Engineer Programvare

Portefølje

Ingeniørtjenester med eksempler på utført arbeid.

Engineer ApS ble registrert i 2025. Mesteparten av arbeidet som vises her, ble levert før det av gründeren vår inne i kundenes egne team; noe er levert av selskapet selv, og noe siden i ansettelsesforhold. De samme hendene og de samme standardene — vi sier hva som er hva i stedet for å viske det ut.

Finn erfaring det er verdt å bruke

Bruk oversikten til å finne erfaring som er relevant for prosjektet deres.

Se tjenestene våre Ta kontakt →

Les beslutningene bak arbeidet

Eksemplene beskriver levert arbeid på tvers av tjenester. Les en gjennomgang for å se situasjonen og beslutningene.

Prosjekter Ta kontakt →

Bla etter kategori

PROGRAMVAREUTVIKLING

Backend-utvikling
Frontend-utvikling
Fullstack-utvikling
API-er og integrasjon
Webutvikling
Designsystemer og brukergrensesnitt
UX/UI-design
Testing og kvalitetssikring
Ytelsesoptimalisering
Python
Internasjonalisering

DATA & DATABASER

Databaser
PostgreSQL
SQL
Data engineering
Datapipelines (ETL/ELT)
Dataanalyse
Datastyring
Datavarehus
GIS / geodata

SKY, INFRASTRUKTUR & DEVOPS

Skytjenester · DevOps · Automatisering og CI/CD · Infrastruktur · Linux og servere · Systemadministrasjon · Nettverk og VPN · Overvåking og observerbarhet · Pålitelighet og sikkerhetskopier · Sikkerhet · Plattformarkitektur · Migrering og modernisering

RÅDGIVNING & LEDELSE

Teknisk ledelse · Teamledelse · Mentorarbeid og coaching · Smidig og Scrum · Prosjektledelse · Produkt og krav · Interessenter og rapportering · Dokumentasjon · Løsningsarkitektur · Merkevare og markedsføring

Alt nedenfor **er levert** — ikke foreslått, ikke planlagt.

Hver oppføring forteller hva arbeidet endret, og lenker til tjenestene den viser.

Det vi tilbyr Hva det koster →

{ Programvareutvikling }

Plattform- og løsningsarkitektur

Bestemme hvordan helheten skal bygges

Formen på et system tegnet opp på papiret først, slik at delene passer sammen og kostnaden er kjent før noen skriver kode.

Tok i bruk streng samtidighet i Swift 6 over C

Tok i bruk fullstendig streng samtidighet i Swift 6 uten actors, og koblet en blokkerende C-hendelsesløkke til main actor gjennom én produsent, én konsument og én rekkefølge — etter å ha slått fast at én task per hendelse mister rekkefølgen grensesnittet er avhengig av.

En avgrenset hendelsesflyt i en macOS-app

Avgrenset hendelsesbuffere og logging etter en registrert macOS-hendelse med vekst på 41 MB i sekundet og 111 GB komprimerte sider på en maskin med 36 GB; en registrert full testkjøring reduserte kjernens logglinjer fra 610 996 til 1 411.

Bygde en native macOS-klient i Swift 6

Bygde en native meldingsklient for macOS i Swift 6 og SwiftUI — 11 141 linjer fordelt på 53 filer — over C-grensesnittet til en Rust-kjerne lenket inn som et statisk arkiv fra en låst revisjon.

Bygde inn en QR-enkoder på 522 linjer

Bygde inn en QR-enkoder — Reed-Solomon over GF(256), fast moduloppsett, åtte maskemønstre — på 522 linjer i stedet for å ta inn en tredje kjøretidsavhengighet, og verifiserte den ved å lese den ferdige matrisen tilbake med en uavhengig skrevet dekoder.

Bygde en databasedrevet CV-generator

Bygde en databasedrevet generator for CV, referanser, porteføljе og søknadsbrev i Python — 41 moduler, 10 580 linjer — som gjengir seks utdataformater fra én SQLite-kilde med 23 tabeller, satt sammen av en idempotent pipeline i 19 trinn.

Offentlige tjenester på en liten server

Driftet selskapets offentlige tjenester på en vert med én kjerne og 512 MB — en git-forge, en webserver for sju domener, Tor, to servere for alternative protokoller og utestenging av inntrengere — med 464 MB brukbart minne som designbegrensning; backupautomatisering er en separat implementasjon.

Bygde hele kjerneinfrastrukturen fra bunnen av

Designet og bygde, som en av de første ansatte, hele kjerneinfrastrukturen og de tilhørende prosessene fra bunnen av for en SaaS-oppstartsbedrift innen grønn energi, og la grunnlaget for rask vekst.

Bygde infrastruktur som fortsatt kjører år senere

Planla og implementerte ny infrastrukturfunksjonalitet for interne og eksterne systemer, og bygde løsninger så solide at de fortsatt kjører år senere med minimale endringer.

Tok plattformens grunnleggende tekniske valg

Tok plattformens grunnleggende beslutninger i ukene etter at repoet ble opprettet i november 2025 — lagdelingen, datatilgang med databasen først og kravet om null advarsler — og de står seg fortsatt ni måneder senere.

Modellerte 348 tabeller i 34 Postgres-skjemaer

Modellerte det maritime domenet i 348 normaliserte tabeller fordelt på 34 PostgreSQL-skjemaer — fagfolk, selskaper, skip, stillinger, omtaler og resten — med SMALLINT-oppslag og UUID v7-nøkler.

Designet et funksjonsbasert datalag i PostgreSQL

Designet et funksjonsbasert datalag i PostgreSQL — 1 275 lagrede funksjoner fordelt på 34 skjemaer — slik at hver lesing og skriving går gjennom en funksjon databasen kan gi tilgang til, i stedet for gjennom en tabell.

Migrerte API-et til Huma v2 med 100 % paritet

Migrerte HTTP-API-et fra Fiber til Huma v2 — 649 stier og 760 operasjoner — og nådde og holdt 100 % samsvar mellom rutene serveren registrerer og OpenAPI-beskrivelsen den publiserer.

Bygde bytte av organisasjonskontekst for brukerne

Designet et system for å bytte organisasjonskontekst med localStorage på klienten og speiling i informasjonskapsler på serversiden, slik at brukere kan opptre som organisasjoner de administrerer, med autorisasjon etter prinsippet om minste privilegium.

Designet JSON rett fra Postgres til API-et

Designet en JSON-passthrough-arkitektur der PostgreSQL-funksjoner returnerer komplett JSON som Go-API-et videresender uendret, noe som fjernet mellomliggende unmarshalling og frikoblet frontend fra skjemaendringer.

En maritim plattform med klare systemgrenser

Utformet arkitekturen for en lagdelt maritim plattform som skiller en Next.js PWA-frontend, et Go-API (Huma/Fiber) og et funksjonslag i PostgreSQL, med all forretningslogikk i databasen.

Forenklet infrastruktur utenfor AWS

La om infrastrukturen til en internasjonal kunde ved å fjerne ubrukte tjenester, tilpasse kapasiteten og flytte arbeidslaster bort fra AWS. Grunnleggeren beskriver en reduksjon i skykostnadene med en faktor på 10 uten tap av produktivitet.

Et kartprodukt, håndtert fra ende til annen

Utformet, utviklet og støttet infrastruktur, databehandling og en kartapplikasjon over 2 år, med både programvareleveranser og driftsvedlikehold.

Fornyet interne prosesser og sparte 8 000 timer

Fornyet interne prosesser og sparte 8 000 timer ved å forbedre programvarearkitekturen, systemene og effektiviteten i planleggingen.

Ledet en GIS-kartapp til 10 ganger mer omsetning

Ledet programvareutviklingen av en GIS-kartapplikasjon, økte inntektene 10 ganger og posisjonerte produktet som en primær dataressurs.

Én infrastrukturplan for 14 avdelinger

Designet et helhetlig infrastrukturrammeverk for et forskningsinstitutt som berørte 14 avdelinger, med fleksible moduler, samlede datapipelines og strukturerte støttestrategier for varig innføring.

{ Programvareutvikling }

Backend- og API-utvikling

Motoren bak appen eller nettstedet ditt

Delen brukerne aldri ser: hvor dataene ligger, hvilke regler som gjelder, og hvordan andre systemer får lov til å snakke med den.

Nådde den ubrukte halvdelen av kjernen

Tok i bruk den halvdelen av meldingskjernen applikasjonen aldri hadde brukt — overføring av sikkerhetskopier, forsvinnende meldinger, redigering og ny sending av meldinger, verifiserte invitasjoner, proxyer og krypteringspolicy — og kjørte hver test mot det ekte biblioteket uten mocks.

Skrev en parser som verifiserer hvert FFI-kall

Skrev en parser som leser den faktiske C-headeren på 7 308 linjer og verifiserer hvert kallsted, hver enum-konstant og at hver klasse som eier en peker, er final, etter at en håndskrevet plassholder-header lot kall til tre fjernede funksjoner kompilere, lenke og krasje.

Flyttet hver tekststreng inn i et innholdstre

Flyttet alle brukerrettede strenger ut av Python og inn i et innholdstre på 874 filer på tre språk, etter å ha funnet døde oversettelser ingen kunne se var døde, og en kontroll som i det stille vurderte en tredjedel av resultatene.

Aktiverte alle linterregler som feil

Slo på alle reglene i Python-linteren på feilnivå og arbeidet seg gjennom 1 815 funn ned til null, med en skriftlig begrunnelse for hvert av de få unntakene og to av dem sikret av en kontroll i stedet for en kommentar.

Holdt generatoren til 981 tester og 92 % dekning

Holdt generatoren til 981 testtilfeller med et minstekrav på 92 % grendekning og advarsler behandlet som feil, og verifiserte idempotens ved å kjøre hele byggepipelinen to ganger fra en tom fil og kreve at den andre kjøringen ikke endret noe.

Bygde arbeidsflater for rekruttering og sjøfolk

Bygde arbeidsflater for rekruttering og sjøfolks karriere; det registrerte øyeblikksbildet av implementasjonen omfatter 151 lagrede funksjoner i 48 tabeller for stillinger, søknader, sertifikater, rangutvikling og fartstid.

Bygde det sosiale laget: innlegg, feed og grupper

Bygde plattformens sosiale lag — innlegg, feed, grupper, nevninger, en følgergraf og et sammendrag av merkedager — på det samme funksjonsbaserte datalaget som resten av produktet.

Bygde en lojalitetslommebok, ligaer og merker

Bygde lojalitets- og omdømmesystemet — 67 funksjoner over en hovedbok på 31 tabeller, med ligaer, merker og en innløsningsbutikk — med radlås på saldoen for å lukke vinduet for dobbeltforbruk.

Bygde fail-closed hastighetsgrenser mot misbruk

Bygde misbruksvern som stenger ved feil — 22 Redis-baserte hastighetsbegrensere, Cloudflare Turnstile, idempotente forespørsler og en origin-lås — slik at plattformen avviser boter og flom i stedet for å stole på dem som kaller den.

Bygde egen feilovervåking og sporing

Bygde egen feilovervåking og OpenTelemetry-sporing i stedet for å kjøpe dem — rensing av nyttelast, oppdagelse av topper og regresjoner, symbolisering og et syntetisk hjerteslag — bak 11 operatørvisninger.

Flyttet tregt arbeid over i en River-jobbkø

Flyttet tregt arbeid ut av forespørselsbanen og over i en River-jobbkø — 15 worker-moduler, 8 planlagte oppgaver og 20 pg_cron-jobber — slik at en forespørsel returnerer mens arbeidet bak den fortsetter.

Bygde tilgangsrettigheter for Stripe og appkjøp

Bygde laget for betaling og tilgangsrettigheter — Stripe sammen med kjøp i appen hos Apple og Google — som styrer tilgangen til katalogen, søket og eksporten gjennom en tilgangsmodell på 11 tabeller som kontrolleres på serveren.

Bygde en testsuite for Go, SQL og nettleser

Bygde en lagdelt automatisert testsuite — 981 Go-tester, 543 spesifikasjoner for frontend og nettleser, 494 atferdstester i SQL — med mutasjonstesting, egenskapsbaserte tester og en kvalitetsport for universell utforming.

Bygde og vedlikeholdt Django-apper for IPTV

Utviklet og vedlikeholdt Django-webapplikasjoner for IPTV-plattformen, leverte nye funksjoner og forbedret ytelse og stabilitet.

Herdet applikasjonen med nonce-basert CSP

Herdet applikasjonen med nonce-basert CSP, HSTS, SameSite-informasjonskapsler, databaseroller etter prinsippet om minste privilegium og ny kontroll av tilgangsrettigheter på serversiden.

Implementerte en katalogstyrt deep merge

Implementerte en katalogstyrt deep merge av lagrede JSON-innstillinger, som forhindrer krasj ved manglende nøkler når skjemaet utvikler seg.

Bygde e-post med failover og leveringslogging

Bygde e-post som en plattformfunksjon — tre leverandører med failover, leveringswebhooks, logging av sending og levering, maler og kampanjer — bak en oppstartskontroll som ikke lar plattformen starte uten minst én leverandør.

Genererte API-kontrakten med databasen som kilde

Genererte API-kontrakten fra databasen og utover — OpenAPI, en typet TypeScript-klient på 44 076 linjer, 61 mock-handlere og grensene brukergrensesnittet håndhever — med en kontroll i hvert ledd som feiler ved avvik.

Utviklet et rapporteringssystem for dataanalyse

Utviklet et rapporteringssystem for dataanalyse og økte kvartalsinntektene fra programvaren med 400 % gjennom Python-baserte PDF-rapporter.

Ledet 30+ GIS-prosjekter fra utvikling til støtte

Ledet utvikling, produksjonssetting og støtte for over 30 GIS-prosjekter, med ekspertise i teknologiene PostgreSQL, Bash, Python, JavaScript, GDAL, ArcGIS, PostGIS og Mapbox.

Styrte fullstack-utviklingen av GIS-kart

Styrte fullstack-utviklingen av GIS-kart med ansvar for PostgreSQL, Mapbox, ReactJS og NodeJS, og leverte en integrert løsning.

Leverte 8 Power BI-prosjekter med veiledninger

Leverte 8 Power BI-prosjekter med grundige brukerveiledninger, og integrerte Microsoft Power BI-verktøy med et NodeJS-API og Python FastAPI for effektiv dataanalyse og visualisering.

Designet og driftet 6 ETL/ELT-pipelines

Designet, implementerte og administrerte 6 ETL/ELT-pipelines med Google BigQuery, MSSQL, PostgreSQL, Shell-skripting, PL/pgSQL og Transact-SQL, og integrerte data for effektiv behandling i et Python-API.

Tjenesten inngår i arbeid som står andre steder på siden:

Tok i bruk streng samtidighet i Swift 6 over C · Bygde inn en QR-enkoder på 522 linjer · Bygde en databasedrevet CV-generator · Tok plattformens grunnleggende tekniske valg · Designet et funksjonsbasert datalag i PostgreSQL · Migrerte API-et til Huma v2 med 100 % paritet · Bygde bytte av organisasjonskontekst for brukerne · Designet JSON rett fra Postgres til API-et · En maritim plattform med klare systemgrenser

{ Programvareutvikling }

Frontend-utvikling

Skjermbildene folk faktisk bruker

Grensesnitt som laster raskt, fungerer på mobilen og oppfører seg likt i alle nettlesere kundene dine måtte bruke.

Bygde et macOS-designsystem med 48 tokens

Bygde et designsystem med 48 tokens på plattformens eget glassmateriale og skrev linteren som avviser et magisk tall, en hardkodet skriftstørrelse eller en animasjon som ignorerer innstillingen for redusert bevegelse.

Ga alle 17 ikoner navn for skjermlesere

Ga hver kontroll med bare ikon i grensesnittet et navn for skjermlesere etter å ha funnet at send-knappen ble lest opp som «arrow up circle, button», og skrev linteren som krever etiketten innen åtte linjer fra ikonet.

Lintet selve kvalitetsporten på 10 242 linjer

Satte 10 242 linjer JavaScript i kvalitetsporten under en formaterer og en linter etter å ha slått fast at det var den største kodemengden i repoet og den eneste ingenting leste, og rettet 13 funn uten å undertrykke noen.

Kuttet stilarkpakken fra 87 KB til 31 KB

Kuttet stilarkpakken fra 87 KB til 31 KB ved å ta ut en base64-kodet skrift, og fjernet 756 KB fordelt på 22 filer som ble publisert ved hver produksjonssetting og som ingenting refererte til.

Bygde et utskriftsstilark på 637 linjer

Bygde et utskriftsstilark på 637 linjer mot fire dokumenterte atferder i gjengivelsesmotorer, blant dem innfelte overskrifter som ble skrevet ut hvitt på hvitt for enhver leser med nettleseren innstilt på mørkt.

Gjorde innstillingen for redusert bevegelse ærlig

Gjorde innstillingen for redusert bevegelse ærlig etter å ha funnet elleve selektorer som fortsatt animerte under den, fordi en universell transition-none-regel taper på spesifisitet mot enhver regel med en klasse.

Fant at det lyse temaet manglet et helt lag

Fant ut at det lyse temaet hadde manglet et helsides overleggslag helt siden det ble skrevet, ved å kreve at begge temaene maler hver lagdelte flate med like mange lag.

Lukket fargesystemet ved 28 farger

Lukket fargesystemet ved 28 dokumenterte farger med en linter som avviser en verdi som er malt, men udokumentert, dokumentert, men ikke malt, feilmålt, skrevet om som en literal, eller innenfor en perseptuell avstand på 0,02 fra en som allerede finnes.

Feil i universell utforming som kontrollverktøyet deres overser

Fant og rettet 11 feil i universell utforming som nettleseren meldte var i orden — åtte bunntekstlenker som fortsatt lå i tabulatorrekkefølgen bak pointer-events, en scroll-tidslinje som klippet kolofonen av tre sider, og en regelmotor som kjørte 70 av sine 105 regler.

Tilgjengelighetskontroller på 32 sider i begge temaer

Implementerte WCAG 2.2 Level AA-kontroller på 32 representative sider i begge fargetemaer, med en samsvarsoversikt, kontroll av ett AAA-kriterium og det delvise resultatet for kriterium 1.4.13 uttrykkelig angitt.

Byttet 27 skriftstørrelser mot en skala i 6 trinn

Erstattet 27 gjengitte skriftstørrelser, der de nærmeste naboene lå 0,6 % fra hverandre, med en Major Third-skala i seks trinn, og skrev linteren som avviser den tjueåttende.

Et sikkerhetsnett som ikke lenger koster en halvtime

Kuttet nettstedets nettleserdrevne kvalitetsport fra 1 636 sekunder til 615 ved å kjøre kontrollene med de lengste først gjennom en worker-pool begrenset til fire baner, etter å ha målt at alfabetisk rekkefølge kostet 320 sekunder mot 224.

Bygde et flerspråklig Hugo-nettsted på 3 språk

Bygde et flerspråklig statisk nettsted i Hugo med 108 malfiler, blant dem 53 partials, som publiserer hver side i fire representasjoner fra ett innholdstre på tre språk, og på nytt under sju fokuserte underdomener bygd fra det samme treet.

Bygde operatørenes backoffice for plattformen

Bygde back office for operatørene — rundt 40 administrasjonsruter og 128 komponenter for eierskapskrav, moderering, funksjonsflagg, cache og diagnostikk — slik at plattformen kan driftes uten databasetilgang.

Vedlikeholdt Hugo-nettsted, bedret produktets UI

Vedlikeholdt og videreutviklet det eldre statiske Hugo-nettstedet, og bidro samtidig med UI/UX-forbedringer til hovedproduktet for forvaltning av eiendeler.

Bygde et engelsk/dansk i18n-system

Etablerte et engelsk/dansk internasjonaliseringssystem med 12 027 meldinger per språkversjon fordelt på 458 navneromsfiler, med vokabular håndhevet av linter og et budsjett på 400 linjer per fil.

Designet produktets UI og UX fra ende til annen

Designet produktets UI og UX fra ende til annen — katalogrutenett, doble kort- og tabellvisninger, fortløpende validering av krav og brødsmulenavigasjon.

Designet «Liquid Glass»-designsystem for iOS 26

Designet et «Liquid Glass»-designsystem for iOS 26 og et kanonisk komponentregister håndhevet av lint-regler for å hindre at brukergrensesnittet spriker.

Leverte full støtte for Progressive Web App

Leverte full støtte for Progressive Web App — kan installeres og fungerer uten nettforbindelse — med caching under kjøring i Workbox via next-pwa.

Bygde Next.js 16-frontenden: SSR, SSG og CSR

Bygde Next.js 16-frontenden med bevisste strategier for SSR, SSG og CSR, og en gjenbrukbar fabrikk for forhåndshentede serversider som fjerner N+1-kaskaden av hentinger fra hver autentisert side som flyttes over på den.

Designet 30 nettsteder fra Photoshop til HTML

Designet 30 nettsteder og leverte unike og fleksible løsninger ved å konvertere Photoshop-design til HTML.

Utviklet 100 webapplikasjoner med HTML/HTML5

Utviklet 100 webapplikasjoner med HTML/HTML5, CSS/SCSS, Django, WordPress og Joomla, og sikret en variert tilstedeværelse på nett.

Bidro i designprosessene for brukergrensesnitt

Bidro i designprosessene for brukergrensesnitt og sikret intuitive, visuelt tiltalende og brukervennlige grensesnitt i prosjektene.

Presenterte 200 UI/UX-forbedringer

Presenterte 200 UI/UX-forbedringer for GIS-kartapplikasjonen og økte inntektene 10 ganger gjennom bedre programvarefunksjoner.

Tjenesten inngår i arbeid som står andre steder på siden:

En avgrenset hendelsesflyt i en macOS-app · Bygde en native macOS-klient i Swift 6 ·

Tok plattformens grunnleggende tekniske valg · Bygde bytte av organisasjonskontekst for brukerne ·

En maritim plattform med klare systemgrenser · Styrte fullstack-utviklingen av GIS-kart

{ Programvareutvikling }

Fullstack-produktutvikling

Et fungerende produkt, fra ende til annen

Ett team som tar en idé fra skisse til noe kundene dine kan logge inn på, i stedet for at den sendes videre mellom tre leverandører.

Bygde eierskapskrav og avgjørelse for selskaper

Bygde løsningen for eierskapskrav til selskaper fra ende til annen — en bruker gjør krav på et selskap, en administrator avgjør saken, og en godkjenning skriver om autorisasjonsgrafen som bestemmer hvem som får redigere hva.

Tjenesten inngår i arbeid som står andre steder på siden:

Bygde en native macOS-klient i Swift 6 · Bygde en databasedrevet CV-generator ·

Bygde et flerspråklig Hugo-nettsted på 3 språk · Bygde arbeidsflater for rekruttering og sjøfolk ·

Bygde det sosiale laget: innlegg, feed og grupper · Bygde operatørenes backoffice for plattformen ·

Bygde og vedlikeholdt Django-apper for IPTV · Tok plattformens grunnleggende tekniske valg ·

Bygde Next.js 16-frontenden: SSR, SSG og CSR · En maritim plattform med klare systemgrenser ·

Et kartprodukt, håndtert fra ende til annen · Utviklet 100 webapplikasjoner med HTML/HTML5 · Ledet 30+ GIS-prosjekter fra utvikling til støtte · Styrte fullstack-utviklingen av GIS-kart · Ledet en GIS-kartapp til 10 ganger mer omsetning

{ Programvareutvikling }

Utvikling av nettsteder og CMS

Et nettsted du kan oppdatere selv

Et nettsted som er raskt, lett å finne og kan redigeres av dine egne ansatte, uten en regning hver gang en formulering endres.

Publiserte et lesbart onion-speil på Tor

Publiserte nettstedet som et Tor onion-speil på en lesbar adresse som begynner med engineer, med nøkkelen generert utenfor verten slik at den aldri havnet i dette repoet eller på en bærbar PC, og lot det være den ene av fire inngangsdører som aldri telles.

Bygde den visuelle identiteten fra 2 tegninger

Bygde selskapets ikon- og favicon-sett fra to håndlagde tegninger, med alle avledede filer regenerert av skript og en kontroll som feiler når en avledet fil ble committet før kilden.

Bestod 106 av 106 PDF/UA-1-profilkontroller

Dokumenterte PDF/UA-1-validering av ni dokumenter med 106 av 106 bestått kontroller i den konfigurerte profilen for universell utforming, mens arkivvarianten feilet på én av 146 regler.

Rettet et sitemap med ett felles tidsstempel

Rettet et sitemap der 172 av 176 URL-er delte ett endringstidspunkt, ved å hente datoen fra git-historikken etter å ha slått fast at eksporten skriver om hver fil ved hver kjøring.

Erstattet 963 metadatablokker med én graf

Byttet ut 963 anonyme blokker med strukturerte data, som gjentok selskapet 1 671 ganger, mot én lenket graf av 16 typer utledet fra stabile opprinnelsesidentifikatorer.

Speilet nettstedet til Gemini og Gopher

Bygde Gemini- og Gopher-utdata fra samme nettstedskilde; resultatbeskrivelsen oppgir 777 dokumenter i hvert format på det stadiet og null endrede HTML-byte ved tillegget av formatene.

Genererte 495 sider fra en SQLite-eksport

Genererte 495 resultatsider på tre språk fra en skrivebeskyttet SQLite-eksport, med sideadressen skrevet som data, slik at en rettet setning ikke lenger flyttet siden og brøt lenken.

Koden deres på deres egen server

Satte selskapets egen git-forge på Soft Serve i produksjon, privat som standard, uten webpanel og med SSH-porten bundet til loopback bak en hoppvert, og gjorde landingssiden foran den til en byggeartefakt fra hovednettstedet i stedet for en kopi som vedlikeholdes for hånd.

Kombinerte webutvikling med SEO og innhold

Drev frem kundenes suksess på nett ved å kombinere skreddersydd utvikling og design av nettsteder med SEO, innholdsstrategi og tekstforfatting.

Administrerte 40 nettsteder på Apache og Nginx

Administrerte 40 nettsteder på Ubuntu Linux-servere for webhotell med Apache og Nginx, og sikret høy tilgjengelighet og ytelse.

Tjenesten inngår i arbeid som står andre steder på siden:

Kuttet stilarkpakken fra 87 KB til 31 KB · Bygde et utskriftsstilark på 637 linjer · Gjorde innstillingen for redusert bevegelse ærlig · Feil i universell utforming som kontrollverktøyet deres overser · Tilgjengelighetskontroller på 32 sider i begge temaer · Bygde et flerspråklig Hugo-nettsted på 3 språk · Vedlikeholdt Hugo-nettsted, bedret produktets UI · Bygde og vedlikeholdt Django-apper for IPTV · Leverte full støtte for Progressive Web App · Bygde Next.js 16-frontenden: SSR, SSG og CSR · Designet 30 nettsteder fra Photoshop til HTML · Utviklet 100 webapplikasjoner med HTML/HTML5

{ Programvareutvikling }

Klar for AI-agenter

Synlig når kundene spør en AI

Et nettsted eller en app som AI-assistentene kundene dine bruker, kan finne, lese og handle ut fra, kontrollert ved hver utgivelse i stedet for lovet én gang.

Drev utviklingen som 51 skriftlige initiativer

Drev utviklingen av nettstedet som 51 skriftlige initiativer fordelt på 54 plandokumenter, hvert med sine åpne avkrysningsbokser, sine målinger og beslutningene det avslo — blant dem seks av åtte funn om agentberedskap som ble avvist med begrunnelsen registrert.

Tjenesten inngår i arbeid som står andre steder på siden:

Rettet et sitemap med ett felles tidsstempel · Erstattet 963 metadatablokker med én graf · Tilgjengelighetskontroller på 32 sider i begge temaer · Bygde et flerspråklig Hugo-nettsted på 3 språk

{ Programvareutvikling }

Verktøy for AI-agenter og MCP-servere

La AI bruke programvaren du allerede har

Programvaren du allerede har, åpnet som verktøy en AI-agent kan kalle, uten at noe som allerede virker, må skrives om.

Tjenesten inngår i arbeid som står andre steder på siden:

Bygde operatørenes backoffice for plattformen · Flyttet tregt arbeid over i en River-jobbkø · Genererte API-kontrakten med databasen som kilde · Migrerte API-et til Huma v2 med 100 % paritet · Designet JSON rett fra Postgres til API-et

{ Programvareutvikling }

GIS- og geodataløsninger

Kart som svarer på spørsmål

Alt som skjer på et sted — fartøy, kjøretøy, tomter, sensorer — lagt på et kart du kan søke i, måle på og dele.

Ledet 30 vellykkede GIS-prosjekter til levering

Styrte gjennomføringen av 30 vellykkede GIS-prosjekter og viste lederskap i leveransen av innovative løsninger på tvers av fagfeltet.

Optimaliserte prognose- og investeringsstrategier

Optimaliserte prognose- og investeringsstrategier for 11 nettselskaper og økte driftseffektiviteten gjennom datadrevne GIS-løsninger.

Automatiserte leveransen av 20 GIS-datapipelines

Automatiserte leveransen av 20 GIS-datapipelines og ETL-prosesser for applikasjonsdata, og effektiviserte infrastrukturautomatisering og rapportering.

Automatiserte GIS-SaaS-utrulling i GitHub Actions

Automatiserte produksjonssettingen, databehandlingen og rapporteringssystemet for en GIS SaaS-applikasjon med GitHub Actions CI/CD, Python, Bash og SQL.

Drev 100 datavarehus: Postgres, MS SQL, BigQuery

Utformet, opprettet og administrerte 100 databaser for datavarehus i PostgreSQL, MS SQL og Google BigQuery, hovedsakelig med GIS- og tidsseriedata, og optimaliserte ytelse og skalerbarhet.

Publiserte 500 datarapporter om strøm og GIS

Publiserte 500 analyserapporter om strøm- og GIS-data, basert på grundige undersøkelser og feilsøking, for å sikre presis innsikt i geografiske stordata og tidsseriedata.

Løste 1 000 problemer i geografiske data

Løste 1 000 problemer i geografiske data og tidsseriedata med GDAL, ArcGIS, PostGIS, Mapbox, QGIS, SQL (PL/pgSQL, Transact-SQL) og Bash, og sikret behandling av stordata med høy kvalitet.

Gjorde GIS-kart 10 ganger raskere på PostgreSQL

Forbedret ytelsen i en geografisk kartapplikasjon 10 ganger gjennom et strategisk databaseskifte fra MSSQL til PostgreSQL, inkludert omleggingen av dataene, og optimaliserte samtidig behandlingen og datasikkerheten.

Gjorde en geodatapipeline 50 ganger raskere

Registrerte en forbedring på 50 ganger i en geografisk datapipeline i grunnleggerens arbeid hos Utiligize gjennom SQL- og datamodellendringer i PostgreSQL, MS SQL og Google Cloud BigQuery.

Tjenesten inngår i arbeid som står andre steder på siden:

Et kartprodukt, håndtert fra ende til annen · Ledet 30+ GIS-prosjekter fra utvikling til støtte · Styrte fullstack-utviklingen av GIS-kart · Ledet en GIS-kartapp til 10 ganger mer omsetning

{ Programvareutvikling }

UI/UX-design og designsystemer

Gjøre bruken selvforklarende

Oppsett, ordvalg og flyt lagt opp slik at folk fullfører det de kom for å gjøre, i stedet for å ringe kundestøtten din.

Skrev om feilmeldingene som skyldte på brukerne

Skrev om applikasjonens feilmeldinger i tråd med en skriftlig tonestandard etter at en avvist innlogging ga brukeren skylden for en skrivefeil når leverandøren egentlig krevde et appspesifikt passord, og dekket det med en test som navngir leverandøren.

Tjenesten inngår i arbeid som står andre steder på siden:

Bygde den visuelle identiteten fra 2 tegninger · Bygde et macOS-designsystem med 48 tokens · Ga alle 17 ikoner navn for skjermlesere · Bygde en native macOS-klient i Swift 6 · Bestod 106 av 106 PDF/UA-1-profilkontroller · Bygde et utskriftsstilark på 637 linjer · Gjorde innstillingen for redusert bevegelse ærlig · Fant at det lyse temaet manglet et helt lag ·

Lukket fargesystemet ved 28 farger · Feil i universell utforming som kontrollverktøyet deres overser · Tilgjengelighetskontroller på 32 sider i begge temaer · Byttet 27 skriftstørrelser mot en skala i 6 trinn · Kombinerte webutvikling med SEO og innhold · Vedlikeholdt Hugo-nettsted, bedret produktets UI · Designet produktets UI og UX fra ende til annen · Designet «Liquid Glass»-designsystem for iOS 26 · Designet 30 nettsteder fra Photoshop til HTML · Bidro i designprosessene for brukergrensesnitt · Presenterte 200 UI/UX-forbedringer

{ Programvareutvikling }

Internasjonalisering og lokalisering

På mer enn ett språk

Programvare som leses naturlig i alle markeder du selger i, også datoene, valutaene og måten navn sorteres på.

Bygde en tallkontroll på tvers av språk

Bygde en innholdskontroll på tvers av språk som feiler når en oversettelse mister et tall den engelske teksten oppgir, og når et språk bruker en notasjon som ikke hører hjemme i det — og fant to danske beskrivelser som manglet en måleverdi, og seksten ukrainske tekstbiter med engelske anførselstegn.

Tjenesten inngår i arbeid som står andre steder på siden:

Flyttet hver tekststreng inn i et innholdstre · Bygde en databasedrevet CV-generator · Speilet nettstedet til Gemini og Gopher · Genererte 495 sider fra en SQLite-eksport · Tilgjengelighetskontroller på 32 sider i begge temaer · Bygde et flerspråklig Hugo-nettsted på 3 språk · Bygde et engelsk/dansk i18n-system

{ Programvareutvikling }

Utvikling av skreddersydde AI-agenter

En AI-assistent som kan faget ditt

En agent koblet til systemene du allerede bruker, testet på saker dine egne fagfolk skriver, under oppsyn av folkene dine og aldri låst til én modell.

Fikk seks innsamlere til å si hva de hoppet over

Fikk seks rapporteringsinnsamlere til å oppgi den delen av inndataene de aldri leste — filer som forble lukket, sonder som ikke var installert, journallinjer som ikke ble gjenkjent — og tilpasset terskelverdier til 116 målinger, etter å ha slått fast at en blind innsamling og en vert der alt er i orden, gir den samme siden.

Skrev tester for selve kontrollverktøyene

Skrev tester for selve kontrollene etter å ha slått fast at en kontroll som bare får ren inndata, en dag vil melde rent fordi den ikke leste noe — plantet en stavefeil for å bekrefte at stavekontrollen finner den, og hentet et id-område fra databasen i stedet for fra et tall i testen.

Bygde en commit-port med 22 lintere

Bygde en commit-port med 22 enlinjes lintere pluss fem som fortjener et avsnitt, uten advarselsnivå og uten tillatte undertrykkinger i koden, som dekker HTML, CSS, JavaScript, Python, YAML, Markdown, shell, lenker, staving, hemmeligheter og typografi.

Bygde repoets kontrollmotor med 268 kontroller

Bygde repoets kontrollmotor — 268 registrerte commit-kontroller, 277 lint-regler og 15 egne ESLint-regler — pluss 146 tester av selve kontrollene, slik at det er bygget, ikke gjennomgangen, som holder standarden.

Tjenesten inngår i arbeid som står andre steder på siden:

Bygde egen feilovervåking og sporing

Sammenlign utgangspunktet deres

Velg et sammenlignbart arbeid og identifiser forskjellene i deres egen situasjon.

Se hva det koster Ta kontakt →

Drøft en relevant tilnærming

Be oss gjøre relevant erfaring til en tilnærming dere kan vurdere.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://software.engineer.company/nb/portfolio/