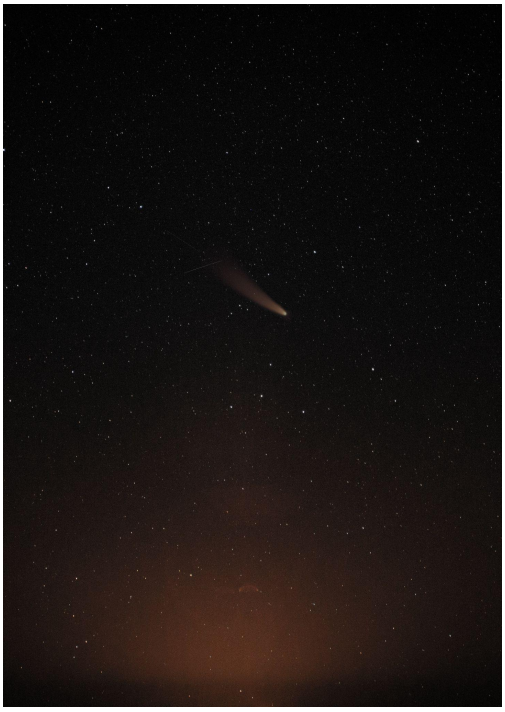


CLOUD AND DISTRIBUTED COMPUTING

Software architecture documentation

EXPOSEE

Diese Dokumentation beschreibt die während des Kurses entwickelte Applikation „Astronomer“ sowie deren zugrundeliegende Architektur. Die Anwendung bindet unter anderem eine öffentliche API der NASA ein um anzuzeigen, ob und welche Asteroiden am Nachthimmel vorbeiziehen. Die Anwendung liefert hierbei Daten wie beispielsweise den Durchmesser sowie die Entfernung zur Erde. Ein umschalten zwischen verschiedenen Maßeinheiten, das Erstellen eines Profils sowie das Speichern und beobachten von bestimmten Asteroiden zählen dabei zur Hauptaufgabe der Webanwendung und soll Hobbyastrologen und interessierte bei der Beobachtung des Nachthimmels unterstützen.

Felix Jacob | Patrick Schuster | Nicole Wagner | Adrian Ebert

Cloud and Distributed Computing
Prof. R. Oberhauser

Inhaltsverzeichnis

1. Einführung und Ziele

PS

Zielsetzung der Anwendung ist es, einem Nutzer, welcher interessiert an Astronomie ist, eine Plattform zu bieten. Mit dieser soll es ihm ermöglicht werden, von der NASA erfasste Daten für eigene Zwecke anzuschauen und zu speichern. Grundstein der Anwendung ist eine der von der NASA öffentlich zugänglichen API genannt “Asteroids - NeoWs”. NeoWs steht für “Near Earth Object Web Service” und ist eine REST-basierte API. Die zurückgelieferten Daten zeigen dem User nicht nur verschiedenste Eigenschaften von Asteroiden an, wie unter anderem den Namen oder den Durchmesser, sondern ebenfalls die Entfernung des Himmelskörpers zur Erde.

Als Erweiterung stehen dem Anwender unterschiedliche Anwendungsfunktionen innerhalb der Applikation zur verfügung. Ist der Benutzer in der Anwendung registriert und angemeldet hat er die Möglichkeit, nicht nur Asteroiden zu suchen und zu speichern, sondern ebenfalls die Maßangaben vom analog amerikanischen Messsystem ins metrische System zu übertragen. Er kann sich auf Wunsch eine Karte anzeigen lassen, in der ein aktueller Stand des Nachthimmels zu sehen ist. Möchte der Benutzer mehr über einen bestimmten Asteroiden wissen, so kann er ihn sich direkt in der “NASA JPL Small-Body Database” anzeigen lassen, welche genauere Informationen sowie ein Bild der Umlaufbahn des Kometen liefert.

Ziel der Anwendung ist es, dem Benutzer eine Einfache Übersicht über den Nachthimmel und dessen Bewegungen zu geben. Dies kann nicht nur für Astrofotografen eine enorme Hilfe sein, auch werden durch die Anwendung Hobbyastrologen unterstützt, die mit den Angaben aus der API ihr Teleskop besser kalibrieren können.

1.1. Aufgabenstellung

In Begleitung an den Kurs “Cloud and Distributed Computing” war die Anforderung, eine Plattformübergreifende Anwendung zu entwickeln, welche sich nicht nur auf einem Computerbrowser, sondern ebenfalls auf einem mobilen Browser öffnen und bedienen lässt. Vorgabe ist, dass die Anwendung mit gängigen Systemen und Entwicklungssprachen aus der Softwareentwicklung aufgesetzt und entwickelt wird. Die empfangenen Daten sollen möglichst in Echtzeit empfangen werden. Dies soll mit Hilfe einer API realisiert werden. Als zugrundeliegende Entwicklungsmethode wird in Sprints, mit entsprechenden Milestone-präsentationen gearbeitet, sodass der *Project Owner* im laufe des Semesters den aktuellsten Entwicklungsfortschritt mitbekommt. Als Dokumentation kommt während den einzelnen Sprints Trello zum Einsatz.

2. Randbedingungen

2.1. Technische Randbedingungen

PS

Randbedingung	Erklärung
Hardware-Vorgaben	Soweit waren keine Hardware-vorgaben definiert. Limitiert ist das system lediglich auf die Zuverlässigkeit und die Verfügbarkeit der bwCloud SCOPE.
Software-Vorgaben	Die Anwendung soll auf mobilen sowie Desktop-Browsern verfügbar sein.
	Jeder Microservice soll einen eigenen Docker Container erhalten.
	Es sollen Daten in Echtzeit von einer API geliefert werden.
	Als Backend System wurden verschiedene Optionen gegeben, siehe hierfür Lösungsstrategie.
	Zu Dokumentationszwecken der API soll Swagger verwendet werden.
	Als Datenbanksystem soll eine NoSQL-Datenbank zum Einsatz kommen.
	Package Namen sollen mit dem Vorsatz “de.hsaalen.” versehen werden.
	Um eine klare Projektstruktur zu erhalten soll auf BitBucket das Trello-board verwendet werden, sodass der PO jederzeit über den aktuellen Stand bescheid weiß.
	Um eine genaue Code-versionierung zu bekommen, soll das Tool Git verwendet werden.
Programmiervorgaben	Der Programmcode soll in Englisch kommentiert werden. Funktionen und Funktionsabschnitte sollen entsprechend

	im Code beschrieben werden.
Datenstruktur	Alle Informationen die von der API zurückgeliefert werden, sollen neben dem analog Amerikanischen Messsystem ebenfalls in das Metrische System konvertiert werden (Benutzerentscheidung)
	Bei einer Systemanmeldung sollen vom Nutzer Benutzernamen, E-Mail und ein selbstgewähltes Passwort abgefragt werden, welches nicht im Klartext in der Datenbank gespeichert werden soll.

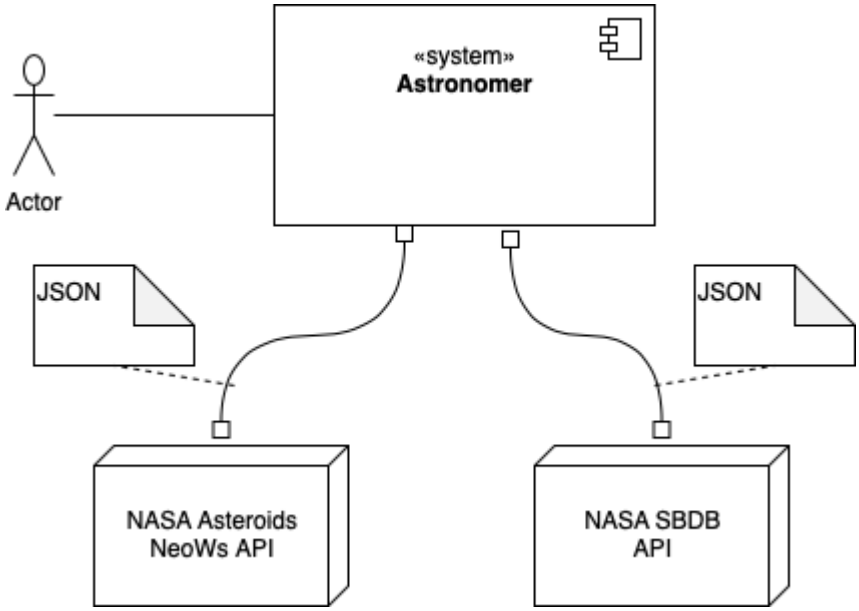
-PS

3. Kontextabgrenzung

3.1. Technischer Kontext

NW

Die folgende Abbildung zeigt die Abgrenzung des systems zu seiner Umwelt und welche Schnittstellen existieren.



-NW

3.2. Externe Schnittstellen

3.2.1. NASA Asteroid NeoWs API

https://api.nasa.gov/

3.2.2. NASA-JPL Small Body Database

https://ssd.jpl.nasa.gov/sbdb.cgi

3.2.3. Fireball-API

https://ssd-api.jpl.nasa.gov/doc/fireball.html

4. Lösungsstrategie

PS

Entwickelt wurde die Anwendung unter zuhilfenahme folgender Technologien und Entwicklungssprachen, ausgehend von der Aufgabenstellung. Das Team hat sich aus gründen des einfachen Zusammenspiels zwischen den unterschiedlichen Services auf eine Entwicklung im sogenannten "MEAN-Stack" festgelegt. Bei dem Akronym handelt es sich um die zugrundeliegenden Technologien der Applikation, auf die im folgenden kurz genauer eingegangen wird:

Das Frontend sowie die UI wurden mit Hilfe des TypeScript-basierten Frameworks Angular entwickelt. Die für die UI verwendeten Design Elemente stammen aus einer Komponentenbibliothek namens Angular Material und sind stark an das Design einer Google Applikation angelehnt. Angular Material bietet zudem die Möglichkeit, das Design selbstständig und einfach anzupassen, weshalb sie von den Entwicklern als Design Bibliothek herangezogen wurde. Angular als Framework wurde zudem im Kurs "Software Architecture" vorgestellt, wodurch bereits ein erster Kontakt mit dem Framework hilfreich war.

Als zugrundeliegende Datenbank kommt die NoSQL-Datenbank MongoDB zum Einsatz.

Als Basis-Plattform für die Applikation wird während der Entwicklung auf Node.js im Zusammenspiel mit Express.js gesetzt, welche das Entwickeln von Webapplikationen vereinfacht.

Um die Anwendung vereinfacht zu Hosten werden mit Hilfe von Docker einzelne Container mit Images erstellt und anschließend in der bwCloud SCOPE (Science, Operations and Education) gehostet. Der Landesdienst stellt einen kostenlosen IaaS für Studierende und Mitarbeiter von Hochschule und Universitäten aus Baden-Württemberg zur Verfügung.

Für ein vollumfängliches und automatisiertes Testing aller Funktionen im Frontend kommt das End-to-End Testing Framework Cypress zum Einsatz. Es bietet, im vergleich zu anderen Frameworks wie Selenium, den Vorteil eines komplettpakets aller nötigen teilkomponenten und profitiert von einer einfacheren Handhabung auf verschiedenen Browser Umgebungen.

-PS

AE

Bei der Fireball Anwendung sollte eine Map erstellt werden welche die Orte den Zeitpunkt und die Stärke des Verglühendes Asteroiden visualisieren soll. Diese wurde in Python geschrieben da es eine Komplettpaket bittet. Die Anwendung gibt Daten zu verglühten Asteroiden in der Erdatmosphäre von dem 01.01.1900 bis zum 15.05.2021 die größe der

roten Kreise stellt die Stärke in kt (Kilotonne) des Asteroidens dar. Zudem ist die Karte interaktive dabei kann man über die Kreise hovern und man bekommt Informationen zu den Asteroiden angezeigt. Python wurde verwendet da es Bibliotheken direkt von der NASA existieren die eine einfache Implementierung ermöglichen.

-AE

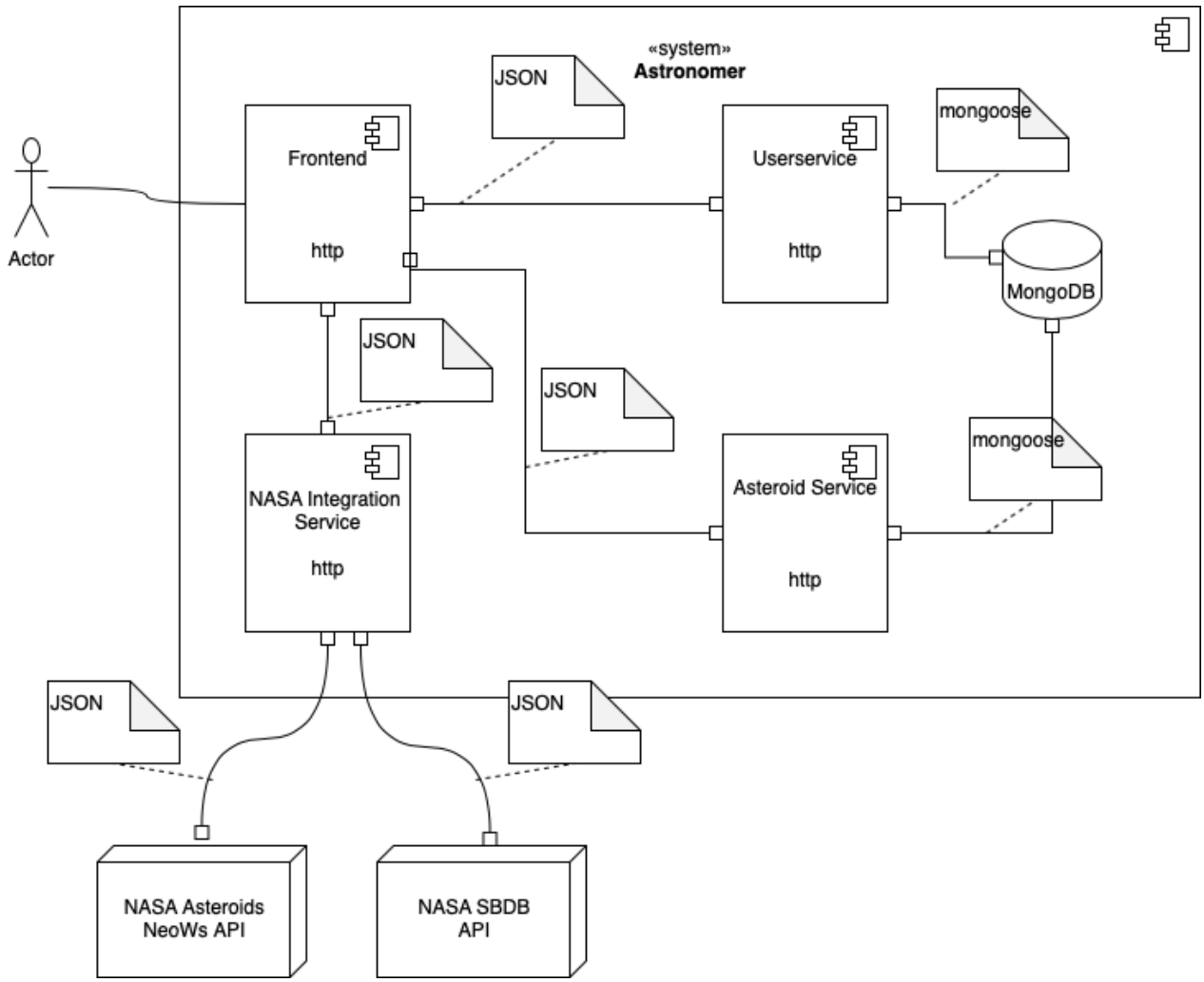
5. Bausteinsicht

5.1. Whitebox Gesamtsystem

NW

Übersichtsdiagramm

Die folgende Abbildung zeigt die inneren Bestandteile von dem System und deren Abhängigkeiten.

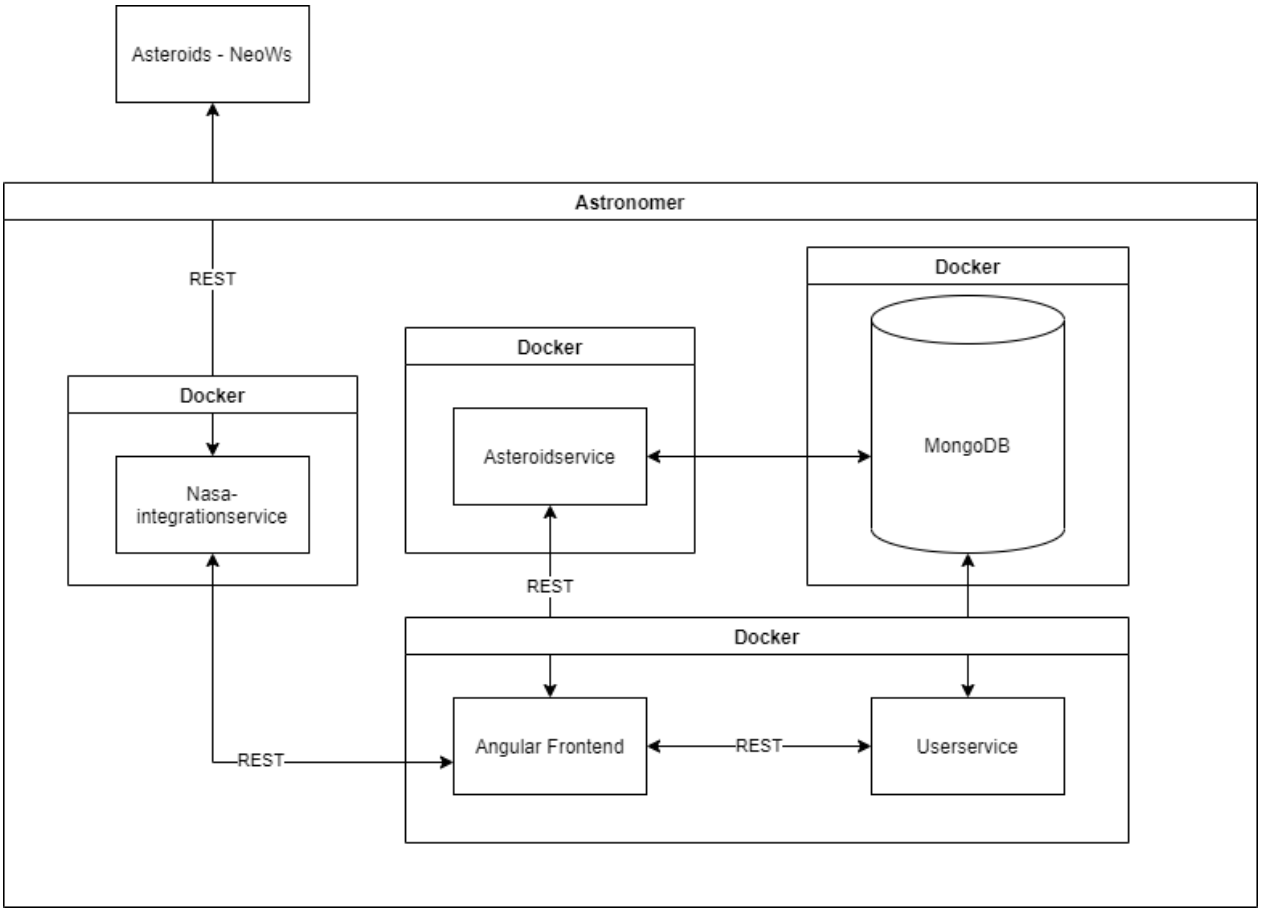


Das Frontend und die einzelnen Backend-Services sind voneinander getrennt. Die Userdaten und die Daten von gespeicherten Asteroiden werden von verschiedenen Services verwaltet und in verschiedenen Collections in der DB gespeichert, da dies auch von zwei verschiedenen Teammitgliedern implementiert wurde und so jeder separat arbeiten konnte.

Übersichtsdiagramm mit Container

Die folgende Abbildung zeigt die Aufteilung der Services auf die Docker-Container.

(Bild-AE)



Dieses Bild stammt aus dem ersten Milestone.

Der Userservice und das Frontend sind in einem Docker-Container, da sonst Probleme mit dem Speichern von den Profilbildern auftraten. Der Userservice muss diese in dem assets-folder von dem Frontend speichern, da dieses sonst nicht darauf zugreifen kann. Ein kopieren des Bildes über die Container hinaus war jedoch nicht möglich und es konnte auch sonst keine andere Lösung gefunden werden.

Alle anderen Services haben ihren eigenen Containern, so wie es gefordert war.

Enthaltene Bausteine

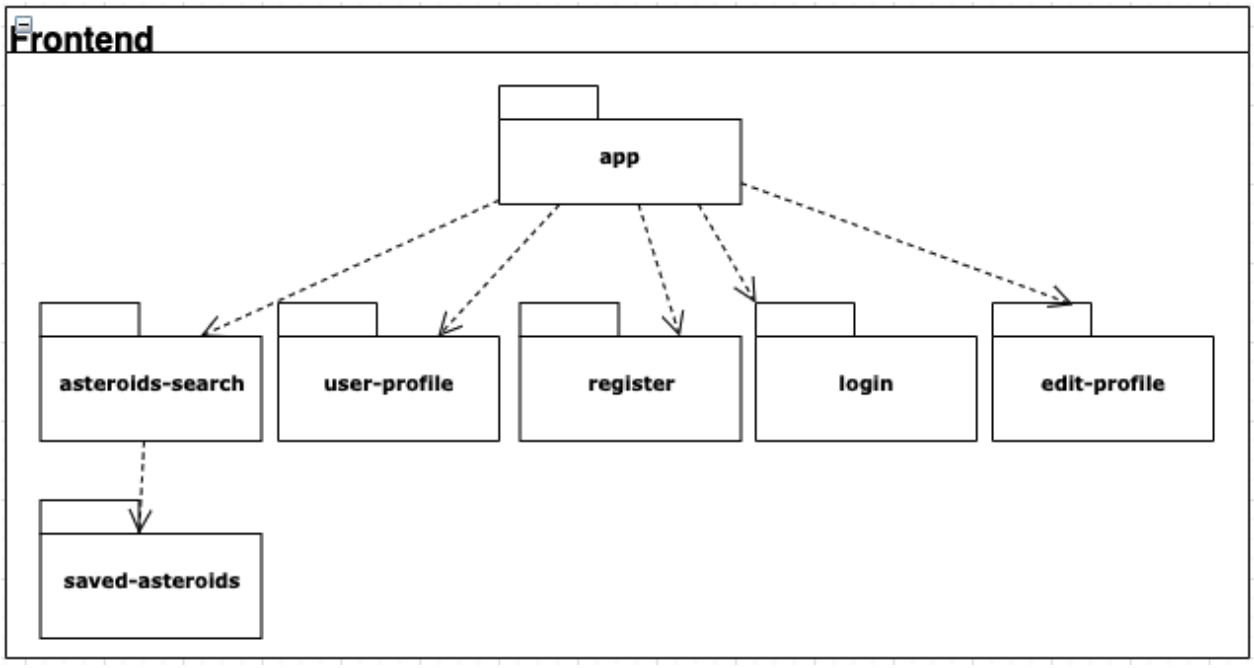
Name	Verantwortung
Frontend	Bildet die Benutzeroberfläche des Systems
NASA Integration Service	Ist die Schnittstelle zu den API's von NASA
Userservice	Ist eine Schnittstelle zur Datenbank und verwaltet Userbezogene Daten
Asteroid Service	Ist eine Schnittstelle zur Datenbank und verwaltet Informationen zu gespeicherten Asteroiden

5.2. Ebene 2

5.2.1. Whitebox Frontend

NW

Folgende Abbildung zeigt das Innenleben vom Frontend:



Das Frontend ist durch die Anwendung von Angular in Komponenten (Components) geteilt. D.h. jedes Feld im Bild ist eine Komponente.

Component	Kurzbeschreibung
app	Grundkomponente, diese wird dem User durchgehend angezeigt (Oberste Tool-Bar)
asteroids-search	"Startseite" der Astronomer Anwendung. Hier werden alle aktuellen Asteroiden angezeigt und (falls der Benutzer angemeldet ist) die gespeicherten Asteroiden
saved-asteroids	Fungiert als sogenannte Child-Component und wird von der asteroid-search Component aufgerufen
user-profile	Komponente für das Benutzerprofil
edit-profile	Komponente zum Bearbeiten des Profils
login	Komponente zum Anmelden
register	Komponente zum Registrieren

-NW

5.2.2. Whitebox *Asteroid Service*

FJ

Der Asteroid Service ist eine Node/Express App zur Verwaltung der gespeicherten Asteroiden der User:Innen.

Der Service verfügt über zwei REST - Schnittstellen.

Links zu Swagger - Dokumentation:

Auf BW-Cloud Server: http://193.197.231.179:3001/api-docs/

Wenn Service lokal ausgeführt wird: http://localhost:3001/api-docs/

-FJ

5.2.3. Whitebox NasaIntegrationService

NW

Der NasaIntegrationService ruft über http-Anfragen die NASA Asteroids NeoWs und NASA SBDB API auf.

Swagger-Dokumentation: http://193.197.231.179:3000/api-docs/

-NW

5.2.4. Whitebox Userservice

AE

Der Userservice wird über http-Anfragen aufgerufen um Daten vom User abzuspeichern, abzufragen oder zu entferne.

Swagger-Dokumentation: http://193.197.231.179:3002/api-docs/

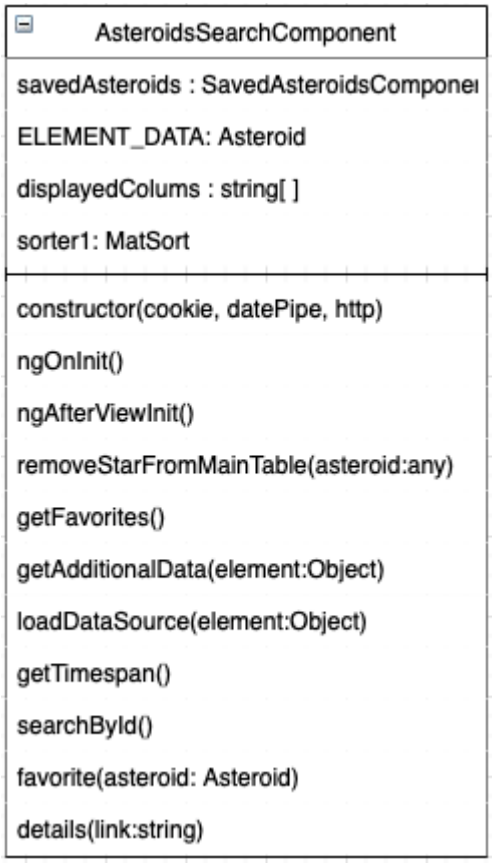
-AE

5.3. Ebene 3

5.3.1. Whitebox AsteroidSearchComponent.ts

NW

Hinweis: Es werden nur die wichtigsten Klassen aus dem Frontend genauer dokumentiert. Des weiteren werden beim folgenden Klassendiagramm nur die wichtigsten Variablen angezeigt, welche eine kleine Erklärung benötigen.



Variablen

Variable	Kurzbeschreibung
savedAsteroids	Mit Hilfe von 'ViewChild' wird die Child-Component 'SavedAsteroidsComponent' in die Variable savedAsteroids geschrieben. Dadurch kann von der Parent-Component (AsteroidsSearchCompnent) eine Funktion aus der Child-Component aufgerufen werden.

ELEMET_DATA	Array um die Tabellen zu füllen
displayedColumns	Tabellenspalten die angezeigt werden sollen
sorter1	wird zum sortieren der mat-table verwendet

Methoden

Methode	**Kurzbeschreibung**
ngOnInit	Sobald die Komponente geladen wird werden die Inhalte dieser Methode ausgeführt (Daten von der API laden und der Tabelle zuweisen)
ngAferViewInit	Sobald die View geladen ist, wird diese Methode ausgeführt (Wird verwendet um das default sort von einem Benutzer zuzuweisen)
removeStarFromMainTable	Sobald in der Child-Component 'SavedAsteroidsComponent' ein Asteroid rausgeschmissen wird, wird diese Funktion aufgerufen, um in der Tabelle den Stern zu demarkieren.
getFavorites	Ruft den UserService auf, um die Favoriten eines Users zu bekommen
getAdditionalData	Ruft über den NasaIntegrationService weitere Informationen (orbit_class) zu einem Asteroiden auf (falls es Informationen dazu gibt)
loadDataSource	Ruft den NasaIntegrationService auf um allgemeine Daten zu einem Asteroiden zu bekommen
getTimespan	ermöglicht die Suche zu einer gewissen Zeitspanne (über den NasaIntegrationService)
searchById	Ermöglicht die Suche nach einem Asteroiden über die Id (über den NasaIntegrationService)
favorite	Wird aufgerufen, sobald ein Asteroid gespeichert oder entspeichert wird
detail	Wird aufgerufen um die SBDB Seite eines Asteroiden zu öffnen

-NW

5.3.2. Whitebox SavedAsteroidsComponent.ts

JS

Um einen übersichtlichen Code zu gewährleisten wurde die Tabelle der Saved-Asteroids in eine eigene Component verschoben.
Sie steht in einem Parent-Child Verhältnis zu der AsteroidSearch-Component.

Die Daten der Tabelle, sind genauso wie in der AsteroidSearchComponent organisiert. Das Sortieren der Tabelle wird ebenso umgesetzt. Es wird die bevorzugte Sortierung-Strategie aus dem Userprofil, bzw. dem entsprechenden Cookie in der 'ngAfterViewinit()' Funktion gelesen und auf die DataSource der Tabelle angewandt.

Die Funktion 'loadSavedAsteroids()' wird verwendet um die Daten der gespeicherten Asteroiden vom AsteroidService abzufragen und in die DataSource der Tabelle einzufügen.
Hierzu wird zuerst eine get-Request an den Userservice geschickt und die IDs aller gespeicherten Asteroiden zu bekommen.
Anschließend werden die IDs in einer weiteren get-Request an den AsteroidService gesendet, welcher die Daten der entsprechenden Asteroiden zurückliefert.
Schließlich wird die DataSource der Tabelle aktualisiert damit die geladenen Asteroiden-Daten in der Tabelle angezeigt werden.

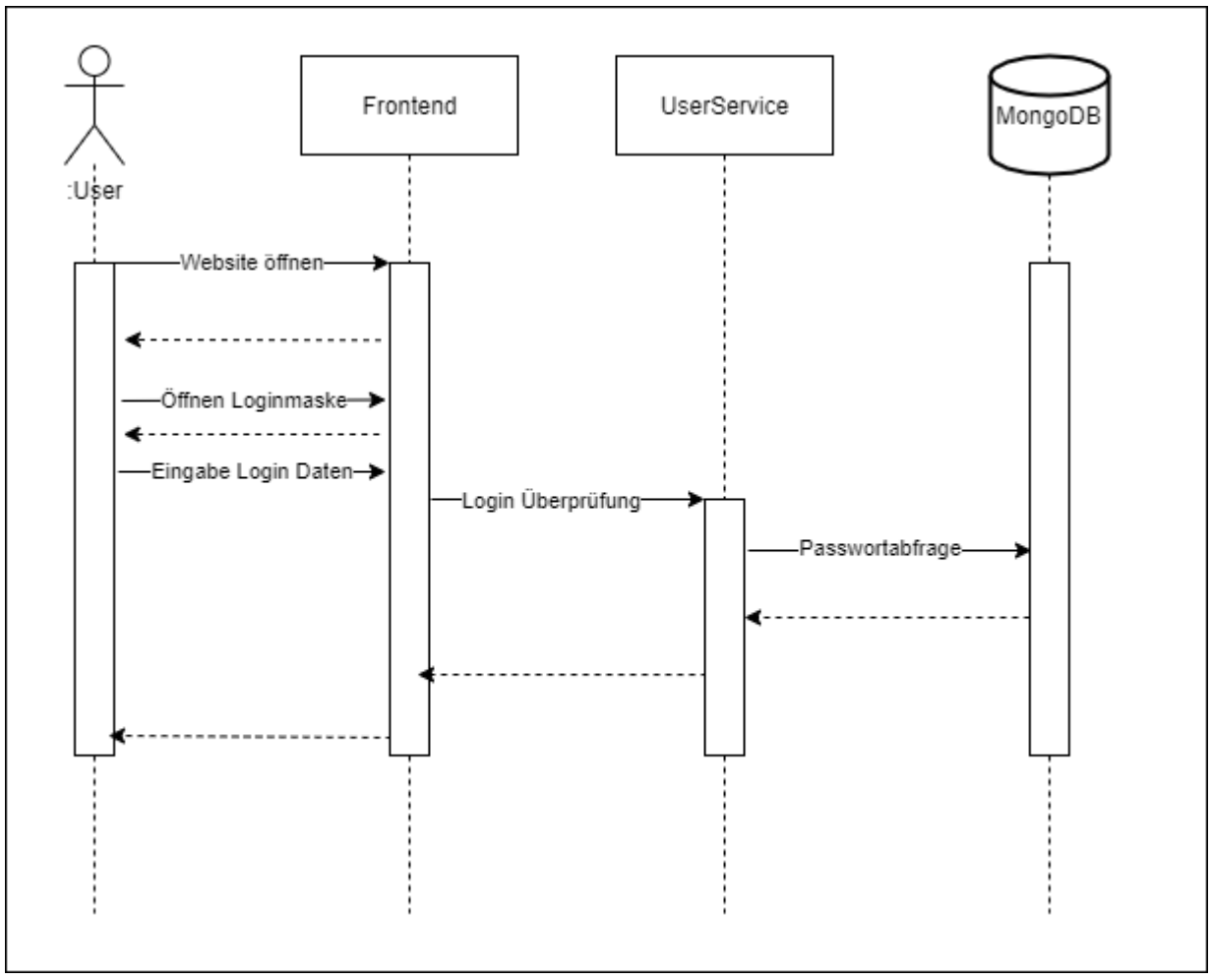
-JS

6. Laufzeitsicht

AE

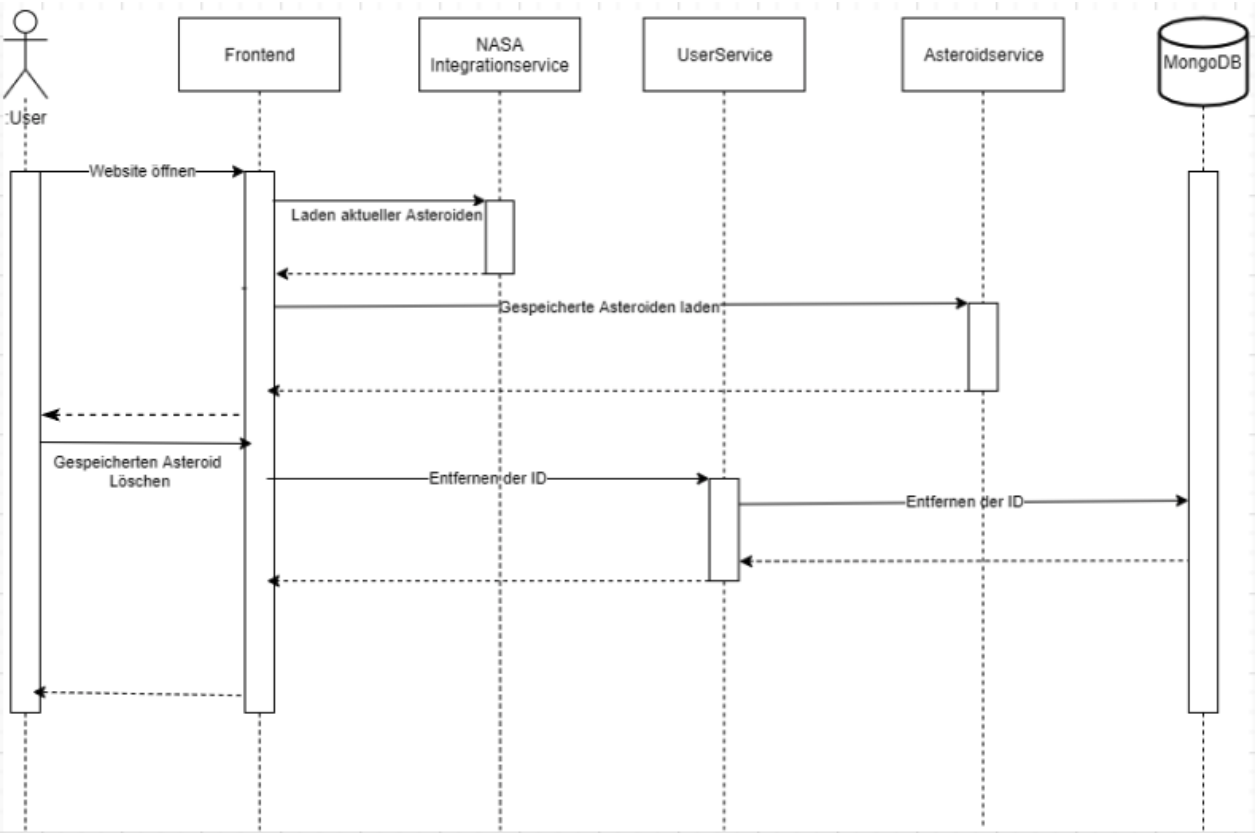
Login Scenario

In diesem Scenario öffnet ein User unsere Website und meldet sich dort mit seinem bestehenden Account an.



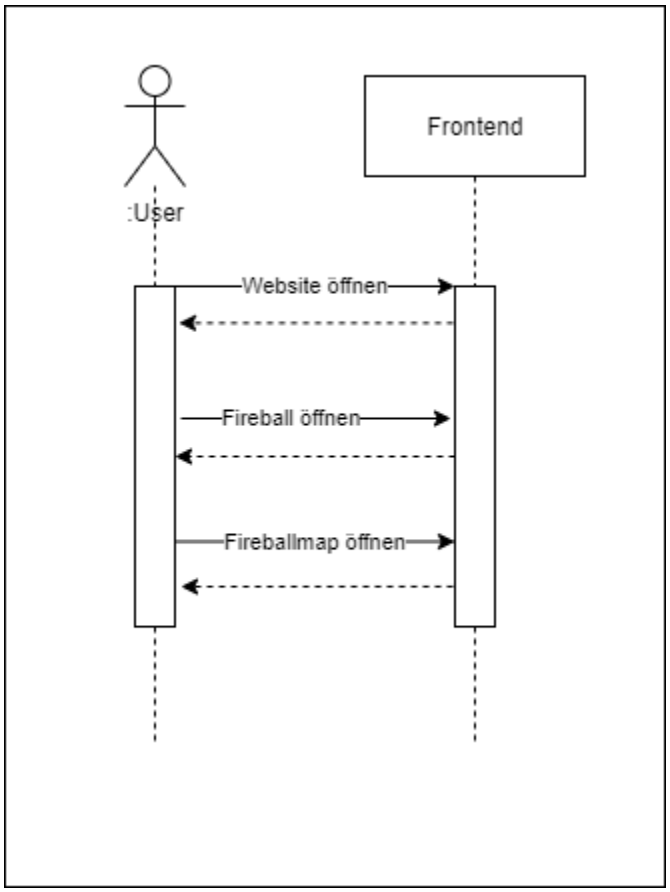
Gespeicherte ID entfernen Scenario

In diesem Scenario löscht ein User der bereits angemeldet ist eine seiner gespeicherten Asteroiden. Dies wird gemacht indem die ID des Asteroidens aus der Datenbank gelöscht wird.



Fireballmap öffnen Scenario

In diesem Scenario öffnet ein User unsere Website und navigiert zur Fireball Seite auf dieser öffnet er dann die Map. Die Fireballmap wurde durch ein Python Script welches sich im "Fireball" Ordner befindet.

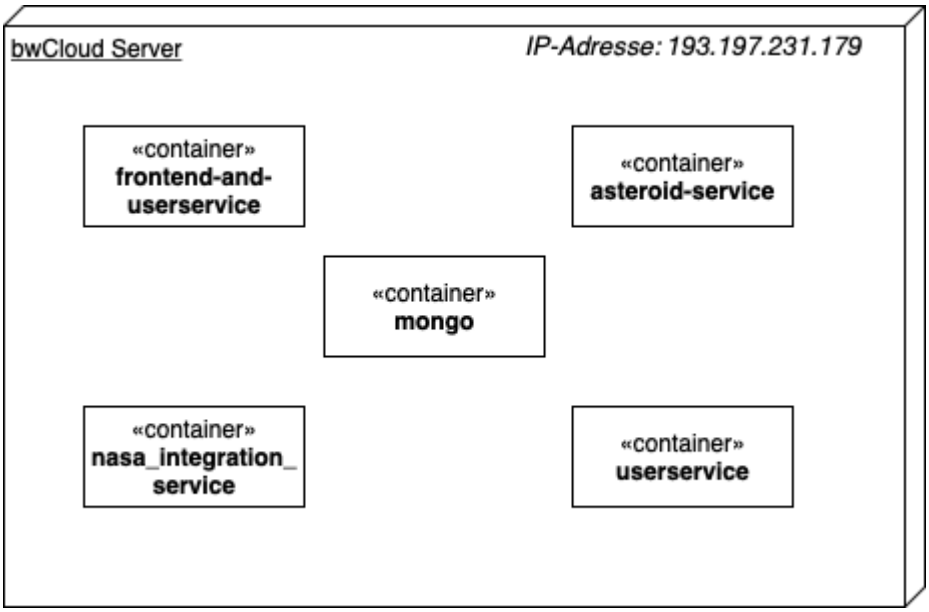


-AE

7. Verteilungssicht

NW

Die folgende Abbildung zeigt die Container im verwendeten Server:



-NW

Die Docker-Container werden alle mit hilfe von Docker-Compose gestartet. Hierfür befindet sich auf oberster Ebene die docker-compose.yml Datei. (FJ)

8. Entwurfsentscheidungen

Technologie	Entscheidungsgrundlage
MongoDB	MongoDB war manchen Team-Mitgliedern schon bekannt. Es kam keine andere Datenbank in die nähere Auswahl.
Angular	Bietet zusammen mit Angular Material und einer großen Community sowie ausführlicher Dokumentation eine gute Grundlage, um schnell teilerfolge in der Anwendungsentwicklung sehen zu können. Als Alternative bot sich die Softwarebibliothek react an, was lediglich aus persönlichen Präferenzen ausgeschieden ist.
Node.js und Express.js	Gut kombinierbar als MEAN-Stack.
NASA SBDB API	Wurde hinzugenommen, um eine NoSQL Datenbank zu rechtfertigen. (Sodass Asteroiden nicht immer das gleiche Schema haben, sondern manche das Feld ‘orbit_class’)
bwCloud SCOPE	Die bwCloud SCOPE (Science, Operations and Education) wurde verwendet, da diese für Studierende aus Baden-Württemberg kostenlos ist und ausreichend ressourcen für eine Applikation dieser größe bereitstellt.

9. Risiken und technische Schulden

Bekannte Bugs:

NW

1. Die Search-Tabelle lässt sich nicht nach den Favoriten Sortieren. Schuld daran ist das verwendete *ngIf in der HTML Datei von search-asteroids in Zeile 87.
   Damit wird eingestellt ob der Stern in der Spalte ausgefüllt ist oder nicht. Leider konnte keine alternative gefunden werden, diese Funktionalität ohne *ngIf durchzuführen.
   Der Bug, dass das Sortieren einer Spalte nicht mehr Funktioniert, sobald *ngIf in der Spalte verwendet wird, ist wohl ein bekannt Bug von Angular Material und es konnte im allgemeinen (also auch in der Community) noch keine Lösung gefunden werden.

2. Die Input-Felder zum Suchen der Asteroiden werden manchmal falsch dargestellt. Die Ursache dafür konnte nicht gefunden werden und somit auch keine Lösung

3. Es kann keine Zeitspanne ausgewählt werden, welche über eine Monatsgrenze geht (z.B.: 30-05-2021 bis 03-06-2021)

4. Wenn beim Login “Enter” gedrückt wird, wird das passwort in Klartext angezeigt.

5. Die Diameter-Spalte sortiert sich nicht richtig. Es blieb keine Zeit den Fehler dahinter zu suchen.

6. Beim default-sortieren der Tabelle der gespeicherten Asteroiden wird anfangs nicht angezeigt, welche Spalte sortiert ist. (Es wird nur nicht angezeigt in der Spalte oben, sortiert ist es jedoch). Dafür konnte aus zeitlichen Gründen auch keine Lösung mehr gefunden werden.

-NW

PS

7. In der mobilen kann es manchmal vorkommen, dass die Ansichtstabelle sich nicht an der größe des Smartphone Bildschirms anpasst. Dies liegt auch an der Hohen genauigkeit, mit der die Daten zurückgeliefert werden. Ein möglicher workaround wäre, den Smartphone Bildschirm quer zu nehmen oder den Browser auf Desktop Ansicht umzustellen (getestet IPhone 12 Pro).

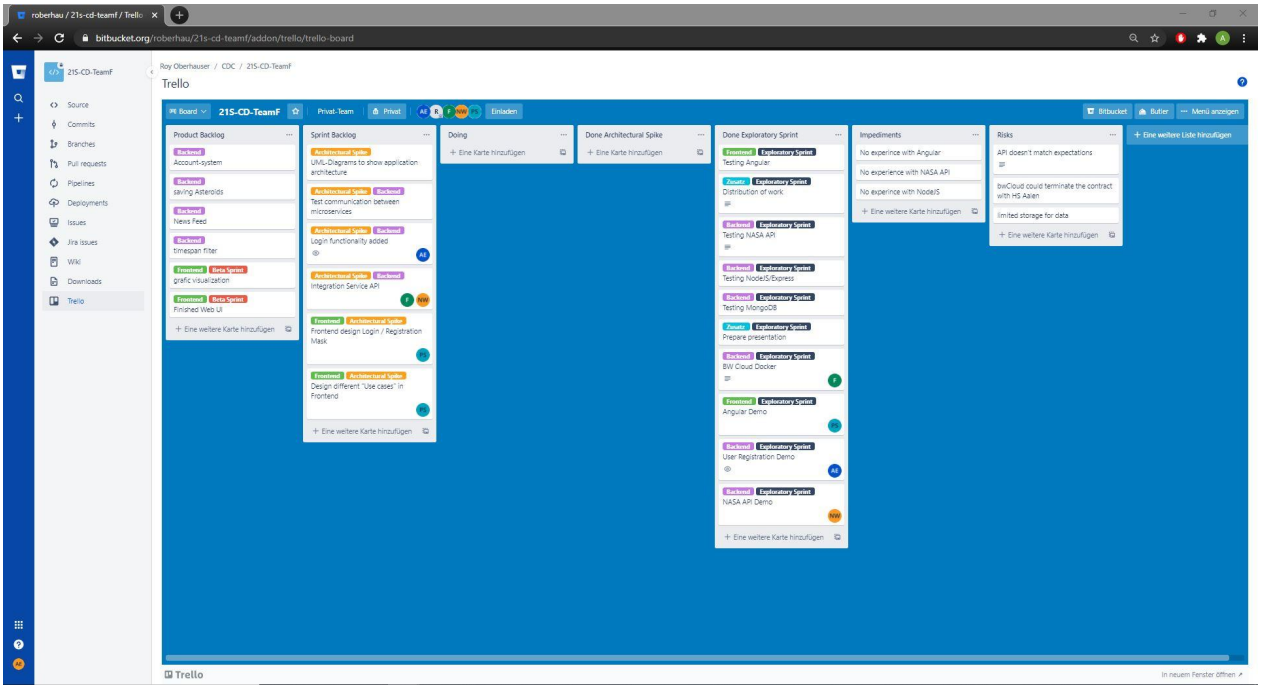
-PS

PS

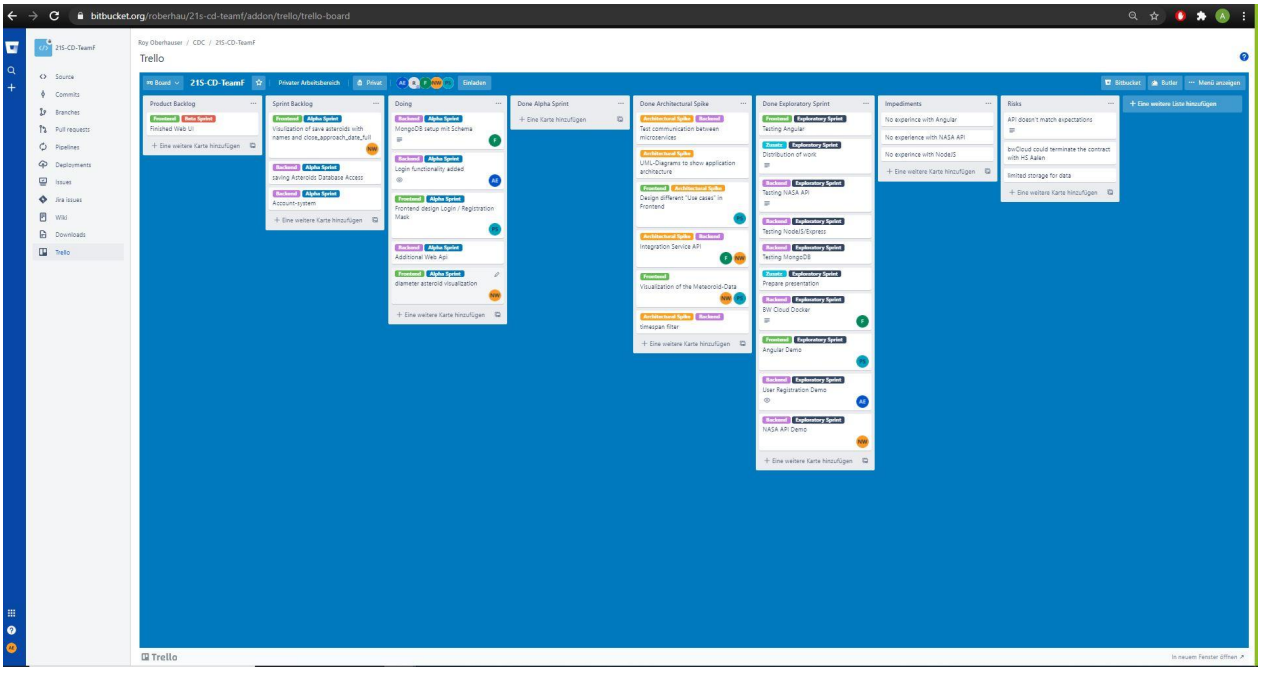
1. Im Cypress Testscript kann das markieren eines Asteroids als Favorit nicht validiert werden. Aus unbekannten Gründen wird der testuser, nachdem der “Stern” eines Asteroiden gedrückt wurde, abgemeldet. im Snapshot selbst ist davon jedoch nichts zu sehen. Wird der Testuser jedoch manuell eingeloggt, so erscheint unter der Favoritenansicht gewünschter Favorit.

10. Anhang

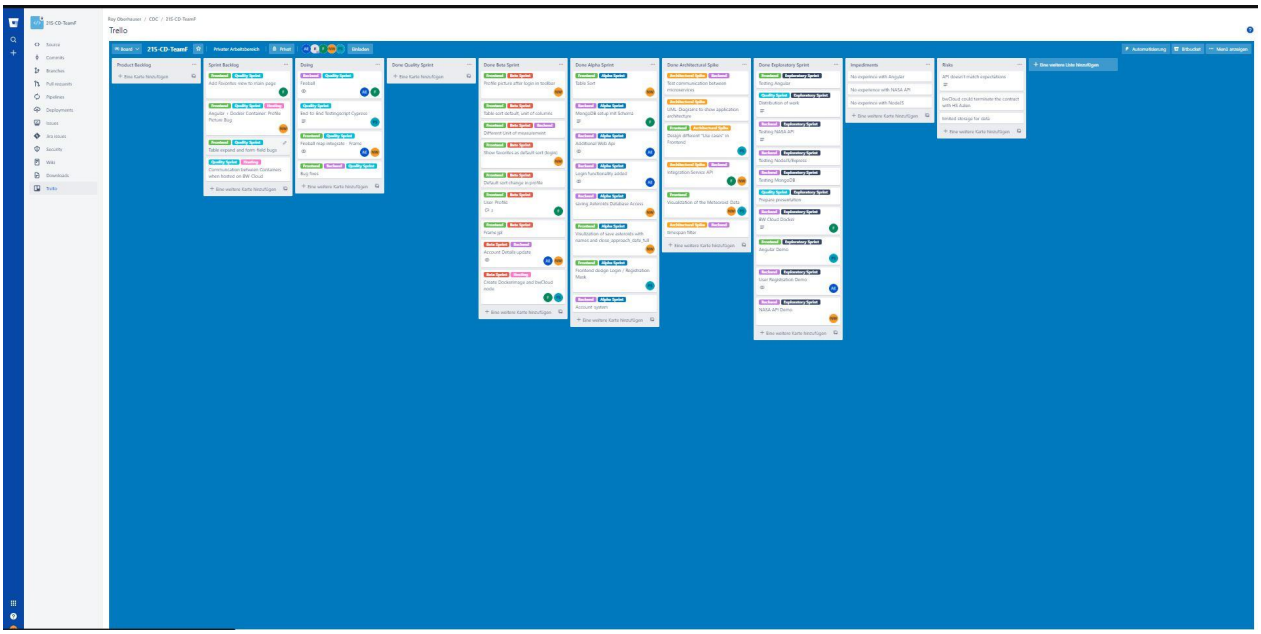
Exploratory Sprint



Architectural Spike



Alpha Sprint



Beta Sprint