Run Server: npm run dev1  
Run Client: Thunder Client  
Run Database: MySQL

DELETE Request

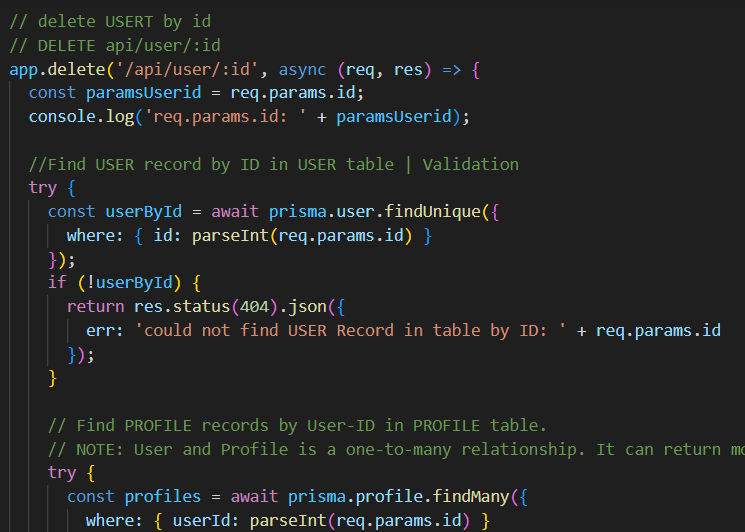
* localhost:3000/api/user/5

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

**REST API server example**



serverDeleteRequest.js



serverDeleteRequest.js

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

import express from 'express';

const app = express();

const prisma = new PrismaClient();

// custom middleware

app.use((req, res, next) => {

  console.log(`${req.url} ${new Date()}`);

  next(); // call the next middleware in the stack

})

//----------------------------------------------------------

// RUN THIS SERVER:

// nodemon --watch --exec node src/serverDeleteRequest.js

//----------------------------------------------------------

/\*

// SEE: prisma.schema.prisma

// USER and POST is one-to-many relationship.

// USER and PROFILES is one-to-many relationship.

// TO DELETE A POST RECORED, WE NEED FIRST TO

// 1. DELETE THE POST WITH FOREIGN-KEY: POST-authorId=USER-id

// 2. DELETE THE PROFILE WITH FOREIGN-KEY: PROFILEID-userid=USER-id

// 3. DELETE THE USER WITH id (USER-id=POST-authorId=PROFILEID-userid)

// FOR EXAMPLE. TO DELETE A USER-id=18;

// CHECK:

// select \* from post where authorId=18;

// select \* from profile where userId=18;

// select \* from user where Id=18;

// DELETE:

// delete from post where authorId=18;

// commit;

// delete from profile where userId=18;

// commit;

// delete from user where id =18;

// commit;

\*/

// delete USERT by id

// DELETE api/user/:id

app.delete('/api/user/:id', async (req, res) => {

  const paramsUserid = req.params.id;

  console.log('req.params.id: ' + paramsUserid);

  //Find USER record by ID in USER table | Validation

  try {

    const userById = await prisma.user.findUnique({

      where: { id: parseInt(req.params.id) }

    });

    if (!userById) {

      return res.status(404).json({

        err: 'could not find USER Record in table by ID: ' + req.params.id

      });

    }

    // Find PROFILE records by User-ID in PROFILE table.

    // NOTE: User and Profile is a one-to-many relationship. It can return more than one records.

    try {

      const profiles = await prisma.profile.findMany({

        where: { userId: parseInt(req.params.id) }

      });

      console.log('PROFILES records found by userId: ' + JSON.stringify(profiles));

      // delete all Profile records with Foreign key (PROFILE-User-ID = USER-ID)

      // If found process looping all found records and delete them by id.

      if (profiles && profiles.length > 0) {

        for (let i = 0; i < profiles.length; i++) {

          console.log('process delete PROFILES with ID): ' + profiles[i].id);

          let deletedProfilesById = await prisma.profile.delete({

            where: { id: parseInt(profiles[i].id) }

          });

        }

      }

    }

    catch (err) {

      console.log('Delete Profil records failed with err: ' + JSON.stringify(err));

      console.log('Profile record could not deleted with userId: ' + req.params.id);

    }

    // Find POST records by Author-ID in POST table.

    // NOTE: User and POST is a one-to-many relationship. It can return more than one records.

    try {

      const posts = await prisma.post.findMany({

        where: { authorId: parseInt(req.params.id) }

      });

      console.log('POST records found by authorId: ' + JSON.stringify(posts));

      // delete all Post records with Foreign key (POST-Author-ID = USER-ID)

      // If found process looping all found records and delete them by id.

      if (posts && posts.length > 0) {

        for (let i = 0; i < posts.length; i++) {

          console.log('process delete POST with ID): ' + posts[i].id);

          let deletedPostById = await prisma.post.delete({

            where: { id: parseInt(posts[i].id) }

          });

        }

      }

    }

    catch (err) {

      console.log('Delete POST records faild with err: ' + JSON.stringify(err));

      console.log('Post record could not deleted with authorId: ' + req.params.id);

    }

    // delete User

    try {

      const deletedUser = await prisma.user.delete({

        where: {

          id: parseInt(req.params.id)

        }

      })

      console.log('delete User successfully. deleted-User: ' + JSON.stringify(deletedUser));

      return res.status(200).json(deletedUser);

    }

    catch (err) {

      console.log('Delete User record failed with Id: ' + req.params.id);

      console.log('err: ' + JSON.stringify(err));

      return res.status(404).json(err);

    }

  }

  catch (err) {

    console.log('User record not found with id: ' + paramsUserid);

    return res.status(404).json(err);

  }

});

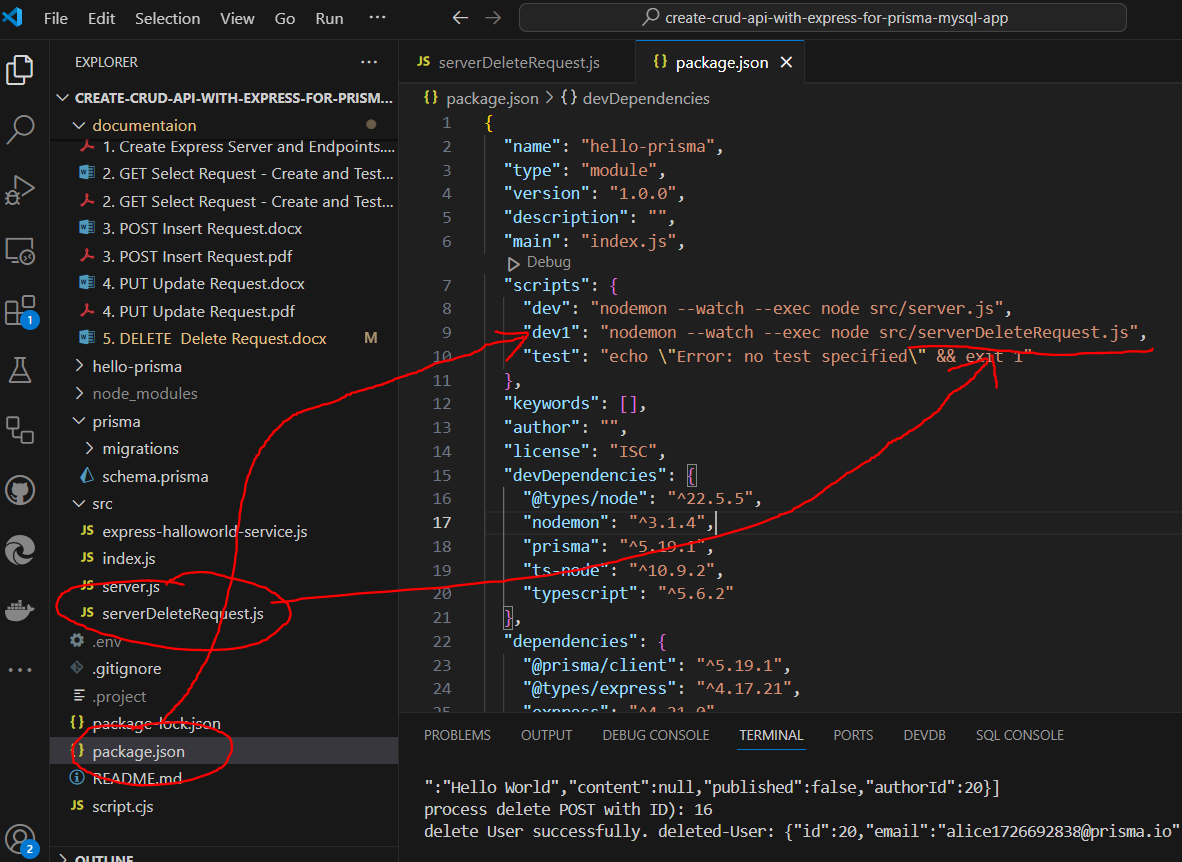
app.listen(3000, () => {

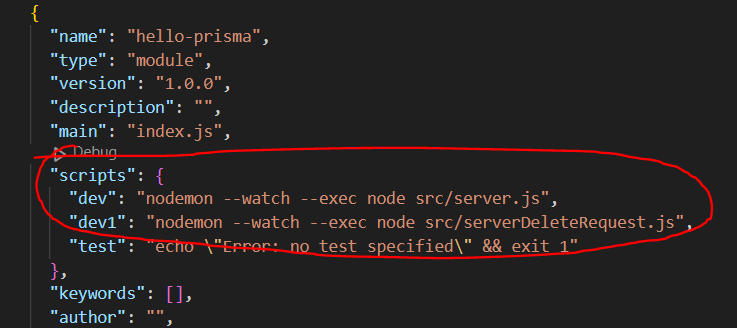
  console.log('Server is running at port 3000');

})

On package.json:   add dev1.

"dev1": "nodemon --watch --exec node src/serverDeleteRequest.js",





**Package.json**

{

  "name": "hello-prisma",

  "type": "module",

  "version": "1.0.0",

  "description": "",

  "main": "index.js",

  "scripts": {

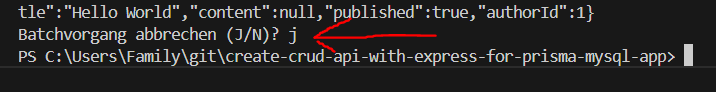
    "dev": "nodemon --watch --exec node src/server.js",

    "dev1": "nodemon --watch --exec node src/serverDeleteRequest.js",

    "test": "echo \"Error: no test specified\" && exit 1"

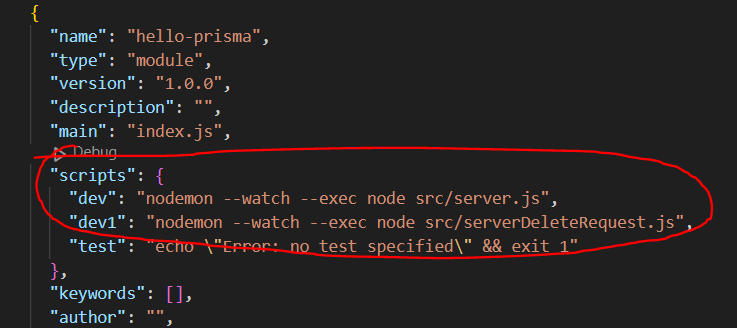
  },

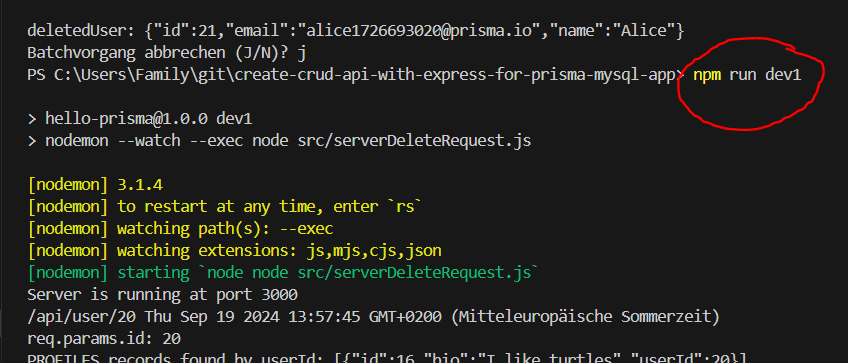
* RE-RUN SERVER…   
  (As long as change something in the serverDeleteRequest.js. We need to re-run the server.)
* Ctrl + C 🡪 Type 🡪 J 🡪 Stop the Server

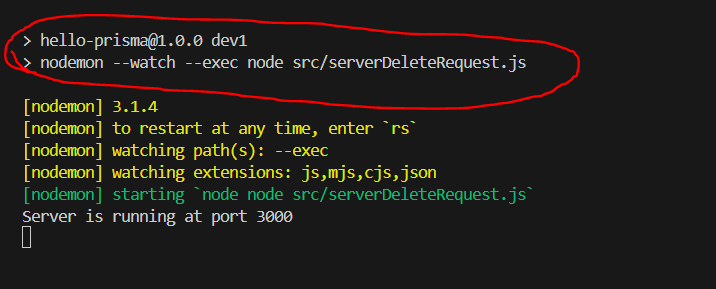


* RUN SERVER… | npm run dev1

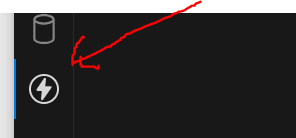
C:\Users\Family\git\create-crud-api-with-express-for-prisma-mysql-app> npm run dev1







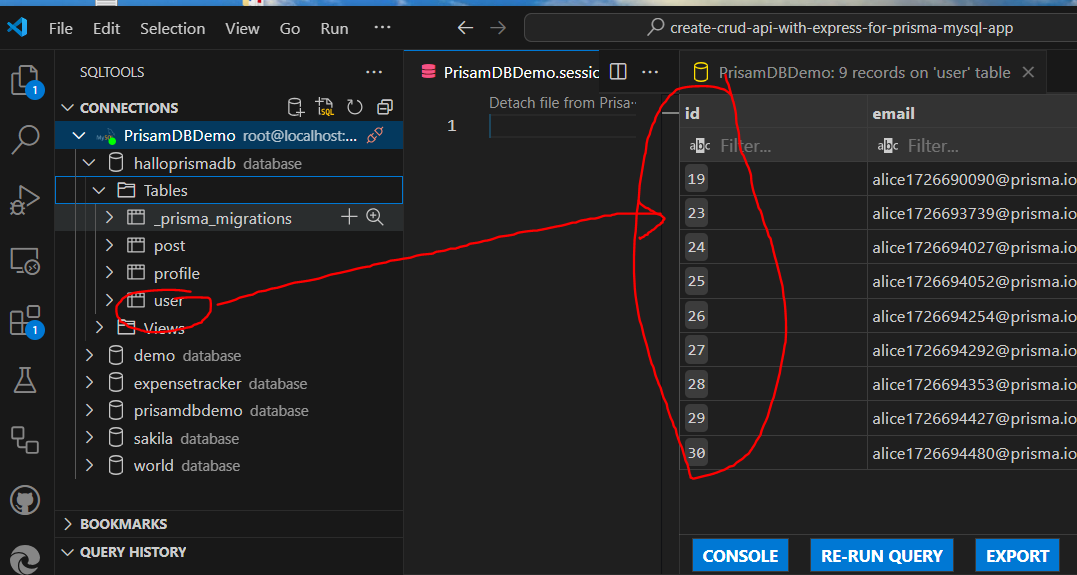
* Thnder Client | http Client



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

**REST API server example**

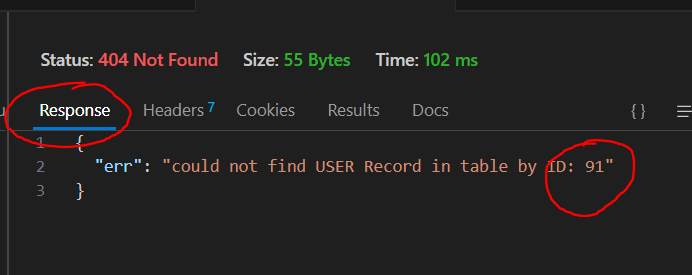
 **MySQL database USER table without ID=91. Use for ERROR validation.**



**ERROR Validation**

**DELETE 🡪 localhost:3000/api/user/91**

**:id = 91**



GET Response deleted error info with Status 404 Not Found OK 😊

{

"err": "could not find USER Record in table by ID: 91"

}

LOG 🡪

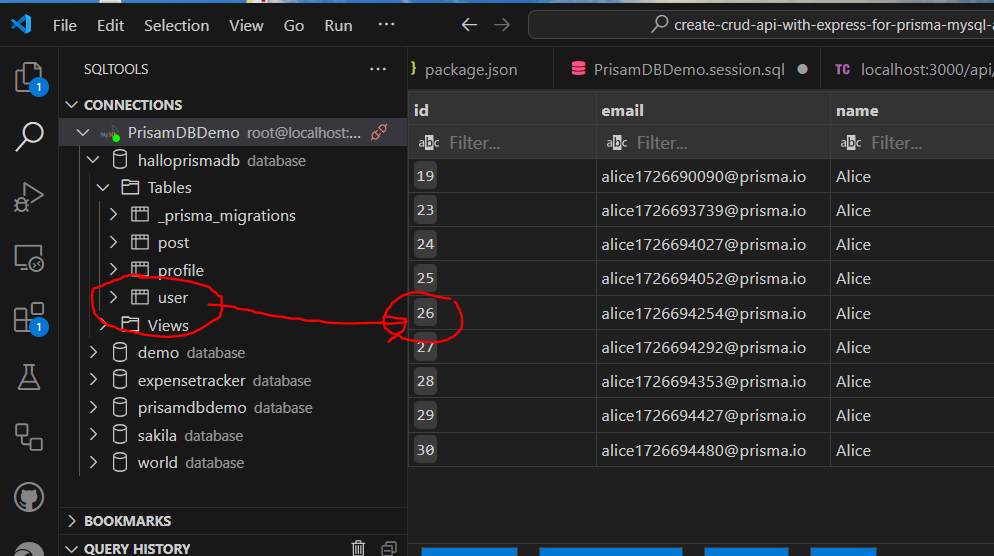


**DELETE USER with ID=26**

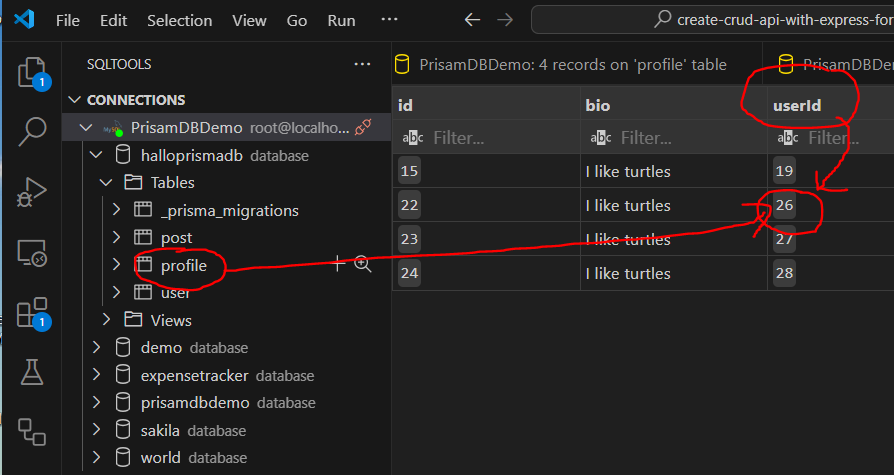
**DELETE 🡪 localhost:3000/api/user/26**

**:id = 26**

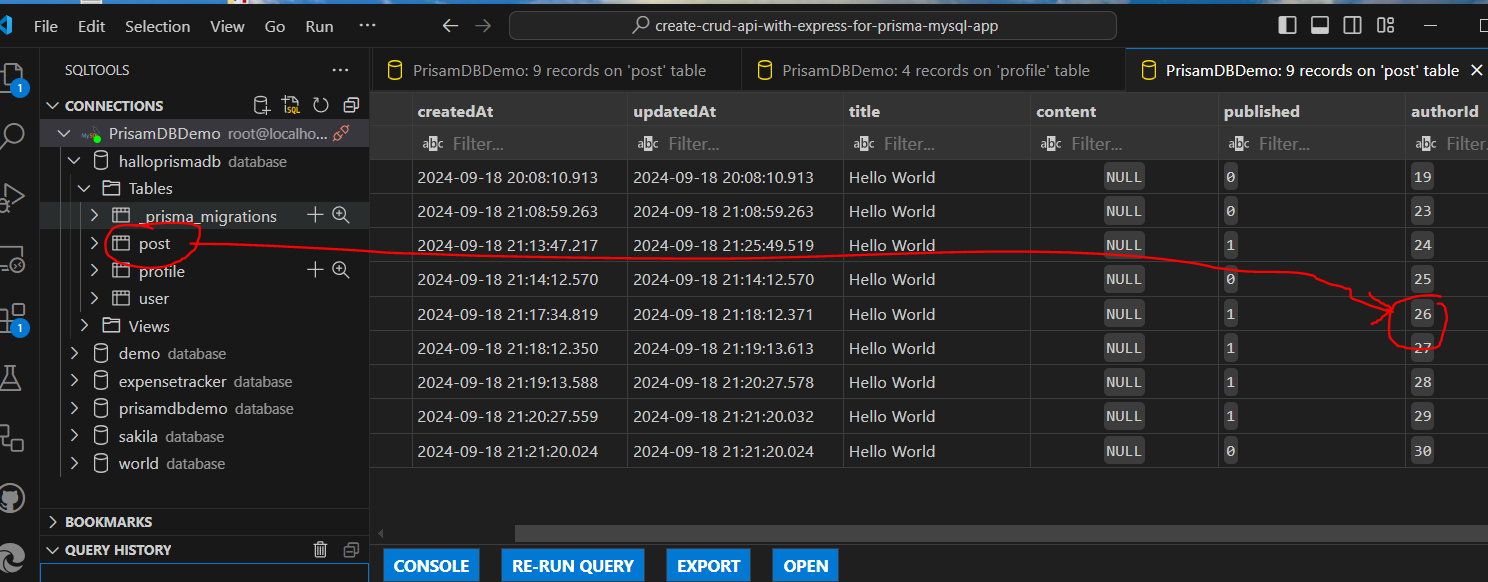
User contains id=26



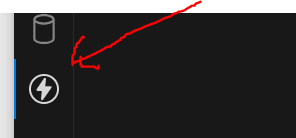
Profile contains foreign key userId=26



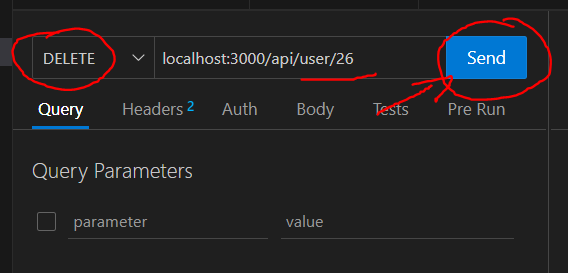
POST contains foreign key authorId 26

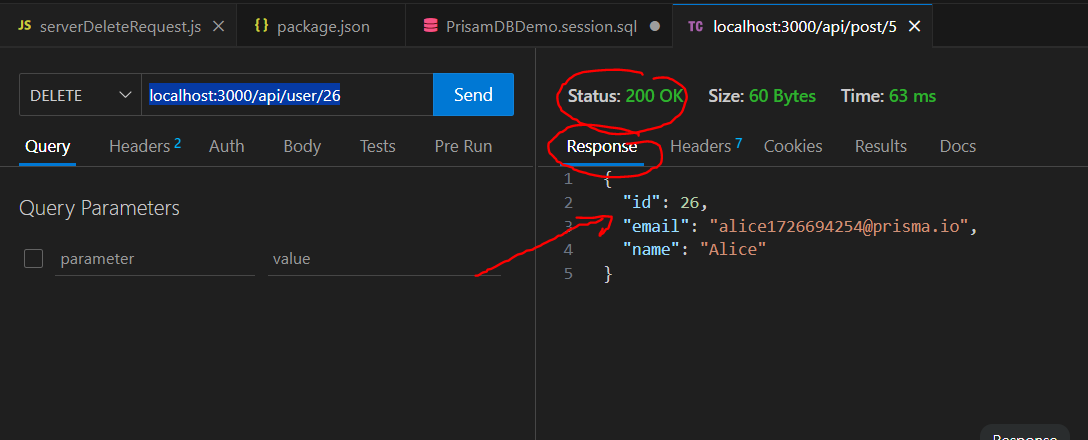


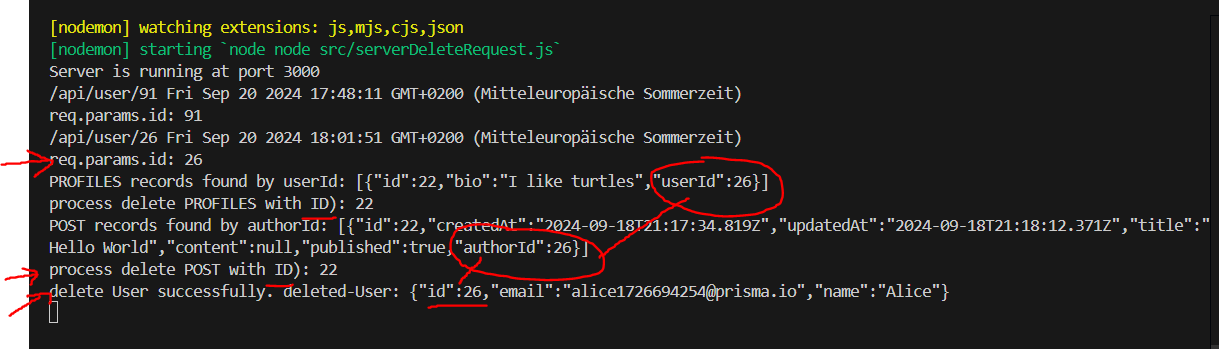
Click 🡪 Thunder CLient



DELETE 🡪 localhost:3000/api/user/26

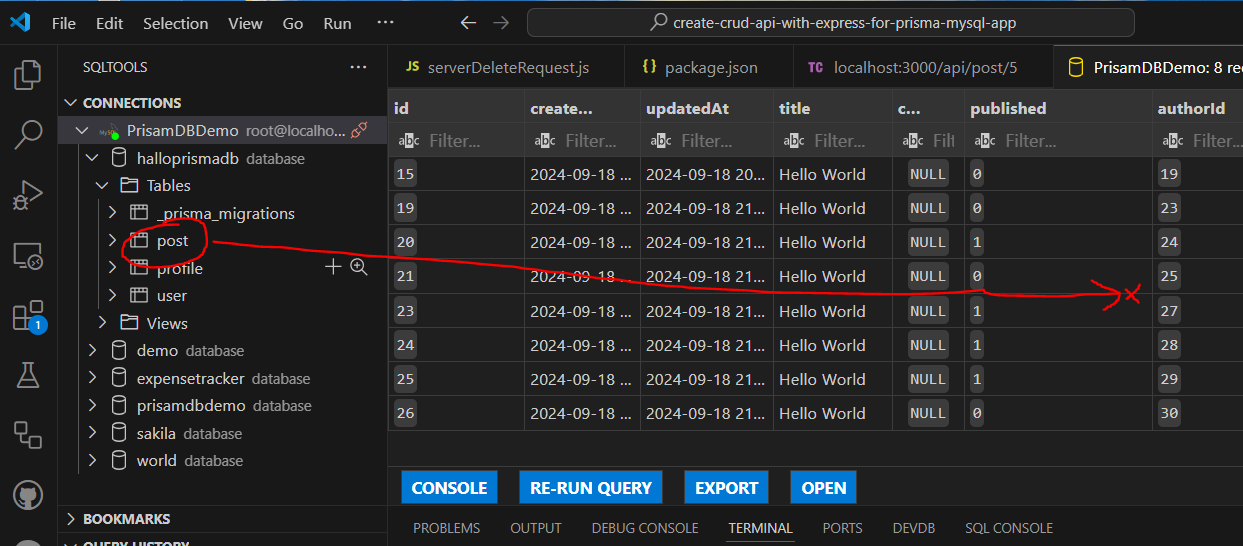


**USER, POST, PROFILE were delted with Status 200 OK and Response message with id=26**

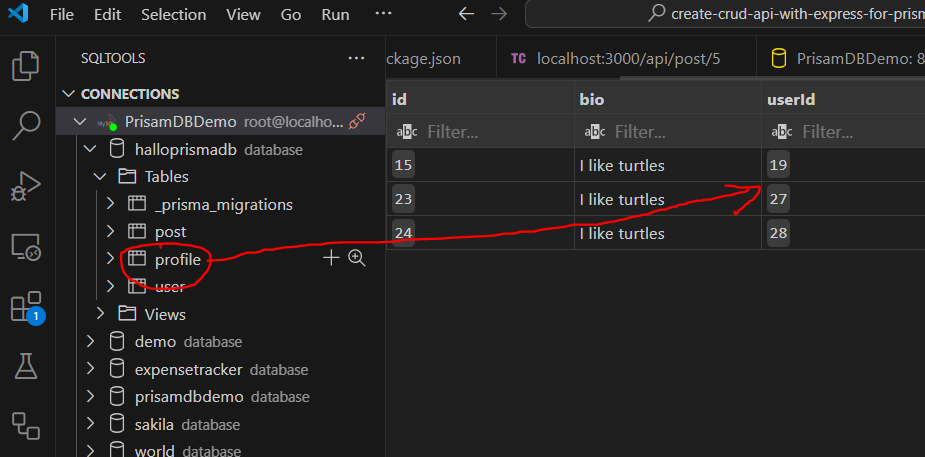
LOG 🡪  


My SQL Database

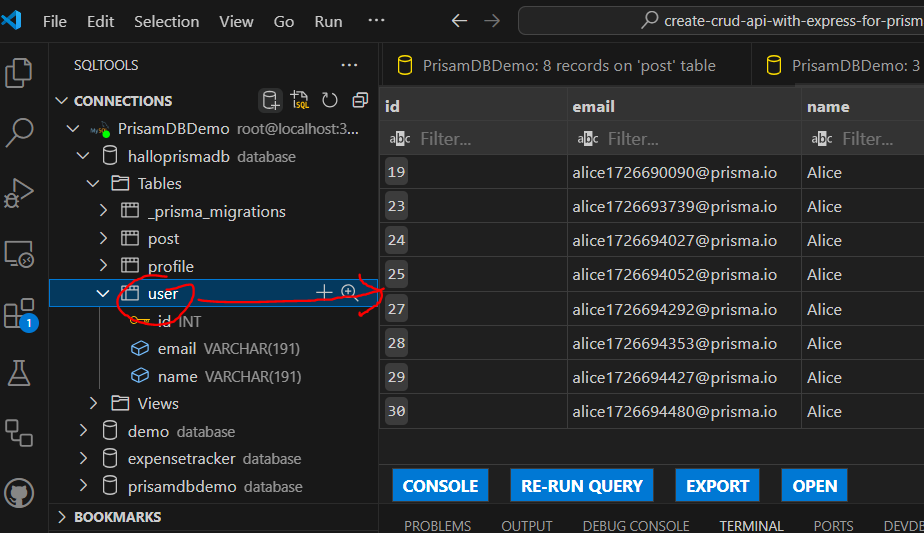
POST authorId=26 was deleted.



Profile contains userId=26 was deleted



User contans id=26 was deleted



**USER, POST, PROFILE were delted successfully** 😊