



Schriftliche Ausarbeitung

Dokumentation Anwendung „Digitaler Briefkasten“
im Rahmen des Modul „AWE1“

Prüfer:

Christian Heuermann

Erstellt von:

Jonathan Brockhausen, Phillip Röring, Julius Figge

Studiengang:

Angewandte Informatik B.Sc.

Eingereicht am:

8. Juni 2020

Inhaltsverzeichnis

Abbildungsverzeichnis	IV
Tabellenverzeichnis	V
Listingverzeichnis	VI
1 Installation [Jonathan Brockhausen]	1
2 Fachkonzept [Jonathan Brockhausen]	3
2.1 Technologien	3
2.2 Frameworks	4
2.3 Datenbank	4
3 Entity-Relationship Diagramm [Jonathan Brockhausen]	5
4 Projekt-Architektur [Philipp Röring]	7
4.1 Klassendiagramm	9
5 Abläufe [Philipp Röring]	10
6 Schnittstellen [Philipp Röring]	11
6.1 Schnittstellenbeschreibung REST-API	11
6.2 Login	13
6.3 Webschnittstelle	13
7 Security [Julius Figge]	14
7.1 Aktive Security Bestandteile	14
7.2 Passive Security Bestandteile	15
8 Test	16
8.1 Testklassen [Jonathan Brockhausen]	16
8.2 manuelle-„Klicktests“ [Julius Figge]	17
9 Use-Cases [Julius Figge]	18
10 GUI-Konzept [Julius Figge]	20
11 Konzepte [Julius Figge]	23
11.1 Arbeitskonzept	23
11.2 MVC-Pattern	23
11.3 Jackson-JSON	23

12 Projektplanung	24
12.1 Projektstrukturplan	24
12.2 Soll-Ist-Vergleich	24
12.3 Arbeitsaufteilung	25
13 Schlussbetrachtung	26
13.1 Bewertung	26
13.2 Fazit	27
Anhang	28
Quellenverzeichnis	53

Abbildungsverzeichnis

Abbildung 1: ERD des Projekts	5
Abbildung 2: Grobe Ordnerstruktur	7
Abbildung 3: Ordnerstruktur der Ressourcen	8
Abbildung 4: Klassendiagramm Advantage	9
Abbildung 5: Ablauf der Anwendung	10
Abbildung 6: Testanwendung der Kern-Anwendung	16
Abbildung 7: Use-Case Diagramm	18
Abbildung 8: Farben Konzept	20
Abbildung 9: Ideen Konzept	21
Abbildung 10: Rechtsklick Umsetzung	22
Abbildung 11: Dropdown Umsetzung	22
Abbildung 12: Administrator - Use-Case Diagramm	32
Abbildung 13: Kontaktformular - Use-Case Diagramm	33
Abbildung 14: GUI-Konzept - Login	34
Abbildung 15: GUI-Konzept - Registrierung	35
Abbildung 16: GUI-Konzept - Willkommen	35
Abbildung 17: GUI-Konzept - Idee erstellen	36
Abbildung 18: GUI-Umsetzung - Login	37
Abbildung 19: GUI-Umsetzung - Registrierung	38
Abbildung 20: GUI-Umsetzung - Ideen	39
Abbildung 21: GUI-Umsetzung - Idee erstellen	40
Abbildung 22: GUI-Umsetzung - Idee ansehen	41
Abbildung 23: GUI-Umsetzung - Admin Ansicht	42
Abbildung 24: GUI-Umsetzung - Spezialist Ansicht	43
Abbildung 25: Projektstrukturplan	44
Abbildung 26: Klassendiagramm Model Ideen	51
Abbildung 27: Klassendiagramm Model User	52

Tabellenverzeichnis

Tabelle 1: Bewertung der Anwendung	26
Tabelle 2: GUI-Testdurchführung	29

Listingverzeichnis

Listing 1: Antwort /api/ideas/	47
Listing 2: Antwort /api/ideas/{id}	48

1 Installation [\[Jonathan Brockhausen\]](#)

Das Programm setzt eine installierte Java Runtime Environment der Version 11 voraus.

Kompilieren und Starten des Programmes

Wenn bereits eine JAR vorhanden ist, kann direkt zu Punkt 2 gegangen werden.

1. Das Programm kann mithilfe des Maven Wrappers kompiliert werden:
 - a) unter Linux / MacOS Systemen mit: `chmod +x mvnw & ./mvnw clean compile compile package`
 - b) unter Windows mit: `mvnw.cmd clean compile package`
2. Danach kann die Zieldatei aus dem Projekt Root-Verzeichnis ausgeführt werden.
 - a) `java -jar target/digitaler-briefkasten-1.0.1-ABGABE.jar` (Die Versionsnummer kann abweichen!)
3. Nach dem erfolgreichen Start ist die Oberfläche unter `http://localhost:8080` erreichbar.

Test-Zugangsdaten

Grundsätzlich existieren drei verschiedene Arten von Accounts:

1. Administrator
2. Spezialist
3. User

Zur Nutzung des Systems als User kann ein neuer User-Account registriert werden.

Ein Administrator-Account und ein Spezialisten-Account können mithilfe der Methoden in der Klasse `HelperScriptsNoTests` angelegt werden. Die Zugangsdaten sind wie folgt:

Administrator

1. **Username:** admin
2. **Passwort:** hierKönnteIhreWerbungStehen

Spezialist

1. **Username:** SpeziusMaximus__[Bezeichnung der jeweiligen Produktparte]

Beispiel: SpeziusMaximus_INTERNAL

2. **Passwort:** boringProphet

2 Fachkonzept [\[Jonathan Brockhausen\]](#)

Im Folgenden werden die im Projekt verwendeten Technologien aufgeführt.

2.1 Technologien

Java 11

Zum Ziele der größten Kompatibilität mit den weiteren Technologien haben wir uns für Version 11 des Java Development Kits als grundlegende Java-Version entschieden. Unter Java 11 können wir die ordnungsgemäße Funktionalität unserer Software garantieren, ein Update auf neuere Versionen kann zu Fehlern mit den Dependencies führen.

Maven (Wrapper)

Um die notwendigen Dependencies des Projekts bereitzustellen, nutzen wir Maven mit einem Wrapper. Maven erlaubt in der POM.XML-Datei die einfache und übersichtliche Verwaltung von Dependencies. Die wichtigsten Dependencies sind im nächsten Überabschnitt aufgeführt. Der Wrapper unterstützt das Kompilieren des Projekts, da vorher Maven geladen wird und somit keine Dependencies manuell installiert werden müssen; sie werden beim Ausführen des Wrappers automatisch geladen.

GitHub

GitHub ist die populärste und weitverbreiteste Plattform für Kollaboration auf Basis der Open-Source-Versionsverwaltung Git. Da alle drei Gruppenmitglieder mit der Plattform ansatzweise vertraut sind und die Integration in die verwendete IDE problemlos möglich ist, ist GitHub die ideale Plattform für das Projekt. Durch die GitHub CI-Tools wird die Funktionalität der Programmbestandteile automatisiert mit jedem Commit sichergestellt. Diese Funktion ist in Abschnitt 11.1 näher dargestellt.

OpenProject, Teams, Telegram

Für das Projektmanagement wurde OpenProject genutzt, eine Open-Source webbasierte Projektmanagement Suite. Auf OpenProject wird in Unterabschnitt 12.3 weiter eingegangen. Für die Kommunikation während des Projekts wurden Microsoft Teams für die Synchronisations-Calls und Pair Programming und Telegram für Messaging genutzt.

2.2 Frameworks

Springboot

Springboot wird genutzt um einen Tomcat-Webserver zu starten und unseren Programmcode darin zu deployen. Es ist ein in der Branche übliches Framework, um die Produktion von Enterprise-Web-Anwendungen zu vereinfachen. Die bereitgestellte Standardkonfiguration bietet eine gut auf die Bedürfnisse der Anwendung anpassbare Grundlage. Dazu wurden in den „application.properties“ unter anderem die Konfiguration der Programmbezeichnung, Port und Datenbankanbindung vorgenommen.

Thymeleaf

Thymeleaf rendert HTML-Dateien. Das Framework erlaubt es, Informationen im Frontend einfach anzeigen und gleichzeitig ordentlich zu implementieren. Mithilfe von Thymeleaf vermeiden wir an einigen Stellen Konflikte oder Umwege. Dadurch, dass Java-Objekte und -Logik direkt in die HTML-Dateien eingebunden werden können und Thymeleaf daraus sichere Frontend-Seiten rendert, fließen Front- und Backend zwar in der Programmierung dichter zusammen, sind jedoch im Betrieb sauber getrennt. Weiterhin ermöglicht Thymeleaf Integration mit Spring und dessen Security-Tools.

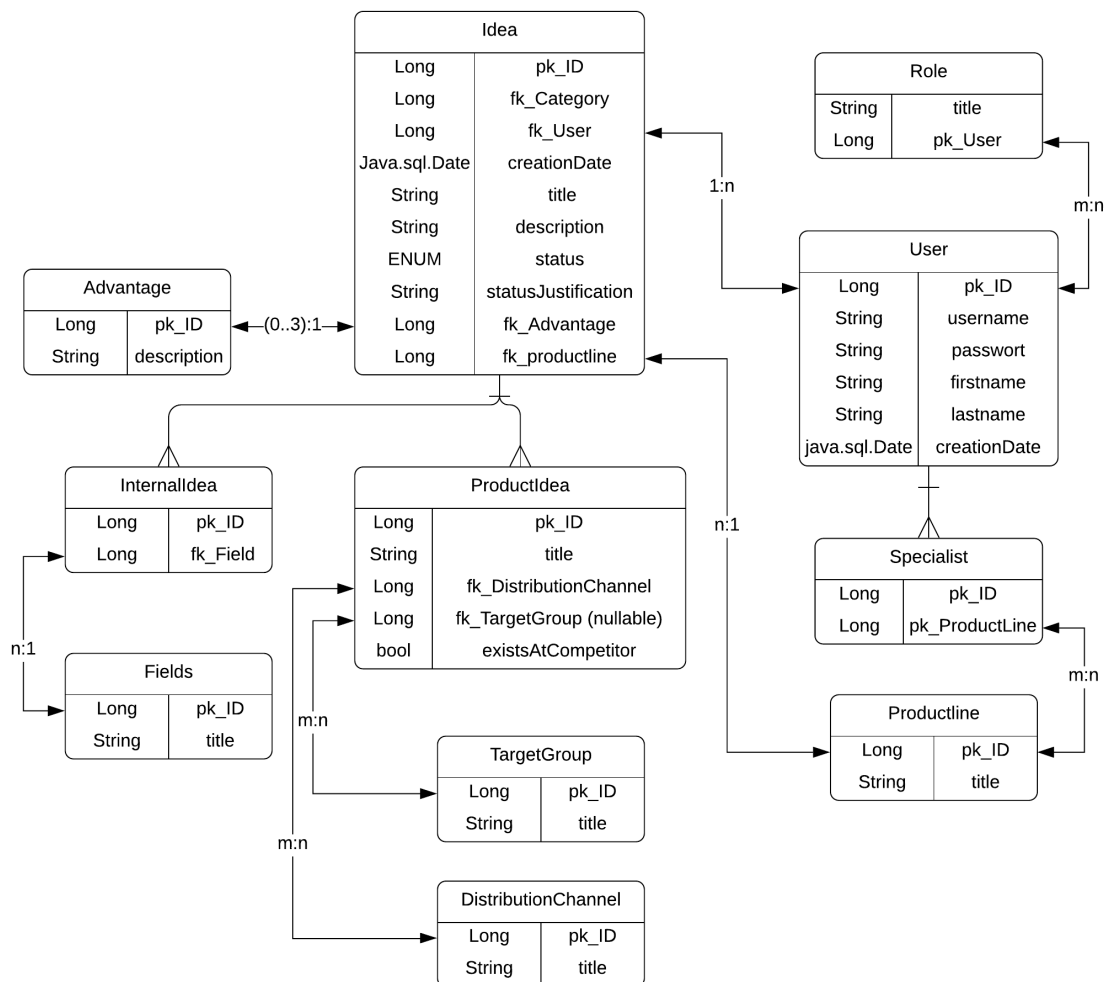
2.3 Datenbank

Als relationale Datenbank wurde eine h2-Datenbank verwendet. Wir nutzen JPA um die SQL-Tabellen mit Java-Klassen und -Objekten zu erzeugen. Durch die Verwendung von JPA wird vollständig auf die Erstellung von manuellen SQL-Queries verzichtet. Die h2-Datenbank ist dateibasiert und dadurch leichtgewichtiger und einfacher zu betreiben als eine server-basierte MSSQL- oder MySQL-Datenbank.

3 Entity-Relationship Diagramm [Jonathan Brockhausen]

In Abb. 1 ist das zugrundeliegende Entity-Relationship Diagramm dargestellt.

Abbildung 1: ERD des Projekts



Quelle: Eigene Darstellung

Die Datenstruktur wurde auf Basis dieses Entity-Relationship Diagramms implementiert. Im Folgenden werden einige Design-Entscheidungen erläutert.

Zuordnung von Fachspezialisten und Ideen

Um die automatische Zuordnung von Fachspezialisten zu Ideen umzusetzen, haben wir die Produktparte als Zuordnungskriterium gewählt. Die Klasse des Fachspezialisten erbt von der Klasse des Benutzers mit der zusätzlichen Eigenschaft, dass ihm eine oder mehrere Produktparten zugewiesen sind. Einer Produktparte können mehrere Fachspezialisten zugeordnet

sein. Mit dieser m:n-Beziehung erreicht das Programm die größtmögliche Flexibilität. Jeder Idee (Intern und Produkt) ist eine Produktparte zugeordnet. Aus den Projektanforderungen ergeben sich mehrere Produktparten, die bei der Auslieferung bereits vorhanden sind. Da interne Ideen gemäß der Anforderungen keine Produktparte besitzen, bekommen Sie die dem Benutzer verborgene Produktparte „INTERNAL“ zugewiesen. Diese ermöglicht für interne Ideen dieselbe Logik zu verwenden. Durch diese Umsetzung ist auch eine Erweiterung um weitere Ideenkategorien ohne Änderungen am übrigen Programm möglich.

Umsetzung von Status

Wir haben uns dagegen entschieden, den Status in eine eigene Entität im Sinne des ERD auszulagern. Erweiterungen und Änderungen der Status im Echtbetrieb erfordern ohnehin an einigen Stellen Änderungen in der Programmlogik, weshalb es keinen Sinn macht, Status als Datenbank-Entitäten umzusetzen.

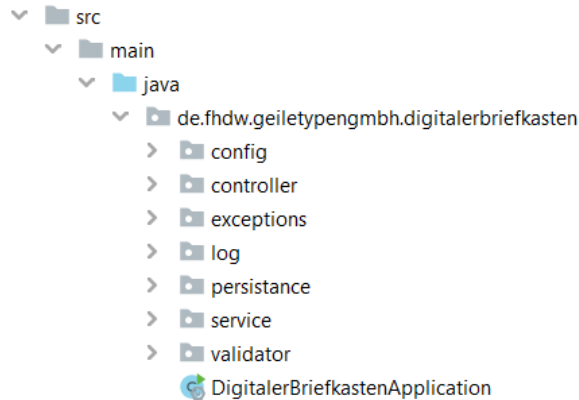
Vererbung von Ideen zu interne und Produktidee

Die Struktur der Vererbung von Ideen ermöglicht es, weitere Kategorien von Ideen anzulegen ohne die bestehenden zu verändern. Hierbei können bereits angelegte Eigenschaften genutzt und neue hinzugefügt werden.

4 Projekt-Architektur [\[Philipp Röring\]](#)

In Abb. 2 ist die grobe Ordnerstruktur des Quellcodes dargestellt.

Abbildung 2: Grobe Ordnerstruktur



Quelle: Eigene Darstellung

In dem Projektordner befindet sich der Unterordner `src/main/java/de/fhdw/geiletypengmbh/digitalerbriefkasten`. In diesem befindet sich der erstellte Quelltext. Die Unterordner (Packages) dessen werden folgend grob erklärt.

- config

Hierin ist die *SecurityConfig* der Anwendung enthalten.

- controller

Hierin liegen die Controller. Diese stellen die erreichbaren Endpunkte für die Benutzeroberfläche sowie die REST-API zur Verfügung.

- exceptions

Hierin befinden sich eigene Exceptions. Diese werden an die Benutzeroberfläche im Fehlerfall weitergeleitet. Ein Beispiel dafür ist die *IdeaNotFoundException*.

- log

Hierin liegen alle Klassen, die zum Logging von Ereignissen dienen.

- persistence

Das Package *persistence* dient zum Speichern der Daten (Objekte bzw. Entities) in der Datenbank. Es ist untergliedert in die packages

- model

Hierin liegen die Datenklassen (Entities). Sie entsprechen dem ER-Diagramm.

- repo

Hierin liegen die *Repositories*. Sie dienen zum Lesen, Schreiben, etc. der Datenklassen in die Datenbank.

- service

Hierin liegen die *Services*. Sie dienen als Abstraktionsschicht über den *Repositories*. Damit kann zusätzliche Logik z.B. vor dem Speichern einer Entität implementiert werden. Auch Hilfsmethoden befinden sich in den Services.

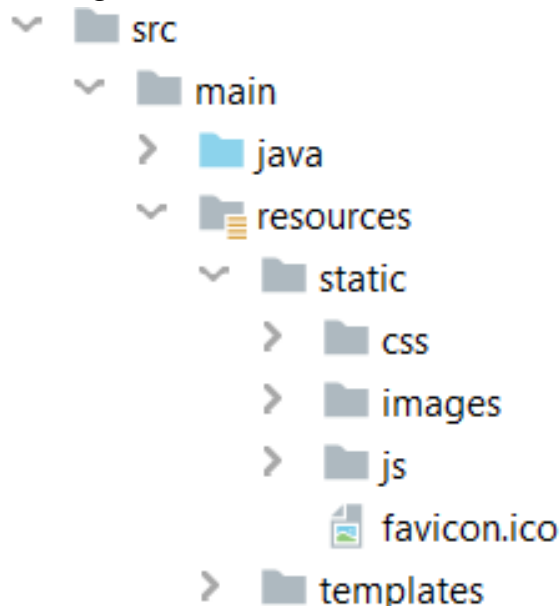
- validator

Hierin liegt die Validationsklasse für die Benutzererstellung. Sie enthält Prüfungen, wie z.B. die Prüfung auf Mindest-Passwortlänge.

Darüber hinaus befindet sich in dem Package noch die Hauptklasse der Anwendung *Digitaler-BriefkastenApplication*, durch welche sie gestartet wird.

Neben *src/main/java* existiert auch der Ordner *src/main/resources*. In diesem sind die Komponenten für die Weboberfläche der Anwendung enthalten. In Abb. 3 wird die Struktur von *resources* dargestellt.

Abbildung 3: Ordnerstruktur der Ressourcen



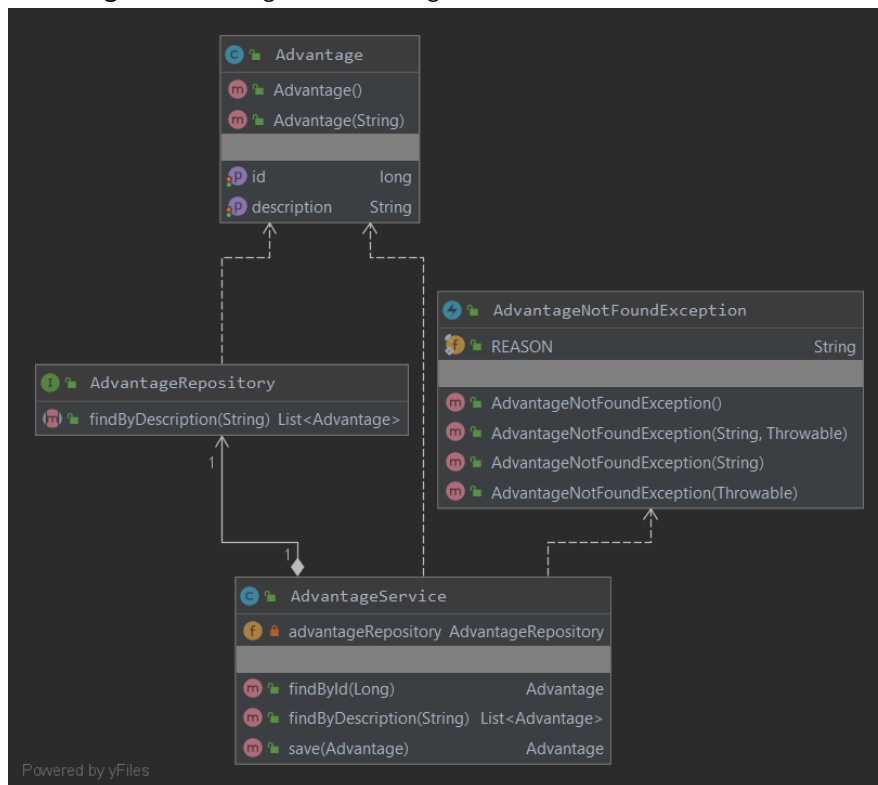
Quelle: Eigene Darstellung

Der Unterordner *static* beinhaltet css-, Bild- und JavaScript-Dateien. In *Templates* befinden sich die HTML Dateien. Die Testklassen der Anwendung befinden sich in dem Ordner *src/test/java/de/fhdw/geiletypengmbh/digitalerbriefkasten*.

4.1 Klassendiagramm

In Abb. 4 wird die Architektur am Beispiel der Klasse *Advantage* erklärt:

Abbildung 4: Klassendiagramm Advantage



Quelle: Eigene Darstellung

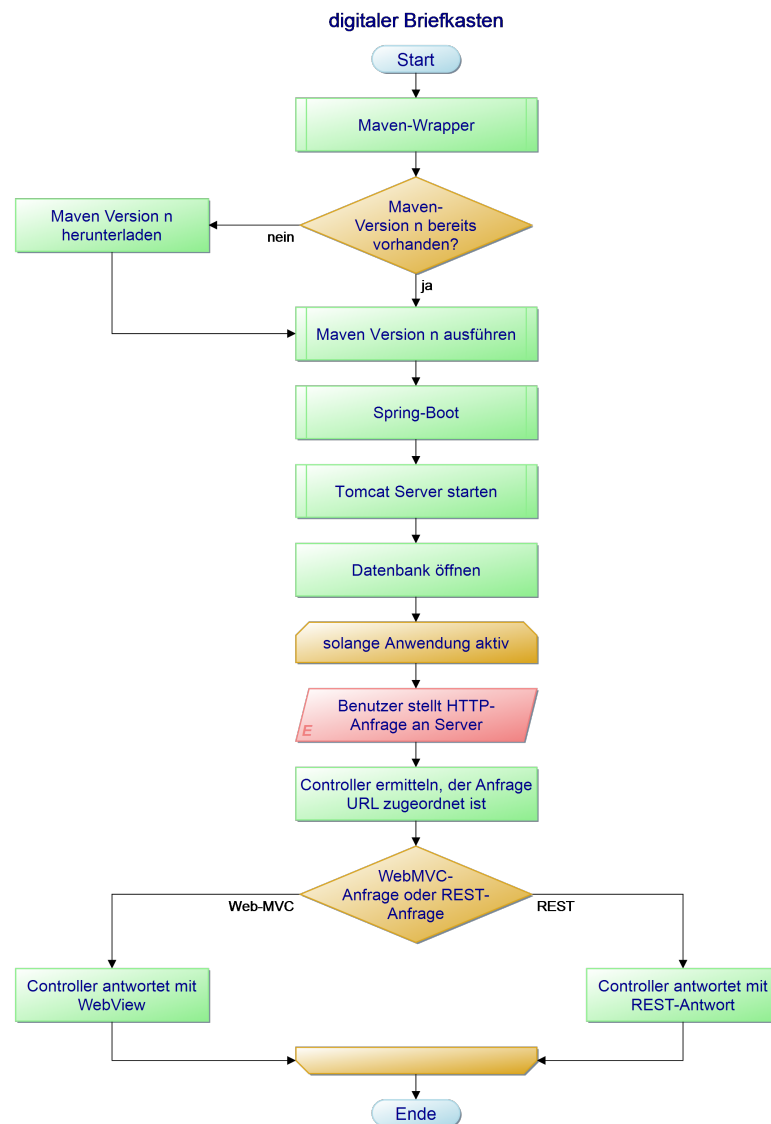
Die Klasse *Advantage* stellt die Entität dar, welche von den verschiedenen Schichten der Anwendung verwendet wird. Das *AdvantageRepository* ist eine Erweiterung des *JpaRepository* und dient somit zur Interaktion mit der Datenbank und *Advantage*-Objekten. Darauf aufbauende Logik wird in dem *AdvantageService* implementiert. Dieser wirft zum Beispiel eine *AdvantageNotFoundException* in der Methode *findById()*, wenn keine *Advantage* in der Datenbank gefunden wird.

Auf ein Klassendiagramm der gesamten Anwendung wurde verzichtet, da es zu unübersichtlich ist. In Anhang 6 sind Klassendiagramme des *Models* zu finden, die dem ER-Diagramm entsprechen.

5 Abläufe [Philipp Röring]

In Abb. 5 wird der grobe Ablauf der Anwendung in Form eines Programmablaufplans skizziert. Dabei sollte beachtet werden, dass Maven nur während der Entwicklung der Anwendung relevant ist und nicht in der kompilierten Anwendung enthalten ist.

Abbildung 5: Ablauf der Anwendung



Quelle: Eigene Darstellung

6 Schnittstellen [\[Philipp Röring\]](#)

Die Anwendung ist über eine REST-API erreichbar. Alle Antworten sind im JSON-Format.

6.1 Schnittstellenbeschreibung REST-API

Die Schnittstelle stellt folgende Services bereit:

- HTTP-Methode: GET

Relativer Pfad: */api/ideas/*

Antwort: Array, das alle Ideen (Produkt-/ Interne Ideen) beinhaltet.

Beispielantwort: siehe Anhang 5.1

- HTTP-Methode: GET

Relativer Pfad: */api/ideas/{id}*

Antwort: Idee (Produkt-/ Interne Idee) in JSON-Format

Beispielantwort: siehe Anhang 5.2

- HTTP-Methode: GET

Relativer Pfad: */api/ideas/title/{title}*

Antwort: Idee (Produkt-/ Interne Idee) in JSON-Format

Beispielantwort: siehe Anhang 5.2

- HTTP-Methode: GET

Relativer Pfad: */api/ideas/submitted*

Antwort: Array, das alle Ideen (Produkt-/ Interne Ideen), die veröffentlicht sind, beinhaltet.

Beispielantwort: siehe Anhang 5.1

- HTTP-Methode: POST

Relativer Pfad: */api/ideas/*

Mitzugebener HTTP-Body: Idee in JSON-Format (Syntax siehe GET-Methoden)

Bei Erfolg

HTTP-Status 201 im HTTP-Header, sowie die erstellte Idee als JSON im HTTP-Body

Bei Fehler

Fehlerrückmeldung

Beispielantwort: siehe Anhang 5.2

- HTTP-Methode: DELETE

Relativer Pfad: */api/ideas/{id}*

Antwort: HTTP-Status 200 bei Erfolg, ansonsten Fehlerrückmeldung

- HTTP-Methode: PUT

Relativer Pfad: */api/ideas/{id}*

Mitzugebener HTTP-Body: Idee in JSON-Format (Syntax siehe GET-Methoden)

Antwort: HTTP-Status 200 bei Erfolg, ansonsten Fehlerrückmeldung

- Fehlerrückmeldung der API

Beispielantwort:

```
{
  "timestamp": "2020-05-28 12:20:59.386",
  "status": 404,
  "error": "Not Found",
  "message": "Keine entsprechende Idee gefunden",
  "path": "/api/ideas/111"
}
```

Der Status der Antwort entspricht dem Status des HTTP-Header der Antwort.

6.2 Login

Um die GET- */api/ideas/* sowie die POST-, DELETE- und PUT- Aufrufe der API nutzen zu können, muss sich mit einem Benutzer der Rolle *API_USER* angemeldet werden. Dies wird hier nicht genauer spezifiziert, da es keine Anforderung war. Die Funktionalität wurde aber bereits in der Basis für zukünftige Erweiterungen hinzugefügt und kann mit einem REST-Client getestet werden. Ein *API_USER* kann mit Hilfe des Skripts *HelperScriptNoTests* erstellt werden. Die restlichen GET-Methoden können ohne Authentifizierung aufgerufen werden. Diese entsprechen den Ansichten, die auch in der Benutzeroberfläche ohne Anmeldung angesehen werden können.

6.3 Webschnittstelle

Die restlichen Schnittstellen, die von der Benutzeroberfläche verwendet werden, verwenden das Format *application/x-www-form-urlencoded*. Jenes Format ermöglicht die native Verwendung von HTML-Forms ohne die Formulardaten per JavaScript umformen zu müssen. Um diese Aufrufe testen zu können, können die Entwickleroptionen eines Webbrowsers oder ein REST-Client verwendet werden. Dazu muss in den HTTP-Header des Requests der Cookie *JSESSIONID* und das *X-CSRF-TOKEN* eingefügt werden. Diese können dem Browser in den Entwickleroptionen nach einem Login entnommen werden.

7 Security [Julius Figge]

Die Security der Anwendung wird durch mehrere Bestandteile sichergestellt, diese lassen sich in aktive und passive Elemente unterteilen.¹

7.1 Aktive Security Bestandteile

Zuerst ist der Login sowie die Registrierung abgesichert. Nutzer müssen ein Passwort mit mindestens 8 Zeichen wählen, welches im Backend inklusive Salt gehashed gespeichert wird. Hierzu wird BCrypt als Password-Encoder genutzt. Dieser wurde mit einer Stärke von 10 verwendet, um die beste Balance zwischen Sicherheit und Performance zu erreichen. Mit dem Login bekommen Nutzer einen Cookie in Form einer JSession ID, mit dem sie sich in weiteren Requests authentifizieren und über den sie identifiziert werden können.

Der nächste Bestandteil ist die URL-Zugriffskontrolle in der Klasse „SecurityConfig“ im Package „config“. In dieser wird festgelegt, welche Requests durch Spring Security zugelassen werden. Nicht authentifizierte Nutzer haben hier nur Zugriff auf statische Elemente (wie z.B. Grafiken, Javascript und CSS), die Registrierung und die Ideenansicht. Authentifizierte Nutzer werden anhand ihrer Rolle unterschieden, welche im Backend überprüft wird. Nutzer, Spezialisten und Administratoren können nur auf die jeweils für sie relevanten Seiten zugreifen. Der durch Spring Security erstellte JSession-Cookie wird beim Ausloggen invalidiert und gelöscht.

Darüber hinaus ist die Anwendung so konfiguriert, dass ein automatischer Session Timeout nach 15 Minuten erfolgt, auch hierbei wird die Session, und somit der Session-Cookie, invalidiert.

Außerdem werden alle Abfragen durch das Backend geprüft. An relevanten Stellen wird in den jeweiligen Controllern bereits vor der Bearbeitung des Requests die Rolle des aktuellen Users überprüft. Damit wird sichergestellt, dass Funktionen, die insbesondere dem Administrator oder Spezialisten vorbehalten sind, nur durch diese durchgeführt werden können.

Des weiteren werden übertragene Informationen in den bearbeitenden Services auf die Berechtigung diese anzufragen, zu verändern oder zu speichern geprüft.

Durch diese Kontrolle an mehreren Stellen können wir sicherzustellen welche Art von Requests (un-, authentifziert), welcher User mit welcher Rolle auf welche Daten wie zugreifen (lesen, bearbeiten, schreiben) dürfen.

¹Zu beachten ist, dass wir das Programm unter der Prämisse entwickelt haben, dass im Livebetrieb eine zusätzliche SSL-Verschlüsselung für den Traffic genutzt wird.

7.2 Passive Security Bestandteile

Zu den passiven Bestandteilen gehört das Loggen von Anmeldeversuchen, Anmeldung, Registrierung und Abmeldung vom System.² Dies wird durch mehrere Klassen im Package „log“ sichergestellt. Diese implementieren einen jeweiligen Application-Listener, beispielhaft für den fehlerhaften Login der ApplicationListener `AuthenticationFailureBadCredentialsEvent`. Beim Auftreten eines passenden Applicationevents wird mit Hilfe eines Loggers, den „slf4j“ bereitstellt der aktuelle Zeitstempel sowie Nutzernamen und IP-Adresse geloggt. Hierbei ist anzumerken, dass die Logs zusätzlich außerhalb der Konsole in eine Datei geschrieben werden. Diese ist auf 5Mb begrenzt und rotiert oberhalb dieser Grenze automatisch. Zudem sind die zu schreibenden Logs eingeschränkt, sie sind auf Package-Level anhand der jeweiligen Log-Typen angepasst. Damit stellen wir sicher, dass nur relevante Informationen festgehalten werden und diese auch unabhängig vom Programm zur Auswertung zur Verfügung stehen. Des Weiteren sind Fehlermeldungen eingeschränkt um nicht aus versehen Informationen durchsickern zu lassen. Beispielhaft zeigt der Login ausschließlich eine Fehlermeldung über fehlerhafte Daten an - jedoch nicht ob der Nutzernamen oder das Passwort falsch war. Darüber hinaus werden Exceptions gefiltert und nur ausgewählte (respektive unsere eigenen) in Form von einer jeweils angepassten Nachricht auf der Error-Seite angezeigt. Damit stellen wir sicher, dass nicht versehentlich Exceptions, Stacktraces oder Debug-Logs an das Frontend gelangen und für den Nutzer sichtbar sein könnten.

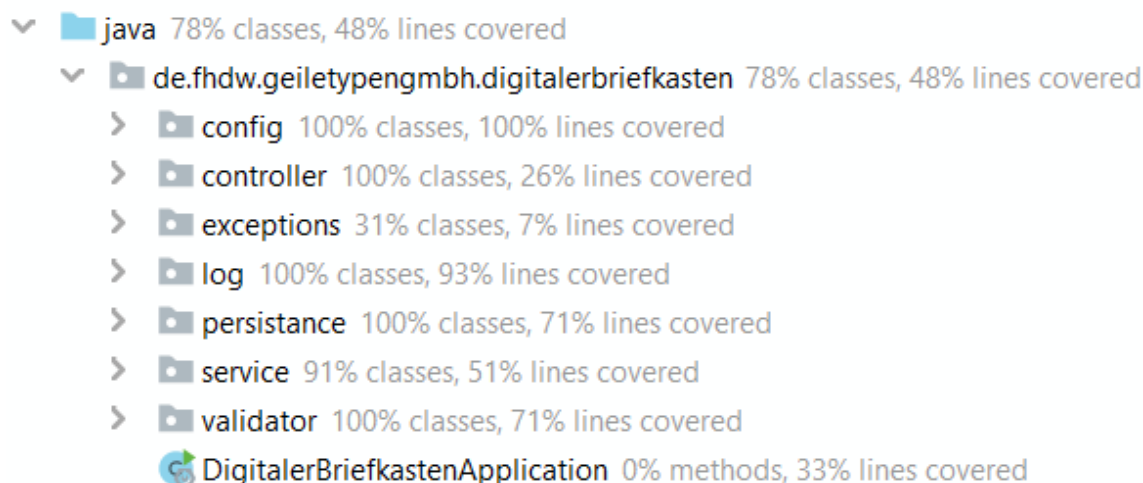
²Das Loggen von Session Timeouts konnte aufgrund von Komplikationen zum Abgabezeitpunkt nicht fertiggestellt werden.

8 Test

8.1 Testklassen [Jonathan Brockhausen]

Als Testframework haben wir JUnit verwendet. Wir haben dedizierte Testklassen, in denen wir die relevantesten Teile der geschriebenen Programmlogik mit Tests abdecken können. Wie in Abschnitt 11.1 bereits geschrieben wurde, können wir durch die Einbindung der GitHub CI eine Sicherstellung der Funktionalität erreichen. Im Folgenden werden die konkreten Testklassen und die Abdeckung erläutert. Die Testabdeckung der Kern-Anwendung ist in Abb. 6 dargestellt.

Abbildung 6: Testanwendung der Kern-Anwendung



Quelle: Eigene Darstellung

Insgesamt erreichen wir eine Testabdeckung von 78%. Von den Tests wird nahezu die gesamte Funktionalität des Programms abgedeckt. Elemente, die nicht getestet werden, sind im wesentlichen die Exceptions und Teile der Controller, in denen beispielsweise das funktionell identische Anlegen von Produktparten, Zielgruppen, Vertriebskanälen und Handlungsfeldern nur einmal stellvertretend getestet wird.

Die Tests sind auf zwei Klassen aufgeteilt:

`IdeaControllerIntegTest` und `UserControllerIntegTest`. Wir trennen damit die Integrationstests von Ideen- und Benutzerfunktionalität voneinander. In beiden Testklassen wird zunächst mit Helfermethoden die Testumgebung vorbereitet indem Objekte und Variablen initialisiert werden, die von den Tests genutzt werden. Die tatsächlichen Testmethoden sind jeweils sprechend benannt und benutzen größtenteils die vom Spring-Framework bereitgestellten MockMvc-Funktionalitäten um Anfragen zu senden und die Antworten auszuwerten. Die einzelnen Tests testen jeweils eine vollständige Funktionalität des Programms. Es kann

mehrere Tests geben, die eine Funktionalität mit deren Abwandlungen testen. Aufgrund des logisch zusammenhängenden Aufbaus der Testklassen und -Methoden wird auf eine detaillierte Erläuterung verzichtet.

8.2 manuelle-„Klicktests“ [\[Julius Figge\]](#)

Zur Überprüfung der GUI sollen manuelle Klicktests durchgeführt werden. Diese sollen dokumentiert werden um Fehler möglichst gezielt beheben zu können.

Zu notierende Informationen

Für die Auswertung relevant sind zum einen die Programmrevision (Git Commit Hash, Datum) sowie der verwendete Branch. Darüber hinaus ist das genutzte Betriebssystem sowie der genutzte Browser (inklusive Build zu notieren). Bei Darstellungsfehlern ist es sinnvoll, zudem Screenshots zu hinterlegen sowie die Bildschirmauflösung zu notieren. Diese Informationen sammeln wir gezielt sehr detailliert, um Fehler besser eingrenzen zu können.

Testvorbereitung

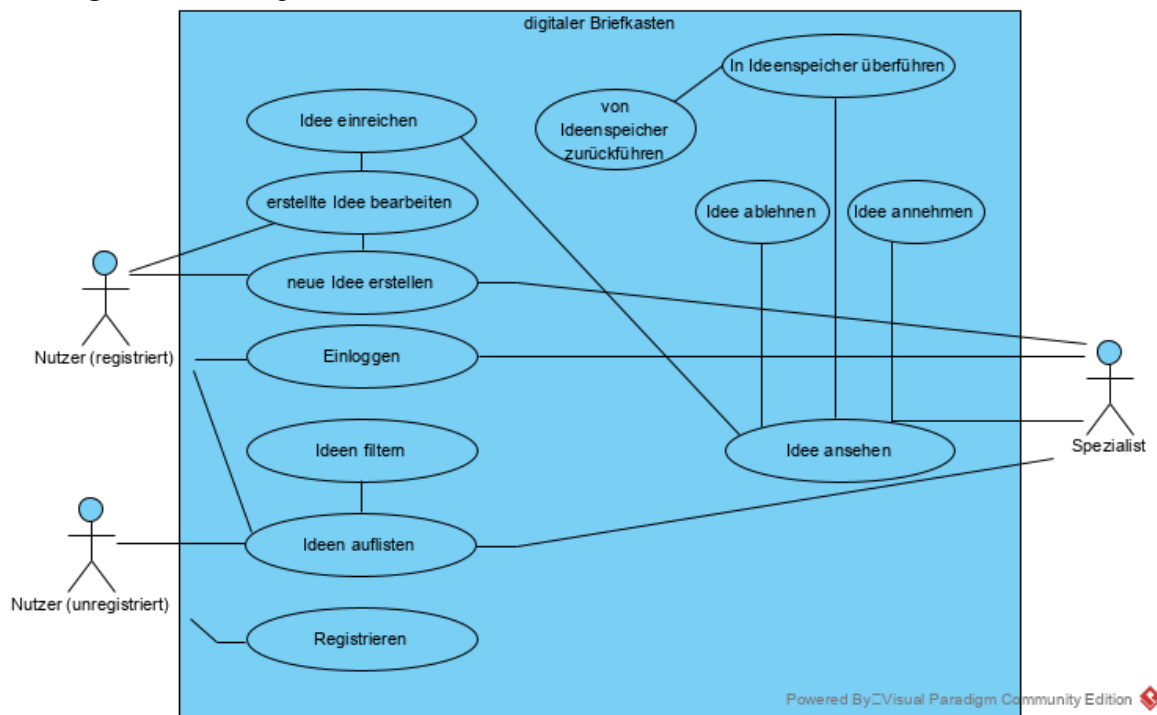
1. Zum Testen wird der neueste Stand des Master-Branches verwendet.
2. Hierzu ist zunächst die Datenbank zu löschen und mit Hilfe der in „HelperScriptsNoTests“ vorhandenen Tests zu füllen.
3. Der Code soll kompiliert werden und die entstandene „Jar“-Datei ausgeführt werden.
4. Nach Möglichkeit soll der Test auf mehreren Browsern ausgeführt werden. Hierbei ist zu beachten, dass alle Addons zu deaktivieren sind, um eventuelle Komplikationen auszuschließen.
5. Die Entwicklerkonsole ist zu öffnen um hier entstehende Fehler und Warnungen mit in die Testergebnisse aufzunehmen.
6. Nachdem diese Voraussetzung geschaffen ist, sind die Tests durchzuführen und die obigen Informationen zu notieren.

Zur Testdurchführung ist die Tabelle im Anhang 1 auf S.29 zu verwenden.

9 Use-Cases [Julius Figge]

Im Nachfolgenden sind die Use-Cases des Programms dargestellt (siehe Abb. 7). Diese sind den Projektvorgaben entnommen.³

Abbildung 7: Use-Case Diagramm



Quelle: Eigene Darstellung

Für das Use-Case Diagramm sind drei Rollen von Relevanz. Zuerst der **unregistrierte Nutzer**, was den Zugriff ohne besondere Rechte darstellt. Des weiteren der **eingeloggte Nutzer** der mehr Möglichkeiten hat. Hierzu gehört auch der **Administrator**. Dieser hat über die Möglichkeiten des Nutzers hinaus weitere administrative Rechte.⁴ Jedoch besitzt er nicht die Rechte der dritten Rolle des **Spezialisten**.

Die Use-Cases lassen sich in zwei „Kern“-Kategorien unterteilen. Das sind zunächst die **Account** bezogenen Use-Cases.

Hierzu gehören der Vorgang des Einloggens sowie das Registrieren. Zu diesem ist anzumerken, dass Spezialisten sich lediglich einloggen können. Durch ihre weitreichenden Rechte dürfen diese sich nicht normal registrieren, sondern werden durch den Administrator angelegt.

Der zweite Use-Case ist die **Erstellung und Verwaltung von Ideen**.

³Vgl. Heuermann, Christian (2020)

⁴Der Admin ist als eigener Use-Case im Anhang dargestellt. Siehe Anhang 2.1 auf S.32

Eingereichte Ideen lassen sich durch alle Nutzer jeder Rolle einsehen und filtern. Darüber hinaus haben alle eingeloggtten Nutzer die Möglichkeit Ideen zu erstellen, zu bearbeiten und zur Bewertung einzureichen.

Diese eingereichten Ideen werden durch Spezialisten bewertet oder gespeichert.

Des weiteren existiert die Möglichkeit, für alle Nutzer, dem Administrator der Plattform über ein Kontaktformular Nachrichten zu senden.⁵

⁵Das zugehörige Use-Case Diagramm findet sich im Anhang 2.2 auf S.32

10 GUI-Konzept [Julius Figge]

Wir haben uns entschieden, statt eines GUI-Mockups unser GUI-Konzept direkt im Prototypen mit auszuliefern. Das lässt sich durch mehrere Punkte begründen. Zuerst hatten wir zum Zeitpunkt der ersten Präsentation bereits einen funktionierenden Prototypen und konnten diesen direkt mit dem GUI-Konzept ausstatten. Dadurch war nicht nur ein Mockup vorhanden, sondern es konnte bereits während der Konzeptpräsentation mit der GUI interagiert werden. Des weiteren hatten wir dadurch die Möglichkeit die Zeit für die Erstellung eines Konzeptes direkt in die Entwicklung der GUI zu stecken.

Die GUI wurde unter Nutzung von Bootstrap 4 in Kombination mit Font-Awesome für die Icons entwickelt. Dadurch war es uns möglich eine konsistente, verständliche und klare Oberfläche zu entwickeln. Hierbei wurde viel Wert auf Einheitlichkeit und Konsistenz gelegt. Alle Seiten werden auf weißem Hintergrund dargestellt. Buttons und Informationen sind generell in grau (beziehungsweise Schwarz) gehalten. Abweichend hiervon treten Farben nur auf, um die Aufmerksamkeit des Nutzers auf sich zu ziehen oder um Hinweise hervorzuheben. Diese Farben sind in Abb. 8 abgebildet. Die Verwendung wird im weiteren näher erläutert.

Abbildung 8: Farben-Konzept



Quelle: Eigene Darstellung

Grundlegend sind die Elemente der Anwendung zentriert, wie im Weiteren zu sehen ist. Damit erreichen wir in Kombination mit der Nutzung von Bootstrap eine nahezu 100-prozentige Kompatibilität zu mobilen Endgeräten.⁶

Grundlegende Elemente dieses Konzeptes sind zum einen der Login-Screen (siehe 3 im Anhang auf S.34). In diesem Screenshot ist auch das Logo der Anwendung zu sehen, welches ebenfalls in den typischen Farben gehalten wurde. Dieses soll durch seine gleichzeitig humorvolle als auch simple Darstellung der Anwendung einen hohen Wiedererkennungswert geben.

Die zweite zentrale Komponente des Konzeptes ist die Übersicht aller Ideen (siehe Abb. 9). Auffallend ist hier die Gliederung der Ideen in Tabellen. Bereits zu diesem Zeitpunkt war

⁶Hierbei ist jedoch anzumerken, dass dieses Feature nicht gefordert war und somit auch nicht weitergehend getestet wurde. Allerdings sind bereits die Voraussetzungen für eine mögliche Erweiterung der Anwendung geschaffen.

geplant die Ideen nach Typ zu gliedern und in einer Übersicht mit ihren wichtigsten Eigenschaften darzustellen. Zu diesem Zeitpunkt noch per Klick⁷, sollte es möglich sein die Idee im Detail inklusive aller Informationen darzustellen. Diese Entscheidung begründet sich damit, das Gleichgewicht zwischen der verfügbaren Information auf einer Seitenansicht und der Übersichtlichkeit zu wahren. Darüber hinaus findet sich auch hier die Farbgestaltung wieder. Grundsätzlich ist die Oberfläche monochrom gehalten. Icons dienen der schnelleren Identifikation der verschiedenen Tabellen und der Übersichtlichkeit. Farbakzente sind zum einen zur Führung der Nutzer gedacht, siehe beispielhaft in dem Hyperlink auf den Ersteller der Ideen⁸. Zum anderen sind diese in den Status der Ideen mit einbezogen, hierdurch lässt sich erheblich schneller ein Überblick verschaffen.

Abbildung 9: Ideen Konzept

Digitaler Briefkasten Home Create Idea List Ideas Logout				
Not submitted Ideas				
title	description	creator	creation date	status
konzept 1	Testbeschreibung	konzept	2020-05-20	ⓘ not submitted
Product-Ideas				
title	description	creator	creation date	status
weitere Idee	Testbeschreibung	konzept	2020-05-20	⌚ pending
Geht nicht junge	geheime Nachricht	konzept	2020-05-20	✅ accepted
Internal-Ideas				
title	description	creator	creation date	status
weitere Idee	Testbeschreibung	konzept	2020-05-20	⌚ pending
Geht nicht junge	geheime Nachricht	konzept	2020-05-20	✅ accepted

Quelle: Eigene Darstellung

Ein weiterer relevanter Punkt der sich beispielhaft in dieser Abbildung (Siehe Abb. 9) findet, ist die Navigationsleiste. Diese ist im Konzept nur nach dem Login vorhanden. Im fertigen Produkt wurde diese aber auf jeder Seite inkludiert.⁹ Zugleich ist sie zentrales Steuerelement der Anwendung. Auf der linken Seite findet sich das Logo dauerhaft präsent wieder. Daneben

⁷Diese Funktionalität wurde im weiteren durch ein Rechtsklick Menü erweitert (Siehe 3.2 S.37).

⁸Dieser Hyperlink stand beispielhaft für die Weiterleitung auf eine Detailseite auf der Tabellensicht.

⁹Ebenso wurde ein Footer eingefügt. Vgl. 3.2 S.37

werden zur Verfügung stehende Seiten angezeigt, wobei die aktuelle hervorgehoben ist.¹⁰ Auf der rechten Seite findet sich der Logout Button, auch dieser ist hervorgehoben um vom Nutzer wahrgenommen zu werden.

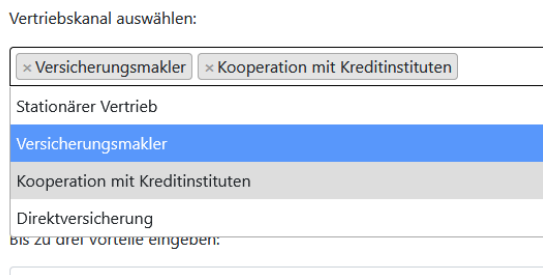
Die weiteren Konzeptteile der GUI finden sich im Anhang 3 auf S.34.

Abbildung 10: Rechtsklick Umsetzung



Quelle: Eigene Darstellung

Abbildung 11: Dropdown Umsetzung



Quelle: Eigene Darstellung

Hervorzuheben ist, dass gegenüber des Konzeptes in der Umsetzung¹¹ einige Elemente hinzugekommen sind. Die wichtigsten hierbei sind das bereits genannte Rechtsklick Menü (Siehe Abb. 10). Sowie intelligente Dropdown Menüs (Siehe Abb. 11). Diese sollen dem Nutzer die Möglichkeit geben intuitiv die Anwendung zu bedienen und erweitern diese durch dynamische Menüs welche sich in die Oberfläche einpassen.

¹⁰Im fertigen Produkt ist diese Sicht abhängig von den verschiedenen Rollen. Vgl. 3.2 S.37

¹¹Siehe Anhang 3.2 auf S.37

11 Konzepte [Julius Figge]

11.1 Arbeitskonzept

Unsere Teamarbeit haben wir auf den Austausch untereinander ausgerichtet. So konnten wir uns unsere unterschiedlichen Kompetenzen zu Nutze machen und haben beispielhaft im Mob-Programming¹² Wissen vermittelt und uns gegenseitig unterstützt.

Darüber hinaus haben wir nach dem „All hands on Deck“-Prinzip¹³ gearbeitet.

Das haben wir zum einen aufbauend auf Teaminterne Kommunikation (beispielhaft über Telegram), aber insbesondere auch über die „Github CI“ erreicht. Diese war konfiguriert bei jedem Commit alle Tests durchzuführen und bei Problemen Email-Benachrichtigungen zu versenden.

Ausserdem haben wir „Sonarlint“ eingesetzt um unsere Code-Qualität zu überprüfen und stetig zu verbessern. Dadurch haben wir nicht nur unsere Code Qualität sichergestellt, sondern konnten auch auftretenden Probleme möglichst schnell erkennen und beheben.

Hierbei ist positiv anzumerken, dass durch das gemeinsame Arbeiten Wissensilos effektiv aufgebrochen wurden und der Lerneffekt im Zuge des Projekts für alle beteiligten maximiert wurde.

11.2 MVC-Pattern

In unserer Anwendung benutzen wir das Architekturmuster Model View Controller. Dieses Muster haben wir explizit ausgewählt, da Springboot zusammen mit Thymeleaf als Frontend hierfür sehr gut geeignet ist. Dadurch erreichen wir eine strikte Trennung der verschiedenen Ebenen und eine bessere Anpassbarkeit des Programmes.

11.3 Jackson-JSON

Wir haben uns für die Nutzung der „Jackson“-JSON library entschieden unter anderem, da diese Annotations mitliefert welche wir direkt in unseren Code einbinden können. Diese benutzen wir um die Serealisierung und Deserealisierung von Datenbank-Einträgen zu verwalten. Dadurch ist es einfach im Code zu kontrollieren welche Einträge wie serealisiert werden. Das ist insbesondere für die Entwicklung der API relevant.

¹²Hiermit ist das gemeinse Programmieren über ein Videotelefonat gemeint, bei dem abwechselnd eine Person den Bildschirm teilt.

¹³Dieses bezeichnet den Ansatz, bei auftretenden Problemen und Fehlern sich zuerst auf die Behebung dieser zu konzentrieren, bevor weitergehende Aufgaben bearbeitet werden.

12 Projektplanung [Jonathan Brockhausen]

12.1 Projektstrukturplan

Das Projekt wurde von uns in vier Phasen aufgeteilt, *Vorbereitung*, *Implementierung*, *„Dokumentation und Tests“* und *Abschluss*. Der Projektstrukturplan ist im Anhang auf Seite 44 dargestellt.

12.2 Soll-Ist-Vergleich

Der vor dem Projekt von uns festgelegte und in der Präsentation des Fachkonzepts vermittelte Soll-Zustand ist die vollständige Umsetzung der Muss-Features und die in der angegebenen Reihenfolge begonnene Umsetzung der Kann-Features. Im Anhang auf Seite 45 ist die Übersicht der Features dargestellt. Alle Soll-Features wurden anforderungsgemäß umgesetzt. Die Umsetzung der Kann-Features wurde gemäß der im Fachkonzept vorgestellten Priorisierung durchgeführt. Im vorgegebenen Projektzeitraum wurden dabei die drei am höchsten priorisierten Kann-Features implementiert. Im Einklang mit dem gesamten Projekt wurde bei allen Features darauf geachtet, dass sie gut erweiter- und wartbar sind.

Die Programmierung der REST-API wurde von uns als wichtigstes Kann-Feature priorisiert. Besonders im Unternehmenskontext kommen oft Schnittstellen zwischen sehr verschiedenen Programmen vor. Mit der Verwendung einer REST-API haben wir eine in gewissen Maßen standardisierte Schnittstelle, die durch das universelle Rückgabeformat JSON eine Anbindung im Unternehmen unterstützt. Die Filterung wurde noch nicht vollständig umgesetzt, ist aber auf der bestehenden Code-Basis umsetzbar.

Als zweites Kann-Feature haben wir ein Kontaktformular umgesetzt. Das Kontaktformular bietet für Benutzer und Administratoren gleichzeitig eine an das System angebundene Anlaufstelle für Problemmeldungen und Anfragen. Die Nachrichten laufen im Administrator-Interface auf und sind dort für alle Administratoren sicht- und bearbeitbar.

Der Administrator war das dritte Kann-Feature welches von Anfang an hoch priorisiert war und früh im Programm umgesetzt wurde. Neben der Benutzerverwaltung können Administratoren Spezialisten anlegen und weitere Einträge in den Vorlauftabellen anlegen. Außerdem kommen die oben genannten Kontaktnachrichten im Administrator-Interface an. Der modulare Aufbau des Administrator-Interfaces macht es einerseits übersichtlich für den Nutzer und andererseits gut erweiterbar um weitere Funktionen.

Die übrigen Kann-Features wurden zunächst nicht implementiert. Das Projekt kann jedoch um diese Features erweitert werden ohne bestehende Logik zu sehr verändern zu müssen. Eine Mail-Server-Anbindung wäre ein sinnvoller nächster Schritt der die Einbindung einiger

weiterer Features und die Erweiterung von bestehenden Features ermöglicht (beispielsweise das Kontaktformular).

12.3 Arbeitsaufteilung

Für die Arbeitsaufteilung wurden die regelmäßigen Synchronisations-Calls genutzt. Die anstehenden Aufgaben wurden in Arbeitspakete aufgeteilt und gemeinsam im Team verteilt. Hierbei wurden persönliche Fähigkeiten und Vertrautheit mit der speziellen Code-Stelle besonders in Betracht gezogen. Durch sehr regelmäßige Kommunikation und Abgleiche wurde sichergestellt, dass alle Teammitglieder informiert waren, wer an welcher Stelle arbeitet und somit Konflikte im Code vermieden. Für das Festhalten der Arbeitspakete und der individuellen Fortschritte wurde das Projektmanagement-Tool OpenProject verwendet. Der Umfang des Projekts lässt über die Sinnhaftigkeit eines dedizierten Projektmanagementtools sicherlich streiten, aber insgesamt konnte so deutlich besser eine Struktur in die Arbeitsaufteilung gebracht werden.

13 Schlussbetrachtung [\[Philipp Röring\]](#)

13.1 Bewertung

In Anhang 4.2 ist zu sehen, dass alle Muss-Features der Anwendung implementiert wurden. Die Kann-Features wurden gemäß der vorgestellten Priorisierung implementiert. Es gab somit bezüglich der Implementierung keine Abweichungen von der Projektplanung.

Trotz der Aufteilung des Entwicklungsaufwandes wurde bei Unklarheiten mehrmals zusammen nach Lösungen gesucht. Auch im Rahmen der Qualitätssicherung wurde der Quelltext gemeinsam überprüft und besprochen. Unabhängig davon stand es jedem Entwickler frei, nach Absprache den Quelltext eines anderen zu überarbeiten, wenn er Fehler bzw. Unschönheiten in diesem gefunden hat. Die Zusammenarbeit wird allgemein als sehr gut eingeschätzt. Lediglich die rein digitale Kommunikation aufgrund der aktuellen Covid-19 Situation hat die Zusammenarbeit in geringem Maße erschwert. Die Entwickler sind sich jedoch einig, dass dies nicht die hohe Qualität der erstellten Anwendung gesenkt hat.

Durch den sehr frühen Beginn der Entwicklung konnten die Laufzeit des Gesamtprojektes sowie die einzeln vergebenen Deadlines für die Entwickler fristgerecht eingehalten werden. Auch das Aneignen von Know-How über Spring, Spring-boot und Thymeleaf konnte in die Projektlaufzeit integriert werden.

In den letzten Releases der Anwendung liefen alle Tests, die automatischen sowie die manuellen GUI-Tests, fehlerfrei durch. Es folgt eine kurze eigene Bewertung der Qualität der Anwendung (1-10 Punkte).

Tabelle 1: Bewertung der Anwendung

Änderbarkeit	Benutzbarkeit	Effizienz	Funktionalität	Übertragbarkeit	Zuverlässigkeit
8	8	6	8	10	9

Die etwas niedrige Effizienz im Gegensatz zu sehr hoher Übertragbarkeit ist auf die Java Programmiersprache zurückzuführen. Darüber hinaus wurde zwar während der Entwicklung auf Performanz geachtet, allerdings wurde diese aufgrund der mittleren Benutzeranzahl nicht gegenüber anderen Qualitätsmerkmalen priorisiert. Die Bewertung der Zuverlässigkeit wurde nach dem ständigen Bestehen der Tests sowie dem fehlenden Aufkommen von Programmabbrüchen bewertet. Die hohe Änderbarkeit resultiert aus strukturiertem Quelltext, bei dem sich an die Standard-Struktur von Spring Projekten gehalten wurde. Die Funktionalität und Benutzbarkeit wurde von Dritt-Testern bewertet.

13.2 Fazit

Der Soll-Ist Vergleich hat gezeigt, dass das Entwicklerteam gut zusammenarbeiten kann und für weitere Projekte bestens geeignet ist. Es wäre allerdings von Vorteil, wenn die Entwickler für weitere Projekte eine Vorlaufzeit bekommen um sich in benötigte Technologien einzuarbeiten. Darüber hinaus ist davon auszugehen, dass eine persönliche Kommunikation die Synergie des Teams verstärken würde.

Anhang

Anhangsverzeichnis

Anhang 1:	Testdurchführung [Julius Figge]	29
Anhang 2:	Weitere Use-Cases [Julius Figge]	32
Anhang 2.1:	Administrator	32
Anhang 2.2:	Kontaktformular	32
Anhang 3:	GUI-Konzept [Julius Figge]	34
Anhang 3.1:	Konzept	34
Anhang 3.2:	Umsetzung	37
Anhang 4:	Projektplanung	44
Anhang 4.1:	Projektstrukturplan	44
Anhang 4.2:	Soll-Ist-Vergleich Muss- und Kann-Features	45
Anhang 5:	Schnittstellen [Philipp Röring]	47
Anhang 5.1:	Antwort /api/ideas/	47
Anhang 5.2:	Antwort /api/ideas/{id}	48
Anhang 6:	Klassendiagramme [Philipp Röring]	50
Anhang 6.1:	Model	50

Anhang 1 Testdurchführung [\[Julius Figge\]](#)

Tabelle 2: GUI-Testdurchführung

Aktion	erwartetes Ergebnis	Reaktion
Registrieren eines neuen Nutzers		
Bereits bestehenden Nutzernamen verwenden (admin)	Fehlermeldung - Nutzernamen existiert bereits	
zu kurzer Nutzernamen (<3)	Fehlermeldung - Daten falsch	
zu kurzes Passwort (<=7)	Fehlermeldung - zu kurzes Passwort	
nicht übereinstimmende Passwörter	Fehlermeldung - nicht stimmende Passwort	
mit korrekten Daten	eingeloggt sein	
Ausloggen aus dem Account	ausgeloggt sein	
Einloggen in erstellten Account		
mit falschem Passwort	Fehlermeldung	
mit falschem Nutzernamen	Fehlermeldung	
mit richtigem Passwort	eingeloggt sein	
Erstellen von beispielhaften Ideen		
Erstellen einer „internen Idee“	Idee erscheint in Tabelle nicht eingereichter Ideen	
Erstellen einer „Produkt-Idee“	Idee erscheint in Tabelle nicht eingereichter Ideen	
Erstellen einer beliebigen Idee mit fehlerhaften Werten	Fehlerhafte Attribute werden hervorgehoben	
Erstellen einer Idee von der bereits selber Name bei selbem Typ vorhanden	Fehlermeldung über Duplikat	
Bearbeiten der internen Idee	Änderungen werden übernommen	
Bearbeiten der Produkt-Idee	Änderungen werden übernommen	
Ideenübersicht		
Filtern der nicht eingereichten Ideen nach Attributen	nur Ideen mit passenden Attributen werden angezeigt	

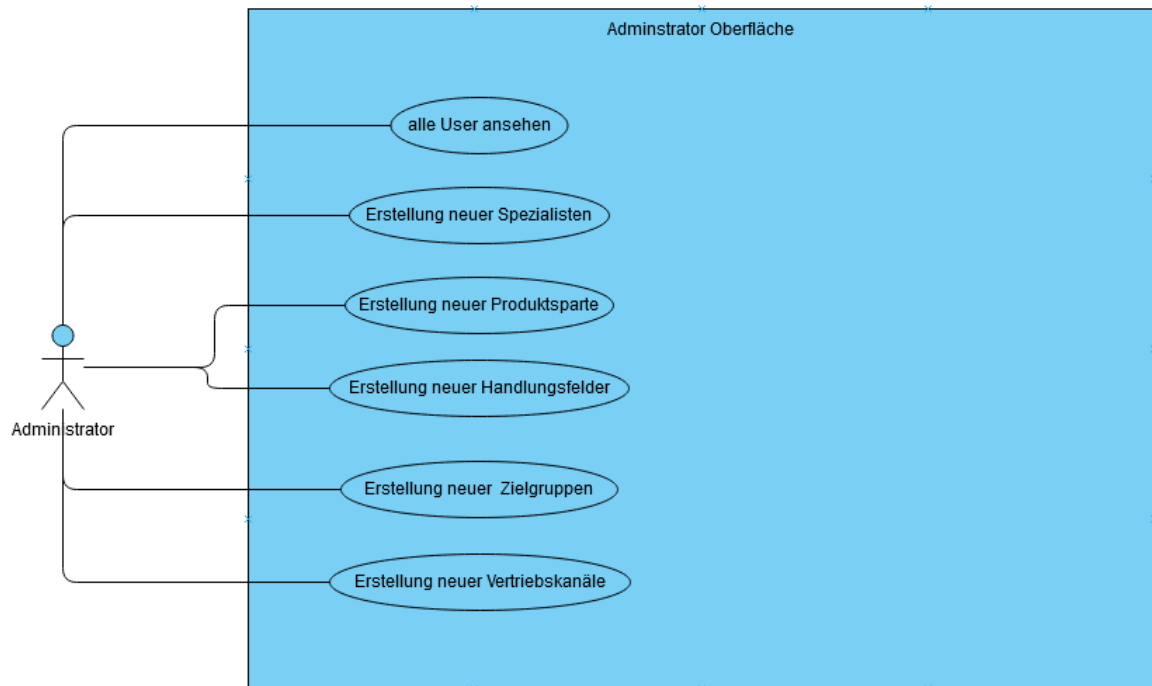
Einreichen der erstellten Ideen	erfolgreicher Transfer in jeweilige Tabelle	
Ausloggen aus dem Account	ausgeloggt sein	
Idee Übersicht als nicht eingeloggtter Nutzer		
Filtern der Ideen in beiden Tabellen	nur Ideen mit passenden Attributen werden angezeigt	
Kontaktformular		
Kontaktformular im Footer aufrufen	Kontaktformular erscheint	
Nachricht mit Titel, Nachricht und eigener E-Mail-Adresse absenden	Nachricht abgesendet	
Nachricht mit Titel, Nachricht und falsch formatierter E-Mail-Adresse absenden	Fehler	
Spezialist für „internen Idee“		
Einloggen als passender (Ideen sollten ihm zugewiesen sein) Spezialist (Zugangsdaten siehe <code>Manual.md</code>)	eingeloggt sein	
Übersicht zu entscheidender Ideen filtern	nur Ideen mit passenden Attributen werden angezeigt	
Entscheiden ohne Begründung	fehlendes Attribut wird hervorgehoben	
Idee in Ideenspeicher verschieben	Idee liegt in Ideenspeicher	
Spezialist für „Produkt-Idee“		
Account zu anderem Spezialist wechseln	Eingeloggt und Idee liegt in Ideenspeicher	
Entscheiden über Idee aus Ideenspeicher mit Auswahl „zur Entscheidung freigegeben“	Idee liegt in eigenen zu entscheidenden Ideen	
Idee aus Entscheidungsübersicht bewerten	Idee erscheint auf passender Tabelle in Ideenübersicht	
Ausloggen	Ausgeloggt und angenommene Idee mit Begründung in Ideenübersicht	
Administrator		

Account zu Administrator wechseln (Zugangsdaten siehe <code>Manual.md</code>)	Eingeloggt auf Admin-Seite	
Existierende User anschauen	registrierter Account sowie alle Spezialisten werden aufgelistet	
Neuen Fachspezialisten anlegen	Spezialist taucht in Userliste auf	
Neue Produktparte anlegen	Produktparte angelegt	
Neue Handlungsfeld anlegen	Handlungsfeld angelegt	
Neue Zielgruppe anlegen	Zielgruppe angelegt	
Neue Vertriebskanal anlegen	Vertriebskanal angelegt	
Vertriebskanal mit dem selben Titel anlegen	Fehler wird angezeigt	
Kontaktnachricht ansehen und als beantwortet kennzeichnen	Keine ungelesenen Nachrichten	
Ausloggen	Ausgeloggt	

Anhang 2 Weitere Use-Cases [Julius Figge]

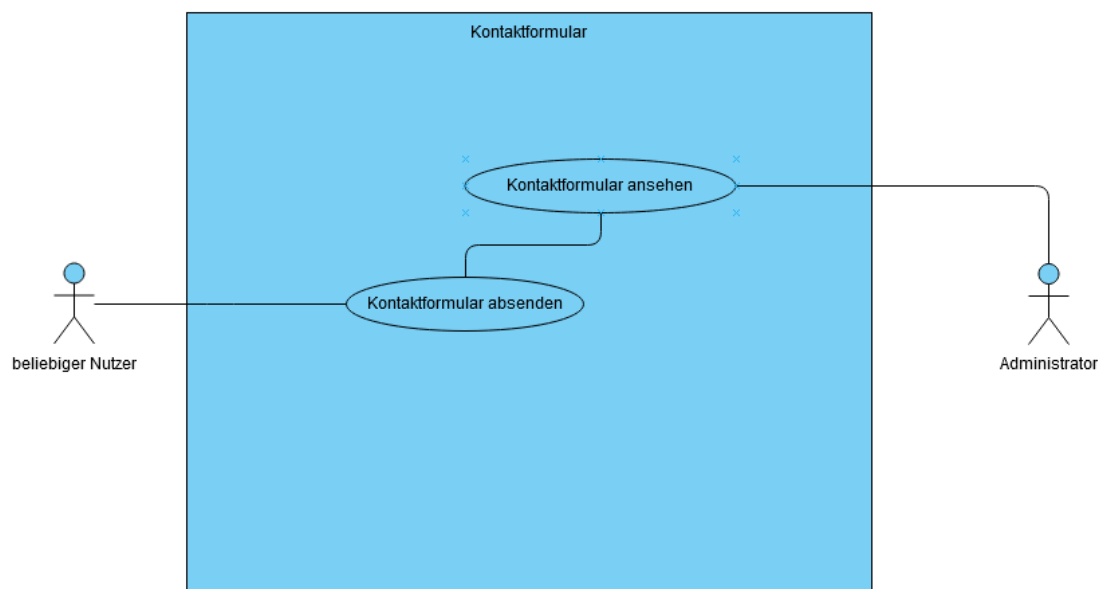
Anhang 2.1 Administrator

Abbildung 12: Administrator - Use-Case Diagramm



Quelle: Eigene Darstellung

Anhang 2.2 Kontaktformular

Abbildung 13: Kontaktformular - Use-Case Diagramm

Quelle: Eigene Darstellung

Anhang 3 GUI-Konzept [Julius Figge]

Anhang 3.1 Konzept

Abbildung 14: GUI-Konzept - Login




Digitaler
Briefkasten

Log in

[Create an account](#)


Quelle: Eigene Darstellung

Abbildung 15: GUI-Konzept - Registrierung

Digitaler
Briefkasten

Quelle: Eigene Darstellung

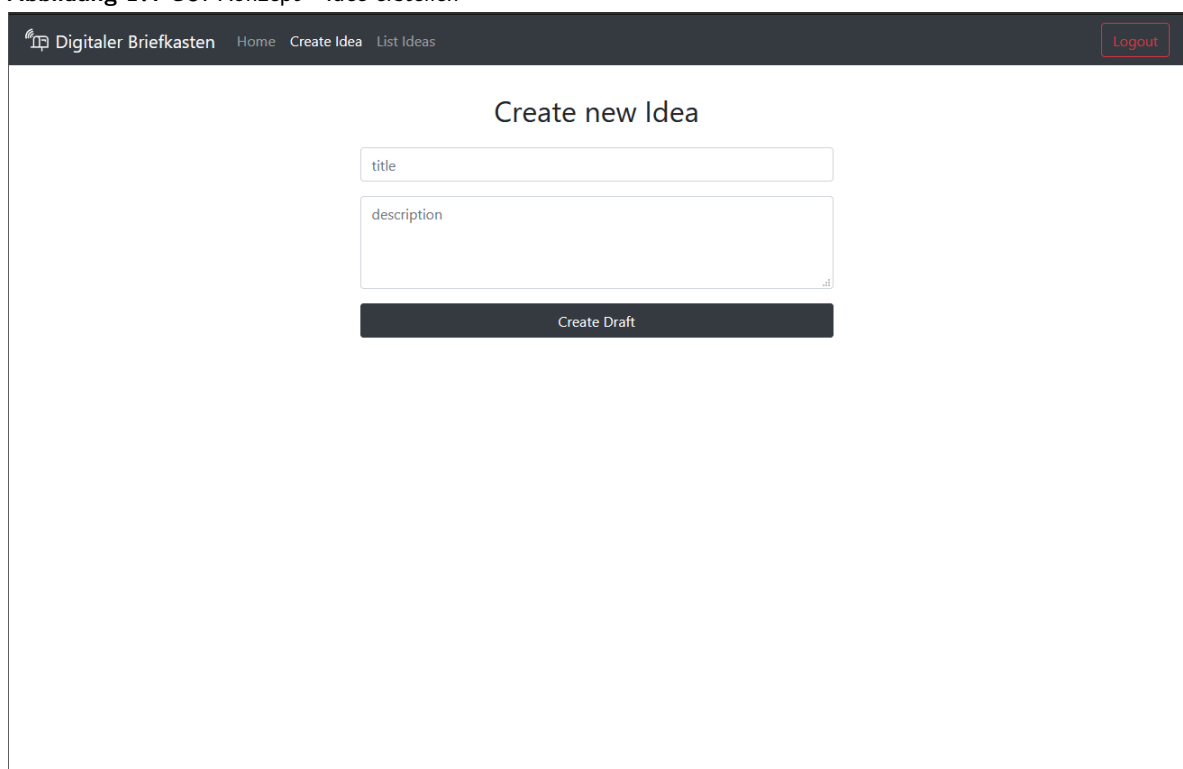
Abbildung 16: GUI-Konzept - Willkommen

 Digitaler Briefkasten Home Create Idea List Ideas [Logout](#)

Welcome konzept!
This is just a Prototype - so please be aware of that.

Quelle: Eigene Darstellung

Abbildung 17: GUI-Konzept - Idee erstellen



The image shows a web application interface for creating a new idea. At the top, there is a dark navigation bar with the text 'Digitaler Briefkasten' and a menu with 'Home', 'Create Idea', and 'List Ideas'. A 'Logout' button is in the top right corner. The main content area is titled 'Create new Idea' and contains two input fields: 'title' and 'description'. Below these fields is a dark button labeled 'Create Draft'.

Digitaler Briefkasten Home Create Idea List Ideas Logout

Create new Idea

title

description

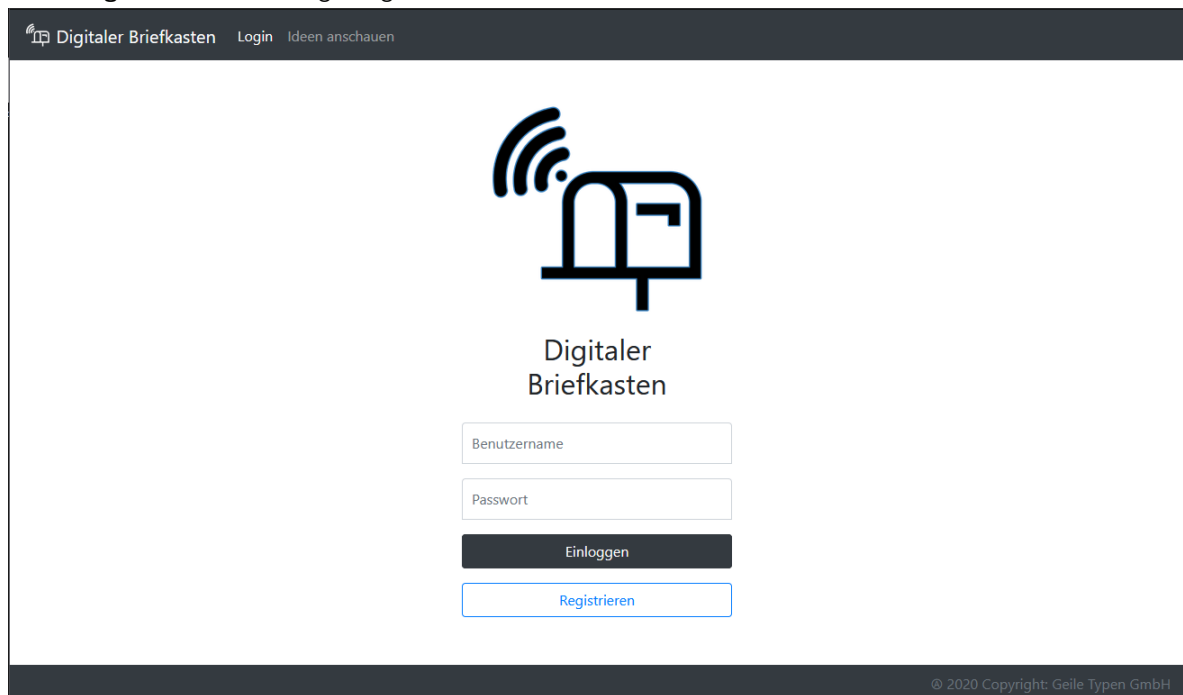
Create Draft

Quelle: Eigene Darstellung

Anhang 3.2 Umsetzung

Die im folgenden dargestellten GUI Bestandteile stellen die wichtigsten Teile der Oberfläche dar. Auf die Abbildung aller Bestandteile wurde aufgrund der zu großen Menge, zur Wahrung der Übersichtlichkeit, verzichtet.

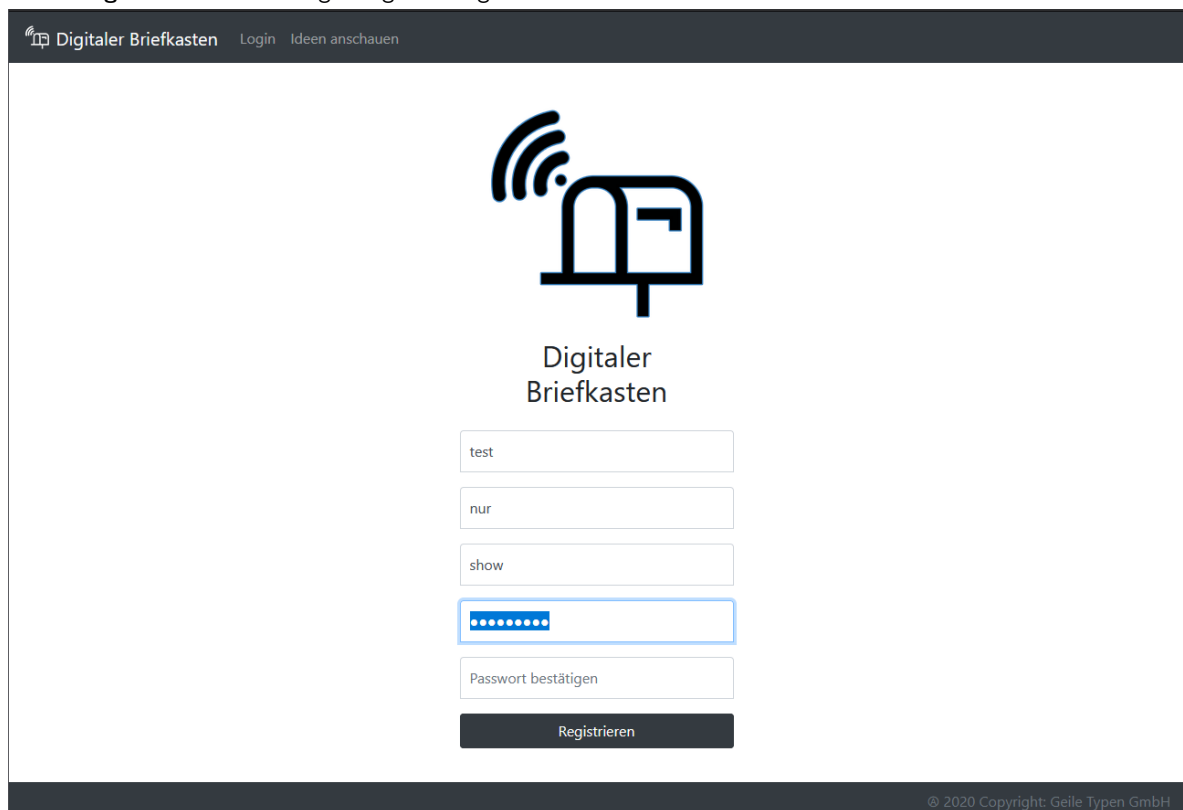
Abbildung 18: GUI-Umsetzung - Login



The screenshot shows a web application interface for 'Digitaler Briefkasten'. At the top, a dark navigation bar contains a home icon, the text 'Digitaler Briefkasten', and links for 'Login' and 'Ideen anschauen'. The main content area is white and features a large black icon of a mailbox with signal waves. Below the icon, the text 'Digitaler Briefkasten' is displayed. The login form consists of two input fields: 'Benutzername' and 'Passwort'. Below these are two buttons: a dark 'Einloggen' button and a light blue 'Registrieren' button. A dark footer bar at the bottom right contains the copyright notice '© 2020 Copyright: Geile Typen GmbH'.

Quelle: Eigene Darstellung

Abbildung 19: GUI-Umsetzung - Registrierung



Digitaler Briefkasten

© 2020 Copyright: Geile Typen GmbH

Quelle: Eigene Darstellung

Abbildung 20: GUI-Umsetzung - Ideen

Digitaler Briefkasten
 Home
 Idee erstellen
 Ideen anschauen
 Logout

Deine nicht eingereichten Ideen

	Typ	Titel	Beschreibung	Ersteller	Erstellungsdatum	Status
⋮	💡	Geheime Weltherrschaft	Na was denn sonst	Umse tzung	20.05.2020	❓ Nicht eingereicht

Produkt-Ideen

	Titel	Beschreibung	Ersteller	Erstellungsdatum	Status	Status Begründung
⋮	Eine weitere	Geheimer Test	Umse tzung	20.05.2020	🕒 auf Bewertung wartend	

Interne-Ideen

	Titel	Beschreibung	Ersteller	Erstellungsdatum	Status	Status Begründung
⋮	Beispiel	Test	Umse tzung	20.05.2020	🕒 auf Bewertung wartend	

[Nutzen: Umsetzung, Rolle: Nutzer]
 [Kontaktformular](#)
 © 2020 Copyright: Geile Typen GmbH

Quelle: Eigene Darstellung

Abbildung 21: GUI-Umsetzung - Idee erstellen

The screenshot shows a web application interface for creating a new product idea. The header bar is dark grey with the logo 'Digitaler Briefkasten' on the left, navigation links 'Home', 'Idee erstellen', and 'Ideen anschauen' in the center, and a 'Logout' button on the right. The main content area is white and titled 'Neue Produktidee erstellen'. The form consists of several input fields: a 'Titel' field, a larger 'Beschreibung' field, and three selection fields for 'Zielgruppe auswählen:', 'Vertriebskanal auswählen:', and 'Produktsparte auswählen:'. The 'Produktsparte' dropdown is currently set to 'KFZ'. Below these is a checkbox labeled 'Existiert bereits Vergleichbares (bei der Konkurrenz)?'. Underneath is a label 'Bis zu drei Vorteile eingeben:' followed by three 'Vorteil' input fields. At the bottom of the form is a dark grey button labeled 'Produkt-Idee erstellen'. The footer is dark grey and contains the text '[Nutzer: Umsetzung , Rolle: Nutzer] Kontaktformular' on the left and '© 2020 Copyright: Geile Typen GmbH' on the right.

Digitaler Briefkasten Home Idee erstellen Ideen anschauen Logout

Neue Produktidee erstellen

Titel

Beschreibung

Zielgruppe auswählen:

Vertriebskanal auswählen:

Produktsparte auswählen:

KFZ

☐ Existiert bereits Vergleichbares (bei der Konkurrenz)?

Bis zu drei Vorteile eingeben:

Vorteil

Vorteil


Vorteil



Produkt-Idee erstellen

[Nutzer: Umsetzung , Rolle: Nutzer] [Kontaktformular](#) © 2020 Copyright: Geile Typen GmbH

Quelle: Eigene Darstellung

Abbildung 22: GUI-Umsetzung - Idee ansehen

 Digitaler Briefkasten [Home](#) [Idee erstellen](#) [Ideen anschauen](#) [Logout](#)

  **Geheime Weltherrschaft**

Titel	Geheime Weltherrschaft
Beschreibung	Na was denn sonst
Produktlinie	KFZ
Vorteile	Nenene
Zielgruppen	Paare
Vertriebskanäle	Versicherungsmakler
Existiert bereits Vergleichbares?	<input checked="" type="checkbox"/>
Ersteller	Umsetzung
Erstellungsdatum	20.05.2020
Status	Nicht eingereicht
Status-Begründung	

[\[Nutzer: Umsetzung , Rolle: Nutzer \]](#) [Kontaktformular](#) © 2020 Copyright: Geile Typen GmbH

Quelle: Eigene Darstellung

Abbildung 23: GUI-Umsetzung - Admin Ansicht

The screenshot displays the 'Admin Ansicht' (Admin View) of the 'Digitaler Briefkasten' application. The top navigation bar includes a home icon, the application name 'Digitaler Briefkasten', the current page 'Adminpanel', and links for 'Idee erstellen' and 'Ideen anschauen'. A 'Logout' button is located in the top right corner. The main content area features a large grey banner with the text 'Willkommen admin!'. Below this, there are several interactive elements: a link 'Alle registrierten User ansehen', a section titled 'Fachspezialisten anlegen' which contains a form with fields for 'Username', 'Vorname', 'Nachname', 'Passwort', and 'Passwort bestätigen', a 'Fachgebiet auswählen' dropdown menu, and a 'Spezialist anlegen' button. Below the form are five more links: 'Produktparte anlegen', 'Handlungsfeld anlegen', 'Zielgruppe anlegen', and 'Vertriebskanal anlegen'. The footer contains user information '[Nutzer: admin , Rolle: Administrator]', a 'Kontaktformular' link, and a copyright notice '© 2020 Copyright: Geile Typen GmbH'.

Digitaler Briefkasten Adminpanel Idee erstellen Ideen anschauen Logout

Willkommen admin!

Alle registrierten User ansehen

Fachspezialisten anlegen

Username

Vorname

Nachname

Passwort

Passwort bestätigen

Fachgebiet auswählen:

Spezialist anlegen

Produktparte anlegen

Handlungsfeld anlegen

Zielgruppe anlegen

Vertriebskanal anlegen

[Nutzer: admin , Rolle: Administrator] Kontaktformular © 2020 Copyright: Geile Typen GmbH

Quelle: Eigene Darstellung

Abbildung 24: GUI-Umsetzung - Spezialist Ansicht

Digitaler Briefkasten Ideen bewerten Idee erstellen ▾ Ideen anschauen Logout

Willkommen SpeziusMaximus_KFZ!

Anstehende Entscheidungen

	Typ	Titel	Beschreibung	Ersteller	Erstellungsdatum	Status
		Eine weitere	Geheimer Test	Umse tzung	20.05.2020	auf Bewertung wartend

Ideen im Speicher

	Typ	Titel	Beschreibung	Ersteller	Erstellungsdatum	Status
--	-----	-------	--------------	-----------	------------------	--------

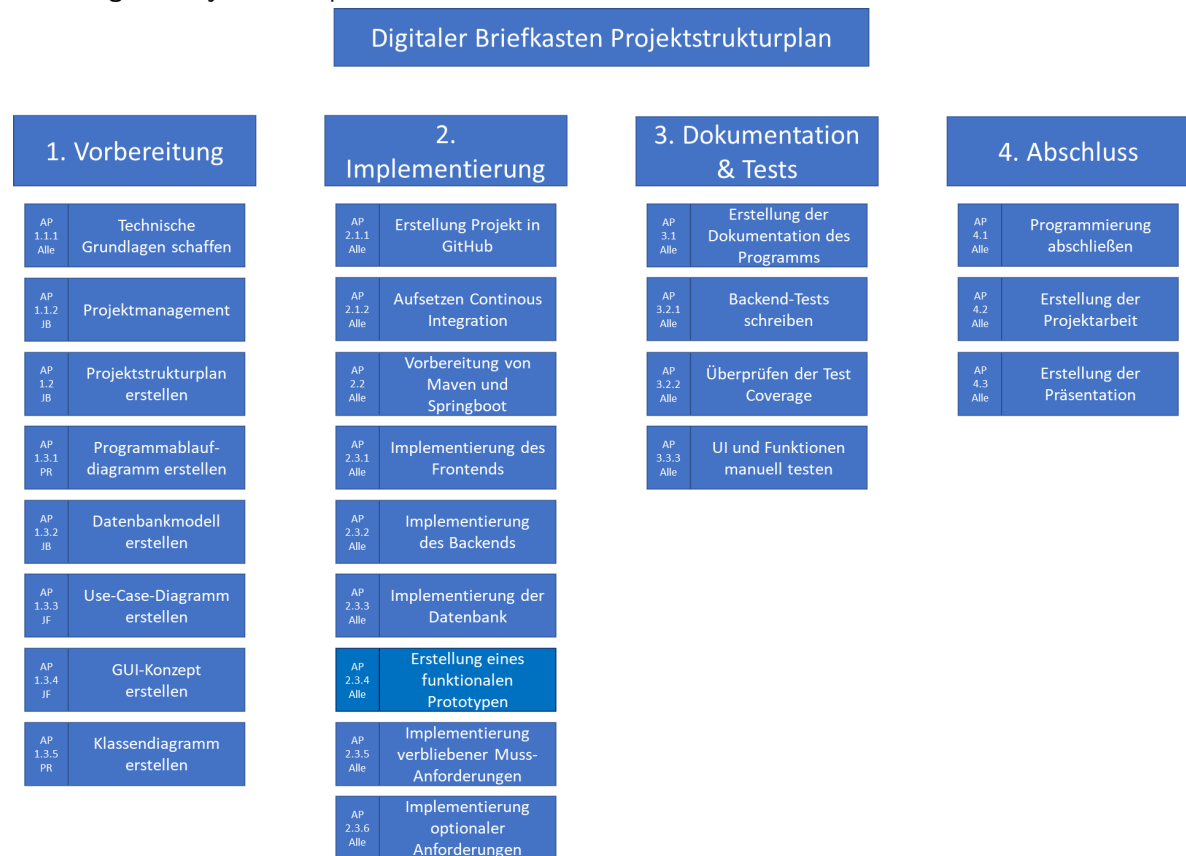
[Nutzen: SpeziusMaximus_KFZ, Rolle: Spezialist] [Kontaktformular](#) © 2020 Copyright: Geile Typen GmbH

Quelle: Eigene Darstellung

Anhang 4 Projektplanung

Anhang 4.1 Projektstrukturplan

Abbildung 25: Projektstrukturplan



Quelle: Eigene Darstellung

Anhang 4.2 Soll-Ist-Vergleich Muss- und Kann-Features

	Anforderung	Umsetzung
Muss	Noch nicht registrierte Mitarbeiter können sich am System registrieren	Umgesetzt
Muss	Registrierte Mitarbeiter können sich am System anmelden	Umgesetzt
Muss	Registrierte Mitarbeiter können neue Ideen erfassen	Umgesetzt
Muss	Registrierte Mitarbeiter können sich eine Liste ihrer eingereichten Ideen anzeigen lassen	Umgesetzt
Muss	Registrierte Mitarbeiter können ihre Ideen solange bearbeiten oder auch löschen solange dieses noch nicht zur Bewertung an einen Fachspezialisten übergeben wurden.	Umgesetzt
Muss	Nicht registrierte Mitarbeiter können vorhandene Ideen lesen, sich eine Übersicht der Ideen anzeigen lassen und die Übersicht filtern	Umgesetzt
Muss	Diese Funktionen stehen auch registrierten Mitarbeitern zur Verfügung	Umgesetzt
Muss	Neue Ideen werden Fachspezialisten zur Bewertung zugeordnet	Umgesetzt
Muss	Die Zuordnung erfolgt automatisch sobald die Idee vom registrierten Mitarbeiter zur Bewertung eingereicht wurde	Umgesetzt
Muss	Fachspezialisten können eine Idee entweder annehmen, ablehnen oder für einen späteren Zeitpunkt in einen sog. Ideenspeicher überführen / sie aus dem Ideenspeicher zurückholen	Umgesetzt
Muss	Fachspezialisten begründen ihre Entscheidung transparent und für alle sichtbar in der Anwendung	Umgesetzt
Muss	Fachspezialisten können ihnen zugewiesene Ideen in einer Liste sehen und diese Liste filtern	Umgesetzt

	Anforderung	Umsetzung
Kann	REST-API	Teilweise umgesetzt, lauffähig und erweiterbar
Kann	Kontaktformular auch unregistriert	Umgesetzt, erweiterbar um E-Mail-Einbindung
Kann	Administrator verwaltet Benutzer	Umgesetzt, erweiterbar
Kann	Dokumentenupload zu einer Idee	Nicht umgesetzt, mit Erweiterung der Datenbank umsetzbar
Kann	Profilfoto	Nicht umgesetzt, mit Erweiterung der Datenbank umsetzbar
Kann	Fachspezialist: E-Mail Benachrichtigung bei neuer Idee	Nicht umgesetzt, erfordert E-Mail-Einbindung
Kann	Benutzer: E-Mail Benachrichtigung bei Änderung einer Idee	Nicht umgesetzt, erfordert E-Mail-Einbindung
Kann	PDF-Report über erstellte Ideen quartalsweise	Nicht umgesetzt

Anhang 5 Schnittstellen [\[Philipp Röring\]](#)

Anhang 5.1 Antwort /api/ideas/

Listing 1: Antwort /api/ideas/

```

1  [
2    {
3      "id": 145,
4      "title": "Nachmieter für Häuschen in Detmold gesucht!",
5      "description": "Och joa ich habe da 'ne Idee",
6      "creator": {
7        "type": "de.fhdw.geiletypengmbh.digitalerbriefkasten.
8        persistence.model.account.User",
9        "id": 1,
10       "username": "API_USER",
11       "roles": [
12         {
13           "name": "API_USER"
14         }
15       ],
16       "lastName": "USER",
17       "firstName": "API",
18       "creationDate": "2020-05-24"
19     },
20     "creationDate": "2020-05-27",
21     "status": "NOT_SUBMITTED",
22     "productLine": {
23       "id": 2,
24       "title": "INTERNAL",
25       "specialists": []
26     },
27     "advantages": [
28       {
29         "id": 146,
30         "description": "Nur"
31       },
32       {
33         "id": 147,
34         "description": "Ein"
35       },
36       {
37         "id": 148,
38         "description": "Vorteil"
39       }
40     ],
41     "specialist": null,
42     "field": {
43       "id": 21,
44       "title": "Kostensenkung"
45     }
46   ],
47   {
48     "id": 149,
49     "title": "[Reserviert] Nachmieter für Häuschen in Detmold gesucht
50     !",
    "description": "Hmmm.. Naja irgendwas wird es schon werden.",
  }

```

```

51     "creator": {
52         "type": "de.fhdw.geiletypengmbh.digitalerbriefkasten.
53         persistence.model.account.User",
54         "id": 1,
55         "username": "API_USER",
56         "roles": [
57             {
58                 "name": "API_USER"
59             }
60         ],
61         "lastName": "USER",
62         "firstName": "API",
63         "creationDate": "2020-05-24"
64     },
65     "creationDate": "2020-05-27",
66     "status": "NOT_SUBMITTED",
67     "productLine": {
68         "id": 3,
69         "title": "KFZ",
70         "specialists": []
71     },
72     "advantages": [
73         {
74             "id": 150,
75             "description": ""
76         },
77         {
78             "id": 151,
79             "description": ""
80         },
81         {
82             "id": 152,
83             "description": ""
84         }
85     ],
86     "specialist": null,
87     "targetGroups": [
88         {
89             "id": 17,
90             "title": "Singles"
91         }
92     ],
93     "distributionChannels": [
94         {
95             "id": 14,
96             "title": "Kooperation mit Kreditinstituten"
97         }
98     ],
99     "existsComparable": true
100 }
101 ]

```

Anhang 5.2 Antwort /api/ideas/{id}

Listing 2: Antwort /api/ideas/{id}

```

1 {
2   "type": "de.fhdw.geiletypengmbh.digitalerbriefkasten.
3   persistence.model.ideas.ProductIdea",
4   "id": 149,
5   "title": "[Reserviert] Nachmieter für Häuschen in Detmold gesucht!",
6   "description": "Hmmm.. Naja irgendwas wird es schon werden.",
7   "creator": {
8     "type": "de.fhdw.geiletypengmbh.digitalerbriefkasten.
9     persistence.model.account.User",
10    "id": 1,
11    "username": "API_USER",
12    "roles": [
13      {
14        "name": "API_USER"
15      }
16    ],
17    "lastName": "USER",
18    "firstName": "API",
19    "creationDate": "2020-05-24"
20  },
21  "creationDate": "2020-05-27",
22  "status": "NOT_SUBMITTED",
23  "productLine": {
24    "id": 3,
25    "title": "KFZ",
26    "specialists": [
27      {
28        "type": "de.fhdw.geiletypengmbh.digitalerbriefkasten.
29        persistence.model.account.Specialist",
30        "id": 153,
31        "username": "SpeziusMaximus_KFZ",
32        "roles": [
33          {
34            "name": "SPECIALIST"
35          }
36        ],
37        "lastName": "Maximus",
38        "firstName": "Spezius",
39        "creationDate": "2020-05-27"
40      }
41    ]
42  },
43  "advantages": [],
44  "specialist": null,
45  "targetGroups": [
46    {
47      "id": 17,
48      "title": "Singles"
49    }
50  ],
51  "distributionChannels": [
52    {
53      "id": 12,
54      "title": "Stationärer Vertrieb"
55    },
56    {
57      "id": 14,
58      "title": "Kooperation mit Kreditinstituten"

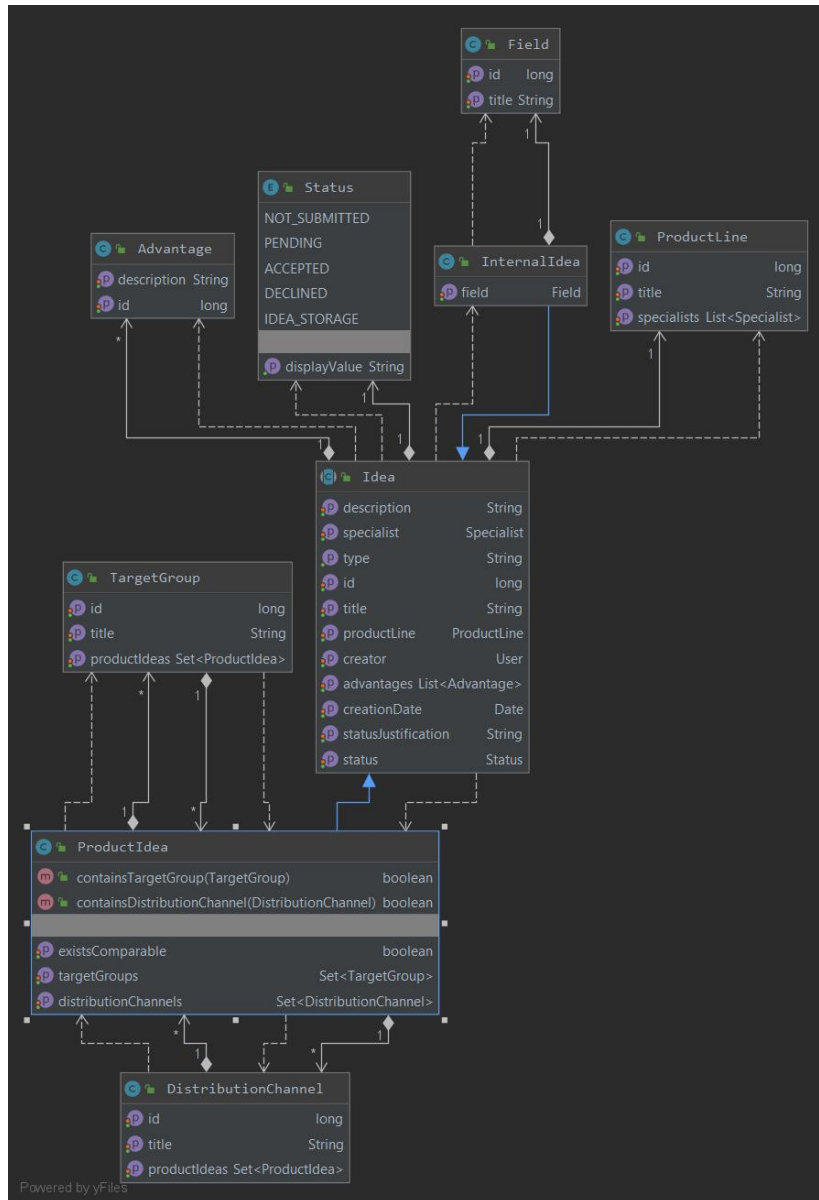
```

```
59     }  
60 ],  
61 "existsComparable": true  
62 }
```

Anhang 6 Klassendiagramme [\[Philipp Röring\]](#)

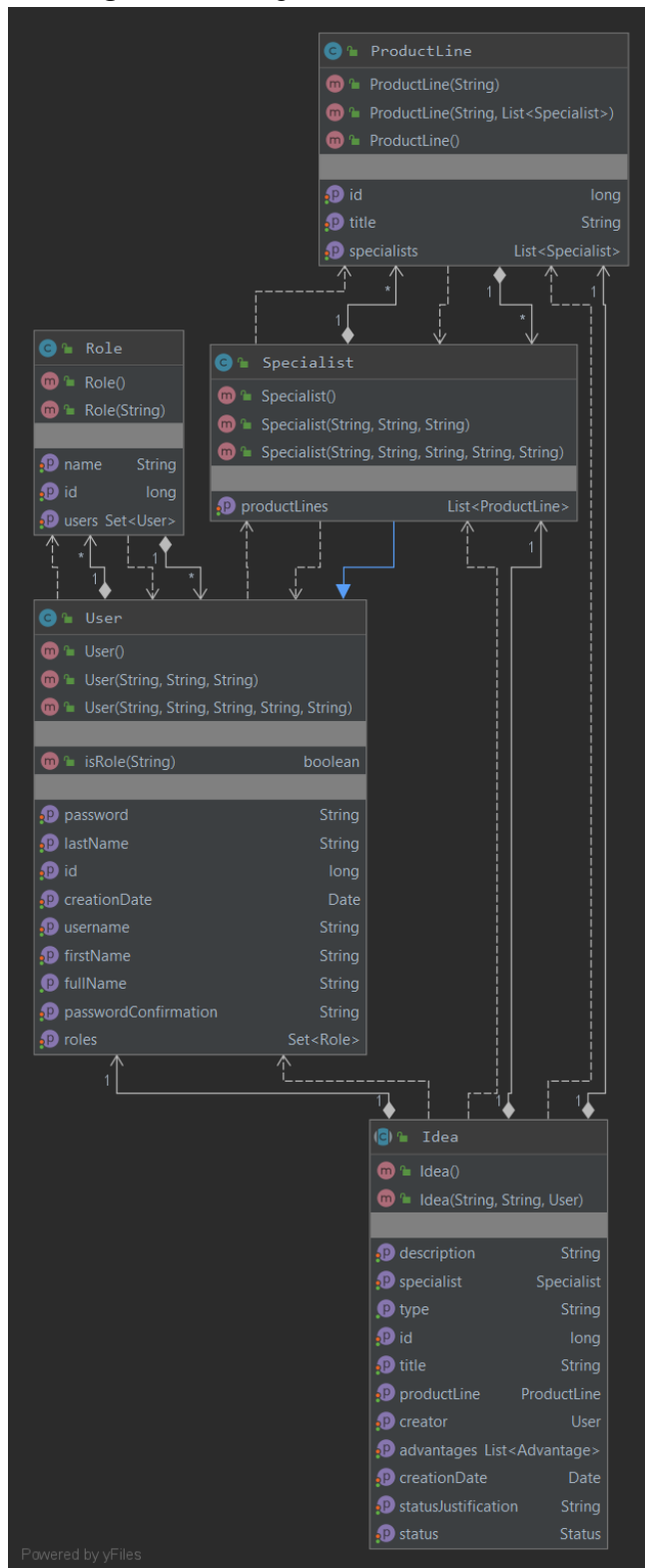
Anhang 6.1 Model

Abbildung 26: Klassendiagramm Model Ideen



Quelle: Eigene Darstellung

Abbildung 27: Klassendiagramm Model User



Quelle: Eigene Darstellung

Quellenverzeichnis

Internetquellen

Heuermann, Christian (2020). *Projektvorgaben Softwareprojekt - Meine Idee Initiative*.

Horn, Torsten (2007). *Vererbung und Polymorphie mit relationalen Datenbanken*. URL: <https://www.torsten-horn.de/techdocs/sql-vererbung.htm> (besucht am 21. Mai 2020).