INTRODUCCIÓN

# PROCESO DE ENTREVISTAS

Daniel Blanco Calviño

# ENTREVISTA CONDUCTUAL

- Conocer más de cerca al candidato.
  - **Experiencia**
  - **Soft Skills** (comunicación, trabajo en equipo, resolución de conflictos etc.)

- **Filtro básico** para saber si el candidato encaja con el puesto y el equipo.

- Tipo de entrevista típico. Lo tienen todas las empresas.

# TIPOS DE ENTREVISTAS TÉCNICAS

- **Conversacionales**
  - Se presentan unos temas técnicos y se hacen preguntas al candidato.

- **Mini-proyectos**
  - Se le pide al candidato trabajar en un proyecto pequeño fuera de la entrevista.
  - Aplican los mismos puntos que al crear tu portfolio. Buen código, tests, buena presentación etc.

- **Resolución de problemas.**
  - Entrevistas de unos 40 minutos colaborando con el entrevistador para resolver un problema de algoritmos y estructuras de datos.

# 1. PRESENTACIÓN DEL PROBLEMA

- **El entrevistador te describe el problema.**
  - Puede ser de forma oral.
  - Normalmente te lo dejan de forma escrita.

- **Presta atención a todos los detalles.**
  - No se suele dar información sobrante.
  - Seguramente necesites toda la información para la solución óptima.

- **Pregunta todas las dudas que tengas.**
  - No comiences sin antes haber entendido todo.

# 2. TRABAJA CON UN EJEMPLO

- **No debes escribir nada de código aún.**
  - Fallo muy grave y común.

- **Elige un caso como ejemplo.**
  - No elijas un ejemplo muy pequeño / grande o con casos especiales.

# 3. FUERZA BRUTA

- **Solución poco óptima para el problema.**
  - No intentes sacar la mejor solución al problema directamente.

- **No te preocupes si es muy obvia e ineficiente.**

- **Habla con tu entrevistador.**
  - Se va a valorar tu capacidad de trabajo en equipo.

# 4. OPTIMIZACIÓN

- **Fíjate en la información no usada.**
  - Te puede dar pistas para optimizar tu solución.

- **Optimización BUD:**
  - Cuellos de botella.
  - Trabajo innecesario.
  - Trabajo dublicado.

- **Concesiones espacio-tiempo.**

- **Precomputa información.**

- **Piensa en la mejor complejidad posible.**

# 5. IMPLEMENTACIÓN

- **Escribe buen código.**
  - Se valorará positivamente el código legible.

- **No te preocupes si no te acuerdas del nombre de una función.**
  - La sintaxis no es determinante.

# 6. TEST

- **Importantísimo probar a línea a línea tu solución.**
  - Es muy sencillo que se te haya pasado algún caso.
  - Fíjate en los límites (primeras y últimas posiciones de arrays, valores null etc.)

- **Si encuentras algún problema, analízalo.**
  - Si no te da tiempo a solucionarlo comenta cual sería la solución.

- **Si tienes tiempo, refactoriza el código.**