsdauer in Jahren	13.860 Euro / 14.718,72 Euro = ~ 0,9417

Nach diesem Ergebnis lässt sich die Amortisationszeit auf ca. 1 Jahr setzen.

3 Projektdurchführung

Die Durchführung des Projektes teilen wir unter den Mitgliedern nach ihren jeweiligen Stärken ein, sodass sich drei der fünf Mitglieder für das Entwickeln der Backend-Software beteiligen und zwei für die Frontend-Software. Zudem stellt sich ein Mitglied bereit, die Software in Docker zu containerisieren. Durch die anhaltende Corona-Krise besteht nach wie vor die Hürde der stetigen Kommunikation, da sich zwei der Gruppenmitglieder im Home-Office befinden. Das lösen wir durch fast tägliche Meetings auf diversen Kommunikationsservices, sowie mit Chaträumen auf WhatsApp.

3.1 Implementierung des Backends

Für das Backend, welches mit Springboot gebaut wird, kann man sich auf einer von den Entwicklern angebotenen Webseite ein Gerüst, welches man sich bereits leicht konfigurieren lassen kann. Dazu gehören die Java-Version (11), das Build-Management-Tool (Gradle) und auch Plugins, von denen Spring Security zum Authentifizieren und Spring JPA zum Anbinden der Datenbank genutzt wird. Durch diese Tools erspart man sich Zeit, die nahezu selbe Applikation erneut schreiben zu müssen.

Als erstes werden grundlegende REST-Endpunkte eingebaut, welche die zuvor konzipierten Entitäten per Abfrage zurückgeben können. Der Rückgabewert ist in dem Fall das POJO der Entität im JSON-Format. Hierfür wird eine Abfrage an einen vorher festgelegten Endpunkt an eine Controller-Klasse abgewartet, welche über den Service das Objekt vom Repository abfragt. Dieses läuft in der frühen Phase der Entwicklung auf einer In-Memory-Datenbank, um das schnelle Testen zu erleichtern. Zudem wird das Plugin „Flyway“ installiert, welche den Prozess der Datenbankerstellung und dessen Befüllung erleichtert. Zudem kann man bereits mit der Anfrage mit Parametern experimentieren, welche später eventuell erforderlich sein werden. Anschließend werden Endpunkte eingeführt, mit denen man Einträge hinzufügen kann (POST), sowie bearbeiten (PUT) und löschen (DELETE) kann. Zudem muss noch beachtet werden, dass in der Datenbank eine Verbindung zwischen „User“ und „ReportEntry“ (Berichtsheft-Eintrag) besteht, damit auch alle Einträge eines bestimmten Users abgefragt werden können. Einträge eines gelöschten Users sollen mitgelöscht werden.

Mit dem Fertigstellen der grundlegenden Funktionen beginnt die Implementierung von JWT (JSON Web Token). Das soll das Registrieren und Authentifizieren der Benutzer ermöglichen, sowie den Service sicherer machen. Grundlegend soll mit Hilfe von Spring Security jeder Endpunkt bis auf „authenticate“ und „register“ nur nach einem erfolgreichen Login aufrufbar sein. Das eingegebene Passwort wird beim Registrieren durch die in Springboot mitgelieferte Hashfunktion „Bcrypt“ verschlüsselt. Wenn ein Benutzer sich erfolgreich anmeldet, liefert das Backend einen zeitlich begrenzten Token an das Frontend, der es dem Benutzer ermöglicht auf zuvor gesperrte Endpunkte zuzugreifen. Es ist zusätzlich möglich aus dem Token den Benutzernamen des aktuellen Users herauszulesen, sodass eine personalisierte Anzeige trotz gleichen Endpunktes möglich ist. Konkret für das Projekt bedeutet das: Jeder User sieht nur seine eigenen Einträge im Berichtsheft.

Somit wurde das Backend fertiggestellt und braucht nur noch ein Frontend welches es sich zu Nutzen macht.

3.2 Implementierung des Frontends

Das Frontend soll dem Benutzer eine Oberfläche im Browser bieten, mit der man ohne Programmierverständnis an die Informationen des Backends kommt. Der Beginn der Implementation ähnelt sich dem der Springboot Applikation, man lässt sich das Gerüst einer React-Anwendung- mit dem npm-Modul „create-react-app“ generieren. Das sorgt ebenfalls dafür, dass man keinen Code schreiben muss, welcher für die meisten Applikationen gleich aufgebaut ist.

Das Grundgerüst, welches mit simplen, nicht stilisierten Komponenten aufgebaut ist, wird als erstes implementiert. Hierbei wird auch sichergestellt, dass alle Anfragen an das Backend richtig formuliert sind, sowie sich das Token nach dem Anmelden im Header der Anfrage befindet. Wenn das Backend nicht die genau erwünschte Anfrage erhält, bekommt man im Frontend einen „Bad Request“ (400) oder „Unauthorized“ (401). Eine simple Anzeige der Berichtshefteinträge, welche farblich anzeigt, ob sich an einem bestimmten Tag ein Eintrag befindet, wird implementiert. Wichtig ist, dass der Code stabil ist, eine gutaussehende Anzeige, wird danach mit React-Bootstrap erreicht. Ein konfigurierter Datepicker wird eingebaut. Dieser soll beim Auswählen eines Wochentages die gesamte Woche, die den Tag beinhaltet anzeigen. Dafür wurde ein eigenes Skript geschrieben. Text-Inputs und Buttons, die das Bearbeiten der User und Einträge ermöglicht werden eingebaut.

Wenn man alle Funktionen des Backends fehlerfrei nutzen kann, gilt es React-Bootstrap zu nutzen, um dem Frontend mit relativ wenig Aufwand einen professionellen Eindruck zu geben.

3.3 Deployment

Zunächst gilt es zwei sogenannte Dockerfiles für jeweils Front- und Backend zu schreiben. Diese sollen den Code innerhalb des Containers kompilieren und starten. Der große Vorteil von Docker ist, dass diese Container komplett unabhängig vom Host-System agieren. Das heißt, dass jede Maschine mit einer Docker-Installation das Projekt starten kann.

Zudem gibt es im Root-Verzeichnis eine „Docker-Compose“-Datei, die die beiden Dockerfiles aufruft, einige Konfigurationen vornimmt, sowie Parameter und Ports einpflegt. In dieser Datei wird auch ein dritter Container für die Datenbank definiert. Dieser ist so konfiguriert, dass hinzugefügte Daten auch nach dem Beenden persistiert werden.

4 Abschlussphase

4.1 Übergabe an den Auftraggeber

Die Übergabe an den Auftraggeber findet in der BS statt. Die fertige Anwendung wird vom Entwickler-Team in einer Live-Demo vorgeführt. Anschließend wird der Quellcode Schritt für Schritt mit den Auftraggebern durchgegangen. Besonders Interessante Code-Teile werden hervorgehoben. Nach mündlicher Besprechung der Anwendung werden Quellcode, ERM und Struktogramm in digitaler Form an den Auftraggeber übergeben.

4.2 SOLL / IST Vergleich

Für den Soll-Ist-Vergleich ist es zunächst notwendig, den SOLL-Zustand noch einmal hinzuziehen. Dieser Ist im Lastenheft unter Kapitel 3.3 zu finden.

Erfolgreich umgesetzte „must-have“ Kriterien:

- Eine **Loginmaske** wurde als Startseite in der Web-Applikation implementiert
- Ebenfalls auf der Starteseite wurde die **Registermaske** implementiert
- Eine **Profilseite**, in der das Profil auch bearbeitet werden kann, wird nach erfolgreichem Login/ register aufgerufen
- Eine **Formvorlage**, in der das Berichtsheft gefüllt werden kann wurde erfolgreich implementiert. Die erstellten Ausbildungsnachweise können in einer **Datenbank gespeichert werden**. Die Einträge können ebenfalls **bearbeitet und gelöscht** werden. Die gespeicherten Ausbildungsnachweise können als **PDF gespeichert** werden.

Eine Mail-Funktion haben wir aus Zeitgründen leider nicht mehr implementieren können. Die darauf aufbauende „Feedback“-Funktion konnten aus demselben Grund nicht mehr implementiert werden

Diese Funktionen sind als „nice-to-have“ geplant worden und daher nicht in die Zeitplanung mit ein.

SOLL-IST-Zeitvergleich

Die Soll-, als auch die tasächlich für das Projekt benötigten IST-Stunden für werden in dem Anhang A aufgelistet. Abweichungen sind nicht aufgetreten.

4.3 Gruppen-Fazit

Da die Gruppe mit verschiedenem Vorwissen in dieses Project gestartet ist, fällt der Ehrfahrungszuwachs der einzelnen Gruppenmitglieder leicht unterschiedlich aus. Die Entwickler haben sich jedoch gegenseitig die Vorkenntnisse im jeweiligen Fachgebiet nähergebracht. So haben

die Entwickler mit Vorkenntnissen im Bereich „React/ Typescript“ den anderen Gruppenmitgliedern zu Beginn des Projekts eine Einführung gegeben und anders herum.

Die Kommunikation und Aufgabenverteilung innerhalb der Gruppe hat hervorragend funktioniert. So konnte das Entwickler-Team effizient an der Entwicklung der Anwendung arbeiten.

Alle geplanten Core-Features konnten innerhalb des geplanten Zeitraumes implementiert werden. Die Anwendung funktioniert einwandfrei und wurde bereits auf Fehler getestet.

Abschließend bleibt nur zu sagen, das Projekt ist ein voller Erfolg!

4.4 Ausblick

Alle vorgesehenen Core-Features wurden in der geplanten Zeit implementiert. Nachdem die Anwendung an den Auftraggeber übergeben wird, könnten wir uns vorstellen, die Applikation in regelmäßigen Abständen mit Updates zu versehen. Hier würde die Priorität in erster Linie um die beiden nicht implementierten „nice-to-have“ Features liegen.

Außerdem soll die Anwendung mindestens einmal im Quartal mit Sicherheitsupdates versehen werden.

Glossar

Begriff	Erklärung
GUI	„graphical user interface“ oder auch „Benutzeroberfläche“
CCNA	Cisco Certified Network Associate ist ein Zertifikat, welches grundlegendes Wissen im Bereich Netzwerke belegt.
JSON	JavaScript Object Notation
JWT	JSON Web Token
NPM	Node Package Manager
POJO	Plain Old Java Object
REST	Representational State Transfer
Repository	ein verwaltetes Verzeichnis zur Speicherung und Beschreibung digitaler Objekte
DB	Data Base (Datenbank)

Quellenverzeichnis

Digitale Quellen

Spring Boot
https://spring.io/projects/spring-boot [letzter Zugriff: am 11.09.2020]

React – A JavaScript libary for building user Interfaces
https://reactjs.org/docs/getting-started.html [letzter Zugriff: am 11.09.2020]

Docker: Empowering App Development for Developers
https://www.docker.com [letzter Zugriff: am 11.09.2020]

React Bootstrap
https://reactjs.org/docs/getting-started.html [letzter Zugriff: am 11.09.2020]

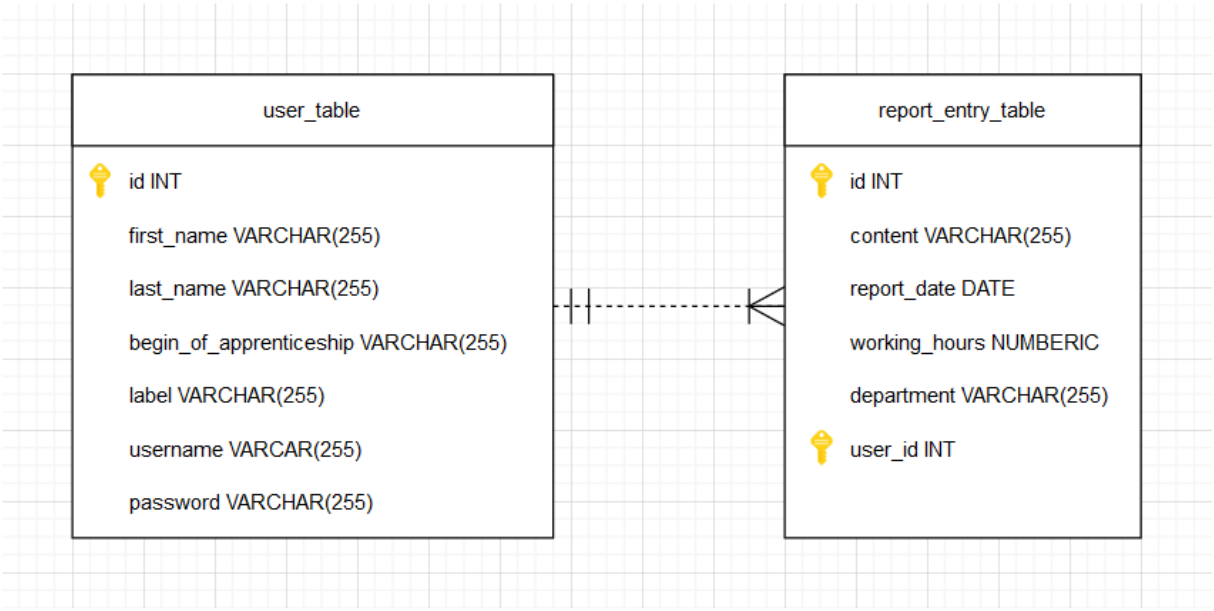
JSON Web Tokens – jwt.io
https://jwt.io [letzter Zugriff: am 11.09.2020]

Anhänge

A: Projektzeitplan: Vergleich geplante und tatsächliche Zeitliche Dauer

Projektphasen und Tätigkeiten	zeitliche Dauer	
	geplante	tatsächliche
Planungsphase	**Std.**	**Std.**
• Analyse des IST-Zustandes	1 Std.	1 Std.
• Erstellung des SOLL-Konzeptes	1 Std.	1 Std.
• Vergleich und Auswahl technischer Lösungen	2 Std.	2 Std.
• Durchführung einer Kosten-Nutzen-Analyse	2 Std.	2 Std.
Realisierungsphase	**Std.**	**Std.**
• Entwicklungskosten	233 Std.	233 Std.
Evaluationsphase	**Std.**	**Std.**
• Erstellung der Projektdokumentation	10 Std.	10 Std.
• Präsentation des Projektergebnisses dem Auftraggeber	1 Std.	1 Std.
Gesamt	**250 Std.**	**250 Std.**

B: ERM Diagramm BETT



C: Struktogramm BETT