



Modul SET SoS 2020

Miniprojekt

ClubLink – Die simple Vereinsverwaltung für Fußballvereine

ClubLink

a database project by:



Unser Projekt trägt den Namen „ClubLink“ und soll eine Anwendung zur Verwaltung von wachsenden Fußballvereinen liefern. Hierzu sollen einfach strukturierte und übersichtliche Methoden dazu dienen, alltägliche Aufgaben von Trainern, Buchhaltern und/oder Schatzmeistern zu vereinfachen.

Inhaltsverzeichnis

0. Einleitung.....	4
0.1. Prolog	4
1. Planungsphase	5
1.1. Planung und Vorbereitung	5
1.2. Spezifikation von Funktionalitäten.....	5
1.3. Aufstellen von Anwendungsfällen	6
1.4. Aufteilung der Rollen	6
1.5.1. Anforderungsanalyse	6
1.5.2. Softwarearchitektur	7
2. Implementierung des Projektes.....	9
2.1. Technischer Aufbau	9
2.2.1 Implementierung der Datenbank.....	10
2.2.2 Implementierung des User-Interface	11
3. Testverfahren.....	14
3.1. Aufstellen von Testfällen.....	14
3.2. Durchführung der Testfälle	14
4. Fazit	15
4.1.1. Gegenüberstellung von Planung und Ergebnis	15
4.1.2 Ideen zur Optimierung und Erweiterung	15
4.2. Epilog.....	16
Quellenverzeichnis	17

Abbildungsverzeichnis

Abbildung 1.1: Aufbau der Software-Architektur	8
Abbildung 1.2: Granulare Darstellung des XAMPP-Softwarestacks	8
Abbildung 2.1: Das Entity-Relationship-Diagramm	10
Abbildung 2.2: Elemente der Tabelle "role_ability"	11
Abbildung 2.3: Startseite mit Verlinkungen zur Ostfalia Homepage	12
Abbildung 2.4: Overlay im Login-Fenster	12
Abbildung 2.5: Overlay im Registrierungs-Fenster	13
Abbildung 2.6: Dashboard für einen Benutzer mit Rolle "Accountant"	13
Abbildung 4.1: Der Kommunikationsprozess zwischen Test und Entwicklung	15

0. Einleitung

0.1. Prolog

ClubLink – Ein Name, der im Zuge eines Brainstormings zu Anfang des Projektzeitraumes, entstanden ist. Nach schneller Übereinkunft entschied sich das Team bei diesem Projektnamen zu bleiben, um die geforderte datenbankbasierte Applikation mit diesem zu taufen. Im Zuge der Bearbeitung soll eine Anwendung entstehen, welche Fußballvereinen die Möglichkeit geben soll Finanzen sowie Bestände schnell und effizient zu managen. Hinter den Kulissen sollen innerhalb einer Datenbank diverse Aufgaben übernommen werden, die zwar händisch möglich, jedoch sehr zeitaufwendig wären. Hierbei sollte sich zeigen wie sinnvoll die Idee ist, da in diesem Anwendungsbereich viele Daten abgespeichert und dokumentiert werden müssen. Diese Umgebung bildet ein optimales Einsatzgebiet für eine Datenbank.

1. Planungsphase

1.1. Planung und Vorbereitung

In den ersten Schritten sollten die Grundmerkmale der Anwendung festgelegt werden. Hierzu gehörten, Name und Projektentwurf, aber auch ein Spezifizieren der Zielgruppe hierbei gehen insbesondere die Auswahl von Sprache und Nutzung von Fachbegriffen mit ein. Sprachlich fiel der Entschluss, die Applikation in englischer Sprache zu gestalten, um eine große Zielgruppe einzubinden. Die Verwendung von Fachbegriffen ist dabei eher projektspezifisch. ClubLink soll dabei für ein durchschnittliches Vereinsmitglied einfach und verständlich sein. Dies gewährleistet zudem eine Möglichkeit der Nutzung auch bei geringeren Sprachkenntnissen.

1.2. Spezifikation von Funktionalitäten

Ein weiterer essenzieller Bestandteil der Planung lag im groben spezifizieren von konkreten Funktionen und Aufgaben, über welche die Software verfügen soll. Im Zuge dessen haben wir in der Planungsphase folgende beispielhafte Fälle entwickelt, welche von unserer Software aufgefangen und bearbeitet werden:

- ➔ Jeder User soll eine Individuelle eindeutige Zugangskennung erhalten können, über welche nur dieser mit der Software interagieren kann.
- ➔ Für die Nutzer/Innen der Datenbank sollen Rollen und damit verbundene Rechte von einer Administration verteilt werden können.
- ➔ Verschiedene Berechtigungsstufen sollen Zugang auf bestimmte Daten beschränken oder eröffnen können.

Diese Funktionen und Schritte sollen im Folgenden an expliziten Anwendungsfällen veranschaulicht werden.

1.3. Aufstellen von Anwendungsfällen

- ➔ Ein **Trainer** erhält eine Übersicht, in welcher Lagerbestände („stocks“) eingesehen, angelegt und gelöscht werden können.
- ➔ Ein **Accountant** kann neue Finanzbudgets und/oder Kosten anlegen und löschen, auf welche ein **Trainer** oder **Member** keinen Zugriff hat.
- ➔ Ein **Administrator** kann einem **User** die einzelnen Rollen: „**Accountant**“, „**Trainer**“, „**Member**“ oder „**Player**“ zuweisen.

Diese Anwendungsfälle sind beispielhaft und sollen Aufschluss darüber geben, wie die Software zum Ende der Implementierung arbeiten soll.

1.4. Aufteilung der Rollen

Der Prozess der teaminternen Rollenverteilung ist ein entscheidender Schritt in der Softwareentwicklung. Um die Stärken jedes Teammitglieds ermitteln und somit das bestmögliche Arbeitsumfeld zu schaffen zu können, sind wir wie folgt vorgegangen: Jedes Mitglied schreibt zunächst eigene Stärken auf, welche bei der Softwareentwicklung zum Tragen kommen könnten. Danach beurteilten die jeweils anderen Teammitglieder ebenfalls die Stärken der Teamkollegen. Zum Schluss wurden Überschneidungen gesucht und mit diesen Vorschlägen für mögliche Rollen getroffen.

Das Ergebnis war, dass die Teammitglieder Julien Müsker und Kerim Sjenar aufgrund ihrer Leidenschaft zur Programmierung für die Implementierung und Entwicklung der Datenbank so zuständig sein sollten, während Faruk Bal mit einem Sinn für Ordnung und strukturierte Abläufe sowie einem kritischen Blick zum Softwaretester des Projektes wurde.

Weiterhin erklärte sich Kerim Sjenar bereit als Projektleiter die Kommunikation mit den Auftraggebern zu übernehmen.

1.5.1. Anforderungsanalyse

Bei der Anforderungsanalyse ging das Team sehr strukturiert vor und hat diese grob in zwei Entwicklungsprozesse eingeteilt. Nämlich das Backend und Frontend der Applikation. Nach einigen Grundüberlegungen und dem Aufstellen der Anwendungsbeispiele, wurde deutlich welche Anforderungen unsere Software bzw.

unser Projekt definieren würden. Es fing an bei der Datenbank, wie diese strukturiert ist und welche Daten und Relationen grob benötigt werden. Die Anforderungen wurden simultan zur Softwareentwicklung erweitert, wodurch sich die Anforderungen auch in eine Art Meilensteine entwickelt haben. Dies hat sich positiv auf die Entwicklung, sowie das Test-Management ausgewirkt, da die Anforderungen in Paketen implementiert wurden und anschließend auch isoliert testbar waren. Der zweite Entwicklungsprozess war die Entwicklung des Frontends und der Einbindung der Datenbank in diese. Das Ziel dabei war eine simple und übersichtliche Darstellung der Applikation und der jeweiligen Funktionen. Hier war die große Schwierigkeit, eine Plattform zu finden, um die Datenbank ohne großen Aufwand einzubinden, so dass die Implementierung mit einer möglichst kurzen Einarbeitungsphase, relativ schnell umgesetzt werden kann.

➔ *Siehe Anhang: „Anforderungsanalyse“*

1.5.2. Softwarearchitektur

Wenn man die wesentlichen Aspekte von Software-Architektur beschreiben soll, muss man zunächst den Begriff „Software-Architektur“ selber klären. Wikipedia und der Standard ISO/IEC 42010 definieren Software-Architektur als die grobgranulare Strukturierung eines Software-Systems. Dazu passt der Begriff „Architektur“: Die Architektur eines Gebäudes gibt die groben Strukturen vor. Die meisten Architektur-Entwürfe zeigen dementsprechend die Aufteilung des Systems in größere Komponenten.

In diesem Projekt war die Grundlage der Architektur die Einbindung der Datenbank und ihrer Funktionalitäten im Backend mittels XAMPP und im Frontend mittels „Laravel“. Dadurch wurde eine einfache Einrichtung der Benutzeroberfläche, sowie der darin inbegriffenen Manipulation der Datenbank möglich. Der Fokus bestand darin die Implementation aufgrund fehlender Grundlagen in diesem Bereich, mithilfe von kompakten Software-Stacks wie Laravel und XAMPP durchzuführen. Dadurch wurde die Anpassung der Kommunikationsschnittstellen um einiges erleichtert und die Entwicklung beschleunigt, da das Hauptaugenmerk auf dem Entwurf und der Implementation der wesentlichen Datenbank und ihrer Funktionen lag.

Der Technische Aufbau soll im Folgenden die Konzepte von Laravel und XAMPP, sowie weiteren Bestandteilen der Entwicklung, näherbringen.

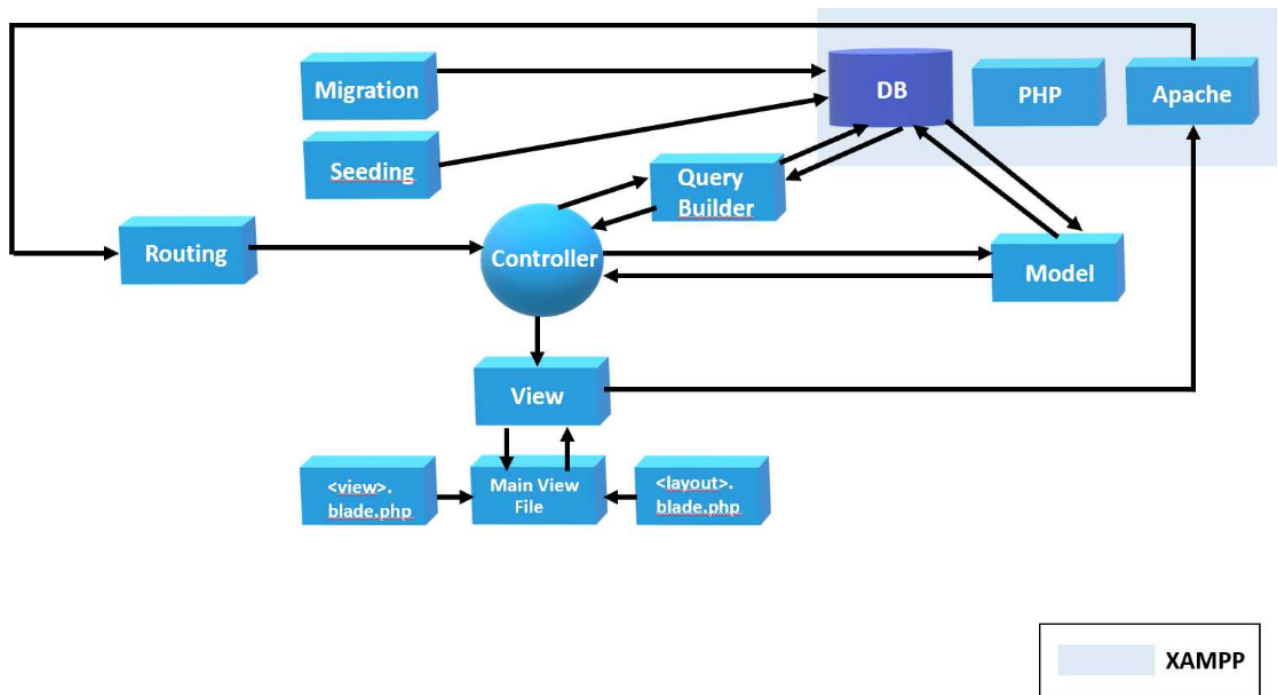


Abbildung 1.1: Aufbau der Software-Architektur

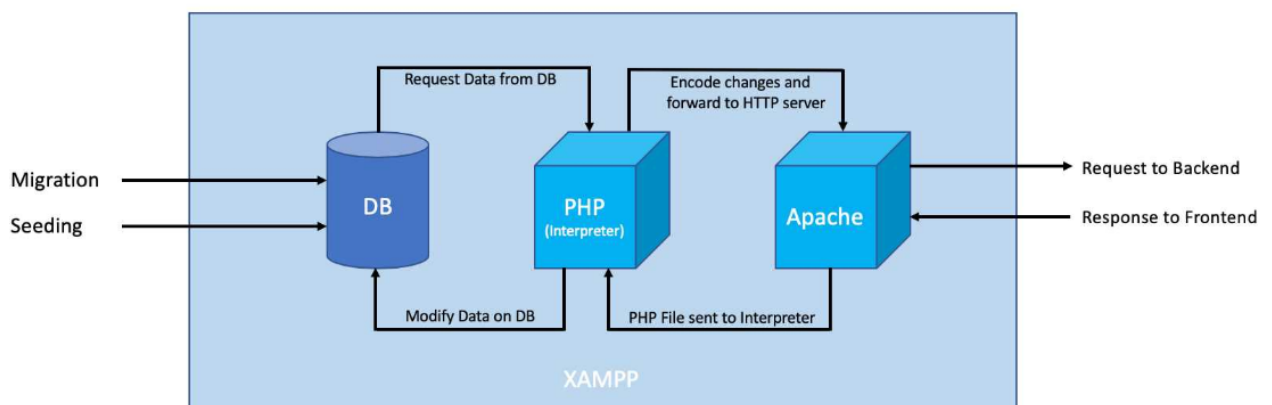


Abbildung 1.2: Granulare Darstellung des XAMPP-Softwarestacks

2. Implementierung des Projektes

2.1. Technischer Aufbau

Nach der Fertigstellung der Planung und Vorarbeiten, war es nun an der Zeit die Überlegungen in der Praxis anzuwenden und die Software nach unseren Spezifikationen zu Implementieren. An dieser Stelle war eine Menge an Recherche nötig, da zum Schluss eine funktionierende Anwendung mit funktionalem „back-end“ so wie ansehnlichen und nutzerfreundlichen „Frontend“ entstehen sollte. Da kein Teammitglied bisher an ähnlichen Projekten gearbeitet hat, mussten viele Zusammenhänge erst verstanden und dazu einige Quellen zur Orientierung gefunden werden. Zu Anfang fiel die Wahl auf **XAMPP** als Grundlage und geeigneten „Software-Stack“ zur Implementierung des „Backends“. Mit **XAMPP** haben die Entwickler eine Verwaltungsmöglichkeit die eine Datenbank mit dem relationalen Datenbank-Management-System **MySQL** auf einem **Apache** Webserver zu verwalten. Im nächsten Schritt fand die Suche nach einem geeigneten Dienst für „front-end“ Entwicklung statt, die Wahl fiel hierbei auf **Laravel**, ein open-source PHP-Webframework welches sich gut mit einem RDBMS verknüpfen lässt und über den PHP-Dependency-Manager **Composer** Installiert wurde. **Laravel** bietet zudem das Command-Line-Interface („CLI“) **Artisan** und die Möglichkeit von „Database-Seeding“ wobei es sich um die schnelle Initialisierung von Datensätzen innerhalb der Datenbank handelt, was für spätere Testfälle von Nutzen sein kann. Ein weiterer Vorteil ist die von **Laravel** bereitgestellte Datenbank-Migration, welche die Daten von unserem lokalen Rechner auf den Webserver laden kann. Durch den in **XAMPP** integrierten PHP-Interpreter ist es nun möglich die Datenbank mit der Hilfe von PHP zu befüllen. Für die Bearbeitung von Daten und generieren von dynamischen Inhalten auf der Nutzeroberfläche war letztlich noch die kostenlose Serverumgebung „open-source-server-environment“ **Node.js** benötigt.

2.2.1 Implementierung der Datenbank

Auf Basis von XAMPP sollte das Projekt nun vorerst eine Datenbank erhalten, welche die Realisierung der vorangestellten Funktionen und Anwendungsfälle ermöglicht. Es war also zunächst das Anlegen verschiedener Tabellen nötig, welche die Daten abspeichern und auch außerhalb des „Development-Teams“ nachvollziehbar strukturieren. In Abbildung 2.1 folgt das Entity-Relationship-Diagramm:

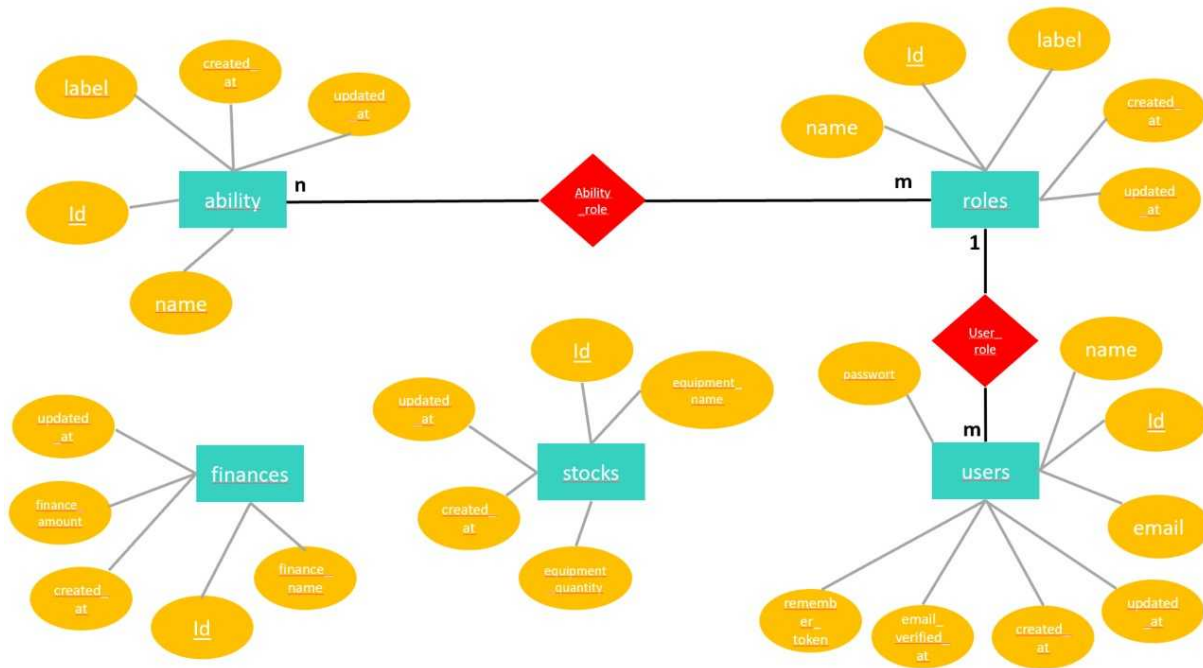


Abbildung 2.1: Das Entity-Relationship-Diagramm

Für die Datenbanknutzer/Innen war eine Nutzertabelle („users“) nötig, welche im Zuge der Entwicklung vorerst die Attribute ID, Name und Passwort als erhielt, jedoch später um Zusatzattribute wie Zeitstempel („timestamps“) und sog. „Tokens“ erweitert wurde, welche Aufschluss geben sollen ob zum Beispiel eine E-Mail verifiziert ist oder ein Passwort für spätere Logins „gemerkt“ werden soll. Weiterhin kamen unter anderem auch eine Rollentabelle („roles“), Fähigkeitentabelle („abilities“) oder eine Finanztable („finances“) dazu, welche ebenfalls mit entsprechenden Attributen versehen wurden. Um nun auch Verknüpfungen erstellen zu können sollten weitere Tabellen hinzukommen, welche diese festhalten und somit einen Zusammenhang zwischen sonst unabhängigen Tabellen bilden. Ein Beispiel hierfür wäre die Table „ability_role“, welche als Attribute unter anderem eine Fähigkeiten-ID sowie eine Rollen-ID erhält. Diese sollen hier als sog. „foreign key“ fungieren. Hiermit wird ein

Attribut einer anderen Tabelle als Schlüssel („key“) für die damit verbundene Tabelle bestimmt, um so eine Relation herzustellen.
















#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/> 1	role_id 	bigint(20)		UNSIGNED	No	None			 Change  Drop  More
<input type="checkbox"/> 2	ability_id  	bigint(20)		UNSIGNED	No	None			 Change  Drop  More
<input type="checkbox"/> 3	created_at	timestamp			Yes	NULL			 Change  Drop  More
<input type="checkbox"/> 4	updated_at	timestamp			Yes	NULL			 Change  Drop  More

Abbildung 2.2: Elemente der Tabelle "role_ability"

In dieser Abbildung 2.2 erkennt man beispielsweise die Fähigkeiten-ID welche als „foreign-key“ initialisiert sei, was wiederum an dem silberfarbenen Schlüsselsymbol zu erkennen ist.

Analog zu genannter Verknüpfung sollen im weiteren auch Nutzer („user“) an die verschiedenen Rollen („roles“) gekoppelt werden können.

2.2.2 Implementierung des User-Interface

Für bessere Übersicht und Nutzerfreundlichkeit ist eine gute grafische Oberfläche ein essenzieller Faktor, somit bestand der nächste Entwicklungsschritt in der Implementierung des User-Interface. Wie in der Planung beschlossen, sollte Laravel hier als Tool zur Erstellung abhelfen. Da innerhalb der Recherche auch ein Beispiel für eine Website-Oberfläche mit Laravel entdeckt wurde, sollte diese im Folgenden als Orientierung dienen und auf das „ClubLink“-Projekt angepasst werden. Gebettet ist das User-Interface hierbei in HTML inklusive CSS, weshalb auch hier diverse Nachforschungen von den Software-Entwicklern eingeleitet werden mussten, um mit Syntax und Funktion der Sprache umgehen zu können. Zur Aufmachung des Interface sei vorerst die Startseite zu beschreiben, bei welcher der Titel der Applikation, so wie Verlinkungen zum Ostfalia-Netzwerk und der „Laravel“-Homepage dargestellt werden. Im Weiteren sind die Buttons **Login** und **Register** im oberen Bildschirmrand dargestellt, welche mit den jeweiligen zugehörigen Ansichten gelinkt sind. Hilfreich ist hier die Funktionalität von „Laravel“, da an dieser Stelle mit „blade“ eine PHP-Template Engine bereitgestellt wird, welche ebenfalls die Implementierung von HTML Code erlaubt. So erhalten die zugehörigen PHP-Dateien im Dateiordner „views“ auch die Extension „.blade.php“. Im **Login** Bildschirm sollen beschriftete Eingabefelder für E-Mail und Passwort so wie ein Login-Button das Anmelden ermöglichen, wo hingegen

beim **Register** Bildschirm des Weiteren noch ein Name und eine Passwortwiederholung abgefragt werden. Loggt man sich als Nutzer erfolgreich ein sobald die Registrierung abgeschlossen ist, so öffnet sich der verlinkte „Home-Bildschirm“, in welchem Funktionen und Möglichkeiten dargestellt sind.

Beispielhafte Einblicke sollen die folgenden Grafiken geben:

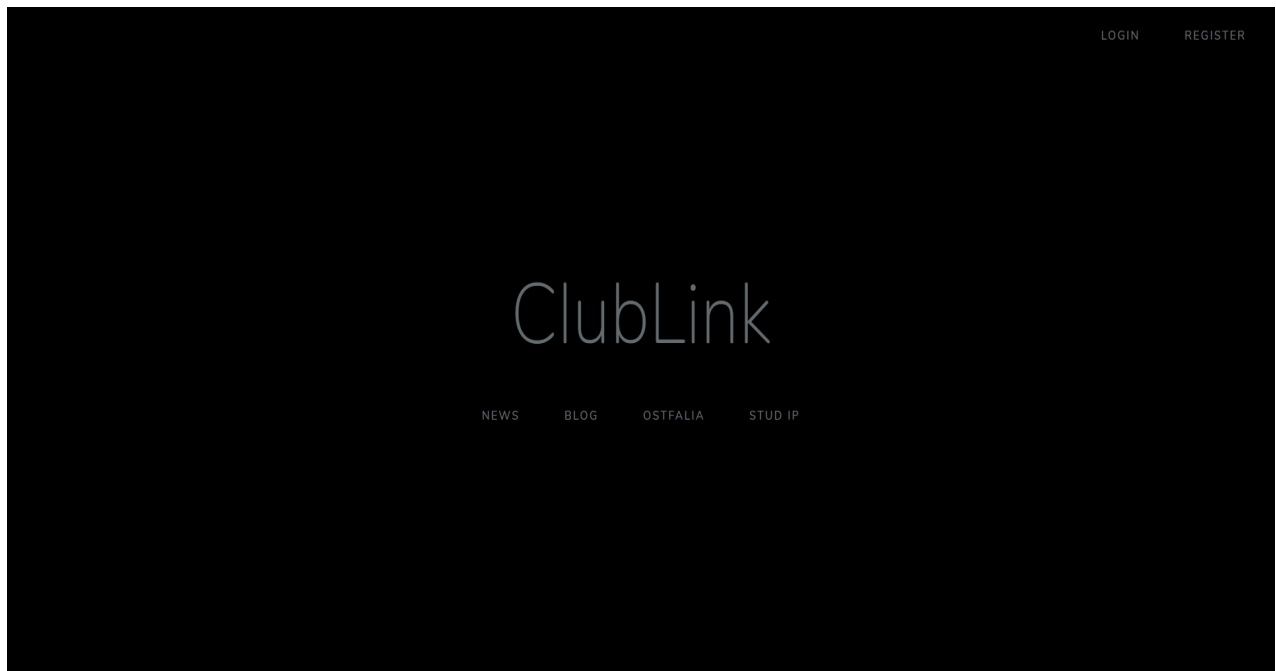
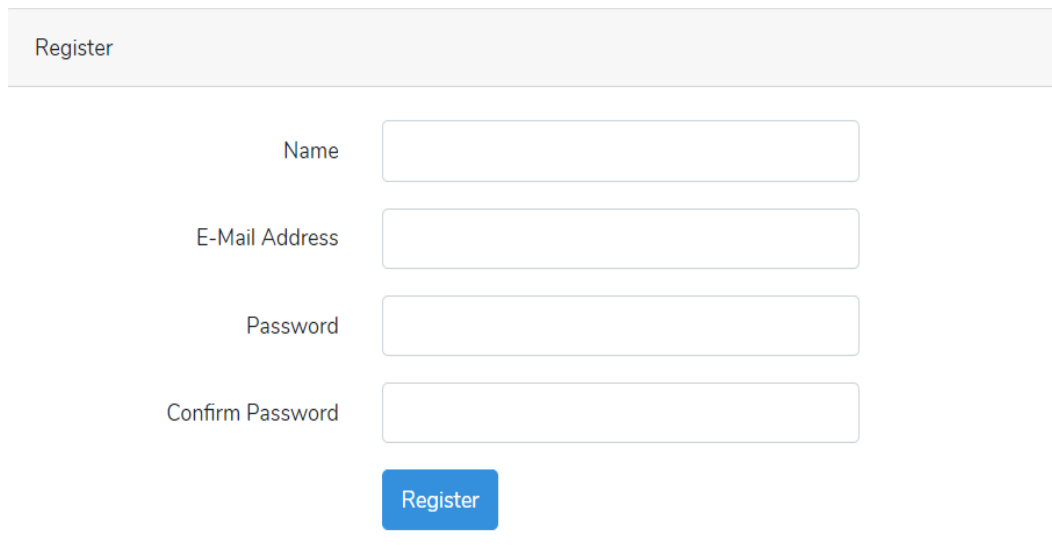


Abbildung 2.3: Startseite mit Verlinkungen zur Ostfalia Homepage

Abbildung 2.4: Overlay im Login-Fenster



Register

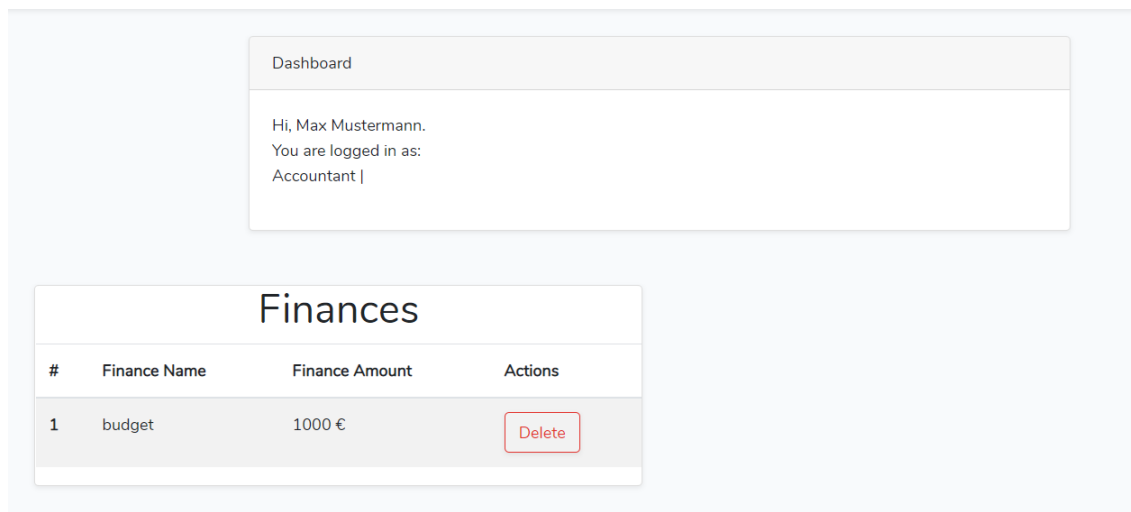
Name

E-Mail Address

Password

Confirm Password

Abbildung 2.5: Overlay im Registrierungs-Fenster



Dashboard

Hi, Max Mustermann.
You are logged in as:
Accountant |

Finances

#	Finance Name	Finance Amount	Actions
1	budget	1000 €	<input type="button" value="Delete"/>

Abbildung 2.6: Dashboard für einen Benutzer mit Rolle "Accountant"

3. Testverfahren

3.1. Aufstellen von Testfällen

Das Testen einer Software bzw. einer Datenbank beruft sich nicht auf wie fälschlicher Weise gedacht auf Fehlerbehebung, sondern auf Fehlererkennung. Der Tester erprobt, inwiefern die Anforderungen des Kunden umgesetzt wurden und ob es zu Fehlern kommt. Zu einem vollständigen Testverfahren müssen alle gestellten Anforderungen an die Software von den Testern des Teams beobachtet, geprüft und evaluiert werden. Testergebnisse sollen hierbei möglichst strukturiert dokumentiert werden, so dass bei späteren Auswertungen auch jeder Schritt und Vorgang genau nachvollzogen werden kann. Die jeweiligen Testfälle sollen über einzelne Indizes definiert sein, ergo: genau **eine** Test-ID soll den jeweiligen Testfall beschreiben. Etwa: *ID = Rg1* für den *ersten Registrierungstestfall*. Im Weiteren soll jeder Test auch explizit eine vorher gestellte Anforderung bedienen, auch dies soll der Übersicht zugunsten in der Testdokumentation festgehalten werden.

3.2. Durchführung der Testfälle

Zur Durchführung der Tests war es nun essenziell, dass die Dokumentation später für die Entwickler nachvollziehbar ist. Daher ist die Gliederung hier tabellarisch und für jeden Testfall alleinstehend gewählt.

➔ *Siehe Anhang: „**Testprotokoll**“*

4. Fazit

4.1.1. Gegenüberstellung von Planung und Ergebnis

Nach Auswertung der Testphase lag es nun daran die gesammelten Informationen zu nutzen, um den Software-Entwicklern mittels *Feedback* aufzuzeigen wo bestehende Probleme liegen. Jene Probleme mussten demnach klassifiziert und priorisiert werden, damit eine systematische Beseitigung ermöglicht wird. Zum Schluss soll der Kunde also eine funktionierende und möglichst fehlerfreie Software erhalten. Es ist hier wichtig, dass die Entwickler sich dabei erst auf die höher priorisierten Probleme fixieren, um die erheblichsten Schwachstellen zu eliminieren. Im Anschluss können dann die kleineren sog. „bugs“ behoben werden.

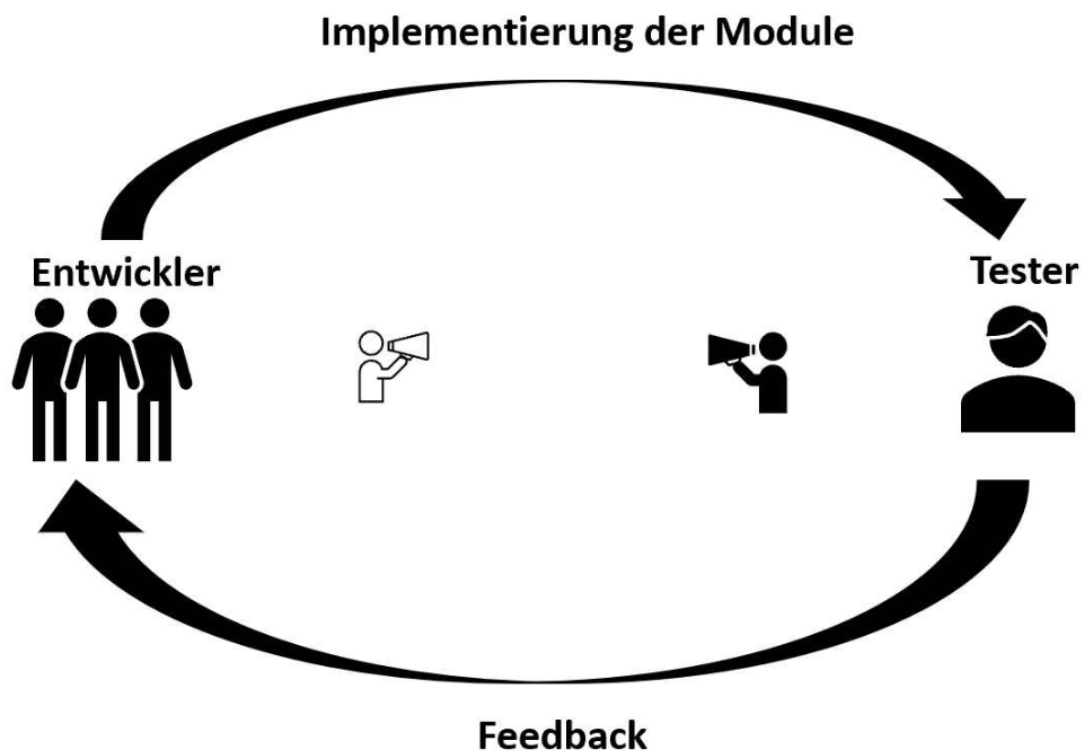


Abbildung 4.1: Der Kommunikationsprozess zwischen Test und Entwicklung

4.1.2 Ideen zur Optimierung und Erweiterung

Der Prozess der Softwareentwicklung ist stets ein fließender, dabei ist eine tatsächliche „Fertigstellung“ meist optional, da immer die Möglichkeit der Verbesserung besteht. Ein „perfekter“ Zustand kann folglich nie erreicht werden. Im Projekt „ClubLink“ waren bis zuletzt diverse Möglichkeiten angesetzt, die aufgrund des Zeitrahmens aber leider nie fertig implementiert werden konnten. Zum einen stand der Einbau einer „password-reset“ Funktion im Raum, welche dem Nutzer ermöglichen

sollte ein Passwort zu ändern. Dies wäre hilfreich, sobald ein Passwort falsch initialisiert oder sogar vergessen worden ist. Weiterhin sind natürlich diverse Erweiterungen von bereits genutzten Konzepten möglich, es könnten in etwa weitere Funktionen durch neue Tabellen und verbundene Relationen realisiert werden. Es ist erkennbar, dass die Möglichkeiten hier schier unendlich sind, weshalb „ClubLink“ an dieser Stelle in aktueller Form mit dem etwaigen Zusatz „Alpha 1.0“ fertiggestellt sein soll.

4.2. Epilog

Während wir als Team bei der Entwicklung der Software sehr viel über die Erstellung einer solchen gelernt haben, ist auch der Aspekt des dahinterstehenden Prozesses klar geworden. Die Entstehung einer Software ist letztlich keine einfache schnelle Implementierung sondern ein Zusammenspiel aus Teamwork, Voranstellungen und vor allem einer ausführlichen und strukturierten Planung. Auch wenn es keine einfache Aufgabe war und viel Arbeit und Recherche einher ging, war das Projekt „ClubLink“ dennoch eine tolle Erfahrung für uns als Team.

Quellenverzeichnis

- Prof. Dr. Steiner, Susanne – Datenbank, Skript 2020
- Dr. Marhenke, Jörg – SoftwareEngineering, Skript 2020
- Scotch, Chris - „Simple Laravel Login Authentication [<https://github.com/scotch-io/simple-laravel-login-authentication>]
- Scotch, Chris – “Simple and Easy Laravel Login Authentication” [<https://scotch.io/tutorials/simple-and-easy-laravel-login-authentication>]
- Lloyd, Holly; Prosper Otemuyiwa – “Creating your first Laravel App and Adding Authentication” [<https://auth0.com/blog/creating-your-first-laravel-app-and-adding-authentication/>]