

Trabajo de final de grado

Título: Chat online con cliente web

Alumno: Dylan Hurtado López



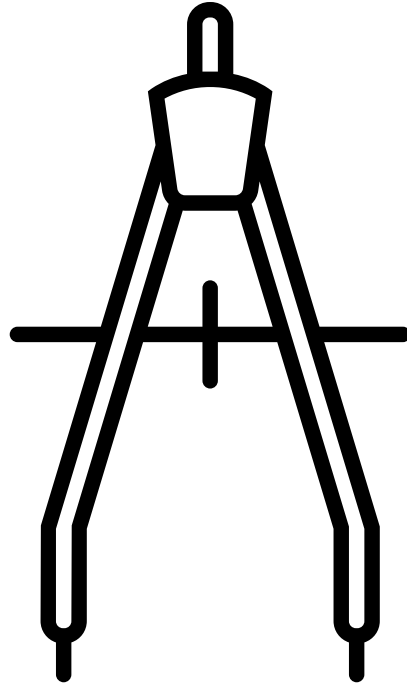
COSMOS

Objetivos



- Aprender nuevas tecnologías orientadas al desarrollo web.
- Desenvolverse en un territorio desconocido y salir victorioso.
- Asentar los conocimientos aprendidos para un cercano futuro laboral
- Tener un proyecto completo para que vean que me tomo mi trabajo en serio.
- Volverme un desarrollador full-stack.

Tecnologías más importantes a destacar

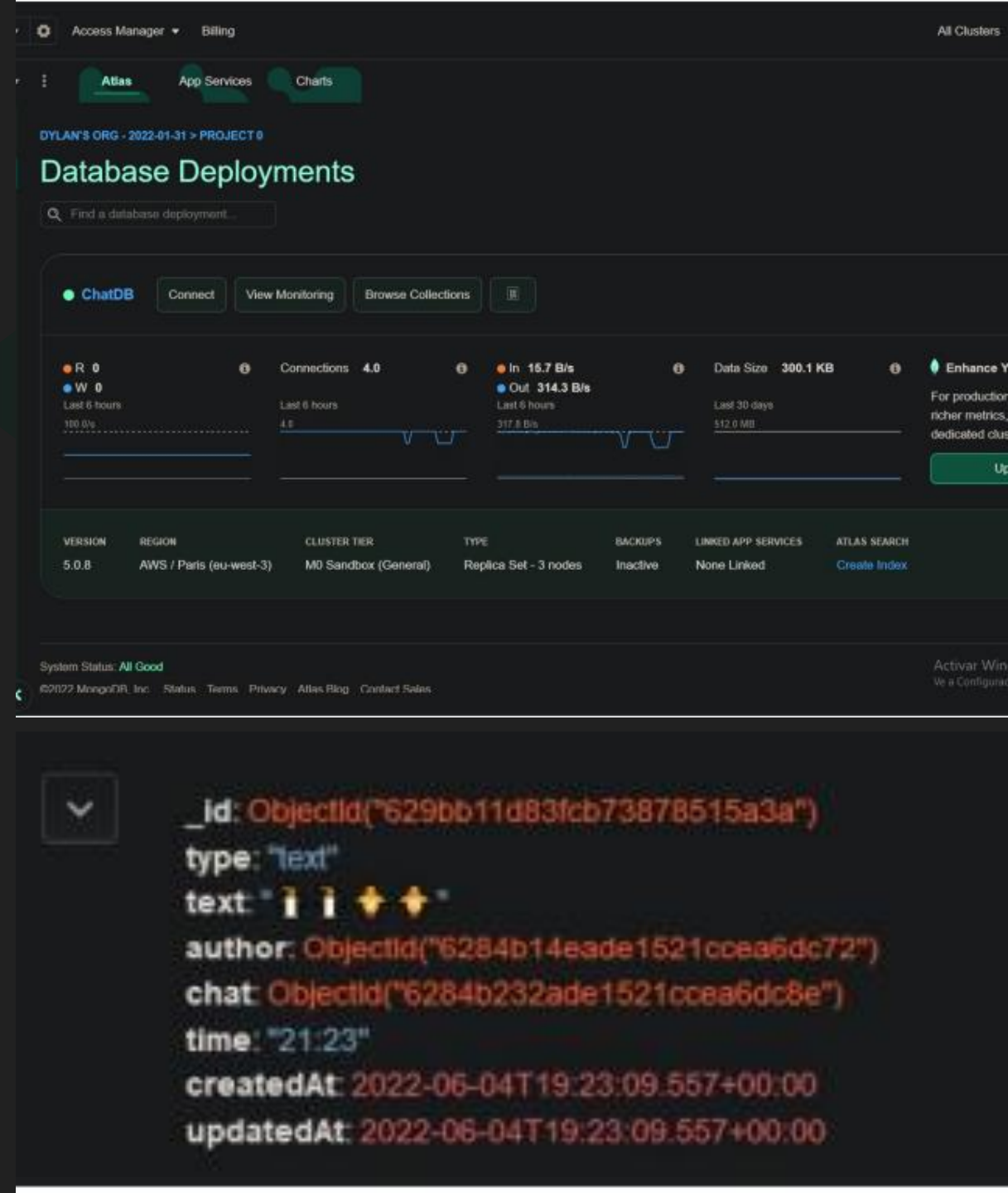


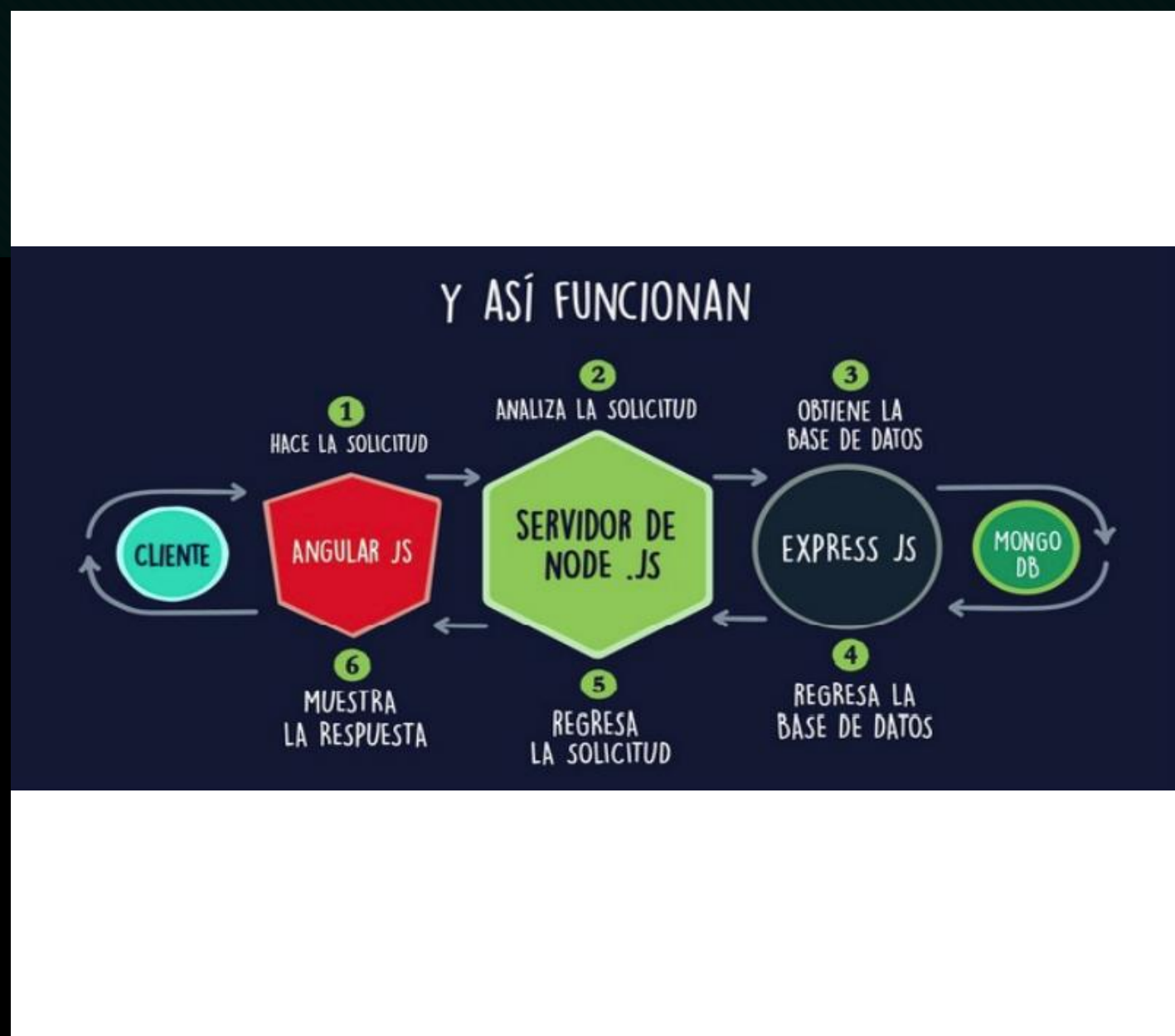
¿Por qué una base de datos no relacional?

- En el desarrollo web es muy popular.
- Quería romper con el modelo relacional
- Son mucho más flexibles a la hora de crear esquemas de información.
- Gestión de datos no estructurados o semiestructurados sin tantas restricciones
- Ofrecen una mayor escalabilidad.

MongoDB Atlas

- **MongoDB Atlas** es un servicio que te permite crear y administrar tu BBDD **MongoDB** desde cualquier lugar del mundo, a través de su plataforma.
- No hace falta configuraciones previas
- Ni establecer protocolos de seguridad.





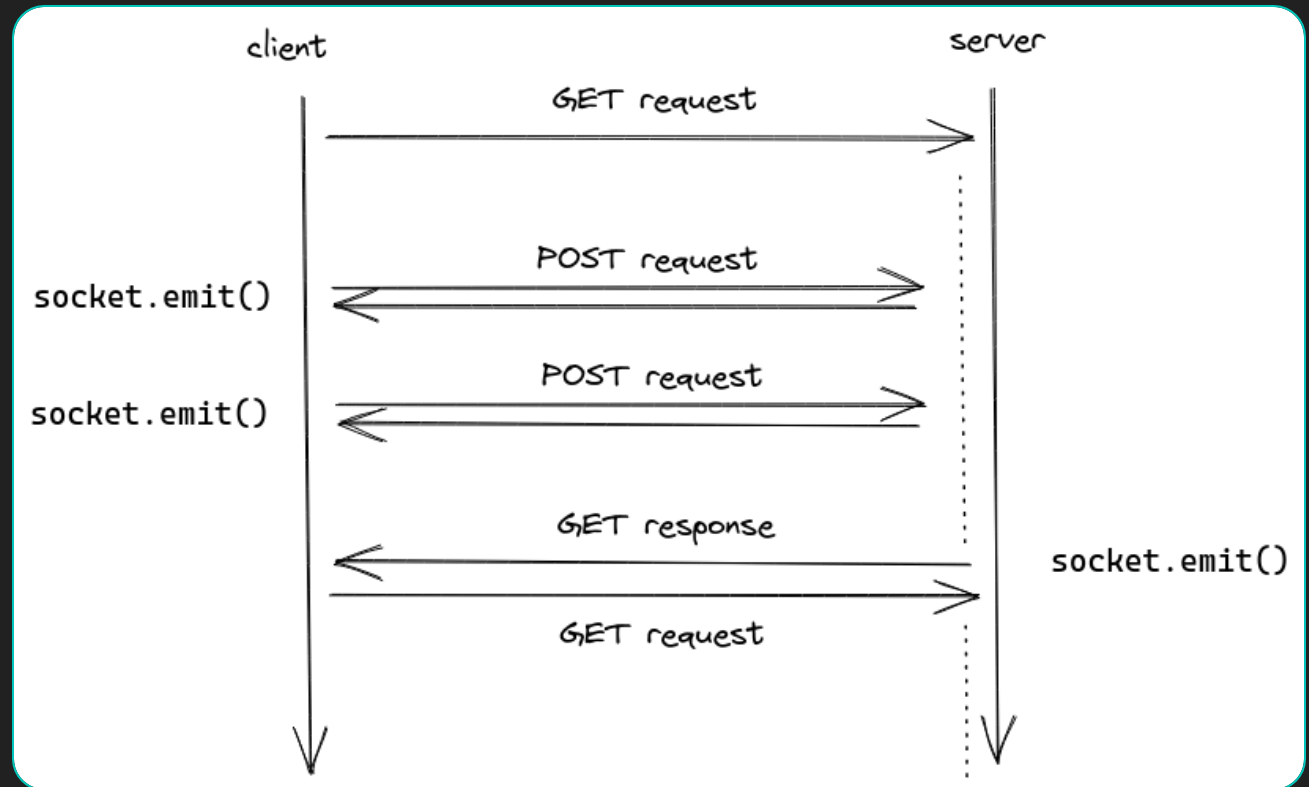
Testing con Jest

- Es un framework JavaScript de testing mantenido por Meta (Facebook).
- La mayor ventaja que tiene es que no necesitas de mucha configuración inicial.

```
test("getById works", async () => {  
  const messages = await Message.find({});  
  const id = messages[0]._id;  
  await api.get(process.env.API_MAINENDPOINT + `message/${id}`).expect(200);  
  
  const thisIdNotExist = "62488c3da0f2dc4c561e292a";  
  await api  
    .get(process.env.API_MAINENDPOINT + `message/${thisIdNotExist}`)  
    .expect(404);  
  
  await api  
    .get(process.env.API_MAINENDPOINT + `message/${1234}`)  
    .expect(400);  
});
```


Websockets y Socket.io

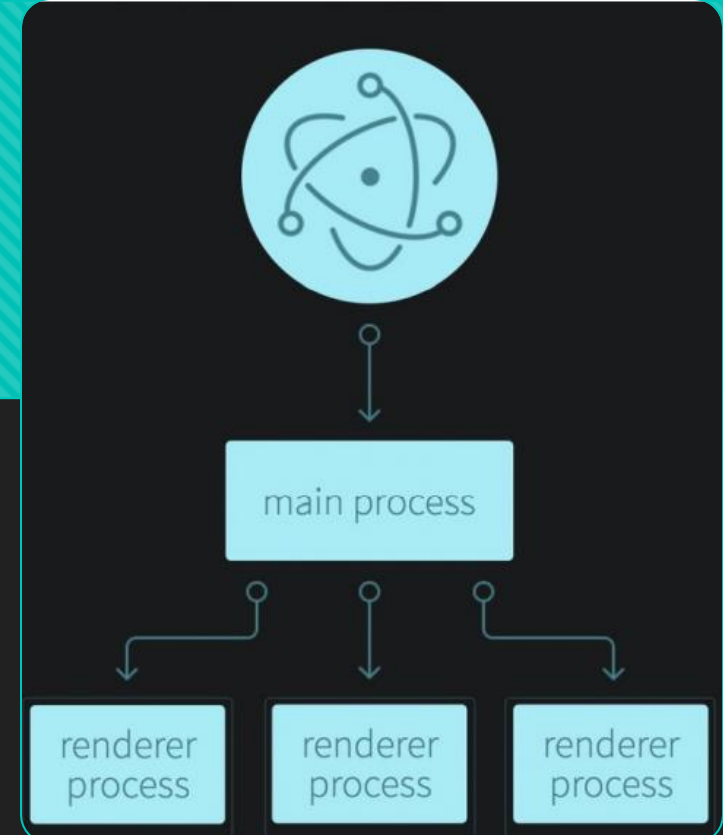
- WebSocket es una tecnología que proporciona un canal de comunicación bidireccional sobre un único socket TCP. Está diseñada para ser implementada en navegadores y servidores web, pero puede utilizarse por cualquier aplicación cliente/servidor.
- Socket.IO es una biblioteca de JavaScript basada en eventos para aplicaciones web en tiempo real. Permite la comunicación bidireccional en tiempo real entre clientes web y servidores.



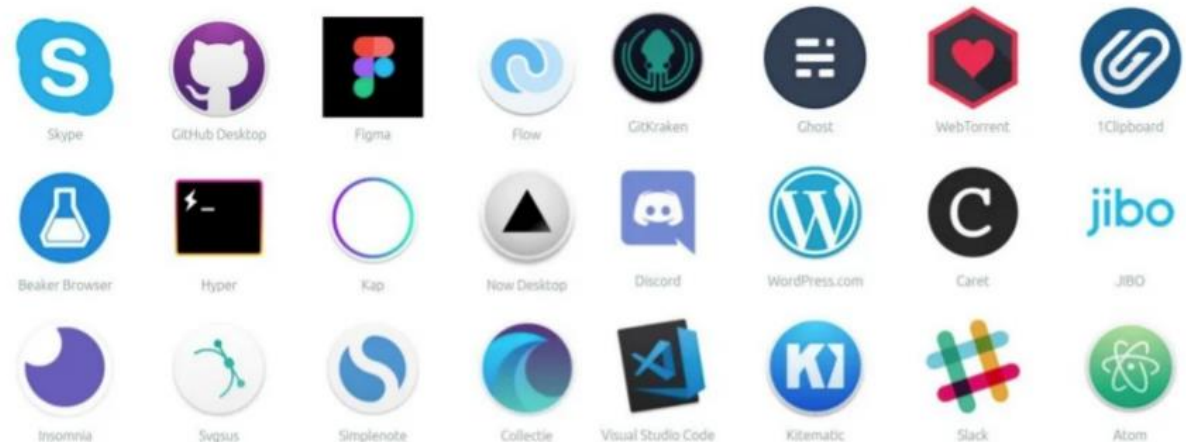
Electron

○ Es una plataforma para desarrollar aplicaciones de escritorio usando tecnologías web (HTML, CSS y JavaScript) creada y mantenida por Github.

○ Electron.js funciona creando dos tipos de procesos, el proceso main y el proceso renderer . El proceso main engloba a los demás procesos de tipo renderer agregar texto



Apps built on Electron



Fase de diseño

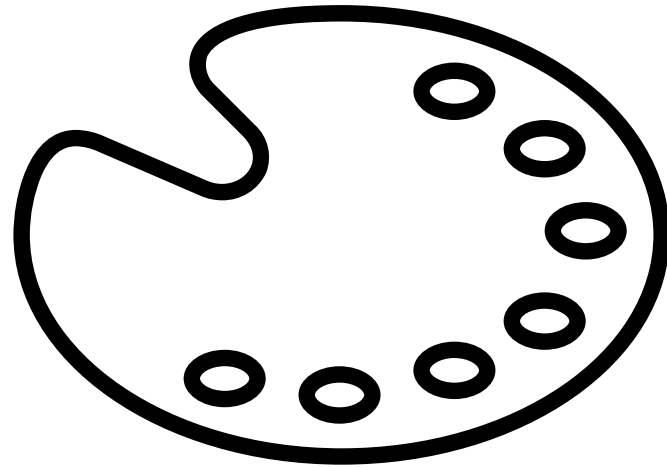
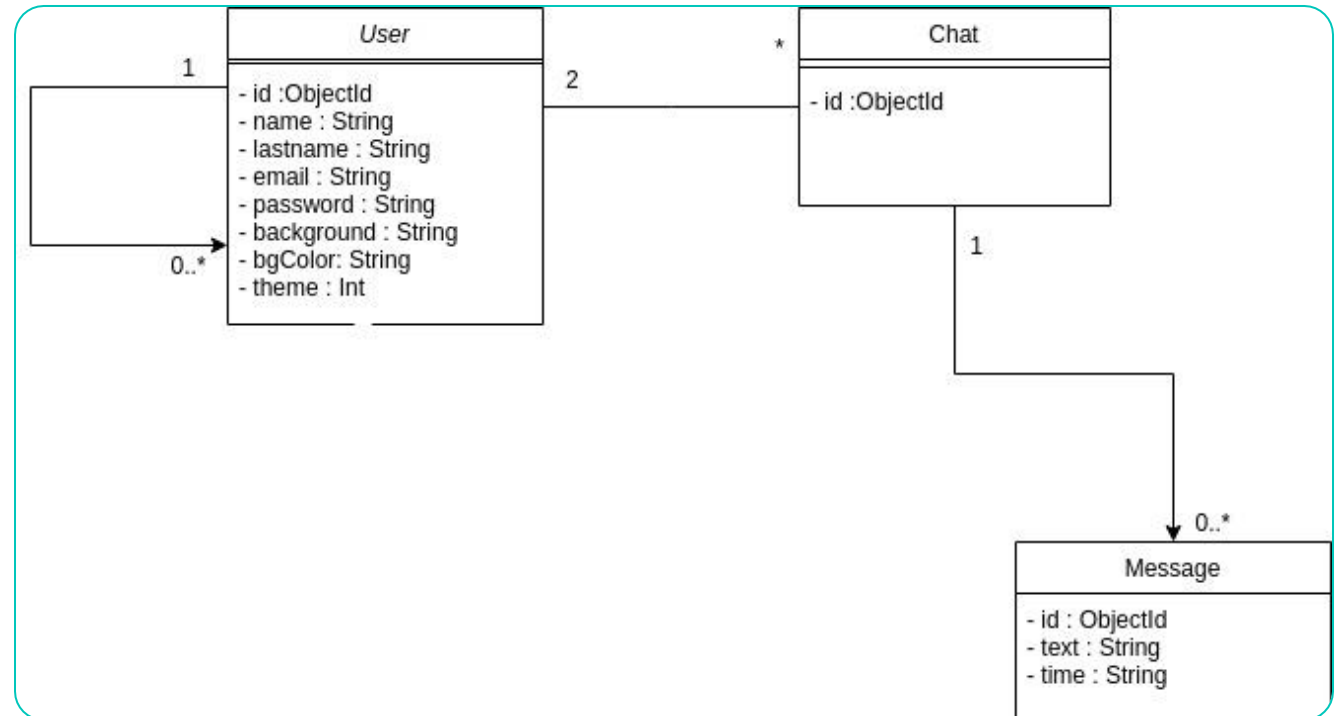
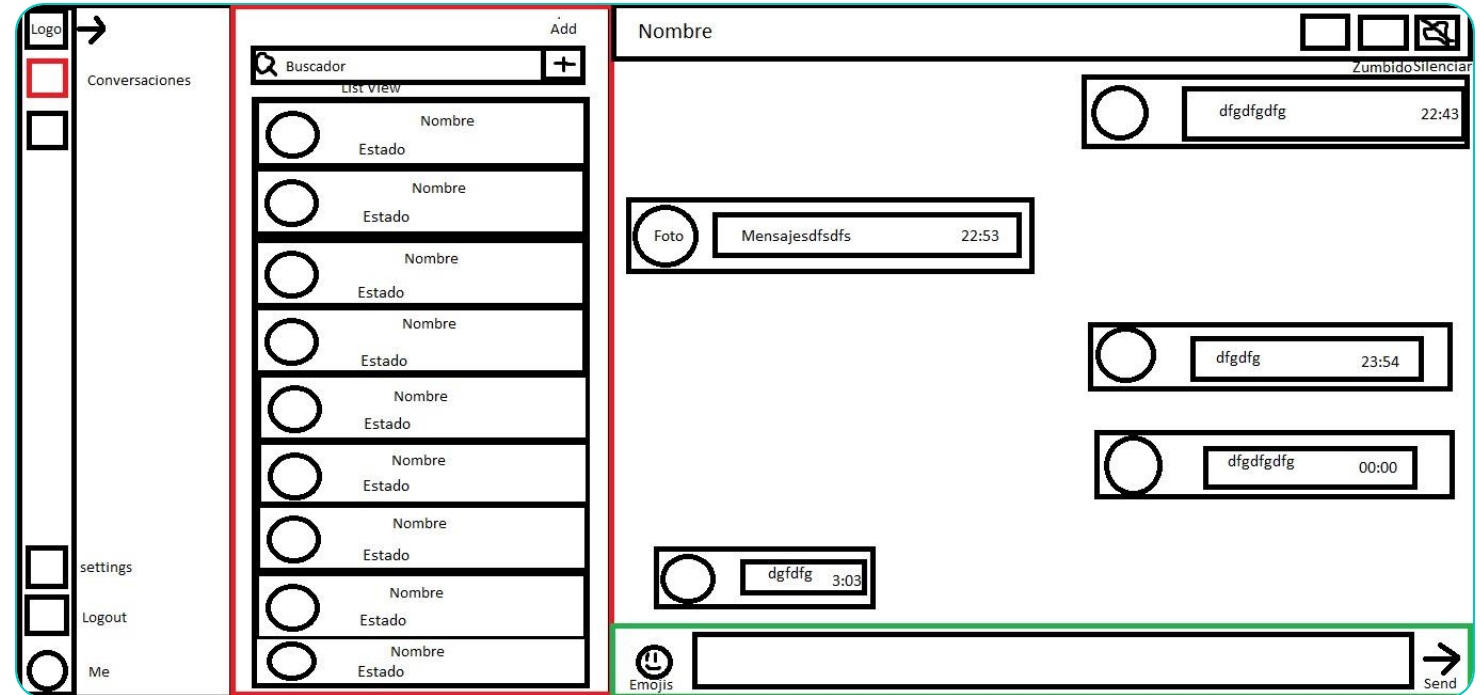


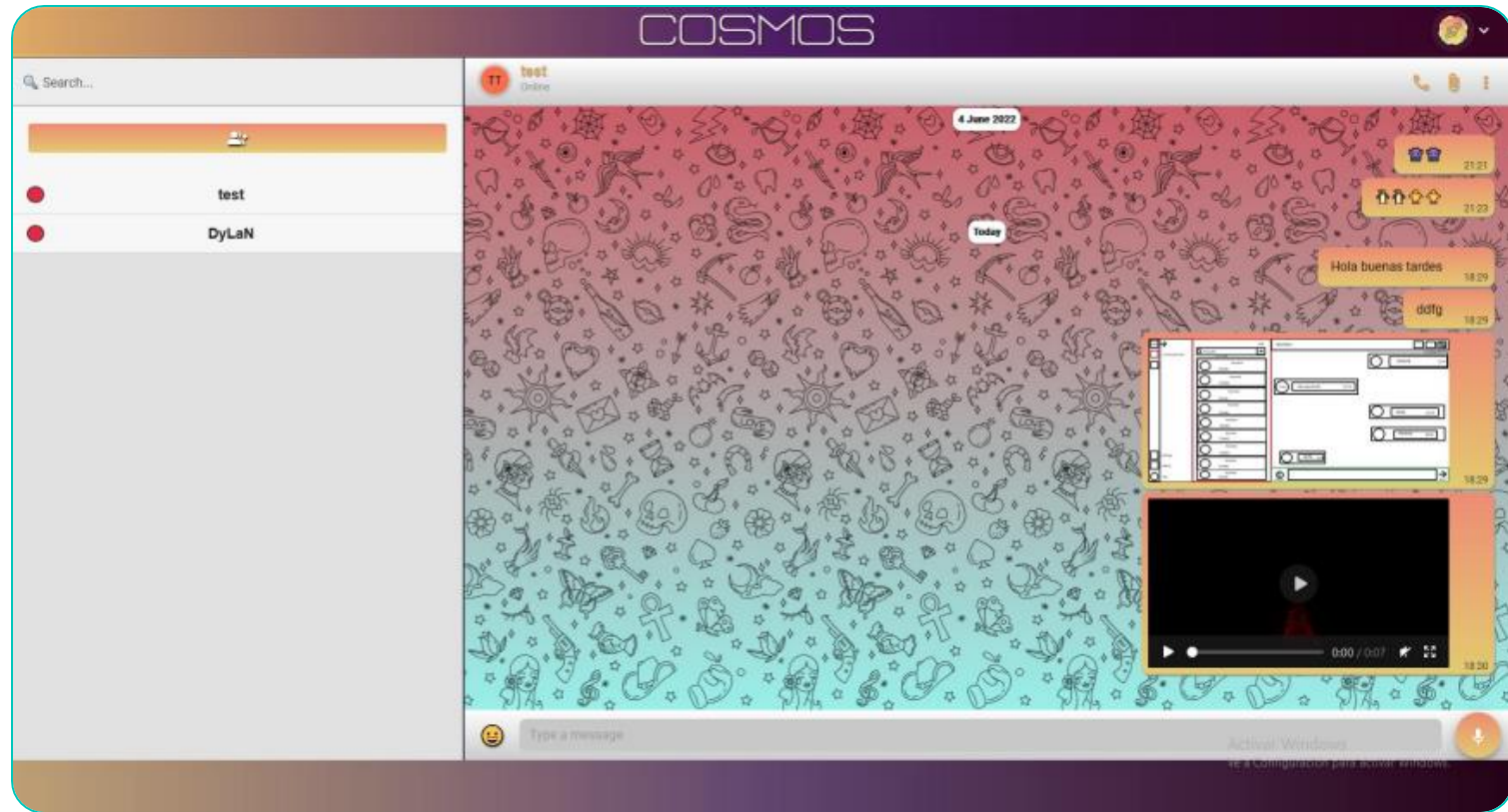
Diagrama de clases



Boceto de la interfaz



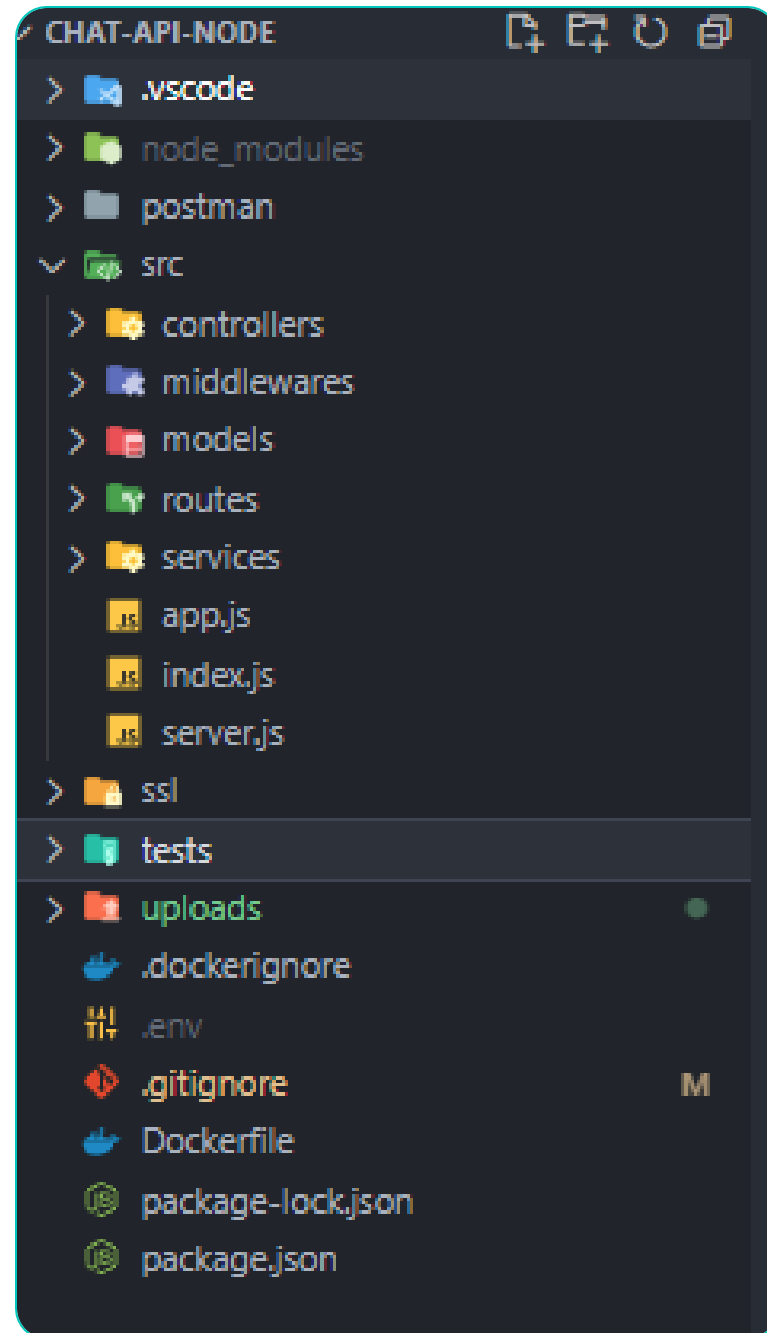
La interfaz final



Proyecto con Node.Js



Estructura del proyecto



Algunas partes más importantes

```
src > models > message.js > MessageSchema
1  const mongoose = require("mongoose");
2  const Schema = mongoose.Schema;
3
4  const MessageSchema = Schema(
5    {
6      type: {
7        type: String,
8        require: true,
9      },
10     text: {
11       type: String,
12       require: false,
13     },
14     time: {
15       type: String,
16       require: false,
17     },
18     url: {
19       type: String,
20       require: false,
21     },
22     author: { type: Schema.Types.ObjectId, ref: "User" },
23     chat: { type: Schema.Types.ObjectId, ref: "Chat" },
24   },
25   {
26     versionKey: false,
27     timestamps: true,
28   }
29 );
30
31 module.exports = mongoose.model("Message", MessageSchema);
```

Algunas partes más importantes

```
const express = require("express");
const app = express("./app");
const fs = require("fs");
const server = require("https").createServer(app, {
  key: fs.stat("../ssl/key.pem", () => {}),
  cert: fs.stat("../ssl/cert.pem", () => {}),
});
const chalk = require("chalk");
const io = require("socket.io")(server, {
  cors: {
    origins: [
      "http://localhost:4200/**",
      "http://localhost:5000/**",
      "https://cosmos-chat.netlify.app/**",
      "https://chat-api-node-prod-cosmos-chat-bultlu.mol.mogenius.io:80/**",
    ],
  },
});

server.listen(5000, function () {
  console.log(
    `\n>> ${chalk.bold.magenta(
      "WebSockets are listening at port:"
    )} ${chalk.bold.yellow(5000)}\n`
  );
});

let usersConnected = [];
io.on("connection", function (socket) {
  let { payload } = socket.handshake.query;

  if (!payload) {
    console.log(`${chalk.red(`Sin payload`)}\n`);
  } else {
    const userId = JSON.parse(payload).id;
    const userEmail = JSON.parse(payload).email;
    const userChats = JSON.parse(payload).chats;
    console.log(
      chalk.bold.magenta("User connected: " + userId + " >> ") +
      chalk.bold.blue(userEmail)
    );
  }
});
```

Algunas partes más importantes


```
const express = require("express");
const multipart = require("connect-multiparty");
const UserController = require("../controllers/user");
const md_upload_file = multipart({ uploadDir: "./uploads" });

const api = express.Router();

api.post("/user/register", UserController.register);
api.post("/user/login", UserController.login);
api.get("/user/:email", UserController.getByEmail);
api.get("/user/full-data/:id", UserController.getFullUserById);
api.put("/user/add-contact/:id", UserController.addContact);
api.put("/user/edit-profile/:id", UserController.editProfile);
api.put("/user/edit-settings/:id", UserController.editSettings);
api.put("/user/upload-avatar/:id", [md_upload_file], UserController.uploadAvatar);
api.get("/user/file/:fileName", [md_upload_file], UserController.getFile);

module.exports = api;
```

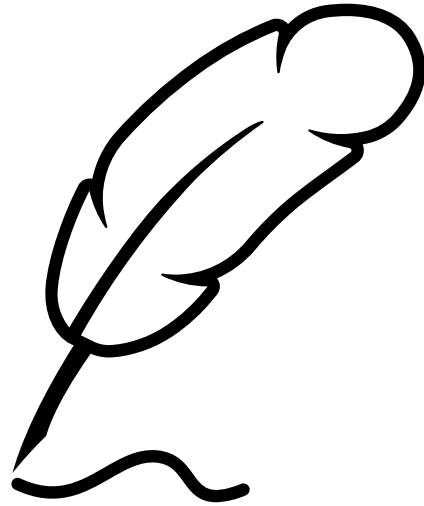
Algunas partes más importantes

```
src >  app.js > ...
1  const express = require("express");
2  const cors = require("cors");
3  const app = express();
4  app.use(express.json());
5  app.use(cors());
6
7  app.use(express.urlencoded({ extended: true }));
8  // --- Load Data ---
9  const user_routes = require("./routes/user");
10 const message_routes = require("./routes/message");
11 const chat_routes = require("./routes/chat");
12 const notFound = require("./middlewares/notFound");
13 const handleErrors = require("./middlewares/handleErrors");
14
15 // --- Base Routes ---
16 app.use(process.env.API_MAINENDPOINT, user_routes);
17 app.use(process.env.API_MAINENDPOINT, message_routes);
18 app.use(process.env.API_MAINENDPOINT, chat_routes);
19 // --- Middlewares ---
20 app.use(notFound);
21 app.use(handleErrors);
22 // --- Memory space fixed ---
23 var bodyParser = require("body-parser");
24 app.use(bodyParser.json({ limit: "50mb" }));
25 app.use(bodyParser.urlencoded({ limit: "50mb", extended: true }));
26 module.exports = app;
```

Algunas partes más importantes

```
const { MONGO_DB_URI, MONGO_DB_TEST_URI, NODE_ENV } = process.env;  
const PORT = process.env.PORT;  
  
const connectionString = NODE_ENV === "test" ? MONGO_DB_TEST_URI : MONGO_DB_URI;
```

Documentación

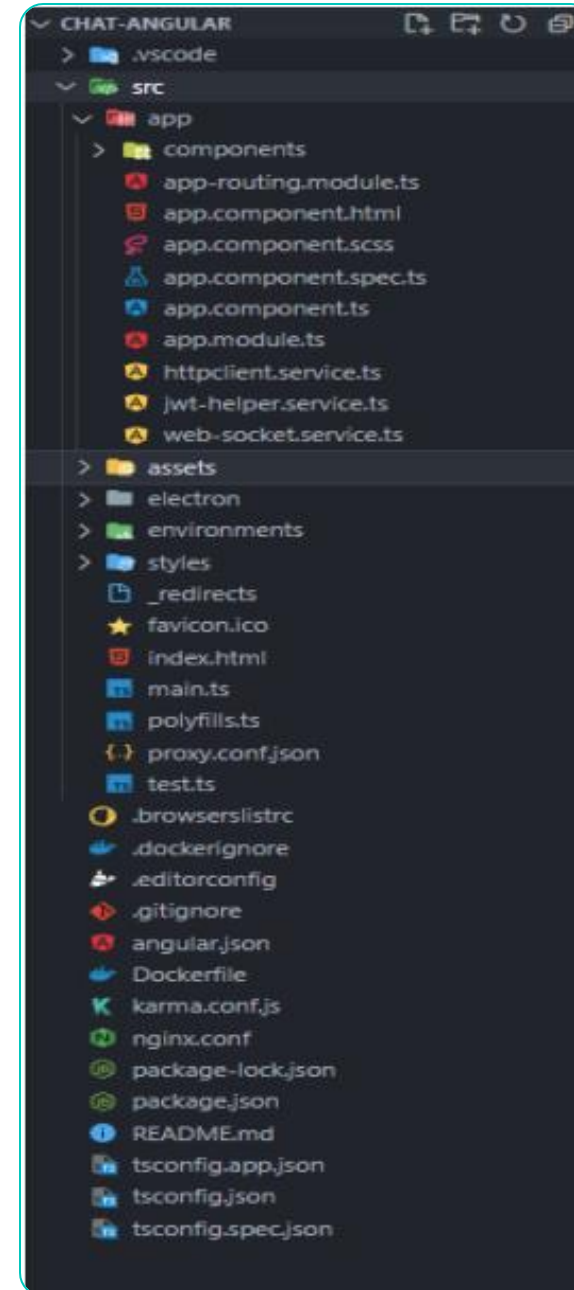


<https://documenter.getpostman.com/view/19205685/UyxqB3Vi>

Proyecto con Angular



Estructura del proyecto



```
export class HttpClientService {  
  private url:string=environment.APIOption != "CLOUD" ? environment.APILocalUri : environment.APICloudUri;  
  
  constructor(private http: HttpClient) {  
  }  
}
```

```
public getUserByEmail(auth_token:string,email:string){  
  const headers = new HttpHeaders({  
    'Content-Type': 'application/json',  
    'Authorization': `Bearer ${auth_token}`  
  })  
  return this.http.get<any>(this.url+`user/${email}`, { headers: headers, observe: 'response' });  
}
```

Algunas partes más importantes

Algunas partes más importantes

```
ngOnInit() {  
  this.socket.disconnect();  
  this.error=false;  
  this.entityForm = new FormGroup({  
    inputEmail: new FormControl("", [Validators.email,Validators.required]),  
    inputPass: new FormControl("", [Validators.required])  
  });  
  
  if(this.cookieService.get('token')!="" && this.cookieService.get('payload')!=""){  
    if (this.jwtService.isTokenValid(this.cookieService.get('token'))){  
      this.socket.connect();  
      this.router.navigate(["../home"]);  
    }  
  }  
}
```

Algunas partes más importantes

The image shows a VS Code editor with four files open, illustrating the communication between a parent component and a child component in Angular.

app.component.ts

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   title = 'default-project';
10
11   getMensaje(e){
12     console.log(e);
13   }
14 }
15
```

app.component.html

```
1 <h1>Componente padre</h1>
2 <app-hijo (miEvento) = "getMensaje($event)" ></app-hijo>
3
```

hijo.component.ts

```
1 import { Component, OnInit, Output, EventEmitter } from '@angular/core';
2
3 @Component({
4   selector: 'app-hijo',
5   templateUrl: './hijo.component.html',
6   styleUrls: ['./hijo.component.css']
7 })
8 export class HijoComponent implements OnInit {
9   mensaje = "Hola mundo";
10
11   // Creamos el evento mediante a la Clase EventEmitter junto al tipo de dato
12   // Como tenemos un string, en mensaje utilizaremos string
13   @Output() miEvento = new EventEmitter<string>();
14
15   // Utilizamos el método emit para arrojar/lanzar
16   // el evento que acabamos de crear y que contiene el valor asignado
17   // a la variable mensaje
18   ejecutarEvento(){
19     this.miEvento.emit(this.mensaje);
20   }
21
22   constructor() { }
23
24   ngOnInit(): void {
25   }
26 }
27
```

hijo.component.html

```
1 <p>Componente hijo</p>
2 <button (click) = "ejecutarEvento()">Haz click</button>
3
```

Annotations and Arrows:

- A green box highlights the `getMensaje(e)` method in `app.component.ts`.
- A green box highlights the `(miEvento) = "getMensaje($event)"` attribute in `app.component.html`.
- A red arrow points from the green box in `app.component.html` to the `miEvento` property in `hijo.component.ts`.
- An orange box highlights the `ejecutarEvento()` method in `hijo.component.ts`.
- An orange box highlights the `<button (click) = "ejecutarEvento()">` in `hijo.component.html`.
- An orange arrow points from the orange box in `hijo.component.html` to the `ejecutarEvento()` method in `hijo.component.ts`.

Algunas partes más importantes

The image displays four code snippets from an Angular project in VS Code, illustrating the relationship between a parent component, a child component, and their respective HTML templates. Annotations include a green arrow pointing from the `usuarioInChild` attribute in the parent HTML to the `@Input('usuarioInChild')` decorator in the child component, and a red arrow pointing from the `{{elUsuarioInChild}}` interpolation in the child HTML to the `elUsuarioInChild` property in the child component.

```
TS app.component.ts X
src > app > TS app.component.ts > AppComponent > (set) usuarioInParent
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   title = 'default-project';
10  private _usuarioInParent = 'User0104';
11  public get usuarioInParent(): string {
12    return this._usuarioInParent;
13  }
14  public set usuarioInParent(value: string) {
15    this._usuarioInParent = value;
16  }
17 }
```

```
TS hijo.component.ts X
src > app > hijo > TS hijo.component.ts > ...
1 import { Component, OnInit, Input } from '@angular/core';
2
3 @Component({
4   selector: 'app-hijo',
5   templateUrl: './hijo.component.html',
6   styleUrls: ['./hijo.component.css']
7 })
8 export class HijoComponent implements OnInit {
9   @Input('usuarioInChild') elUsuarioInChild;
10
11   constructor() { }
12
13   ngOnInit(): void {
14   }
15 }
16
17
```

```
app.component.html X
src > app > app.component.html > app-hijo
1 <h1>Padre works!</h1>
2 <app-hijo [usuarioInChild]="usuarioInParent"></app-hijo>
3
```

```
hijo.component.html X
src > app > hijo > hijo.component.html > ...
1 <p>Bienvenido {{elUsuarioInChild}}</p>
2
```

Algunos de los problemas encontrados



Paso de datos entre componentes sin relación padre-hijo



Criteria	Local Storage	Session Storage	Cookies
Storage Capacity	5-10 mb	5-10 mb	4 kb
Auto Expiry	No	Yes	Yes
Server Side Accessibility	No	No	Yes
Data Transfer HTTP Request	No	No	Yes
Data Persistence	Till manually deleted	Till browser tab is closed	As per expiry TTL set



RxJS

Reactive Extensions Library for JavaScript

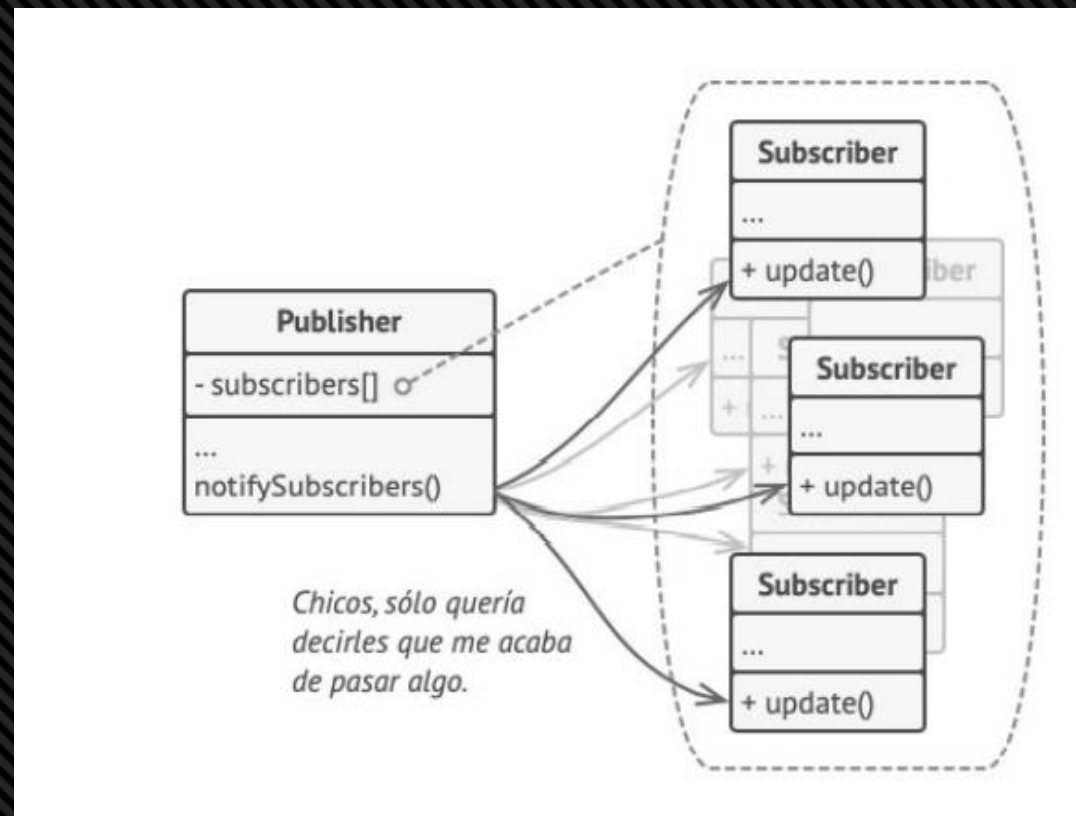
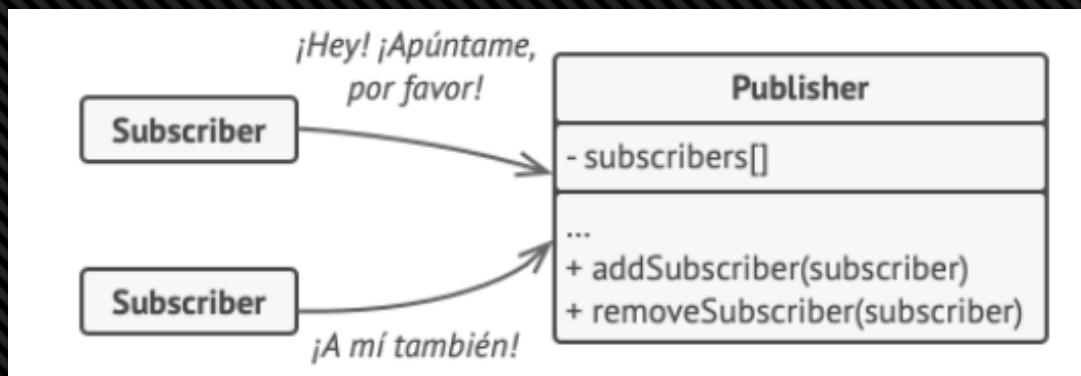
[GET STARTED](#)

[API DOCS](#)

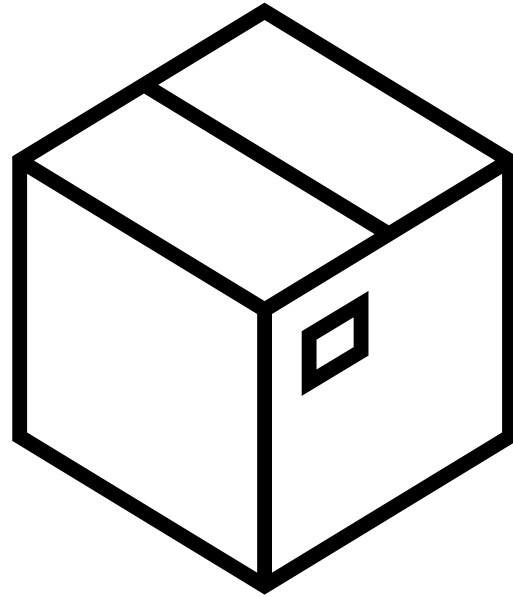
Varias peticiones HTTP a la API no funcionaban

El patrón Observer





Fase de empacquetado



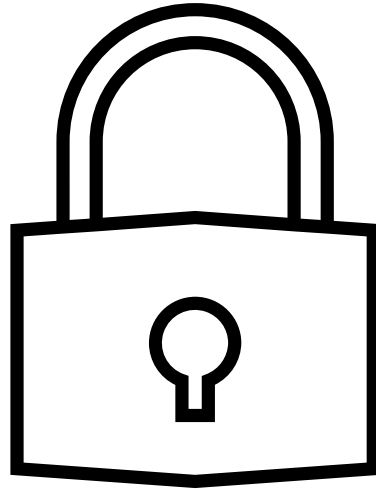
Docker y Docker Hub

- **Docker** es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de visualización de aplicaciones en múltiples sistemas operativos.
- **Docker hub** es un repositorio público en la nube, similar a Github, para distribuir los contenidos. Hay multitud de imágenes, de carácter gratuito, que se pueden descargar y así no tener que hacer el trabajo desde cero al poder aprovechar “plantillas”.

Docker Compose

- **Docker Compose** es una herramienta para definir y ejecutar aplicaciones de **Docker** de varios contenedores.
- En **Compose**, se usa un archivo YAML para configurar los servicios de la aplicación.
- Después, con un solo comando, se crean y se inician todos los servicios de la configuración.

Seguridad, Conexión Segura, SSL



¿Qué es una conexión segura?

Las conexiones seguras aseguran el nivel máximo de privacidad durante la transmisión de datos a través de Internet. En este sentido, uno de los procedimientos más sencillos para disfrutar de conexiones seguras es utilizar certificados SSL.

Cuando una página web tiene el cifrado SSL, el navegador establece una conexión segura con el servidor web y evita que nadie pueda acceder o verlo que el usuario escribe en el navegador. Para crear una conexión SSL hay que adquirir un certificado SSL, el cual creará dos llaves: una privada y una pública.

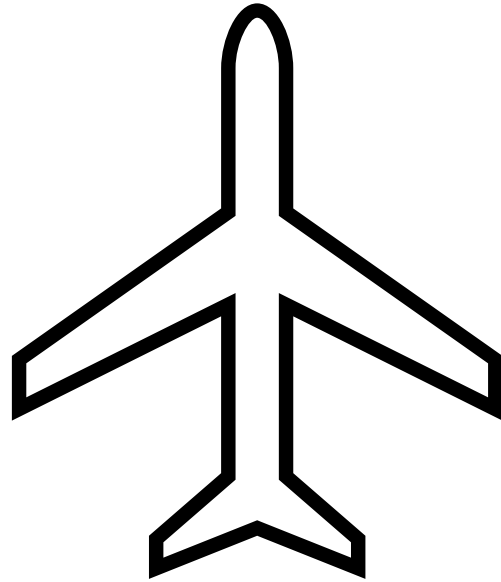
¿Qué es el Certificado Digital?

- Un certificado digital es un fichero informático firmado electrónicamente por un prestador de servicios de certificación, considerado por otras entidades como una autoridad para este tipo de contenido, que vincula unos datos de verificación de firma a un firmante.
- Tiene una estructura de datos que contiene información sobre la entidad (por ejemplo, una clave pública, una identidad o un conjunto de privilegios).
- Teniendo un certificado y unas claves pasamos de tener una conexión HTTP no segura a una HTTPS totalmente segura.

Explicación gráfica



Fase de despliegue





netlify

**Build & deploy
websites**



```
{
  "todos": [
    {
      "id": 1,
      "title": "Take over world",
      "completed": false,
      "editing": false
    },
    {
      "id": 2,
      "title": "Pinia",
      "completed": false,
      "editing": false
    }
  ]
}
```

Code to cloud. Blazing fast.



¿De qué
manera se
despliega ?

- Para el control de versiones he utilizado Git y Github.
- Ambos servicios visto anteriormente estan enlazados con los repositorios de Github
- Despliegación y test cada vez que se fusionen ramas o se hagan cambios.



Muchas gracias por atender !