Carchain soll in der Lage sein Bilder für Autos speichern und abrufen zu können. Da das Speichern und Abfragen von Bildern an der Blockchain sowohl das Aufkommen von Transaktionen so wie den Speicherplatz sehr stark belasten würde, wurde diese Funktion ausgelagert. Es wurde daher ein REST-Backend zur Speicherung und Abfrage von Bildern implementiert. Das Backend bietet fünf Schnittstellen beziehungsweise Endpunkte an, die für die Speicherung und Abfrage der Bilder genutzt werden können. Das Backend wurde in NodeJs implementiert und hat eine Anbindung an eine Postgres Datenbank, in der die Bilder verwaltet werden. Die Bilder werden beim Empfang in Base64 umkodiert und in der Datenbank als Text gespeichert. Wenn ein Bild hochgeladen wird, wird dies für ein spezielles Auto hochgeladen. Daher muss im Voraus das Auto in der Datenbank angelegt worden sein, da ansonsten das Bild mit keinem Auto verknüpft werden kann. Es werden in dem Backend zwei (HTTP-) PUT, zwei GET und eine DELETE Funktion bereitgestellt.

Diese Funktionen sind hier nun kurz mit Übergabeparameter und Return Values angegeben:

**put /addCar**

Erwartet application/json mit carid

{

"carid": "string"

}

Return

{

"success": 1,

"error": false,

"msg": "Message received-insert succeed"

}

**put /addImage**

Erwartet application/json mit carid (string) und einem Bild (File)

{

"carid": "string"

"carimage": "File"

}

Return

{

"success": 0,

"error": true,

"msg": "Message received-insert succeed",

"imageid": 2

}

**get /getImage/:imageId**

Erwartet imageId in URL 🡪 193.196.54.51:3005/getImage/X (Bild Id)

Return

{

"success": 1,

"error": false,

"msg": "Message received",

"image": "base64EncodedImage"

}

**get /getImages/:carid**

Erwartet carid in URL 🡪 193.196.54.51:3005/getImages/String (Car Id)

Return

{

"success": 1,

"error": false,

"msg": "Message received with id:TÜ-LF-308",

"imagejson": "[

{id:2,image:base64EncodedImage},

{id3,image:base64EncodedImage},… ]

}

**delete /deleteCar/:carid**

Erwartet carid in URL 🡪 193.196.54.51:3005/deleteCar/String (Car Id)

Return

{

"success": 1,

"error": false,

"msg": "Car and all images deleted"

}

Das Backend wird wie bereits erwähnt genutzt, um die Bilder für die Autos zu speichern. Es wird hier nun der Vorgang schematisch dargestellt.

Ein Bild, das Text, Karte enthält.

Automatisch generierte Beschreibung

Beim Hinzufügen eines Bildes wird dieses, mit der ID, welche ein Auto repräsentiert, an den Server geschickt. Der Server prüft mit einer Datenbank Abfrage, ob die ID in der Datenbank enthalten ist. Ist diese ID enthalten, wird das Bild, nachdem es in Base64 umkodiert wurde, in der Datenbank gespeichert und mit der ID verknüpft. Die ID des Bildes wird anschließend in der http Repsonse mitgeschickt. Man kann mehrere Bilder einem Auto hinzufügen. Diese Bilder können alle gesammelt abgefragt werden. Hierzu muss man nur den entsprechenden Befehl mit der Auto Id aufrufen. Natürlich muss man zu Beginn die Auto Id anlegen. Es ist auch möglich, die Bilder einzeln anhand ihrer ID abzufragen. Das Backend wurde mithilfe von nodejs entwickelt. Das nodejs Programm wird auf dem Server aufgesetzt und die genutzten Packages installiert. Dies wären:

* express: Starten der Server App
* bodyParser: Bearbeitung des http Body
* http: Server Funktion
* multer: Für File upload
* fs: Für Filesystem
* crypto-js: Für die Hashfunktion des Tokens

Die Server Applikation wird auf dem Port 3005 gestartet.