DE ESTUDIANTE A PROGRAMADOR FULLSTACK

allfunds

RUBÉN SAIZ SERRANO

¿QUIÉN SOY YO?


TodoFP.es
1H-090
1H-091
18
19
20
INFORMÁTICA Y COMUNICACIONES

¿QUIÉN SOY YO?

- Ex estudiante de **IES La Senia** (DAW).
- Programador Fullstack en allfunds desde hace 3 años.
- Amante de las motos. 

MiYazJE - Overview

Web developer at Allfunds. MiYazJE has 47 repositories available. Follow their code on GitHub.

GitHub

ÍNDICE

MI CAMINO DESDE DAW HASTA FULLSTACK

Entré con ganas y muy motivado.

Pero costaba mucho aprender la sintaxis...

```java
import java.util.Scanner;

class Matricula {···

public class Main {

    Run | Debug
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);

        int C = s.nextInt();
```

A más aprendía, más dudas surgían.

Aprender a programar es como hacer un puzzle con piezas que todavía no tienes.
Hay que saltar al vacío.

Poco a poco todo iba teniendo más sentido.

Y entonces llegó la programación competitiva.

Acepta el Reto y concursos como:

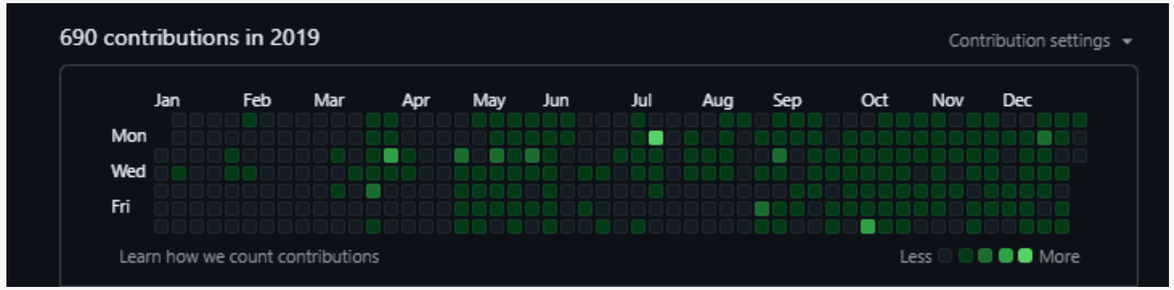
- Programame
- Las 12 Uvas

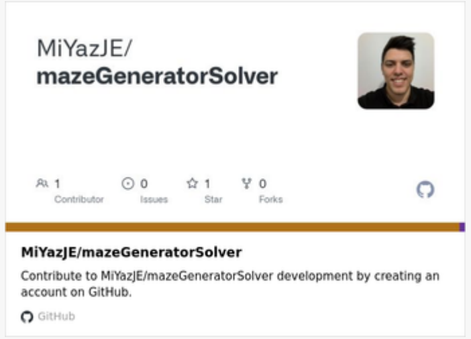
Estadísticas Acepta el Reto

Envíos	Envíos AC	Intentados	Resueltos
1523	635	470	462

Pero lo que realmente marcó una diferencia fue empezar a practicar mucho fuera de clase:

Haciendo muchos side projects





¡Llegó el momento de hacer las practicas en empresa!

Entré en una startup llamada Boatjump que estaba en la Lanzadera.

Todo iba genial. Tenía un senior que me acompañaba con todo.

Estaba flipando. Motivación al 200%

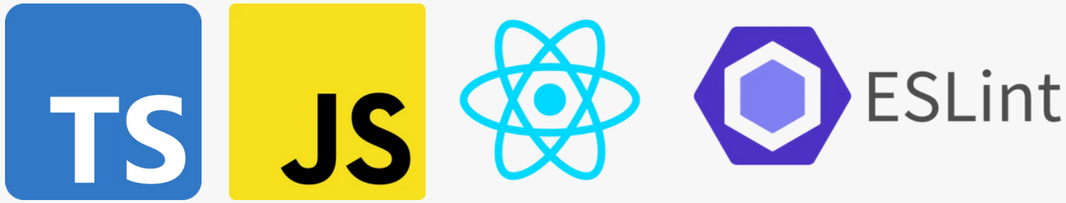
Entonces las prácticas fueron genial, no?

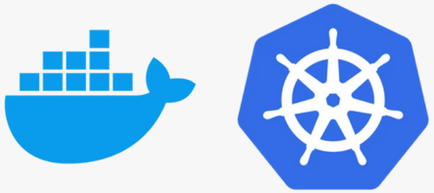
¿Te contrataron?

Mi primer paso profesional

- 🔍 6 meses buscando empleo tras terminar DAW
- 📉 Rechazos… (por ambas partes)
- 💡 Aprendí React por mi cuenta durante ese tiempo
- 👨‍💻 Entré en Gedesco como frontend
- 🔄 Descubrí mi interés por el backend
- 🚀 Llegué a Allfunds como fullstack (Node, React, microservicios)
- ❤️ Elegir bien la primera empresa sí importa

TECNOLOGÍAS Y HERRAMIENTAS



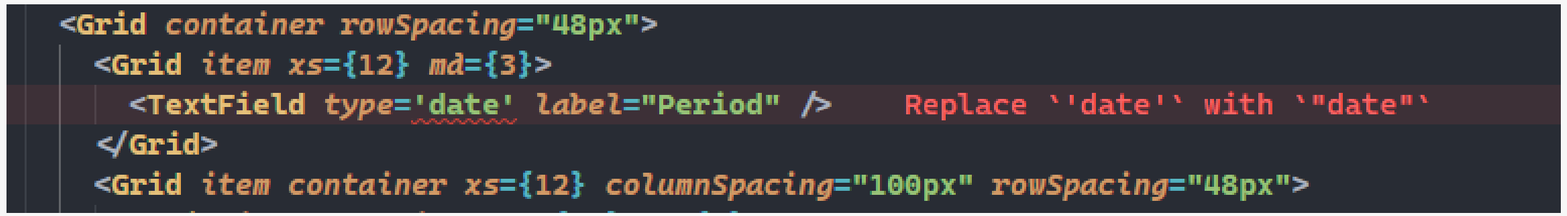


ESLint

Es una herramienta que analiza tu código JavaScript (o TypeScript) para detectar errores, malas prácticas y problemas de estilo. Funciona como un *revisor automático* que te avisa cuando rompes alguna regla predefinida o del equipo.

¿Por qué es importante en empresas?

- Homogeneidad en el código. Todo el equipo escribe con el mismo estilo.
- Prevención temprana de errores (por ejemplo: variables no usadas, condiciones inseguras, etc.).
- Mejora el onboarding (la gente nueva se adapta antes)
- Integración en pipelines (CI/CD)
- Reduce fricción en revisiones de código (los debates sobre "estilo" desaparecen)



Typescript

Es un superset de JavaScript que añade tipado estático al lenguaje.

¿Por qué usar TypeScript en empresas?

- Detecta errores antes de ejecutar
- Facilita el trabajo en equipo
- Mejora el mantenimiento y escalabilidad
- Autocompletado y desarrollo más ágil

DISCLAIMER

TypeScript no se ejecuta en el navegador ni en Node.js.

```typescript
export interface SearchInstrumentsOnlinePayload {
  instrumentTypes: InstrumentTypes[];
  q: string;
  limit: number;
  skip: number;
  mifidTestReference?: string;
}

export type InstrumentsOnlineSearchResponse = Record<
  InstrumentTypes,
  {
    data: Instrument[];
    error?: string;
  }
>;

async function searchInstrumentsOnline(
  params: SearchInstrumentsOnlinePayload,
  { signal }: { signal: GenericAbortSignal },
) {
  const { data } = await unicajaAxios.get<InstrumentsOnlineSearchResponse>(
    "/recurring-advisory/instruments-online/search",
    {
      params,
      signal,
    },
  );
  return data;
}

export { searchInstrumentsOnline };
```

Node.js es un entorno que permite ejecutar JavaScript en el servidor.
Está optimizado para trabajar con operaciones rápidas gracias a su modelo asíncrono y no bloqueante.

Express.js es un framework minimalista para Node que facilita la creación de servidores web y APIs REST de forma rápida y sencilla.

Ejemplo de un endpoint

```
4   const validator = require("./instruments-online.validator");
5   const controller = require("./instruments-online.controller");
6
7   const router = Router();
8
9   router.get(
10    "/search",
11    middlewares.validator(validator.search),
12    middlewares.controller(controller.search),
13  );
14
15  module.exports = router;
16
```

Docker

- Te ayuda a tener el despliegue y ejecución de aplicaciones mediante el uso de contenedores.
- Los contenedores permiten empaquetar una aplicación junto con todas sus dependencias
- A diferencia de las máquinas virtuales tradicionales, los contenedores comparten el sistema operativo del host, lo que los hace más eficientes.

Utilizamos también docker compose para orquestar la interconectividad entre nuestros microservicios.

```
FROM node:22-alpine as build    Tag recommendations availab

ARG NPM_TOKEN
RUN npm set //registry.npmjs.org/:_authToken=$NPM_TOKEN

WORKDIR /app
COPY package*.json .
RUN npm ci

FROM node:22-alpine    Tag recommendations available

RUN apk add --no-cache tzdata
ENV TZ Europe/Madrid

WORKDIR /app
COPY --from=build /app .
COPY . .

EXPOSE 3000

USER node
CMD ["node", "src/server.js"]
```

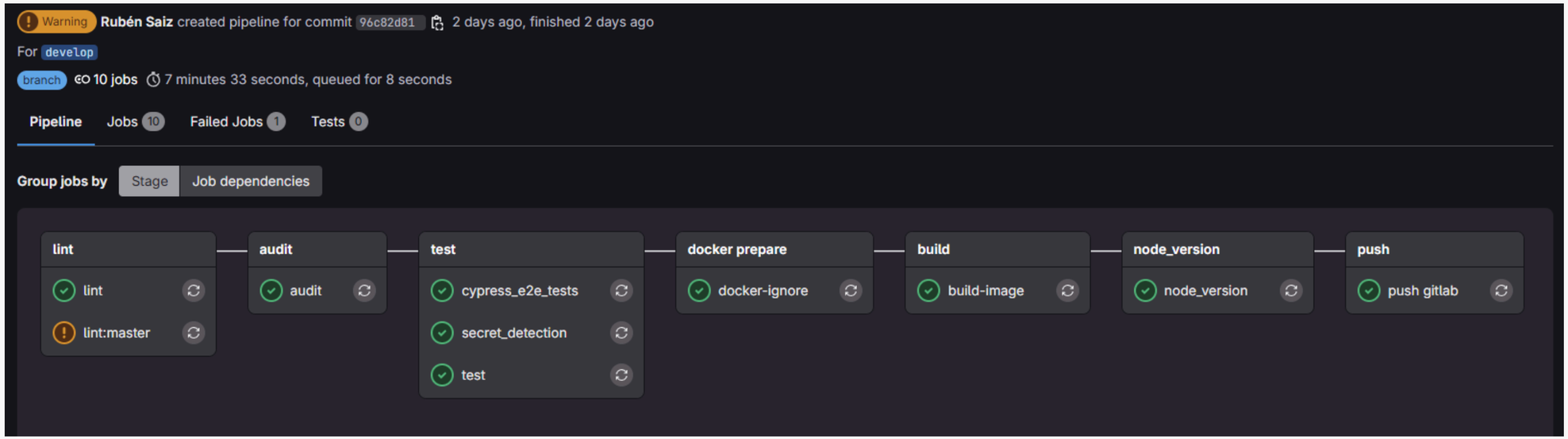
CI/CD

- CI: Continuous Integration
- CD: Continuous Delivery/Deployment

Son un conjunto de tareas automatizadas como compilar, testear, construir imágenes Docker, y desplegar tu aplicación. Estas son las más comunes.

Pipelines con Gitlab

```yaml
cypress_e2e_tests:      ruben_saiz_allfunds,
  stage: test
  image: cypress/browsers:node-22.11.0-chrome
  needs: []
  tags:
    - gitlab-allfunds-channels
  rules:
    - if: "$NP_DISABLE_TEST ≠ 'true'"
      when: always
  cache:
    paths:
      - node_modules/
      - ~/.cache/Cypress
  before_script: ···
  script:
    - npm run dev &
    - npx wait-on http://localhost:8080
    - npm run cy:headless-pipeline
```



Kubernetes

Es una plataforma que automatiza el despliegue, escalado y gestión de aplicaciones en contenedores.
Organiza tus contenedores en un clúster y se asegura de que estén siempre funcionando como deben.

Define tu estado deseado de tu infra y k8s hará lo que podrá.

```yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: nextportfolio
  name: finaccess-onboarding
  labels:
    app: finaccess-onboarding
spec:
  replicas: 1
  selector:
    matchLabels:
      app: finaccess-onboarding
  template:
    metadata:
      labels:
        app: finaccess-onboarding
    spec:
      containers:
      - name: finaccess-onboarding
        image: [redacted].amazonaws.com/finaccess-onboarding:latest
        ports:
        - containerPort: 80
```

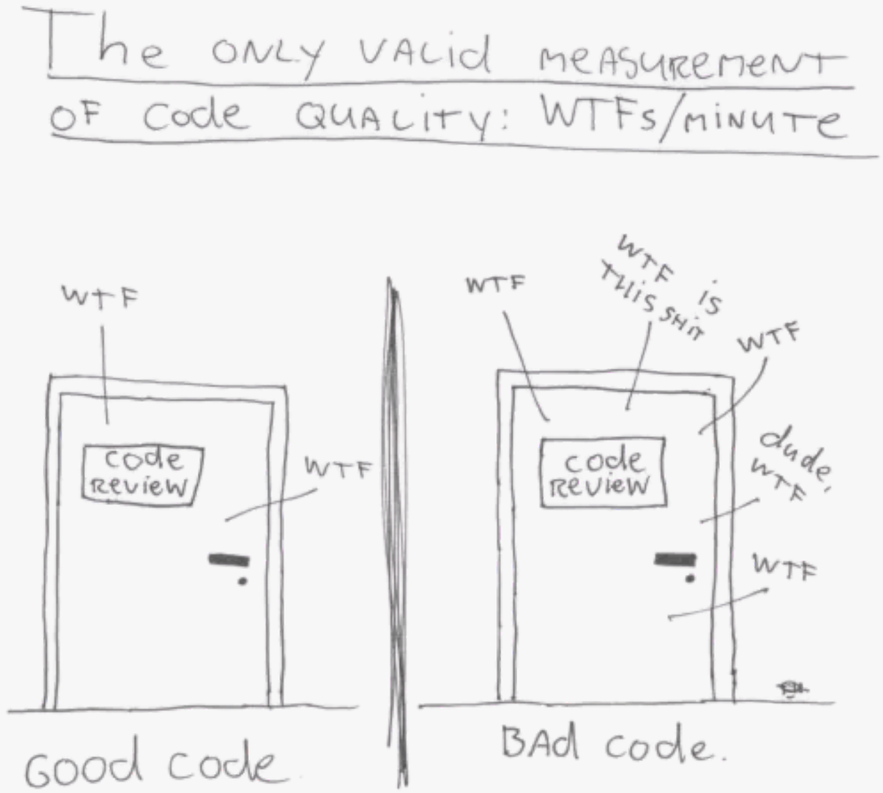
🤩Buenas prácticas y metodologías

- Escribir código no es solo hacerlo funcionar.
- Es mantenerlo limpio, legible, testeado y bien documentado.
- En Allfunds trabajamos con metodologías ágiles, sprints de dos semanas, reuniones diarias...
- Aprender a trabajar en equipo es tan importante como saber programar.

📝Code reviews

Es el proceso en el que uno o más miembros del equipo revisan el código escrito por otro desarrollador antes de que este sea integrado.

- Mejora la calidad del código
- Fomenta el aprendizaje y la colaboración
- Asegura consistencia y estándares




let me merge.
[adult swim]
LET ME MEEEEERGE!
imgflip.com
[adult swim]

🧪 Testing

Consiste en escribir pruebas automáticas para verificar que el código hace lo que debe hacer.

Tipo de Test	Descripción	Velocidad	Propósito
Unit Test	Prueba una unidad de código (función, método, clase) en aislamiento, sin dependencias.	🔥 **Muy rápido**	Verificar que componentes individuales del código funcionan correctamente.
Integration Test	Verifica cómo interactúan varias unidades o componentes entre sí.	⚡ **Moderado**	Asegurar que los componentes interactúan correctamente entre ellos.
End-to-End Test (E2E)	Simula el flujo completo de un usuario desde el inicio hasta el final.	🐢 **Lento**	Validar que la aplicación funcione como un todo desde la perspectiva del usuario.
Regression Test	Asegura que el código nuevo o las modificaciones no hayan roto funcionalidades existentes.	⚡ **Moderado a lento**	Verificar que el sistema siga funcionando correctamente después de cambios.

No test no party 😔

¿Qué pasa sin testing?

- Miedo a tocar el código
- Bugs constantes en producción
- Desarrolladores apagando fuegos en lugar de avanzar
- Clientes frustrados, genera desconfianza

Programación competitiva

Es una práctica que implica resolver problemas algorítmicos y de programación dentro de un tiempo limitado.

- Resolución de problemas
- Tiempo y Espacio Limitados

Plataformas en línea:

- Acepta el Reto
- LeetCode
- Codeforces
- HackerRank
- TopCoder
- AtCoder
- Codewars

¿Y cómo se piensa algorítmicamente?

Haciéndose a uno mismo las preguntas adecuadas:

- ¿Qué hay que lograr?
- ¿Qué información tienes?
- ¿Puedes transformar esta información a una estructura de datos mejor?

En resumen, intentamos dividir el problema:

- Identificando patrones
- Buscando repeticiones

¿Cómo se calcula el coste de un algoritmo? 🤔

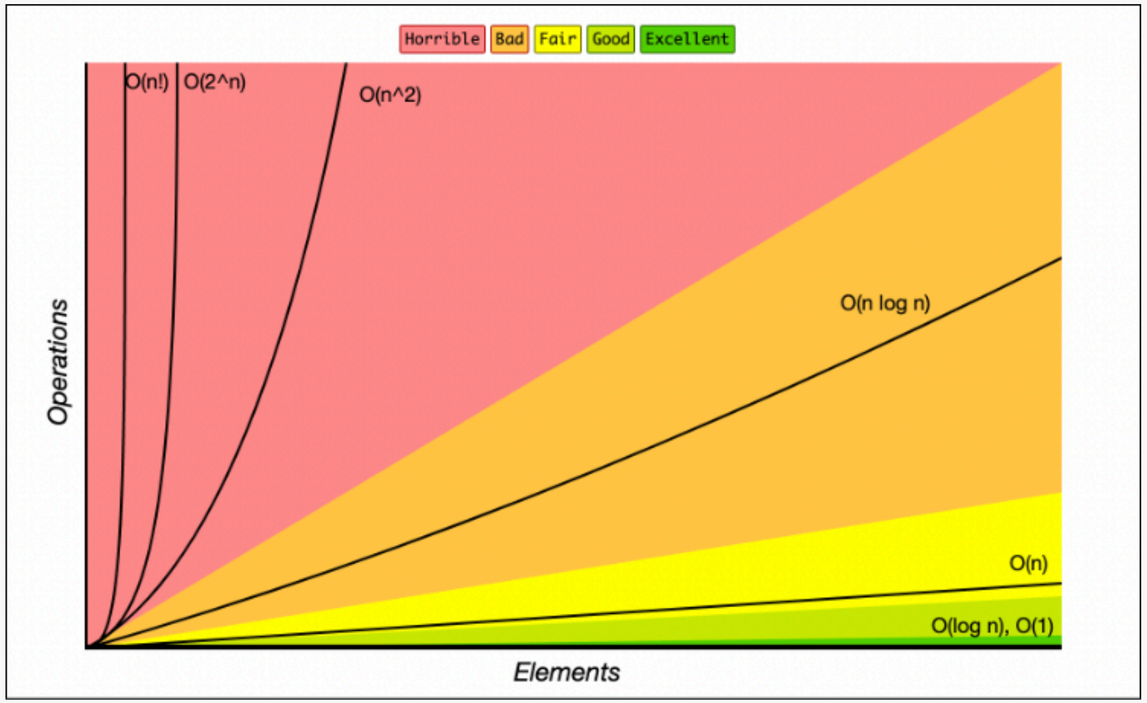
1. ¿El tiempo de ejecución? ✅ ❌
2. ¿La cantidad de líneas de código? ❌
3. ¿La cantidad de cafés que tomas? ❌
4. ¿Por la silla gaming? ❌

Notación Big O

Es una forma de medir la eficiencia de un algoritmo en términos de cuánto tiempo o espacio necesita para ejecutarse en función del tamaño de la entrada.

Tipo de notación	Notación	Ejemplos de algoritmos
O(1)	Constant time	Acceder a un elemento de un array por índice
O(log n)	Logarithmic time	Búsqueda binaria
O(n)	Linear time	Recorrer un array
O(n log n)	Linearithmic time	Algoritmo de ordenación QuickSort
O(n^2)	Quadratic time	Algoritmo de ordenación BubbleSort
O(2^n)	Exponential time	Algoritmo de Fibonacci recursivo
O(n!)	Factorial time	Algoritmo de permutaciones

n	O(1)	O(log n)	O(n)	O(n log n)	O(n^2)	O(2^n)	O(n!)
10	1	3.32	10	33.2	100	1024	3628800
100	1	6.64	100	664	10000	1.27e+30	9.33e+157
1000	1	9.97	1000	9967	1000000	1.07e+301	4.02e+2567



Consejos prácticos

- Aprende de la manera que más se adapte a ti
- Esto es un juego constante de prueba y error: ***Te caes, te levantas***
- Participa en comunidades: online o presenciales
- Nunca dejes de practicar

No se trata de ser el mejor, sino de ser mejor que ayer

¿Preguntas? 🙋‍♂️