1.- Dentro del proyecto creas una carpeta llamada clase, dentro de ella creas un archivo llamado Server.ts

2.- Después de ello copias el código:

```
import express from 'express';
export default class Server{
public app: express.Application;
public port:number=3000;
constructor(){
this.app=express();
}
start(callback: Function){
this.app.listen(this.port, callback());
}
}
```

3.- Posteriormente te vas hacia el archivo index.ts y copias el siguiente código:

```
import Server from './clases/server';
const server = new Server();
 //levantar el express
    server.start(()=>{
    console.log(`Servidor corriendo en puerto ${server.port}`);});
```

4.-Instalas el types express con la siguiente instrucción:

npm install @types/expresss -–save-dev

5.- Dentro del cmd ejecutas la instrucción:

npm install –g sass

6. te diriges al buscador del cmd y ejecutas el powershell como administrador

7.- dentro del power Shell ejecutas la siguiente instrucción:

Set-ExecutionPolicy Unrestricted

(después le das S para que aceptar todas las condiciones)

8.- en el cmd ejecutas:

tsc –w (para compilar)

9.- te vas a la terminal del Visual Studio Code y ejecutas:

Nodemon dist

10.- te vas a index.ts y copias el siguiente código (importas la librería moongose y copias el código de conectar DB):

```
import mongoose  from 'mongoose';
import Server from './clases/server';
const server = new Server();
  //conectar BD
    mongoose.connect('mongodb://localhost:27017/fotosgram2',
    {useNewUrlParser: true, useUnifiedTopology: true, useCreateIndex: true},
 (err)=>{
            if(err) throw err;
            console.log('base de datos ONLINE2');
        }
    )
 //levantar el express
    server.start(()=>{
    console.log(`Servidor corriendo en puerto ${server.port}`);});
```

11.- te vas al cmd y ejecutas:

npm install @types/moongose –save-dev

(En el Visual Studio Guardas los archivos y verifica que el servidor se levante sin problemas)

12.-Ahora creas una nueva carpeta llamada routes y dentro de ella creas un archivo llamado usuarios.ts

13.- Dentro del archivo usuarios.ts copias el siguiente código:

```
import {Router,Request, Response} from 'express';
import { Usuario } from '../model/usuario.models';
const userRoutes= Router();
userRoutes.post('/create',(req:Request, res:Response)=>{
    const user={
        nombre:req.body.nombre,
        email: req.body.email,
        password:  req.body.password,
        avatar:req.body.avatar
    }
    Usuario.create(user).then(userBD=>{
    res.json({
    ok: true,
    user: userBD
    }
    );
    }).catch(err => {
        res.json({
```

```
        ok:false,
        err
        }
            );

    })
});
export default userRoutes;
```

14.- Ahora creas una nueva carpeta llamada model y dentro de ella creas un archivo llamado usuario.models.ts

15.- Dentro del archivo usuario.models.ts copias el siguiente código:

```
import {model, Schema } from 'mongoose';
const usuariosSchema= new Schema ({
nombre:{
type: String,
required: [true, 'El nombre es necesario']
},
avatar:{
    type: String,
    default: 'av-1.png'
},
email:{
    type: String,
    unique: true,
    required: [true, 'El correo es necesario']
},
password:{
    type: String,
    required: [true, 'La contraseña es necesaria']
}
});

export const Usuario=model('Usuario', usuariosSchema);
```

Nota: Si te muestra error en nombre, email y password, te diriges al cmd y ejecutas:

npm update

16.- te diriges hacia el archivo index.ts y copias el siguiente código:

```
import mongoose  from 'mongoose';
import Server from './clases/server';
import userRoutes from './routes/usuarios';
```

```
import bodyParser from 'body-parser';

const server = new Server();
//body-parser
server.app.use(bodyParser.urlencoded({extended:true}));
server.app.use(bodyParser.json());
    //rutas de app
    server.app.use('/user',userRoutes);
    //conectar BD
    mongoose.connect('mongodb://localhost:27017/fotosgram2',
    {useNewUrlParser: true, useUnifiedTopology: true, useCreateIndex: true},
 (err)=>{
            if(err) throw err;
            console.log('base de datos ONLINE2');
        }
    )
    //levantar el express
    server.start(()=>{
    console.log(`Servidor corriendo en puerto ${server.port}`);});
```

17.- En el cmd ejecutas:

npm install @types/body-parser --save-dev

18.- Guardas y verificas que no haya error

19.- te diriges a usuarios.ts y dentro copias el siguiente código:

```
import {Router,Request, Response} from 'express';
import { Usuario } from '../model/usuario.models';
import bcrypt from 'bcrypt';

const userRoutes= Router();
userRoutes.post('/create',(req:Request, res:Response)=>{
    const user={
        nombre:req.body.nombre,
        email: req.body.email,
        password:  bcrypt.hashSync(req.body.password,10),
        avatar:req.body.avatar
    }
    Usuario.create(user).then(userBD=>{
    res.json({
    ok: true,
    user: userBD
    }
    );
    }).catch(err => {
```

```
        res.json({
        ok:false,
        err
        }
            );

    })
});
export default userRoutes;
```

20.- En el cmd ejecutas:

npm i --save-dev @types/bcrypt

21.- Guardas y haces la prueba en postman

22.- Dentro del archivo usuarios.ts copias el siguiente código:

```
import {Router,Request, Response} from 'express';
import { Usuario } from '../model/usuario.models';
import bcrypt from 'bcrypt';
const userRoutes= Router();
userRoutes.post('/login',(req:Request, res:Response)=>{
    const body= req.body;
   Usuario.findOne({email: body.email},(err, userBD)=>{
        if (err) throw err;
       if(!userBD){
       return res.json({
            ok: false,
            mensaje: 'Usuario/contraseña no son correctos'
            }
            );

       } if (userBD.compararPassword(body.password)){
        return res.json({
            ok: true,
            token: 'wqyuyqwuwyuqwywuq'
            }
            );
       } else{
        res.json({
            ok:false,
            mensaje: 'Usuario/contraseña no son correctos****'
        });
       }
    })
})
```

```
userRoutes.post('/create',(req:Request, res:Response)=>{
    const user={
        nombre:req.body.nombre,
        email: req.body.email,
        password: bcrypt.hashSync(req.body.password,10),
        avatar:req.body.avatar
    }
    Usuario.create(user).then(userBD=>{
    res.json({
    ok: true,
    user: userBD
    }
    );
    }).catch(err => {
        res.json({
        ok:false,
        err
        }
            );

    })
});
export default userRoutes;
```

23.- te diriges al archivo usuario.models.ts y copias el siguiente código:

```
import {model, Schema, Document} from 'mongoose';
import bcrypt from 'bcrypt';
const usuariosSchema= new Schema ({
nombre:{
type: String,
required: [true, 'El nombre es necesario']
},
avatar:{
    type: String,
    default: 'av-1.png'
},
email:{
    type: String,
    unique: true,
    required: [true, 'El correo es necesario']
},
password:{
    type: String,
```

```
    required: [true, 'La contraseña es necesaria']
}
});
usuariosSchema.method('compararPassword',function(password:string=''):
boolean {
     if (bcrypt.compareSync(password,this.password)){
     return true;
     }else{
     return false;
     }
    });
    interface IUsuario extends Document{
     nombre:string;
     avatar:string;
     email:string;
     password:string;
 compararPassword(password:string):boolean;
}

export const Usuario=model<IUsuario>('Usuario', usuariosSchema);
```

24.- Por ultimo te diriges hacia el archivo **tsconfig.json y ya dentro de él, buscas la línea número 35 y copias el siguiente código:**

```
  "noImplicitThis": false, /* Raise error on 'this' expressions with an impl
ied 'any' type. */
```

25.- Guardas todo y haces las pruebas de login.