Réussir son entrentien d'embauche Tech

Edition Fusianima

Réussir son entretien d'embauche Tech

Par Fusianima Expert

ÉDITIONS FUSIANIMA

Lire la version interactive sur Fusianima.com

Table des matières

Chapitre 1

Le Mindset du Développeur Gagnant : Plus qu'un Codeur

Module : Le Mindset du Développeur Gagnant : Plus qu'un Codeur

Réussir son entretien tech ne se résume pas à aligner des lignes de code parfaites. Les entreprises recherchent aujourd'hui des profils capables d'apporter une valeur ajoutée concrète au-delà de la syntaxe. Ce module vous aide à transformer votre vision du métier pour devenir le candidat que tout le monde veut recruter.

Point 1 : Décoder les attentes réelles des recruteurs tech

Beaucoup de candidats pensent que le test technique est l'unique critère de sélection. C'est une erreur. Le recruteur cherche avant tout à valider votre potentiel de collaboration et votre fiabilité.

• La capacité d'apprentissage : Dans un monde où les frameworks changent tous les deux ans, votre curiosité est plus précieuse que vos acquis.

• L'intelligence émotionnelle : Savez-vous accepter une critique sur votre code (Code Review) sans le prendre personnellement ?

• La communication : Pouvez-vous expliquer un problème technique complexe à un chef de produit qui ne code pas ?

• La fiabilité : Le recruteur veut savoir s'il peut vous confier une tâche et dormir tranquille.

Point 2 : Adopter la posture de "Solutionneur de Problèmes"

Le code n'est qu'un coût pour l'entreprise ; la solution est ce qui rapporte de l'argent.

Un développeur gagnant pense "Business" avant de penser "Algorithme".

• Privilégier le "Pourquoi" : Avant de demander "quelles technos on utilise ?", demandez "quel problème essayons-nous de résoudre pour l'utilisateur ?".

• Le pragmatisme : Parfois, la meilleure solution technique n'est pas de coder, mais d'utiliser un outil existant ou de simplifier le besoin.

• La gestion de la dette technique : Montrez que vous comprenez l'équilibre entre sortir une fonctionnalité rapidement et maintenir un code propre.

• L'impact utilisateur : Gardez toujours en tête que derrière chaque ligne de code se trouve un humain qui utilise votre produit.

Point 3 : Surmonter le syndrome de l'imposteur avant de postuler

Le syndrome de l'imposteur est le premier frein à l'embauche. Il vous fait douter de vos compétences et vous empêche de négocier votre juste valeur.

• Désacraliser l'expertise : Personne ne connaît 100% d'une stack technique. Même les seniors passent 50% de leur temps sur Stack Overflow ou la documentation officielle.

• Valoriser le transfert de compétences : Si vous venez d'un autre domaine, votre rigueur, votre organisation ou votre esprit d'analyse sont des atouts majeurs.

• Transformer le "Je ne sais pas" : En entretien, remplacez-le par : "Je n'ai pas encore utilisé cette technologie, mais voici comment je m'y prendrais pour monter en compétence en deux semaines."

• Documenter ses succès : Avant de postuler, listez trois projets dont vous êtes fier et détaillez les obstacles que vous avez surmontés.

LE CONSEIL PRO : Ne vous présentez jamais comme un simple "exécutant". Lors de vos entretiens, utilisez le "Nous" pour parler de l'entreprise et projetez-vous comme un partenaire technique. Le recruteur doit sentir que vous n'êtes pas là pour recevoir des ordres, mais pour aider l'équipe à atteindre ses objectifs grâce à la technologie.

Chapitre 2

CV et Portfolio : Ton Ticket d'Entrée Magnétique

CV et Portfolio : Ton Ticket d'Entrée Magnétique

Dans le monde de la tech, ton CV ne sert pas seulement à lister tes diplômes. C'est un outil marketing conçu pour franchir deux barrières : les algorithmes de tri (ATS) et l'œil critique du recruteur technique.

Étape 1 : Dompter les algorithmes ATS (Applicant Tracking Systems)

La majorité des grandes entreprises utilisent des logiciels pour filtrer les candidatures avant même qu'un humain ne les voie. Voici comment optimiser ton CV pour ne pas être éliminé par erreur :

• Mise en page simple : Évite les colonnes complexes, les graphiques de compétences (type barres de progression) et les images. Les ATS lisent de gauche à droite et de haut en bas.

• Mots-clés stratégiques : Analyse l'offre d'emploi et intègre précisément les technologies demandées (ex: "React", "Node.js", "AWS") dans une section Compétences Techniques dédiée.

• Format de fichier : Privilégie le format PDF pour conserver la mise en forme, tout en t'assurant que le texte est bien sélectionnable (pas de CV "image").

• Hiérarchie claire : Utilise des titres de sections standards comme "Expériences Professionnelles", "Formation" et "Projets".

Étape 2 : Transformer ton GitHub en preuve de compétences

Pour un développeur, un profil GitHub vide est une opportunité manquée. Ton code est ta vérité terrain.

• Soigne ton profil : Ajoute une photo, une bio claire et un lien vers ton LinkedIn ou ton site personnel.

• Le README est ton meilleur ami : Chaque projet important doit avoir un fichier README.md détaillé comprenant :

• Une description claire du "Pourquoi" du projet.

• La stack technique utilisée.

• Une procédure d'installation simple (cloner, installer, lancer).

• Des captures d'écran ou un lien vers une démo en ligne.

• La qualité avant la quantité : Mieux vaut trois dépôts (repos) impeccables, documentés et avec un code propre, que cinquante projets abandonnés après le "Hello World".

• Activité régulière : Ton graphique de contributions (les petits carrés verts) montre ton engagement et ta régularité dans l'apprentissage.

Étape 3 : Mettre en valeur tes projets personnels percutants

Tes projets personnels remplacent souvent l'expérience manquante. Ils doivent démontrer que tu sais résoudre des problèmes réels.

• Choisis des projets "Réels" : Évite les énièmes "To-Do List". Crée quelque chose qui répond à un besoin (ex: une extension de navigateur, un outil d'automatisation, une API publique).

• Explique tes choix techniques : Dans ton CV, ne liste pas juste le projet. Précise

pourquoi tu as choisi telle base de données ou tel framework.

• Quantifie l'impact : Si ton projet a des utilisateurs, mentionne-le. Ex: "Application utilisée par 50 personnes quotidiennement" ou "Réduction du temps de chargement de 40% grâce à l'optimisation des requêtes".

• L'aspect visuel : Pour les profils Front-end ou Fullstack, le design compte. Un projet visuellement propre donne immédiatement une impression de professionnalisme.

Étape 4 : La structure type du CV Tech efficace

Voici l'ordre recommandé pour structurer tes informations de manière percutante :

• En-tête : Nom, poste visé, lien GitHub, lien LinkedIn, email, téléphone.

• Accroche : 2-3 lignes résumant ton expertise et tes objectifs.

• Stack Technique : Langages, Frameworks, Outils (Git, Docker, etc.).

• Expériences / Projets : En utilisant la méthode Action + Contexte + Résultat.

• Formation : Diplômes et certifications pertinentes (Cloud, Sécurité, etc.).

LE CONSEIL PRO : Ne te contente pas de dire que tu connais un langage. Dans ton CV, crée un lien hypertexte direct vers le code source de ton meilleur projet correspondant à cette technologie. Faciliter le travail du recruteur technique, c'est déjà marquer des points.

Chapitre 3

LinkedIn : Devenir une Cible de Choix pour les Chasseurs de Tête

Module : LinkedIn : Devenir une Cible de Choix pour les Chasseurs de Tête

Dans l'écosystème Tech, LinkedIn n'est pas un simple CV en ligne, c'est un moteur de recherche. Pour être contacté sans postuler, vous devez optimiser votre profil afin d'apparaître en tête des résultats des recruteurs.

Étape 1 : Configurer un Profil "Aimant à Recruteurs"

• La Photo de Profil : Utilisez une photo professionnelle, lumineuse, avec un fond neutre. Évitez les photos de vacances recadrées.

• La Bannière : Ne laissez pas le fond bleu par défaut. Personnalisez-la avec une image évoquant votre univers technique (code, architecture cloud, setup hardware) ou votre logo personnel.

• Le Titre (Headline) : C’est l'élément le plus important. Ne mettez pas juste "Développeur". Utilisez la structure : [Poste Actuel/Cible] | [Technologies Clés] | [Valeur Ajoutée].

• Exemple : Développeur Fullstack Senior | React, Node.js, AWS | Expert en Scalabilité Web.

Étape 2 : Le Référencement (SEO) et les Mots-Clés Techniques

Les chasseurs de tête utilisent des filtres précis. Voici comment être visible sur leurs radars :

• Section "Infos" (Résumé) : Rédigez un paragraphe qui résume vos années d'expérience, vos projets majeurs et vos stacks technologiques de prédilection.

• Détail des Expériences : Pour chaque poste, ne vous contentez pas d'une liste de tâches. Listez les technologies utilisées, les méthodologies (Agile, Scrum) et surtout des résultats chiffrés (ex: "Optimisation du temps de chargement de 40%").

• Compétences (Skills) : Ajoutez les 50 compétences autorisées. Priorisez les langages, frameworks et outils les plus demandés du marché actuel.

• Mots-clés stratégiques : Intégrez des termes comme "Architecture Microservices", "CI/CD", "DevOps" ou "Unit Testing" pour augmenter votre score de pertinence.

Étape 3 : Activer le Mode "Open to Work" (Stratégique)

• Le Badge Vert : Vous pouvez l'afficher pour tous ou seulement pour les recruteurs. Pour une recherche discrète, activez la visibilité uniquement auprès des recruteurs hors de votre entreprise actuelle.

• Préférences de Recherche : Remplissez avec soin les lieux de travail (Remote, Hybride), les titres de postes visés et les types de contrats (CDI, Freelance).

Étape 4 : Techniques de Networking pour l'Accès au "Marché Caché"

Le but est d'obtenir des entretiens via des recommandations, contournant ainsi le processus de candidature classique.

• Demander des Recommandations : Sollicitez vos anciens collègues ou managers pour qu'ils rédigent un avis sur votre profil. La preuve sociale est un facteur de réassurance massif pour un chasseur de tête.

• Interagir intelligemment : Commentez les posts techniques de leaders d'opinion ou d'entreprises cibles. Cela augmente votre visibilité algorithmique.

• L'Approche Directe (Soft Networking) : Ne demandez pas de travail directement. Contactez un développeur de l'entreprise visée pour un "café virtuel" afin de poser des questions sur la stack technique et la culture d'entreprise.

• La Cooptation : Une fois le contact établi, demandez naturellement : "Est-ce que ton équipe recrute en ce moment ? Serait-il possible de passer par ton lien de cooptation ?"

LE CONSEIL PRO :

Utilisez l'outil "LinkedIn Social Selling Index" (SSI). C'est un score gratuit donné par LinkedIn qui mesure votre efficacité sur la plateforme. Un score élevé (au-dessus de 60) garantit que vos publications et votre profil sont mis en avant prioritairement auprès des recruteurs de votre secteur.

Chapitre 4

Dompter le Screening RH : L'Art du Premier Contact

Dompter le Screening RH : L'Art du Premier Contact

Le screening RH est la première porte d'entrée dans le processus de recrutement. Souvent court (15 à 30 minutes), son but n'est pas d'évaluer votre génie technique, mais de vérifier la cohérence de votre parcours, votre motivation et votre adéquation culturelle avec l'entreprise.

Étape 1 : Forger un "Elevator Pitch" Percutant

L'entretien commence presque toujours par : "Présentez-vous". Vous devez être capable de résumer votre valeur ajoutée en 90 secondes maximum.

• Le Passé (Qui êtes-vous ?) : Votre intitulé de poste actuel, votre spécialité technique (ex: Fullstack JS, DevOps Cloud) et vos années d'expérience.

• Le Présent (Vos réalisations) : Mentionnez un projet concret ou une stack technologique maîtrisée qui résonne avec l'annonce.

• Le Futur (Pourquoi eux ?) : Pourquoi vous postulez maintenant et comment vous allez aider l'entreprise à atteindre ses objectifs.

• L'Accroche : Terminez par une question ou une ouverture pour inviter au dialogue.

Étape 2 : Répondre à la Question des Prétentions Salariales

C'est souvent le moment le plus redouté. L'objectif du RH est de vérifier que vous entrez dans le budget prévu pour le poste.

• Faites vos devoirs : Renseignez-vous sur les salaires du marché via des plateformes comme Glassdoor, Figures ou les études de cabinets de recrutement Tech.

• Donnez une fourchette : Ne donnez jamais un chiffre fixe. Proposez une plage de 5 000 € à 10 000 € d'amplitude (ex: "Je vise entre 45k€ et 50k€ selon les avantages").

• Parlez en "Package" : Rappelez que vous considérez l'offre globale (primes, télétravail, BSPCE, mutuelle, matériel) et pas seulement le fixe.

• La botte secrète : Si vous ne voulez pas donner de chiffre immédiatement, dites : "Je préfère d'abord comprendre l'étendue des responsabilités avant de définir une attente précise".

Étape 3 : Réussir l'Entretien Téléphonique de 15 Minutes

Ce format ultra-court nécessite une préparation logistique et mentale impeccable pour passer à l'étape technique.

• L'environnement : Isolez-vous dans un endroit calme avec une excellente réception réseau. Évitez les kits mains libres de mauvaise qualité.

• Le matériel : Ayez sous les yeux votre CV, l'offre d'emploi et un carnet de notes.

• L'attitude : Souriez en parlant ! Cela s'entend au téléphone et projette une image dynamique et positive.

• Les réponses courtes : Le temps est compté. Soyez factuel, évitez les monologues et laissez le recruteur poser ses questions.

• La validation des "Must-haves" : Confirmez clairement votre disponibilité, votre zone de mobilité et votre niveau d'anglais si nécessaire.

Étape 4 : Conclure et Préparer la Suite

Ne terminez jamais l'appel sans savoir ce qu'il se passe après. C'est ici que vous

montrez votre professionnalisme.

• Posez une question pertinente : Demandez par exemple comment est structurée l'équipe technique ou quels sont les prochains défis du département.

• Demandez le process : "Quelles sont les prochaines étapes et sous quel délai puis-je espérer un retour ?"

• Le mail de remerciement : Envoyez un court email dans les 2 heures suivant l'appel pour confirmer votre intérêt et résumer un point clé discuté.

LE CONSEIL PRO : Le recruteur RH est votre meilleur allié. S'il vous apprécie, il "vendra" votre profil au Manager Technique avec enthousiasme. Ne le voyez pas comme un obstacle, mais comme un partenaire à qui vous devez faciliter le travail de sélection.

Chapitre 5

Recherche de Guerre : Scanner l'Entreprise et sa Stack

Module : Recherche de Guerre — Scanner l'Entreprise et sa Stack

En entretien tech, la différence entre un candidat "moyen" et un candidat "exceptionnel" réside souvent dans la préparation stratégique. Ce module vous enseigne comment infiltrer l'écosystème d'une entreprise pour comprendre ses secrets avant même d'avoir franchi la porte.

Étape 1 : Analyser les fondations Business

Avant de plonger dans le code, vous devez comprendre la raison d'être de l'entreprise. Un développeur qui comprend le business est un développeur qui apporte de la valeur.

• Le Business Model : Comment l'entreprise gagne-t-elle de l'argent ? Est-ce du SaaS, de l'E-commerce, ou de la prestation de service ?

• La phase de croissance : S'agit-il d'une startup en Early Stage (besoin de vitesse), d'une Scale-up (besoin de structure) ou d'un Grand Groupe (besoin de stabilité) ?

• L'actualité récente : Consultez les dernières levées de fonds ou les nouveaux marchés conquis sur des sites comme Crunchbase ou Maddyness.

Étape 2 : Décoder la Stack Technique (L'espionnage légal)

Votre objectif est de dresser une carte précise des technologies utilisées pour ne jamais être pris au dépourvu lors des tests techniques.

• Utilisez StackShare : Recherchez l'entreprise sur stackshare.io pour voir leur liste

d'outils déclarés (langages, bases de données, serveurs).

• L'extension Wappalyzer : Naviguez sur leur site web avec cette extension active. Elle vous révélera les frameworks Front-end, les outils de tracking et les serveurs utilisés.

• Analysez les offres d'emploi : Relisez attentivement les annonces passées de l'entreprise. Elles mentionnent souvent des outils secondaires (CI/CD, monitoring, tests unitaires) absents de l'annonce principale.

• Explorez GitHub : Cherchez l'organisation de l'entreprise sur GitHub. Regardez leurs projets Open Source et les langages les plus fréquents dans leurs dépôts publics.

Étape 3 : Infiltrer la Culture "Engineering"

Comprendre comment l'équipe travaille est aussi important que de connaître les langages utilisés.

• Le Blog Technique : De nombreuses entreprises (comme Malt, Alan ou Doctolib) partagent leurs défis techniques sur Medium ou un blog dédié. C'est une mine d'or pour citer des exemples précis en entretien.

• Profils LinkedIn des Leads : Regardez le parcours des CTO et des Engineering Managers. S'ils viennent de boîtes portées sur la "Clean Architecture", attendez-vous à des questions sur la qualité du code.

• La méthodologie : Cherchez des indices sur leur organisation. Sont-ils en Agile/Scrum, en Kanban, ou pratiquent-ils le Pair Programming ?

Étape 4 : Identifier les Défis et "Pain Points"

Le but ultime de votre recherche est d'anticiper les problèmes que l'entreprise essaie de résoudre en vous recrutant.

• Le passage à l'échelle (Scalability) : Si l'entreprise explose en nombre d'utilisateurs, leur défi sera la performance et la gestion de la charge.

• La dette technique : Une entreprise qui migre d'un monolithe vers des microservices cherche des profils capables de gérer cette transition complexe.

• La sécurité : Pour une Fintech ou une Healthtech, la protection des données est le défi numéro un. Préparez vos arguments sur ce sujet.

LE CONSEIL PRO : Ne vous contentez pas de stocker ces informations, utilisez-les comme des questions. Au lieu de demander "Quelle est votre stack ?", dites : "J'ai vu sur votre blog technique que vous migriez vers Kubernetes pour améliorer votre déploiement, comment gérez-vous cette transition au sein de l'équipe mobile ?". C'est le meilleur moyen de prouver votre proactivité et votre intérêt réel.

Chapitre 6

Le Live Coding sans Stress : Méthodes et Astuces

Le Live Coding sans Stress : Méthodes et Astuces

L'exercice du Live Coding n'est pas un examen de dactylographie, mais une fenêtre ouverte sur votre manière de réfléchir et de collaborer. Voici comment transformer cette épreuve stressante en une démonstration de force tranquille.

Étape 1 : Maîtriser la méthode du "Think Aloud" (Penser à voix haute)

Le recruteur ne peut pas lire dans votre esprit. Verbaliser votre raisonnement est crucial pour qu'il puisse vous aider et valider votre logique.

• Expliquez vos intentions avant de taper la moindre ligne de code (ex: "Je vais commencer par vérifier si la liste est vide").

• Justifiez vos choix technologiques (ex: "J'utilise un dictionnaire ici pour garantir une recherche en temps constant").

• Avouez vos hésitations : il vaut mieux dire "Je réfléchis à la meilleure structure ici" que de rester silencieux pendant deux minutes.

• Utilisez le "Nous" pour inclure le recruteur dans votre démarche de résolution de problème.

Étape 2 : Adopter une structure de résolution itérative

Ne vous lancez pas tête baissée dans l'écriture du code final. Suivez une méthodologie par étapes pour éviter de vous perdre.

• Compréhension et Clarification : Reformulez le problème et posez des questions sur les cas particuliers (ex: "Que se passe-t-il si l'entrée est nulle ?").

• Le Pseudo-code : Écrivez la logique en français ou avec des commentaires avant de coder. C'est votre feuille de route.

• La Solution Brute (Brute Force) : Proposez d'abord une solution qui fonctionne, même si elle n'est pas optimale. Il est plus facile d'optimiser un code qui tourne.

• Optimisation : Améliorez la performance (complexité temporelle et spatiale) une fois que la logique de base est validée.

Étape 3 : Gérer la panique face à un bug en direct

Le bug n'est pas un échec, c'est une opportunité de montrer vos capacités de débogage et votre résilience.

• Restez calme : Prenez une respiration profonde. Un développeur qui panique perd ses moyens ; un développeur qui analyse reste professionnel.

• Expliquez le bug : Dites ce que vous observez (ex: "Je m'attendais à recevoir un entier, mais j'obtiens undefined").

• Utilisez des "Print" ou le Debugger : Ne devinez pas, vérifiez vos hypothèses en affichant les valeurs intermédiaires.

• Sollicitez un indice : Si vous bloquez plus de 5 minutes, demandez poliment : "J'explore cette piste, mais je sens que je passe à côté de quelque chose, auriez-vous un petit indice ?".

Étape 4 : Finaliser et tester sa solution

Ne dites jamais "J'ai fini" sans avoir vérifié votre travail. La rigueur est une qualité très recherchée.

• Relisez votre code à voix haute pour traquer les fautes de frappe ou les erreurs de logique évidentes.

• Testez avec un exemple simple : Faites défiler manuellement une donnée d'entrée à travers vos fonctions.

• Vérifiez les "Edge Cases" : Testez mentalement votre code avec une liste vide, un seul élément ou des valeurs extrêmes.

LE CONSEIL PRO : Considérez le recruteur comme un collègue avec qui vous essayez de résoudre un ticket technique, plutôt que comme un juge. Si vous rendez l'échange collaboratif et agréable, vous marquez des points même si votre code n'est pas parfait à 100 %.

Chapitre 7

Le Test Technique à la Maison : Vise la Perfection

Le Test Technique à la Maison : Vise la Perfection

Le test technique à la maison n’est pas seulement un exercice de codage. C’est une simulation de votre futur quotidien en entreprise. Les correcteurs ne regardent pas uniquement si le code fonctionne, mais surtout comment vous l'avez construit.

Étape 1 : Adopter les principes du Clean Code

Votre code doit être lisible par un humain avant d'être exécutable par une machine. Un code "propre" montre que vous êtes un développeur organisé et rigoureux.

• Nommage explicite : Bannissez les noms de variables vagues comme data ou x. Utilisez des noms qui décrivent l'intention, comme userList ou calculateTotalPrice.

• Fonctions à responsabilité unique : Une fonction ne doit faire qu'une seule chose. Si elle dépasse 20 lignes, elle est probablement trop complexe.

• DRY (Don't Repeat Yourself) : Évitez les copier-coller. Si une logique se répète, extrayez-la dans une fonction ou une classe réutilisable.

• Commentaires utiles : Ne commentez pas ce que fait le code (le code doit être explicite), mais pourquoi vous avez fait ce choix technique particulier.

Étape 2 : Une gestion Git digne d'un professionnel

L'historique de vos commits raconte l'histoire de votre réflexion. Un projet avec un seul commit final "Initial commit" est un red flag pour les recruteurs.

• Commits atomiques : Faites des petits commits réguliers. Chaque commit doit représenter une unité de travail logique (ex: "Ajout du service d'authentification").

• Messages clairs : Utilisez des messages explicites au présent. Évitez les "fix", "update" ou "test". Préférez "Ajout de la validation du formulaire de contact".

• Structure propre : Ne poussez jamais de fichiers inutiles (fichiers de configuration locale, dossiers node_modules, etc.). Utilisez un fichier .gitignore impeccable.

Étape 3 : La Documentation (Le README est votre vitrine)

Le README est le premier fichier que le correcteur va lire. S'il ne parvient pas à lancer votre projet en 2 minutes, votre note chutera drastiquement.

• Introduction : Présentez brièvement le projet et les technologies utilisées.

• Installation et Lancement : Listez chaque commande nécessaire pour cloner, installer les dépendances et démarrer l'application.

• Choix techniques : Expliquez pourquoi vous avez choisi telle bibliothèque plutôt qu'une autre. Cela montre votre capacité d'analyse.

• Améliorations possibles : Listez ce que vous auriez fait avec plus de temps (optimisation de performance, fonctionnalités bonus).

Étape 4 : Les Tests Unitaires pour rassurer

Un code sans test est considéré comme "cassé" par défaut dans de nombreuses équipes tech. Les tests prouvent que votre solution est fiable et maintenable.

• Couverture des cas critiques : Ne testez pas tout, mais concentrez-vous sur la logique métier principale.

• Gestion des cas limites (Edge Cases) : Montrez que vous avez pensé aux erreurs (ex: que se passe-t-il si l'API renvoie une erreur ou si l'utilisateur saisit un texte vide ?).

• Lisibilité des tests : Un test doit servir de documentation. On doit comprendre ce que fait votre application juste en lisant les titres de vos tests.

LE CONSEIL PRO : Ne cherchez pas à en faire trop avec des bibliothèques complexes que vous ne maîtrisez pas. Il vaut mieux un code simple, parfaitement testé et documenté, qu'une architecture complexe truffée de bugs et de raccourcis techniques. La clarté gagne toujours sur la sophistication.

Chapitre 8

Algorithmes et Logique : Décrypter l'Indécryptable

Algorithmes et Logique : Décrypter l'Indécryptable

Dans un entretien technique, l'objectif du recruteur n'est pas seulement de vérifier si vous savez coder, mais surtout de comprendre votre manière de réfléchir face à un problème complexe.

Étape 1 : Maîtriser les Structures de Données Essentielles

Avant de résoudre un algorithme, vous devez choisir le bon outil. Voici les structures incontournables à connaître :

• Les Tableaux (Arrays) : Parfaits pour stocker des listes simples. L'accès à un élément via son index est instantané, mais insérer un élément au début peut être coûteux en temps.

• Les Listes Chaînées (Linked Lists) : Idéales pour des insertions ou suppressions fréquentes. Chaque élément pointe vers le suivant.

• Les Dictionnaires / Tables de Hachage (Hash Maps) : L'outil le plus puissant en entretien. Il permet de stocker des paires "clé-valeur" pour retrouver une information quasi immédiatement.

• Les Piles et Files (Stacks & Queues) : Utiles pour gérer l'ordre de traitement (Premier entré, dernier sorti ou inversement).

• Les Arbres (Trees) : Notamment l'arbre binaire de recherche, essentiel pour organiser des données de manière hiérarchique et accélérer les recherches.

Étape 2 : Comprendre la Complexité (La Notation Big O)

La notation Big O permet de mesurer l'efficacité d'un algorithme en fonction de la quantité de données à traiter.

• O(1) - Complexité Constante : Le temps d'exécution reste le même, peu importe la taille des données (ex: accéder au premier élément d'un tableau).

• O(n) - Complexité Linéaire : Le temps augmente proportionnellement au nombre d'éléments (ex: parcourir une liste pour trouver une valeur).

• $O(n^2)$ - Complexité Quadratique : Souvent le résultat de boucles imbriquées. À éviter sur de grandes quantités de données car les performances chutent lourdement.

• O(log n) - Complexité Logarithmique : La stratégie "diviser pour régner" (ex: recherche dichotomique). C'est le Graal de l'optimisation.

Étape 3 : S'entraîner sur des Exercices Types

Certains problèmes reviennent systématiquement sous différentes formes. Entraînez-vous à résoudre ceux-ci :

• Le FizzBuzz : Énumérer des nombres en remplaçant certains par des mots selon des règles de division. Teste la maîtrise des conditions de base.

• Inversion de chaîne : Inverser "Hello" en "olleH" sans utiliser de fonction préfabriquée. Teste la manipulation d'index.

• La recherche de doublons : Trouver si un nombre apparaît deux fois dans une liste. C'est l'exercice parfait pour utiliser un Dictionnaire (Hash Map).

• Le Palindrome : Vérifier si un mot se lit dans les deux sens. Teste la logique de comparaison symétrique.

Étape 4 : La Méthodologie du "Tableau Blanc"

Même si vous bloquez sur le code, la méthode suivante vous sauvera dans 90% des cas :

• Clarifiez l'énoncé : Posez des questions. "Les nombres sont-ils entiers ?", "La liste peut-elle être vide ?".

• Expliquez votre stratégie à voix haute : Ne restez pas silencieux. Le recruteur veut entendre votre cheminement intellectuel.

• Écrivez un pseudo-code : Structurez votre logique en français (ou anglais) avant de toucher au clavier.

• Codez la solution simple : Ne cherchez pas l'optimisation immédiate. Faites d'abord quelque chose qui marche.

• Optimisez : Une fois la solution trouvée, demandez-vous comment réduire la complexité Big O.

LE CONSEIL PRO : Ne dites jamais "Je ne sais pas". Si vous êtes bloqué, proposez une solution "brute" (même inefficace) et expliquez pourquoi elle n'est pas optimale. Les recruteurs valorisent l'honnêteté intellectuelle et la capacité à rebondir bien plus qu'une réponse apprise par cœur.

Chapitre 9

Architecture et Système : Penser comme un Lead Dev

Module : Architecture et Système : Penser comme un Lead Dev

Lors d'un entretien tech, on ne juge pas seulement votre capacité à écrire du code, mais votre aptitude à concevoir des systèmes entiers. Passer de développeur à Lead Dev, c'est arrêter de regarder uniquement la ligne de code pour regarder la structure globale du bâtiment.

Étape 1 : Comprendre la Scalabilité (La croissance sans la casse)

La scalabilité est la capacité d'une application à gérer une augmentation soudaine d'utilisateurs sans s'effondrer. Il existe deux manières principales de l'aborder :

• La Scalabilité Verticale : Cela consiste à ajouter de la puissance (CPU, RAM) à une machine existante. C'est simple, mais limité physiquement par la taille du serveur.

• La Scalabilité Horizontale : C'est l'approche privilégiée par les géants du web. Au lieu d'un gros serveur, on utilise plusieurs petits serveurs travaillant en parallèle.

• Le Load Balancer (Équilibreur de charge) : C'est le "chef d'orchestre" indispensable qui répartit le trafic entrant entre vos différents serveurs pour éviter qu'un seul ne soit surchargé.

Étape 2 : Choisir le bon stockage de données

Le choix de la base de données définit la solidité de votre infrastructure. On distingue généralement deux grandes familles :

• SQL (Bases relationnelles) : Idéal pour les données structurées et les transactions

complexes (ex: un système bancaire, un inventaire e-commerce). C'est la garantie de la cohérence.

• NoSQL (Bases non-relationnelles) : Parfait pour la flexibilité et les volumes de données massifs (ex: fils d'actualités, messagerie instantanée, logs). C'est la garantie de la rapidité et de la souplesse.

• La mise en cache (Redis) : Pour booster les performances, on utilise une "mémoire flash" qui stocke les résultats fréquents afin de ne pas solliciter la base de données principale inutilement.

Étape 3 : Dessiner une architecture web robuste

Pour réussir l'exercice du tableau blanc en entretien, vous devez savoir dessiner le flux d'une requête. Voici le schéma type d'une architecture moderne :

• Le Client : L'utilisateur sur son navigateur ou son smartphone.

• Le DNS : Le carnet d'adresses qui traduit le nom de domaine en adresse IP.

• Le CDN (Content Delivery Network) : Des serveurs partout dans le monde qui livrent les images et fichiers statiques au plus proche de l'utilisateur.

• API Gateway / Load Balancer : Le point d'entrée unique qui sécurise et oriente les requêtes.

• Services / Microservices : Les blocs de logique métier qui traitent l'information.

• Base de données : Le lieu de stockage final, souvent séparé en "Lecture" et "Écriture" pour plus de performance.

Étape 4 : Adopter la méthodologie du Lead Dev en entretien

Face à un problème d'architecture (ex: "Concevez un système comme Twitter"), ne commencez jamais par dessiner. Suivez cet ordre :

• Clarifier les besoins : Posez des questions sur le nombre d'utilisateurs actifs, le volume de données et les fonctionnalités prioritaires.

• Définir les APIs : Listez les points d'entrée principaux (ex: POST /tweet, GET /timeline).

• Schématiser l'ensemble : Tracez les boîtes (Serveurs, Bases de données, Cache) et reliez-les par des flèches.

• Identifier les goulots d'étranglement : Expliquez ce qui se passe si le trafic multiplie par 100 et proposez des solutions (plus de cache, division de la base de données).

LE CONSEIL PRO : En entretien de System Design, il n'y a jamais une seule "bonne" réponse. Ce que le recruteur évalue, c'est votre capacité à justifier vos compromis (Trade-offs). Expliquez toujours pourquoi vous choisissez une solution plutôt qu'une autre (ex: "Je privilégie la rapidité de lecture ici, quitte à perdre un peu de fraîcheur de donnée pendant quelques secondes").

Chapitre 10

Soft Skills : Pourquoi ton Attitude bat ton Code

Soft Skills : Pourquoi ton Attitude bat ton Code

Dans le monde de la tech, savoir coder est le minimum syndical. Ce qui fera la différence entre toi et un autre candidat avec le même CV, c'est ta capacité à t'intégrer dans une équipe et à faire grandir le projet.

Voici comment démontrer que tu n'es pas juste une "machine à coder", mais un atout stratégique pour l'entreprise.

1. L'Empathie Technique : Parler "humain"

L'empathie technique est la capacité à comprendre que ton code sera lu par d'autres et que ton produit sera utilisé par des non-techniciens.

• Vulgarisation : Explique des concepts complexes sans utiliser de jargon inutile pour montrer que tu peux communiquer avec le marketing ou le design.

• Code lisible : Précise que tu écris du code "pour les humains" et pas seulement pour la machine (nommage clair, commentaires utiles).

• Focus utilisateur : Montre que tu te demandes toujours : "Est-ce que cette fonctionnalité aide vraiment l'utilisateur final ?" au lieu de te concentrer uniquement sur la performance brute.

2. Travailler en Équipe Agile : Le "Nous" avant le "Je"

Les recruteurs cherchent des joueurs d'équipe, pas des "rockstars" isolées qui cassent l'ambiance du bureau.

• Transparence : Explique que tu n'hésites pas à dire "je ne sais pas" ou "je suis bloqué" lors d'un Stand-up meeting pour ne pas ralentir l'équipe.

• Collaboration : Mentionne ton goût pour le Pair Programming ou la conception collaborative.

• Ego-less Programming : Démontre que ton objectif est la réussite du sprint, même si cela signifie travailler sur une tâche moins "glamour" pour aider un collègue.

3. Réception Constructive des Feedbacks

La revue de code (Code Review) est le moment où ton attitude est la plus visible. Un candidat sur la défensive est un "red flag" immédiat.

• Détachement : Montre que tu ne confonds pas une critique sur ton code avec une critique sur ta personne.

• Gratitude : Remercie systématiquement pour les retours. Un feedback est une opportunité d'apprentissage gratuite.

• Argumentation saine : Si tu n'es pas d'accord, explique ton choix de manière factuelle et reste ouvert à changer d'avis si l'autre argument est meilleur.

4. La Curiosité Intellectuelle : Plus loin que le framework

Le secteur évolue plus vite que n'importe quel autre. Ta capacité à apprendre est plus précieuse que ce que tu sais déjà aujourd'hui.

• Veille active : Parle des blogs, podcasts ou newsletters que tu suis (ex: TechCrunch, Medium, Documentation officielle).

• Expérimentation : Mentionne tes projets personnels ou les nouvelles technologies que tu testes par simple plaisir de comprendre.

• Esprit critique : Ne te contente pas de copier-coller une solution sur Stack Overflow

; montre que tu cherches à comprendre le "pourquoi" derrière le bug.

LE CONSEIL PRO : Lors de l'entretien, utilise la méthode STARR (Situation, Tâche, Action, Résultat, Réflexion) pour raconter une fois où tu as reçu un feedback difficile ou aidé un collègue en difficulté. Les histoires concrètes prouvent tes soft skills bien mieux que les grands discours théoriques.

Chapitre 11

L'Entretien 'Culture Fit' : Devenir l'Évidence

Introduction : Au-delà du code, l'humain

Dans l'univers de la Tech, vos compétences techniques (Hard Skills) vous permettent d'obtenir l'entretien, mais c'est votre adéquation culturelle (Culture Fit) qui vous fera décrocher le poste. L'objectif de cet entretien n'est pas de juger si vous êtes "sympa", mais de déterminer si vos valeurs et votre méthode de travail sont en phase avec l'équipe déjà en place.

Étape 1 : Décoder et s'aligner sur les valeurs de l'entreprise

Avant l'entretien, vous devez mener une véritable enquête pour comprendre l'ADN de la société. Une entreprise qui prône la "vélocité" n'attend pas la même chose qu'une entreprise misant sur la "stabilité".

• Analysez le site carrière : Repérez les mots-clés récurrents (ex: "Bienveillance", "Ownership", "Radical Candor").

• Consultez le "Culture Deck" : De nombreuses startups publient un document détaillant leur philosophie de travail.

• Identifiez les piliers : Est-ce une culture basée sur la hiérarchie ou sur l'autonomie totale ?

• Préparez votre narration : Pour chaque valeur identifiée, trouvez un moment de votre carrière où vous l'avez mise en pratique.

Étape 2 : Maîtriser l'art de raconter la résolution de conflits

Le recruteur veut savoir comment vous réagissez quand les choses tournent mal. Un désaccord sur une architecture technique ou une priorité de ticket est inévitable.

• Utilisez la méthode STAR : Structurez vos réponses par Situation, Tâche, Action et Résultat.

• Restez factuel : Ne critiquez jamais vos anciens collègues ou managers. Concentrez-vous sur le problème technique ou organisationnel, pas sur les personnes.

• Montrez votre capacité d'écoute : Expliquez comment vous avez tenté de comprendre le point de vue opposé avant de proposer un compromis.

• Valorisez la conclusion : Insistez sur ce que vous avez appris de cette situation pour ne pas la reproduire.

Étape 3 : Prouver que vous êtes un excellent coéquipier (Teammate)

Dans la Tech, personne ne travaille seul. Votre capacité à intégrer une équipe de développeurs, de Product Managers et de Designers est cruciale.

• Parlez de partage de connaissances : Mentionnez vos expériences de Pair Programming, de rédaction de documentation ou de mentorat de profils plus juniors.

• Démontrez votre humilité : Admettez que vous ne savez pas tout et que vous accueillez les Code Reviews comme une opportunité de progresser, et non comme une critique.

• Illustrez votre communication : Donnez des exemples de moments où vous avez vulgarisé un concept complexe pour des profils non-techniques.

• Mettez en avant l'esprit collectif : Utilisez le "Nous" plutôt que le "Je" lorsque vous parlez des succès d'un projet précédent.

Étape 4 : Poser les bonnes questions pour confirmer le "Fit"

L'entretien est à double sens. Poser des questions sur la culture prouve votre intérêt et votre maturité professionnelle.

• Sur le quotidien : "Comment l'équipe gère-t-elle l'équilibre entre la dette technique et la livraison de nouvelles fonctionnalités ?"

• Sur l'évolution : "Comment les feedbacks sont-ils donnés au sein de l'équipe technique ?"

• Sur l'échec : "Que se passe-t-il dans l'entreprise lorsqu'un déploiement en production échoue ?"

LE CONSEIL PRO : Ne jouez pas un rôle. Le "Culture Fit" est une sécurité pour vous aussi. Si vous devez simuler une personnalité qui n'est pas la vôtre pour être embauché, vous risquez d'être malheureux et de faire un burn-out après quelques mois. Soyez la meilleure version professionnelle de vous-même, tout en restant authentique.

Chapitre 12

Inverser les Rôles : Les Questions qui Impressionnent

Pourquoi poser des questions est votre meilleur atout

L’entretien d'embauche n’est pas un interrogatoire, c’est une conversation bidirectionnelle. Poser des questions pertinentes permet d’atteindre deux objectifs majeurs :

• Évaluer l'entreprise : Déterminer si l'environnement technique et humain vous convient réellement.

• Démontrer votre expertise : Montrer que vous ne vous contentez pas de coder, mais que vous comprenez les enjeux de production et de qualité.

Étape 1 : Sonder la "Santé Technique" auprès des développeurs

Ces questions s'adressent directement à vos futurs pairs. Elles visent à découvrir la réalité du terrain derrière les promesses marketing.

• "Comment gérez-vous la dette technique au quotidien ?" : Une excellente question pour savoir si l'équipe a le temps de refactoriser ou si elle court sans cesse après les fonctionnalités.

• "Quel est votre processus de Code Review et quels critères privilégiez-vous ?" : Cela montre que vous accordez de l'importance à la qualité logicielle et à l'apprentissage collectif.

• "Combien de temps s'écoule entre un 'commit' et la mise en production ?" : Cette question évalue la maturité de leur CI/CD et l'efficacité de leur pipeline d'automatisation.

• "Quelle est la politique de tests (unitaires, intégration, end-to-end) sur le projet ?" : Indispensable pour savoir si vous allez travailler dans un environnement sécurisé ou risqué.

Étape 2 : Évaluer le management et la vision produit

Ces questions sont destinées au CTO, Engineering Manager ou Lead Dev. Elles portent sur l'organisation et votre évolution future.

• "Comment sont arbitrées les priorités entre les besoins produit et les impératifs techniques ?" : Cela permet de comprendre l'équilibre des forces entre les Product Managers et les développeurs.

• "Pouvez-vous me décrire une erreur technique majeure récente et comment l'équipe l'a gérée ?" : Une question puissante pour tester la culture du "Blameless Post-mortem" (recherche de solution sans pointer du doigt).

• "Quels sont les parcours de carrière possibles pour un profil technique chez vous ?" : Montre que vous vous projetez sur le long terme et que vous avez de l'ambition.

• "Comment se déroule le processus d'onboarding technique pour une nouvelle recrue ?" : Évalue le soin apporté à l'accueil et à la formation des collaborateurs.

Étape 3 : Poser les questions "Expert" pour briller

Pour vous démarquer, posez des questions qui touchent aux enjeux de haut niveau (architecture, scalabilité, sécurité).

• "Quels sont les plus gros défis de scalabilité auxquels vous faites face actuellement ?" : Positionne votre profil sur une vision système globale plutôt que purement syntaxique.

• "Pourquoi avoir choisi [Techno X] plutôt que [Techno Y] pour ce projet spécifique ?" : Démontre que vous comprenez que les choix technologiques sont des compromis

(trade-offs).

• "Comment assurez-vous la sécurité et la conformité des données dans votre architecture ?" : Un point critique souvent oublié qui prouve votre professionnalisme et votre rigueur.

Étape 4 : Les questions "Culture & Collaboration"

Parce que la technique ne fait pas tout, il faut valider l'adéquation humaine.

• "Comment l'équipe communique-t-elle au quotidien (synchrone vs asynchrone) ?" : Crucial si vous travaillez en télétravail ou sur plusieurs fuseaux horaires.

• "Quelle est la place accordée à la veille technologique et au partage de connaissances ?" (BBL, conférences, formations internes).

• "Qu'est-ce qui retient les développeurs chez vous sur le long terme ?" : Une question ouverte qui en dit long sur l'ambiance et la rétention des talents.

LE CONSEIL PRO : Ne posez pas toutes ces questions d'un coup. Sélectionnez-en 3 ou 4 en fonction de l'interlocuteur. Écoutez attentivement les réponses et rebondissez dessus : la capacité d'écoute est une soft skill aussi recherchée que la maîtrise d'un langage de programmation.

Chapitre 13

Négociation de Salaire : Valoriser son Talent

Module : Négociation de Salaire – Valoriser son Talent

La négociation salariale est souvent perçue comme le moment le plus stressant du processus de recrutement. Pourtant, dans la Tech, c'est une discussion standard où la préparation et la connaissance de sa propre valeur font toute la différence.

Étape 1 : Connaître le marché pour fixer son prix

Avant de donner un chiffre, vous devez savoir ce que vous "valez" réellement sur le marché actuel. Ne vous basez pas sur votre salaire actuel, mais sur la valeur du poste visé.

• Consultez des grilles de salaires spécialisées (ex: Figures, Silkhom, ou les rapports annuels de cabinets de recrutement Tech).

• Utilisez des plateformes comme Glassdoor ou Levels.fyi pour comparer les rémunérations selon la taille de l'entreprise.

• Échangez avec vos pairs (développeurs, PM, designers) pour obtenir des retours concrets sur les packages pratiqués.

• Définissez une fourchette de négociation : votre "prix idéal" et votre "seuil critique" en dessous duquel vous ne descendrez pas.

Étape 2 : Négocier le "Package Global" (Avantages Tech)

Le salaire fixe n'est qu'une partie de l'équation. Dans le secteur Tech, les avantages peuvent représenter une plus-value financière et de confort considérable.

• Le Télétravail (Remote) : Négociez le nombre de jours par semaine ou un statut "full remote" qui peut vous faire économiser des frais de transport et de logement.

• Les BSPCE (Stock-Options) : Très courants en startup, ils vous permettent de devenir actionnaire. Demandez la valorisation actuelle et les conditions de "vesting" (temps de présence requis).

• Le Matériel (Stack technique) : Un développeur performant a besoin d'outils de qualité. Vous pouvez demander un budget équipement spécifique (MacBook Pro, écran 4K, chaise ergonomique).

• Formation et Conférences : Demandez un budget annuel pour passer des certifications ou assister à des événements majeurs (Devoxx, AWS Summit, etc.).

Étape 3 : La psychologie de la négociation "Gagnant-Gagnant"

L'objectif est d'obtenir ce que vous voulez sans paraître arrogant ou rigide. Le recruteur doit sentir que vous êtes un partenaire, pas un adversaire.

• L'ancrage : Soyez souvent le premier à donner une fourchette haute. Cela définit le point de départ de la discussion en votre faveur.

• Le "Nous" : Utilisez un langage collaboratif. Par exemple : "Comment pouvons-nous nous rapprocher de ce chiffre pour que ce projet soit une réussite pour nous deux ?".

• L'argumentation factuelle : Ne dites pas "Je veux X car j'ai un crédit", mais "Mon expertise sur React et mon expérience en architecture Cloud justifient ce positionnement".

• Le silence : Une fois votre proposition faite, taisez-vous. Laissez le recruteur traiter l'information. Le silence est un outil de négociation puissant.

Étape 4 : Réagir à une offre inférieure à vos attentes

Si l'offre tombe et qu'elle est décevante, ne fermez pas la porte immédiatement. Utilisez la technique de la contre-proposition.

• Remerciez le recruteur pour l'offre et réitérez votre enthousiasme pour le poste.

• Soulevez le point de blocage avec diplomatie : "Le projet me passionne, cependant le salaire proposé est en deçà du marché pour mon profil".

• Proposez des alternatives : "Si le budget fixe est bloqué, pouvons-nous compenser par une prime à la performance ou des jours de repos supplémentaires ?".

• Demandez une clause de révision : Proposez de faire un point sur le salaire après 6 mois ou à la fin de la période d'essai.

LE CONSEIL PRO : Ne donnez jamais un chiffre fixe, donnez toujours une fourchette dont le bas correspond à votre objectif réel. Si vous visez 60k€, annoncez : "Au vu de mes compétences et des responsabilités, je projette une rémunération comprise entre 60k€ et 65k€". Cela laisse une marge de manœuvre au recruteur tout en protégeant votre minimum.

Chapitre 14

L'Après-Entretien : Suivi et Persévérance

L'Après-Entretien : Maximiser vos chances et Capitaliser sur l'expérience

L’entretien ne s’arrête pas au moment où vous quittez la salle ou la visioconférence. La phase de suivi est déterminante pour consolider votre image de candidat professionnel et proactif.

Étape 1 : L'Email de Remerciement Stratégique

L'email de remerciement n'est pas une simple formalité de politesse ; c'est un outil marketing pour réaffirmer votre valeur technique et votre intérêt.

• Le timing : Envoyez votre email dans les 24 heures ouvrées suivant l'entretien.

• L'objet : Soyez clair, par exemple : "Remerciements - Entretien [Nom du Poste] - [Votre Prénom et Nom]".

• Le contenu : Remerciez pour le temps accordé et mentionnez un point technique spécifique abordé durant l'échange (ex: un challenge lié à l'architecture Microservices ou l'usage de React).

• La valeur ajoutée : Si vous avez évoqué un problème technique sans solution immédiate, vous pouvez joindre un lien vers un article ou un repo GitHub pertinent.

• L'appel à l'action : Réitérez votre enthousiasme et précisez que vous restez disponible pour tout test technique complémentaire.

Étape 2 : Gérer l'Attente et les Relances

Savoir patienter sans paraître désespéré est un signe de maturité professionnelle.

• La règle d'or : Ne relancez jamais avant la date limite fixée par le recruteur à la fin de l'entretien.

• La relance proactive : Si vous n'avez pas de nouvelles après 7 jours (ou la date prévue), envoyez un email court pour demander où en est le processus de recrutement.

• La gestion du stress : Continuez à postuler ailleurs. Tant qu'un contrat n'est pas signé, vous êtes toujours en recherche active.

Étape 3 : Demander un Feedback Constructif en cas de Refus

Un refus n'est pas un échec, c'est une source de données pour votre prochain entretien Tech.

• L'état d'esprit : Répondez au mail de refus avec courtoisie et professionnalisme.

• La demande précise : Ne demandez pas "Pourquoi n'ai-je pas été pris ?", mais plutôt "Quels sont les points d'amélioration (techniques ou comportementaux) que vous me conseilleriez pour progresser ?".

• Le focus technique : Si le refus est lié au test technique, demandez si une correction type ou des ressources d'apprentissage sont disponibles.

• L'acceptation : Ne discutez jamais la décision. Remerciez simplement pour les conseils fournis.

Étape 4 : Maintenir le Lien pour le Futur

Le monde de la Tech est petit. Un "non" aujourd'hui peut devenir un "oui" dans six mois pour un autre projet.

• LinkedIn : Ajoutez vos interlocuteurs sur LinkedIn avec un message personnalisé :

"Ravi d'avoir échangé avec vous sur [Sujet Tech], au plaisir de suivre l'évolution de [Nom de l'entreprise]".

• Le vivier de talents : Autorisez l'entreprise à conserver votre CV. Les besoins des startups et des entreprises tech évoluent très rapidement.

• La veille partagée : Si vous publiez des articles techniques ou des projets Open Source, cela permet de rester visible auprès des recruteurs qui vous ont dans leur réseau.

LE CONSEIL PRO : Considérez chaque entretien comme un audit gratuit de vos compétences. Notez immédiatement après chaque échange les questions auxquelles vous avez eu du mal à répondre. Créez un document "FAQ Tech Personnelle" et travaillez ces points pour que, lors du prochain entretien, cette faiblesse soit devenue une force maîtrisée.

FIN

Merci d'avoir lu "Réussir son entretien d'embauche Tech"

Une œuvre écrite par Fusianima Expert

Lire la version interactive et commenter

Découvrir les autres œuvres de l'auteur