

ToTheLoo

Jamal, Jonas, Lena, Niklas und Patrick

Allgemeines zur App

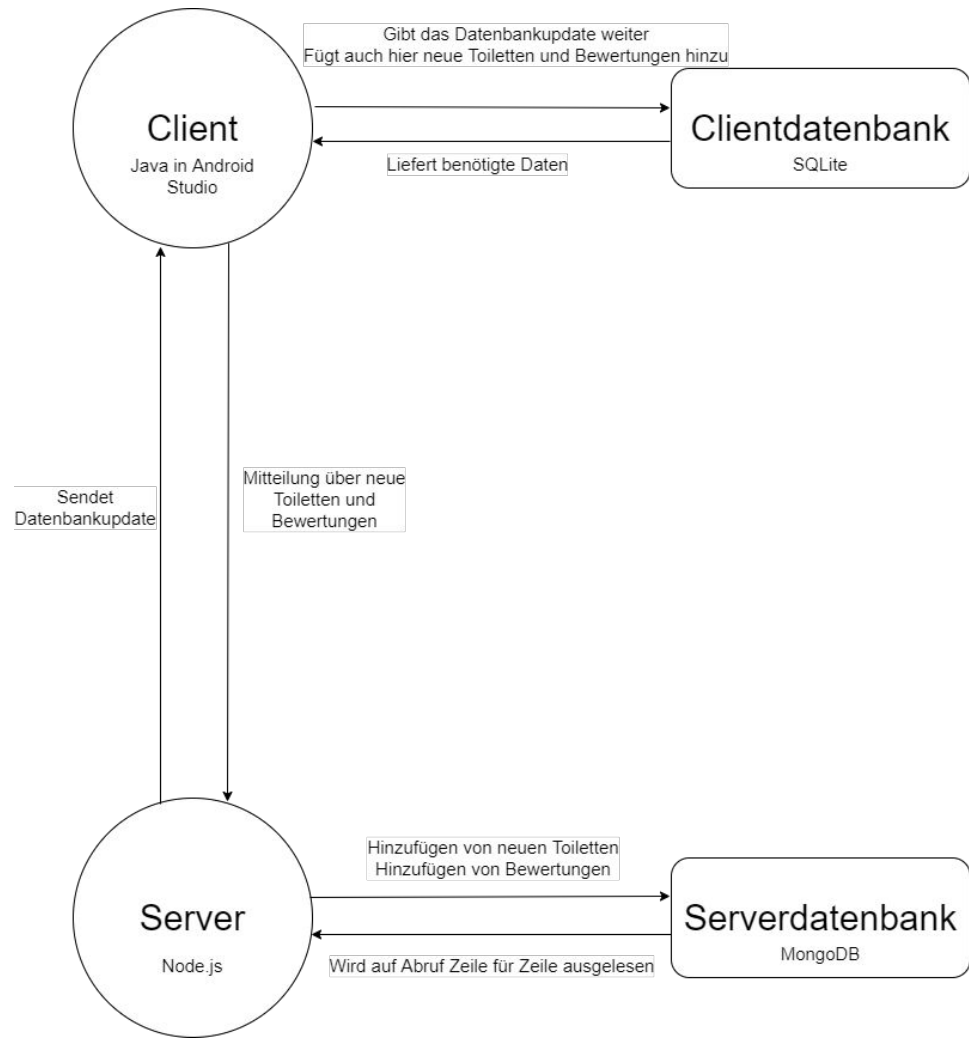
Was soll sie können?

- Anzeige der Toiletten in der Umgebung
- Anzeige von Details einer ausgewählten Toilette
- Routing zu einer ausgewählten Toilette
- Toiletten Reviews erstellen

Nice-to-haves:

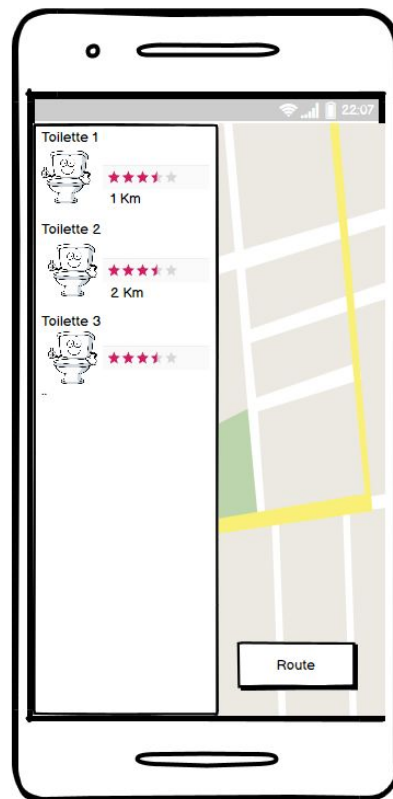
- Filteroptionen für die anzuzeigenden Toiletten
- Hinzufügen neuer Toiletten

Kommunikation



Erste UI-Entwürfe





Kurze Präsentation des aktuellen UI

Klientendatenbank Schema

- Relationales Datenbankmodell

ID, Name, Breitengrad, Längengrad, Tag, Wegbeschreibung, Beschreibung,
Bewertungen(Benutzer, Bewertungstext, Sterne)

Klientendatenbank Implementation

- SQLite
- Vorteil: bereits in Android Studio implementiert
 - ⇒ Erstellung der Datenbank simpel
 - ⇒ Kommunikation mit Datenbank einfach
- Eingabe von Daten als String
- Ausgabe als Cursor Objekt

Aufbau des Backends

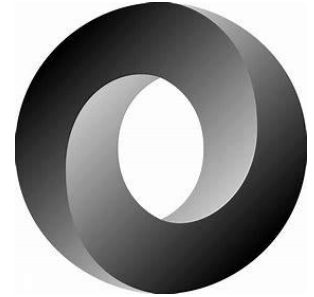
Usecase des Backends: Speicherung von Daten aller Nutzer um eine möglichst große Datenbasis aufzubauen



- Javascriptbasiertes Serverframework
- Benutzung von Express-Modul für REST API Erstellung



- Mongo=humongous
- NoSQL - dokumentenbasiert
- verwaltet JSON-Dokumente
- läuft auf der VM



- Übertragung der Daten über das Http-Protokoll im JSON-Format

Anmerkungen zu Folie 9

- Das Backend unserer Applikation hat grundsätzlich und ausschließlich die Funktion Daten zu empfangen, aufzubereiten und abzuspeichern sowie Daten an das Frontend zu liefern. Aufgrundessen haben wir uns entschieden, dass ein einfacher Nodejs Server, kombiniert mit dem Expressframework ausreicht.
- Die MongoDB als dokumentenbasierte Datenbank haben wir gewählt, da sie in der Softwareentwicklung immer etablierter ist und ein NoSql-Schema für unsere Datenstruktur durchaus passend ist.

Datenmodell - Momentaner Aufbau eines LooObject

```
1 {
2   "name": "WonderToilet",
3   "price": 0.5,
4   "place": {
5     "longitude": "53.540307",
6     "latitude": "10.021677",
7     "tag": "ItsATag",
8     "navigationDescription": "Just go around the corner where
9   },
10  "kind": {
11    "description": "Pissoir",
12    "icon": "ItsANIcon"
13  },
14  "rating": [
15    {
16      "user": "Anon",
17      "ratingText": "It was wonderful",
18      "stars": "4"
19    },
20    {
21      "user": "Anon1",
22      "ratingText": "It smelled a bit!",
23      "stars": "2"
24    },
25    {
26      "user": "Anon2",
27      "ratingText": "Like heaven",
28      "stars": "5"
29    }
30  ]
31 }
```

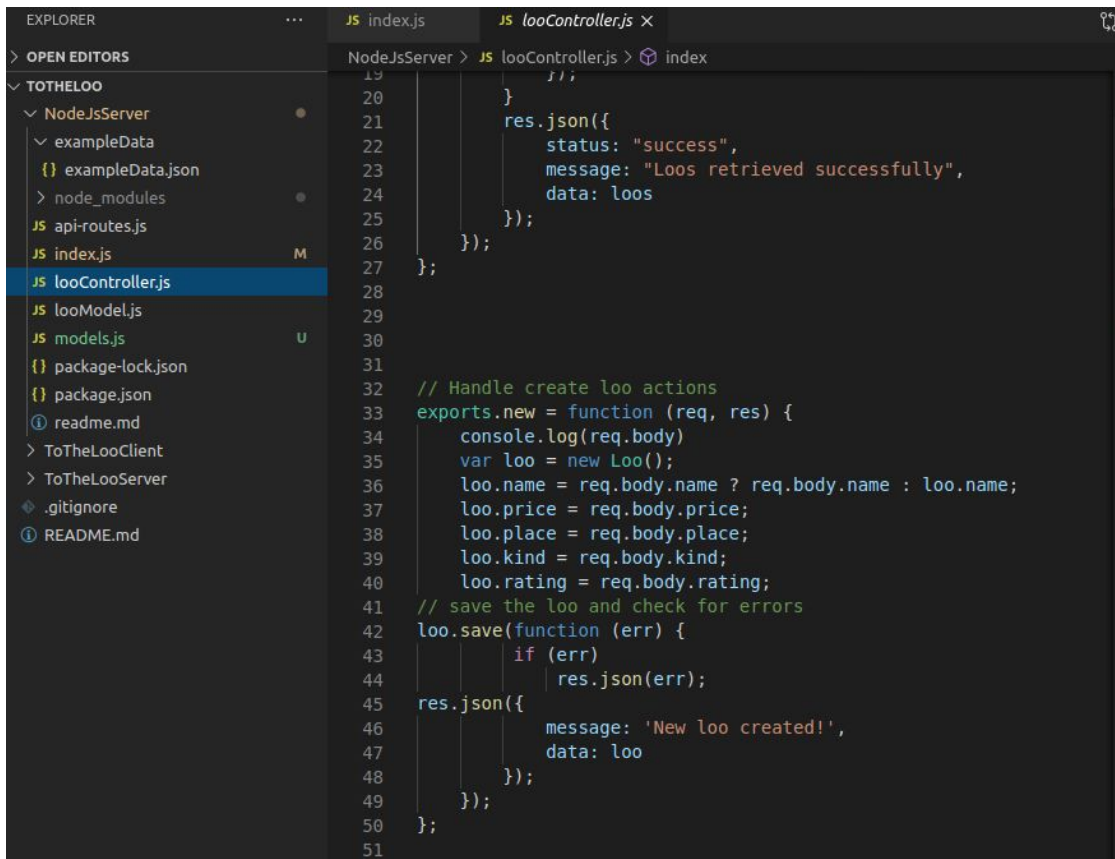
Anmerkungen zu Folie 10

- Eine Entität in unserer Datenbank (eine “*Loo*”) hat die folgenden Attribute: *Name*, *Price*, *Place*, *Kind*, *Rating*. Wobei *Name* und *Rating* optional sind. Aufgrund der Beschaffenheit unserer App kann der *Place*, der hauptsächlich aus Längen- und Breitengrad besteht, nicht leer sein. Auch erwarten wir von einem Benutzer, der eine neue *Loo* anlegt, dass er einen Preis angibt und, ob die Toilette ein Pissoir, WC, etc. ist (wird abgedeckt durch *Kind*)
- *Rating* wird als ein Array in unserer Datenstruktur repräsentiert. Ein *Rating* enthält die Anzahl der abgegebenen Sterne und die dazugehörige Beschreibung, die optional ist. Zusätzlich gibt es noch das Felder *User*, mit dem schon auf die Implementation eines Usermanagements vorgriffen wird. Dies existiert noch nicht in unserer Anwendung, könnte allerdings sinnvoll sein um weitere benutzerspezifische Konfigurationen oder Funktionen anzuzeigen, sowie für die Einführung eines Gamificationkonzeptes.

Weiterführung zu Anmerkungen zu Folie 10 - rein technische Attribute

- Ein weiteres Attribut einer Loo ist die Version. Die Version ist ein rein technisches Feld, welches jedes Mal hochgesetzt wird, wenn ein Update an einem Looobjekt durchgeführt wurde. Falls nun ein anderer Nutzer seine App öffnet, kann über dieses Feld ermittelt werden, welche neuen Loos es gibt. Hat der spezifische Nutzer nun eine Loo die outdated ist, liefert der Server die neueste Version dieser Loo.
- Für die Identifizierung einer Loo benötigt es eine technische Id. Diese wird für die meisten Operationen im Front- sowie im Backend benutzt

Controller - Handling der Requests



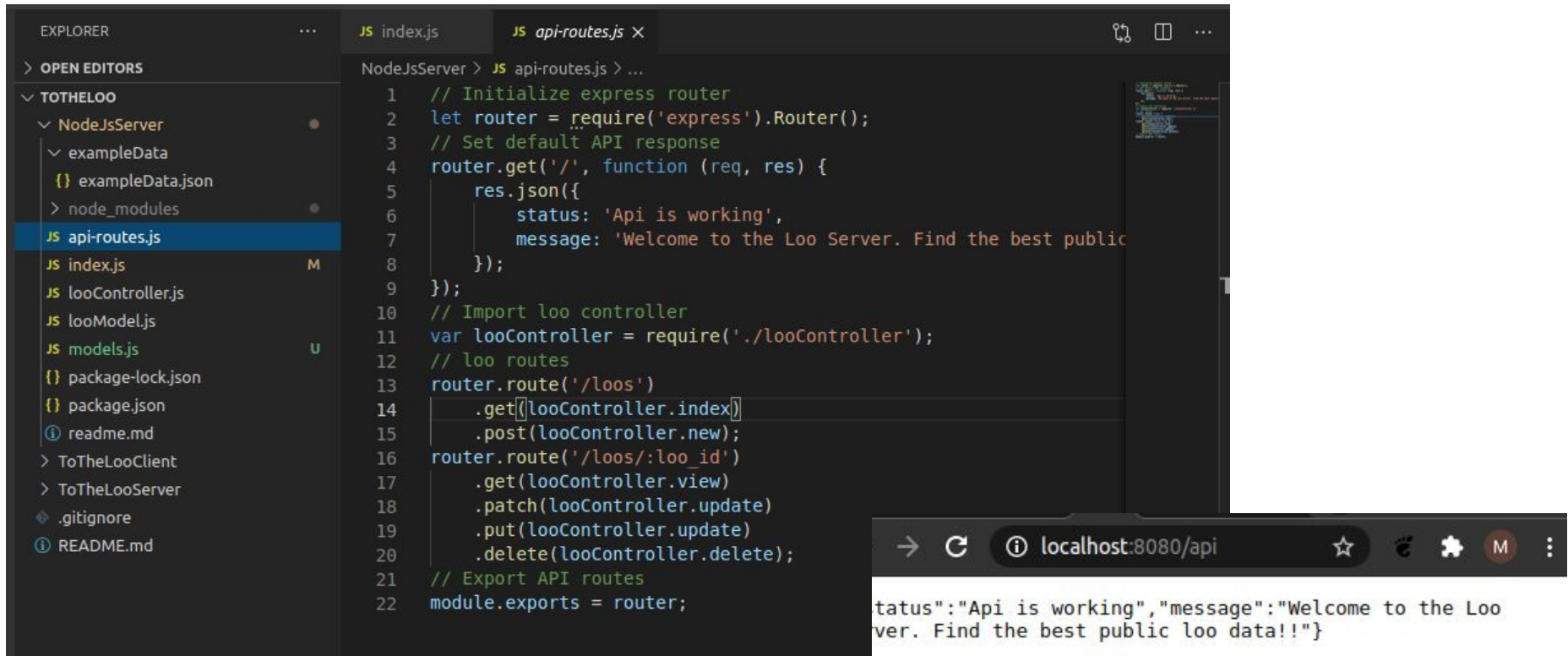
The image shows a screenshot of the Visual Studio Code editor. On the left, the Explorer sidebar displays the project structure for 'TOTHELOO'. The 'NodeJsServer' folder is expanded, showing files like 'exampleData.json', 'node_modules', 'api-routes.js', 'index.js', 'looController.js' (which is selected), 'looModel.js', and 'models.js'. The main editor area shows the code for 'looController.js'. The code defines a new Express route handler for the POST endpoint '/loos'. It logs the request body, creates a new 'Loo' object, sets its properties (name, price, place, kind, rating) from the request body, and then calls 'loos.save()' to persist the data. If there is an error, it sends the error back to the client. Otherwise, it sends a success message and the created 'loos' object.

```
19 // ...
20 }
21 res.json({
22   status: "success",
23   message: "Loos retrieved successfully",
24   data: loos
25 });
26 });
27 };
28
29
30
31
32 // Handle create loos actions
33 exports.new = function (req, res) {
34   console.log(req.body)
35   var loos = new Loo();
36   loos.name = req.body.name ? req.body.name : loos.name;
37   loos.price = req.body.price;
38   loos.place = req.body.place;
39   loos.kind = req.body.kind;
40   loos.rating = req.body.rating;
41   // save the loos and check for errors
42   loos.save(function (err) {
43     if (err)
44       res.json(err);
45     res.json({
46       message: 'New loos created!',
47       data: loos
48     });
49   });
50 };
51
```

Anmerkungen zum Controller

- Im Backend benutzen wir Controller zur Definition und Handling der REST Schnittstellen. Wir definieren die Routen in der index.js bzw. in api-routes.js. Dort wird definiert welche Funktion des Controllers für die jeweilige Route verantwortlich ist und was dann passieren soll. Zudem ist in der api-routes auch definiert welche Art von request an die jeweilige Schnittstelle gestellt werden darf. Wir benutzen POST, GET und PATCH Requests.
- Momentan ist die eigentliche Logik für das Requesthandling in einem Service ausgelagert um den Controller etwas schlanker zu halten.

Bisher eingestellte Routen (Schnittstellen)



The image shows a development environment with VS Code and a web browser. In VS Code, the Explorer sidebar on the left shows the project structure for 'TOTHELOO', with 'api-routes.js' selected. The main editor displays the code in 'api-routes.js', which initializes an Express router and sets up several routes. The browser window at the bottom right shows the URL 'localhost:8080/api' and the JSON response from the API.

VS Code Explorer:

- TOTHELOO
 - NodeJsServer
 - exampleData
 - exampleData.json
 - node_modules
 - api-routes.js**
 - index.js
 - looController.js
 - looModel.js
 - models.js
 - package-lock.json
 - package.json
 - readme.md
 - ToTheLooClient
 - ToTheLooServer
 - .gitignore
 - README.md

VS Code Editor (api-routes.js):

```
1 // Initialize express router
2 let router = require('express').Router();
3 // Set default API response
4 router.get('/', function (req, res) {
5   res.json({
6     status: 'Api is working',
7     message: 'Welcome to the Loo Server. Find the best public
8   });
9 });
10 // Import loo controller
11 var looController = require('./looController');
12 // loo routes
13 router.route('/loos')
14   .get([looController.index])
15   .post(looController.new);
16 router.route('/loos/:loo_id')
17   .get(looController.view)
18   .patch(looController.update)
19   .put(looController.update)
20   .delete(looController.delete);
21 // Export API routes
22 module.exports = router;
```

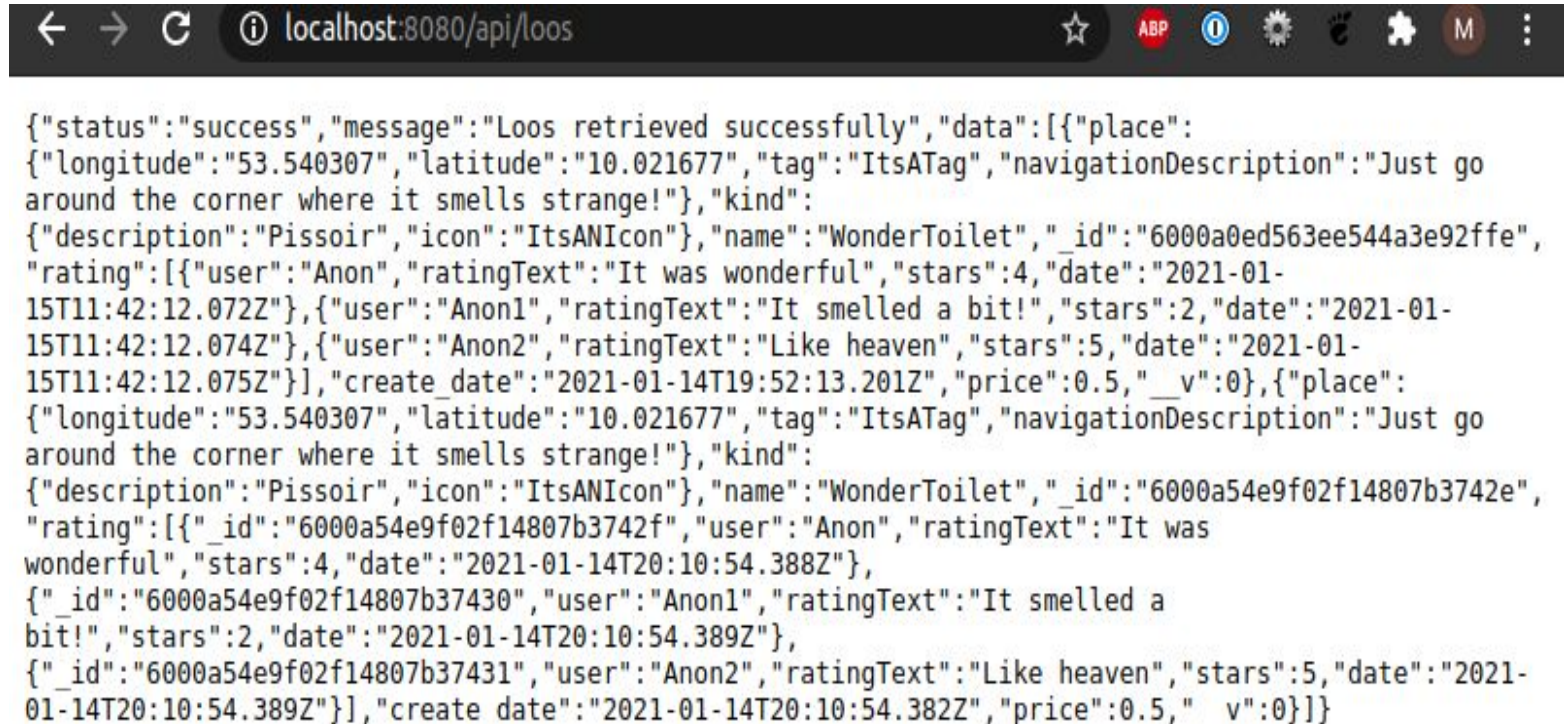
Browser (localhost:8080/api):

```
{
  "status": "Api is working",
  "message": "Welcome to the Loo
  ver. Find the best public loo data!!"}
}
```


Anmerkungen zu bisherige Routes

- Die Routen (Schnittstellen sind hier noch sehr rudimentär). Es werden nur die simpelsten CRUD-Operationen angeboten.

Ausgabe des get-Requests auf alle Loos



```
{
  "status": "success",
  "message": "Loos retrieved successfully",
  "data": [
    {
      "place": {
        "longitude": "53.540307",
        "latitude": "10.021677",
        "tag": "ItsATag",
        "navigationDescription": "Just go around the corner where it smells strange!"
      },
      "kind": {
        "description": "Pissoir",
        "icon": "ItsANIcon",
        "name": "WonderToilet",
        "_id": "6000a0ed563ee544a3e92ffe",
        "rating": [
          {
            "user": "Anon",
            "ratingText": "It was wonderful",
            "stars": 4,
            "date": "2021-01-15T11:42:12.072Z"
          },
          {
            "user": "Anon1",
            "ratingText": "It smelled a bit!",
            "stars": 2,
            "date": "2021-01-15T11:42:12.074Z"
          },
          {
            "user": "Anon2",
            "ratingText": "Like heaven",
            "stars": 5,
            "date": "2021-01-15T11:42:12.075Z"
          }
        ],
        "create_date": "2021-01-14T19:52:13.201Z",
        "price": 0.5,
        "__v": 0
      },
      "place": {
        "longitude": "53.540307",
        "latitude": "10.021677",
        "tag": "ItsATag",
        "navigationDescription": "Just go around the corner where it smells strange!"
      },
      "kind": {
        "description": "Pissoir",
        "icon": "ItsANIcon",
        "name": "WonderToilet",
        "_id": "6000a54e9f02f14807b3742e",
        "rating": [
          {
            "_id": "6000a54e9f02f14807b3742f",
            "user": "Anon",
            "ratingText": "It was wonderful",
            "stars": 4,
            "date": "2021-01-14T20:10:54.388Z"
          },
          {
            "_id": "6000a54e9f02f14807b37430",
            "user": "Anon1",
            "ratingText": "It smelled a bit!",
            "stars": 2,
            "date": "2021-01-14T20:10:54.389Z"
          },
          {
            "_id": "6000a54e9f02f14807b37431",
            "user": "Anon2",
            "ratingText": "Like heaven",
            "stars": 5,
            "date": "2021-01-14T20:10:54.389Z"
          }
        ],
        "create_date": "2021-01-14T20:10:54.382Z",
        "price": 0.5,
        "__v": 0
      }
    ]
  }
}
```