Westfälische Hochschule

University of Applied Sciences
Gelsenkirchen Bocholt Recklinghausen

Bachelorarbeit

Titel der Arbeit // Title of Thesis

Konzeption und Entwicklung einer Continuous-Integration-Strategie für Kundenprojekte auf Basis der Shopware-Platform

---

Akademischer Abschlussgrad: Grad, Fachrichtung (Abkürzung) // Degree

Bachelor of Science (B.Sc.)

Autorenname, Geburtsort // Name, Place of Birth

Frederik Bußmann, Coesfeld

Studiengang // Course of Study

Informatik.Softwaresysteme

Fachbereich // Department

Wirtschaft und Informationstechnik

Erstprüferin/Erstprüfer // First Examiner

Prof. Dr.-Ing. Martin Schulten

Zweitprüferin/Zweitprüfer // Second Examiner

Martin Knoop

Abgabedatum // Date of Submission

xx.xx.2023

Eidesstattliche Versicherung

Bußmann, Frederik

Name, Vorname // Name, First Name

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem Titel

Konzeption und Entwicklung einer Continuous-Integration-Strategie für Kundenprojekte auf Basis der Shopware-Platform

selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Stadtlohn, den

Ort, Datum, Unterschrift // Place, Date, Signature

Abstract

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Unternehmensporträt der best it GmbH 1
1.2 Zielsetzung 1
1.3 Struktur der Arbeit 2

2 Fachlicher Hintergrund **3**
2.1 Continuous Software Engineering 3
2.2 Begrifflichkeiten und Prinzipien von Continuous Integration 4
2.3 Übersicht über die Shopware-Platform 6

3 Analyse und Konzept **9**

4 Entwicklung der CI-Strategie **10**

5 Evaluierung **11**

6 Schlussfolgerungen und Ausblick **12**

Literaturverzeichnis **13**

Abkürzungsverzeichnis

API . . . . . Application Programming Interface

CD . . . . . Continuous Deployment

CDE . . . . . Continuous Delivery

CI . . . . . Continuous Integration

CMS . . . . . Content Management System

DI . . . . . Dependency Injection

ORM . . . . . Object-Relational Mapping

VCS . . . . . Version Control System

Abbildungsverzeichnis

1 Zusammenhang zwischen CI, CDE und CD 4

1 Einleitung

Im Rahmen dieser Arbeit werden verschiedene Aspekte betrachtet, um ein Konzept für das Einbinden von Continuous Integration (CI) in Shopware-basierten Projekten zu erarbeiten. Shopware ist eine E-Commerce-Plattform und eine der meistgenutzten Online-Shop-Lösungen in Deutschland[1], welche es Unternehmen ermöglicht, ihre Präsenz im digitalen Markt auszubauen. Das Unternehmen best it GmbH hat für seine Kunden einige Shopware-Projekte im Einsatz, für welche im Nachfolgenden eine CI-Strategie zur automatisierten Prüfung und Auslieferung der entwickelten Software erstellt werden soll.

1.1 Unternehmensporträt der best it GmbH

Die best it GmbH ist eine Digitalagentur, die im Jahr 2000 von Manuel Strotmann gegründet wurde und sich auf die Entwicklung von E-Commerce-Lösungen spezialisiert hat. Die Firma ist in verschiedenen Bereichen des Online-Handels tätig und realisiert für ihre Kunden eine Vielzahl von Shopware-Projekten. Neben der Konzeption und Entwicklung von Online-Shops bietet das Unternehmen auch Design- und Beratungsdienstleistungen für Kunden im E-Commerce-Bereich an. Mittlerweile beschäftigt die best it GmbH über 100 Mitarbeiter an drei Standorten in Deutschland und Österreich.

1.2 Zielsetzung

Das Ziel dieser Arbeit ist die Entwicklung einer Continuous-Integration-Strategie für auf Shopware basierende Kundenprojekte der Firma best it. Um eine Konzeption zu erstellen, die den Anforderungen des Unternehmens entspricht, werden die Ziele der Strategie nachfolgend definiert:

- **Höhere Entwicklungsgeschwindigkeit:** Die Einführung einer umfangreichen CI-Strategie soll die Effizienz der Softwareentwicklungsteams verbessern und die Zeit bis zum Software-Release senken.
- **Frühe Fehlererkennung:** CI soll dazu beitragen, Fehler frühzeitig im Entwicklungsprozess zu erkennen und zu beheben, was die Qualität des Endprodukts verbessert.
- **Bessere Kommunikation:** Die CI-Strategie und die damit verbundenen Prozesse, die sich für Entwicklerteams ergeben, sollen zu einer verbesserten Kollaboration führen.

Bei der Konzeption sollen diese Ziele verfolgt und die Maßnahmen der zu erarbeiteten CI-Strategie dementsprechend ausgerichtet werden. Die Strategie soll dabei nicht nur die technischen Aspekte von Continuous Integration berücksichtigen, sondern auch die organisatorischen und kulturellen Veränderungen, die mit der Einführung von CI einhergehen. Darüber hinaus soll die Strategie flexibel genug sein, um sich an zukünftige Veränderungen und Entwicklungen anpassen zu können.

[1] Vgl. eCommerceDB. *The Most Commonly Used Shop Software Among Online Shops in Germany - Shopware and Salesforce Share Rank No. 1.* 2023.

1.3 Struktur der Arbeit

Im Laufe dieser Arbeit wird eine CI-Strategie konzeptioniert und entwickelt. Die Arbeit ist in fünf Hauptabschnitte unterteilt, die jeweils unterschiedliche Aspekte des Prozesses abdecken.

Fachlicher Hintergrund

Zunächst wird der fachliche Hintergrund für die Arbeit festgelegt, der Abschnitt umfasst eine Einführung in das Continuous-Software-Engineering und die Prinzipien und Praktiken von Continuous Integration, sowie eine Übersicht über die Shopware-Plattform. Dieser Abschnitt dient dazu, ein grundlegendes Verständnis für die Themen und Technologien zu schaffen, die in der Arbeit behandelt werden.

Analyse und Konzept

Dieser Abschnitt befasst sich mit der Analyse der aktuellen Situation und der Entwicklung eines Konzepts für die CI-Strategie. Dies beinhaltet die Identifizierung von Herausforderungen und Anforderungen, die Berücksichtigung von Best Practices und die Ausarbeitung eines Plans für die Implementierung der Strategie. Die Konzeptionierung stützt sich dabei auf die im vorherigen Abschnitt aufgezeigte Fachliteratur.

Entwicklung der CI-Strategie

In diesem Abschnitt wird die Entwicklung der CI-Strategie beschrieben. Dies umfasst die Auswahl und Konfiguration der benötigten Tools, die Definition von Prozessen und Workflows und die Implementierung von Automatisierungen und Tests. Der Fokus liegt dabei auf der praktischen Anwendbarkeit des im vorherigen Abschnitt entwickelten Konzepts.

Evaluierung

Die Auswertung der implementierten CI-Strategie wird im folgenden Abschnitt behandelt. Dies beinhaltet die Durchführung von Tests und Messungen, um die Wirksamkeit und Effizienz der Strategie zu bewerten. Dabei wird die umgesetzte Strategie im Hinblick auf die im fachlichen Hintergrund aufgezeigten Prinzipien geprüft. Die Ergebnisse dieser Evaluierung werden analysiert und interpretiert, um Rückschlüsse auf den Erfolg der Implementierung zu ziehen.

Schlussfolgerung und Ausblick

Der letzte Abschnitt fasst die Ergebnisse der Arbeit zusammen und es werden Schlussfolgerungen über die CI-Strategie und dessen Nutzung im Unternehmen gezogen. Darüber hinaus wird ein Ausblick auf mögliche zukünftige Entwicklungen und Verbesserungen gegeben. Dieser Abschnitt dient dazu, die Arbeit abzurunden und einen Ausblick auf weitere Forschungs- und Entwicklungsarbeiten in diesem Bereich zu geben.

2 Fachlicher Hintergrund

Für die Erarbeitung einer geeigneten CI-Strategie wird zunächst ein Einblick in die Disziplin des Continuous-Software-Engineering gegeben. Anschließend werden die Begrifflichkeiten und Prinzipien von Continuous Integration definiert. Darüber hinaus wird eine Übersicht über die Funktionen und Mechanismen der Shopware-Plattform gegeben.

2.1 Continuous Software Engineering

Continuous-Software-Engineering fasst die Prinzipien der Continuous Integration (CI), Continuous Delivery (CDE), und Continuous Deployment (CD) zusammen. Shahin et al. definieren den Begriff als einen Bereich der Softwareentwicklung, bei dem es um die Entwicklung, Auslieferung und das schnelle Feedback von Software und Kunde geht. Die Disziplin umfasst Geschäftsstrategie und Planung, sowie Entwicklung und den Betrieb der Software.[2] Diese kontinuierliche Integrierung von Software ist sehr kompatibel mit den häufigen iterationen in der agilen Softwareentwicklung und wurde unter anderem durch die agile Methodik des Extreme Programming bekannt.[3] Nachfolgend werden die Bereiche der agilen Softwareentwicklung und der CI, CDE und CD kurz erläutert.

Agile Software Development

Agile Softwareentwicklung ist ein Ansatz zur Softwareentwicklung, der auf Flexibilität und Kundeninteraktion setzt. Im Gegensatz zu traditionellen, plangetriebenen Methoden, die die Anforderungen und Lösungen am Anfang des Projekts festlegen, erlaubt die agile Methodik Änderungen und Anpassungen während des gesamten Entwicklungsprozesses. Dies wird durch iterative Entwicklung und regelmäßiges Feedback erreicht. Zu den wichtigsten Prinzipien der agilen Softwareentwicklung gehören die kontinuierliche Auslieferung von Software, Offenheit für sich ändernde Anforderungen und enge Zusammenarbeit zwischen Teams und Entwicklern.[4] Gren und Lenberg fassen Agile als die Reaktionsfähigkeit im Hinblick auf sich ständig ändernde Anforderungen und Umgebungen zusammen.[5]

Continuous Integration

Continuous Integration (CI) ist ein Softwareentwicklungsprozess, bei dem Entwickler ihre Änderungen regelmäßig, oft mehrmals täglich, in ein gemeinsames Repository integrieren. Jede dieser Integrationen wird dann von einem automatisierten Build-System überprüft, um sicherzustellen, dass die Änderungen mit der bestehenden Codebase kompatibel sind und keine Fehler verursachen. Dieser Prozess ermöglicht es Teams, Probleme frühzeitig zu erkennen und zu beheben, was die Qualität der Software verbessert und die Zeit bis zur Auslieferung der Software reduziert.[6]

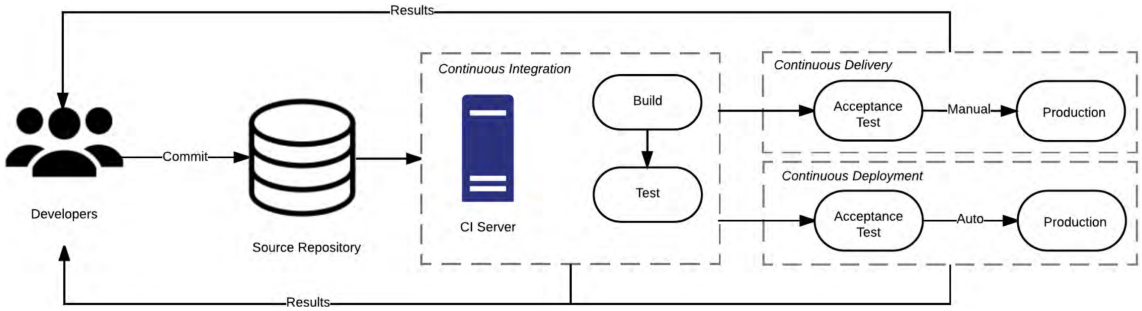
[2] Vgl. Shahin, Ali Babar und Zhu. „Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices“. 2017, S. 3910–3911.
[3] Vgl. Fitzgerald und Stol. „Continuous software engineering: A roadmap and agenda“. 2017.
[4] Vgl. Beck, Beedle, Bennekum et al. *Manifesto for Agile Software Development*. 2001.
[5] Vgl. Gren und Lenberg. „Agility is responsiveness to change“. 2020.
[6] Vgl. Fowler. *Continuous Integration*. 2006.

Continuous Delivery

Continuous Delivery (CDE) erweitert das Konzept der Continuous Integration, indem es sicherstellt, dass die Software stetig in einem Zustand ist, der sicher in die Produktionsumgebung ausgerollt werden kann. Dies wird durch das Einführen von Integrationstests, die die Funktionalität der vollständigen Software inklusive aller Module testen, erreicht. Das Ziel von CDE ist es, den Prozess der Softwareauslieferung zu beschleunigen und zuverlässiger zu machen, indem menschliche Fehler minimiert und schnelles Feedback über Probleme in der Produktionsumgebung ermöglicht wird.[7]

Continuous Deployment

Continuous Deployment (CD) ist der nächste Schritt nach Continuous Delivery. Bei CD wird jede Änderung, die den automatisierten Testprozess besteht, automatisch in die Produktionsumgebung eingespielt.[8] Dies bedeutet, dass neue Features und Updates mehrmals täglich an die Endbenutzer ausgeliefert werden können, was eine schnelle Reaktion auf Marktbedingungen und Kundenfeedback ermöglicht. Es ist jedoch zu beachten, dass CD eine hohe Reife der Entwicklungsprozesse und Testautomatisierung erfordert, um die Anzahl der in der Produktionsumgebung aufkommenden Fehler zu minimieren. Der Zusammenhang zwischen Continuous Integration, Delivery und Deployment wird in Abbildung 1 aufgezeigt.



Quelle: Übernommen von Shahin, Ali Babar und Zhu, 2017

Abbildung 1: Zusammenhang zwischen CI, CDE und CD

2.2 Begrifflichkeiten und Prinzipien von Continuous Integration

Um die Bedeutung und den Nutzen von Continuous Integration und des Continuous-Software-Engineering für die Softwareentwicklung nachvollziehen zu können, ist es hilfreich, einen Blick auf die historische Entwicklung und die grundlegenden Prinzipien der CI zu werfen. Ein Kernprinzip hinter Continuous Integration wurde bereits im Jahr 1991 von Grady Booch definiert. Hierbei werden Software-Releases nicht als ein großes Ereignis betrachtet, sondern regelmäßig durchgeführt,

[7] Vgl. Humble und Farley. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. 2010.

[8] Vgl. Shahin, Ali Babar und Zhu. „Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices“. 2017, S. 3911.

wobei die vollständige Software stetig größer wird.[9] Kent Beck popularisierte im Jahr 1998 die Disziplin des „Extreme Programming“, wobei großer Wert auf das frühe und regelmäßige Testen und Integrieren der entwickelten Komponenten einer Software gelegt wird. Beck behauptet hierbei, dass ein Feature, für welches es keine automatisierten Tests gibt, auch nicht funktioniert.[10] Im Jahr 2006 fasste Software-Entwickler Martin Fowler einige Bereiche dieser Methodiken in dem Artikel „Continuous Integration“ unter dem gleichnamigen Begriff zusammen. Fowler beschreibt CI als einen Prozess, bei dem Teammitglieder ihre Arbeit regelmäßig Integrieren, wobei Integration als der Build-Prozess, inklusive automatisierter Tests, für die vollständige Software mitsamt der erarbeiteten Änderungen zu verstehen ist.[11] Ein Ziel dieser Vorgehensweise ist die Reduktion der „Cycle Time“, welche die Zeitspanne von der Entwicklung eines Features bis zum Erhalten des Kundenfeedbacks beschreibt.[12] Da die erfolgreiche Implementierung von Continuous Integration auf der Kombination von verschiedenen Methoden zur Verwaltung von Softwareprojekten fußt, werden im Folgenden einige dieser Schlüsselaspekte kurz dargestellt.

Regelmäßige Integration

Die namensgebende Methodik der CI ist das regelmäßige Integrieren von Software und damit Verbunden ist der Software-Release. In der traditionellen Softwareentwicklung wird der Release als ein einmaliges, großes Ereignis betrachtet. Als Integration wird der Prozess des Einbindens einer einzeln entwickelten Komponente in die bisher bestehende Gesamtheit einer Software bezeichnet, wobei der Release das Zusammenfinden aller Komponenten und das Ausführen des Build-Prozesses bis hin zur fertigen, ausführbaren und auslieferbaren Software beschreibt. In der Vergangenheit wurde bei einem Release jede Einzelkomponente der Software manuell integriert und getestet, wobei dies als eigene Phase in der Entwicklung einer Applikation galt. In einem CI-gestützten Projekt wird der Prozess des Software-Releases vollständig automatisiert, sodass jedes Teammitglied eine entwickelte Komponente schnell Integrieren und einen Software-Build erzeugen kann, was sich positiv auf die Cycle Time auswirkt.[13] In der Regel finden das Bauen und Testen der Software in einer „Pipeline“ statt, welche die genutzten CI-Tools wie Test-Suites und Static-Code-Analysis ausführt.[14]

Automatisierte Tests

Neben der regelmäßigen Integrierung von Code gelten automatisierte Software-Tests und Quality-Checks als ein wichtiger Aspekt von CI. Hierbei werden oftmals Unit-Tests verwendet, welche eine einzelne Softwarekomponente auf ihre Funktion prüfen, abgekapselt von anderen Komponenten.[15] Wenn ein Test fehlschlägt, wird in der Regel die Pipeline unterbrochen. Neben den typischerweise schnell durchlaufenden Unit-Tests, die aufgrund ihrer isolierten Natur spezifische Komponenten prüfen, können nach dem Build-Prozess umfassende End-To-End-Tests für die gesamte Software

[9] Vgl. Booch. *Object oriented design with applications.* 1991.
[10] Vgl. Beck. „Extreme programming: A humanistic discipline of software development“. 1998, S. 2–4.
[11] Vgl. Fowler. *Continuous Integration.* 2006.
[12] Vgl. Elazhary, Werner, Li et al. „Uncovering the Benefits and Challenges of Continuous Integration Practices“. 2022, S. 2580.
[13] Vgl. Fowler. *Continuous Integration.* 2006.
[14] Vgl. Elazhary, Werner, Li et al. „Uncovering the Benefits and Challenges of Continuous Integration Practices“. 2022, S. 2571.
[15] Vgl. Ahmad, Haleem und Beg. „Test Driven Development with Continuous Integration: A Literature Review“. 2013, S. 280.

durchgeführt werden. Diese Systemübergreifenden Tests, welche das Zusammenspiel verschiedener Komponenten testen, erhöhen zwar die Dauer des Test-Prozesses, können aber wichtige Einsicht in die Qualität der Software gewähren.[16] Oftmals kommen auch Static-Code-Analysis Tools zum Einsatz, um eine Pipeline bei Unstimmigkeiten in der Syntax des eingeführten Codes abzubrechen.[17]

Reproduzierbarkeit

Ein weiterer zentraler Aspekt von Continuous Integration ist die Reproduzierbarkeit der CI-Pipeline. In einem CI-Umfeld werden der Build- und Testing-Prozess vollständig automatisiert und standardisiert, was bedeutet, dass diese unter den gleichen Bedingungen und mit den gleichen Ergebnissen wiederholt werden können. Dies ist von entscheidender Bedeutung, um sicherzustellen, dass die Software in verschiedenen Umgebungen (z.B. Entwicklung, Test, Produktion) konsistent funktioniert. Durch das Zentralisieren der Software und der CI-Tools in einer Versionskontrolle (VCS), in Verbindung mit dem Automatisieren des Build- und Testing-Prozesses, wird das Wiederherstellen von früheren Ständen in der Entwicklung und das Nachvollziehen von Fehlern im Entwicklungsprozess vereinfacht.[18]

Feedback

Schnelles Feedback ist ein weiteres grundlegendes Prinzip von Continuous Integration. Durch die Automatisierung des Build- und Test-Prozesses können Entwickler schnell Feedback über den Status ihrer Änderungen erhalten. Wenn ein Problem auftritt, z.B. ein Test fehlschlägt oder ein Build abbricht, wird das Team sofort benachrichtigt, sodass das Problem schnell behoben werden kann. Dies reduziert die Zeit und den Aufwand, die benötigt werden, um Fehler zu finden und zu beheben, und verbessert die Qualität der Software. Darüber hinaus fördert das schnelle Feedback die Kommunikation und Zusammenarbeit im Team, da alle Mitglieder ständig über den Status des Projekts informiert sind.[19,20]

2.3 Übersicht über die Shopware-Platform

Shopware wurde als Online-Shop-Software im Jahr 2000 durch Stefan Hamann ins Leben gerufen[21] und bietet heute in ihrer aktuellen Major-Version 6 eine moderne E-Commerce-Plattform auf Basis des PHP-Frameworks „Symfony“. Das Symfony-Framework wird neben Shopware noch von anderen PHP-Basierten Projekten wie dem CMS „Drupal“, dem Shop-System „Magento“ und einigen weiteren Programmen[22] als Grundlage genutzt und bildet somit ein erprobtes Fundament für die Shopware-Plattform. Shopware selbst ist nach der Installation bereits voll funktionsfähig und kann

[16] Vgl. Fowler. *Continuous Integration*. 2006.
[17] Vgl. Zampetti, Scalabrino, Oliveto et al. „How Open Source Projects Use Static Code Analysis Tools in Continuous Integration Pipelines“. 2017, S. 338.
[18] Vgl. Duvall, Matyas und Glover. *Continuous Integration: Improving Software Quality and Reducing Risk*. 2007, S. 74–75.
[19] Vgl. Elazhary, Werner, Li et al. „Uncovering the Benefits and Challenges of Continuous Integration Practices“. 2022, S. 2573.
[20] Vgl. Elazhary, Storey, Ernst et al. „ADEPT: A Socio-Technical Theory of Continuous Integration“. 2021, S. 26–27.
[21] Vgl. Shopware. *The story behind Shopware AG*. 2023.
[22] Vgl. Symfony SAS. *Projects using Symfony - Popular PHP projects using Symfony components or based on the Symfony framework*. 2023.

mit einem Backend und optional mit einem Frontend oder für das Konsumieren der mitgelieferten API eingerichtet werden. Die Software kann auf verschiedenen Plattformen gehostet werden, darunter Linux-Server und containerisierte Umgebungen. Darüber hinaus bietet Shopware als Unternehmen auch eine eigene Hosting-Lösung an, die speziell auf die Anforderungen der Software zugeschnitten ist. Die Plattform kann sowohl im Einzelbetrieb als auch als Cluster genutzt werden, um eine hohe Verfügbarkeit und Skalierbarkeit zu gewährleisten. Nachfolgend wird der Aufbau des Symfony-Frameworks dargestellt und die Architektur der darauf basierenden Shopware-Platform aufgezeigt.

Symfony-Framework

Symfony, entwickelt im Jahr 2005 von Fabien Potencier, ist ein PHP-basiertes Full-Stack-Framework. Das Projekt besteht aus verschiedenen Einzelkomponenten, welche unabhängig voneinander verwendet werden können, somit ist es sehr flexibel. Gleichzeitig bietet Symfony eine Reihe von Konventionen und Best Practises für das Erstellen und Nutzen von Komponenten, welche die Entwicklung von Anwendungen erleichtern und beschleunigen. Das Framework ist weit verbreitet und stellt eine robuste Grundlage für die Entwicklung umfangreicher Softwareprojekte dar. Es beinhaltet essenzielle Funktionen, wie Dependency Injection (DI), Profiling, Object-Relational Mapping (ORM), Session Handling, Routing, Formularverwaltung und eine Template-Engine. Durch den Einsatz des Paket-Managers „Composer“ können diese Grundfunktionen erweitert und zusätzliche Features in ein Projekt importiert werden. Symfony bietet eine Reihe von Konsolenbefehlen, die das Erstellen von eigenen Komponenten, die Durchführung von Datenbankmigrationen, das Ausführen von Tests und vieles mehr erleichtern. Die umfangreiche Dokumentation und die aktive Entwicklercommunity des Frameworks bieten einen hervorragenden Support für Entwickler. Zudem gewährleistet Symfony eine Langzeitunterstützung für jede Hauptversion mit einem Support-Zeitraum von drei Jahren, was die Zuverlässigkeit und Stabilität des Frameworks unterstreicht.[23]

Architektur der Shopware-Platform

Lorem ipsum

Lorem ipsum

Lorem ipsum

Lorem ipsum

Lorem ipsum

[23] Vgl. Engebreth und Sahu. „Introduction to Symfony“. 2023, S. 273–278.

```
Storefront
├── Controller
├── DependencyInjection
├── Event
├── Framework
├── Migration
├── Page
├── Pagelet
├── Resources
├── Test
├── Theme
├── .gitignore
├── composer.json
├── phpunit.xml.dist
├── README.md
└── Storefront.php
```

3 Analyse und Konzept

4 Entwicklung der CI-Strategie

5 Evaluierung

6 Schlussfolgerungen und Ausblick

Literaturverzeichnis

[1] eCommerceDB. *The Most Commonly Used Shop Software Among Online Shops in Germany - Shopware and Salesforce Share Rank No. 1.* eCommerceDB GmbH. 2023. URL: https://ecommercedb.com/insights/the-most-commonly-used-shop-softwares-among-online-shops-in-germany-shopware-and-salesforce-share-rank-no-1/4210 (besucht am 22. 06. 2023).

[2] Mojtaba Shahin, Muhammad Ali Babar und Liming Zhu. „Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices“. In: *IEEE Access* 5 (2017), S. 3909–3943. DOI: 10.1109/ACCESS.2017.2685629.

[3] Brian Fitzgerald und Klaas-Jan Stol. „Continuous software engineering: A roadmap and agenda“. In: *The Journal of Systems and Software* 123 (2017), S. 176–189. DOI: 10.1016/j.jss.2015.06.063.

[4] Kent Beck, Mike Beedle, Arie van Bennekum et al. *Manifesto for Agile Software Development.* 2001. URL: https://agilemanifesto.org/ (besucht am 30. 06. 2023).

[5] Lucas Gren und Per Lenberg. „Agility is responsiveness to change“. In: *Proceedings of the Evaluation and Assessment in Software Engineering.* ACM, Apr. 2020. DOI: 10.1145/3383219.3383265.

[6] Martin Fowler. *Continuous Integration.* Fowler, Martin. 2006. URL: https://martinfowler.com/articles/continuousIntegration.html (besucht am 26. 06. 2023).

[7] Jez Humble und David Farley. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation.* Addison-Wesley Signature Series (Fowler). Pearson Education, 2010. ISBN: 978-0-321-67022-9.

[8] Grady Booch. *Object oriented design with applications.* 1. Aufl. Benjamin/Cummings Pub. Co, 1991. ISBN: 978-0-805-30091-8.

[9] Kent Beck. „Extreme programming: A humanistic discipline of software development“. In: *Fundamental Approaches to Software Engineering.* Springer Berlin Heidelberg, 1998. DOI: 10.1007/bfb0053579.

[10] Omar Elazhary, Colin Werner, Ze Shi Li et al. „Uncovering the Benefits and Challenges of Continuous Integration Practices“. In: *IEEE Transactions on Software Engineering* 48.7 (Juli 2022), S. 2570–2583. DOI: 10.1109/tse.2021.3064953.

[11] Sheikh Fahad Ahmad, Mohd Haleem und Mohd Beg. „Test Driven Development with Continuous Integration: A Literature Review“. In: *International Journal of Computer Applications Technology and Research* 2 (Mai 2013), S. 281–285. DOI: 10.7753/IJCATR0203.1013.

[12] Fiorella Zampetti, Simone Scalabrino, Rocco Oliveto et al. „How Open Source Projects Use Static Code Analysis Tools in Continuous Integration Pipelines“. In: *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR).* 2017, S. 334–344. DOI: 10.1109/MSR.2017.2.

[13] Paul M. Duvall, Steve Matyas und Andrew Glover. *Continuous Integration: Improving Software Quality and Reducing Risk*. 1. Aufl. Addison-Wesley Signature Series. Pearson Education, 2007. ISBN: 978-0-321-63014-8.

[14] Omar Elazhary, Margaret-Anne Storey, Neil A. Ernst et al. „ADEPT: A Socio-Technical Theory of Continuous Integration“. In: *2021 IEEE/ACM 43rd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. 2021, S. 26–30. DOI: 10.1109/ICSE-NIER52604.2021.00014.

[15] Shopware. *The story behind Shopware AG*. Shopware AG. 2023. URL: https://www.shopware.com/en/company/story/ (besucht am 25. 06. 2023).

[16] Symfony SAS. *Projects using Symfony - Popular PHP projects using Symfony components or based on the Symfony framework*. Symfony SAS. 2023. URL: https://symfony.com/projects (besucht am 24. 06. 2023).

[17] Gunnard Engebreth und Satej Kumar Sahu. „Introduction to Symfony“. In: *PHP 8 Basics: For Programming and Web Development*. Berkeley, CA: Apress, 2023, S. 273–283. ISBN: 978-1-4842-8082-9. DOI: 10.1007/978-1-4842-8082-9_15.