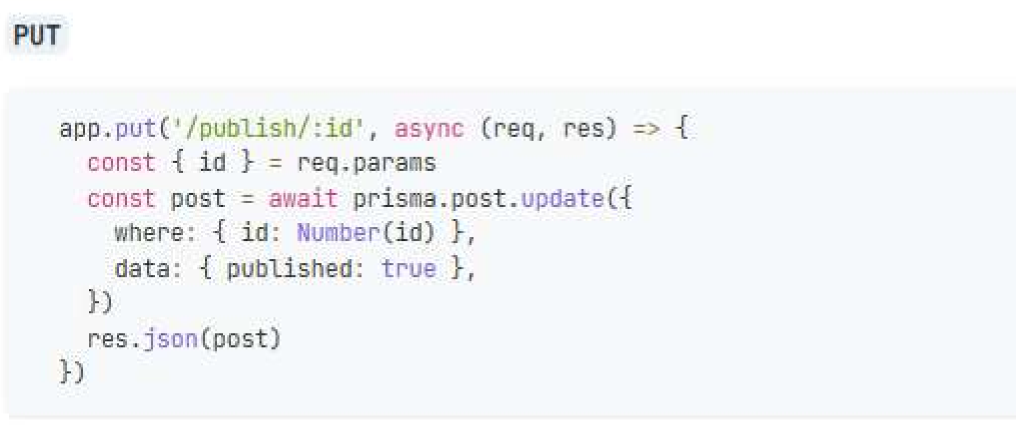
PUT Request

➔ localhost:3000/api/updatePostByID/1

https://www.prisma.io/docs/orm/overview/prisma-in-your-stack/rest

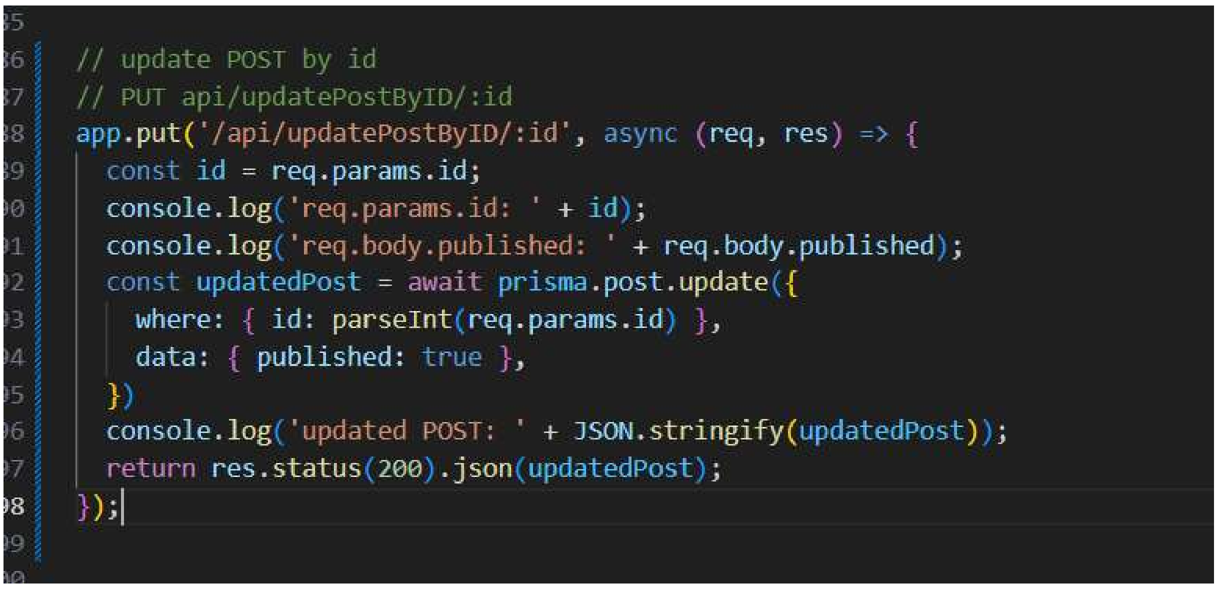
REST API server example



PUT

```
app.put('/publish/:id', async (req, res) => {
  const { id } = req.params
  const post = await prisma.post.update({
    where: { id: Number(id) },
    data: { published: true },
  })
  res.json(post)
})
```

Server.js



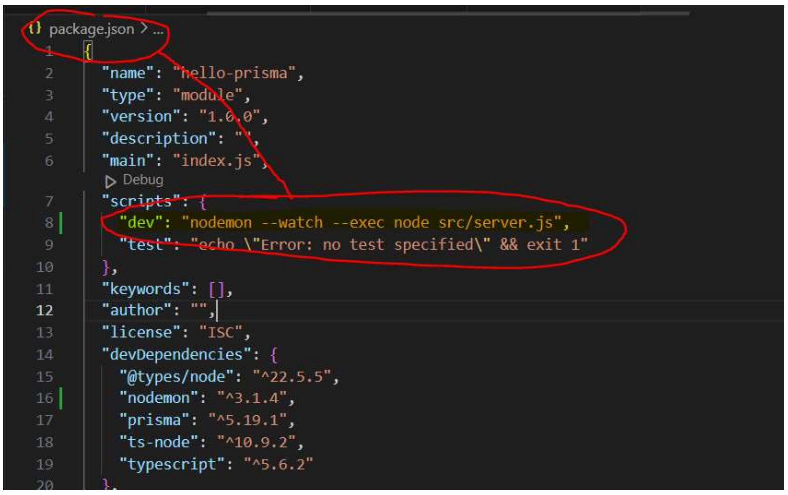
```
// update POST by id
// PUT api/updatePostByID/:id
app.put('/api/updatePostByID/:id', async (req, res) => {
  const id = req.params.id;
  console.log('req.params.id: ' + id);
  console.log('req.body.published: ' + req.body.published);
  const updatedPost = await prisma.post.update({
    where: { id: parseInt(req.params.id) },
    data: { published: true },
  })
  console.log('updated POST: ' + JSON.stringify(updatedPost));
  return res.status(200).json(updatedPost);
});
```

Server.js

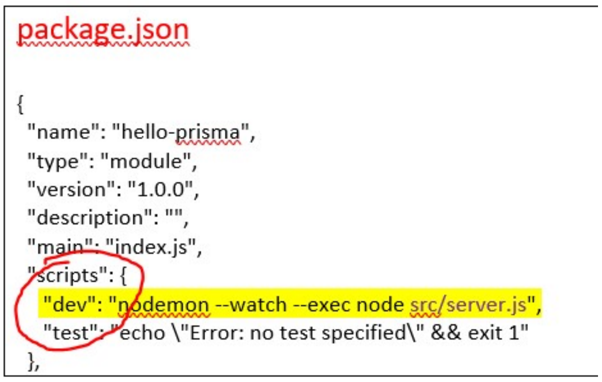
```
import { PrismaClient } from '@prisma/client';
import express from 'express';

const app = express();
const prisma = new PrismaClient();

// update POST by id
// PUT api/updatePostByID/:id
app.put('/api/updatePostByID/:id', async (req, res) => {
  const id = req.params.id;
  console.log('req.params.id: ' + id);
  console.log('req.body.published: ' + req.body.published);
  const updatedPost = await prisma.post.update({
    where: { id: parseInt(req.params.id) },
    data: { published: true },
  })
  console.log('updated POST: ' + JSON.stringify(updatedPost));
  return res.status(200).json(updatedPost);
});


app.listen(3000, () => {
    console.log('Server is running at port 3000');
})
```

On package.json:

```
"dev": "nodemon --watch --exec node src/server.js",
```



```
{
  "name": "hello-prisma",
  "type": "module",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "dev": "nodemon --watch --exec node src/server.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "@types/node": "^22.5.5",
    "nodemon": "^3.1.4",
    "prisma": "^5.19.1",
    "ts-node": "^10.9.2",
    "typescript": "^5.6.2"
  },
```



package.json

```
{
 "name": "hello-prisma",
 "type": "module",
 "version": "1.0.0",
 "description": "",
 "main": "index.js",
 "scripts": {
   "dev": "nodemon --watch --exec node src/server.js",
   "test": "echo \"Error: no test specified\" && exit 1"
 },
```

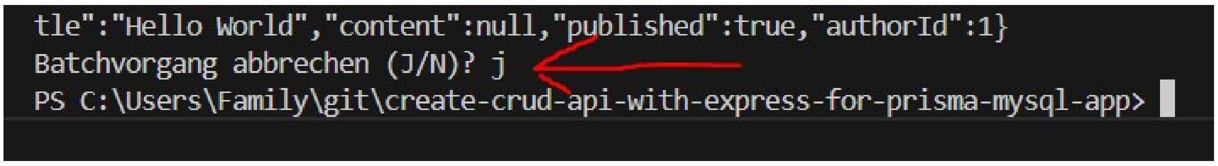
package.json

```
{
 "name": "hello-prisma",
 "type": "module",
 "version": "1.0.0",
 "description": "",
 "main": "index.js",
 "scripts": {
   "dev": "nodemon --watch --exec node src/server.js",
   "test": "echo \"Error: no test specified\" && exit 1"
 },
```

➔ RE-RUN SERVER…

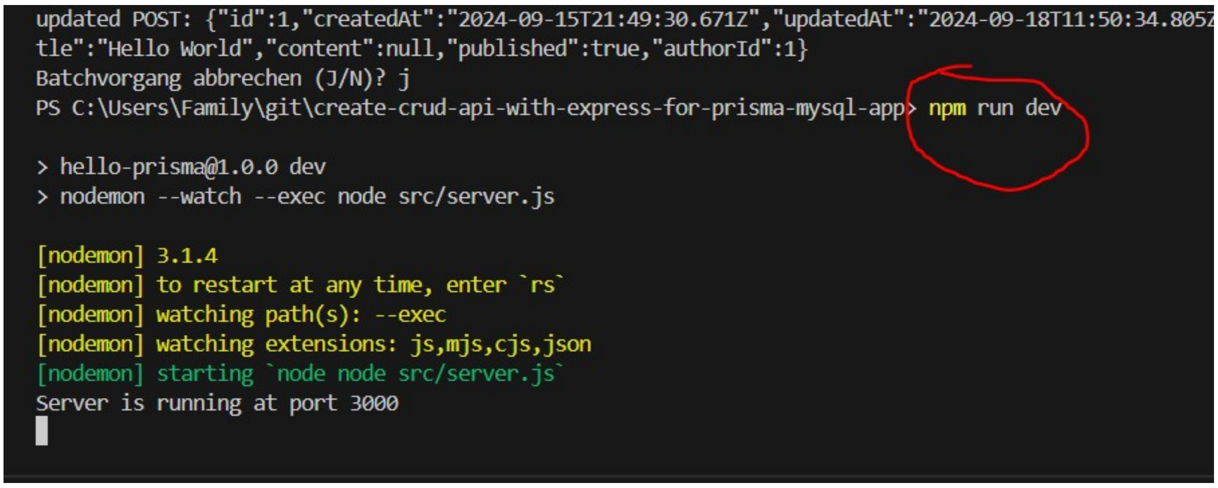
(As long as change something in the server.js. We need to re-run the server.)

➔ Ctrl + C → Type → J → Stop the Server



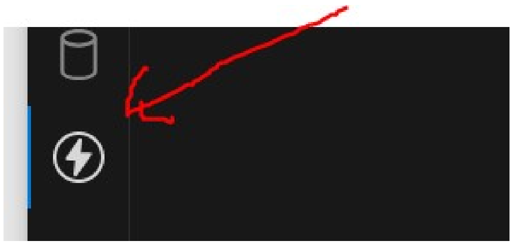
```
tle":"Hello World","content":null,"published":true,"authorId":1}
Batchvorgang abbrechen (J/N)? j
PS C:\Users\Family\git\create-crud-api-with-express-for-prisma-mysql-app>
```

➔ RUN SERVER… | npm run dev



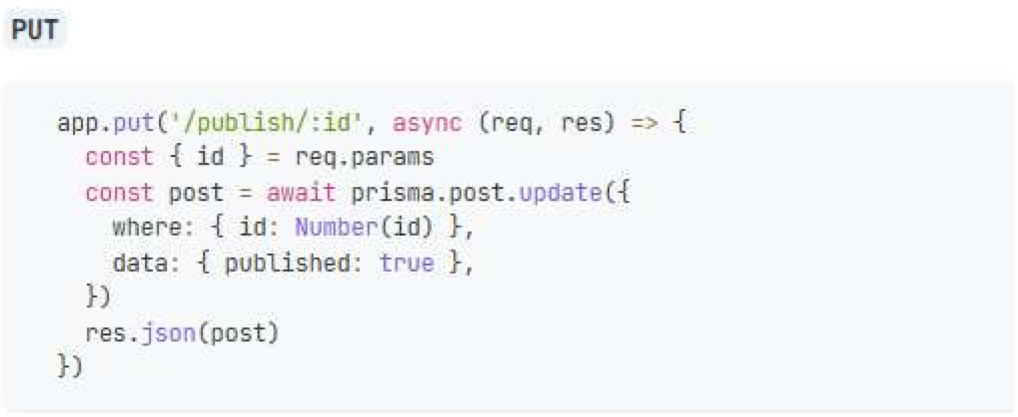
```
updated POST: {"id":1,"createdAt":"2024-09-15T21:49:30.671Z","updatedAt":"2024-09-18T11:50:34.805Z
tle":"Hello World","content":null,"published":true,"authorId":1}
Batchvorgang abbrechen (J/N)? j
PS C:\Users\Family\git\create-crud-api-with-express-for-prisma-mysql-app> npm run dev

> hello-prisma@1.0.0 dev
> nodemon --watch --exec node src/server.js

[nodemon] 3.1.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): --exec
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node node src/server.js`
Server is running at port 3000
```

Thnder Client | http Client



https://www.prisma.io/docs/orm/overview/prisma-in-your-stack/rest

REST API server example

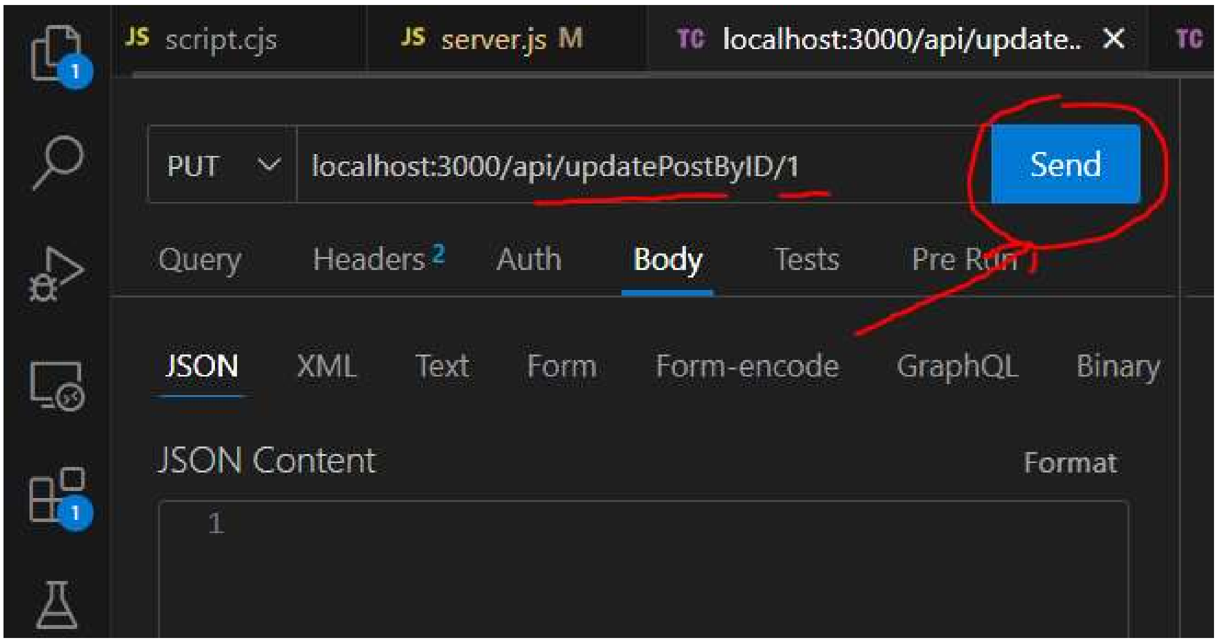


PUT

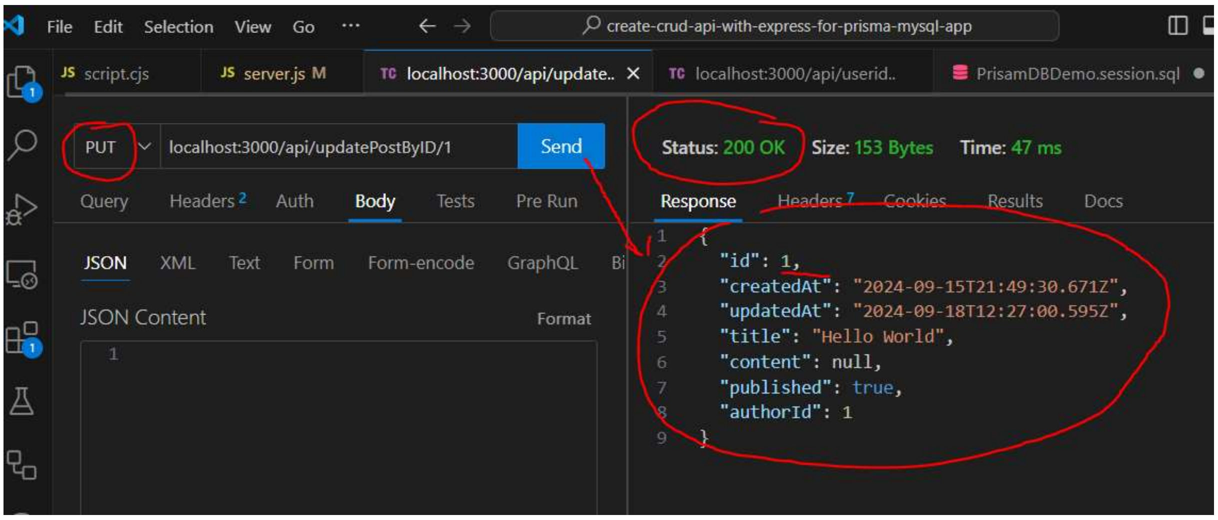
```
app.put('/publish/:id', async (req, res) => {
  const { id } = req.params
  const post = await prisma.post.update({
    where: { id: Number(id) },
    data: { published: true },
  })
  res.json(post)
})
```

PUT → localhost:3000/api/updatePostByID/1

:id = 1



GET Response updated info with Status 200 OK 😊



```
{
  "id": 1,
  "createdAt": "2024-09-15T21:49:30.671Z",
  "updatedAt": "2024-09-18T12:27:00.595Z",
  "title": "Hello World",
  "content": null,
  "published": true,
  "authorId": 1
}
```

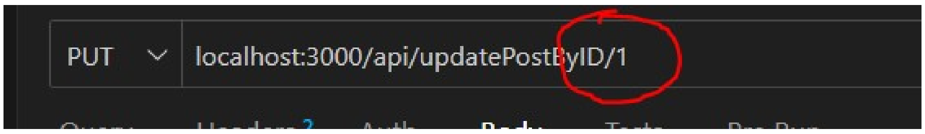
```
{
  "id": 1,
  "createdAt": "2024-09-15T21:49:30.671Z",
  "updatedAt": "2024-09-18T12:27:00.595Z",
  "title": "Hello World",
  "content": null,
  "published": true,
  "authorId": 1
}
```

LOG →

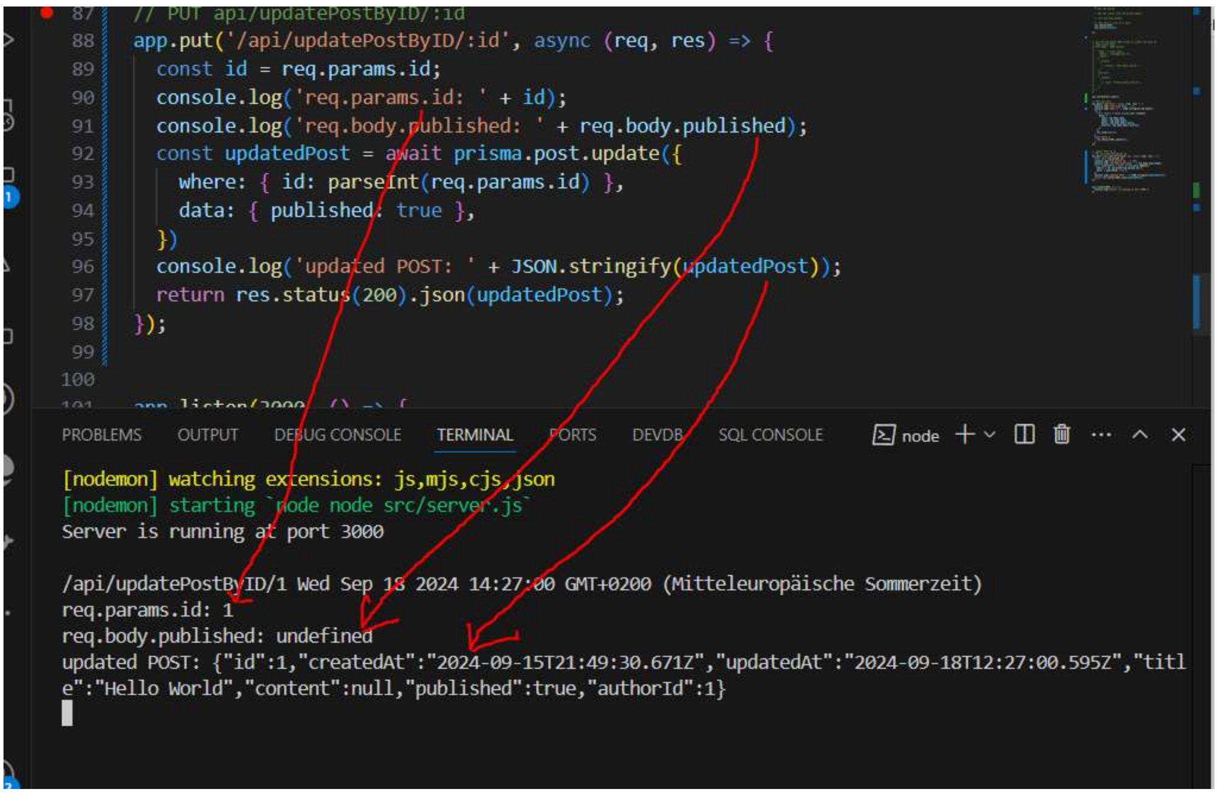
req.body.published: undefined is correct because we do not give any data in the PUT Request Body.

req.params.id: 1 is correct the id was given in the pararmeter

localhost:3000/api/updatePostByID/1



```
PUT    localhost:3000/api/updatePostByID/1
```



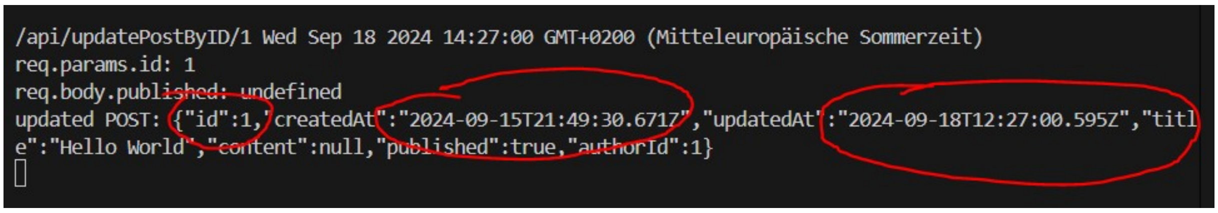
```
// PUT api/updatePostByID/:id
app.put('/api/updatePostByID/:id', async (req, res) => {
  const id = req.params.id;
  console.log('req.params.id: ' + id);
  console.log('req.body.published: ' + req.body.published);
  const updatedPost = await prisma.post.update({
    where: { id: parseInt(req.params.id) },
    data: { published: true },
  })
  console.log('updated POST: ' + JSON.stringify(updatedPost));
  return res.status(200).json(updatedPost);
});
```

```
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node node src/server.js`
Server is running at port 3000

/api/updatePostByID/1 Wed Sep 18 2024 14:27:00 GMT+0200 (Mitteleuropäische Sommerzeit)
req.params.id: 1
req.body.published: undefined
updated POST: {"id":1,"createdAt":"2024-09-15T21:49:30.671Z","updatedAt":"2024-09-18T12:27:00.595Z","titl
e":"Hello World","content":null,"published":true,"authorId":1}
```

LOG OK 😋

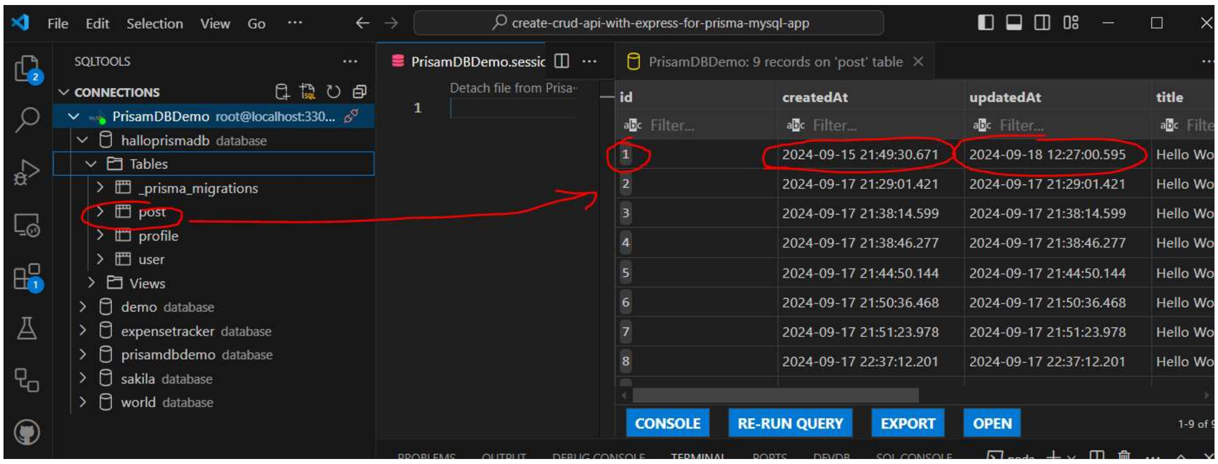
MySQL database

LOG INFO



```
/api/updatePostByID/1 Wed Sep 18 2024 14:27:00 GMT+0200 (Mitteleuropäische Sommerzeit)
req.params.id: 1
req.body.published: undefined
updated POST: {"id":1,"createdAt":"2024-09-15T21:49:30.671Z","updatedAt":"2024-09-18T12:27:00.595Z","titl
e":"Hello World","content":null,"published":true,"authorId":1}
```

Database updated INFO → OK



id	createdAt	updatedAt	title
1	2024-09-15 21:49:30.671	2024-09-18 12:27:00.595	Hello Wo
2	2024-09-17 21:29:01.421	2024-09-17 21:29:01.421	Hello Wo
3	2024-09-17 21:38:14.599	2024-09-17 21:38:14.599	Hello Wo
4	2024-09-17 21:38:46.277	2024-09-17 21:38:46.277	Hello Wo
5	2024-09-17 21:44:50.144	2024-09-17 21:44:50.144	Hello Wo
6	2024-09-17 21:50:36.468	2024-09-17 21:50:36.468	Hello Wo
7	2024-09-17 21:51:23.978	2024-09-17 21:51:23.978	Hello Wo
8	2024-09-17 22:37:12.201	2024-09-17 22:37:12.201	Hello Wo