



**WebSocket, protocolo de internet de comunicación dúplex entre un servidor y cliente.**

**<https://www.npmjs.com/package/socket.io>**

**mkdir socket-server-nodejs**

**cd socket-server-nodejs**

**npm init -- yes**

**mkdir src**

**npm i express socket.io @types/socket.io**

/src/package.json

```
{
  "name": "socket-server-nodejs",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start" : "node src/.",
    "dev" : "nodemon src/."
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "@types/socket.io": "^2.1.2",
    "express": "^4.17.1",
    "socket.io": "^2.2.0"
  }
}
```

/src/index.js

```
const app = require("express")();
const http = require("http").Server(app);
const io = require("socket.io")(http);

const notas = {};
const messages = [];

io.on("connection", socket => {
  let anteriorId;
  const unirNota = actualId => {
    socket.leave(anteriorId);
    socket.join(actualId, () =>
      console.log(`Socket ${socket.id} se ha unido a la nota: ${actualId}`)
    );
    anteriorId = actualId;
  };

  socket.on("getDoc", docId => {
    unirNota(docId);
    socket.emit("nota", notas[docId]);
  });

  socket.on("addDoc", doc => {
    notas[doc.id] = doc;
    unirNota(doc.id);
    io.emit("notas", Object.keys(notas));
    socket.emit("nota", doc);
  });

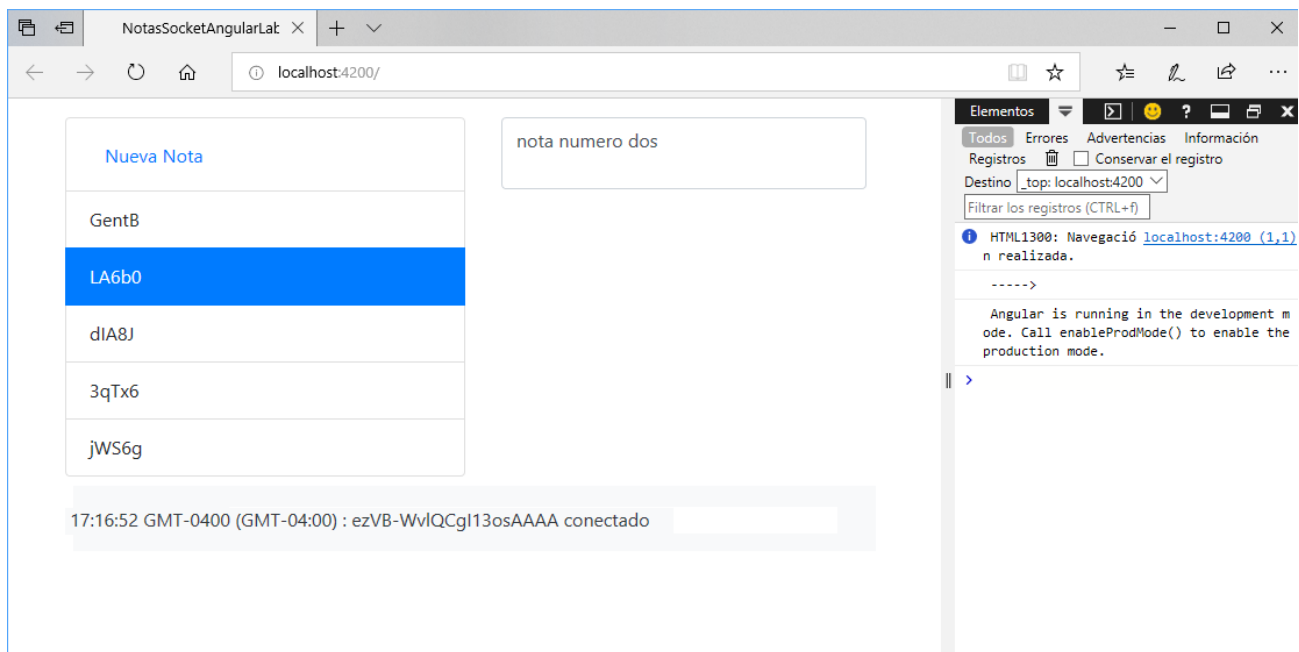
  socket.on("editDoc", doc => {
    notas[doc.id] = doc;
    socket.to(doc.id).emit("nota", doc);
  });

  socket.on("message", msg => {
    const resp = `${new Date()} : ${msg}`;
    messages.push(resp);
  });
});
```

```
console.log(`Socket ID: ${socket.id} conectado`);
io.emit("notas", Object.keys(notas));
messages.push(`${new Date()} : ${socket.id} conectado`);
io.emit("message", messages);

socket.on("disconnect", () => {
  console.log(`Socket ID: ${socket.id} desconectado`);
  messages.push(`${new Date()} : ${socket.id} desconectado`);
  io.emit("message", messages);
});
});

http.listen(3000, () => {
  console.log("Servidor socket escuchando en el puerto 3000");
});
```



NotasSocketAngularLat x + v

localhost:4200/

Nueva Nota

GentB

LA6b0

dIA8J

3qTx6

jWS6g

nota numero dos

17:16:52 GMT-0400 (GMT-04:00) : ezVB-WvlQCgI13osAAAA conectado

Elementos

Todos Errores Advertencias Información

Registros ☐ Conservar el registro

Destino \_top: localhost:4200

Filtrar los registros (CTRL+f)

HTML1300: Navegación a localhost:4200 (1,1) no realizada.

----->

Angular is running in the development mode. Call enableProdMode() to enable the production mode.



**<https://www.npmjs.com/package/ngx-socket-io>**

**ng new socket-angular-lab**

**cd socket-angular-lab**

**npm i ngx-socket-io**

**ng generate class modelo/nota**

**ng generate component componentes/notas-lista**

**ng generate component componentes/nota**

**ng generate service servicio/nota**

**ng serve - o**

/src/app/app/module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { SocketIoModule, SocketIoConfig } from 'ngx-socket-io';

import { AppRoutingModuleModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { NotasListaComponent } from './componentes/notas-lista/notas-lista.component';
import { NotaComponent } from './componentes/nota/nota.component';
import { MonitorComponent } from './componentes/monitor/monitor.component';

const config: SocketIoConfig = { url: 'http://localhost:3000', options: {} };

@NgModule({
  declarations: [
    AppComponent,
    NotasListaComponent,
    NotaComponent,
    MonitorComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModuleModule,
    FormsModule,
    SocketIoModule.forRoot(config)
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```



/src/app/modelo/nota.ts

```
export class Nota {  
  id: string;  
  doc: string;  
}
```

/src/app/servicios/nota.service.ts

```
import { Injectable } from '@angular/core';
import { Socket } from 'ngx-socket-io';
import { Nota } from '../modelo/nota';

@Injectable({
  providedIn: 'root'
})
export class NotaService {
  notaActual = this.socket.fromEvent<Nota>('nota');
  notas = this.socket.fromEvent<string[]>('notas');

  constructor(private socket: Socket) { }

  obtenerNota(id: string) {
    this.socket.emit('getDoc', id);
  }

  nuevaNota() {
    this.socket.emit('addDoc', { id: this.docId(), doc: '' });
  }

  editarNota(nota: Nota) {
    this.socket.emit('editDoc', nota);
  }

  getMessage() {
    return this.socket.fromEvent('message');
  }

  sendMessage(msg: string) {
    this.socket.emit('message', msg);
  }
}
```





```
private docId() {  
  let text = '';  
  const possible = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789';  
  
  for (let i = 0; i < 5; i++) {  
    text += possible.charAt(Math.floor(Math.random() * possible.length));  
  }  
  
  return text;  
}
```

/src/app/componentes/notas-lista/notas-lista.component.ts

```
import { Component, OnInit, OnDestroy } from '@angular/core';
import { Observable, Subscription } from 'rxjs';
import { NotaService } from 'src/app/servicio/nota.service';

@Component({
  selector: 'app-notas-lista',
  templateUrl: './notas-lista.component.html',
  styleUrls: ['./notas-lista.component.css']
})
export class NotasListaComponent implements OnInit, OnDestroy {

  notas: Observable<string[]>;
  idActual: string;
  private _docSub: Subscription;

  constructor(private notaServicio: NotaService) { }

  ngOnInit() {
    this.notas = this.notaServicio.notas;
    this._docSub = this.notaServicio.notaActual.subscribe(resp => {
      this.idActual = resp.id;
    });
  }

  ngOnDestroy() {
    this._docSub.unsubscribe();
  }

  cargarNota(id: string) {
    this.notaServicio.obtenerNota(id);
  }

  nuevaNota() {
    this.notaServicio.nuevaNota();
  }
}
```



/src/app/componentes/notas-lista/notas-lista.component.html

```
<ul class="list-group">
  <li class="list-group-item">
    <button class="btn btn-link" (click)="nuevaNota()">Nueva Nota</button>
  </li>
  <li class="list-group-item"
    [ngClass]="{'active': docId === idActual}"
    (click)="cargarNota(docId)"
    *ngFor="let docId of notas | async">
    {{ docId }}
  </li>
</ul>
```

/src/app/componentes/notas/notas.component.ts

```
import { Component, OnInit, OnDestroy } from '@angular/core';
import { Subscription } from 'rxjs';
import { startWith } from 'rxjs/operators';
import { Nota } from './../../../../modelo/nota';
import { NotaService } from 'src/app/servicio/nota.service';

@Component({
  selector: 'app-nota',
  templateUrl: './nota.component.html',
  styleUrls: ['./nota.component.css']
})
export class NotaComponent implements OnInit, OnDestroy {

  nota: Nota;
  private _docSub: Subscription;

  constructor(private notaServicio: NotaService) { }

  ngOnInit() {
    this._docSub = this.notaServicio.notaActual.pipe(
      startWith({id: '', doc: 'Seleccione una nota existente o cree una nueva para empesar'})
    ).subscribe(resp => {
      this.nota = resp;
    });
  }

  ngOnDestroy() {
    this._docSub.unsubscribe();
  }

  editarNota() {
    this.notaServicio.editarNota(this.nota);
  }
}
```



/src/app/componentes/notas/notas.component.html

```
<div class="p-2">
  <textarea
    class="form-control"
    [(ngModel)]='nota.doc'
    (keyup)='editarNota()'
    placeholder='Empesar a escribir ...'>
  </textarea>
</div>
```

/src/app/componentes/monitor/monitor.component.ts

```
import { Component, OnInit, OnDestroy } from '@angular/core';
import { Observable, Subscription } from 'rxjs';
import { NotaService } from 'src/app/servicio/nota.service';

@Component({
  selector: 'app-monitor',
  templateUrl: './monitor.component.html',
  styleUrls: ['./monitor.component.css']
})
export class MonitorComponent implements OnInit, OnDestroy {

  private _docSub: Subscription;
  message: string[];

  constructor(private notaServicio: NotaService) { }

  ngOnInit() {
    this.notaServicio.getMessage().subscribe(resp => {
      this.message = resp as string[];
    });
  }

  ngOnDestroy() {
    this._docSub.unsubscribe();
  }
}
```



/src/app/componentes/monitor/monitor.component.html

```
<div class="p-3 mb-2 bg-light text-dark">
  <div *ngFor="let msg of message">
    {{msg}}
  </div>
</div>
```

/src/app/app.component.html

```
<div class="container">
  <div class="row mt-2">
    <div class="col-6 p-2">
      <app-notas-lista></app-notas-lista>
    </div>
    <div class="col-6">
      <app-nota></app-nota>
    </div>
  </div>
  <div class="row">
    <div class="col-12">
      <app-monitor></app-monitor>
    </div>
  </div>
</div>
```