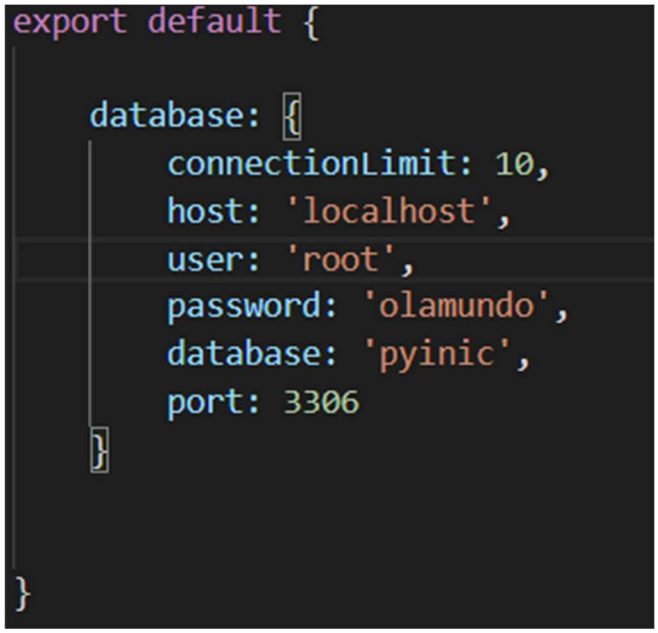
Manual Tecnico

A continuación se describe de forma mas clara posible la descripción del servidor nodejs el cual es el encargado de conectarse a la base de datos, hacer consultas, y responder al cliente cuando haga las peticiones correspondientes.

Se utilizo la V14.11.0 de Node JS

```
PS C:\Users\ASUS\Documents\2S_2020\PracticasIniciales\Lab\projecti\client> node -v
v14.11.0
PS C:\Users\ASUS\Documents\2S_2020\PracticasIniciales\Lab\projecti\client>
```

Credenciales para el uso de Mysql de forma local



- Se limita a un límite de 10
- Se especifica el host, como es local es local host
- Se define el usuario root
- Se define la contraseña
- Se define es esquema o base de datos a utiliza
- Se define el puerto al cual se va a conectar

Se conecta a la base de datos y se exporta el pool para que se puedan realizar las consultas en el controller

erver > src > TS database.ts > ...

```
1   import mysql from 'promise-mysql';
2   import keys from './keys';
3
4   const pool = mysql.createPool(keys.database);
5
6   pool.getConnection()
7       .then(connection => {
8           pool.releaseConnection(connection);
9           console.log('DB is Connected');
10      });
11
12  export default pool;
13
```

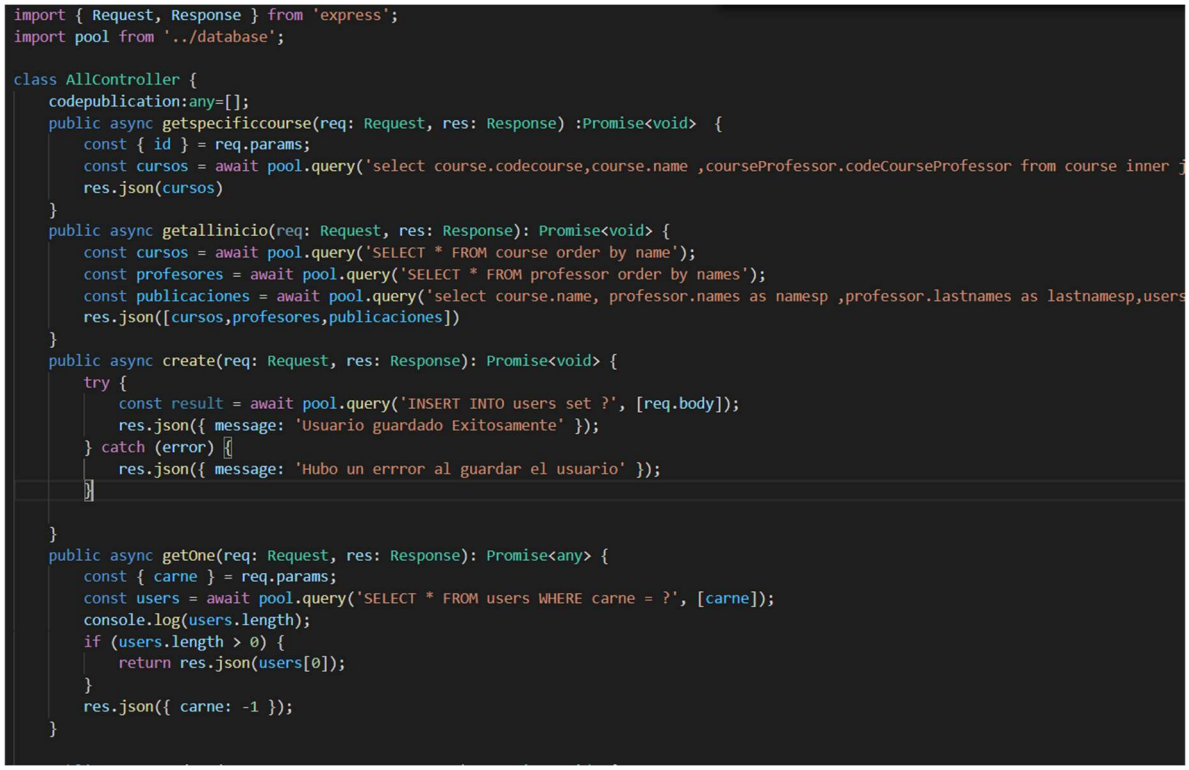
Se levanta servidor nodejs que estará a la escucha en el puerto 3000 para la peticiones del cliente y se configura el router general.

```
import express, { Application } from 'express';
import morgan from 'morgan';
import cors from 'cors';
import allRoutes from './routes/allRoutes';

class Server {

    public app: Application;

    constructor() {
        this.app = express();
        this.config();
        this.routes();
    }

    config(): void {
        this.app.set('port', process.env.PORT || 3000);

        this.app.use(morgan('dev'));
        this.app.use(cors());
        this.app.use(express.json());
        this.app.use(express.urlencoded({extended: false}));
    }

    routes(): void {
        this.app.use('/api', allRoutes);
    }

    start() {
        this.app.listen(this.app.get('port'), () => {
            console.log('Server on port', this.app.get('port'));
        });
    }

}
```

Se configura el router principal para hacer uso del controller dependiendo de la petición que realice el cliente, estas pueden ser get o post

```
import { timeStamp } from 'console';
import { Router } from 'express';

import allController from '../controllers/allController';

class AllRoutes {

    public router: Router = Router();

    constructor() {
        this.config();
    }

    config(): void {
        this.router.get('/:carne', allController.getOne);
        this.router.post('/register', allController.create);
        this.router.post('/update', allController.update);
        this.router.get('/getallinicio/all', allController.getallinicio);
        this.router.get('/getallinicio/all/:id', allController.getspecificcourse);
        this.router.post('/registerpublication', allController.createpublication);
        this.router.post('/publicacion/filtrar', allController.filtrar);
        this.router.post('/addcomentary', allController.createcomentariy);
        this.router.post('/updateuser', allController.updateuser);
        this.router.get('/publicacion/:id', allController.getonepublication);
        this.router.get('/pensum/all/all', allController.getcourses);


    }

}

const allRoutes = new AllRoutes();
export default allRoutes.router;
```

En el controller hace uso del pool para hacer las consultas, y responde con un json de los datos dependiendo de cada caso



```
import { Request, Response } from 'express';
import pool from '../database';

class AllController {
    codepublication:any=[];
    public async getspecificcourse(req: Request, res: Response) :Promise<void>  {
        const { id } = req.params;
        const cursos = await pool.query('select course.codecourse,course.name ,courseProfessor.codeCourseProfessor from course inner j
        res.json(cursos)
    }
    public async getallinicio(req: Request, res: Response): Promise<void> {
        const cursos = await pool.query('SELECT * FROM course order by name');
        const profesores = await pool.query('SELECT * FROM professor order by names');
        const publicaciones = await pool.query('select course.name, professor.names as namesp ,professor.lastnames as lastnamesp,users
        res.json([cursos,profesores,publicaciones])
    }
    public async create(req: Request, res: Response): Promise<void> {
        try {
            const result = await pool.query('INSERT INTO users set ?', [req.body]);
            res.json({ message: 'Usuario guardado Exitosamente' });
        } catch (error) {
            res.json({ message: 'Hubo un errror al guardar el usuario' });
        }

    }
    public async getOne(req: Request, res: Response): Promise<any> {
        const { carne } = req.params;
        const users = await pool.query('SELECT * FROM users WHERE carne = ?', [carne]);
        console.log(users.length);
        if (users.length > 0) {
            return res.json(users[0]);
        }
        res.json({ carne: -1 });
    }
```