Engineer

{ Développement backend et API }

Le moteur de votre application ou de votre site

La partie invisible pour les utilisateurs : où résident les données, quelles règles s'appliquent et comment les autres systèmes peuvent y accéder.

Vous cherchez encore à préciser vos besoins ?

Dites-nous ce qui ne fonctionne pas ou ce que vous essayez de construire. Nous vous dirons ce que cela demanderait, son coût approximatif et si nous sommes les bonnes personnes pour le réaliser.

À partir de 1 400 DKK de l'heure, hors TVA — et un tarif horaire réduit pour une réservation plus longue.

Contactez-nous Consulter les tarifs

Une sélection de réalisations illustrant cette prestation.

Transformez la prestation en résultat

Mettez « Développement backend et API » au service d'un résultat pour votre entreprise. Nous pouvons relier le travail technique aux utilisateurs, aux opérations et aux décisions qu'il doit soutenir. La partie invisible pour les utilisateurs : où résident les données, quelles règles s'appliquent et comment les autres systèmes peuvent y accéder.

Contactez-nous

Examinez les réalisations pertinentes

Consultez les réalisations liées sur cette page pour examiner notre démarche. Interrogez-nous sur les compromis et les conditions qui ont conduit à chaque résultat. 6 pipelines ETL/ELT conçus et exploités — 6 pipelines ETL sur trois moteurs de bases de données

6 pipelines ETL/ELT conçus et exploités Contactez-nous →

6 pipelines ETL/ELT conçus et exploités

Conçu, mis en œuvre et administré 6 pipelines ETL/ELT avec Google BigQuery, MSSQL, PostgreSQL, des scripts Shell, PL/pgSQL et Transact-SQL, en intégrant les données pour leur traitement efficace par une API Python.

8 projets Power BI et leurs manuels

Livré 8 projets Power BI accompagnés de manuels complets, intégrant les outils Microsoft Power BI à une API NodeJS et à Python FastAPI pour l'analyse et la visualisation des données.

Développement cartographique GIS full-stack

Dirigé le développement full-stack d'une application cartographique GIS, avec PostgreSQL, Mapbox, ReactJS et NodeJS, pour livrer une solution intégrée.

Plus de 30 projets GIS, du code à l'assistance

Dirigé le développement, le déploiement et l'assistance de plus de 30 projets GIS, avec PostgreSQL, Bash, Python, JavaScript, GDAL, ArcGIS, PostGIS et Mapbox.

Système de reporting d'analyse de données

Développé un système de reporting d'analyse de données, augmentant de 400 % le chiffre d'affaires logiciel trimestriel grâce à des rapports PDF produits en Python.

Une plateforme maritime aux frontières système claires

Conçu une plateforme maritime en couches séparant un frontend PWA Next.js, une API Go (Huma/Fiber) et une couche de fonctions PostgreSQL, avec toute la logique métier dans la base de données.

JSON transmis directement de Postgres à l'API

Conçu une architecture de transmission directe du JSON où les fonctions PostgreSQL renvoient des documents complets, transmis tels quels par l'API Go, supprimant la désérialisation intermédiaire et découplant le frontend des changements de schéma.

Changement de contexte d'organisation

Conçu un système de changement de contexte d'organisation avec localStorage côté client et réplication dans un cookie côté serveur, permettant d'agir au nom des organisations administrées tout en appliquant le moindre privilège.

Migration vers Huma v2 avec parité de 100 %

Migré l'API HTTP de Fiber vers Huma v2 — 649 chemins et 760 opérations — puis atteint et maintenu une correspondance de 100 % entre les routes enregistrées par le serveur et la description OpenAPI publiée.

Contrat d'API généré à partir de la base

Généré le contrat d'API à partir de la base de données — OpenAPI, client TypeScript typé de 44 076 lignes, 61 gestionnaires de simulation et limites appliquées par l'interface — avec un contrôle à chaque étape qui échoue en cas de divergence.

E-mails avec basculement et journal de livraison

Conçu l'envoi d'e-mails comme une capacité de plateforme — trois fournisseurs avec basculement, webhooks de livraison, journaux d'envoi et de livraison, modèles et campagnes — avec un contrôle au démarrage qui refuse de lancer le service sans fournisseur.

Couche PostgreSQL fondée sur les fonctions

Conçu une couche de données fondée sur les fonctions PostgreSQL — 1 275 fonctions stockées réparties dans 34 schémas — pour que chaque lecture et écriture passe par une fonction dont les droits sont contrôlables, plutôt que par une table.

Fusion profonde pilotée par un catalogue

Mis en œuvre une fusion profonde des préférences JSON stockées, pilotée par un catalogue, pour éviter les erreurs dues aux clés absentes lors de l'évolution du schéma.

Application durcie avec une CSP à nonces

Durci l'application avec une CSP fondée sur des nonces, HSTS, des cookies SameSite, des rôles de base de données au moindre privilège et une nouvelle vérification des droits côté serveur.

Choix techniques fondateurs de la plateforme

Défini les choix fondateurs de la plateforme dans les semaines suivant l'ouverture du dépôt en novembre 2025 — architecture en couches, accès aux données centré sur la base et zéro avertissement — toujours en vigueur neuf mois plus tard.

Applications Django pour IPTV

Développé et maintenu des applications web Django pour la plateforme IPTV, en livrant de nouvelles fonctionnalités et en améliorant les performances et la stabilité.

Suite de tests Go, SQL et navigateur

Conçu une suite de tests automatisés en couches — 981 tests Go, 543 spécifications frontend et navigateur, 494 tests de comportement SQL — avec tests par mutation, tests fondés sur les propriétés et contrôle obligatoire de l'accessibilité.

Droits d'accès Stripe et achats intégrés

Conçu la couche de paiement et de droits d'accès — Stripe et achats intégrés Apple et Google — en contrôlant l'accès à l'annuaire, à la recherche et à l'export au moyen d'un modèle de 11 tables vérifié côté serveur.

Traitements lents confiés à une file River

Sorti les traitements lents du chemin des requêtes pour les confier à une file de tâches River — 15 modules de traitement, 8 tâches planifiées et 20 tâches pg_cron — afin de répondre aux requêtes pendant la poursuite du travail en arrière-plan.

Surveillance des erreurs et traçage internes

Conçu en interne la surveillance des erreurs et le traçage OpenTelemetry plutôt que de les acheter — assainissement des charges utiles, détection des pics et régressions, symbolication et signal de vie synthétique — accessibles dans 11 vues d'exploitation.

Limitation de débit et protection contre les abus

Conçu des protections contre les abus qui refusent l'accès en cas de défaillance — 22 limiteurs de débit Redis, Cloudflare Turnstile, idempotence des requêtes et verrouillage de l'origine — pour rejeter robots et inondations de requêtes sans faire confiance aux appelants.

Portefeuille de fidélité, ligues et badges

Conçu le système de fidélité et de réputation — 67 fonctions sur un registre de 31 tables, avec ligues, badges et boutique d'échange — en verrouillant la ligne du solde pour fermer la fenêtre de double dépense.

Couche sociale : publications, fil et groupes

Conçu la couche sociale de la plateforme — publications, fil, groupes, mentions, graphe d'abonnements et récapitulatif d'occasions — sur la même couche de données fondée sur les fonctions que le reste du produit.

Espaces de recrutement et carrière des marins

Construit les espaces de recrutement et de carrière des marins ; l'instantané d'implémentation consigné comprend 151 fonctions stockées dans 48 tables pour postes, candidatures, certificats, progression de grade et temps de navigation.

Générateur de CV piloté par base de données

Conçu en Python un générateur de CV, références, portfolio et lettres de motivation piloté par base de données — 41 modules, 10 580 lignes — produisant six formats à partir d'une source SQLite de 23 tables assemblée par un pipeline idempotent de 19 étapes.

981 tests et 92 % de couverture

Vérifié le générateur avec 981 cas de test et un seuil de couverture des branches de 92 %, en traitant les avertissements comme des échecs, et vérifié l'idempotence en exécutant deux fois tout le pipeline depuis un fichier vide, sans changement autorisé au second passage.

Chaque règle du linter traitée comme une erreur

Activé comme erreurs toutes les règles du linter Python, corrigé 1 815 constats pour atteindre zéro et documenté la justification de chaque rare exemption, dont deux soutenues par un contrôle plutôt que par un commentaire.

Encodeur QR intégré en 522 lignes

Intégré dans le dépôt un encodeur QR — Reed-Solomon sur GF(256), disposition fixe des modules et huit masques — en 522 lignes plutôt que d'ajouter une troisième dépendance d'exécution, puis relu la matrice finale avec un décodeur écrit indépendamment.

Tous les textes déplacés dans le contenu

Déplacé chaque texte destiné aux utilisateurs hors de Python vers une arborescence de 874 fichiers en trois langues, après avoir trouvé des traductions inutilisées dont l'inactivité était invisible et un contrôle n'évaluant silencieusement qu'un tiers des réalisations.

Concurrence stricte de Swift 6 sur C

Adopté le mode complet de concurrence stricte de Swift 6 sans acteurs, en reliant une boucle d'événements C bloquante à l'acteur principal avec un producteur, un consommateur et un ordre unique, après avoir établi qu'une tâche par événement perdait l'ordre nécessaire à l'interface.

Parseur vérifiant chaque appel FFI

Écrit un parseur lisant le véritable en-tête C de 7 308 lignes et vérifiant chaque site d'appel, chaque constante d'énumération et le caractère final de toute classe propriétaire de pointeurs, après qu'un en-tête de substitution manuscrit a laissé compiler et lier des appels à trois fonctions supprimées, puis provoqué un plantage.

Moitié jusque-là inutilisée du cœur exploitée

Exploité la moitié du cœur de messagerie jusque-là inutilisée — transfert des sauvegardes, messages éphémères, modification et renvoi de messages, invitations vérifiées, proxys et politique de chiffrement — en exécutant chaque test contre la vraie bibliothèque, sans simulations.

Vous recherchez une prestation en Développement backend et API ?

Consultez les réalisations liées pour évaluer l'expérience pertinente. Dites-nous ce que vous devez construire ; nous pourrons discuter du périmètre, de la démarche de réalisation et de la transmission.

Contactez-nous Portfolio complet

Définissez le périmètre avant de commander les travaux

Présentez le processus de travail, les interfaces existantes et les utilisateurs qui ont besoin du changement. Nous pouvons définir une livraison testable et une transmission permettant à votre équipe d'assurer la maintenance des éléments livrés.

Consulter les tarifs Contactez-nous →

Précisez vos besoins avec nous

Dites-nous ce que vous attendez de « Développement backend et API », y compris l'échéance et l'équipe qui en assurera la maintenance. Nous pouvons discuter d'une façon concrète d'avancer.

Contactez-nous

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/services/backend-development/