Modul 431 IPERKA

# Idee

Ich habe mir überlegt das ich eine kleine Todo Applikation machen will. Das Ziel ist, das man ein Todo erstellen, lesen, bearbeiten und löschen kann (CRUD). Was ich auch noch machen will ist das das Todo in einer Datenbank gespeichert wird. Auf der Index Seite werden alle Todos angezeigt es gibt verschieden Routen eine, um das Todo zu bearbeiten, eine, um das Todo zu löschen und eine, um ein Todo zu erstellen. Das Ganze werde ich in Java Spring Boot umsetzten. Ich mache das Projekt allein also ohne Partner.

Inhalt

[Idee 1](#_Toc103870607)

[Planung 2](#_Toc103870608)

[Create Todo 2](#_Toc103870609)

[Reade Todo 2](#_Toc103870610)

[Update Todo 2](#_Toc103870611)

[Delete Todo 2](#_Toc103870612)

[Projekt Dokumentation 2](#_Toc103870613)

# Planung

Ich werde die Planung an meinen User Stories orientiert ich habe die User Stories in vier Kategorien aufgeteilt create, read, update und delete.

Als nächste mach ich mir ein Neus Spring Boot Projekt inzwischen weiss ich auch schon, wie das geht und welche Dependencies ich brauche weiss ich auch.

### Datenbank

* Wie man eine Datenbank einbindet, muss ich mich noch informieren.
* Die Datenbank braucht aber nur eine Tabelle, weil ich nur Todos speichern will.

### Create Todo

* Eine Route, um ein Todo zu erstellen.
* Eine View, auf der man ein Todo erstellen kann, die /createtodo Route führt auf diese View.
* Wenn man auf der /createtodo Route ist und ein neues Todo erstellt muss dieses Todo in der Datenbank gespeichert werden.
* Wenn man auf der Index Seite ist, kann man auf einen Link klicken der zu der /createtodo Route führt.

### Reade Todo

* Auf der Index Seite eine View die alle Todos anzeigt.
  + Ich bin mir noch nicht sicher, wie ich das machen werde. Ich weiss ich muss mir alle Todos aus der Datenbank holen und dann in einem Modell speichern und mit Thymeleaf und einer each Schleife durch Iterieren und in einem Form anzeigen lassen. Ich bin mir aber nicht sicher ob das so funktioniert ich muss mir nochmals Gedanken dazu machen.
* Eine zweite Route, um die Todos zu sehen die /Todos heisst.
  + Ich denke ich kann, wenn ich die Viewfuction für die / route schreibe auf der alle Todos angezeigt werden auch einfach die /Todos Route hinschreiben damit ich die gleiche Viewfuction verwenden kann.

### Update Todo

* Eine Route, um das Todo zu bearbeiten.
* Auf der Index Seite gibt es einen Link der zu der /edittodo Route führt.
* Das Editierte Todo muss in der Datenbank gespeichert werden.

### Delete Todo

* Eine Route, um Todos zu löschen.
  + Wenn man ein Todo löscht, muss es aus der Datenbank gelöscht werden.
* Auf der Index Seite gibt es einen Link, um ein Todo zu löschen.

# Projekt Dokumentation

## Alle Routen

Bei meinem Todo Projekt habe ich zuerst eine neue Klasse gemacht diese habe ich todoController genannt. Wie der Name der Klasse schon aussangt handelt es sich bei dieser Klasse um meinen Controller. In dieser Klasse habe ich dann alle Routen definiert.

Begonnen habe ich mit der Index Route auf dieser soll später alle Todos angezeigt werden. Das getmapping habe ich / genannt, weil es sich um die Index Route handelt. Die Viewfunction dazu habe ich getTodos genannt, weil, um auf die Page zu gelangen muss man ein get Request machen. In dieser Viewfunction passiert noch nichts ausser das ich meine View für diese Route zurückgebe.

Als nächstes habe ich die createtodo Route gemacht auf dieser kann man später ein Neus Todo erstellen. Das getmapping habe ich /createtodo genannt und Funktion dazu heisst getCreateTodos. Auch hier gebe ich wieder die View für diese Route zurück. Bei dieser Route ist nicht nur ein Get Request möglich, sondern auch ein Post Request, weil wenn ich ein Todo erstelle, dann will ich mir nichts vom Server holen, sondern ich will etwas senden, und zwar mein Neus Todo. Also habe ich ein postmapping gemacht und die Route dazu habe ich gleich genannt wie beim getmapping /createtodo. Ich gebe beim Post Request die gleiche View zurück wie beim getmapping.

Man kann natürlich nicht nur ein Todo erstellen, sondern auch bearbeiten darum brauche ich auch eine Route, auf der man ein Todo bearbeiten kann. Bei dieser Route musste ich das gleiche machen wie bei der createtodo Route ein Get und Post mapping das die View zurückgibt. Das Get und Post mapping habe ich /updatetodo benannt, weil man das Todo bearbeiten will.

Die letzte Route, die ich noch machen musste, war die, um ein Todo zu löschen. Hier war wieder dasselbe Spiel wie bei der createtodo und updatetodo Route ich musste ein Get und Post mapping machen das ich deletetodo genannt habe und die View dazu zurückgeben.

## Hibernate, H2 und Entität

Damit ich meine Todos auch speichern kann brauche ich eine Datenbank. Ich habe in Spring Boot noch nie eine Datenbank benutzt darum habe ich zuerst geschaut welches ORM Framework ich benutzen kann. Bei meiner Suche bin ich auf Hibernat gestossen das mir gut gefallen hat. Da ich beim Erstellen des Projektes nicht wusste welche Technologien ich verwenden kann für eine Datenbank musste ich noch das package für Hibernate hinzufügen im pom File.

Ich habe auch noch ein Datenbankmanagementsystem gebraucht. Da ich die Datenbank nicht auf meinem Speicher wollte habe ich ein Datenbanksystem gesucht das die Daten im Arbeitsspeicher als Datenspeicher benutzt. Bei meiner Suche bin ich auf das H2 Datenbanksystem gestossen.

Damit ich in meiner Datenbank auch ein Tabel habe musste ich in meinem Projekt eine neue Klasse machen, die ich Todo genannt habe. In dieser Klasse habe ich mit einer Annotation gesagt das es sich bei dieser Klasse um eine Entität handelt das habe ich mit der Entity Annotation gemacht. Die Klasse hat zwei Attribute den Primary Key mit dem Namen Id und ein Attribut für den Namen des Todos das ich todoName genannt habe. In dieser Klasse habe ich dann auch einen leeren Konstruktor und Getter, Setter für die beiden Attribute gemacht.