Algunas preguntas típicas de entrevista:
* Que es una lista, un hashset, hashmap. Diferencias y Orden de ejecucion de sus algoritmos principales.
* Cómo solucionar colisiones infinitas en un hash.
* En que situacion usaria una estructura de pila.
* Orden de la búsqueda de un elemento en un hash.
* Que es CORS?
* Que es SOLID?
* Que es una clase?
* Que es un objeto?
* Que es polimorfismo?
* Que es sobrecarga?
* Que es una clase abstracta?, Que es una interfaz?, que diferencia hay entre una clase abstracta y una interfaz?
* Como haría un sistema que tiene que guardar elementos, encontrar el mayor de todos y poder retornar todos los ele mentos.
* Como es el proceso en el cual se pushea codigo al repositorio, no arruine todo el sistema que ya está funcionando?
* Si tuviera que hacer una web muy compleja como haría?
* Que diferencia hay entre un http post y un http get? Por donde van los parámetros en uno y en otro?
* Conta de una vez que tuviste un deadline muy apretado, como hiciste para cumplirlo. Si no lo cumpliste, como res olviste la situacion.
* Diferencia entre dom y html.
* Diferencia entre div y span.
* Feature favorito de js ES6, justificar por que.
* Que son los hooks? cuales use, donde y por que?
* Que es redux? Para que sirve?
* Que le agregaria a js?
* Diferencias entre BD relacional y no relacional, cuando usaría una y cuando otra.
* Cómo funciona node? Hablar de event loop, single thread, callbackstack.
* Diferencias entre Angular y React.
* Cuando usar un memo / useCallback y cuando no
* Que es el useContext, cuando y como se usa. Cuando no usarlo. Por que no usarlo.
* Que es redux, como se hace una petición desde un componente a una api pasando por redux. Diferencias entre use Context.
* Cual es el problema los css.module en react.
* Como se uplodea una imagen desde el front hasta el aws s3 bucket.
* Que librerias graficas use, cuándo usaría alguna u otra.
* Que es Sass, que beneficios trae.
* Se dice que React es mas facil de leer y entender que otros frameworks js, por que?.
* Que es una arquitectura de microservicios? Cuando y por que use una? que beneficios trajo respecto a una arquitec tura tradicional?
* Diferencia entre const, let y var.
* Hablar del scope de las variables y funciones en js.
* Diferencia entre null y undefined en js.
* Cuales son las ventajas que creo yo que tiene ts sobre js.
* Diferencia entre == y === en js.
* Explicar como funciona react.
* Diferencia entre el dom real y el dom virtual.
* Explicar el ciclo de vida de componentes de react.
* Que son los high order components.
* Explicar distintas maneras de enviar props a los componentes, ventajas y desventajas de cada una.
* Que observo cuando hago un codereview?
* Cómo hago para mantener calidad en el código.
* Que es el pool thread?
* Explicar con algún ejemplo el uso de un load balancer.
* Que beneficios nos trae nextJs, por qué usarlo?
* Cómo se definen las url dinámicas en nextjs? para qué sirven? Citar un ejemplo.

* Cuál fue el desafío más grande que tuve con React.
* Disenar una interfaz identificando componentes claves para una app de conversion de unidades, metros a yardas, k m a millas…
* Hacer en JS una función para saber si dos palabras son anagramas.
* Que es una rest API, como se define una rest API.
* Que es una arquitectura monolítica y que es una arquitectura de microservicios. Pros y contras.
* Cómo se comunican los distintos servicios en una arquitectura de microservicios.
* Puede dos microservicios compartir las bases de datos? por que es una mala práctica.
* Nombrar todas las partes de redux y su funcion.
* Diferencia entre mandar los parámetros de una petición a una api por header y crear un nuevo endpoint.
* Cuales son las partes del useEffect, para que sirve.
* Armar un algoritmo eficiente para calcular camino más corto entre nodo A y nodo B en un grafo.
* Armar un algoritmo eficiente para mergear K listas ordenadas de manera ascendiente: https://leetcode.com/proble ms/merge-k-sorted-lists/
* Disenar en un pizarron el juego de la batalla naval.
*¿Qué es un componente?
*¿Qué es un middleware?
*¿Cuál es la diferencia entre var, let y const?
*¿Cuál es la diferencia entre JavaScript y Node?
*¿Qué ventajas tiene typescript? (todos responden el tipado como única opción pero hay varias más)

Recomendaciones preparación entrevista laboral:
Bueno después de tener más de 30 entrevistas algunas en inglés, conseguí el primer trabajo en IT , he recopilado alg unos tips, que los voy a ir enumerando aquí espero que les sirva.. BUSCAR TRABAJO ES UN TRABAJO
NO COMPARARSE, si vieron a alguien con menos conocimientos que uno mismo que consiguió trabajo antes que nosotros, eso debería motivarnos más y no frustrarnos. algo positivo es preguntar cómo lo lograron. hay mucha gent e buena allá afuera dispuesta a ayudar, solo es cuestión de preguntar, ANIMENSE !
PREPARARSE: esto es clave al momento de llegar a la entrevista, saber quienes son las o la persona que nos van a entrevistar, vean su perfil busquen toda info necesaria para poder generar empatía desde el minuto uno. estudien a la empresa, vean su visión y misión y traten de alinearse.
VENDERSE: esto hace la diferencia entre un buen candidato y uno que no lo es, inclusive más allá de su habilidad t écnica. Hagan de cuenta que cada entrevista es la puerta de entrada a una empresa, piensen en cómo convencer al rec lutador a que nos deje pasar. como aquel vendedor que pasa casa por casa sabiendo que el NO ya lo tiene antes de go lpear la puerta y solo va por el SI.. Aprender a generar en los demás que somos lo que están buscando y que podemo s aportar valor a su empresa.
SER UNO MISMO: traten de mostrarse lo más relajados posible, piensen en que es una simple charla para sacar los nervios. Tener la mente fresca y calma hace dar las respuestas de mejor calidad.. Cuando nos pregunten sobre nosotr os, no debemos extendernos más de 3 minutos, dar una breve storytelling para no aburrir y no contar cosas que no se an relevantes.
PRENDER SI O SI LA CÁMARA, aunque el reclutador no lo haga, miren siempre la cámara para generar el face to face y no leer ni mirar a los costados, eviten distracciones y sonrían que la buena onda se contagia!
NO MENTIR si no saben sobre algo digan que no saben y listo, no mientan que queda muy mal cuando se dan cuent a. muestren predisposición a aprender eso que no saben. Imagina un hipotetico caso que entras a la empresa mintiend o sobre cuestiones tecnicas y habilidades blandas que no tienes, la vas a pasar muy mal y lo que es peor que se sume n a un equipo de trabajo que maneja una tecnologia que jamas viste, ante tantas reuniones y tareas sin poder resolver, vas a tener una muy mala experiencia dentro de esa empresa.. Entonces ir con LA VERDAD te llevará siempre a bu en puerto.
NO MOSTRARSE DESESPERADOS ante una posible oportunidad, los reclutadores tienen la capacidad de identifi car a alguien que haría cualquier cosa solo por conseguir el primer trabajo asi sea mentir o decir a todo que SI. Trate n de hacer match con lo que están buscando.
FIT CULTURAL: importantísimo! muchos se quedan afuera por este detalle sin saberlo.. Imaginen a un desarrollad or con 15 años de experiencia que puede hacer absolutamente todo pero que le gusta trabajar solo. NO LE SIRVE A NINGUNA EMPRESA entonces, mostrarse que sabemos trabajar en equipo, que tenemos capacidad de adaptarnos a los cambios y que somos muy sociables y colaborativos es clave para poder integrarse a un equipo de desarrollo.. en tonces alguien con mucha capacidad técnica pero con pocas SOFT SKILLS puede restar efectividad a un grupo, ento

nces no se trata solo de cumplir las expectativas técnicas sino también las culturales de la empresa.. Las famosas SO FT SKILLS identifiquen cuales son las suyas y háganlas notar en la entrevista.
EVALUACIÓN TÉCNICA: muchas empresas optan enviar un challenge para hacer en un determinado tiempo, que el fin no se trata estrictamente si lo pudiste terminar o no sino en cómo lo hiciste, ver si tienes buenas prácticas y en l a manera que decidiste resolverlo, ahi se ve el seniority claramente. busca resolver las cosas de manera simple y senc illa, cuanto menos es mejor.. las empresas buscan a alguien que sea objetivo y no que pierda tiempo en cosas irreleva ntes. Otras empresas optan por hacer preguntas técnicas directamente con el equipo técnico y ahí es donde saber con claridad la teoría hace la diferencia. y para el Live Coding les recomiendo practicar hackerrank eso les va a dar más s oltura.
NETWORKING: esto es muy importante, piensen en lo siguiente.. el Tech Lead de una empresa decide abrir una va cante para un puesto JUNIOR, el departamento de HR publica la vacante en LinkedIn y en una hora se postulan 600 personas, el departamento de HR se colapsa inmediatamente, imposible poner a todos en HR a analizar los CV, les ll evaría meses y es improductivo.. ahí es donde hacer networking puede acercarse a esa vacante, escribiéndoles a los Tech Lead de las empresas poniéndote a su disposición junto con tu CV como así también a los developer que estén trabajando ahí dentro.. todas las empresas tienen un programa de referidos en donde recomiendan a otros developer y asi acortan todo este proceso. ANIMATE Y EMPEZA a escribirles!
RECOMENDACIONES EXTRAS: de ser posible graben las entrevistas y analizarlas después tranquilo es fundamen tal para que la próxima sea mejor, anoten las preguntas que les hicieron y vean cuales respondieron mal y cuales no pudieron, y vean en que pueden mejorar, tomen notas.. porque suele pasar que por los nervios después de las entrevi stas no nos acordamos de nada..
RECLUTADORES: muchos reclutadores te escriben por LinkedIn para pedirte una entrevista que les encantó tu perf il y ni siquiera lo han visto, ni mucho menos tu CV que quizás se lo enviaste por mail bueno esa es la mejor oportuni dad para sorprender! hagan proyectos tengan los deploy a mano para mostrarlos y contar con entusiasmo como los hi cieron y que están aprendiendo actualmente. Que en caso de que esa vacante no sea para vos es probable te avisen si surge alguna.
ESPECIALIZARSE: esto es un punto que no solo te puede hacer conseguir rápido un trabajo sino que te puede perm itir ganar más.. Hay mucha gente con el mismo stack tecnológico y eso hace que la competencia sea más fuerte y las vacantes están saturadas de postulantes, entonces especializarse en un área en concreto ayuda muchísimo.. puede ser especializarse en Frontend con Angular, Svelte, o Backend con Java o .Net también como desarrollador Mobile con Kotlin o React Native está también Devops, QA... y muchas más! indaguen, busquen información e hagan un roadm ap y empiecen a ser uno de los pocos y no del montón.
DE ESTO NO SE TIENE QUE ESCAPAR NADIE: hay empresas en las que no quieren tener QA (analistas de calid ad) que verifican que esta bien lo que has hecho, entonces todas tus implementaciones tienen que ir directamente a p roduccion o a la rama elegida por el flujo de trabajo. y para que eso sea posible tu codigo tiene que pasar por unas "T UBERIAS" que pueden ser (segun el arquitecto de datos que haya decidido configurar en los pipelines) puden ser co rrer los test de unitarios, integracion, coverage y build ej.. osea que lo que vos hagas se integre a lo que ya este hecho y que no haga que lo que anda deje de funcionar, se entiende? si te vas a dedicar al desarrollo es OBLIGATORIO sa ber y hacer TESTING osea cualquier aplicación que hagan en React o Angular o Vue o Backend con Node HAGAN TEST eso es clave para poder trabajar en una empresa. APRENDAN TEST UNITARIOS Y DE INTEGRACIÓN y también CI/CD de github Action o de Gitlab Pipelines.