

LeoTurnier - Turnierverwaltung

DIPLOMARBEIT

verfasst im Rahmen der

Reife- und Diplomprüfung

an der

Höheren Abteilung für Informatik

Eingereicht von:

Hain Lukas

Ecker Benjamin

Betreuer:

Franz Gruber-Leitner

Projektpartner:

Thomas Stütz

Leonding, April 2022

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Die Arbeit wurde bisher in gleicher oder ähnlicher Weise keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Die vorliegende Diplomarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Leonding, April 2022

S. Schwammal & S. Schwammal

Abstract

Brief summary of our amazing work. In English. This is the only time we have to include a picture within the text. The picture should somehow represent your thesis. This is untypical for scientific work but required by the powers that are. Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Zusammenfassung

Es wurde ein Prototyp einer Application entwickelt, die das Erstellen und Durchführen von Turnieren für den Turnierorganisator vereinfachen und rationalisieren soll. Die Turniere können unabhängig von der Sportart durchgeführt werden, außerdem werden 3 verschiedene Turniermodi ermöglicht, KO, jeder gegen jeden und eine Kombination aus beiden.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Inhaltsverzeichnis

1	Einleitung	1
1.1	Gliederung	1
2	Problemstellung	2
2.1	Ausgangssituation	2
2.2	Problemstellung	2
2.3	Aufgabenstellung	2
2.4	Ziele	3
3	Systemarchitektur	4
3.1	Verwirklichung der Anforderungen	4
3.2	Verwendete Technologien	4
4	Umsetzung	10
5	Zusammenfassung	11
	Literaturverzeichnis	V
	Abbildungsverzeichnis	VI
	Tabellenverzeichnis	VII
	Quellcodeverzeichnis	VIII
	Anhang	IX

1 Einleitung

1.1 Gliederung

- Kapitel 1: Einleitung

Hier wird die Diplomarbeit in ihre verschiedenen Kapitel gegliedert, zusammen mit einer kurzen Beschreibung des Kapitels.

- Kapitel 2: Problemstellung

Hier werden Ausgangssituation, Problemstellung, Aufgabenstellung und Ziele des Projekts beschrieben.

- Kapitel 3: Systemarchitektur

Hier wird beschrieben, wie die Anforderungen verwirklicht werden und welche Komponenten die Gesamtanwendung hat. Außerdem wird die Aufteilung der Diplomarbeit erläutert und alle verwendeten Technologien wie z.B. Programmiersprachen, Frameworks, IDE's aufgelistet sowie erklärt und deren Verwendung begründet.

- Kapitel 4a - Quarkus Backend

Hier werden Anforderungen, Datenmodell, verwendete Datenbank, Dokumentation der Endpoints, etc. erläutert und gerechtfertigt.

- Kapitel 4b - Angular Frontend

Hier werden Anforderungen, ein kurzes Benutzerhandbuch, verwendete Libs, Vorstellung der Verschiedenen Rollen und ihrer Rechte, etc. erläutert und gerechtfertigt.

- Kapitel 5: Zusammenfassung

Hier werden wichtigste Ergebnisse des Projekts, mögliche Erweiterungen (was könnte aus diesem Projekt noch Thema werden, das hier nicht betrachtet wurde) und eigene Erfahrungen erläutert.

2 Problemstellung

2.1 Ausgangssituation

Die HTL Leonding ist eine Höhere Technische Lehranstalt. Derzeit wird sie von ca. 1000 Schülern besucht, welche in eine der 4 Zweige:

- Informatik
- Medientechnik
- Elektronik
- Medizintechnik

unterrichtet werden. Öfters werden auch Sportevents in verschiedenen Sportarten wie zum Beispiel ein Fußball oder Volleyball veranstaltet.

2.2 Problemstellung

Derzeit werden an unserer Schule Sportturniere lediglich auf Papier festgehalten, was es relativ umständlich zu verwalten macht und sehr viel Zeit in Anspruch nimmt. Auch sich über den aktuellen Stand eines Turniers zu informieren ist nur mündlich oder an einer Pinnwand möglich.

2.3 Aufgabenstellung

Mit einer neuen Applikation wollen wir dieses bisherige Verfahren ersetzen und die Gestaltung und Verwaltung von Sportturnieren unserer Schule vereinfachen. Dabei soll die Applikation so gestaltet werden, dass sie mehrer Turnier-Modi unterstützt und auch unabhängig von der Sportart ist.

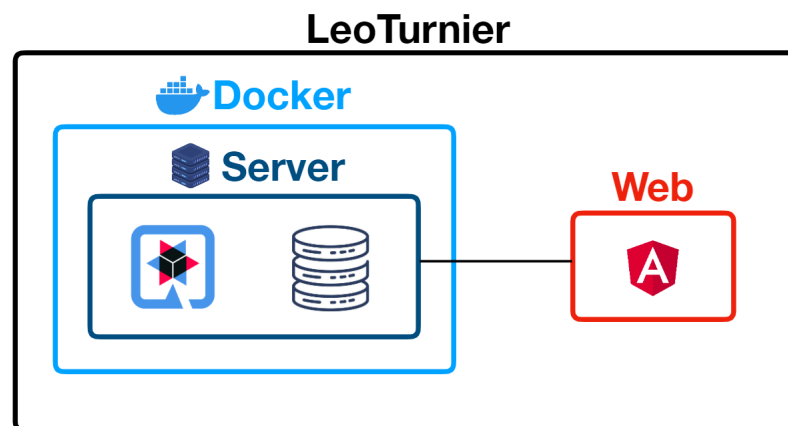
2.4 Ziele

Im Rahmen dieser Diplomarbeit soll nun eine Applikation erstellt werden die das Verwalten und die Informationsbeschaffung eines Turniers komplett digitalisiert und um vielfaches beschleunigen und vereinfacht. Die Informationsbeschaffung soll für jeder Person mit einem Internetzugang möglich sein, wobei das Verwalten nur ausgewählten Personen vorbehalten ist.

3 Systemarchitektur

3.1 Verwirklichung der Anforderungen

Abbildung 1: system architecture



3.2 Verwendete Technologien

3.2.1 Git



Git ist das mit Abstand am weitesten verbreitete Versionskontrollsystem der Welt. Der Name Git wird aus der britischen Umgangssprache übersetzt und bedeutet "Blödmann". [1] Es ist ein Open-Source Projekt, das ursprünglich 2005 von dem Entwickler des Linux Betriebssystem-Kernels entwickelt wurde. Es ist außerdem mit sowohl mit Windows als auch Linux Systemen kompatibel und auch in vielen verschiedenen IDEs integriert. Git ist ein verteiltes Versionskontrollsystem, was bedeutet, dass der Versionsverlauf nicht nur an einem Ort gespeichert ist, wie es bei älteren Versionskontrollsystemen der Fall war, sondern in jeder Arbeitskopie der gesamte Verlauf aller Änderungen im Repository (Aufbewahrungsort) enthalten sind.

[2]

Was ist ein Versionskontrollsystem?

Ein Versionskontrollsystem wie Git wird in der Softwareentwicklung verwendet, um Änderungen des Quellcodes zu speichern und zu verwalten. Hier gibt es drei Arten von Versionskontrollsystemen: [2]

- lokale Versionsverwaltung (Local Version Control System - LVCS):

Hier werden Dateien lokal einfach nur in ein separates Verzeichnis kopiert. [2]

- zentrale Versionsverwaltung (Centralized Version Control System - CVCS):

Hier gibt es einen zentralen Server, der alle versionierten Dateien verwaltet und es Personen ermöglicht, Dateien von diesem zentralen Repository abzuholen und auf ihren PC zu übertragen. [2]

- verteilten Versionsverwaltung (Distributed Version Control System - DVCS):

Wie weiter oben schon angesprochen ist diese Variante jene, die von Git benutzt wird. Hier gibt es zwar ein zentrales Repository, aber jede Person hat eine Kopie des Repositories und somit die vollständigen Projektdaten. [2]

Git Befehle

Git verwendet zum Verwalten eines Repositories das Terminal, hierzu gibt es einige wichtige Befehle. Zuerst kann man mit "git init" ein leeres Repository erstellen oder ein existierendes nochmal initialisieren, alternativ kann man mit "git clone" ein existierendes Repository kopieren. Damit ein File von Git gespeichert werden kann, muss man es zunächst mit "git add" zum Repository hinzufügen, das Ganze kann dann mit "git rm" wieder rückgängig gemacht werden. Mit "git status" wird der momentane Status aller Files im Repository angezeigt, das heißt, ob das File neu im Repository ist, ob es seit dem letzten Mal Speichern verändert wurde, oder ob es erst gar nicht im Repository vorhanden ist. Nun kann man mit "git commit" den momentanen Status des Repositories als neue Version speichern und mit "git push" an ein "remote" Repository senden. Als letzter wichtiger Befehl gilt noch "git branch", hiermit kann man das Repository in verschiedene "Branches" aufteilen und so neue Features getrennt voneinander entwickeln, und diese dann mit "git merge" im Hauptbranch zusammenfügen. [2]

3.2.2 Github



Github ist ein Cloud-basiertes Repository Hosting Service, das die verteilte Versionskontrolle von Git zur Verfügung stellt. Es wurde 2008 von Chris Wanstrath, P. J. Hyett, Scott Chacon und Tom Preston-Werner gestartet. Zu dem Zeitpunkt war Git noch relativ unbekannt, weshalb es noch keine anderen Optionen gab. Die Software wurde in der Programmiersprache "Ruby on Rails and Erlang" entwickelt. [3] Das Ziel von Github ist es, eine benutzerfreundliche Oberfläche für Git zur Verfügung zu stellen, mit der man auch mit weniger technischem Wissen die Vorteile von Git ausnutzen kann. [4] Als Unternehmen verdient GitHub Geld, indem es gehostete private Code-Repositories sowie andere geschäftsorientierte Pläne verkauft, die es Unternehmen erleichtern, Teammitglieder und Sicherheit zu verwalten. [4]

Github Issues

Github verfügt über einen integrierten issue tracker, mit dem sich Issues auf GitHub verwalten lassen. Ein Issue ist eine Aufgabe im Projekt, die mit dem Titel kurz, und dann in der Beschreibung genauer beschrieben wird, und einem bestimmten Entwickler zugeteilt wird. Ein Issue kann den Status "open" oder "closed" haben, "open" bedeutet, dass die Aufgabe noch nicht erfüllt wurde, und "closed" bedeutet dass die Aufgabe schon erfüllt ist. Github Issues haben gegenüber von anderen issue trackern vor allem einem Vorteil, weil es direkt dort ist, wo auch der Quellcode zu finden ist, jedoch ist es weit nicht so mächtig wie manch andere Optionen, weshalb sich bei dieser Arbeit nicht für Github Issues entschieden wurde. [5]

Github Actions

GitHub Actions ist ein in Github integriertes Tool, um Prozesse in einem Softwareprojekt zu automatisieren. Dadurch kann man Workflows für dein Repository erstellen. Ein Workflow besteht aus einem oder mehreren Jobs, wobei ein Job aus einem oder mehreren Schritten besteht. Man kann festlegen, ob Workflows in einem Container oder in einer virtuellen Maschine ausgeführt werden sollen. Die Ausführung kann unter den gängigen Betriebssystemen Windows, Linux und macOS erfolgen. Workflows werden durch Events wie beispielsweise ein Push auf das Repository ausgelöst und ausgeführt. Wenn ein Workflow ausgeführt wird, arbeitet er alle seine Jobs, sowie Schritte ab und erstellt dir ein

umfangreiches Feedback mit Logs und Ausführungszeiten. Das Feedback kann individuell für jeden Schritt angepasst werden. Eine Action wird in der Web-Oberfläche von GitHub erstellt. Praktischerweise kann eine erstellte Action geteilt und wiederverwendet werden. [6]

Github Packages

Ein Package ist ein wiederverwendbares Stück Software, das von einer globalen Registry in die lokale Umgebung eines Entwicklers heruntergeladen und in den Anwendungscode integriert werden kann. Da Pakete als wiederverwendbare "Bausteine" fungieren und in der Regel allgemeine Anforderungen erfüllen (z. B. API-Fehlerbehandlung), können sie zur Verkürzung der Entwicklungszeit beitragen. Github Packages ist ein Package Managing Service, der die Veröffentlichung von Packages erleichtert und vollständig in GitHub integriert ist. Alles befindet sich an einem Ort, sodass man zum Suchen und Veröffentlichen von Paketen dieselben Such-, Browsing- und Verwaltungstools verwenden kann wie für Repositories. [7]

3.2.3 IntelliJ IDEA



IntelliJ IDEA ist eine intelligente, kontextsensitive IDE für Java und andere JVM-Sprachen wie Kotlin, Scala und Groovy, die sich für zahlreiche Anwendungsbereiche eignet. Auch bei der Entwicklung von Full-Stack-Webanwendungen hilft IntelliJ IDEA Ultimate mit leistungsstarken integrierten Tools, Unterstützung für JavaScript und verwandte Technologien sowie erweiterte Unterstützung für gängige Frameworks wie Spring, Spring Boot, Jakarta EE, Micronaut, Quarkus oder Helidon. Darüber hinaus kann IntelliJ IDEA mit kostenlosen Plugins von JetBrains erweitern, um die IDE mit anderen Programmiersprachen wie Go, Python, SQL, Ruby und PHP einzusetzen.

3.2.4 WebStorm



Obwohl es möglich wäre mit IntelliJ ein Angular Projekt zu entwickeln haben wir uns bei der Frontend Entwicklung für die IDE Webstorm entschieden. Diese ist genauso wie IntelliJ von JetBrains doch enthält mehr support für die Programmiersprachen

JavaScript und TypeScript, sowie einen Built-in Debugger für Client-Side JavaScript und Node.js

3.2.5 Quarkus



Quarkus ist ein Framework zur Erstellung von Java-Anwendungen mit dem Java speziell für Container optimiert wird. Es bietet eine effektive Plattform für Serverless-, Cloud- und Kubernetes Umgebungen. Der Hersteller Redhat wirbt auch mit schnellen Startzeiten und geringen Speicherplatzverbrauch. Dies erzielt Quarkus dadurch dass es den Code schon Buildvorgangs verarbeitet.

3.2.6 Maven

Maven

Maven ist ein Build Automation Tool welches hauptsächlich für Java verwendet wird und folgt dem Ansatz Konvention vor Konfiguration. TODO

3.2.7 Angular



Angular ist ein Client-Side JavaScript Framework zur Erstellung von Single-Page-Webapplications. Seine komponentenbasierte Architektur welche View und Logik trennt macht es für den Entwickler leicht Applikationen zu warten und testen. Dabei verwendet Angular TypeScript für den Code-Behind und HTML als Auszeichnungssprache. Die vielen gut integrierten Libraries decken Features wie Routing, Verwaltung von Formularen und Client-Server Kommunikation ab.

3.2.8 PostgreSQL



PostgreSQL ist eine objektrelationale Datenbank welche sich durch ihre Stabilität und freie Verfügbarkeit auszeichnet. Sie hält sich dabei sehr eng an den SQL-Standard. Es werden eine Vielzahl von Datentypen und Operatoren unterstützt und Entwickler können

auch eigene Datentypen definieren. Ein weiterer großer Vorteil von PostgreSQL ist auch dass es auf jeder Hardware und auf beinahe jedem Betriebssystem läuft.

3.2.9 Docker



Docker ist eine Software zur Containervirtualisierung . Die Container sind voneinander isoliert und haben ihre eigene Software, Libraries sowie Konfigurationsdateien. Kommunizieren können sie über vorher genau definierte Kanäle. Da alle Container auf dem selben OS-Kernel laufen brauchen sie weniger Ressourcen wie eine Virtuelle Maschine.

4 Umsetzung

Siehe tolle Daten in Tab. 1.

Listing 1: Some code

```
1  # Program to find the sum of all numbers stored in a list (the not-Pythonic-way)
2
3  # List of numbers
4  numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]
5
6  # variable to store the sum
7  sum = 0
8
9  # iterate over the list
10 for val in numbers:
11     sum = sum+val
12
13 print("The sum is", sum)
```

	Regular Customers	Random Customers
Age	20-40	>60
Education	university	high school

Tabelle 1: Ein paar tabellarische Daten

5 Zusammenfassung

Aufzählungen:

- Itemize Level 1
 - Itemize Level 2
 - Itemize Level 3 (vermeiden)
- 1. Enumerate Level 1
 - a. Enumerate Level 2
 - i. Enumerate Level 3 (vermeiden)

Desc Level 1

Desc Level 2 (vermeiden)

Desc Level 3 (vermeiden)

Literaturverzeichnis

- [1] , „Was ist Git?“ letzter Zugriff am 18.08.2022. Online verfügbar: <https://www.atlassian.com/de/git/tutorials/what-is-git>
- [2] —, „GIT FACHBEGRIFFE EINFACH ERKLÄRT,“ letzter Zugriff am 18.08.2022. Online verfügbar: <https://www.arocom.de/fachbegriffe/webentwicklung/git>
- [3] Saurabh Mhatre, „The untold story of Github,“ 2016, letzter Zugriff am 18.08.2022. Online verfügbar: <https://smhatre59.medium.com/the-untold-story-of-github-132840f72f56>
- [4] , „Was ist GitHub? Eine Einführung in GitHub für Einsteiger,“ 2020, letzter Zugriff am 18.08.2022. Online verfügbar: <https://kinsta.com/de/wissensdatenbank/was-ist-github/>
- [5] —, „About issues,“ letzter Zugriff am 19.08.2022. Online verfügbar: <https://docs.github.com/en/issues/tracking-your-work-with-issues/about-issues>
- [6] Cem Caylak, „GitHub Actions im Java Projekt,“ 2020, letzter Zugriff am 19.08.2022. Online verfügbar: <https://www.adesso.de/de/news/blog/github-actions-im-java-projekt.jsp#:~:text=GitHub%20Actions%20ist%20ein%20hauseigenes,einem%20oder%20mehreren%20Schritten%20besteht.>
- [7] Liz Parody, „GitHub Package Registry: Pros and Cons for the Node.js Ecosystem,“ 2019, letzter Zugriff am 19.08.2022. Online verfügbar: <https://nodesource.com/blog/github-package-registry/>

Abbildungsverzeichnis

1 system architecture 4

Tabellenverzeichnis

1	Ein paar tabellarische Daten	10
---	--	----

Quellcodeverzeichnis

1	Some code	10
---	---------------------	----

Anhang