Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Thema:

Aufbau eines Development- und IT Operations-Zyklus

sowie die Gestaltung eines Kundenportals

Praxisprojekt

Fachbereich I - Wirtschafts- und Gesellschaftswissenschaften

Studiengang Wirtschaftsinformatik Online

Erstbetreuer: Prof. Dr.-Ing. Alexander Huber

vorgelegt von: Jan Zimmermann  
920822  
Niemegker Straße 18  
12689 Berlin  
01749916408  
E-Mail: s79873@bht-berlin.de

Abgabetermin: 30. Juni 2022

Sperrvermerk

Diese Arbeit enthält vertrauliche Daten der Holzkontor Preußen GmbH. Eine Weitergabe der Arbeit im Ganzen oder in Teilen sowie das Anfertigen von Kopien (auch digital) - sind grundsätzlich untersagt. Ausnahmen bedürfen der schriftlichen Genehmigung.

Anschrift:

Holzkontor Preußen GmbH

Heiko Schurwin

Sophienwerderweg 60

13597 Berlin

Berlin, den 30. Juni 2022

Inhaltsverzeichnis

Inhaltsverzeichnis III

Verzeichnis der Abkürzungen und Akronyme IV

Abbildungsverzeichnis V

Tabellenverzeichnis VII

1 Einleitung 1

1.1 Betriebsprofil 1

1.2 Projektdefinition und Zielsetzung 1

1.3 Methodik 2

2 Grundlagen und Definition 3

2.1 Development- und IT Operations-Zyklus 3

2.2 Enterprise Resource Planning System 4

3 Ausgangssituation 5

4 Konzepterstellung 6

4.1 Entwicklungsumgebung 6

4.2 Development- und IT Operations-Zyklus 9

4.3 Kundenportal 10

5 Umsetzung 15

5.1 Entwicklungsumgebung 15

5.1.1 Webserver 16

5.1.2 integrierte Entwicklungsumgebung 17

5.2 Development- und IT Operations-Zyklus 20

5.3 Entwicklung 22

6 Projektabschluss und Fazit 25

A Fragebögen 26

B Mockups 28

C Kundenportal 31

Literaturverzeichnis 39

Verzeichnis der Abkürzungen und Akronyme

API Application Programming Interface

BHT Berliner Hochschule für Technik

CI/CD Continuous Integration/Continuous Deployment

CSS Cascading Style Sheets

CSV Comma-separated values

DevOps Development- und IT Operations

ERP-System Enterprise Resource Planning System

HKP Holzkontor Preußen

HTML Hypertext Markup Language

IDE integrierte Entwicklungsumgebung

OCR Optical Character Recognition

PDF Portable Document Format

PHP PHP: Hypertext Preprocessor

SQL Structured Query Language

Abbildungsverzeichnis

[**Abb. 2.1:** Vergleich der Methoden – DevOps deckt den gesamten Lebenszyklus ab 3](#_Toc105061034)

[**Abb. 4.1:** Anfragen pro Sekunde 7](#_Toc105061035)

[**Abb. 4.2:** Speicherverbrauch 7](#_Toc105061036)

[**Abb. 4.3:** Antwortzeit 8](#_Toc105061037)

[**Abb. 4.4:** aktueller Wiegebeleg 12](#_Toc105061038)

[**Abb. 4.5:** aktuelle Rechnung 13](#_Toc105061039)

[**Abb. 4.6:** Mockup – Wiegungen 14](#_Toc105061040)

[**Abb. 5.1:** Screenshot der Plugins (Strg+Alt+S) 15](#_Toc105061041)

[**Abb. 5.2:** Screenshot XAMPP 16](#_Toc105061042)

[**Abb. 5.3:** Screenshot phpMyAdmin 17](#_Toc105061043)

[**Abb. 5.4:** Terminal PhpStorm 17](#_Toc105061044)

[**Abb. 5.5:** IDE komplett 17](#_Toc105061045)

[**Abb. 5.6:** IDE – Projektüebrsicht 18](#_Toc105061046)

[**Abb. 5.7:** IDE – geöffnete Datei 18](#_Toc105061047)

[**Abb. 5.8:** IDE – geöffnete Datei 19](#_Toc105061048)

[**Abb. 5.9:** IDE – gefundene Fehler und Probleme 19](#_Toc105061049)

[**Abb. 5.10:** IDE – SonarLint 19](#_Toc105061050)

[**Abb. 5.11:** IDE – Git Verknüpfung 19](#_Toc105061051)

[**Abb. 5.12:** GitLab – Versionen 20](#_Toc105061052)

[**Abb. 5.13:** GitLab – Anforderungen 20](#_Toc105061053)

[**Abb. 5.14:** GitLab – Meilensteine 20](#_Toc105061054)

[**Abb. 5.15:** GitLab – CI/CD-Pipeline 21](#_Toc105061055)

[**Abb. 5.16:** Docker-Container 21](#_Toc105061056)

[**Abb. 5.17:** TeamCity CI/CD-Pipeline 22](#_Toc105061057)

[**Abb. 5.18:** TeamCity Auswertung 22](#_Toc105061058)

[**Abb. 5.19:** Grafana Warnungen 22](#_Toc105061059)

[**Abb. 5.20:** Kundenportal Lokalität DE 23](#_Toc105061060)

[**Abb. 5.21:** Kundenportal Lokalität EN 23](#_Toc105061061)

[**Abb. 5.22:** Kundenportal Filter- und Suchfunktion sowie Option CSV/PDF 24](#_Toc105061062)

[**Abb. 5.23:** Kundenportal Ausgabe als CSV 24](#_Toc105061063)

[**Abb. 5.24:** Kundenportal Beispiel neuer Wiegebeleg 24](#_Toc105061064)

[**Abb. B.1:** Mockup – Disposition 28](#_Toc105061065)

[**Abb. B.2:** Mockup – Nutzer 29](#_Toc105061066)

[**Abb. B.3:** Mockup – Nutzer anlegen 29](#_Toc105061067)

[**Abb. B.4:** Mockup – Fahrzeuge 30](#_Toc105061068)

[**Abb. B.5:** Mockup – Fahrzeuge anlegen 30](#_Toc105061069)

[**Abb. B.6:** Mockup – Vertragswesen 31](#_Toc105061070)

[**Abb. B.7:** Mockup – Buchhaltung 31](#_Toc105061071)

[**Abb. C.1:** Kundenportal – Loginseite 32](#_Toc105061072)

[**Abb. C.2:** Kundenportal – Nutzer anzeigen 32](#_Toc105061073)

[**Abb. C.3:** Kundenportal – Nutzer erstellen 33](#_Toc105061074)

[**Abb. C.4:** Kundenportal – Nutzer bearbeiten 33](#_Toc105061075)

[**Abb. C.5:** Kundenportal – Kunden bearbeiten 34](#_Toc105061076)

[**Abb. C.6:** Tabelle – Sorten 34](#_Toc105061077)

Tabellenverzeichnis

[**Tab. 4.1:** Auswertung der Frameworks 8](#_Toc105061078)

[**Tab. 4.2:** Fragebogen 11](#_Toc105061079)

[**Tab. 5.1:** Plugins der IDE 16](#_Toc105061080)

[**Tab. A.1:** Fragebogen – Annahme – Pamela Dziecol 11.04.2022 26](#_Toc105061081)

[**Tab. A.2:** Fragebogen – Disposition – Reiner Uderhardt 12.04.2022 26](#_Toc105061082)

[**Tab. A.3:** Fragebogen – Einkauf/Verkauf – Michael Ludwig 13.04.2022 27](#_Toc105061083)

[**Tab. A.4:** Fragebogen – Buchhaltung – Karina Herzog 14.04.2022 27](#_Toc105061084)

[**Tab. A.5:** Fragebogen – Controlling – Heiko Schurwin 14.04.2022 28](#_Toc105061085)

# Einleitung

Die Kundenbindung ist ein wichtiges Instrument für Betriebe, um langfristig am Markt bestehen und die eigene Position gegenüber der Konkurrenz aufrechterhalten gar verbessern zu können (vgl. Töpfer, 2020, S. 3). Hierbei trägt gerade die Kundenbetreuung eine entscheidende Rolle bei, besonders dann, wenn Kunden Zugriffsmöglichkeiten auf Informationen und Dokumente haben, außerhalb der eigentlichen Öffnungszeiten (vgl. Töpfer, 2020, S. 25). Die Umsetzung eines solchen Kundenportals sowie die kontinuierliche Erweiterung und Verbesserung der darin zur Verfügung stehenden Dokumente ist ein wichtiger Ansatz diese Kundenbindung zu erhöhen (vgl. Töpfer, 2020, S. 1 f. vgl. 2020, S. 43 f.). Diese Fortlaufende Entwicklung kann z. B. über ein Development- und IT Operations-Zyklus erreicht werden (vgl. Alt et al., 2017, S. 27 f.).

## Betriebsprofil

Der Entsorgungsfachbetrieb, Holzkontor Preußen GmbH, mit Sitz in Berlin ist ein mittelständischer Betrieb, der sich auf das Recycling von Holz spezialisiert hat. Ein regelmäßig zu durchlaufender Geschäftsprozess ist die Anlieferung von Holz, das recycelt werden soll. Dieser wird wie folgt ausgeführt. Das Holz wird vom Kunden angeliefert oder von Fahrzeugen der HKP abgeholt. Nach der Anlieferung wird das Holz eingewogen, vom Fahrzeug abgeladen und geprüft. Anschließend wird das entladene Fahrzeug ausgewogen und der Wiegebeleg erstellt. Die verschiedenen Holzqualitäten werden dabei getrennt weiterverarbeitet, sortiert und geschreddert. Hochwertigere Hackschnitzel werden recycelt und z. B. für die Spanplattenherstellung verwendet. Hackschnitzel minderwertiger Qualität dienen zur Energiegewinnung (z. B. zum Heizen oder zur Stromerzeugung). Die Holzkontor Preußen GmbH ist hierarchisch als Linienorganisation aufgebaut und untergliedert sich in verschiedene Abteilungen. Dabei zeichnen sich die Geschäftsführer durch Innovation, Technologieaffinität und Weitsicht aus. Was zum einen an der mir zur Verfügung gestellten Möglichkeit und zum anderen an dem Einsatz z. B. eines Elektro-Baggers zu erkennen ist. Folgend werden das Projekt und das gestellte Projektziel definiert.

## Projektdefinition und Zielsetzung

Ziel dieser Arbeit ist es, für die Umsetzung eines Kundenportals einen Development- und IT Operations-Zyklus (DevOps) aufzubauen und dafür geeignete Software zu identifizieren und auszuwählen. Durch den Einsatz eines Kundenportals sollen die Kundenbindung gestärkt und die Stärke im Bereich Dienstleistungen ausgebaut werden. Durch den DevOp -Zyklus wird die kontinuierliche Entwicklung begleitet durch parallele Tests ermöglicht. Über das zu Kundenportal, welches im Projektverlauf entwickelt und bereitgestellt wird, sollen die Kunden jederzeit ihre Unterlagen abrufen können. Daraus ergeben sich zwei Vorteile: Die Kunden erhalten unabhängig von den Geschäftszeiten die benötigten Informationen und Belege und das Unternehmen spart Zeit und Geld, da die Belegerstellung kein Personal erfordert. Die Optimierung des Wiegeprozesses und der Erstellung der Anlieferscheine ist ein weiteres Projektziel. Im nächsten absatz wird die Methodik zur Umsetzung beschrieben.

## Methodik

Beginnend wird geeignete Software zum Aufbau eines Development- und IT Operations-Zyklus herausgesucht und mithilfe von Kriterien verglichen. Diesbezüglich wurde eine Literaturrecherche basierend auf die zehn Schritte nach Hirt und Nordhausen (vgl. Nordhausen & Hirt, 2020, S. 10) durchgeführt. Der Zeitrahmen für diese Literaturrecherche wurde bei wissenschaftlichen Artikeln auf 5 Jahre und bei themenbezogenen Fachbüchern auf 15 Jahre festgelegt. Die bei der Suche verwendeten Begriffe, Schlagworte und Suchstrings sind Abwandlungen des Titels dieser Arbeit. Vorrangig dienten dabei Google Scholar, Springer Link und Webopac als Literaturdatenbanken. Für die Sammlung, Sichtung und Organisation der über die Recherche ausgewählten Literatur wurde das Literaturverzeichnisprogramm Zotero genutzt,. Anhand der Literatur erfolgte die Auswahl der für den DevOp-Zyklus geeigneten Software und eine Bestimmung der Kriterien zur Bewertung der Software. . Anschließend wurde mit Hilfe der Bewertungskriterien die Software „PhpStorm“ als Entwicklungsumgebung (IDE) ausgewählt, installiert und eingerichtet. In „PHPStorm“ erfolgten die kontinuierliche Entwicklung, Tests und Bereitstellung der Software zum Kundenportal Das Kapitel „Entwicklungsumgebung“ beschreibt die Softwareentwicklung im Detail.

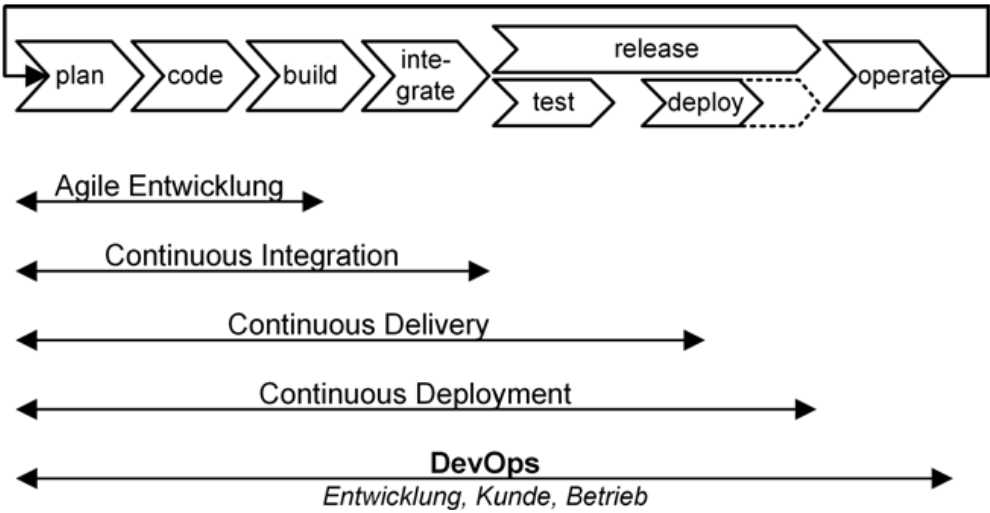
# Grundlagen und Definition

Für das Verständnis der fachlichen Ausführungen werden zunächst die wichtigsten Begriffe erläutert . Der Fokus liegt hierbei auf Development- und IT Operations sowie Enterprise Resource Planning System.

## Development- und IT Operations-Zyklus

Development und IT Operations kommen aus dem englischen und können mit Softwareentwicklung und dem IT-Betrieb übersetzt werden (vgl. Alt et al., 2017, S. 23). Beide Begriffe finden auch als ein Portmanteau ‚DevOps‘ Verwendung, wodurch die Zusammenarbeit dieser beiden Bereiche in den Vordergrund gerückt wird (vgl. Halstenberg et al., 2020, S. 1). Das Bemühen einer Zusammenarbeit über die Grenzen der eigenen Abteilung hinaus ist eines der Ziele von DevOps. Hierbei kommt es zu einer neuen Interpretation von bekannten und erfolgreichen Konzepten (vgl. Alt et al., 2017, S. 23; vgl. Halstenberg et al., 2020, S. 1).

DevOpsdecken , den kompletten Softwarelebenszyklus ab(vgl. Halstenberg et al., 2020, S. 16).



(Quelle: Halstenberg et al., 2020, S. 16)

**Abb. 2.1**: Vergleich der Methoden – DevOps deckt den gesamten Lebenszyklus ab

**Plan:** Anforderungen können mithilfe z. B. eines Kanban Boards an die zu entwickelnde Software festgelegt werden darüber hinaus ist es möglich, Probleme zu verfolgen (vgl. Halstenberg et al., 2020, S. 16 f. vgl. Katal et al., 2019, S. 4).

**Code:** Entwicklung der Software in einer geeigneten Entwicklungsumgebung unter Verwendung eines Programms zur Versionierung (vgl. Halstenberg et al., 2020, S. 18 f. vgl. Katal et al., 2019, S. 3 f.).

**Deployment Pipeline:** ist der Zusammenschluss der Punkte ‚build‘, ‚integrate‘ und ‚test‘, welche in der Abbildung separat aufgeführt werden. Unter build ist in dem Zusammenhang die Kompilierung der Software zu verstehen. Des Weiteren sorgt der integrate Prozess dafür, dass alle Abhängigkeiten zusammengeführt werden. Im Anschluss erfolgt automatisch ein kleiner Test der Grundfunktionalität (vgl. Halstenberg et al., 2020, S. 19; vgl. Katal et al., 2019, S. 5).

**Release & Deploy:** sind zwei Prozesse, die direkt aneinander anschließen können und beschreiben Bereitstellung und Freigabe einer getesteten Anwendung (vgl. Halstenberg et al., 2020, S. 20).

**Operate:** nach der Bereitstellung der Anwendung, muss diese Betrieben werden da erst im laufenden Betrieb weitere Fehlerquellen erkenntlich werden (vgl. Halstenberg et al., 2020, S. 21).

**Monitor:** der Erfolg von DevOps hängt möglicherweise von der Art und Weise ab, wie das System überwacht wird. Durch eine Überwachung ist es mögliche, positive und negative Auswirkungen zu erkennen und darauf zu reagieren (vgl. Katal et al., 2019, S. 4).

**Feedback & Improve:** die Möglichkeit während der Entwicklung Feedback zu erhalten und etwaige Fehler zu beseitigen (vgl. Halstenberg et al., 2020, S. 21; vgl. Katal et al., 2019, S. 7).

## Enterprise Resource Planning System

Enterprise Resource Planning System (ERP-Systeme) sind große, modulare Anwendungen, die für die meisten Geschäftsprozesse eines Betriebs konzipiert sind (vgl. Pekša, 2018, S. 1). Dabei bilden diese für viele Betriebe die Grundlage und sind ein nützliches Instrument zur Koordinierung von Ressourcen, Informationen und Abläufen (vgl. Vadivelu et al., 2018, S. 1). Gemäß Schoeneberg handelt es sich bei ERP-Systemen um Softwarelösungen in denen die unterschiedlichsten Bereiche eines Betriebes abgebildet werden können (vgl. Schoeneberg, 2011, S. 1).

# Ausgangssituation

In diesem Kapitel wird die Ausgangssituation des Auftraggebers, die Holzkontor Preußen GmbH, beschrieben. Das Unternehmen nutzt die ERP-Lösung von rona:systems GmbH. Bei Problemen in der IT werden externe Berater oder rona:systems GmbH involviert, um diese zu lösen. Zum Projetstart gab es keinen Ansatz eines DevOps-Zyklus oder eines digitalen Kundenportals. Die Kunden mussten ihre Anfragen telefonisch, per E-Mail oder persönlich stellen. Nur die Mitarbeiter der HKP konnten die Kunden-Dokumente wie z. B. Wiegebelege, Rechnungen und sonstige Belege erstellen. Zum Projektstart waren Dokumente in ausgedruckter Form der Goldstandard. Das führte zu Wartezeiten in der Kundenbetreuung und der Kundenservice war auf die Geschäftszeiten eingeschränkt. . Dieser Umstand war für das Unternehmen und die Kunden ineffizient. Daraus resultierte Unzufriedenheit beim Kunden und bei den Mitarbeitern der HKP, da diese ihre Kunden nicht immer wie gewünscht betreuen konnte. Das Management der HKP erkannte die Notwendigkeit, bestehende Geschäftsprozesse durch Digitalisierung effizienter zu gestalten, um so Zeit und Kosten zu sparen und den Kundenservice zu verbessern sowie die Kundenbindung zu erhöhen. Diese Erkenntnis war ausschlaggebend für die Einführung des DevOps-Zyklus zu einem neuen Kundenportal. Dieser Auftrag wurde innerhalb des Praxisprojekts bearbeitet. Der präzisen Beschreibung des Projektauftrags und der Zielformulierung schießen sich die Ist-Analyse und die Anforderungserhebung (Requirements Engineering) mit der Soll-Konzeption an. In dieser Projektphase werden die Anforderungen konkretisiert. Dabei werden verschiedenen Methoden, wie die Befragung der Beteiligten (Management, Mitarbeiter, Kunden etc.), die Analyse der bestehenden Prozesse, Sichten von Dokumenten (Arbeitsanweisungen, Spezifikationen, Prozessbeschreibungen, Vorgaben zu Qualitätsstandards etc.), der Analyse der bestehenden Systemlandschaft und der Aufnahme der Anforderungen anhand von Dokumenten und Befragungen, angewandt. Die Ist-Analyse und Anforderungserhebung startete mit den Interviews der Mitarbeiter aus den Abteilungen „Annahme“, „Disposition“, „Einkauf/Verkauf“, „Buchhaltung“, „Controlling“ und „Betriebsführung“. Dabei wurden die aktuelle Situation und die Anforderungen an das Kundenportal erhoben und dokumentiert und ein grobesSoll-Konzept erstellt. Das Soll-beschreibt die Rahmenbedingungen und Möglichkeiten des digitalen Kundenportals. Durch die Aufnahme der Anforderungen und der daraus entwickelten Soll-Konzeption ließ sich die durch DevOp gestützte Softwareentwicklung zum Kundenprotal planen. Das nächste Kapitel beschreibt die Entwicklung der Soll-Konzeption detailliert.

# Konzepterstellung

In der Konzepterstellung wird die Grob-Konzeption zur Umsetzung der DevOps und der Aufbau sowie die Arbeit mit der Entwicklungsumgebung (IDE) beschrieben. Daran schließt sich die Beschreibung zur Entwicklung des Kundenportals an.

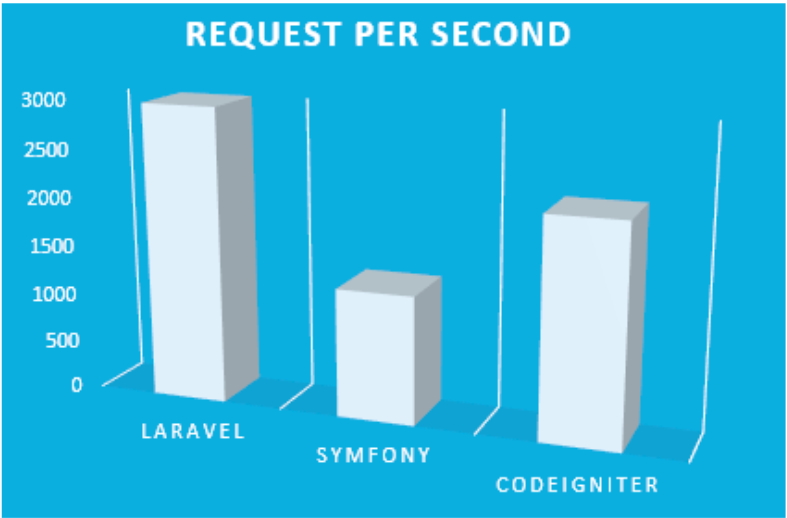
## Entwicklungsumgebung

Die Entwicklung des Kundenprotals erfolgte mit der IDE PhPStorm. Folgende Entwicklungs-Sprachen wurden dabei genutzt:

* Structured Query Language (SQL): ist eine Sprache zur Spezifizierung von Datenbankabfragen in einer relationalen Datenbank (vgl. Dobslaw et al., 2018, S. vi).
* PHP: Hypertext Preprocessor (PHP): gilt als eine der am weitesten verbreiteten Skriptsprachen in der Web-Anwendungsentwicklung, weil sie große Flexibilität und SQL Unterstützung bietet (vgl. Laaziri et al., 2019, S. 1 f.).
* Hypertext Markup Language (HTML): ist durch die Nutzung HTML-Tag definiert, wobei jeder Tag einen anderen Dokumentinhalt darstellt und definiert, wie der Browser den Inhalt formatieren und anzeigen muss (vgl. Eltahawey, 2016, S. 2).
* Cascading Style Sheets (CSS): bietet Gestaltungsmöglichkeiten, um z. B. Schriftart, Farbe und Größe von HTML-Elementen zu ändern, wofür es drei unterschiedliche Optionen gibt (Inline-Styles, Internal-Styles und External-Styles) (vgl. Eltahawey, 2016, S. 14).
* Javascript: ist eine Sprache, die auf Funktionalität ausgerichtet ist, aber auch zur Steuerung des Verhaltens und des Aussehens einer Website verwendet werden kann (vgl. Eltahawey, 2016, S. 1).

Nach Sichtung einiger Quellen und den dort beschriebenen Vorteilen von PHP-Frameworks wurde eines dieser zur Realisierung des Projektes implementiert. Frameworks helfen unter anderem den Entwicklern, Webanwendungen schneller und einfacher zu erstellen, indem sie ein grundlegendes Framework-Modell sowie einen vollständigen Satz von Application Programming Interface (API), Bibliotheken und Erweiterungen bereitstellen (vgl. Laaziri et al., 2019, S. 2). Laaziri bewertete verschiedene Frameworks hinsichtlich ihrer Performance bei der Kompilierung und Ausführung von Programmcode. Basierend auf diesen Validierungsergebnissen erfolgte die Auswahl des Frameworks Laravel.

Anfragen pro Sekunde

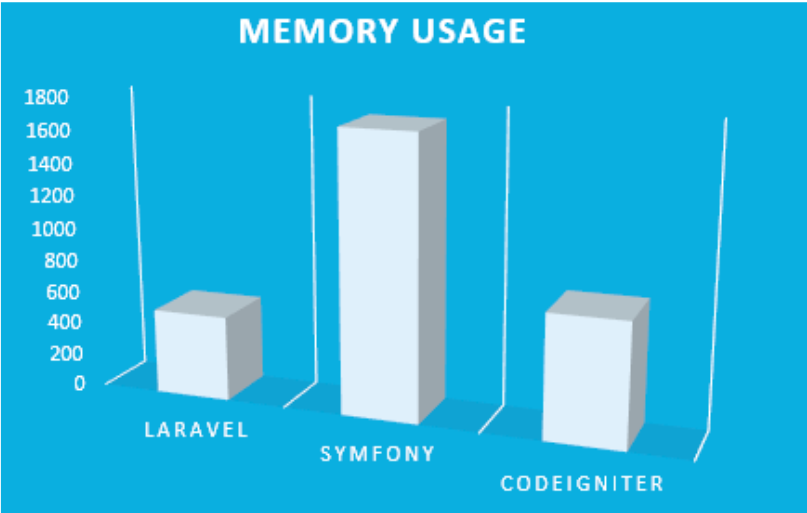


(Quelle: Laaziri et al., 2019, S. 5)

**Abb. 4.1**: Anfragen pro Sekunde

Das Framework Laravel verarbeitet bis zu 3000 Anfragen pro Sekunde (vgl. Laaziri et al., 2019, S. 7).

* Speicherverbrauch

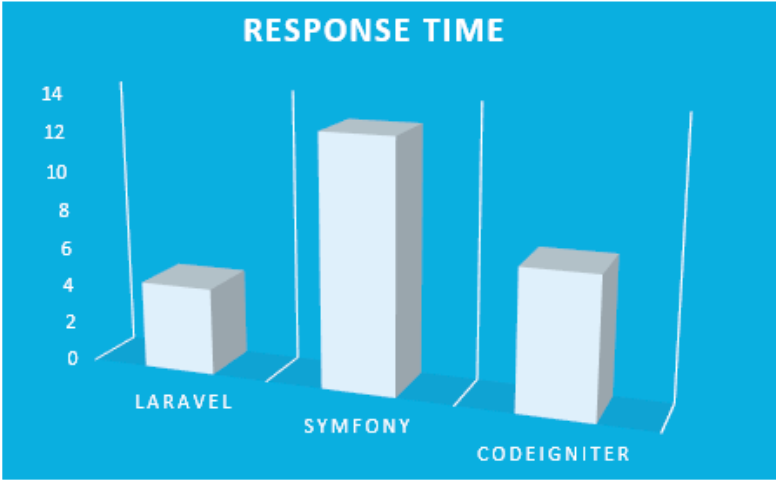


(Quelle: Laaziri et al., 2019, S. 6)

**Abb. 4.2**: Speicherverbrauch

In den Tests Laaziri et al. beanspruchte Laravel in der Ausführung durchschnittlich 518 KB Arbeitsspeicher. (vgl. Laaziri et al., 2019, S. 7).

* Antwortzeit



(Quelle: Laaziri et al., 2019, S. 6)

**Abb. 4.3**: Antwortzeit

Laravel weist mit 4,46 Millisekunden die geringste Antwortzeit auf, verglichen mit CodeIgniter mit 7,2, gefolgt von Symfony mit 12 Millisekunden (vgl. Laaziri et al., 2019, S. 7).

Des Weiteren fließen die Ergebnisse von Adamu et al. als Sicherheitsaspekte in die abschließende Bewertung mit ein, denn eine hohe Sicherheit entscheidet über die Effektivität und den Gesamterfolg jeder Anwendung und gerade Frameworks haben viele großartige und einfach zu verwendende Sicherheitsmodule integriert, um die mit Frameworks entwickelte Anwendung so sicher wie möglich zu machen (vgl. Adamu et al., 2020, S. 5). Laravel bietet im Vergleich zu den anderen getesteten Frameworks den größten Umfang an Sicherheitsfunktionen, wie z. B. eine Benutzer-Authentifizierung, einen besseren Schutz vor Cross-Site-Scripting, SQL-Injektion und Cross-Site Request Forgery sowie vor weiterem Angriffspotenzial. (vgl. Adamu et al., 2020, S. 6).

Tab. 4.1: Auswertung der Frameworks

|  |  |  |  |
| --- | --- | --- | --- |
| Kriterium/Framework | Laravel | Symfony | CodeIgniter |
| Anfragen pro Sekunde | + | - | 0 |
| Speicherverbrauch | + | - | 0 |
| Antwortzeit | + | - | 0 |
| Sicherheitsaspekt | + | - | - |

Unter Berücksichtigung der vorliegend betrachteten Kriterien, wird zusätzlich zur IDE PhpStorm das Framework Laravel genutzt, um das Kundenportal umzusetzen. Die genaue Konzeptionierung des Kundenportals wird im Anschluss näher beleuchtet.

## Development- und IT Operations-Zyklus

Die HKP ist ein kleines Unternehmen ohne eigene IT-Abteilung. Daher ist die effiziente und effektive Umsetzung der täglichen Arbeitsaufgaben ohne regelmäßiges Eingreifen in den Ablauf der IT-Prozesse eine der wichtigsten Anforderungen an den DevOp-Zyklus.. Viele der Geschäftsprozesse in der HKP haben Beziehungen und Abhängigkeiten untereinander. Daraus resultieren viele Schnittstellen zwischen den Prozessen und die Anforderung, viele diese Prozesse im DevOps-Zyklus mit wenig Software zu integrieren.

Für die kollaborative und kontrollierte Entwicklung haben sich Versionsverwaltungsprogramme, wie Gitlab bewährt. Programme zur Versionskontrolle ermöglichen die parallele Entwicklung durch mehrere Softwareentwickler, das Zurücksetzen auf frühere Entwicklungsstände und das Erstellen von Auswertungen.

(vgl. Valdivia, 2019, S. 8). GitLab bietet viele Optionen zur Umsetzung von Continuous Integration (Ein commit durchläuft mehrere Bereiche des DevOps-Zyklus) und weitere Funktionen, wie z. B. das Anlegen von Anforderungen und Meilensteinen sowie die Verfolgung des Fortschrittes.

, sprich es werden bei einem commit mehrere Bereiche des DevOps-Zyklus durchlaufen (vgl. Than & Phyu, 2019, S. 1).

In die IDE PhpStrom lassen sich eine Reihe von zusätzlichen Entwicklungswerkzeugen (Tools), die den Entwickler in seiner Arbeit unterstützen, integrieren. Die Software TeamCity ist ein Java-basierter Server zur kontinuierlichen Entwicklung (CD - Continuous Development ) und Integration (CI - Continuous Integration). D (vgl. Cepuc et al., 2020, S. 1 f.). Mit Hilfe von TeamCity können in intelligenten Tests Duplikate von Codeabschnitten gefunden und Statistiken über die Build-Dauer, der Code-Qualität und der Erfolgsrate erstellt werden. (vgl. Cepuc et al., 2020, S. 2).

Für die Realisierung des DevOps-Zyklus wurde zudem die Open-Source-Anwendung Docker genutzt. Diese steuert zentral die kontrollierte Ausführung verschiedener Programme und unterstützt die Prozessentwicklung durch das Packen von Abhängigkeiten in eine Standardform.

(vgl. Rad et al., 2017, S. 2). Gerade Docker dient hierbei als Bindeglied zwischen den einzelnen Anwendungen, da sowohl PhpStorm als auch TeamCity Docker unterstützen. Des Weiteren wird über Docker-Container der MySQL-Server und nginx, die Webserver-Software, betrieben.

Für die Entwicklung des Kundenportals wurde die IDE PhpStorm um viele Plugins, wie Prettier (dient der Formatierung des Codes), Mintlify (für die Dokumentation des Quellcodes) und Quodana (verbessert die Code Qualität) erweitert.

## Kundenportal

Folgendes Kapitel widmet sich dem spannenden Thema der Softwareentwicklung. Jede Softwareentwicklung beginnt mit der Analyse der Ausgangssituation (Ist-Analyse) und der Spezifizierung der Anforderungen (Requirements engineering). Aus den Ergebnissen der Anforderungsanalyse erstellt der Entwickler zusammen mit dem Auftraggeber das Soll-Konzept. Vor der Entwicklung wurden in mehreren Interviews mit den einzelnen Fachbereichen die Anforderungen aufgenommen.

In den ersten Gesprächen mit der Unternehmensleitung wurden folgend genannte Kern-Funktionalitäten des Portals bestimmt. . **Waagemodul:** Kunden sollen die Daten zu gespeicherten Wiegungen selbständig einsehen und drucken können. Dabei darf der Kunde nur die Wiegebelege zu seiner Kundennummer aufrufen können. ,

**Dispositionsmodul:** Über das Dispositionsmodul soll der Containertransport geplant werden. Der Disponent erhält nach Auftragseingang eine Nachricht und plant die Lieferung oder Abholung der Container. Im Dispositionsmodul sind die Auftragsdaten, eine Terminplanung und die Verfügbarkeit von Ressourcen (Mitarbeiter, Container und dazu passende Fahrzeuge) zu integrieren.

**Vertragsmodul:** Dauerhafte Verfügbarkeit der Preise und Verbindung zum Ein-/Verkauf um diese nachzuverhandeln, zuzüglich der Ausdruck als Portable Document Format (PDF).

**Buchhaltungsmodul:** Angelehnt an die Möglichkeiten des Waagemoduls, kann der Kunde Rechnungen raussuchen, um diese als PDF auszudrucken.

**Controllingmodul:** Es soll die Option bestehen, diverse Auswertungen zu tätigen. Dies soll mit dem Controllingmodul umgesetzt werden.

**Stammdatenpflege:** In diesem Bereich sollen Kunden eigenständig Nutzer anlegen, um entsprechende Rechte an Angestellte vergeben zu können. Fahrzeuge müssen hinterlegt werden, um unter anderem Lieferschein anlegen zu können, mit derer eine spätere Anlieferung erfolgen kann.

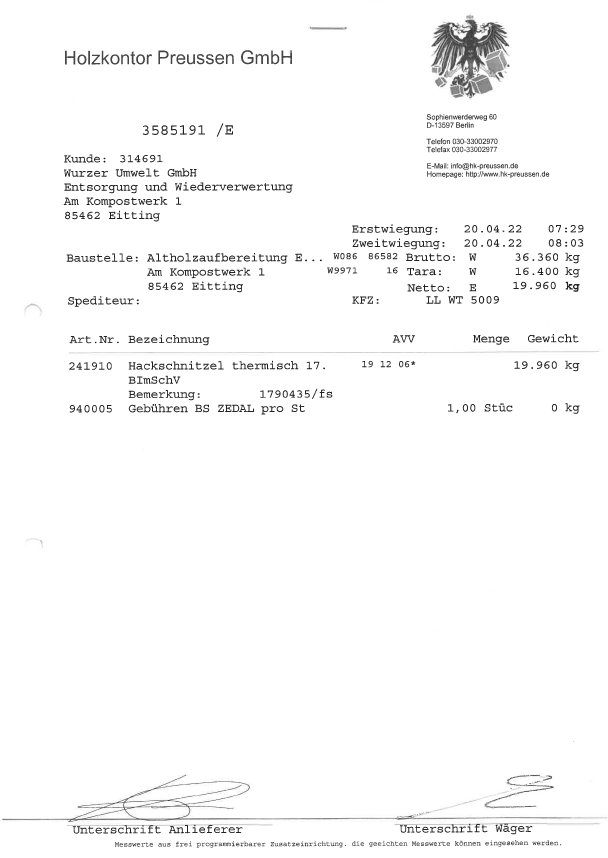
Der Spezifizierung der Kernfunktionalitäten, die im Kundenportal realisiert werden müssen, folgten die Interviews in den Fachabteilungen . Darin wurden die speziellen Anforderungen der Fachbereiche aufgenommen . In den Interviews wurde folgender auf die besonderen Anforderungen der Fachbereiche abgestimmter Fragebogen verwendet.

Tab. 4.2: Fragebogen

|  |  |
| --- | --- |
| Frage | Antwort |
| 1. Welche Grundfunktionalitäten müssen angeboten werden? |  |
| 1. Welche Daten dürfen Kunden sehen? |  |
| 1. Nach welchen Kriterien soll gesucht und gefiltert werden? |  |
| 1. Welche Rechte werden benötigt, um Zugriff auf diesen Bereich zu haben? |  |
| 1. Welche Unterseiten sind notwendig? |  |
| 1. Wie soll ein PDF, falls erforderlich aufgebaut werden? |  |
| 1. Welche Informationen sollen dort abgebildet werden? |  |
| 1. Ist es erforderlich Daten als Comma-separated values (CSV) Datei ausdrucken zu können? |  |
| 1. Welche Daten sollen hierbei ausgegeben werden? |  |

Im nächsten Schritt werden die Fragebögen ausgewertet. Die genauen Antworten können im Anhang (A) eingesehen werden.

**Annahme:** Mitarbeiter sollen Wiegebelege auch bearbeiten. Zum Benutzerkreis, die das Recht zur Bearbeitung der Wiegebelege erhalten, gehören Kraftfahrer, das Rechnungswesen (Financing) und die Unternehmensleitung (Management). Die Anwendung muss auch für ungeübte Benutzer intuitiv sein. Eine einfache und intuitive Suche nach bestimmten Daten (wie z. B. Wiegebelege) und das Filtern und die Sortierung dieser sind zu gewährleisten. Das soll die selbständige Nutzung des Kundenportals durch den Anwender ohne Rückfragen bei der Administration sicherstellen. Unterseiten strukturieren die Prozesskette zur Suche, Bearbeitung und zum Ausdruck von Wiegebelegen. Der Anwender soll Daten, die xlsx-Dateiformat vorliegen, in ein Tabellenkalkulationsprogramm (z. B. Excel oder Google Tabellen) laden und von dort ausdrucken können. Der Wiegebeleg sollte sich an dem aktuellen orientieren.

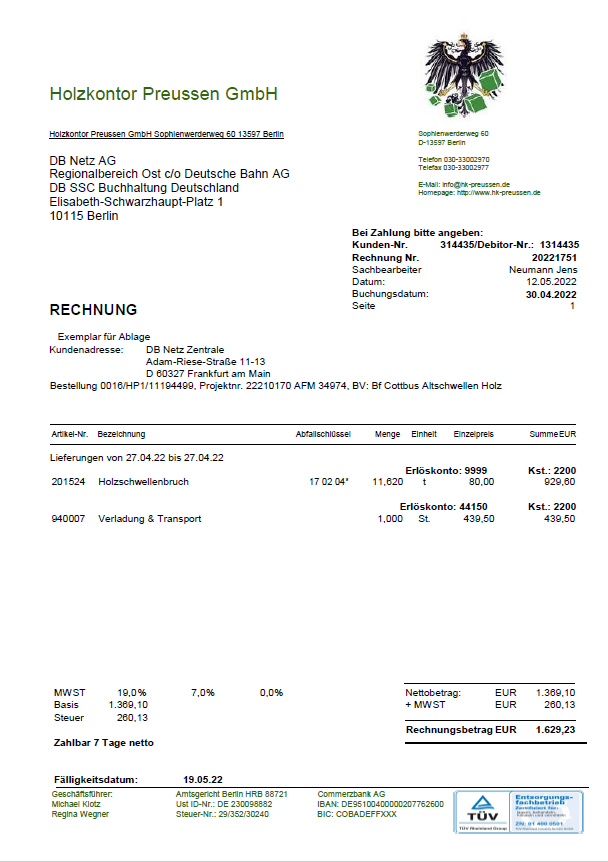
****

**Abb. 4.4**: aktueller Wiegebeleg

**Disposition:** Containeraktionen (Stellen, Tauschen, Abholen) muss in Einklang mit der tatsächlichen Behälteranzahl gebracht werden, wenn keine Container zur Verfügung stehen, kann auch kein neuer Container gestellt werden. Die Logik muss hierbei eine entscheidende Rolle spielen. Informationen über Anmeldungen dieser Tauschaktionen sollen bestmöglich direkt per E-Mail erfolgen, dass diese sinnvoll disponiert werden können. Lieferscheine sollen im Anschluss digital ausgedruckt werden können. Ein Kalender mit geplanten Containeraktionen soll genauso umgesetzt werden, wie die Aufträge per Drag and Drop den einzelnen Fahrzeugen zuordnen zu können.

**Vertragswesen:** Kunden dürfen nur ihre jeweiligen Preise sehen, welche als PDF ausgedruckt werden können. Nachverhandlungen sollen erst ab einer gewissen Zeit möglich sein direkt über einen Button, wo der Einkäufer eine Information per E-Mail erhält und sich dann mit dem Kunden in Verbindung setzen kann. Es müssen Preise den Kunden und Baustellen zugeordnet werden können, da einige Kunden unterschiedliche Preise für unterschiedliche Baustellen haben. Hierbei spielt die Qualität eine wichtige Rolle. Einbringen von Reklamationen und bildlicher Dokumentation, was bei der Annahme wichtig werden könnte. Zusätzlich wäre interessant, wenn Kunden bei Reklamation automatisch eine Notiz erhalten, um entsprechend reagieren zu können.

**Buchhaltung:** Kunden müssen die jeweiligen Rechnungen suchen und als PDF abspeichern können. Es muss bedacht werden, dass nicht jeder angestellte Zugriff auf die Rechnungen haben darf. Bei der Vorlage soll die aktuelle Rechnung als Grundlage genutzt werden.

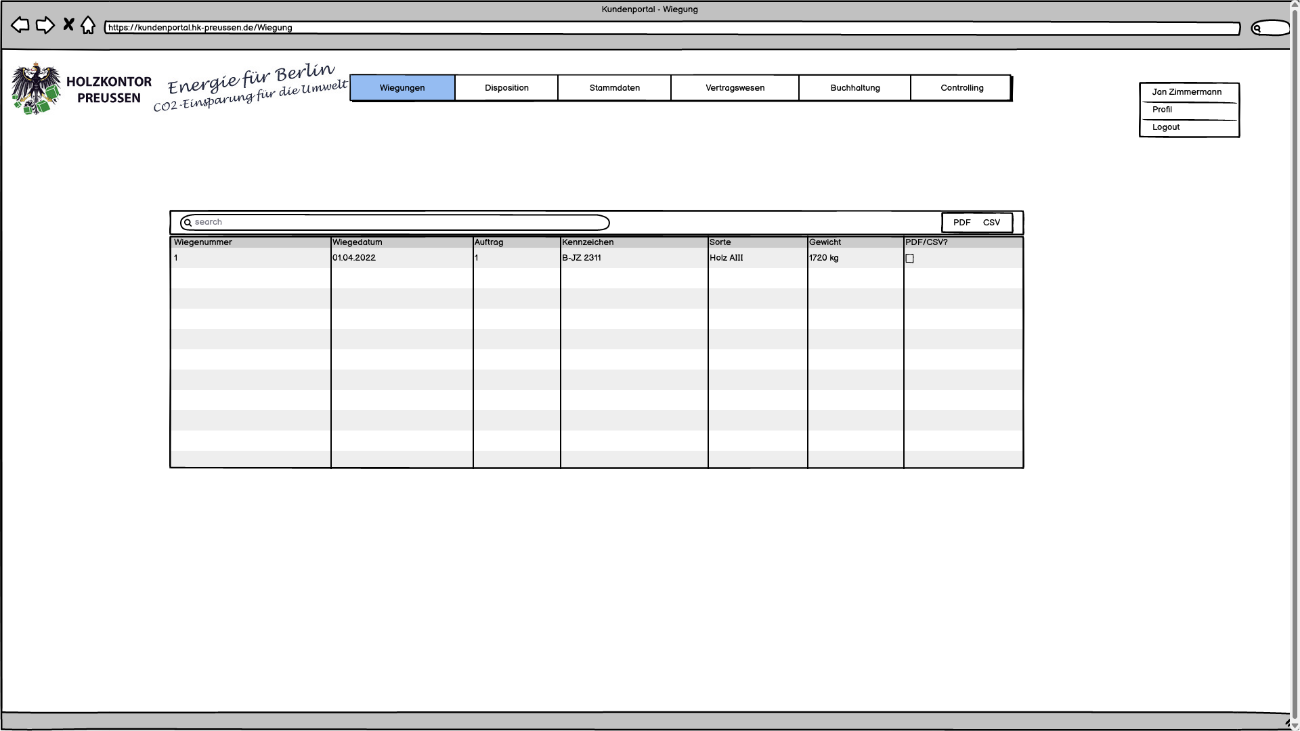
****

**Abb. 4.5**: aktuelle Rechnung

**Controlling:** Je mehr Auswertungen und Statistiken gezogen werden können, desto besser, da das Abfallrecht sehr komplex ist und behördlich viele Dokumente und Statistiken einzureichen sind, um den Status des Entsorgungsfachbetriebes aufrechterhalten zu können.

Aus den in den Interviews erhobenen Informationen konnte das Soll-Konzept zur Realisierung der Anforderungen und das Grobkonzept zur Gestaltung der Oberfläche bestimmt werden. In weiteren Feedbackgespräche ist die Konzeption verfeinert worden. Nachdem die erste Runde der Interviews für viele Informationen gesorgt haben, wie die Seiten explizit gestaltet werden müssen, um allen Anforderungen gerecht zu werden, fanden regelmäßige Feedbackgespräche statt.

Der Anlayse- und Defnintionsphase schließt sich die Entwurfsphase an. Diese geht der eigentlichen Implementierung voraus, In der Entwurfspahse werden die zu entwickelnden Systemkomponenten bestimmt, die Struktur der Daten und die Beziehungen der Daten untereinander modelliert sowie die Gestaltung der Benutzeroberfläche (GUI) entworfen. Für den Entwurf der GUI hat sich die Erstellung von mockups bewährt. Mockups sind eine grafische Abbildung der künftigen GUI ohne Funktionalität. Sie dienen vorrangig der Visualisierung und ermöglichen so die Vorstellung vom Programm, ohne das eine Codezeile geschrieben wurde. Folgende Abbildung zeigt beispielhaft ein mockup aus dem Projekt. Weitere Mockups sind zusätzlich zur Verfügbarkeit im GitLab auch im Anhang (B) zu finden.



**Abb. 4.6**: Mockup – Wiegungen

Wie aus den Anforderungen aus dem Bereich Waage hervorging, sollten die Wiegebelege bestmöglich über eine Suchfunktion sortier- und auffindbar sein. Des Weiteren muss die Möglichkeit bestehen, diese über eine Combobox zu selektieren und als PDF abspeichern und drucken zu können. Zusätzlich zur Druckfunktion als PDF soll, wenn möglich eine Option eingebettet werden, mit der die Wiegebelege in Form einer CSV-Tabelle heruntergeladen werden könne, damit diese im Anschluss vom Kunden geprüft und für weitere Auswertungen genutzt werden können.

Da nun alle wesentlichen internen und externen Bedingungen innerhalb dieses Konzeptes beschrieben sind sowie ein grober Plan zur Gestaltung existiert, stellt sich die Frage – wie lässt sich dieses Konzept nun erfolgreich und pragmatisch umsetzen? Mit dieser Frage befasst sich das folgende Kapitel.

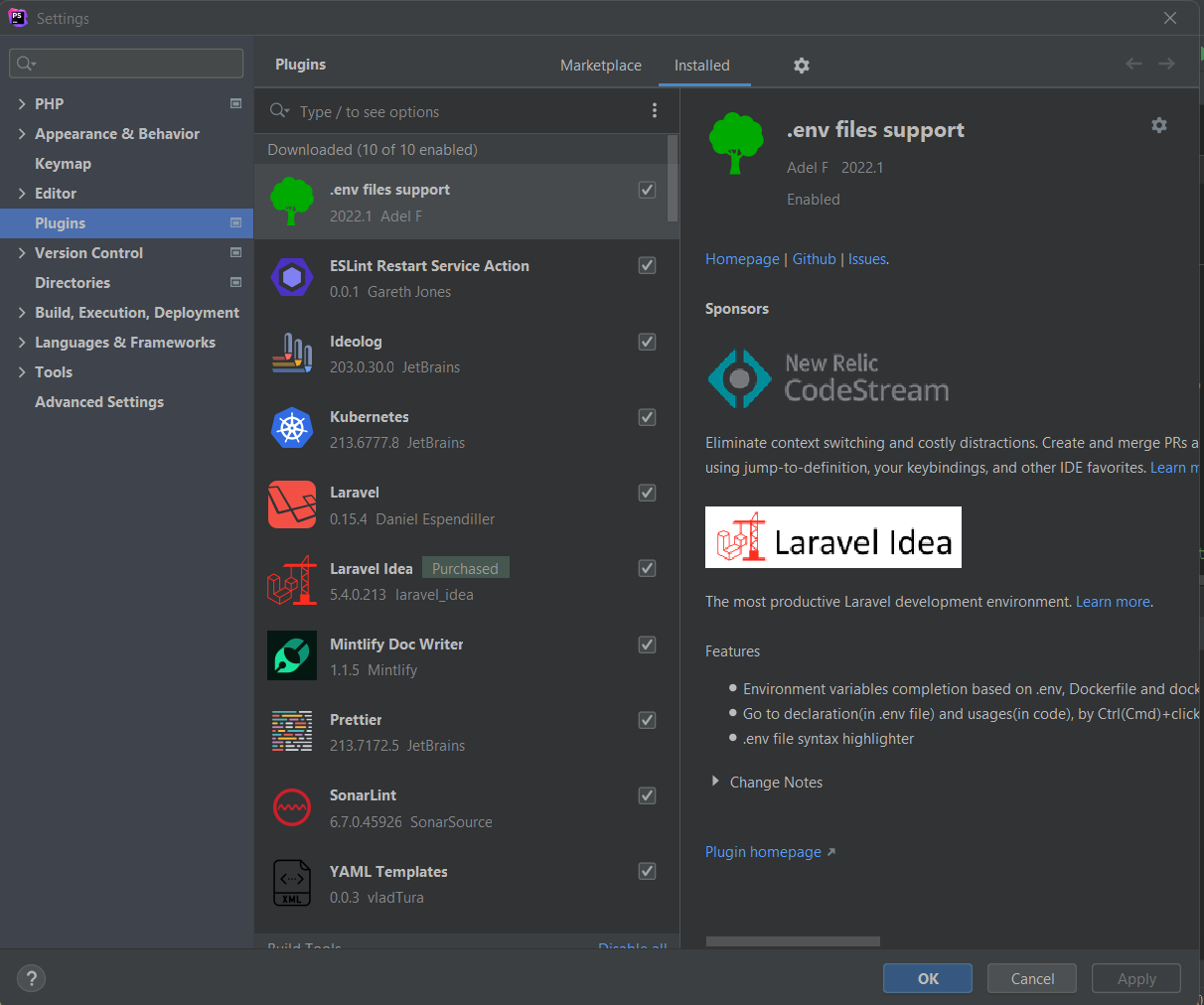
# Umsetzung

Dieses Kapitel beschreibt die Implementierung der Software mit Hilfe der IDE PhpStorm und des Frameworks Laravel. Das Kapitel beginnt mit einem Einblick in die IDE, dem sich die Demonstration des DevOps-Zyklus anschließt. Am Ende des Kapitels wird die Implementierung der Softwarekomponenten beschrieben.

## Entwicklungsumgebung

Vor Beginn der eigentlichen Entwicklung war die Entwicklungsumgebung mit allen dafür benötigten Komponenten zu installieren und zu konfigurieren. Für die Entwicklung des Kundenportals wurden die IDE PhPStorm, das PHP-Framework Laravel, das Datenbankmanagementsystem (DBMS) MySQL und einige für PhPStrom verfügbare Plugins installiert. Die Einrichtung der Entwicklungsumgebung beginnt mit der Installation und Konfiguration IDE PhpStorm. PhpStrom lässt sich durch Installationsroutinen und fertigen Installations- und Konfigurationsskripts einfach und schnell installieren und einrichten. Bestehende Abhängigkeiten zwischen einzelnen Modulen werden durch die automatisierte Installation automatisch erkannt und aufgelöst. Wird für die Installation einer Komponente in PhPStorm zusätzliche Software benötigt, installiert die Installationsroutine diese automatisch. Im zweiten Schritt wurde das DBMS MySQL installiert und eingerichtet. Durch die Installation von für die Entwicklung des Kundenportals nützlichen Plugins wurde die Einrichtung der Entwicklungsumgebung abgeschlossen. Die Tabelle Tab. 5.1 enthält die Funktionsbeschreibungen zu einigen der installierten Plugins.

Der Aufbau der Entwicklungsumgebung mithilfe der von JetBrains genutzten IDE PhpStorm war ein leichtes Unterfangen, obwohl sich die Registrierung als kompliziert herausstellte. Es wird im Moment, da diese Anwendung bisher nur für das Studium genutzt wird und noch kein konkreter Termin zur Veröffentlichung des Portals besteht, die kostenlose Möglichkeit für Studierende genutzt. Innerhalb dieser IDE wurden zusätzlich 87 Plugins installiert, einige sind standardmäßig enthalten, um die Arbeit zu vereinfachen, von denen auf die zehn wichtigsten näher eingegangen wird.



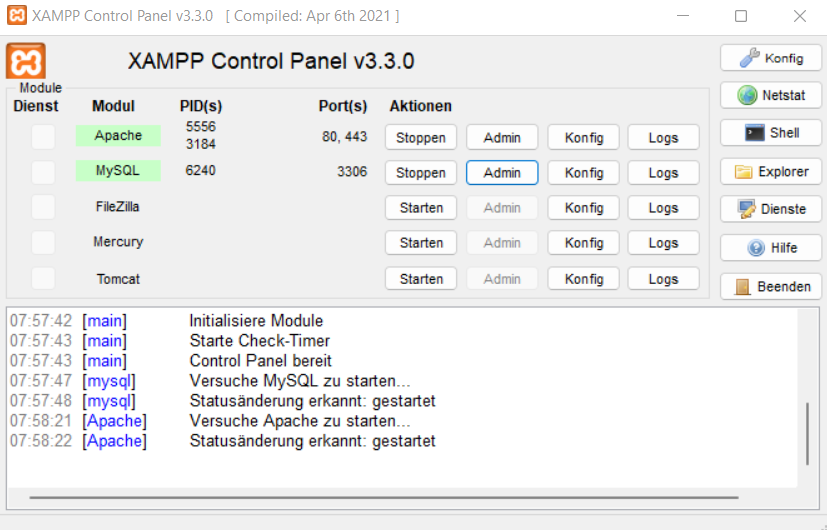
**Abb. 5.1**: Screenshot der Plugins (Strg+Alt+S)

Tab. 5.1: Plugins der IDE

|  |  |
| --- | --- |
| Plugin | Funktion |
| Git & GitHub | Bietet typische Git Standardfunktionen, um ein Repository zu Klonen oder zu erstellen.  Des Weiteren können push, pull, merge, commit Anfragen gestellt werden. |
| Laravel | Umsetzung aller für Laravel typischen Datei- und Komponentenvarianten wie z. B. Routes, Views, Controller, Blade, Templates, … |
| Laravel Idea | Erweitert die Funktionen des Laravel Plugins um Codegenerierung und unterstützt dabei das genutzte Livewire Paket. |
| .env files support | In der env-Datei wird unter Anderem der Zugang zur Datenbank gespeichert. Hierfür bietet dieses Plugin eine Unterstützung zusätzlich für Laravel. |
| SonarLint | Ein mächtiges Plugin, dass Bugs und Schwachstellen im Code findet und behebt. Des Weiteren werden Korrekturen und Rechtschreibprüfungen während des Schreibvorgangs angezeigt. |
| Grazie | Intelligente Rechtschreib- und Grammatikprüfung für das Geschriebene, funktioniert auch für Programmiersprachen. |
| Mintlify Doc Writer | Es kann nachträglich eine automatische Dokumentation für den Code umgesetzt werden. |
| Prettier | Nachdem der Code geschrieben wurde, bietet dieses Plugin die Möglichkeit, den Code ansehnlich zu formatieren. |
| Docker | Direkte Option, aus der IDE heraus, Docker-Container zu starten. |
| Quodana | Ist eine Plattform zur Überwachung der Codequalität im laufenden Entwicklungsprozess. |

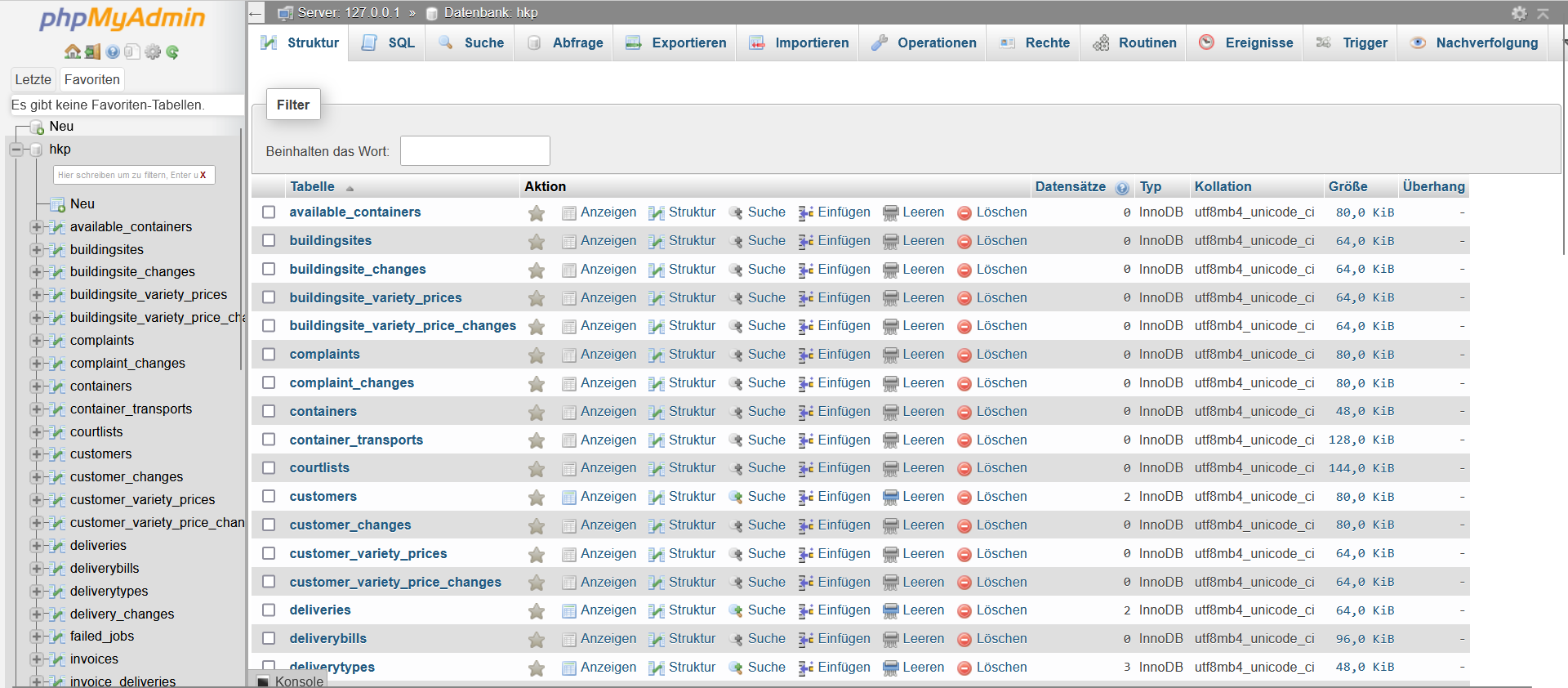
### Webserver

Anders als im Konzept beschrieben wurde zunächst xampp genutzt, um den MySQL-Server und phpMyAdmin zu betreiben.



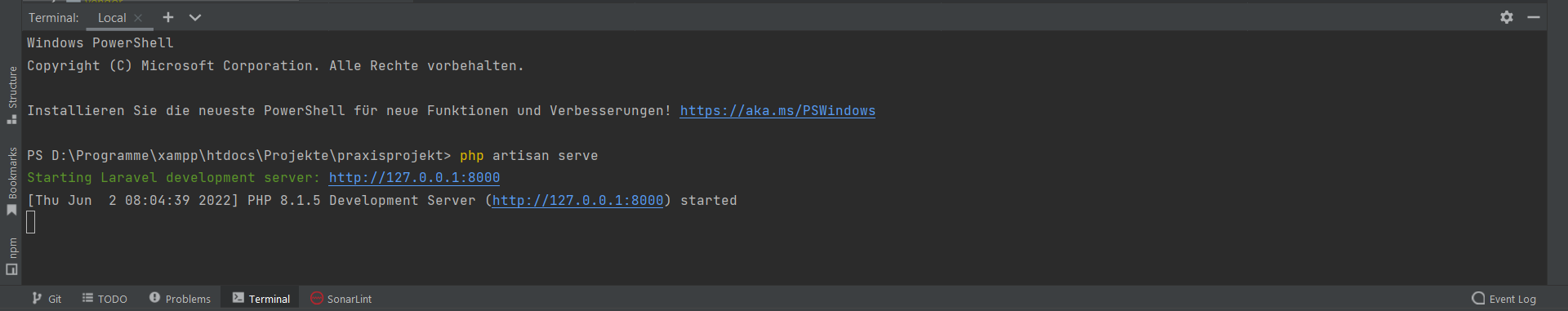
**Abb. 5.2**: Screenshot XAMPP

PhpMyAdmin bot dabei die Möglichkeit, die notwendige Datenbank „hkp“ anzulegen.



**Abb. 5.3**: Screenshot phpMyAdmin

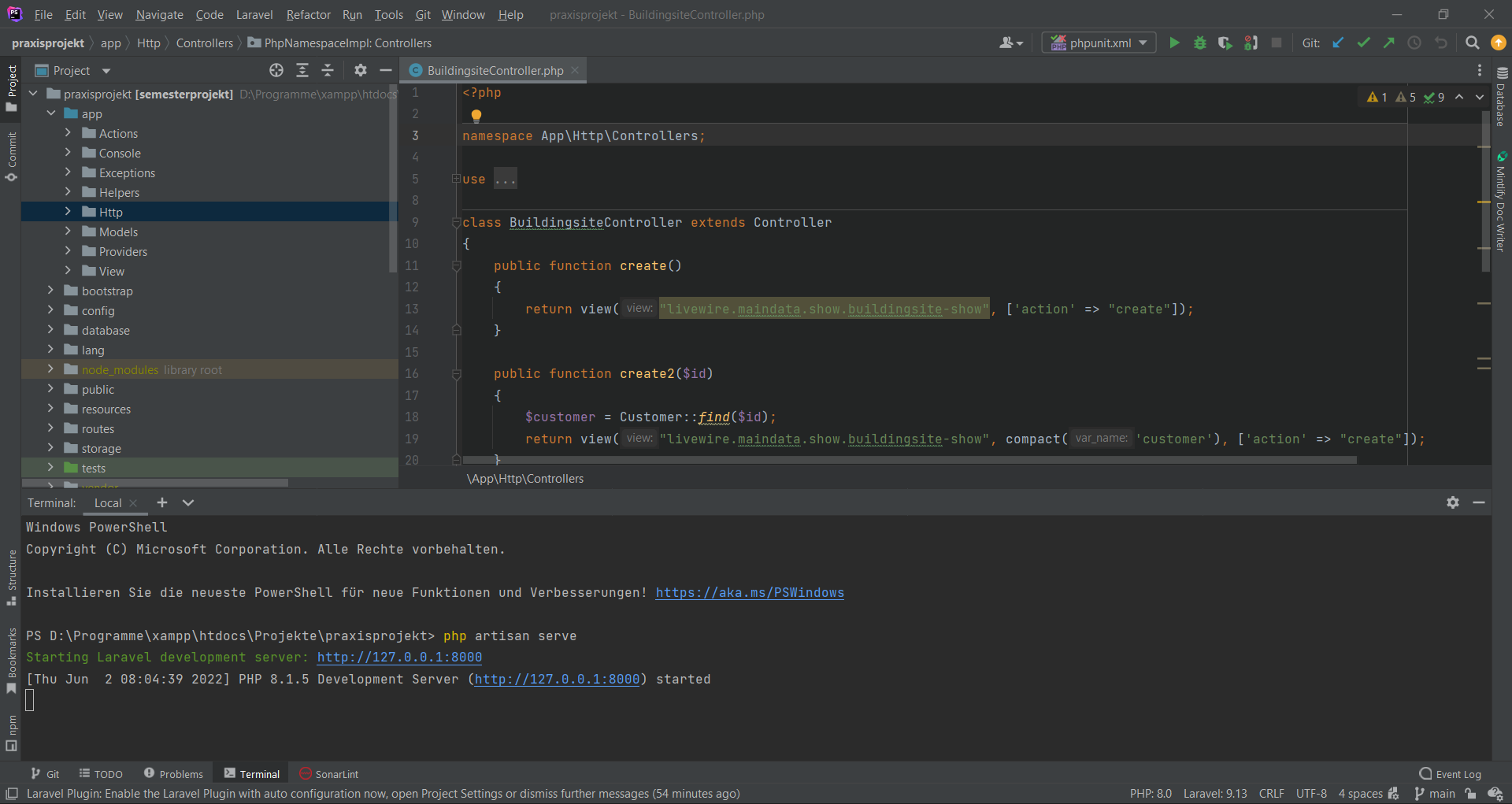
Direkt in PhpStorm wird ein Terminal zur Verfügung gestellt, über das mit dem Befehl „php artisan serve“ ein Laravel-Entwicklungsserver gestartet werden kann.



**Abb. 5.4**: Terminal PhpStorm

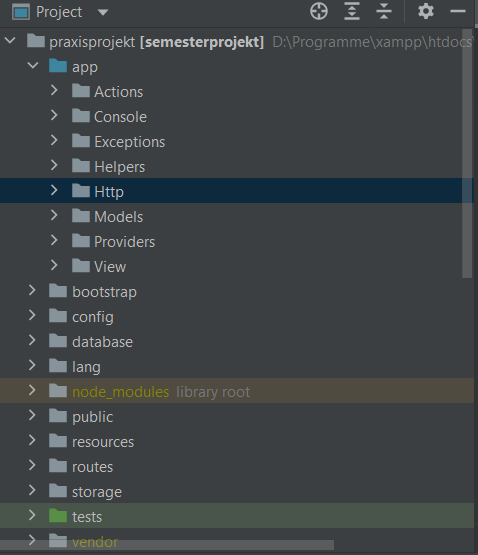
### integrierte Entwicklungsumgebung

Die IDE ist sehr komplex und deren Funktionen werden nachfolgend anhand von Screenshots und einer kurzen Beschreibung zur Nutzung vorgestellt.



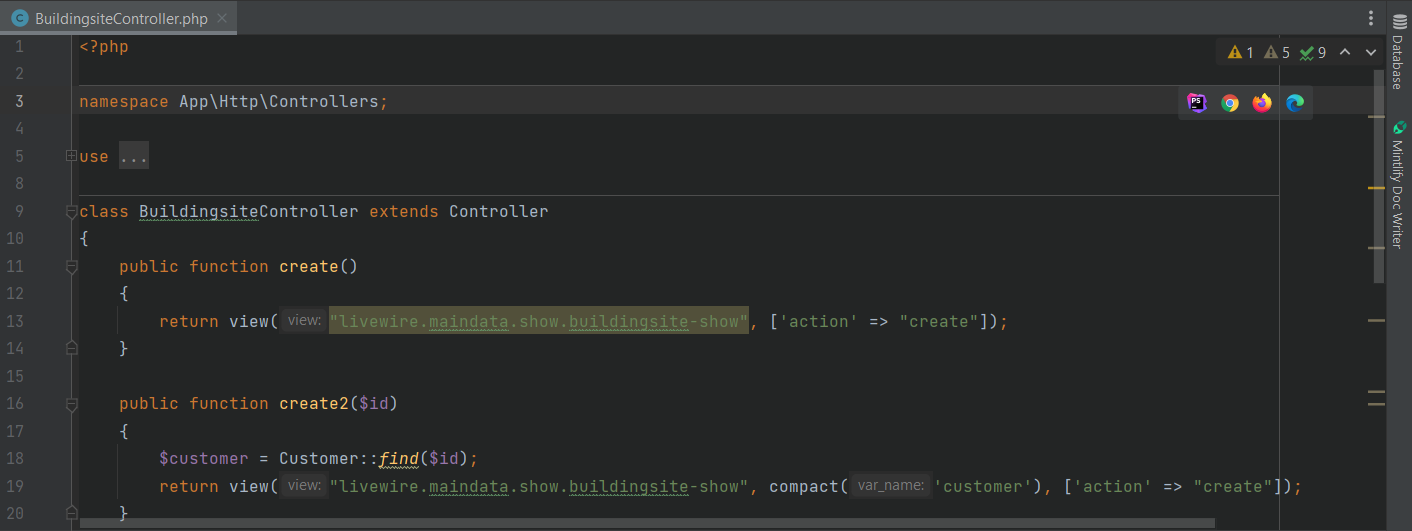
**Abb. 5.5**: IDE komplett

Grob kann PhpStorm in drei Arbeitsbereiche unterteilt werden, wobei diese individuell angepasst werden können. Der linke Bereich enthält die Projektübersicht, die Auflistung aller Dateien und Komponenten des Projektes.



**Abb. 5.6**: IDE – Projektüebrsicht

Im Rechten Abschnitt befinden sich die geöffneten Dateien, welche bearbeitet werden können.

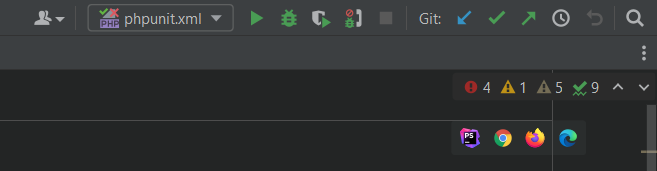


**Abb. 5.7**: IDE – geöffnete Datei

Über die Hervorhebung (Highlighting) von Codeabschnitten wird die Verknüpfung zu anderen Komponenten visualisiert.

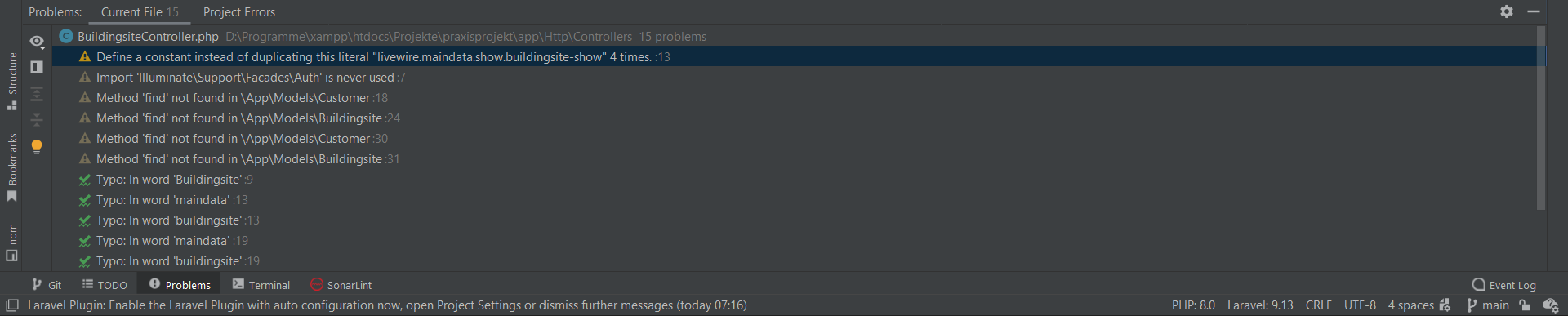
Innerhalb der geöffneten Datei wird einzelner Code hervorgehoben, über den eine Verknüpfung zu anderen Komponenten stattfindet.

Zusätzlich gibt es direkt Möglichkeiten um auf die Code-Qualität, die Fehler und auf das Repository zugreifen zu können.

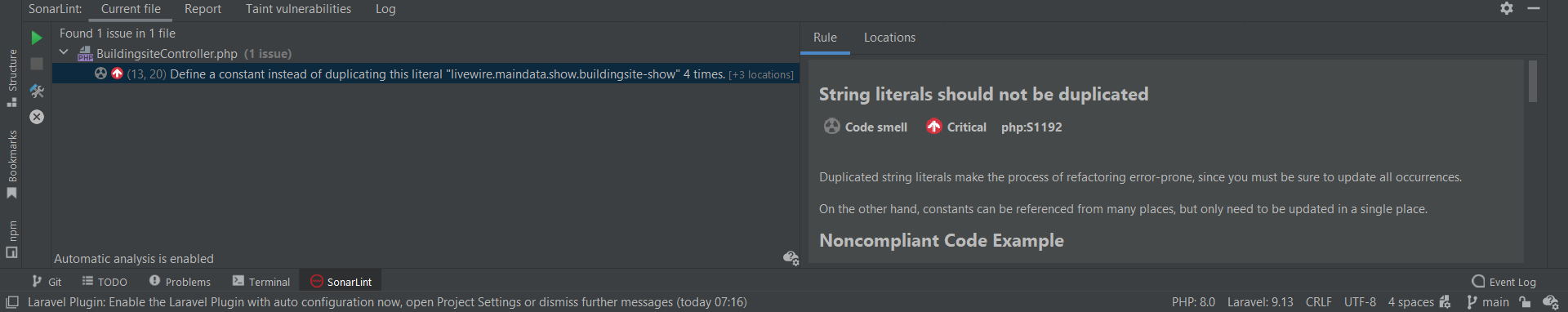


**Abb. 5.8**: IDE – geöffnete Datei

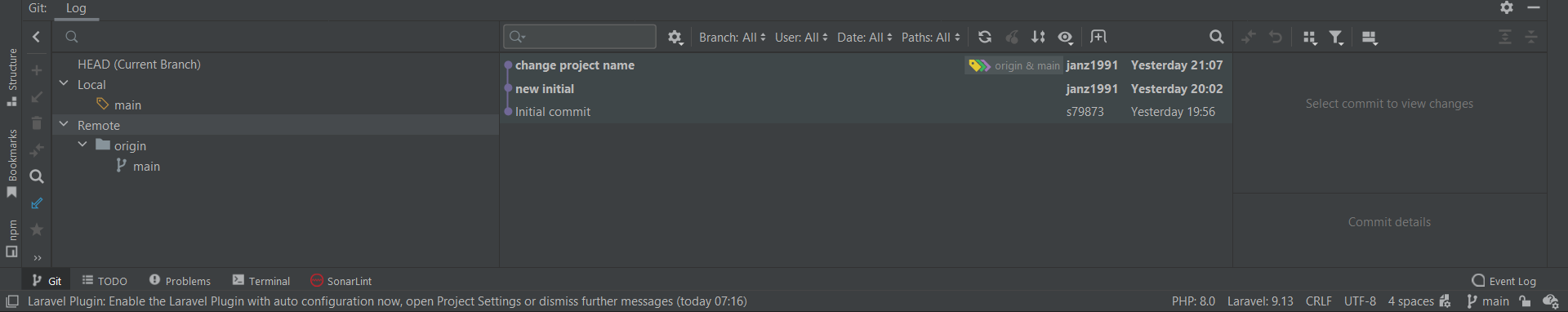
Im unteren Bereich von PhpStorm bietet sich die Möglichkeit zusätzliche Informationen abzurufen.



**Abb. 5.9**: IDE – gefundene Fehler und Probleme



**Abb. 5.10**: IDE – SonarLint



**Abb. 5.11**: IDE – Git Verknüpfung

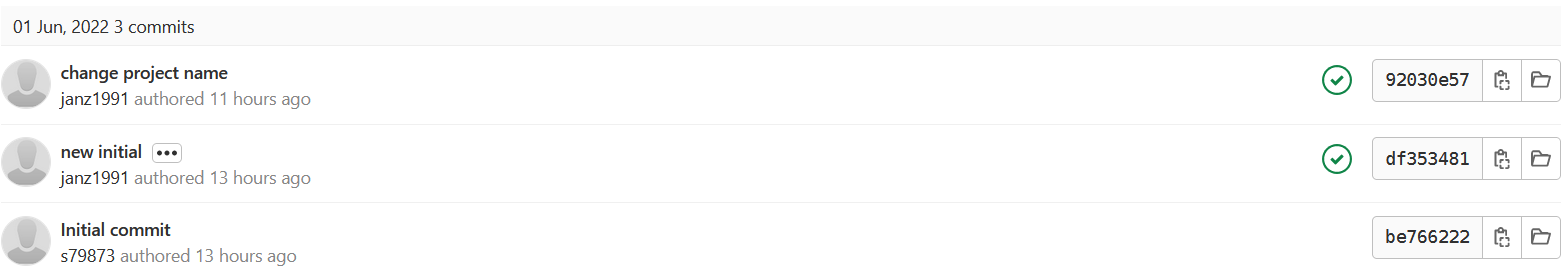
Zu diesen zusätzlichen Informationen zählen das Anzeigen von Fehler und die leichte Behebung dieser. Zudem bietet SonarLint die Funktion, die Code-Qualität zu steigern. Des Weiteren wird die Option zum Git-Repository gegeben.

Auf die großen Vorteile der genutzten Framworks Laravel, Livewire und Jetsream wird erst in der Entwicklung des Kundenportals eigegangen.

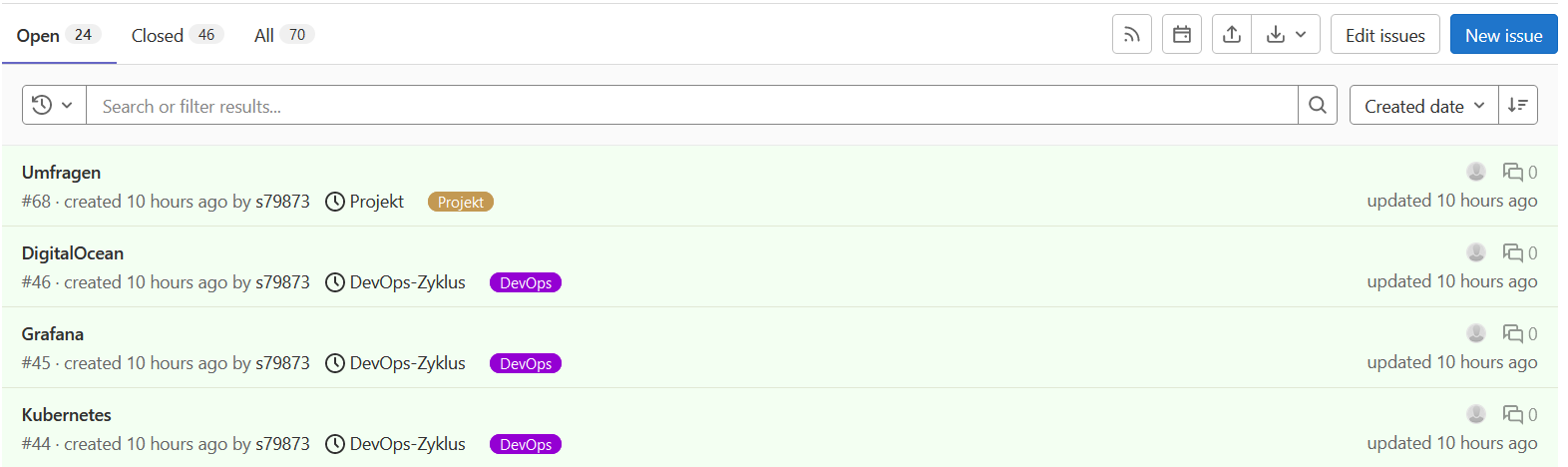
## Development- und IT Operations-Zyklus

Der DevOps-Zyklus orientiert sich stark an dem Vorsatz, Anwendungen im laufenden Betrieb schnellstmöglich ändern zu können und dabei auf ältere Versionen zurückspringen zu können. Wobei der Entwickler jederzeit Informationen zur Qualität des Codes erhält. Dadurch wird die Qualität des Codes stetig verbessert. .

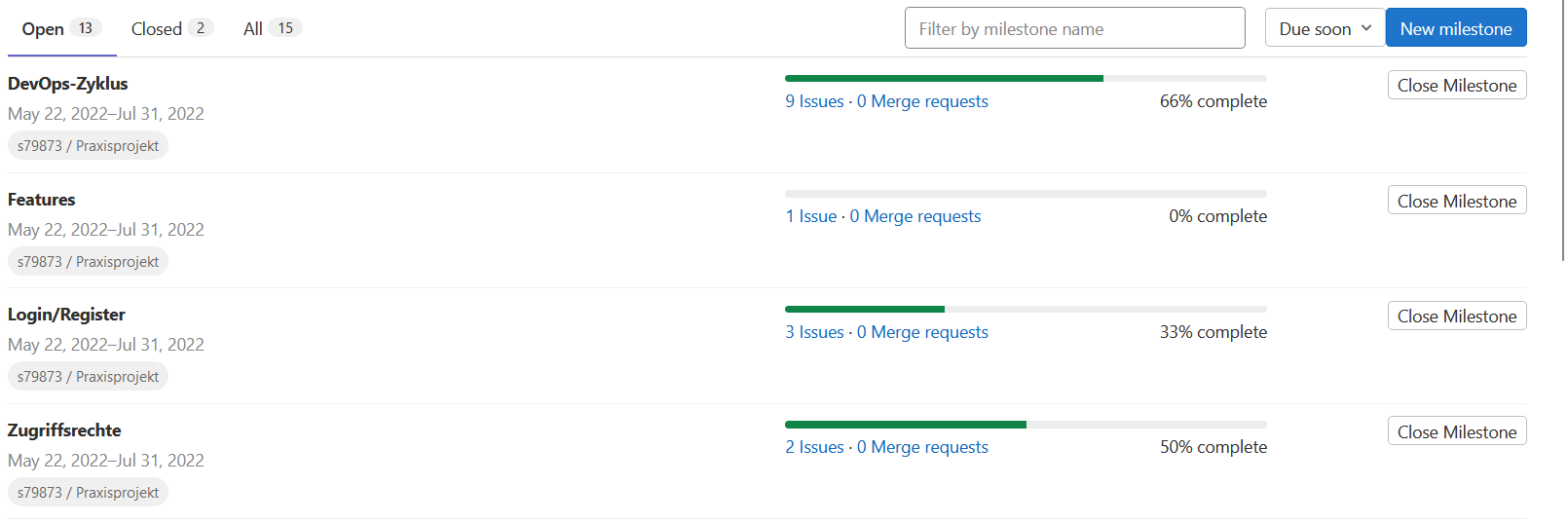
Zur Versionskontrolle kam die Software Gitlab zum Einsatz. GitLab ermöglicht über die Versionskontrolle hinweg die Option, Anforderungen und Meilensteine nachzuverfolgen.



**Abb. 5.12**: GitLab – Versionen

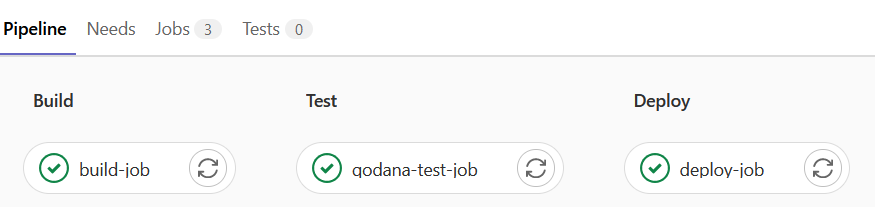


**Abb. 5.13**: GitLab – Anforderungen



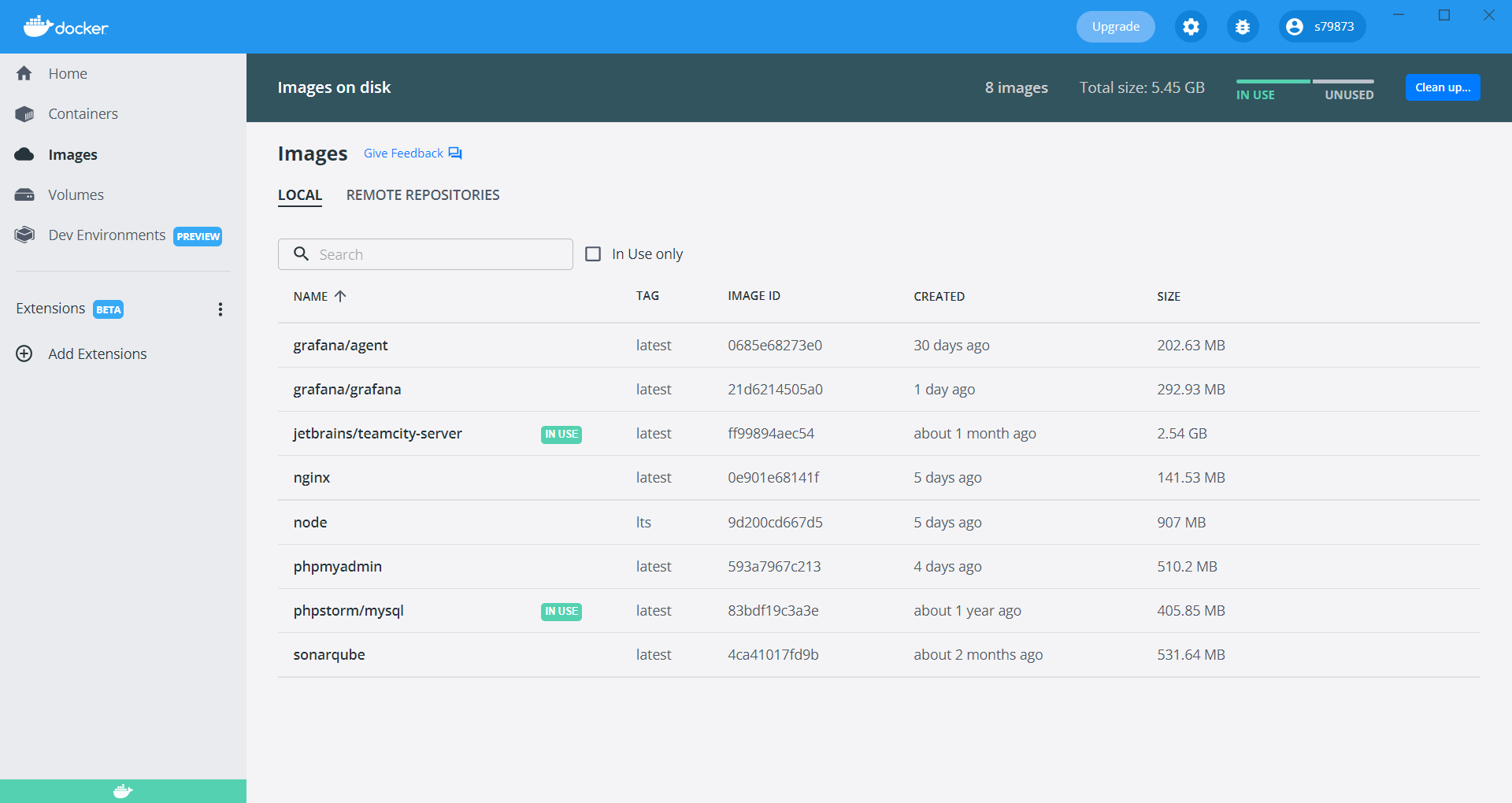
**Abb. 5.14**: GitLab – Meilensteine

Bei jeder Änderung, die in GitLab gepusht wird, findet über die Continuous Integration/Continuous Deployment-Pipeline (CI/CD-Pipeline) eine Prüfung des Codes statt. Das aktuelle Projekt wird dabei aufgebaut, getestet und testweise bereitgestellt.



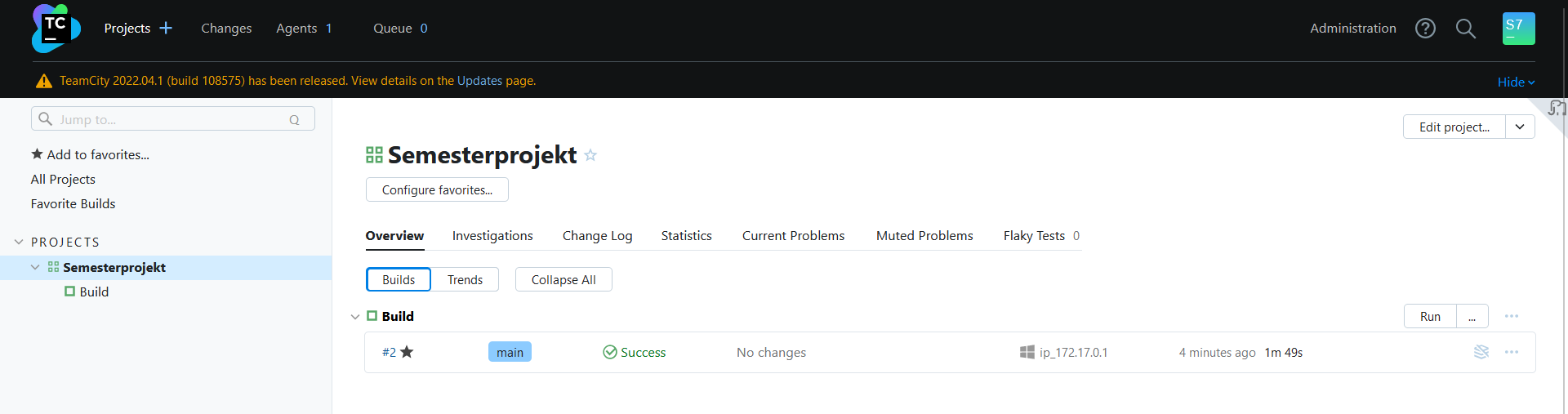
**Abb. 5.15**: GitLab – CI/CD-Pipeline

Docker sorgt für eine Verknüpfung des Projektes über GitLab hinweg. So werden Docker-Container genutzt, um bei jeder Änderung in GitLab zusätzliche Prüfungen der Qualität des Codes anzustoßen und diese zu überwachen.

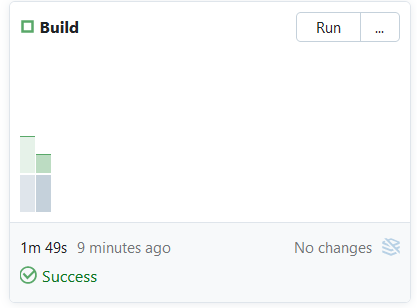


**Abb. 5.16**: Docker-Container

Dabei realisiert der TeamCity-Server eine zusätzliche Code-Überprüfung , Ein Agent übernimmt in Docker den Aufbau der Anwendung. Die Prüfung des Codes erfolgt auf dem Server. TeamCity verfügt über weitere Funktionen, die zur Auswertung genutzt werden können. Hier kann unter anderem abgelesen werden, wie lange die Prüfung dauert und ob diese erfolgreich war.

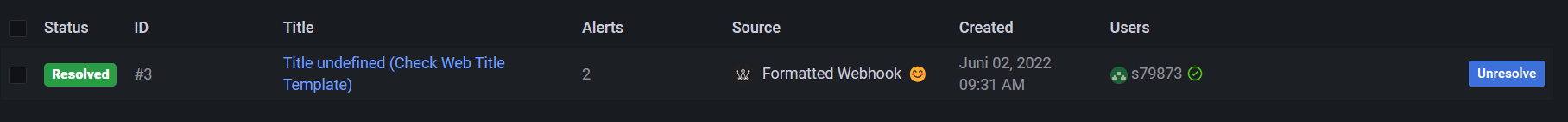


**Abb. 5.17**: TeamCity CI/CD-Pipeline



**Abb. 5.18**: TeamCity Auswertung

Grafana als weiterer Docker-Container überwacht die gesamten Prozesse. Hierüber können diverse Überwachungen durchgeführt werden. Wovon bisher erst die Warnungen implementiert sind.

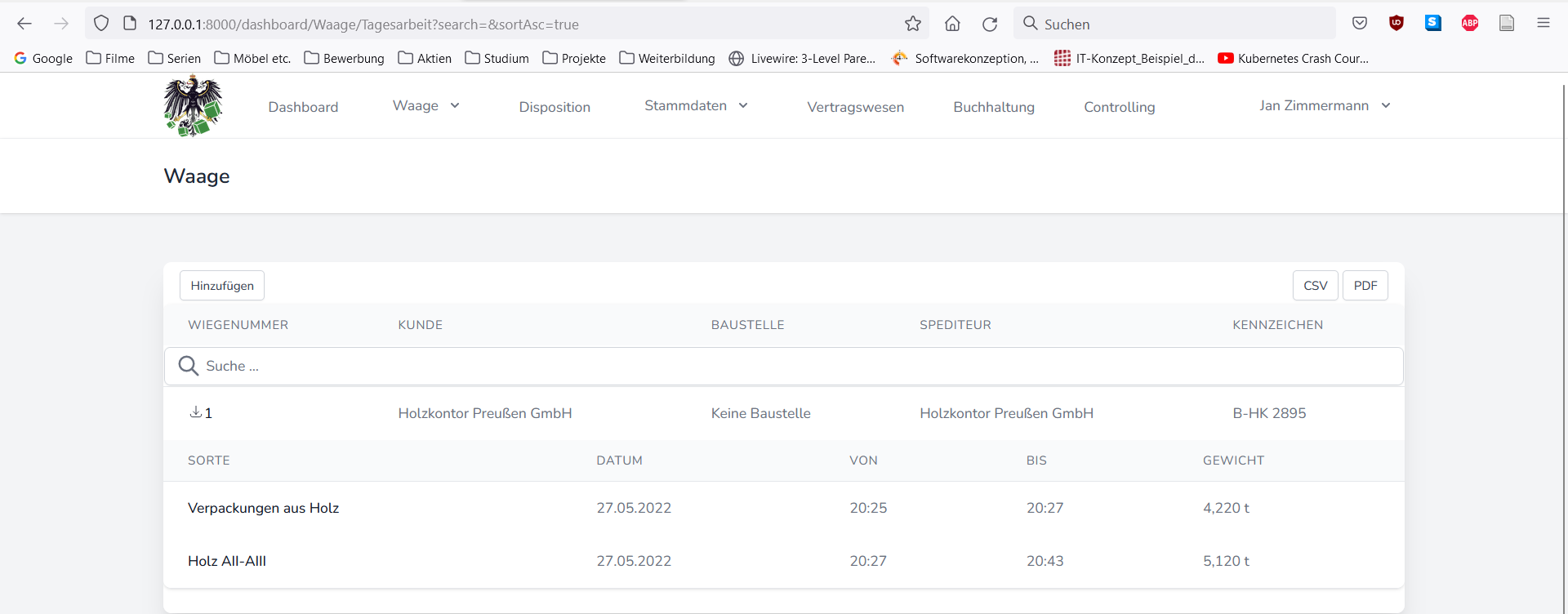


**Abb. 5.19**: Grafana Warnungen

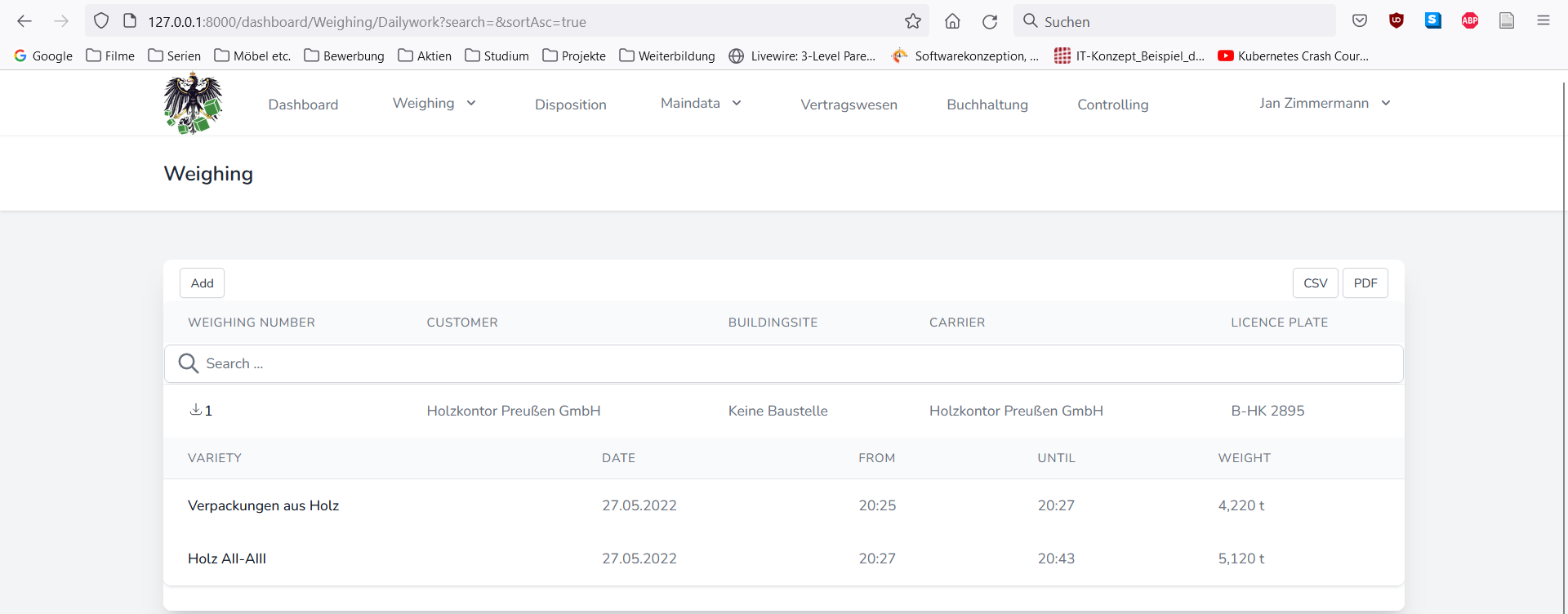
## Entwicklung

Dieser Abschnitt dokumentiert die Implementierungsphase deatilliert.

Laravel, Livewire und Jetstream bringen über die zusätzlichen Komponenten wie z. B. den Blade-Dateien, den Routes oder den Migrations sehr viel Dynamik in die Programmierung. Gerade die Migrations, Seeder und Trigger sorgen dafür, dass die Datenbank über das Repository hinaus jederzeit in einem neuen System implementiert werden kann. Jetstream bringt über die x-jet-Komponenten eine hohe Replizierbarkeit für einzelne Codeabschnitte mit. Dadurch ist es möglich, schnell eine Navigation aufzubauen, die verschiedene Gestaltungsoptionen ermöglicht. , So können z. B. gewählte Navigationspunkte hervorgehoben oder Informationen nagezeigt werden, wenn der Mauszeiger über einen Menüpunkt fährt. Livewire ist hervorragend geeignet, um Daten aus der Datenbank zu lesen oder diese zurückzuschreiben, da dies mitunter in Echtzeit passieren kann, was für Such- oder Sortierfunktionen sehr hilfreich ist. Dafür implementiert Livewire eine Verknüpfung von Komponenten zu Blade-Dateien, die nicht an die Routes gebunden sind. Die schnelle Anpassung von Sprachen erleichtert die Administration und die Anwendung des Kundenportals gleichermaßen. In der Konfiguration können Standardsprachen und dazugehörige Regionen (locations) konfiguriert werden. Anhand der IP-Adresse, die der Benutzer beim Aufruf des Kundenportals übermittelt, werden die Sprache der Anwendung und in der Adresszeile automatisch auf die zur IP-Adresse passenden Sprache umgestellt (Jede öffentliche IP-Adresse gehört einem bestimmten Adressbereich an, anhand dessen das zu diesem Adressbereich zugewiesene Land identifiziert werden kann).



**Abb. 5.20**: Kundenportal Lokalität DE

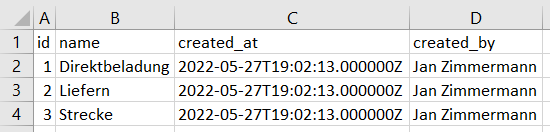


**Abb. 5.21**: Kundenportal Lokalität EN

In nahezu allen Bereichen der Webseite bietet sich die Möglichkeiten Daten als CSV zu drucken, dabei kann über die Such- und Filterfunktion das Ergebnis angepasst werden. Zusätzlich dazu wird Pagination genutzt, um die Ergebnisse auf der Seite auf zehn zu begrenzen.

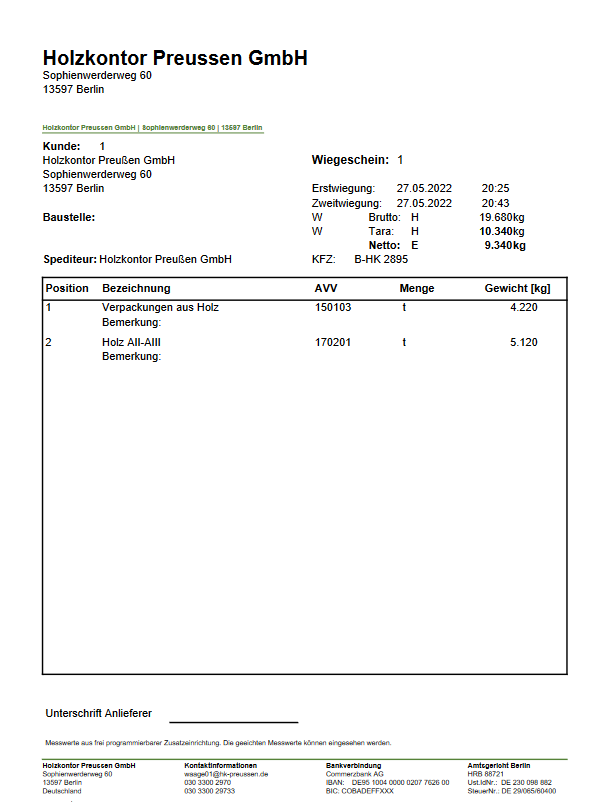


**Abb. 5.22**: Kundenportal Filter- und Suchfunktion sowie Option CSV/PDF



**Abb. 5.23**: Kundenportal Ausgabe als CSV

Dabei ist der Seitenaufbau der einzelnen Module nahezu identische gestaltet. Es sind Such- und Sortierfunktionen implementiert. Im rechten oberen Rand befindet sich die Möglichkeit, alle zu dem Zeitpunkt gefilterten Daten als CSV auszugeben, die PDF-Funktion ist derzeit noch nicht implementiert in diesem Zusammenhang. Links oberhalb der Tabelle können neue Eingaben getätigt werden und mit Klick auf die ID innerhalb der Tabelle, kann dieser Eintrag verändert werden. Links neben der Zahl gibt es die Möglichkeit diesen Eintrag als PDF herunterzuladen. Eine nähere Auflistung der Webseite sowie einige Codebeispiele befinden sich im Anhang (C).



**Abb. 5.24**: Kundenportal Beispiel neuer Wiegebeleg

# Projektabschluss und Fazit

Das nun folgende Fazit wird in zwei Teilbereiche getrennt. Beginnend kommt es zum Aufzeigen des aktuellen Projektstatus, darüber hinaus wird ein Ausblick auf eine darauffolgende Bachelorarbeit gegeben. Im Anschluss wird ein persönliches Fazit gezogen.

Nach zwölf Wochen, in denen sich umfangreich mit dem Thema DevOps, ERP-System und der Programmierung auseinandergesetzt wurde, obwohl kaum Erfahrung in diesen Bereichen bestanden, konnten schon viele Meilensteine auf dem Weg zum fertigen Kundenportal umgesetzt und neue Erfahrungen gesammelt werden, welche HKP definitiv neue Möglichkeiten eröffnen. Nichtsdestotrotz ist es noch ein weiter Weg zur vollständigen Implementierung des Kundenportals. Dieses Projekt dient dabei als gute Grundlage für die anschließende Bachelorarbeit, in der noch weitere Punkte umgesetzt werden sollen. Zum einen wird versucht, ein Optical Character Recognition (OCR) zu implementieren, um Lieferscheine von Kunden auslesen zu können zum anderen gibt dies noch weitere Optionen, wie z B. den gesamten Waageprozess anders zu gestalten. Inwieweit dies jedoch umgesetzt werden kann, wird sich während der Bachelorarbeit herausstellen.

Mir persönlich hat dieses Projekt sehr viel Freude bereitet und ich konnte mich an vielen neuen umfangreichen und abwechslungsreichen Aufgaben probieren. Dabei fand ich besonders spannend, das erlernte Wissen aus den verschiedenen Modulen sinnvoll miteinander verknüpfen und nutzen zu können. Erst in diesem Semester habe ich so richtig begonnen in PHP zu programmieren und der Fortschritt, den ich dabei erzielen konnte, war enorm, was einen Schub für die Motivation zur Folge hatte. Der Aufbau des DevOps-Zyklus sowie die Nutzung der IDE PhpStorm und des Frameworks Laravel stellten dabei eine besondere Herausforderung dar, weil die Einarbeitung und das Einlesen besonders schleppend verliefen. Dieses schleppende Anlernen zeigte sich auch in diesem Dokument, denn es viel mir grundsätzlich nicht leicht, einen derart langen zusammenhängenden Text zu formulieren und meine Aussagen anhand von Zitaten zu belegen. Besonders in den Abschnitten des DevOps und der IDE konnte ich dies persönlich feststellen.

Der Zugewinn an Erfahrung überwiegt dabei gegenüber den Widrigkeiten, die ich während der Projektphase hatte und ich denke, dass ich noch langfristig von dieser Projektphase profitieren werde.

Anhang

Fragebögen

Wie erwähnt können die Fragebögen mit den jeweiligen Antworten der Fachabteilungen hier eingesehen werden.

Tab. A.1: Fragebogen – Annahme – Pamela Dziecol 11.04.2022

|  |  |
| --- | --- |
| Frage | Antwort |
|  | Auflistung der der Wiegebelge und der einzelnen Sorten/Positionen, Ausdruck als PDF, HKP soll bearbeiten können |
|  | Kunden nur eigene Wiegebelege, HKP alle Wiegebelege |
|  | Wiegebelegnummer, Sorte, Kunde, Baustelle, Auftragsnummer |
|  | Kraftfahrer (eigene), Disponenten (alle), Buchhaltung und Betriebsleitung |
|  | Wiegebeleg anlegen, bearbeiten und eine Tagesliste |
|  | An dem aktuellen orientieren |
|  | Siehe oben |
|  | Ausdruck nur der eigenen Datensätze, dann ja |
|  | Nur relevante Daten, also keine Schlüsselwerte, sondern eher Kundenname etc. |

Tab. A.2: Fragebogen – Disposition – Reiner Uderhardt 12.04.2022

|  |  |
| --- | --- |
| Frage | Antwort |
|  | Anmelden/Beantragen von Container Tausch, Stellen, Abholen  Anlegen von Lieferscheinen um als Selbstanlieferer Material anliefern zu können |
|  | Aufstellung der aktuellen Container auf dem Gelände der Kunden, die Logik muss stimmen |
|  | Baustellen und Sorten |
|  | Disponent und Kraftfahrer, vielleicht Betriebsleiter je nach Größe des Unternehmens und der Struktur (HKP Kraftfahrer mehr Berechtigung) |
|  | Containeraktion, Containerplanung, vielleicht ein Kalender, um die Aufträge zuordnen zu können, im Kopf die Fahrzeuge und dann links die Uhrzeiten und die Fahraufträge reinschieben |
|  | Nur für Lieferschein relevant, Darstellung wie Wiegebeleg nur angepasst an den Lieferschein |
|  | Siehe oben |
|  | Für HKP relevant, aber Kunden sollen das eher nicht sehen |
|  | Je mehr desto besser, kann im Nachhinein ausgewertet werden |

Tab. A.3: Fragebogen – Einkauf/Verkauf – Michael Ludwig 13.04.2022

|  |  |
| --- | --- |
| Frage | Antwort |
|  | Preise anzeige, suchen, anfragen und ausdrucken |
|  | Nur eigene ganz wichtig, Preise dürfen niemals Kundenübergreifend ersichtlich sein!!! |
|  | Sortennummer, Baustelle, da Preise für Baustellen unterschiedlich sein können |
|  | Nur Vertriebler und Betriebsleiter |
|  | Noch nicht sicher vielleicht für HKP anlegen |
|  | Unser Kopfbogen mit den Daten zu den Sorten und Preisen |
|  | Frage entfernt |
|  | Nur eigene Daten dann ist der Export ok |
|  | Weniger ist besser, nur der Preis, die Sorte und Baustelle und vielleicht die Laufzeit |

Tab. A.4: Fragebogen – Buchhaltung – Karina Herzog 14.04.2022

|  |  |
| --- | --- |
| Frage | Antwort |
|  | Rechnungen müssen selbstständig rausgesucht werden können, als PDF ausdrucken, wenn möglich |
|  | Rechnungen, wichtig, nicht jeder darf Zugriff auf Rechnungen haben |
|  | Kunde, Adresse etc. Sorten Baustelle, Summe der Rechnung und Rechnungsdatum |
|  | Buchhaltung oder Betriebsleitung, bei Dispositionsrechnung Dispo |
|  | Eigentlich nur Rechnung suchen |
|  | Siehe aktuelle Rechnung |
|  | Frage entfernt |
|  | Wenn die Kunden dann besser vergleichen können, wäre dies gut, weil dann weniger Anrufe kommen, ist auch wichtig fürs Steuerbüro |
|  | Ähnlich wie bei der Suchfunktion |

Tab. A.5: Fragebogen – Controlling – Heiko Schurwin 14.04.2022

|  |  |
| --- | --- |
| Frage | Antwort |
|  | Mehr Auswertungen sind immer gut, wir sind ein Dienstleister und bieten deshalb den besten Service, auch wichtig für Abfallstatistik etc. die Auflagen für Entsorgungsfachbetriebe sind sehr hoch |
|  | Betriebsleitung alle, ansonsten nur für deren Bereich relevante Daten |
|  | Eher welche Daten sollen angezeigt werden, wie soll der Aufbau sein |
|  | Muss je nach Abteilung eingestellt werden, der eine soll mehr Auswertungen erhalten, der andere weniger |
|  | Gesplittet nach Abteilung oder Funktion der Datensätze, welche Daten sind relevant für die Abfallstatistik, Auswertung über Sortierung etc. alle Daten, welche die Behörde einfordert in einen Bereich, ein klick alles runterladen |
|  | Aufbereitet für die Behörde oder bei einer Prüfung oder Verlängerung der des Zertifikats als Entsorgungsfachbetrieb |
|  | Frage entfernt |
|  | Ja definitiv, HKP gibt diese auch in der Form bei der Behörde ab. |
|  | Relevante Daten für die Behörde |

Mockups

Wie beschrieben sind die übrigen Mockups hier aufzufinden. Des Weiteren besteht die Möglichkeit, Zugriff auf das verbundene GitLab zu erhalten: <https://gitlab.bht-berlin.de/s79873/praxisprojekt>

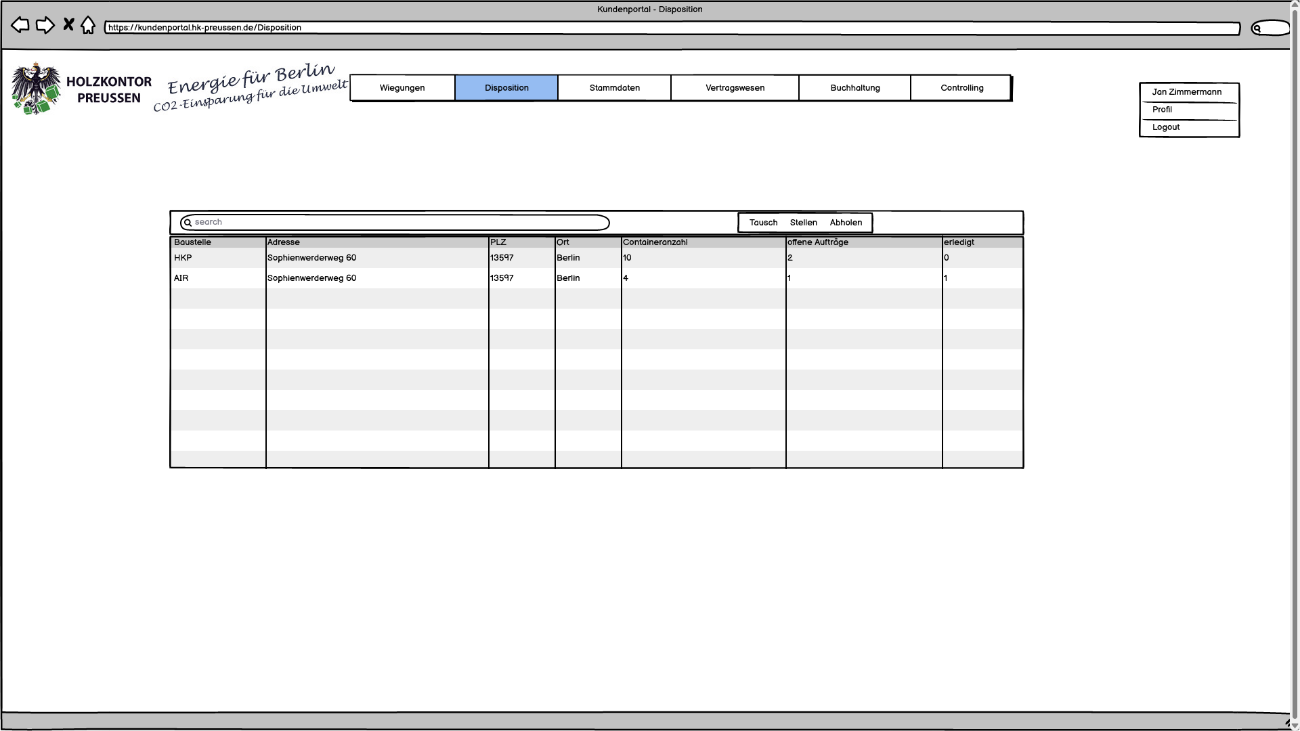


Abb. B.1: Mockup – Disposition

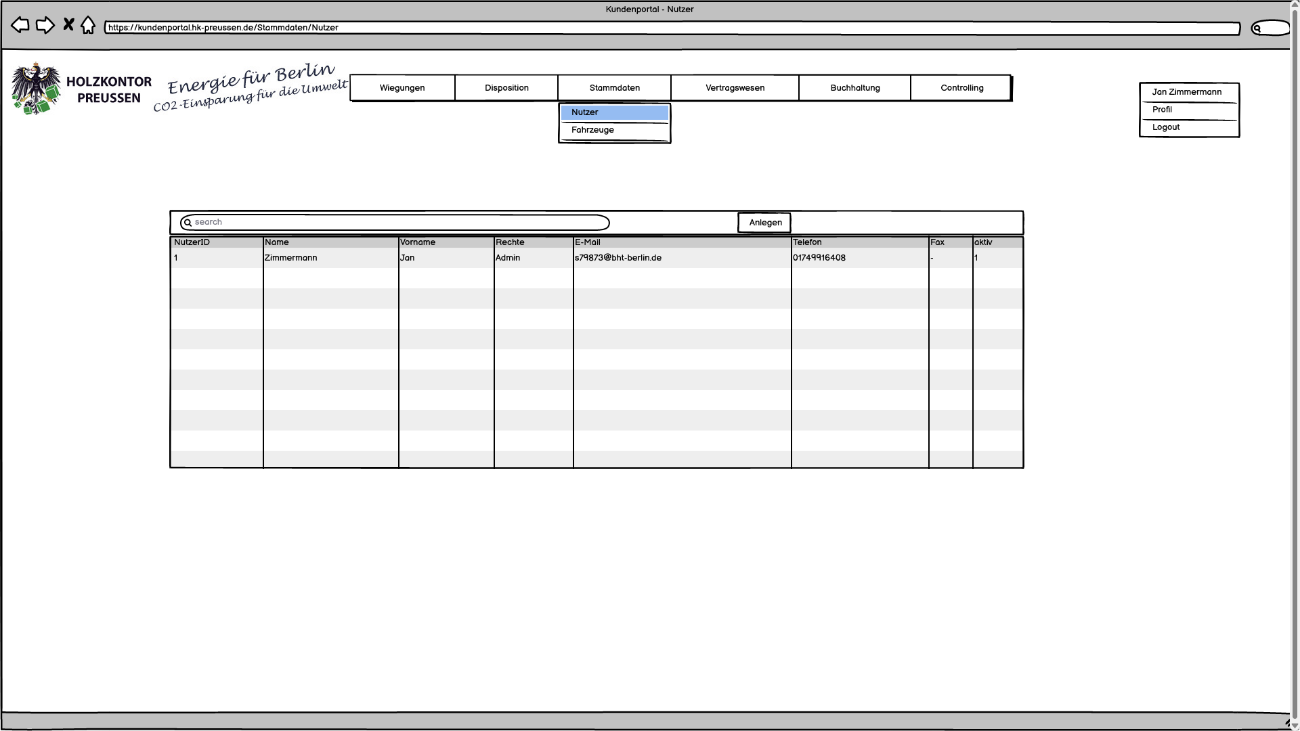


Abb. B.2: Mockup – Nutzer

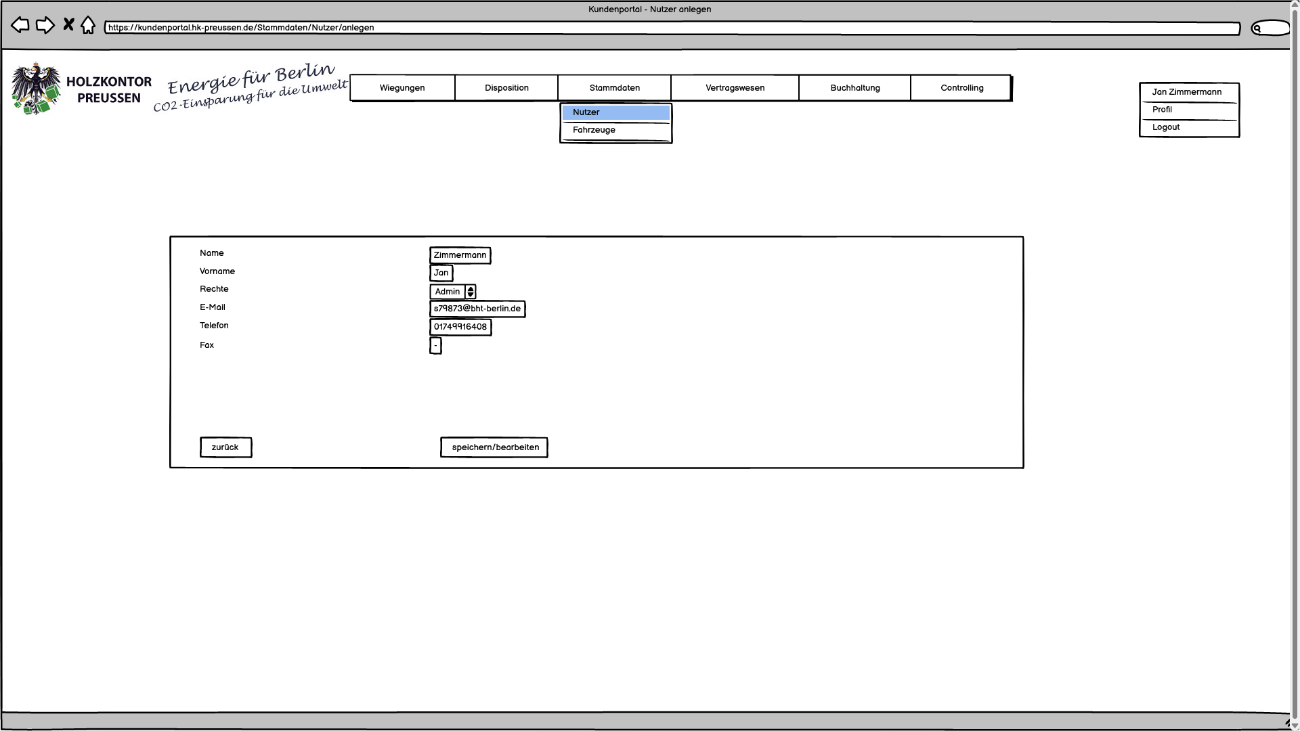


Abb. B.3: Mockup – Nutzer anlegen

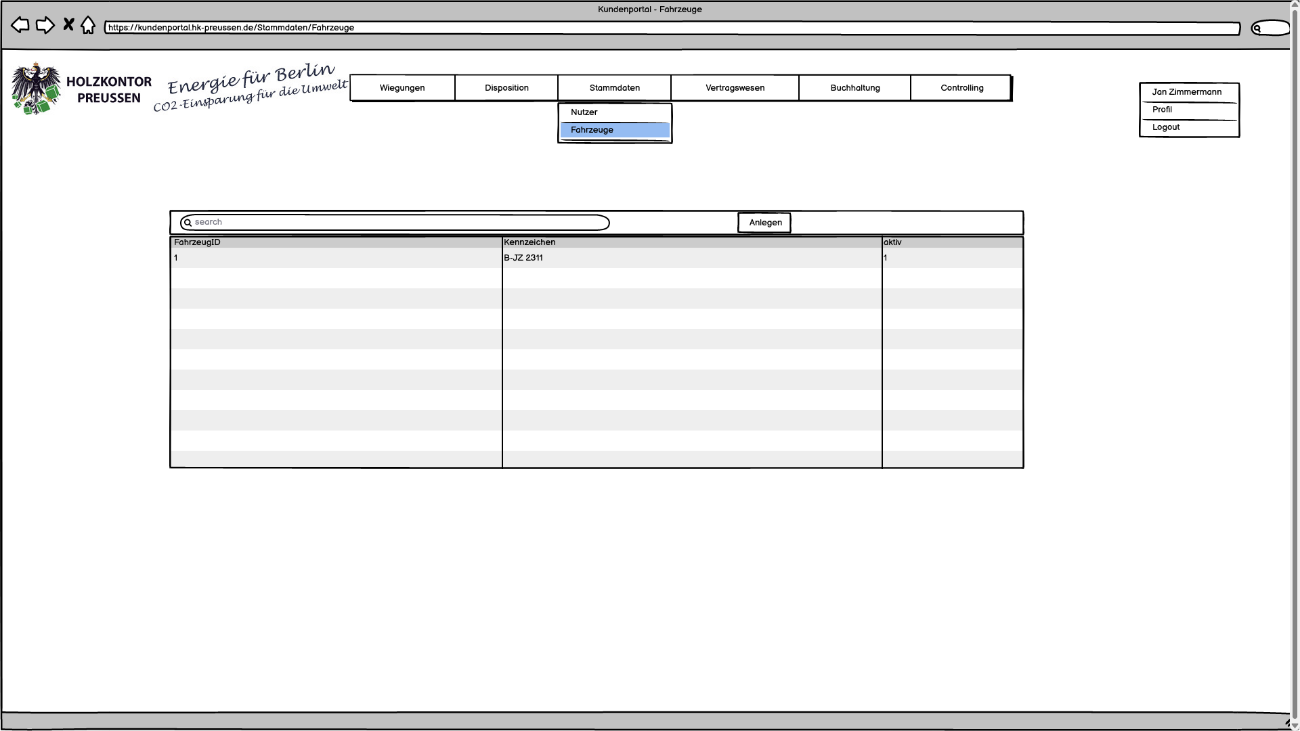


Abb. B.4: Mockup – Fahrzeuge

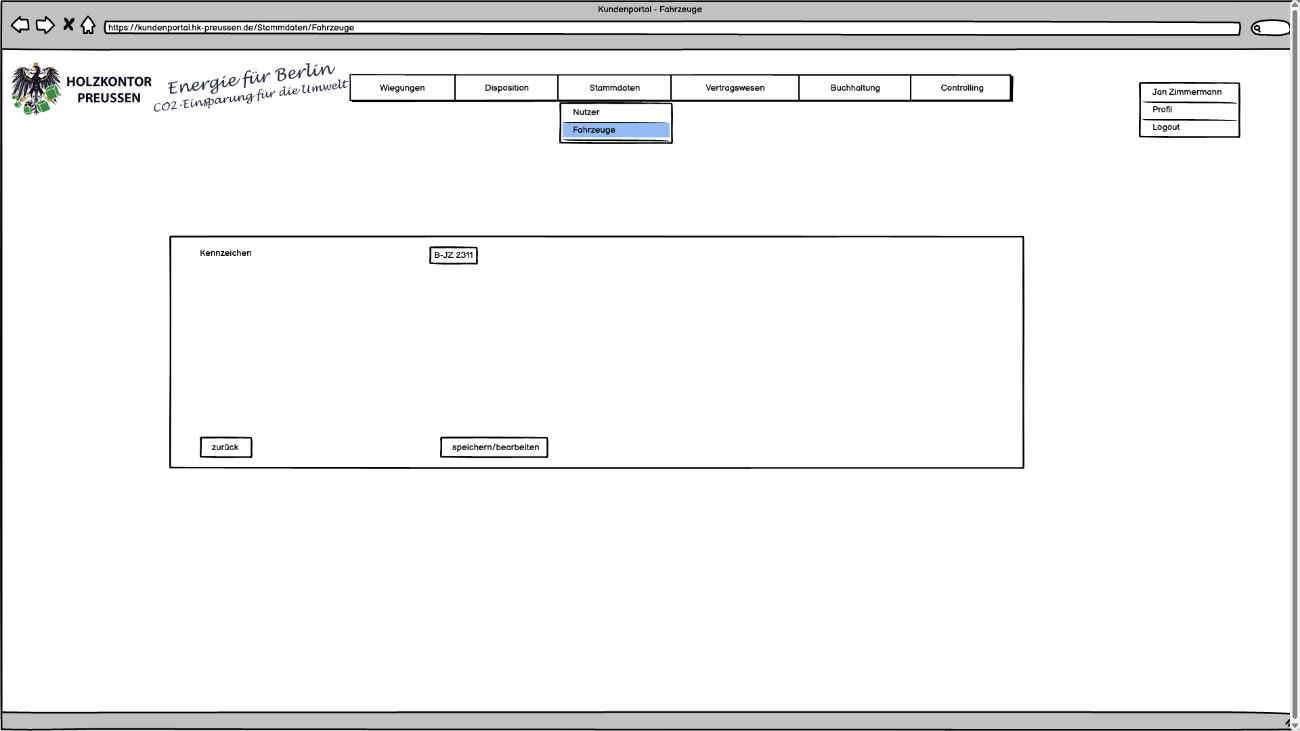


Abb. B.5: Mockup – Fahrzeuge anlegen

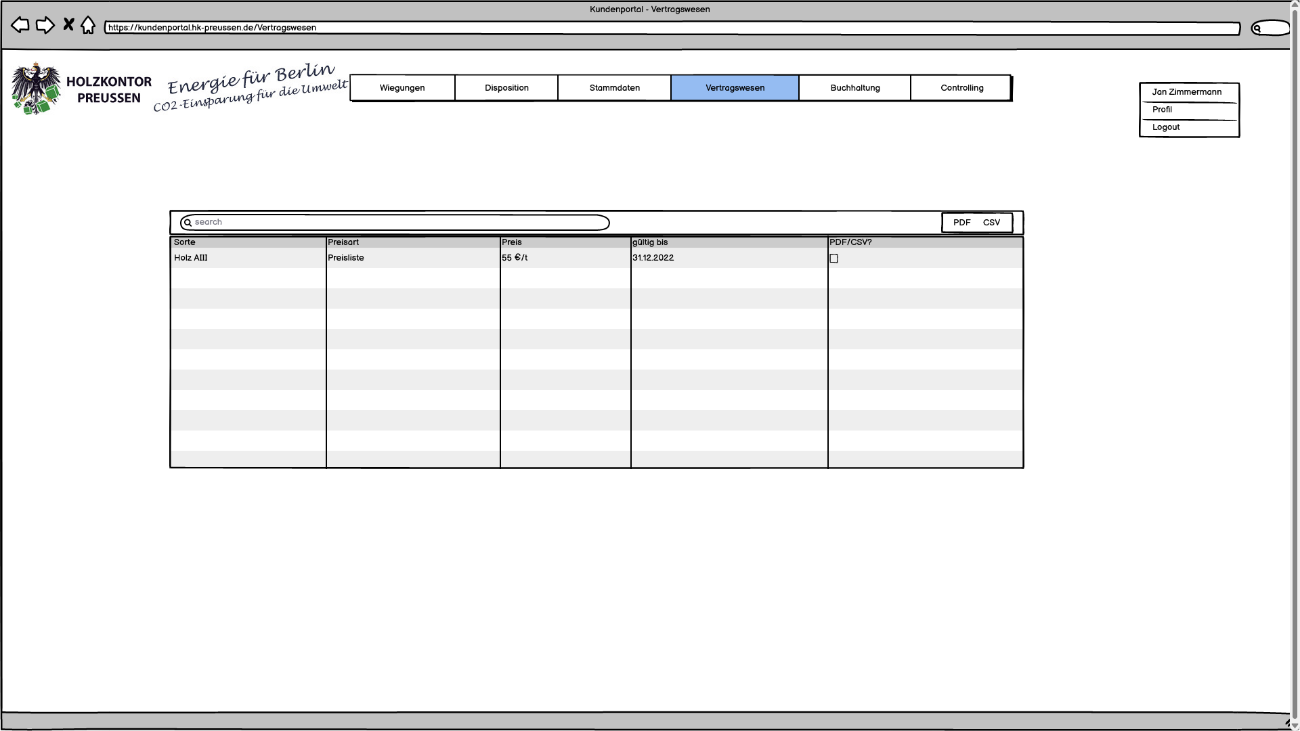


Abb. B.6: Mockup – Vertragswesen

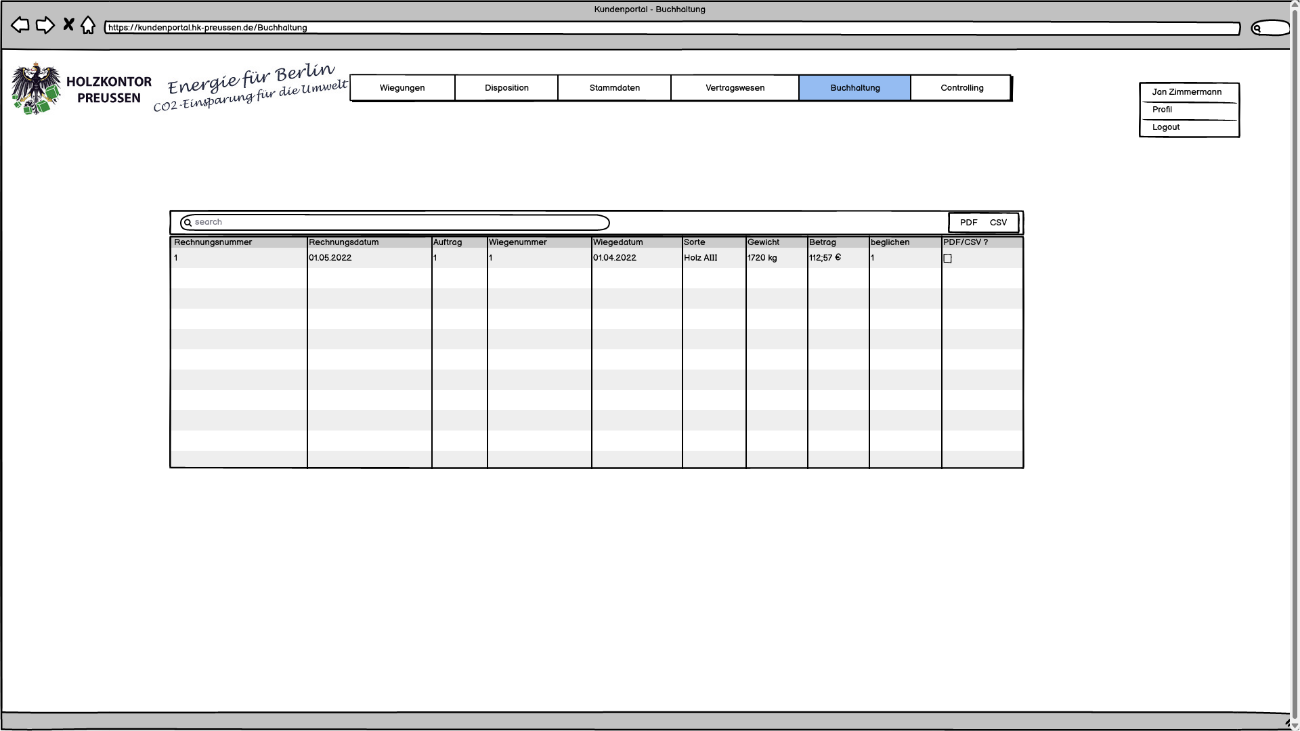


Abb. B.7: Mockup – Buchhaltung

Kundenportal

Wie beschrieben findet nachfolgend eine nähere Auflistung des Kundenportals statt. Darüber hinaus werden einige Codebeispiele gegeben. Des Weiteren besteht die Möglichkeit, Zugriff auf das verbundene GitLab zu erhalten: <https://gitlab.bht-berlin.de/s79873/praxisprojekt>

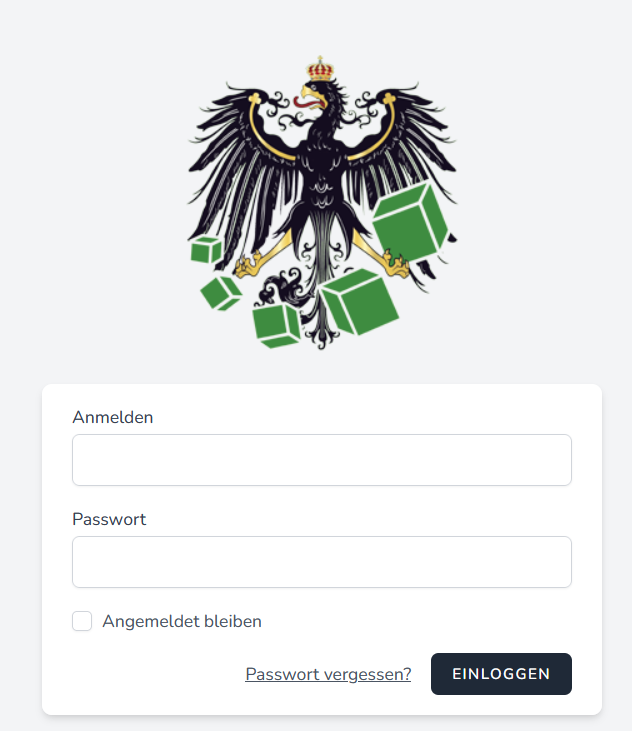


Abb. C.1: Kundenportal – Loginseite

Die Ansicht der Nutzer ist hierbei beispielhaft angegeben. Diese Ansichten finden sich auch bei den übrigen Unterseiten.

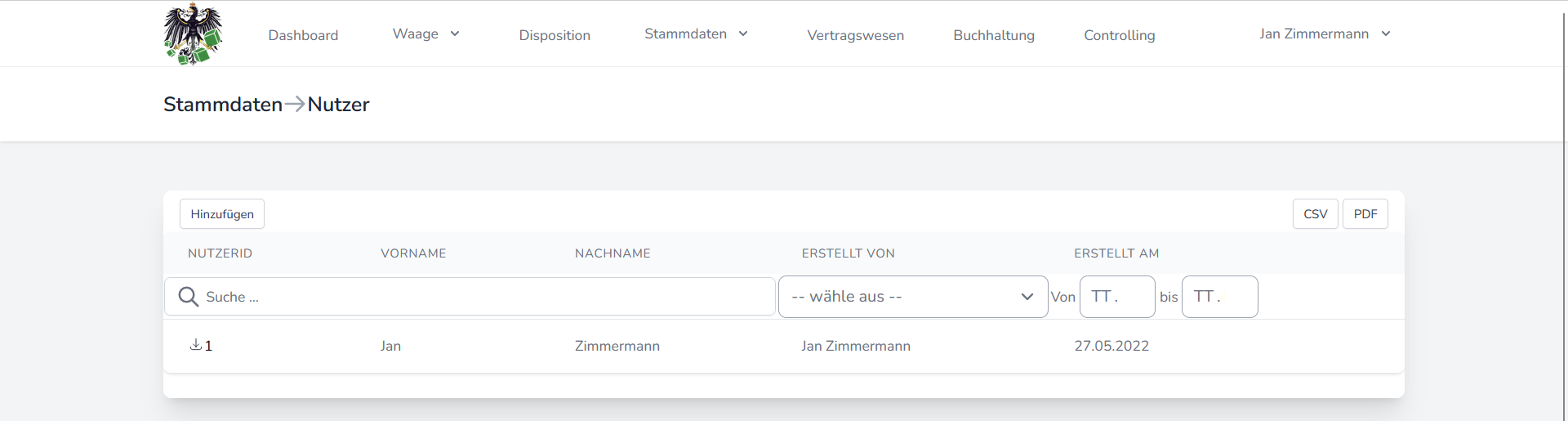


Abb. C.2: Kundenportal – Nutzer anzeigen

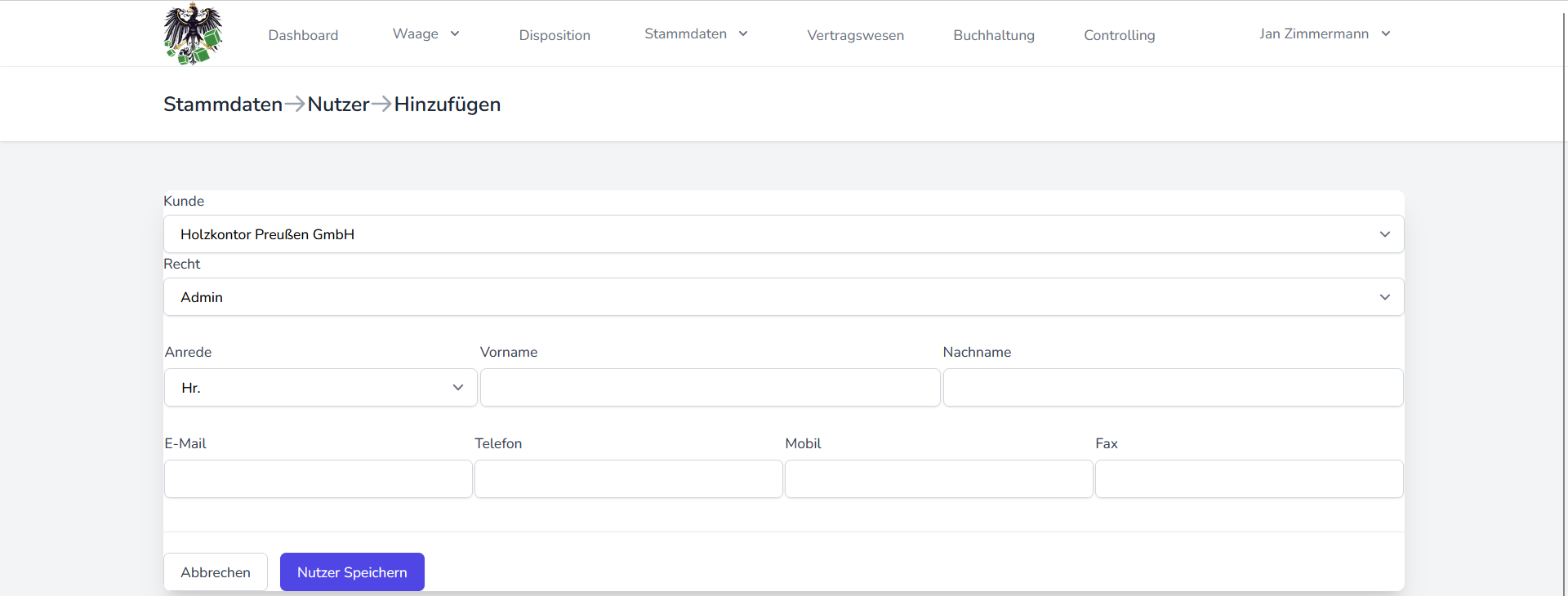


Abb. C.3: Kundenportal – Nutzer erstellen

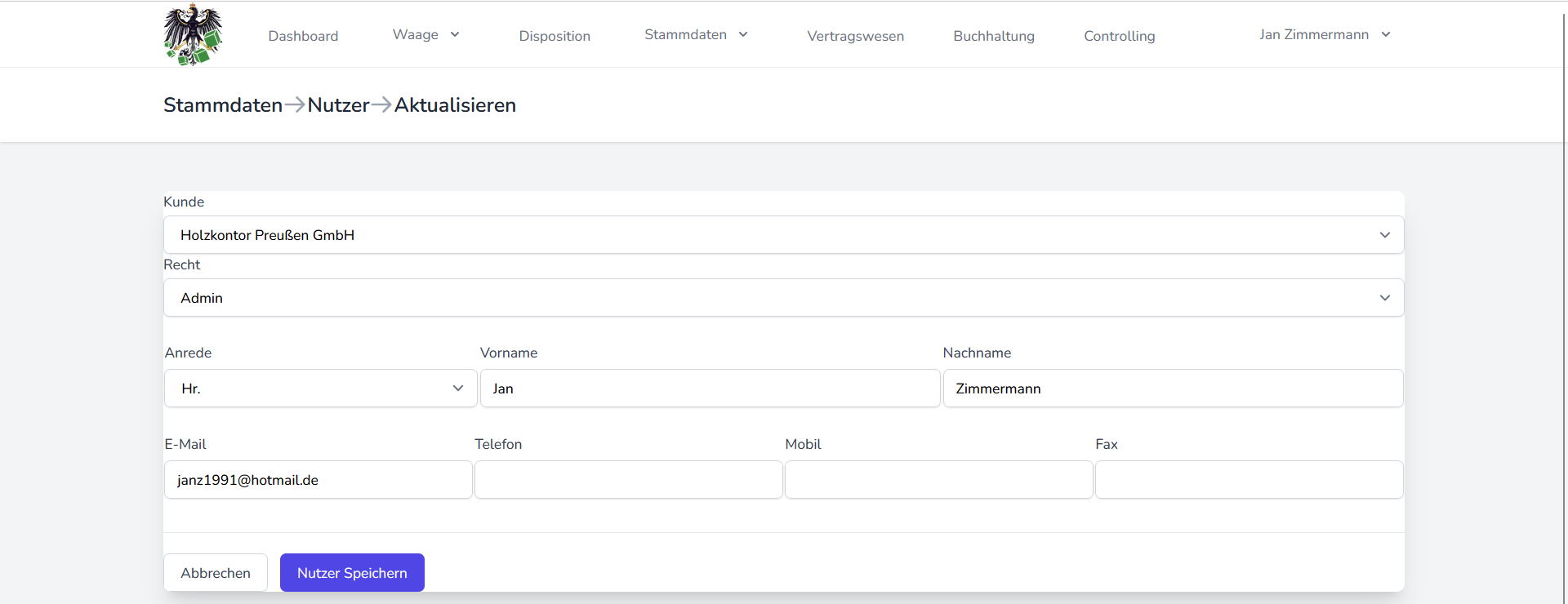


Abb. C.4: Kundenportal – Nutzer bearbeiten

Eine Ausnahme hierbei bilden die Kunden, da zu jedem Kunden Baustellen, Nutzer und Kennzeichen zugeordnet sind. Wenn also ein Kunde bearbeitet wird, so findet auch eine Auflistung der jeweiligen Baustellen, Nutzern und Kennzeichen statt, welche hierbei wiederum bearbeitet und angelegt werden können.

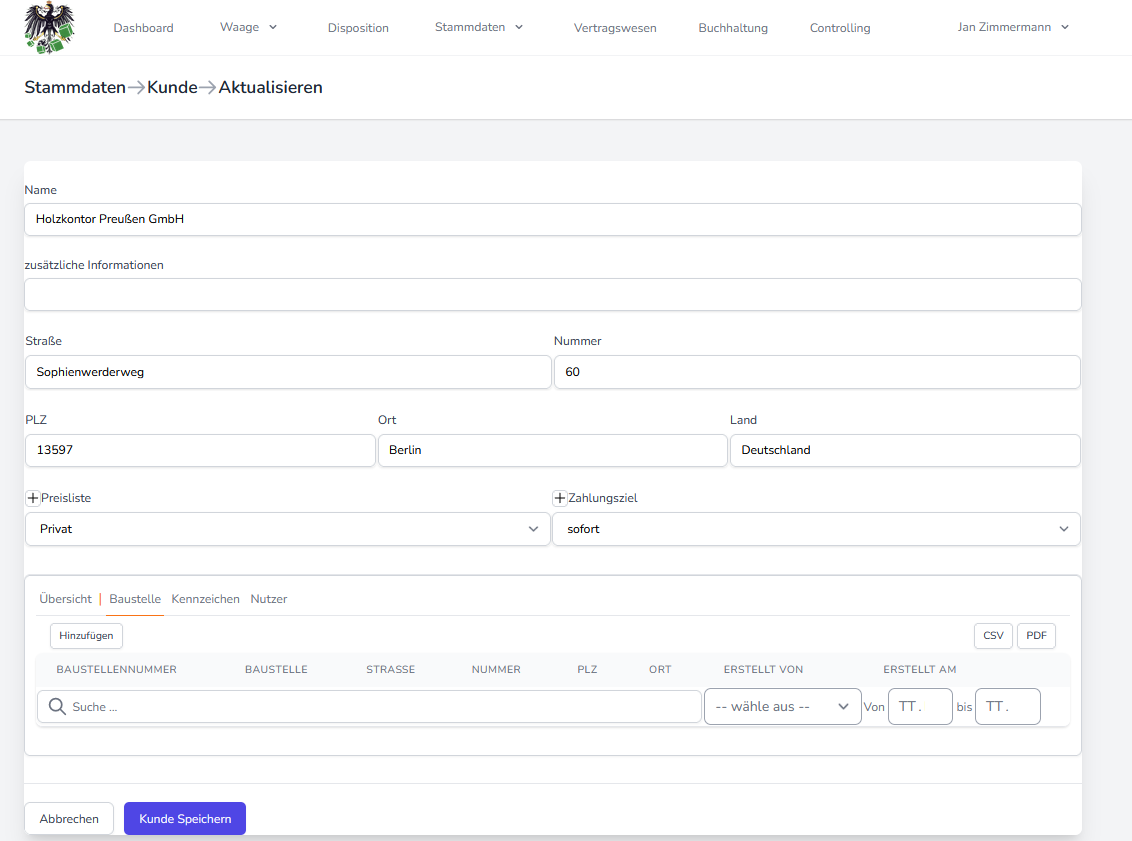


Abb. C.5: Kundenportal – Kunden bearbeiten

Im Anhang soll nun noch ein wenig auf die Programmierung der einzelnen Komponenten eingegangen werden.

Jeder Tabelleneintrag in der Datenbank hat ein Pendant in den Models.

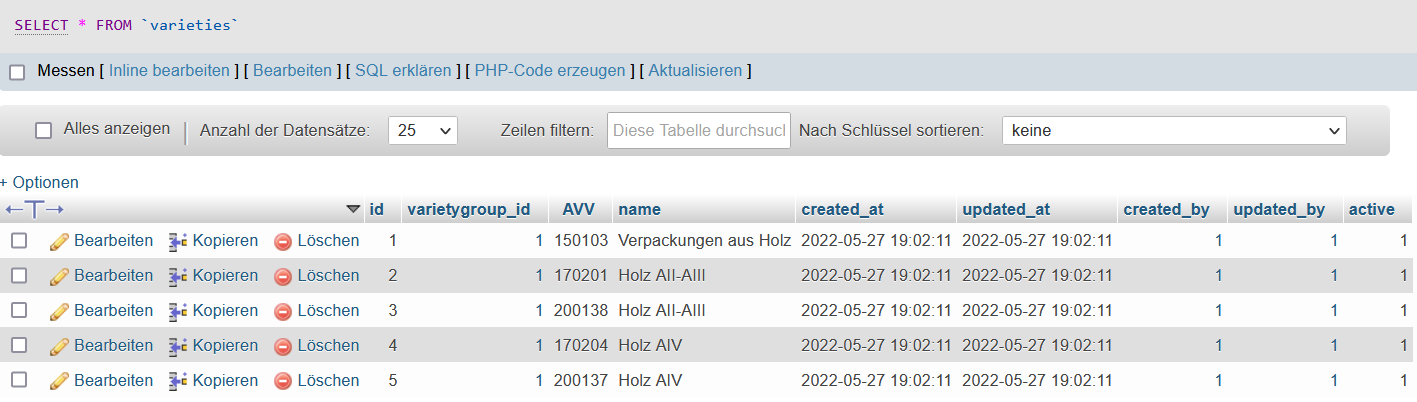


Abb. C.6: Tabelle – Sorten

<?php  
  
namespace App\Models;  
  
use Illuminate\Database\Eloquent\Factories\HasFactory;  
use Illuminate\Database\Eloquent\Model;  
  
class Variety extends Model  
{  
 use HasFactory;  
  
 protected $fillable = [  
 'name',  
 'AVV',  
 'varietygroup\_id',  
 'created\_by',  
 'updated\_by',  
 'active'  
 ];  
  
 public function getcreator()  
 {  
 return $this->belongsTo(User::class, 'created\_by');  
 }  
  
 public function getgroup()  
 {  
 return $this->belongsTo(Varietygroup::class, 'varietygroup\_id');  
 }  
}

Es wird hierbei explizit aufgezeigt, welche Beziehungen zu anderen Tabellen/Models bestehen.

Für das Bearbeiten und Anlegen einer Sorte wird genau eine Komponente und eine Blade-Datei benötigt.

<?php  
  
namespace App\Http\Livewire\Maindata\Edit;  
  
use App\Models\Variety;  
use App\Models\Varietygroup;  
use Illuminate\Support\Facades\Auth;  
use Livewire\Component;  
  
class VarietyEditComponent extends Component  
{  
 public $values;  
 public $valuesID;  
 public $submitAction;  
 public $name = '';  
 public $AVV = '';  
 public $varietygroup\_id = 1;  
 public $varietygroup;  
 public $active = 1;  
  
 protected $rules = [  
 'name' => 'filled',  
 'AVV' => 'filled',  
 ];  
  
 protected $listeners = ['updateVarietygroup'];  
  
 public function updateVarietygroup()  
 {  
 return redirect(request()->header('Referer'));  
 }  
  
 public function mount(){  
 $this->varietygroup = Varietygroup::*all*();  
 if ($this->values) {  
 $this->valuesID = $this->values->id;  
 $this->name = $this->values->name;  
 $this->AVV = $this->values->AVV;  
 $this->varietygroup\_id = $this->values->varietygroup\_id;  
 }  
 }  
  
 public function submit()  
 {  
 if ($this->submitAction == "create") {  
 $this->validate();  
 $values = Variety::*create*([  
 'name' => $this->name,  
 'AVV' => $this->AVV,  
 'varietygroup\_id' => $this->varietygroup\_id,  
 'created\_by' => Auth::*user*()->id,  
 'updated\_by' => Auth::*user*()->id,  
 'active' => $this->active,  
 ]);  
 $values->save();  
 redirect(route('variety'));  
 } elseif ($this->submitAction == "update") {  
 $this->validate();  
 $values = Variety::*find*($this->valuesID);  
 $values->name = $this->name;  
 $values->AVV = $this->AVV;  
 $values->varietygroup\_id = $this->varietygroup\_id;  
 $values->updated\_by = Auth::*user*()->id;  
 $values->active = $this->active;  
 $values->save();  
 redirect(route('variety'));  
 }  
 }  
  
 public function render()  
 {  
 return view('livewire.maindata.edit.variety-edit-component');  
 }  
}

<div>  
 @livewire('maindata.edit.create.varietygroup-component')  
 <div class="space-y-8 divide-y divide-gray-200">  
 <div class="space-y-8 divide-y divide-gray-200">  
 <div>  
 <div class="mt-6 grid grid-cols-1 gap-y-6 gap-x-4 sm:grid-cols-6">  
 <div class="sm:col-span-6">  
 <label for="name" class="block text-sm font-medium text-gray-700"> {{ \_\_('Variety') }} </label>  
 <div class="mt-1 flex rounded-md shadow-sm">  
 <input wire:model.defer="name" type="text" name="name" id="name"  
 autocomplete="name"  
 class="shadow-sm focus:ring-indigo-500 focus:border-indigo-500 block w-full sm:text-sm border border-gray-300 rounded-md">  
 </div>  
 <x-jet-input-error for="name" class="mt-2"/>  
 </div>  
 <div class="sm:col-span-6">  
 <label for="AVV" class="block text-sm font-medium text-gray-700"> AVV </label>  
 <div class="mt-1 flex rounded-md shadow-sm">  
 <input wire:model.defer="AVV" type="text" name="AVV" id="AVV"  
 autocomplete="AVV"  
 class="shadow-sm focus:ring-indigo-500 focus:border-indigo-500 block w-full sm:text-sm border border-gray-300 rounded-md">  
 </div>  
 <x-jet-input-error for="AVV" class="mt-2"/>  
 </div>  
 <div class="sm:col-span-6">  
 <div class="flex">  
 <button wire:click="$emit('openModalVarietygroup')" class="border-solid border-2 border-gray-300 rounded">  
 <svg xmlns="http://www.w3.org/2000/svg" class="h-4 w-4" fill="none" viewBox="0 0 24 24"  
 stroke="currentColor">  
 <path stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M12 4v16m8-8H4"/>  
 </svg>  
 </button>  
 <label for="varietygroup\_id" class="block text-sm font-medium text-gray-700"> {{ \_\_('Varietygroup') }} </label>  
 </div>  
 <div class="mt-1 flex rounded-md shadow-sm">  
 <select wire:model.defer="varietygroup\_id" id="varietygroup\_id" name="varietygroup\_id"  
 class="shadow-sm focus:ring-indigo-500 focus:border-indigo-500 block w-full sm:text-sm border border-gray-300 rounded-md">  
 @foreach ($varietygroup as $group)  
 <option id="{{$group->id}}" value="{!! $group->id !!}">{!! $group->name !!}</option>  
 @endforeach  
 </select>  
 </div>  
 <x-jet-input-error for="varietygroup\_id" class="mt-2"/>  
 </div>  
 </div>  
 </div>  
 </div>  
 <div class="pt-5">  
 <div class="flex justify-start">  
 <a type="button" href="{{ route('variety') }}"  
 class="bg-white py-2 px-4 border border-gray-300 rounded-md shadow-sm text-sm font-medium text-gray-700 hover:bg-gray-50 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-indigo-500">  
 {{ \_\_('Cancel') }}  
 </a>  
 <button disabled wire:loading type="button"  
 class="ml-3 inline-flex justify-center py-2 px-4 border border-transparent shadow-sm text-sm font-medium rounded-md text-white bg-indigo-600 hover:bg-indigo-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-indigo-500">  
 {{ \_\_('Please wait ...') }}  
 </button>  
 <button wire:click="submit" wire:loading.remove type="button"  
 class="ml-3 inline-flex justify-center py-2 px-4 border border-transparent shadow-sm text-sm font-medium rounded-md text-white bg-indigo-600 hover:bg-indigo-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-indigo-500">  
 {{ \_\_('Variety') }} {{ \_\_('Save') }}  
 </button>  
 </div>  
 </div>  
 </div>  
</div>

Diese Beispiele für den Code können im GitLab näher beleuchtet werden. Die einzelnen Komponenten sind immer ähnlich aufgebaut. Exemplarisch habe ich eine Komponente und eine Blade-Datei aufgezeigt. Es gibt aber noch wesentlich mehr Dateien, die sich rein mit den Sorten befassen. Je Unterseite gibt es ca. sieben Dateien, um alle Ansichten, von der Übersicht in der Tabelle, über das Anlegen bis hin zur Bearbeitung darzustellen.

Literaturverzeichnis

Adamu, J., Hamzah, R., & Rosli, M. M. (2020). Security issues and framework of electronic medical record: A review. *Bulletin of Electrical Engineering and Informatics*, *9*(2). https://doi.org/10.11591/eei.v9i2.2064

Alt, R., Auth, G., & Kögler, C. (2017). *Innovationsorientiertes IT-Management mit DevOps*. Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-18704-0

Cepuc, A., Botez, R., Craciun, O., Ivanciu, I.-A., & Dobrota, V. (2020). Implementation of a Continuous Integration and Deployment Pipeline for Containerized Applications in Amazon Web Services Using Jenkins, Ansible and Kubernetes. *2020 19th RoEduNet Conference: Networking in Education and Research (RoEduNet)*, 1–6. https://doi.org/10.1109/RoEduNet51892.2020.9324857

Dobslaw, F., Persson, T., & Wikhög, M. (2018). *MID SWEDEN UNIVERSITY*. 48.

Eltahawey, A. O. (2016). *Hyper Text Markup Language HTML: A Tutorial*. 26.

Halstenberg, J., Pfitzinger, B., & Jestädt, T. (2020). *DevOps: Ein Überblick*. Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-31405-7

Katal, A., Bajoria, V., & Dahiya, S. (2019). DevOps: Bridging the gap between Development and Operations. *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, 1–7. https://doi.org/10.1109/ICCMC.2019.8819631

Laaziri, M., Benmoussa, K., Khoulji, S., & Kerkeb, M. L. (2019). A Comparative study of PHP frameworks performance. *Procedia Manufacturing*, *32*, 864–871. https://doi.org/10.1016/j.promfg.2019.02.295

Nordhausen, T., & Hirt, J. (2020, Oktober 13). *Manual zur Literaturrecherche in Fachdatenbanken* [Publikation]. Manual zur Literaturrecherche in Fachdatenbanken. https://www.researchgate.net/profile/Julian-Hirt/publication/345700600\_RefHunter\_Manual\_zur\_Literaturrecherche\_in\_Fachdatenbanken\_Version\_50/links/5fab1f49a6fdcc331b94483a/RefHunter-Manual-zur-Literaturrecherche-in-Fachdatenbanken-Version-50.pdf

Pekša, J. (2018). Extensible Portfolio of Forecasting Methods for ERP Systems: Integration Approach. *Information Technology and Management Science*, *21*, 64–68. https://doi.org/10.7250/itms-2018-0010

Rad, B. B., Bhatti, H. J., & Ahmadi, M. (2017). *An Introduction to Docker and Analysis of its Performance*. 9.

Schoeneberg, K.-P. (2011). *Kritische Erfolgsfaktoren von IT-Projekten. Eine empirische Analyse von ERP-Implementierungen am Beispiel der Mineralölbranche*. Rainer Hampp Verlag. https://doi.org/10.1688/9783866187122

Than, P. P., & Phyu, M. P. (2019). Continuous integration for Laravel applications with GitLab. *Proceedings of the International Conference on Advanced Information Science and System*, 1–6. https://doi.org/10.1145/3373477.3373479

Töpfer, A. (2020). *Strategische Positionierung und Kundenzufriedenheit: Anforderungen – Umsetzung – Praxisbeispiele*. Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-32019-5

Vadivelu, K., Balaji, N., Poongavanam, N., Tamilselvan, S., & Rajakumar, R. (2018). *CLOUD - ERP: IMPLEMENTATION STRATEGIES, BENEFITS AND CHALLENGES*. 6.

Valdivia, R. G. B. (2019). *Collaborative Learning Using Git with GitLab in Students of the Engineering Programming Course*. 10.

Abschließende Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig, ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Berlin, den 30. Juni 2022