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 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 Verschidenen Rollen und ihrer Rechte, etc. erläutert und gerechtfertigt.

- Kapitel 5: Zusammenfassung

  Hier werden wichtigste Ergebnisse des Projekts , mögliche Erweitungen (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 verschiedernen 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 Tunrniers 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 unabhänigig von der Sportart ist.

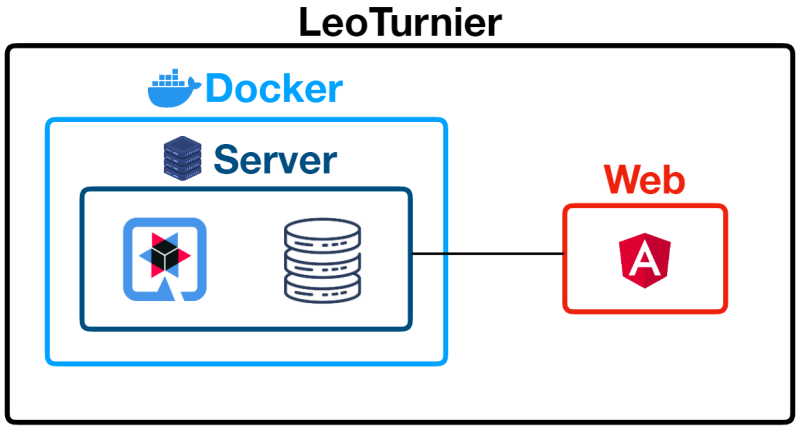
2.4 Ziele

Im Rahmen dieser Diplomarbeit soll nun eine Applikation erstellt werden die das Verwalten und die Informationsbesschafung 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 Systemarchitechtur

3.1 Verwirklichung der Anforderungen

Abbildung 1: system architecture



TODO

4 Technologien

4.1 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 Betriebsystem-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]

4.1.1 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 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]

4.1.2 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, dass heißt, ob das File neu im Repository ist, ob es seit dem letzten Mal Speichern verändert wurde, oder ob es erst garnicht 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]

4.2 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, wesshalb 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]

4.2.1 Github Issues

Github verfügt über einen integrierten issue tracken, 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 vorallem 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, wesshalb sich bei dieser Arbeit nicht für Github Issues entschieden wurde. [5]

4.2.2 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]

4.2.3 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]

4.3 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 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 Plugins von JetBrains erweitern, um die IDE mit anderen Programmiersprachen wie Go, Python, SQL, Ruby und PHP einzusetzen. [8]

4.3.1 Was ist eine IDE?

Eine IDE (Integrated Development Environment) oder integrierte Entwicklungsumgebung ist Software für die Anwendungsentwicklung, die gängige Entwicklertools in einer zentralen grafischen Oberfläche vereint. Eine typische IDE besteht aus folgenden Komponenten: [9]

Quellcode-Editor

Ein Texteditor, der eine Programmierung von Software-Code mit folgenden Features unterstützt: Syntaxhervorhebung mit visuellen Hinweisen, sprachspezifische Autovervollständigung und eine Bug-Prüfung, während der Code geschrieben wird. [9]

Automatisierung lokaler Builds

Dienstprogramme, mit denen sich einfache wiederholbare Aufgaben im Rahmen der Entwicklung lokaler Software-Builds zur Nutzung durch die Entwickler automatisieren lassen, wie beispielsweise die Kompilierung von Quell- in Binärcode, dessen Paketierung und die Ausführung automatischer Tests. [9]

Debugger

Ein Programm zur Prüfung anderer Programme, mit dem sich die Position von Bugs im Originalcode grafisch anzeigen lässt. [9]

4.3.2 Codeanalyse

Intellij bietet intelligente Programmierhilfen an. Durch eine Indizierung des Quellcodes legt die IDE eine virtuelle Karte des Projekts an. Anhand der Informationen in dieser virtuellen Karte kann sie Fehler erkennen, kontextabhängige Completion-Vorschläge anbieten oder Refactorings durchführen. [8]

4.3.3 Versionsverwaltung

IntelliJ IDEA unterstützt die gängigen Versionsverwaltungen (VCS) wie Git, Subversion, Mercurial und Perforce. Man kann ein VCS-Projekt direkt auf dem Startbildschirm klonen, die Unterschiede zwischen zwei Revisionen untersuchen, Branches verwalten, Commits durchführen und pushen, Konflikte bereinigen oder den Verlauf überprüfen. [8]

4.3.4 JVM-Frameworks

IntelliJ IDEA Ultimate bietet Unterstützung für Frameworks und Technologien zur Entwicklung von Anwendungen und Microservices. Die IDE bietet spezielle Unterstützung unter anderem für Quarkus, Spring, Spring Boot, Jakarta EE, JPA und Reactor. [8]

4.4 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

4.5 Java

Java ist nicht nur eine Programmiersprache, sondern ebenso ein Laufzeitsystem, was Oracle durch den Begriff Java Platform verdeutlichen will. So gibt es neben der Programmiersprache Java durchaus andere Sprachen, die eine Java-Laufzeitumgebung ausführen, etwa diverse Skriptsprachen wie Groovy, JRuby, Jython, Kotlin oder Scala. Skriptsprachen auf der Java-Plattform werden immer populärer; sie etablieren eine andere Syntax, nutzen aber die JVM und die Bibliotheken. Zu der Programmiersprache und JVM kommt ein Satz von Standardbibliotheken für Datenstrukturen, Zeichenkettenverarbeitung, Datum/Zeit-Verarbeitung, grafische Oberflächen, Ein-/Ausgabe, Netzwerkoperationen und mehr. Das bildet die Basis für höherwertige Dienste wie Datenbankanbindungen oder Webservices. Integraler Bestandteil der Standardbibliothek seit Java 1.0 sind weiterhin Threads. Sie sind leicht zu erzeugende Ausführungsstränge, die unabhängig voneinander arbeiten können. Mittlerweile unterstützen alle populären Betriebssysteme diese »leichtgewichtigen Prozesse« von Haus aus, sodass die JVM diese parallelen Programmteile nicht nachbilden muss, sondern auf das Betriebssystem verweisen kann. Auf den neuen Multi-Core-Prozessoren sorgt das Betriebssystem für eine optimale Ausnutzung der Rechenleistung, da Threads wirklich nebenläufig arbeiten können. [10]

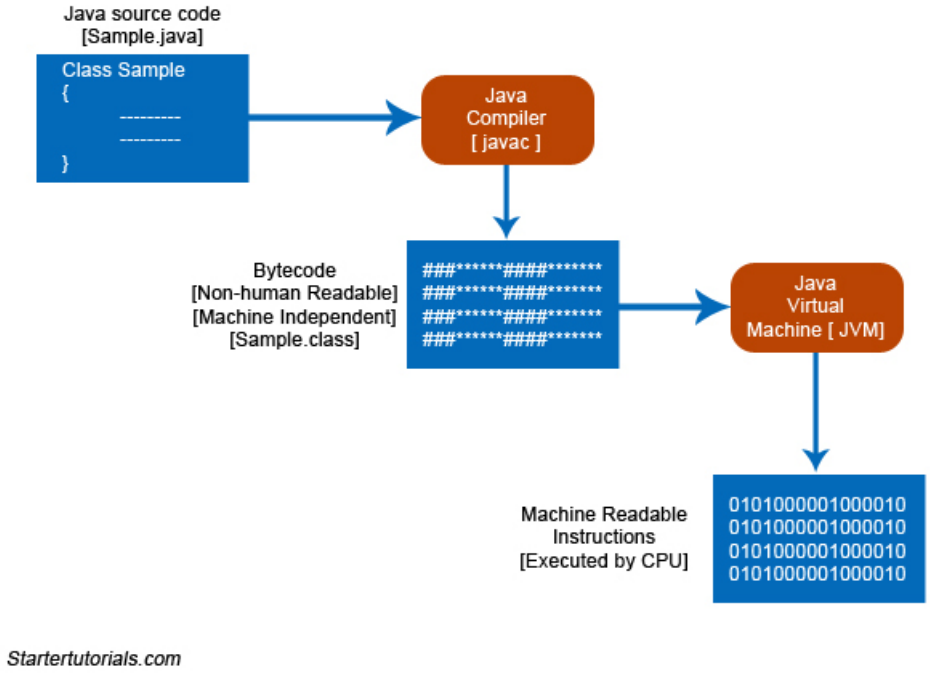
4.5.1 Java Life Cycle

Der Life Cycle (Lebenszyklus) eines Java Programms gibt an, was genau mit dem Programm passiert, wie es vom Quellcode, den man in der IDE eingibt, zum Maschinenode wird, der aus 0 und 1 besteht. Es gibt 3 Hauptphasen im Lebenszyklus:

- Bearbeitung des Programms
- Kompilierung des Quellcodes
- Ausführung des Byte Codes

Zuerst wird der Code in die IDE bzw. den Editor eingegeben, wenn man damit fertig ist, wird der Code in einem oder mehreren Files gespeichert. Diese Files haben die Endung ".java". Als nächstes wird das Programm kompiliert. Der Name des Java Compilers ist "javac". Der Input an den Compiler ist der Java Quellcode aus den ".java" Files, der Output des Compilers ist Platform-unabhängiger Code namens "Byte Code". Das

File, das vom Compiler generiert wurde, hat die Endung ".class". Als letztes wird das Programm ausgeführt. Der Bytecode, der vom Compiler generiert wurde, wird von der "Java Virtual Mashine" oder JVM ausgeführt, der input an die JVM ist der Bytecode aus dem ".class" File, der Output ist Maschinenode, bestehend aus 0 und 1, und wird von der CPU ausgeführt. [11]



4.6 Quarkus

Quarkus ist ein Kubernetes-natives Full-Stack Java Framework für JVMs (Java Virtual Machines) und native Kompilierung, mit dem Java speziell für Container optimiert wird. Es bietet eine effektive Plattform für Serverless-, Cloud- und Kubernetes-Umgebungen. Quarkus wurde speziell für beliebte Java-Standards, Frameworks und Libraries wie Eclipse MicroProfile und Spring sowie für Apache Kafka, RESTEasy (JAX-RS), Hibernate ORM (JPA), Spring, Infinispan, Camel und viele mehr konzipiert. Die Dependency-Injection-Lösung von Quarkus basiert auf CDI (Contexts and Dependency Injection) und bietet ein Framework, mit dessen Hilfe die Funktionalität erweitert und ein Framework konfiguriert, gebootet und in Ihre Anwendung integriert werden kann. Quarkus wurde als benutzerfreundliches Tool konzipiert und integriert Funktionen, die keine oder nur wenig Konfiguration erfordern. Entwickler können die passenden Java Frameworks

für ihre Anwendungen auswählen, die dann im JVM-Modus laufen oder kompiliert und im nativen Modus ausgeführt werden können. Das ganz auf Entwickler abgestimmte Quarkus integriert außerdem folgende Funktionen:

- Live-Codierung zur umgehenden Prüfung der Auswirkungen von Code-Änderungen sowie eine schnelle Problembehebung
- Einheitliche imperative und reaktive Programmierung mit einem eingebetteten gemanagten Event Bus
- Einheitliche Konfiguration
- Einfaches Generieren nativer Programmdateien

Ob man die Anwendungen auf einer Public Cloud oder in einem intern gehosteten Kubernetes-Cluster hosten, Eigenschaften wie ein schneller Start sowie ein niedriger Speicherverbrauch sind wichtig, um die Gesamtkosten für das Hosting niedrig zu halten. Quarkus wurde auf Basis einer Container-first-Philosophie entwickelt und soll so einen niedrigen Speicherverbrauch und schnelle Startzeiten auf folgende Weise ermöglichen:

- Erstklassiger Support für Graal/SubstrateVM
- Metadaten-Verarbeitung gleichzeitig mit dem Build
- Reduzierung der Reflektionsnutzung
- Preboot nativer Images

Bei der Anwendungsentwicklung mit Quarkus wird im Vergleich zum traditionellen Java nur ein Zehntel des Speichers belegt. Dazu bietet es bis zu 300 Mal schnellere Startzeiten. Beide Aspekte ermöglichen eine deutliche Reduzierung der Kosten für Cloud-Ressourcen. Quarkus ist so konzipiert, dass bei der Anwendungsentwicklung der bewährte imperative Code problemlos mit dem nicht-blockierenden reaktiven Code kombiniert werden kann. Dies erweist sich als nützlich sowohl für Java-Entwickler, die an das imperative Modell gewöhnt sind und auch dabei bleiben möchten, und all diejenigen, die einen cloudnativen/reaktiven Ansatz bevorzugen. Das Quarkus-Entwicklungsmodell lässt sich flexibel an alle zu erstellenden Anwendungstypen anpassen. Quarkus ist eine effiziente Lösung zur Ausführung von Java in dieser neuen Welt von Serverless-Architekturen, Microservices, Containern, Kubernetes, Function-as-a-Service (FaaS) und Clouds, weil es für all diese Technologien entwickelt wurde. [12]

4.6.1 RESTEasy JAX-RS

JAX-RS ist eine JCP-Spezifikation, die eine Java-API für RESTful Web Services über das HTTP-Protokoll bereitstellt. Resteasy ist eine portable Implementierung dieser Spezifikation, die in jedem Servlet-Container laufen kann. Eine engere Integration mit JBoss Application Server ist ebenfalls verfügbar, um die Benutzererfahrung in dieser Umgebung zu verbessern. Während JAX-RS nur eine serverseitige Spezifikation ist, hat Resteasy JAX-RS durch das RESTEasy JAX-RS Client Framework innovativ auf den Client übertragen. Dieses clientseitige Framework ermöglicht es, ausgehende HTTP-Anfragen mit Hilfe von JAX-RS-Annotationen und Schnittstellen-Proxys auf Remote-Server abzubilden. [13]

4.6.2 Hibernate ORM

Bei Hibernate handelt es sich um ein Framework zur Abbildung von Objekten auf relationalen Datenbanken für die Programmiersprache Java - es wird auch als Object Relational Mapping Tool bezeichnet. Hibernate ist in der Programmiersprache Java implementiert und steht als Open Source zur freien Verfügung. Hibernate nutzt das Konzept der Reflexion, um so zur Laufzeit einer Software sogenannte Meta- Informationen über die Struktur von Objekten und die zugrunde liegenden Klassen zu ermitteln, die dann auf die Tabellen einer Datenbank abgebildet werden. [14]

4.7 Apache Maven

Maven

Maven ist ein Build-Tool der Apache Software Foundation für die Projektverwaltung. Entwickler können damit Java-Projekte automatisieren. Maven ist ein Teil der Apache Software Foundation und wird von dieser gehostet. Das Tool ist aus einem Teil des Jakarta-Projekts hervorgegangen. Maven vereinfacht den Softwareerstellungsprozess, bietet ein einheitliches System für die Entwicklerarbeit, hochwertige Projektinformationen, Richtlinien für das Festlegen von Best Practices und erleichtert die Migration zu neuen Funktionen durch mehr Transparenz. Maven beschreibt sowohl, wie Software gebaut wird, als auch deren Abhängigkeiten. Apache hilft sowohl bei der Verwaltung von Projekten und dient als Tool zum Verständnis. Es hilft also dabei, den Zustand eines Projekts sehr schnell darzustellen. Das Programm wird von Entwicklern und Projekt-

managern von Java-Anwendungsprojekten verwendet. Das Tool kann nützlich sein, um den aktuellen Zustand eines Projektes auf einem Blick für technisch weniger versierte Personen, wie Führungspersönlichkeiten und Investoren, zusammenzufassen. Maven basiert auf dem Objektmodell. Projekte werden als eine Pom.xml-Datei gespeichert. Das Tool verwaltet Projekt-Builds, Reporting und Dokumentation aus der zentralen XML-Informationsbasis. Mit seiner Standard-Plug-in-Architektur kann Maven über Standard-Input mit so gut wie jeder Anwendung zusammenarbeiten. Maven vereinfacht den Prozess für das Erstellen von Java-Anwendungen erheblich, da Nutzer den Status des Projektes viel einfacher einschätzen können. Der Name Maven kommt aus dem Jiddischen und bedeutet Akkumulator von Wissen. Obwohl Maven das Entwickeln von Software anhand von Konventionen ermöglicht, sind reproduzierbare Builds derzeit kein Feature. Die Entwickler arbeiten aber daran. [15]

4.8 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 Formulare und Client-Server Kommunikation ab.

4.9 PostgreSQL

Dieses DBMS (Datenbank-Managementsystem) bietet eine liberale Lizenz, flexiblen Umgang mit Datentypen, ist skalierbar und vielseitig nutzbar. Der Definition nach ist PostgreSQL ein „objektrelationales“ Datenbank-Managementsystem (DBMS). Es ist Open Source und orientiert sich sehr eng am SQL-Standard. Bestehende SQL-Anwendungen können durch den ANSI-SQL-Standards ohne großen Aufwand integriert werden. Die Funktionsweise entspricht dem klassischen Client-Server-Prinzip, bei dem unterschiedliche Clients Anfragen an den Datenbankserver senden. In der Praxis ist

es mit PostgreSQL möglich, komplexere Anwendungen und Workloads zu entwickeln und zu betreiben als mit anderen Lösungen. Es unterstützt eine Vielzahl Datentypen und Operatoren. Entwickler können hier bei Bedarf eigene Datentypen definieren. PostgreSQL ist nicht nur für Webanwendungen sehr gut geeignet, es kann auch große Enterprise-Datenbanken abbilden. PostgreSQL findet auch in Desktop-Systemen Einsatz: Es ist das Datenbanksystem von Linux und macOS. Diese Systeme liefern das Datenbanksystem standardmäßig mit. [16]

4.9.1 Vorteile

Offene Lizenz

Einer der größten Vorteile von PostgreSQL ist die offene Lizenz. Es ist unter der eigenen PostgreSQL-Lizenz veröffentlicht. Generell fallen dabei keine Lizenzkosten an. Diese liberale Lizenz erlaubt es, das System beliebig zu vertreiben. Änderungen müssen nicht der Community zur Verfügung gestellt werden. Die Verwendung von PostgreSQL hat keine nachgelagerten Auswirkungen darauf, wie Weiterentwicklungen und darauf basierende Lösungen verwendet werden dürfen oder veröffentlicht werden müssen. [16]

Großes Potenzial

Die Datenbankgröße ist nicht durch das System begrenzt, sondern einzig durch den zur Verfügung stehenden Speicher. Da Postgres flexibel und kompatibel aufgebaut ist, können eine Vielzahl an Anwendungen auf diese Datenbanklösung bauen. Dazu bietet PostgreSQL die Möglichkeit, Trigger einzusetzen und bietet Schnittstellen zu vielen verschiedenen Programmiersprachen, was den Einsatzzweck und die Zusammenarbeit mit unterschiedlichsten Anwendungen potenziell vielfältig macht. [16]

Sichere Replikation

Ein weiterer wichtiger Punkt ist die detaillierte Zugriffskontrolle und sichere Datenreplikation. In der Praxis ist je nach Anwendungsfall asynchrone oder synchrone Replikation möglich: Synchrone Replikation, um sicherzustellen, dass kritische Transaktionen auf beiden Servern ausgeführt wurden. Asynchrone Replikation für maximale Geschwindigkeit bei weniger kritischen Transaktionen. Ein weiterer Vorteil ist die freie Wahl

der Hardware und des Server-OS. PostgreSQL kann auf diversen Linux-Distributionen, BSD, Windows oder macOS gehostet werden. [16]

4.10 Docker

Die Docker-Technologie verwendet den Linux Kernel und seine Funktionen wie Cgroups und namespaces, um Prozesse zu isolieren, damit diese unabhängig voneinander ausgeführt werden können. Diese Unabhängigkeit ist der Zweck der Container – die Fähigkeit, mehrere Prozesse und Apps getrennt voneinander betreiben zu können. So wird die Infrastruktur besser genutzt und gleichzeitig die Sicherheit bewahrt, die sich aus der Arbeit mit getrennten Systemen ergibt. Containertools, einschließlich Docker, arbeiten mit einem Image-basierten Bereitstellungsmodell. Das macht es einfacher, eine Anwendung oder ein Paket von Services mit all deren Abhängigkeiten in mehreren Umgebungen gemeinsam zu nutzen. Docker automatisiert außerdem die Bereitstellung der Anwendung (oder Kombinationen von Prozessen, die eine Anwendung darstellen) innerhalb dieser Container-Umgebung. Diese Tools bauen auf Linux-Containern auf und geben den Nutzern so nie dagewesenen Zugriff auf Anwendungen. Sie ermöglichen eine deutlich schnellere Bereitstellung und Kontrolle von Versionen sowie deren Verbreitung. [17]

4.10.1 Vorteile

Modularität

Der Docker-Ansatz für die Containerisierung konzentriert sich darauf, nur einen Teil der Anwendung für eine Reparatur oder Aktualisierung außer Betrieb zu nehmen, ohne dass die gesamte Anwendung außer Betrieb genommen werden muss. Zusätzlich zu diesem auf Microservices basierenden Ansatz können Sie Prozesse in mehreren Apps gemeinsam verwenden – so ähnlich wie bei einer serviceorientierten Architektur (SOA). [17]

Layer und Image-Versionskontrolle

Jedes Docker-Image besteht aus einer Reihe von Layern. Diese Layer werden in einem einzelnen Image vereint. Wenn sich das Image ändert, wird ein Layer erstellt. Jedes

Mal, wenn ein Nutzer einen Befehl wie run oder copy eingibt, wird ein neuer Layer erstellt. Mit Docker werden diese Layer für neuer Container wiederverwendet, was die Entwicklung enorm beschleunigt. Zwischenzeitliche Änderungen werden von den Images gemeinsam verwendet, was die Geschwindigkeit, Größe und Effizienz weiter verbessert. Versionskontrolle ist ein fester Bestandteil des Layering. Bei jeder neuen hat man im Prinzip ein eingebautes Änderungsprotokoll und damit volle Kontrolle über die Container-Images. [17]

Rollback

Das Beste am Layering ist wahrscheinlich das Rollback, also das Zurücksetzen auf die vorherige Version. Jedes Image hat Layers. Wenn man mit der aktuellen Iteration eines Image nicht zufrieden ist, kann man einfach auf die vorherige Version zurücksetzen. Dieser Ansatz unterstützt eine agile Entwicklung und sorgt, was die Tools angeht, für eine kontinuierliche Integration und Bereitstellung (Continuous Integration/Continuous Deployment - CI/CD). [17]

Schnelle Bereitstellung

Neue Hardware bereitzustellen und zum Laufen zu bringen, dauerte normalerweise Tage und war mit einem enormen Aufwand verbunden. Mit Docker-basierten Containern kann die Bereitstellung auf Sekunden reduziert werden. Indem man für jeden Prozess einen Container erstellt, kann man ähnliche Prozesse schnell mit neuen Apps teilen. Und da zum Hinzufügen oder Verschieben eines Containers das Betriebssystem nicht gebootet werden muss, sind die Bereitstellungszeiten wesentlich kürzer. Darüber hinaus kann man bei dieser Entwicklungsgeschwindigkeit einfach und kosteneffizient Daten erstellen und die von den Containern erzeugten Daten ohne Bedenken löschen. Die Docker-Technologie ist also ein granularer, kontrollierbarer, auf Microservices basierter Ansatz, der deutlich effizienter ist. [17]

4.10.2 Einschränkungen

Docker für sich allein ist für die Verwaltung einzelner Container bestens geeignet. Wenn man beginnt, mehr und mehr Container und containerisierte Apps zu verwenden, die in Hunderte von Bestandteilen zerlegt sind, können die Verwaltung und Orchestrierung sehr schwierig werden. Irgendwann muss man einen Schritt zurückgehen und Container

gruppieren, um Dienste wie Vernetzung, Sicherheit, Telemetrie usw. in den Containern bereitzustellen.

5 Umsetzung

Siehe tolle Daten in Tab. 1.

Listing 1: Some code

```
# Program to find the sum of all numbers stored in a list (the not-Pythonic-way)

# List of numbers
numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]

# variable to store the sum
sum = 0

# iterate over the list
for val in numbers:
    sum = sum+val

print("The sum is", sum)
```

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

Tabelle 1: Ein paar tabellarische Daten

6 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/

[8] , „Was ist IntelliJ IDEA?" letzter Zugriff am 20.08.2022. Online verfügbar: https://www.jetbrains.com/de-de/idea/features/

[9] ——, „Was ist eine IDE?" 2019, letzter Zugriff am 20.08.2022. Online verfügbar: https://www.redhat.com/de/topics/middleware/what-is-ide

[10] C. Ullenboom, *Java ist auch eine Insel*, 15. Aufl. Galileo Computing, 2020.

[11] , „Life cycle of a java program," 2014, letzter Zugriff am 20.08.2022. Online verfügbar: https://www.startertutorials.com/corejava/life-cycle-java-program.html#:~:text=There%20are%20three%20main%20stages,Compiling%20the%20source%20code

[12] ——, „Was ist Quarkus?" 2020, letzter Zugriff am 20.08.2022. Online verfügbar: https://www.redhat.com/de/topics/cloud-native-apps/what-is-quarkus

[13] ——, „RESTEasy JAX-RS," letzter Zugriff am 20.08.2022. Online verfügbar: https://docs.jboss.org/resteasy/docs/3.0.6.Final/userguide/html_single/index.html

[14] Redaktion ComputerWeekly.de, „Hibernate," 2020, letzter Zugriff am 20.08.2022. Online verfügbar: https://www.itwissen.info/Hibernate-hibernate.html

[15] , „Maven," letzter Zugriff am 20.08.2022. Online verfügbar: https://www.computerweekly.com/de/definition/Maven

[16] Patrick Woods, „Was ist PostgreSQL?" 2021, letzter Zugriff am 20.08.2022. Online verfügbar: https://www.plusserver.com/blog/was-ist-postgresql

[17] , „Docker – Funktionsweise, Vorteile, Einschränkungen," 2018, letzter Zugriff am 20.08.2022. Online verfügbar: https://www.redhat.com/de/topics/containers/what-is-docker?sc_cid=7013a000002wLw0AAE&gclid=Cj0KCQjwjIKYBhC6ARIsAGEds-KwmuTopp7YzOBl8IVrXiFvqwtPWlxMCQlrUTA9mDJokqoQZRqMcNcaAv5LEALw_wcB&gclsrc=aw.ds

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Anhang