

Universidad de San Carlos de Guatemala  
Facultad de Ingeniería  
Escuela de Ciencias y Sistemas  
Manejo e Implementación de Archivos  
Sección A

Manual Técnico  
Proyecto 2

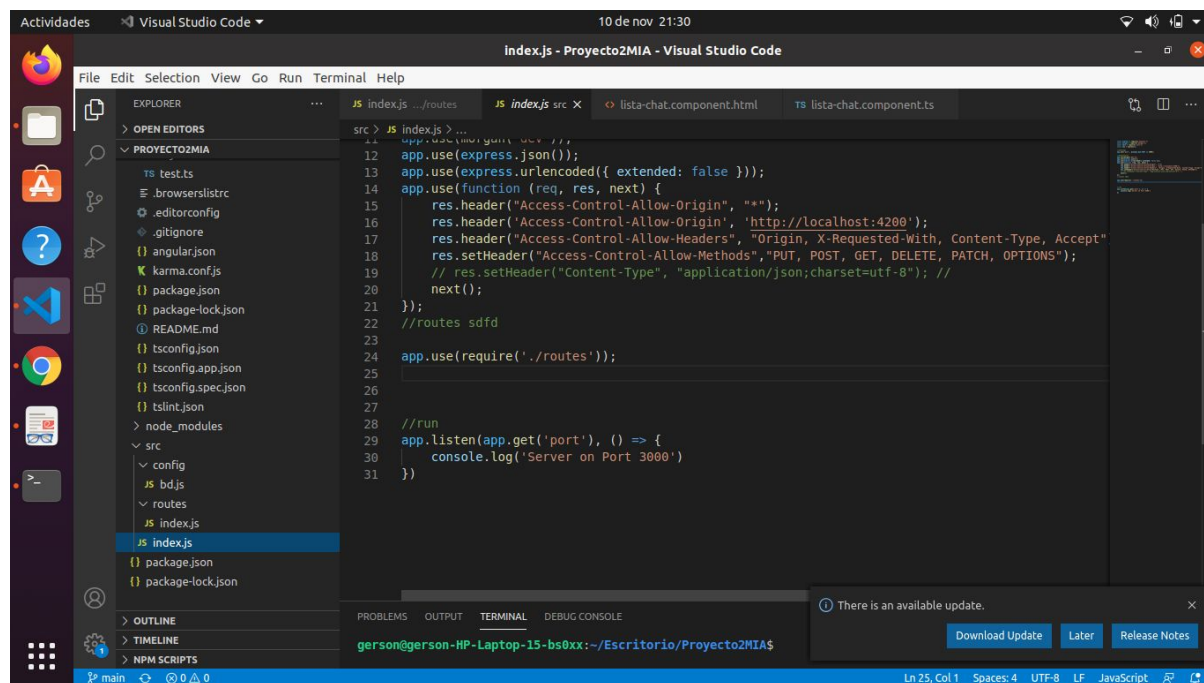
Gerson Alejandro Belteon Urbina  
201807228  
10/11/2020



Este proyecto consiste en una aplicación web desarrollada utilizando el framework Angular, que consume una base de datos Oracle a través de un servidor NodeJS utilizando métodos http.

## Backend

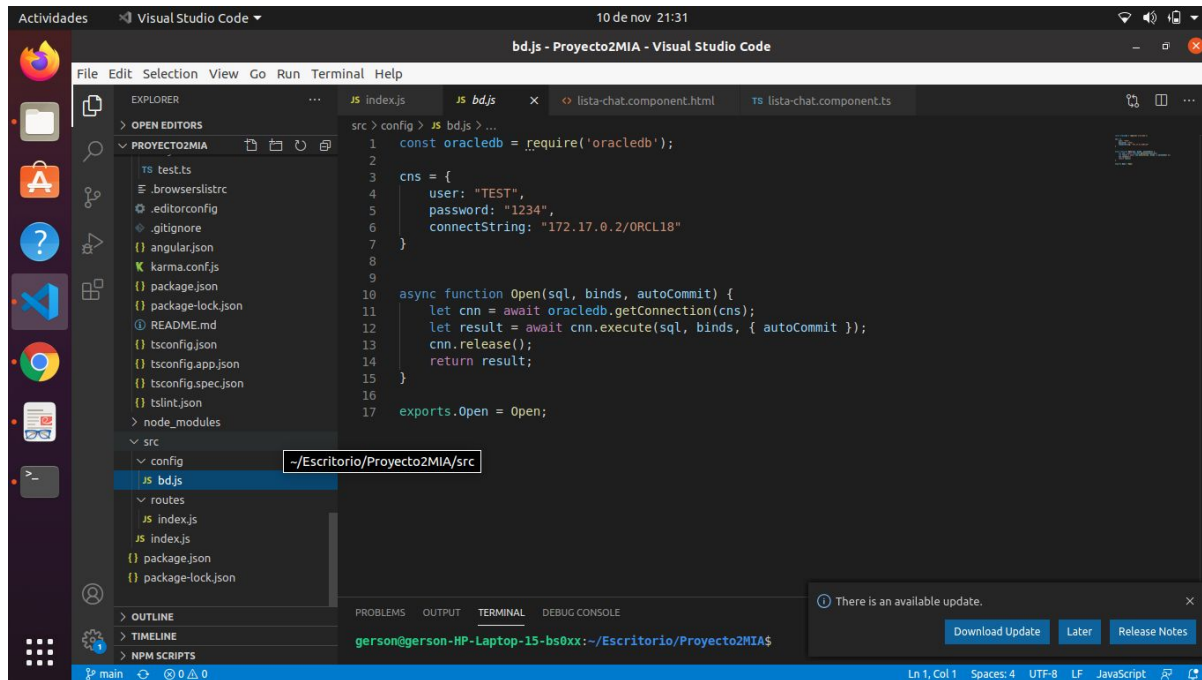
Este servidor se encarga de realizar las consultas a la base de datos y recibir peticiones http del frontend para posteriormente enviar la información.



```
11 app.use(morgan('dev'));
12 app.use(express.json());
13 app.use(express.urlencoded({ extended: false }));
14 app.use(function (req, res, next) {
15   res.header("Access-Control-Allow-Origin", "*");
16   res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
17   res.setHeader("Access-Control-Allow-Methods", "PUT, POST, GET, DELETE, PATCH, OPTIONS");
18   // res.setHeader("Content-Type", "application/json;charset=utf-8"); //
19   next();
20 });
21 //routes sdfd
22
23 app.use(require('./routes'));
24
25
26
27
28 //run
29 app.listen(app.get('port'), () => {
30   console.log('Server on Port 3000')
31 })
```

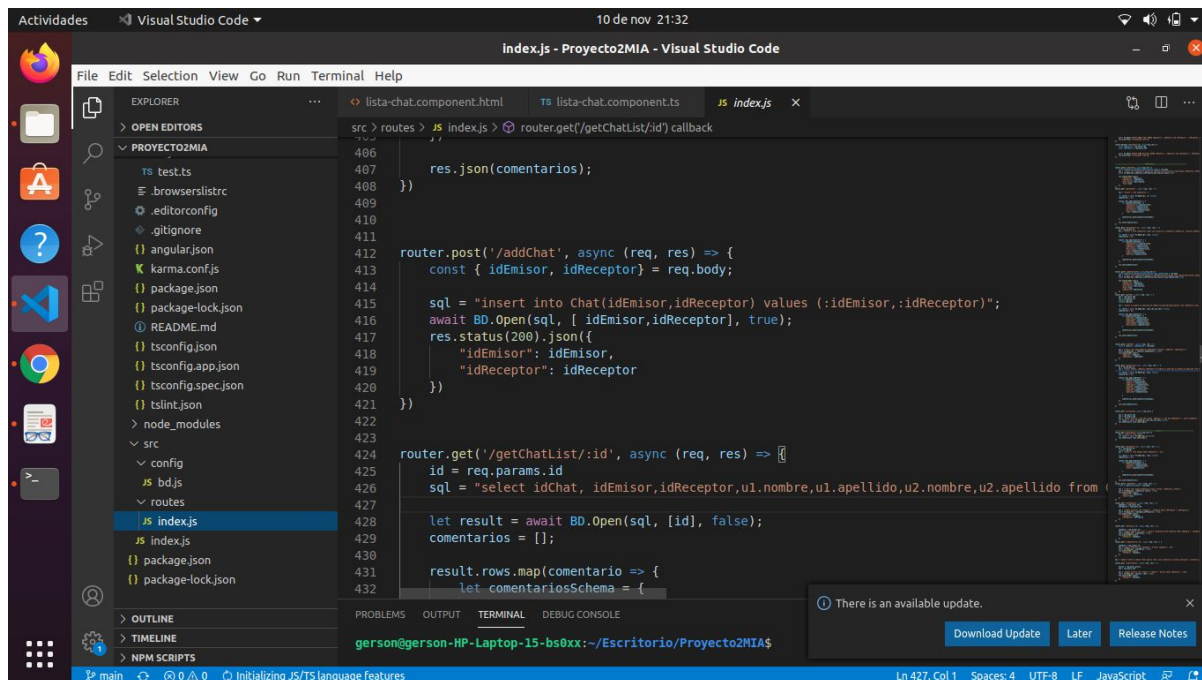
De esta manera se levanta el servidor





```
1 const oracledb = require('oracledb');
2
3 cns = {
4   user: "TEST",
5   password: "1234",
6   connectString: "172.17.0.2/ORCL18"
7 }
8
9
10 async function Open(sql, binds, autoCommit) {
11   let cnn = await oracledb.getConnection(cns);
12   let result = await cnn.execute(sql, binds, { autoCommit });
13   cnn.release();
14   return result;
15 }
16
17 exports.Open = Open;
```

Conexión con la base de datos.

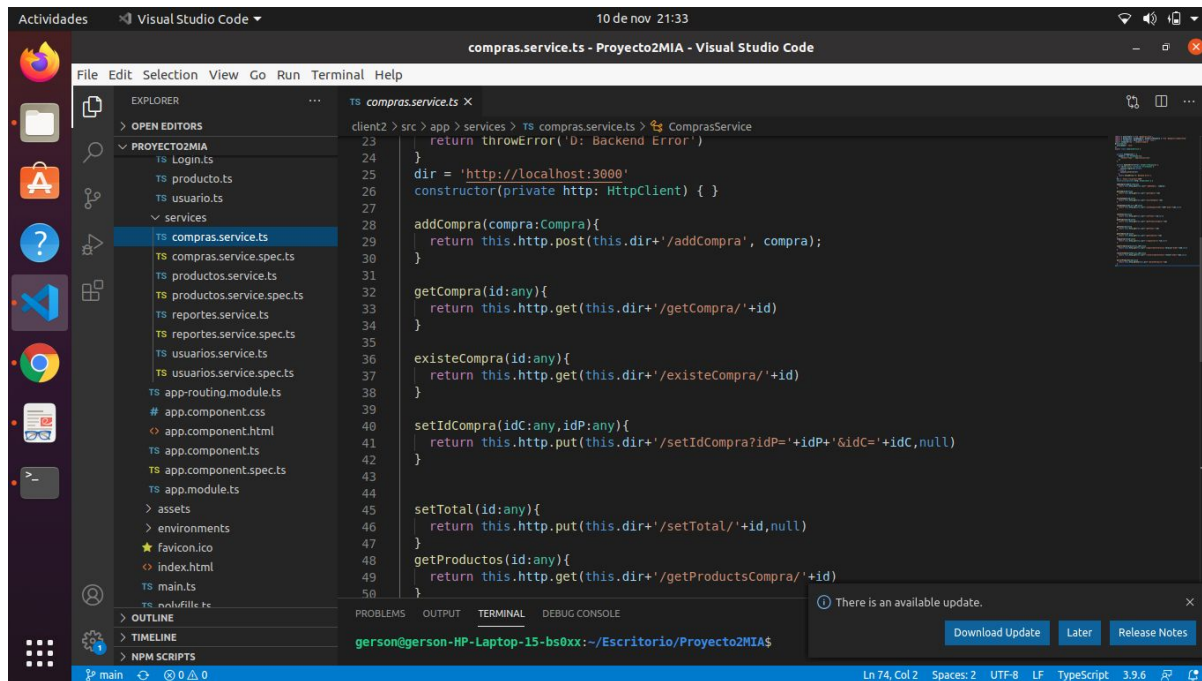


```
406 res.json(comentarios);
407
408 })
409
410
411 router.post('/addChat', async (req, res) => {
412   const { idEmisor, idReceptor } = req.body;
413
414   sql = "insert into Chat(idEmisor,idReceptor) values (:idEmisor,:idReceptor)";
415   await BD.Open(sql, [ idEmisor,idReceptor], true);
416   res.status(200).json({
417     "idEmisor": idEmisor,
418     "idReceptor": idReceptor
419   })
420 })
421
422
423 router.get('/getChatList/:id', async (req, res) => {
424   id = req.params.id
425   sql = "select idChat, idEmisor,idReceptor,u1.nombre,u1.apellido,u2.nombre,u2.apellido from ";
426
427   let result = await BD.Open(sql, [id], false);
428   comentarios = [];
429
430   result.rows.map(comentario => {
431     let comentariosSchema = {
432
```

De esta manera se realizan las consultas a la base de datos oracle desde nodejs.



Frontend:

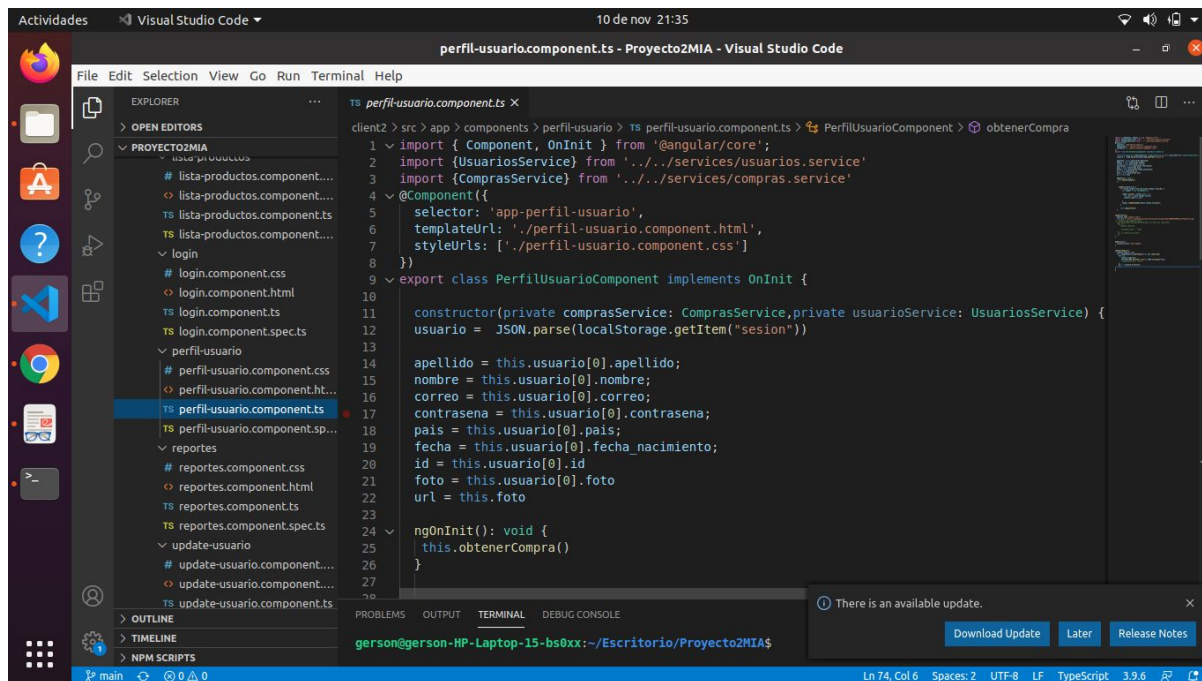


The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying the project structure. The main editor window shows the `compras.service.ts` file. The code defines a `ComprasService` class that interacts with a REST API. The terminal at the bottom shows the user's shell prompt.

```
compras.service.ts - Proyecto2MIA - Visual Studio Code

client2 > src > app > services > TS compras.service.ts > ComprasService
23   return throwError('D: Backend Error')
24 }
25 dir = 'http://localhost:3000'
26 constructor(private http: HttpClient) { }
27
28 addCompra(compra: Compra){
29   return this.http.post(this.dir+'/addCompra', compra);
30 }
31
32 getCompra(id:any){
33   return this.http.get(this.dir+'/getCompra/'+id)
34 }
35
36 existeCompra(id:any){
37   return this.http.get(this.dir+'/existeCompra/'+id)
38 }
39
40 setIdCompra(idC:any,idP:any){
41   return this.http.put(this.dir+'/setIdCompra?idP='+idP+'&idC='+idC,null)
42 }
43
44
45 setTotal(id:any){
46   return this.http.put(this.dir+'/setTotal/'+id,null)
47 }
48
49 getProductos(id:any){
50   return this.http.get(this.dir+'/getProductosCompra/'+id)
51 }
```

De esta manera se envían peticiones http al servidor de nodejs, a través de lo que se conoce en Angular como servicio.



The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying the project structure. The main editor window shows the `perfil-usuario.component.ts` file. The code defines a `PerfilUsuarioComponent` class that implements the `OnInit` interface. The terminal at the bottom shows the user's shell prompt.

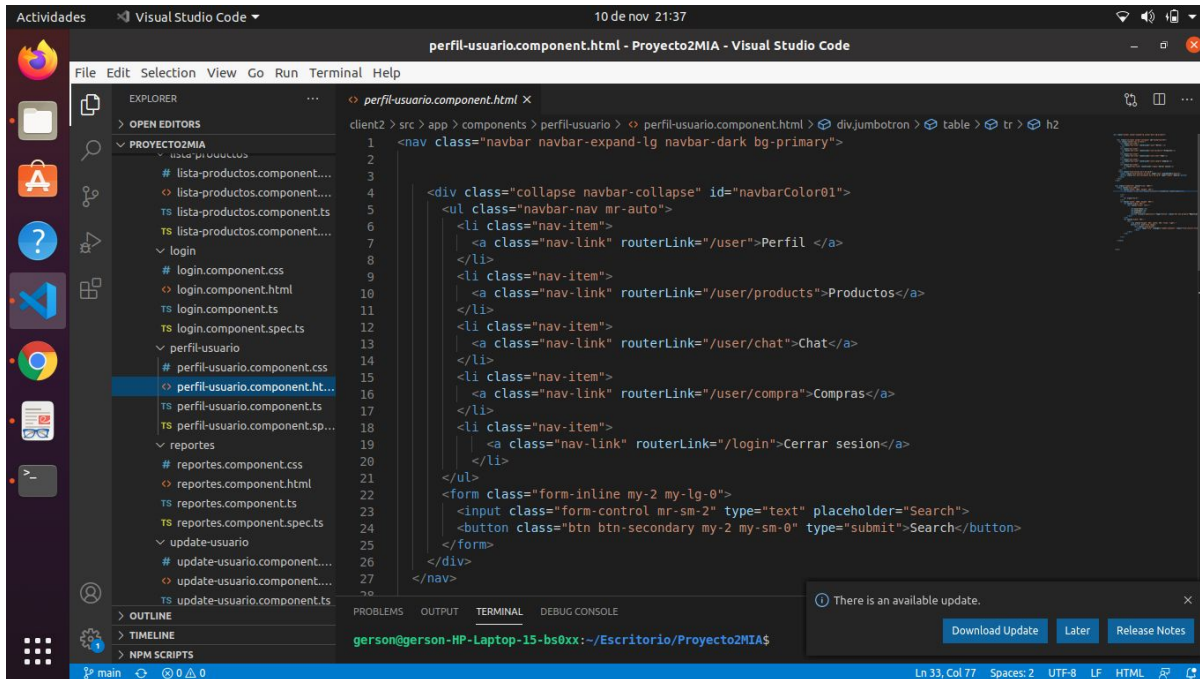
```
perfil-usuario.component.ts - Proyecto2MIA - Visual Studio Code

client2 > src > app > components > perfil-usuario > TS perfil-usuario.component.ts > PerfilUsuarioComponent > obtenerCompra
1  import { Component, OnInit } from '@angular/core';
2  import { UsuariosService } from '../services/usuarios.service';
3  import { ComprasService } from '../services/compras.service';
4  @Component({
5    selector: 'app-perfil-usuario',
6    templateUrl: './perfil-usuario.component.html',
7    styleUrls: ['./perfil-usuario.component.css']
8  })
9  export class PerfilUsuarioComponent implements OnInit {
10
11    constructor(private comprasService: ComprasService, private usuarioService: UsuariosService) {
12      usuario = JSON.parse(localStorage.getItem("sesion"))
13    }
14
15    apellido = this.usuario[0].apellido;
16    nombre = this.usuario[0].nombre;
17    correo = this.usuario[0].correo;
18    contrasena = this.usuario[0].contrasena;
19    pais = this.usuario[0].pais;
20    fecha = this.usuario[0].fecha_nacimiento;
21    id = this.usuario[0].id
22    foto = this.usuario[0].foto
23    url = this.foto
24
25    ngOnInit(): void {
26      this.obtenerCompra()
27    }
28 }
```





Angular trabaja con componentes que se generan automáticamente utilizando el comando `ng g component` y se generan 3 archivos, un ts como el de la imagen anterior, un html y css para la parte de la vista.



The screenshot shows the Visual Studio Code editor with the file `perfil-usuario.component.html` open. The Explorer sidebar on the left shows the project structure for 'Proyecto2MIA', including files for 'lista-productos', 'login', 'perfil-usuario', 'reportes', and 'update-usuario'. The main editor area displays the HTML template for the 'perfil-usuario' component. The code includes a navigation bar with links for 'Perfil', 'Productos', 'Chat', 'Compras', and 'Cerrar sesion'. It also features a search form with a text input and a submit button. The status bar at the bottom indicates the current position is Line 33, Column 77, with 2 spaces, UTF-8 encoding, and LF line endings.

```
1 <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
2
3
4 <div class="collapse navbar-collapse" id="navbarColor01">
5   <ul class="navbar-nav mr-auto">
6     <li class="nav-item">
7       <a class="nav-link" routerLink="/user">Perfil </a>
8     </li>
9     <li class="nav-item">
10      <a class="nav-link" routerLink="/user/products">Productos</a>
11    </li>
12    <li class="nav-item">
13      <a class="nav-link" routerLink="/user/chat">Chat</a>
14    </li>
15    <li class="nav-item">
16      <a class="nav-link" routerLink="/user/compra">Compras</a>
17    </li>
18    <li class="nav-item">
19      <a class="nav-link" routerLink="/login">Cerrar sesion</a>
20    </li>
21  </ul>
22  <form class="form-inline my-2 my-lg-0">
23    <input class="form-control mr-sm-2" type="text" placeholder="Search">
24    <button class="btn btn-secondary my-2 my-sm-0" type="submit">Search</button>
25  </form>
26 </div>
27 </nav>
```

Este es el archivo html de un componente, desde el se envía información al archivo ts o se recibe información de él para mostrar datos almacenados en arreglos o variables a través de objetos html.

Diagrama Entidad Relación:



