Projektdokumentation M151

von Davide Marcoli am 1. Nov. 2022

Beschreibung

In meiner Applikation, dem Infoboard, können Informationen, wie zum Beispiel Anleitungen, Blogs oder Announcements geschrieben werden. Das Spezielle daran ist, dass sie mit einem WYSIWYG (**W**hat **y**ou **s**ee **i**s **w**hat **y**ou **g**et) - Editor verfasst werden und man sie nach Belieben formatieren kann. Ob Fett, Kursiv oder doch mit einem Code-Block - Kein Problem! Diesen Posts können Kategorien hinzugefügt werden, die der Admin in seinem Adminbereich erstellen, aktualisieren oder löschen kann. Sie werden klar mit einer nur einmal verwendbaren Farbe gekennzeichnet, sodass man schnell Posts in dieser Kategorie finden kann. Es gibt auch einen ausgeklügelten Suchalgorithmus, mit dem man eine Suche nach Titel, Kategorie, Inhalt oder Autor machen kann. Wenn man nicht von allen Suchmöglichkeiten aufs Mal Gebrauch machen will, kann man seine Suche mit folgenden Keywords präfixen: “title:”, “content:”, “category:” or “author:” / “@”.

Architektur

Frontend (View)

Hier interagiert der User mit unserer Applikation. Er hat hier CRUD Funktionen zur Verfügung, damit er die Daten verändern kann. Ich werde das Frontend mit Angular und Typescript umsetzen. Hier werde ich auch mit Hilfe von Services und einem HTTP Client, Anfragen an das Backend senden.

Backend (Controller)

Das Backend wird bei mir mit Spring Boot erstellt. Ich werde hier eine REST API entwickeln, auf die dann von aussen zugegriffen werden kann.

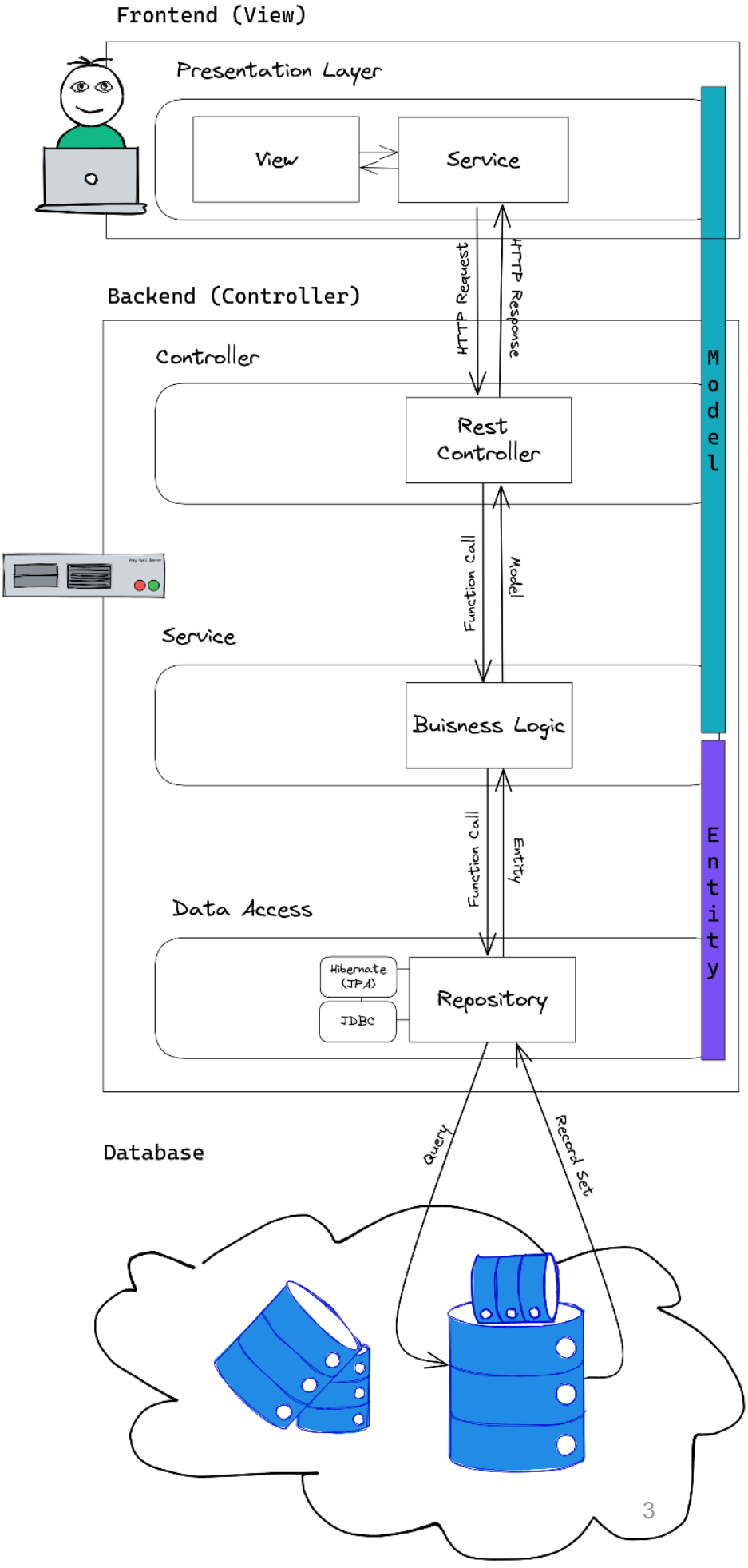
Das Backend ist in 3 Layers aufgeteilt:

Controller

Hier sind die Endpoints definiert und die Daten kommen erstmals mit dem Backend in Kontakt.

Service

Hier werden die erhaltenen Daten verarbeitet. Es werden Checks durchgeführt und Validationen gemacht.



Repository

Hier wird direkt mit Hilfe von JPA auf die Datenbank zugegriffen. Viele Queries, die für Datenbankabfragen gebraucht werden (zum Beispiel save, findAll, etc…), werden automatisch generiert. Dies ist eine tolle Erfahrung für jeden Entwickler. Es werden ebenfalls ohne das Eingreifen des Programmierers Dinge wie die SQL-Injection verhindert. Um das muss man sich hier nicht mehr kümmern.

Frameworks

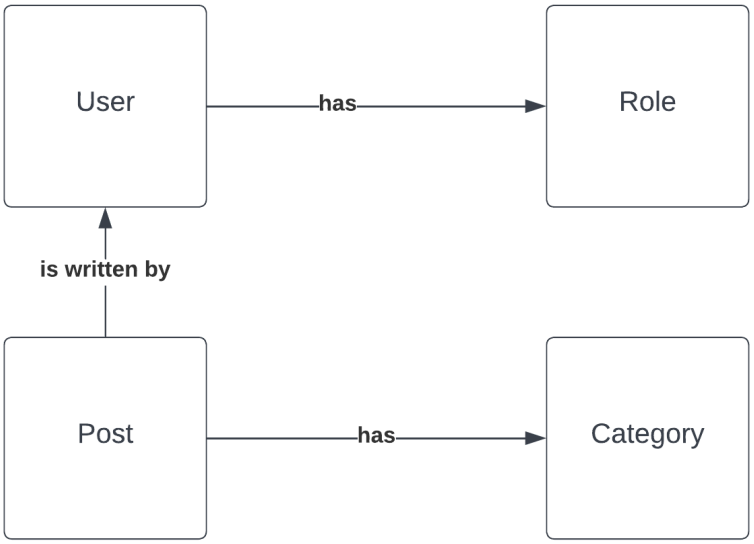
Frontend

Im Frontend verwende ich Angular Version 13.

Backend

Für das Backend benutze ich Spring Boot 2.7.5, was Spring Version 5.3.15 (hier ersichtlich) verwendet.

ERM



Dies ist eine einfache Darstellung meiner Datenbank. Dies sind meine Tabellen. Ich habe auch noch mehrere Zwischentabellen, jedoch werden diese hier nicht eingezeichnet, da sie nichts zum Verständnis der Struktur beitragen.

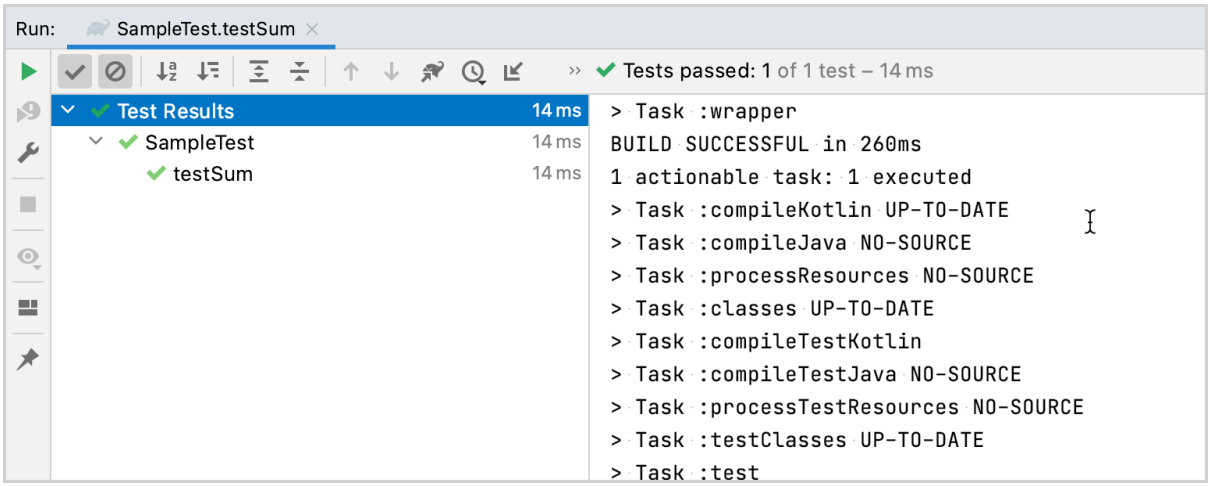
Der User kann mehrere Rollen haben. Der Post hat einen Autor und kann mehrere Kategorien haben.

Testkonzept

Es ist wichtig, die Applikation von verschiedenen Seiten zu betrachten.

Unit Tests

Es gibt einmal die Seite der Logik: Machen meine Funktionen das, was sie sollten. Hier kann man Unit Tests verwenden, die sich perfekt dafür eignen. Ich hatte in der Vergangenheit auch schon JUnit5 gebraucht, was mir eine gute Grundlage für diese Test bietet. Das Tolle an diesen Tests ist, dass sie bei jedem Build ausgeführt werden können und man so sofort sehen kann, wenn man ein Codestück mit seinen Änderungen negativ getroffen hat.



Quelle: https://kotlinlang.org/docs/jvm-test-using-junit.html#run-a-test

Es gibt eine tolle Übersicht über alle Tests, sodass man sich sofort nach dem Laufen der Tests zurechtfindet.

End-To-End-Tests

Andererseits gibt es die visuelle Seite, das Frontend. Für diese können End-to-End-Tests gebraucht werden. Hier wird überprüft, ob das Frontend korrekt lädt und wie es sich verhält.

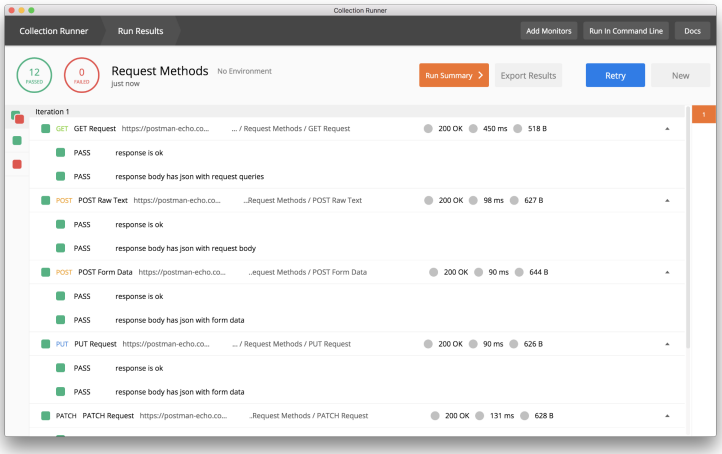
```
filters
 name_or_email_cont
 not_name_or_email_cont

admin.courses.course.pricing
 should automatically show a simplified version of the new product form when there are no products
 should allow me to add a product when there are no products
 should allow me to add a product when there are already products

admin.courses.course.sales_page
 should allow me to delete a default block
 should allow me to add back the default block
 should allow me to add a rich text block
- Failed: unknown error: Element is not clickable at point (327, 1280). Other element would receive the click: <button ng-
click="toggleInsertBlockMenu(block)" what="insert block button" class="btn btn-sm btn-default add-block-button">...</button>
(Session info: chrome=44.0.2403.89)
(Driver info: chromedriver=2.14.313457 (3d645c400edf2e2c500566c9aa096063e707c9cf),platform=Linux 3.13.0-35-generic x86_64) (WARNING: The server
did not provide any stacktrace information)
Command duration or timeout: 29 milliseconds
```

Quelle: https://medium.com/how-we-build-fedora/e2e-testing-with-angular-protractor-and-rails-725fbefb8149

Das Gute an diesem Test ist, dass man genau sieht, welches HTML-Element betroffen ist und man so schnell auf die Spur des Problems kommt.

Postman-Tests

Für das Testen der Endpoints können Postman-Tests verwendet werden. Hier kann man schauen, ob die Daten im korrekten Format zurückkommen und ob eine gute Fehlerbehandlung mit korrekten Status Codes vorhanden ist. Diese sind auch sehr hilfreich, wenn man noch kein Frontend hat, um die Daten entgegenzunehmen.



So kann man schnell überprüfen, ob noch alle Endpoints funktionieren und niemand an den Security-Configurationen herumgeschraubt hat. So kann man sofort sehen, wenn ein Endpoint verändert wurde, da der Test dann nicht mehr durch läuft.

Quelle: https://blog.postman.com/writing-tests-in-postman/

Reflektion

Ich bin sehr zufrieden mit dem, was ich in den letzten Wochen geschafft habe. Ich bin sehr stolz auf das Zusammenspiel von Frontend und Backend. Wichtigen Meldungen werden dem Benutzer ansprechend präsentiert und er weiss immer, was falsch gelaufen ist. Das Login ist sehr robust und das Rollensystem funktioniert wie gewünscht. Die Rollen unterscheiden sich sehr und ich bin froh, dass ich dies im Frontend so gut hingekriegt habe. Die Dokumentation habe ich leider erst spät angefangen (jetzt ist 21:42 Uhr 🙂). Jedoch habe ich auch die rechtzeitig fertigstellen können.