

# Front-End Web Development & Web Services Project

## Korte uitleg

Voor dit project heb ik een website gemaakt waarop je alle liedjes uit de Gentse studentencodex kan beluisteren. Bij het bezoeken van de website zal je eerst een loginscherm vinden. Hier kan je kiezen om in te loggen, een nieuw account aan te maken of de website als gast te bezoeken. Het hebben van een account heeft nu weinig voordelen, maar was bedoeld om zo je eigen playlists te kunnen aanmaken. Maar door tijdsgebrek heb ik deze feature niet meer kunnen afwerken.

The screenshot shows the login interface of the 'Online StudentenCodex' website. At the top, the title 'Online StudentenCodex' is displayed in a bold, orange font. Below it is a navigation bar with three links: 'Home', 'Afspeellijsten', and 'Mijn Account'. The main content area is titled 'Log in op je account' and contains a login form. The form has two input fields: 'Gebruikersnaam:' and 'Wachtwoord:'. Below these fields is a 'Log in' button. At the bottom of the form, there are two links: 'Ik heb nog geen account' and 'Verdergaan als gast'.

The screenshot shows the registration interface of the 'Online StudentenCodex' website. At the top, the title 'Online StudentenCodex' is displayed in a bold, orange font. Below it is a navigation bar with three links: 'Home', 'Afspeellijsten', and 'Mijn Account'. The main content area is titled 'Maak een nieuw account' and contains a registration form. The form has three input fields: 'Gebruikersnaam:', 'Wachtwoord:', and 'Herhaal wachtwoord:'. Below these fields is a 'Registreer' button. At the bottom of the form, there are two links: 'Ik heb al een account' and 'Verdergaan als gast'.

## Online StudentenCodex

[Home](#) [Afspeellijsten](#) [Mijn Account](#)

Log in op je account

Gebruikersnaam:

Wachtwoord:

[Log in](#)

[Ik heb nog geen account](#)

[Verdergaan als gast](#)

[Nieuw wachtwoord en e-mailadres](#)

Na het eerste scherm krijg je een lijst te zien met alle beschikbare nummers. Als je op een van deze klikt dan kan je het beluisteren. Je hebt ook een optie om je account te beheren, hier kan je, na verificatie, je gebruikersnaam of wachtwoord aanpassen. Ik weet niet precies waarom maar wanneer ik dit probeerde te implementeren wou mijn databank zich niet aanpassen dus deze functie werkt niet naar behoren.

## Online StudentenCodex

[Home](#) [Afspeellijsten](#) [Mijn Account](#)

Welkom op de Hoofdpagina

p37: Io Vivat  
p38: Gaudeamus Igitur  
p40: De slag om het Gravensteen  
p42: Oude roldersklacht  
p44: Ruiterslied  
p46: Ave confrater  
p50: De Vlaamse Leeuw  
p52: Wilhelmus van Nassouwe  
p54: Die stem van Zuid-Afrika  
p56: Vlaenderen boven al  
p57: Het lied der Vlamingen  
p58: Ons vaderland  
p59: Beiaardlied  
p60: Blijheidslied  
p61: De Schelde  
p62: Mijn Vlaenderen heb ik hartelijk lief

## Online StudentenCodex

[Home](#) [Afspeellijsten](#) [Mijn Account](#)

←

### Io Vivat

Te vinden op p37 in de Gentse studenten codex



Online StudentenCodex

Home Afspeellijsten Mijn Account

Verificatie

Gebruikersnaam:

Wachtwoord:

Log in

Ik heb nog geen account

Online StudentenCodex

Home Afspeellijsten Mijn Account

naam: Arno ✎

wachtwoord: hash ✎

Online StudentenCodex

Home Afspeellijsten Mijn Account

Afspeellijsten zijn momenteel niet beschikbaar

## Extra technologie

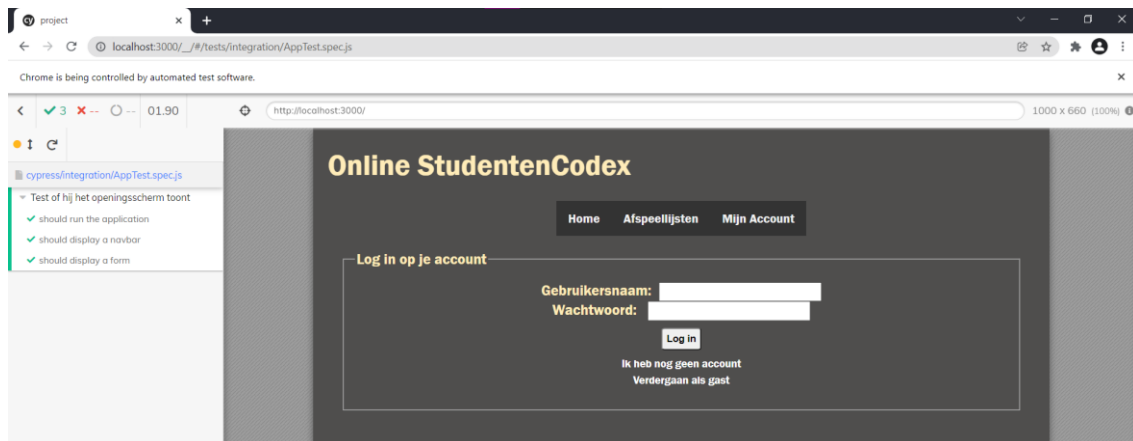
ik heb in mijn API 2 extra technologieën verwerkt. Ik gebruikte node-input-validator om invoervalidatie af te handelen bij het aanmaken van een nieuw account. Om ervoor te zorgen dat wachtwoorden gehashed zouden worden alvorens ze in de databank opgeslagen worden heb ik een package genaamd bcrypt gebruikt.

## Gekende bugs

Bij het inloggen of registreren, moet je soms 2 keer op de login knop drukken voor je effectief ingelogd bent.

## Testen

Ik heb een test voorzien die bij het bezoeken van de website controleren of de webpagina weldegelijk wordt ingeladen.

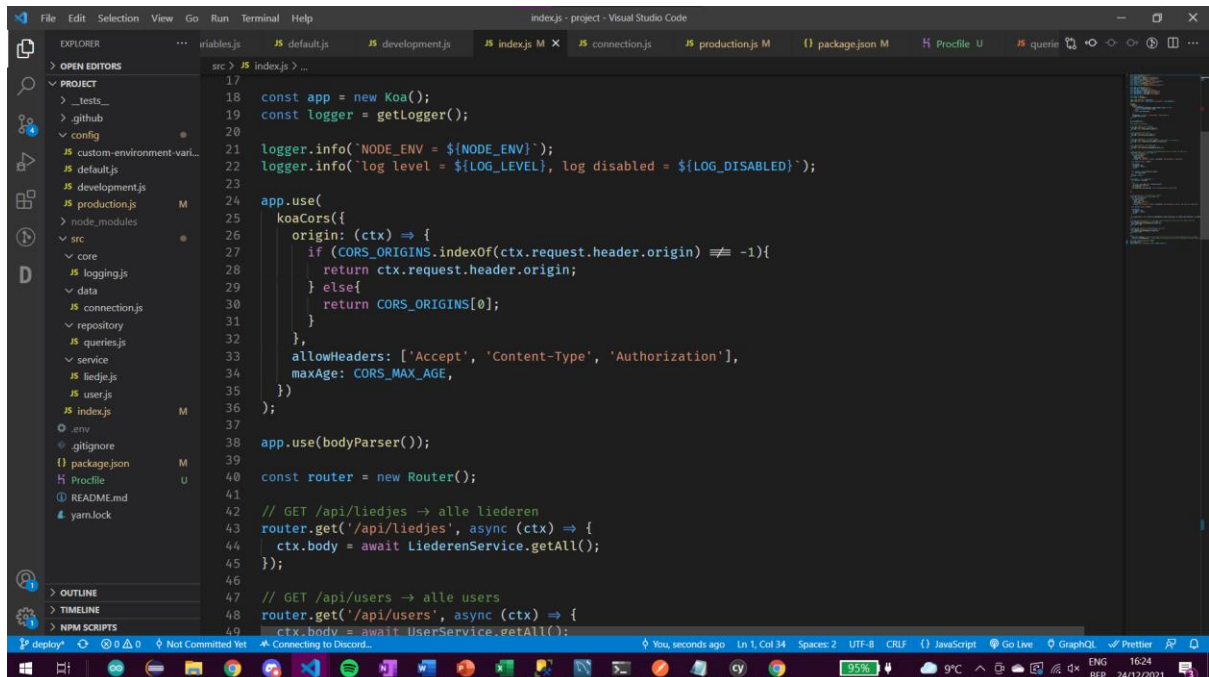


Ik heb 2 testen voorzien die elk controleren of de connectie met de databank goed zit.

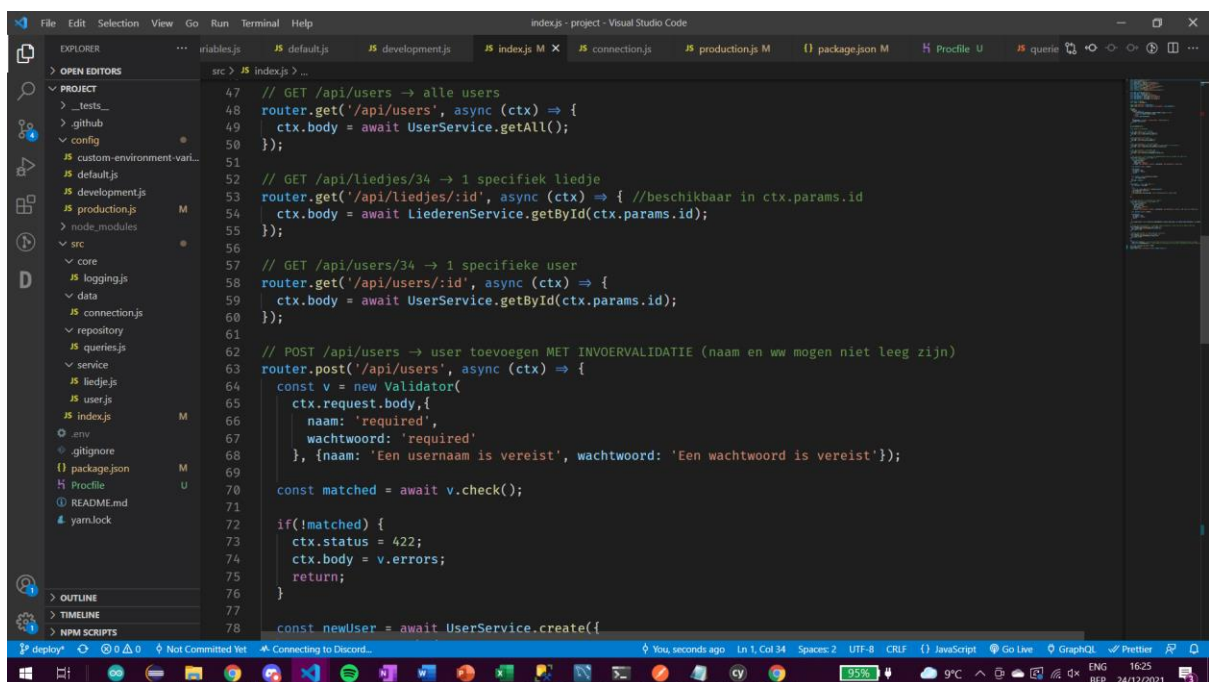
```
1  const ex = require('../src/repository/queries');
2
3  test('getAlIiedjes', async () => {
4    const lied = {paginaNr: 37, naam: 'Io Vivat', iframe: `<iframe width="506" height="285"
5  src="https://www.youtube.com/embed/j9aD2wCvzfQ"
6  title="YouTube video player"
7  frameborder="0" allow="accelerometer;
8  autoplay; clipboard-write; encrypted-media;
9  gyroscope; picture-in-picture" allowfullscreen></iframe>`}
10   const result = await ex.getAlIiedjes();
11   expect(result[0]).toEqual(lied);
12 });
13
14 test('getUserById', async () => {
15   const id = 'fc82abc3-ac43-4d01-918c-32358684ee37'
16   const user = {naam: 'Arno', wachtwoord: '$2a$10$mWHSZtJks76yXqg.2Rw3ceCe02Wo23lqDDth103FGHaUcseDreUHm', id};
17   const result = await ex.getUserById(id);
18   expect(result[0]).toEqual(user);
19 });
```

## screenshots API

### src/index.js



```
17
18 const app = new Koa();
19 const logger = getLogger();
20
21 logger.info(`NODE_ENV = ${NODE_ENV}`);
22 logger.info(`log level = ${LOG_LEVEL}, log disabled = ${LOG_DISABLED}`);
23
24 app.use(
25   koaCors({
26     origin: (ctx) => {
27       if (CORS_ORIGINS.indexOf(ctx.request.header.origin) !== -1){
28         return ctx.request.header.origin;
29       } else{
30         return CORS_ORIGINS[0];
31       }
32     },
33     allowHeaders: ['Accept', 'Content-Type', 'Authorization'],
34     maxAge: CORS_MAX_AGE,
35   })
36 );
37
38 app.use(bodyParser());
39
40 const router = new Router();
41
42 // GET /api/liedjes -> alle liederen
43 router.get('/api/liedjes', async (ctx) => {
44   ctx.body = await LiederenService.getAll();
45 });
46
47 // GET /api/users -> alle users
48 router.get('/api/users', async (ctx) => {
49   ctx.body = await UserService.getAll();
50 });
```



```
47 // GET /api/users -> alle users
48 router.get('/api/users', async (ctx) => {
49   ctx.body = await UserService.getAll();
50 });
51
52 // GET /api/liedjes/34 -> 1 specifiek liedje
53 router.get('/api/liedjes/:id', async (ctx) => { //beschikbaar in ctx.params.id
54   ctx.body = await LiederenService.getById(ctx.params.id);
55 });
56
57 // GET /api/users/34 -> 1 specifieke user
58 router.get('/api/users/:id', async (ctx) => {
59   ctx.body = await UserService.getById(ctx.params.id);
60 });
61
62 // POST /api/users -> user toevoegen MET INVOERVALIDATIE (naam en ww mogen niet leeg zijn)
63 router.post('/api/users', async (ctx) => {
64   const v = new Validator(
65     ctx.request.body, {
66       naam: 'required',
67       wachtwoord: 'required'
68     }, {naam: 'Een gebruikersnaam is vereist', wachtwoord: 'Een wachtwoord is vereist'});
69
70   const matched = await v.check();
71
72   if(!matched) {
73     ctx.status = 422;
74     ctx.body = v.errors;
75     return;
76   }
77
78   const newUser = await UserService.create({
```

The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying a project structure. The main editor window shows the `index.js` file with the following code:

```
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
```

```
const newUser = await UserService.create({
  ... ctx.request.body,
});
ctx.body = newUser;
});

router.post('/', async (req, res) => {
  const postData = req.body;

  try {
    const ids = await db('posts').insert(postData);
    res.status(201).json(ids);
  } catch (err) {
    res.status(500).json({message: "Error creating new post", error: err})
  }
});

// PUT /api/users/34 → 1 specifieke user updaten
router.put('/api/users/:id', async (ctx) => {
  const v = new Validator(
    ctx.request.body,{
      naam: 'required',
      wachtwoord: 'required',
      id: 'required'
    }, {naam: 'Een gebruikersnaam is vereist', wachtwoord: 'Een wachtwoord is vereist', id: 'Een id is vereist'});

  const matched = await v.check();

  if(!matched) {
    ctx.status = 422;
```

The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying a project structure. The main editor window shows the `index.js` file with the following code:

```
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
```

```
if(!matched) {
  ctx.status = 422;
  ctx.body = v.errors;
  return;
}

ctx.request.body = await UserService.updateById(ctx.request.body.naam, ctx.request.body.wachtwoord, ctx.request.body.id);
});

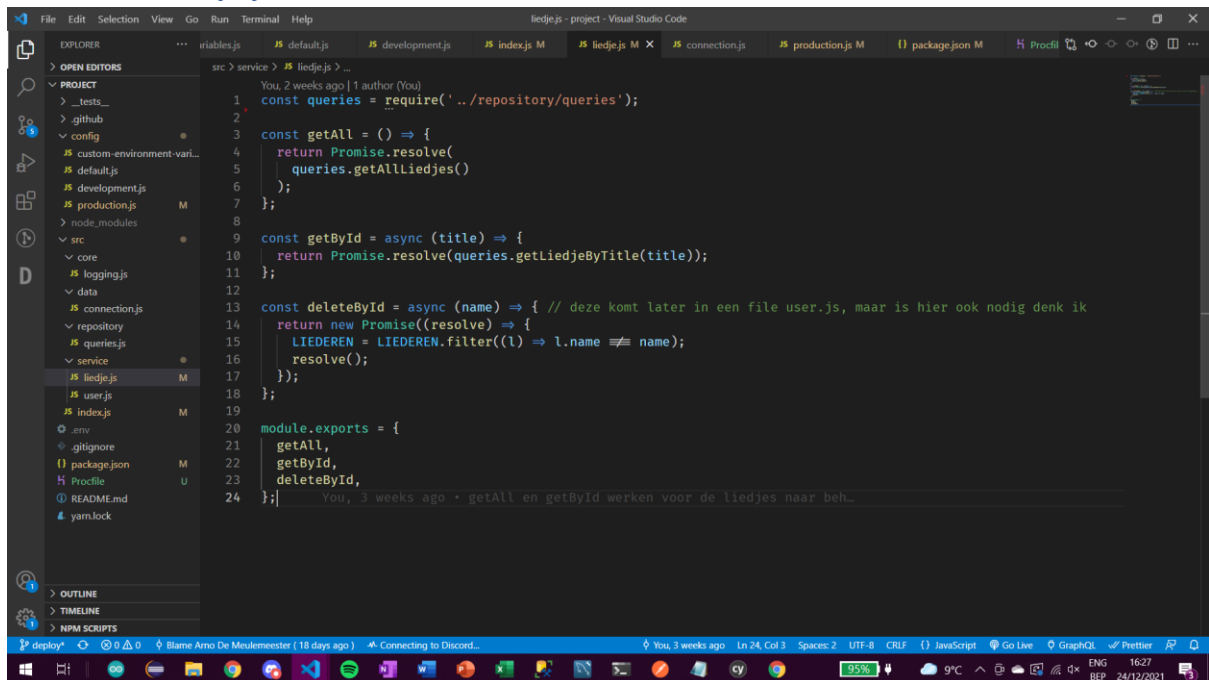
// DELETE /api/transactions/34 → 1 specifiek liedje verwijderen → werkt voor een of andere reden niet
router.delete('/api/liedjes/:id', async (ctx) => {
  await LiederenService.deleteById(ctx.params.id);
  ctx.status = 204;
});

// DELETE /api/users/34 → 1 specifieke user verwijderen
router.delete('/api/users/:id', async (ctx) => {
  await UserService.deleteById(ctx.params.id);
  ctx.status = 204;
});

app
  .use(router.routes()) // bij de juiste method de juiste url de juiste functie middleware(s) aan te roepen
  .use(router.allowedMethods()); // gaat kijken als het method & url zijn die niet kunnen een http 405 (=method not allowed)

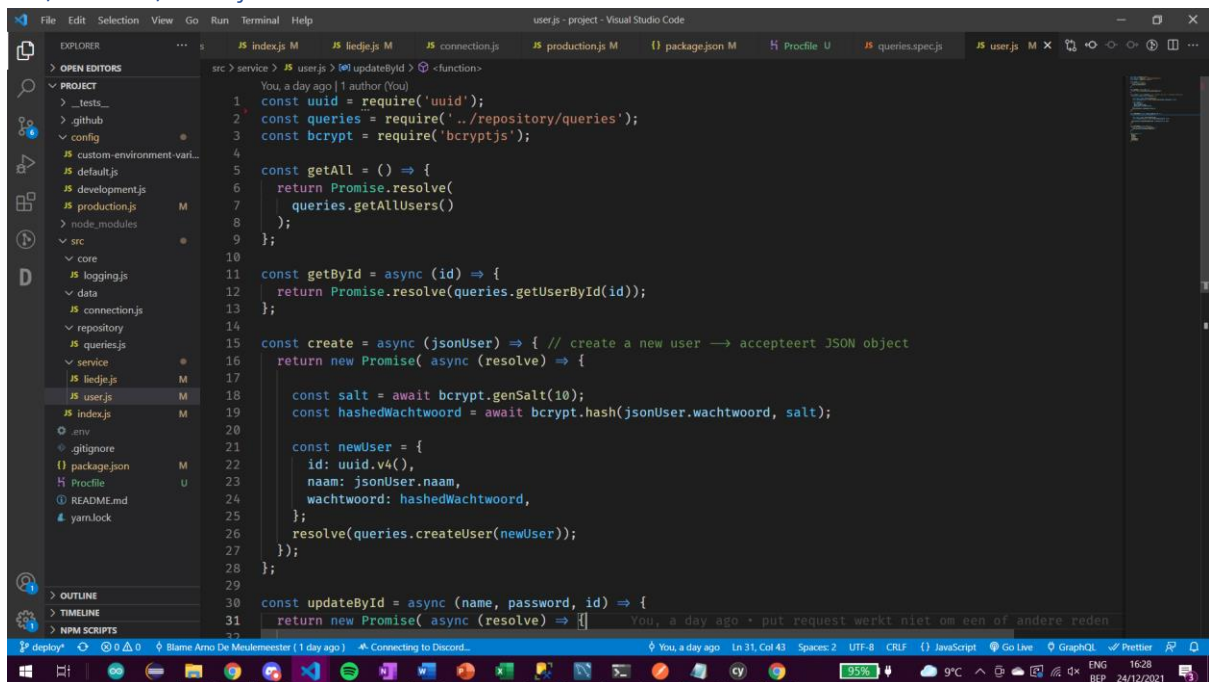
const port = process.env.PORT || 9000;
app.listen(port);
logger.info(`Server listening on http://${HOST}:${port}`);
```

## src/service/liekje.js

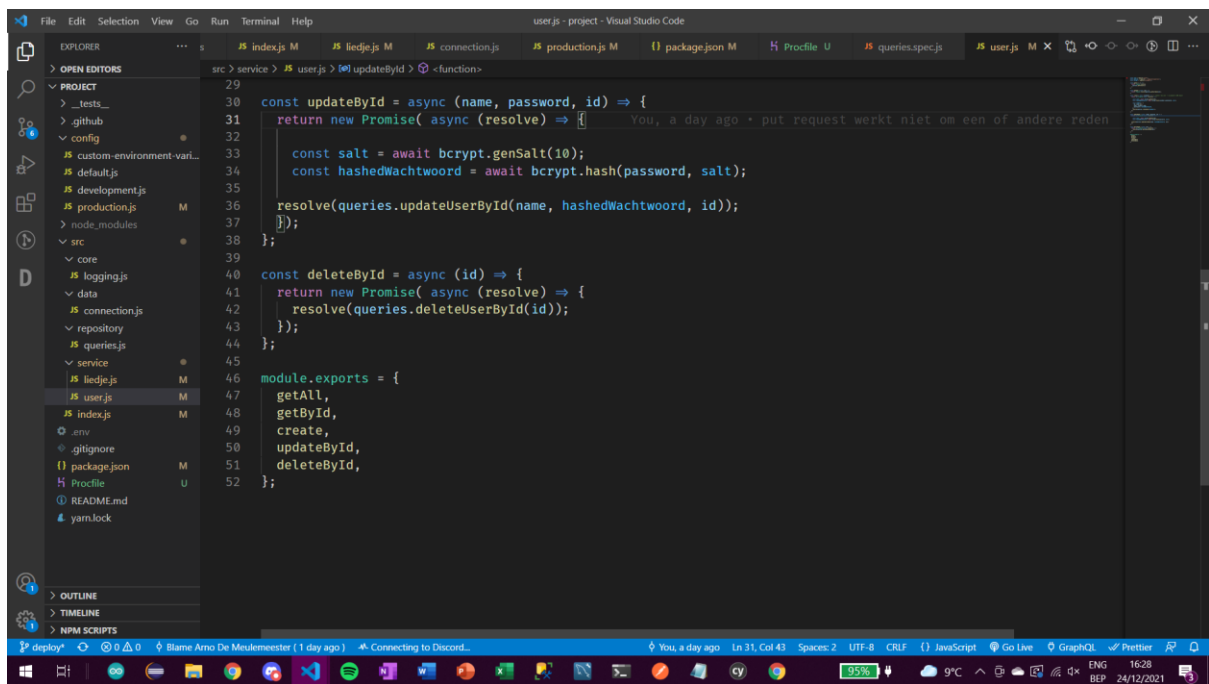


```
1 const queries = require('../repository/queries');
2
3 const getAll = () => {
4   return Promise.resolve(
5     queries.getAllLiedjes()
6   );
7 };
8
9 const getById = async (title) => {
10   return Promise.resolve(queries.getLiedjeByTitle(title));
11 };
12
13 const deleteById = async (name) => { // deze komt later in een file user.js, maar is hier ook nodig denk ik
14   return new Promise((resolve) => {
15     LIEDEREN = LIEDEREN.filter((l) => l.name !== name);
16     resolve();
17   });
18 };
19
20 module.exports = {
21   getAll,
22   getById,
23   deleteById,
24 };
25
26 You, 3 weeks ago • getAll en getById werken voor de liedjes naar beh...
```

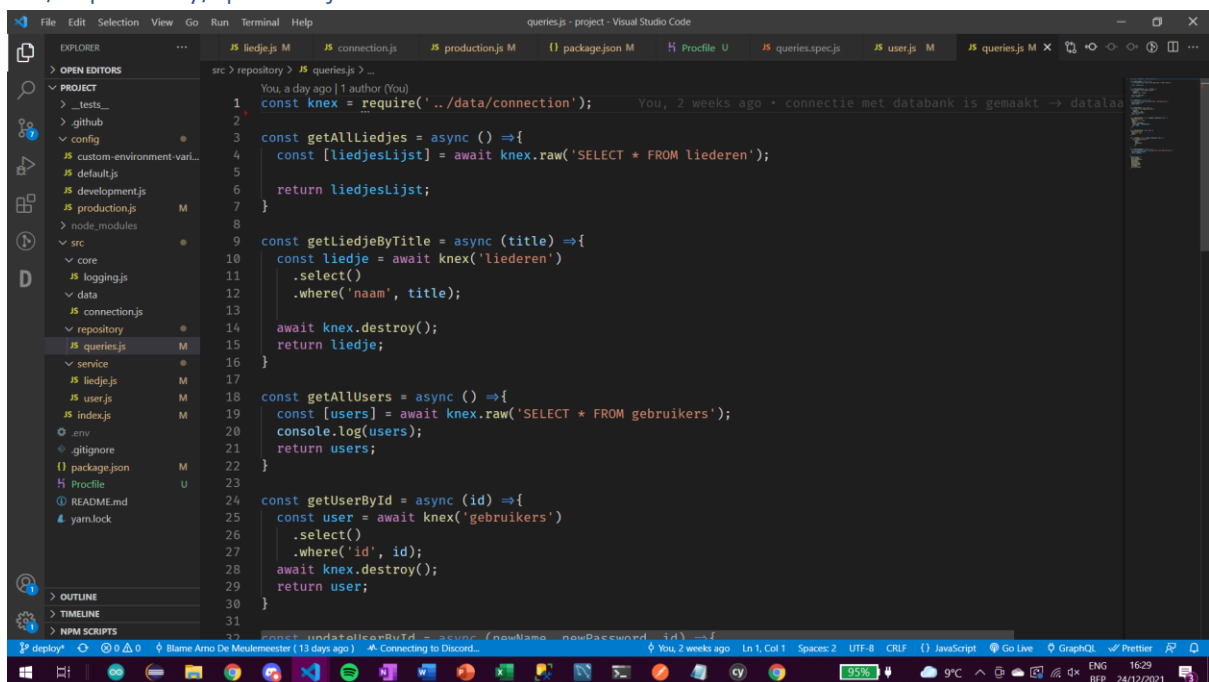
## src/service/user.js



```
1 const uuid = require('uuid');
2 const queries = require('../repository/queries');
3 const bcrypt = require('bcryptjs');
4
5 const getAll = () => {
6   return Promise.resolve(
7     queries.getAllUsers()
8   );
9 };
10
11 const getById = async (id) => {
12   return Promise.resolve(queries.getUserById(id));
13 };
14
15 const create = async (jsonUser) => { // create a new user -> accepteert JSON object
16   return new Promise(async (resolve) => {
17
18     const salt = await bcrypt.genSalt(10);
19     const hashedWachtwoord = await bcrypt.hash(jsonUser.wachtwoord, salt);
20
21     const newUser = {
22       id: uuid.v4(),
23       naam: jsonUser.naam,
24       wachtwoord: hashedWachtwoord,
25     };
26     resolve(queries.createUser(newUser));
27   });
28 };
29
30 const updateById = async (name, password, id) => {
31   return new Promise(async (resolve) => {
32
33     You, a day ago • put request werkt niet om een of andere reden...
```



## src/repository/queries.js





queries.js - project - Visual Studio Code

```
src > repository > JS queries.js > ...
31
32 const updateUserById = async (newName, newPassword, id) =>{
33   knex('gebruikers')
34     .where('id', '=', id)
35     .update({
36       naam: `${newName}`,
37       wachtwoord: `${newPassword}`,
38       id: `${id}`
39     });
40 }
41
42 const deleteUserById = async (id) =>{
43   knex('gebruikers')
44     .where('id', id)
45     .del();
46 }
47
48 const createUser = async ({naam, wachtwoord, id}) =>{
49   await knex('gebruikers')
50     .insert({
51       naam,
52       wachtwoord,
53       id
54     });
55 }
56
57 const getAllUserNames = async () =>{
58   const [userNames] = await knex.raw('SELECT naam FROM gebruikers');
59   console.log(userNames);
60   return userNames
61 }
62
63 module.exports = {
```

queries.js - project - Visual Studio Code

```
src > repository > JS queries.js > ...
48 const createUser = async ({naam, wachtwoord, id}) =>{
49   await knex('gebruikers')
50     .insert({
51       naam,
52       wachtwoord,
53       id
54     });
55 }
56
57 const getAllUserNames = async () =>{
58   const [userNames] = await knex.raw('SELECT naam FROM gebruikers');
59   console.log(userNames);
60   return userNames
61 }
62
63 module.exports = {
64   getAllLiedjes,
65   getLiedjeByTitle,
66   getAllUsers,
67   getUserById,
68   updateUserById,
69   deleteUserById,
70   createUser,
71   getAllUserNames,
72 }
73
```