Engineer Software

Backend-udvikling

33 resultater

Forbind delene af jeres system

Arbejdet under »Backend-udvikling« viser en del af et fungerende system. Vi kan hjælpe med at forbinde den del med resten af jeres drift. Den del, brugerne aldrig ser: hvor data ligger, hvilke regler der gælder, og hvordan andre systemer må tale med den.

Se vores ydelser Kontakt os →

Find en sammenlignelig udfordring

Åbn et eksempel, der ligner jeres situation. Gennemgangen beskriver beslutningerne bag resultatet, ud over teknologiens navn. Har designet og drevet 6 ETL/ELT-pipelines — 6 ETL-pipelines på tre databasemotorer

Har designet og drevet 6 ETL/ELT-pipelines Kontakt os →

Har forbedret GIS-kortydeevne 10x med PostgreSQL

Har gjort en geografisk kortapplikation 10 gange hurtigere gennem en strategisk databaseovergang fra MSSQL til PostgreSQL, herunder selve omlægningen af data, og optimeret behandlingen og datasikkerheden.

Har leveret 8 Power BI-projekter med manualer

Har gennemført 8 Power BI-projekter med grundige manualer og integreret Microsoft Power BI-værktøjer med NodeJS API og Python FastAPI til effektiv dataanalyse og visualisering.

Har styret full-stack-udviklingen af GIS-kort

Har styret full-stack-udviklingen af GIS-kort med ansvar for PostgreSQL, Mapbox, ReactJS og NodeJS for at skabe en integreret løsning.

Har ledet over 30 GIS-projekter

Har ledet udvikling, idriftsættelse og support af over 30 GIS-projekter og udvist ekspertise i PostgreSQL, Bash, Python, JavaScript, GDAL, ArcGIS, PostGIS og Mapbox.

Har udviklet et system til analyserapporter

Har udviklet et rapporteringssystem til dataanalyse og øget den kvartalsvise softwareomsætning med 400 % gennem Python-baserede PDF-rapporter.

Har udviklet 100 webapplikationer med HTML/HTML5

Har udviklet 100 webapplikationer med HTML/HTML5, CSS/SCSS, Django, WordPress og Joomla og sikret en alsidig onlinetilstedeværelse.

En maritim platform med klare systemgrænser

Har designet arkitekturen for en lagdelt maritim platform, der adskiller en Next.js PWA-frontend, et Go (Huma/Fiber) API og et PostgreSQL-funktionslag og holder al forretningslogik i databasen.

Har designet en JSON-passthrough-arkitektur

Har designet en JSON passthrough-arkitektur, hvor PostgreSQL-funktioner returnerer komplet JSON, som Go-API'et videresender uændret, så der ikke er nogen mellemliggende unmarshalling, og frontenden er afkoblet fra skemaændringer.

Har bygget skift af organisationskontekst

Har designet et system til skift af organisationskontekst med localStorage i klienten og en spejlet cookie på serveren, så brugere kan optræde som de organisationer, de administrerer, mens autorisationen følger mindste privilegium.

Har migreret API'et til Huma v2 med 100 % paritet

Har migreret HTTP-API'et fra Fiber til Huma v2 — 649 paths og 760 operationer — og opnået og fastholdt 100 % paritet mellem de ruter, serveren registrerer, og den OpenAPI-beskrivelse, den udgiver.

Har genereret API-kontrakten ud fra databasen

Har genereret API-kontrakten udad fra databasen — OpenAPI, en typet TypeScript-klient på 44.076 linjer, 61 mock handlers og de grænser, UI'et håndhæver — med en kontrol i hvert led, der fejler ved afvigelser.

Har bygget e-mail med failover og leveringslog

Har bygget e-mail som en funktion i selve platformen — tre udbydere med failover, webhooks for levering, logning af afsendelse og levering, skabeloner og kampagner — sikret af et opstartstjek, så platformen ikke starter uden mindst én udbyder.

Har designet et funktionsbaseret Postgres-datalag

Har designet et funktionsbaseret datalag i PostgreSQL — 1.275 lagrede funktioner fordelt på 34 skemaer — så al læsning og skrivning går gennem en funktion, databasen kan tildele rettigheder til, og ikke direkte gennem en tabel.

Har implementeret en katalogdrevet deep-merge

Har implementeret en katalogdrevet deep-merge for lagrede JSON-præferencer, hvilket forhindrer nedbrud på grund af manglende nøgler, når skemaet udvikler sig.

Har hærdet applikationen med nonce-baseret CSP

Har hærdet applikationen med nonce-baseret CSP, HSTS, SameSite-cookies, least-privilege-databaseroller og fornyet kontrol af rettigheder på serveren.

Har fastlagt platformens grundbeslutninger

Har fastlagt platformens grundlæggende beslutninger i ugerne efter, at repositoryet blev oprettet i november 2025 — lagdelingen, database-first-dataadgangen og standarden med nul advarsler — og de holder stadig ni måneder senere.

Har bygget og vedligeholdt Django-apps til IPTV

Har udviklet og vedligeholdt Django-webapplikationer til IPTV-platformen, udgivet nye funktioner og forbedret ydeevnen og stabiliteten.

Har bygget en testsuite til Go, SQL og browseren

Har bygget en lagdelt automatiseret testsuite — 981 Go-tests, 543 frontend- og browserspecs, 494 SQL-adfærdstests — med mutationstest, property-based tests og en tilgængelighedsgate.

Har bygget adgangsret via Stripe og in-app-køb

Har bygget betalings- og rettighedslaget — Stripe side om side med app-køb via Apple og Google — så adgangen til kataloget, søgningen og eksporten styres af en adgangsmodel på 11 tabeller, der kontrolleres på serveren.

Har flyttet langsomme opgaver til en River-jobkø

Har flyttet langsomt arbejde væk fra request-stien og over på en River-jobkø — 15 worker-moduler, 8 planlagte opgaver og 20 pg_cron-jobs — så en request returnerer, mens arbejdet bag den kører videre.

Har bygget egen fejlovervågning og tracing

Har bygget egen fejlovervågning og OpenTelemetry-tracing i stedet for at købe dem — sanitering af payloads, detektion af spikes og regressioner, symbolication og et syntetisk heartbeat — bag 11 operatørvisninger.

Har bygget fail-closed rate limiting mod misbrug

Har bygget fail-closed misbrugskontroller — 22 Redis-baserede rate limiters, Cloudflare Turnstile, idempotens på requests og en origin-lås — så platformen afviser bots og flood-angreb i stedet for at stole på dem, der kalder den.

Har bygget platformens operatør-back-office

Har bygget operatørernes back-office — omkring 40 admin-ruter og 128 komponenter, der dækker ejerskabskrav, moderation, feature flags, cache og diagnostik — så platformen kan drives uden databaseadgang.

Har bygget ejerskabskrav og deres afgørelse

Har bygget ejerskabskrav på virksomheder fra ende til anden — en bruger gør krav på en virksomhed, en administrator afgør sagen, og en godkendelse omskriver den autorisationsgraf, der afgør, hvem der må redigere hvad.

Har bygget loyalitetspoint, ligaer og badges

Har bygget loyalitets- og omdømmesystemet — 67 funktioner over en hovedbog på 31 tabeller, med ligaer, badges og en indløsningsshop — og med en rækkelås på saldoen til at lukke vinduet for dobbeltforbrug.

Har bygget det sociale lag: opslag, feed, grupper

Har bygget platformens sociale lag — opslag, feed, grupper, mentions, en følgergraf og et digest over mærkedage — på det samme funktionsbaserede datalag som resten af produktet.

Har bygget job- og karriereværktøj til søfarende

Har bygget job- og karriereværktøjer til søfarende; det registrerede implementeringsøjebliksbillede omfatter 151 lagrede funktioner i 48 tabeller til stillinger, ansøgninger, certifikater, rangudvikling og sejltid.

Har bygget en databasedrevet CV-generator

Har bygget en databasedrevet generator til CV, referencer, portefølje og ansøgninger i Python — 41 moduler, 10.580 linjer — der genererer seks outputformater fra én SQLite-kilde med 23 tabeller, som en idempotent pipeline i 19 trin bygger op.

Har indlejret en QR-encoder på 522 linjer

Har indlejret en QR-encoder — Reed-Solomon over GF(256), fast modullayout, otte maskemønstre — på 522 linjer i stedet for at tage en tredje runtime-afhængighed ind og verificeret den ved at læse den færdige matrix tilbage med en uafhængigt skrevet dekoder.

Har flyttet alle tekststrenge til et indholdstræ

Har flyttet hver streng, brugeren ser, ud af Python og ind i et indholdstræ på 874 filer på tre sprog, efter at have fundet døde oversættelser, som ingen kunne se var døde, og et tjek, der ubemærket kun bedømte en tredjedel af resultaterne.

Har indført Swift 6 strict concurrency over C

Har indført Swift 6's complete strict concurrency uden actors og bygget bro fra en blokerende C-hændelsesløkke til main actor gennem én producent, én forbruger og én rækkefølge — efter at have fastslået, at en opgave pr. hændelse mister den rækkefølge, grænsefladen afhænger af.

Har skrevet en parser, der tjekker hvert FFI-kald

Har skrevet en parser, der læser den rigtige C-header på 7.308 linjer og verificerer hvert kaldssted, hver enum-konstant, og at hver klasse, der ejer en pointer, er final, efter at en håndskrevet pladsholder-header lod kald til tre fjernede funktioner kompilere, linke og crashe.

Har taget den ubrugte halvdel af kernen i brug

Har taget den halvdel af beskedkernen i brug, som applikationen aldrig havde brugt — backupoverførsel, forsvindende beskeder, redigering og gensendelse af beskeder, verificerede invitationer, proxyer og krypteringspolitik — og kørt hver test mod det rigtige bibliotek uden mocks.

Sæt fokus på den tekniske indsats

Beskriv arbejdsgangen, de eksisterende grænseflader og de brugere, der har brug for ændringen. Vi kan definere en testbar leverance og en overdragelse, der gør teamet i stand til at vedligeholde løsningen.

Se, hvad det koster Kontakt os →

Sæt emnet ind i jeres sammenhæng

Fortæl os, hvordan »Backend-udvikling« indgår i jeres system. Vi kan hjælpe med at vælge den relevante ydelse og afgrænse arbejdet.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://software.engineer.company/da/categories/backend/