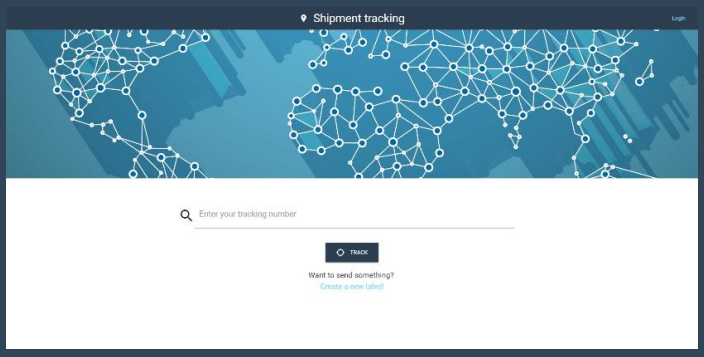
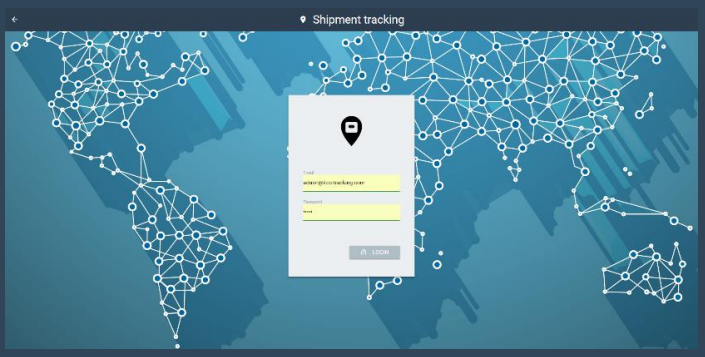
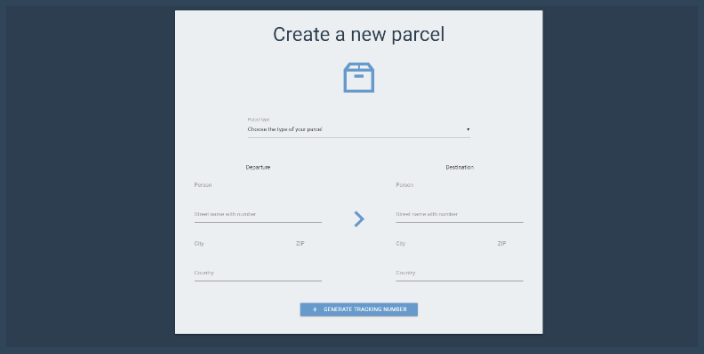
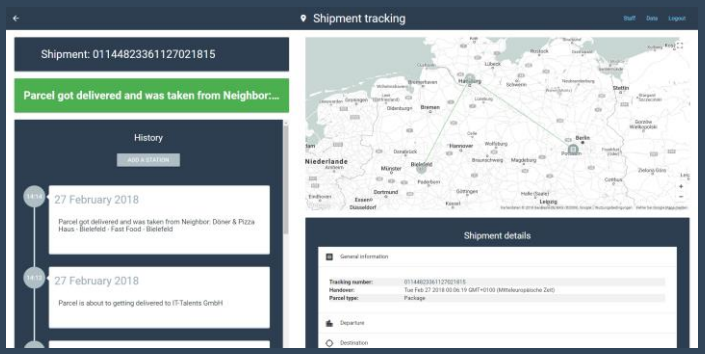
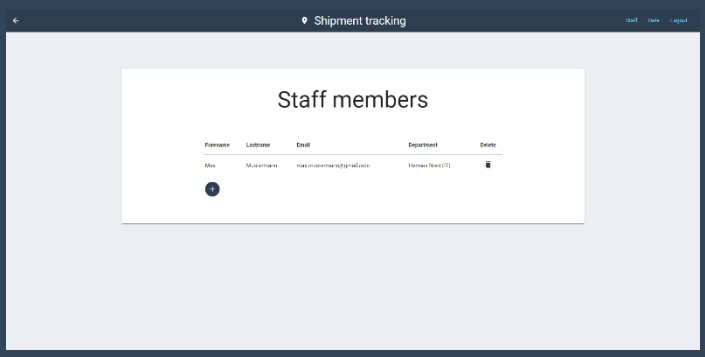
Shipment tracking

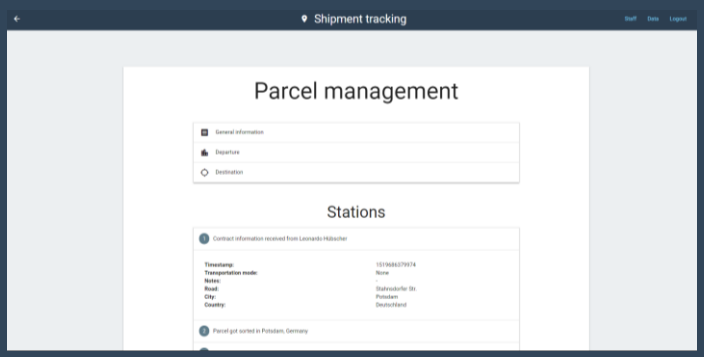












Die Startanweisungen

Um den Deploy-Prozess zu vereinfachen, habe ich eine Vagrant-Box eingerichtet, welche durch den Befehl *vagrant up* gestartet wird. Während des Startprozesses wird ein nginx-Server gestartet, welcher den Zugriff auf das Frontend ermöglicht. Des Weiteren wird am Ende das Backend gestartet.

Der erste Startvorgang dauert vermutlich etwas länger, da die Maschine erst erstellt werden muss. In dieser Phase werden die erforderlichen Dateien für die Linux-Distribution (12.0.4 LTS Precise), die MongoDB-Instanz, Java 8 sowie Nginx herunterladen. Je nach Internetverbindung kann dieser Prozess ein paar Minuten dauern.

Nachdem die Maschine installiert wurde, benötigt das Backend etwa eine halbe Minute um zu starten. Das Frontend informiert den Nutzer, ob das Backend läuft, oder nicht.

Ports
Die Website ist nun von der Hostmaschine unter der Adresse **http://localhost:4200** erreichbar. Das Backend läuft auf Port 2018 und die MongoDB-Instanz läuft auf dem Standardport 27017.

Zugangsdaten
Um sich auf der Website einzuloggen, muss man folgende Daten in die **Login-Form** eintragen:

E-Mail: admin@itcc-tracking.com
Password: admin

Die Features

- Einfaches Tracking einer Sendung
- Leichtes Erstellen von neuen Labels für weitere Sendungen
- Umfassende Details zum aktuellen Status einer Sendung – inklusive einer GoogleMaps, einem Verlauf und weiteren Sendungsdetails
- Rechtverwaltung – Der Admin kann weiteren Mitarbeitern Zugriff auf die Plattform gewähren, um den Status von Sendungen ändern zu können
- Flexibles Datenmodell – Der Admin kann weitere Optionen für Transportmittel, Sendungsarten oder Sendungsaktionen erstellen oder entfernen

Der Tech-Stack

Da ich schon etwas erfahrener in dem Umgang mit Java bin, habe ich mich dazu entschlossen, für das Backend ebenfalls Java in Kombination mit dem **Spring-Framework** zu verwenden. Ich verwende das Spring-Framework seit kurzem auch für ein anderes Projekt und so war diese Competition die ideale Gelegenheit mehr Erfahrung in dem Umgang zu gewinnen. Um die Übersichtlichkeit des Codes zu verbessern, verwende ich das **Lombok-Plugin**, welches Getter- und Setter-Methoden im Precompiling generiert. Als Lint-Checker verwende ich SonarLint und den von IntelliJ eingebauten Lint-Checker. Wenn man die Quelldateien selbst kompilieren möchte, so ist es erforderlich das Lombok-Plugin zu installieren.

Für die Persistierung der Daten verwende ich eine **MongoDB**-Instanz. Ich habe zuvor noch keine bis kaum Erfahrungen mit Nichtrelationalen Datenbanken gemacht und wollte diese einfach mal ausprobieren.

Für das Frontend habe ich das **Angular2-Framework** verwendet, welches auf Typescript basiert. Als Bootstrap kommt das **Materializecss-Framework** zum Einsatz. Mit diesem habe ich schon etwas Erfahrung, sodass mir der Einstieg hier etwas leichter fiel.

Das Backend

Das Backend ist ein REST-Microservice, welcher grobgesehen aus drei Teilen besteht: Controller, Service und Model. Der Controller ist dafür verantwortlich die Domain Transfer Objects (DTO´s), welche über eine Route gesendet werden zu extrahieren, die Rechte zu überprüfen und die Eingaben zu validieren. Anschließend werden die Informationen an den Service weitergeleitet, welcher für alle Aufgaben rund um die Datenbank verantwortlich ist. Das Model ist der „rein“ logische Teil, der auch ohne dem Spring-Framework funktionieren würde. Dieser wird im Gegensatz zu den anderen beiden Teilen getestet.

Der REST-Client ist weiter unterteilt in Authorization, FixedData, ParcelManagement und StaffMangement.
Bei der Authorization werden wie der Names bereits suggeriert, die Zugangsdaten überprüft. Hier findet grob gesehen alles statt, was mit der Restriktion von Funktionen zu tun hat. Außerdem enthält die Authorization eine Ping-Methode, mit Hilfe derer das Frontend testet, ob das Backend läuft.

Die FixedDate-Komponente ist hingegen für alle statischen Daten zuständig - hiermit wird der Admin befähigt, die zur Auswahl stehenden Sendungsarten (Paket, Sportausrüstung etc), Transportmittel (Fähre, Bahn, Transporter) und Actions (Sendungsinformationen erhalten, Sendung sortiert in … etc) zu verwalten. Er kann neue hinzufügen oder alte löschen.

Die StaffMangement-Komponente kümmert sich hingegen um alles, was mit den Mitarbeitern zu tun hat – es können neue Mitarbeiter hinzugefügt, oder alte entfernt werden.

Das ParcelMangement kümmert sich um alles was mit der Sendung an sich zu tun hat. Es können neue Labels/ Sendungsnummern angefragt werden und Stationen zu Paketen hinzugefügt werden.

Die Datenbankarchitektur

Eine Mongo-Datenbank ist in mehrere Collections untergliedert. Jede Collection wird durch ein Repository im Spring-Framework repräsentiert. Die statischen Daten werden folgendermaßen gespeichert:

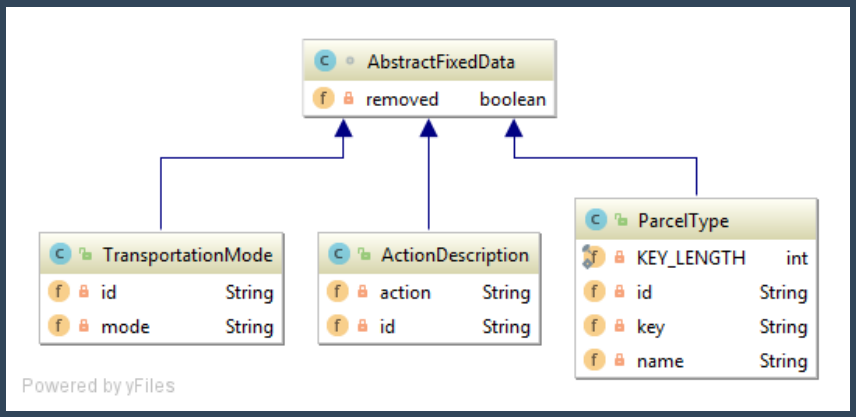


Abbildung 1Datenlayout der statischen Daten

Standardmäßig haben alle Objekte eine ID, da diese als Fremdschlüssel verwendet werden. Zudem hat jedes Feld ein „removed“-Flag, denn von Admin gelöschte statische Daten werden nicht wirklich gelöscht, damit ältere Sendungen weiterhin vollständig richtig angezeigt werden.

Für die Benutzerverwaltung werden die folgenden zwei Datenbankobjekte benötigt:

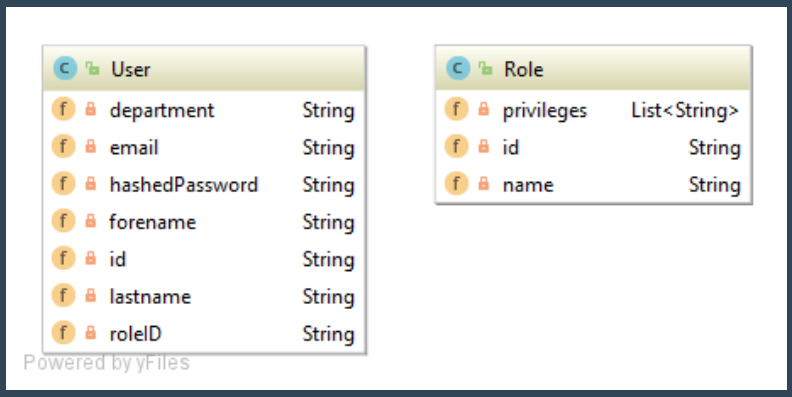


Abbildung 2Datenbankobjekte für User-Management

Im User werden alle erforderlichen Daten für den Login und ein paar zusätzliche Informationen für die Anzeige gespeichert. Es gibt momentan zwei Roleobjekte: Staff und Admin. Für beide Rollen werden die autorisierten Privilegien gespeichert. Dadurch wird letztendlich bestimmt, was ein Staff darf und welche Rechte ein Admin hat.

Nach einem erfolgreichen Login, wird für die aktuelle Session ein Authorization-Token generiert, welches an den Nutzer gebunden ist. Dieses erlaubt es, dass es nicht erforderlich ist die Credentials bei jeder Anfrage mitzusenden. Ein Token läuft nach einer festgelegten Zeit von zehn Minuten nachdem keine Aktion mehr vom Nutzer ausgeführt wurde, ab. Alternativ wird ein Token gelöscht, sobald sich ein Nutzer selbst abmeldet.

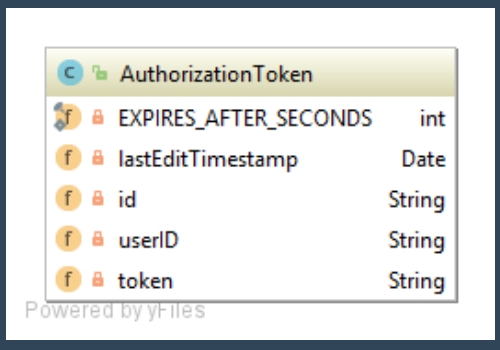


Abbildung 3 Authorization-Token Datenbankobjekt

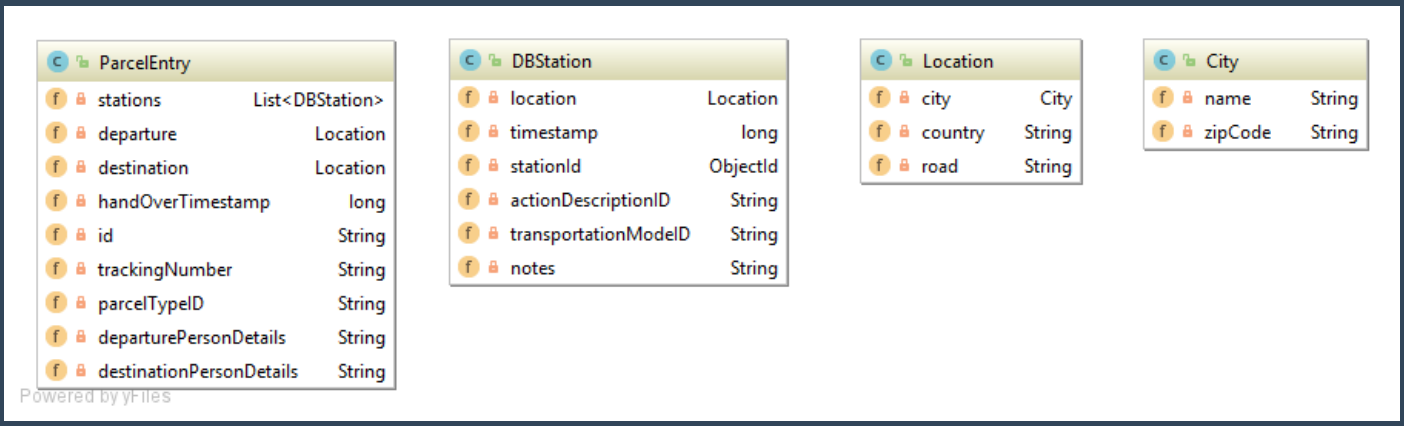
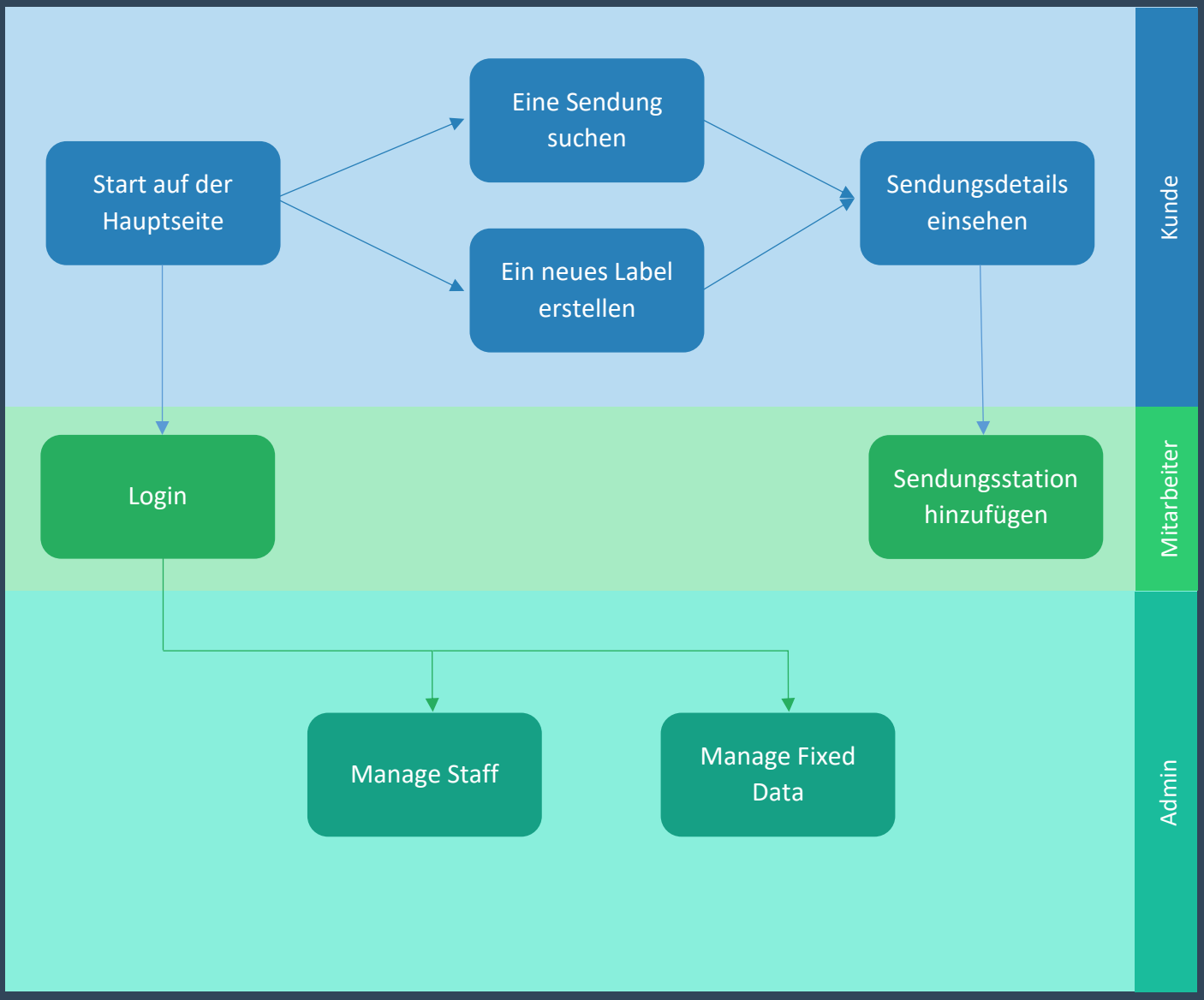


Abbildung 4 Datenbankobjecte für das Abspeichern einer Sendung

Die Datenobjekte für eine Sendung sind etwas komplexer. Eine Sendung besteht aus einer Absende- und Empfängeradresse, einer Sendungsverfolgungsnummer, hat einen bestimmten Typ und einen Zeitstempel, wann das Label erstellt wurde. Zudem durchläuft eine Sendung mehrere Stationen. Eine Station besteht wiederum aus einem Ort, einem Zeitstempel wann diese Station erreicht bzw. durchlaufen wurde, und welche Aktion mit welchem Transportmittel durchgeführt wurde. Zudem ist es möglich weitere Notizen hinzuzufügen, welche für interne Zwecke bei Hermes oder für externe Zwecke genutzt werden können.

Das Frontend/ Der Workflow



Die grobe Architektur des Frontends

Die Angular-Architektur lässt sich grob in vier Kategorien unterteilen: Services, Datenobjekte und Hilfsmethoden, Routing und die eigentlichen Komponenten.

Die Datenobjekte repräsentieren die DTO´s und sind für die Kommunikation mit dem Server erforderlich. Diese werden nachfolgend nicht weiter betrachtet. Die Services enthalten jegliche Methoden, um das REST-Backend anzusprechen. Sie sind dafür verantwortlich, dass die Komponenten die Daten vom Server abrufen können. Es gibt vier Services (Authorization, DataMangement, ParcelMangement und StaffMangement), für jede Controller-Komponente genau eine.

Das Routing kümmert sich darum, die entsprechenden Komponenten, abhängig von der aktuellen URL, anzuzeigen und die Rechte auf der Frontend-Seite zu überprüfen. Die eigentlichen Komponenten stellen das HTML-Interface für den Nutzer bereit.

Zusätzlich zu den standardmäßigen Bibliotheken, wurden folgende Komponenten hinzugefügt:

- Materializecss – Eine Bootstrap-Bibliothek für Komponenten im Materialize-Design
- ngui/map – Eine Komponente , mit Hilfe derer man Google Maps einbinden kann
- angular-vertical-timeline – Eine Komponente, um einen vertikalen Zeitstrahl anzuzeigen
- angular2-text-mask – Eine Komponente, um Nutzereingaben einzuschränken

Detaillierter Workflow anhand eines Beispiels

In diesem Abschnitt möchte ich eine Interaktion mit dem System vom Frontend bis hin zum Backend detaillierter erklären. Dies werde ich anhand des Mitarbeiterworkflows demonstrieren.

Zunächst muss sich ein Mitarbeiter authentifizieren. Dazu ruft er die Login-Seite auf und gibt seine Credentials ein. Beim Senden wird nun im Authorization-Service im Frontend das Password mit SHA-256 gehasht und es wird ein GET-Request an die Route /authorize mit den Zugangsdaten gesendet.

Der Authorization-Controller im Backend empfängt diese Daten und leitet diese an den AuthorizationService im Backend weiter. Der Service hasht nun erneut das (zuvor bereits gehashte Passwort) und gibt nun den User zurück, der die entsprechenden Credentials hat. Wenn die Zugangsdaten stimmen, so wird ein neues Authorization-Token für den Nutzer generiert und an das Frontend zurückgegeben.

Zurück im Frontend wartet die Login-Komponente bereits auf die Response vom Server. Der Service vom Frontend speicher speichert das Authorization-Token als Cookie ab, um es bei zukünftigen Abfragen mitsenden zu können. Der Nutzer wird nach Erhalt der Antwort, von der Login-Komponente zur vorherigen Seite weitergeleitet.

Der Nutzer ist nun authentifiziert.

Weitere Dokumentation auf **Github**

Ich habe für dieses Projekt ein Github-Repository angelegt und mit Issues gearbeitet. Folgende Issues sind für das Nachvollziehen meiner Entscheidungen von besonderem Interesse:

https://github.com/deeps96/ITCC-Tracking/issues/14	Lifecycle of a parcel
https://github.com/deeps96/ITCC-Tracking/issues/3	Package history
https://github.com/deeps96/ITCC-Tracking/issues/2	Thinking about the tracking number schema

Ausblick

Momentan ist das Design der Website nur für Desktopgeräte mit einer FullHD-Auflösung optimiert. Mit etwas mehr Zeit, könnte man das Design responsive machen und somit ebenfalls für Mobilgeräte optimieren.

Zusätzlich könnte man Optionen einbauen, bei denen der Nutzer bei Statusänderungen benachrichtigt wird, oder einen Ablage-/ Terminwunsch hinterlegen kann.