HTTP Statuscodes:

An dieser Stelle standen wir vor der Entscheidung, ob wir die für uns in Frage kommenden Statuscodes einfach auf paar wenige, allumfassende reduzieren wollten, oder eben eine größere, umfassendere Menge auflisten wollten.
Wir entschieden uns für die letztere Wahl, aus einer Reihe aus Gründen:
Zum einen, weil wir glauben dass das Nutzen von spezifischeren Fehlercodes grundsätzlich eine bessere Art zu arbeiten darstellt, da sie Nutzern genauere, und damit potentiell hilfreichere Informationen zur verfügung stellt.
Zum anderen, weil es im Rahmen dieses Projektes darum geht, die Arbeits und Funktionsweise von verteilten Systemen im Netz kennenzulernen, und es kaum schaden kann, sich dafür weiter mit den verschiedenen HTML Statuscodes auseinanderzusetzen.

Deshalb haben wir eine vielleicht etwas zu ausufernde Liste an Statuscodes zusammengestellt, die wir aber trotzdem grundsätzlich für sinnvoll halten.

HTTP Status code	Bezeichnung	Beschreibung	Anwendbare Methoden	Kommentar
200 OK	OK	Die Anfrage war erfolgreich. Die Antwort enthält je nach Anfragemethode die angeforderten Daten.	GET, HEAD, PUT, POST, DELETE	Alles hat funktioniert! Es gibt nicht unbedingt viel was man dazu sagen muss. Außer, dass es schon schön ist, wenn mal etwas einfach funktioniert. Als Oberhaupt der ganzen 200er Methoden Familie, wäre dieser Befehl für uns im Zweifel die Statusmeldung, auf die wir zurückfallen würden sollten wir zwar wissen das ein Vorgang wie vorgesehen funktioniert hat, wir aber keine genaue, spezifischere Statusmeldung geben können.

201 Created	Erzeugt	Die Anfrage hat dazu geführt, dass erfolgreich eine neue Ressource erzeugt wurde. Deren URI kann im Antwortheader Location mitGETeilt werden.	PUT, POST	Vorteilhaft, um eben einfach genaueres, besseres Feedback zu geben, was gerade passiert ist. In unserem Falle wollen wir den frei definierbaren Befehl POST alleinig zum Erzeugen neuer Ressourcen und Repräsentationen nutzen. Allerdings kann man die Fähigkeit von PUT, dies auch zu tun, nicht vergessen, will man sich an die HTTP Standards halten.
303 See Other	Siehe anderswo	Die Antwort kann unter einer anderen URI mit der GET-Methode geholt werden. Zweck ist es, nach einer POST-Anfrage nahtlos zu einer ausgewählten Ressource weiterleiten zu können.	POST	
304 Not Modified	Nicht verändert	Die Ressource hat sich seit der letzten Anfrage nicht geändert, die Antwort enthält keine Daten. Dies ist z.B. der Fall, wenn der Client durch einen If-Modified-Since-Header erklärt, dass er die Ressource in einem älteren Stand bereits besitzt.	GET, HEAD	Manche Anfragen lassen sich, etwa um die Bandbreite und den Server zu schonen, aus dem Cache des lokalen Nutzers laden. Dies macht natürlich nur dann Sinn, wenn es keine wesentlichen Veränderungen seit der letzten Abfrage gegeben hat. Fraglich wäre, ob Header wirklich auch hierzu gehört, denn die Informationen ob sich die Datei seit der letzten Anfrage verändert hat, würden ja oft wohl hier zu finden sein. Allerdings ist es gut möglich, dass diese Info auch auf anderem Wege überprüft werden kann.

400 Bad Request	Ungültige Anfrage	Der Client hat eine fehlerhafte Anfrage geschickt, die der Server aufgrund fehlerhafter Syntax nicht bearbeiten kann.	GET, HEAD , PUT, POST, DELETE	Als Oberhaupt der ganzen 400er Methoden Familie, wäre dieser Befehl für uns im Zweifel die Statusmeldung, auf die wir zurückfallen würden sollten wir zwar wissen das es einen Fehler beim Client gegeben, aber keine genaue, spezifischere Statusmeldung geben können
401 Unauthorized	Unautorisiert	Die Anfrage kann ohne Autorisierung nicht verarbeitet werden. Zu diesem Zweck muss der Server einen WWW_Authenticate-Antworthe ader mitschicken, der erläutert, wie die Authentifizierung vorzunehmen ist. HTTP definiert die zwei Methoden „Basic“ und „Digest“ vor.	GET, HEAD, PUT, POST, DELETE	Im momentanen Zustand der Applikation noch nicht genutzt, und wahrscheinlich auch weit über den Rahmen dieses Projektes hinausgehend, ist die Möglichkeit, den Zugang zu bestimmten Informationen zu beschränken extrem wichtig. Unsere Applikation würde mit extrem persönlichen, und damit sensiblen Daten arbeiten, was es umso wichtiger macht, nicht einfach alles und jeden auf jede Ressource zugreifen zulassen. Und selbst außerhalb von Ängsten vor Datendiebstahl, sollte es normalen Nutzern nicht möglich sein, ausversehen die Daten anderer Nutzer zu bearbeiten oder zu löschen.

403 Forbidden	Verboten	Der Zugriff ist dauerhaft verboten. Eine Authorisation wird nicht anerkannt, und der Client soll die Anfrage nicht noch einmal stellen.	GET, HEAD, PUT, POST, DELETE	Eigentlich wie oben die 401, nur in diesem Falle noch wesentlicher. Hier hat nicht die Identifikation fehlgeschlagen, nein, es wurde ein Zugriff versucht, den wir grundsätzlich so auf diese Ressource nicht erlauben wollen. Nützlich, um auch intern solche Vorgänge extra und gesondert zu dokumentieren.
404 Not Found	Nicht gefunden	Der Server konnte die vom Client angeforderte Ressource nicht finden. Häufigste Ursache ist ein sogenannter Toter Link. Der 404-Status kann ebenfalls zurückgeliefert werden, wenn kein anderer Statuscode zutreffend ist oder der Client bewusst ohne Angabe eines näheren Grundes abgewiesen werden soll.	GET, HEAD, PUT, POST, DELETE	Der wahrscheinlich meist gesendete Fehlercode darf natürlich nicht fehlen. Im Zweifel hat man sich wahrscheinlich vertippt.
405 Method Not Allowed	Methode nicht erlaubt	Die Anfrage-Methode (wie GET oder POST) ist nicht erlaubt. Diese Antwort kann z.B. zurückkommen, wenn man WebDAV-Methoden verwendet, ohne dass der Server diese beherrscht. In einem Allow-Antwortheader muss der Server erlaubte Methoden auflisten.	GET, HEAD, PUT, POST, DELETE	Könnte helfen, falls wir bestimmte HTTP Methoden in bestimmten Fällen gesondert verbieten möchten.

408 Request Time-out	Anfrage-Zeitüberschreitung	Die Anfrage wurde vom Client nicht in der vom Server vorgegebenen Zeit beendet. Der Fehler kann z.B. beim Hochladen großer Dateien auftreten.	GET, HEAD, PUT, POST, DELETE	Eigentlich immer notwendig. Es kann nur von Vorteil sein, genau zurückzugeben dass der Fehler an einem Timeout gelegen hat. Zwar ließe sich darüber nachdenken, ob dies nicht eher der der REST Architektur hinterliegenden Logik widerspricht, besonders der Zustandslosigkeit, aber bei bestimmten Fällen, wie etwa größeren Upload oder anderen Szenarios könnte dies dennoch auftreten.
409 Conflict	Konflikt	Die Anfrage kann nicht bearbeitet werden, weil sich dadurch ein Konfliktfall bei der angefragten Ressource ergeben würde. Soll z.B. eine Ressource per PUT-Methode verändert werden, die aber auf dem Server bereits eine neuere Version aufweist, würde die Bearbeitung einen Konflikt erzeugen.	PUT, POST	Auf jeden Fall vonnöten. Mehrere Clients, die gleichzeitig per Put die Repräsentation einer Ressource abändern, oder auch einfach der Versuch, eine Präsentation zu erstellen, die es bereits so unter dieser URI gibt, können hier Fehler generieren.
410 Gone	Verschwunden	Die Ressource existiert nicht mehr, der Server kennt aber keine Weiterleitungsadresse. Dieser Zustand ist als dauerhaft anzunehmen.	GET, HEAD, PUT, DELETE	Nur der Befehl POST, von uns genutzt zur Erstellung neuer Repräsentationen, wird hiervon nicht betroffen, da er nicht auf bereits existierende Repräsentationen zugreift.
412 Precondition Failed	Vorbedingung missglückt	Eine vom Client vorgegebene Bedingung ist nicht erfüllbar.	GET, HEAD, PUT, POST, DELETE	Eher weit gefasst, dient dieser Fehlercode in unseren Augen gut, um einen genaueren Status zu senden.

413 Request Entity Too Large	Anfrage-Entität zu groß	Der Inhalt der Anfrage (z.B. eine hochgeladene Datei) ist zu groß, die Anfrage wird abgelehnt.	PUT	Sollte in unserer Anwendung jetzt nicht unbedingt vorkommen, schließlich sind JSON Dateien nicht unbedingt für ihre Größe bekannt, dennoch sollte das nützlich sein, falls jemand den Roman Krieg und Frieden bei uns ablegen möchte.
414 Request-URI Too Large	Anfrage-URI zu lang	Die Anfrage-URI ist zu lang. Der HTTP-Standard definiert keine feste Obergrenze für URI-Längen, d.h., das Erscheinen dieses Fehlers hängt vom Server-Programm und seiner internen Einstellung ab.	GET, HEAD, PUT, POST, DELETE	
416 Requested range not satisfiable	Anfrage-Bereich nicht erfüllbar	Der angefragte Teilbereich der Ressource existiert nicht oder ist ungültig. Dies kann z.B. im Zusammenhang mit partiellen Downloads auftreten.	GET, HEAD, PUT, DELETE	
423 Locked	gesperrt	Die angeforderte Ressource ist zurzeit gesperrt (RFC 4918)	GET, HEAD, PUT, POST, DELETE	Nützlich, um bestimmte Ressourcen zu schützen oder URIs freizuhalten.
429 Too Many Requests	zu viele Anfragen	Der Client hat zu viele Anfragen in einem bestimmten Zeitraum gesendet. (RFC 6585)	GET, HEAD, PUT, POST, DELETE	Zwar gehen wir nicht davon aus, dass unsere Applikation der nächste große Hit wird, aber man weiß ja nie. Ebenso nützlich um die Auslastung des Servers zu schonen, wie auch mit DDOS attacken umzugehen.

500 Internal Server Error	Interner Server-Fehler	Der Server entdeckt einen internen Fehler und kann deshalb die Anfrage nicht bearbeiten. Fehler in Programmen auf dem Server, z.B. PHP-Skripten, können diese Antwort provozieren. (Wird von uns genutzt, wenn kein anderer, genauerer Statuscodes übernommen werden kann.)		Als Oberhaupt der ganzen 500er Methoden Familie, wäre dieser Befehl für uns im Zweifel die Statusmeldung, auf die wir zurückfallen würden sollten wir zwar wissen das es einen Fehler beim Server, oder anderswo in unserer internen Struktur gegeben hat, wir aber keine genaue, spezifischere Statusmeldung geben können.
501 Not Implemented	Nicht implementiert	Der Server verfügt nicht über die nötige Funktionalität, um die Anfrage zu verarbeiten. Diese Antwort ist angemessen, wenn die in der Anfrage verwendete HTTP-Methode nicht erkannt oder unterstützt wird.		Wird genutzt, sollten HTTP Methoden genutzt werden, die wir nicht kennen oder nutzen wollen. In diesem Fall kommt dann natürlich keine der fünf von uns genutzten Methoden in Frage.
503 Service Unavailable	Dienst nicht verfügbar	Der Dienst ist derzeit nicht verfügbar. Dies kann aufgrund hohen Datenaufkommens oder wegen Wartungsarbeiten passieren.	GET, HEAD, PUT, POST, DELETE	Leider ist dieser Anschluss gerade nicht vorhanden. Vielleicht ist der Server gerade down wegen Wartungsarbeiten, vielleicht ist er auch einfach Kaputt. Es ist auf jeden Fall als Client gut zu wissen, dass es an den anderen liegt.
505 HTTP Version Not Supported	HTTP-Version nicht unterstützt	Die angeforderte HTTP-Version wird nicht unterstützt.	GET, HEAD, PUT, POST, DELETE	
507 Insufficient Storage	Speicher des Servers reicht nicht	Die Anfrage konnte nicht bearbeitet werden, weil der Speicherplatz des Servers dazu zurzeit nicht mehr ausreicht.	PUT, POST	

	aus			
508 Loop Detected	Endlosschleife	Die Operation wurde nicht ausgeführt, weil die Ausführung in eine Endlosschleife gelaufen wäre. Definiert in der Binding-Erweiterung für WebDAV gemäß RFC 5842, weil durch Bindings zyklische Pfade zu WebDAV-Ressourcen entstehen können.	GET, HEAD, PUT, POST, DELETE	Um ehrlich zu sein, kann ich mir nicht mal vorstellen, wie man das überhaupt bei uns anstellen würde, aber interessant, dass es das gibt.
511 Network Authentication Required	Identifizierung benötigt	Der Client muss sich zuerst authentifizieren um Zugang zum Netzwerk zu erhalten (RFC 6585)	GET, HEAD, PUT, POST, DELETE	Geht in die Richtung des 401 Codes, aber hier eben auf Serverseitiger Ebene. Wenn ich das richtig verstehe, will der Server gar nicht erst mit einem Reden, bevor man sich nicht Autorisiert hat.